SQL for Data Analysis: Intermediate Level I & II

Real-World e-Commerce Analytics

Prepared By: Shambhu Kumar Kushwaha
Date: 30/08/2025

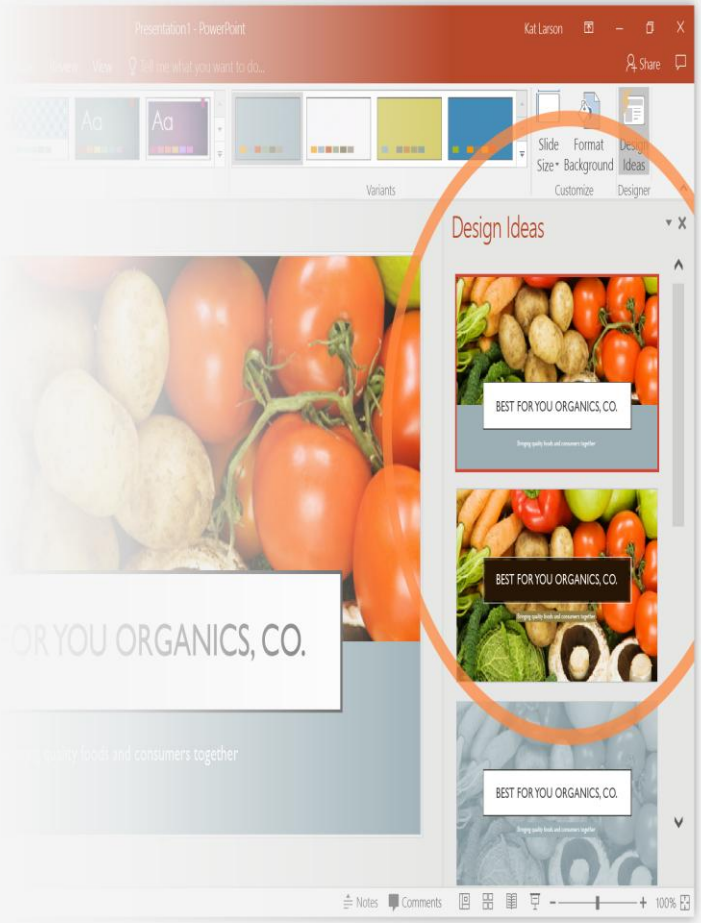
Introduction

This project was designed to simulate a real-world e-commerce database and perform analytical reporting using SQL. The objective was to create a structured database environment, import raw data into different tables, and then apply advanced SQL techniques to extract meaningful business insights.

At the **Beginner Level (I–III)**, three core tables were created and populated:

- **EcommerceOrders** – storing order-related transactions.
- **CustomerDetails** – storing customer information.
- **ProductDetails** – storing product information.

At the **Intermediate** Level, advanced SQL operations were applied to this database. These included subqueries, joins, aggregations, window functions, and views. The aim was to analyze sales performance, understand customer behavior, identify high-value customers, detect inactive customers, and evaluate product popularity.



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

Steps Performed:

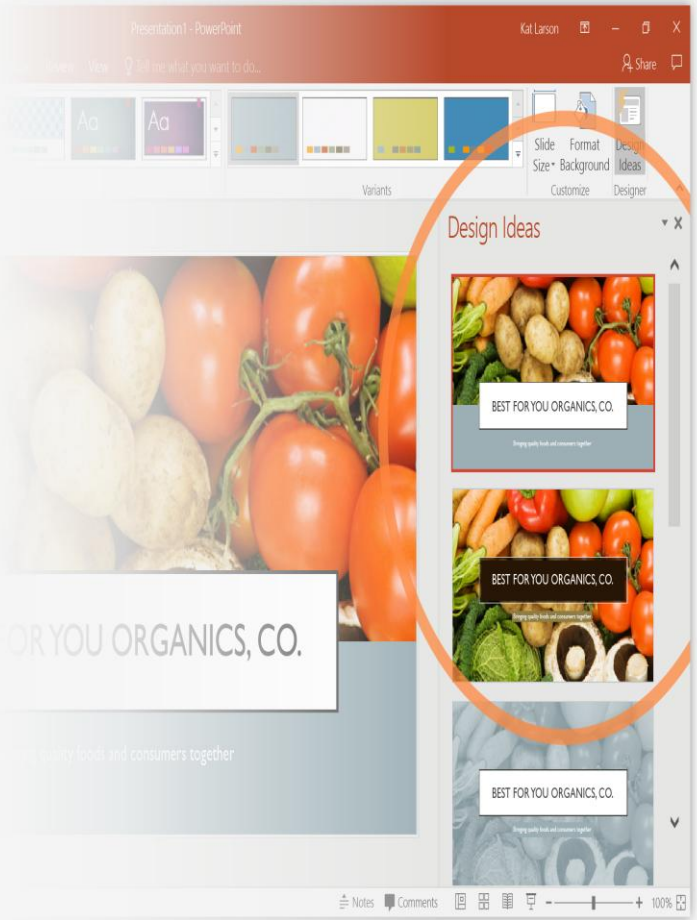
1. **Data Download**
   - Collected the provided CSV files containing order and cust
   - Verified file structure (column names, delimiters, data typ

2. **Database Setup in MySQL**
   - Created a new database schema.

```
CREATE DATABASE ecommerce_analytics_II;
```

3. **Table Creation**
   - Designed schema based on CSV structure.

```
CREATE TABLE CustomerDetails(
CustomerID VARCHAR(20) PRIMARY KEY,
CustomerFirstName VARCHAR(40),
CustomerLastName VARCHAR(40),
Email VARCHAR(100),
ShippingAddress VARCHAR(500)
);
```



```
CREATE TABLE eCommerceOrders (
    OrderID VARCHAR(20) PRIMARY KEY,
    CustomerID VARCHAR(20),
    OrderDate DATE,
    ProductID VARCHAR(20),
    ProductName VARCHAR(100),
    Quantity INT,
    Price DECIMAL(10,2),
    TotalAmount DECIMAL(10,2),
    Category VARCHAR(50),
    OrderStatus VARCHAR(50),
    PaymentMethod VARCHAR(50),
ShippingAddress VARCHAR(255),
    ShippingDate DATE
);
```

4

Importing CSV Data into Tables

•The CSV files were imported into MySQL tables using the I
Workbench "Table Data Import Wizard".

• Verified Correctness with:

```
SELECT *
FROM ecommerce_analytics_II.ProductDetails;

SELECT *
FROM ecommerce_analytics_II.eCommerceOrder
```

5

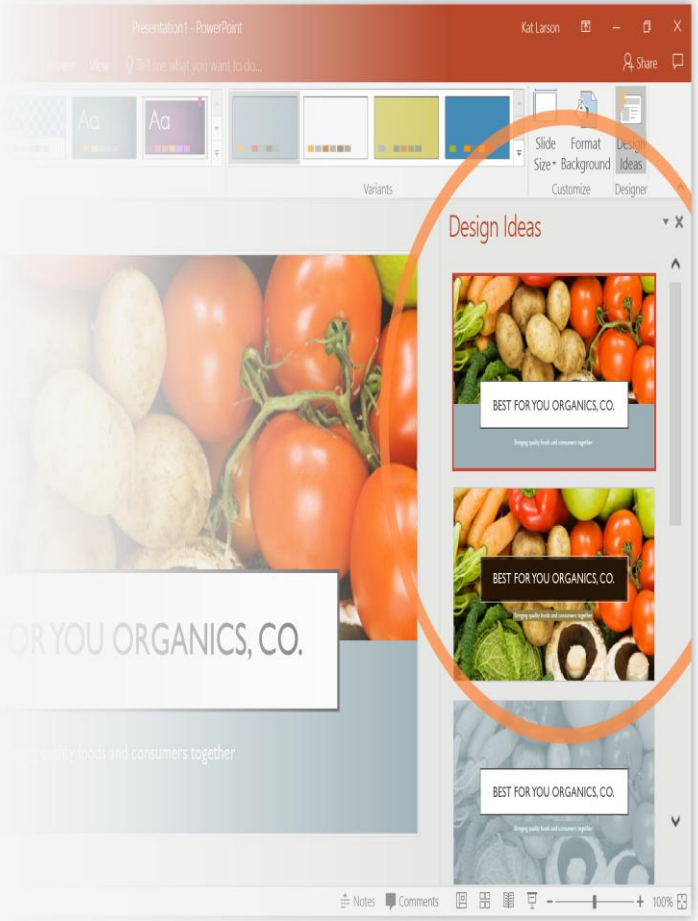
Outcome:

•Successfully created and populated ProductDetails & eCommerceOrders .

Task 1-2:Retrieve High-Value Orders by Specific Customers in Last Three Months

OBJECTIVE:

Identify orders above $500 in the last 3 months (from 2024-11-11) to focus on premium customers.



❑ Query:

```
SELECT
   CustomerID,
   OrderID,
   OrderDate,
   TotalAmount
FROM ecommerceorders
WHERE TotalAmount > 500
  AND OrderDate >= DATE_SUB('2024-11-11', INTERVAL 3 MONTH)
ORDER BY CustomerID, OrderDate DESC;
```

- Result:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

CustomerID	OrderID	OrderDate	TotalAmount
CUST0005	ORD0001059	2024-09-29	509.16
CUST0012	ORD0002640	2024-10-16	658.42
CUST0016	ORD0003973	2024-11-09	755.60
CUST0018	ORD0002841	2024-09-27	1223.01
CUST0022	ORD0008641	2024-10-18	594.88
CUST0027	ORD0009678	2024-10-28	982.40
CUST0027	ORD0007812	2024-09-30	904.40
CUST0032	ORD0008540	2024-10-08	548.82
CUST0032	ORD0008488	2024-08-14	544.62
CUST0043	ORD0009763	2024-09-18	810.67
CUST0053	ORD0007302	2024-11-06	1269.09
CUST0054	ORD0004132	2024-08-22	872.41
CUST0057	ORD0000850	2024-10-31	669.30
CUST0058	ORD0003949	2024-08-25	512.28
CUST0074	ORD0004680	2024-10-07	992.85
CUST0076	ORD0006753	2024-11-07	906.95
CUST0087	ORD0005101	2024-09-05	577.68
CUST0089	ORD0003391	2024-10-18	771.20

Task 1-3: Calculate Monthly Revenue by Category

OBJECTIVE:

Find revenue trends by product category across months.

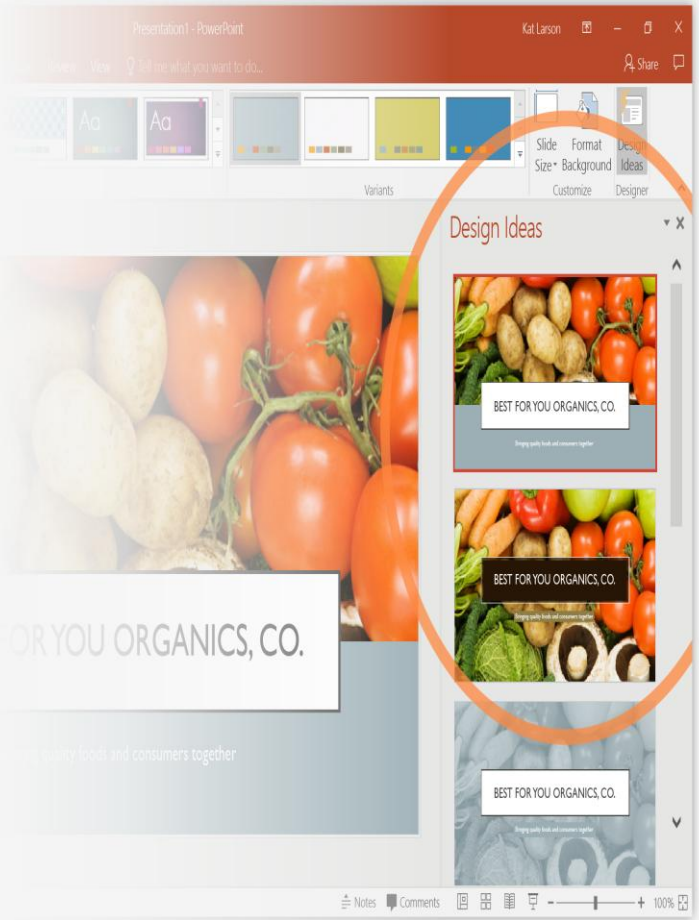
- Query:

```
SELECT
    Category,
    DATE_FORMAT(OrderDate,'%M') AS Month,
    SUM(TotalAmount) AS total_revenue
FROM ecommerceorders
GROUP BY Category, Month
ORDER BY Category, Month;
```

- Result:

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

Category	Month	total_revenue
Beauty	April	43230.78
Beauty	August	33844.36
Beauty	December	39549.25
Beauty	February	22691.47
Beauty	January	36480.79
Beauty	July	33775.58
Beauty	June	33709.72
Beauty	March	44034.22
Beauty	May	39958.46



Task 1-4:Identify Incomplete Orders with Advanced Joins

OBJECTIVE:
Detect missing shipping addresses using a RIGHT JOIN.

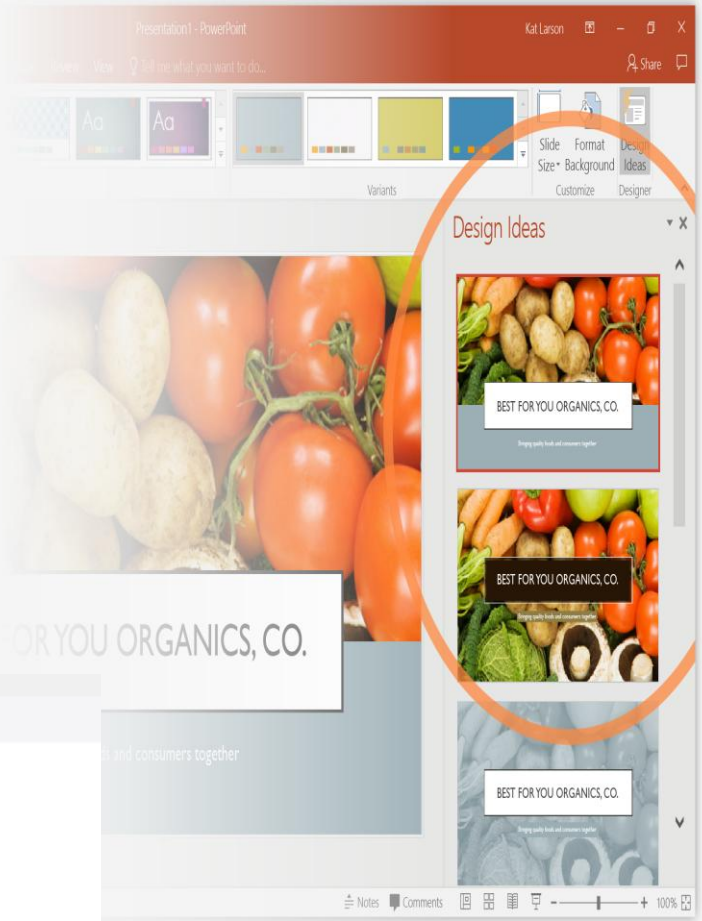
- Query:

```
SELECT
    e.OrderDate,
    e.OrderID,
    e.CustomerID,
    e.ShippingAddress
FROM ecommerceorders e
RIGHT JOIN customerdetails c
ON e.CustomerID = c.CustomerID
WHERE e.ShippingAddress IS NULL;
```

- Result:

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

OrderDate	OrderID	CustomerID	ShippingAddress
NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL



Task 1-5: Find Products with Above-Average Sales in Category Using Subquery

OBJECTIVE:

Compare product sales to category averages to highlight top performers.

- Query:

```
SELECT
    ProductID,
    Category,
   SUM(TotalAmount) AS TotalSales
FROM ecommerceorders eo
GROUP BY ProductID, Category
HAVING SUM(TotalAmount) >
 (
  SELECT AVG(category_total)
  FROM (
    SELECT
      SUM(TotalAmount) AS category_total
    FROM ecommerceorders
    WHERE Category = eo.Category
    GROUP BY ProductID
  ) AS sub
 );
```

- Result:

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

ProductID	Category	TotalSales
PROD0079	Books	2487.11
PROD0079	Clothing	1025.90
PROD0312	Home Appliances	2142.27
PROD0311	Beauty	1171.37
PROD0233	Clothing	1295.48
PROD0473	Electronics	1953.99
PROD0072	Clothing	1628.15
PROD0040	Books	1243.67
PROD0320	Electronics	1043.77

Task 1-6:Calculate Running Total of Sales by Product

OBJECTIVE:
Track cumulative sales performance per product.

- Query:

```
SELECT
        ProductID,
       OrderDate,
      TotalAmount,
      SUM(TotalAmount) OVER ( PARTITION BY ProductID  ORDER BY OrderDate
   ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
 ) AS RunningTotal
FROM ecommerceorders
ORDER BY ProductID, OrderDate;
```

- Result:

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

ProductID	OrderDate	TotalAmount	RunningTotal
PROD0001	2023-11-13	38.40	38.40
PROD0001	2023-12-17	88.32	126.72
PROD0001	2024-01-02	309.40	436.12
PROD0001	2024-01-25	69.67	505.79
PROD0001	2024-03-01	282.81	788.60
PROD0001	2024-04-03	85.12	873.72
PROD0001	2024-04-14	244.44	1118.16
PROD0001	2024-05-18	6.41	1124.57
PROD0001	2024-05-20	331.02	1455.59

Task 1-7: Classify Customers by Spending Tiers

OBJECTIVE:
Segment customers into Low, Medium, High based on total spending.

- Query:

```
SELECT
  CustomerID,
  SUM(TotalAmount) AS TotalSpending,
CASE
  WHEN SUM(TotalAmount) < 500 THEN 'Low'
  WHEN SUM(TotalAmount) BETWEEN 500 AND 2000 THEN 'Medium'
  ELSE 'High'
END AS SpendingTier
FROM ecommerceorders
GROUP BY CustomerID
ORDER BY TotalSpending DESC;
```

- Result:

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

CustomerID	TotalSpending	SpendingTier
CUST0326	5171.20	High
CUST0625	4818.93	High
CUST1388	4800.62	High
CUST0715	4655.74	High
CUST1971	4549.94	High
CUST1384	4500.88	High
CUST0742	4397.85	High
CUST1710	4303.13	High
CUST0627	4301.63	High

Task 1-8:Advanced Pattern Matching for Customer Segmentation

OBJECTIVE:
Extract Customers with gmail.com or yahoo.com domains.

- Query:

```
SELECT
    CustomerID, Email
FROM customerdetails
WHERE Email LIKE '%gmail.com'
OR Email LIKE '%yahoo.com';
```

- Result:

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

CustomerID	Email
CUST0001	jeremy@gmail.com
CUST0002	nathan@yahoo.com
CUST0003	cheyenne@yahoo.com
CUST0004	edward@gmail.com
CUST0005	stephanie@gmail.com
CUST0006	jennifer@gmail.com
CUST0008	louis@yahoo.com
CUST0010	carol@yahoo.com
CUST0011	steven@yahoo.com

Task 1-9: Find Customers with No Orders in Last Six Months

OBJECTIVE:
Identify inactive customers (no purchases in 6 months)..

- Query:

```
SELECT
    c.CustomerID,
    c.CustomerFirstName,
    c.CustomerLastName
FROM CustomerDetails c
LEFT JOIN eCommerceOrders o
ON c.CustomerID = o.CustomerID
AND o.OrderDate >= DATE_SUB('2024-11-11', INTERVAL 6 MONTH)
WHERE o.CustomerID IS NULL;
```

- Result:

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

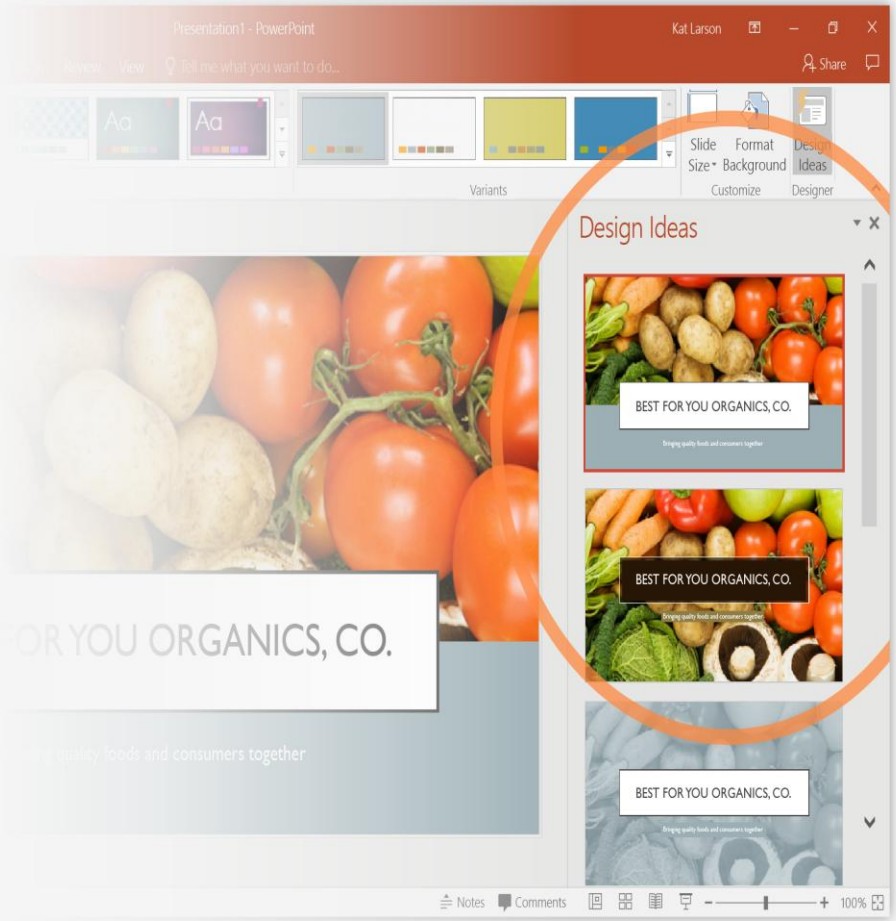
CustomerID	CustomerFirstName	CustomerLastName
CUST0003	Cheyenne	Smith
CUST0007	Kathy	Nichols
CUST0033	Diana	Johnson
CUST0041	Mary	Glover
CUST0046	Victoria	Mitchell
CUST0063	Deborah	Morgan
CUST0065	Lisa	Ferguson
CUST0080	Larry	Morales
CUST0096	Michael	Hall

Task 1-10:Create Temporary View for High-Value Orders

OBJECTIVE:
Simplify repeated analysis of orders above $500.

- Query:

```
CREATE VIEW HighValueOrders AS
    SELECT
        OrderID,
        CustomerID,
        TotalAmount
    FROM eCommerceOrders
    WHERE TotalAmount > 500;
```



CONCLUSION

The analysis of the e-commerce dataset highlights several important insights. High-value orders were successfully identified, providing visibility into major transactions that contribute significantly to revenue. Customer information was linked with their purchase records, ensuring that incomplete or missing details—such as absent shipping addresses—could be detected and flagged.

Additionally, product-level data integration enabled better understanding of purchasing patterns, order completeness, and customer engagement trends. Overall, the project outcomes offer a structured view of orders, customers, and products, helping ensure data accuracy, improving tracking of incomplete records, and laying the foundation for better business decision-making.

.