## 🔶 Python Task: Data Pipeline Development

**Title**: Build a Mini Data Ingestion and Processing Pipeline

**Objective**:
Create a Python module that:

- Ingests data from a public API (e.g., OpenWeather, NewsAPI)
- Processes the data (cleans, normalizes, and filters it)
- Saves results to a CSV/JSON file and inserts them into a local SQLite/PostgreSQL database

**Requirements**:

- Use `requests` or `httpx` for API integration
- Use `pydantic` or `marshmallow` for data validation
- Implement proper logging (e.g., using Python `logging` module)
- Implement error handling with retry mechanisms
- Structure the code modularly using functions and/or OOP principles

**Deliverables**:

- Source code in a GitHub repo or as a ZIP archive
- README with setup instructions and how to run the script
- Sample output files
- SQL schema or database dump file

**Bonus (Optional)**:

- Add scheduling (e.g., using `schedule` or `APScheduler`)

- Add CLI interface using `argparse` or `click`
- Containerize with Docker

---

## 🔷 SQL Task: E-Commerce Reporting Dashboard

**Title**: Write SQL Queries for Business Intelligence

**Objective**:
Use an e-commerce database schema to generate business insights with SQL queries.

**Dataset Schema**:

- `customers (customer_id, name, email, created_at)`
- `orders (order_id, customer_id, total_amount, order_date)`
- `order_items (order_item_id, order_id, product_id, quantity, price)`
- `products (product_id, name, category, price)`

**Tasks**:

1. List the top 10 customers by total spend.
2. Generate a report of daily revenue and order count for the last 30 days.
3. Identify most sold products in the last 3 months.
4. Calculate the conversion rate (orders/site visits) if given a `site_visits` table.
5. Show total revenue broken down by product category.

**Requirements**:

- Use joins, CTEs, and window functions where appropriate
- Avoid suboptimal query patterns

- Create reusable views or materialized views if possible

**Deliverables**:

- Sample dataset or source location of dataset used

**Bonus (Optional)**:

- Integrate queries into a Python reporting script
- Use PostgreSQL advanced features (e.g., JSON fields, triggers, indexes)
- Create a stored procedure for monthly reporting

---

**Instructions**:

- Start with one task at a time.
- Push all work to a version-controlled GitHub repository.
- Keep code clean, documented, and well-structured.
- Reach out if clarification is needed on any task.