Starbucks_Capstone_notebook

March 5, 2021

1 Starbucks Capstone Challenge

1.0.1 Introduction

This data set contains simulated data that mimics customer behavior on the Starbucks rewards mobile app. Once every few days, Starbucks sends out an offer to users of the mobile app. An offer can be merely an advertisement for a drink or an actual offer such as a discount or BOGO (buy one get one free). Some users might not receive any offer during certain weeks.

Not all users receive the same offer, and that is the challenge to solve with this data set.

Your task is to combine transaction, demographic and offer data to determine which demographic groups respond best to which offer type. This data set is a simplified version of the real Starbucks app because the underlying simulator only has one product whereas Starbucks actually sells dozens of products.

Every offer has a validity period before the offer expires. As an example, a BOGO offer might be valid for only 5 days. You'll see in the data set that informational offers have a validity period even though these ads are merely providing information about a product; for example, if an informational offer has 7 days of validity, you can assume the customer is feeling the influence of the offer for 7 days after receiving the advertisement.

You'll be given transactional data showing user purchases made on the app including the timestamp of purchase and the amount of money spent on a purchase. This transactional data also has a record for each offer that a user receives as well as a record for when a user actually views the offer. There are also records for when a user completes an offer.

Keep in mind as well that someone using the app might make a purchase through the app without having received an offer or seen an offer.

2

2.0.1

. . BOGO ( ). .
.
. .
. BOGO 5 . 7 7 .
. . .
.

2.0.2 Example

To give an example, a user could receive a discount offer buy 10 dollars get 2 off on Monday. The offer is valid for 10 days from receipt. If the customer accumulates at least 10 dollars in purchases during the validity period, the customer completes the offer.

However, there are a few things to watch out for in this data set. Customers do not opt into the offers that they receive; in other words, a user can receive an offer, never actually view the offer, and still complete the offer. For example, a user might receive the "buy 10 dollars get 2 dollars off offer", but the user never opens the offer during the 10 day validity period. The customer spends 15 dollars during those ten days. There will be an offer completion record in the data set; however, the customer was not influenced by the offer because the customer never viewed the offer.

2.0.3

10 2 . 10 . 10 .
. . . "10 2 " 10 . 15 . .

2.0.4 Cleaning

This makes data cleaning especially important and tricky.

You'll also want to take into account that some demographic groups will make purchases even if they don't receive an offer. From a business perspective, if a customer is going to make a 10 dollar purchase without an offer anyway, you wouldn't want to send a buy 10 dollars get 2 dollars off offer. You'll want to try to assess what a certain demographic group will buy when not receiving any offers.

2.0.5

.
. 10 10 2 . .

2.0.6 Final Advice

Because this is a capstone project, you are free to analyze the data any way you see fit. For example, you could build a machine learning model that predicts how much someone will spend based on demographics and offer type. Or you could build a model that predicts whether or not someone will respond to an offer. Or, you don't need to build a machine learning model at all. You could develop a set of heuristics that determine what offer you should send to each customer (i.e., 75 percent of women customers who were 35 years old responded to offer A vs 40 percent from the same demographic to offer B, so send offer A).

2.0.7

capstone . . . . (75 A 40 B A).

3 Data Sets

The data is contained in three files:

- portfolio.json - containing offer ids and meta data about each offer (duration, type, etc.)
- profile.json - demographic data for each customer
- transcript.json - records for transactions, offers received, offers viewed, and offers completed

Here is the schema and explanation of each variable in the files:

portfolio.json * id (string) - offer id * offer_type (string) - type of offer ie BOGO, discount, informational * difficulty (int) - minimum required spend to complete an offer * reward (int) - reward given for completing an offer * duration (int) - time for offer to be open, in days * channels (list of strings)

profile.json * age (int) - age of the customer * became_member_on (int) - date when customer created an app account * gender (str) - gender of the customer (note some entries contain 'O' for other rather than M or F) * id (str) - customer id * income (float) - customer's income

transcript.json * event (str) - record description (ie transaction, offer received, offer viewed, etc.) * person (str) - customer id * time (int) - time in hours since start of test. The data begins at time t=0 * value - (dict of strings) - either an offer id or transaction amount depending on the record

Note: If you are using the workspace, you will need to go to the terminal and run the command `conda update pandas` before reading in the files. This is because the version of pandas in the workspace cannot read in the transcript.json file correctly, but the newest version of pandas can. You can access the termnal from the orange icon in the top left of this notebook.

You can see how to access the terminal and how the install works using the two images below. First you need to access the terminal:

Then you will want to run the above command:

Finally, when you enter back into the notebook (use the jupyter icon again), you should be able to run the below cell without any errors.

In [1470]:
```
# Load Library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
import json
% matplotlib inline

import seaborn as sns

sns.set()
```

In [1471]:
```
# read in the json files
portfolio = pd.read_json('data/portfolio.json', orient='records', lines=True)
profile = pd.read_json('data/profile.json', orient='records', lines=True)
transcript = pd.read_json('data/transcript.json', orient='records', lines=True)
```

4 data analysis

4.0.1 Data Understanding

4.0.2 portfolio

```
In [1472]: portfolio.head(10)
           #   id          (string) - offer id
           #   offer_type (string) - type of offer ie BOGO, discount, informational
           #   difficulty (int)    - minimum required spend to complete an offer
           #   reward     (int)    - reward given for completing an offer
           #   duration   (int)    - time for offer to be open, in days
           #   channels   (list of strings)
```

Out[1472]:

	channels	difficulty	duration	\
0	[email, mobile, social]	10	7	
1	[web, email, mobile, social]	10	5	
2	[web, email, mobile]	0	4	
3	[web, email, mobile]	5	7	
4	[web, email]	20	10	
5	[web, email, mobile, social]	7	7	
6	[web, email, mobile, social]	10	10	
7	[email, mobile, social]	0	3	
8	[web, email, mobile, social]	5	5	
9	[web, email, mobile]	10	7	

	id	offer_type	reward
0	ae264e3637204a6fb9bb56bc8210ddfd	bogo	10
1	4d5c57ea9a6940dd891ad53e9dbe8da0	bogo	10
2	3f207df678b143eea3cee63160fa8bed	informational	0
3	9b98b8c7a33c4b65b9aebfe6a799e6d9	bogo	5
4	0b1e1539f2cc45b7b9fa7c272da2e1d7	discount	5
5	2298d6c36e964ae4a3e7e9706d1fb8c2	discount	3
6	fafdcd668e3743c1bb461111dcafc2a4	discount	2
7	5a8bc65990b245e5a138643cd4eb9837	informational	0
8	f19421c1d4aa40978ebb69ca19b0e20d	bogo	5
9	2906b810c7d4411798c6938adc9daaa5	discount	2

```
In [1473]: portfolio.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 6 columns):
channels      10 non-null object
difficulty    10 non-null int64
duration      10 non-null int64
id            10 non-null object
offer_type    10 non-null object
reward        10 non-null int64
```

```
dtypes: int64(3), object(3)
memory usage: 560.0+ bytes
```

```
In [1474]: # checking the count of each offer type
           portfolio.offer_type.value_counts()
```

```
Out[1474]: discount         4
           bogo             4
           informational    2
           Name: offer_type, dtype: int64
```

4.0.3 profile

```
In [1475]: profile.head()
           #   age              (int)   - age of the customer
           #   became_member_on (int)   - date when customer created an app account
           #   gender           (str)   - gender of the customer (note some entries contain 'O'
           #   id               (str)   - customer id
           #   income           (float) - customer's income
```

```
Out[1475]:    age  became_member_on gender                                id    income
           0  118           20170212   None  68be06ca386d4c31939f3a4f0e3dd783       NaN
           1   55           20170715      F  0610b486422d4921ae7d2bf64640c50b  112000.0
           2  118           20180712   None  38fe809add3b4fcf9315a9694bb96ff5       NaN
           3   75           20170509      F  78afa995795e4d85b5d9ceeca43f5fef  100000.0
           4  118           20170804   None  a03223e636434f42ac4c3df47e8bac43       NaN
```

```
In [1476]: profile.info()
```

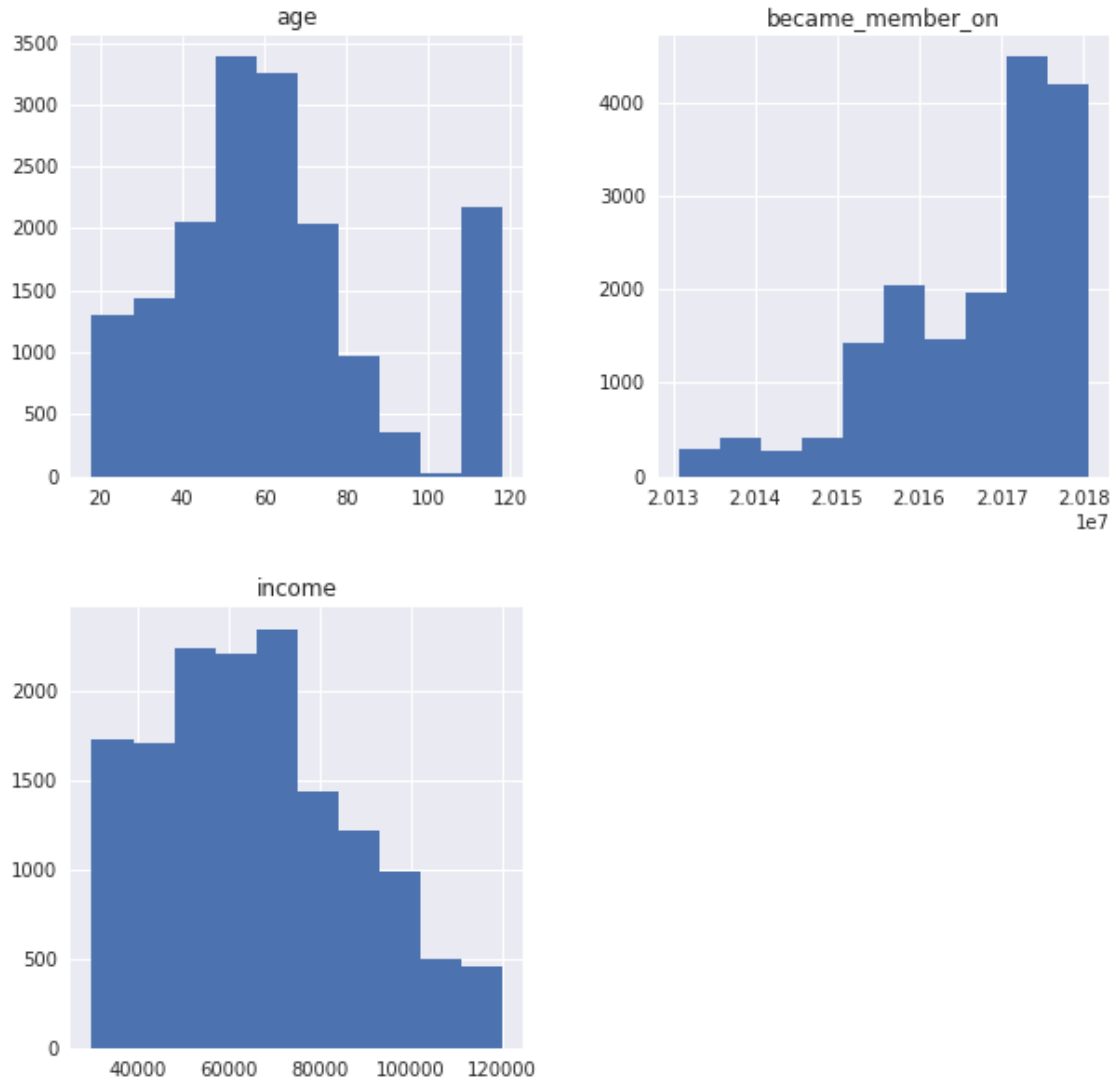
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17000 entries, 0 to 16999
Data columns (total 5 columns):
age                 17000 non-null int64
became_member_on    17000 non-null int64
gender              14825 non-null object
id                  17000 non-null object
income              14825 non-null float64
dtypes: float64(1), int64(2), object(2)
memory usage: 664.1+ KB
```

```
In [1477]: profile.describe()
```

```
Out[1477]:                 age  became_member_on         income
           count  17000.000000      1.700000e+04   14825.000000
           mean      62.531412      2.016703e+07   65404.991568
           std       26.738580      1.167750e+04   21598.299410
           min       18.000000      2.013073e+07   30000.000000
```

```
25%        45.000000       2.016053e+07    49000.000000
50%        58.000000       2.017080e+07    64000.000000
75%        73.000000       2.017123e+07    80000.000000
max       118.000000       2.018073e+07   120000.000000
```

In [1478]:
```
profile.hist(figsize=(10,10));
```



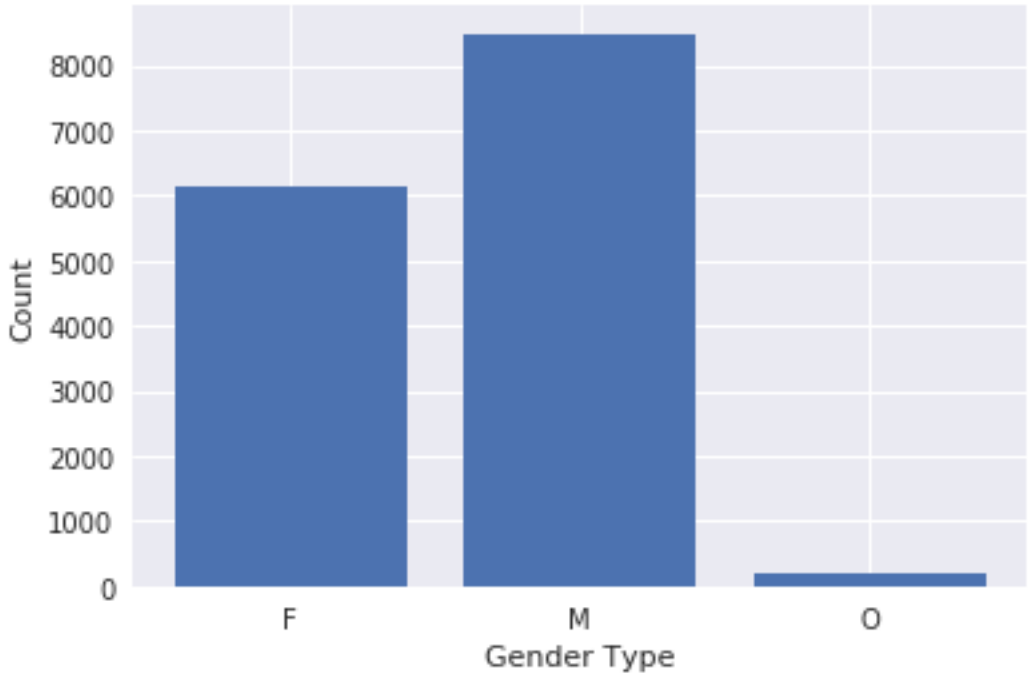
In [1479]:
```
profile['gender'].unique()
```

Out[1479]:
```
array([None, 'F', 'M', 'O'], dtype=object)
```

In [1480]:
```
gender=profile.gender.value_counts()
gender
```

Out[1480]:
```
M    8484
F    6129
O     212
Name: gender, dtype: int64
```

In [1481]:
```
profile_gender_counts = profile.gender.value_counts()
x = ['M','F','O']
data = profile_gender_counts
plt.bar(x,height = data);
xlocs, xlabs = plt.xticks()

plt.xlabel('Gender Type');
plt.ylabel('Count');
```



In [1482]:
```
# is any missing values across columns
profile.isnull().sum()
```

Out[1482]:
```
age                  0
became_member_on     0
gender            2175
id                   0
income            2175
dtype: int64
```

In [1483]:
```
# creating a dataframe with only the customers with age = 118
# this data frame will include the coressponding gender and income columns to the cus
df_118 = profile[['gender','income','age']][profile['age']==118]
```

```
In [1484]: df_118.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 2175 entries, 0 to 16994
Data columns (total 3 columns):
gender    0 non-null object
income    0 non-null float64
age       2175 non-null int64
dtypes: float64(1), int64(1), object(1)
memory usage: 68.0+ KB
```

we have 2175 customer's without any info for gender,income and age

4.0.4 transcript

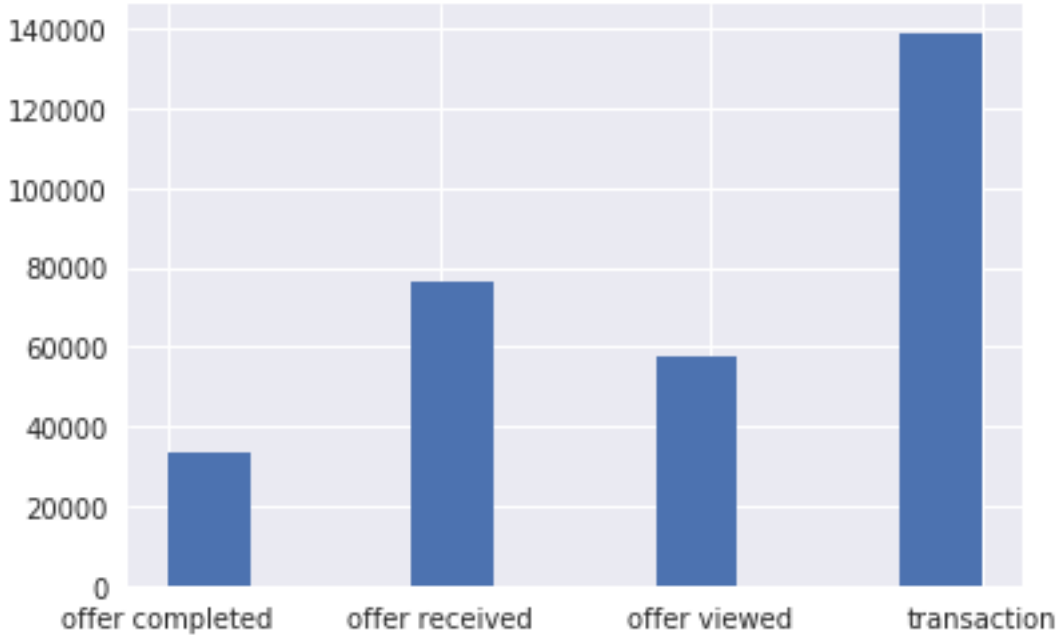
```
In [1485]: transcript.head()
           #   event  (str)             - record description (ie transaction, offer received, of
           #   person (str)             - customer id
           #   time   (int)             - time in hours since start of test. The data begins at
           #   value  (dict of strings) - either an offer id or transaction amount depending on
```

Out[1485]:

	event	person	time	value
0	offer received	78afa995795e4d85b5d9ceeca43f5fef	0	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}
1	offer received	a03223e636434f42ac4c3df47e8bac43	0	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}
2	offer received	e2127556f4f64592b11af22de27a7932	0	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}
3	offer received	8ec6ce2a7e7949b1bf142def7d0e0586	0	{'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}
4	offer received	68617ca6246f4fbc85e91a2a49552598	0	{'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}

```
In [1486]: transcript.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 306534 entries, 0 to 306533
Data columns (total 4 columns):
event     306534 non-null object
person    306534 non-null object
time      306534 non-null int64
value     306534 non-null object
dtypes: int64(1), object(3)
memory usage: 9.4+ MB
```

In [1487]:
```
# checking the count of each event type
x=transcript.event
x.hist();
```

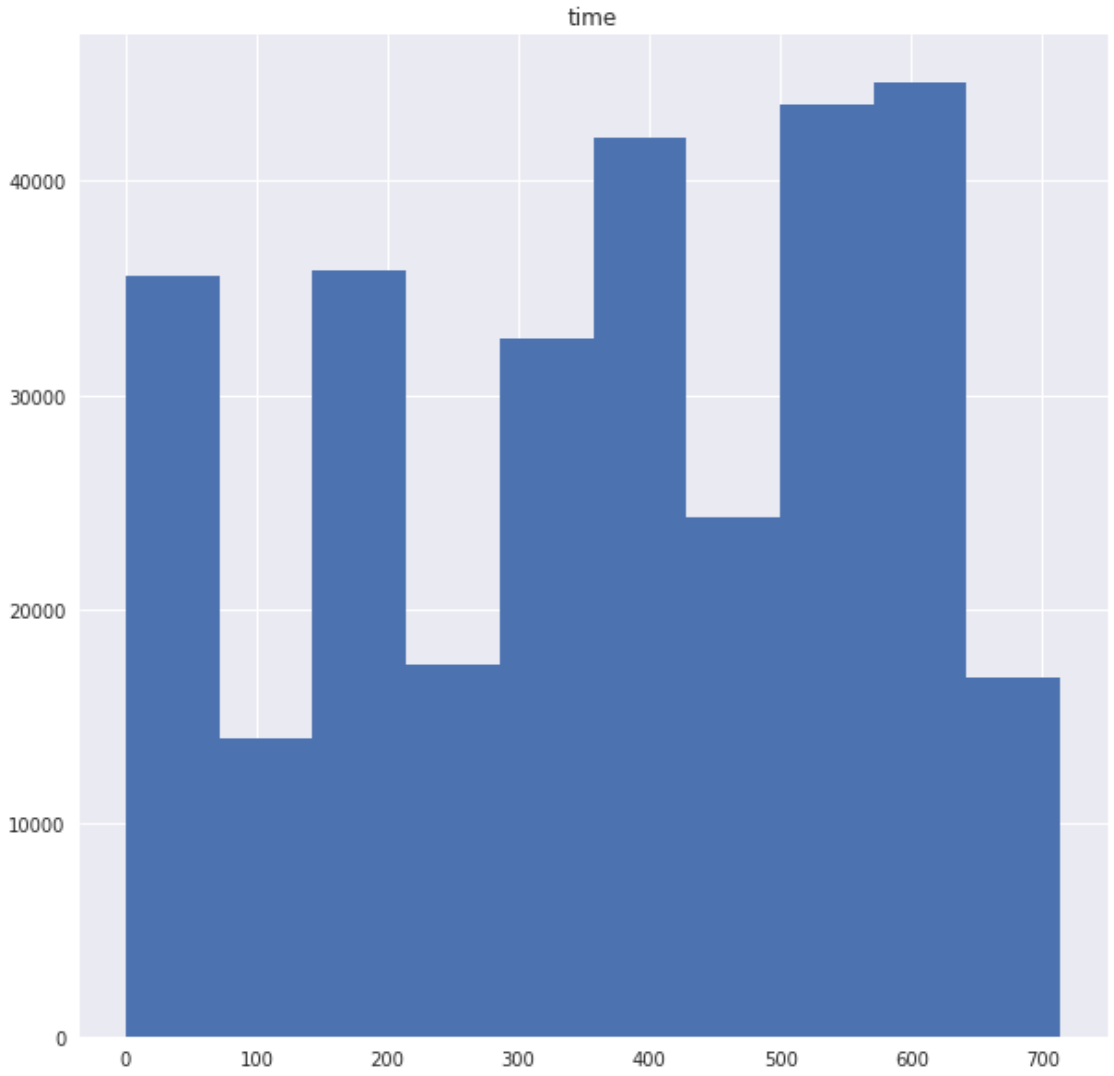


In [1488]:
```
transcript.describe()
```

Out[1488]:

	time
count	306534.000000
mean	366.382940
std	200.326314
min	0.000000
25%	186.000000
50%	408.000000
75%	528.000000
max	714.000000

In [1489]:
```
transcript.hist(figsize=(10,10));
```




```
In [1490]: transcript[['value']][transcript['event']=='transaction'].head()

Out[1490]:                                value
           12654  {'amount': 0.8300000000000001}
           12657                {'amount': 34.56}
           12659                {'amount': 13.23}
           12670                {'amount': 19.51}
           12671                {'amount': 18.97}

In [1491]: transcript[['value']][transcript['event']=='offer received'].head()

Out[1491]:                                                  value
           0  {'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}
           1  {'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}
```

```
2  {'offer id': '2906b810c7d4411798c6938adc9daaa5'}
3  {'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}
4  {'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}
```

```
In [1492]: transcript[['value']][transcript['event']=='offer viewed'].head()
```

```
Out[1492]:                                                  value
12650  {'offer id': 'f19421c1d4aa40978ebb69ca19b0e20d'}
12651  {'offer id': '5a8bc65990b245e5a138643cd4eb9837'}
12652  {'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}
12653  {'offer id': 'ae264e3637204a6fb9bb56bc8210ddfd'}
12655  {'offer id': '5a8bc65990b245e5a138643cd4eb9837'}
```

```
In [1493]: transcript[['value']][transcript['event']=='offer completed'].head()
```

```
Out[1493]:                                                   value
12658  {'offer_id': '2906b810c7d4411798c6938adc9daaa5...
12672  {'offer_id': 'fafdcd668e3743c1bb461111dcafc2a4...
12679  {'offer_id': '9b98b8c7a33c4b65b9aebfe6a799e6d9...
12692  {'offer_id': 'ae264e3637204a6fb9bb56bc8210ddfd...
12697  {'offer_id': '4d5c57ea9a6940dd891ad53e9dbe8da0...
```

4.0.5 Review Business Questions:

- What is the predict for event when have gender, age group, income group and offer type will complet offer or not in 20018 year ?

- What is the predict for most offer type in 20018 year ?

4.0.6 Cleaning the Datasets

Covert columns values to columns zero and one for ML

```
In [1494]: # Create function cxtract data into new columns from a column with iterable values li
           def extract_from_iterable_col(df, old_col, drop_old_col=False, unique_values=None, un

               '''
               Extract data into new columns from a column with iterable values like lists or di
               If extracting from dictionaries, the new column names will be the dictionary keys
               values will be dictionary values. If extracting from lists, one-hot encode the li

               Args:
                   (1) df (Pandas dataframe) - data containing a column with iterable values
                   (2) old_col (str) - name of column to extract data from
                   (3) drop_old_col (bool) - whether or not to drop the old column after extract
                   (4) unique_values (list[str]) - pass in unique values if data type of column
                   (5) unique_values (list[str]) - pass in unique keys if data type of column is
                   * Pass in arg (4) for a list column or arg (5) for a dictionary column
```

```
        Returns:
            Same data with new columns extracted from the old column (Pandas dataframe).
        '''

        df = df.copy()

        if unique_keys is not None: # for dicts
            for k in unique_keys:
                if ' ' not in k: # to skip duplicate key with space
                    df[old_col+'_'+k] = df[old_col].apply(lambda d: d[k] if k in d # get
                                                          else (d[k.replace('_', ' ')] if k.replace('
                                                                else np.NaN)) # val is nan if key is

        elif unique_values is not None: # for lists
            for v in unique_values:
                new_col = df[old_col].apply(lambda lst: int(v in lst)) # 1 if val is in l
                if np.var(new_col): # if new col is not constant
                    df[old_col+'_'+v] = new_col # add new col

        if drop_old_col:
            df.drop(old_col, axis=1, inplace=True)

        return df
```

portfolio

```
In [1495]: # new portfolio
           portfolio=portfolio[['id','offer_type','channels']]
```

```
In [1496]: # add titel name for every offer
           for row in range(10):
               portfolio.at[row, 'offer_name']=row

           portfolio['offer_name']=portfolio['offer_name'].astype(int)
           portfolio.head(10)
```

Out[1496]:

	id	offer_type	\
0	ae264e3637204a6fb9bb56bc8210ddfd	bogo	
1	4d5c57ea9a6940dd891ad53e9dbe8da0	bogo	
2	3f207df678b143eea3cee63160fa8bed	informational	
3	9b98b8c7a33c4b65b9aebfe6a799e6d9	bogo	
4	0b1e1539f2cc45b7b9fa7c272da2e1d7	discount	
5	2298d6c36e964ae4a3e7e9706d1fb8c2	discount	
6	fafdcd668e3743c1bb461111dcafc2a4	discount	
7	5a8bc65990b245e5a138643cd4eb9837	informational	
8	f19421c1d4aa40978ebb69ca19b0e20d	bogo	
9	2906b810c7d4411798c6938adc9daaa5	discount	

```
                        channels  offer_name
0       [email, mobile, social]           0
1  [web, email, mobile, social]           1
2          [web, email, mobile]           2
3          [web, email, mobile]           3
4                  [web, email]           4
5  [web, email, mobile, social]           5
6  [web, email, mobile, social]           6
7       [email, mobile, social]           7
8  [web, email, mobile, social]           8
9          [web, email, mobile]           9
```

In [1497]:
```
# Extract list values from offer_type data
portfolio['offer_type'].unique()
```

Out[1497]:
```
array(['bogo', 'informational', 'discount'], dtype=object)
```

In [1498]:
```
# Extract list values from offer_type data
portfolio = extract_from_iterable_col(portfolio, 'offer_type', unique_values=['bogo',
profile['offer_type'].value_counts()
```

In [1499]:
```
# Extract list values from offer_type data
portfolio = extract_from_iterable_col(portfolio, 'channels', unique_values=['web', 'e
```

In [1500]:
```
# rename column id to offer_id
portfolio = portfolio.rename(columns = {'id':'offer_id'})
portfolio.head()
```

Out[1500]:
```
                          offer_id     offer_type  offer_name  \
0  ae264e3637204a6fb9bb56bc8210ddfd           bogo           0
1  4d5c57ea9a6940dd891ad53e9dbe8da0           bogo           1
2  3f207df678b143eea3cee63160fa8bed  informational           2
3  9b98b8c7a33c4b65b9aebfe6a799e6d9           bogo           3
4  0b1e1539f2cc45b7b9fa7c272da2e1d7       discount           4

   offer_type_bogo  offer_type_informational  offer_type_discount  \
0                1                         0                    0
1                1                         0                    0
2                0                         1                    0
3                1                         0                    0
4                0                         0                    1

   channels_web  channels_mobile  channels_social
0             0                1               1
1             1                1               1
2             1                1               0
3             1                1               0
4             1                0               0
```

profile Missing data for better analysis.

```
In [1501]: # Drop missing data for profile "age = 118" or age up 90 years.
           profile=profile[profile['age']<100]

In [1502]: # Drop missing data for profile "gender = 'O'".
           profile=profile[profile['gender']!='O']
           profile['gender'].value_counts()

Out[1502]: M    8482
           F    6115
           Name: gender, dtype: int64

In [1503]: # get year for became_member_on
           profile['became_member_on'] = pd.to_datetime(profile['became_member_on'], format='%Y%
           # set reg_year from became_member_on column that has the year which customers become
           profile['reg_year'] = profile['became_member_on'].dt.year
           # drop became_member_on column
           profile=profile.drop(['became_member_on'], axis=1)

           profile.head()

Out[1503]:     age gender                                id    income  reg_year
           1    55      F  0610b486422d4921ae7d2bf64640c50b  112000.0      2017
           3    75      F  78afa995795e4d85b5d9ceeca43f5fef  100000.0      2017
           5    68      M  e2127556f4f64592b11af22de27a7932   70000.0      2018
           8    65      M  389bc3fa690240e798340f5a15918d5c   53000.0      2018
           12   58      M  2eeac8d8feae4a8cad5a6af0499a211d   51000.0      2017

In [1504]: profile['reg_year'].value_counts()

Out[1504]: 2017    5501
           2018    3611
           2016    2992
           2015    1564
           2014     658
           2013     271
           Name: reg_year, dtype: int64

In [1505]: # replace gender one for man zero for woman
           profile['gender']=profile['gender'].replace(['M'],'1').replace(['F'],'0').astype(int)
           profile['gender'].value_counts()

Out[1505]: 1    8482
           0    6115
           Name: gender, dtype: int64

In [1506]: # set age group
           profile['age']=(((profile['age']/10).astype(int))*10).astype(int)
           profile['age'].value_counts()
```

```
Out[1506]: 50    3485
           60    2952
           40    2269
           70    1757
           30    1503
           20    1353
           80     821
           90     252
           10     205
           Name: age, dtype: int64

In [1507]: profile['age'].unique()

Out[1507]: array([50, 70, 60, 20, 40, 30, 90, 80, 10])

In [1508]: profile['income']=((profile['income']/10000).astype(int)).astype(int)
           profile.head()
```

	age	gender	id	income	reg_year
1	50	0	0610b486422d4921ae7d2bf64640c50b	11	2017
3	70	0	78afa995795e4d85b5d9ceeca43f5fef	10	2017
5	60	1	e2127556f4f64592b11af22de27a7932	7	2018
8	60	1	389bc3fa690240e798340f5a15918d5c	5	2018
12	50	1	2eeac8d8feae4a8cad5a6af0499a211d	5	2017

```
In [1509]: profile.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 14597 entries, 1 to 16999
Data columns (total 5 columns):
age         14597 non-null int64
gender      14597 non-null int64
id          14597 non-null object
income      14597 non-null int64
reg_year    14597 non-null int64
dtypes: int64(4), object(1)
memory usage: 684.2+ KB


In [1510]: sorted(profile['income'].unique())

Out[1510]: [3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

In [1511]: # Extract list values from income data
           # profile = extract_from_iterable_col(profile, 'income', unique_values=['3','4','5','

In [1512]: # rename column id to person_id
           profile = profile.rename(columns = {'id':'person_id'})
           profile.head()
```

Out[1512]:

	age	gender	person_id	income	reg_year
1	50	0	0610b486422d4921ae7d2bf64640c50b	11	2017
3	70	0	78afa995795e4d85b5d9ceeca43f5fef	10	2017
5	60	1	e2127556f4f64592b11af22de27a7932	7	2018
8	60	1	389bc3fa690240e798340f5a15918d5c	5	2018
12	50	1	2eeac8d8feae4a8cad5a6af0499a211d	5	2017

```
In [1513]: profile.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 14597 entries, 1 to 16999
Data columns (total 5 columns):
age          14597 non-null int64
gender       14597 non-null int64
person_id    14597 non-null object
income       14597 non-null int64
reg_year     14597 non-null int64
dtypes: int64(4), object(1)
memory usage: 684.2+ KB
```

transcript

```
In [1514]: # Extract dictionary values from value data
           transcript = extract_from_iterable_col(transcript, 'value', unique_keys=['amount', 'o
           transcript.head()
```

Out[1514]:

	event	person	time	value_amount	\
0	offer received	78afa995795e4d85b5d9ceeca43f5fef	0	NaN	
1	offer received	a03223e636434f42ac4c3df47e8bac43	0	NaN	
2	offer received	e2127556f4f64592b11af22de27a7932	0	NaN	
3	offer received	8ec6ce2a7e7949b1bf142def7d0e0586	0	NaN	
4	offer received	68617ca6246f4fbc85e91a2a49552598	0	NaN	

	value_offer_id	value_reward
0	9b98b8c7a33c4b65b9aebfe6a799e6d9	NaN
1	0b1e1539f2cc45b7b9fa7c272da2e1d7	NaN
2	2906b810c7d4411798c6938adc9daaa5	NaN
3	fafdcd668e3743c1bb461111dcafc2a4	NaN
4	4d5c57ea9a6940dd891ad53e9dbe8da0	NaN

```
In [1515]: x=transcript.groupby(['value_offer_id','event'])['time'].count().reset_index(name ='t
           x=pd.pivot_table(x, index=["value_offer_id"], columns=["event"], values=["total"], ag
           x.head(10)
```

Out[1515]:

	total		
event	offer completed	offer received	offer viewed
value_offer_id			
0b1e1539f2cc45b7b9fa7c272da2e1d7	3420.0	7668.0	2663.0

```
2298d6c36e964ae4a3e7e9706d1fb8c2          5156.0         7646.0       7337.0
2906b810c7d4411798c6938adc9daaa5          4017.0         7632.0       4118.0
3f207df678b143eea3cee63160fa8bed             NaN         7617.0       4144.0
4d5c57ea9a6940dd891ad53e9dbe8da0          3331.0         7593.0       7298.0
5a8bc65990b245e5a138643cd4eb9837             NaN         7618.0       6687.0
9b98b8c7a33c4b65b9aebfe6a799e6d9          4354.0         7677.0       4171.0
ae264e3637204a6fb9bb56bc8210ddfd          3688.0         7658.0       6716.0
f19421c1d4aa40978ebb69ca19b0e20d          4296.0         7571.0       7264.0
fafdcd668e3743c1bb461111dcafc2a4          5317.0         7597.0       7327.0
```

same offer not completed [3f207df678b143eea3cee63160fa8bed - 5a8bc65990b245e5a138643cd4eb9837]

same offer completed more than viewed [0b1e1539f2cc45b7b9fa7c272da2e1d7 - 9b98b8c7a33c4b65b9aebfe6a799e6d9]

In [1516]:
```
# excluding all events of 'transaction' from our clean_transcript dataset
transcript = transcript[transcript['event'] != 'transaction']

# excluding all events of 'offer received'
transcript = transcript[transcript['event'] != 'offer received']

transcript = extract_from_iterable_col(transcript, 'event', unique_values=['offer vie
transcript.head()
```

Out[1516]:
```
                                  person  time  value_amount  \
12650  389bc3fa690240e798340f5a15918d5c     0           NaN
12651  d1ede868e29245ea91818a903fec04c6     0           NaN
12652  102e9454054946fda62242d2e176fdce     0           NaN
12653  02c083884c7d45b39cc68e1314fec56c     0           NaN
12655  be8a5d1981a2458d90b255ddc7e0d174     0           NaN

                          value_offer_id  value_reward  event_offer viewed  \
12650  f19421c1d4aa40978ebb69ca19b0e20d           NaN                   1
12651  5a8bc65990b245e5a138643cd4eb9837           NaN                   1
12652  4d5c57ea9a6940dd891ad53e9dbe8da0           NaN                   1
12653  ae264e3637204a6fb9bb56bc8210ddfd           NaN                   1
12655  5a8bc65990b245e5a138643cd4eb9837           NaN                   1

       event_offer completed
12650                      0
12651                      0
12652                      0
12653                      0
12655                      0
```

In [1517]:
```
# new transcript
transcript=transcript[['person','value_offer_id','event_offer viewed','event_offer co
```

```
In [1518]: # drop null values
           transcript=transcript.dropna()

In [1519]: # rename columns
           transcript = transcript.rename(columns = {'person':'person_id'})
           transcript = transcript.rename(columns = {'value_offer_id':'offer_id'})
           transcript = transcript.rename(columns = {'event_offer completed':'event_offer_comple
           transcript = transcript.rename(columns = {'event_offer viewed':'event_offer_viewed'})
           transcript.head()

Out[1519]:                              person_id                          offer_id  \
           12650  389bc3fa690240e798340f5a15918d5c  f19421c1d4aa40978ebb69ca19b0e20d
           12651  d1ede868e29245ea91818a903fec04c6  5a8bc65990b245e5a138643cd4eb9837
           12652  102e9454054946fda62242d2e176fdce  4d5c57ea9a6940dd891ad53e9dbe8da0
           12653  02c083884c7d45b39cc68e1314fec56c  ae264e3637204a6fb9bb56bc8210ddfd
           12655  be8a5d1981a2458d90b255ddc7e0d174  5a8bc65990b245e5a138643cd4eb9837

                  event_offer_viewed  event_offer_completed
           12650                   1                      0
           12651                   1                      0
           12652                   1                      0
           12653                   1                      0
           12655                   1                      0

In [1520]: transcript['event_offer_completed'].value_counts()

Out[1520]: 0    57725
           1    33579
           Name: event_offer_completed, dtype: int64
```

Merging the three clean datasets (Portfolio, Profile and Transaction) into ONE Master Clean Dataset

```
In [1521]: inner_join_df= pd.merge(transcript,portfolio, on='offer_id', how='inner')

In [1522]: inner_join_df= pd.merge(inner_join_df, profile, on='person_id', how='inner')

In [1523]: inner_join_df=inner_join_df.drop(['person_id'], axis=1)
           inner_join_df=inner_join_df.drop(['offer_id'], axis=1)

In [1524]: inner_join_df.head()

Out[1524]:    event_offer_viewed  event_offer_completed offer_type  offer_name  \
           0                   1                      0       bogo           8
           1                   0                      1       bogo           8
           2                   1                      0       bogo           8
           3                   0                      1       bogo           8
           4                   1                      0   discount           9
```

```
   offer_type_bogo  offer_type_informational  offer_type_discount  \
0                1                         0                    0
1                1                         0                    0
2                1                         0                    0
3                1                         0                    0
4                0                         0                    1

   channels_web  channels_mobile  channels_social  age  gender  income  \
0             1                1                1   60       1       5
1             1                1                1   60       1       5
2             1                1                1   60       1       5
3             1                1                1   60       1       5
4             1                1                0   60       1       5

   reg_year
0      2018
1      2018
2      2018
3      2018
4      2018
```

In [1525]:
```
inner_join_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 80932 entries, 0 to 80931
Data columns (total 14 columns):
event_offer_viewed          80932 non-null int64
event_offer_completed       80932 non-null int64
offer_type                  80932 non-null object
offer_name                  80932 non-null int64
offer_type_bogo             80932 non-null int64
offer_type_informational    80932 non-null int64
offer_type_discount         80932 non-null int64
channels_web                80932 non-null int64
channels_mobile             80932 non-null int64
channels_social             80932 non-null int64
age                         80932 non-null int64
gender                      80932 non-null int64
income                      80932 non-null int64
reg_year                    80932 non-null int64
dtypes: int64(13), object(1)
memory usage: 9.3+ MB
```

In [1526]:
```
inner_join_df.describe()
```

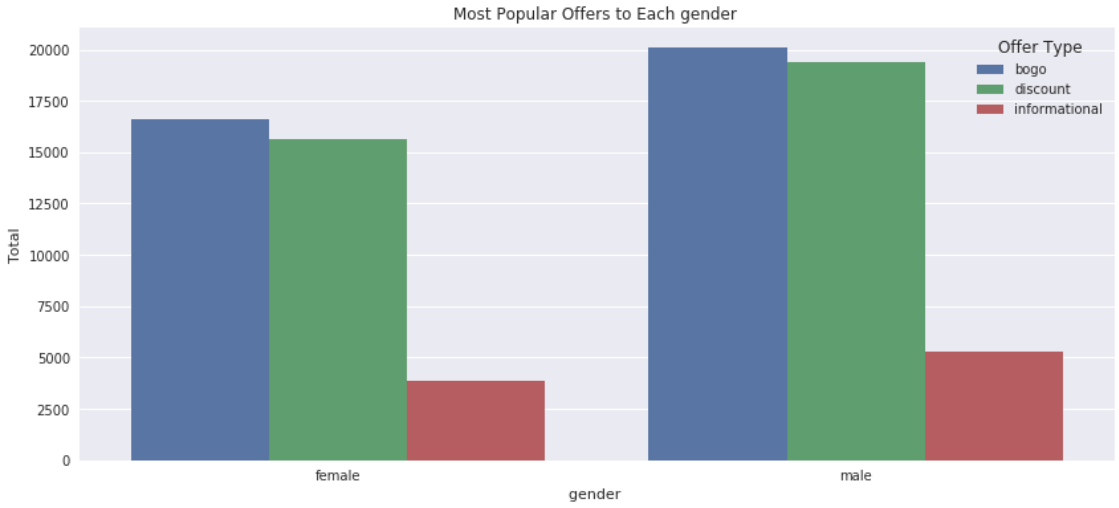
Out[1526]:
```
         event_offer_viewed  event_offer_completed    offer_name  \
count          80932.000000           80932.000000  80932.000000
mean               0.605842               0.394158      4.587135
```

```
std              0.488672              0.488672      2.903200
min              0.000000              0.000000      0.000000
25%              0.000000              0.000000      2.000000
50%              1.000000              0.000000      5.000000
75%              1.000000              1.000000      7.000000
max              1.000000              1.000000      9.000000

       offer_type_bogo  offer_type_informational  offer_type_discount  \
count     80932.000000              80932.000000         80932.000000
mean          0.453455                  0.113527             0.433018
std           0.497832                  0.317238             0.495496
min           0.000000                  0.000000             0.000000
25%           0.000000                  0.000000             0.000000
50%           0.000000                  0.000000             0.000000
75%           1.000000                  0.000000             1.000000
max           1.000000                  1.000000             1.000000

       channels_web  channels_mobile  channels_social           age  \
count  80932.000000     80932.000000     80932.000000  80932.000000
mean       0.812386         0.932289         0.707718     50.551327
std        0.390406         0.251251         0.454814     17.238883
min        0.000000         0.000000         0.000000     10.000000
25%        1.000000         1.000000         0.000000     40.000000
50%        1.000000         1.000000         1.000000     50.000000
75%        1.000000         1.000000         1.000000     60.000000
max        1.000000         1.000000         1.000000     90.000000

             gender        income      reg_year
count  80932.000000  80932.000000  80932.000000
mean       0.552983      6.289169   2016.536982
std        0.497188      2.128978      1.176529
min        0.000000      3.000000   2013.000000
25%        0.000000      5.000000   2016.000000
50%        1.000000      6.000000   2017.000000
75%        1.000000      8.000000   2017.000000
max        1.000000     12.000000   2018.000000
```

Quick Data Analysis on the Master DataSet

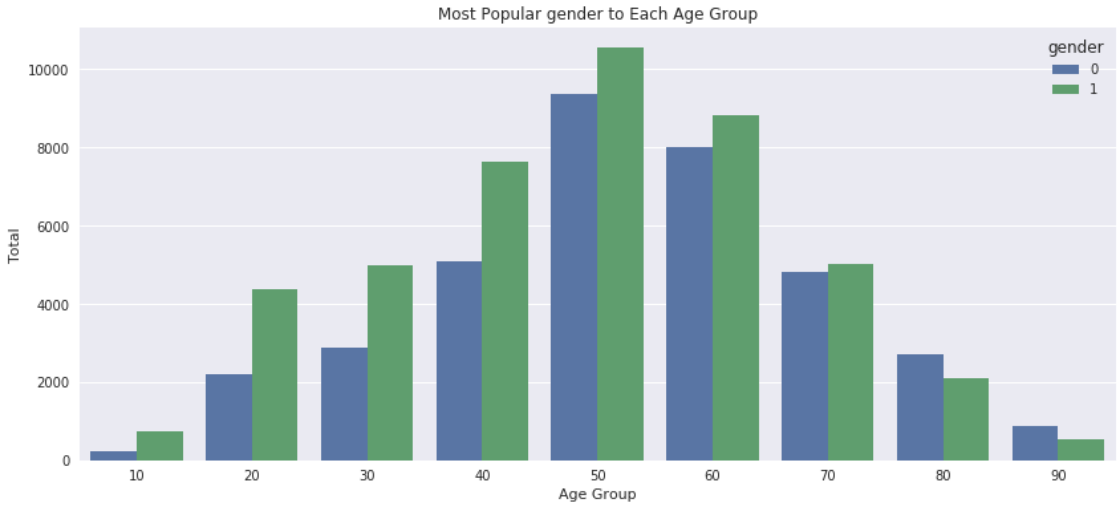
In [1528]:
```
# What is the common offer each age group

plt.figure(figsize=(14, 6))
g = sns.countplot(x="gender", hue="offer_type", data=inner_join_df)
plt.title('Most Popular Offers to Each gender ')
plt.ylabel('Total')
plt.xlabel('gender ')
xlabels = ['female','male']
g.set_xticklabels(xlabels)
```

```
plt.xticks(rotation = 0)
plt.legend(title='Offer Type')
plt.show();
```

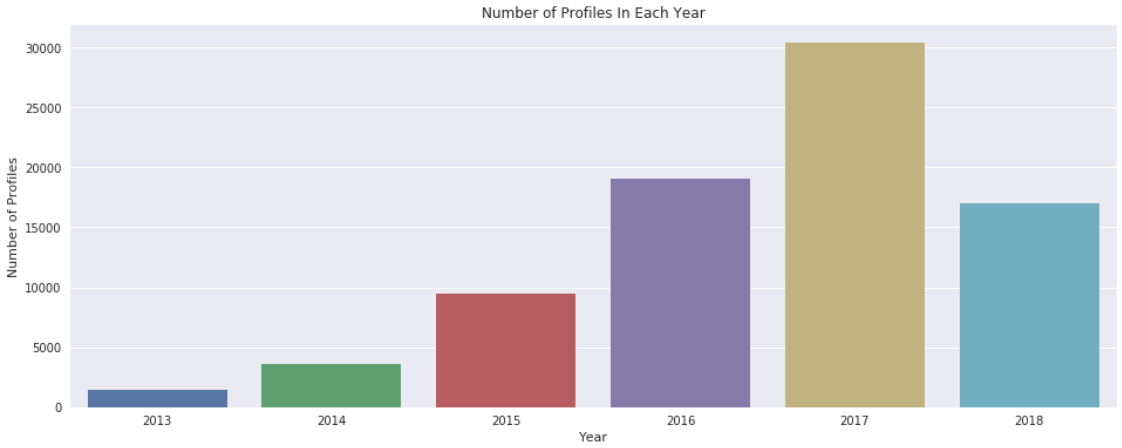


In [1530]:
```
# What is the common offer each age group

plt.figure(figsize=(14, 6))
g = sns.countplot(x="age", hue="offer_type", data=inner_join_df)
plt.title('Most Popular Offers to Each Age Group')
plt.ylabel('Total')
plt.xlabel('Age Group')
plt.xticks(rotation = 0)
plt.legend(title='Offer Type')
plt.show();
```

In [1533]:
```
# What is the common gender each age group

plt.figure(figsize=(14, 6))
g = sns.countplot(x="age", hue="gender", data=inner_join_df)
plt.title('Most Popular gender to Each Age Group')
plt.ylabel('Total')
plt.xlabel('Age Group')
plt.xticks(rotation = 0)
plt.legend(title='gender')
plt.show();
```



In [1535]:
```
# How many registration members got each year?

plt.figure(figsize=(16, 6))
sns.countplot(inner_join_df['reg_year'])
plt.title('Number of Profiles In Each Year')
plt.ylabel('Number of Profiles')
plt.xlabel('Year')
plt.xticks()
plt.show();
```




```
In [1537]: inner_join_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 80932 entries, 0 to 80931
Data columns (total 14 columns):
event_offer_viewed          80932 non-null int64
event_offer_completed       80932 non-null int64
offer_type                  80932 non-null object
offer_name                  80932 non-null int64
offer_type_bogo             80932 non-null int64
offer_type_informational    80932 non-null int64
offer_type_discount         80932 non-null int64
channels_web                80932 non-null int64
channels_mobile             80932 non-null int64
channels_social             80932 non-null int64
age                         80932 non-null int64
gender                      80932 non-null int64
income                      80932 non-null int64
reg_year                    80932 non-null int64
dtypes: int64(13), object(1)
memory usage: 11.8+ MB
```

```
In [1538]: inner_join_df=inner_join_df.drop(['offer_type'], axis=1)
```

4.0.7 model

```
In [1539]: from sklearn.model_selection import train_test_split, cross_val_score
           from sklearn.linear_model import LogisticRegression
           from sklearn.dummy import DummyClassifier
           from sklearn.pipeline import Pipeline
           from sklearn.model_selection import GridSearchCV
           from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
In [1540]: label = inner_join_df['event_offer_completed'].copy()
           train = inner_join_df.drop(['event_offer_completed'], axis=1)

           #Dividing the data into train and test
           X_train, X_test, Y_train, Y_test = train_test_split(train, label,test_size=0.2, rando
           X_train.shape, X_test.shape, Y_train.shape, Y_test.shape

Out[1540]: ((64745, 12), (16187, 12), (64745,), (16187,))

In [1541]: train.head()

Out[1541]:    event_offer_viewed  offer_name  offer_type_bogo  offer_type_informational  \
           0                   1           8                1                         0
           1                   0           8                1                         0
           2                   1           8                1                         0
           3                   0           8                1                         0
           4                   1           9                0                         0

              offer_type_discount  channels_web  channels_mobile  channels_social  age  \
           0                    0             1                1                1   60
           1                    0             1                1                1   60
           2                    0             1                1                1   60
           3                    0             1                1                1   60
           4                    1             1                1                0   60

              gender  income  reg_year
           0       1       5      2018
           1       1       5      2018
           2       1       5      2018
           3       1       5      2018
           4       1       5      2018

In [1542]: X_train.head()

Out[1542]:        event_offer_viewed  offer_name  offer_type_bogo  \
           75830                   1           2                0
           33471                   1           5                0
           22601                   1           7                0
           4857                    0           9                0
           1130                    1           8                1

                  offer_type_informational  offer_type_discount  channels_web  \
           75830                         1                    0             1
           33471                         0                    1             1
           22601                         1                    0             0
           4857                          0                    1             1
           1130                          0                    0             1

                  channels_mobile  channels_social  age  gender  income  reg_year
```

```
75830                1                0   30       0       4       2017
33471                1                1   70       0       7       2018
22601                1                1   60       1       6       2018
4857                 1                0   50       1       6       2018
1130                 1                1   40       1       8       2017
```

```
In [1543]: X_test.head()
```

```
Out[1543]:        event_offer_viewed  offer_name  offer_type_bogo  \
           71618                   1           5                0
           18680                   1           0                1
           25871                   0           6                0
           80924                   0           4                0
           27844                   1           8                1

                  offer_type_informational  offer_type_discount  channels_web  \
           71618                         0                    1             1
           18680                         0                    0             0
           25871                         0                    1             1
           80924                         0                    1             1
           27844                         0                    0             1

                  channels_mobile  channels_social  age  gender  income  reg_year
           71618                1                1   60       1       6      2015
           18680                1                1   20       1       6      2014
           25871                1                1   20       1       3      2017
           80924                0                0   20       1       7      2016
           27844                1                1   30       0       5      2018
```

```
In [1544]: pd.DataFrame(Y_train)['event_offer_completed'].value_counts()
```

```
Out[1544]: 0    39271
           1    25474
           Name: event_offer_completed, dtype: int64
```

```
In [1545]: pd.DataFrame(Y_test)['event_offer_completed'].value_counts()
```

```
Out[1545]: 0    9761
           1    6426
           Name: event_offer_completed, dtype: int64
```

```
In [1546]: label.value_counts()
```

```
Out[1546]: 0    49032
           1    31900
           Name: event_offer_completed, dtype: int64
```

Support Vector Machine

```
In [1547]: # defining a function to calculate the accuracy for the models we will try below
           def predict_score(model):
               pred = model.predict(X_test)

               # Calculate the absolute errors
               errors = abs(pred - Y_test)

               # Calculate mean absolute percentage error
               mean_APE = 100 * (errors / Y_test)
               accuracy = 100 - np.mean(mean_APE)

               return round(accuracy, 4),pred

In [1548]: from sklearn.tree import DecisionTreeRegressor
           dtr = DecisionTreeRegressor().fit(X_train,Y_train)
           print(f'Accuracy of DTR classifier on training set: {round(dtr.score(X_train, Y_train
           accuracy,pred=predict_score(dtr)
           print(f'Prediction Accuracy: {accuracy}%')

Accuracy of DTR classifier on training set: 100.0%.
Prediction Accuracy: 100.0%


In [1550]: from sklearn.svm import SVC
           svm = SVC(gamma = 'auto').fit(X_train, Y_train)
           print(f'Accuracy of SVM classifier on training set: {round(svm.score(X_train, Y_train
           accuracy,pred=predict_score(svm)
           print(f'Prediction Accuracy: {accuracy}%')

Accuracy of SVM classifier on training set: 100.0%.
Prediction Accuracy: 100.0%


In [1551]: pd.DataFrame(pred)[0].value_counts()

Out[1551]: 0    9761
           1    6426
           Name: 0, dtype: int64
```

4.0.8 Model Evaluation

The above table, shows the accuracy score related with using different models of supervised learning. As presented on the table, we had 100% accuracy in both training and testing .

4.0.9 Review Questions:

- What is the predict for event when have gender, age group, income group and offer type will complet offer or not in 2018 year?

```
In [1566]: X_test_2018=X_test.copy()
           X_test_2018['event_offer_completed']=pred

           X_test_2018=X_test_2018[X_test_2018['reg_year']==2018]

           X_test_2018.head()
```

Out[1566]:

	event_offer_viewed	offer_name	offer_type_bogo	offer_type_informational	offer_type_discount	channels_web	channels_mobile	channels_social	age	gender	income	reg_year	event_offer_completed
4141	0	5	0	0	1	1	1	1	50	0	5	2018	1
66598	0	6	0	0	1	1	1	1	70	1	6	2018	1
38071	0	1	1	0	0	1	1	1	70	0	5	2018	1
48170	0	1	1	0	0	1	1	1	40	1	3	2018	1
78948	0	4	0	0	1	1	0	0	40	0	5	2018	1

```
In [1583]: X_test_2018.shape[0],X_test_2018['event_offer_completed'].sum()
```

Out[1583]: (1009, 1009)

The predict for event will complet offer or not in 2018 year

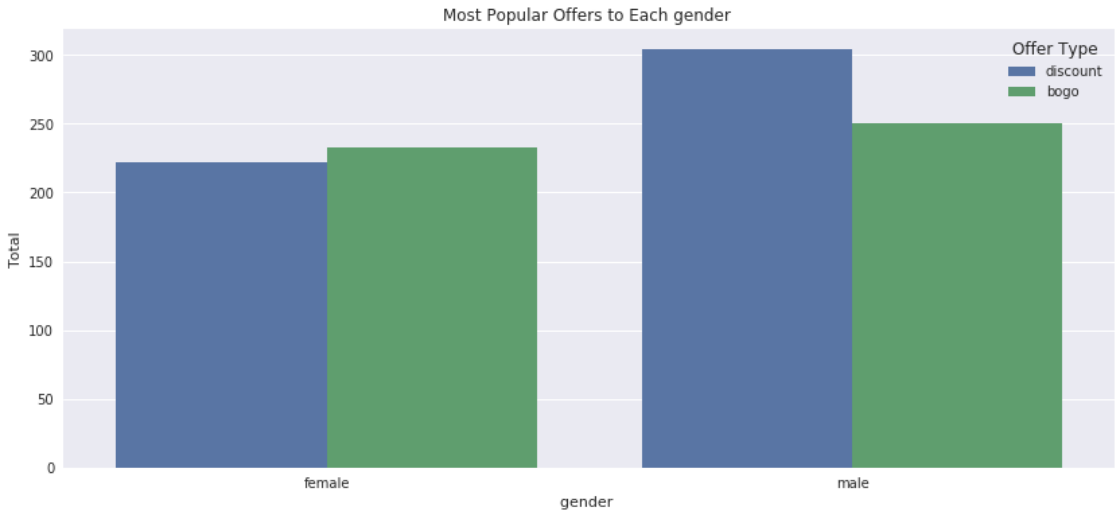
- What is the predict for most offer type in 20018 year ?

```
In [1587]: X_test_2018= pd.merge(X_test_2018, portfolio, on='offer_name', how='inner')
           X_test_2018.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1009 entries, 0 to 1008
```

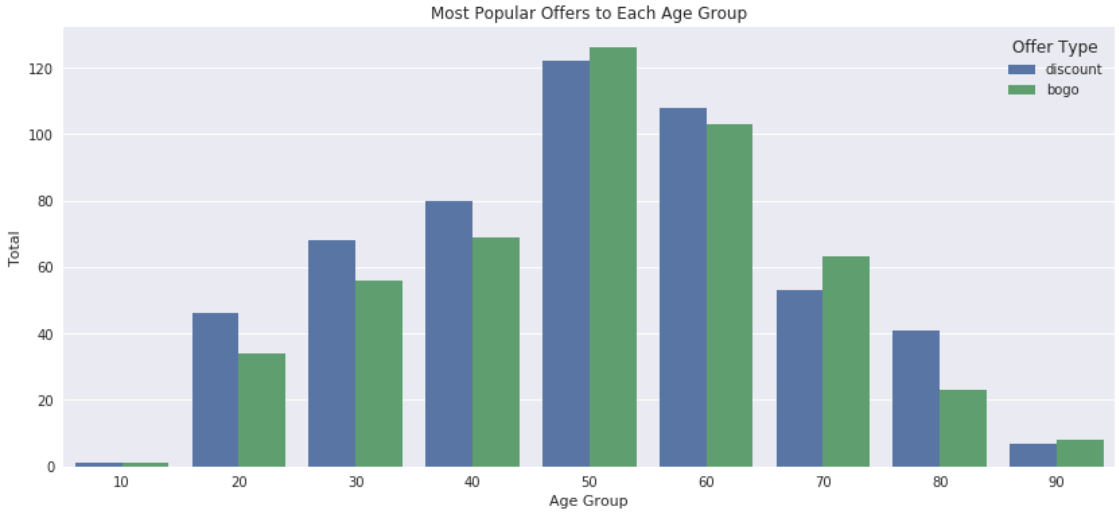
```
Data columns (total 21 columns):
event_offer_viewed            1009 non-null int64
offer_name                    1009 non-null int64
offer_type_bogo_x             1009 non-null int64
offer_type_informational_x    1009 non-null int64
offer_type_discount_x         1009 non-null int64
channels_web_x                1009 non-null int64
channels_mobile_x             1009 non-null int64
channels_social_x             1009 non-null int64
age                           1009 non-null int64
gender                        1009 non-null int64
income                        1009 non-null int64
reg_year                      1009 non-null int64
event_offer_completed         1009 non-null int64
offer_id                      1009 non-null object
offer_type                    1009 non-null object
offer_type_bogo_y             1009 non-null int64
offer_type_informational_y    1009 non-null int64
offer_type_discount_y         1009 non-null int64
channels_web_y                1009 non-null int64
channels_mobile_y             1009 non-null int64
channels_social_y             1009 non-null int64
dtypes: int64(19), object(2)
memory usage: 173.4+ KB
```

In [1588]:
```
# What is the common offer each age group

            plt.figure(figsize=(14, 6))
            g = sns.countplot(x="gender", hue="offer_type", data=X_test_2018)
            plt.title('Most Popular Offers to Each gender ')
            plt.ylabel('Total')
            plt.xlabel('gender ')
            xlabels = ['female','male']
            g.set_xticklabels(xlabels)
            plt.xticks(rotation = 0)
            plt.legend(title='Offer Type')
            plt.show();
```

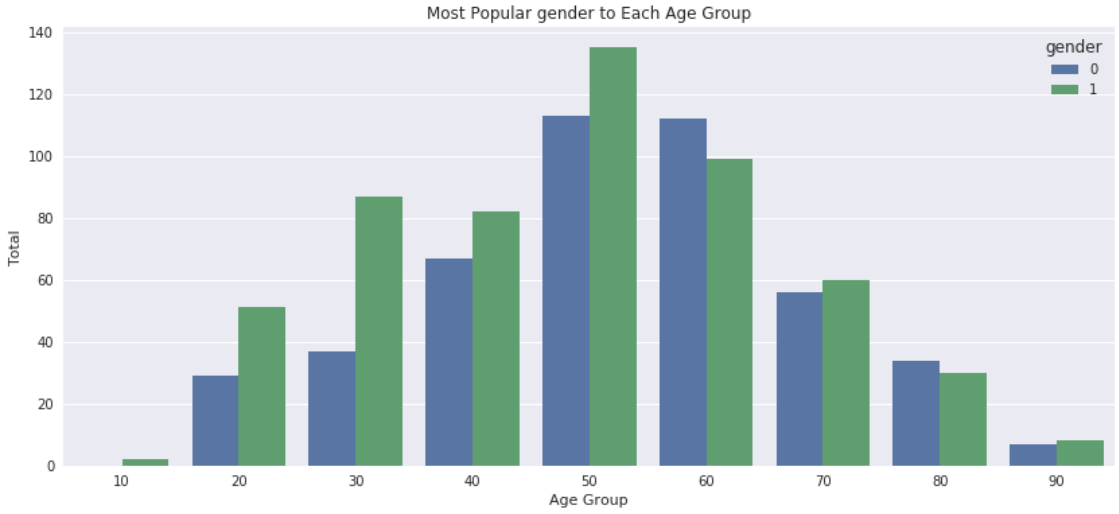


In [1589]: *# What is the common offer each age group*

```
plt.figure(figsize=(14, 6))
g = sns.countplot(x="age", hue="offer_type", data=X_test_2018)
plt.title('Most Popular Offers to Each Age Group')
plt.ylabel('Total')
plt.xlabel('Age Group')
plt.xticks(rotation = 0)
plt.legend(title='Offer Type')
plt.show();
```



In [1586]:
```
# What is the common gender each age group

plt.figure(figsize=(14, 6))
g = sns.countplot(x="age", hue="gender", data=X_test_2018)
plt.title('Most Popular gender to Each Age Group')
plt.ylabel('Total')
plt.xlabel('Age Group')
plt.xticks(rotation = 0)
plt.legend(title='gender')
plt.show();
```



In [1591]:
```
# How many registration members got each year?

plt.figure(figsize=(16, 6))
sns.countplot(X_test_2018['offer_type'])
plt.title('Number of Profiles In Each offer_type')
plt.ylabel('Number of Profiles')
plt.xlabel('offer_type')
plt.xticks()
plt.show();
```

Conclusion All offer will complet in 2018 year, the most offer type is discount more than bogo and informational offer will not complet.

In [ ]: