SQL CONCEPTS

1,SQL Exercise - ADVANCED CONCEPTS:

Exercise 1: Ranking and Window Functions

=>SELECT \*

FROM (

SELECT \*,

ROW\_NUMBER() OVER (PARTITION BY Category ORDER BY Price DESC) AS RowNum

FROM Products

) ranked

WHERE RowNum <= 3;

=>SELECT \*

FROM (

SELECT \*,

RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS Rank

FROM Products

) ranked

WHERE Rank <= 3;

=>SELECT \*

FROM (

SELECT \*,

DENSE\_RANK() OVER (PARTITION BY Category ORDER BY Price DESC) AS DenseRank

FROM Products

) ranked

WHERE DenseRank <= 3;

1. SQL Exercise - Index:

Exercise 1: Creating a Non-Clustered Index

CREATE NONCLUSTERED INDEX idx\_ProductName ON Products(ProductName);

Exercise 2: Creating a Clustered Index

CREATE CLUSTERED INDEX idx\_OrderDate ON Orders(OrderDate);

Exercise 3: Creating a Composite Index

CREATE NONCLUSTERED INDEX idx\_CustomerID\_OrderDate

ON Orders(CustomerID, OrderDate);

1. SQL Exercise - Stored procedure:

Exercise 1: Create a Stored Procedure

Step 1: Define the stored procedure with a parameter for DepartmentID

CREATE PROCEDURE sp\_GetEmployeesByDepartment

@DeptID INT

Step 2: Write the SQL query to select employee details based on the DepartmentID

SELECT

E.EmployeeID,

E.FirstName,

E.LastName,

D.DepartmentName,

E.Salary,

E.JoinDate

FROM Employees E

INNER JOIN Departments D ON E.DepartmentID = D.DepartmentID

WHERE E.DepartmentID = @DeptID;

Step 3: Create a stored procedure named sp\_InsertEmployee with the given code

CREATE PROCEDURE sp\_InsertEmployee

@FirstName VARCHAR(50),

@LastName VARCHAR(50),

@DepartmentID INT,

@Salary DECIMAL(10,2),

@JoinDate DATE

AS

BEGIN

INSERT INTO Employees (FirstName, LastName, DepartmentID, Salary, JoinDate)

VALUES (@FirstName, @LastName, @DepartmentID, @Salary, @JoinDate);

END;

Exercise 4: Execute a Stored Procedure

Step 1: SQL Command to Execute the Stored Procedure

EXEC sp\_GetEmployeesByDepartment @DeptID = 3;

Step 2: Execute and Review the Results

| **EmployeeID** | **FirstName** | **LastName** | **DepartmentName** | **Salary** | **JoinDate** |
| --- | --- | --- | --- | --- | --- |
| 3 | Michael | Johnson | IT | 7000.00 | 2018-07-30 |

Exercise 5: Return Data from a Stored Procedure

Step 1: Define the stored procedure with a parameter for DepartmentID

CREATE PROCEDURE sp\_GetEmployeeCountByDepartment

@DeptID INT

Step 2: Write the SQL query to count employees in the department

AS

BEGIN

SELECT

COUNT(\*) AS TotalEmployees

FROM Employees

WHERE DepartmentID = @DeptID;

END;

Step 3: Save the procedure by executing it

EXEC sp\_GetEmployeeCountByDepartment @DeptID = 1;

1. SQL Exercise - Functions:

Exercise 7: Return Data from a Scalar Function

Step 1: First, Define the Scalar Function (if not already created)

CREATE FUNCTION fn\_CalculateAnnualSalary (@EmpID INT)

RETURNS DECIMAL(10,2)

AS

BEGIN

DECLARE @AnnualSalary DECIMAL(10,2);

SELECT @AnnualSalary = Salary \* 12

FROM Employees

WHERE EmployeeID = @EmpID;

RETURN @AnnualSalary;

END;

Step 2: Execute the Function for EmployeeID = 1

SELECT dbo.fn\_CalculateAnnualSalary(1) AS AnnualSalary;

Step 3: Verify the Result

| **AnnualSalary** |
| --- |
| 60000.00 |