KSR Datavizon

MySQL Real-World Assignment: E-Commerce Sales & Customer Analytics

---

Functions Covered: String • Numeric • Date/Time • Aggregate • Window |

Goal: Solve a close-to-real e-commerce analytics problem using MySQL built-in functions in one project.

Index

1. Business Context

You are a data analyst for an e-commerce company. The business wants clean customer and order data, monthly revenue reporting, customer segmentation (top spenders, repeat buyers), order trend analysis (MoM growth, customer purchase patterns), and insights on best products and best cities.

You must solve using MySQL built-in functions in the following categories:

- String functions (e.g., CONCAT, TRIM, LOWER/UPPER, SUBSTRING, REPLACE, LENGTH, LEFT/RIGHT)
- Numeric functions (e.g., ROUND, CEIL, FLOOR, ABS, MOD, POWER)
- Date/Time functions (e.g., DATE, YEAR, MONTH, DATE_FORMAT, DATEDIFF, TIMESTAMPDIFF, DATE_ADD)
- Aggregate functions (e.g., COUNT, SUM, AVG, MIN, MAX, COUNT(DISTINCT), GROUP_CONCAT)
- Window functions (e.g., ROW_NUMBER, RANK, DENSE_RANK, NTILE, LAG, LEAD, FIRST_VALUE, LAST_VALUE)

2. Database Setup (Schema)

Run the following SQL to create database and tables:

```
CREATE DATABASE IF NOT EXISTS ecommerce_analytics;
USE ecommerce_analytics;

DROP TABLE IF EXISTS order_items;
DROP TABLE IF EXISTS orders;
DROP TABLE IF EXISTS products;
DROP TABLE IF EXISTS customers;

CREATE TABLE customers (
  customer_id INT PRIMARY KEY,
  full_name VARCHAR(100),
  email VARCHAR(120),
  phone VARCHAR(20),
  city VARCHAR(50),
  created_at DATETIME
);

CREATE TABLE products (
  product_id INT PRIMARY KEY,
  product_name VARCHAR(120),
  category VARCHAR(60),
  mrp DECIMAL(10,2),
  cost DECIMAL(10,2)
);

CREATE TABLE orders (
  order_id INT PRIMARY KEY,
  customer_id INT,
  order_date DATETIME,
  status VARCHAR(20),          -- Delivered / Cancelled / Returned
```

```
  payment_mode VARCHAR(20),   -- UPI / Card / NetBanking / COD
  coupon_code VARCHAR(30),
  shipping_city VARCHAR(50),
  FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);

CREATE TABLE order_items (
  order_item_id INT PRIMARY KEY,
  order_id INT,
  product_id INT,
  quantity INT,
  selling_price DECIMAL(10,2),    -- per unit selling price after product-level
discounts
  discount_pct DECIMAL(5,2),      -- e.g., 10.00 means 10%
  FOREIGN KEY (order_id) REFERENCES orders(order_id),
  FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```

3. Sample Data Inserts

Insert this sample data (students can add more rows if desired):

```
INSERT INTO customers VALUES
(1,'  koti konda  ','koti.konda@gmail.com','+91-98765 43210','Hyderabad','2025-08-
01 10:20:00'),
(2,'Jaswitha Konda','JASWITHA@GMAIL.COM','9876543211','Hyderabad','2025-09-11
11:00:00'),
(3,'Daniwik Konda','daniwik@gmail.com','98765-11111','Bengaluru','2025-10-05
09:15:00'),
(4,'Ayesha Khan','ayesha.khan@gmail.com','+91 99999 88888','Pune','2025-11-20
14:05:00'),
(5,'Ravi Teja','ravi.teja@outlook.com','88888 77777','Chennai','2025-12-15
18:45:00'),
(6,'Sita Rao','sita.rao@gmail.com','77777 66666','Hyderabad','2026-01-07
08:40:00');

INSERT INTO products VALUES
(101,'Noise Smartwatch X2','Wearables',2999,1600),
(102,'Boat Earbuds Airdopes','Audio',1999,1100),
(103,'Laptop Backpack 25L','Accessories',1499,700),
(104,'Mixer Grinder 750W','Home',4999,3500),
(105,'Running Shoes Pro','Fashion',3999,2300);

INSERT INTO orders VALUES
(5001,1,'2026-01-02 10:10:00','Delivered','UPI','NEW10','Hyderabad'),
(5002,1,'2026-01-18 16:35:00','Delivered','Card',NULL,'Hyderabad'),
(5003,2,'2026-01-10 12:00:00','Cancelled','COD','NEW10','Hyderabad'),
(5004,3,'2026-01-22 19:20:00','Delivered','UPI','FEST20','Bengaluru'),
(5005,4,'2026-02-03 09:05:00','Delivered','NetBanking',NULL,'Pune'),
(5006,5,'2026-02-05 13:45:00','Returned','Card','FEST20','Chennai'),
(5007,6,'2026-02-10 20:10:00','Delivered','UPI',NULL,'Hyderabad'),
```

```
(5008,3,'2026-02-11 11:30:00','Delivered','UPI','NEW10','Bengaluru');

INSERT INTO order_items VALUES
(9001,5001,101,1,2699,10),
(9002,5001,103,1,1299,0),
(9003,5002,102,2,1799,10),
(9004,5003,105,1,3999,0),
(9005,5004,104,1,4499,10),
(9006,5005,101,1,2799,7),
(9007,5005,102,1,1899,5),
(9008,5006,103,2,1199,20),
(9009,5007,105,1,3599,10),
(9010,5008,102,1,1699,15);
```

4. Task List (A–E)

A) Data Cleaning & String Functions (10 marks)

1. Create a query that outputs: customer_id, cleaned name (TRIM), email in lowercase (LOWER), and phone digits only (use REPLACE multiple times).
2. Create a Customer Tag: CUST-<customer_id>-<CITY> where CITY is uppercase (CONCAT + UPPER). Example: CUST-1-HYDERABAD.

B) Numeric Functions: Revenue, Discount, Profit (15 marks)

3. For each order_item, compute gross_amount, discount_amount, net_amount using ROUND(..., 2).
4. Compute profit = net_amount − (quantity * cost). Also create loss_abs = ABS(profit) only when profit is negative.

C) Date & Time Analysis (15 marks)

5. For each order: extract day/month/year and month label using DATE_FORMAT(order_date, '%b-%Y').
6. Customer tenure in days: DATEDIFF(CURDATE(), DATE(created_at)).
7. Find customers who placed an order within 7 days of signup using TIMESTAMPDIFF(DAY, created_at, order_date) <= 7.

D) Aggregate Functions (20 marks)

8. Monthly KPI report (Delivered only): month, total orders, total revenue (sum of net_amount), average order value.
9. Top 3 cities by revenue (Delivered only) including unique customers: COUNT(DISTINCT customer_id).
10. Category performance: category, total qty, total revenue, total profit; filter categories with revenue > 5000 using HAVING.

E) Window Functions (40 marks)

11. Top 3 customers by Delivered revenue using DENSE_RANK() over revenue desc.
12. For each city, rank products by revenue using RANK() OVER (PARTITION BY shipping_city ORDER BY revenue DESC).

13. Segment customers into 4 tiers by Delivered revenue using NTILE(4). Map tiers: 1=Platinum, 2=Gold, 3=Silver, 4=Bronze.
14. For each customer: show order_date and net_order_value, previous order value using LAG, difference from previous order, next order date using LEAD.
15. For each customer: first order date/value and last order date/value using FIRST_VALUE and LAST_VALUE with full frame: ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING.

5. Submission Deliverables

- 1) One .sql file containing all queries with task-wise comments (A, B, C, D, E).
- 2) Output proof: screenshots or exported results for: Monthly KPI report, Top customers (ranked), Customer tiers, LAG/LEAD trend output.
- 3) (Optional) Create views for key reports (monthly_kpi_view, customer_tiers_view).

6. Rubric (100 marks)

Section	What we expect	Marks
A) String functions	Cleaning name/email/phone + customer tag	10
B) Numeric functions	Gross/discount/net, profit & loss handling	15
C) Date/Time functions	Month label, tenure, 7-day signup order logic	15
D) Aggregate functions	Monthly KPIs, top cities, category report + HAVING	20
E) Window functions	Rankings, NTILE, LAG/LEAD, FIRST/LAST with frames	35
Quality	Formatting, comments, correctness, readability	5

7. Interview Questions (with answers)

Q: Difference between GROUP BY and Window functions?

A: GROUP BY reduces rows by aggregating. Window functions compute over a set of rows but keep each row in the output.

Q: Why can LAST_VALUE() give unexpected results? How to fix it?

A: Default window frame may end at the current row. Use a full frame: ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING.

Q: RANK vs DENSE_RANK vs ROW_NUMBER?

A: ROW_NUMBER gives unique sequence. RANK leaves gaps on ties. DENSE_RANK does not leave gaps.

Q: What happens in NTILE(4) if rows are not divisible by 4?

A: The first buckets get one extra row until all rows are assigned.

Q: Why use HAVING instead of WHERE?

A: WHERE filters before aggregation. HAVING filters after GROUP BY/aggregation.

Q: DATEDIFF vs TIMESTAMPDIFF?

A: DATEDIFF returns day difference between dates. TIMESTAMPDIFF supports different units (DAY, MONTH, YEAR, etc.).

Q: How to remove special characters from phone using built-in functions?

A: Use nested REPLACE calls to remove '+', spaces, hyphens and country prefix if needed.

Q: How to calculate Month-over-Month growth using window functions?

A: Aggregate monthly revenue first, then use LAG(revenue) over month order and compute (revenue - prev)/prev.