

Supply Chain Management System

*

1st Team Member
Narayanaroyal Marisetty
nmariset@buffalo.edu
50541106

2nd Team Member
Sanjana Sunchu
ssunchu@buffalo.edu
50538607

3rd Team Member
Ruthvik Bathala
rbathala@buffalo.edu
50544944

I. PROBLEM STATEMENT

Small retail businesses face a significant challenge in affording and utilizing sophisticated database management systems (DBMS), which hinders their ability to effectively manage and leverage their data, leading to difficulties in making informed business decisions and competing with larger enterprises. This is due to limited financial resources, lack of technical expertise, inflexibility of traditional DBMS solutions, and the need for customized solutions to meet unique and changing business needs. To address this issue, we aim to develop a low-cost and high-performance database system tailored for retail businesses, with a focus on streamlining operations, improving data accuracy, and tracking product information, managing sales data, monitoring supplier performance, handling logistics and tracking orders. This system will provide small retail businesses with a competitive edge, enabling them to efficiently store, process, and analyze their data, ultimately helping them grow and succeed in their respective markets.

II. WHY DO WE NEED A DATABASE INSTEAD OF AN EXCEL FILE?

In contrast to Excel files, databases offer distinct advantages, databases ensure data integrity through primary keys like CustomerID, ProductID, and OrderID, maintaining consistency across tables like Customers, Products, and Orders. While Excel lacks structured data types, databases ensure consistency with INT, VARCHAR, and FLOAT columns. Unlike Excel's limitations with real-time updates, databases seamlessly handle large-scale additions and enable efficient retrieval through indexing. With SQL queries, businesses extract insights more effectively than Excel's limited analytical capabilities. Additionally, databases allow concurrent access, unlike Excel's single-user restriction, providing robust security against unauthorized access.

III. TARGET USER

The users: The database is intended for

- **Supply Chain Managers:** They use the database to keep track of products, make sure they have the right amount in stock, and make the supply chain run smoothly and cost-effectively.
- **Sales and Marketing Teams:** They can utilize the database to analyze customer preferences and behavior,

track sales performance, manage customer relationships, and identify potential market opportunities, which helps in decision making to improve business.

- **Procurement Teams:** They use the database to optimize supplier relationships, track purchases, and minimize costs to ensure the company's procurement needs are met at the most competitive prices
- **Logistics Coordinators:** They use the database to coordinate shipments, track logistics, and optimize warehouse organization to ensure timely delivery and minimize costs.

The Administrator: Database administrators or the IT operations team in the company are responsible for maintaining and monitoring the database. They ensure the security, performance and recovery of the database by collaborating with the admin and various other departments in the company such as Sales, logistics, marketing and finance.

The Real-life scenario: Consider a scenario where an e-commerce company like Amazon. Sales representatives of Amazon handle customer inquiries, place orders, and track deliveries which helps them to keep track of each customer. The operations team manages inventory across warehouses, shipping logistics, and ensures timely delivery. The marketing team analyzes customer data to personalize recommendations and promotions. Also, Database administrators oversee Amazon's vast database infrastructure, ensuring high availability, data integrity, and security.

Data Source and Cleaning Steps: Extracted the data from Mendeley Data resource and Read the data which is the form of comma separated file using pandas library. Checked the null values of data and dropped those features which are null across the majority of the records. Mapped the values of the specific feature "Late delivery risk" to no or yes in order to better interpret the information. Performed Feature Engineering for the features "OrderDate", "ShippingDate" which is splitted into Order time and date, shipping date and times. Mapped the values of the specific feature "Product Status" to not available/available in order to better interpret the information. Also dropped the duplicated records and after performing all the steps now the data is updated and cleaned.

IV. ER DIAGRAM

The figure below illustrates the Entity relationships

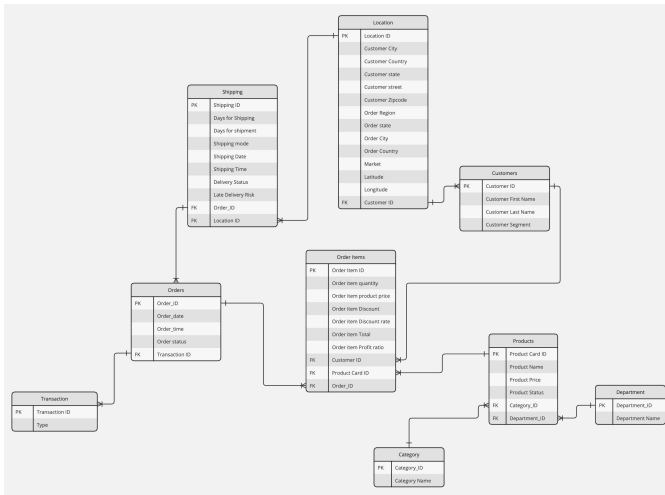


Fig. 1.

V. DATA SCHEMA

Customer's Data Has the details of the Customer's Name and what segment they fall under.

Department's Data It has the details of the department name where all the products belong to.

Category's Data Represents the Category of the product.

Product's Data Has all the details of the Product's Name, price, status.

Transaction's Data Represents the type of the category the customer do.

Orders's Data It has all the details of the orders including the ordertime, date, status.

OrderItems's Data Represents order item details, including ID, quantity, price, discounts, total amount, profit ratio, and references to customer, product, and order ID.

Location's Data Represents location and order information for customer purchases, including details like city, country, state, street, zipcode, order destination, region, market, and geographical coordinates, linked to customer IDs.

Shipping's Data Represents shipping information for orders, including shipping details such as ID, duration, mode, date, time, delivery status, and associated order and location IDs.

VI. RELATIONSHIP BETWEEN TABLES

- 1) **Location and Shipping:** One-to-Many, A location can have many shipments sent to it, but each shipment is sent to only one location.
- 2) **Location and Customers:** Many-to-One, Many customers may share the same location, but each customer primarily belongs to one location

- 3) **Shipping and Orders:** One-to-Many, Each shipping can have multiple orders.
- 4) **Orders and Transaction:** One-to-One, Each order is a transaction.
- 5) **Orders and Order Items:** One-to-Many, as each order contains multiple order items.
- 6) **Customers and Order Items:** One-to-Many, Each customer will order multiple items.
- 7) **Product and Order Items:** Many-to-One, Each product can have multiple order items.
- 8) **Products and Category:** Many-to-One, with multiple products belonging to a single category.
- 9) **Products and Department:** Many-to-One, if each product is associated with a single department, which can encompass multiple products.

VII. ATTRIBUTES

Customer's Data

- *CustomerID INT (Primary Key) : Id is the customer's ID which is used to uniquely define the customer's data and will not be null.*
- *CustomerFirstName VARCHAR : Represents the first name of the customer and will not be null.*
- *CustomerLastName VARCHAR : Represents the last name of the customer and will not be null.*
- *CustomerSegment VARCHAR : Represents the type of the customer and will not be null.*

Department's Data

- *DepartmentID INT (Primary Key) : Id is the Department's ID which is used to uniquely define the Department's data and will not be null.*
- *DepartmentName VARCHAR : Represents the department name of the department and will not be null.*

Category's Data

- *CategoryID INT (Primary Key) : Id is the Category's ID which is used to uniquely define the Category's data and will not be null.*
- *CategoryName VARCHAR : Represents the category name of the category and will not be null.*

Product's Data

- *ProductID INT (Primary Key) : Id is the Product's ID which is used to uniquely define the Product's data and will not be null.*
- *ProductName VARCHAR : Represents the product name of the Products and will not be null.*
- *ProductPrice FLOAT : Represents the price of the product and will not be null.*
- *ProductStatus VARCHAR : Represents the status of the product stock and will not be null.*
- *CategoryID INT (Foreign Key to Category table) : Represents the category id from category table which is based on category results and will not be null.*

- *DepartmentID VARCHAR (Foreign Key to Department table) : Represents the Department id from the Department table which is based on Department results and will not be null.*

- *CategoryName VARCHAR : Represents the category name of the category and will not be null.*

Transaction's Data

- *TransactionID INT (Primary Key) : Id is the Transaction's ID which is used to uniquely define the Transaction's data and will not be null.*

- *TransactionType VARCHAR : Represents the type of the category the customer do and will not be null.*

Order's Data

- *OrderID INT (Primary Key) : Id is the Product's ID which is used to uniquely define the Product's data and will not be null.*

- *OrderDate TEXT : Represents the Dates of the order and will not be null.*

- *OrderTime TIME : Represents the time of the order and will not be null.*

- *OrderStatus VARCHAR : Represents the status of the product stock and will not be null.*

- *TransactionID INT (Foreign Key to Transaction table) : Represents the Transaction id from Transaction table which is based on Transaction results and will not be null.*

OrderItem's Data

- *OrderitemID INT (Primary Key) : Id is the Orderitem's ID which is used to uniquely define the Orderitem's data and will not be null.*

- *OrderitemQuantity INT : Represents the number of products per order and will not be null.*

- *OrderitemProductPrice FLOAT : Represents the price of the product without discount and will not be null.*

- *OrderitemDiscount FLOAT : Represents the order item discount value and will not be null.*

- *OrderitemDiscountRate FLOAT : Represents the Order item discount percentage and will not be null.*

- *OrderitemTotal FLOAT : Represents the Total amount per order and will not be null.*

- *OrderitemProfitRatio FLOAT : Represents the Order Item Profit Ratio and will not be null.*

- *CustomerID INT (Foreign Key to Customers table) : Represents the Customer id from Customer's table which is based on Customer results and will not be null.*

- *ProductID INT (Foreign Key to Products table) : Represents the Product id from Product's table which is based on Product results and will not be null.*

- *OrderID INT (Foreign Key to Orders table) : Represents the Order id from Order's table which is based on Order results and will not be null*

Location's Data

- *LocationID INT (Primary Key) : Id is the Location's ID which is used to uniquely define the Location's data and will not be null.*

- *CustomerCity VARCHAR : Represents the City where the customer made the purchase and will not be null.*

- *CustomerCountry VARCHAR : Represents the Country where the customer made the purchase and will not be null.*

- *CustomerState CHAR(2) : Represents the State to which the store where the purchase is registered belongs to and will not be null.*

- *CustomerStreet TEXT : Represents the Street to which the store where the purchase is registered belongs to and will not be null.*

- *CustomerZipcode INT : Represents the Customer Zipcode and will not be null.*

- *OrderRegion VARCHAR : Represents the Region of the world where the order is delivered and will not be null.*

- *OrderCity VARCHAR : Represents the Destination city of the order and will not be null.*

- *OrderState VARCHAR : Represents the State of the region where the order is delivered and will not be null.*

- *OrderCountry VARCHAR : Represents the Destination country of the order and will not be null.*

- *Market VARCHAR : Represents the Market to where the order is delivered and will not be null.*

- *Latitude FLOAT : Represents the Latitude corresponding to location of store and will not be null.*

- *Longitude FLOAT : Represents the Longitude corresponding to location of store and will not be null.*

- *CustomerID INT (Foreign Key to Customers table) : Represents the Customer id from Customer's table which is based on Customer results and will not be null.*

Shipping's Data

- *ShippingID INT (Primary Key) : Id is the Shipping's ID which is used to uniquely define the Shipping's data and will not be null.*

- *DaysForShipping INT : Represents the City where the customer made the purchase and will not be null.*

- *DaysForShipment INT : Represents the Country where the customer made the purchase and will not be null.*

- *Shippingmode CHAR(2) : Represents the mode of the shipping and will not be null.*

- *ShippingDate TEXT : Represents the date of the shipment and will not be null.*

- *ShippingTime INT : Represents the time of the shipment and will not be null.*

- *DeliveryStatus VARCHAR : Represents the Delivery status of orders and will not be null.*

- *LateDeliveryRisk VARCHAR : Represents whether the delivery is late or not and will not be null.*

- *OrderID VARCHAR (Foreign Key to Orders table) : Represents the Order id from Order's table which is based on Order results and will not be null.*

- *LocationID VARCHAR (Foreign Key to Location table) : Represents the Location id from Location's table which is based on Location results and will not be null.*

VIII. ACTIONS ON FOREIGN KEY WHEN PRIMARY KEY DELETED:

When deleting an Order record with OrderID = 100 referenced by OrderItems, outcomes vary based on the foreign key action:

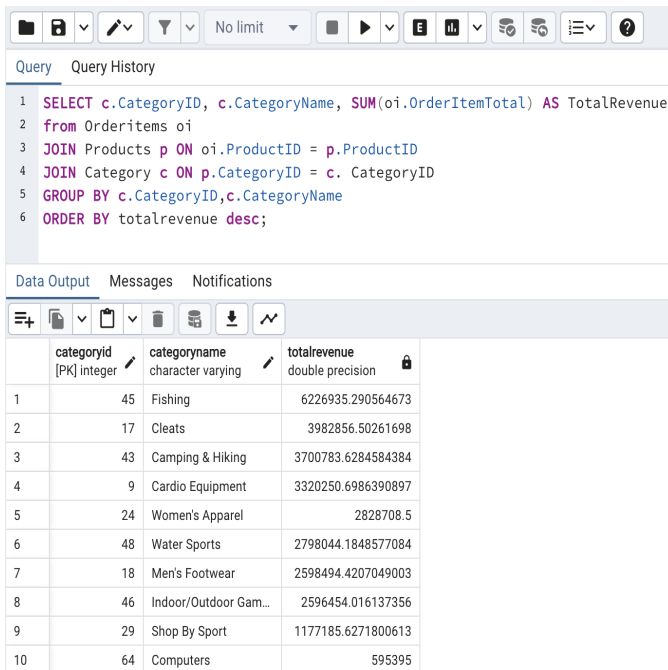
- **NO ACTION:** Blocks the deletion of Order if OrderItems reference it, preventing data inconsistency. Attempting deletion results in error.
- **CASCADE:** Automatically deletes all OrderItems linked to OrderID = 100, maintaining data integrity.
- **SET NULL:** Updates OrderID in related OrderItems to NULL, effectively disconnecting them from the deleted Order.
- **SET DEFAULT:** Changes OrderID in OrderItems to a predefined default value (e.g., 0) when the Order is deleted.

IX. APPROACH USED TO CREATE TABLES FROM CSV:

The main table contains various fields related to sales and customer information, including order details such as type, shipping duration, sales per customer, delivery status, and product details like name and price. It also stores customer demographics like name, address, and segment, along with geographic coordinates. So, We created a main table comprising of all the attributes to import the data from the CSV file, from which required data can be inserted into appropriate tables.

X. PERFORMED QUERIES:

Query 1 : Sales Distribution Across Regions

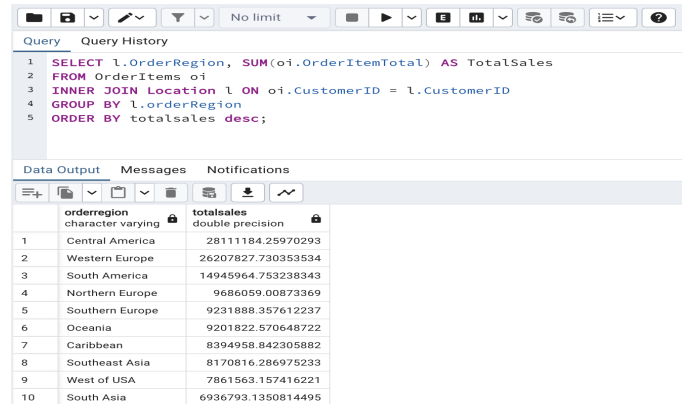


```
1 SELECT c.CategoryID, c.CategoryName, SUM(oi.OrderItemTotal) AS TotalRevenue
2 FROM OrderItems oi
3 JOIN Products p ON oi.ProductID = p.ProductID
4 JOIN Category c ON p.CategoryID = c.CategoryID
5 GROUP BY c.CategoryID, c.CategoryName
6 ORDER BY totalrevenue desc;
```

	categoryid [PK] integer	categoryname character varying	totalrevenue double precision
1	45	Fishing	6226935.290564673
2	17	Cleats	3982856.50261698
3	43	Camping & Hiking	3700783.6284584384
4	9	Cardio Equipment	3320250.6986390897
5	24	Women's Apparel	2828708.5
6	48	Water Sports	2798044.1848577084
7	18	Men's Footwear	2598494.4207049003
8	46	Indoor/Outdoor Gam...	2596454.016137356
9	29	Shop By Sport	1177185.6271800613
10	64	Computers	595395

Fig. 2.

Query 2 : Total revenue generated by each category

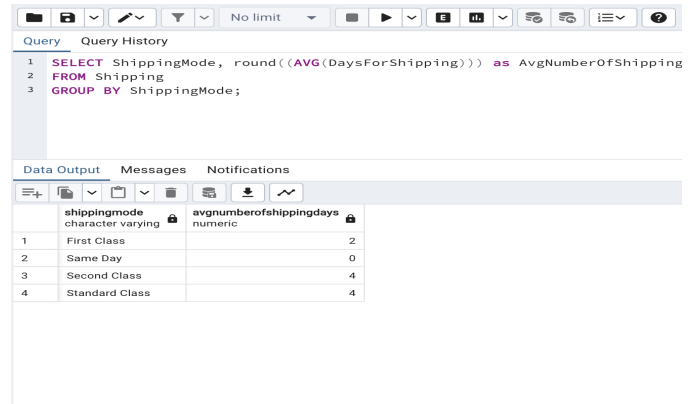


```
1 SELECT l.OrderRegion, SUM(oi.OrderItemTotal) AS TotalSales
2 FROM OrderItems oi
3 INNER JOIN Location l ON oi.CustomerID = l.CustomerID
4 GROUP BY l.orderRegion
5 ORDER BY totalsales desc;
```

orderregion character varying	totalsales double precision
Central America	28111184.25970293
Western Europe	26207827.730353534
South America	14945964.753238343
Northern Europe	9686059.00873369
Southern Europe	9231888.357612237
Oceania	9201822.570648722
Caribbean	8394958.842305882
Southeast Asia	8170816.286975233
West of USA	7861563.157416221
South Asia	6936793.1350814495

Fig. 3.

Query 3 : Average shipping days by shipping mode

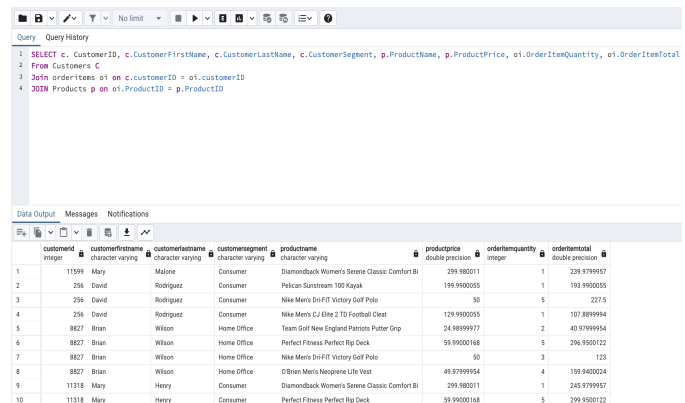


```
1 SELECT ShippingMode, round((AVG(DaysForShipping))) as AvgNumberOfShipping
2 FROM Shipping
3 GROUP BY ShippingMode;
```

shippingmode character varying	avgnumberofshippingdays numeric
First Class	2
Same Day	0
Second Class	4
Standard Class	4

Fig. 4.

Query 4 : The below query fetches data on customers and their purchases, including customer ID, first name, last name, segment, and product details like ID, name, price, quantity, and total spent per product



```
1 SELECT c.CustomerID, c.CustomerFirstName, c.CustomerLastName, c.CustomerSegment, p.ProductName, p.ProductPrice, oi.OrderItemQuantity, oi.OrderItemTotal
2 FROM Customers c
3 JOIN OrderItems oi ON c.CustomerID = oi.CustomerID
4 JOIN Products p ON oi.ProductID = p.ProductID
```

customerid integer	customerfirst character varying	customerlast character varying	customersegment character varying	productname character varying	productprice double precision	orderitemquantity integer	orderitemtotal double precision
11599	Mary	Malone	Consumer	Diamondback Women's Serene Classic Comfort BI	299.180011	1	239.9799957
256	David	Rodriguez	Consumer	Pelican Sunstream 100 Kayak	199.9900055	1	193.9900055
256	David	Rodriguez	Consumer	Nike Men's Dri-FIT Victory Golf Polo	90	5	227.5
256	David	Rodriguez	Consumer	Nike Men's C2 Ultra 2 TD Football Cleat	159.9900055	1	107.9999994
8827	Brian	Wilson	Home Office	Team Golf New England Pannos Putter Grip	24.9899977	2	45.9799954
8827	Brian	Wilson	Home Office	Perfect Fitness Perfect Flip Deck	59.99000168	5	296.9500132
8827	Brian	Wilson	Home Office	Nike Men's Dri-FIT Victory Golf Polo	90	3	123
8827	Brian	Wilson	Home Office	O'Brien Meris Neoprene Life Vest	49.97999954	4	159.9400024
11318	Mary	Henry	Consumer	Diamondback Women's Serene Classic Comfort BI	299.180011	1	245.9799957
11318	Mary	Henry	Consumer	Perfect Fitness Perfect Flip Deck	59.99000168	5	299.9500132

Fig. 5.

XI. BCNF PROOF:

For a table to be in BCNF, it must satisfy the condition that every non-trivial functional dependency has its left-hand side as a superkey. The following Functional dependencies are listed for every table based on the domain knowledge.

Customer's Table:

- CustomerID \rightarrow CustomerFirstName, CustomerLastName, CustomerSegment

Department's Table:

- DepartmentID \rightarrow DepartmentName

Category's Table:

- CategoryID \rightarrow CategoryName

Product's Table:

- ProductID \rightarrow ProductName, ProductPrice, ProductStatus, CategoryID, DepartmentID

Transaction's Table:

- TransactionID \rightarrow TransactionType

Order's Table:

- OrderID \rightarrow OrderDate, OrderTime, OrderStatus, TransactionID

OrderItem's Table:

- OrderItemID \rightarrow OrderitemQuantity, OrderitemProductPrice, OrderitemDiscount, OrderitemDiscountRate, OrderitemTotal, OrderitemProfitRatio, CustomerID, ProductID, OrderID

Location's Table:

- LocationID \rightarrow CustomerCity, CustomerCountry, CustomerState, CustomerStreet, CustomerZipcode, OrderRegion, OrderCity, OrderState, OrderCountry, Market, Latitude, Longitude, CustomerID

Shipping's Table:

- ShippingID \rightarrow DaysForShipping, DaysForShipment, ShippingMode, ShippingDate, ShippingTime, DeliveryStatus, LateDeliveryRisk, OrderID, LocationID

Each table's non-trivial dependencies have a primary key on the left-hand side, which functionally determines all other attributes in the table. This condition confirms that the left-hand side of each functional dependency is a super key, satisfying the requirement for BCNF in all tables.

For Example: Consider the functional dependency from the Customer's Table. The left-hand side, CustomerID, is a primary key and it uniquely determines all other attributes in the table. As a result, it qualifies as a superkey. Additionally, the right-hand side of the functional dependency does not contain any attribute from the left-hand side, fulfilling the BCNF condition of non-triviality. Therefore, it can be concluded that the given table satisfies the BCNF criteria.

Similarly, the remaining tables exhibit the same characteristics as the Customer's Table. Hence, they all adhere to the BCNF criteria, eliminating the need for further decomposition.

Since all tables are already in BCNF, no further decomposition is required. Therefore, **the Entity-Relationship (E/R) diagram remains unchanged** from its initial state.

XII. USE OF INDEXING:

When handling with larger dataset, one of the primary challenges encountered was the longer retrieval times for queries, especially those involving grouping operations. As the dataset size increased, the time taken to search through the data and retrieve relevant information also increased significantly.

To address this issue, indexing concepts were adopted to improve retrieval efficiency. The execution time taken for a query before indexing can be addressed in figure 6.

Query	Query History
1	explain analyse SELECT ShippingMode, round ((AVG (DaysForShipping))) as AvgNumberOfShippingDays
2	FROM Shipping
3	GROUP BY ShippingMode;
4	

Data Output	Messages	Notifications
QUERY PLAN		
text		
10	Worker 1: Sort Method: quicksort Memory: 23kB	
11	-> Partial HashAggregate (cost=6439.42..6439.46 rows=4 width=45) (actual time=41.797..41.798 rows=1 loops=3)	
12	Group Key: shippingmode	
13	Batches: 1 Memory Usage: 24kB	
14	Worker 0: Batches: 1 Memory Usage: 24kB	
15	Worker 1: Batches: 1 Memory Usage: 24kB	
16	-> Parallel Seq Scan on shipping (cost=0.00..5752.61 rows=137361 width=17) (actual time=0.004..9.542 rows=109889 loop...	
17	Planning Time: 0.181 ms	
18	Execution Time: 142.441 ms	
Total rows: 18 of 18 Query complete 00:00:00.177		

Fig. 6.

The execution time taken for a query after use of indexing can be addressed in figure 7.

Query	Query History
1	explain analyse SELECT ShippingMode, round ((AVG (DaysForShipping))) as AvgNumberOfShippingDays
2	FROM Shipping
3	GROUP BY ShippingMode;
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	

Data Output	Messages	Notifications
QUERY PLAN		
text		
10	Worker 1: Sort Method: quicksort Memory: 23kB	
11	-> Partial HashAggregate (cost=6439.42..6439.46 rows=4 width=45) (actual time=43.887..43.889 rows=3 loops=3)	
12	Group Key: shippingmode	
13	Batches: 1 Memory Usage: 24kB	
14	Worker 0: Batches: 1 Memory Usage: 24kB	
15	Worker 1: Batches: 1 Memory Usage: 24kB	
16	-> Parallel Seq Scan on shipping (cost=0.00..5752.61 rows=137361 width=17) (actual time=0.010..9.900 rows=109889 loop...	
17	Planning Time: 2.735 ms	
18	Execution Time: 113.652 ms	

Fig. 7.

Overall, by addressing the retrieval time issue through indexing, able to improve the performance of the database when handling larger datasets, ensuring that queries were executed more efficiently.

XIII. QUERYING:

Query 1: Find customers whose total order value exceeds a specified limit, useful for identifying high-value customers.

Query Query History	
<pre>1 SELECT 2 c.CustomerFirstName, 3 c.CustomerLastName, 4 SUM(o1.OrderItemTotal) AS TotalOrderValue 5 FROM Customers c 6 JOIN OrderItems oi ON c.CustomerID = oi.CustomerID 7 GROUP BY c.CustomerID 8 HAVING SUM(o1.OrderItemTotal) > 1000 9 ORDER BY TotalOrderValue DESC;</pre>	
Data Output Messages Notifications	
customerfirstname	customerlastname
character varying	character varying
1 Mary	Smith
2 Mary	Duncan
3 Mary	Patterson
4 Betty	Spears
5 Mary	Butler
6 Betty	Phillips
7 Ashley	Smith
8 Jerry	Smith
9 Kelly	Smith
Total rows: 1000 of 11319 Query complete 00:00:00.214	

Fig. 8.

Query 2: Most Popular Product in Each Category.

Query Query History	
<pre>1 SELECT CategoryName, ProductName, ProductSales 2 FROM (3 SELECT 4 c.CategoryName, p.ProductName, Sales.ProductSales, 5 RANK() OVER (PARTITION BY c.CategoryID ORDER BY Sales.ProductSales DESC) as SalesRank 6 FROM Category c 7 JOIN Products p ON p.CategoryID = c.CategoryID 8 JOIN (9 SELECT ProductID, SUM(OrderItemTotal) AS ProductSales 10 FROM OrderItems 11 GROUP BY ProductID 12) AS Sales ON p.ProductID = Sales.ProductID 13) AS RankedProducts 14 WHERE SalesRank = 1;</pre>	
Data Output Messages Notifications	
categoryname	productname
character varying	character varying
1 Soccer	Elevation Training Mask 2.0
2 Baseball & Softball	adidas Men's F10 Messi TRX FG Soccer Cleat
3 Basketball	SOLE E25 Elliptical
4 Lacrosse	Under Armour Men's Tech II T-Shirt
5 Tennis & Racquet	Nike Men's Comfort 2 Slide
6 Hockey	Nike Women's Legend V-Neck T-Shirt
7 Cardio Equipment	Nike Men's Free 5.0+ Running Shoe
8 Strength Training	SOLE E35 Elliptical
9 Fitness Accessories	Under Armour Hustle Storm Medium Duffel Bag
Total rows: 51 of 51 Query complete 00:00:00.128	

Fig. 9.

Query 3: Top 10 highest-selling products

Query Query History	
<pre>308 SELECT ProductName, SUM(OrderItemTotal) AS TotalSales 309 FROM OrderItems 310 JOIN Products ON OrderItems.ProductID = Products.ProductID 311 GROUP BY ProductName 312 ORDER BY TotalSales DESC 313 LIMIT 10;</pre>	
Data Output Messages Explain X Notifications	
productname	totalsales
character varying	double precision
1 Field & Stream Sportsman 16 Gun Fire Safe	6226935.290565484
2 Perfect Fitness Perfect Rip Deck	3973180.3627707656
3 Diamondback Women's Serene Classic Comf...	3700783.628458234
4 Nike Men's Free 5.0+ Running Shoe	3295693.298668299
5 Nike Men's Dri-FIT Victory Golf Polo	2828708.5
6 Pelican Sunstream 100 Kayak	2785518.0852497993
7 Nike Men's CJ Elite 2 TD Football Cleat	2598494.420704925
8 O'Brien Men's Neoprene Life Vest	2596454.0161372237
9 Under Armour Girls' Toddler Spine Surge Runni	1140770.9567904666

Fig. 10.

Query 4: Insert Values into Customer's table

Query Query History	
<pre>286 INSERT INTO Customers (CustomerID, CustomerFirstName, CustomerLastName, CustomerSegment) 287 VALUES (180518, 'John', 'Doe', 'Consumer'); 288 289 select * from customers where CustomerID = 180518</pre>	
Data Output Messages Explain X Notifications	
customerid	customerfirstname
[PK] integer	character varying
1 180518	John
customerlastname	customersegment
character varying	character varying
	Consumer

Fig. 11.

Query 5: Update Values in Customer's table

Query Query History	
<pre>293 UPDATE Customers 294 SET CustomerSegment = 'Corporate' 295 WHERE CustomerID = 180518; 296 297 select * from customers where CustomerID = 180518</pre>	
Data Output Messages Explain X Notifications	
customerid	customerfirstname
[PK] integer	character varying
1 180518	John
customerlastname	customersegment
character varying	character varying
	Corporate

Fig. 12.

Query 6: Delete specific record in Customer's table

Query Query History	
<pre>1 DELETE FROM Customers 2 WHERE CustomerID = 180518;</pre>	
Data Output Messages Explain X Notifications	
DELETE 1	
Query returned successfully in 90 msec.	

Fig. 13.

Query 7: Products with the highest profit margins

Query	Query History
316	SELECT
317	p.ProductName,
318	AVG(o1.OrderItemProfitRatio) AS AvgProfitMargin
319	FROM Products p
320	JOIN OrderItems o1 ON p.ProductID = o1.ProductID
321	GROUP BY p.ProductID
322	ORDER BY AvgProfitMargin DESC
323	LIMIT 10;
324	
325	
326	
327	
328	
329	
330	

productname	avgprofitmargin
character varying	double precision
1 Polar FT4 Heart Rate Monitor	0.2398333364999998
2 Bowflex SelectTech 1090 Dumbbells	0.23299999953
3 Diamondback Boys' Insight 24 Performanc...	0.22551724331034487
4 Elevation Training Mask 2.0	0.1940540529459459
5 MDGolf Pittsburgh Penguins Putter	0.19226414898113212
6 Ogio Race Golf Shoes	0.19147540927868856
7 Bag Boy M330 Push Cart	0.18449275415942024
8 Merrell Men's All Out Flash Trail Running Sho	0.17658823412941174
9 adidas Brazuca 2014 Official Match Ball	0.1755384594615384
Total rows: 10 of 10	Query complete 00:00:00.293

Fig. 14.

Query 8: The Top 3 most profitable products in each category

Query

Query History

```
334 WITH RankedProducts AS (  
335     SELECT  
336         c.CategoryID,c.CategoryName,p.ProductName,  
337         SUM(o1.OrderItemTotal) AS TotalSales,  
338         RANK() OVER (PARTITION BY c.CategoryID ORDER BY SUM(o1.OrderItemTotal) DESC) AS SalesRank  
339     FROM Category c  
340     JOIN Products p ON c.CategoryID = p.CategoryID  
341     JOIN OrderItems o1 ON p.ProductID = o1.ProductID  
342     GROUP BY c.CategoryID, c.CategoryName, p.ProductName  
343 )  
344 SELECT  
345     CategoryName,  
346     ProductName,  
347     TotalSales  
348 FROM RankedProducts  
349 WHERE SalesRank <= 3;  
350  
351
```

Query Output

Messages

Explain

Notifications

	categoryname	productname	totalsales
	character varying	character varying	double precision
1	Soccer	Elevation Training Mask 2.0	16653.069851989996
2	Soccer	Nike Men's Free RN Max Training Shoe	7256.879884560005
3	Baseball & Softball	adidas Men's F10 Messi TRX FG Soccer Cleat	50354.21066776998
4	Baseball & Softball	adidas Kids' FS Messi FG Soccer Cleat	24626.510212519977
5	Baseball & Softball	adidas Brazuca 2014 Official Match Ball	9386.549972200002
6	Basketball	SOLE E25 Elliptical	9414.899902
Total rows: 6 of 6			
Query complete 00:00:00.107			

Fig. 15.

XIV. QUERY EXECUTION ANALYSIS

Query 1: This query involves multiple table joins and an aggregation operation. Each join and aggregation operation result in a high execution cost due to full table scans or inefficient join algorithms.

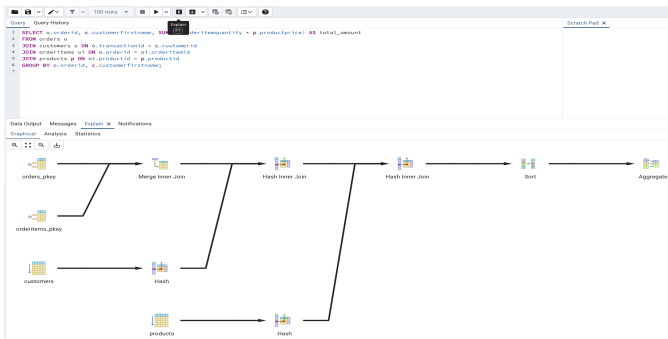


Fig. 16.

Query 2: This query involves a LEFT JOIN operation followed by a GROUP BY aggregation. The database needs to perform full table scans or inefficient join algorithms leading to high execution costs.

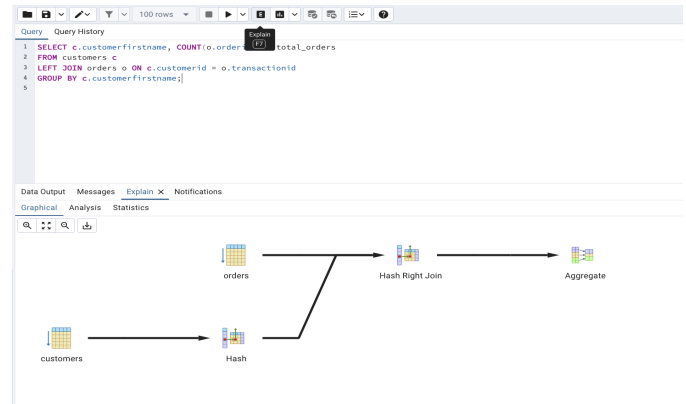


Fig. 17.

Query 3: This query involves multiple table joins and a COUNT(DISTINCT) aggregation operation. The database needs to perform full table scans or inefficient join algorithms, leading to high execution costs.

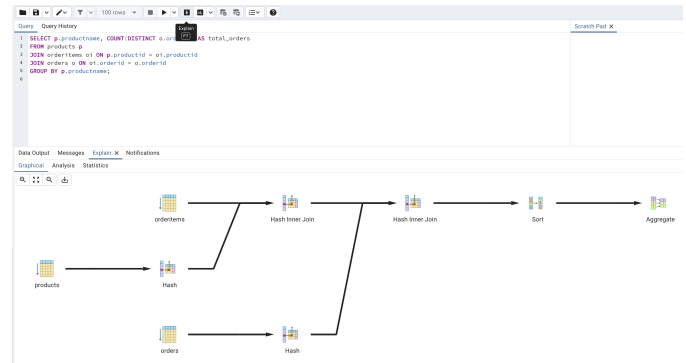


Fig. 18. Enter Caption

To optimize the performance of the three problematic queries, the plan is to create appropriate indexes on the columns involved in the join conditions, filtering criteria, and group by clauses. Specifically, creating indexes on the columns used in the WHERE clause predicates, JOIN conditions, and GROUP BY columns. This indexing strategy will facilitate faster data retrieval and query processing, reducing the need for full table scans and inefficient join algorithms. By creating indexes on the relevant columns, aim is to improve query execution times and overall database performance.

XV. WEB APPLICATION

We developed a web application using Streamlit, catering to two distinct user roles: Administrators and End users. Administrators possess the privilege to modify the database through the application, granting them control over database management tasks. Conversely, the End users are provided with access to view and analyze the data presented within the application. This segregation of user roles ensures efficient database management while facilitating seamless data visualization and analysis for the end users.



Fig. 19. User Interface

Administrator View: The Administrator has the capability to not only view data through selective querying but also possesses the authority to insert, delete, and update records within the database. This comprehensive access empowers the Administrator to effectively manage the database, ensuring smooth data operations. The access Password: **admin123**

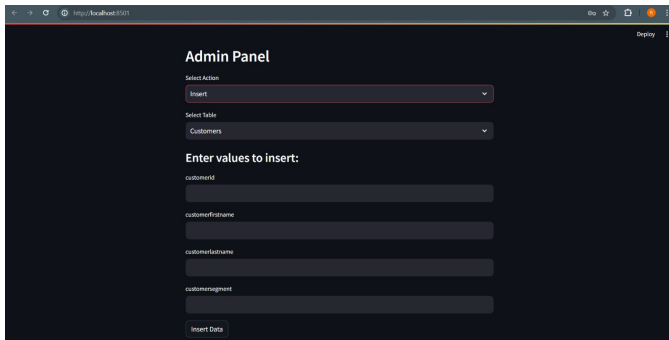


Fig. 20. Insert Operation

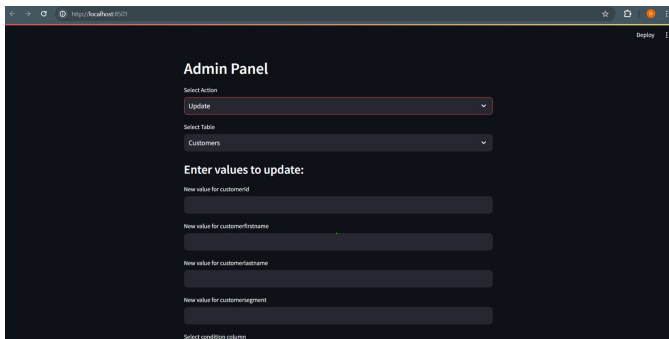


Fig. 21. Update Operation

End User View: The end user is limited to viewing data within the application. Through the selection of specific tables, the user gains access to view the data contained within those tables. This restricted functionality ensures that the end user can only retrieve and visualize information without the ability to make modifications to the database.

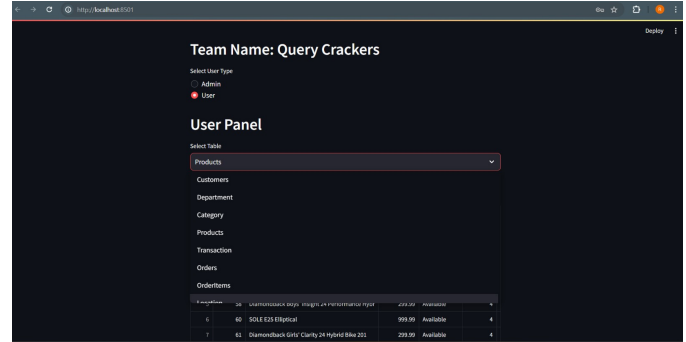


Fig. 22. Selection of Table

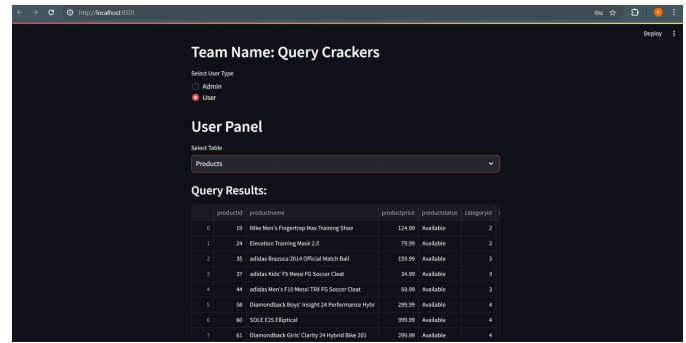


Fig. 23. Product Table Data

XVI. REFERENCES

- [1] **Dataset Link:** <https://data.mendeley.com/datasets/8gx2fvg2k6/5>
- [2] **Dataset cleaning:** <https://colab.research.google.com/drive/19PL3p0qaWpY2mYwlMSRWaiQmapbT9ZaG?usp=sharing>
- [3] <https://docs.streamlit.io/develop/tutorials/databases/postgresql>