Project

Summary

A & D, a streaming services that's been loosing more customers than usual the past few months and would like to use data science to figue out how to reduce customer churn

- I have access to data on A & D Music's customer;s including subscription details and music listenning history.

- Tasks to gather, clean, and explore the data to provide insights about the recent customer churn issues, the prepare it for modelling in the future ¶

1. Scope the data science project
2. Gather the data in python
3. Clean the data in pythonthe data
4. Explore & Visualize the data
5. Preparethe data for modelling

1. Scope the Project

My plan is to use a supervised learning technique to predict which customers are most likely to cancel their subscription using **the past three months of customer data which includes subscription and listening history**.

2. Gather Data

Read the following files into Python:

- Customer data: A & D_music_customers.csv
- Listing history: A & D_music_listening_history.xlsx

In [115]:
```
# Read in the customer data
import pandas as pd
customers = pd.read_csv('A&D_music_customers.csv')
customers.head()
```

Out[115]:

	Customer_ID	Customer_Name	Email	Member_Since	Subscription_Plan
0	5001	Harmony Greene	Email: harmonious.vibes@email.com	3/13/2023	Basic (Ads)
1	5002	Aria Keys	Email: melodious.aria@email.edu	3/13/2023	NaN
2	5004	Lyric Bell	Email: rhythmical.lyric@email.com	3/13/2023	NaN
3	5267	Rock Bassett	Email: groovy.rock@email.com	3/20/2023	Basic (Ads)
4	5338	Rhythm Dixon	Email: beats.by.rhythm@email.edu	3/20/2023	NaN

In [116]:
```
# Read in the listening history
listenning_history = pd.read_excel('A&D_music_listening_history.xlsx')
listenning_history.head()
```

Out[116]:

	Customer_ID	Session _ID	Audio_Order	Audio_ID	Audio_Type
0	5001	100520	1	101	Song
1	5001	100520	2	102	Song
2	5001	100520	3	103	Song
3	5001	100520	4	104	Song
4	5001	100520	5	105	Song

In [117]:
```
# Where might I find listening history data beyond the ID's?
# I'm checking the other tabs
# Read in the audio data
audio = pd.read_excel('A&D_music_listening_history.xlsx', sheet_name=1)
audio.head()
```

Out[117]:

	ID	Name	Genre	Popularity
0	Song-101	Dance All Night	Pop	1
1	Song-102	Unbreakable Beat	Pop	2
2	Song-103	Sunset Boulevard	Pop Music	5
3	Song-104	Glowing Hearts	Pop Music	10
4	Song-105	Pop Rocks	Pop Music	52

In [118]:
```
# Read in the session data
sessions = pd.read_excel('A&D_music_listening_history.xlsx', sheet_name=2)
sessions.head()
```

Out[118]:

	Session_ID	Session_Log _In_Time
0	100520	2023-03-13 18:29:00
1	100522	2023-03-13 22:15:00
2	100525	2023-03-14 10:01:00
3	100527	2023-03-13 14:14:00
4	100538	2023-03-21 12:23:00

3. Clean Data

a. Convert Data Types

Check the data types of the data in the tables and convert to numeric and datetime values as necessary.

In [5]:
```
customers.head()
```

Out[5]:

	Customer_ID	Customer_Name	Email	Member_Since	Subscription_Plan
0	5001	Harmony Greene	Email: harmonious.vibes@email.com	3/13/2023	Basic (Ads)
1	5002	Aria Keys	Email: melodious.aria@email.edu	3/13/2023	NaN
2	5004	Lyric Bell	Email: rhythmical.lyric@email.com	3/13/2023	NaN
3	5267	Rock Bassett	Email: groovy.rock@email.com	3/20/2023	Basic (Ads)
4	5338	Rhythm Dixon	Email: beats.by.rhythm@email.edu	3/20/2023	NaN

In [6]:
```
# Check the data types
customers.dtypes
```

Out[6]:
```
Customer_ID           int64
Customer_Name        object
Email                object
Member_Since         object
Subscription_Plan    object
Subscription_Rate    object
Discount?            object
Cancellation_Date    object
dtype: object
```

In [7]:
```
listenning_history.head()
```

Out[7]:

	Customer_ID	Session _ID	Audio_Order	Audio_ID	Audio_Type
0	5001	100520	1	101	Song
1	5001	100520	2	102	Song
2	5001	100520	3	103	Song
3	5001	100520	4	104	Song
4	5001	100520	5	105	Song

In [8]:
```
listenning_history.dtypes
```

Out[8]:
```
Customer_ID     int64
Session _ID     int64
Audio_Order     int64
Audio_ID        int64
Audio_Type     object
dtype: object
```

In [9]:
```
audio.head()
```

Out[9]:

	ID	Name	Genre	Popularity
0	Song-101	Dance All Night	Pop	1
1	Song-102	Unbreakable Beat	Pop	2
2	Song-103	Sunset Boulevard	Pop Music	5
3	Song-104	Glowing Hearts	Pop Music	10
4	Song-105	Pop Rocks	Pop Music	52

In [10]:
```
audio.dtypes
```

Out[10]:
```
ID            object
Name          object
Genre         object
Popularity     int64
dtype: object
```

In [11]:
```
sessions.head()
```

Out[11]:

	Session_ID	Session_Log _In_Time
0	100520	2023-03-13 18:29:00
1	100522	2023-03-13 22:15:00
2	100525	2023-03-14 10:01:00
3	100527	2023-03-13 14:14:00
4	100538	2023-03-21 12:23:00

In [12]:
```
sessions.dtypes
```

Out[12]:
```
Session_ID                        int64
Session_Log _In_Time     datetime64[ns]
dtype: object
```

In [13]:
```
customers.head()
```

Out[13]:

	Customer_ID	Customer_Name	Email	Member_Since	Subscription_Plan
0	5001	Harmony Greene	Email: harmonious.vibes@email.com	3/13/2023	Basic (Ads)
1	5002	Aria Keys	Email: melodious.aria@email.edu	3/13/2023	NaN
2	5004	Lyric Bell	Email: rhythmical.lyric@email.com	3/13/2023	NaN
3	5267	Rock Bassett	Email: groovy.rock@email.com	3/20/2023	Basic (Ads)
4	5338	Rhythm Dixon	Email: beats.by.rhythm@email.edu	3/20/2023	NaN

In [14]:
```
customers.dtypes
```

Out[14]:
```
Customer_ID           int64
Customer_Name        object
Email                object
Member_Since         object
Subscription_Plan    object
Subscription_Rate    object
Discount?            object
Cancellation_Date    object
dtype: object
```

In [15]:
```
customers.Member_Since = pd.to_datetime(customers.Member_Since)
customers.Subscription_Rate = customers.Subscription_Rate.astype(str)
customers.Subscription_Rate = pd.to_numeric(customers.Subscription_Rate.str.re
place('$', '', regex=True))
customers.Cancellation_Date = pd.to_datetime(customers.Cancellation_Date)
```

In [16]:
```
customers.dtypes
```

Out[16]:
```
Customer_ID                   int64
Customer_Name                object
Email                        object
Member_Since         datetime64[ns]
Subscription_Plan            object
Subscription_Rate           float64
Discount?                    object
Cancellation_Date    datetime64[ns]
dtype: object
```

b. Resolve Data Issues

Check for missing data, inconsistent text and typos, duplicate data and outliers.

i. Missing Data

In [17]:
```
# Look for NaN values in the data
customers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30 entries, 0 to 29
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Customer_ID        30 non-null     int64
 1   Customer_Name      30 non-null     object
 2   Email              30 non-null     object
 3   Member_Since       30 non-null     datetime64[ns]
 4   Subscription_Plan  25 non-null     object
 5   Subscription_Rate  30 non-null     float64
 6   Discount?          7 non-null      object
 7   Cancellation_Date  13 non-null     datetime64[ns]
dtypes: datetime64[ns](2), float64(1), int64(1), object(4)
memory usage: 2.0+ KB
```

- Subscription_Plan, Discount?, Cancellation_Date those values has 'NaN values

In [18]:
```
# Look for NaN values in the data
listenning_history.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 505 entries, 0 to 504
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Customer_ID  505 non-null    int64
 1   Session _ID  505 non-null    int64
 2   Audio_Order  505 non-null    int64
 3   Audio_ID     505 non-null    int64
 4   Audio_Type   505 non-null    object
dtypes: int64(4), object(1)
memory usage: 19.9+ KB
```

- No NaN values in listenning_history data

In [19]:
```
# Look for NaN values in the data
audio.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17 entries, 0 to 16
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   ID          17 non-null     object
 1   Name        17 non-null     object
 2   Genre       17 non-null     object
 3   Popularity  17 non-null     int64
dtypes: int64(1), object(3)
memory usage: 672.0+ bytes
```

- No NaN Vslues

In [20]:
```
# Look for NaN values in the data
sessions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90 entries, 0 to 89
Data columns (total 2 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Session_ID            90 non-null     int64
 1   Session_Log _In_Time  90 non-null     datetime64[ns]
dtypes: datetime64[ns](1), int64(1)
memory usage: 1.5 KB
```

- No Nan values

In [21]:
```
# the customers dataframe has null values in the field Subscription_Plan, Disc
ount?, Cancellation_Date
customers.head()
```

Out[21]:

	Customer_ID	Customer_Name	Email	Member_Since	Subscription_Plan
0	5001	Harmony Greene	Email: harmonious.vibes@email.com	2023-03-13	Basic (Ads)
1	5002	Aria Keys	Email: melodious.aria@email.edu	2023-03-13	NaN
2	5004	Lyric Bell	Email: rhythmical.lyric@email.com	2023-03-13	NaN
3	5267	Rock Bassett	Email: groovy.rock@email.com	2023-03-20	Basic (Ads)
4	5338	Rhythm Dixon	Email: beats.by.rhythm@email.edu	2023-03-20	NaN

In [22]:
```
# looking into subscription plan - all nan subscription plans are $2.99
customers[customers.Subscription_Plan.isna()]
```

Out[22]:

	Customer_ID	Customer_Name	Email	Member_Since	Subscription_Plan	S
1	5002	Aria Keys	Email: melodious.aria@email.edu	2023-03-13	NaN	
2	5004	Lyric Bell	Email: rhythmical.lyric@email.com	2023-03-13	NaN	
4	5338	Rhythm Dixon	Email: beats.by.rhythm@email.edu	2023-03-20	NaN	
5	5404	Jazz Saxton	Email: jazzy.sax@email.com	2023-03-20	NaN	
11	5827	Rhythm Franklin	Email: rhythmic.franklin@email.edu	2023-03-28	NaN	

- The above cases are the NaN for Subscription_Plan and it seems like all these situations have a rate of 2.99

In [23]:
```
# chck the unique subscription Rates
customers[['Subscription_Rate', 'Subscription_Plan']].drop_duplicates()
```

Out[23]:

	Subscription_Rate	Subscription_Plan
0	2.99	Basic (Ads)
1	2.99	NaN
6	9.99	Premium (No Ads)
15	99.99	Premium (No Ads)
21	7.99	Premium (No Ads)

- This NaN should actually be basic ads

In [24]:
```
# fill missing subscription plan values with 'Basic(Ads)'
customers['Subscription_Plan'] = customers['Subscription_Plan'].fillna('Basic
(Ads)')
customers.head()
```

Out[24]:

	Customer_ID	Customer_Name	Email	Member_Since	Subscription_Plan
0	5001	Harmony Greene	Email: harmonious.vibes@email.com	2023-03-13	Basic (Ads)
1	5002	Aria Keys	Email: melodious.aria@email.edu	2023-03-13	Basic (Ads)
2	5004	Lyric Bell	Email: rhythmical.lyric@email.com	2023-03-13	Basic (Ads)
3	5267	Rock Bassett	Email: groovy.rock@email.com	2023-03-20	Basic (Ads)
4	5338	Rhythm Dixon	Email: beats.by.rhythm@email.edu	2023-03-20	Basic (Ads)

In [25]:
```
# look into Discount?
customers[['Customer_ID', 'Discount?']].tail()
```

Out[25]:

	Customer_ID	Discount?
25	7224	Yes
26	7401	Yes
27	7579	NaN
28	7581	Yes
29	7583	Yes

- NaN seeems to mean No

In [26]:
```
customers['Discount?'].value_counts()
```

Out[26]:
```
Yes    7
Name: Discount?, dtype: int64
```

In [27]:
```
# I'm changing data to numeric
import numpy as np

customers['Discount?'] = np.where(customers['Discount?']=='Yes', 1, 0)
customers.head()
```

Out[27]:

	Customer_ID	Customer_Name	Email	Member_Since	Subscription_Plan
0	5001	Harmony Greene	Email: harmonious.vibes@email.com	2023-03-13	Basic (Ads)
1	5002	Aria Keys	Email: melodious.aria@email.edu	2023-03-13	Basic (Ads)
2	5004	Lyric Bell	Email: rhythmical.lyric@email.com	2023-03-13	Basic (Ads)
3	5267	Rock Bassett	Email: groovy.rock@email.com	2023-03-20	Basic (Ads)
4	5338	Rhythm Dixon	Email: beats.by.rhythm@email.edu	2023-03-20	Basic (Ads)

In [28]:
```
# looking into cancellation NaN seems to mean not cancelled yet I'll leave it as is
```

ii. Inconsistent Text & Typos

In [29]:
```
# Look for inconsistent text & typos
customers.describe()
```

Out[29]:

	Customer_ID	Subscription_Rate	Discount?
count	30.000000	30.000000	30.000000
mean	6276.333333	8.556667	0.233333
std	814.255587	17.517840	0.430183
min	5001.000000	2.990000	0.000000
25%	5759.500000	2.990000	0.000000
50%	6196.000000	2.990000	0.000000
75%	6823.500000	7.990000	0.000000
max	7583.000000	99.990000	1.000000

*** customer ID min max that seems fine**

*** subscription rate 2.99 - 99 that seems really high**

In [30]:
```
# I'm going to look at it further more, I'm going to choose who pays more than 7.99 a month and the reson I choose 7.99 is because that was my 75% range
customers[customers['Subscription_Rate'] > 7.99]
```

Out[30]:

	Customer_ID	Customer_Name	Email	Member_Since	Subscription_Plan
6	5581	Reed Sharp	Email: sharp.tunes@email.com	2023-03-21	Premium (No Ads)
7	5759	Carol Kingbird	Email: songbird.carol@email.com	2023-03-22	Premium (No Ads)
8	5761	Sonata Nash	Email: musical.sonata@email.com	2023-03-28	Premium (No Ads)
12	6029	Chord Campbell	Email: campbell.chordify@email.com	2023-03-29	Premium (No Ads)
14	6163	Melody Parks	Email: park.of.melodies@email.com	2023-04-05	Premium (No Ads)
15	6229	Symphony Rhodes	Email: rhodes.symphony@email.com	2023-04-06	Premium (No Ads)

- I can see here these customers all pay 9.99 and this customer_ID 6229 is paying 99.99 which i belive is a typo because they all have the same plan somebody just added extra nine on accident

In [31]:
```
# To fix the above error (99.99 typo) Row 15 column 5
customers.iloc[15, 5] = 9.99
```

In [32]:
```
customers.describe()
```

Out[32]:

	Customer_ID	Subscription_Rate	Discount?
count	30.000000	30.000000	30.000000
mean	6276.333333	5.556667	0.233333
std	814.255587	3.058998	0.430183
min	5001.000000	2.990000	0.000000
25%	5759.500000	2.990000	0.000000
50%	6196.000000	2.990000	0.000000
75%	6823.500000	7.990000	0.000000
max	7583.000000	9.990000	1.000000

- Now I can see the typo has been fixed

In [33]:
```
# check the date range of the customers
customers['Member_Since'].max()
```

Out[33]:
```
Timestamp('2023-05-16 00:00:00')
```

- I have march through may data here

In [34]:
```
# Look at listenning history
listenning_history.describe()
```

Out[34]:

	Customer_ID	Session _ID	Audio_Order	Audio_ID
count	505.000000	505.000000	505.000000	505.000000
mean	6112.247525	105225.554455	4.138614	112.063366
std	832.861221	3625.879577	2.669008	24.670285
min	5001.000000	100520.000000	1.000000	101.000000
25%	5267.000000	101925.000000	2.000000	103.000000
50%	6029.000000	105116.000000	4.000000	105.000000
75%	6822.000000	109654.000000	6.000000	109.000000
max	7583.000000	111333.000000	15.000000	205.000000

- Everything looks pretty normal

In [35]:
```
listenning_history.head()
```

Out[35]:

	Customer_ID	Session _ID	Audio_Order	Audio_ID	Audio_Type
0	5001	100520	1	101	Song
1	5001	100520	2	102	Song
2	5001	100520	3	103	Song
3	5001	100520	4	104	Song
4	5001	100520	5	105	Song

In [36]:
```
# we have categorical field caled audio type
listenning_history['Audio_Type'].value_counts()
```

Out[36]:
```
Song        463
Podcast      42
Name: Audio_Type, dtype: int64
```

- there are songs and podcast

In [37]:
```
audio.head()
```

Out[37]:

	ID	Name	Genre	Popularity
0	Song-101	Dance All Night	Pop	1
1	Song-102	Unbreakable Beat	Pop	2
2	Song-103	Sunset Boulevard	Pop Music	5
3	Song-104	Glowing Hearts	Pop Music	10
4	Song-105	Pop Rocks	Pop Music	52

In [38]:
```
# look into genre
audio['Genre'].value_counts()
```

Out[38]:
```
Pop Music     3
Hip Hop       3
Comedy        3
Pop           2
Country       2
Jazz          2
True Crime    2
Name: Genre, dtype: int64
```

- I have Pop Music Genre and Pop genre that seems to be a duplicate

In [39]:
```
# pop and pop music should be mapped to the same value
audio.Genre = np.where(audio.Genre == 'Pop Music', 'Pop', audio.Genre)
```

In [40]:
```
audio['Genre'].value_counts()
```

Out[40]:
```
Pop           5
Hip Hop       3
Comedy        3
Country       2
Jazz          2
True Crime    2
Name: Genre, dtype: int64
```

In [41]:
```
sessions.head()
```

Out[41]:

	Session_ID	Session_Log _In_Time
0	100520	2023-03-13 18:29:00
1	100522	2023-03-13 22:15:00
2	100525	2023-03-14 10:01:00
3	100527	2023-03-13 14:14:00
4	100538	2023-03-21 12:23:00

In [42]:
```
# Look at loging time range
sessions['Session_Log _In_Time'].max()
```

Out[42]:
```
Timestamp('2023-05-31 06:03:00')
```

- It seems to be good

iii. Duplicate Rows

In [43]:
```
# Look for duplicate rows
customers[customers.duplicated()]
```

Out[43]:

	Customer_ID	Customer_Name	Email	Member_Since	Subscription_Plan	Subscription_Rate	Dis

In [44]:
```
sessions[sessions.duplicated()]
```

Out[44]:

	Session_ID	Session_Log _In_Time

In [45]:
```
listenning_history[listenning_history.duplicated()]
```

Out[45]:

	Customer_ID	Session _ID	Audio_Order	Audio_ID	Audio_Type

- I do not have duplicated rows in all the data

iv. Outliers

In [46]:
```
# Look for outliers - I'm checking min max values
customers.describe()
```

Out[46]:

	Customer_ID	Subscription_Rate	Discount?
count	30.000000	30.000000	30.000000
mean	6276.333333	5.556667	0.233333
std	814.255587	3.058998	0.430183
min	5001.000000	2.990000	0.000000
25%	5759.500000	2.990000	0.000000
50%	6196.000000	2.990000	0.000000
75%	6823.500000	7.990000	0.000000
max	7583.000000	9.990000	1.000000

In [47]:
```
sessions.describe()
```

Out[47]:

	Session_ID
count	90.000000
mean	105619.788889
std	3616.208569
min	100520.000000
25%	102149.000000
50%	105390.500000
75%	109658.250000
max	111333.000000

In [48]:
```
listenning_history.describe()
```

Out[48]:

	Customer_ID	Session _ID	Audio_Order	Audio_ID
count	505.000000	505.000000	505.000000	505.000000
mean	6112.247525	105225.554455	4.138614	112.063366
std	832.861221	3625.879577	2.669008	24.670285
min	5001.000000	100520.000000	1.000000	101.000000
25%	5267.000000	101925.000000	2.000000	103.000000
50%	6029.000000	105116.000000	4.000000	105.000000
75%	6822.000000	109654.000000	6.000000	109.000000
max	7583.000000	111333.000000	15.000000	205.000000

In [49]:
```
audio.describe()
```

Out[49]:

	Popularity
count	17.000000
mean	21.058824
std	23.381271
min	1.000000
25%	4.000000
50%	10.000000
75%	28.000000
max	80.000000

- Everthing is pretty good here

c. Create New Columns

Create two new columns that will be useful for EDA and modeling:

- Cancelled: whether a customer cancelled or not
- Email: Remove the "Email:" from the email addresses

In [50]:
```
customers.head()
```

Out[50]:

	Customer_ID	Customer_Name	Email	Member_Since	Subscription_Plan
0	5001	Harmony Greene	Email: harmonious.vibes@email.com	2023-03-13	Basic (Ads)
1	5002	Aria Keys	Email: melodious.aria@email.edu	2023-03-13	Basic (Ads)
2	5004	Lyric Bell	Email: rhythmical.lyric@email.com	2023-03-13	Basic (Ads)
3	5267	Rock Bassett	Email: groovy.rock@email.com	2023-03-20	Basic (Ads)
4	5338	Rhythm Dixon	Email: beats.by.rhythm@email.edu	2023-03-20	Basic (Ads)

- I have cancellation date but I don't have a column for whther a customer cancelled or not

In [51]:
```
# Create a 'Cancelled' column
customers['Cancelled'] = np.where(customers['Cancellation_Date'].notna(), 1,
0)
customers.head()
```

Out[51]:

	Customer_ID	Customer_Name	Email	Member_Since	Subscription_Plan
0	5001	Harmony Greene	Email: harmonious.vibes@email.com	2023-03-13	Basic (Ads)
1	5002	Aria Keys	Email: melodious.aria@email.edu	2023-03-13	Basic (Ads)
2	5004	Lyric Bell	Email: rhythmical.lyric@email.com	2023-03-13	Basic (Ads)
3	5267	Rock Bassett	Email: groovy.rock@email.com	2023-03-20	Basic (Ads)
4	5338	Rhythm Dixon	Email: beats.by.rhythm@email.edu	2023-03-20	Basic (Ads)

In [52]:
```
# Create an updated 'Email' column without the Email: portion
customers['Email'] = customers.Email.str[6:] # only read charactors starting f
rom position 6 and onward
customers.head()
```

Out[52]:

	Customer_ID	Customer_Name	Email	Member_Since	Subscription_Plan
0	5001	Harmony Greene	harmonious.vibes@email.com	2023-03-13	Basic (Ads)
1	5002	Aria Keys	melodious.aria@email.edu	2023-03-13	Basic (Ads)
2	5004	Lyric Bell	rhythmical.lyric@email.com	2023-03-13	Basic (Ads)
3	5267	Rock Bassett	groovy.rock@email.com	2023-03-20	Basic (Ads)
4	5338	Rhythm Dixon	beats.by.rhythm@email.edu	2023-03-20	Basic (Ads)

4. EDA

Try to better understand the customers who cancelled:

- How long were they members before they cancelled?
- What percentage of customers who cancelled had a discount vs customers who didn't cancel?

In [53]:
```
customers.head()
```

Out[53]:

	Customer_ID	Customer_Name	Email	Member_Since	Subscription_Plan
0	5001	Harmony Greene	harmonious.vibes@email.com	2023-03-13	Basic (Ads)
1	5002	Aria Keys	melodious.aria@email.edu	2023-03-13	Basic (Ads)
2	5004	Lyric Bell	rhythmical.lyric@email.com	2023-03-13	Basic (Ads)
3	5267	Rock Bassett	groovy.rock@email.com	2023-03-20	Basic (Ads)
4	5338	Rhythm Dixon	beats.by.rhythm@email.edu	2023-03-20	Basic (Ads)

In [54]:
```
#view the customers who cancelled
customers[customers['Cancellation_Date'].notna()].head()
```

Out[54]:

	Customer_ID	Customer_Name	Email	Member_Since	Subscription_Plan
2	5004	Lyric Bell	rhythmical.lyric@email.com	2023-03-13	Basic (Ads)
5	5404	Jazz Saxton	jazzy.sax@email.com	2023-03-20	Basic (Ads)
7	5759	Carol Kingbird	songbird.carol@email.com	2023-03-22	Premium (No Ads)
12	6029	Chord Campbell	campbell.chordify@email.com	2023-03-29	Premium (No Ads)
13	6092	Benny Beat	rhythmic.benny@email.com	2023-04-01	Basic (Ads)

In [55]:
```
# How long were customers members before they cancelled?
(customers['Cancellation_Date'] - customers['Member_Since']).mean()
```

Out[55]:
```
Timedelta('46 days 07:23:04.615384615')
```

- about 1.5 months but that might just because we have 3 months of data

In [56]:
```
# Cancellation rate for those who had a discount
discount_yes = customers[customers['Discount?'] ==1]
discount_yes
```

Out[56]:

	Customer_ID	Customer_Name	Email	Member_Since	Subscription_Plan
21	6822	Kiki Keys	kiki.keys.piano@email.com	2023-05-01	Premium (No Ads)
22	6824	Greta Groove	groovy.greta@email.com	2023-05-01	Premium (No Ads)
23	7087	Harmony Heart	heartfelt.harmony@email.com	2023-05-01	Premium (No Ads)
25	7224	Melody Fitzgerald	fitzgerald.melody@email.com	2023-05-08	Premium (No Ads)
26	7401	Reed Murphy	murphy.reed.music@email.com	2023-05-08	Premium (No Ads)
28	7581	Lyric Keys	keysoflyric@email.com	2023-05-16	Premium (No Ads)
29	7583	Melody Singer	melodic.singer@email.com	2023-05-16	Premium (No Ads)

In [57]:
```
# Cancellation rate for those who had a discount
discount_yes.Cancelled.sum() / discount_yes.Cancelled.count()
```

Out[57]: 0.8571428571428571

In [58]:
```
# Cancellation rate for those who did not have a discount
discount_no = customers[customers['Discount?'] == 0]
discount_no.head()
```

Out[58]:

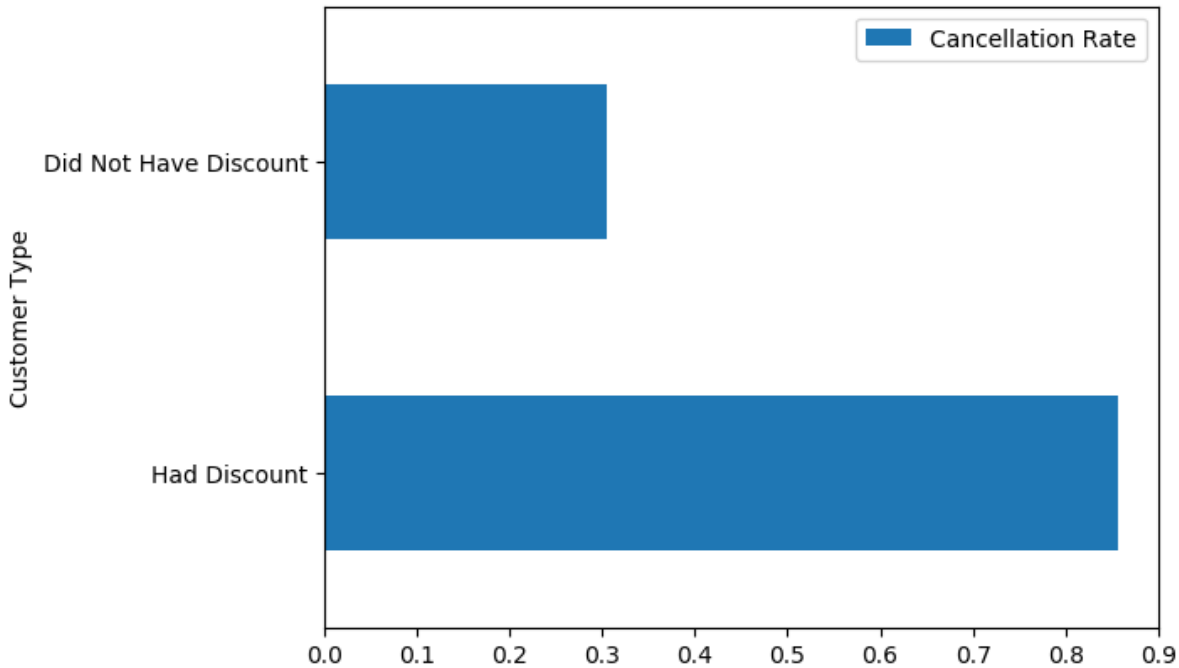
	Customer_ID	Customer_Name	Email	Member_Since	Subscription_Plan
0	5001	Harmony Greene	harmonious.vibes@email.com	2023-03-13	Basic (Ads)
1	5002	Aria Keys	melodious.aria@email.edu	2023-03-13	Basic (Ads)
2	5004	Lyric Bell	rhythmical.lyric@email.com	2023-03-13	Basic (Ads)
3	5267	Rock Bassett	groovy.rock@email.com	2023-03-20	Basic (Ads)
4	5338	Rhythm Dixon	beats.by.rhythm@email.edu	2023-03-20	Basic (Ads)

In [59]:
```
discount_no.Cancelled.sum() / discount_no.Cancelled.count()
```

Out[59]: 0.30434782608695654

- What those results ar telling me about - people who got a discount are much more likely to cancele than people who didn't get the discount

In [60]:
```
# Visualize the cancellation rate for those with a discount vs those without a discount
pd.DataFrame([['Had Discount', 0.8571428571428571],
                ['Did Not Have Discount', 0.30434782608695654]],
                columns = ['Customer Type', 'Cancellation Rate']).plot.barh(x ='Customer Type', y='Cancellation Rate' ,);
```



🙈 The people who have a discount have a much higher cancellation rate more than two times as much

Better understand the customers' listening histories:

- Join together the listening history and audio tables
- How many listening sessions did each customer have in the past 3 months?
- What were the most popular genres that customers listened to?

In [61]:
```
listenning_history.head()
```

Out[61]:

	Customer_ID	Session _ID	Audio_Order	Audio_ID	Audio_Type
0	5001	100520	1	101	Song
1	5001	100520	2	102	Song
2	5001	100520	3	103	Song
3	5001	100520	4	104	Song
4	5001	100520	5	105	Song

In [62]:
```
audio.head()
```

Out[62]:

	ID	Name	Genre	Popularity
0	Song-101	Dance All Night	Pop	1
1	Song-102	Unbreakable Beat	Pop	2
2	Song-103	Sunset Boulevard	Pop	5
3	Song-104	Glowing Hearts	Pop	10
4	Song-105	Pop Rocks	Pop	52

In [63]:
```
sessions.head()
```

Out[63]:

	Session_ID	Session_Log _In_Time
0	100520	2023-03-13 18:29:00
1	100522	2023-03-13 22:15:00
2	100525	2023-03-14 10:01:00
3	100527	2023-03-13 14:14:00
4	100538	2023-03-21 12:23:00

In [64]:
```
# Split the ID in the audio data so the column can be joined with other tables
audio_clean = pd.DataFrame(audio.ID.str.split('-').to_list()).rename(columns=
{0:'Type', 1:'Audio_ID'})
audio_clean.head()
```

Out[64]:

	Type	Audio_ID
0	Song	101
1	Song	102
2	Song	103
3	Song	104
4	Song	105

In [65]:
```
audio.dtypes
```

Out[65]:
```
ID            object
Name          object
Genre         object
Popularity     int64
dtype: object
```

In [66]:
```
audio_clean.dtypes
```

Out[66]:
```
Type        object
Audio_ID    object
dtype: object
```

In [67]:
```
listenning_history.dtypes
```

Out[67]:
```
Customer_ID     int64
Session _ID     int64
Audio_Order     int64
Audio_ID        int64
Audio_Type     object
dtype: object
```

In [68]:
```
# Adding new field to the original aaudio table
audio_all = pd.concat([audio_clean, audio], axis=1)
audio_all.head()
```

Out[68]:

	Type	Audio_ID	ID	Name	Genre	Popularity
0	Song	101	Song-101	Dance All Night	Pop	1
1	Song	102	Song-102	Unbreakable Beat	Pop	2
2	Song	103	Song-103	Sunset Boulevard	Pop	5
3	Song	104	Song-104	Glowing Hearts	Pop	10
4	Song	105	Song-105	Pop Rocks	Pop	52

In [74]:
```
audio_all.dtypes
```

Out[74]:
```
Type          object
Audio_ID       int32
ID            object
Name          object
Genre         object
Popularity     int64
dtype: object
```

In [73]:
```
audio_all['Audio_ID'] = audio_all['Audio_ID'].astype('int')
```

In [75]:
```
# merging Audio_all and  and listenning history
df = listenning_history.merge(audio_all, how = 'left', on = 'Audio_ID')
df
```

Out[75]:

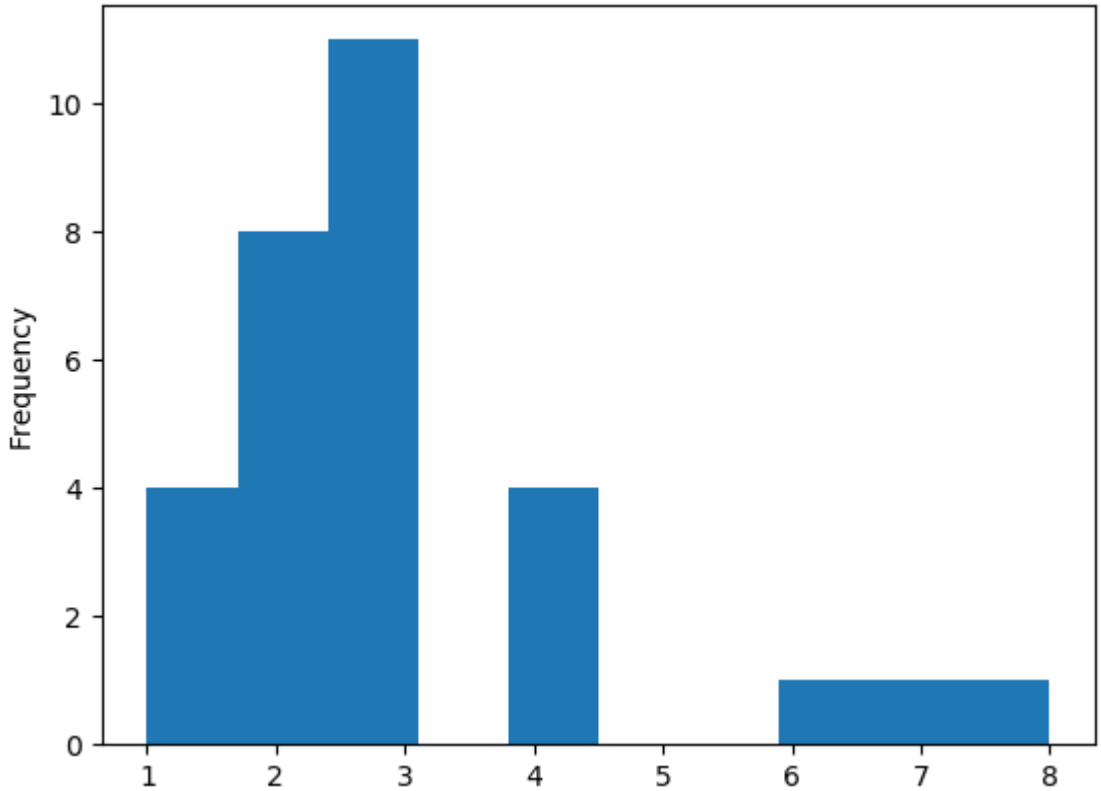
	Customer_ID	Session_ID	Audio_Order	Audio_ID	Audio_Type	Type	ID	Name
0	5001	100520	1	101	Song	Song	Song-101	Dance All Night
1	5001	100520	2	102	Song	Song	Song-102	Unbreakable Beat
2	5001	100520	3	103	Song	Song	Song-103	Sunset Boulevard
3	5001	100520	4	104	Song	Song	Song-104	Glowing Hearts
4	5001	100520	5	105	Song	Song	Song-105	Pop Rocks
...	...	...	...	...	...	...	...	...
500	7579	111282	4	111	Song	Song	Song-111	Moonlit Serenade
501	6588	111286	1	201	Podcast	Podcast	Podcast-201	Jokes on Jokes
502	5763	111333	1	110	Song	Song	Song-110	Boss Moves
503	5763	111333	2	108	Song	Song	Song-108	Chase the Dream
504	5763	111333	3	110	Song	Song	Song-110	Boss Moves

505 rows × 10 columns

In [78]:
```
df.groupby('Customer_ID')['Session _ID'].nunique()
```

Out[78]:
```
Customer_ID
5001    8
5002    4
5004    1
5267    7
5338    4
5404    1
5581    3
5759    2
5761    3
5763    6
5826    3
5827    1
6029    2
6092    3
6163    3
6229    2
6406    3
6584    2
6586    2
6588    3
6821    2
6822    3
6824    4
7087    3
7158    3
7224    4
7401    3
7579    2
7581    2
7583    1
Name: Session _ID, dtype: int64
```

In [81]:
```
# The number of listening sessions that each customer had in the past 3 months
# to find the unique session ids I use nunique
df.groupby('Customer_ID')['Session _ID'].nunique().plot.hist();
```



- Most customers have about 2 or 3 listening sessions a.nd also we have a few exreme listeners out there who likes to listen to alot of songs

In [82]:
```
#The most popular genres that customers listened to
df.Genre.value_counts()
```

Out[82]:
```
Pop           267
Hip Hop        88
Country        68
Jazz           48
Comedy         19
True Crime     15
Name: Genre, dtype: int64
```

- For all my customers, they listen to a lot of Pop songs and not so many true crime podcasts

5. Prep for Modeling

Create a DataFrame that is ready for modeling with each row representing a customer and the following numeric, non-null columns:

- Customer ID
- Whether a customer cancelled or not
- Whether a customer received a discount or not
- The number of listening sessions
- Percent of listening history consisting of Pop
- Percent of listening history consisting of Podcasts

In [84]:
```
# Create a dataframe ready for modeling
model_df = customers[['Customer_ID', 'Cancelled', 'Discount?']]
model_df.head()
```

Out[84]:

	Customer_ID	Cancelled	Discount?
0	5001	0	0
1	5002	0	0
2	5004	1	0
3	5267	0	0
4	5338	0	0

In [85]:
```
df.head()
```

Out[85]:

	Customer_ID	Session_ID	Audio_Order	Audio_ID	Audio_Type	Type	ID	Name	Genre
0	5001	100520	1	101	Song	Song	Song-101	Dance All Night	Pop
1	5001	100520	2	102	Song	Song	Song-102	Unbreakable Beat	Pop
2	5001	100520	3	103	Song	Song	Song-103	Sunset Boulevard	Pop
3	5001	100520	4	104	Song	Song	Song-104	Glowing Hearts	Pop
4	5001	100520	5	105	Song	Song	Song-105	Pop Rocks	Pop

In [87]:
```
df.groupby('Customer_ID')['Session _ID'].nunique()
```

Out[87]:
```
Customer_ID
5001    8
5002    4
5004    1
5267    7
5338    4
5404    1
5581    3
5759    2
5761    3
5763    6
5826    3
5827    1
6029    2
6092    3
6163    3
6229    2
6406    3
6584    2
6586    2
6588    3
6821    2
6822    3
6824    4
7087    3
7158    3
7224    4
7401    3
7579    2
7581    2
7583    1
Name: Session _ID, dtype: int64
```

In [89]:
```
df.groupby('Customer_ID')['Session _ID'].nunique().rename('Number_of_Session
s')
```

Out[89]:
```
Customer_ID
5001    8
5002    4
5004    1
5267    7
5338    4
5404    1
5581    3
5759    2
5761    3
5763    6
5826    3
5827    1
6029    2
6092    3
6163    3
6229    2
6406    3
6584    2
6586    2
6588    3
6821    2
6822    3
6824    4
7087    3
7158    3
7224    4
7401    3
7579    2
7581    2
7583    1
Name: Number_of_Sessions, dtype: int64
```

In [91]:
```
# Calculate the number of listening sessions for each customer
number_of_listening_sessions = df.groupby('Customer_ID')['Session _ID'].nuniqu
e().rename('Number_of_Sessions').to_frame().reset_index()

number_of_listening_sessions.head()
```

Out[91]:

	Customer_ID	Number_of_Sessions
0	5001	8
1	5002	4
2	5004	1
3	5267	7
4	5338	4

In [92]:
```
# Add the above frame to the model dataframe
model_df = model_df.merge(number_of_listening_sessions, how='left', on='Custom
er_ID')
model_df.head()
```

Out[92]:

	Customer_ID	Cancelled	Discount?	Number_of_Sessions
0	5001	0	0	8
1	5002	0	0	4
2	5004	1	0	1
3	5267	0	0	7
4	5338	0	0	4

In [95]:
```
df.Genre
```

Out[95]:
```
0           Pop
1           Pop
2           Pop
3           Pop
4           Pop
        ...
500       Jazz
501     Comedy
502    Hip Hop
503    Hip Hop
504    Hip Hop
Name: Genre, Length: 505, dtype: object
```

In [94]:
```
# Calculate dummy variables for each genre
pd.get_dummies(df.Genre)
```

Out[94]:

	Comedy	Country	Hip Hop	Jazz	Pop	True Crime
0	0	0	0	0	1	0
1	0	0	0	0	1	0
2	0	0	0	0	1	0
3	0	0	0	0	1	0
4	0	0	0	0	1	0
...	...	...	...	...	...	...
500	0	0	0	1	0	0
501	1	0	0	0	0	0
502	0	0	1	0	0	0
503	0	0	1	0	0	0
504	0	0	1	0	0	0

505 rows × 6 columns

In [96]:
```
#combine it with the customer_ID
pd.concat([df['Customer_ID'], pd.get_dummies(df.Genre)], axis=1)
```

Out[96]:

	Customer_ID	Comedy	Country	Hip Hop	Jazz	Pop	True Crime
0	5001	0	0	0	0	1	0
1	5001	0	0	0	0	1	0
2	5001	0	0	0	0	1	0
3	5001	0	0	0	0	1	0
4	5001	0	0	0	0	1	0
...	...	...	...	...	...	...	...
500	7579	0	0	0	1	0	0
501	6588	1	0	0	0	0	0
502	5763	0	0	1	0	0	0
503	5763	0	0	1	0	0	0
504	5763	0	0	1	0	0	0

505 rows × 7 columns

In [98]:
```
# Group it by customer
genres = pd.concat([df['Customer_ID'], pd.get_dummies(df.Genre)], axis=1).grou
pby('Customer_ID').sum().reset_index()
genres.head()
```

Out[98]:

	Customer_ID	Comedy	Country	Hip Hop	Jazz	Pop	True Crime
0	5001	0	0	26	0	34	0
1	5002	0	22	0	0	0	0
2	5004	0	0	0	0	9	0
3	5267	0	0	22	0	23	0
4	5338	0	18	0	0	0	0

In [99]:
```
listenning_history.head()
```

Out[99]:

	Customer_ID	Session _ID	Audio_Order	Audio_ID	Audio_Type
0	5001	100520	1	101	Song
1	5001	100520	2	102	Song
2	5001	100520	3	103	Song
3	5001	100520	4	104	Song
4	5001	100520	5	105	Song

In [102]:
```
# Add a column for total songs/ podcast listened to
total_audio = listenning_history.groupby('Customer_ID')['Audio_ID'].count().re
name('Total_Audio').to_frame().reset_index()
total_audio.head()
```

Out[102]:

	Customer_ID	Total_Audio
0	5001	60
1	5002	22
2	5004	9
3	5267	45
4	5338	18

In [105]:
```
# create a master audio table to calculate percentages
df_audio = genres.merge(total_audio, how='left', on ='Customer_ID')
df_audio.head()
```

Out[105]:

	Customer_ID	Comedy	Country	Hip Hop	Jazz	Pop	True Crime	Total_Audio
0	5001	0	0	26	0	34	0	60
1	5002	0	22	0	0	0	0	22
2	5004	0	0	0	0	9	0	9
3	5267	0	0	22	0	23	0	45
4	5338	0	18	0	0	0	0	18

In [106]:
```
# Percent pop
model_df['Percent_Pop'] =df_audio.Pop / df_audio['Total_Audio']*100
model_df.head()
```

Out[106]:

	Customer_ID	Cancelled	Discount?	Number_of_Sessions	Percent_Pop
0	5001	0	0	8	56.666667
1	5002	0	0	4	0.000000
2	5004	1	0	1	100.000000
3	5267	0	0	7	51.111111
4	5338	0	0	4	0.000000

In [110]:
```
# Percent podcasts
model_df['Percet_Podcast'] = ((df_audio['Comedy'] + df_audio['True Crime'])
/ df_audio['Total_Audio'])*100
model_df.tail()
```

Out[110]:

	Customer_ID	Cancelled	Discount?	Number_of_Sessions	Percent_Pop	Percet_Podcast
25	7224	1	1	4	100.000000	0.000000
26	7401	1	1	3	45.454545	27.272727
27	7579	0	0	2	0.000000	0.000000
28	7581	1	1	2	92.857143	7.142857
29	7583	1	1	1	0.000000	100.000000

In [111]:
```
model_df
```

Out[111]:

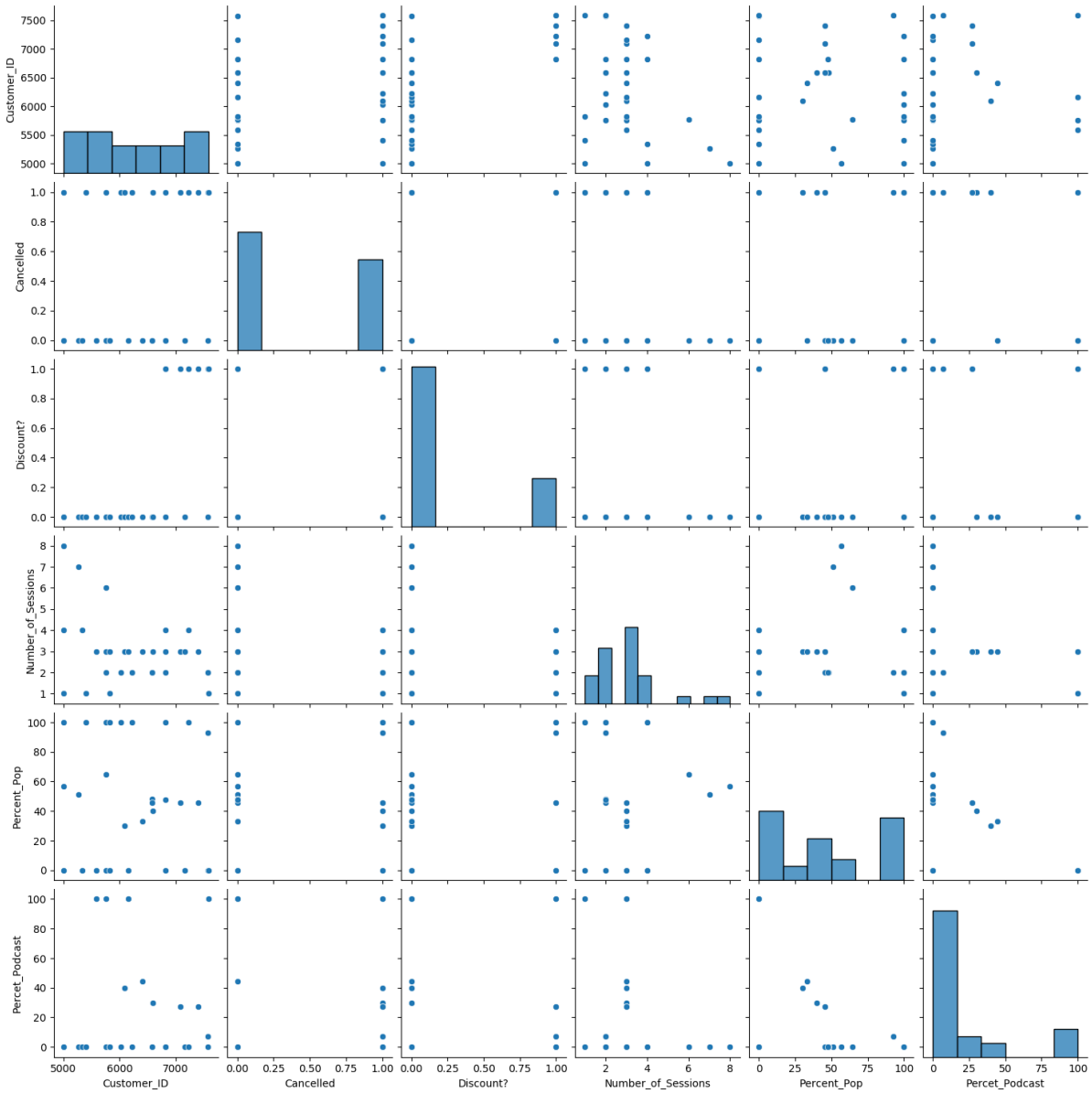
	Customer_ID	Cancelled	Discount?	Number_of_Sessions	Percent_Pop	Percet_Podcast
0	5001	0	0	8	56.666667	0.000000
1	5002	0	0	4	0.000000	0.000000
2	5004	1	0	1	100.000000	0.000000
3	5267	0	0	7	51.111111	0.000000
4	5338	0	0	4	0.000000	0.000000
5	5404	1	0	1	100.000000	0.000000
6	5581	0	0	3	0.000000	100.000000
7	5759	1	0	2	100.000000	0.000000
8	5761	0	0	3	0.000000	100.000000
9	5763	0	0	6	64.516129	0.000000
10	5826	0	0	3	0.000000	0.000000
11	5827	0	0	1	100.000000	0.000000
12	6029	1	0	2	100.000000	0.000000
13	6092	1	0	3	30.000000	40.000000
14	6163	0	0	3	0.000000	100.000000
15	6229	1	0	2	100.000000	0.000000
16	6406	0	0	3	33.333333	44.444444
17	6584	0	0	2	48.148148	0.000000
18	6586	0	0	2	45.454545	0.000000
19	6588	1	0	3	40.000000	30.000000
20	6821	0	0	2	47.619048	0.000000
21	6822	0	1	3	0.000000	0.000000
22	6824	1	1	4	100.000000	0.000000
23	7087	1	1	3	45.454545	27.272727
24	7158	0	0	3	0.000000	0.000000
25	7224	1	1	4	100.000000	0.000000
26	7401	1	1	3	45.454545	27.272727
27	7579	0	0	2	0.000000	0.000000
28	7581	1	1	2	92.857143	7.142857
29	7583	1	1	1	0.000000	100.000000

Visualize the relationships in the modeling DataFrame using a pair plot:

- What are some of your observations?
- What variables might do a good job predicting customer cancellation?

In [112]:
```
import seaborn as sns
```

In [113]:
```
sns.pairplot(model_df);
```



- I do not have much that much data it's kind of hard to see any relationship

In [114]:
```
# I'm going to look at the correlations
model_df.corr()
```

Out[114]:

	Customer_ID	Cancelled	Discount?	Number_of_Sessions	Percent_Pop	Per
Customer_ID	1.000000	0.269942	0.648514	-0.337083	-0.076129	
Cancelled	0.269942	1.000000	0.471825	-0.333739	0.585630	
Discount?	0.648514	0.471825	1.000000	-0.048877	0.112675	
Number_of_Sessions	-0.337083	-0.333739	-0.048877	1.000000	-0.131156	
Percent_Pop	-0.076129	0.585630	0.112675	-0.131156	1.000000	
Percet_Podcast	0.083083	-0.035414	0.062938	-0.125459	-0.487193	

Observations

- A Discount is correlated with a cancellation
- The more listening sessions, the fewr cancellations
- the more pop music, the more cancellations
- Podcast listening history seems unrelated to cancellations