

Practical 1: SQL Fundamentals (Databricks-Basic SQL Syntax)

1. SELECT Statement

Q1. Display all columns for all transactions.

Expected output: All columns

The screenshot shows the Databricks SQL interface. The query editor contains the following SQL code:

```
1 select * from retail_sales;
```

The results are displayed in a table with 10 columns: Transaction ID, Date, Customer ID, Gender, Age, Product Category, Quantity, Price per Unit, and Total Amount. The table contains 8 rows of data.

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
1	1	2023-11-24	CUST001	Male	34	Beauty	3	50	150
2	2	2023-02-27	CUST002	Female	26	Clothing	2	500	1000
3	3	2023-01-13	CUST003	Male	50	Electronics	1	30	30
4	4	2023-05-21	CUST004	Male	37	Clothing	1	500	500
5	5	2023-05-06	CUST005	Male	30	Beauty	2	50	100
6	6	2023-04-25	CUST006	Female	45	Beauty	1	30	30
7	7	2023-03-13	CUST007	Male	46	Clothing	2	25	50
8	8	2023-02-22	CUST008	Male	30	Electronics	4	25	100

722 ms | 1,000 rows returned

Q2. Display only the Transaction ID, Date, and Customer ID for all records.

Expected output: Transaction ID, Date, Customer ID

The screenshot shows the Databricks SQL interface. The query editor contains the following SQL code:

```
1 select 'Transaction ID',  
2 'Date',  
3 'Customer ID'  
4 from retail_sales;
```

The results are displayed in a table with 4 columns: Transaction ID, Date, and Customer ID. The table contains 8 rows of data.

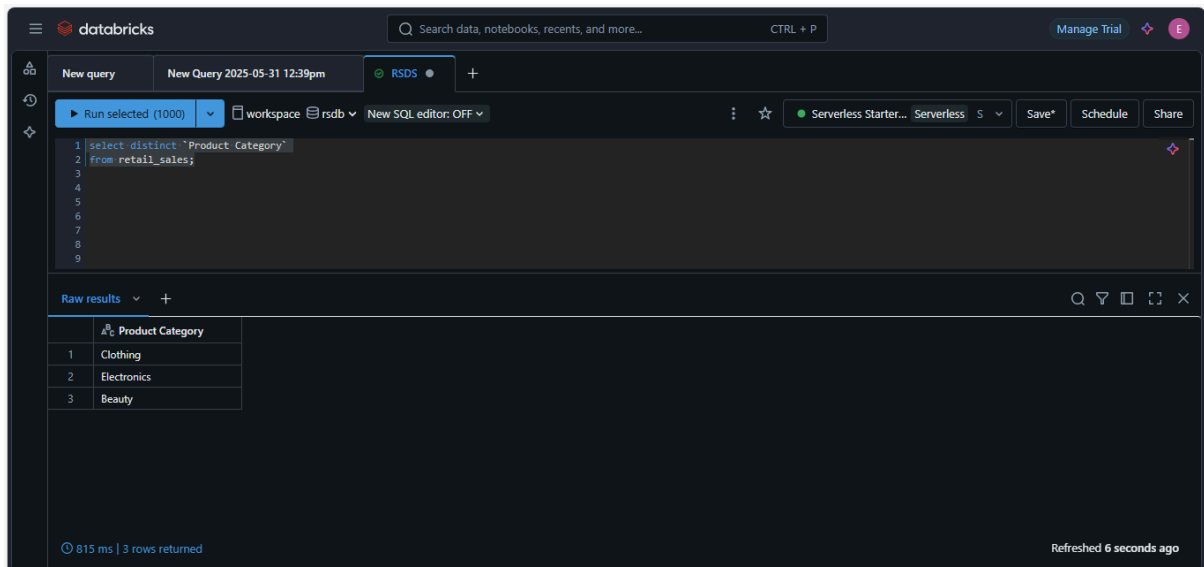
	Transaction ID	Date	Customer ID
1	1	2023-11-24	CUST001
2	2	2023-02-27	CUST002
3	3	2023-01-13	CUST003
4	4	2023-05-21	CUST004
5	5	2023-05-06	CUST005
6	6	2023-04-25	CUST006
7	7	2023-03-13	CUST007
8	8	2023-02-22	CUST008

1 s 681 ms | 1,000 rows returned

2. SELECT DISTINCT Statement

Q3. Display all the distinct product categories in the dataset.

Expected output: Product Category



The screenshot shows the Databricks workspace with a new query editor. The SQL query is:

```
1 select distinct "Product Category"
2 from retail_sales;
```

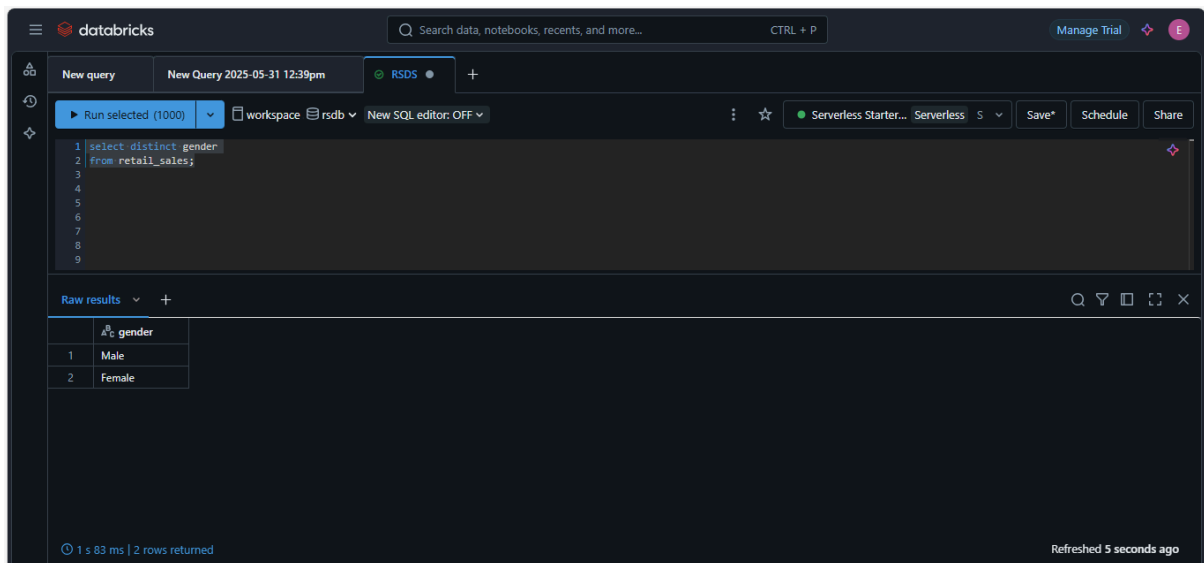
The query is executed, and the results are displayed in a table with the following data:

	Product Category
1	Clothing
2	Electronics
3	Beauty

At the bottom, it indicates "815 ms | 3 rows returned" and "Refreshed 6 seconds ago".

Q4. Display all the distinct gender values in the dataset.

Expected output: Gender



The screenshot shows the Databricks workspace with a new query editor. The SQL query is:

```
1 select distinct gender
2 from retail_sales;
```

The query is executed, and the results are displayed in a table with the following data:

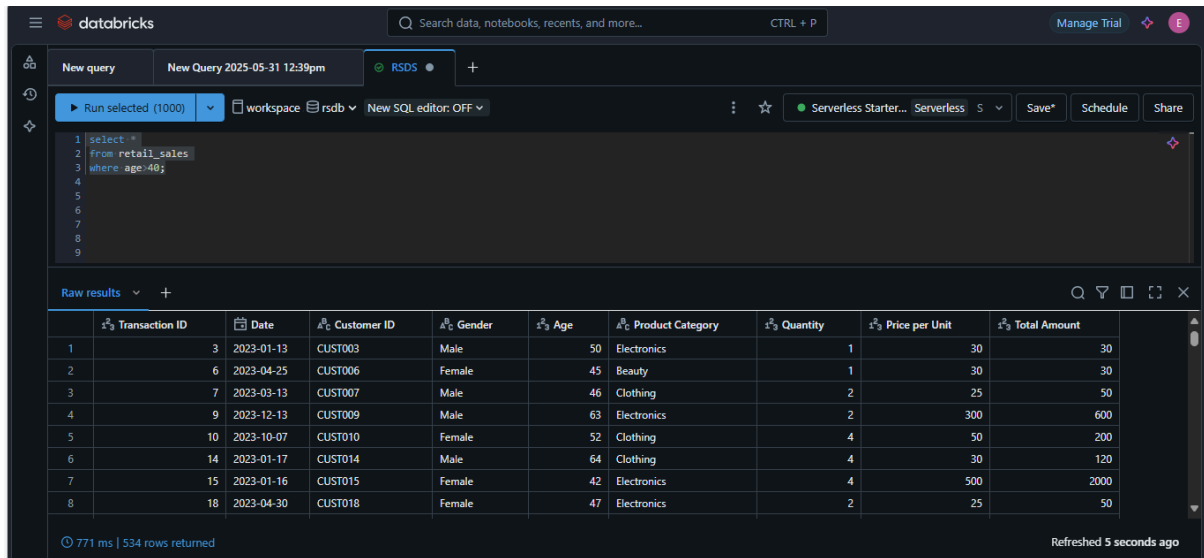
	gender
1	Male
2	Female

At the bottom, it indicates "1 s 83 ms | 2 rows returned" and "Refreshed 5 seconds ago".

3. WHERE Clause

Q5. Display all transactions where the Age is greater than 40.

Expected output: All columns

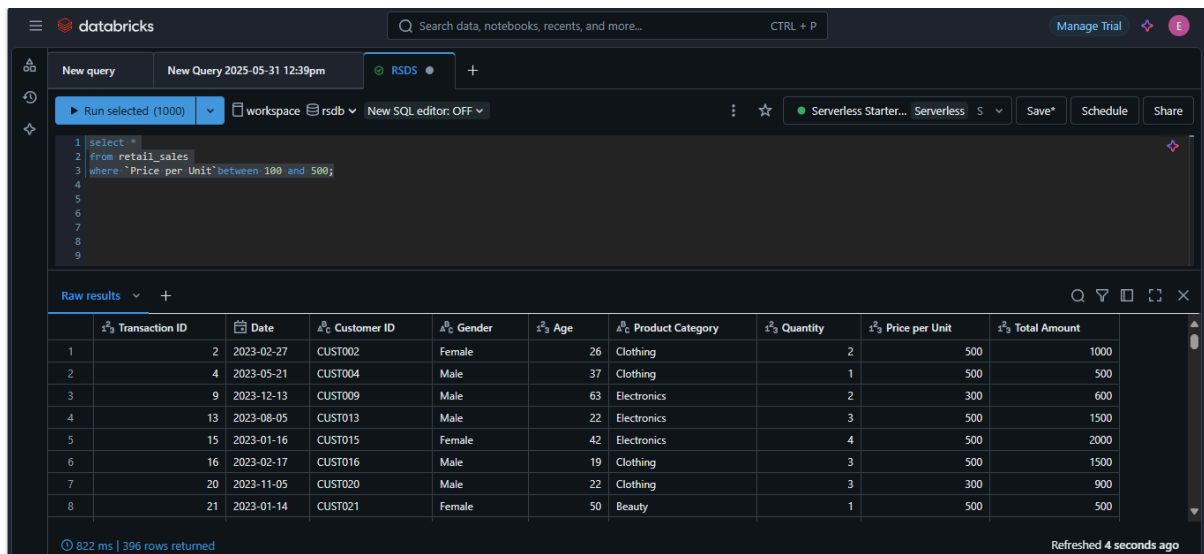


The screenshot shows the Databricks workspace with a new query editor. The query is: `select * from retail_sales where age > 40;`. The results are displayed in a table with 10 columns: Transaction ID, Date, Customer ID, Gender, Age, Product Category, Quantity, Price per Unit, and Total Amount. The table contains 8 rows of data where the age is greater than 40.

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
1	3	2023-01-13	CUST003	Male	50	Electronics	1	30	30
2	6	2023-04-25	CUST006	Female	45	Beauty	1	30	30
3	7	2023-03-13	CUST007	Male	46	Clothing	2	25	50
4	9	2023-12-13	CUST009	Male	63	Electronics	2	300	600
5	10	2023-10-07	CUST010	Female	52	Clothing	4	50	200
6	14	2023-01-17	CUST014	Male	64	Clothing	4	30	120
7	15	2023-01-16	CUST015	Female	42	Electronics	4	500	2000
8	18	2023-04-30	CUST018	Female	47	Electronics	2	25	50

Q6. Display all transactions where the Price per Unit is between 100 and 500.

Expected output: All columns



The screenshot shows the Databricks workspace with a new query editor. The query is: `select * from retail_sales where Price per Unit between 100 and 500;`. The results are displayed in a table with 10 columns: Transaction ID, Date, Customer ID, Gender, Age, Product Category, Quantity, Price per Unit, and Total Amount. The table contains 8 rows of data where the price per unit is between 100 and 500.

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
1	2	2023-02-27	CUST002	Female	26	Clothing	2	500	1000
2	4	2023-05-21	CUST004	Male	37	Clothing	1	500	500
3	9	2023-12-13	CUST009	Male	63	Electronics	2	300	600
4	13	2023-08-05	CUST013	Male	22	Electronics	3	500	1500
5	15	2023-01-16	CUST015	Female	42	Electronics	4	500	2000
6	16	2023-02-17	CUST016	Male	19	Clothing	3	500	1500
7	20	2023-11-05	CUST020	Male	22	Clothing	3	300	900
8	21	2023-01-14	CUST021	Female	50	Beauty	1	500	500

Q7. Display all transactions where the Product Category is either 'Beauty' or 'Electronics'.

Expected output: All columns

The screenshot shows the Databricks SQL interface. The query editor contains the following SQL code:

```
1 select *
2 from retail_sales
3 where "Product Category"='Beauty' or "Product Category"='Electronics';
4
5
6
7
8
9
```

The results table displays 8 rows of data. The status bar indicates 822 ms and 649 rows returned, refreshed 4 seconds ago.

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
1	1	2023-11-24	CUST001	Male	34	Beauty	3	50	150
2	3	2023-01-13	CUST003	Male	50	Electronics	1	30	30
3	5	2023-05-06	CUST005	Male	30	Beauty	2	50	100
4	6	2023-04-25	CUST006	Female	45	Beauty	1	30	30
5	8	2023-02-22	CUST008	Male	30	Electronics	4	25	100
6	9	2023-12-13	CUST009	Male	63	Electronics	2	300	600
7	12	2023-10-30	CUST012	Male	35	Beauty	3	25	75
8	13	2023-08-05	CUST013	Male	22	Electronics	3	500	1500

Q8. Display all transactions where the Product Category is not 'Clothing'.

Expected output: All columns

The screenshot shows the Databricks SQL interface. The query editor contains the following SQL code:

```
1 select *
2 from retail_sales
3 where "Product Category" not in ('Clothing');
4
5
6
7
8
9
```

The results table displays 8 rows of data. The status bar indicates 795 ms and 649 rows returned, refreshed 34 seconds ago.

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
1	1	2023-11-24	CUST001	Male	34	Beauty	3	50	150
2	3	2023-01-13	CUST003	Male	50	Electronics	1	30	30
3	5	2023-05-06	CUST005	Male	30	Beauty	2	50	100
4	6	2023-04-25	CUST006	Female	45	Beauty	1	30	30
5	8	2023-02-22	CUST008	Male	30	Electronics	4	25	100
6	9	2023-12-13	CUST009	Male	63	Electronics	2	300	600
7	12	2023-10-30	CUST012	Male	35	Beauty	3	25	75
8	13	2023-08-05	CUST013	Male	22	Electronics	3	500	1500

Q9. Display all transactions where the Quantity is greater than or equal to 3.
Expected output: All columns

The screenshot shows the Databricks SQL interface. The query editor contains the following SQL code:

```
1 select *
2 from retail_sales
3 where Quantity >= 3;
```

The results pane displays 8 rows of data. The status bar indicates 743 ms and 504 rows returned.

	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
1	1	2023-11-24	CUST001	Male	34	Beauty	3	50	150
2	8	2023-02-22	CUST008	Male	30	Electronics	4	25	100
3	10	2023-10-07	CUST010	Female	52	Clothing	4	50	200
4	12	2023-10-30	CUST012	Male	35	Beauty	3	25	75
5	13	2023-08-05	CUST013	Male	22	Electronics	3	500	1500
6	14	2023-01-17	CUST014	Male	64	Clothing	4	30	120
7	15	2023-01-16	CUST015	Female	42	Electronics	4	500	2000
8	16	2023-02-17	CUST016	Male	19	Clothing	3	500	1500

4. Aggregate Functions
Q10. Count the total number of transactions.
Expected output: Total_Transactions

The screenshot shows the Databricks SQL interface. The query editor contains the following SQL code:

```
1 select count('Transaction ID') as Total_Transactions
2 from retail_sales;
```

The results pane displays 1 row of data. The status bar indicates 851 ms and 1 row returned.

Total_Transactions
1000

Q11. Find the average Age of customers.

Expected output: Average_Age

The screenshot shows the Databricks SQL interface. The query editor contains the following SQL code:

```
1 select avg(age) as Average_Age
2 from retail_sales;
```

The results table shows the average age of customers:

	1.2 Average_Age
1	41.392

At the bottom, it indicates "837 ms | 1 row returned" and "Refreshed 4 seconds ago".

Q12. Find the total quantity of products sold.

Expected output: Total_Quantity

The screenshot shows the Databricks SQL interface. The query editor contains the following SQL code:

```
1 select sum(Quantity) as Total_Quantity
2 from retail_sales;
```

The results table shows the total quantity of products sold:

	1.2 Total_Quantity
1	2514

At the bottom, it indicates "650 ms | 1 row returned" and "Refreshed 4 seconds ago".

Q13. Find the maximum Total Amount spent in a single transaction.
Expected output: Max_Total_Amount

The screenshot shows the Databricks SQL interface. The query editor contains the following SQL code:

```
1 select max("Total Amount") as Max_Total_Amount
2 from retail_sales;
```

The results pane shows the following table:

	Max_Total_Amount
1	2000

At the bottom, it indicates "810 ms | 1 row returned" and "Refreshed 5 seconds ago".

Q14. Find the minimum Price per Unit in the dataset.
Expected output: Min_Price_per_Unit

The screenshot shows the Databricks SQL interface. The query editor contains the following SQL code:

```
1 select min("Price per Unit") as Min_Price_per_Unit
2 from retail_sales;
```

The results pane shows the following table:

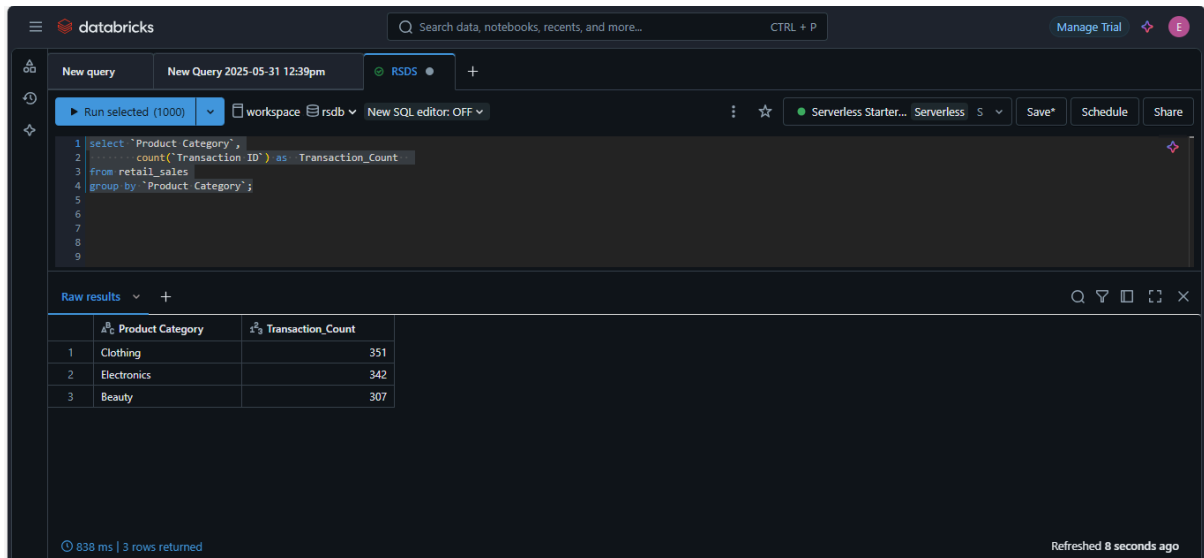
	Min_Price_per_Unit
1	25

At the bottom, it indicates "824 ms | 1 row returned" and "Refreshed 4 seconds ago".

5. GROUP BY Statement

Q15. Find the number of transactions per Product Category.

Expected output: Product Category, Transaction_Count



The screenshot shows the Databricks SQL interface. The query editor contains the following SQL code:

```
1 select "Product Category",
2        count("Transaction ID") as Transaction_Count
3 from retail_sales
4 group by "Product Category";
```

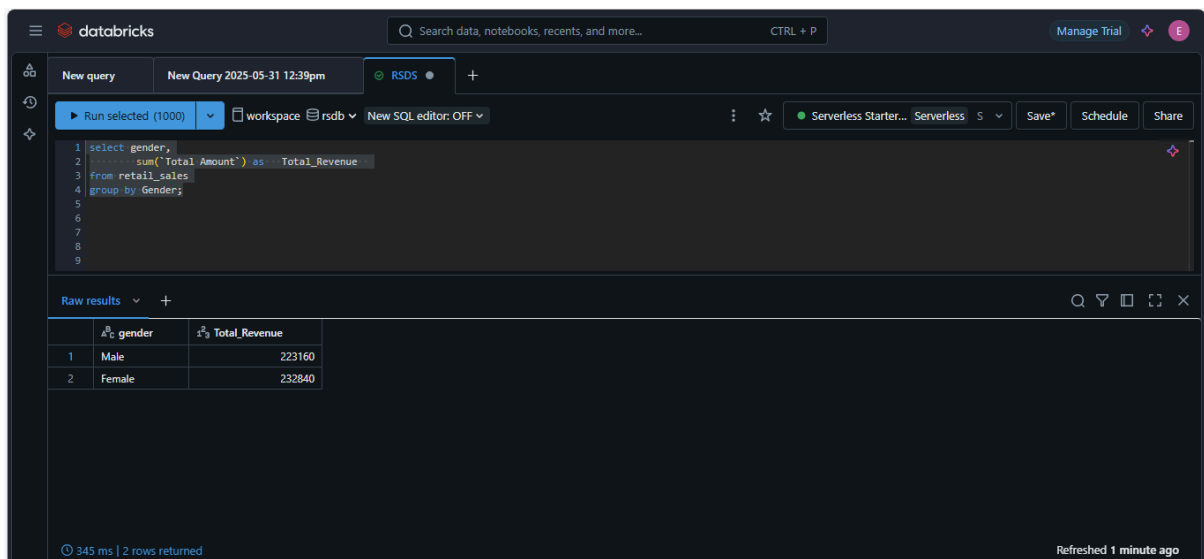
The results table shows 3 rows:

	Product Category	Transaction_Count
1	Clothing	351
2	Electronics	342
3	Beauty	307

At the bottom, it indicates "838 ms | 3 rows returned" and "Refreshed 8 seconds ago".

Q16. Find the total revenue (Total Amount) per gender.

Expected output: Gender, Total_Revenue



The screenshot shows the Databricks SQL interface. The query editor contains the following SQL code:

```
1 select gender,
2        sum("Total Amount") as Total_Revenue
3 from retail_sales
4 group by Gender;
```

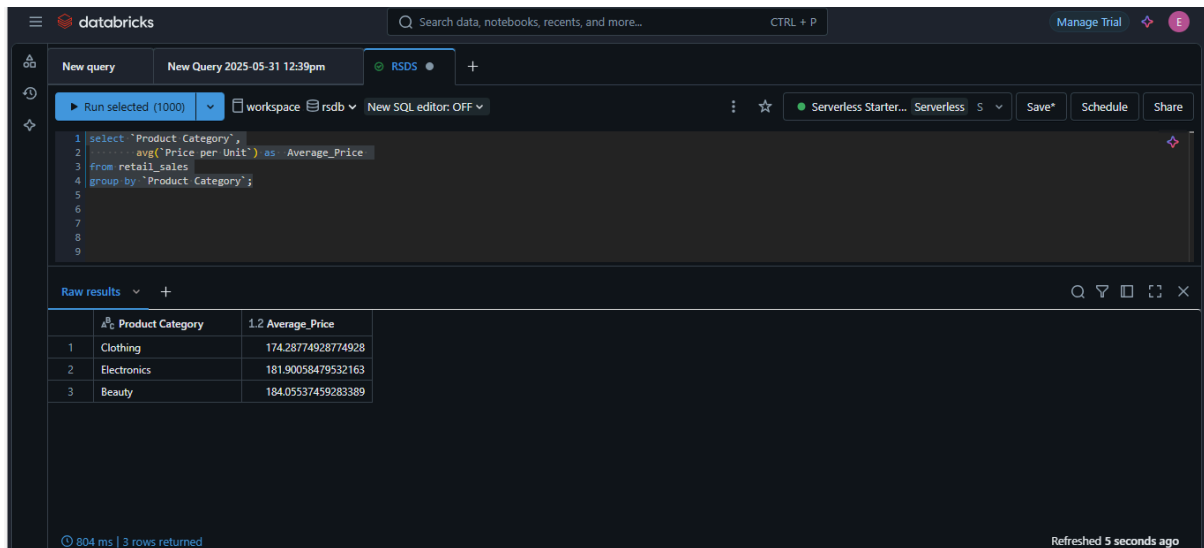
The results table shows 2 rows:

	gender	Total_Revenue
1	Male	223160
2	Female	232840

At the bottom, it indicates "345 ms | 2 rows returned" and "Refreshed 1 minute ago".

Q17. Find the average Price per Unit per product category.

Expected output: Product Category, Average_Price



The screenshot shows the Databricks SQL interface. The query editor contains the following SQL code:

```
1 select "Product Category",  
2       avg("Price per Unit") as Average_Price  
3 from retail_sales  
4 group by "Product Category";
```

The results are displayed in a table with 3 rows:

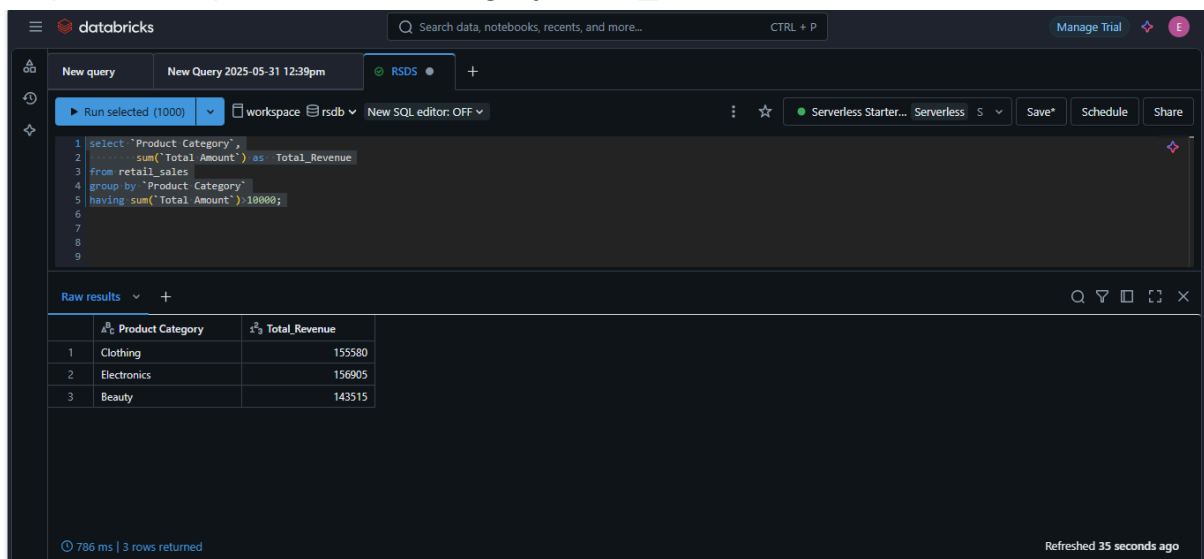
	Product Category	Average_Price
1	Clothing	174.28774928774928
2	Electronics	181.90058479532163
3	Beauty	184.05537459283389

At the bottom, it indicates "804 ms | 3 rows returned" and "Refreshed 5 seconds ago".

6. HAVING Clause

Q18. Find the total revenue per product category where total revenue is greater than 10,000.

Expected output: Product Category, Total_Revenue



The screenshot shows the Databricks SQL interface. The query editor contains the following SQL code:

```
1 select "Product Category",  
2       sum("Total Amount") as Total_Revenue  
3 from retail_sales  
4 group by "Product Category"  
5 having sum("Total Amount") > 10000;
```

The results are displayed in a table with 3 rows:

	Product Category	Total_Revenue
1	Clothing	155580
2	Electronics	156905
3	Beauty	143515

At the bottom, it indicates "786 ms | 3 rows returned" and "Refreshed 35 seconds ago".

Q19. Find the average quantity per product category where the average is more than 2.

Expected output: Product Category, Average_Quantity

The screenshot shows the Databricks SQL interface. The query editor contains the following SQL code:

```
1 select 'Product Category',
2       avg(Quantity) as Average_Quantity
3 from retail_sales
4 group by 'Product Category'
5 having avg(Quantity) > 2;
```

The results table shows the following data:

	Product Category	Average_Quantity
1	Clothing	2.547008547008547
2	Electronics	2.482456140350877
3	Beauty	2.511400651465798

75 ms | 3 rows returned

7. CASE Statement

Q20. Display a column called Spending_Level that shows 'High' if Total Amount > 1000, otherwise 'Low'.

Expected output: Transaction ID, Total Amount, Spending_Level

The screenshot shows the Databricks SQL interface. The query editor contains the following SQL code:

```
1 select 'Transaction ID',
2       'Total Amount',
3       case
4         when 'Total Amount' > 1000 then 'High'
5         else 'Low'
6       end as Spending_Level
7 from retail_sales;
```

The results table shows the following data:

	Transaction ID	Total Amount	Spending_Level
1	1	150	Low
2	2	1000	Low
3	3	30	Low
4	4	500	Low
5	5	100	Low
6	6	30	Low
7	7	50	Low
8	8	100	Low

842 ms | 1,000 rows returned

Q21. Display a new column called Age_Group that labels customers as:

- 'Youth' if Age < 30
- 'Adult' if Age is between 30 and 59
- 'Senior' if Age >= 60

Expected output: Customer ID, Age, Age_Group

The screenshot shows the Databricks SQL interface. At the top, there's a search bar and a 'New query' button. Below that, a 'Run selected (1000)' button is visible. The SQL editor contains the following query:

```
1 select 'Customer ID',
2       Age,
3       case
4         when Age < 30 then 'Youth'
5         when Age between 30 and 59 then 'Adult'
6         when Age >= 60 then 'Senior'
7       end as Age_Group
8 from retail_sales;
9
```

Below the query editor, the 'Raw results' section displays a table with 8 rows and 3 columns: Customer ID, Age, and Age_Group. The data is as follows:

	Customer ID	Age	Age_Group
1	CUST001	34	Adult
2	CUST002	26	Youth
3	CUST003	50	Adult
4	CUST004	37	Adult
5	CUST005	30	Adult
6	CUST006	45	Adult
7	CUST007	46	Adult
8	CUST008	30	Adult

At the bottom left, it says '822 ms | 1,000 rows returned'. At the bottom right, it says 'Refreshed 4 seconds ago'.