

Project: Practical exercise (Google Big Query)

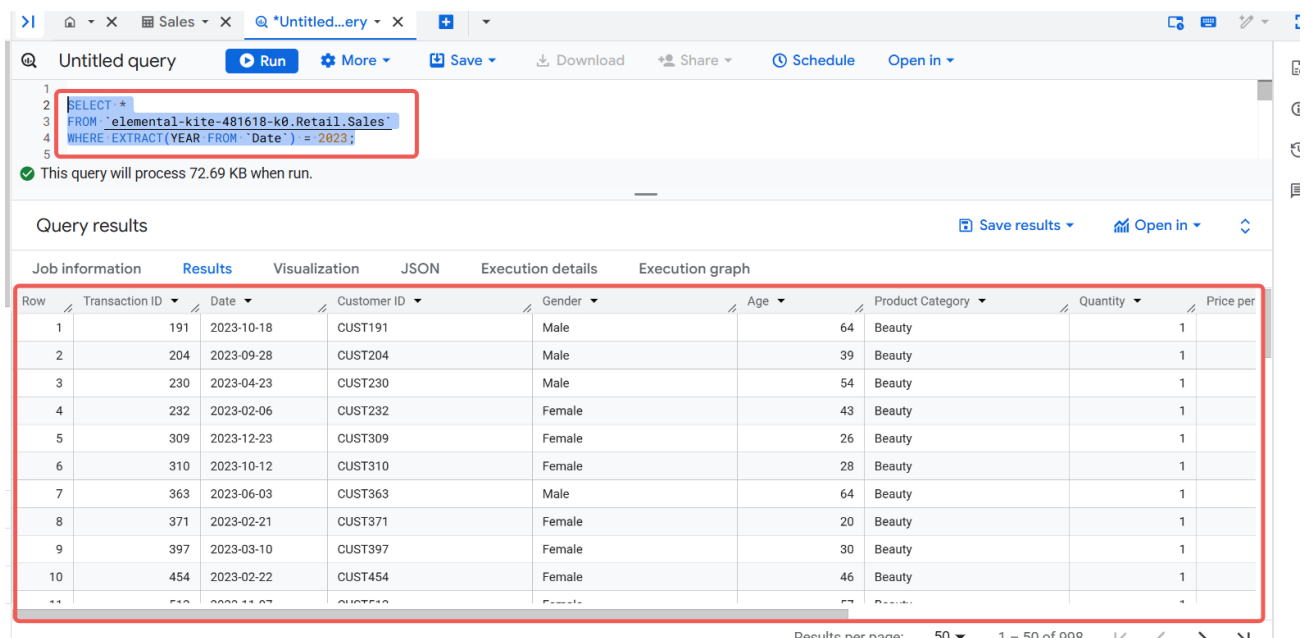
Dataset: Retail Sales Dataset

WHERE Clause

Q1. Filter all transactions that occurred in the year 2023.

Expected output: All columns

ANS:



The screenshot shows the Google BigQuery interface. At the top, there's a query editor with the following SQL query:

```
1 SELECT *
2 FROM `elemental-kite-481618-k0.Retail.Sales`
3 WHERE EXTRACT(YEAR FROM `Date`) = 2023;
```

Below the query editor, it says "This query will process 72.69 KB when run." Underneath, there's a section for "Query results" with tabs for "Job information", "Results", "Visualization", "JSON", "Execution details", and "Execution graph". The "Results" tab is selected, showing a table with 11 rows and 10 columns. The columns are: Row, Transaction ID, Date, Customer ID, Gender, Age, Product Category, Quantity, and Price per. The data shows transactions from 2023-10-18 to 2023-02-22, all with a quantity of 1 and a price per of 1.

Row	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per
1	191	2023-10-18	CUST191	Male	64	Beauty	1	1
2	204	2023-09-28	CUST204	Male	39	Beauty	1	1
3	230	2023-04-23	CUST230	Male	54	Beauty	1	1
4	232	2023-02-06	CUST232	Female	43	Beauty	1	1
5	309	2023-12-23	CUST309	Female	26	Beauty	1	1
6	310	2023-10-12	CUST310	Female	28	Beauty	1	1
7	363	2023-06-03	CUST363	Male	64	Beauty	1	1
8	371	2023-02-21	CUST371	Female	20	Beauty	1	1
9	397	2023-03-10	CUST397	Female	30	Beauty	1	1
10	454	2023-02-22	CUST454	Female	46	Beauty	1	1
11	510	2023-11-07	CUST510	Female	57	Beauty	1	1

At the bottom right, it says "Results per page: 50" and "1 - 50 of 998".

Filtering + Conditions

Q2. Display all transactions where the Total Amount is more than the average Total Amount of the entire dataset.

Expected output: All columns

ANS:

The screenshot shows a SQL query editor interface. The query is as follows:

```
SELECT  
  'Transaction ID',  
  'Customer ID',  
  'Total Amount' AS Total_Amount,  
  Date  
FROM `elemental-kite-481618-k0.Retail.Sales`  
WHERE `Total_Amount` >  
      (SELECT AVG(`Total_Amount`)  
       FROM `elemental-kite-481618-k0.Retail.Sales`);
```

Below the query editor, a status bar indicates "Query completed" and "Using on-demand processing quota".

The "Query results" section shows a table with the following data:

Row	f0_	f1_	Total_Amount	Date
1	Transaction ID	Customer ID	500	2023-01-14
2	Transaction ID	Customer ID	500	2023-04-23
3	Transaction ID	Customer ID	500	2023-07-05
4	Transaction ID	Customer ID	500	2023-03-03
5	Transaction ID	Customer ID	500	2023-01-17
6	Transaction ID	Customer ID	500	2023-08-23

Aggregate Functions

Q3. Calculate the total revenue (sum of Total Amount). Expected output: Total_Revenue

ANS:

```
13
14
15
16 SELECT
17     SUM('Total Amount') AS Total_Revenue
18 FROM 'elemental-kite-481618-k0.Retail.Sales';
19
20
21
```

✓ Query completed
Using on-demand processing quota

Query results [Save results](#) [Open in](#)

Job information **Results** Visualization JSON Execution details Execution graph

Row	Total_Revenue
1	456000

DISTINCT

Q4. Display all distinct Product Categories in the dataset.

Expected output: Product_Category

ANS

```
19
20
21 SELECT DISTINCT
22     'Product Category'
23 FROM 'elemental-kite-481618-k0.Retail.Sales';
24
25
26
```

✓ This query will process 0 B when run.

Query results [Save results](#) [Open in](#)

Job information **Results** Visualization JSON Execution details Execution graph

Row	Product Category
1	

GROUP BY

Q5. For each Product Category, calculate the total quantity sold.

Expected output: Product Category, Total Quantity

ANS:

The screenshot shows a SQL query editor interface. At the top, there's a toolbar with icons for home, close, and tabs. The active tab is titled '*Untitled...ery'. Below the toolbar, the query text is displayed in a monospace font. The query is: `SELECT 'Product Category', SUM(Quantity) AS Total_Quantity FROM `elemental-kite-481618-k0.Retail.Sales` GROUP BY `Product Category`;`. The query is highlighted with a red box. Below the query, a green checkmark icon indicates that the query will process 17.98 KB when run. A button labeled 'Run' is visible. Below the query editor, there's a section titled 'Query results' with a 'Save' button. The results are displayed in a table with columns 'Product Category' and 'Total_Quantity'. The table has three rows: 'Beauty' with a total quantity of 771, 'Clothing' with 894, and 'Electronics' with 849. The table is also highlighted with a red box.

```
SELECT 'Product Category',
SUM(Quantity) AS Total_Quantity
FROM `elemental-kite-481618-k0.Retail.Sales`
GROUP BY `Product Category`;
```

✓ This query will process 17.98 KB when run.

Using on-demand processing quota

Query results

Save

Job information	Results	Visualization	JSON	Execution details	Execution
Row	Product Category	Total_Quantity			
1	Beauty	771			
2	Clothing	894			
3	Electronics	849			

CASE Statement

Q6. Create a column called Age_Group that classifies customers as 'Youth' (<30), 'Adult' (30-59), 'Senior' (60+)

Expected output: Customer_ID, Age, Age_Group

ANS:

The screenshot shows a SQL query editor with a query titled "Untitled query". The query is as follows:

```
SELECT
  `Customer ID`,
  Age,
  CASE
    WHEN Age < 30 THEN '01.Youth'
    WHEN Age BETWEEN 30 AND 59 THEN '02.Adult'
    ELSE '03.Senior'
  END AS Age_Group
FROM `elemental-kite-481618-k0.Retail.Sales`;
```

Below the query, a status bar indicates "Query completed" and "Using on-demand processing quota".

The "Query results" section shows a table with the following data:

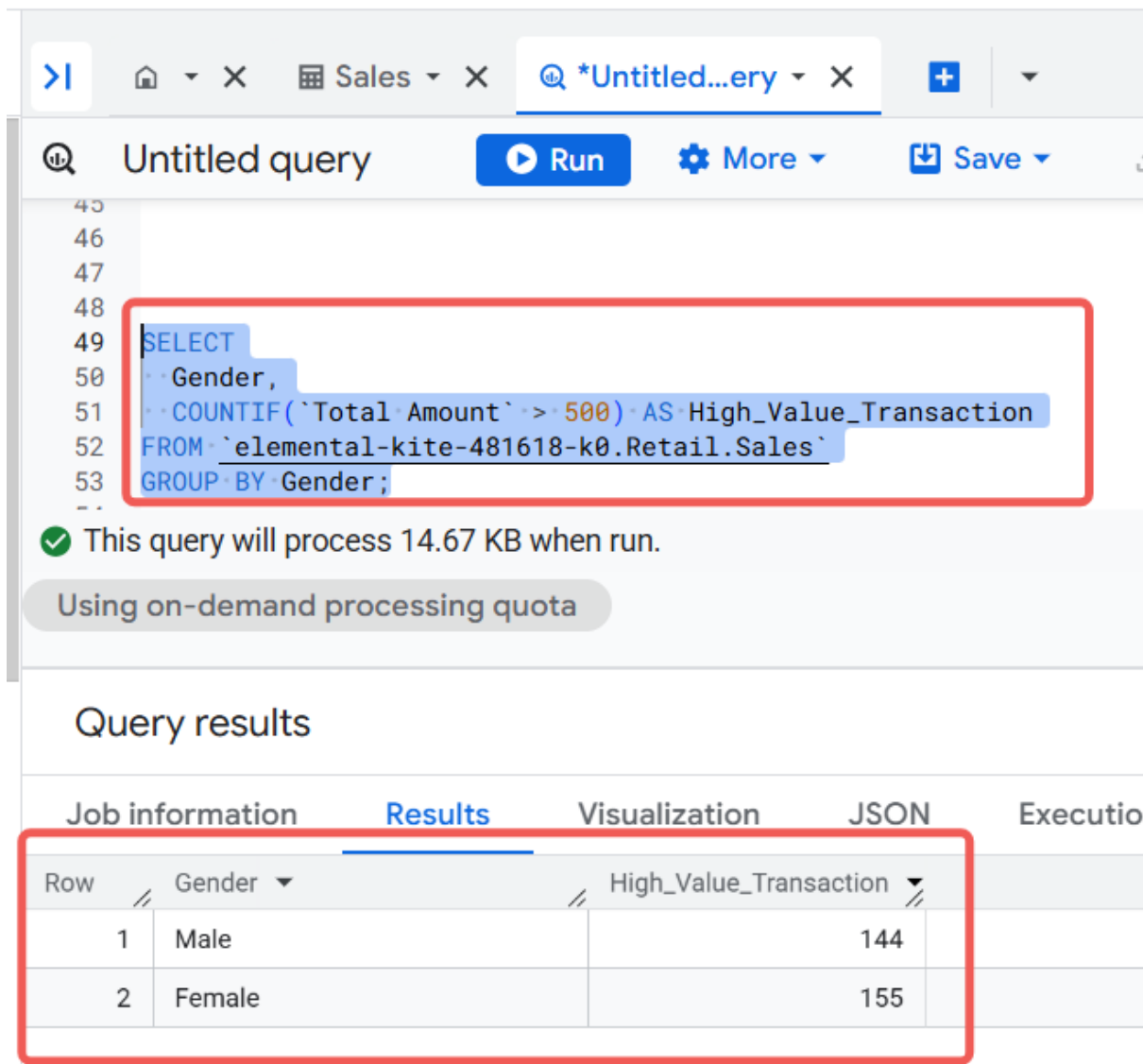
Row	Customer ID	Age	Age_Group
1	CUST191	64	03.Senior
2	CUST204	39	02.Adult
3	CUST230	54	02.Adult
4	CUST232	43	02.Adult
5	CUST309	26	01.Youth
6	CUST310	28	01.Youth

Conditional Aggregation

Q7. For each Gender, count how many high-value transactions occurred (where the Total Amount > 500).

Expected output: Gender, High_Value_Transactions

ANS:



The screenshot shows a SQL query editor interface. The query is as follows:

```
SELECT  
  Gender,  
  COUNTIF(`Total Amount` > 500) AS High_Value_Transaction  
FROM `elemental-kite-481618-k0.Retail.Sales`  
GROUP BY Gender;
```

Below the query editor, a status message indicates: "This query will process 14.67 KB when run. Using on-demand processing quota".

The "Query results" section shows a table with the following data:

Row	Gender	High_Value_Transaction
1	Male	144
2	Female	155

HAVING Clause

Q8. For each Product Category, show only those categories where the total revenue exceeds 5,000. Expected output: Product_Category, Total_Revenue

ANS:

The screenshot shows a SQL query editor interface. The query is as follows:

```
56  
57 SELECT  
58   `Product_Category`,  
59   SUM(`Total_Amount`) AS Total_Revenue  
60 FROM `elemental-kite-481618-k0.Retail.Sales`  
61 GROUP BY `Product_Category`  
62 HAVING SUM(`Total_Amount`) > 5000;  
63
```

Below the query editor, a status message indicates: "This query will process 17.98 KB when run. Using on-demand processing quota".

The "Query results" section shows a table with the following data:

Row	Product_Category	Total_Revenue
1	Beauty	143515
2	Clothing	155580
3	Electronics	156905

Calculated Fields

Q9. Display a new column called Unit_Cost_Category that labels a transaction as: – 'Cheap' if Price per Unit < 50 – 'Moderate' if Price per Unit between 50 and 200 – 'Expensive' if Price per Unit > 200

Expected output: Transaction_ID, Price_per_Unit, Unit_Cost_Category

ANS:

The screenshot shows a SQL query editor interface. The query is as follows:

```
SELECT
  `Transaction ID`,
  `Price per Unit`,
  CASE
    WHEN `Price per Unit` < 50 THEN '01.Cheap'
    WHEN `Price per Unit` BETWEEN 50 AND 200 THEN '02.Moderate'
    ELSE '03.Expensive'
  END AS Unit_Cost_Category
FROM `elemental-kite-481618-k0.Retail.Sales`;
```

Below the query, a status message indicates: "This query will process 15.63 KB when run." and "Using on-demand processing quota".

The "Query results" section displays a table with the following data:

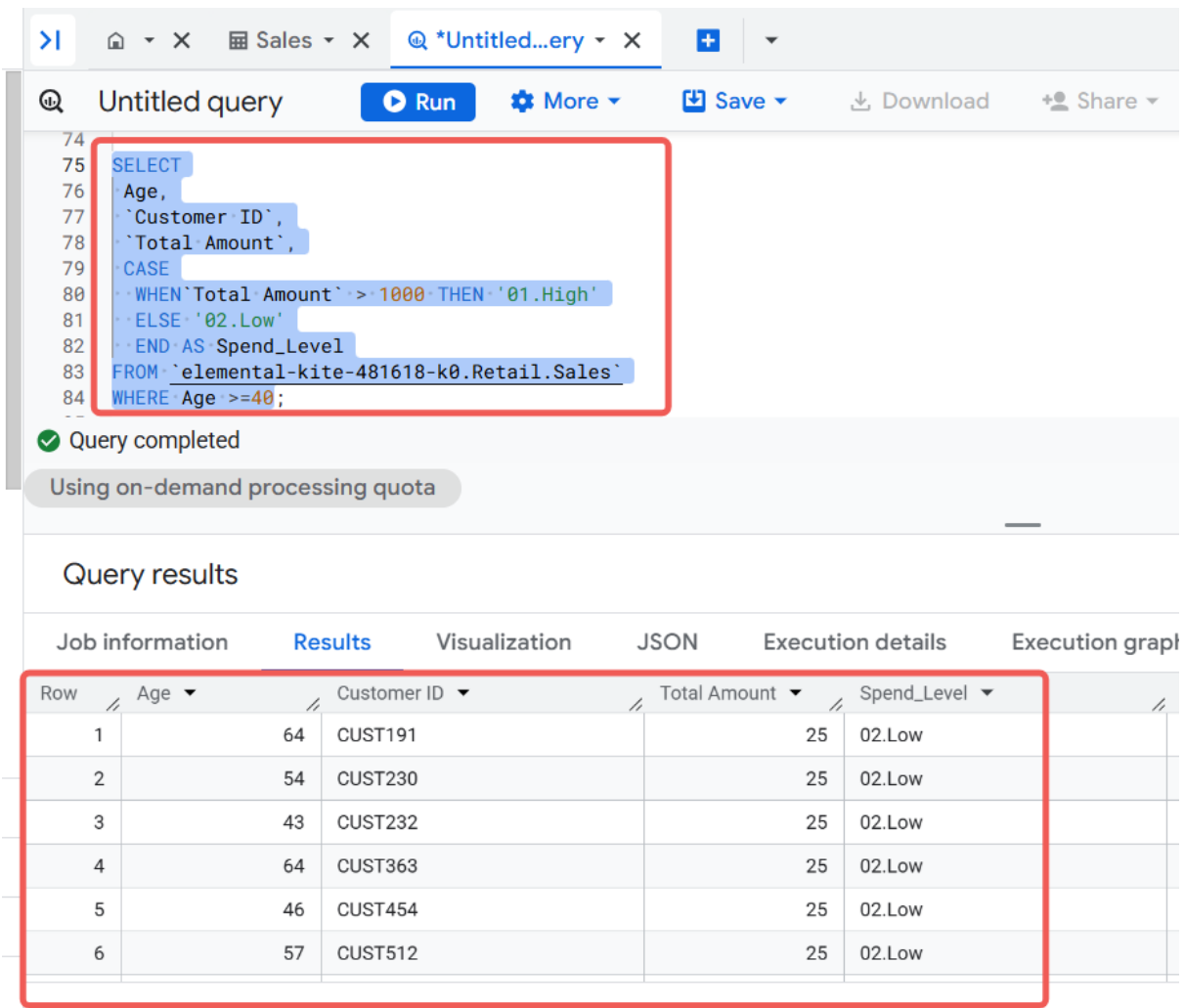
Row	Transaction ID	Price per Unit	Unit_Cost_Category
1	191	25	01.Cheap
2	204	25	01.Cheap
3	230	25	01.Cheap
4	232	25	01.Cheap
5	309	25	01.Cheap
6	310	25	01.Cheap
7	363	25	01.Cheap

Combining WHERE + CASE

Q10. Display all transactions from customers aged 40 or older and add a column Spending_Level showing 'High' if Total Amount > 1000, otherwise 'Low'.

Expected output: Customer_ID, Age, Total_Amount, Spending_Level

ANS:



```
SELECT
  Age,
  `Customer ID`,
  `Total Amount`,
  CASE
    WHEN `Total Amount` > 1000 THEN '01.High'
    ELSE '02.Low'
  END AS Spend_Level
FROM `elemental-kite-481618-k0.Retail.Sales`
WHERE Age >= 40;
```

Query completed

Using on-demand processing quota

Query results

Job information	Results	Visualization	JSON	Execution details	Execution graph
Row	Age	Customer ID	Total Amount	Spend_Level	
1	64	CUST191	25	02.Low	
2	54	CUST230	25	02.Low	
3	43	CUST232	25	02.Low	
4	64	CUST363	25	02.Low	
5	46	CUST454	25	02.Low	
6	57	CUST512	25	02.Low	