Analytics
Career Connect

# ZERO TO ADVANCE IN SQL


## Wasim Patwari

Empowering Learners in Computer Science & Data Analytics

WhatsApp:91- 9607157409

# DAY 01

## INTRODUCTION TO SQL AND RELATIONAL DATABASES

- Understand what SQL is and its role in managing databases.
- Learn about relational databases and their components.
- Study basic SQL commands: SELECT, INSERT, UPDATE, DELETE.
- Practice writing simple queries and retrieving data from a database

### Example:

Write a SQL query to retrieve all the columns from the "customers" table.

### Practice Questions:

1. Write a SQL query to insert a new record into the "employees" table.
2. Write a SQL query to update the "quantity" column of the "products" table to 10 where the "product_id" is 5.
3. Write a SQL query to delete all records from the "orders" table where the "status" is 'cancelled'.

### Dataset:

**Click Here** to View Dataset in Excel Sheet

# DAY 02

## FILTERING AND SORTING DATA

- Learn about the WHERE clause and its usage for filtering data.
- Study the ORDER BY clause for sorting data.
- Practice writing queries with filtering and sorting

### Example:

Write a SQL query to retrieve all the columns from the "employees" table where the "salary" is greater than 50000.

### Practice Questions:

1. Write a SQL query to retrieve all the columns from the "products" table where the "category" is 'Electronics' and the "price" is less than 1000.
2. Write a SQL query to retrieve the names of all customers from the "customers" table in alphabetical order.
3. Write a SQL query to retrieve the total number of orders from the "orders" table.

### Dataset:

**Click Here** to View Dataset in Excel Sheet

# DAY 03

## JOINING TABLES

- Understand the concept of joining tables.
- Learn about different types of joins: INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL JOIN.
- Practice writing queries that involve joining multiple tables.

### Example:

Write a SQL query to retrieve the customer name and order date from the "customers" and "orders" tables, joining them on the "customer_id" column.

### Practice Questions:

1. Write a SQL query to retrieve the product name, category, and supplier name from the "products", "categories", and "suppliers" tables, joining them on the appropriate columns.
2. Write a SQL query to retrieve the employee name and department name from the "employees" and "departments" tables, joining them on the "department_id" column.
3. Write a SQL query to retrieve the customer name and order amount from the "customers" and "orders" tables, joining them on the "customer_id" column, and only including orders with amounts greater than 1000.

### Dataset:

**Click Here** to View Dataset in Excel Sheet

# DAY 04

## AGGREGATING DATA

- Study aggregate functions like COUNT, SUM, AVG, MAX, and MIN.
- Learn to use the GROUP BY clause to group data.
- Practice writing queries that involve aggregating data.

### Example:

Write a SQL query to retrieve the total number of orders for each customer from the "orders" table.

### Practice Questions:

1. Write a SQL query to retrieve the average price of products in each category from the "products" table.
2. Write a SQL query to retrieve the maximum salary for each department from the "employees" table.
3. Write a SQL query to retrieve the total revenue generated by each customer from the "orders" and "order_items" tables.

### Dataset:

**Click Here** to View Dataset in Excel Sheet

# DAY 05

## DATA MANIPULATION

- Study advanced SQL commands: UPDATE, DELETE, and INSERT INTO SELECT.
- Understand how to modify existing data in a database.
- Practice writing queries for data manipulation.

### Example:

Write a SQL query to update the "quantity" column of the "products" table to 20 for all products with a price greater than 100.

### Practice Questions:

1. Write a SQL query to delete all records from the "customers" table where the "last_login_date" is older than 1 year.
2. Write a SQL query to insert new records into the "employees" table, selecting data from the "temp_employees" table.
3. Write a SQL query to update the "discount" column of the "orders" table by increasing it by 5% for all orders placed before a specific date.

### Dataset:

**Click Here** to View Dataset in Excel Sheet

# DAY 06

## ADVANCED FILTERING AND SORTING

- Learn about advanced filtering techniques: LIKE, IN, BETWEEN, and NULL.
- Study complex sorting options using multiple columns.
- Practice writing queries with advanced filtering and sorting.

### Example:

Write a SQL query to retrieve all the customers whose names start with 'J' and have a city containing 'York'.

### Practice Questions:

1. Write a SQL query to retrieve all the products with a price either above 1000 or below 500.
2. Write a SQL query to retrieve the employees who were hired between a specific date range.
3. Write a SQL query to retrieve all the customers who do not have a phone number specified in the database.

DATASET

### Dataset:

**Click Here** to View Dataset in Excel Sheet

# DAY 07

## WORKING WITH FUNCTIONS

- Study various SQL functions: String functions, Date functions, and Numeric functions.
- Learn how to use these functions in your queries.
- Practice writing queries that involve SQL functions

### Example:

Write a SQL query to retrieve the length of the product names from the "products" table.

### Practice Questions:

1. Write a SQL query to retrieve the current date and time.
2. Write a SQL query to retrieve the uppercase names of all the employees from the "employees" table.
3. Write a SQL query to retrieve the average price of products after applying a 10% discount from the "products" table.

### Dataset:

**Click Here** to View Dataset in Excel Sheet

# DAY 08

## SUBQUERIES

- Understand the concept of subqueries and their usage.
- Learn to write subqueries in different parts of a SQL statement.
- Practice writing queries that include subqueries.

### Example:

Write a SQL query to retrieve all the products with a price higher than the average price of all products

### Practice Questions:

1. Write a SQL query to retrieve the names of all employees who have a salary higher than the maximum salary of the 'Sales' department.
2. Write a SQL query to retrieve all the customers who have placed an order after the latest order date for a specific product.
3. Write a SQL query to retrieve all the products that belong to categories with more than 10 products.

### Dataset:

**Click Here** to View Dataset in Excel Sheet

# DAY 09

## VIEWS AND INDEXES

- Study views and their importance in database design.
- Learn to create and use views in SQL.
- Understand indexes and their role in optimizing query performance.
- Practice creating views and indexes.

### Example:

Create a view named "high_salary_employees" that retrieves all the employees with a salary greater than 50000 from the "employees" table

### Practice Questions:

1. Create a view named "order_summary" that retrieves the total order amount and the number of orders for each customer from the "orders" table.
2. Create an index on the "email" column of the "customers" table for faster searching.
3. Create a view named "product_inventory" that retrieves the product name and the available quantity for each product from the "products" and "inventory" tables.

### Dataset:

**Click Here** to View Dataset in Excel Sheet

# DAY 10

## DATA INTEGRITY AND CONSTRAINTS

- Learn about data integrity and the role of constraints.
- Understand different types of constraints: NOT NULL, UNIQUE, PRIMARY KEY, FOREIGN KEY.
- Practice creating tables with constraints.

### Example:

Create a table named "employees" with columns for employee ID, name, and email, where the employee ID is the primary key and the email must be unique.

### Practice Questions:

1. Create a table named "orders" with columns for order ID, customer ID, and order date, where the order ID is the primary key and the customer ID references the "customers" table.
2. Create a table named "products" with columns for product ID, name, and price, where the product ID is the primary key and the price cannot be null.
3. Create a table named "categories" with columns for category ID and name, where the category ID is the primary key and the name must be unique.

### Dataset:

**Click Here** to View Dataset in Excel Sheet

# DAY 11

## MODIFYING TABLES

- Study commands for modifying existing tables: ALTER TABLE, DROP TABLE, and RENAME TABLE.
- Learn how to add, modify, and delete columns in a table.
- Practice modifying table structures

### Example:

Alter the "employees" table to add a new column named "address" of type VARCHAR(100).

### Practice Questions:

1. Rename the table "customer_details" to "client_details".
2. Delete the "quantity" column from the "products" table.
3. Modify the "orders" table to change the data type of the "order_date" column to DATE.

### Dataset:

**Click Here** to View Dataset in Excel Sheet

# DAY 12

## ADVANCED JOINS AND SUBQUERIES

- Study advanced join techniques: self-joins, non-equijoins, and complex join conditions.
- Learn to use correlated subqueries and EXISTS operator.
- Practice writing queries with advanced joins and subqueries.

### Example:

Write a SQL query to retrieve all employees and their respective managers from the "employees" table using a self-join.

### Practice Questions:

1. Write a SQL query to retrieve all the orders that do not have any corresponding items in the "order_items" table using a subquery.
2. Write a SQL query to retrieve all the products along with the total quantity sold for each product from the "products" and "order_items" tables using a join and subquery.
3. Write a SQL query to retrieve all the customers who have placed an order in the same month and year as their registration date.

### Dataset:

**Click Here** to View Dataset in Excel Sheet

# DAY 13

## TRANSACTIONS AND LOCKING

- Understand the concept of transactions and their importance.
- Learn about different transaction states and properties (ACID).
- Study locking and concurrency control in SQL.
- Practice writing queries with transactions.

### Example:

Write a SQL query to start a transaction, update the "inventory" table by reducing the quantity of a product, and commit the transaction.

### Practice Questions:

1. Write a SQL query to start a transaction, delete all records from the "orders" table, and roll back the transaction.
2. Write a SQL query to update the "balance" column of the "accounts" table by adding a specific amount for a specific account, ensuring the consistency of the transaction.
3. Write a SQL query to lock a specific row in the "employees" table to prevent other transactions from modifying it.

### Dataset:

**Click Here** to View Dataset in Excel Sheet

# DAY 14

## ADVANCED TOPICS

- Explore advanced topics like stored procedures, triggers, and user-defined functions.
- Learn about SQL optimization techniques.
- Practice writing queries involving advanced topics.

### Example:

Create a stored procedure named "get_customer_orders" that takes a customer ID as input and retrieves all the orders placed by that customer.

### Practice Questions:

1. Create a trigger named "update_inventory" that automatically updates the quantity in the "inventory" table when an order is placed.
2. Create a user-defined function named "calculate_discount" that takes the order total as input and returns the discount amount based on specific conditions.
3. Write a SQL query to optimize a slow-performing query by adding appropriate indexes and rewriting the query structure.

### Dataset:

**Click Here** to View Dataset in Excel Sheet

# DAY 15

## ADVANCED QUERYING

### AIM:

Understand subqueries and nested queries and their usage in SQL queries.

### Practice Questions:

- "SQL Subqueries" on w3schools.com (https://www.w3schools.com/sql/sql_subqueries.asp)
- "Subqueries in SQL: A Complete Guide" on sqlshack.com (https://www.sqlshack.com/subqueries-in-sql-a-complete-guide/)

### Example:

1. Retrieve all customers who have made at least one purchase.
2. Find the names of customers who have not made any purchases.
3. Get the order details for orders with a total quantity greater than the average quantity of all orders.

# PRACTICE QUESTIONS

1. Write a query to find the top 5 customers with the highest total order amount.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id, order_date, order_amount)

2. Retrieve the names of customers who have placed orders in the past 30 days.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id, order_date)

3. Find the products that have been ordered at least three times.
Dataset: Products (product_id, product_name), Order_Items (order_id, product_id, quantity)

4. Retrieve the order details for orders placed by customers from a specific city.
Dataset: Customers (customer_id, customer_name, city), Orders (order_id, customer_id, order_date), Order_Details (order_id, product_id, quantity)

5. Write a query to find the customers who have placed orders for products with a price greater than $100.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id, order_date), Products (product_id, product_name, price), Order_Details (order_id, product_id, quantity)

6. Get the average order amount for each customer.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id, order_date, order_amount)

7. Find the products that have never been ordered.
Dataset: Products (product_id, product_name), Order_Items (order_id, product_id, quantity)

8. Retrieve the names of customers who have placed orders on weekends (Saturday or Sunday).
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id, order_date)

9. Get the total order amount for each month.
Dataset: Orders (order_id, order_date, order_amount)

10. Write a query to find the customers who have placed orders for more than two different products.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id, order_date), Order_Items (order_id, product_id, quantity)

# DAY 16

## JOINS

### AIM:

Understand different types of joins and their applications in SQL queries.

### Practice Questions:

- "SQL Joins" on mode.com (https://mode.com/sql-tutorial/sql-joins/)
- "A Visual Explanation of SQL Joins" by Coding Horror (https://blog.codinghorror.com/a-visual-explanation-of-sql-joins/)

### Example:

1. Retrieve the order details along with the customer names for all orders.
2. Find the products and their corresponding categories.
3. Get a list of customers and their total order amounts.

# PRACTICE QUESTIONS

1. Retrieve the order details along with the customer name and product name for each order.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id, order_date), Order_Items (order_id, product_id, quantity)

2. Find the products and their corresponding suppliers' names.
Dataset: Products (product_id, product_name, supplier_id), Suppliers (supplier_id, supplier_name)

3. Get a list of customers who have never placed an order.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id)

4. Retrieve the names of customers along with the total quantity of products they ordered.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id), Order_Items (order_id, product_id, quantity)

5. Find the products that have been ordered by customers from a specific country.
Dataset: Products (product_id, product_name), Orders (order_id, customer_id), Customers (customer_id, country)

6. Get the total order amount for each customer, including those who have not placed any orders.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id, order_amount)

7. Retrieve the order details for orders placed by customers with a specific occupation.
Dataset: Customers (customer_id, customer_name, occupation), Orders (order_id, customer_id, order_date), Order_Items (order_id, product_id, quantity)

8. Find the customers who have placed orders for products with a price higher than the average price of all products.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id, order_date), Products (product_id, product_name, price), Order_Items (order_id, product_id, quantity)

9. Retrieve the names of customers along with the total number of orders they have placed.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id)

10. Get a list of products and the total quantity ordered for each product.
Dataset: Products (product_id, product_name), Order_Items (order_id, product_id, quantity)

# DAY 17

## ADVANCED FILTERING AND SORTING

### AIM:

Learn advanced filtering techniques and sorting options in SQL queries.

### Practice Questions:

- "SQL Wildcards" and "SQL Regular Expressions" on tutorialspoint.com(https://www.tutorialspoint.com/sql/sql-regular-expressions.htm)
- "Sorting Rows with NULL Values in SQL" on stackoverflow.com(https://stackoverflow.com/questions/18411081/sorting-rows-with-null-values-in-sql)

### Example:

1. Retrieve all employees whose names start with 'J'.
2. Find the products with names containing the word 'red'.
3. Get the list of employees sorted by their hire date in descending order.

# PRACTICE QUESTIONS

1. Retrieve all customers with names starting with 'A' and ending with 'n'.
Dataset: Customers (customer_id, customer_name)

2. Find the products with names containing at least one digit.
Dataset: Products (product_id, product_name)

3. Get the list of employees sorted by their salary in ascending order. NULL values should appear at the end.
Dataset: Employees (employee_id, employee_name, salary)

4. Retrieve the customers whose names contain exactly five characters.
Dataset: Customers (customer_id, customer_name)

5. Find the products with names starting with 'S' and ending with 'e'.
Dataset: Products (product_id, product_name)

6. Get the list of employees sorted by their last name and then by their first name.
Dataset: Employees (employee_id, first_name, last_name, salary)

7. Retrieve the orders placed on a specific date and sort them by the customer name in alphabetical order.
Dataset: Orders (order_id, order_date, customer_id)

8. Find the products with names containing exactly three letters.
Dataset: Products (product_id, product_name)

9. Get the list of employees sorted by their salary in descending order. NULL values should appear at the beginning.
Dataset: Employees (employee_id, employee_name, salary)

10. Retrieve the customers whose names contain a space character.
Dataset: Customers (customer_id, customer_name)

# DAY 18

## AGGREGATIONS AND GROUPING

### AIM:

Understand aggregate functions and grouping data using the GROUP BY clause.

### Practice Questions:

- "SQL Aggregate Functions" on sqlservertutorial.net (https://www.sqlservertutorial.net/sql-server-aggregate-functions/)
- "GROUP BY Clause" on geeksforgeeks.org (https://www.geeksforgeeks.org/sql-group-by/)

### Example:

1. Calculate the total order amount for each customer.
2. Find the average salary for each department.
3. Get the maximum and minimum quantities ordered for each product.

# PRACTICE QUESTIONS

1. Calculate the total quantity and total amount for each order.
Dataset: Orders (order_id, order_date), Order_Items (order_id, product_id, quantity, amount)

2. Find the average age and the number of employees for each job title.
Dataset: Employees (employee_id, employee_name, age, job_title)
3. Get the total number of products in each category.
Dataset: Products (product_id, product_name, category_id), Categories (category_id, category_name)

4. Calculate the average rating and the number of reviews for each product.
Dataset: Products (product_id, product_name), Reviews (product_id, rating)

5. Find the customers with the highest and lowest total order amounts.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id, order_amount) 6. Get the maximum and minimum ages for each department.
Dataset: Employees (employee_id, employee_name, age, department)

7. Calculate the total sales amount and the number of orders for each month.
Dataset: Orders (order_id, order_date, order_amount)

8. Find the average price and the number of products for each supplier.
Dataset: Products (product_id, product_name, price, supplier_id), Suppliers (supplier_id, supplier_name)

9. Get the maximum and minimum prices for each product category.
Dataset: Products (product_id, product_name, category_id), Categories (category_id, category_name)

10. Calculate the average rating and the number of reviews for each product category.
Dataset: Products (product_id, product_name, category_id), Reviews (product_id, rating)

# DAY 19

## ADVANCED DATA MANIPULATION

### AIM:

Learn to update and delete data, and handle transactions and concurrency.

### Practice Questions:

- "SQL UPDATE Statement" and "SQL DELETE Statement" on techonthenet.com (https://www.techonthenet.com/sql/update.php)
- "SQL Transactions" on tutorialspoint.com (https://www.tutorialspoint.com/sql/sql-transactions.htm)

### Example:

1. Update the email address of a specific customer.
2. Delete all orders placed by a certain customer.
3. Insert a new product into the database and ensure transactional integrity.

# PRACTICE QUESTIONS

1. Increase the salary of all employees by 10%. Dataset: Employees (employee_id, employee_name, salary)

2. Delete all orders older than 1 year and their associated order items.
Dataset: Orders (order_id, order_date), Order_Items (order_id, product_id, quantity)

3. Insert a new category into the database and update all products of a specific category to the new category in a single transaction.
Dataset: Categories (category_id, category_name), Products (product_id, product_name, category_id)

4. Update the discount percentage for all products in a specific price range.
Dataset: Products (product_id, product_name, price, discount_percentage)

5. Delete all reviews with a rating lower than 3.
Dataset: Reviews (product_id, rating, review_text)

6. Insert a new customer into the database along with their associated orders and order items in a single transaction.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id, order_date), Order_Items (order_id, product_id, quantity)

7. Increase the salary of all employees in a specific department by 15%.
Dataset: Employees (employee_id, employee_name, salary, department)

8. Delete all products that have not been ordered.
Dataset: Products (product_id, product_name), Order_Items (order_id, product_id, quantity)

9. Insert a new supplier into the database along with their associated products and ensure that all the records are inserted or none at all.
Dataset: Suppliers (supplier_id, supplier_name), Products (product_id, product_name, supplier_id)

10. Update the order dates for all orders placed on weekends to the following Monday.
Dataset: Orders (order_id, order_date)

# DAY 20

## ADVANCED DATABASE CONCEPTS

### AIM:

Understand views and their usage, as well as indexing and optimization techniques.

### Practice Questions:

- "SQL Views" on oracle.com (https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/CREATE-VIEW.html)
- "SQL Indexing and Performance Tuning" on sqlshack.com (https://www.sqlshack.com/sql-indexing-and-performance-tuning/)

### Example:

1. Create a view to retrieve the list of products and their quantities in stock.
2. Optimize a slow-performing query using proper indexing techniques.

# PRACTICE QUESTIONS

1. Create a view to display the total sales amount for each product.
Dataset: Products (product_id, product_name), Order_Items (order_id, product_id, quantity, amount)

2. Optimize a query that retrieves the order details for a specific customer, sorting them by the order date in descending order.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id, order_date, order_amount)

3. Create an index on the "last_name" column of the "Employees" table and measure the performance improvement on a specific query.
Dataset: Employees (employee_id, first_name, last_name, salary)

4. Create a view to display the average rating and the number of reviews for each product.
Dataset: Products (product_id, product_name), Reviews (product_id, rating)

5. Optimize a query that retrieves the top 10 customers with the highest total order amounts.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id, order_amount)

6. Create an index on the "order_date" column of the "Orders" table and analyze the query performance for a specific date range.
Dataset: Orders (order_id, order_date, order_amount)

7. Create a view to display the average salary for each department.
Dataset: Employees (employee_id, employee_name, salary, department)
8. Optimize a query that retrieves the list of products with their respective categories, filtering them by a specific category.
Dataset: Products (product_id, product_name, category_id), Categories (category_id, category_name)

9. Create an index on the "product_name" column of the "Products" table and analyze the query performance for a specific search term.
Dataset: Products (product_id, product_name, category_id)

10. Create a view to display the total order amount for each customer.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id, order_amount)

# DAY 21

## ADVANCED SQL FUNCTIONS

### AIM:

Explore window functions and stored procedures.

### Practice Questions:

- "SQL Window Functions" on postgresql.org (https://www.postgresql.org/docs/current/tutorial-window.html)
- "SQL Stored Procedures" on tutorialspoint.com (https://www.tutorialspoint.com/sql/sql-stored-procedures.htm)

### Example:

1. Calculate the cumulative sales amount for each product using a window function.
2. Create a stored procedure to insert a new customer into the database.

# PRACTICE QUESTIONS

1. Retrieve the top 3 customers based on their total order amounts, and calculate the percentage of each customer's order amount compared to the total.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id, order_amount)

2. Create a stored procedure to update the salary of an employee and log the change in a separate table.
Dataset: Employees (employee_id, employee_name, salary), Salary_Log (log_id, employee_id, old_salary, new_salary, modified_date)

3. Calculate the average rating for each product and assign a rank based on the rating using a window function.
Dataset: Products (product_id, product_name), Ratings (product_id, rating)

4. Implement a stored procedure to insert a new order along with its order items into the database.
Dataset: Orders (order_id, order_date), Order_Items (order_id, product_id, quantity, amount)

5. Retrieve the top 5 products based on the cumulative sales amount using a window function.
Dataset: Products (product_id, product_name), Order_Items (order_id, product_id, quantity, amount)

6. Create a stored procedure to calculate the total order amount for a specific customer and return the result.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id, order_amount

7. Calculate the average rating for each product category and assign a rank based on the rating using a window function.
Dataset: Products (product_id, product_name, category_id), Ratings (product_id, rating), Categories (category_id, category_name)

8. Implement a stored procedure to delete a customer and all associated orders and order items from the database.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id), Order_Items (order_id, product_id, quantity)

9. Retrieve the top 3 employees based on their total sales amounts using a window function.
Dataset: Employees (employee_id, employee_name), Orders (order_id, employee_id, order_amount)

10. Create a stored procedure to update the quantity in stock for a specific product and log the change in a separate table.
Dataset: Products (product_id, product_name, quantity_in_stock), Stock_Log (log_id, product_id, old_quantity, new_quantity, modified_date)

# DAY 22

## ADVANCED SQL CONCEPTS

### AIM:

Explore data modeling and normalization concepts, as well as advanced SQL techniques.

### Practice Questions:

- "Database Normalization" on studytonight.com (https://www.studytonight.com/dbms/database-normalization.php)
- "Advanced SQL" on tutorialspoint.com (https://www.tutorialspoint.com/advanced_sql/index.htm)

### Example:

1. Design a database schema for an online bookstore using entity-relationship modeling.
2. Write a recursive SQL query to find all employees and their subordinates in a hierarchical organization structure.

# PRACTICE QUESTIONS

1. Normalize the given unnormalized table into 3rd normal form (3NF).
Dataset: Employees (employee_id, employee_name, department, city, country)

2. Write a recursive SQL query to find all ancestors of a specific employee in a hierarchical employee table.
Dataset: Employees (employee_id, employee_name, manager_id)

3. Use advanced SQL techniques to pivot the given table and transform rows into columns.
Dataset: Sales (product_id, month, amount)

4. Design a database schema for a university system using entity-relationship modeling.
Dataset: Students (student_id, student_name), Courses (course_id, course_name), Enrollments (enrollment_id, student_id, course_id)

5. Write a recursive SQL query to find all dependent employees under a specific manager in a hierarchical organization structure.
Dataset: Employees (employee_id, employee_name, manager_id)

6. Use advanced SQL techniques to unpivot the given table and transform columns into rows.
Dataset: Sales (product_id, month1_amount, month2_amount, month3_amount)

7. Design a database schema for an online marketplace using entity-relationship modeling.
Dataset: Customers (customer_id, customer_name), Products (product_id, product_name), Orders (order_id, customer_id, product_id)

8. Write a recursive SQL query to find all categories and their subcategories in a hierarchical category table.
Dataset: Categories (category_id, category_name, parent_category_id)

9. Use advanced SQL techniques to perform a cross join between two tables.
Dataset: Table1 (column1), Table2 (column2)

10. Design a database schema for a music streaming service using entity-relationship modeling.
Dataset: Users (user_id, username), Songs (song_id, song_name), Playlists (playlist_id, user_id, song_id)

# DAY 23

## SQL PERFORMANCE OPTIMIZATION

### AIM:

Learn techniques for optimizing SQL queries and improving performance.

### Practice Questions:

- "Tips for Optimizing SQL Queries" on dev.to (https://dev.to/techgirl1908/tips-for-optimizing-sql-queries-28f3)
- SQL Query Optimization Techniques" on tutorialgateway.org (https://www.tutorialgateway.org/sql-query-optimization-techniques/)

### Example:

1. Identify and eliminate redundant or unnecessary joins in a complex query.
2. Rewrite a subquery as a join to improve query performance.
3. Use appropriate indexes to optimize query execution

# PRACTICE QUESTIONS

1. Optimize a query that retrieves customer details along with their total order amounts for a specific date range.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id, order_date, order_amount)

2. Identify and eliminate unnecessary joins in a query that retrieves product details and their corresponding categories.
Dataset: Products (product_id, product_name, category_id), Categories (category_id, category_name)

3. Rewrite a subquery as a join in a query that retrieves the order details along with the customer names for all orders.
Dataset: Orders (order_id, customer_id, order_date), Customers (customer_id, customer_name)

4. Optimize a query that calculates the average rating for each product by using appropriate indexes.
Dataset: Products (product_id, product_name), Ratings (product_id, rating)

5. Identify and eliminate redundant joins in a query that retrieves employee details along with their department information.
Dataset: Employees (employee_id, employee_name, department_id), Departments (department_id, department_name)

6. Rewrite a subquery as a join in a query that retrieves the names of customers who have placed at least two orders.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id)

8. Identify and eliminate unnecessary joins in a query that retrieves product details and their corresponding suppliers' names.
Dataset: Products (product_id, product_name, supplier_id), Suppliers (supplier_id, supplier_name)

9. Rewrite a subquery as a join in a query that retrieves the names of customers who have placed orders in the past 30 days.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id, order_date)

10. Optimize a query that retrieves the top 5 products with the highest sales amounts by using appropriate indexes.
Dataset: Products (product_id, product_name), Order_Items (order_id, product_id, quantity, amount)

# DAY 24

## ADVANCED SQL TECHNIQUES

### AIM:

Explore advanced SQL techniques and concepts, including recursive queries, common table expressions, and window functions.

### Practice Questions:

- • "SQL Recursive Queries" on postgresql.org (https://www.postgresql.org/docs/current/queries-with.html)
- "Introduction to Common Table Expressions" on sqlshack.com (https://www.sqlshack.com/introduction-to-common-table-expressions-ctes-in-sql-server/)
- "Window Functions" on sqlite.org (https://www.sqlite.org/windowfunctions.html)

### Example:

1. Write a recursive SQL query to find all employees and their subordinates in a hierarchical organization structure.
2. Use a common table expression to calculate the running total of sales amounts for each product.
3. Apply window functions to calculate moving averages of product ratings.

# PRACTICE QUESTIONS

1. Write a recursive SQL query to find all categories and their subcategories in a hierarchical category table.
Dataset: Categories (category_id, category_name, parent_category_id)

2. Use a common table expression to calculate the running total of order amounts for each customer.
Dataset: Customers (customer_id, customer_name), Orders (order_id, customer_id, order_amount)

3. Apply window functions to calculate the average rating and the maximum rating for each product.
Dataset: Products (product_id, product_name), Ratings (product_id, rating)

4. Write a recursive SQL query to find all employees and their direct reports in a hierarchical employee table.
Dataset: Employees (employee_id, employee_name, manager_id)

5. Use a common table expression to calculate the cumulative sum of quantities for each product
Dataset: Products (product_id, product_name), Order_Items (order_id, product_id, quantity)
6. Apply window functions to calculate the minimum and maximum order amounts for each month.
Dataset: Orders (order_id, order_date, order_amount)

7. Write a recursive SQL query to find all ancestors of a specific employee in a hierarchical employee table.
Dataset: Employees (employee_id, employee_name, manager_id)

8. Use a common table expression to calculate the average rating and the number of reviews for each product.
Dataset: Products (product_id, product_name), Reviews (product_id, rating)

9. Apply window functions to calculate the rank and dense rank of sales amounts for each product.
Dataset: Products (product_id, product_name), Order_Items (order_id, product_id, amount)

10. Write a recursive SQL query to find all dependent employees under a specific manager in a hierarchical organization structure.
Dataset: Employees (employee_id, employee_name, manager_id)

Analytics
Career Connect

# Kickstart Your Data Analytics Journey Today!

**Internship Program :** Get real-world experience through hands-on projects, preparing you for the professional world.
**Mentorship Program :** Learn to upskill effectively with expert guidance and personalized self-learning strategies
**Job Assistance Program :** Connect with top employers and gain the tools to confidently secure your dream job.

**If you want to become a Data Analyst with the help of free resources and without investing in expensive courses, Connect With Us.**

**Follow us for more**

LinkedIn

Analytics Career Connect

Wasim Patwari