HR DATABASE MANAGEMENT SYSTEM

Project Introduction:

This project involves analysing and managing the data of a large organizational system, specifically focusing on employee-related information, job roles, departments, and locations. The goal is to optimize the company's operations by leveraging structured data stored across several interconnected tables. The core of the project lies in exploring the relationships between employees, their respective jobs, departments, and locations, as well as handling information regarding employee dependents and company regions.
The database structure is designed with a focus on the following key areas:

1. **Employee Management:**
    - The employees' personal details, job roles, and departmental affiliations are tracked, enabling the organization to manage employee data efficiently.
    - The system records each employee's salary, job title, hire date, and other details to assist in payroll processing and employee management.
2. **Department and Job Management:**
    - Each department within the company is linked to its specific location and the jobs offered within it.
    - The employees' job roles, salaries, and associated departments are interlinked to provide a comprehensive view of the organizational structure.
3. **Geographical and Regional Data:**
    - Locations, countries, and regions provide valuable insights into where employees and departments are located globally.
    - This geographical data helps the company optimize its resources across different locations and regions.
4. **Employee Dependents:**
    - The dependents of employees are recorded to facilitate benefits management and to understand the scope of resources required for employee support.

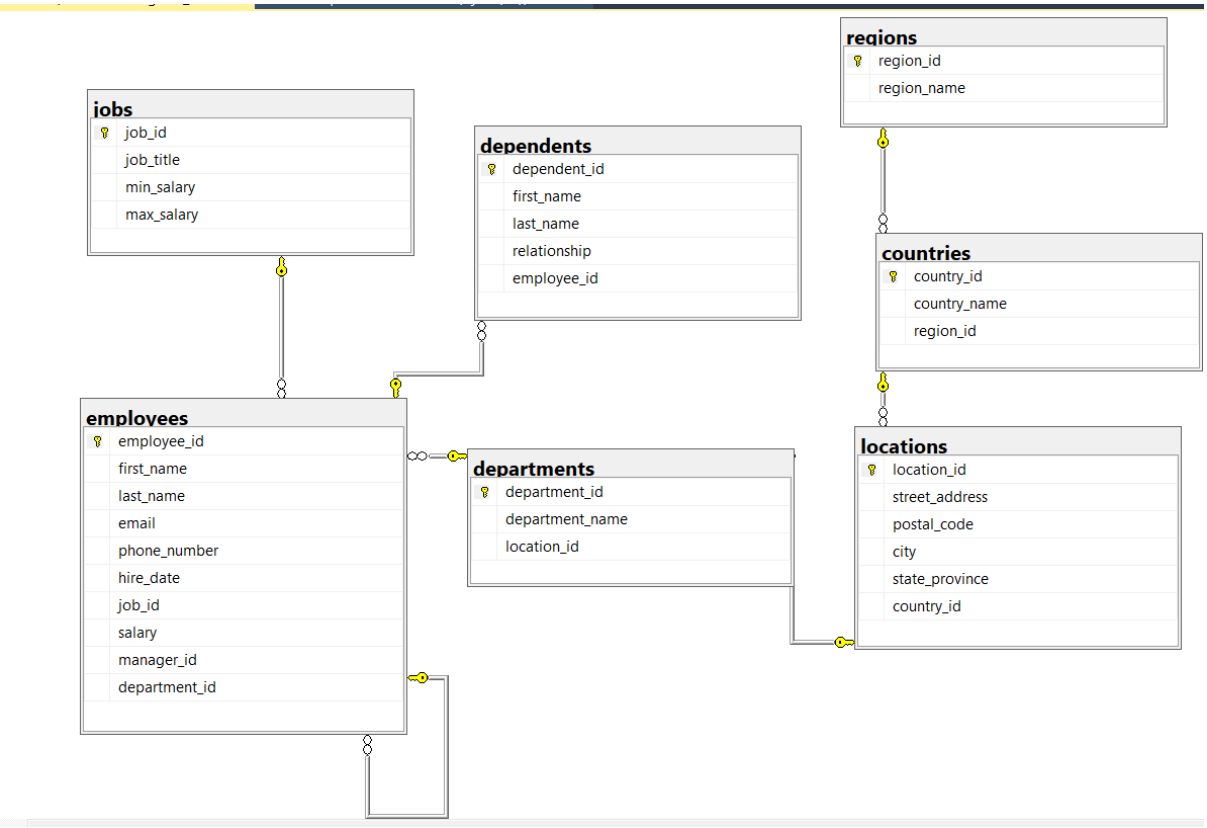
The data model incorporates several relationships between tables, allowing for advanced queries to answer complex questions, such as:

- How many employees work in each department?
- What are the most common job titles within specific regions?
- How do employee salaries vary by location or department?

By leveraging this database, the company can not only manage day-to-day operations more effectively but also derive insights to support strategic decisions related to human resources, departmental performance, and global operations.

Entity Relationship Diagram

The Entity-Relationship Diagram (ERD) illustrates the relationships between key entities in the database, such as employees, jobs, departments, and locations. It shows how these entities are interconnected to manage and analyze organizational data efficiently.



Task 1

1)WRITE A QUERY FOR SELECT STATEMENTS :-

Syntax of SELECT STATEMENT:-
SELECT
select_list
FROM
table_name; */

A. To get data from all the rows and columns in the employees table:

```
select * from employees
```

B. select data from the employee id, first name, last name, and hire date of all rows inthe employees table:

```
select employee_id, first_name, last_name, hire_date from employees
```

C. to get the first name, last name, salary, and new salary:

D. Increase the salary two times and named as New_SALARY from employees table

```
select first_name, last_name, salary, (salary*2) as  new_salary from employees
```

/*

2)WRITE A QUERY FOR ORDER BY STATEMENTS :-

Syntax of ORDER BY Statements:-

```
SELECT
 select_list
FROM
 table_name
ORDER BY
 sort_expression1 [ASC | DESC],
 sort_expression 2[ASC | DESC]; */
```

A. returns the data from the employee id, first name, last name, hire date, and salary column of the employees table:

B. to sort employees by first names in alphabetical order:

```
select employee_id, first_name, last_name, hire_date, salary from employees
order by
first_name
```

C. to sort the employees by the first name in ascending order and the last name in descending order:

```
select employee_id, first_name, last_name, hire_date, salary from employees
order by
first_name ,
last_name desc
```

D. to sort employees by salary from high to low:

```
select employee_id, first_name, last_name, hire_date, salary from employees
order by
salary desc
```

E. to sort the employees by values in the hire_date column from:

```
select employee_id, first_name, last_name, hire_date, salary from employees
order by
hire_date
```

F. sort the employees by the hire dates in descending order:

```
select employee_id, first_name, last_name, hire_date, salary from employees
order by
hire_date desc
```

/*

3)WRITE A QUERY FOR DISTINCT STATEMENTS :-

Syntax of DISTINCT Statements:-

```
SELECT DISTINCT
 column1, column2, ...
FROM
 table1; */
```

A. selects the salary data from the salary column of the employees table and sorts them from high to low:

```
select * from employees order by salary desc
```

B. select unique values from the salary column of the employees table:

```
select distinct salary from employees
```

C. selects the job id and salary from the employees table:

```
select job_id, salary from employees
```

D. to remove the duplicate values in job id and salary:

```
select distinct job_id, salary from employees
```

E. returns the distinct phone numbers of employees: */

```
select distinct phone_number from employees
```

/*

4)WRITE A QUERY FOR TOP N STATEMENTS :-

Syntax of TOP N Statements(N=Will be any nos)

```
SELECT TOP N
 column_list
FROM
 table1
ORDER BY column_list*/
```

A. returns all rows in the employees table sorted by the first_name column.

```
select * from employees order by first_name
```

B. to return the first 5 rows in the result set returned by the SELECT clause:

```
select top 5 * from employees order by first_name
```

C. to return five rows starting from the 4th row:

```
select top 5 * from employees where first_name not in (select top 2 first_name from employees order by first_name) order by first_name
```

D. gets the top five employees with the highest salaries.

```
select top 5 * from employees order by salary desc
```

E. to get employees who have the 2nd highest salary in the company

```
select * from employees where salary = (select top 1 salary from employees where salary < (select max(salary) from employees) order by salary desc)
```

5)WRITE A QUERY FOR WHERE CLAUSE and COMPARISON OPERATORS :-

A. query finds employees who have salaries greater than 14,000 and sorts the results sets based on the salary in descending order.

```
select * from employees where salary > 14000 order by salary desc
```

B. query finds all employees who work in the department id 5.

```
select * from employees where department_id = 5
```

C. query finds the employee whose last name is Chen

```
select * from employees where last_name ='chen'
```

D. To get all employees who joined the company after January 1st, 1999

```
select * from employees where hire_date > '1999-01-01'
```

E. to find the employees who joined the company in 1999,

```
select * from employees where year(hire_date)= '1999'
```

F. statement finds the employee whose last name is Himuro

```
select * from employees where last_name ='Himuro'
```

G. the query searches for the string Himuro in the last_name column of the employees table.

```
select * from employees where last_name like '%Himuro%'
```

H. to find all employees who do not have phone numbers:

```
select * from employees where phone_number is null
```

I. returns all employees whose department id is not 8.

```
select * from employees where department_id != 8
```

J. finds all employees whose department id is not eight and ten.

```
select * from employees where department_id not in (8,10)
```

K. to find the employees whose salary is greater than 10,000,

```
select * from employees where salary > 10000
```

L. finds employees in department 8 and have the salary greater than 10,000:

```
select * from employees where salary > 10000 and department_id = 8
```

M. the statement below returns all employees whose salaries are less than 10,000:

```
select * from employees where salary < 10000
```

N. finds employees whose salaries are greater than or equal 9,000:

```
select * from employees where salary >= 9000
```

O. finds employees whose salaries are less than or equal to 9,000:

```
select * from employees where salary <= 9000
```

6)WRITE A QUERY FOR:-

```
CREATE TABLE courses (
  course_id INT PRIMARY KEY,
  course_name VARCHAR(100) NOT NULL
);
```

A.adds a new column named credit_hours to the courses table.

```
alter table courses add credit_hours int;
```

B. adds the fee and max_limit columns to the courses table and places these columns after the course_name column.

```
alter table courses
add fee decimal(10, 2),
  max_limit int;
```

C. changes the attribute of the fee column to NOT NULL.

```
alter table courses
alter column fee decimal(10, 2) not null;
```

D. to remove the fee column of the courses table

```
alter table courses
drop column fee;
```

E. removes the max_limit and credit_hours of the courses table.

```
alter table courses
drop column max_limit, credit_hours;
```

6)WRITE A QUERY FOR:-

```
CREATE TABLE projects (
 project_id INT PRIMARY KEY,
 project_name VARCHAR(255),
 start_date DATE NOT NULL,
 end_date DATE NOT NULL
);
CREATE TABLE project_milestones(
 milestone_id INT PRIMARY KEY,
 project_id INT,
 milestone_name VARCHAR(100)
);
```

A. to add an SQL FOREIGN KEY constraint to the project_milestones table to enforce the relationship between the projects and project_milestones tables.

```
alter table project_milestones
add constraint fk_project_id
foreign key (project_id) references projects(project_id);
```

B. Suppose the project_milestones already exists without any predefined foreign key and you want to define a FOREIGN KEY constraint for the project_id column so write a Query to add a FOREIGN KEY constraint to existing table

```
alter table project_milestones
add constraint fk_project_id
foreign key (project_id) references projects(project_id);
```

TASK 2

1)WRITE A QUERY FOR LOGICAL OPERATORS and OTHER ADVANCED OPERATORS:-

Part 1:-

A. finds all employees whose salaries are greater than 5,000 and less than 7,000:

```
select * from employees where salary > 5000 and salary < 7000
```

B. finds employees whose salary is either 7,000 or 8,000:

```
select * from employees where salary = 8000 or salary = 7000
```

C. finds all employees who do not have a phone number:

```
select * from employees where phone_number is null
```

D. finds all employees whose salaries are between 9,000 and 12,000.

```
select * from employees where salary between 9000 and 12000
```

E. finds all employees who work in the department id 8 or 9.

```
select * from employees where department_id in (8,9)
```

F. finds all employees whose first name starts with the string jo

```
select * from employees where first_name like 'jo%'
```

G. finds all employees with the first names whose the second character is h

```
select * from employees where SUBSTRING(first_name,2,1) ='h'
```

H. finds all employees whose salaries are greater than all salaries of employees in the department 8:

```
select * from employees where salary > (select MAX(salary) from employees where department_id = 8)
```

Part 2:-

A. finds all employees whose salaries are greater than the average salary of every department:

```
select * from employees where salary >(select avg(salary) from employees)
```

B. finds all employees who have dependents:

```
select distinct e.employee_id, e.first_name, e.last_name from employees e join dependents d on e.employee_id = d.employee_id
```

C. to find all employees whose salaries are between 2,500 and 2,900:

```
select * from employees where salary between 2500 and 2900
```

D. to find all employees whose salaries are not in the range of 2,500 and 2,900:

```
select * from employees where salary not between 2500 and 2900 order by salary
```

E. to find all employees who joined the company between January 1, 1999, and December 31, 2000:

```
select * from employees where hire_date between '1999-01-01' and '2000-12-31'
```

F. to find employees who have not joined the company from January 1, 1989 to December 31, 1999:

```
select * from employees where hire_date not between '1999-01-01' and '1999-12-31'
```

G. to find employees who joined the company between 1990 and 1993:

```
select * from employees where year(hire_date) between '1990' and '1993'
```

Part 3:-

A. to find all employees whose first names start with Da

```
select * from employees where first_name like 'Da%'
```

B. to find all employees whose first names end with er

```
select * from employees where first_name like '%er'
```

C. to find employees whose last names contain the word an:

```
select * from employees where last_name like '%an%'
```

D. retrieves employees whose first names start with Jo and are followed by at most 2 characters:

```
select * from employees where first_name like 'jo_' or first_name like 'jo__'
```

E. to find employees whose first names start with any number of characters and are followed by atmost one character:

```
select * from employees where first_name like '_%'
```

F. to find all employees whose first names start with the letter S but not start with Sh:

```
select * from employees where first_name like 'S%' and first_name not like '%sh'
```

Part 4:-

A. retrieves all employees who work in the department id 5.

```
select * from employees where department_id = 5
```

B. To get the employees who work in the department id 5 and with a salary not greater than 5000.

```
select * from employees where department_id = 5 and salary < 5000
```

C. statement gets all the employees who are not working in the departments 1, 2, or 3.

```
select * from employees where department_id not in (1,2,3)
```

D. retrieves all the employees whose first names do not start with the letter D.

```
select * from employees where first_name not like 'D%'
```

E. to get employees whose salaries are not between 5,000 and 1,000.

```
select * from employees where salary not between 1000 and 5000
```

part 5

A. Write a query to get the employees who do not have any dependents by above image

```
select distinct e.employee_id, e.first_name, e.last_name  from employees e left join dependents d on e.employee_id = d.employee_id  where d.dependent_id is NULL
```

B. To find all employees who do not have the phone numbers

```
select * from employees where phone_number is null
```

C. To find all employees who have phone numbers

```
select * from employees where phone_number is not null
```

Task 3

JOINS:-

inner joins

1) Write a Query to

A. To get the information of the department id 1,2, and 3

```
select * from departments where department_id in (1,2,3)
```

B. To get the information of employees who work in the department id 1, 2 and 3

```
select e.employee_id, e.first_name, e.last_name,e.department_id from employees e
join departments d on e.department_id = d.department_id where d.department_id in (1,2,3)
```

Write a Query to get the first name, last name, job title, and department name of employees who work in department id 1, 2, and 3.

LEFT JOIN

Write a Query :

A. To query the country names of US, UK, and China

```
select * from countries where country_id in ('US','UK','CN')
```

B. query retrieves the locations located in the US, UK and China:

```
select l.city,c.country_name from countries c
join locations l on c.country_id = l.country_id
where c.country_id in ('US','UK','CN')
```

C. To join the countries table with the locations table

```
select c.country_name,l.city from countries c
join locations l on c.country_id = l.country_id
```

D. to find the country that does not have any locations in the locations table

```
select c.country_name,l.city from countries c
left join locations l on c.country_id = l.country_id
where l.city is null
```

Write a query to join 3 tables: regions, countries, and locations

```
select r.region_name,c.country_name,l.city from countries c
join locations l on c.country_id = l.country_id
join regions r on c.region_id = r.region_id
```

self-join

The manager_id column specifies the manager of an employee. Write a query statement to joins the employees table to itself to query the information of who reports to whom.

```
select e.first_name, e.last_name, m.first_name as manager from employees e join employees m on e.manager_id = m.employee_id
```

Now write a Query To include the president in the result set:-

```
select e.first_name, e.last_name, m.first_name as manager  from employees e left join employees m on e.manager_id = m.employee_id
```

FULL OUTER JOIN

```
CREATE TABLE fruits (
fruit_id INT PRIMARY KEY,
fruit_name VARCHAR (255) NOT NULL,
basket_id INTEGER
);
CREATE TABLE baskets (
basket_id INT PRIMARY KEY,
basket_name VARCHAR (255) NOT NULL
);
INSERT INTO baskets (basket_id, basket_name)
VALUES
(1, 'A'),
(2, 'B'),
(3, 'C');
INSERT INTO fruits (
fruit_id,
fruit_name,
basket_id
)
VALUES
(1, 'Apple', 1),
(2, 'Orange', 1),
(3, 'Banana', 2),
(4, 'Strawberry', NULL);
```

Question:-

A. Write a query to returns each fruit that is in a basket and each basket that has a fruit, but also returns each fruit that is not in any basket and each basket that does not have any fruit.

```
select * from fruits f full outer join baskets b on f.basket_id = b.basket_id
```

B. Write a query to find the empty basket, which does not store any fruit

```
select * from fruits f full outer join baskets b on f.basket_id = b.basket_id where f.fruit_id is null
```

C. Write a query which fruit is not in any basket

```
select * from fruits f full outer join baskets b on f.basket_id = b.basket_id where b.basket_id is null
```

CROSS JOIN

```
CREATE TABLE sales_organization (
sales_org_id INT PRIMARY KEY,
sales_org VARCHAR (255)
);
CREATE TABLE sales_channel (
channel_id INT PRIMARY KEY,
channel VARCHAR (255)
);
INSERT INTO sales_organization (sales_org_id, sales_org)
VALUES
(1, 'Domestic'),
(2, 'Export');
INSERT INTO sales_channel (channel_id, channel)
VALUES
(1, 'Wholesale'),
(2, 'Retail'),
```

(3, 'eCommerce'),
(4, 'TV Shopping');

Write a Query To find the all possible sales channels that a sales organization

```
select so.sales_org,sc.channel from sales_channel sc cross join sales_organization so
```

Task 4

Write a Query

A. to group the values in department_id column of the employees table:

```
select department_id from employees group by department_id
```

B. to count the number of employees by department:

```
select department_id, count(employee_id) as total_employees from employees group by department_id
```

C. returns the number of employees by department

```
select department_id, count(employee_id) as total_employees from employees group by department_id
```

D. to sort the departments by headcount:

```
select department_id, count(employee_id) as total_employees from employees group by department_id order by total_employees desc
```

E. to find departments with headcounts are greater than 5:

```
select department_id, count(employee_id) as total_employees from employees group by department_id having count(*) >5
```

F. returns the minimum, maximum and average salary of employees in each department.

```
select department_id, min(salary) as min_salary, max(salary) as max_salary from employees group by department_id
```

G. To get the total salary per department,

```
select department_id, SUM(salary) as total_salary from employees group by department_id
```

H. groups rows with the same values both department_id and job_id columns in the same group then return the rows for each of these groups

```
select department_id,job_id ,count(employee_id) as total_employees from employees group by department_id,job_id order by department_id,job_id
```

Write a Query

A. To get the managers and their direct reports, and to group employees by the managers and to count the direct reports.

```
select m.first_name ,m.last_name, COUNT(e.employee_id) as direct_reporters from employees e join employees m on e.manager_id = m.employee_id
```

```
group by m.first_name,m.last_name
```

B. To find the managers who have at least five direct reports

```
select m.first_name ,m.last_name, COUNT(e.employee_id) as direct_reporters from employees e join employees m
on e.manager_id = m.employee_id
group by m.first_name,m.last_name having COUNT(e.employee_id) >= 5
```

C. calculates the sum of salary that the company pays for each department and selects only the departments with the sum of salary between 20000 and 30000.

```
select d.department_name, SUM(salary) as total_salary from employees e
join departments d on e.department_id = d.department_id
group by d.department_name having sum(salary) between 20000 and 30000
```

D. To find the department that has employees with the lowest salary greater than 10000

```
select d.department_name, min(salary) as min_salary from employees e
join departments d on e.department_id = d.department_id
group by d.department_name having min(salary) > 10000
```

E. To find the departments that have the average salaries of employees between 5000 and 7000

```
select d.department_name, avg(salary) as avg_salary from employees e
join departments d on e.department_id = d.department_id
group by d.department_name having avg(salary) between 5000 and 7000
```

Task 5

Write a Query to combine the first name and last name of employees and dependents

```
select CONCAT(first_name,' ',last_name) as full_name from employees
union
select CONCAT(first_name,' ',last_name) as full_name from dependents
```

Write a Query to Applies the INTERSECT operator to the A and B tables and sorts the combined result set by the id column in descending order.

Create Table A

```
CREATE TABLE A (
  id INT PRIMARY KEY
);
```

Insert Data into Table A

```
INSERT INTO A (id) VALUES
(1),
(2),
(3);
```

Create Table B

```
CREATE TABLE B (
  id INT PRIMARY KEY
);
```

Insert Data into Table B

```
INSERT INTO B (id) VALUES
(2),
(3),
(4);
```

Query to Find Intersection and Sort in Descending Order

```
select id from A
intersect
select id from B
order by id desc
```

Write a Query

A. finds all employees who have at least one dependent.

```
select distinct e.employee_id, e.first_name, e.last_name from employees e
where exists
(select 1 from dependents d where e.employee_id = d.employee_id )
```

B . finds employees who do not have any dependents:

```
select distinct e.employee_id, e.first_name, e.last_name from employees e
where not exists
(select 1 from dependents d where e.employee_id = d.employee_id )
```

Questions:-

A. Suppose the current year is 2000. How to use the simple CASE expression to get the work anniversaries of employees by

```
select first_name, last_name,
case
when 2000 - year(hire_date) = 0 then 'Anniversary this year'
else concat(2000 - year(hire_date), ' Anniversary')
end as work_anniversary
from employees
where YEAR(hire_date) <= 2000
```

B. Write a Query If the salary is less than 3000, the CASE expression returns “Low”. If the salary is between 3000 and 5000, it returns “average”. When the salary is greater than 5000, the CASE expression returns “High”.

```
select first_name, last_name, salary,
case
when salary < 3000 then 'Low'
when salary between 3000 and 5000 then 'Average'
else 'High'
end as salary_range
from employees
```

Write a Query to update Sarah’s last name from Bell to Lopez

```
update employees set last_name = 'Lopez' where first_name = 'Sarah' and last_name = 'Bell'
select * from employees where first_name = 'Sarah'
```

How to make sure that the last names of children are always matched with the last name of parents in the employees table,

```
update  d
set d.last_name = e.last_name
from dependents d
join employees e on e.employee_id = d.employee_id
where d.relationship = 'Child'

select e.last_name,d.last_name from dependents d join employees e on e.employee_id = d.employee_id
```

Write a Query :-

A. Combine Above two queries using subquery inorder find all departments located at the location whose id is 1700 and find all employees that belong to the location 1700 by using the department id list of the previous query

```
SELECT employee_id, first_name, last_name
FROM employees
WHERE department_id IN
(SELECT department_id FROM departments WHERE location_id = 1700)
ORDER BY first_name , last_name;
```

B. to find all employees who do not locate at the location 1700:

```
SELECT employee_id, first_name, last_name
FROM employees
WHERE department_id not IN
(SELECT department_id FROM departments WHERE location_id = 1700)
ORDER BY first_name , last_name;
```

C. finds the employees who have the highest salary:

```
select * from employees where salary = (select max(salary) from employees )
```

D. finds all employees who salaries are greater than the average salary of all employees:

```
select * from employees where salary > (select avg(salary) from employees )
```

E. finds all departments which have at least one employee with the salary is greater than 10,000:

```
select department_name from departments where department_id in (select department_id from employees where salary > 10000)
```

F. finds all departments that do not have any employee with the salary greater than 10,000:

```
select department_name from departments where department_id not in (select department_id from employees where salary > 10000 )
```

G. to find the lowest salary by department:

```
select d.department_name, (select min(salary) from employees e where e.department_id = d.department_id) AS lowest_salary from departments  d
```

H. finds all employees whose salaries are greater than the lowest salary of every department:

```
select first_name,last_name, salary from employees e where salary > (select min(salary) from employees  where department_id = e.department_id)
```

I. finds all employees whose salaries are greater than or equal to the highest salary of every department

```
select first_name,last_name, salary from employees e where salary >= (select max(salary) from employees  where department_id = e.department_id)
```

J. returns the average salary of every department

```
select department_name, (select avg(salary) from employees e where e.department_id= d.department_id) as avg_salary from departments d
```

K. to calculate the average of average salary of departments :

```
select avg(avg_dept_salary) as avg_salary_of_departments
from (select d.department_name, (select avg(salary) from employees e where e.department_id= d.department_id) as avg_dept_salary from departments d)
AS department_avg_salaries
```

L. finds the salaries of all employees, their average salary, and the difference between the salary of each employee and the average salary.

```
select e.first_name,e.last_name,e.salary,
(select avg(salary) from employees) as avg_salary,
e.salary - (select avg(salary) from employees) as salary_diff
from employees e
```