notbook: Olympic-Events

Creater: Alexander Gandji

Date: 23.08.2022

Table of Contents

Discriptive States

In [128…

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import kendalltau
from scipy.stats import spearmanr
!pip install -U pandasql
!pip install pandoc
from pandasql import sqldf
pysqldf=lambda q: sqldf(q, globals())
```

```
Requirement already satisfied: pandasql in /Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages (0.7.3)
Requirement already satisfied: pandas in /Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages (from pandasql) (1.4.3)
Requirement already satisfied: numpy in /Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages (from pandasql) (1.21.5)
Requirement already satisfied: sqlalchemy in /Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages (from pandasql) (1.4.39)
Requirement already satisfied: python-dateutil>=2.8.1 in /Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages (from pandas->pandasql) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages (from pandas->pandasql) (2022.1)
Requirement already satisfied: greenlet!=0.4.17 in /Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages (from sqlalchemy->pandasql) (1.1.1)
Requirement already satisfied: six>=1.5 in /Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages (from python-dateutil>=2.8.1->pandas->pandasql) (1.16.0)
WARNING: Error parsing requirements for path: [Errno 2] No such file or directory: '/Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages/path-16.0.0.dist-info/METADATA'
WARNING: Error parsing requirements for more-itertools: [Errno 2] No such file or directory: '/Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages/more_itertools-8.10.0.dist-info/METADATA'
WARNING: Error parsing requirements for fastcache: [Errno 2] No such file or directory: '/Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages/fastcache-1.1.0.dist-info/METADATA'
WARNING: Error parsing requirements for contextlib2: [Errno 2] No such file or directory: '/Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages/contextlib2-0.6.0.post1.dist-info/METADATA'
WARNING: Error parsing requirements for cached-property: [Errno 2] No such file or directory: '/Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages/cached_property-1.5.2.dist-info/METADATA'
WARNING: Error parsing requirements for async-generator: [Errno 2] No such file or directory: '/Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages/async_generator-1.10.dist-info/METADATA'
WARNING: Error parsing requirements for asn1crypto: [Errno 2] No such file or directory: '/Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages/asn1crypto-1.4.0.dist-info/METADATA'
Requirement already satisfied: pandoc in /Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages (2.2)
Requirement already satisfied: ply in /Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages (from pandoc) (3.11)
Requirement already satisfied: plumbum in /Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages (from pandoc) (1.7.2)
WARNING: Error parsing requirements for path: [Errno 2] No such file or directory: '/Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages/path-16.0.0.dist-info/METADATA'
WARNING: Error parsing requirements for more-itertools: [Errno 2] No such file or directory: '/Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages/more_itertools-8.10.0.dist-info/METADATA'
WARNING: Error parsing requirements for fastcache: [Errno 2] No such file or directory: '/Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages/fastcache-1.1.0.dist-info/METADATA'
WARNING: Error parsing requirements for contextlib2: [Errno 2] No such file or directory: '/Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages/contextlib2-0.6.0.post1.dist-info/METADATA'
WARNING: Error parsing requirements for cached-property: [Errno 2] No such file or directory: '/Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages/cached_property-1.5.2.dist-info/METADATA'
WARNING: Error parsing requirements for async-generator: [Errno 2] No such file or directory: '/Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages/async_generator-1.10.dist-info/METADATA'
WARNING: Error parsing requirements for asn1crypto: [Errno 2] No such file or directory: '/Users/alexandergandji/opt/anaconda3/lib/python3.9/site-packages/asn1crypto-1.4.0.dist-info/METADATA'
```

Note

For each line of SQL code will be the snytax in mardown posted

In [129…

```
#read csv files
nationality=pd.read_csv('/Users/alexandergandji/Desktop/Course_Folder/Capstone Project SQL Data Science /Sports
athlete=pd.read_csv('/Users/alexandergandji/Desktop/Course_Folder/Capstone Project SQL Data Science /Sports/ath
sports=pd.read_csv('/Users/alexandergandji/Desktop/Course_Folder/Capstone Project SQL Data Science /Sports/spor
results=pd.read_csv('/Users/alexandergandji/Desktop/Course_Folder/Capstone Project SQL Data Science /Sports/res
olympic_games=pd.read_csv('/Users/alexandergandji/Desktop/Course_Folder/Capstone Project SQL Data Science /Spor
olympic_medal=pd.read_csv('/Users/alexandergandji/Desktop/Course_Folder/Capstone Project SQL Data Science /Spor
```

Most Successful Nation in Olympic History

In [130…

```
#sql medal overall count
medals1=pysqldf('select Year, NOC_ID, Sport_ID, count(distinct Medal_ID) as medals_won, count(distinct case whe
```

In [131…

```
all_time_medals_ov=pysqldf('select nationality.NOC as Nations, sum(medals_won) as all_time_medals, sum(gold) as
all_time_medals_ov.set_index('Nations', inplace=True)
```

SQL Syntax */Getting the overall medal count for the different nations/*

select

```
nationality.NOC as Nations,
sum(medals_won) as all_time_medals,
sum(gold) as gold_medals,
sum(silver) as silver_medals,
sum(bronze) as bronze_medals
```

from(

```
select
    Year,
    NOC_ID,
    Sport_ID,
    count(distinct Medal_ID) as medals_won,
    count(distinct case when Medal_ID=1 then 1 else null end) as gold,
    count(distinct case when Medal_ID=2 then 1 else null end) as silver,
    count(distinct case when Medal_ID=3 then 1 else null end) as bronze
    from results
    where Medal_ID !=4
    group by Year, NOC_ID, Sport_ID order by Year
```

) as medals1

```
left join nationality

    on medals1.NOC_ID=nationality.NOC_ID
```

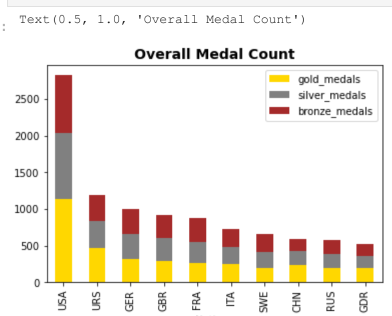
group by nationality.NOC

order by sum(medals_won) desc limit 10;

In [132…

```
all_time_medals_ov[['gold_medals', 'silver_medals', 'bronze_medals']].plot(kind='bar', stacked=True,
                                                                           color=['gold', 'grey', 'brown'])
plt.title('Overall Medal Count', size=14, fontweight='bold')
```

Out[132]: Text(0.5, 1.0, 'Overall Medal Count')



In [133…

```
all_time_medals_ov
```

Out[133]:

Nations	all_time_medals	gold_medals	silver_medals	bronze_medals
USA	2823	1131	901	791
URS	1197	471	373	353
GER	1004	319	345	340
GBR	919	289	320	310
FRA	879	264	286	329
ITA	722	256	225	241
SWE	657	200	215	242
CHN	595	239	184	172
RUS	584	202	183	199
GDR	519	192	165	162

As we can see, the USA has gathered the most medals over the last 120 years and is followed by UDSSR, Germany, Great Britain and France.

Hypothesis=> The US is the most successful Nation

The US has the most medals accumulated over 120 years (overall, gold, silver and bronze).

Medals USA

In [134…

```
#sql for getting the count of the most successful events for the US
usa_m=pysqldf('Select results.Year, sports.Sport, results.NOC_ID, results.Medal_ID, count(distinct case when re
```

In [135…

```
usa_dis_m=pysqldf('Select Sport, (sum(gold_medal)+sum(silver_medal)+sum(bronze_medal)) as total_medals, sum(gol
```

SQL Syntax */Medals of the US/*

Select

```
Sport,
(sum(gold_medal)+sum(silver_medal)+sum(bronze_medal)) as total_medals,
sum(gold_medal) as number_of_gold_medals,
sum(silver_medal) as number_of_silver_medals,
sum(bronze_medal) as number_of_bronze_medals
```

from (

```
Select
    results.Year,
    sports.Sport,
    results.NOC_ID,
    results.Medal_ID,
    count(distinct case when results.Medal_ID=1 then 1 else null end) as gold_medal,
    count(distinct case when results.Medal_ID=2 then 1 else null end) as silver_medal,
    count(distinct case when results.Medal_ID=3 then 1 else null end) as bronze_medal

from results
    left join sports
        on results.Sport_ID=sports.Sport_ID
where NOC_ID=217
group by 1,2,3,4
order by results.Year
```

) as usa_m

where Medal_ID != 4

group by Sport

order by (sum(gold_medal)+sum(silver_medal)+sum(bronze_medal)) desc;

In [136…

```
usa_dis_m
```

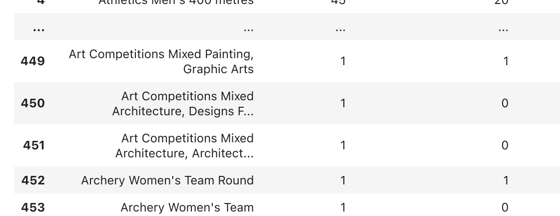
Out[136]:

	Sport	total_medals	number_of_gold_medals	number_of_silver_medals	number_of_bronze_medals
0	Athletics Men's 110 metres Hurdles	57	20	20	17
1	Athletics Men's Shot Put	51	19	20	12
2	Athletics Men's Long Jump	49	23	15	11
3	Athletics Men's 200 metres	46	17	18	11
4	Athletics Men's 400 metres	45	20	13	12
...	...	...	...	...	...
449	Art Competitions Mixed Painting, Graphic Arts	1	1	0	0
450	Art Competitions Mixed Architecture, Designs F...	1	0	1	0
451	Art Competitions Mixed Architecture, Architect...	1	0	1	0
452	Archery Women's Team Round	1	1	0	0
453	Archery Women's Team	1	0	0	1

454 rows × 5 columns

In [244…

```
data=usa_dis_m.total_medals.head(10)
lables=usa_dis_m.Sport.head(10)
colors=sns.color_palette('pastel')[0:9]
plt.pie(data, labels=lables, colors=colors, autopct='%.0f%%')
plt.title('Most Successful Events of the US', size=16, fontweight='bold')
plt.show()
```



In [138…

```
#most successful events of the US
usa_dis_m[['Sport', 'total_medals']].sort_values('total_medals', ascending=False).head(10)
```

Out[138]:

	Sport	total_medals
0	Athletics Men's 110 metres Hurdles	57
1	Athletics Men's Shot Put	51
2	Athletics Men's Long Jump	49
3	Athletics Men's 200 metres	46
4	Athletics Men's 400 metres	45
5	Athletics Men's Pole Vault	44
6	Athletics Men's 400 metres Hurdles	41
7	Athletics Men's 100 metres	41
8	Diving Men's Springboard	39
9	Swimming Men's 100 metres Backstroke	37

When it comes to the total medal count,

- Athletics Men's 110 metres Hurdles
- Athletcis Men's Shot Put
- Athletcis Men's Long Jump
- Athletcis Men's 200 meters
- Athletcis Men's 400 meters

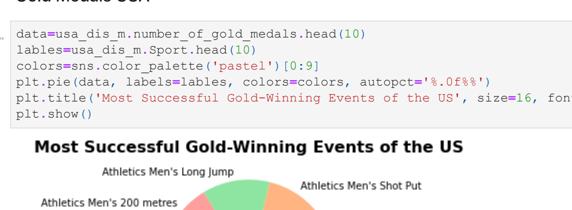
are the most successful events of the US.

Next, what will it look like for just gold, silver and bronze medals

Gold Medals USA

In [245…

```
data=usa_dis_m.number_of_gold_medals.head(10)
lables=usa_dis_m.Sport.head(10)
colors=sns.color_palette('pastel')[0:9]
plt.pie(data, labels=lables, colors=colors, autopct='%.0f%%')
plt.title('Most Successful Gold-Winning Events of the US', size=16, fontweight='bold')
plt.show()
```



In [140…

```
#most succesful events of the US based on their gold medal count
usa_dis_m[['Sport', 'number_of_gold_medals']].sort_values('number_of_gold_medals', ascending=False).head(10)
```

Out[140]:

	Sport	number_of_gold_medals
2	Athletics Men's Long Jump	23
0	Athletics Men's 110 metres Hurdles	20
4	Athletics Men's 400 metres	20
1	Athletics Men's Shot Put	19
6	Athletics Men's 400 metres Hurdles	19
5	Athletics Men's Pole Vault	18
29	Athletics Men's 4 x 400 metres Relay	17
19	Swimming Men's 4 x 200 metres Freestyle Relay	17
7	Athletics Men's 100 metres	17
3	Athletics Men's 200 metres	17

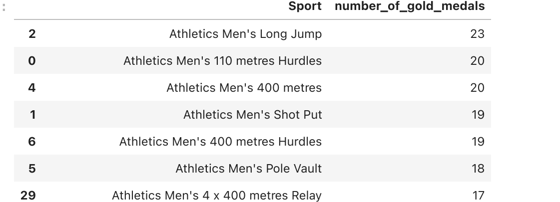
Throughout the last 120 years, the US has won the most gold medals in Athletics Men's 400 metres Hurdles.

What will we see when it comes to silver medals?

Silver Medals

In [141…

```
data=usa_dis_m.number_of_silver_medals.head(10)
lables=usa_dis_m.Sport.head(10)
colors=sns.color_palette('pastel')[0:9]
plt.pie(data, labels=lables, colors=colors, autopct='%.0f%%')
plt.title('Most Successful Events-Silver for the US', size=16, fontweight='bold')
plt.show()
```



In [142…

```
#most succesful events of the US based on their silver medal count
usa_dis_m[['Sport', 'number_of_silver_medals']].sort_values('number_of_silver_medals', ascending=False).head(10
```

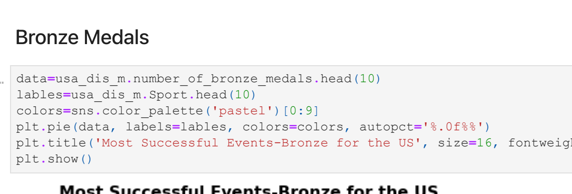
Out[142]:

	Sport	number_of_silver_medals
0	Athletics Men's 110 metres Hurdles	20
1	Athletics Men's Shot Put	20
3	Athletics Men's 200 metres	18
2	Athletics Men's Long Jump	15
7	Athletics Men's 100 metres	15
9	Swimming Men's 100 metres Backstroke	14
4	Athletics Men's 400 metres	13
5	Athletics Men's Pole Vault	13
11	Athletics Men's High Jump	13
6	Athletics Men's 400 metres Hurdles	12

Bronze Medals

In [143…

```
data=usa_dis_m.number_of_bronze_medals.head(10)
lables=usa_dis_m.Sport.head(10)
colors=sns.color_palette('pastel')[0:9]
plt.pie(data, labels=lables, colors=colors, autopct='%.0f%%')
plt.title('Most Successful Events-Bronze for the US', size=16, fontweight='bold')
plt.show()
```



As we can see, the US's most succesfull overall medal events (Top 5) are;

- Athletics Men's 110 metres Hurdles
- Athletics Men's Shot Put
- Athletics Men's Long Jump
- Athletics Men's 200 metres
- Athletics Men's 400 metres

The most succesfull gold medal events (Top 5) are;

- Athletics Men's 110 metres Hurdles
- Athletics Men's Shot Put
- Athletics Men's Long Jump
- Athletics Men's 400 metres Hurdles
- Athletics Men's 400 metres

From 1896-2016 Number of Events for Women and Men

Hypothesis=> In 2016 women and men have an equal number of particpants

In [144…

```
#sql to get the numbers of sports events of men and women
men_women=pysqldf('Select results.Year, olympic_games.Games, count(distinct case when results.Sex_ID=1 then res
```

In [145…

```
#sql difference between men and women events (in numbers)
dif_men_women=pysqldf('select Year, Games, sum(male_medal_count) as events_men, sum(female_medal_count) as even
```

SQL Syntax */Difference in Numbers between Male and Female/*

Select

```
Year,
Games,
sum(male_medal_count) as events_men,
sum(female_medal_count) as events_women,
(sum(female_medal_count) - sum(male_medal_count)) as difference_between_men_women
```

from (

```
Select
    results.Year,
    olympic_games.Games,
    count(distinct case when results.Sex_ID=1 then results.Sport_ID else null end) as
male_medal_count,
    count(distinct case when results.Sex_ID=2 then results.Sport_ID else null end) as
female_medal_count
from results
left join olympic_games
    on results.Games_ID=olympic_games.Games_ID
group by results.Year, olympic_games.Games
```

) as men_women

group by Year, Games;

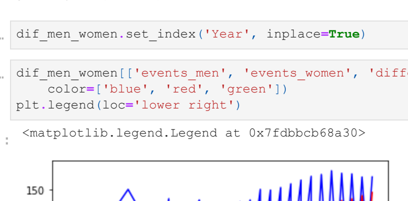
In [146…

```
dif_men_women.set_index('Year', inplace=True)
```

In [147…

```
dif_men_women[['events_men', 'events_women', 'difference_between_men_women']].plot.line(
    color=['blue', 'red', 'green'])
plt.legend(loc='lower right')
```

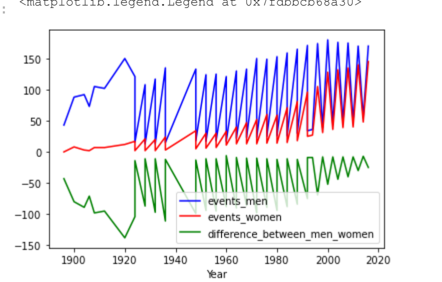
Out[147]: <matplotlib.legend.Legend at 0x7fdbbcb68a30>



In [148…

```
dif_men_women[['events_men', 'events_women', 'difference_between_men_women']].plot.line(
    color=['blue', 'red', 'green'], legend=None)
plt.title('Differencen in number of events between Men and Women in 120 years', size=12, fontweight='bold')
```

Out[148]: Text(0.5, 1.0, 'Differencen in number of events between Men and Women in 120 years')



As we can see, men generally had more events than women, but the trend shows that the difference between men and women is getting smaller.

We have to reject the hypothesis because women did not have the same number of events as men in 2016.

Who is the US's biggest competitor

Answering this question, I have set the observation time frame to the last 4 summer and winter Olympic games (2002-2016)

In [149…

```
comp=pysqldf('Select Year, NOC_ID, Sport_ID, count(distinct Medal_ID) as medals_won from results where Medal_ID
```

In [150…

```
#medal count in the time range 2004-2016 only summer games
comp_summer=pysqldf('select nationality.NOC, Sum(medals_won) as summer_medals from comp left join nationality o
```

SQL Syntax */Medal Summer Games/*

select

```
nationality.NOC,
Sum(medals_won) as summer_medals
```

from (

```
Select
    Year,
    NOC_ID,
    Sport_ID,
    count(distinct Medal_ID) as medals_won
from results
where Medal_ID in (1,2,3)
group by 1,2,3
order by Year
```

) as comp_sum

```
left join nationality
    on comp_sum.NOC_ID=nationality.NOC_ID
```

Where Year in (2004, 2008, 2012, 2016)

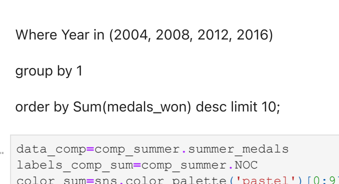
group by 1

order by Sum(medals_won) desc limit 10;

In [151…

```
data_comp=comp_summer.summer_medals
labels_comp_sum=comp_summer.NOC
color_sum=sns.color_palette('pastel')[0:9]
plt.pie(data_comp, labels=labels_comp_sum, colors=color_sum, autopct='%.0f%%')
plt.title('Medals won in the Summer Games 2004-2016', size=12, fontweight='bold')
```

Out[151]: Text(0.5, 1.0, 'Medals won in the Summer Games 2004-2016')



In [152…

```
#medal tabel summer games 2004-2016
comp_summer
```

Out[152]:

	NOC	summer_medals
0	USA	435
1	CHN	321
2	RUS	300
3	GBR	210
4	GER	176
5	AUS	160
6	FRA	151
7	JPN	141
8	ITA	115
9	KOR	110

In [153…

```
comp_gold=pysqldf('Select Year, NOC_ID, Sport_ID, count(distinct Medal_ID) as medals_won from results where Med
```

In [154…

```
#count of gold medals by nation in the time range 2004-2016 (summer games)
comp_summer_gold=pysqldf('select nationality.NOC, Sum(medals_won) as summer_gold_medals from comp_gold left joi
```

SQL Syntax */Gold Medals Summer Games/*

Select

```
nationality.NOC,
Sum(medals_won) as summer_gold_medals
```

from (

```
Select
    Year,
    NOC_ID,
    Sport_ID,
    count(distinct Medal_ID) as medals_won
from results
where Medal_ID = 1
group by 1,2,3
order by Year
```

) as comp_gold

```
left join nationality
    on comp_gold.NOC_ID=nationality.NOC_ID
```

Where Year in (2004, 2008, 2012, 2016)

group by 1

order by Sum(medals_won) desc limit 10;

In [155…

```
#gold medal tabel summer games 2004-2016
comp_summer_gold
```

Out[155]:

	NOC	summer_gold_medals
0	USA	164
1	CHN	147
2	RUS	94
3	GBR	84
4	GER	57
5	AUS	46
6	KOR	44
7	JPN	44
8	FRA	39
9	ITA	34

China was the biggest competitor for the US in the last four summer games. We can see that in the overall medal and gold medal count.

In [156…

```
#counting medals by nation time range 2002-2014
comp_winter=pysqldf('select nationality.NOC, Sum(medals_won) as winter_medals from comp left join nationality o
```

SQL Syntax */Medal Winter Games/*

Select

```
nationality.NOC,
Sum(medals_won) as winter_medals
```

from (

```
Select
    Year,
    NOC_ID,
    Sport_ID,
    count(distinct Medal_ID) as medals_won
from results
where Medal_ID in (1,2,3)
group by 1,2,3
order by Year
```

) as comp_winter

```
left join nationality
    on comp.NOC_ID=nationality.NOC_ID
```

Where Year in (2002, 2006, 2010, 2014)

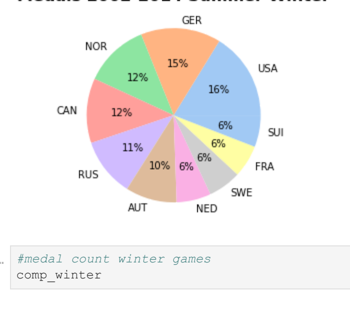
group by 1

order by Sum(medals_won) desc limit 10';

In [157…

```
data_winter=comp_winter.winter_medals
lables_winter=comp_winter.NOC
color_winter=sns.color_palette('pastel')[0:9]
plt.pie(data_winter, labels=lables_winter, colors=color_winter, autopct='%.0f%%')
plt.title('Medals 2002-2014 Summer Winter', size=16, fontweight='bold')
```

Out[157]: Text(0.5, 1.0, 'Medals 2002-2014 Summer Winter')



In [158…

```
#medal count winter games
comp_winter
```

Out[158]:

	NOC	winter_medals
0	USA	124
1	GER	114
2	NOR	92
3	CAN	92
4	RUS	83
5	AUT	73
6	NED	49
7	SWE	47
8	FRA	46
9	SUI	45

In [159…
```
#counting gold medals by nation time range 2002-2014
comp_winter_gold=pysqldf('select nationality.NOC, Sum(medals_won) as gold_winter_medals from comp_gold left joi
```

SQL Syntax */Gold Medals Winter Games/*

Select

```
nationality.NOC,
Sum(medals_won) as winter_gold_medals
```

from (

```
Select
    Year,
    NOC_ID,
    Sport_ID,
    count(distinct Medal_ID) as medals_won
from results
where Medal_ID = 1
group by 1,2,3
order by Year
```

) as comp_winter_gold

```
left join nationality
    on comp_winter_gold.NOC_ID=nationality.NOC_ID
```

Where Year in (2002, 2006, 2010, 2014)

group by 1

order by Sum(medals_won) desc limit 10;

In [160…
```
#gold medal count winter games
comp_winter_gold
```

Out[160]:

	NOC	gold_winter_medals
0	GER	41
1	CAN	38
2	USA	37
3	NOR	34
4	RUS	29
5	SUI	20
6	AUT	20
7	NED	18
8	KOR	17
9	SWE	14

But in the winter games, we can see a different trend, the USA was number one in the overall Medal count, but on the gold medal level, they were only number 3.

One of the goals of the study was to figure out who was/is the most successful nation in the period of 2004-2016 (summer games) and if China was a runner-up. The data shows that the US is, in this period, the most successful nation as well, and the biggest competitor for the US in the Olympic summer games is indeed China. Additionally, I wanted to understand why China was a runner-up. Is it because they were less successful in the same events/disciplines as the US and, therefore, automatically runner-up, or did they succeed in fewer events than the US? Were these events different to the US?

To make a fair comparison, I used a time filter of the last four Olympic summer games to see which events/disciplines were the most successful for the US and China.

China's most succesful events/disciplines vs US's most succesful events/disciplines

China Medal 2004-2016

In [161…
```
china_m=pysqldf('Select results.Year, sports.Sport, sports.Sport_ID, results.NOC_ID, results.Medal_ID, count(di
```

In [162…
```
#China's medal count and most successful events between 2004-2016
china_dis_m=pysqldf('Select Sport, (sum(gold_medal)+sum(silver_medal)+sum(bronze_medal)) as total_medals, sum(g
```

SQL Syntax */China Medal 2004-2016/*

Select

```
Sport,
(sum(gold_medal)+sum(silver_medal)+sum(bronze_medal)) as total_medals,
sum(gold_medal) as number_of_gold_medals,
sum(silver_medal) as number_of_silver_medals,
sum(bronze_medal) as number_of_bronze_medals
```

from (

```
Select
    results.Year,
    sports.Sport,
    sports.Sport_ID,
    results.NOC_ID,
    results.Medal_ID,
    count(distinct case when results.Medal_ID=1 then 1 else null end) as gold_medal,
    count(distinct case when results.Medal_ID=2 then 1 else null end) as silver_medal,
    count(distinct case when results.Medal_ID=3 then 1 else null end) as bronze_medal
from results
    left join sports
        on results.Sport_ID=sports.Sport_ID
where NOC_ID=42
and results.Season_ID != 2
and results.Year in (2004, 2008, 2012, 2016)
group by 1,2,3,4,5
order by results.Year
```

) as china_m

where Medal_ID != 4

group by 1

order by (sum(gold_medal)+sum(silver_medal)+sum(bronze_medal)) desc;

In [163…
```
china_dis_m.head(10)
```

Out[163]:

	Sport	total_medals	number_of_gold_medals	number_of_silver_medals	number_of_bronze_medals
0	Table Tennis Men's Singles	9	3	4	2
1	Table Tennis Women's Singles	8	4	3	1
2	Diving Women's Springboard	8	4	3	1
3	Trampolining Men's Individual	6	2	1	3
4	Diving Women's Platform	6	3	2	1
5	Diving Men's Springboard	6	3	1	2
6	Badminton Women's Singles	6	3	2	1
7	Trampolining Women's Individual	5	1	1	3
8	Shooting Women's Air Rifle, 10 metres	5	2	1	2
9	Diving Men's Platform	5	2	2	1

An overview in which disciplines China succeded in the last four summer olympics

USA Medal 2004-2016

In [164…
```
usa_m_04_16=pysqldf('Select results.Year, sports.Sport, results.NOC_ID, results.Medal_ID, count(distinct case w
```

In [165…
```
#the US's medal count and most successful events between 2004-2016
usa_m_04_16_dis=pysqldf('Select Sport, (sum(gold_medal)+sum(silver_medal)+sum(bronze_medal)) as total_medals, s
```

SQL Syntax */USA Medal 2004-2016/*

Select

```
Sport,
(sum(gold_medal)+sum(silver_medal)+sum(bronze_medal)) as total_medals,
sum(gold_medal) as number_of_gold_medals,
sum(silver_medal) as number_of_silver_medals,
sum(bronze_medal) as number_of_bronze_medals
```

from (

```
Select
    results.Year,
    sports.Sport,
    results.NOC_ID,
    results.Medal_ID,
    count(distinct case when results.Medal_ID=1 then 1 else null end) as gold_medal,
    count(distinct case when results.Medal_ID=2 then 1 else null end) as silver_medal,
    count(distinct case when results.Medal_ID=3 then 1 else null end) as bronze_medal
from results
    left join sports
        on results.Sport_ID=sports.Sport_ID
where NOC_ID=217
and results.Year in (2004, 2008, 2012, 2016)
and results.Season_ID!=2
group by 1,2,3,4
order by results.Year
```

) as usa_m_04_16

where Medal_ID != 4

group by 1

order by (sum(gold_medal)+sum(silver_medal)+sum(bronze_medal)) desc;

In [166…
```
usa_m_04_16_dis.head(10)
```

Out[166]:

	Sport	total_medals	number_of_gold_medals	number_of_silver_medals	number_of_bronze_medals
0	Athletics Women's 100 metres Hurdles	8	3	2	3
1	Swimming Men's 200 metres Individual Medley	7	4	2	1
2	Swimming Men's 100 metres Backstroke	7	4	2	1
3	Athletics Men's 400 metres	7	2	2	3
4	Swimming Men's 400 metres Individual Medley	6	3	2	1
5	Swimming Men's 200 metres Backstroke	6	4	1	1
6	Gymnastics Women's Individual All-Around	6	4	2	0
7	Gymnastics Women's Balance Beam	6	1	3	2
8	Beach Volleyball Women's Beach Volleyball	6	3	1	2
9	Swimming Women's 400 metres Freestyle	5	1	2	2

Here we have the most successful event for the USA in the last four summer games.

The next step was to check if there were no disciplines they had in common.

In [167…
```
#merging with an inner join the US medal table with the chinese medal table to see what they have in common
in_com=pd.merge(usa_m_04_16_dis['Sport'], china_dis_m['Sport'], how='inner')
```

In [168…
```
#events/disciplines in common
in_com
```

Out[168]:

	Sport
0	Swimming Men's 200 metres Individual Medley
1	Swimming Men's 100 metres Backstroke
2	Gymnastics Women's Individual All-Around
3	Gymnastics Women's Balance Beam
4	Beach Volleyball Women's Beach Volleyball
...	...
68	Athletics Women's Shot Put
69	Athletics Women's Marathon
70	Athletics Women's Discus Throw
71	Athletics Women's 10,000 metres
72	Archery Men's Individual

73 rows × 1 columns

These are the disciplines the USA and China had in common in the last four summer games. Knowing that challenges the hypothesis already. China had 143 medal-winning events, 73 of which they had in common with the US. Therefore I checked the gold, silver and bronze medals to determine which Nation was more successful.

In [169…
```
#merging the US medal table with table of similar events with China
usa_vs_china=usa_m_04_16_dis.merge(in_com, how='inner')
```

In [170…
```
#merging the chinese medal table with table of common events with the US
china_vs_usa=china_dis_m.merge(in_com, how='inner')
```

In [171…
```
usa_vs_china.sort_values('Sport', inplace=True)
```

In [172…
```
china_vs_usa.sort_values('Sport', inplace=True)
```

In [173…
```
usa_vs_china.rename(columns={'total_medals':'total_medals_us'}, inplace=True)
```

I merged the US and China table with the table of the events they had in common and eventually added and edited the new columns so that I could compare the both nations with one another.

In [174…
```
#adding the medal count of each event that China and the US have in common to the US table
usa_vs_china[['total_medals_china', 'number_of_gold_medals_china', 'number_of_silver_medals_china',
              'number_of_bronze_medals_china']]=china_vs_usa[['total_medals', 'number_of_gold_medals',
                'number_of_silver_medals', 'number_of_bronze_medals']]
```

In [175…
```
#realign(reindex the cloumns)
usa_vs_china.head()
com_usa_china=usa_vs_china.reindex(columns=
['Sport', 'total_medals_us', 'total_medals_china', 'number_of_gold_medals', 'number_of_gold_medals_china',
 'number_of_silver_medals', 'number_of_silver_medals_china','number_of_bronze_medals',
 'number_of_bronze_medals_china'])
```

In [176…
```
com_usa_china
```

Out[176]:

	Sport	total_medals_us	total_medals_china	number_of_gold_medals	number_of_gold_medals_china	number_of_silver_me
72	Archery Men's Individual	1	1	0	0	
43	Archery Men's Team	2	1	0	0	
7	Athletics Men's 110 metres Hurdles	5	3	1	0	
17	Athletics Men's Triple Jump	4	2	2	2	
71	Athletics Women's 10,000 metres	1	1	0	0	
...	...	...	...	...	...	
28	Tennis Women's Doubles	2	2	2	0	
18	Volleyball Women's Volleyball	3	2	0	0	
44	Weightlifting Women's Super-Heavyweight	1	1	0	0	
27	Wrestling Women's Flyweight, Freestyle	2	2	0	1	
26	Wrestling Women's Middleweight, Freestyle	2	2	0	1	

73 rows × 9 columns

In [177…
```
#checking if China has any event where they have more medals total than the US
com_usa_china[com_usa_china.total_medals_us<com_usa_china.total_medals_china]
```

Out[177]:

Sport	total_medals_us	total_medals_china	number_of_gold_medals	number_of_gold_medals_china	number_of_silver_medals	num

China had no event/discipline where they gathered more medals overall than the US.

In [178…
```
#checking if the US has any event where they have more medals total than China
com_usa_china[com_usa_china.total_medals_us>com_usa_china.total_medals_china]
```

Out[178]:

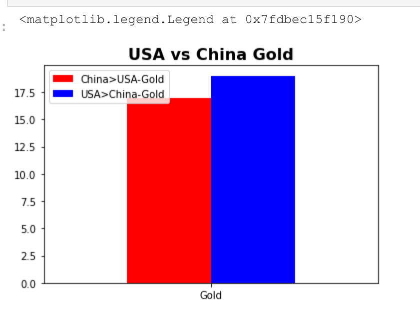
	Sport	total_medals_us	total_medals_china	number_of_gold_medals	number_of_gold_medals_china	number_of_silver_me
43	Archery Men's Team	2	1	0	0	
7	Athletics Men's 110 metres Hurdles	5	3	1	0	
17	Athletics Men's Triple Jump	4	2	2	2	
4	Beach Volleyball Women's Beach Volleyball	6	4	3	4	
42	Boxing Women's Middleweight	2	1	2	0	
41	Diving Men's Platform	2	1	1	1	
40	Diving Men's Synchronized Platform	2	1	0	0	
39	Diving Men's Synchronized Springboard	2	1	0	0	
6	Fencing Women's Sabre, Individual	5	3	2	2	
38	Fencing Women's Sabre, Team	2	1	0	0	
25	Gymnastics Men's Horizontal Bar	3	2	0	1	
37	Gymnastics Men's Individual All-Around	2	1	1	1	
36	Gymnastics Men's Team All-Around	2	1	0	0	
3	Gymnastics Women's Balance Beam	6	4	1	4	
24	Gymnastics Women's Horse Vault	3	2	1	0	
2	Gymnastics Women's Individual All-Around	6	4	4	1	
16	Gymnastics Women's Team All-Around	4	2	2	1	
15	Gymnastics Women's Uneven Bars	4	3	0	2	
35	Judo Women's Half-Heavyweight	2	1	2	0	
23	Shooting Women's Skeet	3	2	1	2	
1	Swimming Men's 100 metres Backstroke	7	5	4	2	
14	Swimming Men's 200 metres Freestyle	4	3	1	2	
0	Swimming Men's 200 metres Individual Medley	7	5	4	2	
13	Swimming Men's 4 x 200 metres Freestyle Relay	4	3	4	2	
22	Swimming Men's 400 metres Freestyle	3	2	0	0	
5	Swimming Women's 100 metres Backstroke	5	3	3	3	
12	Swimming Women's 100 metres Breaststroke	4	3	1	2	
21	Swimming Women's 100 metres Butterfly	3	2	1	1	
20	Swimming Women's 100 metres Freestyle	3	2	1	0	
19	Swimming Women's 200 metres Breaststroke	3	2	3	1	
11	Swimming Women's 200 metres Individual Medley	4	3	0	1	
10	Swimming Women's 4 x 100 metres Medley Relay	4	3	2	1	
9	Swimming Women's 4 x 200 metres Freestyle Relay	4	3	3	2	
8	Swimming Women's 400 metres Individual Medley	4	3	0	1	
18	Volleyball Women's Volleyball	3	2	0	0	

But lets see how each nation perferemd based on the medal type:

USA vs China Gold

In [179…
```
#checking if China has any event where they have more gold medals than the US
china_g=len(com_usa_china[com_usa_china.number_of_gold_medals<com_usa_china.number_of_gold_medals_china])
#checking if the US has any event where they have more gold medals than China
us_g=len(com_usa_china[com_usa_china.number_of_gold_medals>com_usa_china.number_of_gold_medals_china])
gold_diff=pd.DataFrame({'China>USA-Gold':china_g, 'USA>China-Gold':us_g}, index=['Gold'])
gold_diff.plot(kind='bar', color=['red', 'blue'])
plt.xticks(rotation=0)
plt.title('USA vs China Gold', size=16, fontweight='bold')
plt.legend(loc='upper left')
```

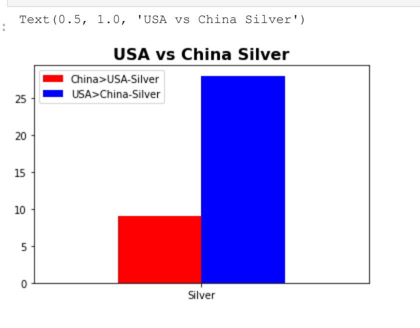
Out[179]: `<matplotlib.legend.Legend at 0x7fdbec15f190>`



USA vs China Silver

In [180…
```
#checking if China has any event where they have more silver medals than the US
china_sil=len(com_usa_china[com_usa_china.number_of_silver_medals<com_usa_china.number_of_silver_medals_china])
#checking if the US has any event where they have more silver medals than China
usa_sil=len(com_usa_china[com_usa_china.number_of_silver_medals>com_usa_china.number_of_silver_medals_china])
silver_diff=pd.DataFrame({'China>USA-Silver':china_sil, 'USA>China-Silver':usa_sil}, index=['Silver'])
silver_diff.plot(kind='bar', color=['red', 'blue'])
plt.xticks(rotation=0)
plt.title('USA vs China Silver', size=16, fontweight='bold')
```

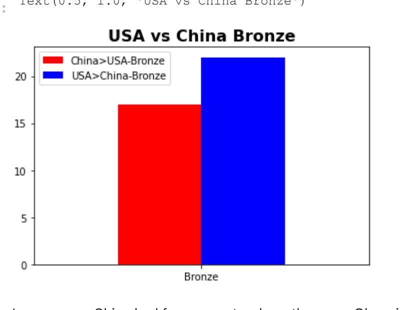
Out[180]: `Text(0.5, 1.0, 'USA vs China Silver')`



USA vs China Bronze

In [181…
```
#checking if China has any event where they have more bronze medals than the US
china_b=len(com_usa_china[com_usa_china.number_of_bronze_medals<com_usa_china.number_of_bronze_medals_china])
#checking if the US has any event where they have more bronze medals than China
usa_b=len(com_usa_china[com_usa_china.number_of_bronze_medals>com_usa_china.number_of_bronze_medals_china])
bronze_diff=pd.DataFrame({'China>USA-Bronze':china_b, 'USA>China-Bronze':usa_b}, index=['Bronze'])
bronze_diff.plot(kind='bar', color=['red', 'blue'])
plt.xticks(rotation=0)
plt.title('USA vs China Bronze', size=16, fontweight='bold')
```

Out[181]: `Text(0.5, 1.0, 'USA vs China Bronze')`



In summary, China had fewer events where they won Olympic medals (143=China; 173=USA), which would support the hypothesis that China was a runner-up because they won fewer medals. Additionally I have discovered that the USA and China had 73 events in common by analysing both profiles. Here, I could observe that the US always came out on top of China in all medal types. What does that mean for the hypothesis?

Hypothesis=> China was a runner-up because it succeed in fewer events than the most successful nation, but these events are unique to the ones of the most successful nation

All in all, China came short of the US and had more common disciplines than unique ones (51% in common with the US). Therefore the hypothesis is false and needs to be rejected.

Most Succesful Athlete in Olympic History

In [182…
```
#getting a medal count of the 5 most succesful athletes, on a overall medal level
best_ath=pysqldf('select athlete.Name, nationality.NOC, athlete.Sex, count(results.Medal_ID) as number_of_medal
```

In [183…
```
best_ath.set_index('Name', inplace=True)
```

SQL Syntax */Most Successful Athlete in Olympic History/*

select

```
athlete.Name,
nationality.NOC,
athlete.Sex,
count(results.Medal_ID) as number_of_medals,
count(case when results.Medal_ID=1 then 1 else null end) as number_of_gold_medals,
count(case when results.Medal_ID=2 then 1 else null end) as number_of_silver_medals,
count(case when results.Medal_ID=3 then 1 else null end) as number_of_bronze_medals
```

from results

left join athlete

```
on results.Athlete_ID=athlete.Athlete_ID
```

left join nationality

```
on results.NOC_ID=nationality.NOC_ID
```

where results.Medal_ID in (1,2,3)

group by 1,2,3

order by count(results.Medal_ID) desc limit 5;

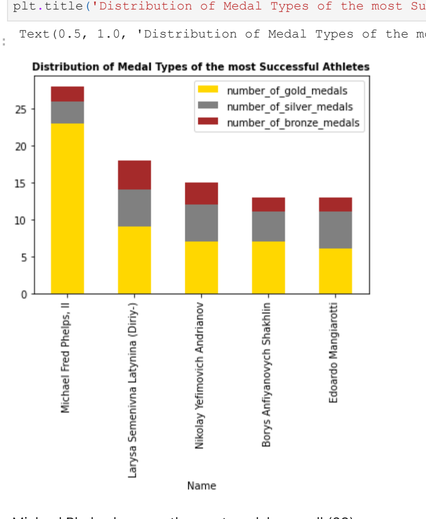
In [184…
```
#table of most successful athletes based on medal count
best_ath
```

Out[184]:

Name	NOC	Sex	number_of_medals	number_of_gold_medals	number_of_silver_medals	number_of_bronze_medals
Michael Fred Phelps, II	USA	M	28	23	3	2
Larysa Semenivna Latynina (Diriy-)	URS	F	18	9	5	4
Nikolay Yefimovich Andrianov	URS	M	15	7	5	3
Borys Anfiyanovych Shakhlin	URS	M	13	7	4	2
Edoardo Mangiarotti	ITA	M	13	6	5	2

In [185…
```
best_ath[['number_of_gold_medals', 'number_of_silver_medals', 'number_of_bronze_medals']].plot(kind='bar',
                                                                    stacked=True, color=['gold', 'grey', 'brown'])
plt.title('Distribution of Medal Types of the most Successful Athletes', size=10, fontweight='bold')
```

Out[185]: `Text(0.5, 1.0, 'Distribution of Medal Types of the most Successful Athletes')`



Michael Phelps has won the most medals overall (28).

Athletes with the Gold Medals

In [186…
```
#most successful gold medal winning athletes
best_ath_gold=pysqldf('select athlete.Name, nationality.NOC, athlete.Sex, count(results.Medal_ID) as number_of_
best_ath_gold
```

Out[186]:

	Name	NOC	Sex	number_of_gold_medals
0	Michael Fred Phelps, II	USA	M	23
1	Raymond Clarence "Ray" Ewry	USA	M	10
2	Frederick Carlton "Carl" Lewis	USA	M	9
3	Larysa Semenivna Latynina (Diriy-)	URS	F	9
4	Mark Andrew Spitz	USA	M	9

SQL Syntax */Most successful Gold winning athlete/*

select

```
athlete.Name,
nationality.NOC,
athlete.Sex,
count(results.Medal_ID) as number_of_gold_medals
```

from results

```
left join athlete
    on results.Athlete_ID=athlete.Athlete_ID
left join nationality
    on results.NOC_ID=nationality.NOC_ID
```

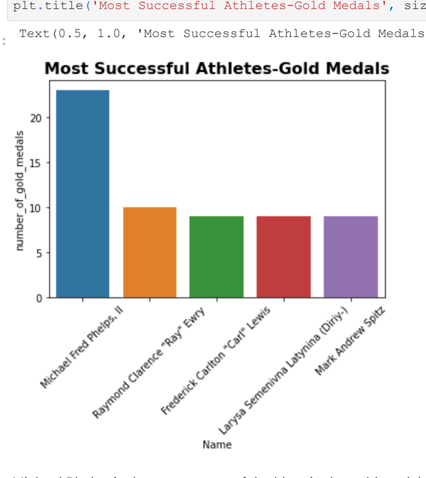
where results.Medal_ID=1

group by 1,2,3

order by count(results.Medal_ID) desc limit 5;

In [187…
```
sns.barplot(x='Name', y='number_of_gold_medals', data=best_ath_gold)
plt.xticks(rotation=45)
plt.title('Most Successful Athletes-Gold Medals', size=16, fontweight='bold')
```

Out[187]: `Text(0.5, 1.0, 'Most Successful Athletes-Gold Medals')`



Micheal Phelps is the most successful athlete in the gold medal segment.

Conversion Rate of most Successful Athletes and Medals won

In [188…
```
#CVR overall Medals of the most succesful athletes
ath_per=pysqldf('Select athlete.Name, nationality.NOC, athlete.Sex, count(case when Medal_ID!=4 then 1 else nul
```

SQL Syntax */CVR overall Medals of the most succesful athletes/*

Select

```
athlete.Name,
nationality.NOC,
athlete.Sex,
count(case when Medal_ID!=4 then 1 else null end) as won_medal,
count(Medal_ID) as participation,
round(count(case when Medal_ID!=4 then 1 else null end)/count(Medal_ID),2) as CVR_Overall
```

from results

```
left join athlete
    on results.Athlete_ID=athlete.Athlete_ID
left join nationality
    on results.NOC_ID=nationality.NOC_ID
```

where athlete.Athlete_ID in (86767, 73148, 93760, 16196, 29505)

and results.Sport_ID in (645, 640, 638, 632, 646, 644, 645, 652, 639, 591, 589, 592, 590, 376, 377, 369, 366, 374, 368, 365, 380, 389, 388, 387, 385, 384, 386, 390, 326, 327, 333, 336, 325)

group by 1,2,3;

In [189…
```
ath_per.head()
```

Out[189]:

	Name	NOC	Sex	won_medal	participation
0	Borys Anfiyanovych Shakhlin	URS	M	13	24
1	Edoardo Mangiarotti	ITA	M	13	14
2	Larysa Semenivna Latynina (Diriy-)	URS	F	18	19
3	Michael Fred Phelps, II	USA	M	28	30
4	Nikolay Yefimovich Andrianov	URS	M	15	24

In [190…
```
#adding CVR rate of every athlete
ath_per['CVR_Overall']=round(ath_per.won_medal/ath_per.participation,4)
```

In [191…
```
ath_per.set_index('Name', inplace=True)
```

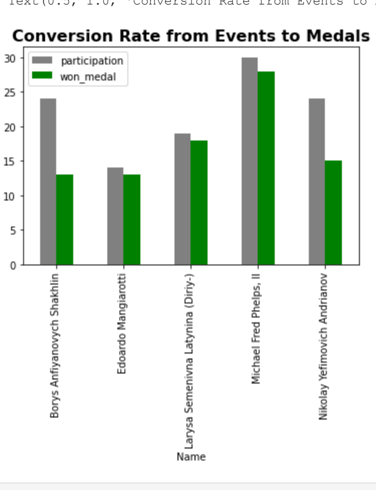
In [192…
```
#most succesful athletes based on medals and CVR
ath_per
```

Out[192]:

Name	NOC	Sex	won_medal	participation	CVR_Overall
Borys Anfiyanovych Shakhlin	URS	M	13	24	0.5417
Edoardo Mangiarotti	ITA	M	13	14	0.9286
Larysa Semenivna Latynina (Diriy-)	URS	F	18	19	0.9474
Michael Fred Phelps, II	USA	M	28	30	0.9333
Nikolay Yefimovich Andrianov	URS	M	15	24	0.6250

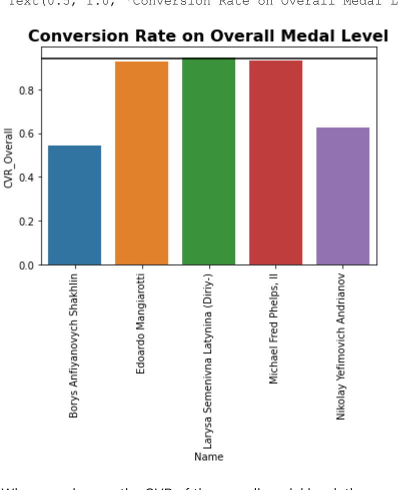
In [193…
```
ath_per[['participation', 'won_medal']].plot(kind='bar', color=['grey', 'green'])
plt.title('Conversion Rate from Events to Medals', size=16, fontweight='bold')
```

Out[193]: Text(0.5, 1.0, 'Conversion Rate from Events to Medals')



In [194…
```
sns.barplot(x='Name', y='CVR_Overall', data=ath_per.reset_index())
plt.xticks(rotation=90)
plt.axhline(y=.94, color='black', linestyle='-')
plt.title('Conversion Rate on Overall Medal Level', size=16, fontweight='bold')
```

Out[194]: Text(0.5, 1.0, 'Conversion Rate on Overall Medal Level')



When we observe the CVR of the overall medal level, then we can see that Micheal Phelps is not the most successfull athlete. Based on the CVR, Larysa Semenivna Latynina (Diriy-) is the most successful athlete.

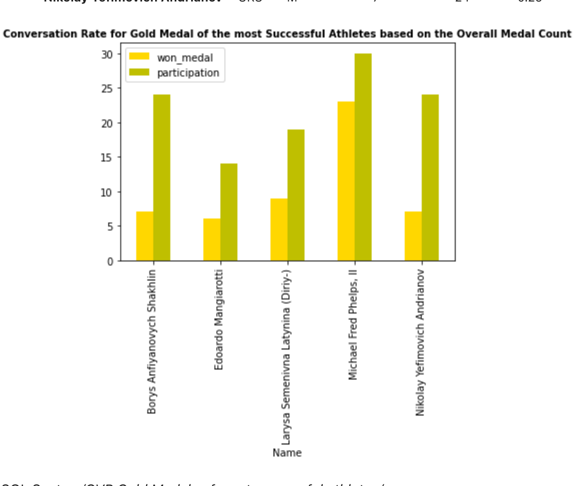
Conversion Rate of most Successful Athletes and Gold Medals won

Based on the overall medal count, Michael Phelps is the most successful athlete. But looking at the conversation rate a different picture is observable. In the overall medal conversation rate ranking, Micheal Phelps is just a runner-up to Larysa Semenivna Latynina (Diriy-).

In [195…
```
#table of most succesful medal winning athletes and gold medal CVR
ath_per_gold=pysqldf('Select athlete.Name, nationality.NOC, athlete.Sex, count(case when Medal_ID=1 then 1 else
#adding the Conversion rate
ath_per_gold['CVR_Gold']=round(ath_per_gold.won_medal/ath_per_gold.participation, 2)
ath_per_gold.set_index('Name', inplace=True)
ath_per_gold[['won_medal', 'participation']].plot(kind='bar', color=['gold', 'y'])
plt.title('Conversation Rate for Gold Medal of the most Successful Athletes based on the Overall Medal Count',
ath_per_gold
```

Out[195]:

Name	NOC	Sex	won_medal	participation	CVR_Gold
Borys Anfiyanovych Shakhlin	URS	M	7	24	0.29
Edoardo Mangiarotti	ITA	M	6	14	0.43
Larysa Semenivna Latynina (Diriy-)	URS	F	9	19	0.47
Michael Fred Phelps, II	USA	M	23	30	0.77
Nikolay Yefimovich Andrianov	URS	M	7	24	0.29



SQL Syntax */CVR Gold Medals of most successful athletes/*

Select

```
athlete.Name,
nationality.NOC,
athlete.Sex,
count(case when Medal_ID=1 then 1 else null end) as won_medal,
count(Medal_ID) as participation
round(count(case when Medal_ID=1 then 1 else null end)/count(Medal_ID),2) as CVR-Gold
```

from results

```
left join athlete
    on results.Athlete_ID=athlete.Athlete_ID
left join nationality
    on results.NOC_ID=nationality.NOC_ID
```

where athlete.Athlete_ID in (86767, 73148, 93760, 16196, 29505)

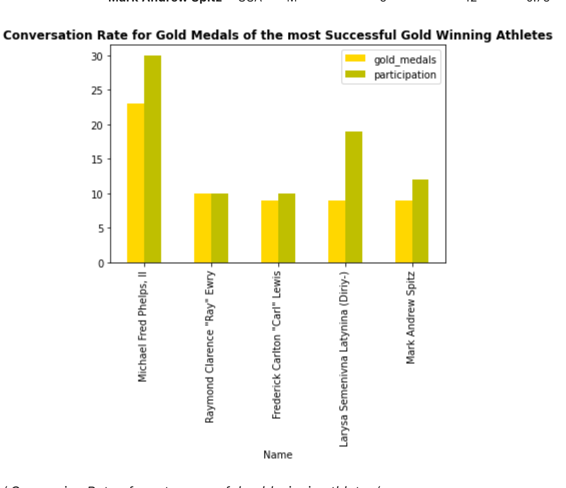
and results.Sport_ID in (645, 640, 638, 632, 646, 644, 645, 652, 639, 591, 589, 592, 590, 376, 377, 369, 366, 374, 368, 365, 380, 389, 388, 387, 385, 384, 386, 390, 326, 327, 333, 336, 325)

group by 1,2,3;

In [196…
```
#CVR of gold medals, between the most succesful gold winning athletes
ath_per_gold_g=pysqldf('select athlete.Name, nationality.NOC, athlete.Sex, count(case when results.Medal_ID=1 t
ath_per_gold_g.set_index('Name', inplace=True)
ath_per_gold_g['CVR_Gold']=round(ath_per_gold_g.gold_medals/ath_per_gold_g.participation, 2)
ath_per_gold_g[['gold_medals', 'participation']].plot(kind='bar', color=['gold', 'y'])
plt.title('Conversation Rate for Gold Medals of the most Successful Gold Winning Athletes', size=12, fontweight
ath_per_gold_g
```

Out[196]:

Name	NOC	Sex	gold_medals	participation	CVR_Gold
Michael Fred Phelps, II	USA	M	23	30	0.77
Raymond Clarence "Ray" Ewry	USA	M	10	10	1.00
Frederick Carlton "Carl" Lewis	USA	M	9	10	0.90
Larysa Semenivna Latynina (Diriy-)	URS	F	9	19	0.47
Mark Andrew Spitz	USA	M	9	12	0.75



/ Conversion Rate of most successful gold winnig athletes/

select

```
athlete.Name,
nationality.NOC,
athlete.Sex,
count(case when results.Medal_ID=1 then 1 else null end) as gold_medals,
count(results.Medal_ID) as participation,
round(count(case when results.Medal_ID=1 then 1 else null end)/count(results.Medal_ID),2) as
CVR-Gold
```

from results

```
inner join athlete
    on results.Athlete_ID=athlete.Athlete_ID
inner join nationality
    on results.NOC_ID=nationality.NOC_ID
```

group by 1,2,3

having count(case when results.Medal_ID=1 then 1 else null end) >0

order by count(case when results.Medal_ID=1 then 1 else null end) desc limit 5

Comparing both gold medal CVR tables, we see two different pictures. We can see in the graph of the athletes with the most overall medals that Michael Phelps has the most gold medals and the best CVR of 77%

But, when we look at the gold medal CVR of the most successful gold medal athletes, we see that Michael Phelps has the most gold medals but the third best CVR, Raymond Clarence "Ray" Ewry (100%) and Frederick Carlton "Carl" Lewis (90%). At this point, it just depends on the interpretation of these numbers and values. We wanted to see who had the highest count of medals. Therefore Michael Phelps is considered as the most successful athlete.

Hypothesis => The most successful athletes are men

- A men will hold the most gold medals

The Hypothesis is right because Michael Phelps hast the highest overall medal and gold medal count

What Role does Age play in Winning a Olympic Medal?

An essential question I wanted to answer in this case study was, if there is a correlation between age and winning an Olympic medal. First, we need to understand what age range we are dealing with.

In [197…
```
#youngest and oldest particpants
pysqldf('select min(Age), max(Age) from results;')
```

Out[197]:

	min(Age)	max(Age)
0	10	97

The youngest olympian participant was 10 years old, and the oldest 97. First, the dataset seems faulty. Therefore we check what discipline and what year the participants participated.

In [198…
```
#checking if the weird age range is realistic
pysqldf('select *, sports.Sport from results left join sports on results.Sport_ID=sports.Sport_ID where results
```

Out[198]:

	Athlete_ID	Age	Height	Weight	NOC_ID	Games_ID	Year	Season_ID	City	Sport_ID	Medal_ID	Sex_ID	Sport_ID	
0	27349	10	176	67	82	1	1896	1	Athina	375	3	1	375	Gym Parall
1	40797	10	176	67	82	1	1896	1	Athina	633	4	1	633	Sw Me Fr
2	44695	10	176	67	91	1	1896	1	Athina	376	4	1	376	Gym Pom
3	61364	97	176	67	217	10	1928	1	Amsterdam	68	4	1	68	Comp Sculp

The height and weight are unrealistic due to the ffill() function used during the formatting phase. Nonetheless, these values could be used for the age analysis but could not for examining of the height and weight.

Let's get an idea of how the age of all participants is distributed.

In [199…
```
#1=Male, 2=Female
age_groups=pysqldf('Select count(case when Age >=10 and Age <=19 then Medal_ID else null end) as age_10_19, cou
```

SQL Syntax */Age distribution of participants/*

Select

```
count(case when Age >=10 and Age <=19 then Medal_ID else null end) as age_10_19,
count(case when Age >=20 and Age <=29 then Medal_ID else null end) as age_20_29,
count(case when Age >=30 and Age <=39 then Medal_ID else null end) as age_30_39,
count(case when Age >=40 and Age <=49 then Medal_ID else null end) as age_40_49,
count(case when Age >=50 and Age <=59 then Medal_ID else null end) as age_50_59,
count(case when Age >=60 and Age <=69 then Medal_ID else null end) as age_60_69,
count(case when Age >=70 and Age <=79 then Medal_ID else null end) as age_70_79,
count(case when Age >=80 and Age <=89 then Medal_ID else null end) as age_80_89,
count(case when Age >=90 and Age <=99 then Medal_ID else null end) as age_90_99
```

from results;

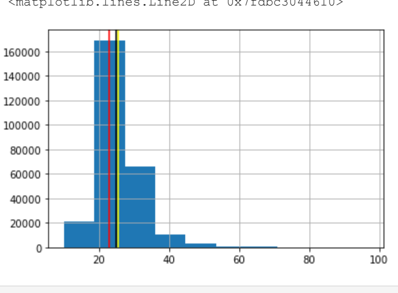
In [200…
```
age_groups
```

Out[200]:

	age_10_19	age_20_29	age_30_39	age_40_49	age_50_59	age_60_69	age_70_79	age_80_89	age_90_99
0	33228	183537	44541	7305	1742	602	136	9	2

In [201…
```
#histogram age
results.Age.hist()
plt.axvline(x=23, color='red')
plt.axvline(x=25, color='Black')
plt.axvline(x=25.628709, color='yellow')
```

Out[201]: <matplotlib.lines.Line2D at 0x7fdbc3044610>



In [202…
```
results.Age.describe()
```

Out[202]:
```
count    271102.000000
mean         25.628933
std           6.495034
min          10.000000
25%          21.000000
50%          25.000000
75%          28.000000
max          97.000000
Name: Age, dtype: float64
```

In [203…
```
import statistics
statistics.mode(results.Age)
```

Out[203]: 23

In [204…
```
#calculating median
statistics.median(results.Age)
```

Out[204]: 25.0

As we can see, the Age distribution is right-skewed. This means most of the data is behind the mean, indicating that most participants are below 26 years old.

Next, we wanted to understand the age profile of medal-winning athletes.

In [205…
```
#age range of medal winnig athletes
pysqldf('select min(Age), max(Age) from results where Medal_ID in (1,2,3);')
```

Out[205]:

	min(Age)	max(Age)
0	10	73

In [206…
```
#1=Male, 2=Female
age_groups_medal=pysqldf('Select count(case when Age >=10 and Age <=19 then Medal_ID else null end) as age_10_1
```

SQL Syntax */Age Groups of Medal winning athletes/*

Select

```
count(case when Age >=10 and Age <=19 then Medal_ID else null end) as age_10_19,
count(case when Age >=20 and Age <=29 then Medal_ID else null end) as age_20_29,
count(case when Age >=30 and Age <=39 then Medal_ID else null end) as age_30_39,
count(case when Age >=40 and Age <=49 then Medal_ID else null end) as age_40_49,
count(case when Age >=50 and Age <=59 then Medal_ID else null end) as age_50_59,
count(case when Age >=60 and Age <=69 then Medal_ID else null end) as age_60_69,
count(case when Age >=70 and Age <=79 then Medal_ID else null end) as age_70_79
```

from results

where Medal_ID in (1,2,3);

In [207…
```
age_groups.index=['all_athletes']
```

In [208…
```
#age distribution of non-medal winning atheltes
age_groups
```

Out[208]:

	age_10_19	age_20_29	age_30_39	age_40_49	age_50_59	age_60_69	age_70_79	age_80_89	age_90_99
all_athletes	33228	183537	44541	7305	1742	602	136	9	2

In [209…
```
age_groups_medal.index=['medal_winning_athletes']
```

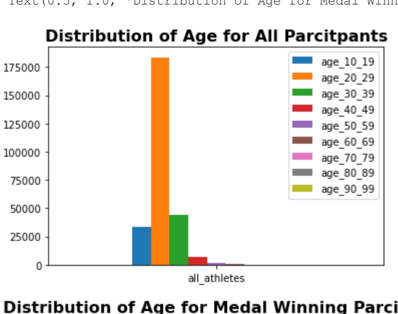
In [210…
```
#age distribution of medal winning atheltes
age_groups_medal
```

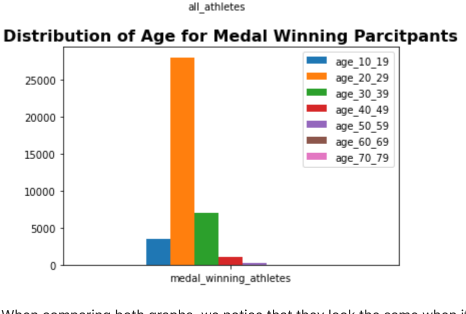
Out[210]:

	age_10_19	age_20_29	age_30_39	age_40_49	age_50_59	age_60_69	age_70_79
medal_winning_athletes	3481	28002	7003	1039	209	44	5

In [211…
```
age_groups.plot(kind='bar')
plt.xticks(rotation=0)
plt.title('Distribution of Age for All Parcitpants', size=16, fontweight='bold')
age_groups_medal.plot(kind='bar')
plt.xticks(rotation=0)
plt.title('Distribution of Age for Medal Winning Parcitpants', size=16, fontweight='bold')
```

Out[211]: Text(0.5, 1.0, 'Distribution of Age for Medal Winning Parcitpants')





When comparing both graphs, we notice that they look the same when it comes to the distribution of age. The only difference is that there are no medal-winning athletes in their 80s and 90s.

By analyzing the medal count of the age distribution, you could say you are most likely to win a medal in your twenties. But, would we see a similar picture regarding gold, silver and bronze medals?

Gold Medal

In [212…
```
#age distribution bsed on gold winning athletes
age_groups_gold=pysqldf('Select count(case when Age >=10 and Age <=19 then Medal_ID else null end) as age_10_19
age_groups_silver=pysqldf('Select count(case when Age >=10 and Age <=19 then Medal_ID else null end) as age_10_
age_groups_bronze=pysqldf('Select count(case when Age >=10 and Age <=19 then Medal_ID else null end) as age_10_
```

SQL Syntax */Age group distribution of Gold Medals/*

Select

```
count(case when Age >=10 and Age <=19 then Medal_ID else null end) as age_10_19,
count(case when Age >=20 and Age <=29 then Medal_ID else null end) as age_20_29,
count(case when Age >=30 and Age <=39 then Medal_ID else null end) as age_30_39,
count(case when Age >=40 and Age <=49 then Medal_ID else null end) as age_40_49,
count(case when Age >=50 and Age <=59 then Medal_ID else null end) as age_50_59,
count(case when Age >=60 and Age <=69 then Medal_ID else null end) as age_60_69,
count(case when Age >=70 and Age <=79 then Medal_ID else null end) as age_70_79
```

from results

where Medal_ID=1;

SQL Syntax */Age group distribution of Silver Medals/*

Select

```
count(case when Age >=10 and Age <=19 then Medal_ID else null end) as age_10_19,
count(case when Age >=20 and Age <=29 then Medal_ID else null end) as age_20_29,
count(case when Age >=30 and Age <=39 then Medal_ID else null end) as age_30_39,
count(case when Age >=40 and Age <=49 then Medal_ID else null end) as age_40_49,
count(case when Age >=50 and Age <=59 then Medal_ID else null end) as age_50_59,
count(case when Age >=60 and Age <=69 then Medal_ID else null end) as age_60_69,
count(case when Age >=70 and Age <=79 then Medal_ID else null end) as age_70_79
```

from results

where Medal_ID=2;

SQL Syntax */Age group distribution of Bronze Medals/*

Select

```
count(case when Age >=10 and Age <=19 then Medal_ID else null end) as age_10_19,
count(case when Age >=20 and Age <=29 then Medal_ID else null end) as age_20_29,
count(case when Age >=30 and Age <=39 then Medal_ID else null end) as age_30_39,
count(case when Age >=40 and Age <=49 then Medal_ID else null end) as age_40_49,
count(case when Age >=50 and Age <=59 then Medal_ID else null end) as age_50_59,
count(case when Age >=60 and Age <=69 then Medal_ID else null end) as age_60_69,
count(case when Age >=70 and Age <=79 then Medal_ID else null end) as age_70_79
```

from results

where Medal_ID=3;

In [213…
```
age_groups_gold.index=['gold_medals']
```

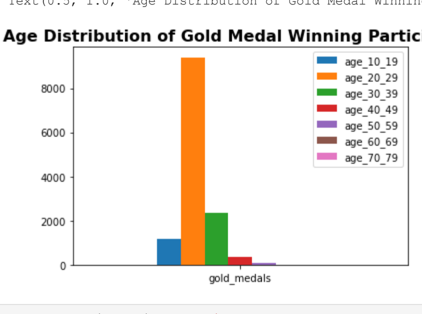
In [214…
```
#age distribution of gold medal winning atheltes
age_groups_gold
```

Out[214]:

	age_10_19	age_20_29	age_30_39	age_40_49	age_50_59	age_60_69	age_70_79
gold_medals	1179	9403	2367	343	70	10	0

In [215…
```
age_groups_gold.plot(kind='bar')
plt.xticks(rotation=0)
plt.title('Age Distribution of Gold Medal Winning Participants', size=16, fontweight='bold')
```

Out[215]: Text(0.5, 1.0, 'Age Distribution of Gold Medal Winning Participants')



In [216…
```
age_groups_silver.index=['silver_medals']
```

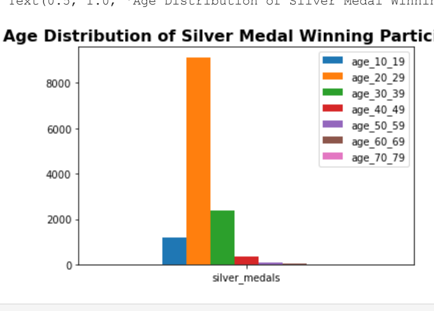
In [217…
```
#age distribution of silver medal winning atheltes
age_groups_silver
```

Out[217]:

	age_10_19	age_20_29	age_30_39	age_40_49	age_50_59	age_60_69	age_70_79
silver_medals	1165	9132	2371	350	73	22	3

In [218…
```
age_groups_silver.plot(kind='bar')
plt.xticks(rotation=0)
plt.title('Age Distribution of Silver Medal Winning Participants', size=16, fontweight='bold')
```

Out[218]: Text(0.5, 1.0, 'Age Distribution of Silver Medal Winning Participants')



In [219…
```
age_groups_bronze.index=['bronze_medals']
```

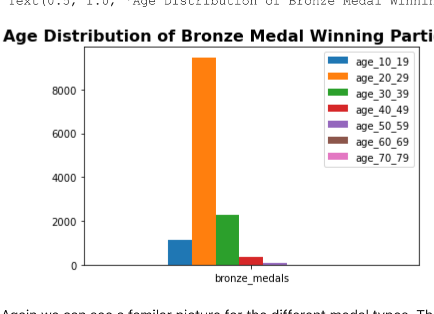
In [220…
```
#age distribution of bronze medal winning atheltes
age_groups_bronze
```

Out[220]:

	age_10_19	age_20_29	age_30_39	age_40_49	age_50_59	age_60_69	age_70_79
bronze_medals	1137	9467	2265	346	66	12	2

In [221…
```
age_groups_bronze.plot(kind='bar')
plt.xticks(rotation=0)
plt.title('Age Distribution of Bronze Medal Winning Participants', size=16, fontweight='bold')
```

Out[221]: Text(0.5, 1.0, 'Age Distribution of Bronze Medal Winning Participants')



Again we can see a familar picture for the different medal types. The strongest age group was 20-29. Can we observe a different trend between men and women?

Age Distribution Men vs Women

In [222…
```
#age distribution by gender
men_women_age=pysqldf('Select count(case when Age >=10 and Age <=19 then Medal_ID else null end) as age_10_19,
```

SQL Synatx */Age group distribution between Men and Women/*

Select

```
count(case when Age >=10 and Age <=19 then Medal_ID else null end) as age_10_19,
count(case when Age >=20 and Age <=29 then Medal_ID else null end) as age_20_29,
count(case when Age >=30 and Age <=39 then Medal_ID else null end) as age_30_39,
count(case when Age >=40 and Age <=49 then Medal_ID else null end) as age_40_49,
count(case when Age >=50 and Age <=59 then Medal_ID else null end) as age_50_59,
count(case when Age >=60 and Age <=69 then Medal_ID else null end) as age_60_69,
count(case when Age >=70 and Age <=79 then Medal_ID else null end) as age_70_79
```

from results

where Sex_ID=1

and Medal_ID in (1,2,3)

UNION

Select

```
count(case when Age >=10 and Age <=19 then Medal_ID else null end) as age_10_19,
count(case when Age >=20 and Age <=29 then Medal_ID else null end) as age_20_29,
count(case when Age >=30 and Age <=39 then Medal_ID else null end) as age_30_39,
count(case when Age >=40 and Age <=49 then Medal_ID else null end) as age_40_49,
count(case when Age >=50 and Age <=59 then Medal_ID else null end) as age_50_59,
count(case when Age >=60 and Age <=69 then Medal_ID else null end) as age_60_69,
count(case when Age >=70 and Age <=79 then Medal_ID else null end) as age_70_79
```

from results

where Sex_ID=2

and Medal_ID in (1,2,3);

In [223…
```
men_women_age.index=['Male', 'Female']
```

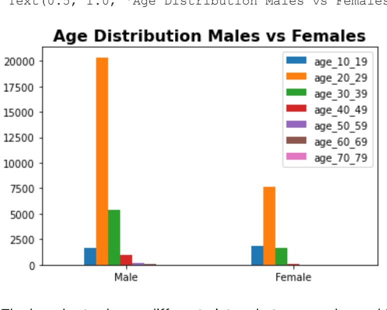
In [224…
```
#distribution of medals won by age group and gender
men_women_age
```

Out[224]:

	age_10_19	age_20_29	age_30_39	age_40_49	age_50_59	age_60_69	age_70_79
Male	1614	20353	5383	937	198	40	5
Female	1867	7649	1620	102	11	4	0

In [225…
```
men_women_age.plot(kind='bar')
plt.xticks(rotation=0)
plt.title('Age Distribution Males vs Females', size=16, fontweight='bold')
```

Out[225]: Text(0.5, 1.0, 'Age Distribution Males vs Females')



The bar charts show a different picture between males and females. Female athletes tend to be more successful in the age group 10-19 than males, but the age group, 10-19, is bigger than the female age group 30-39. Males tend to be more successful in the age group 30-39 than 10-19. As we already have established, females had fewer events than males, and they did not participate for the entire 120 years. Nonetheless, the age distribution trend was fascinating.

After getting an idea of how the age is distributed, I wanted to understand the relationship between age and medals won. To be more specific, does age impact the number of medals won? I focused mainly on the summer game athletes because they provide the most usable data.

Correlation

Correlation between Age and Medals won

In [226…
```
#table with aggreated medals counts based on athlete and age
age_med=pysqldf('Select Athlete_ID, Age, medals_won, gold_won, silver_won, bronze_won from (select Athlete_ID, 
```

SQL Synatx */Correlation between Age and all types of Medals won/*

Select

```
Athlete_ID,
Age,
medals_won,
gold_won,
silver_won,
bronze_won
```

from (

select

```
Athlete_ID,
Age,
count(case when Medal_ID!=4 then 1 else null end) as medals_won,
count(case when Medal_ID=1 then 1 else null end) as gold_won,
count(case when Medal_ID=2 then 1 else null end) as silver_won,
count(case when Medal_ID=3 then 1 else null end) as bronze_won
```

from results

where Medal_ID in (1,2,3)

and Season_ID=1

group by 1,2

) as dis_age_med

where medals_won > 0;

In [227…
```
#proof that each athlete can appear with a different count as they progress in age
age_med[age_med.Athlete_ID==86767]
```

Out[227]:

	Athlete_ID	Age	medals_won	gold_won	silver_won	bronze_won
19471	86767	19	8	6	0	2
19472	86767	23	8	8	0	0
19473	86767	27	6	4	2	0
19474	86767	31	6	5	1	0

In [228…
```
#proof that no athlete appears with a medals_won count of 0
age_med[age_med.medals_won==0]
```

Out[228]:

Athlete_ID	Age	medals_won	gold_won	silver_won	bronze_won

In [229…
```
#table with aggreated medals counts based on athlete and age
age_med
```

Out[229]:

	Athlete_ID	Age	medals_won	gold_won	silver_won	bronze_won
0	5	27	1	1	0	0
1	11	21	1	0	1	0
2	11	22	1	0	0	1
3	21	31	1	0	1	0
4	26	54	1	0	1	0
...	...	...	...	...	...	...
30364	134749	27	1	0	1	0
30365	134751	26	1	1	0	0
30366	134753	25	1	1	0	0
30367	134753	29	1	0	1	0
30368	134765	20	1	1	0	0

30369 rows × 6 columns

As we just saw, 20-29 is the most dominant age group compared to all the other age groups.

To get a fair comparison/analysis, I have randomly sample 50 data points of every age. The code below shows the final version. After 46 years, I could not get any sample of 50 anymore.

```python
#using concat to to combine the data frames
age_meds=pd.concat([age_med[age_med.Age==14].sample(n=50), age_med[age_med.Age==15].sample(n=50),
                    age_med[age_med.Age==16].sample(n=50), age_med[age_med.Age==17].sample(n=50),
                    age_med[age_med.Age==18].sample(n=50), age_med[age_med.Age==19].sample(n=50),
                    age_med[age_med.Age==20].sample(n=50), age_med[age_med.Age==21].sample(n=50),
                    age_med[age_med.Age==22].sample(n=50), age_med[age_med.Age==23].sample(n=50),
                    age_med[age_med.Age==24].sample(n=50), age_med[age_med.Age==25].sample(n=50),
                    age_med[age_med.Age==26].sample(n=50), age_med[age_med.Age==27].sample(n=50),
                    age_med[age_med.Age==28].sample(n=50), age_med[age_med.Age==29].sample(n=50),
                    age_med[age_med.Age==30].sample(n=50), age_med[age_med.Age==31].sample(n=50),
                    age_med[age_med.Age==32].sample(n=50), age_med[age_med.Age==33].sample(n=50),
                    age_med[age_med.Age==34].sample(n=50), age_med[age_med.Age==35].sample(n=50),
                    age_med[age_med.Age==36].sample(n=50), age_med[age_med.Age==37].sample(n=50),
                    age_med[age_med.Age==38].sample(n=50), age_med[age_med.Age==39].sample(n=50),
                    age_med[age_med.Age==40].sample(n=50), age_med[age_med.Age==41].sample(n=50),
                    age_med[age_med.Age==42].sample(n=50), age_med[age_med.Age==43].sample(n=50),
                    age_med[age_med.Age==44].sample(n=50), age_med[age_med.Age==45].sample(n=50),
                    age_med[age_med.Age==46].sample(n=50)])
age_meds
```

Out[230]:

	Athlete_ID	Age	medals_won	gold_won	silver_won	bronze_won
24624	111140	14	1	1	0	0
3085	14804	14	1	1	0	0
18124	80581	14	2	0	1	1
16245	72433	14	1	1	0	0
23615	106232	14	1	0	1	0
...	...	...	...	...	...	...
24674	111407	46	1	0	0	1
13529	59657	46	1	0	1	0
27345	123157	46	1	0	0	1
24069	108417	46	1	0	1	0
13023	57201	46	1	0	0	1

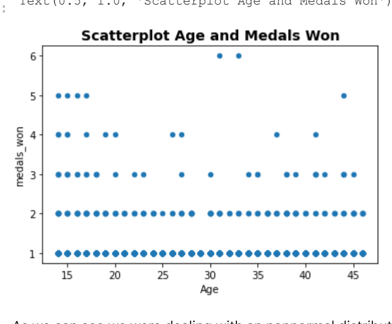
1650 rows × 6 columns

The sample size was 1650 athletes, that won at least one medal

n=1650

```python
#checking if there is a correlation between age and medals won
age_meds.plot.scatter(x='Age', y='medals_won')
plt.title('Scatterplot Age and Medals Won', size=14, fontweight='bold')
```

Out[231]: Text(0.5, 1.0, 'Scatterplot Age and Medals Won')



As we can see we were dealing with an nonnormal distribution, therefore I have used the Kendall or Sperman correlation

```python
#continous=> age and number of medals
#monotonic=> we cannot observe a monotonic development
corr_kdt=kendalltau(age_meds.Age, age_meds.medals_won)
corr_kdt
```

Out[232]: KendalltauResult(correlation=-0.05885045306368485, pvalue=0.0035146636686267507)

The correlation was weak and not statistically significant. To ensure we were not missing any trend or possibility that age and the medals won correlate, I have created a scatter plot of athletes that have not won a medal.

I have selected the same age range as I did and randomly chose 50 samples of every age.

```python
#getting the table of all non-medal winning athletes and their age
age_no_med=pysqldf('Select Athlete_ID, Age, no_medals_won from (select Athlete_ID, Age, count(case when Medal_I
```

```python
age_no_meds=pd.concat([age_no_med[age_no_med.Age==14].sample(n=50), age_no_med[age_no_med.Age==15].sample(n=50)
                       age_no_med[age_no_med.Age==16].sample(n=50), age_no_med[age_no_med.Age==17].sample(n=50
                       age_no_med[age_no_med.Age==18].sample(n=50), age_no_med[age_no_med.Age==19].sample(n=50
                       age_no_med[age_no_med.Age==20].sample(n=50), age_no_med[age_no_med.Age==21].sample(n=50
                       age_no_med[age_no_med.Age==22].sample(n=50), age_no_med[age_no_med.Age==23].sample(n=50
                       age_no_med[age_no_med.Age==24].sample(n=50), age_no_med[age_no_med.Age==25].sample(n=50
                       age_no_med[age_no_med.Age==26].sample(n=50), age_no_med[age_no_med.Age==27].sample(n=50
                       age_no_med[age_no_med.Age==28].sample(n=50), age_no_med[age_no_med.Age==29].sample(n=50
                       age_no_med[age_no_med.Age==30].sample(n=50), age_no_med[age_no_med.Age==31].sample(n=50
                       age_no_med[age_no_med.Age==32].sample(n=50), age_no_med[age_no_med.Age==33].sample(n=50
                       age_no_med[age_no_med.Age==34].sample(n=50), age_no_med[age_no_med.Age==35].sample(n=50
                       age_no_med[age_no_med.Age==36].sample(n=50), age_no_med[age_no_med.Age==37].sample(n=50
                       age_no_med[age_no_med.Age==38].sample(n=50), age_no_med[age_no_med.Age==39].sample(n=50
                       age_no_med[age_no_med.Age==40].sample(n=50), age_no_med[age_no_med.Age==41].sample(n=50
                       age_no_med[age_no_med.Age==42].sample(n=50), age_no_med[age_no_med.Age==43].sample(n=50
                       age_no_med[age_no_med.Age==44].sample(n=50), age_no_med[age_no_med.Age==45].sample(n=50
                       age_no_med[age_no_med.Age==46].sample(n=50)])
age_no_meds
```

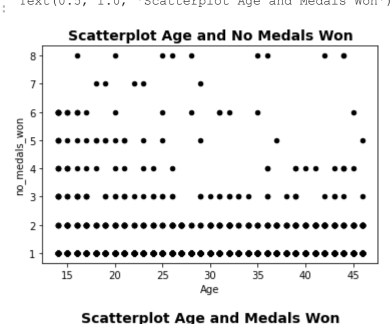
Out[234]:

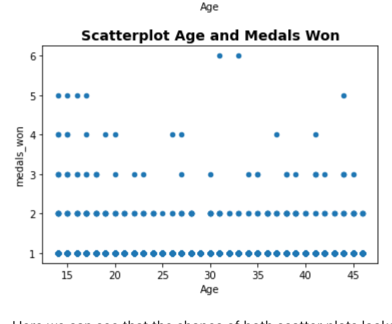
	Athlete_ID	Age	no_medals_won
31398	30793	14	1
133975	132164	14	1
23002	22661	14	1
23386	23049	14	6
39901	39223	14	1
...	...	...	...
6179	5955	46	2
19915	19484	46	2
125235	123157	46	1
27474	26951	46	2
21287	20905	46	1

1650 rows × 3 columns

```python
age_no_meds.plot.scatter(x='Age', y='no_medals_won', color='black')
plt.title('Scatterplot Age and No Medals Won', size=14, fontweight='bold')
age_meds.plot.scatter(x='Age', y='medals_won')
plt.title('Scatterplot Age and Medals Won', size=14, fontweight='bold')
```

Out[235]: Text(0.5, 1.0, 'Scatterplot Age and Medals Won')





Here we can see that the shapes of both scatter plots looked familiar. Therefore the hypothesis has to be rejected, because we did not have a solid and significant correlation.

Hypothesis - There is a negative correlation between Age and Medals

After analyzing age and medals won, I could not find a statistically significant correlation between these two variables. Therefore I had to reject the hypothesis.

Development of Height and Weight of Olympic Summer Athletes

For the following analysis, I just analyised the height and weight of the summer game athletes. To do so, I used the average height and weight of the athletes and I used the time period of 120 years.

It is important to keep in mind that I have used ffill() to get ride of the na-values, as I already mentioned these values should be evaluated if they are relistic or not.

```python
#impact of of the ffill function
ffill_=pysqldf('select Height, Weight, count(Height), count(Weight) from results where Year between 1920 and 19
```

```python
#distribution of height and weight
ffill_.head(10)
```

Out[237]:

	Height	Weight	count(Height)	count(Weight)
0	176	67	38816	38816
1	170	67	346	346
2	173	67	274	274
3	178	67	246	246
4	168	67	244	244
5	167	67	219	219
6	180	67	201	201
7	183	67	195	195
8	175	67	186	186
9	165	67	152	152

As we can see the combincation 176 cm and 67 Kg, which were used in the ffill function, created a huge amount of data and seemed very unrealstic, therefore they were not included in the analysis.

SQL Synatx */Impact of of the ffill function/*

Select

```
Height,
Weight,
count(Height),
count(Weight)
```

from results

where Year between 1920 and 1960

group by 1,2

order by count(Height) desc;

```python
#development of weight and height over 120 years of summer games
dev_ath=pysqldf('select results.Year, avg(results.Height) as Height, avg(results.Weight) as Weight from results
```

/Development of Weight and Height over 120 Years/

select

```
results.Year,
avg(results.Height) as Height,
avg(results.Weight) as Weight
```

from results

```
left join sports
    on results.Sport_ID=sports.Sport_ID
```

where Season_ID=1

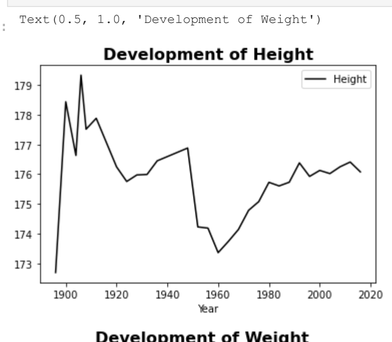
and Weight!=176

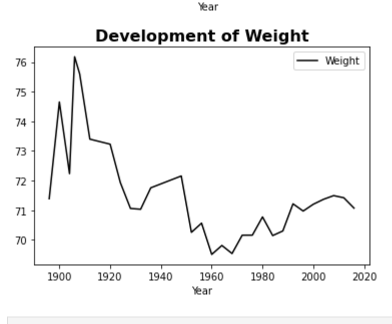
and Weight!=67

group by 1;

```python
dev_ath.plot(x='Year', y='Height', color='black')
plt.title('Development of Height', size=16, fontweight='bold')
dev_ath.plot(x='Year', y='Weight', color='black')
plt.title('Development of Weight', size=16, fontweight='bold')
```

Out[239]: Text(0.5, 1.0, 'Development of Weight')






```python
#development of weight and height of Male athletes over 120 Years
dev_ath_m=pysqldf('select results.Year, avg(results.Height) as Height, avg(results.Weight) as Weight from resul
```

/Development of Weight and Height of Male athletes over 120 Years/

select

```
results.Year,
avg(results.Height) as Height,
avg(results.Weight) as Weight
```

from results

where results.Sex_ID=1

and Season_ID=1

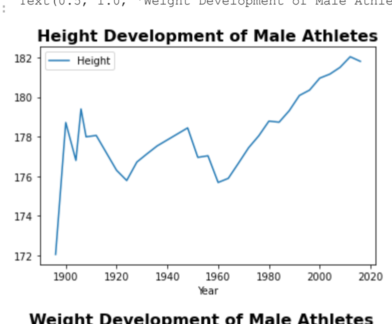
and Weight!=176

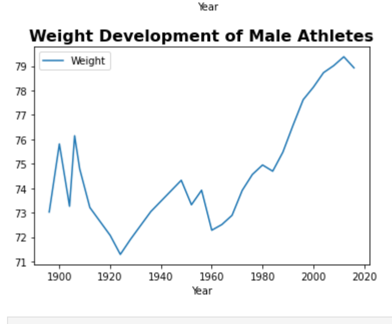
and Weight!=67

group by 1;

```python
dev_ath_m.plot(x='Year', y='Height')
plt.title('Height Development of Male Athletes', size=16, fontweight='bold')
dev_ath_m.plot(x='Year', y='Weight')
plt.title('Weight Development of Male Athletes', size=16, fontweight='bold')
```

Out[241]: Text(0.5, 1.0, 'Weight Development of Male Athletes')






```python
#development of weight and height of Female athletes over 120 Years
dev_ath_f=pysqldf('select results.Year, avg(results.Height) as Height, avg(results.Weight) as Weight from resul
```

SQL Syntax */Development of Weight and Height of Female athletes over 120 Years/*

select

```
results.Year,
avg(results.Height) as Height,
avg(results.Weight) as Weight
```

from results

where results.Sex_ID=2

and Season_ID=1

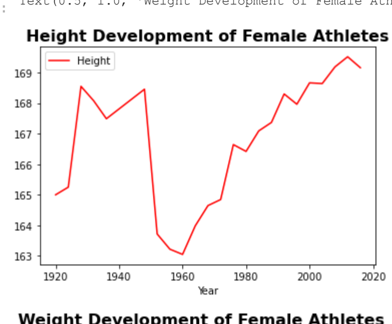
and Weight!=176

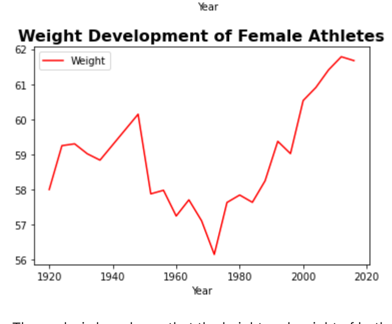
and Weight!=67

group by 1;

```python
dev_ath_f.plot(x='Year', y='Height', color='red')
plt.title('Height Development of Female Athletes', size=16, fontweight='bold')
dev_ath_f.plot(x='Year', y='Weight', color='red')
plt.title('Weight Development of Female Athletes', size=16, fontweight='bold')
```

Out[243]: Text(0.5, 1.0, 'Weight Development of Female Athletes')





The analysis has shown that the height and weight of both genders have increased.

Sources

The Correlation Coefficient (r) - Anon, 2021

URL: https://sphweb.bumc.bu.edu/otlt/MPH-Modules/PH717-QuantCore/PH717-Module9-Correlation-Regression/PH717-Module9-Correlation-Regression4.html.