

# SQL for Data Analysis: Beginner Level III

## Real-World e-Commerce Analytics



Prepared By: Shambhu Kumar Kushwaha  
Date: 28/08/2025

# Introduction

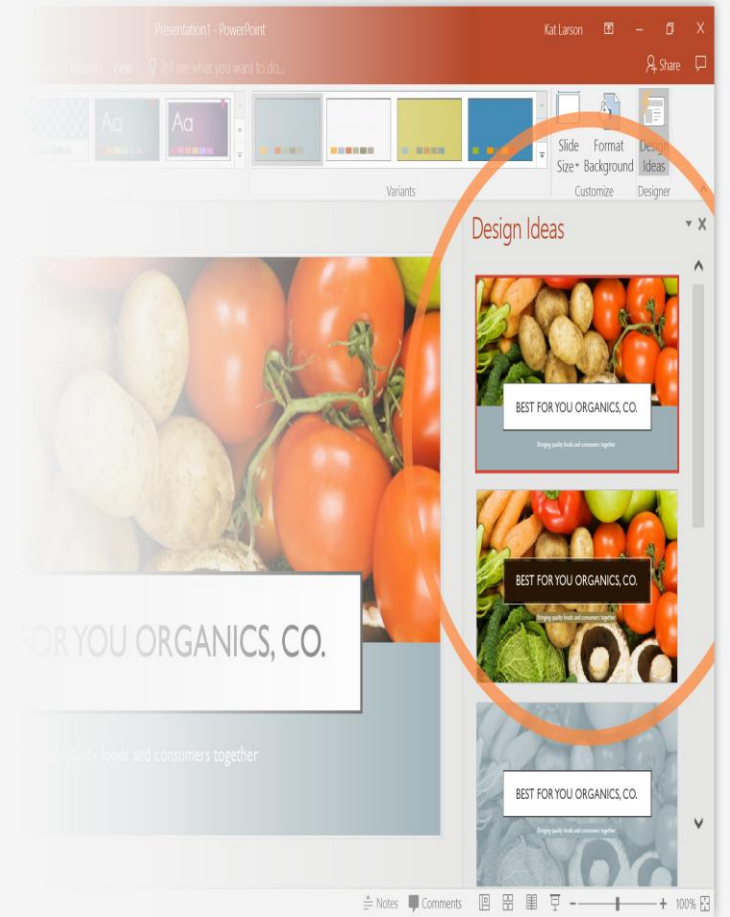
This project is designed to provide a step-by-step, beginner-friendly journey into SQL and data analysis using an e-commerce business scenario. The goal is to gradually build a relational database, write SQL queries to solve real-world business problems, and interpret insights that can help in decision-making.

❑ The project was divided into **three levels**:

•**Level I – EcommerceOrders Table:**

At the beginner level, we created a table called **EcommerceOrders**, which contained basic order-related information such as OrderID, CustomerID, ProductID, OrderDate, and TotalAmount. This stage introduced us to the process of creating tables, importing datasets, and understanding how raw transaction data is stored in a database. By working on this table, we learned how to handle order data and run simple SQL queries to extract meaningful insights.

[Level I Dataset](#)



---

### •Level II – CustomerDetails Table:

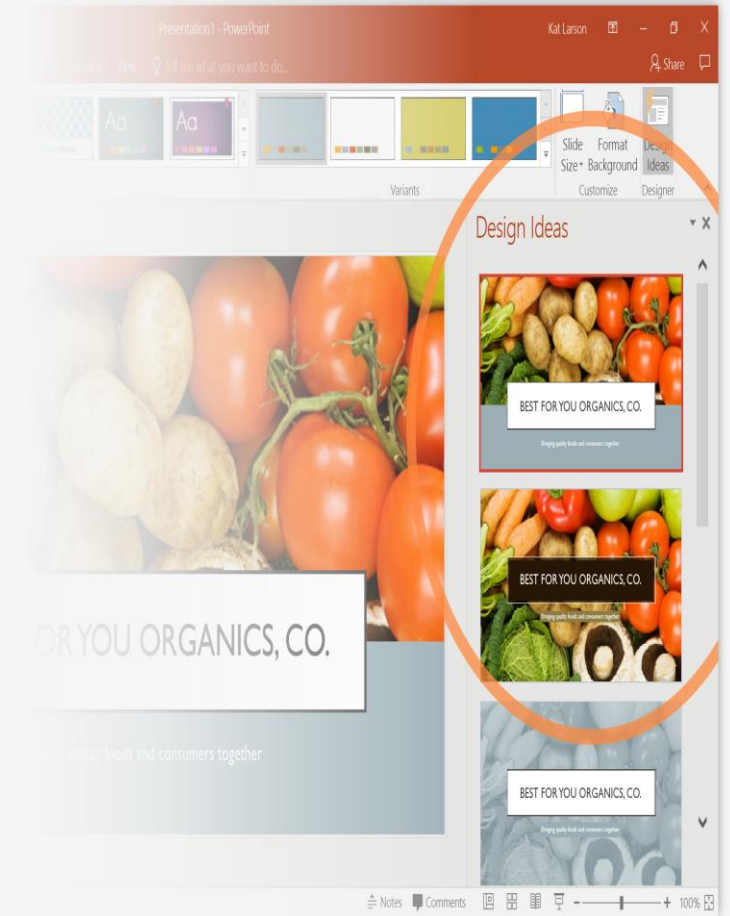
At the intermediate level, we introduced the **CustomerDetails** table. This table included important attributes such as CustomerID, CustomerName, Location, and Contact Information. By linking this table with the EcommerceOrders table, we explored the concept of **relationships between tables** (using primary keys and foreign keys). This helped us to analyze customer demographics, purchasing behavior, and geographic distribution of sales.

[Level II Dataset](#)

### •Level III – ProductDetails Table:

At the advanced level, we created the **ProductDetails** table, which stored information like ProductID, ProductName, Category, and Price. Combining this table with the previous ones allowed us to perform deeper analyses such as identifying best-selling products, analyzing product categories, and measuring sales performance.

[Level III Dataset](#)



# Task 1-1: Data Download, Import, and Database Connection

## Steps Performed:

### 1 Data Download

- Collected the provided CSV files containing order and cust
- Verified file structure (column names, delimiters, data typ

### 2 Database Setup in MySQL

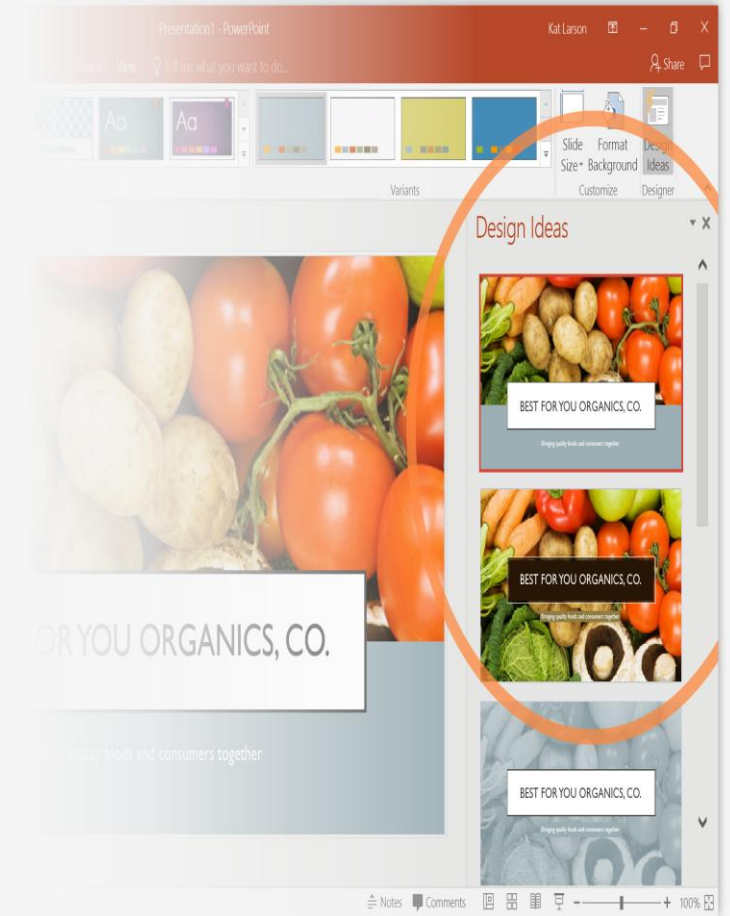
- Created a new database schema.

```
CREATE DATABASE ecommerce_analytics;
```

### 3 Table Creation

- Designed schema based on CSV structure.

```
CREATE TABLE productdetails (  
  ProductID VARCHAR(25) PRIMARY KEY,  
  ProductName VARCHAR(100),  
  Price DECIMAL(10,2),  
  Category VARCHAR(50)  
);
```





4

## Importing CSV Data into Tables

- The CSV files were imported into MySQL tables using the I
- Verified Correctness with:

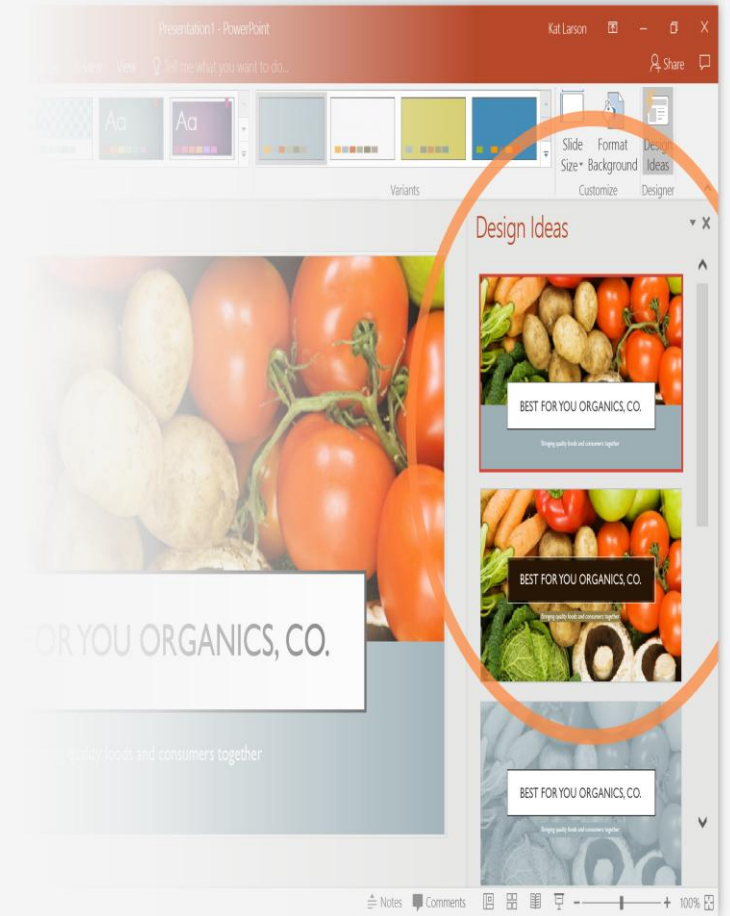
`SELECT *`

`FROM ecommerce_analytics.ProductDetails;`

5

## Outcome:

- Successfully created and populated **ProductDetails**.
- Products now connected with orders and customers, comp



# Task 1-2: Find Orders within a Specific Date Range

**OBJECTIVE:**

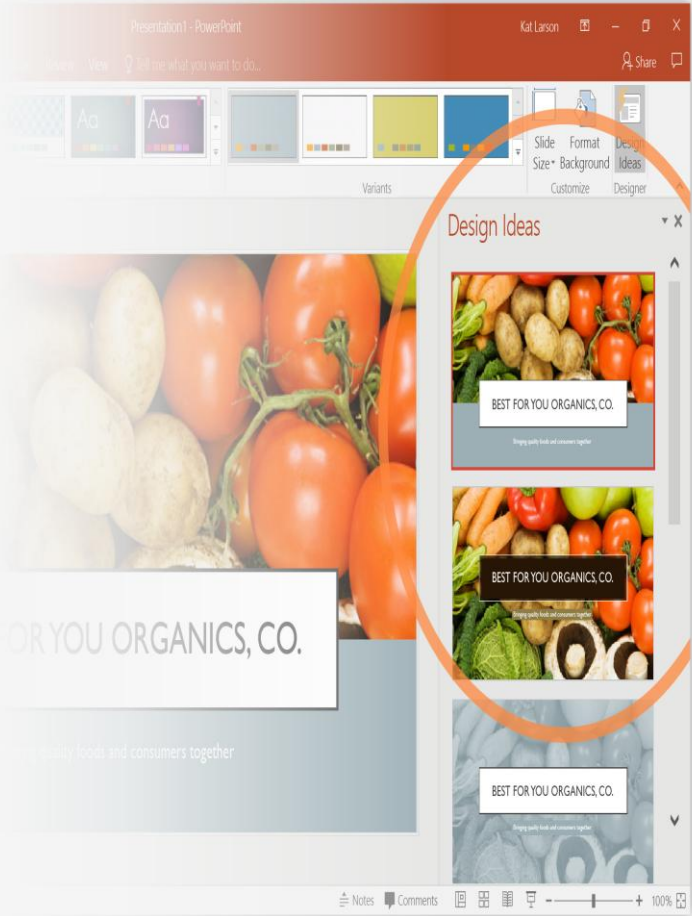
Retrieve orders placed between Jan 1, 2024 – Mar 31, 2024.

❑ Query:

```
SELECT
  OrderID,
  OrderDate,
  CustomerID,
  TotalAmount
FROM ecommerceorders
WHERE OrderDate BETWEEN '2024-01-01' AND '2024-03-31'
ORDER BY OrderDate;
```

❑ Result:

Result Grid				
Filter Rows:				
	OrderID	OrderDate	CustomerID	TotalAmount
▶	ORD0000215	2024-01-01	CUST0038	218.35
	ORD0000267	2024-01-01	CUST0028	162.40
	ORD0000284	2024-01-01	CUST0082	132.92
	ORD0000096	2024-01-02	CUST0081	77.58
	ORD0000998	2024-01-02	CUST0100	125.31



# Task 1-3: Retrieve Orders with Specific Statuses

## OBJECTIVE:

Find all orders with statuses Shipped or Delivered.

❑ Query:

**SELECT**

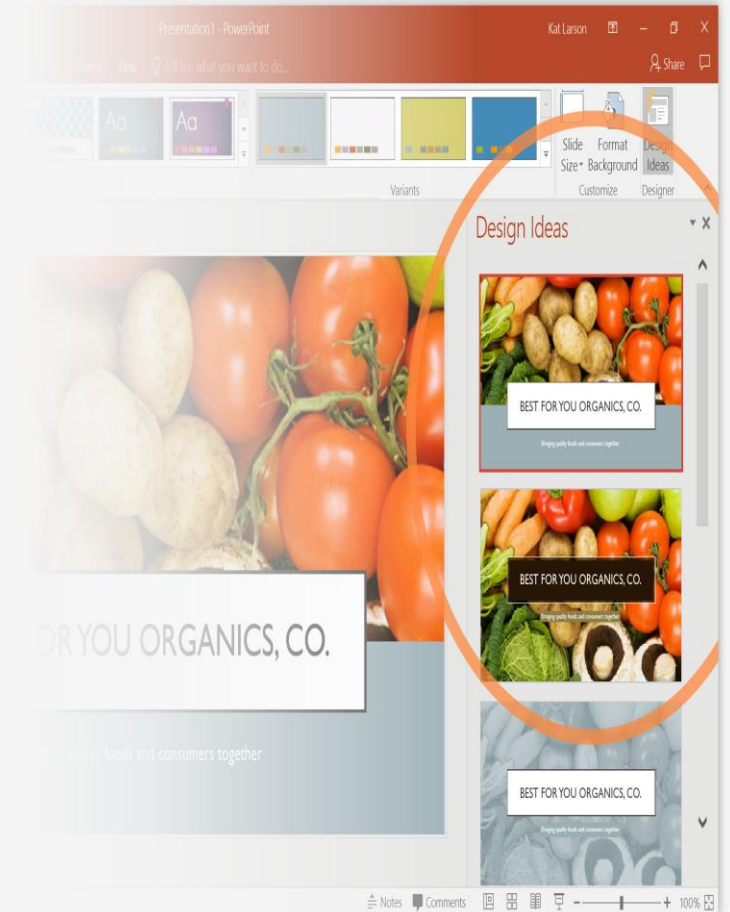
OrderID,  
OrderStatus,  
CustomerID

**FROM** ecommerceorders

**WHERE** OrderStatus IN ('Shipped', 'Delivered');

❑ Result:

Result Grid			
Filter Rows: <input type="text"/>			
Edit:			
	OrderID	OrderStatus	CustomerID
▶	ORD0000000	Shipped	CUST0050
	ORD0000001	Delivered	CUST0062
	ORD0000002	Shipped	CUST0018
	ORD0000003	Shipped	CUST0019
	ORD0000004	Delivered	CUST0061



# Task 1-4: Customers with Above-Average Order Amounts

## OBJECTIVE:

Identify high-value customers spending above the average order :

### Query:

**SELECT**

CustomerID,  
OrderID,  
TotalAmount

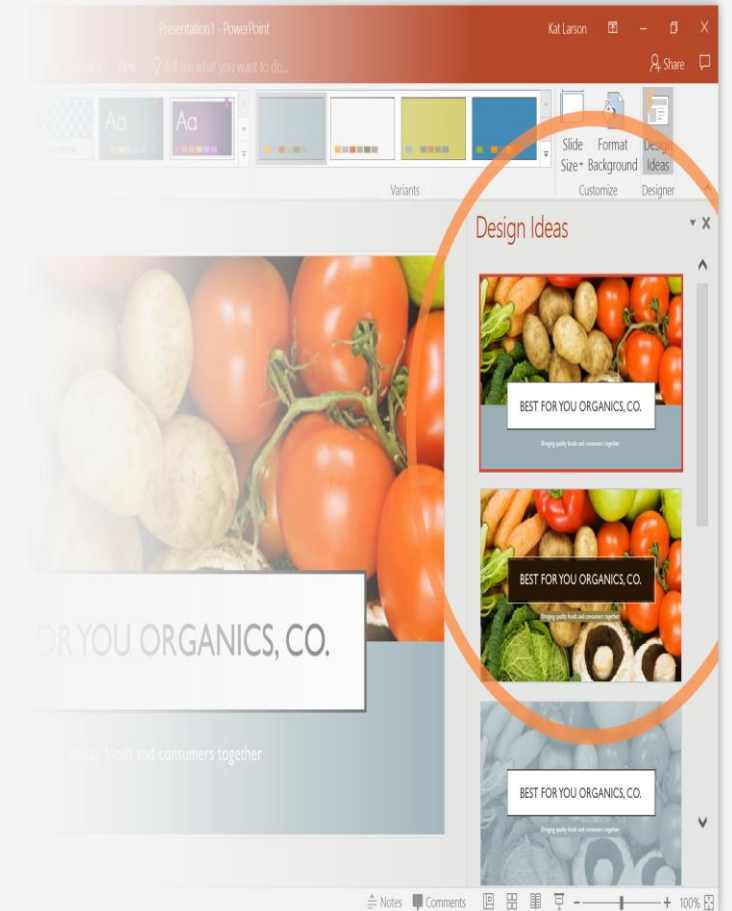
FROM ecommerceorders

WHERE TotalAmount > (  
SELECT AVG(TotalAmount)  
FROM ecommerceorders)

ORDER BY TotalAmount DESC;

### Result:

Result Grid				Filter Rows:	Edit:
	CustomerID	OrderID	TotalAmount		
▶	CUST0092	ORD0000329	1772.64		
	CUST0048	ORD0000577	1694.00		
	CUST0042	ORD0000535	1647.09		
	CUST0051	ORD0000310	1553.13		
	CUST0027	ORD0000798	1482.30		





# Task 1-5: List Categories with Average Price Above Threshold

## OBJECTIVE:

Retrieve categories where average product price > \$100.

❑ Query:

SELECT

Category,

AVG(Price) AS "average price per category"

FROM productdetails

GROUP BY Category

HAVING AVG(Price) > 100

ORDER BY AVG(Price) DESC;

❑ Result:

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	Category	average price per category		
▶	Books	222.350000		
	Home Appliances	212.346000		
	Beauty	206.822000		
	Electronics	200.350000		

# Task 1-6: Rank Products by Popularity within Category




## OBJECTIVE:

Rank products by quantity ordered per category.

❑ Query:

```
WITH CTE1 AS (  
  SELECT  
    e.Category,  
    e.ProductID,  
    p.ProductName,  
    e.Quantity  
  FROM productdetails AS p  
  JOIN ecommerceorders AS e  
    ON p.ProductID = e.ProductID  
)  
CTE2 AS (  
  SELECT *,  
    RANK() OVER(PARTITION BY Category ORDER BY Quantity DESC) AS rn  
  FROM CTE1  
)  
SELECT *  
FROM CTE2  
ORDER BY Category, rn;
```

❑ Result:

Result Grid    Filter Rows: <input type="text"/>   Export:    Wrap Cell Content: 					
	Category	ProductID	ProductName	Quantity	rn
▶	Beauty	PROD0022	Perfume	10	1
	Beauty	PROD0021	Lipstick	9	2
	Beauty	PROD0025	Shampoo	9	2
	Beauty	PROD0024	Facewash	9	2
	Beauty	PROD0025	Shampoo	9	2

# Task 1-7: Find Products In-Stock but Never Ordered

## OBJECTIVE:

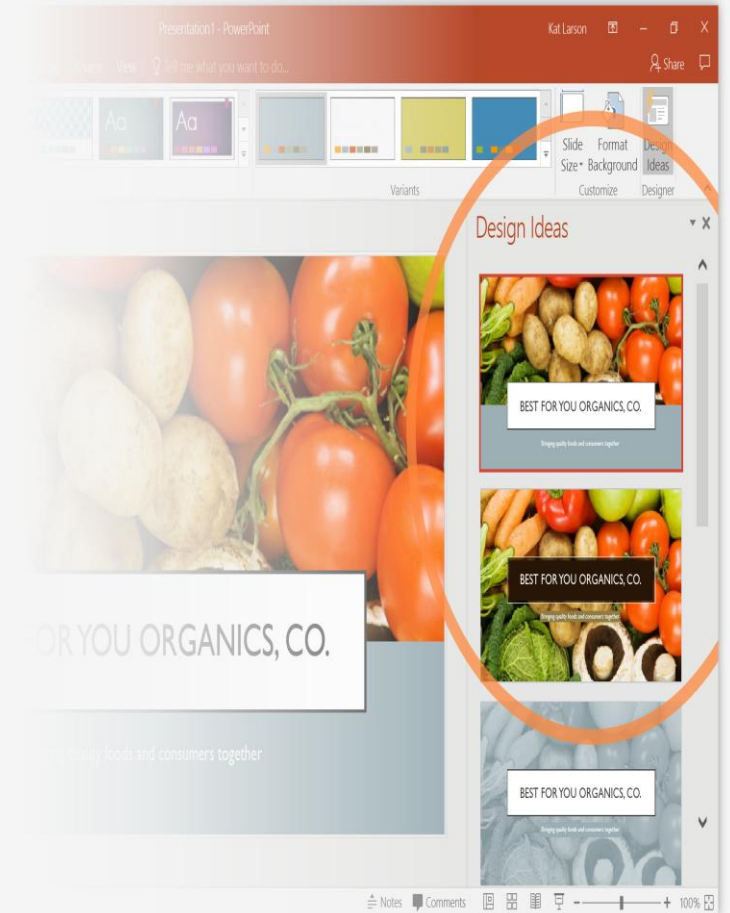
List products available in stock but never purchased.

### ❑ Query:

```
SELECT
    ProductID,
    ProductName
FROM productdetails
WHERE ProductID NOT IN (
    SELECT DISTINCT ProductID
    FROM ecommerceorders
);
```

### ❑ Result:

Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
	ProductID	ProductName			
▶	PROD0031	Dice			
*	NULL	NULL			



# Task 1-8:Count High-Value Orders Using Conditional Aggregation

**OBJECTIVE:**

Retrive the largest 10 orders by quantity.

❑ Query:

```
SELECT
    DATE_FORMAT(OrderDate, "%M") AS Month,
    COUNT(CASE WHEN TotalAmount > 500 THEN 1 END) AS High_Value_Order_Count,
    SUM(CASE WHEN TotalAmount > 500 THEN TotalAmount ELSE 0 END) AS High_Value_Order_Total
FROM ecommerceorders
GROUP BY DATE_FORMAT(OrderDate, "%M")
ORDER BY MIN(OrderDate);
```

Result:

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

	Month	High_Value_Order_Count	High_Value_Order_Total
▶	November	16	13580.40
	December	15	11321.07
	January	16	14420.77
	February	19	15121.54
	March	4	3906.86



## OBJECTIVE:

❏ Query:

CustomerID,  
CustomerName,  
Email

WHERE Email LIKE "%@gmail.com";

<div> <div>Result Grid</div> <div>  Filter Rows: <input type="text"/> </div> <div> <div>Edit:</div> <div> </div> <div>Export/Import:</div> <div> </div> <div>Wrap Cell Content:</div> <div> </div> </div> </div>			
	CustomerID	CustomerName	Email
▶	CUST0003	Elizabeth Woods	elizabeth@gmail.com
	CUST0006	Theodore Mcgrath	theodore@gmail.com
	CUST0011	Sean Green	sean@gmail.com
	CUST0012	Kimberly Smith	kimberly@gmail.com
	CUST0016	Cheryl Bradley	cheryl@gmail.com

# Task 1-10: Calculate Sales Totals Using Rollups

## OBJECTIVE:

Generate multi-level sales summary by Category → Product.

❑ Query:

SELECT

Category,

ProductID,

SUM(TotalAmount) AS Total\_Sales

FROM ecommerceorders

GROUP BY Category, ProductID WITH ROLLUP;

Result:

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
Category	ProductID	Total_Sales	
Beauty	PROD0021	10867.38	
Beauty	PROD0022	11690.30	
Beauty	PROD0023	11844.39	
Beauty	PROD0024	12000.23	
Beauty	PROD0025	9208.81	

# CONCLUSION

The **Level III SQL Analysis** expanded the foundation built in **Levels I & II** by incorporating **product-level insights**.

❖ **Key Insights Gained:**

- ✓ Seasonal sales trends (Task 1-2).
- ✓ High-value customer & order segmentation (Tasks 1-4, 1-8).
- ✓ Premium product categories (Task 1-5).
- ✓ Product popularity ranking (Task 1-6).
- ✓ Unordered stock detection (Task 1-7).
- ✓ Multi-level rollup reporting (Task 1-10).

☞ By combining **orders, customers, and products**, we now have a **360° view of e-commerce performance**, supporting **marketing, operations, and strategic planning**.

