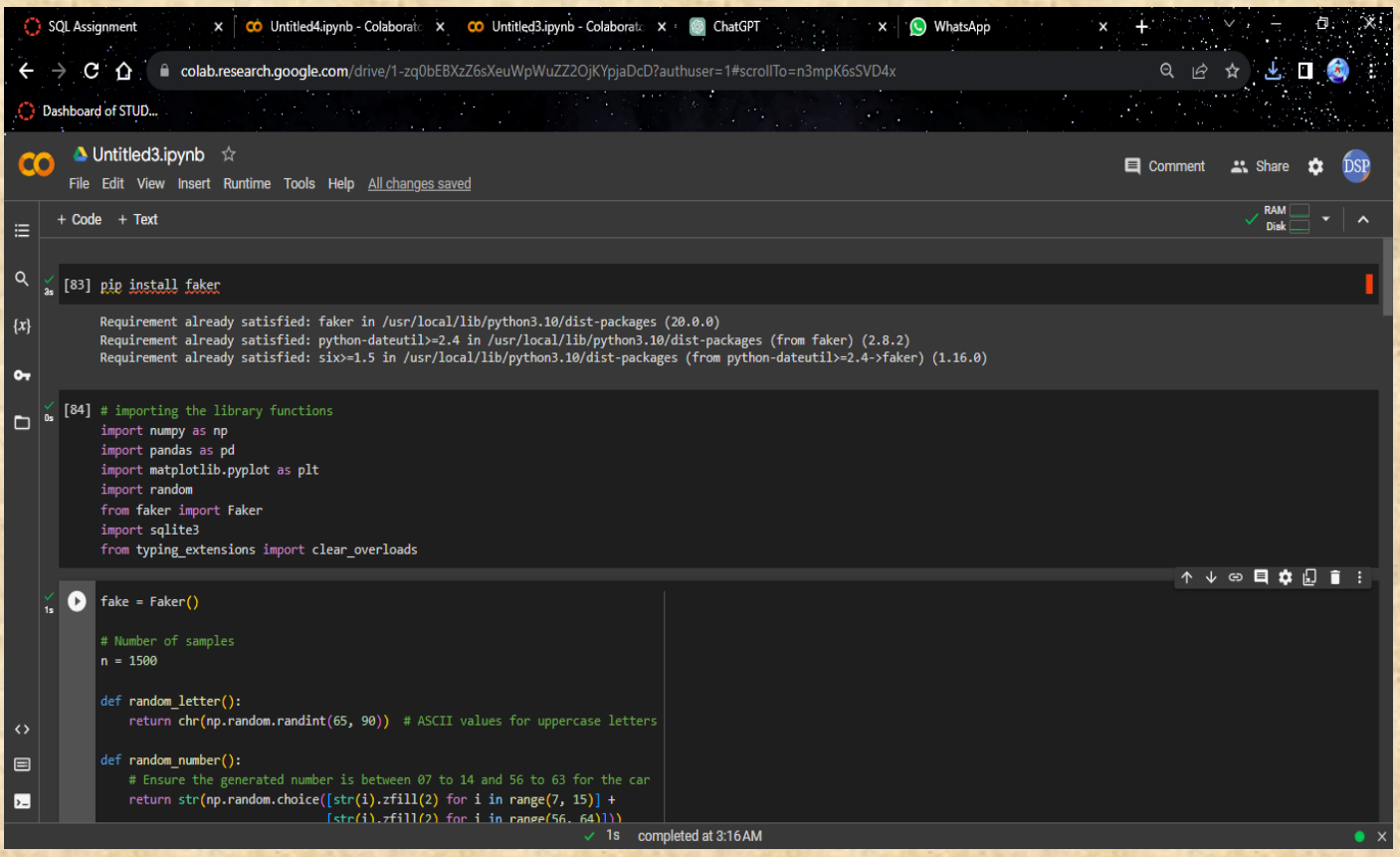
SQL ASSIGNMENT

Report on Creating a Database Schema for Synthetic Car Data

Name : **Dhavalkumar Sureshbhai Pithadiya**

Student ID : 22003118

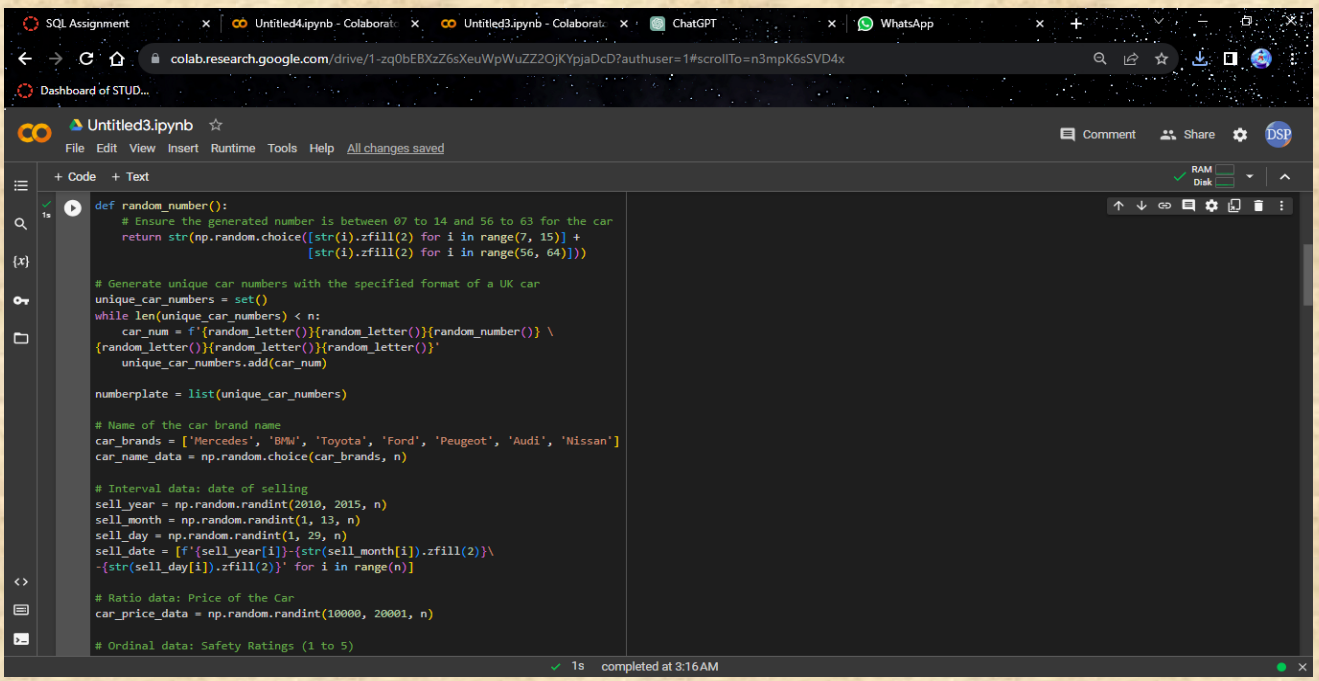
Introduction: The **aim** of this report is to document the process of generating synthetic car data using Python creating a relational database schema, and integrating the generated data into an SQLite database.



```
[83] pip install faker

Requirement already satisfied: faker in /usr/local/lib/python3.10/dist-packages (20.0.0)
Requirement already satisfied: python-dateutil>=2.4 in /usr/local/lib/python3.10/dist-packages (from faker) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.4->faker) (1.16.0)

[84] # importing the library functions
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import random
from faker import Faker
import sqlite3
from typing_extensions import clear_overloads

fake = Faker()

# Number of samples
n = 1500

def random_letter():
    return chr(np.random.randint(65, 90))  # ASCII values for uppercase letters

def random_number():
    # Ensure the generated number is between 07 to 14 and 56 to 63 for the car
    return str(np.random.choice([str(i).zfill(2) for i in range(7, 15)] +
                                [str(i).zfill(2) for i in range(56, 64)]))
```

completed at 3:16 AM

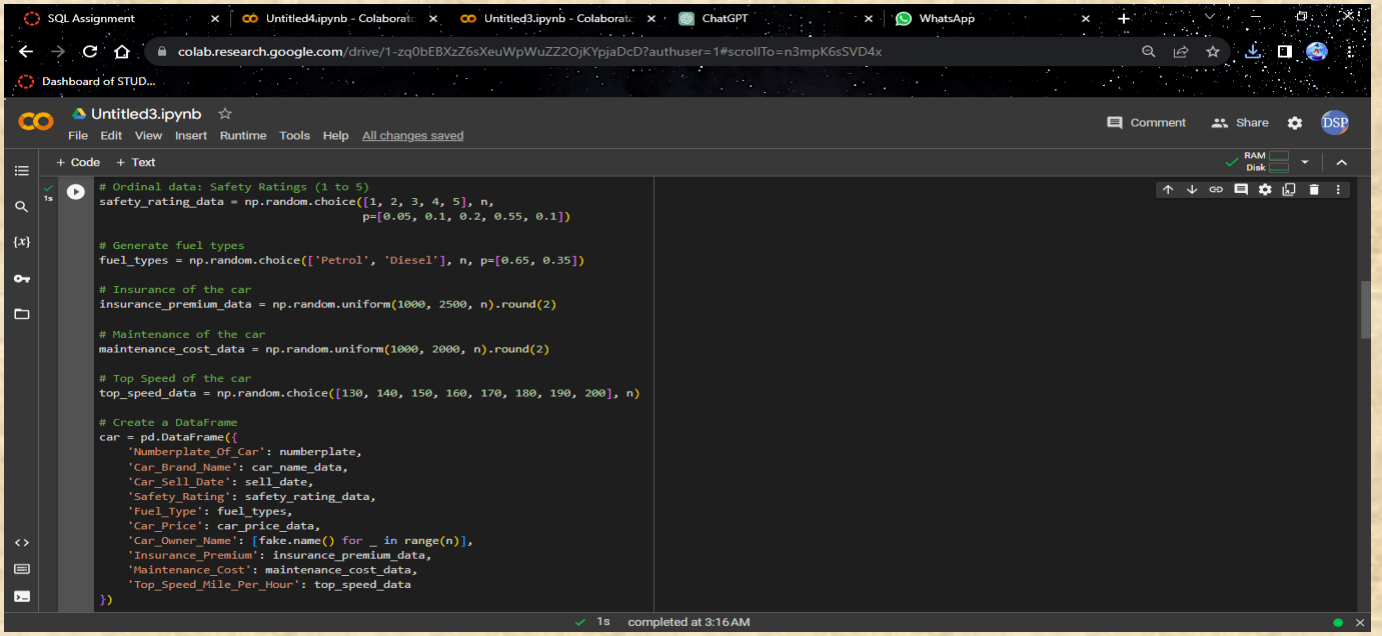
Here, I have generated data for car which was sold out in 2010 to 2015 with the help of google colab. For initial base we have installed necessary library functions. In order to get fake data we have used faker library function. To make reliable data I have taken 1200 rows and 10 columns with different parameters.



```
def random_number():
    # Ensure the generated number is between 07 to 14 and 56 to 63 for the car
    return str(np.random.choice([str(i).zfill(2) for i in range(7, 15)] +
                                [str(i).zfill(2) for i in range(56, 64)]))

# Generate unique car numbers with the specified format of a UK car
unique_car_numbers = set()
while len(unique_car_numbers) < n:
    car_num = f'{random_letter()}{random_letter()}{random_number()} \
{random_letter()}{random_letter()}{random_letter()}'
    unique_car_numbers.add(car_num)

numberplate = list(unique_car_numbers)

# Name of the car brand name
car_brands = ['Mercedes', 'BMW', 'Toyota', 'Ford', 'Peugeot', 'Audi', 'Nissan']
car_name_data = np.random.choice(car_brands, n)

# Interval data: date of selling
sell_year = np.random.randint(2010, 2015, n)
sell_month = np.random.randint(1, 13, n)
sell_day = np.random.randint(1, 29, n)
sell_date = [f'{sell_year[i]}-{str(sell_month[i]).zfill(2)}\
-{str(sell_day[i]).zfill(2)}' for i in range(n)]

# Ratio data: Price of the Car
car_price_data = np.random.randint(10000, 20001, n)

# Ordinal data: Safety Ratings (1 to 5)
```

completed at 3:16 AM

In given data I have provided all four type data Nominal, Ordinal, Interval, Ratio(NOIR). This includes details such as car unique number plates(because it can not be repeated), car brand names, selling dates, prices, safety ratings, fuel types, insurance premiums, maintenance costs, and top speeds. We can see in the below image.



DataFrame Creation and Sorting: The generated data is organized into a Pandas DataFrame named 'car.' To enhance readability, the DataFrame is sorted based selling dates in ascending order.

Futhermore,

SQLite Database Connection and Schema Creation:

A connection to an SQLite database, named 'car_database.db,' is established. The code defines three tables - 'car_data,' 'car_expense,' and 'car_owner' - incorporating foreign key relationships. This schema is designed to capture the various aspects of car-related data and expenses.

```python
# Sort DataFrame by 'Car_Sell_Date'
car.sort_values(by='Car_Sell_Date', ascending=True, inplace=True)

# Print DataFrame without index column
print(car.head(10).to_string(index=False))

# Remove NaN values
car = car.dropna()

# Connect to SQLite database
conn = sqlite3.connect('car_database.db')

# Create a cursor object to execute SQL queries
cursor = conn.cursor()

# Create tables with foreign and composite keys
cursor.execute('''
CREATE TABLE IF NOT EXISTS car_data (
    Numberplate_Of_Car TEXT PRIMARY KEY,
    Car_Brand_Name TEXT,
    Car_Sell_Date DATE,
    Safety_Rating INTEGER,
    Fuel_Type TEXT,
    Car_Price_Pounds INTEGER,
    Top_Speed_Mph INTEGER
);
''')

cursor.execute('''
```

Following this we have to make transformation, which is

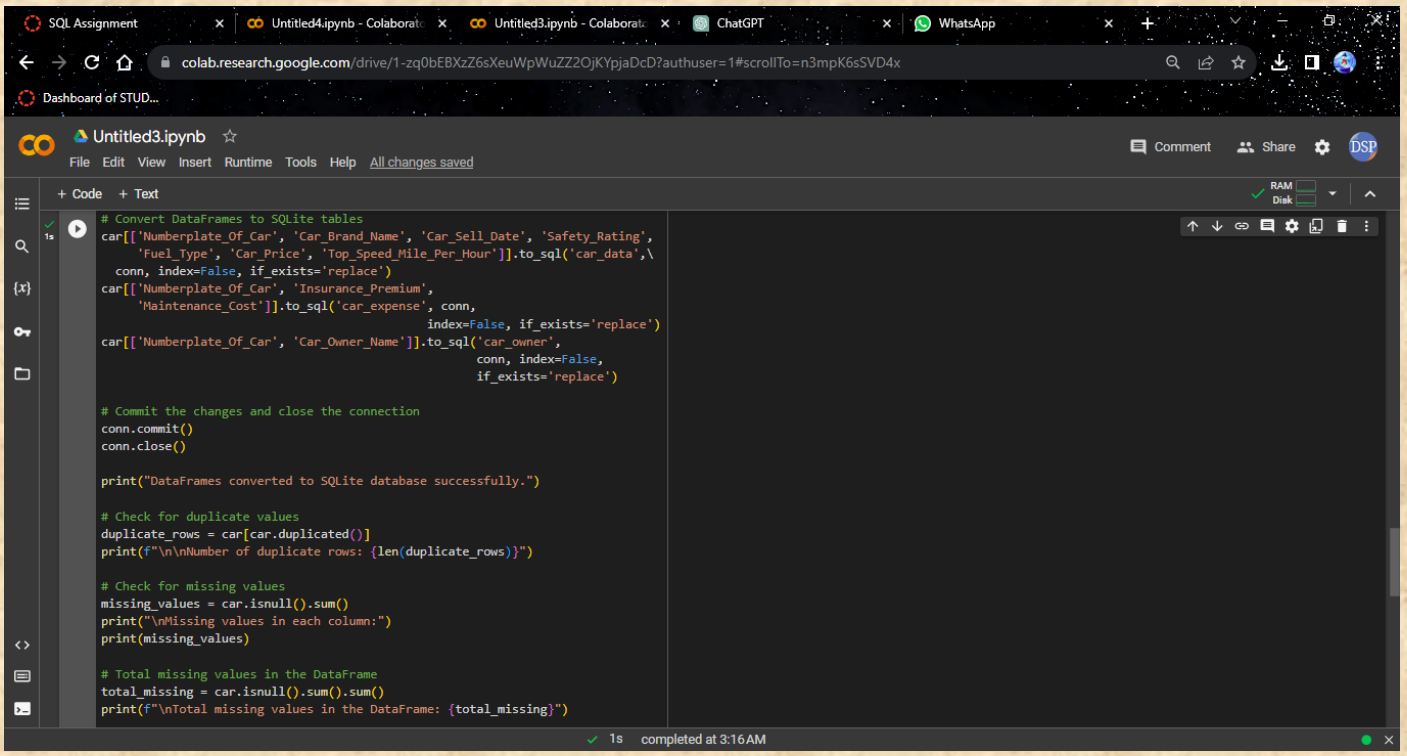
```python
cursor.execute('''
CREATE TABLE IF NOT EXISTS car_expense (
    Numberplate_Of_Car TEXT PRIMARY KEY,
    Insurance_Premium_Pounds REAL,
    Maintenance_Cost_Pounds REAL,
    FOREIGN KEY (Numberplate_Of_Car) REFERENCES car_data(Numberplate_Of_Car)
);
''')

cursor.execute('''
CREATE TABLE IF NOT EXISTS car_owner (
    Numberplate_Of_Car TEXT PRIMARY KEY,
    Car_Owner_Name TEXT,
    FOREIGN KEY (Numberplate_Of_Car) REFERENCES car_data(Numberplate_Of_Car)
);
''')

# Commit the changes and close the connection
conn.commit()
conn.close()

# Connect again for data insertion
conn = sqlite3.connect('car_database.db')
cursor = conn.cursor()

# Convert DataFrames to SQLite tables
car[['Numberplate_Of_Car', 'Car_Brand_Name', 'Car_Sell_Date', 'Safety_Rating',
     'Fuel_Type', 'Car_Price', 'Top_Speed_Mile_Per_Hour']].to_sql('car_data',\
  conn, index=False, if_exists='replace')
```

Data Insertion into the Database:

Using the SQLite connection, the code converts the Pandas DataFrames into corresponding tables in the database with use of foreign keys and composite keys as well , ensuring seamless integration of the generated car data.

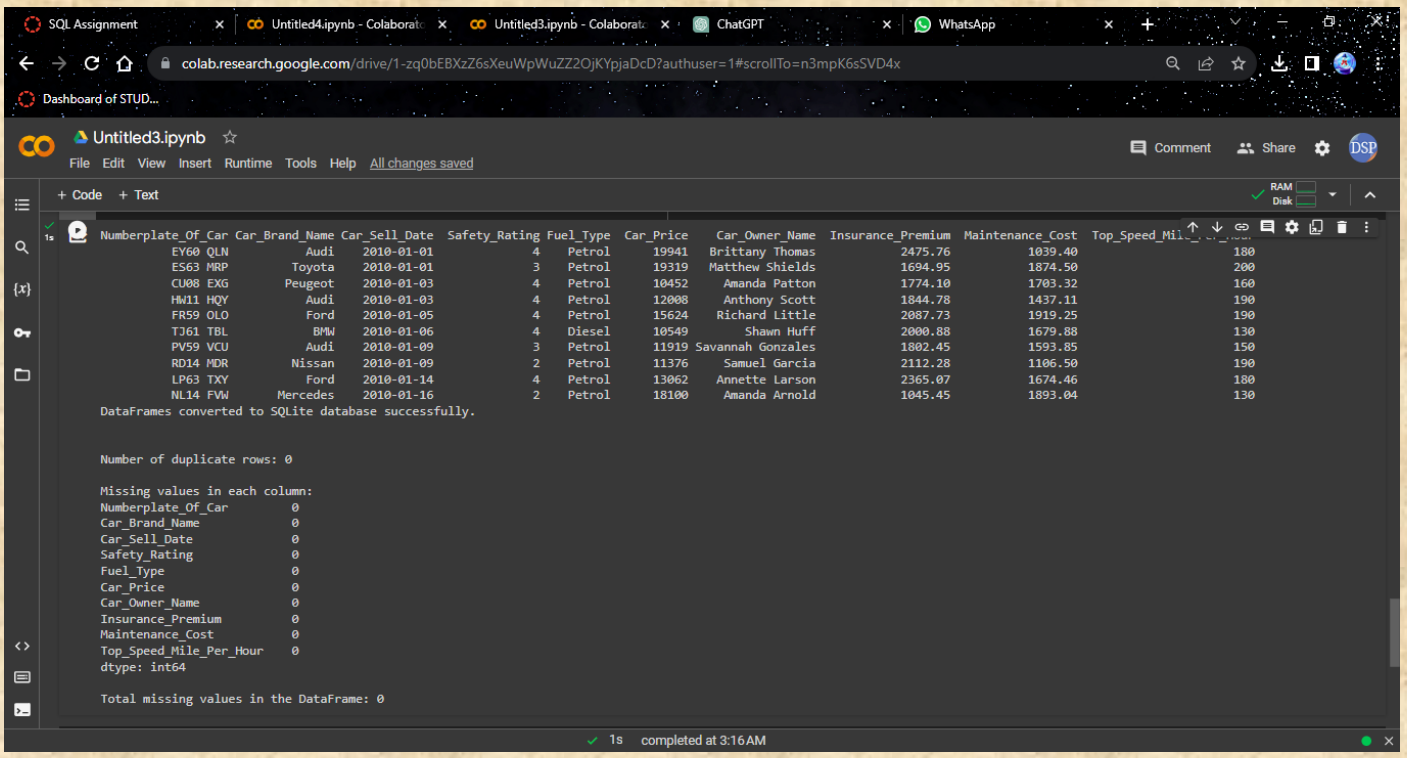
After completing that we just do ...

Data Validation: The code includes checks for data integrity, identifying and removing duplicate rows, as well as handling missing values within the DataFrame.

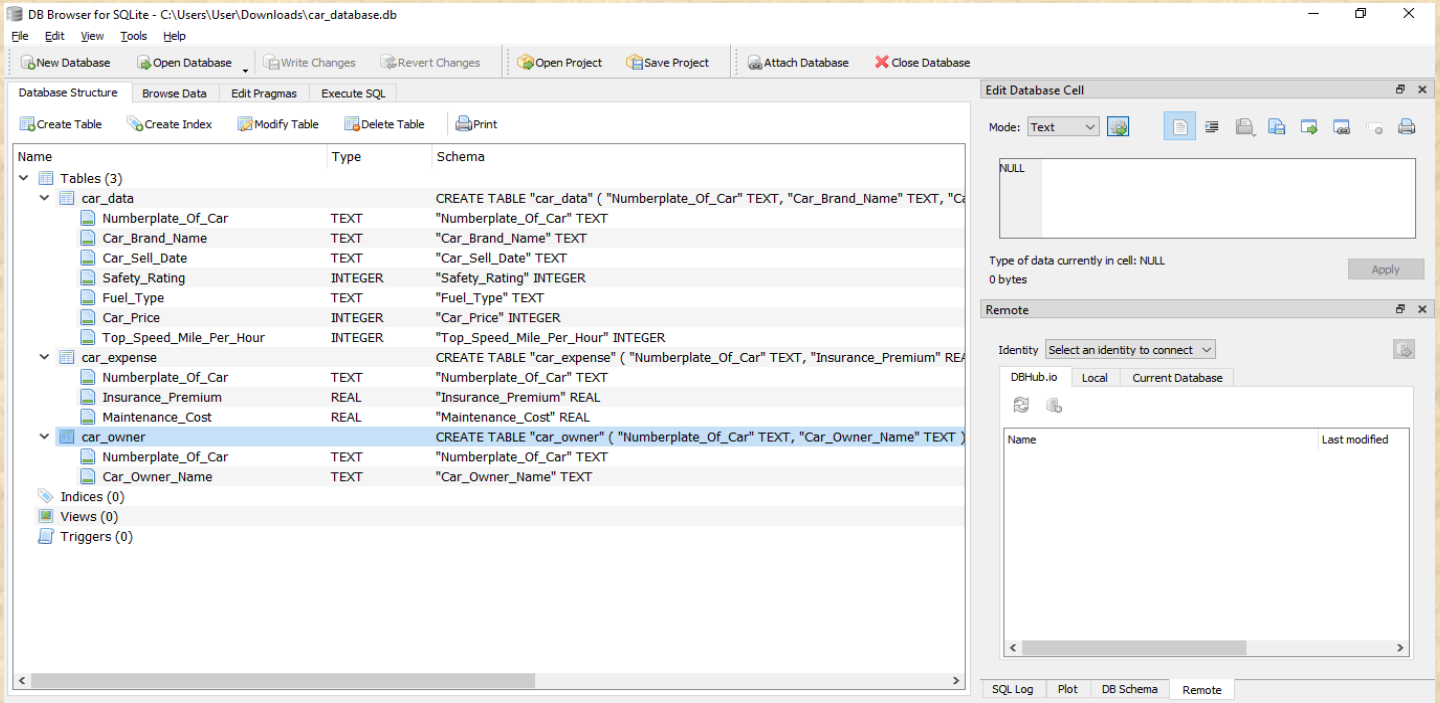


Here we got the final

Conclusion: The Python script successfully accomplishes the task of generating synthetic car data, creating a relational database schema, and populating the SQLite database. The integration of these components lays the groundwork for further analysis and querying of the car-related information.



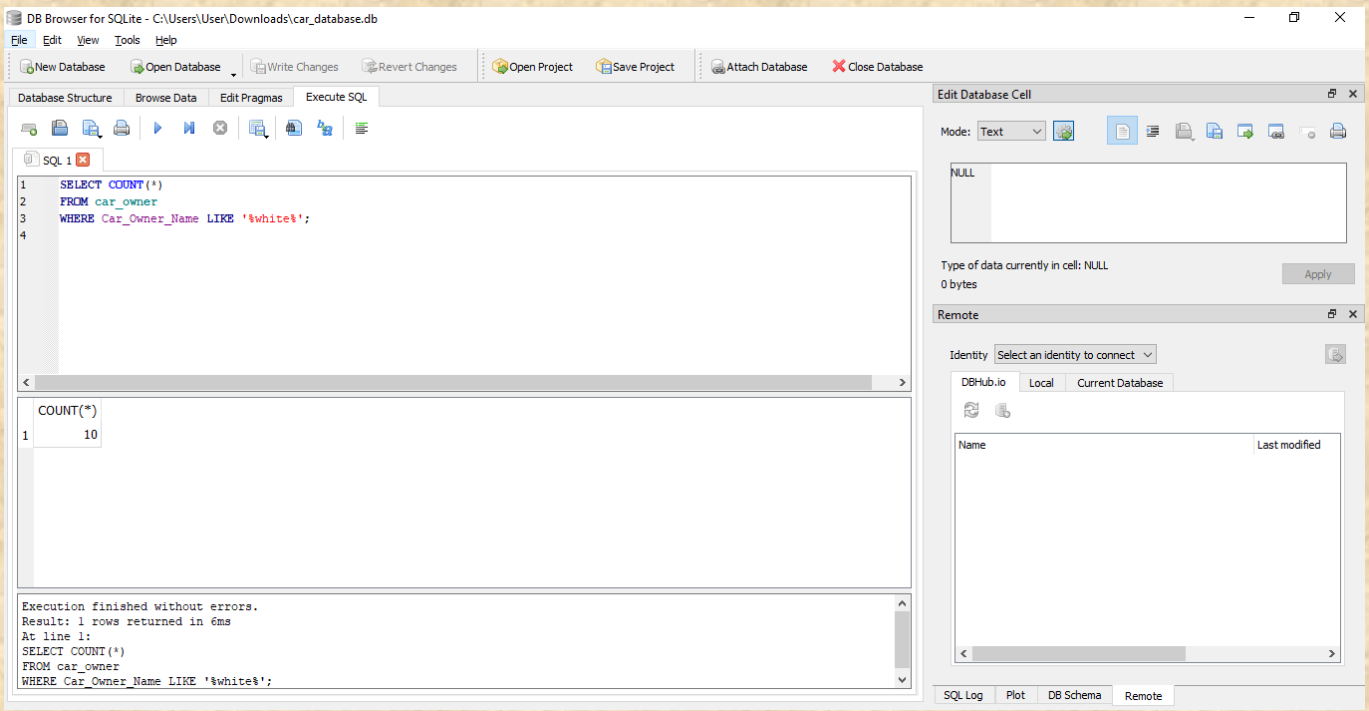
After running all over the code we generate fake car database in our programme so here database file look like as given in the below picture



Here we create 3 table for car data in that 1st table represents basic car details such as Numberplate, Car Brand, Selling price of car and date, Safety rating , fuel type and top speed , 2nd Table represent expense of insurance and maintenance and 3rd Table stands for car owner name.

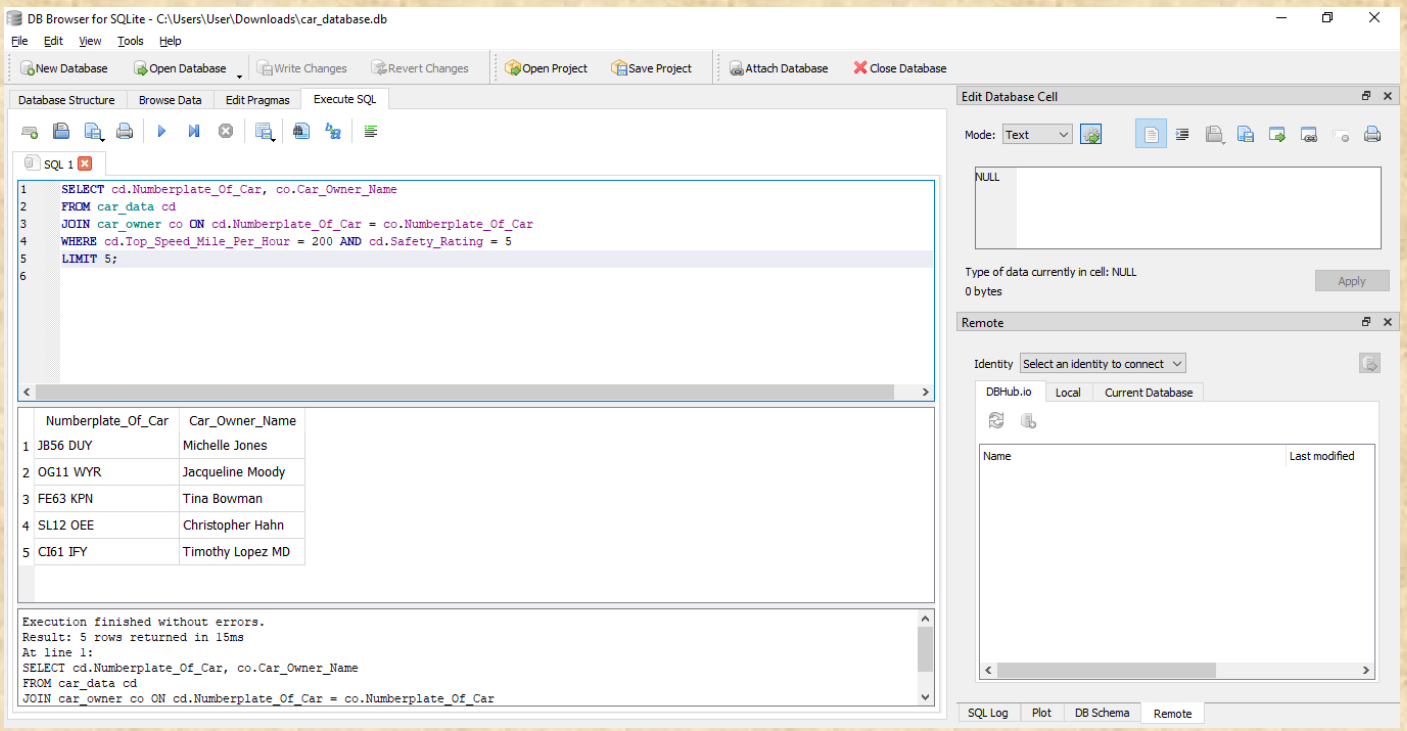
These 3 Table represent distinct values and parameters which is most useful for car information

In order to learn more about our car database schema, we can run just simple query to execute SQL. So here is small example is given in the image.

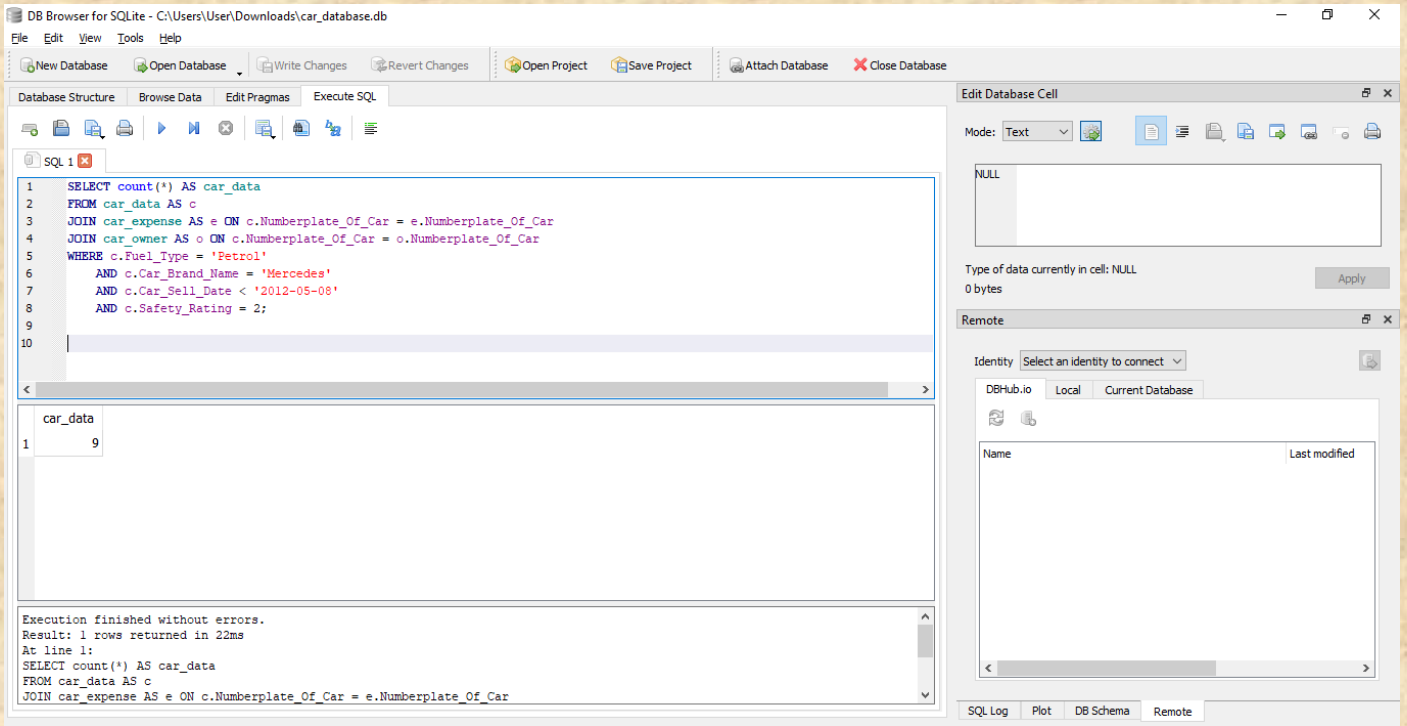


In this window we have just try to find name of the car owner with the last name white so we find in total 10 person with given surname.

Similarly, Consider another example for given car data with highest rating and top speed so if we would like to know car owner name than we can use join function to combine two table and output will be like as below:

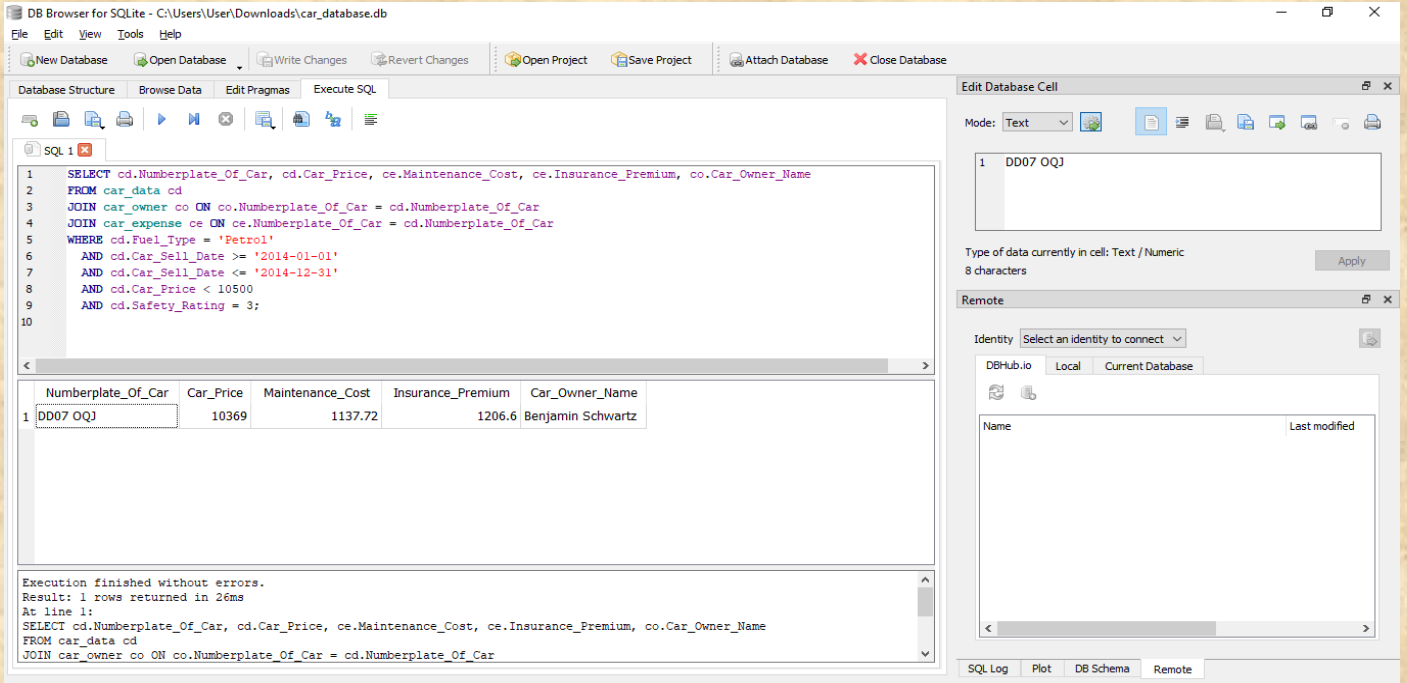


Another example over here where we can connect table with using our specified input code (like JOIN, ON, AND, WHERE) for the sql schema execute so we can get appropriate answer like below



In our given Database all of the three table has a single column which is same that is called foreign Key. In order to join table we are using nothing else but foreign key and This query selects common value information for a petrol car sold in 2014, with a price less

than £10500 and a safety rating of 3. We can Adjust the date range according to our specific requirements.



For Coding and Database file checkout Github repository [Over Here.]