SQL ASSIGNMENT

Name : PAVAN KUMAR KURVA
Student ID : 22065713

INTRODUCTION:

The process of data generation is a fundamental step in populating databases with relevant and meaningful information. In this context, a structured approach is essential to ensure data integrity, efficiency, and ethical considerations. This document outlines the systematic data generation process along with the database schema and ethical considerations involved.

DATA GENERATION PROCESS:

1**. IMPORT LIBRARIES:** The code imports `sqlite3` for database operations and `Faker` for generating fake data.

2. CONNECT TO DATABASE: It establishes a connection to an SQLite database named `shoe_database.db`.

3. DEFINE SCHEMA: The code defines two tables: `Brands` and `Shoes`, specifying their fields and relationships.

4. GENERATE RANDOM DATA: Using `Faker`, it generates fake data for brands and shoes, including brand name, country, founding year, CEO, revenue, headquarters, shoe name, shoe type, shoe size, price, and rating.

5. INSERT DATA: It inserts the generated data into the respective tables, ensuring that duplicate brands are not inserted.

6. COMMIT AND CLOSE CONNECTION: Finally, it commits the changes to the database and closes the connection.

DATABASE SCHEMA:

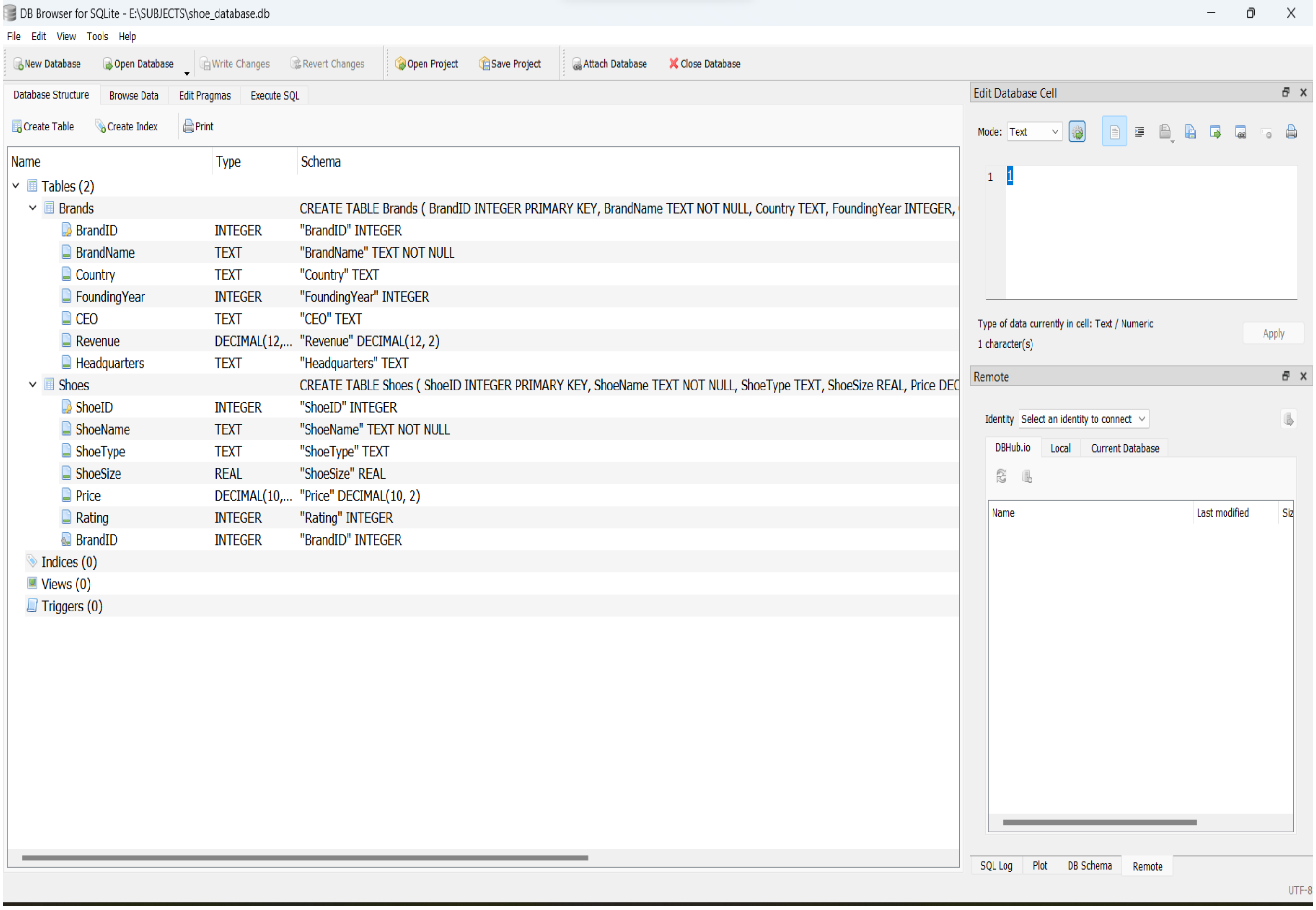
1. BRANDS TABLE:

- `BrandID` (Primary Key): Unique identifier for each brand.
- `BrandName`: Name of the brand.
- `Country`: Country where the brand originated.
- `FoundingYear`: Year the brand was founded.
- `CEO`: CEO of the brand.
- `Revenue`: Annual revenue of the brand.
- `Headquarters`: Location of the brand's headquarters.
- `UNIQUE (BrandName, Country) `: Compound unique key constraint to prevent duplicate brand entries.

2. SHOES TABLE:

- `ShoeID` (Primary Key): Unique identifier for each shoe.
- `ShoeName`: Name of the shoe.
- `ShoeType`: Type or category of the shoe.
- `ShoeSize`: Size of the shoe.
- `Price`: Price of the shoe.
- `Rating`: Rating of the shoe.

- `BrandID` (Foreign Key): References the `BrandID` in the Brands table, establishing a relationship between brands and shoes.



Type of data currently in cell: Text / Numeric
1 character(s)

Name	Type	Schema
Tables (2)		
Brands		CREATE TABLE Brands (BrandID INTEGER PRIMARY KEY, BrandName TEXT NOT NULL, Country TEXT, FoundingYear INTEGER,
BrandID	INTEGER	"BrandID" INTEGER
BrandName	TEXT	"BrandName" TEXT NOT NULL
Country	TEXT	"Country" TEXT
FoundingYear	INTEGER	"FoundingYear" INTEGER
CEO	TEXT	"CEO" TEXT
Revenue	DECIMAL(12,...	"Revenue" DECIMAL(12, 2)
Headquarters	TEXT	"Headquarters" TEXT
Shoes		CREATE TABLE Shoes (ShoeID INTEGER PRIMARY KEY, ShoeName TEXT NOT NULL, ShoeType TEXT, ShoeSize REAL, Price DEC
ShoeID	INTEGER	"ShoeID" INTEGER
ShoeName	TEXT	"ShoeName" TEXT NOT NULL
ShoeType	TEXT	"ShoeType" TEXT
ShoeSize	REAL	"ShoeSize" REAL
Price	DECIMAL(10,...	"Price" DECIMAL(10, 2)
Rating	INTEGER	"Rating" INTEGER
BrandID	INTEGER	"BrandID" INTEGER
Indices (0)		
Views (0)		
Triggers (0)		

JUSTIFICATION FOR SEPARATE TABLES AND ETHICAL DISCUSSION:

1. **SEPARATE TABLES:** The decision to split data into separate tables for brands and shoes follows the principles of database normalization. Each table represents a distinct entity with its attributes, reducing data redundancy and improving data integrity. This separation also enables efficient querying and management of related data.

2. **ETHICAL CONSIDERATION:** Generating fake data ensures privacy and confidentiality, as real-world data may contain sensitive information about individuals or companies. Additionally, by not using real data, there is no risk of inadvertently disclosing proprietary or personal information. However, it's important to ensure that the generated data does not resemble or mimic real data closely to avoid any potential confusion or misuse.

EXAMPLE QUERIES:

1. SELECTING SHOE INFORMATION:

```
SELECT ShoeName, ShoeType, ShoeSize, Price, Rating
FROM Shoes
WHERE Price > 200.0 AND Rating >= 4
ORDER BY Price DESC;
```

2. JOINING TABLES TO GET BRAND INFORMATION:

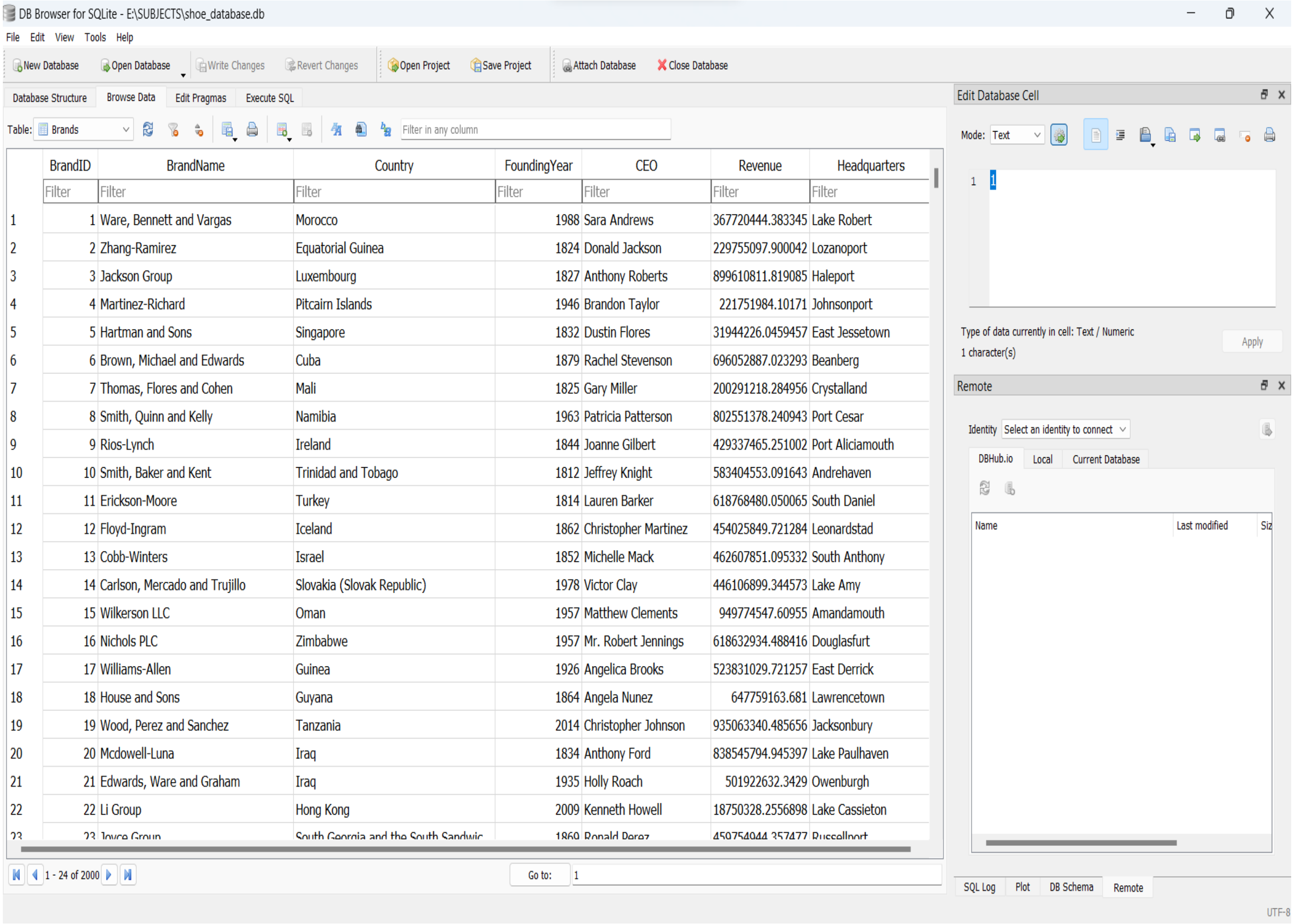
```
SELECT s.ShoeName, s.ShoeType, s.Price, b.BrandName, b.Country
FROM Shoes s
INNER JOIN Brands b ON s.BrandID = b.BrandID
WHERE s.Rating = 5;
```

3. AGGREGATE FUNCTION USAGE:

```
SELECT AVG(Price) AS AvgPrice, MAX(Rating) AS MaxRating
FROM Shoes;
```

4. FILTERING BASED ON DATE:

```
SELECT BrandName, CEO, Revenue
FROM Brands
WHERE FoundingYear > 2000
ORDER BY Revenue DESC;
```



	BrandID	BrandName	Country	FoundingYear	CEO	Revenue	Headquarters
1	1	Ware, Bennett and Vargas	Morocco	1988	Sara Andrews	367720444.383345	Lake Robert
2	2	Zhang-Ramirez	Equatorial Guinea	1824	Donald Jackson	229755097.900042	Lozanoport
3	3	Jackson Group	Luxembourg	1827	Anthony Roberts	899610811.819085	Haleport
4	4	Martinez-Richard	Pitcairn Islands	1946	Brandon Taylor	221751984.10171	Johnsonport
5	5	Hartman and Sons	Singapore	1832	Dustin Flores	31944226.0459457	East Jessetown
6	6	Brown, Michael and Edwards	Cuba	1879	Rachel Stevenson	696052887.023293	Beanberg
7	7	Thomas, Flores and Cohen	Mali	1825	Gary Miller	200291218.284956	Crystalland
8	8	Smith, Quinn and Kelly	Namibia	1963	Patricia Patterson	802551378.240943	Port Cesar
9	9	Rios-Lynch	Ireland	1844	Joanne Gilbert	429337465.251002	Port Aliciamouth
10	10	Smith, Baker and Kent	Trinidad and Tobago	1812	Jeffrey Knight	583404553.091643	Andrehaven
11	11	Erickson-Moore	Turkey	1814	Lauren Barker	618768480.050065	South Daniel
12	12	Floyd-Ingram	Iceland	1862	Christopher Martinez	454025849.721284	Leonardstad
13	13	Cobb-Winters	Israel	1852	Michelle Mack	462607851.095332	South Anthony
14	14	Carlson, Mercado and Trujillo	Slovakia (Slovak Republic)	1978	Victor Clay	446106899.344573	Lake Amy
15	15	Wilkerson LLC	Oman	1957	Matthew Clements	949774547.60955	Amandamouth
16	16	Nichols PLC	Zimbabwe	1957	Mr. Robert Jennings	618632934.488416	Douglasfurt
17	17	Williams-Allen	Guinea	1926	Angelica Brooks	523831029.721257	East Derrick
18	18	House and Sons	Guyana	1864	Angela Nunez	647759163.681	Lawrencetown
19	19	Wood, Perez and Sanchez	Tanzania	2014	Christopher Johnson	935063340.485656	Jacksonbury
20	20	Mcdowell-Luna	Iraq	1834	Anthony Ford	838545794.945397	Lake Paulhaven
21	21	Edwards, Ware and Graham	Iraq	1935	Holly Roach	501922632.3429	Owenburgh
22	22	Li Group	Hong Kong	2009	Kenneth Howell	18750328.2556898	Lake Cassieton
23	23	Joyce Group	South Georgia and the South Sandwic	1869	Ronald Perez	459754944.357477	Russellport

These example queries demonstrate different types of SQL operations such as selection, joining, aggregation, and filtering, showcasing the versatility of the database schema and the types of analysis that can be performed on the data.

EXECUTED CODE:

```python
import sqlite3
from faker import Faker

fake = Faker()

conn = sqlite3.connect('shoe_database.db')
c = conn.cursor()

# Create Brands table
c.execute('''CREATE TABLE IF NOT EXISTS Brands (
               BrandID INTEGER PRIMARY KEY,
               BrandName TEXT NOT NULL,
               Country TEXT,
               FoundingYear INTEGER,
               CEO TEXT,
               Revenue DECIMAL(12, 2),
               Headquarters TEXT,
               UNIQUE (BrandName, Country) -- Compound Unique Key
            )''')

# Create Shoes table
c.execute('''CREATE TABLE IF NOT EXISTS Shoes (
               ShoeID INTEGER PRIMARY KEY,
               ShoeName TEXT NOT NULL,
               ShoeType TEXT,
               ShoeSize REAL,
               Price DECIMAL(10, 2),
               Rating INTEGER,
               BrandID INTEGER,
               FOREIGN KEY (BrandID) REFERENCES Brands(BrandID)
            )''')

# Generate random data for Brands and Shoes
for _ in range(1001):
    brand_name = fake.company()
    country = fake.country()
    founding_year = fake.random_int(min=1800, max=2022)
    ceo = fake.name()
    revenue = fake.random.uniform(1000000.0, 1000000000.0)
    headquarters = fake.city()

    c.execute("INSERT OR IGNORE INTO Brands (BrandName, Country, FoundingYear, CEO, Revenue, Headquarters) VALUES (?, ?, ?, ?, ?, ?)",
              (brand_name, country, founding_year, ceo, revenue, headquarters))

    brand_id = c.lastrowid  # Retrieve the last inserted row id

    shoe_name = fake.catch_phrase()
    shoe_type = fake.word()
    shoe_size = fake.random.uniform(6.0, 13.0)
    price = fake.random.uniform(50.0, 300.0)
    rating = fake.random_int(min=1, max=5)
    c.execute("INSERT INTO Shoes (ShoeName, ShoeType, ShoeSize, Price, Rating, BrandID) VALUES (?, ?, ?, ?, ?, ?)",
              (shoe_name, shoe_type, shoe_size, price, rating, brand_id))

conn.commit()
conn.close()
```