

Data Mining and Discovery Assignment

Generating and Populating a Grocery Store Database with Sample Data

Name: Vyjayanthi Kadapanatham

Student id: 22075064

Git Hub: <https://github.com/vyjayanthik/Data-Mining-and-Discovery->

Introduction:

The following Python script illustrates the creation and population of an SQLite database tailored for a virtual grocery store environment. The script sets up three key tables: Products, Customers, and Transactions, each playing a pivotal role in simulating a typical grocery store's operation. By generating sample data for products, customers, and transactions, this script offers a practical demonstration of database management in a retail setting. Leveraging randomized data generation techniques, it ensures a diverse and dynamic dataset, laying the groundwork for comprehensive analysis and exploration of grocery store operations and customer behaviour.

Data Generation:

```
import sqlite3
import random

# Connect to SQLite database
conn = sqlite3.connect('grocery1_store.db')
cursor = conn.cursor()

# Drop tables if they exist
cursor.execute(''DROP TABLE IF EXISTS Products'')
cursor.execute(''DROP TABLE IF EXISTS Customers'')
cursor.execute(''DROP TABLE IF EXISTS Transactions'')
```

The provided Python script demonstrates the utilization of SQLite in creating and managing a database tailored for a hypothetical grocery store. It begins by establishing a connection to the database file named "grocery1_store.db" and subsequently drops any existing tables associated with Products, Customers, and Transactions to ensure a clean slate. By executing SQL commands, the script sets up the schema for these tables, defining their respective fields such as product details, customer information, and transaction records. This process allows for the creation of a structured database environment that can efficiently store and organize data relevant to a grocery store's operations. Using Python's SQLite library, the script facilitates seamless interaction with the database, enabling further analysis and manipulation of the stored information for various business insights and decision-making purposes.

Database Schema:

The database schema encompasses three tables: Products, Customers, and Transactions. The Products table stores details such as product name, category, price, and available quantity. Customer information, including names, contact details, and addresses, is stored in the Customers table. The Transactions table tracks individual transactions, linking customers and products via foreign keys while recording transaction date, quantity, and total price. This schema provides a structured foundation for managing and analysing data within a virtual grocery store environment.

Products Table:

```
# Create Products table
cursor.execute('''CREATE TABLE IF NOT EXISTS Products (
    product_id INTEGER PRIMARY KEY,
    product_name TEXT NOT NULL,
    category TEXT,
    price REAL,
    quantity INTEGER
)''')
```

Columns:

- product_id (Integer, Primary Key): Unique identifier for each product.
- product_name (Text, Not Null): Name of the product.
- category (Text): Category to which the product belongs (e.g., Fruits, Vegetables, Dairy).
- price (Real): Price of the product.
- quantity (Integer): Quantity of the product available in stock.

Customers Table:

```
# Create Customers table with additional columns
cursor.execute('''CREATE TABLE IF NOT EXISTS Customers (
    customer_id INTEGER PRIMARY KEY,
    first_name TEXT NOT NULL,
    last_name TEXT NOT NULL,
    email TEXT,
    phone_number TEXT,
    address TEXT,
    city TEXT
)''')
```

Columns:

- customer_id (Integer, Primary Key): Unique identifier for each customer.
- first_name (Text, Not Null): First name of the customer.
- last_name (Text, Not Null): Last name of the customer.
- email (Text): Email address of the customer.
- phone_number (Text): Phone number of the customer.
- address (Text): Address of the customer.
- city (Text): City of the customer.

Transactions Table:

```
# Create Transactions table
cursor.execute('''CREATE TABLE IF NOT EXISTS Transactions (
    transaction_id INTEGER PRIMARY KEY,
    customer_id INTEGER,
    product_id INTEGER,
    transaction_date DATE,
    quantity INTEGER,
    total_price REAL,
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),
    FOREIGN KEY (product_id) REFERENCES Products(product_id)
)''')
```

Columns:

- transaction_id (Integer, Primary Key): Unique identifier for each transaction.
- customer_id (Integer): Foreign key referencing the 'customer_id' in the Customers table.
- product_id (Integer): Foreign key referencing the 'product_id' in the Products table.
- transaction_date (Date): Date of the transaction.
- quantity (Integer): Quantity of the product purchased in the transaction.
- total_price (Real): Total price of the transaction.

Justification and Ethical Discussion:

The decision to utilize separate tables for Products, Customers, and Transactions in the provided database schema adheres to the principles of database normalization, ensuring efficient data management and retrieval. This approach minimizes data redundancy and maintains data integrity by organizing related information into distinct entities. Ethical considerations in data management for a grocery store encompass the protection of customer privacy and data security, necessitating responsible handling of sensitive information and implementation of measures to prevent unauthorized access and data breaches. Additionally, ethical data analysis practices should be upheld, ensuring transparency and respect for customer privacy while leveraging data insights to enhance the shopping experience and mitigate potential ethical concerns.

Example Queries:

Query 1:

```
1 SELECT * FROM Products WHERE category = 'Fruits';
2
```

| | product_id | product_name | category | price | quantity |
|---|------------|--------------|----------|-------|----------|
| 1 | 1 | Apple | Fruits | 1.99 | 100 |
| 2 | 2 | Banana | Fruits | 0.99 | 150 |

The query "SELECT * FROM Products WHERE category = 'Fruits';" retrieves all products from the "Fruits" category in the database. This query helps to identify and analyse the fruit products available in the grocery store inventory. It provides a concise overview of the fruits offered, including their names, categories, prices, and quantities available for sale. This

information can be useful for inventory management, pricing strategies, and understanding customer preferences for fruits.

Query 2:

```
1 SELECT Products.product_name, SUM(Transactions.total_price) AS total_sales
2 FROM Transactions
3 JOIN Products ON Transactions.product_id = Products.product_id
4 GROUP BY Products.product_name;
5
```

| | product_name | total_sales |
|---|----------------|-------------|
| 1 | Apple | 342.28 |
| 2 | Banana | 182.16 |
| 3 | Carrot | 85.5 |
| 4 | Cheese | 1185.0 |
| 5 | Chicken Breast | 1557.4 |

The query calculates the total sales for each product by summing up the total price of transactions associated with each product. It joins the Transactions and Products tables based on the product_id column, then groups the results by product_name. The result provides a summary of the total sales generated by each product in the grocery store. This information can help identify the best-selling products, assess product popularity, and inform inventory management and marketing strategies.

Query 3:

```
1 SELECT Transactions.transaction_id, Transactions.transaction_date, Customers.first_name, Customers.last_name, Products.product_name, Transactions.quantity, Transactions.total_price
2 FROM Transactions
3 JOIN Products ON Transactions.product_id = Products.product_id
4 JOIN Customers ON Transactions.customer_id = Customers.customer_id
5 WHERE Transactions.transaction_date = '2024-03-03';
6
7
```

| | transaction_id | transaction_date | first_name | last_name | product_name | quantity | total_price |
|---|----------------|------------------|------------|-----------|----------------|----------|-------------|
| 1 | 1 | 2024-03-03 | Olivia | John | Chicken Breast | 10 | 59.9 |
| 2 | 2 | 2024-03-03 | Jane | Michael | Tomato | 7 | 5.25 |
| 3 | 3 | 2024-03-03 | Emily | Emily | Tomato | 6 | 4.5 |
| 4 | 4 | 2024-03-03 | Sophia | Jane | Milk | 3 | 7.5 |
| 5 | 5 | 2024-03-03 | Michael | Sophia | Tomato | 2 | 1.5 |

The query retrieves transaction details for all purchases made on March 3, 2024. It joins the Transactions table with the Products and Customers tables using their respective IDs to fetch additional details such as product names and customer names. The result includes transaction ID, date, customer first and last names, product name, quantity purchased, and total price for each transaction made on the specified date. This information can be useful for analysing sales performance on a particular day, understanding customer buying behaviour, and identifying popular products.

Query 4:

```
1 SELECT * FROM Customers WHERE city = 'New York';
2
```

| | customer_id | first_name | last_name | email | phone_number | address | city |
|---|-------------|------------|-----------|-----------------------------|--------------|---------------|----------|
| 1 | 3 | Emily | Emily | emily.emily@yahoo.com | 715-255-5353 | 242 Main St | New York |
| 2 | 13 | John | Sophia | john.sophia@yahoo.com | 506-513-5896 | 522 Elm St | New York |
| 3 | 27 | William | Michael | william.michael@outlook.com | 675-290-7631 | 139 Maple Ave | New York |
| 4 | 32 | Sophia | John | sophia.john@gmail.com | 714-805-9856 | 459 Cedar St | New York |
| 5 | 35 | Emily | Sophia | emily.sophia@hotmail.com | 324-820-5326 | 693 Cedar St | New York |

The query retrieves all customer records from the "Customers" table where the city is specified as "New York". This provides a concise list of customers residing in New York City, allowing for targeted analysis or communication with customers in that geographical area. This information can be valuable for understanding customer demographics, conducting marketing campaigns, and optimizing services tailored to the New York City customer base.

Conclusion:

| | customer_id | first_name | last_name | email | phone_number | address | city |
|----|-------------|------------|-----------|-----------------------------|--------------|---------------|--------------|
| | Filter | Filter | Filter | Filter | Filter | Filter | Filter |
| 1 | 1 | James | John | james.john@hotmail.com | 595-337-2287 | 727 Cedar St | Phoenix |
| 2 | 2 | John | William | john.william@yahoo.com | 663-512-3710 | 148 Cedar St | Houston |
| 3 | 3 | Emily | Emily | emily.emily@yahoo.com | 715-255-5353 | 242 Main St | New York |
| 4 | 4 | William | James | william.james@outlook.com | 820-462-3597 | 996 Broadway | Phoenix |
| 5 | 5 | Olivia | John | olivia.john@outlook.com | 695-408-6919 | 652 Main St | Houston |
| 6 | 6 | Olivia | Jane | olivia.jane@gmail.com | 111-497-9549 | 817 Broadway | Philadelphia |
| 7 | 7 | Emily | Jane | emily.jane@hotmail.com | 796-448-2334 | 381 Cedar St | Chicago |
| 8 | 8 | Olivia | Emily | olivia.emily@yahoo.com | 444-292-6745 | 299 Oak St | Los Angeles |
| 9 | 9 | Olivia | John | olivia.john@gmail.com | 151-699-9335 | 677 Broadway | Phoenix |
| 10 | 10 | John | James | john.james@hotmail.com | 414-912-1083 | 858 Broadway | Houston |
| 11 | 11 | Olivia | William | olivia.william@yahoo.com | 414-338-4531 | 920 Cedar St | Philadelphia |
| 12 | 12 | Michael | Jane | michael.jane@hotmail.com | 941-327-9726 | 783 Maple Ave | Philadelphia |
| 13 | 13 | John | Sophia | john.sophia@yahoo.com | 506-513-5896 | 522 Elm St | New York |
| 14 | 14 | Jane | Jane | jane.jane@hotmail.com | 450-344-4484 | 726 Maple Ave | Los Angeles |
| 15 | 15 | Olivia | Emily | olivia.emily@yahoo.com | 988-861-7128 | 576 Broadway | Phoenix |
| 16 | 16 | James | Emily | james.emily@outlook.com | 800-143-3045 | 503 Cedar St | Philadelphia |
| 17 | 17 | William | John | william.john@yahoo.com | 887-239-7212 | 638 Elm St | Phoenix |
| 18 | 18 | Jane | Michael | jane.michael@hotmail.com | 680-823-3569 | 490 Maple Ave | Philadelphia |
| 19 | 19 | Michael | Olivia | michael.olivia@outlook.com | 108-310-3137 | 878 Broadway | Houston |
| 20 | 20 | Michael | Michael | michael.michael@hotmail.com | 295-324-1593 | 274 Elm St | Los Angeles |
| 21 | 21 | William | John | william.john@outlook.com | 394-202-4032 | 818 Oak St | Phoenix |
| 22 | 22 | Olivia | Sophia | olivia.sophia@outlook.com | 833-482-6478 | 998 Elm St | Chicago |
| 23 | 23 | James | Sophia | james.sophia@yahoo.com | 692-423-7287 | 522 Cedar St | Philadelphia |

The database schema developed for the grocery store simulation exhibits a structured approach to data management, adhering to principles of normalization to ensure efficiency and integrity. By separating product, customer, and transaction data into distinct tables, the schema minimizes redundancy and facilitates organized data retrieval. Ethical considerations surrounding customer privacy and data security underscore the importance of responsible data handling and analysis practices. Upholding transparency and safeguarding customer information are paramount, aligning with ethical principles in data management. Moving forward, the schema serves as a foundational framework for further analysis and optimization within the grocery store environment, emphasizing the importance of ethical data practices in fostering trust and integrity.