Employee Payroll Management System Using SQL and Flask

Chandan Pandit
MSc Data Science
Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar
pandit27chandan@gmail.com, 202418043@dau.ac.in
GitHub Repository

June 18, 2025

Abstract

This project presents a web-based Employee Payroll Management System developed using Python (Flask) and PostgreSQL. The system automates payroll processes by enabling accurate salary computation, secure employee data storage, and streamlined management of departments, attendance, and salary structures. Core features include a normalized relational database schema, ER modeling, RESTful route integration, and automated payroll generation. By combining a user-friendly web interface with robust backend logic, the solution improves accuracy, minimizes manual effort, and supports scalable and efficient HR operations.

Payroll Management, Flask, SQL, Web Application, Database Design, ER Modeling, Normalization, Automation

1 Introduction

Payroll management is a critical operation in any organization, encompassing salary calculations, taxes, bonuses, deductions, leave balances, and timely salary disbursements. As organizations expand, manually managing payroll becomes inefficient, time-consuming, and prone to human errors.

1.1 What is Payroll Management?

Payroll management is the systematic process of calculating, distributing, and recording employee compensation. It includes components such as attendance tracking, allowances, statutory deductions, bonuses, and adherence to organizational policies and government regulations.

1.2 Why Automate Payroll Using Databases?

Automating payroll processes through a database system eliminates repetitive tasks, reduces the likelihood of errors, and ensures accuracy and consistency in calculations. A centralized database allows secure data storage, fast retrieval, and supports real-time updates and queries.

1.3 Importance in HR and Business Systems

A robust payroll system improves organizational efficiency by reducing administrative overhead, increasing transparency, and enhancing employee satisfaction. It ensures compliance with legal and tax regulations, thereby streamlining core HR functions.

This project, titled *Employee Payroll Management System using Flask and SQL*, addresses these needs by integrating a web-based interface built with Python (Flask) and a normalized relational database. The system provides modules for employee management, attendance tracking, salary structure definition, and automated payroll generation, offering a comprehensive solution for modern HR operations.

2 Problem Statement

In many organizations, payroll is still managed manually or using outdated software tools, leading to errors in salary calculations, delays in disbursements, and poor management of employee records. Manual handling also results in inefficient tracking of attendance, leaves, deductions, and tax components, increasing administrative workload and reducing overall efficiency.

To address these challenges, there is a need for a reliable, secure, and automated system that integrates payroll functionalities with employee and attendance management. This project aims to develop a web-based Employee Payroll Management System using Python (Flask) and SQL, which automates salary calculations, securely stores employee data, manages departmental and attendance records, and ensures timely and accurate payroll generation.

2.1 Objectives of the Project

- Design and implement a robust web-based payroll management system using Python (Flask) and SQL.
- Automate core payroll processes, including:
  - Employee data management
  - Salary and allowance calculations
  - Attendance and leave tracking
  - Deductions (e.g., tax, provident fund)
  - Payroll generation and payslip reporting
- Improve accuracy and reduce manual errors in payroll processing.
- Provide an intuitive interface for HR staff to manage data efficiently.
- Enable fast and secure querying, updating, and reporting of payroll information.

3 Tools and Technologies

- **Backend Language**: Python
- **Web Framework**: Flask
- **Database**: PostgreSQL
- **Query Language**: SQL
- **Database Management Tool**: pgAdmin
- **ER Diagram Tool**: draw.io / Lucidchart
- **Front-end**: HTML, CSS, Bootstrap

4 ER Diagram

The Entity-Relationship (ER) diagram models the relationships between entities in the payroll system.

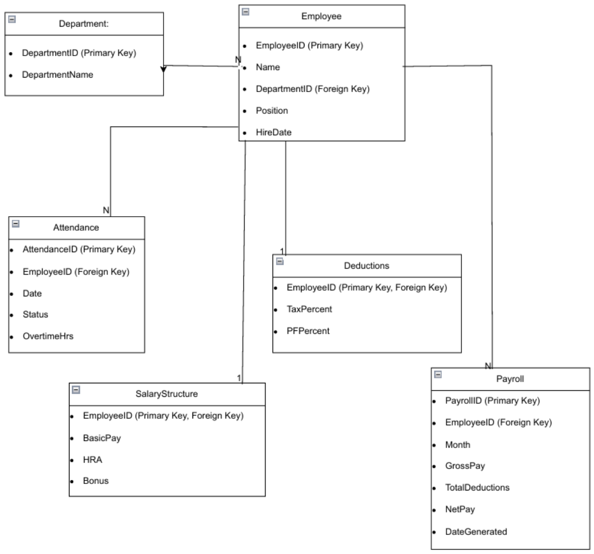


Figure 1: ER Diagram for Employee Payroll Management System

4.1 Relationships Between Entities

Relationship	**Type**
Department ↔ Employee	One-to-Many
Employee ↔ Attendance	One-to-Many
Employee ↔ SalaryStructure	One-to-One
Employee ↔ Deductions	One-to-One
Employee ↔ Payroll	One-to-Many

Table 1: Relationships Between Entities

4.2 Tables and Attributes

- **Employee**

Column	**Type**	**Description**
EmployeeID	INT (PK)	Unique employee ID
Name	VARCHAR	Full name
DepartmentID	INT (FK)	Linked to Department
Position	VARCHAR	Job title
HireDate	DATE	Date of joining

Table 2: Employee Table Schema

- **Department**

Column Description	Type
DepartmentID Unique department ID	INT (PK)
DepartmentName E.g., HR, IT, Finance	VARCHAR

Table 3: Department Table Schema

- **Attendance**

Column	Type	Description
AttendanceID	INT (PK)	Unique attendance ID
EmployeeID	INT (FK)	Linked to Employee
Date	DATE	Attendance date
Status	VARCHAR	Present / Absent / Leave / Half Day
OvertimeHrs	FLOAT	Optional

Table 4: Attendance Table Schema

- **SalaryStructure**

Column	Type	Description
EmployeeID	INT (PK, FK)	Linked to Employee
BasicPay	DECIMAL	Fixed base salary
HRA	DECIMAL	House Rent Allowance
Bonus	DECIMAL	Monthly bonus (if any)

Table 5: SalaryStructure Table Schema

- **Deductions**

Column	Type	Description
EmployeeID	INT (PK, FK)	Linked to Employee
TaxPercent	FLOAT	Tax deduction %
PFPercent	FLOAT	Provident Fund %

Table 6: Deductions Table Schema

- **Payroll**

Column	Type	Description
PayrollID	INT (PK)	Unique payroll entry
EmployeeID	INT (FK)	Linked to Employee
Month	VARCHAR	E.g., ‘May 2025’
GrossPay	DECIMAL	Calculated salary
TotalDeductions	DECIMAL	From Deductions
NetPay	DECIMAL	Final salary
DateGenerated	DATE	Date of payroll creation

Table 7: Payroll Table Schema

5 Normalization

Normalization organizes data to reduce redundancy and improve integrity, achieving 1NF, 2NF, and 3NF.

5.1 Unnormalized Form (UNF)

EmpID	Name	Dept	BasicPay	HRA	Bonus	Tax%	PF%	Attendances
101	Amit	HR	40000	8000	2000	10	12	01-May-P, 02-May-A
102	Neha	Finance	35000	7000	1500	8	10	01-May-P, 02-May-P

Table 8: Unnormalized Form

Issues: Non-atomic ‘Attendances’ column and redundant ‘Department’ data.

5.2 First Normal Form (1NF)

EmpID	Name	Dept	BasicPay	HRA	Bonus	Tax%	PF%
101	Amit	HR	40000	8000	2000	10	12
102	Neha	Finance	35000	7000	1500	8	10

Table 9: 1NF: Employee Table

EmpID	Date	Status
101	2025-05-01	Present
101	2025-05-02	Absent
102	2025-05-01	Present
102	2025-05-02	Present

Table 10: 1NF: Attendance Table

Achieved: All fields are atomic.

5.3 Second Normal Form (2NF)

EmpID	Name	Dept
101	Amit	HR
102	Neha	Finance

Table 11: 2NF: Employee Table

EmpID	BasicPay	HRA	Bonus	Tax%	PF%
101	40000	8000	2000	10	12
102	35000	7000	1500	8	10

Table 12: 2NF: SalaryStructure Table

Achieved: No partial dependencies.

5.4 Third Normal Form (3NF)

DeptID	DeptName
1	HR
2	Finance

Table 13: 3NF: Department Table

EmpID	Name	DeptID
101	Amit	1
102	Neha	2

Table 14: 3NF: Employee Table

Achieved: No transitive dependencies.

5.5 Conclusion

The database is optimized for minimal redundancy and efficient querying.

6 SQL Implementation

The SQL code creates the schema, inserts data, and performs payroll calculations.

6.1 Table Creation

```sql
CREATE TABLE Department (
   DepartmentID INT PRIMARY KEY,
   DepartmentName VARCHAR(50) NOT NULL
);

CREATE TABLE Employee (
   EmployeeID INT PRIMARY KEY,
   Name VARCHAR(100) NOT NULL,
   DepartmentID INT,
   Position VARCHAR(50),
   HireDate DATE,
   FOREIGN KEY (DepartmentID) REFERENCES Department(DepartmentID)
);

CREATE TABLE SalaryStructure (
   EmployeeID INT PRIMARY KEY,
   BasicPay DECIMAL(10,2) NOT NULL,
   HRA DECIMAL(10,2),
   Bonus DECIMAL(10,2),
   FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
);

CREATE TABLE Deductions (
   EmployeeID INT PRIMARY KEY,
   TaxPercent FLOAT,
   PFPercent FLOAT,
   FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
);

CREATE TABLE Attendance (
   AttendanceID INT PRIMARY KEY,
   EmployeeID INT,
   Date DATE,
   Status VARCHAR(20) CHECK (Status IN (’Present’, ’Absent’, ’Leave’, ’Half Day’)),
   OvertimeHrs FLOAT DEFAULT 0,
   FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
);

CREATE TABLE Payroll (
   PayrollID INT PRIMARY KEY,
   EmployeeID INT,
   Month VARCHAR(20),
   GrossPay DECIMAL(10,2),
   TotalDeductions DECIMAL(10,2),
   NetPay DECIMAL(10,2),
   DateGenerated DATE,
   FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
);
```

Listing 1: SQL Code for Table Creation

6.2 Sample Data Insertion

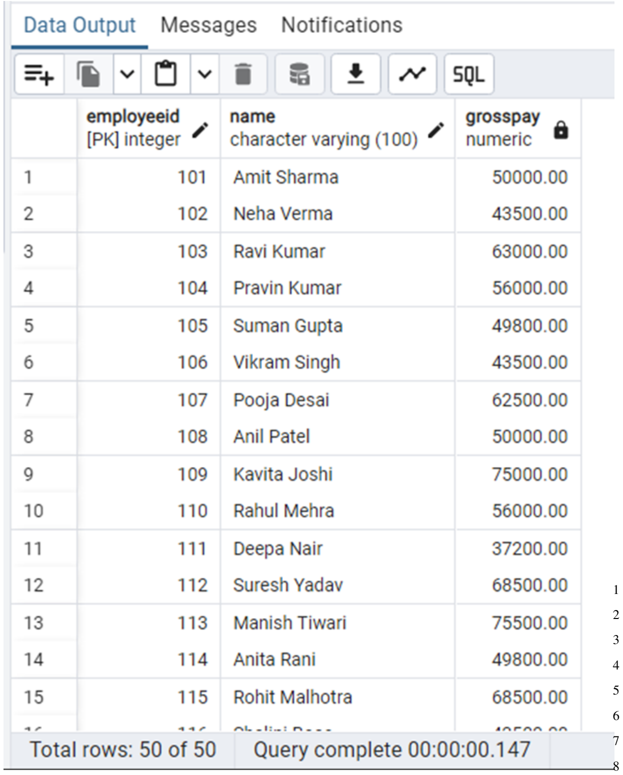
```sql
INSERT INTO Department VALUES
(1, ’HR’),
(2, ’Finance’),
(3, ’IT’),
(4, ’Sales’);

INSERT INTO Employee VALUES
(101, ’Amit Sharma’, 1, ’HR Manager’, ’2022-03-15’),
(102, ’Neha Verma’, 2, ’Accountant’, ’2021-11-20’),
(103, ’Ravi Kumar’, 3, ’Software Engineer’, ’2023-01-10’);

INSERT INTO SalaryStructure VALUES
(101, 40000, 8000, 2000),
(102, 35000, 7000, 1500),
(103, 50000, 10000, 3000);

INSERT INTO Deductions VALUES
(101, 10.0, 12.0),
(102, 8.0, 10.0),
(103, 12.0, 15.0);

INSERT INTO Attendance VALUES
(1, 101, ’2025-05-01’, ’Present’, 2),
(2, 101, ’2025-05-02’, ’Absent’, 0),
(3, 102, ’2025-05-01’, ’Present’, 1),
(4, 103, ’2025-05-01’, ’Present’, 3);
```

Listing 2: Sample Data Insertion for Tables

6.3 Monthly Gross Salary Calculation

```sql
SELECT
   E.EmployeeID,
   E.Name,
   (S.BasicPay + COALESCE(S.HRA, 0) + COALESCE(S.Bonus, 0)) AS GrossPay
FROM Employee E
JOIN SalaryStructure S ON E.EmployeeID = S.EmployeeID;
```

Listing 3: Monthly Gross Salary Calculation



	employeeid [PK] integer	name character varying (100)	grosspay numeric
1	101	Amit Sharma	50000.00
2	102	Neha Verma	43500.00
3	103	Ravi Kumar	63000.00
4	104	Pravin Kumar	56000.00
5	105	Suman Gupta	49800.00
6	106	Vikram Singh	43500.00
7	107	Pooja Desai	62500.00
8	108	Anil Patel	50000.00
9	109	Kavita Joshi	75000.00
10	110	Rahul Mehra	56000.00
11	111	Deepa Nair	37200.00
12	112	Suresh Yadav	68500.00
13	113	Manish Tiwari	75500.00
14	114	Anita Rani	49800.00
15	115	Rohit Malhotra	68500.00

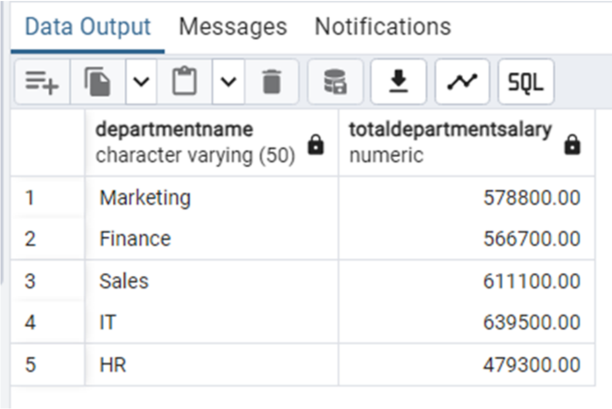
Total rows: 50 of 50   Query complete 00:00:00.147

Figure 2: Monthly Gross Salary Calculation

6.4 Department-wise Monthly Salary Expense

```
SELECT
   D.DepartmentName,
   SUM(S.BasicPay + COALESCE(S.HRA, 0) +
       COALESCE(S.Bonus, 0)) AS
       TotalDepartmentSalary
FROM Employee E
JOIN Department D ON E.DepartmentID = D.
    DepartmentID
JOIN SalaryStructure S ON E.EmployeeID = S.
    EmployeeID
GROUP BY D.DepartmentName;
```

Listing 4: Department-wise Monthly Salary Expense



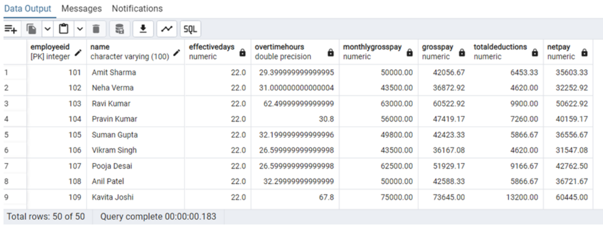
	departmentname character varying (50)	totaldepartmentsalary numeric
1	Marketing	578800.00
2	Finance	566700.00
3	Sales	611100.00
4	IT	639500.00
5	HR	479300.00

Figure 3: Department-wise Monthly Salary Expense

6.5 Net Pay Calculation

```
SELECT
   e.EmployeeID,
   e.Name,
   COALESCE(SUM(
      CASE
         WHEN a.Status = ’Present’ THEN 1
         WHEN a.Status = ’Half Day’ THEN 0.5
         ELSE 0
      END
   ), 0) AS EffectiveDays,

   COALESCE(SUM(a.OvertimeHrs), 0) AS
       OvertimeHours,

   (s.BasicPay + COALESCE(s.HRA, 0) + COALESCE(
       s.Bonus, 0)) AS MonthlyGrossPay,

   ROUND((
      (s.BasicPay + COALESCE(s.HRA, 0) +
          COALESCE(s.Bonus, 0) +
      (COALESCE(SUM(a.OvertimeHrs), 0) * (s.
          BasicPay / (30.0 * 8.0)) * 1.5)) *
      COALESCE(SUM(
         CASE
            WHEN a.Status = ’Present’ THEN 1
            WHEN a.Status = ’Half Day’ THEN 0.5
            ELSE 0
         END
      ), 0) / 30.0
   )::NUMERIC, 2) AS GrossPay,

   ROUND((
      (s.BasicPay * COALESCE(d.TaxPercent, 0) /
           100 + s.BasicPay * COALESCE(d.
          PFPercent, 0) / 100) *
      COALESCE(SUM(
         CASE
            WHEN a.Status = ’Present’ THEN 1
            WHEN a.Status = ’Half Day’ THEN 0.5
            ELSE 0
         END
      ), 0) / 30.0
   )::NUMERIC, 2) AS TotalDeductions,

   ROUND((
      (
          (s.BasicPay + COALESCE(s.HRA, 0) +
              COALESCE(s.Bonus, 0) +
          (COALESCE(SUM(a.OvertimeHrs), 0) * (s.
              BasicPay / (30.0 * 8.0)) * 1.5))
              *
          COALESCE(SUM(
             CASE
                WHEN a.Status = ’Present’ THEN 1
                WHEN a.Status = ’Half Day’ THEN
                    0.5
                ELSE 0
             END
          ), 0) / 30.0
       ) - (
          (s.BasicPay * COALESCE(d.TaxPercent,
              0) / 100 + s.BasicPay * COALESCE(
              d.PFPercent, 0) / 100) *
          COALESCE(SUM(
             CASE
                WHEN a.Status = ’Present’ THEN 1
                WHEN a.Status = ’Half Day’ THEN
                    0.5
                ELSE 0
             END
          ), 0) / 30.0
       )
   )::NUMERIC, 2) AS NetPay

FROM
   Employee e
JOIN
   SalaryStructure s ON e.EmployeeID = s.
       EmployeeID
JOIN
   Deductions d ON e.EmployeeID = d.EmployeeID
LEFT JOIN
   Attendance a ON e.EmployeeID = a.EmployeeID
GROUP BY
   e.EmployeeID, e.Name, s.BasicPay, s.HRA, s.
       Bonus, d.TaxPercent, d.PFPercent
ORDER BY
   e.EmployeeID;
```

Listing 5: Net Pay Calculation per Employee



Data Output | Messages | Notifications

	employeeid [PK] integer	name character varying (100)	effectivedays numeric	overtimehours double precision	monthlygrosspay numeric	grosspay numeric	totaldeductions numeric	netpay numeric
1	101	Amit Sharma	22.0	29.399999999999995	50000.00	42056.67	6453.33	35603.33
2	102	Neha Verma	22.0	31.000000000000004	43500.00	36872.92	4620.00	32252.92
3	103	Ravi Kumar	22.0	62.49999999999999	63000.00	60522.92	9900.00	50622.92
4	104	Pravin Kumar	22.0	30.8	56000.00	47419.17	7260.00	40159.17
5	105	Suman Gupta	22.0	32.199999999999996	49800.00	42423.33	5866.67	36556.67
6	106	Vikram Singh	22.0	26.599999999999998	43500.00	36167.08	4620.00	31547.08
7	107	Pooja Desai	22.0	26.599999999999998	62500.00	51929.17	9166.67	42762.50
8	108	Anil Patel	22.0	32.29999999999999	50000.00	42588.33	5866.67	36721.67
9	109	Kavita Joshi	22.0	67.8	75000.00	73645.00	13200.00	60445.00

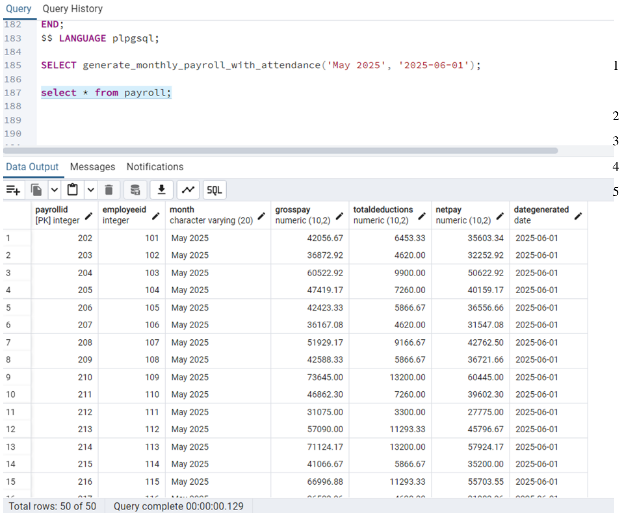
Total rows: 50 of 50 Query complete 00:00:00.183

Figure 4: Net Pay Calculation

6.6 Payroll Generation Function

```
CREATE OR REPLACE FUNCTION
    generate_monthly_payroll_with_attendance(
    input_month TEXT, run_date DATE)
RETURNS void AS $$
DECLARE
   emp_id INTEGER;
   basic NUMERIC(10,2) := 0;
   hra NUMERIC(10,2) := 0;
   bonus NUMERIC(10,2) := 0;
   gross_monthly NUMERIC(10,2);
   tax_pct FLOAT := 0;
   pf_pct FLOAT := 0;
   effective_days FLOAT := 0;
   overtime_hours FLOAT := 0;
   overtime_pay NUMERIC(10,2);
   gross NUMERIC(10,2);
   deductions NUMERIC(10,2);
   net NUMERIC(10,2);
   r RECORD;
BEGIN
   FOR r IN SELECT e.EmployeeID FROM Employee e
        LOOP
      emp_id := r.EmployeeID;
      RAISE NOTICE ’Processing EmployeeID: %’,
          emp_id;

      SELECT COALESCE(ss.BasicPay, 0), COALESCE
           (ss.HRA, 0), COALESCE(ss.Bonus, 0)
      INTO basic, hra, bonus
      FROM SalaryStructure ss
      WHERE ss.EmployeeID = emp_id;
      IF NOT FOUND THEN
         RAISE NOTICE ’No SalaryStructure for
             EmployeeID %’, emp_id;
         CONTINUE;
      END IF;

      gross_monthly := basic + hra + bonus;

      SELECT COALESCE(d.TaxPercent, 0),
          COALESCE(d.PFPercent, 0)
      INTO tax_pct, pf_pct
      FROM Deductions d
      WHERE d.EmployeeID = emp_id;
      IF NOT FOUND THEN
         RAISE NOTICE ’No Deductions for
             EmployeeID %’, emp_id;
         CONTINUE;
      END IF;

      SELECT
         COALESCE(SUM(
            CASE
               WHEN a.Status = ’Present’ THEN 1
               WHEN a.Status = ’Half Day’ THEN
                   0.5
               ELSE 0
            END
         ), 0),
         COALESCE(SUM(a.OvertimeHrs), 0)
      INTO effective_days, overtime_hours
      FROM Attendance a
      WHERE a.EmployeeID = emp_id
        AND TRIM(TO_CHAR(a.Date, ’Month’)) || ’
           ’ || TO_CHAR(a.Date, ’YYYY’) =
           input_month;

      overtime_pay := ROUND((overtime_hours * (
          basic / (30.0 * 8.0)) * 1.5)::
          NUMERIC, 2);
      gross := ROUND(((gross_monthly +
          overtime_pay) * effective_days /
          30.0)::NUMERIC, 2);
        deductions := ROUND(((basic * (tax_pct + pf_pct) / 100) * effective_days / 30.0)::NUMERIC, 2);
        net := ROUND((gross - deductions)::NUMERIC, 2);

        INSERT INTO Payroll (
            PayrollID, EmployeeID, Month, GrossPay, TotalDeductions, NetPay, DateGenerated
        ) VALUES (
            nextval(’payroll_id_seq’),
            emp_id,
            input_month,
            gross,
            deductions,
            net,
            run_date
        );
    END LOOP;
END;
$$ LANGUAGE plpgsql;

-- Run the function for May 2025
SELECT generate_monthly_payroll_with_attendance(’May 2025’, ’2025-06-01’);
```

Listing 6: Monthly Payroll Generation Function



```
END;
$$ LANGUAGE plpgsql;

SELECT generate_monthly_payroll_with_attendance('May 2025', '2025-06-01');

select * from payroll;
```

	payrollid [PK] integer	employeeid integer	month character varying (20)	grosspay numeric (10,2)	totaldeductions numeric (10,2)	netpay numeric (10,2)	dategenerated date
1	202	101	May 2025	42056.67	6453.33	35603.34	2025-06-01
2	203	102	May 2025	36872.92	4620.00	32252.92	2025-06-01
3	204	103	May 2025	60522.92	9900.00	50622.92	2025-06-01
4	205	104	May 2025	47419.17	7260.00	40159.17	2025-06-01
5	206	105	May 2025	42423.33	5866.67	36556.66	2025-06-01
6	207	106	May 2025	36167.08	4620.00	31547.08	2025-06-01
7	208	107	May 2025	51929.17	9166.67	42762.50	2025-06-01
8	209	108	May 2025	42588.33	5866.67	36721.66	2025-06-01
9	210	109	May 2025	73645.00	13200.00	60445.00	2025-06-01
10	211	110	May 2025	46862.30	7260.00	39602.30	2025-06-01
11	212	111	May 2025	31075.00	3300.00	27775.00	2025-06-01
12	213	112	May 2025	57090.00	11293.33	45796.67	2025-06-01
13	214	113	May 2025	71124.17	13200.00	57924.17	2025-06-01
14	215	114	May 2025	41066.67	5866.67	35200.00	2025-06-01
15	216	115	May 2025	66996.88	11293.33	55703.55	2025-06-01

Total rows: 50 of 50 Query complete 00:00:00.129

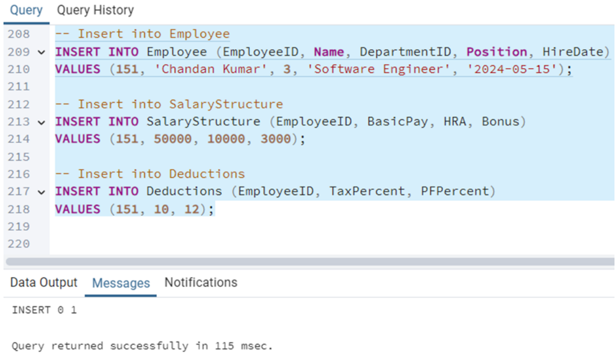
Figure 5: Payroll Generation Function Output

7 Testing and Validation

7.1 Test Case 1: Insert Employee and Salary Data

```
INSERT INTO Department (DepartmentID, DepartmentName) VALUES (3, ’IT’);
INSERT INTO Employee (EmployeeID, Name, DepartmentID, Position, HireDate)
VALUES (151, ’Chandan Kumar’, 3, ’Software Engineer’, ’2024-05-15’);
INSERT INTO SalaryStructure (EmployeeID, BasicPay, HRA, Bonus)
VALUES (151, 50000, 10000, 3000);
INSERT INTO Deductions (EmployeeID, TaxPercent, PFPercent)
VALUES (151, 10, 12);
```

Expected Result: Data inserted successfully; no foreign key errors.



```
-- Insert into Employee
INSERT INTO Employee (EmployeeID, Name, DepartmentID, Position, HireDate)
VALUES (151, 'Chandan Kumar', 3, 'Software Engineer', '2024-05-15');

-- Insert into SalaryStructure
INSERT INTO SalaryStructure (EmployeeID, BasicPay, HRA, Bonus)
VALUES (151, 50000, 10000, 3000);

-- Insert into Deductions
INSERT INTO Deductions (EmployeeID, TaxPercent, PFPercent)
VALUES (151, 10, 12);
```

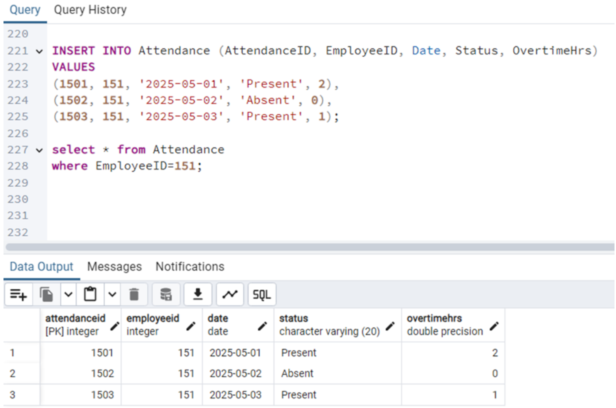
```
INSERT 0 1

Query returned successfully in 115 msec.
```

Figure 6: Insert Employee and Salary Data

7.2 Test Case 2: Record Attendance

```
INSERT INTO Attendance (AttendanceID, EmployeeID, Date, Status, OvertimeHrs)
VALUES
(301, 103, ’2025-05-01’, ’Present’, 2),
(302, 101, ’2025-05-02’, ’Absent’, 0),
(303, 103, ’2025-05-03’, ’Present’, 1);
```

Expected Result: 3 records inserted for May attendance.



```
INSERT INTO Attendance (AttendanceID, EmployeeID, Date, Status, OvertimeHrs)
VALUES
(1501, 151, '2025-05-01', 'Present', 2),
(1502, 151, '2025-05-02', 'Absent', 0),
(1503, 151, '2025-05-03', 'Present', 1);

select * from Attendance
where EmployeeID=151;
```

	attendanceid [PK] integer	employeeid integer	date date	status character varying (20)	overtimehrs double precision
1	1501	151	2025-05-01	Present	2
2	1502	151	2025-05-02	Absent	0
3	1503	151	2025-05-03	Present	1

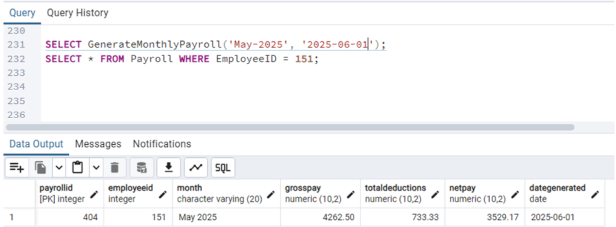
Figure 7: Record Attendance

7.3 Test Case 3: Run Payroll Function

```
SELECT generate_monthly_payroll_with_attendance
    (’May 2025’, ’2025-06-01’);
```

```sql
SELECT * FROM Payroll WHERE EmployeeID = 151;
```

Expected Result: One row added to the Payroll table for EmployeeID 151.



Query Query History

```sql
SELECT GenerateMonthlyPayroll('May-2025', '2025-06-01');
SELECT * FROM Payroll WHERE EmployeeID = 151;
```

Data Output Messages Notifications

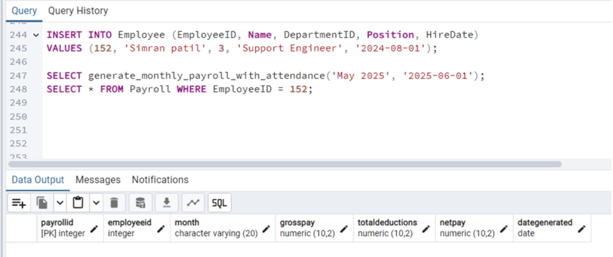
	payrollid [PK] integer	employeeid integer	month character varying (20)	grosspay numeric (10,2)	totaldeductions numeric (10,2)	netpay numeric (10,2)	dategenerated date
1	404	151	May 2025	4262.50	733.33	3529.17	2025-06-01

Figure 8: Run Payroll Function

7.4 Test Case 4: Missing Data Edge Case

```sql
INSERT INTO Employee (EmployeeID, Name,
    DepartmentID, Position, HireDate)
VALUES (152, ’Simran Patil’, 3, ’Support
    Engineer’, ’2024-08-01’);
SELECT generate_monthly_payroll_with_attendance
    (’May 2025’, ’2025-06-01’);
```

Expected Result: Employee 152 is skipped due to missing SalaryStructure entry.



Query Query History

```sql
INSERT INTO Employee (EmployeeID, Name, DepartmentID, Position, HireDate)
VALUES (152, 'Simran patil', 3, 'Support Engineer', '2024-08-01');

SELECT generate_monthly_payroll_with_attendance('May 2025', '2025-06-01');
SELECT * FROM Payroll WHERE EmployeeID = 152;
```

Data Output Messages Notifications

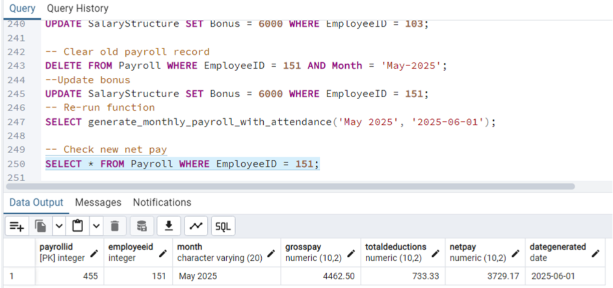
	payrollid [PK] integer	employeeid integer	month character varying (20)	grosspay numeric (10,2)	totaldeductions numeric (10,2)	netpay numeric (10,2)	dategenerated date

Figure 9: Missing Data Edge Case

7.5 Test Case 5: Update Bonus and Recalculate Payroll

```sql
UPDATE SalaryStructure SET Bonus = 6000 WHERE
    EmployeeID = 151;
DELETE FROM Payroll WHERE EmployeeID = 151 AND
    Month = ’May 2025’;
SELECT generate_monthly_payroll_with_attendance
    (’May 2025’, ’2025-06-01’);
SELECT * FROM Payroll WHERE EmployeeID = 151;
```

Expected Result: NetPay reflects the updated bonus amount.



Query Query History

```sql
UPDATE SalaryStructure SET Bonus = 6000 WHERE EmployeeID = 103;

-- Clear old payroll record
DELETE FROM Payroll WHERE EmployeeID = 151 AND Month = 'May-2025';
--Update bonus
UPDATE SalaryStructure SET Bonus = 6000 WHERE EmployeeID = 151;
-- Re-run function
SELECT generate_monthly_payroll_with_attendance('May 2025', '2025-06-01');

-- Check new net pay
SELECT * FROM Payroll WHERE EmployeeID = 151;
```

Data Output Messages Notifications

	payrollid [PK] integer	employeeid integer	month character varying (20)	grosspay numeric (10,2)	totaldeductions numeric (10,2)	netpay numeric (10,2)	dategenerated date
1	455	151	May 2025	4462.50	733.33	3729.17	2025-06-01

Figure 10: Update Bonus and Recalculate Payroll

8 User Interface Overview

8.1 Home Page

The system provides a clean and user-friendly web interface as the entry point for HR and administrative users. The home page (shown below) acts as the central navigation hub for all key payroll functionalities.

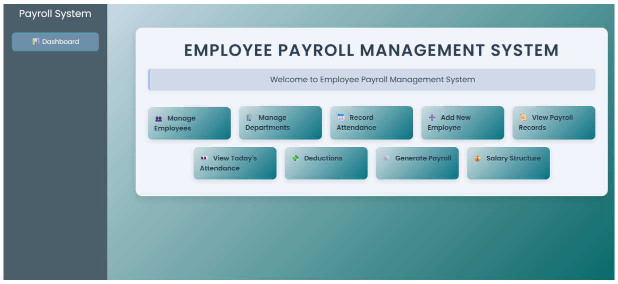


Figure 11: Front Page of the Employee Payroll Management System

The front page is designed using HTML, CSS, and Bootstrap and is styled to offer a professional dashboard layout. It includes the following interactive buttons:

- **Manage Employees**: Add, view, update, or delete employee records.
- **Manage Departments**: Define and manage various departments within the organization.
- **Record Attendance**: Capture daily attendance for payroll calculations.
- **Add New Employee**: Quickly register a new employee into the system.
- **View Payroll Records**: Display previously generated payroll entries.
- **View Today’s Attendance**: Monitor daily attendance logs in real-time.

- **Deductions**: Manage deduction policies such as tax and provident fund.
- **Generate Payroll**: Automatically compute salary based on rules and attendance.
- **Salary Structure**: Define basic pay, allowances, and deductions for employees.

The dashboard is accessible through the left-side navigation panel, providing quick access to modules. The system is built using Flask for backend routing and PostgreSQL for data storage.

9 Conclusion

The Employee Payroll Management System automates monthly salary generation based on employee attendance and predefined salary components. By integrating a web-based interface using Python (Flask) with a structured PostgreSQL database, the system ensures data consistency, accuracy in calculations, and transparency in payroll processing. This solution significantly reduces manual errors, enhances operational efficiency, and simplifies payroll-related tasks for HR departments.

10 Future Scope

- Enhance the frontend UI with modern JavaScript frameworks such as React or Vue.js for a more dynamic and responsive user experience.
- Integrate the system with a full-featured HR management portal to support recruitment, performance tracking, and leave management.
- Implement automated email notifications to send monthly payslips and alerts to employees.
- Add a built-in tax calculator with real-time statutory compliance checks based on updated government policies.
- Enable multi-user roles with authentication for HR, finance, and admin access control.