Data Engineer Interview Test Tasks

Загальний контекст завдань

Ви працюєте інженером даних в інтернет-книгарні. Компанія потребує створити аналітичну систему для відстеження продажів книг.

Завдання пов'язані між собою: спочатку ви створите схему бази даних (Завдання 1), а потім напишете Python скрипт для обробки даних з цієї бази (Завдання 2).

Чудово, якщо вийде протестувати на реальній базі PostgreSQL (наприклад, https://neon.tech), якщо ні, можна просто написати код та додати посилання на репозиторій чи надіслати файли архівом.

Завдання 1: PostgreSQL Schema Design

Контекст

Створіть базову схему бази даних для інтернет-книгарні. Ця схема буде використовуватися для зберігання інформації про книги та їх подальшої обробки ETL процесами.

Детальні вимоги:

TODO 1: Створіть основну таблицю `books`

Таблиця повинна містити наступні поля:

```
CREATE TABLE books (
    -- book_id: автоінкремент (SERIAL), первинний ключ
    -- title: текстове поле до 500 символів, НЕ може бути NULL
    -- price: десяткове число (10 цифр загалом, 2 після коми),
    -- genre: текстове поле до 100 символів, може бути NULL
```

```
    -- stock_quantity: ціле число, за замовчуванням 0,
    -- last_updated: часова мітка, за замовчуванням поточний час
);
```

TODO 2: Створіть таблицю для обробленх даних `books_processed`

Ця таблиця буде використовуватися Python скриптом для збереження ETL результатів:

```
CREATE TABLE books_processed (
    -- processed_id: автоінкремент, первинний ключ
    -- book_id: ціле число (копія з основної таблиці)
    -- title: текстове поле до 500 символів (копія з основної таблиці)
    -- original_price: оригінальна ціна з основної таблиці
    -- rounded_price: ціна округлена до 1 знака після коми
    -- genre: жанр (копія з основної таблиці)
    -- price_category: текстове поле ('budget' або 'premium')
    -- processed_at: часова мітка обробotки, за замовчуванням поточний час
);
```

TODO 3: Створіть індекси для оптимізації

Створіть індекси з точними назвами:

```
-- 1. idx_books_genre - для швидкого пошуку книг за жанром
-- 2. idx_books_last_updated - для пошуку книг за датою останнього оновлення
-- 3. idx_books_price_range - для пошуку книг у певному ціновому діапазоні
```

TODO 4: Додайте тестові дані

Вставте рівно 6 записів книг з наступними вимогами:

- Мінімум 3 різних жанри
- Ціни від 200 до 800 грн (щоб протестувати категорії 'budget'/'premium')
- Різні дати last_updated
- Різну кількість stock_quantity

Приклад формату:

```
INSERT INTO books (title, price, genre, stock_quantity, last_updated) VALUES
('Назва книги 1', 299.99, 'фантастика', 15, '2025-01-15 10:30:00'),
-- додайте ще 5 записів
```

Завдання 2: Python ETL Script

Контекст

Тепер, коли схема бази даних створена, напишіть ETL скрипт для щоденної обробки даних книг. Аналітична команда потребує агрегованих даних для дашбордів.

Детальні технічні вимоги:

Файл: `books_etl.py`

Залежності для встановлення:

```
pip install pandas sqlalchemy psycopg2-binary
```

Параметри підключення до БД:

Скрипт має читати параметри з environment variables:

- `DB_HOST` - хост бази даних
- `DB_PORT` - порт (за замовчуванням 5432)
- `DB_NAME` - назва бази даних
- `DB_USER` - користувач
- `DB_PASSWORD` - пароль

Детальна реалізація функцій:

1. Функція `connect_to_db()`:

```
def connect_to_db():
    """
    Створити SQLAlchemy engine для підключення до PostgreSQL
    Використовуйте environment variables для параметрів підключення
    Поверніть engine об'єкт
```

```
    Обробіть помилки підключення
    """
    # TODO: ваша реалізація
```

2. Функція `extract_books(engine, cutoff_date)`:

```
def extract_books(engine, cutoff_date):
    """
    Витягнути книги з таблиці books де last_updated >= cutoff_date

    Параметри:
    - engine: SQLAlchemy engine
    - cutoff_date: рядок в форматі 'YYYY-MM-DD'

    Поверніть: pandas DataFrame з колонками:
    book_id, title, price, genre, stock_quantity, last_updated

    Виведіть кількість знайдених записів
    """
    # TODO: ваша реалізація
```

3. Функція `transform_data(df)`:

```
def transform_data(df):
    """
    Трансформувати дані згідно бізнес-правил:

    1. Створити колонку 'original_price' (копія 'price')
    2. Округлити 'price' до 1 знака після коми та зберегти як нову колонку з
назвою 'rounded_price'
    3. Створити 'price_category':
       - 'budget' якщо rounded_price < 500
       - 'premium' якщо rounded_price >= 500
    4. Видалити оригінальну колонку 'price'

    Поверніть: трансформований DataFrame
    Виведіть кількість оброблених записів
    """
    # TODO: ваша реалізація
```

4. Функція `load_data(df, engine)`:

```
def load_data(df, engine):
    """
    Зберегти оброблені дані в таблицю books_processed

    Використовуйте df.to_sql() з параметрами:
    - if_exists='append' (додавати до існуючих даних)
    - index=False (не зберігати індекс DataFrame)
    - chunksize=1000 (пакетна обробка)

    Виведіть кількість збережених записів
    Обробіть помилки збереження
    """
    # TODO: ваша реалізація
```

5. Функція `main()`:

```
def main():
    """
    Головна функція:
    1. Перевірити аргументи командного рядка (має бути рівно 1 - дата)
    2. Валідувати формат дати (YYYY-MM-DD)
    3. Викликати всі ETL функції в правильному порядку
    4. Обробити помилки та вивести підсумкову статистику
    """
    if len(sys.argv) != 2:
        print("Використання: python books_etl.py YYYY-MM-DD")
        print("Приклад: python books_etl.py 2025-01-01")
        sys.exit(1)

    # TODO: ваша реалізація
```

Приклад використання:

```
python books_etl.py 2025-01-01
```

Очікуваний вивід:

```
Підключено до бази даних успішно
Витягнуто 4 записів з таблиці books
Трансформовано 4 записів
Збережено 4 записів в books_processed
ETL процес завершено успішно
```

Обов'язкова обробка помилок:

- Помилки підключення до БД
- Помилки SQL запитів
- Невірний формат дати
- Порожній результат запиту - скрипт має завершити роботу з інформаційним повідомленням, наприклад: "Нових книг для обробки за вказану дату не знайдено. Роботу завершено."

Інструкції

Порядок виконання:

1. ✅ **Спочатку виконайте Завдання 1** (створіть схему БД та тестові дані)
2. ✅ **Потім виконайте Завдання 2** (напишіть та протестуйте Python скрипт)
3. ✅ **Перевірте зв'язок** між завданнями (скрипт має працювати з вашою схемою)

Що здавати (може бути посилання на репозиторій чи архів):

- `books_schema.sql` - повне рішення Завдання 1
  `books_etl.py` - повний робочий Python скрипт
- `requirements.txt` - список Python залежностей
- `README.md` - інструкції з запуску (які environment variables потрібні, як запустити скрипт)

Критерії оцінювання:

- **Функціональність:** код працює без помилок
- **Повнота:** виконані всі TODO пункти
- **Якість коду:** правильна обробка помилок, читабельність
- **Розуміння:** правильні відповіді на теоретичні питання

Удачі! 🚀

Примітка: Завдання імітують реальний робочий процес - спочатку проектування схеми БД, потім створення ETL пайплайну для роботи з нею.