EMSE 6586

DATABASE CREATION AND ANALYSIS OF NOBEL PRIZE WINNERS

Anbu Ezhilmathi Nambi
G33350186

INTRODUCTION

- The Nobel Prize is an esteemed international accolade awarded yearly across multiple fields for exceptional achievements.
- The dataset encompasses information on more than 900 laureates dating back to 1901.
- The goal of this project is to construct a well-organized SQL database from the Nobel Prize dataset using Python scripts.
- This database facilitates the analysis of the dataset, allowing for the identification of trends and patterns among laureates across different prize categories and eras.
- The aim of this project is to provide a comprehensive understanding of the dataset and reveal valuable insights.

DATA TRANSLATION

01 FETCH JSON DATA

Fetching Data from Nobel Prize APIs

02 PARSE JSON DATA

After fetching the JSON data, it is parsed to extract the necessary information.

03 CREATE SQLITE DATABASE SCHEMA

Set up a SQLite database to store the parsed data

04 CREATE DATABASE TABLES

Design and create database tables to efficiently store and query the Nobel Prize data.

ABOUT THE DATA

- The dataset contains detailed information on Nobel laureates from 1901 to 2023.
- It includes data from all six Nobel Prize categories: Peace, Literature, Chemistry, Physics, Medicine, and Economic Sciences.
- Each record includes details about the laureates such as names, birthdates, birthplaces, and affiliations.
- The dataset provides specifics on the prize, including the year of the award, the motivations for each prize, and information on instances where multiple laureates shared a prize.
- The data is in JSON format, which simplifies the process of data handling and analysis.

1. Fetch the JSON Data from API

```python
# Step 1: Fetch JSON data
# prize
response1 = requests.get("https://api.nobelprize.org/v1/prize.json")
data1 = response1.json()
# laureate
response2 = requests.get("https://api.nobelprize.org/v1/laureate.json")
data2 = response2.json()
```

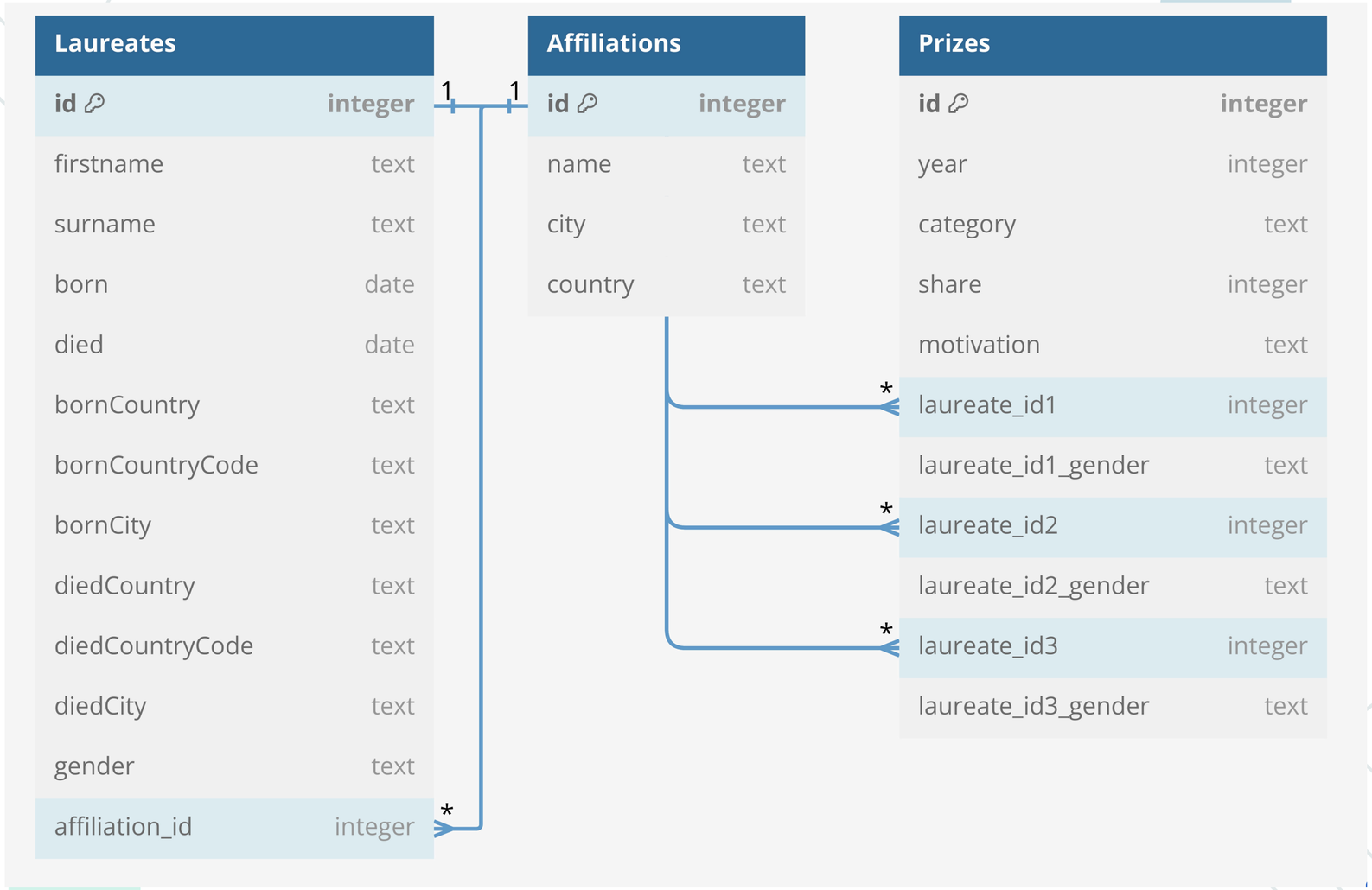
2. Sample of Prize data.json


```json
{
    "id": "1",
    "firstname": "Wilhelm Conrad",
    "surname": "R\u00f6ntgen",
    "born": "1845-03-27",
    "died": "1923-02-10",
    "bornCountry": "Prussia (now Germany)",
    "bornCountryCode": "DE",
    "bornCity": "Lennep (now Remscheid)",
    "diedCountry": "Germany",
    "diedCountryCode": "DE",
    "diedCity": "Munich",
    "gender": "male",
    "prizes": [
        {
            "year": "1901",
            "category": "physics",
            "share": "1",
            "motivation": "\"in recognition of the extraordinary services he has rendered by the discovery of the remarkable rays subsequently named after him\"",
            "affiliations": [
                {
                    "name": "Munich University",
                    "city": "Munich",
                    "country": "Germany"
                }
            ]
        }
    ]
}
```


3. Sample of Laureate data.json


```json
{
    "year": "2023",
    "category": "chemistry",
    "laureates": [
        {
            "id": "1029",
            "firstname": "Moungi",
            "surname": "Bawendi",
            "motivation": "\"for the discovery and synthesis of quantum dots\"",
            "share": "3"
        },
        {
            "id": "1030",
            "firstname": "Louis",
            "surname": "Brus",
            "motivation": "\"for the discovery and synthesis of quantum dots\"",
            "share": "3"
        },
        {
            "id": "1031",
            "firstname": "Aleksey",
            "surname": "Yekimov",
            "motivation": "\"for the discovery and synthesis of quantum dots\"",
            "share": "3"
        }
    ]
}
```

ENTITY RELATIONSHIP DIAGRAM



1. Creating SQLite Database

```
# Step 3: Create SQLite database schema
conn = sqlite3.connect('nobel_prizes.db')
cursor = conn.cursor()
```

2. Create Laureates table

```
# Step 4: Create database tables

# Create Laureates table
cursor.execute('''CREATE TABLE IF NOT EXISTS Laureates (
                id INTEGER PRIMARY KEY,
                firstname TEXT,
                surname TEXT,
                born DATE,
                died DATE,
                bornCountry TEXT,
                bornCountryCode TEXT,
                bornCity TEXT,
                diedCountry TEXT,
                diedCountryCode TEXT,
                diedCity TEXT,
                gender TEXT,
                affliation_id INTEGER
                )''')
```

3. Create Affiliations table

```
# Create Affiliations table
cursor.execute('''CREATE TABLE Affiliations (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                name TEXT,
                city TEXT,
                country TEXT
                )''')
```

4. Create Prize table

```
# Create Prizes table
cursor.execute('''CREATE TABLE Prizes (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                year INTEGER,
                category TEXT,
                share INTEGER,
                motivation TEXT,
                laureate_id1 INTEGER,
                laureate_id1_gender TEXT,
                laureate_id2 INTEGER,
                laureate_id2_gender TEXT,
                laureate_id3 INTEGER,
                laureate_id3_gender TEXT
                )''')
```

1. Inserting Values into Laureates Table

```
for laureate in laureate_data:
    cursor.execute('''INSERT INTO Laureates VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,?)''',
                (laureate['id'],
                 laureate.get('firstname', None),
                 laureate.get('surname', None),
                 laureate.get('born', None),
                 laureate.get('died', None),
                 laureate.get('bornCountry', None),
                 laureate.get('bornCountryCode', None),
                 laureate.get('bornCity', None),
                 laureate.get('diedCountry', None),
                 laureate.get('diedCountryCode', None),
                 laureate.get('diedCity', None),
                 laureate.get('gender', None),
                None))
```

2. Inserting Values into Affiliations Table

```
cursor.execute("INSERT INTO Affiliations (name, city, country) VALUES (?, ?, ?)", (name, city, country))
```

3. Inserting Values into Prizes Table

```
cursor.execute("INSERT INTO Prizes (year, category, share, motivation) VALUES (?, ?, ?, ?)", (year, category, share, motivation))
```

SAMPLE OUTPUT

1 Laureates who won more than one Nobel Prize:

	firstname	surname	prize_count
0	Marie	Curie	2
1	John	Bardeen	2
2	Linus	Pauling	2
3	Frederick	Sanger	2
4	International Committee of the Red Cross	None	3
5	Office of the United Nations High Commissioner...	None	2
6	Barry	Sharpless	2

2 First Females to win Nobel Prize in each category

	Year	Full Name	Category	Birth Country
0	1903	Marie Curie	physics	Russian Empire (now Poland)
1	1905	Bertha von Suttner	peace	Austrian Empire (now Czech Republic)
2	1909	Selma Lagerlöf	literature	Sweden
3	1911	Marie Curie	chemistry	Russian Empire (now Poland)
4	1947	Gerty Cori	medicine	Austria-Hungary (now Czech Republic)
5	2009	Elinor Ostrom	economics	USA

3 Youngest Nobel Laureates

	Age	Full Name	Year	Category
0	17	Malala Yousafzai	2014	peace
1	25	Lawrence Bragg	1915	physics
2	31	Carl D. Anderson	1936	physics
3	31	Paul A.M. Dirac	1933	physics
4	31	Tsung-Dao Lee	1957	physics
5	31	Werner Heisenberg	1932	physics

4 Top 10 Affiliations that won Nobel Prizes:

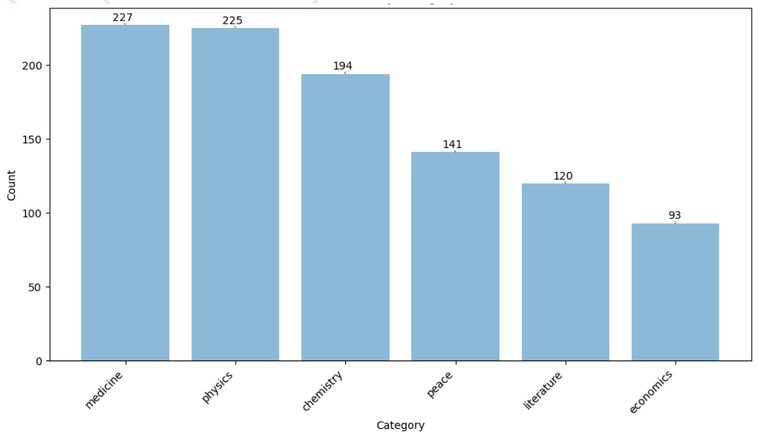
	Affiliation Name	Prize Count
0	University of California	36
1	Harvard University	28
2	Massachusetts Institute of Technology (MIT)	23
3	Stanford University	22
4	University of Chicago	19
5	California Institute of Technology (Caltech)	19
6	University of Cambridge	18
7	Columbia University	18
8	Princeton University	17
9	Rockefeller University	13

5 Gender Distribution: A Trend Analysis from 2009 to 2023

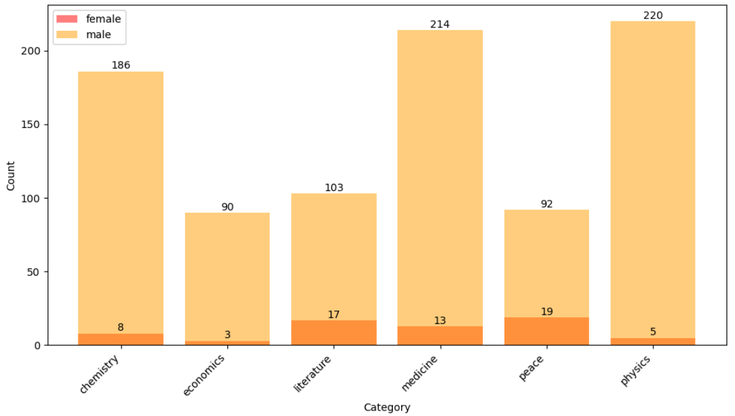
gender year	female	male	org	Total	Male %	Female %
2009	5	8	0	13	61.538462	38.461538
2010	0	11	0	11	100.000000	0.000000
2011	3	10	0	13	76.923077	23.076923
2012	0	9	1	10	90.000000	0.000000
2013	1	11	1	13	84.615385	7.692308
2014	2	11	0	13	84.615385	15.384615
2015	2	8	1	11	72.727273	18.181818
2016	0	11	0	11	100.000000	0.000000
2017	0	11	1	12	91.666667	0.000000
2018	4	9	0	13	69.230769	30.769231
2019	1	13	0	14	92.857143	7.142857
2020	4	7	1	12	58.333333	33.333333
2021	1	12	0	13	92.307692	7.692308
2022	2	10	2	14	71.428571	14.285714
2023	4	7	0	11	63.636364	36.363636

SAMPLE OUTPUT

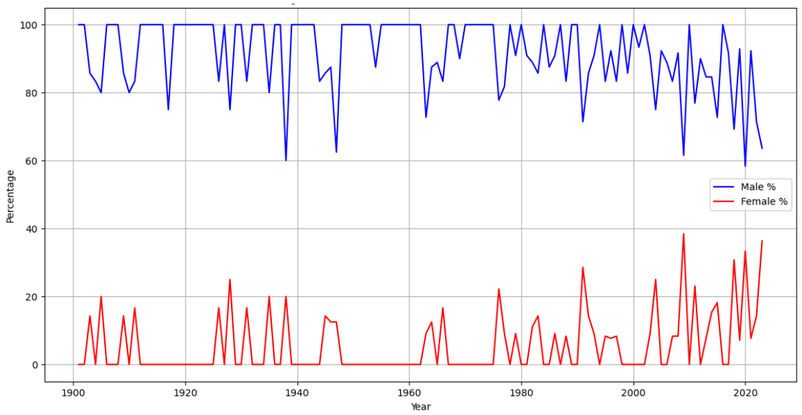
1. Prize by Category



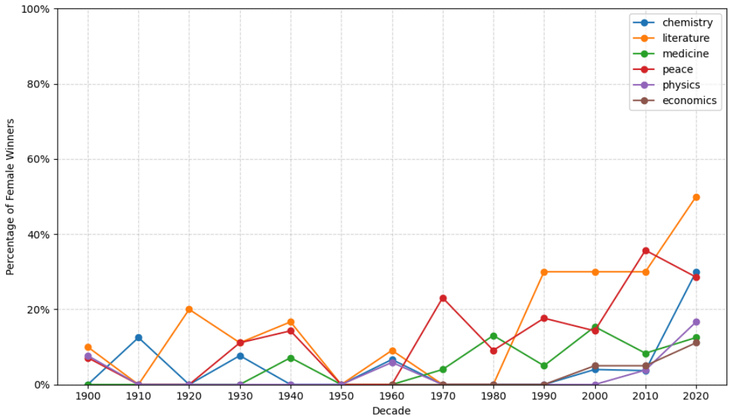
2. Prize won by Category and Gender



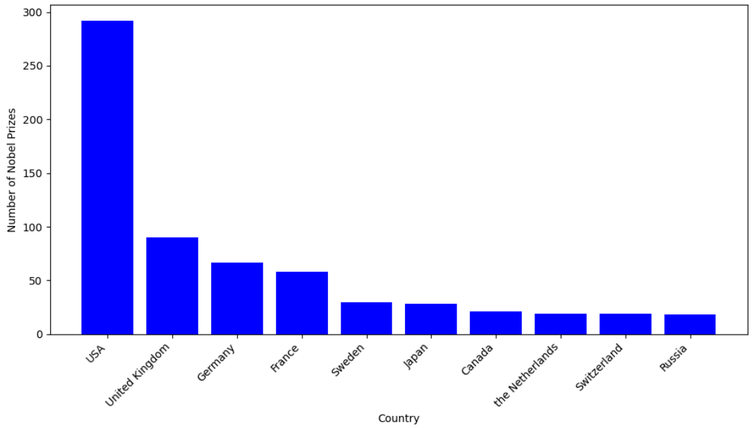
3. Percentage of Male vs Female Nobel Laureates Over the Years



4. Percentage Female Winners by Decade and Category



5. Top 10 Countries with Most Nobel Prizes by Birth Country of Laureates



CHALLENGES

- When working with Nobel Prize data from different databases, it's common to face inconsistencies and missing values. This makes it crucial to ensure data consistency, handle missing values, and clean up inconsistencies such as empty strings, 'None', or 'Unknown' values, as seen in SQL queries. It can be a challenging task, but it's essential for data cleaning, integration, and integrity.
- Integrating data from various sources like laureates, affiliations, and prizes is complex. Effective table joining and handling different data formats or schemas require careful attention to ensure successful data integration.
- Maintaining data quality requires ensuring the accuracy and integrity of data. To achieve this, it is important to validate data entries, verify relationships between entities, and prevent duplicate records. In particular, avoiding duplicated data entries is crucial as data is often collected from multiple columns that may have overlapping content.
- It is crucial to collect complete data that includes laureate affiliation, country, and prize category to avoid any impact on the analysis results due to missing data. Taking necessary measures to prevent data loss is important.

OVERCOME CHALLENGES

- To ensure consistency across datasets, use data cleaning techniques to standardize data formats. This involves trimming whitespace, converting text to a consistent case, and handling special characters.
- Another important aspect to consider is replacing missing values, which can be done using appropriate placeholders or statistical methods to impute missing values.
- Validating data is crucial to ensure consistent and accurate information in the database.
- To prevent duplicate entries in a database, it's important to establish procedures for detecting and resolving them. You can achieve this by using SQL's GROUP BY and HAVING clauses to find repeated entries or implementing uniqueness constraints within the database design.

CONCLUSION

- Constructed a well-organized SQL database from the Nobel Prize dataset using Python scripts.
- Dataset encompasses information on over 900 laureates spanning from 1901 to 2023 across six Nobel Prize categories.
- Facilitates analysis to identify trends and patterns among laureates, enabling a comprehensive understanding of the dataset and revealing valuable insights.
- Managed inconsistencies, missing values, and data integration complexities through data cleaning, validation, and de-duplication techniques.
- Ensured accuracy and integrity by standardizing formats, replacing missing values, and implementing procedures for detecting and resolving duplicates.

THANK YOU