**TASK1: Extract the structures of the following 7 tables using the 'DESCRIBE' SQL command from the HR database using MySQL Workbench :**

**1. regions**

**2. countries**

**3. locations**

**4. departments**

**5. jobs**

**6. employees**

**7. job_history**

## Table Name: region

| Column Name | Data Type | Description |
|---|---|---|
| region_id | int unsigned | Unique identifier for the region |
| region_name | varchar(25) | Name of the region |

## Table Name: country

| Column Name | Data Type | Description |
|---|---|---|
| country_idchar | char(2) | Unique identifier for the country (e.g., ISO 3166-1 alpha-2 code) |
| region_id | int unsigned | Identifier for the region the country belongs to (e.g., continent, subcontinent) |
| country_name | varchar(40) | Name of the country |

## Table Name: location

| Column Name | Data Type | Description |
|---|---|---|
| location_id | int unsigned | Unique identifier for the location |

| | | |
|---|---|---|
| street_address | varchar(40) | Street address of the location |
| postal_code | varchar(12) | Postal code of the location |
| city | varchar(30) | City of the location |
| state_province | varchar(25) | State or province of the location |
| country_id | char(2) | Country of the location (foreign key to country table) |

### Table Name: department

| Column Name | Data Type | Description |
|---|---|---|
| department_id | int unsigned | Unique identifier for the department |
| department_name | varchar(30) | Name of the department |
| manager_id | int unsigned | Identifier of the department manager (foreign key to employee table) |
| location_id | int unsigned | Identifier of the department's location (foreign key to location table) |

### Table Name: job

| Column Name | Data Type | Description |
|---|---|---|
| job_id | varchar(10) | Unique identifier for the job |
| job_title | varchar(35) | Title of the job |
| min_salary | decimal(8,0) unsigned | Minimum salary for the job |
| max_salary | decimal(8,0) unsigned | Maximum salary for the job |

## Employee Table Structure

### Table Name: employee

| **Column Name** | **Data Type** | **Description** |
|---|---|---|
| employee_id | int unsigned | Unique identifier for the employee |
| first_name | varchar(20) | Employee's first name |
| last_name | varchar(25) | Employee's last name |
| email | varchar(25) | Employee's email address |
| phone_number | varchar(20) | Employee's phone number |
| hire_date | date | Date when the employee was hired |
| job_id | varchar(10) | Job title of the employee (foreign key to job table) |
| salary | decimal(8,2) | Employee's salary |
| commission_pct | decimal(2,2) | Employee's commission percentage |
| manager_id | int unsigned | ID of the employee's manager (foreign key to employee table |
| department_id | int unsigned | ID of the employee's department (foreign key to department table) |

**Table Name: job_history**

| **Column Name** | **Data Type** | **Description** |
|---|---|---|
| employee_id | int unsigned | Employee ID (foreign key to employee table) |
| start_date | date | Start date of the job |
| end_date | date | End date of the job |
| job_id | varchar(10) | Job ID (foreign key to job table) |
| department_id | int unsigned | Department ID (foreign key to department table) |

**TASK2: The following business queries have been generated using 4 tables: regions, countries, locations, and departments. Generate MySQL queries**

**for all the business queries with the help of BARD using the table structure retrieved in Task 1. Then, execute these queries in MySQL to produce the actual outputs.**

a) **Find the total number of countries in each region.**

**QUERY:**

```
SELECT r.region_name, COUNT(c.country_id) AS total_countries
FROM regions r
JOIN countries c ON r.region_id = c.region_id
GROUP BY r.region_name;
```

b) **Find the top 10 largest cities by population.**

**QUERY:**

```
SELECT city,location_id as count

FROM locations

ORDER BY location_id DESC

LIMIT 10;
```

c) **Find the average salary of employees in each department**

d) **QUERY:**

```
SELECT d.department_name, AVG(e.salary) AS average_salary

FROM employees e

JOIN departments d ON e.department_id = d.department_id

GROUP BY d.department_name;
```

**d) Find the total sales for each country in the last quarter.**

**QUERY:**

```
SELECT c.country_name, SUM(od.quantity * p.price) AS total_sales

FROM orders o

JOIN order_details od ON o.order_id = od.order_id

JOIN products p ON od.product_id = p.product_id

JOIN customers c ON o.customer_id = c.customer_id
```

WHERE o.order_date BETWEEN '2023-10-01' AND '2023-12-31'  -- Adjust dates for the desired quarter

GROUP BY c.country_name;

**e) Find the top 10 most popular products, based on the number of orders.**

**QUERY:**

```
SELECT p.product_name, COUNT(od.order_id) AS total_orders
FROM products p
JOIN order_details od ON p.product_id = od.product_id
GROUP BY p.product_name
ORDER BY total_orders DESC
 LIMIT 10;
```

**f) Find the customers who have placed the most orders in the last year.**

**QUERY:**

```
SELECT c.customer_name, COUNT(o.order_id) AS total_orders
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
WHERE o.order_date >= CURDATE() - INTERVAL 1 YEAR
GROUP BY c.customer_id
ORDER BY total_orders DESC
LIMIT 10;
```

**g) Find the employees who have generated the most sales in the last quarter.**

**QUERY:**

```
SELECT e.employee_id, e.first_name, e.last_name, SUM(od.unit_price * od.quantity * (1 - od.discount)) AS total_sales
FROM employees e
JOIN orders o ON e.employee_id = o.employee_id
JOIN order_details od ON o.order_id = od.order_id
WHERE o.order_date >= CURDATE() - INTERVAL 3 MONTH
GROUP BY e.employee_id, e.first_name, e.last_name
ORDER BY total_sales DESC;
```

**Task3 : Create additional business queries using BARD with the three tables: jobs, employees, and job_history**

**Generate MySQL queries for the aforementioned business queries using BARD and the table structure retrieved in Task 1.**

**Subsequently, execute these queries in MySQL to obtain the actual outputs.**

**Query 1: Find the average salary for each job title.**

```
SELECT j.job_title, AVG(e.salary) AS average_salary
FROM employees e
JOIN jobs j ON e.job_id = j.job_id
GROUP BY j.job_title;
```

**Query 2: Determine the number of employees currently employed in each job.**

```
SELECT j.job_title, COUNT(e.employee_id) AS number_of_employees
FROM employees e
JOIN jobs j ON e.job_id = j.job_id
WHERE e.employee_id NOT IN (
   SELECT employee_id
   FROM job_history
)
GROUP BY j.job_title;
```

**Task4:  Generate additional business queries using BARD with the three tables: departments, jobs, employees.**

**Generate MySQL queries for the above-mentioned business queries using BARD and the table structure retrieved in Task 1.**

**Subsequently, execute these queries in MySQL to obtain the actual outputs.**

**Query 1: Find the average salary of employees in each department.**

**SQL**

```
SELECT d.department_name, AVG(e.salary) AS average_salary
FROM departments d
JOIN employees e ON d.department_id = e.department_id
GROUP BY d.department_name;
```

**Query 2: Determine the number of employees in each department.**

**SQL**

```
SELECT d.department_name, COUNT(e.employee_id) AS employee_count
FROM departments d
JOIN employees e ON d.department_id = e.department_id
GROUP BY d.department_name;
```

**Task5: Extract the unique queries from tasks 2-4. Based on the output from these unique queries, write the summary of your analysis. You can use BARD to generate the summary.**

## Task 5: Extracting Unique Queries and Summarizing Analysis

### Extracting Unique Queries

Based on the provided tasks, we can identify the following core query types:

1. **Aggregations:**
   - Calculate averages (average salary by department, job title)
   - Count occurrences (number of employees by department, job title)
2. **Comparisons:**
   - Find maximum or minimum values (highest paid employee, department with highest average salary)
   - Rank data (top 10 largest cities, top 10 most popular products)
3. **Joins:**
   - Combine data from multiple tables (employees and departments, orders and customers)
4. **Filtering:**
   - Select data based on specific criteria (last quarter sales, employees hired in a specific year)

### Summary of Analysis

The provided tasks and their corresponding SQL queries demonstrate fundamental SQL operations for data analysis. These operations include:

- **Data Retrieval:** Selecting specific columns and rows from tables.

- **Data Calculation:** Performing calculations on numerical data (e.g., averages, sums).
- **Data Aggregation:** Grouping data and applying aggregate functions.
- **Data Comparison:** Ordering data based on specific criteria and filtering results.
- **Data Joining:** Combining data from multiple tables based on related columns.

By effectively combining these operations, we can extract valuable insights from the underlying data. The specific queries provided in the tasks showcase how to apply these SQL concepts to address various business questions.