

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. At the top, there are tabs for 'sales' and '*Untitled...ery'. Below the tabs, there's a search bar and buttons for 'Run', 'Schedule', and 'Open in'. The main area contains a SQL query with line numbers 1 through 7. Below the query, there's a green checkmark icon and a message: 'This query will process 44.31 KB when run.'

```
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`;
7
```

✓ This query will process 44.31 KB when run.

The screenshot shows the 'Query results' section of the interface. It has a tabbed header with 'Job information', 'Results' (selected), 'Visualization', 'JSON', 'Execution details', and 'Execution graph'. Below the header is a table with 5 columns: 'Row', 'total_orders', 'unique_orders', 'unique_customers', and 'categories'. The first row shows the results of the query.

Job information	Results	Visualization	JSON	Execution details	Execution graph
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.

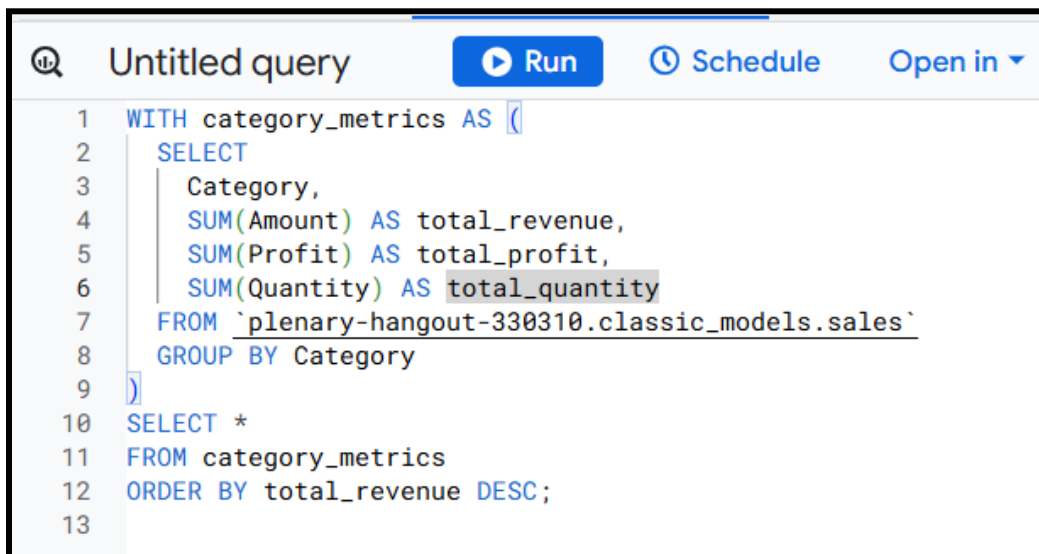
2 Revenue & Profit Analysis by Category (Foundational)

Business question

Which product categories drive revenue and profit?

Query -

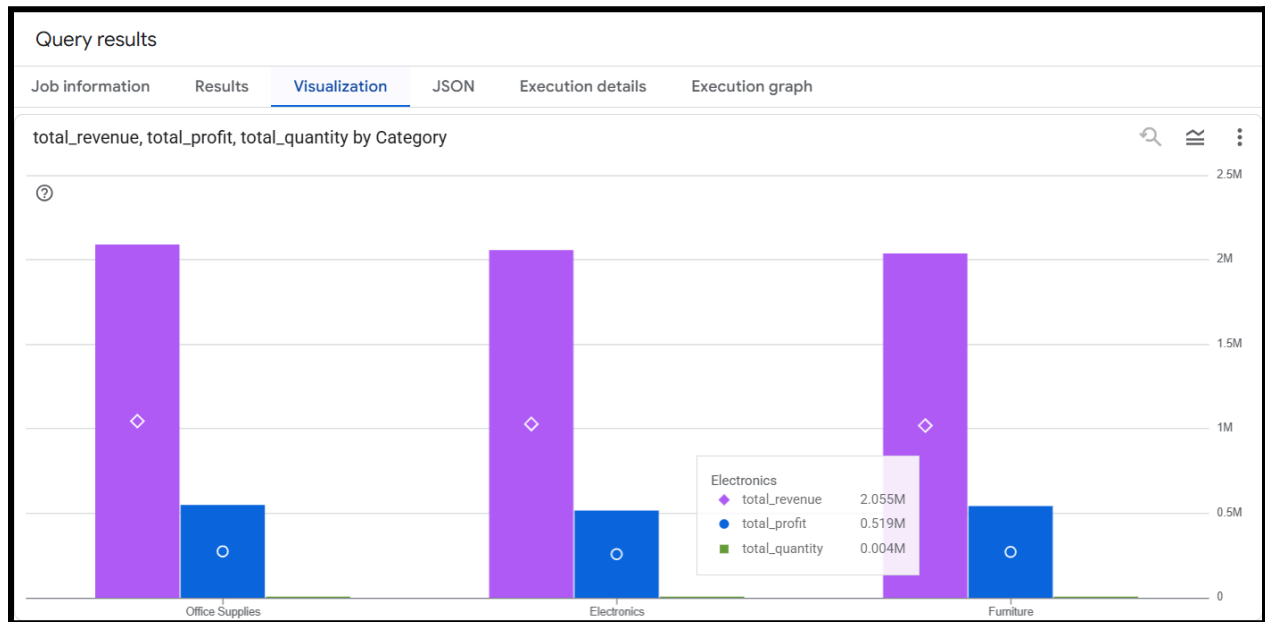
```
WITH category_metrics AS (  
  SELECT  
    Category,  
    SUM(Amount) AS total_revenue,  
    SUM(Profit) AS total_profit,  
    SUM(Quantity) AS total_quantity  
  FROM `plenary-hangout-330310.classic_models.sales`  
  GROUP BY Category  
)  
SELECT *  
FROM category_metrics  
ORDER BY total_revenue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

A screenshot of a SQL query editor interface. At the top, there's a search icon, the text "Untitled query", and three buttons: "Run" (with a play icon), "Schedule" (with a clock icon), and "Open in" (with a dropdown arrow). Below the header, the SQL query is displayed in a monospaced font, with line numbers 1 through 13 on the left. The query is a Common Table Expression (CTE) named 'category_metrics' followed by a SELECT statement that joins the CTE and orders the results by total_revenue in descending order.

```
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;  
13
```

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** ($\approx 2.09\text{M}$) and also delivers the **highest profit** ($\approx 0.55\text{M}$), despite selling fewer units than Furniture.

Electronics follows closely in revenue ($\approx 2.05\text{M}$) 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

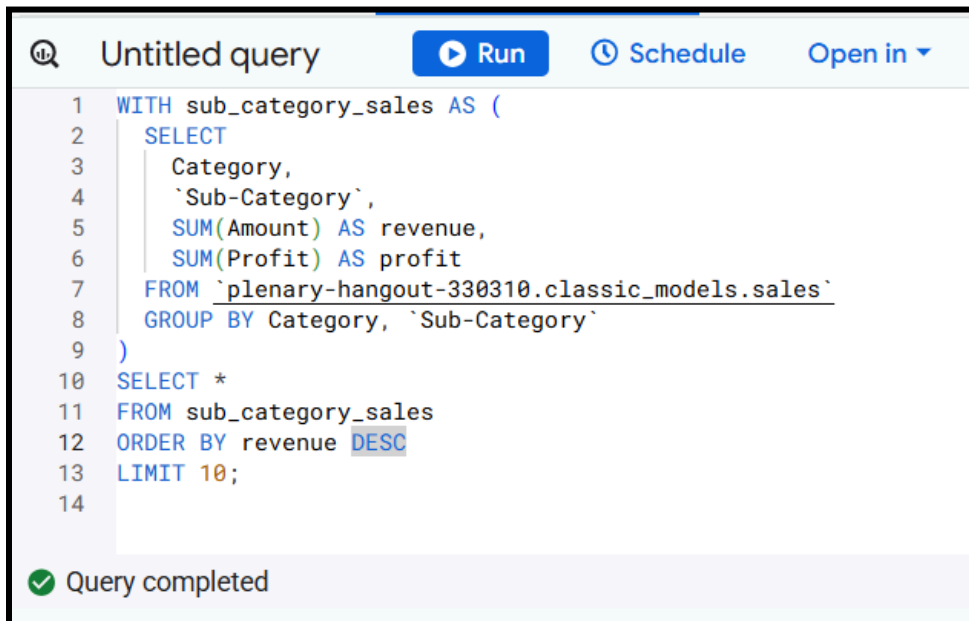
Which sub-categories should we prioritize for inventory and marketing?

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



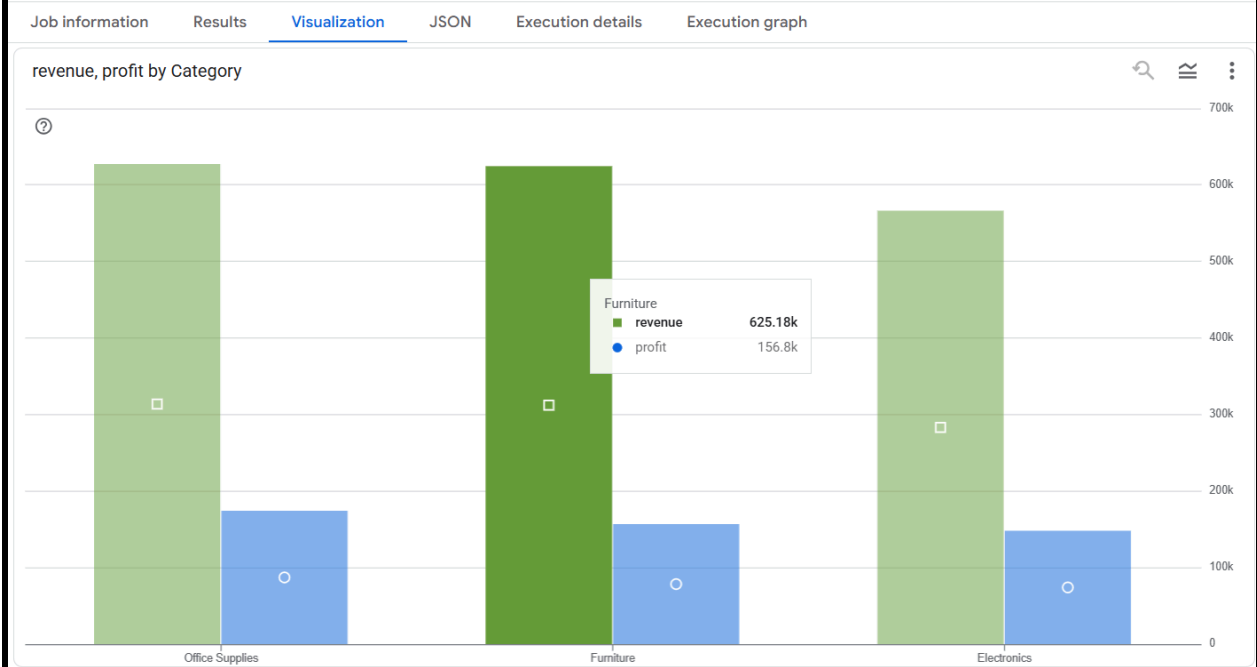
```
1 WITH sub_category_sales AS (  
2   SELECT  
3     Category,  
4     `Sub-Category`,  
5     SUM(Amount) AS revenue,  
6     SUM(Profit) AS profit  
7   FROM `plenary-hangout-330310.classic_models.sales`  
8   GROUP BY Category, `Sub-Category`  
9 )  
10 SELECT *  
11 FROM sub_category_sales  
12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

Query results



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;
```


5 Monthly Sales Trend (Time Intelligence)

Business question

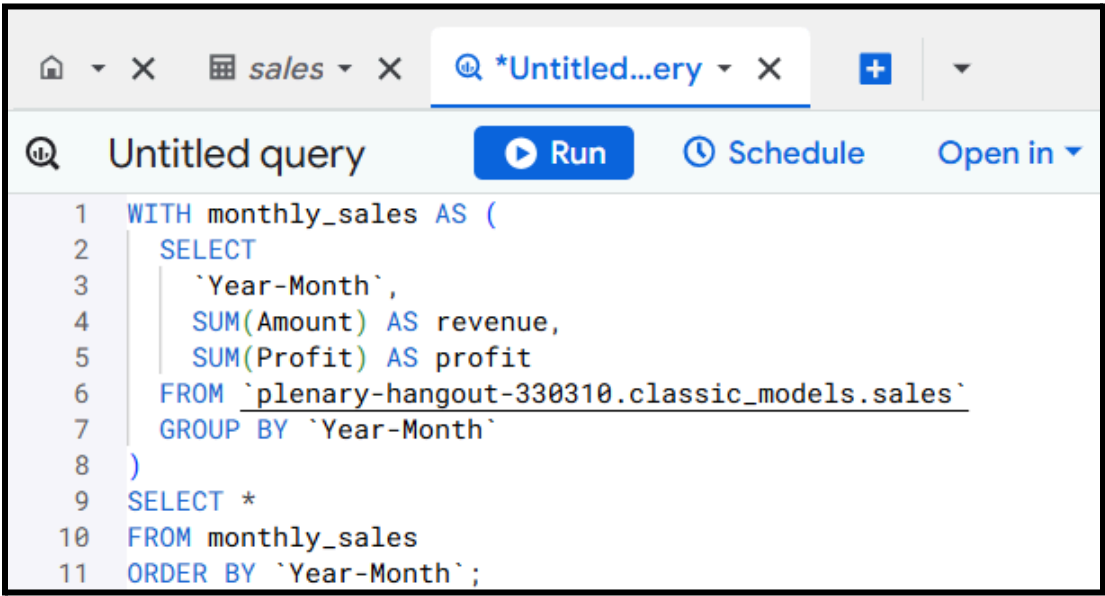
How do sales and profits trend over time?

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

A screenshot of a web-based SQL editor interface, likely Google BigQuery. The top bar shows a search icon, a dropdown menu with 'sales', and a tab titled '*Untitled...ery'. Below this, the editor area has a search icon, the text 'Untitled query', and three buttons: 'Run' (with a play icon), 'Schedule' (with a clock icon), and 'Open in' (with a dropdown arrow). The SQL query is displayed in a monospaced font with syntax highlighting. Line numbers 1 through 11 are visible on the left side of the query text.

```
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`;
```

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;
```

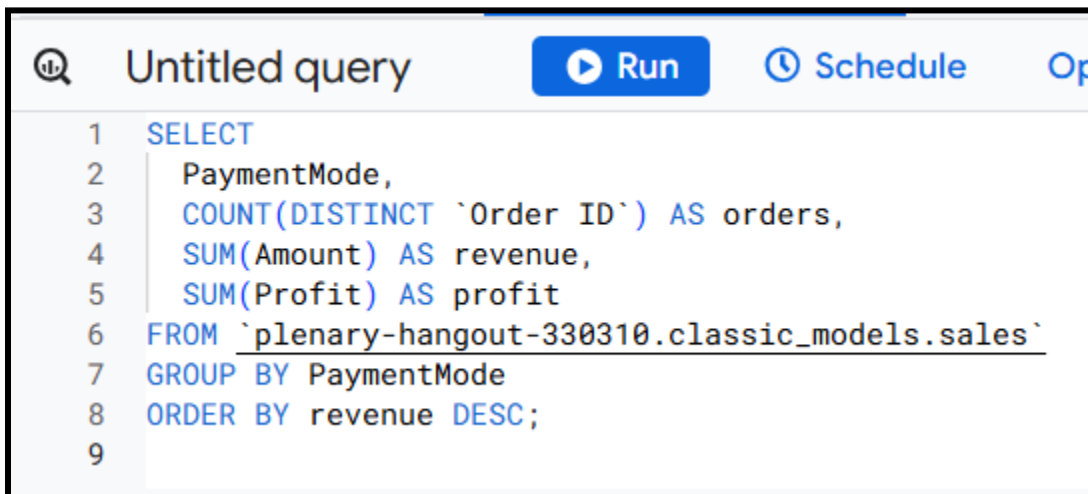
Groups sales by payment method

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

A screenshot of a SQL query editor interface. At the top, there is a search icon, the text "Untitled query", and two buttons: "Run" (with a play icon) and "Schedule" (with a clock icon). Below the header, the SQL query is displayed in a monospaced font, with line numbers 1 through 9 on the left. The query is:

```
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
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						