Enhancing Business Intelligence for E-Bike Sales through a Relational Database

1st Satyam Tiwari
satyamti
Buffalo, USA

2nd Aditi Soni
asoni3
Buffalo, USA

3rd Ben Pu
wanchaop
Buffalo, USA

***Abstract*—This project focuses on improving e-bike sales strategies by transitioning from Excel files to a robust relational database. The goal is to identify high-spending customers and popular products for revenue growth and identify new potential customers based on their demographics and needs. The database offers data integrity, efficient retrieval, and scalability. It empowers departments and contributes to data-driven decision-making in the e-bike sales domain, fostering growth and competitiveness.**

I. Problem Statement

The problem at hand involves managing critical data for a company that sells e-bikes, which includes transaction records, a new customer list, customer demography data, and customer address information. To optimize operations and boost revenue, it is essential to identify high-spending customers and popular products. The challenge is to efficiently process and analyze this data to extract valuable insights. Additionally, we have a list of potential customers who might be align with our current customers in accordance with their demographics and bike-part needs once we visualize our data in form of a relational database the insights can help us recognize which of these customers can be targeted directly by the company.

Background: The background of this problem lies in the growing e-bike industry, which has seen an influx of data from transactions, customer information, and demographics. The company's need to identify high-spending customers and popular products is vital for its sustainability and growth. The manual analysis of data stored in separate Excel files has proven to be time-consuming, error-prone, and not scalable. Therefore, the need for a more robust and efficient data management solution is evident. Also, it is very hard to read buying trends and predict popular products or important customers which makes having a relational database a necessity.

Potential Contribution: By transitioning from Excel files to a relational database, this project can make a significant contribution to the e-bike sales domain. The use of a database offers several advantages, including:

1. Data Integrity: Databases ensure data accuracy and consistency, reducing the risk of errors often associated with Excel files. This, in turn, builds trust in the data, which is crucial for decision-making.

2. Efficient Data Retrieval: Databases allow for efficient querying and retrieval of information. This speed and efficiency can enable timely identification of high-spending customers and popular products.

3. Scalability: As the company expands, a database can easily accommodate the increasing data volume. This scalability is crucial for a growing e-bike business.

4. Data Analysis: Databases support the use of SQL queries, data warehousing, and data analysis tools, making it easier to extract valuable insights from the data. These insights can be used to develop targeted marketing strategies and optimize inventory management.

5. Historical Data: Databases can efficiently store and manage historical data, enabling trend analysis and long-term decision-making.

In conclusion, transitioning to a relational database is not merely a technical upgrade but a strategic move that can significantly contribute to the success and growth of the e-bike sales company. It empowers the company to make data-driven decisions, identify high-value customers and products, and ultimately increase revenue while ensuring data integrity and scalability.

II. TARGET USER

In a real-life scenario, the database for the e-bike sales company would be used and administered by the following individuals and departments:

1. Sales and Marketing Teams: The sales and marketing teams will extensively use the database to identify high-spending customers, popular products, and purchase trends. This information will help them create targeted marketing campaigns, loyalty programs, and promotional offers. They will run queries and reports to understand customer preferences and design strategies accordingly.

2. Data Scientists and Data Engineers: Data scientists and data engineers may be employed to work with the database, developing advanced machine learning models to predict cus-

tomer behavior, optimize pricing, and enhance the overall customer experience.

Additionally, Customer Service Representatives, Inventory Management Team, Business Analysts, IT Administrators, Management and Executives can also make use of the relational databases.

In this real-life scenario, a well-designed database acts as a central repository of information, accessible to various departments and personnel with different roles and responsibilities. It facilitates data-driven decision-making, streamlines operations, and contributes to the company's overall success. Database administration is a crucial responsibility to ensure the integrity, security, and efficiency of data management within the organization.

III. Tables

A. 1. Customer Demographics

The customer demographics table is a table which includes all the demographic information of the current customers of the company it has the followings columns.

Customer id .
*Primary Key
*FK to Customer Address and Transaction Table
*Datatype – number
*Cannot be null
*No default value
*Uniquely identifies every customer.
*Delete Cascade ON

First name
*Datatype- varchar
*Cannot be null
*No default value
*Stores the first name of the customers

Last name
*Datatype- varchar
*Cannot be null
*No default value
*Stores the last name of the customers

Gender
*Datatype- varchar
*can be null
*No default value
*Stores the gender information of the customers

Past 3 years purchases
*Datatype- number
*can be null
*No default value
*Stores the number of purchases the customer made in the last 3 years

DOB
*Datatype- date
*can be null
*No default value
*Stores the birthday information of the customers

Job title
*Datatype- varchar
* can be null
* No default value
*Stores the job title of the customers

Job industry category
* Datatype- varchar
* can be null
* No default value
* Stores the industry name in which the customers work.

Wealth segment
* Datatype- varchar
*can be null
* No default value
* Stores the wealth segment to which the customer belongs.

Deceased indicator
* Datatype- varchar
* cannot be null
* Default value - N
* Stores the information on whether a customer is dead
.

Owns car
* Datatype- varchar
* can be null.
* No default value.
* Stores the information on whether the customer owns a car.

For the Customer Demographics table
- The primary key is 'customer id', and it uniquely identifies each row.
- The attributes (non-prime) like 'first name', 'last name', 'gender', etc., are fully functionally dependent on the primary key.

Hence, we can say that the table confirms to the norms of BCNF.

The customer demographics table is related to both the customer address table and the transactions table via the Customer id column.

B. 2. Customer Addresses

The customer address table is a table which includes all known addresses information of the current customers of the company it has the followings columns.

Customer id
* Composite PK / Candidate Key
* FK for Customer Demography
* Datatype – number
* Cannot be null.
* No default value.
* Uniquely identifies every customer.
*Delete Cascade ON

Postcode
*Composite PK / Candidate Key
* Datatype – number
* Cannot be null.
* No default value.
* Stores the postal code of the addresses.

State
* Datatype – varchar
* Cannot be null
* No default value.
* Stores the state names of the addresses.

Address
* Datatype – varchar
* Cannot be null.
* No default value.
* Stores the addresses.

Country
* Datatype – varchar
* Cannot be null.
* No default value.
* Stores the country of the address.

Property valuation
* Datatype – varchar
* Cannot be null.
* No default value.
* Stores the values of the property ranked based on range.

For the Customer Address table
- The primary key is ‘customer id‘ + postcode is part of the primary key, it ensures uniqueness.
- A single customer can have multiple addresses so customer id alone will not be able uniquely identify every row, however once we add postcode to it our issue is resolved.
- All attributes are dependent on the primary key.

Hence, we can say that the table confirms to the norms of BCNF.

The customer address table is related to the customer demographics table via the Customer id column.

C. 3. Transactions

The transaction table is a table which includes all the transaction information made by the current customers of the company it has the followings columns.

transaction id
*Primary Key
*Datatype – number
*Cannot be null.
*No default value.
*Uniquely identifies every transaction.

product id
*Foreign Key from the product table
*Datatype – number
*Cannot be null.
*No default value.
*Identifies the product id for which the transaction happened .
*Delete Cascade OFF

customer id
*Foreign Key from the customer table
*Datatype – number
*Cannot be null.
*No default value.
*Identifies the customer who made the transaction.
*Delete Cascade ON

transaction date
*Datatype- date
*cannot be null.
*No default value.
*Stores the date of the transaction.

online order
*Datatype- varchar
*cannot be null.
*No default value.
*Tells whether transaction was online or not

order status
*Datatype- varchar
*cannot be null.
*No default value.
*Stores the current status of the order .

For the Transaction table
- The primary key is ‘transaction id‘, which uniquely identifies each row.
- Non-prime attributes like ‘product id‘, ‘transaction date‘, etc., are fully functionally dependent on the primary key.

Hence, we can say that the table confirms to the norms of BCNF.

The transactions table is related to the customer demographics table via the Customer id column and to the products table via the product id column.

D. 4. Product Table

This table contains information about all the products that are available in the company it has the followings columns..

Product id
*Primary Key
*FK to Transaction Table
*Datatype – number
*Cannot be null.
*No default value.
*Uniquely identifies every transaction.
*Delete Cascade OFF

Product name
*Datatype – varchar
*Cannot be null.
*No default value.
*Name of the products.

Brand
*Datatype – varchar
*Can be null.
*No default value.
*Name of the brand the product belongs to.

Type/category
*Datatype – varchar
*Can be null.
*No default value.
*Category to which the product belongs to.

List price
*Datatype – number
*Cannot be null.
*No default value.
*Listing price of the product.

Standard cost
*Datatype – number
*Cannot be null.
*No default value.
*Standard cost price of the product in the market.

Product availability
*Datatype – varchar
*Cannot be null.
*Default Value – In stock.
*Whether or not a product is currently available.

Product class
*Datatype – varchar
*Can be null.
*No default value.
*Product categorized by size.

For the Product table
- The primary key is ‘product id‘, which uniquely identifies each row.
- Non-prime attributes like ‘brand‘, ‘category‘,’product name’ etc., are fully functionally dependent on the primary key.

Hence, we can say that the table confirms to the norms of BCNF.

The Product table is related to the transactions table via the product id column.

E. 5. Potential Customer List

This table contains the demographic information on a list of people who might become potential customers of the company, as of now this is a stand-alone independent table but once we have the relational database setup we can perform analytics and use visualization to identify new customers to target, it has the followings columns.

Potential customer id
*Primary Key
*Datatype – number
*Cannot be null
*No default value
*Uniquely identifies every potential customer.

First name
*Datatype- varchar
*Cannot be null
*No default value
*Stores the first name of the potential customers

Last name
*Datatype- varchar
*Cannot be null
*No default value
*Stores the last name of the potential customers

Gender
*Datatype- varchar
*can be null
*No default value
*Stores the gender information of the potential customers

Past 3 years purchases
*Datatype- number
*can be null
*No default value
*Stores the number of bike related purchases the customer

made in the last 3 years

DOB
*Datatype- date
*can be null
*No default value
*Stores the birthday information of the potential customers

Job title
*Datatype- varchar
*can be null
*No default value
*Stores the job title of the potential customers

Job industry category
*Datatype- varchar
*can be null
*No default value
*Stores the industry name in which the potential customers work

Wealth segment
*Datatype- varchar
*can be null
*No default value
*Stores the wealth segment to which the potential customer belongs.

Deceased indicator
*Datatype- varchar
*cannot be null
*Default value - N
*Stores the information on whether a potential customer is dead.

Owns car
*Datatype- varchar
*can be null.
*No default value.
*Stores the information on whether the potential customer owns a car.

Postcode
*Datatype – number
*Cannot be null.
*No default value.
*Stores the postal code of the addresses.

State
*Datatype – varchar
*Cannot be null.
*No default value.
*Stores the state names of the addresses.

Address
*Datatype – varchar
*Cannot be null.
*No default value.
*Stores the addresses.

Country
*Datatype – varchar
*Cannot be null.
*No default value.
*Stores the country of the address.

Property valuation
*Datatype – varchar
*Cannot be null.
*No default value.
*Stores the values of the property ranked based on range.

For the potential customer table -
- The primary key is ‘product customer id‘, and it uniquely identifies each row.
- Non-prime attributes like ‘first name‘, ‘last name‘, ‘gender‘, etc., are fully functionally dependent on the primary key.

Hence, we can say that the table confirms to the norms of BCNF.

IV. Entity-Relationship Diagram

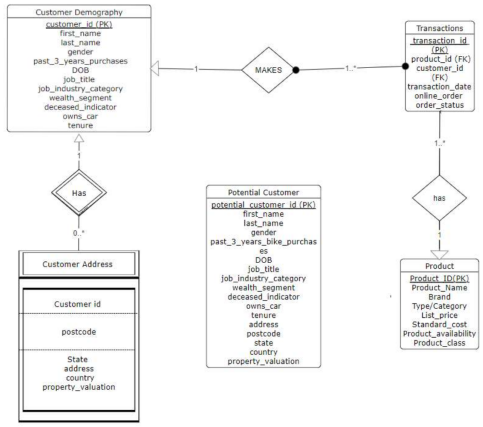

Fig. 1. Entity-Relationship Diagram.

ER Diagram Explanation

Strong Entity in the ER Diagram:

1.**Product**:

· Entity Type: The ”Product” entity represents E-bike products and includes attributes like product name, brand, description, price, etc.

· Primary Key: The primary key of this entity is ‘product_id‘, which ensures the uniqueness of each product in the table.

· Relationship: The ”Product” entity is related to the ”Transaction” entity in a one-to-many relationship. This relationship

allows multiple transactions to be associated with a single product, reflecting sales and purchase data.

2.**Transaction**:

· Entity Type: The ”Transaction” entity records details about each transaction, including transaction date, order status, product information, and more.

· Primary Key: The primary key of this entity is ‘transaction_id‘, serving as a unique identifier for each transaction record.

· Foreign Key: The ”Transaction” entity contains foreign keys, such as ‘product_id‘ and ‘customer_id‘, establishing relationships with the ”Product” and ”Customer Demography” entities.

· Relationship: It is associated with both the ”Product” and ”Customer Demography” entities in one-to-many relationships. This allows multiple transactions to be associated with one product and one customer, reflecting the buying history of customers.

3.**Customer Demography:**

· Entity Type: The ”Customer Demography” entity contains demographic information about customers, including their names, gender, job title, and more.

· Primary Key: The primary key is ‘customer_id‘, ensuring that each customer is uniquely identified in the database.

· Relationship: It is related to the ”Transaction” entity in a one-to-many relationship, signifying that a customer can have multiple transactions, tracking their purchase history.

Weak Entity in the ER Diagram:

1.**Customer Addresses:**

· Entity Type: The ”Customer Addresses” entity represents addresses associated with customers in the database. It includes attributes like address, postcode, state, country, and property valuation.

· Composite Primary Key: The unique identification of records in this entity relies on a composite primary key, which is formed by combining two attributes: customer_id and postcode. This combination ensures the uniqueness of each address associated with a specific customer in a particular postal code.

· Weak Entity: ”Customer Addresses” is considered a weak entity because it cannot be uniquely identified by its attributes alone. It relies on the related strong entity, ”Customer Demography,” to provide identity through the composite primary key.

· Relationship: ”Customer Addresses” is related to the ”Customer Demography” entity through a one-to-many relationship. This relationship indicates that one customer can have multiple addresses, reflecting the possibility of customers having multiple places of residence.

Independent Entity in the ER Diagram:

Potential Customer:

· Entity Type: The ”Potential Customer” entity represents a separate group of individuals or entities that are not yet associated with any other entities in the ER diagram.

· Autonomous Entity: ”Potential Customer” is an independent or autonomous entity because it does not have direct relationships or dependencies on other entities in the diagram.

· Use Case: This entity may represent individuals who have expressed interest in the company’s products but have not yet become actual customers. It can serve as a way to track and manage potential leads or prospects.

Relationships in the ER Diagram:

1.Customer Demography and Transaction Table Relationship:

Customer Demography Entity: This entity represents customer demographic information. It has a primary key, ‘customer_id‘, which uniquely identifies each customer.

Transaction Table Entity: The transaction table has a primary key, ‘transaction_id‘, and foreign keys, ‘product_id‘ and ‘customer_id‘. This indicates a one-to-many relationship between customers and their transactions, where one customer can have multiple transactions. The diamond shape between these entities is labeled ”Has” or ”Is Associated With” to signify this one-to-many relationship.

2.Product Table and Transaction Table Relationship:

Product Entity: The product table represents various E-bike products. It has a primary key, ‘product_id‘, which uniquely identifies each product.

Transaction Table Entity: The transaction table includes a foreign key, ‘product_id‘, linking it to the product table. This relationship is also one-to-many, indicating that one product can be associated with multiple transactions. The diamond shape in this relationship is labeled ”Has” or ”Is Associated With.”

This ER diagram visually represents the relationships between these entities, helping you understand how they interact and are connected in the database. It highlights the one-to-many relationships between customers and transactions and products and transactions, as well as the composite primary key in the ”Customer Addresses” entity, which establishes its relationship with the ”Customer Demography” entity.

This ER diagram is a valuable tool for understanding the structure and connections within your database, and it will be a useful reference for your database management and optimization efforts

V. 1. Do you specifically run into any problems while handling the larger dataset?Did you try to adopt some indexing concepts to resolve this? Briefly describe the questions you faced and how you solved them.

1) Challenges with the larger Dataset::

1) Slow Query Performance: Retrieving data from larger tables resulted in slow query performance, especially when filtering or sorting.
2) Inefficient Joins: Joining multiple large tables was resource-intensive and impacted query speed.
3) Indexing Decisions: Deciding which columns to index and which indexes to create was challenging.

2) Adopting Indexing:: We have a transaction table with over 40000 rows of data and to run queries that involved both Transaction and Customer Demographics table was a hassle while trying to get data between certain dates

We created an index on the transaction table in the transaction date column to make our filtering operations faster

CREATE INDEX ”index id” ON Transactions (”Transaction date”);

VI. Testing DB with Queries

Now that our Database setup, we will test it with 10 queries 2- Insert, 2-Delete, 2-Update and 4-Select

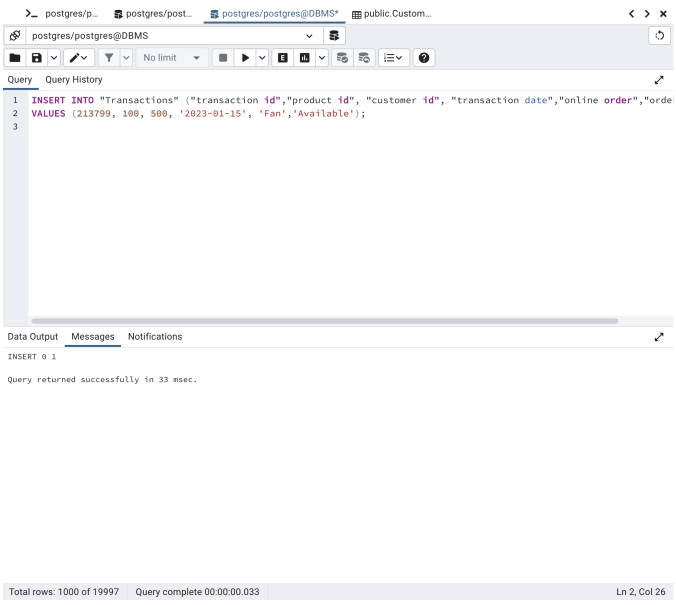

Fig. 2. Insert Query 1 .

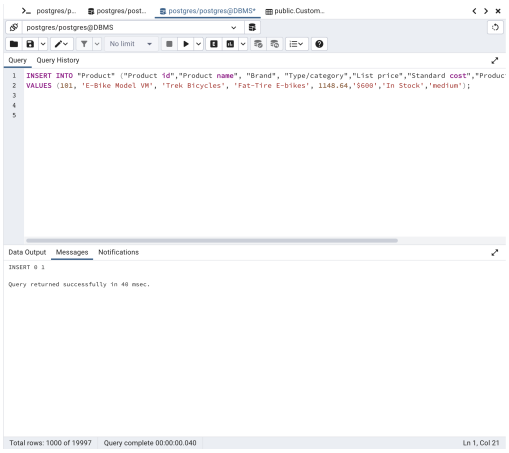

Fig. 3. Insert Query 2.

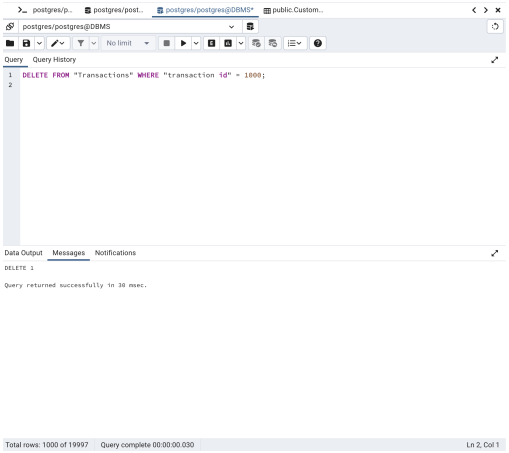

Fig. 4. Delete Query 1 .

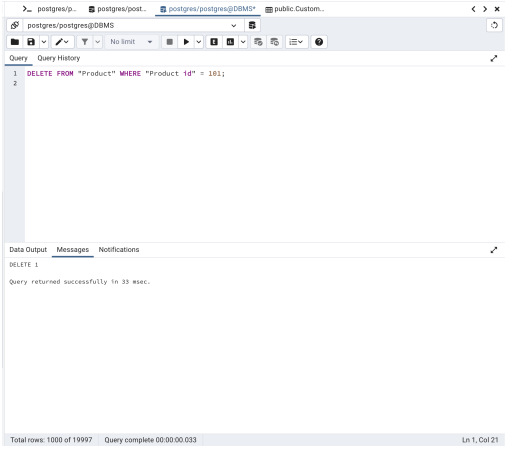

Fig. 5. Delete Query 2.

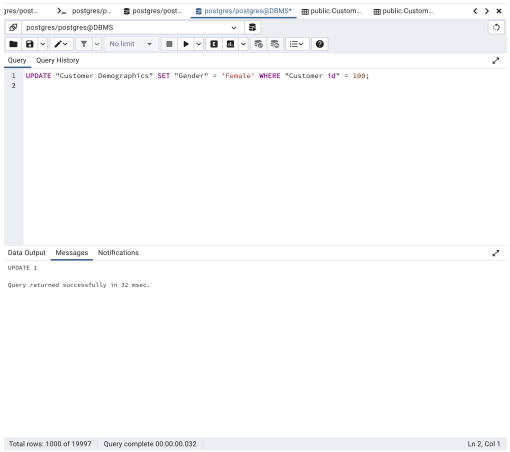

Fig. 6. Update Query 1 .

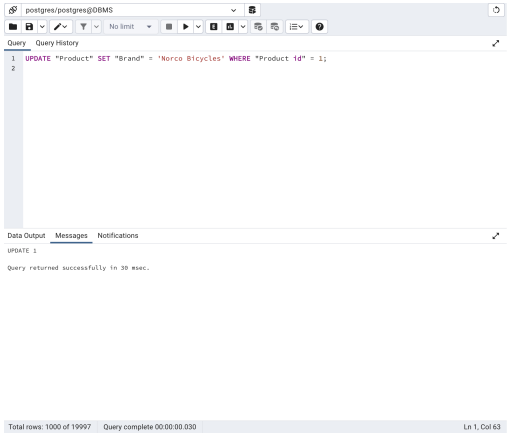

Fig. 7. Update Query 2.

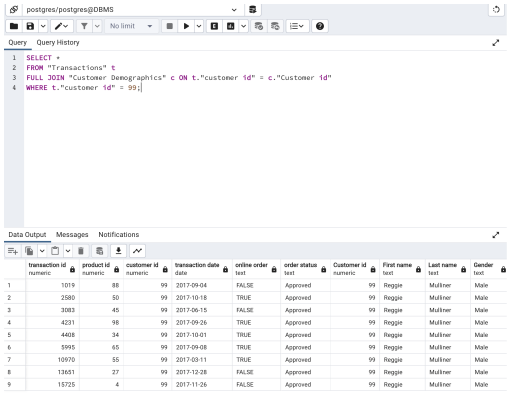

Fig. 8. Select Query 1 .

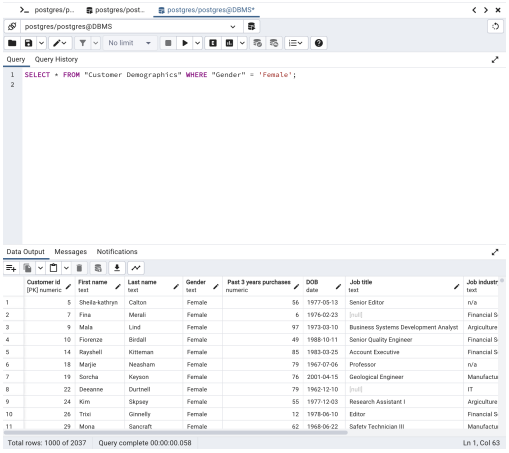

Fig. 9. Select Query 2.

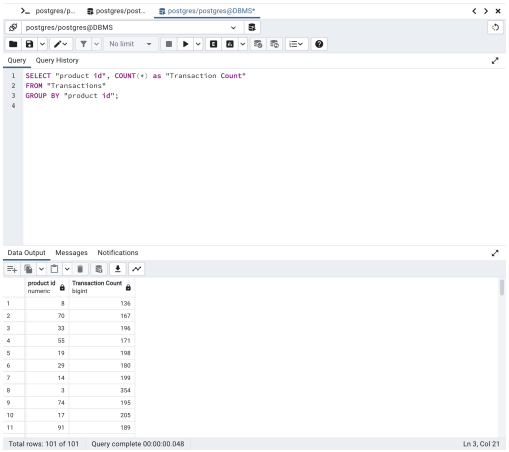

Fig. 10. Select Query 3.

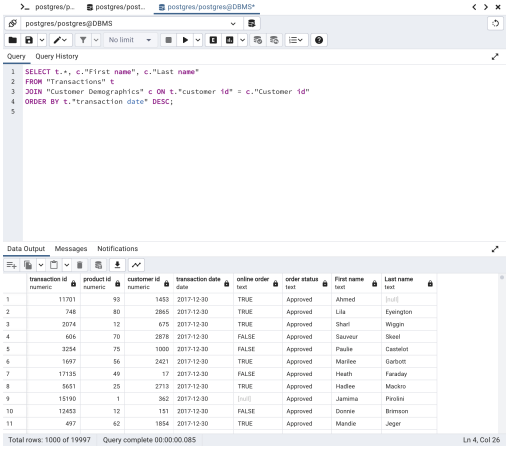

Fig. 11. Select Query 4 .

VII. 3. Query execution analysis: identify three problematic queries (show their cost), where the performance can be improved. Provide a detailed execution plan (you may use EXPLAIN in PostgreSQL) on how you plan to improve these queries.

Problematic Query 1

SELECT * FROM ”Transactions” t JOIN ”Customer Demographics” c ON ”customer id” = ”Customer id” WHERE ”transaction date” ¿= ’2000-01-01’ AND ”transaction date” ¡= ’2023-02-01’ AND ”customer id” = 1000 ;

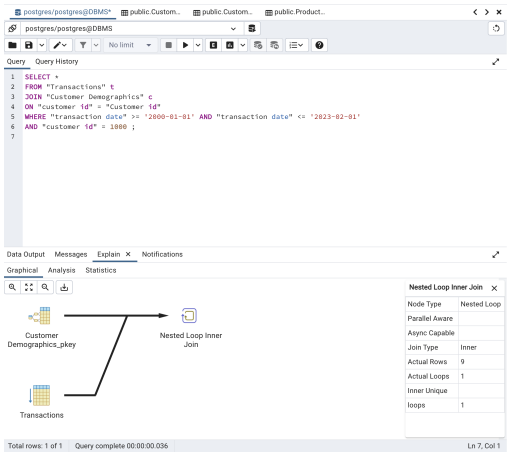

Fig. 12. Problematic Query 1 .

To make this query perform better we need to index both the tables Transactions and Customer Demographics

CREATE INDEX idxcustomeridtransactions ON ”Transactions” (”customer id”);

CREATE INDEX idxcustomeriddemographics ON ”Customer Demographics” (”Customer id”);

CREATE INDEX idxtransactiondate ON ”Transactions” (”transaction date”)

Problematic Query 2

SELECT ”Customer id”, AVG(”Customer Demographics”.”Past 3 years purchases”) AS ”Average Purchase” FROM ”Customer Demographics” WHERE ”Owns car” = ’Yes’ GROUP BY ”Customer id”;

Since there is no index on the ”Past 3 years purchases” column, the query performs a slow sequential scan on the entire ”Customer Demographics” table.

The GROUP BY operation could be resource-intensive, since there are a large number of distinct customers.

CREATE INDEX idxPTYPurchases ON ”Customer Demographics” (”Past 3 years purchases”);

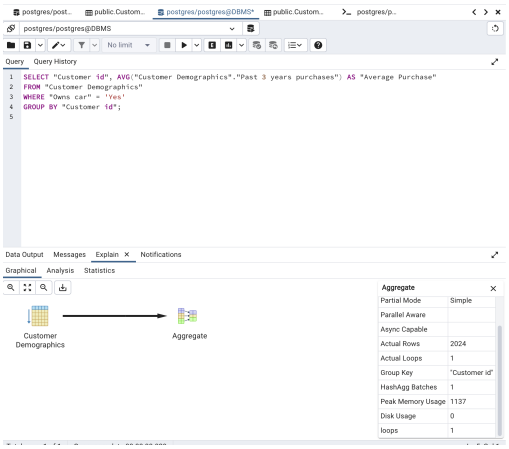

Fig. 13. Problematic Query 2.

Problematic Query 3

SELECT * FROM ”Customer Demographics” c JOIN ”Transactions” t ON c.”Customer id” = t.”customer id” WHERE t.”online order” = ’TRUE’;

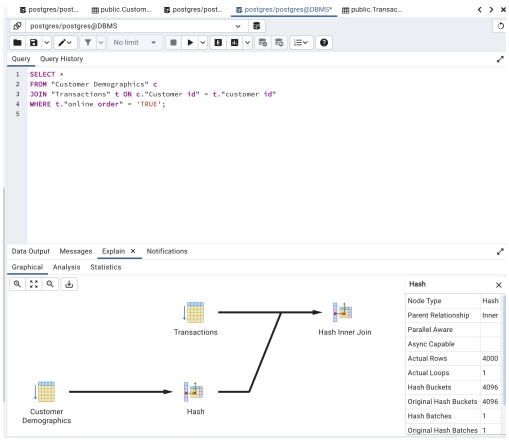

Fig. 14. Problematic Query 3.

The JOIN operation without proper indexing is resource-intensive, especially if both tables have a large number of rows.

CREATE INDEX idxamount ON ”Transactions” (”online order”);

REFERENCES

1) Microsoft Corporation. (n.d.). SQL Server Documentation. https://docs.microsoft.com/en-us/sql/https://docs.microsoft.com/en-us/sql/

2) Oracle Corporation. (n.d.). Oracle Database Documentation. https://docs.oracle.com/en/database/

3) PostgreSQL Global Development Group. (n.d.). PostgreSQL Documentation. https://www.postgresql.org/docs/

4) Redgate. (n.d.). SQL Server Central. https://www.sqlservercentral.com/

5) Smartdraw https://app.smartdraw.com/