Problem Statement:

Efficiently managing employee payroll is crucial for organizations, ensuring timely and accurate compensation for employees while minimizing the risk of errors. Manual payroll processes, especially in organizations with large employee counts, varying salary structures and overtime calculations are prone to delays, inaccuracies, and security vulnerabilities. These inefficiencies can negatively impact employee satisfaction, company morale, and operational efficiency.

The challenge is to develop a **Payroll Management System Database** that automates the payroll process. The system should streamline payroll calculations by integrating employee information, attendance records, overtime hours, tax withholdings, benefits, and bank details to generate accurate payrolls, thus eliminating manual errors and ensuring secure transactions.

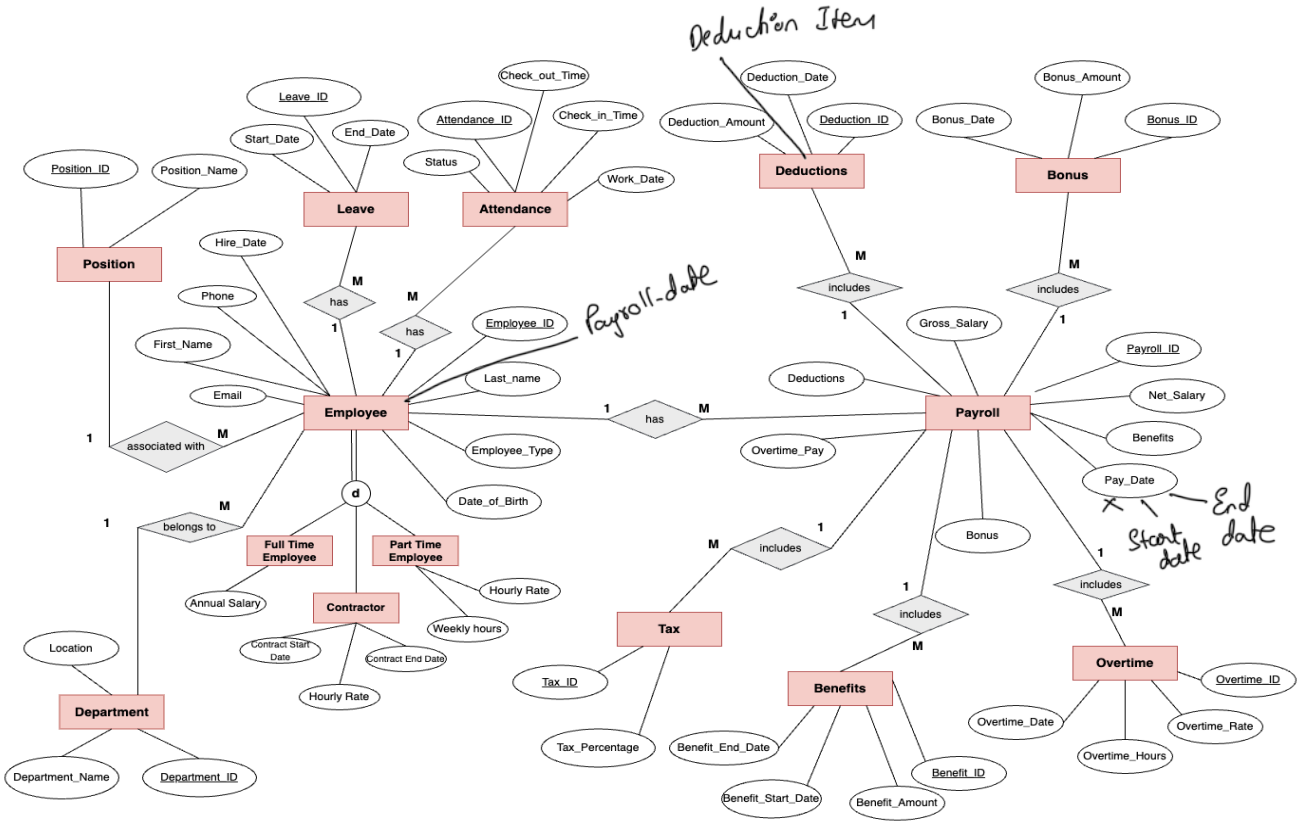
Problem Definition:

The **Payroll Management System Database** aims to automate the complex task of payroll management. The system will handle the following key functions:

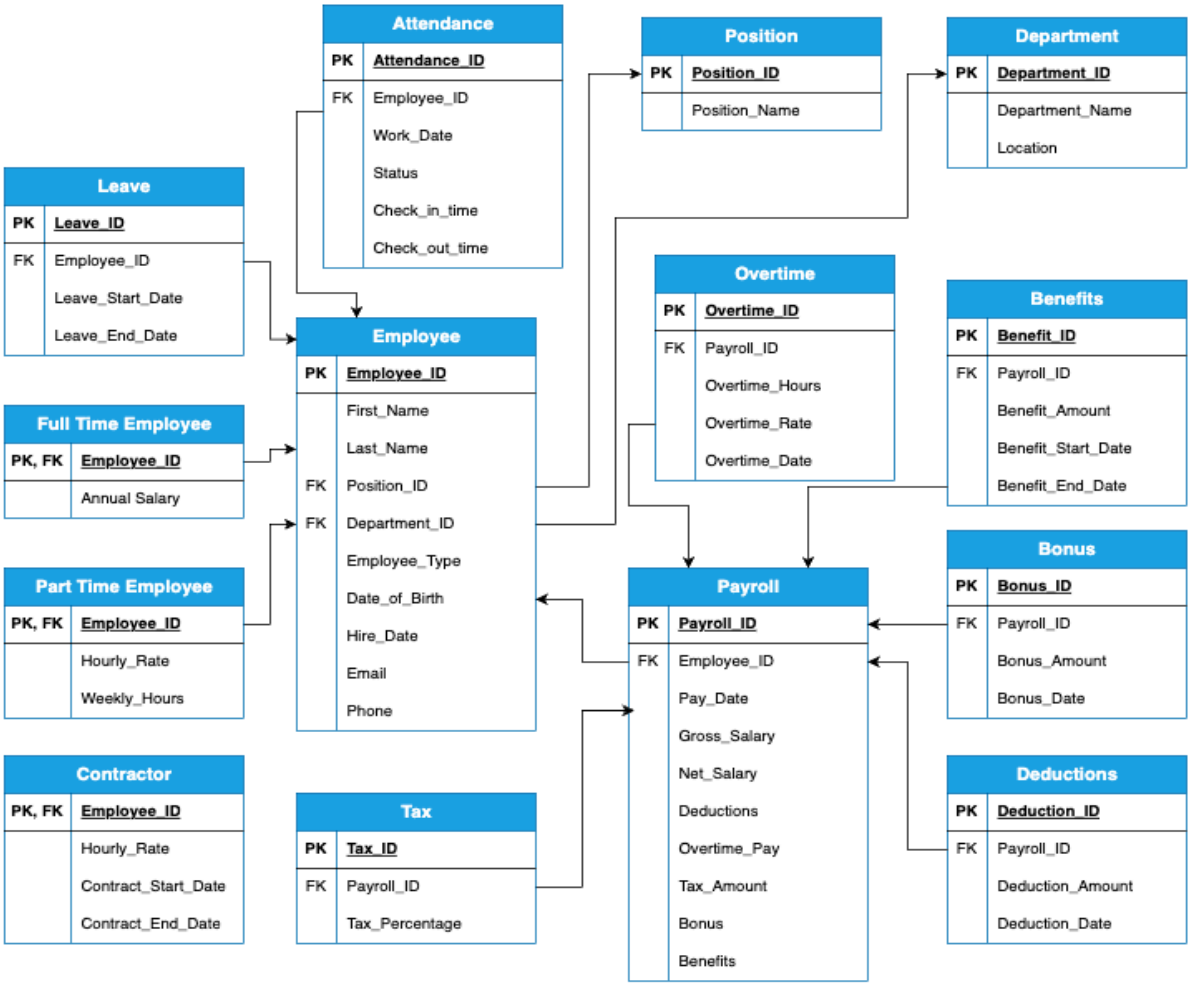
- **Employee Data Management**: Storing comprehensive employee details, such as name, ID, bank account information, and salary or stipend amount.
- **Attendance and Leave Tracking**: Tracking employee attendance, leaves, and half-days to ensure that payroll calculations reflect the accurate number of working days.
- **Payroll Calculation**: Automating the calculation of salaries or stipends based on employee attendance and leave data, applying company-specific remuneration rules.
- **Overtime Calculation:** Accurately computing overtime payments based on employee overtime hours and pre-defined rates, ensuring fair compensation and adherence to labour regulations.
- **Tax Withholdings:** Automatically applying tax withholding rules based on the employee's tax bracket and other applicable regulations, ensuring compliance with governmental tax policies.
- **Benefits Management:** Integrating benefits such as health insurance, retirement plans, and allowances into the payroll system to account for both pre-tax and post-tax deductions, providing a clear view of the total compensation package.
- **Bank Integration**: Facilitating secure and timely salary deposits by integrating the system with employee bank details.
- **Data Security and Authorization**: Ensuring that sensitive employee data is secure through robust access control mechanisms, protecting information from unauthorised access.

This system will replace manual payroll processes, improving accuracy, reducing processing time, and mitigating security risks. It will be designed to be scalable, accommodating both small and large organizations with varying numbers of employees.

ER Diagram:



Relational Model:



Entities:

Employee

- **PK**: Employee_ID
- **FK**: Position_ID (references Position), Department_ID (references Department)
- **Attributes**: First_Name, Last_Name, Email, Phone, Date_of_Birth, Hire_Date, Employee_Type

Department

- **PK**: Department_ID
- **Attributes**: Department_Name, Location

Position

- **PK**: Position_ID
- **Attributes**: Position_Name

FullTimeEmployee

- **PK & FK**: Employee_ID (references Employee)
- **Attributes**: Annual_Salary

PartTimeEmployee

- **PK & FK**: Employee_ID (references Employee)
- **Attributes**: Hourly_Rate, Weekly_Hours

Contractor

- **PK & FK**: Employee_ID (references Employee)
- **Attributes**: Hourly_Rate, Contract_Start_Date, Contract_End_Date

Attendance

- **PK**: Attendance_ID
- **FK**: Employee_ID (references Employee)
- **Attributes**: Work_Date, Status, Check_in_Time, Check_out_Time

Leave

- **PK**: Leave_ID
- **FK**: Employee_ID (references Employee)
- **Attributes**: Leave_Type, Leave_Start_Date, Leave_End_Date

Payroll

- **PK**: Payroll_ID
- **FK**: Employee_ID (references Employee)
- **Attributes**: Pay_Date, Gross_Salary, Net_Salary, Deductions, Overtime_Pay, Bonus

Bonus

- **PK**: Bonus_ID
- **FK**: Payroll_ID (references Payroll)
- **Attributes**: Bonus_Amount, Bonus_Date

Deductions

- **PK**: Deduction_ID
- **FK**: Payroll_ID (references Payroll)
- **Attributes**: Deduction_Type, Deduction_Amount, Deduction_Date

Benefits

- **PK**: Benefit_ID
- **FK**: Employee_ID (references Employee)
- **Attributes**: Benefit_Type, Benefit_Amount, Benefit_Start_Date, Benefit_End_Date

Tax

- **PK**: Tax_ID
- **FK**: Payroll_ID (references Payroll)
- **Attributes**: Tax_Percentage, Tax_Amount

Overtime

- **PK**: Overtime_ID
- **FK**: Payroll_ID (references Payroll)
- **Attributes**: Overtime_Hours, Overtime_Rate, Overtime_Date

Relations:

1. Employee-Department:
    - Relationship: Each employee belongs to one department.
    - Cardinality: 1 Employee → 1 Department, 1 Department → M Employees
    - PK/FK: Employee.Department_ID (FK) references Department.Department_ID (PK)
2. Employee-Position:
    - Relationship: Each employee holds one position.
    - Cardinality: 1 Employee → 1 Position, 1 Position → M Employees
    - PK/FK: Employee.Position_ID (FK) references Position.Position_ID (PK)
3. Employee-Attendance:
    - Relationship: Each employee has multiple attendance records.
    - Cardinality: 1 Employee → M Attendance records
    - PK/FK: Attendance.Employee_ID (FK) references Employee.Employee_ID (PK)
4. Payroll-Employee:
    - Relationship: Each employee has one payroll record.
    - Cardinality: 1 Employee → 1 Payroll, 1 Payroll → 1 Employee
    - PK/FK: Payroll.Employee_ID (FK) references Employee.Employee_ID (PK)
5. Payroll-Bonus:
    - Relationship: A payroll includes one or more bonuses.
    - Cardinality: 1 Payroll → M Bonuses, 1 Bonus → 1 Payroll
    - PK/FK: Bonus.Payroll_ID (FK) references Payroll.Payroll_ID (PK)
6. Payroll-Deductions:

    - Relationship: A payroll includes multiple deductions.
    - Cardinality: 1 Payroll → M Deductions, 1 Deduction → 1 Payroll
    - PK/FK: Deductions.Payroll_ID (FK) references Payroll.Payroll_ID (PK)
7. Payroll-Overtime:
    - Relationship: A payroll can include multiple overtime entries.
    - Cardinality: 1 Payroll → M Overtime, 1 Overtime → 1 Payroll
    - PK/FK: Overtime.Payroll_ID (FK) references Payroll.Payroll_ID (PK)
8. Payroll-Tax:
    - Relationship: A payroll includes one tax record.
    - Cardinality: 1 Payroll → M Taxes, 1 Tax → 1 Payroll
    - PK/FK: Tax.Payroll_ID (FK) references Payroll.Payroll_ID (PK)
9. Employee-Leave:
    - Relationship: Each employee has multiple leave records.
    - Cardinality: 1 Employee → M Leaves, 1 Leave → 1 Employee
    - PK/FK: Leave.Employee_ID (FK) references Employee.Employee_ID (PK)
10. Employee-Benefits:
    - Relationship: Each employee has multiple benefit records.
    - Cardinality: 1 Employee → M Benefits, 1 Benefit → 1 Employee
    - PK/FK: Benefits.Employee_ID (FK) references Employee.Employee_ID (PK)
11. Employee-Full-Time/Part-Time/Contractor:
    - Relationship: Generalization (d) relationship.
    - Cardinality: 1 Employee → 1 Full-Time/Part-Time/Contractor
    - PK/FK: FullTimeEmployee.Employee_ID (FK), PartTimeEmployee.Employee_ID (FK), Contractor.Employee_ID (FK) all reference Employee.Employee_ID (PK)
    - The generalization enforces that each employee can only be one of the subtypes.

Dimensions and Hierarchies:

1. Employee Dimension

Attributes (Levels):

- EmployeeID
- FirstName
- LastName
- EmployeeType (FullTime, PartTime, Contractor)
- HireDate
- Phone
- Email

Hierarchies:

- Employee Type Hierarchy:
    1. Employee (Superclass)
        - ❖ FullTimeEmployee (Subclass)
        - ❖ PartTimeEmployee (Subclass)
        - ❖ Contractor (Subclass)

2. Department Dimension

Attributes (Levels):

- DepartmentID
- DepartmentName
- Location

3. Position Dimension

Attributes (Levels):

- PositionID
- PositionName
- BonusPercentage

Measures:

Additive Measures: These measures can be summed across dimensions.

- Total Deductions
- Total Taxes
- Total Overtime Pay
- Total overtime Hours
- Total leave Days
- Total Attendance Days
- Total Full time Employees
- Total part time Employees
- Total Contractors

Calculated Measures: These measures are derived from additive measures through calculations.

- Average Salary
- Average Hours Worked

Count Measures: These measures represent counts of occurrences or entities.

- Number of Employees(full time, part time or contractors)
- Number of leaves
- Number of Benefits enrollment

PostgreSQL Implementation:

1. **Employee Table** (100 employees)

```
INSERT INTO Employee (Employee_ID, First_Name, Last_Name, Position_ID, Department_ID, Date_of_Birth, Hire_Date, Email, Phone)
VALUES
(generate_series(1, 100),
(ARRAY['John', 'Jane', 'Alice', 'Bob', 'Carol', 'Dave', 'Eve', 'Ryan', 'Rachel', 'Tom'])[floor(random() * 10 + 1)],
(ARRAY['Smith', 'Johnson', 'Williams', 'Brown', 'Jones', 'Garcia', 'Miller'])[floor(random() * 7 + 1)],
floor(random() * 10 + 1)::int,
floor(random() * 5 + 1)::int,
date '1980-01-01' + (floor(random() * 14600)) * interval '1 day',
date '2010-01-01' + (floor(random() * 5114)) * interval '1 day',
'email'||generate_series(1, 100)||'@company.com',
'(' || floor(random() * 900 + 100)::int || ') ' || floor(random() * 900 + 100)::int || '-' || floor(random() * 9000 + 1000)::int);
```

Messages

	ee_id [PK] integer	first_name text	last_name text	position_id integer	department_id integer	date_of_birth date	hire_date date	email text	phone text
1	1	Tom	Smith	4	4	1987-01-25	2021-06-14	email1@company.com	(408) 326-8948
2	2	Bob	Williams	6	4	1993-12-09	2018-01-24	email2@company.com	(979) 600-8967
3	3	Eve	Garcia	3	4	1983-10-25	2010-04-05	email3@company.com	(901) 358-6109
4	4	Carol	Smith	7	4	1999-12-28	2013-03-30	email4@company.com	(382) 354-3816
5	5	Tom	Smith	3	1	2002-06-18	2015-06-06	email5@company.com	(226) 731-6169
6	6	Eve	Brown	5	4	2009-03-28	2021-05-09	email6@company.com	(940) 753-8485

Total rows: 100 of 100 | Query complete 00:00:00.056 | Ln 207, Col 59

2. Attendance Table (Each employee will have attendance records for 30 random workdays within a 3-month timeframe.)

```
INSERT INTO Attendance (Attendance_ID, Employee_ID, Work_Date, Status, Check_in_time, Check_out_time)
SELECT
    generate_series(1, 3000) AS Attendance_ID,
    Employee_ID,
    date '2024-01-01' + (floor(random() * 90)) * interval '1 day' AS Work_Date,
    (ARRAY['Present', 'Absent'])[floor(random() * 2 + 1)] AS Status,
    time '09:00:00' + (floor(random() * 60)) * interval '1 minute' AS Check_in_time,
    time '17:00:00' + (floor(random() * 60)) * interval '1 minute' AS Check_out_time
FROM Employee
ORDER BY random()
LIMIT 3000;
```

Data Output | Messages | Graph Visualiser | Notifications

	attendance_id [PK] integer	employee_id integer	work_date date	status text	check_in_time time without time zone	check_out_time time without time zone
1	1	54	2024-01-01	Present	09:00:00	17:00:00
2	2	54	2024-01-01	Present	09:00:00	17:00:00
3	3	54	2024-01-01	Present	09:00:00	17:00:00
4	4	54	2024-01-01	Present	09:00:00	17:00:00
5	5	54	2024-01-01	Present	09:00:00	17:00:00
6	6	54	2024-01-01	Present	09:00:00	17:00:00

Total rows: 1000 of 3000 | Query complete 00:00:00.055

3. **Position Table** (10 positions)

```
INSERT INTO Position (Position_ID, Position_Name)
VALUES
(1, 'Manager'),
(2, 'Software Engineer'),
(3, 'Data Scientist'),
(4, 'HR Specialist'),
(5, 'Accountant'),
(6, 'Project Manager'),
(7, 'Developer'),
```

(8, 'Tester'),
(9, 'Designer'),
(10, 'Marketing Specialist');

	position_id [PK] integer	position_name text
1	1	Manager
2	2	Software Engineer
3	3	Data Scientist
4	4	HR Specialist
5	5	Accountant
6	6	Project Manager
7	7	Developer
8	8	Tester
9	9	Designer
10	10	Marketing Specialist

4. **Department Table** (5 departments)

INSERT INTO Department (Department_ID, Department_Name, Location)
VALUES
(1, 'HR', 'New York'),
(2, 'Finance', 'Boston'),
(3, 'IT', 'San Francisco'),
(4, 'Marketing', 'Seattle'),
(5, 'Operations', 'Chicago');

	department_id [PK] integer	department_name text	location text
1	1	HR	New York
2	2	Finance	Boston
3	3	IT	San Francisco
4	4	Marketing	Seattle
5	5	Operations	Chicago

5. **Overtime Table** (500 records)

```
SELECT
   generate_series(1, 500) AS Overtime_ID,
   floor(random() * 100 + 1)::int AS Payroll_ID,
   floor(random() * 10 + 1)::int AS Overtime_Hours,
   floor(random() * 50 + 10)::decimal(5, 2) AS Overtime_Rate,
   date '2023-01-01' + (floor(random() * 180)) * interval '1 day' AS Overtime_Date;
```

	overtime_id [PK] integer	payroll_id integer	overtime_hours integer	overtime_rate numeric (5,2)	overtime_date date
1	1	1	7	20.97	2024-03-15
2	2	1	7	68.82	2024-10-31
3	3	1	10	43.10	2024-11-06
4	4	1	5	41.22	2024-10-10
5	5	1	3	47.64	2024-06-23
6	6	1	3	50.12	2024-09-30

Total rows: 100 of 100 Query complete 00:00:00.040

6. **Payroll Table** (100 employees with one payroll each)

```
INSERT INTO Payroll (Payroll_ID, Employee_ID, Pay_Date, Gross_Salary, Net_Salary, Deductions, Overtime_Pay, Bonus)
SELECT
    generate_series(1, 100) AS Payroll_ID,
    Employee_ID,
    NOW() - interval '1 day' * generate_series(1, 100) AS Pay_Date,
    round(CAST(random() * 50000 + 50000 AS numeric), 2) AS Gross_Salary,
    round(CAST(random() * 40000 + 30000 AS numeric), 2) AS Net_Salary,
    round(CAST(random() * 5000 + 1000 AS numeric), 2) AS Deductions,
    round(CAST(random() * 500 + 100 AS numeric), 2) AS Overtime_Pay,
    round(CAST(random() * 2000 + 500 AS numeric), 2) AS Bonus
FROM Employee
LIMIT 100;
```

	payroll_id [PK] integer	employee_id integer	pay_date date	gross_salary numeric (10,2)	net_salary numeric (10,2)	deductions numeric (10,2)	overtime_pay numeric (10,2)	bonus numeric (10,2)
1	1	1	2024-10-08	58870.62	67926.15	1763.30	203.93	1624.82
2	2	1	2024-10-07	70775.44	32100.68	1480.86	300.40	1828.71
3	3	1	2024-10-06	59927.40	67835.87	2624.72	448.37	2314.51
4	4	1	2024-10-05	98272.34	61356.60	4359.87	331.56	1059.46
5	5	1	2024-10-04	56873.18	44517.13	1862.90	109.89	569.18
6	6	1	2024-10-03	75798.89	62841.44	3500.73	102.33	1337.72

Total rows: 100 of 100 Query complete 00:00:00.040

7. **Bonus Table** (100 records)

```
INSERT INTO Bonus (Bonus_ID, Payroll_ID, Bonus_Amount, Bonus_Date)
SELECT
    generate_series(1, 100) AS Bonus_ID,
    generate_series(1, 100) AS Payroll_ID,
    floor(random() * 1000 + 500)::decimal(10, 2) AS Bonus_Amount,
    date '2023-12-25' AS Bonus_Date;
```

	bonus_id [PK] integer	payroll_id integer	bonus_amount numeric (10,2)
1	1	1	1589.05
2	2	1	428.69
3	3	1	561.07
4	4	1	359.34
5	5	1	1844.47
6	6	1	1051.95

Total rows: 100 of 100 Query complete 00:00:00.056

8. **Deductions Table** (100 records)

```
INSERT INTO Deductions (Deduction_ID, Payroll_ID, Deduction_Type, Deduction_Amount, Deduction_Date)
SELECT
    generate_series(1, 100) AS Deduction_ID,
    generate_series(1, 100) AS Payroll_ID,
    (ARRAY['Tax', 'Health Insurance', 'Retirement'])[floor(random() * 3 + 1)] AS Deduction_Type,
    floor(random() * 500 + 100)::decimal(10, 2) AS Deduction_Amount,
    date '2023-01-31' AS Deduction_Date;
```

	deduction_id [PK] integer	payroll_id integer	deduction_type text	deduction_amount numeric (10,2)	deduction_date date
1	1	1	Tax	591.23	2024-01-09
2	2	1	Tax	512.58	2024-06-12
3	3	1	Retirement	139.33	2024-05-21
4	4	1	Health Insurance	873.13	2024-10-25
5	5	1	Health Insurance	908.61	2024-12-16
6	6	1	Retirement	129.77	2024-06-08

Total rows: 100 of 100 Query complete 00:00:00.060

9. **Benefits Table** (100 records)

```
INSERT INTO Benefits (Benefit_ID, Employee_ID, Benefit_Type, Benefit_Amount, Benefit_Start_Date, Benefit_End_Date)
SELECT
    generate_series(1, 100) AS Benefit_ID,
    Employee_ID,
    (ARRAY['Health', 'Dental', 'Retirement'])[floor(random() * 3 + 1)] AS Benefit_Type,
    round(CAST(random() * 1000 + 1000 AS numeric), 2) AS Benefit_Amount,
    date '2024-01-01' + (floor(random() * 365)) * interval '1 day' AS Benefit_Start_Date,
    date '2025-01-01' + (floor(random() * 365)) * interval '1 day' AS Benefit_End_Date
FROM Employee
ORDER BY random()
LIMIT 100;
```

	benefit_id [PK] integer	employee_id integer	benefit_type text	benefit_amount numeric (10,2)	benefit_start_date date	benefit_end_date date
1	1	10	Health	1002.00	2024-01-01	2025-01-01
2	2	10	Health	1002.00	2024-01-01	2025-01-01
3	3	10	Health	1002.00	2024-01-01	2025-01-01
4	4	10	Health	1002.00	2024-01-01	2025-01-01
5	5	10	Health	1002.00	2024-01-01	2025-01-01
6	6	10	Health	1002.00	2024-01-01	2025-01-01

Total rows: 100 of 100 Query complete 00:00:00.084

10. **Tax Table** (100 records)

```
INSERT INTO Tax (Tax_ID, Payroll_ID, Tax_Percentage, Tax_Amount)
SELECT
    generate_series(1, 100) AS Tax_ID,
    generate_series(1, 100) AS Payroll_ID,
    floor(random() * 5 + 10)::decimal(5, 2) AS Tax_Percentage,
    floor(random() * 1000 + 500)::decimal(10, 2) AS Tax_Amount;
```

	tax_id [PK] integer	payroll_id integer	tax_percentage numeric (5,2)	tax_amount numeric (10,2)
1	1	1	18.40	663.50
2	2	1	19.52	1386.12
3	3	1	22.23	805.40
4	4	1	23.93	1229.61
5	5	1	19.43	792.31
6	6	1	24.92	1251.74

Total rows: 100 of 100 Query complete 00:00:00.041

11. **Leave Table** (200 records)

```
INSERT INTO Leave (Leave_ID, Employee_ID, Leave_Type, Leave_Start_Date, Leave_End_Date)
SELECT
    generate_series(1, 100) AS Leave_ID,
    Employee_ID,
    (ARRAY['Sick', 'Vacation', 'Personal'])[floor(random() * 3 + 1)] AS Leave_Type,
    date '2024-01-01' + (floor(random() * 30)) * interval '1 day' AS Leave_Start_Date,
    date '2024-02-01' + (floor(random() * 30)) * interval '1 day' AS Leave_End_Date
FROM Employee
ORDER BY random()
LIMIT 100;
```

	leave_id [PK] integer	employee_id integer	leave_type text	leave_start_date date	leave_end_date date
1	1	38	Sick	2024-01-01	2024-02-01
2	2	38	Sick	2024-01-01	2024-02-01
3	3	38	Sick	2024-01-01	2024-02-01
4	4	38	Sick	2024-01-01	2024-02-01
5	5	38	Sick	2024-01-01	2024-02-01
6	6	38	Sick	2024-01-01	2024-02-01

Total rows: 100 of 100 Query complete 00:00:00.048

12. **Full-Time Employee Table** (for full-time employees, assume 40 employees)

```
INSERT INTO FullTimeEmployee (Employee_ID, Annual_Salary)
SELECT Employee_ID,
   round(CAST(random() * 50000 + 50000 AS numeric), 2) AS Annual_Salary
FROM Employee
ORDER BY random()
LIMIT 40;
```

	employee_id [PK] integer	hourly_rate numeric (5,2)	weekly_hours integer
1	92	15.80	10
2	84	16.36	10
3	21	16.55	11
4	65	17.67	11
5	57	17.69	11
6	77	17.83	11

Total rows: 30 of 30 Query complete 00:00:00.141

13. PartTimeEmployee Table:

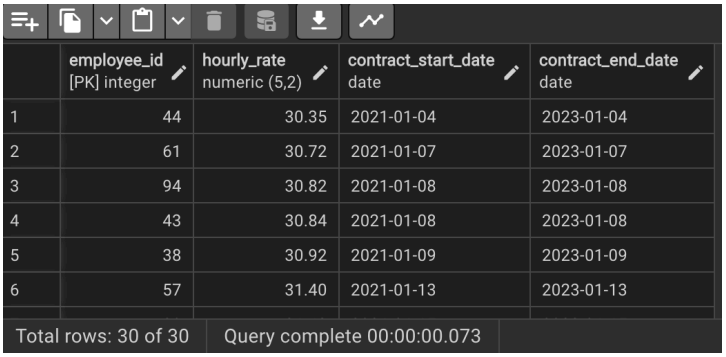
```
INSERT INTO PartTimeEmployee (Employee_ID, Hourly_Rate, Weekly_Hours)
SELECT Employee_ID,
   round(CAST(random() * 30 + 15 AS numeric), 2) AS Hourly_Rate,
   floor(random() * 20 + 10) AS Weekly_Hours
FROM Employee
ORDER BY random()
LIMIT 30;
```

	employee_id [PK] integer	annual_salary numeric (10,2)
1	27	50682.60
2	77	51688.14
3	78	51994.45
4	90	52150.82
5	86	52222.16
6	81	52640.83

Total rows: 40 of 40 Query complete 00:00:00.048

14. Contractor Table:

```
INSERT INTO Contractor (Employee_ID, Hourly_Rate, Contract_Start_Date, Contract_End_Date)
SELECT Employee_ID,
   round(CAST(random() * 40 + 30 AS numeric), 2) AS Hourly_Rate,
   date '2021-01-01' + (floor(random() * 365)) * interval '1 day' AS Contract_Start_Date,
   date '2023-01-01' + (floor(random() * 365)) * interval '1 day' AS Contract_End_Date
FROM Employee
ORDER BY random()
LIMIT 30;
```



	employee_id [PK] integer	hourly_rate numeric (5,2)	contract_start_date date	contract_end_date date
1	44	30.35	2021-01-04	2023-01-04
2	61	30.72	2021-01-07	2023-01-07
3	94	30.82	2021-01-08	2023-01-08
4	43	30.84	2021-01-08	2023-01-08
5	38	30.92	2021-01-09	2023-01-09
6	57	31.40	2021-01-13	2023-01-13

Total rows: 30 of 30 | Query complete 00:00:00.073

Data Population Methodology:

The data was populated using a combination of static and randomized values to ensure realistic but varied dataset entries. Key attributes such as employee details, payroll records, and attendance logs were generated using predefined lists and static data for consistency. This approach enabled the creation of controlled, static sets of information, such as fixed department names and employee types, while ensuring relational integrity across tables through correct foreign key references.