Imports

In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

Read Data to Data Frame

In [7]:
```
df = pd.read_csv('tennis_stats.csv')
#print(df.head())
print(df.info())
print('\n\n')
print(df.describe)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1721 entries, 0 to 1720
Data columns (total 24 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   Player                      1721 non-null   object 
 1   Year                        1721 non-null   int64  
 2   FirstServe                  1721 non-null   float64
 3   FirstServePointsWon         1721 non-null   float64
 4   FirstServeReturnPointsWon   1721 non-null   float64
 5   SecondServePointsWon        1721 non-null   float64
 6   SecondServeReturnPointsWon  1721 non-null   float64
 7   Aces                        1721 non-null   int64  
 8   BreakPointsConverted        1721 non-null   float64
 9   BreakPointsFaced            1721 non-null   int64  
 10  BreakPointsOpportunities    1721 non-null   int64  
 11  BreakPointsSaved            1721 non-null   float64
 12  DoubleFaults                1721 non-null   int64  
 13  ReturnGamesPlayed           1721 non-null   int64  
 14  ReturnGamesWon              1721 non-null   float64
 15  ReturnPointsWon             1721 non-null   float64
 16  ServiceGamesPlayed          1721 non-null   int64  
 17  ServiceGamesWon             1721 non-null   float64
 18  TotalPointsWon              1721 non-null   float64
 19  TotalServicePointsWon       1721 non-null   float64
 20  Wins                        1721 non-null   int64  
 21  Losses                      1721 non-null   int64  
 22  Winnings                    1721 non-null   int64  
 23  Ranking                     1721 non-null   int64  
dtypes: float64(12), int64(11), object(1)
memory usage: 322.8+ KB
None



<bound method NDFrame.describe of                    Player  Year  FirstServe  FirstSe
rvePointsWon  \
0               Pedro Sousa  2016        0.88                0.50
1          Roman Safiullin  2017        0.84                0.62
```

```
2              Pedro Sousa  2017        0.83                0.60
3       Rogerio Dutra Silva  2010        0.83                0.64
4      Daniel Gimeno-Traver  2017        0.81                0.54
...                     ...   ...         ...                 ...
1716            Yann Marti  2010        0.41                0.66
1717         Mikhail Elgin  2012        0.41                0.60
1718  Alexander Kudryavtsev  2012        0.40                0.64
1719           Ivan Nedelko  2011        0.37                0.30
1720      Alexander Zverev  2013        0.36                0.50

      FirstServeReturnPointsWon  SecondServePointsWon  \
0                          0.38                  0.50
1                          0.26                  0.33
2                          0.28                  0.53
3                          0.34                  0.59
4                          0.00                  0.33
...                         ...                   ...
1716                       0.23                  0.45
1717                       0.30                  0.36
1718                       0.21                  0.42
1719                       0.15                  0.24
1720                       0.40                  0.24

      SecondServeReturnPointsWon  Aces  BreakPointsConverted  \
0                           0.39     0                  0.14
1                           0.07     7                  0.00
2                           0.44     2                  0.38
3                           0.33     2                  0.33
4                           0.33     1                  0.00
...                          ...   ...                   ...
1716                        0.50     1                  0.75
1717                        0.60     3                  0.60
1718                        0.42     4                  0.25
1719                        0.41     1                  0.00
1720                        0.36     1                  0.33

      BreakPointsFaced  ...  ReturnGamesWon  ReturnPointsWon  \
0                    7  ...            0.11             0.38
1                    7  ...            0.00             0.20
2                   10  ...            0.16             0.34
3                    5  ...            0.14             0.34
4                    2  ...            0.00             0.20
...                ...  ...             ...              ...
1716                 8  ...            0.23             0.36
1717                13  ...            0.33             0.43
1718                13  ...            0.07             0.29
1719                 5  ...            0.00             0.26
1720                 8  ...            0.33             0.39

      ServiceGamesPlayed  ServiceGamesWon  TotalPointsWon  \
0                      8             0.50            0.43
1                      9             0.67            0.41
2                     17             0.65            0.45
3                     15             0.80            0.49
4                      2             0.50            0.35
...                  ...              ...             ...
1716                  12             0.67            0.45
1717                  10             0.40            0.45
1718                  14             0.57            0.40
1719                   6             0.17            0.26
```

```
1720                   8            0.25            0.37

      TotalServicePointsWon  Wins  Losses  Winnings  Ranking
0                      0.50     1       2     39820      119
1                      0.57     0       1     17334      381
2                      0.59     4       1    109827      119
3                      0.63     0       0      9761      125
4                      0.50     0       1     32879      272
...                     ...   ...     ...       ...      ...
1716                   0.53     0       1      9117     1062
1717                   0.46    16      20     89755      831
1718                   0.51     0       1     44566      628
1719                   0.26     0       1     17527      264
1720                   0.33     0       1      8869        4

[1721 rows x 24 columns]>
```

Variable Descriptors (Provided by Code Academy)

Identifying Data

- Player: name of the tennis player
- Year: year data was recorded

Service Game Columns (Offensive)

- Aces: number of serves by the player where the receiver does not touch the ball
- DoubleFaults: number of times player missed both first and second serve attempts
- FirstServe: % of first-serve attempts made
- FirstServePointsWon: % of first-serve attempt points won by the player
- SecondServePointsWon: % of second-serve attempt points won by the player
- BreakPointsFaced: number of times where the receiver could have won service game of the player
- BreakPointsSaved: % of the time the player was able to stop the receiver from winning service game when they had the chance
- ServiceGamesPlayed: total number of games where the player served
- ServiceGamesWon: total number of games where the player served and won
- TotalServicePointsWon: % of points in games where the player served that they won

Return Game Columns (Defensive)

- FirstServeReturnPointsWon: % of opponents first-serve points the player was able to win
- SecondServeReturnPointsWon: % of opponents second-serve points the player was able to win
- BreakPointsOpportunities: number of times where the player could have won the service game of the opponent
- BreakPointsConverted: % of the time the player was able to win their opponent's service game when they had the chance
- ReturnGamesPlayed: total number of games where the player's opponent served

- ReturnGamesWon: total number of games where the player's opponent served and the player won
- ReturnPointsWon: total number of points where the player's opponent served and the player won
- TotalPointsWon: % of points won by the player

Outcomes

- Wins: number of matches won in a year
- Losses: number of matches lost in a year
- Winnings: total winnings in USD($) in a year
- Ranking: ranking at the end of year

Check for Data Cleanliness

In [ ]:
```
"""
Player                      1721 non-null   object --> !!!!Object is fine, nominal data
Year                        1721 non-null   int64  --> int64 is fine
FirstServe                  1721 non-null   float64--> !!!! float64 is fine, but need t
FirstServePointsWon         1721 non-null   float64--> float64 is fine
FirstServeReturnPointsWon   1721 non-null   float64--> float64 is fine
SecondServePointsWon        1721 non-null   float64--> float64 is fine
SecondServeReturnPointsWon  1721 non-null   float64--> float64 is fine
Aces                        1721 non-null   int64  --> int64 is fine
BreakPointsConverted        1721 non-null   float64--> float64 is fine
BreakPointsFaced            1721 non-null   int64  --> int64 is fine
BreakPointsOpportunities    1721 non-null   int64  --> int64 is fine
BreakPointsSaved            1721 non-null   float64--> float64 is fine
DoubleFaults                1721 non-null   int64  --> int64 is fine
ReturnGamesPlayed           1721 non-null   int64  --> int64 is fine
ReturnGamesWon              1721 non-null   float64--> float64 is fine
ReturnPointsWon             1721 non-null   float64--> float64 is fine
ServiceGamesPlayed          1721 non-null   int64  --> int64 is fine
ServiceGamesWon             1721 non-null   float64--> float64 is fine
TotalPointsWon              1721 non-null   float64--> float64 is fine
TotalServicePointsWon       1721 non-null   float64--> float64 is fine
Wins                        1721 non-null   int64  --> int64 is fine
Losses                      1721 non-null   int64  --> int64 is fine
Winnings                    1721 non-null   int64  --> int64 is fine
Ranking                     1721 non-null   int64  --> int64 is fine

"""
```

Data Cleaning / Inspection

In [10]:
```
print(df.Player.unique())
#No apparent blanks / unknowns when checking this data

print('\n\n')
print(df.FirstServe.unique())
#Not familiar enough with Tennis to confirm what first serve signifies, but float64 app
```

```
['Pedro Sousa' 'Roman Safiullin' 'Rogerio Dutra Silva'
 'Daniel Gimeno-Traver' 'Andres Artunedo Martinavarro' 'Eduardo Struvay'
 'Riccardo Bellotti' 'Austin Krajicek' 'Guilherme Clezar' 'Blaz Kavcic'
 'Ze Zhang' 'John Millman' 'Jurgen Melzer' 'Kristijan Mesaros'
 'Fabiano De Paula' 'Salvatore Caruso' 'Philipp Davydenko'
 'Kenny De Schepper' 'Simone Bolelli' 'Alexander Ward' 'Darian King'
 'Carlos Berlocq' 'Marco Cecchinato' 'Gonzalo Escobar' 'Omar Jasika'
 'Rubin Statham' 'Roberto Bautista Agut' 'Flavio Cipolla' 'Louis Wessels'
 'Diego Schwartzman' 'Gerard Granollers' 'Frederik Nielsen'
 'Riccardo Ghedin' 'Aldin Setkic' 'Matteo Viola' 'Joao Domingues'
 'Gregoire Barrere' 'Mate Pavic' 'Laurent Lokoli' 'Kevin Anderson'
 'John Isner' 'Christopher Eubanks' 'Maximo Gonzalez' 'Gerald Melzer'
 'Santiago Giraldo' 'Daniel Brands' 'Walter Trusendi' 'Axel Michon'
 'Fernando Verdasco' 'Andrea Arnaboldi' 'Konstantin Kravchuk'
 'Bradley Klahn' 'Dusan Lajovic' 'Dimitar Kuzmanov'
 'Frederico Ferreira Silva' 'Carlos Gomez-Herrera' 'Blaz Rola'
 'Aslan Karatsev' 'Matthias Bachinger' 'Jerzy Janowicz' 'Rudolf Molleker'
 'Andrej Martin' 'Lucas Gomez' 'Mischa Zverev' 'Denis Istomin'
 'Maximilian Marterer' 'Yannick Mertens' 'Marsel Ilhan' 'Markus Eriksson'
 'Guido Pella' 'Marcos Giron' 'Mikael Ymer' 'Thomas Fabbiano'
 'Ruben Bemelmans' 'Bernard Tomic' 'Alexandr Dolgopolov' 'Ivo Karlovic'
 'Yuki Bhambri' 'Alessandro Giannessi' 'Gilles Muller' 'Thiago Monteiro'
 'Tigre Hank' 'Filip Krajinovic' 'Cedrik-Marcel Stebe' 'Radu Albot'
 'Liam Broady' 'Mikhail Kukushkin' 'Nikoloz Basilashvili' 'Oliver Golding'
 'Sekou Bangoura' 'Frank Dancevic' 'Alexandar Lazov' 'Mohamed Safwat'
 'Alessandro Bega' 'Fred Gil' 'Kimmer Coppejans' 'Niels Desein'
 'Jan Choinski' 'John-Patrick Smith' 'Zhe Li' 'Christian Lindell'
 'Jordi Samper-Montana' 'Hyeon Chung' 'Franko Skugor' 'Miljan Zekic'
 'Damir Dzumhur' 'Laslo Djere' 'Dmitry Tursunov' 'Marco Chiudinelli'
 'Quentin Halys' 'Lorenzo Giustino' 'Edan Leshem' 'Philipp Kohlschreiber'
 'Julien Benneteau' 'Mate Delic' 'Paolo Lorenzi' 'Nicolas Jarry'
 'Maxime Hamou' 'Matthew Ebden' 'Juan Martin del Potro' 'Jimmy Wang'
 'Daniel Evans' 'Philip Bester' 'Pablo Andujar' 'David Vega Hernandez'
 'Aleksandr Nedovyesov' 'Juan Ignacio Londero' 'Victor Estrella Burgos'
 'Casper Ruud' 'Alex Bolt' 'Takuto Niki' 'Andre Ghem' 'Gael Monfils'
 'Mathias Bourgue' 'Di Wu' 'James Ward' 'Jiri Vesely' 'Kevin King'
 'Facundo Arguello' 'Tommy Robredo' 'Horacio Zeballos' 'Javier Marti'
 'Nicola Kuhn' 'Frances Tiafoe' 'Dennis Novak' 'Taro Daniel'
 'Dustin Brown' 'Tallon Griekspoor' 'Lukas Lacko' 'Luke Saville'
 'Joao Souza' 'Mackenzie McDonald' 'Martin Klizan' 'Nicolas Mahut'
 'Facundo Bagnis' 'Mikhail Elgin' 'Ricardo Hocevar' 'Tommy Paul'
 'Fabio Fognini' 'Kyle Edmund' 'Nick Kyrgios' 'Richard Gasquet'
 'Rhyne Williams' 'Cem Ilkel' 'Emilio Gomez' 'Adrian Ungur'
 'Kei Nishikori' 'Joao Sousa' 'Tommy Haas' 'Sam Groth' 'Lukas Rosol'
 'Florian Mayer' 'Jan Mertl' 'Pedja Krstin' 'Yasutaka Uchiyama'
 'Tristan Lamasine' 'Milos Raonic' 'Marcel Granollers'
 'Albert Ramos-Vinolas' 'Grega Zemlja' 'Henri Laaksonen' 'Jurgen Zopp'
 'Michael Mmoh' 'Steve Darcis' 'Andrey Golubev' 'Jason Kubler'
 'Sergiy Stakhovsky' 'Vincent Millot' 'Yen-Hsun Lu' 'Christian Garin'
 'Cristobal Saavedra-Corvalan' 'Nils Langer' 'Bastian Trinker' 'Evan King'
 'Marco Trungelliti' 'Yannick Hanfmann' 'Yuichi Sugita'
 'Yoshihito Nishioka' 'Federico Gaio' 'Leonardo Mayer' 'Brydan Klein'
 'Daniel Masur' 'Pablo Cuevas' 'Igor Sijsling' 'Peter Gojowczyk'
 'Ernests Gulbis' 'Robin Kern' 'Attila Balazs' 'Denis Kudla' 'Rajeev Ram'
 'Roberto Carballes Baena' 'Nicolas Kicker' 'Matteo Trevisan' 'Dudi Sela'
 'Jeremy Chardy' 'Victor Baluda' 'Mikhail Youzhny' 'Benjamin Mitchell'
 'Andrey Rublev' 'Blake Mott' 'Gonzalo Lama' 'Tim Smyczek' 'Mirza Basic'
 'Go Soeda' 'Donald Young' 'Yannick Maden' 'Guido Andreozzi'
 'Hiroki Moriya' 'Borna Coric' 'Norbert Gombos' 'Yann Marti' 'Jason Jung'
 'Tatsuma Ito' 'JC Aragone' 'Thai-Son Kwiatkowski' 'Marinko Matosevic'
```

```
 'Adrian Mannarino' 'Isak Arvidsson' 'Matthew Barton' 'Carlos Salamanca'
 'Anil Yuksel' 'Marcus Willis' 'Robin Haase' 'Peter Polansky'
 'Ramkumar Ramanathan' 'Brayden Schnur' 'Tobias Kamke' 'Feliciano Lopez'
 'Ivan Dodig' 'Lorenzo Sonego' "Christopher O'Connell"
 'Pablo Carreno Busta' 'David Goffin' 'Grigor Dimitrov'
 'Federico Delbonis' 'Lucas Miedler' 'Matteo Donati' 'Steven Diez'
 'Daniel Munoz de la Nava' 'Karen Khachanov' 'Ryan Harrison' 'Gianni Mina'
 'David Guez' 'Arthur De Greef' 'Jaume Munar' 'Viktor Troicki'
 'Ilya Ivashka' 'Viktor Galovic' 'Alexios Halebian' 'Stan Wawrinka'
 'Andrey Kuznetsov' 'Teymuraz Gabashvili' 'Adrian Menendez-Maceiras'
 'Andreas Seppi' 'Jozef Kovalik' 'Alejandro Gonzalez' 'Tim Puetz'
 'Thiemo de Bakker' 'Ante Pavic' 'Joris De Loore' 'Stephane Robert'
 'Guillermo Garcia-Lopez' 'Gleb Sakharov' 'Marius Copil' 'David Ferrer'
 'Sam Querrey' 'James Duckworth' 'Jordan Thompson' 'Andreas Haider-Maurer'
 'Benjamin Becker' 'Thomaz Bellucci' 'Noah Rubin' 'Illya Marchenko'
 'Daniil Medvedev' 'Vaclav Safranek' 'Steve Johnson' 'Alexander Zverev'
 'Alex de Minaur' 'Alexander Sarkissian' 'Takao Suzuki' 'Tennys Sandgren'
 'Albano Olivetti' 'Yan Bai' 'Thanasi Kokkinakis' 'Renzo Olivo'
 'Jose Hernandez-Fernandez' 'Petros Chrysochos' 'Lucas Pouille'
 'Vishnu Vardhan' 'Stefano Napolitano' 'James McGee' 'Constant Lestienne'
 'Dominic Thiem' 'Adam Pavlasek' 'Petru-Alexandru Luncanu'
 'Kevin Krawietz' 'Calvin Hemery' 'Evgeny Donskoy' 'Pierre-Hugues Herbert'
 'Pere Riba' 'Michal Przysiezny' 'Andres Molteni' 'Alexander Kudryavtsev'
 'Tomas Berdych' 'Jack Sock' 'Lamine Ouahab' 'Dominik Koepfer'
 'Malek Jaziri' 'Mitchell Krueger' 'Edward Corrie' 'Reda El Amrani'
 'Denis Shapovalov' 'Daniel Altmaier' 'Vasek Pospisil' 'Aljaz Bedene'
 'Yaraslav Shyla' 'Maxime Authom' 'Jan-Lennard Struff' 'Elias Ymer'
 'Stefan Kozlov' 'Patrick Kypson' 'Finn Tearney' 'Nicolas Almagro'
 'Gianluigi Quinzi' 'Janko Tipsarevic' 'Luca Vanni' 'Tsung-Hua Yang'
 'Gilles Simon' 'Jesse Witten' 'Marton Fucsovics' 'Ricardas Berankis'
 'Andrew Whittington' 'Gastao Elias' 'Stefanos Tsitsipas' 'Nikola Cacic'
 'Josko Topic' 'Collin Altamirano' 'Jan Satral' 'Alexandre Muller'
 'Bjorn Fratangelo' 'Alexander Bublik' 'Cameron Norrie' 'Reilly Opelka'
 'Daniel Elahi Galan' 'Adrien Bossel' 'Christian Harrison'
 'Gianluca Mager' 'Michael Linzer' 'Ruben Ramirez Hidalgo' 'Chuhan Wang'
 'Paul-Henri Mathieu' 'Lloyd Harris' 'Dennis Novikov' 'Pedro Cachin'
 'Guillermo Olaso' 'Taylor Fritz' 'Zhizhen Zhang' 'Marvin Moeller'
 'Saketh Myneni' 'Caio Zampieri' 'Agustin Velotti' 'Stefano Travaglia'
 'Ivan Nedelko' 'Nicolaas Scholtz' 'Filip Peliwo' 'Marin Cilic'
 'Amine Ahouda' 'Sebastian Ofner' 'Inigo Cervantes' 'Ernesto Escobedo'
 'Laurynas Grigelis' 'Nino Serdarusic' 'Benjamin Bonzi'
 'Tim Van Rijthoven' 'Peter Torebko' 'Luis Patino' 'Eric Quigley'
 'Romain Jouan' 'Johan Nikles' 'Mubarak Shannan Zayid' 'Jonathan Eysseric'
 'Nikola Milojevic' 'Jared Donaldson' 'Marcos Baghdatis'
 'Fernando Romboli' 'Jaroslav Pospisil' 'Egor Gerasimov'
 'Geoffrey Blancaneaux' 'Yibing Wu' 'Alexandar Lazarov' 'Akira Santillan'
 'Antoine Bellier' 'Wishaya Trongcharoenchaikul' 'Marko Tepavac'
 'Daniel Dutra da Silva' 'Benoit Paire' 'Kamil Majchrzak'
 'Sergio Gutierrez-Ferrol' 'Roberto Ortega-Olmedo' 'Sherif Sabry'
 'Enrique Lopez Perez' 'Marc Polmans' 'N Vijay Sundar Prashanth'
 'Orlando Luz' 'Evgeny Tyurnev' 'Sergey Betov' 'Fred Simonsson'
 'Matteo Berrettini' 'Ryan Shane' 'Roberto Marcora' 'Yusuke Takahashi'
 'Manuel Sanchez' 'Xin Gao' 'Alejandro Gomez' 'Nathan Pasha'
 'Boy Westerhof' 'Germain Gigounon' 'Kento Takeuchi']
```

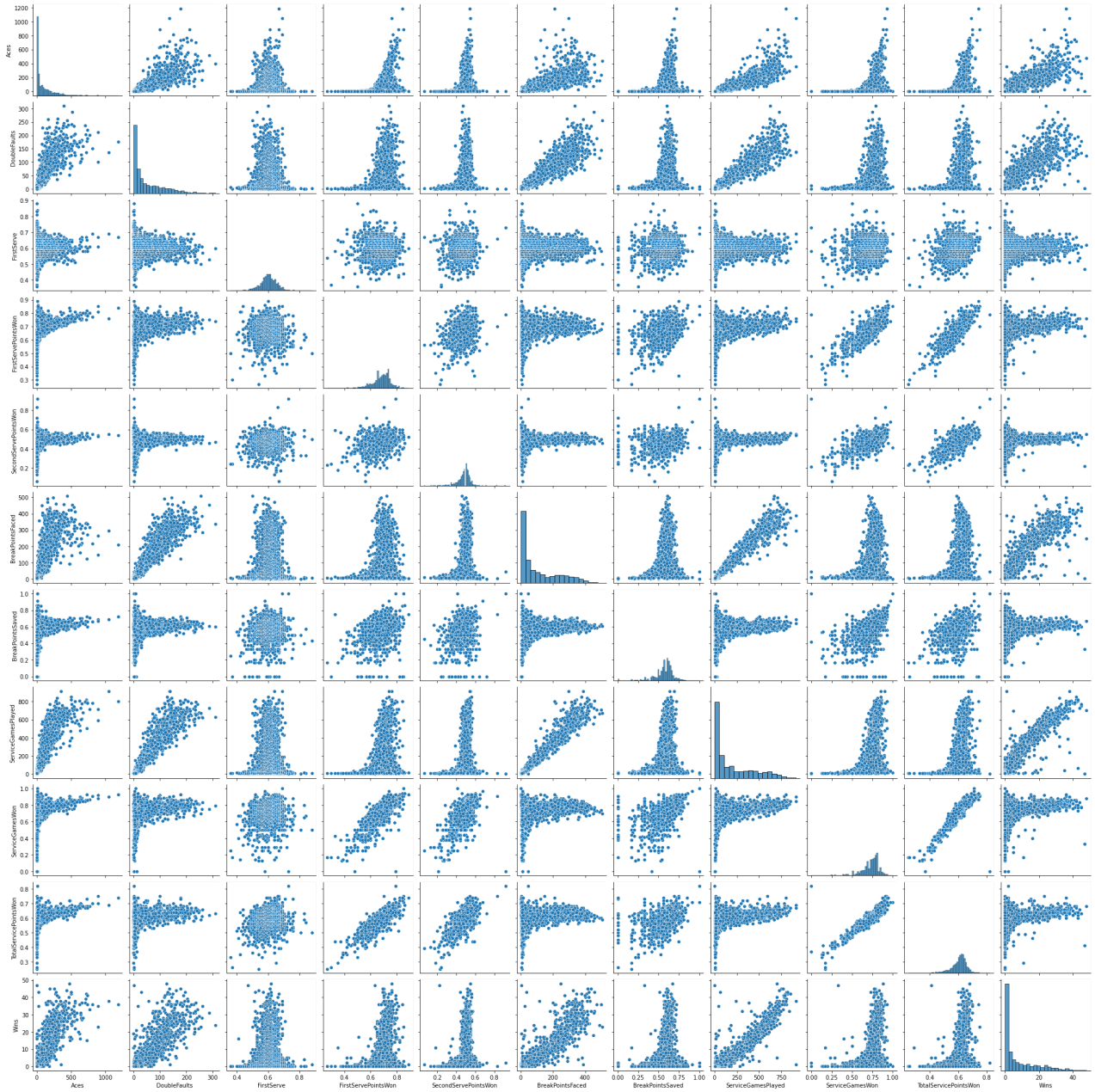
```
[0.88 0.84 0.83 0.81 0.77 0.76 0.75 0.74 0.73 0.72 0.71 0.7  0.69 0.68
 0.67 0.66 0.65 0.64 0.63 0.62 0.61 0.6  0.59 0.58 0.57 0.56 0.55 0.54
```

```
0.53 0.52 0.51 0.5  0.49 0.48 0.47 0.46 0.45 0.44 0.43 0.42 0.41 0.4
0.37 0.36]
```

Perform EDA (Exploratory Data Analysis)

In [12]:

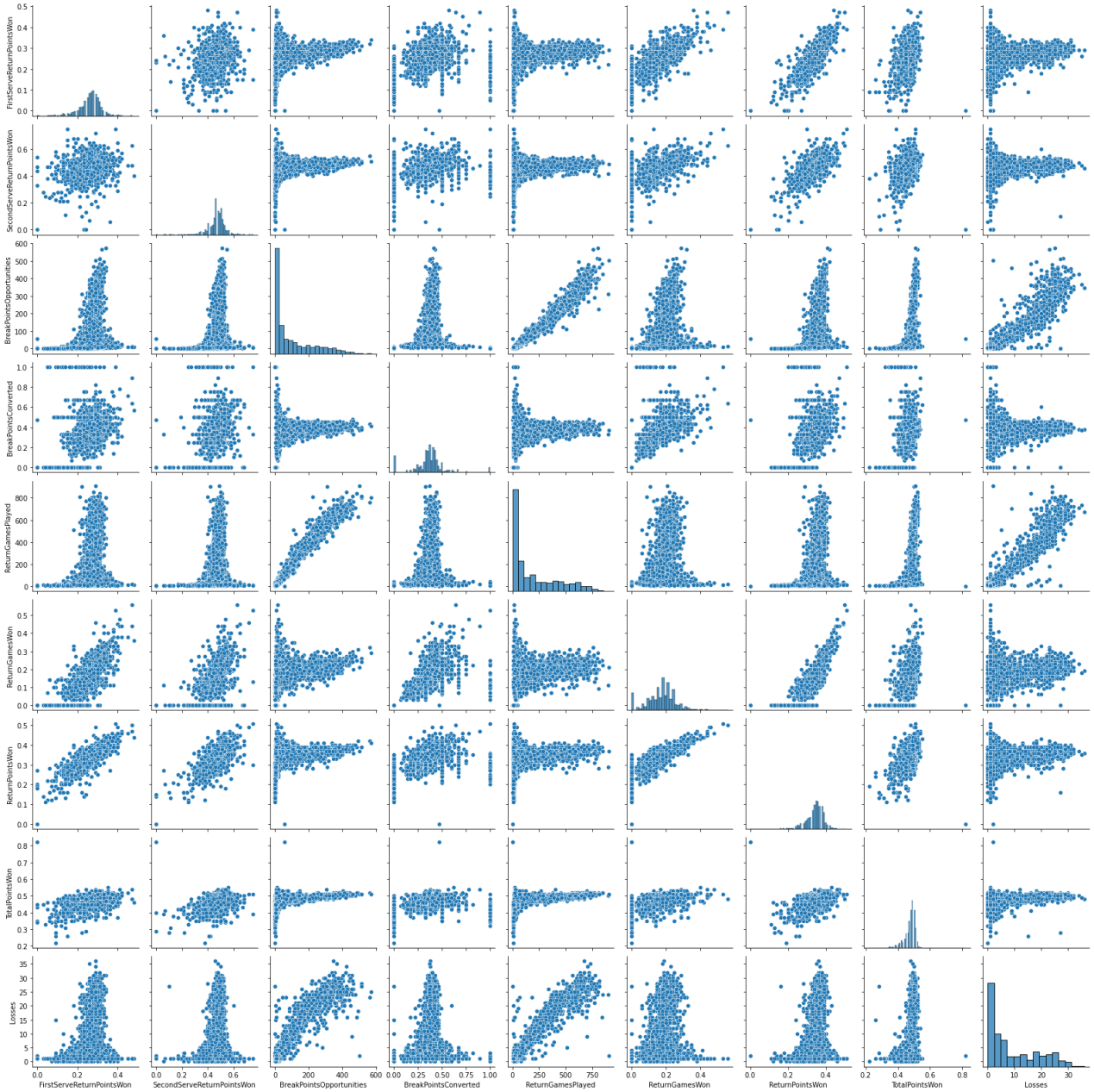
```
# So excited to try out pairplot on a full project on here
# https://seaborn.pydata.org/generated/seaborn.pairplot.html

#To keep the system burden down per chart, first we will break the df into 'offensive v

df_off_win = df[['Aces', 'DoubleFaults', 'FirstServe', 'FirstServePointsWon',\
                 'SecondServePointsWon', 'BreakPointsFaced', 'BreakPointsSaved',\
                'ServiceGamesPlayed', 'ServiceGamesWon', 'TotalServicePointsWon','Wins

sns.pairplot(data=df_off_win, diag_kind="hist")
plt.show()
plt.clf()

"""Postive Correlations Found v Wins:
- Aces
- DoubleFaults
- BreakPointsFaced
- ServiceGamesPlayed
"""
```



<Figure size 432x288 with 0 Axes>

In [14]:

```
df_def_loss = df[['FirstServeReturnPointsWon', 'SecondServeReturnPointsWon', 'BreakPoin
                  'BreakPointsConverted', 'ReturnGamesPlayed', 'ReturnGamesWon', 'Retur

sns.pairplot(data=df_def_loss, diag_kind="hist")
plt.show()
plt.clf()

"""Postive Correlations Found v Losses:
- BreakPointsOpportunities
- ReturnGamesPlayed
"""
```

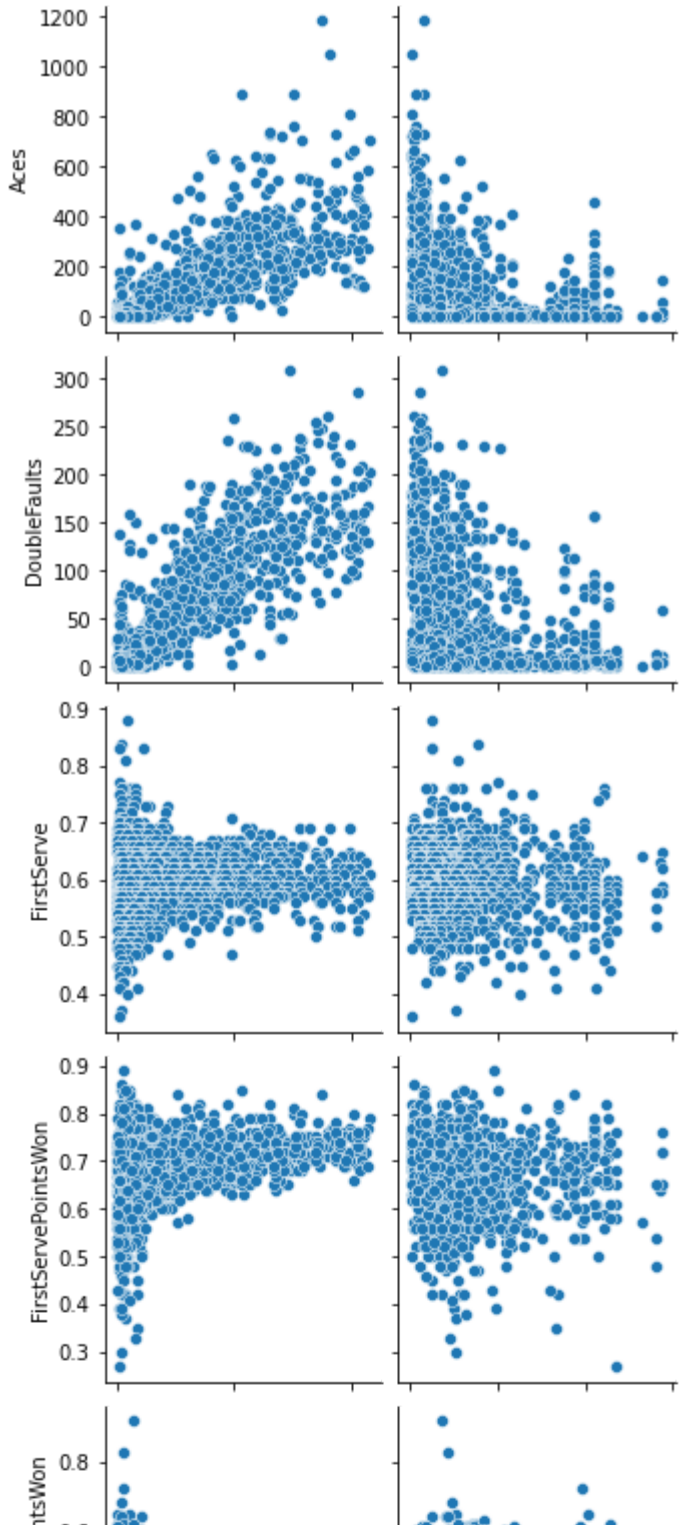


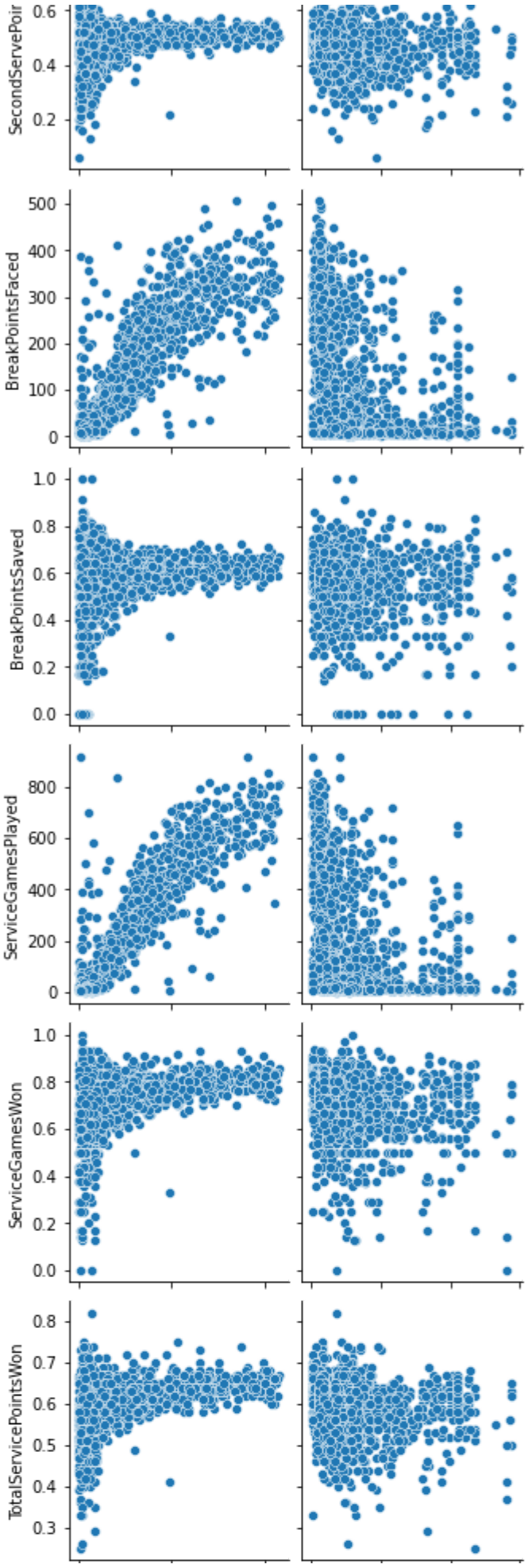
Out[14]: 'Postive Correlations Found v Losses:\n- \n\n'

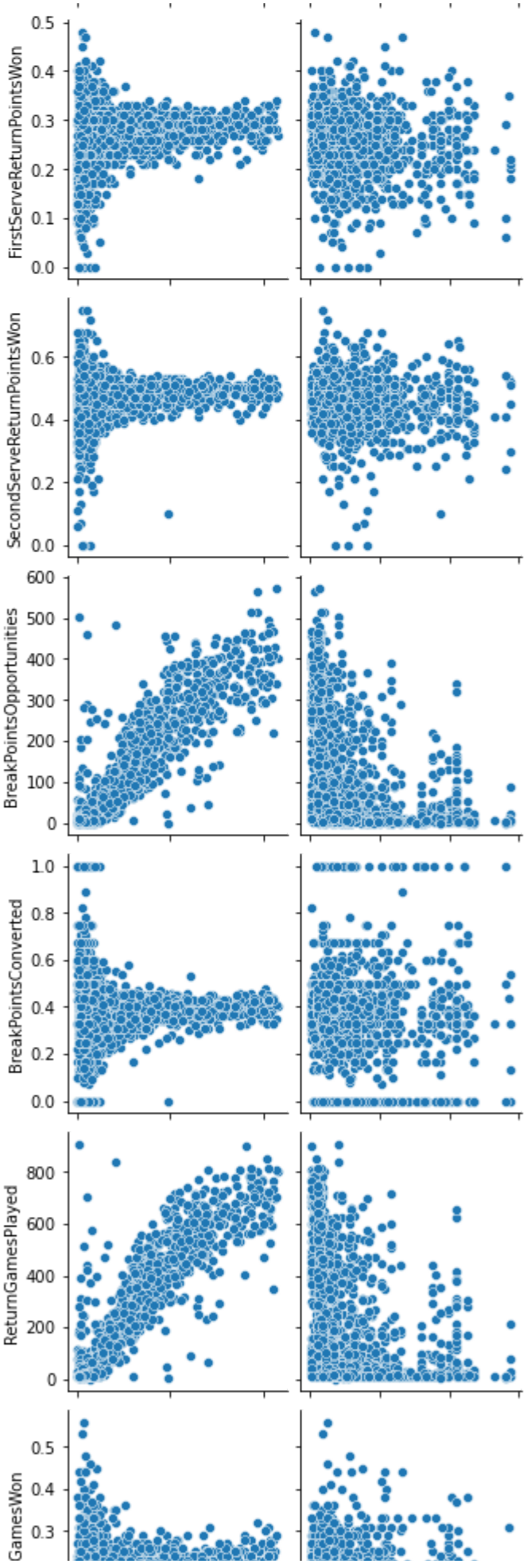
<Figure size 432x288 with 0 Axes>

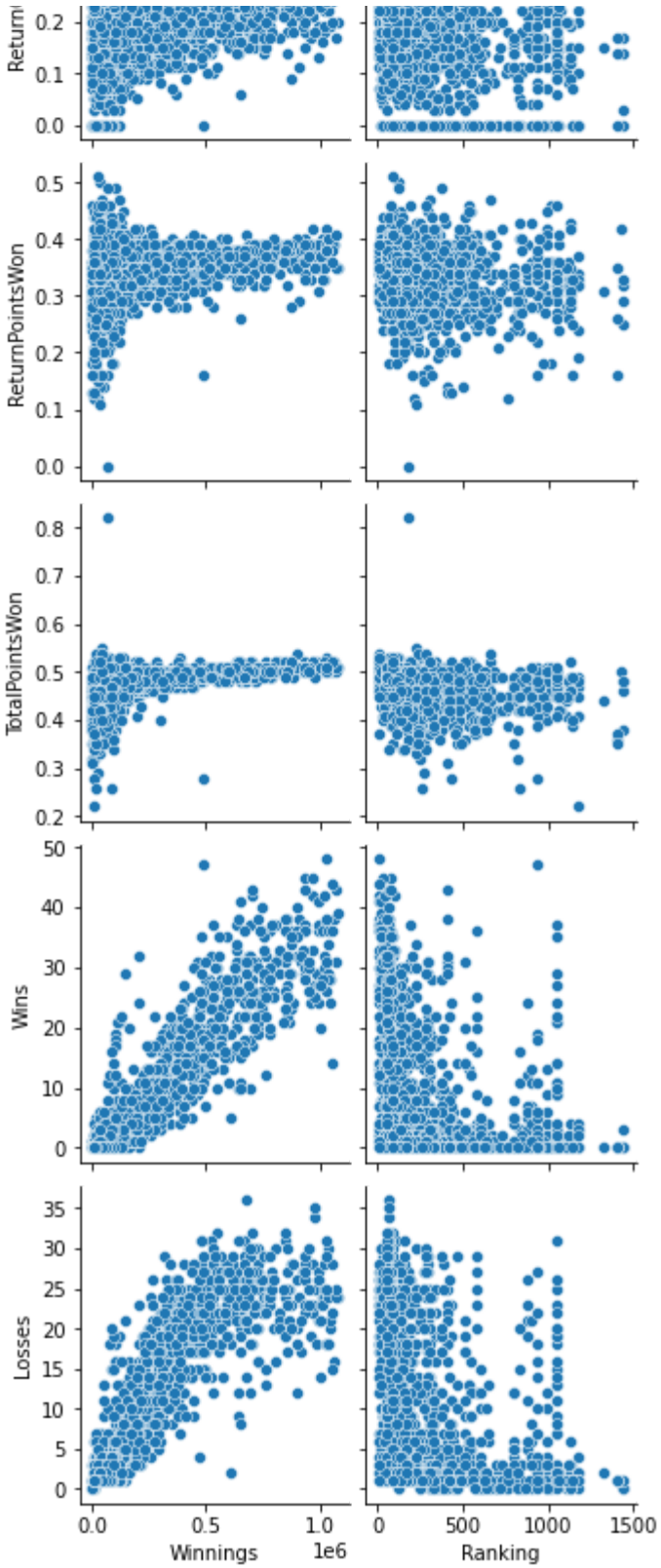
In [15]:
```
#Now we will check offense and defense attributes against Winnings and Ranking

sns.pairplot(data=df, x_vars=['Winnings', 'Ranking'], y_vars=['Aces', 'DoubleFaults', '
                 'SecondServePointsWon', 'BreakPointsFaced', 'BreakPointsSaved',\
                'ServiceGamesPlayed', 'ServiceGamesWon', 'TotalServicePointsWon',\
               'FirstServeReturnPointsWon', 'SecondServeReturnPointsWon', 'BreakPoints
                 'BreakPointsConverted', 'ReturnGamesPlayed', 'ReturnGamesWon', 'Retur
                   'TotalPointsWon', 'Wins','Losses'])
plt.show()
plt.clf()

"""
Postive Correlations Found v Winnings:
- Wins and Losses ??? Not sure I'm reading this right
- (Minor) Aces
- DoubleFaults
```

```
- BreakPointsFaced
- ServiceGamesPlayed
- BreakPointsOpportunities
- ReturnGamesPlayed

Minor Negative Correlations Found v Ranking:
- Aces
- DoubleFaults
- BreakPointsFaced
- ServiceGamesPlayed
- Wins and Losses, but with heavy outliers
"""
```




SecondServePoir
0.6
0.4
0.2
BreakPointsFaced
500
400
300
200
100
0
BreakPointsSaved
1.0
0.8
0.6
0.4
0.2
0.0
ServiceGamesPlayed
800
600
400
200
0
ServiceGamesWon
1.0
0.8
0.6
0.4
0.2
0.0
TotalServicePointsWon
0.8
0.7
0.6
0.5
0.4
0.3


FirstServeReturnPointsWon
SecondServeReturnPointsWon
BreakPointsOpportunities
BreakPointsConverted
ReturnGamesPlayed
GamesWon



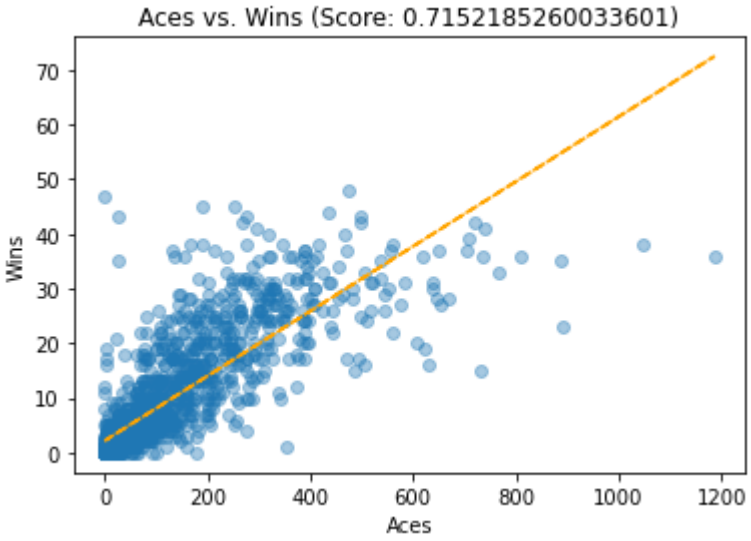
Out[15]:
```
'Postive Correlations Found v Wins:\n- Aces\n- DoubleFaults\n- BreakPointsFaced\n- ServiceGamesPlayed\n'
<Figure size 432x288 with 0 Axes>
```

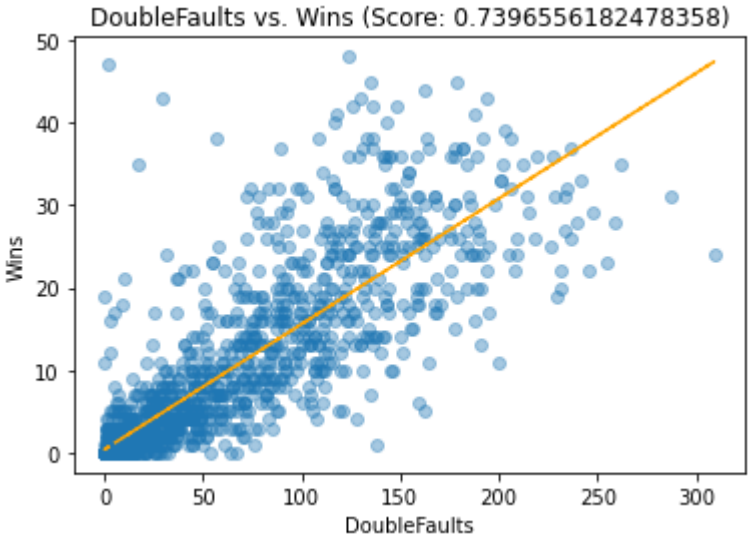
Perform Single Feature Linear Regressions

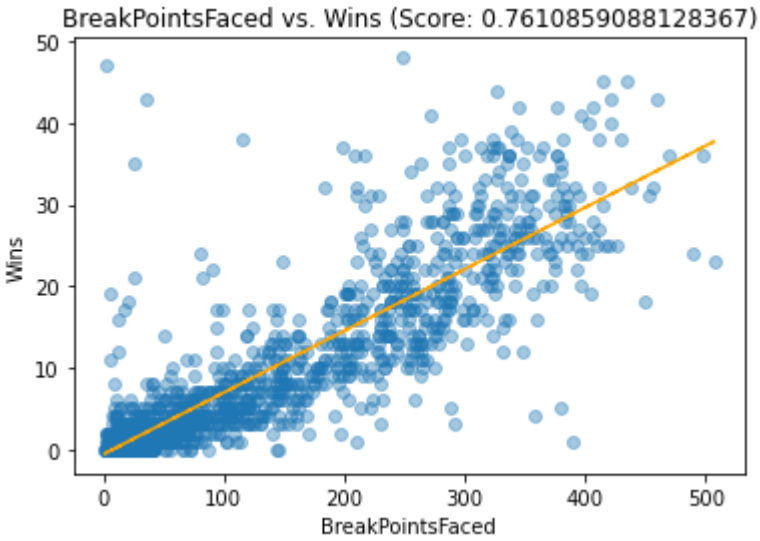
In [30]:
```
#Win Linear Regressions First

#Aces vs Wins
```

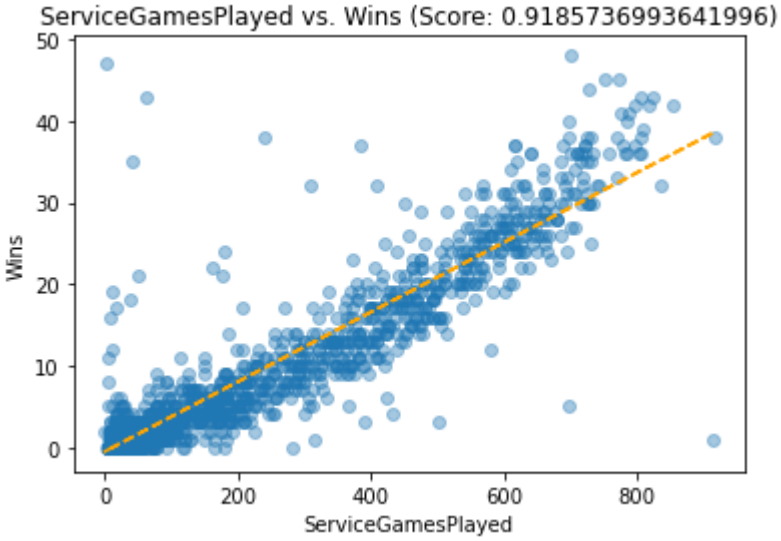
```
features = df[['Aces']]
outcomes = df[['Wins']]

x_train, x_test, y_train, y_test = train_test_split(features, outcomes, train_size=.8,

model = LinearRegression()
model.fit(x_train, y_train)
score = model.score(x_test, y_test)

predictions = model.predict(features)

plt.scatter(features, outcomes, alpha=.4)
plt.plot(features, predictions, ls='--', color='orange')
plt.xlabel('Aces')
plt.ylabel('Wins')
plt.title('Aces vs. Wins (Score: ' + str(score) + ')')
plt.show()
plt.clf()


#DoubleFaults vs Wins
features = df[['DoubleFaults']]
outcomes = df[['Wins']]

x_train, x_test, y_train, y_test = train_test_split(features, outcomes, train_size=.8,

model = LinearRegression()
model.fit(x_train, y_train)
score = model.score(x_test, y_test)

predictions = model.predict(features)

plt.scatter(features, outcomes, alpha=.4)
plt.plot(features, predictions, ls='--', color='orange')
plt.xlabel('DoubleFaults')
plt.ylabel('Wins')
plt.title('DoubleFaults vs. Wins (Score: ' + str(score) + ')')
plt.show()
plt.clf()

#BreakPointsFaced vs Wins
features = df[['BreakPointsFaced']]
outcomes = df[['Wins']]

x_train, x_test, y_train, y_test = train_test_split(features, outcomes, train_size=.8,

model = LinearRegression()
model.fit(x_train, y_train)
score = model.score(x_test, y_test)

predictions = model.predict(features)

plt.scatter(features, outcomes, alpha=.4)
plt.plot(features, predictions, ls='--', color='orange')
plt.xlabel('BreakPointsFaced')
plt.ylabel('Wins')
plt.title('BreakPointsFaced vs. Wins (Score: ' + str(score) + ')')
plt.show()
plt.clf()
```

```
#ServiceGamesPlayed vs Wins
features = df[['ServiceGamesPlayed']]
outcomes = df[['Wins']]

x_train, x_test, y_train, y_test = train_test_split(features, outcomes, train_size=.8,

model = LinearRegression()
model.fit(x_train, y_train)
score = model.score(x_test, y_test)

predictions = model.predict(features)

plt.scatter(features, outcomes, alpha=.4)
plt.plot(features, predictions, ls='--', color='orange')
plt.xlabel('ServiceGamesPlayed')
plt.ylabel('Wins')
plt.title('ServiceGamesPlayed vs. Wins (Score: ' + str(score) + ')')
plt.show()
plt.clf()
```







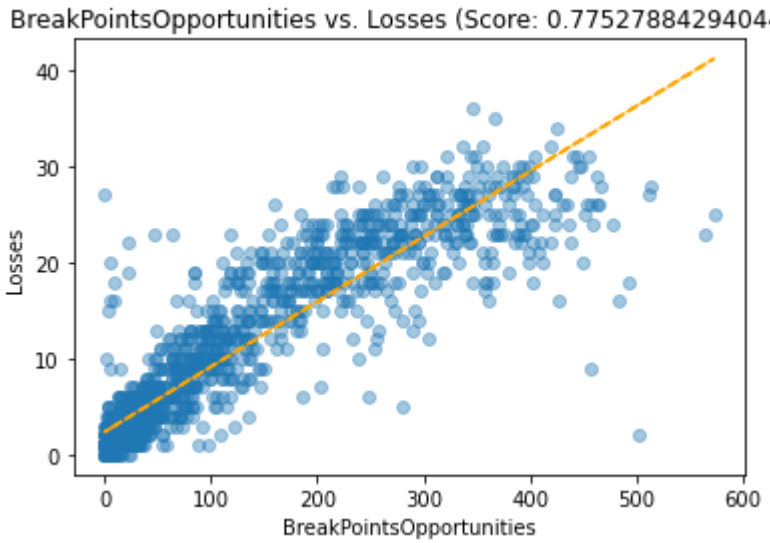


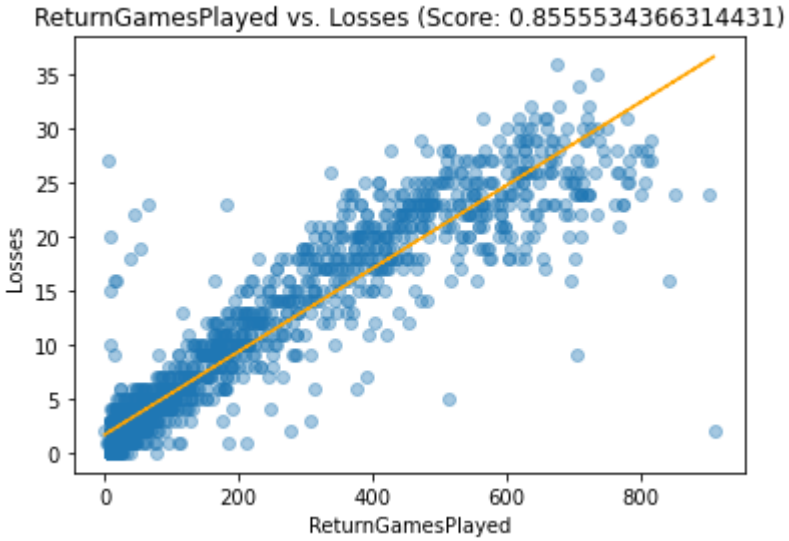
<Figure size 432x288 with 0 Axes>

In [32]:

```
#Loss Linear Regressions Second

#BreakPointsOpportunities vs Losses
features = df[['BreakPointsOpportunities']]
outcomes = df[['Losses']]

x_train, x_test, y_train, y_test = train_test_split(features, outcomes, train_size=.8,

model = LinearRegression()
model.fit(x_train, y_train)
score = model.score(x_test, y_test)

predictions = model.predict(features)

plt.scatter(features, outcomes, alpha=.4)
plt.plot(features, predictions, ls='--', color='orange')
plt.xlabel('BreakPointsOpportunities')
plt.ylabel('Losses')
plt.title('BreakPointsOpportunities vs. Losses (Score: ' + str(score) + ')')
plt.show()
plt.clf()

#ReturnGamesPlayed vs Losses
```

```
features = df[['ReturnGamesPlayed']]
outcomes = df[['Losses']]

x_train, x_test, y_train, y_test = train_test_split(features, outcomes, train_size=.8,

model = LinearRegression()
model.fit(x_train, y_train)
score = model.score(x_test, y_test)

predictions = model.predict(features)

plt.scatter(features, outcomes, alpha=.4)
plt.plot(features, predictions, ls='--', color='orange')
plt.xlabel('ReturnGamesPlayed')
plt.ylabel('Losses')
plt.title('ReturnGamesPlayed vs. Losses (Score: ' + str(score) + ')')
plt.show()
plt.clf()
```






```
<Figure size 432x288 with 0 Axes>
```

Perform Two Feature Linear Regressions

In [41]:

```
#DoubleFaults & BreakPointsFaced v Winnings
features = df[['DoubleFaults', 'BreakPointsFaced']]
outcomes = df[['Winnings']]

x_train, x_test, y_train, y_test = train_test_split(features, outcomes, train_size=.8,

model = LinearRegression()
model.fit(x_train, y_train)
score = model.score(x_test, y_test)
print('Scores')
print('DoubleFaults & BreakPointsFaced: {}'.format(score))

#BreakPointsFaced & ServiceGamesPlayed v Winnings
features = df[['BreakPointsFaced', 'ServiceGamesPlayed']]
outcomes = df[['Winnings']]

x_train, x_test, y_train, y_test = train_test_split(features, outcomes, train_size=.8,

model = LinearRegression()
model.fit(x_train, y_train)
score = model.score(x_test, y_test)
print('BreakPointsFaced & ServiceGamesPlayed: {}'.format(score))

#ServiceGamesPlayed & BreakPointsOpportunities v Winnings
features = df[['ServiceGamesPlayed', 'BreakPointsOpportunities']]
outcomes = df[['Winnings']]

x_train, x_test, y_train, y_test = train_test_split(features, outcomes, train_size=.8,

model = LinearRegression()
model.fit(x_train, y_train)
score = model.score(x_test, y_test)
print('ServiceGamesPlayed & BreakPointsOpportunities: {}'.format(score))

#BreakPointsOpportunities & ReturnGamesPlayed v Winnings
features = df[['BreakPointsOpportunities', 'ReturnGamesPlayed']]
outcomes = df[['Winnings']]

x_train, x_test, y_train, y_test = train_test_split(features, outcomes, train_size=.8,

model = LinearRegression()
model.fit(x_train, y_train)
score = model.score(x_test, y_test)
print('BreakPointsOpportunities & ReturnGamesPlayed: {}'.format(score))

"""
(ServiceGamesPlayed & BreakPointsOpportunities)
and
(BreakPointsOpportunities & ReturnGamesPlayed)

...Seem to show the strongest relationship toward accurate modeling (R^2) during repeat
"""
```

```
Scores
DoubleFaults & BreakPointsFaced: 0.7885528141690712
BreakPointsFaced & ServiceGamesPlayed: 0.8567337073518306
ServiceGamesPlayed & BreakPointsOpportunities: 0.8490642759917163
BreakPointsOpportunities & ReturnGamesPlayed: 0.8451526120806646
```

Out[41]: '\nServiceGamesPlayed & BreakPointsOpportunities seem to show the strongest relationship toward accurate modeling\n'

Perform Multi Feature Linear Regressions

In [48]:

```
#Last: We run 1 Linear Regression with all the variables targeting toward the winnings
features = df[['FirstServe','FirstServePointsWon','FirstServeReturnPointsWon',
'SecondServePointsWon','SecondServeReturnPointsWon','Aces',
'BreakPointsConverted','BreakPointsFaced','BreakPointsOpportunities',
'BreakPointsSaved','DoubleFaults','ReturnGamesPlayed','ReturnGamesWon',
'ReturnPointsWon','ServiceGamesPlayed','ServiceGamesWon','TotalPointsWon',
'TotalServicePointsWon']]
outcome = df[['Winnings']]

x_train, x_test, y_train, y_test = train_test_split(features, outcomes, train_size=.8,

model = LinearRegression()
model.fit(x_train, y_train)
score = model.score(x_test, y_test)
print('Multi Feature Score: {}'.format(score))

#And if we remove a few of the features we no show no strong correlation against winnin
#Such as:

features = df[['FirstServe','FirstServePointsWon',
'SecondServePointsWon','Aces','BreakPointsFaced','BreakPointsOpportunities',
'BreakPointsSaved','DoubleFaults','ReturnGamesPlayed','ServiceGamesPlayed']]
outcome = df[['Winnings']]

x_train, x_test, y_train, y_test = train_test_split(features, outcomes, train_size=.8,

model = LinearRegression()
model.fit(x_train, y_train)
score = model.score(x_test, y_test)
print('Multi Feature Optimized Score: {}'.format(score))
```

```
Multi Feature Score: 0.8179549216035558
Multi Feature Optimized Score: 0.8610701790263873
```