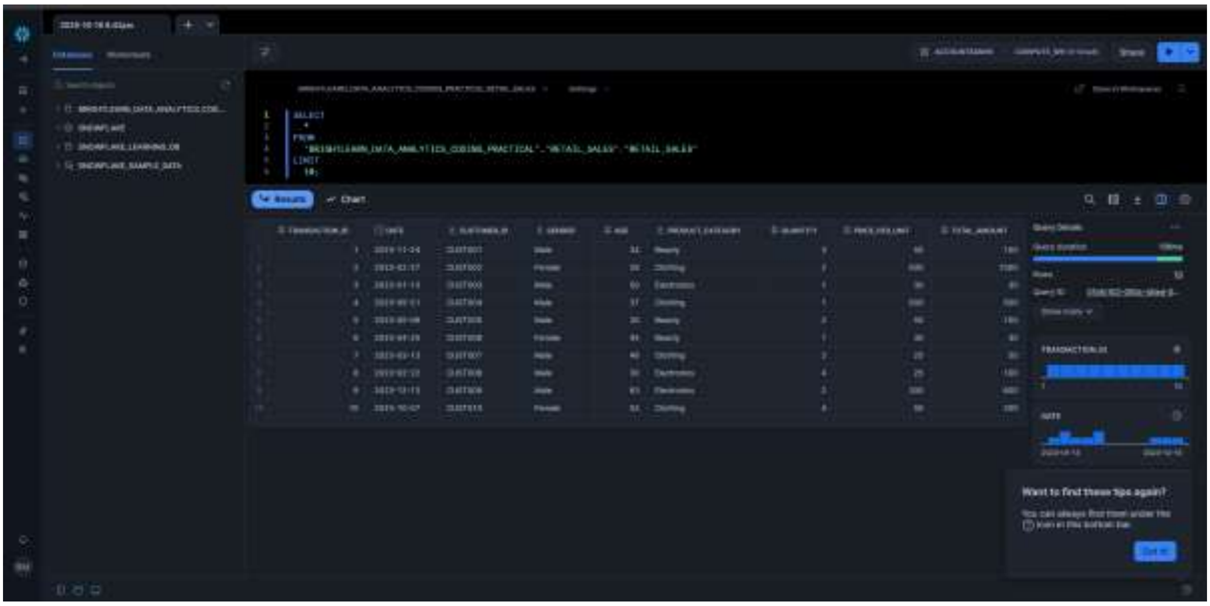
BrightLearn Data Analytics Coding Practical

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

1. SELECT Statement

Q1. Display all columns for all transactions. Expected output: All columns



```
SELECT
  *
FROM
  "BRIGHTLEARN_DATA_ANALYTICS_CODING_PRACTICAL"."RETAIL_SALES"."RETAIL_SALES"
LIMIT
  10;
```

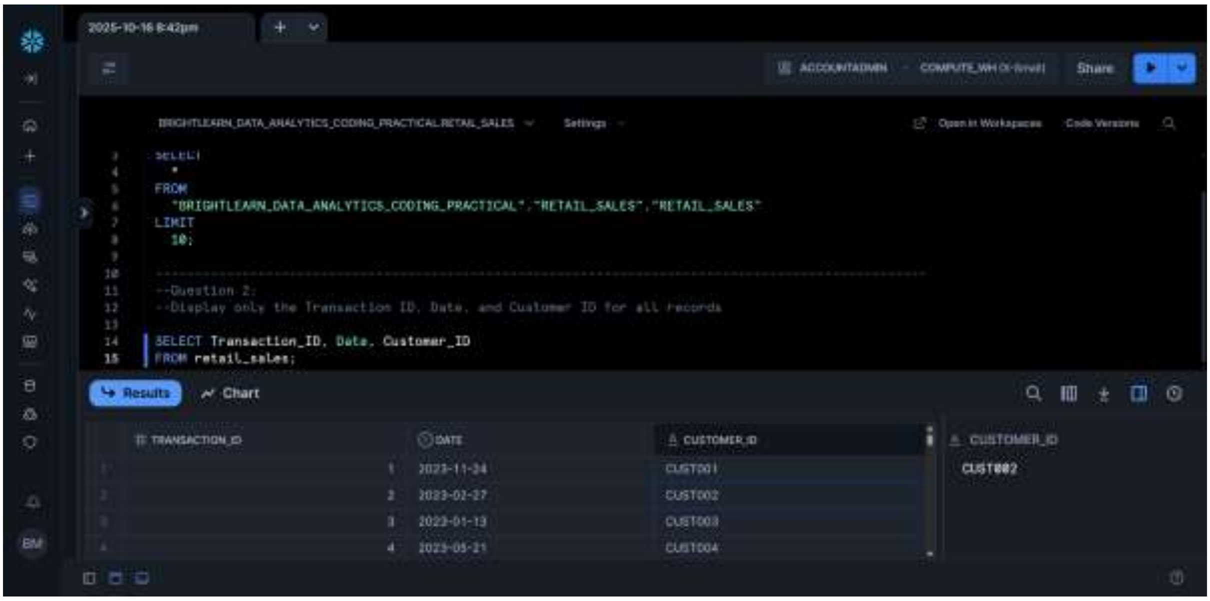
	TRANSACTION_ID	DATE	CUSTOMER_ID	GENDER	AGE	PRODUCT_CATEGORY	QUANTITY	PRICE_PER_UNIT	TOTAL_AMOUNT
1	1	2023-11-24	CUST001	Male	34	Beauty	3	50	150
2	2	2023-02-27	CUST002	Female	26	Clothing	2	500	1000
3	3	2023-01-13	CUST003	Male	50	Electronics	1	30	30
4	4	2023-05-21	CUST004	Male	37	Clothing	1	500	500
5	5	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

Want to find these tips again?

You can always find them under the ⓘ icon in the bottom bar.

Got it

Q2. Display only the Transaction ID, Date, and Customer ID for all records. *Expected output:* Transaction ID, Date, Customer ID



```
SELECT
  *
FROM
  "BRIGHTLEARN_DATA_ANALYTICS_CODING_PRACTICAL"."RETAIL_SALES"."RETAIL_SALES"
LIMIT
  10;

------------------------------------------------------------------------------------------
--Question 2:
--Display only the Transaction ID, Date, and Customer ID for all records

SELECT Transaction_ID, Date, Customer_ID
FROM retail_sales;
```

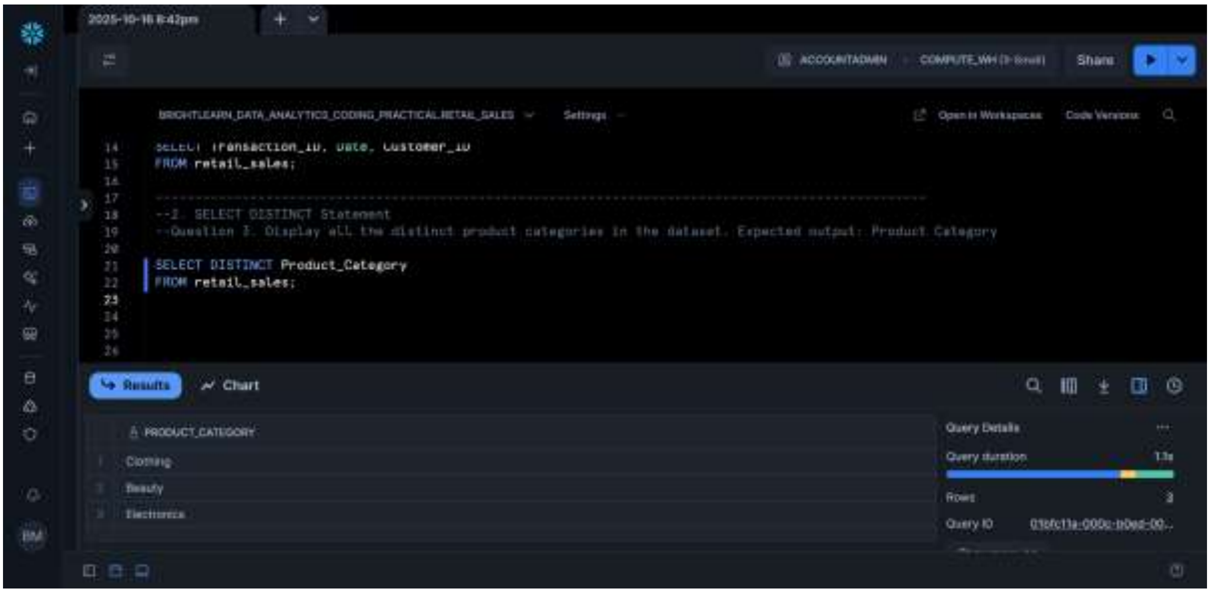
	TRANSACTION_ID	DATE	CUSTOMER_ID
1	1	2023-11-24	CUST001
2	2	2023-02-27	CUST002
3	3	2023-01-13	CUST003
4	4	2023-05-21	CUST004

CUSTOMER_ID

CUST002

2. SELECT DISTINCT Statement

Q3. Display all the distinct product categories in the dataset. *Expected output:* Product Category



```
SELECT Transaction_ID, Date, Customer_ID
FROM retail_sales;

-----------------------------------------------------------------------------------------------------
--2. SELECT DISTINCT Statement
--Question 3. Display all the distinct product categories in the dataset. Expected output: Product Category

SELECT DISTINCT Product_Category
FROM retail_sales;
```

	PRODUCT_CATEGORY
1	Clothing
2	Beauty
3	Electronics

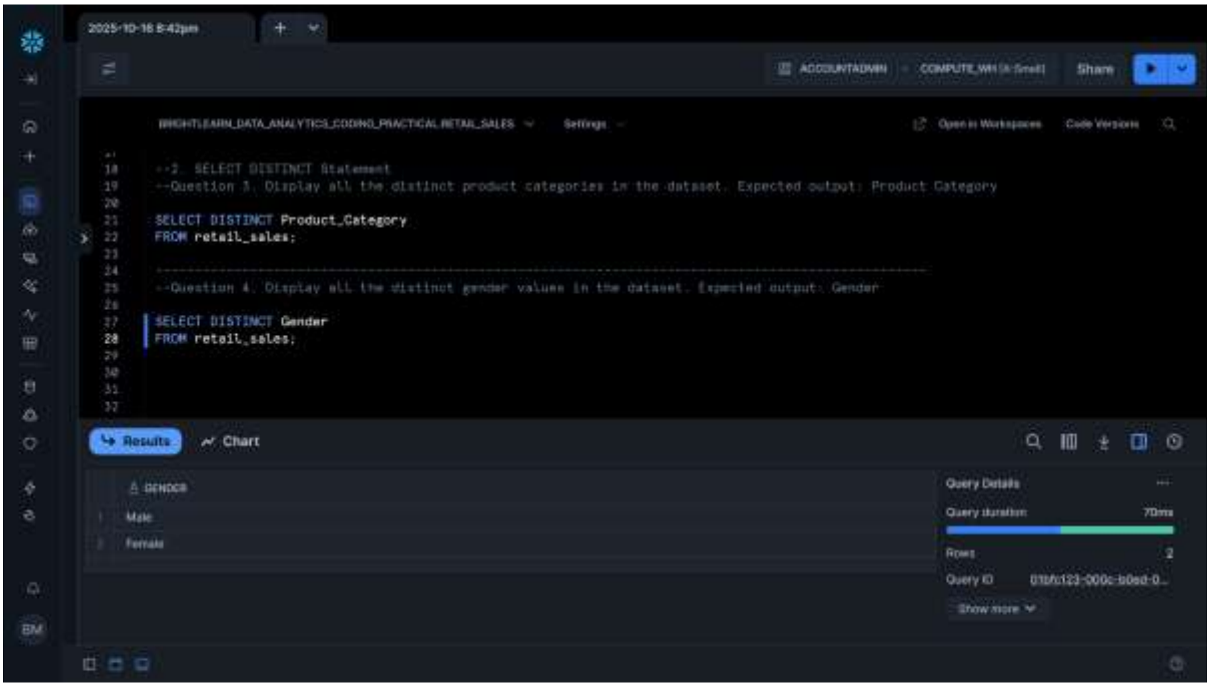
Query Details

Query duration 1.1s

Rows 3

Query ID 01bfc11a-000c-b0ed-00...

Q4. Display all the distinct gender values in the dataset. Expected output: Gender



```
--2. SELECT DISTINCT Statement
--Question 3. Display all the distinct product categories in the dataset. Expected output: Product Category

SELECT DISTINCT Product_Category
FROM retail_sales;

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

SELECT DISTINCT Gender
FROM retail_sales;
```

	GENDER
1	Male
2	Female

Query Details

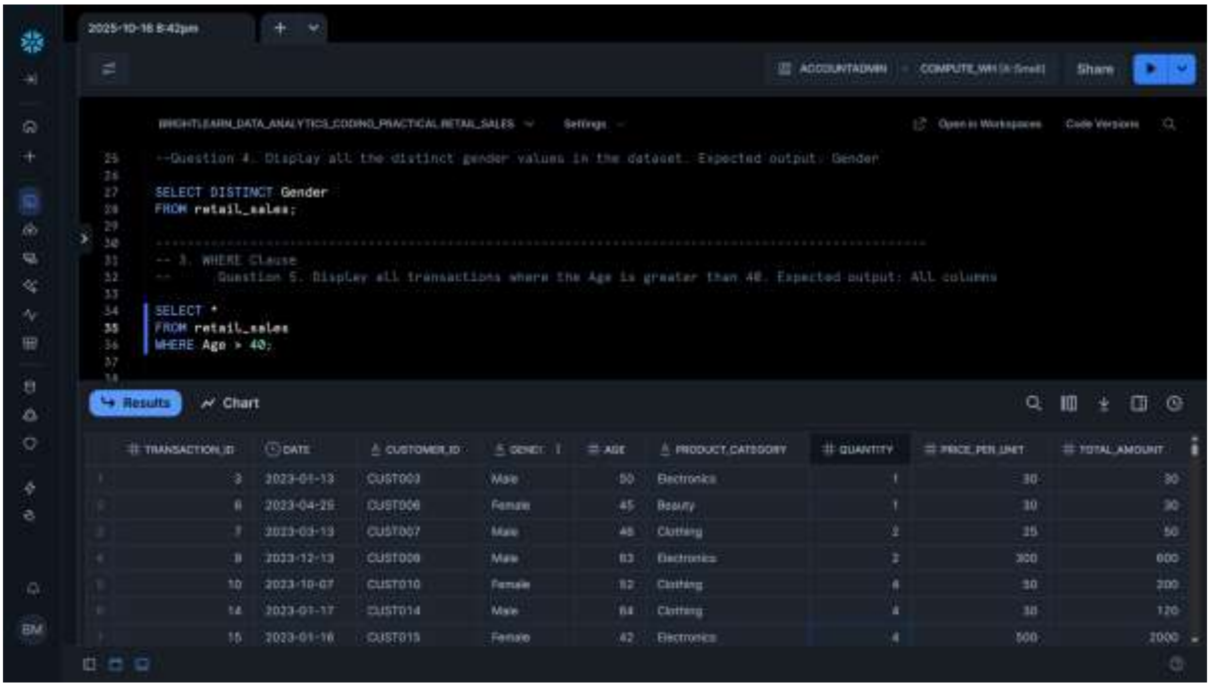
Query duration 70ms

Rows 2

Query ID 01bfc123-000c-b0ed-0...

3. WHERE Clause

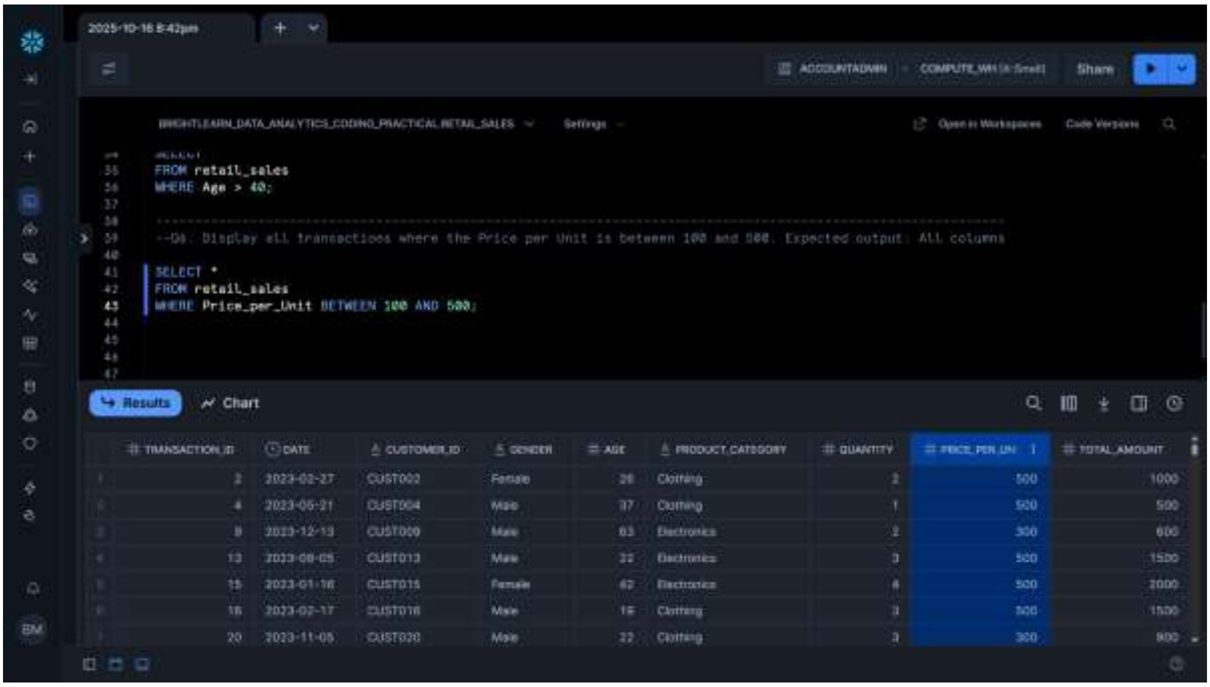
Q5. Display all transactions where the Age is greater than 40. Expected output: All columns



```sql
--Question 4. Display all the distinct gender values in the dataset. Expected output: Gender

SELECT DISTINCT Gender
FROM retail_sales;

-------------------------------------------------------------------------------
-- 3. WHERE Clause
--      Question 5. Display all transactions where the Age is greater than 40. Expected output: All columns

SELECT *
FROM retail_sales
WHERE Age > 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

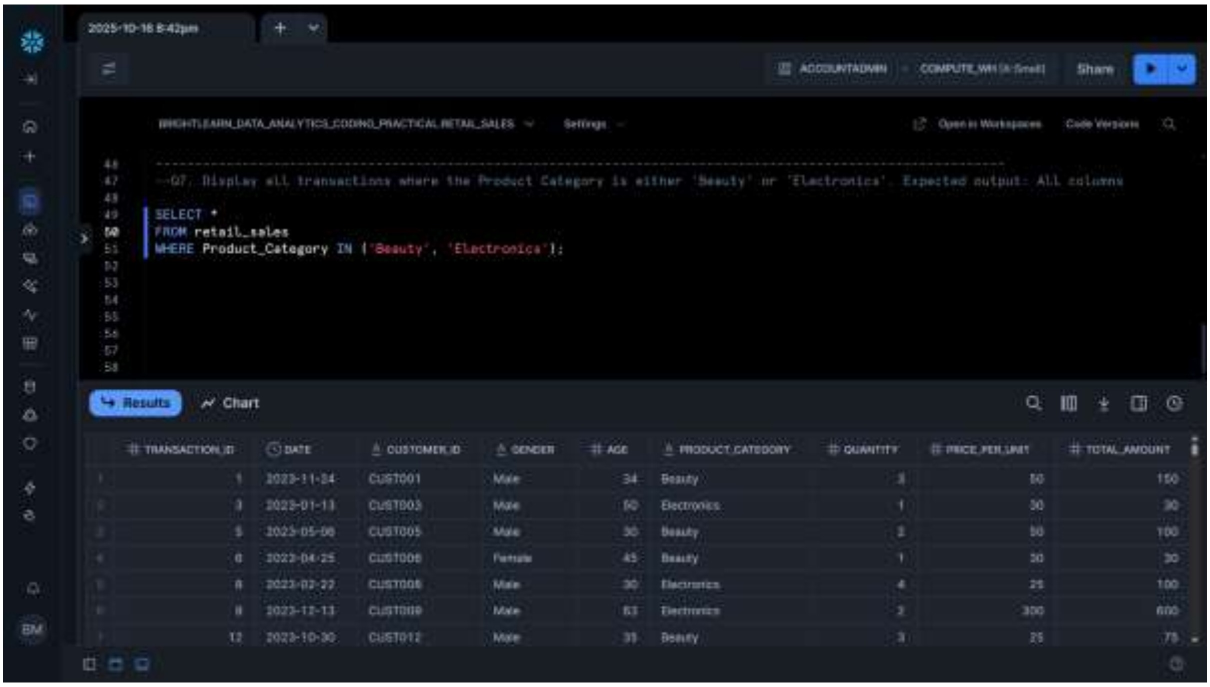
Q6. Display all transactions where the Price per Unit is between 100 and 500. *Expected output:* All columns



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

-------------------------------------------------------------------------------
--Q6. Display all transactions where the Price per Unit is between 100 and 500. Expected output: All columns

SELECT *
FROM retail_sales
WHERE Price_per_Unit 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

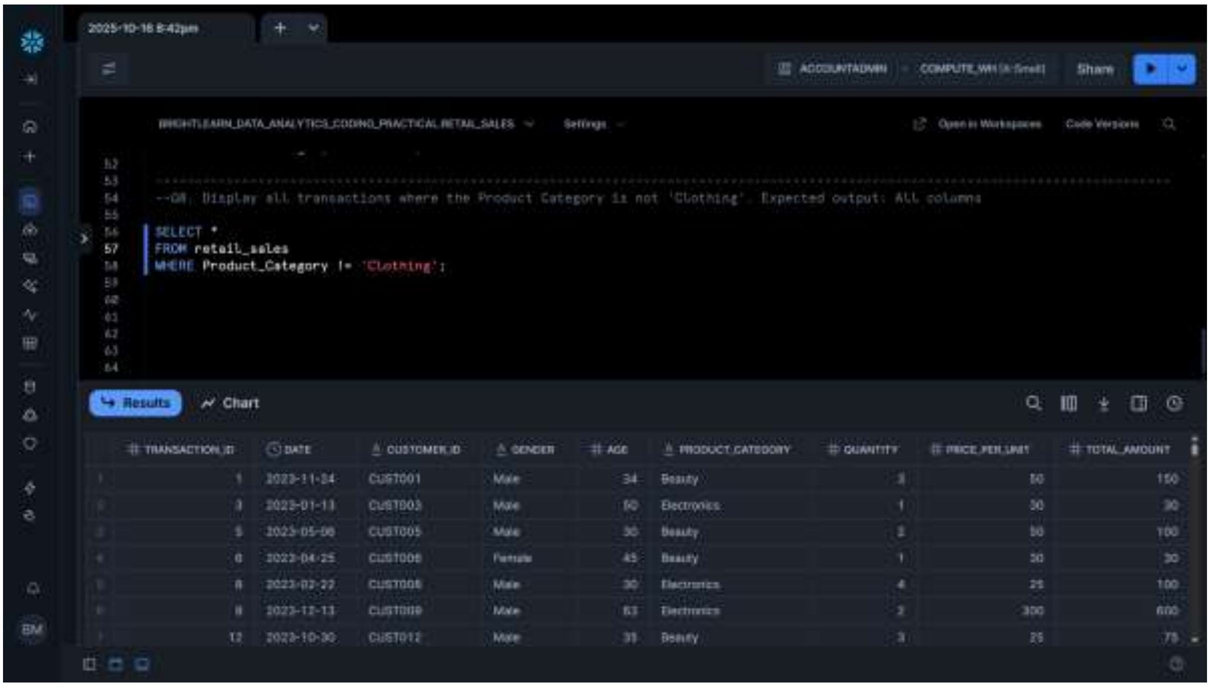
Q7. Display all transactions where the Product Category is either 'Beauty' or 'Electronics'. *Expected output:* All columns



```sql
------------------------------------------------------------------------------------------
--Q7. Display all transactions where the Product Category is either 'Beauty' or 'Electronics'. Expected output: All columns

SELECT *
FROM retail_sales
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

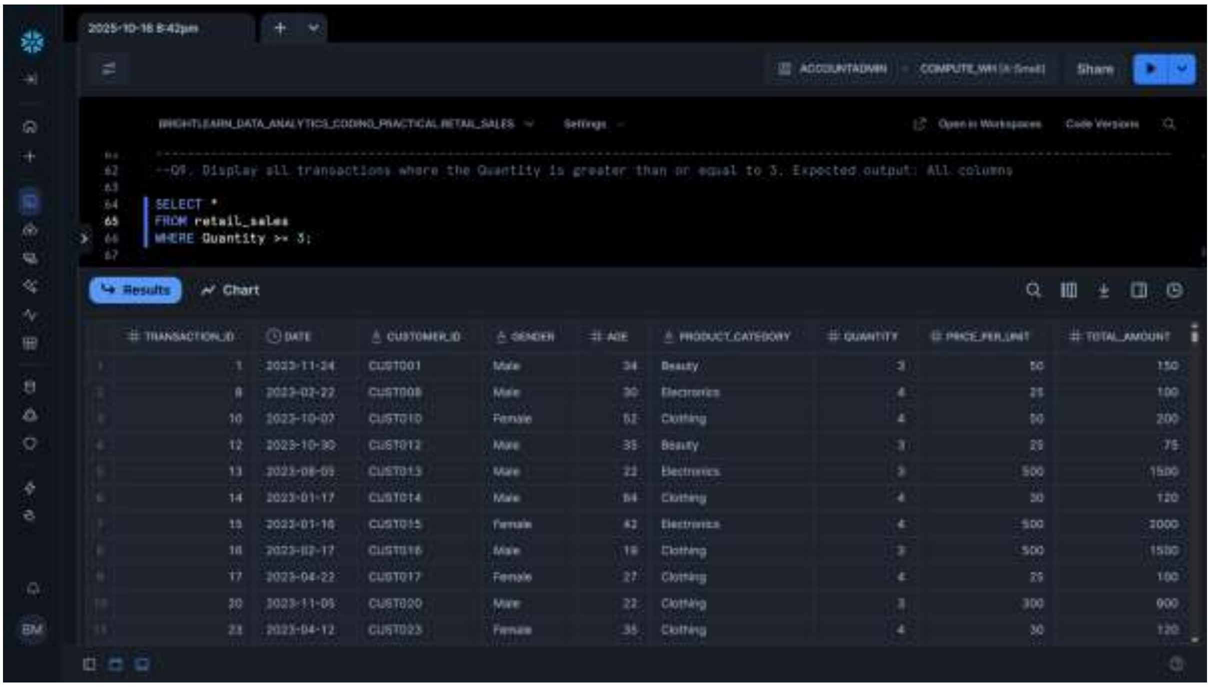
Q8. Display all transactions where the Product Category is not 'Clothing'. Expected output: All columns



```sql
------------------------------------------------------------------------------------------
--Q8. Display all transactions where the Product Category is not 'Clothing'. Expected output: All columns

SELECT *
FROM retail_sales
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

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

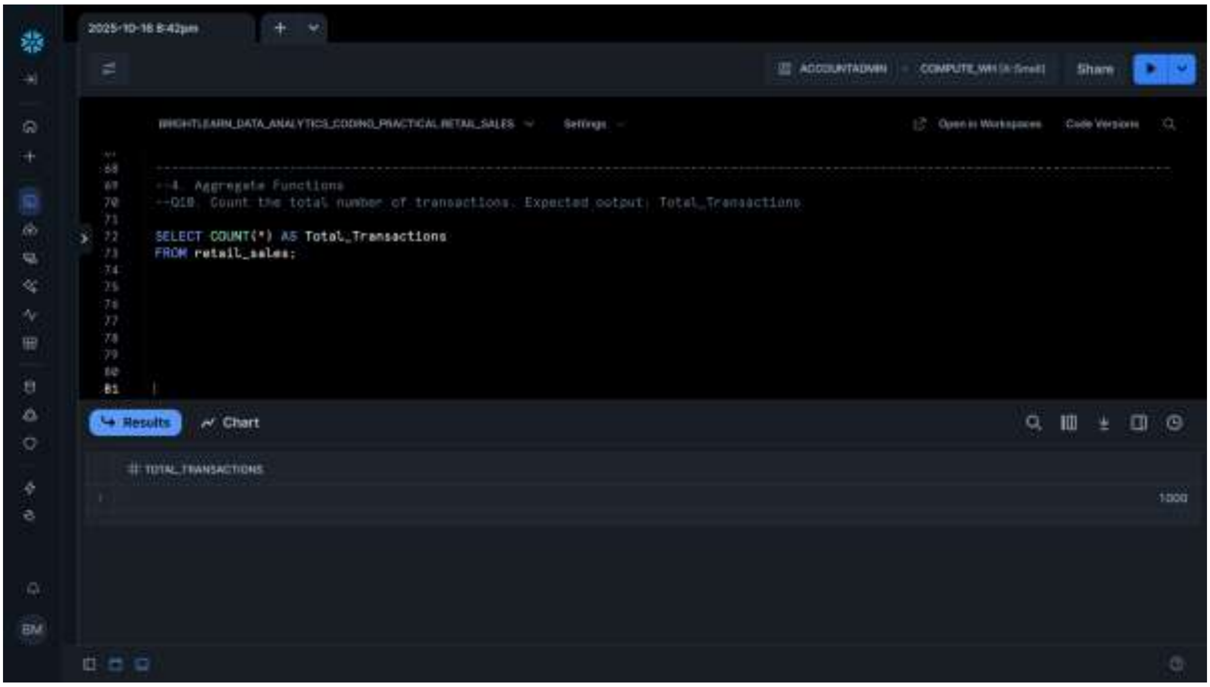


```
--Q9. Display all transactions where the Quantity is greater than or equal to 3. Expected output: All columns

SELECT *
FROM retail_sales
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

4. Aggregate Functions

Q10. Count the total number of transactions. *Expected output:* Total_Transactions

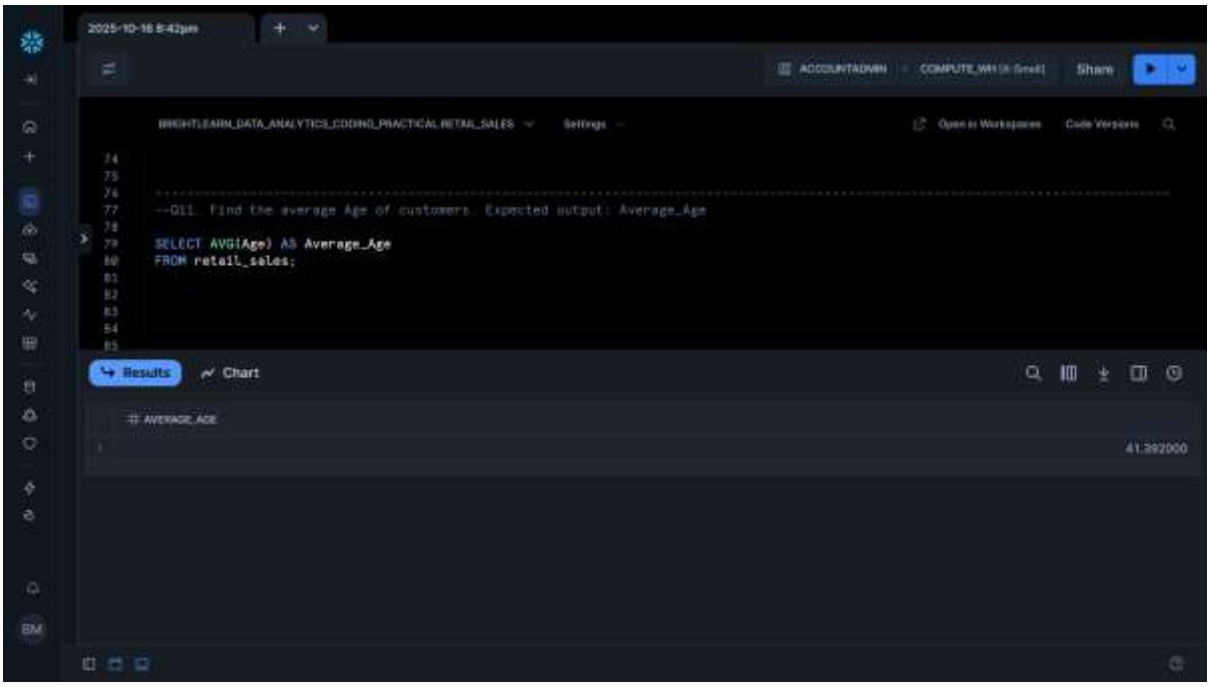


BRIGHTLEARN_DATA_ANALYTICS_CODING_PRACTICAL.RETAIL_SALES

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

SELECT COUNT(*) AS Total_Transactions
FROM retail_sales;
```

TOTAL_TRANSACTIONS
1000

Q11. Find the average Age of customers. *Expected output:* Average_Age

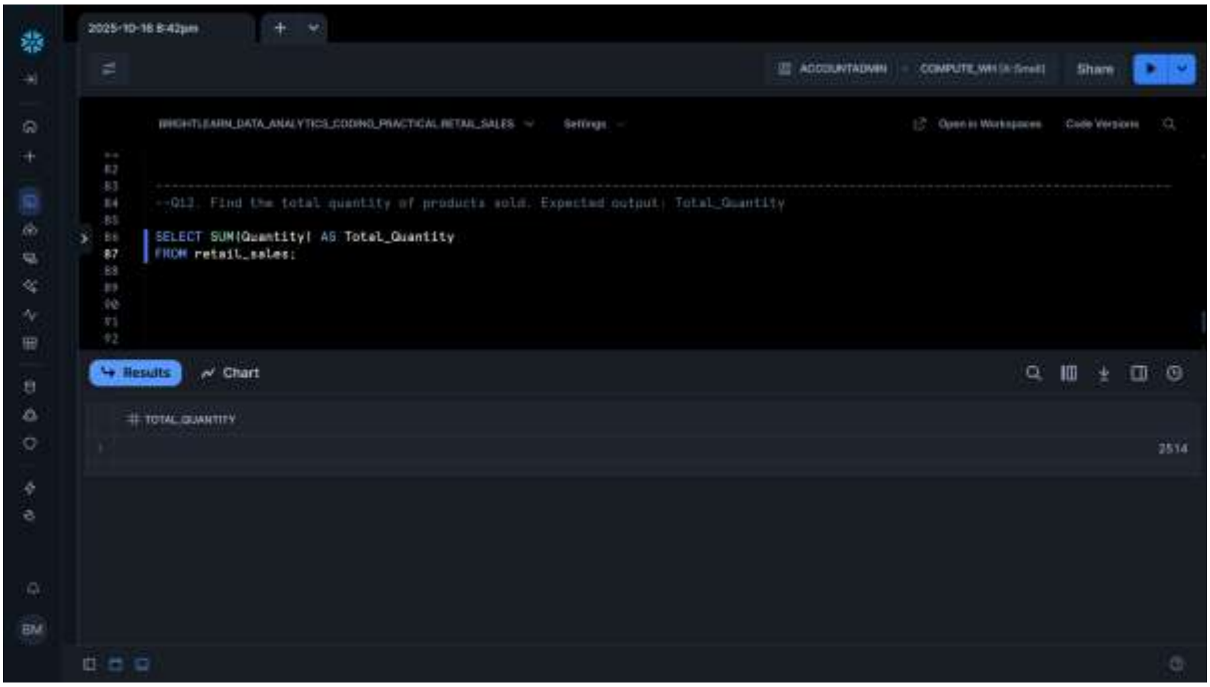


BRIGHTLEARN_DATA_ANALYTICS_CODING_PRACTICAL.RETAIL_SALES

```sql
------------------------------------------------------------------------------------------
--Q11. Find the average Age of customers. Expected output: Average_Age

SELECT AVG(Age) AS Average_Age
FROM retail_sales;
```

AVERAGE_AGE
41.392000

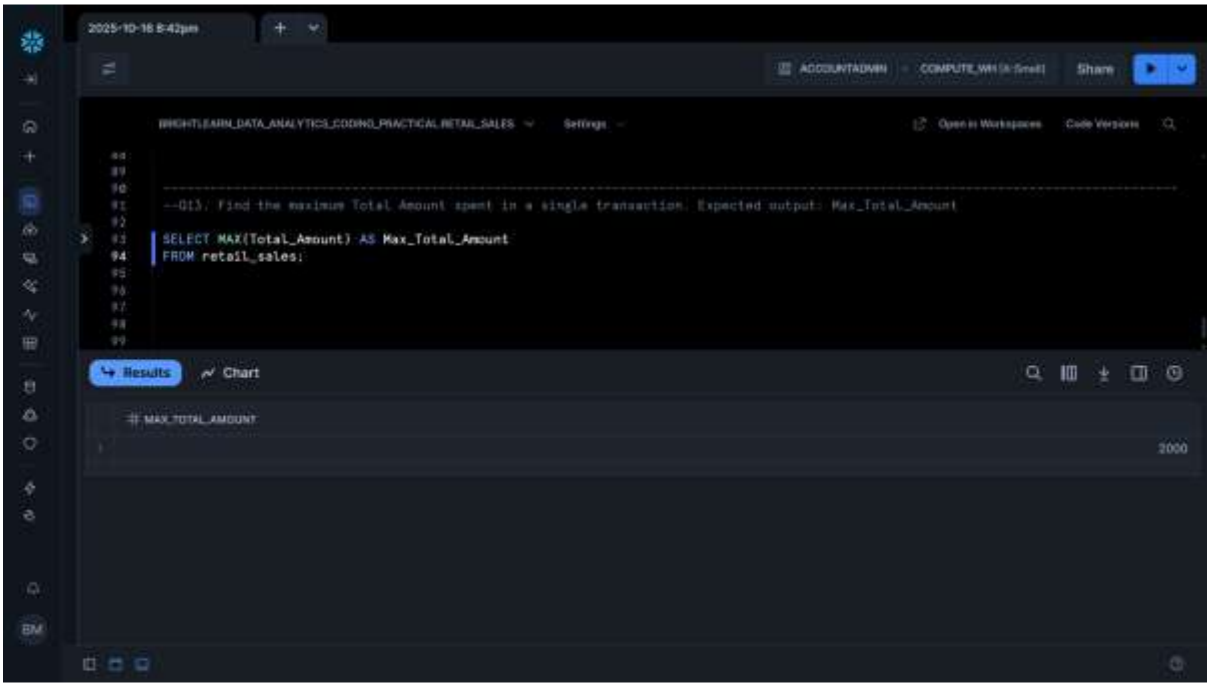
Q12. Find the total quantity of products sold. *Expected output:* Total_Quantity



```
-----------------------------------------------------------------------------------------------------------
--Q12. Find the total quantity of products sold. Expected output: Total_Quantity

SELECT SUM(Quantity) AS Total_Quantity
FROM retail_sales;
```

```
# TOTAL_QUANTITY
                    2514
```

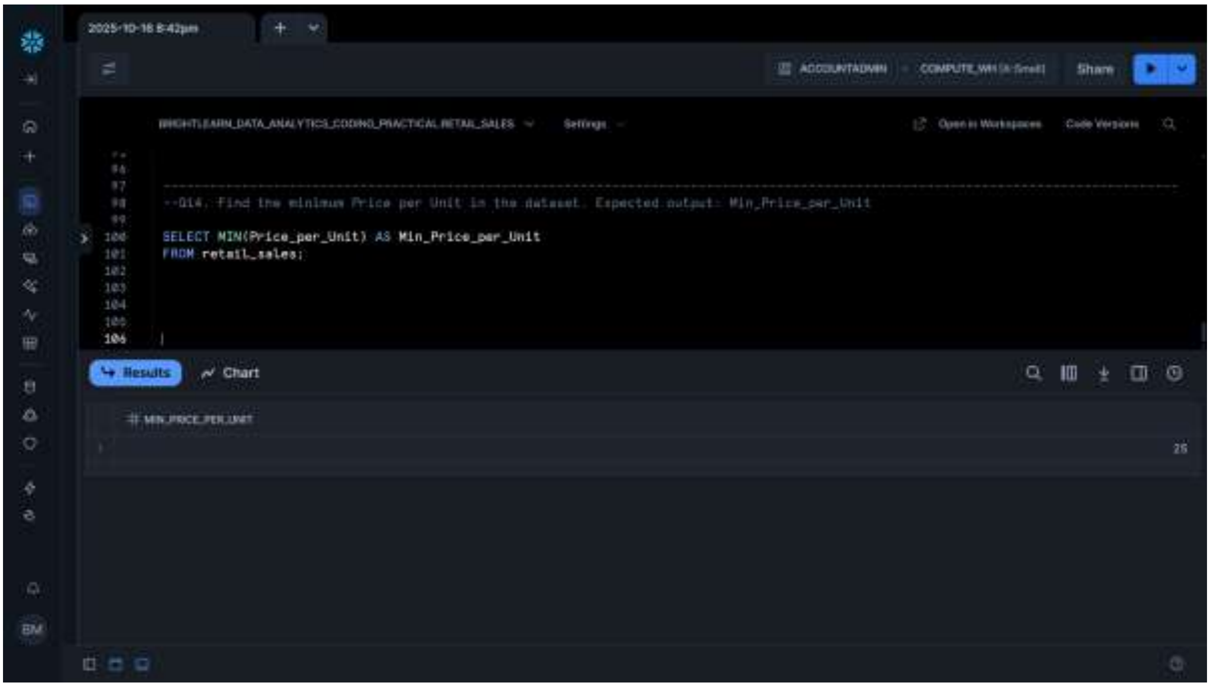
Q13. Find the maximum Total Amount spent in a single transaction. *Expected output:* Max_Total_Amount



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

SELECT MAX(Total_Amount) AS Max_Total_Amount
FROM retail_sales;
```

```
# MAX_TOTAL_AMOUNT
                    2000
```

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

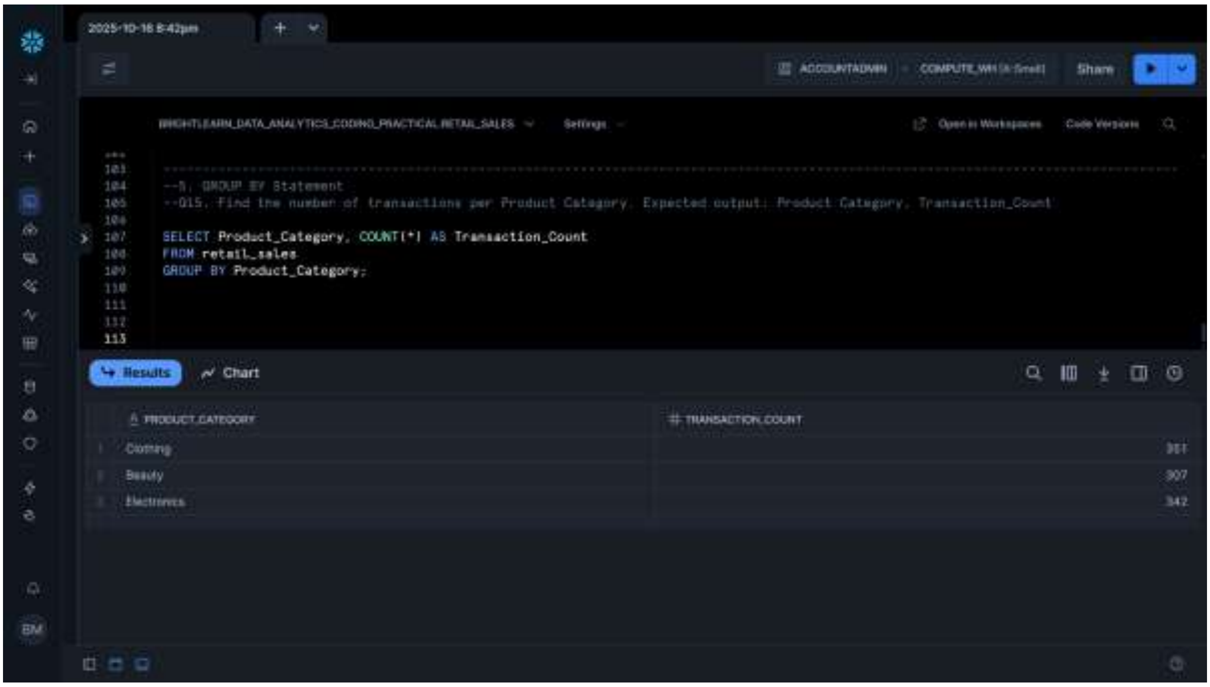


```sql
--------------------------------------------------------------------------------------------------------
--Q14. Find the minimum Price per Unit in the dataset. Expected output: Min_Price_per_Unit

SELECT MIN(Price_per_Unit) AS Min_Price_per_Unit
FROM retail_sales;
```

	# MIN_PRICE_PER_UNIT
1	25

5. GROUP BY Statement

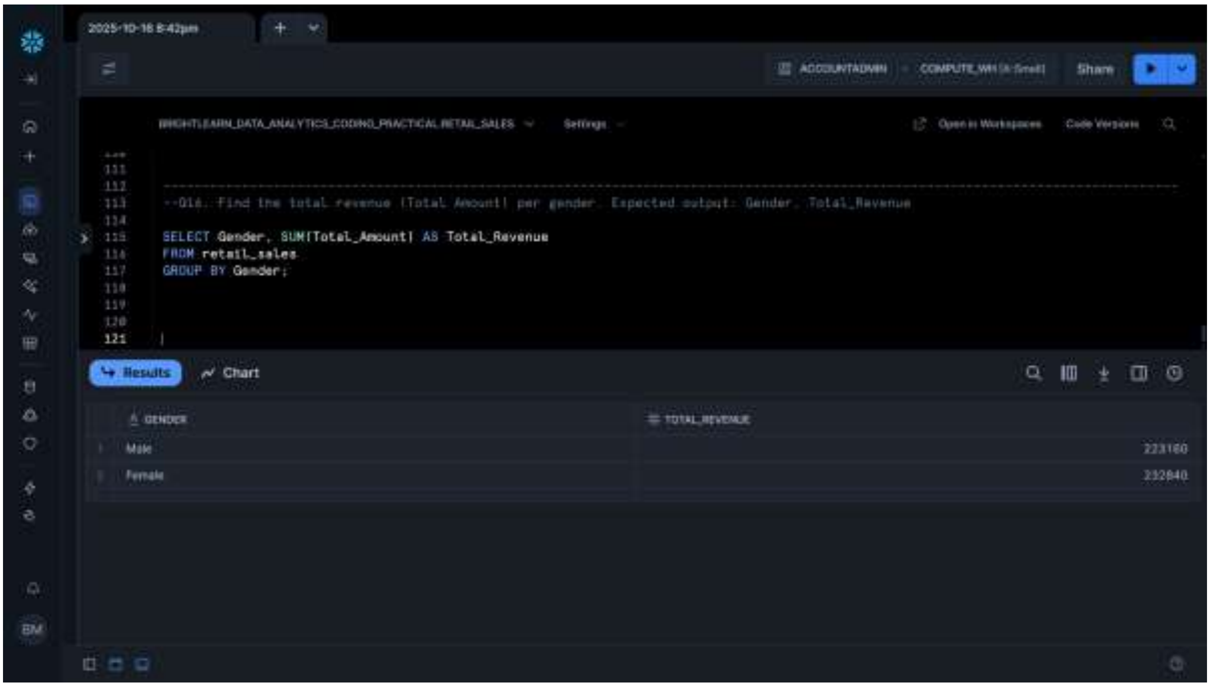
Q15. Find the number of transactions per Product Category. *Expected output:* Product Category, Transaction_Count



```sql
------------------------------------------------------------------------------------------
--5. GROUP BY Statement
--Q15. Find the number of transactions per Product Category. Expected output: Product Category, Transaction_Count

SELECT Product_Category, COUNT(*) AS Transaction_Count
FROM retail_sales
GROUP BY Product_Category;
```

	PRODUCT_CATEGORY	TRANSACTION_COUNT
1	Clothing	351
2	Beauty	307
3	Electronics	342

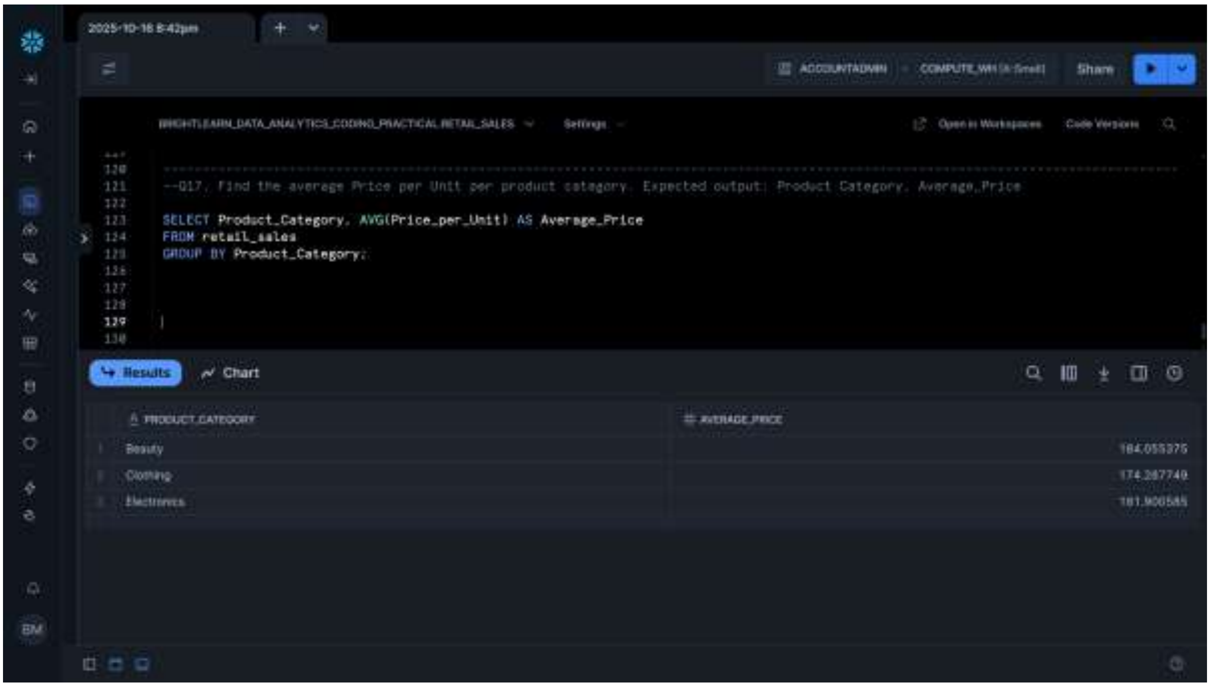
Q16. Find the total revenue (Total Amount) per gender. *Expected output:* Gender, Total_Revenue



```sql
------------------------------------------------------------------------------------------
--Q16. Find the total revenue (Total Amount) per gender. Expected output: Gender, Total_Revenue

SELECT Gender, SUM(Total_Amount) AS Total_Revenue
FROM retail_sales
GROUP BY Gender;
```

	GENDER	TOTAL_REVENUE
1	Male	223160
2	Female	232840

Q17. Find the average Price per Unit per product category. *Expected output:* Product Category, Average_Price

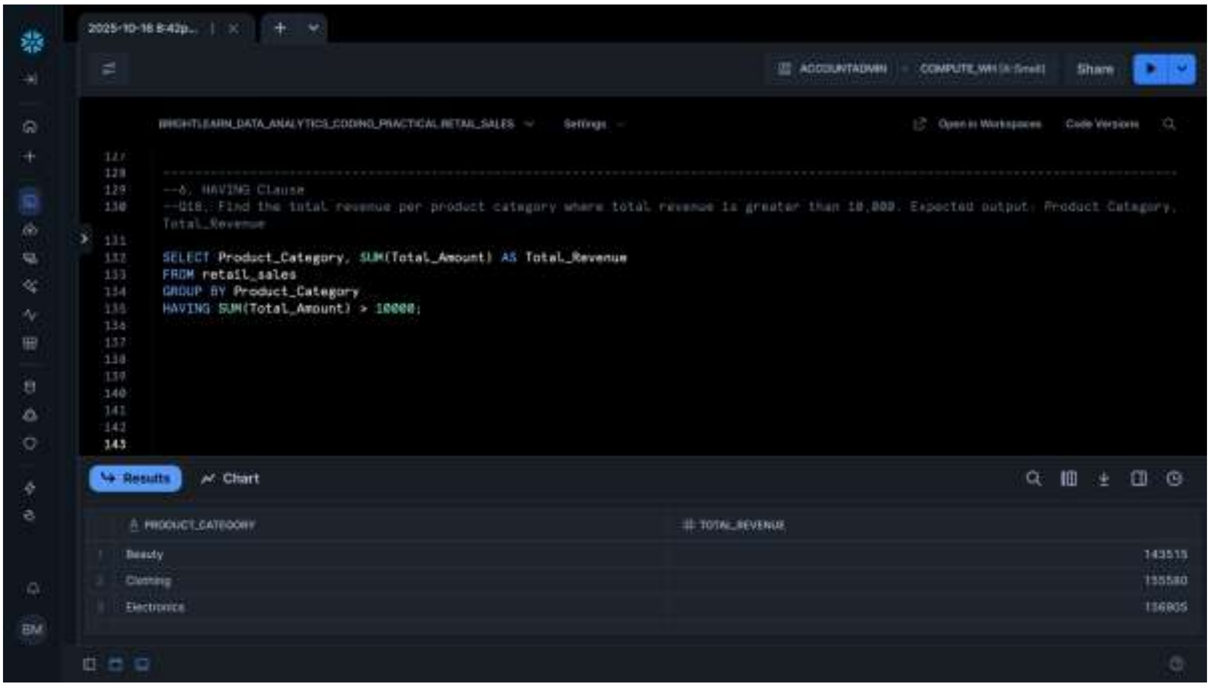


```sql
--Q17. Find the average Price per Unit per product category. Expected output: Product Category, Average_Price

SELECT Product_Category, AVG(Price_per_Unit) AS Average_Price
FROM retail_sales
GROUP BY Product_Category;
```

	PRODUCT_CATEGORY	AVERAGE_PRICE
1	Beauty	184.055375
2	Clothing	174.287749
3	Electronics	181.900585

6. HAVING Clause

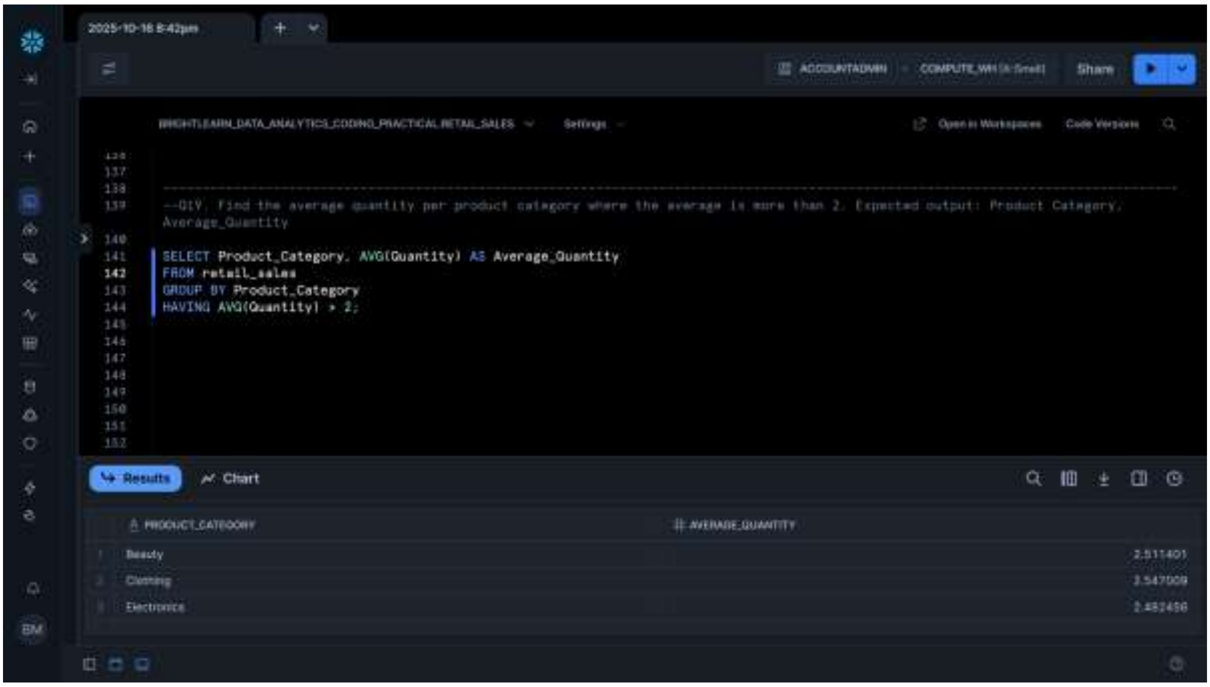
Q18. Find the total revenue per product category where total revenue is greater than 10,000. Expected output: Product Category, Total_Revenue



```sql
------------------------------------------------------------------------------------------------
--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

SELECT Product_Category, SUM(Total_Amount) AS Total_Revenue
FROM retail_sales
GROUP BY Product_Category
HAVING SUM(Total_Amount) > 10000;
```

	PRODUCT_CATEGORY	TOTAL_REVENUE
1	Beauty	143515
2	Clothing	155580
3	Electronics	156905

Q19. Find the average quantity per product category where the average is more than 2. *Expected output:* Product Category, Average_Quantity

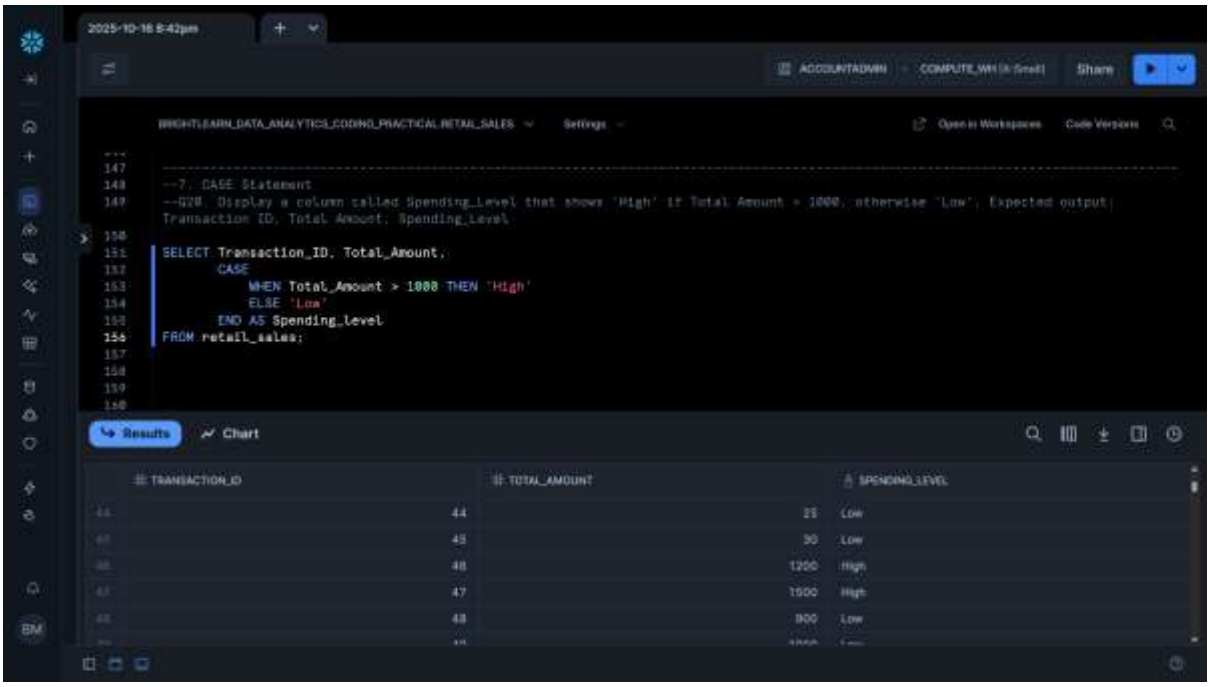


```sql
------------------------------------------------------------------------------------------------
--Q19. Find the average quantity per product category where the average is more than 2. Expected output: Product Category, Average_Quantity

SELECT Product_Category, AVG(Quantity) AS Average_Quantity
FROM retail_sales
GROUP BY Product_Category
HAVING AVG(Quantity) > 2;
```

	PRODUCT_CATEGORY	AVERAGE_QUANTITY
1	Beauty	2.511401
2	Clothing	2.547009
3	Electronics	2.482456

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

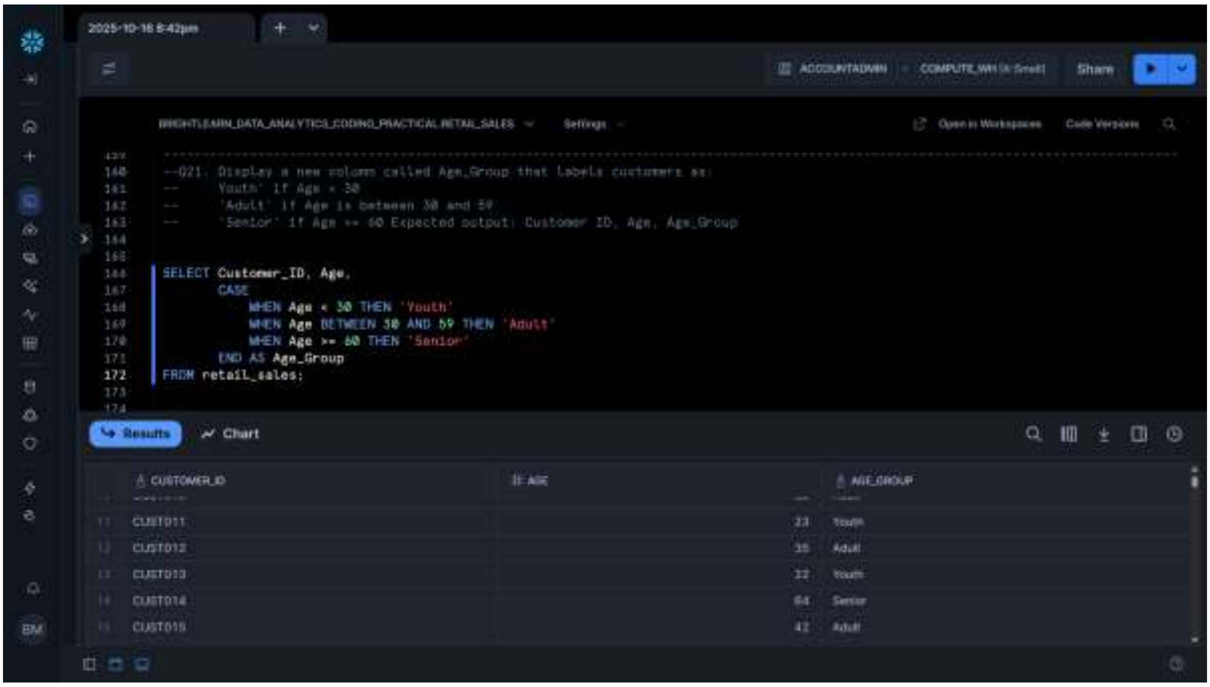


```
--------------------------------------------------------------------------------
--7. CASE Statement
--Q20. Display a column called Spending_Level that shows 'High' if Total Amount > 1000, otherwise 'Low'. Expected output:
Transaction ID, Total Amount, Spending_Level

SELECT Transaction_ID, Total_Amount,
       CASE
           WHEN Total_Amount > 1000 THEN 'High'
           ELSE 'Low'
       END AS Spending_Level
FROM retail_sales;
```

	TRANSACTION_ID	TOTAL_AMOUNT	SPENDING_LEVEL
44	44	25	Low
45	45	30	Low
46	46	1200	High
47	47	1500	High
48	48	900	Low

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



```sql
--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


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;
```

	CUSTOMER_ID	AGE	AGE_GROUP
11	CUST011	23	Youth
12	CUST012	35	Adult
13	CUST013	22	Youth
14	CUST014	64	Senior
15	CUST015	42	Adult