NBA Shot Analysis

Buzzer Beaters

- Everyone remembers buzzer beaters, but how accurate are they?
- Who is the best in the clutch?
- When is the best time to shoot the ball?

About Our Data

- 2014-15 NBA Season
- 281 Unique Shooters
- 474 Unique Defenders
- 128069 Shots Taken
- We have 1 pair of players with the same name.

Data Wrangling

- Shot clock is not started if there is <24 seconds in the quarter.
  - As a result, I had to fill NaN with how many seconds were left on the game clock.
  - There were also NaN values for other game clock values, those had to be dropped.

```
cleaned_df["SHOT_CLOCK"].fillna(cleaned_df["SECONDS"], inplace=True)
```

- Dropped some columns we weren’t using.
- Game clock column is object due to “minutes: seconds”. Must convert.

Splitting the Clock

```
to_clean= cleaned_df["GAME_CLOCK"].str.split(":",expand=True).astype(int)
cleaned_df["GAME_CLOCK_SECONDS"]=to_clean[0]*60 + to_clean[1]
```

```
to_clean.rename(columns={0: "MINUTES", 1: "SECONDS"}, inplace=True)
# print(to_clean)
#merge into cleaned_df
cleaned_df=pd.concat([cleaned_df, to_clean], axis=1)
```

Adding Game Time in Seconds

```
cleaned_df["TOTAL_TIME_SECONDS"] = ""
for index,rows in cleaned_df.iterrows():
    multiplier= 0
    if rows["PERIOD"] == 1:
        multiplier= 36*60
    elif rows["PERIOD"] == 2:
        multiplier= 24*60
    elif rows["PERIOD"] == 3:
        multiplier= 12*60
    cleaned_df.loc[index,["TOTAL_TIME_SECONDS"]]=multiplier+cleaned_df["GAME_CLOCK_SECONDS"][index]

cleaned_df
```

Who is the most clutch?

- Clutch is < 24 seconds left in game
  - 1 score game (+/- 3 pts)
- Ranking regardless of Win or Loss

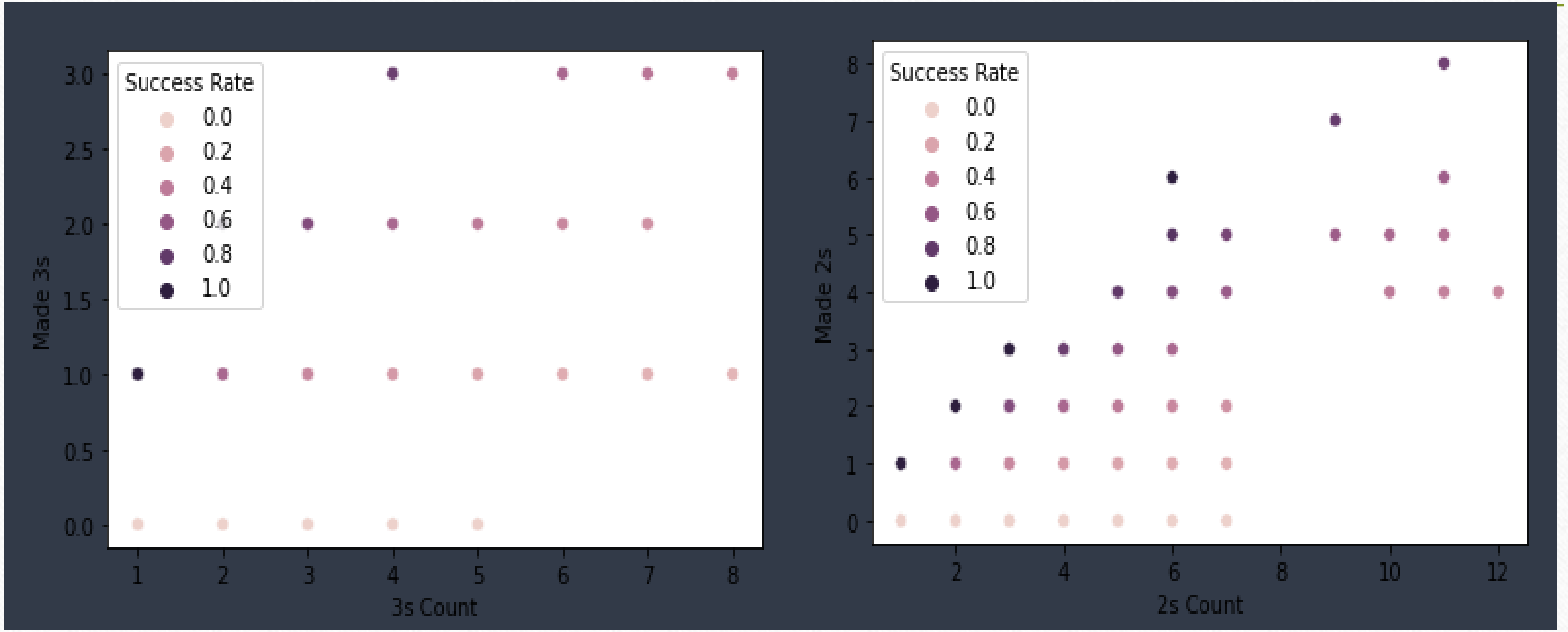
PLAYER_NAME	Shot Count	Made Shots	Success Rate	Weighted Rank
brandon knight	17	10	0.588235	2.647059
james harden	11	8	0.727273	2.181818
jarrett jack	12	8	0.666667	2.166667
tyreke evans	18	8	0.444444	2.111111
manu ginobili	17	7	0.411765	1.852941
lou williams	10	6	0.600000	1.650000
tim duncan	13	6	0.461538	1.615385
john wall	17	6	0.352941	1.588235
nikola vucevic	6	5	0.833333	1.458333
anthony davis	8	5	0.625000	1.406250

2 Pointers v 3 Pointers

PLAYER_NAME	2s Count	Made 2s	Missed 2s	Success Rate	Weighted Rank
brandon knight	11	8	3	0.727273	2.181818
jarrett jack	9	7	2	0.777778	1.944444
lou williams	6	6	0	1.000000	1.750000
manu ginobili	11	6	5	0.545455	1.636364
tim duncan	11	6	5	0.545455	1.636364
nikola vucevic	6	5	1	0.833333	1.458333
james harden	7	5	2	0.714286	1.428571
john wall	9	5	4	0.555556	1.388889
chris paul	10	5	5	0.500000	1.375000
tyreke evans	11	5	6	0.454545	1.363636

PLAYER_NAME	3s Count	Made 3s	Missed 3s	Success Rate	Weighted Rank
james harden	4	3	1	0.750000	0.937500
danny green	4	3	1	0.750000	0.937500
nick young	6	3	3	0.500000	0.875000
tyreke evans	7	3	4	0.428571	0.857143
trey burke	8	3	5	0.375000	0.843750
kyrie irving	2	2	0	1.000000	0.750000
kyle korver	2	2	0	1.000000	0.750000
stephen curry	3	2	1	0.666667	0.666667
evan turner	3	2	1	0.666667	0.666667
avery bradley	3	2	1	0.666667	0.666667

3 pointers v 2 pointers

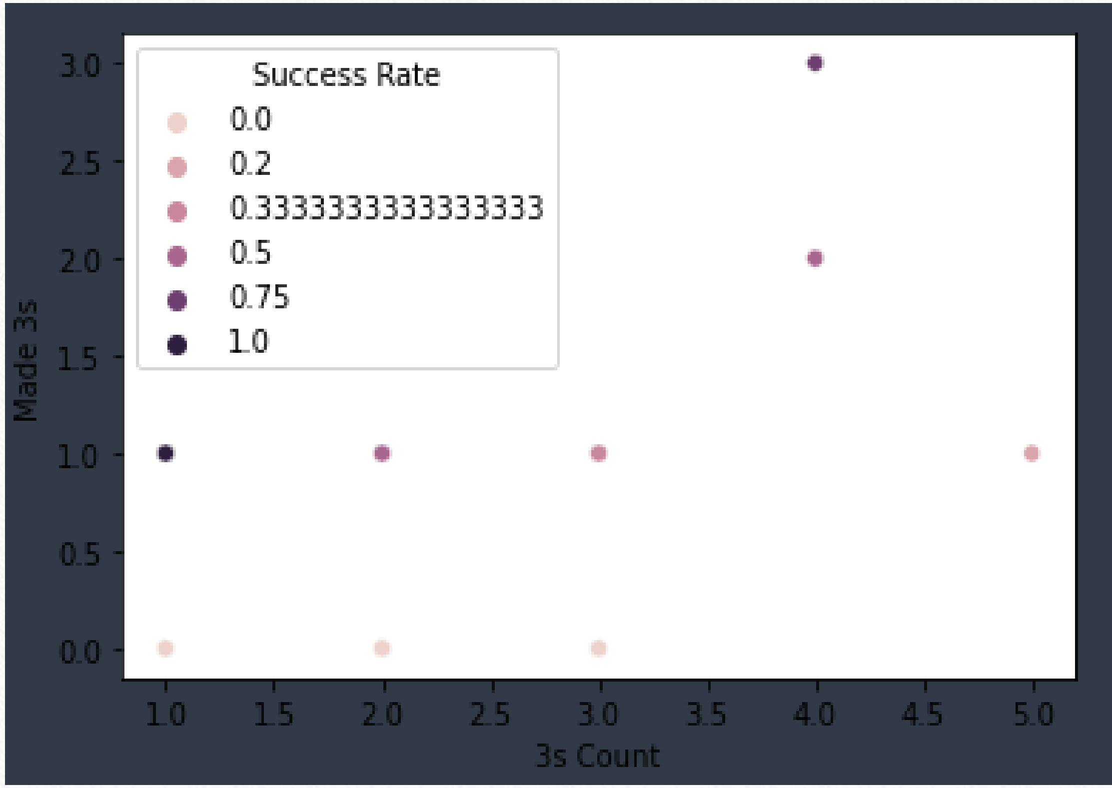


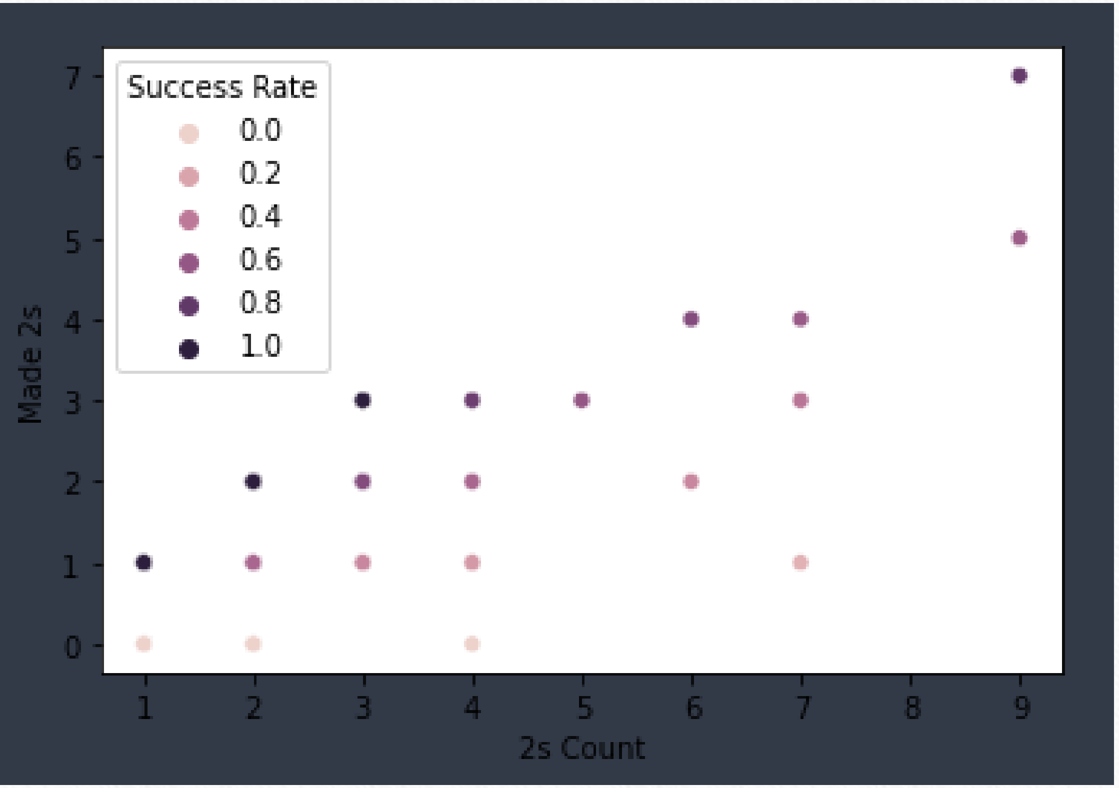
Winners v Losers

PLAYER_NAME	Shot Count	Made Shots	Success Rate	Weighted Rank
jarrett jack	12	8	0.666667	2.166667
tyreke evans	13	8	0.615385	2.153846
kemba walker	12	5	0.416667	1.354167
james harden	4	4	1.000000	1.250000
marc gasol	4	4	1.000000	1.250000
brandon knight	6	4	0.666667	1.166667
anthony davis	6	4	0.666667	1.166667
russell westbrook	7	4	0.571429	1.142857
brandon jennings	7	4	0.571429	1.142857
mike conley	8	4	0.500000	1.125000

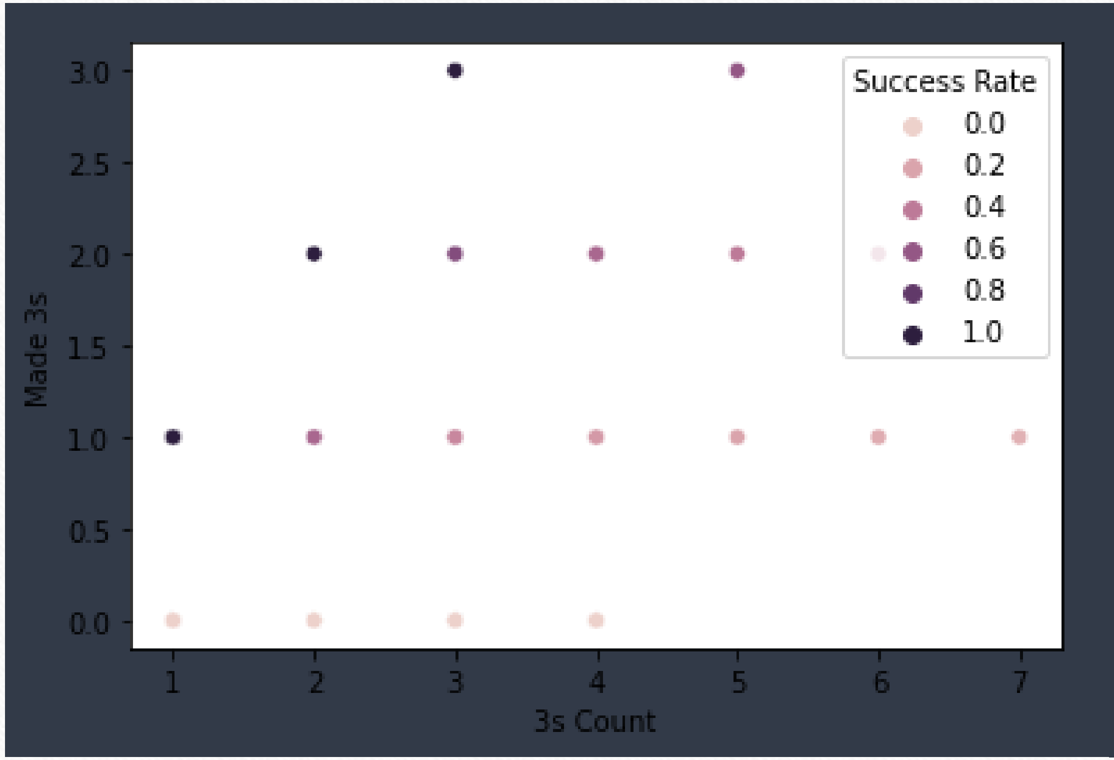
PLAYER_NAME	Shot Count	Made Shots	Success Rate	Weighted Rank
brandon knight	11	6	0.545455	1.636364
chase budinger	5	4	0.800000	1.200000
zaza pachulia	6	4	0.666667	1.166667
lou williams	7	4	0.571429	1.142857
james harden	7	4	0.571429	1.142857
goran dragic	9	4	0.444444	1.111111
tim duncan	10	4	0.400000	1.100000
manu ginobili	13	4	0.307692	1.076923
danny green	3	3	1.000000	1.000000
luis scola	3	3	1.000000	1.000000

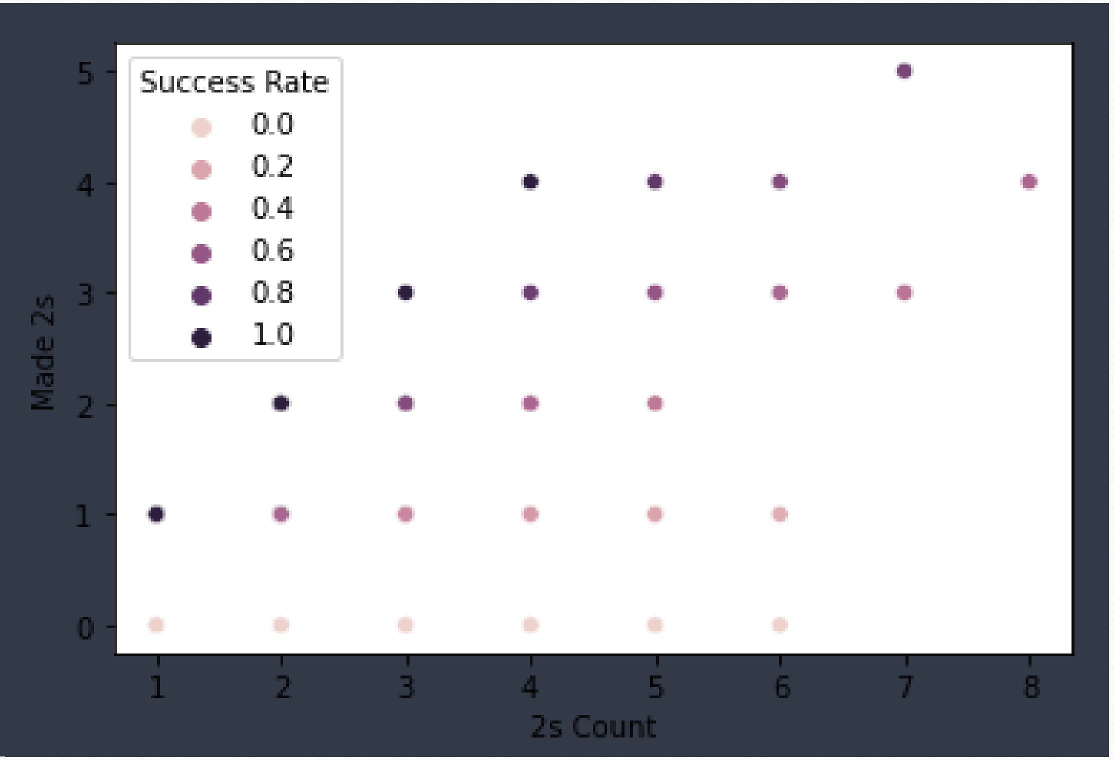
Winners 3 and 2 pointers





Losers 3 and 2 pointers





Are Open Shots Made More Often Than Contested Shots?

- What distance from the defender is considered open?
  - We used 6ft as the cut off.

- Our hypothesis: Open shots will be made at a higher frequency than contested shots.

Finding Defender Distance

Open

12]:
```
#open shots made percentage, finding shots 6ft or greater from closest defender
open_shots = cleaned_df.loc[cleaned_df["CLOSE_DEF_DIST"] >=6]
open_shots
```

12]:

.OSEST_DEFENDER_PLAYER_ID	CLOSE_DEF_DIST	FGM	PTS	player_name	player_id	GAME_CLOC
202711	6.1	0	0	brian roberts	203148	
101127	6.1	0	0	brian roberts	203148	
202721	7.3	0	0	brian roberts	203148	
201961	19.8	0	0	brian roberts	203148	
202738	11.1	1	3	brian roberts	203148	
...	...	...	...	...	...	
201596	8.3	1	2	jarrett jack	101127	
203095	6.9	1	2	jarrett jack	101127	
201147	11.1	1	2	jarrett jack	101127	
201593	6.0	1	2	jarrett jack	101127	
2590	7.7	1	2	jarrett jack	101127	

Contested

]:
```
#finding defenders less than 6ft
in_your_face = cleaned_df.loc[cleaned_df["CLOSE_DEF_DIST"] <6]
in_your_face
```

]:

...	CLOSEST_DEFENDER_PLAYER_ID	CLOSE_DEF_DIST	FGM	PTS	player_name	pl
...	101187	1.3	1	2	brian roberts	
...	202711	0.9	0	0	brian roberts	
...	203900	3.4	0	0	brian roberts	
...	201152	1.1	0	0	brian roberts	
...	101114	2.6	0	0	brian roberts	
...	...	...	...	...	...	
...	203935	0.8	0	0	jarrett jack	
...	202323	0.6	1	2	jarrett jack	
...	201977	4.2	1	2	jarrett jack	
...	202340	3.0	0	0	jarrett jack	
...	202340	2.3	1	2	jarrett jack	

Finding Goals Made

Open

```
[13]: #find shots that were made from these open shots
open_made = open_shots.loc[open_shots["FGM"] == 1]
open_made
```

[13]:

E_CLOCK	SHOT_CLOCK	SHOT_DIST	PTS_TYPE	SHOT_RESULT	...	CLOSEST_DEFENDER_PLAYER_ID	CLOSE_DEF_DIST
10:29	20.8	24.2	3	made	...	202738	11.1
10:13	17.1	24.6	3	made	...	201149	6.0
5:58	19.7	24.7	3	made	...	101139	11.3
5:21	12.8	22.5	3	made	...	201588	7.2
5:19	9.4	23.1	3	made	...	201228	9.1
...	...	...	...	...	...	...	...
10:42	2.9	17.6	2	made	...	201596	8.3
0:55	21.8	17.0	2	made	...	203095	6.9
11:01	9	20.2	2	made	...	201147	11.1
7:27	11	9.4	2	made	...	201593	6.0
6:41	22.7	2.3	2	made	...	2590	7.7

Contested

```
[28]: #shots made with defenders less than 6ft
contested_made = in_your_face.loc[in_your_face["FGM"] == 1]
contested_made
```

[28]:

LOCK	SHOT_CLOCK	SHOT_DIST	PTS_TYPE	SHOT_RESULT	...	CLOSEST_DEFENDER_PLAYER_ID	CLOSE_DEF_DIST
1:09	10.8	7.7	2	made	...	101187	1.3
8:00	3.4	3.5	2	made	...	203486	2.1
11:32	12.1	14.6	2	made	...	202391	1.8
8:55	4.3	5.9	2	made	...	201941	5.4
10:38	6.4	24.7	3	made	...	203923	5.6
...	...	...	...	...	...	...	...
7:46	7	14.5	2	made	...	203935	3.1
5:05	15.3	8.9	2	made	...	203096	5.7
11:28	19.8	0.6	2	made	...	202323	0.6
11:10	23	16.9	2	made	...	201977	4.2
0:12	0:12	5.1	2	made	...	202340	2.3

Total Open Shots Attempted: 10,226

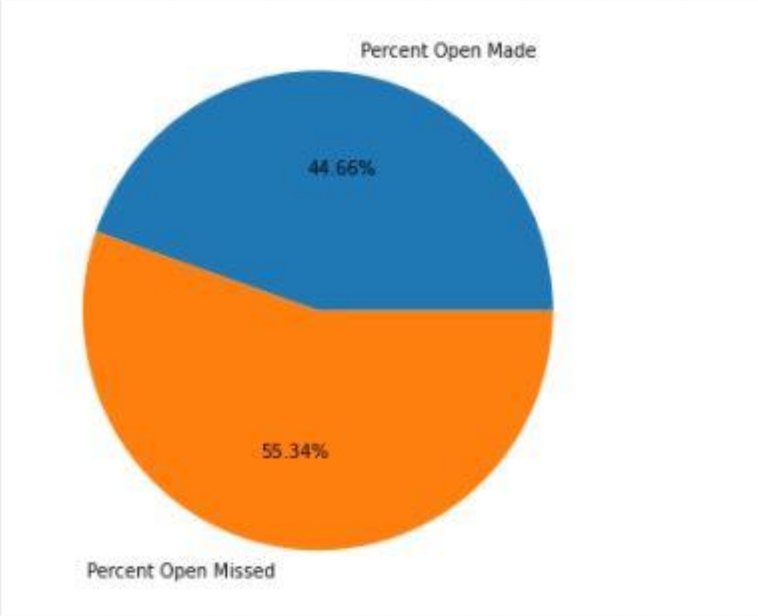
Total Contested Shots Attempted: 47,679

Percent Open Made

```
#percentage made
percentage_open_made = len(open_made)/ len(open_shots) *100
open_made["Percent Open Made"] = percentage_open_made
open_missed = 100 - percentage_open_made
open_made["Percent Open Missed"] = open_missed
open_made
#pie chart for open shots
pie, ax = plt.subplots(figsize=[10,6])
labels = 'Percent Open Made', 'Percent Open Missed'
size = [44.66, 55.34]
ax.pie(size, labels=labels, autopct='%1.2f%%')
pie.savefig("Open.png")
```

```
#player that made the most open shots
all_day = open_made['player_name'].value_counts()
all_day
```

```
al horford           134
blake griffin        129
serge ibaka          104
stephen curry        102
lamarcus aldridge     91
```



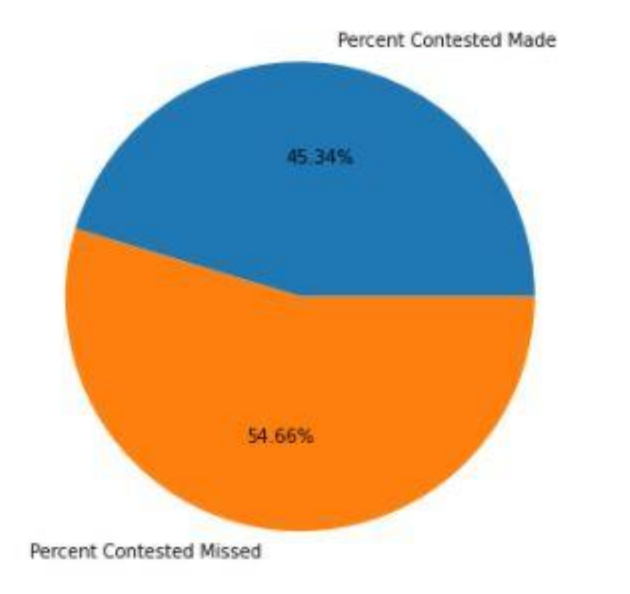
Contested Made

```
#percentage made with close defender
percentage_close_made = len(closeness_made)/ len(in_your_face) *100
contested_made["Percent Contested Made"] = percentage_close_made
contested_missed = 100 - percentage_close_made
contested_missed
contested_made["Percent Contested Missed"] = contested_missed
contested_made

#total_shots_made = len(closeness_made) + len(open_made)
#creating pie chart
pie, ax = plt.subplots(figsize=[10,6])
labels = 'Percent Contested Made', 'Percent Contested Missed'
size = [45.34, 54.66]
ax.pie(size, labels=labels, autopct='%1.2f%%')
pie.savefig("Contested.png")
```

```
#player that made the most contested shots
try_me = contested_made['player_name'].value_counts()
try_me
```

james harden	423
lebron james	410
anthony davis	401
nikola vucevic	400
mnta ellis	399



Conclusion

- Contested shots were made more often than open by only 0.68%.

Who are the NBA's best defensive players based on FG block %?

- How will this be determined/What are our parameters?

- Hypothesis: Whoever they are, their block percentages should roughly add up to 100% when combined with FG success rates.

First Thought ….

In [14]:
```
# Top Performing Defenders
#Sort and display the top ten defensive players by % missed FG's as the closest defender.
top_defense = defender_df.sort_values("Success Rate", ascending=False)
top_defense.head(10)
```

Out[14]:

CLOSEST_DEFENDER	No. of Times Closest Defender	No. of FG made against	No. of successful defenses	Success Rate
Dragic, Zoran	2	0	2	1.00
Ledo, Ricky	1	0	1	1.00
Lucas, Kalin	1	0	1	1.00
James, Bernard	32	7	25	0.78
Udoh, Ekpe	13	3	10	0.77
Green, JaMychal	13	3	10	0.77
Datome, Gigi	4	1	3	0.75
Robinson, Nate	99	27	72	0.73
Mekel, Gal	10	3	7	0.70
Jenkins, John	16	5	11	0.69

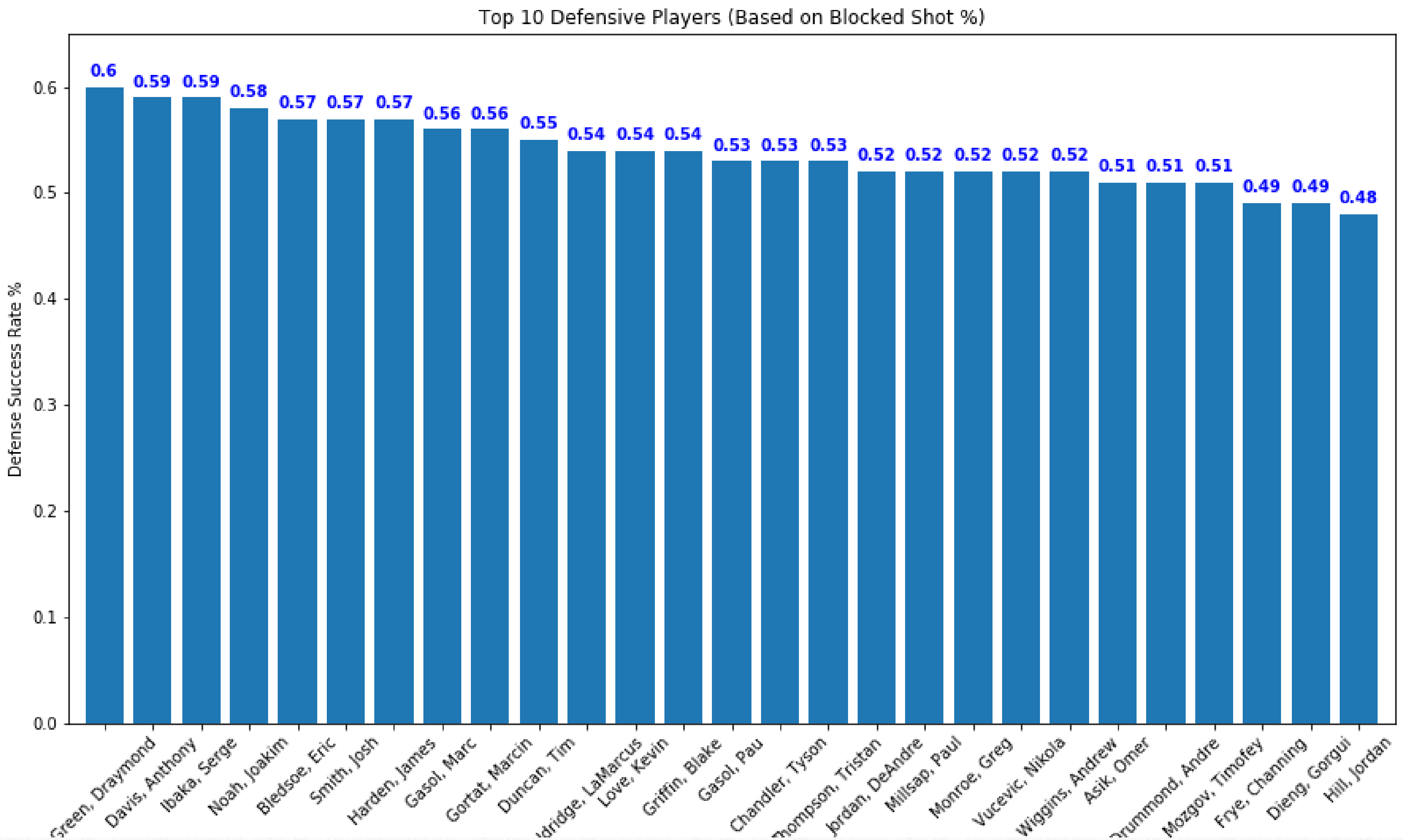
Re-evaluate…

In [15]:

```
# Top frequent defenders
#Sort and display the top ten defensive players by frequency as closest defender.
top_defense = defender_df.sort_values("No. of Times Closest Defender", ascending=False)
top_defense.head(10)
```

Out[15]:

CLOSEST_DEFENDER	No. of Times Closest Defender	No. of FG made against	No. of successful defenses	Success Rate
Ibaka, Serge	814	334	480	0.59
Jordan, DeAndre	795	381	414	0.52
Gasol, Pau	754	354	400	0.53
Green, Draymond	751	301	450	0.60
Millsap, Paul	750	357	393	0.52
Chandler, Tyson	698	329	369	0.53
Vucevic, Nikola	697	335	362	0.52
Frye, Channing	693	355	338	0.49
Love, Kevin	691	315	376	0.54
Gortat, Marcin	688	302	386	0.56



Top 10 Defensive Players (Based on Blocked Shot %)
Defense Success Rate %
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.6
0.59
0.59
0.58
0.57
0.57
0.57
0.56
0.56
0.55
0.54
0.54
0.54
0.53
0.53
0.53
0.52
0.52
0.52
0.52
0.52
0.51
0.51
0.51
0.49
0.49
0.48
Davis, Anthony
Ibaka, Serge
Noah, Joakim
Bledsoe, Eric
Smith, Josh
Harden, James
Gasol, Marc
Gortat, Marcin
Duncan, Tim
Love, Kevin
Griffin, Blake
Gasol, Pau
Chandler, Tyson
Jordan, DeAndre
Millsap, Paul
Monroe, Greg
Vucevic, Nikola
Wiggins, Andrew
Asik, Omer
Drummond, Andre
Mozgov, Timofey
Frye, Channing
Dieng, Gorgui
Hill, Jordan

Is FG success rate affected by quarter or total time left on game clock?

- Hypothesis: Yes, the further along in the game, the more tired the players will be and thus, we will be likely to see a decrease in success rate as game progresses.
- What calculations and functions did we use to determine this?

- FG success rate does decrease by quarter except, it appears, right after half-time when they've all had time to rest.
- NBA's best defenders compared to FG success rates:

Time Remaining on Game Clock (Seconds)	Shot Success Rate (%)
720-0 (4th QTR)	43.90%
1439-720 (3rd QTR)	45.72%
2159-1440 (2nd QTR)	45.11%
2880-2160 (1st QTR)	46.05%

CLOSEST_DEFENDER	No. of Times Closest Defender	No. of FG made against	No. of successful defenses	Success Rate
Green, Draymond	751	301	450	0.600000
Davis, Anthony	609	247	362	0.590000
Ibaka, Serge	814	334	480	0.590000
Noah, Joakim	632	267	365	0.580000
Bledsoe, Eric	650	280	370	0.570000
Smith, Josh	651	278	373	0.570000
Harden, James	607	264	343	0.570000
Gasol, Marc	629	274	355	0.560000
Gortat, Marcin	688	302	386	0.560000
Duncan, Tim	637	286	351	0.550000

Does field goal percentage (FG%) decrease in the final 5 seconds of the shot clock?

- Is there a correlation between FG% and the time remaining on the 24 second shot clock?
- Hypothesis – Yes, there is a correlation. We think the FG% will be lower in the last 5 seconds of the shot clock.
- How will we test our hypothesis?

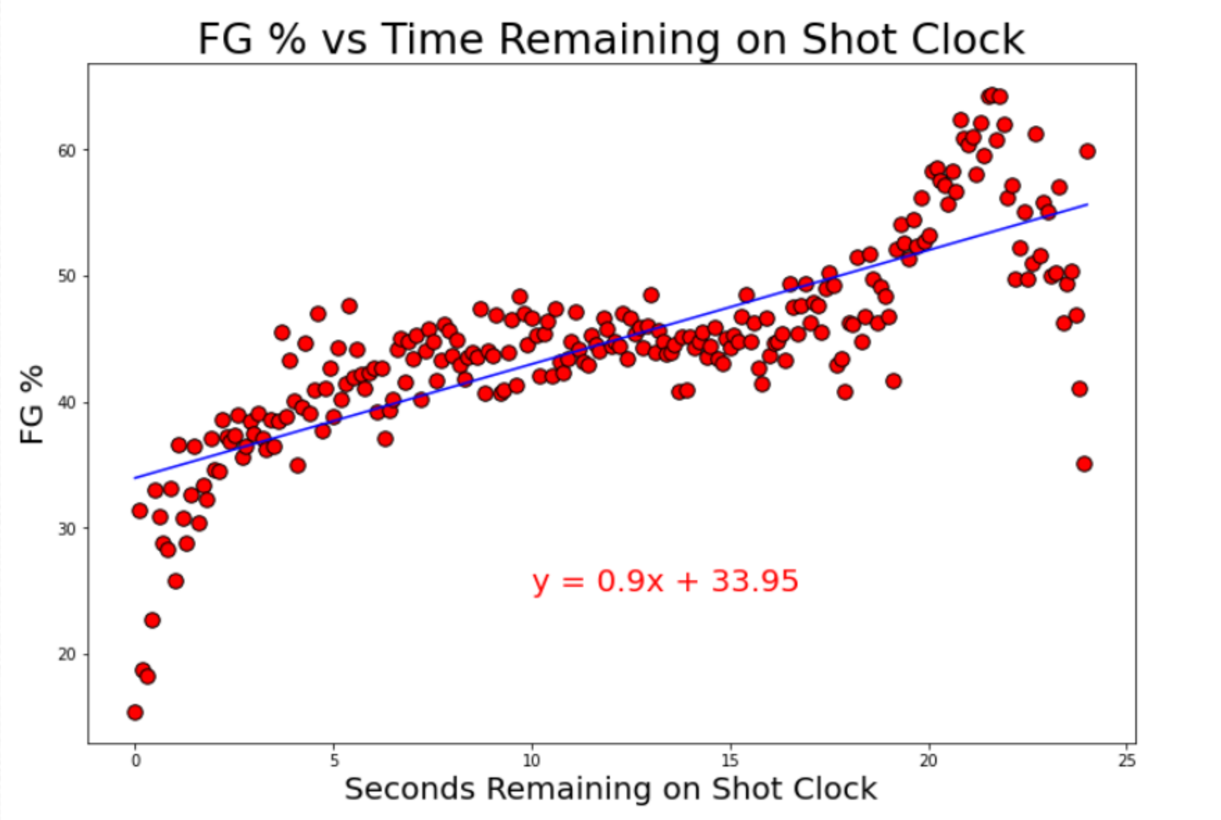
```python
# find all the different timepoints
shot_clock_timepoints = cleaned_df.groupby(["SHOT_CLOCK"])
shot_clock_remaining = shot_clock_timepoints["SHOT_CLOCK"].unique()
# calculate number of shots at each timepoint
shot_clock_shots = shot_clock_timepoints["SHOT_CLOCK"].count()
# calculate number of shots made at each timepoint
shots_made = shot_clock_timepoints["FGM"].sum()
# calculate FG % at each timepoint
fg_percent = shots_made / shot_clock_shots * 100

# make a new dataframe for graphing
graph_df = pd.DataFrame({
    "FG Attempts": shot_clock_shots,
    "FGs Made": shots_made,
    "FG%": fg_percent
})

# reset the index
graph_df = graph_df.reset_index()
graph_df = graph_df.rename(columns={"SHOT_CLOCK": "Shot Clock"})

graph_df
```

Correlation between FGs and time left on shot clock



- The r-value is: 0.803
- There is a strong correlation between these data points

Is there a difference between 2-point shots and 3-point shots?

```python
two_pointer = cleaned_df.loc[cleaned_df["PTS_TYPE"] <= 2]
three_pointer = cleaned_df.loc[cleaned_df["PTS_TYPE"] > 2]
```

	GAME_ID	W	FINAL_MARGIN	SHOT_NUMBER	PERIOD	GAME_CLOCK	SHOT_CLOCK	SHOT_DIST	PTS_TYPE
0	21400899	W	24	1	1	1:09	10.8	7.7	2
2	21400899	W	24	3	1	0:00	0.0	10.1	2
3	21400899	W	24	4	2	11:47	10.3	17.2	2
4	21400899	W	24	5	2	10:34	10.9	3.7	2
5	21400899	W	24	6	2	8:15	9.1	18.4	2
...	...	...	...	...	...	...	...	...	...
128064	21400006	L	-16	5	3	1:52	18.3	8.7	2
128065	21400006	L	-16	6	4	11:28	19.8	0.6	2
128066	21400006	L	-16	7	4	11:10	23.0	16.9	2
128067	21400006	L	-16	8	4	2:37	9.1	18.3	2
128068	21400006	L	-16	9	4	0:12	12.0	5.1	[illegible]

93288 rows × 21 columns

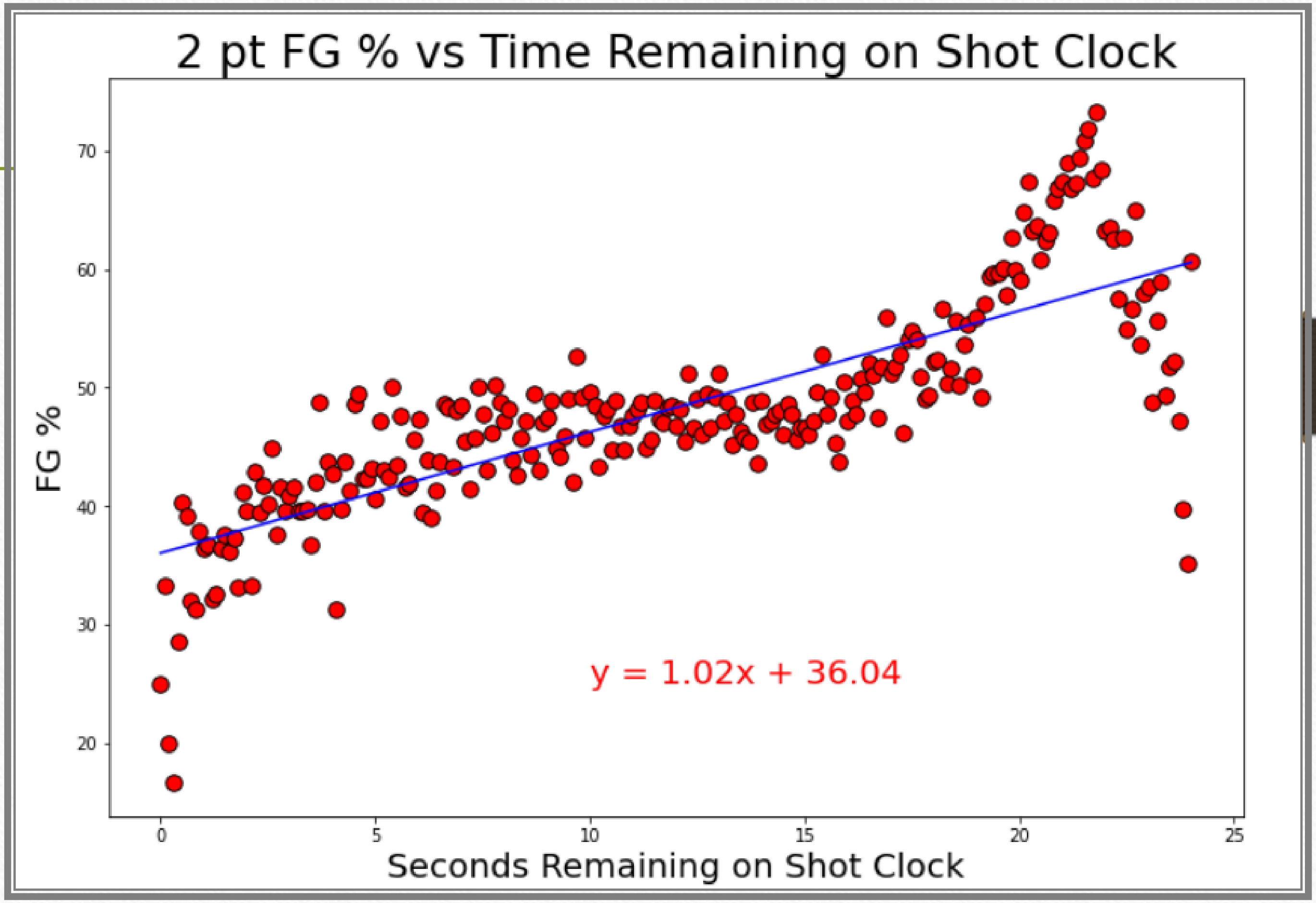
	GAME_ID	W	FINAL_MARGIN	SHOT_NUMBER	PERIOD	GAME_CLOCK	SHOT_CLOCK	SHOT_DIST	PTS_TYPE
1	21400899	W	24	2	1	0:14	3.4	28.2	3
8	21400899	W	24	9	4	5:14	12.4	24.6	3
9	21400890	W	1	1	2	11:32	17.4	22.4	3
10	21400890	W	1	2	2	6:30	16.0	24.5	3
13	21400882	W	15	1	4	9:10	4.4	26.4	3
...	...	...	...	...	...	...	...	...	...
128009	21400138	L	-10	4	3	0:40	16.0	25.9	3
128016	21400138	L	-10	11	4	0:34	21.1	23.0	3
128031	21400116	L	-8	4	4	4:19	14.0	23.8	3
128057	21400033	W	12	2	2	10:02	7.3	22.1	3
128059	21400033	W	12	4	4	8:34	19.8	22.7	3

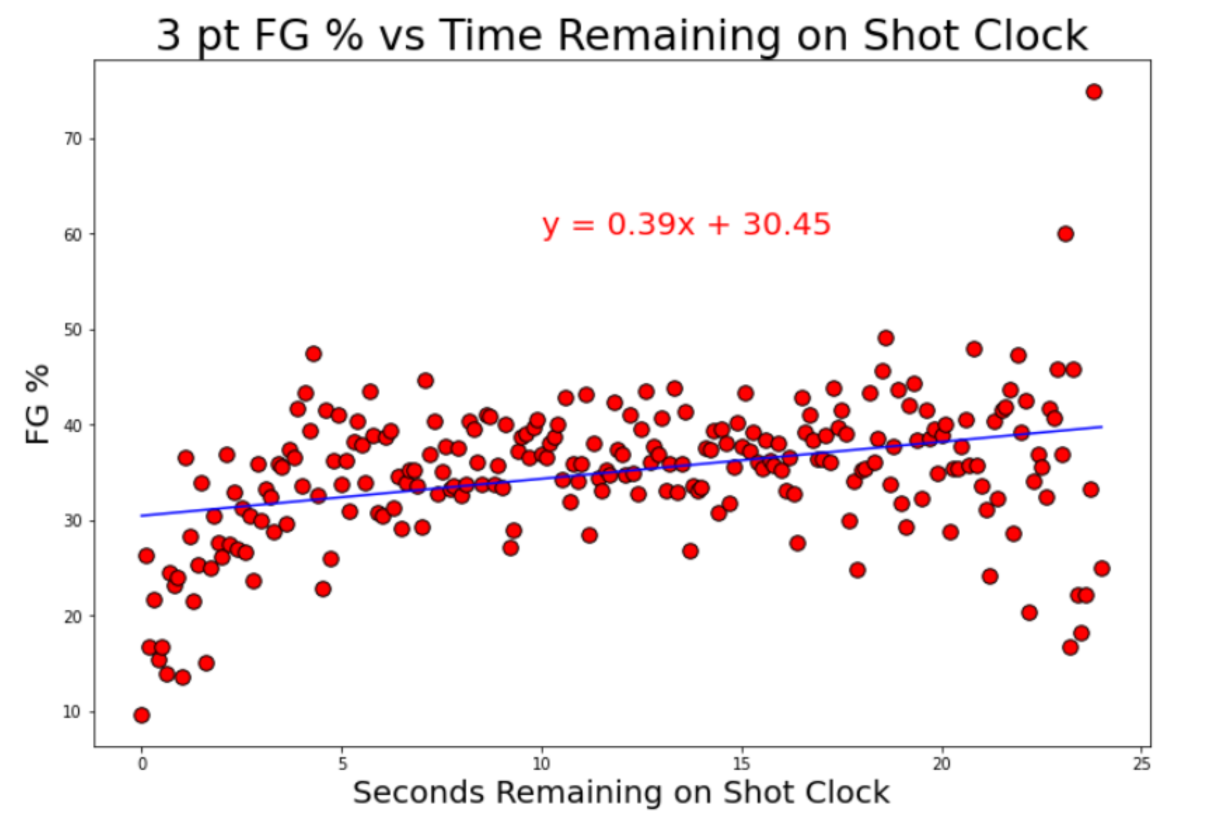
33576 rows × 21 columns

Correlation between 2-point shots and time left on shot clock

- The r-value is: 0.801

- There is a strong correlation between these data points.





Correlation between 3-point shots and time left on shot clock

- The r-value is: 0.365
- There is a weak correlation between these data points.

Conclusion

- The FG% is the lowest in the final 5 seconds of the shot clock.

What is the FG% at other timepoints on the shot clock?

```
# create bins and bin labels
bin_size = [0, 5, 10, 15, 20, 25]
bin_label_size = ["0-5.0", "5.1-11.0", "11.1-15.0", "15.1-20.0", "20.1-24"]

# create index
graph_df["Shot Clock"] = pd.cut(graph_df["Shot Clock"], bin_size, labels=bin_label_size)

# group dataframe by bins
shot_clock_binning = graph_df.groupby(["Shot Clock"])

# create value(s) for dataframe
fg_attempted = shot_clock_binning["FG Attempts"].sum()
fg_made = shot_clock_binning["FGs Made"].sum()
fg_per = fg_made / fg_attempted * 100

# create dataframe
bin_df = pd.DataFrame({
    "FG Attempts": fg_attempted,
    "FGs Made": fg_made,
    "FG%": fg_per
    })

# format cell for cleaner look
bin_df["FG%"] = bin_df["FG%"].map("{:.2f}".format)

bin_df
```

FG% at other shot clock timepoints

	FG Attempts	FGs Made	FG%
Shot Clock			
0-5.0	16140	5904	36.58
5.1-11.0	30698	13380	43.59
11.1-15.0	38555	17234	44.70
15.1-20.0	27391	12958	47.31
20.1-24	13496	7799	57.79

- FG% decreases as shot clock decreases
- Lowest in final 5 seconds
- Highest in first 4 seconds