**Fall 2023 DATA 230 – 11**

**Data Visualization**

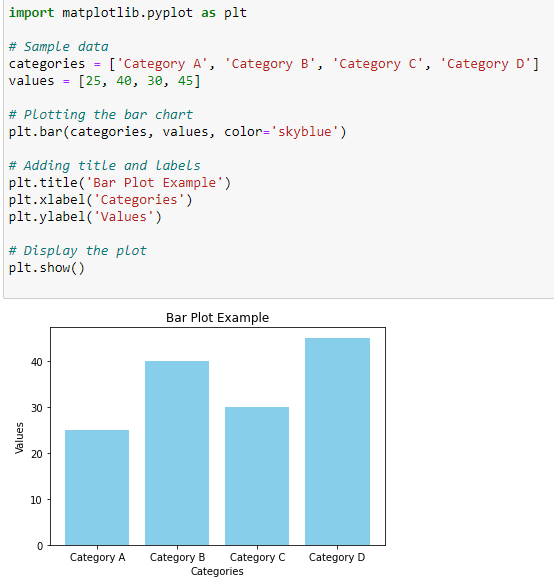
**Homework – 3**

**Name :- Prayag Nikul Purani**

**SJSU Id :- 017416737**

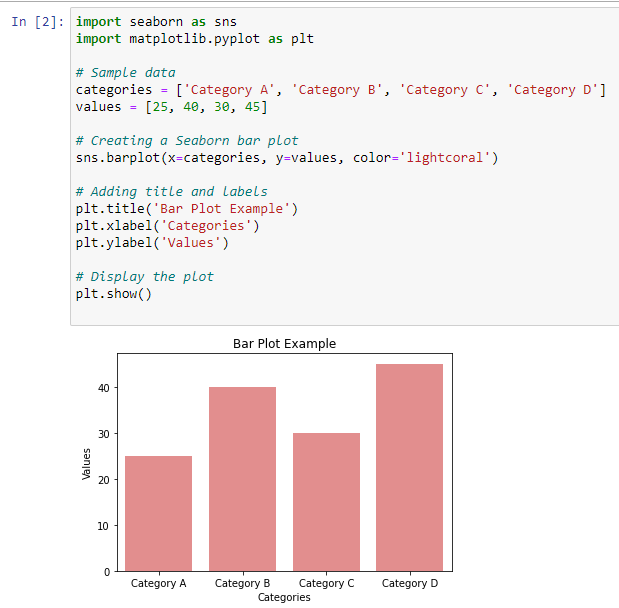
**Question 1:**

**Visualization 1: Bar plots using Matplotlib**



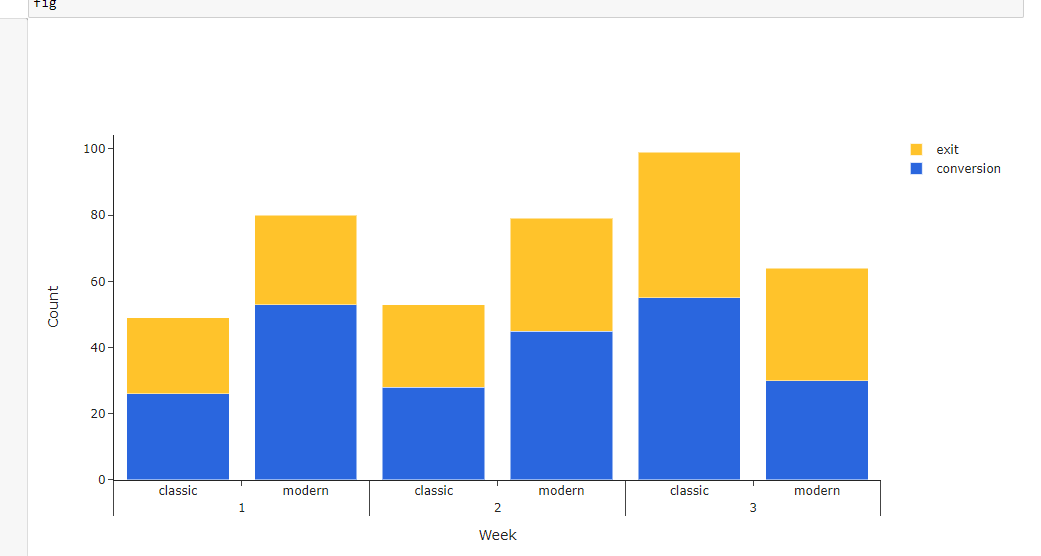
Importing matplotlib: The first line imports the matplotlib.pyplot module, which provides a convenient interface for creating various types of plots. Sample data: Define the data you want to visualize. In this case, we have a list of categories (categories) and corresponding values (values). Create a bar plot: Use the plt.bar() function to create a bar plot. The first argument is the x-axis values (categories), the second argument is the y-axis values (values), and the color parameter sets the color of the bars. Adding labels and title: Use plt.xlabel(), plt.ylabel(), and plt.title() to add labels to the x-axis, y-axis, and a title to the plot, respectively. Display the plot: Finally, use plt.show() to display the plot.

**Visualization 2: Bar Plot using Seaborn**



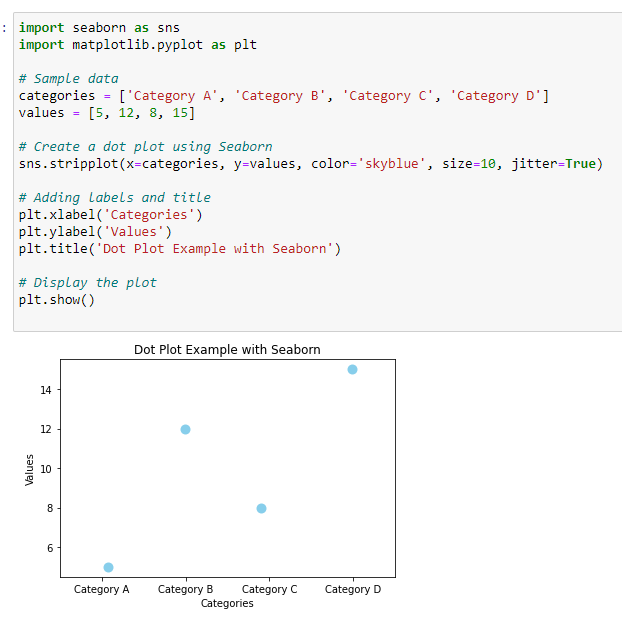
Importing seaborn and matplotlib.pyplot: The first two lines import the seaborn library and matplotlib.pyplot for creating plots. Sample data: Define the data you want to visualize. In this case, we have a list of categories (categories) and corresponding values (values). Create a bar plot using Seaborn: Use the sns.barplot() function to create a bar plot. Specify the x-axis values (x=categories) and y-axis values (y=values). The color parameter sets the color of the bars. Adding labels and title: Use plt.xlabel(), plt.ylabel(), and plt.title() to add labels to the x-axis, y-axis, and a title to the plot, respectively. Display the plot: Finally, use plt.show() to display the plot.

**Visualization 3: Grouped & Stacked bars**

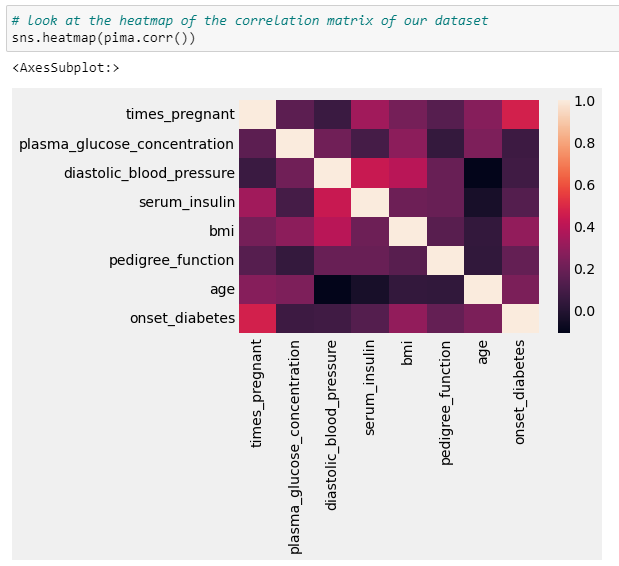


plotly makes it easy to create an interactive stacked or grouped bar chart in Python by assigning the desired type to the layout attribute [barmode](https://plotly.com/python/reference/layout/" \l "layout-barmode" \t "_blank). Unfortunately, barmode only takes either stack or group but not both as an argument. It seems like they are currently working on this option, but there is a workaround for now: subcategory axes! To plot this, we need a categorical x-axis that shows the week, stacked bars to show the number of conversions and exits, and those bars grouped into modern and classic. Since barmode cannot be stack and group at the same time, the grouping must be done by sub-categories. Simply pass a list in the form of [category, sub-category] to x, leave barmode at "stack" and add a trace per response category (here: conversion and exit).

**Visualization 4: Dots plots and Heatmap**



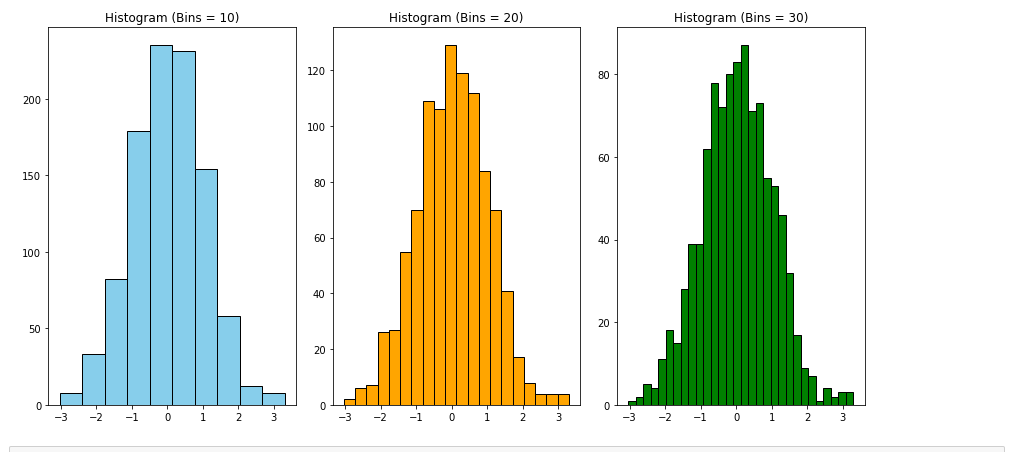
Using seaborn:The sns.stripplot() function is used to create a dot plot in Seaborn. The x parameter specifies the categories, the y parameter specifies the values, and color sets the color of the dots. The size parameter controls the size of the dots, and jitter adds a small amount of random noise to the x-axis position to prevent overlapping dots.



A heatmap can reveal which aesthetics concepts are strongly correlated or inversely correlated with each other. Suppose you have data that includes multiple aesthetics concepts and their correlations. You can create a heatmap to visualize the pairwise correlations between these concepts.

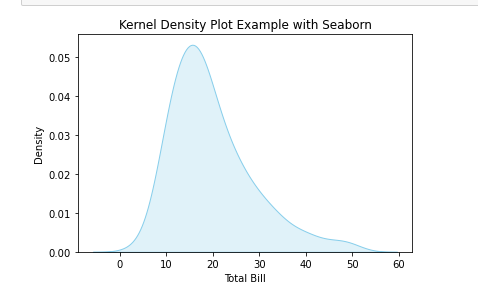
**Question 2:**

**Visualization 1: Histograms with different bin size**



In both examples, we use numpy to generate random data for matplotlib and load the "total\_bill" column from Seaborn's tips dataset for seaborn. We create subplots with different bin sizes using the bins parameter in the plt.hist() function for matplotlib and the sns.histplot() function for seaborn. The edgecolor parameter in matplotlib sets the color of the bin edges. plt.tight\_layout() is used to improve the layout when having multiple subplots.

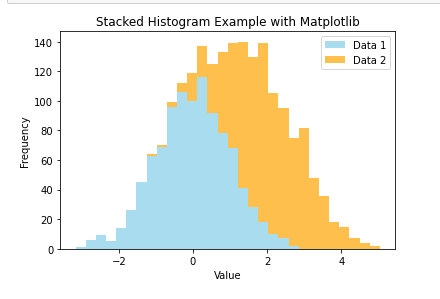
**Visualization 2: Kernal density**



Kernel Density Estimation (KDE) is a non-parametric way to estimate the probability density function of a random variable. In Python, you can create kernel density plots using seaborn or matplotlib. I'll provide examples for both.

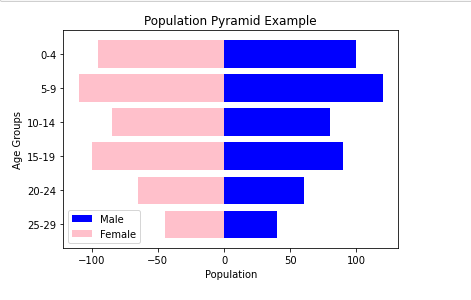
With matplotlib: The plt.hist() function is used to create a histogram for reference (using the same data).The density=True parameter normalizes the histogram to form a probability density. The sns.kdeplot() function is then used to overlay the kernel density plot on top of the histogram. The color parameter sets the color of the plot, and fill=True fills the area under the curve. Kernel density plots provide a smooth estimate of the probability density function and can be useful for visualizing the underlying distribution of your data. You can customize these plots further by adjusting parameters like color, linewidth, and others as needed.

**Visualization 3: Stacked Histogram**



With matplotlib: The plt.hist() function is used with the stacked=True parameter to create a stacked histogram. The color parameter specifies the colors of the stacked bars, and alpha controls the transparency. The label parameter is used for the legend.

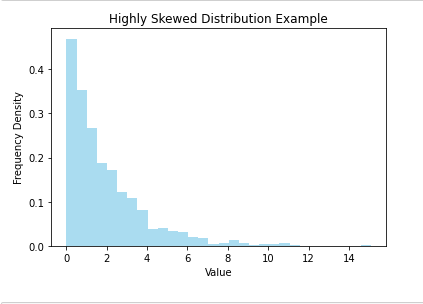
**Visualization 4: Pyramid graph**



A pyramid graph, also known as a population pyramid or age-sex pyramid, is a graphical representation of the distribution of a population by age and gender. It is often used to analyze demographic data and visualize the age and gender structure of a population. The code uses barh to create horizontal bar plots, with positive values for the male population (left side of the pyramid) and negative values for the female population (right side of the pyramid). invert\_yaxis() is used to invert the y-axis, placing the younger age groups at the bottom. align='center' ensures that bars for each age group are centered along the y-axis.legend() is used to display the legend.

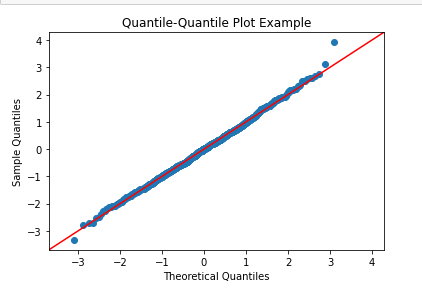
**Question 3:**

**Visualization 1: Empirical cumulative distribution functions**



The empirical cumulative distribution function (ECDF) is a non-parametric way to estimate the cumulative distribution function (CDF) of a random variable based on observed data. It provides a step function that jumps up by 1/*n* at each data point, where *n* is the total number of data points.

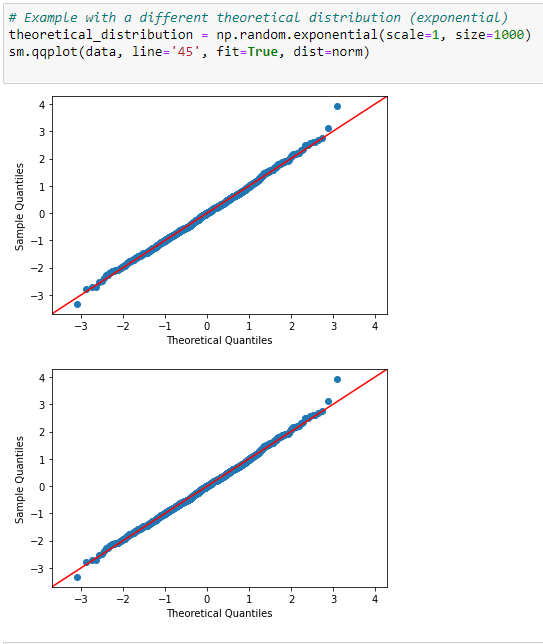
**Visualization 2: Highly skewed distributions**



Highly skewed distributions are characterized by a long tail on one side, indicating that a significant proportion of the data points are concentrated on one side of the distribution. Skewness is a measure of the asymmetry of a probability distribution. A positively skewed distribution has a tail on the right side, and a negatively skewed distribution has a tail on the left side. A positive skewness value indicates a right-skewed distribution, while a negative value indicates a left-skewed distribution. The magnitude of the skewness value gives you an idea of the extent of the skewness.

Remember that the choice of the distribution and its parameters depends on the nature of your data and the characteristics you want to capture. High skewness can impact the performance of certain statistical analyses, so it's essential to be aware of the distribution properties when working with skewed data.

**Visualization 3: Quantile-quantile plots**



The Q-Q plot is a useful tool for assessing the fit of your data to a theoretical distribution visually. If the points in the plot deviate significantly from the reference line, it may indicate departures from the assumed distribution. Quantile-Quantile (Q-Q) plots are used to visually assess whether a dataset follows a particular theoretical distribution. They compare the quantiles of the observed data against the quantiles of a specified theoretical distribution, such as the normal distribution. If the points in the Q-Q plot fall approximately along a straight line, it suggests that the data follows the theoretical distribution.

Top of Form

**Question 4:**

**Visualization 1: Error plots**

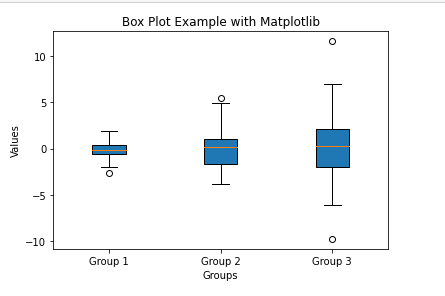


Error plots, also known as residual plots, are used to visually inspect the residuals of a statistical model. Residuals are the differences between the observed values and the predicted values from the model. Examining the residuals can help identify patterns, heteroscedasticity, and other issues in the model.

In the error plot:

* Points above the horizontal line at y=0 indicate overestimation by the model.
* Points below the line indicate underestimation.
* The horizontal line at y=0 serves as a reference for perfect predictions.

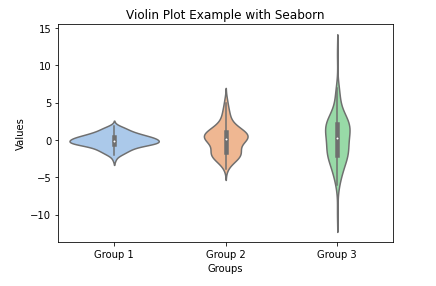
**Visualization 2: Box plot**



Box plots, also known as box-and-whisker plots, are a popular way to visualize the distribution and spread of a dataset. They display the median, quartiles, and potential outliers in a concise manner. Box plots include:

* The box represents the interquartile range (IQR) between the first (Q1) and third (Q3) quartiles.
* The line inside the box represents the median.
* Whiskers extend to the minimum and maximum values within 1.5 times the IQR.
* Points beyond the whiskers are considered potential outliers.

**Visualization 3: Violin Plot**



A violin plot is another type of data visualization that combines aspects of box plots and kernel density plots. It provides insights into the distribution of the data and includes information about the probability density of the distribution. Explanation:

* I generated sample data similar to the box plot example.
* sns.violinplot() in seaborn is used to create the violin plot. The palette parameter is used to set the color palette.

In a violin plot:

* The thick bar in the center represents the interquartile range (IQR) and the median.
* The thinner extensions, or "violins," on both sides show the probability density of the distribution.
* Outliers can also be visualized.