# IST - Database and Data Mining

Anna Källén and Malte Åhman

November 2022

## 1 Introduction

The aim of this project was to practice database querying with the use of a Database Management System (DBMS). First of all, a dataset was chosen and several questions to answer were formulated. These questions were implemented in both relational algebra and SQL, and then performed on said dataset. This report also discusses problems that arose during the project and how these were solved.

## 2 Dataset

The chosen dataset contained information on previous Nobel Prizes, such as the full name of the winner, field of study, year, and a motivation. We found our data on the following links: https://api.nobelprize.org/v1/laureate.json and https://api.nobelprize.org/v1/prize.json

The data was preprocessed using Python. The original tables contained a lot of multiple valued attributes and redundant information, for examples all winners as an attribute to the prize, and thus had to be normalized. The original tables were prizes and winners.

The different columns in the original tables were saved in various dictionaries to separate the data before adding it to new tables. The dictionaries were then converted to Pandas dataframes and translated into SQL insert commands.

After the python preprocessing, PgAdmin4 was used to create tables, insert data and query the data.

## 3 Design and ER Diagram

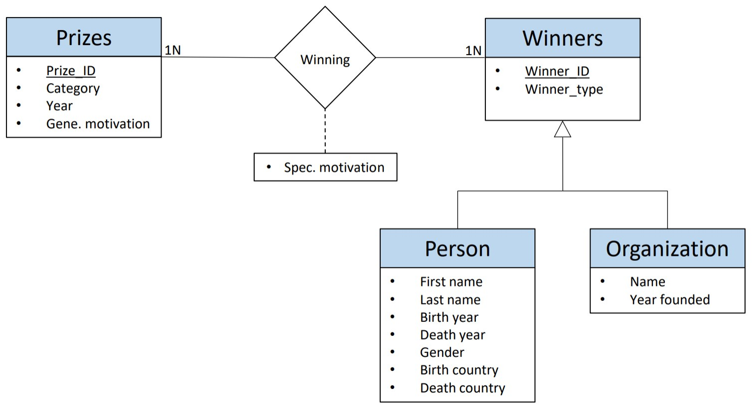


Figure 1: ER diagram of the database

As mentioned before, the original data contained two relations - winners and prizes. The prizes table contained information about year, category, overall motivation and an array of the people who won the prize. The array contained information about the winners name, surname, motivation and the number of winners sharing the prize. We decided to instead have a relation name Prizes that only contains year, category, overall motivation and a prize id to simplify querying. Furthermore, we created a separate relation named Winners, which had a relation - winning - with the prize. We decided to have the motivation as an attribute to the relational entity "winning" since a winner can win many prizes and thus have different motivations. A prize can also be won by many winners with different motivations.

The Winner entity was then divided into two subentities, since a winner can be either an organization or a person and the two subentities have different attributes.

## 4 Questions to Answer

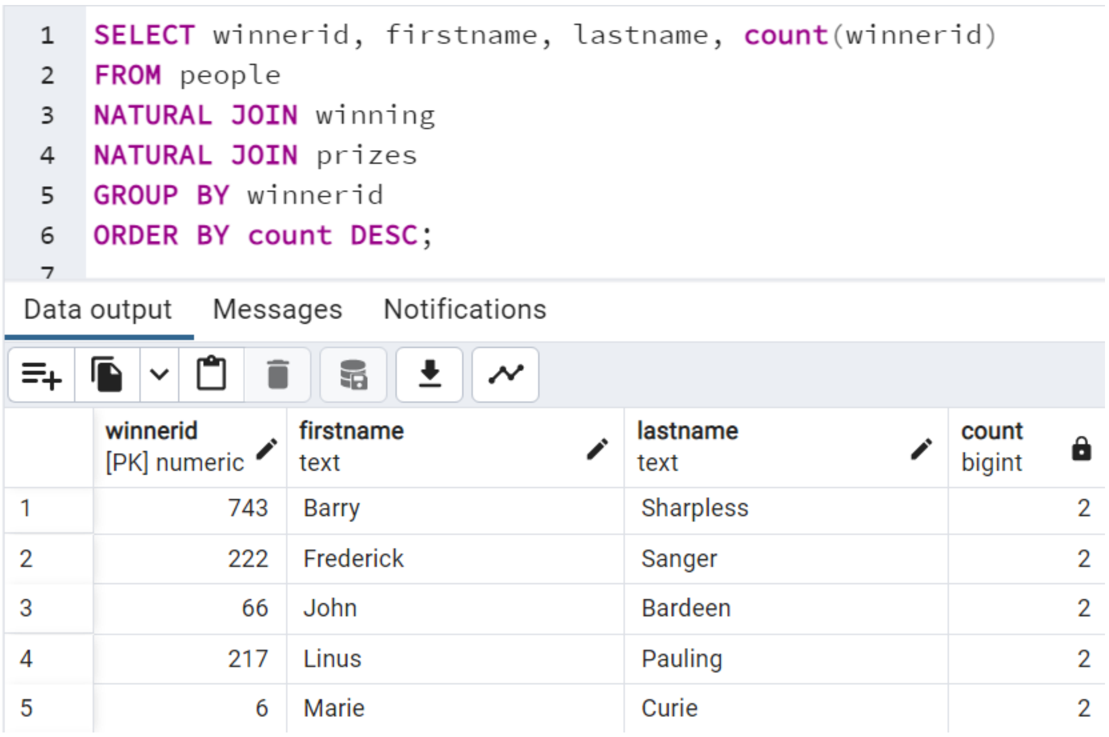
### 4.1 First question

**Natural language**: Give the name(s) of the person(s) that has/have won the most times.

**Relational algebra**:

$$\Pi_{winnerid,firstname,lastname,numprize}({}_{winnerid}G_{count(winnerid)\ as\ numprize}(people \bowtie_{wid=wid} \bowtie winning_{pid=pid}prize)) \tag{1}$$

**SQL**:



```
SELECT winnerid, firstname, lastname, count(winnerid)
FROM people
NATURAL JOIN winning
NATURAL JOIN prizes
GROUP BY winnerid
ORDER BY count DESC;
```

| | winnerid [PK] numeric | firstname text | lastname text | count bigint |
|---|---|---|---|---|
| 1 | 743 | Barry | Sharpless | 2 |
| 2 | 222 | Frederick | Sanger | 2 |
| 3 | 66 | John | Bardeen | 2 |
| 4 | 217 | Linus | Pauling | 2 |
| 5 | 6 | Marie | Curie | 2 |

Figure 2: SQL and result for the first question.

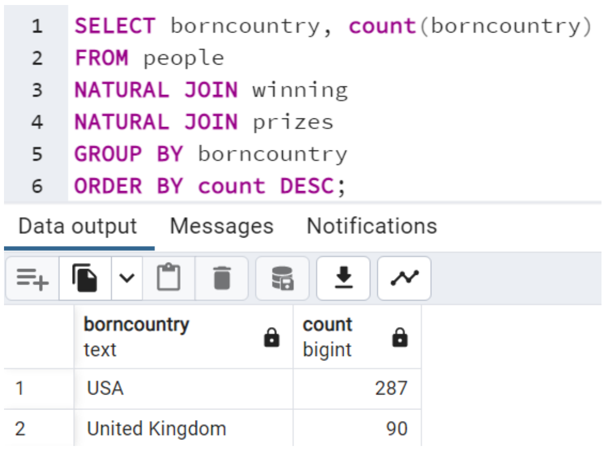
### 4.2 Second question

**Natural language**: What is the most common nationality of Nobel Prize winners?

**Relational algebra**:

$$\Pi_{borncountry,count}(_{borncountry}G_{count(borncountry)\ as\ count}(people \bowtie_{winnerid=winnerid} winning \bowtie_{prizeid=prizeid} prizes)) \tag{2}$$

**SQL**:



```
SELECT borncountry, count(borncountry)
FROM people
NATURAL JOIN winning
NATURAL JOIN prizes
GROUP BY borncountry
ORDER BY count DESC;
```

Data output Messages Notifications

| | borncountry text | count bigint |
|---|---|---|
| 1 | USA | 287 |
| 2 | United Kingdom | 90 |

Figure 3: SQL statement and result for the first query.

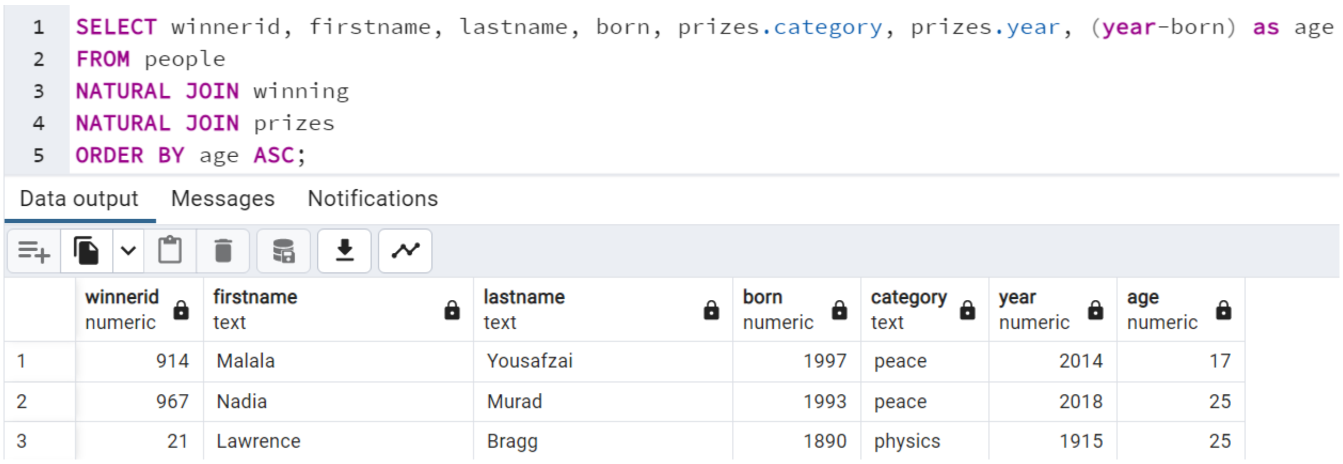
## 4.3 Third question

**Natural language**: How old is the all time youngest Nobel Prize winner?

**Relational algebra**:

$$\Pi_{winnerid,firstname,lastname,born,category,year,(year-born)}(people \bowtie_{winnerid=winnerid} winning \bowtie_{prizeid=prizeid} prizes) \tag{3}$$

**SQL**:



```
SELECT winnerid, firstname, lastname, born, prizes.category, prizes.year, (year-born) as age
FROM people
NATURAL JOIN winning
NATURAL JOIN prizes
ORDER BY age ASC;
```

Data output Messages Notifications

| | winnerid numeric | firstname text | lastname text | born numeric | category text | year numeric | age numeric |
|---|---|---|---|---|---|---|---|
| 1 | 914 | Malala | Yousafzai | 1997 | peace | 2014 | 17 |
| 2 | 967 | Nadia | Murad | 1993 | peace | 2018 | 25 |
| 3 | 21 | Lawrence | Bragg | 1890 | physics | 1915 | 25 |

Figure 4: SQL statement and result for the third question.

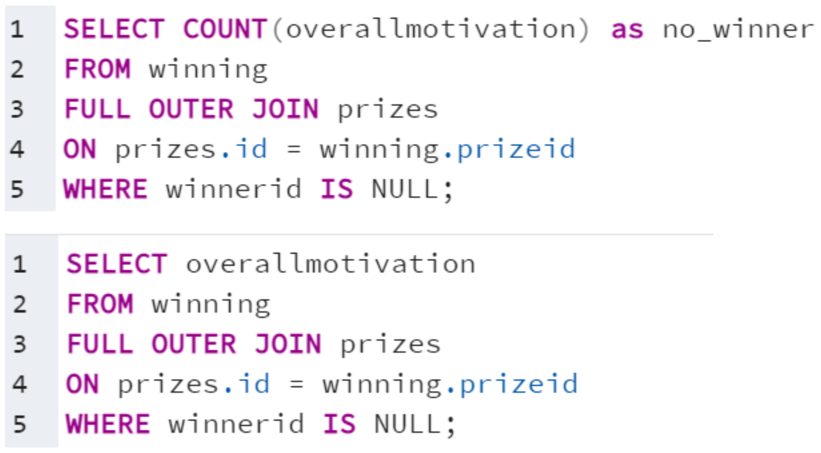
## 4.4 Fourth question

**Natural language**: Are there any prizes that no one won? How many and why?

**Relational algebra**:

$$\Pi_{overallmotivation}(\sigma_{winnerid=null}(prizes ⟗_{winnerid=winnerid} winning)) \tag{4}$$

**SQL**:



```sql
SELECT COUNT(overallmotivation) as no_winner
FROM winning
FULL OUTER JOIN prizes
ON prizes.id = winning.prizeid
WHERE winnerid IS NULL;
```

```sql
SELECT overallmotivation
FROM winning
FULL OUTER JOIN prizes
ON prizes.id = winning.prizeid
WHERE winnerid IS NULL;
```

Figure 5: SQL for question 4.

**Result**: 49 prizes. The motivation in all cases were either ”No Nobel Prize was awarded this year. The prize money was allocated to the Special Fund of this prize section.” or ”No Nobel Prize was awarded this year. The prize money was with 1/3 allocated to the Main Fund and with 2/3 to the Special Fund of this prize section.”.

## 5 Conclusions/summary

Most interesting queries to our database require a lot of joins. The relations organizations and people needed outer join to create a relation with all the Nobel Prize winners. This new relation needed to be joined with winning on winnerid and then joined with prizes on prizeid. These joins took a very long time, and it might have been easier not to divide the data into as many relations as we did. We could have had two relations - one with all prizes including winner id and one with all winners, both organizations and people. With this design, we would have got a lot of functional dependencies in the prizes relation, since one prize would correspond to multiple rows, one for each winner. We would also have got a lot of null values in the winners relation.

Another thing that we spent a lot of time on was setting up PgAdmin for PostgreSQL and making sure that we could run queries. It’s hard to say what we specifically could have done differently but we could often find a solution by googling. In general, the project has helped us understand how to set up a relational database and perform basic operations in SQL. When it comes to what we could have done better, it had been interesting to build a user interface. That way it would have been easier to illustrate the results and show more examples.