```
#
# Purdue University Global
#
# IN402 - Modeling and Predictive Analysis
#
# Unit 7 Assignment / Module 5 Part 1 Competency Assessment
#
# Generating Descriptive Statistics
#
# PyCharm Code
#

# Library and data import.

# import all necessary initial libraries

import sys

# Ignoring warnings

if not sys.warnoptions:
    import warnings

warnings.simplefilter("ignore")

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

# Output Header

print('Unit 7 Assignment / Module 5 Part 1 Competency Assessment Output\n')

from datetime import datetime

print(datetime.now().strftime("%m/%d/%Y %H:%M:%S"), '\n')

# Import the dataset into the development environment.

df = pd.read_csv('/home/codio/workspace/data/IN402/Churn_Modelling.csv')

# In the paper, describe the datasource and what you intend to use
# the libraries for.

# Explorative Analysis.

# Explore the contents of the dataset using .head()

print(df.head(10))

print()

# Explore the contents of the dataset using tail()

print(df.tail(10))

print()

# Check if there are any missing values using isnull() functions,
# and remove them using .dropna() function (if any).

print(df.isna())

print()

# Check if there are null values anywhere

print(df.isnull().sum())

print()

# Check the structure and if there are any missing values using .info() function.

print(df.info())

print()

# Check the descriptive statistics on numeric variables
# using the .describe() function.

print(df.describe())

print()

# Check the variance of each variable

# Importing statistics module

from statistics import variance

# Set attribute values

creditScore = df['CreditScore']

age = df['Age']

tenure = df['Tenure']

balance = df['Balance']

estimatedSalary = df['EstimatedSalary']

# Display variance values

print("Variance of CreditScore is % s "% (variance(creditScore)))

print()

print("Variance of Age is % s "% (variance(age)))

print()

print("Variance of Tenure is % s "% (variance(tenure)))

print()

print("Variance of Balance is % s "% (variance(balance)))

print()

print("Variance of EstimatedSalary is % s " %(variance(estimatedSalary)))

print()

# Build a plot to visualize customers that churned and that did not churn.

# 1st plot - between customers that churned and that did not churn

plt.figure(figsize=(10,5))

sns.countplot(x = "Exited", data = df)

plt.show()

# Calculate the percentage of churned customers

# Percentage of churned customers

total_customers = len(df.index)

customers_churned = df.groupby('Exited').Exited.count()[1]

perc_cust_churned = customers_churned/total_customers

print("Percentage of Churned Customers = ", perc_cust_churned*100, "%")

print()

# Build a histogram of credit scores for all customers

df['CreditScore'].plot.hist(bins=100, figsize=(10,5))

# Identify unique values in the Geography column

df['Geography'].unique()

# Plot the geography for all customers

plt.figure(figsize=(10,5))

sns.countplot(x='Geography', hue='Exited', data=df)

plt.show()

# Plot the geography for churned/non-churned customers

plt.figure(figsize=(10,5))

sns.countplot(x='Geography', hue='Exited', data=df)

plt.show()

# Plot the gender by the churn status

plt.figure(figsize=(10,5))

sns.countplot(x = "Exited", hue = "Gender", data = df)

plt.show()

# Calculate the percentage of customers by gender

churned_by_gender = df.groupby(['Gender'])['Exited'].sum()

print(churned_by_gender)

print()

# Calculate the churn number by gender (Male):

churned_males = churned_by_gender['Male']

print('Churned males: '+ str(churned_males))

print()

# Calculate the churn number by gender (Female):

churned_females = churned_by_gender['Female']

print('Churned females: '+ str (churned_females))

print()

# Plot a histogram to compare the age by churn status

# Compare the age for churned

plt.figure(figsize=(10,5))

df['Age'].plot.hist()

# Plot a boxplot to identify the churned/non-churned customers by age

plt.figure(figsize=(10,5))

sns.boxplot(x="Exited", y="Age", data=df)

plt.ylim(0, 100)

plt.show()

# Plot the tenure for churned/non-churned customers

plt.figure(figsize=(10,5))

sns.countplot(x='Tenure', hue='Exited', data=df)

plt.show()

# Plot the histogram of balance for all customers.

plt.figure(figsize=(10,5))

df['Balance'].plot.hist()

# Plot a number of products by churned/non-churned status

plt.figure(figsize=(10,5))

sns.countplot(x='NumOfProducts', hue='Exited', data=df)

plt.show()

# Plot the Credit Card ownership by churned/non-churned status

plt.figure(figsize=(10,5))

sns.countplot(x = "HasCrCard", hue = "Exited", data = df)

plt.show()

# Calculate the credit card ownership by churned/non-churned status

churned_by_cc = df.groupby(['HasCrCard'])['Exited'].sum()

churned_no_cc = churned_by_cc[0]

print('Churned with credit card: '+ str(churned_no_cc))

print()

# Calculate the credit card ownership by churned/non-churned status

churned_cc = churned_by_cc[1]

print('Churned with no credit card: ' + str (churned_cc))

print()

# Plot the active hours for the customers by churned/non-churned status.

plt.figure(figsize=(10,5))

sns.countplot(x = 'IsActiveMember', hue = "Exited", data = df)

plt.show()

# Plot the estimated salary for all customers

df['EstimatedSalary'].plot.hist(bins=10000, figsize=(10,5))

plt.xlabel('EstimatedSalary')

plt.show()
```