Module 4

What to learn

- Ranking Functions
  - Row_number()
  - Rank()
  - Dense_rank()
- Aggregate Function
  - SUM
  - COUNT
  - AVG
  - MAX
  - MIN
- Group by/Having/ROLLUP

Practice Exercise

Practice 1

Do the hands on the video provided in tutorial site.

Practice 2

Querying All Employees with High Commissions

Given the employee table named **Employee** with the following columns: *EmployeeID*, *Name*, *Position*, *Salary*, *Commission*, and *DepartmentID*, write a subquery to fetch all employees who earn a commission greater than the average commission of all employees. Ensure the query utilizes subquery functionality effectively.

Table: `Employee`

Column	Description	Datatype
EmployeeID	Unique ID of the Employee	INT
Name	Name of the Employee	VARCHAR
Position	Job Position of the Employee	VARCHAR
Salary	Monthly Salary	DECIMAL
Commission	Additional Earnings from Commission	DECIMAL
DepartmentID	Linked Department ID	INT

Practice 3

Find Departments with Above-Average Salaries

Given tables **Employee** and **Department**, write a subquery to fetch department names where the average salary of the employees exceeds $5,000.

Use **DepartmentID** to join and ensure use of subqueries in the filtering process.

Table: `Department`

Column	Description	Datatype
DepartmentID	Unique ID for Each Department	INT
DepartmentName	Name of the Department	VARCHAR

Practice 4

Extract Customers with Past-Due Invoices

Given a table **CustomerInvoices** containing details of customer transactions with the following schema: `CustomerID`, `InvoiceAmount`, `DueDate`, and `PaymentStatus`, write a subquery to find all customers with overdue invoices. Use the *DueDate* column to compare values and assume today's date is 2023-10-20.

Table: `CustomerInvoices`

Column	Description	Datatype
CustomerID	Unique ID for Customers	INT
InvoiceAmount	Amount Due for Invoices	DECIMAL
DueDate	Date Payment was Due	DATE
PaymentStatus	Status of Invoice (Paid, Unpaid)	VARCHAR

Practice 5

Identify Customers with Multiple Orders

Given a table **Orders** with the schema `OrderID`, `CustomerID`, `OrderDate`, and `TotalAmount`, write a subquery to retrieve CustomerIDs of those customers who have placed more than two orders.

Table: `Orders`

Column	Description	Datatype
OrderID	Unique Order ID	INT
CustomerID	Unique Customer ID	INT
OrderDate	Date of Order Placement	DATE
TotalAmount	Total Amount of Order	DECIMAL

Practice 6

Creating and Fetching Data Using Simple Views

Given a table **Products** with columns `ProductID`, `Name`, `Category`, `Price`, and `StockQuantity`, create a view **LowStockProducts** that lists all products having `StockQuantity` below 50. Write a SELECT query to retrieve this view.

Table: `Products`

Column	Description	Datatype
ProductID	Unique ID for Products	INT
Name	Name of the Product	VARCHAR
Category	Product Category	VARCHAR
Price	Price of the Product	DECIMAL
StockQuantity	Quantity of Stock Left	INT

Assignment Exercise

Assignment 1

Table Name: Employee Write a query to rank employees based on their salary for a month Select 4th Highest salary from employee table using ranking function Get department, total salary with respect to a department from employee table. Get department, total salary with respect to a department from employee table order by total salary descending Get department wise maximum salary from employee table order by salary ascending Get department wise minimum salary from employee table order by salary ascending Select department, total salary with respect to a department from employee table where total salary greater than 50000 order by TotalSalary descending

Assignment 2

Property Management System Reporting

You're designing a simplified database reporting solution for a property management company. Consider the database structure below and implement the requirements using both subqueries and views where required.

Database Schema

Tables to work with:

- **Properties**: Contains property details
  - *PropertyID* (INT, Primary Key)
  - *Address* (VARCHAR)
  - *OwnerID* (INT, Foreign Key to Owners)
  - *PropertyType* (VARCHAR)
- **Owners**: List of property owners
  - *OwnerID* (INT, Primary Key)
  - *Name* (VARCHAR)
  - *Phone* (VARCHAR)
- **Transactions**: Records all transactions made
  - *TransactionID* (INT, Primary Key)
  - *PropertyID* (INT, Foreign Key to Properties)
  - *Amount* (DECIMAL)

- *TransactionDate* (DATE)
- **Tenants**: List of tenants renting properties
  - *TenantID* (INT, Primary Key)
  - *Name* (VARCHAR)
  - *Phone* (VARCHAR)
- **Rentals**: Connects tenants to properties
  - *RentalID* (INT, Primary Key)
  - *TenantID* (INT, Foreign Key to Tenants)
  - *PropertyID* (INT, Foreign Key to Properties)
  - *RentStartDate* (DATE)
  - *RentEndDate* (DATE)

Tasks

- Create the tables in SQL and insert realistic data covering at least 10 records per table.
- Create a subquery to list all property owners who have not received any payment (join *Properties* and *Transactions*).
- Create a view *ActiveRentals* to show all active rental agreements (current date falls between *RentStartDate* and *RentEndDate*).
- Create a view *HighValueTransactions* to show all transactions where the *Amount* exceeds $1,000.
- Create a subquery to fetch tenant names who rented properties in a specific date range (e.g., 2023-01-01 to 2023-06-30).
- Write queries using the created views to ensure they work correctly.

Online Reference

No online Reference

Supported Files

SQL Sample.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