Test de Préparation: Data Engineering Avancé

Informations

- **Durée estimée:** 40 minutes
- **Nombre de questions:** 25
- **Niveau:** Avancé
- **Thèmes:** ETL/ELT, Qualité des données, Orchestration, Formats, Optimisation

---

Section 1: ETL vs ELT (5 questions)

Question 1

Quelle est la principale différence entre ETL et ELT?

A) ETL est plus rapide que ELT
B) Dans ETL, la transformation se fait avant le chargement; dans ELT, après
C) ETL est pour les petits volumes, ELT pour les grands
D) Il n'y a pas de différence

Réponse

B) Dans ETL, la transformation se fait avant le chargement; dans ELT, après

Approche	Flux	Avantage
ETL	Extract → **Transform** → Load	Données propres à l'arrivée
ELT	Extract → Load → **Transform**	Exploite la puissance du DWH cloud

ETL: Classique, on-premise, transformation sur serveur ETL ELT: Moderne, cloud, transformation dans le data warehouse (BigQuery, Snowflake)

Question 2

Dans quel scénario ELT est-il préférable à ETL?

A) Quand les données sont déjà propres
B) Quand on utilise un data warehouse cloud avec grande capacité de calcul
C) Quand on a très peu de données
D) Quand on n'a pas besoin de transformer les données

Réponse

B) Quand on utilise un data warehouse cloud avec grande capacité de calcul

ELT est préférable quand: - Data warehouse cloud puissant (BigQuery, Snowflake, Redshift) - Grands volumes de données - Besoin de flexibilité (schéma-on-read) - Transformation avec SQL (dbt)

ETL reste préférable quand: - Contraintes de conformité (données sensibles) - Logique de transformation complexe en Python/Java - Infrastructure on-premise

---

Question 3

Quel outil est spécialisé dans la transformation de données dans un entrepôt (paradigme ELT)?

A) Apache Spark
B) dbt (data build tool)
C) Talend
D) Pentaho

Réponse

B) dbt (data build tool)

dbt (data build tool): - Transformations SQL dans le data warehouse - Versionning avec Git - Tests de données intégrés - Documentation automatique - Lineage des données

```sql
-- models/staging/stg_clients.sql
SELECT
    id AS client_id,
    TRIM(nom) AS nom,
    DATE(created_at) AS date_creation
FROM {{ source('raw', 'clients') }}
WHERE deleted_at IS NULL
```

dbt compile et exécute les transformations directement dans le DWH.

Question 4

Qu'est-ce que le concept de "staging area" dans un pipeline de données?

A) La zone de production
B) Une zone intermédiaire où les données brutes sont stockées avant transformation
C) La base de données de backup
D) L'interface utilisateur

Réponse

B) Une zone intermédiaire où les données brutes sont stockées avant transformation

Architecture en couches (Medallion):

```
Sources → [Bronze/Raw] → [Silver/Staging] → [Gold/Mart] → Rapports
              ↓                ↓                ↓
          Données          Données          Données
          brutes           nettoyées        agrégées
```

La staging area permet: - Isolation des données brutes - Rejeu des transformations si erreur - Traçabilité et audit - Séparation des responsabilités

---

Question 5

Comment gérer un pipeline ETL qui doit s'exécuter après la fin d'un autre pipeline?

A) Exécuter manuellement
B) Utiliser un orchestrateur avec dépendances (Airflow, Prefect)
C) Mettre un timer fixe
D) Dupliquer le code

Réponse

B) Utiliser un orchestrateur avec dépendances (Airflow, Prefect)

Exemple Airflow:

```
from airflow import DAG
from airflow.operators.python import PythonOperator

with DAG('pipeline_bancaire', schedule_interval='@daily') as dag:

    extract_task = PythonOperator(
        task_id='extract_transactions',
        python_callable=extract_transactions
    )

    transform_task = PythonOperator(
        task_id='transform_transactions',
        python_callable=transform_transactions
    )

    load_task = PythonOperator(
        task_id='load_to_dwh',
        python_callable=load_to_dwh
    )

    # Définir les dépendances
    extract_task >> transform_task >> load_task
```

Section 2: Qualité des Données (5 questions)

Question 6

Quelles sont les dimensions principales de la qualité des données?

A) Taille, Couleur, Format
B) Exactitude, Complétude, Cohérence, Actualité
C) Vitesse, Volume, Variété
D) Input, Process, Output

Réponse

B) Exactitude, Complétude, Cohérence, Actualité

Dimensions de la qualité des données (ACCA):

Dimension	Question	Exemple bancaire
Exactitude	Les données sont-elles correctes?	Solde = vraie valeur
Complétude	Manque-t-il des données?	% valeurs manquantes
Cohérence	Les données sont-elles logiques?	Solde ≥ 0 pour épargne
Actualité	Les données sont-elles récentes?	Date dernière MAJ

Autres dimensions: Unicité, Validité, Conformité

Question 7

Comment implémenter une validation de règle métier “Le solde ne peut pas être négatif pour un compte épargne”?

A) Ne rien faire
B) Créer un test de qualité avec assertion
C) Supprimer les lignes concernées
D) Modifier manuellement les données

Réponse

B) Créer un test de qualité avec assertion

```python
def validate_solde_epargne(df):
    """
    Valide que les comptes épargne n'ont pas de solde négatif
    """
    violations = df[(df['type_compte'] == 'EPARGNE') & (df['solde'] < 0)]

    if len(violations) > 0:
        # Logger les violations
        print(f"⚠ {len(violations)} comptes épargne avec solde négatif")
        print(violations[['compte_id', 'solde']])

        # Option: lever une exception
        raise ValueError("Violation de règle métier: solde négatif sur épargne")

    return True

# Avec Great Expectations
expect_column_values_to_be_between(
    column="solde",
    min_value=0,
    row_condition='type_compte=="EPARGNE"'
)
```

Question 8

Quel outil open-source est spécialisé dans les tests de qualité des données?

A) Pandas
B) Great Expectations
C) Matplotlib
D) Flask

Réponse

B) Great Expectations

Great Expectations permet de: - Définir des “expectations” (règles de qualité) - Valider les données automatiquement - Générer des rapports de qualité - Intégrer dans les pipelines

```python
import great_expectations as gx

# Créer un contexte
context = gx.get_context()

# Définir des expectations
validator.expect_column_values_to_not_be_null("client_id")
validator.expect_column_values_to_be_between("age", 18, 120)
validator.expect_column_values_to_be_in_set(
    "segment", ["RETAIL", "PREMIUM", "PRIVATE"]
)

# Valider
results = validator.validate()
```

Question 9

Qu'est-ce que le "data profiling"?

A) La création de profils utilisateurs
B) L'analyse exploratoire automatique de la structure et qualité des données
C) La sécurisation des données
D) La compression des données

Réponse

B) L'analyse exploratoire automatique de la structure et qualité des données

Le data profiling génère automatiquement: - Types de données détectés - Statistiques (min, max, moyenne, distribution) - Taux de valeurs manquantes - Cardinalité (valeurs uniques) - Patterns détectés

```python
# Avec pandas-profiling (ydata-profiling)
from ydata_profiling import ProfileReport

profile = ProfileReport(df, title="Profil des Données Clients")
profile.to_file("rapport_profil.html")

# Informations générées:
# - Aperçu des colonnes
# - Alertes de qualité
# - Corrélations
# - Valeurs manquantes
# - Échantillons
```

Question 10

Comment détecter les doublons dans un dataset client?

A) Compter le nombre de lignes
B) Utiliser duplicated() ou group by sur les clés d'identification
C) Trier les données
D) Calculer la moyenne

Réponse

B) Utiliser duplicated() ou group by sur les clés d'identification

```python
# Méthode 1: duplicated()
doublons = df[df.duplicated(subset=['nom', 'date_naissance', 'telephone'], keep=False)]
print(f"Nombre de doublons: {len(doublons)}")

# Méthode 2: group by et count
doublons_group = df.groupby(['nom', 'date_naissance', 'telephone']).size()
doublons_group = doublons_group[doublons_group > 1]

# Méthode 3: Fuzzy matching pour doublons approximatifs
from fuzzywuzzy import fuzz
# Comparer les noms avec tolérance aux fautes de frappe
```

Les doublons peuvent causer: - Surévaluation du nombre de clients - Erreurs dans les agrégations - Problèmes de conformité (KYC)

Section 3: Formats et Stockage (5 questions)

Question 11

Quel format de fichier est le plus efficace pour le stockage analytique de grandes tables?

A) CSV
B) Parquet
C) JSON
D) Excel

Réponse

B) Parquet

Comparaison des formats:

Format	Type	Compression	Lecture colonnes	Usage
CSV	Texte	Faible	Non	Échange simple
JSON	Texte	Faible	Non	APIs, semi-structuré
Parquet	Binaire	Excellente	Oui	Analytics, Big Data
Avro	Binaire	Bonne	Non	Streaming

Avantages Parquet: - Stockage en colonnes (lecture sélective) - Compression efficace (jusqu'à 90%) - Schéma intégré - Compatible Spark, Pandas, BigQuery

---

Question 12

Qu'est-ce que le partitionnement dans un data lake?

A) Diviser le stockage entre plusieurs serveurs
B) Organiser les fichiers en sous-dossiers basés sur une colonne (ex: date)
C) Compresser les données
D) Crypter les fichiers

Réponse

B) Organiser les fichiers en sous-dossiers basés sur une colonne (ex: date)

Structure partitionnée:

```
data/
├── transactions/
│   ├── year=2024/
│   │   ├── month=01/
│   │   │   └── data.parquet
│   │   ├── month=02/
│   │   │   └── data.parquet
│   │   └── ...
│   └── year=2025/
│       └── ...
```

Avantages: - Lecture sélective (partition pruning) - Meilleure performance des requêtes filtrées - Gestion facilitée (archivage par période)

```
# Écriture partitionnée avec Pandas
df.to_parquet(
    'data/transactions',
    partition_cols=['year', 'month']
)
```

Question 13

Quelle stratégie de mise à jour utiliser pour un historique complet des soldes clients?

A) Écraser les données chaque jour
B) Slowly Changing Dimension (SCD) Type 2
C) Supprimer et recréer
D) Ignorer les changements

Réponse

B) Slowly Changing Dimension (SCD) Type 2

SCD Type 2 conserve l'historique complet:

client_id	solde	date_debut	date_fin	actuel
001	50000	2024-01-01	2024-03-15	0
001	75000	2024-03-16	2024-06-30	0
001	100000	2024-07-01	9999-12-31	1

Types SCD: - **Type 1:** Écraser (pas d'historique) - **Type 2:** Nouvelle ligne avec dates (historique complet) - **Type 3:** Nouvelle colonne (historique limité)

Bancaire: SCD2 essentiel pour audit et conformité.

Question 14

Qu'est-ce que le CDC (Change Data Capture)?

A) Un format de fichier
B) Une technique pour capturer uniquement les changements dans les données sources
C) Un type de base de données
D) Un outil de visualisation

Réponse

B) Une technique pour capturer uniquement les changements dans les données sources

CDC capture: - Insertions (nouvelles lignes) - Mises à jour (modifications) - Suppressions

Avantages: - Réduction du volume transféré - Latence plus faible - Moins de charge sur la source

Outils: Debezium, AWS DMS, Fivetran

```
Source DB → [CDC] → Message Queue → [Consumer] → Data Lake
              ↓
         Capture des
         changements
         en temps réel
```

Question 15

Comment optimiser une requête BigQuery qui scanne trop de données?

A) Ajouter plus de colonnes dans le SELECT
B) Utiliser le partitionnement et le clustering
C) Supprimer les index
D) Augmenter le timeout

Réponse

B) Utiliser le partitionnement et le clustering

Optimisations BigQuery:

```sql
-- Créer une table partitionnée et clusterée
CREATE TABLE `projet.dataset.transactions`
PARTITION BY DATE(date_transaction)
CLUSTER BY agence_id, client_id
AS SELECT * FROM `projet.dataset.transactions_raw`;

-- Requête optimisée
SELECT client_id, SUM(montant)
FROM `projet.dataset.transactions`
WHERE date_transaction BETWEEN '2024-01-01' AND '2024-01-31'
  AND agence_id = 'AG001'
GROUP BY client_id;
-- BigQuery ne scanne que la partition de janvier et le cluster AG001
```

Autres optimisations: - SELECT uniquement les colonnes nécessaires - Éviter SELECT * - Utiliser les vues matérialisées

---

Section 4: Orchestration et Monitoring (5 questions)

Question 16

Quel est le rôle d'Apache Airflow dans un pipeline de données?

A) Stockage des données
B) Orchestration et scheduling des tâches
C) Visualisation des données
D) Machine Learning

Réponse

B) Orchestration et scheduling des tâches

Apache Airflow: - Définit des DAGs (Directed Acyclic Graphs) - Planifie l'exécution (cron-like) - Gère les dépendances entre tâches - Monitore les exécutions - Gère les reprises après échec

```python
from airflow import DAG
from datetime import datetime

with DAG(
    'etl_bancaire_quotidien',
    schedule_interval='0 6 * * *',  # Tous les jours à 6h
    start_date=datetime(2024, 1, 1),
    catchup=False
) as dag:

    t1 = extract_task()
    t2 = transform_task()
    t3 = load_task()
    t4 = notify_task()

    t1 >> t2 >> t3 >> t4
```

Question 17

Qu'est-ce qu'un "backfill" dans le contexte des pipelines de données?

A) Remplir des données manquantes dans le passé
B) Exécuter un pipeline pour des dates historiques
C) Sauvegarder les données
D) Supprimer les anciennes données

Réponse

B) Exécuter un pipeline pour des dates historiques

Backfill = Rejouer un pipeline pour des périodes passées

Scénarios: - Nouveau pipeline à appliquer sur l'historique - Correction d'une erreur passée - Ajout d'une nouvelle métrique à calculer rétroactivement

```bash
# Airflow backfill
airflow dags backfill etl_bancaire \
    --start-date 2024-01-01 \
    --end-date 2024-06-30
```

⚠ Attention: S'assurer que le backfill n'écrase pas des données valides

---

Question 18

Comment alerter l'équipe si un pipeline échoue?

A) Vérifier manuellement chaque matin
B) Configurer des callbacks d'échec avec notifications (email, Slack)
C) Ignorer les échecs
D) Redémarrer automatiquement sans notification

Réponse

B) Configurer des callbacks d'échec avec notifications (email, Slack)

```python
from airflow import DAG
from airflow.operators.python import PythonOperator
from airflow.utils.email import send_email

def failure_callback(context):
    """Appelé quand une tâche échoue"""
    task_instance = context['task_instance']

    # Envoyer email
    send_email(
        to=['data-team@unibank.ht'],
        subject=f"Pipeline {task_instance.dag_id} FAILED",
        html_content=f"Tâche {task_instance.task_id} a échoué"
    )

    # Ou notification Slack
    slack_notification(f"🚨 Pipeline échoué: {task_instance.dag_id}")

with DAG('etl_bancaire',
         default_args={'on_failure_callback': failure_callback}
        ) as dag:
    # ...
```

Question 19

Qu'est-ce que l'idempotence dans un pipeline de données?

A) Un pipeline qui s'exécute une seule fois
B) Un pipeline qui produit le même résultat peu importe le nombre d'exécutions
C) Un pipeline très rapide
D) Un pipeline sans erreurs

Réponse

B) Un pipeline qui produit le même résultat peu importe le nombre d'exécutions

Pipeline idempotent: - Exécuté 1 fois → Résultat A - Exécuté 2 fois → Résultat A (identique) - Exécuté N fois → Résultat A (toujours identique)

Comment garantir l'idempotence:

```
# Mauvais (non idempotent): INSERT à chaque exécution
INSERT INTO table VALUES (...)  # Crée des doublons!

# Bon (idempotent): DELETE + INSERT ou MERGE
DELETE FROM table WHERE date = '2024-01-15';
INSERT INTO table SELECT * FROM staging WHERE date = '2024-01-15';

# Ou avec MERGE (upsert)
MERGE INTO target USING source
ON target.id = source.id
WHEN MATCHED THEN UPDATE ...
WHEN NOT MATCHED THEN INSERT ...
```

Question 20

Quelle métrique surveiller pour détecter une dégradation du pipeline?

A) La couleur des logs
B) Le temps d'exécution, le volume traité, le taux d'erreur
C) Le nom des tâches
D) La version du logiciel

Réponse

B) Le temps d'exécution, le volume traité, le taux d'erreur

Métriques de monitoring:

Métrique	Alerte si	Action
Temps d'exécution	+50% vs moyenne	Investiguer lenteur
Volume traité	-20% vs attendu	Vérifier source
Taux d'erreur	> 1%	Vérifier qualité
Latence des données	> SLA	Optimiser pipeline
Utilisation ressources	> 80%	Scaler infrastructure

```
# Exemple de validation de volume
rows_processed = len(df)
expected_rows = get_expected_rows(date)

if rows_processed < expected_rows * 0.8:
    raise ValueError(f"Volume anormal: {rows_processed} vs {expected_rows} attendus")
```

Section 5: Sécurité et Gouvernance (5 questions)

Question 21

Comment protéger les données sensibles (NIF, numéro de compte) dans un pipeline?

A) Les ignorer
B) Appliquer le masquage ou la tokenisation
C) Les stocker en texte clair
D) Les supprimer complètement

Réponse

B) Appliquer le masquage ou la tokenisation

Techniques de protection:

Technique	Description	Exemple
Masquage	Cacher partiellement	1234-****-****-5678
Tokenisation	Remplacer par token	ACC_TOKEN_X7Y9
Hachage	Transformation irréversible	SHA256(NIF)
Chiffrement	Réversible avec clé	AES(données, clé)

```python
import hashlib

def masquer_compte(numero):
    """Masque le numéro de compte"""
    return numero[:4] + '****' + numero[-4:]

def tokeniser(valeur_sensible, secret):
    """Crée un token irréversible"""
    return hashlib.sha256((valeur_sensible + secret).encode()).hexdigest()[:16]

# Application
df['compte_masque'] = df['numero_compte'].apply(masquer_compte)
```

Question 22

Qu'est-ce que le data lineage?

A) La généalogie des employés
B) Le traçage de l'origine et des transformations des données
C) Le tri des données
D) La suppression des données anciennes

Réponse

B) Le traçage de l'origine et des transformations des données

Data Lineage répond à: - D'où viennent ces données? (origine) - Quelles transformations ont été appliquées? - Qui a accédé aux données? - Quels rapports utilisent ces données?

```
[Source: Core Banking] → [Staging] → [Transform] → [Data Mart] → [Rapport]
        ↓                    ↓            ↓            ↓
     Raw tables          Nettoyage    Agrégation    KPIs
```

Outils: Apache Atlas, DataHub, dbt (lineage automatique)

Importance bancaire: Conformité réglementaire, audit, impact analysis

---

Question 23

Comment gérer les droits d'accès aux données dans un data warehouse?

A) Donner accès à tout le monde

B) Implémenter le RBAC (Role-Based Access Control)
C) Chiffrer tout sans donner accès
D) Utiliser un seul compte partagé

Réponse

B) Implémenter le RBAC (Role-Based Access Control)

```sql
-- BigQuery exemple
-- Créer des rôles
CREATE ROLE data_analyst;
CREATE ROLE data_scientist;
CREATE ROLE data_admin;

-- Attribuer des permissions
GRANT SELECT ON DATASET reporting TO ROLE data_analyst;
GRANT SELECT, INSERT ON ALL TABLES TO ROLE data_scientist;
GRANT ALL PRIVILEGES TO ROLE data_admin;

-- Masquer des colonnes sensibles
CREATE VIEW clients_safe AS
SELECT
    client_id,
    segment,
    ville,
    '***' AS nif_masque  -- NIF masqué pour analysts
FROM clients;

GRANT SELECT ON clients_safe TO ROLE data_analyst;
```

Principe du moindre privilège: Donner uniquement les accès nécessaires.

Question 24

Qu'est-ce que la rétention des données et pourquoi est-ce important?

A) La vitesse de traitement
B) La durée de conservation des données avant archivage/suppression
C) La taille des fichiers
D) Le format de stockage

Réponse

B) La durée de conservation des données avant archivage/suppression

Politique de rétention bancaire (exemple):

Type de données	Rétention	Raison
Transactions	10 ans	Réglementation BRH
Logs d'accès	5 ans	Audit sécurité
Données marketing	3 ans	RGPD / Consentement
Données temporaires	30 jours	Nettoyage

```python
# Implémentation automatique
from datetime import datetime, timedelta
```

```python
def appliquer_retention(df, colonne_date, jours_retention):
    """Supprime les données au-delà de la rétention"""
    date_limite = datetime.now() - timedelta(days=jours_retention)
    return df[df[colonne_date] >= date_limite]

# Archiver avant suppression
df_archive = df[df['date'] < date_limite]
df_archive.to_parquet(f'archive/{year}/data.parquet')
```

Question 25

Comment documenter un pipeline de données pour faciliter la maintenance?

A) Ne pas documenter, le code est suffisant
B) Créer un README, des commentaires, un catalogue de données
C) Documenter uniquement les erreurs
D) Utiliser des noms de variables aléatoires

Réponse

B) Créer un README, des commentaires, un catalogue de données

Éléments de documentation:

1. **README du pipeline:**

```markdown
# Pipeline ETL Transactions
## Description
Charge les transactions quotidiennes vers le data warehouse.

## Schedule
Quotidien à 6h00

## Dépendances
- Source: Core Banking (Oracle)
- Destination: BigQuery

## Contact
data-team@unibank.ht
```

2. **Catalogue de données:**

```yaml
# schema.yml (dbt)
models:
  - name: transactions
    description: "Transactions clients nettoyées et enrichies"
    columns:
      - name: transaction_id
        description: "Identifiant unique de la transaction"
        tests: [unique, not_null]
      - name: montant
        description: "Montant en HTG"
```

3. **Commentaires dans le code:**

```python
def transform_transactions(df):
    """
    Transforme les transactions brutes.

    Étapes:
    1. Suppression des doublons
    2. Conversion des montants
    3. Enrichissement avec référentiels

    Args:
        df: DataFrame des transactions brutes
    Returns:
        DataFrame transformé
    """
```

Résumé et Mnémotechniques

"ACCA" - Qualité des Données

```
A - Accuracy (Exactitude): Données correctes?
C - Completeness (Complétude): Données présentes?
C - Consistency (Cohérence): Données logiques?
A - Actuality (Actualité): Données récentes?
```

Pipeline Pattern

```
Source → Extract → [Staging] → Transform → [Quality Check] → Load → Monitor
           ↓                       ↓              ↓             ↓
       Logs CDC              Validation     Great Expect.   Alertes
```

Formats de Données

```
CSV  → Échange simple, petit volume
JSON → APIs, semi-structuré
Parquet → Analytics, colonnes, compression
Avro → Streaming, schéma évolutif
```

Score et Auto-évaluation

Score	Niveau	Recommandation
23-25	Expert	Prêt pour l'examen
18-22	Avancé	Réviser les points faibles
13-17	Intermédiaire	Revoir le document complet
< 13	Débutant	Étude approfondie nécessaire

Test préparé pour l'examen Data Analyst - UniBank Haiti Thème: Data Engineering - Les fondations de l'analyse