

This query is to check the data:

```
SELECT * FROM "Order Details" LIMIT 10;
```

The screenshot shows a SQL IDE interface with a menu bar (File, Edit, View, Tools, Help) and a toolbar. The main window displays a SQL script with 12 lines. Line 1 is highlighted: `SELECT * FROM "Order Details" LIMIT 10;`. Below the script, the results of the query are shown in a table with 7 columns: Order ID, Amount, Profit, Quantity, Category, and Sub-Category. The table contains 7 rows of data. Below the table, a status message indicates: "Execution finished without errors. Result: 10 rows returned in 14ms. At line 1: SELECT * FROM "Order Details" LIMIT 10;".

Order ID	Amount	Profit	Quantity	Category	Sub-Category
B-25601	1275.0	-1148.0	7	Furniture	Bookcases
B-25601	66.0	-12.0	5	Clothing	Stole
B-25601	8.0	-2.0	3	Clothing	Hankerchief
B-25601	80.0	-56.0	4	Electronics	Electronic Games
B-25602	168.0	-111.0	2	Electronics	Phones
B-25602	424.0	-272.0	5	Electronics	Phones
B-25602	2617.0	1151.0	4	Electronics	Phones

Query 1: Total number of orders

```
SELECT COUNT(*) AS Total_Order FR OM "Order Details";
```

The screenshot shows a SQL IDE interface with a menu bar (File, Edit, View, Tools, Help) and a toolbar. The main window displays a SQL script with 12 lines. Line 3 is highlighted: `SELECT COUNT(*) AS Total_Order FROM "Order Details";`. Below the script, the results of the query are shown in a table with 1 column: Total_Order. The table contains 1 row of data: 1500. Below the table, a status message indicates: "Execution finished without errors. Result: 1 rows returned in 11ms. At line 3: SELECT COUNT(*) AS Total_Order FROM "Order Details";".

Total_Order
1500

Query 2: Total revenue (sum of Amount column)

```
SELECT SUM(Amount) AS Total_ Revenue FROM "Order Details";
```

The screenshot shows a SQL IDE interface with a menu bar (File, Edit, View, Tools, Help) and a toolbar. The main window displays a SQL script with 12 lines. Line 6 is highlighted: `SELECT SUM(Amount) AS Total_ Revenue FROM "Order Details";`. Below the script, the results of the query are shown in a table with 1 column: Total_Revenue. The table contains 1 row of data: 431502.0. Below the table, a status message indicates: "Execution finished without errors. Result: 1 rows returned in 11ms. At line 6: SELECT SUM(Amount) AS Total_ Revenue FROM "Order Details";".

Total_Revenue
431502.0

Query 3: Total profit generated

```
SELECT SUM(Profit) AS Total_Profit FROM "Order Details";
```

The screenshot shows a SQL IDE interface with a menu bar (File, Edit, View, Tools, Help) and a toolbar. The main window displays a SQL query in a text editor and its execution results in a table. The query is:

```
1 SELECT * FROM "Order Details" LIMIT 10;
2
3 SELECT COUNT(*) AS Total_Orders
4 FROM "Order Details";
5
6 SELECT SUM(Amount) AS Total_Revenue
7 FROM "Order Details";
8
9 SELECT SUM(Profit) AS Total_Profit
10 FROM "Order Details";
11
12 SELECT "Order ID", SUM(Amount) AS Total_Sales
```

The results table shows the total profit:

Total_Profit
1 23955.0

Below the table, the execution status is displayed:

```
Execution finished without errors.
Result: 1 rows returned in 9ms
At line 9:
SELECT SUM(Profit) AS Total_Profit
FROM "Order Details";
```

Query 4: Best-selling product (based on total revenue)

```
SELECT "Order ID", SUM(Amount) AS Total_Sales FROM "Order Details" GROUP BY
"Order ID" ORDER BY Total_Sales DESC LIMIT 1;
```

The screenshot shows a SQL IDE interface with a menu bar (File, Edit, View, Tools, Help) and a toolbar. The main window displays a SQL query in a text editor and its execution results in a table. The query is:

```
7 FROM "Order Details";
8
9 SELECT SUM(Profit) AS Total_Profit
10 FROM "Order Details";
11
12 SELECT "Order ID", SUM(Amount) AS Total_Sales
13 FROM "Order Details"
14 GROUP BY "Order ID"
15 ORDER BY Total_Sales DESC
16 LIMIT 5;
17
18 SELECT "Order ID", SUM(Profit) AS Total_Profit
```

The results table shows the top 5 best-selling products:

Order ID	Total_Sales
1 B-26055	8502.0
2 B-25955	6339.0
3 B-25993	6026.0
4 B-25881	5809.0
5 B-25973	5228.0

Below the table, the execution status is displayed:

```
Execution finished without errors.
Result: 5 rows returned in 9ms
At line 12:
SELECT "Order ID", SUM(Amount) AS Total_Sales
FROM "Order Details"
GROUP BY "Order ID"
ORDER BY Total_Sales DESC
LIMIT 5;
```

Query 5: Highest profit item

```
SELECT "Order ID", SUM(Profit) AS Total_Profit FROM "Order Details" GROUP BY
"Order ID" ORDER BY Total_Profit DESC LIMIT 5;
```

The screenshot shows a SQL IDE interface with a menu bar (File, Edit, View, Tools, Help) and a toolbar. The main window displays a SQL query in a text editor and its execution results in a table. The query is:

```
33
34 SELECT "Order ID", SUM(Amount) AS Total_Revenue
35 FROM "Order Details"
36 GROUP BY "Order ID"
37 ORDER BY Total_Revenue DESC
38 LIMIT 5;
39
40 SELECT "Order ID", SUM(Profit) AS Total_Loss
41 FROM "Order Details"
42 GROUP BY "Order ID"
43 ORDER BY Total_Loss ASC
44 LIMIT 5;
```

The results table shows the top 5 highest profit items:

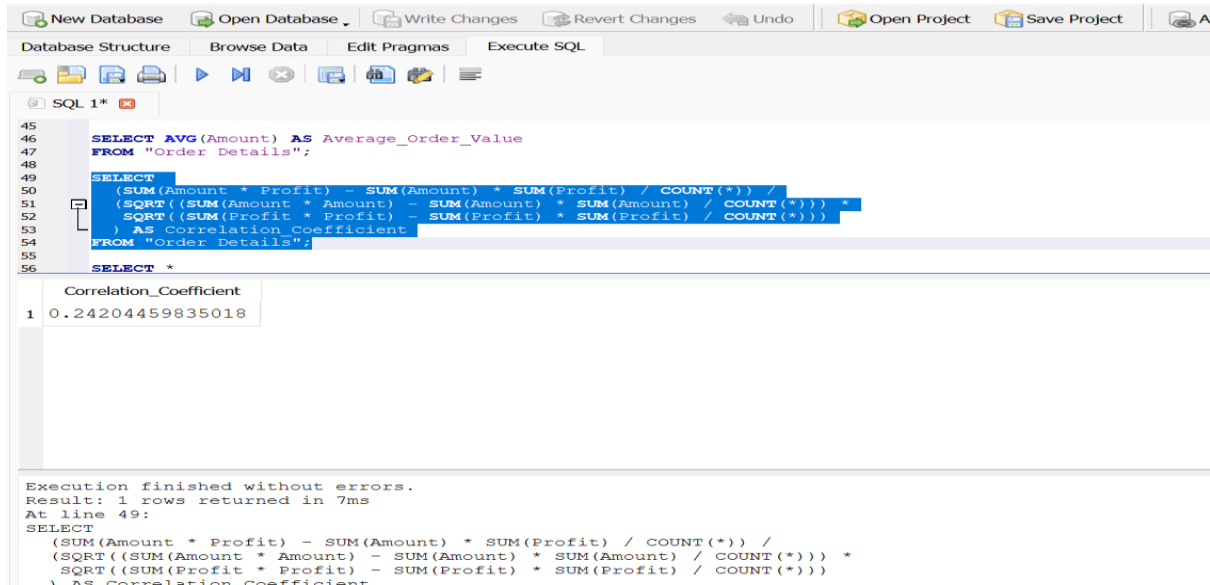
Order ID	Total_Loss
1 B-25798	-1836.0
2 B-25608	-1456.0
3 B-26022	-1303.0
4 B-25601	-1218.0
5 B-25779	-980.0

Below the table, the execution status is displayed:

```
Execution finished without errors.
Result: 5 rows returned in 9ms
At line 40:
SELECT "Order ID", SUM(Profit) AS Total_Loss
FROM "Order Details"
GROUP BY "Order ID"
ORDER BY Total_Loss ASC
LIMIT 5;
```

Query 6: Count of profitable vs loss orders

```
SELECT CASE WHEN Profit > 0 THEN 'Profit' WHEN Profit < 0 THEN 'Loss' ELSE  
'Break-even' END AS Category, COUNT(*) AS Count_Orders FROM "OrderDetails"  
GROUP BY Category;
```



SQL 1*

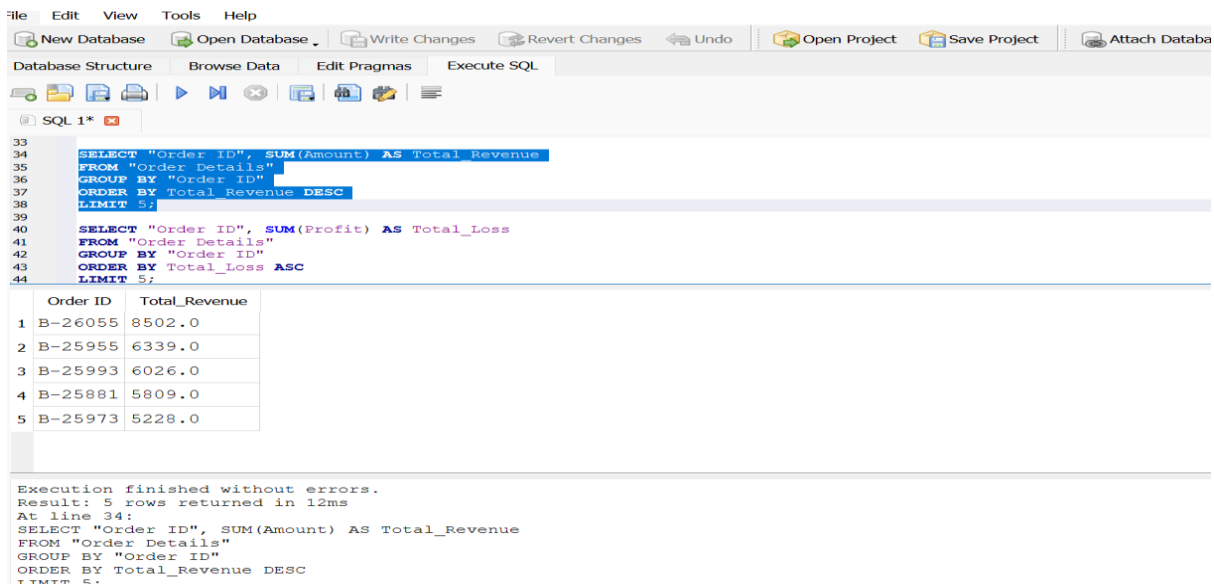
```
45  
46 SELECT AVG(Amount) AS Average_Order_Value  
47 FROM "Order Details";  
48  
49 SELECT  
50 (SUM(Amount * Profit) - SUM(Amount) * SUM(Profit) / COUNT(*)) /  
51 (SQRT((SUM(Amount * Amount) - SUM(Amount) * SUM(Amount) / COUNT(*)) *  
52 (SUM(Profit * Profit) - SUM(Profit) * SUM(Profit) / COUNT(*)))  
53 ) AS Correlation_Coefficient  
54 FROM "Order Details";  
55  
56 SELECT *
```

	Correlation_Coefficient
1	0.24204459835018

Execution finished without errors.
Result: 1 rows returned in 7ms
At line 49:
SELECT
(SUM(Amount * Profit) - SUM(Amount) * SUM(Profit) / COUNT(*)) /
(SQRT((SUM(Amount * Amount) - SUM(Amount) * SUM(Amount) / COUNT(*)) *
SQRT((SUM(Profit * Profit) - SUM(Profit) * SUM(Profit) / COUNT(*)))
) AS Correlation_Coefficient

Query 7: Top 5 highest revenue orders

```
SELECT "Order ID", SUM(Amount) AS Total_Revenue FROM "Order Details"  
GROUP BY "Order ID" ORDER BY Total_Revenue DESC LIMIT 5;
```



SQL 1*

```
33  
34 SELECT "Order ID", SUM(Amount) AS Total_Revenue  
35 FROM "Order Details"  
36 GROUP BY "Order ID"  
37 ORDER BY Total_Revenue DESC  
38 LIMIT 5;  
39  
40 SELECT "Order ID", SUM(Profit) AS Total_Loss  
41 FROM "Order Details"  
42 GROUP BY "Order ID"  
43 ORDER BY Total_Loss ASC  
44 LIMIT 5;
```

	Order ID	Total_Revenue
1	B-26055	8502.0
2	B-25955	6339.0
3	B-25993	6026.0
4	B-25881	5809.0
5	B-25973	5228.0

Execution finished without errors.
Result: 5 rows returned in 12ms
At line 34:
SELECT "Order ID", SUM(Amount) AS Total_Revenue
FROM "Order Details"
GROUP BY "Order ID"
ORDER BY Total_Revenue DESC
LIMIT 5;

Query 8 :Top 5 loss-making orders

```
SELECT "Order ID", SUM(Profit) AS Total_Loss FROM "Order Details" GROUP BY  
"Order ID" ORDER BY Total_Loss ASC LIMIT 5;
```

```

File Edit View Tools Help
New Database Open Database Write Changes Revert Changes Undo Open Project Save
Database Structure Browse Data Edit Pragmas Execute SQL
SQL 1*
33
34 SELECT "Order ID", SUM(Amount) AS Total_Revenue
35 FROM "Order Details"
36 GROUP BY "Order ID"
37 ORDER BY Total_Revenue DESC
38 LIMIT 5;
39
40 SELECT "Order ID", SUM(Profit) AS Total_Loss
41 FROM "Order Details"
42 GROUP BY "Order ID"
43 ORDER BY Total_Loss ASC
44 LIMIT 5;

```

	Order ID	Total_Loss
1	B-25798	-1836.0
2	B-25608	-1456.0
3	B-26022	-1303.0
4	B-25601	-1218.0
5	B-25779	-980.0

```

Execution finished without errors.
Result: 5 rows returned in 9ms
At line 40:
SELECT "Order ID", SUM(Profit) AS Total_Loss
FROM "Order Details"
GROUP BY "Order ID"
ORDER BY Total_Loss ASC
LIMIT 5;

```

Query 9: Average order value

SELECT AVG(Amount) AS Average_Order_Value FROM "Order Details";

```

File Edit View Tools Help
New Database Open Database Write Changes Revert Changes Undo Open Project Save Project Attach Data
Database Structure Browse Data Edit Pragmas Execute SQL
SQL 1*
42 GROUP BY "Order ID"
43 ORDER BY Total_Loss ASC
44 LIMIT 5;
45
46 SELECT AVG(Amount) AS Average_Order_Value
47 FROM "Order Details";
48
49 SELECT
50 (SUM(Amount * Profit) - SUM(Amount) * SUM(Profit) / COUNT(*)) /
51 (SQRT((SUM(Amount * Amount) - SUM(Amount) * SUM(Amount) / COUNT(*))) *
52 SQRT((SUM(Profit * Profit) - SUM(Profit) * SUM(Profit) / COUNT(*))) ) AS
53 Correlation_Coefficient

```

	Average_Order_Value
1	287.668

```

Execution finished without errors.
Result: 1 rows returned in 9ms
At line 46:
SELECT AVG(Amount) AS Average_Order_Value
FROM "Order Details";

```

Query 10: Correlation between Amount and Profit (Insight)

SELECT (SUM(Amount * Profit) - SUM(Amount) * SUM(Profit) / COUNT(*)) /
 (SQRT((SUM(Amount * Amount) - SUM(Amount) * SUM(Amount) / COUNT(*))) *
 SQRT((SUM(Profit * Profit) - SUM(Profit) * SUM(Profit) / COUNT(*)))) AS
 Correlation_Coefficient FROM "Order Details";

SQL 1*

```

45 SELECT AVG(Amount) AS Average_Order_Value
46 FROM "Order Details";
47
48
49
50 SELECT
51 (SUM(Amount * Profit) - SUM(Amount) * SUM(Profit) / COUNT(*)) /
52 (SQRT((SUM(Amount * Amount) - SUM(Amount) * SUM(Amount) / COUNT(*))) *
53 (SQRT((SUM(Profit * Profit) - SUM(Profit) * SUM(Profit) / COUNT(*))) *
54 ) AS Correlation_Coefficient
55 FROM "Order Details";
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

Correlation_Coefficient

1	0.24204459835018
---	------------------

Execution finished without errors.
Result: 1 rows returned in 7ms
At line 49:
SELECT
(SUM(Amount * Profit) - SUM(Amount) * SUM(Profit) / COUNT(*)) /
(SQRT((SUM(Amount * Amount) - SUM(Amount) * SUM(Amount) / COUNT(*))) *
SQRT((SUM(Profit * Profit) - SUM(Profit) * SUM(Profit) / COUNT(*)))
) AS Correlation_Coefficient

Query 11: Show all columns sorted by highest profit

SELECT *FROM "Order Details" ORDER BY Amount DESC limit 25;

SQL 1*

```

49 SELECT
50 (SUM(Amount * Profit) - SUM(Amount) * SUM(Profit) / COUNT(*)) /
51 (SQRT((SUM(Amount * Amount) - SUM(Amount) * SUM(Amount) / COUNT(*))) *
52 (SQRT((SUM(Profit * Profit) - SUM(Profit) * SUM(Profit) / COUNT(*))) *
53 ) AS Correlation_Coefficient
54 FROM "Order Details";
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

FROM "Order Details"
ORDER BY Amount DESC limit 25;

	Order ID	Amount	Profit	Quantity	Category	Sub-Category
1	B-26055	5729.0	64.0	14	Furniture	Chairs
2	B-25993	4363.0	305.0	5	Furniture	Tables
3	B-25973	4141.0	1698.0	13	Electronics	Printers
4	B-25923	3873.0	891.0	6	Electronics	Phones
5	B-25757	3151.0	-35.0	7	Clothing	Trousers
6	B-25955	2927.0	146.0	8	Furniture	Bookcases
7	B-26093	2847.0	712.0	8	Electronics	Printers

Execution finished without errors.
Result: 25 rows returned in 27ms
At line 56:
SELECT
FROM "Order Details"
ORDER BY Amount DESC limit 25;

Query 12: Show all columns sorted by highest revenue

SELECT *FROM "Order Details" ORDER BY Profit DESC limit 50;

SQL 1*

```

53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

FROM "Order Details"
ORDER BY Amount DESC limit 25;
FROM "Order Details"
ORDER BY Profit DESC limit 50;

	Order ID	Amount	Profit	Quantity	Category	Sub-Category
1	B-25973	4141.0	1698.0	13	Electronics	Printers
2	B-25602	2617.0	1151.0	4	Electronics	Phones
3	B-25761	2188.0	1050.0	5	Furniture	Bookcases
4	B-25923	3873.0	891.0	6	Electronics	Phones
5	B-25830	1954.0	782.0	3	Electronics	Phones
6	B-26073	1514.0	742.0	4	Electronics	Printers
7	B-25853	2093.0	721.0	5	Furniture	Chairs

Execution finished without errors.
Result: 50 rows returned in 16ms
At line 60:
SELECT
FROM "Order Details"
ORDER BY Profit DESC limit 50;

