CSCI 585 - Assignment 2 Rubrics

SQL Queries

Total Marks: 6 marks

Submission/Question	Marks Distribution
Q1.sql	2
Q2.sql	1
Q3.sql	1
Q4.sql	1
Q5.sql	1
Bonus of Q5	1

Total Marks: **6** marks (**6** points on SQL queries, **1** Bonus Point). If a student scores 7, please cap the total to 6.

Rubric for Graders:

1. **Late Submission: 10% per day (-0.6)** - [Assignment Due: Oct 08, 11:59 PM]
2. **Submission Format :** Should be as per submission checklist
    - The queries should be in .sql file format. **(-1 if any other format, -0.25 if .txt)**
    - Name of the IDE/server used by the student should be mentioned in the query file. **(-0.5)**
3. **Q1:** (**-2** total deduction cap)
    - Check if all creation queries of all **8** tables are present: (**-0.25** per missing table, **-1** deduction cap)
    - Check primary keys, foreign keys and whether attributes make sense. (**-0.1** per violation, **-0.5** deduction cap)
    - Has a few insertion queries. (**-0.2** if no insertion query at all**)**
4. **Q2:** (**-1** total deduction cap**)**
    - Should use the symptom table. Consider other approaches. Provide grades on how correct the query is.
5. **Q3:** (**-1** total deduction cap)
    - Sickest floor might have many possible interpretations by students. Consider all approaches. Deduction approach is the same as Q2.
6. **Q4:** (**-1** total deduction cap)
    - 4 separate queries or 1 query giving 4 stats as output - number of scans, number of tests, number of employees who self-reported symptoms, number of positive cases
    - Deduct **-0.25** for each wrong stat.
7. **Q5:** (**-1** total deduction cap and **+1** Bonus question)
    - Any query (however simple) is ok. Provide **bonus points (+1)** if the query involves **table division.** For Bonus: Should be mentioned in the Readme that they've used the division operator
    -

along with any assumptions made. **If mentioned in the readme, only then check for bonus points.**

- Table division usually follows the 'all' logic. (For example: Find all employees who tested positive in the given time period)

Sample Solution

Question 1:

Creations

```
-- Create Employee table

CREATE TABLE Employee (
    ID INT PRIMARY KEY,
    name VARCHAR(255),
    office_number VARCHAR(10),
    floor_number INT,
    phone_number VARCHAR(15),
    email_address VARCHAR(255)
);

-- Create Meeting table

CREATE TABLE Meeting (
    meeting_ID INT,
    employee_ID INT,
    room_number VARCHAR(10),
    floor_number INT,
    meeting_start_time INT,
    PRIMARY KEY (meeting_ID,employee_ID),
    FOREIGN KEY (employee_ID) REFERENCES Employee(ID)
);


-- Create Notification table

CREATE TABLE Notification (
    notification_ID INT PRIMARY KEY,
    employee_ID INT,
    notification_date DATE,
    notification_type VARCHAR(20),
    FOREIGN KEY (employee_ID) REFERENCES Employee(ID)
);


-- Create Symptom table
```

-

```
CREATE TABLE Symptom (
    row_ID INT PRIMARY KEY,
    employee_ID INT,
    date_reported DATE,
    symptom_ID INT,
    FOREIGN KEY (employee_ID) REFERENCES Employee(ID)
);

-- Create Scan table

CREATE TABLE Scan (
    scan_ID INT PRIMARY KEY,
    scan_date DATE,
    scan_time INT,
    employee_ID INT,
    temperature DECIMAL(4, 2),
    FOREIGN KEY (employee_ID) REFERENCES Employee(ID)
);

-- Create Test table

CREATE TABLE Test (
    test_ID INT PRIMARY KEY,
    location VARCHAR(255),
    test_date DATE,
    test_time INT,
    employee_ID INT,
    test_result VARCHAR(10),
    FOREIGN KEY (employee_ID) REFERENCES Employee(ID)
);

-- Create Case table

CREATE TABLE Cases (
    case_ID INT PRIMARY KEY,
    employee_ID INT,
    date_ DATE,
    resolution VARCHAR(20),
    FOREIGN KEY (employee_ID) REFERENCES Employee(ID)
);

-- Create HealthStatus table

CREATE TABLE HealthStatus (
    row_ID INT PRIMARY KEY,
    employee_ID INT,
    date_ DATE,
    status VARCHAR(20),
```

- o

```
    FOREIGN KEY (employee_ID) REFERENCES Employee(ID)
);
```

Insertions:

```
INSERT INTO Employee (ID, name, office_number, floor_number, phone_number,
email_address)
VALUES
    (1, 'John Doe', '101', 1, '123-456-7890', 'john@gmail.com'),
    (2, 'Jane Smith', '202', 2, '987-654-3210', 'jane@usc.edu'),
    (3, 'Alice Johnson', '303', 3, '555-555-5555', 'alice@usc.edu'),
    (4, 'Bob Anderson', '404', 4, '444-444-4444', 'bob@hotmail.com'),
    (5, 'Eve Williams', '505', 5, '333-333-3333', 'eve@hotmail.com'),
    (6, 'Charlie Brown', '606', 6, '222-222-2222', 'charlie@gmail.com'),
    (7, 'Grace Davis', '707', 7, '111-111-1111', 'grace@gmail.com'),
    (8, 'Daniel Lee', 'H808', 8, '999-999-9999', 'daniel@gmail.com'),
    (9, 'Olivia Perez', '909', 9, '888-888-8888', 'olivia@gmail.com'),
    (10, 'Sophia Hall', '709', 7, '777-777-7777', 'sophia@hotmail.com'),
    (11, 'Liam Martinez', '407', 4, '666-666-6666', 'liam@gmail.com'),
    (12, 'Emma Young', '201', 2, '555-555-5555', 'emma@usc.edu'),
    (13, 'Noah Taylor', '310', 3, '444-444-4444', 'noah@usc.edu'),
    (14, 'Ava Johnson', '408', 4, '333-333-3333', 'ava@rediffmail.com'),
    (15, 'William Smith', '512', 5, '222-222-2222', 'william@gmail.com');

INSERT INTO Meeting (meeting_ID, employee_ID, room_number, floor_number,
meeting_start_time)
VALUES
    (1, 1, '101', 1, 10),
    (2, 2, '202', 2, 14),
    (3, 3, '303', 3, 11),
    (4, 4, '404', 4, 15),
    (5, 5, '505', 5, 9),
    (6, 6, '606', 6, 14),
    (7, 7, '707', 7, 10),
    (8, 8, '808', 8, 13),
    (9, 9, '909', 9, 12),
    (10, 10, '109', 1, 16),
    (11, 11, '105', 1, 10),
    (12, 12, '208', 2, 14),
    (13, 13, '304', 3, 15),
    (14, 14, '419', 4, 11),
    (15, 15, '522', 5, 12);

INSERT INTO Notification (notification_ID, employee_ID, notification_date,
notification_type)
VALUES
    (1, 1, '2023-09-15', 'mandatory'),
    (2, 2, '2023-09-16', 'optional'),
```

- o

```
    (3, 3, '2023-09-17', 'mandatory'),
    (4, 4, '2023-09-18', 'optional'),
    (5, 5, '2023-09-19', 'mandatory'),
    (6, 6, '2023-09-20', 'mandatory'),
    (7, 7, '2023-09-21', 'optional'),
    (8, 8, '2023-09-22', 'mandatory'),
    (9, 9, '2023-09-23', 'optional'),
    (10, 10, '2023-09-24', 'mandatory'),
    (11, 11, '2023-09-25', 'mandatory'),
    (12, 12, '2023-09-26', 'optional');

INSERT INTO Symptom (row_ID, employee_ID, date_reported, symptom_ID)
VALUES
    (1, 1, '2023-09-10', 5),
    (2, 2, '2023-09-12', 5),
    (3, 3, '2023-09-13', 1),
    (4, 4, '2023-09-14', 2),
    (5, 5, '2023-09-15', 4),
    (6, 6, '2023-09-16', 3),
    (7, 7, '2023-09-17', 5),
    (8, 8, '2023-09-18', 1),
    (9, 9, '2023-09-19', 4),
    (10, 10, '2023-09-20', 2);

INSERT INTO Scan (scan_ID, scan_date, scan_time, employee_ID, temperature)
VALUES
    (1, '2023-09-10', 9, 1, 98.6),
    (2, '2023-09-12', 15, 2, 99.2),
    (3, '2023-09-13', 10, 3, 98.9),
    (4, '2023-09-14', 12, 4, 99.1),
    (5, '2023-09-15', 13, 5, 98.7),
    (6, '2023-09-16', 11, 6, 99.3),
    (7, '2023-09-17', 9, 7, 98.5),
    (8, '2023-09-18', 14, 8, 99.0),
    (9, '2023-09-19', 16, 9, 98.8),
    (10, '2023-09-20', 10, 10, 101.5);

INSERT INTO Test (test_ID, location, test_date, test_time, employee_ID,
test_result)
VALUES
    (1, 'Company', '2023-09-05', 13, 1, 'negative'),
    (2, 'Hospital', '2023-09-08', 11, 2, 'positive'),
    (3, 'Company', '2023-09-09', 13, 3, 'negative'),
    (4, 'Hospital', '2023-09-10', 16, 4, 'positive'),
    (5, 'Clinic', '2023-09-11', 11, 5, 'negative'),
    (6, 'Company', '2023-09-12', 14, 6, 'positive'),
    (7, 'Hospital', '2023-09-13', 15, 7, 'positive'),
    (8, 'Clinic', '2023-09-14', 12, 8, 'negative');
```

- o

```
INSERT INTO cases (case_ID, employee_ID, date, resolution)
VALUES
    (1, 2, '2023-09-08', 'hospitalized'),
    (2, 3, '2023-09-10', 'back to work'),
    (3, 4, '2023-09-11', 'back to work'),
    (4, 5, '2023-09-12', 'hospitalized'),
    (5, 6, '2023-09-13', 'back to work');

INSERT INTO HealthStatus (row_ID, employee_ID, date, status)
VALUES
    (1, 1, '2023-09-10', 'sick'),
    (2, 2, '2023-09-12', 'well'),
    (3, 3, '2023-09-14', 'well'),
    (4, 4, '2023-09-15', 'sick'),
    (5, 5, '2023-09-16', 'well'),
    (6, 11, '2023-09-17', 'sick'),
    (7, 7, '2023-09-18', 'well');
```

Question 2:

```
SELECT symptom_ID, COUNT(*) as symptom_count
FROM Symptom
GROUP BY symptom_ID
ORDER BY symptom_count DESC
LIMIT 1;
```

Question 3:

```
SELECT e.floor_number, COUNT(*) as sickness_count
FROM HealthStatus h
JOIN Employee e ON h.employee_ID = e.ID
WHERE h.status = 'sick'
GROUP BY e.floor_number
ORDER BY sickness_count DESC
LIMIT 1;
```

Question 4:

Number of scans:

```
SELECT COUNT(*) as scan_count
FROM Scan
WHERE scan_date BETWEEN '2023-09-10' AND '2023-09-15';
```

Number of tests:

```
SELECT COUNT(*) as test_count
FROM Test
```

-

```
WHERE test_date BETWEEN '2023-09-05' AND '2023-09-10';
```

Number of self reported employees:

```
SELECT COUNT(DISTINCT employee_ID) as symptom_reporting_count
FROM Symptom
WHERE date_reported BETWEEN '2023-09-14' AND '2023-09-18';
```

Number of positive cases:

```
SELECT COUNT(*) as positive_case_count
FROM cases
WHERE date BETWEEN '2023-09-05' AND '2023-09-10';
```

Question 5:

Average Body Temperature:

```
SELECT AVG(s.temperature) as average_temperature
FROM Scan s
INNER JOIN Test t ON s.employee_ID = t.employee_ID
WHERE t.test_result = 'positive';
```

- o