Dataset: city_temperature.csv

Context: Climate Analysis

In this notebook, I conducted a time-series analysis using SQL to identify **seasonality, trends, and cumulative variations** in global temperature data.
The objective was to produce clean and interpretable outputs suitable for integration into dashboards and strategic environmental reports.

Skills demonstrated: Time-series analysis, cumulative metrics, trend analysis, SQL aggregation
Language: SQL

```
--Exploratory analysis to verify how many days equal to "0" there are in the "Day" column and some examples of cities --w
```

	Day int64	qtd_registros int...	ano_min int64	ano_max int64	exemplos_cidades
0	0	8	2008	2016	['Bissau', 'Lagos', '...

1 row, 5 cols | 10 / page | Page 1 of 1

```
-- Here a temporary table has been created with three extra columns: "data_evento", "category" and "value". The column -
City AS category,
Region,
Country
State FROM read_csv_auto('city_temperature.csv') WHERE Day >= 1; SELECT * FROM base_city_temperature;
```

	data_evento date...	Year int64	Month int64	Day int64	value float64	category object	Region object	C
0	1995-01-01 00:0...	1995	1	1	64.2	Algiers	Africa	A
1	1995-01-02 00:0...	1995	1	2	49.4	Algiers	Africa	A
2	1995-01-03 00:0...	1995	1	3	48.8	Algiers	Africa	A
3	1995-01-04 00:0...	1995	1	4	46.4	Algiers	Africa	A
4	1995-01-05 00:0...	1995	1	5	47.9	Algiers	Africa	A
5	1995-01-06 00:0...	1995	1	6	48.7	Algiers	Africa	A
6	1995-01-07 00:0...	1995	1	7	48.9	Algiers	Africa	A
7	1995-01-08 00:0...	1995	1	8	49.1	Algiers	Africa	A
8	1995-01-09 00:0...	1995	1	9	49	Algiers	Africa	A
9	1995-01-10 00:0...	1995	1	10	51.9	Algiers	Africa	A

2,906,319 rows, 9 cols | 10 / page | Page 1 of 290,632

```
-- Identification of the number of days as the interval between one event and another (average temperature). The -- (citi
data_evento, LAG(data_evento) OVER ( PARTITION BY category ORDER BY data_evento
) THE data_anterior,
DATE_DIFF('day', LAG(data_evento) OVER (PARTITION BY category ORDER BY  data_evento),
data_evento
) AS intervalo_dias FROM base_city_temperature WHERE data_evento IS NOT NULL; SELECT * FROM city_event_intervals;
```

	category object	data_evento date...	data_anterior dat...	intervalo_dias flo...
0	Chattanooga	1995-01-01 00:0...	Nat	Nan
1	Chattanooga	1995-01-02 00:0...	1995-01-01 00:0...	1
2	Chattanooga	1995-01-03 00:0...	1995-01-02 00:0...	1
3	Chattanooga	1995-01-04 00:0...	1995-01-03 00:0...	1
4	Chattanooga	1995-01-05 00:0...	1995-01-04 00:0...	1
5	Chattanooga	1995-01-06 00:0...	1995-01-05 00:0...	1
6	Chattanooga	1995-01-07 00:0...	1995-01-06 00:0...	1
7	Chattanooga	1995-01-08 00:0...	1995-01-07 00:0...	1
8	Chattanooga	1995-01-09 00:0...	1995-01-08 00:0...	1
9	Chattanooga	1995-01-10 00:0...	1995-01-09 00:0...	1

2,906,319 rows, 4 cols | 10 / page | Page 1 of 290,632

```
-- In the code below a table "temp_semanal" was created with a column called "week" whose values are  -- represented by t
DATE_TRUNC('week', data_evento) AS semana, AVG(valor) AS temp_media_semanal FROM base_city_temperature GROUP BY categori
DATE_TRUNC('month', data_evento) AS month, AVG(value) AS temp_media_mensal FROM base_city_temperature GROUP BY category,
DATE_TRUNC('quarter', data_evento) AS quarter, AVG(value) AS temp_media_trimestral FROM base_city_temperature GROUP BY c
s.temp_media_semanal,
m.temp_media_mensal,
t.temp_media_trimestral FROM temp_semanal s LEFT JOIN temp_mensal m ON m.category = s.category AND DATE_TRUNC('month', s
```

	week datetime64... 1994-12-26 00:0...	temp_media_se... 22.87142857142...	temp_media_me... 33.02580645161...	temp_media_trim... 38.72222222222...
40	1995-10-02 00:0...	64.01428571	60.22	48.51590909
41	1995-10-09 00:0...	63.98333333	60.22	48.51590909
42	1995-10-16 00:0...	57.24285714	60.22	48.51590909
43	1995-10-23 00:0...	59.34285714	60.22	48.51590909
44	1995-10-30 00:0...	45.31428571	60.22	48.51590909
45	1995-11-06 00:0...	49.32857143	46.72142857	48.51590909
46	1995-11-13 00:00...	47.15714286	46.72142857	48.51590909
47	1995-11-20 00:00...	47.21666667	46.72142857	48.51590909
48	1995-11-27 00:00...	43.66666667	46.72142857	48.51590909
49	1995-12-04 00:0...	34.73333333	38.48666667	48.51590909

1,325 rows, 4 cols 10 / page « ‹ Page 5 of 133 › »

```
-- In the code below we use windowing functions to return the average temperature (value) of the last -- 7 days (includin
data_evento,
value, AVG(value) OVER( PARTITION BY category ORDER BY data_evento ROWS BETWEEN 6 PRECEDING AND CURRENT ROW ) AS media_m
value
media_movel_7d,
media_movel_30d FROM tendencia_movel WHERE category = 'Paris' ORDER BY data_evento DESC LIMIT 50;
```

	data_evento date... 2020-03-25 00:0...	value float64 40.7 - 64.4	media_movel_7d f.. 43.94285714285...	media_movel_30d f 46.9933333333...
0	2020-05-13 00:0...	43.6	54.78571429	56.61666667
1	2020-05-12 00:0...	49.3	57.08571429	56.97666667
2	2020-05-11 00:0...	50	56.95714286	57.38666667
3	2020-05-10 00:0...	60.1	58.2	57.85666667
4	2020-05-09 00:0...	58.9	57.6	57.97
5	2020-05-08 00:0...	62.1	56.84285714	58.09666667
6	2020-05-07 00:0...	59.5	55.31428571	58.08333333
7	2020-05-06 00:0...	59.7	54.2	58.06
8	2020-05-05 00:0...	48.4	53.01428571	57.79
9	2020-05-04 00:0...	58.7	54.54285714	58.05666667

50 rows, 4 cols 10 / page « ‹ Page 1 of 5 › »

```
-- In the first block of code, a "evolucao_mensal" table was created with the monthly average of each category, that is,
 DATE_TRUNC('month', data_evento) AS month, AVG(value) AS media_mensal FROM base_city_temperature WHERE value IS NOT NULL
 Month
 media_mensal, LAG(media_mensal) OVER ( PARTITION BY category ORDER BY  mes
 ) THE media_mes_anterior,
 ROUND(
 (media_mensal - LAG(media_mensal) OVER (PARTITION BY category ORDER BY month)) /LAG(media_mensal) OVER (PARTITION BY cat
LIMIT 20;
```

	categoria object Paris 100%	mes datetime64[... 1995-01-01 00:0...	media_mensal fl... 37.77931034482...	media_mes_ante... 37.77931034482...	variacao_percent... -22.42 - 21.4
0	Paris	1995-01-01 00:0...	42.15806452	nan	nan
1	Paris	1995-02-01 00:0...	48.16071429	42.15806452	14.24
2	Paris	1995-03-01 00:0...	45.93870968	48.16071429	-4.61
3	Paris	1995-04-01 00:0...	51.22666667	45.93870968	11.51
4	Paris	1995-05-01 00:0...	60.04193548	51.22666667	17.21
5	Paris	1995-06-01 00:0...	63.74	60.04193548	6.16
6	Paris	1995-07-01 00:0...	73.70322581	63.74	15.63
7	Paris	1995-08-01 00:0...	72.65483871	73.70322581	-1.42
8	Paris	1995-09-01 00:0...	60.52	72.65483871	-16.7
9	Paris	1995-10-01 00:0...	60.22	60.52	-0.5

20 rows, 5 cols 10 / page Page 1 of 2

```
-- Como primeira etapa do código abaixo, foi tirada a média da temperatura (valor) de cada mês de cada cidade
-- (categoria) e depois foi utilizada a função de janelamento OVER(ROWS BETWEEN x PRECEDING ...) para retornar a
-- média móvel de um período de 3 meses e outra de um período de 5 meses.

CREATE OR REPLACE TABLE rolling_mensal AS
SELECT
    categoria,
    DATE_TRUNC('month', data_evento) AS mes,
    AVG(valor) AS media_mensal,
    AVG(AVG(valor)) OVER(
        PARTITION BY categoria
        ORDER BY DATE_TRUNC('month', data_evento)
        ROWS BETWEEN 2 PRECEDING AND CURRENT ROW
    ) AS media_3m,

    AVG(AVG(valor)) OVER(
        PARTITION BY categoria
        ORDER BY DATE_TRUNC('month', data_evento)
        ROWS BETWEEN 4 PRECEDING AND CURRENT ROW
    ) AS media_5m
FROM base_city_temperature
WHERE valor IS NOT NULL
GROUP BY categoria, DATE_TRUNC('month', data_evento)
ORDER BY categoria, mes;

SELECT * FROM rolling_mensal;
```

	categoria object Abidjan 0.3% Abilene 0.3% 319 others 99.3%	mes datetime64[...	media_mensal fl...	media_3m float64	media_5m float64
0	Abidjan	1995-01-01 00:0...	79.91612903	79.91612903	79.91612903
1	Abidjan	1995-02-01 00:0...	82.61428571	81.26520737	81.26520737
2	Abidjan	1995-03-01 00:0...	82.54516129	81.69185868	81.69185868
3	Abidjan	1995-04-01 00:0...	83.32	82.82648233	82.09889401
4	Abidjan	1995-05-01 00:0...	82.40322581	82.75612903	82.15976037
5	Abidjan	1995-06-01 00:0...	80.79666667	82.17329749	82.3358679
6	Abidjan	1995-07-01 00:0...	78.5	80.56663082	81.51301075
7	Abidjan	1995-08-01 00:0...	77.53	78.94222222	80.50997849
8	Abidjan	1995-09-01 00:0...	77.26333333	77.76444444	79.29864516
9	Abidjan	1995-10-01 00:0...	79.23548387	78.00960573	78.66509677

92,913 rows, 5 cols 10 / page Page 1 of 9,292

```sql
-- No código abaixo foi utilizada a função SUM para somar todas as temperaturas que passaram dos 65F e que foram
-- abaixo dos 65F. Aqui este valor foi considerado como o "confortável", por isso está sendo utilizado como referencia. 6
-- Alem disso, o EXTRACT(YEAR FROM...) foi utilizado para reiniciar a cada ano.

CREATE OR REPLACE TEMP TABLE acumulado_degree_days AS
SELECT
    categoria,
    data_evento,
    valor AS temp_fahrenheit,


    SUM(
        CASE WHEN valor > 65 THEN valor - 65 ELSE 0 END
    ) OVER (
        PARTITION BY categoria, EXTRACT(YEAR FROM data_evento)
        ORDER BY data_evento
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) AS acima_65,


    SUM(
        CASE WHEN valor < 65 THEN 65 - valor ELSE 0 END
    ) OVER (
        PARTITION BY categoria, EXTRACT(YEAR FROM data_evento)
        ORDER BY data_evento
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    ) AS abaixo_65

FROM base_city_temperature
WHERE data_evento IS NOT NULL
  AND valor IS NOT NULL
ORDER BY categoria, data_evento;


SELECT
    data_evento,
    temp_fahrenheit,
    acima_65,
    abaixo_65
FROM acumulado_degree_days
WHERE categoria = 'Paris'
  AND EXTRACT(YEAR FROM data_evento) = 2010
ORDER BY data_evento;
```

	data_evento date... 2010-01-01 00:0...	temp_fahrenheit f... 22.9 - 81.4	acima_65 float64 0.0 - 351.700000...	abaixo_65 float64 32.3 - 5225.1
150	2010-05-31 00:0...	56.7	16.7	2959.8
151	2010-06-01 00:0...	56.6	16.7	2968.2
152	2010-06-02 00:0...	61.2	16.7	2972
153	2010-06-03 00:0...	64.9	16.7	2972.1
154	2010-06-04 00:0...	69.9	21.6	2972.1
155	2010-06-05 00:0...	72.9	29.5	2972.1
156	2010-06-06 00:0...	67.3	31.8	2972.1
157	2010-06-07 00:0...	64.7	31.8	2972.4
158	2010-06-08 00:0...	67.4	34.2	2972.4
159	2010-06-09 00:0...	63.2	34.2	2974.2

365 rows, 4 cols 10 / page Page 16 of 37

```sql
-- Neste primeiro bloco de código foi calculada a média de cada mês de cada cidade (categoria) utilizando a função
-- AVG() e o DATE_TRUNC para registrar o primeiro valor do mês. Uma tabela de base "media_mensal" foi criada.

CREATE OR REPLACE TABLE media_mensal AS
SELECT
    categoria,
    DATE_TRUNC('month', data_evento) AS mes,
    AVG(valor) AS media_temp_mensal
FROM base_city_temperature
WHERE valor IS NOT NULL
GROUP BY categoria, DATE_TRUNC('month', data_evento)
ORDER BY categoria, mes;

-- Neste segundo bloco de código o LAG() foi utilizado para acessar tanto a média mensal do mês anterior quanto a
-- média mensal do mês referente a 12 meses atrás e com isto foram criadas duas novas colunas chamadas media_ano_
-- anterior e media_mes_anterior.


CREATE OR REPLACE TABLE comparacoes AS
SELECT
    categoria,
    mes,
    media_temp_mensal,
    LAG(media_temp_mensal, 12) OVER(
        PARTITION BY categoria
        ORDER BY mes
    ) AS media_ano_anterior,
    LAG(media_temp_mensal, 1) OVER (
        PARTITION BY categoria
        ORDER BY mes
    ) AS media_mes_anterior,

-- O próximo passo foi então utilizar o CASE WHEN para calcular a porcentagem referente à diferença de temperatura
-- de um mês para o outro (variacao_mom) e de um mesmo mês de um ano para o outro (variacao_yoy).

    CASE
        WHEN LAG(media_temp_mensal, 12) OVER (PARTITION BY categoria ORDER BY mes) IS NOT NULL
        THEN ROUND(
            (media_temp_mensal - LAG(media_temp_mensal, 12) OVER (PARTITION BY categoria ORDER BY mes))
            / LAG(media_temp_mensal, 12) OVER (PARTITION BY categoria ORDER BY mes) * 100, 2
        )
    END AS variacao_yoy,

    CASE
        WHEN LAG(media_temp_mensal, 1) OVER (PARTITION BY categoria ORDER BY mes) IS NOT NULL
        THEN ROUND(
            (media_temp_mensal - LAG(media_temp_mensal, 1) OVER (PARTITION BY categoria ORDER BY mes))
            / LAG(media_temp_mensal, 1) OVER (PARTITION BY categoria ORDER BY mes) * 100, 2
        )
    END AS variacao_mom

FROM media_mensal
ORDER BY categoria, mes;

-- Bloco de código para visualizar a cidade 'Paris' como referência.

SELECT * FROM comparacoes
WHERE categoria = 'Paris'
ORDER BY mes
LIMIT 24;
```

	categoria object Paris 100%	mes datetime64[... 1995-01-01 00:0...	media_temp_me... 36.59677419354...	media_ano_anter... 38.48666666666...	media_mes_ante... 37.77931034482...	variacao_yoy floa... -21.56 - 5.79	variacao_mom fl... -22.42 - 21.4
10	Paris	1995-11-01 00:00...	46.72142857	nan	60.22	nan	-22.42
11	Paris	1995-12-01 00:0...	38.48666667	nan	46.72142857	nan	-17.63
12	Paris	1996-01-01 00:0...	39.23	42.15806452	38.48666667	-6.95	1.93
13	Paris	1996-02-01 00:0...	37.77931034	48.16071429	39.23	-21.56	-3.7
14	Paris	1996-03-01 00:0...	44.84193548	45.93870968	37.77931034	-2.39	18.69
15	Paris	1996-04-01 00:0...	53.57	51.22666667	44.84193548	4.57	19.46
16	Paris	1996-05-01 00:0...	55.54193548	60.04193548	53.57	-7.49	3.68
17	Paris	1996-06-01 00:0...	67.43	63.74	55.54193548	5.79	21.4
18	Paris	1996-07-01 00:0...	68.69032258	73.70322581	67.43	-6.8	1.87
19	Paris	1996-08-01 00:0...	68.16129032	72.65483871	68.69032258	-6.18	-0.77

24 rows, 7 cols 10 / page Page 2 of 3