Final Project

Topic-1 – Retail Sales Data

Working with **Retail Sales Data** stored in csv format.

Project Instructions

1. **Database Design**
   - Create a schema with the following tables: Customers, Products, and Orders.
   - Example columns:
     - Customers(id, name, location)
     - Products(id, name, price)
     - Orders(order_id, customer_id, product_id, quantity, order_date)
2. **ETL Pipeline**
   - Extract data from csv.
   - Clean and transform the data:
     - Handle missing customer names or product prices.
     - Calculate revenue = quantity × price.
     - Apply discount for some products.
   - Load the transformed data into your SQL tables.
   - Document a simple Airflow DAG diagram that would automate this pipeline (Optional).
3. **Data Warehouse**
   - Design a **star schema**:
     - Fact: SalesFact(order_id, product_id, customer_id, quantity, revenue, date_id)
     - Dimensions: ProductDim, CustomerDim, DateDim
   - Write SQL queries:
     - Top 5 best-selling products.
     - Monthly revenue trends.
     - Customer with the highest total purchase value.
4. **Streaming Simulation**
   - Simulate real-time orders using Python (randomly generate new orders every few seconds).
   - Process each incoming order:
     - Update running total revenue.
     - Flag orders with revenue > $1000.
   - Print or insert the results into the database.

Topic-2 – HR Employee Data

Working with **Employee Data** stored in csv format.

Project Instructions

1. **Database Design**
   - Create a schema with the following tables: Employees, Departments, Attendance.
   - Example columns:

        - Employees(emp_id, name, department_id, salary, join_date)
        - Departments(dept_id, dept_name)
        - Attendance(att_id, emp_id, checkin_date, hours_worked)

2. **ETL Pipeline**
    - Extract data from employees.csv.
    - Clean and transform the data:
        - Fill missing salaries with department average.
        - Standardize department names (e.g., “HR” vs “Human Resources”).
        - Add a new column: bonus = 5% of salary.
    - Load into the database tables.
    - Document a simple Airflow DAG diagram for this ETL process (Optional).
3. **Data Warehouse**
    - Design a **star schema**:
        - Fact: HRFact(emp_id, dept_id, salary, bonus, attendance, date_id)
        - Dimensions: EmployeeDim, DeptDim, DateDim
    - Write SQL queries:
        - Average salary by department.
        - Employees with the lowest attendance.
        - Department with the most employees.
4. **Streaming Simulation**
    - Simulate real-time check-in logs (e.g., every second generate an employee login).
    - Process events:
        - Calculate number of employees present.
        - Flag late check-ins after 9:00 AM.
    - Display or insert results into the database.

Topic-3 – Finance Transactions Data

Working with **Bank Transactions Data** stored in csv format.

Project Instructions

1. **Database Design**
    - Create schema with the following tables: Customers, Accounts, Transactions.
    - Example columns:
        - Customers(cust_id, name, address)
        - Accounts(acc_id, cust_id, balance)
        - Transactions(trans_id, acc_id, amount, trans_type, trans_date)
2. **ETL Pipeline**
    - Extract data from transactions.csv.
    - Clean and transform the data:
        - Remove duplicates.
        - Fix missing account IDs.
        - Calculate running account balances.
        - Flag suspicious transactions (amount > 10,000).
    - Load the cleaned data into the database.
    - Document a simple Airflow DAG diagram for this ETL (Optional).

3. **Data Warehouse**
    - Design a **star schema**:
        - Fact: TransactionFact(trans_id, acc_id, amount, trans_type, date_id)
        - Dimensions: CustomerDim, AccountDim, DateDim
    - Write SQL queries:
        - Top 5 customers with the highest spending.
        - Average debit/credit per month.
        - List of flagged suspicious transactions.
4. **Streaming Simulation**
    - Simulate real-time debit/credit transactions (generate random transactions).
    - Process events:
        - Update account balances.
        - Flag overdrafts (balance < 0).
    - Display or insert results into the database.