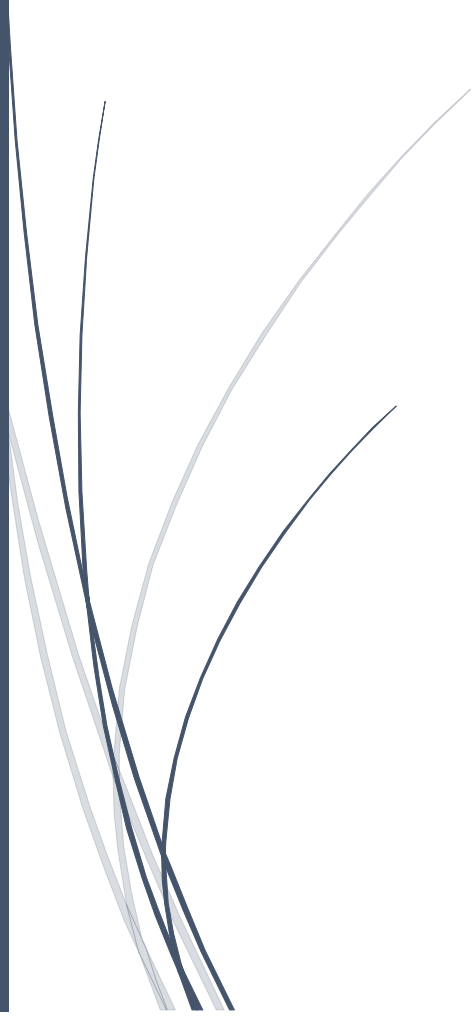
08-10-2023

Trabalho Metodologias de Recolha de Dados

Parte I – API da TomTom | Parte II – SQL

Professor Luís Miguel Sousa



Maria Neves

Parte I – API da TomTom

A Baixa Pombalina, no coração de Lisboa, é uma área histórica reconstruída após o grande terramoto de 1755, sob a supervisão de Marquês de Pombal. Apresenta um *layout* geométrico, ruas largas e edifícios neoclássicos. Um café é considerado um estabelecimento onde se compra e consome café, outras bebidas e também alimentos. São também locais onde as pessoas se encontram, conversam, trabalham ou relaxam.

Código:

```
import requests
import pandas as pd
import json
# Usar a API da TomTom para pesquisar cafés na baixa Pombalina de
Lisboa com base em coordenadas geográficas (latitude, longitude)
num determinado raio
# Definição da chave da API da TomTom
my_api_key = 'KboRp8rfmlBblHkMCuguZVjwtEA0GhB7'

# Latitude e longitude da Baixa Pombalina de Lisboa
latitude = 38.712961
longitude = -9.138617

# Raio de pesquisa em metros
radius = 5000

# URL da API da TomTom
url =
f'https://api.tomtom.com/search/2/categorySearch/café.json?key={my_
api_key}&lat={latitude}&lon={longitude}&radius={radius}'

# Solicitação GET à API da TomTom
response = requests.get(url)

# Verificar o código de status da resposta
if response.status_code == 200:
    data = response.json()
    cafes = data.get('results', [])

    if cafes:
        # Construção de um DataFrame com os resultados da pesquisa,
incluindo nome do café, endereço e categoria
        df = pd.DataFrame([{'Nome do café': cafe.get('poi',
{}).get('name'),
                            'Endereço': cafe.get('address',
{}).get('freeformAddress', ''),
                            'Categoria': cafe.get('poi',
{}).get('categories', '')}
                           for cafe in cafes])
        print(df)
```

```
    else:
        print('Nenhum café encontrado na baixa pombalina de
Lisboa')

else:
    print('Erro ao consultar a API da TomTom')
```

Explicação:

Este código utiliza a chave da API da TomTom, juntamente com coordenadas de latitude e longitude da Baixa Pombalina de Lisboa e também um raio de pesquisa em metros. A solicitação GET é enviada para a API da TomTom e os resultados são obtidos e exibidos num DataFrame Pandas, incluindo o Nome do Café, endereço e categoria.

Metodologia:

- Utilização da API da TomTom para pesquisa por categoria (“Café”) com base em coordenadas geográficas e o raio.
- Os parâmetros, como a chave da API, latitude, longitude e raio. A latitude e a longitude foram definidas com base em informação da EuroVelo (https://euroveloportugal.com/pt/poi/baixa-pombalina-2).
- Uma solicitação GET é feita à API usando a URL construída com estes parâmetros.
- O Código verifica o status da resposta e se for bem-sucedido, extrai os resultados da pesquisa.

Pressupostos:

- A chave da API da TomTom é válida.
- As coordenadas de latitude e longitude estão corretas e representam a área da Baixa Pombalina de Lisboa.
- A API da TomTom dá resultados relevantes e precisos para a categoria "café" dentro do raio especificado.

Critérios de aceitação:

- A resposta da API da TomTom deve ter um código de status 200 (sucesso).
- Deve existir pelo menos um café encontrado dentro do raio definido.
- Os dados devolvidos devem incluir o nome do café, endereço e categoria.

Resultado:

```
                Nome do café                                      Endereço  \
0         Brown Coffee Bean                Rua da Vitória, 1100-619 Lisboa
1                 Club Noir                   Rua da Madalena, 1100 Lisboa
2             Ginginha Cima  Rua das Portas de Santo Antão, 1150-266 Lisboa
3            Café 100 Artes             Rua dos Fanqueiros, 1100-228 Lisboa
4              Vertigo Café              Travessa do Carmo, 1200-369 Lisboa
5  Kiosque De Principe Real       Rua do Jardim do Regedor, 1169-207 Lisboa
6              Delmare Café             Rua dos Sapateiros, 1100-579 Lisboa
7                Kaffeehaus                 Rua Anchieta 3, 1200-023 Lisboa
8      The Corner Irish Pub           Travessa da Queimada, 1200-115 Lisboa
9       Bar Lábios De Vinho                Rua do Norte 52, 1200-283 Lisboa

           Categoria
0  [café, café/pub]
1        [café/pub]
2   [café/pub, pub]
3  [café, café/pub]
4  [café, café/pub]
5  [café, café/pub]
6  [café, café/pub]
7  [café, café/pub]
8   [café/pub, pub]
9        [café/pub]
```

Podemos ver qual o tamanho do DataFrame por forma a ver quantos cafés temos na área:

```
len(df)
```

Resultado:

```
10
```

Portanto, existem 10 cafés na Baixa Pombalina de Lisboa. De seguida, podemos verificar quais deles são efetivamente cafés, por forma a verificar a presença de falsos positivos.

```
#Verificar quais são cafés
def is_cafe(categories):
    return 'café' in categories

cafes_df = df[df['Categoria'].apply(is_cafe)]
print(cafes_df)
```

Resultado:

```
              Nome do café                                  Endereço  \
0         Brown Coffee Bean            Rua da Vitória, 1100-619 Lisboa
3            Café 100 Artes         Rua dos Fanqueiros, 1100-228 Lisboa
4              Vertigo Café          Travessa do Carmo, 1200-369 Lisboa
5  Kiosque De Principe Real  Rua do Jardim do Regedor, 1169-207 Lisboa
6              Delmare Café         Rua dos Sapateiros, 1100-579 Lisboa
7                Kaffeehaus            Rua Anchieta 3, 1200-023 Lisboa

          Categoria
0  [café, café/pub]
3  [café, café/pub]
4  [café, café/pub]
5  [café, café/pub]
6  [café, café/pub]
7  [café, café/pub]
```

A API da TomTom dá-nos resultados que não são cafés, alguns são *pubs*. Apenas 6 têm na sua categoria café. Estes resultados podem ser considerados Falsos Positivos. Já os Falsos Negativos seriam obtidos se nenhum café fosse encontrado na área estabelecida.

De seguida, vamos verificar se existem duplicados no DataFrame extraído da API da TomTom:

```python
# Verificar se existem duplicados com base no nome do café e no endereço
duplicates = df[df.duplicated(['Nome do café', 'Endereço'], keep=False)]

if not duplicates.empty:
    print("Duplicados encontrados:")
    print(duplicates)
else:
    print("Nenhum duplicado encontrado.")
```

Resultado:

```
Nenhum duplicado encontrado.
```

Parte II – SQL

Pergunta 1: Que percentagem das partidas não acabaram devido a um erro?

```
#1. Percentagem das partidas que não acabaram devido a um erro
SELECT
      COUNT(*) AS total_partidas,/* Contar o número total de partidas*/
      SUM(error) AS total_partidas_com_erro,/*Contar o número de partidas
que não acabaram devido a erro*/
    (SUM(error) / COUNT(*)) * 100 AS Percentagem_partidas_com_erro/*Obter a
percentagem de partidas que não foram concluídas devido a um erro*/
FROM GAME.MATCHES;
```

Resultado:

total_partidas	total_partidas_com_erro	Percentagem_partidas_com_erro
90,566	441	0.4869

Pergunta 2: Em média, quantas partidas foram jogadas em cada mês?

```
#Questão 2. Em média, quantas partidas foram jogadas em cada mês?
SELECT
      MONTH(datetime) AS MES, /*selecionar o mês da coluna 'datetime'*/
      COUNT(*) as Partidas_por_mes /*contar o número total de partidas
jogadas por mês*/
FROM GAME.MATCHES
GROUP BY MES /*Contar as partidas jogadas por cada mês*/
ORDER BY MES; /*Ordenar os resultados por mês, de janeiro a dezembro*/
```

Resultado:

	MONTH(datetime)	COUNT(*)
1	1	9,475
2	2	8,648
3	3	9,175
4	4	8,775
5	5	8,876
6	6	8,388
7	7	6,714
8	8	6,182
9	9	6,041
10	10	6,272
11	11	5,911
12	12	6,109

Pergunta 3: Em média, quantas partidas foram jogadas por cada jogador?

```
#Questão 3. Em média, quantas partidas foram jogadas por cada jogador?
WITH GAMESPERPLAYER AS (
  SELECT player1_id AS player
  FROM GAME.MATCHES
  UNION ALL
  SELECT player2_id AS player
  FROM GAME.MATCHES
)
/*criação de uma CTE chamada GAMESPERPLAYER, a CTE vai selecionar o
player1_id e o player2_id da tabela GAME.MATCHES
```

```
UNION ALL para juntar os dois conjuntos de dados, por forma a considerar
ambos os jogadores para o cálculo*/
SELECT
  player, /*Selecionar todos os jogadores que participaram*/
  COUNT(*) AS total_partidas /*Contar o número de partidas jogadas por cada
jogador*/
FROM GAMESPERPLAYER
GROUP BY player /*agrupar os resultados por 'player' para obter uma linha por
cada jogador*/
ORDER BY total_partidas DESC; /*ordenar os resultados por ordem decrescente
tendo por base o número de partidas jogadas por cada jogador*/
```

Resultado:

	player	total_partidas
1	283	2,274
2	485	2,261
3	95	2,248
4	492	2,246
5	178	2,240
6	93	2,233
7	213	2,222
8	409	2,215
9	16	2,208
10	578	2,204
11	863	2,201
12	973	2,198
13	725	2,195
14	629	2,188
15	11	2,185
16	601	2,166
17	683	2,164
18	94	2,154
19	88	2,149

Pergunta 4: Olhando só para as partidas entre homens e mulheres, que género ganha mais frequentemente?

```
#Questão 4. Olhando só para as partidas entre homens e mulheres, que género
ganha mais frequentemente?
/*Criação de uma CTE chamada P3 para verificar o género do vencedor de cada
partida*/
WITH P3 AS (
    SELECT
        CASE
            /*Quando o score é igual a 1, significa que o player 2 ganhou,
por isso vamos buscar o género do player 2*/
              WHEN M.score = 1 THEN P2.gender
            /*Quando o score é igual a -1, significa que o player 1 ganhou,
por isso vamos buscar o género do player 1*/
              WHEN M.score = -1 THEN P1.gender
        END AS winner, M.score
    FROM GAME.MATCHES AS M
    /*Juntar as tabelas que contém a informação dos jogadores (players) e das
partidas (matches)*/
    LEFT JOIN GAME.PLAYERS AS P1 ON M.player1_id = P1.id
    LEFT JOIN GAME.PLAYERS AS P2 ON M.player2_id = P2.id
```

```
    /*Filtrar as partidas entre homens e mulheres*/
    WHERE (P1.gender = 'male' AND P2.gender = 'female') OR (P1.gender =
'female' AND P2.gender = 'male')
)
/*Contar o número de vitórias para cada género*/
SELECT winner, COUNT(*) AS total
FROM P3
/*Excluir valores nulos dos resultados*/
WHERE winner IS NOT NULL
/*Agrupar os resultados pelo género vencedor para obter o número de vitórias
para cada género*/
GROUP BY winner;
```

Resultado:

winner	total
male	20,955
female	20,971

Pergunta 5: Qual o top 10 de jogadores que ganharam mais partidas?

```
SELECT
/*Criar uma lista de IDs de jogadores que venceram*/
    CONCAT(P.first_name, ' ', P.last_name) AS nome_completo,
    /*Contar o número de vitórias de cada jogador*/
    COUNT(*) AS total_vitorias
FROM /*Se o player1 venceu -> score = -1 e se o player 2 venceu -> score =
1*/
(SELECT player2_id AS player_id FROM GAME.MATCHES WHERE score = -1
 UNION ALL
 SELECT player1_id AS player_id FROM GAME.MATCHES WHERE score = 1) AS winners
/*Unir os IDs dos jogadores que venceram com os dados dos jogadores da tabela
players*/
 LEFT JOIN GAME.PLAYERS AS P ON winners.player_id = P.id
/*Agrupar os resultados pelo nome completo dos jogadores*/
GROUP BY nome_completo
/*Ordenar os resultados por forma a colocar os jogadores com mais vitórias no
topo*/
ORDER BY total_vitorias DESC
/*Limitar a pesquisa a 10 resultados*/
LIMIT 10;
```

Resultado:

	nome_completo	total_vitorias
1	Wilson Reis	1,152
2	Aimée Julien	1,147
3	John Pearson	1,144
4	Katie Hughes	1,123
5	Rafael Andrade	1,122
6	Terry Stevens	1,119
7	Gabriel Boucher	1,112
8	Marco Paiva	1,109
9	Fernanda das Neves	1,101
10	Hernando Guzmán	1,099

Pergunta 6: Qual é a idade média dos vencedores e a dos derrotados em Gamo?

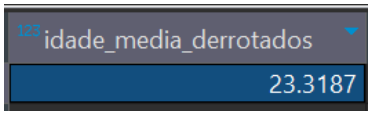
```
6. Qual é a idade média dos vencedores e a dos derrotados em Gamo?
/*Idade média dos vencedores*/
SELECT
/*Calcular a média da diferença de idades dos vencedores*/
      AVG(CASE WHEN M.score = 1 THEN EXTRACT(YEAR FROM datetime) -
EXTRACT(YEAR FROM P2.birth_date)
            WHEN M.score = -1 THEN EXTRACT(YEAR FROM datetime) -
EXTRACT(YEAR FROM P1.birth_date) END) AS idade_media_vencedores
/*Quando o score = 1, calcular a diferença entre o ano do jogo e o ano de
nascimento do player 2
 *Quando o score = -1, calcular a diferença entre o ano do jogo e o ano de
nascimento do player 1,
 *uma vez que score = 1 indica que o player 2 ganhou e score = -1 indica que
o player 1 ganhou*/
FROM GAME.MATCHES M
LEFT JOIN GAME.PLAYERS P1 ON M.score = -1 AND M.player1_id = P1.id
/*Unir as tabelas 'PLAYERS' e 'MATCHES'. Quando o score for -1, indica que o
player1_id ganhou,
 * vamos usar então o player1_id da tabela 'MATCHES' para encontrar o jogador
correspondente na tabela 'PLAYERS'*/
LEFT JOIN GAME.PLAYERS P2 ON M.score = 1 AND M.player2_id = P2.id
/*Unir as tabelas 'PLAYERS' e 'MATCHES'. Quando o score for 1, indica que o
player2_id ganhou,
 * vamos usar então o player2_id da tabela 'MATCHES' para encontrar o jogador
correspondente na tabela 'PLAYERS'*/
```

Resultado:

idade_media_vencedores
23.3074

```
/*Idade média dos derrotados*/
SELECT
    /*Calcular a média da diferença de idades dos derrotados*/
      AVG(CASE WHEN M.score = 1 THEN EXTRACT(YEAR FROM datetime) -
EXTRACT(YEAR FROM P1.birth_date)
            WHEN M.score = -1 THEN EXTRACT(YEAR FROM datetime) -
EXTRACT(YEAR FROM P2.birth_date) END) AS idade_media_derrotados
/*Quando o score = 1, calcular a diferença entre o ano do jogo e o ano de
nascimento do player 1.
 *Quando o score = -1, calcular a diferença entre o ano do jogo e o ano de
nascimento do player 2,
 *uma vez que quando score = 1 -> o player 2 ganhou e score = -1 -> o player
1 ganhou,
 *para calcular os jogadores derrotados, queremos o contrário*/
FROM GAME.MATCHES M
LEFT JOIN GAME.PLAYERS P1 ON M.score = 1 AND M.player1_id = P1.id
/*Unir as tabelas 'PLAYERS' e 'MATCHES'. Quando o score for 1, indica que o
player1_id perdeu,
 * vamos usar então o player1_id da tabela 'MATCHES' para encontrar o jogador
correspondente na tabela 'PLAYERS'*/
LEFT JOIN GAME.PLAYERS P2 ON M.score = -1 AND M.player2_id = P2.id;
/*Unir as tabelas 'PLAYERS' e 'MATCHES'. Quando o score for -1, indica que o
player2_id perdeu,
 * vamos usar então o player2_id da tabela 'MATCHES' para encontrar o jogador
correspondente na tabela 'PLAYERS'*/
```

Resultado:

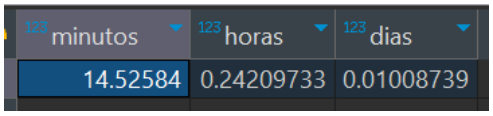


idade_media_derrotados
23.3187

Pergunta 7: “Em média, de X em X minutos/horas/dias alguém começa uma partida”. Determine X.

```
#7. “Em média, de X em X minutos/horas/dias alguém começa uma partida”.
Determine X.
/*Definir uma CTE denominada Intervalo*/
WITH INTERVALO AS (
    SELECT
/*Calcular a diferença de tempo em segundos entre o início de uma partida e a
partida anterior*/
        TIMESTAMPDIFF(SECOND, LAG(datetime) OVER (ORDER BY datetime),
datetime) AS intervalo_segundos
    FROM GAME.MATCHES
)
/*Calcular os valores médios dos intervalos convertidos.
    * Intervalo em minutos dividimos a média dos intervalos em segundos por
60,
    * para horas dividimos por 3600 (60 minutos * 60 segundos) e para dias
dividimos por 86400 (24 horas * 60 minutos * 60 segundos)*/
SELECT
    AVG(intervalo_segundos / 60) AS minutos,
    AVG(intervalo_segundos / 3600) AS horas,
    AVG(intervalo_segundos / 86400) AS dias
FROM INTERVALO
/*IS NOT NULL garante que os valores nulos não afetam o cálculo dos valores
médios*/
WHERE intervalo_segundos IS NOT NULL;
```

Resultado:



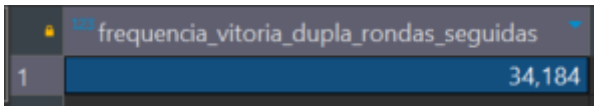
minutos	horas	dias
14.52584	0.24209733	0.01008739

Pergunta 8: Nos jogos com mais de 3 rondas, com que frequência o jogador derrotado ganhou duas rondas seguidas?

```
#8. Nos jogos com mais de 3 rondas, com que frequência o jogador derrotado
ganhou duas rondas seguidas?
/*Criação de uma CTE chamada M1, onde vamos selecionar dados das partidas da
tabela GAME.MATCHES.*/
WITH M1 AS (
    SELECT
        M.id AS partida_id,
        M.score AS score,
        M.player1_id AS player_1,
        M.player2_id AS player_2,
        JSON_UNQUOTE(JSON_EXTRACT(M.rounds, '$[0].winner')) AS round1_winner,
        JSON_UNQUOTE(JSON_EXTRACT(M.rounds, '$[1].winner')) AS round2_winner,
        JSON_UNQUOTE(JSON_EXTRACT(M.rounds, '$[2].winner')) AS round3_winner,
        JSON_UNQUOTE(JSON_EXTRACT(M.rounds, '$[3].winner')) AS round4_winner
```

```
    FROM
        GAME.MATCHES AS M
),
/*Criação de uma segunda CTE chamada M2, onde vamos selecionar dados da CTE
M1.
 * Esta CTE vai filtrar as partidas onde todos os resultados das quatro
rondas não são nulos.*/
M2 AS (
    SELECT
        M1.player_1,
        M1.player_2,
        M1.score,
        M1.round1_winner,
        M1.round2_winner,
        M1.round3_winner,
        M1.round4_winner
    FROM
        M1
    WHERE
        M1.round1_winner IS NOT NULL
        AND M1.round2_winner IS NOT NULL
        AND M1.round3_winner IS NOT NULL
        AND M1.round4_winner IS NOT NULL
),
/*Criação de uma terceira CTE, onde vamos calcular a frequência com que o
jogador derrotado (seja o jogador 1 ou 2 dependendo do score)
 * venceu duas rondas seguidas. Neste passo utilizamos a função SUM e uma
cláusula CASE. Com o CASE vamos verificar se o score é -1 ou 1,
 * Depois verificamos se o jogador derrotado venceu duas rondas seguidas. Se
a condição for atendida ele soma 1 caso contrário adiciona 0.
 * A soma de todas as ocorrências onde a condição é atendida é por fim
calculada*/
M3 AS (
    SELECT
        SUM(
            CASE
                WHEN (
                    (M2.score = 1 OR M2.score = -1) AND
                    (
                        (M2.round1_winner = 2 AND M2.round2_winner = 2) OR
                        (M2.round2_winner = 2 AND M2.round3_winner = 2) OR
                        (M2.round3_winner = 2 AND M2.round4_winner = 2)
                    )
                ) THEN 1
                ELSE 0
            END
        ) AS frequencia_vitoria_dupla_rondas_seguidas
    FROM
        M2
)
/*Vamos selecionar a coluna frequencia_vitoria_dupla_rondas_seguidas da CTE
M3 que contém a frequência calculada*/
SELECT
    M3.frequencia_vitoria_dupla_rondas_seguidas
FROM
    M3;
```

Resultado:



	frequencia_vitoria_dupla_rondas_seguidas
1	34,184