Reading the Data

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import nltk

pd.options.mode.chained_assignment = None
plt.style.use('classic')
plt.style.use('seaborn-ticks')
plt.style.use('seaborn-darkgrid')
plt.style.use('dark_background')

import os
for dirname, _, filenames in os.walk('/data'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

In [2]:
```
def plot_init():
    return plt.figure(figsize=(14,8))
```

In [3]:
```
portfolio = pd.read_json("data/portfolio.json", lines=True)
profile = pd.read_json("data/profile.json", lines=True)
transcript = pd.read_json("data/transcript.json", lines=True)
```

Data Pre-Processing

portfolio.json

In [4]:
```
portfolio.head()
```

Out[4]:

	reward	channels	difficulty	duration	offer_type	id
0	10	[email, mobile, social]	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd
1	10	[web, email, mobile, social]	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0
2	0	[web, email, mobile]	0	4	informational	3f207df678b143eea3cee63160fa8bed
3	5	[web, email, mobile]	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9
4	5	[web, email]	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7

In [5]:
```
portfolio.isna().sum()
```

Out[5]:
```
reward        0
```

```
channels      0
difficulty    0
duration      0
offer_type    0
id            0
dtype: int64
```

In [6]:
```
portfolio["reward"].unique()
```

Out[6]:
```
array([10,  0,  5,  3,  2], dtype=int64)
```

In [7]:
```
portfolio["channels"]
```

Out[7]:
```
0         [email, mobile, social]
1    [web, email, mobile, social]
2            [web, email, mobile]
3            [web, email, mobile]
4                    [web, email]
5    [web, email, mobile, social]
6    [web, email, mobile, social]
7         [email, mobile, social]
8    [web, email, mobile, social]
9            [web, email, mobile]
Name: channels, dtype: object
```

In [8]:
```
portfolio["difficulty"].unique()
```

Out[8]:
```
array([10,  0,  5, 20,  7], dtype=int64)
```

In [9]:
```
portfolio["duration"].unique()
```

Out[9]:
```
array([ 7,  5,  4, 10,  3], dtype=int64)
```

In [10]:
```
portfolio["offer_type"].unique()
```

Out[10]:
```
array(['bogo', 'informational', 'discount'], dtype=object)
```

In [11]:
```
portfolio["id"].unique()
```

Out[11]:
```
array(['ae264e3637204a6fb9bb56bc8210ddfd',
       '4d5c57ea9a6940dd891ad53e9dbe8da0',
       '3f207df678b143eea3cee63160fa8bed',
       '9b98b8c7a33c4b65b9aebfe6a799e6d9',
       '0b1e1539f2cc45b7b9fa7c272da2e1d7',
       '2298d6c36e964ae4a3e7e9706d1fb8c2',
       'fafdcd668e3743c1bb461111dcafc2a4',
       '5a8bc65990b245e5a138643cd4eb9837',
       'f19421c1d4aa40978ebb69ca19b0e20d',
       '2906b810c7d4411798c6938adc9daaa5'], dtype=object)
```

profile.json

In [12]:
```
profile.head()
```

Out[12]:

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

In [13]:
```
profile.isna().sum()
```

Out[13]:
```
gender              2175
age                    0
id                     0
became_member_on       0
income              2175
dtype: int64
```

In [14]:
```
profile[profile["gender"].isna()]
```

Out[14]:

	gender	age	id	became_member_on	income
0	None	118	68be06ca386d4c31939f3a4f0e3dd783	20170212	NaN
2	None	118	38fe809add3b4fcf9315a9694bb96ff5	20180712	NaN
4	None	118	a03223e636434f42ac4c3df47e8bac43	20170804	NaN
6	None	118	8ec6ce2a7e7949b1bf142def7d0e0586	20170925	NaN
7	None	118	68617ca6246f4fbc85e91a2a49552598	20171002	NaN
...	...	...	...	...	...
16980	None	118	5c686d09ca4d475a8f750f2ba07e0440	20160901	NaN
16982	None	118	d9ca82f550ac4ee58b6299cf1e5c824a	20160415	NaN
16989	None	118	ca45ee1883624304bac1e4c8a114f045	20180305	NaN
16991	None	118	a9a20fa8b5504360beb4e7c8712f8306	20160116	NaN
16994	None	118	c02b10e8752c4d8e9b73f918558531f7	20151211	NaN

2175 rows × 5 columns

In [15]:
```
profile.isna().sum()/len(profile)
```

Out[15]:
```
gender              0.127941
age                 0.000000
id                  0.000000
became_member_on    0.000000
income              0.127941
dtype: float64
```

In [16]:
```
profile = profile.dropna()
```

In [17]:
```
profile["gender"].unique()
```

Out[17]:
```
array(['F', 'M', 'O'], dtype=object)
```

In [18]:
```
profile["age"].unique()
```

Out[18]:
```
array([ 55,  75,  68,  65,  58,  61,  26,  62,  49,  57,  40,  64,  78,
        42,  56,  33,  46,  59,  67,  53,  22,  96,  69,  20,  45,  54,
        39,  41,  79,  66,  29,  44,  63,  36,  76,  77,  30,  51,  27,
        73,  74,  70,  89,  50,  90,  60,  19,  72,  52,  18,  71,  83,
        43,  47,  32,  38,  34,  85,  48,  35,  82,  21,  24,  81,  25,
        37,  23, 100,  28,  84,  80,  87,  86,  94,  31,  88,  95,  93,
        91,  92,  98, 101,  97,  99], dtype=int64)
```

In [19]:
```
sorted(profile["id"].unique())[:10]
```

Out[19]:
```
['0009655768c64bdeb2e877511632db8f',
 '0011e0d4e6b944f998e987f904e8c1e5',
 '0020c2b971eb4e9188eac86d93036a77',
 '0020ccbbb6d84e358d3414a3ff76cffd',
 '003d66b6608740288d6cc97a6903f4f0',
 '00426fe3ffde4c6b9cb9ad6d077a13ea',
 '004b041fbfe44859945daa2c7f79ee64',
 '004c5799adbf42868b9cff0396190900',
 '005500a7188546ff8a767329a2f7c76a',
 '0056df74b63b4298809f0b375a304cf4']
```

In [20]:
```
sorted(profile["became_member_on"].unique())[:10]
```

Out[20]:
```
[20130729,
 20130730,
 20130731,
 20130801,
 20130802,
 20130803,
 20130804,
 20130805,
 20130806,
 20130807]
```

In [21]:
```
sorted(profile["income"].unique())[:10]
```

Out[21]:
```
[30000.0,
 31000.0,
 32000.0,
 33000.0,
 34000.0,
 35000.0,
 36000.0,
 37000.0,
 38000.0,
 39000.0]
```

transcript.json

In [22]:
```
transcript.head()
```

Out[22]:

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

In [23]:
```
transcript.isna().sum()
```

Out[23]:
```
person    0
event     0
value     0
time      0
dtype: int64
```

In [24]:
```
sorted(transcript["person"].unique())[:10]
```

Out[24]:
```
['0009655768c64bdeb2e877511632db8f',
 '00116118485d4dfda04fdbaba9a87b5c',
 '0011e0d4e6b944f998e987f904e8c1e5',
 '0020c2b971eb4e9188eac86d93036a77',
 '0020ccbbb6d84e358d3414a3ff76cffd',
 '003d66b6608740288d6cc97a6903f4f0',
 '00426fe3ffde4c6b9cb9ad6d077a13ea',
 '004b041fbfe44859945daa2c7f79ee64',
 '004c5799adbf42868b9cff0396190900',
 '005500a7188546ff8a767329a2f7c76a']
```

In [25]:
```
transcript["event"].unique()
```

Out[25]:
```
array(['offer received', 'offer viewed', 'transaction', 'offer completed'],
      dtype=object)
```

In [26]:
```
transcript["value"]
```

Out[26]:
```
0         {'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}
1         {'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}
2         {'offer id': '2906b810c7d4411798c6938adc9daaa5'}
3         {'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}
4         {'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}
                              ...
306529                    {'amount': 1.5899999999999999}
```

```
306530                                    {'amount': 9.53}
306531                                    {'amount': 3.61}
306532                       {'amount': 3.5300000000000002}
306533                                    {'amount': 4.05}
Name: value, Length: 306534, dtype: object
```

In [27]:
```
transcript["time"].unique()
```

Out[27]:
```
array([  0,   6,  12,  18,  24,  30,  36,  42,  48,  54,  60,  66,  72,
        78,  84,  90,  96, 102, 108, 114, 120, 126, 132, 138, 144, 150,
       156, 162, 168, 174, 180, 186, 192, 198, 204, 210, 216, 222, 228,
       234, 240, 246, 252, 258, 264, 270, 276, 282, 288, 294, 300, 306,
       312, 318, 324, 330, 336, 342, 348, 354, 360, 366, 372, 378, 384,
       390, 396, 402, 408, 414, 420, 426, 432, 438, 444, 450, 456, 462,
       468, 474, 480, 486, 492, 498, 504, 510, 516, 522, 528, 534, 540,
       546, 552, 558, 564, 570, 576, 582, 588, 594, 600, 606, 612, 618,
       624, 630, 636, 642, 648, 654, 660, 666, 672, 678, 684, 690, 696,
       702, 708, 714], dtype=int64)
```

Exploratory Data Analysis (EDA)

In [28]:
```
profile
```

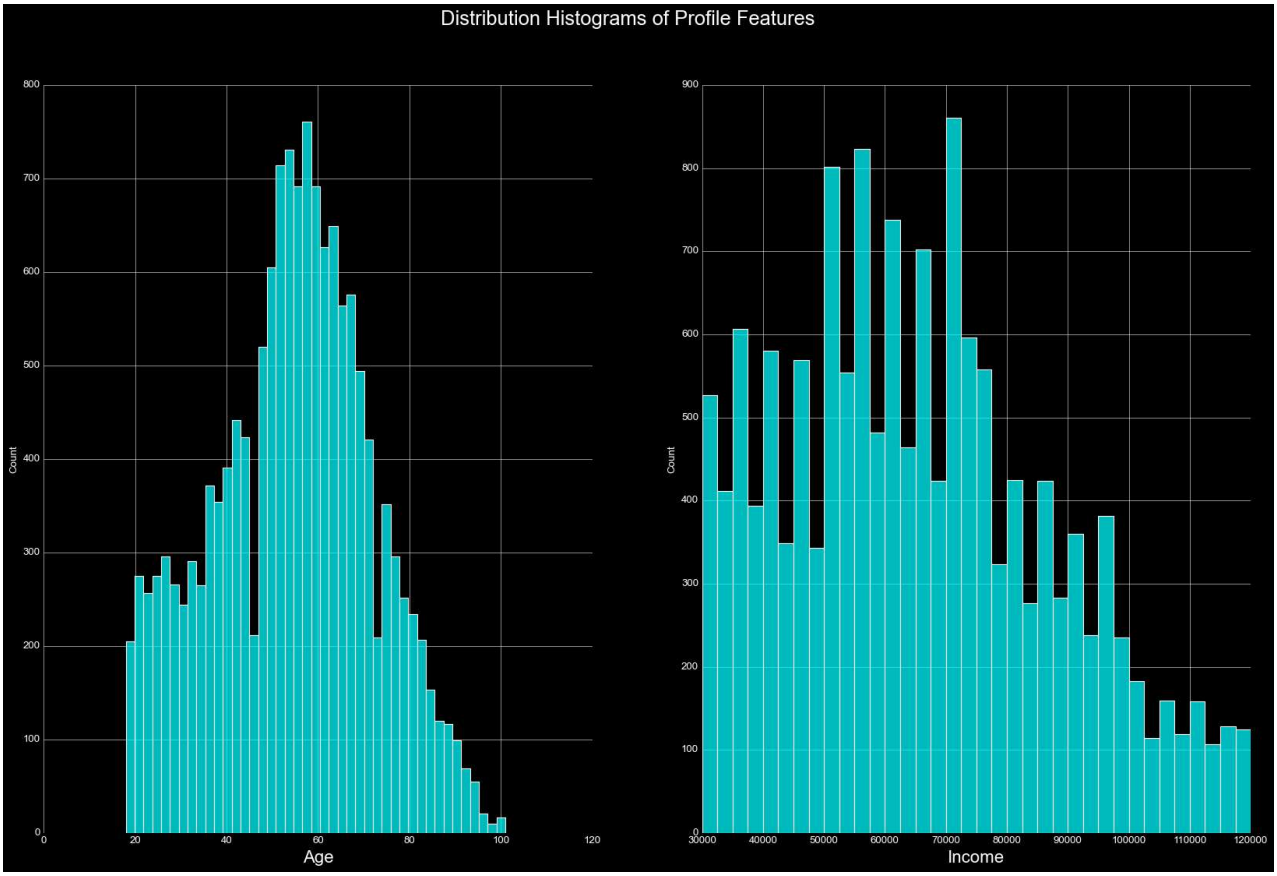
Out[28]:

	gender	age	id	became_member_on	income
1	F	55	0610b486422d4921ae7d2bf64640c50b	20170715	112000.0
3	F	75	78afa995795e4d85b5d9ceeca43f5fef	20170509	100000.0
5	M	68	e2127556f4f64592b11af22de27a7932	20180426	70000.0
8	M	65	389bc3fa690240e798340f5a15918d5c	20180209	53000.0
12	M	58	2eeac8d8feae4a8cad5a6af0499a211d	20171111	51000.0
...	...	...	...	...	...
16995	F	45	6d5f3a774f3d4714ab0c092238f3a1d7	20180604	54000.0
16996	M	61	2cb4f97358b841b9a9773a7aa05a9d77	20180713	72000.0
16997	M	49	01d26f638c274aa0b965d24cefe3183f	20170126	73000.0
16998	F	83	9dc1421481194dcd9400aec7c9ae6366	20160307	50000.0
16999	F	62	e4052622e5ba45a8b96b59aba68cf068	20170722	82000.0

14825 rows × 5 columns

In [29]:
```
fig, axs = plt.subplots(1, 2, figsize=(25,15))
fig.suptitle("Distribution Histograms of Profile Features", size=23)
f1 = sns.histplot(profile["age"], ax=axs[0], color="#00fbff")
f1.set_xlabel("Age", size=20)
f2 = sns.histplot(profile["income"], ax=axs[1], color="#00fbff")
f2.set_xlabel("Income", size=20)
```

Out[29]:
```
Text(0.5, 0, 'Income')
```

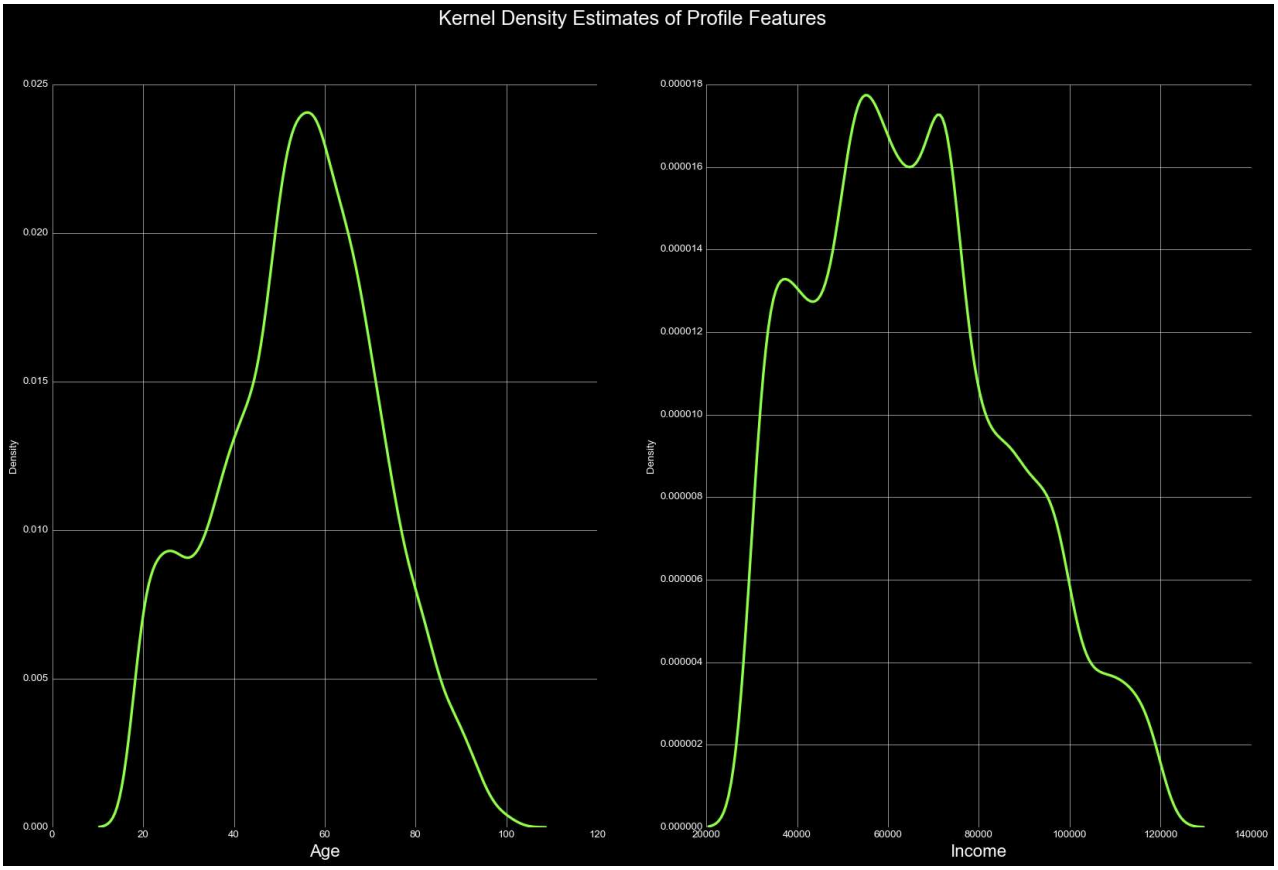


In [30]:

```
fig, axs = plt.subplots(1, 2, figsize=(25,15))
fig.suptitle("Kernel Density Estimates of Profile Features", size=23)
f1 = sns.kdeplot(profile["age"], ax=axs[0], color="#93FF4C",linewidth=3)
f1.set_xlabel("Age", size=20)
f2 = sns.kdeplot(profile["income"], ax=axs[1], color="#93FF4C", linewidth=3)
f2.set_xlabel("Income", size=20)
```

Out[30]:

```
Text(0.5, 0, 'Income')
```

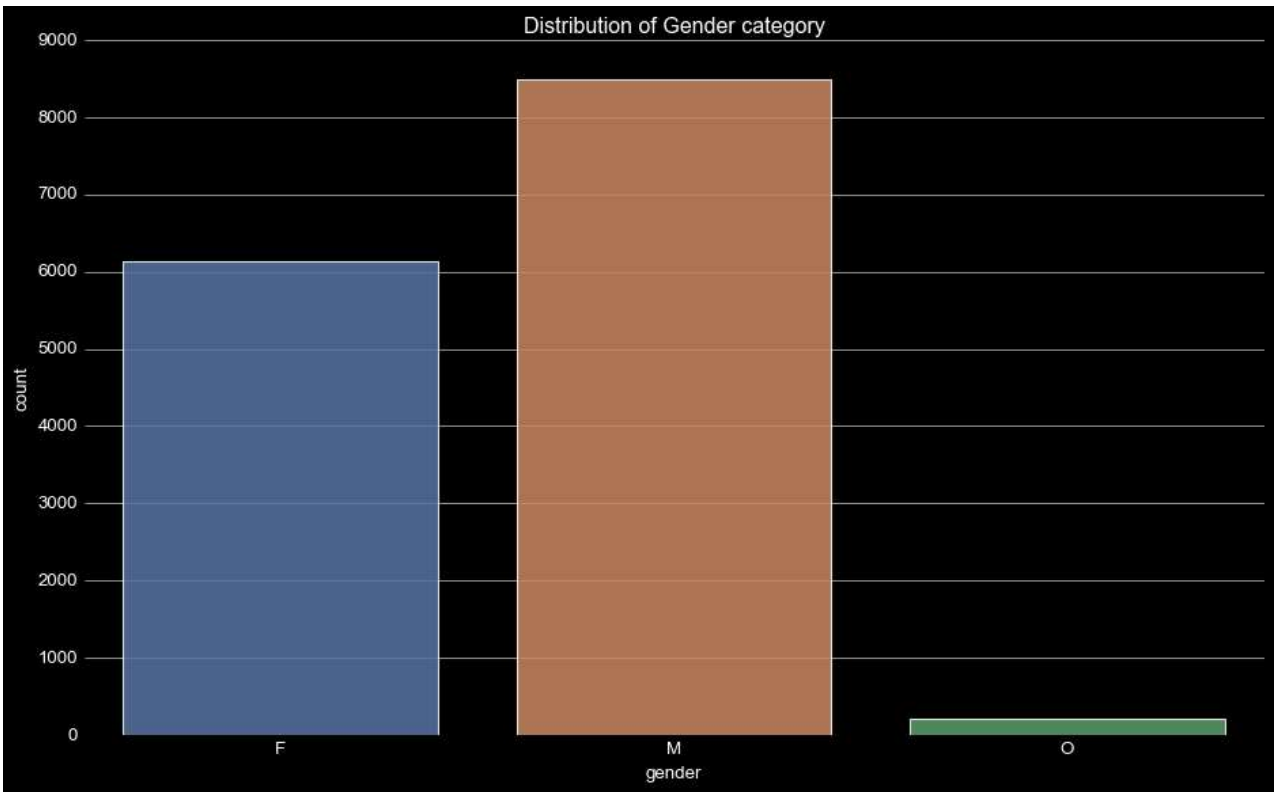


In [31]:

```
plot_init()
sns.countplot(x=profile["gender"], palette="deep", alpha=.85)
plt.title("Distribution of Gender category")
```

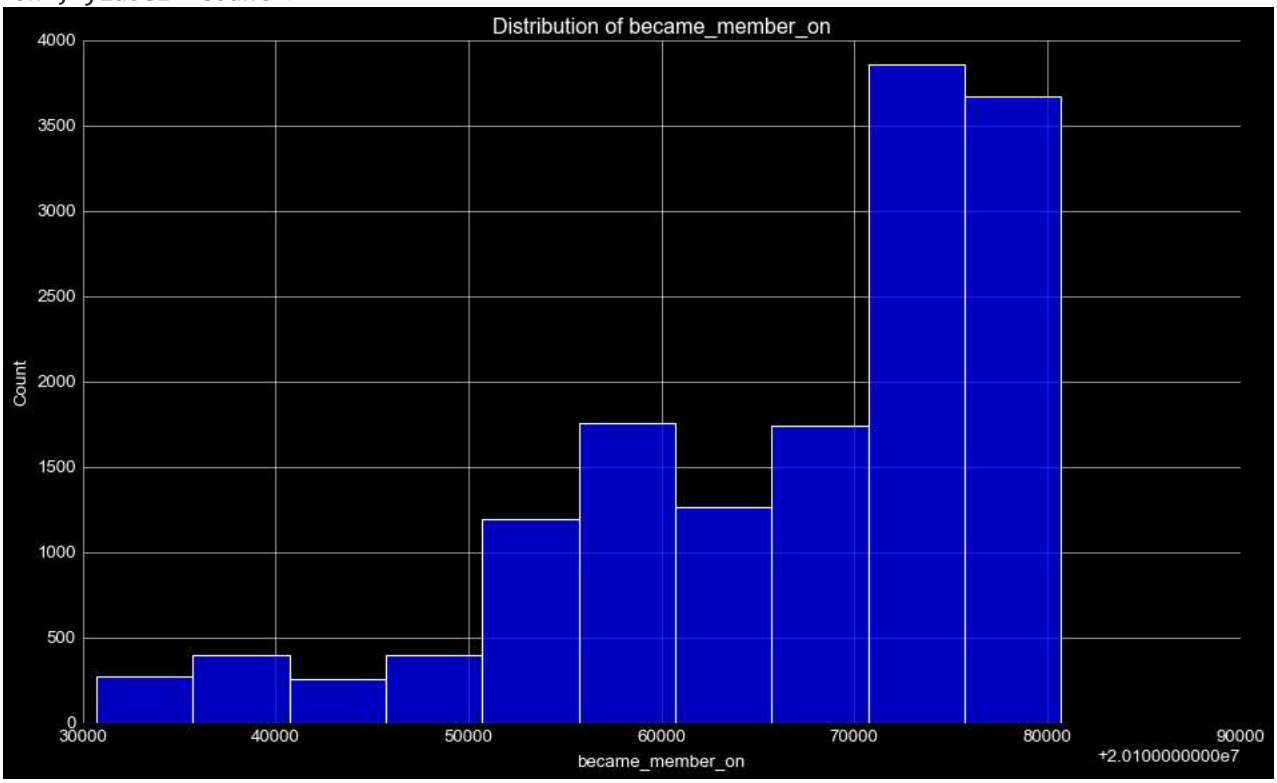
Out[31]:

```
Text(0.5, 1.0, 'Distribution of Gender category')
```



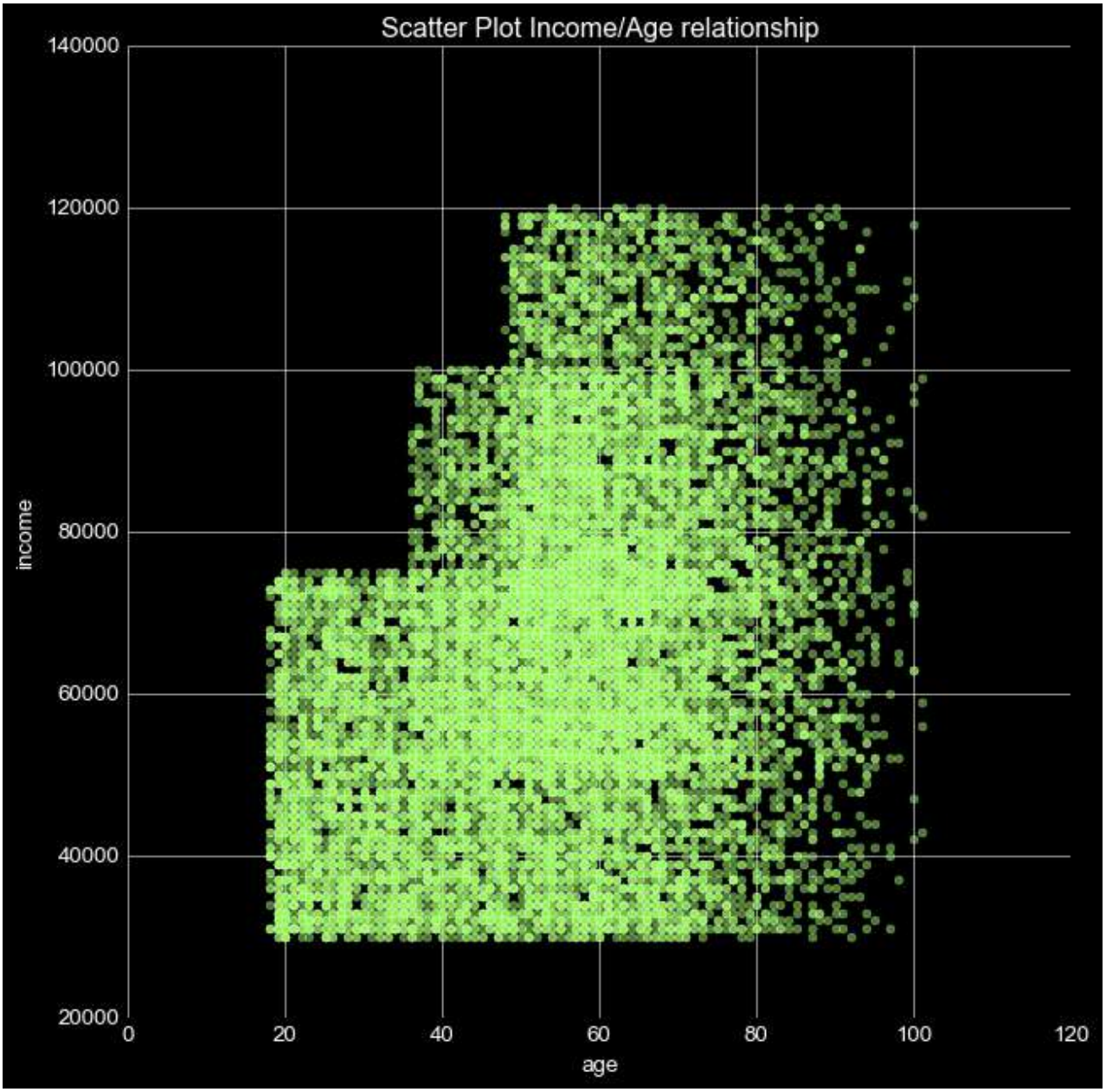
In [32]:
```
plot_init()
plt.title("Distribution of became_member_on")
sns.histplot(profile["became_member_on"], binwidth=5000)
```

Out[32]:
```
<AxesSubplot:title={'center':'Distribution of became_member_on'}, xlabel='became_member_
on', ylabel='Count'>
```



In [33]:
```
plot_init()
plt.figure(figsize=(9,9))
plt.title("Scatter Plot Income/Age relationship")
sns.scatterplot(x="age",y="income",color='#93FF4C',data=profile, alpha=.50)
```

Out[33]:
```
<AxesSubplot:title={'center':'Scatter Plot Income/Age relationship'}, xlabel='age', ylab
el='income'>
<Figure size 1120x640 with 0 Axes>
```

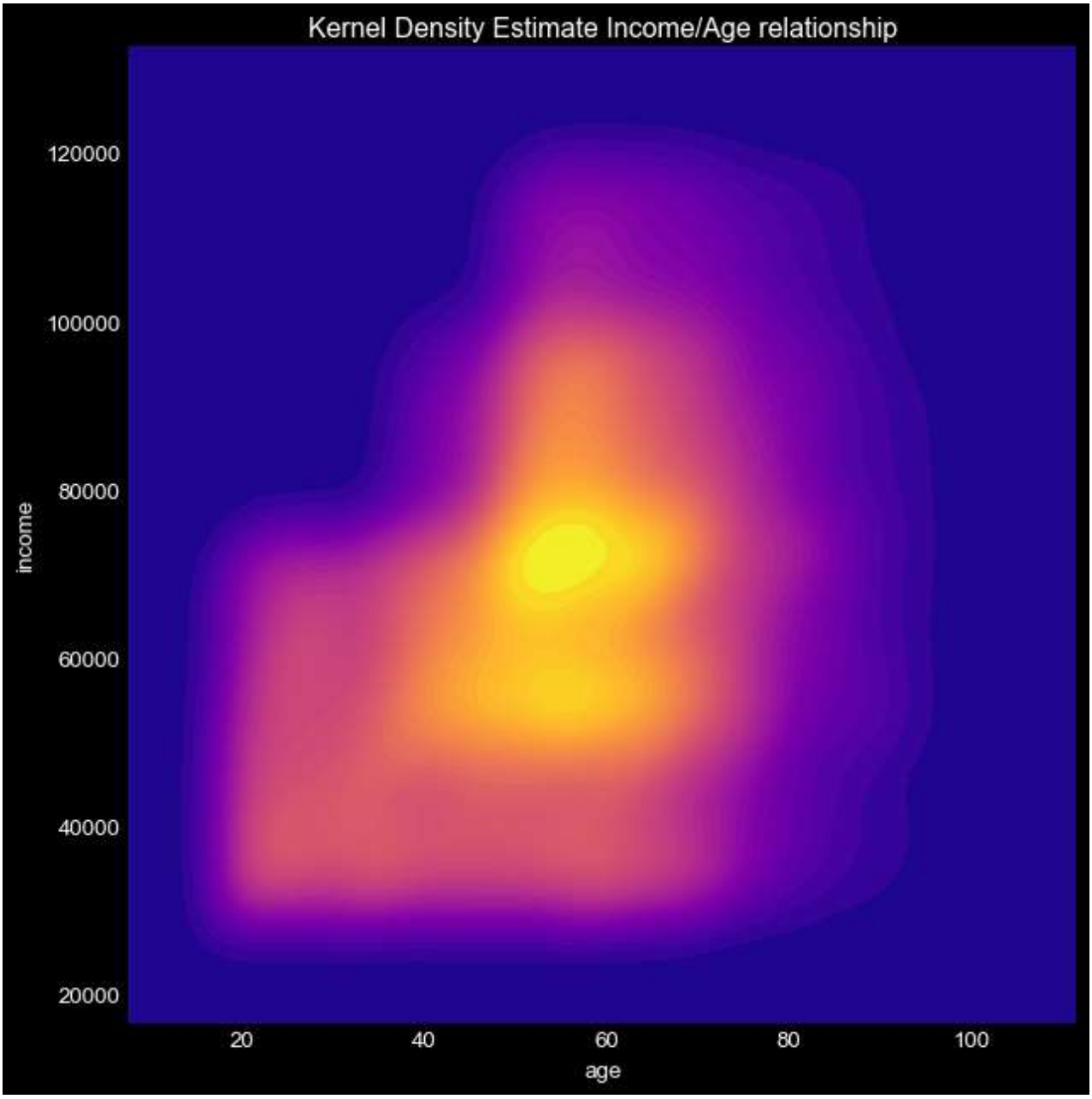


In [34]:

```
plot_init()
plt.figure(figsize=(9,9))
plt.title("Kernel Density Estimate Income/Age relationship")
sns.kdeplot(x="age",y="income",data=profile, cmap="plasma", fill=True,
        thresh=0, levels=50, legend=True)
```

Out[34]:

```
<AxesSubplot:title={'center':'Kernel Density Estimate Income/Age relationship'}, xlabel
='age', ylabel='income'>
<Figure size 1120x640 with 0 Axes>
```

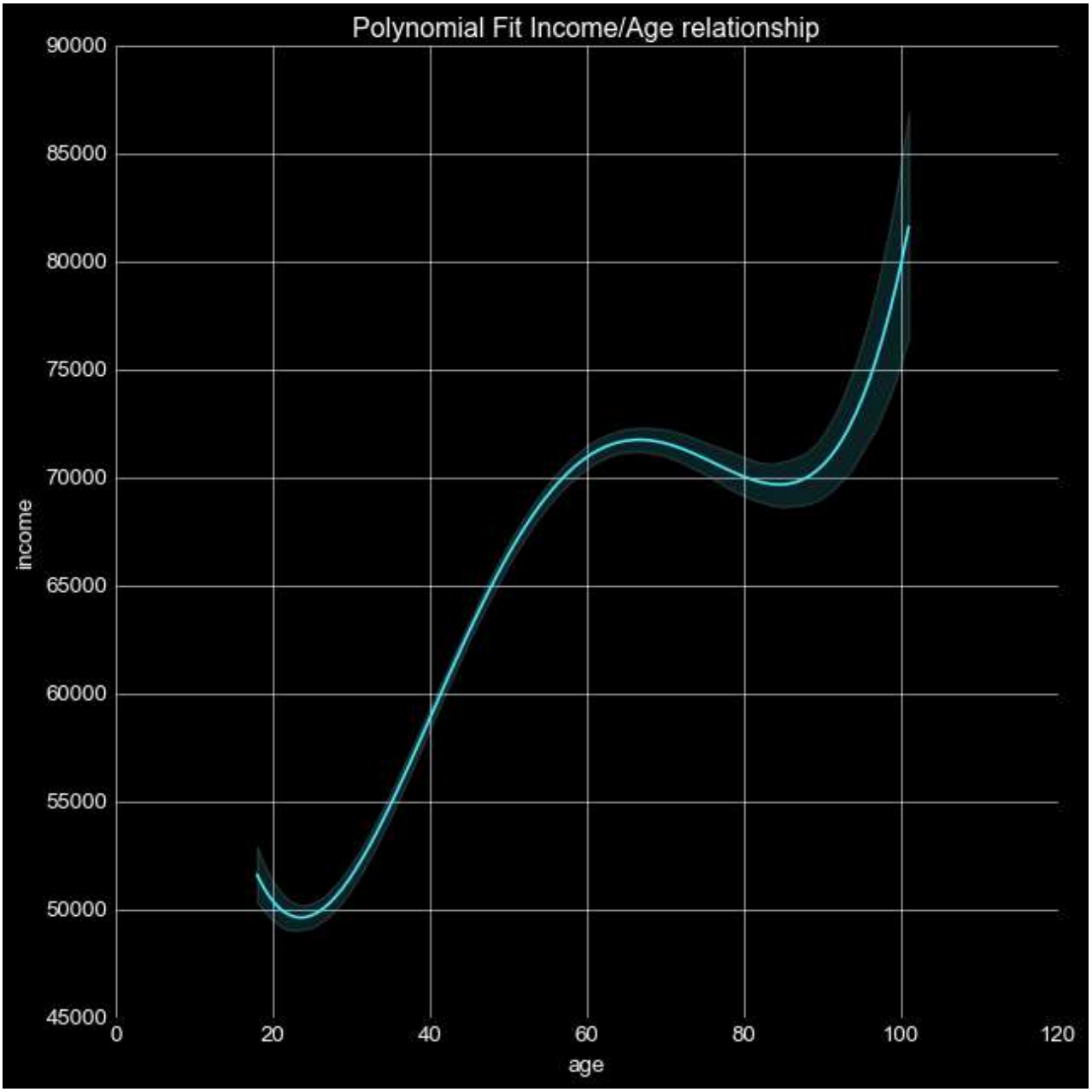


In [35]:
```
np.corrcoef(profile["age"],profile["income"])
```

Out[35]:
```
array([[1.        , 0.30670279],
       [0.30670279, 1.        ]])
```

In [36]:
```
plot_init()
plt.figure(figsize=(9,9))
plt.title("Polynomial Fit Income/Age relationship")
sns.regplot(x="age",y="income", order=4, data=profile, scatter=False, color="#4CE9FF")
```

Out[36]:
```
<AxesSubplot:title={'center':'Polynomial Fit Income/Age relationship'}, xlabel='age', yl
abel='income'>
<Figure size 1120x640 with 0 Axes>
```

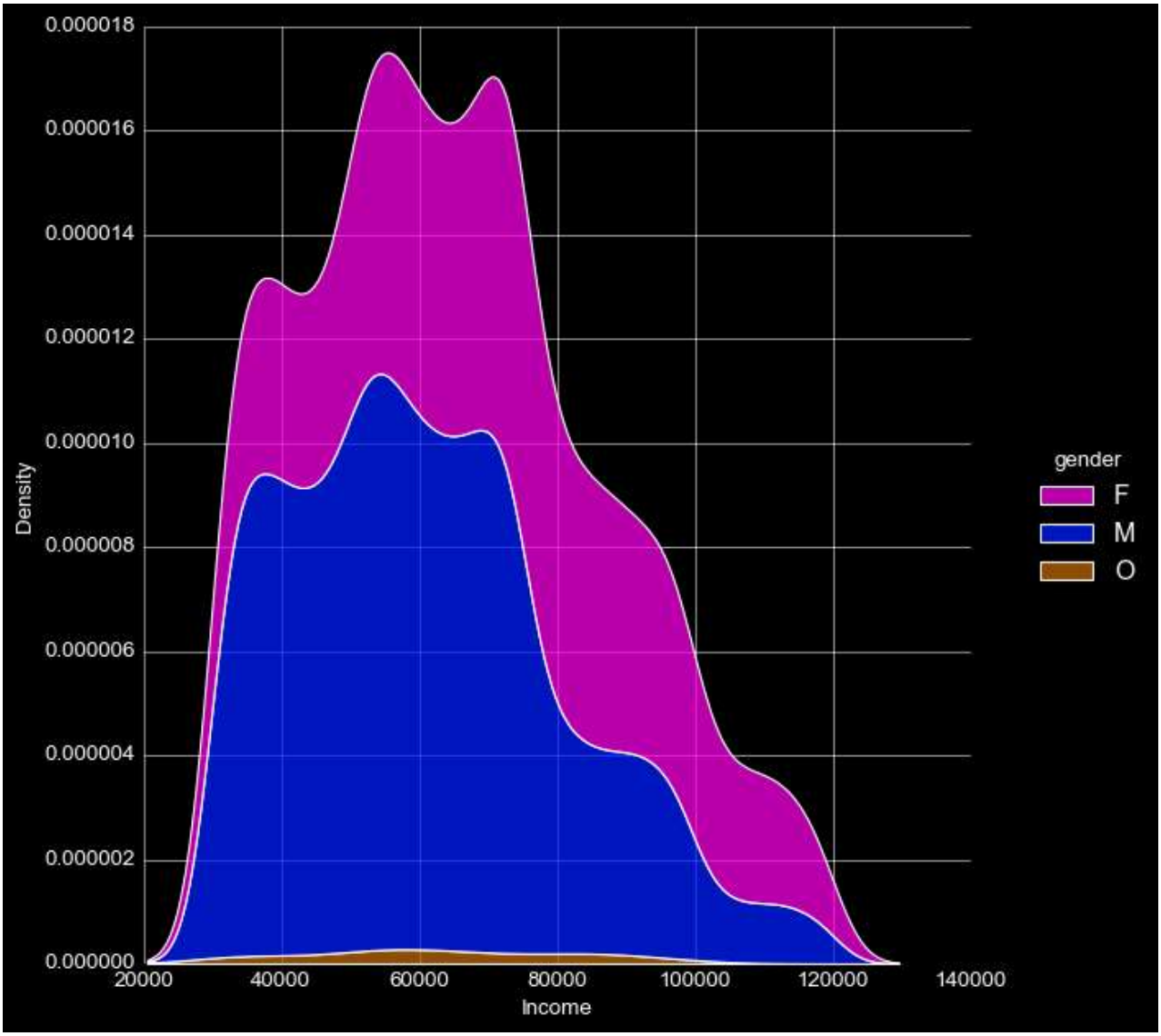


In [37]:

```
d = sns.displot(x="income", multiple="stack", hue="gender",data=profile, height=8, kind
axes = d.axes.flatten()
axes[0].set_xlabel('Income')
```

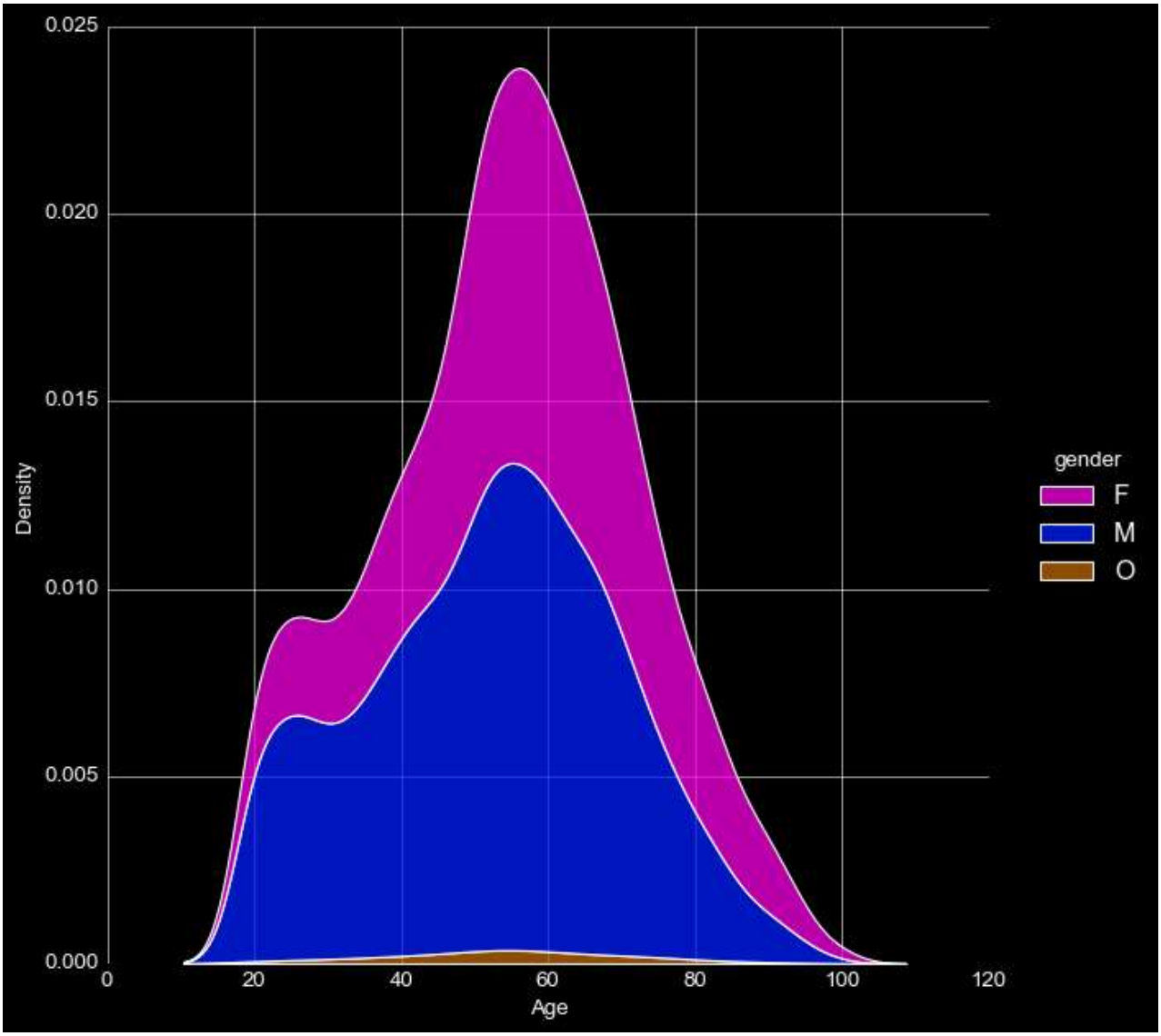
Out[37]:

```
Text(0.5, 8.844444444444436, 'Income')
```



In [38]:
```
d = sns.displot(x="age", multiple="stack", hue="gender",data=profile, height=8, kind="k
axes = d.axes.flatten()
axes[0].set_xlabel('Age')
```

Out[38]:
```
Text(0.5, 8.844444444444436, 'Age')
```

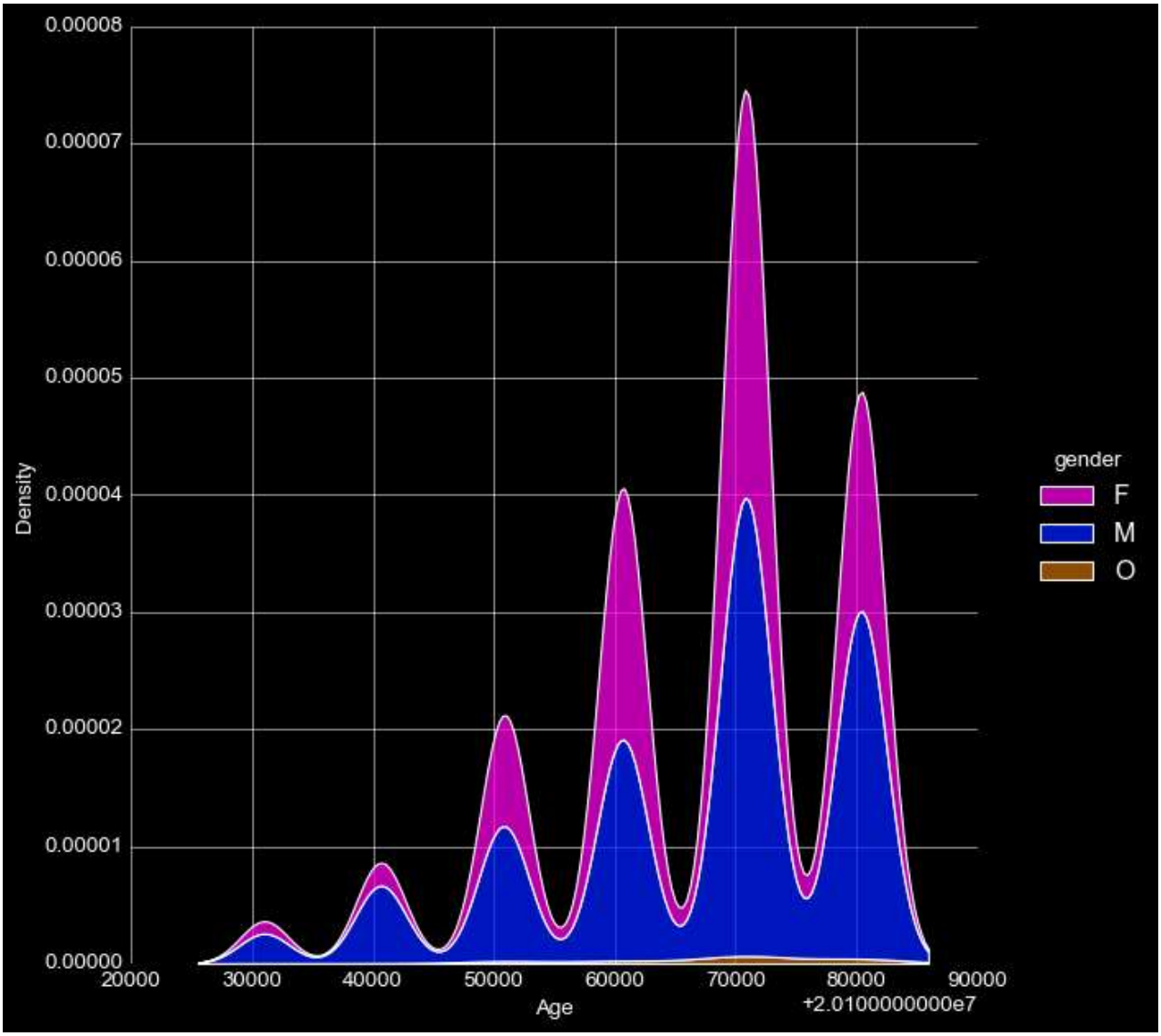


In [39]:

```
d = sns.displot(x="became_member_on", multiple="stack", hue="gender",data=profile, heig
axes = d.axes.flatten()
axes[0].set_xlabel('Age')
```

Out[39]:

```
Text(0.5, 8.844444444444436, 'Age')
```



In [40]:

```
age_cat_array = []
for i in range(len(profile)):
    retr_age = profile["age"].iloc[i]
    if retr_age>=18 and retr_age<=25:
        age_cat_array.append("18-25")
    elif retr_age>=26 and retr_age<=30:
        age_cat_array.append("26-30")
    elif retr_age>=31 and retr_age<=40:
        age_cat_array.append("31-40")
    elif retr_age>=41 and retr_age<=50:
        age_cat_array.append("41-50")
    elif retr_age>=51 and retr_age<=59:
        age_cat_array.append("51-59")
    elif retr_age>=60:
        age_cat_array.append(">60")
```

In [41]:

```
profile["age_group"] = age_cat_array
profile["income"].describe()
```

Out[41]:

```
count     14825.000000
mean      65404.991568
std       21598.299410
```

```
min       30000.000000
25%       49000.000000
50%       64000.000000
75%       80000.000000
max      120000.000000
Name: income, dtype: float64
```

In [42]:

```
income_cat_array = []
for i in range(len(profile)):
    retr_inc = profile["income"].iloc[i]
    if retr_inc>=30000 and retr_inc<=45000:
        income_cat_array.append("30000-45000")
    elif retr_inc>=46000 and retr_inc<=55000:
        income_cat_array.append("46000-55000")
    elif retr_inc>=56000 and retr_inc<=65000:
        income_cat_array.append("56000-65000")
    elif retr_inc>=66000 and retr_inc<=79000:
        income_cat_array.append("66000-75000")
    elif retr_inc>=80000:
        income_cat_array.append(">80000")
    else:
        print(f"ERROR: on value {retr_inc} position {i}")
profile["income_group"] = income_cat_array
profile
```

Out[42]:

	gender	age	id	became_member_on	income	age_group	inc
1	F	55	0610b486422d4921ae7d2bf64640c50b	20170715	112000.0	51-59	
3	F	75	78afa995795e4d85b5d9ceeca43f5fef	20170509	100000.0	>60	
5	M	68	e2127556f4f64592b11af22de27a7932	20180426	70000.0	>60	6
8	M	65	389bc3fa690240e798340f5a15918d5c	20180209	53000.0	>60	4
12	M	58	2eeac8d8feae4a8cad5a6af0499a211d	20171111	51000.0	51-59	4
...	...	...	...	...	...	...	
16995	F	45	6d5f3a774f3d4714ab0c092238f3a1d7	20180604	54000.0	41-50	4
16996	M	61	2cb4f97358b841b9a9773a7aa05a9d77	20180713	72000.0	>60	6
16997	M	49	01d26f638c274aa0b965d24cefe3183f	20170126	73000.0	41-50	6
16998	F	83	9dc1421481194dcd9400aec7c9ae6366	20160307	50000.0	>60	4
16999	F	62	e4052622e5ba45a8b96b59aba68cf068	20170722	82000.0	>60	

14825 rows × 7 columns

In [43]:

```
portfolio
```

Out[43]:

	reward	channels	difficulty	duration	offer_type	id
0	10	[email, mobile, social]	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd

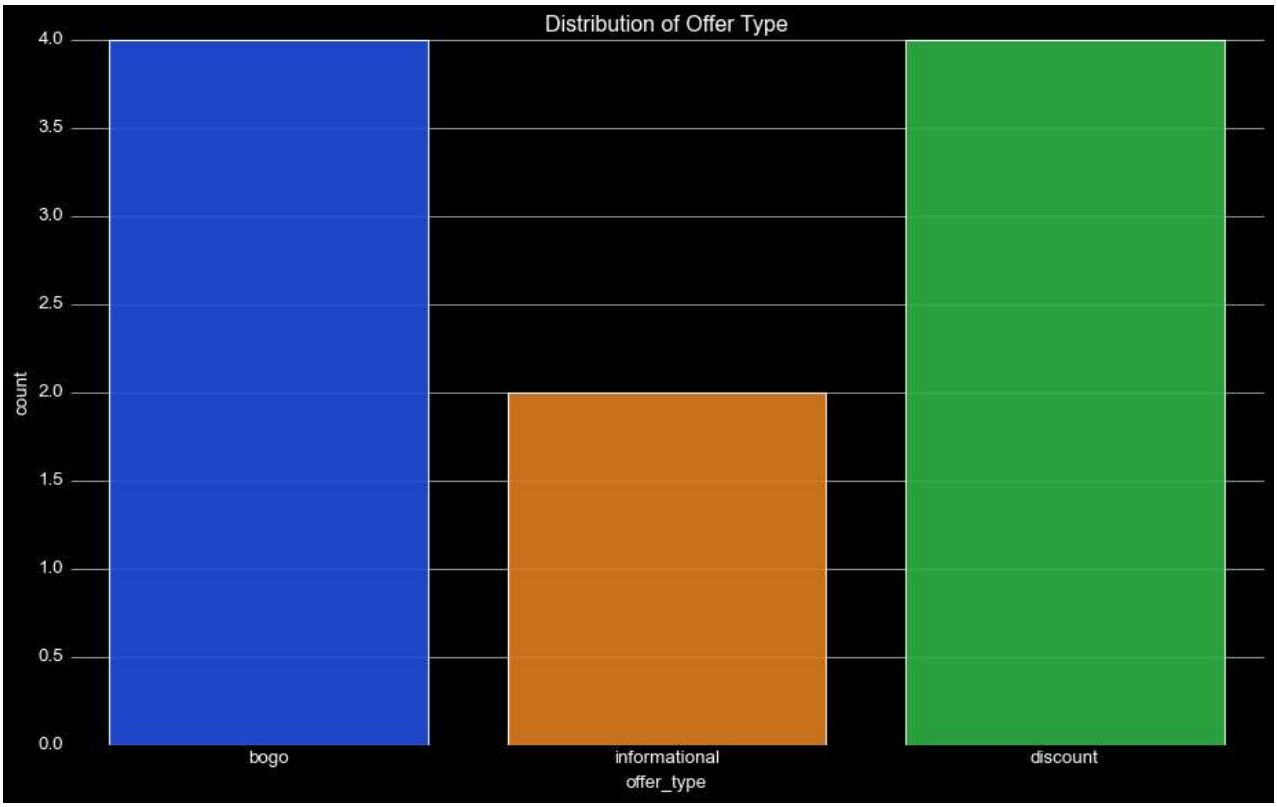
	reward	channels	difficulty	duration	offer_type	id
1	10	[web, email, mobile, social]	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0
2	0	[web, email, mobile]	0	4	informational	3f207df678b143eea3cee63160fa8bed
3	5	[web, email, mobile]	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9
4	5	[web, email]	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7
5	3	[web, email, mobile, social]	7	7	discount	2298d6c36e964ae4a3e7e9706d1fb8c2
6	2	[web, email, mobile, social]	10	10	discount	fafdcd668e3743c1bb461111dcafc2a4
7	0	[email, mobile, social]	0	3	informational	5a8bc65990b245e5a138643cd4eb9837
8	5	[web, email, mobile, social]	5	5	bogo	f19421c1d4aa40978ebb69ca19b0e20d
9	2	[web, email, mobile]	10	7	discount	2906b810c7d4411798c6938adc9daaa5

In [44]:

```
plot_init()
plt.title("Distribution of Offer Type")
sns.countplot(x=portfolio["offer_type"], palette="bright", alpha=.9)
```

Out[44]:

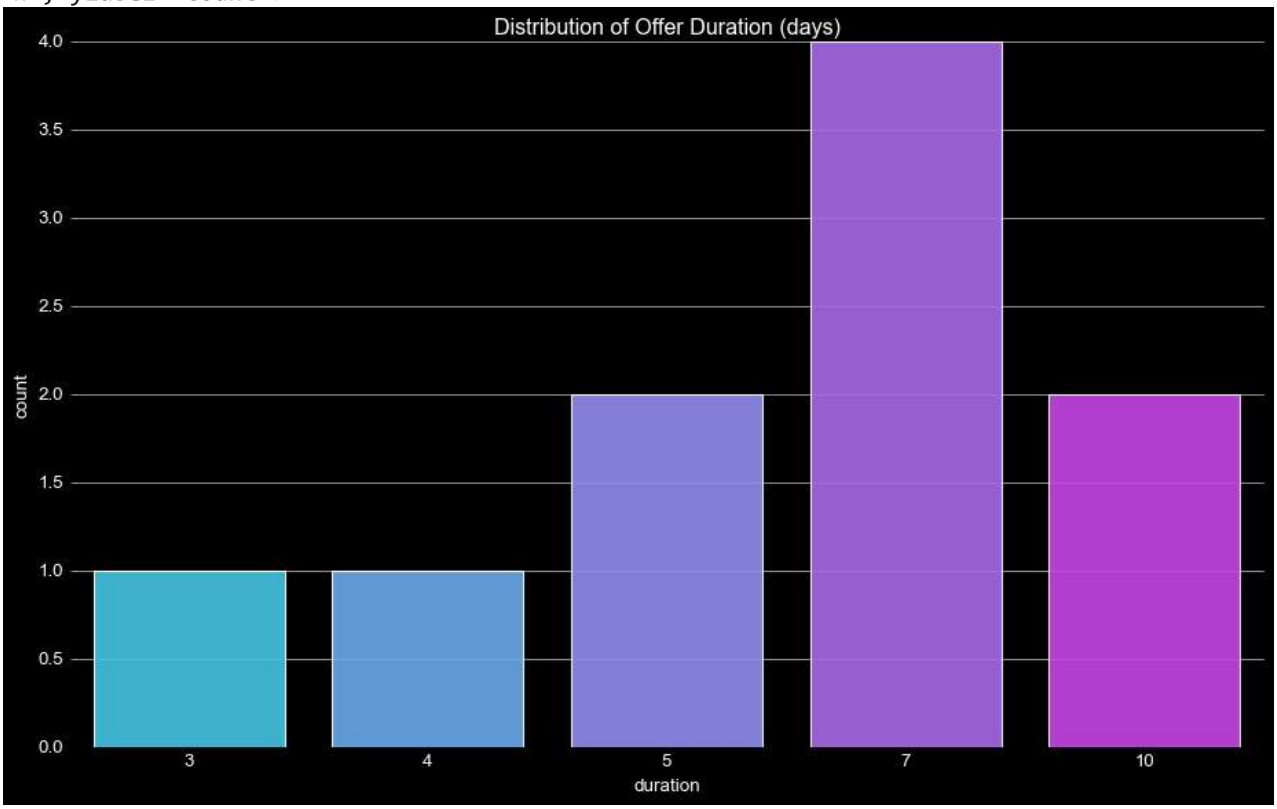
```
<AxesSubplot:title={'center':'Distribution of Offer Type'}, xlabel='offer_type', ylabel
='count'>
```



In [45]:

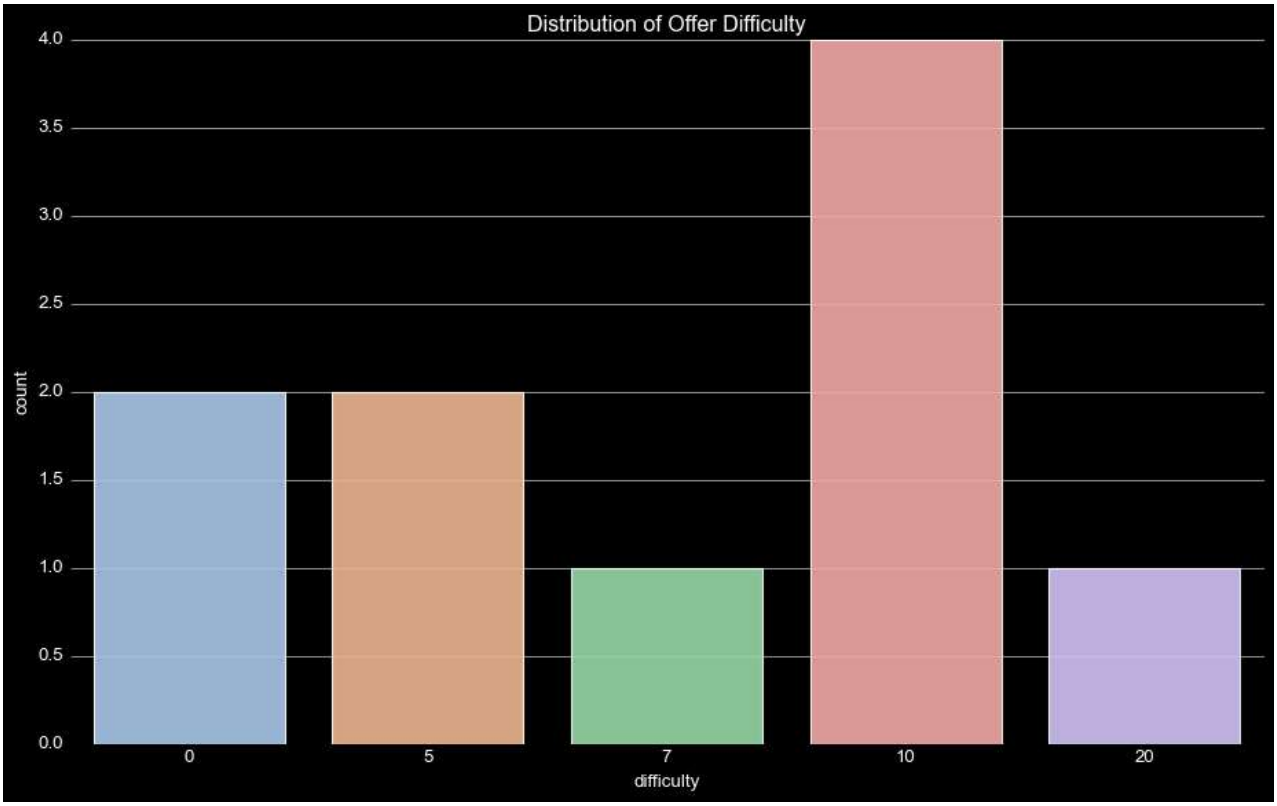
```
plot_init()
plt.title("Distribution of Offer Duration (days)")
sns.countplot(x=portfolio["duration"], palette="cool", alpha=.9)
```

Out[45]:
```
<AxesSubplot:title={'center':'Distribution of Offer Duration (days)'}, xlabel='duratio
n', ylabel='count'>
```



In [46]:
```
plot_init()
plt.title("Distribution of Offer Difficulty")
sns.countplot(x=portfolio["difficulty"], palette="pastel", alpha=.9)
```

Out[46]:
```
<AxesSubplot:title={'center':'Distribution of Offer Difficulty'}, xlabel='difficulty', y
label='count'>
```

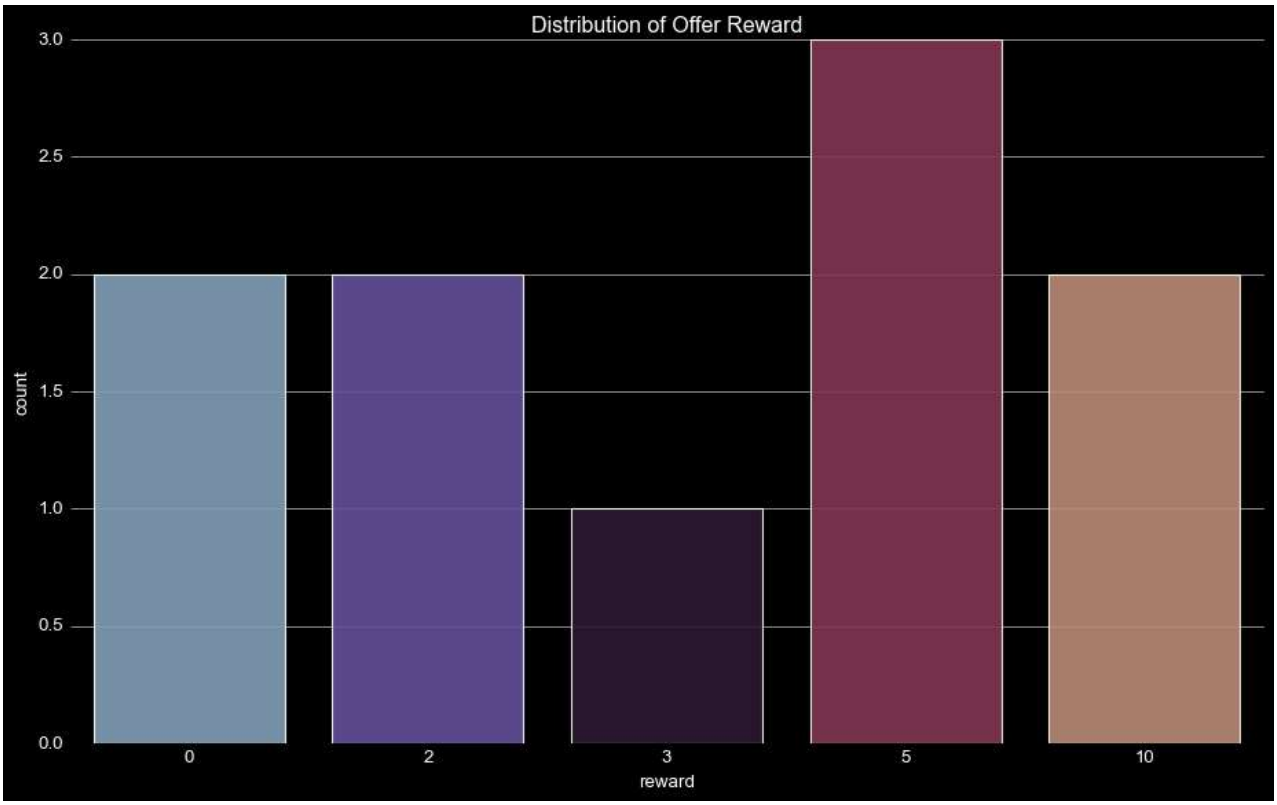


In [47]:

```
plot_init()
plt.title("Distribution of Offer Reward")
sns.countplot(x=portfolio["reward"], palette="twilight", alpha=.9)
```

Out[47]:

```
<AxesSubplot:title={'center':'Distribution of Offer Reward'}, xlabel='reward', ylabel='c
ount'>
```

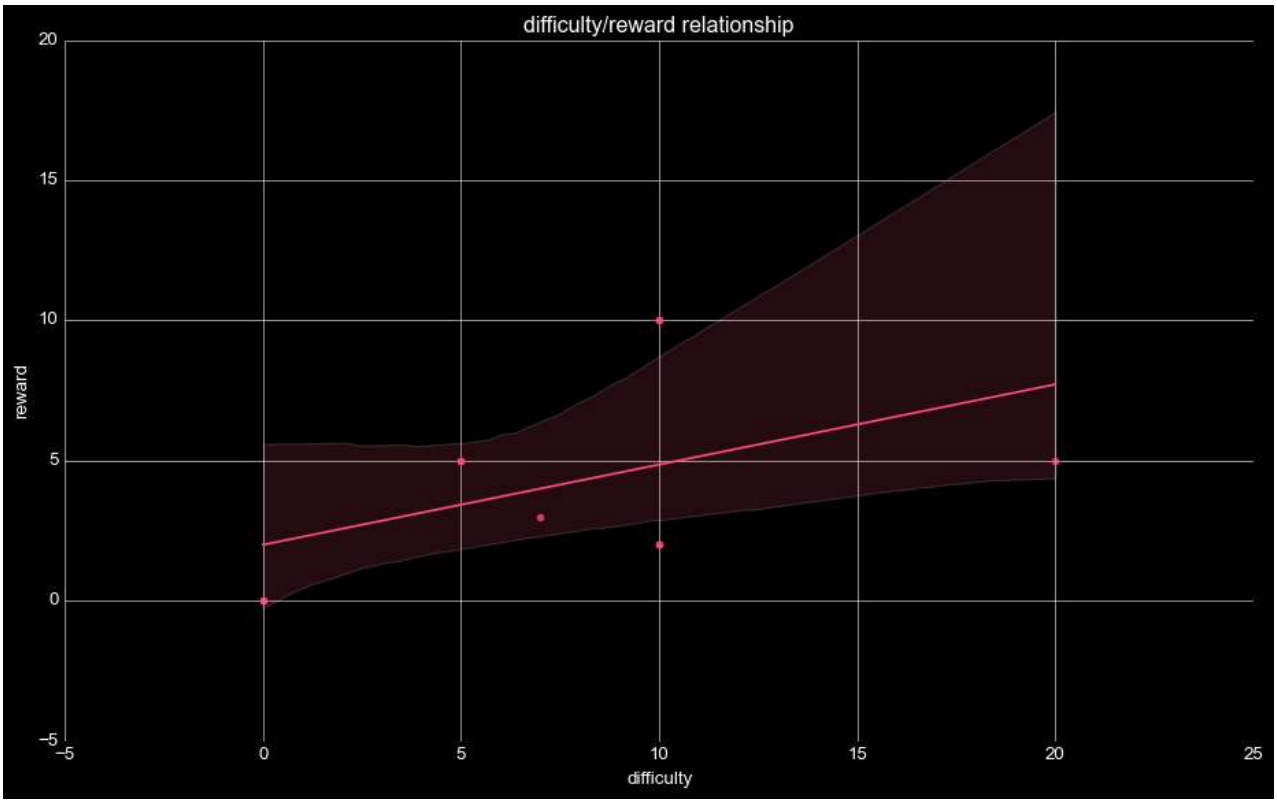


In [48]:

```
plot_init()
```

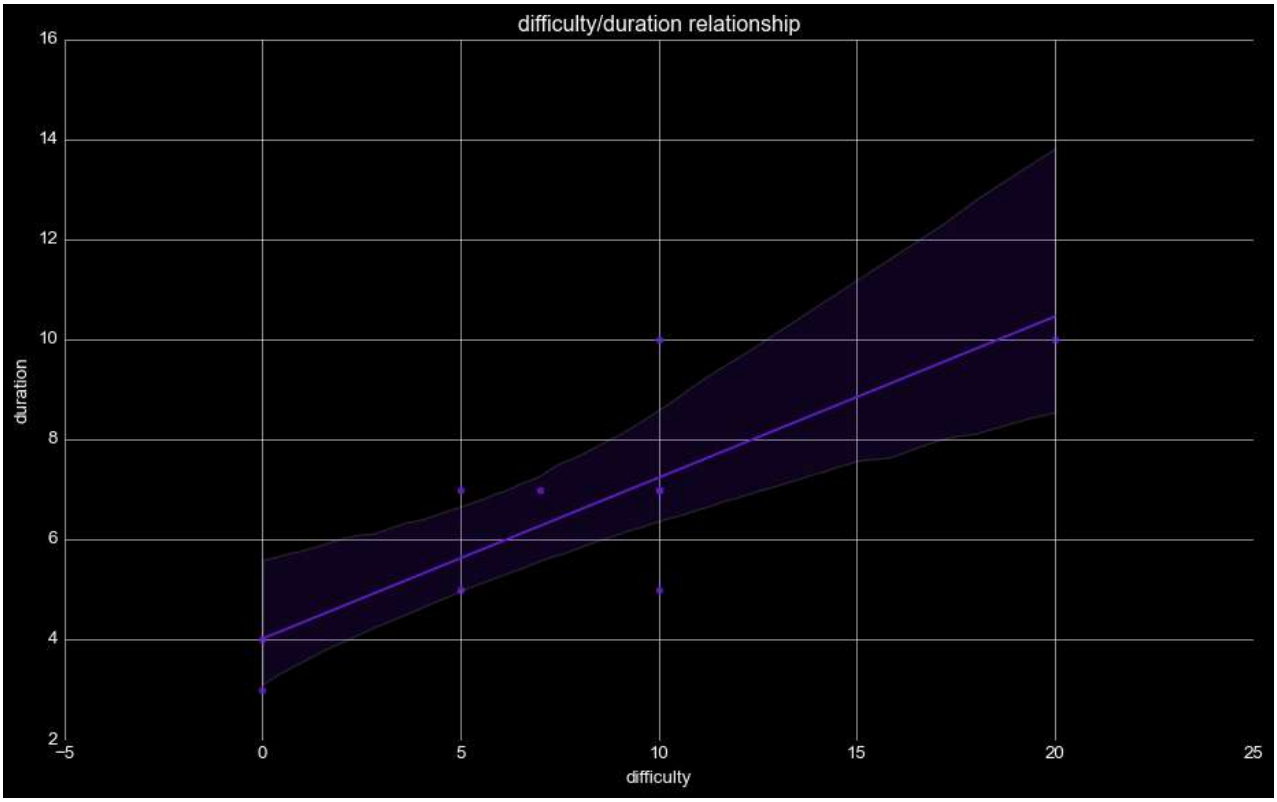
```
plt.title("difficulty/reward relationship")
sns.regplot(x="difficulty",y="reward", data=portfolio, color="#FF4C7A")
```

Out[48]:
```
<AxesSubplot:title={'center':'difficulty/reward relationship'}, xlabel='difficulty', yla
bel='reward'>
```



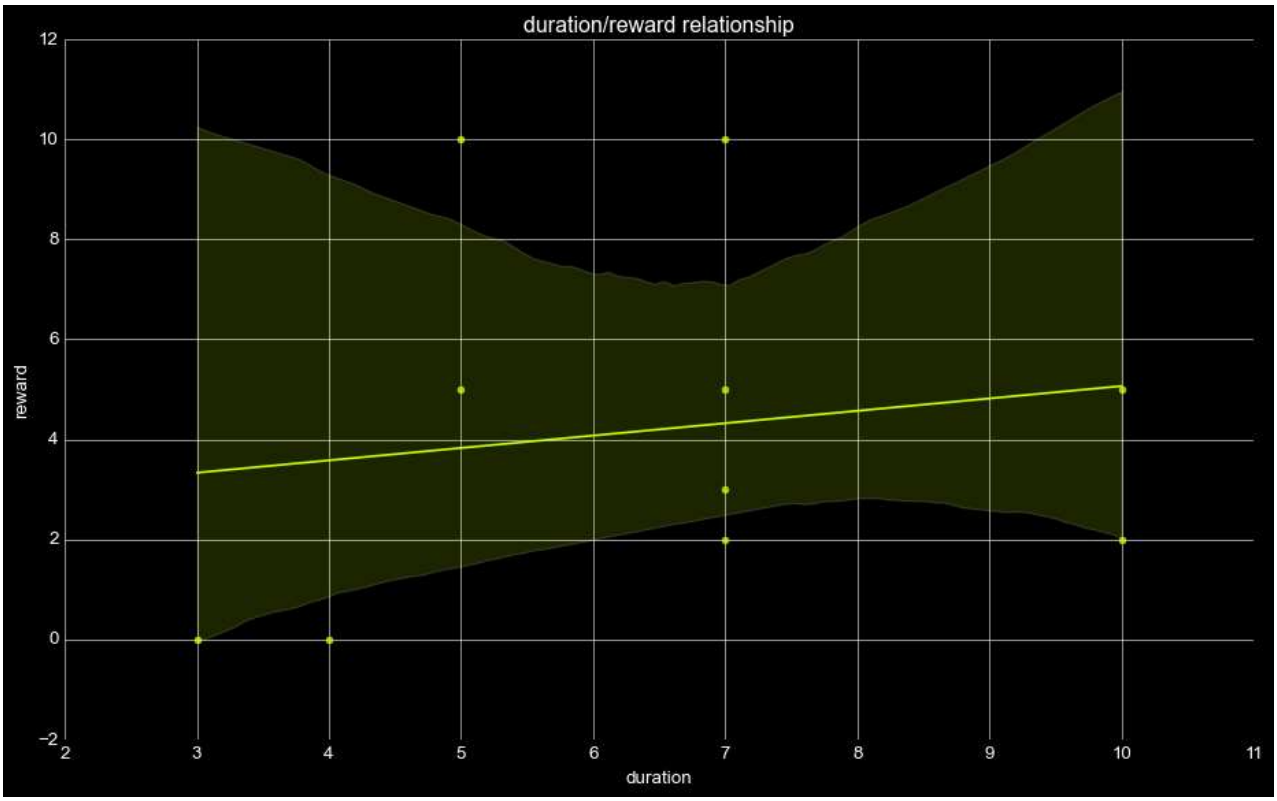
In [49]:
```
plot_init()
plt.title("difficulty/duration relationship")
sns.regplot(x="difficulty",y="duration", data=portfolio, color="#661ED5")
```

Out[49]:
```
<AxesSubplot:title={'center':'difficulty/duration relationship'}, xlabel='difficulty', y
label='duration'>
```



In [50]:
```
plot_init()
plt.title("duration/reward relationship")
sns.regplot(x="duration",y="reward", data=portfolio, color="#C6FF06")
```

Out[50]:
```
<AxesSubplot:title={'center':'duration/reward relationship'}, xlabel='duration', ylabel
='reward'>
```



In [51]:
```
portfolio
```

Out[51]:

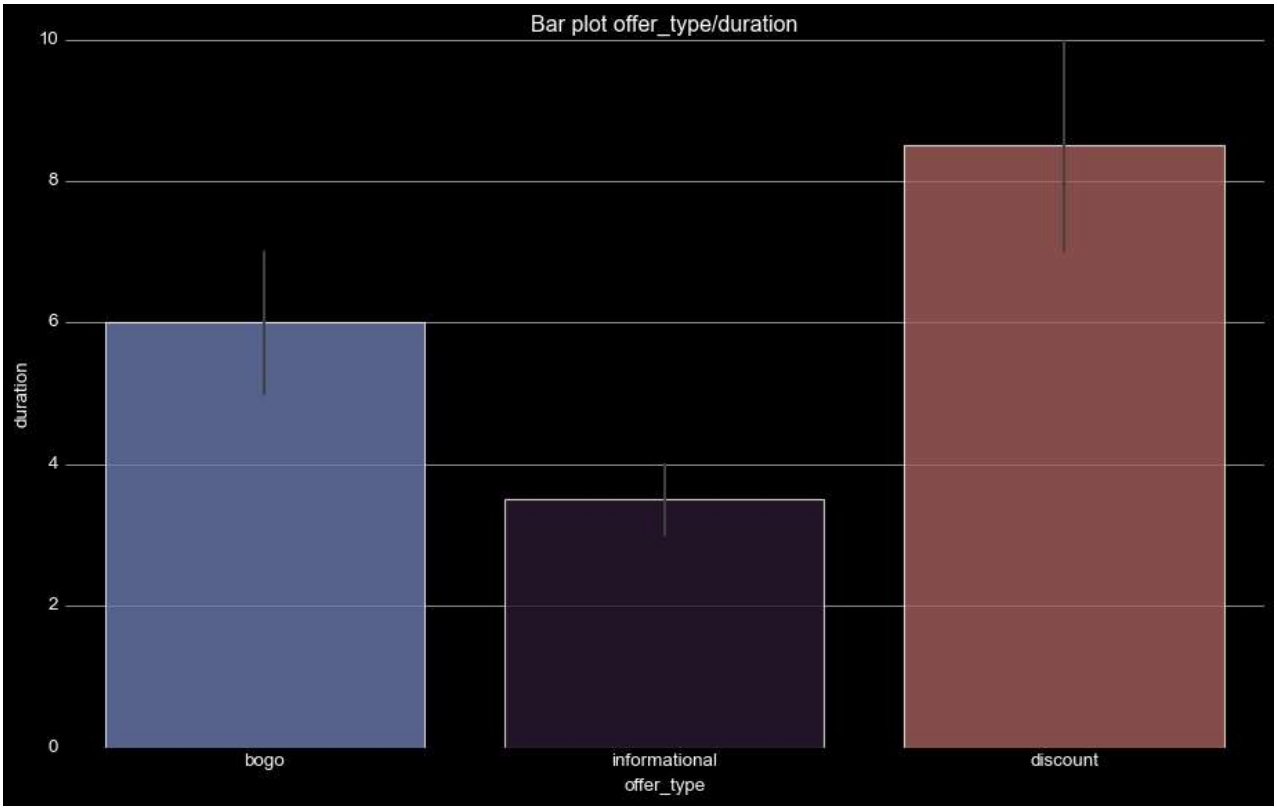
	reward	channels	difficulty	duration	offer_type	id
0	10	[email, mobile, social]	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd
1	10	[web, email, mobile, social]	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0
2	0	[web, email, mobile]	0	4	informational	3f207df678b143eea3cee63160fa8bed
3	5	[web, email, mobile]	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9
4	5	[web, email]	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7
5	3	[web, email, mobile, social]	7	7	discount	2298d6c36e964ae4a3e7e9706d1fb8c2
6	2	[web, email, mobile, social]	10	10	discount	fafdcd668e3743c1bb461111dcafc2a4
7	0	[email, mobile, social]	0	3	informational	5a8bc65990b245e5a138643cd4eb9837
8	5	[web, email, mobile, social]	5	5	bogo	f19421c1d4aa40978ebb69ca19b0e20d
9	2	[web, email, mobile]	10	7	discount	2906b810c7d4411798c6938adc9daaa5

In [52]:

```
plot_init()
plt.title("Bar plot offer_type/duration")
sns.barplot(x="offer_type",y="duration",data=portfolio, palette="twilight", alpha=.8)
```

Out[52]:

```
<AxesSubplot:title={'center':'Bar plot offer_type/duration'}, xlabel='offer_type', ylabe
l='duration'>
```

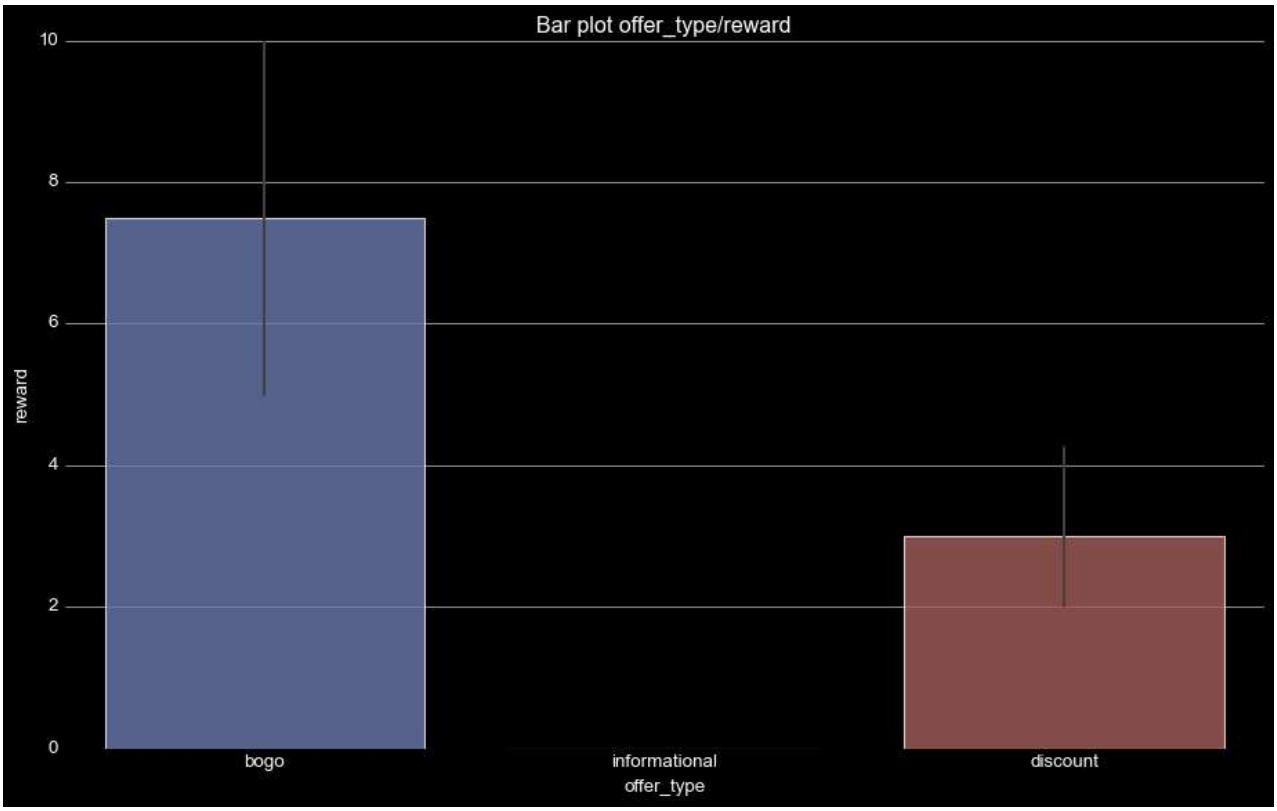


In [53]:

```
plot_init()
plt.title("Bar plot offer_type/reward")
sns.barplot(x="offer_type",y="reward",data=portfolio, palette="twilight", alpha=.8)
```

Out[53]:

```
<AxesSubplot:title={'center':'Bar plot offer_type/reward'}, xlabel='offer_type', ylabel
='reward'>
```



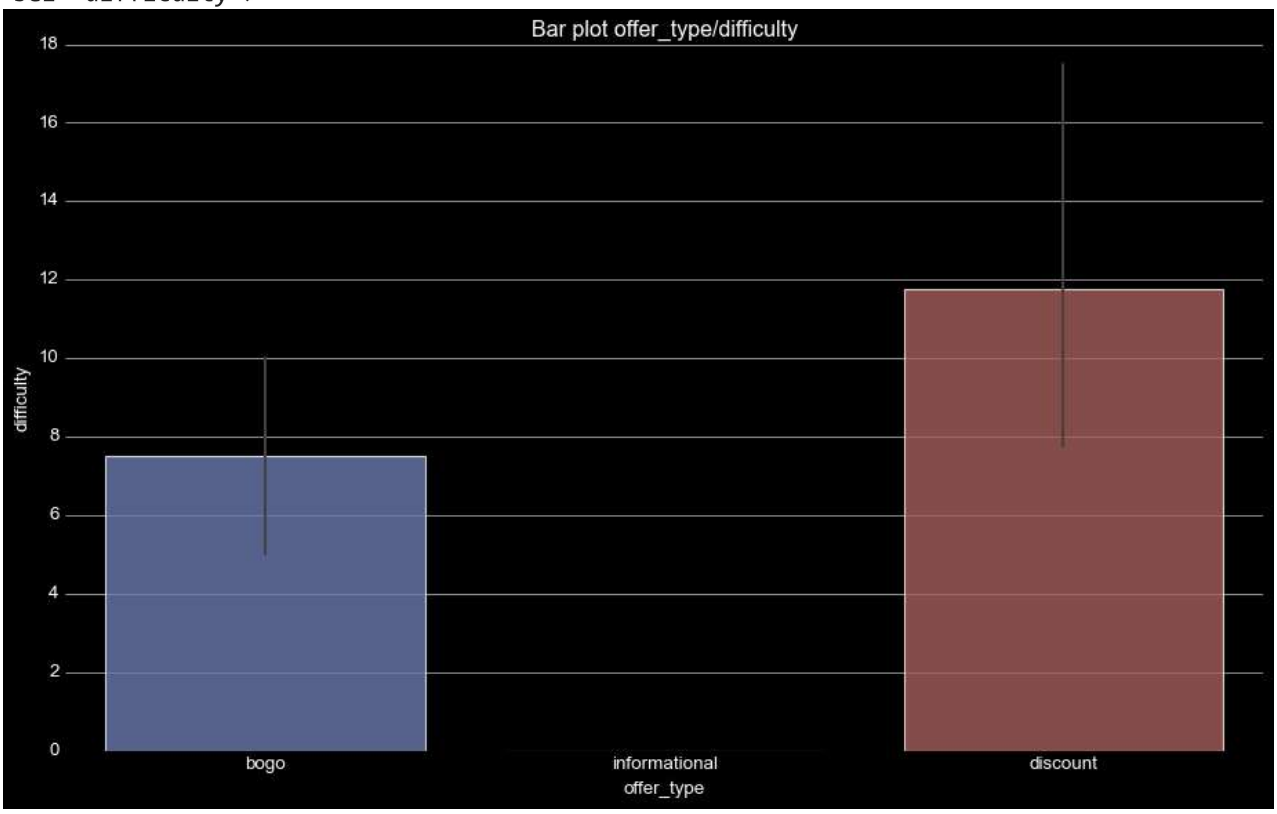
In [54]:

```
plot_init()
```

```
plt.title("Bar plot offer_type/difficulty")
sns.barplot(x="offer_type",y="difficulty",data=portfolio, palette="twilight", alpha=.8)
```

Out[54]:

```
<AxesSubplot:title={'center':'Bar plot offer_type/difficulty'}, xlabel='offer_type', yla
bel='difficulty'>
```



In [55]:

```
portfolio
```

Out[55]:

	reward	channels	difficulty	duration	offer_type	id
0	10	[email, mobile, social]	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd
1	10	[web, email, mobile, social]	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0
2	0	[web, email, mobile]	0	4	informational	3f207df678b143eea3cee63160fa8bed
3	5	[web, email, mobile]	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9
4	5	[web, email]	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7
5	3	[web, email, mobile, social]	7	7	discount	2298d6c36e964ae4a3e7e9706d1fb8c2
6	2	[web, email, mobile, social]	10	10	discount	fafdcd668e3743c1bb461111dcafc2a4
7	0	[email, mobile, social]	0	3	informational	5a8bc65990b245e5a138643cd4eb9837
8	5	[web, email, mobile, social]	5	5	bogo	f19421c1d4aa40978ebb69ca19b0e20d

	reward	channels	difficulty	duration	offer_type	id
9	2	[web, email, mobile]	10	7	discount	2906b810c7d4411798c6938adc9daaa5

In [56]:
```
dum = pd.get_dummies(portfolio['channels'].explode())
portfolio = pd.concat([portfolio, dum.groupby(level=0).sum()], axis=1).drop('channels',
```

In [57]:
```
portfolio
```

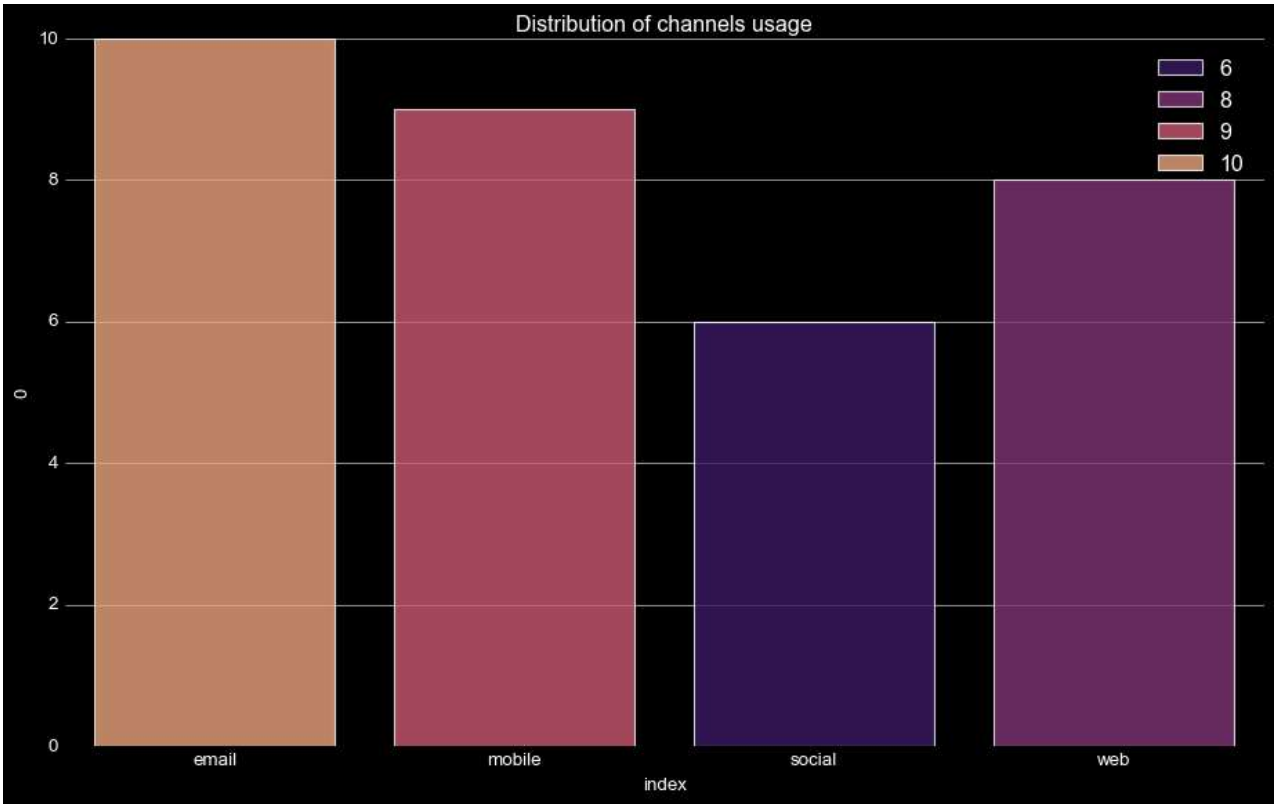
Out[57]:

	reward	difficulty	duration	offer_type	id	email	mobile	social
0	10	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd	1	1	1
1	10	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0	1	1	1
2	0	0	4	informational	3f207df678b143eea3cee63160fa8bed	1	1	0
3	5	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9	1	1	0
4	5	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7	1	0	0
5	3	7	7	discount	2298d6c36e964ae4a3e7e9706d1fb8c2	1	1	1
6	2	10	10	discount	fafdcd668e3743c1bb461111dcafc2a4	1	1	1
7	0	0	3	informational	5a8bc65990b245e5a138643cd4eb9837	1	1	1
8	5	5	5	bogo	f19421c1d4aa40978ebb69ca19b0e20d	1	1	1
9	2	10	7	discount	2906b810c7d4411798c6938adc9daaa5	1	1	0

In [58]:
```
plot_init()
plt.title("Distribution of channels usage")
sns.barplot(portfolio[["email","mobile","social","web"]].sum().reset_index()["index"],p
            alpha=.8, palette="magma", hue=portfolio[["email","mobile","social","web"]].
```

```
C:\Users\palla\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variables as keyword args: x, y. From version 0.12, the only valid posit
ional argument will be `data`, and passing other arguments without an explicit keyword w
ill result in an error or misinterpretation.
  warnings.warn(
```

Out[58]:
```
<AxesSubplot:title={'center':'Distribution of channels usage'}, xlabel='index', ylabel
='0'>
```



In [59]:
```
transcript["event"].unique()
```

Out[59]:
```
array(['offer received', 'offer viewed', 'transaction', 'offer completed'],
      dtype=object)
```

In [60]:
```
transcript[transcript["event"]=="offer completed"].iloc[0]
```

Out[60]:
```
person                    9fa9ae8f57894cc9a3b8a9bbe0fc1b2f
event                                       offer completed
value     {'offer_id': '2906b810c7d4411798c6938adc9daaa5...
time                                                      0
Name: 12658, dtype: object
```

In [61]:
```
clvals = []
for i in range(len(transcript)):
    row = transcript["value"][i]
    event = transcript["event"][i]
    if event=="offer received" or event=="offer viewed" or event=="offer completed":
        try:
            r = row["offer id"]
        except:
            r = row["offer_id"]
    elif event=="transaction":
        r = row["amount"]

    clvals.append(r)
```

In [62]:
```
tt = transcript.drop("value", axis=1)
tt["value"] = clvals
transcript = tt
transcript
```

Out[62]:

	person	event	time	value
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	0	9b98b8c7a33c4b65b9aebfe6a799e6d9
1	a03223e636434f42ac4c3df47e8bac43	offer received	0	0b1e1539f2cc45b7b9fa7c272da2e1d7
2	e2127556f4f64592b11af22de27a7932	offer received	0	2906b810c7d4411798c6938adc9daaa5
3	8ec6ce2a7e7949b1bf142def7d0e0586	offer received	0	fafdcd668e3743c1bb461111dcafc2a4
4	68617ca6246f4fbc85e91a2a49552598	offer received	0	4d5c57ea9a6940dd891ad53e9dbe8da0
...	...	...	...	...
306529	b3a1272bc9904337b331bf348c3e8c17	transaction	714	1.59
306530	68213b08d99a4ae1b0dcb72aebd9aa35	transaction	714	9.53
306531	a00058cf10334a308c68e7631c529907	transaction	714	3.61
306532	76ddbd6576844afe811f1a3c0fbb5bec	transaction	714	3.53
306533	c02b10e8752c4d8e9b73f918558531f7	transaction	714	4.05

306534 rows × 4 columns

In [63]:
```
transcript[transcript["event"]=="transaction"]["value"].mean()
```

Out[63]:
```
12.77735615639814
```

In [64]:
```
transcript[transcript["event"]=="transaction"]["value"].median()
```

Out[64]:
```
8.89
```

In [65]:
```
transcript[transcript["event"]=="transaction"]["value"].min()
```

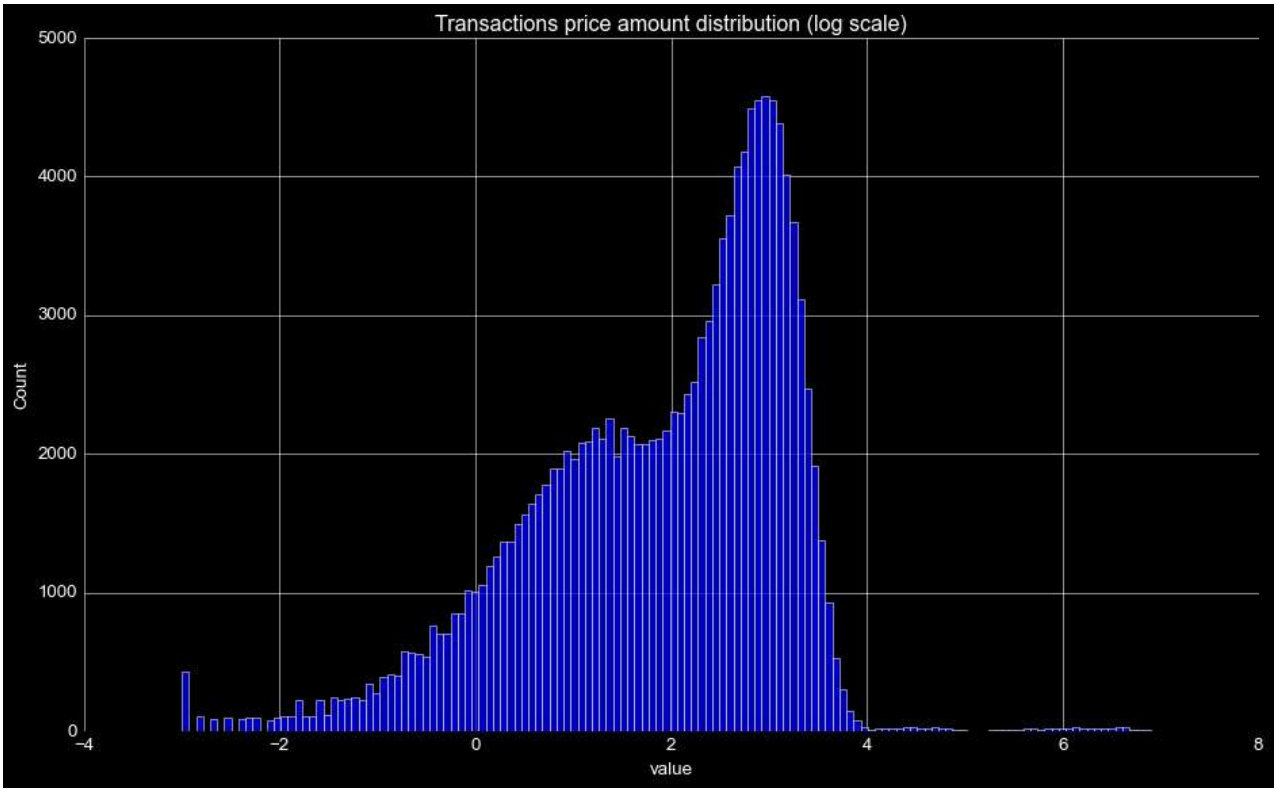
Out[65]:
```
0.05
```

In [66]:
```
transcript[transcript["event"]=="transaction"]["value"].max()
```

Out[66]:
```
1062.28
```

In [67]:
```
plot_init()
plt.title("Transactions price amount distribution (log scale)")
sns.histplot(x=np.log(transcript[transcript["event"]=="transaction"]["value"].astype('f
```

Out[67]:
```
<AxesSubplot:title={'center':'Transactions price amount distribution (log scale)'}, xlab
el='value', ylabel='Count'>
```



In [68]:

```
plot_init()
plt.title("Transactions price amount distribution (log scale)")
sns.kdeplot(x=np.log(transcript[transcript["event"]=="transaction"]["value"].astype('fl
```

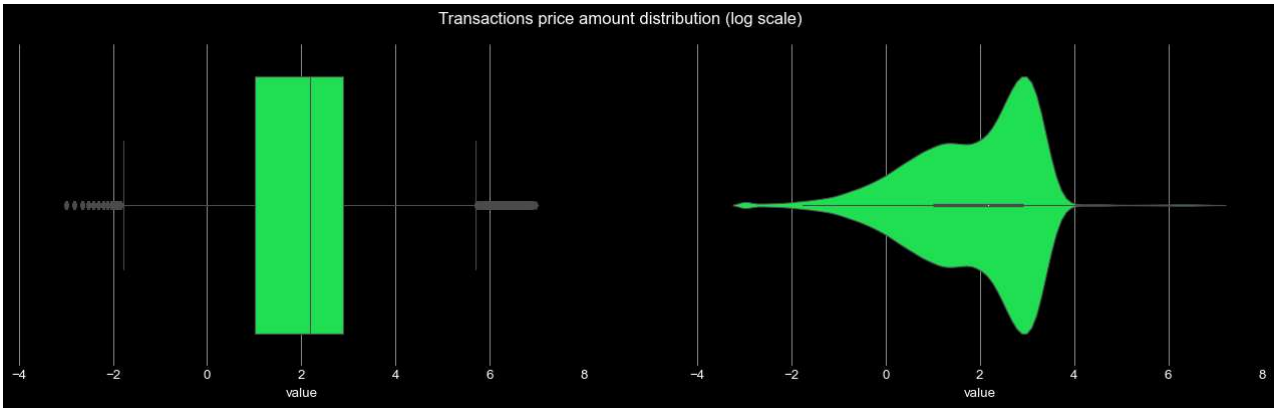
```
C:\Users\palla\anaconda3\lib\site-packages\seaborn\distributions.py:1699: FutureWarning:
The `bw` parameter is deprecated in favor of `bw_method` and `bw_adjust`. Using 0.1 for
`bw_method`, but please see the docs for the new parameters and update your code.
  warnings.warn(msg, FutureWarning)
```

Out[68]:

```
<AxesSubplot:title={'center':'Transactions price amount distribution (log scale)'}, xlab
el='value', ylabel='Density'>
```

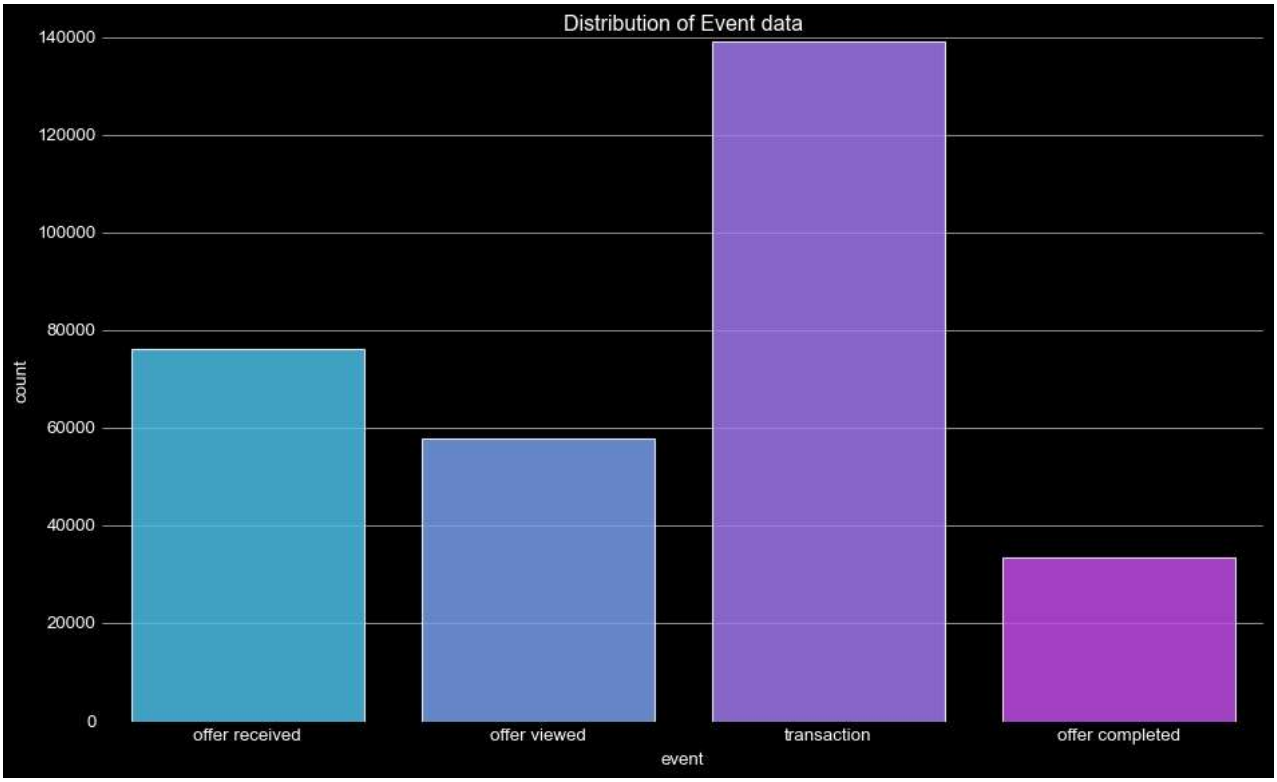
In [69]:
```
fig, axs = plt.subplots(1, 2, figsize=(20,5))

fig.suptitle("Transactions price amount distribution (log scale)", size=15)
sns.boxplot(x=np.log(transcript[transcript["event"]=="transaction"]["value"].astype('fl
sns.violinplot(x=np.log(transcript[transcript["event"]=="transaction"]["value"].astype(
```

Out[69]:
```
<AxesSubplot:xlabel='value'>
```



In [70]:
```
plot_init()
plt.title("Distribution of Event data")
sns.countplot(transcript["event"], palette="cool", alpha=.85)
```

Out[70]:
```
C:\Users\palla\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pas
s the following variable as a keyword arg: x. From version 0.12, the only valid position
al argument will be `data`, and passing other arguments without an explicit keyword will
result in an error or misinterpretation.
  warnings.warn(
<AxesSubplot:title={'center':'Distribution of Event data'}, xlabel='event', ylabel='coun
t'>
```



In [71]:
```
transcript[transcript["event"]=="transaction"]["time"].mean()
```

Out[71]:
```
381.58433427130035
```

In [72]:
```
ts_pf = transcript.merge(profile, left_on="person", right_on="id").drop("id", axis=1)
ts_pf
```

Out[72]:

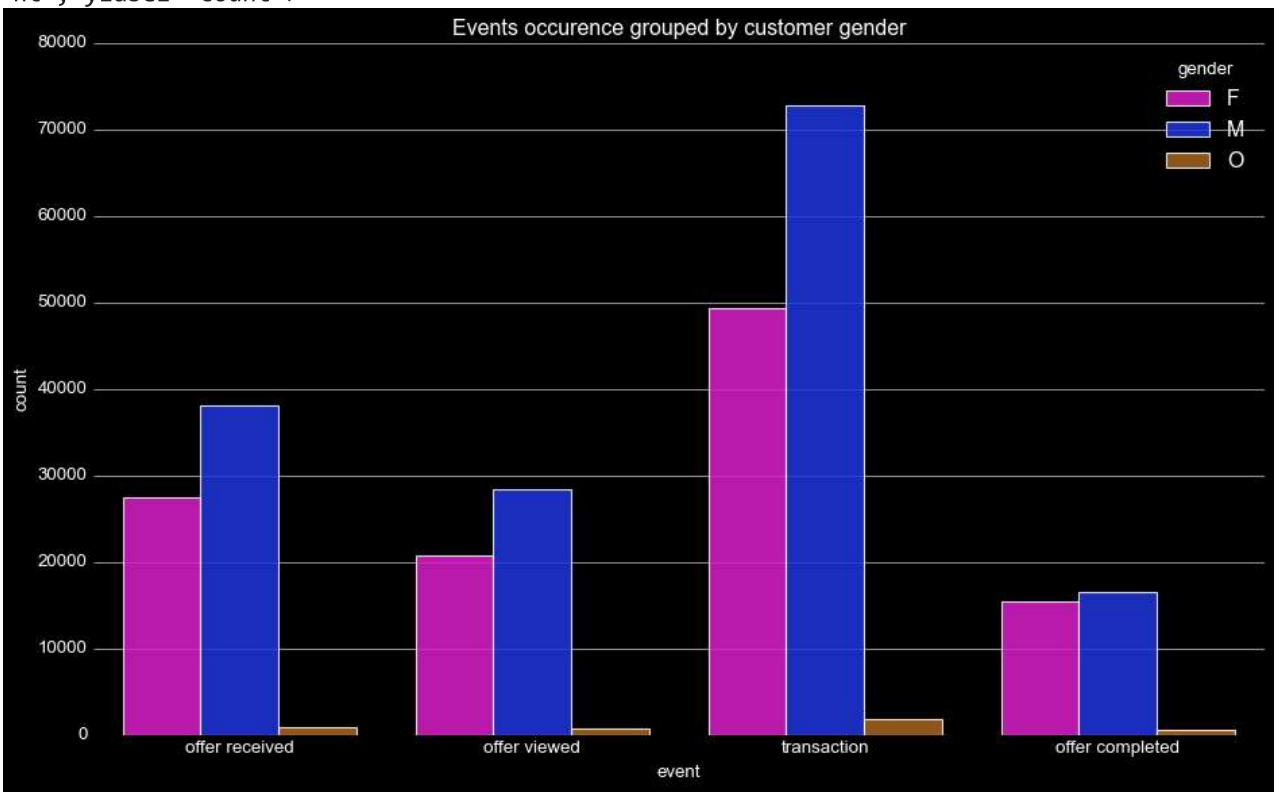
	person	event	time	value	gende
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	0	9b98b8c7a33c4b65b9aebfe6a799e6d9	
1	78afa995795e4d85b5d9ceeca43f5fef	offer viewed	6	9b98b8c7a33c4b65b9aebfe6a799e6d9	
2	78afa995795e4d85b5d9ceeca43f5fef	transaction	132	19.89	
3	78afa995795e4d85b5d9ceeca43f5fef	offer completed	132	9b98b8c7a33c4b65b9aebfe6a799e6d9	
4	78afa995795e4d85b5d9ceeca43f5fef	transaction	144	17.78	
...	...	...	...	...	
272757	9fcbff4f8d7241faa4ab8a9d19c8a812	offer viewed	504	3f207df678b143eea3cee63160fa8bed	M
272758	9fcbff4f8d7241faa4ab8a9d19c8a812	offer received	576	4d5c57ea9a6940dd891ad53e9dbe8da0	M
272759	9fcbff4f8d7241faa4ab8a9d19c8a812	offer viewed	576	4d5c57ea9a6940dd891ad53e9dbe8da0	M
272760	3045af4e98794a04a5542d3eac939b1f	offer received	576	4d5c57ea9a6940dd891ad53e9dbe8da0	

	person	event	time	value	gende
272761	3045af4e98794a04a5542d3eac939b1f	offer viewed	576	4d5c57ea9a6940dd891ad53e9dbe8da0	

272762 rows × 10 columns

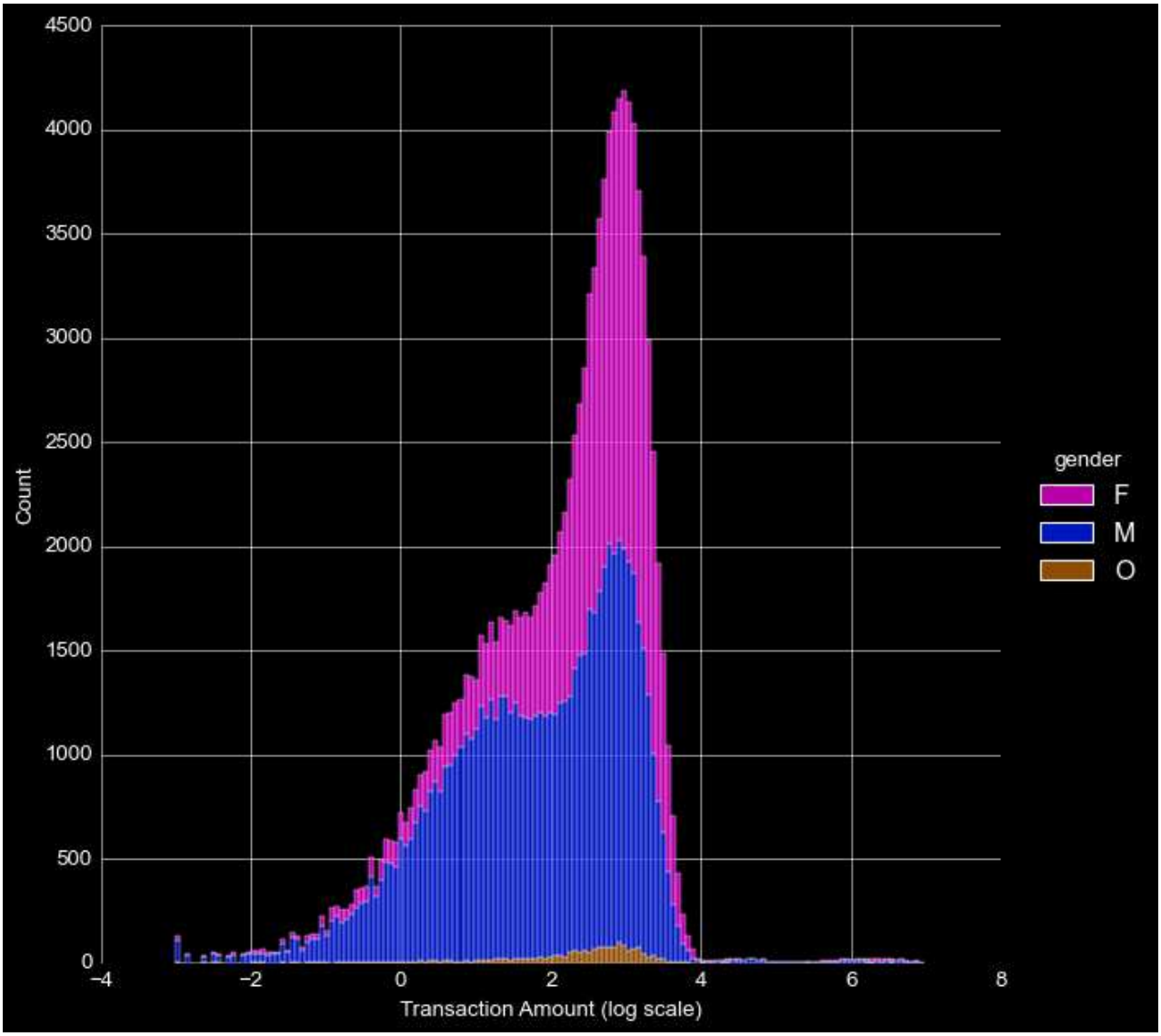
In [73]:
```
ts_pf_trimmed_transaction = ts_pf[ts_pf["event"]=="transaction"]
ts_pf_trimmed_transaction["log_value"] = np.log(ts_pf_trimmed_transaction["value"].asty
plot_init()
plt.title("Events occurence grouped by customer gender")
sns.countplot(x="event", hue="gender", data=ts_pf, palette=["#fa00e5","#001eff","#bf670
```

Out[73]:
```
<AxesSubplot:title={'center':'Events occurence grouped by customer gender'}, xlabel='eve
nt', ylabel='count'>
```



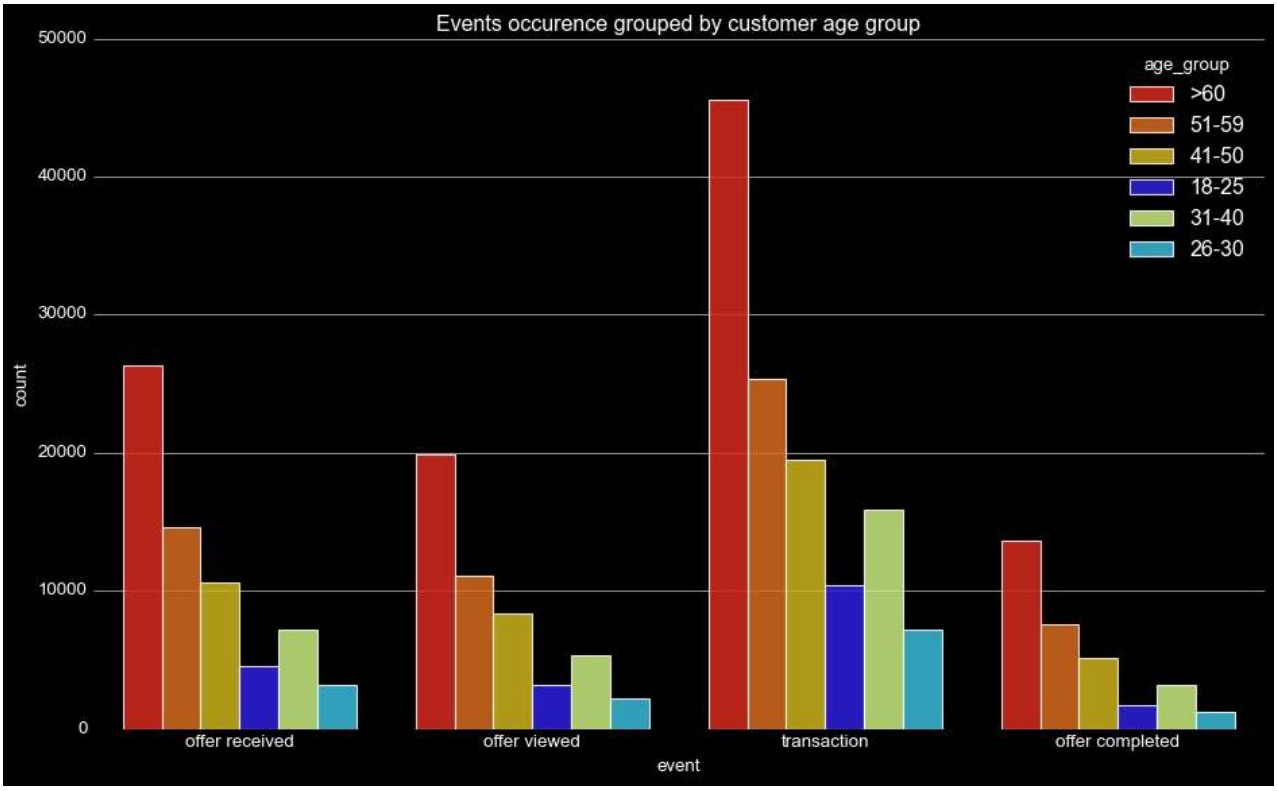
In [74]:
```
d = sns.displot(x="log_value", multiple="stack", hue="gender",data=ts_pf_trimmed_transa
axes = d.axes.flatten()
axes[0].set_xlabel('Transaction Amount (log scale)')
```

Out[74]:
```
Text(0.5, 8.844444444444436, 'Transaction Amount (log scale)')
```



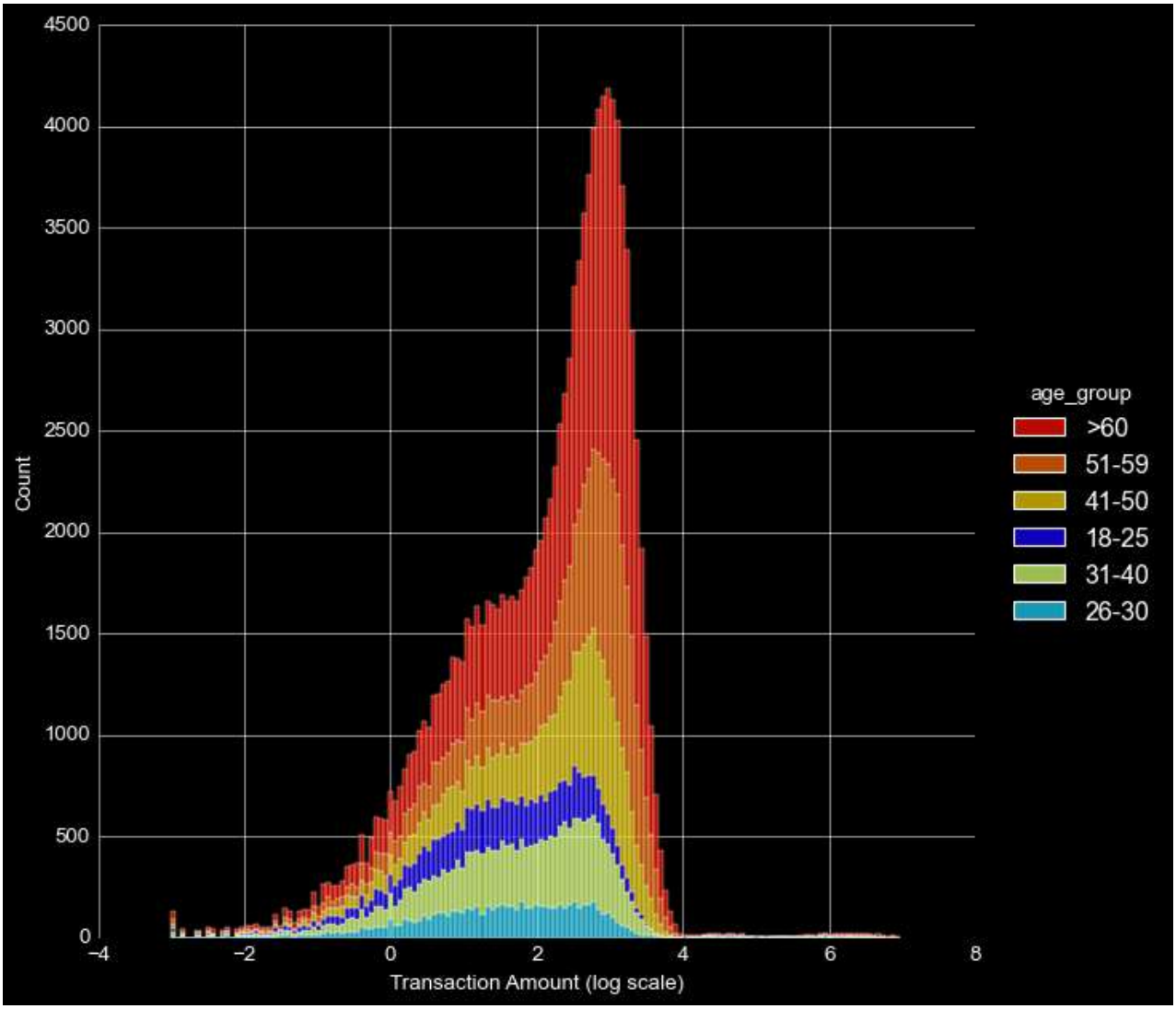
In [75]:
```
age_group_palette_tr_natu = ["#f71000","#f76700","#ebcb00","#1500ff","#d4ff70","#20cef5
plot_init()
plt.title("Events occurence grouped by customer age group")
sns.countplot(x="event", hue="age_group", data=ts_pf, palette=age_group_palette_tr_natu
```

Out[75]:
```
<AxesSubplot:title={'center':'Events occurence grouped by customer age group'}, xlabel
='event', ylabel='count'>
```



In [76]:
```
d = sns.displot(x="log_value", hue="age_group",data=ts_pf_trimmed_transaction, height=8
axes = d.axes.flatten()
axes[0].set_xlabel('Transaction Amount (log scale)')
```

Out[76]:
```
Text(0.5, 8.844444444444436, 'Transaction Amount (log scale)')
```

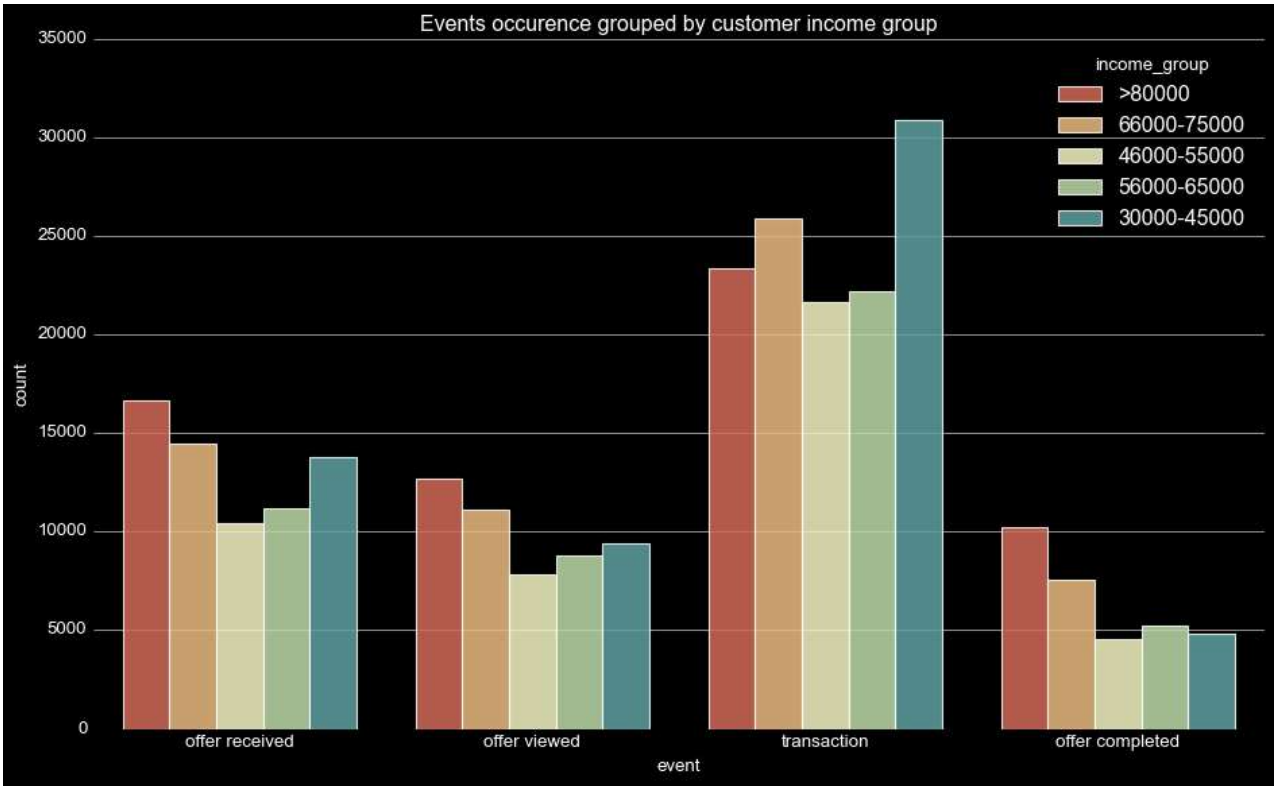


In [77]:

```
plot_init()
plt.title("Events occurence grouped by customer income group")
sns.countplot(x="event", hue="income_group", data=ts_pf, alpha=.85, palette="Spectral")
```

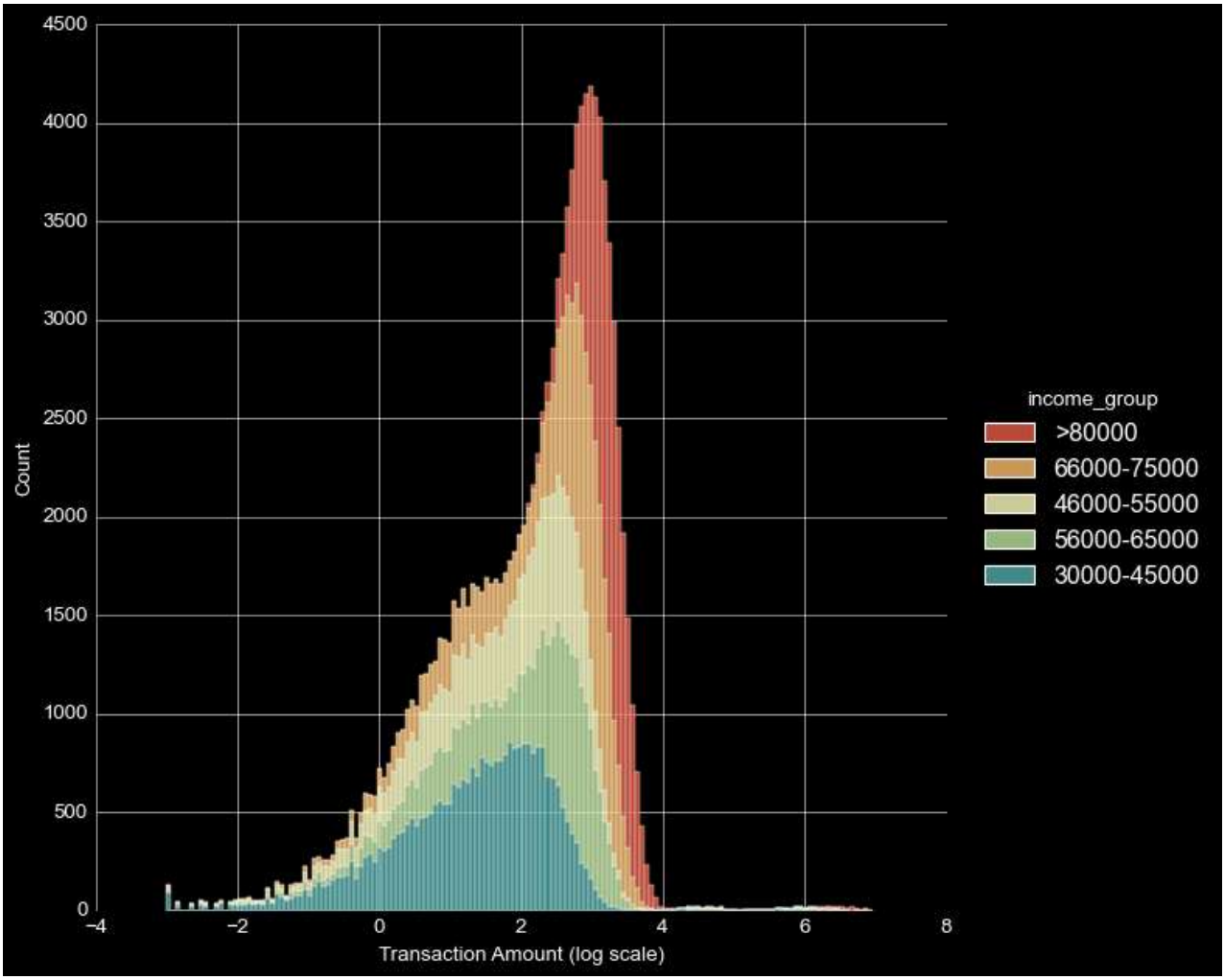
Out[77]:

```
<AxesSubplot:title={'center':'Events occurence grouped by customer income group'}, xlabe
l='event', ylabel='count'>
```



In [78]:
```
d = sns.displot(x="log_value", multiple="stack", hue="income_group",data=ts_pf_trimmed_
axes = d.axes.flatten()
axes[0].set_xlabel('Transaction Amount (log scale)')
```

Out[78]:
```
Text(0.5, 8.844444444444436, 'Transaction Amount (log scale)')
```



Among income groups, people on the lowest end tend to incur the most in transactions; nonetheless, they are the cheapest spenders. It would be interesting to analyze which group generates the most money for Starbucks by examining the product between the number of transactions and the mean transaction value.

In [79]:
```
ts_pf_trimmed_transaction["value"] = ts_pf_trimmed_transaction["value"].astype("float64
a = ts_pf_trimmed_transaction.groupby("income_group")["value"].count().reset_index()
a.rename(columns={"value":"transactions"},inplace=True)
b = ts_pf_trimmed_transaction.groupby("income_group")["value"].mean().reset_index()
b
```

Out[79]:

	income_group	value
0	30000-45000	6.458926
1	46000-55000	9.310162
2	56000-65000	11.788251
3	66000-75000	16.040742
4	>80000	28.148975

In [80]:
```
a["value_represented"] = a["transactions"] * b["value"]
a = a.sort_values("value_represented", ascending=False)
```

```
a
```

Out[80]:

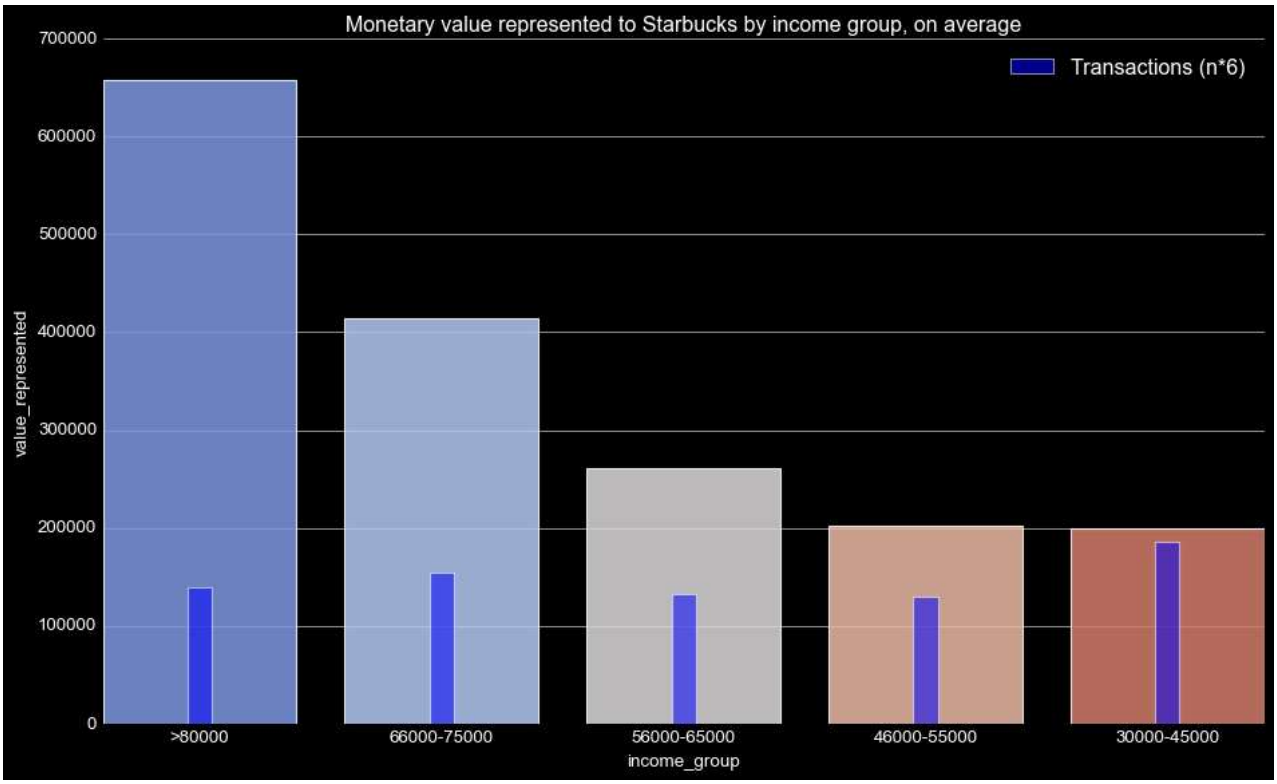
	income_group	transactions	value_represented
4	>80000	23348	657222.26
3	66000-75000	25862	414845.68
2	56000-65000	22199	261687.38
1	46000-55000	21669	201741.90
0	30000-45000	30879	199445.18

In [81]:

```
plot_init()
plt.title("Monetary value represented to Starbucks by income group, on average")
sns.barplot(x="income_group",y="value_represented", palette="coolwarm",data=a, alpha=.8
plt.bar(a["income_group"],a["transactions"]*6, width=0.1, color="blue", alpha=.55, labe
#transactions multiplied by a factor of n*6 as this way the data is more clearly visual
plt.legend()
```

Out[81]:

```
<matplotlib.legend.Legend at 0x1ac8e738c10>
```

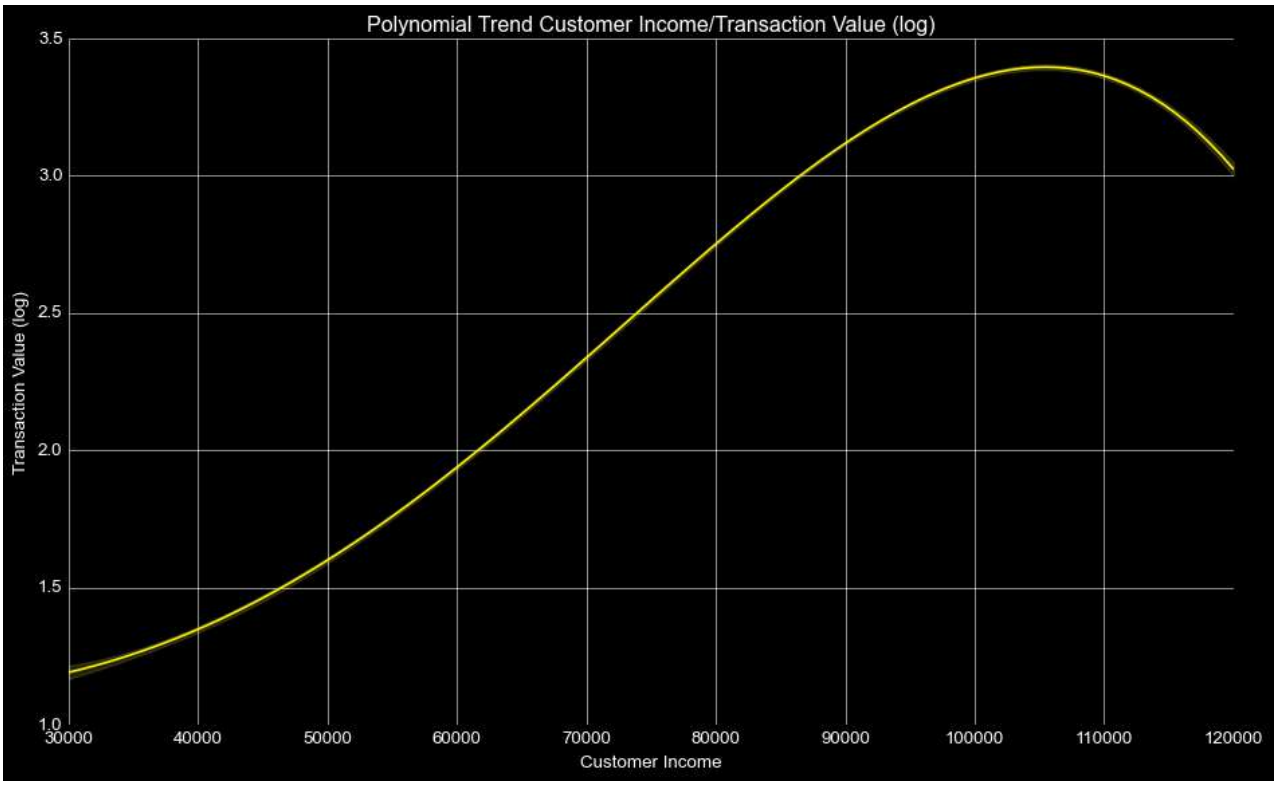


The money generated by each group follows the same order as the income categories. Though the 30000-45000 income group generates most transactions, its meager transaction value puts them last. The analysis reveals that people with more money spend more money.

In [82]:

```
plot_init()
plt.title("Polynomial Trend Customer Income/Transaction Value (log)")
sns.regplot(x="income",y="log_value", data=ts_pf_trimmed_transaction, scatter=False, or
plt.xlabel("Customer Income")
plt.ylabel("Transaction Value (log)")
```

Out[82]: Text(0, 0.5, 'Transaction Value (log)')



In [83]:
```
np.corrcoef(ts_pf_trimmed_transaction["income"],ts_pf_trimmed_transaction["log_value"])
```

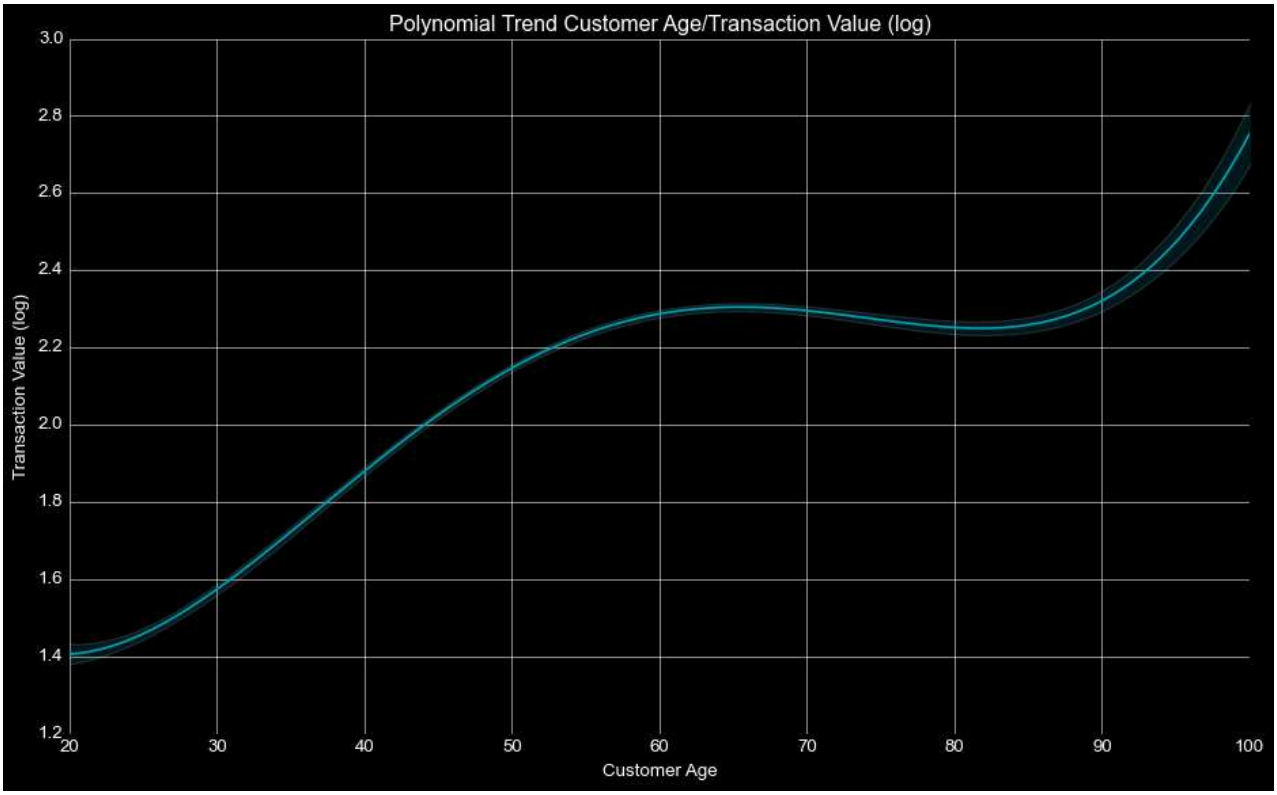
Out[83]: 0.5452444199862206

In [84]:
```
np.corrcoef(ts_pf_trimmed_transaction["income"],ts_pf_trimmed_transaction["log_value"])
```

Out[84]: 0.2972914775261102

In [85]:
```
plot_init()
plt.title("Polynomial Trend Customer Age/Transaction Value (log)")
sns.regplot(x="age",y="log_value", data=ts_pf_trimmed_transaction, scatter=False, order
plt.xlabel("Customer Age")
plt.ylabel("Transaction Value (log)")
plt.xlim(20,100)
```

Out[85]: (20.0, 100.0)



In [86]:
```
np.corrcoef(ts_pf_trimmed_transaction["age"],ts_pf_trimmed_transaction["log_value"])[0]
```

Out[86]:
```
0.22949111541962366
```

In [87]:
```
np.corrcoef(ts_pf_trimmed_transaction["age"],ts_pf_trimmed_transaction["log_value"])[0]
```

Out[87]:
```
0.05266617205654303
```

In [88]:
```
ts_pf_trimmed_transaction
```

Out[88]:

	person	event	time	value	gender	age	became_member_on
2	78afa995795e4d85b5d9ceeca43f5fef	transaction	132	19.89	F	75	20170509
4	78afa995795e4d85b5d9ceeca43f5fef	transaction	144	17.78	F	75	20170509
7	78afa995795e4d85b5d9ceeca43f5fef	transaction	222	19.67	F	75	20170509
8	78afa995795e4d85b5d9ceeca43f5fef	transaction	240	29.72	F	75	20170509
9	78afa995795e4d85b5d9ceeca43f5fef	transaction	378	23.93	F	75	20170509
...	...	...	...	...	...	...	...
272744	47683732768a4f7db7abb710ca22e66e	transaction	684	7.10	M	24	20171108
272747	3873fe915496482eb589fa316ae7b0db	transaction	510	1.57	M	58	20170905
272750	3873fe915496482eb589fa316ae7b0db	transaction	588	1.21	M	58	20170905
272751	3873fe915496482eb589fa316ae7b0db	transaction	612	2.65	M	58	20170905
272755	8578196a074a4f328976e334fa9383a3	transaction	702	4.62	M	48	20180610

123957 rows × 11 columns

In [89]:
```
portfolio
```

Out[89]:

	reward	difficulty	duration	offer_type	id	email	mobile	social
0	10	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd	1	1	1
1	10	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0	1	1	1
2	0	0	4	informational	3f207df678b143eea3cee63160fa8bed	1	1	0
3	5	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9	1	1	0
4	5	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7	1	0	0
5	3	7	7	discount	2298d6c36e964ae4a3e7e9706d1fb8c2	1	1	1
6	2	10	10	discount	fafdcd668e3743c1bb461111dcafc2a4	1	1	1
7	0	0	3	informational	5a8bc65990b245e5a138643cd4eb9837	1	1	1
8	5	5	5	bogo	f19421c1d4aa40978ebb69ca19b0e20d	1	1	1
9	2	10	7	discount	2906b810c7d4411798c6938adc9daaa5	1	1	0

In [90]:
```
ts_pf_offers_portfolio = ts_pf[ts_pf["event"]!="transaction"]
ts_pf_offers_portfolio.rename(columns={"value": "id"}, inplace=True)
ts_pf_offers_portfolio = ts_pf_offers_portfolio.merge(portfolio)
ts_pf_offers_portfolio
```

Out[90]:

	person	event	time	id	gende
0	78afa995795e4d85b5d9ceeca43f5fef	offer received	0	9b98b8c7a33c4b65b9aebfe6a799e6d9	
1	78afa995795e4d85b5d9ceeca43f5fef	offer viewed	6	9b98b8c7a33c4b65b9aebfe6a799e6d9	
2	78afa995795e4d85b5d9ceeca43f5fef	offer completed	132	9b98b8c7a33c4b65b9aebfe6a799e6d9	
3	e2127556f4f64592b11af22de27a7932	offer received	408	9b98b8c7a33c4b65b9aebfe6a799e6d9	M
4	e2127556f4f64592b11af22de27a7932	offer viewed	420	9b98b8c7a33c4b65b9aebfe6a799e6d9	M
...	...	...	...	...	.
148800	8578196a074a4f328976e334fa9383a3	offer viewed	504	4d5c57ea9a6940dd891ad53e9dbe8da0	M
148801	9fcbff4f8d7241faa4ab8a9d19c8a812	offer received	576	4d5c57ea9a6940dd891ad53e9dbe8da0	M
148802	9fcbff4f8d7241faa4ab8a9d19c8a812	offer viewed	576	4d5c57ea9a6940dd891ad53e9dbe8da0	M

	person	event	time	id	gende
148803	3045af4e98794a04a5542d3eac939b1f	offer received	576	4d5c57ea9a6940dd891ad53e9dbe8da0	
148804	3045af4e98794a04a5542d3eac939b1f	offer viewed	576	4d5c57ea9a6940dd891ad53e9dbe8da0	

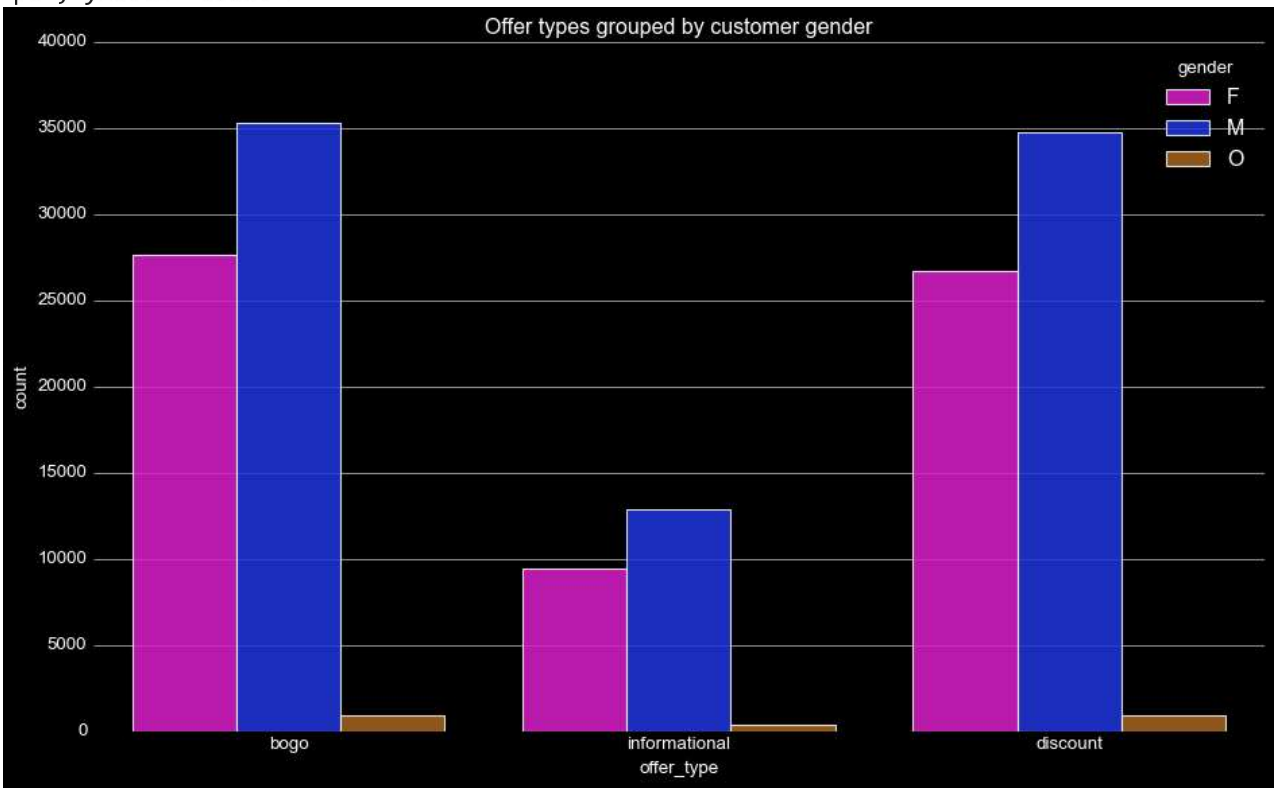
148805 rows × 18 columns

In [91]:
```
len(ts_pf_offers_portfolio)+len(ts_pf_trimmed_transaction)==len(ts_pf)
```

Out[91]:
```
True
```

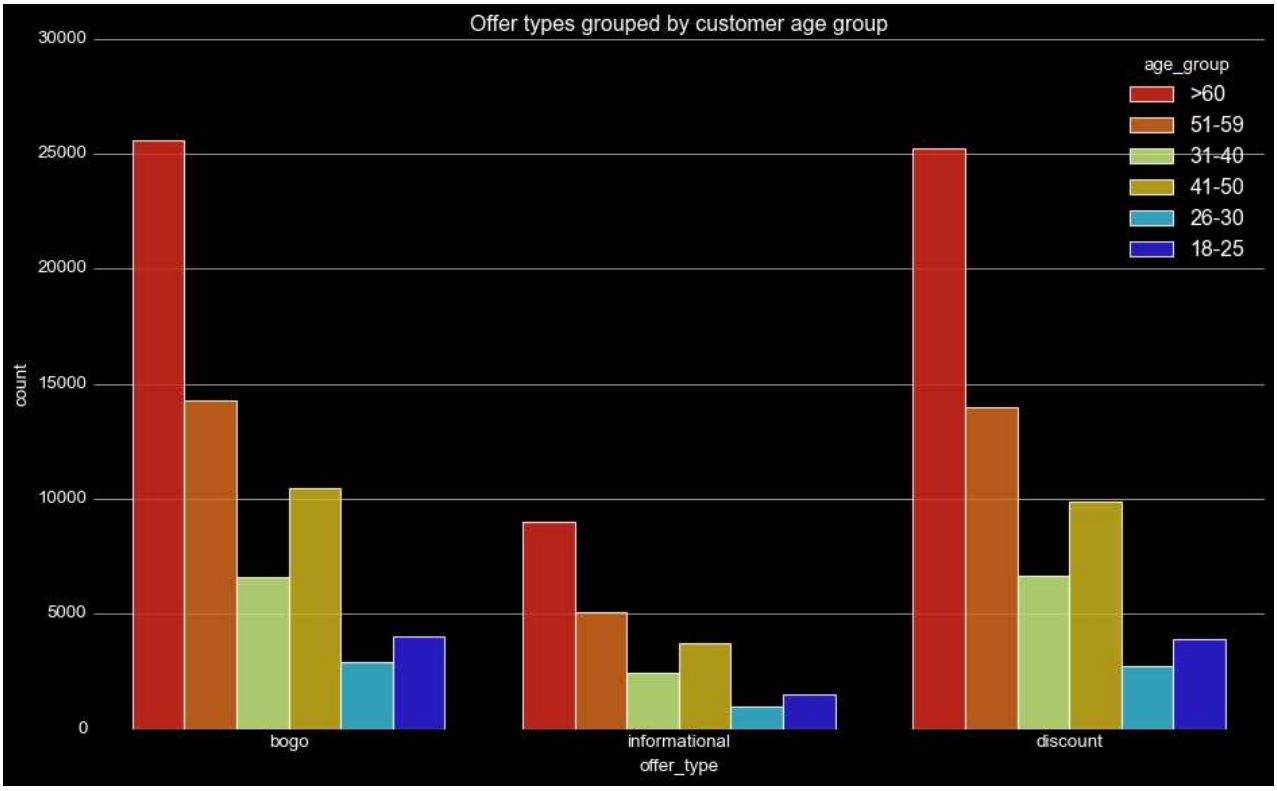
In [92]:
```
tpop = ts_pf_offers_portfolio
plot_init()
plt.title("Offer types grouped by customer gender")
sns.countplot(x="offer_type",hue="gender", data=tpop, palette=["#fa00e5","#001eff","#bf
```

Out[92]:
```
<AxesSubplot:title={'center':'Offer types grouped by customer gender'}, xlabel='offer_ty
pe', ylabel='count'>
```



In [93]:
```
age_group_palette_tr_natu2 = ["#f71000","#f76700","#d4ff70","#ebcb00","#20cef5","#1500f
plot_init()
plt.title("Offer types grouped by customer age group")
sns.countplot(x="offer_type",hue="age_group", data=tpop, palette=age_group_palette_tr_n
```

Out[93]:
```
<AxesSubplot:title={'center':'Offer types grouped by customer age group'}, xlabel='offer
_type', ylabel='count'>
```

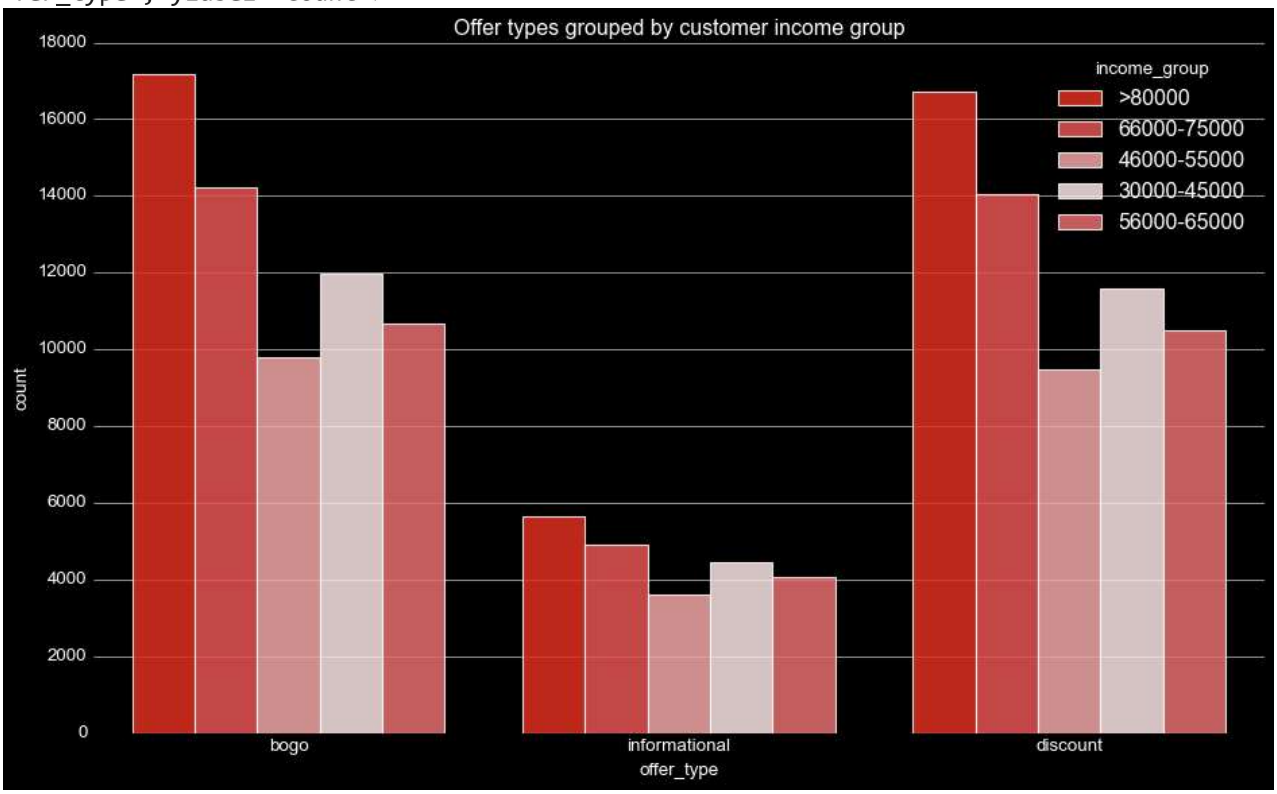


In [94]:

```
inc_nat_pal = ["#ff1500","#ff3b3b","#ff9c9c","#ffe3e3","#fc5b5b"]
plot_init()
plt.title("Offer types grouped by customer income group")
sns.countplot(x="offer_type",hue="income_group", data=tpop, palette=inc_nat_pal, alpha=
```

Out[94]:

```
<AxesSubplot:title={'center':'Offer types grouped by customer income group'}, xlabel='of
fer_type', ylabel='count'>
```

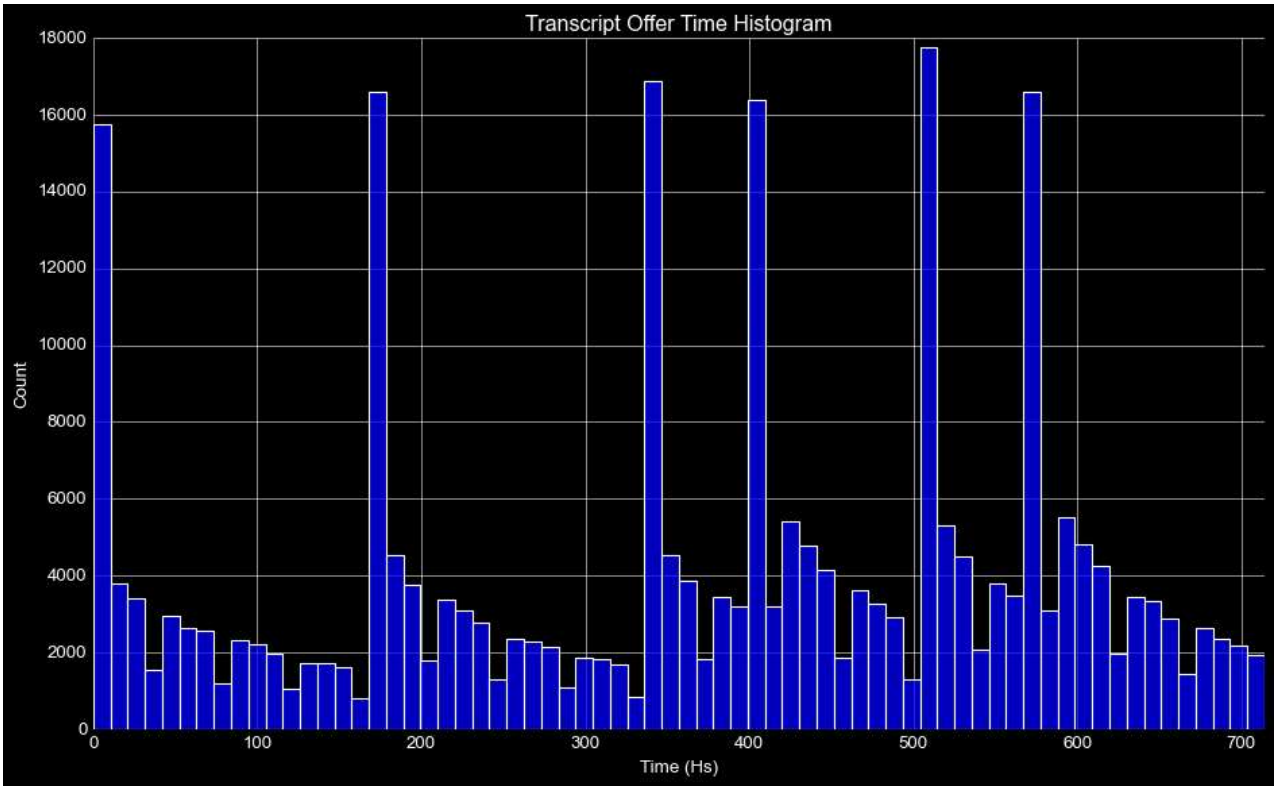


In [95]:

```
plot_init()
```

```
plt.title("Transcript Offer Time Histogram")
sns.histplot(ts_pf["time"])
plt.xlabel("Time (Hs)")
plt.xlim(0,714)
```

Out[95]: (0.0, 714.0)



In [96]:
```
ts_pf["time"].max()/24
```

Out[96]: 29.75

In [97]:
```
tptt = ts_pf_trimmed_transaction
```

Analyzing offer efficiency

In [98]:
```
tpop["time_in_days"] = tpop["time"]/24
tptt["time_in_days"] = tptt["time"]/24
portfolio["offer_name"] = ["offer_bogo_a","offer_bogo_b","offer_informational_a","offer
"offer_bogo_d","offer_discount_d"]
transcript = transcript.sort_values(["person","time"]).reset_index()
transcript_old = transcript
transcript_old
```

Out[98]:

	index	person	event	time	valu
0	55972	0009655768c64bdeb2e877511632db8f	offer received	168	5a8bc65990b245e5a138643cd4eb983
1	77705	0009655768c64bdeb2e877511632db8f	offer viewed	192	5a8bc65990b245e5a138643cd4eb983

	index	person	event	time	valu
2	89291	0009655768c64bdeb2e877511632db8f	transaction	228	22.
3	113605	0009655768c64bdeb2e877511632db8f	offer received	336	3f207df678b143eea3cee63160fa8be
4	139992	0009655768c64bdeb2e877511632db8f	offer viewed	372	3f207df678b143eea3cee63160fa8be
...	...	...	...	...	
306529	258361	ffff82501cea40309d5fdd7edcca4a07	transaction	576	14.
306530	258362	ffff82501cea40309d5fdd7edcca4a07	offer completed	576	2906b810c7d4411798c6938adc9daa
306531	262475	ffff82501cea40309d5fdd7edcca4a07	offer viewed	582	2906b810c7d4411798c6938adc9daa
306532	274809	ffff82501cea40309d5fdd7edcca4a07	transaction	606	10.
306533	289924	ffff82501cea40309d5fdd7edcca4a07	transaction	648	18.

306534 rows × 5 columns

In [99]:
```
portfolio
```

Out[99]:

	reward	difficulty	duration	offer_type	id	email	mobile	social
0	10	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd	1	1	1
1	10	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0	1	1	1
2	0	0	4	informational	3f207df678b143eea3cee63160fa8bed	1	1	0
3	5	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9	1	1	0
4	5	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7	1	0	0
5	3	7	7	discount	2298d6c36e964ae4a3e7e9706d1fb8c2	1	1	1
6	2	10	10	discount	fafdcd668e3743c1bb461111dcafc2a4	1	1	1
7	0	0	3	informational	5a8bc65990b245e5a138643cd4eb9837	1	1	1
8	5	5	5	bogo	f19421c1d4aa40978ebb69ca19b0e20d	1	1	1
9	2	10	7	discount	2906b810c7d4411798c6938adc9daaa5	1	1	0

In [100...
```
import time
import sys
transcript
```

Out[100...

	index	person	event	time	valu
0	55972	0009655768c64bdeb2e877511632db8f	offer received	168	5a8bc65990b245e5a138643cd4eb98

	index	person	event	time	valu
1	77705	0009655768c64bdeb2e877511632db8f	offer viewed	192	5a8bc65990b245e5a138643cd4eb98
2	89291	0009655768c64bdeb2e877511632db8f	transaction	228	22.
3	113605	0009655768c64bdeb2e877511632db8f	offer received	336	3f207df678b143eea3cee63160fa8be
4	139992	0009655768c64bdeb2e877511632db8f	offer viewed	372	3f207df678b143eea3cee63160fa8be
...	...	...	...	...	
306529	258361	ffff82501cea40309d5fdd7edcca4a07	transaction	576	14.
306530	258362	ffff82501cea40309d5fdd7edcca4a07	offer completed	576	2906b810c7d4411798c6938adc9daa
306531	262475	ffff82501cea40309d5fdd7edcca4a07	offer viewed	582	2906b810c7d4411798c6938adc9daa
306532	274809	ffff82501cea40309d5fdd7edcca4a07	transaction	606	10.
306533	289924	ffff82501cea40309d5fdd7edcca4a07	transaction	648	18.

306534 rows × 5 columns

In [ ]:

```
%%time

#transaction classifier algorithm

initial_df = pd.DataFrame(columns={"person","transaction_value","dependent","offer_id"}
offer_status = ["void","void",np.nan]
offer_status[2] = time
last_person = "void"
average_time = []

transactions_classified1 = ["", 0, "n",""]
transactions_classified2 = ["", 0, "n",""]
transactions_classified3 = ["", 0, "n",""]
transactions_classified4 = ["", 0, "n",""]
transactions_classified5 = ["", 0, "n",""]
transactions_classified6 = ["", 0, "n",""]



revreset = 0
cnt = 1

for i in range(len(transcript)):
    start = time.time()

    time_simple = transcript["time"][i]
    time_day = transcript["time"][i]/24
    person = transcript["person"][i]
    event = transcript["event"][i]
    if event!="transaction":
```

```
        offer_id = transcript["value"][i]
    elif event=="transaction":
        trans_value = transcript["value"][i]


    #ofer_status update

    if event=="offer received":
        offer_status[0] = "offer received, not viewed"
        offer_status[1] = offer_id
        offer_status[2] = time_simple
        last_person = person

    elif event=="offer viewed":
        offer_status[0] = "offer viewed"
        offer_status[1] = offer_id
        offer_status[2] = time_simple
        last_person = person

    elif event=="transaction":

        if person==last_person:
            if offer_status[0]=="offer received, not viewed":
                offer_status[0] = "transaction (independent)"
                dep = "n"
                offer_status[1] = f"${trans_value}"
                offer_id = np.nan

            elif offer_status[0]=="offer completed":
                offer_status[0] = "transaction (independent)"
                dep = "n"
                offer_status[1] = f"${trans_value}"
                offer_id = np.nan


            elif offer_status[0]=="offer viewed":
                offer_difficulty_min = portfolio[portfolio.offer_id==offer_id]['difficu
                if trans_value>=offer_difficulty_min:
                    offer_status[0] = "transaction (offer-viewed)"
                    dep = "y"
                    offer_status[1] = f"${trans_value} - {offer_id}"
                else:
                    offer_status[0] = "transaction (independent)"
                    dep = "n"
                    offer_status[1] = f"${trans_value}"
                    offer_id = np.nan



                #there can exist transactions just after the visualisation of an offer,
                #but these occur likely because of the visualisation of the offer, so i

            elif offer_status[0]=="void":
                offer_status[0] = "transaction (independent)"
                dep = "n"
                offer_status[1] = f"${trans_value}"
                offer_id = np.nan

        elif person!=last_person:
                offer_status[0] = "transaction (independent)"
```

```
                dep = "n"
                offer_status[1] = f"${trans_value}"
                offer_id = np.nan



        if revreset>=50000 and cnt!=6:
            time.sleep(5) #cool down cpu
            cnt = cnt+1
            revreset = 0
            print(f"{i}: reached current array limit, changing to array {cnt} - t.: {ti



        tr_to_append = [person,trans_value,dep,offer_id]
        #distributes data across arrays to drastically reduce time when stacking

        if cnt==1:
            transactions_classified1 = np.vstack((transactions_classified1,tr_to_append
        elif cnt==2:
            transactions_classified2 = np.vstack((transactions_classified2,tr_to_append
        elif cnt==3:
            transactions_classified3 = np.vstack((transactions_classified3,tr_to_append
        elif cnt==4:
            transactions_classified4 = np.vstack((transactions_classified4,tr_to_append
        elif cnt==5:
            transactions_classified5 = np.vstack((transactions_classified5,tr_to_append
        elif cnt==6:
            transactions_classified6 = np.vstack((transactions_classified6,tr_to_append



    elif event=="offer completed":
        offer_status[0] = "offer completed"
        offer_status[1] = offer_id
        last_person = person

    revreset = revreset+1



    print(f"{person[:6]} - {offer_status[0]} - {offer_status[1]} - {time_day}")
    print("")
```

In [106…

```
transactions_classified1 = np.delete(transactions_classified1, (0), axis=0)
transactions_classified2 = np.delete(transactions_classified2, (0), axis=0)
transactions_classified3 = np.delete(transactions_classified3, (0), axis=0)
transactions_classified4 = np.delete(transactions_classified4, (0), axis=0)
transactions_classified5 = np.delete(transactions_classified5, (0), axis=0)
transactions_classified6 = np.delete(transactions_classified6, (0), axis=0)

z = np.vstack((transactions_classified1,transactions_classified2))
z = np.vstack((z,transactions_classified3))
z = np.vstack((z,transactions_classified4))
z = np.vstack((z,transactions_classified5))
z = np.vstack((z,transactions_classified6))
```

```
full_transactions_classified = z
```

In [107…

```
full_transactions_classified = pd.DataFrame(full_transactions_classified)
full_transactions_classified.rename(columns={0:"person",1:"transaction_value",2:"depend
portfolio.rename(columns={"id": "offer_id"},inplace=True)
full_transactions_classified
```

Out[107…

	person	transaction_value	dependent	o
0	0009655768c64bdeb2e877511632db8f	22.16	y	5a8bc65990b245e5a138643cd4e
1	0009655768c64bdeb2e877511632db8f	8.57	n	
2	0009655768c64bdeb2e877511632db8f	14.11	n	
3	0009655768c64bdeb2e877511632db8f	13.56	y	fafdcd668e3743c1bb461111dc
4	0009655768c64bdeb2e877511632db8f	10.27	n	
…	…	…	…	
138948	ffff82501cea40309d5fdd7edcca4a07	13.17	n	
138949	ffff82501cea40309d5fdd7edcca4a07	7.79	n	
138950	ffff82501cea40309d5fdd7edcca4a07	14.23	n	
138951	ffff82501cea40309d5fdd7edcca4a07	10.12	y	2906b810c7d4411798c6938adc9
138952	ffff82501cea40309d5fdd7edcca4a07	18.91	y	2906b810c7d4411798c6938adc9

138953 rows × 4 columns

In [108…

```
trans_to_use = full_transactions_classified.merge(portfolio, on='offer_id')
trans_to_use = trans_to_use[trans_to_use['offer_type']!='informational']
class_trans_dependent_merged = trans_to_use.merge(portfolio)
transcript
```

Out[108…

	index	person	event	time	valu
0	55972	0009655768c64bdeb2e877511632db8f	offer received	168	5a8bc65990b245e5a138643cd4eb983
1	77705	0009655768c64bdeb2e877511632db8f	offer viewed	192	5a8bc65990b245e5a138643cd4eb983
2	89291	0009655768c64bdeb2e877511632db8f	transaction	228	22.1
3	113605	0009655768c64bdeb2e877511632db8f	offer received	336	3f207df678b143eea3cee63160fa8be
4	139992	0009655768c64bdeb2e877511632db8f	offer viewed	372	3f207df678b143eea3cee63160fa8be
…	…	…	…	…	
306529	258361	ffff82501cea40309d5fdd7edcca4a07	transaction	576	14.2

	index	person	event	time	valu
306530	258362	ffff82501cea40309d5fdd7edcca4a07	offer completed	576	2906b810c7d4411798c6938adc9daa
306531	262475	ffff82501cea40309d5fdd7edcca4a07	offer viewed	582	2906b810c7d4411798c6938adc9daa
306532	274809	ffff82501cea40309d5fdd7edcca4a07	transaction	606	10.
306533	289924	ffff82501cea40309d5fdd7edcca4a07	transaction	648	18.

306534 rows × 5 columns

In [110…

```
profile.rename(columns={"id": "person"}, inplace=True)
transcript_profile = transcript.merge(profile)
portfolio
```

Out[110…

	reward	difficulty	duration	offer_type	offer_id	email	mobile	social
0	10	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd	1	1	1
1	10	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0	1	1	1
2	0	0	4	informational	3f207df678b143eea3cee63160fa8bed	1	1	0
3	5	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9	1	1	0
4	5	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7	1	0	0
5	3	7	7	discount	2298d6c36e964ae4a3e7e9706d1fb8c2	1	1	1
6	2	10	10	discount	fafdcd668e3743c1bb461111dcafc2a4	1	1	1
7	0	0	3	informational	5a8bc65990b245e5a138643cd4eb9837	1	1	1
8	5	5	5	bogo	f19421c1d4aa40978ebb69ca19b0e20d	1	1	1
9	2	10	7	discount	2906b810c7d4411798c6938adc9daaa5	1	1	0

In [111…

```
compcount = transcript_profile[transcript_profile.event=="offer completed"].groupby("va
compcount = transcript_profile[transcript_profile.event=="offer completed"].groupby("va
transcript_profile
```

Out[111…

	index	person	event	time	valu
0	55972	0009655768c64bdeb2e877511632db8f	offer received	168	5a8bc65990b245e5a138643cd4eb98
1	77705	0009655768c64bdeb2e877511632db8f	offer viewed	192	5a8bc65990b245e5a138643cd4eb98
2	89291	0009655768c64bdeb2e877511632db8f	transaction	228	22.
3	113605	0009655768c64bdeb2e877511632db8f	offer received	336	3f207df678b143eea3cee63160fa8b

	index	person	event	time	valu
4	139992	0009655768c64bdeb2e877511632db8f	offer viewed	372	3f207df678b143eea3cee63160fa8b
...	...	...	...	...	
272757	258361	ffff82501cea40309d5fdd7edcca4a07	transaction	576	14.
272758	258362	ffff82501cea40309d5fdd7edcca4a07	offer completed	576	2906b810c7d4411798c6938adc9daa
272759	262475	ffff82501cea40309d5fdd7edcca4a07	offer viewed	582	2906b810c7d4411798c6938adc9daa
272760	274809	ffff82501cea40309d5fdd7edcca4a07	transaction	606	10.
272761	289924	ffff82501cea40309d5fdd7edcca4a07	transaction	648	18.

272762 rows × 11 columns

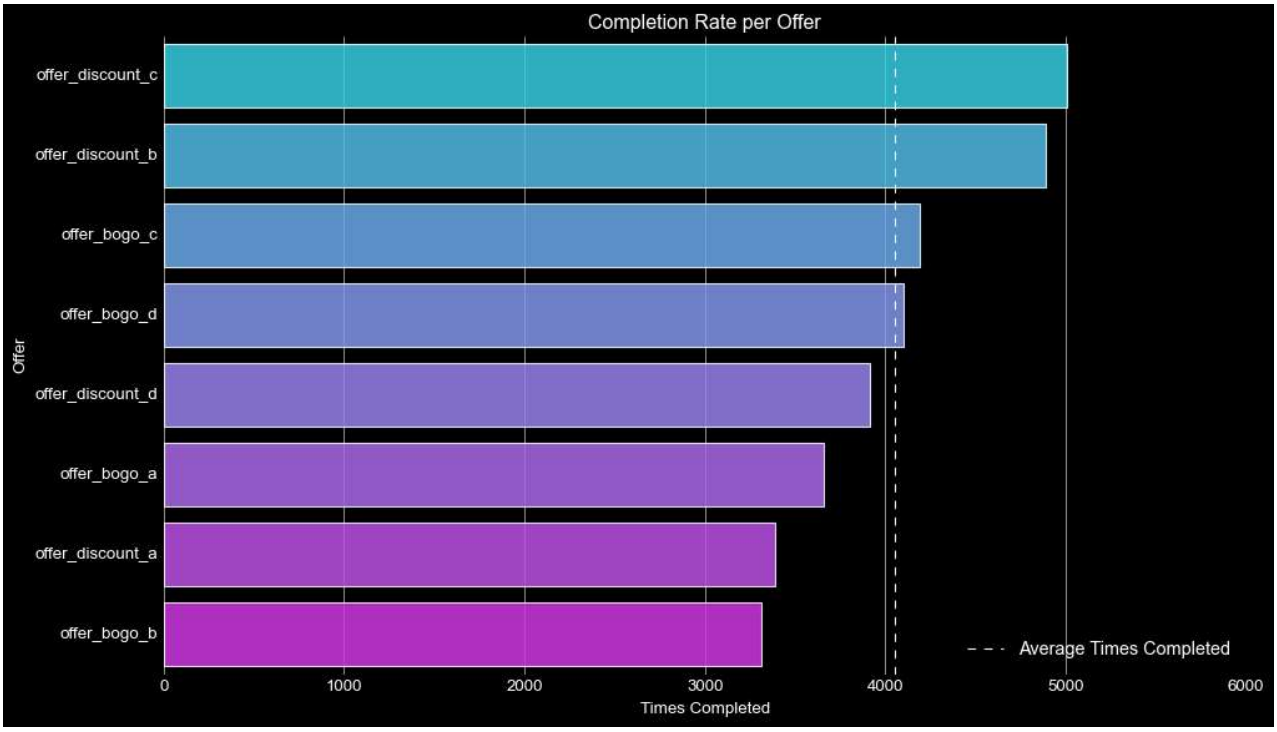
In [112...]
```
portfolio.merge(compcount)
```

Out[112...]

	reward	difficulty	duration	offer_type	offer_id	email	mobile	social
0	10	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd	1	1	1
1	10	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0	1	1	1
2	5	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9	1	1	0
3	5	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7	1	0	0
4	3	7	7	discount	2298d6c36e964ae4a3e7e9706d1fb8c2	1	1	1
5	2	10	10	discount	fafdcd668e3743c1bb461111dcafc2a4	1	1	1
6	5	5	5	bogo	f19421c1d4aa40978ebb69ca19b0e20d	1	1	1
7	2	10	7	discount	2906b810c7d4411798c6938adc9daaa5	1	1	0

In [113...]
```
plot_init()
plt.title("Completion Rate per Offer")
sns.barplot(y="offer_name",x="count_completed",data=portfolio.merge(compcount).sort_val
            alpha=.85)
plt.ylabel("Offer")
plt.xlabel("Times Completed")
plt.axvline(np.mean(portfolio.merge(compcount).sort_values("count_completed", ascending
plt.legend(loc=4, fontsize=13)
```

Out[113...]
```
<matplotlib.legend.Legend at 0x1ac8ecfaca0>
```

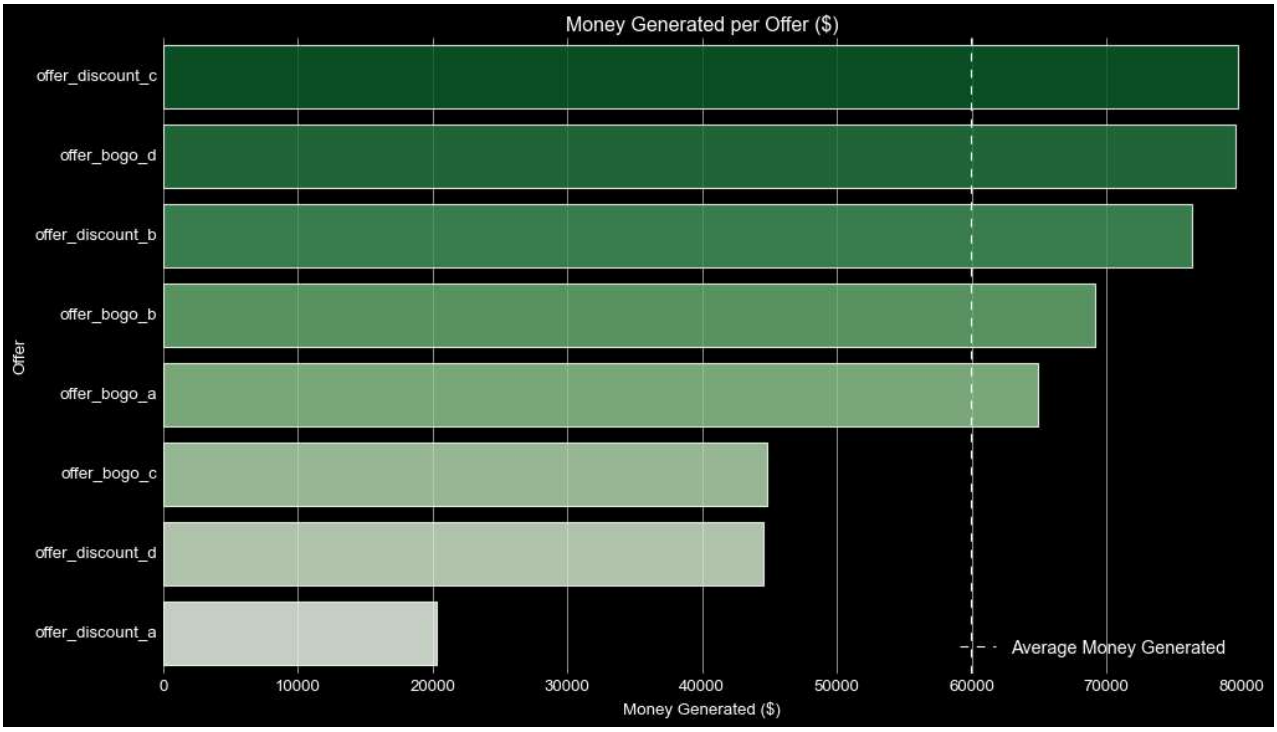


In [114…

```
class_trans_dependent_merged["transaction_value"] = class_trans_dependent_merged["trans

money_generated_by_offer = class_trans_dependent_merged.groupby("offer_id")["transactio

money_generated_by_offer = money_generated_by_offer.merge(portfolio[["offer_id","offer_
money_generated_by_offer.rename(columns={"transaction_value":"gross_total_transactions"

plot_init()
plt.title("Money Generated per Offer ($)")
sns.barplot(y="offer_name",x="gross_total_transactions",data=money_generated_by_offer,
            alpha=.85)
plt.ylabel("Offer")
plt.xlabel("Money Generated ($)")
plt.axvline(np.mean(money_generated_by_offer["gross_total_transactions"]), linestyle="-
plt.legend(loc=4, fontsize=13)
```

Out[114…

```
<matplotlib.legend.Legend at 0x1ac98d70910>
```

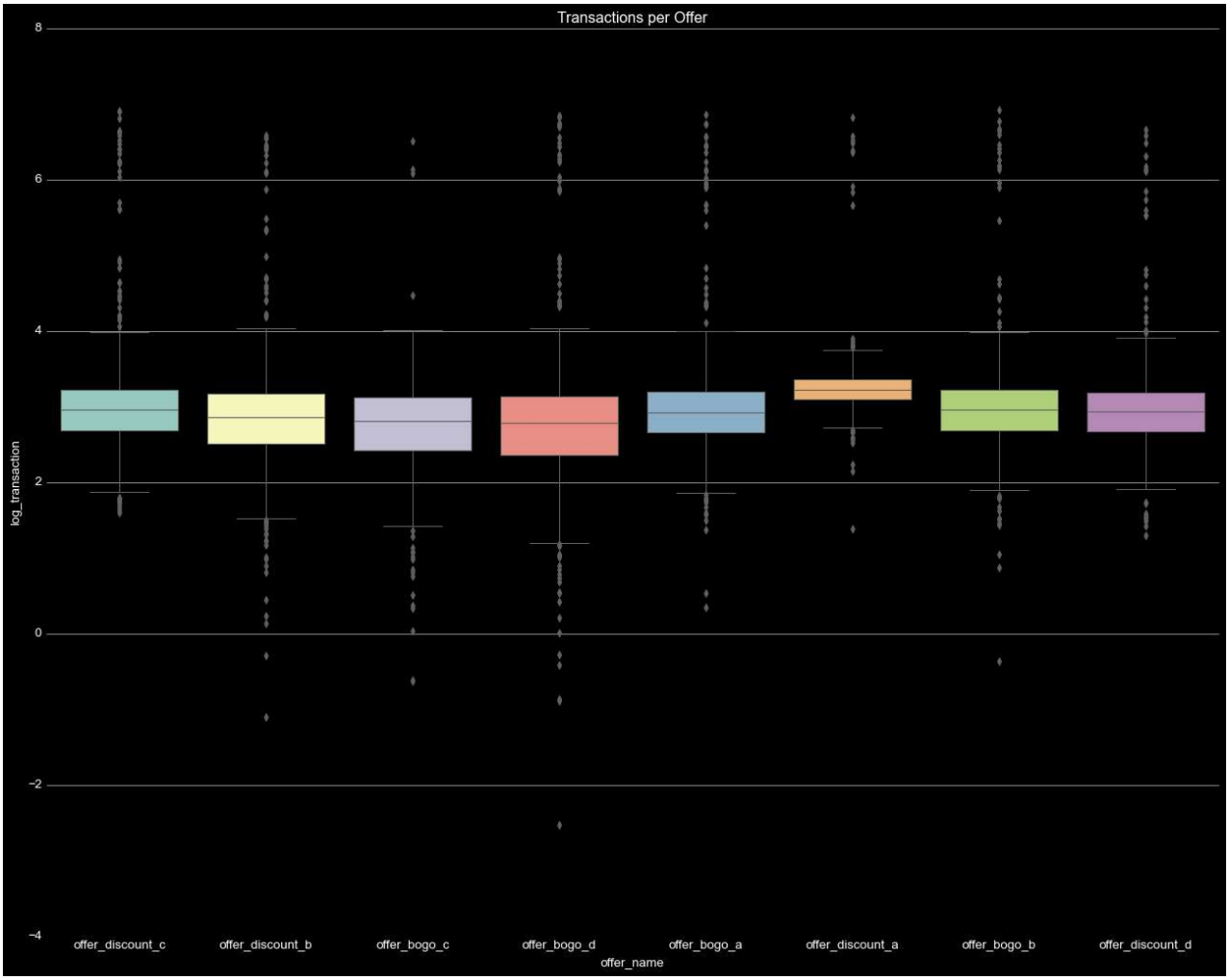


In [115…

```
mm = trans_to_use[trans_to_use.offer_id!=np.nan].merge(portfolio)
mm['log_transaction'] = np.log(mm['transaction_value'].astype('float'))
plt.figure(figsize=(20,15))
plt.title('Transactions per Offer')
sns.boxplot(x='offer_name', y='log_transaction', data=mm)
```

Out[115…

```
<AxesSubplot:title={'center':'Transactions per Offer'}, xlabel='offer_name', ylabel='log
_transaction'>
```

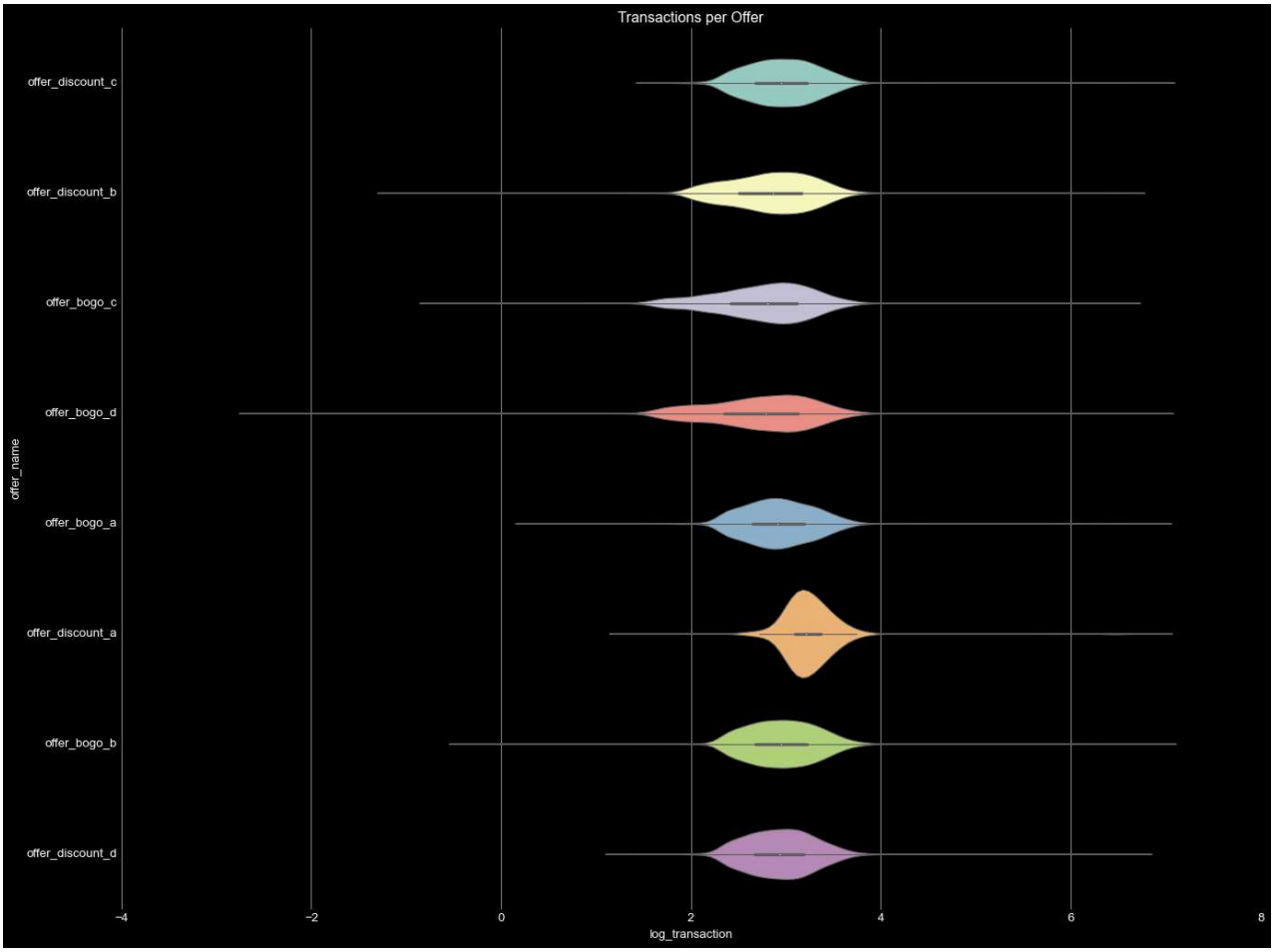


In [116…

```
mm = trans_to_use[trans_to_use.offer_id!=np.nan].merge(portfolio)
mm['log_transaction'] = np.log(mm['transaction_value'].astype('float'))
plt.figure(figsize=(20,15))
plt.title('Transactions per Offer')
sns.violinplot(y='offer_name', x='log_transaction', data=mm)
```

Out[116…

```
<AxesSubplot:title={'center':'Transactions per Offer'}, xlabel='log_transaction', ylabel
='offer_name'>
```



In [117…

```
data_to_use = transcript[transcript.event!="transaction"]
```

In [118…

```
#Legacy
def OfferClassif(selected_person):
    selected_person_cont = []
    offer_id_cont = []
    status_cont = []
    status_desc_cont = []
    data_constrained = data_to_use[data_to_use.person==selected_person]
    data_constrained = data_constrained.rename(columns={"index": "i_old"}).drop("i_old"
    for ii in range(len(data_constrained)):

        data_constrained = data_constrained.rename(columns={"index": "i_old"}).drop("i_

        event = data_constrained["event"][ii]
        offer_id = data_constrained["value"][ii]
        offer_time = data_constrained["time"][ii]

        offer_set = [0,np.nan]

        datalen = len(data_constrained)
        current_index = ii

        #success case
        if event=="offer received":
            #print(f"{ii} Offer {offer_id} received on time {offer_time}")

            # --- start viewed case ---
```

```
            find_subsq_viewed = data_constrained.loc[  (data_constrained["value"]==offe
            try:
                find_subsq_viewed = find_subsq_viewed.reset_index().iloc[0] #picks most
            except:
                find_subsq_viewed = ""

            subsq_viewed_Trial = any(find_subsq_viewed)

            if subsq_viewed_Trial==True:
                #find_subsq_viewed = int(find_subsq_viewed)
                find_subsq_viewed = find_subsq_viewed["time"]
                #print(f"Subsequent Viewed: {subsq_viewed_Trial} on time {find_subsq_vi
            else:
                #print(f"Subsequent Viewed: {subsq_viewed_Trial}")
                find_subsq_viewed = np.nan
            # ---  end viewed case ---



            # --- start completed case ---

            find_subsq_completed = data_constrained.loc[  (data_constrained["value"]==o
            try:
                find_subsq_completed = find_subsq_completed.reset_index().iloc[0] #pick
            except:
                find_subsq_completed = ""

            subsq_completed_Trial = any(find_subsq_completed)

            if subsq_completed_Trial==True:
                find_subsq_completed = find_subsq_completed["time"]
            else:
                find_subsq_completed = np.nan

            # --- end completed case ---

            #classif


            if subsq_viewed_Trial==False and subsq_completed_Trial==True:
                    status = "success"
                    status_desc = "completed without viewing"


            if subsq_viewed_Trial==True and subsq_completed_Trial==False:
                    status = "failure"
                    status_desc = "viewed but not completed"

            if subsq_viewed_Trial==False and subsq_completed_Trial==False:
                    status = "failure"
                    status_desc = "not viewed and not completed"


            if subsq_viewed_Trial==True and subsq_completed_Trial==True:
                if find_subsq_viewed>find_subsq_completed:
                    status = "success"
                    status_desc = "completed without viewing"

                if find_subsq_viewed<=find_subsq_completed:
                    status = "success"
```

```
                    status_desc = "viewed and completed"


            #print(status,status_desc)
            selected_person_cont.append(selected_person)
            offer_id_cont.append(offer_id)
            status_cont.append(status)
            status_desc_cont.append(status_desc)

    return pd.Series([selected_person_cont,offer_id_cont,status_cont,status_desc_cont])
```

In [119…

```
import swifter
persons = pd.DataFrame(data_to_use['person'].unique(), columns={'person'})
s = pd.DataFrame()
s = persons['person'].swifter.apply(OfferClassif)
s = np.array(s)
final_classif_offers = pd.DataFrame()
pref = pd.DataFrame(np.array(pd.DataFrame(np.column_stack(s)).sum(axis=1))).T
final_classif_offers['person'] = pref[0][0]
final_classif_offers['offer_id'] = pref[1][0]
final_classif_offers['status'] = pref[2][0]
final_classif_offers['status_desc'] = pref[3][0]
final_classif_offers.to_csv('classifoffers.csv')
```

In [120…

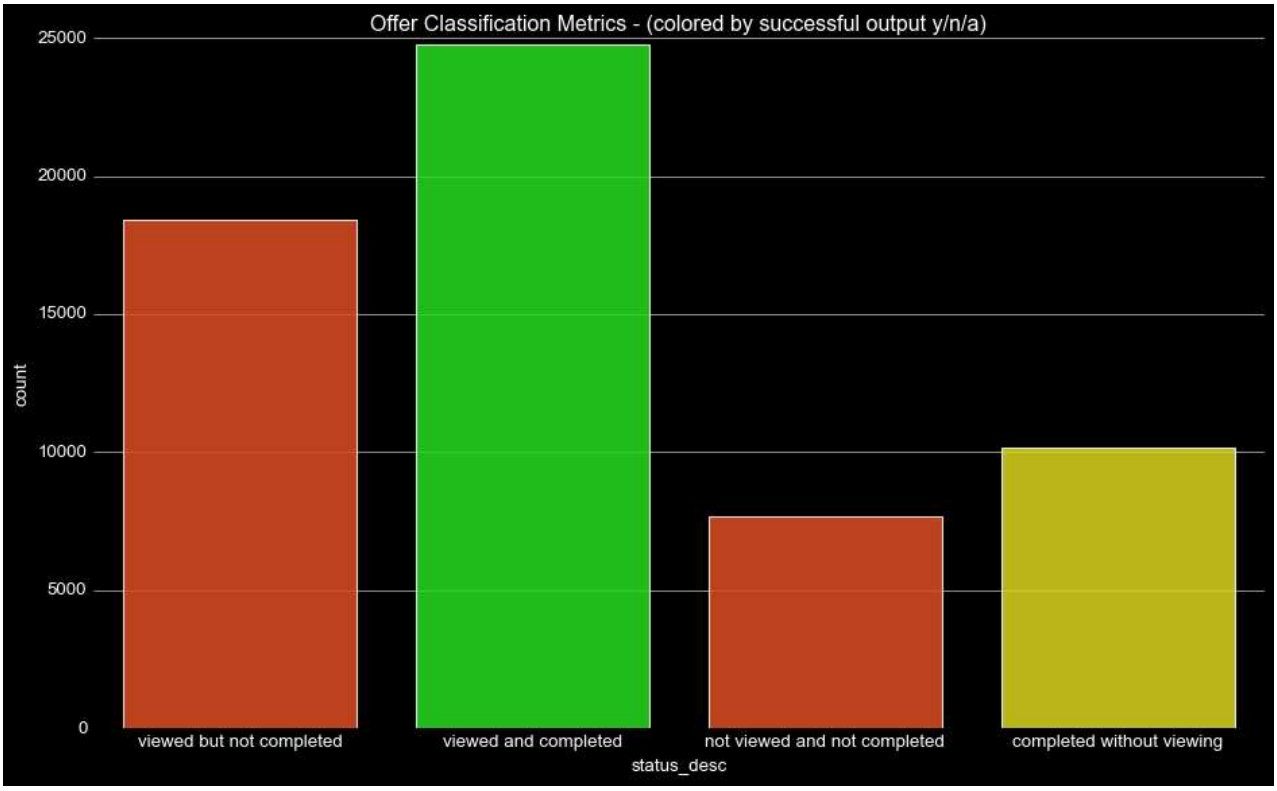
```
from urllib.request import Request, urlopen
req = Request('https://ivanachille.com/publicdata/starbucks-marketing/classifoffers.csv
req.add_header('User-Agent', 'Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:77.0) Gecko/20
content = urlopen(req)
final_classif_offers = pd.read_csv(content)
offers_class_to_use = final_classif_offers.merge(portfolio, on='offer_id')
offers_class_to_use = offers_class_to_use[offers_class_to_use['offer_type']!="informati
```

In [121…

```
plot_init()
plt.title('Offer Classification Metrics - (colored by successful output y/n/a)')
sns.countplot(x='status_desc',data=offers_class_to_use,
              palette=['#fc3d03','#0bfc03','#fc3d03','#fcf403'], alpha=.85,
              order=final_classif_offers['status_desc'].value_counts().reset_index()['i
```

Out[121…

```
<AxesSubplot:title={'center':'Offer Classification Metrics - (colored by successful outp
ut y/n/a)'}, xlabel='status_desc', ylabel='count'>
```

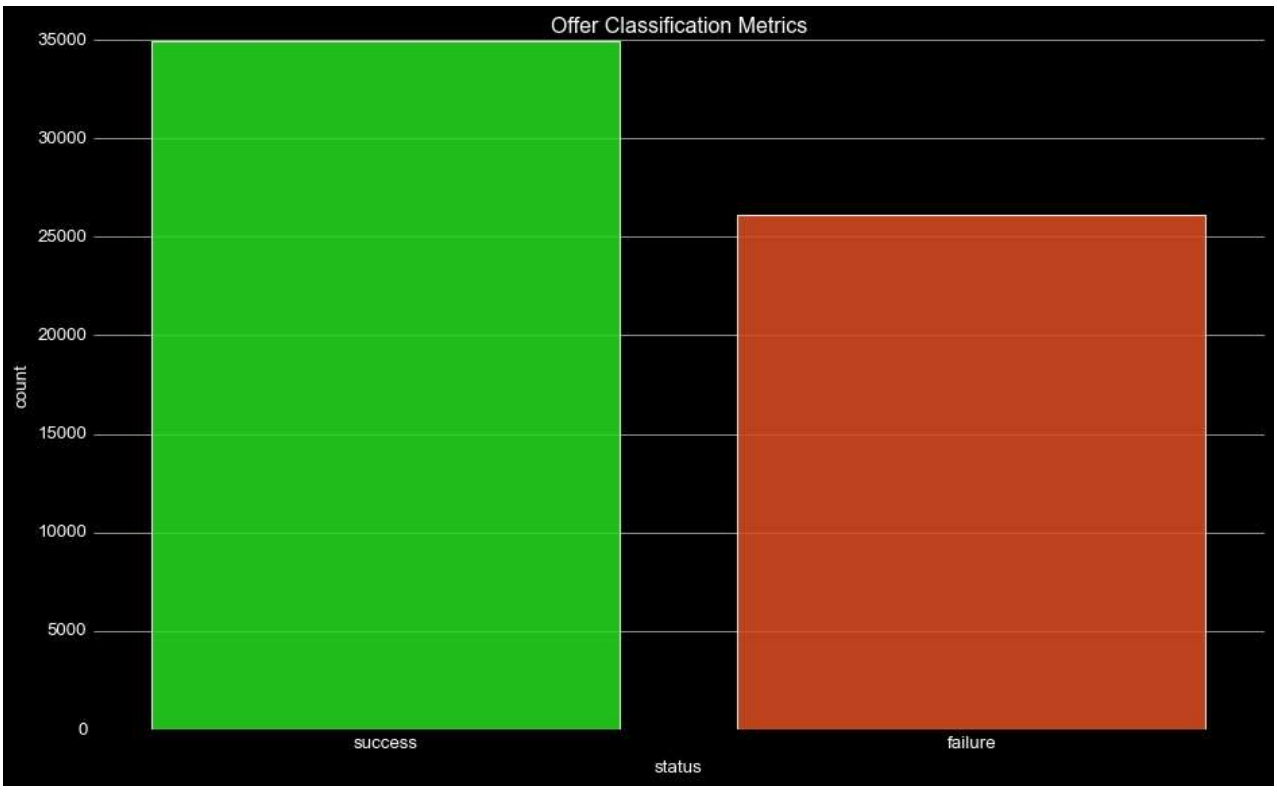


In [122…

```
plot_init()
plt.title('Offer Classification Metrics')
sns.countplot(x='status',data=offers_class_to_use, palette=['#0bfc03','#fc3d03'], alpha
```

Out[122…

```
<AxesSubplot:title={'center':'Offer Classification Metrics'}, xlabel='status', ylabel='c
ount'>
```



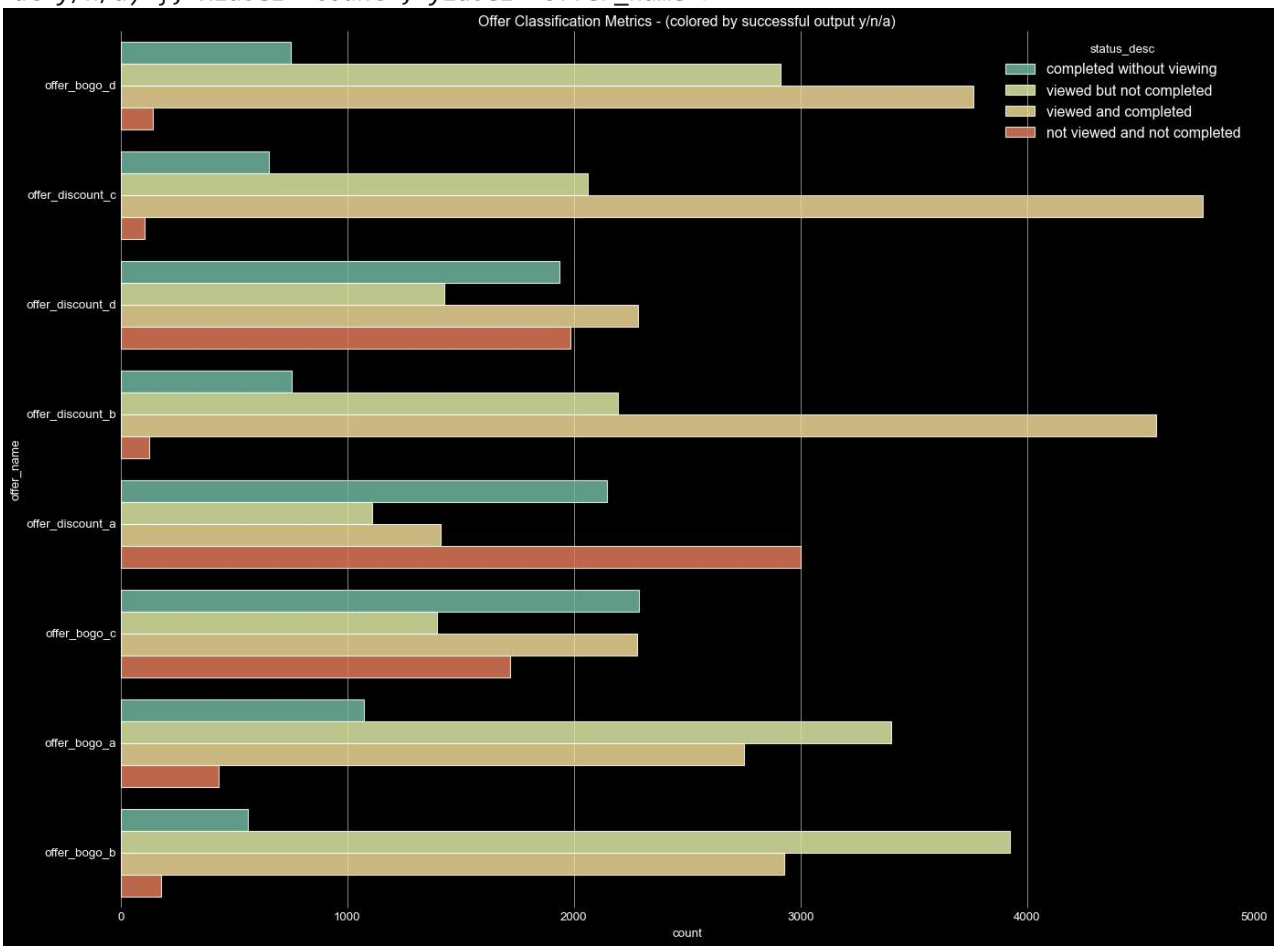
In [123…

```
plt.figure(figsize=(20,15))
plt.title('Offer Classification Metrics - (colored by successful output y/n/a)')
```

```
sns.countplot(y='offer_name', hue='status_desc', data=offers_class_to_use, palette='Spe
```

Out[123…

```
<AxesSubplot:title={'center':'Offer Classification Metrics - (colored by successful outp
ut y/n/a)'}, xlabel='count', ylabel='offer_name'>
```

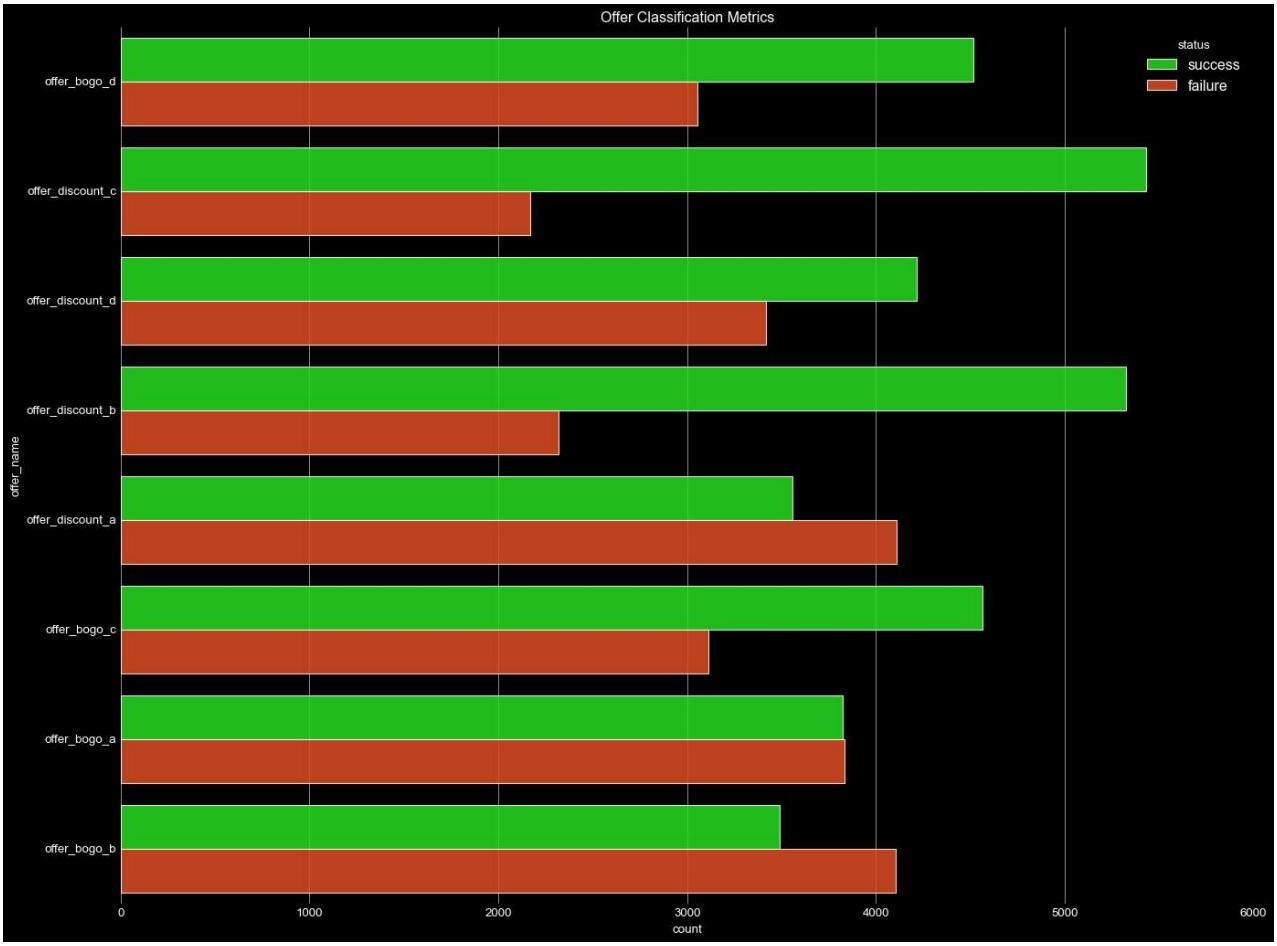


In [124…

```
plt.figure(figsize=(20,15))
plt.title('Offer Classification Metrics')
sns.countplot(y='offer_name', hue='status', data=offers_class_to_use, palette=['#0bfc03
```

Out[124…

```
<AxesSubplot:title={'center':'Offer Classification Metrics'}, xlabel='count', ylabel='of
fer_name'>
```

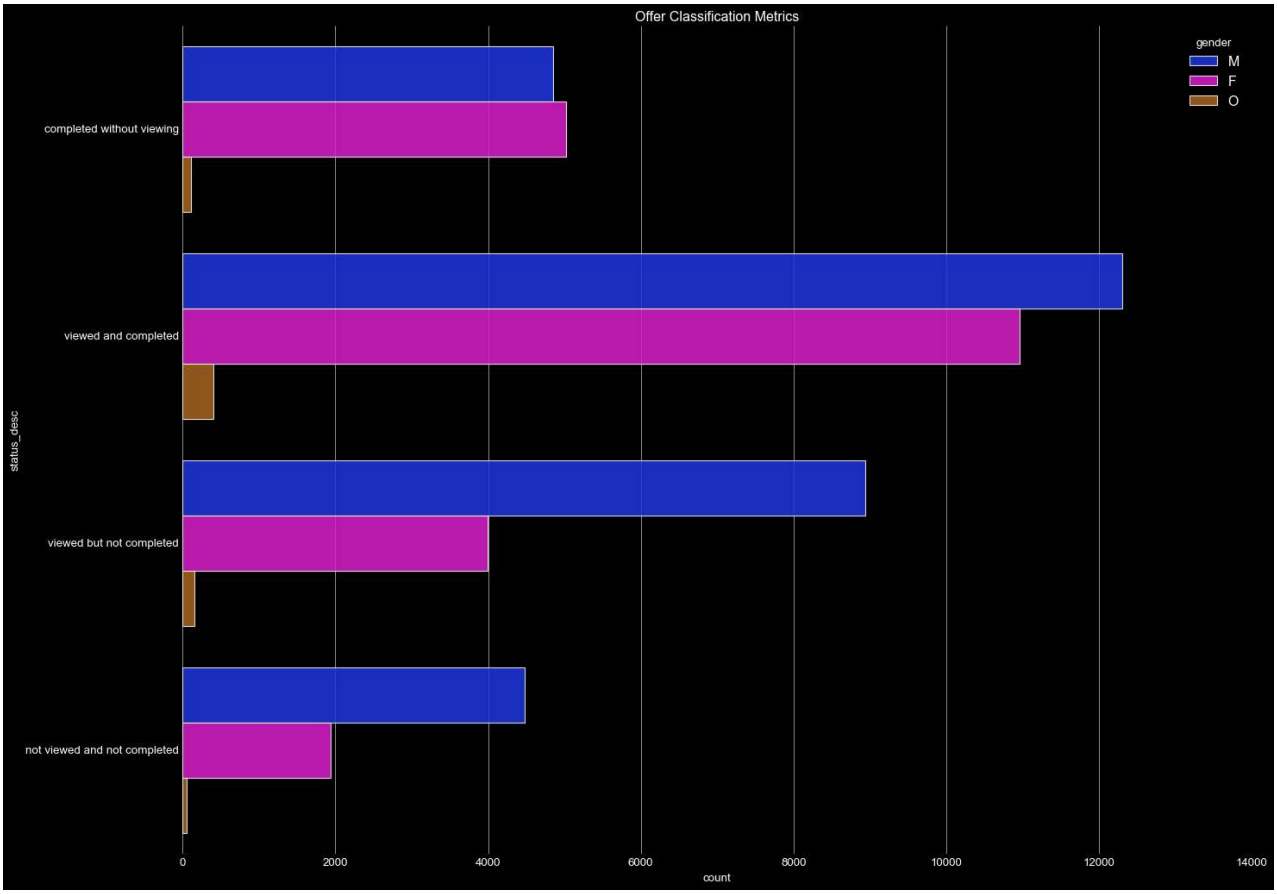


In [125…

```
plt.figure(figsize=(20,15))
plt.title('Offer Classification Metrics')
sns.countplot(y='status_desc', hue='gender', data=offers_class_to_use.merge(profile, on
```

Out[125…

```
<AxesSubplot:title={'center':'Offer Classification Metrics'}, xlabel='count', ylabel='st
atus_desc'>
```

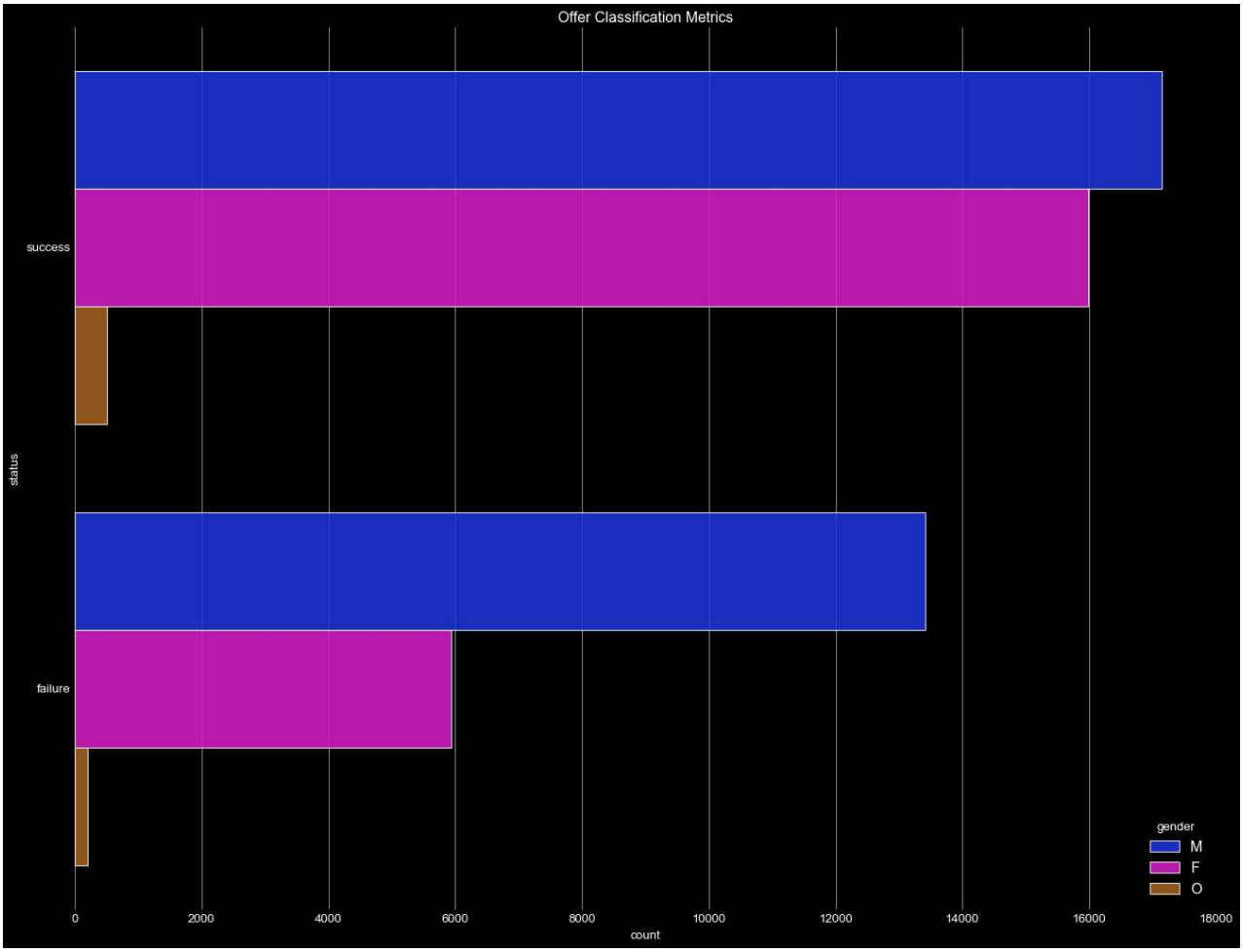


In [126…

```
plt.figure(figsize=(20,15))
plt.title('Offer Classification Metrics')
sns.countplot(y='status', hue='gender', data=offers_class_to_use.merge(profile, on='per
```

Out[126…

```
<AxesSubplot:title={'center':'Offer Classification Metrics'}, xlabel='count', ylabel='st
atus'>
```

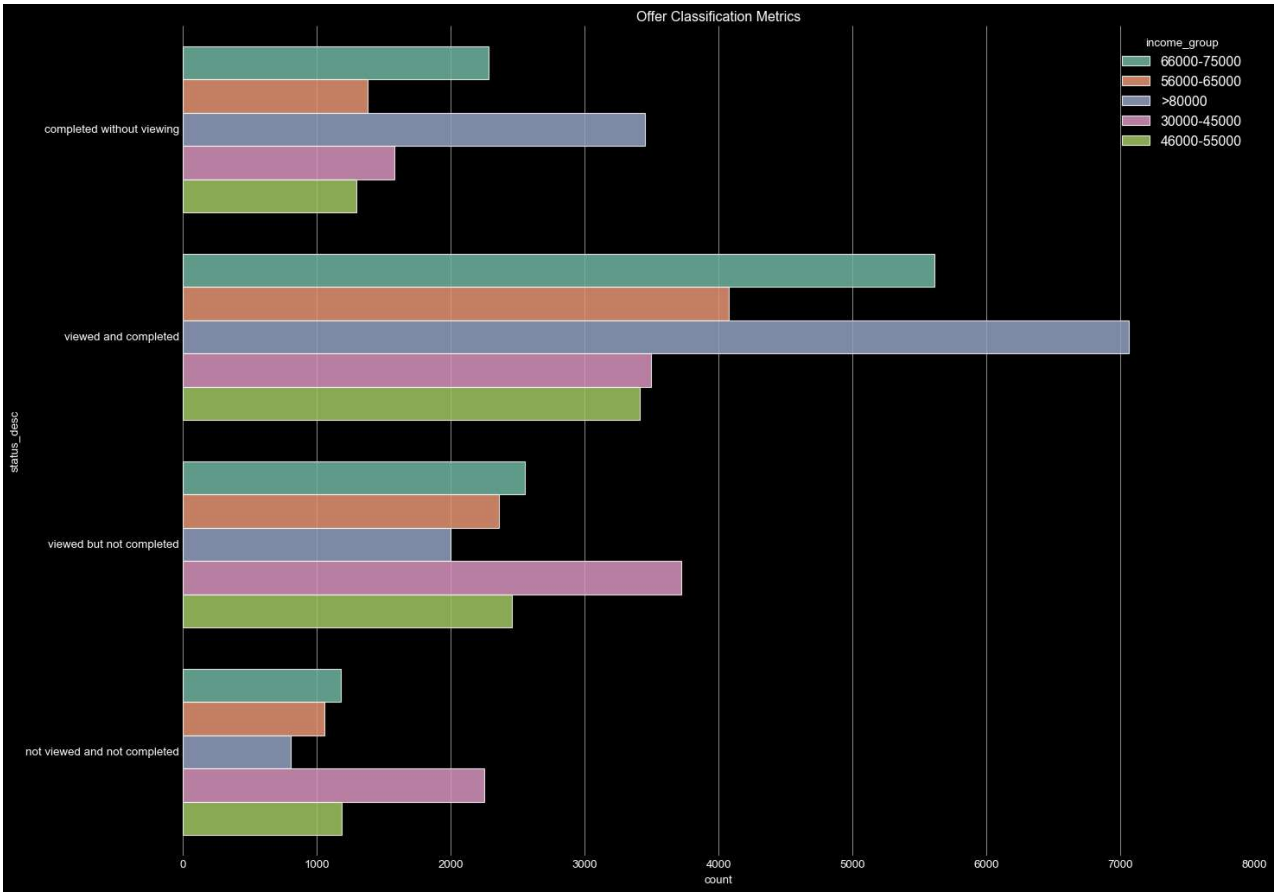


In [127…

```
plt.figure(figsize=(20,15))
plt.title('Offer Classification Metrics')
sns.countplot(y='status_desc', hue='income_group', data=offers_class_to_use.merge(profi
```

Out[127…

```
<AxesSubplot:title={'center':'Offer Classification Metrics'}, xlabel='count', ylabel='st
atus_desc'>
```

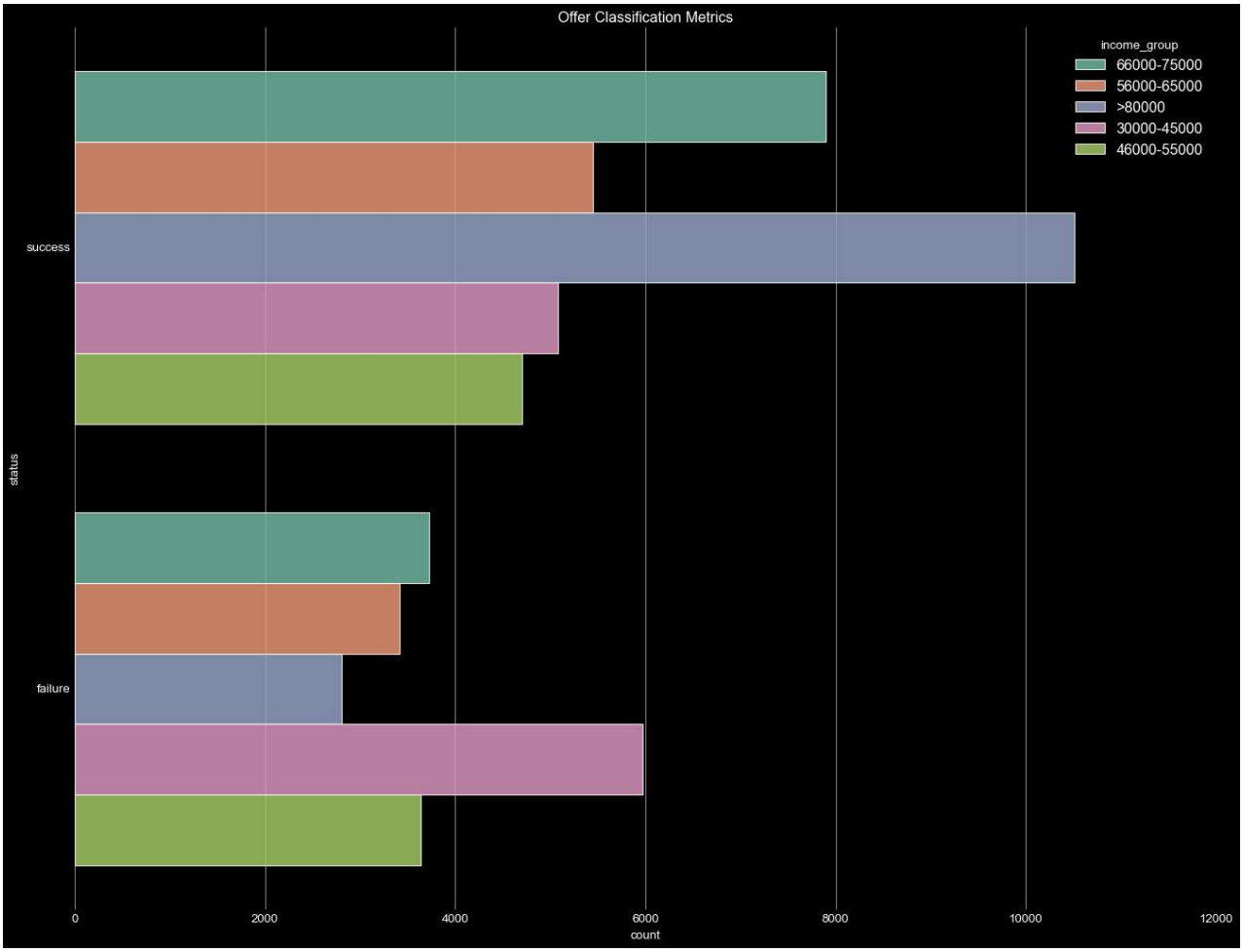


In [128…

```
plt.figure(figsize=(20,15))
plt.title('Offer Classification Metrics')
sns.countplot(y='status', hue='income_group', data=offers_class_to_use.merge(profile, o
```

Out[128…

```
<AxesSubplot:title={'center':'Offer Classification Metrics'}, xlabel='count', ylabel='st
atus'>
```

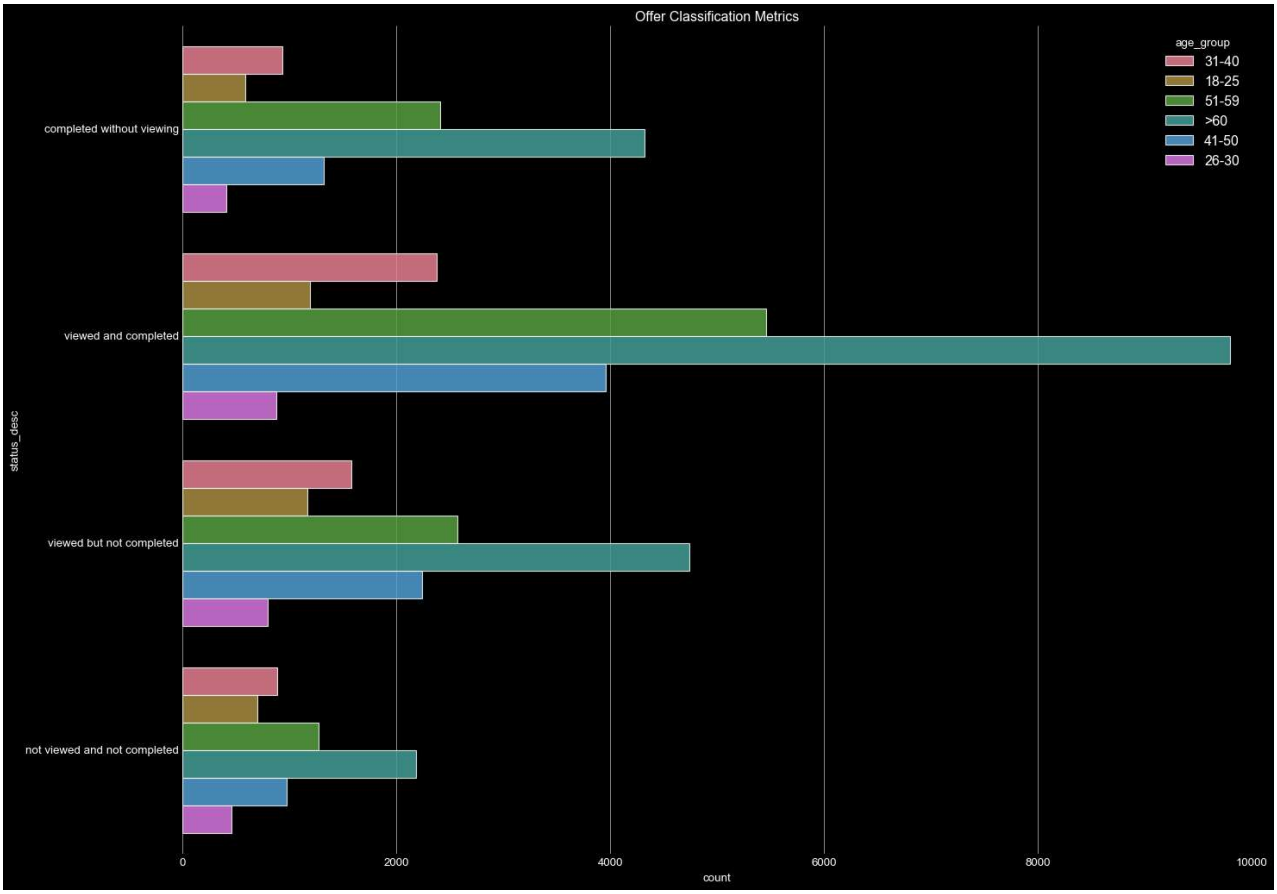


In [129…

```
plt.figure(figsize=(20,15))
plt.title('Offer Classification Metrics')
sns.countplot(y='status_desc', hue='age_group', data=offers_class_to_use.merge(profile,
```

Out[129…

```
<AxesSubplot:title={'center':'Offer Classification Metrics'}, xlabel='count', ylabel='st
atus_desc'>
```

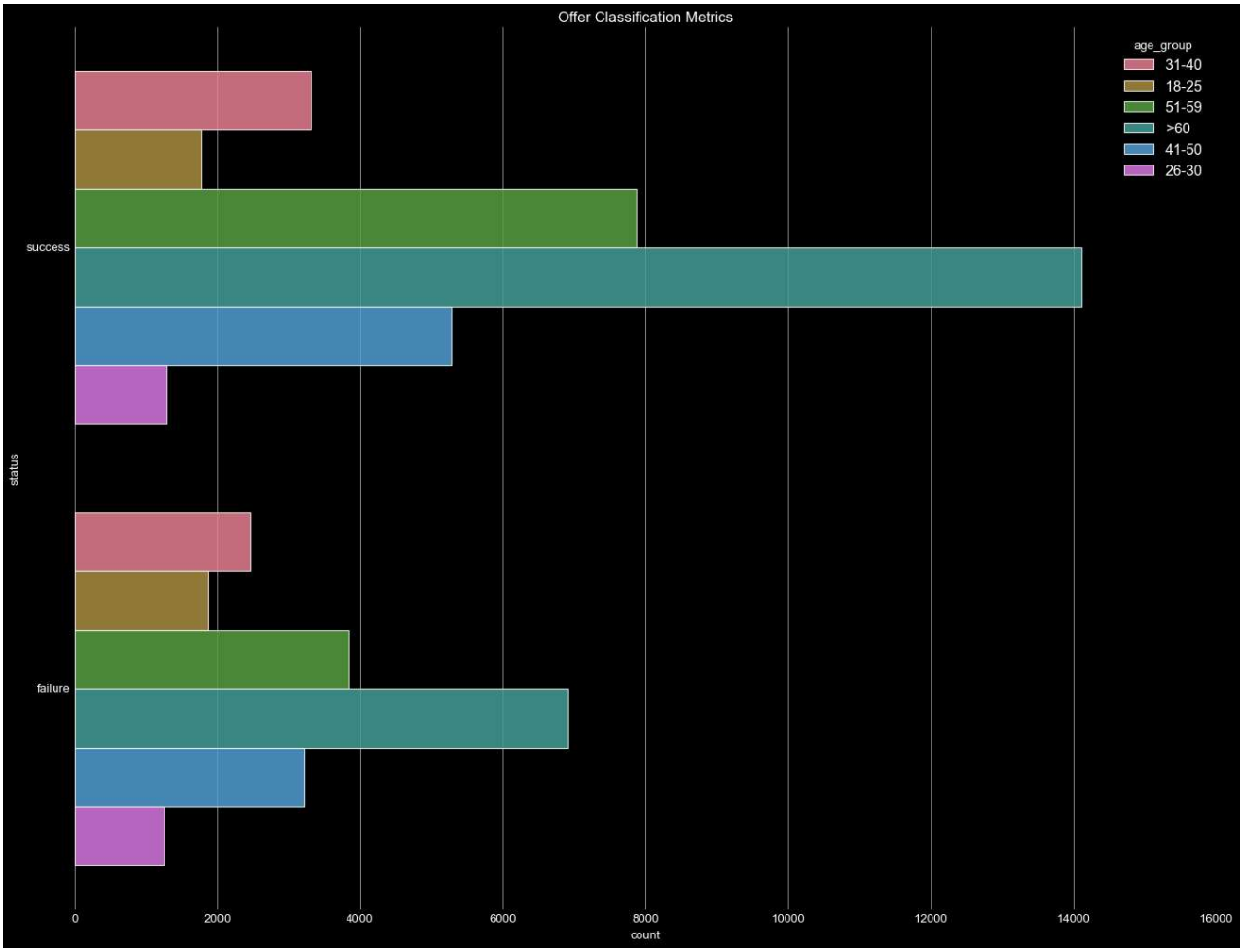


In [130…

```
plt.figure(figsize=(20,15))
plt.title('Offer Classification Metrics')
sns.countplot(y='status', hue='age_group', data=offers_class_to_use.merge(profile, on='
```

Out[130…

```
<AxesSubplot:title={'center':'Offer Classification Metrics'}, xlabel='count', ylabel='st
atus'>
```



In [131…
```
cols_to_use = ['person','transaction_value','dependent','offer_id']

mrg = offers_class_to_use.merge(trans_to_use[cols_to_use], on=["offer_id","person"])
offers_class_to_use.shape
```

Out[131…
```
(61042, 14)
```

In [132…
```
mrg.shape
```

Out[132…
```
(29458, 16)
```

In [133…
```
#correcting merge error on duplicate offers with different status (received more than o
mrg.loc[(mrg[mrg.status=='failure']).index, "transaction_value"] = 0
mrg.loc[(mrg[mrg.status=='failure']).index, "dependent"] = "n"

mrg['log_transaction'] = np.log(mrg['transaction_value'].astype('float'))
```

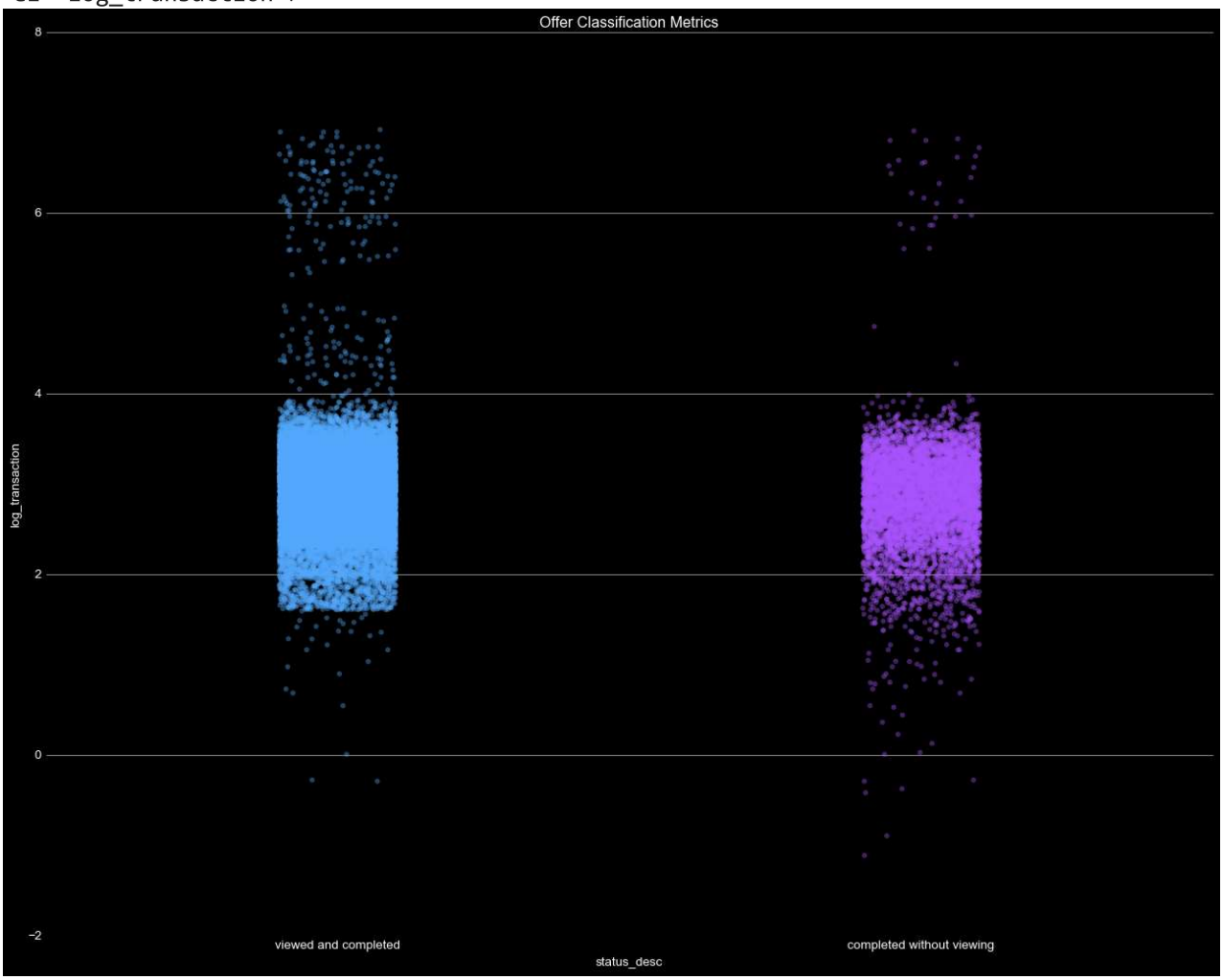
```
C:\Users\palla\anaconda3\lib\site-packages\pandas\core\arraylike.py:364: RuntimeWarning:
divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

In [134…
```
offers_success = mrg[mrg.status=='success']

plt.figure(figsize=(20,15))
```

```
plt.title('Offer Classification Metrics')
sns.stripplot(x='status_desc', y='log_transaction', data=offers_success, palette='cool'
```

Out[134…

```
<AxesSubplot:title={'center':'Offer Classification Metrics'}, xlabel='status_desc', ylab
el='log_transaction'>
```

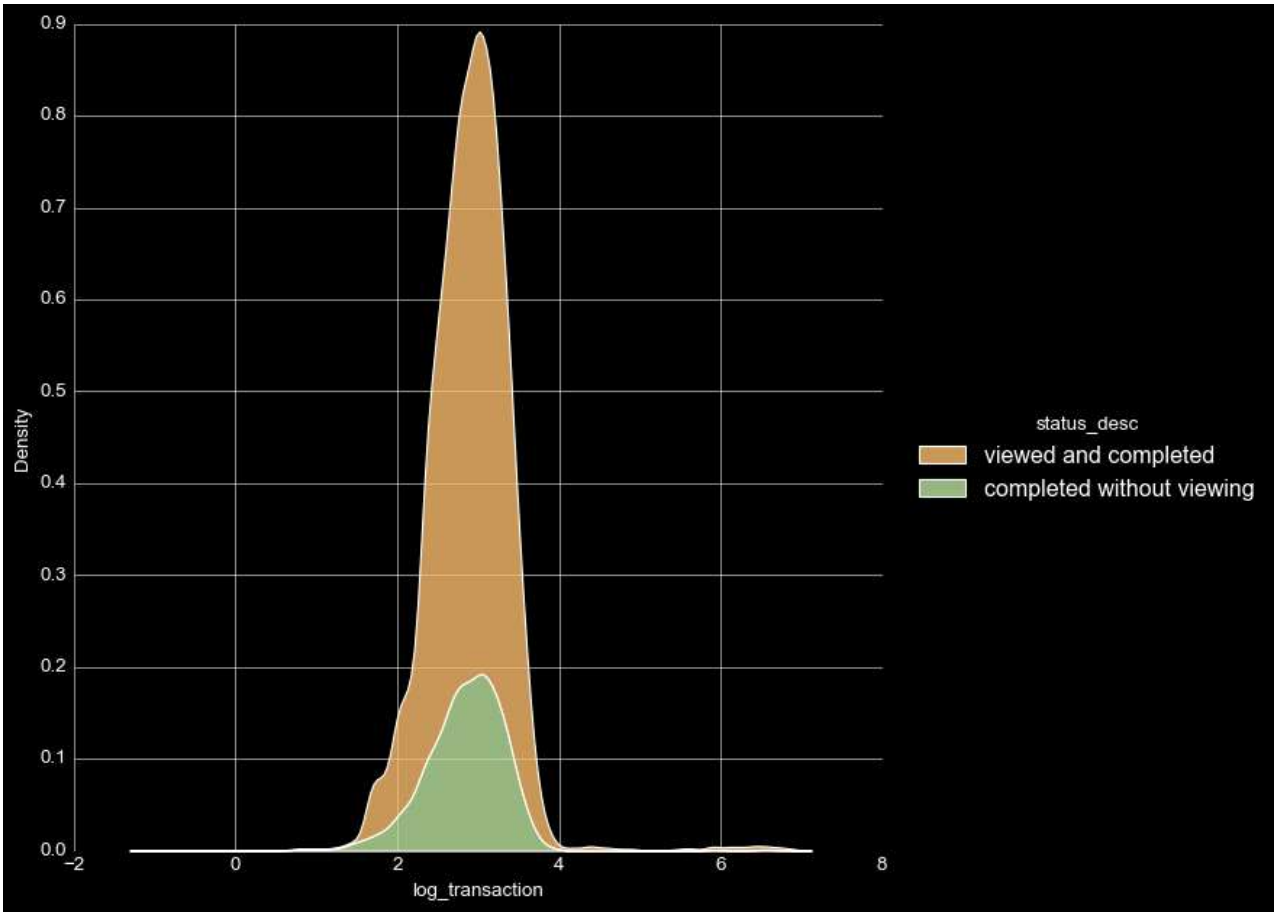


In [135…

```
sns.displot(x='log_transaction',hue='status_desc', data=offers_success, height=8, kind=
```

Out[135…

```
<seaborn.axisgrid.FacetGrid at 0x1ac9c24c5e0>
```



Modelling

The objective of the modeling stage is to determine whether an offer will fail or succeed, given specified entry parameters such as offer type, reward, difficulty, channels, and profile data, and to classify the different groups of customers. When sending an offer to a person, it is handy for sales purposes to know the probabilities of success of the offer for both the specific attributes of the person and the offer configuration.

In [136…

```
mrg_profile = mrg.merge(profile, on='person')
mrg_profile
```

Out[136…

	Unnamed: 0	person	offer_id	status	statu
0	18	0020ccbbb6d84e358d3414a3ff76cffd	f19421c1d4aa40978ebb69ca19b0e20d	success	view con
1	17	0020ccbbb6d84e358d3414a3ff76cffd	2298d6c36e964ae4a3e7e9706d1fb8c2	success	view con
2	20	0020ccbbb6d84e358d3414a3ff76cffd	9b98b8c7a33c4b65b9aebfe6a799e6d9	success	view con
3	32	004b041fbfe44859945daa2c7f79ee64	f19421c1d4aa40978ebb69ca19b0e20d	success	view con
4	33	004b041fbfe44859945daa2c7f79ee64	fafdcd668e3743c1bb461111dcafc2a4	success	view con

	Unnamed: 0	person	offer_id	status	statu
...	...	...	...	...	
29147	75593	fd91690fcf5b477aa5b5bdedb238e907	4d5c57ea9a6940dd891ad53e9dbe8da0	success	view con
29148	75596	fd91690fcf5b477aa5b5bdedb238e907	4d5c57ea9a6940dd891ad53e9dbe8da0	failure	view con
29149	75899	fec10086d3c54d929bdd8215df31d9f3	4d5c57ea9a6940dd891ad53e9dbe8da0	success	view con
29150	76000	ff1e4421836a43138e7a66b0f5f61879	4d5c57ea9a6940dd891ad53e9dbe8da0	success	view con
29151	76245	fff0f0aac6c547b9b263080f09a5586a	4d5c57ea9a6940dd891ad53e9dbe8da0	failure	view con

29152 rows × 23 columns

In [137...]
```
modeldata = mrg_profile[['offer_type','reward','difficulty','email','mobile','social','
                'gender','age','income','status']]
```

In [138...]
```
modeldata
```

Out[138...]

	offer_type	reward	difficulty	email	mobile	social	web	gender	age	income	status
0	bogo	5	5	1	1	1	1	F	24	60000.0	success
1	discount	3	7	1	1	1	1	F	24	60000.0	success
2	bogo	5	5	1	1	0	1	F	24	60000.0	success
3	bogo	5	5	1	1	1	1	F	55	74000.0	success
4	discount	2	10	1	1	1	1	F	55	74000.0	success
...	...	...	...	...	...	...	...	...	...	...	...
29147	bogo	10	10	1	1	1	1	F	69	74000.0	success
29148	bogo	10	10	1	1	1	1	F	69	74000.0	failure
29149	bogo	10	10	1	1	1	1	M	82	92000.0	success
29150	bogo	10	10	1	1	1	1	F	58	44000.0	success
29151	bogo	10	10	1	1	1	1	M	67	80000.0	failure

29152 rows × 11 columns

In [139...]
```
from sklearn import preprocessing
encoder = preprocessing.LabelEncoder()
modeldata['offer_type'] = encoder.fit_transform(modeldata['offer_type'])
```

```
modeldata['gender'] = encoder.fit_transform(modeldata['gender'])
modeldata['status'] = encoder.fit_transform(modeldata['status'])
```

In [140…

```
modeldata
```

Out[140…

	offer_type	reward	difficulty	email	mobile	social	web	gender	age	income	status
0	0	5	5	1	1	1	1	0	24	60000.0	1
1	1	3	7	1	1	1	1	0	24	60000.0	1
2	0	5	5	1	1	0	1	0	24	60000.0	1
3	0	5	5	1	1	1	1	0	55	74000.0	1
4	1	2	10	1	1	1	1	0	55	74000.0	1
…	…	…	…	…	…	…	…	…	…	…	…
29147	0	10	10	1	1	1	1	0	69	74000.0	1
29148	0	10	10	1	1	1	1	0	69	74000.0	0
29149	0	10	10	1	1	1	1	1	82	92000.0	1
29150	0	10	10	1	1	1	1	0	58	44000.0	1
29151	0	10	10	1	1	1	1	1	67	80000.0	0

29152 rows × 11 columns

In [141…

```
sub300 = modeldata[modeldata.status==1].head(500).append(modeldata[modeldata.status==0]
```

In [142…

```
sub300 = modeldata[modeldata.status==1].head(500).append(modeldata[modeldata.status==0]
from sklearn.manifold import TSNE
manifold = TSNE(n_components=2, verbose=1, perplexity=1, n_iter=300, random_state=42)
manifold_results = manifold.fit_transform(sub300)
```

```
[t-SNE] Computing 4 nearest neighbors...
[t-SNE] Indexed 1000 samples in 0.001s...
[t-SNE] Computed neighbors for 1000 samples in 0.003s...
[t-SNE] Computed conditional probabilities for sample 1000 / 1000
[t-SNE] Mean sigma: 0.000000
[t-SNE] KL divergence after 250 iterations with early exaggeration: 79.455482
[t-SNE] KL divergence after 300 iterations: 1.766485
```

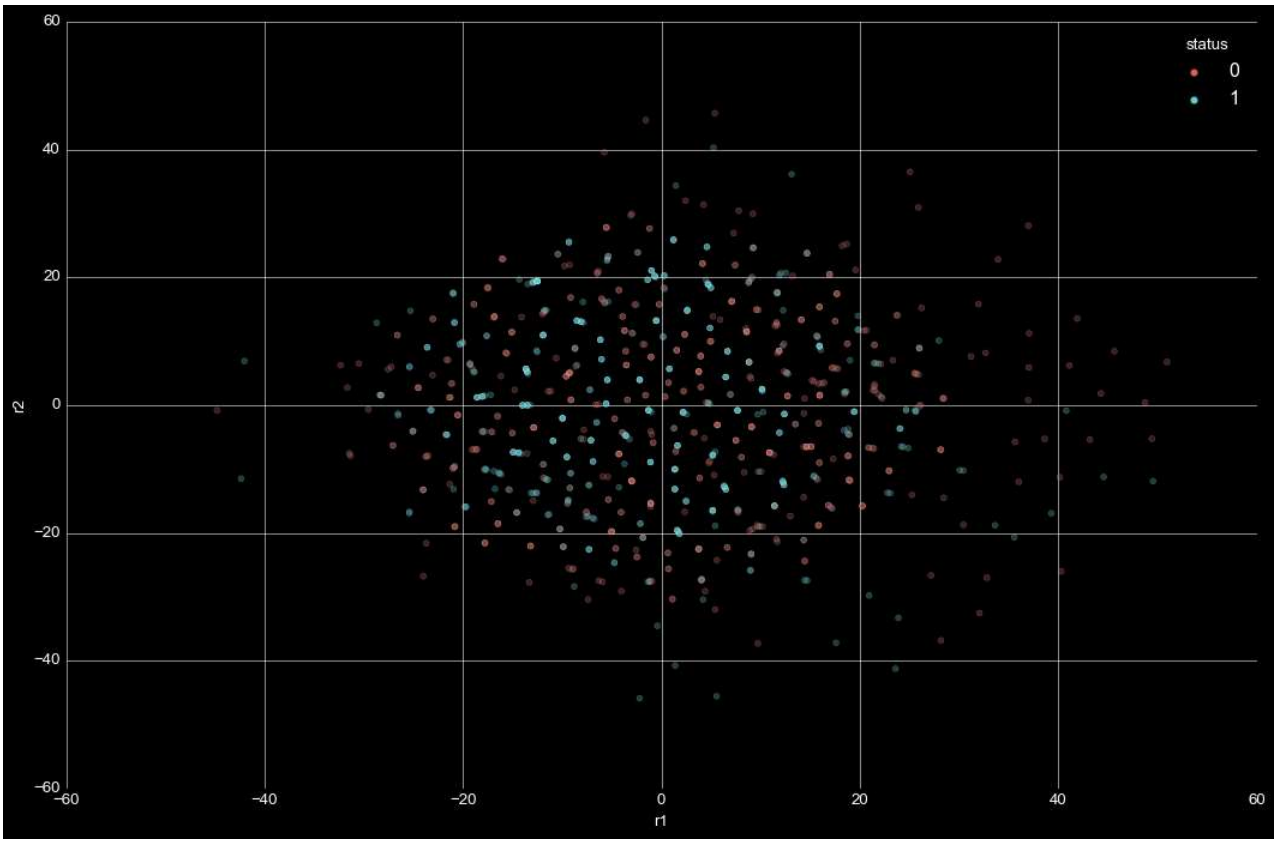
In [143…

```
sub300['r1'] = manifold_results[:,0]
sub300['r2'] = manifold_results[:,1]

plt.figure(figsize=(16,10))
sns.scatterplot(
    x="r1", y="r2",
    hue="status",
    palette='hls',
    data=sub300,
    legend="full",
```

```
    alpha=0.3
)
```

Out[143…

```
<AxesSubplot:xlabel='r1', ylabel='r2'>
```



In [144…

```
modeldata['status'].value_counts()/len(modeldata)
```

Out[144…

```
1    0.9599
0    0.0401
Name: status, dtype: float64
```

When merging all different datasets, a significant decrease in rows can be noticed because of the deletion of invalid rows during cleaning and other stages of data preprocessing. This is a necessary evil to access all the data dimensions and proceed with the modeling.

There is a vast imbalance between the classes, which must be dealt with to train the model correctly. We will use oversampling to correct this issue.

In [145…

```
from sklearn.model_selection import train_test_split
X = modeldata[['offer_type','reward','difficulty','email','mobile','social','web',
               'gender','age','income']]
y = modeldata['status']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=
```

In [146…

```
#from imblearn.datasets import SMOTE
from imblearn.over_sampling import SMOTE
sm = SMOTE(sampling_strategy='minority', random_state=42)
X_train_oversampled, y_train_oversampled = sm.fit_resample(X_train,y_train)
X_test_oversampled, y_test_oversampled = sm.fit_resample(X_test,y_test)
```

In [147…
```
len(X_train_oversampled)
```

Out[147…
```
39236
```

In [148…
```
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report
```

In [149…
```
from sklearn.svm import LinearSVC
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

classification_pipeline = make_pipeline(StandardScaler(),
                                        LinearSVC(random_state=42, tol=1e-6, max_iter=5
                                                  C=0.9))
```

In [150…
```
classification_pipeline.fit(X_train_oversampled,y_train_oversampled)
```

Out[150…
```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('linearsvc',
                 LinearSVC(C=0.9, max_iter=5000, random_state=42, tol=1e-06))])
```

In [151…
```
y_pred = classification_pipeline.predict(X_test_oversampled)
f1_score(y_pred,y_test_oversampled)
```

Out[151…
```
0.6719138359569179
```

In [152…
```
classification_report(y_pred,y_test_oversampled)
```

Out[152…
```
'              precision    recall  f1-score   support\n\n           0       0.85
0.67      0.75     10611\n           1       0.58      0.80      0.67      6119\n\n    a
ccuracy                           0.72     16730\n   macro avg       0.72      0.73
0.71     16730\nweighted avg       0.75      0.72      0.72     16730\n'
```

In [153…
```
from sklearn.neighbors import KNeighborsClassifier
knn_classifier = KNeighborsClassifier(n_neighbors=3)
```

In [154…
```
knn_classifier.fit(X_train_oversampled,y_train_oversampled)
```

Out[154…
```
KNeighborsClassifier(n_neighbors=3)
```

In [155…
```
y_pred = knn_classifier.predict(X_test_oversampled)

f1_score(y_pred,y_test_oversampled)
```

Out[155…
```
0.7667362976226002
```

In [156…
```
classification_report(y_pred,y_test_oversampled)
```

Out[156…
```
'              precision    recall  f1-score   support\n\n           0       0.52
0.87      0.65      4989\n           1       0.92      0.66      0.77     11741\n\n    a
ccuracy                           0.72     16730\n   macro avg       0.72      0.76
0.71     16730\nweighted avg       0.80      0.72      0.73     16730\n'
```

In [157…
```
from sklearn.svm import SVC
svm = make_pipeline(StandardScaler(),
                    SVC(random_state=42, gamma='scale', kernel="rbf", degree=7,
                       C=1.5))
```

In [158…
```
svm.fit(X_train_oversampled,y_train_oversampled)
```

Out[158…
```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('svc', SVC(C=1.5, degree=7, random_state=42))])
```

In [159…
```
y_pred = svm.predict(X_train_oversampled)
f1_score(y_pred,y_train_oversampled)
```

Out[159…
```
0.7914634759438943
```

In [160…
```
y_pred = svm.predict(X_test_oversampled)
```

In [161…
```
f1_score(y_pred,y_test_oversampled)
```

Out[161…
```
0.7885613207547171
```

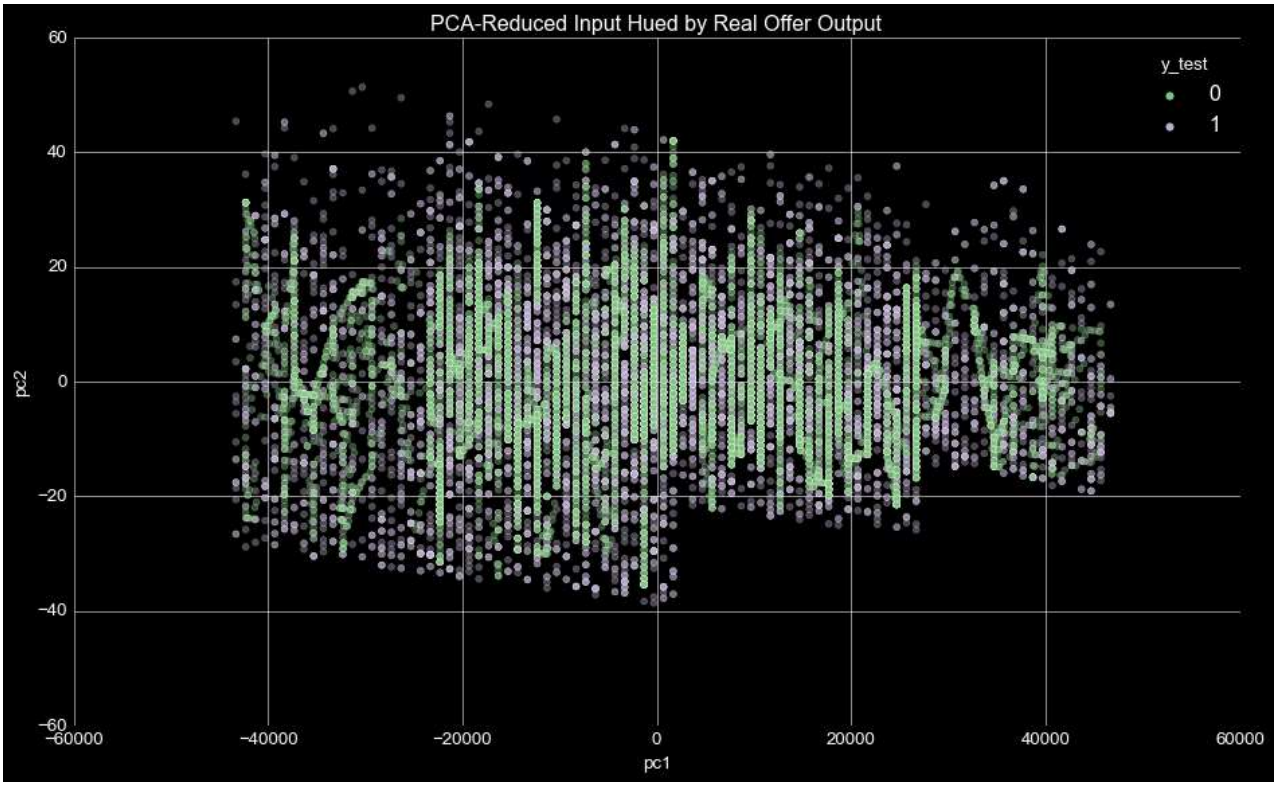
In [162…
```
classification_report(y_pred,y_test_oversampled)
```

Out[162…
```
'              precision    recall  f1-score   support\n\n           0       0.77
0.79      0.78      8135\n           1       0.80      0.78      0.79      8595\n\n    a
ccuracy                           0.79     16730\n   macro avg       0.79      0.79
0.79     16730\nweighted avg       0.79      0.79      0.79     16730\n'
```

In [163…
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pc = pca.fit_transform(X_test_oversampled)

pcadf = pd.DataFrame(pc, columns=['pc1','pc2'])
pcadf['y_test'] = y_test_oversampled
pcadf['y_pred'] = y_pred
```

In [164…
```
plot_init()
plt.title('PCA-Reduced Input Hued by Real Offer Output')
sns.scatterplot(x='pc1',y='pc2', hue='y_test', data=pcadf, palette='Accent', alpha=0.4)
```

Out[164…
```
<AxesSubplot:title={'center':'PCA-Reduced Input Hued by Real Offer Output'}, xlabel='pc
1', ylabel='pc2'>
```

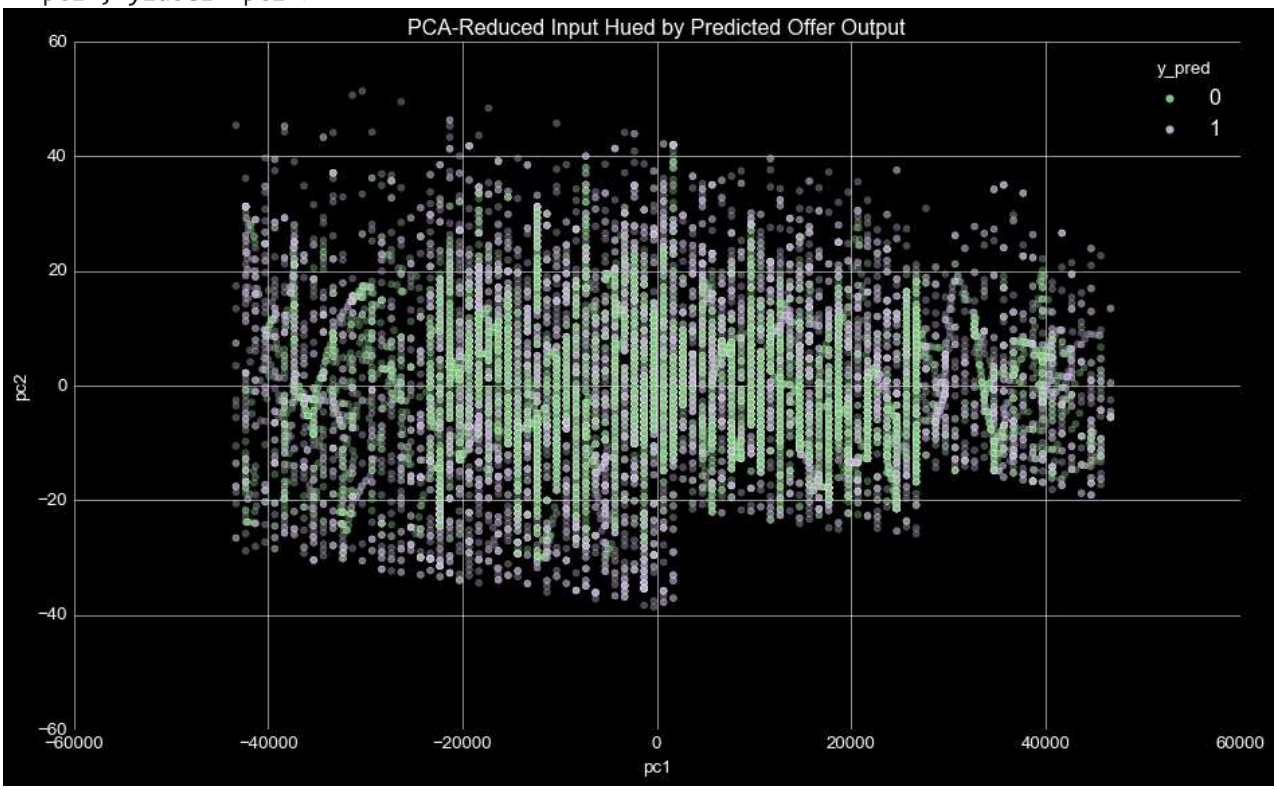


In [165…

```
plot_init()
plt.title('PCA-Reduced Input Hued by Predicted Offer Output')
sns.scatterplot(x='pc1',y='pc2', hue='y_pred', data=pcadf, palette='Accent', alpha=0.4)
```

Out[165…

```
<AxesSubplot:title={'center':'PCA-Reduced Input Hued by Predicted Offer Output'}, xlabel
='pc1', ylabel='pc2'>
```



SVC shows excellent results with a solid and balanced F1 Score. This classification model is reasonably good.

Customer Clustering

In [166…
```
modeldata = profile[['gender','age','income']]

from sklearn import preprocessing
encoder = preprocessing.LabelEncoder()
modeldata['gender'] = encoder.fit_transform(modeldata['gender'])
X_train, X_test= train_test_split(X, test_size=0.30, random_state=42)
```

In [167…
```
from sklearn.cluster import DBSCAN
```

In [168…
```
dbscan = DBSCAN(eps=200, min_samples=4).fit(modeldata)
```

In [169…
```
from sklearn.metrics import silhouette_score
silhouette_score(modeldata,dbscan.labels_)
```

Out[169…
```
0.9814642895434896
```

In [170…
```
len(set(dbscan.labels_))
```

Out[170…
```
91
```

In [171…
```
class_predict_md = dbscan.fit(modeldata)
modeldata['class'] = class_predict_md.labels_
```

In [172…
```
import plotly.express as px
fig = px.scatter_3d(modeldata, x='age', y='income', z='gender',
                    color='class', size_max=.05)
fig.show()
```

In [173…]
```
modeldata
```

Out[173…]

	gender	age	income	class
1	0	55	112000.0	0
3	0	75	100000.0	1
5	1	68	70000.0	2
8	1	65	53000.0	3
12	1	58	51000.0	4
…	…	…	…	…
16995	0	45	54000.0	46
16996	1	61	72000.0	34
16997	1	49	73000.0	48
16998	0	83	50000.0	28
16999	0	62	82000.0	66

14825 rows × 4 columns

In [174…]
```
profile_classed = pd.concat((modeldata,profile), axis=1)
mrg_profile_classed = mrg.merge(profile_classed, on='person')
mrg_profile_classed = mrg_profile_classed[mrg_profile_classed.status=='success']
mrg_profile_classed
```

Out[174…]

	Unnamed: 0	person	offer_id	status	statu
0	18	0020ccbbb6d84e358d3414a3ff76cffd	f19421c1d4aa40978ebb69ca19b0e20d	success	view con
1	17	0020ccbbb6d84e358d3414a3ff76cffd	2298d6c36e964ae4a3e7e9706d1fb8c2	success	view con
2	20	0020ccbbb6d84e358d3414a3ff76cffd	9b98b8c7a33c4b65b9aebfe6a799e6d9	success	view con

	Unnamed: 0	person	offer_id	status	statu
3	32	004b041fbfe44859945daa2c7f79ee64	f19421c1d4aa40978ebb69ca19b0e20d	success	view con
4	33	004b041fbfe44859945daa2c7f79ee64	fafdcd668e3743c1bb461111dcafc2a4	success	view con
...	...	...	...	...	
29145	75503	fd3e1ba4df4b4db0af9cf727eb1651ac	4d5c57ea9a6940dd891ad53e9dbe8da0	success	view con
29146	75504	fd3e1ba4df4b4db0af9cf727eb1651ac	4d5c57ea9a6940dd891ad53e9dbe8da0	success	view con
29147	75593	fd91690fcf5b477aa5b5bdedb238e907	4d5c57ea9a6940dd891ad53e9dbe8da0	success	view con
29149	75899	fec10086d3c54d929bdd8215df31d9f3	4d5c57ea9a6940dd891ad53e9dbe8da0	success	view con
29150	76000	ff1e4421836a43138e7a66b0f5f61879	4d5c57ea9a6940dd891ad53e9dbe8da0	success	view con

27983 rows × 27 columns

In [175...]
```
mrg_profile_classed['transaction_value'] = mrg_profile_classed['transaction_value'].ast
mrg_profile_classed[['transaction_value','class']].groupby("class")['transaction_value'
```

Out[175...]
```
count    91.000000
mean     23.242979
std       8.565376
min       8.631111
25%      16.826451
50%      21.729073
75%      27.328544
max      58.320000
Name: transaction_value, dtype: float64
```

User Test Case

Let us assume we are creating an efficient offer for middle-aged women and an expectations report of the offer before launching it.

In [176...]
```
middle_aged_woman = mrg_profile.loc[ (mrg_profile['age']>=40) & (mrg_profile['age']<=60
```

In [177...]
```
middle_aged_woman.columns
```

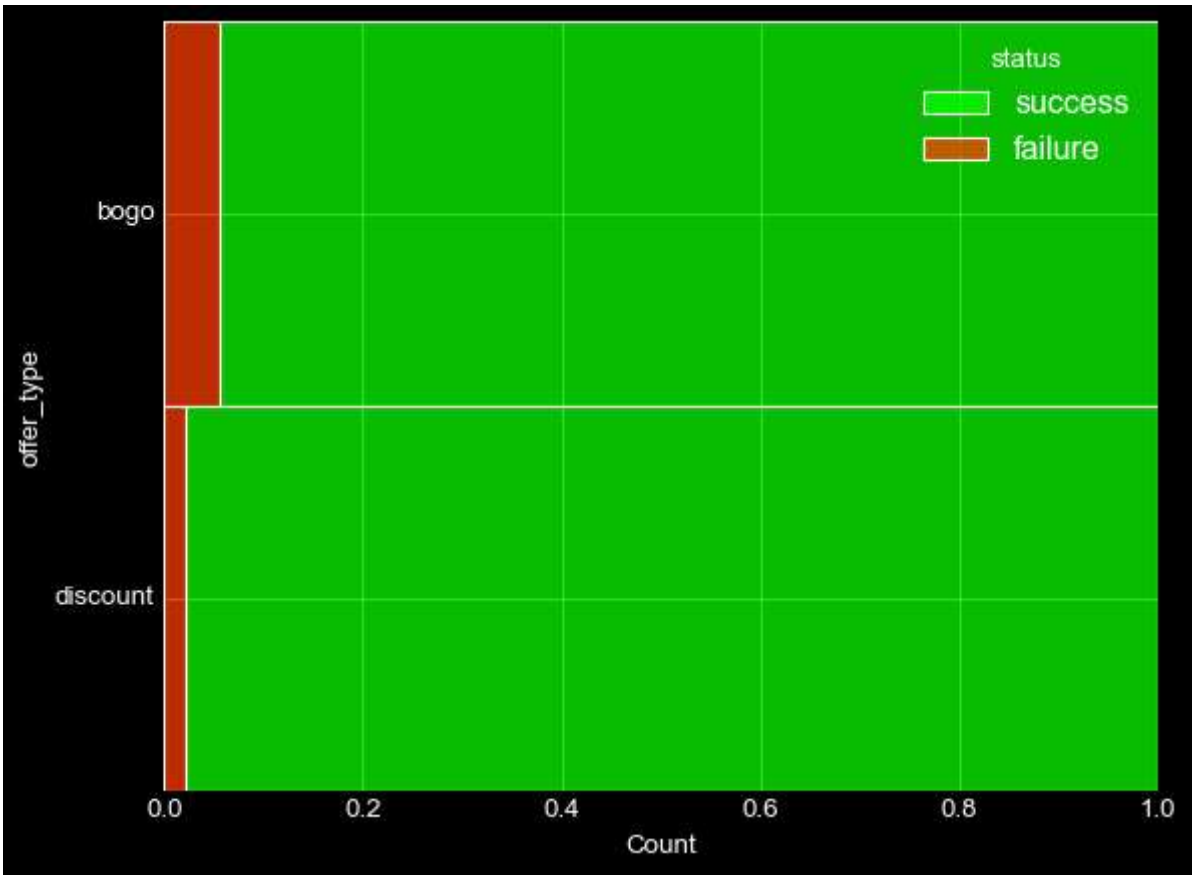
Out[177...]
```
Index(['Unnamed: 0', 'person', 'offer_id', 'status', 'status_desc', 'reward',
       'difficulty', 'duration', 'offer_type', 'email', 'mobile', 'social',
       'web', 'offer_name', 'transaction_value', 'dependent',
       'log_transaction', 'gender', 'age', 'became_member_on', 'income',
```

```
       'age_group', 'income_group'],
      dtype='object')
```

In [178…
```
middle_aged_woman['difficulty_cat'] = middle_aged_woman['difficulty'].astype('string')
middle_aged_woman['reward_cat'] = middle_aged_woman['reward'].astype('string')
```

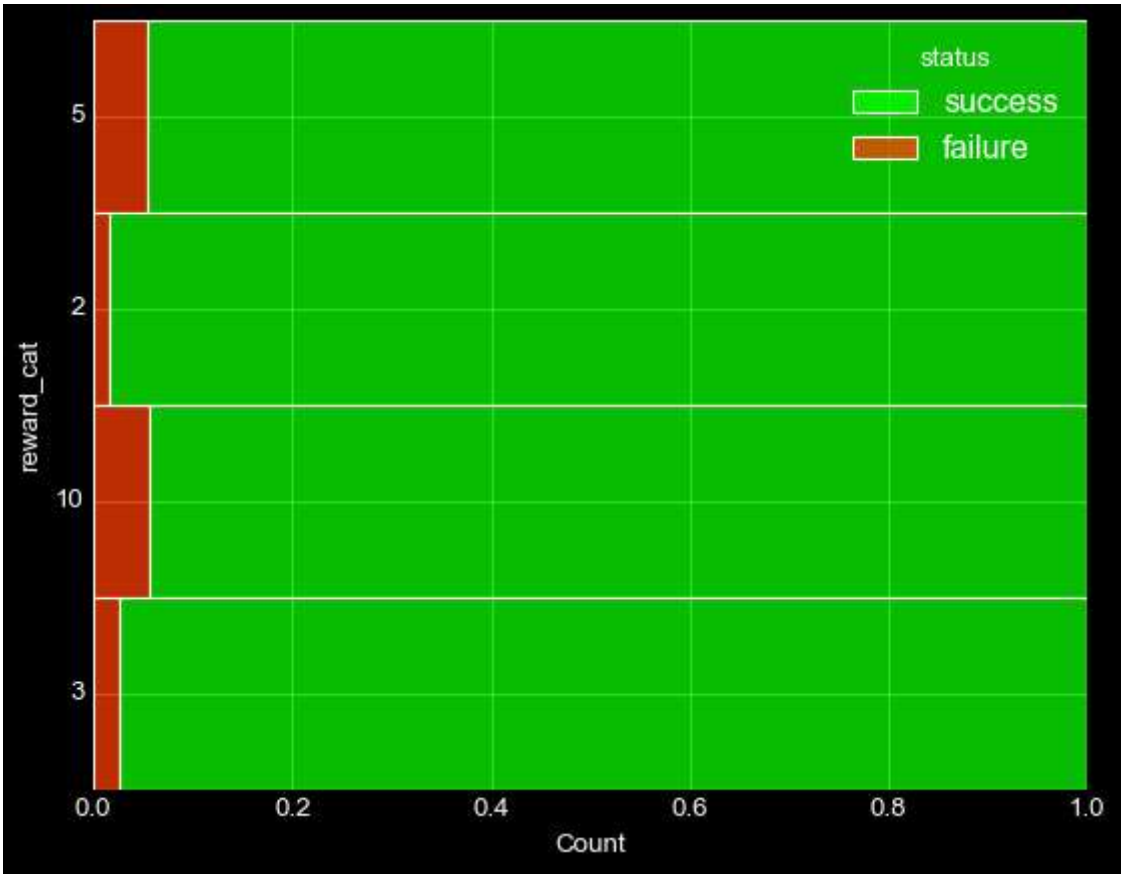
In [179…
```
sns.histplot(y='offer_type', hue='status', multiple='fill',palette=['#0bfc03','#fc3d03'
```

Out[179…
```
<AxesSubplot:xlabel='Count', ylabel='offer_type'>
```



In [180…
```
sns.histplot(y='reward_cat', hue='status', multiple='fill',palette=['#0bfc03','#fc3d03'
```

Out[180…
```
<AxesSubplot:xlabel='Count', ylabel='reward_cat'>
```

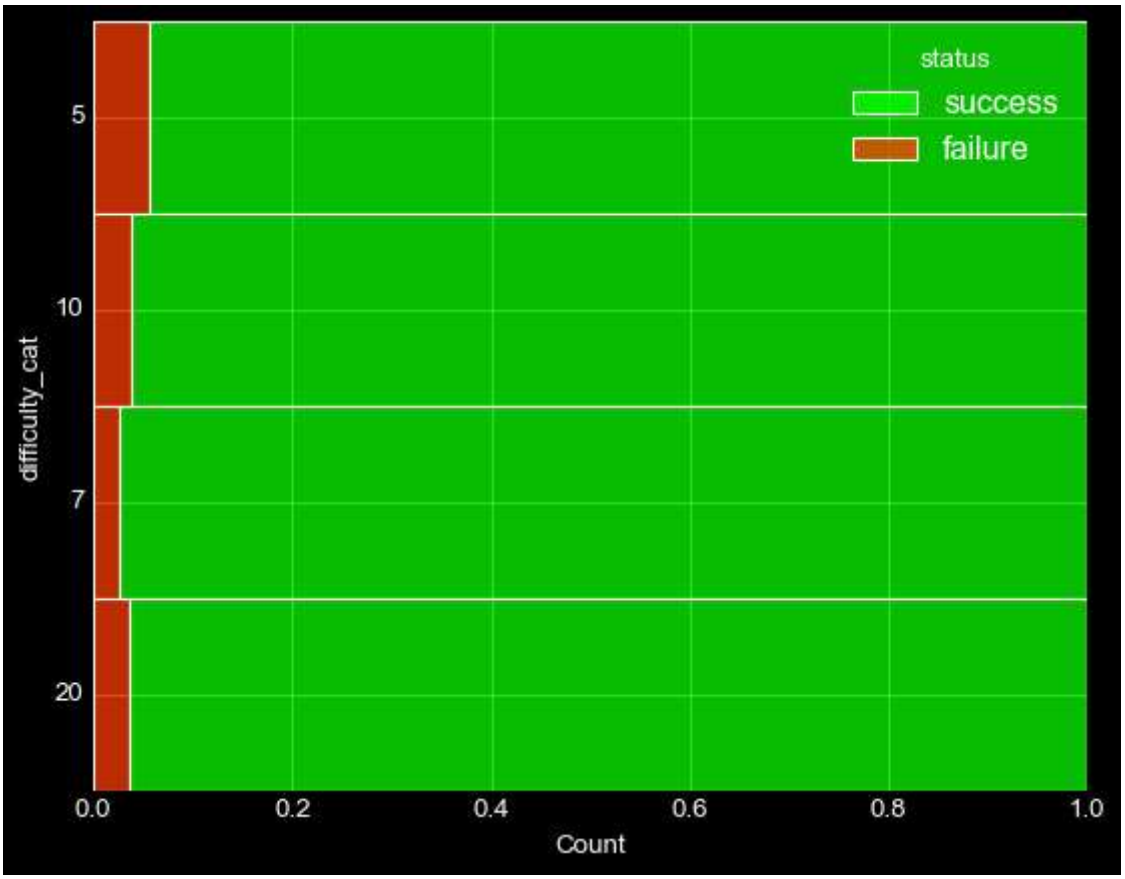


In [181…

```
sns.histplot(y='difficulty_cat', hue='status', multiple='fill',palette=['#0bfc03','#fc3
```

Out[181…

```
<AxesSubplot:xlabel='Count', ylabel='difficulty_cat'>
```



In [182…
```
fict_offer = pd.DataFrame([2,7,10,'discount',1,1,1,1], columns=['disco'])
middle_aged_woman_sample = middle_aged_woman[['gender','age','income']].sample(n=1000,
#convert gender to model pretrained category
middle_aged_woman_sample['gender'] = middle_aged_woman_sample['gender'].replace('F',0)
middle_aged_woman_sample
```

Out[182…

	gender	age	income
18643	0	43	70000.0
27181	0	57	102000.0
3450	0	55	66000.0
16869	0	60	93000.0
12124	0	49	55000.0
…	…	…	…
18645	0	43	70000.0
10336	0	54	37000.0
15799	0	57	100000.0
9974	0	49	82000.0
22668	0	56	55000.0

1000 rows × 3 columns

In [183…
```
middle_aged_woman_sample = middle_aged_woman_sample.reset_index().drop('index',axis=1)
offer_type = np.full((1000,),1).tolist()
reward = np.full((1000,),2).tolist()
difficulty = np.full((1000,),7).tolist()
duration = np.full((1000,),10).tolist()

channel_true = np.full((1000,),1).tolist()
offer_s_sm = pd.DataFrame({'offer_type': offer_type,
                           'reward': reward,
                           'difficulty': difficulty,
                           'email': channel_true,
                          'mobile': channel_true,
                           'social': channel_true,
                           'web': channel_true})
middle_aged_woman_final = pd.concat((offer_s_sm,middle_aged_woman_sample), axis=1)
maw_predict = svm.predict(middle_aged_woman_final)
middle_aged_woman_final['predicted'] = maw_predict
middle_aged_woman_final['predicted'].value_counts()
```

Out[183…
```
0    989
1     11
Name: predicted, dtype: int64
```

When thrown these offer parameters, the classification model predicted that this offer, tested on 1000 middle-aged women, would succeed on 989 of them and fail on 11. So this is would be a pretty good offer and would likely yield similar results in the real world. Different types of testing

and fine-tuning can be done using this model to simulate and produce suitable offers that are likely to succeed in real life.

In [ ]: