Module 3

What to learn

- String Functions
  - ASCII
  - CHAR,
  - CHARINDEX
  - CONCAT
  - FORMAT
  - LEFT
  - LEN
  - LOWER
  - LTRIM
  - PATINDEX
  - REPLACE
  - RTRIM
  - SOUNDEX
  - SPACE
  - STR
  - STUFF
  - SUBSTRING
  - TRIM
  - REPLICATE
  - REVERSE
  - RIGHT
- Date Functions
  - Dateadd
  - Datename
  - Getdate
  - Day
- Mathematical Functions
- System functions

Practice Exercise

Practice 1

Do the hands on from videos provided on the tutorial site.

Practice 2

Inner Join on Employees and Departments

Create two tables with the following structure: **Employees** (EmployeeID, EmployeeName, DepartmentID) and **Departments** (DepartmentID, DepartmentName). Populate them with sample data. Write an SQL query to retrieve a list of employees along with their department names using an **INNER JOIN**. Ensure the query only returns employees with matching departments.

Practice 3

Left Outer Join Between Orders and Customers

Create two tables: **Orders** (OrderID, OrderDate, CustomerID) and **Customers** (CustomerID, CustomerName). Write a query to retrieve all orders, including those that do not have corresponding customer information, using a **LEFT OUTER JOIN**.

Practice 4

Right Outer Join for Courses and Enrollments

Create two tables: **Courses** (CourseID, CourseName, Instructor) and **Enrollments** (EnrollmentID, StudentName, CourseID). Write a query to fetch all courses, ensuring that it includes courses that have no enrollments, using a **RIGHT OUTER JOIN**.

Practice 5

Full Outer Join Comparing Projects and Employees

Create two tables: **Projects** (ProjectID, ProjectName) and **EmployeeProjects** (EmployeeID, ProjectID). Populate the tables with sample data. Write a query using a **FULL OUTER JOIN** to display all projects and employees (even if they are not matched).

Practice 6

Self-Join to Find Employee Hierarchies

Create an **Employees** table with the following columns: EmployeeID, EmployeeName, ManagerID. Populate the table with hierarchical data where some employees have managers. Write a query to list each employee along with their manager's name using a **SELF-JOIN**.

Practice 7

Exploring Null Handling in Outer Joins

Using the **Customers** and **Orders** tables created earlier, write a query to retrieve all customers and their order details (if any). Focus on how **NULL** values in the result set behave when customers don't have corresponding orders.

Practice 8

Combining Inner and Outer Joins

Create three tables: **Suppliers** (SupplierID, SupplierName), **Products** (ProductID, ProductName, SupplierID), and **Orders** (OrderID, ProductID). Write a query using inner and outer joins to retrieve products and their supplier details, including those products that haven't been ordered yet.

Practice 9

Using Aggregate Functions with Joins

Using the **Orders** and **Customers** tables, write a query that retrieves the total number of orders placed by each customer using a **LEFT OUTER JOIN**. Include customers with zero orders in the result using **COALESCE** to handle NULL values.

Practice 10

Performing Multi-Level Self-Joins

Create an **Employees** table with columns for EmployeeID, EmployeeName, and ManagerID. Write a query that retrieves all employees along with both their direct managers and the higher-level manager for each employee (manager of their manager) using multiple **SELF-JOIN** operations.

Assignment Exercise

Assignment 1

Write a query that displays the FirstName and the length of the FirstName for all employees whose name starts with the letters ‘A’, ‘J’ or ‘M’. Give each column an appropriate label. Sort the results by the employees’ FirstName

Assignment 2

Write a query to display the FirstName and Salary for all employees. Format the salary to be 10 characters long, left-padded with the $ symbol. Label the column SALARY.

Assignment 3

Write a query to display the employees with their code, first name, last name and hire date who hired either on seventh day of any month or seventh month in any year.

Assignment 4

Write a query to display the length of first name for employees where last name contains character ‘c’ after 2nd position.

Assignment 5

Write a query to extract the last 4 character of PhoneNumber.

Assignment 6

Write a query to update the portion of the PhoneNumber in the employees table, within the phone number the substring ‘124’ will be replaced by ‘999’.

Assignment 7

Write a query to calculate the age in year.

Assignment 8

Write a query to get the distinct Mondays from HireDate in employees tables.

Assignment 9

Write a query to get the FirstName and HireDate from Employees table where HireDate between ‘1987-06-01’ and ‘1987-07-30’

Assignment 10

Write a query to display the current date in the following format. Sample output : 12:00 AM Sep 5, 2014

Assignment 11

Write a query to get the FirstName, LastName who joined in the month of June.

Assignment 12

Write a query to get first name, hire date and experience of the employees.

Assignment 13

Write a query to get first name of employees who joined in 1987.

Assignment 14

Real-World Application: Employee-Department Management System

Scenario:

You are working for a company and need to design a database system for managing employees and their respective departments.

Functional Flow:

Create two tables: **Employees** (EmployeeID, EmployeeName, DepartmentID, JoinDate, Salary) and **Departments** (DepartmentID, DepartmentName, ManagerID) with appropriate constraints and relationships.

Populate these tables with realistic data.

Using an **INNER JOIN**, retrieve a list of employees along with their department names.

Using a **LEFT OUTER JOIN**, fetch all employees along with department information, ensuring that employees without a department are included in the result.

Using a **RIGHT OUTER JOIN**, retrieve all departments and their employees, ensuring that departments without employees are also displayed.

Execute a **FULL OUTER JOIN** to include all employees and all departments, even if data is not matched.

Using a **SELF-JOIN**, fetch the supervisors for each employee, showing the hierarchy up to two levels if applicable (employees reporting and their immediate supervisors).

Business Logic:

The **Departments** table must have a unique manager assigned to each department.
An employee can either belong to one department or none at all (null value for DepartmentID).
Salary information should only be displayed for employees who have joined after a specific date (e.g., 2020-01-01).
In the hierarchy setup, attempt to avoid circular reporting relationships (e.g., an employee cannot be their own manager).

Technical Requirements:

Make use of aggregate functions like SUM and COUNT to display the total salary expenditure of each department and the total number of employees in each department, including departments without employees.
Create views to simplify queries and represent data for reporting:
- List of employees with their departments (if any).
- List of departments with their assigned employees and supervisors.

Ensure the use of appropriate constraints, e.g., **NOT NULL**, **FOREIGN KEY**, and **CHECK** constraints for data integrity.

Deliverables:

SQL scripts for table creation and sample data population.
Queries for all join conditions (inner, left, right, full, self).
At least three views as described in the technical requirements.
A report summarizing the learnings, findings, or edge cases encountered during the task implementation.

Online Reference

No online Reference

Supported Files

Employees.sql

Introduction to Relational Databases

Introduction to Select Statement

Filtering Results with WHERE Statements

Utilizing Joins

Executing Sub queries and Unions

Aggregating Data

Advanced Data Aggregations

Built in Functions

Query Optimization

Modifying Data

Advanced Data Modification

Stored Procedure

Transaction

Error handling

Designing Tables

triggers