

DATA WRANGLING

UNIT - 1

INTRODUCTION TO DATA WRANGLING

INTRODUCTION

Data Wrangling is referred to as *data munging*. It is the process of transforming and mapping data from one "raw" data form into another format to make it more appropriate and valuable for various downstream purposes such as analytics. The goal of data wrangling is to assure quality and useful data. Data analysts typically spend the majority of their time in the process of data wrangling compared to the actual analysis of the data.

The process of data wrangling may include further munging, data visualization, data aggregation, training a statistical model, and many other potential uses. Data wrangling typically follows a set of general steps, which begin with extracting the raw data from the data source, "munging" the raw data (e.g., sorting) or parsing the data into predefined data structures, and finally depositing the resulting content into a data sink for storage and future use.

Video link

<https://youtu.be/sJzrRN7voE?si=Z2B6SR3cB1rpPYLe>

Importance of Data Wrangling

Data wrangling software has become an indispensable part of data processing. The primary importance of using data wrangling tools can be described as follows:

1. Making raw data usable. Accurately wrangled data guarantees that quality data is entered into the downstream analysis.
2. Getting all data from various sources into a centralized location so it can be used.
3. Piecing together raw data according to the required format and understanding the business context of data.

4. Automated data integration tools are used as data wrangling techniques that clean and convert source data into a standard format that can be used repeatedly according to end requirements. Businesses use this standardized data to perform crucial, cross-data set analytics.
5. Cleansing the data from the noise or flawed, missing elements.
6. Data wrangling acts as a preparation stage for the **data mining process**, which involves gathering data and making sense of it.
7. Helping business users make concrete, timely decisions.

The Data Wrangling Process

Data wrangling involves several key steps:

- **Data Collection:** Gathering data from various sources, such as databases, spreadsheets, and APIs.
- **Data Cleaning:** Identifying and addressing missing values, duplicates, and inconsistencies.
- **Data Transformation:** Restructuring data, encoding categorical variables, and normalizing data.
- **Data Enrichment:** Adding additional relevant data or features to enhance analysis.
- **Data Validation:** Ensuring data quality and consistency before proceeding with analysis.



Common Data Wrangling Challenges

1. Data Inconsistency
2. Missing Data
3. Duplicate Records
4. Data Integration
5. Scalability

- 1. Data Inconsistency:** Data inconsistency is one of the most common issues faced during data wrangling. This occurs when data from different sources or even within the same source varies in format, structure, or meaning. For instance, dates might be formatted differently, or addresses might follow different conventions.
- 2. Missing Data:** Missing data can severely impact the quality of analysis. It can arise from various sources, including incomplete data entry, system errors, or data corruption. Handling missing data appropriately is crucial to ensure the accuracy of any subsequent analysis.
- 3. Duplicate Records:** Duplicate records can skew analysis results, leading to inaccurate insights and decisions. Identifying and eliminating duplicates is a key step in the data wrangling process.
- 4. Data Integration:** Integrating data from multiple sources into a single cohesive data set can be complex. Each source may have its own schema, format, and conventions, making it challenging to merge the data without losing valuable information.
- 5. Scalability:** As datasets grow larger, the process of data wrangling becomes more time-consuming and resource-intensive. Efficiently wrangling large datasets requires robust tools and processes that can scale accordingly.

Data Acquisition

There are four methods of acquiring data:

- Collecting new data;
- Converting/transforming legacy data;
- Sharing/exchanging data;
- Purchasing data.

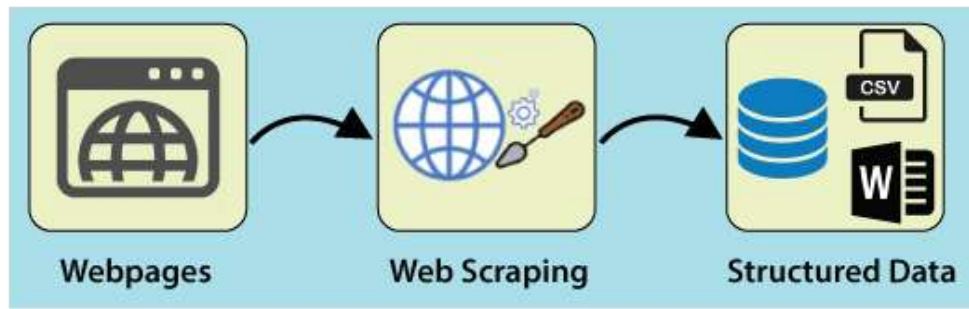
This includes automated collection (e.g., of sensor-derived data), the manual recording of empirical observations, and obtaining existing data from other sources.

Web Scraping

Web Scraping is a technique to extract a large amount of data from several websites. The term "**scraping**" refers to obtaining the information from another source (webpages) and saving it into a local file.

For example: Suppose you are working on a project called "**Phone comparing website**," where you require the price of mobile phones, ratings, and model names to make comparisons between the different mobile phones. If you collect these details by checking various sites, it will take much time.

In that case, web scrapping plays an important role where by writing a few lines of code you can get the desired results.



- Web Scrapping extracts the data from websites in the unstructured format. It helps to collect these unstructured data and convert it in a structured form.
- Startups prefer web scrapping because it is a cheap and effective way to get a large amount of data without any partnership with the data selling company.

Accessing Databases

Accessing databases in data acquisition techniques involves retrieving, processing, and managing data stored in various database systems. This process is essential for gathering and analyzing data to inform decision-making, research, and other applications.

Here's an overview of the key concepts and techniques involved:

1. Understanding Databases

➤ Types of Databases:

- **Relational Databases:** Organize data into tables (e.g., MySQL, PostgreSQL, Oracle). Data is accessed using Structured Query Language (SQL).
- **NoSQL Databases:** Use various data models (e.g., document-based like MongoDB, key-value stores like Redis, graph databases like Neo4j). They provide flexibility for handling unstructured or semi-structured data.
- **Data Warehouses:** Optimized for analytical queries, integrating data from multiple sources (e.g., Amazon Redshift, Google BigQuery).

2. Connection to Databases

- **Database Drivers:** Software components that enable applications to connect to a database. Common drivers include JDBC for Java applications and ODBC for various programming languages.
- **Connection Strings:** Strings containing information about the data source, including the database type, server address, database name, user credentials, and any required options (e.g., timeout settings).

3. Data Retrieval Techniques

- **SQL Queries:** Used in relational databases to fetch, insert, update, and delete data. Basic operations include:
 - **SELECT:** Retrieve specific columns from tables.
 - **JOIN:** Combine rows from two or more tables based on related columns.
 - **WHERE:** Filter results based on specific conditions.
- **API Access:** Many modern applications provide RESTful or GraphQL APIs to access data programmatically. This is common in NoSQL databases and cloud services.
- **ORM (Object-Relational Mapping):** Tools (like SQLAlchemy for Python or Entity Framework for .NET) that allow developers to interact with the database using high-level programming languages instead of writing raw SQL.

4. Data Processing and Management

- **ETL (Extract, Transform, Load):** A process that involves extracting data from various sources, transforming it into a suitable format, and loading it into a destination database or data warehouse.
- **Data Cleaning:** Ensuring that the data is accurate, complete, and free from duplicates or inconsistencies.
- **Data Aggregation:** Summarizing data to provide insights, often used in analytics. Techniques include grouping data and performing calculations (e.g., averages, counts).

5. Data Security and Compliance

- **Authentication and Authorization:** Ensuring that only authorized users can access and manipulate data. This can involve user roles, permissions, and secure connection protocols.
- **Data Encryption:** Protecting sensitive data both in transit (while being transmitted) and at rest (when stored).
- **Compliance:** Adhering to regulations regarding data privacy and protection (e.g., GDPR, HIPAA), which often dictate how data must be accessed and managed.

6. Tools and Technologies

- **Database Management Systems (DBMS):** Software that provides tools for creating, managing, and interacting with databases (e.g., MySQL, MongoDB, Microsoft SQL Server).
- **Data Visualization Tools:** Applications that help visualize data (e.g., Tableau, Power BI) often require database connections to retrieve the necessary data.
- **Scripting and Automation:** Using scripts (Python, R, etc.) to automate data retrieval and processing tasks, enabling more efficient data acquisition.

File input/output (I/O)

File input/output (I/O) in data acquisition techniques involves the processes of reading data from and writing data to files. This is essential for collecting, storing, and managing data in various formats.

Here's a brief overview:

1. File Formats

- **Text Files:** Common formats like CSV and TXT for storing readable data.
- **Binary Files:** Compact, non-readable formats that store data efficiently (e.g., images).
- **Structured Formats:** JSON and XML for hierarchical data representation, often used for data interchange.
- **Excel Files:** Widely used in business settings for data manipulation.

2. File Input Techniques

- **Reading Data:** Involves opening a file and loading its contents for processing.
 - **Line-by-Line Reading:** Efficient for large files to minimize memory usage.
 - **Buffered Reading:** Reads chunks of data at a time for better performance.
 - **Libraries:** Tools like Pandas (Python) simplify reading complex formats.

3. File Output Techniques

- **Writing Data:** Saving processed data to files.
 - **Appending Data:** Adding new information without overwriting existing data.
 - **Creating New Files:** Saving data in the same or different formats.
 - **Exporting Data:** Converting data to formats for use in other applications.

4. Data Processing Steps

- **Parsing:** Transforming raw file data into a structured format.
- **Transformation:** Cleaning and modifying data before writing it to a file.
- **Validation:** Ensuring data integrity before processing.

5. Error Handling and Performance

- **Error Handling:** Managing potential I/O errors (e.g., file not found) using exception handling.
- **Performance:** Considerations include file size and I/O speed. Techniques like streaming and chunking can optimize performance for large datasets.

6. Use Cases

- **Data Collection:** Gathering data from sources (e.g., sensors) and saving it for analysis.
- **Data Sharing:** Exporting data for sharing with stakeholders.
- **Backup and Archiving:** Saving data to prevent loss.

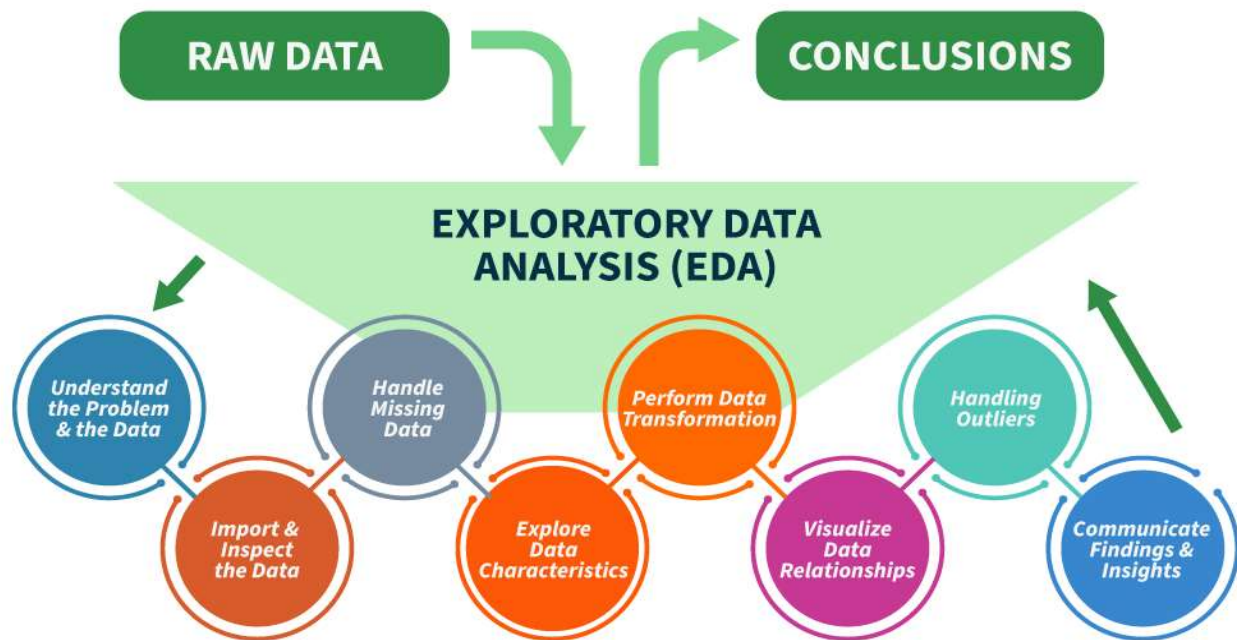
File I/O is a critical aspect of data acquisition that enables efficient data management through various formats and techniques, facilitating the analysis and sharing of information.

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a critical step in the data analysis process that involves examining and visualizing datasets to understand their underlying structures and characteristics. EDA is particularly useful for identifying data quality issues, which can significantly impact the accuracy and reliability of any subsequent analysis.

Here's a detailed explanation of EDA, focusing on its role in understanding data quality issues:

Steps for Performing Exploratory Data Analysis



1. Purpose of EDA

- **Understand Data:** Gain insights into the structure, patterns, and relationships within the data.
- **Identify Quality Issues:** Detect problems such as missing values, outliers, and inconsistencies.

2. Key Components of EDA

- **Summary Statistics:** Calculating measures like mean, median, mode, and standard deviation to understand central tendencies and variability.
- **Data Visualization:** Creating graphs and plots (like histograms, box plots, and scatter plots) to visualize distributions and relationships in the data.

3. Identifying Data Quality Issues

- **Missing Values:** Checking for and analyzing gaps in data, which can affect analysis.
- **Outliers:** Identifying unusual data points that may skew results.
- **Data Types:** Ensuring that data types are correct (e.g., numbers stored as strings), which can lead to analysis errors.
- **Duplicates:** Finding and addressing repeated entries that can distort results.

4. Techniques Used in EDA

- **Histograms:** Visualizing the distribution of numerical data to identify patterns or anomalies.
- **Box Plots:** Detecting outliers and understanding data spread.
- **Scatter Plots:** Exploring relationships between two variables.
- **Correlation Matrices:** Assessing how variables are related to each other.

5. Benefits of EDA

- **Informed Decisions:** Understanding data quality allows for better decision-making in data cleaning and preprocessing.
- **Enhanced Analysis:** Identifying issues early helps improve the accuracy and reliability of subsequent analyses.

In summary, EDA is a crucial step in the data analysis process that helps identify and understand data quality issues, ensuring the integrity and reliability of the data before conducting further analyses or drawing conclusions.

Data cleaning techniques:

- **Data cleansing or data cleaning** is the process of detecting and correcting (or removing) corrupt or inaccurate records from a record set, table, or database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing, modifying, or deleting the dirty or coarse data.
- Data cleansing may be performed interactively with data wrangling tools, or as batch processing through scripting.

- After cleansing, a data set should be consistent with other similar data sets in the system. The inconsistencies detected or removed may have been originally caused by user entry errors, by corruption in transmission or storage, or by different data dictionary definitions of similar entities in different stores.
- Data cleaning differs from data validation in that validation almost invariably means data is rejected from the system at entry and is performed at the time of entry, rather than on batches of data.

Here are some common data cleaning techniques explained in simple terms, along with examples for better understanding:

1. Handling Missing Values

What It Is: Missing values are gaps in your data where information is not available.

Techniques:

- **Removal:** You can delete rows or columns with missing values if they are not too many.
 - **Example:** If you have a dataset of 1,000 entries and only 5 have missing ages, you might choose to remove those 5 rows.
- **Imputation:** Fill in the missing values with reasonable estimates.
 - **Example:** If the average age in a dataset is 30 and one entry is missing age, you could fill that in with 30 (mean imputation).

2. Removing Duplicates

What It Is: Sometimes, the same entry can appear more than once in a dataset.

- **Technique:**
- **Identifying and Dropping Duplicates:** Find and remove these duplicate entries.
 - **Example:** If your dataset contains two identical entries for a customer, you would keep one and remove the other to ensure each customer is counted only once.

3. Standardizing Data Formats

What It Is: Ensuring that all data entries follow a consistent format.

Techniques:

- **Consistent Formatting:** Make sure all data is in the same style.
 - **Example:** If you have dates in different formats (e.g., "MM/DD/YYYY" and "YYYY-MM-DD"), you can convert them all to "YYYY-MM-DD" for uniformity.

4. Correcting Inconsistencies

What It Is: Fixing errors or variations in data entries.

Technique:

- **Typographical Errors:** Correct common misspellings or variations in names.
 - **Example:** If "USA" and "United States" are used interchangeably, choose one term and change all instances to that term for consistency.

5. Removing Outliers

What It Is: Outliers are data points that are much higher or lower than the rest of the data and can skew results.

Technique:

- **Identifying Outliers:** Use statistical methods to find them.
 - **Example:** If most ages in your dataset range from 20 to 60, but one entry is 200, that entry is an outlier. You might choose to investigate or remove it.

6. Data Validation

What It Is: Ensuring that the data follows certain rules.

Techniques:

- **Range Checks:** Check if data falls within expected limits.
 - **Example:** If you have a field for “age,” you can ensure that no ages are negative or unrealistically high (like 150 years).

7. Data Transformation

What It Is: Changing data into a different format or structure.

Techniques:

- **Binning:** Grouping continuous data into categories.
 - **Example:** If you have ages ranging from 1 to 100, you could create bins like "0-18", "19-35", "36-50", etc., to simplify analysis.

8. Data Integration

What It Is: Combining data from multiple sources while keeping it clean.

Techniques:

- **Combining Datasets:** When you merge datasets, make sure to eliminate duplicates and resolve conflicts.
 - **Example:** If you have one dataset with customer information and another with their orders, you would merge them and remove any duplicate customer entries.

9. Documentation and Audit Trails

What It Is: Keeping a record of all cleaning steps taken.

Technique:


- **Documenting Changes:** Write down what changes were made and why.
 - **Example:** If you removed certain duplicates, you might note in your records that "Removed duplicate entries for customer ID 123."

Data cleaning techniques are vital for preparing your data for analysis. By handling missing values, removing duplicates, standardizing formats, correcting inconsistencies, and more, you can ensure your data is accurate and reliable. This leads to better insights and decisions based on the data.

1. Handling Missing Values

Example:

python

 Copy code

```
import pandas as pd

# Sample DataFrame
data = {
    'Name': ['Alice', 'Bob', 'Charlie', None],
    'Age': [25, None, 30, 22],
    'City': ['New York', 'Los Angeles', None, 'Chicago']
}
df = pd.DataFrame(data)


# Remove rows with any missing values
df_cleaned = df.dropna()
print("After removing missing values:\n", df_cleaned)

# Fill missing values with the mean for the Age column
df['Age'].fillna(df['Age'].mean(), inplace=True)
print("\nAfter filling missing Age values:\n", df)
```

2. Removing Duplicates

Example:

python

 Copy code


```
# Sample DataFrame with duplicates
data = {
    'Name': ['Alice', 'Bob', 'Bob', 'Charlie'],
    'Age': [25, 30, 30, 22]
}
df = pd.DataFrame(data)

# Remove duplicate rows
df_cleaned = df.drop_duplicates()
print("After removing duplicates:\n", df_cleaned)
```

3. Removing Outliers

Example:

python

 Copy code

```
# Sample DataFrame with outliers
data = {
    'Age': [25, 30, 29, 200, 22] # 200 is an outlier
}
df = pd.DataFrame(data)

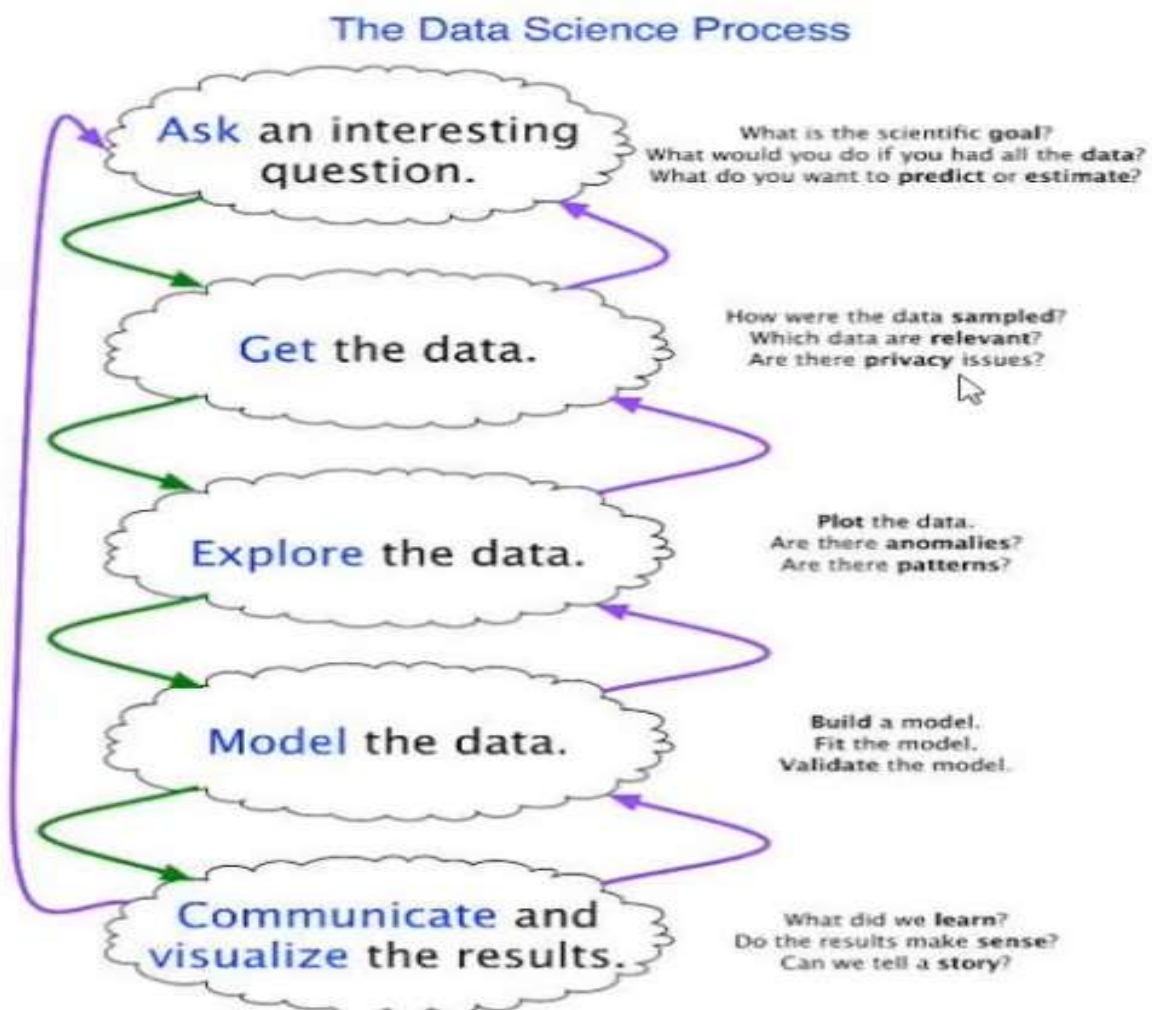
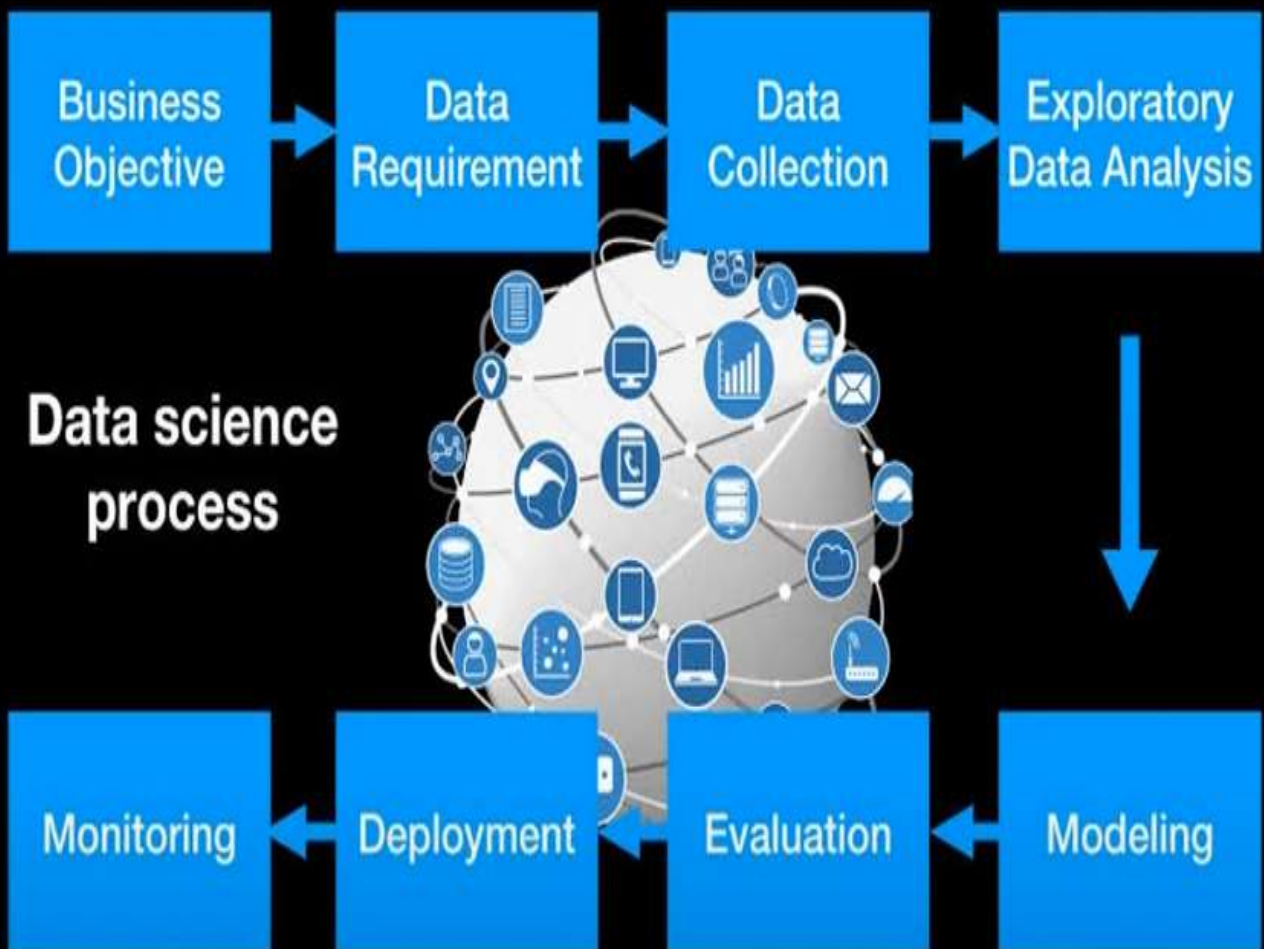
# Define a threshold for outliers (e.g., 1.5 times the IQR)
Q1 = df['Age'].quantile(0.25)
Q3 = df['Age'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Remove outliers
df_cleaned = df[(df['Age'] >= lower_bound) & (df['Age'] <= upper_bound)]
print("After removing outliers:\n", df_cleaned)
```


Exploratory Data Analysis (EDA)

IMPORTANCE OF EDA

- Identifying the most important variables/features in your dataset.
- Testing a hypothesis or checking assumptions related to the dataset.
- To check the quality of data for further processing and cleaning.
- Deliver data-driven insights to business stakeholders.
- Verify expected relationships actually exist in the data.
- To find unexpected structure or insights in the data.



Two Categories of Data

- **Structured Data types**

Example: csv file, excel file, database file

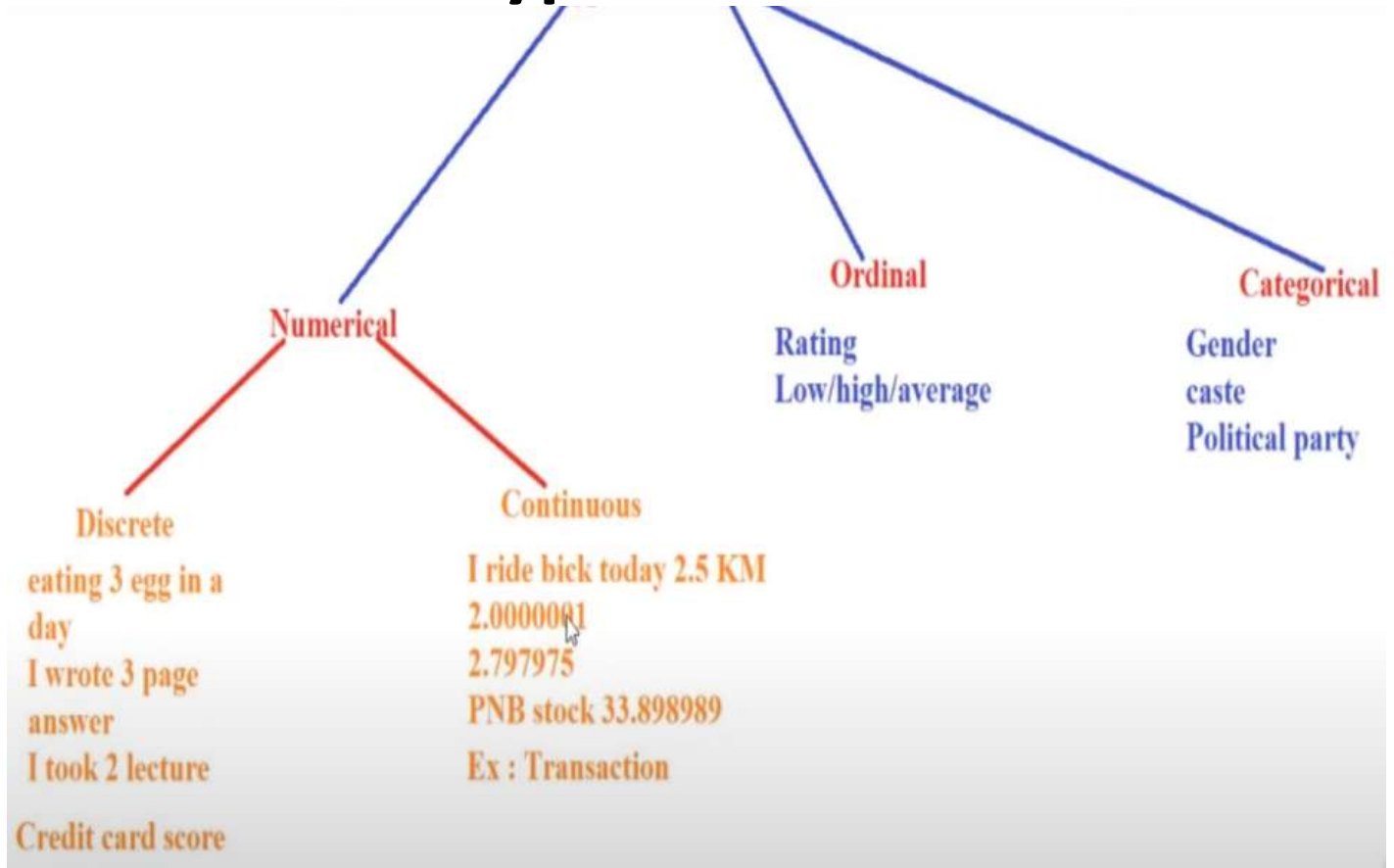
- **Unstructured Data types**

Examples: text, Images, videos, audio,

Major Types of Data

- Numerical
- Categorical
- Ordinal

Types of Data



Numerical

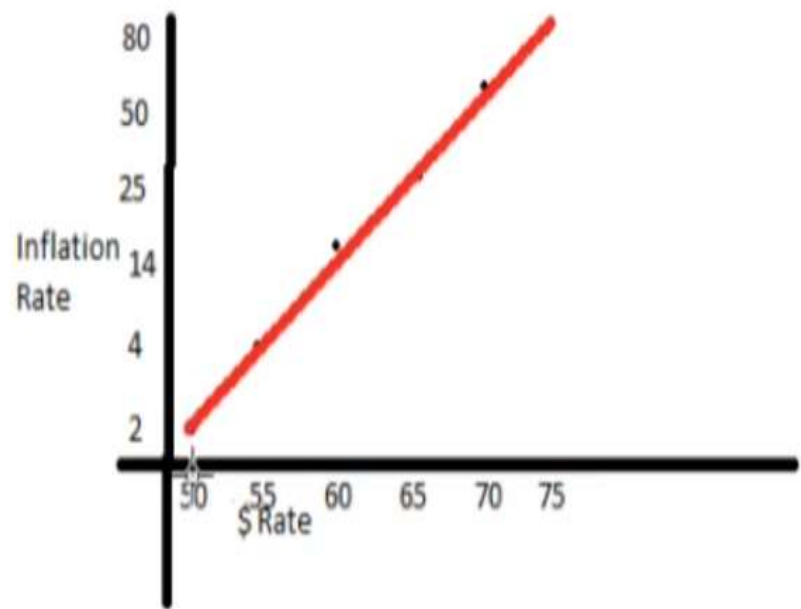
- **Represents some sort of quantitative measurement**
 - Heights of people, page load times, stock prices, etc.
- **Discrete Data**
 - Integer based; often counts of some event.
 - How many purchases did a customer make in a year?
 - How many times did I flip "heads"?
- **Continuous Data**
 - Has an infinite number of possible values
 - How much time did it take for a user to check out?
 - How much rain fell on a given day?

Categorical

- Qualitative data that has no inherent mathematical meaning
 - Gender, Yes/no (binary data), Race, State of Residence, Product Category, Political Party, etc.
- You can assign numbers to categories in order to represent them more compactly, but the numbers don't have mathematical meaning

Categorical								
	A	B	C	D	E	F	G	H
1	Categorical	Categorical						
2	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score
3	female	group B	bachelor's degree	standard	none	72	72	74
4	female	group C	some college	standard	completed	69	90	88
5	female	group B	master's degree	standard	none	90	95	93
6	male	group A	associate's degree	free/reduced	none	47	57	44
7	male	group C	some college	standard	none	76	78	75
8	female	group B	associate's degree	standard	none	71	83	78
9	female	group B	some college	standard	completed	88	95	92
10	male	group B	some college	free/reduced	none	40	43	39
11	male	group D	high school	free/reduced	completed	64	64	67
12	female	group B	high school	free/reduced	none	38	60	50
13	male	group C	associate's degree	standard	none	58	54	52
14	male	group D	associate's degree	standard	none	40	52	43
15	female	group B	high school	standard	none	65	81	73
16	male	group A	some college	standard	completed	78	72	70
17	female	group A	master's degree	standard	none	50	53	58
18	female	group C	some high school	standard	none	69	75	78
19	male	group C	high school	standard	none	88	89	86
20	female	group B	some high school	free/reduced	none	18	32	28
21	male	group C	master's degree	free/reduced	completed	46	42	46
22	female	group C	associate's degree	free/reduced	none	54	58	61
23	male	group D	high school	standard	none	66	69	62

\$	Inflation Rate
50	2
55	4
60	14
65	25
70	50
75	75-80

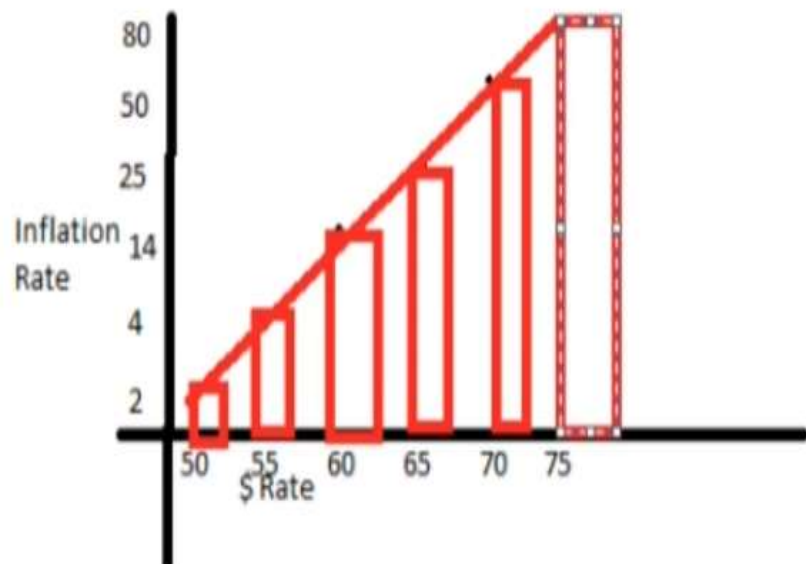


variable / feature / column = Inflation rate , \$ Rate

Relation = Linear Relation

Line Graph

\$	Inflation Rate
50	2
55	4
60	14
65	25
70	50
75	75-80



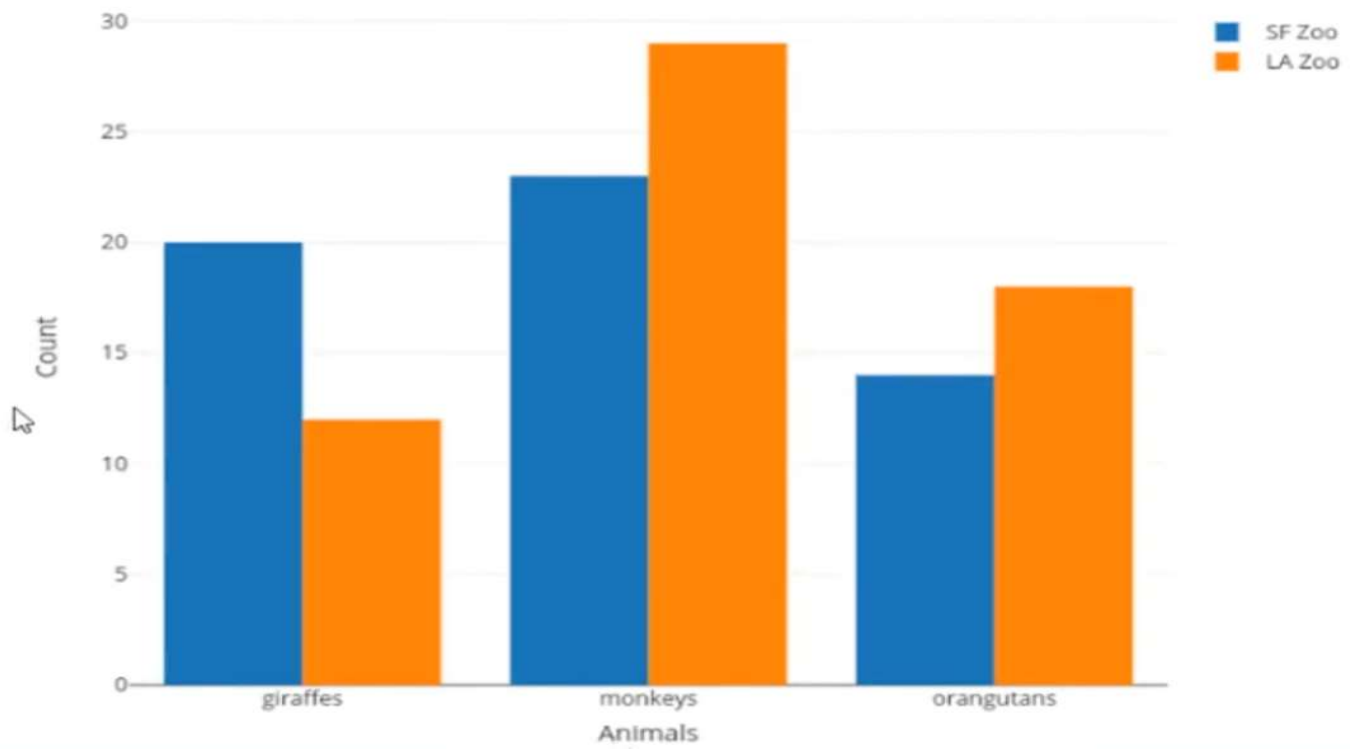
variable / feature / column = Inflation rate , \$ Rate

Relation = Linear Relation

Line Graph

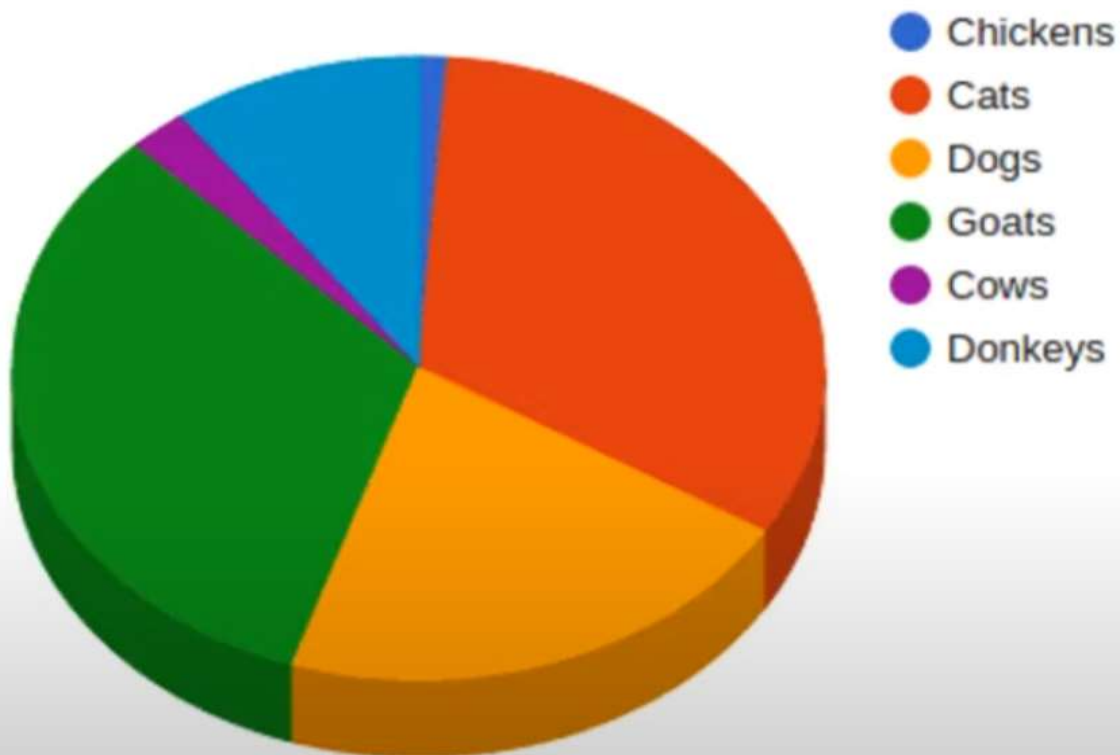
Types of Graphs

1. Bar Chart

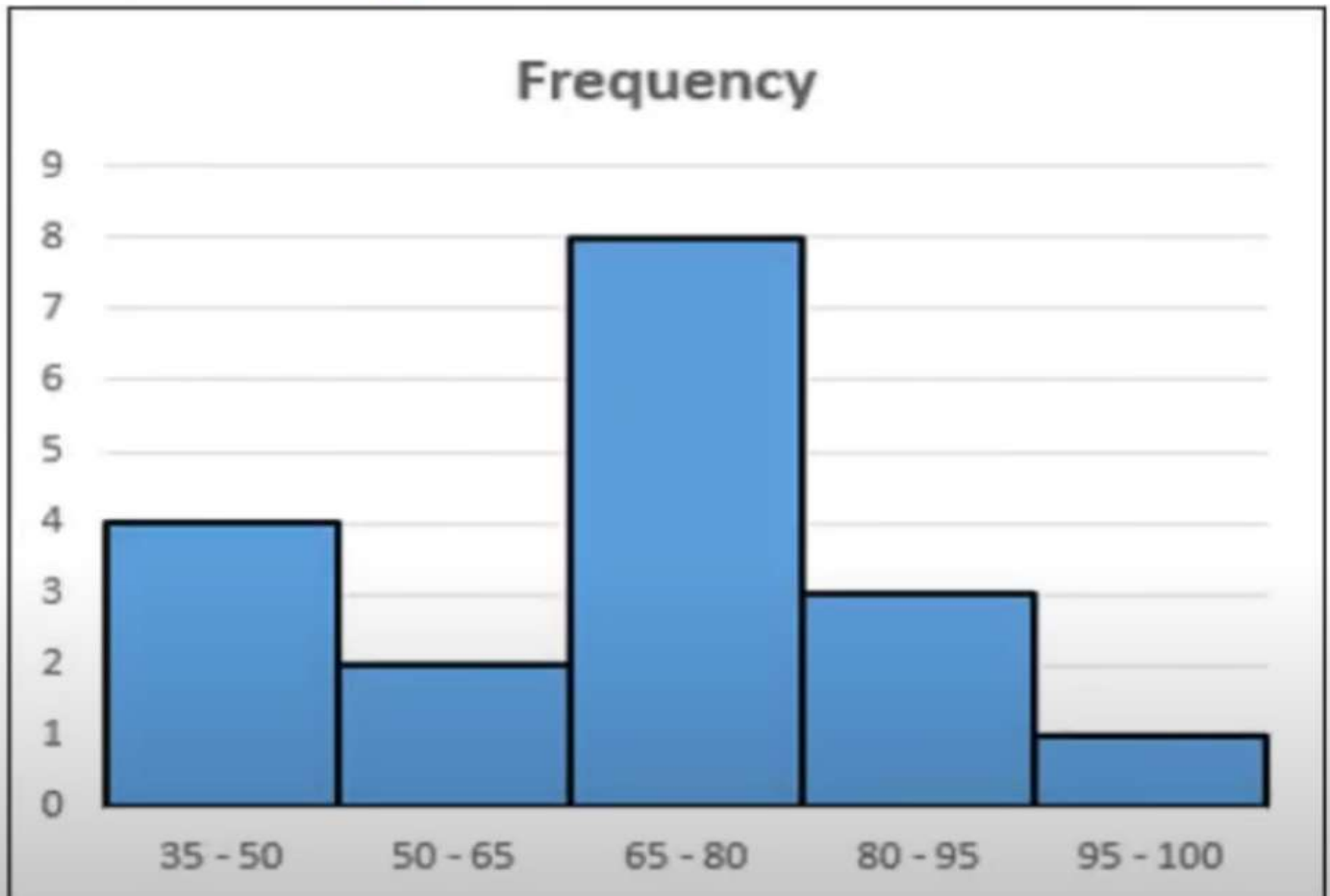


2. Pie Chart

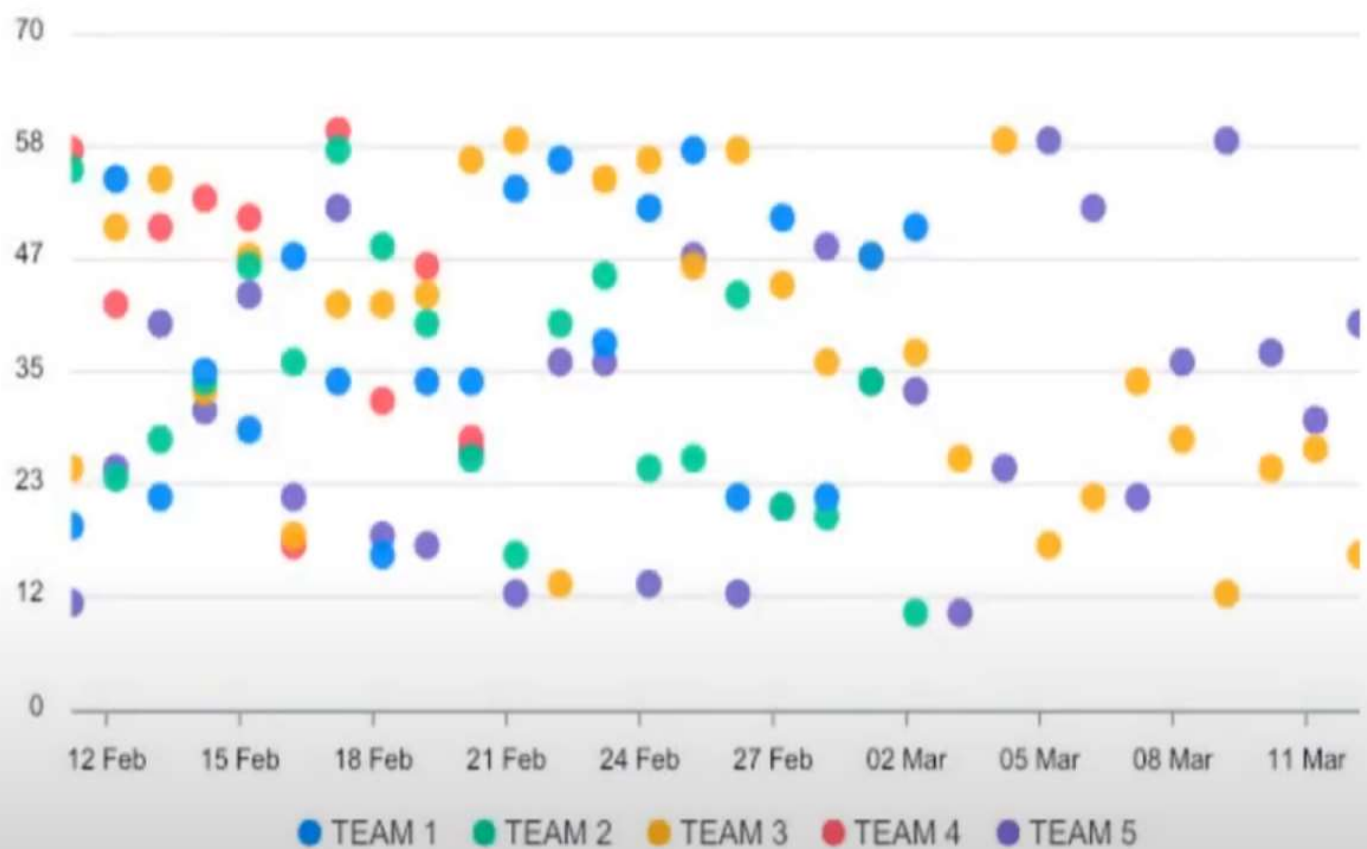
Animals



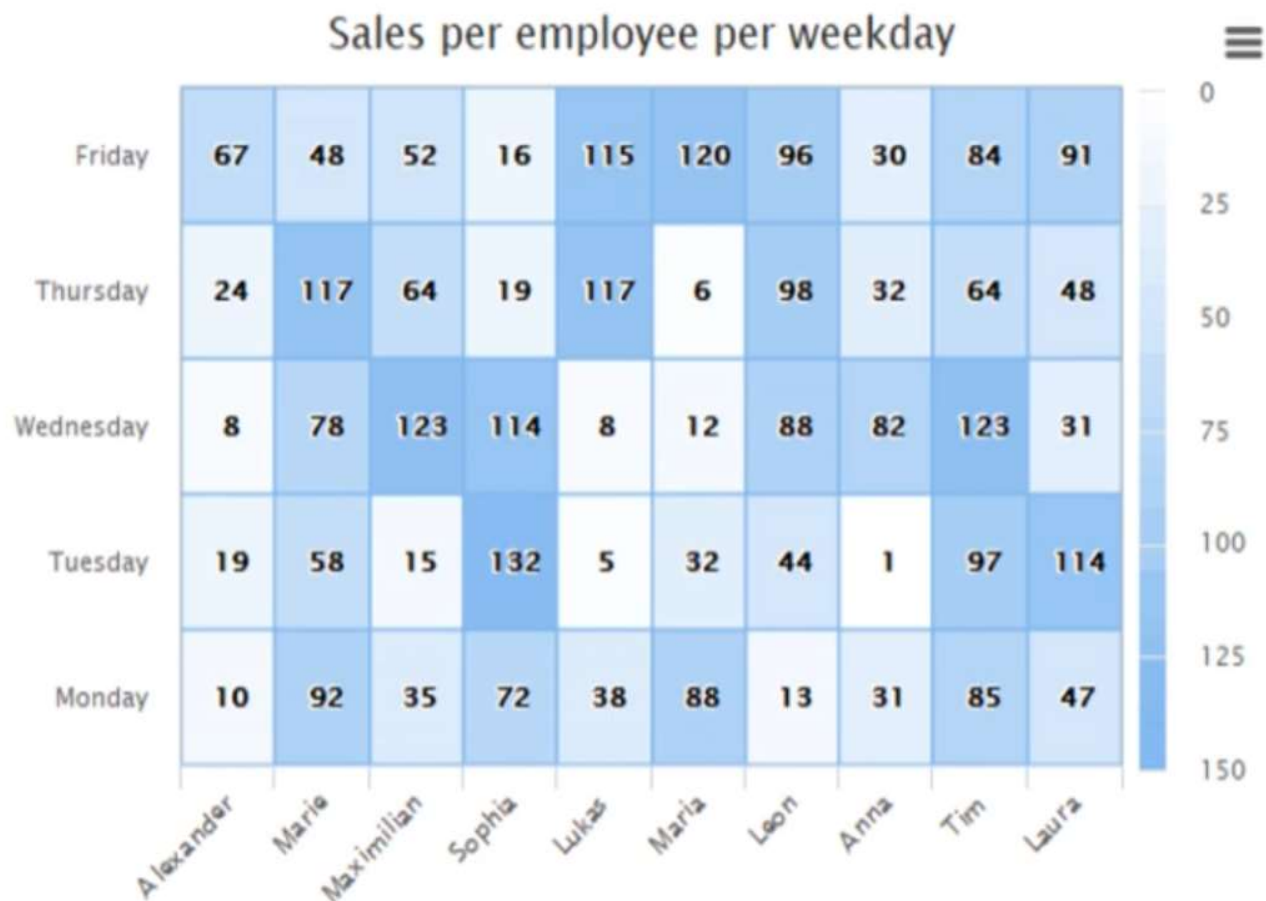
3. Histogram



4. Scatter Plot

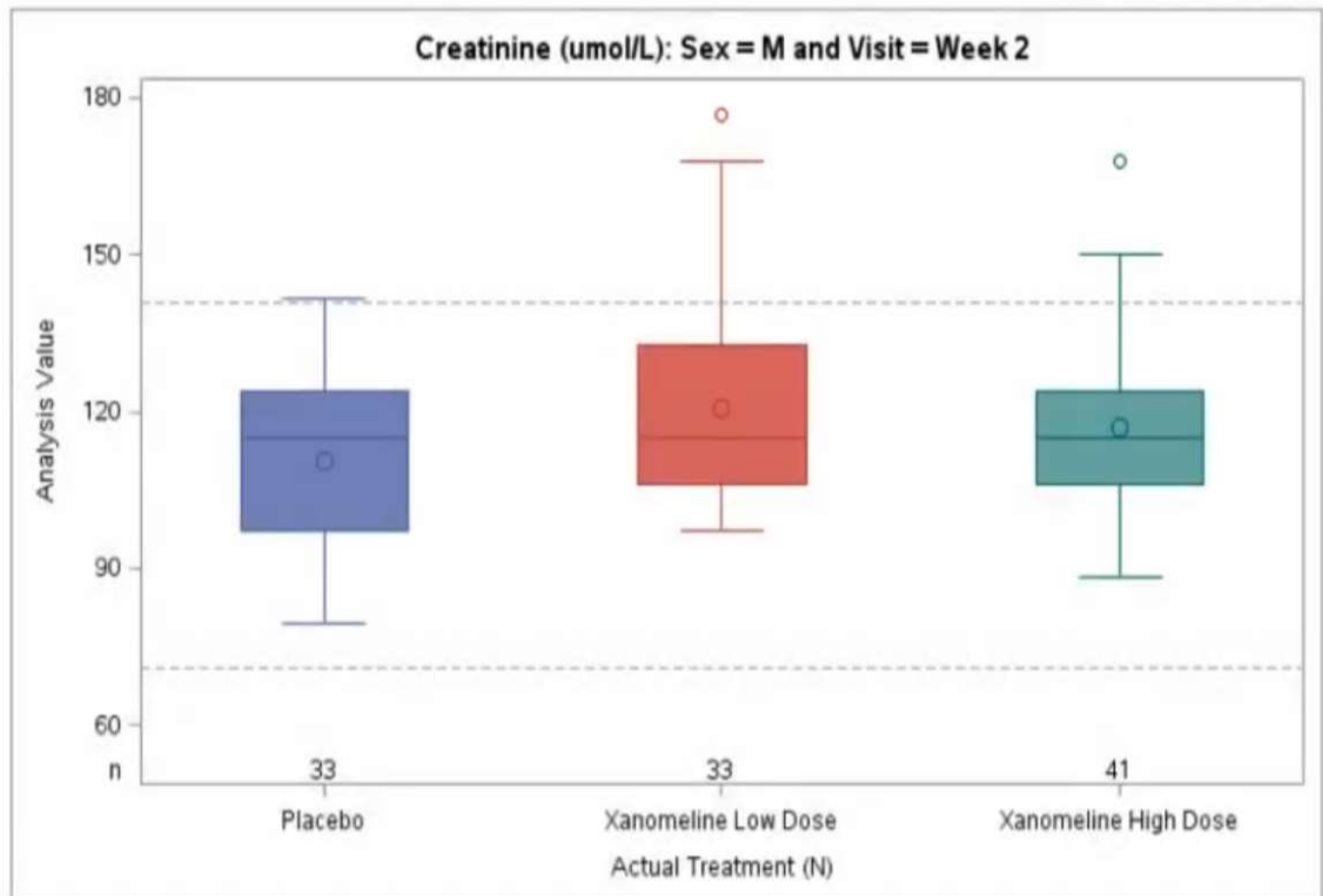


5. Heatmap



6. Box Plot

08:35 Sunday, August 14, 2016

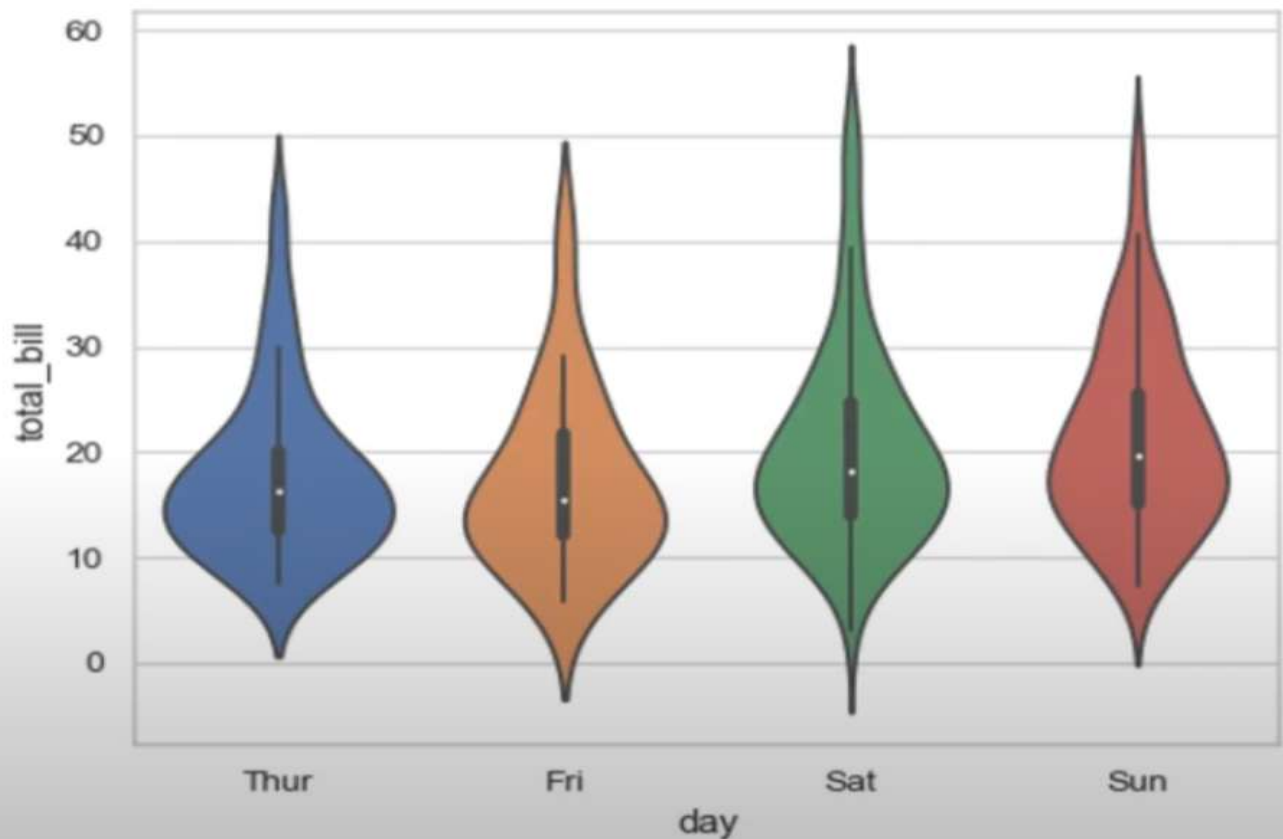


7. Line Plot

Product Trends by Month

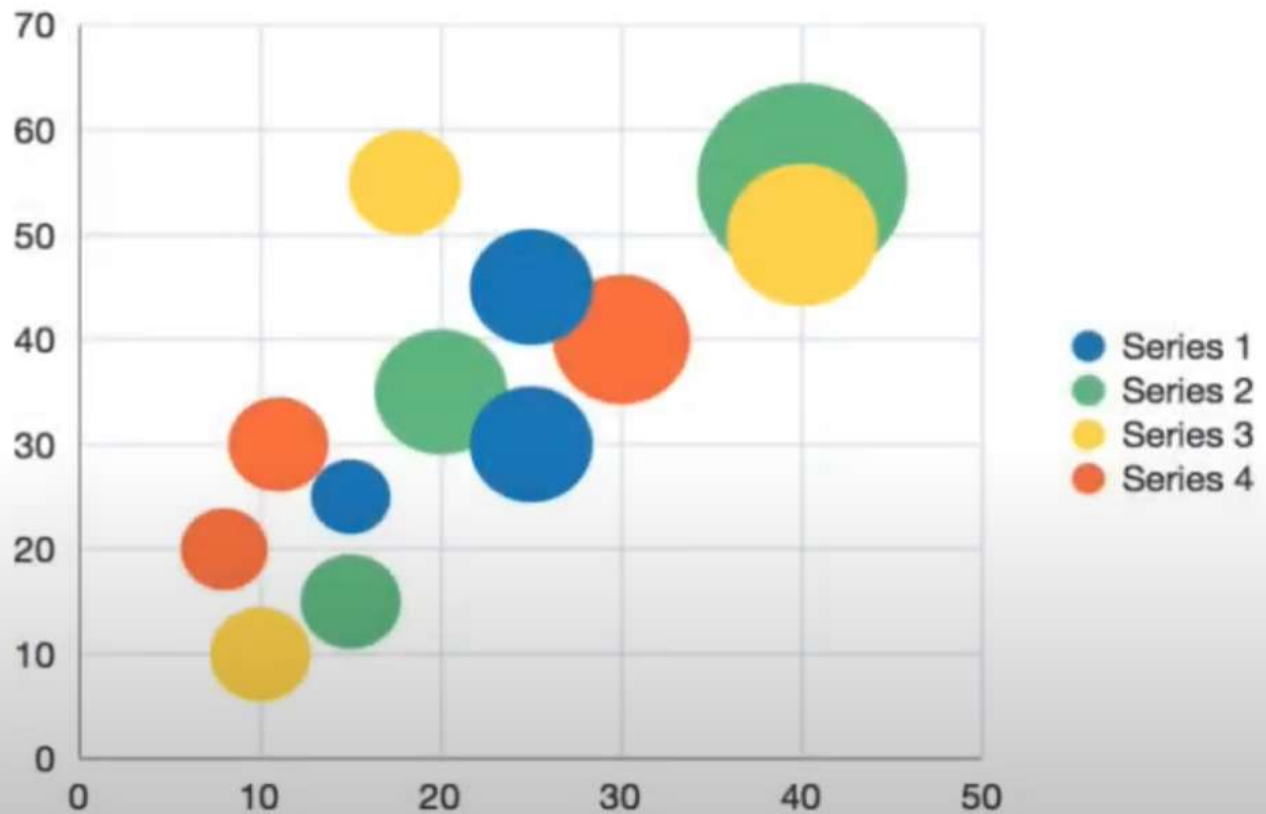


8. Violin Plot



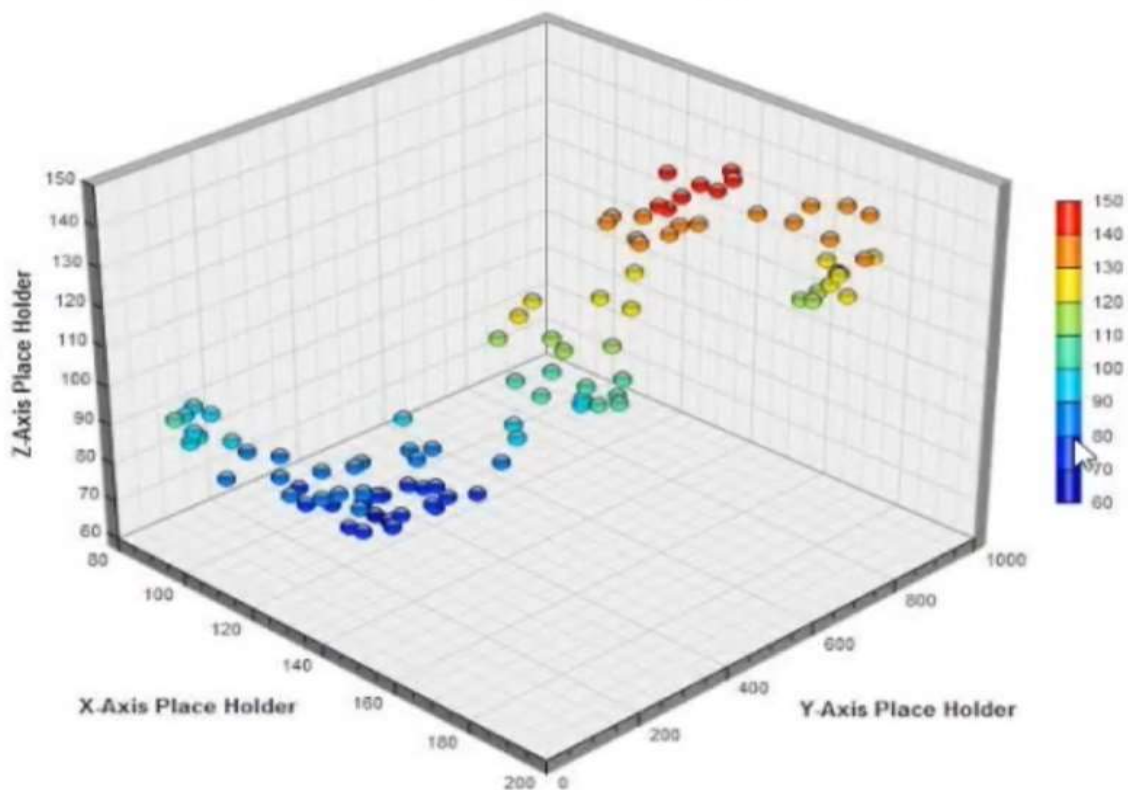
9. Bubble Plot

Bubble Chart



10. 3D Scatter Plot

3D Scatter Chart (1)



Advantages

- **Helps You Know Your Data:**
 - EDA lets you see what your data looks like, understand its structure, and notice any patterns or trends. It's like getting to know the data before you work with it.
- **Finds Problems Early:**
 - It helps you spot issues like missing data, errors, or unusual values early on. This way, you can fix these problems before they cause trouble in your analysis.
- **Guides Your Next Steps:**
 - EDA gives you ideas about what analysis or models might work best for your data. It points you in the right direction, making the next steps easier.
- **Visualizes Data Easily:**
 - EDA often involves creating simple charts or graphs that make it easier to understand and explain your data.

Disadvantages

- **Can Be Time-Consuming:**
 - EDA can take a lot of time, especially if your data is large or complex. You may need to try different ways of looking at the data to fully understand it.
- **Subjective Results:**
 - Different people might interpret the same data differently during EDA. This means the insights you gain can be influenced by who is doing the analysis.
- **Doesn't Provide Final Answers:**
 - EDA is just a starting point. It gives you insights but not the final conclusions. You'll need to do more detailed analysis to get solid results.
- **Risk of Over interpreting:**
 - There's a chance you might see patterns that aren't really important and try to make them fit into your analysis, leading to incorrect conclusions.

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv('data.csv')

# Display the first few rows
print("First 5 rows of the dataset:")
print(data.head())

# Basic statistics summary
print("\nBasic statistics summary:")
print(data.describe())

# Check for missing values
print("\nMissing values in each column:")
print(data.isnull().sum())

# Plot histogram for each numerical column
print("\nGenerating histograms...")
data.hist(figsize=(10, 8))
plt.show()
```

1. **Load the Data:** Reads the data from a CSV file into a DataFrame.
2. **View First 5 Rows:** Displays the first 5 rows to give a quick look at the dataset.
3. **Basic Statistics:** Provides a summary of statistics like mean, min, max, and quartiles for numerical columns.
4. **Missing Values:** Checks for any missing values in each column.
5. **Histograms:** Plots histograms for each numerical column to visualize their distributions