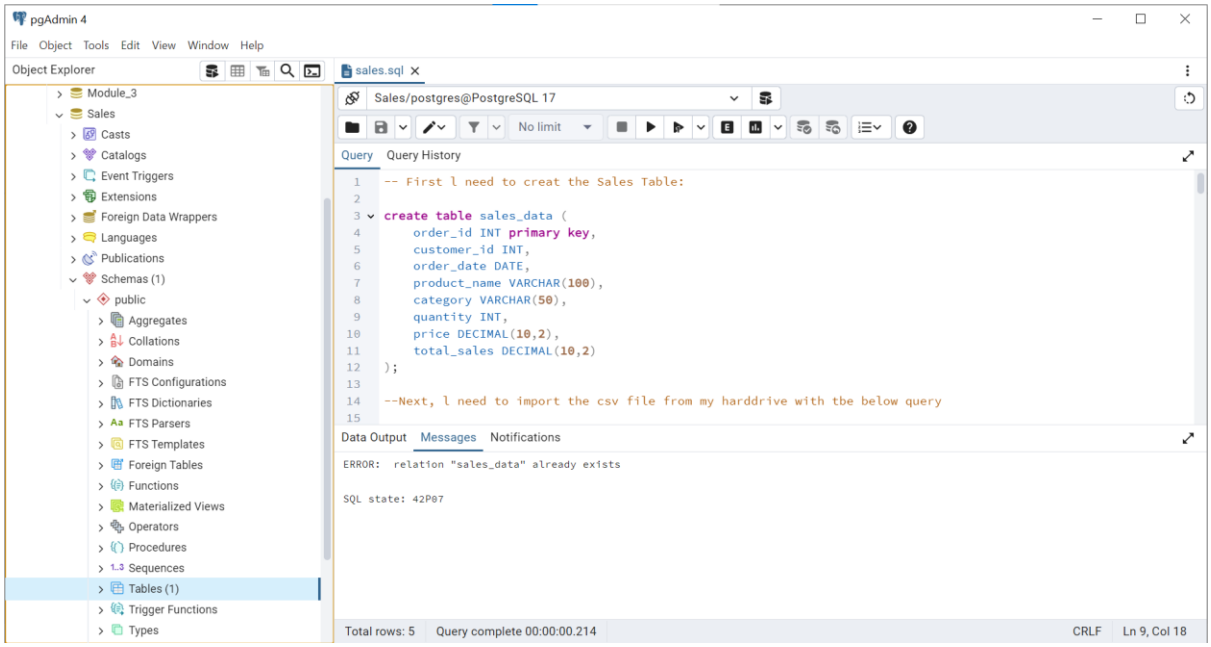
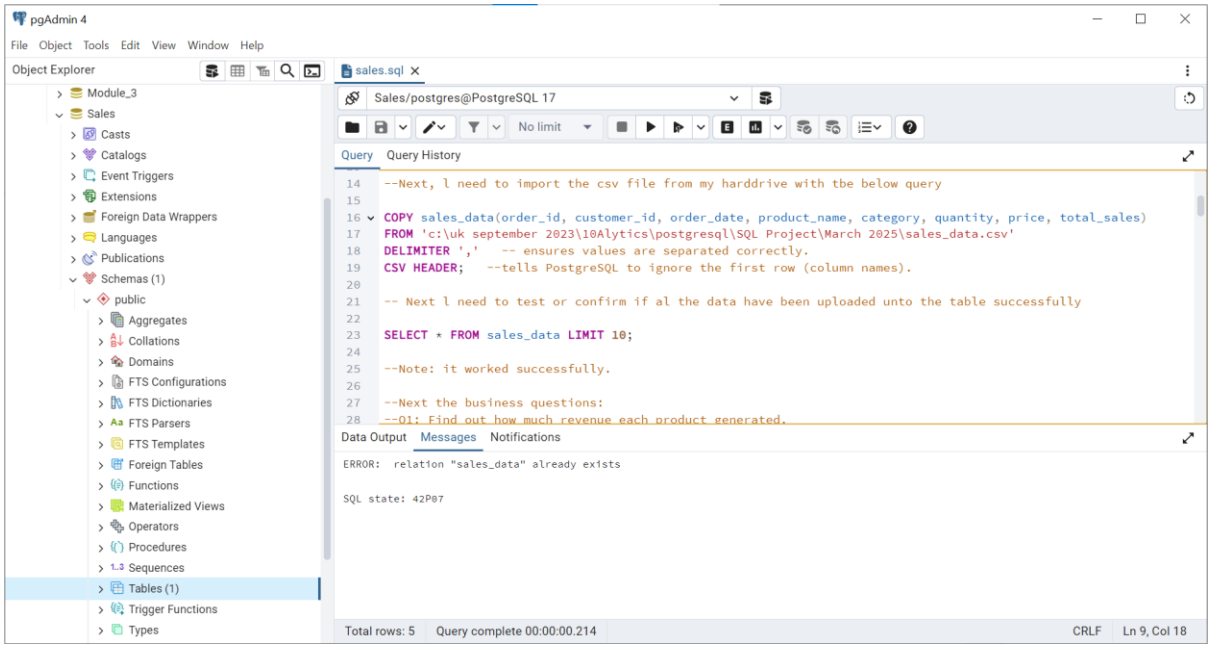
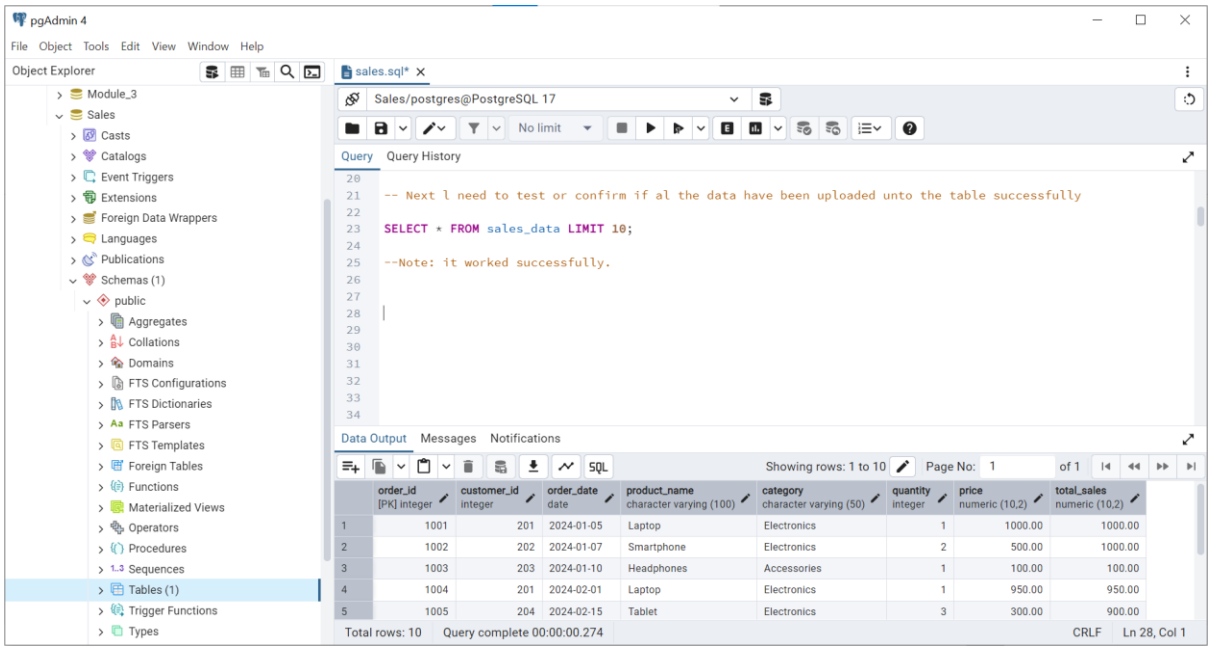
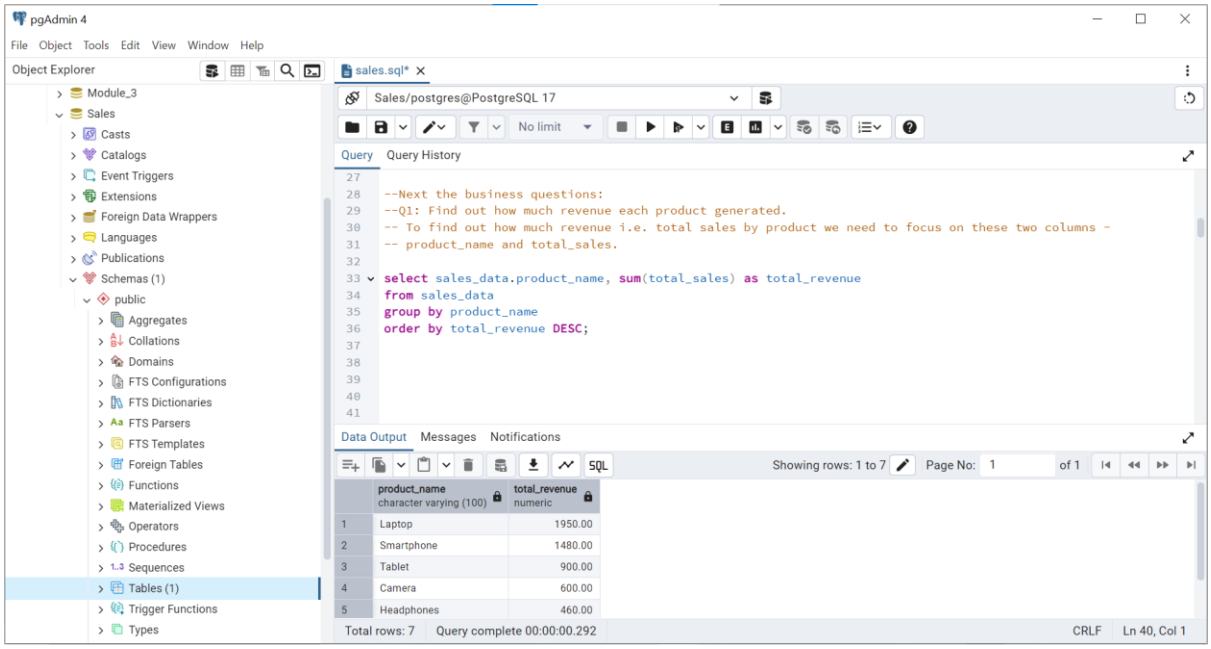
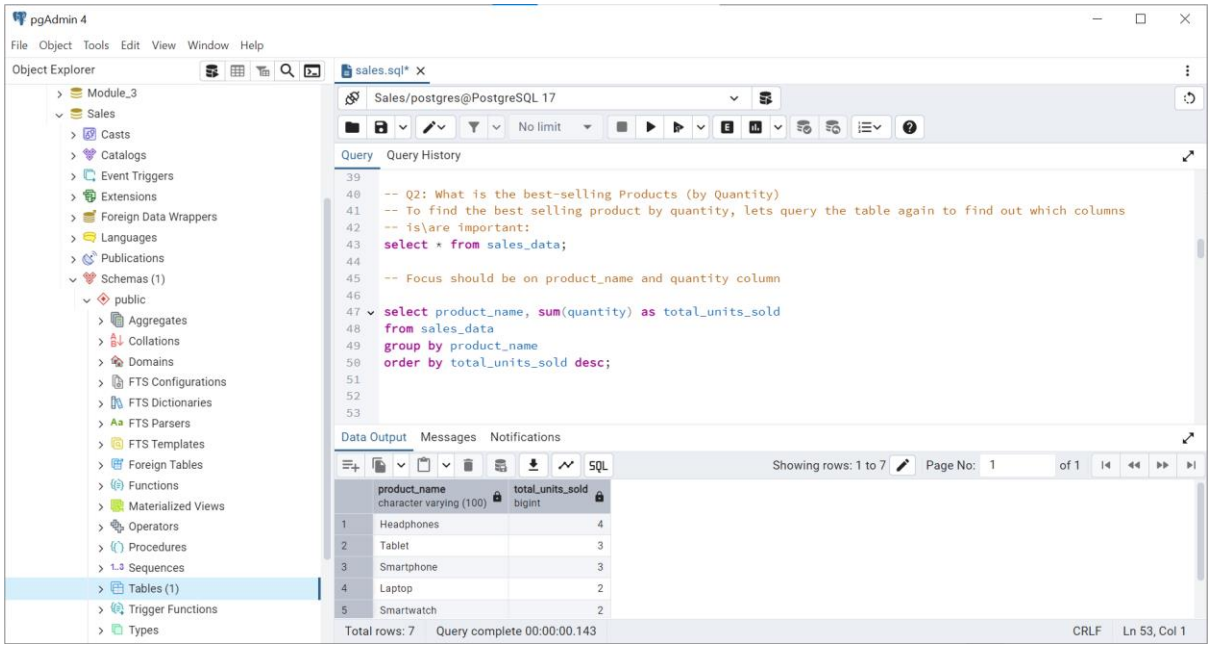
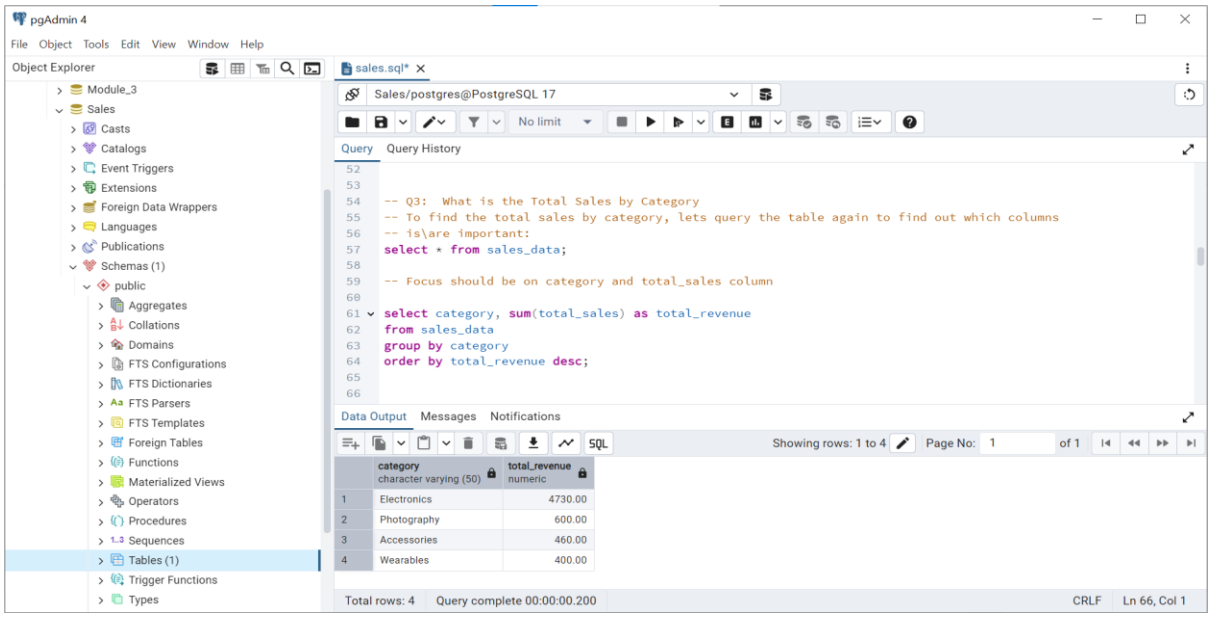

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
sales.sql
Sales/postgres@PostgreSQL 17
No limit
Query Query History
1 -- First l need to creat the Sales Table:
2
3 create table sales_data (
4 order_id INT primary key,
5 customer_id INT,
6 order_date DATE,
7 product_name VARCHAR(100),
8 category VARCHAR(50),
9 quantity INT,
10 price DECIMAL(10,2),
11 total_sales DECIMAL(10,2)
12);
13
14 --Next, l need to import the csv file from my harddrive with tbe below query
15
Data Output Messages Notifications
ERROR: relation "sales_data" already exists
SQL state: 42P07
Total rows: 5 Query complete 00:00:00.214 CRLF Ln 9, Col 18
Module_3
Sales
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (1)
Trigger Functions
Types

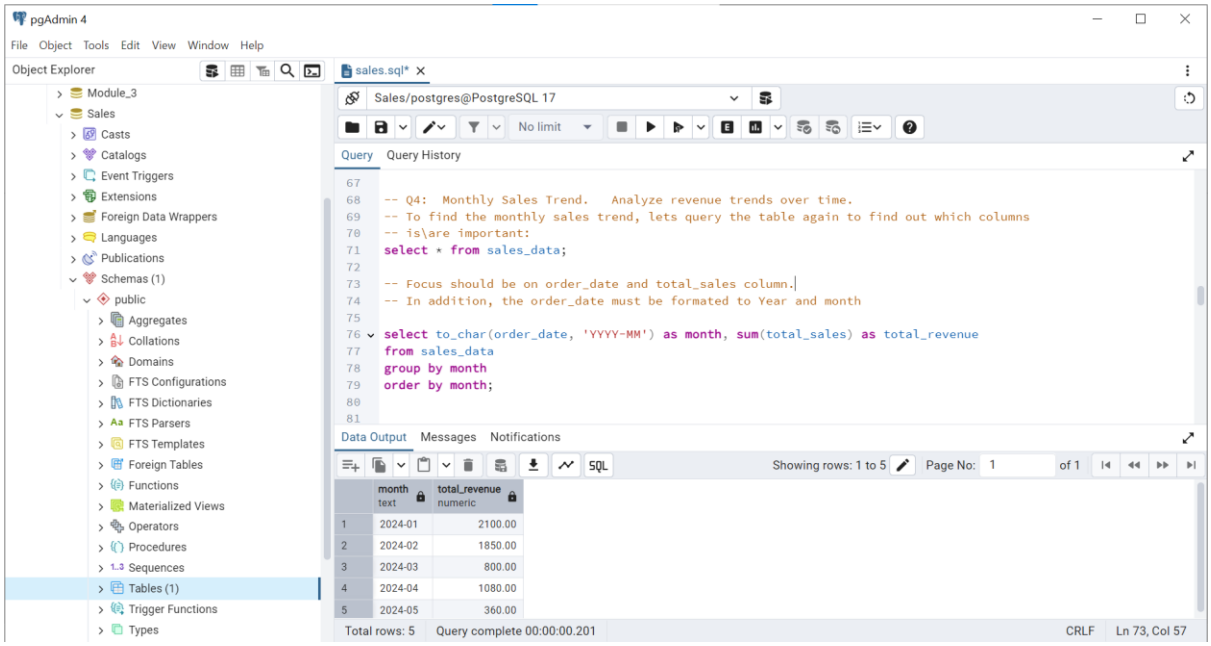

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
sales.sql
Sales/postgres@PostgreSQL 17
No limit
Query Query History
14 --Next, l need to import the csv file from my harddrive with tbe below query
15
16 COPY sales_data(order_id, customer_id, order_date, product_name, category, quantity, price, total_sales)
17 FROM 'c:\uk september 2023\10Alytics\postgresql\SQL Project\March 2025\sales_data.csv'
18 DELIMITER ',' -- ensures values are separated correctly.
19 CSV HEADER; --tells PostgreSQL to ignore the first row (column names).
20
21 -- Next l need to test or confirm if al the data have been uploaded unto the table successfully
22
23 SELECT * FROM sales_data LIMIT 10;
24
25 --Note: it worked successfully.
26
27 --Next the business questions:
28 --Q1: Find out how much revenue each product generated.
Data Output Messages Notifications
ERROR: relation "sales_data" already exists
SQL state: 42P07
Total rows: 5 Query complete 00:00:00.214 CRLF Ln 9, Col 18
Module_3
Sales
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (1)
Trigger Functions
Types

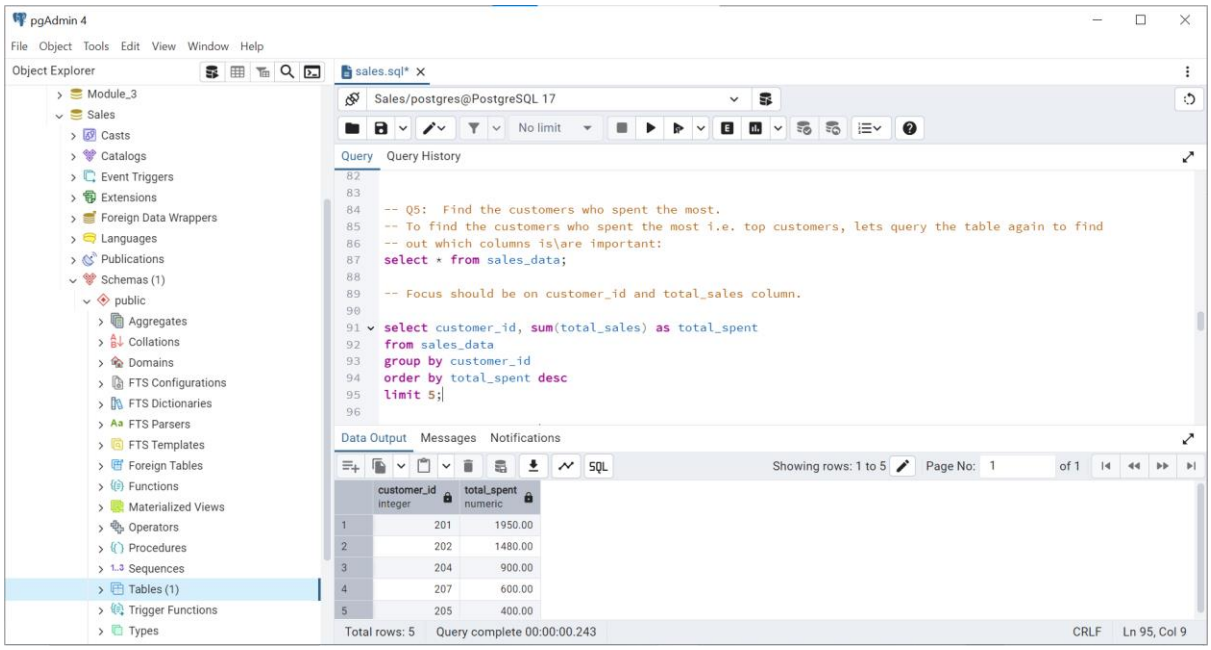

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
sales.sql*
Sales/postgres@PostgreSQL 17
No limit
Query Query History
20
21 -- Next l need to test or confirm if al the data have been uploaded unto the table successfully
22
23 SELECT * FROM sales_data LIMIT 10;
24
25 --Note: it worked successfully.
26
27
28
29
30
31
32
33
34
Data Output Messages Notifications
Showing rows: 1 to 10 Page No: 1 of 1
order_id [PK] integer
customer_id integer
order_date date
product_name character varying (100)
category character varying (50)
quantity integer
price numeric (10,2)
total_sales numeric (10,2)
1 1001 201 2024-01-05 Laptop Electronics 1 1000.00 1000.00
2 1002 202 2024-01-07 Smartphone Electronics 2 500.00 1000.00
3 1003 203 2024-01-10 Headphones Accessories 1 100.00 100.00
4 1004 201 2024-02-01 Laptop Electronics 1 950.00 950.00
5 1005 204 2024-02-15 Tablet Electronics 3 300.00 900.00
Total rows: 10 Query complete 00:00:00.274 CRLF Ln 28, Col 1
Module_3
Sales
Casts
Catalogs
Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (1)
Trigger Functions
Types

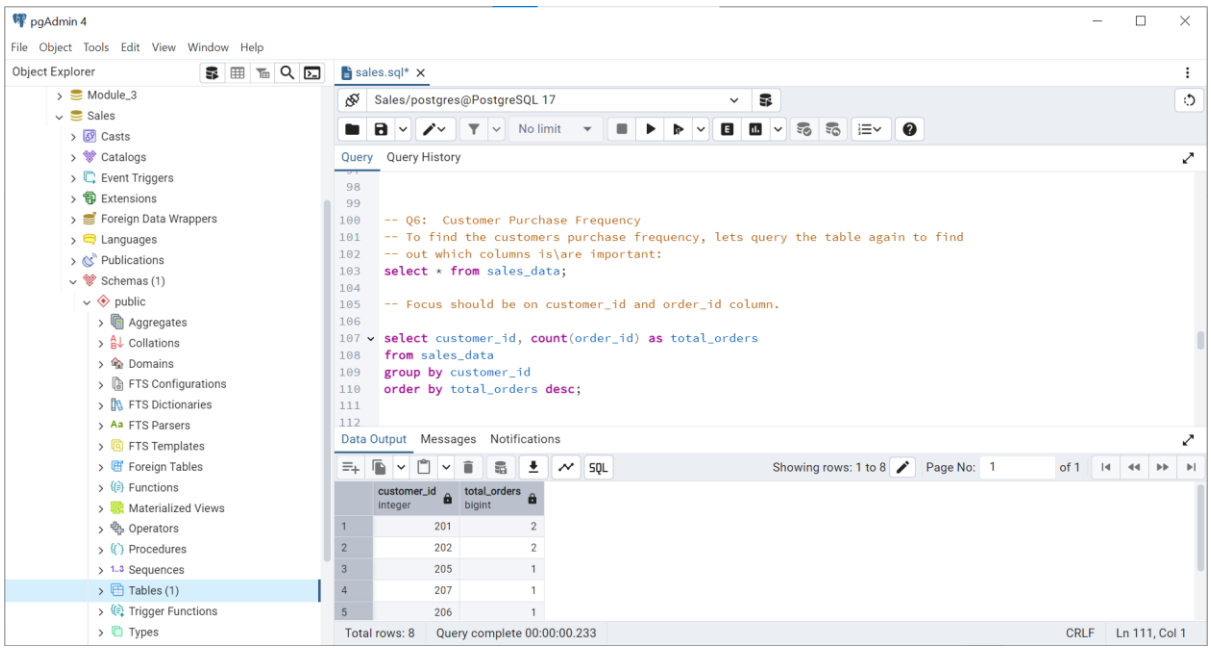

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
sales.sql*
Sales/postgres@PostgreSQL 17
No limit
Query Query History
--Next the business questions:
--Q1: Find out how much revenue each product generated.
-- To find out how much revenue i.e. total sales by product we need to focus on these two columns -
-- product_name and total_sales.
select sales_data.product_name, sum(total_sales) as total_revenue
from sales_data
group by product_name
order by total_revenue DESC;
Data Output Messages Notifications
Showing rows: 1 to 7 Page No: 1 of 1
product_name character varying (100)
total_revenue numeric
1 Laptop 1950.00
2 Smartphone 1480.00
3 Tablet 900.00
4 Camera 600.00
5 Headphones 460.00
Total rows: 7 Query complete 00:00:00.292
CRLF Ln 40, Col 1

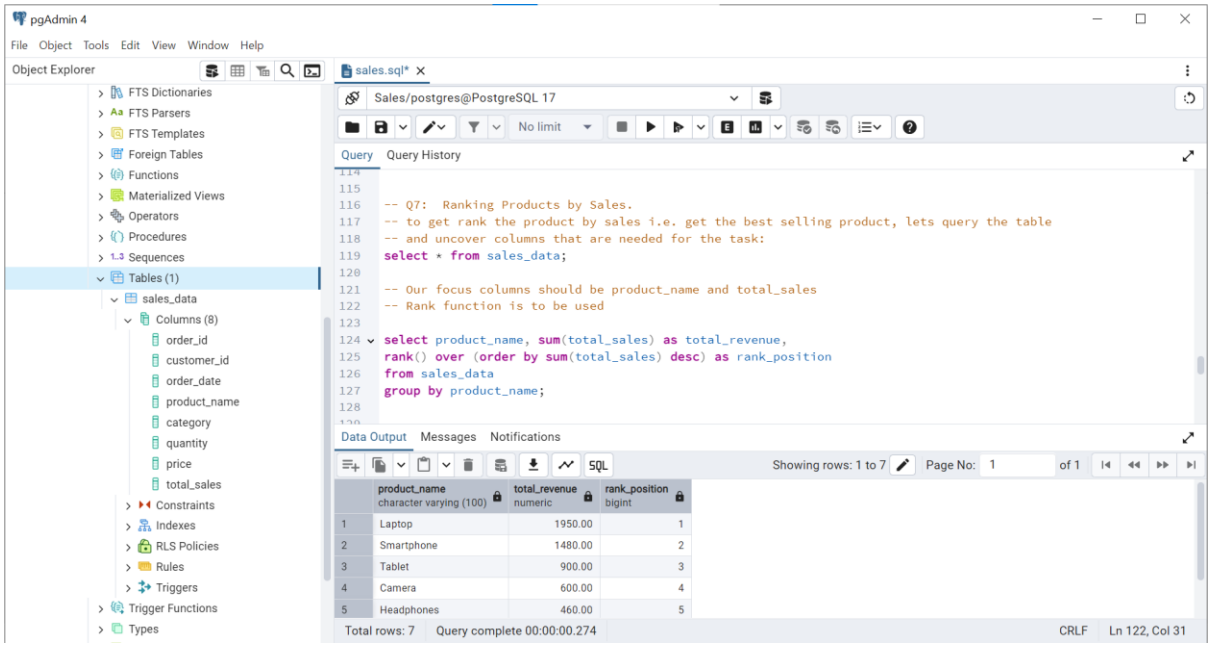

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
sales.sql*
Sales/postgres@PostgreSQL 17
No limit
Query Query History
-- Q2: What is the best-selling Products (by Quantity)
-- To find the best selling product by quantity, lets query the table again to find out which columns
-- is\are important:
select * from sales_data;
-- Focus should be on product_name and quantity column
select product_name, sum(quantity) as total_units_sold
from sales_data
group by product_name
order by total_units_sold desc;
Data Output Messages Notifications
Showing rows: 1 to 7 Page No: 1 of 1
product_name character varying (100)
total_units_sold bigint
1 Headphones 4
2 Tablet 3
3 Smartphone 3
4 Laptop 2
5 Smartwatch 2
Total rows: 7 Query complete 00:00:00.143
CRLF Ln 53, Col 1

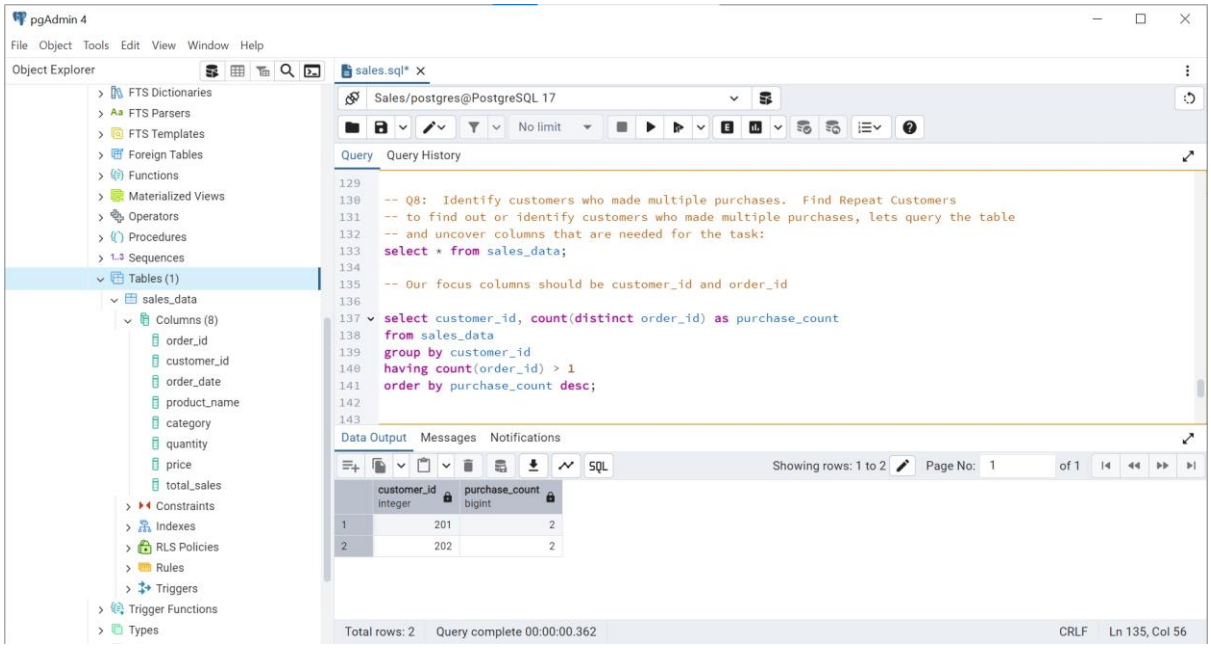

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
sales.sql*
Sales/postgres@PostgreSQL 17
No limit
Query Query History
-- Q3: What is the Total Sales by Category
-- To find the total sales by category, lets query the table again to find out which columns
-- is\are important:
select * from sales_data;
-- Focus should be on category and total_sales column
select category, sum(total_sales) as total_revenue
from sales_data
group by category
order by total_revenue desc;
Data Output Messages Notifications
Showing rows: 1 to 4 Page No: 1 of 1
category character varying (50)
total_revenue numeric
1 Electronics 4730.00
2 Photography 600.00
3 Accessories 460.00
4 Wearables 400.00
Total rows: 4 Query complete 00:00:00.200
CRLF Ln 66, Col 1

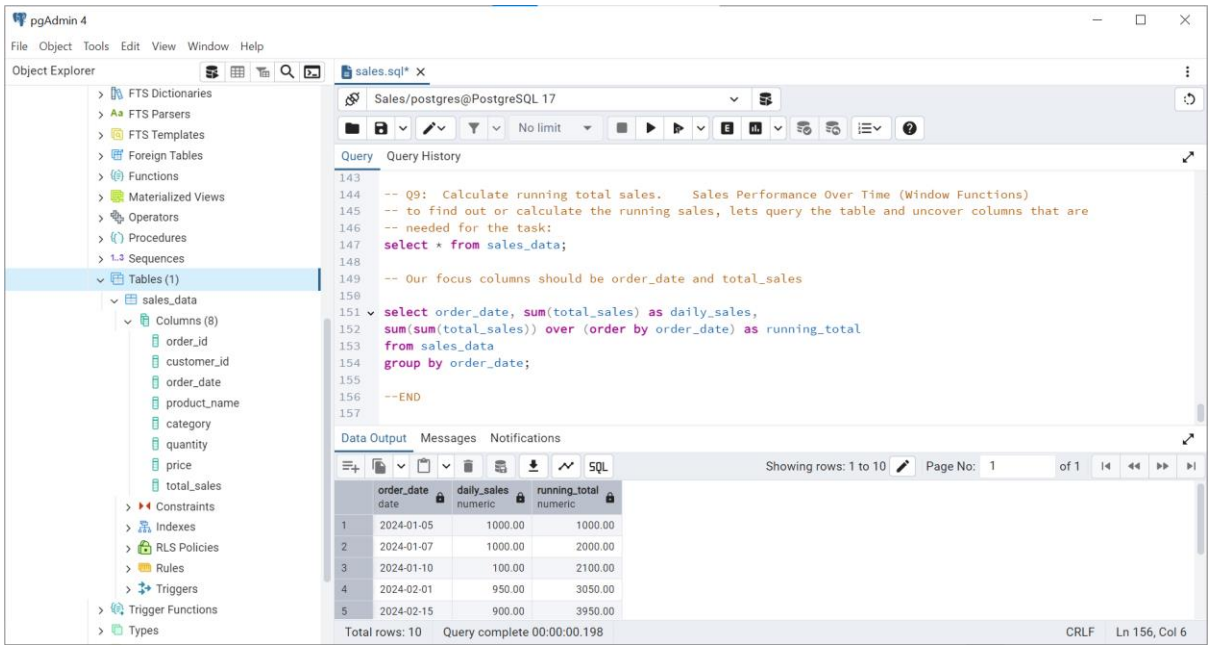

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
sales.sql*
Sales/postgres@PostgreSQL 17
No limit
Query Query History
-- Q4: Monthly Sales Trend. Analyze revenue trends over time.
-- To find the monthly sales trend, lets query the table again to find out which columns
-- is\are important:
select * from sales_data;
-- Focus should be on order_date and total_sales column.
-- In addition, the order_date must be formated to Year and month
select to_char(order_date, 'YYYY-MM') as month, sum(total_sales) as total_revenue
from sales_data
group by month
order by month;
Data Output Messages Notifications
Showing rows: 1 to 5 Page No: 1 of 1
month text | total_revenue numeric
1 2024-01 2100.00
2 2024-02 1850.00
3 2024-03 800.00
4 2024-04 1080.00
5 2024-05 360.00
Total rows: 5 Query complete 00:00:00.201 CRLF Ln 73, Col 57


pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
sales.sql*
Sales/postgres@PostgreSQL 17
No limit
Query Query History
-- Q5: Find the customers who spent the most.
-- To find the customers who spent the most i.e. top customers, lets query the table again to find
-- out which columns is\are important:
select * from sales_data;
-- Focus should be on customer_id and total_sales column.
select customer_id, sum(total_sales) as total_spent
from sales_data
group by customer_id
order by total_spent desc
limit 5;
Data Output Messages Notifications
Showing rows: 1 to 5 Page No: 1 of 1
customer_id integer | total_spent numeric
1 201 1950.00
2 202 1480.00
3 204 900.00
4 207 600.00
5 205 400.00
Total rows: 5 Query complete 00:00:00.243 CRLF Ln 95, Col 9


pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
sales.sql*
Sales/postgres@PostgreSQL 17
No limit
Query Query History
-- Q6: Customer Purchase Frequency
-- To find the customers purchase frequency, lets query the table again to find
-- out which columns is\are important:
select * from sales_data;
-- Focus should be on customer_id and order_id column.
select customer_id, count(order_id) as total_orders
from sales_data
group by customer_id
order by total_orders desc;
Data Output Messages Notifications
Showing rows: 1 to 8 Page No: 1 of 1
customer_id integer | total_orders bigint
1 201 2
2 202 2
3 205 1
4 207 1
5 206 1
Total rows: 8 Query complete 00:00:00.233 CRLF Ln 111, Col 1


pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
sales.sql*
Sales/postgres@PostgreSQL 17
No limit
Query Query History
-- Q7: Ranking Products by Sales.
-- to get rank the product by sales i.e. get the best selling product, lets query the table
-- and uncover columns that are needed for the task:
select * from sales_data;
-- Our focus columns should be product_name and total_sales
-- Rank function is to be used
select product_name, sum(total_sales) as total_revenue,
rank() over (order by sum(total_sales) desc) as rank_position
from sales_data
group by product_name;
Data Output Messages Notifications
Showing rows: 1 to 7 Page No: 1 of 1
product_name character varying (100) total_revenue numeric rank_position bigint
1 Laptop 1950.00 1
2 Smartphone 1480.00 2
3 Tablet 900.00 3
4 Camera 600.00 4
5 Headphones 460.00 5
Total rows: 7 Query complete 00:00:00.274 CRLF Ln 122, Col 31


pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
sales.sql*
Sales/postgres@PostgreSQL 17
No limit
Query Query History
-- Q8: Identify customers who made multiple purchases. Find Repeat Customers
-- to find out or identify customers who made multiple purchases, lets query the table
-- and uncover columns that are needed for the task:
select * from sales_data;
-- Our focus columns should be customer_id and order_id
select customer_id, count(distinct order_id) as purchase_count
from sales_data
group by customer_id
having count(order_id) > 1
order by purchase_count desc;
Data Output Messages Notifications
Showing rows: 1 to 2 Page No: 1 of 1
customer_id integer purchase_count bigint
1 201 2
2 202 2
Total rows: 2 Query complete 00:00:00.362 CRLF Ln 135, Col 56


pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
sales.sql*
Sales/postgres@PostgreSQL 17
No limit
Query Query History
-- Q9: Calculate running total sales. Sales Performance Over Time (Window Functions)
-- to find out or calculate the running sales, lets query the table and uncover columns that are
-- needed for the task:
select * from sales_data;
-- Our focus columns should be order_date and total_sales
select order_date, sum(total_sales) as daily_sales,
sum(sum(total_sales)) over (order by order_date) as running_total
from sales_data
group by order_date;
--END
Data Output Messages Notifications
Showing rows: 1 to 10 Page No: 1 of 1
order_date date daily_sales numeric running_total numeric
1 2024-01-05 1000.00 1000.00
2 2024-01-07 1000.00 2000.00
3 2024-01-10 100.00 2100.00
4 2024-02-01 950.00 3050.00
5 2024-02-15 900.00 3950.00
Total rows: 10 Query complete 00:00:00.198 CRLF Ln 156, Col 6