## Indian Institute of Information Technology Surat

****

# Lab Report on

# Advanced Database Management (CS 604) Practical

**Submitted by**

### [RAHUL KUMAR SINGH] (UI21CS44)

**Course Faculty**

### Mr. Rishi Sharma

## Department of Computer Science and Engineering

## Indian Institute of Information Technology Surat

## Gujarat-394190, India

**Jan-2024**

## Lab No: 1

**Aim: Perform basic SQL Query on three tables (Employee, Title, Bonus)**

**Description:** Create a Database for an Organization and create the following tables in the Organization Database:

*Employee(EMP\_ID(PK), FIRST\_NAME, LAST\_NAME, SALARY, JOINING\_DATE, DEPARTMENT)*

*Bonus (EMP\_REF\_ID(FK EMP\_ID), BONUS\_AMOUNT, BONUS\_DATE)*

*Title (EMP\_REF\_ID(FKEMP\_ID), EMP\_TITLE, AFFECTED\_FROM)*

Insert a minimum of 50 records in each table. Retrieve the following information from the Organization database:

1. SQL query to print all Employee details from the Employee table order by FIRST\_NAME Ascending and DEPARTMENT Descending.

2. SQL query to fetch the count of employees working in the department ‘Admin’.

3. SQL query to fetch Employee names with salaries >= 50000 and <= 100000.

4. SQL query to print details of the Workers who are also Managers.

5. SQL query to fetch duplicate records having matching data in some fields of a table.

6. SQL query to show only even rows from a table.

7. SQL query to show records from one table that another table does not have. Find employees in employee table that do not exist in bonus table (i.e. who did not get bonus)

8. SQL query to show the to pn(say10) records of a table.

9. Find people who have the same salary

10. SQL query to fetch the first 50% records from a table.

11. Find the highest 2 salaries without LIMIT or TOP.

12. Create a trigger to ensure that no employee of age less than 18 can be inserted in the database.

13. Create a trigger which will work before deletion in the employee table and create a duplicate copy of the record in another table employee\_backup.

14. Create a trigger to count the number of new tuples inserted using each insert statement.

## Source Code:

**Database Creation:**

CREATE DATABASE IF NOT EXISTS Organization;

USE Organization;

**Create the Employee table**

CREATE TABLE IF NOT EXISTS Employee (

EMP\_ID INT PRIMARY KEY,

FIRST\_NAME VARCHAR(50),

LAST\_NAME VARCHAR(50),

SALARY DECIMAL(10, 2),

JOINING\_DATE DATE,

DEPARTMENT VARCHAR(50)

);

**Create the Bonus table**

CREATE TABLE IF NOT EXISTS Bonus (

EMP\_REF\_ID INT,

BONUS\_AMOUNT DECIMAL(10, 2),

BONUS\_DATE DATE,

FOREIGN KEY (EMP\_REF\_ID) REFERENCES Employee(EMP\_ID)

);

**Create the Title table**

CREATE TABLE IF NOT EXISTS Title (

EMP\_REF\_ID INT,

EMP\_TITLE VARCHAR(50),

AFFECTED\_FROM DATE,

FOREIGN KEY (EMP\_REF\_ID) REFERENCES Employee(EMP\_ID)

);

**Task 1:**

SELECT \* FROM Employee ORDER BY FIRST\_NAME ASC, DEPARTMENT DESC;

**Task 2:**

SELECT COUNT(\*) FROM Employee WHERE DEPARTMENT = 'Admin';

**Task 3:**

SELECT FIRST\_NAME, LAST\_NAME FROM Employee WHERE SALARY BETWEEN 50000 AND 100000;

**Task 4:**

SELECT Employee.\* FROM Employee INNER JOIN Title ON Employee.EMP\_ID = Title.EMP\_REF\_ID WHERE Title.EMP\_TITLE = 'Manager';

**Task 5:**

SELECT SALARY, DEPARTMENT, COUNT(\*) FROM Employee GROUP BY SALARY, DEPARTMENT HAVING COUNT(\*) > 1;

**Task 6:**

WITH RankedRows AS (SELECT \*, ROW\_NUMBER() OVER (ORDER BY (SELECT NULL)) AS RowNum FROM Employee) SELECT \* FROM RankedRows WHERE RowNum % 2 = 0;

**Task 7:**

SELECT Employee.\* FROM Employee LEFT JOIN Bonus ON Employee.EMP\_ID = Bonus.EMP\_REF\_ID WHERE Bonus.EMP\_REF\_ID IS NULL;

**Task 8:**

SELECT \* FROM Employee LIMIT 10;

**Task 9:**

SELECT SALARY, COUNT(\*) FROM Employee GROUP BY SALARY HAVING COUNT(\*) > 1;

**Task 10:**

WITH RankedRows AS (SELECT \*, ROW\_NUMBER() OVER (ORDER BY (SELECT NULL)) AS RowNum FROM Employee) SELECT \* FROM RankedRows WHERE RowNum <= (SELECT COUNT(\*)/2 FROM Employee);

**Task 11:**

WITH RankedRows AS (SELECT \*, ROW\_NUMBER() OVER (ORDER BY Salary DESC) AS RowNum FROM Employee) SELECT \* FROM RankedRows WHERE RowNum <= 2;

**Task 12:**

DELIMITER //

CREATE TRIGGER age\_insert\_employee

BEFORE INSERT ON Employee

FOR EACH ROW

BEGIN

DECLARE emp\_age INT;

SET emp\_age = YEAR(CURDATE()) - YEAR(NEW.JOINING\_DATE) - (DATE\_FORMAT(CURDATE(), '%m%d') < DATE\_FORMAT(NEW.JOINING\_DATE, '%m%d'));

IF emp\_age < 18 THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Cannot insert employee with age less than 18.';

END IF;

END;

//

DELIMITER ;

**Task 13:**

CREATE TABLE IF NOT EXISTS Employee\_backup (

EMP\_ID INT PRIMARY KEY,

FIRST\_NAME VARCHAR(50),

LAST\_NAME VARCHAR(50),

SALARY DECIMAL(10, 2),

JOINING\_DATE DATE,

DEPARTMENT VARCHAR(50)

);

DELIMITER //

CREATE TRIGGER before\_delete\_employee BEFORE DELETE ON Employee FOR EACH ROW

BEGIN

INSERT INTO employee\_backup (EMP\_ID, FIRST\_NAME, LAST\_NAME, SALARY, JOINING\_DATE, DEPARTMENT) VALUES (OLD.EMP\_ID, OLD.FIRST\_NAME, OLD.LAST\_NAME, OLD.SALARY, OLD.JOINING\_DATE, OLD.DEPARTMENT);

END;

//

DELIMITER ;

**Task 14:**

CREATE TABLE insert\_count (

table\_name VARCHAR(255) PRIMARY KEY,

insert\_count INT DEFAULT 0

);

DELIMITER //

CREATE TRIGGER after\_insert\_count\_employee

AFTER INSERT ON Employee

FOR EACH ROW

BEGIN

INSERT INTO insert\_count (table\_name, insert\_count) VALUES ('Employee', 1) ON DUPLICATE KEY UPDATE insert\_count = insert\_count + 1;

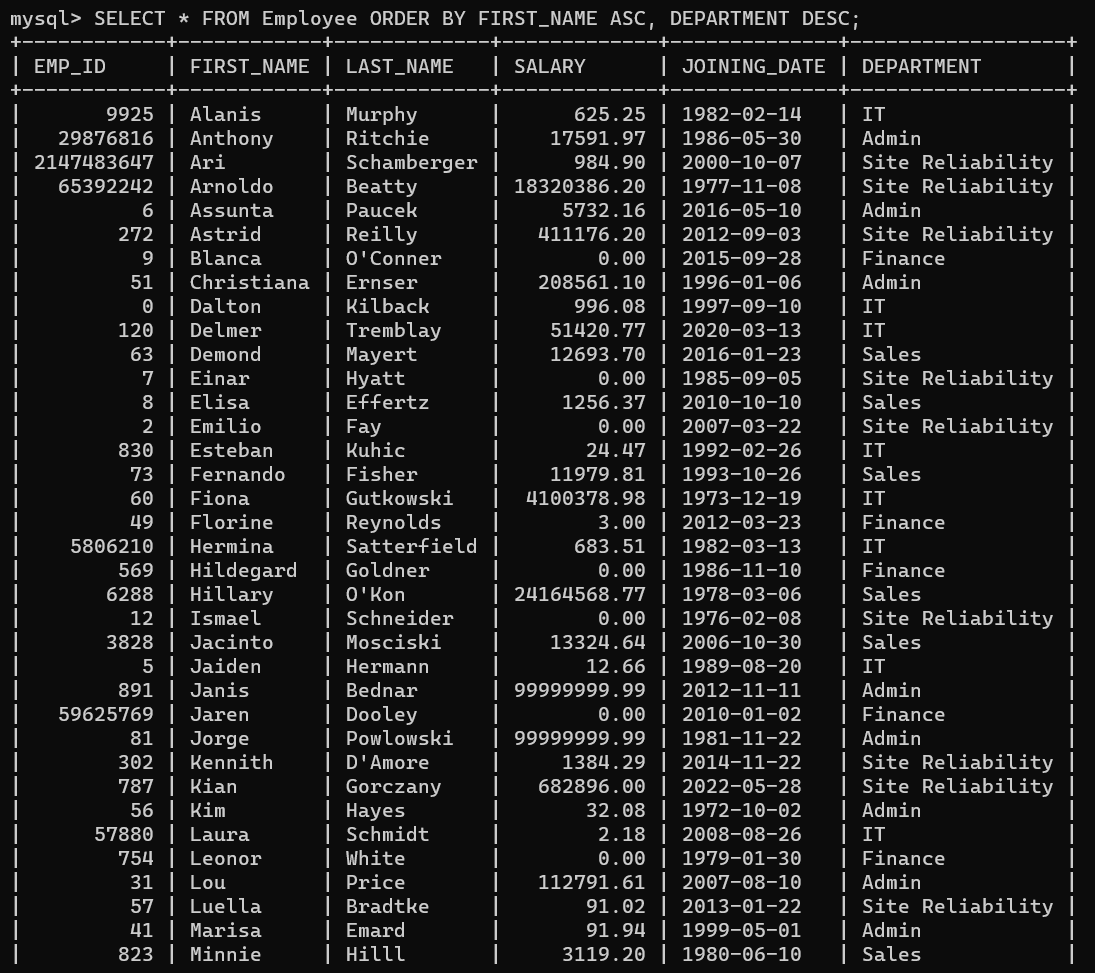
END;

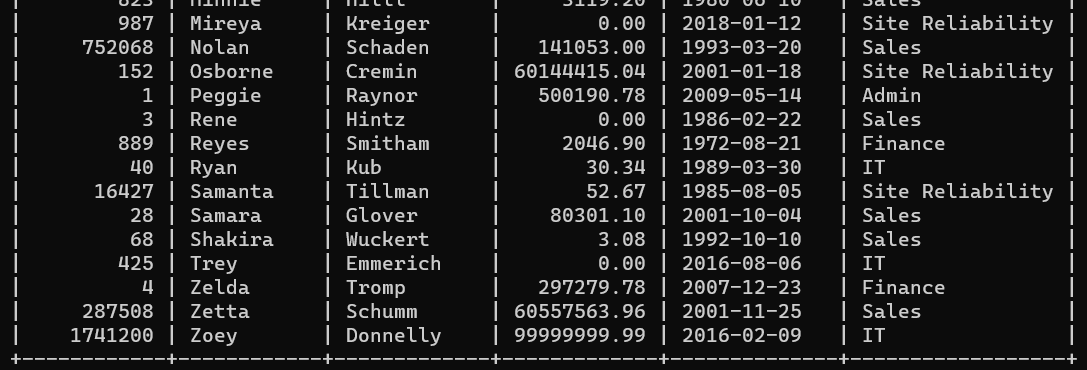
//

DELIMITER ;

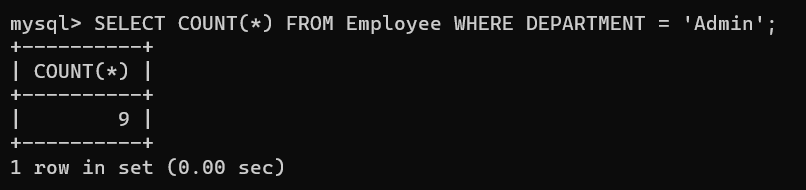
## Output:

**Task 1:**

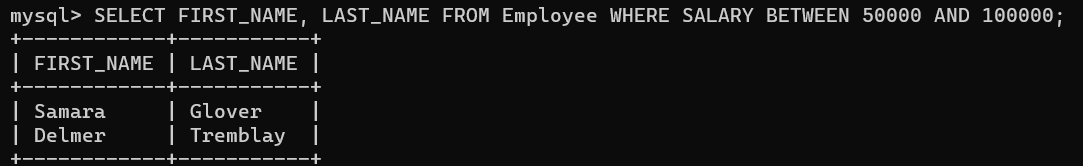
****

****

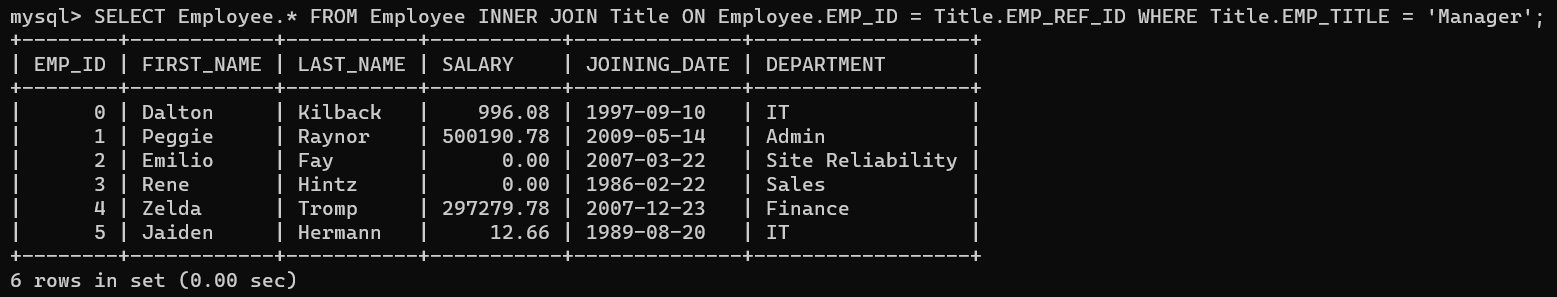
**Task 2:**

****

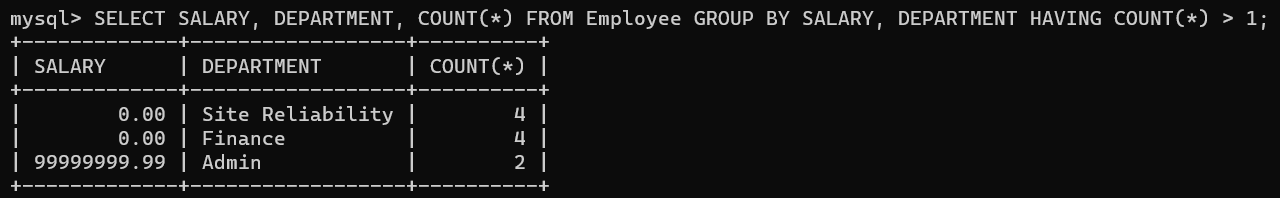
**Task 3:**

****

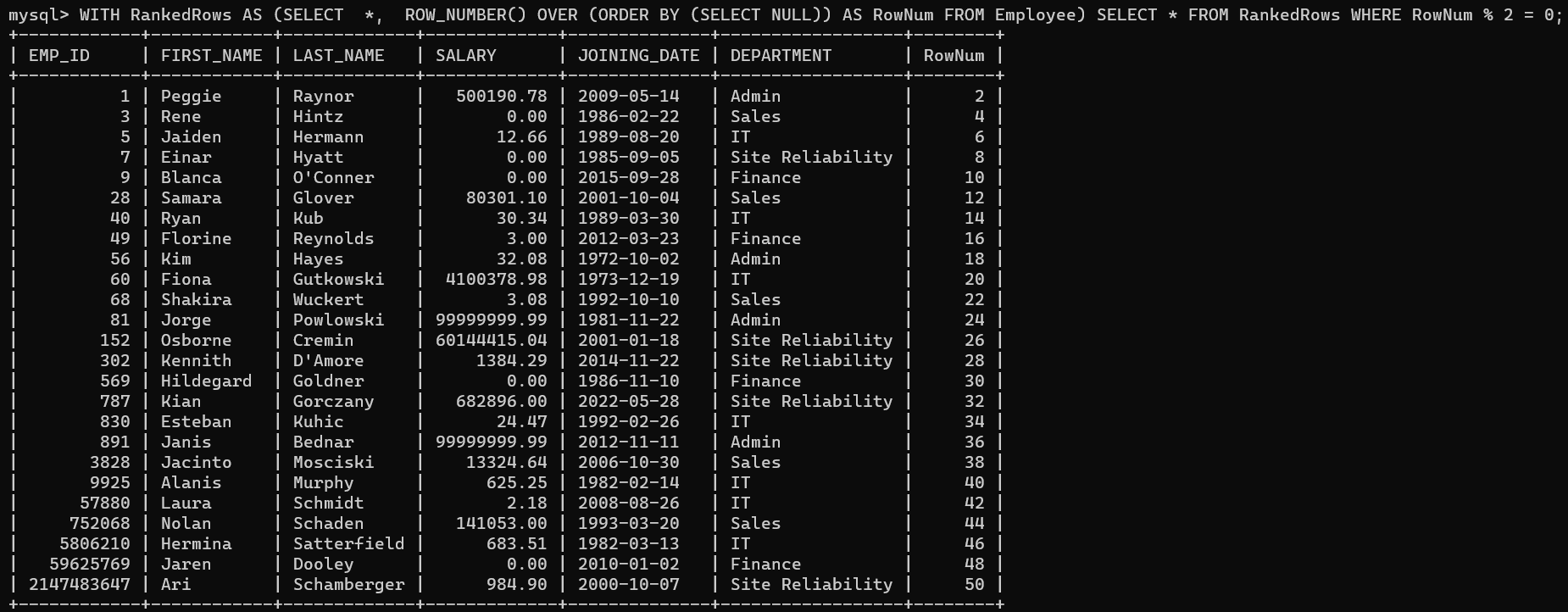
**Task 4:**

****

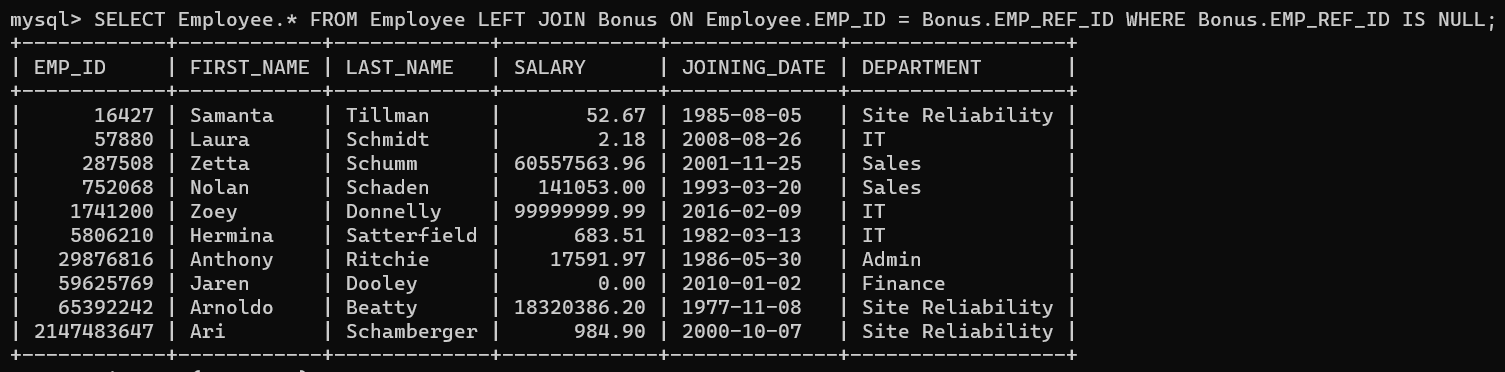
**Task 5:**

****

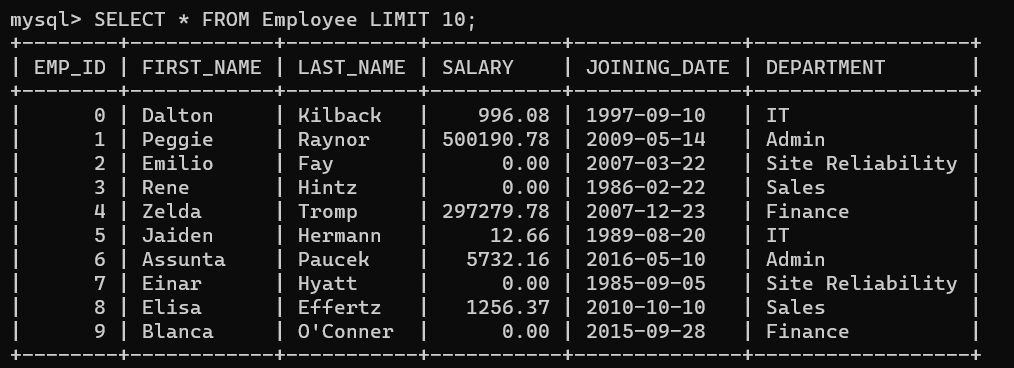
**Task 6:**

****

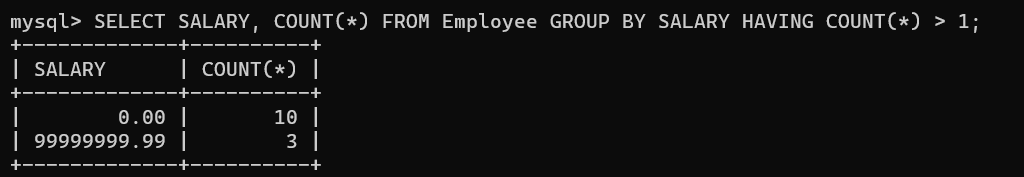
**Task 7:**

****

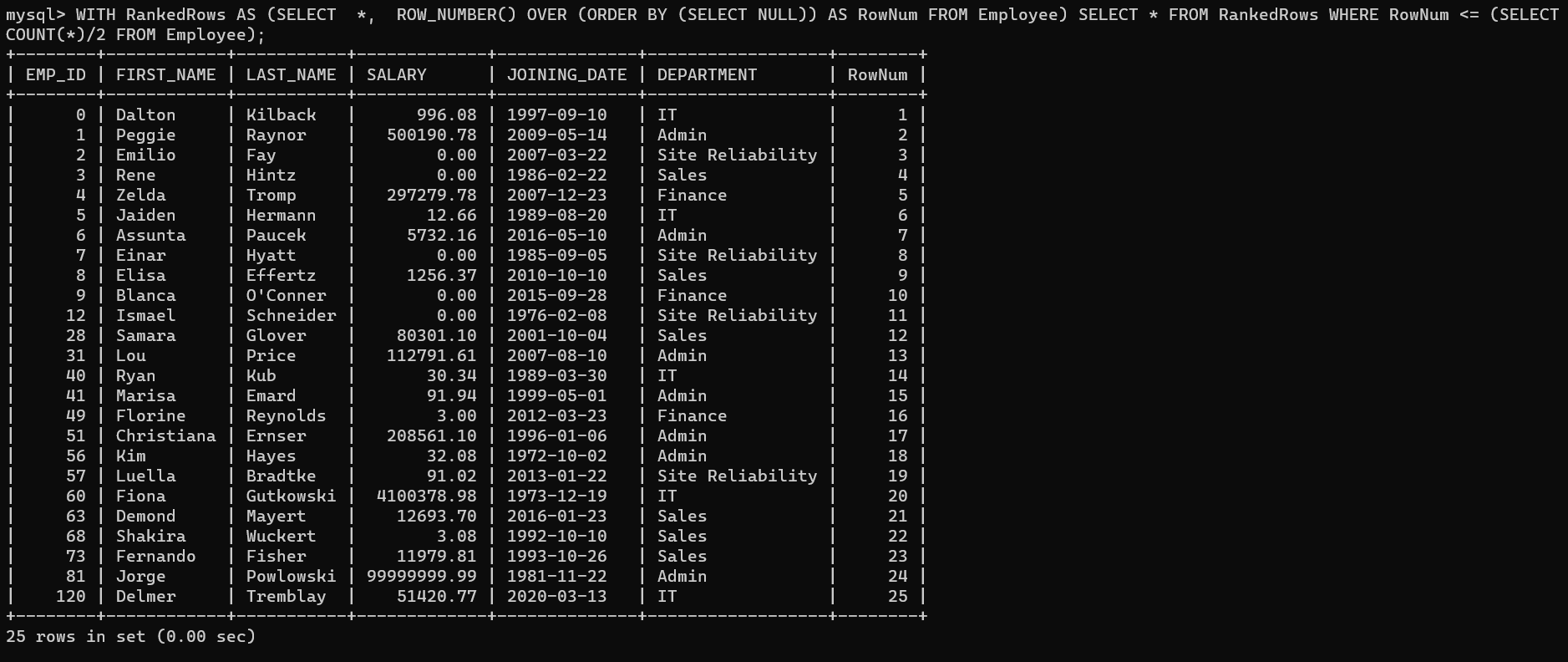
**Task 8:**

****

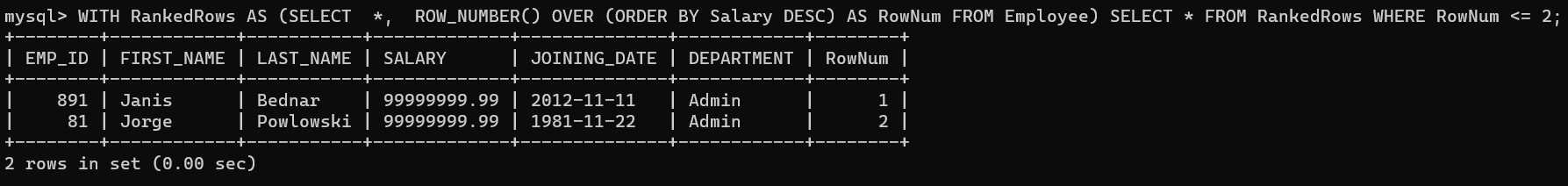
**Task 9:**

****

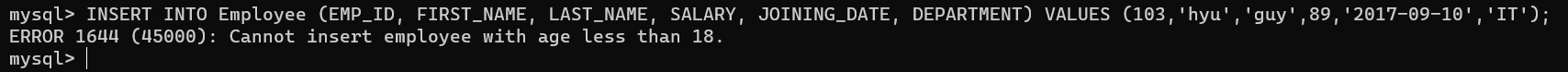
**Task 10:**

****

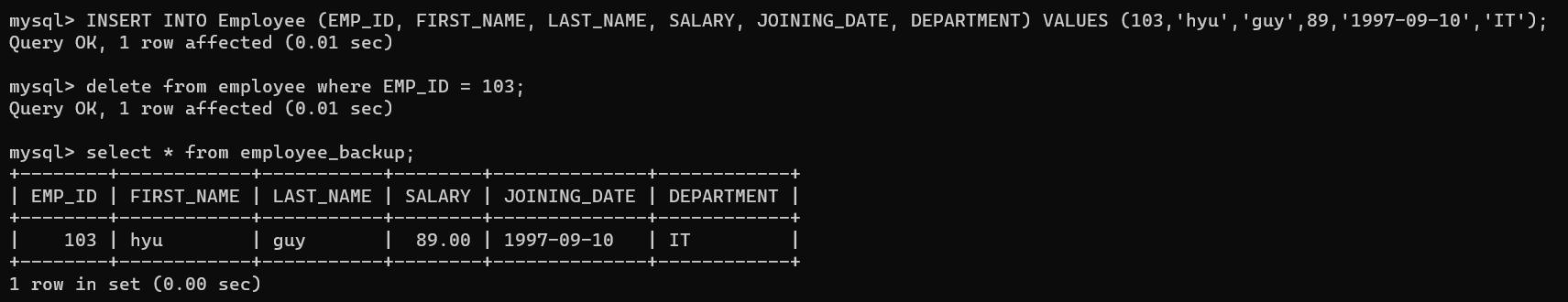
**Task 11:**

****

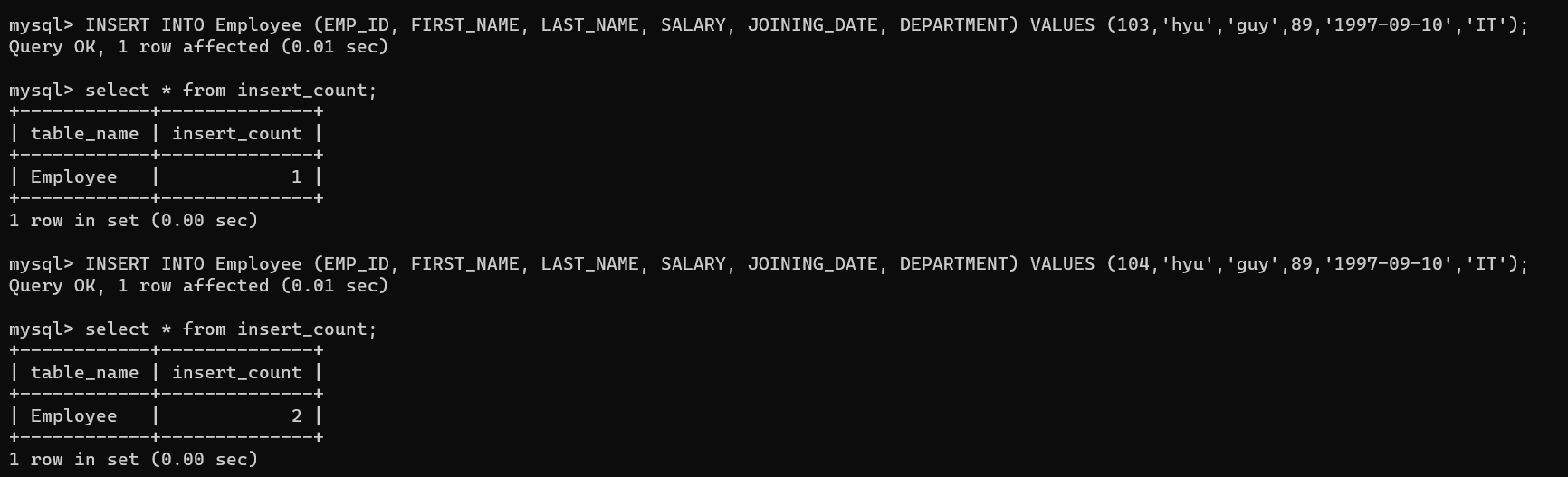
**Task 12:**

****

**Task 13:**

****

**Task 14:**

****

## Conclusion:

* Triggers are powerful mechanisms in SQL that can be used to enforce data integrity, automate tasks, and maintain historical records.
* Triggers enhance the reliability and security of the database by enforcing rules and executing actions automatically in response to specific events.
* Using basic sql statements to solve complex queries.