Faculty of Sciences of the
University of Porto

Department of Computer Science

Advanced Topics in Databases

Practical Assignment

Andreia Genro Aníbal Silva Rui Fernandes

June, 2022

Abstract

This report describes the practical assignment of the Advanced Topics in Databases course.

This practical assignment consists in creating a data warehouse and conducting data analysis on it, as well as creating graphical reports using the Python library `matplotlib`.

In this report, we briefly describe our approach to the problem and discuss the decisions we made.

Contents

List of Figures

1 Introduction

The data warehouse contains data from national swimming competitions at the master level (*i.e.* class of competitive swimming for swimmers 25 years and older), namely Troféu Pescada 2021 and the Summer 2021 Championship.

Structure of the Report

The remainder of the report is structured as follows:

- In Section 3, **Data Analysis & Visualization**, we provide some insight into the data
- Finally, Section 4, **Conclusions & Future Work**, concludes the report and suggests remarks for future work.

2 Data Model

In this section, we describe the data tables contained in the data warehouse.

Some modifications were made in the original script. This was mainly motivated due to the fact that same athletes and clubs had different `ids` for different meets, when in our perspective they should be uniquely identified across tournaments. They were defined as:

- `athleteid` = `firstname` + `lastname` + `birthdate` + `inc_id`, where the `inc_id` increments when two athletes share the same first and last name and the birthdate;
- `clubid` = `code` + `nation` + `region`;
- `resultid` = `meetid` + `resultid`;
- `swimstyleid` = `distance` + `relaycount` + `stroke`;
- `eventid` = `meetid` + `eventid`;
- The `license`, which originally was `meetid` + `clubid` + `idx` was replaced by `athleteid` + `meetid` because the same athlete could have changed between teams for different tournaments.

Regarding our data model, we built two fact tables, one that gathers information regarding a club in a tournament, while another gathered information of a given athlete in a tournament. Our schema is depicted in Figure 1.

This Figure shows two fact tables, one related to overall statistics of a team in a tournament, while the other containing the overall statistics of an athlete. Regarding the first table, *Club de factos*, we grouped our data by `meetid`, `clubid` and `swimstyleid` to extract the following statistics:

- `number_of_players` - the number of players;
- `average_age` - the average age of players;
- `total_points` - the total points a given team had in a tournament;
- `total_distance` - the total distance all the players swam in a team;
- `average_swimtime` - the average swimtime all the players swam.

Regarding the second fact table, *Athletes de factos*, we grouped our data by `athleteid`, `meetid` and `from_club` in order to extract the following statistics:

- `average_swimtime` - The average swimtime a given athlete swam

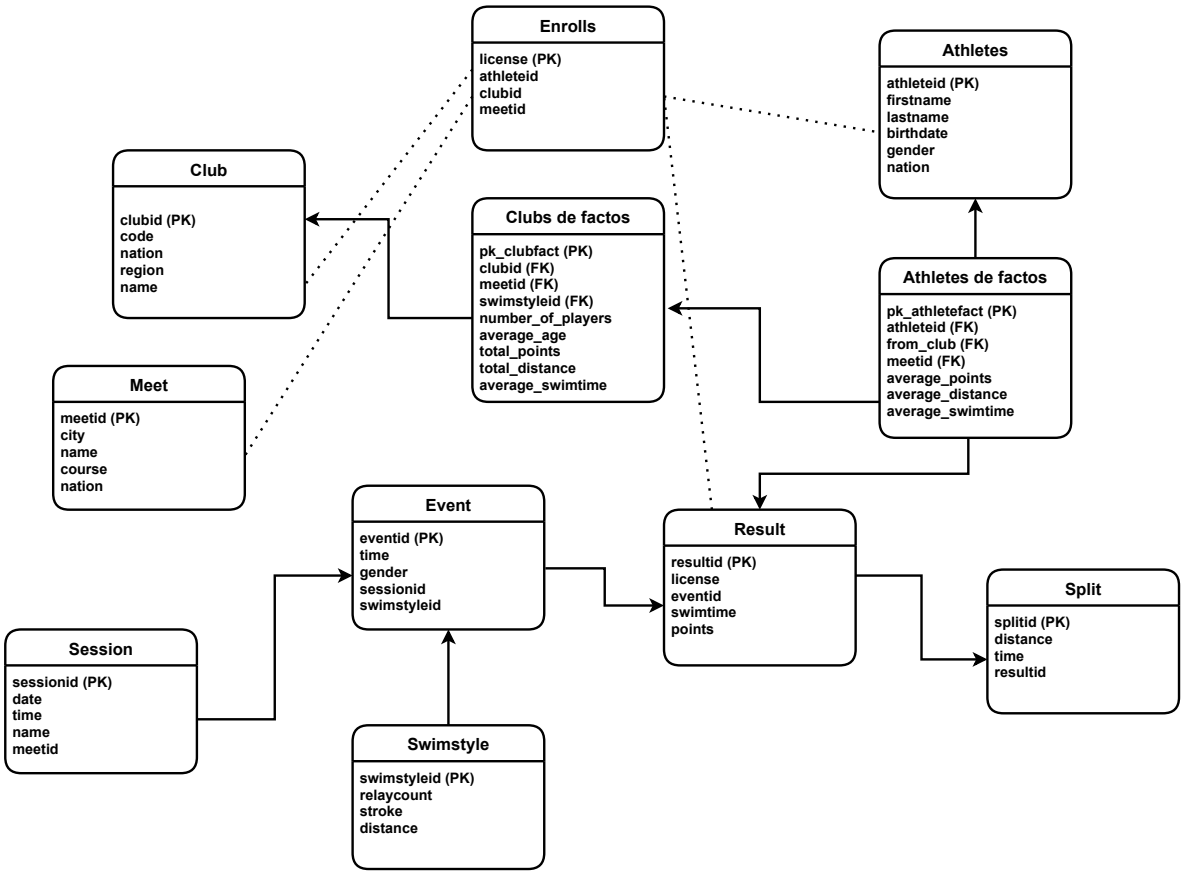

Figure 1: Datawarehouse schema.

– `average_points` - The average points an athlete had

– `average_distance` - The average distance an athlete swam

for a given tournament.

3 Data Analysis & Visualization

3.1 Number of Athletes by Age

To determine to determine the average age of the athletes, we can run the following SQL query:

```
SELECT AVG(age(birthdate))
FROM annp_final.athlete;
```

From this, we can see that the average age of the athletes is 46 years, 6 months and 31 days.

We can also determine who's the youngest athlete by running the following SQL query:

```
SELECT *
FROM annp_final.athlete
ORDER BY age(birthdate) ASC
LIMIT 1;
```

- **Name:** Ana Mónica Eloi
- **Gender:** F
- **Birthdate:** 29/12/1996
- **Age:** 25 years

On the other hand, we can learn information about the oldest athlete by running the following SQL query:

```
SELECT *
FROM annp_final.athlete
ORDER BY age(birthdate) DESC
LIMIT 1;
```

- **Name:** Virgílio Zacarias Costa
- **Gender:** M
- **Birthdate:** 21/07/1931
- **Age:** 90 years

Finally, to determine the number of athletes by age, we can run the following SQL query using the PostgreSQL's built-in `age` function:

```
SELECT COUNT(*), EXTRACT(YEAR FROM age(birthdate)) AS age
FROM annp_final.athlete
GROUP BY age
ORDER BY age ASC;
```

We can then plot the result, as illustrated in Figure 3.

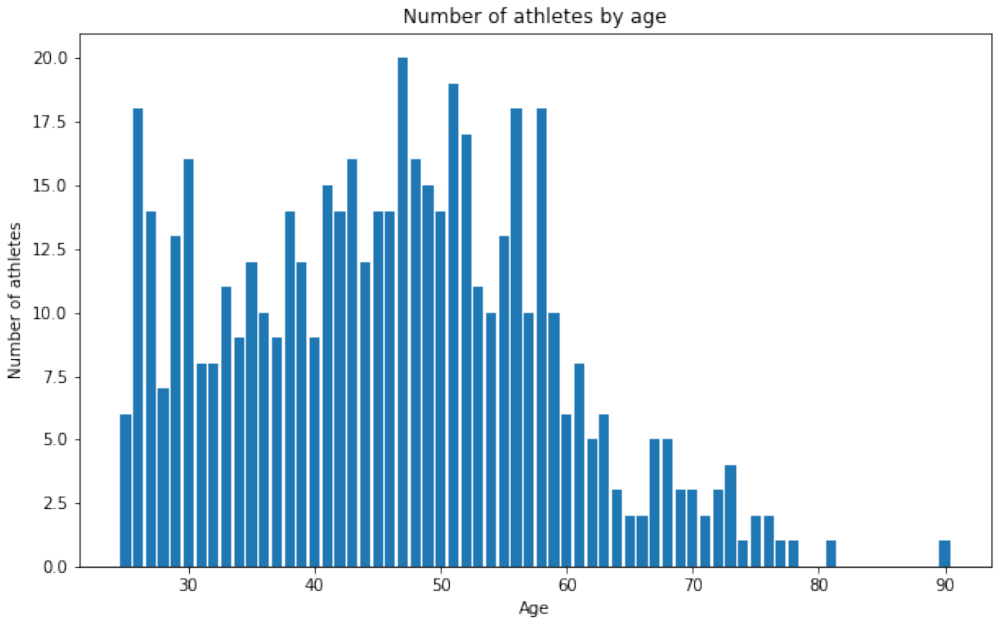

Figure 2: Number of athletes by age

3.2 Number of Athletes by Nation

To determine the number of athletes by nation, we can run the following SQL query:

```
SELECT COUNT(*) nationCount, nation
FROM annp_final.athlete
GROUP BY nation
ORDER BY nationCount ASC;
```

We can then plot the result, as illustrated in Figure 3.

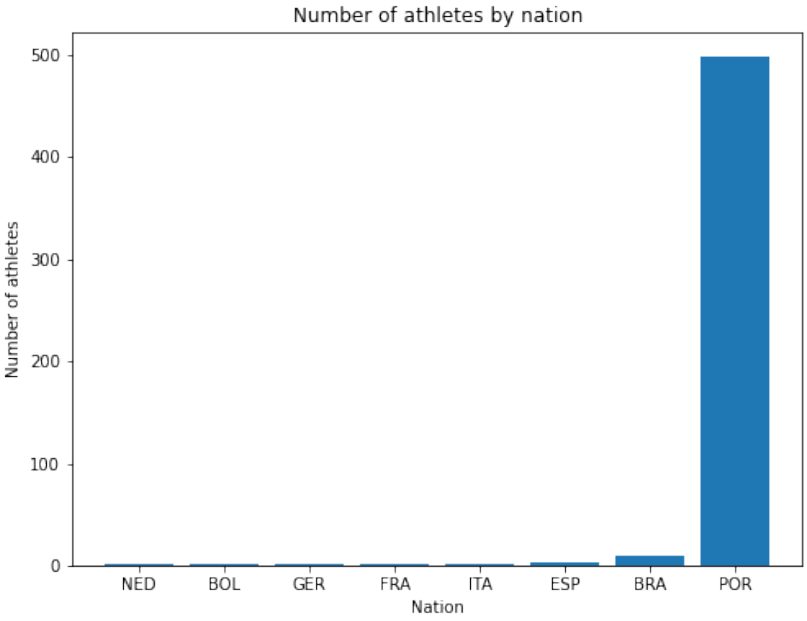

Figure 3: Number of athletes by nation

To have another perspective, we can also plot in a pie chart, as illustrated in Figure 4.

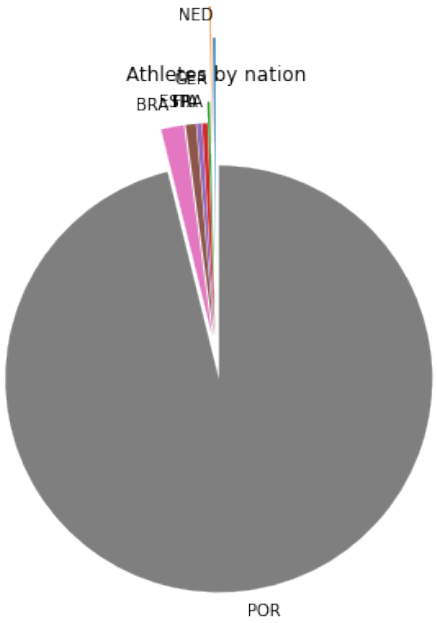

Figure 4: Number of athletes by nation

3.3 Number of Athletes by Gender

To determine the number of athletes by gender, we can run the following SQL query:

```
SELECT COUNT(*), gender
FROM annp_final.athlete
GROUP BY gender;
```

We can then plot the result, as illustrated in Figure 5.

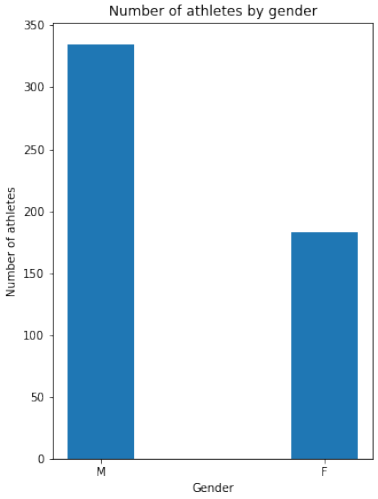

Figure 5: Number of athletes by gender

We can also plot this in a pie chart, as illustrated in Figure 6.

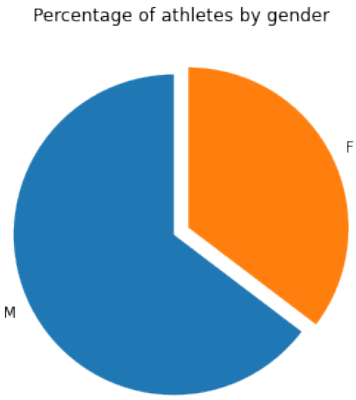

Figure 6: Percentage of athletes by gender

3.4 Number of Events by Gender

To determine the number of events by gender, we can run the following SQL query:

```
SELECT COUNT(*), gender
FROM annp_final.event
GROUP BY gender;
```

We can then plot the result, as illustrated in Figure 7.

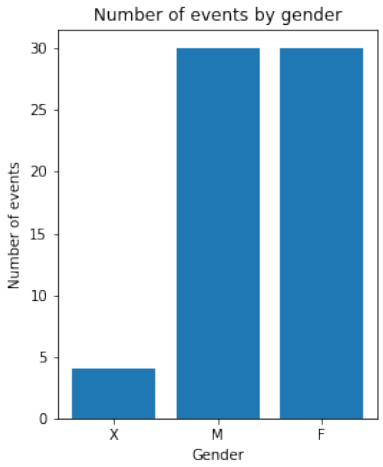

Figure 7: Number of events by gender

Here, the value `X` refers to events that allow athletes from both genders to participate. We can also plot this in a pie chart, as illustrated in Figure 8.

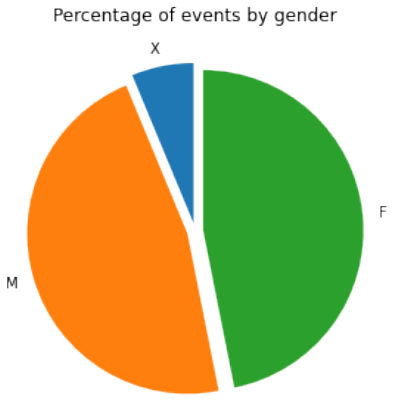

Figure 8: Percentage of events by gender

3.5 Number of Clubs by Nation

We can determine the number of clubs by each nation by running the following SQL query:

```
SELECT nation, COUNT(*) AS nationCount
FROM annp_final.club
GROUP BY nation
ORDER BY nationCount ASC;
```

COMPLETAR COM TEXTO.

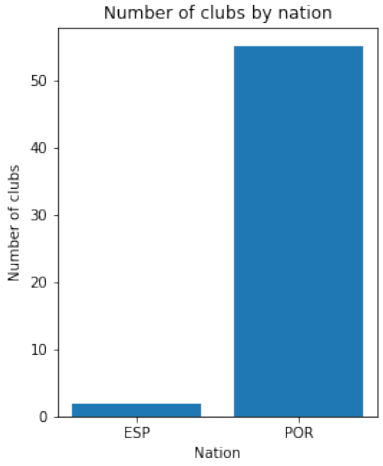

Figure 9: Number of clubs by nation

COMPLETAR COM TEXTO.

Figure 10: Percentage of clubs by nation

3.6 Number of Clubs by Region

COMPLETAR COM TEXTO.

```
SELECT region, COUNT(*) AS regionCount
FROM annp_final.club
WHERE region SIMILAR TO '[A-Z]+'
GROUP BY region
ORDER BY regionCount ASC;
```

COMPLETAR COM TEXTO.

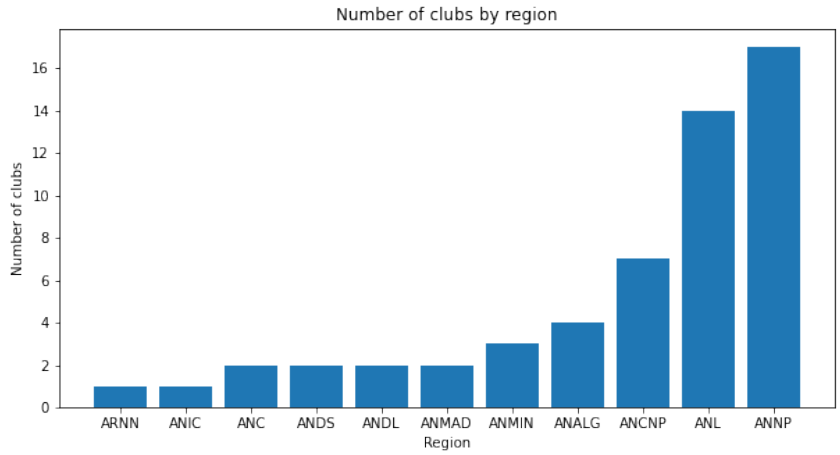

Figure 11: Number of clubs by region

3.7 Swim Styles

COMPLETAR COM TEXTO (style com mais distance) É FRESSTYLE.

```
SELECT *
FROM annp_final.swimstyle
ORDER BY distance DESC
LIMIT 1;
```

COMPLETAR COM TEXTO (style com menos distance) FLY.

```
SELECT *
FROM annp_final.swimstyle
ORDER BY distance ASC
LIMIT 1;
```

3.8 Results

3.8.1 Average Swim Time

COMPLETAR COM TEXTO (resultado = 00:02:23.769068).

```
SELECT AVG(swimtime)
FROM annp_final.result;
```

3.8.2 Average Number of Points

COMPLETAR COM TEXTO (resultado = 340).

```
SELECT AVG(points)::numeric(10, 1)
FROM annp_final.result;
```

3.9 Club facts statistics

3.9.1 Overall Statistics

Given the Club facts table, we can plot some of their statistics. Below, we present a query that fetches these statistics for all possible combinations between `meetid` and `clubid`. Note that instead we used the `code` column from `club` table to easily read each team for a given tournament.

```
SELECT
CASE GROUPING(cd.meetid)
    WHEN 1 THEN 'all_meets'
    ELSE cd.meetid
END AS "Tournament",
CASE GROUPING(c.code)
    WHEN 1 THEN 'all_clubs'
    ELSE c.code
END AS "Team",
   ROUND(AVG(average_age), 0) AS "Average Age",
   ROUND(AVG(average_swimtime), 2) AS "Average Swimtime",
   ROUND(SUM(total_points)) AS "Total Points",
   ROUND(SUM(number_of_players)) AS "Total Players"
FROM (
    SELECT CAST(meetid AS VARCHAR(255)),
       clubid,
       average_age,
       total_points,
```

```
        average_swimtime,
        number_of_players
    FROM annp_final.club_defacto) cd
JOIN annp_final.club c ON c.clubid = cd.clubid
GROUP BY CUBE (cd.meetid, c.code);
```

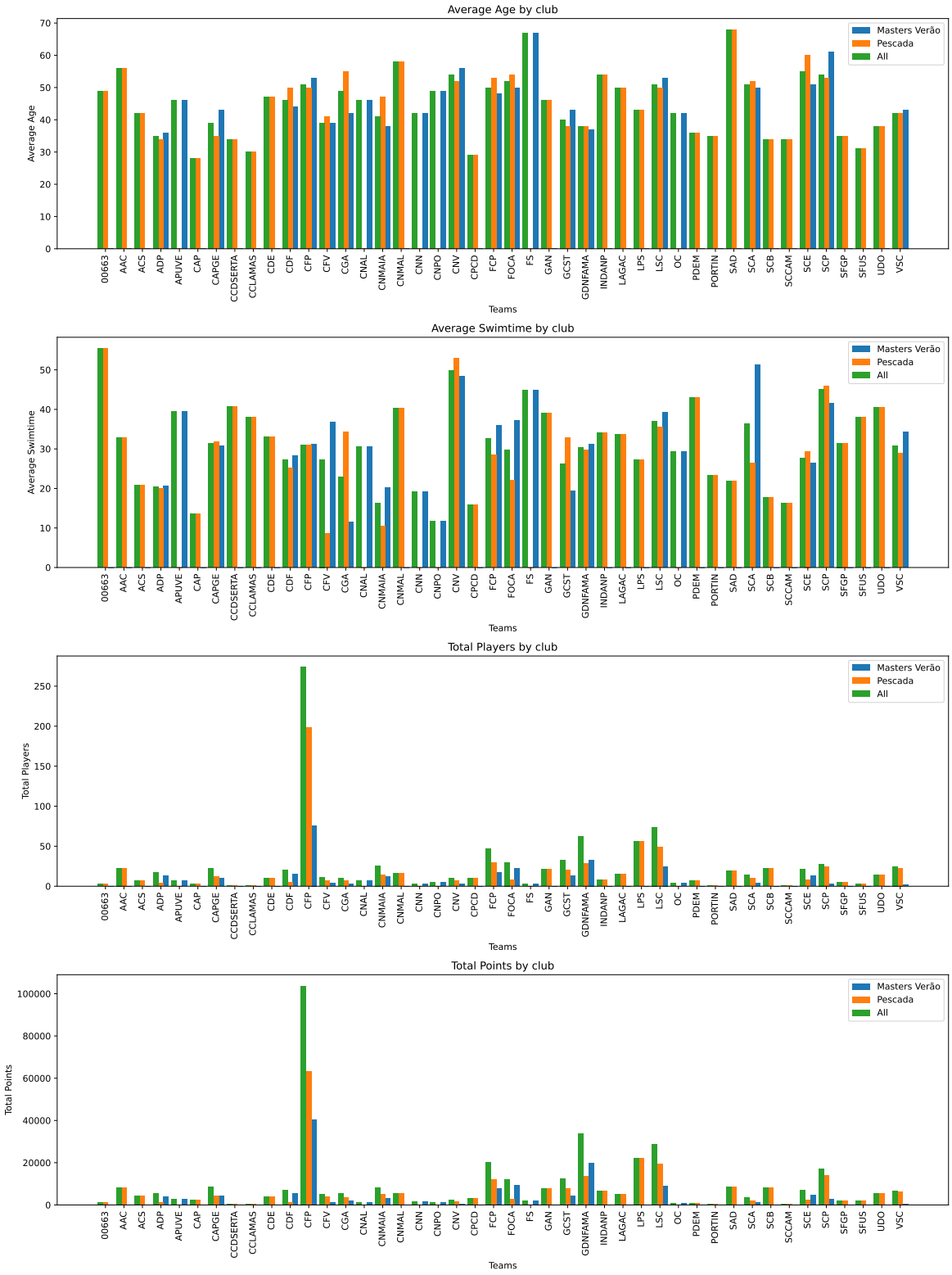

Figure 12: Statistics from fact Club table.

3.9.2 Statistics given a swimstyle

Next, we also show the overall statistics given a swim style. For that, we use the query

```
SELECT
    CASE GROUPING(c.code)
        WHEN 1 THEN 'all_clubs'
        ELSE c.code
    END AS "Team",
    CASE GROUPING(cd.swimstyleid)
        WHEN 1 THEN 'all_styles'
        ELSE cd.swimstyleid
    END AS "SwimStyle",
    ROUND(AVG(average_swimtime), 2) AS "Average Swimtime",
    ROUND(SUM(total_points)) AS "Total Points",
    ROUND(SUM(number_of_players)) AS "Total Players"
FROM annp_final.club_defacto cd
JOIN annp_final.club c on c.clubid = cd.clubid
GROUP BY CUBE (c.code, cd.swimstyleid)
ORDER BY "Total Points" DESC
```

To filter the amount of information this table has, we filter, for a given swim style, the top5 teams that had the higher total of points. This is done for all the tournaments. This pre-processing step was done in python and the result is depicted in Figure 13. Note that not all the bar plots have 5 teams. This is due to the lack of data presented from both tournaments.

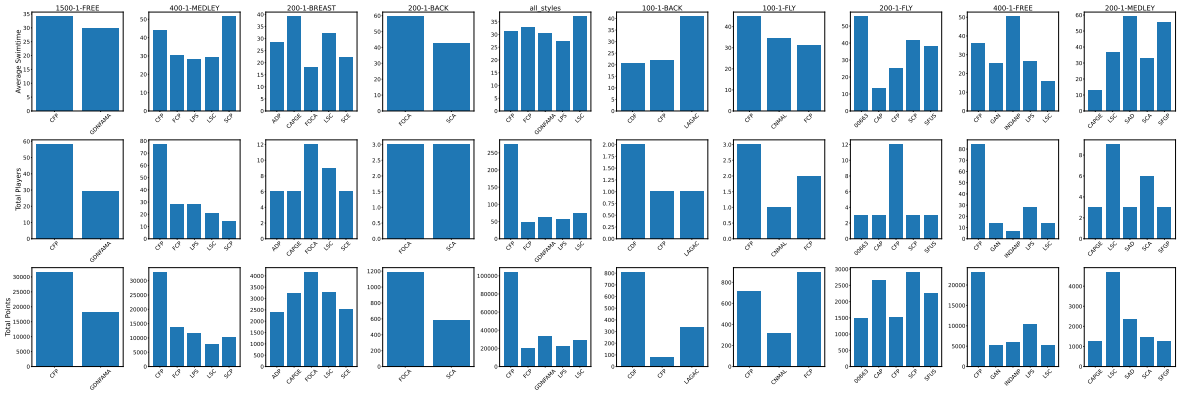

Figure 13: Top 5 teams per swim style for a given statistic.

3.10 Athlete facts statistics

```
SELECT
    CASE GROUPING(a.firstname)
        WHEN 1 THEN 'all_players'
        ELSE a.firstname
    END AS "Athletes",
    CASE GROUPING(af.meetid)
        WHEN 1 THEN 'all_meets'
        ELSE af.meetid
    END AS "Tournament",
    ROUND(AVG(average_points), 2) AS "Average Points",
    ROUND(AVG(average_distance), 2) AS "Average Distance",
    ROUND(AVG(average_swimtime), 2) AS "Average Swimtime"
FROM (
    SELECT
        athleteid,
        CAST(meetid as VARCHAR(255)),
        average_points,
        average_distance,
        average_swimtime
    FROM annp_final.athlete_defacto) af
JOIN annp_final.athlete a ON a.athleteid = af.athleteid
GROUP BY CUBE (a.firstname, af.meetid)
ORDER BY "Average Points" DESC
LIMIT 50;
```

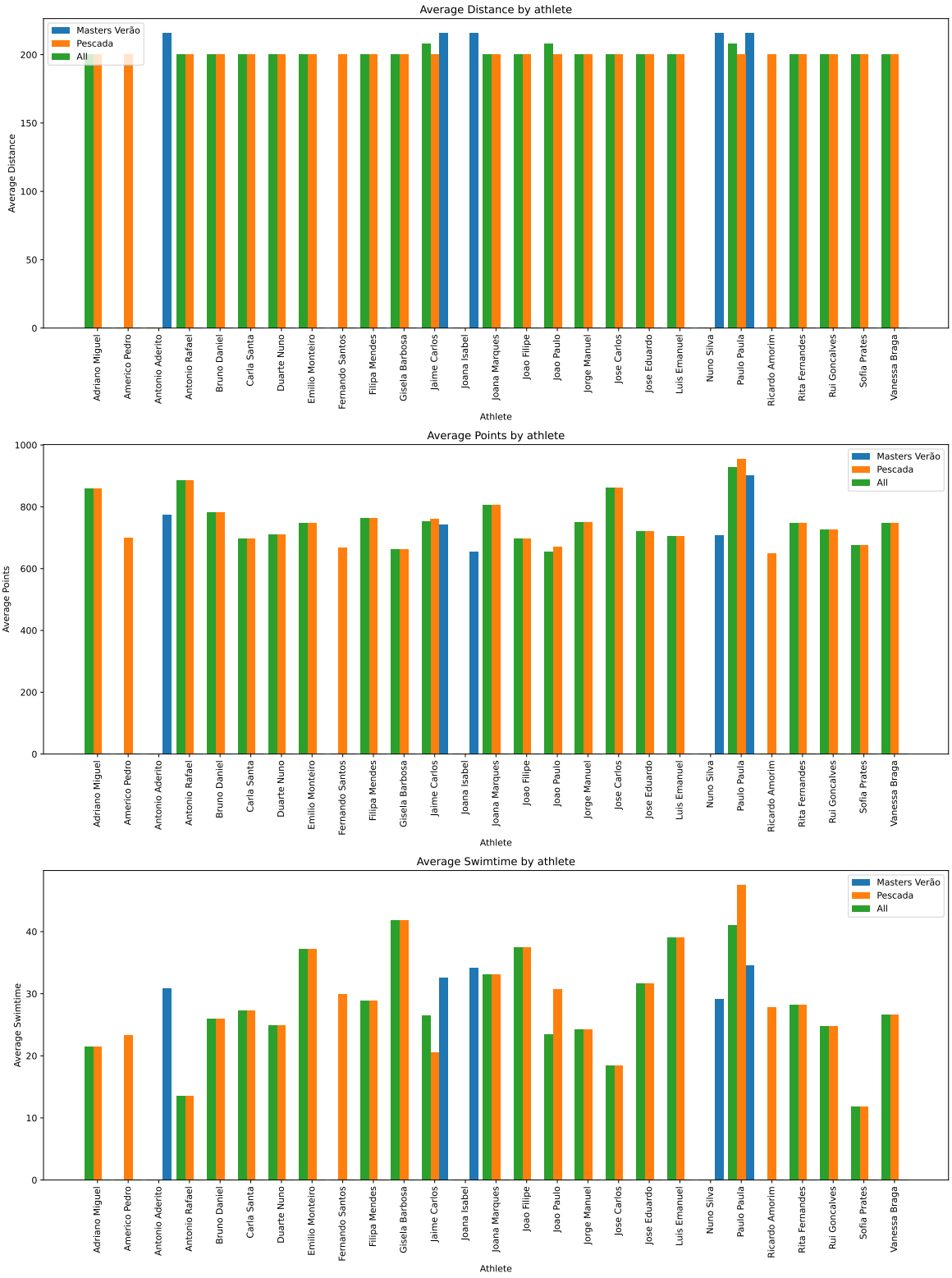

Figure 14: Statistics from fact Club table.

4 Conclusions & Future Work