SQL for Data Analysis: Beginner Level II

Real-World e-Commerce Analytics

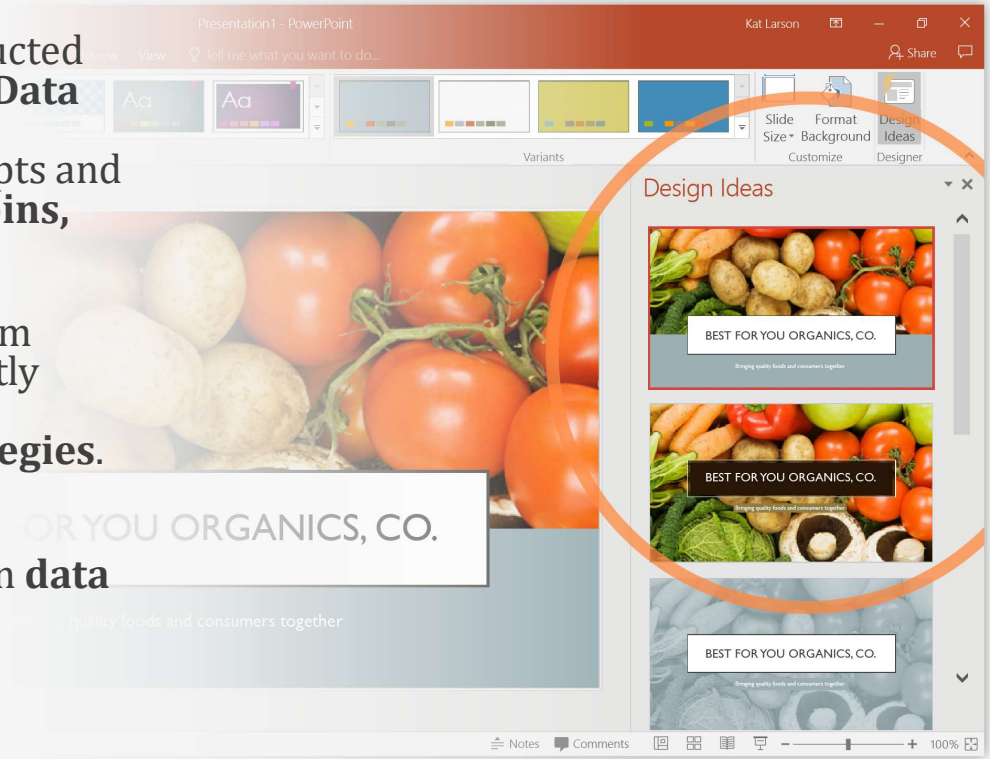
Prepared By: Shambhu Kumar Kushwaha
Date: 25/08/2025

Introduction

- ✓ This report documents SQL-based analyses conducted on an e-commerce dataset as part of the **SQL for Data Analysis: Beginner Level II** project.
  The analysis builds upon foundational SQL concepts and extends into **data cleaning, conditional logic, joins, aggregations, and reporting techniques**.

- ✓ The objective is to extract meaningful insights from transactional and customer data, which can directly support **business decision-making, sales optimization, and customer engagement strategies**.

- ✓ A total of **18 SQL tasks** were completed, each addressing a practical scenario commonly faced in **data analytics for e-commerce**.



Data Source :[Dataset]

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

Steps Performed:

1. **Data Download**
   - Collected the provided CSV files containing order and customer datasets.
   - Verified file structure (column names, delimiters, data types).

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 CustomerDetails (
CustomerID INT PRIMARY KEY,
CustomerFirstName VARCHAR(50),
CustomerLastName VARCHAR(50),
City VARCHAR(100),
Email VARCHAR(100) );
```

4

Importing CSV Data into Tables

- The CSV files were imported into MySQL tables using the MySQL Workbench "Table Data Import Wizard".
- Verified Correctness with:

```
SELECT *
FROM ecommerce_analytics.CustomerDetails;
```

5

Data Enrichment – Full Name

❖ Concatenated first and last names into a **CustomerName** column:

```
ALTER TABLE CustomerDetails
ADD COLUMN CustomerName VARCHAR(255);

SET SQL_SAFE_UPDATES = 0;
UPDATE CustomerDetails
SET CustomerName = CONCAT(CustomerFirstName, ' ', CustomerLastName);
SET SQL_SAFE_UPDATES = 1;
```

✓ **Outcome**: Database connected, data imported, customer names enriched for easier reporting.

Task 1-2: Order Report with Aliases

OBJECTIVE:
Create a readable report of orders with custom column names and customer details.

❑ Query:

```
SELECT
      e.OrderID AS "Order Number",
     e.OrderDate AS "Order Date",
     e.CustomerID AS "Customer Number",
     c.CustomerName AS "Customer Name",
     c.City AS City,
    e.TotalAmount AS "Order Value"
FROM ecommerceorders AS e
JOIN customerdetails AS c
ON e.CustomerID = c.CustomerID;
```

❑ Result:

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

Order Number	Order Date	Customer Number	Customer Name	City	Order Value
ORD0000000	2024-07-20	CUST0050	Caroline Chambers	West Corey	49.18
ORD0000001	2024-02-07	CUST0062	Matthew Smith	North Jenniferfurt	578.34
ORD0000002	2024-09-19	CUST0018	Bobby Flores	Rodriguezside	645.88
ORD0000003	2024-04-27	CUST0019	Tasha Rodriguez	Jonesberg	380.28
ORD0000004	2024-10-07	CUST0061	Michelle Hughes	Kellerstad	142.22
ORD0000005	2024-05-11	CUST0057	Patrick Rogers	North Davidborough	165.42
ORD0000006	2024-08-31	CUST0012	Kimberly Smith	Chadland	461.20

Task 1-3: Calculate Discounted Order Value

OBJECTIVE:
Apply a 10% discount to all orders and display adjusted totals.

- Query:

```
SELECT
    OrderID,
    TotalAmount,
    (TotalAmount * 0.9) AS TotalAmount_After_Discount
FROM ecommerceorders;
```

- Result:

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

OrderID	TotalAmount	TotalAmount_After_Discount
ORD0000000	49.18	44.262
ORD0000001	578.34	520.506
ORD0000002	645.88	581.292
ORD0000003	380.28	342.252
ORD0000004	142.22	127.998
ORD0000005	165.42	148.878
ORD0000006	461.20	415.080
ORD0000007	126.56	113.904

➢ **Outcome**: Report generated with both **original** and **discounted totals** per order.

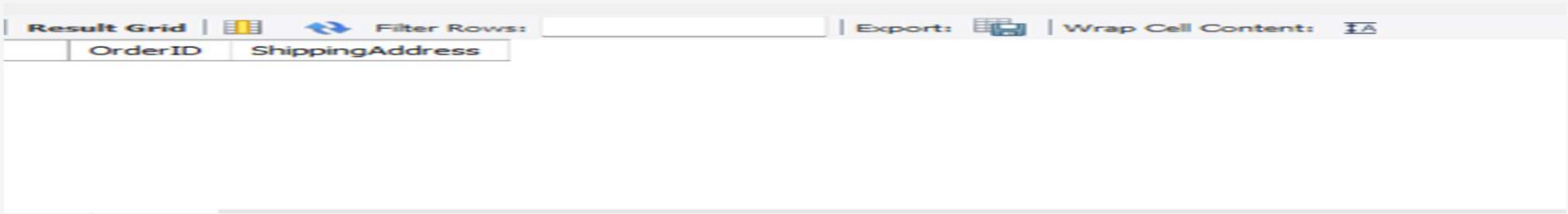
Task 1-4: Identify Missing Shipping Addresses

OBJECTIVE:
Find orders with incomplete shipping details. Replace with “NULL” values with “Pending”.

- Query:

```
SELECT
  OrderID,
  IFNULL(ShippingAddress, 'Pending') AS ShippingAddress
FROM ecommerceorders
WHERE ShippingAddress IS NULL;
```

- Result:



➢ **Outcome**: Orders missing addresses are clearly flagged with “Pending”.

Task 1-5:Classify Orders by Value

OBJECTIVE:
Categorize orders into **High Value** and **Regular** using a CASE statement.

- Query:

```
SELECT
    OrderID,
    TotalAmount,
   CASE
        WHEN TotalAmount >= 500 THEN 'High Value'
  ELSE 'Regular'
 END AS "Order Type"
FROM ecommerceorders;
```

- Result:

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

OrderID	TotalAmount	Order Type
ORD0000000	49.18	Regular
ORD0000001	578.34	High Value
ORD0000002	645.88	High Value
ORD0000003	380.28	Regular
ORD0000004	142.22	Regular
ORD0000005	165.42	Regular
ORD0000006	461.20	Regular

➢ **Outcome**: High-value orders (≥ $500) flagged for priority.

Task 1-6: Extract Month from Order Date

OBJECTIVE:
Analyze sales by month to detect seasonality.

- Query:

```
SELECT
    OrderID,
    OrderDate,
    MONTH(OrderDate) AS "Order Month"
FROM ecommerceorders;,
```

- Result:

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

OrderID	OrderDate	Order Month
ORD0000000	2024-07-20	7
ORD0000001	2024-02-07	2
ORD0000002	2024-09-19	9
ORD0000003	2024-04-27	4
ORD0000004	2024-10-07	10
ORD0000005	2024-05-11	5
ORD0000006	2024-08-31	8
ORD0000007	2024-06-23	6

➢ **Outcome**: Orders grouped by month for seasonal analysis.

Task 1-7: Unified View of Delivered & Shipped Orders

OBJECTIVE:
Combine Orders with statuses “Delivered” AND “Shipped”.

- Query:

```
SELECT OrderID, OrderDate, OrderStatus
FROM ecommerceorders
WHERE OrderStatus = "Delivered"
UNION
SELECT OrderID, OrderDate, OrderStatus
FROM ecommerceorders
WHERE OrderStatus = "Shipped";
```

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

OrderID	OrderDate	OrderStatus
ORD0000001	2024-02-07	Delivered
ORD0000004	2024-10-07	Delivered
ORD0000006	2024-08-31	Delivered
ORD0000007	2024-06-23	Delivered
ORD0000008	2023-12-06	Delivered
ORD0000011	2024-09-20	Delivered
ORD0000012	2024-05-21	Delivered
ORD0000014	2023-12-15	Delivered

➢ **Outcome**: Single consolidated view of **active + completed** orders.

Task 1-8: Top 10 Orders by Quantity

OBJECTIVE:
Retrive the largest 10 orders by quantity.

- Query:

```
SELECT
    OrderID,
    OrderDate,
    Quantity,
    TotalAmount
FROM ecommerceorders
ORDER BY Quantity DESC, OrderDate DESC
LIMIT 10;
```

- Result:

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

OrderID	OrderDate	Quantity	TotalAmount
ORD0000776	2024-10-04	10	676.10
ORD0000604	2024-09-25	10	1185.80
ORD0000577	2024-07-14	10	1694.00
ORD0000517	2024-06-30	10	788.50
ORD0000808	2024-06-22	10	689.80
ORD0000661	2024-06-21	10	1293.50
ORD0000941	2024-06-06	10	1323.90
ORD0000288	2024-04-25	10	557.20

- **Outcome**: Bulk orders identified, sorted by recency.

Task 1-9: High-Revenue Product Categories

OBJECTIVE:
Identify Categories with total sales >$10,000.

- Query:

```
WITH CTE1 AS  (
        SELECT Category,
            SUM(TotalAmount) as Total_sales_per_category
         FROM ecommerceorders
        GROUP BY category)
SELECT  Category, Total_sales_per_category
FROM CTE1
HAVING Total_sales_per_category >10000;
```

- Result:

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

Category	Total_sales_per_category
Books	43557.00
Home Appliances	56219.57
Electronics	50938.25
Beauty	55611.11
Toys	52916.26
Clothing	47618.23

➢ **Outcome**: Focused view of **top-performing categories**.

Task 1-10: Orders by Month

OBJECTIVE:
Count Orders Per Month.

- Query:

```
SELECT
    DATE_FORMAT(OrderDate, '%Y-%m') AS Order_Month,
    COUNT(OrderID) AS Total_Orders
FROM ecommerceorders
GROUP BY DATE_FORMAT(OrderDate, '%Y-%m')
ORDER BY Order_Month;
```

Result:

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

Order_Month	Total_Orders
2023-11	43
2023-12	87
2024-01	95
2024-02	95
2024-03	62
2024-04	90
2024-05	78
2024-06	80

➢ **Outcome**: Monthly order trend chart ready for visualization.

Task 1-11: Average Order Value by Payment Method

OBJECTIVE:
Analyze Customer spending across payment types.

❑ Query:

```
SELECT
    PaymentMethod,
    AVG(TotalAmount) AS "Average Order Value"
FROM ecommerceorders
GROUP BY PaymentMethod;
```

Result:

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

PaymentMethod	average order value
Credit Card	315.075895
PayPal	309.353301
Cash	281.410838

➢ **Outcome**: Monthly order trend chart ready for visualization.

Task 1-12: Orders by Status

OBJECTIVE:
Track the distribution of orders statuses.

- Query:

```
SELECT
    OrderStatus,
    COUNT(OrderStatus) AS total_order
FROM ecommerceorders
GROUP BY OrderStatus
ORDER BY total_order DESC;
```

Result:

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

OrderStatus	total_order
Delivered	696
Shipped	198
Canceled	106

- **Outcome**: Overview of order fulfillment stages.

Task 1-13: Highest Quantity Per Customer

OBJECTIVE:
Find each customer's largest order.

- Query:

```
SELECT
    e.CustomerID,
    e.OrderID,
    e.Quantity
FROM ecommerceorders e
WHERE e.Quantity = (
                    SELECT MAX(Quantity)
                    FROM ecommerceorders
                    WHERE CustomerID = e.CustomerID);
```

Result:

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

CustomerID	OrderID	Quantity
CUST0043	ORD0000007	4
CUST0029	ORD0000008	5
CUST0063	ORD0000009	3
CUST0003	ORD0000020	4
CUST0073	ORD0000024	8
CUST0003	ORD0000038	4
CUST0083	ORD0000039	6
CUST0020	ORD0000061	8

- **Outcome**: Highlights customers with bulk order tendencies.

Task 1-14: Annual Revenue

OBJECTIVE:
Summarize revenue per year.

❑ Query:

```
SELECT
    YEAR(OrderDate) AS order_year,
    SUM(TotalAmount) AS total_revenue
FROM ecommerceorders
GROUP BY YEAR(OrderDate)
ORDER BY order_year;
```

Result:

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

order_year	total_revenue
2023	40909.57
2024	265950.85

➢ **Outcome**: Year-over-year growth trends identified.

Task 1-15: Unique Customers by City

OBJECTIVE:
Count distinct customers across cities.

- Query:

```
SELECT
    c.City,
    COUNT(DISTINCT e.CustomerID) AS total_customers
 FROM ecommerceorders e
 JOIN customerdetails c
ON e.CustomerID = c.CustomerID
GROUP BY c.City
 ORDER BY total_customers DESC;
```

Result:

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

City	total_customers
Lake Michael	8
East Seanstad	2
Austintown	1
Ayalaview	1
Chadland	1
Chelsealand	1
Davidbury	1
Davidmouth	1

➢ **Outcome**: Geographic distribution of customer base.

Task 1-16: Top 5 Customers by Spending

OBJECTIVE:
Identify **VIP Customers** for loyalty programs.

- Query:

```
SELECT
    CustomerID,
     SUM(TotalAmount) AS total_spent
FROM ecommerceorders
GROUP BY CustomerID
ORDER BY total_spent DESC
LIMIT 5;
```

Result:

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

CustomerID	total_spent
CUST0033	6243.43
CUST0085	6107.14
CUST0073	5802.76
CUST0051	5417.26
CUST0077	5407.44

➢ **Outcome**: Top 5 customers by spending extracted.

CONCLUSION

This analysis demonstrates the power of SQL in extracting actionable insights from eCommerce data.
Key outcomes include:

- ❑ Better visibility into orders and customers
- ❑ Insights into revenue, high-value transactions, and promotions
- ❑ Identification of VIP customers & bulk buyers
- ❑ Trend analysis for monthly & yearly performance
- ❑ Regional and categorical insights for marketing and inventory planning

By leveraging SQL techniques such as joins, CASE statements, aggregations, and unions, the dataset has been transformed into a decision-support tool for business growth.

.