SQL TASKS

```sql
CREATE TABLE employees (
employee_id INT PRIMARY KEY,
first_name VARCHAR(50),
last_name VARCHAR(50),
department_id INT
);
CREATE TABLE departments (
department_id INT PRIMARY KEY,
department_name VARCHAR(100)
);
CREATE DATABASE companydb;
USE companydb;
INSERT INTO employees VALUES
(1,'sreedev','dasappan',1),
(2,'yadhu','radhakrishnan',2),
(3,'unni','yeshudas',3);
INSERT INTO departments VALUES
(1,'it'),
(2,'design'),
(3,'hr');
SELECT e.first_name,e.last_name,d.department_name
FROM employees e INNER JOIN departments d ON
e.department_id=d.department_id;


--indendation


SELECT
e.first_name,
e.last_name,
d.department_name
FROM
employees e
INNER JOIN
departments d
ON
e.department_id=d.department_id
WHERE
d.department_name='it';
SELECT * FROM
employees
WHERE
employee_id=1;
ALTER TABLE employees ADD salary DECIMAL(10,2);
UPDATE employees SET salary=10000.02 WHERE salary is null;


--Stored procedure


CREATE PROCEDURE UpdateEmployeeSalary(
@emp_id INT,
@new_salary DECIMAL)
AS
```

```
BEGIN
UPDATE employees SET salary=@new_salary WHERE employee_id=@emp_id;
END;
EXEC UpdateEmployeeSalary @emp_id=1,@new_salary=25000.90;


CREATE PROCEDURE GetEmployeeDetails(@EmployeeID INT)
AS
BEGIN
SELECT * FROM employees WHERE employee_id = @EmployeeID;
END;
DROP PROCEDURE GetEmployeeDetails;
CREATE PROCEDURE GetEmployeeDetails(@EmployeeID INT)
AS
BEGIN
IF @EmployeeID IS NULL
BEGIN
RAISERROR('EmployeeID cannot be NULL', 16, 1);
RETURN;
END
ELSE
BEGIN
SELECT * FROM employees WHERE employee_id=@EmployeeID;
END
END;
EXEC GetEmployeeDetails @EmployeeID=1;

--sp using try catch


DROP PROCEDURE UpdateEmployeeSalary;
EXEC UpdateEmployeeSalary @EmployeeID=null,@NewSalary=10000000000002344444;
SELECT * FROM employees;
SELECT e.first_name,e.last_name,d.department_name,e.salary
FROM employees e INNER JOIN departments d ON
e.department_id=d.department_id
WHERE e.salary>5000 AND e.salary<12000;


--schema


CREATE SCHEMA HR;
CREATE TABLE HR.employees (
employee_id INT PRIMARY KEY,
first_name VARCHAR(50),
last_name VARCHAR(50),
department_id INT,
FOREIGN KEY(department_id) REFERENCES HR.departments(department_id)
);
CREATE TABLE HR.departments (
department_id INT PRIMARY KEY,
department_name VARCHAR(100)
);
ALTER TABLE HR.employees
ADD email VARCHAR(200) UNIQUE,
dob DATE NOT NULL,
title VARCHAR(50);
```

```sql
SELECT * FROM HR.employees;
CREATE SCHEMA Finance;
CREATE TABLE Finance.Salaries(
SalaryID INT PRIMARY KEY,
employee_id INT,
MonthlySalary DECIMAL (10, 2),
PayDate DATE,
FOREIGN KEY (employee_id) REFERENCES HR.employees(employee_id)
);


INSERT INTO HR.departments(department_id, department_name)
VALUES (1, 'IT'),(2, 'HR');
SELECT * FROM Finance.Salaries;
SELECT * FROM HR.employees;
ALTER TABLE HR.employees
DROP COLUMN title;
INSERT INTO HR.employees (employee_id, first_name, last_name, department_id,email,dob)
VALUES (1, 'John', 'Doe', 1, 'john.doe@example.com','1990-01-01'),
(2, 'Jane', 'Smith', 2,'jane.smith@example.com','1985-05-05');
INSERT INTO Finance. Salaries (SalaryID, employee_id, MonthlySalary, PayDate)
VALUES(1, 1, 5000, '2023-01-01'),
(2, 2, 4000, '2023-01-01');
CREATE TABLE HR.JobTitles(
JobTitleID INT PRIMARY KEY, JobTitle VARCHAR (50));


--adding job title reference to employee table


ALTER TABLE HR. Employees
ADD JobTitleID INT,
FOREIGN KEY (JobTitleID) REFERENCES HR.JobTitles(JobTitleID);
INSERT INTO HR.JobTitles(JobTitleID, JobTitle) VALUES (1, 'Developer'), (2, 'HR
Manager');
UPDATE HR.Employees
SET JobTitleID=1 WHERE employee_id = 1;
UPDATE HR.Employees
SET JobTitleID=2 WHERE employee_id= 2;
SELECT * FROM Finance.Salaries;
SELECT * FROM HR.employees;


--create a view


CREATE VIEW HR.EmployeeSalaryView AS
SELECT e.employee_id, e.first_name, e.last_name, jt.JobTitle,s.MonthlySalary
FROM HR.employees e
JOIN Finance.Salaries s ON e.employee_id= s.employee_id
JOIN HR.JobTitles jt ON e.JobTitleID = jt.JobTitleID;
SELECT * FROM EmployeeSalaryView;
SELECT *
FROM HR.EmployeeSalaryView;


--indexing
--clustered index
```

```sql
CREATE TABLE Orders (
OrderID INT PRIMARY KEY,
CustomerID INT,
OrderDate DATETIME,
TotalAmount DECIMAL(10, 2)
);
INSERT INTO Orders (OrderID, CustomerID, OrderDate, TotalAmount)
VALUES (1, 101, '2024-01-15 10:30:00', 250.50),
(2, 102, '2024-02-10 14:45:00', 480.00),
(3, 103, '2024-03-05 16:20:00', 125.75),
(4, 104, '2024-03-18 12:10:00', 890.25),
(5, 105, '2024-04-01 09:15:00', 320.00);
SELECT OrderID, OrderDate, TotalAmount FROM Orders
WHERE CustomerID = 105;
CREATE TABLE Posts (
PostID INT PRIMARY KEY,
UserID INT,
PostDate DATETIME,
PostContent NVARCHAR (MAX)
);
SELECT * FROM Posts;
SELECT PostID, PostDate, PostContent
FROM Posts
WHERE UserID = 101
AND PostDate> DATEADD (MONTH, -1, GETDATE());
INSERT INTO Posts (PostID, UserID, PostDate, PostContent)
VALUES (1, 101, '2024-11-25 14:30:00', 'Excited for the holidays!');
INSERT INTO Posts (PostID, UserID, PostDate, PostContent)
VALUES (2, 102, '2024-11-15 09:45:00', 'Preparing for a big meeting.');
INSERT INTO Posts (PostID, UserID, PostDate, PostContent)
VALUES (3, 101, '2024-12-01 18:10:00', 'New project launched today!');
INSERT INTO Posts (PostID, UserID, PostDate, PostContent)
VALUES (4, 103, '2024-10-05 08:20:00', 'Autumn vibes everywhere!');
INSERT INTO Posts (PostID, UserID, PostDate, PostContent)
VALUES (5, 101, '2024-12-05 12:25:00', 'Attended a fantastic webinar.');

CREATE INDEX idx_custid ON Posts(UserID);

SELECT * FROM Posts WHERE UserID=101;

ALTER INDEX idx_custid ON Posts DISABLE;


---INDEXING IN BANK APPLICATION

CREATE TABLE Transactions(
TransactionID INT PRIMARY KEY,
AccountID INT,
TransactionDate DATETIME,
TransactionAmount DECIMAL (10, 2),
Description NVARCHAR (255)
);
INSERT INTO Transactions (TransactionID, AccountID, TransactionDate, TransactionAmount,
Description)
VALUES
(1, 1001, '2024-11-01 09:15:30', 250.00, 'Deposit'),
```

```
(2, 1002, '2024-07-02 14:30:45', -100.50, 'ATM Withdrawal'),
(3, 1001, '2024-05-03 16:20:10', -75.25, 'Online Purchase'),
(4, 1003, '2024-03-04 11:05:00', 500.00, 'Salary Credit'),
(5, 1002, '2024-02-05 13:50:25', -200.00, 'Bill Payment');

TRUNCATE TABLE Transactions;

SELECT * FROM Transactions;

SELECT * FROM Transactions
WHERE TransactionAmount>100 AND TransactionDate> DATEADD(MONTH,-6,GETDATE());



CREATE INDEX idx_account_id_date_amount ON Transactions (AccountID, TransactionDate,
TransactionAmount);

CREATE TABLE Products (
ProductID INT PRIMARY KEY,
CategoryID INT,
Product_Name NVARCHAR(255),
Price DECIMAL(10, 2),
Stock INT
);

SELECT ProductID, Product_Name, Price FROM Products
WHERE CategoryID= 5
AND Price BETWEEN 100 AND 500;

INSERT INTO Products (ProductID, CategoryID, Product_Name, Price, Stock)
VALUES
(1, 1, 'Wireless Mouse', 100.99, 150),
(2, 5, 'Bluetooth Headphones', 270.99, 75),
(3, 5, 'Gaming Keyboard', 350.50, 40),
(4, 5, 'USB-C Charging Cable', 1200.49, 200),
(5, 4, '27-inch Monitor', 199.99, 30);



CREATE INDEX idx_categoryid_price ON Products(CategoryID, Price);


--pivot

CREATE TABLE sales(
year INT,
product VARCHAR(20),
sales INT
);

INSERT INTO sales
VALUES(2019,'apple',500),
(2019,'apple',200),
(2019,'orange',300),
(2020,'orange',400),
(2020,'apple',100);

SELECT * FROM sales;
```

```sql
SELECT product,[2019],[2020] FROM
(SELECT * FROM sales)S PIVOT
(SUM(sales) FOR year IN ([2019],[2020]))P;


--merging


CREATE TABLE Sales1(
prod_id INT,
sold INT,
revenue INT
);
CREATE TABLE newSales1(
prod_id INT,
sold INT,
revenue INT
);

INSERT INTO Sales1 VALUES
(101,50,500),
(102,30,300);

INSERT INTO newSales1 VALUES
(101,20,200),
(103,30,300);

MERGE INTO Sales1 AS targett
USING newSales1 AS sourcet ON
targett.prod_id=sourcet.prod_id
WHEN MATCHED THEN
UPDATE SET
targett.sold=targett.sold+sourcet.sold,
targett.revenue=targett.revenue+sourcet.revenue
WHEN NOT MATCHED THEN
INSERT (prod_id,sold,revenue)
VALUES(sourcet.prod_id,sourcet.sold,sourcet.revenue);

SELECT * FROM Sales1;


--stored procedure

CREATE TABLE Students (
Student_id INT PRIMARY KEY,
first_name VARCHAR(25),
last_Name VARCHAR(25),
dob DATE,
admission_date DATE);

CREATE PROCEDURE ManageStudent
@Action NVARCHAR(10),
@Studentid INT = NULL,
@FirstName NVARCHAR(50) = NULL,
@LastName NVARCHAR(50) = NULL,
@DOB DATE = NULL,
@AdmissionDate DATE = NULL
```

```
AS
BEGIN
IF @Action = 'CREATE'
BEGIN
INSERT INTO Students(Student_id,first_name, last_Name, dob, admission_date)
VALUES (@Studentid, @FirstName, @LastName, @DOB, @AdmissionDate);
END

ELSE IF @Action = 'READ'
BEGIN
SELECT * FROM Students WHERE Student_id = @Studentid;
END

ELSE IF @Action = 'UPDATE'
BEGIN
    UPDATE Students
    SET
        first_name = COALESCE(@FirstName, first_name),
        last_name = COALESCE(@LastName, last_name),
        dob = COALESCE(@DOB, dob),
        admission_date = COALESCE(@AdmissionDate, admission_date)
    WHERE Student_id = @Studentid;
END

ELSE IF  @Action = 'DELETE'
BEGIN
DELETE FROM Students WHERE Student_id=@Studentid;
END

END;

EXEC ManageStudent
@Action = 'CREATE',
@Studentid = 101 ,
@FirstName = 'Sreedev',
@LastName = 'Dasappan',
@DOB ='2000-1-1',
@AdmissionDate ='2024-9-1';

EXEC ManageStudent
@Action='READ',
@Studentid=101;


EXEC ManageStudent
@Action = 'UPDATE',
@Studentid = 101 ,
@FirstName = 'Jane';

EXEC ManageStudent
@Action='DELETE',
@Studentid=101;
```