□ Project Title

Supermart Profit Prediction & Retail Business Analytics System

---

□ Business Problem

Retail businesses face challenges in predicting profit due to fluctuating sales, discount strategies, seasonal demand, and product-level variations. Business teams need a reliable system to **analyze historical performance** and **predict future profitability** before making pricing or promotional decisions.

□ Project Objective

The objectives of this project were:

- To analyze historical supermarket sales data
- To understand the relationship between sales, discounts, and profit
- To build a machine learning model for profit prediction
- To deploy an interactive application for real-time business decision support
- To demonstrate the use of **Excel, SQL, Python, and Machine Learning** in a single end-to-end project

---

□ Dataset Overview

The dataset contains historical grocery sales transactions with attributes such as:

- Order details (Order ID, Customer Name)
- Product information (Category, Sub-category)
- Location details (City, Region, State)
- Financial metrics (Sales, Discount, Profit)
- Time-related information (Order Date)

This dataset represents a realistic retail environment suitable for business analytics and predictive modeling.

□ TOOLS & TECHNOLOGIES USED

- **Excel** – Initial data understanding & validation
- **SQL** – Structured querying & business-level analysis
- **Python** – Data cleaning, EDA, feature engineering
- **Machine Learning (Scikit-learn)** – Model development
- **Streamlit** – Model deployment & dashboard creation

☐ Step 1: Excel Analysis (WHY & HOW)

☐ Why Excel was used

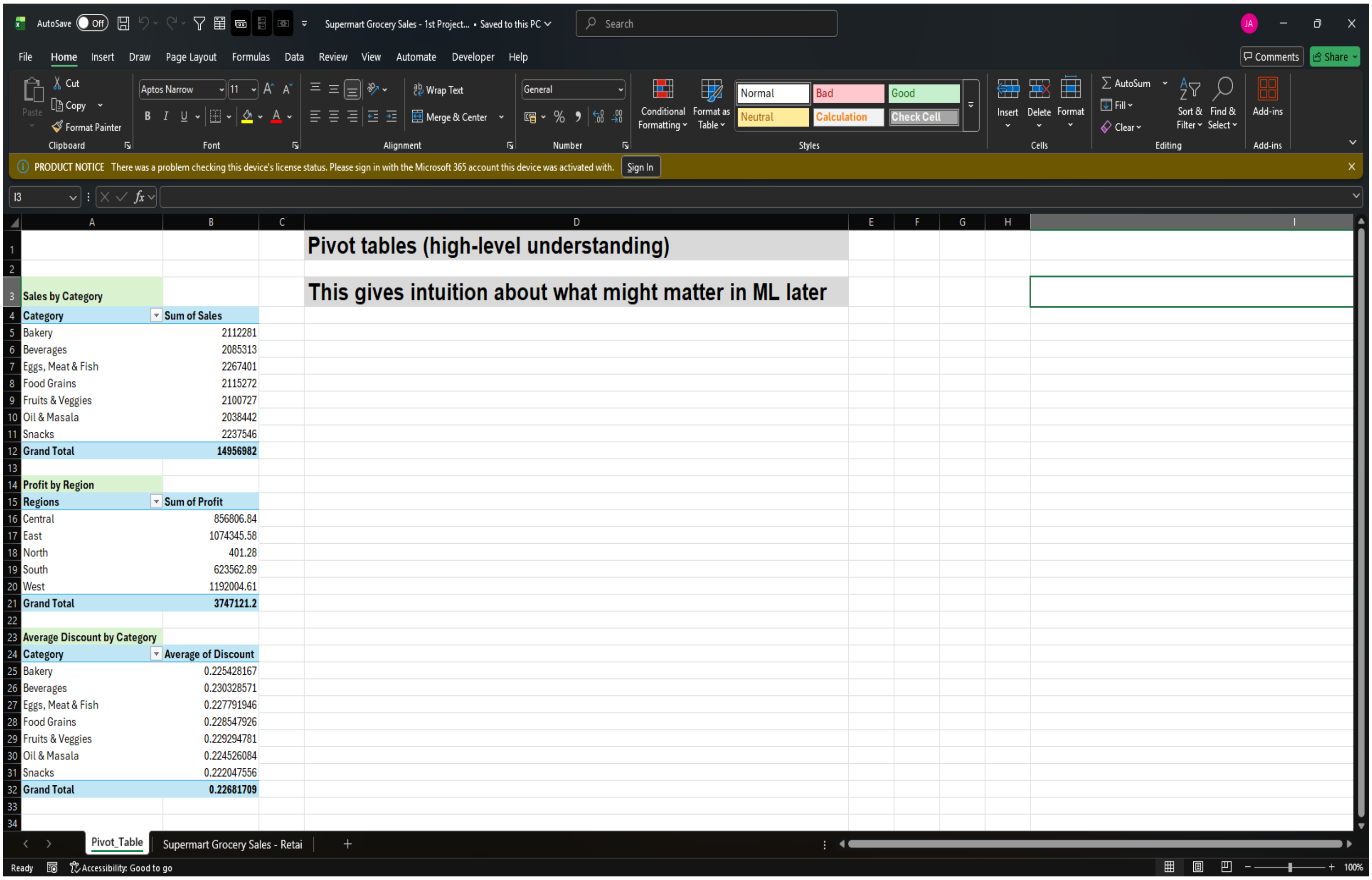
Excel was used as the **first-level analysis tool** to quickly understand the data before moving to advanced tools.

☐ What was done in Excel

- Reviewed raw data structure and column meanings
- Checked for missing values and incorrect data types
- Identified outliers in Sales, Discount, and Profit
- Created basic pivot tables:
  - Category-wise sales and profit
  - Region-wise performance
- Used charts to visually understand:
  - Sales vs Profit trends
  - Discount impact on profit

☐ Outcome:

Excel helped validate data quality and provided initial business insights that guided further analysis in SQL and Python.


AutoSave Off
Supermart Grocery Sales - 1st Project... • Saved to this PC
Search
File Home Insert Draw Page Layout Formulas Data Review View Automate Developer Help
Comments Share
Cut Copy Format Painter Paste Clipboard
Aptos Narrow 11 Font
Wrap Text Merge & Center Alignment
General Number
Conditional Formatting Format as Table
Normal Bad Good Neutral Calculation Check Cell Styles
Insert Delete Format Cells
AutoSum Fill Clear Sort & Filter Find & Select Editing
Add-ins
I3
Pivot_Table
Supermart Grocery Sales - Retai
Ready Accessibility: Good to go 100%

PRODUCT NOTICE There was a problem checking this device's license status. Please sign in with the Microsoft 365 account this device was activated with. Sign In

Pivot tables (high-level understanding)

This gives intuition about what might matter in ML later

Sales by Category

Category	Sum of Sales
Bakery	2112281
Beverages	2085313
Eggs, Meat & Fish	2267401
Food Grains	2115272
Fruits & Veggies	2100727
Oil & Masala	2038442
Snacks	2237546
Grand Total	**14956982**

Profit by Region

Regions	Sum of Profit
Central	856806.84
East	1074345.58
North	401.28
South	623562.89
West	1192004.61
Grand Total	**3747121.2**

Average Discount by Category

Category	Average of Discount
Bakery	0.225428167
Beverages	0.230328571
Eggs, Meat & Fish	0.227791946
Food Grains	0.228547926
Fruits & Veggies	0.229294781
Oil & Masala	0.224526084
Snacks	0.222047556
Grand Total	**0.22681709**

□ Step 2: SQL Analysis (WHY & HOW)

□ Why SQL was used

SQL was used to perform **structured, business-driven analysis** similar to how data is analyzed in real organizations using databases.

□ What was done using SQL

- Queried total sales and profit by:
  - Category
  - Sub-category
  - Region
- Identified top-performing and low-performing products
- Analyzed discount impact using grouped aggregations
- Validated business KPIs such as:
  - Average discount per region
  - Profit contribution by category

□ **Example business questions answered via SQL:**

- Which category generates the highest profit?
- Which region is most sensitive to discounts?
- How does profit vary across different product segments?

□ **Outcome:**
SQL helped transform raw data into **business-ready insights** and ensured the analysis aligned with real-world reporting practices.

□ Step 3: Data Cleaning & Preprocessing (Python)

Python was used for deeper data preparation:

- Cleaned and standardized column names
- Converted numerical and date fields to proper formats
- Removed irrelevant columns for modeling
- Handled encoding of categorical variables
- Prepared a clean dataset suitable for machine learning

□ Step 4: Exploratory Data Analysis (EDA)

EDA was performed using Python libraries:

- Analyzed distributions of sales, profit, and discount
- Studied relationships such as:
  - Discount vs Profit
  - Category vs Profit

- Identified seasonal patterns using year and month

This step confirmed many insights initially observed in Excel and SQL.

□ Step 5: Feature Engineering

Additional features were created to enhance model learning:

- Year, Month, Quarter
- Weekend indicator
- One-hot encoding for categorical variables
- Numeric-only feature selection for ML compatibility

□ Step 6: Machine Learning Modeling

Baseline Model: Linear Regression

- Used to establish a baseline performance
- Provided interpretability of linear relationships

Advanced Model: Random Forest Regressor

- Captured non-linear patterns
- Improved handling of complex feature interactions
- Provided better business-level insights

Models were evaluated using:

- Mean Absolute Error (MAE)
- R^2 Score

□ Step 7: Model Deployment using Streamlit

The trained Random Forest model was deployed using Streamlit Cloud:

- Users can input order details
- Real-time profit predictions are generated
- Business warnings and insights are displayed
- The app converts ML output into actionable insights

This step transformed the analysis into a **production-style business application**.

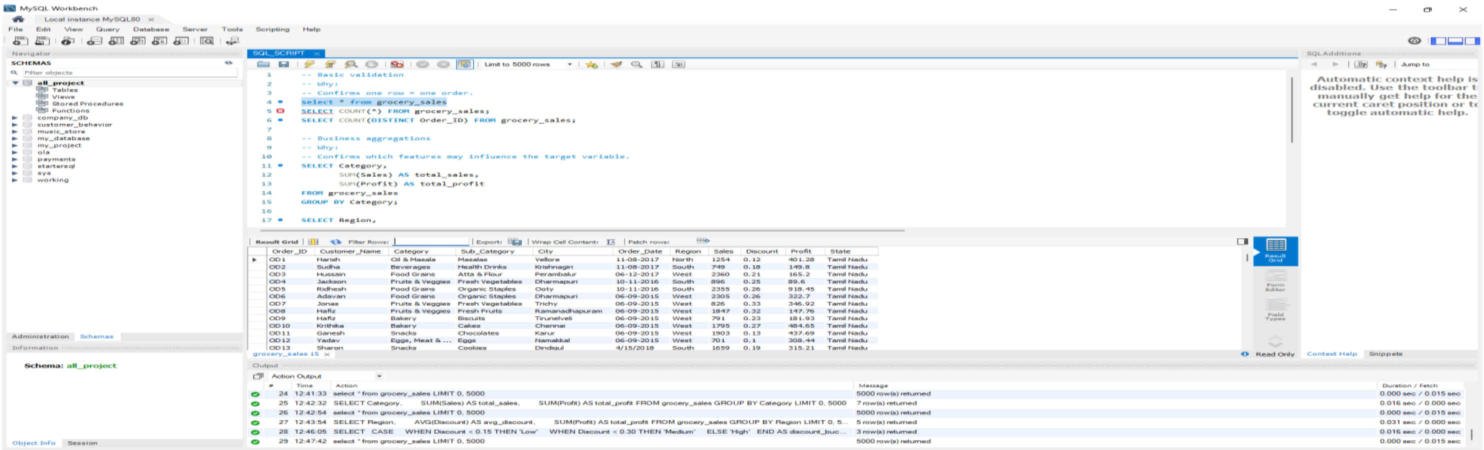
□ Business Impact & Insights

- High discounts negatively impact profitability
- Certain categories consistently perform better
- Profit prediction supports pricing and promotion decisions
- Scenario testing reduces financial risk

□ Final Conclusion

This project demonstrates a complete retail analytics solution by combining Excel, SQL, Python, Machine Learning, and Streamlit deployment. Excel and SQL were used for initial data validation and business-level analysis, while Python enabled advanced analytics and modeling. The final deployed application allows stakeholders to predict profit in real time and make informed business decisions. This project reflects real-world data science workflows and highlights the importance of combining technical skills with business understanding.

SQl Results :



```
-- Basic validation
-- Why:
-- Confirms one row = one order.
select * from grocery_sales
SELECT COUNT(*) FROM grocery_sales;
SELECT COUNT(DISTINCT Order_ID) FROM grocery_sales;

-- Business aggregations
-- Why:
-- Confirms which features may influence the target variable.
SELECT Category,
       SUM(Sales) AS total_sales,
       SUM(Profit) AS total_profit
FROM grocery_sales
GROUP BY Category;

SELECT Region,
```

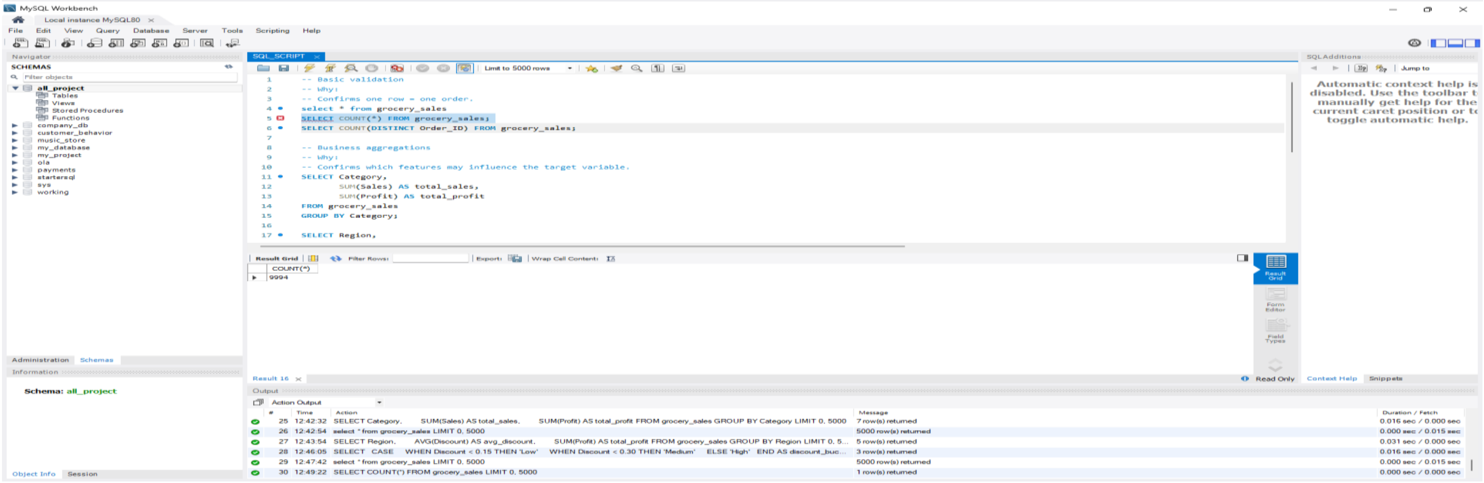
Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

Order_ID	Customer_Name	Category	Sub_Category	City	Order_Date	Region	Sales	Discount	Profit	State
OD1	Harish	Oil & Masala	Masalas	Vellore	11-08-2017	North	1254	0.12	401.28	Tamil Nadu
OD2	Sudha	Beverages	Health Drinks	Krishnagiri	11-08-2017	South	749	0.18	149.8	Tamil Nadu
OD3	Hussain	Food Grains	Atta & Flour	Perambalur	06-12-2017	West	2360	0.21	165.2	Tamil Nadu
OD4	Jackson	Fruits & Veggies	Fresh Vegetables	Dharmapuri	10-11-2016	South	896	0.25	89.6	Tamil Nadu
OD5	Ridhesh	Food Grains	Organic Staples	Ooty	10-11-2016	South	2355	0.26	918.45	Tamil Nadu
OD6	Adavan	Food Grains	Organic Staples	Dharmapuri	06-09-2015	West	2305	0.26	322.7	Tamil Nadu
OD7	Jonas	Fruits & Veggies	Fresh Vegetables	Trichy	06-09-2015	West	826	0.33	346.92	Tamil Nadu
OD8	Hafiz	Fruits & Veggies	Fresh Fruits	Ramanadhapuram	06-09-2015	West	1847	0.32	147.76	Tamil Nadu
OD9	Hafiz	Bakery	Biscuits	Tirunelveli	06-09-2015	West	791	0.23	181.93	Tamil Nadu
OD10	Krithika	Bakery	Cakes	Chennai	06-09-2015	West	1795	0.27	484.65	Tamil Nadu
OD11	Ganesh	Snacks	Chocolates	Karur	06-09-2015	West	1903	0.13	437.69	Tamil Nadu
OD12	Yadav	Eggs, Meat & ...	Eggs	Namakkal	06-09-2015	West	701	0.1	308.44	Tamil Nadu
OD13	Sharon	Snacks	Cookies	Dindigul	4/15/2018	South	1659	0.19	315.21	Tamil Nadu

grocery_sales 15

Output
Action Output

```
 #  Time      Action                                                                                                              Message                Duration / Fetch
24  12:41:33  select * from grocery_sales LIMIT 0, 5000                                                                           5000 row(s) returned   0.000 sec / 0.015 sec
25  12:42:32  SELECT Category,    SUM(Sales) AS total_sales,    SUM(Profit) AS total_profit FROM grocery_sales GROUP BY Category LIMIT 0, 5000   7 row(s) returned   0.016 sec / 0.000 sec
26  12:42:54  select * from grocery_sales LIMIT 0, 5000                                                                           5000 row(s) returned   0.000 sec / 0.015 sec
27  12:43:54  SELECT Region,    AVG(Discount) AS avg_discount,    SUM(Profit) AS total_profit FROM grocery_sales GROUP BY Region LIMIT 0, 5...   5 row(s) returned   0.031 sec / 0.000 sec
28  12:46:05  SELECT   CASE    WHEN Discount < 0.15 THEN 'Low'    WHEN Discount < 0.30 THEN 'Medium'    ELSE 'High'   END AS discount_buc...   3 row(s) returned   0.016 sec / 0.000 sec
29  12:47:42  select * from grocery_sales LIMIT 0, 5000                                                                           5000 row(s) returned   0.000 sec / 0.015 sec
```



```
-- Basic validation
-- Why:
-- Confirms one row = one order.
select * from grocery_sales;
SELECT COUNT(*) FROM grocery_sales;
SELECT COUNT(DISTINCT Order_ID) FROM grocery_sales;

-- Business aggregations
-- Why:
-- Confirms which features may influence the target variable.
SELECT Category,
       SUM(Sales) AS total_sales,
       SUM(Profit) AS total_profit
FROM grocery_sales
GROUP BY Category;

SELECT Region,
```

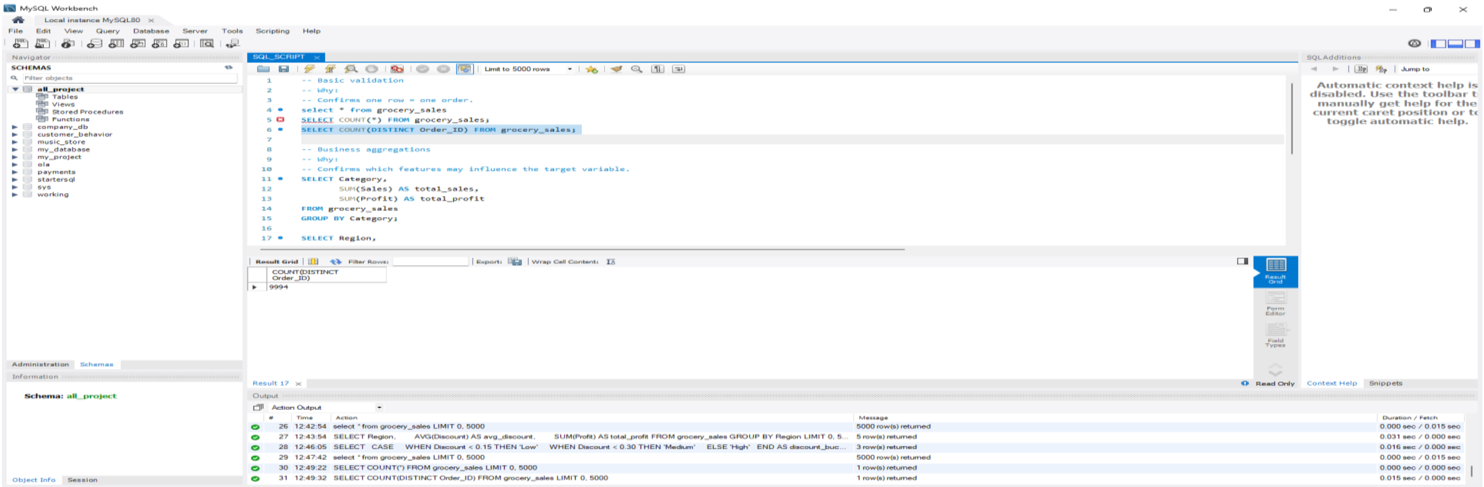
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

COUNT(*)
9994

Result 16

Output
Action Output

```
 #  Time      Action                                                                                                              Message                Duration / Fetch
25  12:42:32  SELECT Category,    SUM(Sales) AS total_sales,    SUM(Profit) AS total_profit FROM grocery_sales GROUP BY Category LIMIT 0, 5000   7 row(s) returned   0.016 sec / 0.000 sec
26  12:42:54  select * from grocery_sales LIMIT 0, 5000                                                                           5000 row(s) returned   0.000 sec / 0.015 sec
27  12:43:54  SELECT Region,    AVG(Discount) AS avg_discount,    SUM(Profit) AS total_profit FROM grocery_sales GROUP BY Region LIMIT 0, 5...   5 row(s) returned   0.031 sec / 0.000 sec
28  12:46:05  SELECT   CASE    WHEN Discount < 0.15 THEN 'Low'    WHEN Discount < 0.30 THEN 'Medium'    ELSE 'High'   END AS discount_buc...   3 row(s) returned   0.016 sec / 0.000 sec
29  12:47:42  select * from grocery_sales LIMIT 0, 5000                                                                           5000 row(s) returned   0.000 sec / 0.015 sec
30  12:49:22  SELECT COUNT(*) FROM grocery_sales LIMIT 0, 5000                                                                    1 row(s) returned      0.000 sec / 0.000 sec
```



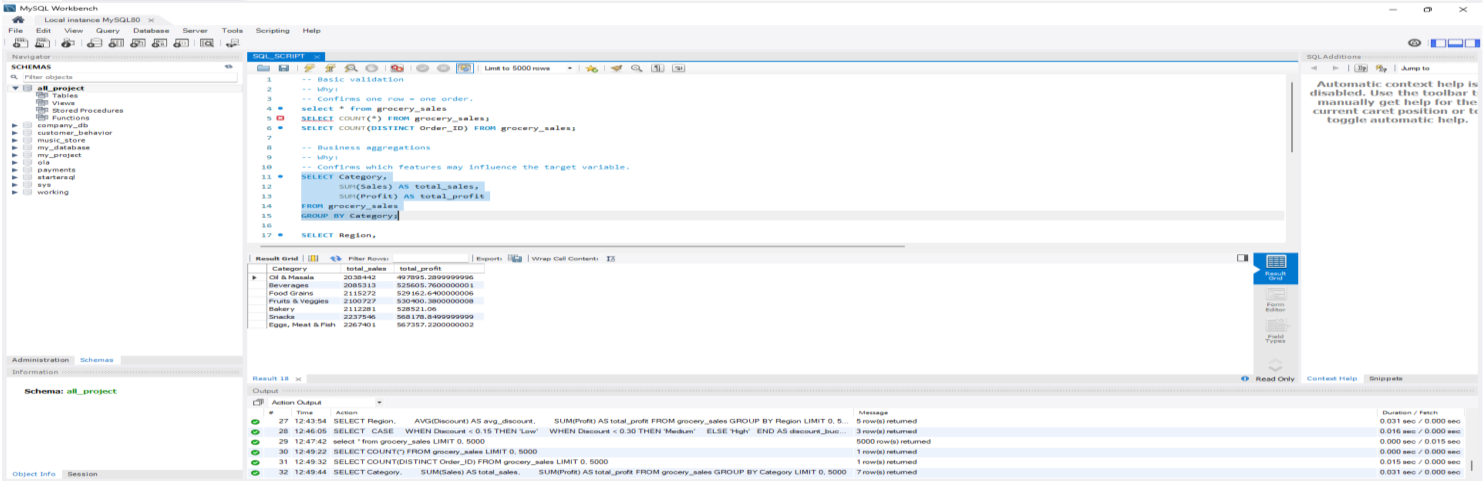
```sql
-- Basic validation
-- Why:
-- Confirms one row = one order.
select * from grocery_sales
SELECT COUNT(*) FROM grocery_sales;
SELECT COUNT(DISTINCT Order_ID) FROM grocery_sales;

-- Business aggregations
-- Why:
-- Confirms which features may influence the target variable.
SELECT Category,
       SUM(Sales) AS total_sales,
       SUM(Profit) AS total_profit
FROM grocery_sales
GROUP BY Category;

SELECT Region,
```

	COUNT(DISTINCT Order_ID)
▶	9994

```
   #  Time      Action                                                                                                     Message                  Duration / Fetch
  26  12:42:54  select * from grocery_sales LIMIT 0, 5000                                                                  5000 row(s) returned     0.000 sec / 0.015 sec
  27  12:43:54  SELECT Region,    AVG(Discount) AS avg_discount,    SUM(Profit) AS total_profit FROM grocery_sales GROUP BY Region LIMIT 0, 5...   5 row(s) returned   0.031 sec / 0.000 sec
  28  12:46:05  SELECT  CASE   WHEN Discount < 0.15 THEN 'Low'   WHEN Discount < 0.30 THEN 'Medium'   ELSE 'High'  END AS discount_buc...   3 row(s) returned   0.016 sec / 0.000 sec
  29  12:47:42  select * from grocery_sales LIMIT 0, 5000                                                                  5000 row(s) returned     0.000 sec / 0.015 sec
  30  12:49:22  SELECT COUNT(*) FROM grocery_sales LIMIT 0, 5000                                                           1 row(s) returned        0.000 sec / 0.000 sec
  31  12:49:32  SELECT COUNT(DISTINCT Order_ID) FROM grocery_sales LIMIT 0, 5000                                          1 row(s) returned        0.015 sec / 0.000 sec
```



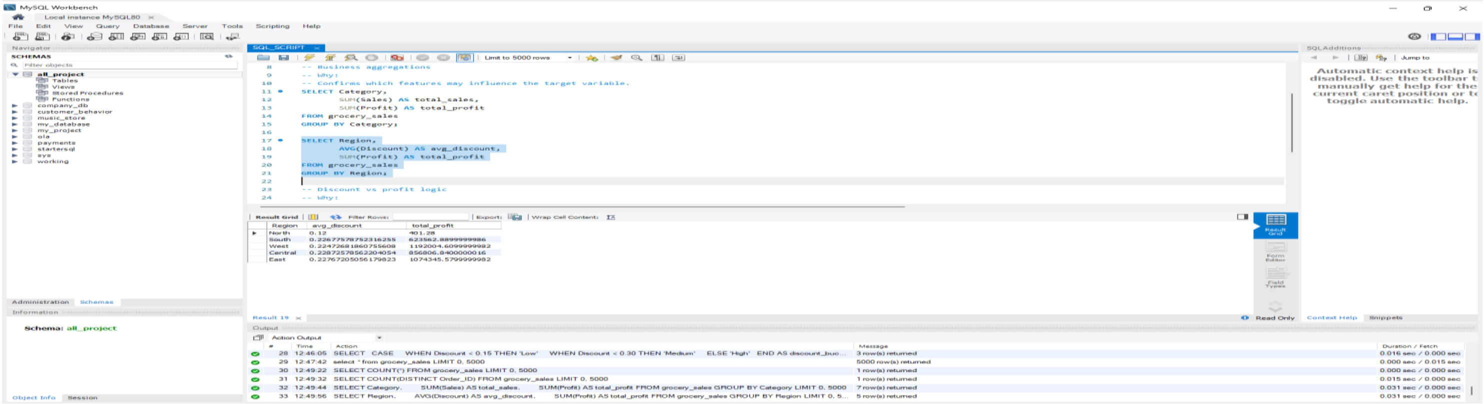
```sql
-- Basic validation
-- Why:
-- Confirms one row = one order.
select * from grocery_sales
SELECT COUNT(*) FROM grocery_sales;
SELECT COUNT(DISTINCT Order_ID) FROM grocery_sales;

-- Business aggregations
-- Why:
-- Confirms which features may influence the target variable.
SELECT Category,
       SUM(Sales) AS total_sales,
       SUM(Profit) AS total_profit
FROM grocery_sales
GROUP BY Category;

SELECT Region,
```

	Category	total_sales	total_profit
▶	Oil & Masala	2038442	497895.2899999996
	Beverages	2085313	525605.7600000001
	Food Grains	2115272	529162.6400000006
	Fruits & Veggies	2100727	530400.3800000008
	Bakery	2112281	528521.06
	Snacks	2237546	568178.8499999999
	Eggs, Meat & Fish	2267401	567357.2200000002

```
   #  Time      Action                                                                                                     Message                  Duration / Fetch
  27  12:43:54  SELECT Region,    AVG(Discount) AS avg_discount,    SUM(Profit) AS total_profit FROM grocery_sales GROUP BY Region LIMIT 0, 5...   5 row(s) returned   0.031 sec / 0.000 sec
  28  12:46:05  SELECT  CASE   WHEN Discount < 0.15 THEN 'Low'   WHEN Discount < 0.30 THEN 'Medium'   ELSE 'High'  END AS discount_buc...   3 row(s) returned   0.016 sec / 0.000 sec
  29  12:47:42  select * from grocery_sales LIMIT 0, 5000                                                                  5000 row(s) returned     0.000 sec / 0.015 sec
  30  12:49:22  SELECT COUNT(*) FROM grocery_sales LIMIT 0, 5000                                                           1 row(s) returned        0.000 sec / 0.000 sec
  31  12:49:32  SELECT COUNT(DISTINCT Order_ID) FROM grocery_sales LIMIT 0, 5000                                          1 row(s) returned        0.015 sec / 0.000 sec
  32  12:49:44  SELECT Category,    SUM(Sales) AS total_sales,    SUM(Profit) AS total_profit FROM grocery_sales GROUP BY Category LIMIT 0, 5000   7 row(s) returned   0.031 sec / 0.000 sec
```



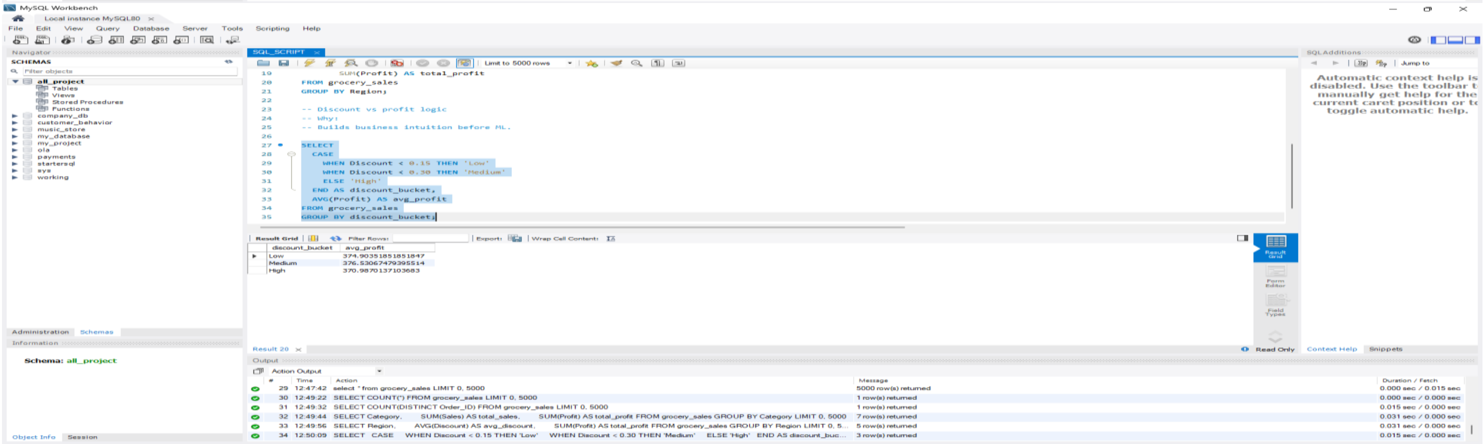
```sql
-- Business aggregations
-- Why:
-- Confirms which features may influence the target variable.
SELECT Category,
       SUM(Sales) AS total_sales,
       SUM(Profit) AS total_profit
FROM grocery_sales
GROUP BY Category;

SELECT Region,
       AVG(Discount) AS avg_discount,
       SUM(Profit) AS total_profit
FROM grocery_sales
GROUP BY Region;

-- Discount vs profit logic
-- Why:
```

	Region	avg_discount	total_profit
▶	North	0.12	401.28
	South	0.22677578752316255	623562.8899999986
	West	0.22472681860755608	1192004.6099999982
	Central	0.22872578562204054	856806.8400000016
	East	0.22767205056179823	1074345.5799999982

```
   #  Time      Action                                                                                                     Message                  Duration / Fetch
  28  12:46:05  SELECT  CASE   WHEN Discount < 0.15 THEN 'Low'   WHEN Discount < 0.30 THEN 'Medium'   ELSE 'High'  END AS discount_buc...   3 row(s) returned   0.016 sec / 0.000 sec
  29  12:47:42  select * from grocery_sales LIMIT 0, 5000                                                                  5000 row(s) returned     0.000 sec / 0.015 sec
  30  12:49:22  SELECT COUNT(*) FROM grocery_sales LIMIT 0, 5000                                                           1 row(s) returned        0.000 sec / 0.000 sec
  31  12:49:32  SELECT COUNT(DISTINCT Order_ID) FROM grocery_sales LIMIT 0, 5000                                          1 row(s) returned        0.015 sec / 0.000 sec
  32  12:49:44  SELECT Category,    SUM(Sales) AS total_sales,    SUM(Profit) AS total_profit FROM grocery_sales GROUP BY Category LIMIT 0, 5000   7 row(s) returned   0.031 sec / 0.000 sec
  33  12:49:56  SELECT Region,    AVG(Discount) AS avg_discount,    SUM(Profit) AS total_profit FROM grocery_sales GROUP BY Region LIMIT 0, 5...   5 row(s) returned   0.031 sec / 0.000 sec
```



```sql
       SUM(Profit) AS total_profit
FROM grocery_sales
GROUP BY Region;

-- Discount vs profit logic
-- Why:
-- Builds business intuition before ML.

SELECT
  CASE
    WHEN Discount < 0.15 THEN 'Low'
    WHEN Discount < 0.30 THEN 'Medium'
    ELSE 'High'
  END AS discount_bucket,
  AVG(Profit) AS avg_profit
FROM grocery_sales
GROUP BY discount_bucket;
```

	discount_bucket	avg_profit
▶	Low	374.90351851851847
	Medium	376.53067479395514
	High	370.9870137103683

```
   #  Time      Action                                                                                                     Message                  Duration / Fetch
  29  12:47:42  select * from grocery_sales LIMIT 0, 5000                                                                  5000 row(s) returned     0.000 sec / 0.015 sec
  30  12:49:22  SELECT COUNT(*) FROM grocery_sales LIMIT 0, 5000                                                           1 row(s) returned        0.000 sec / 0.000 sec
  31  12:49:32  SELECT COUNT(DISTINCT Order_ID) FROM grocery_sales LIMIT 0, 5000                                          1 row(s) returned        0.015 sec / 0.000 sec
  32  12:49:44  SELECT Category,    SUM(Sales) AS total_sales,    SUM(Profit) AS total_profit FROM grocery_sales GROUP BY Category LIMIT 0, 5000   7 row(s) returned   0.031 sec / 0.000 sec
  33  12:49:56  SELECT Region,    AVG(Discount) AS avg_discount,    SUM(Profit) AS total_profit FROM grocery_sales GROUP BY Region LIMIT 0, 5...   5 row(s) returned   0.031 sec / 0.000 sec
  34  12:50:09  SELECT  CASE   WHEN Discount < 0.15 THEN 'Low'   WHEN Discount < 0.30 THEN 'Medium'   ELSE 'High'  END AS discount_buc...   3 row(s) returned   0.015 sec / 0.000 sec
```