Data Schema

```
--Create Database
Create database Payroll_db;
```

Departments Table

```
--Creating Departments Table
create table Departments(DepartmentID int primary key identity,DepartmentName
nvarchar(100) not null);
```

Employees Table

```
--Creating Employees Table
create table Employees(EmployeeID int primary key identity,Name nvarchar(50) not
null,DepartmentID int foreign key references Departments(DepartmentID),HireDate date);
```

Salaries Table

```
--Creating Salaries Table
create table Salaries(EmployeeID int foreign key references
Employees(EmployeeID),BaseSalary decimal(10,2),Bonus decimal(10,2),Deductions
decimal(10,2));
```

Salary History Table

```
--Creating SalaryHistory Table
CREATE TABLE SalaryHistory (
    HistoryID int primary key identity,
    EmployeeID int,
    BaseSalary decimal(18, 2),
    Bonus decimal(18, 2),
    Deductions decimal(18, 2),
    UpdateDate datetime default GETDATE(),
    Employee_ID int foreign key references Employees(EmployeeID)
);
```

SQL Queries

List all employees along with their department names.

```
   select emp.EmployeeID,emp.Name,dep.DepartmentName from Employees emp INNER JOIN
Departments dep ON emp.DepartmentID=dep.DepartmentID;
```

OUTPUT

EmployeeID	Name	DeaprtmentName
1	Smrithi V T	HR
2	Latha V J	Staff Project
3	Vavachi	Employee Leave

Calculate the net salary for each employee using: Net Salary = BaseSalary + Bonus - Deductions.

```
    select emp.EmployeeID, emp.Name,
       (sal.BaseSalary + sal.Bonus - sal.Deductions) AS NetSalary
    from Employees emp
    INNER JOIN Salaries sal ON emp.EmployeeID = sal.EmployeeID;
```

OUTPUT

EmployeeID	Name	NetSalary
1	Smrithi V T	50600.00
2	Latha V J	20100.00
3	Vavachi	35100.00

Identify the department with the highest average salary.

```
   select top 1 dep.DepartmentName,AVG(sal.BaseSalary + sal.Bonus - sal.Deductions) as
AvgSalary
```

```
   from Departments dep
   INNER JOIN Employees emp ON dep.DepartmentID = emp.DepartmentID
   INNER JOIN Salaries sal ON emp.EmployeeID = sal.EmployeeID
   group by dep.DepartmentName
   order by AvgSalary desc;
```

OUTPUT

```
DepartmentName  AvgSalary
HR              50600.000000
```

Stored Procedures:
Add Employee: A procedure to insert a new employee into the Employees table, ensuring valid DepartmentID and other constraints.

```
CREATE PROCEDURE AddEmployee
    @EmployeeID INT,
    @Name VARCHAR(100),
    @DepartmentID INT,
    @HireDate DATE,
    @BaseSalary DECIMAL(10, 2),
    @Bonus DECIMAL(10, 2),
    @Deductions DECIMAL(10, 2)
AS
BEGIN
    -- Ensure Department exists
    IF NOT EXISTS (SELECT 1 FROM Departments WHERE DepartmentID = @DepartmentID)
    BEGIN
        PRINT 'Invalid DepartmentID';
        RETURN;
    END

    -- Insert Employee record
    INSERT INTO Employees (EmployeeID, Name, DepartmentID, HireDate)
    VALUES (@EmployeeID, @Name, @DepartmentID, @HireDate);

    -- Insert Salary record
    INSERT INTO Salaries (EmployeeID, BaseSalary, Bonus, Deductions)
    VALUES (@EmployeeID, @BaseSalary, @Bonus, @Deductions);

    PRINT 'Employee added successfully.';
END;
```

Update Salary: A procedure to update the salary details of an employee,automatically logging the changes into a SalaryHistory table.

```
CREATE PROCEDURE UpdateSalary
    @EmpID INT,
    @New_BaseSalary DECIMAL(18, 2),
    @New_BonusDECIMAL(18, 2),
    @New_Deductions DECIMAL(18, 2)
AS
BEGIN
    -- Logging the old salary to the SalaryHistory table
    INSERT INTO SalaryHistory (EmployeeID, BaseSalary, Bonus, Deductions)
    SELECT EmployeeID, BaseSalary, Bonus, Deductions
    FROM Salaries
    WHERE EmployeeID = @EmpID;

    -- Update the salary details
    UPDATE Salaries
    SET BaseSalary = @New_BaseSalary, Bonus = @New_Bonus, Deductions = @New_Deductions
    WHERE EmployeeID = @EmpID;
END;
```

Calculate Payroll: A procedure to compute and return the total payroll cost for a department or the entire organization.

```
CREATE PROCEDURE CalculatePayroll
```

```
    @DeptID INT = NULL
AS
BEGIN
    IF @DeptID IS NULL
    BEGIN
        -- Total payroll for the entire organization
        SELECT SUM(S.BaseSalary + S.Bonus - S.Deductions) AS TotalPayroll
        FROM Salaries S;
    END
    ELSE
    BEGIN
        -- Total payroll for a specific department
        SELECT SUM(S.BaseSalary + S.Bonus - S.Deductions) AS TotalPayroll
        FROM Salaries S
        JOIN Employees E ON S.EmployeeID = E.EmployeeID
        WHERE E.DepartmentID = @DeptID;
    END
END;
```

Views:
EmployeeSalaryView: A view that combines Employees, Departments, and Salaries to provide a detailed report of employee salaries with department names and net salaries.

```
CREATE VIEW EmployeeSalaryView AS
SELECT E.EmployeeID, E.Name, D.DepartmentName,
       S.BaseSalary, S.Bonus, S.Deductions,
       (S.BaseSalary + S.Bonus - S.Deductions) AS NetSalary
FROM Employees E
JOIN Departments D ON E.DepartmentID = D.DepartmentID
JOIN Salaries S ON E.EmployeeID = S.EmployeeID;
```

HighEarnerView: A view that lists employees earning above a certain threshold (e.g., a parameter or predefined limit).

```
CREATE VIEW HighEarnerView AS
SELECT E.EmployeeID, E.Name, D.DepartmentName,
       S.BaseSalary, S.Bonus, S.Deductions,
       (S.BaseSalary + S.Bonus - S.Deductions) AS NetSalary
FROM Employees E
JOIN Departments D ON E.DepartmentID = D.DepartmentID
JOIN Salaries S ON E.EmployeeID = S.EmployeeID
WHERE (S.BaseSalary + S.Bonus - S.Deductions) > 100000;
```

Add a SalaryHistory table to log salary updates with triggers.

```
CREATE TRIGGER LogSalaryChange
ON Salaries
AFTER UPDATE
AS
BEGIN
    INSERT INTO SalaryHistory (EmployeeID, BaseSalary, Bonus, Deductions)
    SELECT EmployeeID, BaseSalary, Bonus, Deductions
    FROM inserted;
END;
```

Optimize the queries and stored procedures using proper indexing and query execution plans.

```
-- Index on DepartmentID in Employees table
CREATE INDEX idx_DepartmentID ON Employees (DepartmentID);

-- Index on EmployeeID in Salaries table
CREATE INDEX idx_EmployeeID ON Salaries (EmployeeID);

-- Index on EmployeeID in SalaryHistory table
CREATE INDEX idx_SalaryHistory_EmployeeID ON SalaryHistory (EmployeeID);
```