**Waffle Charts & Word Cloud**

Waffle charts are a visualization technique that represent categorical data in the form of square tiles or cells. These resemble a grid of equal-sized squares, with each square representing a specific value or category. The size or color of the squares indicate the magnitude or proportion of each category. Waffle charts effectively show the proportion or percentage of different categories within an overall composition. The grid-like structure of waffle charts makes it easy to understand and interpret data even for nontechnical audiences. Let's now explore the areas in which you can employ waffle charts. You can use waffle charts for market share analysis, demographic representation, project progress tracking, budget allocation, survey responses, election results, and product sales analysis.

Let's go through each use case. Visualize market share data, showing the proportion of companies or products within a specific market. Display demographic data, such as age groups or ethnicities within a population. Represent completion status of tasks or milestones, providing a visual overview of progress. Demonstrate allocation of budgetary resources across categories or departments within an organization.

Summarize survey responses, displaying the distribution of answers to multiple choice questions. Provide a clear visualization of voting outcomes, representing the distribution of votes for candidates or parties. And lastly, illustrate product sales by representing the distribution of sales across categories or regions.

Furthermore, by using the pywaffle library in Python, you can easily create visually appealing waffle charts to effectively communicate categorical data. You can import Waffle class from the pywaffle library and provide the values parameter with the data containing the four continents with the total immigrants from the Canada immigration data set. The rows and columns parameters determine the size of the chart grid. Lastly, we customize the chart by adding a title and positioning the legend.

Now that we understand waffle charts, let's learn about word cloud. Word cloud, also known as tag cloud or text cloud, is a popular data visualization method to visually present textual data in an engaging and informative manner. It presents a concise summary of the textual content by providing a visual overview of the most commonly used words within a given text or collection of documents. A word cloud is simply a depiction of the importance of different words in a body of text. It works in a simple way, a word appears bigger and bolder in the word cloud depending on how many times it appears in a source of textual data. Here is an example of a text on recruitment. This cloud shows us that phrases like recruitment, talent, candidates, and so on stand out in the document. Let's now explore the areas in which we can employ word cloud. These include social media analysis, customer feedback analysis, content analysis, market research, resume or job description analysis. Let's go through each use case. Word cloud helps extract and visualize popular topics or sentiments from social media conversations or hashtags. Summarize customer reviews or feedback to identify recurring themes or sentiments. It assists in analyzing textual content, such as articles, blogs, or research papers to uncover prevalent keywords or themes. Word cloud helps analyze survey responses, interviews, or focus group transcripts to extract key insights. It also highlights important skills or keywords in resumes or job descriptions to assess their relevance. In this video, you learned that waffle charts are a visualization technique that represent categorical data in the form of square tiles or cells. There are different areas in which you can use waffle charts. Word cloud is a popular data visualization method to visually present textual data in an engaging and informative manner. You can use word cloud in different areas to visually present textual data in an engaging and informative manner.

**Seaborn and Regression Plots**

Although Seaborn is another data visualization library, it's based on Matplotlib. Seaborn offers a range of built-in themes and color palettes that improve the visual appeal of your plots with minimal effort. Seaborn makes creating plots very efficient, therefore, with Seaborn, you can generate plots with code that is five times less than with Matplotlib. Seaborn integrates well with statistical libraries such as NumPy and SciPy, allowing you to easily combine statistical analysis with visualizations. It provides specialized plot types such as regression plots, distribution plots, and categorical plots, that are particularly useful for analyzing data and modeling relationships. While Pandas and Matplotlib are powerful tools for data manipulation and basic visualization, Seaborn complements them by providing a higher level interface for creating visually appealing and informative statistical graphics. Seaborn works well, especially when dealing with more complex visualizations and statistical analyses. Let's see how we can use Seaborn to create a statistical graphic. Let's look into regression plots. Let's say we have a data frame called df\_total, representing total immigration to Canada from 1980 to 2013. The data frame displays the year in one column and the corresponding total immigration in another. We want to create a scatter plot and a regression line to highlight any trends in the data. With Seaborn, you can do all this with one line of code. We first import Seaborn, and let's import it as SNS, then we call the Seaborn Regplot function. We tell it to use the data frame df\_total, and to plot the column year on the horizontal axis and the column total on the vertical axis. The output of this one line of code is a scatter plot with a regression line, and not just that, but also a 95% confidence interval. Seaborn's Regplot function also accepts additional parameters for any personal customization, so you can change the color, for example, using the color parameter. Let's go ahead and change the color to green.

Also, you can change the marker shape as well using the marker parameter. Let's go ahead and change the shape of our markers to a plus marker instead of the default circular marker. Let's try to plot some categorical data. In our Canada immigration data set, there are some categorical features such as country, region, and continent. Why not plot continents for their count in the data set? Using a single line of code, we can create a bar plot representing the count of records for each continent in the data using the counterplot function. The x parameter specifies the categorical variable to be plotted on the x-axis, the country, and the data parameter sets the data set to be used, df\_Canada. Though we haven't used all observations from the data set, it's evident from the plot that most of the observations in the data are from Africa. Let's try to plot the Bohr plot on the categorical data from a slice of the df\_Canada data set. Here we have plotted the continent by the total column of data. Seaborn has been grouped by the categorical variable continent and plotted the aggregated values of total, with confidence interval. You'll explore more in the lab session on Seaborn. In this video, you learned that Seaborn is a Data Visualization library based on Matplotlib. Seaborn was built primarily to provide a high-level interface for drawing statistical graphics. Scatter Plots and Regression Lines can be created with one line of code using Seaborn. Seaborn's Regplot function accepts additional parameters for personal customization.

**Introduction to Folium**

Folium is a powerful data visualization library in Python that was built primarily to help people visualize geospatial data. With Folium, you can create a map of any location in the world using latitude and longitude values. You can also create a map and superimpose markers and clusters on top of the map for interesting visualizations, you can also create maps of different styles, such as street level maps, stamen maps, and a couple of others, which we will look into in just a moment. Creating a world map with Folium is straightforward. First, you need to import Folium and then you call the map function. That is all. What's interesting about the maps created by Folium is that they are interactive, so you can zoom in and out after the map is rendered, which is a helpful feature. The default map style is open street map, which shows a straight view of an area when you are zoomed in and shows the borders of the world countries when you are zoomed out all the way. Now, let's create a world map centered around Canada. To do that, we pass in Canada's latitude and longitude values using the location parameter. With Folium, you can set the initial zoom level using the zoom\_start parameter. Use initial because you can easily change the zoom level after the map is rendered by zooming in or out, you can play with this parameter to determine the initial zoom level for different values. Let's set the zoom level for our map of Canada to four. The result will be a world map centered around Canada. Another feature of Folium is that you can create different map styles using the tiles parameter. Let's create a stamen toner map of Canada. This style is great for visualizing and exploring river meanders and coastal zones. Another style is stamen terrain. Let's create a map of Canada in stamen terrain. This style is excellent for visualizing hill shading and natural vegetation colors. In this video, you learned that Folium is a data visualization library in Python that helps people visualize geospatial data. With Folium, you can create maps of different styles, such as street level maps, stamen maps, and more. A feature of Folium is that you can create different map styles using the tiles parameter.

**Maps with Markers**

With Folium you can easily add markers on the map. Let's first render a world map centered around Canada. First, import Folium, then create the map object. Remember that the location parameter specifies the latitude and longitude coordinates of the center point of the map. The zoom\_start sets the initial zoom level of the map. In this case, zoom\_start = 4 provides a zoomed out view of Canada. You can display it by simply calling canada\_map. Markers play a vital role in enhancing interactivity and adding context to maps. They represent specific locations or points of interest, providing additional information when clicked. Markers are like signposts that guide us through the map, highlighting important elements. Ontario is a Canadian province that contains about 40% of the Canadian population. It is considered Canada's most populous province. Let's add a marker for Ontario province, one of the largest provinces in Canada to our map. Using the folium.Marker function, we specify the location parameter as 51.2538, -85.3232, representing the approximate coordinates for Ontario. Additionally, we set the pop-up parameter as Ontario to provide a label when the marker is clicked. The add\_to (canada\_map) method is called on the folium.Marker object to add the marker for Ontario to the canada\_map. This ensures that the marker is included as part of the map’s layers and will be displayed when the canada\_map is rendered or saved. Alternatively, we can create the marker using FeatureGroup. Let's go ahead and create a feature group named Ontario. Now, when a feature group is created, it is empty, so what's next is to start creating what is called children and adding them to the feature group. Let's create a child in the form of a red circular mark located at the center of the Ontario province. We specify the location of the child by passing in its latitude and longitude values. Once we're done adding children to the feature group, we add the feature group to the map. The result is a red circular mark superimposed on top of the map and added to the center of the province of Ontario. Now, it would be helpful if we could label this marker in order to let other people know what it represents. To do that, we use the marker function and the pop-up parameter to pass in the text we want to add to this marker. The result is our marker displays Ontario when clicked on. In the lab session, we will view a real-world example and explore the crime rate in San Francisco. We’ll create a map of San Francisco and superimposed thousands of these markers on top of the map. Also, how you can generate marker clusters to make your map look less congested. This module's lab session is interesting, so ensure you complete it. Displaying multiple markers on a map is essential when using larger maps. Suppose you want to show multiple markers on the map. How can you do that? It's simple. Create a list of all the locations, then pass this list to the marker function, adding an underscore to function through a loop and your map with multiple markers will be ready for display. When creating code to display multiple markers on a map, you can call the MarkerCluster object add an underscore to the function and pass your map here. It's the same as creating a list of locations. Then pass this list to the marker function through the loop. But this time, instead of map, use marker\_cluster to add the underscore to functions and your map with multiple markers will be displayed. The markers within the marker\_cluster will be intelligently grouped based on their proximity when the map is displayed. This clustering feature enhances the visual presentation by preventing overcrowding and ensuring a clear representation, primarily when numerous markers are close. In this video, you learned that with Folium, you can easily add markers on maps. The location parameter specifies the latitude and longitude coordinates of the center point of the map. Markers play a vital role in enhancing interactivity and adding context to maps. The folium.Marker function specifies location parameters. The pop-up parameter provides a label upon being clicked. Markers can be created using feature group.

**Choropleth Maps**

A choropleth map is a thematic map in which areas are shaded or patterned in proportion to the measurement of the statistical variable displayed on the map. Such as population density or per capita income. The higher the measurement, the darker the color. So the map to the left is a choropleth world map showing the infant mortality rate per 1000 births. The darker the color, the higher the infant mortality rate. According to the map, African countries have very high infant mortality rates, with some reporting a rate higher than 160 per 1000 births. Similarly, the map on the right is a choropleth map of the US showing population per square mile by state. Again, the darker the color, the higher the population. According to the map, states in the eastern part of the US tend to be more populous than states in the western part, with California being an exception. Folium is a Python library used for creating interactive maps and visualizations. It provides a simple and intuitive way to generate maps using data from various sources, including GeoJSON, Pandas DataFrames, and NumPy arrays. To create a choropleth map of a region of interest, folium requires a GeoJSON file that includes geospatial data of the region. For a choropleth map of the world, we would need a GeoJSON file that lists each country and any geospatial data to define its borders and boundaries. Here's an example of what a GeoJSON file would include about each country. The example here pertains to the country Brunei. As you can see, the file consists of its name, ID, geometry, shape, and coordinates that define its borders and boundaries. So let's see how we can create a choropleth map of the world showing immigration to Canada. Before creating the map's code, let's quickly recap our data set. Recall that each row represents a country and contains metadata about it, such as where it is located geographically and whether it's developing or developed. Each row also contains numerical figures for annual immigration from that country to Canada from 1980 to 2013, and let's name our data frame df\_canada. So now that we know our data is stored in the data frame df\_ canada, let's see how we can generate a choropleth map of the world showing immigration to Canada. We should be experts now in creating world maps with Folium. So let's go ahead and create a world map. But this time, let's use the Mapbox Bright Tileset. The result is a nice world map displaying the name of every country. Now, let's convert this map into a choropleth map. We first define a variable that points to our JSON file. Then we apply the choropleth function to our world map. We tell it to use the country total columns in our df\_canada data frame and the country names to look up the geospatial information about each country in the GeoJSON file. The result will be a choropleth map of Canada showing the intensity of immigration from different countries worldwide. We explore choropleth maps in more detail in the lab session, so be sure to complete this module's lab session. In this video, you learned that a choropleth map is a thematic map in which areas are shaded or patterned in proportion to the measurement of the statistical variable. When creating a choropleth map, Folium requires a GeoJSON file that includes geospatial data of the region. The Mapbox Bright Tileset displays the name of every country when used on a map.