**Module 4**

Here are a few helpful downloads for this module:

* [Video Transcripts](https://student.emeritus.org/courses/4765/files/2949437?wrap=1)
* [Download Video Transcripts](https://student.emeritus.org/courses/4765/files/2949437/download?download_frd=1)
* [Quick Reference Guide](https://student.emeritus.org/courses/4765/files/2949448?wrap=1)

*#raise NotImplementedError()*

site\_visits\_df = pd.merge(left=site, right=visited, left\_on='name', right\_on='site', how='inner')

*#raise NotImplementedError()*

visited\_renamed = visited.rename(columns = {'site' : 'name'})

site\_visits\_df2 = pd.merge(left=site, right=visited\_renamed, on='name', how='inner')

*#raise NotImplementedError()*

survey\_renamed = survey.rename(columns = {'taken' : 'id'})

survey\_site\_visits = pd.merge(left=site\_visits\_df2, right=survey\_renamed, on='id', how='inner')

survey\_site\_visits\_ = pd.merge(survey\_.rename({'taken': 'id'}, axis = 1),

site\_visits\_df2\_, on = 'id')

*#raise NotImplementedError()*

survey\_site\_visits\_renamed = survey\_site\_visits.rename(columns = {'person' : 'person\_id'})

person\_renamed = person.rename(columns = {'id' : 'person\_id'})

full\_name\_df = pd.merge(left=survey\_site\_visits\_renamed, right=person\_renamed, on='person\_id', how='inner')

left\_ = survey\_site\_visits\_.rename({'person': 'person\_id'}, axis = 1)

right\_ = person\_.rename({'id': 'person\_id'}, axis = 1)

full\_name\_df\_ = pd.merge(left\_, right\_, on = 'person\_id')

*#raise NotImplementedError()*

ans5 = pd.merge(left=df1, right=df2, on='name', how='left')

*# Answer check*

print(type(ans5))

ans5

geofin = pd.merge(left = demographics, right = financials, on = 'id')

ans1 = geofin.query('country == "Kenya"')[['funded\_amount']].agg('mean').values[0]

ans1\_ = pd.merge(demographics\_.loc[demographics\_['country'] == 'Kenya'], financials\_, on = 'id')[['funded\_amount']].mean().values[0]

geofinuse = pd.merge(left = geofin, right = use, on = 'id')

ans2 = geofinuse.query('country == "El Salvador"').groupby('sector')[['funded\_amount']].agg('sum').reset\_index().sort\_values(by='funded\_amount',ascending=**False**)['sector'][0]

ans2\_ = pd.merge(demographics\_.loc[demographics\_['country'] == 'El Salvador'], use\_, on = 'id')['activity'].value\_counts().index[0]

ans3 = geofinuse.query('country == "Pakistan" and sector == "Agriculture"').groupby('sector')[['funded\_amount']].agg('sum').reset\_index()['funded\_amount'][0]

use\_ = pd.read\_csv('data/kiva/use.csv')

p1\_ = pd.merge(use\_, demographics\_, on = 'id')

a\_ = pd.merge(p1\_, financials\_, on = 'id')

b\_ = a\_.loc[a\_['country'] == 'Pakistan']

ans3\_ = b\_.loc[b\_['activity'] == 'Agriculture'][['funded\_amount']].sum().values[0]

ans4 = geofinuse.groupby('sector')[['funded\_amount']].agg('sum').reset\_index().sort\_values(by='funded\_amount',ascending=**False**)['sector'][0]

ans4\_ = pd.merge(financials\_, use\_, on = 'id').groupby('activity')[['funded\_amount']].sum().sort\_values(by = 'funded\_amount', ascending = **False**).index[0]

geofinusecrowd = pd.merge(left = geofinuse, right = crowdsource, on = 'id')

geofinusecrowd['dollar\_to\_lender\_ratio'] = geofinusecrowd['funded\_amount'] / geofinusecrowd['lender\_count']

ans5 = geofinusecrowd.groupby('sector')[['dollar\_to\_lender\_ratio']].agg('max').reset\_index().sort\_values(by='dollar\_to\_lender\_ratio',ascending=**False**)['sector'][0]

ans5\_ = b\_.groupby('activity')[['ratio']].sum().sort\_values('ratio', ascending = **False**).index[0]

px.scatter(gapminder, x='gdpPercap', y='lifeExp')

px.scatter(gapminder, x='gdpPercap', y='lifeExp', color='country')

px.scatter(gapminder, x='gdpPercap', y='lifeExp', color='country', size='pop')

px.scatter(gapminder, x='gdpPercap', y='lifeExp', color='country', size='pop', log\_x=True)

px.scatter(gapminder[gapminder['year'] == 2007], x='gdpPercap', y='lifeExp', color='country', size='pop', log\_x=True)

px.box(gapminder, x='gdpPercap', y='continent', color='continent')

*#raise NotImplementedError()*

ans1 = russian\_states[russian\_states['Economic region'].str.contains('Siberian')]

df.isnull().sum().sort\_values().plot(kind = 'bar')

plt.savefig('images/missing\_plot\_.png')

plt.close();

*#raise NotImplementedError()*

df.isna().sum().sort\_values().plot(kind='bar')

plt.savefig(‘images/missing\_plot.png')

plt.close()

*#raise NotImplementedError()*

ans4 = ans3.apply(**lambda** x : x[-1])

print(type(ans4))

ans4.head()

**Issues**

Activity 4.1 - Problem 5 rename variables:

print(type(ans5))

ans5

Activity 4.4 - Problem 4, asking “***float***”, however, it is expecting float64 to be more specific!

Activity 4.6 @ Problem 6

**Quizes**

The pandas function rename() is used to rename a column in a dataframe. : True

*You are correct! The answer “True” is correct because the*rename()*function is used to change column names in pandas dataframes.*

What is the statement to rename a column named Code to Country in dataframe df? : df.rename(columns ={“Code” : “Country”})

*You are correct! The answer “*df.rename(columns ={“Code” : “Country”})*” is correct because the parameters of the function were formed correctly, with the column names in quotation marks, separated by a colon, and in parentheses.*

The function merge() is used to join two datasets into one and align the rows from each based on their common columns. : True

*You are correct! The answer “True” is correct because the function*merge()*joins two datasets into one new dataframe based on the similar columns in the source datasets.*

Given two dataframes (df1 and df2) with common Entity columns, what will be the Python statement for joining them into a new dataframe (df3)? : df3=pd.merge(left=df1, right=df2, left\_on=”Entity”,right\_on=”Entity”)

*You are correct! The answer “*df3=pd.merge(left=df1, right=df2, left\_on=”Entity”,right\_on=”Entity”)*” is correct because all four parameters are correctly identified (i.e.,*left*,*right*,*left\_on*, and*right\_on*).*

The function merge() is used to join two datasets into one. Which constructor is used to declare the type of join? : how

*You are correct! The answer “how” is correct because the constructor how is used to declare the type of join to be made.*

The left join returns all rows from a table declared using right= and only the matching rows from a table declared using left=. : False

*You are correct! The answer “False” is correct because the left join returns all rows from the*left=*table and only the matching rows from the*right=*table.*

What is the type of join where the resulting table contains only rows from both tables where the joining condition is met? :  Inner join

*You are correct! The answer “Inner join” is correct because in this type of join, the final table has only those rows from both tables where the joining condition is met.*

df.sort\_values(“Column”) sorts a dataframe in ascending or descending order using Column. : True

*You are correct! The answer “True” is correct because the pandas*sort\_values()*function sorts a dataframe in ascending or descending order using Column.*

Using merge(), the join condition of two separate dataframes cannot use more than one column. : False

*You are correct! The answer “False” is correct because a join in two separate dataframes can use more than one column.*

Given two dataframes (df1 and df2) with two join conditions on columns Entity and Year, what is the statement for joining them into a new dataframe (df3)? : df3=pd.merge(left=df1, right=df2, left\_on=[”Entity”,”Year”],right\_on=[”Entity”,”Year”])

*You are correct! The answer “*df3=pd.merge(left=df1, right=df2,left\_on=[”Entity”,”Year”],right\_on=[”Entity”,”Year”])*” is correct because this is the correct syntax in Python to merge df1 and df2 on the columns Entity and Year.*

Consider the following line of code:

df3=pd.merge(left=df1, right=df2, left\_on=”Entity”,right\_on=”Entity”,how=?)

What would you put in the how= parameter to include all the rows from the left table and the rows matching the join condition from the right table? : Left

*You are correct! The answer “Left” is correct because this type of join returns all records from the left table and the matched records from the right table.*

The function reset\_index()is used to set an index of a dataframe. : False

*You are correct! The answer “False” is correct because*reset\_index()*is used to reset the index of a dataframe object to default indexing.*

What function is used to drop all NaN values from a dataframe? : dropna()

*You are correct! The answer “*dropna()*” is correct because the function in pandas is used to drop all the rows from a dataframe that have a NaN value.*

The scatterplot’s primary use is to show relationships between two numeric variables. : True

*You are correct! The answer “True” is correct because the dots in a scatterplot not only report the values of individual data points, but also the patterns when the data is taken as a whole. That shows the relationship between the variables.*

Given a dataframe with the columns Year, Entity, and gdp\_ratio, how would you draw a scatterplot to show the trends of GDP ratio per year for each entity? : px.scatter(df, x= “Year”, y = “gdp\_ratio” , color = “Entity”)

*You are correct! The answer “*px.scatter(df, x= “Year”, y = “gdp\_ratio” , color = “Entity”)*” is correct because this is the correct syntax to create a scatterplot to see the trends of GDP ratio per year for each entity.*

In the Python function px.scatter(), size= is used to set the size of the graph. : False

*You are correct! The answer “False” is correct because in the function px.scatter(), the constructor size= is used to set the size of the markers in the plot according to the given values.*

Given a dataframe df1 with the columns Entity, Code, Year, and Expectancy, how can it be filtered into a dataframe without the column Code? : df1=df1[[“Entity”,”Year”,”Expectancy”]]

*You are correct! The answer “*df1=df1[[“Entity”,”Year”,”Expectancy”]]*” is correct because this is the correct syntax to select all the columns except the column Code.*

To make a scatterplot using a logarithmic scale, which constructor in function px.scatter() is set to true? : log\_x = True

*You are correct! The answer “*log\_x = True*” is correct because this is the constructor that is used to turn on logarithmic scales in a scatterplot.*

In which library is px.histogram(df[“column”]) used to create a histogram? : Plotly

*You are correct! The answer “*Plotly*” is correct because the function px.histogram() is used to create a histogram in the Python library Plotly.*

sns.displot() is used to create a scatterplot in Seaborn. : False

*You are correct! The answer “*False*” is correct because the function sns.displot() in Seaborn is used to create a histogram.*

Using sns.displot() in Seaborn, what constructor would you use to also display the kernel density estimate curve on the histogram? : kde = True

*You are correct! The answer “*kde = True*” is correct because this is the constructor used in function*sns.displot()*to set the kernel density estimate curve on the histogram.*

The violin plot tells the kernel density estimate of a variable in a dataframe. : True

*You are correct! The answer “*True*” is correct because the violin plot shows the sidewise kernel density estimate of a column in a dataframe.*

What does the middle line of a box plot represent? : Median

*You are correct! The answer “*Median*” is correct because the middle line of a box plot is used to show the median of a dataset.*

What are the constructors used in the function px.scatter() to create x- and y-axes marginal plots? : **marginal\_x, marginal\_y**

*You are correct!*

In Seaborn, which plot function is used to quickly visualize and analyze the relationship between two variables and describe their individual distributions on the same plot? : sns.jointplot()

*You are correct! The answer “*sns.jointplot()” *is correct because this type of plot function is used to visualize the relationship between two variables and describe their individual distributions on the same plot.*

The statement df1[df1[“Entities”].str.contains(“in”)] gives only entities whose name contains the substring in. : True

*You are correct! The answer “*True” is*correct because the function*str.contains()*is used to get a subset of the data where the given value contains a substring.*

Which statement would return rows only for entities that start with “F”? : df1[df1[“Entities”].str.startswith(“F”)]

*“*df1[df1[“Entities”].str.startswith(“F”)]” is*correct because the syntax is correct, and the function startswith() is used to filter the data where values start with a given letter.*

str.upper() is used to get all the values that are in uppercase letters. : False

*You are correct! The answer “False*” is*correct because the function upper() is used to convert the values into uppercase letters.*

Which function is used to replace a string value? : str.replace()

*You are correct! The answer “*str.replace()” is*correct because the function replace() is used to*replace a value.

to\_numeric(“column”) converts the data type of column into (blank). : int64

*You are correct! The answer “*int64” is*correct because the function*to\_numeric(“column”)*in Python converts the data type of column into int64.*

**Discussion Activity**

**4.1**

**Does income differentiate customers who purchase wine?**

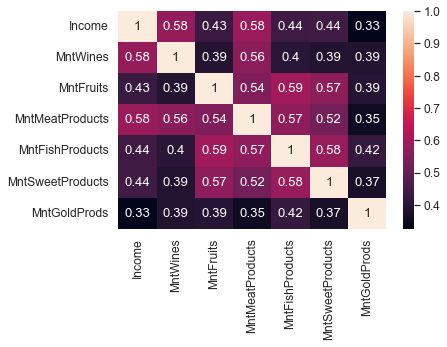
plt.tight\_layout()

sns.set(font\_scale=1.1)

sns.heatmap(data = df[['Marital\_Status', 'Income', 'MntWines', 'MntFruits',

'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',

'MntGoldProds']].corr(), annot=True)



As shown in the heat map, income plays a significant role in wine purchase, they are strongly correlated as well as meat products, there is also high correlation with other purchases too.

**What customers are more likely to participate in the last promotional campaign?**

sns.barplot(data = df[['Marital\_Status', 'Income', 'Education',

'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',

'AcceptedCmp2', 'Response']], x='Education', y='Response')

plt.xticks(rotation=90)

Chart, bar chart

Description automatically generated

I looked at the education level of individuals, as the education gets higher the more participation in promotional campaigns.

**Income versus Monthly Web Visits**

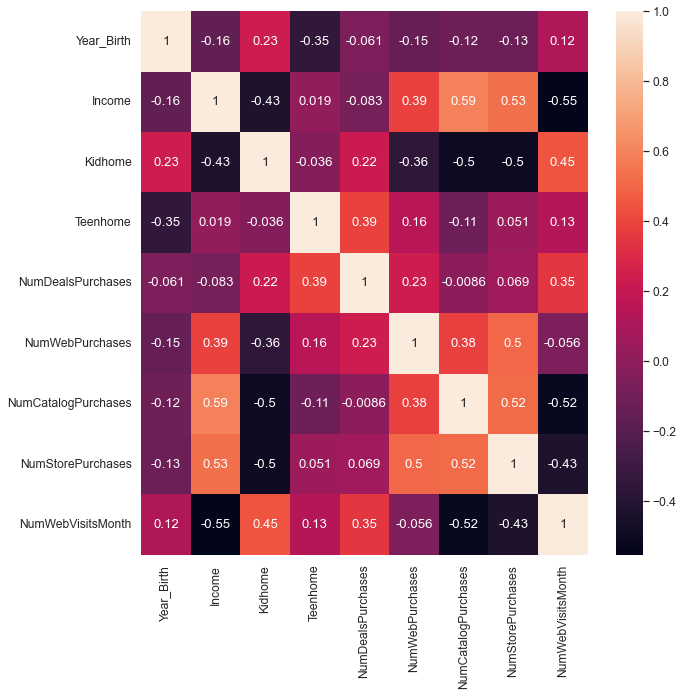
plt.tight\_layout()

plt.subplots(figsize=(10,10))

sns.heatmap(data = df[['Year\_Birth', 'Education', 'Marital\_Status', 'Income', 'Kidhome',

'Teenhome', 'Dt\_Customer','NumDealsPurchases', 'NumWebPurchases',

'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth']].corr(), annot=True)



The result in this dataset is interesting as income increases less frequent web visits, however, catalog and in-store purchases higher as well as web purchases surprisingly.

**Complaints about Product**

sns.barplot(data=df.query('Complain == 1')[['MntWines', 'MntFruits',

'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',

'MntGoldProds', 'Complain']].groupby('Complain').sum().reset\_index()[['MntWines', 'MntFruits',

'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',

'MntGoldProds']])

plt.xticks(rotation=90)

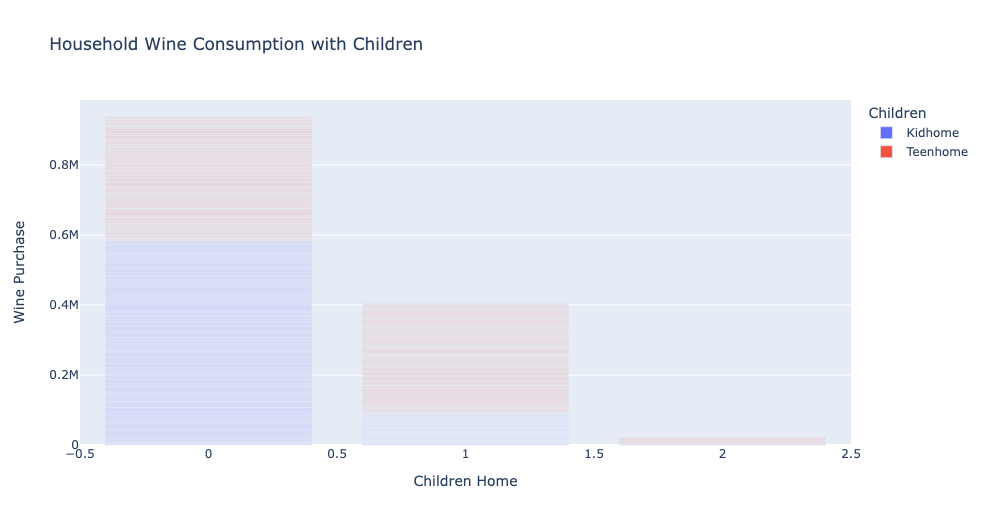
Chart, bar chart

Description automatically generated

Wine and Meat products lead number of complaints.

**Do people with children purchase more wine?**

px.bar(df[['Marital\_Status', 'Kidhome', 'Teenhome', 'MntWines']], x=['Kidhome','Teenhome'], y='MntWines', title='Household Wine Consumption with Children', labels={"value":"Children Home", "MntWines":"Wine Purchase", "variable":"Children"})



Wine consumers are households without any children by far.

**4.2**

**Transformation**

Features in this dataset are not to start with, there are some critical information taken out (deliberately) like “*income*” which makes harder to correlate. I analyzed fields by value\_counts() method: *credit.savings\_status.value\_counts().*

**Identification**

However, I converted almost all object fields to numeric, so, I can correlate them all to see on *heat map*, I cleaned up a few fields which do not correlate:

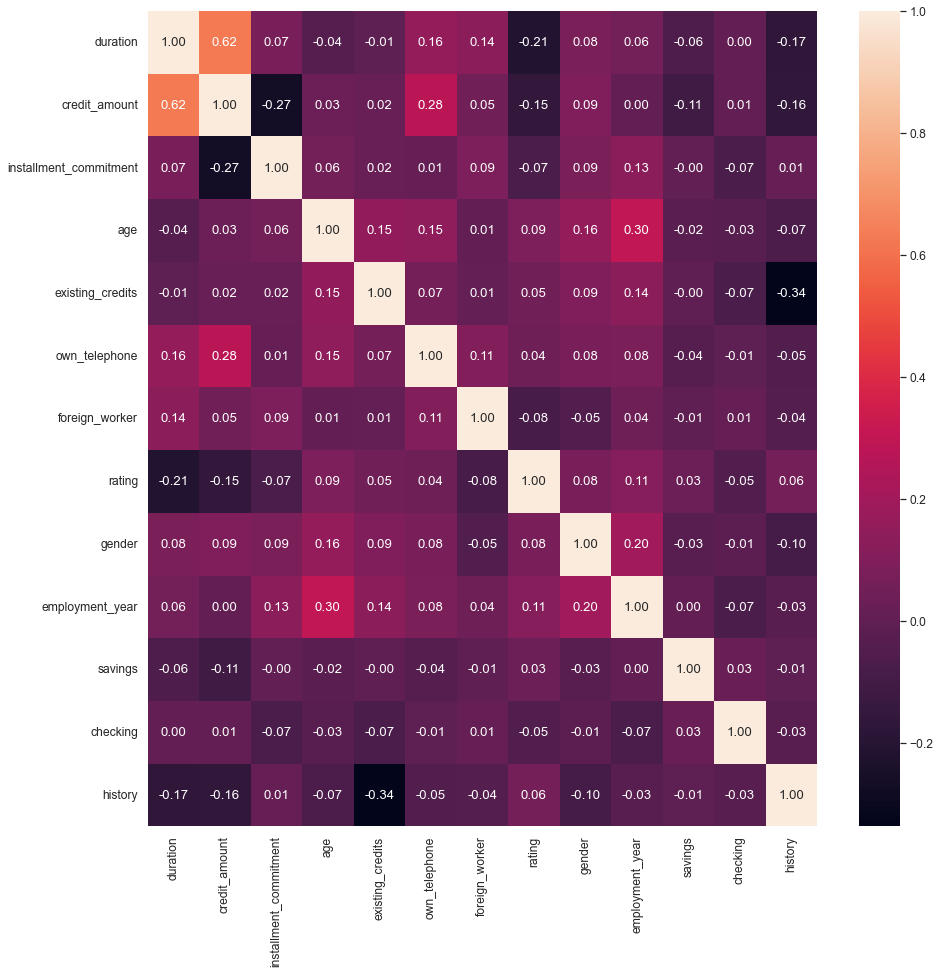
plt.tight\_layout()

plt.subplots(figsize=(15,15))

sns.set(font\_scale=1.1)

sns.heatmap(credit[['duration', 'credit\_amount', 'installment\_commitment', 'age', 'existing\_credits', 'own\_telephone',

'foreign\_worker', 'rating', 'gender', 'employment\_year', 'savings', 'checking', 'history']].corr(), annot=True, fmt='.2f')



Credit *rating* numeric bad=0 or good=1 as seen in heat map there is no strong correlation, two fields stand out *duration* and *credit\_amount* which are negative correlation, there is also weak positive correlation to *employment\_year* (length of employment in years) towards good credit. Other features have negligible impact on the credit rating!

**Deep Dive**

Next, I looked at these 3 features more closely, I used Plotly violin plot to identify thresholds of each for eyeballing.

Credit Amount, risk increases as the amount gets bigger, so, I captured the median value from good credit. $2244.

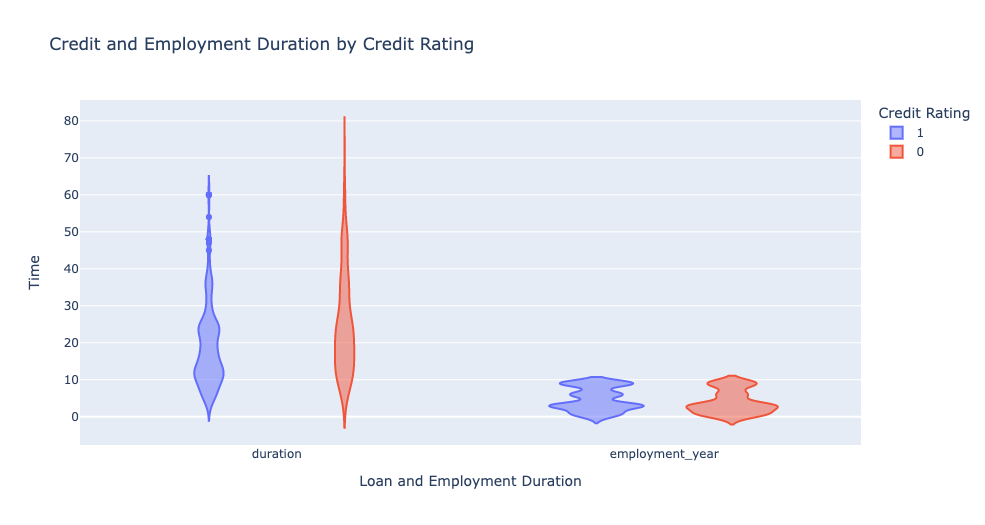
Loan duration, risk increases as loan duration gets longer, so, I captured the upper fence value from good credit. 42 months.

Employment duration, risk gets lower as employment gets longer, so, I captured 3rd quartile value 6 years from the violin plots, although, it has very less influence on correcting numbers by only 2%:

px.violin(credit[['credit\_amount','rating']], color='rating', title='Credit Amount by Credit Rating', labels={"variable":"Credit Amount", "value":"$", "rating":"Credit Rating"})



px.violin(credit[['duration','employment\_year','rating']], color='rating', title='Credit and Employment Duration by Credit Rating', labels={"variable":"Loan and Employment Duration", "value":"Time", "rating":"Credit Rating", "duration":"Loan Duration", "employment\_year":"Employment Duration"})



**Threshold Result**

Finally, I applied the threshold values to the formula to detect people with bad credit ratings:

credit[(credit['duration'] > 42) & (credit['employment\_year'] < 6) & (credit['credit\_amount'] > 2244)].rating.value\_counts(normalize=True)

0 0.588235

1 0.411765

Name: rating, dtype: float64

versus 30% bad, 70% good credit rating in the entire dataset!

 ————— o —————