”Optimizing Sales Data Management and Analysis System”

Shraddha Sunil Falle[1], Nikhil Ambati[2], Jayakrishna Pavuluri[3] [1]UB ID: sfalle [2]UB ID: nambati [3]UB ID: jpavulur

***Abstract*—Addressing issues like data redundancy, sluggish query performance, and constrained analytical capabilities, this project seeks to address the company's challenges with managing and analyzing sales data. To meet the organization's changing data management needs, a transition is necessary from Excel to a database. The project's significance lies in its potential to enhance sales performance, operational effectiveness, cost savings, and the ability to gain insightful information from customers and also to support the company's use of data-driven decision-making.**

I. PROBLEM STATEMENT

Effective administration and analysis of sales data, particularly inside the organization's order processing system, is currently a task. There is a lack of normalization in the current data structure which causes problems including data redundancy, sluggish query performance, constrained analytical capabilities, and worries about data integrity. Also, insufficiently representing complex data interactions between consumers, goods, orders, and workers. According to the organization, this creates considerable challenges for data-driven decision-making, hinders productivity, and restricts the potential to gain insightful information from sales data. The project seeks to develop an Entity Relationship Diagram and normalized database schema to address these problems, providing a more complete and effective solution that will give the business simplified operations, accurate reporting, and improved strategic planning capabilities.

Why do you need a database instead of an Excel file?

Companies must switch from Excel to a database for numerous reasons. Excel is a key tool for managing and analyzing basic data, but as the amount and complexity of sales data increases, so do its limits. The advantages of a database, on the other hand, include enforcement of data integrity through constraints, improving query performance, supporting advanced analytics through structured data models, accurately simulating complex data relationships, scaling to accommodate growing data volumes and user concurrency, enhancing data security through user authentication and encryption, and offering effective version control through structured transaction logs. By providing increased data quality, additional analytics capabilities, scalability, security, and efficient version control, this shift to a relational database is in line with the changing requirements of our organization's data management.

A. Discuss the background of the problem leading to your objectives. Why is it a significant problem?

The changing nature of modern corporate operations serves as the foundation for the problem that led to the project's aims. As organizations grow and data complexity rises, a number of major difficulties have emerged. First off, it has become more and more challenging to effectively manage and analyze data within the limitations of conventional technologies due to the growing complexity of sales data, which is driven by different product ranges and expanding client bases. The existing data management method introduces errors, redundancies, and inconsistencies that could ultimately result in poor decision-making. As a result, preserving data consistency and accuracy has become a vital issue. Additionally, as data volume has increased, query and data retrieval performance has substantially slowed down, affecting productivity and decision-making. Advanced analytics are also urgently needed to dive deeper insights from sales data, but the current data structure's constraints make it difficult to meet these analytical requirements. Additionally, the delicate data relationships between consumers, goods, orders, and staff are crucial to corporate operations; however, the limitations of the current technologies in effectively depicting these complex interactions provide a substantial barrier to data-driven decision-making. The limitations of current tools to scale have also become increasingly obvious as organizations continue to expand, particularly when it comes to supporting multiple users at once and managing expanding data quantities.

B. Explain the potential of your project to contribute to your problem domain. Discuss why this contribution is crucial.

There are many benefits to this project's successful implementation. Firstly, it enables considerable improvements in sales performance through data-driven decision-making, pricing strategies, and precise customer targeting, all of which increase revenue. Secondly, the project results in remarkable efficiency improvements, which translate into significant cost savings through streamlined operations and quick access to important data, uplifting profitability. The project also gives us the tools to gain insights into customer preferences and behavior, which results in increased customer satisfaction and brand loyalty. The project's ability to scale up and down is vital for market growth because it opens up new customer segments and increases sales potential. Moreover, the project includes improved data security and compliance measures, significantly reducing the risks of data breaches and enhancing the defense of our reputation. The project also grants the company a competitive edge, fortifying its position in the fast-paced and fiercely competitive business environment. Sales Enhancement: The project supports data-driven decisions, optimizes pricing strategies, and targets customers, resulting

in increased sales and revenue. Cost reductions: Efficiency improvements resulting from streamlined processes and quick data access lead to impressive cost reductions, boosting overall profitability. Customer satisfaction is enhanced, and brand loyalty is fostered by personalized services that are made possible by a deep understanding of customer preferences and behavior. Market Expansion: The project's scalability is the fundamental of our market expansion strategy because it provides the opportunity to reach out to new clients and increase our sales scope. Reduced risk: Better compliance and data security procedures considerably reduce the possibility of data breaches, enhancing our standing and financial security. Competitive Advantage: The efficiency enhancements made possible by the project give our company a competitive edge, securing its place in a fast-paced, competitive market. Better Decision-Making: Because of improved data quality and accurate data representation, the company can make more informed and data-driven decisions. This leads to improved strategic planning and a competitive advantage in the market.

II. TARGET USERS

The Sales and Marketing Teams as well as Customer Support are two of the main user groups who will communicate with the database. Database administrators (DBAs) are in charge of maintaining and improving the databases, which are used by these teams for important operations.

Sales and Marketing Teams: The database's primary users are the Sales and Marketing Teams. They will rely on it to track sales performance and make strategic decisions. For example, sales representatives use the database to access customer order history, identify potential sales opportunities, and track sales goals. The marketing team uses customer data to evaluate the effectiveness of marketing campaigns, customize promotions, and improve their strategies.

Customer Support Representatives: Another essential user group is customer support representatives. They access the database to quickly retrieve customer information, view order history, and effectively respond to inquiries. This means that a customer calling the support line to ask about a recent order can receive prompt assistance because the support agent can access the database and retrieve the customer's order information.

Real-world Scenario: The database plays a crucial role in day-to-day operations in the context of a successful e-commerce business. Salespeople rely on the database to find sales opportunities, as it provides information about customer preferences, order histories, and prospects for upselling and cross-selling. Marketing teams use the database to evaluate the success of their campaigns and optimize product promotions. The customer support team relies on the database to provide accurate and timely assistance to customers. Database administrators are responsible for maintaining the system's health, enforcing security measures, ensuring data accuracy, and improving query performance. Their work on data backups, access control, and software updates keeps the database accessible and secure.

III. ENTITY-RELATIONSHIP DIAGRAM

Understanding the structural foundations of the Sales and Customer Management System requires understanding the Entity-Relationship Diagram (ERD) presented here. It offers a clear visual representation of the data model of the system and provides insightful information about the complex interactions between entities and their attributes.

A. Entities and Attributes

Order Details: The "Order Details" entity comprises attributes, each with significance, representing a crucial part of the system. Each order detail is uniquely identified by the "Order Detail ID," serving as the primary key. The "Order ID," as a foreign key, establishes a connection with the "Order" entity and identifies the order to which each detail relates. Another foreign key, "Product Code," creates a connection with the "Product" entity and identifies the particular product. Additional attributes include "Order Line Number," "Quantity Ordered," "Price Each," "Sales," and "Employee ID," with the latter acting as a foreign key to identify the employee in charge of completing the order.

Product: The "Product" entity has a primary key in the form of "Product Code," which acts as a vital identifier for specific products. It includes the attribute "Product Line" to divide products into distinct product lines. Additionally, "MSRP" (Manufacturer's Suggested Retail Price) clarifies the product's suggested retail pricing.

Customer: The "Customer" entity holds all necessary client-related data and uses the "Customer ID" as its primary key, ensuring that each customer can be uniquely identified. A complete customer profile is provided by the "Customer Name," "Phone," "Address Line1," "Address Line2," "City," "State," "Postal Code," and "Country" attributes.

B. Entities and Attributes

Line2, City, State, and Country: These attributes serve as foreign keys. The "Postal Code" attribute provides information on the postal code associated with the customer's precise location.

Territories: The "Territories" entity uses "Country" as its primary key and associates territories with particular countries.

Order Status: The "Order Status" entity has "Order ID" as the primary key. The connection between "Customer ID," represented as a foreign key, and the "Customer" entity indicates that the customer initiates the order. "Order Status" states each order's status, while the "Order Date" attribute records the specific time of placing an order.

Department: "Department ID" serves as the primary key for the "Department" entity, which is vital for organizational structuring. The "Department Name" attribute clarifies each department's nomenclature.

Employee: The main attributes of the "Employee" entity are "Employee ID" as the primary key, "Employee Name," "Salary," and "Department ID," which, as a foreign key, creates a connection with the "Department" entity to capture each employee's departmental connection.

C. Attributes and Types in Each Table

Order Details: - Order Detail ID (INT) - Order ID (INT) - Product Code (VARCHAR(255)) - Order Line Number (INT) - Ordered Quantity (INT) - Sales (DECIMAL(10,2)) - Employee ID (INT) - Price Each (DECIMAL(10,2))

Product: - MSRP (INT) - Product Line (text) - Product Code (VARCHAR(255))

Customer: - Customer ID (INT) - Customer Name (VARCHAR(255)) - Phone (VARCHAR(255)) - Address Line1 (VARCHAR(255)) - Address Line2 (VARCHAR(255)) - City (VARCHAR(255)) - State (VARCHAR(255)) - Country (VARCHAR(255)) - Postal Code (VARCHAR(255))

Territories: - Territory (VARCHAR(255)) - Country (VARCHAR(255))

Order Status: - Order ID (INT) - Customer ID (INT) - Order Date (DATE) - Order Status (VARCHAR(255))

Department: - Department Name (VARCHAR(255)) - Department ID (INT)

Employee: - Employee Name (VARCHAR(255)) - Salary (DECIMAL(10,2)) - Employee ID (INT) - Department ID (INT)

D. Primary Keys

Order Details: ”Order Detail ID” attribute is the primary key for the ”Order Details” table.

Product: ”Product Code” attribute is the primary key for the ”Product” table.

Customer: ”Customer ID” attribute is the primary key for the ”Customer” table.

Territories: ”Country” attribute is the primary key for the ”Territories” table.

Order Status: ”Order ID” serves as the primary key for the ”Order Status” table.

Department: ”Department ID” is the primary key for the ”Department” table.

Employee: ”Employee ID” is the primary key for the ”Employee” table.

E. Foreign Keys

Order Details: In the ”Order Details” table, the columns ”Order ID,” ”Product Code,” and ”Employee ID” are designated as foreign keys. ”Order ID” connects to ”Order Status,” ”Product Code” connects to ”Product,” and ”Employee ID” connects to ”Employee.”

Customer: A foreign key is defined between the ”Country” column in the ”Customer” table and the ”Country” primary key in the ”Territories” table.

Order Status: The ”Customer ID” column in the ”Order Status” table serves as a foreign key connecting to the ”Customer ID” primary key in the ”Customer” table.

Employee: A foreign key connects the ”Department ID” primary key in the ”Department” table to the ”Department ID” foreign key in the ”Employee” table.

F. Relationships

Product to Order Details (One-to-Many): There is a one-to-many configuration governing the relational dynamics between the ”Product” and ”Order Details” entities. Different order details for a single product can be intricately linked together. This connection to the ”Product Code” in the ”Order Details” entity (Foreign Key) is made possible by the ”Product Code” within the ”Product” entity, which serves as the Primary Key.

G. Relationships (Continued)

Territories to Customer (One-to-One): A one-to-one correspondence between ”Territories” and ”Customer” denotes that a specific nation corresponds to a single territory. The ”Country” attribute in the ”Territories” entity (Primary Key) and the ”Country” attribute in the ”Customer” entity (Foreign Key) come together to materialize this.

Order Status to Order Details (One-to-Many): A one-to-many relationship is created by the ”Order Status” entity’s connection to the ”Order Details” entity. The ”Order ID” in the ”Order Status” entity (Primary Key) directly links to the ”Order ID” in the ”Order Details” entity (Foreign Key), making this connection evident.

Department to Employee (One-to-One): Employees are assigned to one department per employee in a one-to-one relationship with their respective departments. The connection of employees with their organizational units is emphasized by the ”Department ID” in the ”Department” entity’s (Primary Key) smooth coordination with the ”Department ID” in the ”Employee” entity’s (Foreign Key).

Employee to Order Details (One-to-Many): One employee may be in charge of multiple order details because of the one-to-many relationship that exists between the ”Employee” entity and the ”Order Details” entity. To represent the employees’ responsibility during the order fulfillment process, the ”Employee ID” in the ”Employee” entity (Primary Key) closely connects with the ”Employee ID” in the ”Order Details” entity (Foreign Key).

H. BOYCE-CODD NORMAL FORM (BCNF) DATABASE DESIGN

Boyce-Codd Normal Form (BCNF) is the framework we used to structure the database design for the Sales and Customer Management System. BCNF is a set of rules for arranging data and assuring data integrity in a database.

Selection of Primary Keys

”Order Detail ID” serves as the primary key in the ”Order Details” table.

”Product Code” serves as the primary key for the ”Product” table.

”Customer ID” acts as the ”Customer” table’s primary key.

”Country” serves as the primary key in the ”Territories” table.

”Order ID” serves as the ”Order Status” table’s primary key.

”Department ID” serves as the primary key for the ”Department” table.

”Employee ID” has been designated as the primary key in the ”Employee” table.

I. Non-Prime Attributes Must Rely on Primary Key

Order Details Table: The primary key, ”Order Detail ID,” within the ”Order Details” table is directly related to attributes like ”Order ID,” ”Product Code,” and ”Employee ID.”

Product Table: The ”Product” table ensures that secondary keys like ”Product Line” and ”MSRP” are closely related to the primary key, ”Product Code.”

Customer Table: In the ”Customer” table, there is a close relationship between ”Country” and the primary key, ”Customer ID.”

Order Status Table: The ”Order Status” table shows a definite relationship between ”Customer ID” and the primary key, ”Order ID.”

Employee Table: Within the ”Employee” table, values such as ”Employee Name” and ”Salary” show the relationship with the primary key, ”Employee ID.”

In order to comply with BCNF specifications, we made sure that every attribute had an obvious connection to the primary key, which decreased complexity and redundancy in the data structure.

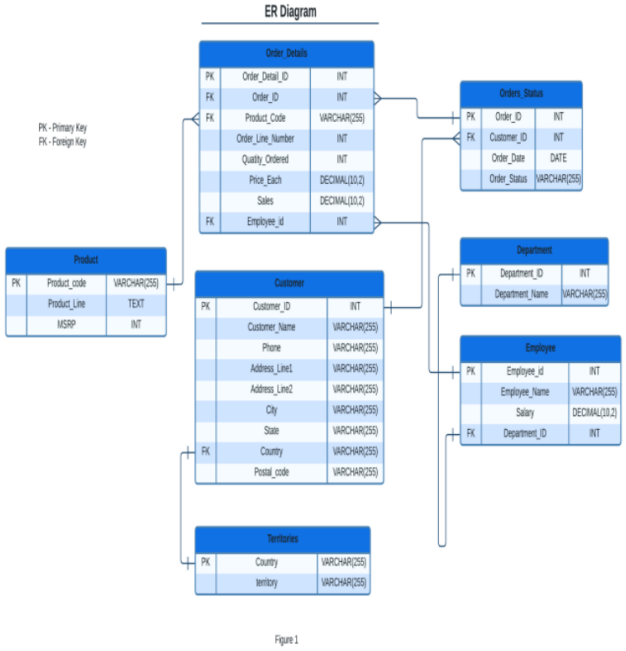


Fig. 1. ER Diagran

IV. Database Table Creation and Data Importation in PostgreSQL

We successfully built and organised seven different PostgreSQL tables in our database system to handle different facets of our sales data. In order to guarantee appropriate organisation and data integrity, the procedure involved defining the schema for every table. We created the tables using SQL scripts and loaded the data into the tables using PostgreSQL's effective bulk data loading features. Precisely prepared CSV files allowed for this importation, guaranteeing precision and coherence in the data flow. The efficient procedure created a solid basis for reliable data management and analysis in addition to efficiently populating our tables.

A. Customer Table

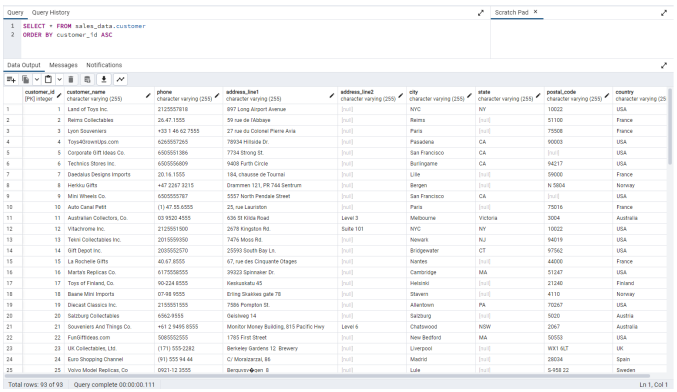


Fig. 2. Customer table from the sales data database

B. Department Table

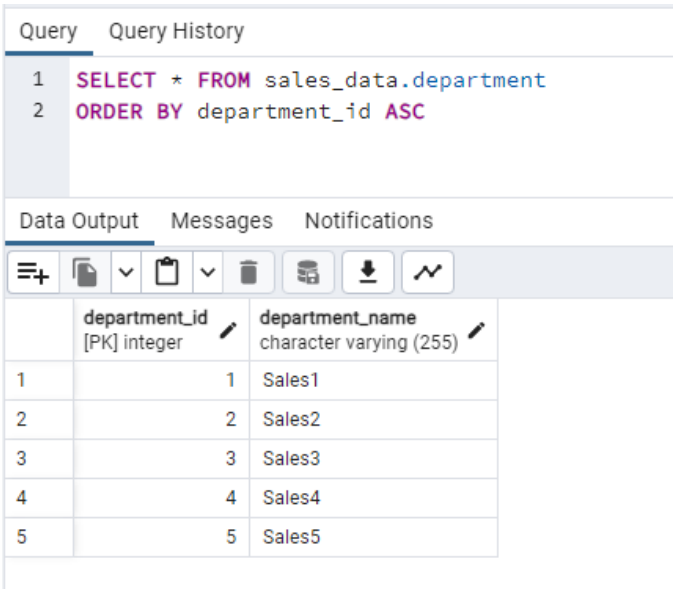


Fig. 3. Department table from the sales data database

C. Employee Table

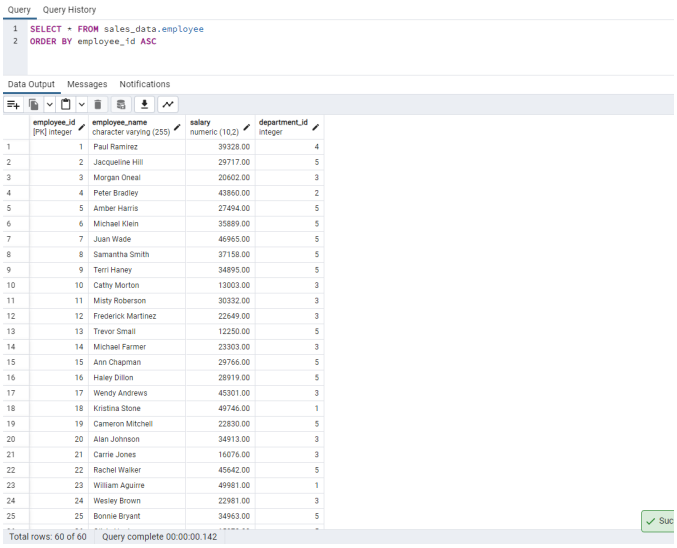


Fig. 4. Employee table from the sales data database

D. Order Details Table

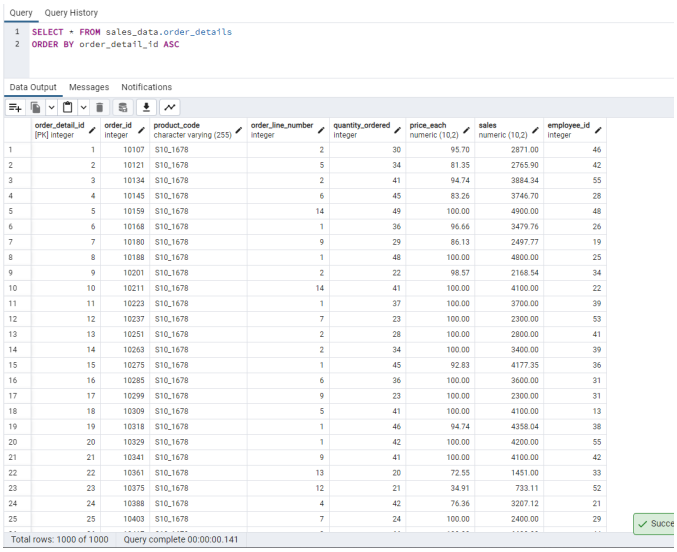


Fig. 5. Order Details table from the sales data database

E. Product Table
F. Terroteries Table
G. Order Status Table

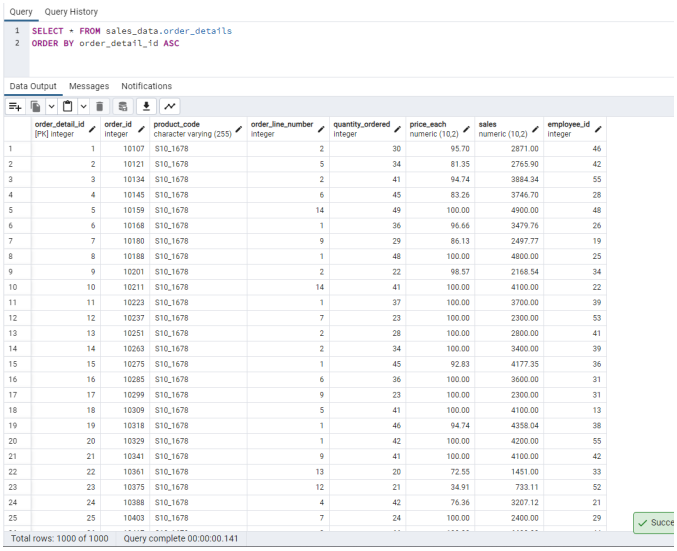


Fig. 6. Product table from the sales data database

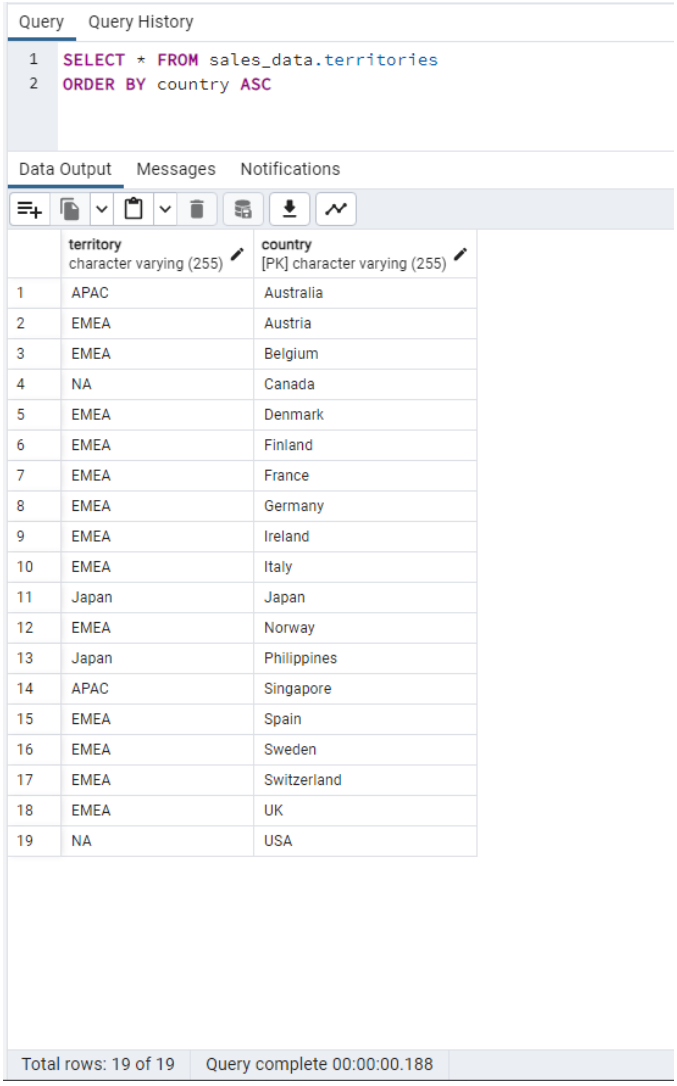


Fig. 7. Territories table from the sales data database

Query Query History

```
SELECT * FROM sales_data.orders_status
ORDER BY order_id ASC
```

Data Output Messages Notifications

	order_id [PK] integer	order_date date	order_status character varying (255)	customer_id integer
1	10100	2003-01-06	Shipped	41
2	10101	2003-01-09	Shipped	73
3	10102	2003-01-10	Shipped	12
4	10103	2003-01-29	Shipped	18
5	10104	2003-01-31	Shipped	24
6	10105	2003-02-11	Shipped	49
7	10106	2003-02-17	Shipped	87
8	10107	2003-02-24	Shipped	1
9	10108	2003-03-03	Shipped	67
10	10109	2003-03-10	Shipped	46
11	10110	2003-03-18	Shipped	78
12	10111	2003-03-25	Shipped	9
13	10112	2003-03-24	Shipped	25
14	10113	2003-03-26	Shipped	40
15	10114	2003-04-01	Shipped	63
16	10115	2003-04-04	Shipped	28
17	10116	2003-04-11	Shipped	91
18	10117	2003-04-16	Shipped	27
19	10118	2003-04-21	Shipped	54
20	10119	2003-04-28	Shipped	20
21	10120	2003-04-29	Shipped	11
22	10121	2003-05-07	Shipped	2
23	10122	2003-05-08	Shipped	68
24	10123	2003-05-20	Shipped	47
25	10124	2003-05-21	Shipped	85

Total rows: 308 of 308 Query complete 00:00:00.182

Fig. 8. Order Status table from the sales data database

V. Addressing Challenges and Driving Improvements in Large Dataset Management

As part of our continuous commitment to upholding a reliable and effective database system, we started an effort to assess and improve the performance of our database operations, especially when managing bigger datasets.

A. Performance Challenges with Large Datasets

As our database expanded, we noticed a small but noticeable lag in query response times, primarily due to an increase in the amount of sales data. These delays, only a few seconds, could negatively affect real-time data analysis capabilities and lead to less-than-ideal user experiences.

The issues were particularly evident in the following scenarios:

- Employee and Customer Name Lookups: The autocomplete feature and search functionality were not as responsive as required.
- Order Details Access: System lags when attempting to retrieve detailed order information affected our order management and reporting procedures.

B. Indexing Strategies Employed

We developed a thorough indexing strategy to address these concerns and improve the efficiency and speed of our database queries.

1) Index Implementation: **1. Employee and Customer Name Indexes:**

- **SQL Implementation:**
  ```
  CREATE INDEX idx_employee_name ON
  sales_data.employee (Employee_Name);
  CREATE INDEX idx_customer_name ON
  sales_data.customer (Customer_Name);
  ```
- **Outcome:** Significantly reduced execution time for name-based searches.

Analysis of Performance Improvement

Our database operations were significantly improved by the addition of these indexes, as the following details demonstrate:

Execution Time of Queries

Large dataset queries executed much more slowly before indexes were added. These queries now run in less than a second thanks to the updated indexing. Faster data retrieval and processing are now possible thanks to this remarkable decrease in query execution time, which has also greatly increased our database operations' throughput and efficiency.

System Scalability

There has been a significant improvement in the database system's capacity to manage growing data volumes. Longer query latency was frequently the outcome of increasing data volume prior to indexing. As of right now, the system shows strong scalability, effectively handling bigger datasets without sacrificing query response time. With this enhancement, our database will be able to function well even as data demands increase.

User Satisfaction

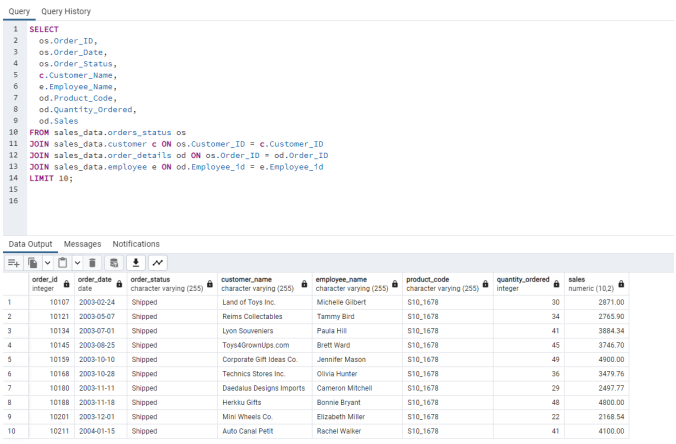
A noticeable improvement in the user experience is a direct benefit of faster query responses. Data analysts and business users, among other internal stakeholders, have reported easier and more productive interactions with the database system. This is particularly significant in situations where prompt data analysis and access are essential to business operations.

VI. Overview of Database SQL Query Testing

We give a summary of the comprehensive testing we did using different SQL queries on our database system in this section. The system's performance and capabilities were assessed during this crucial testing phase. The queries were created to cover insertion, deletion, and updating, among other operations, giving a thorough evaluation of the database's functionality. To further show how well the system handled challenging data retrieval scenarios, a range of select queries were run. Several SQL techniques, including joins, order by, group by, and subqueries, were used in these select queries. The results of these searches were carefully documented, and screen grabs were made to visually record the execution outcomes, demonstrating the efficiency and stability of our database system.

A. *Select Queries*

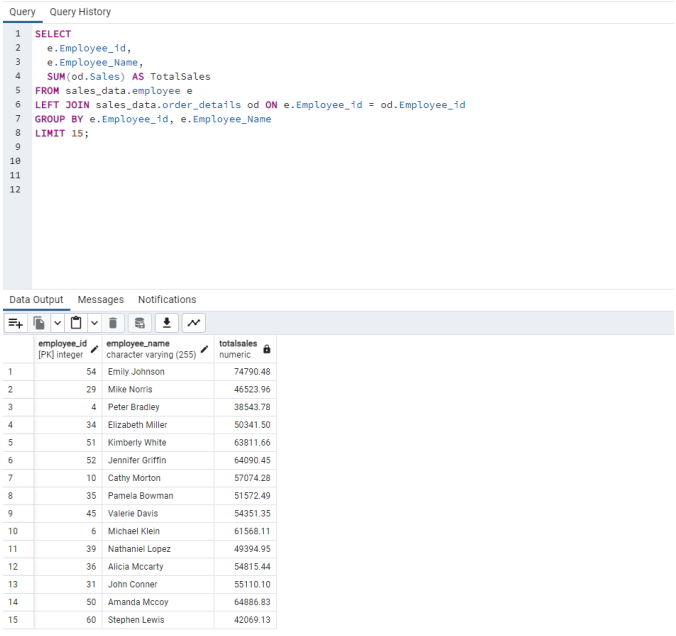
The select queries we used for testing are the main topic of this subsection. These queries are essential for showcasing how effectively the database can retrieve and handle data. They were thoughtfully created to demonstrate how the database handled a range of intricate querying strategies, such as joins, order by, group by, and subqueries. The outcomes of these select queries demonstrated the database system's ability to handle and analyse massive amounts of data, offering insightful information about its scalability and performance.



```
SELECT
  os.Order_ID,
  os.Order_Date,
  os.Order_Status,
  c.Customer_Name,
  e.Employee_Name,
  od.Product_Code,
  od.Quantity_Ordered,
  od.Sales
FROM sales_data.orders_status os
JOIN sales_data.customer c ON os.Customer_ID = c.Customer_ID
JOIN sales_data.order_details od ON os.Order_ID = od.Order_ID
JOIN sales_data.employee e ON od.Employee_id = e.Employee_id
LIMIT 10;
```

	order_id integer	order_date date	order_status character varying (255)	customer_name character varying (255)	employee_name character varying (255)	product_code character varying (255)	quantity_ordered integer	sales numeric (10,2)
1	10107	2003-02-24	Shipped	Land of Toys Inc.	Michelle Gilbert	S10_1678	30	2871.00
2	10121	2003-05-07	Shipped	Reims Collectables	Tammy Bird	S10_1678	34	2765.90
3	10134	2003-07-01	Shipped	Lyon Souveniers	Paula Hill	S10_1678	41	3884.34
4	10145	2003-08-25	Shipped	Toys4GrownUps.com	Brett Ward	S10_1678	45	3746.70
5	10159	2003-10-10	Shipped	Corporate Gift Ideas Co.	Jennifer Mason	S10_1678	49	4900.00
6	10168	2003-10-28	Shipped	Technics Stores Inc.	Olivia Hunter	S10_1678	36	3479.76
7	10180	2003-11-11	Shipped	Daedalus Designs Imports	Cameron Mitchell	S10_1678	29	2497.77
8	10188	2003-11-18	Shipped	Herkku Gifts	Bonnie Bryant	S10_1678	48	4800.00
9	10201	2003-12-01	Shipped	Mini Wheels Co.	Elizabeth Miller	S10_1678	22	2168.54
10	10211	2004-01-15	Shipped	Auto Canal Petit	Rachel Walker	S10_1678	41	4100.00

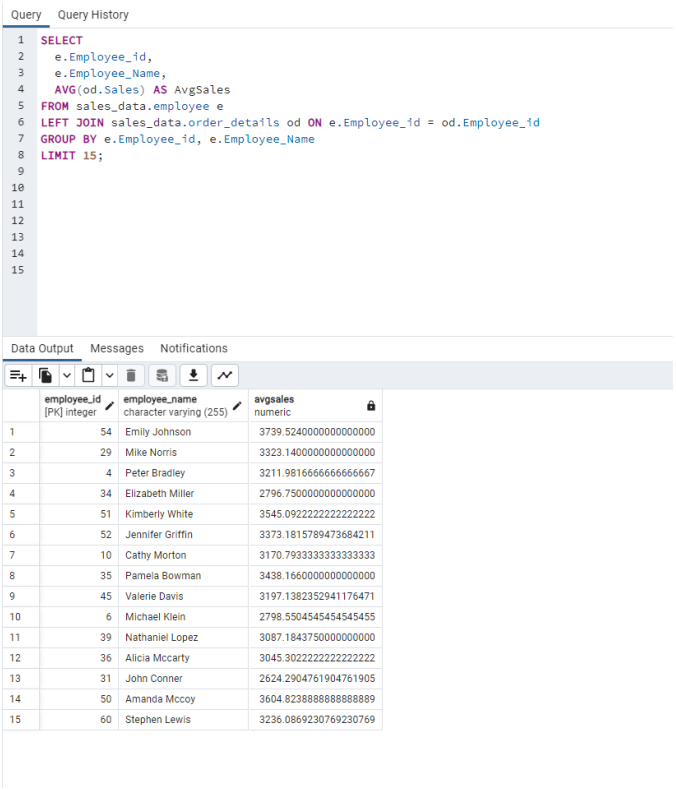
Fig. 9. Select Query 1



```
SELECT
  e.Employee_id,
  e.Employee_Name,
  SUM(od.Sales) AS TotalSales
FROM sales_data.employee e
LEFT JOIN sales_data.order_details od ON e.Employee_id = od.Employee_id
GROUP BY e.Employee_id, e.Employee_Name
LIMIT 15;
```

	employee_id [PK] integer	employee_name character varying (255)	totalsales numeric
1	54	Emily Johnson	74790.48
2	29	Mike Norris	46523.96
3	4	Peter Bradley	38543.78
4	34	Elizabeth Miller	50341.50
5	51	Kimberly White	63811.66
6	52	Jennifer Griffin	64090.45
7	10	Cathy Morton	57074.28
8	35	Pamela Bowman	51572.49
9	45	Valerie Davis	54351.35
10	6	Michael Klein	61568.11
11	39	Nathaniel Lopez	49394.95
12	36	Alicia Mccarty	54815.44
13	31	John Conner	55110.10
14	50	Amanda Mccoy	64886.83
15	60	Stephen Lewis	42069.13

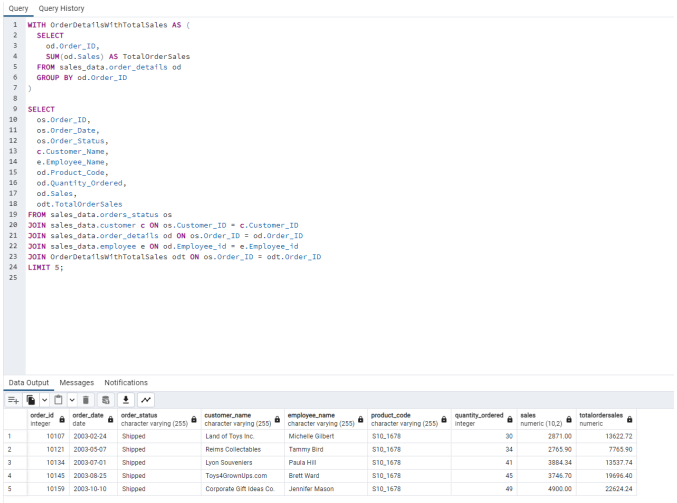
Fig. 10. Select Query 2



```
SELECT
  e.Employee_id,
  e.Employee_Name,
  AVG(od.Sales) AS AvgSales
FROM sales_data.employee e
LEFT JOIN sales_data.order_details od ON e.Employee_id = od.Employee_id
GROUP BY e.Employee_id, e.Employee_Name
LIMIT 15;
```

	employee_id [PK] integer	employee_name character varying (255)	avgsales numeric
1	54	Emily Johnson	3739.5240000000000000
2	29	Mike Norris	3323.1400000000000000
3	4	Peter Bradley	3211.9816666666666667
4	34	Elizabeth Miller	2796.7500000000000000
5	51	Kimberly White	3545.0922222222222222
6	52	Jennifer Griffin	3373.1815789473684211
7	10	Cathy Morton	3170.7933333333333333
8	35	Pamela Bowman	3438.1660000000000000
9	45	Valerie Davis	3197.1382352941176471
10	6	Michael Klein	2798.5504545454545455
11	39	Nathaniel Lopez	3087.1843750000000000
12	36	Alicia Mccarty	3045.3022222222222222
13	31	John Conner	2624.2904761904761905
14	50	Amanda Mccoy	3604.8238888888888889
15	60	Stephen Lewis	3236.0869230769230769

Fig. 11. Select Query 3



```
WITH OrderDetailsWithTotalSales AS (
  SELECT
    od.Order_ID,
    SUM(od.Sales) AS TotalOrderSales
  FROM sales_data.order_details od
  GROUP BY od.Order_ID
)

SELECT
  os.Order_ID,
  os.Order_Date,
  os.Order_Status,
  c.Customer_Name,
  e.Employee_Name,
  od.Product_Code,
  od.Quantity_Ordered,
  od.Sales,
  odt.TotalOrderSales
FROM sales_data.orders_status os
JOIN sales_data.customer c ON os.Customer_ID = c.Customer_ID
JOIN sales_data.order_details od ON os.Order_ID = od.Order_ID
JOIN sales_data.employee e ON od.Employee_id = e.Employee_id
JOIN OrderDetailsWithTotalSales odt ON os.Order_ID = odt.Order_ID
LIMIT 5;
```

	order_id integer	order_date date	order_status character varying (255)	customer_name character varying (255)	employee_name character varying (255)	product_code character varying (255)	quantity_ordered integer	sales numeric (10,2)	totalordersales numeric
1	10107	2003-02-24	Shipped	Land of Toys Inc.	Michelle Gilbert	S10_1678	30	2871.00	13622.72
2	10121	2003-05-07	Shipped	Reims Collectables	Tammy Bird	S10_1678	34	2765.90	7765.90
3	10134	2003-07-01	Shipped	Lyon Souveniers	Paula Hill	S10_1678	41	3884.34	13537.74
4	10145	2003-08-25	Shipped	Toys4GrownUps.com	Brett Ward	S10_1678	45	3746.70	19696.40
5	10159	2003-10-10	Shipped	Corporate Gift Ideas Co.	Jennifer Mason	S10_1678	49	4900.00	22624.24

Fig. 12. Select Query 4

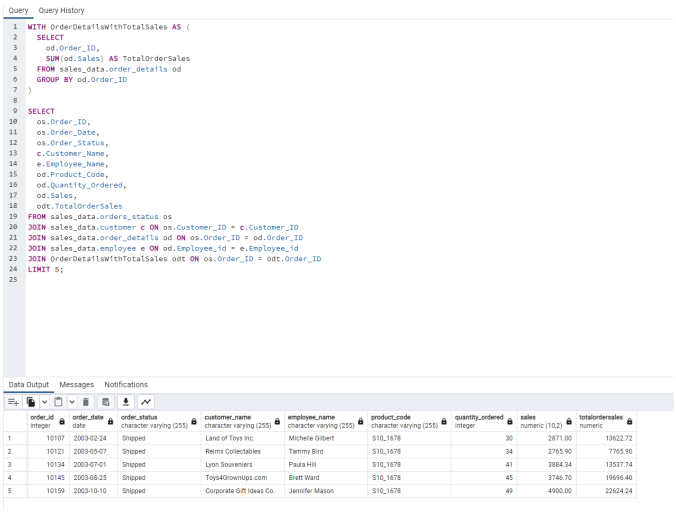


Fig. 13. Select Query 5

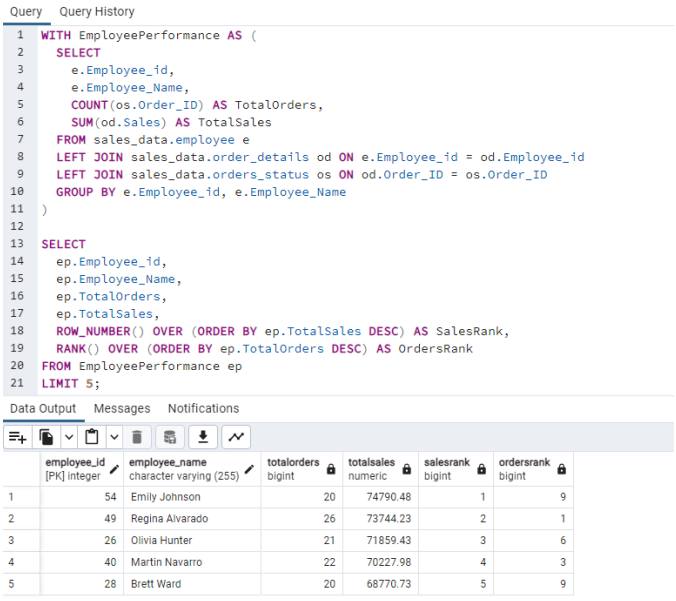


Fig. 14. Select Query 6

B. Insert Queries

We will examine the 'Insert Queries' used in our database testing process in this subsection. When assessing the database's ability to smoothly add new data entries, insert queries are essential. These queries were created expressly to evaluate how well the database performed data insertion tasks, from straightforward inserts of a single record to more intricate operations involving several records. When assessing our database system, the effectiveness of these insert operations was crucial, especially in situations where frequent and dynamic data additions were needed. The process of data insertion involves more than just adding new entries; it also involves making sure that these additions don't interfere with concurrent operations or transactional integrity. Thus, our assessment also includes determining how these insertions

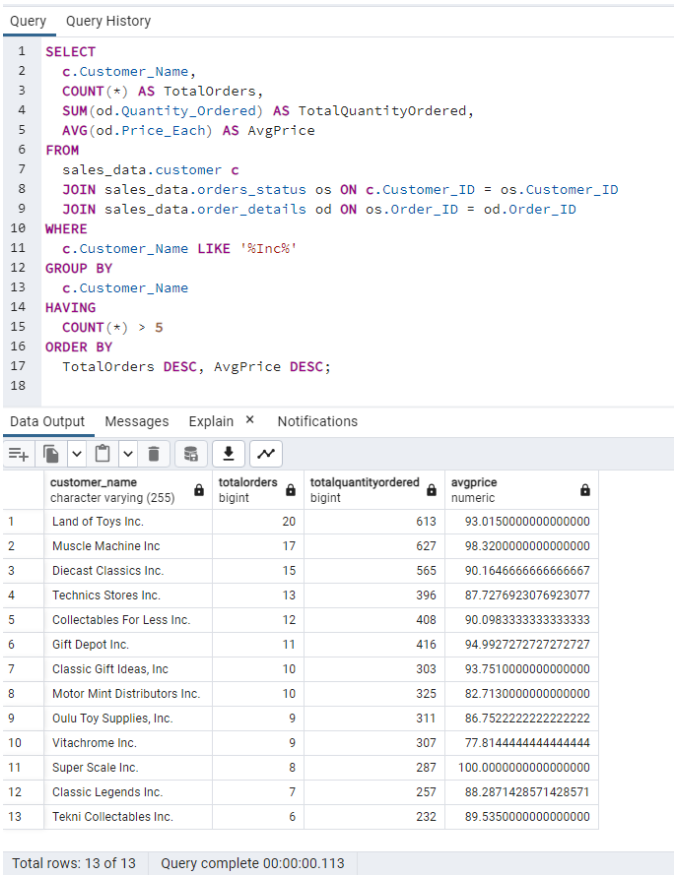


Fig. 15. Select Query 7

affect the latency of concurrent read and write operations as well as the overall throughput of the system. Particularly when scaling the system to meet our enterprise's ever-growing data demands, these insights are priceless.

C. Update Queries

We'll see the update queries we use in our database testing procedure in this subsection. Update queries are essential for determining how well the database can alter and handle current data. These queries were created to test different aspects of data modification, from straightforward updates of a single record to more intricate modifications based on conditions. When assessing the overall efficacy of our database system, the speed and precision of these update operations were crucial, especially in situations where frequent and dynamic data modifications were necessary.

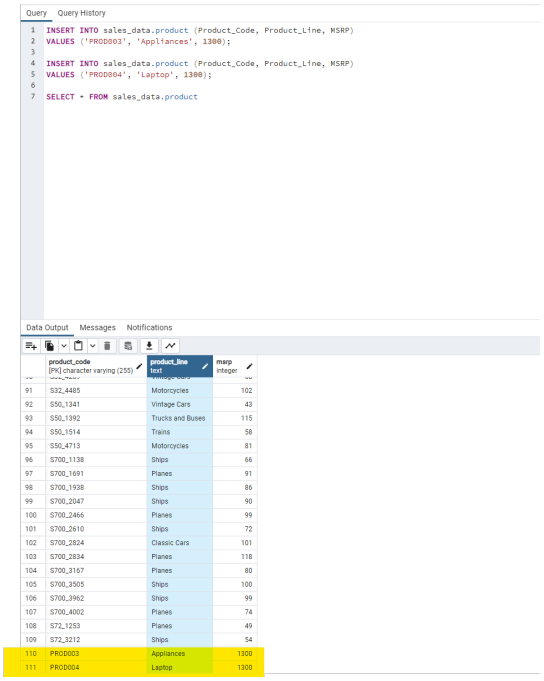


Fig. 16. Insert Query 1

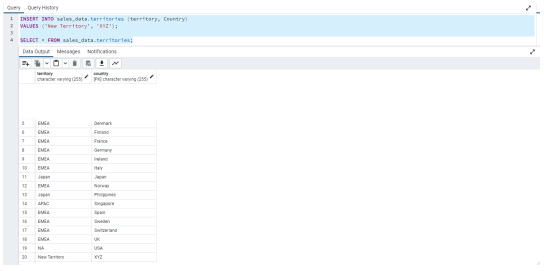


Fig. 17. Insert Query 2

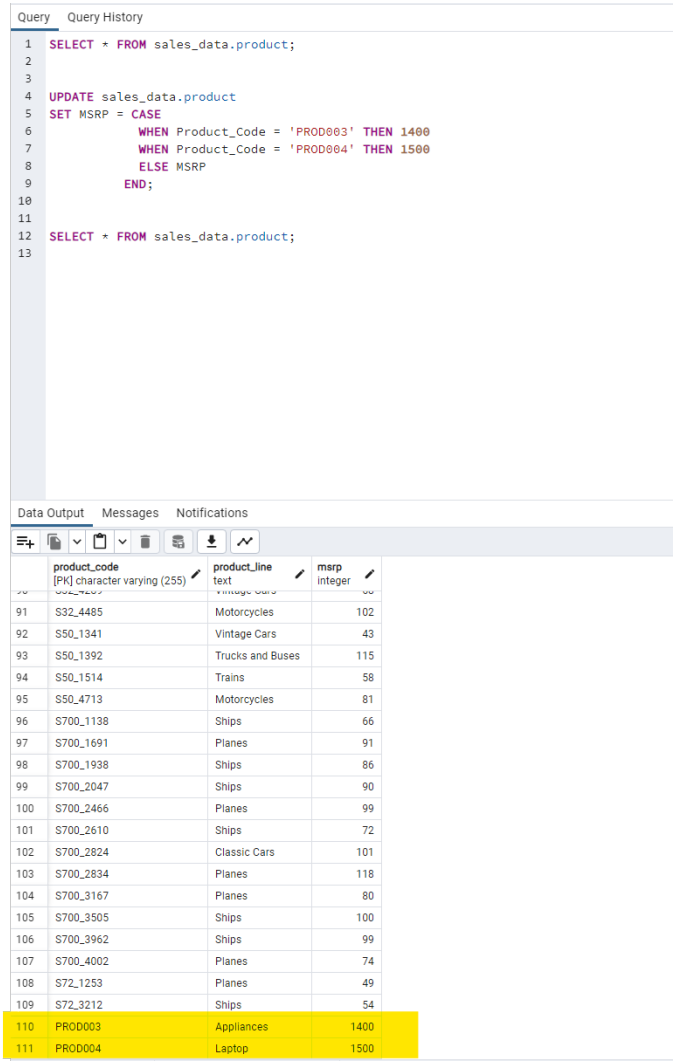


Fig. 18. Update Query 1

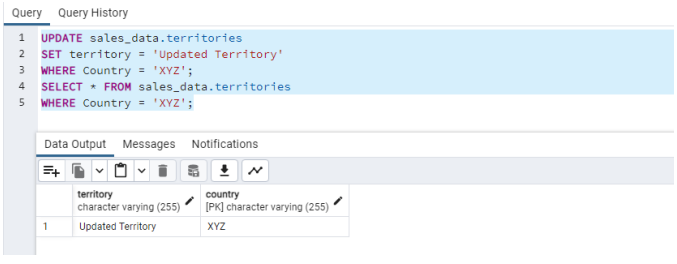
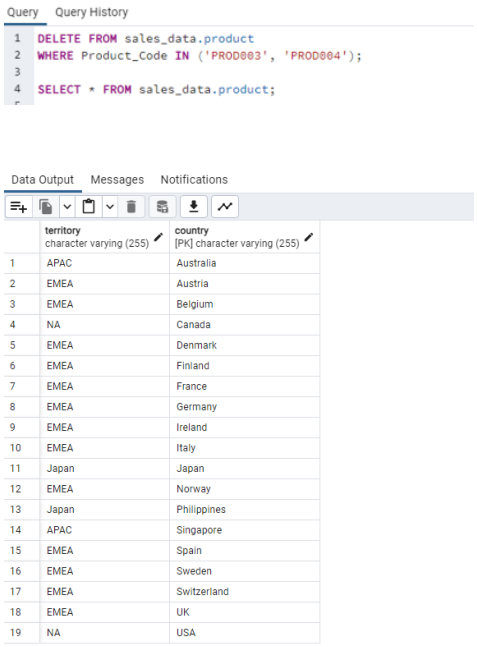


Fig. 19. Update Query 2

D. Delete Queries

We will examine the 'Delete Queries' used in our database testing process in this subsection. When evaluating the database's capacity to efficiently delete data records, delete queries are crucial. These queries were painstakingly designed to assess how well the database performed when it came to data deletion tasks, which included both the deletion of individual records and more intricate operations like conditional data removal. When assessing the overall performance of our database system, the effectiveness and precision of these delete operations were crucial, especially in situations where the smooth management of data removal was necessary.



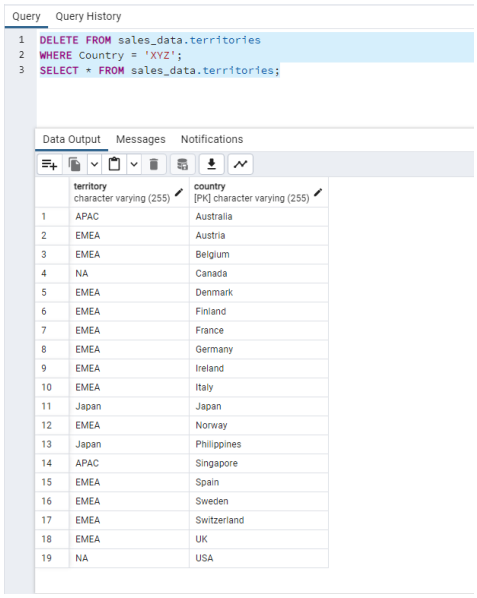
Query Query History

```
1  DELETE FROM sales_data.product
2  WHERE Product_Code IN ('PROD003', 'PROD004');
3
4  SELECT * FROM sales_data.product;
```

Data Output Messages Notifications

	territory character varying (255)	country [PK] character varying (255)
1	APAC	Australia
2	EMEA	Austria
3	EMEA	Belgium
4	NA	Canada
5	EMEA	Denmark
6	EMEA	Finland
7	EMEA	France
8	EMEA	Germany
9	EMEA	Ireland
10	EMEA	Italy
11	Japan	Japan
12	EMEA	Norway
13	Japan	Philippines
14	APAC	Singapore
15	EMEA	Spain
16	EMEA	Sweden
17	EMEA	Switzerland
18	EMEA	UK
19	NA	USA

Fig. 20. Delete Query 1



Query Query History

```
1  DELETE FROM sales_data.territories
2  WHERE Country = 'XYZ';
3  SELECT * FROM sales_data.territories;
```

Data Output Messages Notifications

	territory character varying (255)	country [PK] character varying (255)
1	APAC	Australia
2	EMEA	Austria
3	EMEA	Belgium
4	NA	Canada
5	EMEA	Denmark
6	EMEA	Finland
7	EMEA	France
8	EMEA	Germany
9	EMEA	Ireland
10	EMEA	Italy
11	Japan	Japan
12	EMEA	Norway
13	Japan	Philippines
14	APAC	Singapore
15	EMEA	Spain
16	EMEA	Sweden
17	EMEA	Switzerland
18	EMEA	UK
19	NA	USA

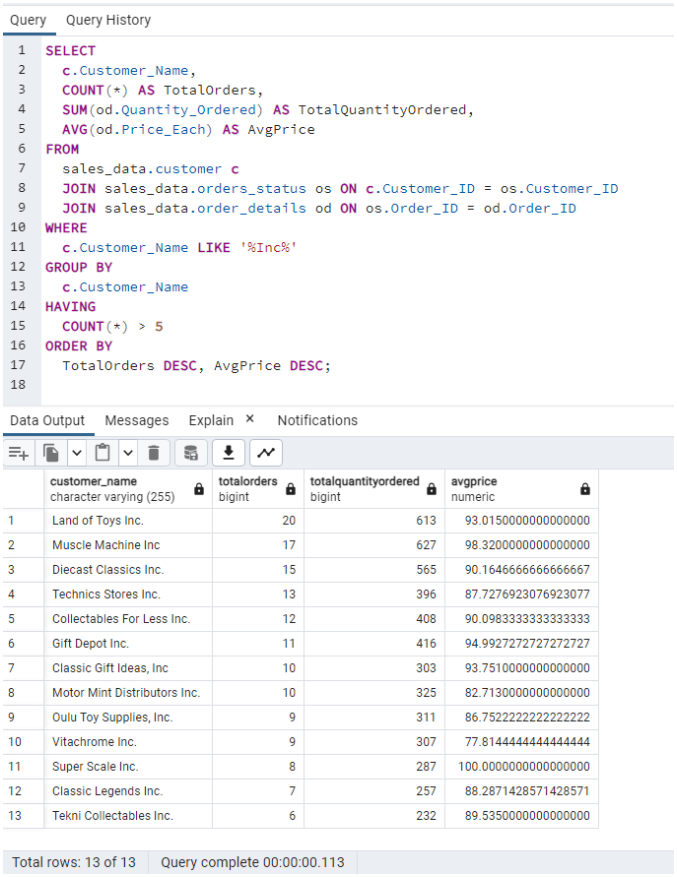
Fig. 21. Delete Query 2

VII. Problematic Queries

We were able to identify problematic queries that took a long time to execute based on the queries that were executed. We will go over these questions in-depth and look into possible optimisations in this section.

A. Problematic Query 1

Description and Analysis of the First Problematic Query



Query Query History

```
1  SELECT
2    c.Customer_Name,
3    COUNT(*) AS TotalOrders,
4    SUM(od.Quantity_Ordered) AS TotalQuantityOrdered,
5    AVG(od.Price_Each) AS AvgPrice
6  FROM
7    sales_data.customer c
8    JOIN sales_data.orders_status os ON c.Customer_ID = os.Customer_ID
9    JOIN sales_data.order_details od ON os.Order_ID = od.Order_ID
10 WHERE
11   c.Customer_Name LIKE '%Inc%'
12 GROUP BY
13   c.Customer_Name
14 HAVING
15   COUNT(*) > 5
16 ORDER BY
17   TotalOrders DESC, AvgPrice DESC;
18
```

Data Output Messages Explain Notifications

	customer_name character varying (255)	totalorders bigint	totalquantityordered bigint	avgprice numeric
1	Land of Toys Inc.	20	613	93.0150000000000000
2	Muscle Machine Inc	17	627	98.3200000000000000
3	Diecast Classics Inc.	15	565	90.1646666666666667
4	Technics Stores Inc.	13	396	87.7276923076923077
5	Collectables For Less Inc.	12	408	90.0983333333333333
6	Gift Depot Inc.	11	416	94.9927272727272727
7	Classic Gift Ideas, Inc	10	303	93.7510000000000000
8	Motor Mint Distributors Inc.	10	325	82.7130000000000000
9	Oulu Toy Supplies, Inc.	9	311	86.7522222222222222
10	Vitachrome Inc.	9	307	77.8144444444444444
11	Super Scale Inc.	8	287	100.0000000000000000
12	Classic Legends Inc.	7	257	88.2871428571428571
13	Tekni Collectables Inc.	6	232	89.5350000000000000

Total rows: 13 of 13 Query complete 00:00:00.113

Fig. 22. Problematic Query 1

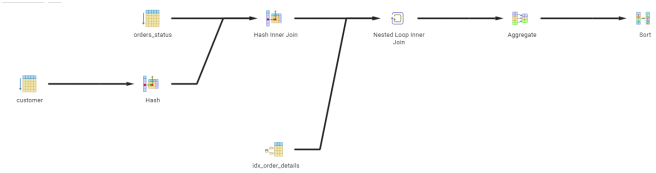


Fig. 23. Explain from PG Admin

Our database testing's first problematic query identified a sizable performance bottleneck. Long execution time for this specific query raised doubts about its effectiveness. We looked into possible optimisations to improve the query's performance as well as the causes of its prolonged execution time.

Optimized Solution

We came up with an optimised solution to deal with Problematic Query 1's performance problems. This approach resulted in a more effective execution process by rewriting the query to include filtering before joining. We were able

to improve overall database performance and drastically cut down on query execution time by putting this optimisation into practice.

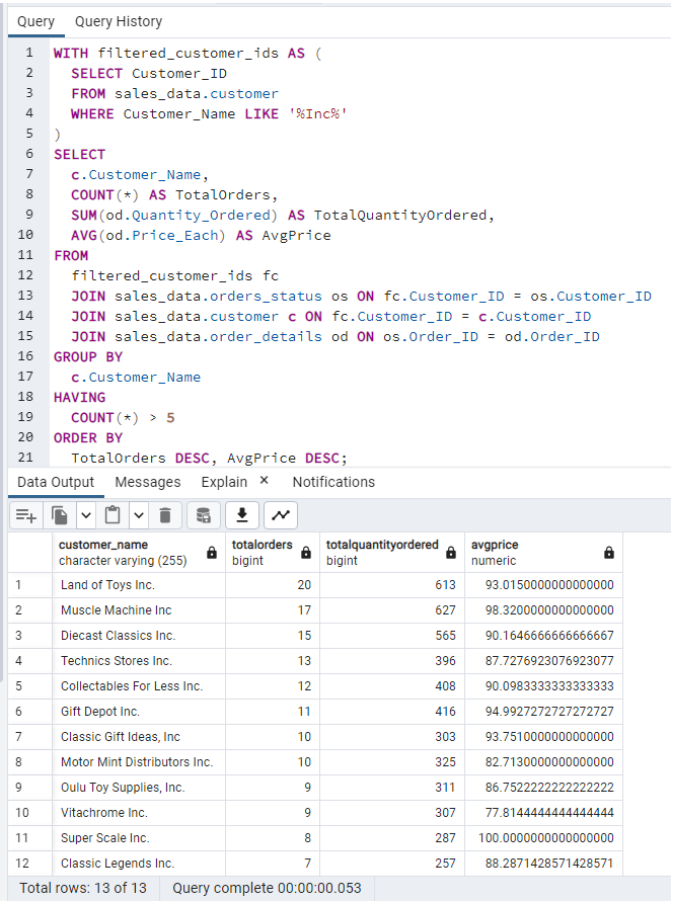


Fig. 24. Optimized Solution for Problematic Query 1

B. *Problematic Query 2*

Description and Analysis of the Second Problematic Query

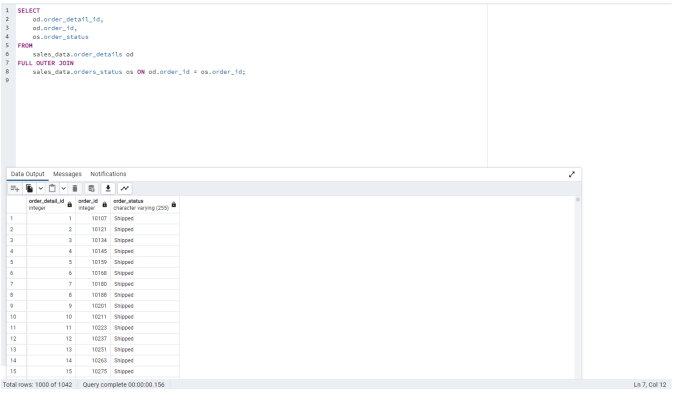


Fig. 25. Problematic Query 2

Again long execution time for this specific query raised doubts about its effectiveness. We looked into possible optimisations to improve the query's performance as well as the causes of its prolonged execution time.

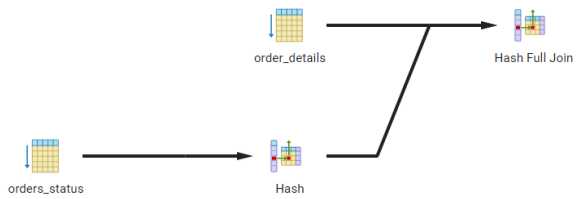


Fig. 26. Explain from PG Admin

Optimized Solution

Our approach to solving the performance problems with Problematic Query 2 was optimised. Applying indexing strategically was required for this. We achieved a significant reduction in query execution time and overall database performance improvement by putting these optimisations into practice.

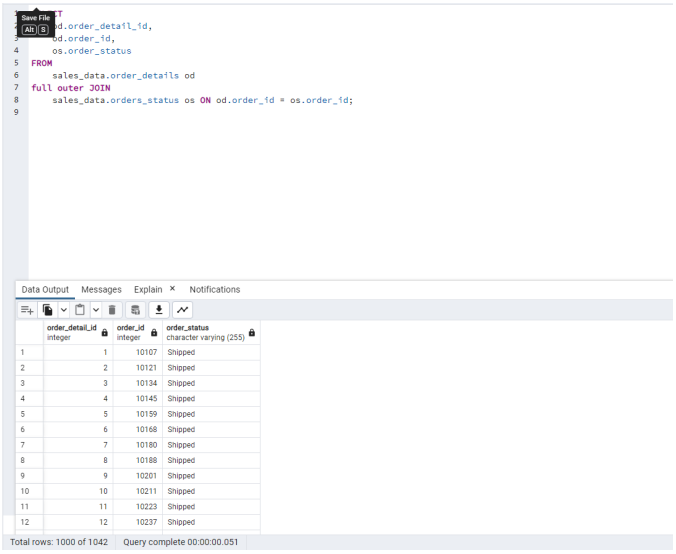


Fig. 27. Optimized Solution for Problematic Query 2

C. *Problematic Query 3*

Description and Analysis of the Third Problematic Query

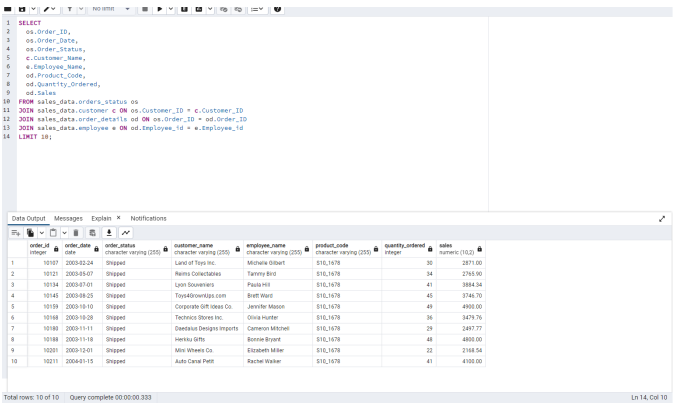


Fig. 28. Problematic Query 3

Again long execution time almost 333 milli seconds for this specific query raised doubts about its effectiveness. We looked

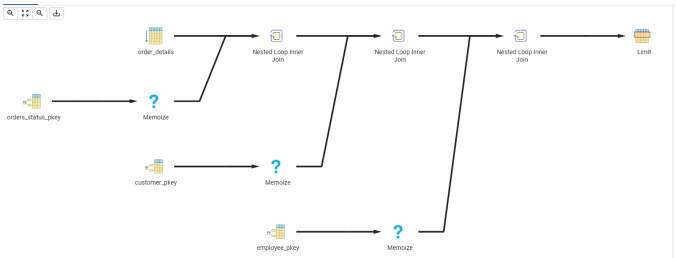


Fig. 29. Explain from PG Admin

into possible optimisations to improve the query's performance as well as the causes of its prolonged execution time.

Optimized Solution

The OrderID, CustomerID, and Employeeid columns in each of the corresponding tables in our database now have indexing implemented. We've significantly improved our query performance by enabling much faster lookup times for join operations by creating these indexes. This optimisation makes sure the database engine can find the required rows quickly rather than having to search through entire tables when our SQL queries involve joining tables on these indexed columns.

```sql
SELECT
  os.Order_ID,
  os.Order_Date,
  os.Order_Status,
  c.Customer_Name,
  e.Employee_Name,
  od.Product_Code,
  od.Quantity_Ordered,
  od.Sales
FROM
  (SELECT Order_ID, Order_Date, Order_Status, Customer_ID FROM sales_data.orders_status) os
JOIN
  sales_data.customer c ON os.Customer_ID = c.Customer_ID
JOIN
  sales_data.order_details od ON os.Order_ID = od.Order_ID
JOIN
  sales_data.employee e ON od.Employee_id = e.Employee_id
LIMIT 10;
```

Data Output | Messages | Explain | Notifications

	order_id integer	order_date date	order_status character varying (255)	customer_name character varying (255)	employee_name character varying (255)	product_code character varying (255)	quantity_ordered integer	sales numeric (10,2)
1	10107	2003-02-24	Shipped	Land of Toys Inc.	Michelle Gilbert	S10_1678	30	2871.00
2	10121	2003-05-07	Shipped	Reims Collectables	Tammy Bird	S10_1678	34	2765.90
3	10134	2003-07-01	Shipped	Lyon Souveniers	Paula Hill	S10_1678	41	3884.34
4	10145	2003-08-25	Shipped	Toys4GrownUps.com	Brett Ward	S10_1678	45	3746.70
5	10159	2003-10-10	Shipped	Corporate Gift Ideas Co.	Jennifer Mason	S10_1678	49	4900.00
6	10168	2003-10-28	Shipped	Technics Stores Inc.	Olivia Hunter	S10_1678	36	3479.76
7	10180	2003-11-11	Shipped	Daedalus Designs Imports	Cameron Mitchell	S10_1678	29	2497.77
8	10188	2003-11-18	Shipped	Herkku Gifts	Bonnie Bryant	S10_1678	48	4800.00
9	10201	2003-12-01	Shipped	Mini Wheels Co.	Elizabeth Miller	S10_1678	22	2168.54
10	10211	2004-01-15	Shipped	Auto Canal Petit	Rachel Walker	S10_1678	41	4100.00

Total rows: 10 of 10 | Query complete 00:00:00.042

Fig. 30. Optimized Solution for Problematic Query 3

VIII. Development of Flask Web Application

We developed a Flask web application that is tightly integrated with a PostgreSQL database. This application comprises two key routes: the root (`/`) route and the `/search` route. The root route is responsible for rendering the HTML template for the index page, while the `/search` route handles incoming POST requests for executing database searches.

Within the `/search` route, the application establishes a connection to the PostgreSQL database, utilizing parameters defined in the `db_params` configuration. To enhance the query generation process, we leveraged the LangChain library.

When a user submits a query through the input form on the index page, the LangChain processes this input and formulates an SQL query. Subsequently, this SQL query is executed against the connected database. The resulting data, including column names, is dynamically rendered on a dedicated results page.

Our application is designed with robustness in mind. It adeptly handles exceptions that may arise during query execution and ensures that the database connection is closed after each query, maintaining system integrity. Additionally, we have incorporated a conditional block in the script to facilitate running the Flask app in debug mode, simplifying the debugging and development process.

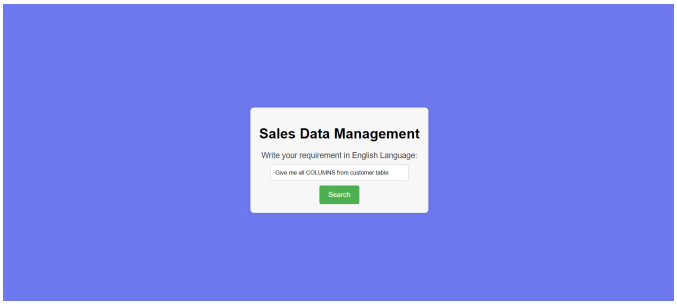


Fig. 31. Website's Home Page

Users can input queries here and accordingly will get the output like:

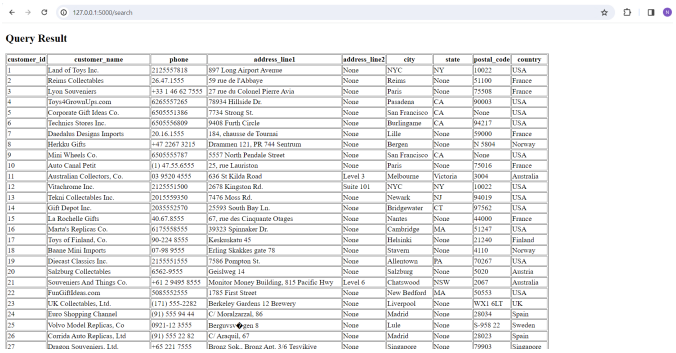


Fig. 32. Website's Result Page

Acknowledgment

I sincerely appreciate Professor Dr. Shamsad Parvin for their insightful advice. I also acknowledge all the TAs for their assistance. Both have significantly improved my educational experience.

References

https://www.kaggle.com/datasets/kyanyoga/sample-sales-data

https://lucid.app/

https://www.lucidchart.com/pages/er-diagrams

https://www.postgresqltutorial.com/

https://www.simplilearn.com/tutorials/sql-tutorial/er-diagram-in-dbms

https://python.langchain.com/docs/use$_cases/qa_structured/sql$

https://platform.openai.com/docs/api-reference/making-requests

https://flask.palletsprojects.com/en/3.0.x/
https://www.codecademy.com/article/sql-indexes
https://www.postgresqltutorial.com/postgresql-tutorial/import-csv-file-into-posgresql-table/