Mestrado em Ciências de Computadores

Construção e análise de uma *datawarehouse* sobre um competição nacional de natação

Realizado por:

Ricardo Ribeiro 201705837

Rita Abreu 201805297

Tatiana Araújo 201805169

Tópicos Avançados em Base de Dados

Junho 2022

1 Introdução

Este trabalho foi realizado no âmbito da cadeira de Tópicos Avançados em Bases de Dados e tem como objetivo analisar uma base de dados. Esta base de dados é composta por vários dados relacionados com a competição nacional de natação de nadadores no nível *Master*.

O primeiro passo do trabalho consistiu em construir uma nova base de dados a partir da base de dados fornecida pelo professor, mas com um esquema *Datawarehouse*. O segundo passo é a análise da base de dados.

2 Análise da Base de dados inicial

A base de dados inicial chamada *db_annp* foi criada por um ficheiro *scrip Python* que lê os ficheiros LXF (lenex format) e gera um *script* SQL para inserir valores na base de dados.

Esta base de dados é contituída por 9 tabelas: *split*, *result*, *swimstyle*, *athlete*, *event*, *club*, *session*, *pool* e *meet* estando connectados pelas chaves primárias.

Na figura 1 encontra-se a esquematização da base de dados.

3 Construção da *Datawarehouse*

Como referido anteriormente, antes de analisar os dados, foi necessário a criação de uma nova base de dados, mas com a estrutura de *datawarehouse*.

Começámos por analisar cada uma das tabelas da base de dados inicial para perceber quais eram as colunas que forneciam informação relevante para mais tarde analisar a base de dados.

Na da tabela *athlete* criámos o atributo *completeName* através da junção do *firstname* com o *lastname*. Nesta tabela retirámos o *swrid*, que é o Id global único dado pelo site *swimrankings.net* e o *clubid*, porque este será colocado na tabela de factos.

Na tabela *club*, retirámos os atributos *type*,*swrid* e *name_en* porque considerámos que não eram atributos essenciais para uma análise futura.

Na tabela *swimstyle*, mantivemos todos os atributos.

Na tabela meet, achamos que os atributos *name_en*, *deadline*, *entrytipe*, *organizer*, *organizer_url*, *result_url*, *startmethod*, *timing*, *type*, *withdrawuntil*, *maxentriesathlete* e *entrystartdate* não eram revelantes para a análise do meet.

Por fim, considerámos que as tabelas *split*, *pool*, *event*, *result* e *session* não nos forneciam informação relevante para a nossa análise.

Depois da análise detalhada de cada um dos atributos passamos para a construção da base de dado com esquema de datawarehouse propriamente dita.

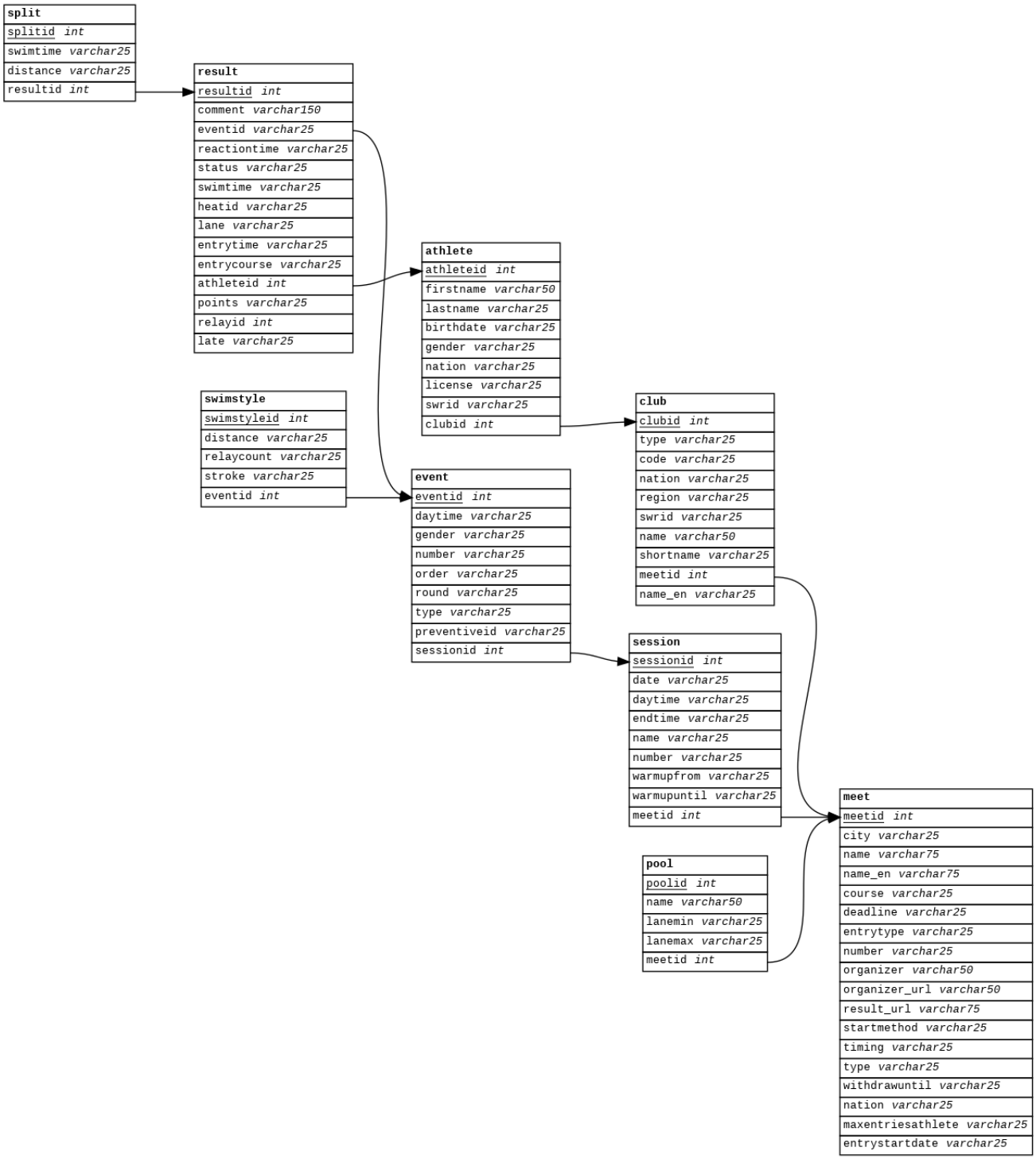

Figure 1: Esquema inicial da base de dados

Assim começámos por criar uma tabela de factos onde colocamos as chaves primárias de *athlete*, *club* , *swimstyle* e *meet*. Para além disto, colocamos 3 variáveis: *swimtime* (obtido através da tabela *swimtime*), *points* (obtido através da tabela *results*) e *poolName*(obtido através da tabela *pool*).

Por fim, criámos as tabelas *club*, *swimstyle*, *athlete* e *meet* como tabelas de dimensões. A tabela de factos está ligada às restantes 4 tabelas usando as chaves *license* (da tabela *athlete*), clubid (da tabela club), *swimstyleid* (da tabela *swimstyle*) e *meetid* (da tabela *meet*).

O esquema datawarehouse está representada na figura 2.

No que diz respeito à criação da *datawarehouse*, tivémos alguns contratempos. Inicialmente, o ficheiro *annp.sql* criado pelo *decode_LXF.py*, funcionava para *MySQL* mas não para *PostgreSQL*. Para funcionar, tivémos que alterar algumas nomenclaturas. Por exemplo, o *"INSERT IGNORE INTO"* teve que ser alterado em todos os inserts para *"ON CONFLICT DO NOTHING"*. Para além disso, tivémos alguns erros que não ocorriam tão frequentemente, mas que impossibilitavam a criação das tabelas. Dois desses erros estavam relacionados a uma coluna com uma palavra reservada (order) na tabela de eventos. Outro exemplo foi o caso do *name.en* ou *organizer.url*, que dava um erro derivado do ".", e, para resolver esse erro, alterámos os pontos para *underscore*. Outro problema que tivémos e que só conseguimos resolver já depois de começar a criação de *queries* e gráficos, foi o facto de apenas conseguirmos inserir um *meet* na *datawarehouse*. Conseguimos resolver o erro ao pesquisar no ficheiro *decode_LXF.py* e descobrir que o *meetid* era fixo (41231234). Para resolver esse problema, incrementamos o *meetid* nesse ficheiro no *loop* resultante de guardar os ficheiros de cada LXF. Depois de alterar o *annp.sql* para funcionar em *PostgreSQL*, criámos também um ficheiro *db_datawarehouse.sql*, que cria as tabelas do datawarehouse e popula-as com os dados do esquema inicial. Para executar os ficheiros, utilizamos *"\i postgres_annp.sql"* e *"\i psql_db_datawarehouse.sql"*.

4 Análise da Base de dados

Com o objetivo de analisar a base dados, utilizámos as ferramentas recomendadas pelo professsor que foram *python* e *PostgreSQL*. Resumidamente, num ficheiro *python* realizamos a conexão à base de dados do *PostgreSQL*, extraímos as informações que queríamos através de uma *query* e, de seguida, representamos essa informação de diferentes formas, maioritariamente sobre a forma de gráficos. Nas subsecções seguintes, vamos explicar detalhamente que informações obtivemos da base de dados e como as representamos.

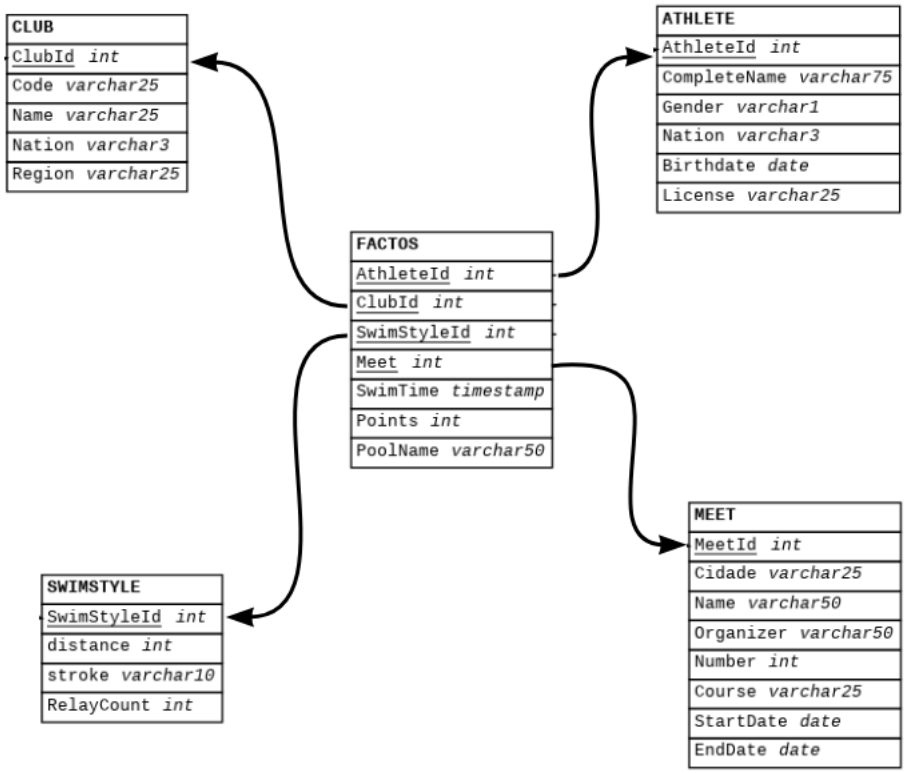

Figure 2: Esquema *datawarehouse* da base de dados

4.1 Considerações gerais de cada meet

Iniciamos a nossa análise por obter algumas informações gerais acerca de cada um dos *meets*. Essas informações estão representadas na tabela 1.

O *meet* com id 41231234 representa o XXII Campeonato Nacional Masters de Verão - OPEN e o 41231235 Troféu Pescada - José Carlos Freitas e Troféu Master ANNP.

Table 1: Tabela com dados de meets

MeetId	Nr de clubes	Nr de Atletas	% Atletas Femininas	% Atletas Masculino
41231234	27	421	35%	65%
41231235	52	225	32%	67%

Para construir esta tabela utilizamos a *query* 1 e a *query* 2

Query 1: Query SQL utilizada para obter o número de clubes por meet.

```
1  select  meet.name as n, count(distinct(code))
2  from club, meet, fact
3  where club.clubid=fact.clubid and meet.meetid = fact.meetid
4  group by n ;
```

Query 2: Query SQL usada para obter o número de atletas por meet, incluindo se são do sexo masculino ou feminino

```
1  select  meet.name as n, count(distinct(fact.athleteid)) as count,
       gender
2  from athlete, meet, fact
3  where fact.athleteid = athlete.athleteid and  meet.meetid = fact.meetid
4  group by rollup (n,gender);
```

4.2 Número de atletas por clube

Para começar a análise da base de dados, achamos pertinente começar por descobrir quantos atletas tinha cada equipa. Esta *query* devolve uma coluna com o código que idêntifica a equipa e outra coluna com o respetivo número de atletas. Esta *query* apenas agrupa os atletas com o mesmo *code* e de seguida conta quantos existem em cada equipa.

Query 3: Query SQL usada para obter o número de atletas por clube.

```
1  SELECT code, COUNT(athleteId) AS N
2  FROM club, fact
3  WHERE club.clubid=fact.clubid
4  GROUP BY ROLLUP (code)
5  ORDER BY N DESC;
```

Com esta *query* construímos o gráfico da figura 3. Neste gráfico conseguimos perceber que a equipa com mais jogadores é a equipa CFP e com menos jogadores é a equipa CAP.

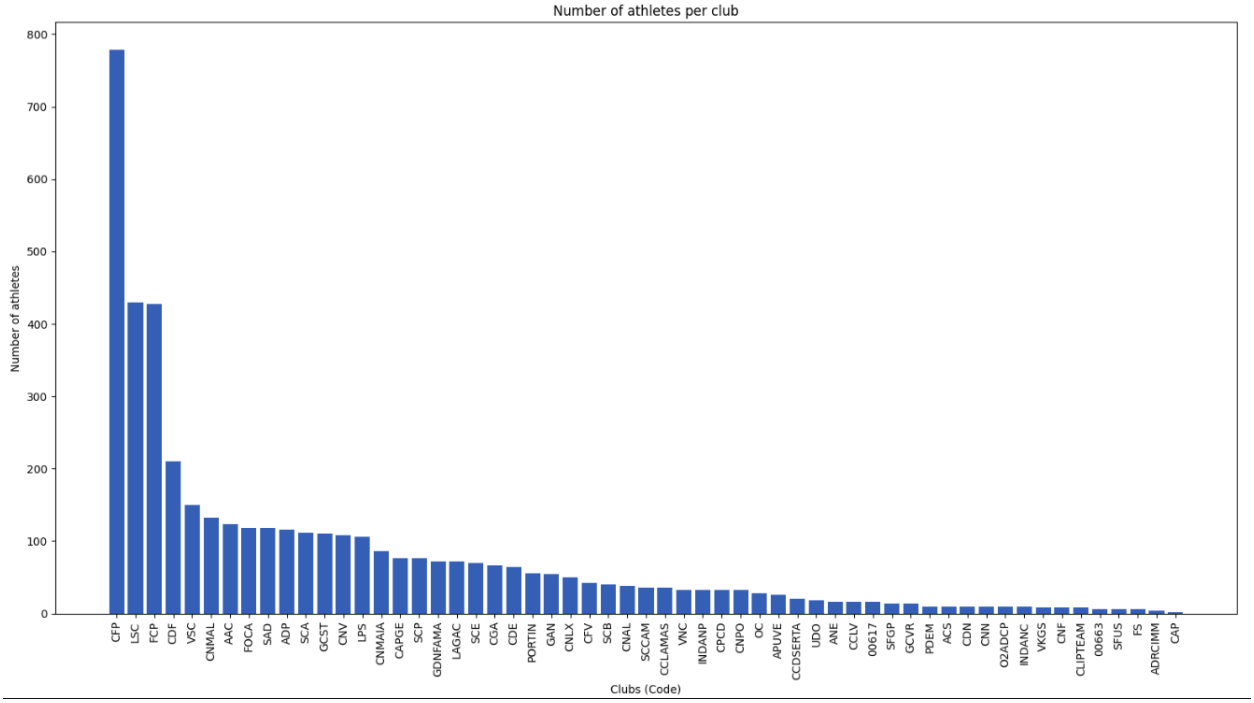

Figure 3: Esquema *datawarehouse* da base de dados

4.3 Idade mínima, Média e Máxima dos atletas por clube

A seguinte *query* devolve uma tabela em que a primeira coluna é o código de cada equipa, a segunda coluna é a idade do atleta mais novo da equipa, a terceira coluna é a idade do atleta mais velho e por fim a última coluna é a idade média dos atletas de cada equipa. Para obter esta tabela selecionamos os atributos indicado anteriormente agrupados pelo código da equipa. Tivémos ainda de ter em atenção que, como temos várias tabelas, precisamos de garantir que estamos a operar sob o mesmo *clubid* assim como com o mesmo *athleteid*. Por essa razão foi necessário adicionar a cláusula *WHERE* da linha 6.

Query 4: Query SQL usada para obter a idade mínima, média e máxima dos atletas por clube

```
1  SELECT code,
2      MIN(EXTRACT(YEAR FROM age(CURRENT_DATE, birthdate::TIMESTAMP))::
           int) AS minAge,
3      Max(EXTRACT(YEAR FROM age(CURRENT_DATE, birthdate::TIMESTAMP))::
           int) AS maxAge,
4      AVG(EXTRACT(YEAR FROM age(CURRENT_DATE, birthdate::TIMESTAMP))::
           int) AS avgAge
5  FROM club, athlete, fact
6  WHERE club.clubid=fact.clubid AND fact.athleteid = athlete.athleteid
7  GROUP BY ROLLUP (code)
8  ORDER BY avgage;
```

De seguida, construímos o gráfico representado na figura 4. Neste gráfico conseguimos perceber que dentro da mesma equipa existe uma grande variedade de idades de atletas.

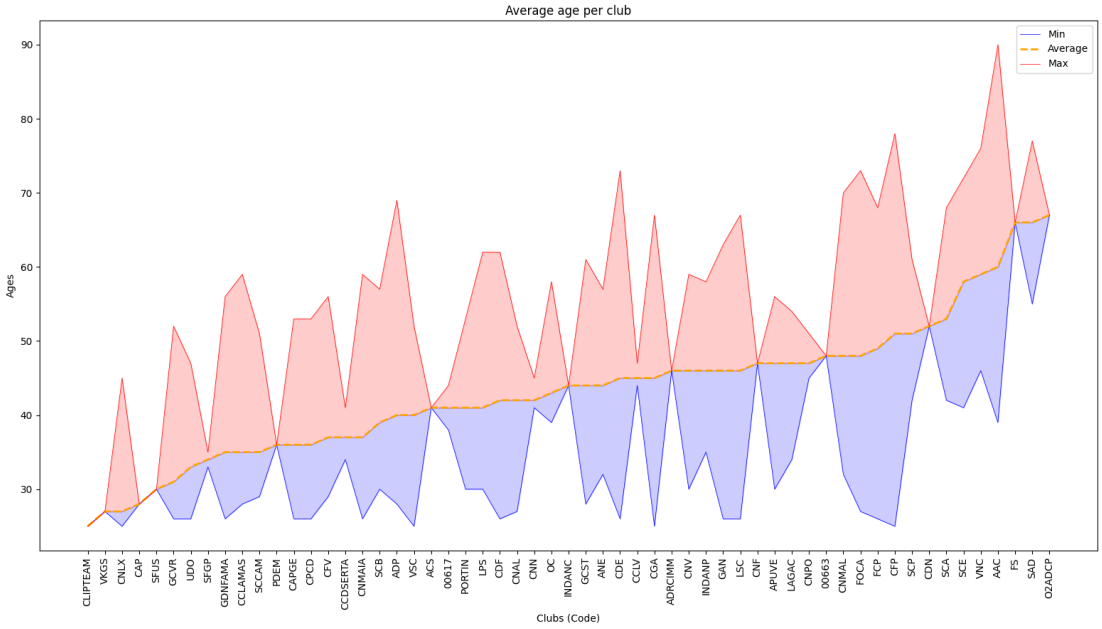

Figure 4: Representação da idade mínima, máxima e média dos atletas por clube

4.4 Os 10 melhores clubes por competição

Para obtermos os melhores clubes para cada competição, consideramos que o melhor clube era aquele que tinha uma maior soma de pontos.

Assim sendo, a *query* seguinte devolve uma coluna com o código da equipa, outra coluna com a soma da pontuação e outra com o *meetid*. Na criação da *query* tivémos em atenção que não podemos considerar os casos em que a pontuação é *NULL* e fizémo-lo na cláusula *WHERE* na linha 3. Nesta cláusula, tivémos ainda de ter a certeza de que o clubid da tabela de factos é o mesmo da tabela *club* assim como o meetid da tabela de factos é o mesmo da tabela *meet*. Como é visível, fizémos uma ordenação descendente para obter os 10 clubes com mais pontos.

Query 5: Query SQL usada para obter os 10 melhores clubles da competição

```
1  SELECT DISTINCT code, SUM(points) AS SUM, fact.meetid
2  FROM club, fact, meet
3  WHERE club.clubid=fact.clubid
4      AND points IS NOT NULL
5      AND meet.meetid = fact.meetid
6  GROUP BY ROLLUP (code, fact.meetid, code)
7  ORDER BY SUM DESC, meetid
```

Por fim, construímos o seguinte gráfico de barras que mostra que o melhor clube no torneio XXII Campeonato Nacional Masters de Verão - OPEN foi o CFP e no torneio Troféu Pescada - José Carlos Freitas e Troféu Master ANNP foi o clube FOCA. Este

gráfico está representado na figura 5.

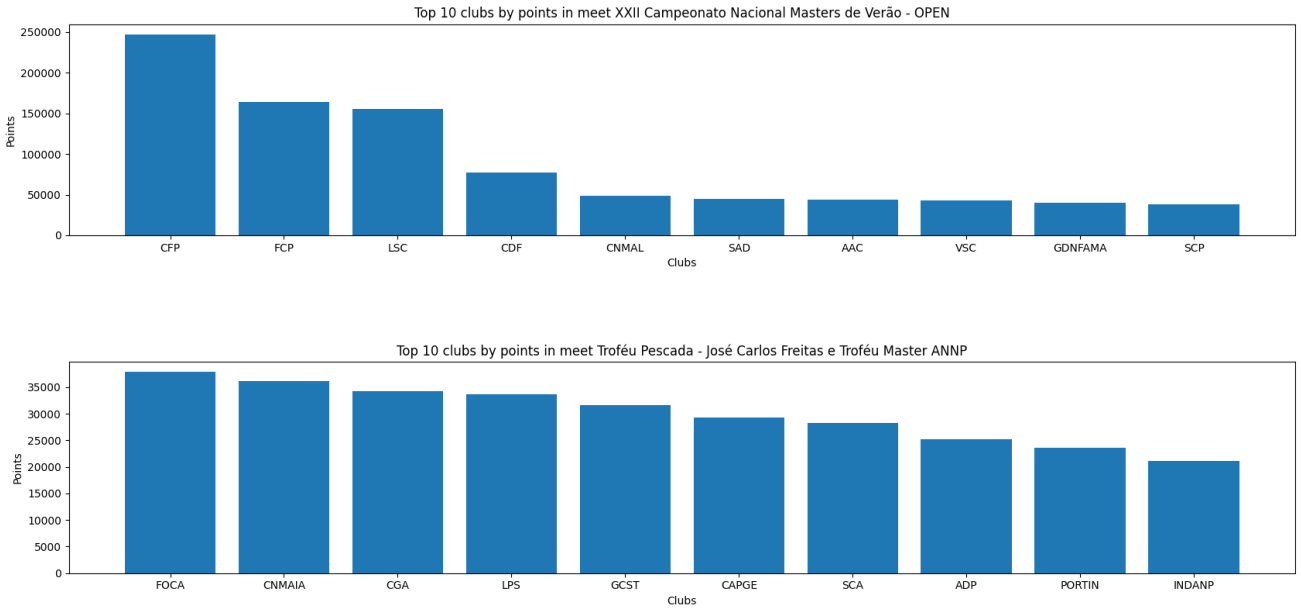

Figure 5: Grafico que representa os 10 melhores clubes por competição

4.5 Pontuação média por estilo de natação e género

Também consideramos importante avaliar como o estilo de natação pode interferir com a pontuação dos atletas. Por essa razão criamos a seguinte *query* que devolve uma coluna para o estilo de natação, outra com a pontuação média e outra com o género.

Query 6: Query SQL usada para obter a pontuação média por estilo de natação e género

```
1  SELECT stroke, AVG(fact.points) AS average, gender
2  FROM swimstyle, fact, athlete
3  WHERE swimstyle.swimstyleid = fact.swimstyleid
4      AND fact.athleteid = athlete.athleteid
5  GROUP BY ROLLUP (stroke,gender)
6  ORDER BY stroke, gender;
```

Com a *querie* que obtivemos construímos o gráfico da figura 6. Neste gráfico é evidente que a pontuação entre rapazes e raparigas é bastante similar, contudo as raparigas destacam-se nos estilos *BREAST* e MEDLEY e o rapazes destacam-se nos restantes estilos de natação.

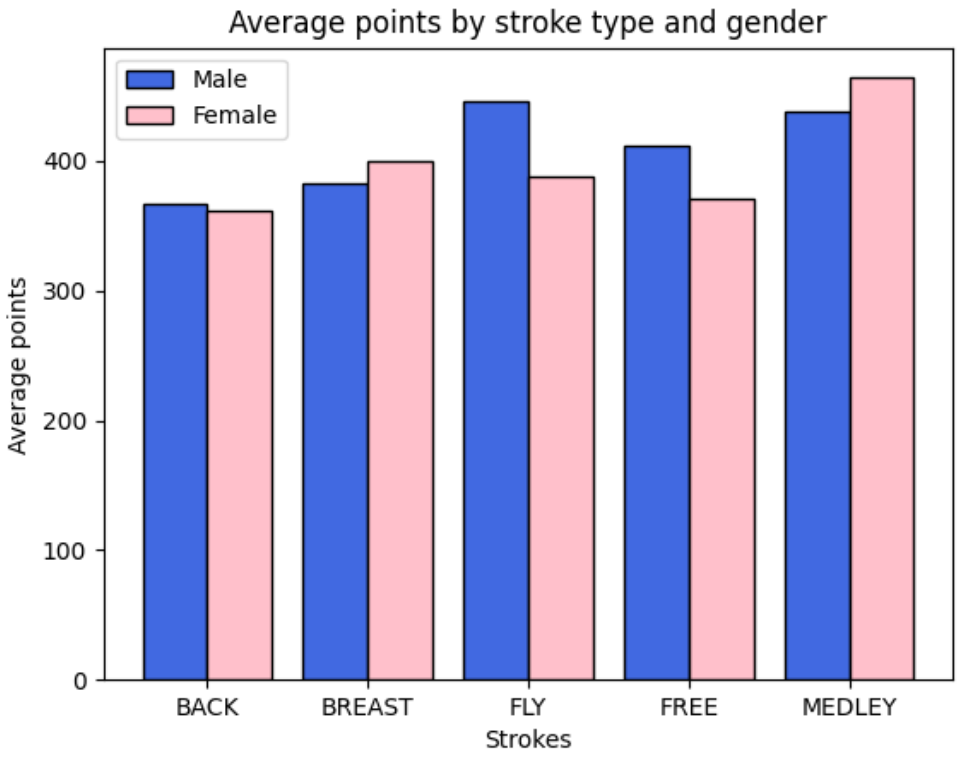

Figure 6: Gráfico que demonstra a pontuação média por estilo de natação e género

4.6 Tempo médio por estilo de natação e género

De seguida, fomos verificar de que forma os tempos são afetados. Esta análise será muito parecida com a análise anterior, mas desta vez temos em consideração o tempo e não a pontuação. Criamos a seguinta *query* que devolve uma coluna com os vários estilos de natação, outra com o tempo média e outra com o género.

Query 7: Query SQL usada para obter o tempo médio por estilo de natação e género

```
1  SELECT stroke, AVG(EXTRACT(epoch FROM swimtime::TIME)*100) AS average,
       gender
2  FROM swimstyle, fact, athlete
3  WHERE swimstyle.swimstyleid = fact.swimstyleid
4      AND fact.athleteid = athlete.athleteid
5  GROUP BY ROLLUP (stroke, gender)
6  ORDER BY stroke, gender;
```

Com a *querie* que obtivemos construímos o gráfico da figura 7. Como era de esperar, obtivemos um gráfico bastante parecido com o anterior.No entanto desta vez as raparigas destacam-se em todos os estilos de natação exceto no estilo MEDLEY.

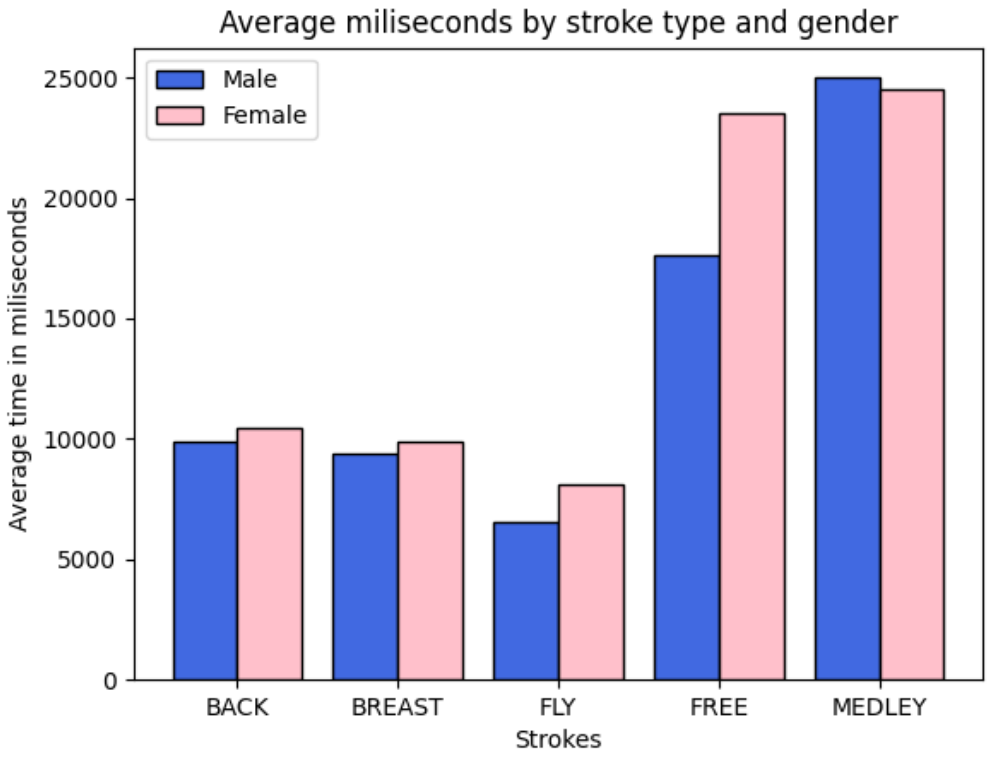

Figure 7: Representação do tempo médio por estilo de natação e género

4.7 Análise por clube

Para fazer uma análise por clube decidimos que seria bom o utilizador conseguir consultar algumas estatísticas de um clube específico que pretende. Para isto, criámos um script *python*, onde questionamos o utilizador sobre qual é o club que pretende consultar e de seguida apresentamos várias estatisticas acerca desse clube.

4.7.1 Distribuição de géneros por clube

A análise da distribuição de géneros por clube é bastante simples, apenas precisamos de criar uma *query* para saber quantos atletas do sexo masculino e do sexo feminimo uma equipa tem.

Nesta query apenas agrupamos os atletas pelo gender e contamos quantos atletas temos. Abaixo encontra-se uma *query* exemplificativa, neste caso o utilizador pretende saber mais informações sobre o clube do Futebol Clube do Porto (*FCP*).

Query 8: Query SQL usada para obter o número de atletas do sexo feminino em cada equipa

```
1    select count (fact.athleteid) as count, gender
2    from athlete, club, fact
3    where fact.athleteid = athlete.athleteid
4        and fact.clubid = club.clubid
```

```
5         and club.code = 'FCP'
6     group by rollup (gender);
```

De seguida, construímos um gráfico circular para esta *query*, representado na figura 8. Como se pode ver 39.7% dos atletas do clube *FCP* são mulheres e 60.3% são homens.

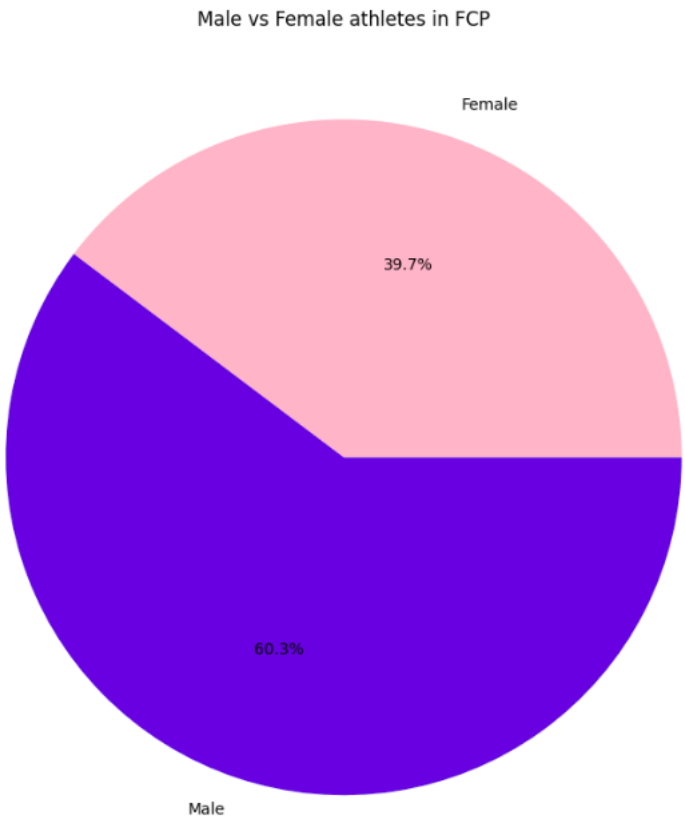

Figure 8: Distribuição de géneros para o clube FCP

4.7.2 5 melhores jogadores de um clube

Desta vez quisemos obter algumas informações acerca dos melhores atletas de uma determinada equipa. Consideramos que um atleta é melhor do que outro se as pontuações médias forem maiores. Assim, construímos a seguinte *query* que nos devolve 5 atletas.

Na primeira coluna da tabela temos o nome completo do atleta e as suas pontuações mínimas, médias e máximas. Como é possivel observar, esta query já é bastante mais complexa do que as anteriores. Na *sub-query* que dá origem à tabela *newTab*, estamos a selecionar o nome de todos os atletas, o *license* e as suas pontuações (mínima, máxima e média). De seguida filtramos aqueles que competem pela equipa que procuramos e ordenamos por ordem descendente.

Query 9: Query SQL usada para obter os 5 melhores jogadores de um clube.

```
1   SELECT newTab.completeName, newTab.min AS minimum, newTab.avg AS
        average, newTab.max AS maximum
2   FROM
3       (SELECT DISTINCT fact.athleteid, license, completeName, MIN(points)
            AS MIN, AVG(points) AS AVG, MAX(points) AS MAX
4       FROM fact, athlete, club
5       WHERE points IS NOT NULL
6           AND fact.athleteid = athlete.athleteid
7           AND fact.clubid = club.clubid
8       GROUP BY fact.athleteid, code, license,completeName)newTab,
9       club,
10      fact
11  WHERE newTab.athleteid = fact.athleteid
12      AND club.clubid = fact.clubid
13      AND club.code = '" + club + "'
14  GROUP BY (newTab.avg, newTab.max, newTab.min, newTab.completeName)
15  ORDER BY AVG DESC
16  LIMIT 5;
```

Mais uma vez, construímos um gráfico para a *query* que está representado na figura 4.7.2. Neste gráfico conseguimos perceber que o melhor atleta do clube FCP é o António Aderito Chaves, com pontuações comprendidas aproximadamente entre 740 e 820 pontos e com pontuação média de aproximadamente 770 pontos. Este atleta mostra uma grande consistência nas suas pontuações, ao contrário do segundo melhor atleta do FCP, José Pedro Soares, que demonstra ser muito menos consistente, visto que as suas pontuações variam entre 600 e 850 pontos, aproximadamente. Como terceiro, quarto e quinto melhores jogadores temos a Carla Santa Bárbara, o João Filipe Carvalho e a Gisela Barbosa Coutinho, respetivamente.

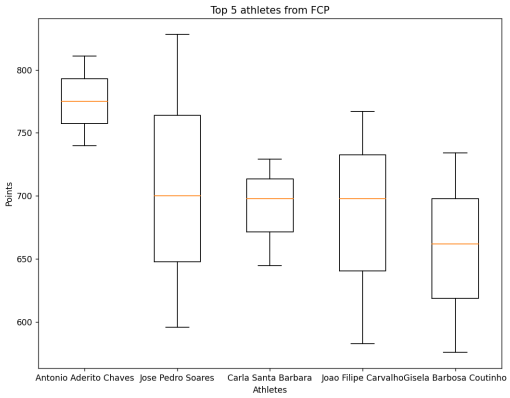

Figure 9: Representação dos 5 melhores jogadores do clube FCP

4.8 Análise por atleta: Posição no *ranking* de um jogador

Mais uma vez, achamos relevante fazer uma análise mais "interativa" e por essa razão

criamos um *script python* que dado um *license* de um atleta, este mostra várias informações de um atleta.

4.8.1 Posição no *ranking* de um jogador

Começamos por indicar a posição do jogador no ranking da competição e alguns dados sobre ele. Neste caso consideramos que os jogadores com mais pontos somados são melhores.

Assim, criamos esta *query* que devolve o nome completo do jogador, soma total dos seus pontos, pontuação máxima, mínima e média e equipa a que pertence.

A posição do ranking não será mostrada sobre a forma de gráfico, mas diretamente no *layout* apresentado. Por exemplo o atleta com a licensa número 208967 encontra-se na posição 237 do ranking.

Query 10: Query SQL usada para obter a posição no ranking de um atleta.

```
1  SELECT license, completeName, sum(points) AS SUM, AVG(points) AS AVG,
       min(points) AS MIN, MAX(points) AS MAX, code
2  FROM fact, athlete, club
3  WHERE points IS NOT NULL AND fact.athleteid = athlete.athleteid AND
       fact.clubid = club.clubid
4  GROUP BY athlete.license, code, license, completeName
5  ORDER BY SUM DESC;
```

4.8.2 Número de vezes que um atleta fez uma prova de determinada distância

Com esta *query* obtemos uma tabela com 2 colunas: 1 para os vários tipos de distâncias e outra para o número de vezes que o atleta fez essa prova.

Query 11: Query SQL usada para obter o número de vezes que um atleta fez uma prova de determinada distância

```
1  SELECT distance, athleteId, count(distance) AS N
2  FROM fact, swimstyle
3  WHERE fact.swimstyleid = swimstyle.swimstyleid
4      AND athleteId= '"+ str(athleteid) +"'
5  GROUP BY (athleteId, distance)
```

.

Utilizamos o jogador com o *license* 208967 como exemplo para a criação do gráfico da figura 10. Este jogador fez 2 provas de 100 metros, 6 de 200 metros e 2 de 400 metros.

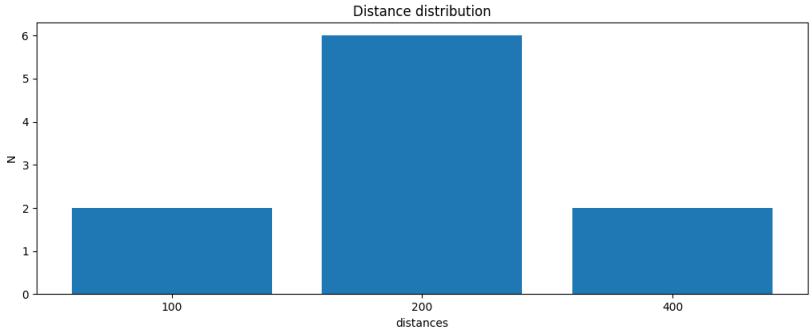

Figure 10: Número de vezes que um atleta fez uma prova de determinada distância

4.8.3 Distribuição dos estilos de natação utilizados pelo atleta

Achamos ainda que seria interessante ver quais eram os estilos preferidos de natação de cada jogador. Assim sendo, criámos a sequinte *query* que nos devolve uma coluna para o estilo de natação e o respetivo número de vezes que cada atleta escolheu esse estilo.

Query 12: Query SQL usada para obter a Distribuição dos estilos de natação utilizados pelo atleta.

```
1    SELECT stroke, license, count(stroke) AS N
2    FROM fact, swimstyle, athlete
3    WHERE fact.swimstyleid = swimstyle.swimstyleid
4        AND athlete.athleteid = fact.athleteid
5        and athlete.athleteid = "+ str(athlete_id) +"
6    GROUP BY (license, stroke)
```

No gráfico da figura 11 podemos ver que o atleta com a licensa número 208967 preferiu o estilo *MEDLEY* e o *BACK* 40% das vezes e 20% das vezes preferiu o estilo *BREAST*.

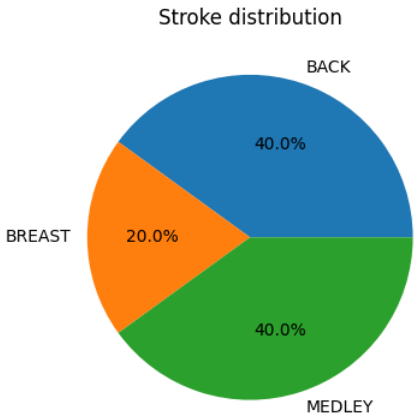

Figure 11: Distribuição dos estilos de natação utilizados pelo atleta

4.8.4 Distribuição da pontuação

Achamos ainda importante mostrar o intervalo por onde se distribuem os pontos para um determinado atleta. Esta *query* é constituída por uma *subquery* que cria uma tabela com as colunas *athleteid*, *license*, *completename*, *min*, *avg* e *max*. Depois apenas precisamos de selecionar o jogador pretendido através do *athleteid*.

Query 13: Query SQL usada para obter a distribuição da pontuação do atleta

```
1   SELECT newTab.completeName, newTab.min AS minimum, newTab.avg AS
        average, newTab.max AS maximum, newTab.license
2   FROM
3       (SELECT DISTINCT fact.athleteid, license, completeName, MIN(points)
            AS MIN, AVG(points) AS AVG, MAX(points) AS MAX
4       FROM fact, athlete, club
5       WHERE points IS NOT NULL
6           AND fact.athleteid = athlete.athleteid
7           AND fact.clubid = club.clubid
8       GROUP BY fact.athleteid, code, license,completeName) newTab,
9       club,
10      fact
11  WHERE newTab.athleteid = fact.athleteid
12      AND club.clubid = fact.clubid
13      AND newTab.athleteid = "+ str(athleteid) +"
14  GROUP BY (newTab.max, newTab.avg, newTab.min, newTab.completeName, code
        , license);
```

Mais uma vez utilizamos como exemplo o atleta com a licensa número 208967 e obtivemos o gráfico da figura 12. Ao analisar este gráfico percebemos que as pontuações deste atleta variam aproximadamente entre 220 e 320, sendo que a pontuação mínima é 220, a máxima é 320 e a média é 270 pontos.

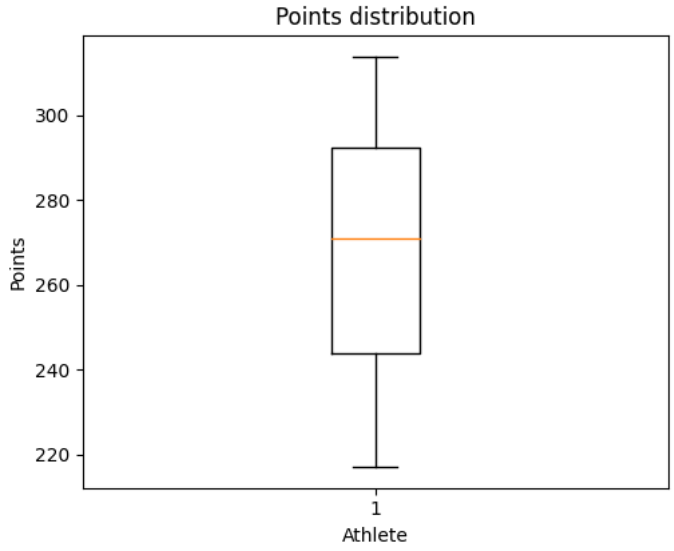

Figure 12: Distribuição da pontuação

4.8.5 Tempo médio para cada estilo de natação

Por fim, vamos consultar quanto tempo um atleta demora em média em cada estilo de natação. Para isso constuímos a *query* seguinte que nos devolve uma tabela com 3 colunas: a primeira para o tipo de estilo de natação, a segunda com a média do tempo e a terceira com o *athleteid*.

Query 14: Query SQL usada para obter a distribuição da pontuação do atleta

```
1  SELECT stroke, AVG(EXTRACT(epoch FROM swimtime::TIME)*100) AS average,
       athlete.athleteid
2  FROM swimstyle, fact, athlete
3  WHERE swimstyle.swimstyleid = fact.swimstyleid
4      AND fact.athleteid = athlete.athleteid
5      AND athlete.athleteid = "+ str(athlete_id)+"
6  GROUP BY stroke, gender, athlete.athleteid
7  ORDER BY stroke, gender;
```

Finalmente, obtemos o gráfico da figura 13 para o atleta com a licensa número 208967. Pela análise do gráfico podemos concluir que o estilo de natação em que este atleta é mais rápido é o *BACK*.

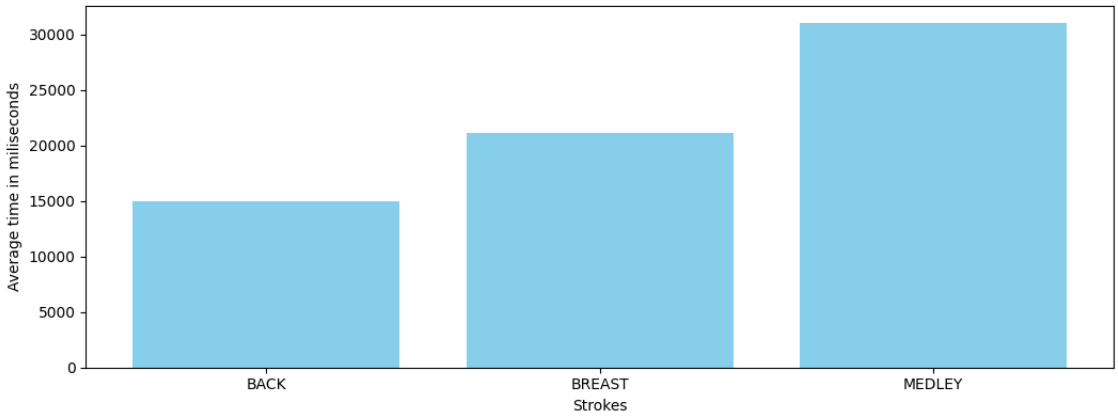

Figure 13: Distribuição da pontuação

4.9 Análise por escalões

Nesta secção iremos analisar a nossa base de dados por escalão. Definimos os escalões com recurso ao regulamento de competições nacionais da categoria *Master* da Federação Portuguesa de Natação. A divisão dos atletas por escalão são apresentados na tabela 2.

Table 2: Tabela de Escalões

Grupo	Escalão Etário	Ano de Nascimento
Master A	25 – 29	94-90
Master B	30 – 34	89-85
Master C	35 – 39	84-80
Master D	40 – 44	79-75
Master E	45 – 49	74-70
Master F	50 – 54	69-65
Master G	55 – 59	64-60
Master H	60 – 64	59-55
Master I	65 – 69	54-50
Master J	70 – 74	49-45
Master K	75 – 79	44-40
Master L	80 – 84	39-35
Master M	85 – 89	34-30
Master N	90 – 94	29-25
Master O	95 - 99	24-20

4.9.1 Análise da pontuação por género e por escalão

Esta *query* é constituída por 2 *subqueries*. A *query* mais interna calcula a idade de cada atleta e cria as colunas *gender*, *atheleteid* e *age*.

De seguida adicionamos à tabela que tínhamos obtido na *query* anterior as colunas *points*, gender, e ainda uma coluna chamada *age_range* que indica a que escalão pertence

um atleta (de acordo com a tabela 2). s Por fim, agrupamos, com o auxílio da função *rollup* o *age_range* e o gender e obtivemos uma tabela com o escalão a pontuação média e o género.

Query 15: Query SQL usada para obter o número de pontos por género e por escalão

```
1   select newTab2.age_range, AVG(newTab2.points) as avg_points, gender
2   from
3       (select
4           fact.athleteid,
5           points,
6           gender,
7           case
8               when newTable.age between 25 and 29 then 'A'
9               when newTable.age between 30 and 34 then 'B'
10              when newTable.age between 35 and 39 then 'C'
11              when newTable.age between 40 and 44 then 'D'
12              when newTable.age between 45 and 49 then 'E'
13              when newTable.age between 50 and 54 then 'F'
14              when newTable.age between 55 and 59 then 'G'
15              when newTable.age between 60 and 64 then 'H'
16              when newTable.age between 65 and 69 then 'I'
17              when newTable.age between 70 and 74 then 'J'
18              when newTable.age between 75 and 79 then 'K'
19              when newTable.age between 80 and 84 then 'L'
20              when newTable.age between 85 and 89 then 'M'
21              when newTable.age between 90 and 94 then 'N'
22              when newTable.age between 95 and 99 then 'O'
23              END as age_range,
24              newTable.age
25              from
26                  (select gender, fact.athleteid, EXTRACT(year FROM age(
                        current_date, birthdate::timestamp)) :: int as age
27                      from athlete, fact
28                      where fact.athleteid = athlete.athleteid)newTable,
29              fact
30  where newTable.athleteId = fact.athleteId and points is not null
31  order by age)newTab2
32  group by rollup(newTab2.age_range, gender)
33  order by newTab2.age_range, gender;
```

No gráfico da figura 4.9.1 conseguimos visualizar uma linha horizontal que representa a pontuação média dos jogadores no geral.

A pontuação feminina é representada a cor de rosa e é possível perceber que as atletas entre o escalão C e o escalão G são as que têm pontuações mais elevadas. A partir do escalão G as suas pontuações começam a decrescer acentuadamente.

Em relação ao escalão masculino verificamos que entre o escalão A e o escalão I é

onde os atletas se destacam, é relevante destacar as categorias A e H que apresentam uma pontuação bastante elevada.

Comparando os 2 sexos é evidente que o escalão masculino apresenta melhores pontuações para todos os escalões exceto para os escalões A, B e E, por margens curtas.

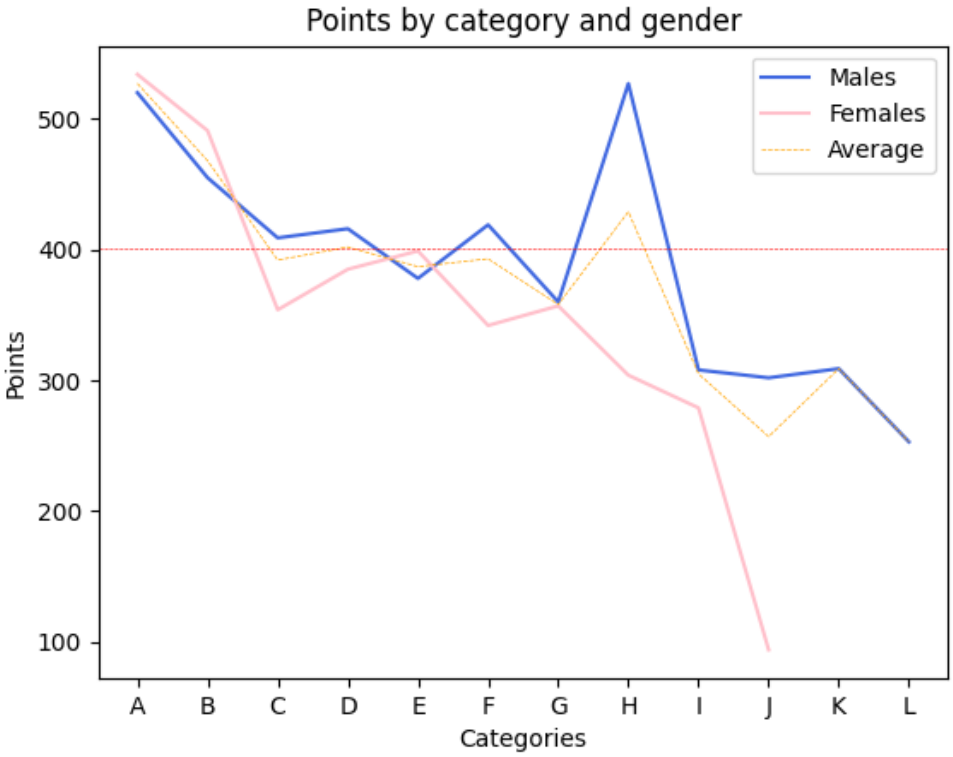

Figure 14: Gráfico que representa a pontuação por género e por escalão

4.9.2 Análise dos tempo por género, por escalão e por *meet*

Esta *query* é idêntica à anterior, mas desta vez queremos obter os tempos mínimos por cada *meet*.

Assim, fizemos a *query* abaixo que é composta por 2 subqueries. A *query* mais interna de todas cria as colunas *gender*, *atheleteid*, *meetid* e *age*.

De seguida adicionamos á tabela que tínhamos obtido na *query* anterior as colunas *meetid*, *points*, gender, e ainda uma coluna chamada *age_range* que indica a que escalão pertence um atleta (de acordo com a tabela 2).

Por fim, agrupamos, com o auxilio da função *cube* o *age_range*, o *meetid* e o gender e obtivemos uma tabela com o escalão a tempo mínimo de natação e o género.

Query 16: Query SQL usada para obter o tempo por género e por escalão

```
1   select newTab2.age_range, MIN(newTab2.swimTimeMiliseconds) as min_Time,
        gender, newTab2.meetid
2   from
3       (select fact.meetid, fact.athleteid, (EXTRACT(epoch FROM swimtime::
            time)*100) as swimTimeMiliseconds, gender,
4       case
5           when newTable.age between 25 and 29 then 'A'
6           when newTable.age between 30 and 34 then 'B'
7           when newTable.age between 35 and 39 then 'C'
8           when newTable.age between 40 and 44 then 'D'
9           when newTable.age between 45 and 49 then 'E'
10          when newTable.age between 50 and 54 then 'F'
11          when newTable.age between 55 and 59 then 'G'
12          when newTable.age between 60 and 64 then 'H'
13          when newTable.age between 65 and 69 then 'I'
14          when newTable.age between 70 and 74 then 'J'
15          when newTable.age between 75 and 79 then 'K'
16          when newTable.age between 80 and 84 then 'L'
17          when newTable.age between 85 and 89 then 'M'
18          when newTable.age between 90 and 94 then 'N'
19          when newTable.age between 95 and 99 then 'O'
20      END as age_range,
21      newTable.age
22          from (select fact.meetid, gender, fact.athleteid, EXTRACT(year
                FROM age (current_date, birthdate::timestamp)) ::
                int as age
23                  from athlete, fact, meet
24                  where fact.athleteid = athlete.athleteid and fact.
                        meetid = meet.meetid)newTable,
25          fact
26  where newTable.athleteId = fact.athleteId and swimtime != '00:00:00'
        and newTable.meetid = fact.meetid
27  order by age)newTab2 where newTab2.swimTimeMiliseconds != 0
```

```
28      group by cube (newTab2.age_range, gender, newTab2.meetid)
29      order by newTab2.age_range, gender;
```

Por fim, criamos o gráfico da figura 15. Neste gráfico é facilmente observado que no geral a competição do XXII Campeonato Nacional Masters de Verão - OPEN teve melhores tempos do que a competição Troféu Pescada José Carlos Freitas e Trofeu Master ANNP para ambas as categorias(masculina e feminina).

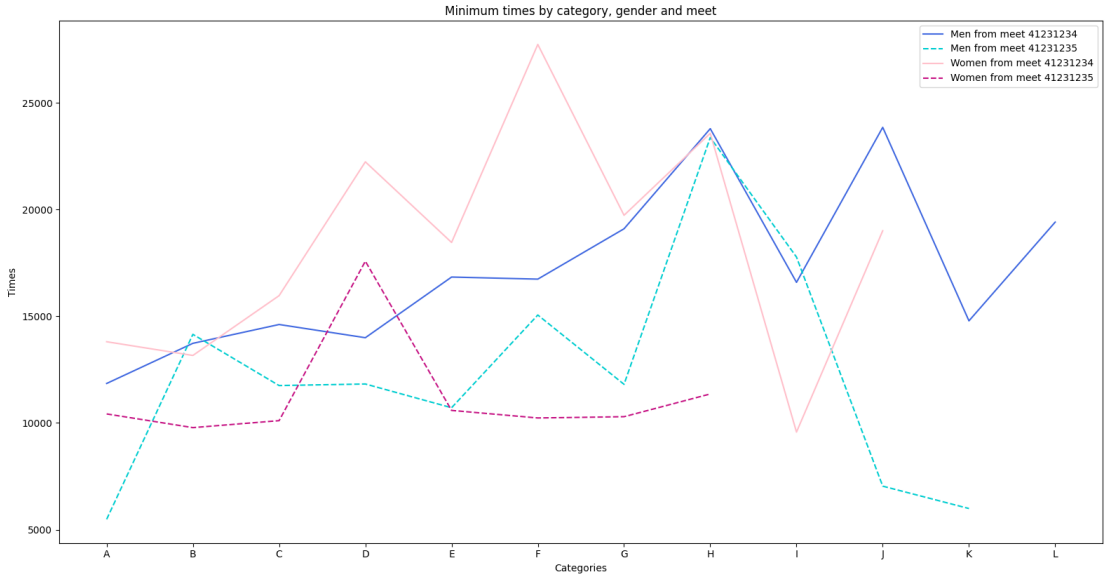

Figure 15: Gráfico que dá os tempo por género, por escalão e por *meet*

5 Conclusão

Apesar de todos os elementos do grupo terem conhecimentos sobre SQL, com este projeto foi possivel aprofundar os nossos conhecimentos. Para além disto, este projeto deu-nos ainda a oportunidade de trabalhar com o *PostgreSQL* que demonstrou ser uma ferramenta bastante útil.

Ao longo do trabalho deparamo-nos com algumas dificuldades. A primeira dificuldade prendeu-se com a construção da *datawarehouse*, isto porque incialmente não estavamos a conseguir perceber como a devíamos estruturar. Depois de alguma persistência e ajuda do professsor conseguimos ultrapassar este problema.

Infelizmente, apenas pouco tempo antes da data limite de entrega do trabalho conseguimos adicionar mais do que um *meet* à nossa base de dados, o que nos levou a não conseguir fazer uma análise suficientemente completa de vários *meets*. Este problema foi uma grande entrave no desenvolvimento do trabalho.

Em suma, apesar dos contratempos que surgiram, consideramos que conseguimos fazer uma boa análise da base de dados.