data_eda

April 16, 2024

1 Exploratory Data Analysis - Data

4 types of data are provided: 1. Ticket Sales Data - sales data from the tickets purchased and scanned at the United Center 2. Email Engagement Data - these contain information about interactions attendees have with the emails sent before and after the game. * KBYG - know before you go emails are the pre-game emails * Postgame - postgame emails sent after the game 3. LiveA Demographic Data - 3rd Party Provider data to augment with ticket sales and email engagment data to profile individuals with demographic characteristics 4. Bulls Theme + Giveaway Schedule - schedule of promotional activity for each game.

[1]:
```
import pandas as pd
```

1.1 Ticket Sales

The ticket sales data represents instances of tickets redeemed at the United Center for Chicago Bulls home games.

We will rename the columns to make them more code friendly. Find the data dictionary below.

Variable Name	Code Friendly Name	Description	Detail	Field Type
Event Information				
Season Name	season_name	Indicates the NBA season that each event occurred within	NBA season runs from October to April	CHARACTER
Event Date	event_date	The date of the game the tickets were purchased for		DATE
Event Weekday	event_weekday	The day of the week of each game		CHARACTER
Opponent (Short)	opponent_short	The nickname of the opposing team the Bulls played	Ex. - Lakers, Pelicans (city/geographic location excluded)	CHARACTER
Ticket Purchase Information				

Variable Name	Code Friendly Name	Description	Detail	Field Type
Parent Ticket Categories	parent_ticket_categories	Indicates the type of ticket being purchased	Group = group purchases: Resale Buyer= Subscription	CHARACTER
Add Date	add_date	The date on which the tickets were purchased and/or added to		DATE
Days before Event	days_before_event	The number of days in advance of the event that the ticket was purchased	Date difference calculation between Add Date and Event Date	INTEGER
Purchaser Email	purchaser_email	The email contact associated with the purchase record		CHARACTER
Total Seats	total_seats	The total number of seats purchased in this specific transaction		INTEGER
Seat Location	seat_location	Indicates the level of the United Center where the seats are located	100 Level = lower bowl of seats, sections all around the arena	CHARACTER
Seat Level	seat_level	Indicates the level of the United Center where the seats are located	Lower Level = 100 Level (open to all fans)Level = 200 and 300 levels, requires specific ticket access	CHARACTER
Attendance Informa-tion				
Arrival Time	arrival_time	Indicates the datetime that the attendee scanned their ticket at the gate	Blank = Un-scanned ticket (indicates a ticket that was sold but not used)	DATETIME
Attendee Email	attendee_email	The email contact associated with the attendee		CHARACTER
Attendee Zip Code	attebdee_zip_code	The zip code of the address associated with the attendee		INTEGER
Mobile Scan	mobile_scan	Indicates whether the ticket was scanned using a mobile device	1 = Mobile scan0 = Non-mobile scan= Un-scanned	BINARY

Variable Name	Code Friendly Name	Description	Detail	Field Type
Scan Category	scan_category	Indicates the method the attendee used to access the event		CHARACTER

[2]:
```
# Load data from CSV file
df_tickets = pd.read_csv("/Users/jm/dev/acl_spring_24_bulls2/data/raw/Ticket␣
 ↪Sales Data/22-23 & 23-24 (through 3.14) Ticket Sales Data.csv",
                      names = [
                        'season_name', 'event_date', 'event_weekday',␣
 ↪'opponent_short',
                        'parent_ticket_categories', 'add_date',␣
 ↪'days_before_event',
                        'purchaser_email', 'total_seats', 'seat_location',␣
 ↪'seat_level',
                        'arrival_time', 'attendee_email', 'attendee_zip_code',␣
 ↪'mobile_scan',
                        'scan_category'
                    ],
                    header = 0
                )
```

The data needs to be adjusted to reflect the data types specified in the data dictionary.

[3]:
```
# Fix Date Data Types
df_tickets["event_date"] = pd.to_datetime(df_tickets['event_date'], format =␣
 ↪"%d-%b-%y")
df_tickets["add_date"] = pd.to_datetime(df_tickets['add_date'], format =␣
 ↪"%d-%b-%y")
df_tickets["arrival_time"] = pd.to_datetime(df_tickets['arrival_time'], format␣
 ↪= "%m/%d/%y %H:%M")

# Convert mobile_scan to a boolean
df_tickets['mobile_scan'] = df_tickets.mobile_scan.astype("bool")
```

Filter to just the season ticket holders using the 'parent_ticket_categories' variable.

[4]:
```
df_season = df_tickets[df_tickets['parent_ticket_categories'] == "Season"]
```

[5]:
```
df_season.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 270086 entries, 300179 to 570264
Data columns (total 16 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
```

```
 0   season_name               270086 non-null  object
 1   event_date                270086 non-null  datetime64[ns]
 2   event_weekday             270086 non-null  object
 3   opponent_short            270086 non-null  object
 4   parent_ticket_categories  270086 non-null  object
 5   add_date                  270086 non-null  datetime64[ns]
 6   days_before_event         270086 non-null  int64
 7   purchaser_email           268908 non-null  object
 8   total_seats               270086 non-null  int64
 9   seat_location             270086 non-null  object
 10  seat_level                270086 non-null  object
 11  arrival_time              237206 non-null  datetime64[ns]
 12  attendee_email            236956 non-null  object
 13  attendee_zip_code         201700 non-null  object
 14  mobile_scan               270086 non-null  bool
 15  scan_category             270086 non-null  object
dtypes: bool(1), datetime64[ns](3), int64(2), object(10)
memory usage: 33.2+ MB
```

[6]:
```
df_season.describe()
```

[6]:
```
                          event_date                       add_date  \
count                         270086                         270086
mean   2023-05-30 20:20:41.311286016  2022-09-15 19:45:03.112341760
min              2022-10-04 00:00:00            2022-01-28 00:00:00
25%              2022-12-28 00:00:00            2022-01-28 00:00:00
50%              2023-03-17 00:00:00            2022-08-17 00:00:00
75%              2023-12-02 00:00:00            2023-02-01 00:00:00
max              2024-03-14 00:00:00            2024-03-14 00:00:00
std                              NaN                            NaN

       days_before_event    total_seats                   arrival_time
count      270086.000000  270086.000000                         237206
mean          257.024748       1.991147  2023-06-01 06:09:33.202279424
min           -61.000000       1.000000            2022-10-04 18:36:00
25%           177.000000       1.000000            2022-12-30 17:31:00
50%           269.000000       2.000000            2023-03-17 18:38:00
75%           336.000000       2.000000            2023-12-02 18:47:00
max           436.000000      14.000000            2024-03-14 20:52:00
std           101.241811       0.899302                            NaN
```

[7]:
```
df_season.describe(include=['O'])
```

[7]:
```
                    season_name event_weekday opponent_short  \
count                    270086        270086         270086
unique                        2             7             29
top     2022-2023 Chicago Bulls           Wed          Bucks
freq                     159078         58950          16972
```

```
       parent_ticket_categories             purchaser_email     seat_location  \
count                    270086                      268908            270086
unique                        1                        3891                 6
top                      Season  michaelsblechman@gmail.com  100 Level Center
freq                     270086                         527             77916

         seat_level             attendee_email attendee_zip_code  \
count        270086                     236956            201700
unique            5                     116211             14895
top     Lower Level  michaelsblechman@gmail.com             60614
freq         122890                        279              4939

       scan_category
count         270086
unique             5
top      Web Browser
freq          146121
```

1.1.1 Ticket Sales Data Summary

There are 270,068 season ticket purchasers that were scanned at Chicago Bulls games at the United Center over the 2022-2023 season and 2023-2024 (through March 19th) season. The data contains 15 columns.

There are missing values in the `purchaser_email`, `attendee_email`, `arrival_time`, and `attendee_zip_code` columns.

1.2 Email Engagement Data

The email engagment data is broken into two files corresponding to the pre-game and post-game email schedule. One email is sent before every Chicago Bulls game and another is sent after every game. The data collected in both of these files represents the clickthrough data for both types of email sent. Both files contain data for both the '22 - '23 season and '23 - '24 season.

Everytime an attendee clicks a link on one of these emails a row is entered.

The two types of email engagment data are: 1. KBYG - Know Before You Go: these emails are sent before the game to ticket holders with pre-game instructions. Usually sent the same day. 2. Post-Game: These emails are sent after the game with promotional material.

1.2.1 Email - Data Dictionary

We will rename the columns to make them more code friendly. Find the data dictionary below.

Variable Name	Code Friendly Name	Description	Detail	Field Type
Email Address	attendee_email	email of the attendee of the game		CHARACTER

Variable Name	Code Friendly Name	Description	Detail	Field Type
Clickthrough Link	clickthrough_url	clickthrough link/URL		CHARACTER
Email Clickthrough Date/Time	clickthrough_dt	clickthrough date and time		DATE
Email Name	email_name	name of the email sent out		CHARACTER
Total Clickthroughs	total_clickthroughs	total count of click throughs		INTEGER
Email Send Date	email_send_dt	date and time the email was sent		DATE
Unique Clickthroughs	unique_clickthroughs	the number of unique clickthroughs		INTEGER
Clickthrough Link Count	clickthrough_link_count	??? count		INTEGER
Season	season	the season		CHARACTER
	email_type	the type of email: kbyg or post		CHARACTER

Since both data sets follow the same data structure we will concatenate the datasets. We have also added in a column to denote which type of email is sent.

[8]:
```
# Load data from CSV file
#Email Address,  Clickthrough Link,  Email Clickthrough Date/Time,  Email␣
 ↪Name,Total Clickthroughs,Email Send Date,Unique Clickthroughs,Clickthrough␣
 ↪Link Count,Season

df_kbyg = pd.read_csv("/Users/jm/dev/acl_spring_24_bulls2/data/raw/Email␣
 ↪Engagement Data/KBYG Clickthrough Data_2223 and 2324 Seasons.csv"
                  ,
                       names = [
                         'attendee_email', 'clickthrough_url', 'clickthrough_dt',
                         'email_name', 'total_clickthroughs', 'email_send_dt',
                         'unique_clickthroughs', 'clickthrough_link_count',␣
 ↪'season'
                      ],
                      header = 0
                  )
df_kbyg['email_type'] = 'kbyg'

df_post = pd.read_csv("/Users/jm/dev/acl_spring_24_bulls2/data/raw/Email␣
 ↪Engagement Data/Postgame Email Clickthrough_2223 and 2324 seasons.csv"
                  ,
                       names = [
                         'attendee_email', 'clickthrough_url', 'clickthrough_dt',
                         'email_name', 'total_clickthroughs', 'email_send_dt',
```

```
                        'unique_clickthroughs', 'clickthrough_link_count',␣
 ↪'season'
                    ],
                    header = 0
                )
df_post['email_type'] = 'post'

df_emails = pd.concat([df_kbyg, df_post])
```

[9]:
```
# Fix Date Data Types
df_emails["clickthrough_dt"] = pd.to_datetime(df_emails['clickthrough_dt'],␣
 ↪format = "%m/%d/%Y %H:%M")
df_emails["email_send_dt"] = pd.to_datetime(df_emails['email_send_dt'], format␣
 ↪= "%m/%d/%Y %H:%M")
```

[10]:
```
df_emails.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 37613 entries, 0 to 23382
Data columns (total 10 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   attendee_email           37613 non-null  object
 1   clickthrough_url         37613 non-null  object
 2   clickthrough_dt          37613 non-null  datetime64[ns]
 3   email_name               37613 non-null  object
 4   total_clickthroughs      37613 non-null  int64
 5   email_send_dt            37613 non-null  datetime64[ns]
 6   unique_clickthroughs     37613 non-null  int64
 7   clickthrough_link_count  37613 non-null  int64
 8   season                   37613 non-null  int64
 9   email_type               37613 non-null  object
dtypes: datetime64[ns](2), int64(4), object(4)
memory usage: 3.2+ MB
```

[11]:
```
df_emails.describe()
```

[11]:
```
                      clickthrough_dt  total_clickthroughs  \
count                           37613         37613.000000
mean   2023-07-09 01:27:44.730811392             1.000239
min              2022-10-22 16:13:00             1.000000
25%              2023-01-14 15:22:00             1.000000
50%              2023-04-10 16:38:00             1.000000
75%              2023-12-31 13:07:00             1.000000
max              2024-03-20 12:16:00             2.000000
std                              NaN             0.015467

                      email_send_dt  unique_clickthroughs  \
```

```
count                           37613                37613.0
mean   2023-07-08 09:45:42.328450304                    1.0
min              2022-10-22 16:00:00                    1.0
25%              2023-01-14 14:00:00                    1.0
50%              2023-04-10 13:00:00                    1.0
75%              2023-12-30 16:00:00                    1.0
max              2024-03-19 12:00:00                    1.0
std                              NaN                    0.0

       clickthrough_link_count        season
count             37613.000000  37613.000000
mean                  1.000239   2272.918645
min                   1.000000   2223.000000
25%                   1.000000   2223.000000
50%                   1.000000   2223.000000
75%                   1.000000   2324.000000
max                   2.000000   2324.000000
std                   0.015467     50.497325
```

[12]:
```
df_emails.describe(include=['O'])
```

[12]:
```
               attendee_email  \
count                   37613
unique                  25295
top     lzerante@corpconc.com
freq                       95

                                        clickthrough_url  \
count                                              37613
unique                                                47
top     https://bulls.qualtrics.com/jfe/form/SV_cNsSkl…
freq                                               10731

                               email_name email_type
count                               37613      37613
unique                                120          2
top     2023_CB_Marketing_Surveys_PostGame       post
freq                                11399      23383
```

Now we have to filter for just those emails that correspond to season ticket holders from the tickets data.

[13]:
```
df_season_emails = df_emails[df_emails['attendee_email'].
 ↪isin(df_season['attendee_email'].unique())]
```

[14]:
```
df_season_emails.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
Index: 8158 entries, 1 to 23372
Data columns (total 10 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   attendee_email           8158 non-null   object
 1   clickthrough_url         8158 non-null   object
 2   clickthrough_dt          8158 non-null   datetime64[ns]
 3   email_name               8158 non-null   object
 4   total_clickthroughs      8158 non-null   int64
 5   email_send_dt            8158 non-null   datetime64[ns]
 6   unique_clickthroughs     8158 non-null   int64
 7   clickthrough_link_count  8158 non-null   int64
 8   season                   8158 non-null   int64
 9   email_type               8158 non-null   object
dtypes: datetime64[ns](2), int64(4), object(4)
memory usage: 701.1+ KB
```

[15]:
```
df_season_emails.describe()
```

[15]:

	clickthrough_dt	total_clickthroughs
count	8158	8158.000000
mean	2023-06-25 18:07:12.378033920	1.000490
min	2022-10-22 16:13:00	1.000000
25%	2023-01-01 10:40:30	1.000000
50%	2023-04-02 17:54:30	1.000000
75%	2024-01-01 14:05:30	1.000000
max	2024-03-20 03:23:00	2.000000
std	NaN	0.022139

	email_send_dt	unique_clickthroughs
count	8158	8158.0
mean	2023-06-25 01:19:15.724442368	1.0
min	2022-10-22 16:00:00	1.0
25%	2022-12-31 14:00:00	1.0
50%	2023-04-02 12:00:00	1.0
75%	2023-12-31 13:00:00	1.0
max	2024-03-19 12:00:00	1.0
std	NaN	0.0

	clickthrough_link_count	season
count	8158.000000	8158.000000
mean	1.000490	2270.305835
min	1.000000	2223.000000
25%	1.000000	2223.000000
50%	1.000000	2223.000000
75%	1.000000	2324.000000
max	2.000000	2324.000000

```
std                   0.022139    50.401971
```

[16]:
```
df_season_emails.describe(include=['O'])
```

[16]:
```
               attendee_email  \
count                    8158
unique                   4776
top     mephgrave@bacardi.com
freq                       44

                                        clickthrough_url  \
count                                               8158
unique                                                46
top     https://bulls.qualtrics.com/jfe/form/SV_cNsSkl…
freq                                                1570

                                email_name email_type
count                                 8158       8158
unique                                 119          2
top     2023_CB_Marketing_Surveys_PostGame       kbyg
freq                                  1282       5143
```

1.2.2 Email Engagement Data Summary

There are 8,158 email clickthroughs by season ticket holder game attendees. This is a drop off from the 37K total email interactions. There are no nulls. The `unique_clickthroughs` columns is useless since all values indicate a clickthrough event occured and therefore is always equal to 1.

[17]:
```
print("Percentage of attendees that are season ticket holders: ",␣
 ↪len(df_season['attendee_email'].unique())/len(df_tickets['attendee_email'].
 ↪unique())*100, '%')
print("Percentage of email clickthroughs (engagement) made by season ticket␣
 ↪holders: ", len(df_season_emails['attendee_email'].unique())/
 ↪len(df_emails['attendee_email'].unique())*100, '%')
```

```
Percentage of attendees that are season ticket holders:  37.113257198334225 %
Percentage of email clickthroughs (engagement) made by season ticket holders:
18.881201818541214 %
```

Interesting Fact

While season ticket holders make up 37% of all gameday scans, they only make up 18.8% of the email engagement either before or after the game. This may point to an opportunity to increase season ticket holder engagement since they seem to engage less than their fair share.

The response variable we intend to use will be based on this data. We will be looking for clusters in our data and hope to see stronger correlations with higher propensities to engage with the emails in those target customers.

[ ]:
```

```

1.3 Live Analytics Data

The Live Anlaytics contains demographic data for ticketholder attendees.

See the data dictionary for relevant information.

[18]:
```
df_liva23 = pd.read_csv("/Users/jm/dev/acl_spring_24_bulls2/data/raw/LiveA␣
↪Demographic Data/22-23 LiveA (Season Ticketholder Attendees).csv")

df_liva24 = pd.read_csv("/Users/jm/dev/acl_spring_24_bulls2/data/raw/LiveA␣
↪Demographic Data/23-24 LiveA (Season Ticketholder Attendees).csv")
```

```
/var/folders/3q/d621fwvs43q2f6s93g46njd00000gn/T/ipykernel_85626/3471165704.py:3
: DtypeWarning: Columns (6) have mixed types. Specify dtype option on import or
set low_memory=False.
  df_liva24 = pd.read_csv("/Users/jm/dev/acl_spring_24_bulls2/data/raw/LiveA
Demographic Data/23-24 LiveA (Season Ticketholder Attendees).csv")
```

[20]:
```
print(df_liva23.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 76884 entries, 0 to 76883
Columns: 253 entries, ult_party_id to vehicle_type
dtypes: float64(217), int64(1), object(35)
memory usage: 148.4+ MB
None
```

[23]:
```
df_liva23.describe()
```

[23]:

	ult_party_id	acct_id	age_two_yr_incr_input_indv	\
count	5.528800e+04	7.688400e+04	48803.000000	
mean	5.221303e+08	2.382599e+07	41.809704	
std	5.934971e+08	8.032407e+06	13.966835	
min	4.693330e+07	1.000120e+05	18.000000	
25%	8.229506e+07	2.071969e+07	30.000000	
50%	1.911863e+08	2.797470e+07	40.000000	
75%	1.040008e+09	2.967570e+07	50.000000	
max	1.746297e+09	3.125330e+07	99.000000	

	age_two_yr_incr_1st_indv	age_two_yr_incr_2nd_indv	race_cd	\
count	49936.000000	24328.000000	0.0	
mean	45.213373	49.783295	NaN	
std	13.630294	15.246266	NaN	
min	18.000000	18.000000	NaN	
25%	34.000000	40.000000	NaN	
50%	44.000000	50.000000	NaN	
75%	54.000000	60.000000	NaN	
max	99.000000	99.000000	NaN	

```
       adult_hh_num  hh_male_18_24_ind  hh_female_18_24_ind  hh_unk_18_24_ind  \
count  52362.000000             2720.0               2582.0             132.0
mean       1.925748                1.0                  1.0               1.0
std        0.964341                0.0                  0.0               0.0
min        1.000000                1.0                  1.0               1.0
25%        1.000000                1.0                  1.0               1.0
50%        2.000000                1.0                  1.0               1.0
75%        2.000000                1.0                  1.0               1.0
max        6.000000                1.0                  1.0               1.0

       …  propn_score_minor_499  client_event_cnt    client_tkt  \
count  …           57505.000000      55575.000000  55575.000000
mean   …             508.406486          0.936824      2.417256
std    …             177.416817          2.187956      9.741104
min    …              64.000000          0.000000      0.000000
25%    …             378.000000          0.000000      0.000000
50%    …             508.000000          0.000000      0.000000
75%    …             636.000000          1.000000      2.000000
max    …             985.000000        115.000000   1231.000000

           client_sp  client_pe_tkt_cnt  client_pe_sp  client_tkt_price  \
count   55575.000000       24954.000000  25261.000000      24954.000000
mean      360.540082           2.574754    366.636250        148.390474
std      1571.368671           1.292205    433.674626        152.124717
min         0.000000           1.000000      5.000000          5.000000
25%         0.000000           2.000000    156.500000         69.172500
50%         0.000000           2.000000    258.720000        110.000000
75%       327.840000           3.000000    428.000000        179.000000
max    155742.680000          34.400000  19188.640000       3500.000000

       client_tkt_price_max  client_tkt_price_min  client_walkup_buyer_ind
count          24954.000000          24954.000000                      0.0
mean             177.148323            126.229601                      NaN
std              205.705139            142.751814                      NaN
min                5.000000              4.250000                      NaN
25%               76.000000             50.400000                      NaN
50%              129.000000             90.000000                      NaN
75%              204.000000            150.000000                      NaN
max             6720.000000           3500.000000                      NaN

[8 rows x 218 columns]
```

[ ]: