BI DATA
LA CLIPPERS

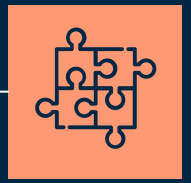
By Rikesh Patel

TABLE OF CONTENTS

01

TICKETS

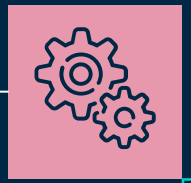
Sales, Accounts, Members



02

MODEL

Choosing which model will provide the most success



03

FORECAST

Integrating everything together to forecast

WHO AM I?

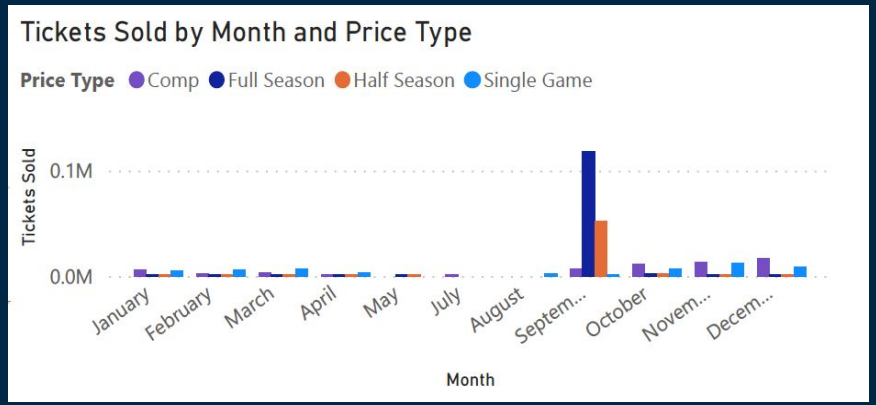
Rikesh Patel
UCLA Undergrad
rikeshpatel.io

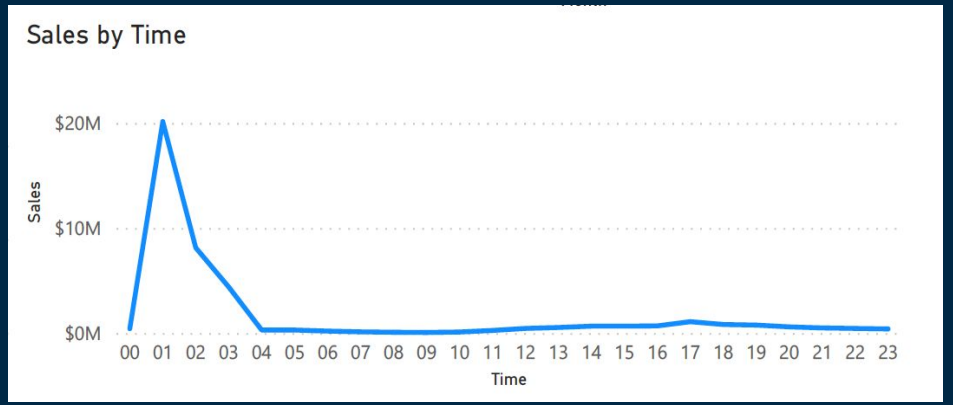
Tickets

01

308.10K	$42.08M	8098
Seats Sold	Total Sales	Accounts

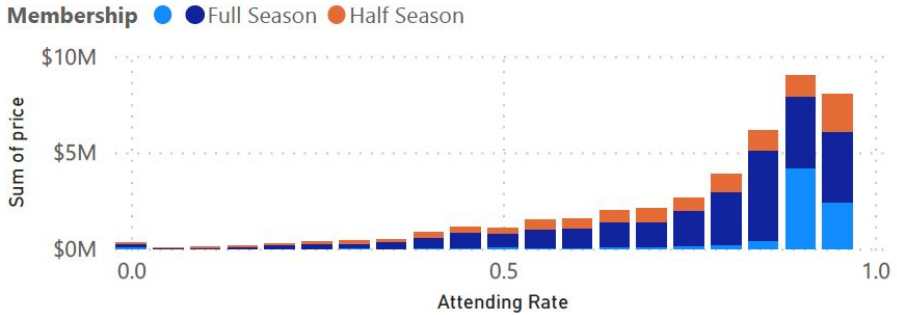
PRIME TIME FOR SALES



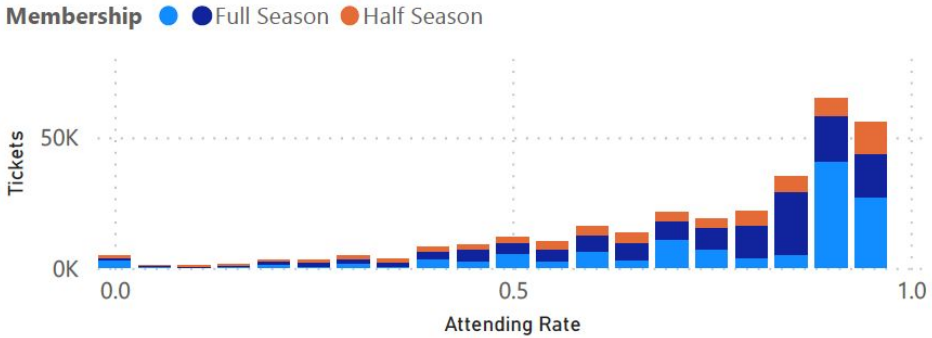


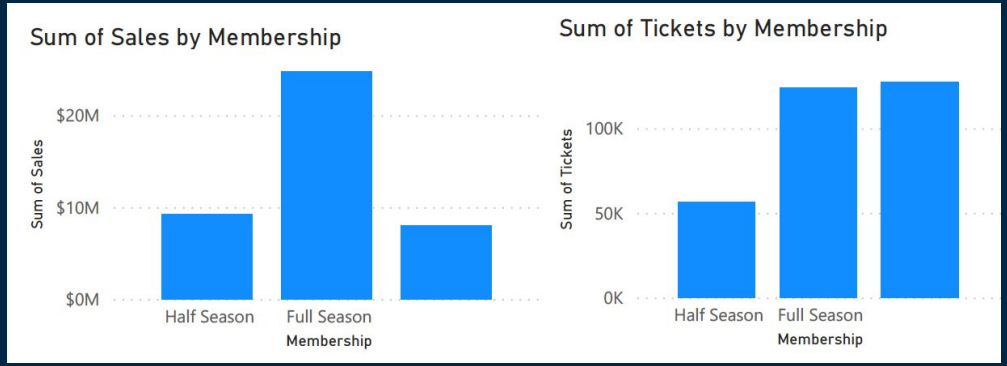
MEMBERSHIPS

Sales by Attending Rate and Membership



Tickets by Attending Rate and Membership




Sum of Sales by Membership
$20M
$10M
$0M
Sum of Sales
Half Season
Full Season
Membership
Sum of Tickets by Membership
100K
50K
0K
Sum of Tickets
Half Season
Full Season
Membership

price_code_type	Attend_Rate	Accounts	AVG_Seats	AVG_Tickets_Bought	AVG_Tickets_Scanned	Tot_AVG_Scans
Comp	64	3113	1512	0.48	0.31	12.92
Full Season	75	1149	2974	2.58	1.96	80.50
Group	88	249	286	1.15	1.02	39.84
Half Season	73	1364	1373	1.00	0.74	30.53
Partial Plan	79	332	120	0.36	0.28	11.81
Single Game	93	2393	1261	0.52	0.49	20.16

Model

02

DATASET

Date	Start (ET)	Opponent	Unnamed: 7	Tm	Opp	W	L	Streak	Occupancy	Vegas Over/Under 21/22	Sales	Attending Rate	Fill Rate	(price, amin)	(price, median)	(price, amax)	Weekday
2021-10-23	10:30p	Memphis Grizzlies	L	114	120	0	2	L 2	6498	41.5	7934	82	78	18.38	107.00	481.00	5
2021-11-16	10:30p	San Antonio Spurs	W	106	92	9	5	W 1	4928	28.5	7126	69	59	20.00	83.00	373.00	1
2021-11-21	3:30p	Dallas Mavericks	W	97	91	10	7	W 1	5085	48.5	6832	74	61	35.76	171.00	573.00	6
2021-11-23	10:30p	Dallas Mavericks	L	104	112	10	8	L 1	5903	48.5	7339	80	70	35.76	159.00	555.00	1
2021-11-26	3:30p	Detroit Pistons	W	107	96	11	8	W 1	5210	24.5	7477	70	62	19.81	83.00	373.00	4
2021-11-28	3:30p	Golden State Warriors	L	90	105	11	9	L 1	6832	48.5	7622	90	82	35.76	281.00	1177.00	6
2021-11-29	10:30p	New Orleans Pelicans	L	104	123	11	10	L 2	4257	39.5	6713	63	51	15.60	83.00	310.00	0
2021-12-01	10:30p	Sacramento Kings	L	115	124	11	11	L 3	4558	36.5	6747	68	54	16.47	77.00	373.00	2
2021-12-08	10:30p	Boston Celtics	W	114	111	14	12	W 2	6179	45.5	7456	83	74	24.23	108.00	410.39	2

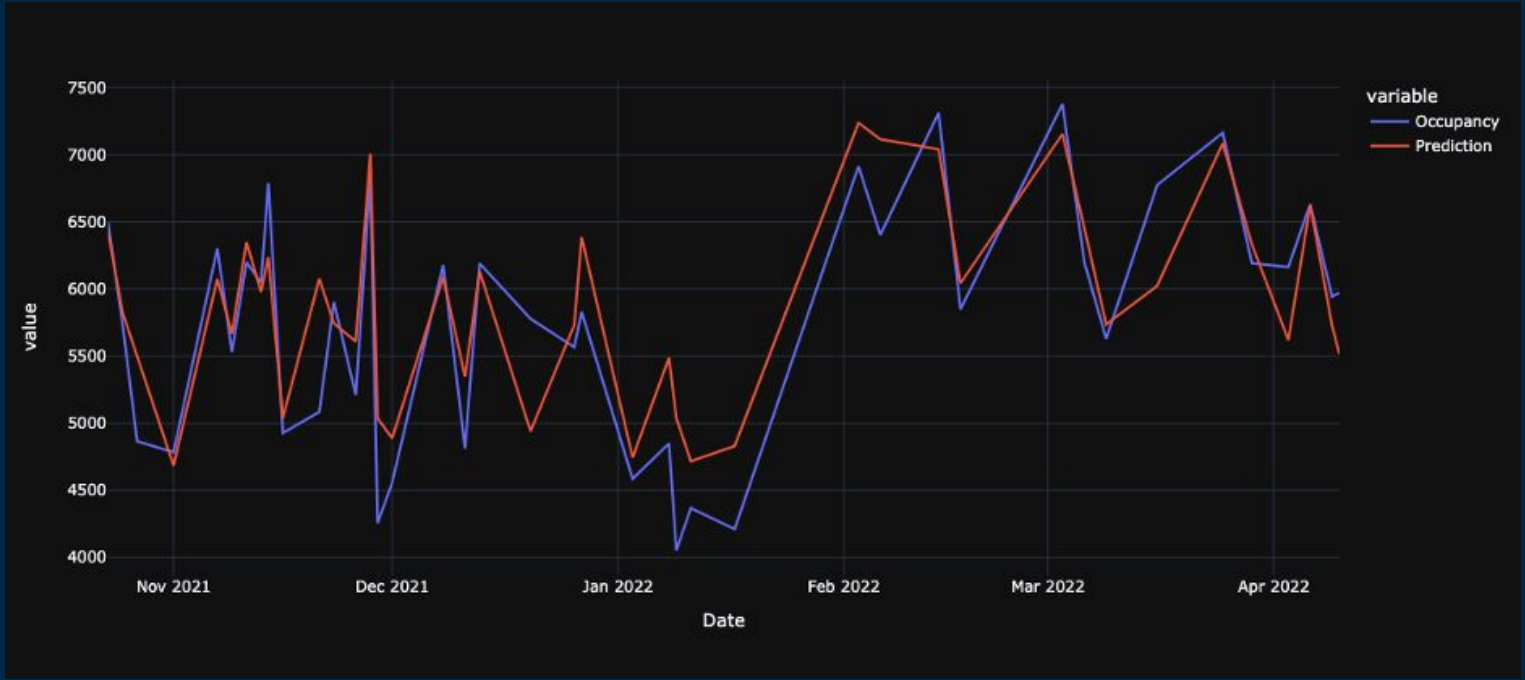
MODELS

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
ridge	Ridge Regression	6.282127e+02	5.679077e+05	7.010735e+02	-2.799000e-01	0.1251	1.150000e-01	0.015
lasso	Lasso Regression	6.369302e+02	5.881137e+05	7.128371e+02	-3.188000e-01	0.1350	1.193000e-01	0.016
ada	AdaBoost Regressor	6.808026e+02	6.550500e+05	7.692246e+02	-4.981000e-01	0.1374	1.254000e-01	0.075
lr	Linear Regression	7.155777e+02	6.898959e+05	7.885182e+02	-8.315000e-01	0.1466	1.325000e-01	0.014
et	Extra Trees Regressor	6.980585e+02	7.461641e+05	7.928199e+02	-1.037500e+00	0.1393	1.240000e-01	0.354
rf	Random Forest Regressor	7.256715e+02	6.825759e+05	8.051119e+02	-9.225000e-01	0.1437	1.310000e-01	0.405
huber	Huber Regressor	7.400967e+02	7.983871e+05	8.111353e+02	-1.039700e+00	0.1529	1.363000e-01	0.043
en	Elastic Net	6.947059e+02	7.720155e+05	8.218678e+02	-4.965000e-01	0.1478	1.291000e-01	0.015
llar	Lasso Least Angle Regression	7.360759e+02	9.102712e+05	8.629876e+02	-1.166300e+00	0.1548	1.358000e-01	0.021
knn	K Neighbors Regressor	7.673833e+02	8.432312e+05	8.791020e+02	-1.152500e+00	0.1591	1.415000e-01	0.062
br	Bayesian Ridge	7.517336e+02	8.941649e+05	8.885432e+02	-6.728000e-01	0.1596	1.393000e-01	0.017
gbr	Gradient Boosting Regressor	8.095444e+02	8.157527e+05	8.886683e+02	-1.454800e+00	0.1546	1.432000e-01	0.042
lightgbm	Light Gradient Boosting Machine	7.529047e+02	8.778159e+05	8.939354e+02	-7.092000e-01	0.1610	1.392000e-01	0.019
dummy	Dummy Regressor	7.529047e+02	8.778159e+05	8.939354e+02	-7.092000e-01	0.1610	1.392000e-01	0.012
omp	Orthogonal Matching Pursuit	8.345117e+02	9.292953e+05	9.048262e+02	-1.592200e+00	0.1578	1.465000e-01	0.013
dt	Decision Tree Regressor	8.588333e+02	9.710548e+05	9.452119e+02	-2.568500e+00	0.1665	1.505000e-01	0.017
par	Passive Aggressive Regressor	1.349294e+03	2.316206e+06	1.426332e+03	-3.797200e+00	0.2687	2.517000e-01	0.015
lar	Least Angle Regression	3.242528e+19	1.059372e+40	3.255221e+19	-1.317463e+34	25.4641	5.982867e+15	0.023

Forecast

03

MODEL ACCURACY

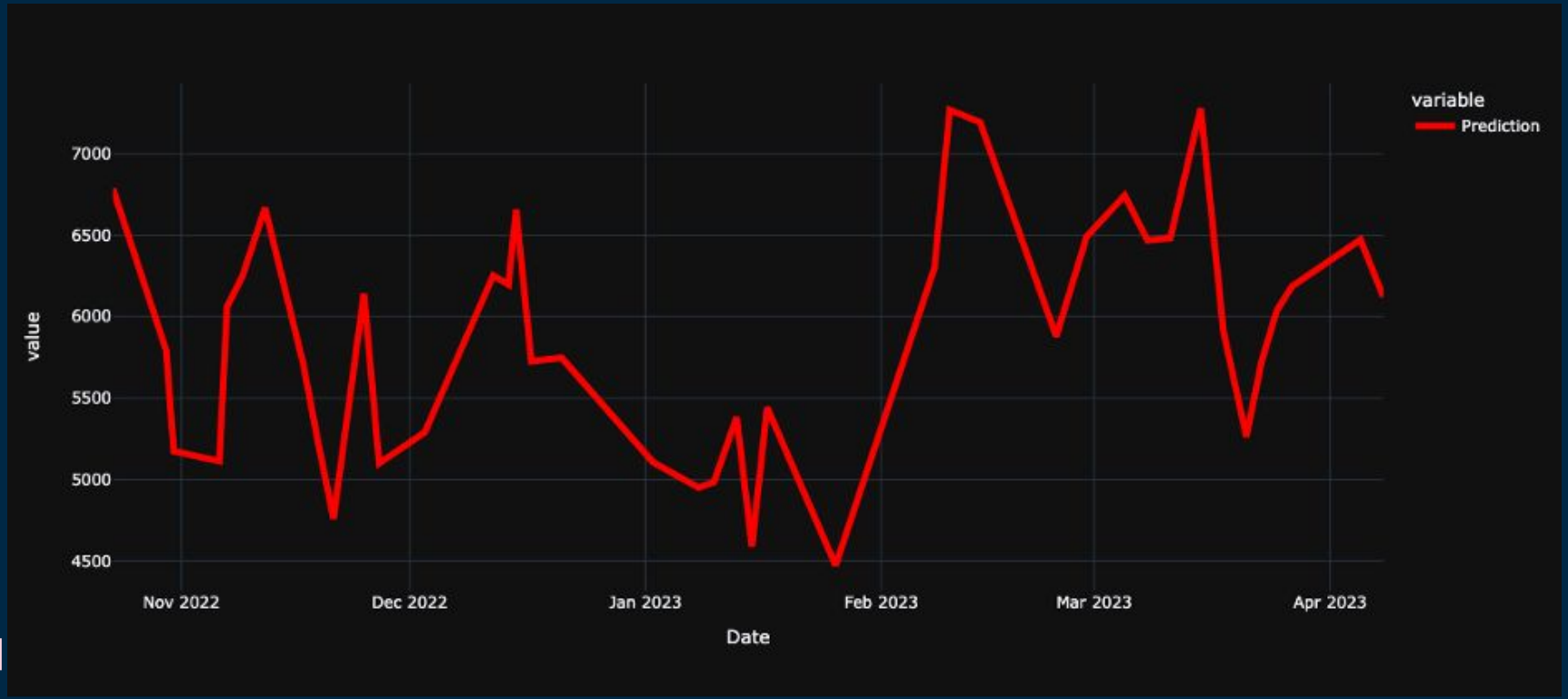


Model	MAE	MSE	RMSE	R2	RMSLE	MAPE
Ridge Regression	347.0587	194679.6313	441.2251	0.7521	0.0827	0.0649

FORECAST

Date	Start (ET)	Opponent	Vegas Over/Under 21/22	Label
2022-12-20	10:30p	San Antonio Spurs	28.5	5256.843262

2022-23 FORECAST



1*)[SQL]Do evening games (6:00pm or later) on average, have a higher attendance than afternoon games (Before 6:00pm)? Also, provide average attendance per game by month.*

```
   Avg_Seats      time
0       5008  12:30:00
1       6090  18:00:00
2       6216  18:30:00
3       6473  19:00:00
4       5687  19:30:00
   Avg_Attendance Month
0            4414    01
1            6619    02
2            6555    03
3            6177    04
4            5700    10
5            5655    11
6            5559    12
```

Based on this queried table, I found that there are strictly more people seated in evening games as opposed to the noon games. Moreover, February and March are the peak of the season when there are the most people attending.

2)[SQL]For all games, provide the average number of fans in arena 2hrs prior to start time, 1hr prior to start time, and at start time.

	Arrival	Avg_Attendance
0	2 Hours Early	4
1	1 Hour Early	705
2	At Start Time	3574
3	15 Minutes Over	4552
4	30 Minutes Over	5176
5	1 Hour Over	5613

Based on scanned tickets data, on average, there are 4 fans are seated 2 hours early, around 700 fans are seated 1 hour early, and nearly 3600 fans are seated by the start of each game.

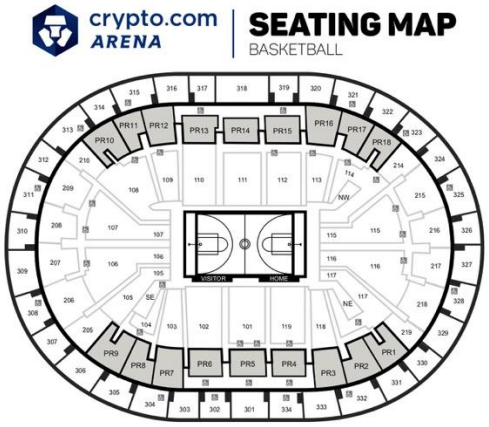
3)[SQL]Count the number of accounts with both tickets bought >=5 and scans >=3. Of these, how many are members?

	Membership	Accounts
0	None	1472
1	Full Season	1123
2	Half Season	1288

There are 3,883 accounts that have bought at least 5 tickets and scanned 3 times, and of which, 2,411 accounts are members with full season or half season passes.

4)Identify and rank the top 10 sections that are, on average, the most filled to their capacity.

	Section	Occupancy_Rate	Sales	Avg_Price
0	102	74	2.964164e+06	247.157881
1	106	74	1.488270e+06	97.012569
2	105	73	1.004143e+06	130.832923
3	107	73	1.412365e+06	99.715152
4	114	73	1.015874e+06	113.000413
5	113	72	7.799040e+05	146.131538
6	115	72	1.485474e+06	104.603474
7	109	71	7.284153e+05	142.658705
8	101	70	3.594127e+06	320.732351
9	112	69	2.659070e+06	242.926176



The most occupied section is section number 102 at an occupancy rate of about 74%. Going down the list of most occupied sections, there is 106, 105, 107, 114, 113, 115, 109, 101, and finally 112.

5)Is there a correlation between the opponent team having a higher Vegas over/under win total (coming into the 21/22 season) and higher attendance in the 21/22 season?

	Occupancy	Vegas Over/Under 21/22
Occupancy	1.000000	0.400282
Vegas Over/Under 21/22	0.400282	1.000000

Comparing the number of tickets scanned for a metric on those attending and the Vegas Over/Under values, I can see a moderately positive correlation between them.

6)Pull data from Basketball Reference around 21/22 game results. Is there a difference in quantity of tickets purchased and attendance between home games within the following seven days from the third game of a win streak(W3), vs other games? Optional: Include code for scraping Basketball Reference and pulling in data.

On the left, we can compare the mean number of tickets and those attending based on the win streak given after the game's result while the right graphs are based on the previously held win streak from the last game before the next game starts. The right graphs based on the last streak are more telling of the trend of greater sales and occupation during a win streak than a losing streak (not counting the single 'L 5' game). Looking at the bottom graph that compares sales and attendance for each game in the season, the gold bars represent the 3rd win in a streak. There does not seem to be any clear trend on a difference for tickets purchased and attendance based on the data; with more 3rd win games in the dataset, I believe I would be more confidence and able to provide a better answer to this question.

```
!pip3 install pycaret -q
!pip3 install Jinja2==3.1.2 -q
# Then restart kernel before moving on

# Import other required libraries to start
!pip3 install pandasql -q
from pandasql import sqldf
import pandas as pd
import numpy as np
from datetime import datetime
from datetime import timedelta
import re
import plotly.express as px
```

```
  Building wheel for pandasql (setup.py) ...
```

```
# Get names of the Excel sheets for reference
sheets = pd.ExcelFile('LAC Data.xlsx').sheet_names
print(sheets)
```

```
['Variable Dictionary', 'Team Data', 'Seating Chart', 'Customer
Package Type', 'Ticket Sales Data', 'Ticket Scan Data']
```

```
# Read in sheets as dataframes with Pandas
attendance = pd.read_excel('LAC Data.xlsx', sheet_name='Ticket Scan
Data')
attendance.scan_datetime = attendance.scan_datetime +
timedelta(hours=-2) # Convert CST to PST by subtracting 2 hours for
standardization
tickets = pd.read_excel('LAC Data.xlsx', sheet_name='Ticket Sales
Data')
customers = pd.read_excel('LAC Data.xlsx', sheet_name='Customer
Package Type')
seats = pd.read_excel('LAC Data.xlsx', sheet_name='Seating Chart')
teams = pd.read_excel('LAC Data.xlsx', sheet_name='Team Data')
```

```
attendance
```

```
                seat_unique_id  \
0        124908643:446:1:100:0
1        124908643:446:1:100:1
2       124908643:446:1:100:10
3       124908643:446:1:100:11
4       124908643:446:1:100:12
...                        ...
236465  124908643:487:1:699:16
236466  124908643:487:1:699:17
236467  124908643:487:1:699:21
236468  124908643:487:1:699:22
236469  124908643:462:1:614:18

                                                event_name
```

```
event_datetime  \
0          LA Clippers 2021-22 Game 2 - Portland 10/25/21 2021-10-25
19:30:00
1          LA Clippers 2021-22 Game 2 - Portland 10/25/21 2021-10-25
19:30:00
2          LA Clippers 2021-22 Game 2 - Portland 10/25/21 2021-10-25
19:30:00
3          LA Clippers 2021-22 Game 2 - Portland 10/25/21 2021-10-25
19:30:00
4          LA Clippers 2021-22 Game 2 - Portland 10/25/21 2021-10-25
19:30:00
...                                                      ...
...
236465  LA Clippers 2021-22 Game 41 - Oklahoma City 4/... 2022-04-10
18:30:00
236466  LA Clippers 2021-22 Game 41 - Oklahoma City 4/... 2022-04-10
18:30:00
236467  LA Clippers 2021-22 Game 41 - Oklahoma City 4/... 2022-04-10
18:30:00
236468  LA Clippers 2021-22 Game 41 - Oklahoma City 4/... 2022-04-10
18:30:00
236469      LA Clippers 2021-22 Game 33 - New York 3/6/22 2022-03-06
19:00:00

                    scan_gate       scan_datetime  section_name
row_name  \
0           Figueroa - Door 7 2021-10-25 21:39:19           101
6
1           Figueroa - Door 7 2021-10-25 21:40:05           101
6
2         Star Plaza - Door 4 2021-10-25 21:36:52           101
6
3        11th Street - Door 7 2021-10-25 20:45:20           101
6
4        11th Street - Door 7 2021-10-25 20:45:20           101
6
...                       ...                 ...           ...      .
..
236465    Star Plaza - Door 5 2022-04-10 19:11:36           218
11
236466    Star Plaza - Door 5 2022-04-10 19:11:36           218
11
236467      Figueroa - Door 5 2022-04-10 20:15:16           218
11
236468      Figueroa - Door 5 2022-04-10 20:15:15           218
11
236469  11th Street - Door 10 2022-03-06 21:39:03           208
10

        seat_number
```

```
0                21
1                20
2                11
3                10
4                 9
...              ...
236465            8
236466            7
236467            3
236468            2
236469            4

[236470 rows x 8 columns]
```

```
# Attendance by Event Time
query1 = '''
SELECT
  COUNT(seat_unique_id)/ COUNT(DISTINCT event_name) AS Avg_Attendance,

  strftime('%H:%M:%S', event_datetime) AS Time
FROM
  attendance
GROUP BY
  Time

'''


query2 = '''
SELECT
  COUNT(seat_unique_id)/ COUNT(DISTINCT event_name) AS Avg_Attendance,

  strftime('%m', event_datetime) AS Month
FROM
  attendance
GROUP BY
  Month

'''
print(sqldf(query1))
print(sqldf(query2))
```

```
   Avg_Seats      time
0       5008  12:30:00
1       6090  18:00:00
2       6216  18:30:00
3       6473  19:00:00
4       5687  19:30:00
   Avg_Attendance Month
0            4414    01
1            6619    02
```

```
2            6555    03
3            6177    04
4            5700    10
5            5655    11
6            5559    12

# Arrival Times for Attending / Seat Fill Speed
query = '''
SELECT '2 Hours Early'                                    AS Arrival,
       Count(seat_unique_id) / Count(DISTINCT event_name) AS
Avg_Attendance
FROM   attendance
WHERE  24 * ( Julianday(scan_datetime) - Julianday(event_datetime) )
<= -2
UNION ALL

SELECT '1 Hour Early'                                     AS Arrival,
       Count(seat_unique_id) / Count(DISTINCT event_name) AS
Avg_Attendance
FROM   attendance
WHERE  24 * ( Julianday(scan_datetime) - Julianday(event_datetime) )
<= -1
UNION ALL

SELECT 'At Start Time'                                    AS Arrival,
       Count(seat_unique_id) / Count(DISTINCT event_name) AS
Avg_Attendance
FROM   attendance
WHERE  24 * ( Julianday(scan_datetime) - Julianday(event_datetime) )
<= 0

UNION ALL

SELECT '15 Minutes Over'                                    AS
Arrival,
       Count(seat_unique_id) / Count(DISTINCT event_name) AS
Avg_Attendance
FROM   attendance
WHERE  24 * ( Julianday(scan_datetime) - Julianday(event_datetime) )
<= 0.25
UNION ALL

SELECT '30 Minutes Over'                                    AS
Arrival,
       Count(seat_unique_id) / Count(DISTINCT event_name) AS
Avg_Attendance
FROM   attendance
WHERE  24 * ( Julianday(scan_datetime) - Julianday(event_datetime) )
<= 0.5
UNION ALL
```

```
SELECT '1 Hour Over'                                    AS Arrival,
       Count(seat_unique_id) / Count(DISTINCT event_name) AS
Avg_Attendance
FROM   attendance
WHERE  24 * ( Julianday(scan_datetime) - Julianday(event_datetime) )
<= 1
'''
sqldf(query)
```

```
           Arrival  Avg_Attendance
0    2 Hours Early               4
1     1 Hour Early             705
2    At Start Time            3574
3  15 Minutes Over            4552
4  30 Minutes Over            5176
5      1 Hour Over            5613
```

```
# 'Loyal' Member Count
query = '''
SELECT
  customer_tickets.Membership,
  COUNT(customer_tickets.Accounts) AS Accounts
FROM
  (
    SELECT
      tickets.customer_unique_id AS Accounts,
      COUNT(tickets.seat_unique_id) AS Tickets,
      COUNT(attendance.seat_unique_id) AS Scans,
      customers.price_code_type AS Membership
    FROM
      tickets
      LEFT JOIN attendance ON attendance.seat_unique_id =
tickets.seat_unique_id
      LEFT JOIN customers ON customers.customer_unique_id =
tickets.customer_unique_id
    GROUP BY
      Accounts
    HAVING
      Tickets >= 5
      AND Scans >= 3
  ) AS customer_tickets
GROUP BY
  customer_tickets.Membership
'''
sqldf(query)
```

```
    Membership  Accounts
0         None      1472
1  Full Season      1123
2  Half Season      1288
```

```
# 10 Most Popular Sections
query = '''
SELECT
  attendance.section_name AS Section,
  100 * COUNT(attendance.seat_unique_id)/ (
    SELECT
      SUM(seat_count) AS capacity
    FROM
      seats
    WHERE
      seats.section_name = attendance.section_name
  ) / Count(DISTINCT attendance.event_name) AS Occupancy_Rate,
  SUM(tickets.price) AS Sales,
  AVG(tickets.price) AS Avg_Price
FROM
  attendance
  LEFT JOIN tickets ON attendance.seat_unique_id =
tickets.seat_unique_id
GROUP BY
  attendance.section_name
ORDER BY
  Occupancy_Rate DESC
LIMIT
  10
'''
sqldf(query)
```

	Section	Occupancy_Rate	Sales	Avg_Price
0	102	74	2.964164e+06	247.157881
1	106	74	1.488270e+06	97.012569
2	105	73	1.004143e+06	130.832923
3	107	73	1.412365e+06	99.715152
4	114	73	1.015874e+06	113.000413
5	113	72	7.799040e+05	146.131538
6	115	72	1.485474e+06	104.603474
7	109	71	7.284153e+05	142.658705
8	101	70	3.594127e+06	320.732351
9	112	69	2.659070e+06	242.926176

```
# Occupancy vs. Vegas Over/Under 21-22 Correlation Table
query = '''
SELECT
  event_name AS Event,
  COUNT(attendance.seat_unique_id) AS Occupancy
FROM
  attendance
GROUP BY
  Event
'''
attendance_df = sqldf(query)
```

```
pattern = r'[0-9]+'
# Remove date string from event/city name and clean data to match City
column for join
attendance_df['City'] = attendance_df['Event'].apply(lambda row:
re.sub(pattern,'',' '.join(str(row).split()[6:8])).replace(' //',
'').replace('//', ''))
attendance_df['Date'] = attendance_df['Event'].apply(lambda row:
str(row.split(' ')[-1]).replace('Orleans', ''))
attendance_df['Date'] = pd.to_datetime(attendance_df['Date']) #
Normalize date formatting
teams = teams.drop(12) # Remove LA Clippers team from list of
opponents
teams['City'] = teams['City'].apply(lambda row: row.replace('San
Francisco','Golden State')) # Fix San Francisco /Golden State city
name mismatch
attendance_df.merge(teams[['City', 'Vegas Over/Under 21/22']],
'inner')
```

	Event	Occupancy	\
0	LA Clippers 2021-22 Game 1 - Memphis 10/23/21	6498	
1	LA Clippers 2021-22 Game 24 - Memphis 1/8/22	4850	
2	LA Clippers 2021-22 Game 10 - San Antonio 11/1...	4928	
3	LA Clippers 2021-22 Game 20 - San Antonio 12/2...	5778	
4	LA Clippers 2021-22 Game 11 - Dallas 11/21/21	5085	
5	LA Clippers 2021-22 Game 12 - Dallas 11/23/21	5903	
6	LA Clippers 2021-22 Game 13 - Detroit 11/26/21	5210	
7	LA Clippers 2021-22 Game 14 - Golden State 11/...	6832	
8	LA Clippers 2021-22 Game 30 - Golden State 2/1...	7311	
9	LA Clippers 2021-22 Game 15 - New Orleans11/29/21	4257	
10	LA Clippers 2021-22 Game 38 - New Orleans 4/3/22	6164	
11	LA Clippers 2021-22 Game 16 - Sacramento 12/1/21	4558	
12	LA Clippers 2021-22 Game 40 - Sacramento 4/9/22	5942	
13	LA Clippers 2021-22 Game 17 - Boston 12/8/21	6179	
14	LA Clippers 2021-22 Game 18 - Orlando 12/11/21	4814	
15	LA Clippers 2021-22 Game 19 - Phoenix 12/13/21	6190	
16	LA Clippers 2021-22 Game 39 - Phoenix 4/6/22	6631	
17	LA Clippers 2021-22 Game 2 - Portland 10/25/21	5738	
18	LA Clippers 2021-22 Game 6 - Portland 11/9/21	5531	
19	LA Clippers 2021-22 Game 21 - Denver 12/26/21	5566	
20	LA Clippers 2021-22 Game 26 - Denver 1/11/22	4369	
21	LA Clippers 2021-22 Game 22 - Brooklyn 12/27/21	5829	
22	LA Clippers 2021-22 Game 23 - Minnesota 1/3/22	4587	
23	LA Clippers 2021-22 Game 8 - Minnesota 11/13/21	6045	
24	LA Clippers 2021-22 Game 25 - Atlanta 1/9/22	4053	
25	LA Clippers 2021-22 Game 27 - Indiana 1/17/22	4212	
26	LA Clippers 2021-22 Game 28 - Los Angeles 2/3/22	6913	
27	LA Clippers 2021-22 Game 32 - Los Angeles 3/3/22	7376	
28	LA Clippers 2021-22 Game 29 - Milwaukee 2/6/22	6403	
29	LA Clippers 2021-22 Game 3 - Cleveland 10/27/21	4866	

30	LA Clippers 2021-22 Game 31 - Houston 2/17/22	5849
31	LA Clippers 2021-22 Game 33 - New York 3/6/22	6198
32	LA Clippers 2021-22 Game 34 - Washington 3/9/22	5629
33	LA Clippers 2021-22 Game 35 - Toronto 3/16/22	6774
34	LA Clippers 2021-22 Game 36 - Philadelphia 3/2...	7162
35	LA Clippers 2021-22 Game 37 - Utah 3/29/22	6192
36	LA Clippers 2021-22 Game 4 - Oklahoma City 11/...	4784
37	LA Clippers 2021-22 Game 41 - Oklahoma City 4/...	5972
38	LA Clippers 2021-22 Game 5 - Charlotte 11/7/21	6303
39	LA Clippers 2021-22 Game 7 - Miami 11/11/21	6200
40	LA Clippers 2021-22 Game 9 - Chicago 11/14/21	6789

	City	Date	Vegas Over/Under 21/22
0	Memphis	2021-10-23	41.5
1	Memphis	2022-01-08	41.5
2	San Antonio	2021-11-16	28.5
3	San Antonio	2021-12-20	28.5
4	Dallas	2021-11-21	48.5
5	Dallas	2021-11-23	48.5
6	Detroit	2021-11-26	24.5
7	Golden State	2021-11-28	48.5
8	Golden State	2022-02-14	48.5
9	New Orleans	2021-11-29	39.5
10	New Orleans	2022-04-03	39.5
11	Sacramento	2021-12-01	36.5
12	Sacramento	2022-04-09	36.5
13	Boston	2021-12-08	45.5
14	Orlando	2021-12-11	22.5
15	Phoenix	2021-12-13	51.5
16	Phoenix	2022-04-06	51.5
17	Portland	2021-10-25	44.5
18	Portland	2021-11-09	44.5
19	Denver	2021-12-26	47.5
20	Denver	2022-01-11	47.5
21	Brooklyn	2021-12-27	56.5
22	Minnesota	2022-01-03	35.5
23	Minnesota	2021-11-13	35.5
24	Atlanta	2022-01-09	47.5
25	Indiana	2022-01-17	42.5
26	Los Angeles	2022-02-03	52.5
27	Los Angeles	2022-03-03	52.5
28	Milwaukee	2022-02-06	54.5
29	Cleveland	2021-10-27	26.5
30	Houston	2022-02-17	27.5
31	New York	2022-03-06	41.5
32	Washington	2022-03-09	33.5
33	Toronto	2022-03-16	35.5
34	Philadelphia	2022-03-25	50.5
35	Utah	2022-03-29	52.5
36	Oklahoma City	2021-11-01	23.5

```
37  Oklahoma City 2022-04-10                    23.5
38      Charlotte 2021-11-07                    38.5
39          Miami 2021-11-11                    48.5
40        Chicago 2021-11-14                    42.5
```

```
# Correlation Graph
attendance_df.merge(teams[['City', 'Vegas Over/Under 21/22']],
'inner').corr().style.background_gradient(cmap='YlOrRd')
```

```
<pandas.io.formats.style.Styler at 0x7f87b55bbd50>
```

```
# Find valuable metrics per game by price code groupings
query = '''
SELECT
  price_code_type,
  100*COUNT(attendance.event_name)/COUNT(tickets.event_name) AS
Attend_Rate,
  COUNT(DISTINCT tickets.customer_unique_id) AS Accounts,
  COUNT(DISTINCT tickets.seat_unique_id)/COUNT(DISTINCT
tickets.event_name) AS AVG_Seats,
  printf("%.2f",100*COUNT(DISTINCT tickets.seat_unique_id)/
COUNT(DISTINCT tickets.customer_unique_id)/COUNT(DISTINCT
tickets.event_name))/100 AS AVG_Tickets_Bought,
  printf("%.2f",100*COUNT(DISTINCT attendance.seat_unique_id)/
COUNT(DISTINCT tickets.customer_unique_id)/COUNT(DISTINCT
tickets.event_name))/100 AS AVG_Tickets_Scanned,
  printf("%.2f",COUNT(DISTINCT tickets.event_name)*100*COUNT(DISTINCT
attendance.seat_unique_id)/ COUNT(DISTINCT
tickets.customer_unique_id)/COUNT(DISTINCT tickets.event_name))/100 AS
Tot_AVG_Scans
FROM
  tickets
  LEFT JOIN attendance ON attendance.seat_unique_id =
tickets.seat_unique_id
GROUP BY
  price_code_type

'''
```

```
sqldf(query)
```

```
  price_code_type  Attend_Rate  Accounts  AVG_Seats
AVG_Tickets_Bought  \
0            Comp           64      3113       1512
0.48
1     Full Season           75      1149       2974
2.58
2           Group           88       249        286
1.15
3     Half Season           73      1364       1373
1.00
```

```
4    Partial Plan           79       332        120
0.36
5     Single Game           93      2393       1261
0.52

   AVG_Tickets_Scanned  Tot_AVG_Scans
0                 0.31          12.92
1                 1.96          80.50
2                 1.02          39.84
3                 0.74          30.53
4                 0.28          11.81
5                 0.49          20.16
```

```
# Repeat data extraction and transformation for ticket sales by game
query = '''
SELECT
  event_name AS Event,
  COUNT(tickets.seat_unique_id) AS Sales
FROM
  tickets
GROUP BY
  Event
'''
sales_df = sqldf(query)
import re

pattern = r'[0-9]+'
sales_df['City'] = sales_df['Event'].apply(lambda row:
re.sub(pattern,'',' '.join(str(row).split()[6:8])).replace(' //',
'').replace('//', ''))
sales_df['Date'] = sales_df['Event'].apply(lambda row:
str(row.split(' ')[-1]).replace('Orleans', ''))
sales_df['Date'] = pd.to_datetime(sales_df['Date']) # Normalize date
formatting
sales_df = sales_df.drop(['Event','City'], axis=1) # Remove duplicate
column when joining
# sales_df.merge(teams[['City', 'Vegas Over/Under 21/22']],
'inner').corr().style.background_gradient(cmap='YlOrRd')

# Repeat data extraction and transformation for ticket sales by date
query = '''
SELECT
  sales_datetime AS Date,
  COUNT(tickets.seat_unique_id) AS Sales
FROM
  tickets
GROUP BY
  Date
'''
tickets_df = sqldf(query)
```

```
tickets_df['Date'] = pd.to_datetime(tickets_df['Date']).dt.normalize()
# Normalize date formatting
tickets_df = tickets_df.groupby('Date', as_index=False).sum()

# Create an URL object of subject to scrape
url = 'https://www.basketball-reference.com/teams/LAC/2022_games.html'
# Scrape HTML table with Pandas
results = pd.read_html(url)[0]
results
```

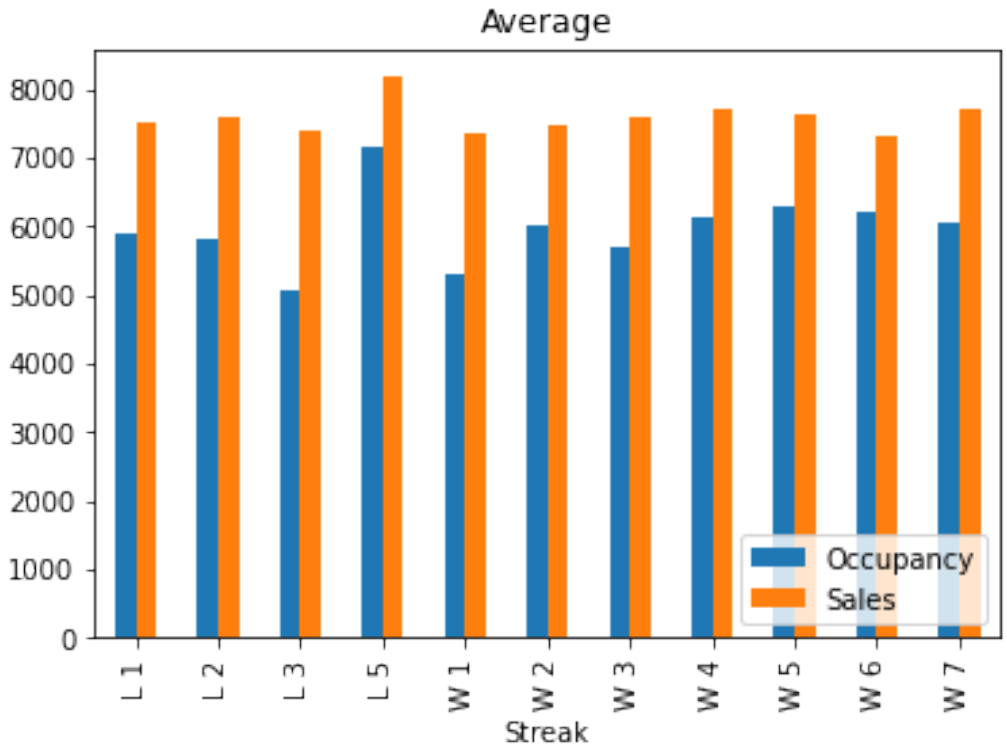
```
     G               Date  Start (ET)  Unnamed: 3 Unnamed: 4 Unnamed:
5  \
0    1  Thu, Oct 21, 2021      10:00p         NaN  Box Score
@
1    2  Sat, Oct 23, 2021      10:30p         NaN  Box Score
NaN
2    3  Mon, Oct 25, 2021      10:30p         NaN  Box Score
NaN
3    4  Wed, Oct 27, 2021      10:30p         NaN  Box Score
NaN
4    5  Fri, Oct 29, 2021      10:00p         NaN  Box Score
@
..  ..                ...         ...         ...        ...        ..
.
83   G               Date  Start (ET)         NaN        NaN
NaN
84  81   Sat, Apr 9, 2022       9:30p         NaN  Box Score
NaN
85  82  Sun, Apr 10, 2022       9:30p         NaN  Box Score
NaN
86  83  Tue, Apr 12, 2022       9:30p         NaN  Box Score
@
87  84  Fri, Apr 15, 2022      10:00p         NaN  Box Score
NaN

                  Opponent Unnamed: 7 Unnamed: 8   Tm  Opp   W   L
Streak  \
0    Golden State Warriors          L        NaN  113  115   0   1
L 1
1        Memphis Grizzlies          L        NaN  114  120   0   2
L 2
2   Portland Trail Blazers          W        NaN  116   86   1   2
W 1
3      Cleveland Cavaliers          L        NaN   79   92   1   3
L 1
4   Portland Trail Blazers          L        NaN   92  111   1   4
L 2
..                    ...        ...        ...  ...  ...  ..  ..
...
83                Opponent        NaN        NaN   Tm  Opp   W   L
```

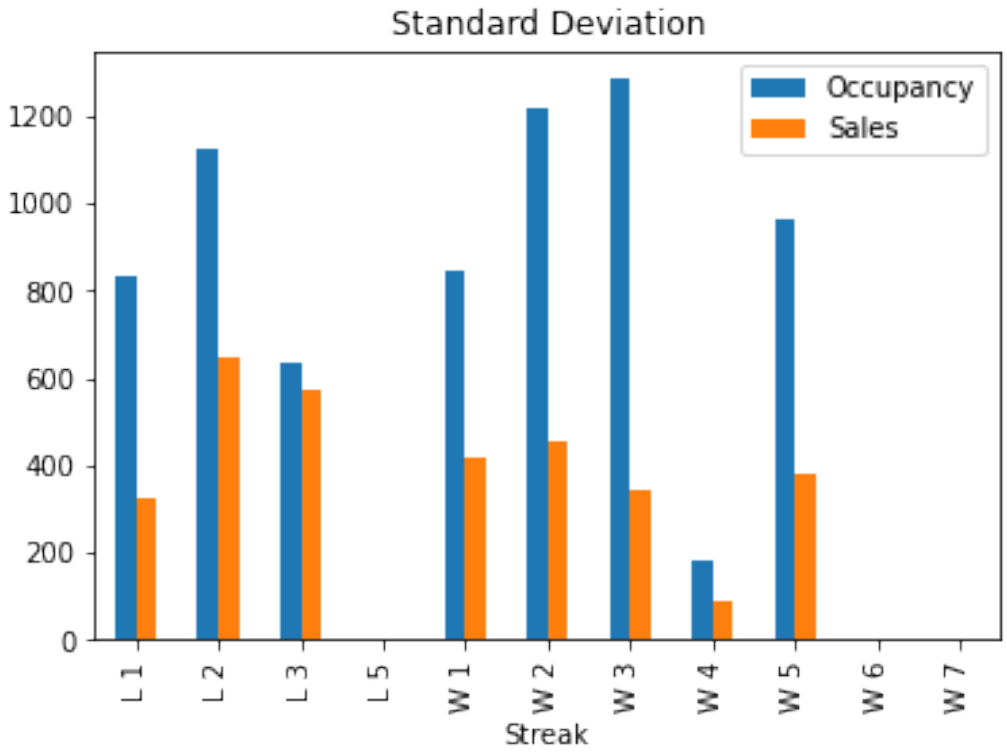
```
Streak
84        Sacramento Kings          W        NaN  117   98  41  40
W 4
85   Oklahoma City Thunder          W        NaN  138   88  42  40
W 5
86  Minnesota Timberwolves          L        NaN  104  109  42  41
L 1
87    New Orleans Pelicans          L        NaN  101  105  42  42
L 2

           Notes
0            NaN
1            NaN
2            NaN
3            NaN
4            NaN
..           ...
83         Notes
84           NaN
85           NaN
86  Play-In Game
87  Play-In Game

[88 rows x 15 columns]
```

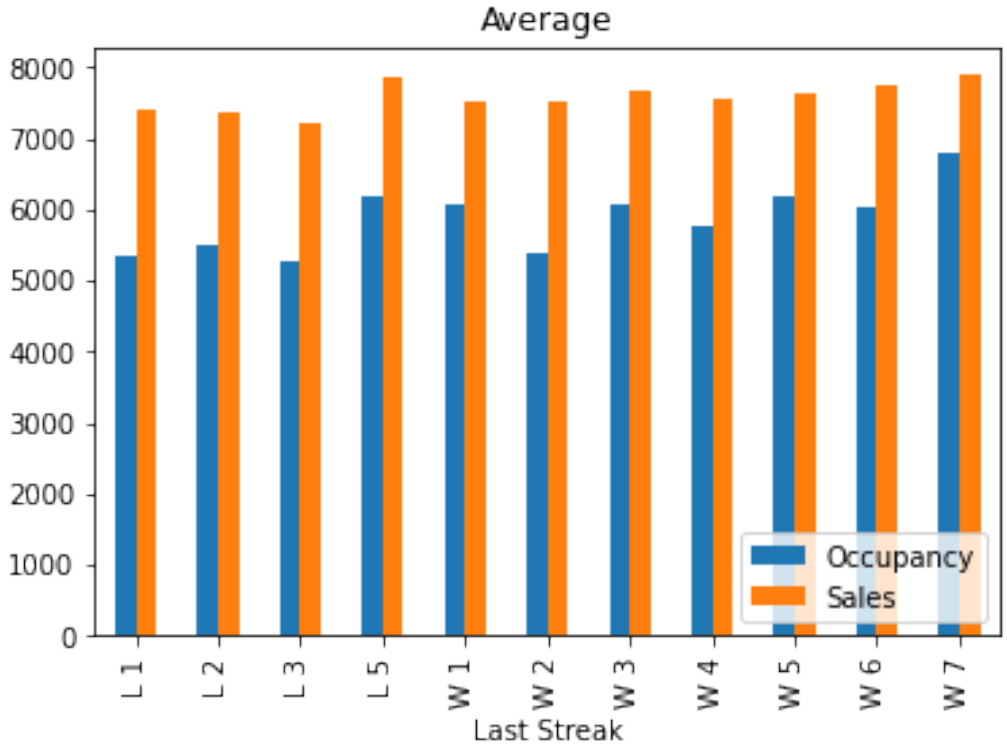
```
# Clean scraped dataframe
results = results.dropna(axis='rows',
thresh=11).dropna(axis='columns') # Remove null rows and columns
results['Date'] = pd.to_datetime(results['Date']) # Format date to
match standardization
results['Date'] = results['Date'].dt.strftime('%D').apply(lambda row:
datetime.strptime(row, '%m/%d/%y').date()) # Change text to date
results['Date'] = pd.to_datetime(results['Date']) # Reformat date to
match standardization
results_df = results.merge(attendance_df,'right',
on='Date').merge(sales_df,'right',
on='Date').sort_values('Date')#.drop('Unnamed: 4', axis=1)

# Compare Games by Streak
results_df.groupby('Streak')[['Occupancy',
'Sales']].mean().plot.bar(title="Average").legend(loc='lower right')
results_df.groupby('Streak')[['Occupancy',
'Sales']].std().plot.bar(title="Standard Deviation").legend(loc='upper
right')

# Compare Games by Last Held Streak from Prior Game
results_df['Last Streak'] =
results_df.sort_values('Date').Streak.shift(1)
results_df.groupby('Last Streak')[['Occupancy',
'Sales']].mean().plot.bar(title="Average").legend(loc='lower right')
```

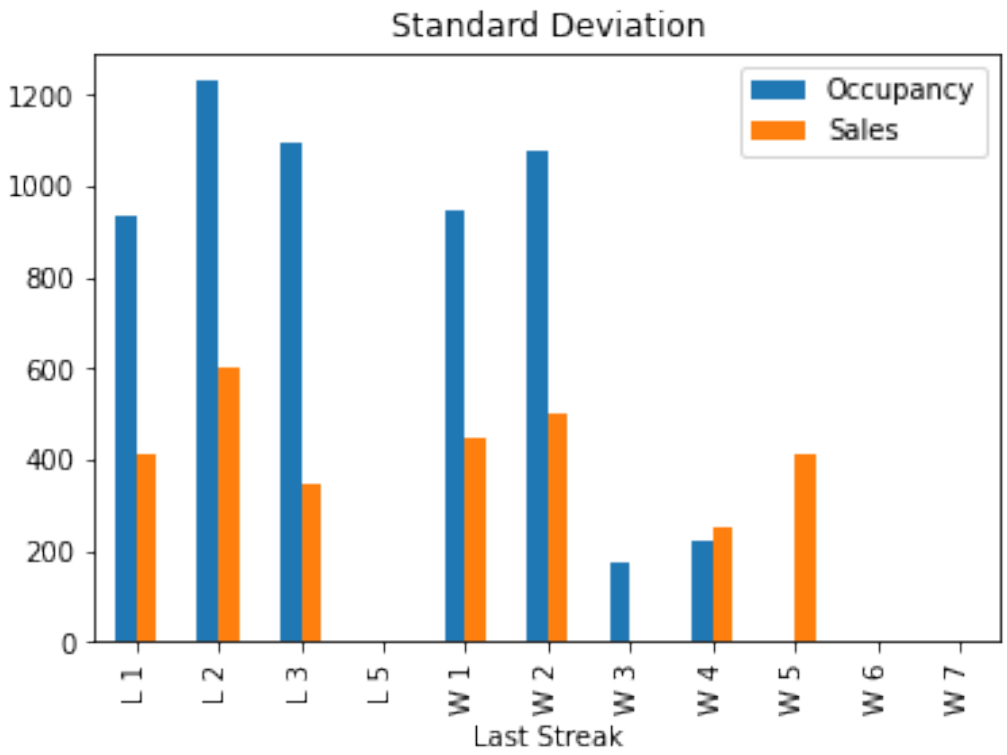
```
results_df.groupby('Last Streak')[['Occupancy',
'Sales']].std().plot.bar(title="Standard Deviation").legend(loc='upper
right')
```

```
<matplotlib.legend.Legend at 0x7f335799f150>
```




Standard Deviation
Occupancy
Sales
1200
1000
800
600
400
200
0
L 1
L 2
L 3
L 5
W 1
W 2
W 3
W 4
W 5
W 6
W 7
Streak


Average
8000
7000
6000
5000
4000
3000
2000
1000
0
Occupancy
Sales
L 1
L 2
L 3
L 5
W 1
W 2
W 3
W 4
W 5
W 6
W 7
Last Streak




```
results_df

     G       Date Start (ET) Unnamed: 4                Opponent
Unnamed: 7  \
0    2 2021-10-23     10:30p  Box Score       Memphis Grizzlies
L
11   3 2021-10-25     10:30p  Box Score  Portland Trail Blazers
W
22   4 2021-10-27     10:30p  Box Score     Cleveland Cavaliers
L
33   6 2021-11-01     10:30p  Box Score   Oklahoma City Thunder
W
36   9 2021-11-07      9:00p  Box Score       Charlotte Hornets
W
37  10 2021-11-09     10:00p  Box Score  Portland Trail Blazers
W
38  11 2021-11-11     10:30p  Box Score              Miami Heat
W
39  12 2021-11-13     10:30p  Box Score  Minnesota Timberwolves
W
40  13 2021-11-14      9:30p  Box Score           Chicago Bulls
L
1   14 2021-11-16     10:30p  Box Score       San Antonio Spurs
W
2   17 2021-11-21      3:30p  Box Score        Dallas Mavericks
W
3   18 2021-11-23     10:30p  Box Score        Dallas Mavericks
```

L						
4	19	2021-11-26	3:30p	Box Score	Detroit Pistons	
W						
5	20	2021-11-28	3:30p	Box Score	Golden State Warriors	
L						
6	21	2021-11-29	10:30p	Box Score	New Orleans Pelicans	
L						
7	22	2021-12-01	10:30p	Box Score	Sacramento Kings	
L						
8	26	2021-12-08	10:30p	Box Score	Boston Celtics	
W						
9	27	2021-12-11	3:30p	Box Score	Orlando Magic	
W						
10	28	2021-12-13	10:30p	Box Score	Phoenix Suns	
W						
12	31	2021-12-20	10:30p	Box Score	San Antonio Spurs	
L						
13	33	2021-12-26	9:00p	Box Score	Denver Nuggets	
L						
14	34	2021-12-27	10:30p	Box Score	Brooklyn Nets	
L						
15	38	2022-01-03	10:30p	Box Score	Minnesota Timberwolves	
L						
16	40	2022-01-08	3:30p	Box Score	Memphis Grizzlies	
L						
17	41	2022-01-09	3:30p	Box Score	Atlanta Hawks	
W						
18	42	2022-01-11	10:30p	Box Score	Denver Nuggets	
W						
19	45	2022-01-17	3:30p	Box Score	Indiana Pacers	
W						
20	54	2022-02-03	10:00p	Box Score	Los Angeles Lakers	
W						
21	55	2022-02-06	9:00p	Box Score	Milwaukee Bucks	
L						
23	59	2022-02-14	10:30p	Box Score	Golden State Warriors	
W						
24	61	2022-02-17	10:30p	Box Score	Houston Rockets	
W						
25	65	2022-03-03	10:00p	Box Score	Los Angeles Lakers	
W						
26	66	2022-03-06	10:00p	Box Score	New York Knicks	
L						
27	68	2022-03-09	10:30p	Box Score	Washington Wizards	
W						
28	72	2022-03-16	10:30p	Box Score	Toronto Raptors	
L						
29	75	2022-03-25	10:30p	Box Score	Philadelphia 76ers	
L						
30	76	2022-03-29	10:00p	Box Score	Utah Jazz	

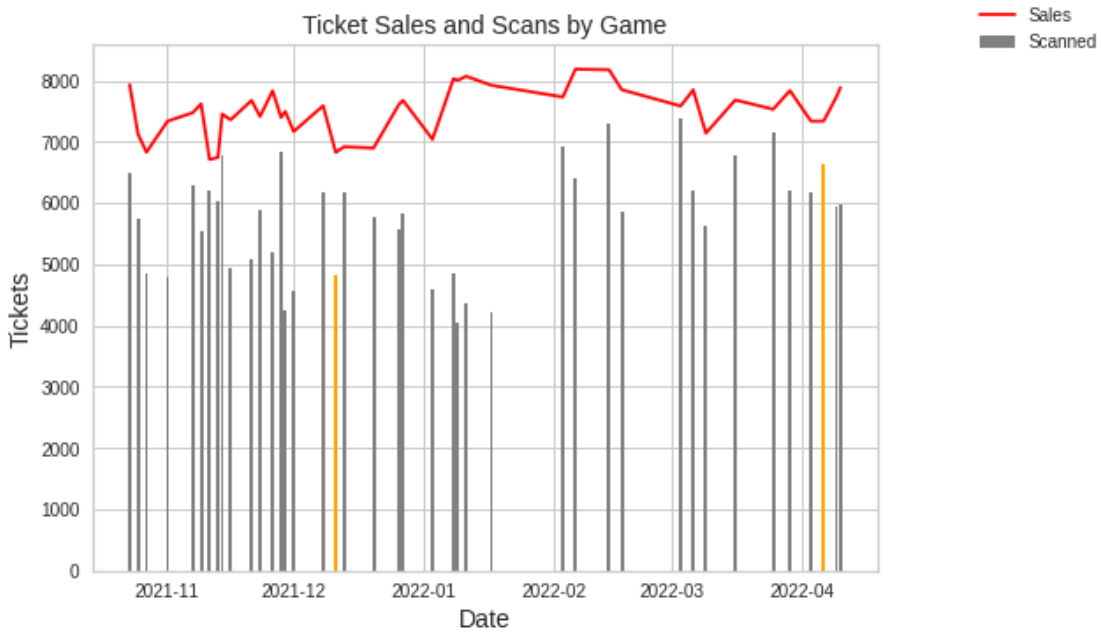
```
W
31  79 2022-04-03      9:30p  Box Score    New Orleans Pelicans
W
32  80 2022-04-06     10:00p  Box Score            Phoenix Suns
W
34  81 2022-04-09      9:30p  Box Score        Sacramento Kings
W
35  82 2022-04-10      9:30p  Box Score   Oklahoma City Thunder
W

     Tm  Opp   W   L Streak  \
0   114  120   0   2    L 2
11  116   86   1   2    W 1
22   79   92   1   3    L 1
33   99   94   2   4    W 1
36  120  106   5   4    W 4
37  117  109   6   4    W 5
38  112  109   7   4    W 6
39  129  102   8   4    W 7
40   90  100   8   5    L 1
1   106   92   9   5    W 1
2    97   91  10   7    W 1
3   104  112  10   8    L 1
4   107   96  11   8    W 1
5    90  105  11   9    L 1
6   104  123  11  10    L 2
7   115  124  11  11    L 3
8   114  111  14  12    W 2
9   106  104  15  12    W 3
10  111   95  16  12    W 4
12   92  116  16  15    L 3
13  100  103  17  16    L 1
14  108  124  17  17    L 2
15  104  122  19  19    L 1
16  108  123  19  21    L 3
17  106   93  20  21    W 1
18   87   85  21  21    W 2
19  139  133  22  23    W 1
20  111  110  27  27    W 1
21  113  137  27  28    L 1
23  119  104  29  30    W 2
24  142  111  30  31    W 1
25  132  111  34  31    W 5
26   93  116  34  32    L 1
27  115  109  35  33    W 1
28  100  103  36  36    L 2
29   97  122  36  39    L 5
30  121  115  37  39    W 1
31  119  100  39  40    W 2
32  113  109  40  40    W 3
```

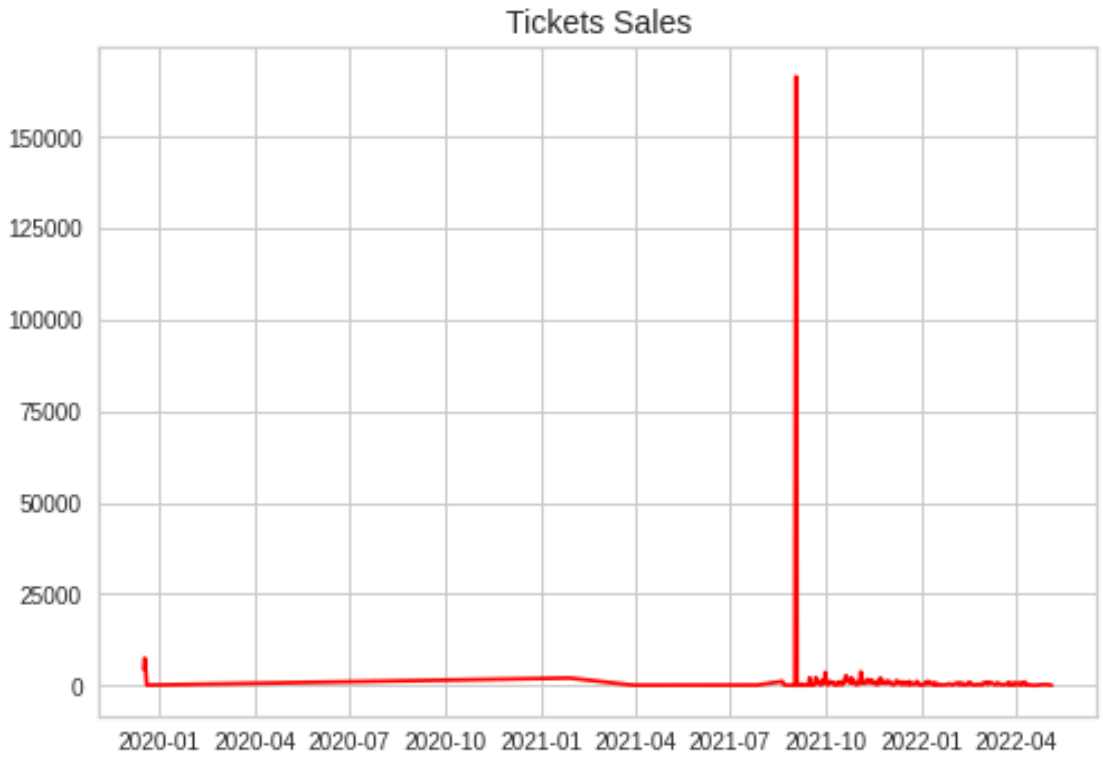
```
34  117   98  41  40    W 4
35  138   88  42  40    W 5

                                                  Event  Occupancy  \
0       LA Clippers 2021-22 Game 1 - Memphis 10/23/21       6498
11     LA Clippers 2021-22 Game 2 - Portland 10/25/21       5738
22    LA Clippers 2021-22 Game 3 - Cleveland 10/27/21       4866
33  LA Clippers 2021-22 Game 4 - Oklahoma City 11/...       4784
36     LA Clippers 2021-22 Game 5 - Charlotte 11/7/21       6303
37      LA Clippers 2021-22 Game 6 - Portland 11/9/21       5531
38        LA Clippers 2021-22 Game 7 - Miami 11/11/21       6200
39    LA Clippers 2021-22 Game 8 - Minnesota 11/13/21       6045
40      LA Clippers 2021-22 Game 9 - Chicago 11/14/21       6789
1   LA Clippers 2021-22 Game 10 - San Antonio 11/1...       4928
2       LA Clippers 2021-22 Game 11 - Dallas 11/21/21       5085
3       LA Clippers 2021-22 Game 12 - Dallas 11/23/21       5903
4      LA Clippers 2021-22 Game 13 - Detroit 11/26/21       5210
5   LA Clippers 2021-22 Game 14 - Golden State 11/...       6832
6   LA Clippers 2021-22 Game 15 - New Orleans11/29/21       4257
7    LA Clippers 2021-22 Game 16 - Sacramento 12/1/21       4558
8        LA Clippers 2021-22 Game 17 - Boston 12/8/21       6179
9      LA Clippers 2021-22 Game 18 - Orlando 12/11/21       4814
10     LA Clippers 2021-22 Game 19 - Phoenix 12/13/21       6190
12  LA Clippers 2021-22 Game 20 - San Antonio 12/2...       5778
13      LA Clippers 2021-22 Game 21 - Denver 12/26/21       5566
14    LA Clippers 2021-22 Game 22 - Brooklyn 12/27/21       5829
15     LA Clippers 2021-22 Game 23 - Minnesota 1/3/22       4587
16       LA Clippers 2021-22 Game 24 - Memphis 1/8/22       4850
17       LA Clippers 2021-22 Game 25 - Atlanta 1/9/22       4053
18       LA Clippers 2021-22 Game 26 - Denver 1/11/22       4369
19      LA Clippers 2021-22 Game 27 - Indiana 1/17/22       4212
20   LA Clippers 2021-22 Game 28 - Los Angeles 2/3/22       6913
21     LA Clippers 2021-22 Game 29 - Milwaukee 2/6/22       6403
23  LA Clippers 2021-22 Game 30 - Golden State 2/1...       7311
24      LA Clippers 2021-22 Game 31 - Houston 2/17/22       5849
25   LA Clippers 2021-22 Game 32 - Los Angeles 3/3/22       7376
26      LA Clippers 2021-22 Game 33 - New York 3/6/22       6198
27    LA Clippers 2021-22 Game 34 - Washington 3/9/22       5629
28      LA Clippers 2021-22 Game 35 - Toronto 3/16/22       6774
29  LA Clippers 2021-22 Game 36 - Philadelphia 3/2...       7162
30        LA Clippers 2021-22 Game 37 - Utah 3/29/22       6192
31   LA Clippers 2021-22 Game 38 - New Orleans 4/3/22       6164
32       LA Clippers 2021-22 Game 39 - Phoenix 4/6/22       6631
34    LA Clippers 2021-22 Game 40 - Sacramento 4/9/22       5942
35  LA Clippers 2021-22 Game 41 - Oklahoma City 4/...       5972

             City  Sales Last Streak
0         Memphis   7934         NaN
11       Portland   7417         L 2
22      Cleveland   7044         W 1
```

```
33  Oklahoma City   7142         L 1
36      Charlotte   7838         W 1
37       Portland   7339         W 4
38          Miami   7340         W 5
39      Minnesota   7731         W 6
40        Chicago   7888         W 7
1     San Antonio   7126         L 1
2          Dallas   6832         W 1
3          Dallas   7339         W 1
4         Detroit   7477         L 1
5    Golden State   7622         W 1
6     New Orleans   6713         L 1
7      Sacramento   6747         L 2
8          Boston   7456         L 3
9         Orlando   7361         W 2
10        Phoenix   7678         W 3
12    San Antonio   7835         W 4
13         Denver   7399         L 3
14       Brooklyn   7497         L 1
15      Minnesota   7170         L 2
16        Memphis   7591         L 1
17        Atlanta   6830         L 3
18         Denver   6921         W 1
19        Indiana   6901         W 2
20    Los Angeles   7610         W 1
21      Milwaukee   7678         W 1
23   Golden State   8033         L 1
24        Houston   8006         W 2
25    Los Angeles   8073         W 1
26       New York   7925         W 5
27     Washington   7734         L 1
28        Toronto   8189         W 1
29   Philadelphia   8178         L 2
30           Utah   7853         L 5
31    New Orleans   7586         W 1
32        Phoenix   7850         W 2
34     Sacramento   7685         W 3
35  Oklahoma City   7534         W 4
```

```
# Graph Comparison of Sales versus Scans
import matplotlib.pyplot as plt

plt.plot(sales_df['Date'].sort_values(), sales_df['Sales'],
color='red', label='Sales')
plt.title('Ticket Sales and Scans by Game', fontsize=14)
plt.xlabel('Date', fontsize=14)
plt.ylabel('Tickets', fontsize=14)

plt.bar(results_df['Date'].sort_values(), results_df['Occupancy'],
color=results_df['Streak'].apply(lambda x: 'orange' if x=="W 3" else
```

```
'grey'), label='Scanned')
plt.grid(True, axis='y')
plt.legend(loc='best', bbox_to_anchor=(1.3, 1.1))
# Highlight date of 3rd winstreak (to include non-home games)
# for date in results[results['Streak']=="W 3"].Date:
#   plt.vlines(date, 7000, 8000, "orange")
plt.show()

# Graph of Ticket Sales over Time
plt.plot(tickets_df['Date'].sort_values(), tickets_df['Sales'],
color='red', label='Sales')
plt.title('Tickets Sales', fontsize=14)
plt.show()
```






```
# Import Quick EDA Report Library (to check for any more interesting
metrics to look for)
# !pip3 install --upgrade pip
# !pip3 install pandas-profiling

from pandas_profiling import ProfileReport
# Create Pandas Profiling Report
profile = ProfileReport(attendance_df, axis=1))
profile.to_file('tickets.html')
```


```
{"version_major":2,"version_minor":0,"model_id":"67d82131840341c688d9c
ebb6482e2e3"}

{"version_major":2,"version_minor":0,"model_id":"3919fbc133894b1b889b2
53d897744b4"}

{"version_major":2,"version_minor":0,"model_id":"d4c24539a277485a901c6
2f4781e6647"}

{"version_major":2,"version_minor":0,"model_id":"ca434fda9d1147eda48c1
16d480a3a2e"}
```


```
# Get data for all accounts
query = '''

SELECT
  tickets.customer_unique_id AS Accounts,
  COUNT(tickets.seat_unique_id) AS Tickets,
```

```
  COUNT(attendance.seat_unique_id) AS Scans,
  customers.price_code_type AS Membership,
  SUM(tickets.price) AS Sales
FROM
  tickets
  LEFT JOIN attendance ON attendance.seat_unique_id =
tickets.seat_unique_id
  LEFT JOIN customers ON customers.customer_unique_id =
tickets.customer_unique_id
GROUP BY
  Accounts

'''
accounts = sqldf(query)

# Get data for all tickets

query = '''

SELECT
  *
FROM
  tickets
  LEFT JOIN attendance ON attendance.seat_unique_id =
tickets.seat_unique_id
  LEFT JOIN customers ON customers.customer_unique_id =
tickets.customer_unique_id

'''
all_tickets = sqldf(query)
all_tickets =
all_tickets.loc[:,~all_tickets.columns.duplicated()].copy()

# Create master table with info by Game for modeling
results_df = results.merge(attendance_df.merge(teams[['City','Vegas
Over/Under 21/22']], 'inner'),'right',
on='Date').merge(sales_df,'right', on='Date').sort_values('Date')
results_df['Attending Rate'] = results_df.apply(lambda x:
int(100*round(x['Occupancy']/x['Sales'], 2)), axis=1)
results_df['Fill Rate'] = results_df.apply(lambda x:
int(100*round(x['Occupancy']/8377, 2)), axis=1)
results_df = results_df.merge(tickets[tickets['price']!
=0].groupby('event_name', as_index=False).agg({"price": [np.mean,
np.std, np.sum, np.min, np.median, np.max]}).round(2),
'right', left_on='Event', right_on='event_name')
results_df['Weekday'] = results_df['Date'].apply(lambda row:
row.weekday())
results_df =
results_df.drop([results_df.columns[18],results_df.columns[19],results
_df.columns[20], results_df.columns[21], 'G','Unnamed: 4', 'Event'],
```

```
axis=1)
results_df
```

	Date	Start (ET)	Opponent	Unnamed: 7	Tm	Opp	W	L
0	2021-10-23	10:30p	Memphis Grizzlies	L	114	120	0	2
1	2021-11-16	10:30p	San Antonio Spurs	W	106	92	9	5
2	2021-11-21	3:30p	Dallas Mavericks	W	97	91	10	7
3	2021-11-23	10:30p	Dallas Mavericks	L	104	112	10	8
4	2021-11-26	3:30p	Detroit Pistons	W	107	96	11	8
5	2021-11-28	3:30p	Golden State Warriors	L	90	105	11	9
6	2021-11-29	10:30p	New Orleans Pelicans	L	104	123	11	10
7	2021-12-01	10:30p	Sacramento Kings	L	115	124	11	11
8	2021-12-08	10:30p	Boston Celtics	W	114	111	14	12
9	2021-12-11	3:30p	Orlando Magic	W	106	104	15	12
10	2021-12-13	10:30p	Phoenix Suns	W	111	95	16	12
11	2021-10-25	10:30p	Portland Trail Blazers	W	116	86	1	2
12	2021-12-20	10:30p	San Antonio Spurs	L	92	116	16	15
13	2021-12-26	9:00p	Denver Nuggets	L	100	103	17	16
14	2021-12-27	10:30p	Brooklyn Nets	L	108	124	17	17
15	2022-01-03	10:30p	Minnesota Timberwolves	L	104	122	19	19
16	2022-01-08	3:30p	Memphis Grizzlies	L	108	123	19	21
17	2022-01-09	3:30p	Atlanta Hawks	W	106	93	20	21
18	2022-01-11	10:30p	Denver Nuggets	W	87	85	21	21
19	2022-01-17	3:30p	Indiana Pacers	W	139	133	22	23
20	2022-02-03	10:00p	Los Angeles Lakers	W	111	110	27	27
21	2022-02-06	9:00p	Milwaukee Bucks	L	113	137	27	28
22	2021-10-27	10:30p	Cleveland Cavaliers	L	79	92		

```
1   3
23 2022-02-14     10:30p   Golden State Warriors          W  119  104
29  30
24 2022-02-17     10:30p         Houston Rockets          W  142  111
30  31
25 2022-03-03     10:00p      Los Angeles Lakers          W  132  111
34  31
26 2022-03-06     10:00p         New York Knicks          L   93  116
34  32
27 2022-03-09     10:30p      Washington Wizards          W  115  109
35  33
28 2022-03-16     10:30p         Toronto Raptors          L  100  103
36  36
29 2022-03-25     10:30p      Philadelphia 76ers          L   97  122
36  39
30 2022-03-29     10:00p               Utah Jazz          W  121  115
37  39
31 2022-04-03      9:30p    New Orleans Pelicans          W  119  100
39  40
32 2022-04-06     10:00p            Phoenix Suns          W  113  109
40  40
33 2021-11-01     10:30p   Oklahoma City Thunder          W   99   94
2   4
34 2022-04-09      9:30p        Sacramento Kings          W  117   98
41  40
35 2022-04-10      9:30p   Oklahoma City Thunder          W  138   88
42  40
36 2021-11-07      9:00p       Charlotte Hornets          W  120  106
5   4
37 2021-11-09     10:00p  Portland Trail Blazers          W  117  109
6   4
38 2021-11-11     10:30p             Miami Heat          W  112  109
7   4
39 2021-11-13     10:30p  Minnesota Timberwolves          W  129  102
8   4
40 2021-11-14      9:30p           Chicago Bulls          L   90  100
8   5

   Streak  Occupancy           City  Vegas Over/Under 21/22  Sales  \
0     L 2       6498        Memphis                    41.5   7934
1     W 1       4928    San Antonio                    28.5   7126
2     W 1       5085         Dallas                    48.5   6832
3     L 1       5903         Dallas                    48.5   7339
4     W 1       5210        Detroit                    24.5   7477
5     L 1       6832   Golden State                    48.5   7622
6     L 2       4257    New Orleans                    39.5   6713
7     L 3       4558     Sacramento                    36.5   6747
8     W 2       6179         Boston                    45.5   7456
9     W 3       4814        Orlando                    22.5   7361
10    W 4       6190        Phoenix                    51.5   7678
```

11	W 1	5738	Portland		44.5	7417
12	L 3	5778	San Antonio		28.5	7835
13	L 1	5566	Denver		47.5	7399
14	L 2	5829	Brooklyn		56.5	7497
15	L 1	4587	Minnesota		35.5	7170
16	L 3	4850	Memphis		41.5	7591
17	W 1	4053	Atlanta		47.5	6830
18	W 2	4369	Denver		47.5	6921
19	W 1	4212	Indiana		42.5	6901
20	W 1	6913	Los Angeles		52.5	7610
21	L 1	6403	Milwaukee		54.5	7678
22	L 1	4866	Cleveland		26.5	7044
23	W 2	7311	Golden State		48.5	8033
24	W 1	5849	Houston		27.5	8006
25	W 5	7376	Los Angeles		52.5	8073
26	L 1	6198	New York		41.5	7925
27	W 1	5629	Washington		33.5	7734
28	L 2	6774	Toronto		35.5	8189
29	L 5	7162	Philadelphia		50.5	8178
30	W 1	6192	Utah		52.5	7853
31	W 2	6164	New Orleans		39.5	7586
32	W 3	6631	Phoenix		51.5	7850
33	W 1	4784	Oklahoma City		23.5	7142
34	W 4	5942	Sacramento		36.5	7685
35	W 5	5972	Oklahoma City		23.5	7534
36	W 4	6303	Charlotte		38.5	7838
37	W 5	5531	Portland		44.5	7339
38	W 6	6200	Miami		48.5	7340
39	W 7	6045	Minnesota		35.5	7731
40	L 1	6789	Chicago		42.5	7888

	Attending Rate	Fill Rate	(price, amin)	(price, median)	(price, amax)
0	82	78	18.38	107.00	481.00
1	69	59	20.00	83.00	373.00
2	74	61	35.76	171.00	573.00
3	80	70	35.76	159.00	555.00
4	70	62	19.81	83.00	373.00
5	90	82	35.76	281.00	1177.00
6	63	51	15.60	83.00	310.00
7	68	54	16.47	77.00	373.00
8	83	74	24.23	108.00	

410.39
9 65 56 15.44 84.00
373.00
10 81 74 19.88 113.83
386.00
11 77 68 15.59 79.00
275.00
12 74 69 19.43 71.00
310.00
13 75 66 21.36 152.00
573.00
14 78 70 31.68 278.00
1320.00
15 64 55 15.44 71.00
275.00
16 64 57 16.84 80.00
379.14
17 59 48 15.17 173.08
600.00
18 63 52 15.17 108.00
462.00
19 61 50 14.54 77.00
310.00
20 91 83 26.05 357.00
1372.00
21 83 76 26.18 171.00
1313.00
22 69 57 11.64 82.71
373.00
23 91 87 35.76 281.00
2164.00
24 73 70 14.85 73.00
834.00
25 91 88 35.76 330.00
1416.00
26 78 74 34.66 159.00
924.00
27 73 67 14.68 77.00
373.00
28 83 81 21.25 72.25
462.00
29 88 85 26.05 159.00
600.00
30 79 74 10.00 107.00
462.00
31 81 74 19.81 105.00
407.00
32 84 79 10.00 108.00
834.00
33 67 56 4.68 71.00

275.00
34 77 71 10.00 79.00
373.00
35 79 71 24.23 79.00
310.00
36 80 75 21.80 76.00
373.00
37 75 66 7.39 84.00
399.00
38 84 74 11.11 107.00
399.00
39 78 72 17.34 80.00
373.00
40 86 81 34.95 109.89
428.00

	Weekday
0	5
1	1
2	6
3	1
4	4
5	6
6	0
7	2
8	2
9	5
10	0
11	0
12	0
13	6
14	0
15	0
16	5
17	6
18	1
19	0
20	3
21	6
22	2
23	0
24	3
25	3
26	6
27	2
28	2
29	4
30	1
31	6
32	2

```
33        0
34        5
35        6
36        6
37        1
38        3
39        5
40        6
```

```
# Import quick modeling library and create forecast model with basic
info
import pycaret
from pycaret.regression import *
model_df = results_df[['Occupancy','Date', 'Start (ET)', 'Opponent',
'Vegas Over/Under 21/22']].sort_values('Date')
stp = setup(data = model_df, target = 'Occupancy', train_size = 0.7,
            categorical_features = ['Start (ET)', 'Opponent'],
            silent = True, session_id=1)
best_model = compare_models(sort='RMSE')
predictions = predict_model(best_model,
data=model_df).sort_values('Date').rename(columns={'Label':'Prediction
'})
fig = px.line(predictions, x='Date', y=["Occupancy", "Prediction"],
template = 'plotly_dark').update_layout(hovermode="x unified")
```

```
                                    Model           MAE           MSE
\
ridge                     Ridge Regression  6.282127e+02  5.679077e+05

lasso                     Lasso Regression  6.369302e+02  5.881137e+05

ada                     AdaBoost Regressor  6.808026e+02  6.550500e+05

lr                       Linear Regression  7.155777e+02  6.898959e+05

et                   Extra Trees Regressor  6.980585e+02  7.461641e+05

rf                 Random Forest Regressor  7.256715e+02  6.825759e+05

huber                     Huber Regressor  7.400967e+02  7.983871e+05

en                            Elastic Net  6.947059e+02  7.720155e+05

llar         Lasso Least Angle Regression  7.360759e+02  9.102712e+05

knn                 K Neighbors Regressor  7.673833e+02  8.432312e+05

br                         Bayesian Ridge  7.517336e+02  8.941649e+05
```

gbr	Gradient Boosting Regressor	8.095444e+02	8.157527e+05
lightgbm	Light Gradient Boosting Machine	7.529047e+02	8.778159e+05
dummy	Dummy Regressor	7.529047e+02	8.778159e+05
omp	Orthogonal Matching Pursuit	8.345117e+02	9.292953e+05
dt	Decision Tree Regressor	8.588333e+02	9.710548e+05
par	Passive Aggressive Regressor	1.349294e+03	2.316206e+06
lar	Least Angle Regression	3.242528e+19	1.059372e+40

	RMSE	R2	RMSLE	MAPE	TT (Sec)
ridge	7.010735e+02	-2.799000e-01	0.1251	1.150000e-01	0.015
lasso	7.128371e+02	-3.188000e-01	0.1350	1.193000e-01	0.016
ada	7.692246e+02	-4.981000e-01	0.1374	1.254000e-01	0.075
lr	7.885182e+02	-8.315000e-01	0.1466	1.325000e-01	0.014
et	7.928199e+02	-1.037500e+00	0.1393	1.240000e-01	0.354
rf	8.051119e+02	-9.225000e-01	0.1437	1.310000e-01	0.405
huber	8.111353e+02	-1.039700e+00	0.1529	1.363000e-01	0.043
en	8.218678e+02	-4.965000e-01	0.1478	1.291000e-01	0.015
llar	8.629876e+02	-1.166300e+00	0.1548	1.358000e-01	0.021
knn	8.791020e+02	-1.152500e+00	0.1591	1.415000e-01	0.062
br	8.885432e+02	-6.728000e-01	0.1596	1.393000e-01	0.017
gbr	8.886683e+02	-1.454800e+00	0.1546	1.432000e-01	0.042
lightgbm	8.939354e+02	-7.092000e-01	0.1610	1.392000e-01	0.019
dummy	8.939354e+02	-7.092000e-01	0.1610	1.392000e-01	0.012
omp	9.048262e+02	-1.592200e+00	0.1578	1.465000e-01	0.013
dt	9.452119e+02	-2.568500e+00	0.1665	1.505000e-01	0.017

```
par       1.426332e+03 -3.797200e+00   0.2687  2.517000e-01     0.015

lar       3.255221e+19 -1.317463e+34  25.4641  5.982867e+15     0.023

INFO:logs:create_model_container: 18
INFO:logs:master_model_container: 18
INFO:logs:display_container: 2
INFO:logs:Ridge(alpha=1.0, copy_X=True, fit_intercept=True,
max_iter=None,
      normalize=False, random_state=1, solver='auto', tol=0.001)
INFO:logs:compare_models() succesfully
completed.....................................
INFO:logs:Initializing predict_model()
INFO:logs:predict_model(estimator=Ridge(alpha=1.0, copy_X=True,
fit_intercept=True, max_iter=None,
      normalize=False, random_state=1, solver='auto', tol=0.001),
probability_threshold=None, encoded_labels=True, drift_report=False,
raw_score=False, round=4, verbose=True,
ml_usecase=MLUsecase.REGRESSION, display=None, drift_kwargs=None)
INFO:logs:Checking exceptions
INFO:logs:Preloading libraries
INFO:logs:Preparing display monitor

              Model       MAE          MSE      RMSE      R2   RMSLE
MAPE
0  Ridge Regression  347.0587  194679.6313  441.2251  0.7521  0.0827
0.0649

predict_model(best_model)

INFO:logs:Initializing predict_model()
INFO:logs:predict_model(estimator=Ridge(alpha=1.0, copy_X=True,
fit_intercept=True, max_iter=None,
      normalize=False, random_state=1, solver='auto', tol=0.001),
probability_threshold=None, encoded_labels=True, drift_report=False,
raw_score=False, round=4, verbose=True,
ml_usecase=MLUsecase.REGRESSION, display=None, drift_kwargs=None)
INFO:logs:Checking exceptions
INFO:logs:Preloading libraries
INFO:logs:Preparing display monitor

              Model       MAE          MSE      RMSE      R2  RMSLE
MAPE
0  Ridge Regression  438.1464  262757.4318  512.5987  0.6008   0.09
0.0778

    Vegas Over/Under 21/22  Start (ET)_2022-10-02 153000  \
0                     23.5                             0.0
1                     26.5                             0.0
2                     41.5                             0.0
```

3	51.5	0.0
4	22.5	0.0
5	54.5	0.0
6	28.5	0.0
7	35.5	0.0
8	27.5	0.0
9	35.5	0.0
10	56.5	0.0
11	41.5	0.0
12	52.5	0.0

	Start (ET)_2022-10-02 210000	Start (ET)_2022-10-02 213000	\
0	1.0	0.0	
1	1.0	0.0	
2	1.0	0.0	
3	1.0	0.0	
4	1.0	0.0	
5	1.0	0.0	
6	1.0	0.0	
7	1.0	0.0	
8	1.0	0.0	
9	1.0	0.0	
10	1.0	0.0	
11	1.0	0.0	
12	1.0	0.0	

	Start (ET)_2022-10-02 220000	Start (ET)_2022-10-02 223000	\
0	0.0	0.0	
1	0.0	0.0	
2	0.0	0.0	
3	0.0	0.0	
4	0.0	0.0	
5	0.0	0.0	
6	0.0	0.0	
7	0.0	0.0	
8	0.0	0.0	
9	0.0	0.0	
10	0.0	0.0	
11	0.0	0.0	
12	0.0	0.0	

	Opponent_Atlanta Hawks	Opponent_Boston Celtics	\
0	0.0	0.0	
1	0.0	1.0	
2	0.0	0.0	
3	0.0	0.0	
4	0.0	1.0	
5	0.0	1.0	
6	0.0	0.0	
7	0.0	1.0	

```
8                       0.0                       1.0
9                       0.0                       0.0
10                      0.0                       1.0
11                      0.0                       1.0
12                      0.0                       0.0

    Opponent_Charlotte Hornets  Opponent_Chicago Bulls  ...
Date_weekday_1  \
0                           0.0                     0.0  ...
0.0
1                           0.0                     0.0  ...
0.0
2                           0.0                     0.0  ...
0.0
3                           0.0                     0.0  ...
0.0
4                           0.0                     0.0  ...
0.0
5                           0.0                     0.0  ...
0.0
6                           0.0                     0.0  ...
0.0
7                           0.0                     0.0  ...
0.0
8                           0.0                     0.0  ...
0.0
9                           0.0                     0.0  ...
0.0
10                          0.0                     0.0  ...
0.0
11                          0.0                     0.0  ...
0.0
12                          0.0                     0.0  ...
0.0

    Date_weekday_2  Date_weekday_3  Date_weekday_4  Date_weekday_5  \
0              0.0             0.0             0.0             0.0
1              1.0             0.0             0.0             0.0
2              0.0             0.0             0.0             1.0
3              1.0             0.0             0.0             0.0
4              0.0             0.0             0.0             1.0
5              0.0             0.0             0.0             0.0
6              0.0             0.0             0.0             0.0
7              1.0             0.0             0.0             0.0
8              0.0             1.0             0.0             0.0
9              0.0             0.0             0.0             0.0
10             0.0             0.0             0.0             0.0
11             0.0             0.0             0.0             0.0
12             0.0             1.0             0.0             0.0
```

```
    Date_weekday_6  Date_is_month_end_0  Date_is_month_start_1
Occupancy  \
0              0.0                  1.0                    1.0
4784
1              0.0                  1.0                    0.0
4866
2              0.0                  1.0                    0.0
4850
3              0.0                  1.0                    0.0
6631
4              0.0                  1.0                    0.0
4814
5              1.0                  1.0                    0.0
6403
6              0.0                  1.0                    0.0
5778
7              0.0                  1.0                    0.0
6774
8              0.0                  1.0                    0.0
5849
9              0.0                  1.0                    0.0
4587
10             0.0                  1.0                    0.0
5829
11             1.0                  1.0                    0.0
6198
12             0.0                  1.0                    0.0
6913

          Label
0   4685.197266
1   5498.680664
2   5487.795410
3   6626.144531
4   5347.124023
5   7115.119141
6   4940.372559
7   6023.887695
8   6043.401367
9   4744.860352
10  6386.342773
11  6453.666504
12  7236.515137

[13 rows x 46 columns]
```

```
# Get schedule data and input to model to predict future

# Create an URL object of subject of scrape
url = 'https://www.basketball-reference.com/teams/LAC/2023_games.html'
```

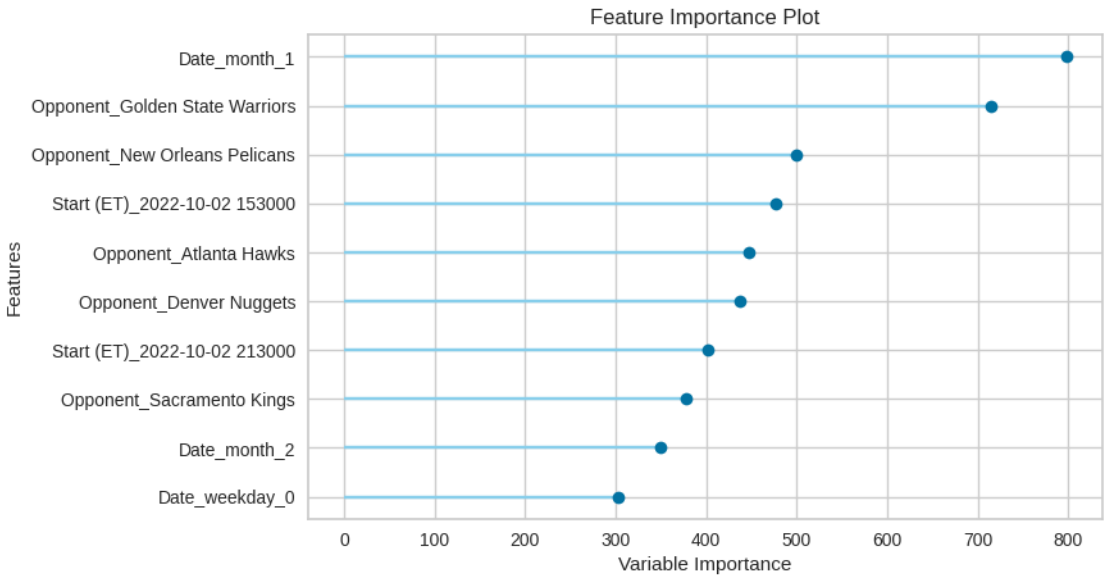
```
lookup_table = results_df.merge(teams[['City','Vegas Over/Under
22/23']], 'inner')[['Opponent', 'City', 'Vegas Over/Under 22/23']]

# Scrape HTML table with Pandas and add table with Vegas Over/Under
values
future_test = pd.read_html(url)
[0].merge(lookup_table[['Opponent','Vegas Over/Under 22/23']],
'inner')[['Date', 'Start (ET)', 'Opponent', 'Vegas Over/Under
22/23','Unnamed: 5']].drop_duplicates()
# Remove null rows and columns and filter out away games
future_test = future_test[(future_test['Date']!="Date") &
(future_test['Unnamed: 5']!="@")].drop('Unnamed: 5', axis=1)
future_test ['Date'] = pd.to_datetime(future_test['Date']) # Reformat
date to match standardization
future_test['Date'] =
future_test['Date'].dt.strftime('%D').apply(lambda row:
datetime.strptime(row, '%m/%d/%y').date()) # Change text to date
future_test['Date'] = pd.to_datetime(future_test['Date']) # Reformat
date to match standardization
future_test =
future_test.sort_values('Date').reset_index(drop=True).rename(columns=
{'Vegas Over/Under 22/23': 'Vegas Over/Under 21/22'})

# Generate predictions on the original and future dataset
predictions = predict_model(best_model,
data=model_df).sort_values('Date').rename(columns={'Label':'Prediction
'})
predictions_future = predict_model(best_model,
data=future_test).sort_values('Date').rename(columns={'Label':'Predict
ion'})

# Plot line of real vs. model
fig = px.line(predictions, x='Date', y=["Occupancy", "Prediction"],
template = 'plotly_dark').update_layout(hovermode="x unified")
fig.show()
# Plot prediction of next season 22-23
fig = px.line(predictions_future, x='Date', y=["Prediction"], template
= 'plotly_dark').update_traces(line_color='#ff0000',
line_width=5).update_layout(hovermode="x unified")
fig.show()
```

```
INFO:logs:Initializing predict_model()
INFO:logs:predict_model(estimator=Ridge(alpha=1.0, copy_X=True,
fit_intercept=True, max_iter=None,
      normalize=False, random_state=1, solver='auto', tol=0.001),
probability_threshold=None, encoded_labels=True, drift_report=False,
raw_score=False, round=4, verbose=True,
ml_usecase=MLUsecase.REGRESSION, display=None, drift_kwargs=None)
INFO:logs:Checking exceptions
```

```
INFO:logs:Preloading libraries
INFO:logs:Preparing display monitor

              Model       MAE           MSE      RMSE      R2    RMSLE
MAPE
0  Ridge Regression  347.0587  194679.6313  441.2251  0.7521  0.0827
0.0649

INFO:logs:Initializing predict_model()
INFO:logs:predict_model(estimator=Ridge(alpha=1.0, copy_X=True,
fit_intercept=True, max_iter=None,
      normalize=False, random_state=1, solver='auto', tol=0.001),
probability_threshold=None, encoded_labels=True, drift_report=False,
raw_score=False, round=4, verbose=True,
ml_usecase=MLUsecase.REGRESSION, display=None, drift_kwargs=None)
INFO:logs:Checking exceptions
INFO:logs:Preloading libraries
INFO:logs:Preparing display monitor
```

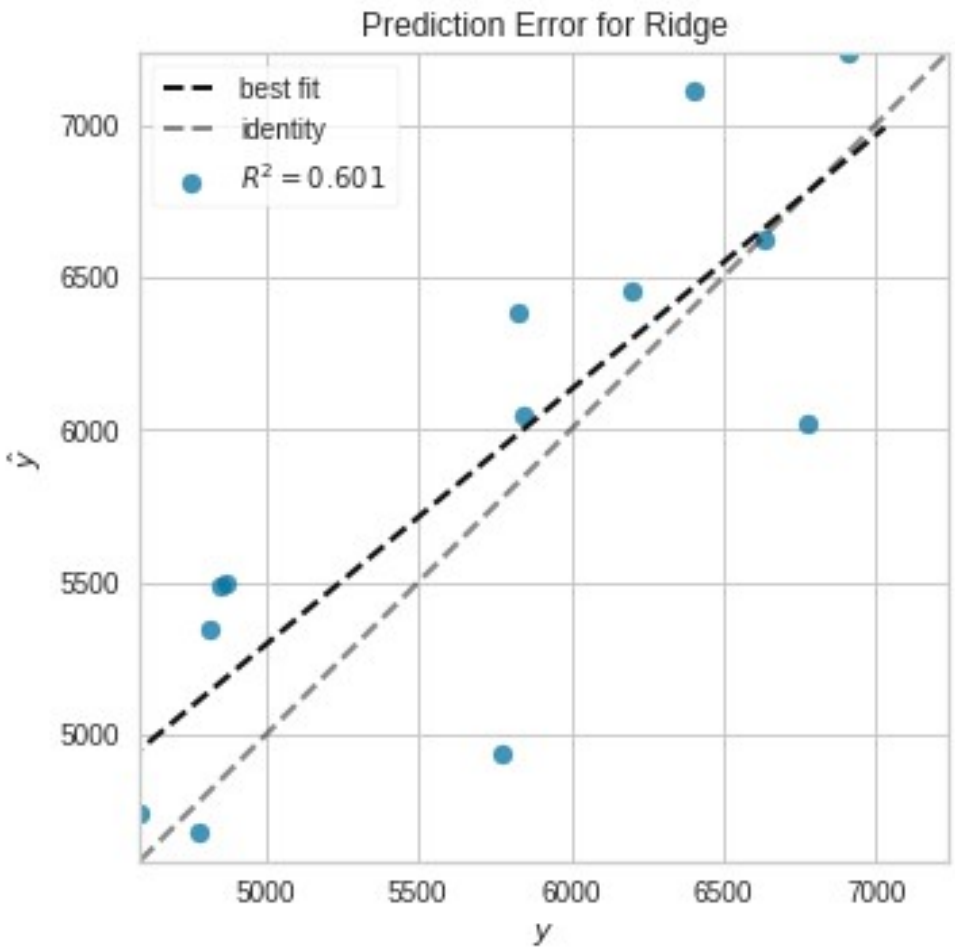
```
# Draft Model using price data for prediction
# stp = setup(data = results_df.drop(['Fill Rate','Attending Rate',
'Sales'], axis=1), target = 'Occupancy', train_size = 0.8,
#             categorical_features = ['Unnamed: 7', 'Start (ET)',
'Opponent', 'Streak', 'Weekday'],
#            # ignore_features = ['Loan_ID'],
#             silent = True, session_id=1)
# best_model = compare_models()
```

```
plot_model(best_model, plot='feature')
```




```
INFO:logs:Visual Rendered Successfully
INFO:logs:plot_model() succesfully
completed.....................................
```

```
plot_model(best_model, plot = 'error')
```




```
INFO:logs:Visual Rendered Successfully
INFO:logs:plot_model() succesfully
completed.....................................
```

```
model_final = finalize_model(best_model)
save_model(model_final, 'model')
```

```
INFO:logs:Initializing finalize_model()
INFO:logs:finalize_model(estimator=Ridge(alpha=1.0, copy_X=True,
fit_intercept=True, max_iter=None,
      normalize=False, random_state=1, solver='auto', tol=0.001),
fit_kwargs=None, groups=None, model_only=True, display=None,
experiment_custom_tags=None, return_train_score=False)
INFO:logs:Finalizing Ridge(alpha=1.0, copy_X=True, fit_intercept=True,
max_iter=None,
      normalize=False, random_state=1, solver='auto', tol=0.001)
INFO:logs:Initializing create_model()
INFO:logs:create_model(estimator=Ridge(alpha=1.0, copy_X=True,
fit_intercept=True, max_iter=None,
```

```
      normalize=False, random_state=1, solver='auto', tol=0.001),
fold=None, round=4, cross_validation=True, predict=True,
fit_kwargs={}, groups=None, refit=True, verbose=False, system=False,
metrics=None, experiment_custom_tags=None, add_to_model_list=False,
probability_threshold=None, display=None, return_train_score=False,
kwargs={})
INFO:logs:Checking exceptions
INFO:logs:Importing libraries
INFO:logs:Copying training dataset
INFO:logs:Defining folds
INFO:logs:Declaring metric variables
INFO:logs:Importing untrained model
INFO:logs:Declaring custom model
INFO:logs:Ridge Regression Imported succesfully
INFO:logs:Starting cross validation
INFO:logs:Cross validating with KFold(n_splits=10, random_state=None,
shuffle=False), n_jobs=-1
INFO:logs:Calculating mean and std
INFO:logs:Creating metrics dataframe
INFO:logs:Finalizing model
INFO:logs:create_model_container: 18
INFO:logs:master_model_container: 18
INFO:logs:display_container: 4
INFO:logs:Ridge(alpha=1.0, copy_X=True, fit_intercept=True,
max_iter=None,
      normalize=False, random_state=1, solver='auto', tol=0.001)
INFO:logs:create_model() succesfully
completed.....................................
INFO:logs:create_model_container: 18
INFO:logs:master_model_container: 18
INFO:logs:display_container: 3
INFO:logs:Ridge(alpha=1.0, copy_X=True, fit_intercept=True,
max_iter=None,
      normalize=False, random_state=1, solver='auto', tol=0.001)
INFO:logs:finalize_model() succesfully
completed.....................................
INFO:logs:Initializing save_model()
INFO:logs:save_model(model=Ridge(alpha=1.0, copy_X=True,
fit_intercept=True, max_iter=None,
      normalize=False, random_state=1, solver='auto', tol=0.001),
model_name=model, prep_pipe_=Pipeline(memory=None,
         steps=[('dtypes',
                 DataTypes_Auto_infer(categorical_features=['Start
(ET)',

'Opponent'],
                                      display_types=False,
features_todrop=[],
                                      id_columns=[],
ml_usecase='regression',
```

```
                                      numerical_features=[],
target='Occupancy',
                                      time_features=[])),
                ('imputer',
                 Simple_Imputer(categorical_strategy='not_available',
                                fill_value_categorical=None,
                                fill_value_numeri...
                ('scaling', 'passthrough'), ('P_transform',
'passthrough'),
                ('binn', 'passthrough'), ('rem_outliers',
'passthrough'),
                ('cluster_all', 'passthrough'),
                ('dummy', Dummify(target='Occupancy')),
                ('fix_perfect', Remove_100(target='Occupancy')),
                ('clean_names', Clean_Colum_Names()),
                ('feature_select', 'passthrough'), ('fix_multi',
'passthrough'),
                ('dfs', 'passthrough'), ('pca', 'passthrough')],
         verbose=False), verbose=True, kwargs={})
INFO:logs:Adding model into prep_pipe
INFO:logs:model.pkl saved in current working directory
INFO:logs:Pipeline(memory=None,
         steps=[('dtypes',
                 DataTypes_Auto_infer(categorical_features=['Start
(ET)',

'Opponent'],
                                      display_types=False,
features_todrop=[],
                                      id_columns=[],
ml_usecase='regression',
                                      numerical_features=[],
target='Occupancy',
                                      time_features=[])),
                ('imputer',
                 Simple_Imputer(categorical_strategy='not_available',
                                fill_value_categorical=None,
                                fill_value_numeri...
                ('dummy', Dummify(target='Occupancy')),
                ('fix_perfect', Remove_100(target='Occupancy')),
                ('clean_names', Clean_Colum_Names()),
                ('feature_select', 'passthrough'), ('fix_multi',
'passthrough'),
                ('dfs', 'passthrough'), ('pca', 'passthrough'),
                ['trained_model',
                 Ridge(alpha=1.0, copy_X=True, fit_intercept=True,
                       max_iter=None, normalize=False, random_state=1,
                       solver='auto', tol=0.001)]],
         verbose=False)
```

```
INFO:logs:save_model() successfully
completed.....................................

Transformation Pipeline and Model Successfully Saved

(Pipeline(memory=None,
          steps=[('dtypes',
                  DataTypes_Auto_infer(categorical_features=['Start
(ET)',

'Opponent'],
                                       display_types=False,
features_todrop=[],
                                       id_columns=[],
ml_usecase='regression',
                                       numerical_features=[],
target='Occupancy',
                                       time_features=[])),
                 ('imputer',
                  Simple_Imputer(categorical_strategy='not_available',
                                 fill_value_categorical=None,
                                 fill_value_numeri...
                 ('dummy', Dummify(target='Occupancy')),
                 ('fix_perfect', Remove_100(target='Occupancy')),
                 ('clean_names', Clean_Colum_Names()),
                 ('feature_select', 'passthrough'), ('fix_multi',
'passthrough'),
                 ('dfs', 'passthrough'), ('pca', 'passthrough'),
                 ['trained_model',
                  Ridge(alpha=1.0, copy_X=True, fit_intercept=True,
                        max_iter=None, normalize=False,
random_state=1,
                        solver='auto', tol=0.001)]],
          verbose=False), 'model.pkl')

# Predict using the model with sample data
test_data = pd.DataFrame([(datetime(2022,12,20),"10:30p","San Antonio
Spurs",28.5)],
                  columns=['Date', 'Start (ET)', 'Opponent', 'Vegas
Over/Under 21/22'])
print(predict_model(model_final,data=test_data)['Label'])
# Prediction: 5257
predict_model(model_final,data=test_data)

INFO:logs:Initializing predict_model()
INFO:logs:predict_model(estimator=Ridge(alpha=1.0, copy_X=True,
fit_intercept=True, max_iter=None,
      normalize=False, random_state=1, solver='auto', tol=0.001),
probability_threshold=None, encoded_labels=True, drift_report=False,
raw_score=False, round=4, verbose=True,
ml_usecase=MLUsecase.REGRESSION, display=None, drift_kwargs=None)
```

```
INFO:logs:Checking exceptions
INFO:logs:Preloading libraries
INFO:logs:Preparing display monitor
INFO:logs:Initializing predict_model()
INFO:logs:predict_model(estimator=Ridge(alpha=1.0, copy_X=True,
fit_intercept=True, max_iter=None,
      normalize=False, random_state=1, solver='auto', tol=0.001),
probability_threshold=None, encoded_labels=True, drift_report=False,
raw_score=False, round=4, verbose=True,
ml_usecase=MLUsecase.REGRESSION, display=None, drift_kwargs=None)
INFO:logs:Checking exceptions
INFO:logs:Preloading libraries
INFO:logs:Preparing display monitor

0    5256.843262
Name: Label, dtype: float32

        Date Start (ET)           Opponent  Vegas Over/Under 21/22  \
0 2022-12-20     10:30p  San Antonio Spurs                    28.5

         Label
0  5256.843262
```