

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 Big Query interface. In the top-left, there's a tab labeled 'Sales' and another labeled '*Untitled...ery'. Below the tabs, the title bar says 'Untitled query'. There are buttons for 'Run', 'More', 'Save', 'Download', 'Share', 'Schedule', and 'Open in'. A message indicates the query will process 72.69 KB when run. The main area is titled 'Query results' and contains a table with 11 columns: Row, Transaction ID, Date, Customer ID, Gender, Age, Product Category, Quantity, and Price per. The table has 10 rows of data. The first row shows a transaction ID of 191 on 2023-10-18 for customer CUST191, male, 64 years old, in the Beauty category, quantity 1, price 1. The last row shows a transaction ID of 510 on 2023-11-07 for customer CUST510, female, 57 years old, in the Beauty category, quantity 1, price 1. The entire code block and the results table are highlighted with a red border.

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

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

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 data analysis interface with a query editor at the top and a results table below. The query editor contains the following SQL code:

```
11 SELECT
12   `Transaction ID`,
13   `Customer ID`,
14   `Total Amount` AS Total_Amount,
15   `Date`
16 FROM `elemental-kite-481618-k0.Retail.Sales`
17 WHERE `Total Amount` >
18   ((SELECT AVG(`Total Amount`))
19   FROM `elemental-kite-481618-k0.Retail.Sales`);
```

A red box highlights the WHERE clause and the subquery. Below the editor, a message says "Query completed".

The results table has the following structure:

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:

The screenshot shows a code editor with a red box highlighting the following SQL query:

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

Below the code, a message says "Query completed" and "Using on-demand processing quota".

The results section has tabs: Job information, **Results**, Visualization, JSON, Execution details, and Execution graph. The Results tab is selected. It shows a table with one row:

Row	Total_Revenue
1	456000

DISTINCT

Q4. Display all distinct Product Categories in the dataset.

Expected output: Product_Category

ANS

The screenshot shows a code editor with a red box highlighting the following SQL query:

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

Below the code, a message says "This query will process 0 B when run."

The results section has tabs: Job information, **Results**, Visualization, JSON, Execution details, and Execution graph. The Results tab is selected. It shows a table with one row:

Row	Product Category
1	Product Category

GROUP BY

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

Expected output: Product Category, Total Quantity

ANS:

The screenshot shows a data processing interface with a query editor at the top and a results table below. The query editor contains the following SQL code:

```
26
27
28 SELECT `Product Category`,
29      SUM(Quantity) AS Total_Quantity
30  FROM `elemental-kite-481618-k0.Retail.Sales`
31 GROUP BY `Product Category`;
```

A red box highlights the entire query. Below the editor, a message says "This query will process 17.98 KB when run." and "Using on-demand processing quota".

The results table has a header row with "Row", "Product Category", and "Total_Quantity". The data rows are:

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 data processing interface with a query editor and a results table.

Query Editor:

```
33
34
35
36 SELECT
37   `Customer ID`,
38   Age,
39   CASE
40     WHEN Age < 30 THEN '01.Youth'
41     WHEN Age BETWEEN 30 AND 59 THEN '02.Adult'
42     ELSE '03.Senior'
43   END AS Age_Group
44 FROM `elemental-kite-481618-k0.Retail.Sales`;
```

A red box highlights the CASE statement logic.

Results Table:

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	CUST210	28	01.Youth

A red box highlights the first five rows of the results table.

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 data processing interface with a query editor and a results table.

Query Editor:

```
45
46
47
48
49 SELECT
50   Gender,
51   COUNTIF(`Total_Amount` > 500) AS High_Value_Transaction
52   FROM `elemental-kite-481618-k0.Retail.Sales`
53 GROUP BY Gender;
```

A red box highlights the SQL code from line 49 to 53.

Results:

Row	Gender	High_Value_Transaction
1	Male	144
2	Female	155

A red box highlights the results table.

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 data processing interface with a query editor and a results table.

Query Editor:

```
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
```

A red box highlights the entire query code.

Job Information:

- This query will process 17.98 KB when run.
- Using on-demand processing quota

Results:

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

A red box highlights the results table.

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 data processing interface with a query editor and a results table. The query editor contains the following SQL code:

```
63
64 SELECT
65   `Transaction ID`,
66   `Price per Unit`,
67   CASE
68     WHEN `Price per Unit` < 50 THEN '01.Cheap'
69     WHEN `Price per Unit` BETWEEN 50 AND 200 THEN '02.Moderate'
70     ELSE '03.Expensive'
71   END AS `Unit_Cost_Category`
72   FROM `elemental-kite-481618-k0.Retail.Sales`;
73
```

A red box highlights the CASE statement logic. Below the editor, a note says "This query will process 15.63 KB when run." and "Using on-demand processing quota".

Query results

The results table displays 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:

```
74
75 SELECT
76     Age,
77     `Customer ID`,
78     `Total Amount`,
79     CASE
80         WHEN `Total Amount` > 1000 THEN '01.High'
81         ELSE '02.Low'
82     END AS Spend_Level
83 FROM `elemental-kite-481618-k0.Retail.Sales`
84 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	