The Store Database

Anca Pitigoi

September 16, 2024

The Store Database: Introduction

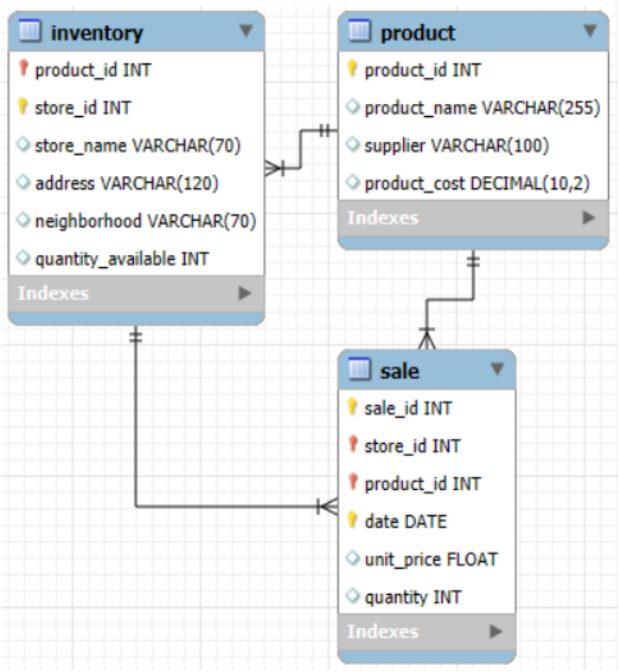
The Store Database project presents SQL querying skills using three data tables. Data cleaning, data validation, and exploratory data analysis will be performed using diverse functions such as joining tables, grouping by certain criteria, and performing calculations.

The datasets were provided by the Google Data Analytics course taken in 2021 for exercise purposes. However, they were not used as training or example resources during the academic courses.

ER Diagram

Figure 1

Store Database ER Diagram



In order to create the SQL schema, the entities, attributes, and relationships need to be identified. The entities are the three tables: inventory, product, and sale. Each table contains attributes that identify the stores, products, or sale transactions. Based on the data, the following relationships can be identified:

- **Product to Sale: One-to-Many**
  The `product_id` in the `product` table can appear multiple times in the `sale` table, as one product can be sold in multiple transactions.
- **Product to Inventory: One-to-Many**
  One product can be available in multiple stores, therefore can appear multiple times in the `inventory` table.
- **Inventory to Sale: One-to-Many**
  A store's id in the `inventory` table can be included multiple times in the `sale` table since a store will have many transactions.

Creating the database

For illustrative purposes, the database was created from scratch based on the ER diagram (See Figure 1) and the values were imported. Indexing was created because the database does not receive new writes and is static. This will allow faster searches, especially when joining the three tables together and querying the sales table.

The product table contains only four attributes: product id, which is also the primary key, product name, supplier, and the product cost.

The inventory table identifies the product, where it is stored, and how many quantities are available for selling. Therefore, the attributes are product id, store id, the store's name, address, neighborhood, and quantity available. The primary key is a composite key (`product_id, store_id`) because some product ids will be present in multiple stores, but only one time should the product be registered in a specific store.

The sale table contains information regarding the individual transactions. Each sale id can appear multiple times because a customer might buy multiple products on the same transaction. Therefore, the primary key is composed of `sale_id`, `product_id`, `store_id`, and `date`. After exploratory analysis was performed, it was observed that some stores can reset the sale id numbers over a period of time and some records were lost because of this process, therefore the date is an important identifying attribute. In the sales table, the date column was initially formatted to "YYYY-MM-DD" using Excel because MySQL was unable to interpret the original date format.

Listing 1

Importing the datasets

```
LOAD DATA LOCAL INFILE 'D:/Documents/Data
Analytics/Portfolio/SQL - Store
Inventory/product.csv'
INTO TABLE product
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(product_id, product_name, supplier,
product_cost);
```

The datasets were imported using the code in Listing 1, but firstly MySQL was set to allow local import of files. This option was chosen because the security implications are minimal since this database is used for educational purposes only, but in practice caution should be ensured. The code specifies the path the data will be loaded from into the indicated table. CSV files are separated by commas, and the fields are enclosed by the double quotes. Moreover, the lines are terminated by \n, meaning a new row in the file corresponds to a new record in the database table. Finally, the first row that contains the column names is ignored, and the column names where the data should be inserted are listed in the order that corresponds to the fields in the CSV file.

Exploratory Data Analysis

In this part of the project the datasets are checked for errors and duplicates, while also exploring details about the stores, such as the number of stores and the variety of items they carry.

The following actions were taken to validate and check for errors to ensure the integrity of the data:

- **Checked for missing** or **null values**, and everything is in order.
- **Checked for referential integrity** and discovered that 32 sale transactions might be erroneous because the combination of `store_id` and `product_id` does not exist in the `inventory` table. This could be because the `inventory` table contains only recent records or updates, meaning that some older sales entries may not correspond to the current inventory. Furthermore, the

company can also move products around in case the demand is too high in some places without recording these transactions.

- **Checked for outliers or inconsistent data**, such as products with negative values, or the quantity of the items being negative or zero in the sale table. There are no unusual values.
- **Checked for index efficiency**, and MySQL is using the store_id index to speed up the query. Moreover, only one row is expected to be scanned, which is considered to be highly efficient.

Main key take-aways for the exploratory data analysis:

- There are 1000 product ids recorded in the database.
- Of 1000 products, 174 have duplicate names mainly because the supplier is different. It is noticeable that there is a large variation in pricing between the suppliers. This information was retrieved using the code in Listing 2.
- There are 34 stores, and all stores have had transactions.
- The product IDs are unique in the inventory, even for products with the same name, because they are assigned to specific stores, potentially due to logistical reasons like warehouse locations or supplier differences.

Listing 2

Selecting duplicate product names & results

```
SELECT p.product_id, p.product_name, p.supplier, p.product_cost
FROM product p
INNER JOIN (
    SELECT product_name
    FROM product
    GROUP BY product_name
    HAVING COUNT(*) > 1) dup ON p.product_name = dup.product_name;
```

product_id	product_name	supplier	product_cost
108	Pepper - Green Thai	T.J. Maxx	1.78
14	Pepper - Green Thai	Ollie's Bargain Outlet	1.52
813	Oranges	Gordmans	1.55
15	Oranges	Burlington Coat Factory	5.76
734	Longos - Chicken Caeser Salad	Ben Franklin	0.06
311	Longos - Chicken Caeser Salad	National Stores	2.19
20	Longos - Chicken Caeser Salad	HomeGoods	3.87

Listing 3

Top performing stores

```
SELECT s.store_id,
       i.store_name,
       ROUND(SUM(s.unit_price *
       s.quantity), 0) AS revenue
FROM sale AS s
JOIN inventory AS i ON s.store_id =
i.store_id
GROUP BY s.store_id, i.store_name
ORDER BY revenue DESC;
```

store_id	store_name	revenue
84879	Ben Franklin	45061152
71053	Family Dollar	37128666
22726	Shopko	36841890
10002	T.J. Maxx	36770894
21777	Walmart	35530993

Actionable questions

Which store has the most sales?

The highest revenue is achieved by Ben Franklin store, topping a little over 45 million dollars (See Code in Listing 3).

What are the top-selling products by quantity, revenue, and store?

Rice paper, banana nut muffins, and tenderloin beef are the most sold items by quantity across all the stores (See Table 1).

Table 1

Top sold products by quantity

product_name	total_quantity
Rice Paper	41298
Muffin - Banana Nut Individual	40519
Beef - Tenderloin Tails	38661
Pepper - Green, Chili	34446
Cup Translucent 9 Oz	32020

The sales data reveals that Sultanas generated the highest revenue, totaling $223,612, followed closely by Chili green peppers with $223,496. Rye flour also contributed significantly, bringing in $197,093 in revenue.

Listing 4

Top performing products grouped by store

```
WITH ranked_products AS (
 SELECT
       s.product_id,
       p.product_name,
       s.store_id,
       i.store_name,
       ROUND(SUM(s.unit_price * s.quantity),0) AS total_revenue,
       ROW_NUMBER() OVER (PARTITION BY s.store_id
                     ORDER BY SUM(s.unit_price * s.quantity) DESC) AS top_performer
FROM
       sale AS s
JOIN
       product AS p ON s.product_id = p.product_id
JOIN
       inventory AS i ON s.store_id = i.store_id
GROUP BY
       s.product_id, p.product_name, s.store_id, i.store_name)
SELECT
       product_id,
       product_name,
       store_name,
       total_revenue
FROM
       ranked_products
WHERE
       top_performer = 1
ORDER BY
      total_revenue DESC;
```

Listing 4 entails a more involved query that firstly ranks the products within each store based on their total quantity sold and assigns the top performer a rank of 1 (`WITH ranked_products AS`). The `PARTITION BY s.store_id` ensures that the ranking is done separately for each store. With this selection, the next query filters the results to show only the top-selling product for each store (`WHERE top_performer = 1`). Finally, the top-performing products are ordered by revenue.

Table 2

Top sold products by store, ordered by total_revenue

product_name	store_name	total_revenue
Flour - Rye	Ben Franklin	921480
Broom - Corn	Walmart	426535
Sultanas	National Stores	356526
Pop Shoppe Cream Soda	Family Dollar	354410
Pop Shoppe Cream Soda	Five Below	353232
Sultanas	HomeGoods	343220
Flour - Rye	Gabe's	338853
Sultanas	Ollie's Bargain ...	331242
Sultanas	Meijer	328628

Some clear favorites among consumers can be observed from these results. From Table 2 it can be observed that at Ben Franklin, rye flour generated the highest revenue, totaling $921,480. Sultanas is the most popular product, appearing as the top performer in multiple stores, including National Stores, HomeGoods, Ollie's Bargain Outlet, Meijer, and Dollar Tree. Additionally, Pop Shoppe cream soda is another popular product, dominating stores such as Family Dollar, Five Below, Fred's, and Stein Mart, with revenue contributions exceeding $300,000 in many cases. These products are clearly in demand across retailers.

Which products generate the most profit? What about the least profit?

Listing 5

Products that generate highest marginal profit

```
SELECT p.product_id,
       p.product_name,
       ROUND(AVG(s.unit_price-
       p.product_cost),2) AS profit
FROM sale AS s
JOIN product AS p ON s.product_id =
p.product_id
GROUP BY p.product_id
ORDER BY profit DESC
LIMIT 7;
```

product_id	product_name	profit
885	Broom - Corn	9.07
248	Sultanas	9.04
117	Flour - Rye	8.54
89	Pop Shoppe Cream Soda	7.21
367	Salmon - Fillets	6.66
99	Cookies - Englishbay Wht	6.63
845	Papadam	6.4

To get the marginal profit for each product, considering variations in supplier costs and not mixing different store prices, the query contains `AVG(s.unit_price-p.product_cost) AS profit` and `GROUP BY product_id` syntaxes (See Listing 5). This way the profit for each unique product is calculated even when it is sold at a discount, or the sale price increases due to demand.

Corn broom and Sultanas are the top-performing products in terms of profit, with margins of $9.07, and $9.04 respectively. Rye flour, Pop Shoppe cream soda, and salmon fillets also show strong profitability. The range of products from groceries to household items all showing great profits suggests that maintaining a diverse inventory could be a beneficial strategy.

Table 3

Products that generate least marginal profits

product_id	product_name	profit
245	Coffee - Hazelnut Cream	0.01
471	Pastry - Choclate Baked	0.02
895	Wine - Red, Antinori Santa	0.02
195	Tart Shells - Sweet, 4	0.02
988	Wine - Manischewitz Con...	0.02
590	Wine - Wyndham Estate ...	0.02
671	Pasta - Linguini, Dry	0.02

Meanwhile, in order to find the least profitable products, the "`DESC`" syntax can be removed from Listing 5. The products with the lowest profit margins are: Hazelnut cream coffee ($0.01), chocolate pastry ($0.02), and the red wine Antinori Santa ($0.02). This could indicate high costs, competitive pricing, or possibly a strategic choice to attract customers with lower prices on these specific products. The company should closely monitor these low-profit products to determine if they are utilizing inventory space effectively and align with the company's overall strategic interests.

Which are the top 3 most profitable stores in 2019? What about the least profitable?

In addition to summing the multiplication of the difference between the acquisition and selling price, the `WHERE` clause is added to focus only on 2019 sale transactions. The most profitable stores were Ben Franklin ($48.8 million), Family Dollar ($39.9 million), and T.J. Maxx ($38 million). In order to find the bottom least profitable stores the "`DESC`" syntax is removed in the `ORDER BY` clause. The least profitable stores were Bi-Mart ($1.9 million), Bargain Hunt ($2 million), and Big Lots ($ 2.3 million).

How does sales performance vary over time (monthly & annually)?

Figure 1

Monthly total revenue by year

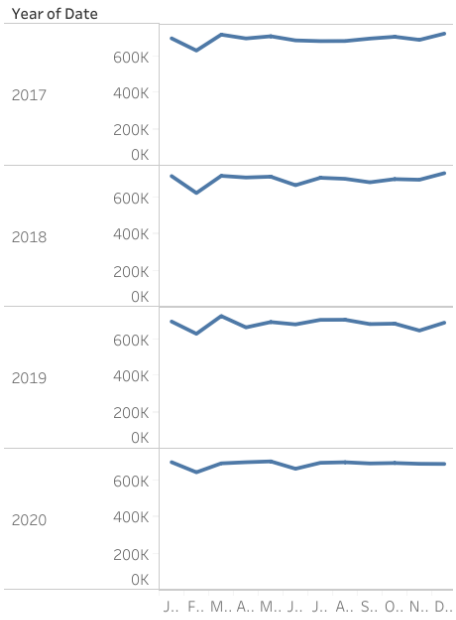
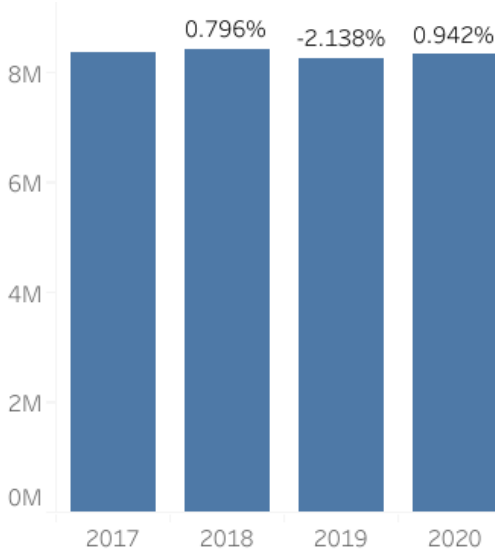


Figure 2

Annual total revenue



The sales data across the years indicates a recurring trend where sales dip notably towards the end of each year, particularly in December (See Figure 1). Additionally, there appears to be a slight decrease in sales around mid-year, around June and July, across multiple years. These patterns suggest that the summer months and the very end of the year typically experience lower sales volumes compared to other months.

When the sales are compared over the years, the total revenue stays constant (See Figure 2). An increase was observed from 2017 to 2018, but next year the revenue went below 2017 levels. A slight increase is observed in 2020, but not as high as 2018. However, these fluctuations can be considered small.

Listing 6

Largest transactions in terms of number of items sold

```
SELECT
      YEAR(s.date) AS transaction_year,
      s.sale_id,
      SUM(s.quantity) AS total_quantity
FROM sale AS s
GROUP BY
      YEAR(s.date),s.sale_id
ORDER BY
      total_quantity DESC
LIMIT 10;
```

transaction_year	sale_id	total_quantity
2019	27811	411
2017	84402	395
2020	84920	392
2017	82315	384
2018	66004	382
2020	37270	374
2017	72981	374

Which is the largest transaction in terms of quantity?

Because the sale id restarts each year and is not unique across the entire dataset, the date should be included in the grouping to uniquely identify transactions. It is not necessary to include the whole date, only the year in order to accurately aggregate and identify the sales (See Listing 6).

The data highlights the annual peak sales transactions, with 2019 recording the highest at 411 units. Notably, 2017 was an active year, featuring multiple top transactions ranging from 374 to 395 units.

Conclusion

In this project, the store database was created and analyzed in order to reveal insights and showcase the SQL skills. Initially, the database was structured according to a diagram detailing the relationships between tables. Following this, data was imported, and the datasets were methodically examined using SQL functions including aggregation, grouping, joining, ordering, and indexing.

Based on the analysis, Ben Franklin store achieved the highest revenue across all years, rice paper is the most sold product in terms of quantity, and Sultanas brings the highest revenue across all stores. In terms of profitability, corn brooms and Sultanas stand out with the highest profit margins, whereas hazelnut cream coffee yields the lowest margins. The sales trends over the years show a noticeable decline in December each year. Also, the best year in terms of revenue was 2018, toping $8.4 million dollars. The highest transaction ever recorded was in 2019, where a customer bought 411 items in one sale.

Further investigation could explore the reasons behind the low profitability of certain products like hazelnut cream coffee could help in deciding whether to continue, modify, or discontinue these products. Another area of interest could be the examination of sales patterns immediately before and after the highest revenue year to understand what drove the peak and how to replicate or surpass it in future years. Moreover, adding the shipping price from a distribution warehouse could reveal whether buying products from a certain supplier would be ideal, since it is already known that the purchasing prices are competitive.

The use of SQL in analytics profoundly enhances store analysis by enabling efficient data manipulation and retrieval, facilitating deep insights into sales trends, customer behavior, and inventory management. This powerful language supports complex queries, aggregation, and joins, which are essential for extracting actionable intelligence from vast amounts of retail data, thereby driving informed business decisions and strategic planning.