

SQL CASE STUDY

DATA IN MOTION TINY SHOP SALES

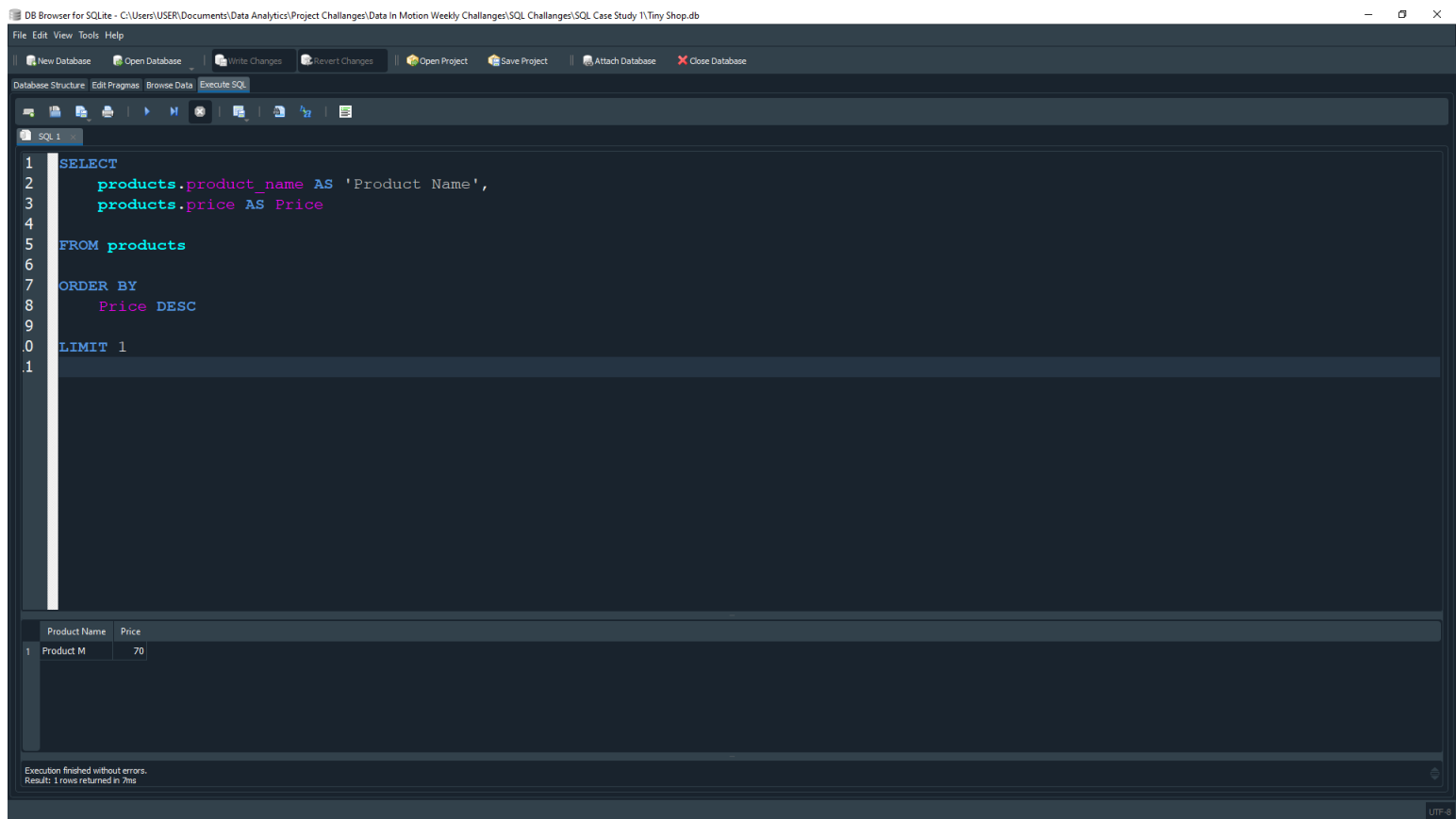


DATA IN MOTION

SQL CASE STUDY 1: TINY SHOP

Jermaine Sangiwa

QUESTION 1: WHICH PRODUCT HAS THE HIGHEST PRICE? ONLY RETURN A SINGLE ROW.



The screenshot shows the DB Browser for SQLite application. The title bar indicates the file path: C:\Users\USER\Documents\Data Analytics\Project Challenges\Data In Motion Weekly Challenges\SQL Challenges\SQL Case Study 1\Tiny Shop.db. The menu bar includes File, Edit, View, Tools, and Help. The toolbar contains buttons for New Database, Open Database, Write Changes, Revert Changes, Open Project, Save Project, Attach Database, and Close Database. The main interface has tabs for Database Structure, Edit Pragma, Browse Data, and Execute SQL. The Execute SQL tab is active, showing a SQL query in a text editor. The query is as follows:

```
1 SELECT
2     products.product_name AS 'Product Name',
3     products.price AS Price
4
5 FROM products
6
7 ORDER BY
8     Price DESC
9
10 LIMIT 1
11
```

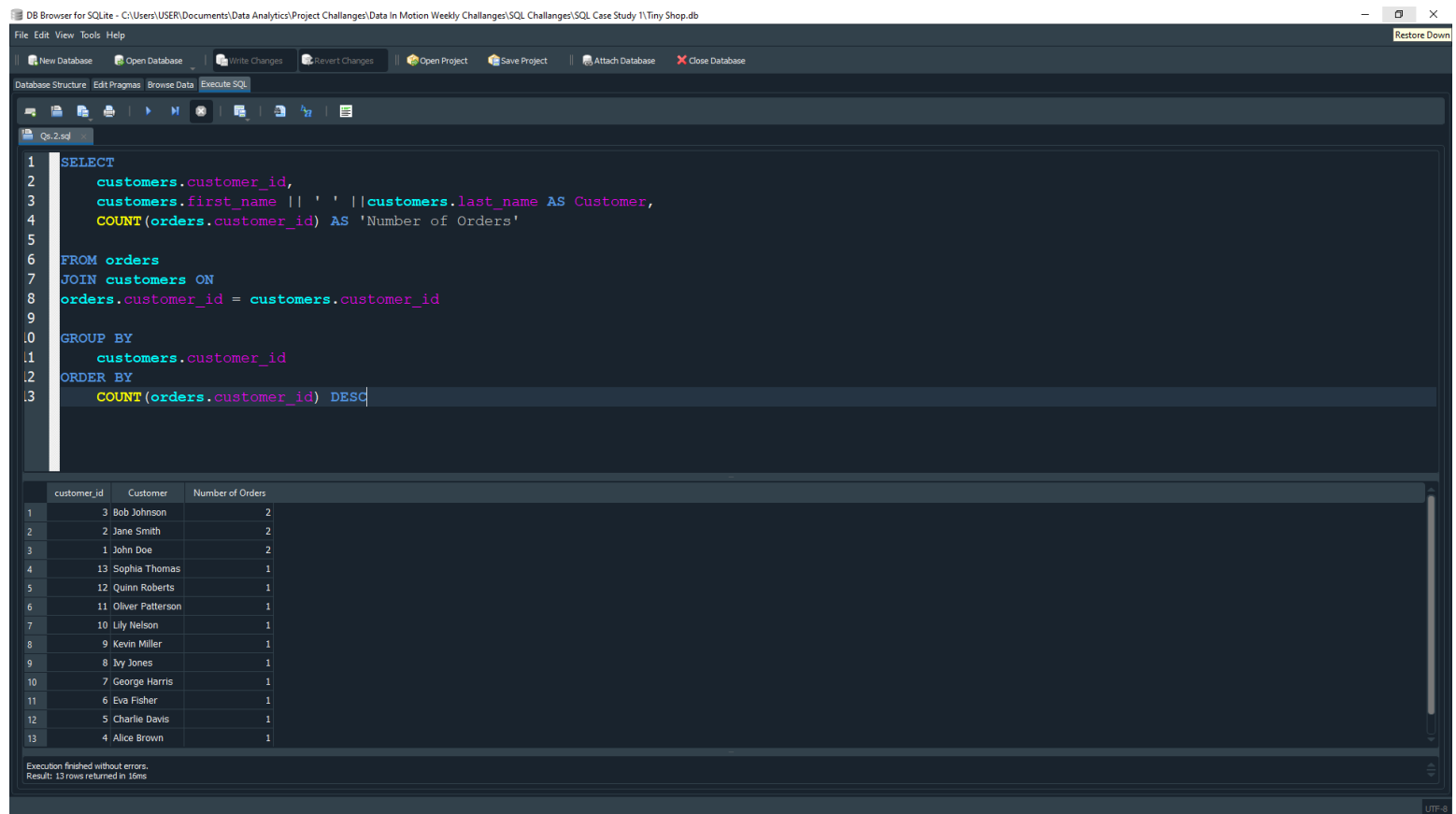
Below the query editor, the results are displayed in a table with two columns: Product Name and Price. The table contains one row:

	Product Name	Price
1	Product M	70

At the bottom of the window, a status bar indicates: Execution finished without errors. Results: 1 rows returned in 7ms. The encoding is UTF-8.

The product with the highest price is **Product M**, with the price of \$70.

QUESTION 2: WHICH CUSTOMER HAS MADE THE MOST ORDERS?



The screenshot shows the DB Browser for SQLite interface. The SQL query in the editor is as follows:

```
1 SELECT
2     customers.customer_id,
3     customers.first_name || ' ' || customers.last_name AS Customer,
4     COUNT(orders.customer_id) AS 'Number of Orders'
5
6 FROM orders
7 JOIN customers ON
8     orders.customer_id = customers.customer_id
9
10 GROUP BY
11     customers.customer_id
12 ORDER BY
13     COUNT(orders.customer_id) DESC
```

The results are displayed in a table with the following data:

	customer_id	Customer	Number of Orders
1	3	Bob Johnson	2
2	2	Jane Smith	2
3	1	John Doe	2
4	13	Sophia Thomas	1
5	12	Quinn Roberts	1
6	11	Oliver Patterson	1
7	10	Lily Nelson	1
8	9	Kevin Miller	1
9	8	Ivy Jones	1
10	7	George Harris	1
11	6	Eva Fisher	1
12	5	Charlie Davis	1
13	4	Alice Brown	1

Execution finished without errors.
Result: 13 rows returned in 10ms

Customers with the greatest number of orders are:

1. **Bob Johnson: 2**
2. **Jane Smith: 2**
3. **John Doe: 2**

QUESTION 3: WHAT'S THE TOTAL REVENUE PER PRODUCT?

DB Browser for SQLite - C:\Users\USER\Documents\Data Analytics\Project Challenges\Data In Motion Weekly Challenges\SQL Challenges\SQL Case Study 1\Tiny Shop.sqbprio (Tiny Shop.db)

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Edit Pragma Browse Data Execute SQL

Qs.2.sql Qs.3.sql Qs.5.sql Qs.6.sql SQL 7

```
1 SELECT
2     products.product_name AS Product,
3     products.price * SUM(order_items.quantity) AS Revenue
4
5 FROM order_items
6 JOIN products ON
7     order_items.product_id = products.product_id
8
9 GROUP BY
10     products.product_name
11 ORDER BY
12     Product
13
```

	Product	Revenue
1	Product A	50
2	Product B	135
3	Product C	160
4	Product D	75
5	Product E	90
6	Product F	210
7	Product G	120
8	Product H	135
9	Product I	150
10	Product J	330
11	Product K	180
12	Product L	195
13	Product M	420

Execution finished without errors.
Result: 13 rows returned in 4ms
4 rows x 1

UTF-8

The total revenue for each product is:

- 1. **Product A:** \$50
- 2. **Product B:** \$135
- 3. **Product C:** \$160
- 4. **Product D:** \$75
- 5. **Product E:** \$90
- 6. **Product F:** \$210
- 7. **Product G:** \$120
- 8. **Product H:** \$135
- 9. **Product I:** \$150
- 10. **Product J:** \$330
- 11. **Product K:** \$180
- 12. **Product L:** \$195
- 13. **Product M:** \$420

QUESTION 4: FIND THE DAY WITH THE HIGHEST REVENUE?

DB Browser for SQLite - C:\Users\USER\Documents\Data Analytics\Project Challenges\Data In Motion Weekly Challenges\SQL Challenges\SQL Case Study 1\Tiny Shop.sqdbpro (Tiny Shop.db)

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Edit Pragma Browse Data Execute SQL

Qs.2.sql Qs.3.sql Qs.4.sql Qs.5.sql Qs.6.sql Qs.7.sql Qs.8.sql Qs.9.sql

```
1 SELECT
2     orders.order_id,
3     STRFTIME('%d', orders.order_date) AS 'Order Day',
4     SUM(products.price) * SUM(order_items.quantity) AS Revenue
5
6 FROM order_items
7 JOIN products
8 JOIN orders ON
9     order_items.product_id = products.product_id AND
10    order_items.order_id = orders.order_id
11
12 GROUP BY
13     orders.order_id
14 ORDER BY
15     orders.order_id
16
```

	order_id	Order Day	Revenue
1	1	01	75
2	2	02	140
3	3	03	90
4	4	04	175
5	5	05	90
6	6	06	100
7	7	07	165
8	8	08	300
9	9	09	285
10	10	10	575
11	11	11	540
12	12	12	165
13	13	13	375
14	14	14	285
15	15	15	460
16	16	16	675

Execution finished without errors.
Result: 16 rows returned in 4ms

UTF-8

The day with the highest revenue is **Day 16**.

QUESTION 5: FIND THE FIRST ORDER (BY DATE) FOR EACH CUSTOMER.

DB Browser for SQLite - C:\Users\USER\Documents\Data Analytics\Project Challenges\Data In Motion Weekly Challenges\SQL Challenges\SQL Case Study\Tiny Shop.sqbp (Tiny Shop.db)

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Edit Pragma Browse Data Execute SQL

Q5.2.sql Q5.3.sql Q5.5.sql

```
1 SELECT
2     customers.customer_id,
3     customers.first_name || ' ' || customers.last_name AS Customer,
4     MIN(orders.order_date) AS 'First Order Date'
5
6 FROM customers
7 LEFT JOIN orders ON
8     customers.customer_id = orders.customer_id
9
10 GROUP BY
11     customers.customer_id, Customer
12
```

	customer_id	Customer	First Order Date
1	1	John Doe	2023-05-01
2	2	Jane Smith	2023-05-02
3	3	Bob Johnson	2023-05-03
4	4	Alice Brown	2023-05-07
5	5	Charlie Davis	2023-05-08
6	6	Eva Fisher	2023-05-09
7	7	George Harris	2023-05-10
8	8	Ivy Jones	2023-05-11
9	9	Kevin Miller	2023-05-12
10	10	Lily Nelson	2023-05-13
11	11	Oliver Patterson	2023-05-14
12	12	Quinn Roberts	2023-05-15
13	13	Sophia Thomas	2023-05-16

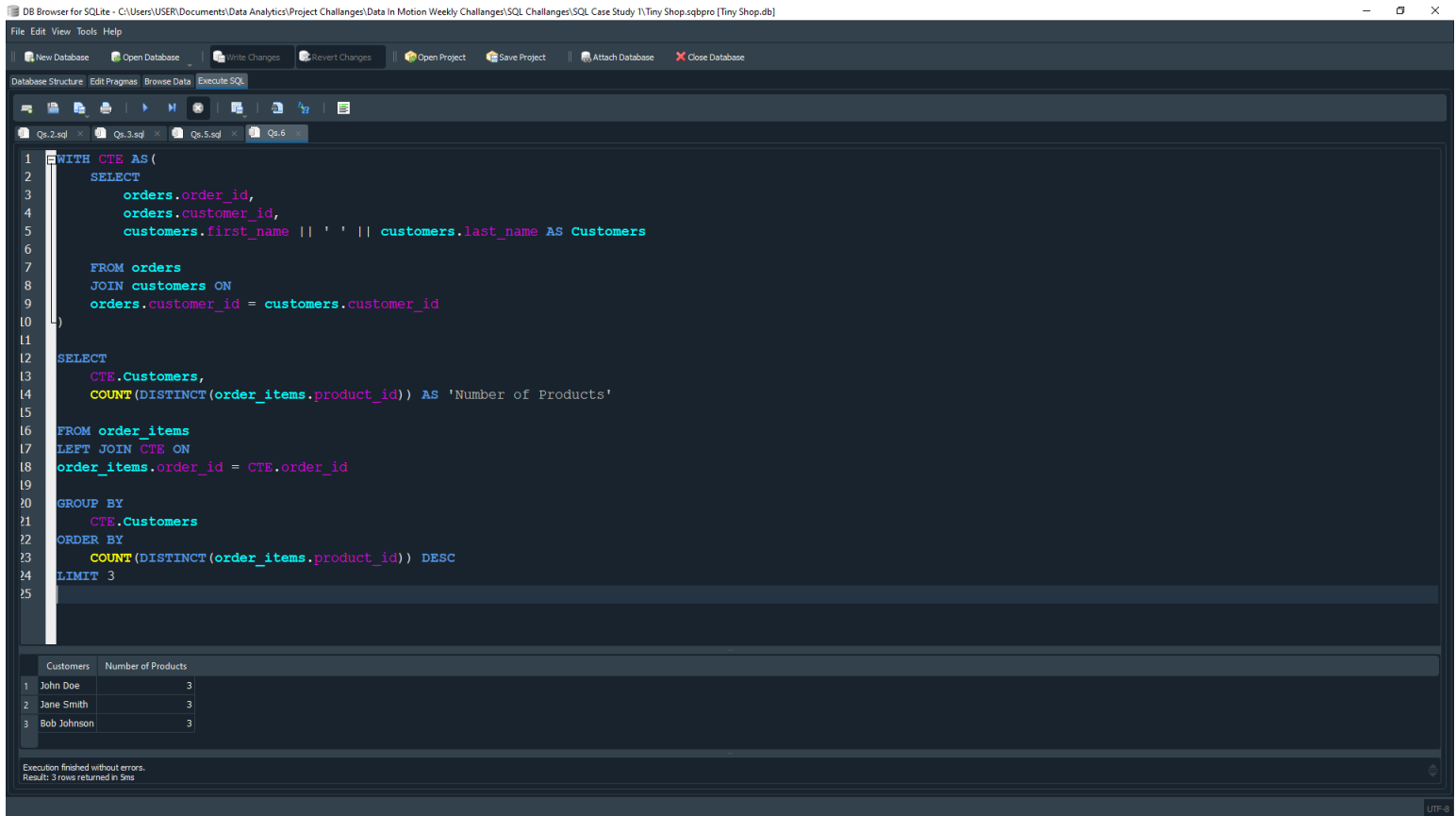
Execution finished without errors.
Result: 13 rows returned in 9ms
At Time: 10

UTF-8

The first order date for each customer is:

- 1. **John Doe:** 2023-05-01
- 2. **Jane Smith:** 2023-05-02
- 3. **Bob Johnson:** 2023-05-03
- 4. **Alice Brown:** 2023-05-07
- 5. **Charlie Davis:** 2023-05-08
- 6. **Eva Fisher:** 2023-05-09
- 7. **George Harris:** 2023-05-10
- 8. **Ivy Jones:** 2023-05-11
- 9. **Kevin Miller:** 2023-05-12
- 10. **Lily Nelson:** 2023-05-13
- 11. **Oliver Patterson:** 2023-05-14
- 12. **Quinn Roberts:** 2023-05-15
- 13. **Sophia Thomas:** 2023-05-16

QUESTION 6: FIND THE TOP 3 CUSTOMERS WHO HAVE ORDERS THE MOST DISTINCT PRODUCTS?



The screenshot shows the DB Browser for SQLite interface. The SQL editor contains the following query:

```
1 WITH CTE AS (  
2     SELECT  
3         orders.order_id,  
4         orders.customer_id,  
5         customers.first_name || ' ' || customers.last_name AS Customers  
6  
7     FROM orders  
8     JOIN customers ON  
9         orders.customer_id = customers.customer_id  
10 )  
11  
12 SELECT  
13     CTE.Customers,  
14     COUNT(DISTINCT(order_items.product_id)) AS 'Number of Products'  
15  
16 FROM order_items  
17 LEFT JOIN CTE ON  
18     order_items.order_id = CTE.order_id  
19  
20 GROUP BY  
21     CTE.Customers  
22 ORDER BY  
23     COUNT(DISTINCT(order_items.product_id)) DESC  
24 LIMIT 3  
25
```

The results pane at the bottom displays the following data:

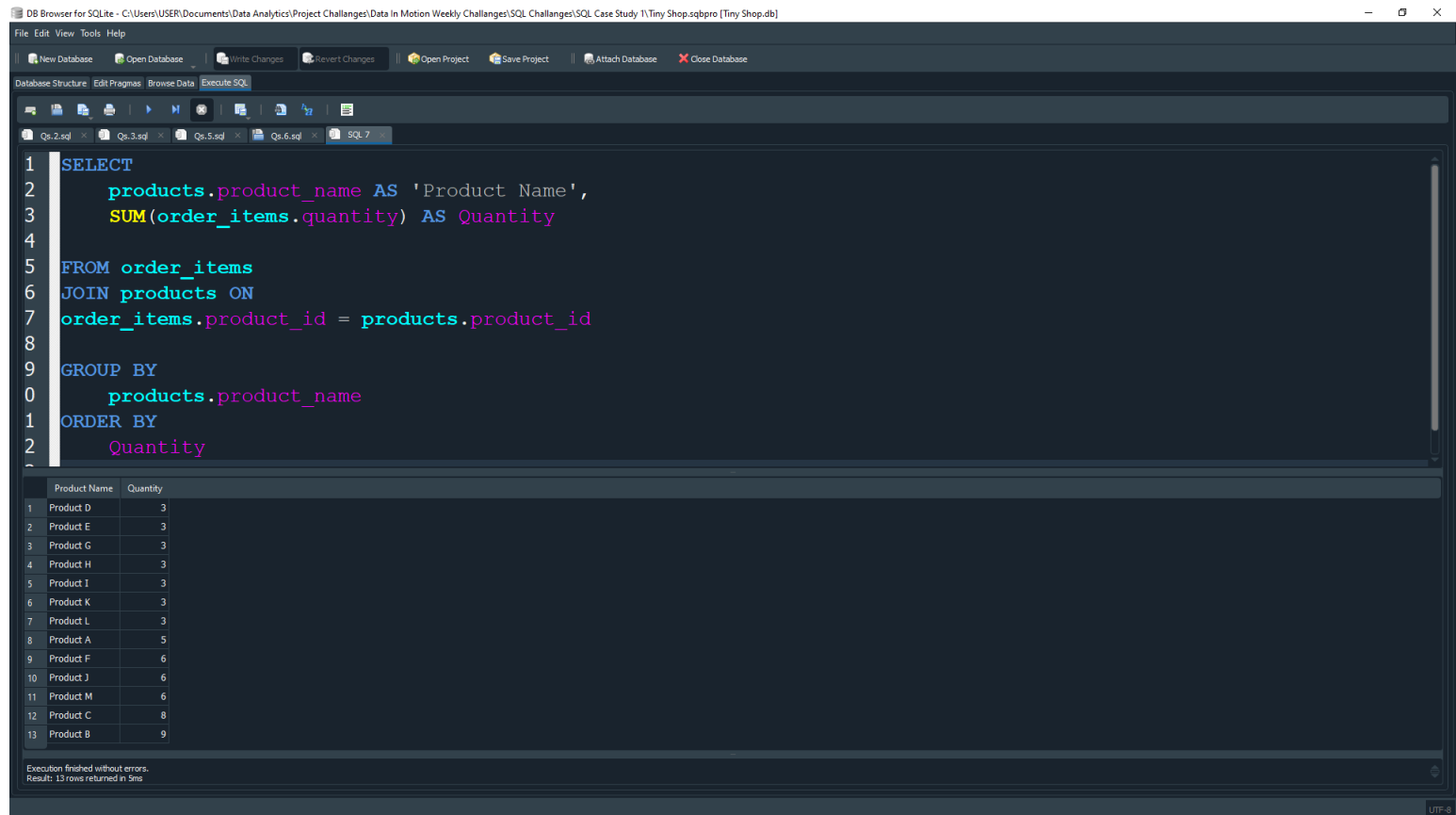
	Customers	Number of Products
1	John Doe	3
2	Jane Smith	3
3	Bob Johnson	3

Execution finished without errors.
Results: 3 rows returned in 5ms

The top 3 customers who have ordered the most distinct products are:

1. **John Doe:** 3
2. **Jane Smith:** 3
3. **Bob Johnson:** 3

QUESTION 7: WHICH PRODUCT HAS BEEN BOUGHT THE LEAST IN TERMS OF QUANTITY?



The screenshot shows the DB Browser for SQLite interface. The SQL query is as follows:

```
1 SELECT
2     products.product_name AS 'Product Name',
3     SUM(order_items.quantity) AS Quantity
4
5 FROM order_items
6 JOIN products ON
7 order_items.product_id = products.product_id
8
9 GROUP BY
10    products.product_name
11 ORDER BY
12    Quantity
13
```

The results are displayed in a table with 13 rows:

	Product Name	Quantity
1	Product D	3
2	Product E	3
3	Product G	3
4	Product H	3
5	Product I	3
6	Product K	3
7	Product L	3
8	Product A	5
9	Product F	6
10	Product J	6
11	Product M	6
12	Product C	8
13	Product B	9

Execution finished without errors.
Result: 13 rows returned in 9ms

The products that have been bought the least in terms of quantity are:

1. **Product D**
2. **Product E**
3. **Product G**
4. **Product H**
5. **Product I**
6. **Product K**
7. **Product L**

All seven of them have a total quantity of **3**

QUESTION 8: WHAT IS THE MEDIAN ORDER TOTAL?

DB Browser for SQLite - C:\Users\USER\Documents\Data Analytics\Project Challenges\Data In Motion Weekly Challenges\SQL Challenges\SQL Case Study 1\Tiny Shop.sqdbpro (Tiny Shop.db)

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Edit Pragma Browse Data Execute SQL

```
1 WITH CTE AS (  
2     SELECT  
3         order_items.order_id,  
4         SUM(products.price) * SUM(order_items.quantity) AS Order_Revenue  
5  
6     FROM order_items  
7     JOIN products ON  
8         order_items.product_id = products.product_id  
9  
10    GROUP BY order_items.order_id  
11 ),  
12 CTE2 AS (  
13     SELECT  
14         Order_Revenue,  
15         ROW_NUMBER() OVER (ORDER BY Order_Revenue) AS Row_Num,  
16         COUNT(*) OVER () AS Total_Count  
17  
18     FROM CTE  
19 )  
20 SELECT  
21     ROUND(AVG(Order_Revenue), 2) AS Median  
22  
23 FROM CTE2  
24  
25 WHERE  
26     Row_Num IN ((Total_Count + 1)/2, (Total_Count + 2)/2)  
27     OR (Total_Count % 2 = 1 AND Row_Num = (Total_Count + 1)/2)  
28
```

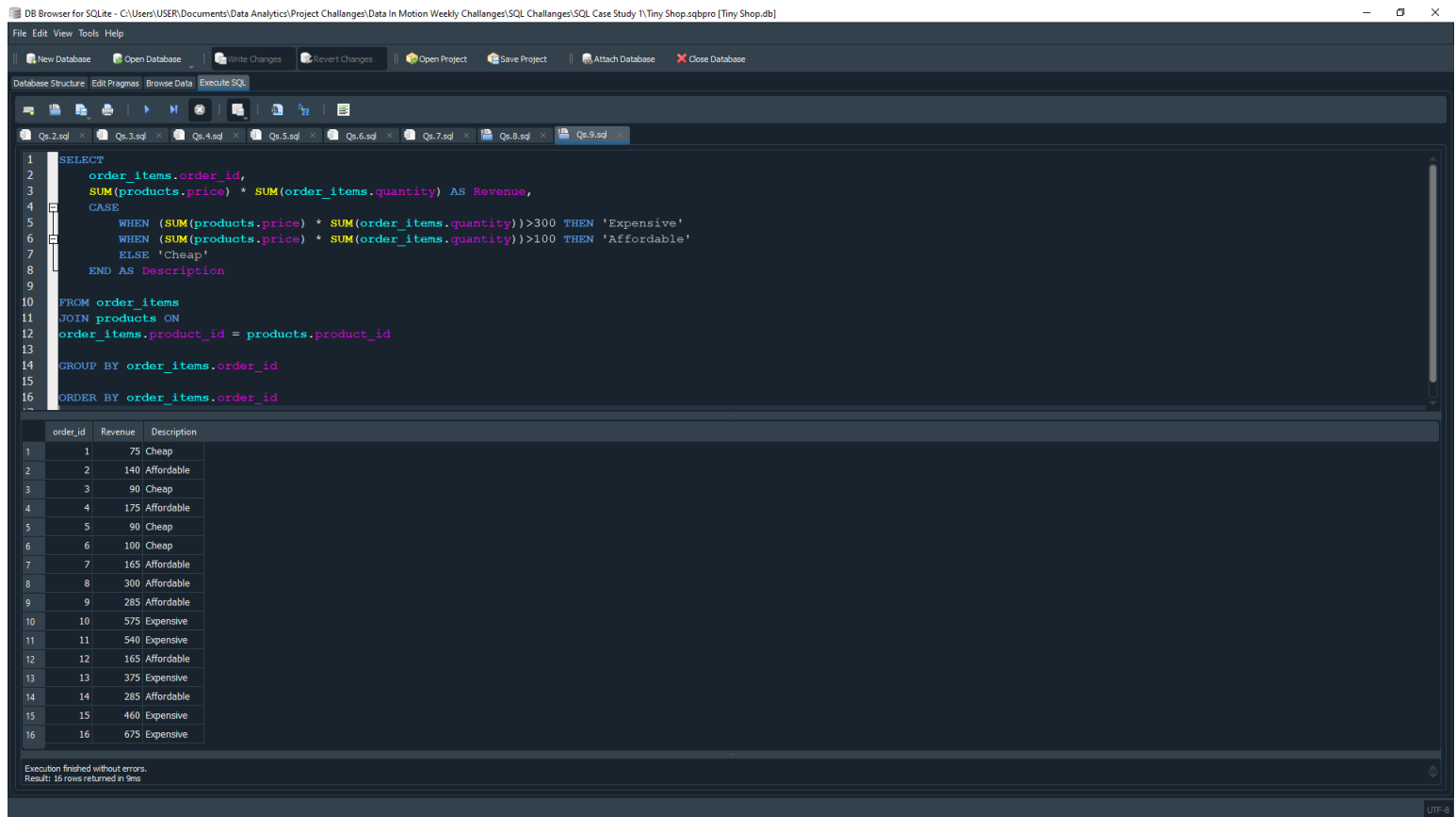
Median
1 230.0

Execution finished without errors.

UTF-8

The median order total is **230.0**.

QUESTION 9: FOR EACH ORDER, DETERMINE IF IT WAS 'EXPENSIVE' ('TOTAL OVER 300'), 'AFFORDABLE' ('TOTAL OVER 100'), OR 'CHEAP'.



The screenshot shows a DB Browser for SQLite window. The SQL editor contains the following query:

```
1 SELECT
2   order_items.order_id,
3   SUM(products.price) * SUM(order_items.quantity) AS Revenue,
4   CASE
5     WHEN (SUM(products.price) * SUM(order_items.quantity)) > 300 THEN 'Expensive'
6     WHEN (SUM(products.price) * SUM(order_items.quantity)) > 100 THEN 'Affordable'
7     ELSE 'Cheap'
8   END AS Description
9
10 FROM order_items
11 JOIN products ON
12   order_items.product_id = products.product_id
13
14 GROUP BY order_items.order_id
15
16 ORDER BY order_items.order_id
```

The results pane displays a table with 16 rows:

order_id	Revenue	Description
1	75	Cheap
2	140	Affordable
3	90	Cheap
4	175	Affordable
5	90	Cheap
6	100	Cheap
7	165	Affordable
8	300	Affordable
9	285	Affordable
10	575	Expensive
11	540	Expensive
12	165	Affordable
13	375	Expensive
14	285	Affordable
15	460	Expensive
16	675	Expensive

Execution finished without errors.
Results: 16 rows returned in 9ms

The results of determining if each order where either 'expensive', 'affordable' or "cheap are:

1. **Order_id 1:** Cheap
2. **Order_id 2:** Affordable
3. **Order_id 3:** Cheap
4. **Order_id 4:** Affordable
5. **Order_id 5:** Cheap
6. **Order_id 6:** Cheap
7. **Order_id 7:** Affordable
8. **Order_id 8:** Affordable
9. **Order_id 9:** Affordable
10. **Order_id 10:** Expensive
11. **Order_id 11:** Expensive
12. **Order_id 12:** Affordable
13. **Order_id 13:** Expensive
14. **Order_id 14:** Affordable
15. **Order_id 15:** Expensive
16. **Order_id 16:** Expensive

QUESTION 10: FIND THE CUSTOMERS WHO HAVE ORDER THE PRODUCT WITH THE HIGHEST PRICE.

DB Browser for SQLite - C:\Users\USER\Documents\Data Analytics\Project Challenges\Data in Motion Weekly Challenges\SQL Challenges\SQL Case Study 1\Tiny Shop.sqbp (Tiny Shop.db)

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Edit Pragma Browse Data Execute SQL

Qs.2.sql Qs.3.sql Qs.4.sql Qs.5.sql Qs.6.sql Qs.7.sql Qs.8.sql Qs.9.sql SQL 12

```
1 WITH CTE AS (
2     SELECT
3         orders.order_id,
4         customers.first_name || ' ' || customers.last_name AS Customers
5
6     FROM orders
7     JOIN customers
8     ON orders.customer_id = customers.customer_id
9
10    GROUP BY orders.order_id
11 )
12 SELECT
13     CTE.order_id,
14     CTE.Customers,
15     MAX(products.price) AS Maximum_Price
16
17 FROM order_items
18 JOIN products
19 JOIN CTE
20 ON order_items.product_id = products.product_id
21 AND order_items.order_id = CTE.order_id
22
23 GROUP BY CTE.customers
24 ORDER BY MAX(products.price) DESC
25
```

	order_id	Customers	Maximum_Price
1	16	Sophia Thomas	70
2	11	Ivy Jones	70
3	15	Quinn Roberts	60
4	10	George Harris	60
5	14	Oliver Patterson	50
6	9	Eva Fisher	50
7	13	Lily Nelson	40
8	8	Charlie Davis	40
9	12	Kevin Miller	30
10	7	Alice Brown	30
11	4	John Doe	20
12	2	Jane Smith	20
13	3	Bob Johnson	20

Execution finished without errors.
Elapsed: 13 msec returned in 0sec

UTF-8

The customers who have ordered the product with the highest price are:

- 1. **Sophia Thomas: \$70**
- 2. **Ivy Jones: \$70**