**Staff Payroll System**

**Staff Payroll system:**

Staff Payroll System is a software solution that help to manage the financial records of the staffs working the company. This system will easy handle salary payments, deductions, bonuses, taxes, and other payroll-related transactions of the employees.

**Creating the Database in PostgreSQL:**

**Create the Database:**

-- Create the database for the Payroll Management System

CREATE DATABASE EmpPayrollDB;

**Create the Tables:**

**Employees Table:**

This structure describes the table columns for storing employee information in the **Employee Payroll System**.

**Structure:**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Description** |
| employee\_id | INT(Primary Key) | Unique ID for each employee |
| employee\_name | VARCHAR(255) | Name of the employee |
| department | VARCHAR(100) | Department where the employee works |
| position | VARCHAR(100) | Job title or position of the employee |
| hire\_date | DATE | Date when the employee was hired |
| base\_salary | DECIMAL(10, 2) | Base salary of the employee |

**Code:**

CREATE TABLE Employees (

employee\_id SERIAL PRIMARY KEY,

employee\_name VARCHAR(255) NOT NULL,

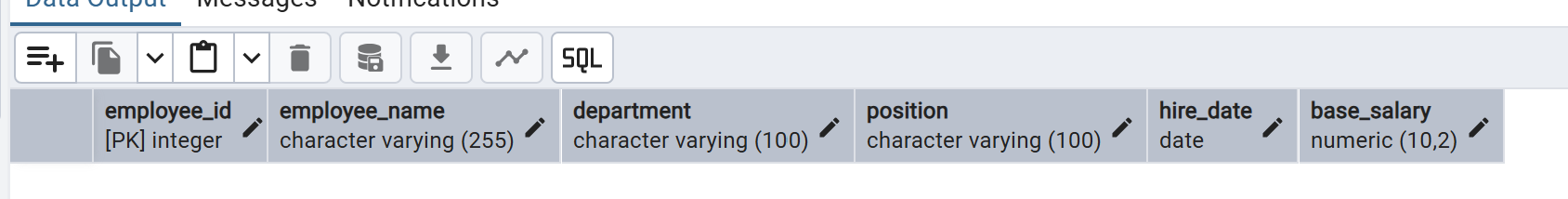
department VARCHAR(100),

position VARCHAR(100),

hire\_date DATE,

base\_salary DECIMAL(10, 2) NOT NULL

);



**Attendance Table:**

This structure describes the columns for tracking employee attendance in the **Employee Payroll System**.

**Structure:**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Description** |
| attendance\_id | INT(Primary Key) | Unique ID for each attendance record, auto-incremented |
| employee\_id | INT | ID of the employee (references Employees table) |
| attendance\_date | DATE | Date of the attendance record |
| status | CHECK(status IN ('Present', 'Absent', 'Leave')) | Attendance status for the day (Present, Absent, Leave) |

**Code:**

CREATE TABLE Attendance (

attendance\_id SERIAL PRIMARY KEY,

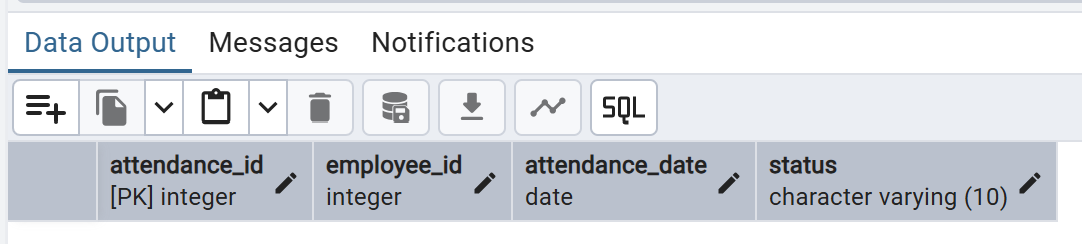
employee\_id INT,

attendance\_date DATE,

status VARCHAR(10) CHECK (status IN ('Present', 'Absent', 'Leave')),

FOREIGN KEY (employee\_id) REFERENCES Employees(employee\_id)

);



**Salaries Table:**

This table stores employee salary details, including base salary, bonuses, deductions, and the specific month and year for the salary record.

**Structure:**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Description** |
| salary\_id | INT(Primary Key) | Unique ID for each salary record |
| employee\_id | INT | ID of the employee (references Employees table) |
| base\_salary | DECIMAL(10, 2) | The base salary for the employee |
| bonus | DECIMAL(10, 2) | Additional bonuses for the employee (optional) |
| deductions | DECIMAL(10, 2) | Deductions from the salary (optional) |
| month | VARCHAR(20) | The month for which the salary is being calculated |
| year | INT | The year for which the salary is being calculated |

**Code:**

CREATE TABLE Salaries (

salary\_id SERIAl PRIMARY KEY,

employee\_id INT,

base\_salary DECIMAL(10, 2) NOT NULL,

bonus DECIMAL(10, 2),

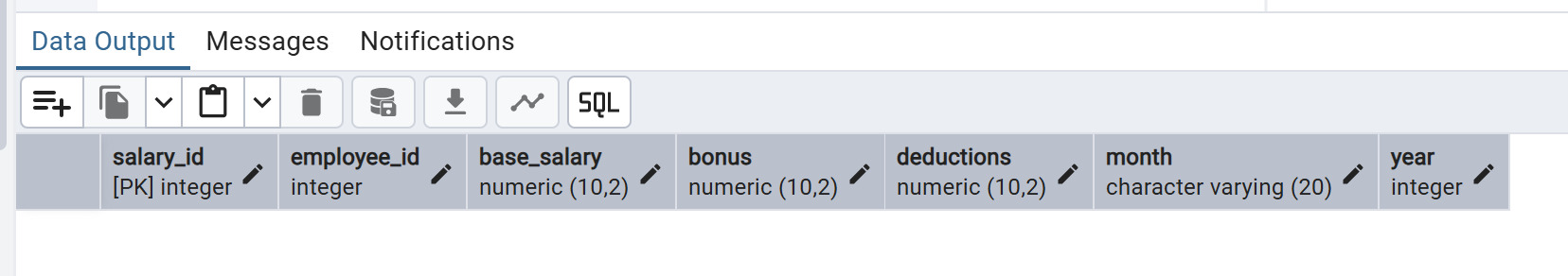
deductions DECIMAL(10, 2),

month VARCHAR(20),

year INT,

FOREIGN KEY (employee\_id) REFERENCES Employees(employee\_id)

);



**Payroll Table:**

This table stores each employee's payroll record, including the total salary paid and the payment date.

**Structure:**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Description** |
| payroll\_id | INT(Primary Key) | Unique ID for each payroll record |
| employee\_id | INT | ID of the employee (references Employees table) |
| total\_salary | DECIMAL(10, 2) | Total salary paid to the employee (after deductions and bonuses) |
| payment\_date | DATE | The date when the salary was paid |

**Code:**

CREATE TABLE Payroll (

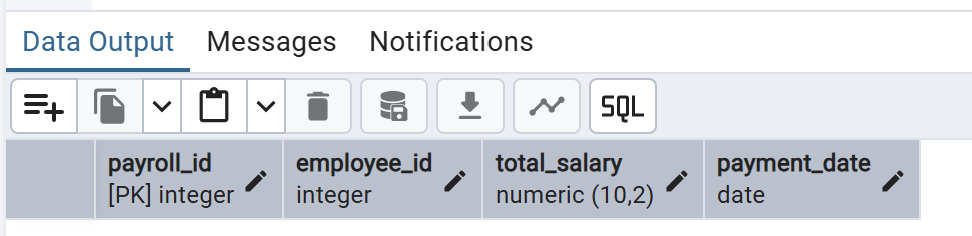
payroll\_id SERIAL PRIMARY KEY,

employee\_id INT,

total\_salary DECIMAL(10, 2),

payment\_date DATE,

FOREIGN KEY (employee\_id) REFERENCES Employees(employee\_id)

);  


**Inserting Data:**

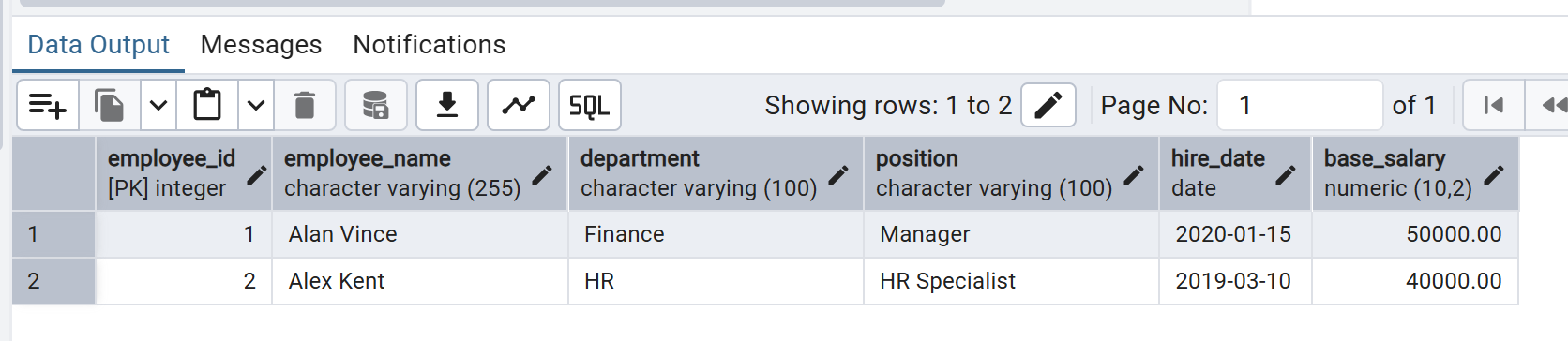
Add some sample data to the tables.

**Inserting Data into Employees Table:**INSERT INTO Employees (employee\_id,employee\_name, department, position, hire\_date, base\_salary)

VALUES

(1, 'Alan Vince', 'Finance', 'Manager', '2020-01-15', 50000.00),

(2, 'Alex Kent', 'HR', 'HR Specialist', '2019-03-10', 40000.00);

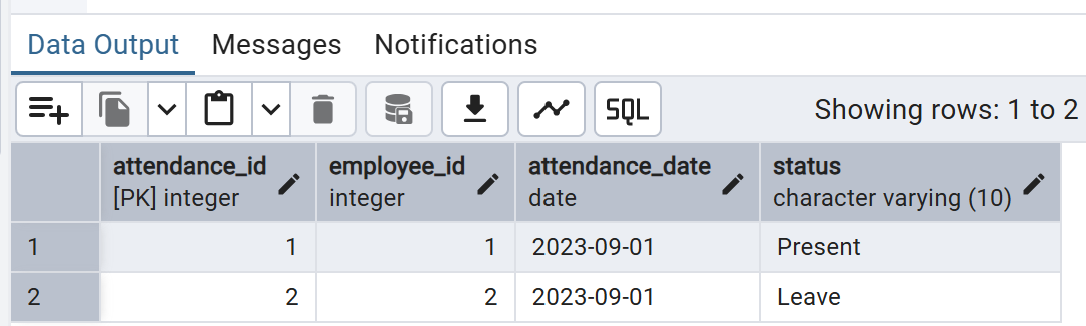


**Inserting Data into Attendance Table:**

INSERT INTO Attendance (employee\_id, attendance\_date, status) VALUES

(1, '2023-09-01', 'Present'),

(2, '2023-09-01', 'Leave');

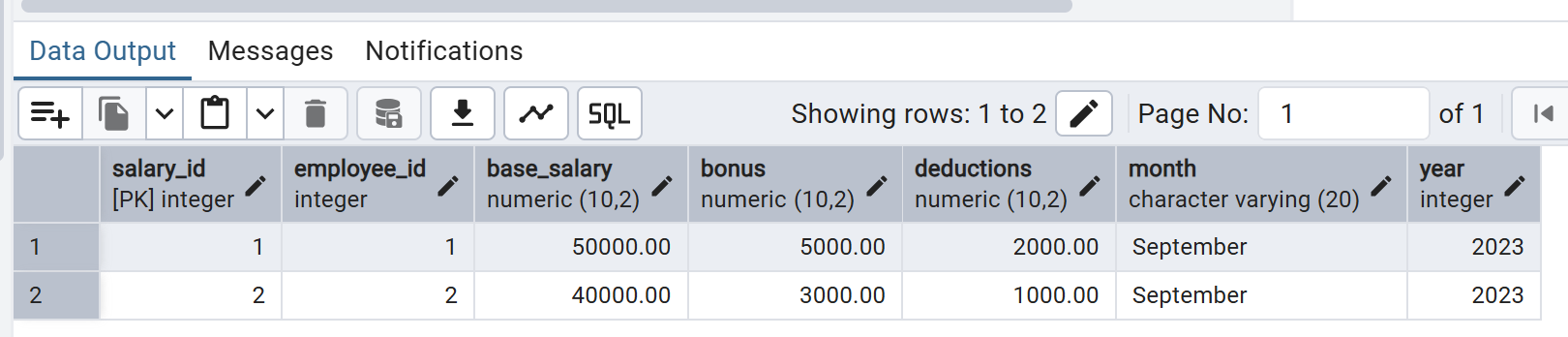


**Inserting Data into Salaries Table:**

INSERT INTO Salaries (employee\_id, base\_salary, bonus, deductions, month, year) VALUES

(1, 50000.00, 5000.00, 2000.00, 'September', 2023),

(2, 40000.00, 3000.00, 1000.00, 'September', 2023);



**Basic Functionalities of Employee Payroll System:**

* **Add New Employees:**
  + Allow users to enter new employee records into the system (name, department, position, salary, etc.).
* **Update Employee Information:**
  + Provide functionality to update existing employee details (e.g., position, salary, department).
* **Delete Employee Records:**
  + Enable users to remove employee records when no longer needed or relevant.
* **Track Employee Attendance:**
  + Record daily attendance for employees, marking them as present, absent, or on leave.
* **Calculate Salary:**.
  + Automatically calculate employee salaries based on base salary, attendance, bonuses, and deductions.
* **Manage Deductions and Bonuses:**
  + Allow users to add or update bonuses and deductions for each employee.
* **Update Payroll Records:**
  + Calculate and store payroll data for each employee, including total salary and payment date.
* **Generate Pay Slips:**
  + Provide functionality to generate detailed pay slips for employees, showing salary breakdowns.
* **Update Stock Quantities:**
  + Allow manual updates to stock levels after receiving new inventory or making sales.
* **Generate Payroll Reports:**
  + Create payroll summaries and reports, such as overall salary expenses, employee attendance, or monthly payroll distributions.

**Writing Queries for Functionality:**

**Query-1: Add New Employee**

INSERT INTO Employees (employee\_id, employee\_name, department, position, hire\_date, base\_salary)

VALUES (3,'Jennith Kery', 'Sales', 'Sales Executive', '2023-10-01', 30000.00);

**Explanation:**

This SQL statement inserts a new employee record into the Employees table. The employee\_name field is set to "Jennith Kery," indicating the employee's name. The employee is assigned to the "Sales" department with the position of "Sales Executive." The hire\_date is set to "2023-10-01," marking the employee's starting date, and the base\_salary is defined as 30,000.00. Each value is mapped to the corresponding column in the Employees table, and this operation will create a new row in the table with this information.

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

**Query-2: Update Employee Information**

UPDATE Employees

SET base\_salary = 55000.00

WHERE employee\_id = 1;

**Explanation:**

This SQL query updates the base\_salary of a specific employee in the Employees table. It changes the base salary to 55,000.00 for the employee whose employee\_id is 1. The UPDATE statement modifies existing records, and in this case, only the record that matches employee\_id = 1 will have its base\_salary updated to the new value. This ensures that only the intended employee's salary is modified while the rest of the information remains unchanged.

**Output :**

A screenshot of a computer

AI-generated content may be incorrect.

**Query-3: Delete Employee Records**

DELETE FROM Employees

WHERE employee\_id = 3;

**Explanation:**

This SQL query deletes a specific employee's record from the Employees table. The DELETE statement removes the record where the employee\_id is 3. This operation permanently deletes the entire row associated with this employee, including all their details such as name, department, position, salary, and hire date. It is important to use a WHERE clause to ensure only the intended employee record is deleted, preventing unintentional data loss.

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

**Query-4: Track Attendance**

INSERT INTO Attendance (employee\_id, attendance\_date, status)

VALUES (1, '2023-09-02', 'Present');

**Explanation:**

This query inserts a new record into the Attendance table, marking the attendance for an employee. The employee with employee\_id = 1 is recorded as being "Present" on the date 2023-09-02. The INSERT INTO statement is used to add this information into the table, specifying the relevant columns: employee\_id, attendance\_date, and status. This is useful for tracking employees' daily attendance.

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

**Query-5: Salary Calculation**

SELECT employee\_id, (base\_salary + bonus - deductions) AS total\_salary

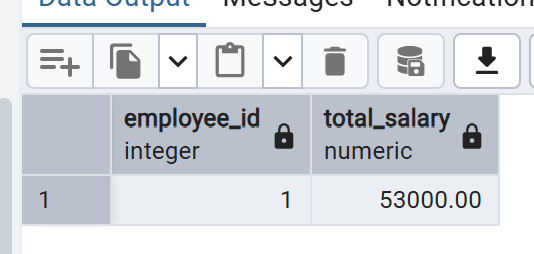
FROM Salaries

WHERE employee\_id = 1 AND month = 'September' AND year = 2023;

**Explanation:**

This query calculates the total salary for the employee with employee\_id = 1 for the month of September in the year 2023. The SELECT statement retrieves the employee's employee\_id along with the calculated total salary. The total salary is derived from adding the base\_salary and bonus and then subtracting any deductions. The query filters records using the WHERE clause to ensure it fetches the salary details only for September 2023. This query is useful for generating salary information for a specific employee within a given time frame.

**Output:**



**Query-6: Update Payroll Table**

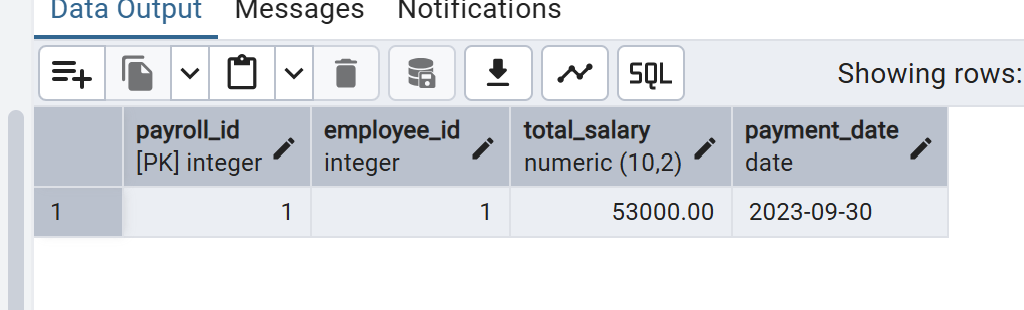
INSERT INTO Payroll (employee\_id, total\_salary, payment\_date) VALUES

(1, 53000.00, '2023-09-30');

**Explanation:**

This SQL INSERT statement is used to add a new record into the Payroll table for an employee with employee\_id = 1. The inserted data includes the employee’s total salary of 53,000.00 and the payment date, which is September 30, 2023. This operation stores the payroll information, ensuring the employee's payment details, such as the amount and payment date, are tracked in the Payroll table. It is particularly useful after calculating an employee's salary and finalizing the payment process.

**Output:**



**Query-7: Generate Pay Slips**

SELECT e.employee\_name, s.base\_salary, s.bonus, s.deductions, p.total\_salary, p.payment\_date

FROM Employees e

JOIN Salaries s ON e.employee\_id = s.employee\_id

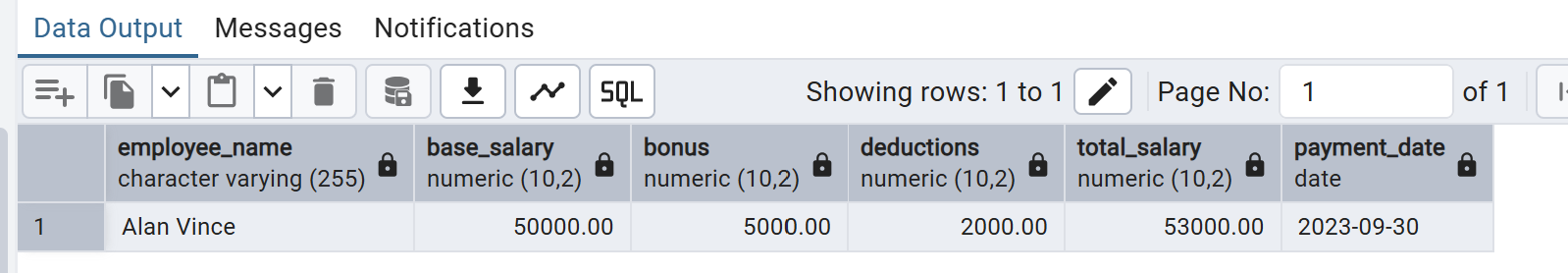
JOIN Payroll p ON e.employee\_id = p.employee\_id

WHERE e.employee\_id = 1 AND s.month = 'September' AND s.year = 2023;

**Explanation:**

This SQL query retrieves a detailed pay slip for an employee with employee\_id = 1 for the month of September 2023. It combines data from three tables: Employees, Salaries, and Payroll. The query selects the employee's name, base salary, bonus, deductions, total salary, and payment date. The JOIN clauses ensure that information from the related tables is pulled together using employee\_id as the common field. The WHERE clause filters the results to show only the details of this specific employee and for the month of September in the year 2023.

**Output:**



**Query-8: Generate Reports**

SELECT e.employee\_name, p.total\_salary, p.payment\_date

FROM Payroll p

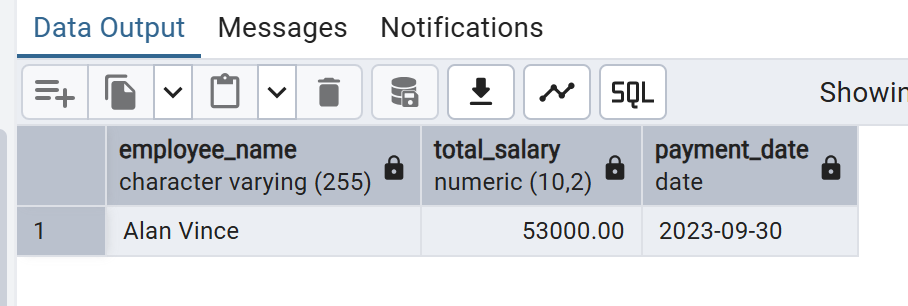
JOIN Employees e ON p.employee\_id = e.employee\_id

WHERE p.payment\_date BETWEEN '2023-09-01' AND '2023-09-30';

**Explanation:**

This SQL query retrieves a detailed pay slip for an employee with employee\_id = 1 for the month of September 2023. It combines data from three tables: Employees, Salaries, and Payroll. The query selects the employee's name, base salary, bonus, deductions, total salary, and payment date. The JOIN clauses ensure that information from the related tables is pulled together using employee\_id as the common field. The WHERE clause filters the results to show only the details of this specific employee and for the month of September in the year 2023.

**Output:**



**Query-9: List Employees by Department**

SELECT department, employee\_name

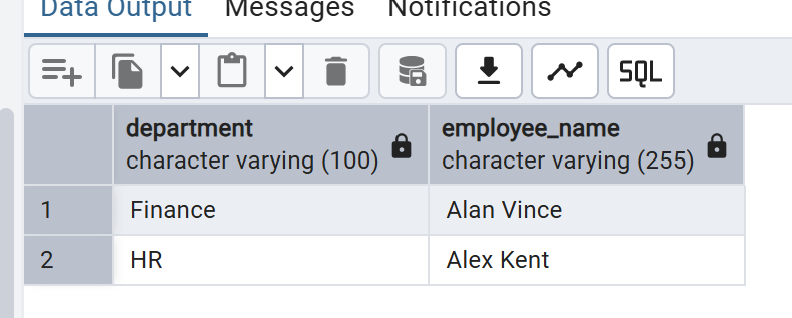
FROM Employees

ORDER BY department, employee\_name;

**Explanation:**

This query retrieves a list of employees, grouped by their department, from the Employees table. The SELECT statement pulls the department and employee\_name columns. The ORDER BY clause ensures that the results are sorted first by the department and then by the employee's name within each department, organizing the employees in alphabetical order under their respective departments. This allows for easy readability and understanding of how employees are distributed across various departments.

**Output:**



**Query-10: View Attendance Summary for an Employee**

SELECT employee\_id,

SUM(CASE WHEN status = 'Present' THEN 1 ELSE 0 END) AS present\_days,

SUM(CASE WHEN status = 'Absent' THEN 1 ELSE 0 END) AS absent\_days,

SUM(CASE WHEN status = 'Leave' THEN 1 ELSE 0 END) AS leave\_days

FROM Attendance

WHERE attendance\_date BETWEEN '2023-09-01' AND '2023-09-30'

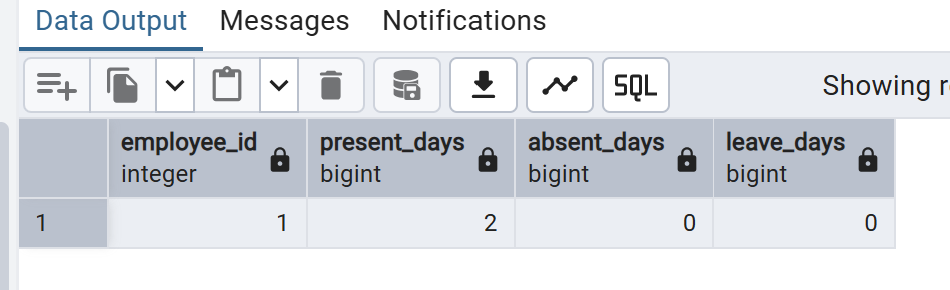
AND employee\_id = 1

GROUP BY employee\_id;

**Explanation:**

This query generates a summary of the number of days an employee (with employee\_id = 1) was present, absent, or on leave during the month of September 2023. The CASE statements count each occurrence of 'Present', 'Absent', or 'Leave' by adding 1 for each match. The SUM function aggregates these counts to provide the total number of days in each status. The WHERE clause restricts the results to the specific date range ('2023-09-01' to '2023-09-30') and the specified employee. The results are grouped by employee\_id to ensure one summarized row per employee.

**Output:**



**Query-11: List Employees with Salaries Above a Certain Threshold**

SELECT employee\_name, base\_salary

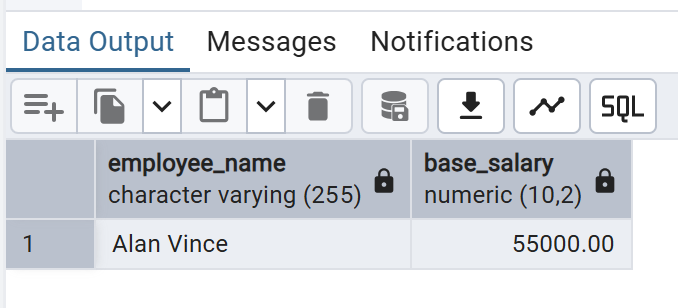
FROM Employees

WHERE base\_salary > 50000;

**Explanation:**

This query retrieves the names and base salaries of all employees from the Employees table whose base salary exceeds 50,000. By specifying the condition in the WHERE clause, it ensures that only employees meeting this salary criterion are included in the results. This can be useful for identifying higher-earning employees, which might assist in salary reviews, budget planning, or performance evaluations.

**Output:**



**Query-12: View Total Deductions for All Employees in a Given Month**

SELECT employee\_id, SUM(deductions) AS total\_deductions

FROM Salaries

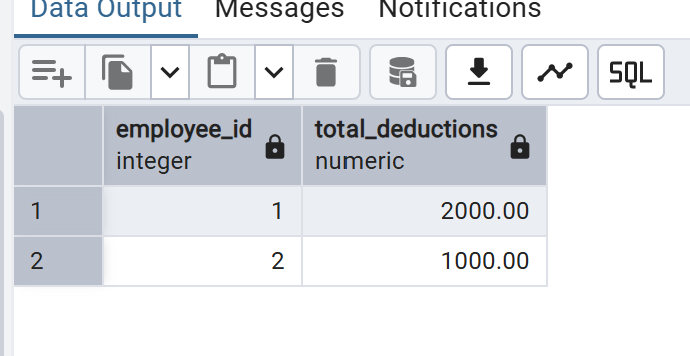
WHERE month = 'September' AND year = 2023

GROUP BY employee\_id;

**Explanation:**

This query calculates the total deductions for each employee for the month of September 2023. It does so by summing up the deductions column for each employee\_id in the Salaries table, based on the specified month and year. The GROUP BY clause ensures that the sum is calculated individually for each employee. This query helps provide an overview of the total deductions applied to employees' salaries within a specific time period.

**Output:**



**Query-13: Calculate Average Salary for a Department**

SELECT department, AVG(base\_salary) AS average\_salary

FROM Employees

WHERE department = 'Sales'

GROUP BY department;

**Explanation:**

This query calculates the average base salary for employees within the 'Sales' department. The AVG() function is used to compute the average salary from the base\_salary column. The GROUP BY clause ensures that the result is grouped by the department, so the average is calculated specifically for the 'Sales' department. This query helps to determine the overall average salary level for a particular department, in this case, 'Sales'.

**Output:**

Output not generated for insufficient data

**Query-14: List Employees with Attendance Status on a Specific Date**

SELECT e.employee\_name, a.status

FROM Employees e

JOIN Attendance a ON e.employee\_id = a.employee\_id

WHERE a.attendance\_date = '2023-09-15';

**Explanation:**

This query retrieves the attendance status for all employees on a specific date, which in this case is '2023-09-15'. The JOIN operation links the Employees and Attendance tables based on the employee\_id. The query returns each employee's name and their attendance status (such as 'Present', 'Absent', or 'Leave') on that particular date. The WHERE clause ensures that only records for '2023-09-15' are shown. This query is useful for tracking employee attendance on a given day.

**Output:**

Output not generated for insufficient data

**Query-15: List Employees with Bonuses in a Given Month**

SELECT e.employee\_name, s.bonus

FROM Employees e

JOIN Salaries s ON e.employee\_id = s.employee\_id

WHERE s.bonus > 0

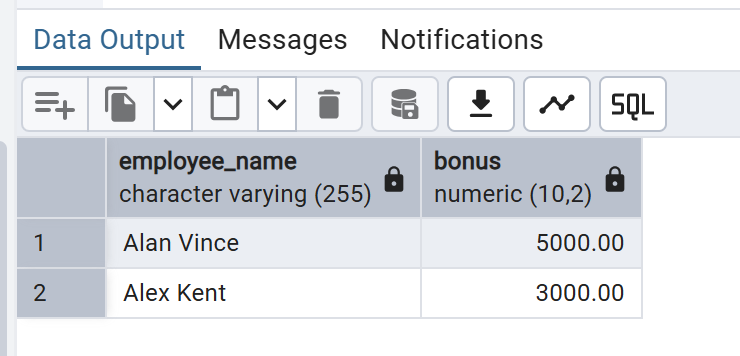
AND s.month = 'September'

AND s.year = 2023;

**Explanation:**

This query retrieves the names of employees who received a bonus during the month of September 2023. It uses a JOIN operation to combine the Employees and Salaries tables based on the employee\_id field. The WHERE clause filters the results to include only those records where the bonus is greater than zero, ensuring that only employees with a bonus are listed. Additionally, it restricts the data to bonuses received in September 2023 by filtering the month and year fields in the Salaries table.

**Output:**



**Query-16: List Employees with Deducted Salaries in a Given Month**

SELECT e.employee\_name, s.deductions

FROM Employees e

JOIN Salaries s ON e.employee\_id = s.employee\_id

WHERE s.deductions > 0

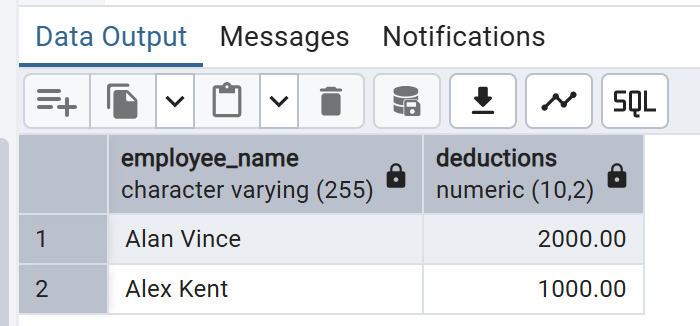
AND s.month = 'September'

AND s.year = 2023;

**Explanation:**

This query identifies employees whose salary was deducted during the month of September 2023. It uses a JOIN to connect the Employees and Salaries tables based on the employee\_id, allowing the query to match employee names with their corresponding deductions. The WHERE clause ensures that only employees with deductions greater than zero are included, while also filtering the results to those related to the month of September in the year 2023. The result will list the employee name along with the deduction amount for that month.

**Output:**



**Query-17: Generate Yearly Salary Report**

SELECT e.employee\_name, SUM(s.base\_salary + s.bonus - s.deductions) AS yearly\_salary

FROM Employees e

JOIN Salaries s ON e.employee\_id = s.employee\_id

WHERE s.year = 2023

GROUP BY e.employee\_name;

**Explanation:**

This query calculates the total yearly salary for each employee by summing the base salary, bonus, and deductions from the Salaries table. It selects the employee's name and calculates the yearly salary using SUM(). The JOIN clause links the Employees and Salaries tables based on the employee's ID, ensuring that salary details for each employee are retrieved. The WHERE clause restricts the query to data from the year 2023, and the results are grouped by employee name using GROUP BY, summarizing the salary data for each individual.

**Output:**

