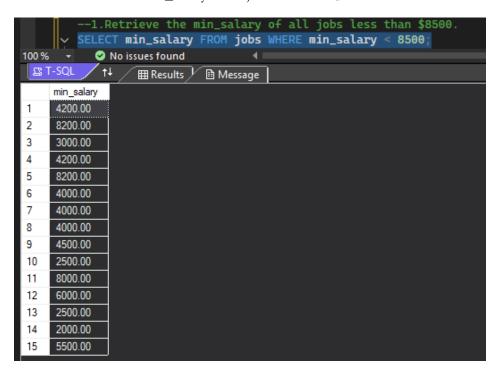
COSC 4301 - Database Theory and Practice

Homework 03

Construct the HR database and insert data using HR.sql.

Develop the following queries and screenshot demonstrating the query:

1. Retrieve the min_salary of all jobs less than \$8500.



2. Increase the min_salary by \$1000 of all jobs for min_salary that are \$1500 - \$8500.

```
--2.Increase the min_salary by $1000 of all jobs for min_salary that are $1500 - $8500.

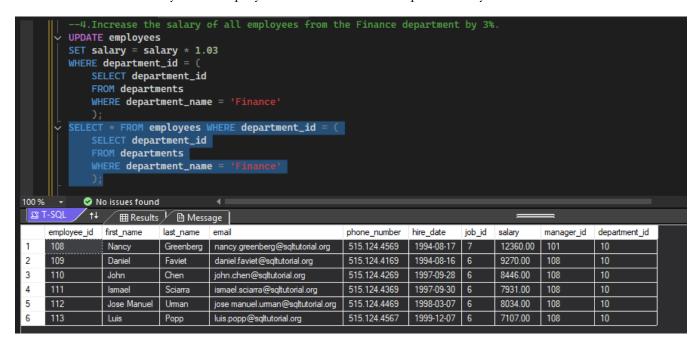
UPDATE jobs

SET min_salary = min_salary + 1000 WHERE min_salary >= 1500 and min_salary <= 8500;
```

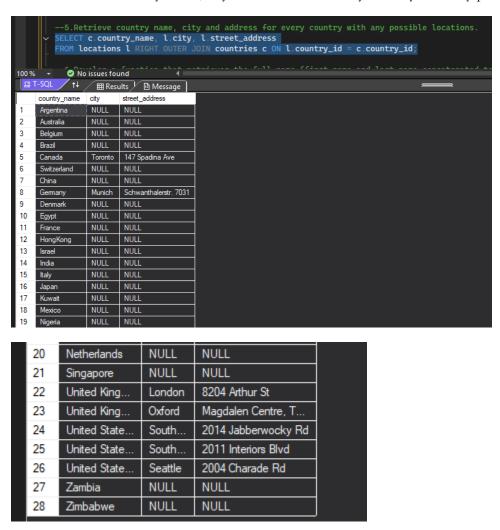
3. Retrieve the salary of all employees from any department that starts with an 'F'.

```
--3.Retrieve the salary of all employees from any department that starts with an 'F'.
        SELECT d.department_name, e.salary FROM employees e
        JOIN departments d ON e.department_id = d.department_id
        WHERE d.department_name like 'F%';
100 % 🔻 🥝 No issues found
🖺 T-SQL 🖊 🕇 🖽 Results 🖊 🗈 Message
    department_name
                   salary
                   12000.00
    Finance
                   9000.00
     Finance
                   8200.00
                   7700.00
     Finance
                   7800.00
     Finance
                   6900.00
```

4. Increase the salary of all employees from the Finance department by 3%.



5. Retrieve country name, city and address for every country with any possible locations.



6. Develop a function that retrieves the full name (first name and last name concatenated together) of the employee with a given ID.

```
--6.Develop a function that retrieves the full name (first name and last name concatenated together) of the employee with a --given ID.

gol

CREATE FUNCTION dbo.concatName (@id INT)

RETURNS VARCHAR(45)

AS

BEGIN

DECLARE @name VARCHAR(45);

SELECT @name = first_name + ' ' + last_name

FROM employees

WHERE employee_id = @id

RETURN @name;

END;

go

SELECT dbo.concatName(100) AS 'Full Name';

| In: 33 C
```

7. Develop a trigger to raise an error if a negative salary is attempted in an update of an employee.

```
--7.Develop a trigger to raise an error if a negative salary is attempted in an update of an employee.

yo

CREATE TRIGGER negativeSalary

ON employees

AFTER UPDATE

AS

BEGIN

IF EXISTS (

SELECT 1 FROM inserted WHERE salary < 0)

BEGIN

RAISERROR('Error: negative salary', 16, 1)

ROLLBACK TRANSACTION;

END

END;

go

UPDATE employees SET salary = -10000 WHERE employee_id = 100;

TOW

ST-SOL

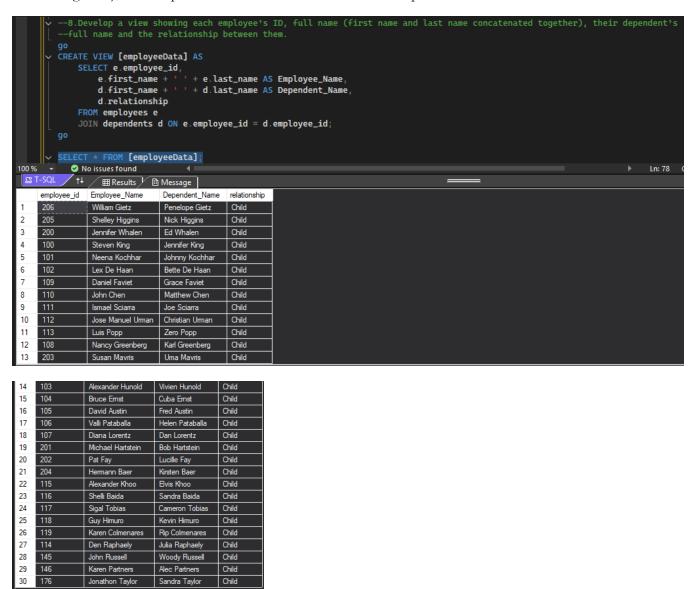
Msg 50000, Level 16, State 1, Procedure negativeSalary, Line 71

Error: negative salary

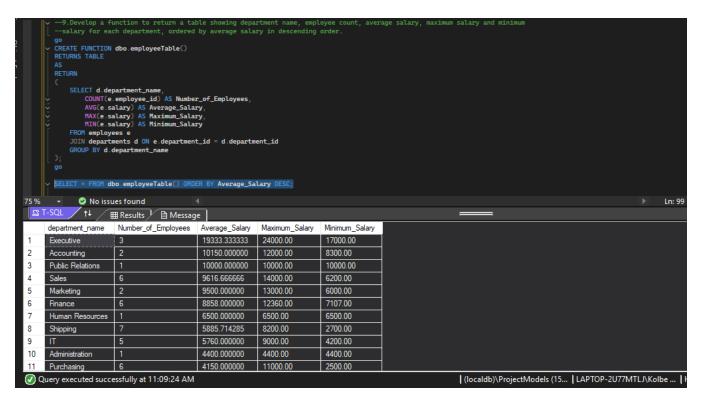
Msg 3609, Level 16, State 1, Line 64

The transaction ended in the trigger. The batch has been aborted.
```

8. Develop a view showing each employee's ID, full name (first name and last name concatenated together), their dependent's full name and the relationship between them.



9. Develop a function to return a table showing department name, employee count, average salary, maximum salary and minimum salary for each department, ordered by average salary in descending order.



10. Develop a stored procedure that adjusts the salary of a given employee within a designated department by a provided percentage increase.

```
-10.Develop a stored procedure that adjusts the salary of a given employee within a designated department by a provided -percentage increase.
     go
CREATE PROCEDURE AdjustSalary
           @employee_id IN
           Adepartment id INT
          epercentage DECIMAL(5,2)
          --Check if employee exists

IF EXISTS (SELECT 1 FROM employees MMERE @employee_id = employees.employee_id AND @department_id = employees.department_id)
               DECLARE @current_salary DECIMAL(8,2)
DECLARE @new_salary DECIMAL(8,2)
               SELECT @current_salary = employees salary
               FROM employees
               WHERE employees employee_id = @employee_id AND employees.department_id = @department_id;
               --Get new salary
SET @new_salary = @current_salary * (1 + @percentage);
               ---Update salary
UPDATE employees
               SET employees salary = @new_salary
      EXEC AdjustSalary 100, 9, 0.10;
                                             employee_id = 100 and department_id = 9)
              No issues found
雪 T-SQL
                     salary
     24000.00
```

```
CREATE PROCEDURE AdjustSalary
         @employee_id INT
         @department_id INT
         @percentage DECIMAL(5,2)
     BEGIN
         --Check if employee exists

IF EXISTS (SELECT 1 FROM employees WHERE @employee_id = employees.employee_id AND @department_id = employees.department_id)
             DECLARE @current_salary DECIMAL(8,2)
             DECLARE @new_salary DECIMAL(8,2)
              --Get current salary
             SELECT @current_salary = employees.salary
             FROM employees
             WHERE employees.employee_id = @employee_id AND employees.department_id = @department_id;
             SET @new_salary = @current_salary * (1 + @percentage);
             UPDATE employees
             SET employees.salary = @new_salary
     EXEC AdjustSalary 100, 9, 0.10;
     SELECT salary FROM employees WHERE employee_id = 100 and department_id = 9;
             No issues found
虚 T-SQL
                      Ⅲ Results
                                    26400.00
```

Full Code:

--1.Retrieve the min_salary of all jobs less than \$8500.

SELECT min_salary FROM jobs WHERE min_salary < 8500;

--2.Increase the min_salary by \$1000 of all jobs for min_salary that are \$1500 - \$8500.

UPDATE jobs

SET min_salary = min_salary + 1000 WHERE min_salary >= 1500 and min_salary <= 8500;

--3. Retrieve the salary of all employees from any department that starts with an 'F'.

SELECT d.department_name, e.salary FROM employees e

JOIN departments d ON e.department_id = d.department_id

WHERE d.department_name like 'F%';

--4. Increase the salary of all employees from the Finance department by 3%.

```
UPDATE employees
SET salary = salary * 1.03
WHERE department_id = (
      SELECT department_id
      FROM departments
      WHERE department_name = 'Finance'
      );
SELECT * FROM employees WHERE department_id = (
      SELECT department_id
      FROM departments
      WHERE department_name = 'Finance'
      );
--5. Retrieve country name, city and address for every country with any possible locations.
SELECT c.country_name, l.city, l.street_address
FROM locations l RIGHT OUTER JOIN countries c ON l.country_id = c.country_id;
--6.Develop a function that retrieves the full name (first name and last name concatenated together) of the
employee with a
--given ID.
go
CREATE FUNCTION dbo.concatName (@id INT)
RETURNS VARCHAR(45)
AS
BEGIN
DECLARE @name VARCHAR(45);
      SELECT @name = first_name + ' ' + last_name
      FROM employees
```

```
WHERE employee_id = @id
       RETURN @name;
END;
go
SELECT dbo.concatName(100) AS 'Full Name';
--7. Develop a trigger to raise an error if a negative salary is attempted in an update of an employee.
go
CREATE TRIGGER negativeSalary
ON employees
AFTER UPDATE
AS
BEGIN
       IF EXISTS (
             SELECT 1 FROM inserted WHERE salary < 0)
       BEGIN
             RAISERROR('Error: negative salary', 16, 1)
             ROLLBACK TRANSACTION;
       END
END;
go
UPDATE employees SET salary = -10000 WHERE employee_id = 100;
--8.Develop a view showing each employee's ID, full name (first name and last name concatenated
together), their dependent's
--full name and the relationship between them.
```

```
CREATE VIEW [employeeData] AS
      SELECT e.employee_id,
             e.first_name + ' ' + e.last_name AS Employee_Name,
             d.first_name + ' ' + d.last_name AS Dependent_Name,
             d.relationship
       FROM employees e
      JOIN dependents d ON e.employee_id = d.employee_id;
go
SELECT * FROM [employeeData];
--9. Develop a function to return a table showing department name, employee count, average salary,
maximum salary and minimum
--salary for each department, ordered by average salary in descending order.
go
CREATE FUNCTION dbo.employeeTable()
RETURNS TABLE
AS
RETURN
(
      SELECT d.department_name,
             COUNT(e.employee_id) AS Number_of_Employees,
             AVG(e.salary) AS Average_Salary,
             MAX(e.salary) AS Maximum_Salary,
             MIN(e.salary) AS Minimum_Salary
      FROM employees e
      JOIN departments d ON e.department_id = d.department_id
```

```
GROUP BY d.department_name
);
go
SELECT * FROM dbo.employeeTable() ORDER BY Average_Salary DESC;
--10. Develop a stored procedure that adjusts the salary of a given employee within a designated department
by a provided
--percentage increase.
go
CREATE PROCEDURE AdjustSalary
      @employee_id INT,
      @department_id INT,
      @percentage DECIMAL(5,2)
AS
BEGIN
      -- Check if employee exists
      IF EXISTS (SELECT 1 FROM employees WHERE @employee_id = employees.employee_id
AND @department_id = employees.department_id)
      BEGIN
             DECLARE @current_salary DECIMAL(8,2)
             DECLARE @new_salary DECIMAL(8,2)
             --Get current salary
             SELECT @current_salary = employees.salary
             FROM employees
             WHERE employees.employee_id = @employee_id AND employees.department_id =
@department_id;
```

```
--Get new salary

SET @new_salary = @current_salary * (1 + @percentage);

--Update salary

UPDATE employees

SET employees.salary = @new_salary

PRINT('Salary Adjusted');

END

ELSE

PRINT('Employee does not exist in specified department')

END;

go

EXEC AdjustSalary 100, 9, 0.10;

SELECT salary FROM employees WHERE employee_id = 100 and department_id = 9;
```