In [1]:
```
import pandas as pd
```

📊🐼 Complete Pandas Tutorial Series for Everyone 🚀📚

1. Python Pandas Tutorial | What is Pandas | Features of Pandas | Data Analysis # 1

https://youtu.be/2NwUfnKhsuU

In this video, you will learn what **Pandas** is and its features in greater detail 🔍 . You will also discover why **Pandas** is such a useful and widely-used library for data science, machine learning, data analytics, and data automation 🚀 .

The video will cover the following points:

- What is **Pandas** and its role in data science? 👩‍💻
- Why **Pandas** is a must-have tool for machine learning 🤖
- How **Pandas** simplifies data analytics 📊
- The key features of **Pandas** and why it's preferred for data automation ⚙️
- How **Pandas** enables efficient data manipulation and analysis 🔧
- Real-world applications of **Pandas** in various industries 🌍

---

2. Python Pandas Tutorial : Series and DataFrame Basics

https://youtu.be/iWsEfpWAJ3U

In this tutorial, you will learn about Python Pandas **Series** and **DataFrame** automation from basics to advance 🔄 .

Pandas 🐼 is a Python library used for data manipulation and analysis. It provides two primary data structures to work with - **Series** 📈 and **DataFrame** 📊 .

A Pandas **Series** is a one-dimensional labeled array that can hold data of any type (integer, float, string, etc.). The labels, called the index 🏷️ , can be integers or strings and provide a way to identify the data. A **Series** can be created from a list, tuple, dictionary, or even another **Series** 🔄 .

A Pandas **DataFrame** is a two-dimensional labeled data structure with columns of potentially different data types. It is similar to a spreadsheet or SQL table 🗂️ . A **DataFrame** can be created from a dictionary of **Series** or a list of dictionaries. For example:

This video tutorial will cover a complete understanding of the following topics:

- What are Pandas **Series** and their indexes? 📊
- Creating Pandas **Series** from different data structures 🔨
- Creating Pandas **DataFrame** from Lists, Dictionaries, etc. 🗃️
- How to modify and create indexes of **Series** and **DataFrame**? 🛠️
- Creating an empty **DataFrame** and appending rows and columns ➕
- Creating **DataFrame** from scalar values 🔢
- Different ways to create Pandas **DataFrame** and **Series** 🔄 .

---

3. Python Pandas Tutorial : Importing Data with Pandas#

https://youtu.be/EEJqtO5Npf8

In this Python Pandas tutorial, you will learn how to import data using Pandas 📩 . This guide includes numerous examples to demonstrate how to read and load data from various file formats 📂 .

The video tutorial will thoroughly cover the following topics:

- How to read **CSV** files? 📄
- Loading data into a Python **DataFrame** 📊 .

- How to read text data and **JSON** files? 📝
- How to read Excel files and load them into a Python **DataFrame**? 📈
- How to extract data from the clipboard 📋 ?
- How to read tables and **HTML** pages with specific criteria? 🌐

---

4. Python Pandas Tutorial : Pandas DataFrame Operations

https://youtu.be/g6XIFIAZISc

In this session, you will learn about Pandas **DataFrame** operations 📊 .

Guys, we can perform lots of operations on Pandas **DataFrame** like how to get rows/columns 📝 , **DataFrame** slicing ✂️ , head or tail 🐾 , set options for rows and columns 🧭 , reset indexes in **DataFrame** 🔄 , info method ℹ️ , data type method 🔢 , and the very useful method **DataFrame** Describe 📋 .

Pandas 🐼 : Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive, and flexible data structures that make data manipulation and easy analysis and time series, among many other things 🧠 .

This video tutorial will cover a complete understanding of the following points:

- How to read the data from **DataFrame** using Head or Tail method? 👀
- How to perform slicing concept to extract the data from Pandas **DataFrame**? 🔪
- When, Why, and How You Should Reshape Your **DataFrame**? 🔄
- Pandas Set Option for Rows / Columns ⚙️
- How To reset Indices, Rows, or Columns From a **DataFrame** 🔃
- Uses of **DataFrame** Info 🧾
- How to check data variable's type using **DataFrame** DataTypes method? 🔍
- What does **DataFrame** Describe method do and why we use it? 🧐
- How to get all the column names from Pandas **DataFrame**? 📄
- More... 🔜

---

5. Python Pandas Tutorial: Split Excel Data into Worksheet by Column Values

https://youtu.be/P8dpG5arKY0

📽️ In this video, we will learn how to **Split Excel Data into Worksheets by Column Values** 📑 .

You will explore how to use **Python Pandas** 🐼 to:

- 📂 Split Excel column data into **multiple worksheets** based on unique column values
- 📁 Save all worksheets into the **same workbook** using Python
- 🧠 Automate repetitive Excel tasks efficiently with **Pandas**
- ⚙️ Simplify your data processing workflows with easy-to-follow code

---

6. Python Pandas Tutorial: Combine Excel Sheets using Python Pandas

https://youtu.be/hTLmfqSQ-hA

⏱️ In this session, you will learn how to **combine Excel sheets** using **Python Pandas** 🐼 — all in just **5 minutes**!

We will cover:

- 📁 Merging data from multiple Excel sheets into a single **DataFrame**
- 🔄 Automating the combination process with minimal code
- 📊 Creating a unified dataset for analysis or reporting
- ⚡ Boosting your productivity with quick data consolidation tips

---

📊 7. Python Pandas Tutorial: DataFrame Operations (Rename or Drop Column, Set, Reset, and Use Indexes)

https://youtu.be/Zj5TDnR_1vU

📽️ In this video, you'll explore various **DataFrame operations** using `pandas` — such as renaming or dropping columns, setting or resetting indexes, and more!

🧠 Working with data in **Pandas** is as intuitive as using **Microsoft Excel**. If you're familiar with DataFrame operations, you'll find it super easy to manage and manipulate data. Pandas provides fast, flexible, and expressive data structures like `Series` and `DataFrames` for high-level statistical and time series analysis.

🎯 What You'll Learn in This Tutorial

- 🔄 **How to rename columns** in a Pandas DataFrame
- ❓ A beginner-friendly explanation of the `inplace` parameter
- ↔️ Applying **axis** correctly (rows vs columns)
- 🗑️ Deleting single or multiple columns based on conditions
- 🧹 Removing rows in a DataFrame using conditions
- 🔧 How to **set, reset, and delete** a DataFrame's index
- 🧪 Various hands-on assignments and operations with DataFrames

---

🚀 By the end of this tutorial, you'll have a solid understanding of these core operations to power up your data wrangling skills!

---

8. Python Pandas Tutorial: Pandas iloc and loc

https://youtu.be/bJo8N7bneJo

🎯 In this tutorial, we are going to dive deep into **Pandas** `iloc` and `loc` — two powerful tools for data selection in a **DataFrame** 📊 .

As you may already know, these two functions are essential for accessing and selecting data in **Pandas** 🐼 .

🔍 When it comes to selecting data in a **DataFrame**, `loc` and `iloc` are fan favorites: they're fast ⚡ , intuitive 💡 , and incredibly efficient 🔧 .

This video tutorial will give you a complete understanding of the following points:

- ❓ What is the difference between `iloc` and `loc` in Pandas?
- 🐍 How do I use `loc` and `iloc` in Python?
- 🔎 Find all rows based on a single condition in a column
- 🧱 Find rows with multiple conditions applied
- 🔢 Select rows by index using `iloc`
- 🎯 Select specific rows and columns using both indices and labels
- 📌 How to get specific rows and columns using `loc` and `iloc` ?
- 🔍 Apply filters on specific columns using `loc`
- 🏷️ Understand the use of **index** and **index label** in `loc` and `iloc`
- 🟰 Select rows where a column value equals a scalar
- ⚙️ Select rows that satisfy multiple boolean conditions while keeping specific columns
- 🧪 More practical examples and use-cases!

---

9. Python Pandas Tutorial: Different ways to filter Pandas DataFrame

https://youtu.be/j4T6OwimfFY

🎥 In this video lecture, you will learn various ways to **filter data** from a **Pandas DataFrame** 📊 .

Pandas 🐼 offers user-friendly tools like `loc`, `iloc`, and boolean expressions for filtering. However, these can become less effective when your filter parameters aren't static. The `query()` method is a more dynamic and readable alternative for such scenarios 🔍.

📌 **Data Filtering** is one of the most common data manipulation tasks — similar to the `WHERE` clause in SQL or the `Filter` feature in MS Excel 🧱. Pandas makes this operation fast and efficient.

📚 In this tutorial, you will learn:

- ✅ Apply filter on a single value
- 🔁 How to use **Boolean values** in filters?
- 🧩 Filter data based on **multiple columns**
- 📌 Filter multiple cells in a single or multiple columns
- ⚖️ Understand logical operators in filtering (`&`, `|`, `~`)
- 🧠 Apply **nested filtering criteria**
- 🔄 Use the `~` **(tilde)** operator for negation in filters
- 🔤 Filter data using **string values**
- 📩 Use `isin()` method for filtering with a list of values
- 💡 Apply **Lambda functions** within filters
- 🔍 Use `query()` along with `nlargest()`, `nsmallest()`, and `loc` for advanced filtering
- 🧪 Tons of real-world examples to master filtering!

---

10. Python Pandas Tutorial: How to handle missing values in Pandas – Part –1

https://youtu.be/5dwjSCMsr6M

Hi Guys, In this session you will learn about how to handle the missing value using Python Pandas or NumPy.

---

11. Python Pandas Tutorial: Several Ways To Handle Missing Value Part-2

https://youtu.be/dm5sEfE5Bow

👩‍🏫 In this session, you will learn **several ways to handle missing values** in **Python Pandas** 🐼 using methods like `isnull()`, `notnull()`, `dropna()`, and `fillna()`.

🎯 **Topics covered in this video:**

- ❓ Types of missing values
- 🛠️ How to deal with missing values effectively
- 🗑️ Deleting rows with missing values using `dropna()`
- 📈 Imputing missing values for **continuous variables**
- 🔠 Imputing missing values for **categorical variables**
- 🔮 Predicting missing values using logic or models
- 🔁 Using **Forward Fill** and **Backward Fill** methods
- 🔎 How to use `fillna()` with `'all'` and `'any'`
- 🧹 Multiple ways to define `dropna()`
- ⚖️ How to use the `thresh` parameter in `dropna()`
- 📍 How to fill nearest values in Pandas (interpolation-style logic)
- 📊 Examples using `axis` to detect and handle missing values across rows and columns

---

12. Python Pandas Tutorial: Pandas Concat and Append Method

https://youtu.be/k7kt6Du1u24

👋 In this session, you will learn various methods to **combine datasets** using **Pandas** 🐼 — specifically focusing on the `append()` and `concat()` methods.

🔗 Combining datasets is a fundamental part of data manipulation, and Pandas provides efficient tools to get the job done quickly and effectively.

🎯 **This video tutorial will give you a complete understanding of the following topics:**

- 📌 How to **append** data using `append()` method
- 📚 How to **combine multiple datasets** efficiently
- 🧩 How to use the `concat()` method for flexible joins
- ⚖️ **Difference** between `append()` and `concat()` methods
- 🏷️ Understanding the **role of index** in both methods
- ✅ What is the use of `verify_integrity=True` and how it helps maintain data consistency
- ➕ More practical examples and use-cases!

---

13. Python Pandas Tutorial: Joining and Merging Pandas DataFrame

https://youtu.be/Lm1RN5o964g

📘 **In this lecture,** you will learn how to perform **Joining** and **Merging** operations on **Pandas DataFrames** 🐼 in an elegant and efficient way.

🔍 In any real-world data science or analytics project using Python, combining multiple datasets is essential to form a meaningful analysis dataset. Pandas offers powerful, high-performance tools to handle these join and merge operations in-memory with ease.

🎯 **This session will cover the following topics in detail:**

- 🗂️ Categories of **Joining Pandas DataFrames**
- 🔗 **Inner Join** (Common Join)

- ⬅️ **Left Join** | Left Outer Join
- ➡️ **Right Join** | Right Outer Join
- 🔁 **Full Join** | Outer Join
- ➕ **Cross Join** (Cartesian Product)
- 🕵️ How to verify which dataset each record came from?
- 🧾 What does the `indicator=True` parameter do in merge operations?
- 🏷️ Setting **suffixes** to identify the origin of columns
- 🔑 Performing joins based on **index** and **primary key**
- 🧬 How to join **multiple DataFrames** using `merge()`
- 🧠 Understanding the usefulness of **Full Join** in Pandas
- ✅ How to verify and validate the **output** of your join operations

---

14. Python Pandas Tutorial: Data Analysis & Visualization With Pivot Table in Pandas

https://youtu.be/ofEmmJDKEX0

👋 Hi Guys! In this video, I'll walk you through how to create **Pivot Tables** with **visualizations** in **Python using Pandas** 🐼 📊 .

🧠 Pivot Tables are a vital concept in **data analysis**, **data wrangling**, and **pre-processing**. They allow you to **summarize and transform data quickly** from large datasets — just like in Microsoft Excel, but with the flexibility of Python!

💡 Using `pandas.pivot_table()`, you can gain a new perspective on your data — and even turn insights into visuals for better storytelling.

🎯 **This video covers the following topics in depth:**

- 📌 Introduction to **Pandas Pivot Tables**
- ❓ **Why use Pivot Tables?** Key use cases
- ⚙️ Syntax & parameters of `pivot_table()` function

- 🎯 How to **select specific columns** in a Pivot Table
- 🧱 Using **aggregate functions** (like sum, mean, count)
- ➕ Apply **multiple aggregate functions** on a single column
- 🚫 How to **handle null/missing values** in Pivot Tables
- 📊 Show **totals** in rows and columns with `margins=True`
- 🔍 How to **filter data** in a Pivot Table
- 📑 Sort the values in your Pivot Table
- 🔬 Create Pivot Table for a specific portion of the data
- 🧪 Use the `query()` method with Pivot Tables (and its benefits)
- 📉 Create **bar chart** from Pivot Table
- 📈 Create **line chart** from Pivot Table
- 🗓️ Plot **sales month-wise** by extracting month from a date column
- ✨ And much more!

---

15. Python Pandas Tutorial: Pandas Apply Function and Vectorization

https://youtu.be/vT-MtZH_oV0

👋 Hey everyone! In this video, we're going to dive deep into the **Pandas `apply()` function** 🐼 — one of the most useful tools for applying transformations across your data in Python.

💊 Applying functions to every row or column in a DataFrame is a common and powerful technique in data preprocessing, cleaning, and feature engineering. This tutorial shows you exactly how it's done — from basics to advanced use cases!

🎯 **This video tutorial will cover the following topics in detail:**

- ❓ What is the **`apply()` function** in Pandas?
- 🛠️ How to **use `apply()` on DataFrames** and Series
- 📊 How to use **`apply()` on Series** for simple transformations

- 🎯 Using `loc` inside **`apply()`** for conditional logic
- 🧾 How to create a **Sales Bucket** using a user-defined function with `apply()`
- ⚡ Use of **lambda (anonymous functions)** with `apply()`
- 🐢 Why **`apply()` can be slow** on large datasets
- 🚀 How to boost speed with **Vectorization**
- ⏱️ Measure the execution time of `apply()` vs vectorized operations
- ✨ And much more!

💡 By the end of this session, you'll be able to choose the most efficient way to apply logic across your DataFrame — making your data workflows cleaner and faster!

---

16. Python Pandas Tutorial: DataFrame Conditional Formatting and Styling

https://youtu.be/bBbxumJakuA

🎨 **Pandas DataFrame Formatting & Styling** can significantly enhance how data is presented to clients or stakeholders by making it more *visually appealing* and *easier to interpret* 📊 .

In this session, you'll explore multiple examples to learn how to **customize and style your DataFrames** using powerful formatting techniques. These are practical skills that improve both clarity and aesthetics in reports, dashboards, or presentations. 💼📈

🚀 **What you will learn:**

- 💯 How to **apply number formatting** in Pivot Tables and DataFrames
- 🌈 How to **set background colors** based on numerical values
- ⚠️ **Highlight missing values** with custom colors
- 🏆 How to **highlight the max or min** values in a column
- 📊 Apply **Conditional Bar Formatting** and align data visually
- 🌡️ Use **background gradients** with `cmap` or Seaborn

- 🖱️ Highlight cells **on hover** for better interactivity
- 💻 Use **HTML & CSS** to customize DataFrame layout
- 📩 **Export DataFrames to Excel** with styles preserved
- 🚫 How to **hide (not delete)** specific rows/columns
- 🔲 Apply **conditional formatting & borders** on headers and indexes
- 📚 Tons of practical examples to bring it all together!

💡 Whether you're preparing a dashboard, client report, or interactive notebook, these styling techniques will give your data the professional polish it deserves.

---

17. Create Fake Data Instantly with Python!

Create Fake Data Instantly with Python!

🧪 **In this session, you'll learn how to generate realistic Fake Data using Python Faker and Pandas**.

📦 **Python Faker** is a powerful library for generating fake, yet realistic, data — perfect for *testing, prototyping, and anonymization*. From names and emails to credit cards and dates, this tool is a developer's best friend when real data isn't available or appropriate. 🔐

🎯 Faker allows you to:

- 🧍 Generate fake **names, addresses, phone numbers, and other personal info**
- 📧 Create fake **emails and usernames**
- 💳 Generate fake **credit card numbers** and financial data
- 📅 Generate random **dates and times**
- 🖼️ Create **fake images and media** for prototyping
- 🌐 **Localize data** to specific languages & regions
- 🧪 Generate **unit test data** easily
- 📄 Generate **random text** with specific formats
- 📊 Produce **machine learning test datasets**

- 📄 Build **Pandas DataFrames** with custom columns like: `['Order_ID', 'Order_Date', 'Customer', 'Region', 'Category', 'Sub_Category', 'Product', 'State', 'Sales', 'Profit', 'Quantity']`
- 👤 Use the **fake profile** method to get full user info including name, username, address, and contact

✨ With **Python Faker**, you'll master fake data creation that looks and feels real — saving time, ensuring privacy, and keeping your dev/test pipelines efficient and clean. 💡

📽️ **This video tutorial includes real examples and use cases to help you start generating fake data like a pro!**

---

18. 📋 Pandas Windows Rolling Function Tutorial 📋

https://youtu.be/nSbN1uqUf9w

📊 **Master Pandas Rolling Functions with Window Operations!**

In this comprehensive guide, we dive deep into the world of **Pandas rolling functions** on Windows, revealing their incredible power for data manipulation and time series analysis. 🧠 Whether you're a seasoned data scientist or a Python beginner, this tutorial will equip you with essential techniques to level up your data skills. 🚀

📋 What You'll Learn:

- 🔍 **Understanding the fundamentals** of rolling functions
- 📆 Exploring **types of rolling windows** (time-based ⏱️ & fixed-size 📏)
- 🧪 Hands-on with **rolling calculations**: mean, sum, min, max
- 📐 Perform advanced **statistical operations** using rolling Window functions
- ⏮️ **Lag** & ⏭️ **Lead functions** in Pandas
- ⚙️ Customize **rolling windows** with size, min_periods, center, etc.
- 📈 **Visualize rolling data** for trends and patterns
- 🕒 Perform **time series analysis** with rolling functions

- 💼 Apply concepts to **real-world scenarios** and datasets
- 💡 Packed with **useful examples** and visualizations!

✨ Whether you're smoothing noisy data, analyzing moving averages, or forecasting with time-based rolling windows — this tutorial gives you all the tools you need using `Pandas` and `Python`.

📽️ **Get ready to roll with confidence!**

👉 19. Python Project : Pandas Live Case Study on Uber Cab's Data

https://youtu.be/zLYC_Py_rk4

🚕 Live Case Study: Uber Cab Data Cleaning & Transformation with Python 🧼📊

In this session, we'll dive into a real-world case study using **Uber Cab's dataset** and solve it using **Python** 🐍 and its data-centric libraries like `pandas`, `numpy`, and `pandas_profiling`.

This hands-on lecture will equip you with an **advanced understanding of data cleaning and transformation** techniques — crucial for any data analysis or data science workflow. 🚀

🔍 Topics Covered:

1. 📩 Reading the Data
2. 👀 View First `n` Rows of Data
3. 👀 View the Last 5 Rows of Data
4. 🔢 Total Number of Rows & Columns
5. 🔢 Total Number of Elements in the Dataset
6. ⚠️ Total NULL Values Per Column

7. ✅ Total Non-NULL Values Per Column
8. ❓ Filter Entries with NULL in `Purpose` Column
9. ✅ Filter Entries with NON-NULL in `Purpose` Column
10. 🧹 Remove `*` from Column Names using `rename()`
11. 🧹 Remove `*` from Column Names using `str.replace()`
12. 🧹 Remove `*` using `lambda` function
13. 📍 Filter START Location = `Fort Pierce`
14. 📍 Filter STOP Location = `Fort Pierce`
15. 📊 Sort by `MILES` (Descending)
16. 🗑️ Drop Rows with NULL in `STOP` Column
17. 📈 Statistical Summary using `describe()`
18. 📝 Generate HTML Report using **Pandas Profiling**
19. 📌 Understand START & STOP Points
20. 📊 Analyze Categorical Values with `value_counts()`
21. 🔁 Find Rides with Same START & STOP
22. 🏆 Favorite START Location by Total Miles
23. 🌟 START Point for Ride with Max Miles
24. 🔍 Check Data Types of Columns
25. ❌ Drop `unknown location` in START & STOP
26. 📌 Most Popular START-STOP Pair
27. ⏱️ Convert `START_DATE` & `END_DATE` to `datetime`
28. 📆 Extract Month from `START_DATE` & Analyze Ride Distribution
29. 📏 Calculate Average Miles per Month
30. 📅 Extract Day from `START_DATE`
31. 🧾 Total Miles by **Category** and **Purpose**

🎁 **Bonus Insight:** Real-world data problems with best practices and clean coding tips! 💡

📽️ Whether you're a data analyst, data scientist, or just curious about Uber rides — this session will give you practical skills to clean, transform, and analyze messy data efficiently. Let's get started! 🔧📈

---

20. Python Data Science: Analyzing LinkedIn Connection Data: A Comprehensive Case Study

https://youtu.be/wHeh-V9igdo

🔗 Analyzing LinkedIn Connection Data: A Comprehensive Case Study 📊

In this insightful case study, we dive deep into the world of **LinkedIn Connection Data** using **Python** 🐍 . Discover the methods, tools, and key findings from examining the intricate web of LinkedIn networking.

This session is more than just an analysis — it's a complete **End-to-End Exploratory Data Analysis (EDA)** and **Data Visualization journey** that brings raw data to life. In today's data-driven world, understanding how to extract insights from platforms like LinkedIn can guide strategic networking, professional growth, and community understanding. 🌐

🧠 What You'll Learn & Explore:

- 📊 **LinkedIn Data Analysis** — Dive into your network's structure and behavior
- 💻 **Python Case Study** — Step-by-step real-world data science pipeline
- 🌐 **Networking Strategies** — Discover connection patterns and hidden opportunities
- 🔍 **Exploratory Data Analysis** — Uncover insights, trends, and anomalies
- 🧪 **Data Science Concepts** — Learn how Pandas, NumPy, Seaborn, and Matplotlib fit in
- 📈 **Data Visualizations** — Visualize your network using bar charts, heatmaps, network graphs & more
- 🎯 **Real-life Applications** — Build strategies for career growth and smarter networking

Whether you're:

- 👩‍💻 A **data science enthusiast**
- 👨‍🏫 A **professional optimizing your LinkedIn strategy**
- 📚 A **Python learner** seeking real-life projects

...this session has something valuable for you. ✅

✨ Bonus:

Uncover anomalies, connection overlaps, top industries, most engaged cities, and relationship clusters in your LinkedIn network!

🚀 Get ready to level-up your data analysis skills and understand your professional connections like never before.

AI with Pandas Locally

`21. AI Assistant for Data Analysis & Machine Learning using Python Pandas & Sketch Library`

https://youtu.be/cJ551vcFGm4

🤖 Welcome to the Future of Data Analysis & Machine Learning! 🚀

In this exciting video, we unveil a **groundbreaking AI assistant** designed to **supercharge your data analysis and machine learning workflows**! Whether you're a data enthusiast, beginner, or an experienced analyst, this tool is here to level-up your productivity and insights. 💡

📦 **Powered by the Pandas Sketch Package**, this intelligent assistant allows you to:

- 📊 Perform EDA (Exploratory Data Analysis) lightning-fast ⚡
- 🧹 Clean, transform, and engineer features with ease
- 📈 Visualize patterns and trends without hassle
- 🧠 Apply machine learning-friendly transformations

🧰 You Will Learn:

- 🛠️ How to display column names of a DataFrame in a **dictionary format**
- 🛠️ How to separate **discrete vs continuous columns**
- 🛠️ Finding **unique sub-categories and their counts**
- 🛠️ Determining **top-performing states** based on sales data 💸
- 🛠️ Exploring **sales segment distributions**
- 🛠️ Identifying the **top-selling sub-category**
- 🛠️ Pinpointing the **state with the highest number of sales**
- 🛠️ How to learn **data analysis using Sketch** 🤓
- 🛠️ Performing **summary stats** and handling **null values**
- 🛠️ How to **detect and handle outliers** 🧯
- 🛠️ Performing **feature engineering & data transformation**
- 📚 Understanding the **Central Limit Theorem** & its real-world value
- 📊 Applying **ANOVA** with practical steps
- 🔍 The key differences between a **Z-test and F-test**
- 📋 Essential **statistics topics** for Data Analysis & Machine Learning
- ✅ **Real-world examples** throughout! 😊

🎯 Whether you're automating reports, exploring new datasets, or prepping for model building, this AI assistant can be a **game-changer in your data journey**.

`22. Mastering in Data Analysis and Pandas DataFrame using Sketch AI`

https://youtu.be/5XBq2bulbBI

🚀📊 Welcome to Your Transformative Journey in Data Analysis with Sketch AI & Pandas DataFrame!

Welcome to an exciting and **transformative journey** in **Mastering Data Analysis** using the powerful combination of **Sketch AI** and **Pandas DataFrame**! 🌟 In this **comprehensive tutorial**, we'll dive deep into the dynamic world of data analysis, equipping you with the skills to **elevate your data analysis game**.

Whether you're a **seasoned data scientist** or a **beginner** eager to explore the realm of **data analytics**, this video offers invaluable insights and practical techniques to help you:

- 🔍 Gain **confidence** and **expertise** in tackling real-world data challenges
- 📈 Master the art of working with **Pandas DataFrame** for data manipulation and analysis
- 🤖 Leverage the power of **Sketch AI** to enhance your analysis workflows
- 💡 Unlock actionable insights from complex datasets, whether it's for business, research, or beyond

🌍 Whether you're analyzing **financial trends**, predicting **market behaviors**, or exploring patterns in **scientific research**, mastering data analysis has become essential in today's **data-driven world** 🌐 . With Sketch AI and Pandas DataFrame as your allies, you will:

- 📊 Understand data patterns and trends with greater clarity
- 📉 Analyze large datasets quickly and efficiently
- 💪 Gain the ability to tackle real-world data problems with precision

🔧 **Are you ready to dive in?** With Sketch AI and Pandas DataFrame as your toolkit, you're on the fast track to becoming a data analysis expert! 🚀

---

23. 🎩 Master Essential Pandas Magic Tricks for Effortless Data Analysis

https://youtu.be/pf21nH211jU

🎩 Master Essential Pandas Magic Tricks for Effortless Data Analysis

Are you ready to transform how you work with data in Python? In this video, we'll explore **6 must-know Pandas functions** that will save you time and make your code more efficient and elegant:

- 🔎 ***select_dtypes()*** – Easily filter and focus on specific data types (e.g., numeric, categorical, or strings), streamlining your data-cleaning process.
- 🔍 ***query()*** – Write cleaner, more readable code with SQL-like syntax to filter rows, making complex filtering much simpler.
- 🔗 ***pipe()*** – Build reusable and modular data pipelines, keeping your data transformations organized and intuitive.
- 🎨 ***applymap()*** – Apply transformations to every single cell in your DataFrame effortlessly, enabling quick, element-wise operations.
- ➕ ***assign()*** – Add new columns dynamically in a chainable, readable way to enhance your workflows.
- 💥 ***explode()*** – Handle lists within your DataFrame by flattening them into individual rows, making your data easier to analyze.

These tricks are perfect for learners and professionals who want to work smarter, not harder, with Pandas. By the end, you'll have powerful tools to handle even the most complex datasets with ease. 🚀

🔥 **Ready to level up your data skills?** Dive in and let's unlock the full potential of Pandas! 💡

`24. Master Data Analysis in Python: Column Summary Function with Visualization using Pandas`

https://youtu.be/KdMD1xEDFCw

🚀 Unlock the Power of Data Analysis in Python!

In this video, we build a **Column Summary Function** that helps you quickly analyze a Pandas DataFrame by generating key statistics, detecting missing values, and visualizing data distributions. 📊 📈

👉 What You'll Learn:

- ✅ **Compute key metrics** like mean, median, variance, skewness, and kurtosis

- ✅ **Detect null values** and memory usage in each column
- ✅ **Visualize histograms, boxplots, and missing values heatmaps**
- ✅ **Improve your data preprocessing and feature engineering skills**

This Video Can Help You in the Following Ways

- 📊 ***Improve Your Data Analysis Skills*** – Learn how to summarize and explore your dataset efficiently.
- 🔍 ***Understand Key Statistical Metrics*** – Get insights into mean, median, variance, skewness, kurtosis, and more to better understand your data.
- 🛠️ ***Detect Missing & Outlier Values*** – Identify null values and visualize data distributions with histograms, boxplots, and heatmaps.
- ⚡ ***Optimize Your Data Preprocessing Workflow*** – Quickly generate column-level summaries to speed up feature engineering in machine learning projects.
- 📈 ***Enhance Your Data Visualization Skills*** – Use Seaborn and Matplotlib to create powerful visualizations for better decision-making.
- 🔥 ***Boost Your Python & Pandas Expertise*** – Master data manipulation, statistical analysis, and visualization in Python for data science.
- 💡 ***Perfect for Beginners & Professionals*** – Whether you're learning Python or a data scientist, this function will save time and improve efficiency!

🔥 ***Ready to level up your data skills?*** Dive into this tutorial and master data analysis with Python! 💡

`Pandas Bonus:`

Your code is a comprehensive guide to performing various essential operations using pandas in Python. Here's a brief breakdown of what each part of the code does, along with some potential improvements or suggestions for clarity:

📦 Importing Pandas

```python
import pandas as pd  # 📦 Import pandas library
```

- This imports the pandas library, which is crucial for data manipulation and analysis.

📊 Basic Exploration

```
# 📊 Load data from CSV file
df = pd.read_csv('file.csv')

# 👀 Basic Exploration
df.head()                                   # Display first 5 rows
df.tail()                                   # Display last 5 rows
df.shape                                    # Get dimensions of df
df.columns                                  # Get column names
df.index                                    # Get row index
df.info()                                   # Show datatypes and memory usage
df.describe()                               # Summary statistics
```

- **Explanation**: This section focuses on loading the data and exploring its basic structure.
  - `df.head()` and `df.tail()` are used to quickly preview the data.
  - `df.shape` provides the dimensions (rows, columns).
  - `df.columns` lists the column names, while `df.index` gives the row index.
  - `df.info()` is helpful to check the data types and memory usage, and `df.describe()` provides summary statistics for numerical columns.

🔍 Selecting and Filtering

```
# 🔍 Selecting and Filtering
df['column']                                # Select single column
df[['col1', 'col2']]                        # Select multiple columns
df.loc[row_label]                           # Select rows by label
df.iloc[row_index]                          # Select rows by position
df[df['column'] > value]                    # Filter by condition
df.query('column > value')                  # Query with string condition
```

- **Explanation**: Here, different ways of selecting and filtering data are explored.
  - `df['column']` for a single column, and `df[['col1', 'col2']]` for multiple columns.

- `.loc[]` is used for label-based selection, and `.iloc[]` is used for position-based selection.
- You can filter rows with conditions using `df[df['column'] > value]` or using the `query()` method for cleaner syntax.

🚑 Handling Missing Data

```
# 🚑 Handling Missing Data
df.dropna()                                 # Remove missing values
df.fillna(value)                            # Fill missing values
df.isna().sum()                             # Count missing values
```

- **Explanation**: Missing data handling is critical in any dataset.
  - `dropna()` removes rows with missing values.
  - `fillna(value)` replaces missing values with a specified value.
  - `isna().sum()` counts how many missing values exist in each column.

🔢 Sorting and Aggregating

```
# 🔢 Sorting and Aggregating
df.sort_values('column')                    # Sort by column
df.sort_values(['col1', 'col2'], ascending=[True, False])  # Multi-column sort
df.groupby('column').agg(['mean', 'sum'])   # Group & aggregate
df['column'].value_counts()                 # Count unique values
```

- **Explanation**: Sorting, aggregation, and grouping are essential for summarizing and organizing data.
  - `sort_values()` allows sorting by one or more columns.
  - `.groupby()` and `.agg()` are powerful tools for performing group-wise operations like mean and sum.
  - `value_counts()` counts the occurrences of each unique value in a column.

🔗 Merging and Reshaping

```
# 🔗 Merging and Reshaping
df.merge(df2, on='key')                     # Merge dataframes
df.concat([df1, df2])                       # Concatenate dataframes
df.pivot(index='A', columns='B', values='C')  # Pivot table
df.melt(id_vars='id')                       # Unpivot (melt) dataframe
```

- **Explanation**: Merging, reshaping, and pivoting allow for more complex data transformations.
  - `merge()` merges two DataFrames based on a common key.
  - `concat()` combines multiple DataFrames along a particular axis (row-wise or column-wise).
  - `pivot()` is used to create pivot tables, and `melt()` unpivots a DataFrame.

🛠️ Column Operations

```
# 🛠️ Column Operations
df['column'].mean()                          # Column average
df['column'].apply(func)                     # Apply function to column
df.rename(columns={'old': 'new'})            # Rename columns
df.drop('column', axis=1)                    # Drop column
df.assign(new_col=df['col1'] * 2)            # Add new column
```

- **Explanation**: This section covers column-wise operations.
  - `mean()` calculates the average of a column.
  - `.apply()` allows for applying a function to each element of a column.
  - `rename()` changes column names, while `drop()` removes columns.
  - `assign()` allows creating a new column dynamically.

🔑 Indexing

```
# 🔑 Indexing
df.set_index('column')                       # Set index
df.reset_index()                             # Reset index
```

- **Explanation**: Indexing helps in setting and resetting the row labels of the DataFrame.
  - `set_index()` sets a specific column as the index.
  - `reset_index()` resets the index back to the default integer-based index.

📤 Exporting

```
# 📤 Exporting
df.to_csv('new_file.csv', index=False)       # Save dataframe to CSV
df.to_excel('file.xlsx')                     # Export to Excel
```

- **Explanation**: After analysis, exporting the data is essential.
  - `to_csv()` and `to_excel()` allow saving DataFrames in CSV and Excel formats.

🔎 Other Handy Tricks

```
# 🔎 Other Handy Tricks
df.duplicated().sum()                    # Count duplicates
df.drop_duplicates()                     # Remove duplicates
df.sample(n=5)                           # Random sample
df.nunique()                             # Count distinct values per column
pd.get_dummies(df, columns=['col'])      # One-hot encode categorical column
```

- **Explanation**: These tricks improve data cleanliness and sampling.
  - `.duplicated().sum()` counts duplicated rows.
  - `drop_duplicates()` removes duplicates.
  - `sample()` provides a random sample of rows.
  - `nunique()` counts the number of unique values per column.
  - `get_dummies()` is used for one-hot encoding categorical data.

🔗 Connect with Me!

Want to learn more? Stay connected for updates and tutorials!

- 🔗 **LinkedIn: Abhisheak Saraswat**
- 📹 **YouTube: ProgrammingIsFunn**