Lesson Summary - week1

Congratulations! You have completed this lesson. At this point in the course, you know:

- Each line in a dataset is a row, and commas separate the values.
- To understand the data, you must analyze the attributes for each column of data.
- Python libraries are collections of functions and methods that facilitate various functionalities without writing code from scratch and are categorized into Scientific Computing, Data Visualization, and Machine Learning Algorithms.
- Many data science libraries are interconnected; for instance, Scikit-learn is built on top of NumPy, SciPy, and Matplotlib.
- The data format and the file path are two key factors for reading data with Pandas.
- The **read_CSV** method in Pandas can read files in CSV format into a Pandas DataFrame.
- Pandas has unique data types like object, float, Int, and datetime.
- Use the **dtype** method to check each column's data type; misclassified data types might need manual correction.
- Knowing the correct data types helps apply appropriate Python functions to specific columns.
- Using **Statistical Summary** with **describe()** provides count, mean, standard deviation, min, max, and quartile ranges for numerical columns.
- You can also use **include='all'** as an argument to get summaries for object-type columns.
- The statistical summary helps identify potential issues like outliers needing further attention.
- Using the **info() Method** gives an overview of the top and bottom 30 rows of the DataFrame, useful for quick visual inspection.
- Some statistical metrics may return "NaN," indicating missing values, and the program can't calculate statistics for that specific data type.
- Python can connect to databases through specialized code, often written in Jupyter notebooks.
- SQL Application Programming Interfaces (APIs) and Python DB APIs (most often used) facilitate the interaction between Python and the DBMS.
- **SQL APIs** connect to DBMS with one or more API calls, build SQL statements as a text string, and use API calls to send SQL statements to the DBMS and retrieve results and statuses.
- **DB-API**, Python's standard for interacting with relational databases, uses **connection objects** to establish and manage database connections and **cursor objects** to run queries and scroll through the results.
- Connection Object methods include the cursor(), commit(), rollback(), and close() commands.

- You can import the database module, use the **Connect API** to open a connection, and then create a cursor object to run queries and fetch results.
- Remember to close the database connection to free up resources.