

SQL RELATIONAL DATABASE FOR ACCIDENT DATA MANAGEMENT



STUDENT NAME: BOBOYE ADEJUWON
STUDENT ID: 202346817



INTRODUCTION

A relational database enables effective data storage, management and retrieval in a structured format i.e. rows and columns. To improve public safety, it is imperative to effectively analyse road traffic accident data. Governments utilize SQL relational databases to efficiently store, manage and retrieve road accident data.

In this project, a relational database has been properly created that addresses the potential technical, ethical, and legal issues that may arise when handling accident data.

Structured query language is used to query the accident data to draw plausible insights that will improve government measures in public safety.

Also, an entity relationship diagram (ERD) is drawn which shows the relationship between various tables in the database with the appropriate primary and foreign keys.



POTENTIAL TECHNICAL ISSUES OF STORING ACCIDENT DATA



Data Integrity: Ensuring stored accident data remains accurate and consistent can pose a challenge. For example, duplicated entries, inconsistent format during entry or corrupted data.

SQL RELATIONAL DATABASE SOLUTION: Relational databases support rules. These rules, such as primary and foreign keys and unique constraints, work together to maintain data accuracy and prevent errors.



Data Security: Issues relating to storing names and restricting unauthorized access to sensitive accident data. Securing accident data and protecting its usage.

SQL RELATIONAL DATABASE SOLUTION: SQL relational database utilizes built-in encryption to regulate access to stored information.



Database Hosting: Choosing the best hosting for the accident database can pose issues concerning security and scalability.



SQL RELATIONAL DATABASE SOLUTION: On-premises hosting provides complete control and faster local network access for larger datasets.

POTENTIAL ETHICAL ISSUES OF STORING ACCIDENT DATA



Data Privacy: Data privacy is of utmost concern. Storing names of victims involved in accidents involves handling sensitive information.

SQL RELATIONAL DATABASE SOLUTION: SQL databases allows anonymisation. Masking names at the database level can prevent breach of sensitive information.



Bias and Fairness: Data collected might be biased such as over-representation of accidents from certain demographic or geographical location.

SQL RELATIONAL DATABASE SOLUTION: SQL database can be queried to detect anomalies and outliers in the accident data addressing bias.



POTENTIAL LEGAL ISSUES OF STORING ACCIDENT DATA



Data Protection and Privacy Laws: The General Data Protection Regulation in the European Union law guides the use of sensitive private information. It covers issues such as informed consent, data retention, identifiable information among others.

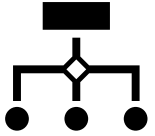
SQL RELATIONAL DATABASE SOLUTION: SQL databases allow automated data retention policies. For example, triggers or scheduled jobs can be configured to delete outdated data records after a specified duration. This allows compliance with data retention laws.

Accountability: In accident cases where legal issues can arise, it is important to keep track of who has accessed the database and when.

SQL RELATIONAL DATABASE SOLUTION: SQL databases support logging mechanisms that keeps track of who accessed or modified data. This provides an audit trail which is valuable in legal matters.



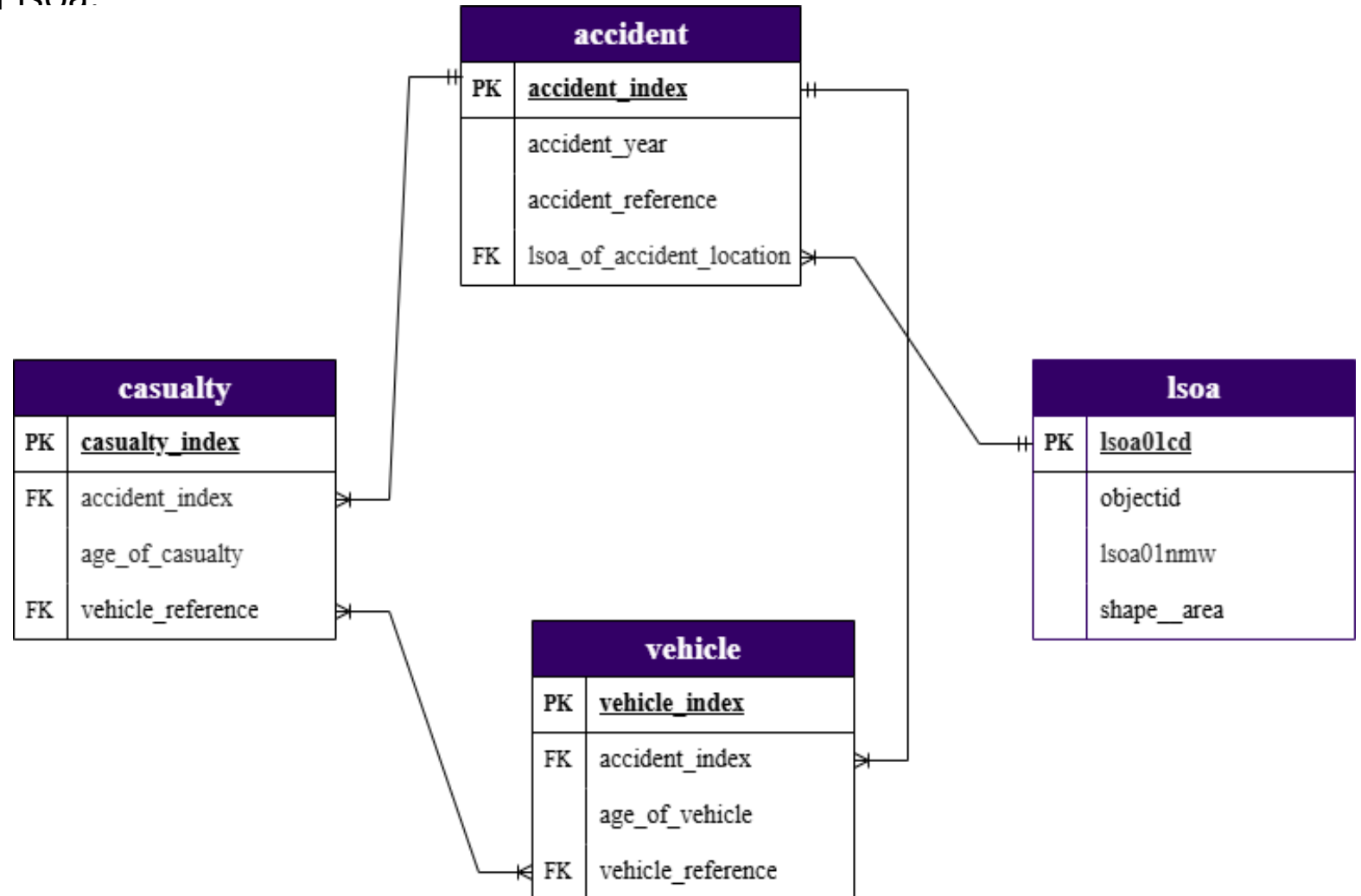
ENTITY-RELATIONSHIP DIAGRAM (ERD)



This entity relationship diagram models the relationship that exists between four tables, namely, accident, casualty, vehicle, and lsoa.

Relationship Notation (ERD)

- One-to-Many between accident and casualty
- One-to-Many between accident and vehicle
- Many-to-many between casualty and vehicle
- Many-to-One between accident and lsoa



SQL COMMANDS

- **Setting up a jupyter notebook interactive development environment to interact with the SQLite database**
- Imported the necessary libraries
- Established a connection to the SQLite database
- Extracted the table names using SQL Master
- Counted number of rows in each table.

```
In [2]: # Import Libraries
import sqlite3
import pandas as pd
import numpy as np

def library():
    print('All necessary libraries imported successfully.')

library()
```

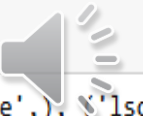
All necessary libraries imported successfully.

```
In [3]: # connect to the accident database file using SQLite
conn = sqlite3.connect('accident_data_v1.0.0_2023.db')


# instantiate cursor object from connection object 'conn' as 'cur'
# cursor allows execution of sql queries
cur = conn.cursor()
```

```
In [4]: # examine all tables in the database
table_info = cur.execute("SELECT name FROM sqlite_master WHERE type='table';")
tables = table_info.fetchall()
print(f'These are the database tables:', tables)
```

These are the database tables: [('accident',), ('casualty',), ('vehicle',), ('lsoa',)]



SQL COMMANDS

 jupyter Big Data and Data Mining Presentation Assignment Last Checkpoint: 1 hour ago

File Edit View Run Kernel Settings Help

Trusted

Python 3 (ipykernel)

print(f'These are the columns in the lsoa table:', lsoa_names)

These are the columns in the lsoa table: ['objectid', 'lsoa01cd', 'lsoa01nm', 'lsoa01nmw', 'shape__area', 'shape__length', 'globalid']

Question 1. The age of the oldest driver/rider in the casualty table

```
[ ]: # SQL command query to find age of the oldest driver in the casualty table
# Execute the query to find the age of the oldest driver in the casualty table
cur.execute("""
SELECT MAX(age_of_casualty)
FROM casualty
WHERE casualty_class = 1 AND age_of_casualty IS NOT NULL;
""") # in the accident statistics form, driver is denoted as '1' in the casualty_class section.

# executing the result using fetch() method
age_of_oldest_driver = cur.fetchone()[0]

# print results
print(f'The oldest driver is aged:', age_of_oldest_driver)
```

Question 2. The total number of vehicle_type = 19 vehicles in the vehicle table

```
[ ]: # SQL command query to extract the total number of vehicles with vehicle_type = 19
cur.execute("""
SELECT COUNT(*) AS Total_vehicles
FROM vehicle
WHERE vehicle_type = 19;
""") # in the accident statistics form, vehicle_type = 19 is a VAN - Goods vehicles 3.5 tonnes mgw and under.

# Executing the result using fetchone() method since it's expected to return a single row
total_vehicles = cur.fetchone()[0]

# Print results
print(f'The total number of vehicles with vehicle_type = 19 is:', total_vehicles)
```

Question 3: The sex of driver, sex of casualty, speed limit and age of vehicle for accidents in all the lower layer super output area (LSOA) regions of Kingston Upon Hull. You will need to do a JOIN on the lsoa table and some of the other tables. In the LSOA table, the codes are in the column lsoa01cd and the place names are in column lsoa01nm. Put the results in a pandas data frame.

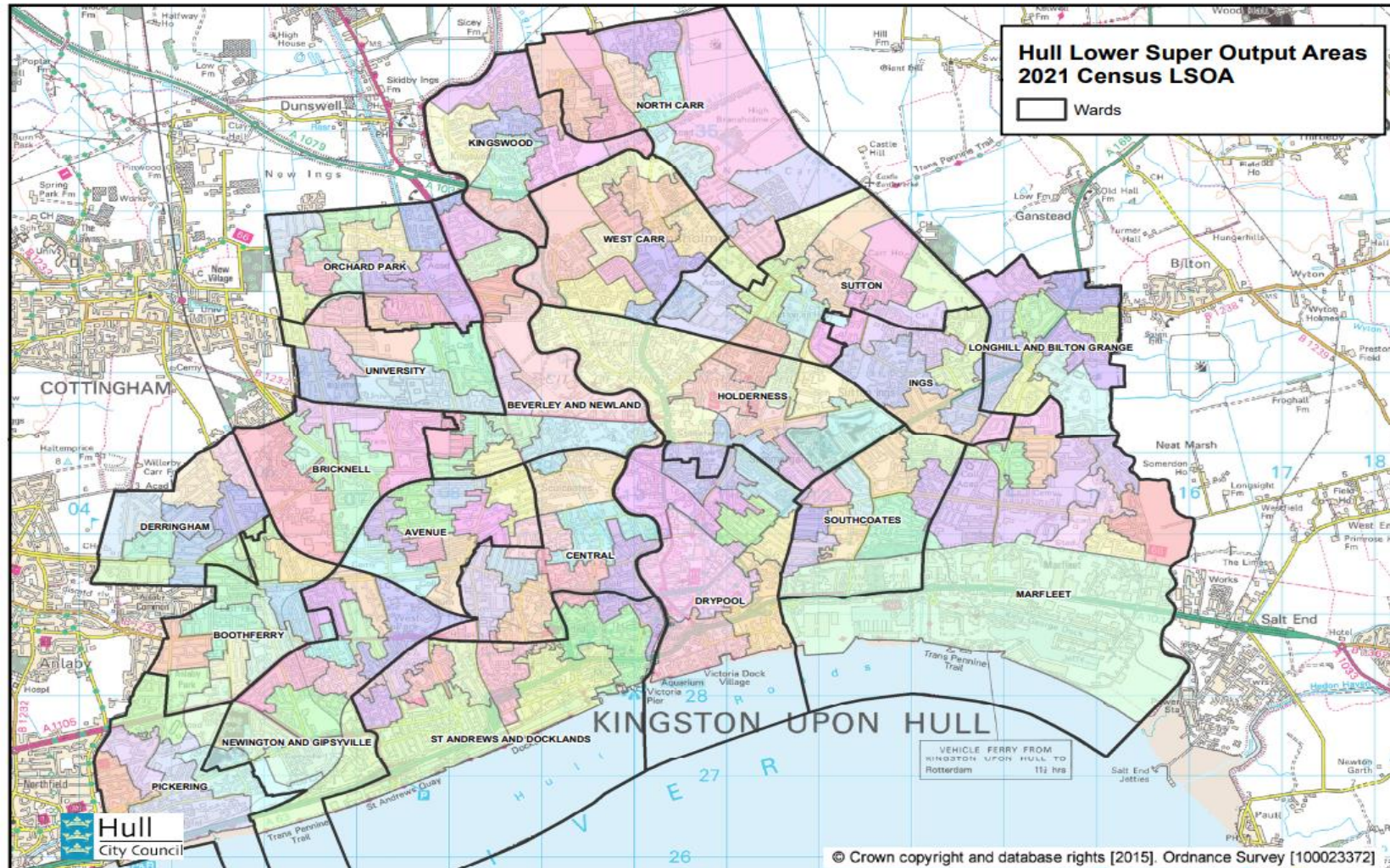
```
[ ]: # SQL query to join the necessary tables and select the required fields
query = """
SELECT
```



MAP OF LOWER LAYER SUPER OUPUT (LSOA) REGIONS OF KINGSTON UPON



I present a map of the LSOA regions of Kingston Upon Hull obtained from Hull Council to further aid our geographic understanding.



<https://data.hull.gov.uk/wp-content/uploads/2021LSOA-Map.pdf>

CONCLUSION

- In this presentation, I have discussed the potential technical, ethical, and legal issues that could arise when storing accident data and explained how an SQL relational database can address these issues.
- Also, I created an Entity-Relationship Diagram (ERD) to model the relationships between tables in the accident database.
- Furthermore, I presented a live code of analysis to extract the oldest age of the driver in the casualty column, the total number of vehicles with vehicle type = 19, and other complex queries joining all four tables to extract relevant information that can aid policy making.

