SQL PROJECT DOCUMANTAION

INTRODUCTION

I have started my project to analyze the data provided by HUBBLEMIND. Throughout this process, I have studied various problem statements outlined by the institute. I imported the data into SQL Server and utilized PostgreSQL to perform analyses and tasks. This project has enhanced my understanding of data manipulation and analysis using SQL, enabling me to draw meaningful insights from the data.

QUARIES

Importing table

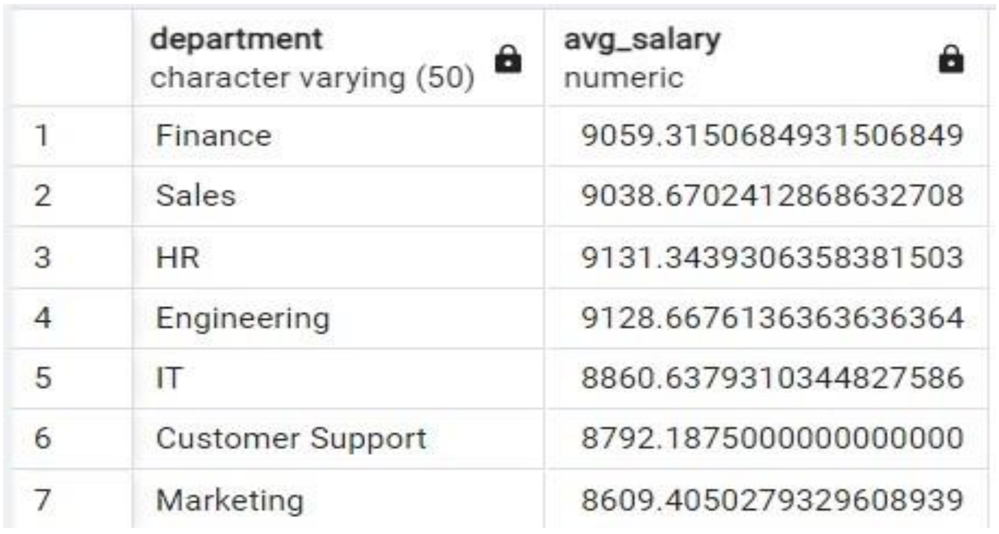
```
create table EPES
(EmployeeID varchar PRIMARY KEY,
 Name varchar(50) not null,
 Department varchar(50) not null,
 JobTitle varchar(50) not null,
 MonthlySalary numeric(30),
 PerformanceScore int,
 YearsAtCompany int,
 HireDate date,
 LastPerformanceReviewDate date,
 Email VARCHAR(255) UNIQUE NOT NULL
);
```

Using this query I have import my table in sql server where the employeeid is my primary key

Week 1: Beginner Level

1: Calculate the average monthly salary for each department.

```
select Department, avg( MonthlySalary) as vg_Salary from EPES
group by Department;
```



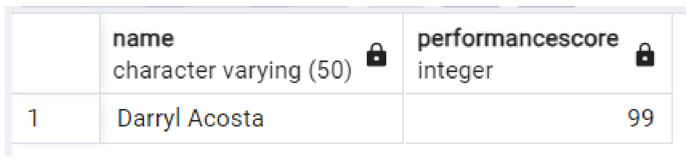
	department character varying (50)	avg_salary numeric
1	Finance	9059.3150684931506849
2	Sales	9038.6702412868632708
3	HR	9131.3439306358381503
4	Engineering	9128.6676136363636364
5	IT	8860.6379310344827586
6	Customer Support	8792.1875000000000000
7	Marketing	8609.4050279329608939

Explanation:

1.SELECT Department, AVG(MonthlySalary): Retrieves the department and calculates the average monthly salary
2.Avg_Salary: Renames the result of the average calculation as `Avg_Salary`.
3.FROM EPES: Specifies the `EPES` table as the data source.
4.GROUP BY Department: Groups the data by department, so the average salary is calculated for each group. (`AVG(MonthlySalary)`).
5.Avg_Salary: Renames the result of the average calculation as `Avg_Salary`.

2: Identify the employee with the highest performance score in the IT department.

```
select name, PerformanceScore from EPES
where Department = 'IT'
order by PerformanceScore desc
limit 1 ;
```



	name character varying (50)	performancescore integer
1	Darryl Acosta	99

Explanation:

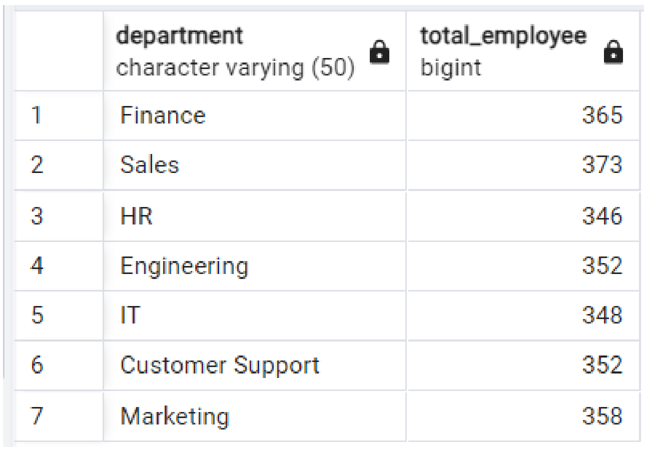
1. **SELECT Name, PerformanceScore**: Retrieves the employee's name and their performance score.
2. **FROM EPES**: Specifies the EPES table as the source.
3. **WHERE Department = 'IT'**: Filters the data to only include employees from the IT department.
4. **ORDER BY PerformanceScore DESC**: Sorts the employees by performance score in descending order (highest first).
5. **LIMIT 1**: Restricts the result to only the top entry, i.e., the employee with the highest performance score.

3: Count the number of employees in each department.

```
select Department, count(EmployeeID) as total_employee from EPES
group by Department;
```

Explanation:

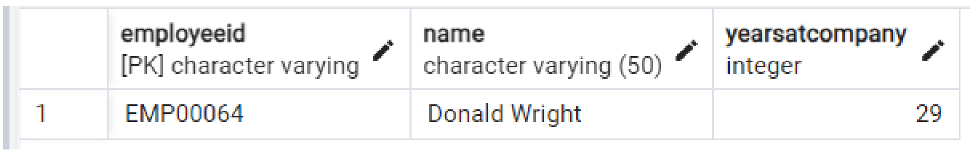
1. **SELECT Department, COUNT(EmployeeID)**: Retrieves the department and counts the number of employees (COUNT(EmployeeID)).
2. **AS total_employee**: Labels the count result as total_employee for readability.
3. **FROM EPES**: Specifies the EPES table as the data source.
4. **GROUP BY Department**: Groups the data by department, so the count is done for each department.



	department character varying (50)	total_employee bigint
1	Finance	365
2	Sales	373
3	HR	346
4	Engineering	352
5	IT	348
6	Customer Support	352
7	Marketing	358

4: Find the employee who has been with the company the longest.

```
select EmployeeID, name, YearsAtCompany  from EPES
order by YearsAtCompany  desc
limit 1;
```



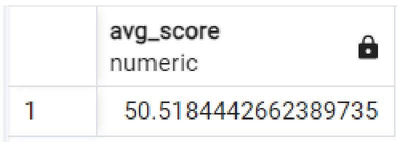
	employeeid [PK] character varying	name character varying (50)	yearsatcompany integer
1	EMP00064	Donald Wright	29

Explanation:

1. **SELECT EmployeeID, Name, YearsAtCompany**: Retrieves the employee's ID, name, and the number of years they have been with the company.
2. **FROM EPES**: Specifies the EPES table as the data source.
3. **ORDER BY YearsAtCompany DESC**: Sorts the employees by YearsAtCompany in descending order, showing the longest-serving employee first.
4. **LIMIT 1**: Limits the result to only the top entry, which is the employee with the longest tenure.

5: Determine the average performance score across all employees.

```
select Avg( PerformanceScore) as Avg_score
from EPES;
```



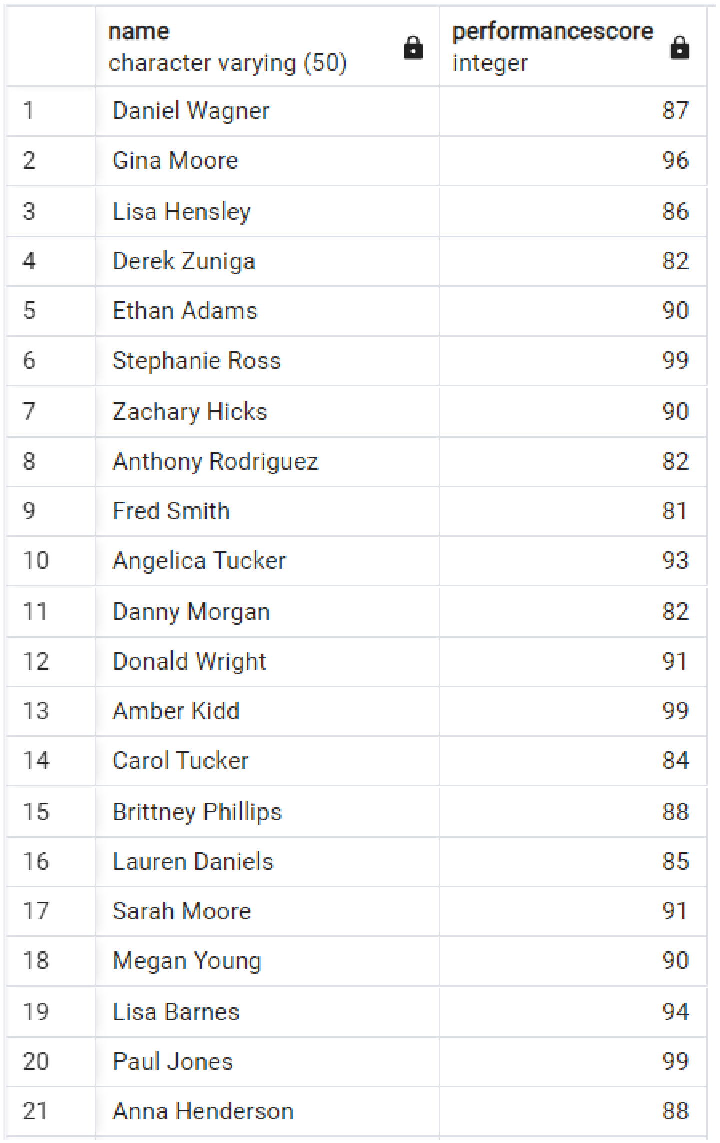
	avg_score numeric
1	50.5184442662389735

Explanation:

1. **SELECT AVG(PerformanceScore)**: Calculates the average of all employees' performance scores using the AVG() function.
2. **AS Avg_score**: Labels the result as Avg_score for clarity.
3. **FROM EPES**: Specifies the EPES table as the data source.

6: List all employees who have a performance score above 80.

```
select name, PerformanceScore from EPES
where PerformanceScore> 80 ;
```



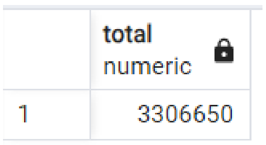
	name character varying (50)	performancescore integer
1	Daniel Wagner	87
2	Gina Moore	96
3	Lisa Hensley	86
4	Derek Zuniga	82
5	Ethan Adams	90
6	Stephanie Ross	99
7	Zachary Hicks	90
8	Anthony Rodriguez	82
9	Fred Smith	81
10	Angelica Tucker	93
11	Danny Morgan	82
12	Donald Wright	91
13	Amber Kidd	99
14	Carol Tucker	84
15	Brittney Phillips	88
16	Lauren Daniels	85
17	Sarah Moore	91
18	Megan Young	90
19	Lisa Barnes	94
20	Paul Jones	99
21	Anna Henderson	88

Explanation:

1. **SELECT Name, PerformanceScore**: Retrieves the employee's name and performance score.
2. **FROM EPES**: Specifies the EPES table as the data source.
3. **WHERE PerformanceScore > 80**: Filters the data to only include employees with a performance score greater than 80.

7: Calculate the total monthly salary expenditure for the Finance department.

```
select sum(monthlysalary) as total
from EPES
where department = 'finance';
```



```
   total
   numeric
1  3306650
```

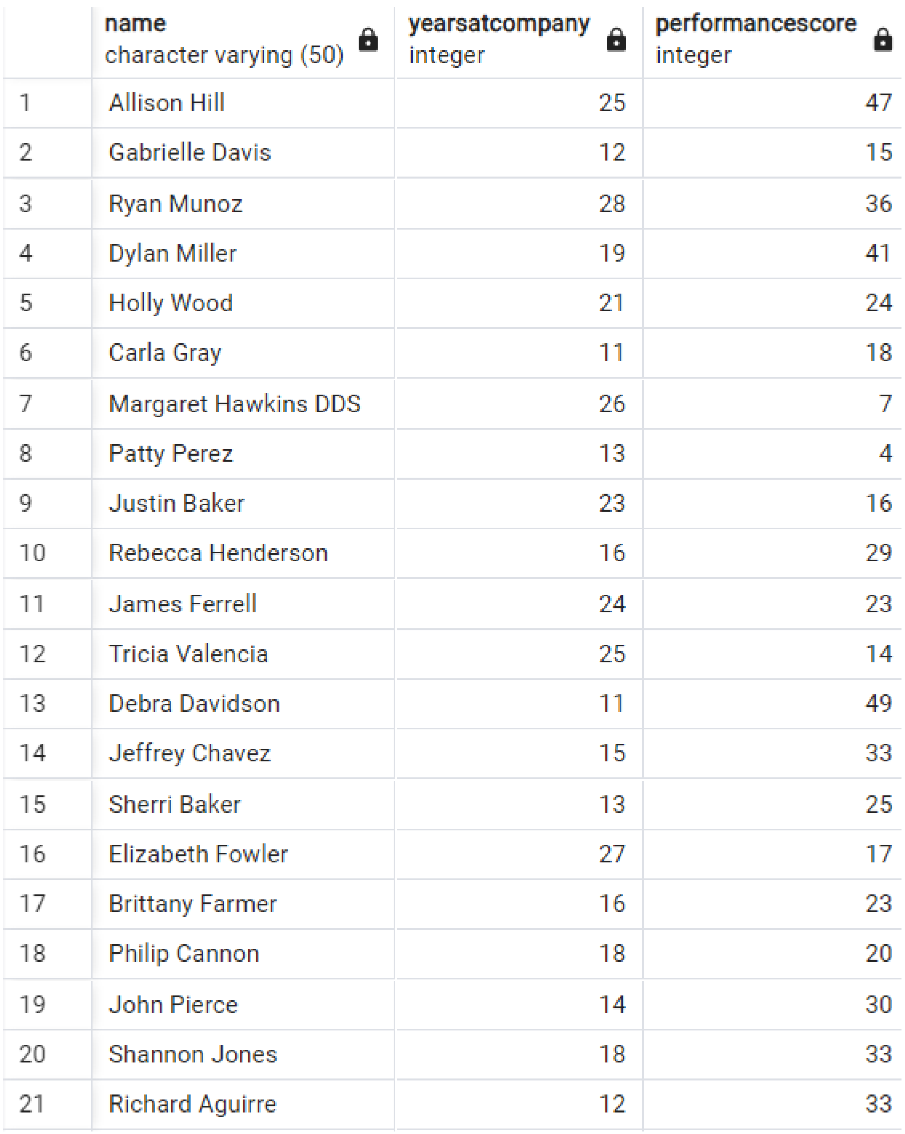
Explanation:

1. **SELECT SUM(MonthlySalary)**: Calculates the total sum of the monthly salary for employees in the Finance department.
2. **AS total**: Labels the result as total for clarity.
3. **FROM EPES**: Specifies the EPES table as the data source.
4. **WHERE Department = 'Finance'**: Filters the data to include only employees in the Finance department.

Week 2: Intermediate Level

1: Create a list of employees who have been with the company for more than 10 years but have a performance score below 50.

```
select name,yearsatcompany, performancescore from EPES
where yearsatcompany> 10
and performancescore <50;
```



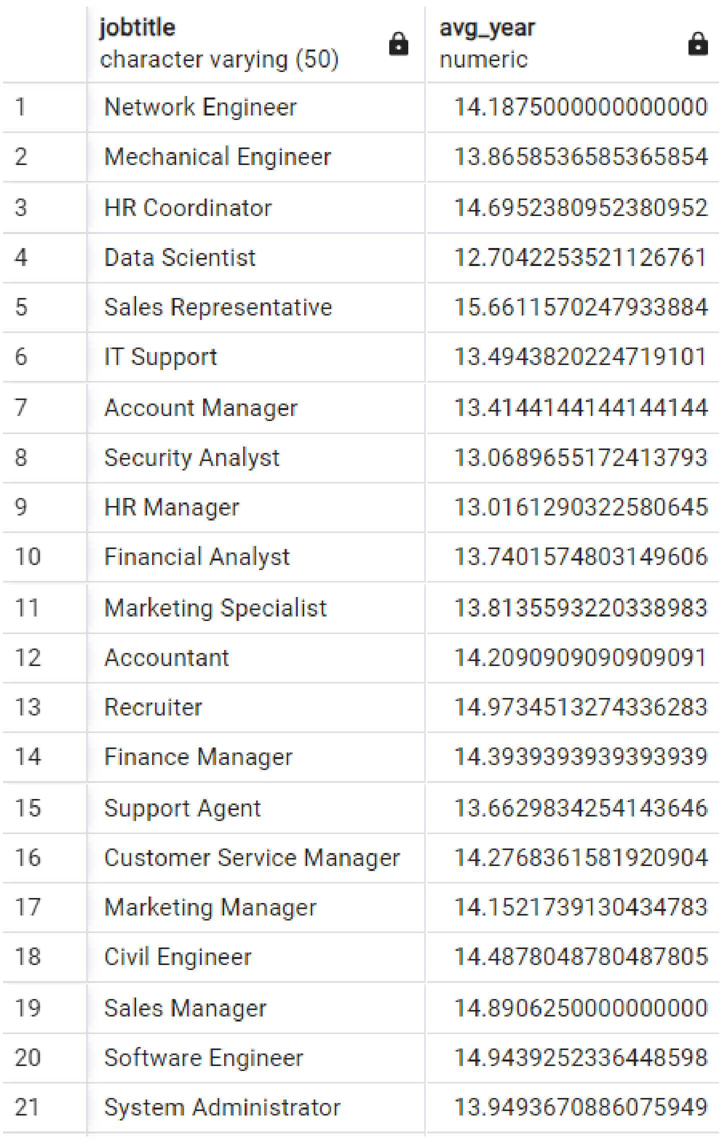
	name character varying (50)	yearsatcompany integer	performancescore integer
1	Allison Hill	25	47
2	Gabrielle Davis	12	15
3	Ryan Munoz	28	36
4	Dylan Miller	19	41
5	Holly Wood	21	24
6	Carla Gray	11	18
7	Margaret Hawkins DDS	26	7
8	Patty Perez	13	4
9	Justin Baker	23	16
10	Rebecca Henderson	16	29
11	James Ferrell	24	23
12	Tricia Valencia	25	14
13	Debra Davidson	11	49
14	Jeffrey Chavez	15	33
15	Sherri Baker	13	25
16	Elizabeth Fowler	27	17
17	Brittany Farmer	16	23
18	Philip Cannon	18	20
19	John Pierce	14	30
20	Shannon Jones	18	33
21	Richard Aguirre	12	33

Explanation:

1. **SELECT Name, YearsAtCompany, PerformanceScore**: Retrieves the employee's name, years at the company, and performance score.
2. **FROM EPES**: Specifies the EPES table as the data source.
3. **WHERE YearsAtCompany > 10 AND PerformanceScore < 50**: Filters the data to include employees who have been with the company for more than 10 years and have a performance score below 50.

2: Determine the average number of years at the company for each job title.

```
select jobtitle, avg(yearsatcompany) as avg_year
from EPES
group by jobtitle;
```



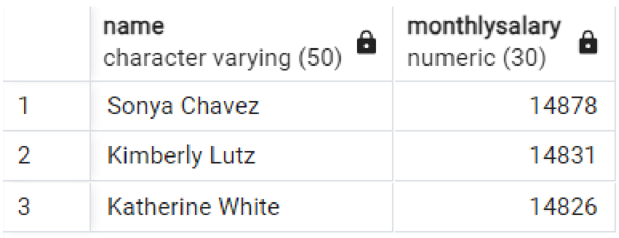
	jobtitle character varying (50)	avg_year numeric
1	Network Engineer	14.1875000000000000
2	Mechanical Engineer	13.8658536585365854
3	HR Coordinator	14.6952380952380952
4	Data Scientist	12.7042253521126761
5	Sales Representative	15.6611570247933884
6	IT Support	13.4943820224719101
7	Account Manager	13.4144144144144144
8	Security Analyst	13.0689655172413793
9	HR Manager	13.0161290322580645
10	Financial Analyst	13.7401574803149606
11	Marketing Specialist	13.8135593220338983
12	Accountant	14.2090909090909091
13	Recruiter	14.9734513274336283
14	Finance Manager	14.3939393939393939
15	Support Agent	13.6629834254143646
16	Customer Service Manager	14.2768361581920904
17	Marketing Manager	14.1521739130434783
18	Civil Engineer	14.4878048780487805
19	Sales Manager	14.8906250000000000
20	Software Engineer	14.9439252336448598
21	System Administrator	13.9493670886075949

Explanation:

1. **SELECT JobTitle, AVG(YearsAtCompany)**: Retrieves each job title and calculates the average number of years employees have been with the company.
2. **AS avg_year**: Labels the result of the average calculation as avg_year.
3. **FROM EPES**: Specifies the EPES table as the data source.
4. **GROUP BY JobTitle**: Groups the data by job title, so the average years at the company is calculated for each group.

3: Find the top 3 highest-paid employees in the HR department.

```
select name, monthlysalary from EPES
where department = 'HR'
order by monthlysalary desc
limit 3;
```



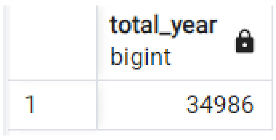
	name character varying (50)	monthlysalary numeric (30)
1	Sonya Chavez	14878
2	Kimberly Lutz	14831
3	Katherine White	14826

Explanation:

1. **SELECT Name, MonthlySalary**: Retrieves the employee's name and monthly salary.
2. **FROM EPES**: Specifies the `EPES` table as the data source.
3. **WHERE Department = 'HR'**: Filters the data to include only employees in the HR department.
4. **ORDER BY MonthlySalary DESC**: Sorts the employees by salary in descending order, showing the highest-paid employees first.
5. **LIMIT 3**: Restricts the result to the top 3 highest-paid employees.

4: Calculate the total number of years all employees have worked at the company combined.

```
select sum(yearsatcompany) as total_year
from EPES;
```



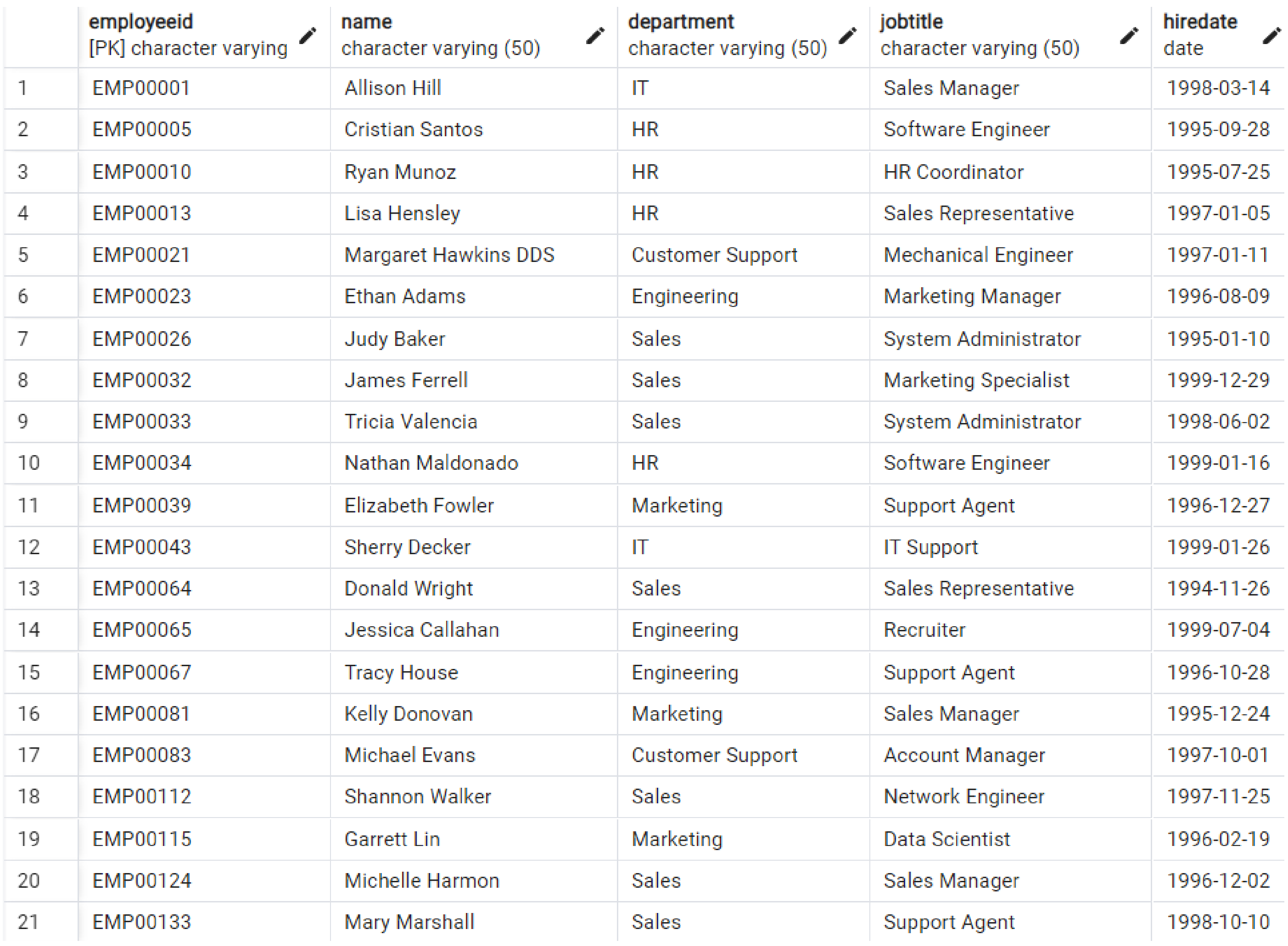
	total_year bigint
1	34986

Explanation:

1. **SELECT SUM(YearsAtCompany)**: Calculates the total sum of the years all employees have worked at the company.
2. **AS total_year**: Labels the result as total_year for clarity.
3. **FROM EPES**: Specifies the EPES table as the data source.

5: Generate a list of employees who were hired before the year 2000.

```
select employeeid, name, department, jobtitle, hiredate
from EPES
where hiredate< '2000-01-01' ;
```



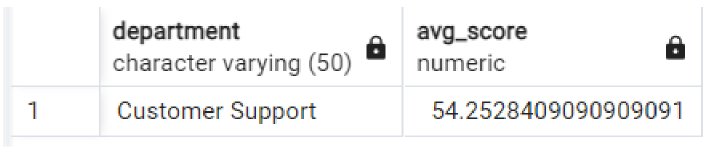
	employeeid [PK] character varying	name character varying (50)	department character varying (50)	jobtitle character varying (50)	hiredate date
1	EMP00001	Allison Hill	IT	Sales Manager	1998-03-14
2	EMP00005	Cristian Santos	HR	Software Engineer	1995-09-28
3	EMP00010	Ryan Munoz	HR	HR Coordinator	1995-07-25
4	EMP00013	Lisa Hensley	HR	Sales Representative	1997-01-05
5	EMP00021	Margaret Hawkins DDS	Customer Support	Mechanical Engineer	1997-01-11
6	EMP00023	Ethan Adams	Engineering	Marketing Manager	1996-08-09
7	EMP00026	Judy Baker	Sales	System Administrator	1995-01-10
8	EMP00032	James Ferrell	Sales	Marketing Specialist	1999-12-29
9	EMP00033	Tricia Valencia	Sales	System Administrator	1998-06-02
10	EMP00034	Nathan Maldonado	HR	Software Engineer	1999-01-16
11	EMP00039	Elizabeth Fowler	Marketing	Support Agent	1996-12-27
12	EMP00043	Sherry Decker	IT	IT Support	1999-01-26
13	EMP00064	Donald Wright	Sales	Sales Representative	1994-11-26
14	EMP00065	Jessica Callahan	Engineering	Recruiter	1999-07-04
15	EMP00067	Tracy House	Engineering	Support Agent	1996-10-28
16	EMP00081	Kelly Donovan	Marketing	Sales Manager	1995-12-24
17	EMP00083	Michael Evans	Customer Support	Account Manager	1997-10-01
18	EMP00112	Shannon Walker	Sales	Network Engineer	1997-11-25
19	EMP00115	Garrett Lin	Marketing	Data Scientist	1996-02-19
20	EMP00124	Michelle Harmon	Sales	Sales Manager	1996-12-02
21	EMP00133	Mary Marshall	Sales	Support Agent	1998-10-10

Explanation:

1. **SELECT EmployeeID, Name, Department, JobTitle, HireDate**: Retrieves the employee's ID, name, department, job title, and hire date.
2. **FROM EPES**: Specifies the EPES table as the data source.
3. **WHERE HireDate < '2000-01-01'**: Filters the data to include only employees who were hired before January 1, 2000.

6: Find the department with the highest average performance score.

```
select department, avg(performancescore) as avg_score
from EPES
group by department
order by avg_score desc
limit 1;
```



	department character varying (50)	avg_score numeric
1	Customer Support	54.2528409090909091

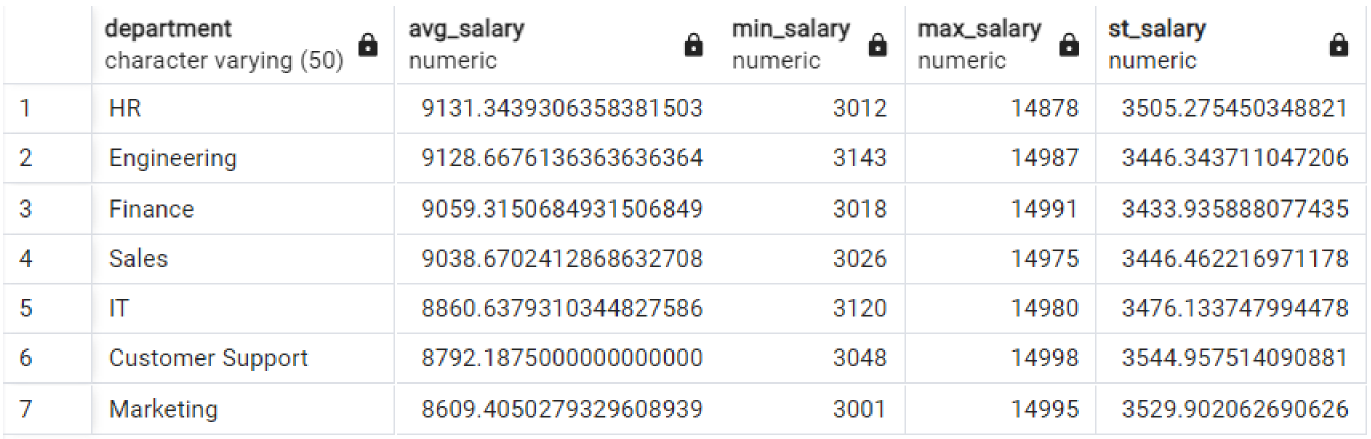
Explanation:

1. **SELECT Department, AVG(PerformanceScore)**: Retrieves the department and calculates the average performance score for that department.

2. **AS avg_score**: Labels the result of the average calculation as avg_score.
3. **FROM EPES**: Specifies the EPES table as the data source.
4. **GROUP BY Department**: Groups the data by department, so the average score is calculated for each department.
5. **ORDER BY avg_score DESC**: Sorts the departments by their average performance score in descending order, showing the highest average score first.
6. **LIMIT 1**: Restricts the result to the top entry, which is the department with the highest average performance score

7: Analyse the distribution of monthly salaries across different departments and identify any significant disparities.

```
select department,
avg(monthlySalary) as avg_salary, min(monthlySalary) as min_salary,
max(monthlySalary) as max_salary,
stddev(monthlySalary) as st_salary
from EPES
group by department
order by avg_salary desc  ;
```



	department character varying (50)	avg_salary numeric	min_salary numeric	max_salary numeric	st_salary numeric
1	HR	9131.3439306358381503	3012	14878	3505.275450348821
2	Engineering	9128.6676136363636364	3143	14987	3446.343711047206
3	Finance	9059.3150684931506849	3018	14991	3433.935888077435
4	Sales	9038.6702412868632708	3026	14975	3446.462216971178
5	IT	8860.6379310344827586	3120	14980	3476.133747994478
6	Customer Support	8792.1875000000000000	3048	14998	3544.957514090881
7	Marketing	8609.4050279329608939	3001	14995	3529.902062690626

Explanation:

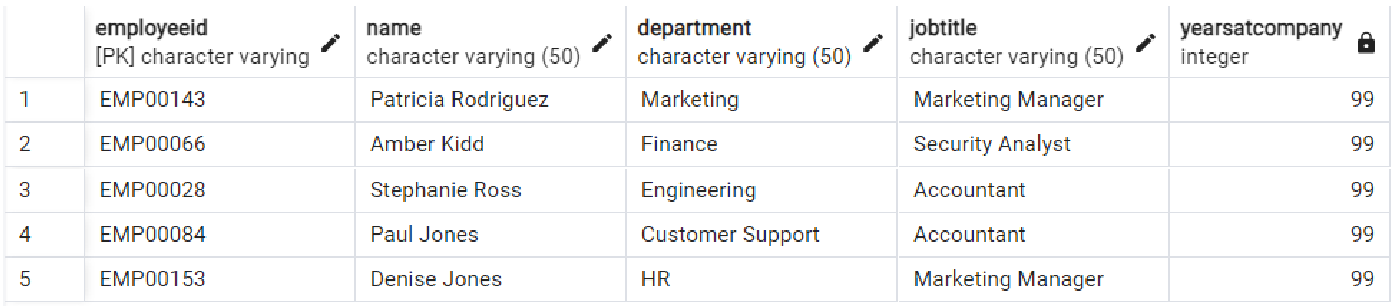
1. **SELECT Department**: Retrieves the department names.
2. **AVG(MonthlySalary) AS avg_salary**: Calculates the average monthly salary for each department and labels it as avg_salary.
3. **MIN(MonthlySalary) AS min_salary**: Finds the minimum monthly salary in each department and labels it as min_salary.
4. **MAX(MonthlySalary) AS max_salary**: Finds the maximum monthly salary in each department and labels it as max_salary.
5. **STDDEV(MonthlySalary) AS st_salary**: Calculates the standard deviation of the monthly salary for each department and labels it as st_salary, which helps to analyze the salary distribution.
6. **FROM EPES**: Specifies the EPES table as the data source.

7. **GROUP BY Department**: Groups the data by department to perform aggregate calculations for each department.
8. **ORDER BY avg_salary DESC**: Sorts the results by average salary in descending order, showing the departments with the highest average salaries first.

Week 3: Advanced Level

1: Identify the top 5 employees with the highest performance scores across all departments and list their department, job title, and years at the company.

```
select employeeid ,
name, department, jobtitle, performancescore ,
yearsatcompany from EPES
order by performancescore desc
limit 5;
```



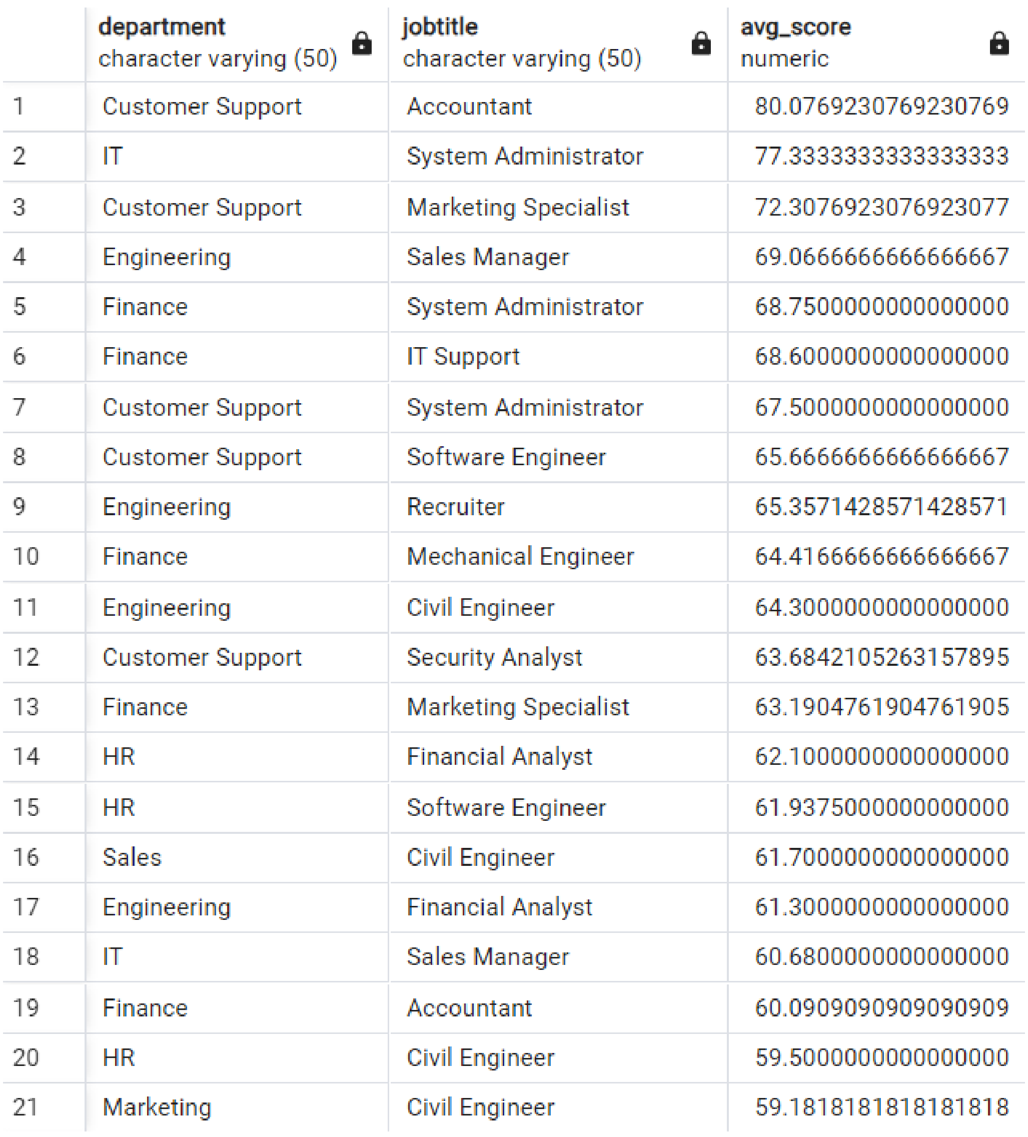
	employeeid [PK] character varying	name character varying (50)	department character varying (50)	jobtitle character varying (50)	yearsatcompany integer
1	EMP00143	Patricia Rodriguez	Marketing	Marketing Manager	99
2	EMP00066	Amber Kidd	Finance	Security Analyst	99
3	EMP00028	Stephanie Ross	Engineering	Accountant	99
4	EMP00084	Paul Jones	Customer Support	Accountant	99
5	EMP00153	Denise Jones	HR	Marketing Manager	99

Explanation:

1. **SELECT EmployeeID, Name, Department, JobTitle, PerformanceScore, YearsAtCompany**: Retrieves the employee's ID, name, department, job title, performance score, and years at the company.
2. **FROM EPES**: Specifies the EPES table as the data source.
3. **ORDER BY PerformanceScore DESC**: Sorts the employees by their performance scores in descending order, displaying the highest scores first.
4. **LIMIT 5**: Restricts the result to the top 5 employees with the highest performance scores.

2: Determine the impact of department and job title on performance scores by analysing the average performance score for each combination.

```
select department, jobtitle,avg(performancescore) as avg_score
from EPES
group by department, jobtitle
order by avg_score desc;
```



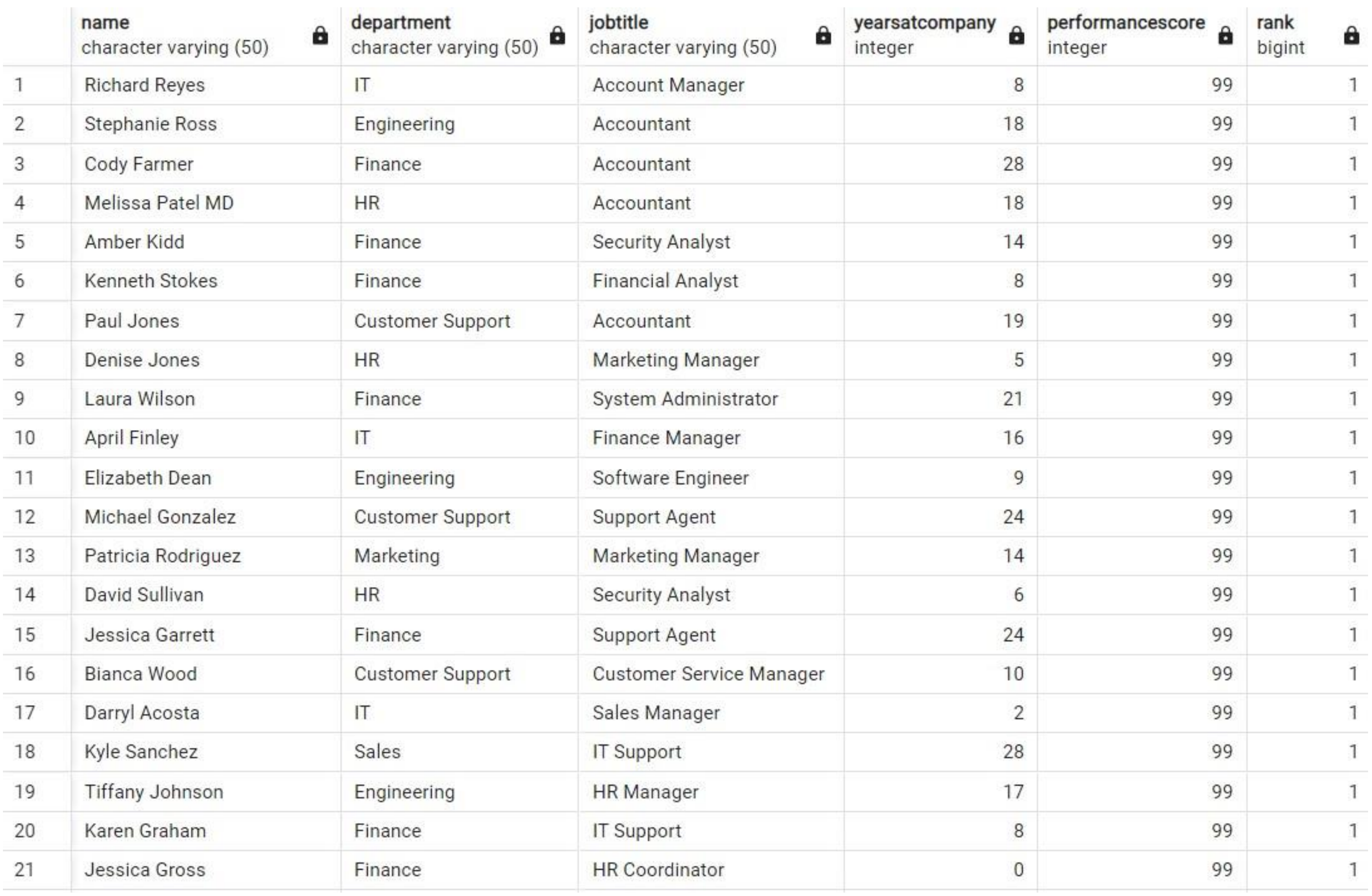
	department character varying (50)	jobtitle character varying (50)	avg_score numeric
1	Customer Support	Accountant	80.0769230769230769
2	IT	System Administrator	77.3333333333333333
3	Customer Support	Marketing Specialist	72.3076923076923077
4	Engineering	Sales Manager	69.0666666666666667
5	Finance	System Administrator	68.7500000000000000
6	Finance	IT Support	68.6000000000000000
7	Customer Support	System Administrator	67.5000000000000000
8	Customer Support	Software Engineer	65.6666666666666667
9	Engineering	Recruiter	65.3571428571428571
10	Finance	Mechanical Engineer	64.4166666666666667
11	Engineering	Civil Engineer	64.3000000000000000
12	Customer Support	Security Analyst	63.6842105263157895
13	Finance	Marketing Specialist	63.1904761904761905
14	HR	Financial Analyst	62.1000000000000000
15	HR	Software Engineer	61.9375000000000000
16	Sales	Civil Engineer	61.7000000000000000
17	Engineering	Financial Analyst	61.3000000000000000
18	IT	Sales Manager	60.6800000000000000
19	Finance	Accountant	60.0909090909090909
20	HR	Civil Engineer	59.5000000000000000
21	Marketing	Civil Engineer	59.1818181818181818

Explanation:

1. **SELECT Department, JobTitle, AVG(PerformanceScore)**: Retrieves the department, job title, and calculates the average performance score for each combination.
2. **AS avg_score**: Labels the average performance score as avg_score for clarity.
3. **FROM EPES**: Specifies the EPES table as the data source.
4. **GROUP BY Department, JobTitle**: Groups the data by both department and job title, allowing the calculation of average performance scores for each unique combination.
5. **ORDER BY avg_score DESC**: Sorts the results by average performance score in descending order, showing the combinations with the highest scores first.

3: Create a performance evaluation report that ranks employees by performance score, including their department, job title, and years at the company.

```
select name, department, JobTitle, YearsAtCompany, PerformanceScore,
    rank() over (order by PerformanceScore desc) as rank
from epes;
```



	name character varying (50)	department character varying (50)	jobtitle character varying (50)	yearsatcompany integer	performancescore integer	rank bigint
1	Richard Reyes	IT	Account Manager	8	99	1
2	Stephanie Ross	Engineering	Accountant	18	99	1
3	Cody Farmer	Finance	Accountant	28	99	1
4	Melissa Patel MD	HR	Accountant	18	99	1
5	Amber Kidd	Finance	Security Analyst	14	99	1
6	Kenneth Stokes	Finance	Financial Analyst	8	99	1
7	Paul Jones	Customer Support	Accountant	19	99	1
8	Denise Jones	HR	Marketing Manager	5	99	1
9	Laura Wilson	Finance	System Administrator	21	99	1
10	April Finley	IT	Finance Manager	16	99	1
11	Elizabeth Dean	Engineering	Software Engineer	9	99	1
12	Michael Gonzalez	Customer Support	Support Agent	24	99	1
13	Patricia Rodriguez	Marketing	Marketing Manager	14	99	1
14	David Sullivan	HR	Security Analyst	6	99	1
15	Jessica Garrett	Finance	Support Agent	24	99	1
16	Bianca Wood	Customer Support	Customer Service Manager	10	99	1
17	Darryl Acosta	IT	Sales Manager	2	99	1
18	Kyle Sanchez	Sales	IT Support	28	99	1
19	Tiffany Johnson	Engineering	HR Manager	17	99	1
20	Karen Graham	Finance	IT Support	8	99	1
21	Jessica Gross	Finance	HR Coordinator	0	99	1

Explanation:

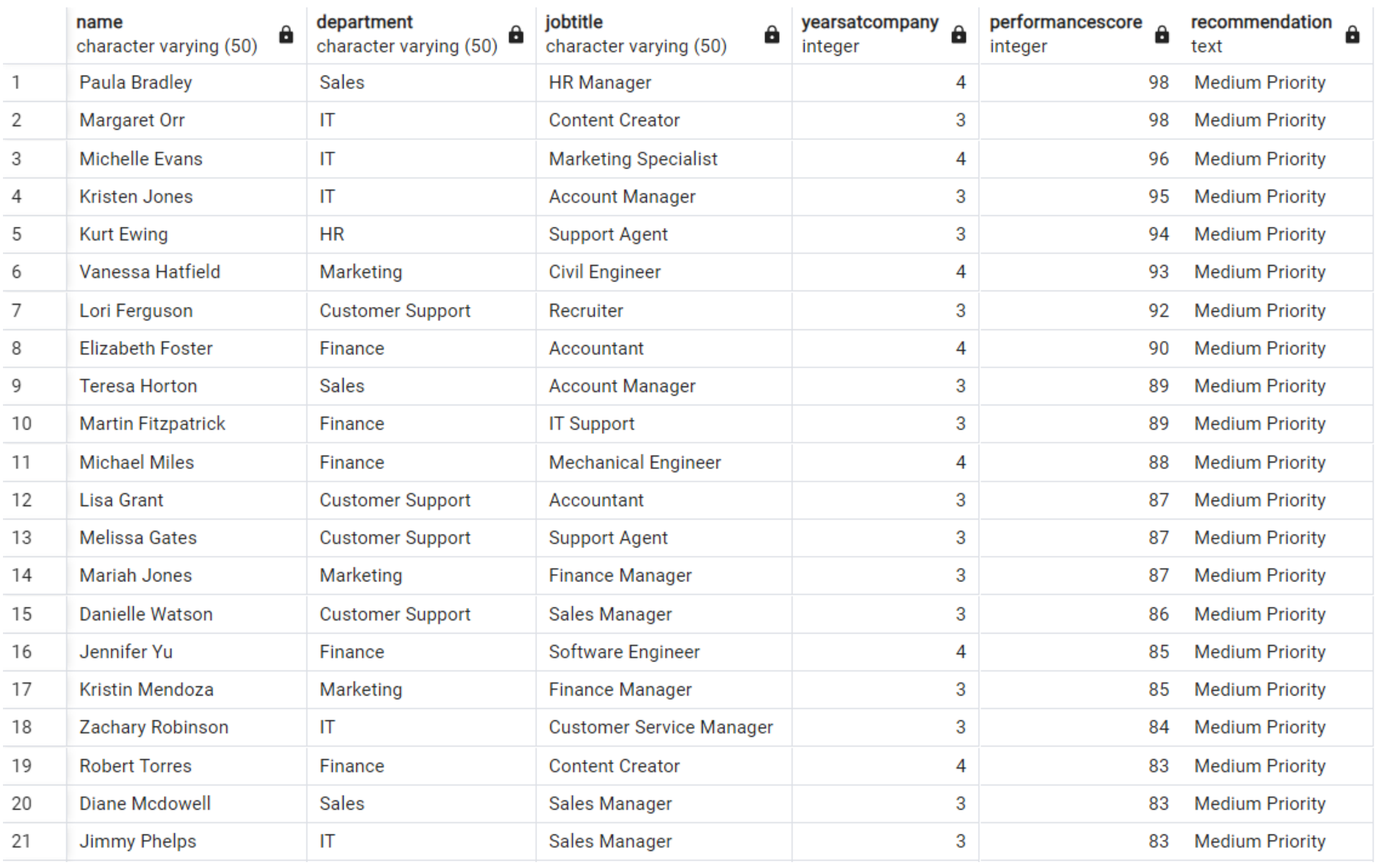
1.SELECT Name, Department, JobTitle, YearsAtCompany, PerformanceScore: Retrieves the employee's name, department, job title, years at the company, and performance score from the EPES table.

2. RANK() OVER (ORDER BY PerformanceScore DESC) AS Rank: Calculates the rank of each employee based on their performance score in descending order, assigning the highest score a rank of 1. Employees with the same score receive the same rank.

3. FROM EPES: Specifies the EPES table as the data source.

4: Develop a recommendation system to suggest potential promotions based on years at the company, job title, and performance score.

```
select Name, Department, JobTitle, YearsAtCompany, PerformanceScore,
case
when YearsAtCompany >= 5 and PerformanceScore>=60 then 'High Priority'
when YearsAtCompany >= 3 and  PerformanceScore>=45 then 'Medium Priority'
else 'Low Priority'
end as Recommendation
from EPES
order by Recommendation desc, PerformanceScore desc, YearsAtCompany desc;
```



	name character varying (50)	department character varying (50)	jobtitle character varying (50)	yearsatcompany integer	performancescore integer	recommendation text
1	Paula Bradley	Sales	HR Manager	4	98	Medium Priority
2	Margaret Orr	IT	Content Creator	3	98	Medium Priority
3	Michelle Evans	IT	Marketing Specialist	4	96	Medium Priority
4	Kristen Jones	IT	Account Manager	3	95	Medium Priority
5	Kurt Ewing	HR	Support Agent	3	94	Medium Priority
6	Vanessa Hatfield	Marketing	Civil Engineer	4	93	Medium Priority
7	Lori Ferguson	Customer Support	Recruiter	3	92	Medium Priority
8	Elizabeth Foster	Finance	Accountant	4	90	Medium Priority
9	Teresa Horton	Sales	Account Manager	3	89	Medium Priority
10	Martin Fitzpatrick	Finance	IT Support	3	89	Medium Priority
11	Michael Miles	Finance	Mechanical Engineer	4	88	Medium Priority
12	Lisa Grant	Customer Support	Accountant	3	87	Medium Priority
13	Melissa Gates	Customer Support	Support Agent	3	87	Medium Priority
14	Mariah Jones	Marketing	Finance Manager	3	87	Medium Priority
15	Danielle Watson	Customer Support	Sales Manager	3	86	Medium Priority
16	Jennifer Yu	Finance	Software Engineer	4	85	Medium Priority
17	Kristin Mendoza	Marketing	Finance Manager	3	85	Medium Priority
18	Zachary Robinson	IT	Customer Service Manager	3	84	Medium Priority
19	Robert Torres	Finance	Content Creator	4	83	Medium Priority
20	Diane Mcdowell	Sales	Sales Manager	3	83	Medium Priority
21	Jimmy Phelps	IT	Sales Manager	3	83	Medium Priority

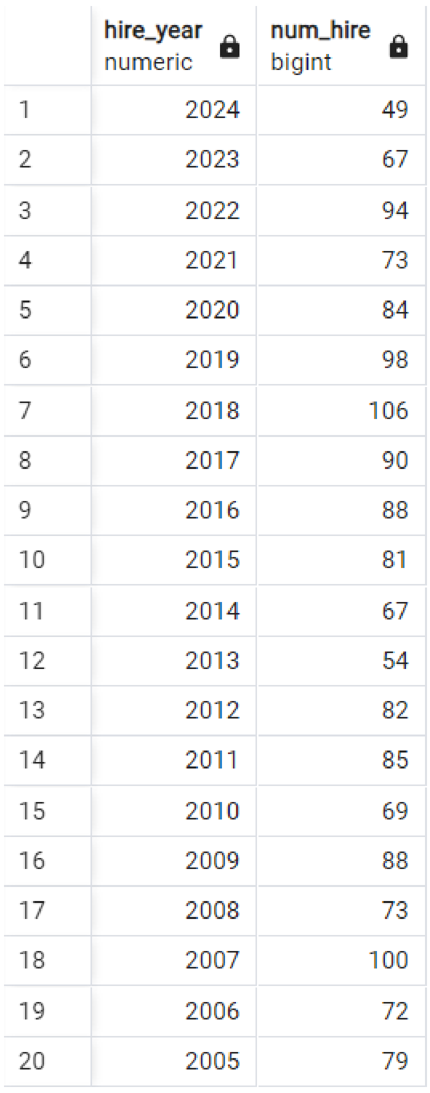
Explanation:
1. SELECT Name, Department, JobTitle, YearsAtCompany, PerformanceScore: Retrieves the employee's name, department, job title, years at the company, and performance score from the EPES table.
2. CASE...END AS Recommendation: Evaluates each employee's years at the company and performance score to assign a promotion recommendation:

- **'High Priority'**: Assigned if an employee has 5 or more years at the company and a performance score of 60 or higher.
- **'Medium Priority'**: Assigned if an employee has 3 or more years at the company and a performance score of 45 or higher.
- **'Low Priority'**: Assigned to all other employees.

3. FROM EPES: Specifies the EPES table as the data source.
4. ORDER BY Recommendation DESC, PerformanceScore DESC, YearsAtCompany DESC: Sorts the results first by recommendation priority (highest to lowest), then by performance score and years at the company, both in descending order.

5: Perform a year-wise analysis of hiring trends, identifying the number
--of employees hired each year and any notable patterns.

```
select extract(year from hiredate) as hire_year,
count(*) as num_hire
from EPES
group by hire_year
order by hire_year desc;
```



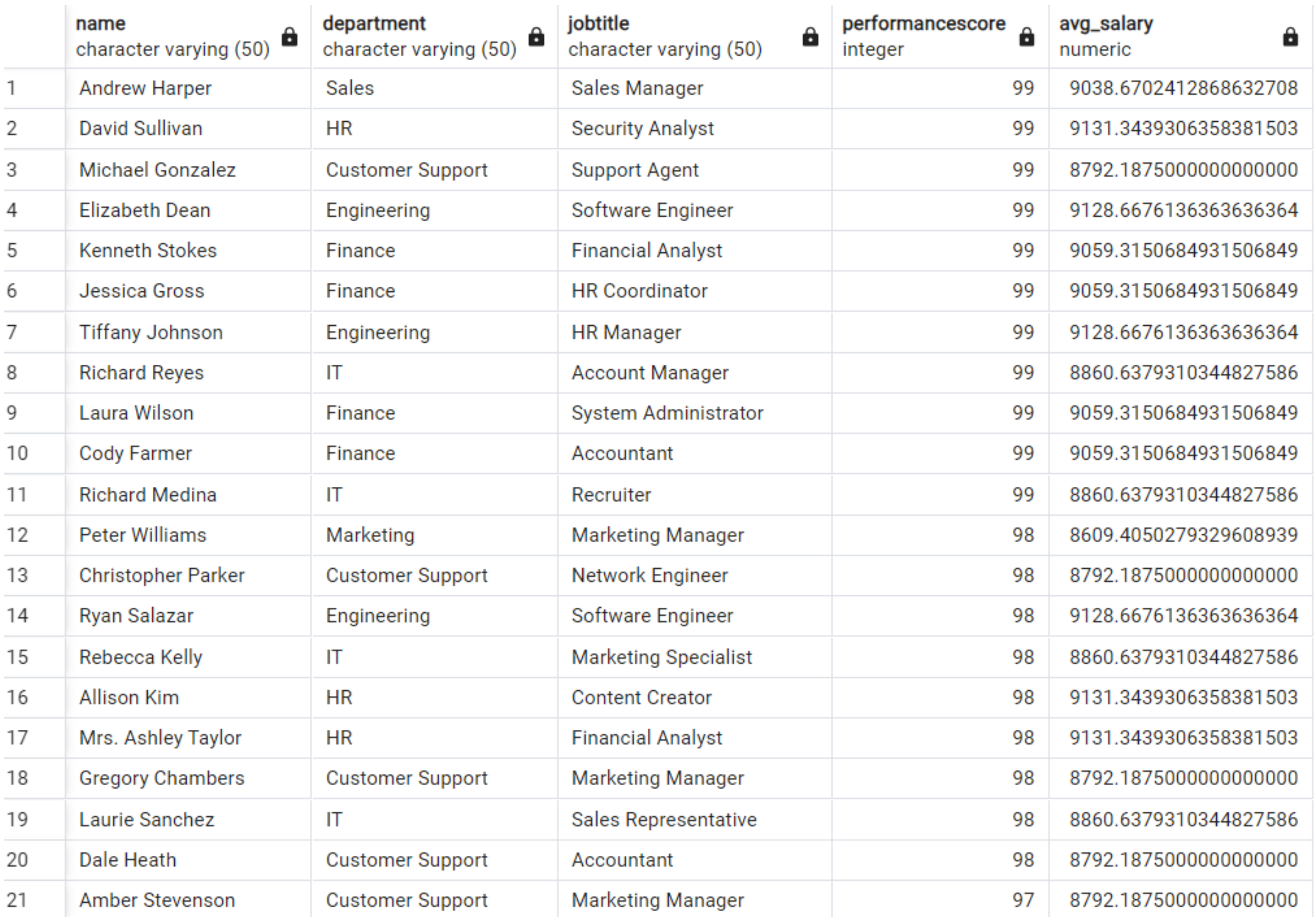
	hire_year numeric	num_hire bigint
1	2024	49
2	2023	67
3	2022	94
4	2021	73
5	2020	84
6	2019	98
7	2018	106
8	2017	90
9	2016	88
10	2015	81
11	2014	67
12	2013	54
13	2012	82
14	2011	85
15	2010	69
16	2009	88
17	2008	73
18	2007	100
19	2006	72
20	2005	79

Explanation:

1.SELECT EXTRACT(YEAR FROM HireDate) AS hire_year: Extracts the year from the HireDate column and labels it as hire_year.
2. COUNT(*) AS num_hire: Counts the total number of employees hired in each year and labels this count as num_hire.
3.FROM EPES: Specifies the EPES table as the data source.
4.GROUP BY hire_year: Groups the results by the extracted hire year, allowing for the counting of employees hired in each specific year.
5.ORDER BY hire_year DESC: Sorts the results by the hire year in descending order, showing the most recent years first.

6: Generate a report to identify employees whose salaries are below the department average and have a high performance score (above 80).

```
with avgsalary as (
select department, avg(monthlysalary) as avg_salary
        from EPES
        group by department
        )
select e.name, e.department, e.jobtitle,e.performancescore, a.avg_salary
from EPES e
join avgsalary a
on e.department = a.department
where performancescore >80 and
a.avg_salary >e.monthlysalary
order by performancescore desc ;
```



	name character varying (50)	department character varying (50)	jobtitle character varying (50)	performancescore integer	avg_salary numeric
1	Andrew Harper	Sales	Sales Manager	99	9038.6702412868632708
2	David Sullivan	HR	Security Analyst	99	9131.3439306358381503
3	Michael Gonzalez	Customer Support	Support Agent	99	8792.1875000000000000
4	Elizabeth Dean	Engineering	Software Engineer	99	9128.6676136363636364
5	Kenneth Stokes	Finance	Financial Analyst	99	9059.3150684931506849
6	Jessica Gross	Finance	HR Coordinator	99	9059.3150684931506849
7	Tiffany Johnson	Engineering	HR Manager	99	9128.6676136363636364
8	Richard Reyes	IT	Account Manager	99	8860.6379310344827586
9	Laura Wilson	Finance	System Administrator	99	9059.3150684931506849
10	Cody Farmer	Finance	Accountant	99	9059.3150684931506849
11	Richard Medina	IT	Recruiter	99	8860.6379310344827586
12	Peter Williams	Marketing	Marketing Manager	98	8609.4050279329608939
13	Christopher Parker	Customer Support	Network Engineer	98	8792.1875000000000000
14	Ryan Salazar	Engineering	Software Engineer	98	9128.6676136363636364
15	Rebecca Kelly	IT	Marketing Specialist	98	8860.6379310344827586
16	Allison Kim	HR	Content Creator	98	9131.3439306358381503
17	Mrs. Ashley Taylor	HR	Financial Analyst	98	9131.3439306358381503
18	Gregory Chambers	Customer Support	Marketing Manager	98	8792.1875000000000000
19	Laurie Sanchez	IT	Sales Representative	98	8860.6379310344827586
20	Dale Heath	Customer Support	Accountant	98	8792.1875000000000000
21	Amber Stevenson	Customer Support	Marketing Manager	97	8792.1875000000000000

Explanation:

1.WITH avgSalary AS (...): This Common Table Expression (CTE) calculates the average monthly salary for each department and creates a temporary table called avgSalary.

- **SELECT Department, AVG(MonthlySalary) AS avg_salary FROM EPES GROUP BY Department**: Retrieves each department and calculates the average salary for that department.

2.SELECT e.Name, e.Department, e.JobTitle, e.PerformanceScore, a.avg_salary: Retrieves the employee's name, department, job title, performance score, and the average salary for their department from the results.
3. FROM EPES e: Specifies the EPES table with an alias e for clarity.
4. JOIN avgSalary a ON e.Department = a.Department: Joins the original table with the CTE avgSalary on the department, allowing access to both employee details and department average salaries.
5.WHERE e.PerformanceScore > 80 AND a.avg_salary > e.MonthlySalary: Filters the results to include only those employees whose performance score is above 80 and whose salary is below their department's average salary.
6.ORDER BY e.PerformanceScore DESC: Sorts the final results by performance score in descending order, showing the highest-performing employees first.