BUSINESS ANALYTICS with PYTHON

Problem: Make strategic decisions regarding data exploration. The guidance provided will direct you to use specific variable types but will not always provide which variables to use. You and your team should work to determine how to glean meaningful insight from the data.

Data Background: The company that acquires, packages, and delivers bulk beverage products. Your company has been collecting various data points and is interesting in gleaning some insights from this data. Unfortunately, there is no one currently at the company that understands how to make sense of the data they have been collecting. This is where your team comes in.

Data Information:

Orders are either New orders or Repeat orders. The company stopped inputting 'Repeat' after some time, but all 'New' orders are labeled.

The company's shipping cost occurs due to free shipping for bulk orders, or occasional free shipping promotions. If the shipping cost is not captured in the data, there was no shipping cost to the company.

Total Cost to the company are Shipping Costs and Avg COA. Purchase total is the gross gain to the company.

Satisfaction is on a scale from 1-5 with 5 being the highest level of satisfaction. The rating provided is the average rating for all orders during the corresponding quarter. For this project, Satisifaction can be treated as a continuous variable.

## Read in and merge data sets appropriately. Take the steps to clean your data. Justify any decisions you made in filling missing values.

#Data Access and Cleaning code:

# import the necessary packages

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

​

# import the data files

data1 = pd.read\_excel('CostbySource.xlsx')

data2 = pd.read\_excel('CustomerDetails.xlsx')

data3 = pd.read\_excel('OrderDetailsbyQuarter.xlsx')

​

# Checking the columns in the data so that we know which variable to merge the data on

data1.columns

data2.columns

data3.columns

​

# Checking for duplicates values

data1.duplicated(keep = 'last').sum()

data2.duplicated(keep = 'last').sum()

data3.duplicated(keep = 'last').sum()

# No duplicates found

​

# Checking for NA values

data1.isna().sum()

data2.isna().sum() # Here need to replace the 'na' values with some string in Referral ID variable

data3.isna().sum() # Here the type of order 'na' values must be replaced with 'Repeat' as the company stopped labelling them

​

# Cleaning data2

# Filling the Referral ID variable 'na' values with None so that it is easy for further analysis

data2['Referral ID'] = data2['Referral ID'].fillna('None')

​

# Cleaning data3

# labelling the 'na' value in Type of Order with 'Repeat'

data3['Type of Order'] = data3['Type of Order'].fillna('Repeat')

# The date variable has 'Na' values. So seeing after the data the Na values needed to be ffilled so that the date gets copied

data3['Date'] = data3['Date'].fillna(method = 'ffill')

# Filling the Na values in the Shipping Costs column with '0' as mentioned in Data information

data3['Shipping Costs'] = data3['Shipping Costs'].fillna(0)

​

# Merging the data

data\_23 = pd.merge(data2, data3, on = 'AccountID')

data\_23.head()

data\_123 = pd.merge(data\_23, data1, on=['Original Source','Type of Order'])

​

# Checking for 'na' values after merging

data\_123.isna().sum()

​

# Printing the top 5 records of the final data frame

data\_123.head()

AccountID Type of Customer Original Source Initial Year Referral ID Date Type of Order Purchase Total Number of Orders Shipping Costs Satisfaction Rating Avg COA

0 Cust1 Public ColdCall 2014 None 2017-03-31 Repeat 469 2 28.715156 4.9 15

1 Cust1 Public ColdCall 2014 None 2017-06-30 Repeat 321 1 13.998584 4.8 15

2 Cust1 Public ColdCall 2014 None 2017-09-30 Repeat 272 2 30.994040 4.9 15

3 Cust1 Public ColdCall 2014 None 2017-12-31 Repeat 473 3 27.140828 5.0 15

4 Cust1 Public ColdCall 2014 None 2018-03-31 Repeat 339 2 19.271104 4.7 15

Justifications:

We had missing values in the variable 'Referral ID' of the dataframe 'data2' -> So we filled it with the value 'None', which means no one referred them.

We had missing values in data3 as follows

Type of Order -> As 'Repeat' were not lablelled later on, the missing values in this column must be 'Repeat' as mentioned in the question. Hence we filled the missing values with 'Repeat'.

Date -> As 'Date' was grouped in Excel as per quarter, we ffilled the Date to the next missing spaces. So that all the orders in the same quarter have the same dates.

Shipping Costs -> As mentioned we filled the missing values in this column with zeroes.

#Plots using categorical variables on the x axis and continuous variables on the y-axis. AVOID datetime variables. Interpret each plot.

#Cat Plot 1 code:

​

sns.catplot(x = 'Type of Order', y = 'Purchase Total', kind = 'bar',

data = data\_123).set(title = 'Average Purchase Total of Orders by its Type')

<seaborn.axisgrid.FacetGrid at 0x1c5754a1ee0>

Cat Plot 1 interpretation:

Based on the above plot we can say that the Average Purchase Total is relatively similar for both the Repeat and New types of orders and that is approximately 300.

#Cat Plot 2 code:

​

sns.catplot(x = 'Type of Customer',y = 'Purchase Total', kind = 'bar',

data = data\_123 ).set(title = 'Average Purchase Total by the Type of customer')

<seaborn.axisgrid.FacetGrid at 0x1c57aceb220>

Cat Plot 2 interpretation:

From the above plot we can understand that Direct type of customers have relatively lower purchase totals on average. Business Customers have the highest average Purchase total which is around $400.

Statistical analysis to confirm your interpretation of one of your categorical plots above. Be sure to investigate the descriptive statistics to explore any violations of statistical assumptions. Proceed with the analysis regardless of whether assumptions have been violated, BUT make sure to note concerns if they exist.

#Explore Assumptions code:

​

# As from the Categorical Plot between Type of order and Purchase Totals, we can see that the Average Purchase Total for

# Repeat is almost the same as that of New

data\_123.columns

from scipy.stats import shapiro

​

sns.displot(data\_123['Purchase Total'], kde = True)

shapiro(data\_123['Purchase Total'])

​

# Since the Shapiro test has a 'p value' that less than 0.05, we can conclude that the data is not normally distributed.

# That being said we still proceed with the analysis, even though the data is not very large and hence it is not normally distributed.

ShapiroResult(statistic=0.9507474899291992, pvalue=5.52254153518561e-09)

#Statistical Analysis code:

#To analyze the above assumption we do the statistical analysis

# Importing the necessary libraries

from scipy.stats import ttest\_ind

data\_New = data\_123[data\_123['Type of Order'] == 'New']

data\_Repeat = data\_123[data\_123['Type of Order'] == 'Repeat']

t,p = ttest\_ind(data\_New['Purchase Total'], data\_Repeat['Purchase Total'])

print(t,p)

0.136304356734281 0.891665331366871

Interpretation:

From the p value which is greater than 0.05, we can say that the averages of Purchase Totals(Gross Gain) of both types are statistically same. It's important that the company looks unto why the 'Repeat' order values are not increasing. (OR) If that is the market norm, tap into more 'New' orders.

#Plots using continuous variables on both the x and y axis. AVOID datetime variables. Interpret each plot.

#Cont Plot 1 Code:

# Plot between Purchase total and Shipping Costs

​

# To plot the best fit line that fits our scatter plot

a, b = np.polyfit(data\_123['Purchase Total'], data\_123['Shipping Costs'], 1)

plt.scatter(data\_123['Purchase Total'], data\_123['Shipping Costs'])

plt.plot(data\_123['Purchase Total'], a\*data\_123['Purchase Total']+b)

[<matplotlib.lines.Line2D at 0x1c57b9a5970>]

Cont Plot 1 Interpretation:

Based on the above plot we can see that the gradient(slope) of the trend line is increasing. That means Shipping cost is increasing as the Purchase total increases but the relationship has low coefficient which is good as we are not over expensing.

We need to check the two data points on the top right(high). They are outliers as they are not normal and increasing the expenses.

# Cont Plot 2 Code:

​

# We are assuming that the Avg COA is continuos

sns.lineplot(x = 'Avg COA', y = 'Satisfaction Rating', data = data\_123, hue = 'Type of Order')

<AxesSubplot:xlabel='Avg COA', ylabel='Satisfaction Rating'>

Cont Plot 2 Interpretation:

As we increase the average COA per the source, we can see a positive trend in the Satisfaction Rating for Repeat orders specifically increasing from Social media. (But) For New orders it drastically decreases transitioning from Referral to Cold call.

#Statistical analysis to confirm your interpretation of one of your continuous plots above. Be sure to investigate the descriptive statistics to explore any violations of statistical assumptions. Proceed with the analysis regardless of whether assumptions have been violated, BUT make sure to note concerns if they exist.

#Explore Assumptions code:

from scipy.stats import shapiro

​

sns.displot(data\_123['Avg COA'], kde = True)

shapiro(data\_123['Avg COA'])

ShapiroResult(statistic=0.7979050874710083, pvalue=7.763783458808031e-20)

sns.displot(data\_123['Satisfaction Rating'], kde = True)

shapiro(data\_123['Satisfaction Rating'])

ShapiroResult(statistic=0.9234814047813416, pvalue=7.283792492762675e-12)

Since the Shapiro test has a 'p value' less than 0.05 for both the plots, we can conclude that the data is not normally distributed. But we will go ahead since the data is not very large.

#Statistical Analysis code:

from scipy.stats import linregress

​

# Running the linear regression for the above continuis plots

res = linregress(data\_123['Avg COA'], data\_123['Satisfaction Rating'])

print('Satisfaction rating, Avg COA, r=', res.rvalue, 'p =', res.pvalue)

print('Slope', res.slope)

Satisfaction rating, Avg COA, r= 0.27949717859813294 p = 2.90267554438116e-07

Slope 0.03901523444626892

Interpretation:

Although r is just 0.28 but p-value is less than 0.05 which shows that there is a linear relationship between Avg COA and Satisfaction Rating. Every one point increase in Avg COA will result in approximately 0.039 increase in Satisfaction Rating.

#Using date as a date-time variable, plot the pattern of a continuous variable over time, grouped by a categorical vairable. Create two plots, one with actual data points, one with a 10 quarter rolling average. Describe and interpret any patterns you see.

NOTE: it may not be appropriate to use the mean as the aggregation function to see business growth.

#Time plot:

df\_time = data\_123.pivot\_table('Number of Orders', index= ['Date','Original Source'], aggfunc = sum)

sns.lineplot(x = 'Date', y = 'Number of Orders', data = df\_time, hue = 'Original Source')

<AxesSubplot:xlabel='Date', ylabel='Number of Orders'>

#Rolling average time plot code:

df\_un = df\_time.unstack()

df\_un.rolling(10).mean().plot()

<AxesSubplot:xlabel='Date'>

Interpretation:

1. As observed from the first plot, the number of orders from social media was in the leading till 2020 and then we can see the Referral orders have overtaken it. There was a dip in the number of orders placed with Social media and cold calls in the beginning of 2020. Contrastingly the number of orders from Referrals has increased.

2. From the rolling average plot, we can observe that Referral orders were almost in the bottom of stack but has seen a tremendous increase over the last ten quarters. Social Media has stayed on the top overall in the last 10 quarters.

# Using Total Cost to the company and gross gains, calculate the ROI for each row of the data. Then plot the average ROI based on Original Source. Describe the patterns that you see.

#ROI Calculations:

data\_123['Total Cost'] = data\_123['Shipping Costs'] + data\_123['Avg COA']

data\_123['ROI'] = (data\_123['Purchase Total'] - data\_123['Total Cost'])/data\_123['Total Cost']

data\_123.head()

AccountID Type of Customer Original Source Initial Year Referral ID Date Type of Order Purchase Total Number of Orders Shipping Costs Satisfaction Rating Avg COA Total Cost ROI

0 Cust1 Public ColdCall 2014 None 2017-03-31 Repeat 469 2 28.715156 4.9 15 43.715156 9.728545

1 Cust1 Public ColdCall 2014 None 2017-06-30 Repeat 321 1 13.998584 4.8 15 28.998584 10.069506

2 Cust1 Public ColdCall 2014 None 2017-09-30 Repeat 272 2 30.994040 4.9 15 45.994040 4.913810

3 Cust1 Public ColdCall 2014 None 2017-12-31 Repeat 473 3 27.140828 5.0 15 42.140828 10.224269

4 Cust1 Public ColdCall 2014 None 2018-03-31 Repeat 339 2 19.271104 4.7 15 34.271104 8.891715

#Plotting

sns.catplot(x = 'Original Source', y = 'ROI', kind = 'bar', data = data\_123)

<seaborn.axisgrid.FacetGrid at 0x1c57cacefd0>

Interpretation: Taking the above plot into consideration, the Return of Investment(ROI) is seen to be highest for Referrals.

#Deriving meaning and actionable insight from this data.

Next steps:

We have found a few good insights from the Analysis above and to consolidate we would break our next steps into 2 sections:

(1) For Short-Term:

We will immediately contact and rectify the experience of Cust1 (11) for the order history on 31-12-2019 as the satisfaction rating of 1 is clearly a lower outlier from the analysis above.

Also the entries of Cust12 (56) and Cust23 (83) on 30-06-2020 and 30-09-2021 on Shipping costs repectively are evident outliers on the Higher side increasing our Total costs by a big margin and bringing our profits down so these values of 98𝑎𝑛𝑑

95 on the Shipping Costs are needed to be studied.

(2) For Long-Term:

We will be needing to build strategy on either focusing completely on Business customers to become the category Leader (OR) we have to put in efforts on expanding our Direct market as well since its Purchase Totals are relatively low. Other than that New Orders should only be through Referrals or at best Social Media. Coldcalls for these are actually dropping customer satifaction and causing opportunity cost to our firm.

When we check the growth in the number of orders for the last ten quarters and the ROI generated through different Sources of orders, we can evidently see that the increase is exponentially large for Referrals. So as a company we would like to introduce Loyalty programs or discounts or Referral bonuses to increase and also to sustain the growth through Referral orders.

Library of Python Commands & Analysis

# Read CSV (.csv) and Excel (.xlsx):

df\_I = pd.read\_csv("indie\_playlist\_tracks\_data.csv")

df\_G = pd.read\_excel('GradOutcomeData.xlsx')

# Explains Metrics

df.describe()

# Check current working directory

import os

# Get current directory:

cwd = os.getcwd()

# Change working directory to a path:

os.chdir('C:\\Users\\Moiz Khan')

# Always enter Imports first

# Where as is Alias (for quick reference)

# Numpy Package:

import numpy as np

# Pandas Package:

import pandas as pd

# Matplotlib Package:

import matplotlib.pyplot as plt

# Seaborn Package:

import seaborn as sns

# Indenting is meaningful. Example:

def avg(team\_ages):

sum = 0

for i in team\_ages:

sum = sum + i

average = sum / len(team\_ages)

return average

# You can pass object to funtions:

obj.some\_method = (a, b , c)

# Objects Main:

• Dicts : Key & Values

• NumPy arrays: Tables

• Strings: Characters

• Tuples: Immutable List

# Data Type Scalars:

• str: String type, which holds Unicode (UTF-8 encoded) strings

• bytes: Raw ASCII bytes (or Unicode encoded as bytes)

• float: Double-precision (64-bit) floating-point number

• int: Arbitrary precision signed integer

• bool: A True or False value (short for Boolean)

# DICT Example:

Team9\_Last: ['K', 'J', 'M', 'S', 'N']

Team9\_Age = [30,31,25,24,35]

print('oldest age', Team9\_Age[-1]) # Last

print('youngest age', Team9\_Age[0]) # First

Team9\_Dict=dict(zip(Team9, Team9\_Last)) # Combine

# Changing keys to upper case and values to lower case:

names\_dict = {k.upper(): v.lower() for k, v in Team9\_Dict.items()}

# Changing values to upper case based on condition:

pres\_dict\_C = {pres\_firstyear: pres\_list.upper() for pres\_firstyear,pres\_list in pres\_dict.items() if pres\_firstyear in range(1830,1839)}

## Set Operations:

set1.difference(set2)

set1.symmetric\_difference(set2)

set1.union(set2)

set1.intersection(set2)

# Function Lambda Example: Important!

S\_Lambda = lambda x : sum(x)/len(x)

S\_Lambda(Team9\_Age)

# Number Generation Random:

np.random.seed(12345)

np.random.randn(10)

# Numpy Commands:

# While doing array manipulations, best pratice is to copy the array to new array and then perform manipulations

array1 = np.array([['g','h','i'],['j','k','l']])

array2 = array1.copy()

array2 = array2.reshape(3,2) # Use Reshape to change the dimensions of the array, typically moving from a one dimensional array to a higher dimension

array3 = array1.copy()

array3 = array2.flatten() # Use flatten to change the dimensions of the array from higher dimension to a one-dimensional array

teamn= np.array([[2,4,0],

[7,1,1],

[7,0,3],

[1,0,1],

[7,0,2]])

teamb = np.array([[11,2,98],

[11,21,91],

[6,28,91],

[8,7,88],

[3,2,97]])

sb = np.sqrt(teamb) # square root

mn = np.mean(teamn) # mean

array\_add = np.add(teamb,teamn) # add

array\_subtract = np.subtract(teamb,teamn) # subtract

# List Combine to create Dataframe:

s = pd.Series(['year','name', 'party'])

df=pd.DataFrame(data=d,columns=s)

df\_slice=df[df['name']=='James Clark'] # Slicing/filtering of dataframe based on condition

print(df\_slice)

df=df.set\_index('year')

df\_slice1=df.loc[1940:1960] # Slicing/filtering for range of values

#.loc is important. Study during Winter Break.

df['party'].value\_counts()['Whig'] # Count of specific value in a column using Value\_counts()

str(df[df['party']=='Whig']).count('Whig') # Count of specific value in a column using count()

df.groupby(['party'])['party'].count() # Count within each group

# Commands for Data Frames:

# Like Dplyer in R and similar to joins in SQL

df\_Merge = pd.merge(df\_I, df\_IT, how='inner', on='playlist\_id')

# Operation on Agg:

df\_count = df.groupby('State').agg({'count':'sum'})

I\_mean = df\_I.groupby('playlist\_name',as\_index=False).agg({'duration':'mean'})

I\_max = df\_I.groupby('playlist\_name',as\_index=False).agg({'duration':'max'})

I\_rename = df\_I.groupby('playlist\_name',as\_index=False).agg({'duration':'mean'}).rename(columns={'duration':'avg\_duration'})

I\_sort = df\_I.groupby('playlist\_name',as\_index=False).agg({'duration':'mean'}).rename(columns={'duration':'avg\_duration'}).sort\_values('avg\_duration',ascending=False)

df\_sc = df\_sc.sort\_values(by='count',ascending=True)

# Checking for duplicates values

data1.duplicated(keep = 'last').sum()

data2.duplicated(keep = 'last').sum()

data3.duplicated(keep = 'last').sum()

# No duplicates found

# Checking for NA values

data1.isna().sum()

data2.isna().sum() # Here need to replace the 'na' values with some string in Referral ID variable

data3.isna().sum() # Here the type of order 'na' values must be replaced with 'Repeat' as the company stopped labelling them

# Cleaning data2

# Filling the Referral ID variable 'na' values with None so that it is easy for further analysis

data2['Referral ID'] = data2['Referral ID'].fillna('None')

# Cleaning data3

# labelling the 'na' value in Type of Order with 'Repeat'

data3['Type of Order'] = data3['Type of Order'].fillna('Repeat')

# The date variable has 'Na' values. So seeing after the data the Na values needed to be ffilled so that the date gets copied

data3['Date'] = data3['Date'].fillna(method = 'ffill')

# Filling the Na values in the Shipping Costs column with '0' as mentioned in Data information

data3['Shipping Costs'] = data3['Shipping Costs'].fillna(0)

# Unstack:

df\_u = df\_mean.unstack('Type of Customer')

# Datatypes in Dataframe for Columns:

df.dtypes

# Group by using two parameters with aggregate:

df\_mean = df.groupby(['Type of Customer', 'Date'])['Purchase Total'].agg(['mean'])

# Rolling Averages:

df\_resample.rolling(10).mean()

# Graphs:

df.plot(kind='bar', stacked=True) #or

df\_Sc.plot.bar(figsize=(10,5),ylabel='Number of counties')

# Matplotlib Plots:

plt.plot(df\_state\_count)

# Seaborn Plots:

sns.catplot(x="PROGRAM", y="SALARY MID", hue="GENDER", kind="bar", data=df)

sns.lineplot(x="GRADTERM", y="SALARY MID", data=df,hue='PROGRAM',size\_order=['AUG 16','DEC 16','MAY 17','AUG 17','DEC 17','MAY 18','AUG 18','DEC 18','MAY 19'])

sns.violinplot(x='Group', y='Outcome',hue="Group", data=df2)

sns.boxplot(x='Group', y='Outcome',hue="Group", data=df2)

sns.histplot(x='Outcome', hue='Group', data=df2)

sns.displot(x= "Number of Orders", hue='Type of Order', bins=10, data=df, kde=True)

# Statistical tests:

# Running the linear regression for the above continuis plots

res = linregress(data\_123['Avg COA'], data\_123['Satisfaction Rating'])

print('Satisfaction rating, Avg COA, r=', res.rvalue, 'p =', res.pvalue)

print('Slope', res.slope)

# T-test:

from scipy.stats import ttest\_ind

data\_New = data\_123[data\_123['Type of Order'] == 'New']

data\_Repeat = data\_123[data\_123['Type of Order'] == 'Repeat']

t,p = ttest\_ind(data\_New['Purchase Total'], data\_Repeat['Purchase Total'])

print(t,p)

#DateTime:

date\_range function allows you to create a DatetimeIndex with a specified length

• Downsampling default is to have the left bin edge inclusive

• 00:00 value is included in the 00:00 to 00:05

interval;

• can specify closed='right’

• Upsampling (converting from a low frequency to higher

frequency) does not require aggregation

• will produce missing data when we use the

asfreq method without any aggregation

• can use fill methods to fill missing data

Problem: Build functions and give insights on existing data for Analysis for Media Client

Solution: Using Panda Dataframes join, aggregate and group to apply Statistics

import numpy as np

import pandas as pd

#The country\_playlist\_tracks\_data.csv data contains a variety of information on nearly 1,550 country music songs popular on Spotify. Load the data and assign it to the variable df. Then collect summary statistics from all columns using the .describe() method.

df = pd.read\_csv('country\_playlist\_tracks\_data.csv')

q1 = df.describe()

q1

df = pd.read\_csv('country\_playlist\_tracks\_data.csv')

q1 = df.describe()

q1

#Finding the artist in the data set with the longest artist\_name. Use a list comprehension to create a new list to hold the lengths of each name, sort the list, then slice the DataFrame to find all values where artist\_name has the same length as the largest value. (Note: recall that lists use zero indexing, so [0] will access the first element in a list and [-1] will access the last element in the list.)

lengths = df["artist\_name"].str.len()

argmax = np.where(lengths == lengths.max())[0]

q2 = df.iloc[argmax]["artist\_name"]

print(q2)

​

df['anlength'] = df['artist\_name'].str.len()

anlsort = sorted(df['anlength'], reverse = True)

​

df[lengths == max(anlsort)]

#Using groupby to group the popularity variable by the playlist\_name. Which playlist\_name has the highest mean popularity? Remember to chain .sort\_values(ascending=False) to sort the resulting data on their values and not alphabetically by playlist\_name.

q3= (df.groupby(['playlist\_name'])['popularity'].mean())

q3.sort\_values(ascending = False)

#Creating a Pandas pivot\_table to count the number of songs from each playlist\_name. Use popularity as the aggregation value again, but this time pass aggfunc='count' to report the number of songs. Which playlist\_name has the most songs? Remember to sort the values in descending order, and include by='popularity' to sort on the counted data.

q4 = pd.pivot\_table(df, index = ['playlist\_name'], values = 'popularity', aggfunc = 'count')

q4.sort\_values('popularity', ascending = False)

#Create a Pandas crosstab to compare the number of songs with each mode by key. Set normalize='all' and margins=True to view the data as a percent of the total. What combination of mode and key is the most rare in our data?

q5 = pd.crosstab(df['mode'], df['key'], normalize='all', margins = True)

q5

#Now we will look at popular indie tracks from Spotify. Load the 'indie\_playlist\_tracks\_data' file and aggregate the data by playlist\_name to count how many songs there are from each playlist. Remember to sort the values from largest to smallest. How do the number of songs from the top Indie playlist compare to the number of song from the top Country playlist?

df2 = pd.read\_csv('indie\_playlist\_tracks\_data.csv')

q6 = df2['playlist\_name'].value\_counts()

q6.sort\_values(ascending = False)

#Using the Indie song data, group song duration data by playlist\_name to find the playlist with the highest average (mean) duration. Then group the India duration data again to find the playlist with the longest max duration. Is there a difference between the two? Remember to sort the values from largest to smallest.

q7a= (df2.groupby(['playlist\_name'])['duration'].mean())

q7a.sort\_values(ascending = False)

#Using Pandas indexing to find the row in the Indie data with the highest-rated song using popularity. How many popularity points above the average popularity is this popularity?

q8 = df2.max('index' == 'popularity')

q8

df2['popularity'].max() - df2['popularity'].mean()

​

#Read in the indie\_playlist\_data.

#Print the shape of the indie\_playlist\_tracks\_data data set and the indie\_playlist\_data data set. Merge the two DataFrames together on playlist\_id and print the shape of the resulting dataframe.

#Notice the number of columns in the resulting data frame one less than the sum of the number of colums in both the original data frames.

df3 = pd.read\_csv('indie\_playlist\_data.csv')

print(df2.shape)

​

print(df3.shape)

​

q9 = pd.merge(df2, df3, how = 'inner', on = 'playlist\_id')

print(q9.shape)

#Because merging has been done on that one common values column i.e 'playlist\_id. It is the same for both the dataframes and hence not repeated.

#Using np.where() to add an indicator variable identifying songs that have popularity scores greater than or equal to 80. Call this new variable popular. Create a pivot table using the combined playlist data. Use 'popular' as the index value and duration to fill the table. Aggregate using the mean. Do the more popular songs tend to be longer or shorter than less popular songs?

q9['popular'] = np.where(q9['popularity'] >= 80, 'Unpopular', 'Popular')

​

q0 = pd.pivot\_table(q9, index = 'popular', values = ['duration'], aggfunc = 'mean')

​

q0

More popular songs tend to be slightly shorter than the unpopular ones. On average 4975 shorter.