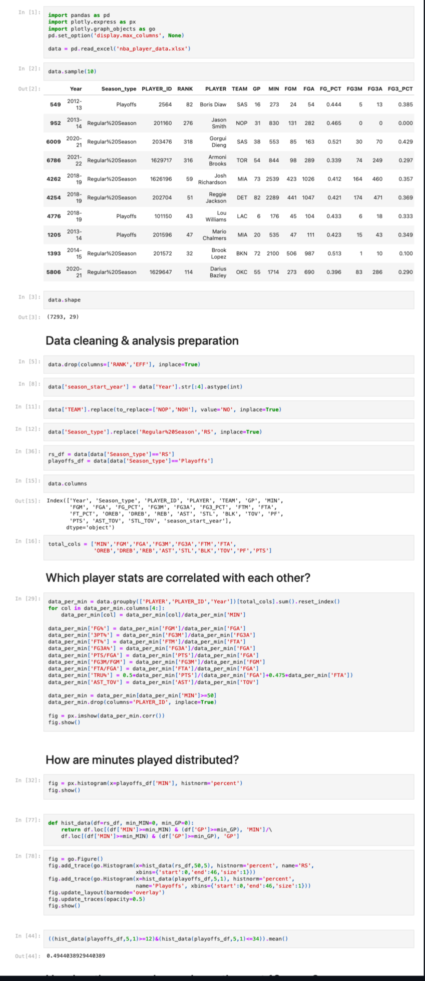

In [1]:
```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
pd.set_option('display.max_columns', None)

data = pd.read_excel('nba_player_data.xlsx')
```

In [2]:
```
data.sample(10)
```

Out[2]:

	Year	Season_type	PLAYER_ID	RANK	PLAYER	TEAM	GP	MIN	FGM	FGA	FG_PCT	FG3M	FG3A	FG3_PCT
549	2012-13	Playoffs	2564	82	Boris Diaw	SAS	16	273	24	54	0.444	5	13	0.385
952	2013-14	Regular%20Season	201160	276	Jason Smith	NOP	31	830	131	282	0.465	0	0	0.000
6009	2020-21	Regular%20Season	203476	318	Gorgui Dieng	SAS	38	553	85	163	0.521	30	70	0.429
6786	2021-22	Regular%20Season	1629717	316	Armoni Brooks	TOR	54	844	98	289	0.339	74	249	0.297
4262	2018-19	Regular%20Season	1626196	59	Josh Richardson	MIA	73	2539	423	1026	0.412	164	460	0.357
4254	2018-19	Regular%20Season	202704	51	Reggie Jackson	DET	82	2289	441	1047	0.421	174	471	0.369
4776	2018-19	Playoffs	101150	43	Lou Williams	LAC	6	176	45	104	0.433	6	18	0.333
1205	2013-14	Playoffs	201596	47	Mario Chalmers	MIA	20	535	47	111	0.423	15	43	0.349
1393	2014-15	Regular%20Season	201572	32	Brook Lopez	BKN	72	2100	506	987	0.513	1	10	0.100
5806	2020-21	Regular%20Season	1629647	114	Darius Bazley	OKC	55	1714	273	690	0.396	83	286	0.290

In [3]:
```
data.shape
```

Out[3]:
```
(7293, 29)
```

Data cleaning & analysis preparation

In [5]:
```
data.drop(columns=['RANK','EFF'], inplace=True)
```

In [8]:
```
data['season_start_year'] = data['Year'].str[:4].astype(int)
```

In [11]:
```
data['TEAM'].replace(to_replace=['NOP','NOH'], value='NO', inplace=True)
```

In [12]:
```
data['Season_type'].replace('Regular%20Season','RS', inplace=True)
```

In [36]:
```
rs_df = data[data['Season_type']=='RS']
playoffs_df = data[data['Season_type']=='Playoffs']
```

In [15]:
```
data.columns
```

Out[15]:
```
Index(['Year', 'Season_type', 'PLAYER_ID', 'PLAYER', 'TEAM', 'GP', 'MIN',
       'FGM', 'FGA', 'FG_PCT', 'FG3M', 'FG3A', 'FG3_PCT', 'FTM', 'FTA',
       'FT_PCT', 'OREB', 'DREB', 'REB', 'AST', 'STL', 'BLK', 'TOV', 'PF',
       'PTS', 'AST_TOV', 'STL_TOV', 'season_start_year'],
      dtype='object')
```

In [16]:
```
total_cols = ['MIN','FGM','FGA','FG3M','FG3A','FTM','FTA',
              'OREB','DREB','REB','AST','STL','BLK','TOV','PF','PTS']
```

Which player stats are correlated with each other?

In [29]:
```
data_per_min = data.groupby(['PLAYER','PLAYER_ID','Year'])[total_cols].sum().reset_index()
for col in data_per_min.columns[4:]:
    data_per_min[col] = data_per_min[col]/data_per_min['MIN']

data_per_min['FG%'] = data_per_min['FGM']/data_per_min['FGA']
data_per_min['3PT%'] = data_per_min['FG3M']/data_per_min['FG3A']
data_per_min['FT%'] = data_per_min['FTM']/data_per_min['FTA']
data_per_min['FG3A%'] = data_per_min['FG3A']/data_per_min['FGA']
data_per_min['PTS/FGA'] = data_per_min['PTS']/data_per_min['FGA']
data_per_min['FG3M/FGM'] = data_per_min['FG3M']/data_per_min['FGM']
data_per_min['FTA/FGA'] = data_per_min['FTA']/data_per_min['FGA']
data_per_min['TRU%'] = 0.5*data_per_min['PTS']/(data_per_min['FGA']+0.475*data_per_min['FTA'])
data_per_min['AST_TOV'] = data_per_min['AST']/data_per_min['TOV']

data_per_min = data_per_min[data_per_min['MIN']>=50]
data_per_min.drop(columns='PLAYER_ID', inplace=True)

fig = px.imshow(data_per_min.corr())
fig.show()
```

How are minutes played distributed?

In [32]:
```
fig = px.histogram(x=playoffs_df['MIN'], histnorm='percent')
fig.show()
```

In [77]:
```
def hist_data(df=rs_df, min_MIN=0, min_GP=0):
    return df.loc[(df['MIN']>=min_MIN) & (df['GP']>=min_GP), 'MIN']/\
    df.loc[(df['MIN']>=min_MIN) & (df['GP']>=min_GP), 'GP']
```

In [78]:
```
fig = go.Figure()
fig.add_trace(go.Histogram(x=hist_data(rs_df,50,5), histnorm='percent', name='RS',
                           xbins={'start':0,'end':46,'size':1}))
fig.add_trace(go.Histogram(x=hist_data(playoffs_df,5,1), histnorm='percent',
                           name='Playoffs', xbins={'start':0,'end':46,'size':1}))
fig.update_layout(barmode='overlay')
fig.update_traces(opacity=0.5)
fig.show()
```

In [44]:
```
((hist_data(playoffs_df,5,1)>=12)&(hist_data(playoffs_df,5,1)<=34)).mean()
```

Out[44]:
```
0.4944038929440389
```