

WeatherApp SQL Script

Create a Database:

```
CREATE DATABASE WeatherApp;
```

Use the Created Database:

```
USE WeatherApp;
```

Create Table for Weather Data:

```
CREATE TABLE Weather (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    city_name VARCHAR(100),  
    temperature DECIMAL(5, 2),  
    humidity INT,  
    wind_speed DECIMAL(5, 2),  
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

Insert Sample Data into the Weather Table:

```
INSERT INTO Weather (city_name, temperature, humidity, wind_speed)  
VALUES  
    ('New York', 22.5, 60, 10.2),  
    ('Los Angeles', 25.3, 40, 8.7),  
    ('Chicago', 18.9, 55, 12.4);
```

Retrieve All Weather Data:

```
SELECT * FROM Weather;
```

Retrieve Weather for a Specific City (e.g., New York):

```
SELECT city_name, temperature, humidity, wind_speed  
FROM Weather  
WHERE city_name = 'New York';
```

Update Temperature for a Specific City (e.g., Los Angeles):

```
UPDATE Weather  
SET temperature = 23.5  
WHERE city_name = 'Los Angeles';
```

Delete Weather Data for a Specific City (e.g., Chicago):

```
DELETE FROM Weather  
WHERE city_name = 'Chicago';
```

Get Average Temperature of All Cities:

```
SELECT AVG(temperature) AS average_temperature FROM Weather;
```

The screenshot shows a database management tool interface. On the left, the 'SCHEMAS' pane displays a tree view with 'sys' and 'weatherapp' schemas. Under 'weatherapp', there is a 'weather' table with columns: id, city_name, temperature, humidity, wind_speed, and timestamp. The main query editor shows 'Query 1' with the following SQL:

```
1 • SELECT AVG(temperature) AS average_temperature FROM Weather;
2 |
```

The 'Result Grid' at the bottom shows a single row with the column 'average_temperature' and the value '23.000000'.

average_temperature
23.000000

Get Weather Data for the Last 7 Days (if timestamp exists):

```
SELECT * FROM Weather
WHERE timestamp > NOW() - INTERVAL 7 DAY;
```

The screenshot shows the same database management tool interface. The query editor shows 'Query 1' with the following SQL:

```
1 • SELECT * FROM Weather
2 WHERE timestamp > NOW() - INTERVAL 7 DAY;
3 |
```

The 'Result Grid' at the bottom shows a table with 6 columns: id, city_name, temperature, humidity, wind_speed, and timestamp. The first two rows are highlighted in blue.

id	city_name	temperature	humidity	wind_speed	timestamp
1	New York	22.50	60	10.20	2025-03-09 22:59:37
2	Los Angeles	23.50	40	8.70	2025-03-09 22:59:37
*	NULL	NULL	NULL	NULL	NULL

Sort Weather Data by Temperature (Descending):

```
SELECT city_name, temperature, humidity, wind_speed
FROM Weather
ORDER BY temperature DESC;
```

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' pane displays the 'weatherapp' database structure, including a table named 'weather'. The main query window, titled 'Query 1', contains the following SQL code:

```
1 • SELECT city_name, temperature, humidity, wind_speed
2 FROM Weather
3 ORDER BY temperature DESC;
4
```

Below the query window, the 'Result Grid' displays the results of the query, limited to 1000 rows. The results are sorted by temperature in descending order.

city_name	temperature	humidity	wind_speed
Los Angeles	23.50	40	8.70
New York	22.50	60	10.20

Create an Additional Table for City Information:

```
CREATE TABLE CityInfo (
    city_id INT PRIMARY KEY AUTO_INCREMENT,
    city_name VARCHAR(100),
    population INT,
    country VARCHAR(100)
);
```

Insert Sample City Information Data:

```
INSERT INTO CityInfo (city_name, population, country)
VALUES
    ('New York', 8419600, 'USA'),
    ('Los Angeles', 3980400, 'USA'),
    ('Chicago', 2716000, 'USA');
```

Join Weather Data with City Information:

```
SELECT w.city_name, w.temperature, w.humidity, w.wind_speed, c.population, c.country
FROM Weather w
JOIN CityInfo c ON w.city_name = c.city_name;
```

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'SCHEMAS' pane displays the 'weatherapp' database structure, including a table named 'weather'. The main query window, titled 'Query 1', contains the following SQL code:

```
1 • SELECT w.city_name, w.temperature, w.humidity, w.wind_speed, c.population, c.country
2 FROM Weather w
3 JOIN CityInfo c ON w.city_name = c.city_name;
4
```

Below the query window, the 'Result Grid' displays the results of the query, limited to 1000 rows. The results show the weather data joined with the city information from the 'CityInfo' table.

city_name	temperature	humidity	wind_speed	population	country
New York	22.50	60	10.20	8419600	USA
Los Angeles	23.50	40	8.70	3980400	USA