**Overview of Data Visualization**

After watching this video, you'll be able to explain what data visualization is and why it's important. Describe the uses for data visualization. Explore the best practices for data visualization. Data visualization is the graphical representation of data and information. It involves the process of creating visual representations of data. Data visualization can take many different forms, from basic charts and graphs to more complex interactive dashboards, maps, and infographics. Basic charts and graphs are the simplest form of data visualization to represent numerical data. Interactive dashboards and maps are more complex forms of visualization to provide real-time information. We need data visualization for several reasons. Data visualization helps us to easily understand complex datasets that might be difficult to comprehend in their raw form. It can highlight patterns, trends, and relationships that might not be immediately apparent from looking at the data. Visualization enables us to communicate insights and findings to stakeholders in a more compelling and understandable way. It allows us to tell a story with data, making it easier for stakeholders to understand and act on the information presented. Visualization helps us identify trends and data patterns that might otherwise be difficult to see. We can gain new insights and make informed decisions by examining data over time or across different dimensions. Presenting data in a more easily understood and interpreted way enables us to identify opportunities, spot potential problems, and make more informed choices. Let's explore how data visualization can help us understand data. The New York Times, a well-known news publication, has used data visualization to help people understand the impact of the COVID-19 pandemic in the United States. The number of daily new reported cases and the number of deaths are shown through a line graph and the vaccinated population percentage is displayed using a bar graph. Airbnb uses smart pricing framework to assist hosts and determining the best possible prices for their listings, thereby increasing their profits using a bar graph, the chart displays the lead time distribution aggregated to a market level. Spotify uses Spotify Pie, a unique data visualization tool that enables users to create customized pie charts based on their listening habits on Spotify. The chart displays the distribution of different genres, artists, and tracks the user has listened to. What you see here is a choropleth map on Netflix's Insight Tool dashboard with other visualizations like line graphs and bar plots. These dynamic dashboards help them gain a deeper understanding of operations, identify areas for improvement, and quickly address any issues. Data visualization is an essential tool for anyone dealing with data. Companies can gain insights into market trends, financial performance, and customer behavior. Healthcare professionals can identify patterns in patient data and develop targeted treatment plants. Visualization helps analyze student performance and inform instructional decisions leading to better academic outcomes. Visualization helps governments make informed decisions and communicate data to the public. Visualization helps scientists and researchers analyze complex data, develop new insights, and share findings effectively. Visualization helps workers and the entertainment industry determined ratings and ultimately signals to the creators what the audience wants. Although data visualization has become an essential tool, not all visualizations are created equal. To create effective visualizations, it's important to follow best practices that can help ensure that the data is accurately represented and the message is clearly communicated. One key best practice is choosing the appropriate visualization type for the presented data. Different types of data require different visualization techniques. Choose the right one to communicate that data effectively. Keep the visualization simple and easy to read. A line chart shows trends over time. Well, a bar chart is ideal for comparing values. To ensure that the data is accurately represented, it's also important to label the axis clearly and provide context for the data being presented. Include a clear title, summarizing the main message and a legend explaining the meaning of any symbols or colors used. Including unnecessary graphics or information, can confuse the audience and distract them from the main message. Keep visualization focused on the main point and use only the necessary data and labels. Lastly, it's important to consider the audience when creating the visualization. Who will view the visualization? What do they need to know? The visualization should be tailored to their needs and understanding. What do you think about this plot or this pie chart? What is the data trying to convey? In these charts, there are a lot of variables and the information is more difficult to understand. This is an example of bad data visualization. This chart represents a team's board. The stacked bars depict the percentage of the work assigned to the individual users and their work progress. Colors are distinguishable and consistent. The legend shows the individual color's meanings. The chart provides more clarity with the values and percentages on top of each bar. Darkhorse Analytics is a company that was created out of a research lab at the University of Alberta in 2008. The company has done fascinating work on data visualization. Darkhorse Analytics specializes in quantitative consulting in several areas, including data visualization and geospacial analysis. Their approach when creating a visual revolves around the idea that less is more effective, more attractive, and more impactful. By following these best practices, data visualizations can become a powerful tool. Checkout the Darkhorse Analytics website for examples of good and bad plots. In this video, you learned that, data visualization is the process of presenting data in a visual format, such as charts, graphs, and maps to help people understand and analyze data easily. Data visualization has diverse use cases such as business, science, health care, and finance. It's important to follow best practices, such as selecting appropriate visualizations for the data being presented, choosing colors and fonts that are easy to read and interpret, and minimizing clutter.

**Types of Plots**

After watching this video, you'll be able to: explore different types of plots available for visualizing data. Identify the characteristics and appropriate use cases for each type of plot. As we know, data visualization represents data using visual formats like graphs, charts and maps. It effectively communicates information, trends and insights concisely, it also uncovers patterns, identify trends and simplifies complex information for easy understanding. Let's now explore the different types of plots, some of these include line plot, bar plot, scatter plot, box plot, histogram. A line plot, also known as a line chart, displays data as a series of data points connected by straight lines. Line plots display trends, such as stock market fluctuations or temperature changes over time, they compare datasets with a continuous independent variable like age or time. Line plots illustrate cause and effect relationships, such as sales revenue changes based on marketing budget. They also visualize continuous data, like height measurements over time. Line plots can be misleading if the scales on the axes are not carefully chosen to reflect the data accurately. The line plot shows a downfall in immigration trends in 1998, however, if you will correct the scale on the y-axis to start with zero, as in this plot, you'll see that the actual trend is not that alarming. Let's now explore the next plot. A bar plot, also known as a bar chart, displays data using rectangular bars, where the height or length of the bars represents the magnitude of the data. The bars can be oriented either vertically or horizontally. A vertically oriented bar chart is often referred to as a vertical bar chart or a column chart. If you're looking for an effective way to compare data, bar plots are ideal for comparing different categories or groups. They excel with discrete data, like comparing sales revenue by product. They show how different categories contribute to the whole and rankings, such as sales percentage or budget allocation. Bar plots can visualize data that you can easily rank, like displaying the bestselling books in the market. Their simplicity and interpretability, make them favored for data visualization. Inaccurate bar choices or axis scales, can lead to misleading plots. In this bar plot, the y-axis starting at six, makes the difference between the values in 1995 and 1998 seem larger than it actually is. However, when the y-axis starts at zero, the plot accurately represents the data without misleading the viewer. Now let's go through the next plot. A scatter plot, is a type of plot that presents values for two variables for a set of data using Cartesian coordinates. The data points are displayed as a collection of points, where one variable's value determines the position on the horizontal axis and the other variable's value determines the position on the vertical axis. When can you use a scatter plot? Let's explore some options. You can use it for examining the relationship between two continuous variables like temperature and energy consumption. Investigating patterns or trends in data, such as house prices versus size, detecting outliers or unusual observations such as outliers in test scores or abnormal stock behavior. Visualizing data with many observations to identify clusters or groups, and exploring complex data. Outliers significantly impact interpretation, requiring consideration of their inclusion or exclusion. On the screen, the data is plotted with and without outliers. With outliers, it shows two clusters, but removing outliers makes the remaining data more visible. Proper outlier handling, enhances accuracy and meaningful insights in scatter plots. Let us learn about the box plot. A box plot, also known as a box and whisker plot, is a type of plot that displays the distribution of a data set, along with key statistical measures. It consists of a box, representing the interquartile range, IQR, a line inside the box representing the median and lines whiskers extending from the box to indicate the range of the data, excluding outliers. Outliers may be represented as individual data points beyond the whiskers. Let's look at some scenarios where box plots come in handy. While comparing the distribution of a continuous variable across different categories or groups. For example, employees salaries across departments, examining spread and skewness of data set, visualizing quartiles and outliers, identifying and analyzing potential outliers within a data set. Visualizing summary statistics, median, quartiles, range in a concise and informative manner, comparing distributions of multiple variables in datasets side by side. Remember, box plots provide valuable information about outliers, such as their presence and extent. Ignoring or mishandling outliers, can distort the interpretation of the data and mask important insights, same as with scatter plots. The last plot we're going to learn about is the histogram. A histogram, is a graphical representation of the distribution of a data set, showing the frequency or relative frequency of values within specific intervals. It consists of bars, where the height represents the data count in each interval. Histograms offer valuable insights into data distribution, outliers, skewness and variability. They visually depict the shape of the data, whether it's symmetric skewed or bimodal. Skewness can be assessed by examining the histogram's shape. Histograms also showcase data variability, allowing you to observe concentrations, gaps and clusters that reveal patterns or subgroups. Apart from issues with scale and inadequate labeling, be careful while choosing the bins to create a histogram. We have generated a random data set and created three histograms, with three different binning options. Too few bins, five represented in green color, an optimal number of bins, 20 represented in blue and too many bins 50, the yellow one. By comparing these histograms, you can observe how the choice of binning affects the representation of the data distribution and results in a misleading representation of the data. Selecting too few or too many bins, can oversimplify or overcomplicate the distribution, respectively. In this video, you learned that, there are various types of plots commonly used in data visualization. Line plots capture trends and changes over time, allowing us to see patterns and fluctuations. Bar plots compare categories or groups, providing a visual comparison of their values. Scatter plots explore relationships between variables, helping us identify correlations or trends. Box plots display the distribution of data, showcasing the median, quartiles and outliers. Histograms illustrate the distribution of data within specific intervals, allowing us to understand its shape and concentration.

**Plot Libraries**

After watching this video, you'll be able to explore some popular plot libraries for data visualization, identify the features of plot libraries in Python. Data visualization is a vital tool for gaining insights, and effectively communicating complex information. In the world of data visualization, there are several plot libraries in Python that provide unique features and capabilities. Some of the popular libraries are Matplotlib, Pandas, Seaborn, Folium, Plotly, and PyWaffle. Each plot library has its own strengths and use cases. By harnessing the power of these plot libraries, you can unlock insights from your data and effectively communicate your findings. Let's start with Matplotlib. It's a general purpose plotting library that provides a wide variety of plots and customization options. It integrates well with other libraries and frameworks such as NumPy, Pandas, Seaborn, and Plotly. This interoperability allows you to combine the strengths of different libraries and create sophisticated and interactive visualizations. Let's explore some of Matplotlib's features. Now you can create line plots, scatter plots, bar charts, histograms, pie charts, box plots, and heat maps, to name a few. Matplotlib also allows you to customize colors, line styles, marker styles, access labels, titles, legends, and so on. Matplotlib's versatility, customization options, integration capabilities, and community support make it a powerful and reliable choice for data visualization tasks in Python. The next plot library is Pandas. Users primarily employ it for data manipulation, but it also offers plotting capabilities. Pandas plotting functions are built on top of Matplotlib, which means you can leverage the power and versatility of Matplotlib by combining it with Pandas data manipulation capabilities. Let's now check for some of its features. It integrates seamlessly with Pandas data structures, making it easy to create line plots, scatter plots, bar charts, histograms, and pie charts directly from your data frames. Pandas visualization capabilities are well suited for exploratory data analysis. Let's explore the third plot library. Like Pandas, Seaborne is also built upon the functionality of Matplotlib. It's a great choice for specialized statistical visualizations. It provides stylish and specialized plot types such as categorical plots, count plots, heat maps, violin plots, scatter plots, bar plots, and many more. Now let's dive into the features of Seaborn. Seaborne comes with various color palettes and styles to customize and enhance the aesthetics of the plot. It offers functions to combine multiple plots into grid layouts, which can be useful for comparing multiple variables or subgroups. Its integration with Pandas allows it to plot data directly from Pandas data frames or series. Seaborn's combination of statistical visualizations default, aesthetics integration with Pandas ease of use, and built in themes make it a popular choice among data scientists and analysts for exploring and communicating insights from data. The next plot library we're going to learn about is Folium. When it comes to geospatial data visualization, Folium is an excellent library. It allows you to create interactive and customizable maps. Whether it's choroplasmaps point maps or heat maps, foleum provides the tools to visually represent your geospatial data. Here is a key feature of Folium. Folium seamlessly integrates with popular data analysis libraries in Python, such as Pandas and NumPy. It's a popular choice for geospatial data visualization and analysis in Python. Let's now explore the fifth plot library. If you want to present your data in an interactive way, plotly is a great choice to explore. It offers highly interactive plots and dashboards. With Plotly, you can create line plots, scatter plots, bar charts, pie charts, 3D plots, and Choropleth maps, to name a few. Here are some of the key features of Plotly. Its Plotly dash framework allows you to build interactive dashboards with rich visualizations and controls. Since Plotly is web based, it enables the rendering and viewing of plots in web browsers. This makes it convenient for sharing visualizations online, embedding them in web applications or dashboards, and collaborating with others. The last plot library we're going to explore is PyWaffle. If you want to visualize categorical data using Waffle charts, PyWaffle is a simple yet effective library. With PyWaffle, you can create waffle charts, square pie charts, donut charts, and many more types of plots by providing a unique way to represent proportions. In this video, you learned that Matplotlib is a plotting library that offers a wide range of plotting capabilities. Pandas is a plotting library that provides integrated plotting functionalities for data analysis. Seaborne is a specialized library for statistical visualizations, offering attractive default aesthetics and color palettes. Foleyum is a Python library that allows you to create interactive and customizable maps. Plotly is an interactive and dynamic library for data visualization that supports a wide range of plot types and interactive features. PyWaffle enables you to visualize proportional representation using squares or rectangles.

**Introduction to Matplotlib**

After watching this video, you'll be able to explain what Matplotlib is and why it was created. Describe the uses for Matplotlib. Matplotlib is one of the most widely used data visualization libraries in Python. It was created by John Hunter, who was an American neurobiologist. John Hunter was part of a research team analyzing electro CTO cartography, ECoG signals, and use proprietary software for this task. However, the team had only one license and was taking turns using it. To overcome this limitation, John Hunter set out to replace the proprietary software with a Matlab based version that could be utilized by him and his teammates and extended by multiple investigators. As a result, Matplotlib was initially developed as an EEG and ECoG visualization tool. Just like Matlab, Matplotlib was equipped with a scripting interface for quick and easy generation of graphics represented by the plot. As for Matplotlib's architecture, it's composed of three main layers. The backend layer, the artist layer, where much of the heavy lifting happens, is the appropriate programming paradigm when writing a web application server, a UI application, or a script to be shared with other developers. The scripting layer is the appropriate layer for everyday purposes. It's considered a lighter scripting interface to simplify common tasks and for quick and easy generation of graphics and plots. The backend layer has three built-in abstract interface classes. FigureCanvas defines end encompasses the area on which the figure is drawn. Renderer, an instance of the renderer class knows how to draw on figure canvas. The event handles user inputs such as keyboard strokes and mouse clicks. The artist is the object that knows how to take the renderer and put ink on the Canvas. Everything you see in a Matplotlib figure is an artist instance. The title, the lines, the tick labels, the images, and so on, all correspond to an individual artist. There are two types of artist objects. Primitive: Line2D, Rectangle, Circle, and Text. Composite: Axis, Tick, Axes, and Figure. The top-level Matplotlib object that contains and manages all the elements in each graphic is the figure artist. The most important composite artist is the axis because it's where most of the Matplotlib API plotting methods are defined, including methods to create and manipulate the ticks, the axis lines, the grid, and the plot background. Each composite artists may contain other composite artists as well as primitive artists. A figure artists can have an axis artist, a rectangle, and a text artist. The artist layer is syntactically heavy. Programmers work directly with the backend and artists layers as they offer greater convenience while integrating Matplotlib with application servers. When it comes to the daily tasks of scientists involving data visualization or exploratory interactions, the scripting layer known as Pyplot works better. Matplotlib scripting layer is the Matplotlib.Pyplot interface, which automatically defines a Canvas and a figure artist instance and connects them. Let's generate a histogram of 10,000 random numbers with the scripting layer. First we import the Pyplot interface and you can see all the methods associated with creating the histogram and other art objects and manipulating them. Whether the hist method or showing the figure is part of the Pyplot interface. Notice the use of Numpy's random module with random.randn for creating random floats from the Pyplot hist method is called, hist creates a sequence of rectangle artists for each histogram bar and adds them to the axes container. To the hist function, we have passed the variable X containing an array of 10,000 random numbers and 100, which means creating 100 bins. The result is a histogram. It's simple to work in the scripting layer. The anatomy of a plot refers to the different components and elements that make up a visual representation of data. There's a reference on the official website of Matplotlib titled Anatomy of figure. This image provides a complete guide to understanding what a plot may include based on your requirements. The main component is the window or Canvas containing the plot or subplots. Matplotlib figure as a Canvas and axis represents an individual plot within a figure, you must first create two-axis parts to make two plots on one figure, the axis provides scales and tick marks for plotting the data. The actual data being plotted is represented as points or markers on the plot like the axis. A good plot must have a title to provide a summary or explanation of the plot. Likewise, labels describe data being plotted on each axis. You may like to include a legend to explain the meaning of different elements or data series and applaud a grid to help visually aligned data points and aid and reading values from the plot and annotations to provide supplemental information or explanations about specific data points are regions in the plot. You can choose symbols for individual data point, colors, and styles. This link will take you to a chapter written by the creators of Matplotlib. If you're interested in learning more about the history of Matplotlib and its architecture, then this is a recommended reading. In this video, you learned that Matplotlib is one of the most widely used data visualization libraries in Python. Matplotlib was initially developed as an EEG, ECoG visualization tool. Matplotlib architecture is composed of three main layers, the backend layer, the artist layer, and the scripting layer. The anatomy of a plot refers to the different components and elements that make up a visual representation of data.

**Basic Plotting with Matplotlib**

After watching this video, you'll be able to explore how to use Matplotlib to create plots by employing Jupyter notebook. Create conventional visualization tools using the Plot function. Matplotlib is a Well established data visualization library that can be integrated in different environments such as Python scripts, Python and IPython shells, web application servers and In Graphical User Interface toolkits. The Jupyter notebook is also one of them. The Jupyter notebook is an open source web application that allows you to create and share documents that contain live code, visualizations and some explanatory text as well. Jupyter has some specialized support for Matplotlib, so if you start a Jupyter notebook, all you have to do is import Matplotlib and you're ready to go. We will now learn how to create almost all of the visualization tools using the Scripting interface. As we proceed in the course, you will appreciate the power of this interface when you find out that you can literally create almost all of the conventional visualization tools, such as histograms, bar charts, boxplots and many others using one function only, the Plot function. Let's start with an example. Let's first import the Scripting interface as plt and let's plot a circular mark at the position 5-5. Notice how the plot was generated within the browser and not in a separate window. If the plot gets generated in the new popup window, then you can enforce generating plots into the browser using what's called the magic function, quote, %matplotlib, end quote, and you pass in inline as the back end. Matplotlib has a number of different backends available. One limitation of this backend is that you cannot modify a figure once it's rendered. So after rendering the figure, there is no way for us to add, for example, a figure title or labels to its axis. We will need to generate a new plot and add a title and the axis labels before calling the show function. A backend that overcomes this limitation is the notebook backend. With the notebook backend in place, if a plt function is called, it applies them to the active figure if it exists. If a figure does not exist, it renders a new figure. So when we call the plt.plot function to plot a circular mark at position 5-5, the backend checks if an active figure exists. Since no active figure exists, it generates a figure and adds a circular mark to position 5-5. And what is beautiful about this backend is that now we can easily add a title or labels to the axis after the plot was rendered without the need of regenerating the figure. Finally, another thing that is great about Matplotlib is that Pandas also has a built in implementation of it. Therefore, plotting in Pandas is as simple as calling the Plot function on a given Pandas series or Pandas data frame. So say we have a data frame of the number of immigrants to Canada from India and China for the years 1980 to 1996. And we're interested in generating a line plot of this data. All we have to do is call the plot, the function on this data frame, which we have called India\_China underscore\_df and pass in kind = line. And there you have it, a line plot of the data in the data frame. Plotting a histogram of the data is not any different. So say we would like to plot a histogram of the India column in our data frame. All we have to do is call the plot function on that column and pass in kind as hist for histogram. And there you have it a histogram of the number of Indian immigrants to Canada from 1980 to 1996. In this video, you learned that Matplotlib is a well established data visualization library that can be integrated in different environments. Jupyter Notebook is an open source web application that allows you to create and share documents. Matplotlib has a number of different backends available. You can easily include the label and title to your plot with plt.

**Dataset on Immigration to Canada**

After watching this video, you'll be able to understand the data set to be used in this course for data visualization. Import data with pandas as a DataFrame in your program. Process data to make it suitable for plotting. We will use the Population Division of the United Nations compiled immigration data pertaining to 45 countries, where for each country, data corresponding to the total number of immigrants from all world countries are reported. In addition to other metadata pertaining to the immigrants countries of origin, in this course, we'll focus on Canada and work primarily with the data set involving immigration to the Great White North. Here is a snapshot of the UN data on immigration to Canada in the form of an Excel file. The first 20 rows contain textual data about the UN department and other information. Row 21 contains the labels of the columns. Following that, each row represents a country along with metadata about the country, such as the continent it resides in, what region it belongs to, and whether the region is developing or developed. Then we have the total number of immigrants from that country for the years 1980 all the way to 2013. Throughout this course, we will be using pandas for any analysis of the data before creating visualization tools. So in order to start creating different types of plots of the data, whether for exploratory analysis or for presentation, we will need to import the data into a pandas DataFrame. To do that, we will need to import the pandas library, as well as the openpyxl library, which is required to extract data from Excel spreadsheets files. Then we will call the pandas function, read excel. To read the data into a pandas DataFrame, notice how we're skipping the first 20 rows to read only the data corresponding to each country. If you want to confirm that you have imported your data correctly, you can always use the head function to display the first five rows in the DataFrame. Now, let's process the data frame so that the country name becomes the index of each row. This should make querying specific countries easier. Also, let's add an extra column which represents the total immigration for each country from 1980 to 2013. So, for Afghanistan it's 58,639, and for Albania it's 15,699, and so on. Now let's name our DataFrame df\_canada. In this video, you learned that, the population division of the United Nations compiled immigration data pertaining to 45 countries. The UN data on immigration to Canada shows data related to the number of people who migrated. In order to start creating different types of plots of the data, you will need to import the data into a Pandas data frame.

**Line Plots**

After watching this video, you'll be able to describe line plot and its function. Determine when to use a line plot, create a line plot from data in the dataset. What is a line plot? As its name suggests, it's a plot that displays information as a series of data points connected by straight lines. It is one of the most basic types of charts and is common in many fields, not just data science. Let's identify when to use a line plot. Line plots are useful for visualizing trends and changes over time, making them a popular choice for time-series data, such as changes in stock prices, website traffic, or temperature fluctuations. You can also use a line plot to show relationships between two variables,they can also be used to compare multiple data series on one chart. Line plots can also effectively highlight sudden changes or anomalies in data. As an example, from our dataset, we can generate a line plot to see the trend of immigrants from Haiti to Canada. Based on this line plot, we see that there is a spike of immigration from Haiti to Canada in 2010. We can then research for justifications of obvious anomalies or changes from a quick Google search for major events in Haiti in 2010 one would easily learn about the tragic earthquake that took place in 2010. Therefore, this influx of immigration to Canada was mainly due to that tragic earthquake. Now, how can we generate this line plot, as we briefly mentioned in an earlier video with matplotlib, all we have to do is call the plot function on the pandas data frame or series containing the data of interest. Before we go over each code. To do that, let's do a quick recap of our data set. Each row represents a country and contains data corresponding to the status of the country in terms of where it is located geographically and whether it is developing or developed. Each row contains numerical figures of annual immigration from that country to Canada, 1980-2013. Now let's process the DataFrame so that the country name becomes the index of each row. This should make querying specific countries easier. Also, let's add an extra column which represents the total immigration for each country, from 1980-2013. For Afghanistan, it's 58,639, for Albania, it's 15,699, and so on. Now, let's name our DataFrame, df\_canada. Now that we know how our data is stored in the DataFrame df\_canada. Let's generate the line plot corresponding to immigration from Haiti. First, we import matplotlib as MPL and its scripting interface as PLT. Then we call the plot function on the row corresponding to Haiti, and we specify kind equals line to generate a line plot. Then to complete the figure, we give it a title and label its axis. Finally, we then call the show function to display the figure. Note that this is the code to generate the line plot using the magic function %matplotlib with the inline backend. There you have it. A line plot that depicts immigration from Haiti to Canada, from1980-2013. In this video, you learned that a line plot is a plot in the form of a series of data points connected by straight-line segments. Line plot is one of the most basic types of chart and is common in many fields. You can generate a line plot by assigning line to kind parameter in the plot() function.