Salesforce.Platform-App-Builder-JPN.v2026-08-18.q110

試験コード： Platform-App-Builder-JPN
試験名称： Salesforce Certified Platform App Builder (Platform-App-Builder日本語版)
認証ベンダー： Salesforce
無料問題の数： 110
バージョン： v2026-08-18
ページの閲覧量 :105
問題集の閲覧量 :1126
https://www.jpnshiken.com/shiken/Salesforce.Platform-App-Builder-JPN.v2026-08-18.q110.html

質問: **1**
Ursa Major Solar社のアプリ開発者は、Salesforceの次のメジャーバージョンにアップグレードされたサンドボックス環境で新しいカスタムアプリの開発に取り組んでおり、本番環境のインスタンスは依然として現在のSalesforceバージョンを使用しています。開発は完了し、変更セットをデプロイする準備が整いました。アプリ開発者は、デプロイを計画する際に何を考慮すべきでしょうか？
A. 本番環境がアップグレードされると、変更セットは自動的にデプロイされます。
B. バージョンが異なるため、デプロイはできません。
C. 変更セットのコンポーネントは、本番環境で次のバージョンにアップグレードされます。
D. 次のバージョンでのみ利用可能な機能がある場合、失敗します。
正解: **(**正解を表示します**)**
A production admin would not solve this by decoration. Change sets move supported metadata between connected Salesforce environments. The receiving org uses inbound change sets and deployment validation to confirm whether the package can be deployed safely. Choose D. It will fail if there is a feature only available in the next version. This is the option that best matches the Salesforce responsibility being tested under App Deployment.
A practical administrator would confirm the object context, the user action, and whether the result must be stored, calculated, displayed, automated, approved, or deployed. Here the selected answer is the one that acts at the correct point in that chain. It keeps the configuration declarative and avoids inventing a process around a problem that Salesforce already has a native feature to solve.
The remaining answers do not control the same Salesforce behavior. A (The change set will be automatically deployed when production is upgrade) would be fragile here because Deployment tooling moves or validates metadata; it does not create the runtime business behavior itself. B (The deployment is not possible due to different versions.) does not fit cleanly; Deployment tooling moves or validates metadata; it does not create the runtime business behavior itself. C (The change set components will be upgraded to the next

version in production.) is only a partial match because A Lightning component changes the interface; it does not automatically solve security, relationship, or data-quality requirements. The result is a cleaner app design because the behavior is handled by the feature Salesforce built for that purpose.

質問: **2**
Universal Containers (UC) の Account オブジェクトには、変化するビジネス要件に合わせて値が定期的に変更される選択リスト項目が複数あります。こうした頻繁な変更により、UC には非アクティブな選択リスト値が多数存在し、システムのパフォーマンスとユーザー エクスペリエンスに影響を与えています。この問題を軽減するために、アプリ ビルダーはどのような対策を講じることができますか？
A. ピックリスト設定で既存のピックリストの上限を設定します。
B. ピックリスト値セットでグローバル値を設定します。
C. ピックリスト設定で、非アクティブなピックリスト値の上限を削除します。
D. 選択リストのフィールドを、ビジネス要件を満たす別のフィールドタイプに変換します。
正解: **B (**コメントを発表する**)**
The platform gives a native tool for this situation. Use B. Set up Global Values in Picklist Value Sets. The decision turns on platform configuration. The Platform App Builder role is to choose the standard Salesforce capability that owns the behavior: data structure, access, automation, user experience, reporting, or deployment.
This is how the same requirement would be handled in a well-governed Salesforce org: identify the controlling object, relationship, field, page, automation, or environment first, and then configure the native capability for that control point. The selected answer satisfies the requirement without requiring users to perform extra interpretation or administrators to maintain an unnecessary custom workaround.
The other choices would leave a gap in the implementation. A (Establish upper bound on existing picklists in Picklist Settings.) misses the mark because That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option. C (Remove upper bound on inactive picklist values in Picklist Settings.) handles a different concern; That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option. D (Convert the picklist fields to a different field type that will still meet the business requirements.) is less defensible because That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option.
In a real org, the builder should include negative testing as well, proving that the rule does not fire or display outside the intended conditions.

質問: **3**
DreamHouse Realty は、顧客のガレージに設置されるリフトの数を追跡したいと考えています。カスタム リフト オブジェクトの [設置予定] カスタム チェックボックス フィールドをチェックし、リフトが販売された翌日に外部システムにアウトバウンド メッセージで通知する必要があります。このタスクを完了するには、どの自動化ツールを使用すればよいですか?
A. 承認プロセス
B. 検証ルール
C. フロー
D. 代入ルール
正解: **(正解を表示します)**
A good app builder does not chase the most familiar label. The best answer is C. Flow. Screen flows guide a user through prompts, collect input, branch through decisions, and update records as part of a controlled interaction.
The selected answer is stronger because it meets both the functional requirement and the maintainability requirement. In a Salesforce implementation, a solution that only works for one user, one page, or one data sample is not good enough. The correct feature has to behave consistently across the intended profiles, record types, apps, and devices that are part of the design.
The distractors are not equal substitutes. A (Approval process) handles a different concern; An approval process is for formal sign-off, not for every automation, display, import, or security requirement. B (Validation rule) is less defensible because A validation rule blocks saves; it does not create records, route approvals, display components, or grant access. D (Assignment rule) would be fragile here because it could produce an inconsistent result once different users, records, or apps are tested. That is the kind of solution a Salesforce admin can document, migrate, and troubleshoot without creating hidden technical debt.
For guided work, a screen flow can ask questions in sequence, branch based on answers, and write results back to one or more records.

質問: **4**
ビジネスユーザーは、レコードのステータスをすばやく編集し、レコードからカスタム期日フィールドを入力する方法を求めている。
Salesforceモバイルアプリでフィードを表示するにはどうすればよいでしょうか？
A. カスタムクイックアクセスリンク
B. カスタムアクション
C. カスタムボタン
D. カスタムURL数式フィールド
正解: **(正解を表示します)**

The right answer avoids a brittle workaround. The best answer is B. Custom action. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, action regions, activity surfaces, and utility bars.
The selected answer is stronger because it meets both the functional requirement and the maintainability requirement. In a Salesforce implementation, a solution that only works for one user, one page, or one data sample is not good enough. The correct feature has to behave consistently across the intended profiles, record types, apps, and devices that are part of the design.
None of the other options gives the same administrative control. A (Custom quick access link) does not fit cleanly; it solves a symptom of the problem rather than the Salesforce mechanism that owns the result. C (Custom button) misses the mark because it addresses a neighboring feature area rather than the exact behavior requested. D (Custom URL formula field) is not enough: A formula is calculated display logic and cannot always store static values or perform record updates. This is the stronger design because it respects security, data quality, user experience, and lifecycle management at the same time.
For Lightning pages, activation and assignment determine who sees the page. Component visibility then controls whether a specific component appears for a given condition.

質問**: 5**
本番環境の組織には、機密情報を含むカスタムオブジェクトが含まれています。データレコードを含み、機密オブジェクトをすべて除外し、毎週更新できるサンドボックスが必要です。アプリ開発者は、これらの要件を満たすためにどのような手順を踏むべきでしょうか？
A. Developer Pro サンドボックスを作成し、データローダーをスケジュールして、選択したオブジェクトデータを毎週ダウンロードします。
B. フルコピーサンドボックスを作成し、サンドボックステンプレートを使用します。
C. 部分コピーサンドボックスを作成し、サンドボックステンプレートを使用します。
D. 開発者サンドボックスを作成し、データローダーが選択したオブジェクトデータを毎週ダウンロードするようにスケジュールします。
正解**: (**正解を表示します**)**
The design should be easy for another admin to maintain. In this case, The proper configuration is C. Create a Partial Copy Sandbox and use a sandbox template. Sandbox selection balances data copy, refresh cadence, isolation, testing depth, and cost. Smaller sandboxes fit independent build work; larger sandboxes fit integration and production-like testing.
The stem gives enough detail to reject surface-level fixes. Salesforce should be configured so the behavior is enforced or rendered by the feature designed for it. When the

requirement touches App Deployment, the builder should avoid crossing into a different layer unless there is a real platform limitation. The selected answer follows that discipline and creates the least surprising result for admins and users.
The wrong answers confuse adjacent platform concepts. A (Create a Developer Pro Sandbox and schedule Data Loader to download selected object data weekly.) is only a partial match because Data Loader is an administrative data tool; it does not replace metadata configuration or user-interface design. B (Create a Full Copy Sandbox and use a sandbox template.) misses the mark because Deployment tooling moves or validates metadata; it does not create the runtime business behavior itself. D (Create a Developer Sandbox and schedule Data Loader to download selected object data weekly.) handles a different concern; Data Loader is an administrative data tool; it does not replace metadata configuration or user-interface design. This keeps the build declarative, traceable in Setup, and easier to support when the org changes later.

質問**: 6**
Universal Containers (UC) では、すべてのユーザーが各商談を 受注済み」とマークする前に、契約書が送信されたことを指定する必要があります。UC は、商談がクローズされたときに、契約書が送信された商談の数と、契約書が送信されていない商談の数を比較してレポートできるようにしたいと考えています。この要件を満たすために、アプリビルダーはどのフィールドタイプを設定すればよいでしょうか？
A. 選択リスト
B. テキスト
C. チェックボックス
D. テキストエリア
正解**: (**正解を表示します**)**
The safest reading of the scenario is practical, not theoretical. Use C. Checkbox. The decision turns on record sharing targets. Manual record sharing can grant access to individual users, public groups, and roles with subordinates. It is a record-level access mechanism, separate from field-level security and permission set assignment.
This is how the same requirement would be handled in a well-governed Salesforce org: identify the controlling object, relationship, field, page, automation, or environment first, and then configure the native capability for that control point. The selected answer satisfies the requirement without requiring users to perform extra interpretation or administrators to maintain an unnecessary custom workaround.
The alternatives are plausible only if the requirement is read too narrowly. A (Picklist) should be rejected because That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option. B (Text) would be fragile here because That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option. D (Text Area) is only a partial match

because That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option. A complete implementation should also check page assignment, field access, record ownership, and mobile behavior where those factors apply.

質問: **7**
DreamHouse Realty のアプリ開発者が、関連ルックアップを介して 2 つの異なるオブジェクトのデータを含むフィールドを持つカスタムオブジェクトを作成しました。この新しいカスタムオブジェクトで ｢あり｣または ｢なし｣のレポートを作成するには何が必要ですか？
A. レポートフィルター
B. 行レベルの数式
C. レポートバケットフィールド
D. カスタムレポートタイプ
正解: **(**正解を表示します**)**
Treat the requirement as something users will depend on daily. Custom report types define reportable object relationships, including whether the report includes only matching child records or parent records with and without children. The proper configuration is D. Custom Report Type. This is the option that best matches the Salesforce responsibility being tested under Data Modeling and Management.
A practical administrator would confirm the object context, the user action, and whether the result must be stored, calculated, displayed, automated, approved, or deployed. Here the selected answer is the one that acts at the correct point in that chain. It keeps the configuration declarative and avoids inventing a process around a problem that Salesforce already has a native feature to solve.
The rejected choices solve different problems. A (Report Filters) is less defensible because Reporting configuration helps analysis, but it does not enforce the requested record behavior. B (Row-Level Formula) should be rejected because A formula is calculated display logic and cannot always store static values or perform record updates. C (Report Bucket Field) would be fragile here because Reporting configuration helps analysis, but it does not enforce the requested record behavior. The configuration remains understandable for future admins because the decision is visible in metadata rather than buried in a workaround.

質問: **8**
アプリ開発者が、モバイルユーザー向けに、リード、連絡先、アカウントから 新規メールを送信｣するオプションを追加したいと考えています。これを実現するためにグローバルアクションを使用するメリットは何ですか？

A. グローバルアクションのレイアウトは、デフォルトのページレイアウトを自動的に複製します。
B. Salesforce Lightning コンポーネント ライブラリには、使用するために事前に構築された既存のグローバル アクションが格納されています。
C. グローバルアクションはレコード固有のものであり、その特定のオブジェクトを検索する際に利用できます。
D. グローバルアクションは、レコードの詳細ページ、フィード、Chatterグループなど、モバイルでアクションが利用可能な場所ならどこでもアクセスできます。
正解: **(**正解を表示します**)**
The correct answer is the one that enforces the intended result. Quick actions are compact, purpose-built entry points. They are especially effective in the mobile app because they reduce navigation, expose only the fields required for the task, and can prepopulate values. The correct response is therefore D. Global actions can be accessed anywhere actions are available in mobile including Record detail pages, feed, and Chatter groups.
The scenario is not asking for a feature list. It is asking which configuration produces the requested result with the fewest moving parts. A Salesforce app builder should favor a standard, visible Setup configuration whenever it fully meets the business need. That approach makes the build easier to validate in sandbox and easier to explain during release review.
The other options fall short for clear platform reasons. A (The global action ' s layout automatically clones the default page layout.) is not enough: Page layouts affect presentation; they do not grant the underlying object or field capability by themselves. B (Salesforce Lightning Component Library houses existing global actions prebuilt for use.) handles a different concern; A Lightning component changes the interface; it does not automatically solve security, relationship, or data-quality requirements. C (Global actions are record-specific and are available when searching that particular Object.) is less defensible because Actions improve user entry, but they are not a substitute for relationship modeling, reporting logic, or back-end automation. Before release, test the configuration with the affected profile and realistic records, not only with a system administrator account.

質問: **9**
ウルサ ・メジャー ・ソーラー(UMS)には、銀河ベンダーを追跡するためのカスタムオブジェクトがあります。このオブジェクトには、銀河ベンダーの所在地を示す4つのカスタムフィールド(ストリート、都市、惑星、銀河)があります。UMSの経営陣は、これらのフィールドを2行の1つの数式フィールドに結合したいと考えています。この要件を満たす数式はどれですか？
A. 通り r (都市 & " 、

B. ストリート & BR () & プラネットギャラクシー_) シティ_c & " 、" & プラネット_ca & ギャラクシー_0
C. 通り & (都市 & " , " & 惑星 & " & Galaxy_0}
D. 通り r & BR () & 都市 & " , " & 惑星 _ * & & 銀河_r
正解**: (**正解を表示します**)**
The correct choice is the one that will still work after release. The best answer is B. Street & BR () & Planet Galaxy_) City_c & " , " & Planet_ca & Galaxy_0. The decision turns on formula logic. Formula and validation expressions must respect Salesforce data types. Picklists require picklist-aware handling, dates require date functions or date arithmetic, and image or text formulas must return the correct type.
This is how the same requirement would be handled in a well-governed Salesforce org: identify the controlling object, relationship, field, page, automation, or environment first, and then configure the native capability for that control point. The selected answer satisfies the requirement without requiring users to perform extra interpretation or administrators to maintain an unnecessary custom workaround.
The distractors are not equal substitutes. A (Street r (City & " ,) handles a different concern; it handles only part of the requirement and leaves the controlling Salesforce behavior unresolved. C (Street & (City & " , " & Planet & " & Galaxy_0}) should be rejected because it works at the wrong scope for the object, field, page, automation, or deployment condition in the stem. D (Street r & BR () & City & " , " & Planet _ * & & Galaxy_r) would be fragile here because it could produce an inconsistent result once different users, records, or apps are tested. That is the kind of solution a Salesforce admin can document, migrate, and troubleshoot without creating hidden technical debt.

質問**: 10**
3つのオプションを選択してください。Cloud Kicksは、新しい機会承認プロセスを設定し、最初の提出に基づいてさまざまなアクション項目を実行したいと考えています。アプリビルダーは、承認プロセスでどの3つのアクションタイプを使用すべきでしょうか？
A. 呼び出し可能なフロー
B. 決定要素
C. メールアラート
D. タスク
E. 送信メッセージ
正解**: C,D,E (**コメントを発表する**)**
The answer comes from Salesforce's separation of responsibilities. Use C. Email Alert; D. Task; E. Outbound Message. The scenario is asking for approval governance, and the reason is straightforward: Approval processes are designed for formal review. They route a request, capture a decision, and can lock or update the record through submission and

final actions. The wording points to this control because the business result must be reliable when real users work with real records.
For the stated requirement - Choose 3 options. Cloud Kicks wants to set up a new opportunity approval process and execute various action items based on the initial submission. Which three action types should an app builder use in the approval. - the selected answer handles the behavior directly. It is not just a way to make the page look cleaner or to remind users what to do. It places the responsibility in the Salesforce feature that owns the outcome. That matters in an App Builder implementation because access, automation, relationships, page rendering, reporting, and deployment each have different enforcement points.
The alternatives are plausible only if the requirement is read too narrowly. A (Invocable Flow) should be rejected because Flow is broad, but it is not correct unless the requirement is specifically about guided or automated logic. B (Decision Element) would be fragile here because it could produce an inconsistent result once different users, records, or apps are tested. A complete implementation should also check page assignment, field access, record ownership, and mobile behavior where those factors apply.

質問**: 11**
Universal Containers では、商談に関連付けられたアカウントの請求状態の値を示すフィールドを商談に含める必要があるという要件があります。この値は静的である必要があります。商談が作成された後、この自動化動作を構成するための推奨される解決策は何ですか？
A. フロー
B. 頂点
C. ロールアップ集計フィールド
D. 数式フィールド
正解**: (**正解を表示します**)**
The useful clue is the business action being requested. The best answer is A. Flow. The selected option is not chosen because it sounds familiar; it is chosen because it gives Salesforce the right instruction at runtime.
Screen flows guide a user through prompts, collect input, branch through decisions, and update records as part of a controlled interaction.
The scenario describes a business outcome, not merely an administrative preference. If the feature is configured correctly, users get the intended result without depending on memory, spreadsheets, or manual policing. That is the professional standard for Platform App Builder work: place the rule in metadata, test it under the correct permissions, and keep the design supportable.
None of the other options gives the same administrative control. B (Apex) is only a partial match because Apex may solve advanced gaps, but it is not the first choice when a native declarative feature fully handles the requirement. C (Roll-up summary field) misses the

mark because it addresses a neighboring feature area rather than the exact behavior requested. D (Formula field) is not enough: A formula is calculated display logic and cannot always store static values or perform record updates. This is the stronger design because it respects security, data quality, user experience, and lifecycle management at the same time.

質問**: 12**

Cloud Kicksの営業マネージャーは、営業担当者が収集する情報を標準化したいと考えています。営業担当者は、推奨事項、営業戦略、そして商談レコードの各営業プロセス段階で入力する必要のある主要項目を把握したいと考えています。アプリ開発者は、この機能を提供するためにどの機能を使用すべきでしょうか？

A. パス

B. その他の赤

C. 承認プロセス

D. グローバルアクション

正解**: (**正解を表示します**)**

The right answer avoids a brittle workaround. The best answer is A. Path. The selected option is not chosen because it sounds familiar; it is chosen because it gives Salesforce the right instruction at runtime. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars.

The scenario describes a business outcome, not merely an administrative preference. If the feature is configured correctly, users get the intended result without depending on memory, spreadsheets, or manual policing. That is the professional standard for Platform App Builder work: place the rule in metadata, test it under the correct permissions, and keep the design supportable.

None of the other options gives the same administrative control. B (Other Red) is only a partial match because it does not provide the same direct administrative control as the selected option. C (Approval Process) misses the mark because An approval process is for formal sign-off, not for every automation, display, import, or security requirement. D (Global Action) is not enough: Actions improve user entry, but they are not a substitute for relationship modeling, reporting logic, or back-end automation. This is the stronger design because it respects security, data quality, user experience, and lifecycle management at the same time.

質問**: 13**

Cloud Kicks は、各シューズカタログごとに販売されたシューズサブスクリプションの数を追跡したいと考えています。Subscription オブジェクトと shoe_c オブジェクトの間にはマ

スター/ディテール関係が存在します。アプリビルダーはどのタイプのフィールドを作成する必要がありますか？
A. ロールアップ集計フィールド
B. ルックアップフィールド
C. マスター/ディテールフィールド
D. 合計フィールド
正解**: A (**コメントを発表する**)**
A production admin would not solve this by decoration. Choose A. Roll-up summary field. The requirement in the stem is: Cloud Kicks wants to start tracking how many shoe subscriptions have been Sold for each shoe catalog. A master-detail relationship exists between the Subscription And the shoe_c objects. Which type of field should an. The selected option addresses that requirement at the point where Salesforce actually evaluates the behavior.
The underlying concept is roll-up summary design. Roll-up summary fields calculate values from child detail records and store the result on the master. The relationship type and field type decide whether the calculation is available declaratively. That is why this answer is more defensible than a workaround. It can be reviewed in Setup, included in the correct deployment path, and tested by impersonating or logging in as a representative user where appropriate.
The remaining answers do not control the same Salesforce behavior. B (Lookup field) does not fit cleanly; A lookup relationship is more loosely coupled and does not provide every master-detail behavior, especially native roll-up behavior. C (Master-detail field) is only a partial match because Master-detail is powerful, but it is wrong when the records need independent ownership or when the child-parent direction is reversed. D (Sum field) misses the mark because it addresses a neighboring feature area rather than the exact behavior requested. The result is a cleaner app design because the behavior is handled by the feature Salesforce built for that purpose.

質問**: 14**
AW Computingには、サービスプラン用のカスタムオブジェクトがあります。サービスプランは、必ず1つの連絡先にのみ関連付ける必要があります。サポートマネージャーは、誤った連絡先が関連付けられている場合、担当者が連絡先を変更できないことに気づきました。アプリビルダーは、ユーザーがフィールドへの適切なアクセス権を持っていること、およびサービスプランに関連付けられた検証がないことを既に確認済みです。この問題の原因は何でしょうか？
A. [読み取り専用] ラジオボタンが選択されています。マスターレコードへの読み取りアクセス権限を持つユーザーが、関連する詳細レコードを作成、編集、または削除できるようにします。

B. [親子関係の変更を許可する] チェックボックス(子レコードは作成後に他の親レコードに親子関係を変更できる)がオフになっています。
C. [読み取り/書き込み] ラジオボタンが選択されています。これにより、マスターレコードへの読み取り/書き込みアクセス権を持つユーザーが、関連する詳細レコードを作成、編集、または削除できるようになります。
D. [親子関係の変更を許可する] チェックボックスがオンになっています。子レコードは作成後に他の親レコードに親子関係を変更できるようになります。
正解**: (**正解を表示します**)**
The implementation choice has to match the object context. Use B. The Allow reparenting checkbox, Child records can be reparented to other parent records after they are created, is uncheckedd. The selected option is not chosen because it sounds familiar; it is chosen because it gives Salesforce the right instruction at runtime.
Lookup relationships connect records while allowing looser ownership and lifecycle control than master- detail. They are appropriate when records should remain more independent or when the relationship is optional.
The scenario describes a business outcome, not merely an administrative preference. If the feature is configured correctly, users get the intended result without depending on memory, spreadsheets, or manual policing. That is the professional standard for Platform App Builder work: place the rule in metadata, test it under the correct permissions, and keep the design supportable.
The other choices would leave a gap in the implementation. A (The Read Only radio button, Allows users with at least Read access to the Master record to create, edit, or delete related Detail records, is selecte) misses the mark because Master-detail is powerful, but it is wrong when the records need independent ownership or when the child-parent direction is reversed. C (The Read/Write radio button, Allows users with at least Read
/Write access to the Master record to create, edit, or delete related Detail records, is selecte) handles a different concern; Master-detail is powerful, but it is wrong when the records need independent ownership or when the child-parent direction is reversed. D (The Allow reparenting checkbox, Child records can be reparented to other parent records after they are created, is checked) is less defensible because That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option.
In a real org, the builder should include negative testing as well, proving that the rule does not fire or display outside the intended conditions.

質問**: 15**

Cloud Kicksの営業担当者が、Salesforce内で他のユーザーが自分の連絡先を表示できないと訴えています。アプリ開発者は、この問題をどのような3つの方法でトラブルシューティングできますか？3つ選択してください。
A. 連絡先レコードを確認し、アカウントにリンクされていることを確認してください。
B. 新しい連絡先を作成し、ユーザーに再度試してもらいます。
C. 問題が発生しているユーザーが連絡先オブジェクトにアクセスできることを確認します。
D. ユーザーがすべてのレコードにアクセスできるように、アカウント共有ルールを作成します。
E. 組織全体の既定の共有設定でアカウントへのアクセスが許可されているかどうかを確認します。
正解**: (**正解を表示します**)**
This scenario should be solved with standard metadata. Use A. Review the Contact record and ensure it is linked to an Account.; C. Verify the users with the issue have access to the Contact object.; E. Confirm whether Default Organization-Wide Sharing Settings provide access to the Account. Manual record sharing can grant access to individual users, public groups, and roles with subordinates. It is a record-level access mechanism, separate from field-level security and permission set assignment.
The selected answer is stronger because it meets both the functional requirement and the maintainability requirement. In a Salesforce implementation, a solution that only works for one user, one page, or one data sample is not good enough. The correct feature has to behave consistently across the intended profiles, record types, apps, and devices that are part of the design.
The alternatives are plausible only if the requirement is read too narrowly. B (Create a new Contact and have the users try again.) would be fragile here because it could produce an inconsistent result once different users, records, or apps are tested. D (Create an Account Sharing Rule to give the users access to all records.) is only a partial match because Sharing controls record visibility; it does not change field type, page composition, or automation timing. A complete implementation should also check page assignment, field access, record ownership, and mobile behavior where those factors apply.

質問**: 16**
Ursa Major Solarは、標準のContactオブジェクトとカスタムのSolar Projectオブジェクト間の関係を構築したいと考えています。Contactは複数のSolar Projectオブジェクトに関連付けられる可能性があり、Solar Projectには複数のContractが関連付けられる可能性があります。アプリ開発者は、データモデルをどのように構成すべきでしょうか？
A. Contact の 1 つのルックアップ関係と Solar Project のルックアップ関係
B. 新しいカスタムオブジェクトのルックアップ関係
C. 新しいカスタムオブジェクト上の2つのマスター/ディテール関係

D. Contact のマスターアバウト関係と Solar Project のマスターアバウト関係
正解: **(**正解を表示します**)**
This question is really testing ownership of the feature. Choose C. Two master-detail relationship on a new custom object. In Salesforce terms, this is a relationship modeling issue. Master-detail relationships create ownership, security, and lifecycle dependency between parent and child records. A junction object uses two master-detail relationships to represent a many-to-many model.
The selected configuration is the professional answer because it is native, declarative, and placed at the right scope. If the requirement later expands, another administrator can inspect the same metadata area and understand how the behavior is being produced. That is far better than a solution that depends on hidden assumptions or user training.
The other options fall short for clear platform reasons. A (One Lookup relationship on Contact and our Lookup relationship on Solar Project) is not enough: A lookup relationship is more loosely coupled and does not provide every master-detail behavior, especially native roll-up behavior. B (The Lookup relationship on a new custom object) handles a different concern; A lookup relationship is more loosely coupled and does not provide every master-detail behavior, especially native roll-up behavior. D (Our Master-About relationship on Contact and our Master-About relationship on Solar Project) should be rejected because Master-detail is powerful, but it is wrong when the records need independent ownership or when the child-parent direction is reversed. Before release, test the configuration with the affected profile and realistic records, not only with a system administrator account.

質問: **17**
Universal Containers (UC) は、5,000 件のリードレコードの複数のフィールドのデータを削除したいと考えています。UC は、削除する必要のあるレコード ID とフィールドを CSV ファイルにエクスポートしました。アプリ開発者は、これらの要件を満たすためにどの 2 つの手順を提案すべきでしょうか？ 2 つ選択してください。
A. 正しいレコードタイプを選択してください。

B. インポートウィザードを使用して、CSVファイルからリード情報を更新します。
C. データローダーを使用して、CSV ファイルでリードを更新します。
D. 設定で 「ヌル値を挿入」を選択します。
正解: **(**正解を表示します**)**
This is a governance decision before it is a configuration click. Salesforce import tools differ by data volume, matching logic, and operation. Data Import Wizard is guided and duplicate-aware for supported objects; Data Loader is stronger for larger updates, upserts, and clearing values. The proper configuration is C. Use Data Loader to update leads using the CSV file.; D. Select Insert Null Values in Settings. This is the option that best matches the Salesforce responsibility being tested under Data Modeling and Management.
A practical administrator would confirm the object context, the user action, and whether the result must be stored, calculated, displayed, automated, approved, or deployed. Here the selected answer is the one that acts at the correct point in that chain. It keeps the configuration declarative and avoids inventing a process around a problem that Salesforce already has a native feature to solve.
The rejected choices solve different problems. A (Select the correct record type.) is less defensible because it is not the most maintainable declarative control for this design. B (Use Import Wizard to update leads using the CSV file.) should be rejected because Import Wizard is useful for guided imports, but it is limited by object support, volume, and operation type. The configuration remains understandable for future admins because the decision is visible in metadata rather than buried in a workaround.

質問: **18**
取引が成立した後、Cloud Kicks (CK) はアカウント所有者に加えて、ユーザーをカスタマーサービスマネージャー (CSM) として割り当てたいと考えています。また、どの CSM がアカウントに割り当てられているかを簡単に追跡および報告できる新しいフィールドも必要です。このリクエストに対して、アプリ開発者はどのソリューションを使用すべきでしょうか？
A. ルックアップフィールド
B. テキストフィールド
C. 選択リストフィールド
D. 複数選択可能なピックリストフィールド
正解: **(**正解を表示します**)**
Read the stem as an implementation requirement. The Platform App Builder role is to choose the standard Salesforce capability that owns the behavior: data structure, access, automation, user experience, reporting, or deployment. The proper configuration is A. Lookup field. This is the option that best matches the Salesforce responsibility being tested under Data Modeling and Management.

A practical administrator would confirm the object context, the user action, and whether the result must be stored, calculated, displayed, automated, approved, or deployed. Here the selected answer is the one that acts at the correct point in that chain. It keeps the configuration declarative and avoids inventing a process around a problem that Salesforce already has a native feature to solve.
The wrong answers confuse adjacent platform concepts. B (Text field) misses the mark because That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option. C (Picklist field) is not enough: That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option. D (Multi-select picklist field) handles a different concern; That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option. This keeps the build declarative, traceable in Setup, and easier to support when the org changes later.

質問**: 19**
Universal Containersのアプリ開発者は、カスタムオブジェクトレコードページにChatterフィードを追加するよう依頼されました。アプリ開発者はどの方法を採用すべきでしょうか？
A. カスタム Chatter フィード コンポーネントを追加します。
B. 標準の関連リストコンポーネントを追加します
C. 標準のChatterフィードコンポーネントを追加します。
D. AppExchange から Chatter フィード コンポーネントを追加します。
正解**: (**正解を表示します**)**
The design should be easy for another admin to maintain. The proper configuration is C. Add the standard Chatter feed component. In Salesforce terms, this is a change-set deployment issue. Change sets move supported metadata between connected Salesforce environments. The receiving org uses inbound change sets and deployment validation to confirm whether the package can be deployed safely.
The selected configuration is the professional answer because it is native, declarative, and placed at the right scope. If the requirement later expands, another administrator can inspect the same metadata area and understand how the behavior is being produced. That is far better than a solution that depends on hidden assumptions or user training.
The wrong answers confuse adjacent platform concepts. A (Add a custom Chatter feed component.) is only a partial match because A Lightning component changes the interface; it does not automatically solve security, relationship, or data-quality requirements. B (Add the standard related list component) misses the mark because A Lightning component changes the interface; it does not automatically solve security, relationship, or data-quality requirements. D (Add the Chatter feed component from the AppExchange.) handles a different concern; A Lightning component changes the interface; it does not automatically

solve security, relationship, or data-quality requirements. This keeps the build declarative, traceable in Setup, and easier to support when the org changes later.

質問: **20**
Universal Containers (UC) は、sales_Organization_c と Pricing_Tier という 2 つのカスタム選択リストフィールドを使用します。Sales Organization_c の値が Canada である顧客に対して Pricing Tier_c が必須となるようにするには、アプリビルダーはどの検証ルールを使用すべきでしょうか？
A. AND (ISPICKVAL (Sales Organization_c, ' Canada '), ISBLANK (TEXT (Pricing_Tier)
B. IF (ISNULL(Sales_Organization_c= ' Canada ' , ISBLANK (TEXT (Pricing_Tier_c)), TRUI
C. ISPICKVAL (Sales Organization_c, ' Canada ') & ISNULL (Pricing_Tier_c)
D. OR (ISPICKVAL (Sales Organization_c, ' Canada '), ISBLANK (TEXT (Pricing_Tier_c));
正解: **(正解を表示します)**
The correct answer is the one that enforces the intended result. Choose A. AND (ISPICKVAL (Sales Organization_c, ' Canada '), ISBLANK (TEXT (Pricing_Tier. In Salesforce terms, this is a data validation issue. Validation rules prevent bad data from being saved. They are appropriate when the requirement is to enforce a condition at save time rather than merely warn, display, or report on the problem later.
The selected configuration is the professional answer because it is native, declarative, and placed at the right scope. If the requirement later expands, another administrator can inspect the same metadata area and understand how the behavior is being produced. That is far better than a solution that depends on hidden assumptions or user training.
The other options fall short for clear platform reasons. C (IF (ISNULL(Sales_Organization_c= ' Canada ' , ISBLANK (TEXT (Pricing_Tier_c)), TRUI) is less defensible because That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option. B (ISPICKVAL (Sales Organization_c, ' Canada ') & ISNULL (Pricing_Tier_c)) handles a different concern; it handles only part of the requirement and leaves the controlling Salesforce behavior unresolved. D (OR (ISPICKVAL (Sales Organization_c, ' Canada '), ISBLANK (TEXT (Pricing_Tier_c));) should be rejected because That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option.
Before release, test the configuration with the affected profile and realistic records, not only with a system administrator account.

質問: **21**
Universal Containersは、サービス内容にドローン配送を追加することになりました。開発者はコードの記述とテストを完了し、変更セットの準備は整っています。しかし、デプロイ

期間は開発者が休暇中の期間と重なります。アプリ開発者は、コードが本番環境に正常にデプロイされるようにするために、どのような対策を講じることができますか？
A. 受信変更セットからApexクラスを削除します。
B. 送信変更セットを検証します。
C. メタデータパッケージセットを使用します。
D. 受信変更セットを検証します。
正解: **(**正解を表示します**)**
This is where declarative precision matters. Use D. Validate the inbound change set. The selected option is not chosen because it sounds familiar; it is chosen because it gives Salesforce the right instruction at runtime.
Change sets move supported metadata between connected Salesforce environments. The receiving org uses inbound change sets and deployment validation to confirm whether the package can be deployed safely.
The scenario describes a business outcome, not merely an administrative preference. If the feature is configured correctly, users get the intended result without depending on memory, spreadsheets, or manual policing. That is the professional standard for Platform App Builder work: place the rule in metadata, test it under the correct permissions, and keep the design supportable.
The other choices would leave a gap in the implementation. A (Remove Apex classes from inbound change set.) misses the mark because Apex may solve advanced gaps, but it is not the first choice when a native declarative feature fully handles the requirement. B (Validate the outbound change set.) is not enough:
Deployment tooling moves or validates metadata; it does not create the runtime business behavior itself. C (Use a metadata package set.) handles a different concern; it handles only part of the requirement and leaves the controlling Salesforce behavior unresolved. In a real org, the builder should include negative testing as well, proving that the rule does not fire or display outside the intended conditions.

質問: **22**
アプリ開発者は、営業チームが行っている作業の一部を自動化する既存のプロセスビルダーを修正する必要があります。生産性を向上させるプロセスビルダーの3つの機能は何ですか？3つ選択してください。
A. 子レコードを作成します。
B. 追加のメッセージを開始します。
C. 関連レコードを更新します。
D. メールアラートを送信します。
E. 関連レコードを削除します。
正解: **(**正解を表示します**)**

Read the stem as an implementation requirement. Lookup relationships connect records while allowing looser ownership and lifecycle control than master-detail. They are appropriate when records should remain more independent or when the relationship is optional. The proper configuration is A. Create a child record. This is the option that best matches the Salesforce responsibility being tested under Business Logic and Process Automation.
A practical administrator would confirm the object context, the user action, and whether the result must be stored, calculated, displayed, automated, approved, or deployed. Here the selected answer is the one that acts at the correct point in that chain. It keeps the configuration declarative and avoids inventing a process around a problem that Salesforce already has a native feature to solve.
The wrong answers confuse adjacent platform concepts. B (Start an additional message.) misses the mark because it addresses a neighboring feature area rather than the exact behavior requested. C (Update a related record.) is not enough: it would require extra assumptions that the scenario does not support. D (Send an email alerts.) handles a different concern; it handles only part of the requirement and leaves the controlling Salesforce behavior unresolved. This keeps the build declarative, traceable in Setup, and easier to support when the org changes later.

質問**: 23**
Cloud Kicksチームに新しく加わったアプリ開発者が、データモデルの使い方を習得しようとしています。標準オブジェクトとカスタムオブジェクトがどのように関連しているかを確認したいと考えています。アプリ開発者は、これらの関係性を確認するためにどの機能を使用すればよいでしょうか？
A. オブジェクトマネージャ
B. フィールドとリレーションシップ
C. スキーマビルダー
D. Lightning App Builder
正解**: (**正解を表示します**)**
The useful clue is the business action being requested. The best answer is C. Schema Builder. The scenario is asking for Lightning page configuration, and the reason is straightforward: Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars. The wording points to this control because the business result must be reliable when real users work with real records.
For the stated requirement - A new app builder on the Cloud Kicks team is getting familiar with the data Model. They want to see how standard objects and custom objects relate. Which functionality should the app builder use to view these. - the selected answer handles the behavior directly. It is not just a way to make the page look cleaner or to remind users what to do. It places the responsibility in the Salesforce feature that owns the

outcome. That matters in an App Builder implementation because access, automation, relationships, page rendering, reporting, and deployment each have different enforcement points.
None of the other options gives the same administrative control. A (Object Manager) does not fit cleanly; it solves a symptom of the problem rather than the Salesforce mechanism that owns the result. B (Fields & Relationships) is only a partial match because it does not provide the same direct administrative control as the selected option. D (Lightning App Builder) is not enough: A Lightning component changes the interface; it does not automatically solve security, relationship, or data-quality requirements. This is the stronger design because it respects security, data quality, user experience, and lifecycle management at the same time.

質問**: 24**
DreamHouse Realtyは、商談が適格段階に進む前に、Expected_Close_Date cというフィールドに値が入力されていることを保証したいと考えています。アプリビルダーはこの要求にどのように対応すべきでしょうか？
A. ページレイアウト
B. 検証ルール
C. アクティビティ履歴
D. レコードタイプ
正解**: (**正解を表示します**)**
Think in terms of supportability first. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars. Choose B). Validation Rule. This is the option that best matches the Salesforce responsibility being tested under User Interface.
A practical administrator would confirm the object context, the user action, and whether the result must be stored, calculated, displayed, automated, approved, or deployed. Here the selected answer is the one that acts at the correct point in that chain. It keeps the configuration declarative and avoids inventing a process around a problem that Salesforce already has a native feature to solve.
The remaining answers do not control the same Salesforce behavior. A (Page Layout) would be fragile here because Page layouts affect presentation; they do not grant the underlying object or field capability by themselves. C (Activity History) is only a partial match because it does not provide the same direct administrative control as the selected option. D (Record Type) misses the mark because it addresses a neighboring feature area rather than the exact behavior requested. The result is a cleaner app design because the behavior is handled by the feature Salesforce built for that purpose.
For Lightning pages, activation and assignment determine who sees the page. Component visibility then controls whether a specific component appears for a given condition.

質問: **25**
Cloud Kicksは、アプリ開発者に対し、Deduplicationを使用してDeliveryカスタムオブジェクトに25,000件のレコードを挿入するよう依頼しました。どのツールを使用すべきでしょうか？
A. インポートウィザード
B. データローダー
C. Lightning オブジェクトクリエーター
D. スキーマビルダー
正解: **A (**コメントを発表する**)**
The important question is which feature owns the result. The proper configuration is A. Import Wizard. The requirement in the stem is: Cloud Kicks asked the app builder to insert a list of 25,000 records using Deduplication for the Delivery custom object. Which tool should be used? The selected option addresses that requirement at the point where Salesforce actually evaluates the behavior.
The underlying concept is data import and matching. Salesforce import tools differ by data volume, matching logic, and operation. Data Import Wizard is guided and duplicate-aware for supported objects; Data Loader is stronger for larger updates, upserts, and clearing values. That is why this answer is more defensible than a workaround. It can be reviewed in Setup, included in the correct deployment path, and tested by impersonating or logging in as a representative user where appropriate.
The rejected choices solve different problems. B (Data Loader) should be rejected because Data Loader is an administrative data tool; it does not replace metadata configuration or user-interface design. C (Lightning Object Creator) would be fragile here because A Lightning component changes the interface; it does not automatically solve security, relationship, or data-quality requirements. D (Schema Builder) does not fit cleanly; it solves a symptom of the problem rather than the Salesforce mechanism that owns the result. The configuration remains understandable for future admins because the decision is visible in metadata rather than buried in a workaround.

質問: **26**
Universal Containersには、100を超えるフィールドを保持するカスタムオブジェクトがあります。アプリビルダーは、これらのフィールドをLightningページの個別のタブに分割したいと考えています。この要件を満たすのに最も適したLightningコンポーネントはどれですか？
A. ハイライトパネル
B. レコードの詳細
C. フィールドセクション
D. アコーディオン

正解: **D** (コメントを発表する)
Think in terms of supportability first. Choose D. Accordion. In Salesforce terms, this is a Lightning page configuration issue. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, action regions, activity surfaces, and utility bars.
The selected configuration is the professional answer because it is native, declarative, and placed at the right scope. If the requirement later expands, another administrator can inspect the same metadata area and understand how the behavior is being produced. That is far better than a solution that depends on hidden assumptions or user training.
The remaining answers do not control the same Salesforce behavior. A (Highlights panel) would be fragile here because it could produce an inconsistent result once different users, records, or apps are tested. B (Record detail) does not fit cleanly; Master-detail is powerful, but it is wrong when the records need independent ownership or when the child-parent direction is reversed. C (Field section) is only a partial match because it does not provide the same direct administrative control as the selected option. The result is a cleaner app design because the behavior is handled by the feature Salesforce built for that purpose.
For Lightning pages, activation and assignment determine who sees the page. Component visibility then controls whether a specific component appears for a given condition.

質問: **27**
Universal Containers社は初めてSalesforceを導入します。経営陣は、Salesforceモバイルアプリで営業チームとマーケティングチームに異なるナビゲーションメニューを設定したいと考えています。アプリ開発者がこの要件を満たすために利用できるオプションはどれですか？
A. 営業およびマーケティング用の公開グループを作成し、各グループ用にモバイルナビゲーションメニューを作成します。
B. 営業およびマーケティングアプリを作成し、それぞれのアプリに該当するプロファイルを割り当てます。
C. 営業とマーケティングの役割を作成し、それぞれの役割にカスタムホームページレイアウトを割り当てます。
D. 営業プロファイルとマーケティングプロファイルの両方に対してモバイルナビゲーションメニューを作成します。
正解: (正解を表示します)
This question is really testing ownership of the feature. Quick actions are compact, purpose-built entry points.
They are especially effective in the mobile app because they reduce navigation, expose only the fields required for the task, and can prepopulate values. The correct response is therefore B. Create sales and marketing apps and assign the respective profiles to each app.

The scenario is not asking for a feature list. It is asking which configuration produces the requested result with the fewest moving parts. A Salesforce app builder should favor a standard, visible Setup configuration whenever it fully meets the business need. That approach makes the build easier to validate in sandbox and easier to explain during release review.
The other options fall short for clear platform reasons. A (Create public groups for sales and marketing and create mobile navigation menus for each group.) is not enough: it would require extra assumptions that the scenario does not support. C (Create roles for sales and marketing and assign a custom homepage layout for each role.) is less defensible because Page layouts affect presentation; they do not grant the underlying object or field capability by themselves. D (Create mobile navigation menus for both the sales and marketing profiles.) should be rejected because it works at the wrong scope for the object, field, page, automation, or deployment condition in the stem. Before release, test the configuration with the affected profile and realistic records, not only with a system administrator account.

質問**: 28**
Ursa Major Solar (UMS) は、顧客からの苦情を追跡するために Cases オブジェクトを使用し、太陽光パネルの既知の問題を表すために Issue__c オブジェクトを使用し、既知の問題を顧客の苦情に関連付けるために Case_Issue__c ジャンクション オブジェクトを使用します。UMS は定期的に監査を実施し、監査ユーザーは case_Issue__c レコードを表示する必要があります。UMS ユーザーが Case Issue_c レコードにアクセスできるようにするには、どのアクセス レベルを設定する必要がありますか？
A. Case および Issue_c への読み取り専用アクセス
B. Case および Case Issue_e への読み取り専用アクセス
C. Case Issue_c に対する読み取り専用アクセス
D. Issue__c および Case Issue_c に対する読み取り専用アクセス
正解**: (**正解を表示します**)**
Start at the layer that controls the outcome. The best answer is A. Read-Only access on Case and Issue_c.
Manual record sharing can grant access to individual users, public groups, and roles with subordinates. It is a record-level access mechanism, separate from field-level security and permission set assignment.
The selected answer is stronger because it meets both the functional requirement and the maintainability requirement. In a Salesforce implementation, a solution that only works for one user, one page, or one data sample is not good enough. The correct feature has to behave consistently across the intended profiles, record types, apps, and devices that are part of the design.
The distractors are not equal substitutes. B (Read-Only access on Case and Case Issue_e) is less defensible because it is not the most maintainable declarative control for

this design. C (Read-Only access on Case Issue_c) should be rejected because it works at the wrong scope for the object, field, page, automation, or deployment condition in the stem. D (Read-Only access on Issue__c and Case Issue_c) would be fragile here because it could produce an inconsistent result once different users, records, or apps are tested. That is the kind of solution a Salesforce admin can document, migrate, and troubleshoot without creating hidden technical debt.

質問: **29**
アプリ開発者は、既存のアプリのどこで編集して、ユーティリティバーにコンポーネントを追加できますか？
A. Lightning App Builder
B. 雷記録ページ
C. アプリマネージャー
D. アプリメニュー
正解: **C (**コメントを発表する**)**
This is a governance decision before it is a configuration click. The proper configuration is C. App Manager.
The requirement in the stem is: Where can an app builder edit an existing app to add Components to the utility bar? The selected option addresses that requirement at the point where Salesforce actually evaluates the behavior.
The underlying concept is Lightning page configuration. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars. That is why this answer is more defensible than a workaround. It can be reviewed in Setup, included in the correct deployment path, and tested by impersonating or logging in as a representative user where appropriate.
The rejected choices solve different problems. A (Lightning App Builder) is less defensible because A Lightning component changes the interface; it does not automatically solve security, relationship, or data- quality requirements. B (Lightning Record Page) should be rejected because A Lightning component changes the interface; it does not automatically solve security, relationship, or data-quality requirements. D (App Menu) does not fit cleanly; it solves a symptom of the problem rather than the Salesforce mechanism that owns the result. The configuration remains understandable for future admins because the decision is visible in metadata rather than buried in a workaround.

質問: **30**
アプリ開発者が、関連する取引先責任者のデータを含めるための数式項目を取引先に作成しようとしていますが、数式エディタでその関係性を見つけることができません。この問題の原因として考えられる数式の制限事項は何でしょうか？

A. コントロールオブジェクトとアカウントオブジェクト: 0.00 または 1 はマスター/ディテール関係を持っています。
B. 数式内に3000文字のデータを保存します。
C. アカウントオブジェクトに到達した数式フィールド。
D. 子レコードを参照できません。
正解: **(**正解を表示します**)**
Start at the layer that controls the outcome. The best answer is C. Formula field that reached on the Account object. The selected option is not chosen because it sounds familiar; it is chosen because it gives Salesforce the right instruction at runtime. Formula and validation expressions must respect Salesforce data types.
Picklists require picklist-aware handling, dates require date functions or date arithmetic, and image or text formulas must return the correct type.
The scenario describes a business outcome, not merely an administrative preference. If the feature is configured correctly, users get the intended result without depending on memory, spreadsheets, or manual policing. That is the professional standard for Platform App Builder work: place the rule in metadata, test it under the correct permissions, and keep the design supportable.
The distractors are not equal substitutes. A (Control and Account objects: 0.00 or 1 have a Master-Detail Relationship.) handles a different concern; Master-detail is powerful, but it is wrong when the records need independent ownership or when the child-parent direction is reversed. B (Store Data: 3000 characters in the formula.) is less defensible because A formula is calculated display logic and cannot always store static values or perform record updates. D (unable to reference the child records.) would be fragile here because it could produce an inconsistent result once different users, records, or apps are tested. That is the kind of solution a Salesforce admin can document, migrate, and troubleshoot without creating hidden technical debt.

質問: **31**
ユニバーサル ·コンテナーズ社は、営業担当者が商談を削除する前に、上司の許可を得ることを求めている。
これらの要件を満たすために何が使えるでしょうか？
A. 2段階の承認プロセス
B. スケジュールされたパスによる承認プロセスとフロー
C. スケジュールによってトリガーされるフローで、承認のために送信するアクションが含まれています。
D. レコードトリガーフローによる承認プロセス
正解: **(**正解を表示します**)**
This is where declarative precision matters. Use D. Approval process with a record triggered flow. The decision turns on Lightning page configuration. Lightning App Builder

controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars.
This is how the same requirement would be handled in a well-governed Salesforce org: identify the controlling object, relationship, field, page, automation, or environment first, and then configure the native capability for that control point. The selected answer satisfies the requirement without requiring users to perform extra interpretation or administrators to maintain an unnecessary custom workaround.
The other choices would leave a gap in the implementation. A (Two-step approval process) misses the mark because An approval process is for formal sign-off, not for every automation, display, import, or security requirement. B (Approval process and flow with a scheduled path) is not enough: Flow is broad, but it is not correct unless the requirement is specifically about guided or automated logic. C (Schedule-triggered flow with Submit for Approval action) handles a different concern; Flow is broad, but it is not correct unless the requirement is specifically about guided or automated logic. In a real org, the builder should include negative testing as well, proving that the rule does not fire or display outside the intended conditions.

質問: **32**
Cloud Kicksは最近、リリース管理戦略にアプリケーションライフサイクル管理プロセスを導入しました。バグ修正や簡単な変更を扱うカテゴリはどれですか？
A. ロール
B. メジャー
C. マイナー
D. パッチ
正解: **(**正解を表示します**)**
The key is to separate behavior from appearance. The best answer is D. Patch. The decision turns on change- set deployment. Change sets move supported metadata between connected Salesforce environments. The receiving org uses inbound change sets and deployment validation to confirm whether the package can be deployed safely.

This is how the same requirement would be handled in a well-governed Salesforce org: identify the controlling object, relationship, field, page, automation, or environment first, and then configure the native capability for that control point. The selected answer satisfies the requirement without requiring users to perform extra interpretation or administrators to maintain an unnecessary custom workaround.
None of the other options gives the same administrative control. A (Roll) does not fit cleanly; it solves a symptom of the problem rather than the Salesforce mechanism that owns the result. B (Major) is only a partial match because it does not provide the same direct administrative control as the selected option. C (Minor) misses the mark because it addresses a neighboring feature area rather than the exact behavior requested. This is the stronger design because it respects security, data quality, user experience, and lifecycle management at the same time.
For deployment work, related metadata matters. Fields, layouts, actions, permissions, tabs, flows, and dependent components may all need to travel together for the target org to behave as expected.

質問**: 33**
ドリームハウス ･リアルティ(DR)は、地方自治体と提携することで、補助金付き住宅事業に事業を拡大している。
DRはSales Cloudを使用しており、Opportunityオブジェクトでフィールド履歴追跡を有効にしています。情報要件の増加に伴い、アプリ開発チームはテキストエリア(ロング)フィールドをリッチテキストフィールドに変更し、最大1,000文字まで入力できるようにして、より詳細な説明を記述できるようにしています。チームはどのような2つの点を考慮すべきでしょうか？
2つの回答を選択してください
A. カスタムフィールドのタイプを変更すると、データが失われる可能性があります。
B. 監査証跡はREST API抽出を通じて利用可能です。
C. すべての長さのリッチテキストフィールドの値がレポートに完全に表示されます。
D. フィールド履歴追跡は、255文字以下の値の変更を記録します。
正解**: A,D (**コメントを発表する**)**
The platform gives a native tool for this situation. Use A. Data loss may occur when changing custom field types.; D. Field History Tracking records value changes of 255 characters or less. The scenario is asking for platform configuration, and the reason is straightforward: The Platform App Builder role is to choose the standard Salesforce capability that owns the behavior: data structure, access, automation, user experience, reporting, or deployment. The wording points to this control because the business result must be reliable when real users work with real records.
For the stated requirement - DreamHouse Realty (DR) is expanding into subsidized housing by partnering with local government entities. DR uses Sales Cloud and has

Enabled field history tracking on the Opportunity object. Due to increased. - the selected answer handles the behavior directly. It is not just a way to make the page look cleaner or to remind users what to do. It places the responsibility in the Salesforce feature that owns the outcome. That matters in an App Builder implementation because access, automation, relationships, page rendering, reporting, and deployment each have different enforcement points.
The other choices would leave a gap in the implementation. B (Audit Trail is available through REST API extracts.) is not enough: it would require extra assumptions that the scenario does not support. C (Rich text field values of all lengths are displayed fully in reports.) handles a different concern; Reporting configuration helps analysis, but it does not enforce the requested record behavior. In a real org, the builder should include negative testing as well, proving that the rule does not fire or display outside the intended conditions.

質問**: 34**
Universal Containers社は、アカウントタイプが 過去の顧客」に変更された際に、そのアカウントに直接関連する連絡先のステータスが 再マーケティング」に更新されるように、アプリ開発者に依頼しました。このタスクを実行するために、アプリ開発者はどの自動化機能を使用すべきでしょうか？
A. 検証ルール
B. Lightningコンポーネント
C. 画面の流れ
D. レコードトリガーフロー
正解**: (**正解を表示します**)**
Read the stem as an implementation requirement. The proper configuration is D. Record-triggered flow. The requirement in the stem is: Universal Containers asked the app builder to ensure when an Account type changes to ' Past-Customer ' the contacts directly related to That account get an updated status of ' Re-Market '
. Which automation should the app. The selected option addresses that requirement at the point where Salesforce actually evaluates the behavior.
The underlying concept is Lightning page configuration. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars. That is why this answer is more defensible than a workaround. It can be reviewed in Setup, included in the correct deployment path, and tested by impersonating or logging in as a representative user where appropriate.
The wrong answers confuse adjacent platform concepts. A (Validation rule) is only a partial match because A validation rule blocks saves; it does not create records, route approvals, display components, or grant access.

B (Lightning component) misses the mark because A Lightning component changes the interface; it does not automatically solve security, relationship, or data-quality requirements. C (Screen flow) is not enough: Flow is broad, but it is not correct unless the requirement is specifically about guided or automated logic. This keeps the build declarative, traceable in Setup, and easier to support when the org changes later.

質問: **35**
Cloud Kicks (CK) は Lightning Experience に移行し、グローバルな従業員全体で Chatter の利用を開始して、ペースの速い販売サイクルをサポートし始めました。CK は Chatter を気に入っていますが、コアチームメンバーからのフィードバックの収集、特に誰が対応可能かを把握することに苦労しています。CK はこの問題を解決するために、Chatter をどのように活用できるでしょうか？ 2 つの回答を選択してください。
A. ストリーム
B. トピック
C. 世論調査
D. 不在
正解: **A,D (**コメントを発表する**)**
A good app builder does not chase the most familiar label. The best answer is A. Streams; D. Out of Office.
The decision turns on Chatter collaboration. Chatter supports record-centered collaboration. Feed tracking surfaces selected field changes, while groups, streams, and post actions organize discussions and follow-up work.
This is how the same requirement would be handled in a well-governed Salesforce org: identify the controlling object, relationship, field, page, automation, or environment first, and then configure the native capability for that control point. The selected answer satisfies the requirement without requiring users to perform extra interpretation or administrators to maintain an unnecessary custom workaround.
The distractors are not equal substitutes. B (Topics) is less defensible because it is not the most maintainable declarative control for this design. C (Polls) should be rejected because it works at the wrong scope for the object, field, page, automation, or deployment condition in the stem. That is the kind of solution a Salesforce admin can document, migrate, and troubleshoot without creating hidden technical debt.
For collaboration, the builder should decide whether the need is record feed visibility, group conversation, field-change tracking, or a quick action in the publisher.
For collaboration, the builder should decide whether the need is record feed visibility, group conversation, field-change tracking, or a quick action in the publisher.

質問: **36**

Cloud Kicksの従業員が、上司の承認を得るために求人情報を提出しました。提出後に従業員が説明欄を編集しようとした場合、どうなりますか？
A. ユーザーは説明欄のみ編集できます。
B. ユーザーには 「レコードロック」通知が表示されます。
C. ユーザーは名前を編集できますが、説明を編集することはできません。
D. ユーザーは、レコードが自分のマネージャーの所有物になったことを確認できます。
正解: **(**正解を表示します**)**
A production admin would not solve this by decoration. The Platform App Builder role is to choose the standard Salesforce capability that owns the behavior: data structure, access, automation, user experience, reporting, or deployment. Choose B. User will be presented with a ' Record Lock ' notification. This is the option that best matches the Salesforce responsibility being tested under Business Logic and Process Automation.
A practical administrator would confirm the object context, the user action, and whether the result must be stored, calculated, displayed, automated, approved, or deployed. Here the selected answer is the one that acts at the correct point in that chain. It keeps the configuration declarative and avoids inventing a process around a problem that Salesforce already has a native feature to solve.
The remaining answers do not control the same Salesforce behavior. A (User will be able to edit the description field only.) would be fragile here because it could produce an inconsistent result once different users, records, or apps are tested. C (User will be able to edit the name, but unable to edit the description.) is only a partial match because it does not provide the same direct administrative control as the selected option.
D (User will see the record is now owned by their manager.) misses the mark because it addresses a neighboring feature area rather than the exact behavior requested. The result is a cleaner app design because the behavior is handled by the feature Salesforce built for that purpose.

質問: **37**
Northern Trail Outfitters社は、Salesforce組織のデイリーバックアップを開始したいと考えています。このタスクには、アプリ開発者はどのツールを推奨すべきでしょうか？
A. フルコピーサンドボックスを更新します
B. パッケージの追加/変更
C. データエクスポートサービス
D. サポートエキスパート
正解: **(**正解を表示します**)**
This is a governance decision before it is a configuration click. Change sets move supported metadata between connected Salesforce environments. The receiving org uses inbound change sets and deployment validation to confirm whether the package can be

deployed safely. The proper configuration is C. Data Export Service. This is the option that best matches the Salesforce responsibility being tested under App Deployment.
A practical administrator would confirm the object context, the user action, and whether the result must be stored, calculated, displayed, automated, approved, or deployed. Here the selected answer is the one that acts at the correct point in that chain. It keeps the configuration declarative and avoids inventing a process around a problem that Salesforce already has a native feature to solve.
The rejected choices solve different problems. A (Refresh full copy sandbox) is less defensible because Deployment tooling moves or validates metadata; it does not create the runtime business behavior itself. B (Add/Change package) should be rejected because it works at the wrong scope for the object, field, page, automation, or deployment condition in the stem. D (Support expert) does not fit cleanly; it solves a symptom of the problem rather than the Salesforce mechanism that owns the result. The configuration remains understandable for future admins because the decision is visible in metadata rather than buried in a workaround.

質問**: 38**
Cloud Kicksは、マネージャーメモ欄を長文テキストエリアから短縮するために、業務プロセス全体を再定義しています。その目的は、マネージャーがコメントをより簡潔に、255文字以内に収めるように促すことです。マネージャーメモ欄には、文字数制限をはるかに超える情報が既に入力されている場合が多くあります。
アプリビルダーが既存の長文テキストエリアフィールドをテキストエリアに変換しようとした場合、既存の情報はどうなりますか？
A. 既存の情報は最初の 255 文字に切り詰められます。
B. 既存の情報があるとエラーメッセージが表示されます。
C. 既存の情報は、255文字を超えていてもそのまま残ります。
D. 現場に存在する既存の情報は完全に失われます。
正解**: (**正解を表示します**)**
This is where declarative precision matters. Field types define what can be stored and how the platform handles the value. Rich text is needed when users must include formatted content, links, or images inside a field. Use A. Preexisting information will truncate to the first 255 characters.
A clean Platform App Builder answer should satisfy the complete scenario, not merely one phrase in it. This item requires the app builder to choose the configuration that produces the requested behavior in normal Salesforce use. The selected answer fits that requirement because it works with the platform model instead of fighting it.
The other choices would leave a gap in the implementation. B (Preexisting information will cause an error message to pop up.) is not enough: it would require extra assumptions that the scenario does not support. C (Preexisting information will remain even if it was over

255 characters.) handles a different concern; it handles only part of the requirement and leaves the controlling Salesforce behavior unresolved. D (Preexisting information in the field will be completely lost.) is less defensible because it is not the most maintainable declarative control for this design. In a real org, the builder should include negative testing as well, proving that the rule does not fire or display outside the intended conditions.

質問**: 39**
Universal Containers では、アカウント オブジェクトの組織全体のデフォルト設定が 「非公開」になっています。マーケティング チームはアカウントを所有していますが、営業チームのアカウントも表示できるようにする必要があります。営業チームとマーケティング チームは、役割階層のまったく異なるブランチに属しています。マーケティング チームが営業所有のアカウントを表示できるようにするには、どの機能を使用すればよいでしょうか？
A. 共有ルール
B. 公開グループ
C. エイリアス
D. 割り当てルール
正解**: (**正解を表示します**)**
The answer is controlled by runtime behavior, not by naming similarity. The Platform App Builder role is to choose the standard Salesforce capability that owns the behavior: data structure, access, automation, user experience, reporting, or deployment. The correct response is therefore A. Sharing rules.
The scenario is not asking for a feature list. It is asking which configuration produces the requested result with the fewest moving parts. A Salesforce app builder should favor a standard, visible Setup configuration whenever it fully meets the business need. That approach makes the build easier to validate in sandbox and easier to explain during release review.
The other options fall short for clear platform reasons. B (Public groups) handles a different concern; it handles only part of the requirement and leaves the controlling Salesforce behavior unresolved. C (Aliases) is less defensible because it is not the most maintainable declarative control for this design. D (Assignment rules) should be rejected because it works at the wrong scope for the object, field, page, automation, or deployment condition in the stem. Before release, test the configuration with the affected profile and realistic records, not only with a system administrator account.
For general platform design, the strongest answer is the native feature that can be enforced, audited, and maintained through Salesforce metadata.

質問**: 40**

Cloud Kicks は、顧客サポートの利用状況が高、中、低のいずれであるかを示すために、アカウントに関連する未解決のケース数と解決済みのケース数を集計したいと考えています。NUM_open_Cases_c と NUM_Closed Cases_c という 2 つの数値フィールドが作成されました。これらのビジネス要件を満たす 2 つのソリューションはどれですか？ 2 つの回答を選択してください。

A. 検証ルール

B. 頂点

C. AppExchange

D. 承認プロセス

正解: **(**正解を表示します**)**

This is where declarative precision matters. Use B. Apex; C. AppExchange. A Lightning component must be exposed for the surface where it is used. A component built for one page type or action surface does not automatically become available everywhere in Lightning Experience.

The selected answer is stronger because it meets both the functional requirement and the maintainability requirement. In a Salesforce implementation, a solution that only works for one user, one page, or one data sample is not good enough. The correct feature has to behave consistently across the intended profiles, record types, apps, and devices that are part of the design.

The other choices would leave a gap in the implementation. A (Validation Rule) misses the mark because A validation rule blocks saves; it does not create records, route approvals, display components, or grant access.

D (Approval Process) is less defensible because An approval process is for formal sign-off, not for every automation, display, import, or security requirement. In a real org, the builder should include negative testing as well, proving that the rule does not fire or display outside the intended conditions.

For AppExchange components, the builder should verify the component target, activate the relevant Lightning page, and test the page in the same form factor used by the target users.

質問: **41**

Universal Containersは、プロファイル設定や権限セットを含めずに、変更セットを介してカスタムタブを本番環境にデプロイしました。カスタムタブの表示設定は何ですか？

A. カスタムタブはすべてのユーザーに対して非表示になっています。

B. カスタムタブは、すべてのユーザーに対してデフォルトで無効になっています。

C. カスタムタブはすべてのユーザーに対してデフォルトで有効になっています。

D. カスタムタブはデプロイされません

正解: **(**正解を表示します**)**

The key is to separate behavior from appearance. Change sets move supported metadata between connected Salesforce environments. The receiving org uses inbound change sets and deployment validation to confirm whether the package can be deployed safely. The best answer is A. Custom tabs are hidden for all users.
A clean Platform App Builder answer should satisfy the complete scenario, not merely one phrase in it. This item requires the app builder to choose the configuration that produces the requested behavior in normal Salesforce use. The selected answer fits that requirement because it works with the platform model instead of fighting it.
None of the other options gives the same administrative control. B (Custom tabs are default off for all users.) is only a partial match because it does not provide the same direct administrative control as the selected option. C (Custom tabs are default on for all users.) misses the mark because it addresses a neighboring feature area rather than the exact behavior requested. D (Custom tabs are NOT deploye) is not enough:
Deployment tooling moves or validates metadata; it does not create the runtime business behavior itself. This is the stronger design because it respects security, data quality, user experience, and lifecycle management at the same time.

質問**: 42**
アプリ開発者は、変更セットを使用してサンドボックスから本番環境に新しいアプリをデプロイする準備をしています。このプロセス中にアプリ開発者が留意すべき2つの事項は何ですか？ 2つ選択してください。
A. デプロイメントエラーが発生した場合、トランザクションは元に戻されます。
B. 受信変更セットを受け取る際は、ユーザーは本番環境からログアウトする必要があります。
C. Salesforce Connect は環境間のリンクを自動的に確立します。
D. 変更セットにはすべてのコンポーネントが含まれていないため、一部の変更を手動で実行する必要がある場合があります。
正解**: A,D (**コメントを発表する**)**
Think in terms of supportability first. In this case, Choose A. Transactions will revert if the deployment errors.; D. Change sets do not include all components and may have to perform some changes Manually.
Change sets move supported metadata between connected Salesforce environments. The receiving org uses inbound change sets and deployment validation to confirm whether the package can be deployed safely.
The stem gives enough detail to reject surface-level fixes. Salesforce should be configured so the behavior is enforced or rendered by the feature designed for it. When the requirement touches App Deployment, the builder should avoid crossing into a different layer unless there is a real platform limitation. The selected answer follows that discipline and creates the least surprising result for admins and users.

The remaining answers do not control the same Salesforce behavior. B (Users should be logged out of production when receiving inbound change sets.) does not fit cleanly; Deployment tooling moves or validates metadata; it does not create the runtime business behavior itself. C (Salesforce Connect automatically establishes a link between environments.) is only a partial match because it does not provide the same direct administrative control as the selected option. The result is a cleaner app design because the behavior is handled by the feature Salesforce built for that purpose.

質問: **43**
Universal Container社は、顧客が一般公開されているウェブサイトからケースを開設できるようにしたいと考えています。ウェブサイト訪問者がこの機能を利用できるようにするために、アプリ開発者は何を使用すべきでしょうか？
A. Web to case
B. 送信メッセージ
C. 画面の流れ
D. メールからケースへ
正解: **A (**コメントを発表する**)**
This question is really testing ownership of the feature. In this case, Choose A. Web-to-case. The Platform App Builder role is to choose the standard Salesforce capability that owns the behavior: data structure, access, automation, user experience, reporting, or deployment.
The stem gives enough detail to reject surface-level fixes. Salesforce should be configured so the behavior is enforced or rendered by the feature designed for it. When the requirement touches Business Logic and Process Automation, the builder should avoid crossing into a different layer unless there is a real platform limitation. The selected answer follows that discipline and creates the least surprising result for admins and users.
The other options fall short for clear platform reasons. B (Outbound message) handles a different concern; it handles only part of the requirement and leaves the controlling Salesforce behavior unresolved. C (Screen flow) is less defensible because Flow is broad, but it is not correct unless the requirement is specifically about guided or automated logic. D (Email-to-case) should be rejected because it works at the wrong scope for the object, field, page, automation, or deployment condition in the stem. Before release, test the configuration with the affected profile and realistic records, not only with a system administrator account.

質問: **44**
Ursa Major Solarでは、惑星オブジェクトに 惑星の詳細」という新しいフィールドを追加し、ユーザーが画像やリンクを含む詳細な説明を記述できるようにする必要があります。

この要件を満たすために、アプリ開発者はどのフィールドタイプを使用すべきでしょうか？
A. リッチテキストエリア
B. Lightning Web コンポーネント
C. URL
D. 長文テキストエリア
正解: **(**正解を表示します**)**
The right answer avoids a brittle workaround. The best answer is A. Rich text area. The decision turns on Lightning page configuration. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars.
This is how the same requirement would be handled in a well-governed Salesforce org: identify the controlling object, relationship, field, page, automation, or environment first, and then configure the native capability for that control point. The selected answer satisfies the requirement without requiring users to perform extra interpretation or administrators to maintain an unnecessary custom workaround.
None of the other options gives the same administrative control. B (Lightning web component) is only a partial match because A Lightning component changes the interface; it does not automatically solve security, relationship, or data-quality requirements. C (URL) misses the mark because it addresses a neighboring feature area rather than the exact behavior requested. D (Long text area) is not enough: That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option. This is the stronger design because it respects security, data quality, user experience, and lifecycle management at the same time.

質問: **45**
Universal Containers社のアプリ開発者は、カスタムオブジェクトレコードページにChatterフィードを追加するよう依頼された。
アプリ開発者はどちらのアプローチを採用すべきでしょうか？
A. 標準関連の第一コンポーネントを追加します。
B. AppleChange から Chatter フィード コンポーネントを追加します。
C. 標準のChatterフィードコンポーネントを追加します。
D. カスタム Chatter フィード コンポーネントを追加します。
正解: **C (**コメントを発表する**)**
Read the stem as an implementation requirement. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars. The correct response is therefore C. Add the standard Chatter feed component.

The scenario is not asking for a feature list. It is asking which configuration produces the requested result with the fewest moving parts. A Salesforce app builder should favor a standard, visible Setup configuration whenever it fully meets the business need. That approach makes the build easier to validate in sandbox and easier to explain during release review.
The wrong answers confuse adjacent platform concepts. A (Add the standard related first component.) is only a partial match because A Lightning component changes the interface; it does not automatically solve security, relationship, or data-quality requirements. B (Add the Chatter feed component from the AppleChange.) misses the mark because A Lightning component changes the interface; it does not automatically solve security, relationship, or data-quality requirements. D (Add a custom Chatter feed component.) handles a different concern; A Lightning component changes the interface; it does not automatically solve security, relationship, or data-quality requirements. This keeps the build declarative, traceable in Setup, and easier to support when the org changes later.

質問**: 46**
Ursa Major Solarは、太陽系に2つ目の太陽を発見したことを受け、`Sun_c`と`Planet_c`間のフィールド関係をマスター -ディテールではなくルックアップに変更したいと考えています。アプリ開発者は、この変更を行う前に、レポートへの影響についてどのような点を考慮すべきでしょうか？
A. Salesforce によって作成された `Sun_c` の `Planet_c` レポートタイプで作成された既存のレポートは使用できなくなりますが、削除されません。
B. 既存のロールアップ集計フィールドは、`Sun_c` オブジェクトおよびそれらが追加されたすべてのレポートから削除されます。
C. 既存のロールアップ集計フィールドは `Sun_c` オブジェクト上に残り、レポート作成に使用できますが、更新されなくなります。
D. Salesforce によって作成された `Sun_c` の下に `Planet_c` レポートタイプで作成された既存のレポートは削除されます。
正解**: A (**コメントを発表する**)**
A clean build puts the rule where Salesforce evaluates it. Choose A. Existing reports created under the
`Sun_c` with `Planet_c` report type made by Salesforce will be unusable, but not deleted.
The requirement in the stem is: After discovering a second sun in the solar system, Ursa Major Solar wants to change the field relationship between `Sun_c` and `Planet_c` to a lookup rather than a master- detail. What should an app builder consider. The selected option addresses that requirement at the point where Salesforce actually evaluates the behavior.

The underlying concept is custom report types. Custom report types define reportable object relationships, including whether the report includes only matching child records or parent records with and without children.
That is why this answer is more defensible than a workaround. It can be reviewed in Setup, included in the correct deployment path, and tested by impersonating or logging in as a representative user where appropriate.
The remaining answers do not control the same Salesforce behavior. B (Existing roll-up summary fields will be deleted from the `Sun_c` object and from any reports where they were added.) does not fit cleanly; Reporting configuration helps analysis, but it does not enforce the requested record behavior. C (Existing roll- up summary fields will remain on the `Sun_c` object and available for reporting, but will no longer update.) is only a partial match because Reporting configuration helps analysis, but it does not enforce the requested record behavior. D (Existing reports created under the `Sun_c` with `Planet_c` report type made by Salesforce will be deleted.) misses the mark because Reporting configuration helps analysis, but it does not enforce the requested record behavior. The result is a cleaner app design because the behavior is handled by the feature Salesforce built for that purpose.

質問: **47**
Universal Containers (UC) では、3 つの別々のカスタムオブジェクトに 「ステータス」という選択リストフィールドが必要です。UC では、このフィールドの値リストを各オブジェクト間で共有する必要があります。アプリ開発者はどの機能を使用すべきでしょうか？
A. 共有カスタムフィールド
B. グローバル選択リスト値 Se
C. 依存選択リスト
D. 関連ピックリスト
正解: **(正解を表示します)**
The platform gives a native tool for this situation. Use B. Global Picklist Value Se. The scenario is asking for platform configuration, and the reason is straightforward: The

Platform App Builder role is to choose the standard Salesforce capability that owns the behavior: data structure, access, automation, user experience, reporting, or deployment. The wording points to this control because the business result must be reliable when real users work with real records.
For the stated requirement - Universal Containers (UC) needs a picklist field called Status on three Separate custom objects. UC has a requirement to share the list of values for this field Across each object. Which feature should an app builder. - the selected answer handles the behavior directly. It is not just a way to make the page look cleaner or to remind users what to do. It places the responsibility in the Salesforce feature that owns the outcome. That matters in an App Builder implementation because access, automation, relationships, page rendering, reporting, and deployment each have different enforcement points.
The other choices would leave a gap in the implementation. A (Shared Custom Field) misses the mark because it addresses a neighboring feature area rather than the exact behavior requested. C (Dependent Picklist) handles a different concern; That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option. D (Related Picklist) is less defensible because That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option. In a real org, the builder should include negative testing as well, proving that the rule does not fire or display outside the intended conditions.

質問**: 48**
ユニバーサルコンテナには、サンドボックス環境から本番環境にデプロイする必要のある変更が含まれています。アプリケーション開発者は、変更セットがデプロイ可能であることを確認するために、どこを確認すればよいでしょうか？
A. デプロイメント設定
B. 受信変更セット
C. 展開状況
D. 受信変更セット
正解**: (**正解を表示します**)**
A production admin would not solve this by decoration. In this case, Choose C. Deployment Status. Change sets move supported metadata between connected Salesforce environments. The receiving org uses inbound change sets and deployment validation to confirm whether the package can be deployed safely.
The stem gives enough detail to reject surface-level fixes. Salesforce should be configured so the behavior is enforced or rendered by the feature designed for it. When the requirement touches App Deployment, the builder should avoid crossing into a different layer unless there is a real platform limitation. The selected answer follows that discipline and creates the least surprising result for admins and users.

The remaining answers do not control the same Salesforce behavior. A (Deployment Settings) would be fragile here because Deployment tooling moves or validates metadata; it does not create the runtime business behavior itself. B (Inbound Change Sets) does not fit cleanly; Deployment tooling moves or validates metadata; it does not create the runtime business behavior itself. D (Inbound Change Sets) misses the mark because Deployment tooling moves or validates metadata; it does not create the runtime business behavior itself. The result is a cleaner app design because the behavior is handled by the feature Salesforce built for that purpose.

質問: **49**

DreamHouse Realtyは、商談が適格段階に進む前に、Expected_Close_Date_cというフィールドに値が入力されていることを保証したいと考えています。アプリビルダーはこの要求にどのように対応すべきでしょうか？

A. レコードタイプ

B. ページレイアウト

C. 検証ロール

D. アクティビティ履歴

正解: **(正解を表示します)**

The correct answer is the one that enforces the intended result. Choose C. Validation Role. In Salesforce terms, this is a Lightning page configuration issue. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars.

The selected configuration is the professional answer because it is native, declarative, and placed at the right scope. If the requirement later expands, another administrator can inspect the same metadata area and understand how the behavior is being produced. That is far better than a solution that depends on hidden assumptions or user training.

The other options fall short for clear platform reasons. A (Record Type) is not enough: it would require extra assumptions that the scenario does not support. B (Page Layout) handles a different concern; Page layouts affect presentation; they do not grant the underlying object or field capability by themselves. D (Activity History) should be rejected because it works at the wrong scope for the object, field, page, automation, or deployment condition in the stem. Before release, test the configuration with the affected profile and realistic records, not only with a system administrator account.

For Lightning pages, activation and assignment determine who sees the page. Component visibility then controls whether a specific component appears for a given condition.

質問: **50**

AW Computing の営業担当者が Chatter の料金表をスクロールしている際に、探している情報が見つかりません。主要アカウントと商談レコードからの投稿のみを表示するフィルターを使用するにはどうすればよいでしょうか？
A. チャッターのドゥークマークを作成します。
B. Chatter ストリームを作成します。
C. Chatter通知を作成します。
D. Chatterグループを作成します。
正解**: (**正解を表示します**)**
The answer is controlled by runtime behavior, not by naming similarity. Chatter supports record-centered collaboration. Feed tracking surfaces selected field changes, while groups, streams, and post actions organize discussions and follow-up work. Choose B. Create a Chatter stream. This is the option that best matches the Salesforce responsibility being tested under User Interface.
A practical administrator would confirm the object context, the user action, and whether the result must be stored, calculated, displayed, automated, approved, or deployed. Here the selected answer is the one that acts at the correct point in that chain. It keeps the configuration declarative and avoids inventing a process around a problem that Salesforce already has a native feature to solve.
The other options fall short for clear platform reasons. A (Create Chatter Dookmarks.) is not enough: it would require extra assumptions that the scenario does not support. C (Create a Chatter notification.) is less defensible because it is not the most maintainable declarative control for this design. D (Create a Chatter group.) should be rejected because it works at the wrong scope for the object, field, page, automation, or deployment condition in the stem. Before release, test the configuration with the affected profile and realistic records, not only with a system administrator account.

質問**: 51**
Universal Containersは、ユーザーがアカウントに関連する契約書を閲覧する際に、価格ガイドライン文書にアクセスできるようにしたいと考えています。アプリ開発者は、この文書への容易なアクセスを実現するために、どの機能を使用すべきでしょうか？
A. 契約オブジェクト上のカスタム詳細ページリンク
B. アカウントオブジェクトに対するクイックアクション
C. アカウントオブジェクトのカスタム詳細ページリンク
D. 契約オブジェクトに対するクイックアクション
正解**: (**正解を表示します**)**
Treat the requirement as something users will depend on daily. Custom links and detail page links give users a direct navigation point from the record they are viewing. They are appropriate when the requirement is quick access to a document, URL, or related resource from a record page. The proper configuration is A. A custom detail page link on the

Contract object. This is the option that best matches the Salesforce responsibility being tested under User Interface.
A practical administrator would confirm the object context, the user action, and whether the result must be stored, calculated, displayed, automated, approved, or deployed. Here the selected answer is the one that acts at the correct point in that chain. It keeps the configuration declarative and avoids inventing a process around a problem that Salesforce already has a native feature to solve.
The rejected choices solve different problems. B (Quick Action on the Account object) should be rejected because Actions improve user entry, but they are not a substitute for relationship modeling, reporting logic, or back-end automation. C (A custom detail page link on the Account object) would be fragile here because Master-detail is powerful, but it is wrong when the records need independent ownership or when the child- parent direction is reversed. D (Quick Action on the Contracts object) does not fit cleanly; Actions improve user entry, but they are not a substitute for relationship modeling, reporting logic, or back-end automation.
The configuration remains understandable for future admins because the decision is visible in metadata rather than buried in a workaround.

質問**: 52**
Universal Containers (UC) は、カスタムの Invoice オブジェクトとカスタム Invoice Line Item オブジェクトを持っています。Invoice Line Item オブジェクトは Invoice オブジェクトへのルックアップ関係を持っています。UC はルックアップ関係をマスター/ディテール関係に変換したいと考えていますが、変換できません。この関係変換を妨げている可能性のある 2 つの理由は何ですか？ 2 つ選択してください。
A. 請求書明細項目には既に2つのマスター/ディテール関係があります。
B. カスタムオブジェクトは、マスター/ディテール関係の詳細側に配置できません。
C. 請求書明細レコードが存在するが、請求書検索フィールドには入力されていない。
D. 請求書オブジェクトにはロールアップ集計フィールドがあります。
正解**: (**正解を表示します**)**
A clean build puts the rule where Salesforce evaluates it. In this case, Choose A. There are already two master- detail relationships on the Invoice Line Item.; C. Invoice Line Item records exist without having the Invoice lookup field populated. Master-detail relationships create ownership, security, and lifecycle dependency between parent and child records. A junction object uses two master-detail relationships to represent a many- to-many model. The stem gives enough detail to reject surface-level fixes. Salesforce should be configured so the behavior is enforced or rendered by the feature designed for it. When the requirement touches Data Modeling and Management, the builder should avoid crossing into a different layer unless there is a real platform limitation.

The selected answer follows that discipline and creates the least surprising result for admins and users.
The remaining answers do not control the same Salesforce behavior. B (Custom objects are unable to be on the detail side of a master-detail Relationship.) does not fit cleanly; Master-detail is powerful, but it is wrong when the records need independent ownership or when the child-parent direction is reversed. D (There is a roll-up summary field on the Invoice object.) misses the mark because it addresses a neighboring feature area rather than the exact behavior requested. The result is a cleaner app design because the behavior is handled by the feature Salesforce built for that purpose.

質問: **53**
AW Computingの営業オペレーションチームは、さまざまな理由でアカウントを削除しています。営業オペレーションディレクターは、営業担当者が積極的に販売活動を行っているアカウントが削除されてしまうことを懸念しています。アプリ開発者は、商談中のアカウントが削除されないようにするにはどうすればよいでしょうか？
A. 営業担当者プロファイルから削除権限を削除します。
B. アカウントレイアウトから削除ボタンを削除します。
C. Account オブジェクトに検証ルールを作成します。
D. アカウントオブジェクトにアプリトリガーを作成します。
正解: **(**正解を表示します**)**
The answer is controlled by runtime behavior, not by naming similarity. The Platform App Builder role is to choose the standard Salesforce capability that owns the behavior: data structure, access, automation, user experience, reporting, or deployment. Choose D. Create an Apps Trigger on the Account object. This is the option that best matches the Salesforce responsibility being tested under Business Logic and Process Automation.
A practical administrator would confirm the object context, the user action, and whether the result must be stored, calculated, displayed, automated, approved, or deployed. Here the selected answer is the one that acts at the correct point in that chain. It keeps the configuration declarative and avoids inventing a process around a problem that Salesforce already has a native feature to solve.
The other options fall short for clear platform reasons. A (Remove the Delete permission from the Sales Rep profile.) is not enough: Permission controls are essential for access, but they do not perform calculations or process automation. B (Remove the delete button on the account layout.) handles a different concern; it handles only part of the requirement and leaves the controlling Salesforce behavior unresolved. C (Create a validation rule on the Account object.) is less defensible because A validation rule blocks saves; it does not create records, route approvals, display components, or grant access. Before release, test the configuration with the affected profile and realistic records, not only with a system administrator account.

質問: **54**
DreamHouse Realtyは、外部システムからSalesforceに物件レコードをインポートしたいと考えています。アプリビルダーは、外部システムからの物件IDを格納するために外部IDフィールドを使用します。外部IDとして許可されているフィールドタイプはどれですか？(2つ選択してください)
A. 電話番号フィールド
B. URLフィールド
C. テキストフィールド
D. 数値フィールド
正解: **(正解を表示します)**
Treat the requirement as something users will depend on daily. The proper configuration is C. Text field; D.
Number field. In Salesforce terms, this is a data import and matching issue. Salesforce import tools differ by data volume, matching logic, and operation. Data Import Wizard is guided and duplicate-aware for supported objects; Data Loader is stronger for larger updates, upserts, and clearing values.
The selected configuration is the professional answer because it is native, declarative, and placed at the right scope. If the requirement later expands, another administrator can inspect the same metadata area and understand how the behavior is being produced. That is far better than a solution that depends on hidden assumptions or user training.
The rejected choices solve different problems. A (Phone field) is less defensible because it is not the most maintainable declarative control for this design. B (URL field) should be rejected because it works at the wrong scope for the object, field, page, automation, or deployment condition in the stem. The configuration remains understandable for future admins because the decision is visible in metadata rather than buried in a workaround.
For imports, the builder should choose the tool based on record count, supported object, operation type, duplicate handling, and the presence of an external identifier.

質問: **55**
Universal Containersには、単一の連絡先Lightningレコードページがあります。あるコンポーネントがページ上で多くのスペースを占めていますが、マーケティングプロファイルを持つユーザーには必要ありません。この問題を解決するために、アプリビルダーは何をすべきでしょうか？
A. コンポーネント可視性フィルター
B. AppExchange
C. 詳細ページレイアウト
D. フィールドレベルのセキュリティ

正解: **A** (コメントを発表する)
The useful clue is the business action being requested. The best answer is A. Component visibility filter.
Change sets move supported metadata between connected Salesforce environments. The receiving org uses inbound change sets and deployment validation to confirm whether the package can be deployed safely.
The selected answer is stronger because it meets both the functional requirement and the maintainability requirement. In a Salesforce implementation, a solution that only works for one user, one page, or one data sample is not good enough. The correct feature has to behave consistently across the intended profiles, record types, apps, and devices that are part of the design.
None of the other options gives the same administrative control. B (AppExchange) is only a partial match because it does not provide the same direct administrative control as the selected option. C (Detail page layouts) misses the mark because Page layouts affect presentation; they do not grant the underlying object or field capability by themselves. D (Field-level security) is not enough: Permission controls are essential for access, but they do not perform calculations or process automation. This is the stronger design because it respects security, data quality, user experience, and lifecycle management at the same time.
For deployment work, related metadata matters. Fields, layouts, actions, permissions, tabs, flows, and dependent components may all need to travel together for the target org to behave as expected.

質問: **56**
アプリ開発者は、ユーザーが開発者による最新機能のユーザー受け入れテストを実施できるサンドボックス環境を構築したいと考えています。サンドボックスには、テンプレートを使用して構成された約500MBのデータが含まれている必要があります。また、サンドボックスは毎週更新される必要があります。これらの要件を満たすサンドボックスはどれでしょうか？
A. 完全なサンドボックス
B. 開発者ファイルサンドボックス
C. 開発者サンドボックス
D. 部分コピーサンドボックス
正解: (正解を表示します)
The answer comes from Salesforce's separation of responsibilities. Use D. Partial Copy sandbox. Sandbox selection balances data copy, refresh cadence, isolation, testing depth, and cost. Smaller sandboxes fit independent build work; larger sandboxes fit integration and production-like testing.
The selected answer is stronger because it meets both the functional requirement and the maintainability requirement. In a Salesforce implementation, a solution that only works for

one user, one page, or one data sample is not good enough. The correct feature has to behave consistently across the intended profiles, record types, apps, and devices that are part of the design.
The alternatives are plausible only if the requirement is read too narrowly. A (Full sandbox) should be rejected because Deployment tooling moves or validates metadata; it does not create the runtime business behavior itself. B (Developer File sandbox) would be fragile here because Deployment tooling moves or validates metadata; it does not create the runtime business behavior itself. C (Developer sandbox) does not fit cleanly; Deployment tooling moves or validates metadata; it does not create the runtime business behavior itself. A complete implementation should also check page assignment, field access, record ownership, and mobile behavior where those factors apply.
For sandboxes, the builder should match the environment to the work: isolated configuration, collaborative testing, data-dependent testing, or production rehearsal.

質問**: 57**
Ursa Major Solar (UNS) は、アカウントの共有にパブリックモデルを使用しています。UNS は、より制限的な共有モデルに移行したいと考えていますが、営業チームが引き続き販売レコードタイプのすべてのアカウントレコードにアクセスできるようにしたいと考えています。この変更を実装するために、アプリ開発者はどの 2 つのアクションを実行する必要がありますか？ 2 つの回答を選択してください。
A. カスタムベースの共有ルールを作成します。
B. 組織全体のデフォルト設定を更新します。
C. 営業プロファイルを更新します。
D. 所有者ベースの共有ルールを作成します。
正解**: (**正解を表示します**)**
The correct choice is the one that will still work after release. The best answer is A. Create a custom-based sharing rule.; B. Update the organization-wide defaults. The scenario is asking for platform configuration, and the reason is straightforward: The Platform App Builder role is to choose the standard Salesforce capability that owns the behavior: data structure, access, automation, user experience, reporting, or deployment. The wording points to this control because the business result must be reliable when real users work with real records.
For the stated requirement - Ursa Major Solar (UNS) uses a public sharing model for accounts. UNS would like to move to a more restrictive sharing model but wants the Sales team to continue to have access to all account records with the sales. - the selected answer handles the behavior directly. It is not just a way to make the page look cleaner or to remind users what to do. It places the responsibility in the Salesforce feature that owns the outcome. That matters in an App Builder implementation because access, automation,

relationships, page rendering, reporting, and deployment each have different enforcement points.
The distractors are not equal substitutes. C (Update the Sales profile.) should be rejected because it works at the wrong scope for the object, field, page, automation, or deployment condition in the stem. D (Create an owner-based sharing rule.) would be fragile here because Sharing controls record visibility; it does not change field type, page composition, or automation timing. That is the kind of solution a Salesforce admin can document, migrate, and troubleshoot without creating hidden technical debt.

質問**: 58**
Ursa Major Solarでは、惑星オブジェクトに 惑星の詳細」という新しいフィールドを追加し、ユーザーが画像やリンクを含む詳細な説明を記述できるようにする必要があります。この要件を満たすために、アプリ開発者はどのフィールドタイプを使用すべきでしょうか？
A. URL
B. 複数選択可能なピックリスト
C. 長文テキストエリア
D. リッチテキストエリア
正解**: (**正解を表示します**)**
The requirement points to a specific Salesforce control. The proper configuration is D. Rich text area. In Salesforce terms, this is a record sharing targets issue. Manual record sharing can grant access to individual users, public groups, and roles with subordinates. It is a record-level access mechanism, separate from field- level security and permission set assignment.
The selected configuration is the professional answer because it is native, declarative, and placed at the right scope. If the requirement later expands, another administrator can inspect the same metadata area and understand how the behavior is being produced. That is far better than a solution that depends on hidden assumptions or user training.
The wrong answers confuse adjacent platform concepts. A (URL) is only a partial match because it does not provide the same direct administrative control as the selected option. B (Multi-select picklist) misses the mark because That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option. C (Long text area) is not enough: That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option. This keeps the build declarative, traceable in Setup, and easier to support when the org changes later.

質問**: 59**

アプリ開発者がカスタムオブジェクトと10個のフィールドを作成したいと考えています。オブジェクト、フィールド、およびリレーションシップを1か所から迅速に作成するには、何を使用すればよいでしょうか？
A. スキーマビルダー
B. Lightning オブジェクトクリエーター
C. フィールド権限の管理
D. 開発者コンソール
正解: **A** (コメントを発表する)
The key is to separate behavior from appearance. The best answer is A. Schema Builder. The Platform App Builder role is to choose the standard Salesforce capability that owns the behavior: data structure, access, automation, user experience, reporting, or deployment. The selected answer is stronger because it meets both the functional requirement and the maintainability requirement. In a Salesforce implementation, a solution that only works for one user, one page, or one data sample is not good enough. The correct feature has to behave consistently across the intended profiles, record types, apps, and devices that are part of the design.
None of the other options gives the same administrative control. B (Lightning Object Creator) is only a partial match because A Lightning component changes the interface; it does not automatically solve security, relationship, or data-quality requirements. C (Manage Field Permissions) misses the mark because Permission controls are essential for access, but they do not perform calculations or process automation. D (Developer Console) is not enough: it would require extra assumptions that the scenario does not support. This is the stronger design because it respects security, data quality, user experience, and lifecycle management at the same time.
For general platform design, the strongest answer is the native feature that can be enforced, audited, and maintained through Salesforce metadata.

質問: **60**
営業担当者は、各アカウントに関連するカスタムフィードバックレコードの詳細を取得したいと考えています。営業担当者は、モバイル端末でのクリック数を最小限に抑えてこれを実現したいと考えています。この要件を満たすために推奨すべきソリューションはどれですか？(2つ選択してください)
A. アカウントで単一の特定アクションを作成する
B. アカウントの親としてフィードバックオブジェクトを作成します
C. アカウントにグローバルアクションを作成します
D. ほとんどのフィールドに事前定義された値を作成します
正解: **A,D** (コメントを発表する)

The key is to separate behavior from appearance. The best answer is A. Create a single-specific action in Account; D. Create predefined values for most of the fields. The decision turns on mobile action design.
Quick actions are compact, purpose-built entry points. They are especially effective in the mobile app because they reduce navigation, expose only the fields required for the task, and can prepopulate values.
This is how the same requirement would be handled in a well-governed Salesforce org: identify the controlling object, relationship, field, page, automation, or environment first, and then configure the native capability for that control point. The selected answer satisfies the requirement without requiring users to perform extra interpretation or administrators to maintain an unnecessary custom workaround.
None of the other options gives the same administrative control. B (Create a feedback object as a parent of Account) is only a partial match because it does not provide the same direct administrative control as the selected option. C (Create a global action on Account) misses the mark because Actions improve user entry, but they are not a substitute for relationship modeling, reporting logic, or back-end automation. This is the stronger design because it respects security, data quality, user experience, and lifecycle management at the same time.

質問: **61**
カスタムオブジェクトを作成する際のSchema Builderの制限事項は何ですか？
A. フィールドとリレーションシップは作成できますが、キャンバスからページレイアウトにフィールドを追加することはできません。
B. カスタムフィールドは、数式フィールドタイプを除くすべてのカスタムオブジェクトに追加できます。
C. 関係は任意のカスタムオブジェクトに対して作成できますが、標準オブジェクトとの関係は Lightning オブジェクトマネージャで構築する必要があります。
D. 新しいオブジェクト、フィールド、またはリレーションシップが作成されるたびに「保存」をクリックする必要があります
正解: **(**正解を表示します**)**
The platform gives a native tool for this situation. Use A. Fields and relationships can be created, but they will be unable to add the fields to the page layout from the canvas. The decision turns on Lightning page configuration. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars.
This is how the same requirement would be handled in a well-governed Salesforce org: identify the controlling object, relationship, field, page, automation, or environment first, and then configure the native capability for that control point. The selected answer satisfies

the requirement without requiring users to perform extra interpretation or administrators to maintain an unnecessary custom workaround.
The other choices would leave a gap in the implementation. B (Custom fields can be added to any custom objects, excluding formula field types.) is not enough: A formula is calculated display logic and cannot always store static values or perform record updates. C (Relationships can be made to any custom objects, but any relationships to standard objects should be built in Lightning Object Manager.) handles a different concern; A Lightning component changes the interface; it does not automatically solve security, relationship, or data-quality requirements. D (" Save " should be clicked each time a new object, field, or relationship is create) is less defensible because it is not the most maintainable declarative control for this design. In a real org, the builder should include negative testing as well, proving that the rule does not fire or display outside the intended conditions.

質問: **62**
Ursa Major Solar では、カスタムオブジェクト `delay_` とカスタムオブジェクト `Size_` の間にルックアップ関係があります。アプリケーション開発者は、各 `delay_` レコードに関連付けられた `Size_` レコードの総数をカウントするロールアップ集計フィールドを作成したいと考えています。現在の構成では、目的の結果を達成する能力にどのような影響がありますか？
A. ロールアップ集計は、`delay_` オブジェクトに数式フィールドを作成することで実現できます。
B. ロールアップ集計フィールドを作成するには、このルックアップ関係をマスター/ディテール関係に変換する必要があります。
C. 関連するすべての `Size_` レコードを選択するフィールドフィルターを使用して、`delay_` オブジェクトにロールアップ集計フィールドを作成する必要があります。
D. ロールアップ集計は、`Size_` オブジェクトに数式フィールドを作成することで実現できます。
正解: **(正解を表示します)**

A production admin would not solve this by decoration. Roll-up summary fields calculate values from child detail records and store the result on the master. The relationship type and field type decide whether the calculation is available declaratively. The correct response is therefore B. This lookup relationship will need to be converted to a master-detail relationship before a roll-up summary field can be created.
The scenario is not asking for a feature list. It is asking which configuration produces the requested result with the fewest moving parts. A Salesforce app builder should favor a standard, visible Setup configuration whenever it fully meets the business need. That approach makes the build easier to validate in sandbox and easier to explain during release review.
The remaining answers do not control the same Salesforce behavior. A (The roll-up summary can be achieved by creating a formula field on the `delay_` object.) would be fragile here because A formula is calculated display logic and cannot always store static values or perform record updates. C (A roll-up summary field will need to be created on the `delay_` object with a field filter that selects all related `Size_` records.) is only a partial match because it does not provide the same direct administrative control as the selected option. D (The roll-up summary can be achieved by creating a formula field on the `Size_` object.) misses the mark because A formula is calculated display logic and cannot always store static values or perform record updates. The result is a cleaner app design because the behavior is handled by the feature Salesforce built for that purpose.

質問**: 63**
Northern Trail Outfitters のマーケティングディレクターは、連絡先にマーケティングコミュニケーションが送信されてから経過した日数を追跡する数式フィールドをアプリビルダーに作成してもらいたいと考えています。ディレクターは整数値のみに関心があります。この差を計算するために日付を返す関数はどれを使用すればよいでしょうか？
A. TODAY()
B. DATEVALUE()
C. NOW()
D. DATETIMEVALUE()
正解**: A (**コメントを発表する**)**
A clean build puts the rule where Salesforce evaluates it. Choose A. TODAY(). In Salesforce terms, this is a formula logic issue. Formula and validation expressions must respect Salesforce data types. Picklists require picklist-aware handling, dates require date functions or date arithmetic, and image or text formulas must return the correct type.
The selected configuration is the professional answer because it is native, declarative, and placed at the right scope. If the requirement later expands, another administrator can inspect the same metadata area and understand how the behavior is being produced. That is far better than a solution that depends on hidden assumptions or user training.

The remaining answers do not control the same Salesforce behavior. B (DATEVALUE()) does not fit cleanly; it solves a symptom of the problem rather than the Salesforce mechanism that owns the result. C (NOW()) is only a partial match because it does not provide the same direct administrative control as the selected option.
D (DATETIMEVALUE()) misses the mark because it addresses a neighboring feature area rather than the exact behavior requested. The result is a cleaner app design because the behavior is handled by the feature Salesforce built for that purpose.
For formulas, data type conversion is critical. TEXT, ISPICKVAL, DATEVALUE, TODAY, NOW, IMAGE, CASE, and LEFT solve different expression problems and cannot be swapped casually.

質問**: 64**
アプリ開発者がAppExchangeからカスタムLightningコンポーネントをインストールしました。レコードページでそのコンポーネントを使用できるように構成するには、次に何をすればよいでしょうか？
A. ページレイアウトエディターを使用してレコードページを編集します > Visualforce コンポーネントをページにドラッグします。
B. Lightning App Builder を使用してレコード ページを編集します > コンポーネントをページにドラッグします。
C. App Manager を使用してレコード ページを編集します > コンポーネントをページにドラッグします。
D. ページレイアウトエディタを使用してレコードページを編集します > コンポーネントをページにドラッグします。
正解**: (**正解を表示します**)**
This is a governance decision before it is a configuration click. A Lightning component must be exposed for the surface where it is used. A component built for one page type or action surface does not automatically become available everywhere in Lightning Experience. The correct response is therefore B. Edit a record page using Lightning App Builder > Drag the component onto the page.
The scenario is not asking for a feature list. It is asking which configuration produces the requested result with the fewest moving parts. A Salesforce app builder should favor a standard, visible Setup configuration whenever it fully meets the business need. That approach makes the build easier to validate in sandbox and easier to explain during release review.
The rejected choices solve different problems. A (Edit a record page using the Page Layout editor > Drag the Visualforce component onto the page.) is less defensible because Page layouts affect presentation; they do not grant the underlying object or field capability by themselves. C (Edit a record page using App Manager > Drag the component onto the page.) would be fragile here because A Lightning component changes the interface; it

does not automatically solve security, relationship, or data-quality requirements. D (Edit a record page using the Page Layout editor > Drag the component onto the page.) does not fit cleanly; Page layouts affect presentation; they do not grant the underlying object or field capability by themselves. The configuration remains understandable for future admins because the decision is visible in metadata rather than buried in a workaround.

質問: **65**
ロールアップ集計フィールドから数式フィールドが参照されるのを防ぐ解決策はどれですか？(2つ選択してください)
A. 数式フィールドのNOW()関数
B. 式フィールドによって参照されるフィールドを更新するクロスオブジェクト式
C. 数式フィールドの CASE() 関数
D. 数式フィールドで参照されるクロスオブジェクトフィールド
正解: **(正解を表示します)**
Treat the requirement as something users will depend on daily. The proper configuration is A. The NOW() function in the formula field; D. A cross-object field referenced in the formula field. In Salesforce terms, this is a formula logic issue. Formula and validation expressions must respect Salesforce data types. Picklists require picklist-aware handling, dates require date functions or date arithmetic, and image or text formulas must return the correct type. The selected configuration is the professional answer because it is native, declarative, and placed at the right scope. If the requirement later expands, another administrator can inspect the same metadata area and understand how the behavior is being produced. That is far better than a solution that depends on hidden assumptions or user training.
The rejected choices solve different problems. B (A cross-object formula updating a field referenced by the formula field) should be rejected because A formula is calculated display logic and cannot always store static values or perform record updates. C (The CASE() function in the formula field) would be fragile here because A formula is calculated display logic and cannot always store static values or perform record updates. The configuration remains understandable for future admins because the decision is visible in metadata rather than buried in a workaround.

質問: **66**
65件中7件目。Universal Containerの営業担当者は、商談がクローズされるまで商談のフィールドを変更できます。営業オペレーションチームは、商談がクローズされた後、「クローズ後フォローアップ日」と「クローズ後フォローアップコメント」フィールドを変更できます。商談がクローズされた後は、残りのフィールドは読み取り専用になります。これらの要件はどのように満たすべきでしょうか？

A. ページレイアウトでフィールドレベルのセキュリティを使用して、フィールドの編集を制限します。
B. レコードタイプを含むページレイアウトでフィールドレベルのセキュリティを使用して、フィールドの編集を制限します。
C. フィールドセットを持つレコードタイプを使用し、フィールドレベルセキュリティを使用してフィールドの編集を制限します。
D. フィールドレベルセキュリティを使用して、営業プロファイルのフィールドを読み取り専用としてマークします。
正解**: (**正解を表示します**)**
A production admin would not solve this by decoration. In this case, Choose B. Use field-level security on page layouts with record types to restrict editing fields. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars.
The stem gives enough detail to reject surface-level fixes. Salesforce should be configured so the behavior is enforced or rendered by the feature designed for it. When the requirement touches User Interface, the builder should avoid crossing into a different layer unless there is a real platform limitation. The selected answer follows that discipline and creates the least surprising result for admins and users.
The remaining answers do not control the same Salesforce behavior. A (Use field-level security on page layouts to restrict editing fields.) would be fragile here because Page layouts affect presentation; they do not grant the underlying object or field capability by themselves. C (Use record types with field sets and restrict editing fields using field-level security.) is only a partial match because Permission controls are essential for access, but they do not perform calculations or process automation. D (Use field-level security to mark fields as read-only on the Sales profile.) misses the mark because Permission controls are essential for access, but they do not perform calculations or process automation. The result is a cleaner app design because the behavior is handled by the feature Salesforce built for that purpose.

質問**: 67**
アプリビルダーはAppExchangeからコンポーネントを正常にダウンロードしましたが、Lightningホームページに追加できません。アプリビルダーがカスタムコンポーネントを追加できない原因として考えられる2つの理由は何ですか？2つ選択してください。
A. App Builder を使用してページにカスタム コンポーネントを追加するには、カスタム タブを作成する必要があります。
B. このコンポーネントをApp Builderでページに追加するには、開発者権限が必要です。
C. App Builder を使用してページにカスタム コンポーネントを追加するには、マイ ドメインがデプロイされている必要があります。

D. このコンポーネントはホームページではなくレコードページ用にタグ付けされているため、アプリビルダーに表示されません。
正解: **C,D (**コメントを発表する**)**
This is where declarative precision matters. Change sets move supported metadata between connected Salesforce environments. The receiving org uses inbound change sets and deployment validation to confirm whether the package can be deployed safely. Use C. My Domain must be deployed to add custom components to the page with the App Builder.; D. The component is tagged for record pages instead of home pages and is not ing up in The App Builder.
A clean Platform App Builder answer should satisfy the complete scenario, not merely one phrase in it. This item requires the app builder to choose the configuration that produces the requested behavior in normal Salesforce use. The selected answer fits that requirement because it works with the platform model instead of fighting it.
The other choices would leave a gap in the implementation. A (A custom tab must be created to add custom components to the page with the App Builder.) misses the mark because A Lightning component changes the interface; it does not automatically solve security, relationship, or data-quality requirements. B (The component requires a developer permission to add it to the page with the App Builder.) is not enough: A Lightning component changes the interface; it does not automatically solve security, relationship, or data- quality requirements. In a real org, the builder should include negative testing as well, proving that the rule does not fire or display outside the intended conditions.

質問: **68**
DreamHouse Realty (DR) は、ケース管理の改善を求めています。プロセス遵守を徹底し、ケースが以前の状態に戻されないようにすること、および特定のケース基準が満たされた場合に特定のフィールドが必須となるようにすることを望んでいます。アプリ開発者は、これらの要件を満たすためにどのようなソリューションを実装すべきでしょうか？
A. ヘルプテキストを使用して検証ルールを設定します。
B. 依存する選択リスト項目を作成し、それらを必須項目として設定します。
C. 承認プロセスを使用して、フィールドの基準が満たされていることを確認します。
D. ページレイアウトでフィールドを必須にします。
正解: **(**正解を表示します**)**

質問: **69**
Universal Containers社はAppExchangeでLightningコンポーネントを購入しました。これらのコンポーネントはどの2つの分野で使用できますか？2つ選択してください。
A. Lightning App Builder
B. 時間切れ
C. 検証ルール

D. クイックアクション
正解: **(**正解を表示します**)**
The safest reading of the scenario is practical, not theoretical. Use A. Lightning App Builder; D. Quick Action. The selected option is not chosen because it sounds familiar; it is chosen because it gives Salesforce the right instruction at runtime. A Lightning component must be exposed for the surface where it is used. A component built for one page type or action surface does not automatically become available everywhere in Lightning Experience.
The scenario describes a business outcome, not merely an administrative preference. If the feature is configured correctly, users get the intended result without depending on memory, spreadsheets, or manual policing. That is the professional standard for Platform App Builder work: place the rule in metadata, test it under the correct permissions, and keep the design supportable.
The alternatives are plausible only if the requirement is read too narrowly. B (Time Failure) would be fragile here because it could produce an inconsistent result once different users, records, or apps are tested. C (Validation Rule) does not fit cleanly; A validation rule blocks saves; it does not create records, route approvals, display components, or grant access. A complete implementation should also check page assignment, field access, record ownership, and mobile behavior where those factors apply.

質問: **70**
Universal Containersは、`Account_Region`というカスタム選択リスト、またはAccountオブジェクトを使用しています。営業担当副社長は、このフィールドの値を商談に表示するように求めています。アプリ開発者は、このソリューションをどのように作成すればよいでしょうか？
A. フィールド履歴追跡
B. クロスオブジェクト式
C. リンクフィールド
D. フィールドレベルのセキュリティ
正解: **A (**コメントを発表する**)**
This is a governance decision before it is a configuration click. The proper configuration is A. Field History tacking. The requirement in the stem is: Universal Containers uses a custom picklist called `Account_Region ` or the Account object. The vice president of sales has asked that the value of this field is visible on Opportunities. How should an app builder. The selected option addresses that requirement at the point where Salesforce actually evaluates the behavior.
The underlying concept is platform configuration. The Platform App Builder role is to choose the standard Salesforce capability that owns the behavior: data structure, access, automation, user experience, reporting, or deployment. That is why this answer is more defensible than a workaround. It can be reviewed in Setup, included in the correct

deployment path, and tested by impersonating or logging in as a representative user where appropriate.
The rejected choices solve different problems. B (Cross-object formula) should be rejected because A formula is calculated display logic and cannot always store static values or perform record updates. C (Linking field) would be fragile here because it could produce an inconsistent result once different users, records, or apps are tested. D (Field-level security) does not fit cleanly; Permission controls are essential for access, but they do not perform calculations or process automation. The configuration remains understandable for future admins because the decision is visible in metadata rather than buried in a workaround.

質問: **71**
Universal Containers は、製品の詳細をキャプチャするための Component というカスタム オブジェクトを作成しました。アプリ ビルダーは、Product の関連リストとして Component を使用するために、どのようなアプローチを取るべきでしょうか?
A. コンポーネントと製品を関連付ける結合オブジェクトを作成します。コンポーネント関連リストを製品ページレイアウトに追加します。
B. 製品に関するロールアップを作成します。製品ページのレイアウトにコンポーネント関連リストを追加します。
C. コンポーネントから製品へのルックアップ関係を作成します。コンポーネント関連リストを製品ページレイアウトに追加します。
D. 製品とコンポーネント間のマスター/ディテール関係を作成します。コンポーネント関連リストを製品ページレイアウトに追加します。
正解: **(**正解を表示します**)**
The answer comes from Salesforce's separation of responsibilities. Use C. Create a lookup relationship on Component to Product. Add the Component related list to the Product page layout. The decision turns on related-list display. Enhanced related lists provide richer Lightning behavior than a basic related list, including more visible columns, sorting, and text wrapping when users need to work with related records from the parent page.
This is how the same requirement would be handled in a well-governed Salesforce org: identify the controlling object, relationship, field, page, automation, or environment first, and then configure the native capability for that control point. The selected answer satisfies the requirement without requiring users to perform extra interpretation or administrators to maintain an unnecessary custom workaround.
The alternatives are plausible only if the requirement is read too narrowly. A (Create a junction object to relate Component and Product. Add the Component related list to the Product page layout.) should be rejected because Page layouts affect presentation; they do not grant the underlying object or field capability by themselves. B (Create a roll-up on Product. Add the Component related list to the Product page layout.) would be fragile here

because Page layouts affect presentation; they do not grant the underlying object or field capability by themselves. D (Create a master-detail relationship on Product to Component. Add the Component related list to the Product page layout.) is only a partial match because Page layouts affect presentation; they do not grant the underlying object or field capability by themselves. A complete implementation should also check page assignment, field access, record ownership, and mobile behavior where those factors apply.

質問: **72**
Dream House Realty (DR) のサービスマネージャーは、ケース管理の改善を求めています。ケースが以前の状態に戻せないように、また特定のケース基準が満たされた場合に特定のフィールドが必須となるように、コンプライアンスを徹底したいと考えています。アプリ開発者は、これらの要件を満たすためにどのようなソリューションを実装すべきでしょうか？
A. 承認プロセス
B. 依存選択リスト
C. ケースフロー
D. 検証ルール
正解: **(正解を表示します)**
Start at the layer that controls the outcome. The best answer is D. Validation rules. The scenario is asking for data validation, and the reason is straightforward: Validation rules prevent bad data from being saved. They are appropriate when the requirement is to enforce a condition at save time rather than merely warn, display, or report on the problem later. The wording points to this control because the business result must be reliable when real users work with real records.
For the stated requirement - The Dream House Realty (DR) service manager asks for some Improvements in case management. They want to enforce compliance so That cases are unable to be reverted to an earlier case status, and to Ensure that certain. - the selected answer handles the behavior directly. It is not just a way to make the page look cleaner or to remind users what to do. It places the responsibility in the Salesforce feature that owns the outcome. That matters in an App Builder implementation because access, automation, relationships, page rendering, reporting, and deployment each have different enforcement points.
The distractors are not equal substitutes. A (Approval process) handles a different concern; An approval process is for formal sign-off, not for every automation, display, import, or security requirement. B (Dependent picklist) is less defensible because That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option. C (Case flow) should be rejected because Flow is broad, but it is not correct unless the requirement is specifically about guided or automated logic. That is the kind of solution a Salesforce admin can document, migrate, and troubleshoot without creating hidden technical debt.

質問: **73**
Universal Containers (UC) は、5 GB 未満の本番データの一部に対してコードをテストしたいと考えています。さらに、UC はこのサンドボックスを毎週週末に更新したいと考えています。これを実現するには、どのタイプのサンドボックスを使用すればよいでしょうか？
A. デベロッパープロ
B. 部分コピー
C. 完全コピー
D. 開発者
正解: **(**正解を表示します**)**
A Partial Copy sandbox is the only option that meets both the data storage and the refresh interval requirements for Universal Containers (UC).

質問: **74**
Cloud Kicksには、プライベート共有設定を持つカスタムオブジェクトがあります。企業は、個々のレコードをケースバイケースで特定の人または部署と共有したいと考えています。ビジネスユーザーが個々のレコードを手動で共有するには、どの3つのオプションがありますか？3つ選択してください。
A. アクセス許可セットグループ
B. 役割と購読
C. 公開グループ
D. IDユーザー
E. プライベートグループ
正解: **B,C,D (**コメントを発表する**)**
Read the stem as an implementation requirement. The proper configuration is B. Roles and Subscriptions; C.
Public Groups; D. ID Users. The requirement in the stem is: Cloud Kicks has a custom object with a private sharing setting. The business wants to share individual records with specific people or departments on a case- by-case basis. Which three options does the business user have. The selected option addresses that requirement at the point where Salesforce actually evaluates the behavior.
The underlying concept is record sharing targets. Manual record sharing can grant access to individual users, public groups, and roles with subordinates. It is a record-level access mechanism, separate from field-level security and permission set assignment. That is why this answer is more defensible than a workaround. It can be reviewed in Setup, included in the correct deployment path, and tested by impersonating or logging in as a representative user where appropriate.
The wrong answers confuse adjacent platform concepts. A (Permission Set Groups) is only a partial match because Permission controls are essential for access, but they do not

perform calculations or process automation. E (Private Groups) is less defensible because it is not the most maintainable declarative control for this design. This keeps the build declarative, traceable in Setup, and easier to support when the org changes later.

質問: **75**
Ursa Major Solar (UMS) のサービスコーディネーター (SC) は、請求書発行前に、特定の地域を担当する技術者が所有する作業指示書の最終確認を行います。作業指示書をクローズする前に、SC は誤って追加されたデータや添付ファイルを修正または削除する必要があります。また、SC は技術者が所有するその他の関連記録にもアクセスする必要があります。プライベートデータモデルを前提とした場合、必要なアクセス権限を提供するソリューションはどれでしょうか？
A. その地域の技術者がSCと更新レコードを使用して割り当てを完了します。
B. 対応する監査グループを持つ SC アクティビティ リストを作成します。
C. SC に 「すべてのデータを変更する」システム権限を持つ権限セットを付与します。
D. SC のプロファイルの作業指示アクセスを 「すべて変更」に変更します。
正解: **(**正解を表示します**)**
This question is really testing ownership of the feature. In this case, Choose B. Create a SC activity list with a corresponding audit group. Lookup relationships connect records while allowing looser ownership and lifecycle control than master-detail. They are appropriate when records should remain more independent or when the relationship is optional.
The stem gives enough detail to reject surface-level fixes. Salesforce should be configured so the behavior is enforced or rendered by the feature designed for it. When the requirement touches Data Modeling and Management, the builder should avoid crossing into a different layer unless there is a real platform limitation.
The selected answer follows that discipline and creates the least surprising result for admins and users.
The other options fall short for clear platform reasons. A (Close an assignment via that update record record by technicians in that region with the SC.) is not enough: it would require extra assumptions that the scenario does not support. C (Give the SC a permission set with the Modify All Data system permission.) is less defensible because Permission controls are essential for access, but they do not perform calculations or process automation. D (Change work order access on the SC ' s profile to Modify all ' .) should be rejected because it works at the wrong scope for the object, field, page, automation, or deployment condition in the stem. Before release, test the configuration with the affected profile and realistic records, not only with a system administrator account.

質問: **76**

アプリビルダーがSalesforceにデータをロードしています。新しいレコードをレガシーシステムにリンクするために、アカウントオブジェクトのレガシーIDを追跡するフィールドが使用されます。今後のデータロードでは、レコードを更新する際にこのIDが使用されます。どの2つのフィールド属性を選択する必要がありますか？2つ選択してください。
A. テキスト(暗号化済み)
B. 外部ID
C. ユニーク
D. リクエスト
正解**: (**正解を表示します**)**
This is a governance decision before it is a configuration click. In this case, The proper configuration is B.
External ID; C. Unique. Salesforce import tools differ by data volume, matching logic, and operation. Data Import Wizard is guided and duplicate-aware for supported objects; Data Loader is stronger for larger updates, upserts, and clearing values.
The stem gives enough detail to reject surface-level fixes. Salesforce should be configured so the behavior is enforced or rendered by the feature designed for it. When the requirement touches Data Modeling and Management, the builder should avoid crossing into a different layer unless there is a real platform limitation.
The selected answer follows that discipline and creates the least surprising result for admins and users.
The rejected choices solve different problems. A (Text (encrypted)) is less defensible because That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option.
D (Request) does not fit cleanly; it solves a symptom of the problem rather than the Salesforce mechanism that owns the result. The configuration remains understandable for future admins because the decision is visible in metadata rather than buried in a workaround.
For imports, the builder should choose the tool based on record count, supported object, operation type, duplicate handling, and the presence of an external identifier.

質問: **77**
Ursa Major Solarには、天体カスタムオブジェクト用のLightningレコードページが1つだけありますが、アプリビルダーがモバイルアプリユーザーのみに制限したいLightningコンポーネントがあります。Lightningアプリビルダーのどの機能を使用すればよいでしょうか？
A. フォームファクターのチェックボックス
B. ハイライトパネル
C. 関連リストのクイックリンク
D. コンポーネント可視性フィルター
正解: **(**正解を表示します**)**
The right answer avoids a brittle workaround. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars.
The best answer is A. Form factor checkboxes.
A clean Platform App Builder answer should satisfy the complete scenario, not merely one phrase in it. This item requires the app builder to choose the configuration that produces the requested behavior in normal Salesforce use. The selected answer fits that requirement because it works with the platform model instead of fighting it.
None of the other options gives the same administrative control. B (Highlights panel) is only a partial match because it does not provide the same direct administrative control as the selected option. C (Related list quick links) misses the mark because it addresses a neighboring feature area rather than the exact behavior requested. D (Component visibility filter) is not enough: A Lightning component changes the interface; it does not automatically solve security, relationship, or data-quality requirements. This is the stronger design because it respects security, data quality, user experience, and lifecycle management at the same time.
For Lightning pages, activation and assignment determine who sees the page. Component visibility then controls whether a specific component appears for a given condition.

質問: **78**
Ursa Major Solarは、需要増加に対応するため営業チームを増強しています。新任営業担当者の短期間での研修の一環として、マネージャーは、営業担当者がさまざまな販売プロセスにおいて必要な時に簡単にサポートを受けられるよう、ヘルプガイドを提供したいと考えています。アプリ開発者はどのようなソリューションを推奨すべきでしょうか？
A. パス
B. チャッターパブリッシャー

C. ジャーニービルダー
D. フロー
正解**: (**正解を表示します**)**
A clean build puts the rule where Salesforce evaluates it. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars. Choose A. Path. This is the option that best matches the Salesforce responsibility being tested under User Interface.
A practical administrator would confirm the object context, the user action, and whether the result must be stored, calculated, displayed, automated, approved, or deployed. Here the selected answer is the one that acts at the correct point in that chain. It keeps the configuration declarative and avoids inventing a process around a problem that Salesforce already has a native feature to solve.
The remaining answers do not control the same Salesforce behavior. B (Chatter Publisher) is not enough: it would require extra assumptions that the scenario does not support. C (Journey Builder) handles a different concern; it handles only part of the requirement and leaves the controlling Salesforce behavior unresolved. D (Flow) is less defensible because Flow is broad, but it is not correct unless the requirement is specifically about guided or automated logic. The result is a cleaner app design because the behavior is handled by the feature Salesforce built for that purpose.
For Lightning pages, activation and assignment determine who sees the page. Component visibility then controls whether a specific component appears for a given condition.

質問**: 79**
Dreamhouse Realty (DR) は多数の販売物件を所有しており、各 Property_ レコード上のすべてのオファーレコードの中で最も価値の高い物件を特定したいと考えています。オブジェクト間にマスター/ディテール関係がある場合、アプリ開発者は DR のニーズを満たすためにどのソリューションを使用すべきでしょうか？
A. ロールアップサマリー
B. 概要
C. テキストエリア(長文)
D. 複数選択ピックリスト
E. リッチテキストエリア
正解**: A (**コメントを発表する**)**
The right answer avoids a brittle workaround. The best answer is A. Roll-up Summary. The scenario is asking for roll-up summary design, and the reason is straightforward: Roll-up summary fields calculate values from child detail records and store the result on the master. The relationship type and field type decide whether the calculation is available declaratively. The wording points to this control because the business result must be reliable when real users work with real records.

For the stated requirement - Dreamhouse Realty (DR) has many properties for sale and wants to Identify the highest value of all offer records on each Property_ record Which solution should the app builder use to meet DR ' s needs provided that there. - the selected answer handles the behavior directly. It is not just a way to make the page look cleaner or to remind users what to do. It places the responsibility in the Salesforce feature that owns the outcome. That matters in an App Builder implementation because access, automation, relationships, page rendering, reporting, and deployment each have different enforcement points.
None of the other options gives the same administrative control. B (Summary) is only a partial match because it does not provide the same direct administrative control as the selected option. C (Text Area (Long)) misses the mark because That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option. D (Multi-select Picklist) is not enough: That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option. This is the stronger design because it respects security, data quality, user experience, and lifecycle management at the same time.

質問: **80**
AW Computing の採用チームは、求職者の採用承諾と入社日を 求人応募」カスタムオブジェクトに記録します。候補者が採用担当者の求人オファーを承諾すると、入社日が入力され、その後のレコード編集で変更されないようにします。アプリビルダーはどの検証式を使用すべきでしょうか？
A. (ISBLANK (Job Accepted_c) || NOT (INCHANGED (Hi_Date_=))
B. NOT (LABLANK (Job hocepted_c)) FRISCHANGED (Hire_Date
C. NOT (ISBLANK (Job_Accepted_c)) 11 ISCHANGED (Hire_Date c)
D. (ISBLANK (Job_Accepted_c) NOT (ISCHANGED (Hire_Date_0})
正解: **(正解を表示します)**
Start at the layer that controls the outcome. Formula and validation expressions must respect Salesforce data types. Picklists require picklist-aware handling, dates require date functions or date arithmetic, and image or text formulas must return the correct type. The best answer is B. NOT (LABLANK (Job hocepted_c)) FRISCHANGED (Hire_Date.
A clean Platform App Builder answer should satisfy the complete scenario, not merely one phrase in it. This item requires the app builder to choose the configuration that produces the requested behavior in normal Salesforce use. The selected answer fits that requirement because it works with the platform model instead of fighting it.
The distractors are not equal substitutes. A ((ISBLANK (Job Accepted_c) || NOT (INCHANGED (Hi_Date_=))) handles a different concern; it handles only part of the requirement and leaves the controlling Salesforce behavior unresolved. C (NOT (ISBLANK (Job_Accepted_c)) 11 ISCHANGED (Hire_Date c)) should be rejected because it works at the wrong scope for the object, field, page, automation, or deployment condition in the

stem. D ((ISBLANK (Job_Accepted_c) NOT (ISCHANGED (Hire_Date_0})) would be fragile here because it could produce an inconsistent result once different users, records, or apps are tested.
That is the kind of solution a Salesforce admin can document, migrate, and troubleshoot without creating hidden technical debt.

質問: **81**
アプリ開発者は、取引先にカスタム同期ボタンを作成し、外部システムに接続する Lightning Web コンポーネントを呼び出すようにしたいと考えています。このアクションは、カスタムステータスフィールドが 同期準備完了」に設定されている場合にのみ使用可能にする必要があります。アプリ開発者は、この機能を取引先レコードページに追加するために何を使用すればよいでしょうか？
A. カスタムリンク
B. 数式フィールド
C. 動的アクション
D. AppExchange製品
正解: **(正解を表示します)**
The important question is which feature owns the result. The proper configuration is C. Dynamic action. The requirement in the stem is: An app builder wants to create a custom Sync button on Account that will Call a Lightning Web Component that connects with an external system. This action Should only be available if the custom Status field is set to. The selected option addresses that requirement at the point where Salesforce actually evaluates the behavior.
The underlying concept is change-set deployment. Change sets move supported metadata between connected Salesforce environments. The receiving org uses inbound change sets and deployment validation to confirm whether the package can be deployed safely. That is why this answer is more defensible than a workaround. It can be reviewed in Setup, included in the correct deployment path, and tested by impersonating or logging in as a representative user where appropriate.
The rejected choices solve different problems. A (Custom link) is less defensible because it is not the most maintainable declarative control for this design. B (Formula field) should be rejected because A formula is calculated display logic and cannot always store static values or perform record updates. D (AppExchange product) does not fit cleanly; it solves a symptom of the problem rather than the Salesforce mechanism that owns the result. The configuration remains understandable for future admins because the decision is visible in metadata rather than buried in a workaround.

質問: **82**

アプリ開発者がユーティリティバーに表示させたいカスタムコンポーネントがあるのですが、そのコンポーネントが利用できません。コンポーネントにはどのようにタグ付けすればよいでしょうか？
A. アプリマネージャーで使用します。
B. Lightning App Builder で使用します。
C. 記録ページで使用します。
D. ユーティリティバーで使用します。
正解: **D (**コメントを発表する**)**
The safest reading of the scenario is practical, not theoretical. Use D. For use on the utility bar. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars.
The selected answer is stronger because it meets both the functional requirement and the maintainability requirement. In a Salesforce implementation, a solution that only works for one user, one page, or one data sample is not good enough. The correct feature has to behave consistently across the intended profiles, record types, apps, and devices that are part of the design.
The alternatives are plausible only if the requirement is read too narrowly. A (For use in App Manager.) should be rejected because it works at the wrong scope for the object, field, page, automation, or deployment condition in the stem. B (For use in Lightning App Builder.) would be fragile here because A Lightning component changes the interface; it does not automatically solve security, relationship, or data-quality requirements. C (For use on record pages.) does not fit cleanly; it solves a symptom of the problem rather than the Salesforce mechanism that owns the result. A complete implementation should also check page assignment, field access, record ownership, and mobile behavior where those factors apply.

質問: **83**
Cloud Kicks (CK) は、すべての配送情報を Shipments_c というカスタムオブジェクトに格納します。CK のアプリビルダーは、部門メンバーがすべての翌日配送を承認できるようにするための承認プロセスを作成する役割を担っています。アプリビルダーは承認リクエストをどこにルーティングすべきでしょうか？
A. 役割
B. Queve
C. 階層構造を感じる
D. 公開グループ
正解: **D (**コメントを発表する**)**
The key is to separate behavior from appearance. The best answer is D. Public group. The scenario is asking for approval governance, and the reason is straightforward: Approval

processes are designed for formal review. They route a request, capture a decision, and can lock or update the record through submission and final actions. The wording points to this control because the business result must be reliable when real users work with real records.
For the stated requirement - Cloud Kicks (CK) captures all shipping information in a custom object called Shipments_c. CK ' s app builder is tasked with creating an approval process to ensure department members can approve all overnight shipments. - the selected answer handles the behavior directly. It is not just a way to make the page look cleaner or to remind users what to do. It places the responsibility in the Salesforce feature that owns the outcome. That matters in an App Builder implementation because access, automation, relationships, page rendering, reporting, and deployment each have different enforcement points.
None of the other options gives the same administrative control. A (Role) does not fit cleanly; it solves a symptom of the problem rather than the Salesforce mechanism that owns the result. B (Queve) is only a partial match because it does not provide the same direct administrative control as the selected option. C (Hierarchy fiele) misses the mark because it addresses a neighboring feature area rather than the exact behavior requested. This is the stronger design because it respects security, data quality, user experience, and lifecycle management at the same time.

質問**: 84**
Cloud Kicksには、プライベート共有設定を持つカスタムオブジェクトがあります。企業は、個々のレコードをケースバイケースで特定の人または部署と共有したいと考えています。ビジネスユーザーが個々のレコードを手動で共有するには、どの3つのオプションがありますか？3つ選択してください。
A. 公開グループ
B. プライベートグループ
C. アクセス許可セットグループ
D. 役割と部下
E. ユーザー
正解**: A,D,E (**コメントを発表する**)**
The answer comes from Salesforce's separation of responsibilities. Use A. Public Groups; D. Roles and Subordinates; E. Users. The selected option is not chosen because it sounds familiar; it is chosen because it gives Salesforce the right instruction at runtime. Manual record sharing can grant access to individual users, public groups, and roles with subordinates. It is a record-level access mechanism, separate from field-level security and permission set assignment.
The scenario describes a business outcome, not merely an administrative preference. If the feature is configured correctly, users get the intended result without depending on

memory, spreadsheets, or manual policing. That is the professional standard for Platform App Builder work: place the rule in metadata, test it under the correct permissions, and keep the design supportable.
The alternatives are plausible only if the requirement is read too narrowly. B (Private Groups) would be fragile here because it could produce an inconsistent result once different users, records, or apps are tested. C (Permission Set Groups) does not fit cleanly; Permission controls are essential for access, but they do not perform calculations or process automation. A complete implementation should also check page assignment, field access, record ownership, and mobile behavior where those factors apply.
For record sharing, object permission must exist first, and then sharing can extend record visibility to users, public groups, roles, or roles and subordinates as appropriate.

質問: **85**
アプリビルダーは、ルックアップで接続された子レコードが削除されたときに、親レコードのフィールドを更新したいと考えています。アプリビルダーはどの自動化を使用すべきでしょうか？
A. レコードトリガーフロー
B. 検証ルール
C. 自動起動フロー
D. クイックアクション
正解: **(**正解を表示します**)**
The right answer avoids a brittle workaround. The best answer is A. Record-triggered flow.
The selected option is not chosen because it sounds familiar; it is chosen because it gives Salesforce the right instruction at runtime. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars.
The scenario describes a business outcome, not merely an administrative preference. If the feature is configured correctly, users get the intended result without depending on memory, spreadsheets, or manual policing. That is the professional standard for Platform App Builder work: place the rule in metadata, test it under the correct permissions, and keep the design supportable.
None of the other options gives the same administrative control. B (Validation rule) is only a partial match because A validation rule blocks saves; it does not create records, route approvals, display components, or grant access. C (Autolaunched flow) misses the mark because Flow is broad, but it is not correct unless the requirement is specifically about guided or automated logic. D (Quick action) is not enough: Actions improve user entry, but they are not a substitute for relationship modeling, reporting logic, or back-end automation.
This is the stronger design because it respects security, data quality, user experience, and lifecycle management at the same time.

質問: **86**
アプリ開発者が組織内で利用可能なカスタム Lightning コンポーネントの一覧を確認するには、どの 2 つの場所にアクセスすればよいですか？
2つの回答を選択してください
A. セットアップのVisualforceコンポーネント
B. Salesforce Optimizer アプリ
C. Lightning App Builder
D. 設定のLightningコンポーネント
正解: **(正解を表示します)**
The important question is which feature owns the result. In this case, The proper configuration is C. Lightning App Builder; D. Lightning components in Setup. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars.
The stem gives enough detail to reject surface-level fixes. Salesforce should be configured so the behavior is enforced or rendered by the feature designed for it. When the requirement touches User Interface, the builder should avoid crossing into a different layer unless there is a real platform limitation. The selected answer follows that discipline and creates the least surprising result for admins and users.
The rejected choices solve different problems. A (Visualforce components in Setup) is less defensible because A Lightning component changes the interface; it does not automatically solve security, relationship, or data- quality requirements. B (Salesforce Optimizer App) should be rejected because it works at the wrong scope for the object, field, page, automation, or deployment condition in the stem. The configuration remains understandable for future admins because the decision is visible in metadata rather than buried in a workaround.
For Lightning pages, activation and assignment determine who sees the page. Component visibility then controls whether a specific component appears for a given condition.

質問: **87**
Northern Trail Outfitters社は、フィールドセールスチームには担当するアカウントのみを表示させたいと考えています。北米とヨーロッパのマーケティングチームは、それぞれ担当地域のアカウントのみを表示できるようにする必要があります。一方、インサイドセールスチームはSalesforce内のすべてのアカウントを表示できるようにする必要があります。これを実現するにはどうすればよいでしょうか？
A. アカウントの組織全体のデフォルト設定を 「公開」に設定します。各マーケティングチームのプロファイルを作成し、役割階層の最上位にインサイドセールスチームの役割を作成します。

B. アカウントの組織全体のデフォルト設定を「公開」に設定します。各マーケティングチームに対して条件ベースの共有ルールを作成し、アカウントに対して「すべて表示」設定のインサイドセールスチームの権限セットを作成します。
C. アカウントの組織全体のデフォルト設定を「非公開」に設定します。各マーケティングチームの権限セットを作成し、アカウントに対して「すべて表示」設定のインサイドセールスチームプロファイルを作成します。
D. アカウントの組織全体のデフォルト設定を「非公開」に設定します。各マーケティングチームに対して条件に基づいた共有ルールを作成し、アカウントに対して「すべて表示」設定のインサイドセールスチームプロファイルを作成します。
正解: **(**正解を表示します**)**
The answer is controlled by runtime behavior, not by naming similarity. Salesforce access is layered. Object permissions allow a user to work with a type of record, field-level security controls visibility and editability for individual fields, and a page layout only organizes what the user sees after permission has already been granted. The correct response is therefore D. Set the Organization Wide Default to Private for accounts.
Create criteria-based sharing rules for Each marketing team, and create an Inside Sales Team profile with the
" View All " setting for Accounts.
The scenario is not asking for a feature list. It is asking which configuration produces the requested result with the fewest moving parts. A Salesforce app builder should favor a standard, visible Setup configuration whenever it fully meets the business need. That approach makes the build easier to validate in sandbox and easier to explain during release review.
The other options fall short for clear platform reasons. A (Set the Organization-Wide Default to Public for accounts. Create profiles for each marketing team, and create an Inside Sales Team role that is at the top of the Role Hierarchy.) is not enough: it would require extra assumptions that the scenario does not support. B (Set the Organization Wide Default to Public for accounts. Create criteria-based sharing rules for Each marketing team, and create an Inside Sales Team permission set with the " View All " setting for Accounts.) handles a different concern; Sharing controls record visibility; it does not change field type, page composition, or automation timing. C (Set the Organization-Wide Default to Private for accounts. Create permission sets for each marketing Team, and create an Inside Sales Team profile with the " View All " setting for accounts.) is less defensible because Permission controls are essential for access, but they do not perform calculations or process automation. Before release, test the configuration with the affected profile and realistic records, not only with a system administrator account.

質問: **88**

DreamHouse Realtyは、Cloud Kicksの買収後、サンドボックスの利用戦略を見直しています。Salesforce COEは既に部分サンドボックスとフルサンドボックスを利用しており、それぞれ独自の定期スケジュールで更新しています。チームは拡大しており、コストを抑えつつ、共同テストのために大規模なサンドボックスプールに移行する前に、各小規模プロジェクトをサンドボックスで開始する必要があります。どのタイプのサンドボックスを最初に検討すべきでしょうか？
A. 開発者向けプロサンドボックス
B. 部分的なサンドボックス
C. 開発者サンドボックス
D. 完全なサンドボックス
正解**: A (**コメントを発表する**)**
Think in terms of supportability first. Choose A. Developer pro sandbox. In Salesforce terms, this is a sandbox strategy issue. Sandbox selection balances data copy, refresh cadence, isolation, testing depth, and cost. Smaller sandboxes fit independent build work; larger sandboxes fit integration and production-like testing.
The selected configuration is the professional answer because it is native, declarative, and placed at the right scope. If the requirement later expands, another administrator can inspect the same metadata area and understand how the behavior is being produced. That is far better than a solution that depends on hidden assumptions or user training.
The remaining answers do not control the same Salesforce behavior. B (Partial sandbox) does not fit cleanly; Deployment tooling moves or validates metadata; it does not create the runtime business behavior itself. C (Developer sandbox) is only a partial match because Deployment tooling moves or validates metadata; it does not create the runtime business behavior itself. D (Full sandbox) misses the mark because Deployment tooling moves or validates metadata; it does not create the runtime business behavior itself. The result is a cleaner app design because the behavior is handled by the feature Salesforce built for that purpose.
For sandboxes, the builder should match the environment to the work: isolated configuration, collaborative testing, data-dependent testing, or production rehearsal.

質問**: 89**
Universal Containersは、候補者選考プロセス中に面接官が生成した情報を記録するため、「Reviews」というカスタムオブジェクトを使用しています。Reviewレコードは、関連するカスタム候補者レコードへのアクセス権を持つすべてのユーザーが閲覧できます。人事担当副社長は、Reviewのコメント欄を人事部以外のユーザーには非公開にしたいと考えています。アプリ開発者はこの要件をどのように満たすべきでしょうか？
A. フィールドを含むページレイアウトを作成し、フィールドレベルのセキュリティを使用して、他のすべてのユーザーからフィールドを非表示にします。
B. 役割と部下とともに人事担当副社長とフィールドを共有する共有ルールを作成します。

C. 人事ユーザー向けにフィールドを含むページレイアウトを作成し、その他のすべてのユーザー向けにフィールドを含まない別のページレイアウトを作成します。
D. 役割に「HR」を持つユーザーとフィールドを共有するApex共有ルールを作成します。
正解: **(**正解を表示します**)**
Start at the layer that controls the outcome. The best answer is A. Create a page layout with the field and use field-level security to hide the field from all other users. The decision turns on Lightning page configuration.
Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars.
This is how the same requirement would be handled in a well-governed Salesforce org: identify the controlling object, relationship, field, page, automation, or environment first, and then configure the native capability for that control point. The selected answer satisfies the requirement without requiring users to perform extra interpretation or administrators to maintain an unnecessary custom workaround.
The distractors are not equal substitutes. B (Create a sharing rule to share the field with the VP of HR with Role and Subordinates.) is less defensible because Sharing controls record visibility; it does not change field type, page composition, or automation timing. C (Create a page layout with the field for HR users and another page layout without the field for all other users.) should be rejected because Page layouts affect presentation; they do not grant the underlying object or field capability by themselves. D (Create an Apex sharing rule to share the field with users that have " HR " in their role.) would be fragile here because Apex may solve advanced gaps, but it is not the first choice when a native declarative feature fully handles the requirement.
That is the kind of solution a Salesforce admin can document, migrate, and troubleshoot without creating hidden technical debt.

質問: **90**
Cloud Kicksには、プライベート共有設定を持つカスタムオブジェクトがあります。企業は、個々のレコードをケースバイケースで特定の人または部署と共有したいと考えています。ビジネスユーザーが個々のレコードを手動で共有するには、どの3つのオプションがありますか？3つ選択してください。
A. アクセス許可セットグループ
B. プライベートグループ
C. ユーザー
D. 公開グループ
E. 役割と部下
正解: **(**正解を表示します**)**

The right answer avoids a brittle workaround. The best answer is C. Users; D. Public Groups; E. Roles and Subordinates. Manual record sharing can grant access to individual users, public groups, and roles with subordinates. It is a record-level access mechanism, separate from field-level security and permission set assignment.
The selected answer is stronger because it meets both the functional requirement and the maintainability requirement. In a Salesforce implementation, a solution that only works for one user, one page, or one data sample is not good enough. The correct feature has to behave consistently across the intended profiles, record types, apps, and devices that are part of the design.
None of the other options gives the same administrative control. A (Permission Set Groups) does not fit cleanly; Permission controls are essential for access, but they do not perform calculations or process automation. B (Private Groups) is only a partial match because it does not provide the same direct administrative control as the selected option.
This is the stronger design because it respects security, data quality, user experience, and lifecycle management at the same time.
For record sharing, object permission must exist first, and then sharing can extend record visibility to users, public groups, roles, or roles and subordinates as appropriate.

質問**: 91**
Universal Containers (UC) は大量のデータを保有しており、データストレージの上限に近づいています。計画されている解決策は、Salesforce のデータストレージを削減するために履歴データをアーカイブすることですが、UC はアーカイブされた情報に対してもレポート、クエリ、ルックアップを使用したいと考えています。この要件を満たすことができるオプションはどれですか？ 2 つ選択してください。
A. 関連オブジェクト
B. カスタムオブジェクト
C. 外部オブジェクト
D. 大きな物体
正解**: (**正解を表示します**)**
A clean build puts the rule where Salesforce evaluates it. Choose C. External objects; D. Big objects. In Salesforce terms, this is a platform configuration issue. The Platform App Builder role is to choose the standard Salesforce capability that owns the behavior: data structure, access, automation, user experience, reporting, or deployment.
The selected configuration is the professional answer because it is native, declarative, and placed at the right scope. If the requirement later expands, another administrator can inspect the same metadata area and understand how the behavior is being produced. That is far better than a solution that depends on hidden assumptions or user training.
The remaining answers do not control the same Salesforce behavior. A (Related objects) would be fragile here because it could produce an inconsistent result once different users,

records, or apps are tested. B (Custom objects) does not fit cleanly; it solves a symptom of the problem rather than the Salesforce mechanism that owns the result. The result is a cleaner app design because the behavior is handled by the feature Salesforce built for that purpose.
For general platform design, the strongest answer is the native feature that can be enforced, audited, and maintained through Salesforce metadata.
For general platform design, the strongest answer is the native feature that can be enforced, audited, and maintained through Salesforce metadata.

質問: **92**
Cloud Kicks(CK)の営業担当者は、成約前に注文内容の確認が必要な場合でも、承認申請を忘れてしまうことがあります。CKは、手動申請をなくすために、商談をセキュアコミットメントステージに自動的に送信したいと考えています。どの機能がビジネス要件を満たしますか？
A. レコードトリガーフローを高速フィールド更新用に最適化
B. カスタムボタンと画面フロー
C. プラットフォームイベントトリガーフロー
D. アクションと関連レコード向けに最適化されたレコードトリガーフロー
正解: **(正解を表示します)**
Think in terms of supportability first. In this case, Choose D. Record-Triggered flow optimized for Actions and Related Records. Record-triggered automation should be designed so administrators can predict order, troubleshoot outcomes, and maintain rules without scattering logic across many overlapping tools.
The stem gives enough detail to reject surface-level fixes. Salesforce should be configured so the behavior is enforced or rendered by the feature designed for it. When the requirement touches Business Logic and Process Automation, the builder should avoid crossing into a different layer unless there is a real platform limitation. The selected answer follows that discipline and creates the least surprising result for admins and users.

The remaining answers do not control the same Salesforce behavior. A (Record-Triggered flow optimized for Fast Field Updates) would be fragile here because Flow is broad, but it is not correct unless the requirement is specifically about guided or automated logic. B (Custom button and screen flow) does not fit cleanly; Flow is broad, but it is not correct unless the requirement is specifically about guided or automated logic. C (Platform Event-Triggered flow) is only a partial match because Flow is broad, but it is not correct unless the requirement is specifically about guided or automated logic. The result is a cleaner app design because the behavior is handled by the feature Salesforce built for that purpose.

質問: **93**
Universal Containers社のマネージャーは、親アカウントレコードから階層構造を形成するための追加アカウントを迅速に作成する方法を求めています。ユーザーが子アカウントを素早く作成できるように、親アカウントに基づいて5つのフィールドを自動入力したいと考えています。アプリ開発者はどのような提案をすべきでしょうか？
A. グローバルクイックアクションをカスタマイズする
B. アカウント階層にパスを追加
C. アカウントにカスタムリンクを追加する
D. カスタムアクションを作成する
正解: **(正解を表示します)**
A clean build puts the rule where Salesforce evaluates it. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, action regions, activity surfaces, and utility bars. The correct response is therefore D. Create a custom action.
The scenario is not asking for a feature list. It is asking which configuration produces the requested result with the fewest moving parts. A Salesforce app builder should favor a standard, visible Setup configuration whenever it fully meets the business need. That approach makes the build easier to validate in sandbox and easier to explain during release review.
The remaining answers do not control the same Salesforce behavior. A (Customize a Global Quick Action) would be fragile here because Actions improve user entry, but they are not a substitute for relationship modeling, reporting logic, or back-end automation. B (Add Path on Account hierarchy) does not fit cleanly; it solves a symptom of the problem rather than the Salesforce mechanism that owns the result. C (Add a custom link on Account) is only a partial match because it does not provide the same direct administrative control as the selected option. The result is a cleaner app design because the behavior is handled by the feature Salesforce built for that purpose.

質問: **94**

Universal Containersでは、Projectsというカスタムオブジェクトを使用しています。マネージャーがプロジェクトを割り当てる際、プロジェクトレコードに 推定時間」というカスタムフィールドを設定します。一度設定されると、ユーザーはその値を減らすことはできますが、増やすことはできません。アプリ開発者はこの要件をどのように満たすことができるでしょうか？
A. ISCHANGED 関数を使用する検証ルールを作成します。
B. カスタムフィールドの数式デフォルト値を作成します
C. PRIORVALUE 関数を使用する検証ルールを作成します。
D. PREVGROUPVAL 関数を使用する数式項目を作成します。
正解**: (**正解を表示します**)**
This scenario should be solved with standard metadata. Validation rules prevent bad data from being saved.
They are appropriate when the requirement is to enforce a condition at save time rather than merely warn, display, or report on the problem later. Use C. Create a validation rule that uses the PRIORVALUE function.
A clean Platform App Builder answer should satisfy the complete scenario, not merely one phrase in it. This item requires the app builder to choose the configuration that produces the requested behavior in normal Salesforce use. The selected answer fits that requirement because it works with the platform model instead of fighting it.
The alternatives are plausible only if the requirement is read too narrowly. A (Create a validation rule that uses the ISCHANGED function.) should be rejected because A validation rule blocks saves; it does not create records, route approvals, display components, or grant access. B (Create a formula default value for the custom fiel) would be fragile here because A formula is calculated display logic and cannot always store static values or perform record updates. D (Create a formula field that uses the PREVGROUPVAL function.) is only a partial match because A formula is calculated display logic and cannot always store static values or perform record updates. A complete implementation should also check page assignment, field access, record ownership, and mobile behavior where those factors apply.

質問**: 95**
Universal Containers社は、最新の広告購入に対する投資対効果を把握したいと考えています。同社は現在、すべてのオブジェクトに対してプライベートセキュリティモデルを採用しています。アプリ開発者はどのような提案をすべきでしょうか？
A. 機会パイプラインレポートについて
B. オンラインアカウント階層とRaft Upサマリーフィールド
C. キャンペーン階層とキャンペーンスタスミックスの設定
D. 公共セキュリティモデルへの変更
正解**: C (**コメントを発表する**)**

The key is to separate behavior from appearance. The best answer is C. Configure Campaign Hierarchies and Campaign stasmics. The Platform App Builder role is to choose the standard Salesforce capability that owns the behavior: data structure, access, automation, user experience, reporting, or deployment.
The selected answer is stronger because it meets both the functional requirement and the maintainability requirement. In a Salesforce implementation, a solution that only works for one user, one page, or one data sample is not good enough. The correct feature has to behave consistently across the intended profiles, record types, apps, and devices that are part of the design.
None of the other options gives the same administrative control. A (On an opportunities pipeline report) does not fit cleanly; Reporting configuration helps analysis, but it does not enforce the requested record behavior.
B (Online Account Hierarchies and Raft Up Summary fields) is only a partial match because it does not provide the same direct administrative control as the selected option.
D (Change to a public security model) is not enough: it would require extra assumptions that the scenario does not support. This is the stronger design because it respects security, data quality, user experience, and lifecycle management at the same time.

質問**: 96**
Cloud Kicksは、デスクトップ版の商談レコード上部の別セクションに、12個のキーフィールドを一度に表示したいと考えています。この機能を有効にするには、アプリビルダーはレコードページにどのコンポーネントを追加すればよいでしょうか？
A. カスタム Lightning Web コンポーネント
B. ハイライトパネル
C. パス
D. アコーディオン
正解**: (**正解を表示します**)**
This scenario should be solved with standard metadata. Use B. Highlights Panel. The selected option is not chosen because it sounds familiar; it is chosen because it gives Salesforce the right instruction at runtime.
Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, action regions, activity surfaces, and utility bars.
The scenario describes a business outcome, not merely an administrative preference. If the feature is configured correctly, users get the intended result without depending on memory, spreadsheets, or manual policing. That is the professional standard for Platform App Builder work: place the rule in metadata, test it under the correct permissions, and keep the design supportable.
The alternatives are plausible only if the requirement is read too narrowly. A (Custom Lightning Web Component) should be rejected because A Lightning component changes

the interface; it does not automatically solve security, relationship, or data-quality requirements. C (Path) does not fit cleanly; it solves a symptom of the problem rather than the Salesforce mechanism that owns the result. D (Accordion) is only a partial match because it does not provide the same direct administrative control as the selected option. A complete implementation should also check page assignment, field access, record ownership, and mobile behavior where those factors apply.

質問: **97**
Cloud Kicks (CK) は、Salesforce 内で顧客アカウントに関するさまざまなトピックについて話し合うための交流やコラボレーションを開始したいと考えています。CK は、すべての従業員がこれらの会話を閲覧できるようにしたいと考えています。Chatter のどの 2 つの機能が CK のビジネスニーズを満たすでしょうか。2 つの回答を選択してください。
A. 新しい公開 Chatter グループを設定します。
B. 新しいプライベート Chatter グループを設定します。
C. Chatterアクションを使用して、完了するタスクを作成します。
D. Account オブジェクトに対して post アクションを使用します。
正解: **(正解を表示します)**
The answer is controlled by runtime behavior, not by naming similarity. Choose A. Set up new public Chatter groups.; D. Use post action on the Account object. The requirement in the stem is: Cloud Kicks (CK) wants to begin socializing and collaborating within Salesforce around Customer accounts to discuss various topics. CK would like all company employees to see these Conversations. Which two features of. The selected option addresses that requirement at the point where Salesforce actually evaluates the behavior.
The underlying concept is Chatter collaboration. Chatter supports record-centered collaboration. Feed tracking surfaces selected field changes, while groups, streams, and post actions organize discussions and follow-up work. That is why this answer is more defensible than a workaround. It can be reviewed in Setup, included in the correct deployment path, and tested by impersonating or logging in as a representative user where appropriate.
The other options fall short for clear platform reasons. B (Set up new private Chatter groups.) handles a different concern; it handles only part of the requirement and leaves the controlling Salesforce behavior unresolved. C (Use Chatter actions to create tasks to complete.) is less defensible because it is not the most maintainable declarative control for this design. Before release, test the configuration with the affected profile and realistic records, not only with a system administrator account.

質問: **98**

取引が成立した場合、四半期ごとの販売競争のリーダーボードに追加される前に、オーナーのマネージャーの承認を得る必要があります。四半期の最終日に商談が成立しましたが、マネージャーは休暇中です。適切な取引がすべて審査され、リーダーボードが最新の状態に保たれるようにするには、どのような対応が推奨されますか？
A. マネージャーのアシスタントに承認依頼を再割り当てしてもらいます。
B. クイックアクションを使用して、承認リクエストを次のレベルの承認者に転送します。
C. マネージャーの委任承認者を設定します。
D. 割り当てルールを使用して、委任された承認者を自動的に割り当てます。
正解: **(**正解を表示します**)**
Start at the layer that controls the outcome. The best answer is C. Set up a delegated approver for the manager. The selected option is not chosen because it sounds familiar; it is chosen because it gives Salesforce the right instruction at runtime. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars.
The scenario describes a business outcome, not merely an administrative preference. If the feature is configured correctly, users get the intended result without depending on memory, spreadsheets, or manual policing. That is the professional standard for Platform App Builder work: place the rule in metadata, test it under the correct permissions, and keep the design supportable.
The distractors are not equal substitutes. A (Have the manager ' s assistant reassign the approval request.) handles a different concern; An approval process is for formal sign-off, not for every automation, display, import, or security requirement. B (Use a quick action to forward the approval request to the next level approver.) is less defensible because An approval process is for formal sign-off, not for every automation, display, import, or security requirement. D (Use an assignment rule to automatically assign a delegated approver.) would be fragile here because it could produce an inconsistent result once different users, records, or apps are tested. That is the kind of solution a Salesforce admin can document, migrate, and troubleshoot without creating hidden technical debt.

質問: **99**
マーケティングディレクターは、昨年、自動車部品を無料で配布しすぎたことを懸念しています。彼らは、営業担当者による割引、サンプル、および無料配布を追跡したいと考えています。無料部品に対するマネージャーの承認も処理する必要があります。これらの要件を満たすために、アプリビルダーはどの機能を使用すべきでしょうか？
A. 高回転販売
B. 引用
C. 承認プロセス
D. チェックボックス
正解: **C (**コメントを発表する**)**

A production admin would not solve this by decoration. Choose C. Approval process. The requirement in the stem is: The marketing director is concerned that too many car parts were given away for free last year. They want to track discounts, samples, and Giveaways by Sales Rep. The manager ' s sign-off on free parts also needs to be. The selected option addresses that requirement at the point where Salesforce actually evaluates the behavior. The underlying concept is approval governance. Approval processes are designed for formal review. They route a request, capture a decision, and can lock or update the record through submission and final actions.
That is why this answer is more defensible than a workaround. It can be reviewed in Setup, included in the correct deployment path, and tested by impersonating or logging in as a representative user where appropriate.
The remaining answers do not control the same Salesforce behavior. A (High Velocity Sales) would be fragile here because it could produce an inconsistent result once different users, records, or apps are tested. B (Quotes) does not fit cleanly; it solves a symptom of the problem rather than the Salesforce mechanism that owns the result. D (Check box) misses the mark because it addresses a neighboring feature area rather than the exact behavior requested. The result is a cleaner app design because the behavior is handled by the feature Salesforce built for that purpose.

質問: **100**
Universal Containers社は、 セミナー」と 参加者」という2つのカスタムオブジェクトを作成しました。これらのオブジェクトの組織全体のデフォルト設定は 非公開」となっています。Universal Containers社は、これらのカスタムオブジェクト間に新しい結合オブジェクトを設定したいと考えています。結合オブジェクト内のレコードは、特定のユーザーグループのみが編集できるようにする必要があります。
アプリ開発者が適切なセキュリティを設定するために取るべき2つの手順はどれですか？ (2つ選択してください)
A. マスターオブジェクトへの読み取りアクセスを許可する所有者ベースの共有ルールを作成します。
B. 両方の結合オブジェクト関係フィールドにルックアップフィルタを設定します。
C. マスター/ディテール関係フィールドの両方で共有設定を読み取り専用に設定します。
D. ジャンクション オブジェクトへの Read アクセスを許可する所有者ベースの共有ルールを作成します。
正解: **(**正解を表示します**)**
The answer is controlled by runtime behavior, not by naming similarity. Choose A. Create owner-based sharing rules that give Read access to the master objects.; C. Set Sharing Settings to Read Only on both Master-Detail relationship fields. The requirement in the stem is: Universal Containers has created two custom objects called Seminars and Attendees. Organization-wide defaults for these objects have been set to Private.

Universal Containers wants to set up a new junction object. The selected option addresses that requirement at the point where Salesforce actually evaluates the behavior.
The underlying concept is record sharing targets. Manual record sharing can grant access to individual users, public groups, and roles with subordinates. It is a record-level access mechanism, separate from field-level security and permission set assignment. That is why this answer is more defensible than a workaround. It can be reviewed in Setup, included in the correct deployment path, and tested by impersonating or logging in as a representative user where appropriate.
The other options fall short for clear platform reasons. B (Set lookup filters on both junction object relationship fields.) handles a different concern; A lookup relationship is more loosely coupled and does not provide every master-detail behavior, especially native roll-up behavior. D (Create an owner-based sharing rule that gives Read access to the junction object.) should be rejected because Sharing controls record visibility; it does not change field type, page composition, or automation timing. Before release, test the configuration with the affected profile and realistic records, not only with a system administrator account.

質問**: 101**
Northern Trail Outfittersのアプリ開発者が、プロジェクト管理アプリのフローを再構成するために、アカウント、プロジェクト、およびプロジェクトマイルストーン用のサンドボックステンプレートを作成しました。アプリ開発者は、どのような種類のテスト環境を作成すべきでしょうか？
A. 部分コピー
B. スクラッチ組織
C. 開発者
D. デベロッパープロ
正解**: A (**コメントを発表する**)**
Treat the requirement as something users will depend on daily. The proper configuration is A. Partial Copy.
The requirement in the stem is: An app builder at Northern Trail Outfitters created a sandbox template for Accounts, Projects, and Project Milestones to reconfigure some flows for the project management app. Which type of testing environment should. The selected option addresses that requirement at the point where Salesforce actually evaluates the behavior.
The underlying concept is sandbox strategy. Sandbox selection balances data copy, refresh cadence, isolation, testing depth, and cost. Smaller sandboxes fit independent build work; larger sandboxes fit integration and production-like testing. That is why this answer is more defensible than a workaround. It can be reviewed in Setup, included in the correct deployment path, and tested by impersonating or logging in as a representative user where appropriate.

The rejected choices solve different problems. B (Scratch Org) should be rejected because it works at the wrong scope for the object, field, page, automation, or deployment condition in the stem. C (Developer) would be fragile here because it could produce an inconsistent result once different users, records, or apps are tested. D (Developer Pro) does not fit cleanly; it solves a symptom of the problem rather than the Salesforce mechanism that owns the result. The configuration remains understandable for future admins because the decision is visible in metadata rather than buried in a workaround.

質問**: 102**
Universal Containers社は、関連するすべての商談を段階別にまとめたチャートをアカウント詳細ページに埋め込みたいと考えています。アプリ開発者は、アカウントページのレイアウトに追加するために、どのタイプのレポートを作成すればよいでしょうか？
A. 商談オブジェクトに関する表形式のレポート
B. アカウントオブジェクトに関する表形式のレポート
C. 商談オブジェクトの概要レポート
D. アカウントオブジェクトの概要レポート
正解**: C (**コメントを発表する**)**
The answer comes from Salesforce's separation of responsibilities. Use C. A summary report on the Opportunity object. The decision turns on embedded reporting. Report chart components depend on an accessible source report and a chart-capable report format. Private reports and reports without charts cannot support the same embedded record-page experience.
This is how the same requirement would be handled in a well-governed Salesforce org: identify the controlling object, relationship, field, page, automation, or environment first, and then configure the native capability for that control point. The selected answer satisfies the requirement without requiring users to perform extra interpretation or administrators to maintain an unnecessary custom workaround.
The alternatives are plausible only if the requirement is read too narrowly. A (A tabular report on the Opportunity object) should be rejected because Reporting configuration helps analysis, but it does not enforce the requested record behavior. B (A tabular report on the Account object) would be fragile here because Reporting configuration helps analysis, but it does not enforce the requested record behavior. D (A summary report on the Account object) is only a partial match because Reporting configuration helps analysis, but it does not enforce the requested record behavior. A complete implementation should also check page assignment, field access, record ownership, and mobile behavior where those factors apply.

質問**: 103**

管理対象外パッケージに含まれる応募者オブジェクトに新しいフィールドが追加されました。採用担当者が応募者レポートの有無にかかわらずポジションを実行したところ、新しいフィールドが列として追加するオプションに表示されないことに気づきました。アプリビルダーはこの問題をどのようにトラブルシューティングすればよいでしょうか？
A. カスタムレポートタイプのフィールドレイアウトにフィールドを追加します。
B. レポートタイプに含めるフィールドレベルのセキュリティを調整します。
C. ポジションと応募者オブジェクトのレポートを許可するにチェックを入れます。
D. 公開レポートの管理権限を持つプロファイルを更新します。
正解: **(**正解を表示します**)**
The answer comes from Salesforce's separation of responsibilities. Managed packages support controlled distribution and upgrades, while unmanaged packages copy editable metadata into the subscriber org and do not follow the same upgrade model. Use A. Add the field to the custom report type field layout.
A clean Platform App Builder answer should satisfy the complete scenario, not merely one phrase in it. This item requires the app builder to choose the configuration that produces the requested behavior in normal Salesforce use. The selected answer fits that requirement because it works with the platform model instead of fighting it.
The alternatives are plausible only if the requirement is read too narrowly. B (Adjust the field level security to include in the report type.) would be fragile here because Reporting configuration helps analysis, but it does not enforce the requested record behavior. C (Check Allow Reports for the position and applicant objects.) does not fit cleanly; Reporting configuration helps analysis, but it does not enforce the requested record behavior. D (Update the profile with the Manage Public Reports permission.) is only a partial match because Reporting configuration helps analysis, but it does not enforce the requested record behavior. A complete implementation should also check page assignment, field access, record ownership, and mobile behavior where those factors apply.

質問**: 104**
Cloud Kicks(CK)は、アカウントレコードページで顧客のサポートレベルを追跡しています。CKは、関連アカウントがプラチナレベルの顧客である場合に、ケースレコードページにテキスト通知を表示したいと考えています。アプリ開発者はこの要件をどのように満たすことができますか？
A. Case Lightning ページを複製します。新しいページにリッチテキスト コンポーネントを追加し、このページをプラチナ アカウントに割り当てます。
B. ケース Lightning ページにリッチ テキスト コンポーネントを追加します。アカウント サポート レベルがプラチナの場合に、リッチ テキスト コンポーネントの表示を設定します。テキストのみのカスタム Lightning Web コンポーネントを作成します。カスタム Lightning Web コンポーネントをケース ページ レイアウトにドラッグします。

C. アカウントのサポートレベルがプラチナの場合に表示するように設定します。テキストのみのカスタム Lightning Web コンポーネントを作成します。ケースページのレイアウトを複製します。
D. カスタム Lightning Web コンポーネントをページにドラッグし、レイアウトを platinum ケースに割り当てます。
正解**: (**正解を表示します**)**
This is a governance decision before it is a configuration click. The proper configuration is B. Add a rich text component to the Case Lightning page Set the component visibility of the rich text component to when the account support level is Platinum. Create a text-only custom Lightning Web Component Drag the custom Lightning Web Component into the Case page layout. In Salesforce terms, this is a Lightning page configuration issue. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars.
The selected configuration is the professional answer because it is native, declarative, and placed at the right scope. If the requirement later expands, another administrator can inspect the same metadata area and understand how the behavior is being produced. That is far better than a solution that depends on hidden assumptions or user training.
The rejected choices solve different problems. A (Clone the Case Lightning page. Add a rich text component to the new page, and assign this page to platinum accounts.) is less defensible because A Lightning component changes the interface; it does not automatically solve security, relationship, or data-quality requirements. C (Set its visibility to when the account support level is platinum. Create a text-only custom Lightning Web Component. Clone the case page layout.) would be fragile here because Page layouts affect presentation; they do not grant the underlying object or field capability by themselves. D (Drag the custom Lightning Web Component into the page, and assign the layout to platinum cases.) does not fit cleanly; A Lightning component changes the interface; it does not automatically solve security, relationship, or data- quality requirements. The configuration remains understandable for future admins because the decision is visible in metadata rather than buried in a workaround.

質問**: 105**
Universal Containersには、モバイルアプリとデスクトップの両方をサポートするLightningレコードページがあります。アプリ開発者がAppExchangeからカスタムLightningコンポーネントをダウンロードしましたが、ユーザーはモバイルデバイスでそのコンポーネントを表示できません。
A. コンポーネントをアクティブ化する必要があります
B. レコードページテンプレートはモバイルデバイスをサポートしていません。
C. このコンポーネントはデスクトップページ用に開発されました。

D. レコードページをアクティブ化する必要があります
正解: **C (**コメントを発表する**)**
The design should be easy for another admin to maintain. A Lightning component must be exposed for the surface where it is used. A component built for one page type or action surface does not automatically become available everywhere in Lightning Experience. The correct response is therefore C. The component has been developed for Desktop Pages. The scenario is not asking for a feature list. It is asking which configuration produces the requested result with the fewest moving parts. A Salesforce app builder should favor a standard, visible Setup configuration whenever it fully meets the business need. That approach makes the build easier to validate in sandbox and easier to explain during release review.
The wrong answers confuse adjacent platform concepts. A (The component needs to be activate) is only a partial match because A Lightning component changes the interface; it does not automatically solve security, relationship, or data-quality requirements. B (The record page template is unable to support mobile devices.) misses the mark because it addresses a neighboring feature area rather than the exact behavior requested. D (The record page needs to be activate) handles a different concern; it handles only part of the requirement and leaves the controlling Salesforce behavior unresolved. This keeps the build declarative, traceable in Setup, and easier to support when the org changes later.

質問: **106**
Cloud Kicksは5年分の販売データを保有しており、顧客が初めて購入した時期を追跡したいと考えています。
アプリ開発者は、要件を満たすためにロールアップサマリーをどのように活用すべきでしょうか？
A. First Order Date という名前の新しい日付フィールドを作成し、次に Type MIN を使用してフィールドを更新するロールアップ サマリーを作成します。
B. 商談完了日に対して、iswon = TRUE のフィルターを使用して、タイプ MIN の 「初回注文日」という新しいロールアップ集計フィールドを作成します。
C. 「初回注文日」という新しい日付フィールドを作成し、「商談完了日」のロールアップ集計を使用して日付を設定します。
D. 商談完了日に対して、SUM 型を使用して 「初回注文日」という名前の新しいロールアップ集計項目を作成します。
正解: **(**正解を表示します**)**
A production admin would not solve this by decoration. Choose B. Create a new roll-up summary field called First Order Date, Using type MIN on the Opportunity Close Date with a filter where iswon = TRUE. In Salesforce terms, this is a roll-up summary design issue. Roll-up summary fields calculate values from child detail records and store the result on

the master. The relationship type and field type decide whether the calculation is available declaratively.
The selected configuration is the professional answer because it is native, declarative, and placed at the right scope. If the requirement later expands, another administrator can inspect the same metadata area and understand how the behavior is being produced. That is far better than a solution that depends on hidden assumptions or user training.
The remaining answers do not control the same Salesforce behavior. A (Create a new date field called First Order Date, then create roll-up summary to update the field using Type MIN.) would be fragile here because it could produce an inconsistent result once different users, records, or apps are tested. C (Create a new date field called First Order Date, then set the date using a roll-up summary on Opportunity Close Date.) is only a partial match because it does not provide the same direct administrative control as the selected option. D (Create a new roll-up summary field called First Order Date using type SUM on Opportunity Close Date.) misses the mark because it addresses a neighboring feature area rather than the exact behavior requested. The result is a cleaner app design because the behavior is handled by the feature Salesforce built for that purpose.

質問: **107**
アプリ開発者は、一部のカスタムフィールドのデータ型を変更する必要があります。カスタムフィールドのデータ型を変更する際に、アプリ開発者が注意すべき2つの制限事項は何ですか？ 2つ選択してください。
A. 数式フィールドのデータ型を任意のデータ型に変更することはできません。
B. Apex コードで参照されているフィールドのデータ型を変更することはできません。
C. 外部IDとして使用されるフィールドのデータ型を数値からテキストに変更することはできません。
D. テキストエリア(長文)フィールドのデータ型をテキストに変更することはできません。
正解: **(**正解を表示します**)**
The key is to separate behavior from appearance. The best answer is A. It is not possible to change the data type of a formula field to any data type.; B. It is not possible to change

the data type of field referenced by Apex code. The selected option is not chosen because it sounds familiar; it is chosen because it gives Salesforce the right instruction at runtime. Formula and validation expressions must respect Salesforce data types. Picklists require picklist-aware handling, dates require date functions or date arithmetic, and image or text formulas must return the correct type.
The scenario describes a business outcome, not merely an administrative preference. If the feature is configured correctly, users get the intended result without depending on memory, spreadsheets, or manual policing. That is the professional standard for Platform App Builder work: place the rule in metadata, test it under the correct permissions, and keep the design supportable.
None of the other options gives the same administrative control. C (It is not possible to change the data type of a field used as an External ID from number to Text.) misses the mark because That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option. D (It is not possible to change the data type of a Text Area (Long) field to Text.) is not enough: That field type either stores the wrong kind of value or fails to support the required behavior as cleanly as the selected option. This is the stronger design because it respects security, data quality, user experience, and lifecycle management at the same time.

質問: **108**
営業チームは、マーケティングチームから毎朝約800件のリードリストを受け取ります。マーケティングチームは、これらのリードの中に現在パイプラインに入っているものがあるかどうかを把握しておらず、毎朝リスト全体を送信します。重複リードが挿入されないようにしながら、これらのリードをSalesforceにインポートするには、どのツールを使用すべきでしょうか？
A. データローダー.io
B. データインポートウィザード
C. データローダー
D. 手動入力
正解: **(**正解を表示します**)**
The answer comes from Salesforce's separation of responsibilities. Salesforce import tools differ by data volume, matching logic, and operation. Data Import Wizard is guided and duplicate-aware for supported objects; Data Loader is stronger for larger updates, upserts, and clearing values. Use B. Data Import Wizard.
A clean Platform App Builder answer should satisfy the complete scenario, not merely one phrase in it. This item requires the app builder to choose the configuration that produces the requested behavior in normal Salesforce use. The selected answer fits that requirement because it works with the platform model instead of fighting it.
The alternatives are plausible only if the requirement is read too narrowly. A (Data Loader.io) should be rejected because Data Loader is an administrative data tool; it does

not replace metadata configuration or user- interface design. C (Data Loader) does not fit cleanly; Data Loader is an administrative data tool; it does not replace metadata configuration or user-interface design. D (Manual entry) is only a partial match because it does not provide the same direct administrative control as the selected option. A complete implementation should also check page assignment, field access, record ownership, and mobile behavior where those factors apply.
For imports, the builder should choose the tool based on record count, supported object, operation type, duplicate handling, and the presence of an external identifier.

質問: **109**
アプリ開発者は、新規商談を作成する際にユーザーに表示されるフィールド数を制限したいと考えています。ユーザーが必須フィールドに入力して保存すると、商談タイプに関連する追加フィールドを含む完全なレコードページが表示されるようにしたいのです。これを実現するにはどうすればよいでしょうか？
A. 商談タイプを最初の商談ページレイアウトの必須フィールドにし、自動化を使用して商談タイプに基づいてレコードタイプを更新します。
B. ユーザープロファイルに基づいて、商談タイプごとに異なるページレイアウトを使用します。
C. 必須フィールドが入力されたら、共有ルールを使用して新しいフィールドをユーザーと共有します。
D. ページレイアウト上の追加セクションを非表示にし、非表示のセクションのフィールドに入力したいときにユーザーが手動で展開する方法を示します。
正解: **A (**コメントを発表する**)**
A clean build puts the rule where Salesforce evaluates it. Choose A. Make the Opportunity type a required field on the initial Opportunity page layout and Use automation to update the record type based on the Opportunity type. The requirement in the stem is: An app builder wants to limit the amount of fields users see When creating a new Opportunity. Once they fill out the required fields and Save, the full record page with additional fields relevant to the Opportunity Type. The selected option addresses that requirement at the point where Salesforce actually evaluates the behavior.
The underlying concept is Lightning page configuration. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars. That is why this answer is more defensible than a workaround. It can be reviewed in Setup, included in the correct deployment path, and tested by impersonating or logging in as a representative user where appropriate.
The remaining answers do not control the same Salesforce behavior. B (Use different page layouts for Opportunity types based on the user profile.) does not fit cleanly; Page layouts affect presentation; they do not grant the underlying object or field capability by

themselves. C (Once the required fields are populated, use a sharing rule to share the new fields with The user.) is only a partial match because Sharing controls record visibility; it does not change field type, page composition, or automation timing. D (Hide additional sections on the page layout and the users how to manually Expand them when they want to fill in the fields in the hidden sections.) misses the mark because Page layouts affect presentation; they do not grant the underlying object or field capability by themselves. The result is a cleaner app design because the behavior is handled by the feature Salesforce built for that purpose.

質問**: 110**

AW Computingでは、候補者オブジェクトに 前職経験」のチェックボックスフィールドと 前職経験年数」の数値フィールドがあります。採用チームは、 前職経験」フィールドがチェックされている場合にのみ数値フィールドを表示したいと考えています。この要件を満たすために、アプリビルダーはどの機能を使用すべきでしょうか？

A. 2 種類の異なるページレイアウトを作成し、 以前の経験」がチェックされている場合にレイアウトを変更するプロセスを作成します。

B. [過去の経験] フィールドと [過去の経験年数] フィールドの間に依存関係を作成します。

C. 過去の経験」がチェックされている場合は、動的フォームを使用して 過去の経験年数」フィールドを表示します。

D. 候補ページレイアウトでVisualforceコンポーネントを使用して、条件付きでフィールドを表示します。

正解**: (**正解を表示します**)**

The useful clue is the business action being requested. Lightning App Builder controls page composition, component visibility, activation, assignments, and specialized surfaces such as record pages, home pages, and utility bars. The best answer is A. Create two different page layouts and a process to change the layout if Previous Experience is checkedd.; C. Use Dynamic Forms to display the Years of Previous Experience field if Previous Experience is checkedd.

A clean Platform App Builder answer should satisfy the complete scenario, not merely one phrase in it. This item requires the app builder to choose the configuration that produces the requested behavior in normal Salesforce use. The selected answer fits that requirement because it works with the platform model instead of fighting it.

None of the other options gives the same administrative control. B (Create a dependency between the Previous Experience and Years of Previous Experience fields.) is only a partial match because it does not provide the same direct administrative control as the selected option. D (Use a Visualforce component on the candidate page layout to conditionally display the fields.) is not enough: Page layouts affect presentation; they do not grant the underlying object or field capability by themselves. This is the stronger design

because it respects security, data quality, user experience, and lifecycle management at the same time.
