Practical 5 exercise (Google Big Query)

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

```sql
---Q1. Filter all transactions that occurred in the year 2023. Expected output: All columns
SELECT *
FROM `airy-gate-478118-h4.Sales.Data`
WHERE EXTRACT(YEAR FROM DATE(Date)) = 2023;
```

Query completed

Using on-demand processing quota

Query results

Save results | Open in

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

Row	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity
1	191	2023-10-18	CUST191	Male	64	Beauty	
2	204	2023-09-28	CUST204	Male	39	Beauty	
3	230	2023-04-23	CUST230	Male	54	Beauty	
4	232	2023-02-06	CUST232	Female	43	Beauty	
5	309	2023-12-23	CUST309	Female	26	Beauty	
6	310	2023-10-12	CUST310	Female	28	Beauty	
7	363	2023-06-03	CUST363	Male	64	Beauty	
8	371	2023-02-21	CUST371	Female	20	Beauty	

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

```sql
---Q2. Display all transactions where the Total Amount is more than the average Total Amount of the entire dataset.
SELECT *
FROM `airy-gate-478118-h4.Sales.Data`
WHERE `Total Amount` > (
  SELECT AVG(`Total Amount`)
  FROM `airy-gate-478118-h4.Sales.Data`
  );
```

Query completed

Using on-demand processing quota

Query results

Save results | Open in

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

Row	Transaction ID	Date	Customer ID	Gender	Age	Product Category	Quantity	Price per Unit	Total Amount
1	21	2023-01-14	CUST021	Female	50	Beauty	1	500	500
2	28	2023-04-23	CUST028	Female	43	Beauty	1	500	500
3	128	2023-07-05	CUST128	Male	25	Beauty	1	500	500
4	220	2023-03-03	CUST220	Male	64	Beauty	1	500	500
5	238	2023-01-17	CUST238	Female	39	Beauty	1	500	500

Results per page: 50 | 1 – 50 of 350

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

```sql
---Q3. Calculate the total revenue (sum of Total Amount).
SELECT SUM(`Total Amount`) AS Total_Revenue
FROM `airy-gate-478118-h4.Sales.Data`;
```

Query completed

Using on-demand processing quota

Query results

Job information | Results | Visualization | JSON | Exe

Row	Total_Revenue
1	456000

Q4. Display all distinct Product Categories in the dataset.

```
19
20  ---Q4. Display all distinct Product Categories in the dataset.
21  SELECT DISTINCT `Product Category` AS Product_Category
22  FROM `airy-gate-478118-h4.Sales.Data`
23
```

Query completed

Using on-demand processing quota

Query results

Job information | Results | Visualization | JSON | Execution d

Row	Product_Category
1	Beauty
2	Clothing
3	Electronics

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

```
24  ---Q5. For each Product Category, calculate the total quantity sold.
25  SELECT
26    `Product Category` AS Product_Category,
27    SUM(`Quantity`)    AS Total_Quantity
28  FROM `airy-gate-478118-h4.Sales.Data`
29  GROUP BY `Product Category`
30  ORDER BY `Product Category`;
```

Syntax error: Expected end of input but got keyword SELECT at [25:1]

Using on-demand processing quota

Query results

Job information | Results | Visualization | JSON | Execution details

Row	Product_Category	Total_Quantity
1	Beauty	771
2	Clothing	894
3	Electronics	849

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

```sql
---Q6. Create a column called Age_Group that classifies customers as 'Youth' (<30), 'Adult' (30–59), and 'Senior' (60+).
SELECT
  `Customer ID` AS Customer_ID,
  `Age`,
  CASE
    WHEN Age < 30 THEN 'Youth'
    WHEN Age BETWEEN 30 AND 59 THEN 'Adult'
    WHEN Age >= 60 THEN 'Senior'
    ELSE 'Unknown'
  END AS Age_Group
  FROM `airy-gate-478118-h4.Sales.Data`;
```

Syntax error: Expected end of input but got keyword SELECT at [25:1]

Using on-demand processing quota

Query results

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

Row	Customer_ID	Age	Age_Group
1	CUST191	64	Senior
2	CUST204	39	Adult
3	CUST230	54	Adult
4	CUST232	43	Adult
5	CUST309	26	Youth

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

```sql
  ---Q7. For each Gender, count how many high-value transactions occurred (where Total Amount > 500).
SELECT
  Gender,
  COUNTIF(`Total Amount` > 500) AS High_Value_Transactions
FROM `airy-gate-478118-h4.Sales.Data`
GROUP BY Gender
ORDER BY Gender;
```

Query completed

Using on-demand processing quota

Query results

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

Row	Gender	High_Value_Tran...
1	Female	155
2	Male	144

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

```sql
---Q8. For each Product Category, show only those categories where the total revenue exceeds 5,000.
SELECT
  `Product Category` AS Product_Category,
  SUM(`Total Amount`) AS Total_Revenue
FROM `airy-gate-478118-h4.Sales.Data`
GROUP BY `Product Category`
HAVING SUM(`Total Amount`) > 5000;
```

Query completed

Using on-demand processing quota

Query results

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

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

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

```sql
---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
SELECT
  `Transaction ID` AS Transaction_ID,
  `Price per Unit` AS Price_per_Unit,
  CASE
    WHEN `Price per Unit` < 50 THEN 'Cheap'
    WHEN `Price per Unit` BETWEEN 50 AND 200 THEN 'Moderate'
    WHEN `Price per Unit` > 200 THEN 'Expensive'
    ELSE 'Unspecified'
  END AS Unit_Cost_Category
FROM `airy-gate-478118-h4.Sales.Data`;
```

Query completed

Using on-demand processing quota

Query results

Save results | Open in

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

Row	Transaction_ID	Price_per_Unit	Unit_Cost_Category
1	191	25	Cheap
2	204	25	Cheap
3	230	25	Cheap
4	232	25	Cheap
5	309	25	Cheap

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

```sql
---Q10. Display all transactions from customers aged 40 or older and add a column Spending_Level showing 'High' if Total Amount > 1000, otherwise 'Low'.
SELECT
  `Customer ID` AS Customer_ID,
  Age,
  `Total Amount` AS Total_Amount,
  CASE
    WHEN `Total Amount` > 1000 THEN 'High'
    ELSE 'Low'
  END AS Spending_Level
FROM `airy-gate-478118-h4.Sales.Data`
WHERE Age >= 40;
```

Query completed

Using on-demand processing quota

Query results

Save

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

Row	Customer_ID	Age	Total_Amount	Spending_Level
1	CUST191	64	25	Low
2	CUST230	54	25	Low
3	CUST232	43	25	Low
4	CUST363	64	25	Low
5	CUST454	46	25	Low