

Task 2

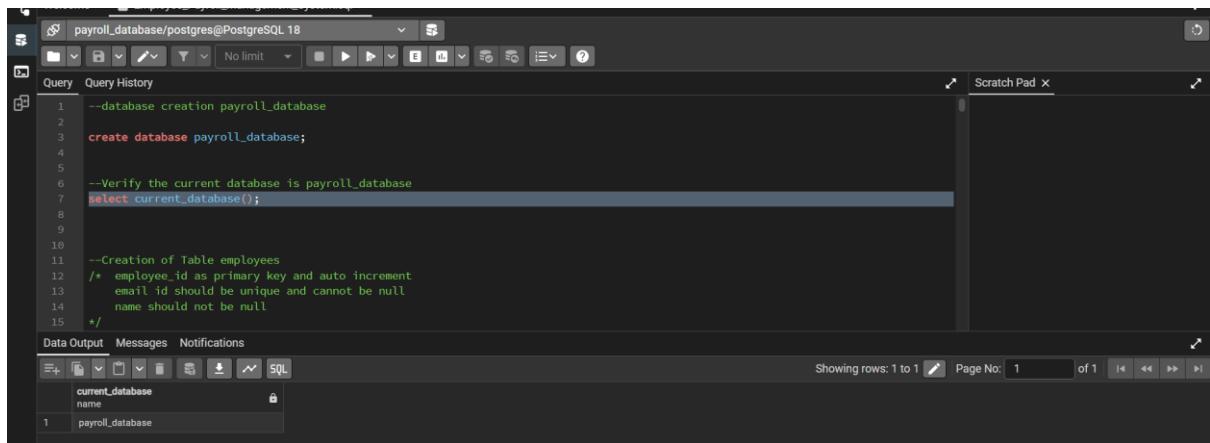
Project: Employee Payroll Management System (PostgreSQL)

Objective:

Design and implement an employee payroll system to store, manage, and analyse salary data.

GITHUB: https://github.com/xrahulcrx/Employee_Payroll_Management_System

Database created and selected;



The screenshot shows the pgAdmin 4 interface. In the Query tab, the following SQL code is run:

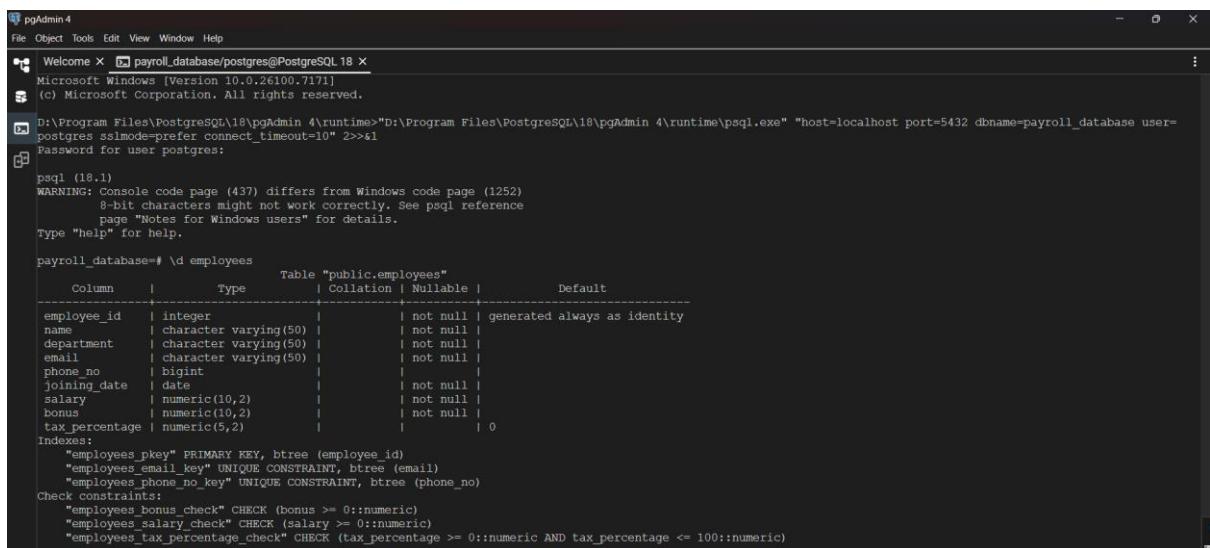
```
--database creation payroll_database
create database payroll_database;

--Verify the current database is payroll_database
select current_database();
```

The result of the query is displayed in the Data Output tab:

current_database	name
1	payroll_database

Table created:



The screenshot shows the pgAdmin 4 interface. The terminal window displays the creation of a table named 'employees' in the 'public' schema:

```
psql (18.1)
WARNING: Console code page (437) differs from Windows code page (1252)
8-bit characters might not work correctly. See pgsql reference
page "Notes for Windows users" for details.
Type "help" for help.

payroll_database=# \d employees
Table "public.employees"
 Column | Type          | Collation | Nullable | Default
-----+---------------+-----------+----------+-
employee_id | integer       |           | not null | generated always as identity
name | character varying(50) |           | not null |
department | character varying(50) |           | not null |
email | character varying(50) |           | not null |
phone_no | bigint        |           | not null |
joining_date | date          |           | not null |
salary | numeric(10,2) |           | not null |
bonus | numeric(10,2) |           | not null |
tax_percentage | numeric(5,2) |           |           | 0
Indexes:
"employees_pkey" PRIMARY KEY, btree (employee_id)
"employees_email_key" UNIQUE CONSTRAINT, btree (email)
"employees_phone_no_key" UNIQUE CONSTRAINT, btree (phone_no)
Check constraints:
"employees_bonus_check" CHECK (bonus >= 0::numeric)
"employees_salary_check" CHECK (salary >= 0::numeric)
"employees_tax_percentage_check" CHECK (tax_percentage >= 0::numeric AND tax_percentage <= 100::numeric)
```

Data Entry:

Insert 10 sample employee records.

```

31
32 -- Sample Data to be entered
33
34 insert into employees (Name, Department, Email, Phone_no, Joining_date, Salary, Bonus, Tax_percentage) values
35 ('John Miller', 'Sales', 'john.miller@example.com', 9876543210, '2025-01-10', 85000.00, 15000.00, 12.5),
36 ('Sarah Johnson', 'IT', 'sarah.johnson@example.com', 9234567890, '2024-11-05', 95000.00, 16000.00, 15.0),
37 ('Michael Davis', 'HR', 'michael.davis@example.com', 9123456789, '2024-09-12', 70000.00, 5000.00, 10.0),
38 ('Emily Clark', 'Finance', 'emily.clark@example.com', 988776555, '2024-08-22', 120000.00, 20000.00, 18.0),
39 ('David Wilson', 'Sales', 'david.wilson@example.com', 9345678901, '2025-02-18', 88000.00, 12000.00, 12.5),
40 ('Sophia Martinez', 'IT', 'sophia.martinez@example.com', 9456781230, '2024-12-01', 105000.00, 15000.00, 15.0),
41 ('Daniel Brown', 'Finance', 'daniel.brown@example.com', 9678901234, '2024-10-30', 115000.00, 18000.00, 18.0),
42 ('Olivia Taylor', 'HR', 'olivia.taylor@example.com', 9789012345, '2025-03-05', 68000.00, 3000.00, 10.0),
43 ('James Anderson', 'Sales', 'james.anderson@example.com', 9567896123, '2024-07-14', 98000.00, 10000.00, 12.5),
44 ('Ava Thomas', 'IT', 'ava.thomas@example.com', 9890123456, '2024-06-20', 95000.00, 9000.00, 15.0),
45 ('Amit Sharma', 'IT', 'amit.sharma@company.com', 9675543210, '2022-03-15', 95000.00, 15000.00, 12.5),
46 ('Pooja Nair', 'Finance', 'nooria.nair@example.com', 9678941234, '2025-10-30', 115000.00, 18000.00, 18.0).

```

Payroll Queries:

- a) Retrieve the list of employees sorted by salary in descending order.

```

62
63 -- a) Retrieve the list of employees sorted by salary in descending order.
64
65 select employee_id, name, department, salary
66 from employees
67 order by salary desc;
68

```

employee_id	[PK] integer	name	character varying (50)	department	character varying (50)	salary	numeric (10,2)
1	4	Emily Clark		Finance		120000.00	
2	7	Daniel Brown		Finance		115000.00	
3	12	Pooja Nair		Finance		115000.00	
4	6	Sophia Martinez		IT		105000.00	
5	2	Sarah Johnson		IT		95000.00	
6	10	Ava Thomas		IT		95000.00	
7	11	Amit Sharma		IT		95000.00	
8	9	James Anderson		Sales		90000.00	
9	14	Rohan Gupta		Sales		88000.00	
10	5	David Wilson		Sales		88000.00	
11	1	John Miller		Sales		85000.00	
12	13	Priya Singh		HR		70000.00	
13	3	Michael Davis		HR		70000.00	
14	8	Olivia Taylor		HR		68000.00	

- b) Find employees with a total compensation (SALARY + BONUS) greater than \$100,000.

```

70
71 -- b) Find employees with a total compensation (SALARY + BONUS) greater than $100,000.
72
73 select employee_id, name, department, (salary + bonus) as Total_Salary
74 from employees
75 where (salary + bonus) > 100000
76 order by Total_Salary desc;
77

```

employee_id	[PK] integer	name	character varying (50)	department	character varying (50)	total_salary	numeric
1	4	Emily Clark		Finance		140000.00	
2	7	Daniel Brown		Finance		133000.00	
3	12	Pooja Nair		Finance		133000.00	
4	6	Sophia Martinez		IT		120000.00	
5	11	Amit Sharma		IT		110000.00	
6	2	Sarah Johnson		IT		105000.00	
7	10	Ava Thomas		IT		104000.00	
8	1	John Miller		Sales		101500.00	
9	14	Rohan Gupta		Sales		101200.00	
10	5	David Wilson		Sales		101200.00	
11	9	James Anderson		Sales		101000.00	

c) Update the bonus for employees in the ‘Sales’ department by 10%.

```

77
78 -- c) Update the bonus for employees in the 'Sales' department by 10%.
79
80 update employees set bonus = bonus * 1.10
81 where department = 'Sales';
82
83 select employee_id, name, department, bonus as New_bonus
84 from employees where department = 'Sales';
85

```

The screenshot shows the pgAdmin interface with a query editor containing the provided SQL code. Below the editor is a results grid displaying four rows of employee data from the 'employees' table, specifically for the 'Sales' department. The columns shown are employee_id, name, department, and new_bonus.

employee_id	name	department	new_bonus
1	John Miller	Sales	16500.00
2	David Wilson	Sales	13200.00
3	James Anderson	Sales	11000.00
4	Rohan Gupta	Sales	13200.00

d) Calculate the net salary after deducting tax for all employees.

```

87
88 -- d) Calculate the net salary after deducting tax for all employees.
89
90 select employee_id, name, department, salary, bonus, tax_percentage,
91       (salary + bonus) as Gross_Salary,
92       round((salary * (tax_percentage / 100)),2) as Tax_deductions,
93       round((salary + bonus) * (1 - tax_percentage / 100) , 2) as Net_Salary
94 from employees
95 order by Net_Salary desc, name;

```

The screenshot shows the pgAdmin interface with a query editor containing the provided SQL code. Below the editor is a results grid displaying 14 rows of employee data, ordered by Net_Salary in descending order. The columns shown are employee_id, name, department, salary, bonus, tax_percentage, gross_salary, tax_deductions, and net_salary.

employee_id	name	department	salary	bonus	tax_percentage	gross_salary	tax_deductions	net_salary
1	Emily Clark	Finance	120000.00	20000.00	18.00	140000.00	21600.00	118800.00
2	Daniel Brown	Finance	115000.00	18000.00	18.00	133000.00	20700.00	109300.00
3	Pooja Nair	Finance	115000.00	18000.00	18.00	133000.00	20700.00	109300.00
4	Sophia Martinez	IT	105000.00	15000.00	15.00	120000.00	15750.00	102250.00
5	Amit Sharma	IT	95000.00	15000.00	12.50	110000.00	11875.00	96250.00
6	Sarah Johnson	IT	95000.00	10000.00	15.00	105000.00	14250.00	89250.00
7	John Miller	Sales	85000.00	16500.00	12.50	101500.00	10625.00	88875.00
8	David Wilson	Sales	88000.00	13200.00	12.50	101200.00	11000.00	88550.00
9	Rohan Gupta	Sales	88000.00	13200.00	12.50	101200.00	11000.00	88550.00
10	Ava Thomas	IT	95000.00	9000.00	15.00	104000.00	14250.00	88400.00
11	James Anderson	Sales	90000.00	11000.00	12.50	101000.00	11250.00	88375.00
12	Michael Davis	HR	70000.00	5000.00	10.00	75000.00	7000.00	67500.00
13	Priya Singh	HR	70000.00	5000.00	10.00	75000.00	7000.00	67500.00
14	Olivia Taylor	HR	68000.00	3000.00	10.00	71000.00	6800.00	63900.00

e) Retrieve the average, minimum, and maximum salary per department.

```

96
97 -- e) Retrieve the average, minimum, and maximum salary per department.
98
99 select department, count(*) as Employee_Count, round(avg(salary), 2) as Average_Salary,
100       min(salary) as Minimum_Salary, max(salary) as Maximum_Salary
101   from employees
102  group by department
103  order by Average_Salary desc, department;

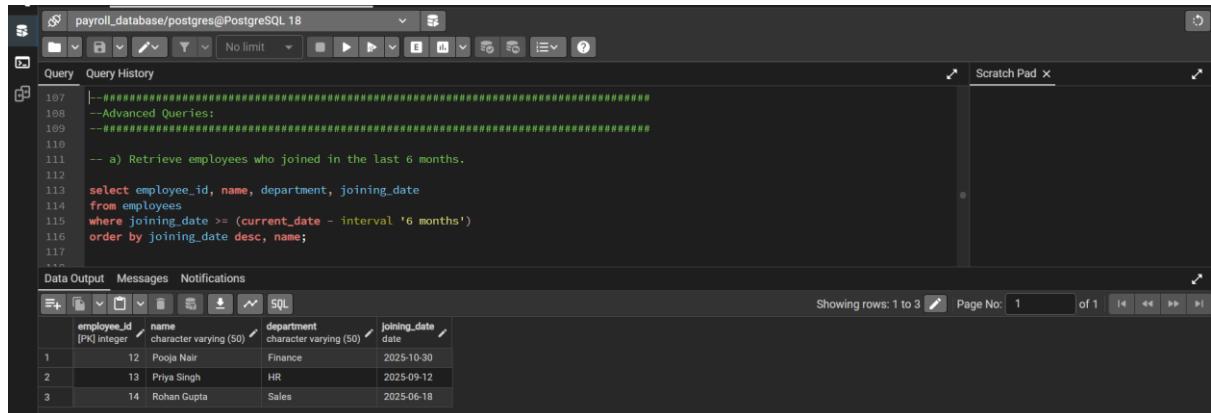
```

The screenshot shows the pgAdmin interface with a query editor containing the provided SQL code. Below the editor is a results grid displaying 4 rows of departmental summary data, ordered by Average_Salary in descending order. The columns shown are department, employee_count, average_salary, minimum_salary, and maximum_salary.

department	employee_count	average_salary	minimum_salary	maximum_salary
Finance	3	116666.67	115000.00	120000.00
IT	4	97500.00	95000.00	105000.00
Sales	4	87750.00	85000.00	90000.00
HR	3	69333.33	68000.00	70000.00

Advanced Queries:

- a) Retrieve employees who joined in the last 6 months.

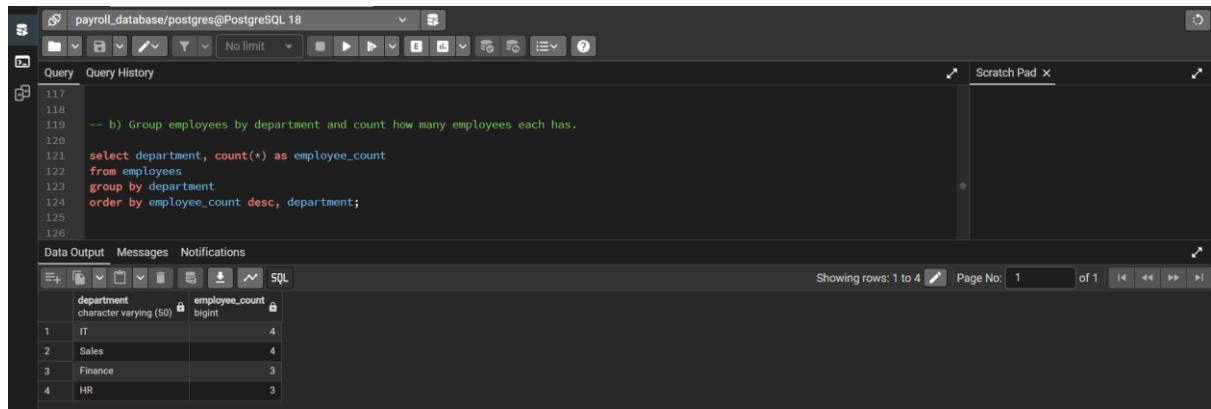


The screenshot shows a DBeaver interface with a SQL editor and a results grid. The SQL code retrieves employees from the 'employees' table where the joining date is within the last six months. The results show three employees: Pooja Nair, Priya Singh, and Rohan Gupta, each with their employee ID, name, department, and joining date.

```
-- a) Retrieve employees who joined in the last 6 months.
select employee_id, name, department, joining_date
from employees
where joining_date >= (current_date - interval '6 months')
order by joining_date desc, name;
```

employee_id	name	department	joining_date
12	Pooja Nair	Finance	2025-10-30
13	Priya Singh	HR	2025-09-12
14	Rohan Gupta	Sales	2025-06-18

- b) Group employees by department and count how many employees each has.

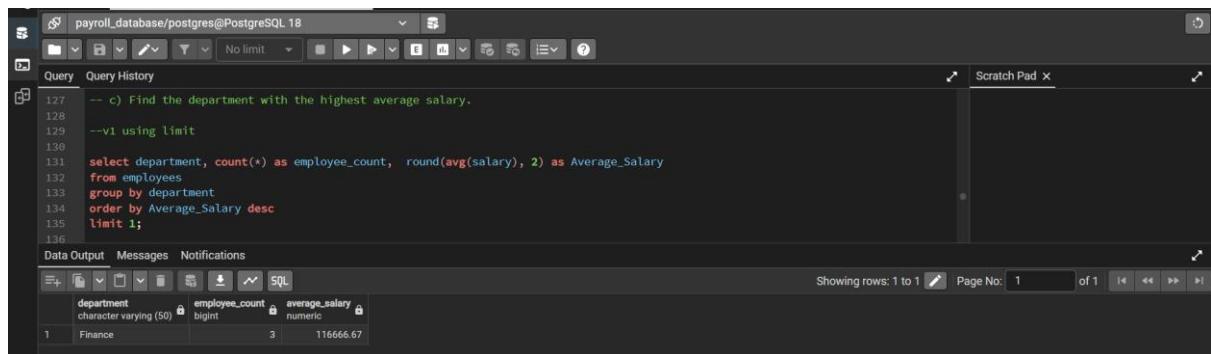


The screenshot shows a DBeaver interface with a SQL editor and a results grid. The SQL code groups employees by department and counts the number of employees in each department. The results show four departments: IT, Sales, Finance, and HR, with counts of 4, 4, 3, and 3 respectively.

```
-- b) Group employees by department and count how many employees each has.
select department, count(*) as employee_count
from employees
group by department
order by employee_count desc, department;
```

department	employee_count
IT	4
Sales	4
Finance	3
HR	3

- c) Find the department with the highest average salary.



The screenshot shows a DBeaver interface with a SQL editor and a results grid. The SQL code groups employees by department, calculates the average salary for each department, and then finds the department with the highest average salary using a LIMIT clause. The result shows the Finance department with an average salary of 116666.67.

```
-- c) Find the department with the highest average salary.
-- v1 using limit
select department, count(*) as employee_count, round(avg(salary), 2) as Average_Salary
from employees
group by department
order by Average_Salary desc
limit 1;
```

department	employee_count	average_salary
Finance	3	116666.67

```

137
138 --v2 using window function
139 select department, employee_count, Average_Salary
140 from (
141     select department, count(*) as employee_count, round(avg(salary), 2) as Average_Salary,
142             rank() over (order by avg(salary) desc) as rnk
143         from employees
144     group by department
145 )ranked
146 where rnk = 1;

```

Query History

department	employee_count	average_salary
Finance	3	116666.67

Data Output Messages Notifications

d) Identify employees who have the same salary as at least one other employee.

```

151 --v1
152 select employee_id, name, department, salary
153 from employees
154 where salary in (
155     select salary from employees
156     group by salary
157     having count(*) > 1
158 )
159 order by salary desc, name;

```

Query History

employee_id	name	department	salary
1	Daniel Brown	Finance	115000.00
2	Pooja Nair	Finance	115000.00
3	Amit Sharma	IT	95000.00
4	Ava Thomas	IT	95000.00
5	Sarah Johnson	IT	95000.00
6	David Wilson	Sales	88000.00
7	Rohan Gupta	Sales	88000.00
8	Michael Davis	HR	70000.00
9	Priya Singh	HR	70000.00

Data Output Messages Notifications

```

162 --v2 using window function
163 select employee_id, name, department, salary
164 from (
165     select employee_id, name, department, salary,
166             count(*) over (partition by salary) as sal_match
167         from employees
168 ) sal
169 where sal_match > 1
170 order by salary desc, name;

```

Query History

employee_id	name	department	salary
1	Daniel Brown	Finance	115000.00
2	Pooja Nair	Finance	115000.00
3	Amit Sharma	IT	95000.00
4	Ava Thomas	IT	95000.00
5	Sarah Johnson	IT	95000.00
6	David Wilson	Sales	88000.00
7	Rohan Gupta	Sales	88000.00
8	Michael Davis	HR	70000.00
9	Priya Singh	HR	70000.00

Data Output Messages Notifications