

BUAN 6320.0W2 - Group Project

Group No.3

Task: Queries with their output

Members:

Roopali Reddy Kallem (RXK180054)

Nguyen Thanh Nguyen (NXN210036)

Nihar Vinnakota (NXV230017)

Haritha Jampani (HXJ220031)

----Query 1: Select all columns and all rows from one table (5 points)

SELECT * FROM customer;

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows the database schema with tables like customer, orders, and payment.
- Query Editor:** Contains the following SQL code:

```
1 set search_path to public;
2
3 ----Query 1: Select all columns and all rows from one table (5 points)
4 SELECT * FROM customer;
```
- Data Output:** Displays the results of the query, showing 10 rows of customer data:

| customer_id | cust_name | cust_email | cust_address | created_by | date_created |
|-------------|------------------|------------------------------|--------------------|------------|--------------|
| 1001 | John Doe | john.doe@example.com | 123 Main Street | admin | 2023-11-17 |
| 1003 | Jane Smith | jane.smith@example.com | 234 Oak Avenue | admin | 2023-11-17 |
| 3005 | Alice Johnson | alice.johnson@example.com | 345 Pine Road | admin | 2023-11-17 |
| 4007 | Bob Brown | bob.brown@example.com | 456 Maple Lane | admin | 2023-11-17 |
| 5009 | Charlie Davis | charlie.davis@example.com | 567 Cedar Blvd | admin | 2023-11-17 |
| 61011 | Alex Bay | alex.bay@example.com | 346 Pine Road | admin | 2023-11-17 |
| 71013 | Cobb Brown | cobb.brown@example.com | 456 Mappleton Lane | admin | 2023-11-17 |
| 81015 | Sophie Hightower | sophie.hightower@example.com | 567 Cedarpine Blvd | admin | 2023-11-17 |
| 91017 | Copper Bron | copper.bron@example.com | 456 Map Lane | admin | 2023-11-17 |
| 101019 | Sleasha Higher | sleasha.higher@example.com | 567 Cedarpine Dr | admin | 2023-11-17 |

Total rows: 10 of 10 Query complete 00:00:00.102

----Query 2: Select five columns and all rows from one table (5 points)

SELECT order_id, customer_id, total_quantity, total_amount, order_status

FROM orders;

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows the database schema with tables like customer, orders, and payment.
- Query Editor:** Contains the following SQL code:

```
6 ----Query 2: Select five columns and all rows from one table (5 points)
7 SELECT order_id, customer_id, total_quantity, total_amount, order_status
8 FROM orders$1
9
10 ----Query 3: Select all columns from all rows from one view (5 points)
11 EXECUTE IMMEDIATE $1;
```
- Data Output:** Displays the results of the query, showing 10 rows of order data:

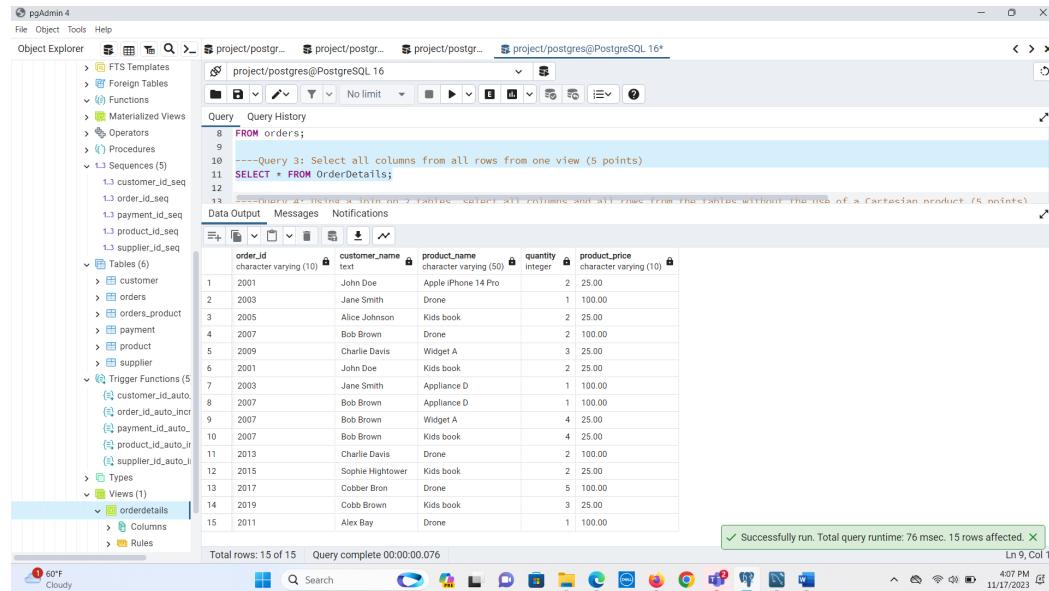
| order_id | customer_id | total_quantity | total_amount | order_status |
|----------|-------------|----------------|--------------|--------------|
| 1001 | 1001 | 2 | 100.00 | Pending |
| 2003 | 1003 | 3 | 200.00 | Delivered |
| 3005 | 1005 | 5 | 50.00 | Processing |
| 4007 | 1007 | 7 | 500.00 | Delivered |
| 5009 | 1009 | 9 | 75.00 | Shipped |
| 62011 | 1011 | 2 | 100.00 | Pending |
| 72013 | 1009 | 3 | 200.00 | Delivered |
| 82015 | 1015 | 5 | 50.00 | Processing |
| 92017 | 1017 | 7 | 500.00 | Delivered |
| 102019 | 1013 | 9 | 75.00 | Shipped |

Total rows: 10 of 10 Query complete 00:00:00.081

Successfully run. Total query runtime: 81 msec. 10 rows affected.

----Query 3: Select all columns from all rows from one view (5 points)

SELECT * FROM OrderDetails;



The screenshot shows the pgAdmin 4 interface with the following details:

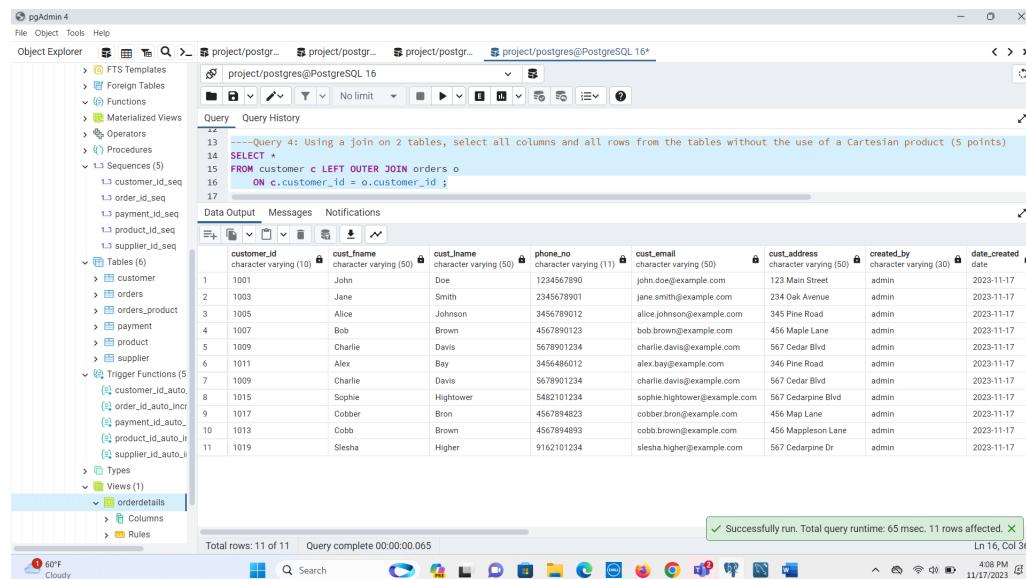
- Object Explorer:** Shows the database schema with various objects like FTS Templates, Functions, Materialized Views, Procedures, Sequences, Tables, Trigger Functions, Types, and Views.
- Query Editor:** Contains the SQL query:

```
8  FROM orders;
9
10 ----Query 3: Select all columns from all rows from one view (5 points)
11  SELECT * FROM OrderDetails;
12
13 ----Query 4: Using a join on 2 tables, select all columns and all rows from the tables without the use of a Cartesian product (5 points)
```
- Data Output:** Displays the results of the query:

| order_id | customer_name | product_name | quantity | product_price |
|----------|---------------|------------------|---------------------|---------------|
| 1 | 2001 | John Doe | Apple iPhone 14 Pro | 25.00 |
| 2 | 2003 | Jane Smith | Drone | 100.00 |
| 3 | 2005 | Alice Johnson | Kids book | 25.00 |
| 4 | 2007 | Bob Brown | Drone | 100.00 |
| 5 | 2009 | Charlie Davis | Widget A | 25.00 |
| 6 | 2001 | John Doe | Kids book | 25.00 |
| 7 | 2003 | Jane Smith | Appliance D | 100.00 |
| 8 | 2007 | Bob Brown | Appliance D | 100.00 |
| 9 | 2007 | Bob Brown | Widget A | 25.00 |
| 10 | 2007 | Bob Brown | Kids book | 25.00 |
| 11 | 2013 | Charlie Davis | Drone | 100.00 |
| 12 | 2015 | Sophie Hightower | Kids book | 25.00 |
| 13 | 2017 | Copper Bron | Drone | 100.00 |
| 14 | 2019 | Cobb Brown | Kids book | 25.00 |
| 15 | 2011 | Alex Bay | Drone | 100.00 |
- Status Bar:** Shows "Successfully run. Total query runtime: 76 msec. 15 rows affected." and the current date and time.

----Query 4: Using a join on 2 tables, select all columns and all rows from the tables without the use of a Cartesian product (5 points)

SELECT * FROM customer c LEFT OUTER JOIN orders o ON c.customer_id = o.customer_id ;



The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** Shows the database schema with various objects like FTS Templates, Functions, Materialized Views, Procedures, Sequences, Tables, Trigger Functions, Types, and Views.
- Query Editor:** Contains the SQL query:

```
13 ----Query 4: Using a join on 2 tables, select all columns and all rows from the tables without the use of a Cartesian product (5 points)
14  SELECT *
15  FROM customer c LEFT OUTER JOIN orders o
16    ON c.customer_id = o.customer_id ;
17
```
- Data Output:** Displays the results of the query:

| customer_id | cust_name | cust_email | cust_address | created_by | date_created |
|-------------|------------------|------------------------------|--------------------|------------|--------------|
| 1 | John Doe | john.doe@example.com | 123 Main Street | admin | 2023-11-17 |
| 2 | Jane Smith | jane.smith@example.com | 234 Oak Avenue | admin | 2023-11-17 |
| 3 | Alice Johnson | alice.johnson@example.com | 345 Pine Road | admin | 2023-11-17 |
| 4 | Bob Brown | bob.brown@example.com | 456 Maple Lane | admin | 2023-11-17 |
| 5 | Charlie Davis | charlie.davis@example.com | 567 Cedar Blvd | admin | 2023-11-17 |
| 6 | Alex Bay | alex.bay@example.com | 346 Pine Road | admin | 2023-11-17 |
| 7 | Charlie Davis | charlie.davis@example.com | 567 Cedar Blvd | admin | 2023-11-17 |
| 8 | Sophie Hightower | sophie.hightower@example.com | 567 Cedarpine Blvd | admin | 2023-11-17 |
| 9 | Copper Bron | copper.bron@example.com | 456 Map Lane | admin | 2023-11-17 |
| 10 | Cobb Brown | cobb.brown@example.com | 456 Mappleton Lane | admin | 2023-11-17 |
| 11 | Slesha Higher | slesha.higher@example.com | 567 Cedarpine Dr | admin | 2023-11-17 |
- Status Bar:** Shows "Successfully run. Total query runtime: 65 msec. 11 rows affected." and the current date and time.

----Query 5: Select and order data retrieved from one table (5 points)

```
SELECT * FROM orders
```

```
ORDER BY delivery_date ;
```

| order_id | customer_id | total_quantity | total_amount | order_status | payment_Status | delivery_date | shipping_address | created_by | |
|----------|-------------|----------------|--------------|--------------|----------------|---------------|------------------|---------------------|-------|
| 1 | 2001 | 1001 | 2 | 100.00 | Pending | Unpaid | 2023-10-31 | 123 Main Street | admin |
| 2 | 2011 | 1011 | 2 | 100.00 | Pending | Unpaid | 2023-10-31 | 125 Main Street | admin |
| 3 | 2017 | 1017 | 7 | 500.00 | Delivered | Paid | 2023-12-15 | 459 Maple Lane | admin |
| 4 | 2007 | 1007 | 7 | 500.00 | Delivered | Paid | 2023-12-15 | 456 Maple Lane | admin |
| 5 | 2003 | 1003 | 3 | 200.00 | Delivered | Paid | 2023-12-18 | 234 Oak Avenue | admin |
| 6 | 2013 | 1009 | 3 | 200.00 | Delivered | Unpaid | 2023-12-18 | 2314 Oaks Avenue | admin |
| 7 | 2009 | 1009 | 9 | 75.00 | Shipped | Paid | 2023-12-22 | 567 Cedar Blvd | admin |
| 8 | 2019 | 1013 | 9 | 75.00 | Shipped | Paid | 2023-12-22 | 5367 Cedarcane Blvd | admin |
| 9 | 2005 | 1005 | 5 | 50.00 | Processing | Unpaid | [null] | 345 Pine Road | admin |
| 10 | 2015 | 1015 | 5 | 50.00 | Processing | Unpaid | [null] | 3445 Pine Road | admin |

----Query 6: Using a join on 3 tables, select 5 columns from the 3 tables. Use syntax that would limit the output to 10 rows (5 points)

```
SELECT c.customer_id, c.cust_lname, o.order_id, o.total_amount, p.payment_status
```

```
FROM customer c INNER JOIN orders o ON c.customer_id = o.customer_id
```

```
INNER JOIN payment p ON c.customer_id = p.customer_id
```

```
limit 10;
```

pgAdmin 4

File Object Tools Help

Object Explorer project/postgres@PostgreSQL 16*

Query Query History

```
22 ----Query 6: Using a join on 3 tables, select 5 columns from the 3 tables. Use syntax that would limit the output to 10 rows (5 points)
23 SELECT c.customer_id, c.cust_name, o.order_id, o.total_amount, p.payment_status
24 FROM customer c INNER JOIN orders o ON c.customer_id = o.customer_id
25      INNER JOIN payment p ON c.customer_id = p.customer_id
26      limit 10;
27
```

Data Output Messages Notifications

| customer_id | cust_name | order_id | total_amount | payment_status |
|-------------|-----------|----------|--------------|----------------|
| 1001 | Doe | 2001 | 100.00 | Paid |
| 1003 | Smith | 2003 | 200.00 | Unpaid |
| 1005 | Johnson | 2005 | 50.00 | Paid |
| 1007 | Brown | 2007 | 500.00 | Unpaid |
| 1009 | Davis | 2013 | 200.00 | Paid |
| 1009 | Davis | 2009 | 75.00 | Paid |
| 1011 | Bay | 2011 | 100.00 | Paid |
| 1013 | Brown | 2019 | 75.00 | Unpaid |
| 1015 | Hightower | 2015 | 50.00 | Paid |
| 1017 | Bron | 2017 | 500.00 | Unpaid |

Total rows: 10 of 10 Query complete 00:00:00.085

Successfully run. Total query runtime: 85 msec. 10 rows affected. Ln 26, Col 13

60°F Cloudy 4:08 PM 11/17/2023

----Query 7: Select distinct rows using joins on 3 tables (5 points)

SELECT DISTINCT *

FROM customer c INNER JOIN orders o ON c.customer_id = o.customer_id

INNER JOIN payment p ON c.customer_id = p.customer_id ;

pgAdmin 4

File Object Tools Help

Object Explorer project/postgres@PostgreSQL 16*

Query Query History

```
29 ----Query 7: Select distinct rows using joins on 3 tables (5 points)
30 SELECT DISTINCT *
31 FROM customer c INNER JOIN orders o ON c.customer_id = o.customer_id
32      INNER JOIN payment p ON c.customer_id = p.customer_id ;
33
```

Data Output Messages Notifications

| customer_id | cust_name | cust_name | phone_no | cust_email | cust_address | created_by | date_created |
|-------------|-----------|-----------|------------|------------------------------|--------------------|------------|--------------|
| 1009 | Charlie | Davis | 5678901234 | charlie.davis@example.com | 567 Cedar Blvd | admin | 2023-11-17 |
| 1009 | Charlie | Davis | 5678901234 | charlie.davis@example.com | 567 Cedar Blvd | admin | 2023-11-17 |
| 1017 | Copper | Bron | 4567894823 | copper.bron@example.com | 456 Maple Lane | admin | 2023-11-17 |
| 1013 | Cobb | Brown | 4567894893 | cobb.brown@example.com | 456 Mappleton Lane | admin | 2023-11-17 |
| 1003 | Jane | Smith | 2345678901 | jane.smith@example.com | 234 Oak Avenue | admin | 2023-11-17 |
| 1007 | Bob | Brown | 4567890123 | bob.brown@example.com | 456 Maple Lane | admin | 2023-11-17 |
| 1005 | Alice | Johnson | 3456789012 | alice.johnson@example.com | 345 Pine Road | admin | 2023-11-17 |
| 1001 | John | Doe | 1234567890 | john.doe@example.com | 123 Main Street | admin | 2023-11-17 |
| 1015 | Sophie | Hightower | 5482101234 | sophie.hightower@example.com | 567 Cedarpine Blvd | admin | 2023-11-17 |
| 1011 | Alex | Bay | 3456486012 | alex.bay@example.com | 346 Pine Road | admin | 2023-11-17 |

Total rows: 10 of 10 Query complete 00:00:00.172

Successfully run. Total query runtime: 172 msec. 10 rows affected. Ln 29, Col 1

60°F Cloudy 4:09 PM 11/17/2023

----Query 8: Use GROUP BY and HAVING in a select statement using one or more tables (5 points)

```

SELECT pr.product_id, pr.product_name, s.supplier_id, s.supplier_name, pr.product_price
FROM product pr INNER JOIN supplier s ON pr.product_id = s.product_id
GROUP BY pr.product_id, pr.product_name, s.supplier_id, s.supplier_name, pr.product_price
HAVING pr.product_price = '999.00' ;

```

The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:

```

34 ----Query 8: Use GROUP BY and HAVING in a select statement using one or more tables (5 points)
35 SELECT pr.product_id, pr.product_name, s.supplier_id, s.supplier_name, pr.product_price
36 FROM product pr INNER JOIN supplier s ON pr.product_id = s.product_id
37 GROUP BY pr.product_id, pr.product_name, s.supplier_id, s.supplier_name, pr.product_price
38 HAVING pr.product_price = '999.00' ;

```

The results table shows:

| product_id | product_name | supplier_id | supplier_name | product_price |
|------------|---------------------|-------------|---------------|---------------|
| 3001 | Apple iPhone 14 Pro | 4001 | Apple Inc. | 999.00 |
| 3011 | Apple iPhone 13 Pro | 4011 | Apple Inc. | 999.00 |

Message bar: Successfully run. Total query runtime: 214 msec. 2 rows affected.

----Query 9: Use IN clause to select data from one or more tables (5 points)

```

SELECT * FROM customer
WHERE customer_id IN ('1001', '1003', '1005');

```

pgAdmin 4

File Object Tools Help

Object Explorer project/postgres@PostgreSQL 16*

Query Query History

```

40 ----Query 9: Use IN clause to select data from one or more tables (5 points)
41 SELECT * FROM customer
42 WHERE customer_id IN ('1001', '1003', '1005');
43
44 ----Query 10: Select length of one column from one table (use LENGTH function) (5 points)

```

Data Output Messages Notifications

| customer_id | cust_name | cust_email | cust_address | created_by | date_created |
|-------------|---------------|---------------------------|-----------------|------------|--------------|
| 1001 | John Doe | john.doe@example.com | 123 Main Street | admin | 2023-11-17 |
| 1003 | Jane Smith | jane.smith@example.com | 234 Oak Avenue | admin | 2023-11-17 |
| 1005 | Alice Johnson | alice.johnson@example.com | 345 Pine Road | admin | 2023-11-17 |

Total rows: 3 of 3 Query complete 00:00:00.103 Ln 40, Col 1

60°F Cloudy

----Query 10: Select length of one column from one table (use LENGTH function) (5 points)

`SELECT LENGTH(cust_address) FROM customer;`

pgAdmin 4

File Object Tools Help

Object Explorer project/postgres@PostgreSQL 16*

Query Query History

```

43
44 ----Query 10: Select length of one column from one table (use LENGTH function) (5 points)
45 SELECT LENGTH(cust_address) FROM customer;
46
47

```

Data Output Messages Notifications

| length |
|--------|
| 15 |
| 14 |
| 13 |
| 14 |
| 14 |
| 13 |
| 18 |
| 18 |
| 12 |
| 16 |

Total rows: 10 of 10 Query complete 00:00:00.199 Ln 43, Col 1

60°F Cloudy

---Query 11: Delete one record from one table. Use select statements to demonstrate the table contents before and after the DELETE statement. Make sure you use ROLLBACK

---afterwards so that the data will not be physically removed (5 points)

SELECT * FROM Product;

The screenshot shows the pgAdmin 4 interface. On the left is the Object Explorer pane, which lists various database objects like Schemas, Tables, Functions, and Procedures. The main area is a query editor with the following content:

```
5 ---Query 11: Delete one record from one table. Use select statements to demonstrate the table contents before and after the DELETE statement.
6 ----afterwards so that the data will not be physically removed (5 points)
7 BEGIN;
8 -- Display data before deletion
9 SELECT * FROM Product;
10
11 -- Temporarily remove reference from Supplier
12 DELETE FROM Supplier WHERE Product_ID = '3003';
13
14 DELETE FROM Orders_Product WHERE Product_ID = '3003';
15
16 -- Delete the product
```

Below the query editor is a results grid titled "Data Output". It shows a table with the following data:

| product_id | inventory_status | product_name | description | estimated_delivery_date | product_price | created_by | date_created |
|------------|------------------|---------------------|------------------------------|-------------------------|---------------|------------|--------------|
| 1 | In Stock | Apple iPhone 14 Pro | The latest iPhone from Apple | 2023-11-04 | 999.00 | admin | 2023-11-17 |
| 2 | In Stock | Widget A | A useful widget | 2023-12-25 | 25.00 | admin | 2023-11-17 |
| 3 | Out of Stock | Gadget B | An interesting gadget | 2024-01-05 | 40.00 | admin | 2023-11-17 |
| 4 | In Stock | Tool C | A durable tool | 2023-12-30 | 15.00 | admin | 2023-11-17 |
| 5 | In Stock | Appliance D | An essential appliance | 2023-12-20 | 100.00 | admin | 2023-11-17 |
| 6 | In Stock | Apple iPhone 13 Pro | iPhone from Apple | 2023-11-04 | 999.00 | admin | 2023-11-17 |
| 7 | In Stock | Kids book | A writing book for kids | 2023-12-25 | 25.00 | admin | 2023-11-17 |
| 8 | Out of Stock | Story book | An interesting story book | 2024-01-05 | 40.00 | admin | 2023-11-17 |
| 9 | In Stock | Spanner kit | A mechanical tool | 2023-12-30 | 15.00 | admin | 2023-11-17 |
| 10 | In Stock | Drone | A remote control gadget | 2023-12-20 | 100.00 | admin | 2023-11-17 |

Total rows: 10 of 10 Query complete 00:00:00.162

Ln 7, Col 7

60°F Cloudy

Windows taskbar icons: Search, Start, File Explorer, Task View, Mail, Microsoft Edge, File Explorer, Intel, Firefox, Chrome, OneDrive, Word, Excel, Powerpoint, Outlook, Edge.

-- Temporarily remove reference from Supplier

DELETE FROM Supplier WHERE Product_ID = '3003';

DELETE FROM Orders_Product WHERE Product_ID = '3003';

-- Delete the product

DELETE FROM Product WHERE Product_ID = '3003';

-- Display data after deletion

SELECT * FROM Product;

pgAdmin 4

File Object Tools Help

Object Explorer Dashboard Properties SQL Statistics Dependencies Dependents Processes project/postgres@PostgreSQL 16*

Query Query History

```

15
16 -- Delete the product
17 DELETE FROM Product WHERE Product_ID = '3003';
18
19 -- Display data after deletion
20 SELECT * FROM Product;
21
22 -- Rollback the transaction
23 ROLLBACK;
24
25

```

Data Output Messages Notifications

| product_id | inventory_status | product_name | description | estimated_delivery_date | product_price | created_by | date_created |
|------------|------------------|---------------------|------------------------------|-------------------------|---------------|------------|--------------|
| 1 | In Stock | Apple iPhone 14 Pro | The latest iPhone from Apple | 2023-11-04 | 999.00 | admin | 2023-11-17 |
| 2 | Out of Stock | Gadget B | An interesting gadget | 2024-01-05 | 40.00 | admin | 2023-11-17 |
| 3 | In Stock | Tool C | A durable tool | 2023-12-30 | 15.00 | admin | 2023-11-17 |
| 4 | In Stock | Appliance D | An essential appliance | 2023-12-20 | 100.00 | admin | 2023-11-17 |
| 5 | In Stock | Apple iPhone 13 Pro | iPhone from Apple | 2023-11-04 | 999.00 | admin | 2023-11-17 |
| 6 | In Stock | Kids book | A writing book for kids | 2023-12-25 | 25.00 | admin | 2023-11-17 |
| 7 | Out of Stock | Story book | An interesting story book | 2024-01-05 | 40.00 | admin | 2023-11-17 |
| 8 | In Stock | Spanner kit | A mechanical tool | 2023-12-30 | 15.00 | admin | 2023-11-17 |
| 9 | In Stock | Drone | A remote control gadget | 2023-12-20 | 100.00 | admin | 2023-11-17 |

Total rows: 9 of 9 Query complete 00:00:00.079 Ln 18, Col 1

60°F Cloudy 4:32 PM 11/17/2023

-- Rollback the transaction

ROLLBACK;

-- Display data after RollBack

SELECT * FROM Product;

pgAdmin 4

File Object Tools Help

Object Explorer Dashboard Properties SQL Statistics Dependencies Dependents Processes project/postgres@PostgreSQL 16*

Query History

```

17 DELETE FROM Product WHERE Product_ID = '30003';
18
19 -- Display data after deletion
20 SELECT * FROM Product;
21
22 -- Rollback the transaction
23 ROLLBACK;
24
25
26 -- Display data after RollBack
27 SELECT * FROM Product;
28

```

Data Output Messages Notifications

| product_id | inventory_status | product_name | description | estimated_delivery_date | product_price | created_by | date_created |
|------------|------------------|---------------------|------------------------------|-------------------------|---------------|------------|--------------|
| 1 | In Stock | Apple iPhone 14 Pro | The latest iPhone from Apple | 2023-11-04 | 999.00 | admin | 2023-11-17 |
| 2 | In Stock | Widget A | A useful widget | 2023-12-25 | 25.00 | admin | 2023-11-17 |
| 3 | Out of Stock | Gadget B | An interesting gadget | 2024-01-05 | 40.00 | admin | 2023-11-17 |
| 4 | In Stock | Tool C | A durable tool | 2023-12-30 | 15.00 | admin | 2023-11-17 |
| 5 | In Stock | Appliance D | An essential appliance | 2023-12-20 | 100.00 | admin | 2023-11-17 |
| 6 | In Stock | Apple iPhone 13 Pro | iPhone from Apple | 2023-11-04 | 999.00 | admin | 2023-11-17 |
| 7 | In Stock | Kids book | A writing book for kids | 2023-12-25 | 25.00 | admin | 2023-11-17 |
| 8 | Out of Stock | Story book | An interesting story book | 2024-01-05 | 40.00 | admin | 2023-11-17 |
| 9 | In Stock | Spanner kit | A mechanical tool | 2023-12-30 | 15.00 | admin | 2023-11-17 |
| 10 | In Stock | Drone | A remote control gadget | 2023-12-20 | 100.00 | admin | 2023-11-17 |

Total rows: 10 of 10 Query complete 00:00:00.068 Ln 25, Col 1

USD/JPY -0.72% 4:33 PM 11/17/2023

---Query 12: Update one record from one table. Use select statements to demonstrate the table contents before and after the UPDATE statement. Make sure you use ROLLBACK

---afterwards so that the data will not be physically removed (5 points)

SELECT * FROM supplier;

pgAdmin 4

File Object Tools Help

Object Explorer Dashboard Properties SQL Statistics Dependencies Dependents Processes project/postgres@PostgreSQL 16*

Query History

```

29 ---Query 12: Update one record from one table. Use select statements to demonstrate the table contents before and after the UPDATE statement.
30 ---afterwards so that the data will not be physically removed (5 points)
31
32
33 SELECT * FROM supplier;
34
35 UPDATE supplier
36 SET supply_frequency = 'Monthly'
37 WHERE supplier_name = 'Apple Inc.';
38
39 SELECT * FROM supplier;
40 ROLLBACK;

```

Data Output Messages Notifications

| supplier_id | product_id | supplier_name | supplier_phone_no | supply_frequency | supplying_since | created_by | date_created | modified |
|-------------|------------|-----------------|-------------------|------------------|-----------------|------------|--------------|----------|
| 1 | 3001 | Apple Inc. | 8001234566 | Monthly | 2010-01-01 | admin | 2023-11-17 | admin |
| 2 | 4003 | Acme Corp | 8001234567 | Monthly | 2018-01-01 | admin | 2023-11-17 | admin |
| 3 | 4005 | Globex Inc | 8002345678 | Weekly | 2019-05-15 | admin | 2023-11-17 | admin |
| 4 | 4007 | Soylent Corp | 8003456789 | Bi-Weekly | 2020-06-20 | admin | 2023-11-17 | admin |
| 5 | 4009 | Intech LLC | 8004567890 | Quarterly | 2018-07-30 | admin | 2023-11-17 | admin |
| 6 | 4011 | Apple Inc. | 8001234567 | Monthly | 2010-01-01 | admin | 2023-11-17 | admin |
| 7 | 4013 | SR Publication | 8001234777 | Monthly | 2018-01-01 | admin | 2023-11-17 | admin |
| 8 | 4015 | Shipping and Co | 8002346578 | Weekly | 2019-05-15 | admin | 2023-11-17 | admin |
| 9 | 4017 | Hard Corp | 8003126789 | Bi-Weekly | 2020-06-20 | admin | 2023-11-17 | admin |
| 10 | 4019 | Intel LLC | 9104567890 | Quarterly | 2018-07-30 | admin | 2023-11-17 | admin |

Total rows: 10 of 10 Query complete 00:00:00.098 Ln 32, Col 1

60°F Cloudy 4:35 PM 11/17/2023

```
UPDATE supplier
```

```
SET supply_frequency = 'Quarterly'
```

```
WHERE supplier_name = 'Apple Inc.';
```

```
SELECT * FROM supplier;
```

The screenshot shows the pgAdmin 4 interface with the following details:

- Object Explorer:** On the left, under the "Schemas (1)" section, the "public" schema is selected. It lists various database objects: Aggregates, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences (5), and Tables (6).
- Query Editor:** The main area contains the following SQL code:

```
32
33 SELECT * FROM supplier;
34
35 UPDATE supplier
36 SET supply_frequency = 'Quarterly'
37 WHERE supplier_name = 'Apple Inc.';
38
39 SELECT * FROM supplier;
40 ROLLBACK;
41
42 -- Advanced Query 1: Supplier Reliability Report
```
- Data Output:** Below the query editor, a table displays the results of the SELECT query. The columns are: supplier_id, product_id, supplier_name, supplier_phone_no, supply_frequency, supplying_since, created_by, date_created, and modified.
- Status Bar:** At the bottom right, it says "Successfully run. Total query runtime: 77 msec. 10 rows affected." and "Ln 39, Col 24".
- System Tray:** At the bottom left, it shows "60°F Cloudy".
- Taskbar:** At the bottom, the taskbar shows various application icons.

```
ROLLBACK;
```

```
-- display after rollback
```

```
SELECT * FROM supplier;
```

pgAdmin 4

File Object Tools Help

Object Explorer Dashboard Properties SQL Statistics Dependencies Dependents Processes project/postgres@PostgreSQL 16*

Query Query History

```

37 WHERE supplier_name = 'Apple Inc.';
38
39 SELECT * FROM supplier;
40 ROLLBACK;
41
42 -- display after rollback
43 SELECT * FROM supplier;
44
45 --Advanced Query 1: Supplier Reliability Report
46 ---This query generates a report on supplier reliability by comparing the estimated and actual delivery dates of products. It involves a sub-
47 SELECT

```

Data Output Messages Notifications

| | supplier_id | product_id | supplier_name | supplier_phone_no | supply_frequency | supplying_since | created_by | date_created | modified |
|----|-------------|------------|-----------------|-------------------|------------------|-----------------|------------|--------------|----------|
| 1 | 4003 | 3003 | Acme Corp | 8001234567 | Monthly | 2018-01-01 | admin | 2023-11-17 | admin |
| 2 | 4005 | 3005 | Globex Inc | 8002345678 | Weekly | 2019-05-15 | admin | 2023-11-17 | admin |
| 3 | 4007 | 3007 | Soylent Corp | 8003456789 | Bi-Weekly | 2020-06-20 | admin | 2023-11-17 | admin |
| 4 | 4009 | 3009 | Initech LLC | 8004567890 | Quarterly | 2018-07-30 | admin | 2023-11-17 | admin |
| 5 | 4013 | 3015 | SR Publication | 8001234777 | Monthly | 2018-01-01 | admin | 2023-11-17 | admin |
| 6 | 4015 | 3013 | Shipping and Co | 8002346378 | Weekly | 2019-05-15 | admin | 2023-11-17 | admin |
| 7 | 4017 | 3017 | Hard Corp | 8003126789 | Bi-Weekly | 2020-06-20 | admin | 2023-11-17 | admin |
| 8 | 4019 | 3019 | Intel LLC | 9104567890 | Quarterly | 2018-07-30 | admin | 2023-11-17 | admin |
| 9 | 4001 | 3001 | Apple Inc. | 8001234566 | Monthly | 2010-01-01 | admin | 2023-11-17 | admin |
| 10 | 4011 | 3011 | Apple Inc. | 8001234576 | Monthly | 2010-01-01 | admin | 2023-11-17 | admin |

Total rows: 10 of 10 Query complete 00:00:00.088 Ln 43, Col 24

60°F Cloudy 4:39 PM 11/17/2023

--Advanced Query 1: Supplier Reliability Report

---This query generates a report on supplier reliability by comparing the estimated and actual delivery dates of products. It involves a sub-query to get the earliest order date per product.

SELECT

```

s.Supplier_Name,
p.Product_Name,
p.Estimated_Delivery_Date,
MIN(o.Delivery_Date) AS Actual_Delivery_Date,

```

CASE

```
WHEN MIN(o.Delivery_Date) <= p.Estimated_Delivery_Date THEN 'On Time'
```

```
ELSE 'Delayed'
```

```
END AS Delivery_Status
```

FROM Supplier s

JOIN Product p ON s.Product_ID = p.Product_ID

JOIN Orders_Product op ON p.Product_ID = op.Product_ID

JOIN Orders o ON op.Order_ID = o.Order_ID

```

GROUP BY s.Supplier_Name, p.Product_Name, p.Estimated_Delivery_Date
ORDER BY s.Supplier_Name, Delivery_Status;

```

--Advanced Query 1: Supplier Reliability Report
---This query generates a report on supplier reliability by comparing the estimated and actual delivery dates of products. It involves a subquery and a CASE statement.

| | supplier_name | product_name | estimated_delivery_date | actual_delivery_date | delivery_status |
|---|-----------------|---------------------|-------------------------|----------------------|-----------------|
| 1 | Acme Corp | Widget A | 2023-12-25 | 2023-12-15 | On Time |
| 2 | Apple Inc. | Apple iPhone 14 Pro | 2023-11-04 | 2023-10-31 | On Time |
| 3 | Initech LLC | Appliance D | 2023-12-20 | 2023-12-15 | On Time |
| 4 | Intel LLC | Drone | 2023-12-20 | 2023-10-31 | On Time |
| 5 | Shipping and Co | Kids book | 2023-12-25 | 2023-10-31 | On Time |

--Advanced Query 2: Customer Spending Analysis by Payment Method

```

SELECT
    c.cust_Fname || ' ' || c.cust_LName AS Customer_Name,
    p.Payment_Method,
    COUNT(*) AS Number_of_Orders,
    SUM(p.Payment_Amount) AS Total_Spent,
    CASE
        WHEN SUM(p.Payment_Amount) > 500 THEN 'High Spender'
        WHEN SUM(p.Payment_Amount) BETWEEN 200 AND 500 THEN 'Medium Spender'
        ELSE 'Low Spender'
    END AS Spending_Category
FROM Payment p
JOIN Customer c ON p.Customer_ID = c.Customer_ID

```

```
GROUP BY c.cust_Fname, c.cust_LName, p.Payment_Method
```

```
ORDER BY Total_Spent DESC;
```

The screenshot shows the pgAdmin 4 interface with a query editor and a results table. The query is:

```
--Advanced Query 2: Customer Spending Analysis by Payment Method
SELECT
    c.cust_Fname || ' ' || c.cust_LName AS Customer_Name,
    p.Payment_Method,
    COUNT(*) AS Number_of_Orders,
    SUM(p.Payment_Amount) AS Total_Spent,
    CASE
        WHEN SUM(p.Payment_Amount) > 500 THEN 'High Spender'
        WHEN SUM(p.Payment_Amount) BETWEEN 200 AND 500 THEN 'Medium Spender'
        ELSE 'Low Spender'
    END AS Spending_Category
```

The results table contains the following data:

| | customer_name | payment_method | number_of_orders | total_spent | spending_category |
|----|------------------|----------------|------------------|-------------|-------------------|
| 1 | Bob Brown | PayPal | 1 | 500 | Medium Spender |
| 2 | Copper Bron | PayPal | 1 | 500 | Medium Spender |
| 3 | Charlie Davis | Visa | 1 | 275 | Medium Spender |
| 4 | Jane Smith | Visa | 1 | 200 | Medium Spender |
| 5 | John Doe | Credit Card | 1 | 100 | Low Spender |
| 6 | Alex Bay | Credit Card | 1 | 100 | Low Spender |
| 7 | Cobb Brown | Visa | 1 | 75 | Low Spender |
| 8 | Alice Johnson | MasterCard | 1 | 50 | Low Spender |
| 9 | Sophie Hightower | MasterCard | 1 | 50 | Low Spender |
| 10 | Slesha Higher | Visa | 1 | 0 | Low Spender |

Total rows: 10 of 10 | Query complete 00:00:00.090 | Ln 79, Col 1

--Advanced Query3 with Subquery: Identifying High Spending Customers

```
SELECT
```

```
c.cust_Fname || ' ' || c.cust_LName AS Customer_Name,
```

```
COUNT(p.Payment_ID) AS Number_of_Payments,
```

```
SUM(p.Payment_Amount::numeric) AS Total_Spending
```

```
FROM Customer c
```

```
JOIN Payment p ON c.Customer_ID = p.Customer_ID
```

```
GROUP BY c.Customer_ID, c.cust_Fname, c.cust_LName
```

```
HAVING SUM(p.Payment_Amount::numeric) > (
```

```
    SELECT AVG(Total_Amount::numeric)
```

```
    FROM (
```

```
        SELECT
```

```

Customer_ID,
SUM(Payment_Amount::numeric) AS Total_Amount
FROM Payment
GROUP BY Customer_ID
) AS SubQuery
)
ORDER BY Total_Spending DESC;

```

pgAdmin 4

File Object Tools Help

Object Explorer Dashboard Properties SQL Statistics Dependencies Dependents Processes project/postgres@PostgreSQL 16*

Query Query History

```

--Advanced Query3 with Subquery: Identifying High Spending Customers
SELECT
    c.cust_Fname || ' ' || c.cust_LName AS Customer_Name,
    COUNT(p.Payment_ID) AS Number_of_Payments,
    SUM(p.Payment_Amount::numeric) AS Total_Spending
FROM Customer c
JOIN Payment p ON c.Customer_ID = p.Customer_ID
GROUP BY c.Customer_ID, c.cust_Fname, c.cust_LName
HAVING SUM(p.Payment_Amount::numeric) > (
    SELECT AVG(Total_Amount::numeric)
)
```

Data Output Messages Notifications

| | customer_name | number_of_payments | total_spending |
|---|---------------|--------------------|----------------|
| 1 | Bob Brown | 1 | 500 |
| 2 | Copper Bron | 1 | 500 |
| 3 | Charlie Davis | 1 | 275 |
| 4 | Jane Smith | 1 | 200 |

Total rows: 4 of 4 Query complete 00:00:00.106 Ln 100, Col 1

60°F Cloudy 4:40 PM 11/17/2023