University of Pisa

Master's degree in
Data Science & Business Informatics

Laboratory Data Science
First assignment

Build a Data Warehouse

Marco Ciompi [537856]
Luftjan Saliaj [606507]
Pasquale Gorrasi [587417]

November 20, 2021

Contents

1 Introduction

1.1 Objectives

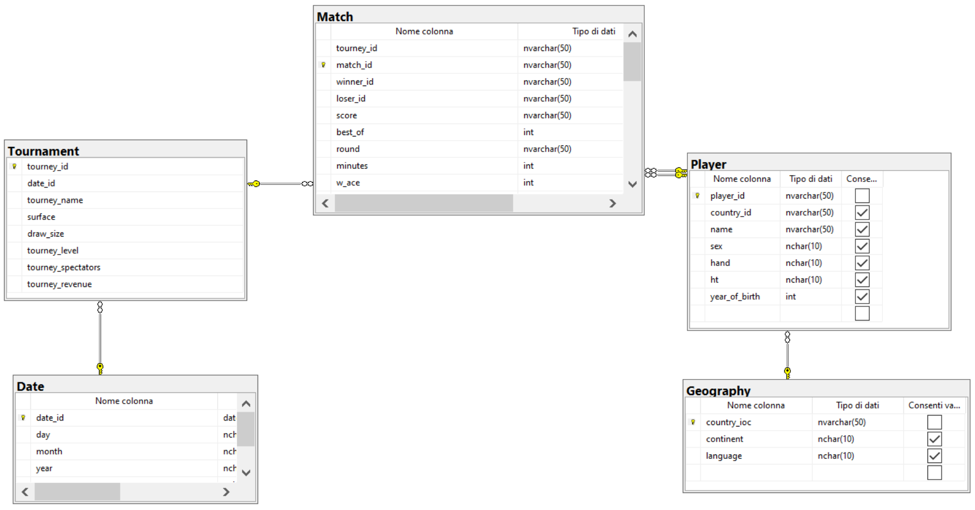
In Part 1 of the project you are required to create and populate a database starting from .csv files and perform different operations on it.

1.2 files

"tennis.csv" contains the main body of data: a fact table with tennis match data. For each match we have information about the tournament, the players involved (winner and loser) and several other metrics. Files *"male players.csv"* and *"female players.csv"* contain the list of male players and female players respectively, while *"geography.csv"* contain a list of IOC codes with country names and continents. In these four files you will findnd all the attributes to reproduce the schema shown in 1. The file *"tennis.csv"* will have to be split appropriately and combined with the other files to achieve this goal. The goal of the following assignments is to build the schema and deploy it on server *lds.di.unipi.it*

2 Assignment 0

Assignment 0 asks to recreate a Database Schema using Microsoft SQL Server Management Studio in the server *lds.di.unipi.it*



Match

Nome colonna	Tipo di dati
tourney_id	nvarchar(50)
match_id	nvarchar(50)
winner_id	nvarchar(50)
loser_id	nvarchar(50)
score	nvarchar(50)
best_of	int
round	nvarchar(50)
minutes	int
w_ace	int

Tournament

tourney_id
date_id
tourney_name
surface
draw_size
tourney_level
tourney_spectators
tourney_revenue

Date

Nome colonna	
date_id	dat
day	nch
month	nch
year	nch

Player

Nome colonna	Tipo di dati	Conse...
player_id	nvarchar(50)	☐
country_id	nvarchar(50)	✓
name	nvarchar(50)	✓
sex	nchar(10)	✓
hand	nchar(10)	✓
ht	nchar(10)	✓
year_of_birth	int	✓
		☐

Geography

Nome colonna	Tipo di dati	Consenti va...
country_ioc	nvarchar(50)	☐
continent	nchar(10)	✓
language	nchar(10)	✓
		☐

Figure 2.1: Database Schema

the keys chosen are:

- *'match id'*: for the fact table Match
- *'tourney id'*: for the table Tournament
- *'date id'*: for the table Date
- *'player id'*: for the table Player
- *'country ioc'*: for the table Geography

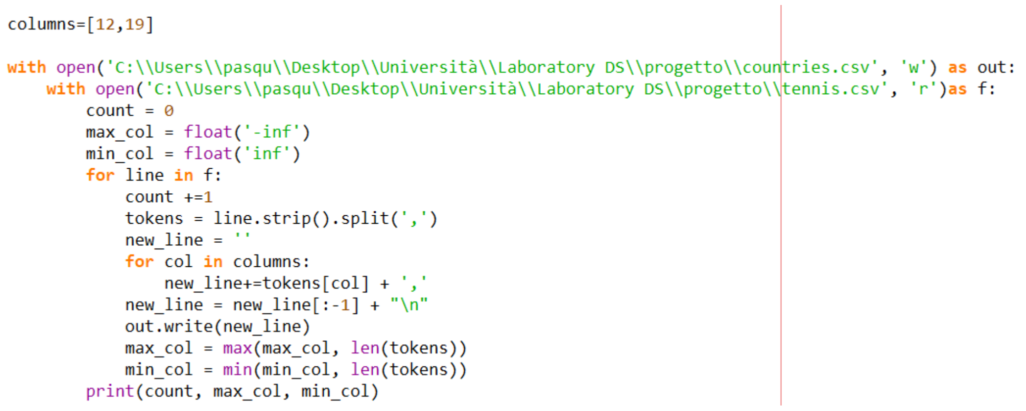
The foreign key relations are the followings:

- FK-Match-Tourneament: the attributes *tourney id*
- FK-Tournament-Date: the attributes *date id*
- FK-Match-Players: the attributes *winner id,loser id* on one side and *player id* on the other
- FK-Player-Geography: the attributes *country id/country ioc*

3 Assignment 1

3.1 Split Tennis.csv

Assigment 1 asks to split the given csv files to obtain the data we need to populate the database schema designed in the previous assignment, without using pandas library. To extract the five .csv files needed from *"tennis.csv"* the following code has been used, modifying for every file the column index to copy from the table.

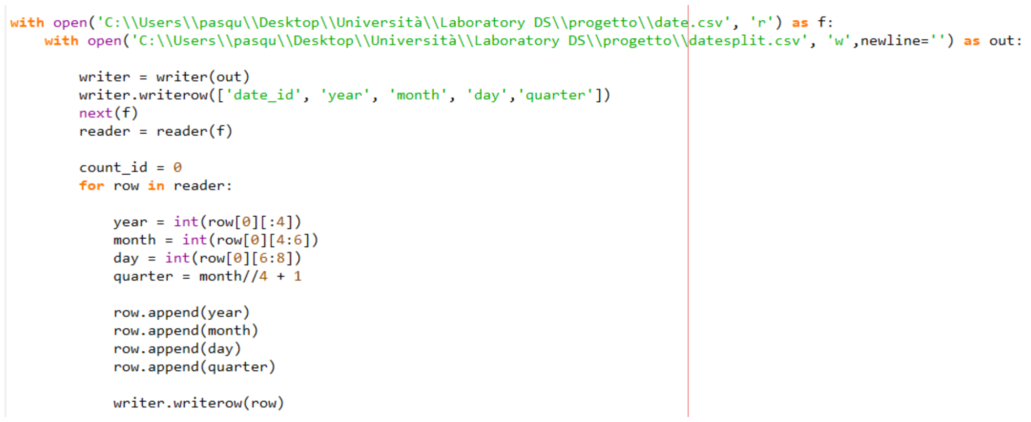


```python
columns=[12,19]

with open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\countries.csv', 'w') as out:
    with open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\tennis.csv', 'r')as f:
        count = 0
        max_col = float('-inf')
        min_col = float('inf')
        for line in f:
            count +=1
            tokens = line.strip().split(',')
            new_line = ''
            for col in columns:
                new_line+=tokens[col] + ','
            new_line = new_line[:-1] + "\n"
            out.write(new_line)
            max_col = max(max_col, len(tokens))
            min_col = min(min_col, len(tokens))
        print(count, max_col, min_col)
```

Figure 3.1: write new csv files from tennis.csv

The csv files *"tournament.csv"* and *"geography.csv"* have not been modified except for the header, while for the other csv files other transformations were necessary.

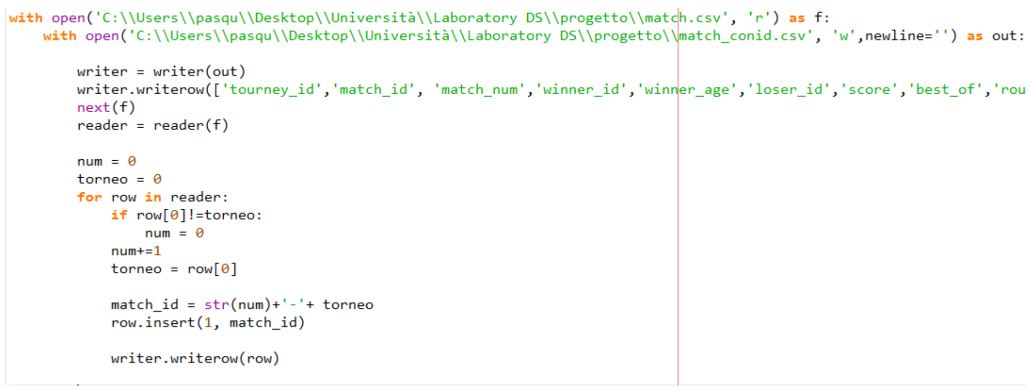
Concerning the file *"date.csv"*, since the original table has just a string with the complete date, a splitting has been operated in order to obtain the attributes *"day,month,year"* and a mathematical operation to obtain the attribute *"quarter"*.



```python
with open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\date.csv', 'r') as f:
    with open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\datesplit.csv', 'w',newline='') as out:

        writer = writer(out)
        writer.writerow(['date_id', 'year', 'month', 'day','quarter'])
        next(f)
        reader = reader(f)

        count_id = 0
        for row in reader:

            year = int(row[0][:4])
            month = int(row[0][4:6])
            day = int(row[0][6:8])
            quarter = month//4 + 1

            row.append(year)
            row.append(month)
            row.append(day)
            row.append(quarter)

            writer.writerow(row)
```

Figure 3.2: code for splitting dates

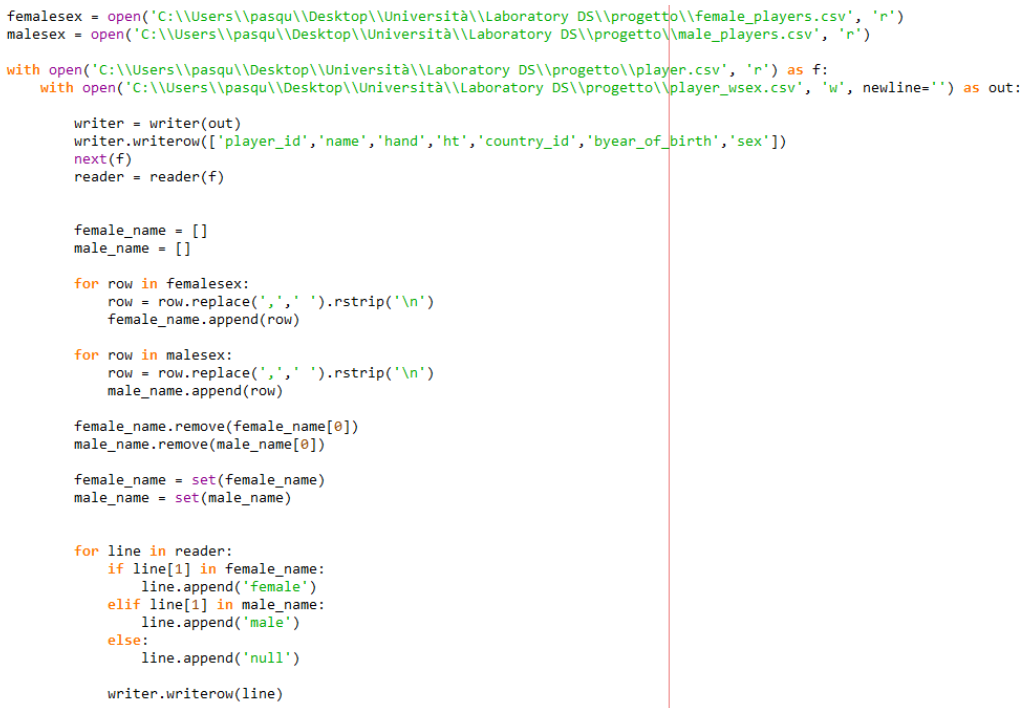
Regarding the file *"match.csv"* we created the attribute *"match-id"* that will be used as a key for the table. We did this by adding to the attribute *"tourney-id"* a number generated by a counter that goes back to 0 when the *"tourney-id"* changes, in order to have a different value for any record.



```
with open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\match.csv', 'r') as f:
    with open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\match_conid.csv', 'w',newline='') as out:

        writer = writer(out)
        writer.writerow(['tourney_id','match_id', 'match_num','winner_id','winner_age','loser_id','score','best_of','rou
        next(f)
        reader = reader(f)

        num = 0
        torneo = 0
        for row in reader:
            if row[0]!=torneo:
                num = 0
            num+=1
            torneo = row[0]

            match_id = str(num)+'-'+ torneo
            row.insert(1, match_id)

            writer.writerow(row)
```

Figure 3.3: code for generating id

Last but not least, the file *"player.csv"* required the greatest efforts. At first we created two different csv files named *winners* and *losers* using the code previously reported in *Figure 3.1*. Then we merged the two files in order to have a complete list of all the players and their features despite the final result of their games. The last step was generating a new attribute *sex* using the other csv files given to us: *female-players.csv* and *male-players.csv* that contains name and surname of all the players regarding their gender. To get the informations we needed from this tables and generating the new attribute we used the following code:



```
femalesex = open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\female_players.csv', 'r')
malesex = open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\male_players.csv', 'r')

with open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\player.csv', 'r') as f:
    with open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\player_wsex.csv', 'w', newline='') as out:

        writer = writer(out)
        writer.writerow(['player_id','name','hand','ht','country_id','byear_of_birth','sex'])
        next(f)
        reader = reader(f)


        female_name = []
        male_name = []

        for row in femalesex:
            row = row.replace(',',' ').rstrip('\n')
            female_name.append(row)

        for row in malesex:
            row = row.replace(',',' ').rstrip('\n')
            male_name.append(row)

        female_name.remove(female_name[0])
        male_name.remove(male_name[0])

        female_name = set(female_name)
        male_name = set(male_name)


        for line in reader:
            if line[1] in female_name:
                line.append('female')
            elif line[1] in male_name:
                line.append('male')
            else:
                line.append('null')

            writer.writerow(line)
```

Figure 3.4: code for generating sex attribute

3.2 Transformation

After the splitting and formatting with python the csv files needed to populate the Data Warehouse, we cleaned them using the python library **pandas**.

The files *"geography.csv"* has just one value that presented the values "Unknown" regarding the Pacific Islands, it has been fixe. *"Date.csv"* did not show missing values, so it's been just cleaned up from duplicate values. The file *"tournament.csv"* had just a few missing values for the attribute *"surface"*, they were replaced by the mode *"Hard"*

The file *"match.csv"* presented a large number of missing values, especially for the attributes regarding the statistics. We decided to eliminate only the rows with missing values in the attributes *score, winner id, loser id*, keeping all the others.

The file *"player.csv"* was cleaned out by duplicates and presented the following missing values:

hand	32
ht	9542
byear-of-birth	2092
sex	30

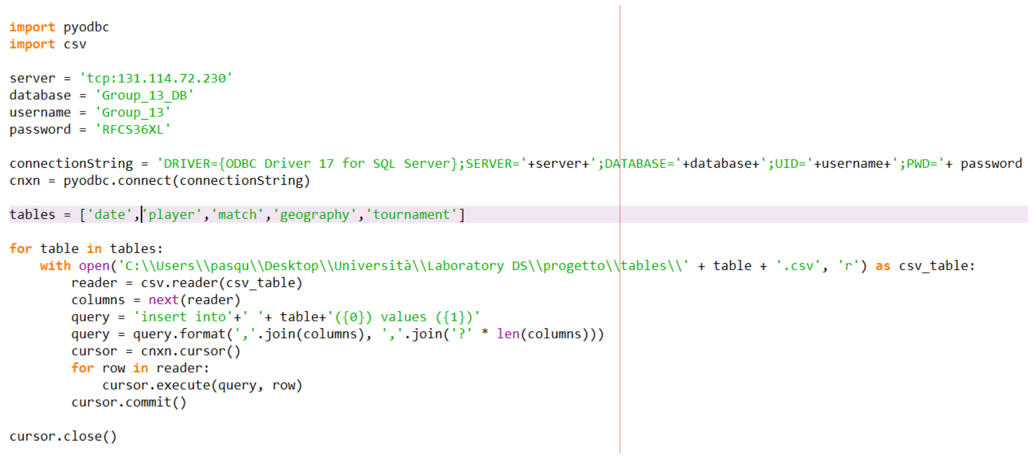
For the attribute *"sex"*, since the missing values were just a few, we inferred them by looking for the player's name on Google.
For the attribute *"ht"* we decided to infer it making an average after grouping players by sex and filling the null values with the average height for their gender.
The attribute *"hand"* has been filled with his mode *U*.
Finally the missing values for the attribute *"byear-of-birth"* were left, after transforming the column that presented the age of the players instead of their birthdate.

Concerning the key relations between *geography* and *player* we verified that all the values that are in the foreign key *'country ioc'* are present also in the primary key in the table players *'country id'* comparing the intersection between the two sets of values. We discovered different values missing or typing errors and we fixed them before uploading the tables. The same was done regarding the key relation between *player id* and *winner id/loser id.*

4 Assignment 2

4.1 Uploading Data

Once fixed all the key relations between the tables we populated the tables on microsoft sql using the following python code:



```python
import pyodbc
import csv

server = 'tcp:131.114.72.230'
database = 'Group_13_DB'
username = 'Group_13'
password = 'RFCS36XL'

connectionString = 'DRIVER={ODBC Driver 17 for SQL Server};SERVER='+server+';DATABASE='+database+';UID='+username+';PWD='+ password
cnxn = pyodbc.connect(connectionString)

tables = ['date','player','match','geography','tournament']

for table in tables:
    with open('C:\\Users\\pasqu\\Desktop\\Università\\Laboratory DS\\progetto\\tables\\' + table + '.csv', 'r') as csv_table:
        reader = csv.reader(csv_table)
        columns = next(reader)
        query = 'insert into'+' '+ table+'({0}) values ({1})'
        query = query.format(','.join(columns), ','.join('?' * len(columns)))
        cursor = cnxn.cursor()
        for row in reader:
            cursor.execute(query, row)
        cursor.commit()

cursor.close()
```

Figure 4.1: python code for populating tables