SQL Assigment

A research team is working on innovative solutions to combat climate change. They are monitoring environmental conditions like temperature and humidity across various locations worldwide. By analyzing this data, they aim to identify patterns, predict extreme weather events, and develop strategies to mitigate climate change impacts. Your task is to build the database that supports their critical research.

1. DDL Create a database with the following 3 tables:
2. **Locations**

* location\_id (Primary Key, INT)
* city (VARCHAR(50), NOT NULL)
* country (VARCHAR(50), NOT NULL)

1. **Sensors**

* sensor\_id (Primary Key, INT)
* sensor\_type (VARCHAR(30), NOT NULL, e.g., Temperature, Humidity)
* location\_id (Foreign Key to Locations, INT, NOT NULL)

1. **Measurements**

* measurement\_id (Primary Key, INT)
* sensor\_id (Foreign Key to Sensors, INT, NOT NULL)
* temperature (NUMERIC(5, 2), CHECK (temperature BETWEEN -50 AND 50))
* humidity (NUMERIC(5, 2), CHECK (humidity BETWEEN 0 AND 100))
* timestamp (TIMESTAMP, NOT NULL), Default value – Currents timpestamp.

Constraints:

* Ensure no duplicate location\_id in **Locations**.
* Ensure no duplicate sensor\_id in **Sensors**.
* Foreign keys should enforce referential integrity.
* **Measurements** table should ensure:

temperature is valid between -50°C and 50°C.

humidity is valid between 0% and 100%.

1. DML Insert the following data into the tables:
2. Insert this data to the locations table

|  |  |  |
| --- | --- | --- |
| **ocation\_id** | **city** | **country** |
| 111 | New York | USA |
| 222 | Berlin | Germany |
| 333 | Kibuts Asisim | Israel |

1. run the following command to populate the sensors table

INSERT INTO Sensors (sensor\_id, sensor\_type, location\_id)

VALUES (1, 'Temperature', 111),

(2, 'Humidity', 111),

(3, 'Temperature', 222),

(4, 'Humidity', 222);

1. Test your foreign key: run the following command to populate the next 2 rows, if it doesn’t run- fix it so that the sensors would be assigned to Kibutz Asisim.

INSERT INTO Sensors (sensor\_id, sensor\_type, location\_id)

VALUES (5, 'Temperature', 3), (6, 'Humidity', 3);

1. populate the last table with the following command.

INSERT INTO Measurements (measurement\_id, sensor\_id, temperature, humidity)

VALUES

(1, 1, 12.5, 60.0),

(2, 2, 10.0, 65.0),

(3, 3, 10.5, 75.0),

(4, 4, 11.0, 80.0),

(5, 5, 24.0, 58.0),

(6, 6, 26.0, 60.0),

(7, 1, 11.5, 55.0),

(8, 2, 9.5, 62.0),

(9, 3, 9.0, 72.0),

(10, 4, 10.0, 75.0),

(11, 5, 23.0, 57.0),

(12, 6, 25.5, 59.0),

(13, 1, 13.0, 60.0),

(14, 2, 8.0, 66.0),

(15, 3, 8.5, 70.0);

Check your check constraint, try to insert the next row (this should fail)

INSERT INTO Measurements (measurement\_id, sensor\_id, temperature, humidity)

VALUES (42,5,NULL,157);

How can you fix it?

1. **Check your measurements table** using a SELECT statement, and order the rows by sensor\_id.  
   You will notice that all the rows for each sensor are showing roughly the same timestamp, which is incorrect.  
   Use the INTERVAL '1 month' to update all measurements with measurement\_id between 7 and 12, setting their timestamps to reflect **last month**.
2. **Update the measurements** with measurement\_id 13 and higher, setting their timestamps to reflect **two months ago**.
3. SELECT
4. How many measurements did each sensor record? Order your results from the sensor with the most measurements to the one with the least. If multiple sensors have the same number of measurements, order them by the sensor number in ascending order.
5. Create a report that shows the city name and the average temperature and humidity there.

--if you have time: make it a nice report. Round the numbers, give the month a name instead of a number, present humidity with the % sign.

1. Create a report that show the month average temperature and humidity in that month.
2. When and where was the temperature lower than the minimal temperature in Berlin