

PRACTICAL EXERCISE 5

ON GOOGLE BIGQUERY

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

Expected output: All columns

The screenshot shows a data processing interface with a query editor and a results table. The query is:

```
1 SELECT * FROM `amino-retail1-04891.sales_retail` LIMIT 1000;
2
3
4
5 ----Q1 Filter all transactions that occurred in the year 2023.
6
7 SELECT *
8 FROM `amino-retail1-04891.sales_retail`
9 WHERE EXTRACT(YEAR FROM DATE) = 2023;
```

The results table has columns: Row, Transaction ID, Date, Customer ID, Gender, Age, Product Category, Quantity, Price per Unit, Total Amount. The data shows 9 transactions from 2023 with various details like gender, age, product category, quantity, and price per unit.

Row	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
1	1V1	2023-10-18	CUST191	Male	60	Beauty	1	25	25
2	204	2023-09-28	CUST204	Male	39	Beauty	1	25	25
3	230	2023-04-01	CUST706	Male	54	Beauty	1	25	25
4	232	2023-02-09	CUST1202	Female	43	Beauty	1	25	25
5	309	2023-12-23	CUST309	Female	26	Beauty	1	25	25
6	312	2023-10-12	CUST316	Female	20	Beauty	1	25	25
7	303	2023-06-03	CUST1963	Male	60	Beauty	1	25	25
8	371	2023-02-21	CUST371	Female	20	Beauty	1	25	25
9	337	2023-09-10	CUST097	Female	30	Beauty	1	25	25

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

Expected output: All columns

The screenshot shows a data processing interface with a query editor and a results table. The query is:

```
20
21
22
23
24
25
26 ----Q2. Display all transactions where the Total Amount is more than the average Total Amount of the entire dataset
27
28 WITH CalculatedTable AS (
29   SELECT
30     *
31   FROM (
32     SELECT
33       AVG(TOTAL_AMOUNT) OVER() AS overall_average
34     FROM `amino-retail1-04891.sales_retail`
35   )
36   SELECT * FROM CalculatedTable
37   WHERE TOTAL_AMOUNT > overall_average;
```

The results table has columns: Row, Transaction ID, Date, Customer ID, Gender, Age, Product Category, Quantity, Price per Unit, Total Amount, overall_average. The data shows 8 transactions from 2023 with their total amount compared to the average total amount of 456.0.

Row	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount	overall_average
1	21	2023-01-14	CUST021	Female	50	Beauty	1	500	500	456.0
2	28	2023-04-23	CUST028	Female	43	Beauty	1	500	500	456.0
3	128	2023-07-05	CUST128	Male	25	Beauty	1	500	500	456.0
4	220	2023-03-08	CUST220	Male	64	Beauty	1	500	500	456.0
5	230	2023-01-17	CUST230	Female	39	Beauty	1	500	500	456.0
6	364	2023-08-29	CUST364	Female	19	Beauty	1	500	500	456.0
7	408	2023-04-15	CUST408	Female	64	Beauty	1	500	500	456.0
8	537	2023-06-09	CUST537	Female	21	Beauty	1	500	500	456.0

Q3. Calculate the total revenue (sum of Total Amount).

Expected output: Total_Revenue

The screenshot shows a data processing interface with multiple tabs at the top: Untitled query, sales, retail, and *Untitled_query. The Untitled query tab is active. The query code is as follows:

```
42
43
44
45
46 --Q3: Calculate the total revenue (sum of Total Amount).
47
48 select sum('total amount') as 'total revenue'
49
50 FROM
51     'inno-retail-94891.sales.retail' ;
```

A note below the code states: "This script will process 325.74 KB when run." and "Using on-demand processing quota".

The "Results" tab is selected in the "Query results" section. The result table has one row:

Row	total revenue
1	456000

At the bottom right, it says "Results per page: 50 ▾ 1 – 1 of 1 | < < > > |".

Q4. Display all distinct Product Categories in the dataset.

Expected output: Product_Category

The screenshot shows a data processing interface with multiple tabs at the top: Untitled query, sales, retail, and *Untitled_query. The Untitled query tab is active. The query code is as follows:

```
46
47
48
49
50
51
52
53
54
55
56 --Q4: Display all distinct Product Categories in the dataset.
57
58 select distinct 'product category'
59
60
61 FROM
62     'inno-retail-94891.sales.retail' ;
```

A note below the code states: "Using on-demand processing quota".

The "Results" tab is selected in the "Query results" section. The result table has three rows:

Row	product category
1	Beauty
2	Clothing
3	Electronics

At the bottom right, it says "Results per page: 50 ▾ 1 – 3 of 3 | < < > > |".

Q5. For each Product Category, calculate the total quantity sold.
Expected output: Product Category, Total Quantity

Untitled query

Run Save Download Share Schedule Open in More

```
65
66
67
68
69
70 --Q5. For each Product Category, calculate the total quantity sold.
71
72
73 select product category','
74      | sum(Quantity) as 'Quantity sold'
75  FROM
76    'imo_retail-04891.sales.retail'
77
78 group by product category';
79
80
81
82
83
84
85 --Q6. Create a column called Age_Group that classifies customers as 'Youth' (<30), 'Adult' (30-59), and 'Senior' (60+).
86
87
88
89
90
91
92
93
94
95
96
97
98
99
```

This script will process 325.74 KB when run.

Query results

Job information Results Visualisation JSON Execution details Execution graph

Save results Open in

Row	product category	Quantity sold
1	Beauty	771
2	Clothing	894
3	Electronics	849

Results per page: 50 ▾ 1 – 3 of 3 | < < > >|

Q6. Create a column called Age_Group that classifies customers as ‘Youth’ (<30), ‘Adult’ (30–59), and ‘Senior’ (60+).

Expected output: Customer_ID, Age, Age_Group

Untitled query

Run Save Download Share Schedule Open in More

```
84
85 --Q6. Create a column called Age_Group that classifies customers as 'Youth' (<30), 'Adult' (30-59), and 'Senior' (60+).
86
87 select case
88     when Age < 30 then 'Youth(<30)'
89     when age between 30 and 59 then 'Adults(30-59)'
90     else 'Senior(60+)' 
91 end as 'Age Group'
92 count(*) as customer
93 FROM
94     'ino_retail-94891.sales.retail'
95
96 group by case
97     when Age < 30 then 'Youth(<30)'
98     when age between 30 and 59 then 'Adults(30-59)'
99     else 'Senior(60+)' 
100
101
102
103
```

This script will process 325.74 KB when run.

Using on-demand processing quote

Query results

Job Information Results Visualisation JSON Execution details Execution graph

Save results Open in

Row	Age Group	customer
1	Senior(60+)	115
2	Adults(30-59)	434
3	Youth(<30)	261

Results per page: 50 ▾ 1 – 3 of 3 {< < > >}

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

Expected output: Gender, High_Value_Transactions

The screenshot shows a data processing interface with a code editor at the top containing the following SQL query:

```
-- Q7. For each Gender, count how many high-value transactions occurred (where Total Amount > 500).
select gender,
       count(*) as high_value_transactions
FROM
  `inno-retail-94891.sales.retail`
where `total amount` > 500
group by gender;
```

A note below the code states: "This script will process 325.74 KB when run." and "Using on-demand processing quota".

The "Results" tab is selected in the interface, displaying the following table:

Row	gender	high_value_transactions
1	Female	165
2	Male	144

At the bottom right, there are buttons for "Save results" and "Open in". Below the table, it says "Results per page: 50" and "1 - 2 of 2".

Q8. For each Product Category, show only those categories where the total revenue exceeds 5,000.

Expected output: Product_Category, Total_Revenue

The screenshot shows a data processing interface with a code editor at the top containing the following SQL query:

```
-- Q8. For each Product Category, show only those categories where the total revenue exceeds 5,000
select 'product category',
       sum(`total amount`) as revenue
FROM
  `inno-retail-94891.sales.retail`
group by 'product category'
having sum(`total amount`) > 5000;
```

A note below the code states: "This script will process 325.74 KB when run." and "Using on-demand processing quota".

The "Results" tab is selected in the interface, displaying the following table:

Row	product category	revenue
1	Beauty	143615
2	Clothing	155280
3	Electronics	166965

At the bottom right, there are buttons for "Save results" and "Open in". Below the table, it says "Results per page: 50" and "1 - 3 of 3".

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

The screenshot shows a Snowflake query editor with the following details:

- Query:** Untitled query
- Script Content:**

```
140 -- Q9. Display a new column called Unit_Cost_Category that labels a transaction as:
141
142 select case
143     when 'price per unit' < 50 then 'cheap(<50)'
144     when 'price per unit' between 50 and 200 then 'Moderate(50-200)'
145     else 'expensive (>200)'
146 end as Unit_Cost_Category,
147 count(transaction_id) as 'total transaction',
148 'price per unit'
149
150 FROM
151     'lmao-retail-94891.sales.retail'
152
153 group by case
154     when 'price per unit' < 50 then 'cheap(<50)'
155     when 'price per unit' between 50 and 200 then 'Moderate(50-200)'
156     else 'expensive (>200)'
157
158 , 'price per unit';
159
```
- Message Bar:** This script will process 325.74 KB when run.
- Results Tab:** Shows the results of the query:

Row	Unit_Cost_Category	total transaction	price per unit
1	cheap(<50)	210	25
2	cheap(<50)	180	30
3	Moderate(50-200)	211	50
4	expensive (>200)	197	300
5	expensive (>200)	199	600

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

The screenshot shows a Snowflake query editor with the following details:

- Query:** Untitled query
- Script Content:**

```
159
160
161
162
163
164
165
166 -- Q10. Display all transactions from customers aged 40 or older and add a column Spending_Level showing 'High' if Total Amount > 1000, otherwise 'Low'.
167
168 select case
169     when 'total amount' > 1000 then 'high (>1000)'
170     else 'Low(<=1000)'
171 end as 'Spending Level',
172 count() as 'transactions'
173
174 FROM
175     'lmao-retail-94891.sales.retail'
176 where age > 40
177 group by case
178     when 'total amount' > 1000 then 'high (>1000)'
179     else 'Low(<=1000)'
180
181
```
- Message Bar:** This script will process 325.74 KB when run.
- Results Tab:** Shows the results of the query:

Row	Spending Level	transactions
1	Low(<=1000)	660
2	high (>1000)	58