

SQL for Data Analysis

Name = Aditya Raj

Screenshots of the Queries and Output

➤ Use of SELECT, WHERE, ORDER BY, GROUP BY

✓ SELECT

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

```
1 SELECT *
2 FROM "E-commerce_Dataset"
3 LIMIT 10;
```

The output is a table with 15 columns: Order_Date, Time, Aging, Customer_Id, Gender, Device_Type, Customer_Login_type, Product_Category, Product, Sales, Quantity, Discount, Profit, Shipping_Cost, Order_Priority, and Payment_method. The first 10 rows of data are displayed.

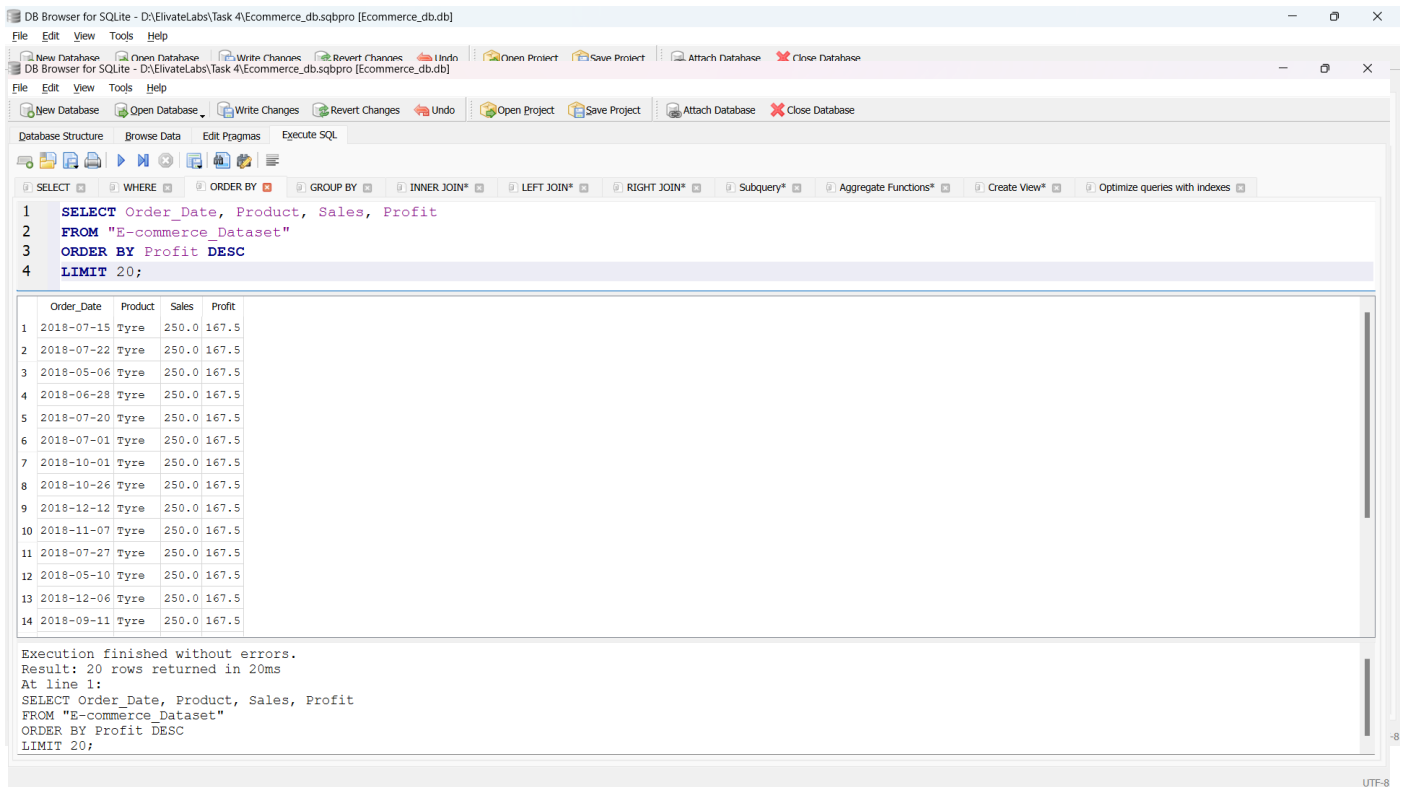
	Order_Date	Time	Aging	Customer_Id	Gender	Device_Type	Customer_Login_type	Product_Category	Product	Sales	Quantity	Discount	Profit	Shipping_Cost	Order_Priority	Payment_method
1	2018-01-02	10:56:33	8.0	37077	Female	Web	Member	Auto & Accessories	Car Media Players	140.0	1.0	0.3	46.0	4.6	Medium	credit_card
2	2018-07-24	20:41:37	2.0	59173	Female	Web	Member	Auto & Accessories	Car Speakers	211.0	1.0	0.3	112.0	11.2	Medium	credit_card
3	2018-11-08	08:38:49	8.0	41066	Female	Web	Member	Auto & Accessories	Car Body Covers	117.0	5.0	0.1	31.2	3.1	Critical	credit_card
4	2018-04-18	19:28:06	7.0	50741	Female	Web	Member	Auto & Accessories	Car & Bike Care	118.0	1.0	0.3	26.2	2.6	High	credit_card
5	2018-08-13	21:18:39	9.0	53639	Female	Web	Member	Auto & Accessories	Tyre	250.0	1.0	0.3	160.0	16.0	Critical	credit_card
6	2018-07-09	21:57:05	8.0	39783	Female	Web	Member	Auto & Accessories	Bike Tyres	72.0	1.0	0.3	24.0	2.4	Critical	credit_card
7	2018-05-16	13:10:30	1.0	26767	Female	Web	Member	Auto & Accessories	Car Mat	54.0	1.0	0.3	54.0	5.4	High	credit_card
8	2018-06-23	18:29:09	7.0	20719	Female	Web	Member	Auto & Accessories	Car Seat Covers	114.0	5.0	0.2	22.6	2.3	Critical	credit_card
9	2018-07-29	11:55:02	7.0	46947	Female	Web	Member	Auto & Accessories	Car Pillow & Neck Rest	231.0	5.0	0.3	116.4	11.6	Critical	credit_card
10	2018-05-16	19:41:10	10.0	31839	Female	Web	Member	Auto & Accessories	Car Media Players	140.0	1.0	0.2	54.4	5.4	Critical	money_order

Execution finished without errors.
Result: 10 rows returned in 8ms
At line 1:
SELECT *
FROM "E-commerce_Dataset"
LIMIT 10;

SQL for Data Analysis

Name = Aditya Raj

✓ WHERE



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

```
1 SELECT Order_Date, Product, Sales, Profit
2 FROM "E-commerce_Dataset"
3 ORDER BY Profit DESC
4 LIMIT 20;
```

The results pane displays a table with 14 rows of data:

	Order_Date	Product	Sales	Profit
1	2018-07-15	Tyre	250.0	167.5
2	2018-07-22	Tyre	250.0	167.5
3	2018-05-06	Tyre	250.0	167.5
4	2018-06-28	Tyre	250.0	167.5
5	2018-07-20	Tyre	250.0	167.5
6	2018-07-01	Tyre	250.0	167.5
7	2018-10-01	Tyre	250.0	167.5
8	2018-10-26	Tyre	250.0	167.5
9	2018-12-12	Tyre	250.0	167.5
10	2018-11-07	Tyre	250.0	167.5
11	2018-07-27	Tyre	250.0	167.5
12	2018-05-10	Tyre	250.0	167.5
13	2018-12-06	Tyre	250.0	167.5
14	2018-09-11	Tyre	250.0	167.5

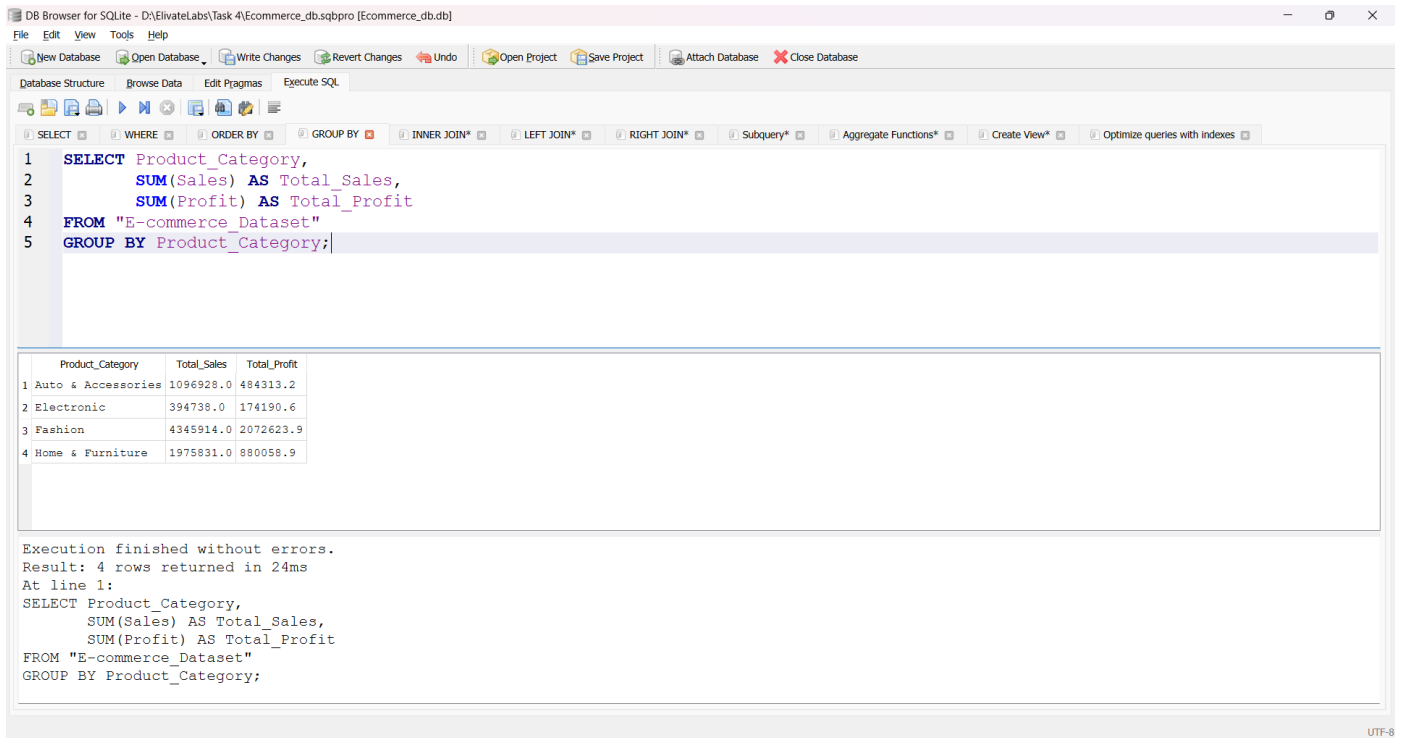
Execution finished without errors.
Result: 20 rows returned in 20ms
At line 1:
SELECT Order_Date, Product, Sales, Profit
FROM "E-commerce_Dataset"
ORDER BY Profit DESC
LIMIT 20;

✓ ORDER BY

SQL for Data Analysis

Name = Aditya Raj

✓ GROUP BY



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

```
1 SELECT Product_Category,
2       SUM(Sales) AS Total_Sales,
3       SUM(Profit) AS Total_Profit
4 FROM "E-commerce_Dataset"
5 GROUP BY Product_Category;
```

The results pane displays the following data:

	Product_Category	Total_Sales	Total_Profit
1	Auto & Accessories	1096928.0	484313.2
2	Electronic	394738.0	174190.6
3	Fashion	4345914.0	2072623.9
4	Home & Furniture	1975831.0	880058.9

Execution finished without errors.
Result: 4 rows returned in 24ms
At line 1:
SELECT Product_Category,
 SUM(Sales) AS Total_Sales,
 SUM(Profit) AS Total_Profit
FROM "E-commerce_Dataset"
GROUP BY Product_Category;

➤ Use of JOINS (INNER, LEFT, RIGHT)

SQL for Data Analysis

Name = Aditya Raj

✓ INNER JOIN

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

```
1 SELECT a.Customer_Id, a.Product AS Product_1, b.Product AS Product_2, a.Sales AS Sales_1, b.Sales AS Sales_2
2 FROM "E-commerce_Dataset" a
3 INNER JOIN "E-commerce_Dataset" b
4 ON a.Customer_Id = b.Customer_Id
5 AND a.Product != b.Product;
```

The results pane displays 14 rows of data:

	Customer_Id	Product_1	Product_2	Sales_1	Sales_2
1	39783	Bike Tyres	Jeans	72.0	218.0
2	18622	Car & Bike Care	Fossil Watch	118.0	159.0
3	56296	Tyre	Jeans	250.0	218.0
4	51112	Car Mat	Suits	54.0	109.0
5	50942	Car Pillow & Neck Rest	Speakers	231.0	130.0
6	42384	Car Media Players	Formal Shoes	140.0	213.0
7	51542	Bike Tyres	Shirts	72.0	196.0
8	42676	Car Mat	T - Shirts	54.0	248.0
9	37689	Car Pillow & Neck Rest	Mixer/Juicer	231.0	83.0
10	26058	Car Media Players	Shirts	140.0	196.0
11	56625	Car & Bike Care	Tyre	118.0	250.0
12	14567	Car Seat Covers	Sports Wear	114.0	85.0
13	38178	Car Pillow & Neck Rest	Car Body Covers	231.0	117.0
14	24153	Car Media Players	Bike Tyres	140.0	72.0

Execution finished without errors.
Result: 28028 rows returned in 176ms
At line 1:
SELECT a.Customer_Id, a.Product AS Product_1, b.Product AS Product_2, a.Sales AS Sales_1, b.Sales AS Sales_2
FROM "E-commerce_Dataset" a
INNER JOIN "E-commerce_Dataset" b
ON a.Customer_Id = b.Customer_Id
AND a.Product != b.Product;

✓ LEFT JOIN

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

```
1 SELECT a.Customer_Id, a.Product AS Product_1, b.Product AS Product_2
2 FROM "E-commerce_Dataset" a
3 LEFT JOIN "E-commerce_Dataset" b
4 ON a.Customer_Id = b.Customer_Id
5 AND a.Product != b.Product;
```

The results pane displays 14 rows of data:

	Customer_Id	Product_1	Product_2
1	37077	Car Media Players	NULL
2	59173	Car Speakers	NULL
3	41066	Car Body Covers	NULL
4	50741	Car & Bike Care	NULL
5	53639	Tyre	NULL
6	39783	Bike Tyres	Jeans
7	26767	Car Mat	NULL
8	20719	Car Seat Covers	NULL
9	46947	Car Pillow & Neck Rest	NULL
10	31839	Car Media Players	NULL
11	22249	Car Speakers	NULL
12	15109	Car Body Covers	NULL
13	18622	Car & Bike Care	Fossil Watch
14	56296	Tyre	Jeans

Execution finished without errors.
Result: 57636 rows returned in 240ms
At line 1:
SELECT a.Customer_Id, a.Product AS Product_1, b.Product AS Product_2
FROM "E-commerce_Dataset" a
LEFT JOIN "E-commerce_Dataset" b
ON a.Customer_Id = b.Customer_Id
AND a.Product != b.Product;

SQL for Data Analysis

Name = Aditya Raj

✓ RIGHT JOIN

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

```
1 SELECT a.Customer_Id, a.Product AS Product_1, b.Product AS Product_2
2 FROM "E-commerce_Dataset" b
3 LEFT JOIN "E-commerce_Dataset" a
4 ON a.Customer_Id = b.Customer_Id
5 AND a.Product != b.Product;
```

The results pane displays 14 rows of data:

	Customer_Id	Product_1	Product_2
1	NULL	NULL	Car Media Players
2	NULL	NULL	Car Speakers
3	NULL	NULL	Car Body Covers
4	NULL	NULL	Car & Bike Care
5	NULL	NULL	Tyre
6	39783	Jeans	Bike Tyres
7	NULL	NULL	Car Mat
8	NULL	NULL	Car Seat Covers
9	NULL	NULL	Car Pillow & Neck Rest
10	NULL	NULL	Car Media Players
11	NULL	NULL	Car Speakers
12	NULL	NULL	Car Body Covers
13	18622	Fossil Watch	Car & Bike Care
14	56296	Jeans	Tyre

Execution finished without errors. Result: 57636 rows returned in 244ms. At line 1: SELECT a.Customer_Id, a.Product AS Product_1, b.Product AS Product_2 FROM "E-commerce_Dataset" b LEFT JOIN "E-commerce_Dataset" a ON a.Customer_Id = b.Customer_Id AND a.Product != b.Product;

➤ Subqueries

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

```
1 SELECT Product, SUM(Sales) AS Total_Sales
2 FROM "E-commerce_Dataset"
3 GROUP BY Product
4 HAVING Total_Sales > (
5     SELECT AVG(Sales) FROM "E-commerce_Dataset"
6 );
```

The results pane displays 13 rows of data:

	Product	Total_Sales
1	Apple Laptop	55250.0
2	Bed Sheets	325151.0
3	Beds	120276.0
4	Bike Tyres	59472.0
5	Car & Bike Care	97468.0
6	Car Body Covers	96642.0
7	Car Mat	44604.0
8	Car Media Players	115640.0
9	Car Pillow & Neck Rest	191499.0
10	Car Seat Covers	94278.0
11	Car Speakers	174075.0
12	Casual Shoes	284382.0
13	Curtains	52394.0

Execution finished without errors. Result: 42 rows returned in 45ms. At line 1: SELECT Product, SUM(Sales) AS Total_Sales FROM "E-commerce_Dataset" GROUP BY Product HAVING Total_Sales > (SELECT AVG(Sales) FROM "E-commerce_Dataset");

SQL for Data Analysis

Name = Aditya Raj

➤ Aggregate Functions (SUM, AVG)

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

```
1 SELECT Payment_method,
2       SUM(Sales) AS Total_Sales,
3       AVG(Sales) AS Average_Sales,
4       COUNT(*) AS Total_Orders
5 FROM "E-commerce_Dataset"
6 GROUP BY Payment_method;
```

The results are displayed in a table with 4 columns: Payment_method, Total_Sales, Average_Sales, and Total_Orders.

Payment_method	Total_Sales	Average_Sales	Total_Orders
1 credit_card	5819379.0	152.595421648836	38137
2 debit_card	109979.0	149.83514986376	734
3 e_wallet	422750.0	151.577626389387	2789
4 money_order	1461269.0	151.757087963444	9629
5 not_defined	34.0	34.0	1

Execution finished without errors.
Result: 5 rows returned in 45ms
At line 1:
SELECT Payment_method,
SUM(Sales) AS Total_Sales,
AVG(Sales) AS Average_Sales,
COUNT(*) AS Total_Orders
FROM "E-commerce_Dataset"
GROUP BY Payment_method;

➤ Creating views for analysis

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

```
1 CREATE VIEW high_value_orders AS
2 SELECT *
3 FROM "E-commerce_Dataset"
4 WHERE Sales > 200;
5
6 SELECT *
7 FROM high_value_orders;
```

The results are displayed in a table with 16 columns: Order_Date, Time, Aging, Customer_Id, Gender, Device_Type, Customer_Login_type, Product_Category, Product, Sales, Quantity, Discount, Profit, Shipping_Cost, Order_Priority, and Payment_method.

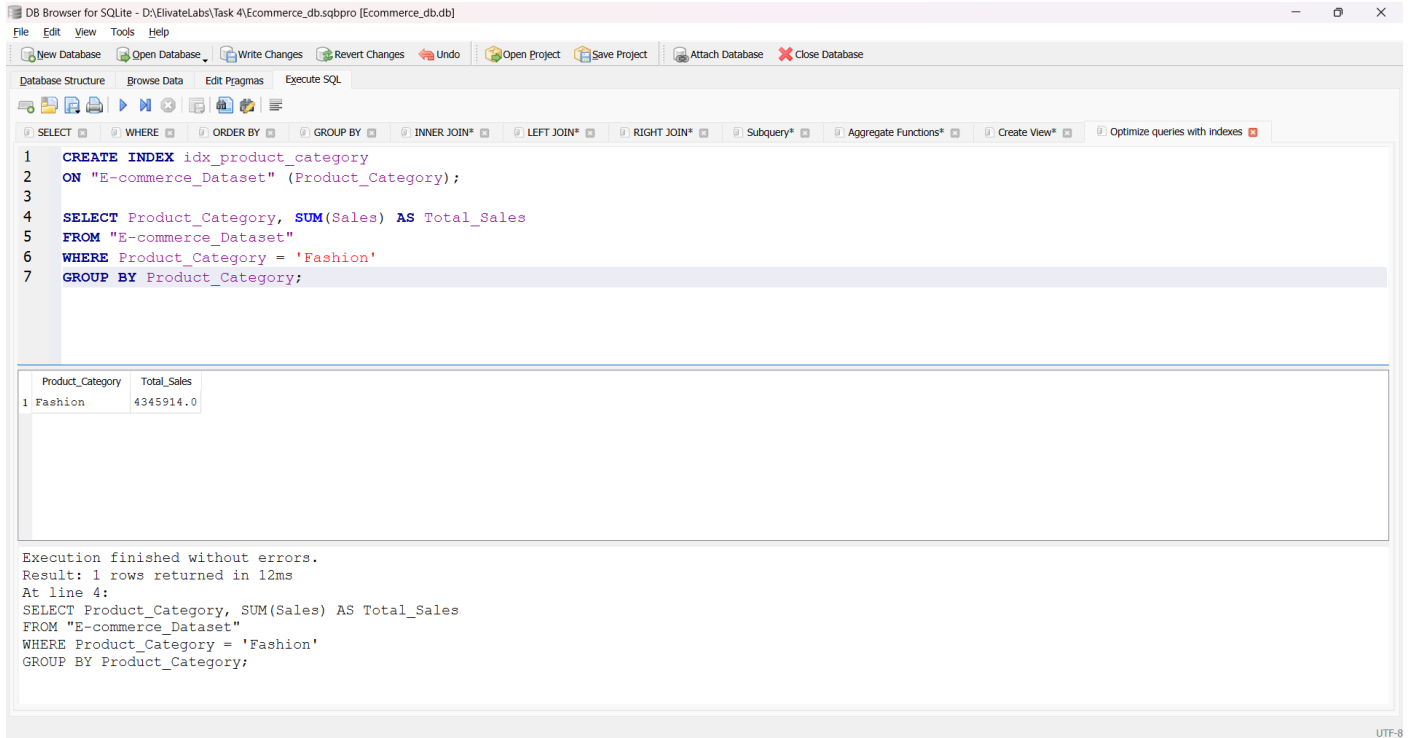
	Order_Date	Time	Aging	Customer_Id	Gender	Device_Type	Customer_Login_type	Product_Category	Product	Sales	Quantity	Discount	Profit	Shipping_Cost	Order_Priority	Payment_method
1	2018-07-24	20:41:37	2.0	59173	Female	Web	Member	Auto & Accessories	Car Speakers	211.0	1.0	0.3	112.0	11.2	Medium	credit_card
2	2018-08-13	21:18:39	9.0	53639	Female	Web	Member	Auto & Accessories	Tyre	250.0	1.0	0.3	160.0	16.0	Critical	credit_card
3	2018-07-29	11:55:02	7.0	46947	Female	Web	Member	Auto & Accessories	Car Pillow & Neck Rest	231.0	5.0	0.3	116.4	11.6	Critical	credit_card
4	2018-07-13	19:58:11	10.0	22249	Female	Web	Member	Auto & Accessories	Car Speakers	211.0	4.0	0.1	122.6	12.3	Critical	credit_card
5	2018-12-01	14:14:28	5.0	56296	Female	Web	Member	Auto & Accessories	Tyre	250.0	1.0	0.3	140.0	14.0	High	credit_card
6	2018-04-21	00:03:41	7.0	50942	Female	Web	Member	Auto & Accessories	Car Pillow & Neck Rest	231.0	5.0	0.3	93.3	9.3	High	credit_card
7	2018-02-18	14:31:30	6.0	26127	Female	Web	Member	Auto & Accessories	Car Speakers	211.0	1.0	0.2	122.6	12.3	Critical	credit_card
8	2018-06-08	07:55:29	6.0	15976	Female	Web	Member	Auto & Accessories	Tyre	250.0	4.0	0.2	150.0	15.0	High	credit_card
9	2018-12-12	01:50:26	8.0	37689	Female	Web	Member	Auto & Accessories	Car Pillow & Neck Rest	231.0	1.0	0.3	144.1	14.4	Critical	credit_card
10	2018-08-25	13:50:30	7.0	38941	Female	Web	Member	Auto & Accessories	Car Speakers	211.0	1.0	0.3	112.0	11.2	Medium	credit_card
11	2018-03-12	13:56:04	5.0	33501	Female	Web	Member	Auto & Accessories	Tyre	250.0	1.0	0.2	165.0	16.5	Medium	credit_card
12	2018-08-21	18:15:12	4.0	38178	Female	Web	Member	Auto & Accessories	Car Pillow & Neck Rest	231.0	1.0	0.3	127.9	12.8	High	credit_card
13	2018-10-01	12:23:13	3.0	27385	Female	Web	Member	Auto & Accessories	Car Speakers	211.0	4.0	0.1	122.6	12.3	Critical	credit_card
14	2018-04-24	21:27:48	4.0	30549	Female	Web	Member	Auto & Accessories	Tyre	250.0	5.0	0.3	132.5	13.3	Critical	credit_card

Execution finished without errors.
Result: 19488 rows returned in 295ms
At line 6:
SELECT *
FROM high_value_orders;

SQL for Data Analysis

Name = Aditya Raj

➤ Optimizing queries with indexes



The screenshot shows the D8 Browser for SQLite interface. The main window displays the following SQL code:

```
1 CREATE INDEX idx_product_category
2 ON "E-commerce_Dataset" (Product_Category);
3
4 SELECT Product_Category, SUM(Sales) AS Total_Sales
5 FROM "E-commerce_Dataset"
6 WHERE Product_Category = 'Fashion'
7 GROUP BY Product_Category;
```

Below the code editor, the results of the query are displayed in a table:

Product_Category	Total_Sales
1 Fashion	4345914.0

Execution finished without errors.
Result: 1 rows returned in 12ms
At line 4:
SELECT Product_Category, SUM(Sales) AS Total_Sales
FROM "E-commerce_Dataset"
WHERE Product_Category = 'Fashion'
GROUP BY Product_Category;

UTF-8