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.

```sql
--1.Retrieve the min_salary of all jobs less than $8500.
SELECT min_salary FROM jobs WHERE min_salary < 8500;
```

100 % No issues found

T-SQL | Results | Message

	min_salary
1	4200.00
2	8200.00
3	3000.00
4	4200.00
5	8200.00
6	4000.00
7	4000.00
8	4000.00
9	4500.00
10	2500.00
11	8000.00
12	6000.00
13	2500.00
14	2000.00
15	5500.00

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

```sql
--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'.

```sql
--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
1	Finance	12000.00
2	Finance	9000.00
3	Finance	8200.00
4	Finance	7700.00
5	Finance	7800.00
6	Finance	6900.00

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

```sql
--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'
    );
```

100 % No issues found

T-SQL | Results | Message

	employee_id	first_name	last_name	email	phone_number	hire_date	job_id	salary	manager_id	department_id
1	108	Nancy	Greenberg	nancy.greenberg@sqltutorial.org	515.124.4569	1994-08-17	7	12360.00	101	10
2	109	Daniel	Faviet	daniel.faviet@sqltutorial.org	515.124.4169	1994-08-16	6	9270.00	108	10
3	110	John	Chen	john.chen@sqltutorial.org	515.124.4269	1997-09-28	6	8446.00	108	10
4	111	Ismael	Sciarra	ismael.sciarra@sqltutorial.org	515.124.4369	1997-09-30	6	7931.00	108	10
5	112	Jose Manuel	Urman	jose manuel.urman@sqltutorial.org	515.124.4469	1998-03-07	6	8034.00	108	10
6	113	Luis	Popp	luis.popp@sqltutorial.org	515.124.4567	1999-12-07	6	7107.00	108	10

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

```sql
--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;
```

100 % No issues found

T-SQL | Results | Message

	country_name	city	street_address
1	Argentina	NULL	NULL
2	Australia	NULL	NULL
3	Belgium	NULL	NULL
4	Brazil	NULL	NULL
5	Canada	Toronto	147 Spadina Ave
6	Switzerland	NULL	NULL
7	China	NULL	NULL
8	Germany	Munich	Schwanthalerstr. 7031
9	Denmark	NULL	NULL
10	Egypt	NULL	NULL
11	France	NULL	NULL
12	HongKong	NULL	NULL
13	Israel	NULL	NULL
14	India	NULL	NULL
15	Italy	NULL	NULL
16	Japan	NULL	NULL
17	Kuwait	NULL	NULL
18	Mexico	NULL	NULL
19	Nigeria	NULL	NULL
20	Netherlands	NULL	NULL
21	Singapore	NULL	NULL
22	United King...	London	8204 Arthur St
23	United King...	Oxford	Magdalen Centre, T...
24	United State...	South...	2014 Jabberwocky Rd
25	United State...	South...	2011 Interiors Blvd
26	United State...	Seattle	2004 Charade Rd
27	Zambia	NULL	NULL
28	Zimbabwe	NULL	NULL

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

```sql
--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';
```

100 % 1 0 Ln: 33

T-SQL Results Message

	Full Name
1	Steven King

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

```sql
--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;
```

100 % 1 0

T-SQL Results Message

```
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.

```sql
--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.
go
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];
```

100 % No issues found Ln: 78

T-SQL Results Message

	employee_id	Employee_Name	Dependent_Name	relationship
1	206	William Gietz	Penelope Gietz	Child
2	205	Shelley Higgins	Nick Higgins	Child
3	200	Jennifer Whalen	Ed Whalen	Child
4	100	Steven King	Jennifer King	Child
5	101	Neena Kochhar	Johnny Kochhar	Child
6	102	Lex De Haan	Bette De Haan	Child
7	109	Daniel Faviet	Grace Faviet	Child
8	110	John Chen	Matthew Chen	Child
9	111	Ismael Sciarra	Joe Sciarra	Child
10	112	Jose Manuel Urman	Christian Urman	Child
11	113	Luis Popp	Zero Popp	Child
12	108	Nancy Greenberg	Karl Greenberg	Child
13	203	Susan Mavris	Uma Mavris	Child
14	103	Alexander Hunold	Vivien Hunold	Child
15	104	Bruce Ernst	Cuba Ernst	Child
16	105	David Austin	Fred Austin	Child
17	106	Valli Pataballa	Helen Pataballa	Child
18	107	Diana Lorentz	Dan Lorentz	Child
19	201	Michael Hartstein	Bob Hartstein	Child
20	202	Pat Fay	Lucille Fay	Child
21	204	Hermann Baer	Kirsten Baer	Child
22	115	Alexander Khoo	Elvis Khoo	Child
23	116	Shelli Baida	Sandra Baida	Child
24	117	Sigal Tobias	Cameron Tobias	Child
25	118	Guy Himuro	Kevin Himuro	Child
26	119	Karen Colmenares	Rip Colmenares	Child
27	114	Den Raphaely	Julia Raphaely	Child
28	145	John Russell	Woody Russell	Child
29	146	Karen Partners	Alec Partners	Child
30	176	Jonathon Taylor	Sandra Taylor	Child

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.

```sql
--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;
```

75 % No issues found Ln: 99

T-SQL | Results | Message

	department_name	Number_of_Employees	Average_Salary	Maximum_Salary	Minimum_Salary
1	Executive	3	19333.333333	24000.00	17000.00
2	Accounting	2	10150.000000	12000.00	8300.00
3	Public Relations	1	10000.000000	10000.00	10000.00
4	Sales	6	9616.666666	14000.00	6200.00
5	Marketing	2	9500.000000	13000.00	6000.00
6	Finance	6	8858.000000	12360.00	7107.00
7	Human Resources	1	6500.000000	6500.00	6500.00
8	Shipping	7	5885.714285	8200.00	2700.00
9	IT	5	5760.000000	9000.00	4200.00
10	Administration	1	4400.000000	4400.00	4400.00
11	Purchasing	6	4150.000000	11000.00	2500.00

Query executed successfully at 11:09:24 AM | (localdb)\ProjectModels (15... | LAPTOP-2U77MTLJ\Kolbe ...

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

```sql
--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;
```

70 % No issues found

T-SQL | Results | Message

	salary
1	24000.00

```sql
--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;
```

70 % No issues found

T-SQL | Results | Message

	salary
1	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.

```
go
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;
```