
✅ Step 1: Identify Business Areas in the ERD

The system includes the following areas:

- **Customers**
- **Orders**
- **Menu & Menu Items**
- **Order Items**
- **Payments**
- **Staff**
- **Reservations**
- **Restaurant Tables**

📊 Step 2: KPIs & Metrics

📦 Sales & Revenue

- **Total Revenue** = Sum of amountPaid from payments
- **Average Order Value** = Total Revenue / Total Orders
- **Number of Orders per Day/Week/Month**
- **Most Popular Menu Items** = Count of itemID in orderItems, grouped by itemID
- **Payment Method Breakdown** = Count of orders per paymentMethod

👥 Customers

- **Total Customers**
- **New Customers This Month** = Count of customers where createdAt is within current month
- **Top Spending Customers** = Group by customerID and sum amountPaid

🍽️ Menu Performance

- **Top Selling Menu Items** = Count or sum of quantity per itemID

- **Least Selling Items**
- **Average Menu Item Price**
- **Menu Categories Performance** (if menu refers to categories)

👨‍🍳 **Staff Management**

- **Staff Count**
- **Roles Breakdown** = Count by role
- **Average Salary**
- **New Hires This Month**

🪑 **Tables & Reservations**

- **Table Utilization Rate** = (Number of Orders / Number of Tables) over time
- **Most Frequently Reserved Tables**
- **Reservation-to-Order Conversion Rate**

❓ **Step 3: Business Questions You Can Answer**

💰 **Revenue/Sales**

- What is the **total revenue** for the past month?
- What are the **most profitable items** on the menu?
- How many **orders are placed per day/week/month**?
- What is the **trend in revenue** over the last 6 months?

👤 **Customers**

- Who are our **top 10 customers** by spend?
- How many **new customers** joined this week/month?
- How many **repeat customers** do we have?

🍔 **Menu**

- What are the **most/least ordered menu items**?

- Which **menu items generate the most revenue**?
- Is any item consistently **underperforming**?

🧾 **Payments**

- What **percentage of payments** were made with each method (cash, card, etc.)?
- How many **orders were not paid in full**?

👥 **Staff**

- What is the **average staff salary**?
- How many staff were **hired this year**?
- Which roles are **most common**?

🗓️ **Reservations**

- What is the **average reservation rate**?
- Are there any **peak times/days** for reservations?
- What is the **conversion rate** of reservations to orders?

🪑 **Tables**

- Which tables are **most frequently used**?
- Are any tables **under-utilized**?
- What is the **average seating capacity per reservation**?

🛠️ **Optional: Advanced KPIs**

- **Customer Lifetime Value (CLV)**
- **Customer Retention Rate**
- **Revenue per Available Table (RevPATT)** = Total Revenue / Total Available Table Hours
- **Order Accuracy Rate** (if you add a review/feedback system)
- **Staff Productivity** = Orders Handled per Staff

✅ **Step 1: Structuring the Implementation**

1. **Views** – Precompute common aggregations for easier querying.
2. **Joins & Subqueries** – Combine data from multiple tables to generate insights.
3. **Functions** – Define reusable logic for metrics like total revenue.
4. **Stored Procedures** – Automate reports for daily, weekly, or monthly data.
5. **Triggers** – Automatically update records when an event occurs (e.g., update order status when payment is made).
6. **Window Functions** – Calculate running totals, rankings, and comparisons over partitions.
7. **Indexes** – Optimize query performance on frequently searched columns.

📌 **Step 2: Writing Queries for KPIs & Business Questions**

1️⃣View: Total Revenue & Average Order Value

```
CREATE VIEW RevenueSummary AS
SELECT
  SUM(amountPaid) AS total_revenue,
  AVG(amountPaid) AS avg_order_value,
  COUNT(orderID) AS total_orders
FROM payments;
```

🔹 **Usage:** SELECT * FROM RevenueSummary;

2️⃣Joins & Subquery: Most Popular Menu Items

```
SELECT mi.name, SUM(oi.quantity) AS total_sold
FROM orderItems oi
JOIN menuItems mi ON oi.itemID = mi.itemID
```

```
GROUP BY mi.name
ORDER BY total_sold DESC
LIMIT 5;
```

- **Finds top 5 best-selling items.**

3️⃣ Window Function: Rank Customers by Spending

```
SELECT c.customerID, c.name, SUM(p.amountPaid) AS total_spent,
    RANK() OVER (ORDER BY SUM(p.amountPaid) DESC) AS spending_rank
FROM customers c
JOIN orders o ON c.customerID = o.customerID
JOIN payments p ON o.orderID = p.orderID
GROUP BY c.customerID, c.name;
```

- **Ranks customers based on total spending.**

4️⃣ Stored Procedure: Monthly Sales Report

```
DELIMITER //
CREATE PROCEDURE MonthlySalesReport(IN reportMonth INT, IN reportYear INT)
BEGIN
  SELECT
    DATE_FORMAT(p.paymentDate, '%Y-%m') AS month,
    SUM(p.amountPaid) AS total_revenue,
    COUNT(o.orderID) AS total_orders
  FROM payments p
  JOIN orders o ON p.orderID = o.orderID
  WHERE MONTH(p.paymentDate) = reportMonth AND YEAR(p.paymentDate) = reportYear
  GROUP BY month;
```

```
END //
```

- **Call with:** CALL MonthlySalesReport(3, 2025); → Generates report for March 2025.

5️⃣ Trigger: Auto-update Order Status When Paid

```
DELIMITER //
CREATE TRIGGER UpdateOrderStatus
AFTER INSERT ON payments
FOR EACH ROW
BEGIN
  UPDATE orders
  SET status = 'Paid'
  WHERE orderID = NEW.orderID;
END //
DELIMITER ;
```

- **Automatically updates order status when payment is received.**

6️⃣ Index: Speed Up Customer Search

```
CREATE INDEX idx_customer_email ON customers(email);
```

- **Optimizes queries filtering by email, e.g.:**

```
SELECT * FROM customers WHERE email = 'customer@example.com';
```

📊 **Reports & Real-Life Business Insights**

✅ **Daily Orders & Revenue**

```
SELECT DATE(o.orderDate) AS order_day, COUNT(o.orderID) AS total_orders,
    SUM(p.amountPaid) AS revenue
```

```
FROM orders o
LEFT JOIN payments p ON o.orderID = p.orderID
GROUP BY order_day
ORDER BY order_day DESC;
```

✅ **Reservation Conversion Rate**

```
SELECT COUNT(r.reservationID) AS total_reservations,
    COUNT(o.orderID) AS converted_orders,
    (COUNT(o.orderID) * 100.0 / COUNT(r.reservationID)) AS conversion_rate
FROM reservations r
LEFT JOIN orders o ON r.customerID = o.customerID AND r.tableID = o.tableID;
```

❓ **Business Questions to Answer**

- What is the total salary expenditure for each role?
- How many staff members are working in each role?
- What is the average salary for each role?
- Who are the current managers in the company?
- Who are the recently hired staff members?
- What is the total revenue generated from all orders?
- What are the revenue trends over time (daily/monthly)?
- Which menu item is the most ordered?
- How many customers have placed orders recently?
- Who are the highest-spending customers?

✅ **KPIs (Key Performance Indicators)**

These are the core metrics your system should track:

1. **Total salaries per role**
2. **Number of staff members per role**
3. **Average salary per role**
4. **List of all managers**
5. **Recently hired staff**
6. **Total revenue from orders**
7. **Daily or monthly revenue trends**
8. **Most ordered items**
9. **Number of active customers**
10. **Top 5 customers by spending**

Key Performance Indicators (KPIs) and Business Questions Addressed by SQL Techniques

In today's data-driven business environment, KPIs are critical for tracking and evaluating the success of various business processes. By leveraging SQL techniques such as views, joins, subqueries, functions, stored procedures, triggers, window functions, and indexes, businesses can derive actionable insights from data, answering key business questions that drive decisions and strategy. Below, we outline important KPIs and business questions, highlighting the relevant SQL techniques used to address them.

---

1. Total Sales Report (By Date)

KPI: Daily sales performance

Business Question: What is the total sales revenue generated each day?

SQL Techniques:

- **Views**: Create an aggregated view that simplifies the retrieval of daily sales data.
- **Joins**: Combine relevant sales tables to gather necessary data for generating the report.

Role-Based Access:

- **Managers, Finance, Executives**: These roles should have access to daily sales data to monitor and track performance.

---

2. Total Orders Report (By Date)

KPI: Daily order count

Business Question: How many orders were processed each day?

SQL Techniques:

- **Views**: Create a view that summarizes orders by date.
- **Joins**: Combine data from order-related tables to get relevant order details.

Role-Based Access:

- **Operations, Managers**: These roles should be able to access data on order volumes to ensure smooth operational processes.

---

3. Monthly Revenue Trend

KPI: Monthly revenue performance

Business Question: What is the trend of revenue over the months?

SQL Techniques:

- **Views**: Summarize revenue data by month for easy tracking.
- **Window Functions**: Can be used to identify trends and running totals over time.

Role-Based Access:

- **Finance, Managers, Executives**: These roles should have access to monitor monthly revenue trends for better financial planning and forecasting.

---

4. Top-Selling Products

KPI: Product sales volume

Business Question: Which products are the top sellers in terms of quantity sold?

SQL Techniques:

- **Views**: Summarize product sales data by quantity sold.
- **Joins**: Combine product and order data to identify top-performing products.

Role-Based Access:

- **Sales, Managers**: These roles need access to identify best-selling products and adjust inventory or marketing strategies accordingly.

---

5. Customer Spend Report

KPI: Total spend by customer

Business Question: Which customers have spent the most?

SQL Techniques:

- **Views**: Create a view that aggregates customer spending.

- **Joins**: Combine data from customer, order, and payment tables to calculate total spending.

Role-Based Access:

- **Finance, Managers**: These roles need to analyze customer spending patterns to improve customer relationships and develop targeted marketing strategies.

---

6. Staff Salary by Role

KPI: Total salary expenses by role

Business Question: How much is spent on salaries per role in the organization?

SQL Techniques:

- **Views**: Summarize salary expenses by role to track compensation costs.
- **Group By**: Aggregate salary data by role to understand total compensation per position.

Role-Based Access:

- **HR, Executives**: These roles need access to salary-related information for budgeting, payroll, and compensation management.

---

7. Recently Hired Staff

KPI: New hires

Business Question: Who are the staff members hired in the last month?

SQL Techniques:

- **Views**: Create a view showing employees hired within a specific time frame.
- **Date Functions**: Filter staff data by hire date to identify recent hires.

Role-Based Access:

- **HR, Executives**: These roles should have access to staff hiring data for planning recruitment, onboarding, and talent management.

Role-Based Views for Data Access

To ensure that data is accessed securely and by the appropriate individuals, **role-based views** are used. These views restrict access to specific data based on the user's role, ensuring that sensitive or confidential information is only accessible to those with the appropriate permissions.

Example of Role-Based Access Control:

- **Sales Report Views**: Access granted to Managers, Finance, and Executives.
- **Order Report Views**: Access granted to Operations and Managers.
- **Revenue Trend Views**: Access granted to Finance, Managers, and Executives.
- **Top-Selling Products Views**: Access granted to Sales and Managers.
- **Customer Spend Views**: Access granted to Finance and Managers.
- **Staff Salary Views**: Access granted to HR and Executives.
- **Manager List Views**: Access granted to Managers and Executives.
- **Recently Hired Staff Views**: Access granted to HR and Executives.

By setting up **role-based views**, businesses can protect sensitive data while still providing the necessary access to key stakeholders for decision-making. This allows for secure, yet effective, data usage throughout the organization.
