COMP63301-Data Engineering Concepts

Workshop 2 Description for Students

1 Introduction

During this workshop, you will explore the same real-world data engineering scenario from Week 1, which is centred around scooter rental activity in Albuquerque, California — a popular destination for summer holidays[1]. With its scenic routes and vibrant tourist culture, Albuquerque attracts both locals and visitors who frequently rent electric scooters to explore the area. These rentals are facilitated through an online application, and each scooter is tracked via GPS, generating rich datasets that include trip start and end times, locations, and vehicle identifiers.

Despite the technological sophistication of GPS tracking, historical data reveals notable issues with data completeness and reliability. You will delve into these imperfections, learning how to identify some of the most common data quality challenges in large-scale, timeseries datasets. This use case not only offers technical depth but also connects with the seasonal habits of young people and tourists — making it both relevant and engaging.

The principal aim of this workshop is to provide you with hands-on experience in modelling and storing time-series historical data within a Relational Database Management System (RDBMS). Building upon the foundational skills you developed in Week_1—particularly through your engagement with Workshop_1 and Coursework_1—you will transition from working with raw, unrefined datasets in CSV format to designing and implementing a structured relational database using **SQLite**.

The dataset in question pertains to the operational history of a Scooter Rental Company based in Albuquerque, California. While CSV files offer a simple and accessible format for initial data exploration and profiling, they are inherently limited when it comes to scalability, data integrity, and efficient querying. By migrating this data into a relational schema, you will learn to identify and define meaningful relationships between attributes, normalise the data to reduce redundancy, and design a schema that mitigates common anomalies such as deletion and update inconsistencies.

This workshop will not only deepen your understanding of relational modelling principles but also demonstrate the tangible benefits of structured data storage—such as improved query performance, enhanced data consistency, and better support for analytical tasks. These are essential skills for any data engineer working with historical datasets in real-world scenarios, particularly in domains where time-series data plays a critical role in business intelligence and operational decision-making.

2 Before Starting Workshop Part 1

Prior to beginning the first part of this workshop, you must download the two CSV files provided via the course's Canvas space. These datasets contain raw GPS-tracked scooter rental records and geo-coded street location data. They will form the basis of the database design activity to be undertaken during the session.

Ensure that the files are saved locally and are accessible from your Python environment. Familiarity with the structure and content of these files will be beneficial as you begin to model the data into a relational format.

[1] This use case data and some of the exploration can be found in book "Data Engineering with Python" by Paul Crickard, 2020, Packt Publishing, and on GitHub (https://github.com/PaulCrickard/ escooter/blob/master/scooter.csv)

3 Workshop Part 1

In this first part of the workshop, you will apply your understanding of relational modelling to transform the raw data into a structured database schema. This exercise will help you appreciate the advantages of relational databases over flat file formats, particularly in terms of data integrity, query efficiency, and scalability.

Student Task

Using the two CSV files you downloaded from the course's Canvas space, design a **Relational Database** to represent the data contained within them. Your design should demonstrate thoughtful consideration of:

- **Normalisation** – ensuring that data is structured efficiently to reduce redundancy.
- **Performance** – enabling effective querying and scalability.
- **Anomaly Prevention** – avoiding update and deletion anomalies through sound schema design.

Once you have completed your design, discuss it with the peer seated next to you. Compare your approaches and reflect on the following questions:

1. How well is my design normalised?
2. How effectively does my design minimise redundancies?
3. How likely is my relational database to perform efficiently under typical query loads?

4 Before Starting Workshop Part 2

It is imperative that you have access to an installation of **SQLite** (`sqlite3`, version 3.37.2 is recommended for consistency across the cohort, although later versions may also be compatible). Prior to the workshop, you should also download the SQLite database file associated with this workshop, which is likewise available on Canvas.

In the following, we provide a short tutorial on how to work with SQLlite from a Python program.

4.1 Working with SQLite from Python

This tutorial provides a brief introduction to working with **SQLite** from a Python script, using embedded SQL for data manipulation. It assumes that you have already installed Python and SQLite on your personal laptop, and that you have downloaded the SQLite database file provided on the course's Canvas space.

4.1.1 Starting SQLite from Python

SQLite is a serverless database engine, which means it does not require a separate server process to be started. Instead, it is embedded directly within Python via the `sqlite3` module. To begin, you must import this module in your Python script:

```
import sqlite3
```

4.1.2 Loading the SQLite Database File

To load the database file, use the `connect()` method. This establishes a connection between your Python script and the SQLite database file:

```
conn = sqlite3.connect("workshop2.db")
```

Ensure the file is located in the same directory as your Python script.

4.1.3 Executing SQL Queries from Python

Once connected, you can create a cursor object to execute SQL statements:

```
cursor = conn.cursor()
```

You may now embed SQL directly within your Python code. For example, to retrieve all records from the `trip` table:

```
cursor.execute("SELECT * FROM trip")
result = cursor.fetchall()
print(result)
conn.close()
return result[:5]
```

4.1.4 Closing the Connection

After completing your queries, it is good practice to close the connection to the database:

```
connection.close()
```

4.1.5 Notes on Data Manipulation

Although Python is used to interface with the database, most data manipulation will be performed using SQL statements embedded within the Python code. This approach allows for efficient querying, filtering, and aggregation of historical data, which is particularly useful when working with time-series datasets such as those from the Scooter Rental Company.

4.1.6 Best Practices

- Always close your database connection to avoid resource leaks.
- Use parameterised queries when working with user input to prevent SQL injection.
- Keep your SQL statements readable and well-structured for maintainability.

This setup provides a robust and scalable way to interact with structured historical data, offering significant advantages over working with raw CSV files alone.

5 Workshop Part 2

In this part of the workshop, you will begin to interact with the SQLite database using SQL queries embedded within Python. The aim is to develop fluency in querying structured data and uncovering meaningful patterns from historical records.

5.1 Exploring the Database

You are encouraged to apply the exploratory mindset developed in earlier sessions, but now through the lens of relational technology. This exercise will help reinforce the advantages of structured querying over manual inspection of raw CSV files.

Student Task

Explore the SQLite database using basic exploratory techniques, similar to those applied in Workshop 1, but this time using SQL embedded within Python. To keep query outputs manageable, apply `LIMIT 5` to each `SELECT` statement.

1. Inspect the column names of the tables loaded into your Python environment.

   Reflective Question

   How does the schema of this database compare with the schema you designed in Part 1 of this workshop?

2.
3. Write a query to select all scooter trips that occurred during the month of May.
4. Write a query to retrieve all trips performed by user `8417864`.
5. Feel free to explore additional patterns or queries that you find interesting or insightful.

   Reflective Question

   How does SQL-based data exploration compare with Python-based exploration over CSV files? What do you perceive as the main advantages and disadvantages of each approach?

6.

5.2 Cleaning the Data

In this section of the workshop, you will practice data cleaning techniques using SQL embedded within Python. The dataset contains missing values that must be addressed to ensure consistency and integrity within the relational database.

You will focus on two key attributes: `DURATION`, which records the length of each scooter trip, and `end_location_name`, which identifies the final location of each journey. By applying structured queries and respecting relational constraints, you will enhance the quality of the dataset and prepare it for further analysis.

Student Task

Use **SQLite** and **Python** to clean the dataset, which contains missing values in the `DURATION` and `end_location_name` attributes. Complete the following tasks:

1. Replace all missing or invalid entries in the `DURATION` column with values formatted as `HH:MM:SS`. Any missing value should be set to `00:00:00`.
2. Replace missing values in the `end_location_name` column with appropriate entries. **Hint:** Consult the `geocode` table and consider using `"Nothing Street"` as a placeholder.
3. Insert the entry for `"Nothing Street"` from the `geocode` table into the `location` table. Ensure that you respect the foreign key constraint on the `geocode_id`. **Important:** Do not hardcode the `geocode_id`; retrieve it dynamically using a query.

   Reflective Question

   How does SQL-based data cleaning compare with Python-based cleaning over CSV files? What do you perceive as the main advantages and disadvantages of each approach?

4.

5.3 Data Analysis

This section of the workshop focuses on applying advanced analytical techniques to the structured dataset using SQL queries embedded within Python. You will explore patterns in trip durations, identify anomalies, and assess vehicle performance across different journey routes.

These tasks are designed to deepen your understanding of relational querying, statistical reasoning, and the integration of SQL with Python for scalable data analysis.

Student Task

Using a combination of **SQL** and **Python**, carry out the following advanced analytical tasks on the scooter rental dataset:

1. For each distinct pair of `(start_location_name, end_location_name)`, identify the user `(user_id)` and trip `(trip_id)` associated with the shortest recorded duration. Report the duration and determine the time of year in which each of these trips occurred.
2. For each distinct pair of `(start_location_name, end_location_name)`, calculate the **average** trip duration.
3. For each distinct pair of `(start_location_name, end_location_name)`, calculate the **median** trip duration.
4. For each distinct pair of `(start_location_name, end_location_name)`, identify any trip durations that may be considered statistical outliers. If outliers are found, report the associated `user_id(s)` and `trip_id(s)`.
5. Determine whether any specific vehicle `(vehicle_id)` is consistently associated with the longest duration trips for each distinct pair of `(start_location_name, end_location_name)`. If so, identify these vehicles. Note that this analysis may include trips previously flagged as outliers.
6. **Reflective Question**

   Based on the results you have obtained, identify and reflect on any observable patterns. Discuss your observations with the peer seated next to you before sharing your insights with the class.
7. **Reflective Question**

   How much of the above queries were you able to write using only SQL?