Accident Analysis Data Warehouse Assignment Report

Introduction

This report summarizes my work on the Accident Analysis Data Warehouse assignment. The goal was to design and implement an ETL pipeline to build a data warehouse for analyzing traffic accidents, using a star schema in a SQLite database (accident_data_warehouse.db) populated from an Excel file (Traffic.xlsx). Below, I outline the key steps I took, the issues I encountered in the ERD and sample data, the assumptions I made, and the final outcomes.

Steps Taken

1. **Selecting the Business Process**:
    - I analyzed the Traffic Flow ERD, which included entities like Accidents, Vehicles, Road Conditions, Locations, and Dates. I chose the business process of **accident analysis**, focusing on analyzing traffic accidents by location, date, vehicle type, road conditions, and severity. This process helps traffic authorities identify high-risk areas, understand contributing factors, and improve road safety.
    - I selected this process because the Accidents entity was central, linking to other entities, and the data in Traffic.xlsx (with sheets for Vehicles, Accidents, and Condition) supported this analysis.
2. **Filling the Design Template**:
    - I designed a star schema to support accident analysis and filled the design template as follows:
        - **Fact Table**: Fact_Accidents (columns: AccidentID, DateID, LocationID, VehicleID, RoadConditionID, VehiclesInvolved, SeverityScore).
        - **Dimension Tables**:
            - Dim_Location (LocationID, LocationName): Stores unique locations.
            - Dim_Vehicle (VehicleID, VehicleType): Stores vehicle types.
            - Dim_RoadCondition (RoadConditionID, Surface, Visibility): Stores road condition data.
            - Dim_Date (DateID, Date, Month, Year, Time): Stores date and time details.
    - This schema enables efficient querying, such as counting accidents by location or analyzing severity by road condition.
3. **Writing the Pseudocode**:
    - I wrote pseudocode to plan the ETL pipeline:
        - **Extract**: Load data from traffic_data.xlsx (Vehicles, Accidents, Condition sheets) and convert timestamps (ReportedAt, RecordedAt) to datetime format.
        - **Transform**: Create dimension tables (Dim_Location, Dim_Vehicle, Dim_RoadCondition, Dim_Date) from the data, then build Fact_Accidents by mapping IDs and converting Severity to SeverityScore (Minor: 1, Moderate: 2, Severe: 3).

        - **Load**: Populate the SQLite database with the star schema tables, starting with dimension tables, then the fact table.
4. **Implementing the ETL Pipeline**:
    - I implemented the ETL pipeline in etl_pipeline.py using Python (pandas, sqlalchemy, logging):
        - Extracted data from the Excel file and converted timestamps.
        - Transformed the data into the star schema, mapping DateID, LocationID, VehicleID, and RoadConditionID for Fact_Accidents.
        - Loaded the data into accident_data_warehouse.db, ensuring proper table creation and population.
    - I also created:
        - generate_snapshots.py: Displays the first 10 rows of each table.
        - queries.sql: Runs SQL queries for analysis (e.g., top accident locations, severity by road condition, accidents by month).

Issues Encountered

1. **ERD Issues**:
    - Unclear how to match Accidents to Condition (no direct link).
    - Ambiguity in VehicleID (primary vehicle unclear for multi-vehicle accidents).
2. **Sample Data Issues**:
    - **Missing Attributes in Accidents**: The Accidents entity in the ERD likely had more attributes (e.g., vehicle id, driver id, roadc condition id), but the Accidents sheet only had AccidentID, ReportedAt, Location, VehiclesInvolved, and Severity, limiting analysis.
    - **Location Mismatch**: Location values in Accidents (e.g., “Morris Lane”) didn’t match Condition (e.g., “Downtown”), leading to random RoadConditionID assignments (impacting road condition queries).

Assumptions Made

1. **Location Mismatch**:
    - Assumed the mismatch was a data error and proceeded with random RoadConditionID assignments, noting it as a known issue.
2. **Single VehicleID**:
    - Assumed the VehicleID in Accidents was the primary vehicle, despite VehiclesInvolved indicating multiple vehicles.
3. **Severity Mapping**:
    - Assumed Severity values (“Minor”, “Moderate”, “Severe”) should map to scores (1, 2, 3) for analysis.

Final Outcomes

- **ETL Pipeline**: etl_pipeline.py successfully loads data into the database with the star schema.
- **Table Snapshots**: generate_snapshots.py shows the first 10 rows of each table.
- **SQL Queries**: queries.sql provides queries for analysis (e.g., top accident locations, severity by road condition).
- **Row Counts**:
    - Dim_Location: 100 rows
    - Dim_Vehicle: 200 rows
    - Dim_RoadCondition: 100 rows
    - Dim_Date: 50 rows
    - Fact_Accidents: 100 rows
- **Limitation**: The location mismatch and missing Accidents attributes limit the analysis scope, but these are data issues, not code issues.

Conclusion

This project taught me how to design and implement a data warehouse using an ETL pipeline. I successfully built a star schema for accident analysis, addressed data issues, and provided SQL queries for insights.