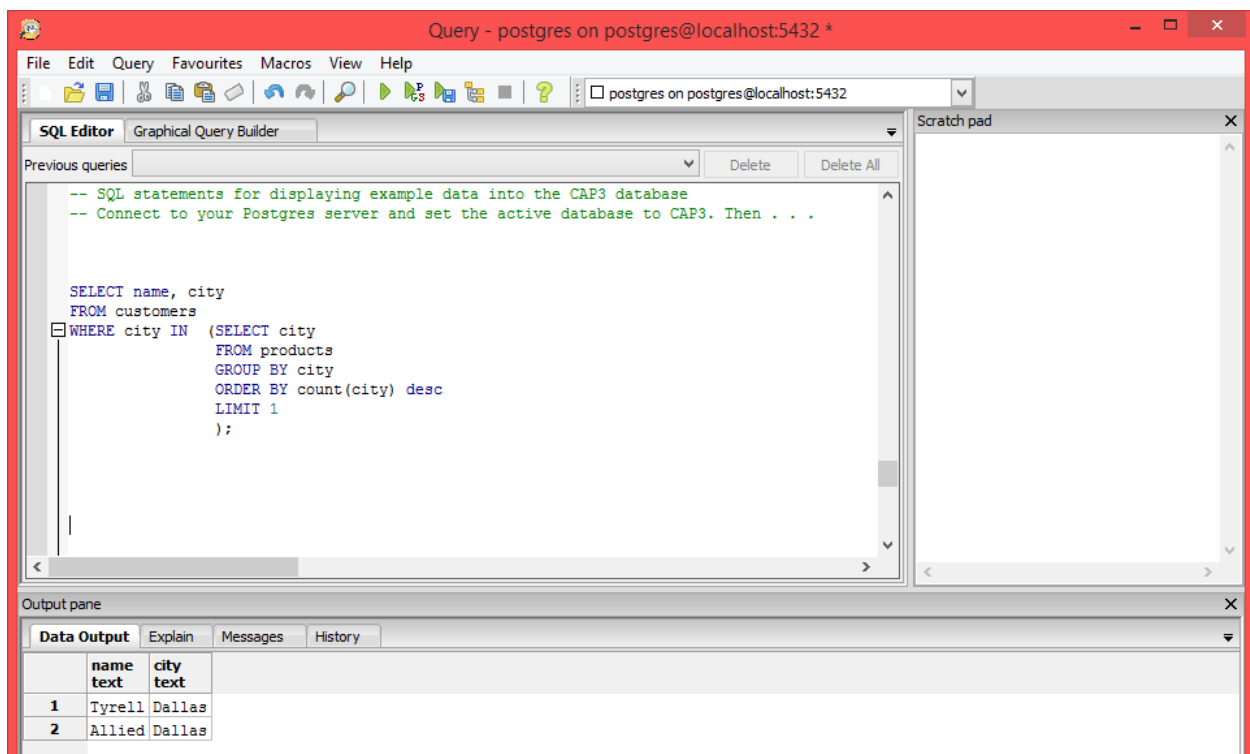


1. Display the name and city of customers who live in **any** city that makes the **most** different kinds of products. (There are two cities that make the most different products. Return the name and city of customers from **either one** of those.)

```
SELECT name, city
FROM customers
WHERE city IN (SELECT city
               FROM products
               GROUP BY city
               ORDER BY count(city) desc
               LIMIT 1
              );
```



The screenshot shows a PostgreSQL SQL Editor window titled "Query - postgres on postgres@localhost:5432 *". The window has a menu bar (File, Edit, Query, Favourites, Macros, View, Help) and a toolbar. The main area is the "SQL Editor" with a "Graphical Query Builder" tab. It contains a query that selects the name and city of customers from the 'customers' table, where the city is in the list of cities that have the most products (from the 'products' table, grouped by city, ordered by count descending, and limited to 1). The query is as follows:

```
-- SQL statements for displaying example data into the CAP3 database
-- Connect to your Postgres server and set the active database to CAP3. Then . . .

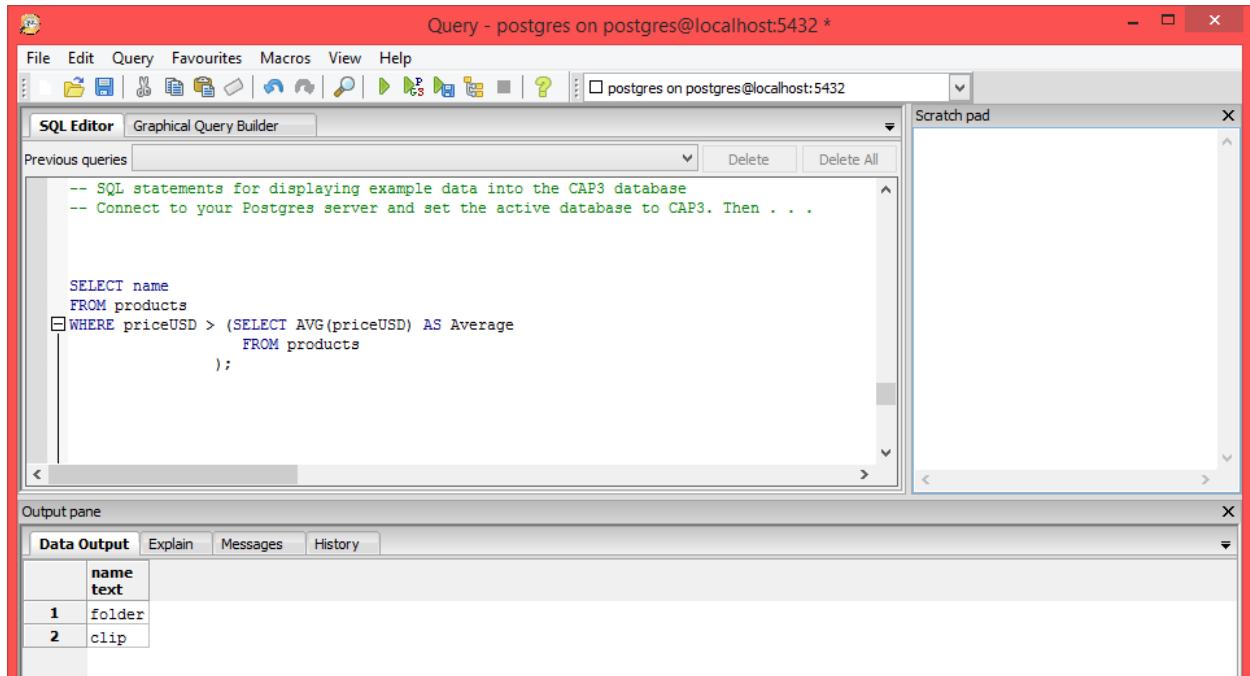
SELECT name, city
FROM customers
WHERE city IN (SELECT city
               FROM products
               GROUP BY city
               ORDER BY count(city) desc
               LIMIT 1
              );
```

Below the SQL Editor is the "Output pane" with tabs for "Data Output", "Explain", "Messages", and "History". The "Data Output" tab is active, showing the results of the query in a table with two columns: "name" and "city". The results are:

| | name | city |
|---|--------|--------|
| 1 | Tyrell | Dallas |
| 2 | Allied | Dallas |

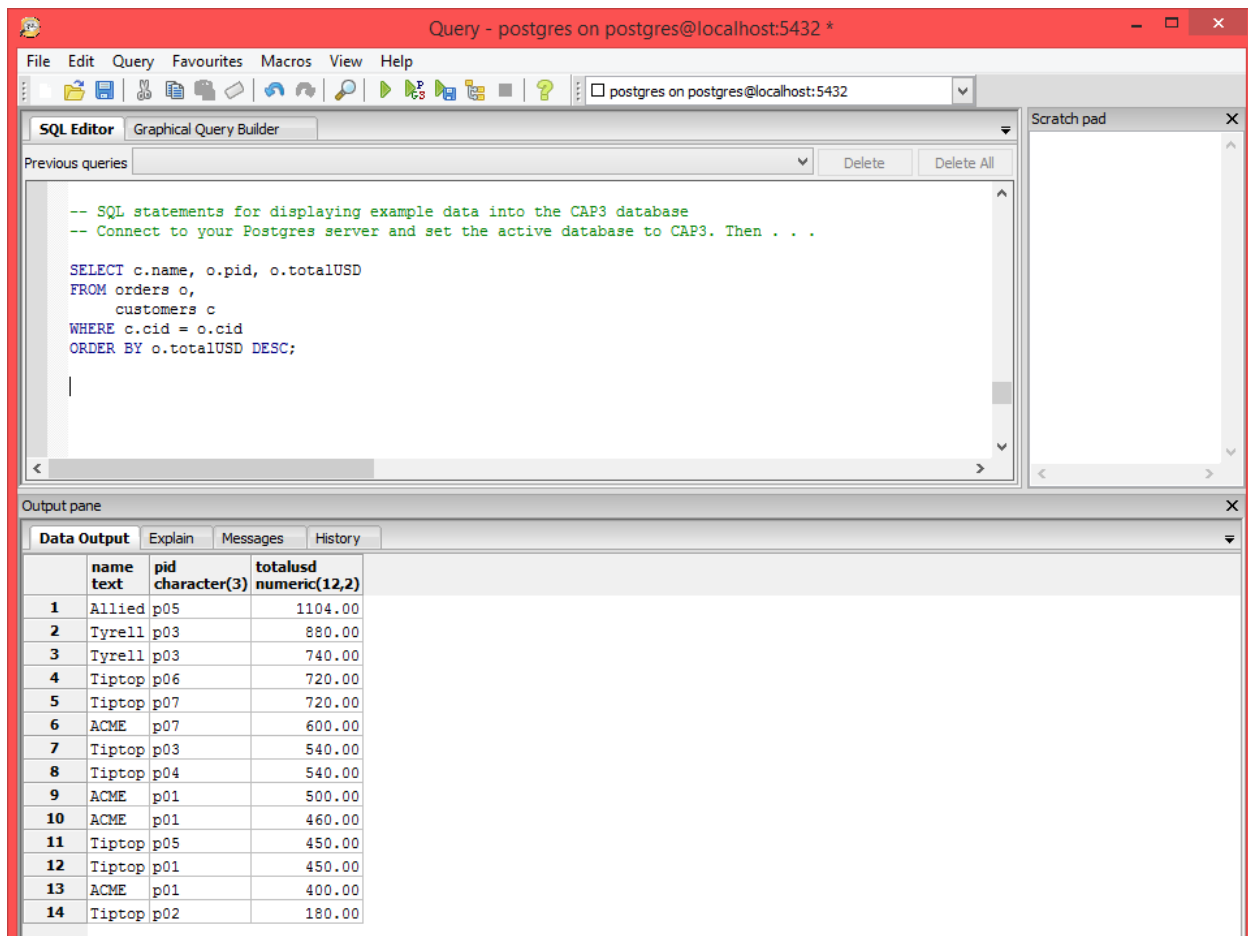
2. Display the names of products whose priceUSD is strictly above the average priceUSD, in reverse-alphabetical order.

```
SELECT name
FROM products
WHERE priceUSD > (SELECT AVG(priceUSD) AS Average
                  FROM products
                  );
```



3. Display the customer name, pid ordered, and the total for all orders, sorted by total from high to low.

```
SELECT c.name, o.pid, o.totalUSD
FROM orders o,
     customers c
WHERE c.cid = o.cid
ORDER BY o.totalUSD DESC;
```



The screenshot shows a PostgreSQL SQL Editor window titled "Query - postgres on postgres@localhost:5432 *". The window has a menu bar (File, Edit, Query, Favourites, Macros, View, Help) and a toolbar. The SQL Editor pane contains the following query:

```
-- SQL statements for displaying example data into the CAP3 database
-- Connect to your Postgres server and set the active database to CAP3. Then . . .

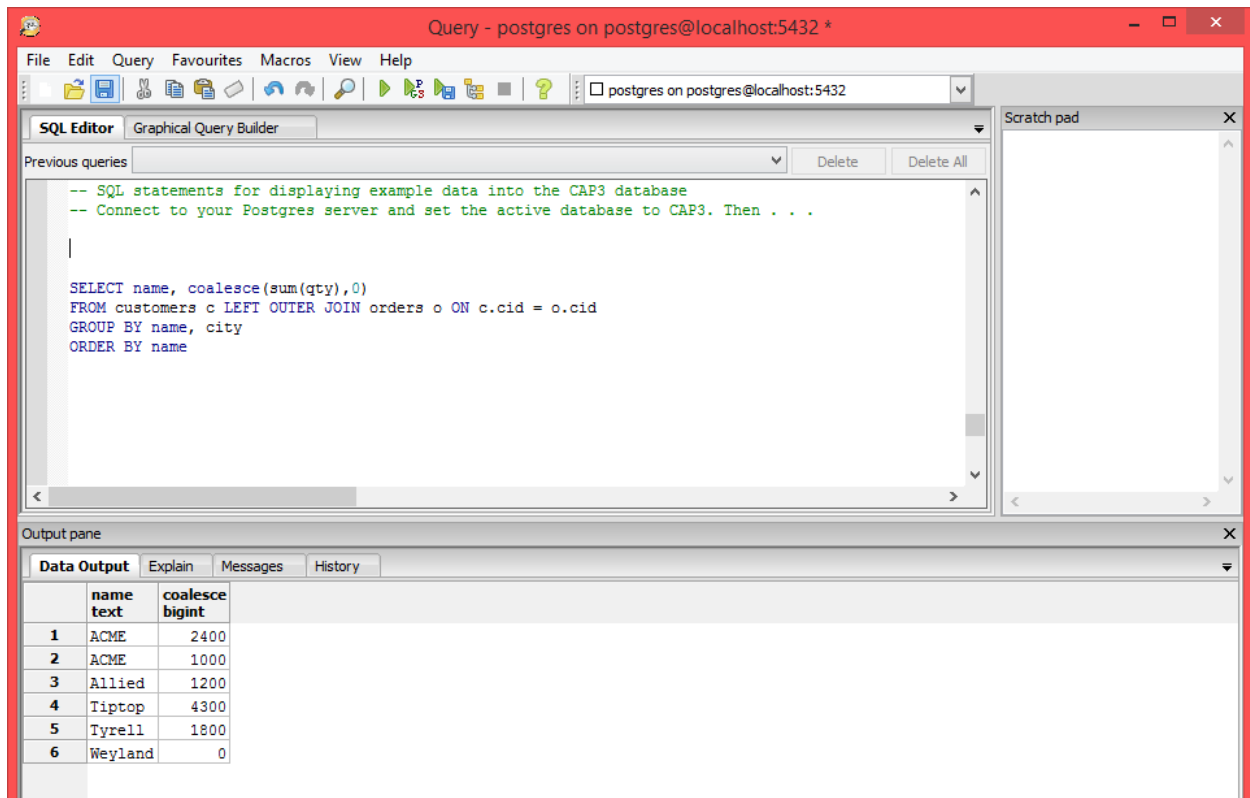
SELECT c.name, o.pid, o.totalUSD
FROM orders o,
     customers c
WHERE c.cid = o.cid
ORDER BY o.totalUSD DESC;
```

The Output pane at the bottom shows the results of the query in a table with 4 columns: name, pid, totalusd, and numeric(12,2). The results are sorted by totalusd in descending order.

| | name | pid | totalusd | numeric(12,2) |
|----|--------|--------------|----------|---------------|
| | text | character(3) | | |
| 1 | Allied | p05 | | 1104.00 |
| 2 | Tyrell | p03 | | 880.00 |
| 3 | Tyrell | p03 | | 740.00 |
| 4 | Tiptop | p06 | | 720.00 |
| 5 | Tiptop | p07 | | 720.00 |
| 6 | ACME | p07 | | 600.00 |
| 7 | Tiptop | p03 | | 540.00 |
| 8 | Tiptop | p04 | | 540.00 |
| 9 | ACME | p01 | | 500.00 |
| 10 | ACME | p01 | | 460.00 |
| 11 | Tiptop | p05 | | 450.00 |
| 12 | Tiptop | p01 | | 450.00 |
| 13 | ACME | p01 | | 400.00 |
| 14 | Tiptop | p02 | | 180.00 |

4. Display all customer names (in alphabetical order) and their total ordered, and nothing more. Use coalesce to avoid showing NULLs.

```
SELECT name, coalesce(sum(qty),0)
FROM customers c LEFT OUTER JOIN orders o ON c.cid = o.cid
GROUP BY name, city
ORDER BY name
```



The screenshot shows a PostgreSQL query editor window titled "Query - postgres on postgres@localhost:5432 *". The window has a menu bar (File, Edit, Query, Favourites, Macros, View, Help) and a toolbar. The main area is the "SQL Editor" with a "Graphical Query Builder" tab. It contains a "Previous queries" list and a "Scratch pad" on the right. The SQL query is as follows:

```
-- SQL statements for displaying example data into the CAP3 database
-- Connect to your Postgres server and set the active database to CAP3. Then . . .

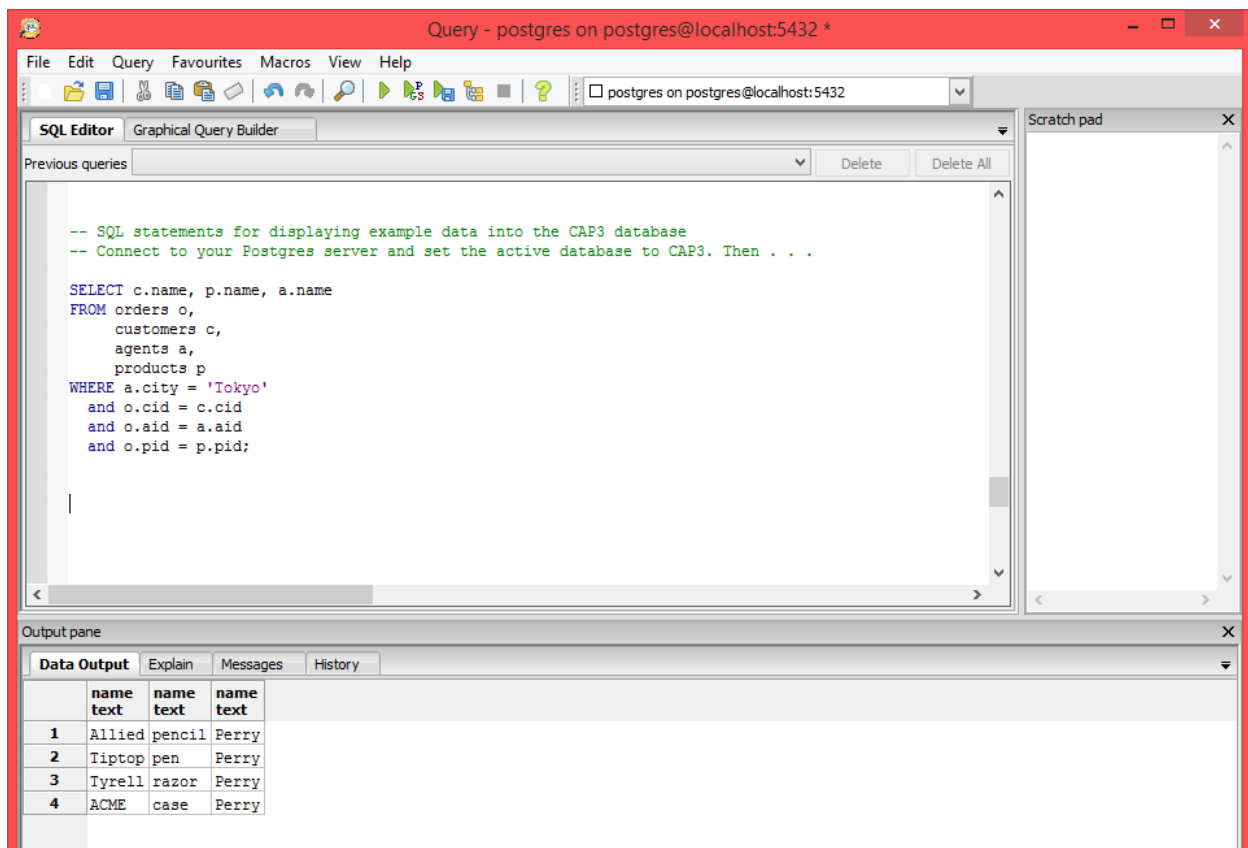
SELECT name, coalesce(sum(qty),0)
FROM customers c LEFT OUTER JOIN orders o ON c.cid = o.cid
GROUP BY name, city
ORDER BY name
```

Below the editor is the "Output pane" with tabs for "Data Output", "Explain", "Messages", and "History". The "Data Output" tab is active, showing a table with 6 rows and 2 columns: "name" (text) and "coalesce" (bigint).

| | name text | coalesce bigint |
|---|--------------|--------------------|
| 1 | ACME | 2400 |
| 2 | ACME | 1000 |
| 3 | Allied | 1200 |
| 4 | Tiptop | 4300 |
| 5 | Tyrell | 1800 |
| 6 | Weyland | 0 |

5. Display the names of all customers who bought products from agents based in Tokyo along with the names of the products they ordered, and the names of the agents who sold it to them.

```
SELECT c.name, p.name, a.name
FROM orders o,
     customers c,
     agents a,
     products p
WHERE a.city = 'Tokyo'
and o.cid = c.cid
and o.aid = a.aid
and o.pid = p.pid;
```



The screenshot shows a PostgreSQL query editor window titled "Query - postgres on postgres@localhost:5432 *". The window has a menu bar (File, Edit, Query, Favourites, Macros, View, Help) and a toolbar. The main area is the "SQL Editor" tab, which contains the following SQL query:

```
-- SQL statements for displaying example data into the CAP3 database
-- Connect to your Postgres server and set the active database to CAP3. Then . . .

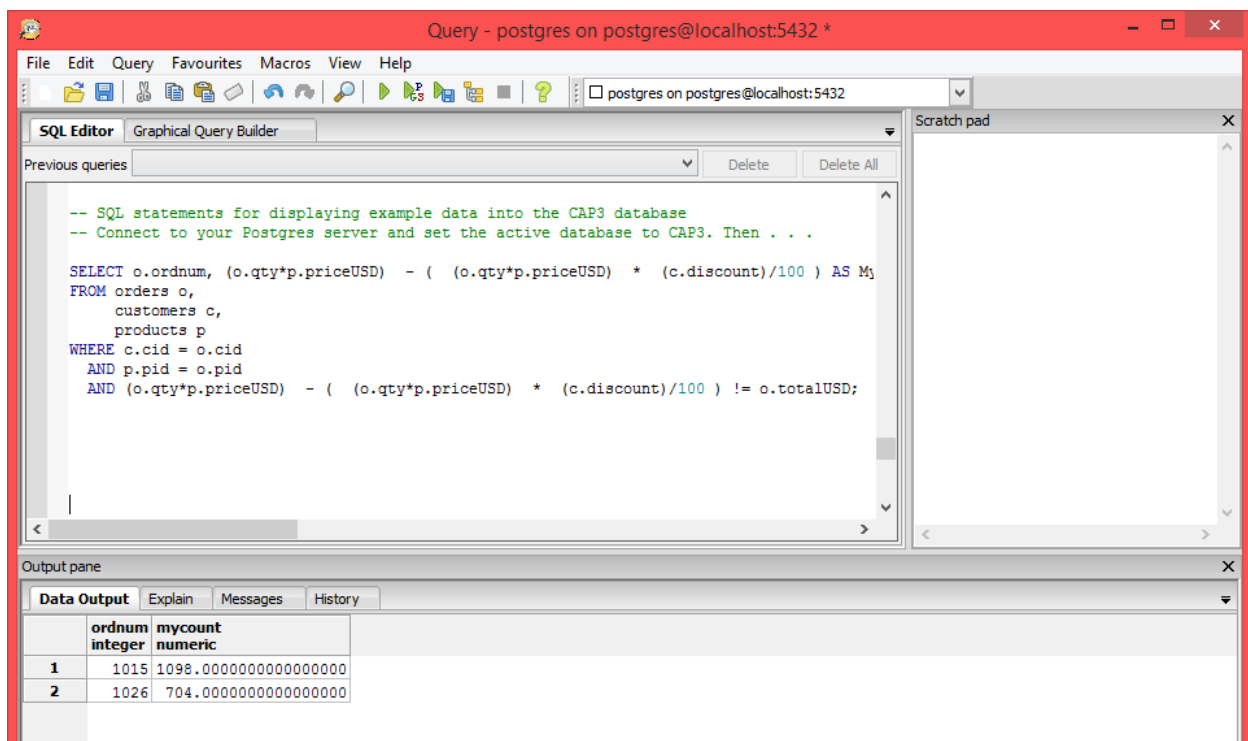
SELECT c.name, p.name, a.name
FROM orders o,
     customers c,
     agents a,
     products p
WHERE a.city = 'Tokyo'
and o.cid = c.cid
and o.aid = a.aid
and o.pid = p.pid;
```

Below the SQL Editor is the "Output pane" with tabs for "Data Output", "Explain", "Messages", and "History". The "Data Output" tab is active, showing a table with 4 rows and 3 columns. The columns are labeled "name text", "name text", and "name text". The rows are numbered 1 through 4.

| | name text | name text | name text |
|---|-----------|-----------|-----------|
| 1 | Allied | pencil | Perry |
| 2 | Tiptop | pen | Perry |
| 3 | Tyrell | razor | Perry |
| 4 | ACME | case | Perry |

6. Write a query to check the accuracy of the dollars column in the Orders table. This means calculating Orders.totalUSD from data in other tables and comparing those values to the values in Orders.totalUSD. Display all rows in Orders where Orders.totalUSD is incorrect, if any.

```
SELECT o.ordnum, (o.qty*p.priceUSD) - ( (o.qty*p.priceUSD) * (c.discount)/100 ) AS MyCount
FROM orders o,
     customers c,
     products p
WHERE c.cid = o.cid
      AND p.pid = o.pid
      AND (o.qty*p.priceUSD) - ( (o.qty*p.priceUSD) * (c.discount)/100 ) != o.totalUSD;
```



The screenshot shows a PostgreSQL SQL Editor window titled "Query - postgres on postgres@localhost:5432 *". The window has a menu bar (File, Edit, Query, Favourites, Macros, View, Help) and a toolbar. The main area is the "SQL Editor" tab, which contains the following SQL query:

```
-- SQL statements for displaying example data into the CAP3 database
-- Connect to your Postgres server and set the active database to CAP3. Then . . .

SELECT o.ordnum, (o.qty*p.priceUSD) - ( (o.qty*p.priceUSD) * (c.discount)/100 ) AS MyCount
FROM orders o,
     customers c,
     products p
WHERE c.cid = o.cid
      AND p.pid = o.pid
      AND (o.qty*p.priceUSD) - ( (o.qty*p.priceUSD) * (c.discount)/100 ) != o.totalUSD;
```

Below the SQL Editor is the "Output pane" with tabs for "Data Output", "Explain", "Messages", and "History". The "Data Output" tab is selected, showing the results of the query in a table:

| | ordnum integer | mycount numeric |
|---|-------------------|-----------------------|
| 1 | 1015 | 1098.0000000000000000 |
| 2 | 1026 | 704.0000000000000000 |

7. What's the difference between a LEFT OUTER JOIN and a RIGHT OUTER JOIN? Give example queries in SQL to demonstrate. (Feel free to use the CAP2 database to make your points here.)

Left Outer Join takes all the values from left hand table and shows the matches with the right hand table, if they exist. If there is no match, it shows NULL. In this case, we get all the rows from the left table, and matching ones from the right table. In the example of CAP 3 database, if I use left outer join and declare customer table as the left table, I can use a query to see which customer did not place any orders. The query would look like:

```
SELECT c.name, c.cid, o.cid, o.ordnum  
FROM customers c LEFT OUTER JOIN orders o ON o.cid = c.cid;
```

This way, I can see all the customer, and their corresponding order numbers, and if there are any customers who didn't order anything, I see the name and cid of the customer and NULLs for the order number. In total, I get 15 rows back, in which I can see that Weyland is the customer in the database that did not order anything.

Right Outer Join is almost the same thing, just the opposite. It takes all the values from the right hand table and matches with the left hand table and if there is no match, it displays NULLs. Continuing from the previous query, I can switch and ask to see all the order numbers and see the names and cids of the corresponding customers. It is unlikely to have an order number without having a customer name and cid, which is why, in the query below, I don't get any NULLs and can see all the 14 rows of orders and their corresponding customer info:

```
SELECT c.name, c.cid, o.cid, o.ordnum  
FROM customers c RIGHT OUTER JOIN orders o ON o.cid = c.cid;
```