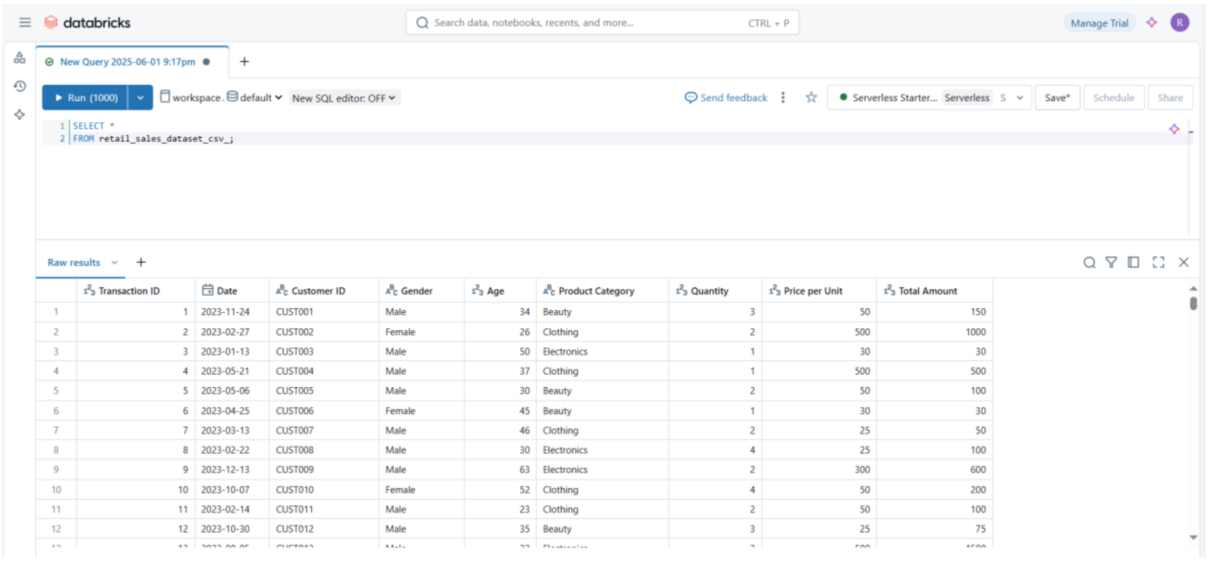
<u>RIBA MAUBANE</u>
<u>DATABRICKS PRACTICAL 1</u>

QUESTION 1

SELECT Statement

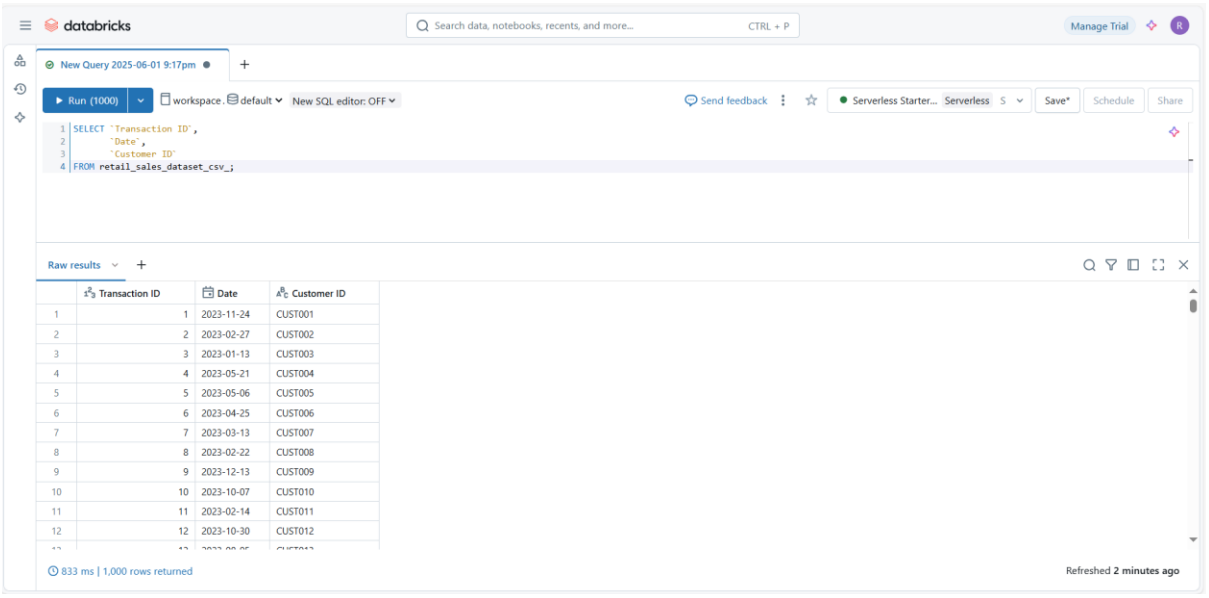
1. <u>Display all columns for all transactions.</u>
Expected output: All columns



```sql
SELECT *
FROM retail_sales_dataset_csv_;
```

	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
9	9	2023-12-13	CUST009	Male	63	Electronics	2	300	600
10	10	2023-10-07	CUST010	Female	52	Clothing	4	50	200
11	11	2023-02-14	CUST011	Male	23	Clothing	2	50	100
12	12	2023-10-30	CUST012	Male	35	Beauty	3	25	75

2. <u>Display only the Transaction ID, Date, and Customer ID for all records.</u>
Expected output: Transaction ID, Date, Customer ID



```sql
SELECT `Transaction ID`,
       `Date`,
       `Customer ID`
FROM retail_sales_dataset_csv_;
```

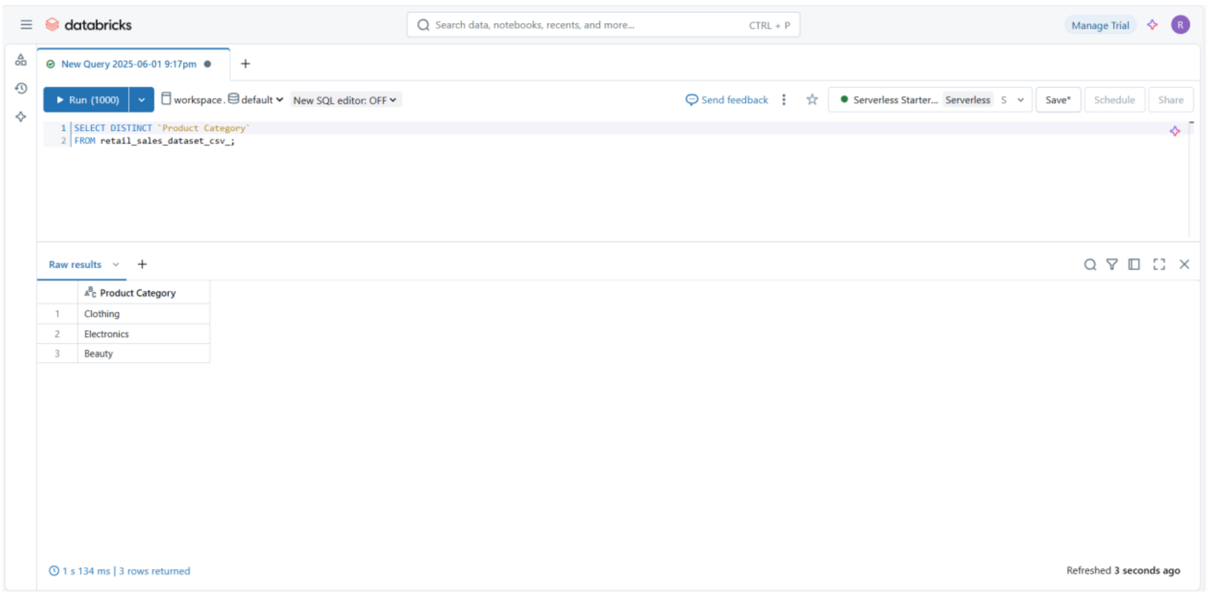
	Transaction ID	Date	Customer ID
1	1	2023-11-24	CUST001
2	2	2023-02-27	CUST002
3	3	2023-01-13	CUST003
4	4	2023-05-21	CUST004
5	5	2023-05-06	CUST005
6	6	2023-04-25	CUST006
7	7	2023-03-13	CUST007
8	8	2023-02-22	CUST008
9	9	2023-12-13	CUST009
10	10	2023-10-07	CUST010
11	11	2023-02-14	CUST011
12	12	2023-10-30	CUST012

833 ms | 1,000 rows returned

Refreshed 2 minutes ago

SELECT DISTINCT Statement

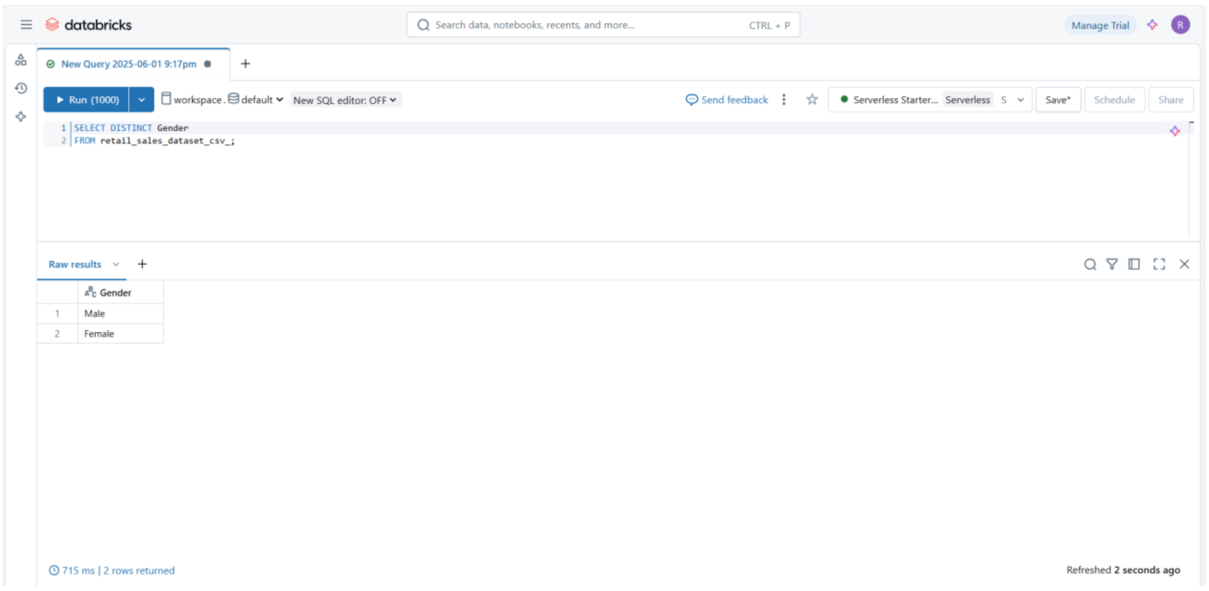
3. Display all the distinct product categories in the dataset.
Expected output: Product Category


```
SELECT DISTINCT `Product Category`
FROM retail_sales_dataset_csv_;
```

	Product Category
1	Clothing
2	Electronics
3	Beauty

```
1 s 134 ms | 3 rows returned
```

4. Display all the distinct gender values in the dataset.
Expected output: Gender

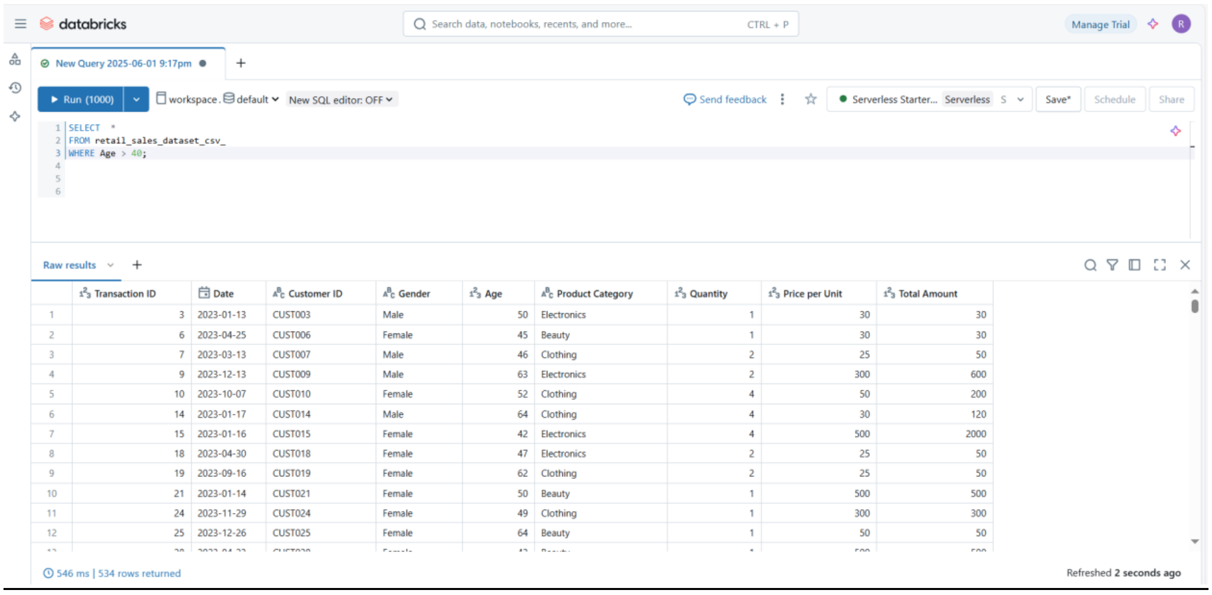

```
SELECT DISTINCT Gender
FROM retail_sales_dataset_csv_;
```

	Gender
1	Male
2	Female

```
715 ms | 2 rows returned
```

WHERE Clause

5. Display all transactions where the Age is greater than 40.
Expected output: All columns



New Query 2025-06-01 9:17pm

```sql
SELECT  *
FROM retail_sales_dataset_csv_
WHERE Age > 40;
```

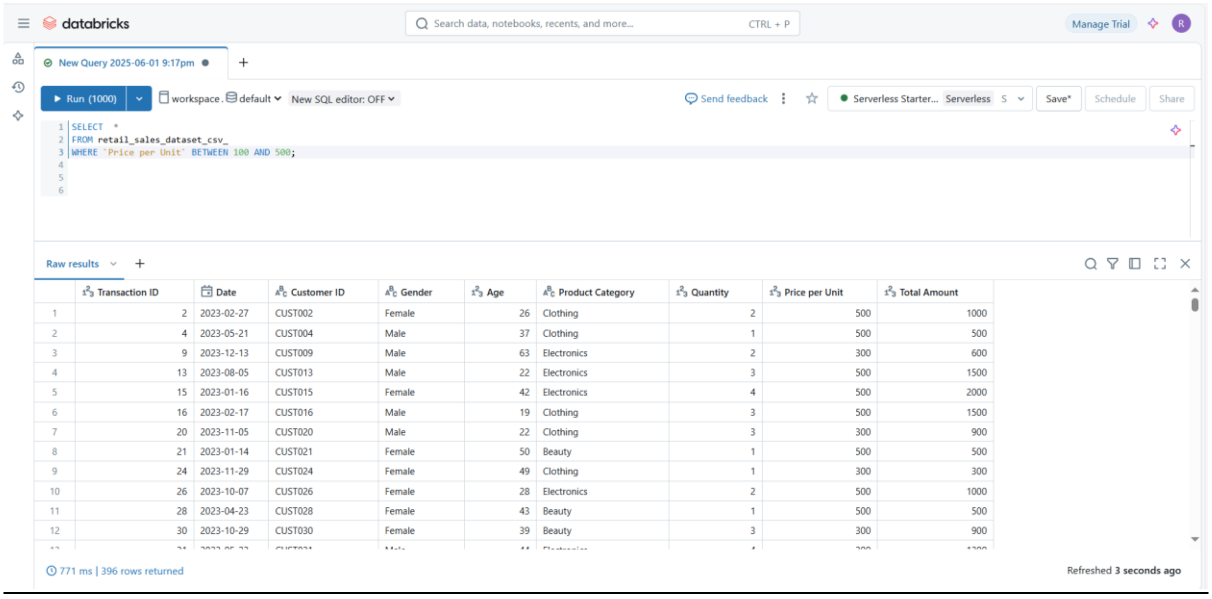
Raw results

	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
9	19	2023-09-16	CUST019	Female	62	Clothing	2	25	50
10	21	2023-01-14	CUST021	Female	50	Beauty	1	500	500
11	24	2023-11-29	CUST024	Female	49	Clothing	1	300	300
12	25	2023-12-26	CUST025	Female	64	Beauty	1	50	50

546 ms | 534 rows returned

Refreshed 2 seconds ago

6. Display all transactions where the Price per Unit is between 100 and 500.
Expected output: All columns



New Query 2025-06-01 9:17pm

```sql
SELECT  *
FROM retail_sales_dataset_csv_
WHERE `Price per Unit` BETWEEN 100 AND 500;
```

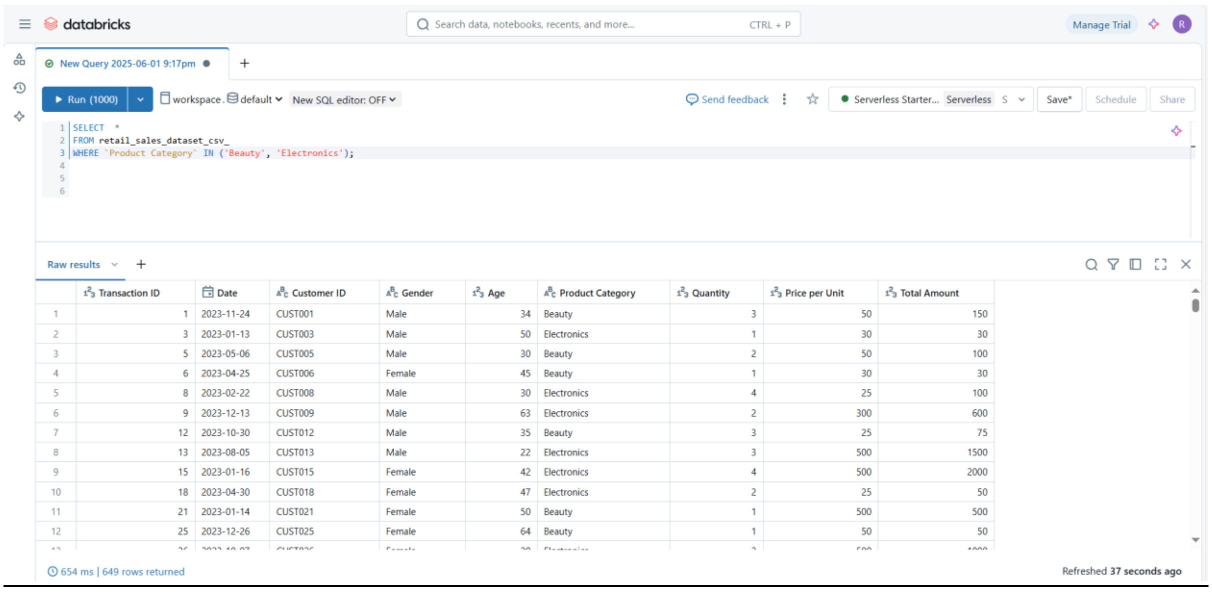
Raw results

	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
9	24	2023-11-29	CUST024	Female	49	Clothing	1	300	300
10	26	2023-10-07	CUST026	Female	28	Electronics	2	500	1000
11	28	2023-04-23	CUST028	Female	43	Beauty	1	500	500
12	30	2023-10-29	CUST030	Female	39	Beauty	3	300	900

771 ms | 396 rows returned

Refreshed 3 seconds ago

7. <u>Display all transactions where the Product Category is either 'Beauty' or 'Electronics'.</u>
Expected output: All columns



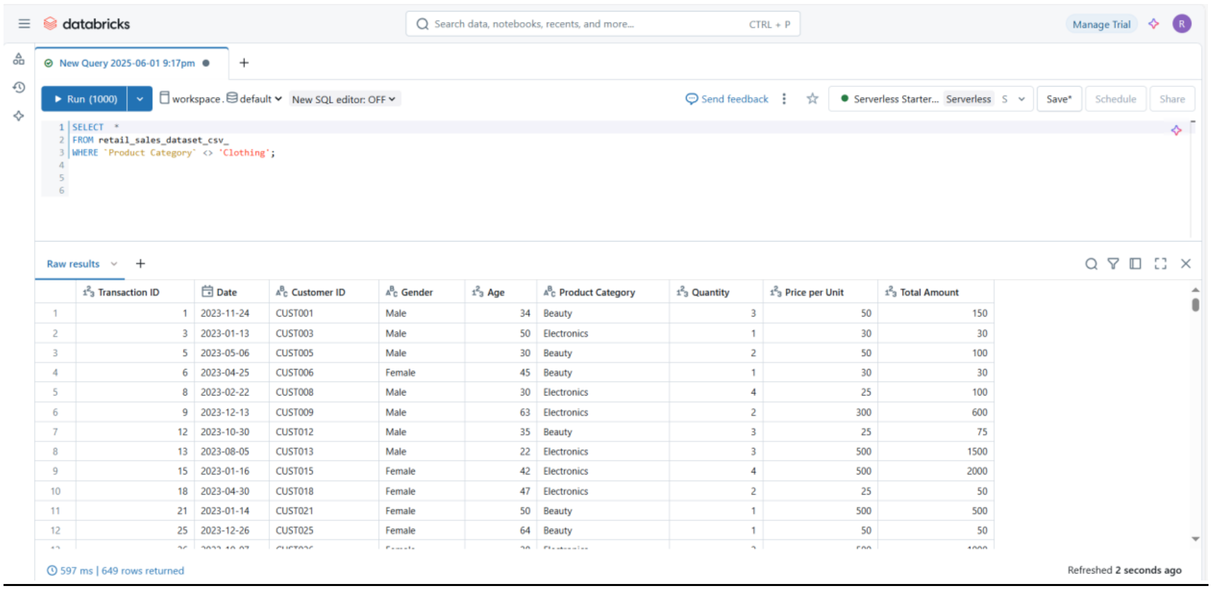
```sql
SELECT  *
FROM retail_sales_dataset_csv_
WHERE `Product Category` IN ('Beauty', 'Electronics');
```

	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
9	15	2023-01-16	CUST015	Female	42	Electronics	4	500	2000
10	18	2023-04-30	CUST018	Female	47	Electronics	2	25	50
11	21	2023-01-14	CUST021	Female	50	Beauty	1	500	500
12	25	2023-12-26	CUST025	Female	64	Beauty	1	50	50

654 ms | 649 rows returned

Refreshed **37 seconds** ago

8. <u>Display all transactions where the Product Category is not 'Clothing'.</u>
Expected output: All columns



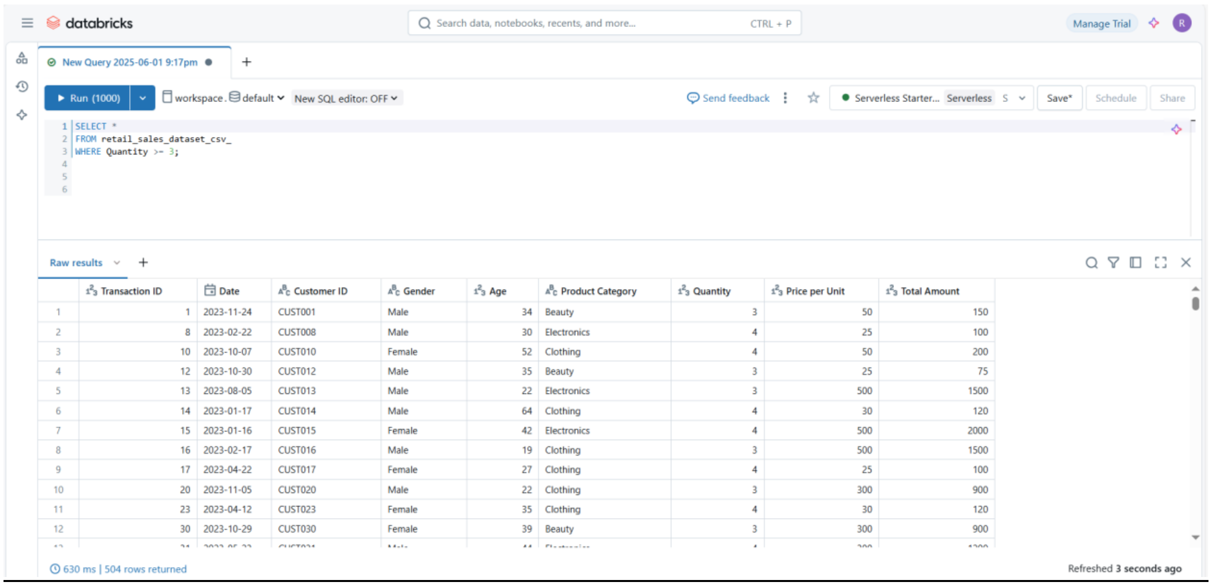
```sql
SELECT  *
FROM retail_sales_dataset_csv_
WHERE `Product Category` <> 'Clothing';
```

	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
9	15	2023-01-16	CUST015	Female	42	Electronics	4	500	2000
10	18	2023-04-30	CUST018	Female	47	Electronics	2	25	50
11	21	2023-01-14	CUST021	Female	50	Beauty	1	500	500
12	25	2023-12-26	CUST025	Female	64	Beauty	1	50	50

597 ms | 649 rows returned

Refreshed **2 seconds** ago

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


databricks
Search data, notebooks, recents, and more... CTRL + P
Manage Trial
New Query 2025-06-01 9:17pm
Run (1000)
workspace . default
New SQL editor: OFF
Send feedback
Serverless Starter... Serverless S
Save*
Schedule
Share
Raw results

```sql
SELECT *
FROM retail_sales_dataset_csv_
WHERE Quantity >= 3;
```

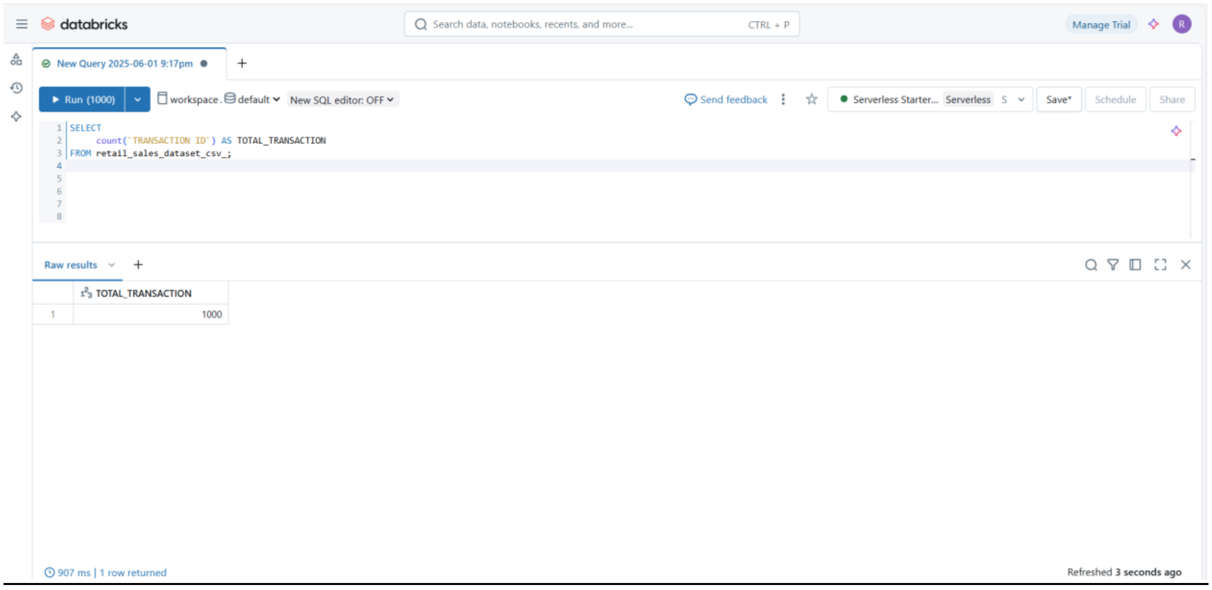
	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
1	1	2023-11-24	CUST001	Male	34	Beauty	3	50	150
2	8	2023-02-22	CUST008	Male	30	Electronics	4	25	100
3	10	2023-10-07	CUST010	Female	52	Clothing	4	50	200
4	12	2023-10-30	CUST012	Male	35	Beauty	3	25	75
5	13	2023-08-05	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
9	17	2023-04-22	CUST017	Female	27	Clothing	4	25	100
10	20	2023-11-05	CUST020	Male	22	Clothing	3	300	900
11	23	2023-04-12	CUST023	Female	35	Clothing	4	30	120
12	30	2023-10-29	CUST030	Female	39	Beauty	3	300	900

630 ms | 504 rows returned

Refreshed 3 seconds ago

Aggregate Functions

10. Count the total number of transactions.
Expected output: Total_Transactions


databricks
Search data, notebooks, recents, and more... CTRL + P
Manage Trial
New Query 2025-06-01 9:17pm
Run (1000)
workspace . default
New SQL editor: OFF
Send feedback
Serverless Starter... Serverless S
Save*
Schedule
Share
Raw results

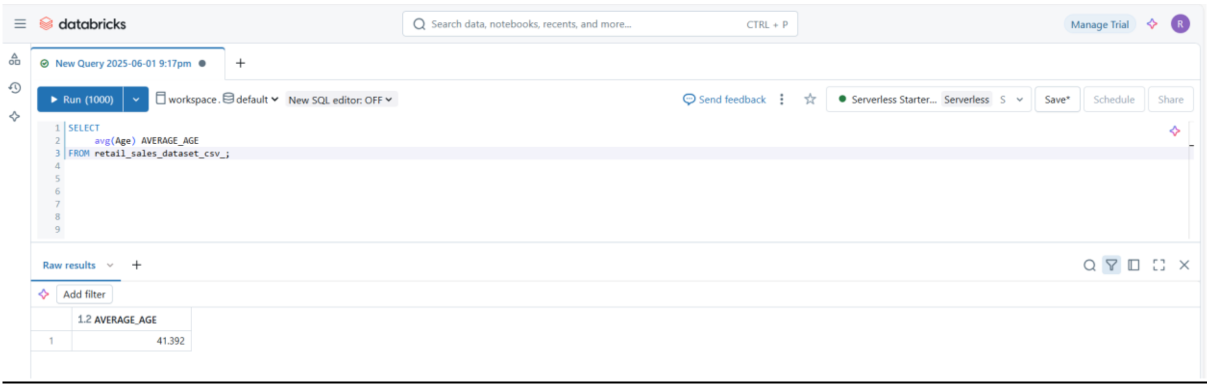
```sql
SELECT
    count('TRANSACTION ID') AS TOTAL_TRANSACTION
FROM retail_sales_dataset_csv_;
```

	TOTAL_TRANSACTION
1	1000

907 ms | 1 row returned

Refreshed 3 seconds ago

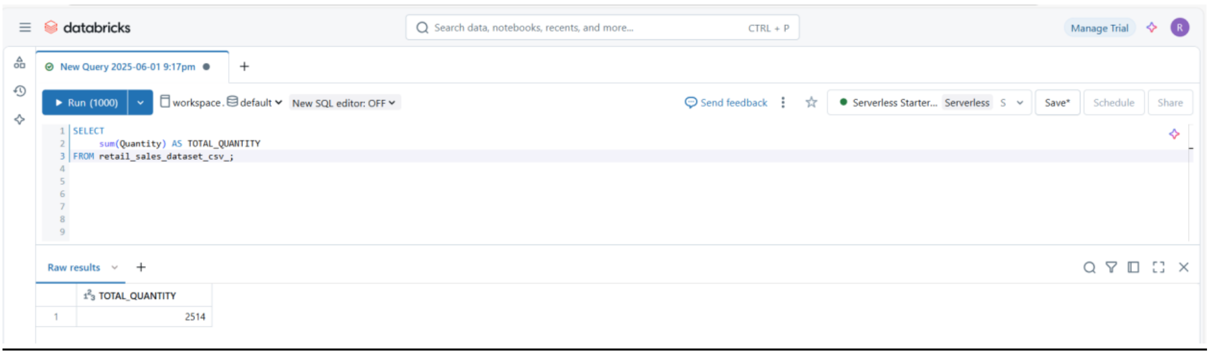
11. Find the average Age of customers.
Expected output: Average_Age



```sql
SELECT
      avg(Age) AVERAGE_AGE
FROM retail_sales_dataset_csv_;
```

```
   1.2 AVERAGE_AGE
1           41.392
```

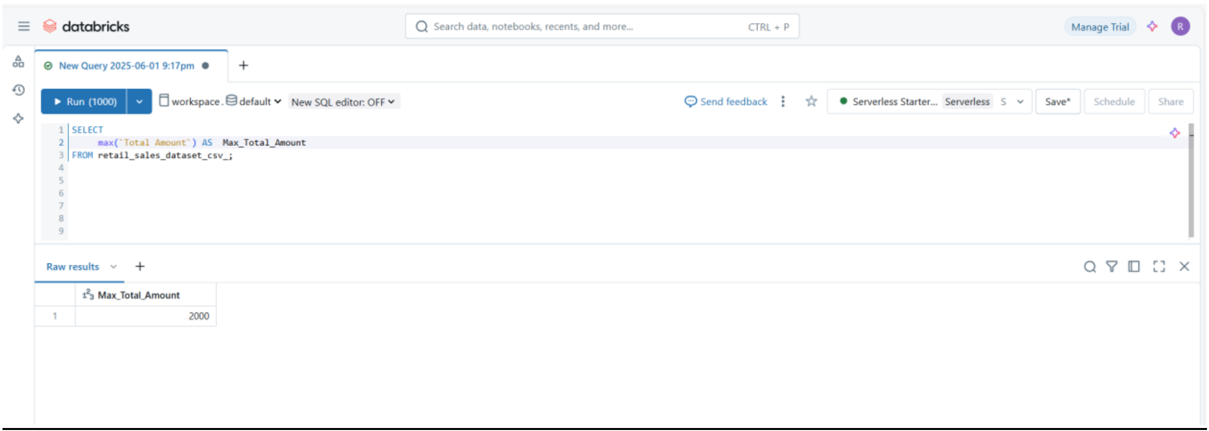
12. Find the total quantity of products sold.
Expected output: Total_Quantity



```sql
SELECT
      sum(Quantity) AS TOTAL_QUANTITY
FROM retail_sales_dataset_csv_;
```

```
   TOTAL_QUANTITY
1            2514
```

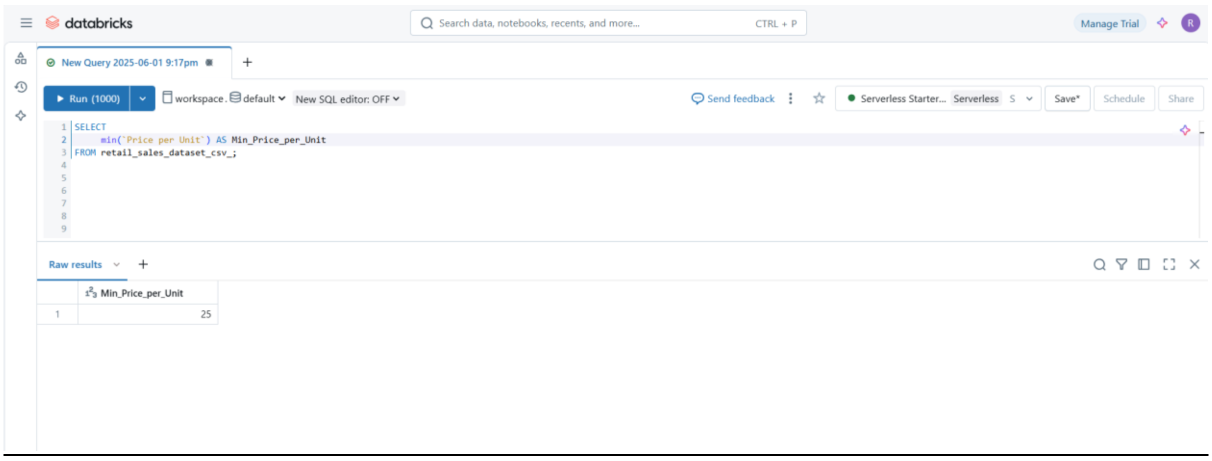
13. Find the maximum Total Amount spent in a single transaction.
Expected output: Max_Total_Amount



```sql
SELECT
      max(`Total Amount`) AS  Max_Total_Amount
FROM retail_sales_dataset_csv_;
```

```
   Max_Total_Amount
1              2000
```

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

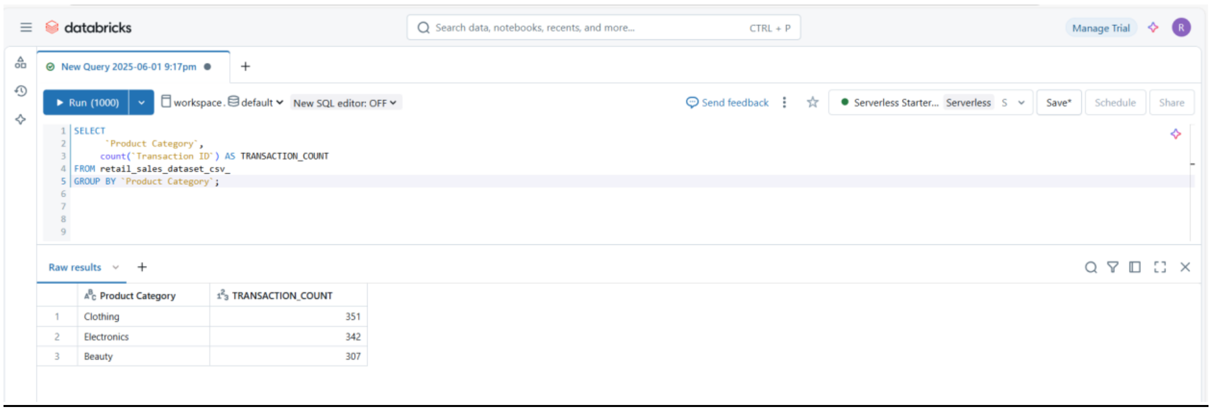


```sql
SELECT
    min(`Price per Unit`) AS Min_Price_per_Unit
FROM retail_sales_dataset_csv_;
```

	Min_Price_per_Unit
1	25

GROUP BY Statement

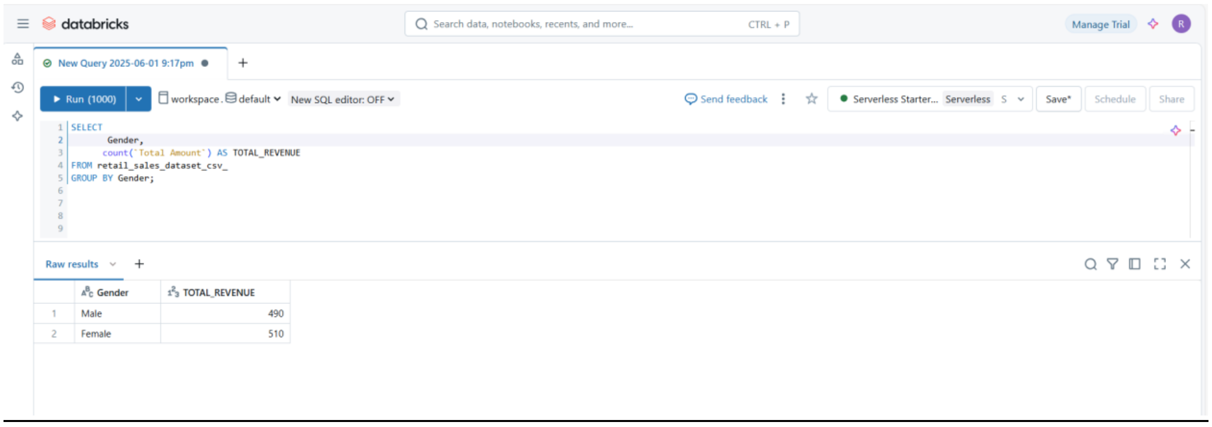
15. Find the number of transactions per Product Category.
Expected output: Product Category, Transaction_Count



```sql
SELECT
    `Product Category`,
    count(`Transaction ID`) AS TRANSACTION_COUNT
FROM retail_sales_dataset_csv_
GROUP BY `Product Category`;
```

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

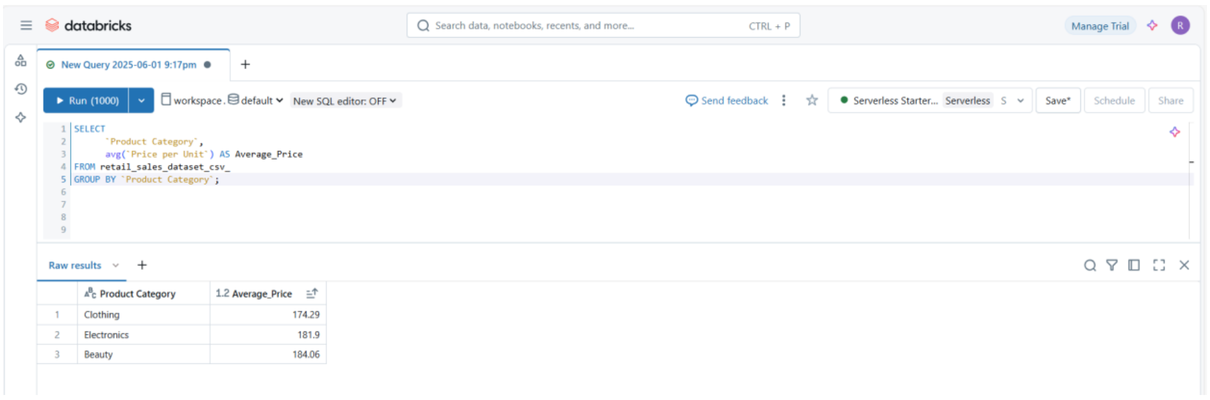
16. Find the total revenue (Total Amount) per gender.
Expected output: Gender, Total_Revenue



```sql
SELECT
    Gender,
    count(`Total Amount`) AS TOTAL_REVENUE
FROM retail_sales_dataset_csv_
GROUP BY Gender;
```

	Gender	TOTAL_REVENUE
1	Male	490
2	Female	510

17. Find the average Price per Unit per product category.
Expected output: Product Category, Average_Price

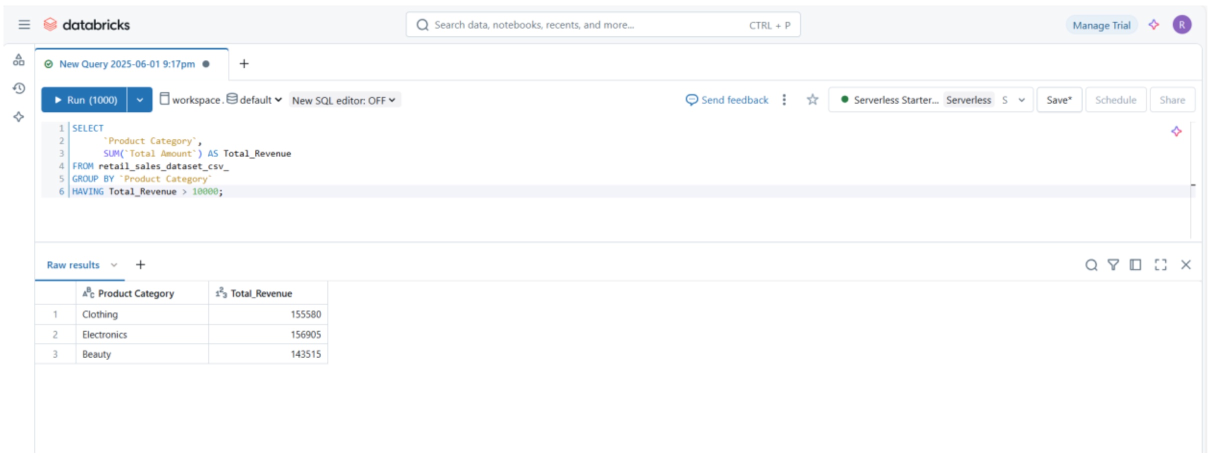


```sql
SELECT
    `Product Category`,
    avg(`Price per Unit`) AS Average_Price
FROM retail_sales_dataset_csv_
GROUP BY `Product Category`;
```

	Product Category	Average_Price
1	Clothing	174.29
2	Electronics	181.9
3	Beauty	184.06

HAVING Clause

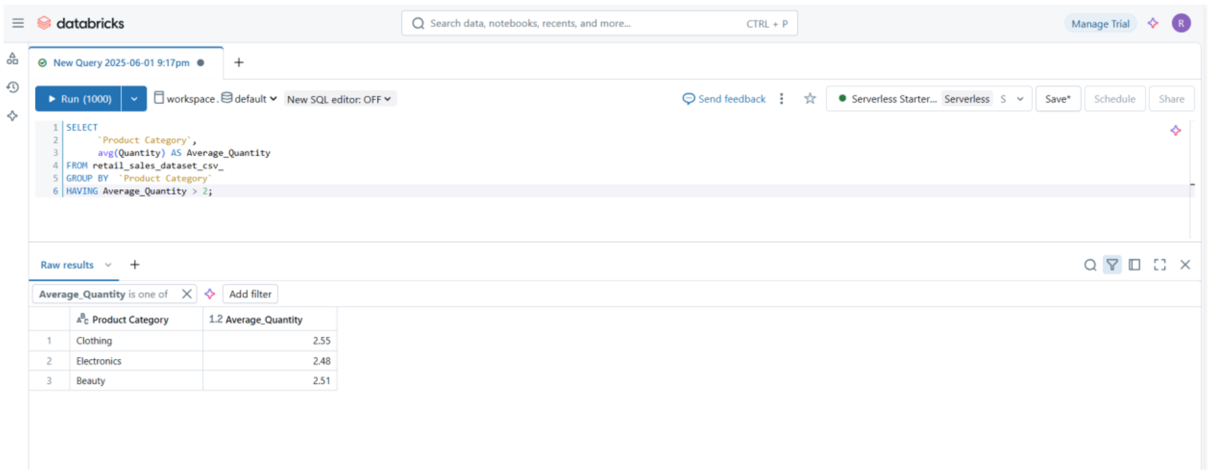
18. Find the total revenue per product category where total revenue is greater than 10,000. Expected output: Product Category, Total_Revenue



```sql
SELECT
    `Product Category`,
    SUM(`Total Amount`) AS Total_Revenue
FROM retail_sales_dataset_csv_
GROUP BY `Product Category`
HAVING Total_Revenue > 10000;
```

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

19. Find the average quantity per product category where the average is more than 2.
Expected output: Product Category, Average_Quantity

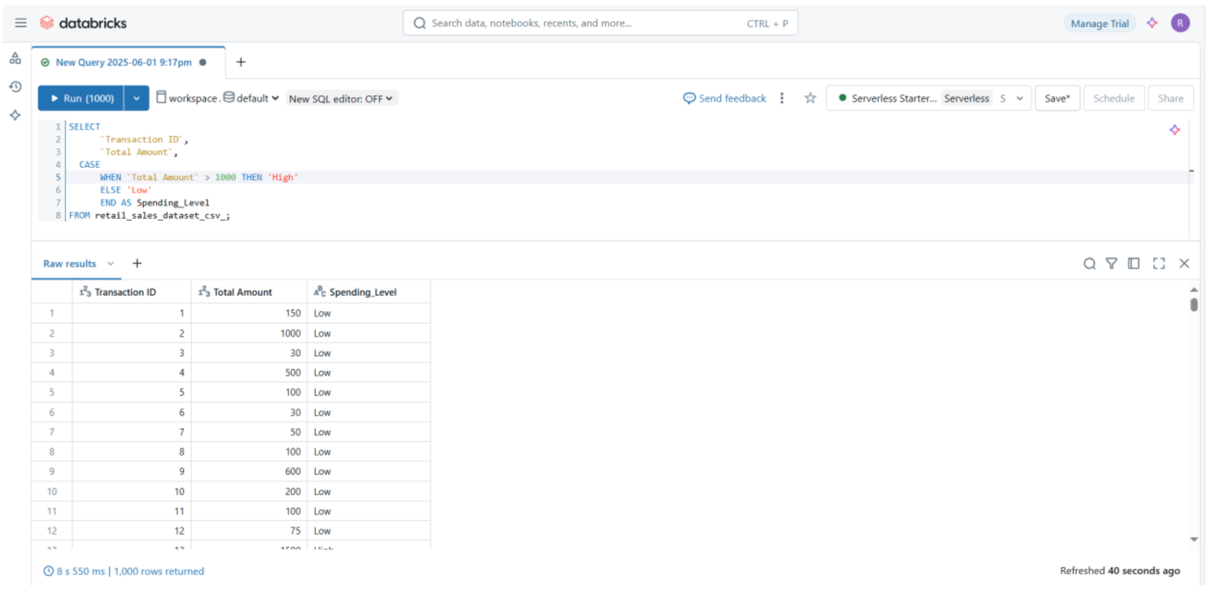


```sql
SELECT
    `Product Category`,
    avg(Quantity) AS Average_Quantity
FROM retail_sales_dataset_csv_
GROUP BY  `Product Category`
HAVING Average_Quantity > 2;
```

	Product Category	Average_Quantity
1	Clothing	2.55
2	Electronics	2.48
3	Beauty	2.51

CASE Statement

20. Display a column called Spending_Level that shows 'High' if Total Amount > 1000, otherwise 'Low'.
Expected output: Transaction ID, Total Amount, Spending_Level

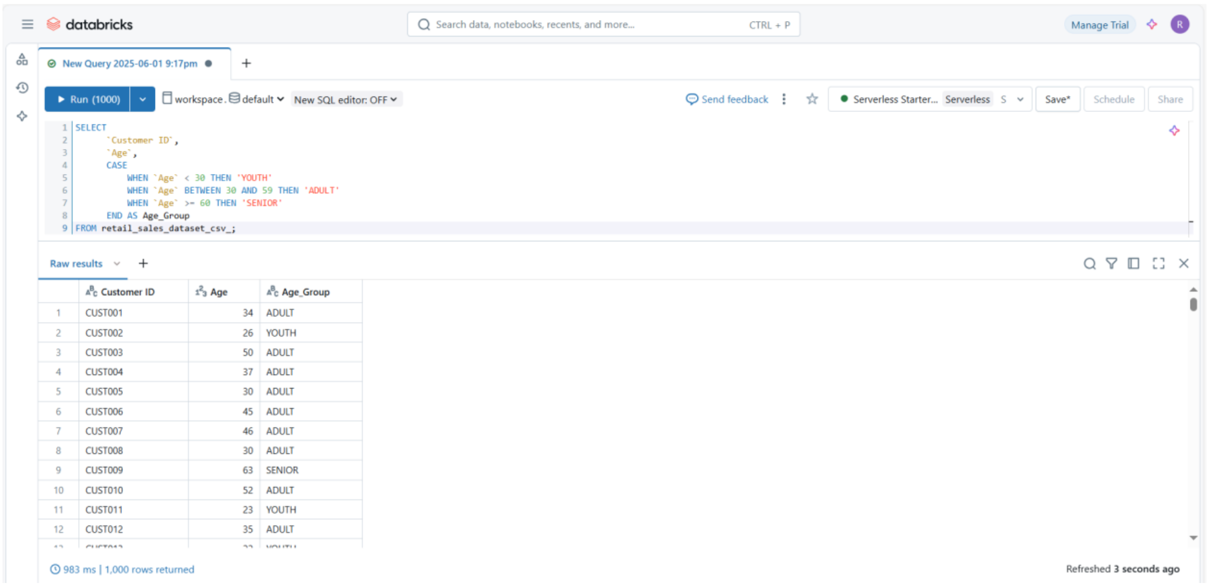


```sql
SELECT
      `Transaction ID`,
      `Total Amount`,
  CASE
      WHEN `Total Amount` > 1000 THEN 'High'
      ELSE 'Low'
      END AS Spending_Level
FROM retail_sales_dataset_csv_;
```

	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
9	9	600	Low
10	10	200	Low
11	11	100	Low
12	12	75	Low

8 s 550 ms | 1,000 rows returned

21. 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



```sql
SELECT
      `Customer ID`,
      `Age`,
      CASE
          WHEN `Age` < 30 THEN 'YOUTH'
          WHEN `Age` BETWEEN 30 AND 59 THEN 'ADULT'
          WHEN `Age` >= 60 THEN 'SENIOR'
      END AS Age_Group
FROM retail_sales_dataset_csv_;
```

	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
9	CUST009	63	SENIOR
10	CUST010	52	ADULT
11	CUST011	23	YOUTH
12	CUST012	35	ADULT

983 ms | 1,000 rows returned