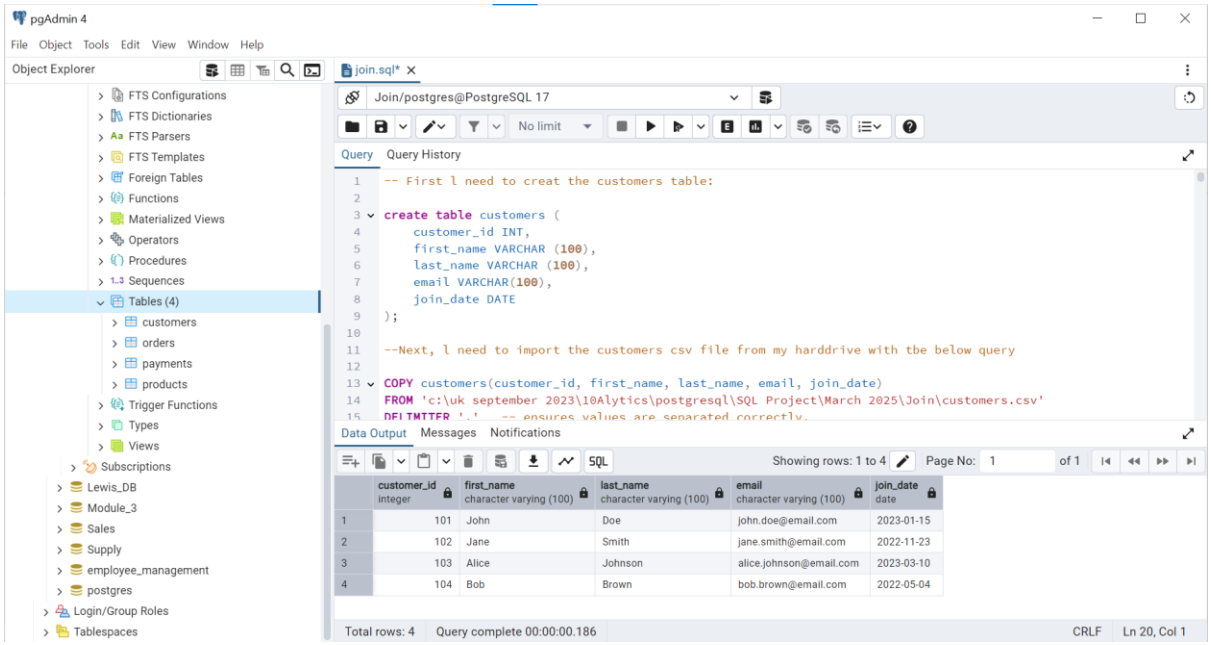
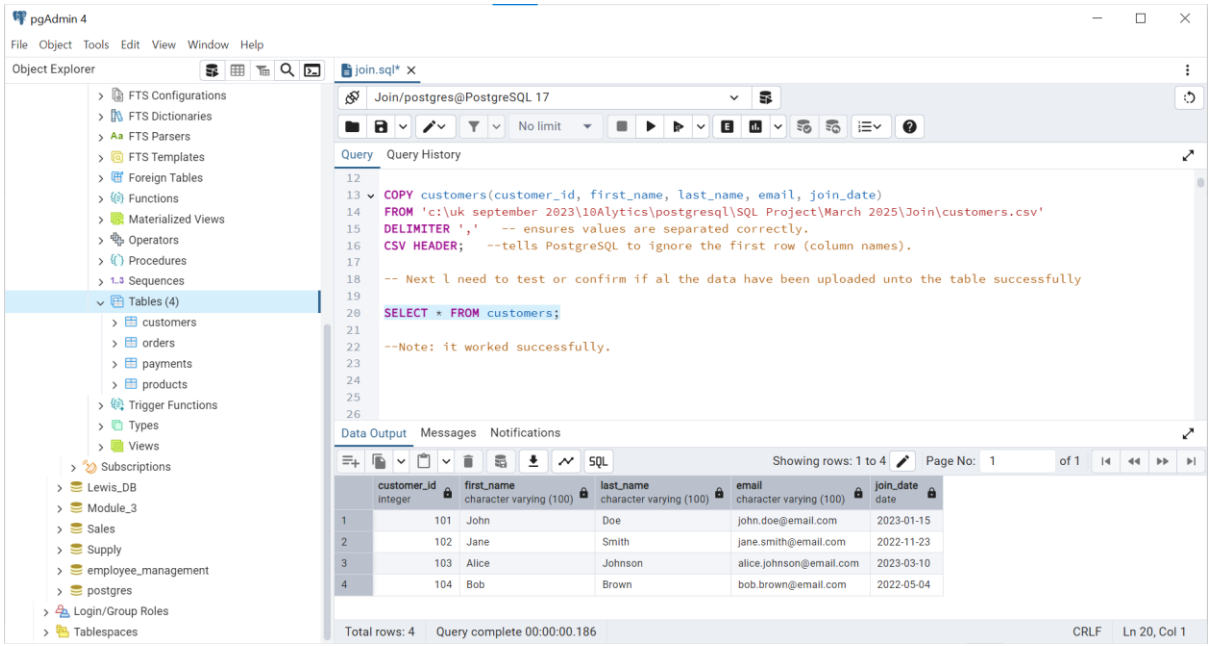
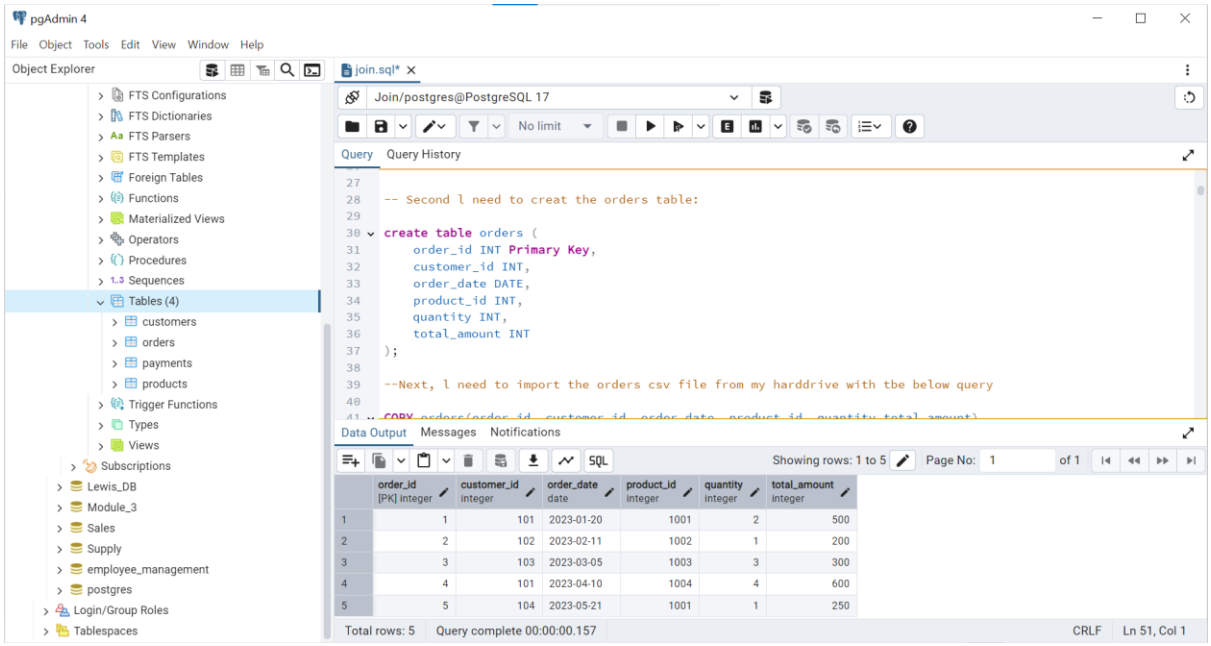
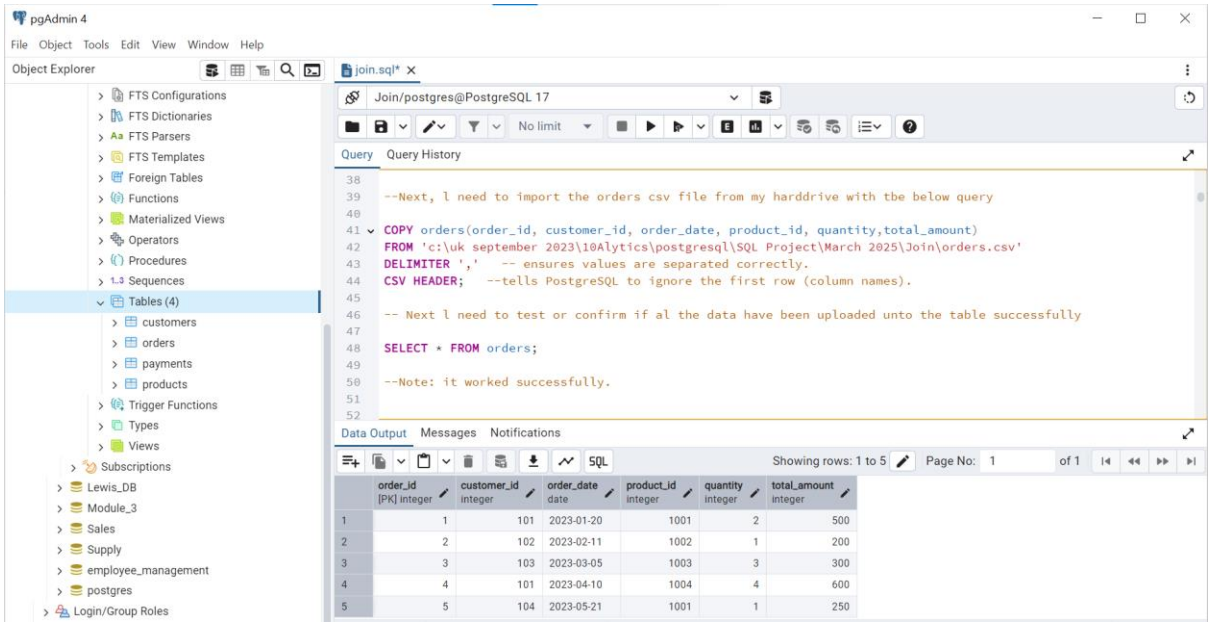
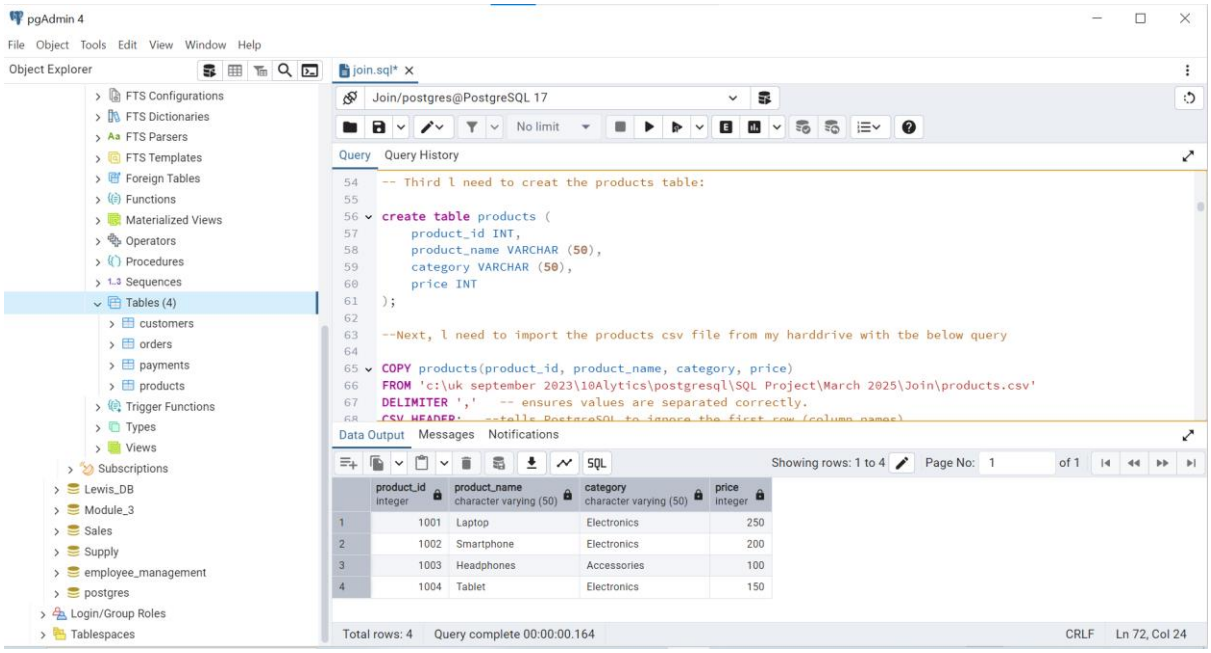
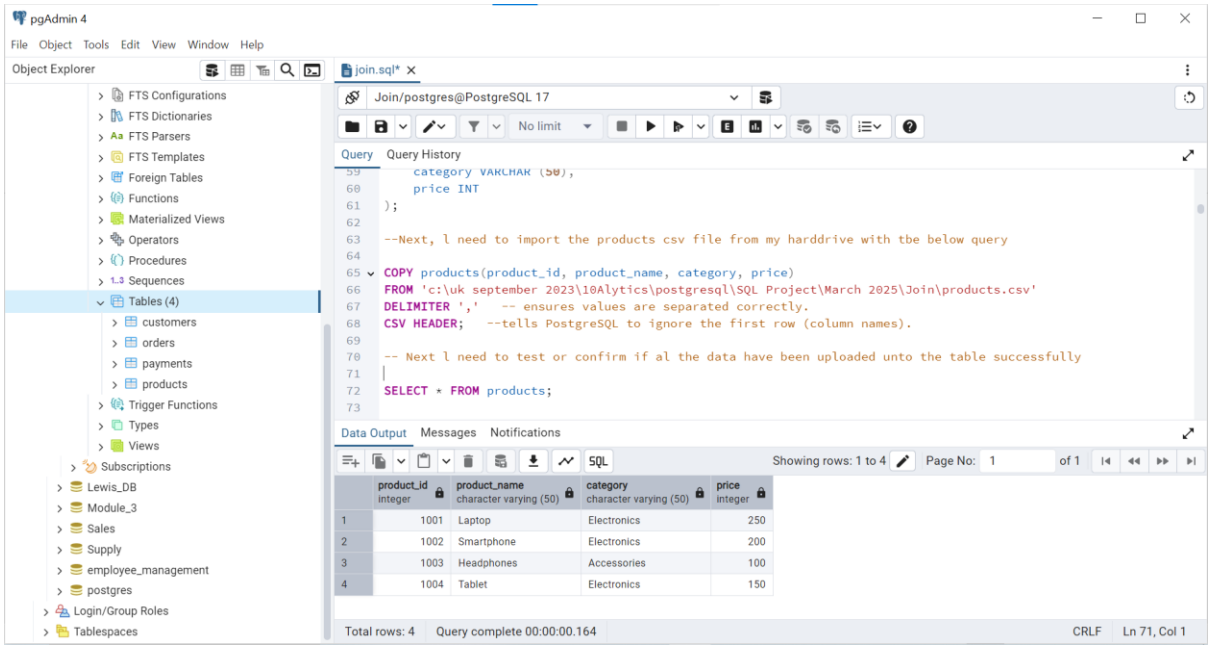

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query Query History
-- First l need to creat the customers table:
create table customers (
customer_id INT,
first_name VARCHAR (100),
last_name VARCHAR (100),
email VARCHAR(100),
join_date DATE
);
--Next, l need to import the customers csv file from my harddrive with tbe below query
COPY customers(customer_id, first_name, last_name, email, join_date)
FROM 'c:\uk september 2023\10Alytics\postgresql\SQL Project\March 2025\Join\customers.csv'
DELIMITER ','
Data Output Messages Notifications
Showing rows: 1 to 4
Page No: 1 of 1
customer_id integer
first_name character varying (100)
last_name character varying (100)
email character varying (100)
join_date date
1 101 John Doe john.doe@email.com 2023-01-15
2 102 Jane Smith jane.smith@email.com 2022-11-23
3 103 Alice Johnson alice.johnson@email.com 2023-03-10
4 104 Bob Brown bob.brown@email.com 2022-05-04
Total rows: 4
Query complete 00:00:00.186
CRLF
Ln 20, Col 1

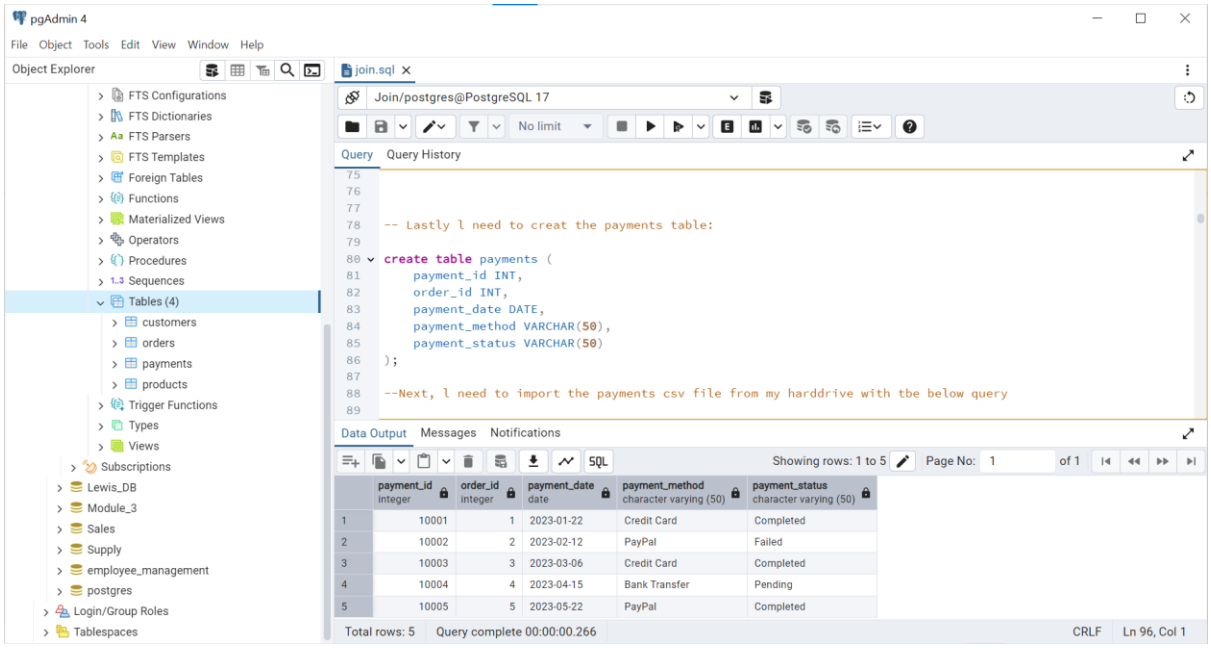

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query Query History
COPY customers(customer_id, first_name, last_name, email, join_date)
FROM 'c:\uk september 2023\10Alytics\postgresql\SQL Project\March 2025\Join\customers.csv'
DELIMITER ','
CSV HEADER;
--tells PostgreSQL to ignore the first row (column names).
-- Next l need to test or confirm if al the data have been uploaded unto the table successfully
SELECT * FROM customers;
--Note: it worked successfully.
Data Output Messages Notifications
Showing rows: 1 to 4
Page No: 1 of 1
customer_id integer
first_name character varying (100)
last_name character varying (100)
email character varying (100)
join_date date
1 101 John Doe john.doe@email.com 2023-01-15
2 102 Jane Smith jane.smith@email.com 2022-11-23
3 103 Alice Johnson alice.johnson@email.com 2023-03-10
4 104 Bob Brown bob.brown@email.com 2022-05-04
Total rows: 4
Query complete 00:00:00.186
CRLF
Ln 20, Col 1

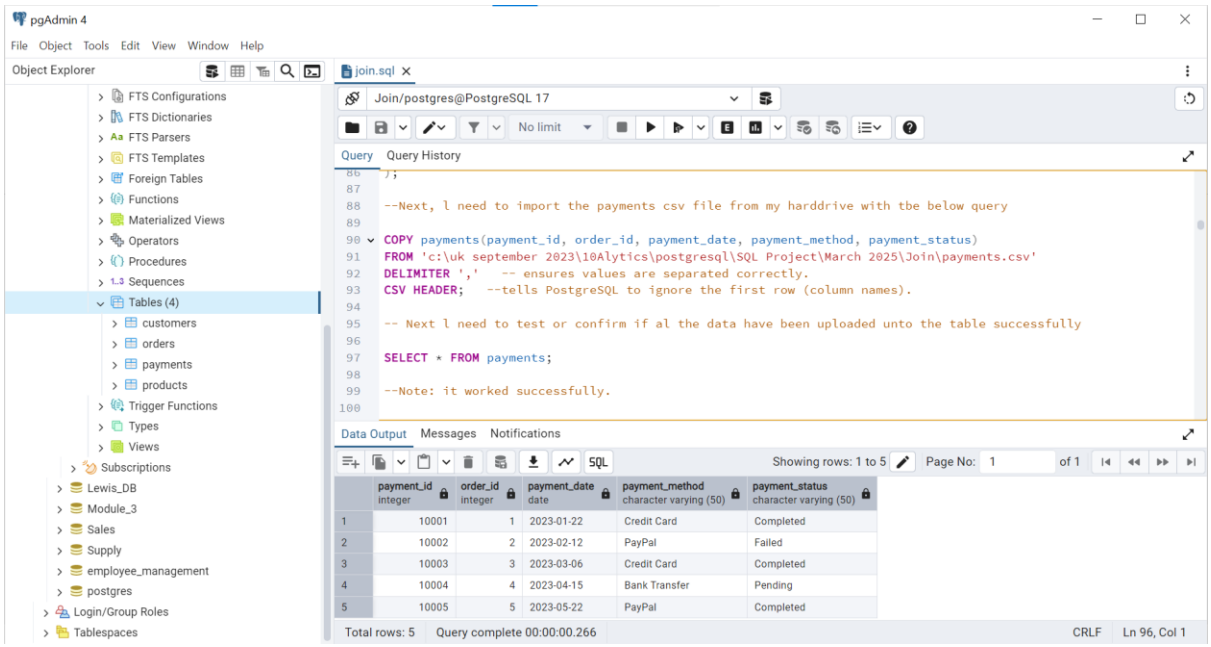

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query Query History
-- Second l need to creat the orders table:
create table orders (
order_id INT Primary Key,
customer_id INT,
order_date DATE,
product_id INT,
quantity INT,
total_amount INT
);
--Next, l need to import the orders csv file from my harddrive with tbe below query
Data Output Messages Notifications
Showing rows: 1 to 5
Page No: 1 of 1
order_id [PK] integer
customer_id integer
order_date date
product_id integer
quantity integer
total_amount integer
1 1 101 2023-01-20 1001 2 500
2 2 102 2023-02-11 1002 1 200
3 3 103 2023-03-05 1003 3 300
4 4 101 2023-04-10 1004 4 600
5 5 104 2023-05-21 1001 1 250
Total rows: 5
Query complete 00:00:00.157
CRLF
Ln 51, Col 1

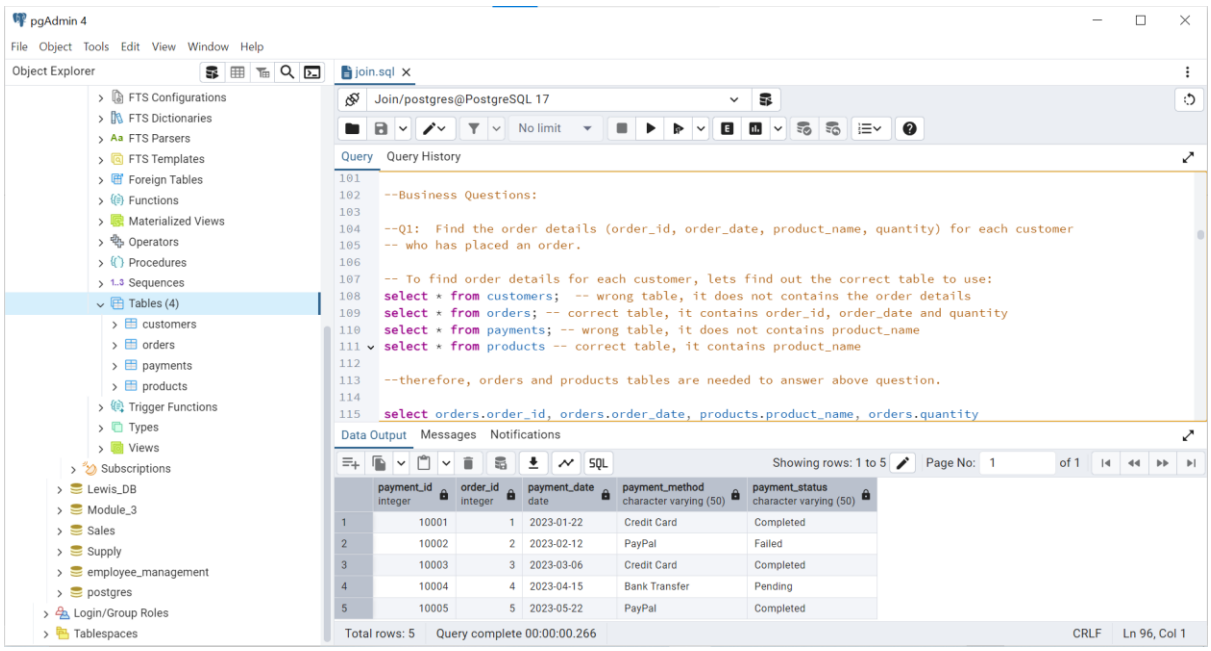

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query Query History
--Next, l need to import the orders csv file from my harddrive with tbe below query
COPY orders(order_id, customer_id, order_date, product_id, quantity,total_amount)
FROM 'c:\uk september 2023\10Alytics\postgresql\SQL Project\March 2025\Join\orders.csv'
DELIMITER ',' -- ensures values are separated correctly.
CSV HEADER; --tells PostgreSQL to ignore the first row (column names).
-- Next l need to test or confirm if al the data have been uploaded unto the table successfully
SELECT * FROM orders;
--Note: it worked successfully.
Data Output Messages Notifications
Showing rows: 1 to 5 Page No: 1 of 1
order_id [PK] integer customer_id integer order_date date product_id integer quantity integer total_amount integer
1 1 101 2023-01-20 1001 2 500
2 2 102 2023-02-11 1002 1 200
3 3 103 2023-03-05 1003 3 300
4 4 101 2023-04-10 1004 4 600
5 5 104 2023-05-21 1001 1 250
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (4)
customers
orders
payments
products
Trigger Functions
Types
Views
Subscriptions
Lewis_DB
Module_3
Sales
Supply
employee_management
postgres
Login/Group Roles

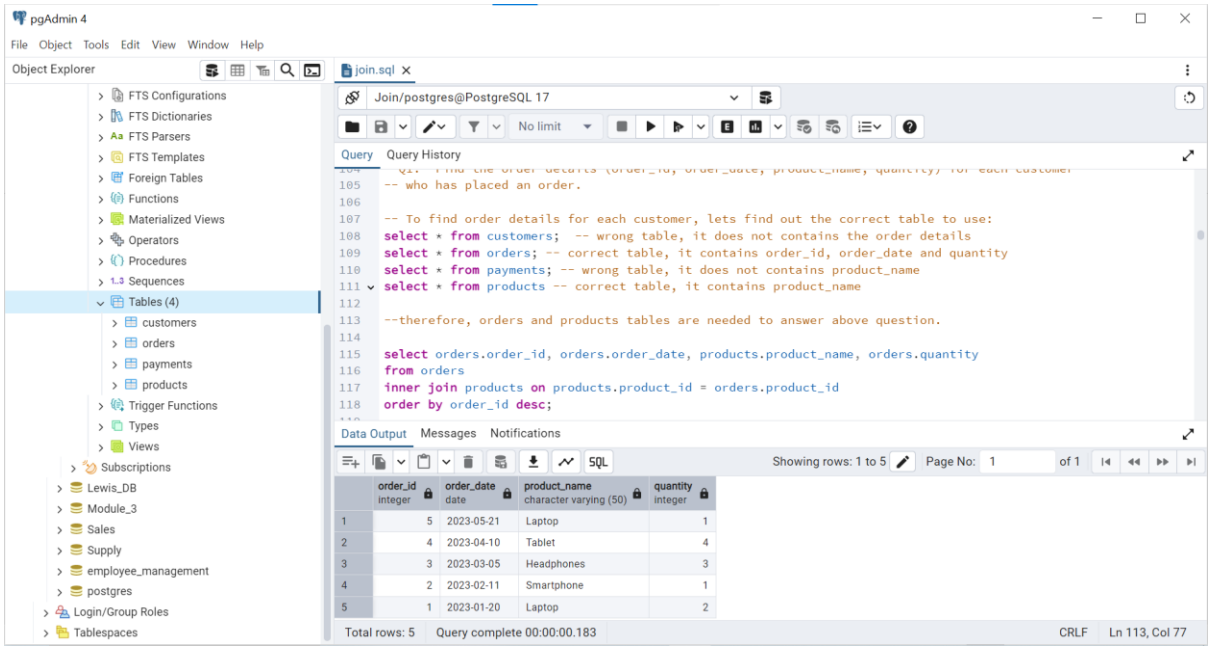

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query Query History
-- Third l need to creat the products table:
create table products (
product_id INT,
product_name VARCHAR (50),
category VARCHAR (50),
price INT
);
--Next, l need to import the products csv file from my harddrive with tbe below query
COPY products(product_id, product_name, category, price)
FROM 'c:\uk september 2023\10Alytics\postgresql\SQL Project\March 2025\Join\products.csv'
DELIMITER ',' -- ensures values are separated correctly.
Data Output Messages Notifications
Showing rows: 1 to 4 Page No: 1 of 1
product_id integer product_name character varying (50) category character varying (50) price integer
1 1001 Laptop Electronics 250
2 1002 Smartphone Electronics 200
3 1003 Headphones Accessories 100
4 1004 Tablet Electronics 150
Total rows: 4 Query complete 00:00:00.164 CRLF Ln 72, Col 24
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (4)
customers
orders
payments
products
Trigger Functions
Types
Views
Subscriptions
Lewis_DB
Module_3
Sales
Supply
employee_management
postgres
Login/Group Roles
Tablespaces

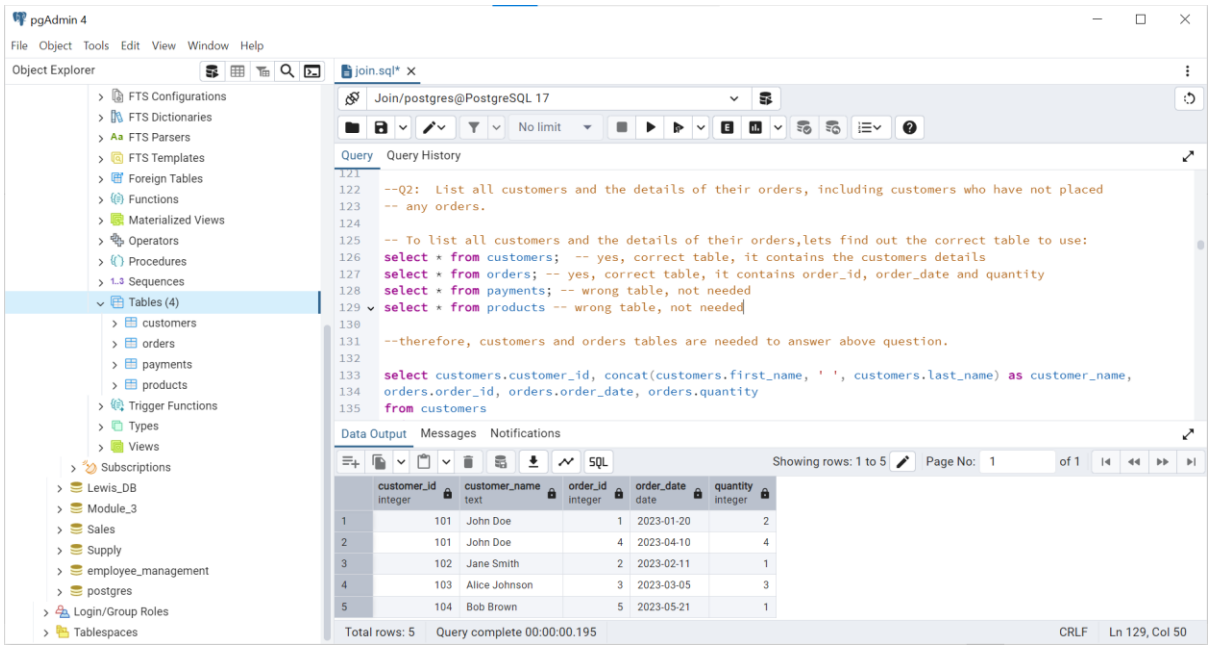

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query Query History
category VARCHAR (50),
price INT
);
--Next, l need to import the products csv file from my harddrive with tbe below query
COPY products(product_id, product_name, category, price)
FROM 'c:\uk september 2023\10Alytics\postgresql\SQL Project\March 2025\Join\products.csv'
DELIMITER ',' -- ensures values are separated correctly.
CSV HEADER; --tells PostgreSQL to ignore the first row (column names).
-- Next l need to test or confirm if al the data have been uploaded unto the table successfully
SELECT * FROM products;
Data Output Messages Notifications
Showing rows: 1 to 4 Page No: 1 of 1
product_id integer product_name character varying (50) category character varying (50) price integer
1 1001 Laptop Electronics 250
2 1002 Smartphone Electronics 200
3 1003 Headphones Accessories 100
4 1004 Tablet Electronics 150
Total rows: 4 Query complete 00:00:00.164 CRLF Ln 71, Col 1
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (4)
customers
orders
payments
products
Trigger Functions
Types
Views
Subscriptions
Lewis_DB
Module_3
Sales
Supply
employee_management
postgres
Login/Group Roles
Tablespaces

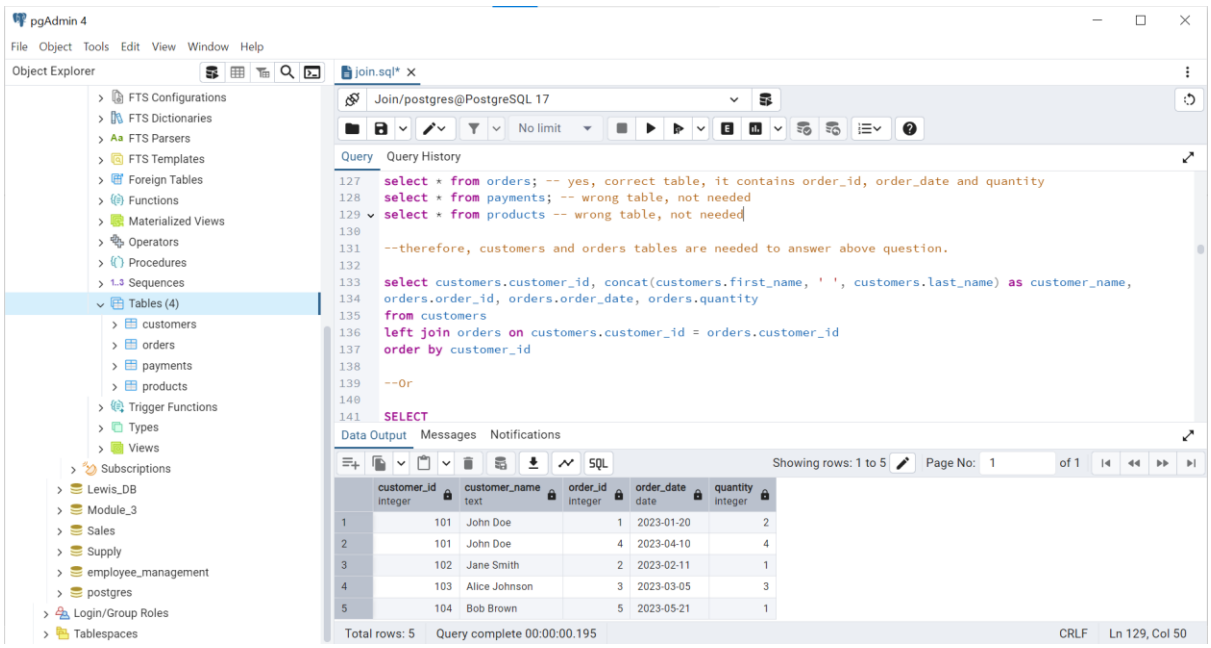

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql
Join/postgres@PostgreSQL 17
No limit
Query Query History
-- Lastly l need to creat the payments table:
create table payments (
payment_id INT,
order_id INT,
payment_date DATE,
payment_method VARCHAR(50),
payment_status VARCHAR(50)
);
--Next, l need to import the payments csv file from my harddrive with tbe below query
Data Output Messages Notifications
Showing rows: 1 to 5 Page No: 1 of 1
payment_id integer | order_id integer | payment_date date | payment_method character varying (50) | payment_status character varying (50)
1 10001 1 2023-01-22 Credit Card Completed
2 10002 2 2023-02-12 PayPal Failed
3 10003 3 2023-03-06 Credit Card Completed
4 10004 4 2023-04-15 Bank Transfer Pending
5 10005 5 2023-05-22 PayPal Completed
Total rows: 5 Query complete 00:00:00.266 CRLF Ln 96, Col 1

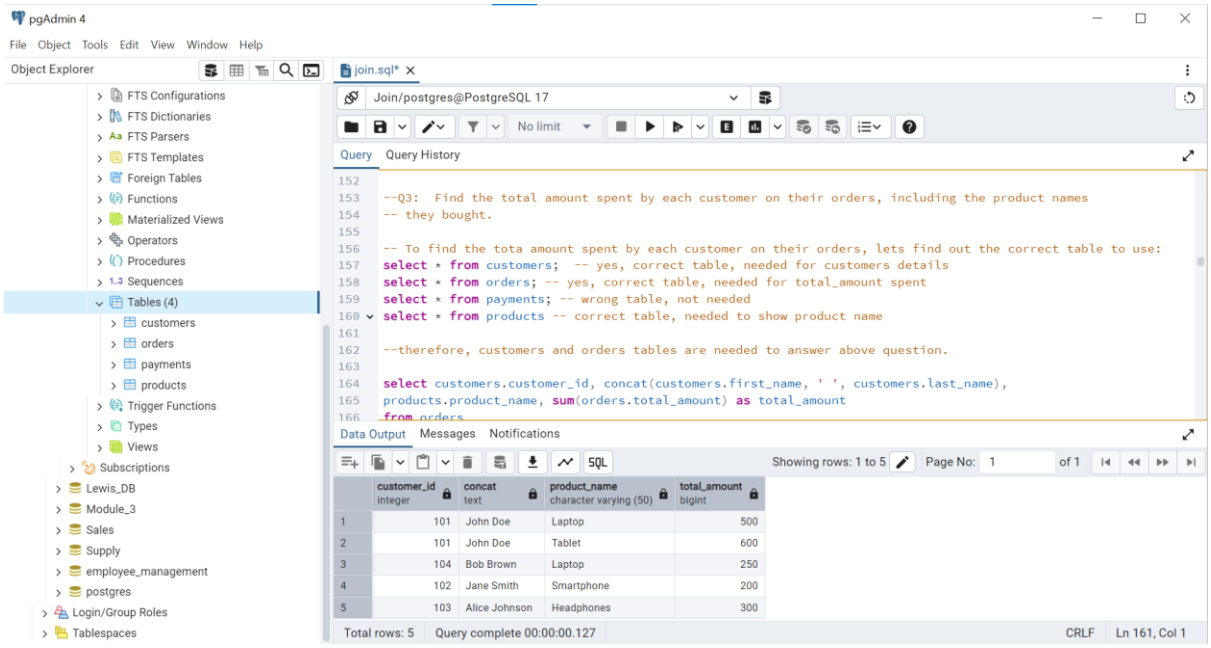

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql
Join/postgres@PostgreSQL 17
No limit
Query Query History
--Next, l need to import the payments csv file from my harddrive with tbe below query
COPY payments(payment_id, order_id, payment_date, payment_method, payment_status)
FROM 'c:\uk september 2023\10Alytics\postgresql\SQL Project\March 2025\Join\payments.csv'
DELIMITER ',' -- ensures values are separated correctly.
CSV HEADER; --tells PostgreSQL to ignore the first row (column names).
-- Next l need to test or confirm if al the data have been uploaded unto the table successfully
SELECT * FROM payments;
--Note: it worked successfully.
Data Output Messages Notifications
Showing rows: 1 to 5 Page No: 1 of 1
payment_id integer | order_id integer | payment_date date | payment_method character varying (50) | payment_status character varying (50)
1 10001 1 2023-01-22 Credit Card Completed
2 10002 2 2023-02-12 PayPal Failed
3 10003 3 2023-03-06 Credit Card Completed
4 10004 4 2023-04-15 Bank Transfer Pending
5 10005 5 2023-05-22 PayPal Completed
Total rows: 5 Query complete 00:00:00.266 CRLF Ln 96, Col 1

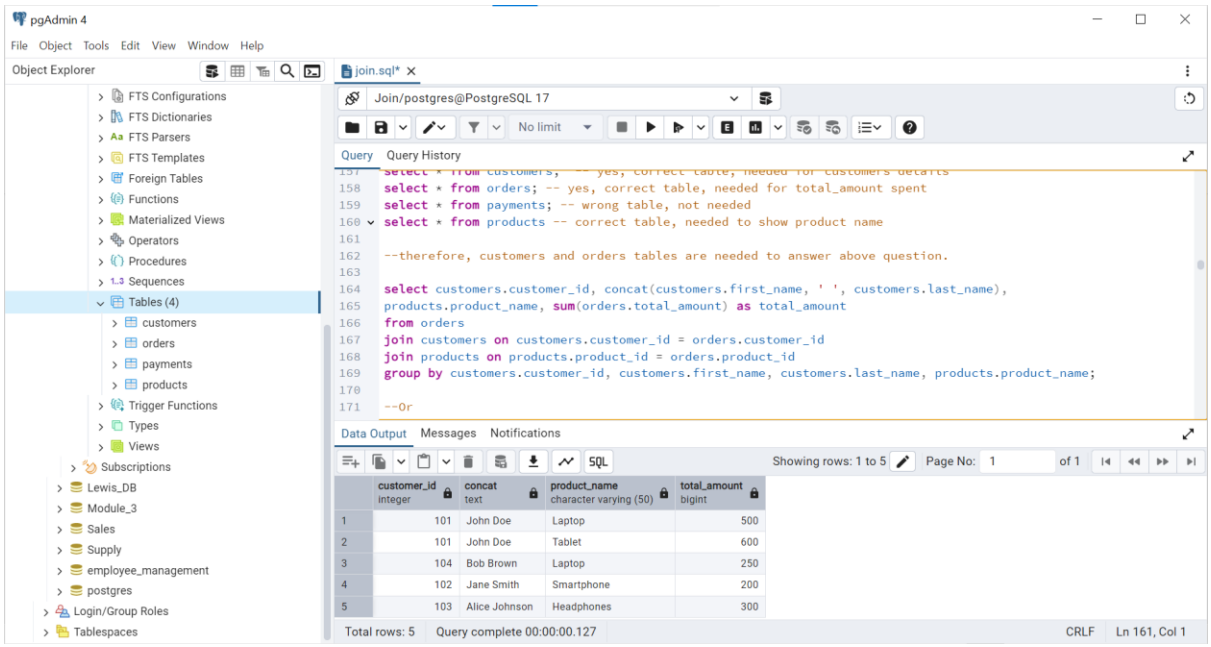

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql
Join/postgres@PostgreSQL 17
No limit
Query Query History
--Business Questions:
--Q1: Find the order details (order_id, order_date, product_name, quantity) for each customer
-- who has placed an order.
-- To find order details for each customer, lets find out the correct table to use:
select * from customers; -- wrong table, it does not contains the order details
select * from orders; -- correct table, it contains order_id, order_date and quantity
select * from payments; -- wrong table, it does not contains product_name
select * from products -- correct table, it contains product_name
--therefore, orders and products tables are needed to answer above question.
select orders.order_id, orders.order_date, products.product_name, orders.quantity
Data Output Messages Notifications
Showing rows: 1 to 5 Page No: 1 of 1
payment_id integer | order_id integer | payment_date date | payment_method character varying (50) | payment_status character varying (50)
1 10001 1 2023-01-22 Credit Card Completed
2 10002 2 2023-02-12 PayPal Failed
3 10003 3 2023-03-06 Credit Card Completed
4 10004 4 2023-04-15 Bank Transfer Pending
5 10005 5 2023-05-22 PayPal Completed
Total rows: 5 Query complete 00:00:00.266 CRLF Ln 96, Col 1

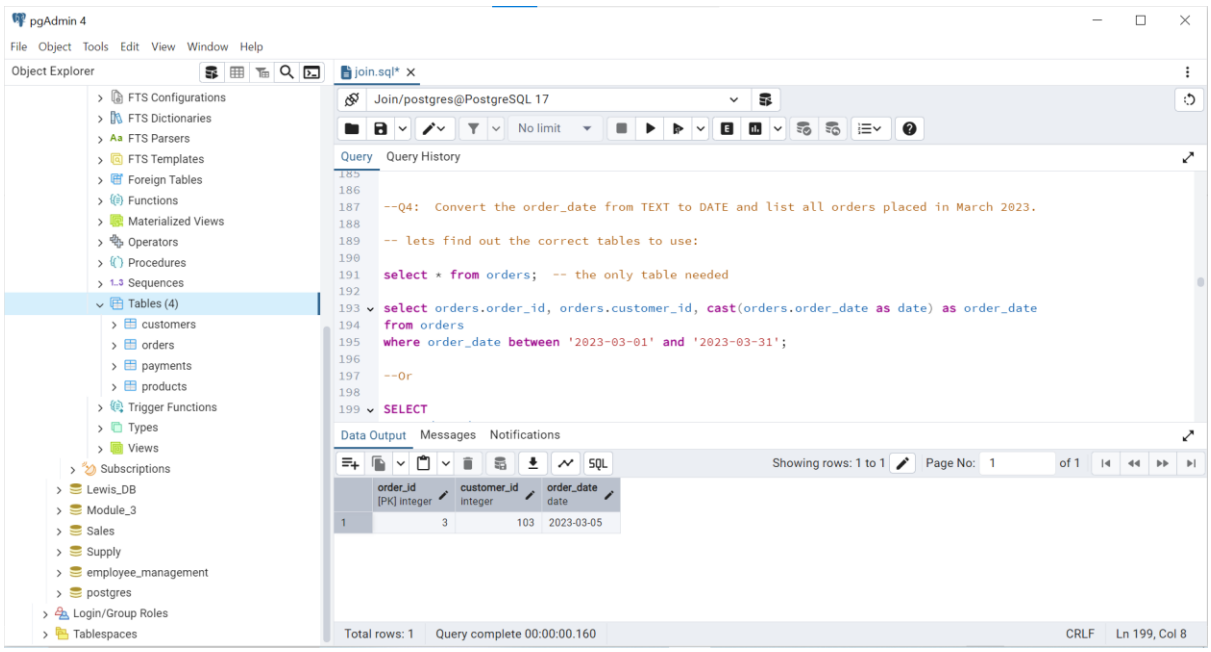

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql
Join/postgres@PostgreSQL 17
No limit
Query Query History
-- who has placed an order.
-- To find order details for each customer, lets find out the correct table to use:
select * from customers; -- wrong table, it does not contains the order details
select * from orders; -- correct table, it contains order_id, order_date and quantity
select * from payments; -- wrong table, it does not contains product_name
select * from products -- correct table, it contains product_name
--therefore, orders and products tables are needed to answer above question.
select orders.order_id, orders.order_date, products.product_name, orders.quantity
from orders
inner join products on products.product_id = orders.product_id
order by order_id desc;
Data Output Messages Notifications
Showing rows: 1 to 5 Page No: 1 of 1
order_id integer
order_date date
product_name character varying (50)
quantity integer
1 5 2023-05-21 Laptop 1
2 4 2023-04-10 Tablet 4
3 3 2023-03-05 Headphones 3
4 2 2023-02-11 Smartphone 1
5 1 2023-01-20 Laptop 2
Total rows: 5 Query complete 00:00:00.183 CRLF Ln 113, Col 77
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (4)
customers
orders
payments
products
Trigger Functions
Types
Views
Subscriptions
Lewis_DB
Module_3
Sales
Supply
employee_management
postgres
Login/Group Roles
Tablespaces

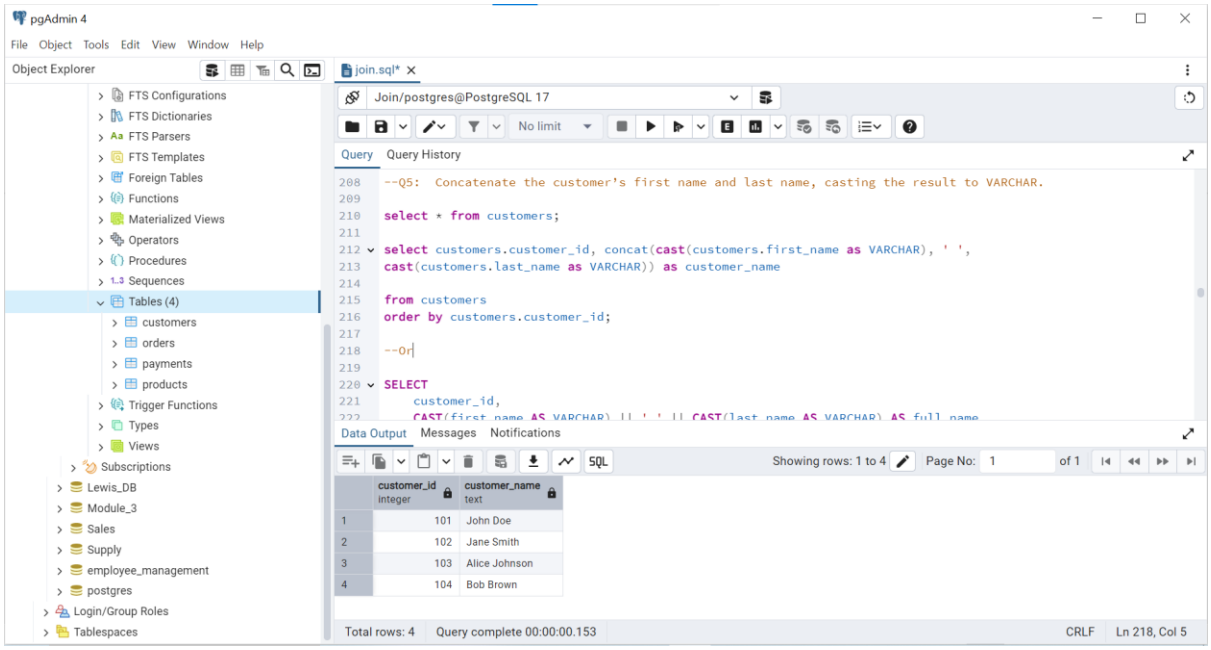

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query Query History
--Q2: List all customers and the details of their orders, including customers who have not placed
-- any orders.
-- To list all customers and the details of their orders,lets find out the correct table to use:
select * from customers; -- yes, correct table, it contains the customers details
select * from orders; -- yes, correct table, it contains order_id, order_date and quantity
select * from payments; -- wrong table, not needed
select * from products -- wrong table, not needed
--therefore, customers and orders tables are needed to answer above question.
select customers.customer_id, concat(customers.first_name, ' ', customers.last_name) as customer_name,
orders.order_id, orders.order_date, orders.quantity
from customers
Data Output Messages Notifications
Showing rows: 1 to 5 Page No: 1 of 1
customer_id integer
customer_name text
order_id integer
order_date date
quantity integer
1 101 John Doe 1 2023-01-20 2
2 101 John Doe 4 2023-04-10 4
3 102 Jane Smith 2 2023-02-11 1
4 103 Alice Johnson 3 2023-03-05 3
5 104 Bob Brown 5 2023-05-21 1
Total rows: 5 Query complete 00:00:00.195 CRLF Ln 129, Col 50
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (4)
customers
orders
payments
products
Trigger Functions
Types
Views
Subscriptions
Lewis_DB
Module_3
Sales
Supply
employee_management
postgres
Login/Group Roles
Tablespaces

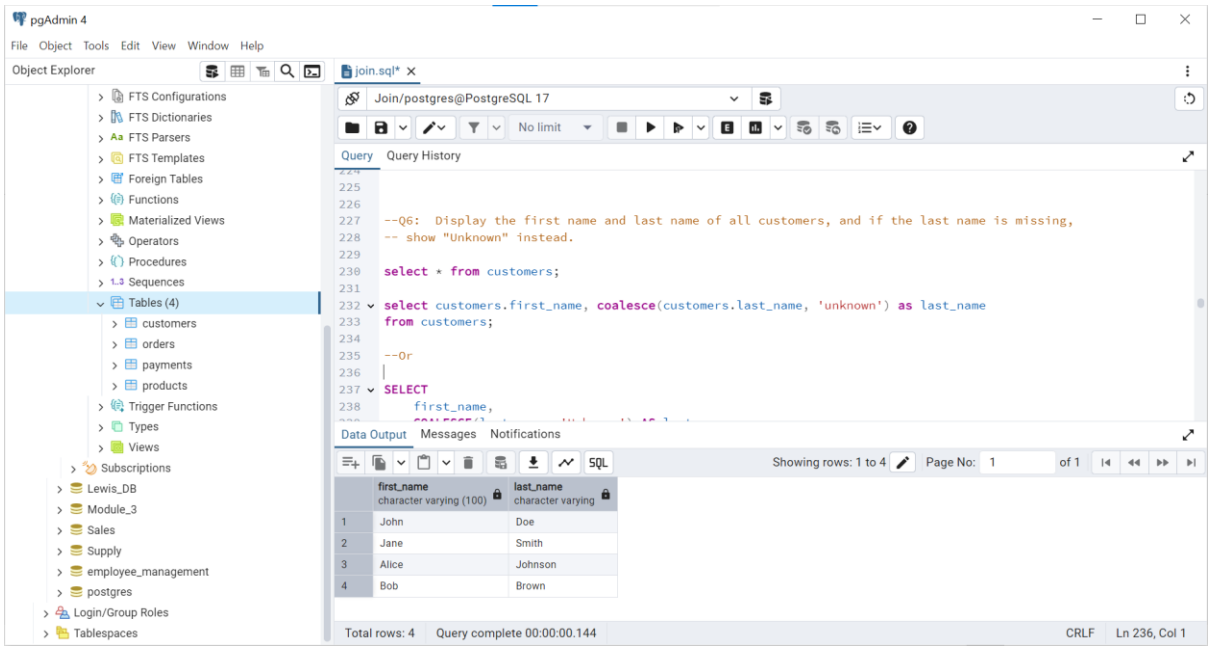

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query Query History
select * from orders; -- yes, correct table, it contains order_id, order_date and quantity
select * from payments; -- wrong table, not needed
select * from products -- wrong table, not needed
--therefore, customers and orders tables are needed to answer above question.
select customers.customer_id, concat(customers.first_name, ' ', customers.last_name) as customer_name,
orders.order_id, orders.order_date, orders.quantity
from customers
left join orders on customers.customer_id = orders.customer_id
order by customer_id
--Or
SELECT
Data Output Messages Notifications
Showing rows: 1 to 5 Page No: 1 of 1
customer_id integer
customer_name text
order_id integer
order_date date
quantity integer
1 101 John Doe 1 2023-01-20 2
2 101 John Doe 4 2023-04-10 4
3 102 Jane Smith 2 2023-02-11 1
4 103 Alice Johnson 3 2023-03-05 3
5 104 Bob Brown 5 2023-05-21 1
Total rows: 5 Query complete 00:00:00.195 CRLF Ln 129, Col 50
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (4)
customers
orders
payments
products
Trigger Functions
Types
Views
Subscriptions
Lewis_DB
Module_3
Sales
Supply
employee_management
postgres
Login/Group Roles
Tablespaces

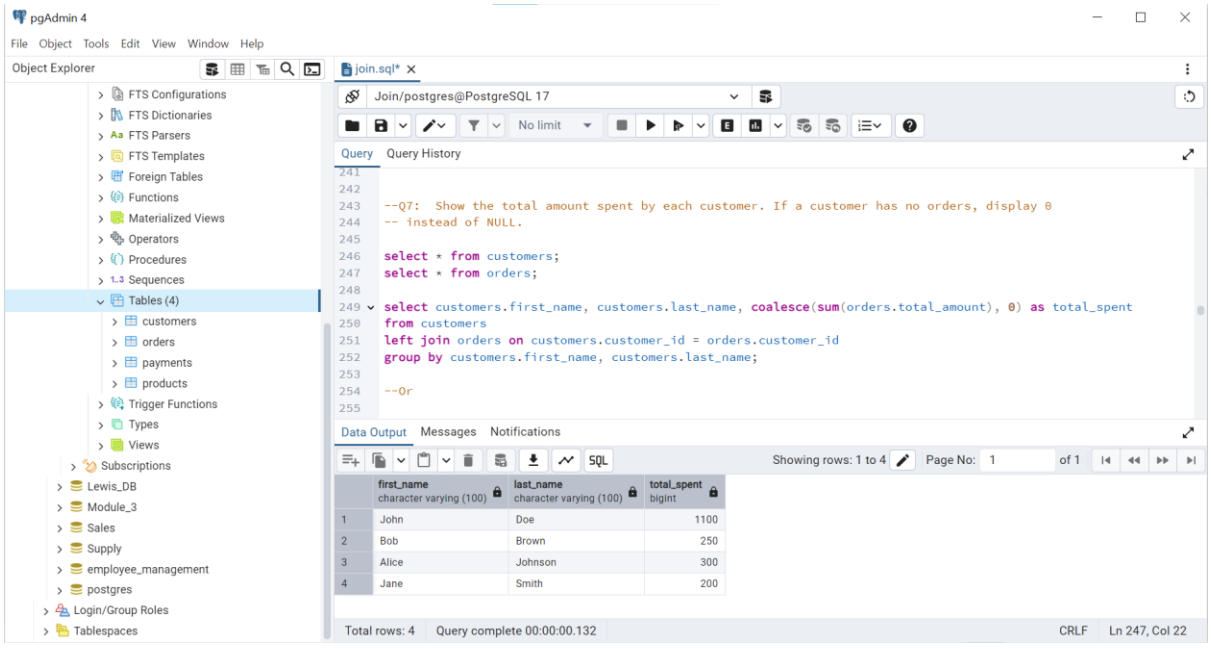

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query Query History
--Q3: Find the total amount spent by each customer on their orders, including the product names
-- they bought.
-- To find the tota amount spent by each customer on their orders, lets find out the correct table to use:
select * from customers; -- yes, correct table, needed for customers details
select * from orders; -- yes, correct table, needed for total_amount spent
select * from payments; -- wrong table, not needed
select * from products -- correct table, needed to show product name
--therefore, customers and orders tables are needed to answer above question.
select customers.customer_id, concat(customers.first_name, ' ', customers.last_name),
products.product_name, sum(orders.total_amount) as total_amount
from orders
Data Output Messages Notifications
Showing rows: 1 to 5 Page No: 1 of 1
customer_id integer
concat text
product_name character varying (50)
total_amount bigint
1 101 John Doe Laptop 500
2 101 John Doe Tablet 600
3 104 Bob Brown Laptop 250
4 102 Jane Smith Smartphone 200
5 103 Alice Johnson Headphones 300
Total rows: 5 Query complete 00:00:00.127 CRLF Ln 161, Col 1

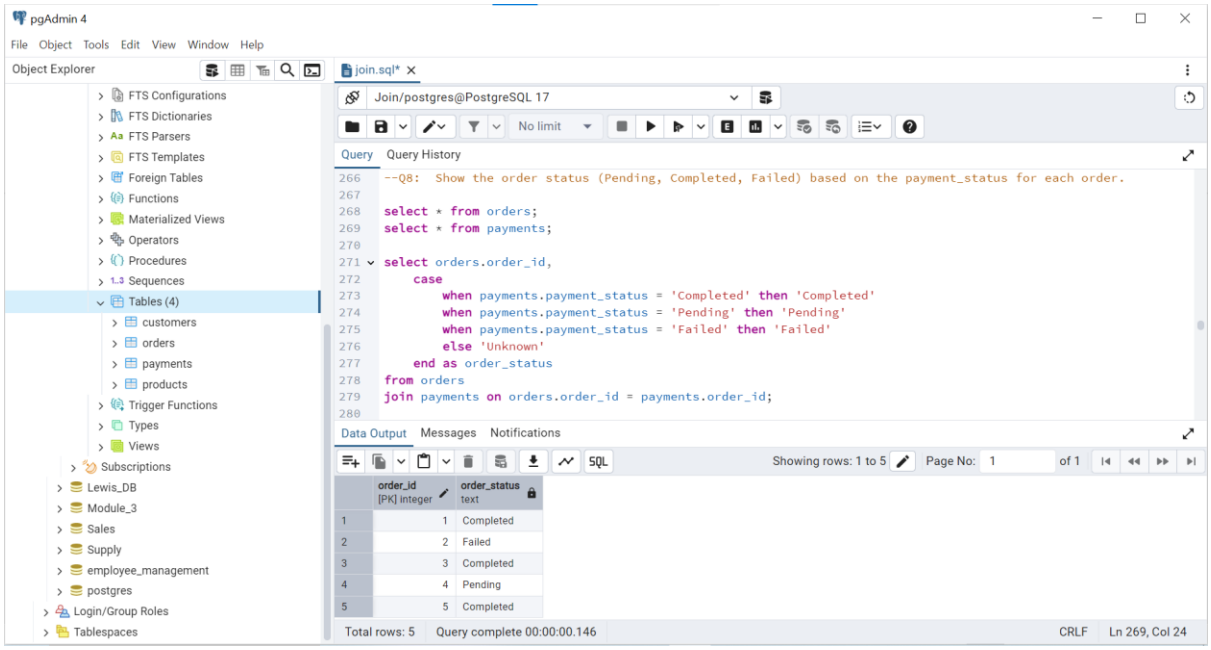

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query Query History
select * from orders; -- yes, correct table, needed for total_amount spent
select * from payments; -- wrong table, not needed
select * from products -- correct table, needed to show product name
--therefore, customers and orders tables are needed to answer above question.
select customers.customer_id, concat(customers.first_name, ' ', customers.last_name),
products.product_name, sum(orders.total_amount) as total_amount
from orders
join customers on customers.customer_id = orders.customer_id
join products on products.product_id = orders.product_id
group by customers.customer_id, customers.first_name, customers.last_name, products.product_name;
--Or
Data Output Messages Notifications
Showing rows: 1 to 5 Page No: 1 of 1
customer_id integer
concat text
product_name character varying (50)
total_amount bigint
1 101 John Doe Laptop 500
2 101 John Doe Tablet 600
3 104 Bob Brown Laptop 250
4 102 Jane Smith Smartphone 200
5 103 Alice Johnson Headphones 300
Total rows: 5 Query complete 00:00:00.127 CRLF Ln 161, Col 1

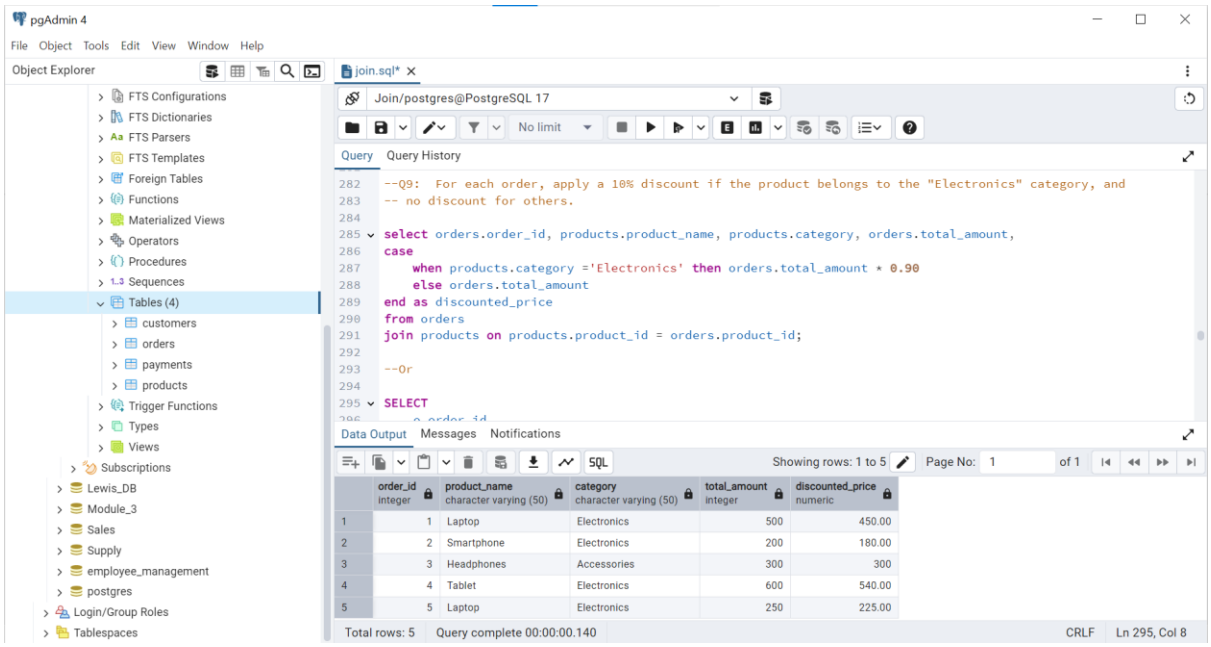

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query Query History
--Q4: Convert the order_date from TEXT to DATE and list all orders placed in March 2023.
-- lets find out the correct tables to use:
select * from orders; -- the only table needed
select orders.order_id, orders.customer_id, cast(orders.order_date as date) as order_date
from orders
where order_date between '2023-03-01' and '2023-03-31';
--Or
SELECT
Data Output Messages Notifications
Showing rows: 1 to 1 Page No: 1 of 1
order_id [PK] integer
customer_id integer
order_date date
1 3 103 2023-03-05
Total rows: 1 Query complete 00:00:00.160 CRLF Ln 199, Col 8

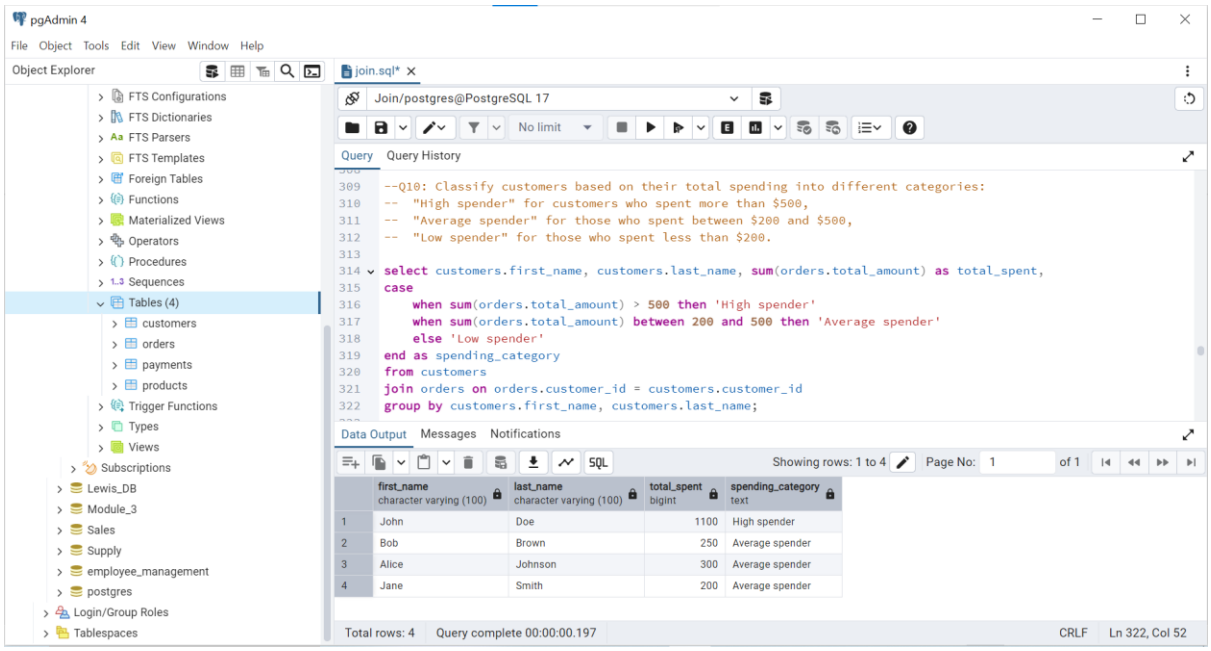

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query Query History
--Q5: Concatenate the customer's first name and last name, casting the result to VARCHAR.
select * from customers;
select customers.customer_id, concat(cast(customers.first_name as VARCHAR), ' ',
cast(customers.last_name as VARCHAR)) as customer_name
from customers
order by customers.customer_id;
--Or
SELECT
customer_id,
Data Output Messages Notifications
Showing rows: 1 to 4 Page No: 1 of 1
customer_id integer
customer_name text
1 101 John Doe
2 102 Jane Smith
3 103 Alice Johnson
4 104 Bob Brown
Total rows: 4 Query complete 00:00:00.153 CRLF Ln 218, Col 5

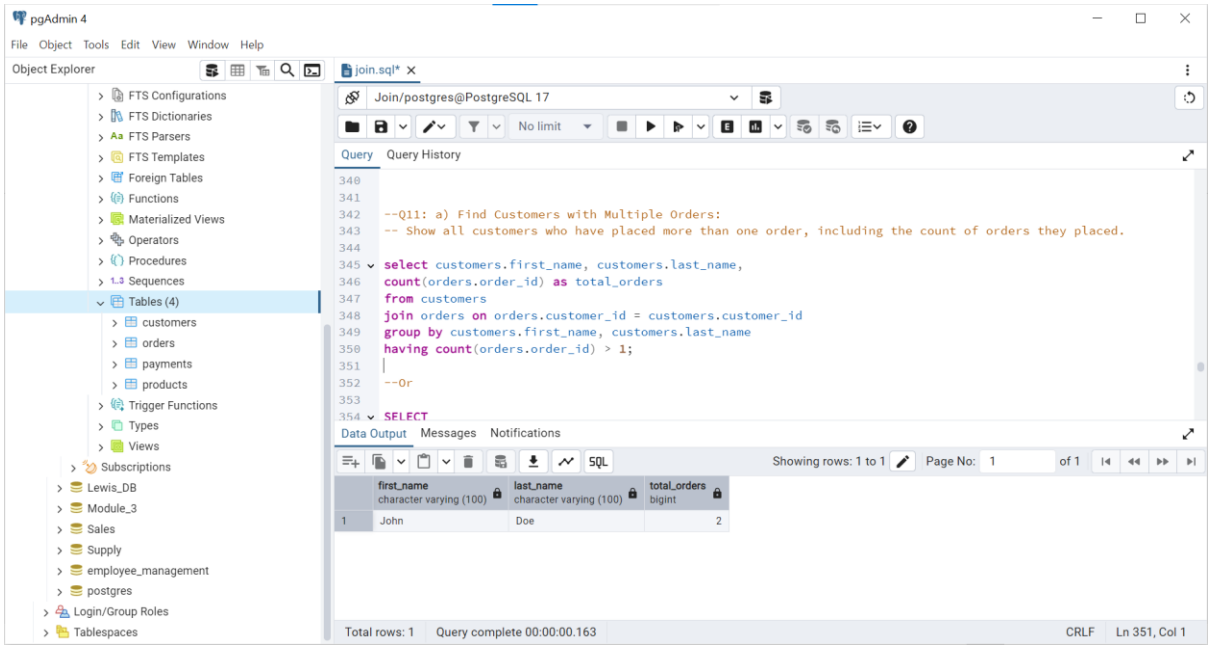

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query Query History
--Q6: Display the first name and last name of all customers, and if the last name is missing,
-- show "Unknown" instead.
select * from customers;
select customers.first_name, coalesce(customers.last_name, 'unknown') as last_name
from customers;
--Or
SELECT
first_name,
Data Output Messages Notifications
Showing rows: 1 to 4 Page No: 1 of 1
first_name character varying (100)
last_name character varying
1 John Doe
2 Jane Smith
3 Alice Johnson
4 Bob Brown
Total rows: 4 Query complete 00:00:00.144 CRLF Ln 236, Col 1

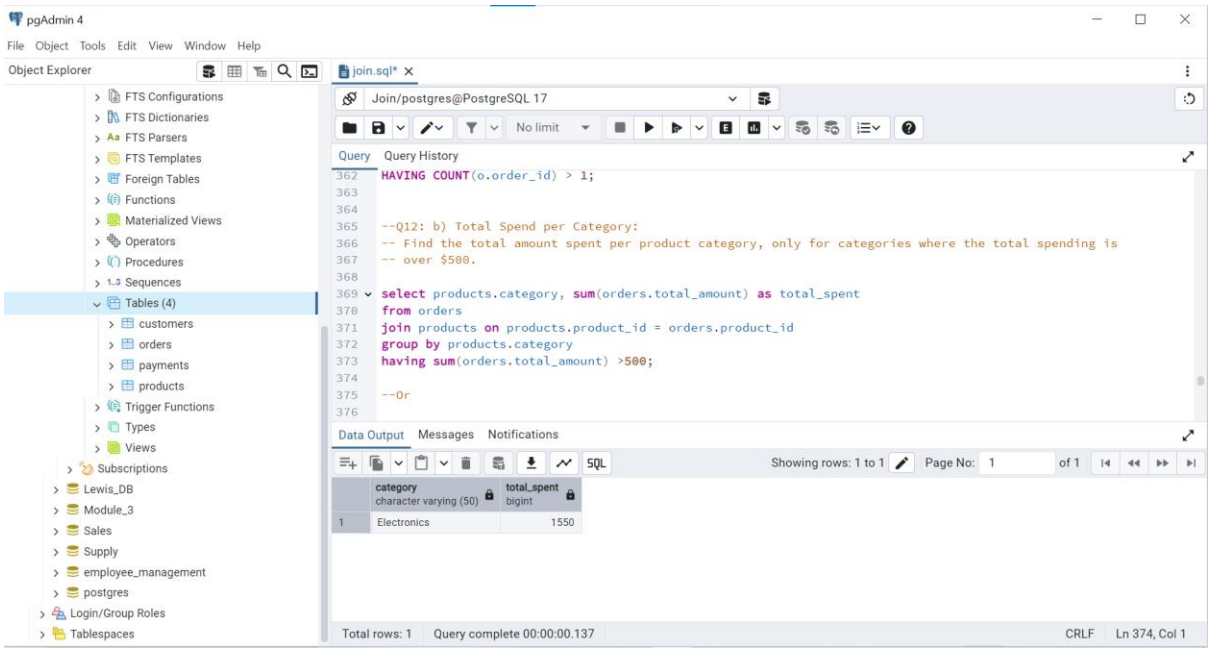

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query Query History
--Q7: Show the total amount spent by each customer. If a customer has no orders, display 0
-- instead of NULL.
select * from customers;
select * from orders;
select customers.first_name, customers.last_name, coalesce(sum(orders.total_amount), 0) as total_spent
from customers
left join orders on customers.customer_id = orders.customer_id
group by customers.first_name, customers.last_name;
--Or
Data Output Messages Notifications
Showing rows: 1 to 4 Page No: 1 of 1
first_name character varying (100)
last_name character varying (100)
total_spent bigint
1 John Doe 1100
2 Bob Brown 250
3 Alice Johnson 300
4 Jane Smith 200
Total rows: 4 Query complete 00:00:00.132 CRLF Ln 247, Col 22

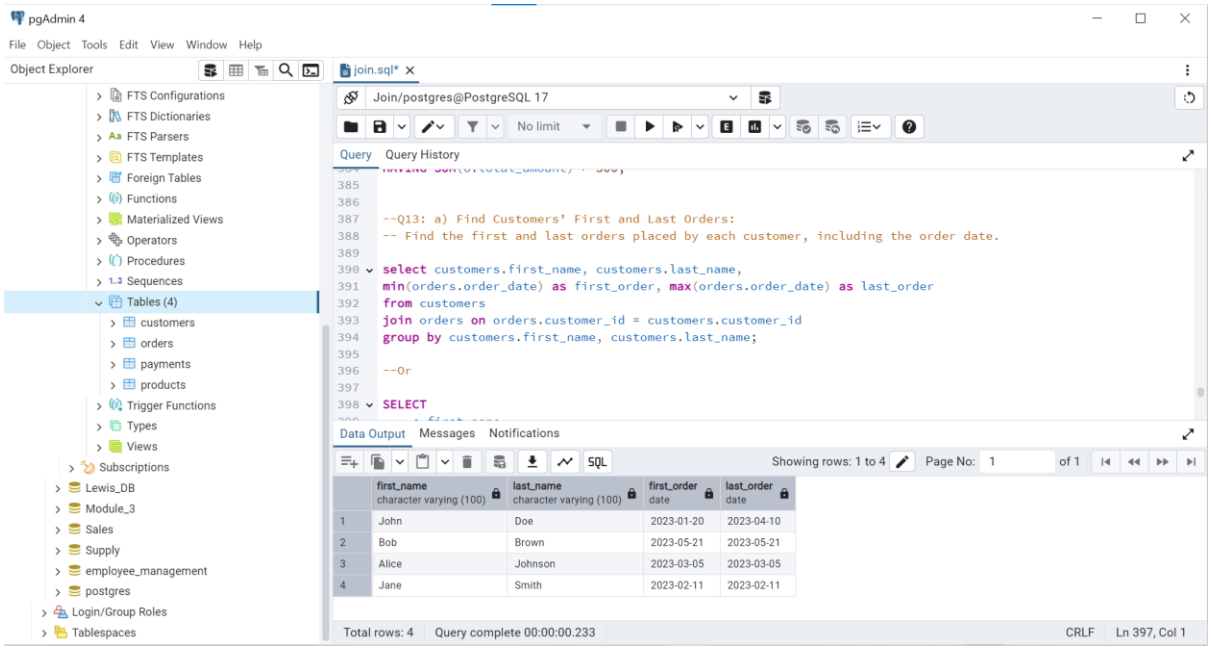

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query Query History
--Q8: Show the order status (Pending, Completed, Failed) based on the payment_status for each order.
select * from orders;
select * from payments;
select orders.order_id,
case
when payments.payment_status = 'Completed' then 'Completed'
when payments.payment_status = 'Pending' then 'Pending'
when payments.payment_status = 'Failed' then 'Failed'
else 'Unknown'
end as order_status
from orders
join payments on orders.order_id = payments.order_id;
Data Output Messages Notifications
Showing rows: 1 to 5 Page No: 1 of 1
order_id [PK] integer order_status text
1 1 Completed
2 2 Failed
3 3 Completed
4 4 Pending
5 5 Completed
Total rows: 5 Query complete 00:00:00.146 CRLF Ln 269, Col 24

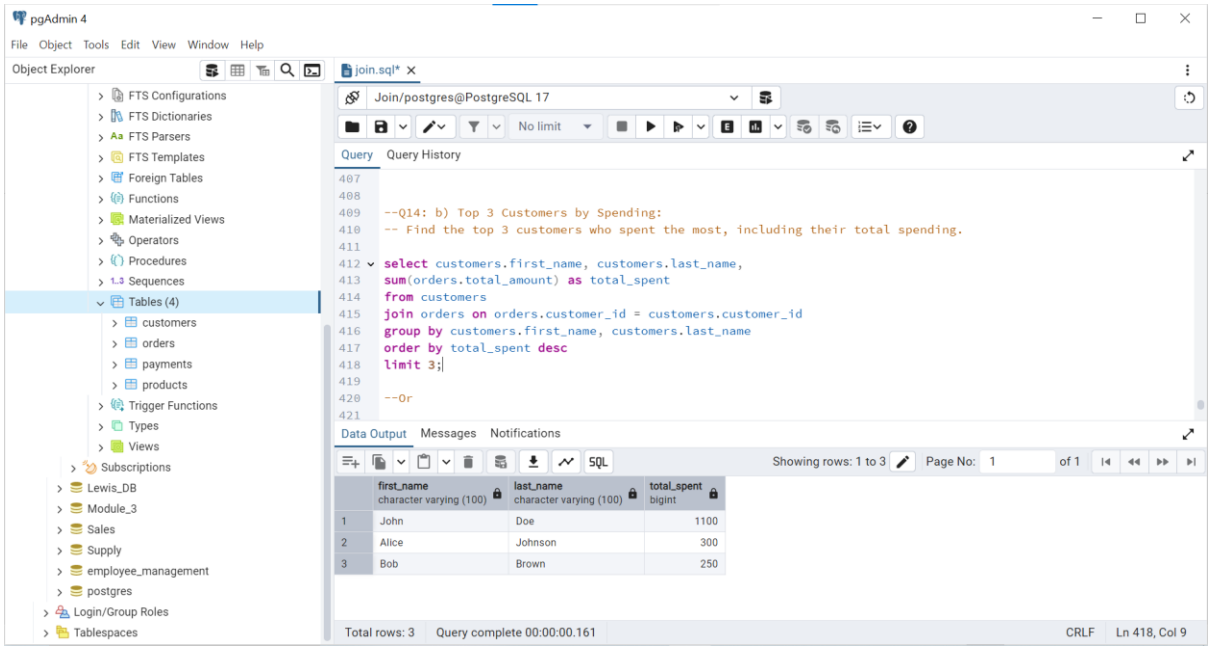

pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query Query History
--Q9: For each order, apply a 10% discount if the product belongs to the "Electronics" category, and
-- no discount for others.
select orders.order_id, products.product_name, products.category, orders.total_amount,
case
when products.category ='Electronics' then orders.total_amount * 0.90
else orders.total_amount
end as discounted_price
from orders
join products on products.product_id = orders.product_id;
--Or
SELECT
Data Output Messages Notifications
Showing rows: 1 to 5 Page No: 1 of 1
order_id integer product_name character varying (50) category character varying (50) total_amount integer discounted_price numeric
1 1 Laptop Electronics 500 450.00
2 2 Smartphone Electronics 200 180.00
3 3 Headphones Accessories 300 300
4 4 Tablet Electronics 600 540.00
5 5 Laptop Electronics 250 225.00
Total rows: 5 Query complete 00:00:00.140 CRLF Ln 295, Col 8


pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query Query History
--Q10: Classify customers based on their total spending into different categories:
-- "High spender" for customers who spent more than $500,
-- "Average spender" for those who spent between $200 and $500,
-- "Low spender" for those who spent less than $200.
select customers.first_name, customers.last_name, sum(orders.total_amount) as total_spent,
case
when sum(orders.total_amount) > 500 then 'High spender'
when sum(orders.total_amount) between 200 and 500 then 'Average spender'
else 'Low spender'
end as spending_category
from customers
join orders on orders.customer_id = customers.customer_id
group by customers.first_name, customers.last_name;
Data Output Messages Notifications
Showing rows: 1 to 4 Page No: 1 of 1
first_name character varying (100) last_name character varying (100) total_spent bigint spending_category text
1 John Doe 1100 High spender
2 Bob Brown 250 Average spender
3 Alice Johnson 300 Average spender
4 Jane Smith 200 Average spender
Total rows: 4 Query complete 00:00:00.197 CRLF Ln 322, Col 52


pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query Query History
--Q11: a) Find Customers with Multiple Orders:
-- Show all customers who have placed more than one order, including the count of orders they placed.
select customers.first_name, customers.last_name,
count(orders.order_id) as total_orders
from customers
join orders on orders.customer_id = customers.customer_id
group by customers.first_name, customers.last_name
having count(orders.order_id) > 1;
--Or
SELECT
Data Output Messages Notifications
Showing rows: 1 to 1 Page No: 1 of 1
first_name character varying (100)
last_name character varying (100)
total_orders bigint
1 John Doe 2
Total rows: 1 Query complete 00:00:00.163 CRLF Ln 351, Col 1


pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query Query History
HAVING COUNT(o.order_id) > 1;
--Q12: b) Total Spend per Category:
-- Find the total amount spent per product category, only for categories where the total spending is
-- over $500.
select products.category, sum(orders.total_amount) as total_spent
from orders
join products on products.product_id = orders.product_id
group by products.category
having sum(orders.total_amount) >500;
--Or
Data Output Messages Notifications
Showing rows: 1 to 1 Page No: 1 of 1
category character varying (50)
total_spent bigint
1 Electronics 1550
Total rows: 1 Query complete 00:00:00.137 CRLF Ln 374, Col 1


pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query Query History
--Q13: a) Find Customers' First and Last Orders:
-- Find the first and last orders placed by each customer, including the order date.
select customers.first_name, customers.last_name,
min(orders.order_date) as first_order, max(orders.order_date) as last_order
from customers
join orders on orders.customer_id = customers.customer_id
group by customers.first_name, customers.last_name;
--Or
SELECT
Data Output Messages Notifications
Showing rows: 1 to 4 Page No: 1 of 1
first_name character varying (100)
last_name character varying (100)
first_order date
last_order date
1 John Doe 2023-01-20 2023-04-10
2 Bob Brown 2023-05-21 2023-05-21
3 Alice Johnson 2023-03-05 2023-03-05
4 Jane Smith 2023-02-11 2023-02-11
Total rows: 4 Query complete 00:00:00.233 CRLF Ln 397, Col 1


pgAdmin 4
File Object Tools Edit View Window Help
Object Explorer
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (4)
customers
orders
payments
products
Trigger Functions
Types
Views
Subscriptions
Lewis_DB
Module_3
Sales
Supply
employee_management
postgres
Login/Group Roles
Tablespaces
join.sql*
Join/postgres@PostgreSQL 17
No limit
Query
Query History
--Q14: b) Top 3 Customers by Spending:
-- Find the top 3 customers who spent the most, including their total spending.
select customers.first_name, customers.last_name,
sum(orders.total_amount) as total_spent
from customers
join orders on orders.customer_id = customers.customer_id
group by customers.first_name, customers.last_name
order by total_spent desc
limit 3;
--Or
Data Output
Messages
Notifications
SQL
Showing rows: 1 to 3
Page No: 1
of 1
first_name character varying (100)
last_name character varying (100)
total_spent bigint
1 John Doe 1100
2 Alice Johnson 300
3 Bob Brown 250
Total rows: 3
Query complete 00:00:00.161
CRLF
Ln 418, Col 9