

Co-op Superstore Database Report

Name:- Deepak Vishwakarma

Student ID:-23035559

Github:- https://github.com/Deepakvishwakarma1/Data_mining_and_discovery.git

-:Introduction to the Data Generation Process:-

The Co-op Superstore database was generated programmatically using Python. The following libraries and techniques were employed to create realistic and comprehensive data:

- Faker Library: Used for generating names, dates, and textual data.
- Random Module: Used to assign randomized attributes like 'Category', 'Role', 'Postcode', and numeric values such as 'Price' and 'Hourly_Rate'.
- Custom Scripts: Ensured the generation of unique alphanumeric IDs (e.g., 'AC001' for Product_ID), deliberate missing values (e.g., NULL Postcode in Customers), and realistic duplicate data in fields like Product_Name and Role.

```
1 !pip install faker
2 import sqlite3
3 import random
4 from faker import Faker
5
6 # Initialize Faker for realistic data
7 fake = Faker()
8 |
9 # Connect to SQLite
10 conn = sqlite3.connect('garston_superstore.db')
11 cursor = conn.cursor()
12
```

-:The database consists of four tables:-

1. Products Table: Central repository for product details, including categories and prices.

DB Browser for SQLite - C:\Users\Deepak Vishwakarma\Downloads\garston_s

File Edit View Tools Help

New Database Open Database Write Changes Revert Change

Database Structure Browse Data Edit Pragmas Execute SQL

Table: Products

	Product_ID	Product_Name	Category	Price
	Filter	Filter	Filter	Filter
1	AC000	Coffee	Beverages	45.02
2	AC001	Soda	Beverages	48.23
3	AC002	Chips	Snacks	12.27
4	AC003	Juice	Beverages	34.59
5	AC004	Coffee	Beverages	4.88
6	AC005	Rice	Groceries	17.32
7	AC006	Cream	Dairy	48.74
8	AC007	Soda	Beverages	43.87
9	AC008	Bagel	Bakery	49.35
10	AC009	Chips	Snacks	17.26
11	AC010	Muffin	Bakery	14.63
12	AC011	Sugar	Groceries	33.36
13	AC012	Muffin	Bakery	23.66
14	AC013	Pasta	Groceries	17.19
15	AC014	Croissant	Bakery	40.89
16	AC015	Popcorn	Snacks	42.72
17	AC016	Cake	Bakery	46.66
18	AC017	Cream	Dairy	41.91
19	AC018	Sugar	Groceries	7.45
20	AC019	Cereal	Groceries	40.25
21	AC020	Water	Beverages	34.02

1 - 21 of 100

2. Employees Table: Stores workforce information like roles and wage.

DB Browser for SQLite - C:\Users\Deepak Vishwakarma\Downloads\garston_supersto

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes

Database Structure Browse Data Edit Pragmas Execute SQL

Table: Employees

	Employee_ID	Name	Role	Hourly_Rate
	Filter	Filter	Filter	Filter
1	1	Daniel Ortiz	Manager	10.51
2	2	Ashley Moore	Cashier	20.31
3	3	Anne Wells	Cashier	22.22
4	4	Lindsey Nguyen MD	Stocker	23.43
5	5	Philip Fernandez	Stocker	22.46
6	6	Joseph Cruz	Cashier	19.54
7	7	Sharon Davis	Manager	13.55
8	8	Linda Berg	Cashier	10.56
9	9	Lisa Wade	Cashier	14.23
10	10	Kimberly Wilson	Stocker	14.49
11	11	Matthew Buck	Cashier	15.19
12	12	John Jackson	Manager	16.82
13	13	Margaret Pennington	Manager	22.0
14	14	Jennifer Fritz	Manager	16.2
15	15	Tom Martinez	Manager	12.27
16	16	Robert Wells	Manager	23.18
17	17	Kimberly White	Manager	11.77
18	18	Miss Dawn Rodriguez	Stocker	20.22
19	19	Kathleen Williams	Cashier	18.52
20	20	Sean Rodriguez	Stocker	12.13
21	21	Barry Allen	Stocker	17.94

1 - 21 of 50

3. Customers Table: Captures demographic details and spending habits.

DB Browser for SQLite - C:\Users\Deepak Vishwakarma\Downloads\garston_superstore (3).db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open Project

Database Structure Browse Data Edit Pragmas Execute SQL

Table: Customers Filter in any column

	Customer ID	Name	Age_Group	Postcode
	Filter	Filter	Filter	Filter
1	CUST0001	Justin Torres	26-35	L19 16
2	CUST0002	Kevin Wilson	66+	L19 10
3	CUST0003	Kevin Dixon	36-45	L19 03
4	CUST0004	Sherry Bush	18-25	L19 17
5	CUST0005	Mrs. Samantha Harris	36-45	L19 18
6	CUST0006	Nicholas Ward	36-45	L19 13
7	CUST0007	Devin Sims	26-35	L19 17
8	CUST0008	Stacey Shepard	56-65	L19 02
9	CUST0009	Robert Roberts	56-65	L19 19
10	CUST0010	Connor Hurley	56-65	L19 20
11	CUST0011	Benjamin Ruiz	36-45	NULL
12	CUST0012	Lori Watson	66+	L19 08
13	CUST0013	Christopher Coleman	26-35	L19 19
14	CUST0014	Jasmine Burton	36-45	NULL
15	CUST0015	Madeline Hernandez	66+	L19 11
16	CUST0016	Ronald Jackson	36-45	L19 16
17	CUST0017	Jeffrey Adams	26-35	L19 07
18	CUST0018	Robin Howard	66+	L19 09
19	CUST0019	Darren Marsh	46-55	NULL
20	CUST0020	Nicolas Martinez	26-35	L19 14
21	CUST0021	Anna Ibarra	66+	L19 03

4. Sales Table: Links the other tables to record transactional data.

DB Browser for SQLite - C:\Users\Deepak Vishwakarma\Downloads\garston_superstore (3).db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo

Database Structure Browse Data Edit Pragmas Execute SQL

Table: Sales

	<u>Sale_ID</u>	<u>Product_ID</u>	<u>Employee_ID</u>	<u>Customer_ID</u>	<u>Sale_Date</u>	<u>Quantity</u>	<u>Total_Price</u>
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
22	22	AC039	31	CUST0189	2024-11-17	3	38.74
23	23	AC057	49	NULL	2024-11-09	3	151.11
24	24	AC095	41	CUST0006	2024-11-06	2	433.08
25	25	AC089	28	CUST0390	2024-11-03	8	479.87
26	26	AC068	4	CUST0055	2024-11-18	7	17.64
27	27	AC059	40	CUST0039	2024-11-07	5	449.9
28	28	AC019	35	CUST0070	2024-11-13	6	172.9
29	29	AC066	47	CUST0061	2024-11-15	5	481.74
30	30	AC065	10	CUST0221	2024-11-04	9	211.85
31	31	AC090	27	CUST0004	2024-10-25	7	402.92
32	32	AC034	21	CUST0362	2024-11-13	5	405.88
33	33	AC034	14	CUST0437	2024-11-18	5	200.19
34	34	AC008	39	CUST0200	2024-10-25	1	350.38
35	35	AC076	8	CUST0265	2024-11-13	5	366.89
36	36	AC025	3	CUST0094	2024-11-11	4	496.59
37	37	AC014	36	CUST0372	2024-11-05	6	102.35
38	38	AC053	6	CUST0230	2024-10-24	5	485.14
39	39	AC051	49	CUST0392	2024-11-12	3	29.71
40	40	AC067	28	NULL	2024-11-12	7	50.01
41	41	AC007	38	CUST0352	2024-11-18	7	173.41
42	42	AC016	35	CUST0232	2024-10-25	7	444.93

22 - 42 of 1,000

-:Schema overview of the Database:-

Graphical Representation of the data:-

Products (Product_ID, Product_Name, Category, Price)

↳ Referenced by: Sales(Product_ID)

Employees (Employee_ID, Name, Role, Hourly_Rate)

↳ Referenced by: Sales(Employee_ID)

Customers (Customer_ID, Name, Age_Group, Postcode)

↳ Referenced by: Sales(Customer_ID)

Sales (Sale_ID, Product_ID, Employee_ID, Customer_ID, Sale_Date, Quantity, Total_Price)

Name	Type	Schema
▼ Tables (5)		
▼ Customers		CREATE TABLE Customers (Customer_ID TEXT PRIMARY KEY, -- Prefixed with "CUST" Name TEXT NOT NULL, Age_Group TEXT NOT NULL, Postcode TEXT NOT NULL)
Customer_ID	TEXT	"Customer_ID" TEXT
Name	TEXT	"Name" TEXT NOT NULL
Age_Group	TEXT	"Age_Group" TEXT NOT NULL
Postcode	TEXT	"Postcode" TEXT
▼ Employees		CREATE TABLE Employees (Employee_ID INTEGER PRIMARY KEY AUTOINCREMENT, Name TEXT NOT NULL, Role TEXT NOT NULL, Hourly_Rate REAL NOT NULL CHECK("Hourly_Rate" > 0))
Employee_ID	INTEGER	"Employee_ID" INTEGER
Name	TEXT	"Name" TEXT NOT NULL
Role	TEXT	"Role" TEXT NOT NULL
Hourly_Rate	REAL	"Hourly_Rate" REAL NOT NULL CHECK("Hourly_Rate" > 0)
▼ Products		CREATE TABLE Products (Product_ID TEXT PRIMARY KEY, -- Alphanumeric ID Product_Name TEXT NOT NULL, Category TEXT NOT NULL, Price REAL NOT NULL CHECK("Price" > 0))
Product_ID	TEXT	"Product_ID" TEXT
Product_Name	TEXT	"Product_Name" TEXT NOT NULL
Category	TEXT	"Category" TEXT NOT NULL
Price	REAL	"Price" REAL NOT NULL CHECK("Price" > 0)
▼ Sales		CREATE TABLE Sales (Sale_ID INTEGER PRIMARY KEY AUTOINCREMENT, Product_ID TEXT NOT NULL, Employee_ID INTEGER NOT NULL, Customer_ID TEXT NOT NULL, Sale_Date TEXT NOT NULL, Quantity INTEGER NOT NULL CHECK("Quantity" > 0), Total_Price REAL NOT NULL CHECK("Total_Price" > 0))
Sale_ID	INTEGER	"Sale_ID" INTEGER
Product_ID	TEXT	"Product_ID" TEXT NOT NULL
Employee_ID	INTEGER	"Employee_ID" INTEGER NOT NULL
Customer_ID	TEXT	"Customer_ID" TEXT
Sale_Date	TEXT	"Sale_Date" TEXT NOT NULL
Quantity	INTEGER	"Quantity" INTEGER NOT NULL CHECK("Quantity" > 0)
Total_Price	REAL	"Total_Price" REAL NOT NULL CHECK("Total_Price" > 0)

-:Textual Schema Overview:-

- **Products Table:**

- Product_ID: TEXT, Primary Key (e.g., 'AC001').
- Product_Name: TEXT, Realistic names like 'Bread'.
- Category: TEXT, Nominal data such as 'Bakery'.
- Price: REAL, Ratio data (>0).

- **Employees Table:**

- Employee_ID: INTEGER, Primary Key (AUTOINCREMENT).
- Name: TEXT, Employee names.
- Role: TEXT, Nominal data (e.g., 'Manager').
- Hourly_Rate: REAL, Ratio data (>0).

- **Customers Table:**

- Customer_ID: TEXT, Primary Key prefixed with 'CUST' (e.g., 'CUST001').
- Name: TEXT, Customer names.
- Age_Group: TEXT, Ordinal data (e.g., '18-25').
- Postcode: TEXT, Nominal data with $\sim 5\%$ NULL values.

- **Sales Table:**

- Sale_ID: INTEGER, Primary Key (AUTOINCREMENT).
- Product_ID: TEXT, Foreign Key referencing Products.
- Employee_ID: INTEGER, Foreign Key referencing Employees.
- Customer_ID: TEXT, Foreign Key referencing Customers, with $\sim 10\%$ NULL values.
- Sale_Date: TEXT, Interval data for recent transactions.
- Quantity: INTEGER, Ratio data (>0).
- Total_Price: REAL, Ratio data (>0).

-:Justification for Separate Tables:-

- **Products Table:**
 - Stores product details centrally to avoid redundancy.
 - Categories (e.g., 'Groceries') aid in filtering and inventory analysis.
- **Employees Table:**
 - Tracks roles and wages, critical for payroll and performance evaluation.
- **Customers Table:**
 - Facilitates segmentation by 'Age_Group'.
 - Includes NULL Postcodes to simulate real-world missing data scenarios.
- **Sales Table:**
 - Links products, employees, and customers, capturing all transactional details.
 - NULL Customer_IDs represent sales to unregistered customers (~10%).

-:Ethical and Data Privacy Discussion:-

The database emphasizes ethical considerations and data privacy:

- **Synthetic Data:** All data is generated programmatically to avoid real-world identifiers.
- **Anonymization:** Attributes like 'Postcode' and 'Name' are fictional and non-unique, preventing traceability.
- **Missing Data:** NULL values are intentionally inserted for realism.
- **Excluded PII:** Sensitive data like email addresses and phone numbers is deliberately omitted.

-:Foreign and Composite Keys:-

Foreign Keys:

- Sales.Product_ID → Products.Product_ID
- Sales.Employee_ID → Employees.Employee_ID
- Sales.Customer_ID → Customers.Customer_ID

Composite Keys:

- While no explicit composite keys are implemented, foreign key relationships ensure relational integrity and prevent redundancy.

-:Data Types Representation:-

The database includes all major data types:

- **Nominal:** Product_Name, Category, Customer_ID, Postcode.
- **Ordinal:** Age_Group, Sale_ID, Employee_ID.
- **Interval:** Sale_Date.
- **Ratio:** Price, Quantity, Hourly_Rate, Total_Price.

-:Missing and Duplicate Data:-

The database deliberately incorporates missing and duplicate data:

- Missing Data:

- ~5% NULL Postcode values in Customers.
- ~10% NULL Customer_ID values in Sales.

- Duplicate Data:

- Common Role values in Employees (e.g., multiple 'Cashiers').
- Overlapping Product_Names across Categories (e.g., 'Bread' in 'Bakery').

-:Example Queries and Results:-

1. Top 10 Customers by Total Spend:-

DB Browser for SQLite - C:\Users\Deepak Vishwakarma\Downloads\garston_superstore (3).db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open Project Save

Database Structure Browse Data Edit Pragma Execute SQL

SQL 1*

```
1 SELECT Customers.Customer_ID, Customers.Name, SUM(Sales.Total_Price) AS Total_Spent
2 FROM Sales
3 JOIN Customers ON Sales.Customer_ID = Customers.Customer_ID
4 WHERE Sales.Customer_ID IS NOT NULL
5 GROUP BY Customers.Customer_ID
6 ORDER BY Total_Spent DESC
7 LIMIT 10;
```

	Customer_ID	Name	Total_Spent
1	CUST0232	Melissa Wright	1980.42
2	CUST0113	Brian West	1788.08
3	CUST0012	Lori Watson	1764.87
4	CUST0184	Amy Gomez	1621.24
5	CUST0248	Rachel Hall	1594.36
6	CUST0182	Allison Johnson	1572.09
7	CUST0364	Hailey Clark	1561.92
8	CUST0446	Aaron Mckinney	1548.21
9	CUST0332	Rhonda Robinson	1537.76
10	CUST0255	Sara Baker	1530.84

Execution finished without errors.
Result: 10 rows returned in 30ms
At line 1:
SELECT Customers.Customer_ID, Customers.Name, SUM(Sales.Total_Price) AS Total_Spent
FROM Sales
JOIN Customers ON Sales.Customer_ID = Customers.Customer_ID
WHERE Sales.Customer_ID IS NOT NULL
GROUP BY Customers.Customer_ID
ORDER BY Total_Spent DESC
LIMIT 10;

Insights:-

- Helps identify high-value customers, which can inform marketing and loyalty programs.
- Allows for segmentation of customers based on total spend, enabling personalized offers.

2. Top 5 Products by Sales Quantity:-

DB Browser for SQLite - C:\Users\Deepak Vishwakarma\Downloads\garston_superstore (3).db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open

Database Structure Browse Data Edit Pragmas Execute SQL

SQL 1*

```
11
12 '''2. Top 5 Products by Sales Quantity'''
13 SELECT Products.Product_Name, SUM(Sales.Quantity) AS Total_Quantity
14 FROM Sales
15 JOIN Products ON Sales.Product_ID = Products.Product_ID
16 GROUP BY Products.Product_Name
17 ORDER BY Total_Quantity DESC
18 LIMIT 5;
```

	Product_Name	Total_Quantity
1	Popcorn	422
2	Soda	366
3	Oil	344
4	Bagel	317
5	Bread	302

Execution finished without errors.
Result: 5 rows returned in 14ms
At line 13:
SELECT Products.Product_Name, SUM(Sales.Quantity) AS Total_Quantity
FROM Sales
JOIN Products ON Sales.Product_ID = Products.Product_ID
GROUP BY Products.Product_Name
ORDER BY Total_Quantity DESC
LIMIT 5;

Insights:-

- Helps identify the most popular products, which can inform restocking decisions and marketing campaigns.
- Insights into sales patterns, such as which categories (e.g., Bakery, Snacks) are performing best.

3. Total Revenue by Employee:-

DB Browser for SQLite - C:\Users\Deepak Vishwakarma\Downloads\garston_superstore (3).db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open Project Save Project

Database Structure Browse Data Edit Pragmas Execute SQL

SQL 1*

```
21 '''3.Total Revenue by Employee'''
22 SELECT Employees.Employee_ID, Employees.Name, SUM(Sales.Total_Price) AS Revenue_Generated
23 FROM Sales
24 JOIN Employees ON Sales.Employee_ID = Employees.Employee_ID
25 GROUP BY Employees.Employee_ID
26 ORDER BY Revenue_Generated DESC;
27
28
```

	Employee_ID	Name	Revenue_Generated
1	4	Lindsey Nguyen MD	7807.84
2	27	Jodi Mendoza	7627.3
3	17	Kimberly White	7186.92
4	42	Amber Miller	7097.63
5	24	Raven Watkins	6942.49
6	6	Joseph Cruz	6849.81
7	50	Andrew Reese	6371.31
8	31	James Perez	6206.55
9	5	Philip Fernandez	6014.02
10	15	Tom Martinez	5992.73

Execution finished without errors.
Result: 50 rows returned in 8ms
At line 22:
SELECT Employees.Employee_ID, Employees.Name, SUM(Sales.Total_Price) AS Revenue_Generated
FROM Sales
JOIN Employees ON Sales.Employee_ID = Employees.Employee_ID
GROUP BY Employees.Employee_ID
ORDER BY Revenue_Generated DESC;

Insights:-

- Measures employee performance by tracking how much revenue each employee contributes to the store.
- Can be used for performance evaluation and reward allocation.

4. Daily Sales Summary:-

DB Browser for SQLite - C:\Users\Deepak Vishwakarma\Downloads\garston_superstore (3).db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open Project Save Project

Database Structure Browse Data Edit Pragmas Execute SQL

SQL 1*

```
28 '''4.Daily Sales Summary '''
29
30 SELECT Sale_Date, SUM(Total_Price) AS Daily_Revenue, SUM(Quantity) AS Total_Items_Sold
31 FROM Sales
32 GROUP BY Sale_Date
33 ORDER BY Sale_Date ASC;
34
35
```

	Sale_Date	Daily_Revenue	Total_Items_Sold
1	2024-10-22	8657.93	198
2	2024-10-23	13005.86	259
3	2024-10-24	8699.25	197
4	2024-10-25	10430.94	248
5	2024-10-26	8731.6	217
6	2024-10-27	8976.01	151
7	2024-10-28	7762.43	168
8	2024-10-29	6449.42	145
9	2024-10-30	7225.17	157
10	2024-10-31	7963.18	161

Execution finished without errors.
Result: 30 rows returned in 14ms
At line 30:
SELECT Sale_Date, SUM(Total_Price) AS Daily_Revenue, SUM(Quantity) AS Total_Items_Sold
FROM Sales
GROUP BY Sale_Date
ORDER BY Sale_Date ASC;

Insights:-

- Allows for trend analysis, helping to identify peak sales days, seasonal patterns, or sales anomalies.
- Useful for financial reporting and operational planning.

5. Most Popular Product Categories:-

DB Browser for SQLite - C:\Users\Deepak Vishwakarma\Downloads\garston_superstore (3).db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo Open Project

Database Structure Browse Data Edit Pragmas Execute SQL

SQL 1*

```
35  '''Most Popular Product Categories
36  '''
37  SELECT Products.Category, SUM(Sales.Quantity) AS Total_Quantity
38  FROM Sales
39  JOIN Products ON Sales.Product_ID = Products.Product_ID
40  GROUP BY Products.Category
41  ORDER BY Total_Quantity DESC;
42
```

	Category	Total_Quantity
1	Snacks	1392
2	Bakery	1141
3	Beverages	1054
4	Groceries	998
5	Dairy	984

Execution finished without errors.
Result: 5 rows returned in 11ms
At line 37:
SELECT Products.Category, SUM(Sales.Quantity) AS Total_Quantity
FROM Sales
JOIN Products ON Sales.Product_ID = Products.Product_ID
GROUP BY Products.Category
ORDER BY Total_Quantity DESC;

Insights:-

- Provides insights into which product categories are selling the most, aiding inventory management and targeted marketing.
- Helps to identify trends in consumer preferences.

-:Conclusion:-

The Co-op Superstore database is a well-structured and carefully crafted system that captures essential aspects of store operations, such as product management, employee records, customer details, and sales transactions. It follows best practices in relational database design, with a focus on normalization, the use of meaningful constraints, and strong relationships through foreign keys. The data is generated synthetically, ensuring it remains realistic while also upholding high ethical standards and ensuring privacy. The database is designed to offer detailed analytical capabilities, providing valuable insights into product performance, top customers, and sales trends. In summary, this database is a powerful tool for streamlining and improving supermarket management and operations.

{Guided By:- Prof. John Evans}

{Submitted By:- Deepak Vishwakarma}

Thank You!

