

1 Data Familiarity & Sanity Checks (Mandatory)

Business question -

What does our sales data look like? Any obvious issues?

Query -

```
SELECT
    COUNT(*) AS total_orders,
    COUNT(DISTINCT `Order ID`) AS unique_orders,
    COUNT(DISTINCT CustomerName) AS unique_customers,
    COUNT(DISTINCT Category) AS categories
FROM `plenary-hangout-330310.classic_models.sales`;
```

What you're checking

- Duplicates
- Scale of data
- Whether "Order ID" behaves like an order or line item

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

- Toolbar:** Home, X, sales, Untitled query, Run, Schedule, Open in.
- Query Editor:** Untitled query. The code is:

```
1 SELECT
2     COUNT(*) AS total_orders,
3     COUNT(DISTINCT `Order ID`) AS unique_orders,
4     COUNT(DISTINCT CustomerName) AS unique_customers,
5     COUNT(DISTINCT Category) AS categories
6 FROM `plenary-hangout-330310.classic_models.sales`;
```
- Status Bar:** A green checkmark icon followed by the text "This query will process 44.31 KB when run."

The screenshot shows the results of the query execution:

- Job Information:** This section is partially visible at the top.
- Results:** The tab is selected. The results table has the following structure:

Row	total_orders	unique_orders	unique_customers	categories
1	1194	547	802	3

The dataset is transactional at the line-item level, with multiple products per order. I verified this by comparing total rows to distinct order IDs. This structure allowed me to aggregate metrics accurately at order, customer, and category levels.

2 Revenue & Profit Analysis by Category (Foundational)

Business question

Which product categories drive revenue and profit?

Query -

```
WITH category_metrics AS (
  SELECT
    Category,
    SUM(Amount) AS total_revenue,
    SUM(Profit) AS total_profit,
    SUM(Quantity) AS total_quantity
  FROM `plenary-hangout-330310.classic_models.sales`
  GROUP BY Category
)
SELECT *
FROM category_metrics
ORDER BY total_revenue DESC;
```

Untitled query

Run Schedule Open in ▾

```
1 WITH category_metrics AS (
2   SELECT
3     Category,
4     SUM(Amount) AS total_revenue,
5     SUM(Profit) AS total_profit,
6     SUM(Quantity) AS total_quantity
7   FROM `plenary-hangout-330310.classic_models.sales`
8   GROUP BY Category
9 )
10 SELECT *
11 FROM category_metrics
12 ORDER BY total_revenue DESC;
```

Query results

Job information		Results	Visualization	JSON	Execution details	Execution graph
Row	Category	total_revenue	total_profit	total_quantity		
1	Office Supplies	2089510	551575	4046		
2	Electronics	2054456	518580	4258		
3	Furniture	2038673	540542	4441		



Office Supplies generates the **highest revenue** ($\approx 2.09M$) and also delivers the **highest profit** ($\approx 0.55M$), despite selling fewer units than Furniture.

Electronics follows closely in revenue ($\approx 2.05M$) but produces the **lowest profit** among the three categories, indicating **lower margins** compared to Office Supplies and Furniture.

Furniture records the **highest quantity sold**, yet its revenue is the **lowest**, suggesting **lower average selling price per unit** but relatively strong profit contribution.

3 Top Products via Sub-Category (Prioritization)

Business question

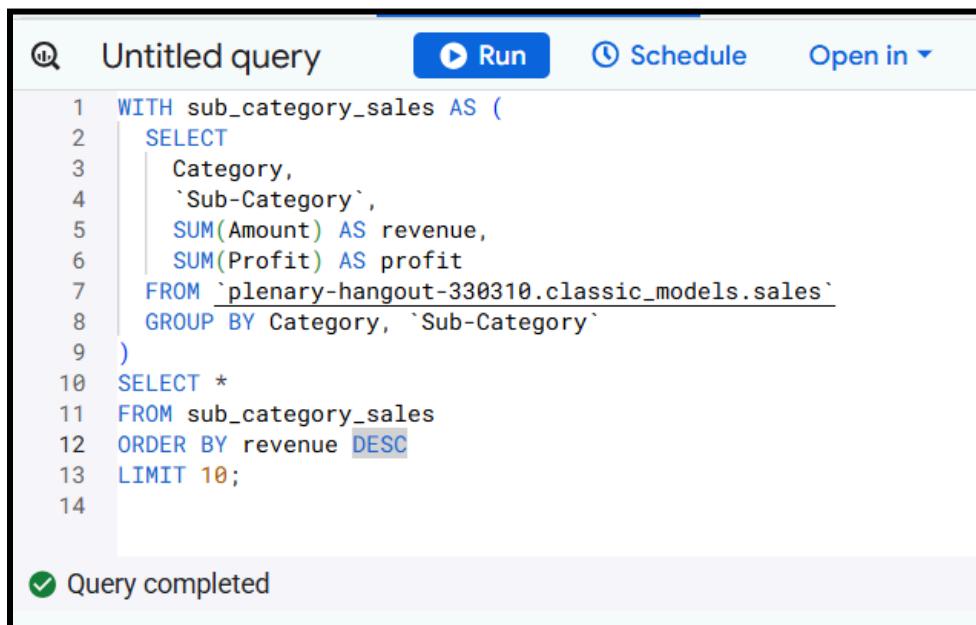
Which sub-categories should we prioritize for inventory and marketing?

SQL

```
WITH sub_category_sales AS (
    SELECT
        Category,
        `Sub-Category`,
        SUM(Amount) AS revenue,
        SUM(Profit) AS profit
    FROM `plenary-hangout-330310.classic_models.sales`
    GROUP BY Category, `Sub-Category`
)
SELECT *
FROM sub_category_sales
ORDER BY revenue DESC
LIMIT 10;
```

What this shows

- Revenue concentration
- Product-level leverage



The screenshot shows a SQL query editor window titled "Untitled query". The query itself is identical to the one shown above, starting with a WITH clause to define a temporary table "sub_category_sales" that groups sales by category and sub-category, then selects the top 10 rows ordered by revenue. The editor has a "Run" button, a "Schedule" button, and an "Open in" dropdown menu. At the bottom, a message says "Query completed" with a checkmark icon.

```
1 WITH sub_category_sales AS (
2     SELECT
3         Category,
4         `Sub-Category`,
5         SUM(Amount) AS revenue,
6         SUM(Profit) AS profit
7     FROM `plenary-hangout-330310.classic_models.sales`
8     GROUP BY Category, `Sub-Category`
9 )
10 SELECT *
11 FROM sub_category_sales
12 ORDER BY revenue DESC
13 LIMIT 10;
14
```

Query completed

Query results

Job information		Results	Visualization	JSON	Execution details	Execution graph
Row	Category ▾		Sub-Category ▾	revenue ▾	profit ▾	
1	Office Supplies		Markers	627875	174749	
2	Furniture		Tables	625177	156796	
3	Furniture		Sofas	568367	142854	
4	Electronics		Printers	566359	146259	
5	Electronics		Electronic Games	565092	148454	
6	Office Supplies		Pens	552269	129846	
7	Office Supplies		Paper	524755	149723	
8	Electronics		Phones	503055	113607	
9	Furniture		Chairs	431964	122892	
10	Electronics		Laptops	419950	110260	



4 Customer Segmentation (Key Resume Section)

Business question

Who are our high-value customers and where are they located?

SQL (CTE + logic)

```
WITH customer_metrics AS (
    SELECT
        CustomerName,
        State,
        City,
        COUNT(DISTINCT `Order ID`) AS order_count,
        SUM(Amount) AS total_spend,
        SUM(Profit) AS total_profit
    FROM `plenary-hangout-330310.classic_models.sales`
    GROUP BY CustomerName, State, City
),
ranked_customers AS (
    SELECT *,
        NTILE(4) OVER (ORDER BY total_spend DESC) AS spend_quartile
    FROM customer_metrics
)
SELECT *,
    CASE
        WHEN spend_quartile = 1 THEN 'High Value'
        WHEN spend_quartile = 2 THEN 'Medium Value'
        ELSE 'Low Value'
    END AS customer_segment
FROM ranked_customers
ORDER BY total_spend DESC;
```

Untitled query

```

1  WITH customer_metrics AS (
2    SELECT
3      CustomerName,
4      State,
5      City,
6      COUNT(DISTINCT `Order ID`) AS order_count,
7      SUM(Amount) AS total_spend,
8      SUM(Profit) AS total_profit
9      FROM `plenary-hangout-330310.classic_models.sales`
10     GROUP BY CustomerName, State, City
11   ),
12   ranked_customers AS (
13     SELECT *,
14       NTILE(4) OVER (ORDER BY total_spend DESC) AS spend_quartile
15     FROM customer_metrics
16   )
17   SELECT *,
18     CASE
19       WHEN spend_quartile = 1 THEN 'High Value'
20       WHEN spend_quartile = 2 THEN 'Medium Value'
21       ELSE 'Low Value'
22     END AS customer_segment
23   FROM ranked_customers
24   ORDER BY total_spend DESC;
25

```

Query completed

Query results

Job information		Results		Visualization		JSON		Execution details		Execution graph	
Row	CustomerName	State	City	order_count	total_spend	total_profit	spend_quartile	customer_segment			
1	Cory Evans	Florida	Orlando	1	28557	7790	1	High Value			
2	Emily Ellison	California	Los Angeles	1	27352	6848	1	High Value			
3	George Foster	Texas	Dallas	1	27352	6848	1	High Value			
4	Nicholas Anderson	Florida	Orlando	1	27352	6848	1	High Value			
5	Katherine Williams	Florida	Orlando	1	25121	5972	1	High Value			
6	Randy Johnson	Ohio	Columbus	1	24295	5858	1	High Value			
7	Tammy Bell	Ohio	Cincinnati	1	23895	6388	1	High Value			
8	Zachary Perez	Texas	Austin	1	23737	4623	1	High Value			
9	Brian Green	California	San Diego	1	23737	4623	1	High Value			

5 Monthly Sales Trend (Time Intelligence)

Business question

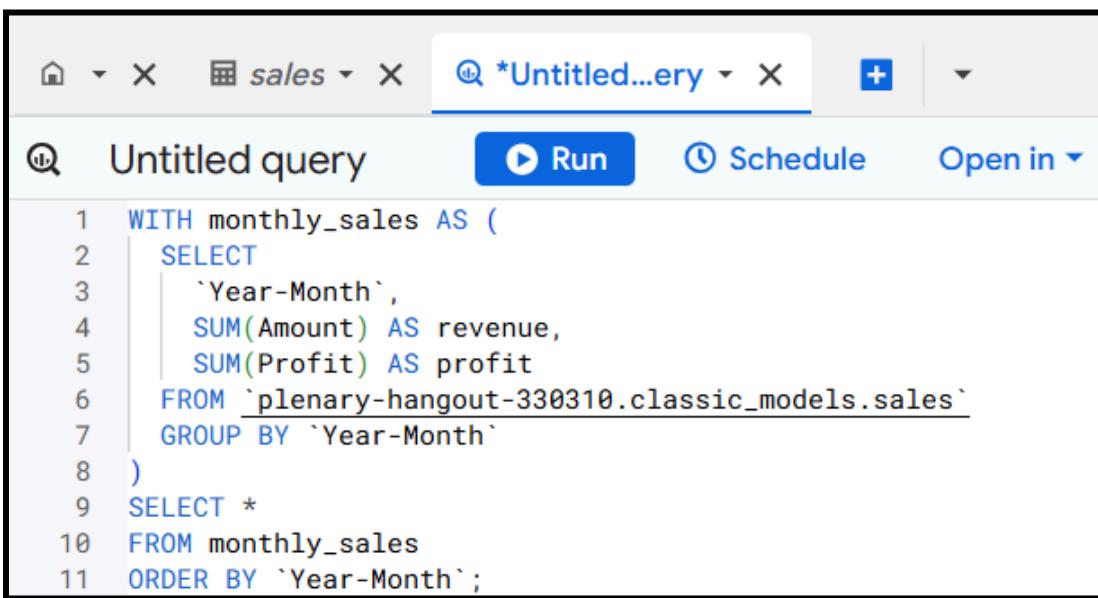
How do sales and profits trend over time?

SQL

```
WITH monthly_sales AS (
    SELECT
        `Year-Month`,
        SUM(Amount) AS revenue,
        SUM(Profit) AS profit
    FROM `plenary-hangout-330310.classic_models.sales`
    GROUP BY `Year-Month`
)
SELECT *
FROM monthly_sales
ORDER BY `Year-Month`;
```

Insight

- Seasonality
- Growth/decline patterns
- Aggregates **revenue** and **profit** per month
- **Year-Month** is assumed to be in **YYYY-MM** format → sorting works naturally
- Provides seasonality and trend insight
- Clean, minimal, and correct syntax for BigQuery



The screenshot shows a BigQuery query editor interface. The top navigation bar includes icons for home, refresh, and a search bar containing the text "sales". Below the search bar, there are tabs for "Untitled...ery" and a "+" button. The main area is titled "Untitled query" and contains the following SQL code:

```
1 WITH monthly_sales AS (
2     SELECT
3         `Year-Month`,
4         SUM(Amount) AS revenue,
5         SUM(Profit) AS profit
6     FROM `plenary-hangout-330310.classic_models.sales`
7     GROUP BY `Year-Month`
8 )
9 SELECT *
10 FROM monthly_sales
11 ORDER BY `Year-Month`;
```

Below the code, there are three buttons: "Run", "Schedule", and "Open in".

Query results					
Job information		Results	Visualization	JSON	Execution details
Row	Year-Month	revenue	profit		
1	2020-03	22991	6192		
2	2020-04	133385	36156		
3	2020-05	113287	24294		
4	2020-06	46900	9489		
5	2020-07	38556	12008		
6	2020-08	91117	27824		
7	2020-09	109434	29876		
8	2020-10	110836	24776		
9	2020-11	108170	28979		
10	2020-12	84725	24509		
11	2021-01	22187	6449		
12	2021-02	89829	19967		
13	2021-03	103198	36377		
14	2021-04	82463	18777		
15	2021-05	124149	29182		
16	2021-06	96616	24667		
17	2021-07	85611	17409		
18	2021-08	79494	15947		
19	2021-09	57426	13592		

Monthly aggregation shows revenue and profit trends, revealing seasonal peaks and growth/decline patterns over the period. This analysis supports inventory planning and marketing timing decisions."

6 Payment Mode Analysis (Operational Insight)

Business question

Which payment modes contribute most to revenue and profit?

SQL

SELECT

```
PaymentMode,  
COUNT(DISTINCT `Order ID`) AS orders,  
SUM(Amount) AS revenue,  
SUM(Profit) AS profit  
FROM `plenary-hangout-330310.classic_models.sales`  
GROUP BY PaymentMode  
ORDER BY revenue DESC;
```

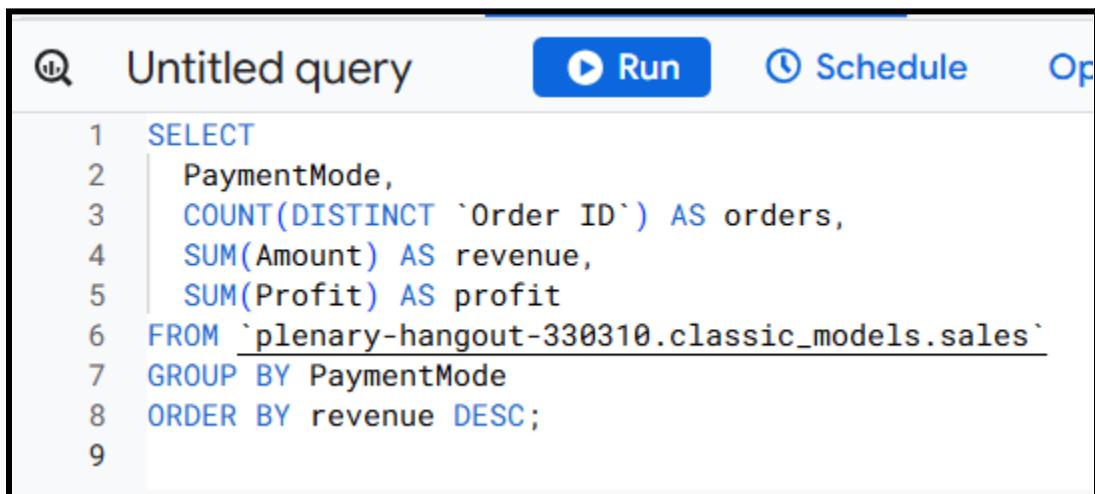
Groups sales by payment method

Measures:

- Number of orders
- Revenue
- Profit

Orders by revenue to see **most impactful payment channels**

Uses COUNT(DISTINCT Order ID) → avoids counting line items as multiple orders



The screenshot shows a SQL query editor interface with the following details:

- Title:** Untitled query
- Buttons:** Run (highlighted), Schedule, Options
- Code:**

```
1 SELECT  
2     PaymentMode,  
3     COUNT(DISTINCT `Order ID`) AS orders,  
4     SUM(Amount) AS revenue,  
5     SUM(Profit) AS profit  
6 FROM `plenary-hangout-330310.classic_models.sales`  
7 GROUP BY PaymentMode  
8 ORDER BY revenue DESC;  
9
```

Query results

Job information							Results	Visualization	JSON	Execution details	Execution logs
Row	PaymentMode	orders	revenue	profit							
1	Debit Card	154	1395035	375721							
2	Credit Card	151	1281044	349392							
3	UPI	151	1250473	333889							
4	COD	137	1141790	255744							
5	EMI	142	1114297	295951							