CSE 560
Data Models and Query Language Semester Project Milestone 1

"Inventory Management and Online Order Tracking"

Group Name: Honeycomb

Eva Pradhan
(evapradh@buffalo.edu)
Harshitha Damineni
(hdaminen@buffalo.edu)
Venkata Datta Viswanadha Vishnubhotla
(vvishnub@buffalo.edu)

Need for Inventory management database implementation

- Businesses need inventory and online order tracking databases to manage their products, orders, and shipments efficiently.
- Manual tracking can be time-consuming and prone to errors, leading to inaccuracies and delays in decision-making.
- An efficient database system should be user-friendly, scalable, and secure, with robust reporting and analytics features to monitor inventory levels and identify trends.
- Real-time updates and notifications for customers improve their satisfaction and loyalty, leading to increased sales for the business.

TARGET USER

- Retail businesses with physical storefronts and online shops that sell products and manage inventory.
- Warehouses and distribution centers that store and move inventory for multiple businesses.
- Manufacturing companies that produce and manage their own inventory of raw materials and finished goods.
- Service-based businesses that maintain an inventory of supplies and equipment for their operations, such as restaurants or construction companies.

DATA DESCRIPTION

- The e-commerce market's dataset contains data on orders, consumers, sellers, products, and reviews. The e-commerce platform Olist, which offers marketplace services to small and medium-sized firms, is the source of the dataset. The dataset is composed of multiple tables, including:
-
- Seller dataset: This file includes details about the market place vendors, including seller ID, seller name, and seller
zip code.
-
- Customers dataset: This file includes details about the customers who placed the orders, including their name, city, state, and zip code, as well as a special customer ID.
- Order items dataset: Each order's items are listed in this file together with details such as the order ID, product ID, seller ID, price, freight value, and quantity.
-
- Geolocation dataset: The latitude and longitude of the buyers' and sellers' locations are included in this file along with other geolocation data.
-
- Order payments dataset: This file includes details on
payments made for each order, including the value, kind, and installments of payments.

DATA DESCRIPTION

Orders dataset: The order ID, customer ID, order status, purchase timestamp, and anticipated delivery date are just a few of the details regarding the orders that are included in this file.

Products dataset: The product ID, product category name, product name, and product description are among the details about the goods offered on the marketplace that are included in this file.

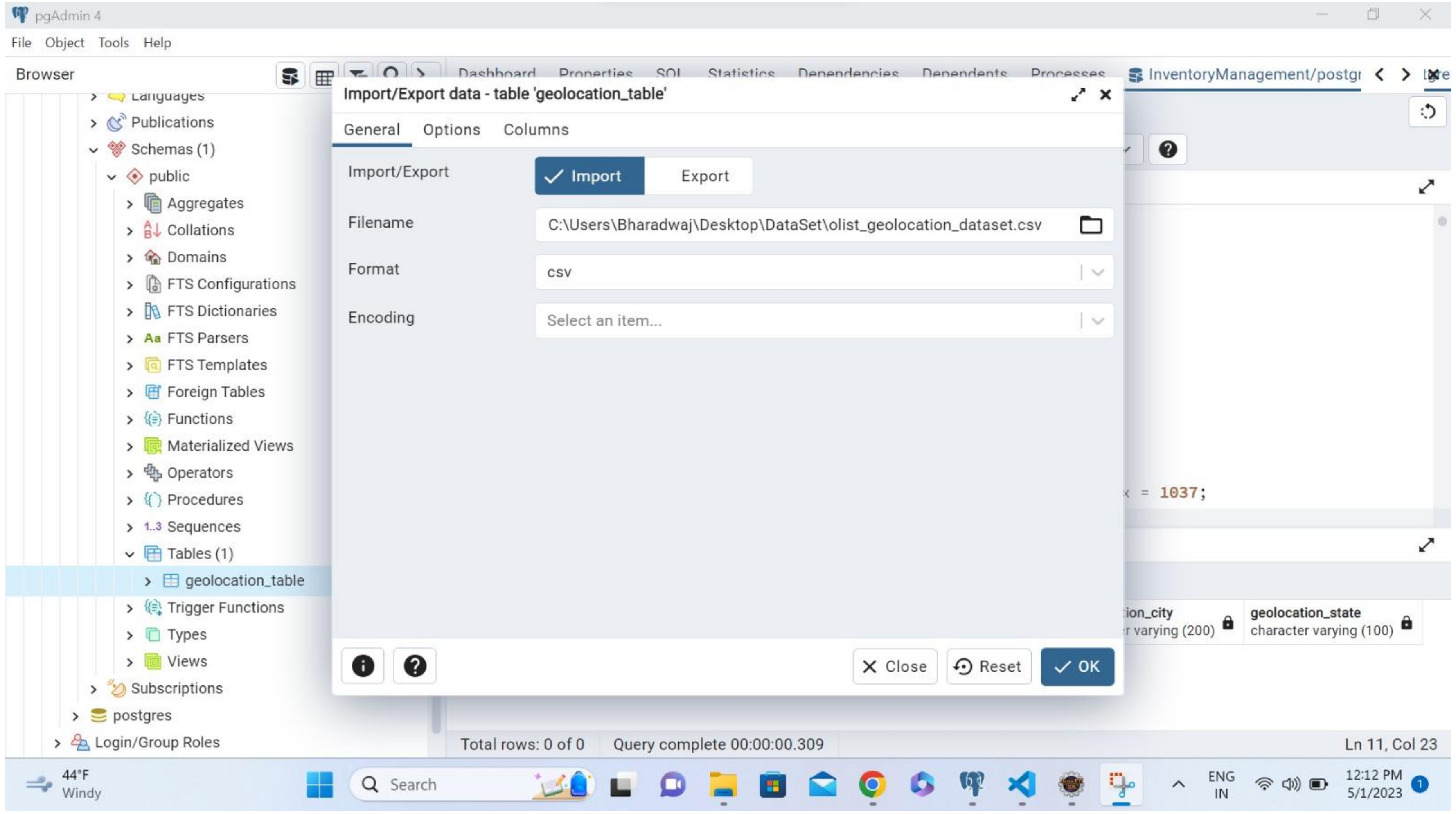
Order reviews dataset: The order ID, review score, review comment title, and review comment message are among the details regarding the reviews that consumers have written for the orders that are included in this file.

Order items dataset: Each item of an order is described in detail in the order Items dataset. For each item, the dataset specifically offers details such as order id, order item id, product id, seller id, shipping limit date, etc.

Product Seller dataset: This table maps the product id to corresponding seller id.

In conclusion, the E-Commerce
Public Dataset by Olist offers a wealth of information for studying the e-commerce business, including consumer behaviour, market trends, and seller performance

Importing CSV files to PostgreSQL



Import/Export data - table 'geolocation_table'

General | Options | Columns

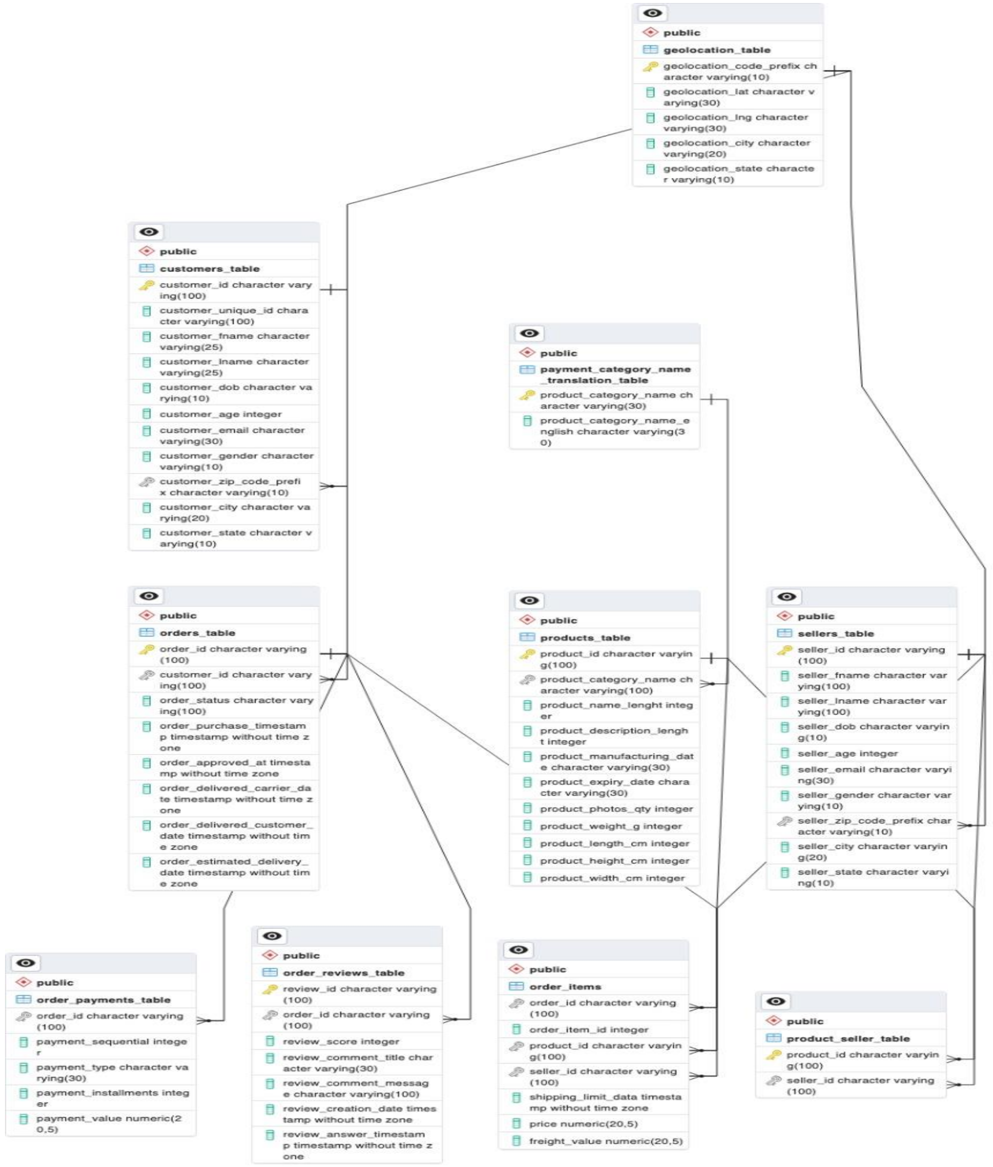
Import/Export: [✓ Import] Export

Filename: C:\Users\Bharadwaj\Desktop\DataSet\olist_geolocation_dataset.csv

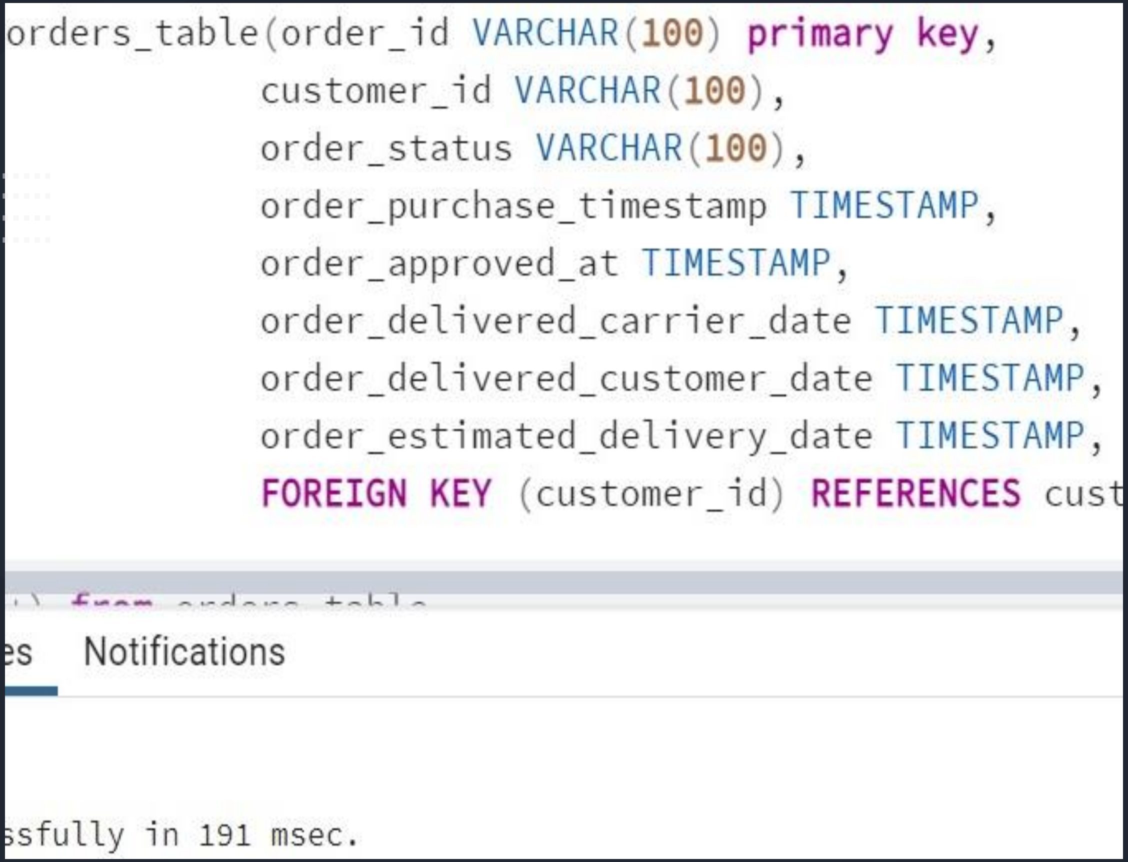
Format: csv

Encoding: Select an item...

ER DIAGRAM


public
geolocation_table
geolocation_code_prefix character varying(10)
geolocation_lat character varying(30)
geolocation_lng character varying(30)
geolocation_city character varying(20)
geolocation_state character varying(10)
public
customers_table
customer_id character varying(100)
customer_unique_id character varying(100)
customer_fname character varying(25)
customer_lname character varying(25)
customer_dob character varying(10)
customer_age integer
customer_email character varying(30)
customer_gender character varying(10)
customer_zip_code_prefix character varying(10)
customer_city character varying(20)
customer_state character varying(10)
public
payment_category_name_translation_table
product_category_name character varying(30)
product_category_name_english character varying(30)
public
orders_table
order_id character varying (100)
customer_id character varying(100)
order_status character varying(100)
order_purchase_timestamp timestamp without time zone
order_approved_at timestamp without time zone
order_delivered_carrier_date timestamp without time zone
order_delivered_customer_date timestamp without time zone
order_estimated_delivery_date timestamp without time zone
public
products_table
product_id character varying(100)
product_category_name character varying(100)
product_name_lenght integer
product_description_lenght integer
product_manufacturing_date character varying(30)
product_expiry_date character varying(30)
product_photos_qty integer
product_weight_g integer
product_length_cm integer
product_height_cm integer
product_width_cm integer
public
sellers_table
seller_id character varying (100)
seller_fname character varying(100)
seller_lname character varying(100)
seller_dob character varying(10)
seller_age integer
seller_email character varying(30)
seller_gender character varying(10)
seller_zip_code_prefix character varying(10)
seller_city character varying(20)
seller_state character varying(10)
public
order_payments_table
order_id character varying (100)
payment_sequential integer
payment_type character varying(30)
payment_installments integer
payment_value numeric(20,5)
public
order_reviews_table
review_id character varying (100)
order_id character varying (100)
review_score integer
review_comment_title character varying(30)
review_comment_message character varying(100)
review_creation_date timestamp without time zone
review_answer_timestamp timestamp without time zone
public
order_items
order_id character varying (100)
order_item_id integer
product_id character varying(100)
seller_id character varying (100)
shipping_limit_data timestamp without time zone
price numeric(20,5)
freight_value numeric(20,5)
public
product_seller_table
product_id character varying(100)
seller_id character varying (100)

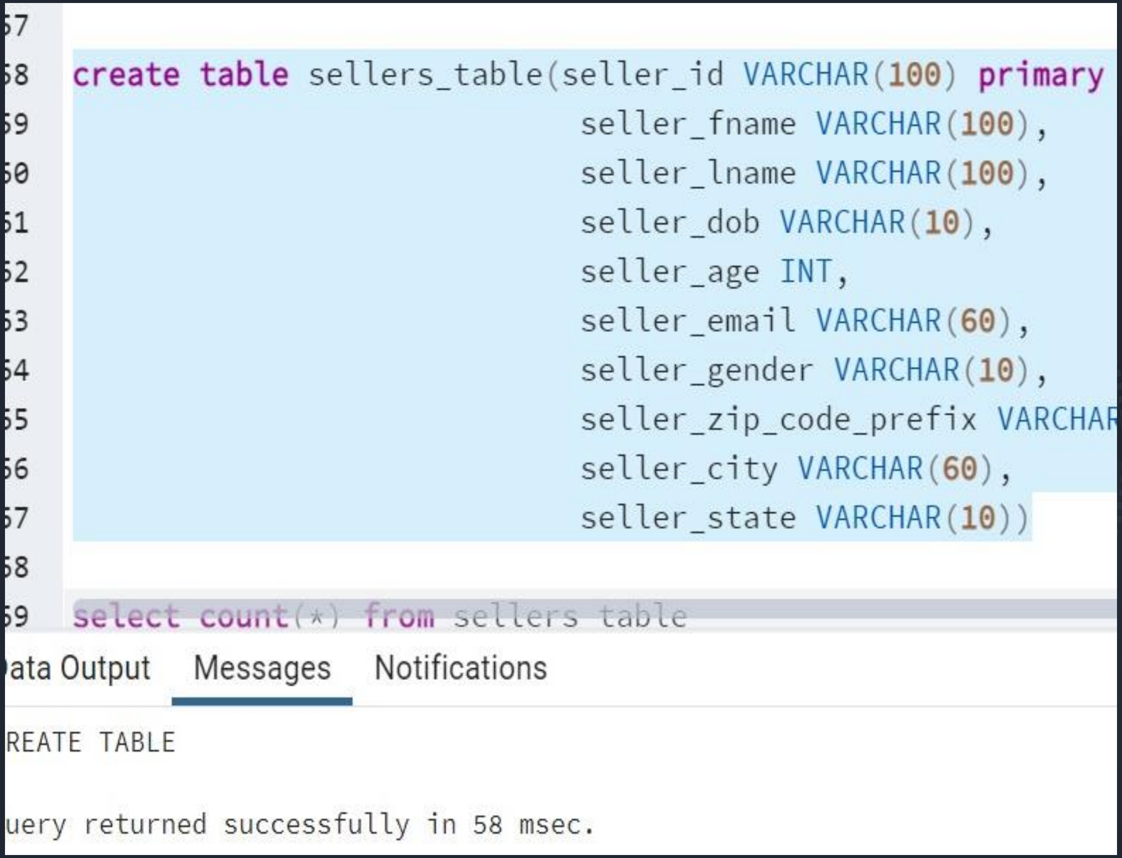
CREATE QUERY



```
orders_table(order_id VARCHAR(100) primary key,
             customer_id VARCHAR(100),
             order_status VARCHAR(100),
             order_purchase_timestamp TIMESTAMP,
             order_approved_at TIMESTAMP,
             order_delivered_carrier_date TIMESTAMP,
             order_delivered_customer_date TIMESTAMP,
             order_estimated_delivery_date TIMESTAMP,
             FOREIGN KEY (customer_id) REFERENCES cust
```

es Notifications

```
ssfully in 191 msec.
```

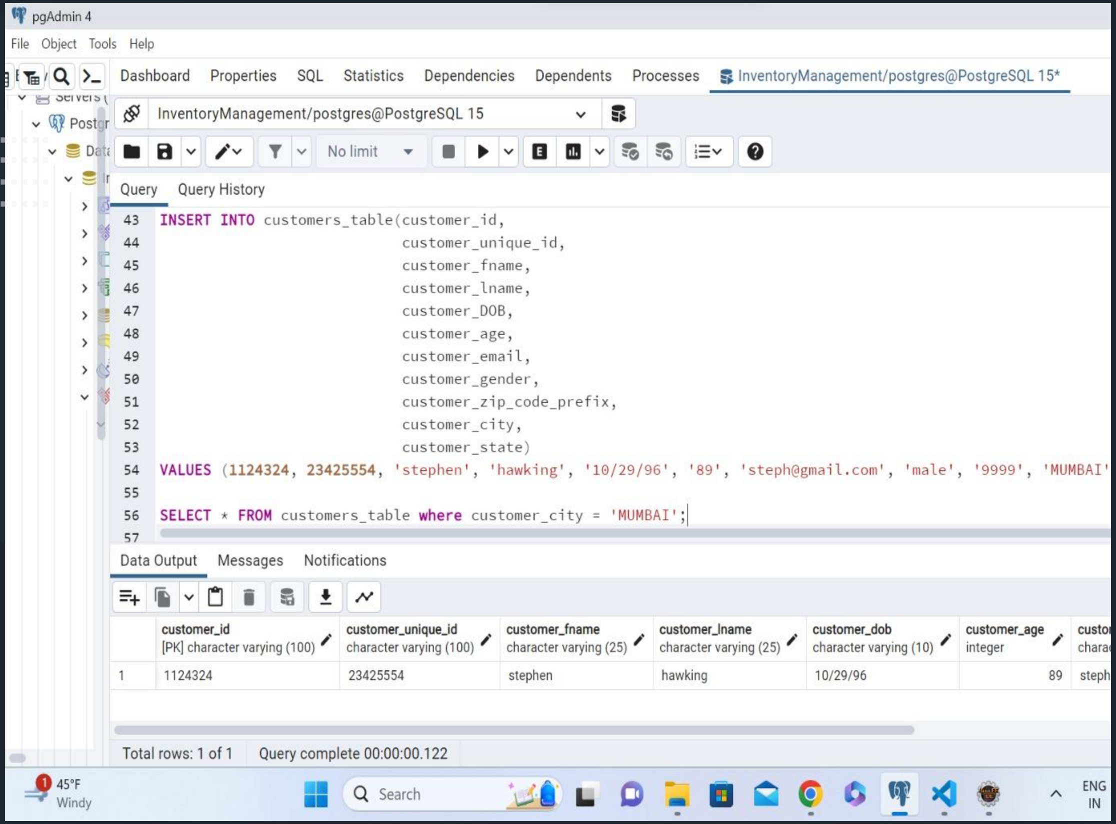


```
create table sellers_table(seller_id VARCHAR(100) primary
                           seller_fname VARCHAR(100),
                           seller_lname VARCHAR(100),
                           seller_dob VARCHAR(10),
                           seller_age INT,
                           seller_email VARCHAR(60),
                           seller_gender VARCHAR(10),
                           seller_zip_code_prefix VARCHAR
                           seller_city VARCHAR(60),
                           seller_state VARCHAR(10))

select count(*) from sellers table
```

ata Output Messages Notifications

```
REATE TABLE

uery returned successfully in 58 msec.
```

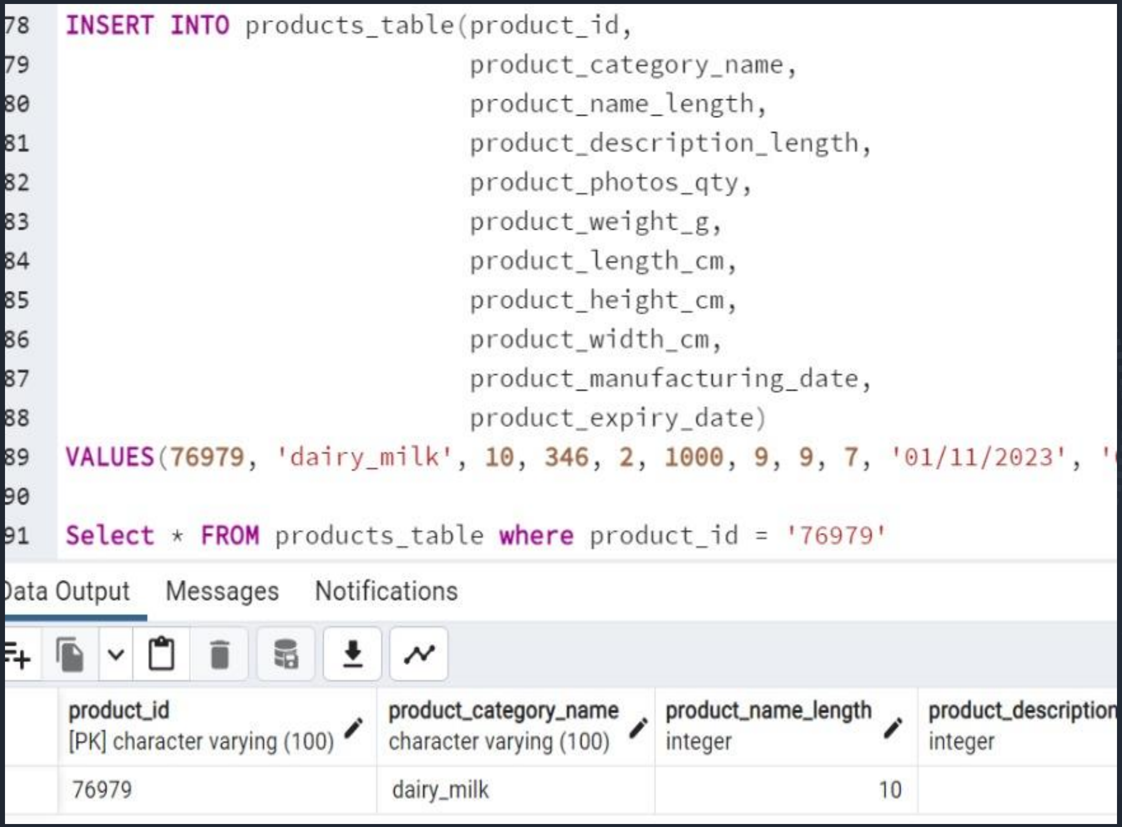
INSERT QUERY


pgAdmin 4
File Object Tools Help
Dashboard Properties SQL Statistics Dependencies Dependents Processes InventoryManagement/postgres@PostgreSQL 15*
InventoryManagement/postgres@PostgreSQL 15
No limit
Query Query History
Data Output Messages Notifications
45°F Windy
Search
ENG IN

```
INSERT INTO customers_table(customer_id,
                            customer_unique_id,
                            customer_fname,
                            customer_lname,
                            customer_DOB,
                            customer_age,
                            customer_email,
                            customer_gender,
                            customer_zip_code_prefix,
                            customer_city,
                            customer_state)
VALUES (1124324, 23425554, 'stephen', 'hawking', '10/29/96', '89', 'steph@gmail.com', 'male', '9999', 'MUMBAI'

SELECT * FROM customers_table where customer_city = 'MUMBAI';
```

	customer_id [PK] character varying (100)	customer_unique_id character varying (100)	customer_fname character varying (25)	customer_lname character varying (25)	customer_dob character varying (10)	customer_age integer	custo chara
1	1124324	23425554	stephen	hawking	10/29/96	89	steph

Total rows: 1 of 1   Query complete 00:00:00.122

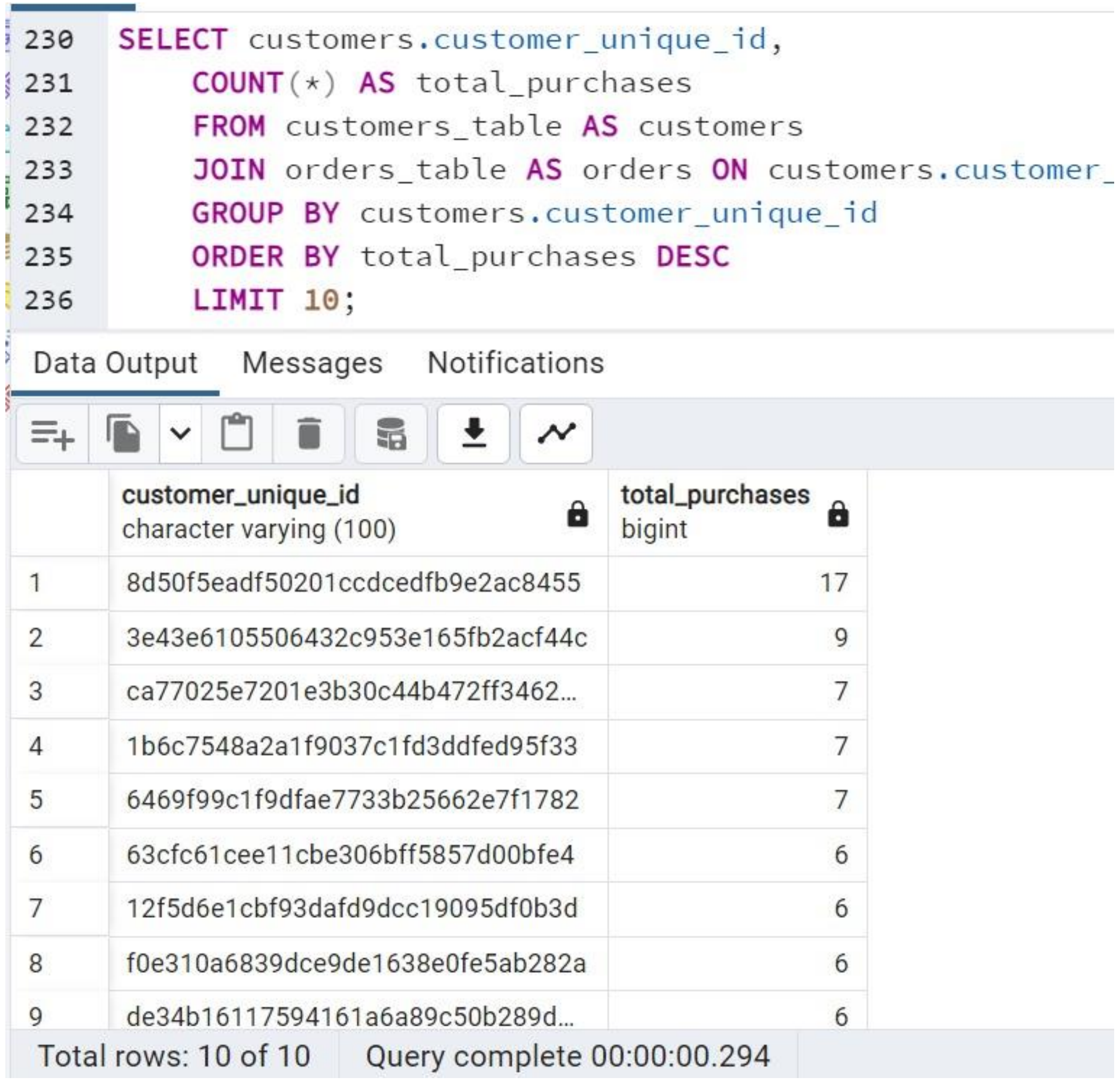


```
INSERT INTO products_table(product_id,
                           product_category_name,
                           product_name_length,
                           product_description_length,
                           product_photos_qty,
                           product_weight_g,
                           product_length_cm,
                           product_height_cm,
                           product_width_cm,
                           product_manufacturing_date,
                           product_expiry_date)
VALUES(76979, 'dairy_milk', 10, 346, 2, 1000, 9, 9, 7, '01/11/2023', '

Select * FROM products_table where product_id = '76979'
```

product_id [PK] character varying (100)	product_category_name character varying (100)	product_name_length integer	product_description integer
76979	dairy_milk	10	

SELECT QUERY

- Query to find the top 10 customers who have made the highest number of purchases:



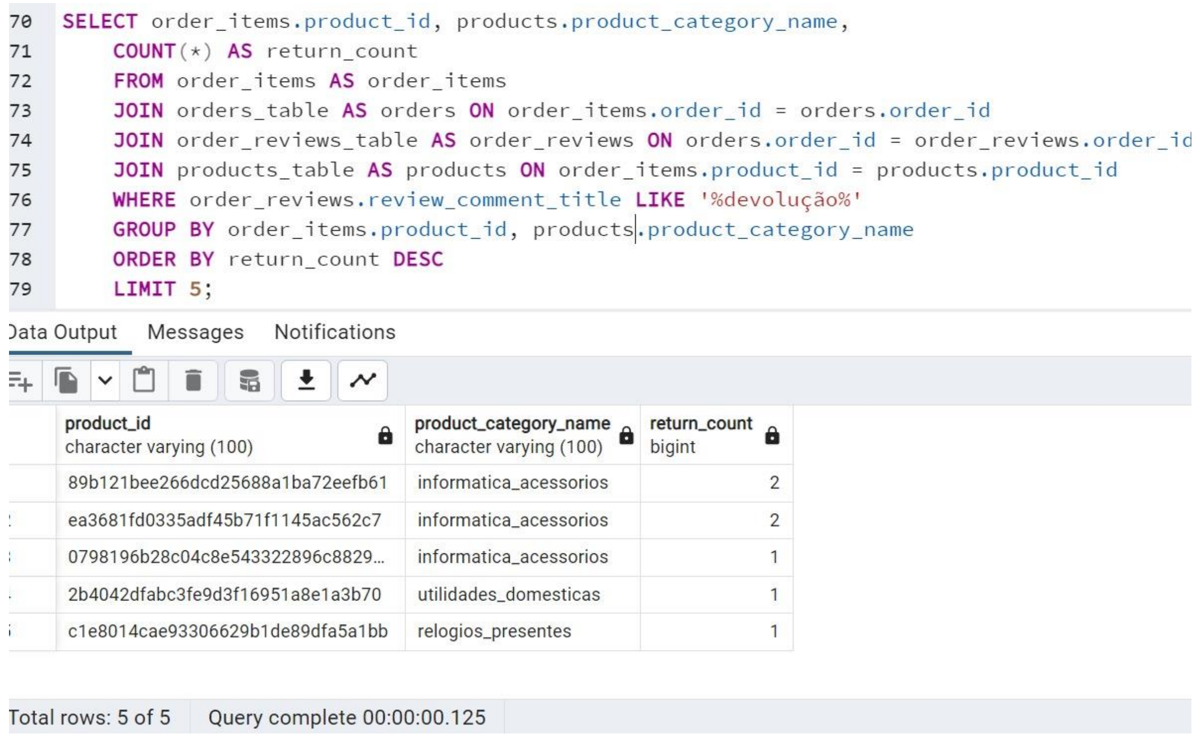
```
SELECT customers.customer_unique_id,
    COUNT(*) AS total_purchases
    FROM customers_table AS customers
    JOIN orders_table AS orders ON customers.customer_
    GROUP BY customers.customer_unique_id
    ORDER BY total_purchases DESC
    LIMIT 10;
```

	customer_unique_id character varying (100)	total_purchases bigint
1	8d50f5eadf50201ccdcedfb9e2ac8455	17
2	3e43e6105506432c953e165fb2acf44c	9
3	ca77025e7201e3b30c44b472ff3462...	7
4	1b6c7548a2a1f9037c1fd3ddfed95f33	7
5	6469f99c1f9dfae7733b25662e7f1782	7
6	63cfc61cee11cbe306bff5857d00bfe4	6
7	12f5d6e1cbf93dafd9dcc19095df0b3d	6
8	f0e310a6839dce9de1638e0fe5ab282a	6
9	de34b16117594161a6a89c50b289d...	6

Total rows: 10 of 10   Query complete 00:00:00.294

SELECT QUERY

- Query to find the top 5 products that have been returned the most:



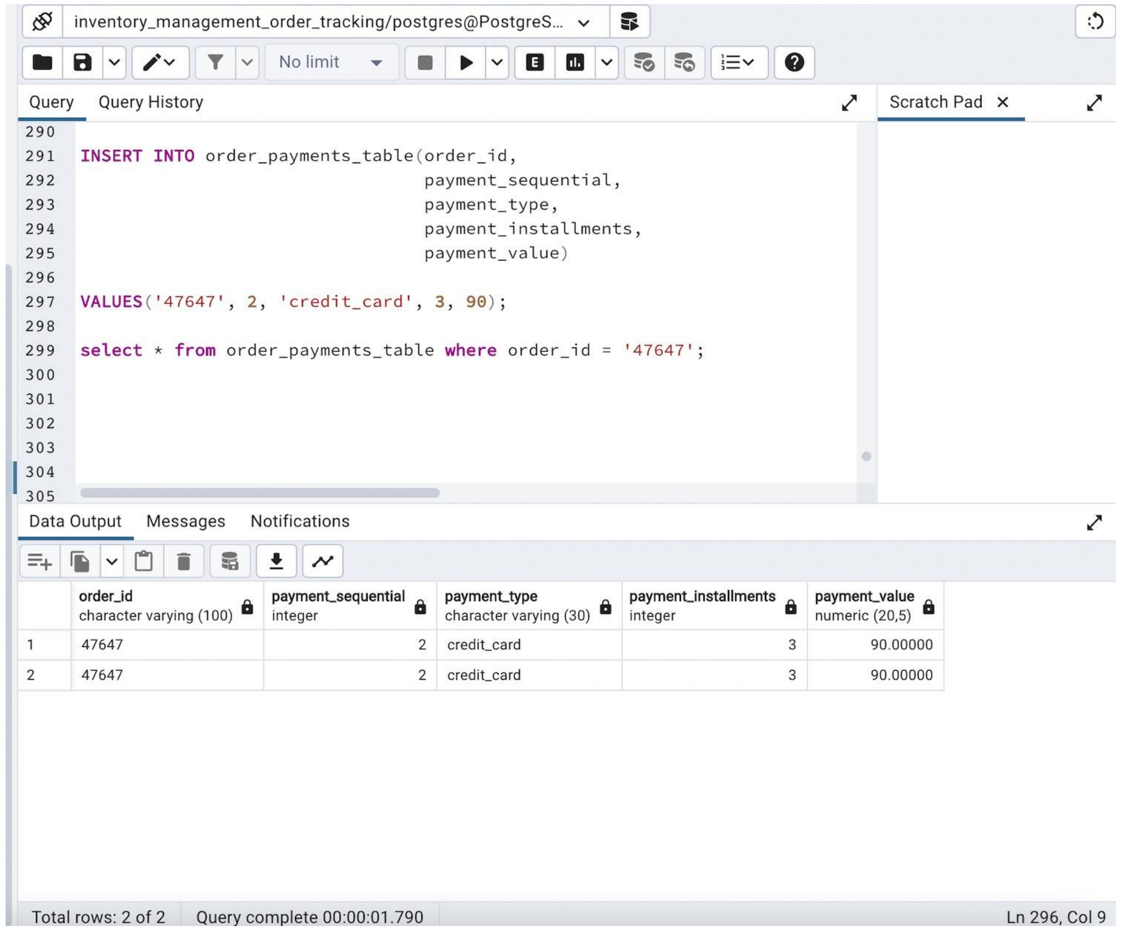
```sql
SELECT order_items.product_id, products.product_category_name,
    COUNT(*) AS return_count
    FROM order_items AS order_items
    JOIN orders_table AS orders ON order_items.order_id = orders.order_id
    JOIN order_reviews_table AS order_reviews ON orders.order_id = order_reviews.order_id
    JOIN products_table AS products ON order_items.product_id = products.product_id
    WHERE order_reviews.review_comment_title LIKE '%devolução%'
    GROUP BY order_items.product_id, products.product_category_name
    ORDER BY return_count DESC
    LIMIT 5;
```

Data Output Messages Notifications

product_id character varying (100)	product_category_name character varying (100)	return_count bigint
89b121bee266dcd25688a1ba72eefb61	informatica_acessorios	2
ea3681fd0335adf45b71f1145ac562c7	informatica_acessorios	2
0798196b28c04c8e543322896c8829…	informatica_acessorios	1
2b4042dfabc3fe9d3f16951a8e1a3b70	utilidades_domesticas	1
c1e8014cae93306629b1de89dfa5a1bb	relogios_presentes	1

Total rows: 5 of 5 Query complete 00:00:00.125

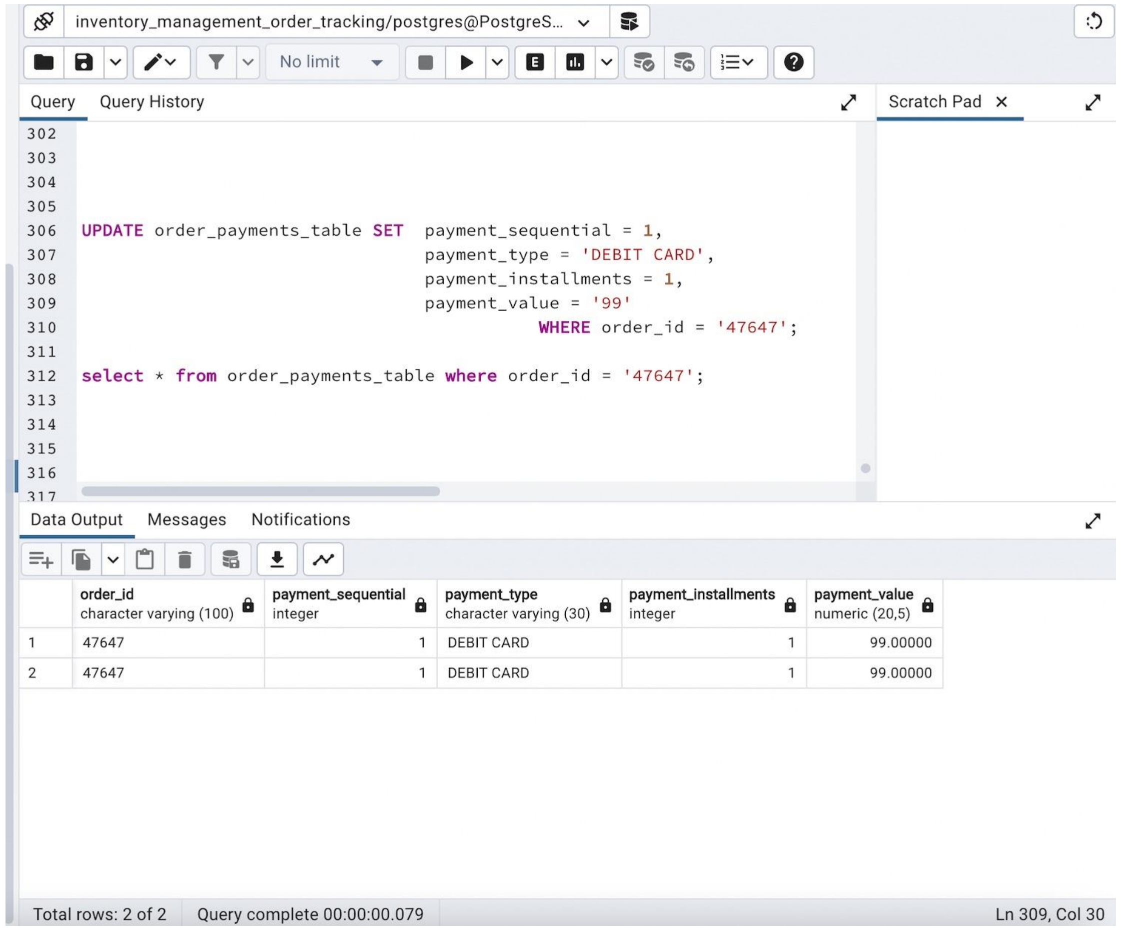
UPDATE QUERY


inventory_management_order_tracking/postgres@PostgreS...

```
INSERT INTO order_payments_table(order_id,
                                 payment_sequential,
                                 payment_type,
                                 payment_installments,
                                 payment_value)

VALUES('47647', 2, 'credit_card', 3, 90);

select * from order_payments_table where order_id = '47647';
```

	order_id character varying (100)	payment_sequential integer	payment_type character varying (30)	payment_installments integer	payment_value numeric (20,5)
1	47647	2	credit_card	3	90.00000
2	47647	2	credit_card	3	90.00000

Total rows: 2 of 2   Query complete 00:00:01.790   Ln 296, Col 9

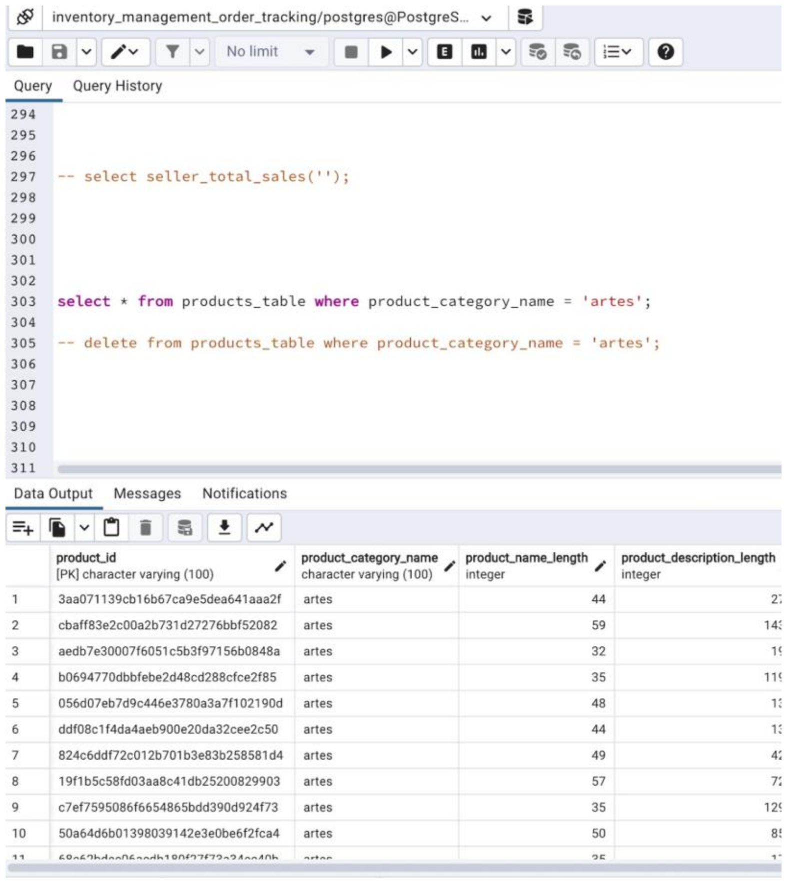

inventory_management_order_tracking/postgres@PostgreS...

```
UPDATE order_payments_table SET  payment_sequential = 1,
                                 payment_type = 'DEBIT CARD',
                                 payment_installments = 1,
                                 payment_value = '99'
                                             WHERE order_id = '47647';

select * from order_payments_table where order_id = '47647';
```

	order_id character varying (100)	payment_sequential integer	payment_type character varying (30)	payment_installments integer	payment_value numeric (20,5)
1	47647	1	DEBIT CARD	1	99.00000
2	47647	1	DEBIT CARD	1	99.00000

Total rows: 2 of 2   Query complete 00:00:00.079   Ln 309, Col 30

DELETE QUERY

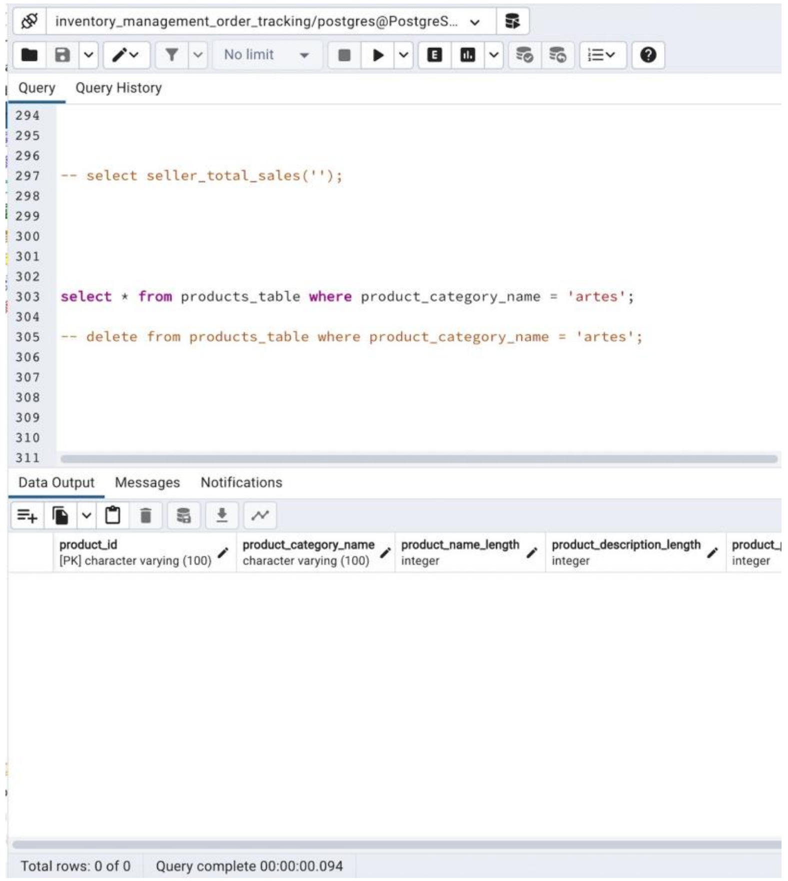


inventory_management_order_tracking/postgres@PostgreS...

```
-- select seller_total_sales('');



select * from products_table where product_category_name = 'artes';

-- delete from products_table where product_category_name = 'artes';
```

Data Output | Messages | Notifications

	product_id [PK] character varying (100)	product_category_name character varying (100)	product_name_length integer	product_description_length integer
1	3aa071139cb16b67ca9e5dea641aaa2f	artes	44	27
2	cbaff83e2c00a2b731d27276bbf52082	artes	59	143
3	aedb7e30007f6051c5b3f97156b0848a	artes	32	19
4	b0694770dbbfebe2d48cd288cfce2f85	artes	35	119
5	056d07eb7d9c446e3780a3a7f102190d	artes	48	13
6	ddf08c1f4da4aeb900e20da32cee2c50	artes	44	13
7	824c6ddf72c012b701b3e83b258581d4	artes	49	42
8	19f1b5c58fd03aa8c41db25200829903	artes	57	72
9	c7ef7595086f6654865bdd390d924f73	artes	35	129
10	50a64d6b01398039142e3e0be6f2fca4	artes	50	85
11	[illegible]	[illegible]	35	[illegible]



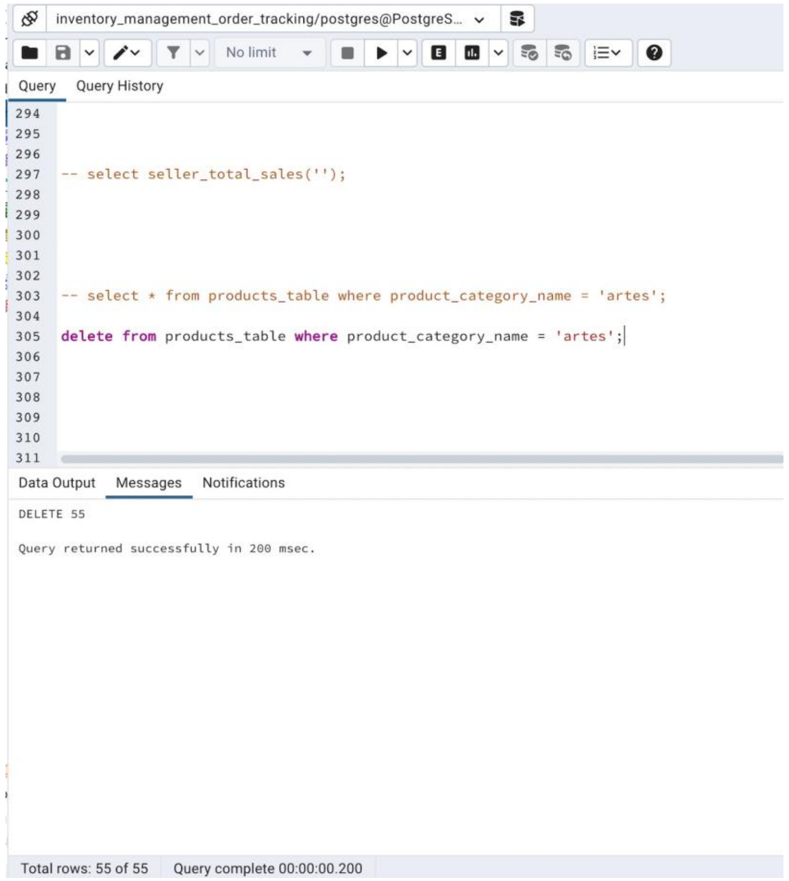
inventory_management_order_tracking/postgres@PostgreS...

```
-- select seller_total_sales('');



select * from products_table where product_category_name = 'artes';

-- delete from products_table where product_category_name = 'artes';
```

Data Output | Messages | Notifications

	product_id [PK] character varying (100)	product_category_name character varying (100)	product_name_length integer	product_description_length integer	product_ integer

Total rows: 0 of 0 Query complete 00:00:00.094



inventory_management_order_tracking/postgres@PostgreS...

```
-- select seller_total_sales('');



-- select * from products_table where product_category_name = 'artes';

delete from products_table where product_category_name = 'artes';
```

Data Output | Messages | Notifications

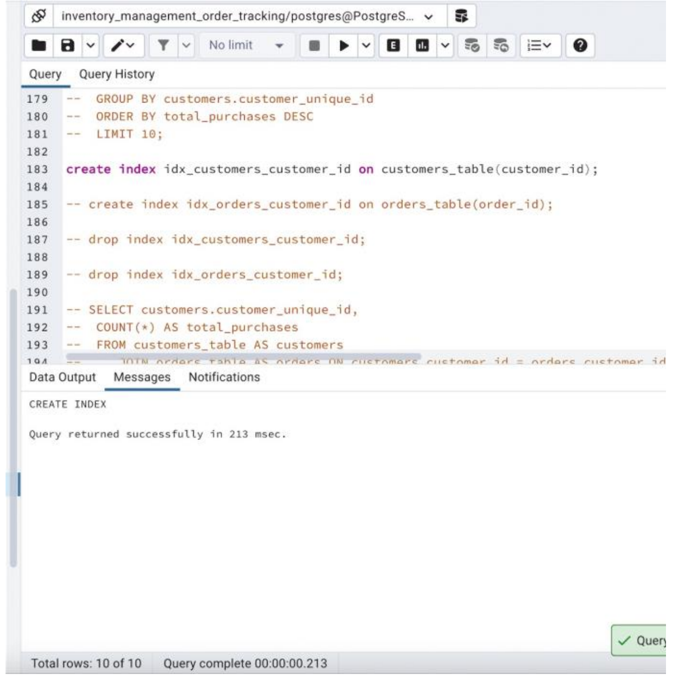
```
DELETE 55

Query returned successfully in 200 msec.
```

Total rows: 55 of 55 Query complete 00:00:00.200

QUERY EXECUTION ANALYSES

1. INDEXING

While running some complex select queries involving one or more databases with huge amount of data, we noticed a larger amount of time for query execution. In the example given, we run a select query involving a join operation on orders_table and customers_table. Upon adding an index to order_id in orders_table we see a reduction in the query execution time. Upon adding an index to customer_id in customers_table also we see a further reduction in the query execution time.

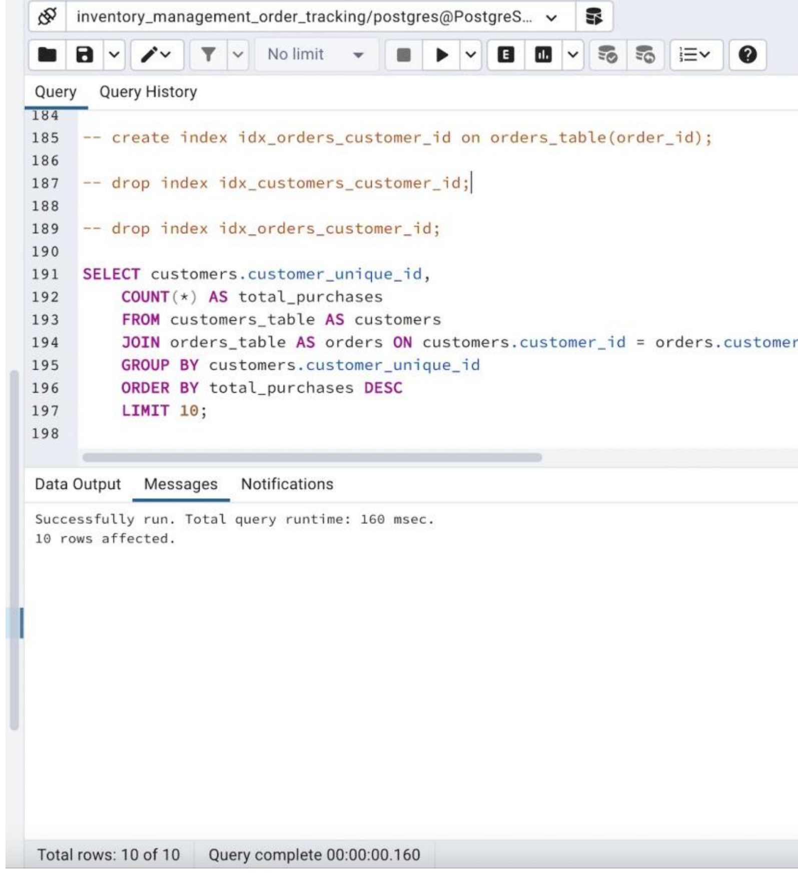

inventory_management_order_tracking/postgres@PostgreS...

```
--  GROUP BY customers.customer_unique_id
--  ORDER BY total_purchases DESC
--  LIMIT 10;

create index idx_customers_customer_id on customers_table(customer_id);

-- create index idx_orders_customer_id on orders_table(order_id);

-- drop index idx_customers_customer_id;

-- drop index idx_orders_customer_id;

-- SELECT customers.customer_unique_id,
--  COUNT(*) AS total_purchases
--  FROM customers_table AS customers
```

Data Output | Messages | Notifications

```
CREATE INDEX

Query returned successfully in 213 msec.
```

Total rows: 10 of 10   Query complete 00:00:00.213

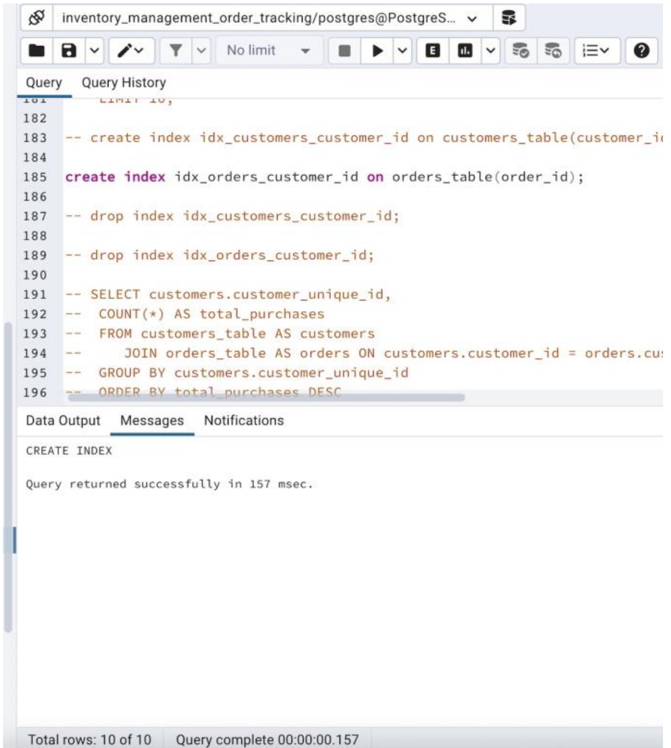

inventory_management_order_tracking/postgres@PostgreS...

```
-- create index idx_orders_customer_id on orders_table(order_id);

-- drop index idx_customers_customer_id;

-- drop index idx_orders_customer_id;

SELECT customers.customer_unique_id,
    COUNT(*) AS total_purchases
    FROM customers_table AS customers
    JOIN orders_table AS orders ON customers.customer_id = orders.custome
    GROUP BY customers.customer_unique_id
    ORDER BY total_purchases DESC
    LIMIT 10;
```

Data Output | Messages | Notifications

```
Successfully run. Total query runtime: 160 msec.
10 rows affected.
```

Total rows: 10 of 10   Query complete 00:00:00.160

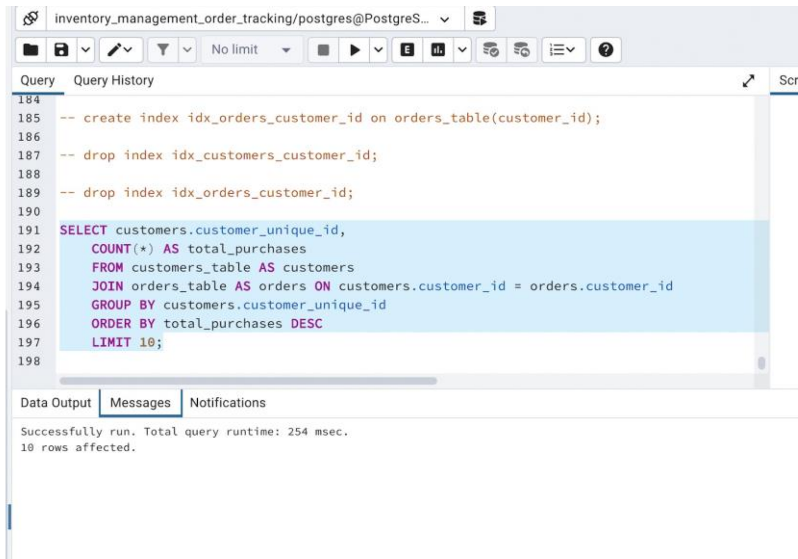

inventory_management_order_tracking/postgres@PostgreS...

```
-- create index idx_customers_customer_id on customers_table(customer_i

create index idx_orders_customer_id on orders_table(order_id);

-- drop index idx_customers_customer_id;

-- drop index idx_orders_customer_id;

-- SELECT customers.customer_unique_id,
--  COUNT(*) AS total_purchases
--  FROM customers_table AS customers
--     JOIN orders_table AS orders ON customers.customer_id = orders.cu
--  GROUP BY customers.customer_unique_id
--  ORDER BY total_purchases DESC
```

Data Output | Messages | Notifications

```
CREATE INDEX

Query returned successfully in 157 msec.
```

Total rows: 10 of 10   Query complete 00:00:00.157


inventory_management_order_tracking/postgres@PostgreS...

```
-- create index idx_orders_customer_id on orders_table(customer_id);

-- drop index idx_customers_customer_id;

-- drop index idx_orders_customer_id;

SELECT customers.customer_unique_id,
    COUNT(*) AS total_purchases
    FROM customers_table AS customers
    JOIN orders_table AS orders ON customers.customer_id = orders.customer_id
    GROUP BY customers.customer_unique_id
    ORDER BY total_purchases DESC
    LIMIT 10;
```

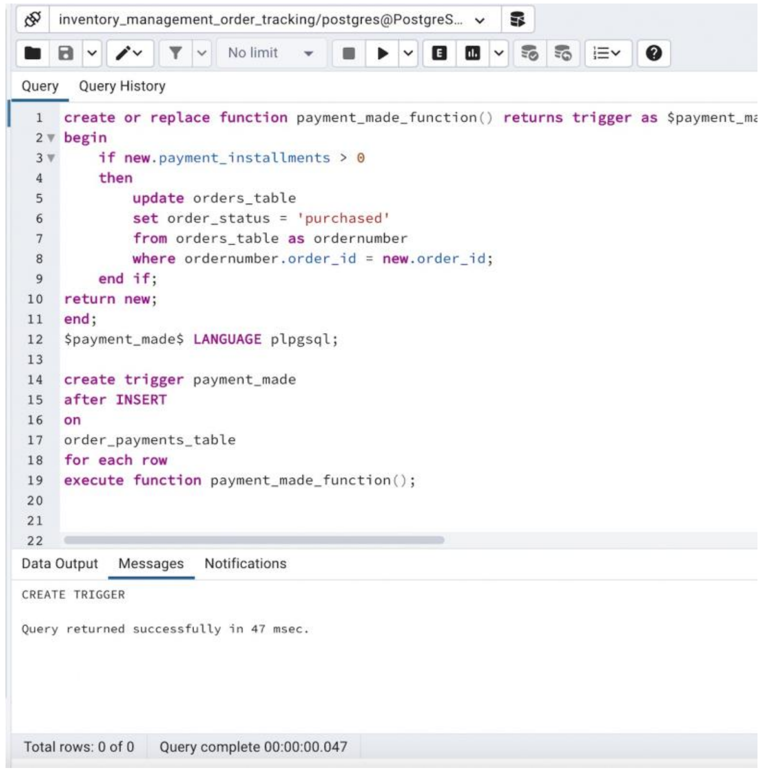
Data Output | Messages | Notifications

```
Successfully run. Total query runtime: 254 msec.
10 rows affected.
```

QUERY EXECUTION ANALYSES

2. TRIGGER COMMAND

- In some cases when one table in the database is updated with insert, update or delete command we also need to update a related table. In such cases trigger command with function implementation helps carry out the changes in related tables. In our case, when a successful payment is made, the order_payments_table has a record inserted, which upon insertion triggers and update on the order_status column of orders_table.


inventory_management_order_tracking/postgres@PostgreS...

No limit

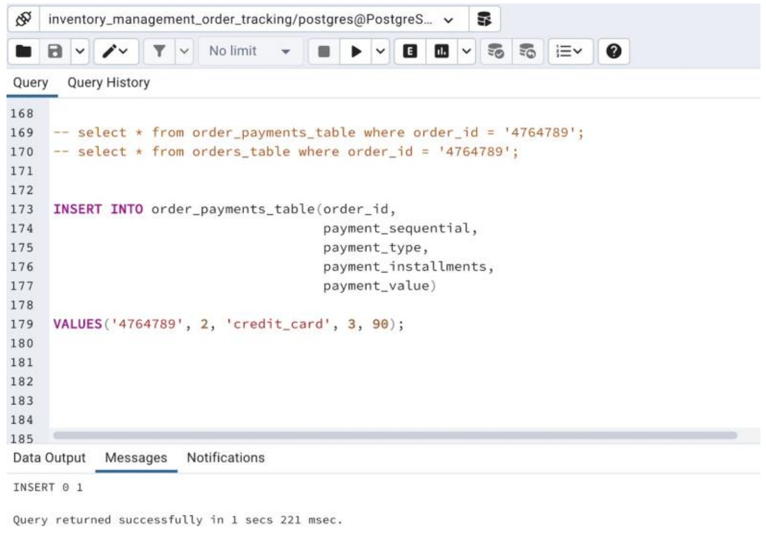
Query | Query History

```
create or replace function payment_made_function() returns trigger as $payment_ma
begin
    if new.payment_installments > 0
    then
        update orders_table
        set order_status = 'purchased'
        from orders_table as ordernumber
        where ordernumber.order_id = new.order_id;
    end if;
return new;
end;
$payment_made$ LANGUAGE plpgsql;

create trigger payment_made
after INSERT
on
order_payments_table
for each row
execute function payment_made_function();
```

Data Output | Messages | Notifications

```
CREATE TRIGGER

Query returned successfully in 47 msec.
```

Total rows: 0 of 0 | Query complete 00:00:00.047


inventory_management_order_tracking/postgres@PostgreS...

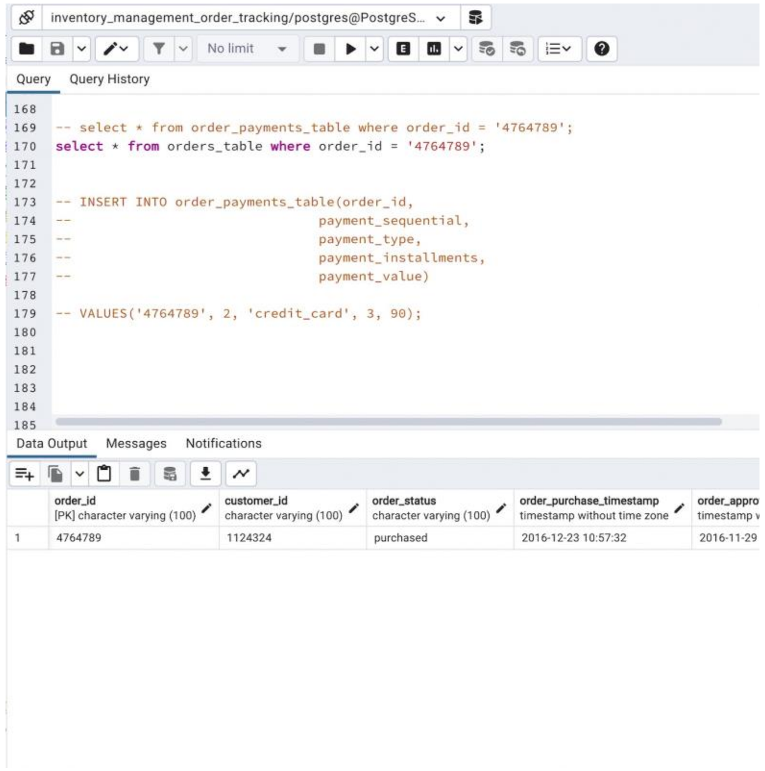
No limit

Query | Query History

```
-- select * from order_payments_table where order_id = '4764789';
-- select * from orders_table where order_id = '4764789';


INSERT INTO order_payments_table(order_id,
                                 payment_sequential,
                                 payment_type,
                                 payment_installments,
                                 payment_value)

VALUES('4764789', 2, 'credit_card', 3, 90);
```

Data Output | Messages | Notifications

```
INSERT 0 1

Query returned successfully in 1 secs 221 msec.
```


inventory_management_order_tracking/postgres@PostgreS...

No limit

Query | Query History

```
-- select * from order_payments_table where order_id = '4764789';
select * from orders_table where order_id = '4764789';


-- INSERT INTO order_payments_table(order_id,
--                                  payment_sequential,
--                                  payment_type,
--                                  payment_installments,
--                                  payment_value)

-- VALUES('4764789', 2, 'credit_card', 3, 90);
```

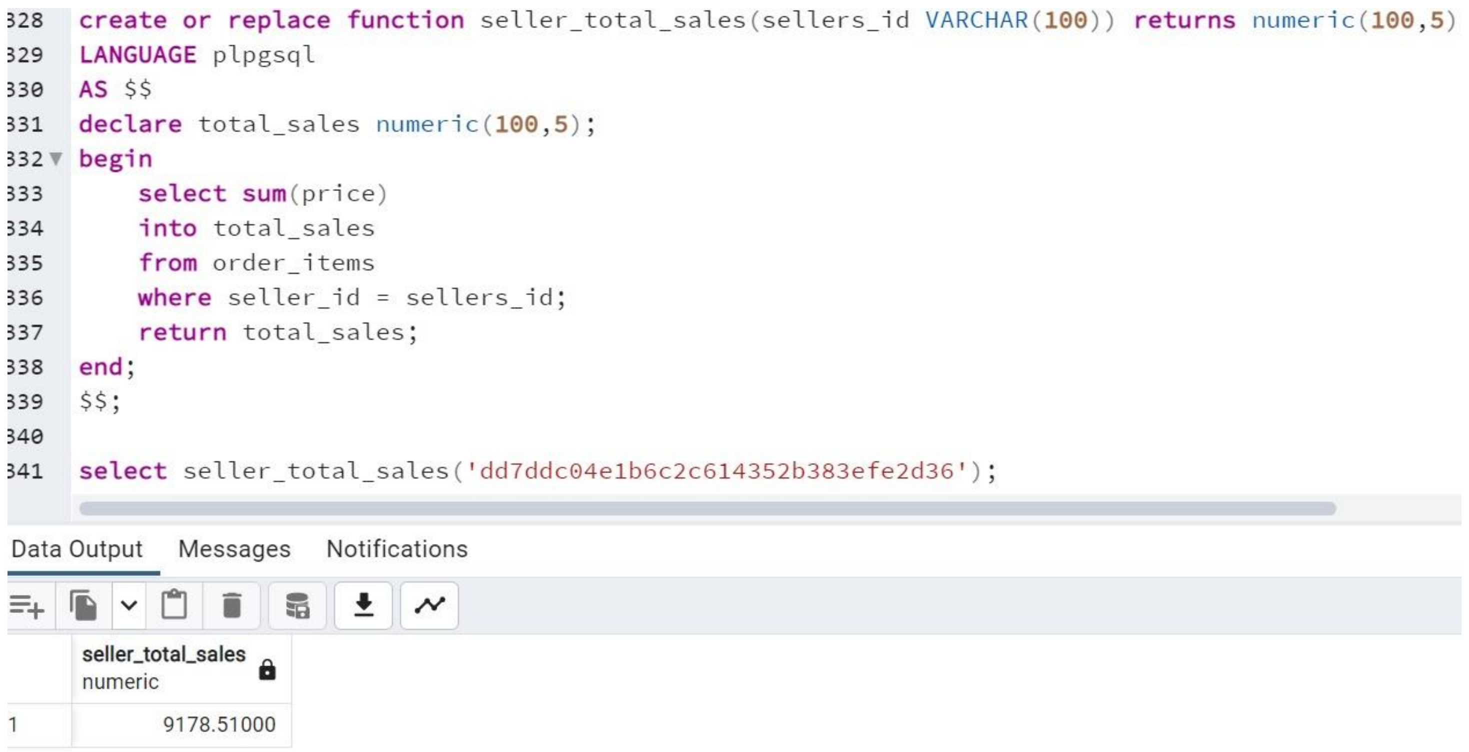
Data Output | Messages | Notifications

	order_id [PK] character varying (100)	customer_id character varying (100)	order_status character varying (100)	order_purchase_timestamp timestamp without time zone	order_appro timestamp v
1	4764789	1124324	purchased	2016-12-23 10:57:32	2016-11-29

QUERY EXECUTION ANALYSES

3. ADDING FUNCTIONS

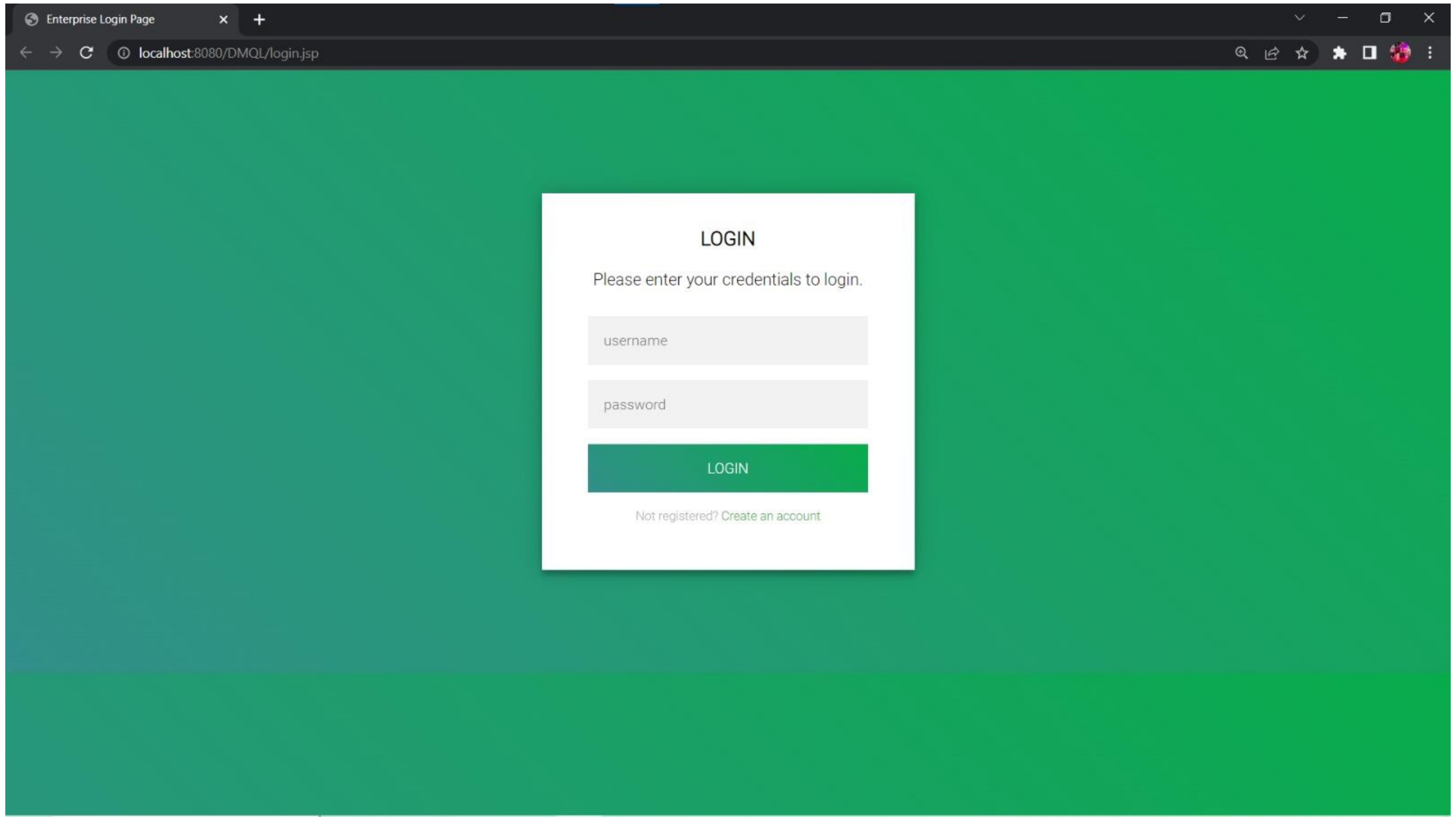
- In certain tables, certain calculations help in data analysis. We can add functions to carry out such operations that are required repeatedly for different values. In the example, we have shown a function which calculates the most up to date total sales amount of a particular seller.



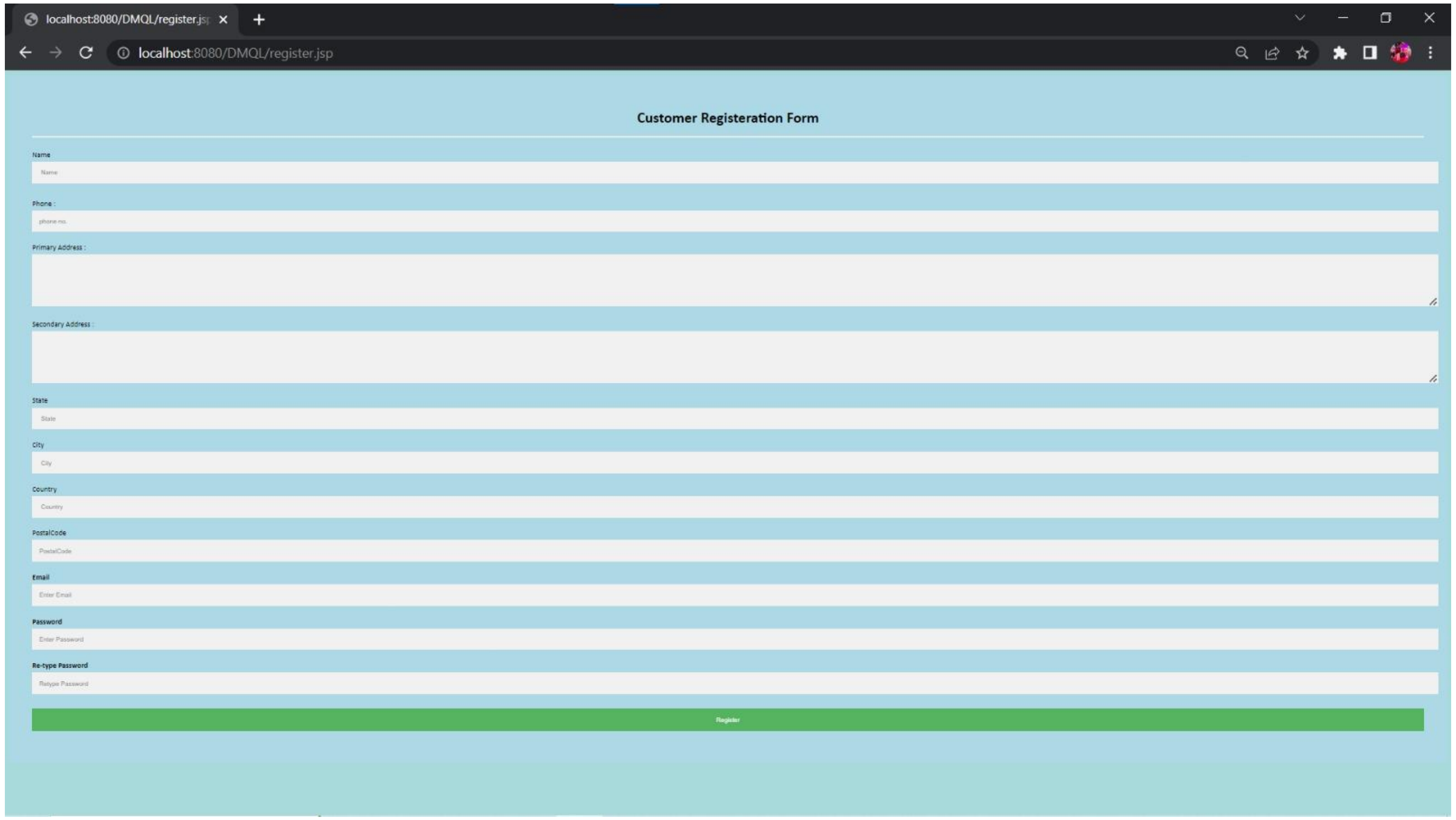
```
create or replace function seller_total_sales(sellers_id VARCHAR(100)) returns numeric(100,5)
LANGUAGE plpgsql
AS $$
declare total_sales numeric(100,5);
begin
    select sum(price)
    into total_sales
    from order_items
    where seller_id = sellers_id;
    return total_sales;
end;
$$;

select seller_total_sales('dd7ddc04e1b6c2c614352b383efe2d36');
```

```
  seller_total_sales
  numeric
1        9178.51000
```

GUI : Login page (existing user)


Enterprise Login Page
localhost:8080/DMQL/login.jsp
LOGIN
Please enter your credentials to login.
username
password
LOGIN
Not registered? Create an account

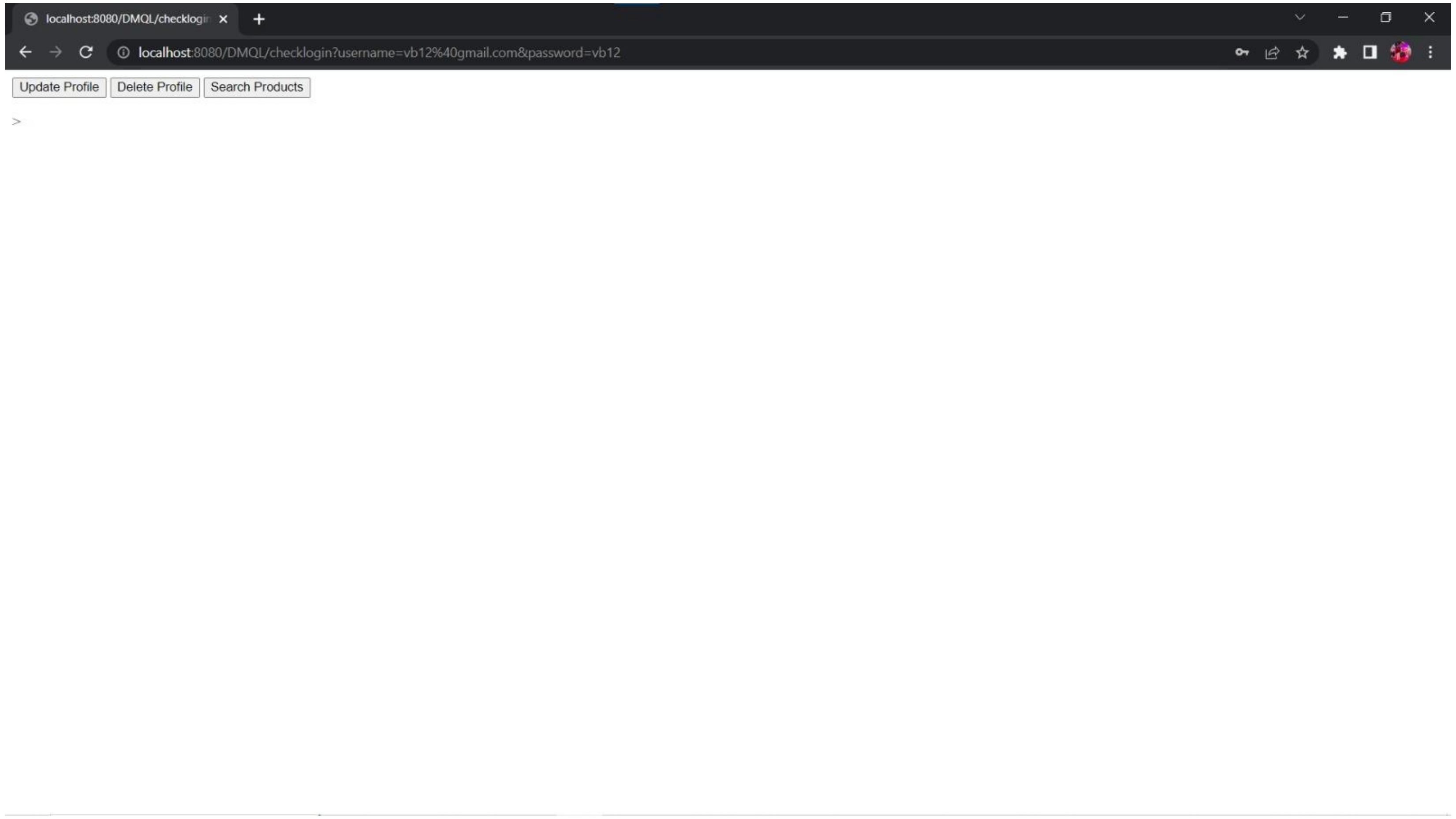
GUI : New customer login


localhost:8080/DMQL/register.jsp
localhost:8080/DMQL/register.jsp

Customer Registeration Form

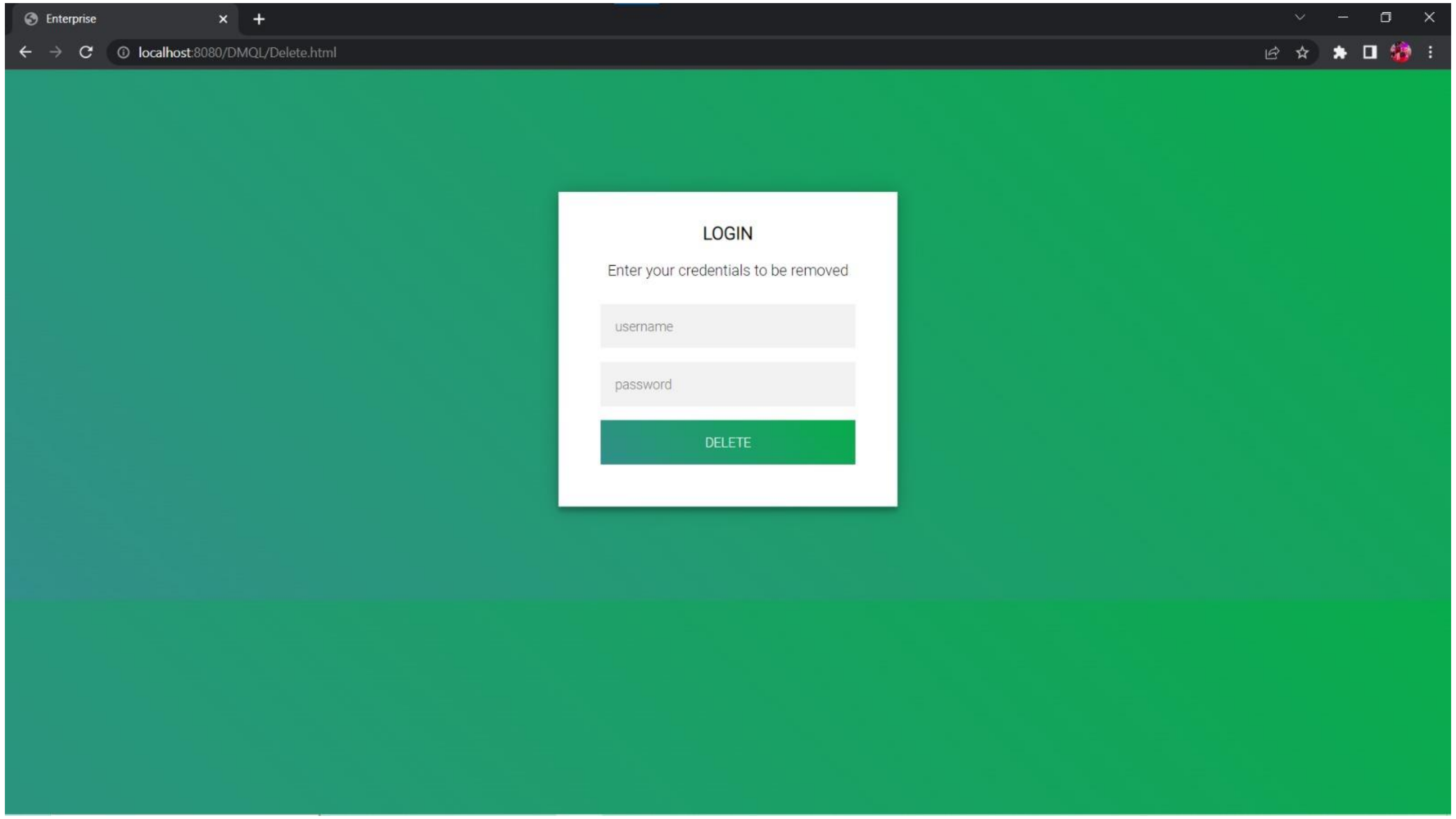
Name	Name
Phone :	phone no.
Primary Address :	
Secondary Address :	
State	State
City	City
Country	Country
PostalCode	PostalCode
Email	Enter Email
Password	Enter Password
Re-type Password	Retype Password

GUI : Actions which can be performed


localhost:8080/DMQL/checklogi
localhost:8080/DMQL/checklogin?username=vb12%40gmail.com&password=vb12
Update Profile
Delete Profile
Search Products

>

GUI : Deleting user from the database

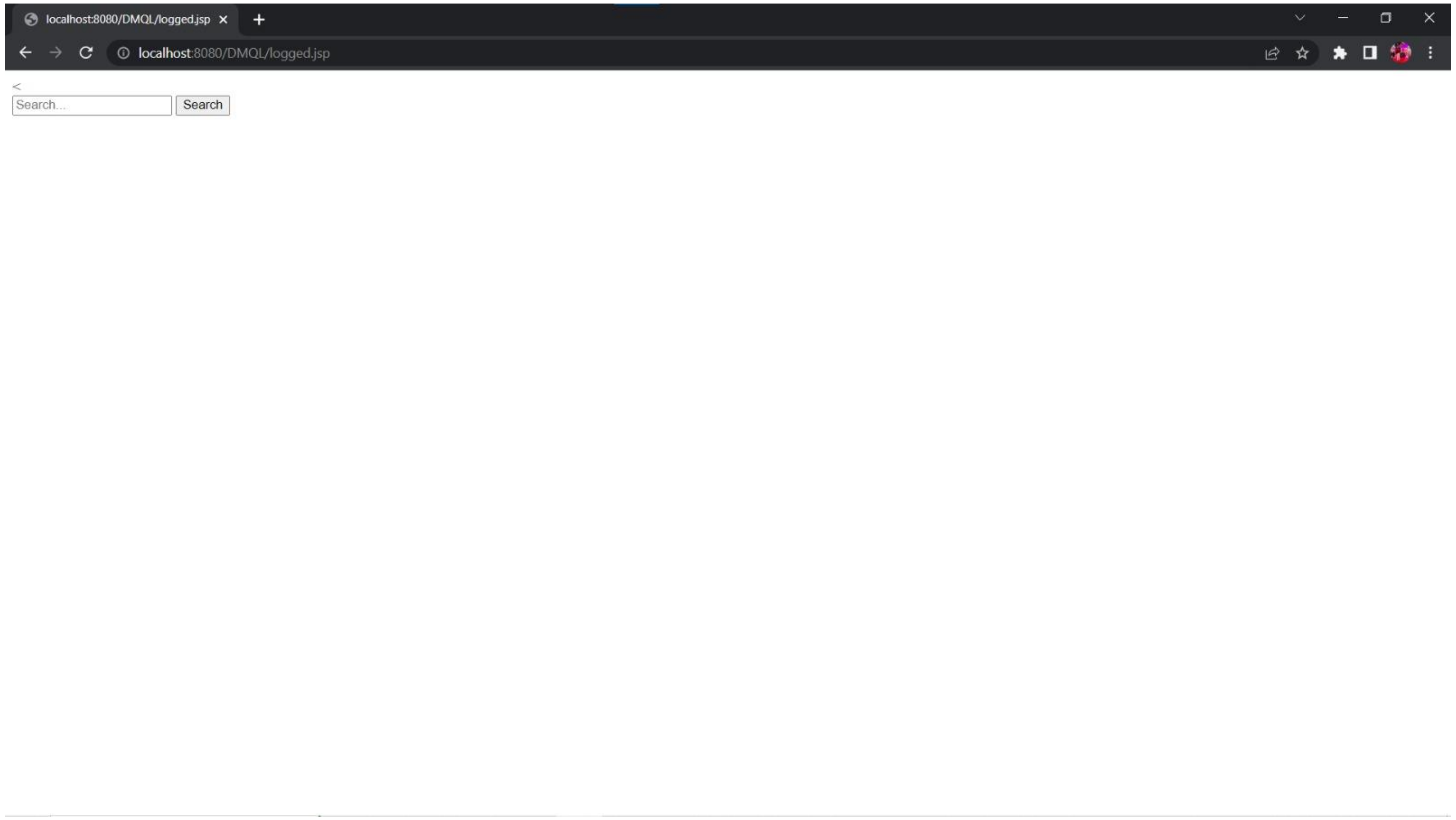

Enterprise

localhost:8080/DMQL/Delete.html

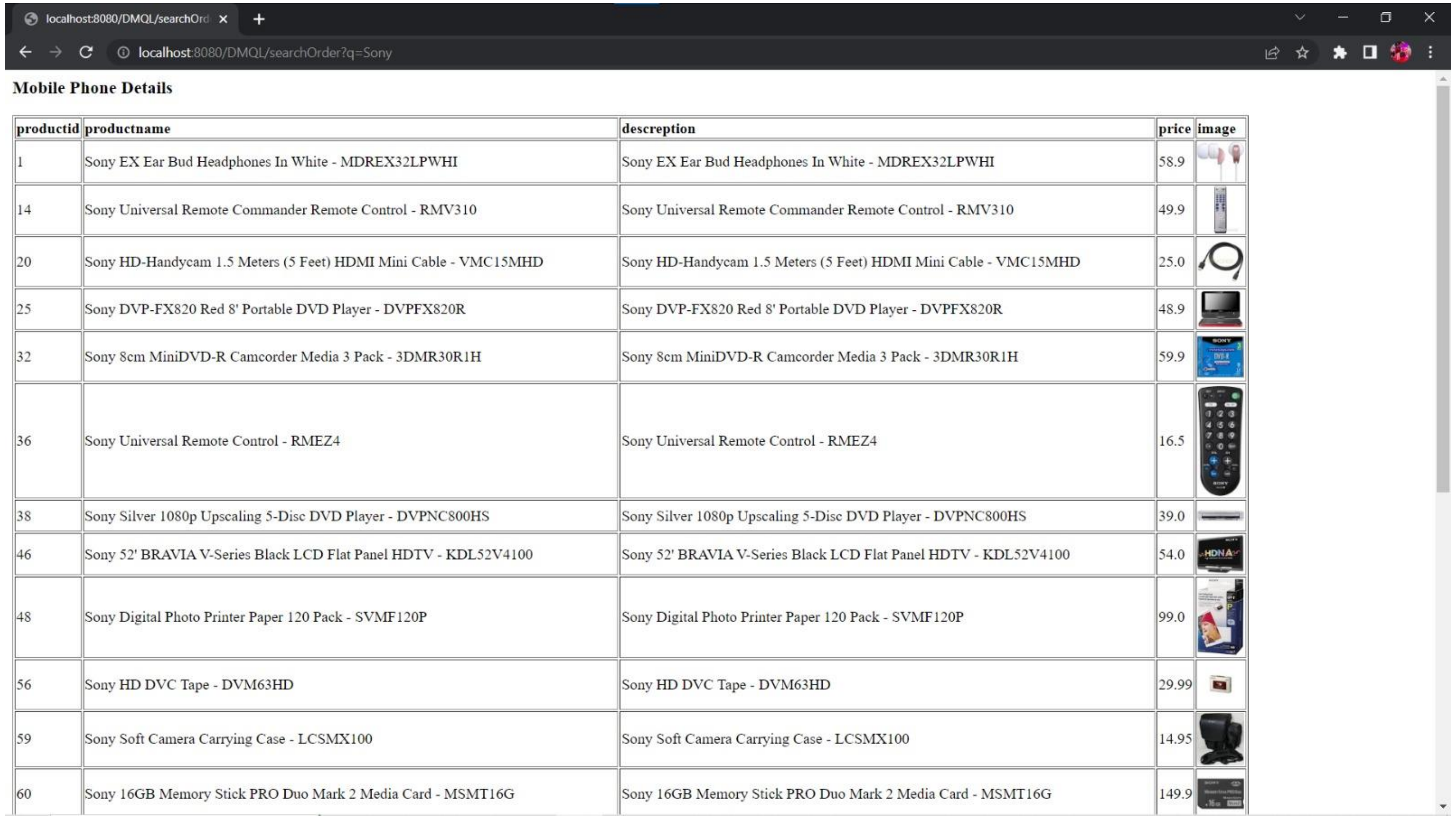
LOGIN

Enter your credentials to be removed

GUI : Customers searching for the products to purchase


localhost:8080/DMQL/logged.jsp
localhost:8080/DMQL/logged.jsp
<
Search...
Search

GUI : Search images for the word 'Sony'



localhost:8080/DMQL/searchOrder?q=Sony

Mobile Phone Details

productid	productname	descreption	price	image
1	Sony EX Ear Bud Headphones In White - MDREX32LPWHI	Sony EX Ear Bud Headphones In White - MDREX32LPWHI	58.9	
14	Sony Universal Remote Commander Remote Control - RMV310	Sony Universal Remote Commander Remote Control - RMV310	49.9	
20	Sony HD-Handycam 1.5 Meters (5 Feet) HDMI Mini Cable - VMC15MHD	Sony HD-Handycam 1.5 Meters (5 Feet) HDMI Mini Cable - VMC15MHD	25.0	
25	Sony DVP-FX820 Red 8' Portable DVD Player - DVPFX820R	Sony DVP-FX820 Red 8' Portable DVD Player - DVPFX820R	48.9	
32	Sony 8cm MiniDVD-R Camcorder Media 3 Pack - 3DMR30R1H	Sony 8cm MiniDVD-R Camcorder Media 3 Pack - 3DMR30R1H	59.9	
36	Sony Universal Remote Control - RMEZ4	Sony Universal Remote Control - RMEZ4	16.5	
38	Sony Silver 1080p Upscaling 5-Disc DVD Player - DVPNC800HS	Sony Silver 1080p Upscaling 5-Disc DVD Player - DVPNC800HS	39.0	
46	Sony 52' BRAVIA V-Series Black LCD Flat Panel HDTV - KDL52V4100	Sony 52' BRAVIA V-Series Black LCD Flat Panel HDTV - KDL52V4100	54.0	
48	Sony Digital Photo Printer Paper 120 Pack - SVMF120P	Sony Digital Photo Printer Paper 120 Pack - SVMF120P	99.0	
56	Sony HD DVC Tape - DVM63HD	Sony HD DVC Tape - DVM63HD	29.99	
59	Sony Soft Camera Carrying Case - LCSMX100	Sony Soft Camera Carrying Case - LCSMX100	14.95	
60	Sony 16GB Memory Stick PRO Duo Mark 2 Media Card - MSMT16G	Sony 16GB Memory Stick PRO Duo Mark 2 Media Card - MSMT16G	149.9	

REFERENCES

https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce

https://www.w3schools.com/sql/

https://www.enterprisedb.com/downloads/postgres-postgresql-downloads

https://extendsclass.com/csv-generator.html

https://www.eclipse.org/downloads/packages/release/neon/3/eclipse-ide-java-ee-developers

https://tomcat.apache.org/download-90.cgi

https://www.postgresqltutorial.com/postgresql-jdbc/connecting-to-postgresql-database/

THANK YOU !