Data Analytics Kickstart

Learn Python, SQL & Power BI Without Overwhelm

Beginner-Friendly Guide for New Data Analysts

By Veronica, aka
Data Geek is my name
Educator | YouTuber | Data Enthusiast

https://www.datageekismyname.com

Table of Contents

📖 Introduction

Welcome to ***Data Analytics Kickstart*** - your step-by step beginners guide to understanding the core tools used in the world of data.

In this eBook, you'll learn how to:

- Clean and transform data using **Python**
- Extract insights using **SQL**
- Create a dashboard with **Power BI**

Whether you're changing careers, starting a side hustle, or just want to understand data better you're in the right place.

Why These Tools?

✓ **Python** is perfect for cleaning and transforming messy data
✓ **SQL** is the go-to language for working with databases
✓ **Power BI** turns your data into beautiful, interactive visuals

What to Expect

Take your time with each section. Practice. Learn.
You don't need to be perfect – Just keep going.

Let's begin your journey into the world of data analytics!

– *Veronica* aka *Data Geek is my name*

Chapter 1: Python for Data Cleaning & Visualizations

Get started with Jupyter Notebook and clean messy data with ease!

🎯 Goal of This Chapter:

Teach beginners how to set up Python using Jupyter Notebook, load a CSV file, and clean data using basic Pandas functions — all in a clear, beginner-friendly format

What you will learn in this chapter:

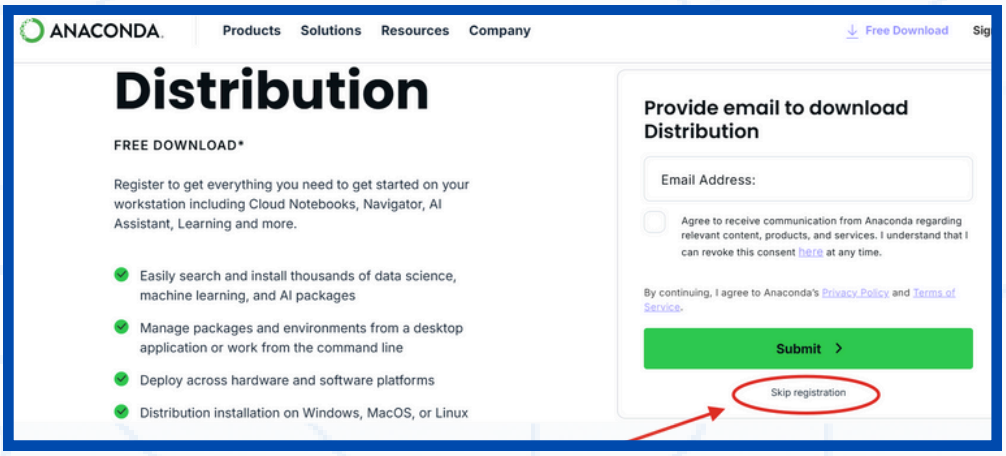
- How to install Anaconda Navigator to use and open Jupyter Notebook
- How to load a dataset using Pandas (library)
- How to remove null values and duplicates
- How to rename columns for clarity

1. Download Anaconda Navigator

a.) Install Anaconda Navigator to use Jupyter Notebook

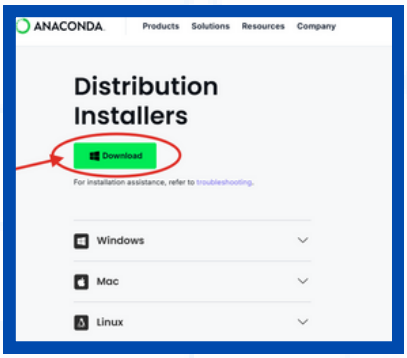
Download anaconda distribution:
https://www.anaconda.com/download

b.) On the distribution page: Enter email or you can click on “Skip Registration”

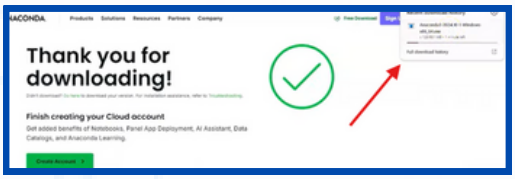


Chapter 1: Python for Data Cleaning & Visualizations

c.) Click the Download Button under “Distribution Installers.

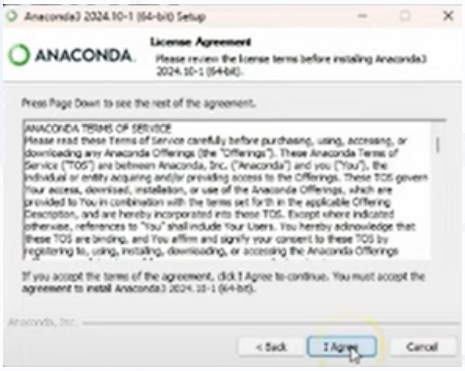


d.) Open downloaded file, shown on top right. Or go to your downloads file on your computer.



e.) On the Welcome window, click “**Next**” button.

f.) Read and Agree to License Agreement.



g.) In the next page “Select Installation Type” Under – Install for: click and select: **“Just Me (recommend)**

h.) Select the file location for installation on your computer. Click Browse and select your file location. You can leave it as is.

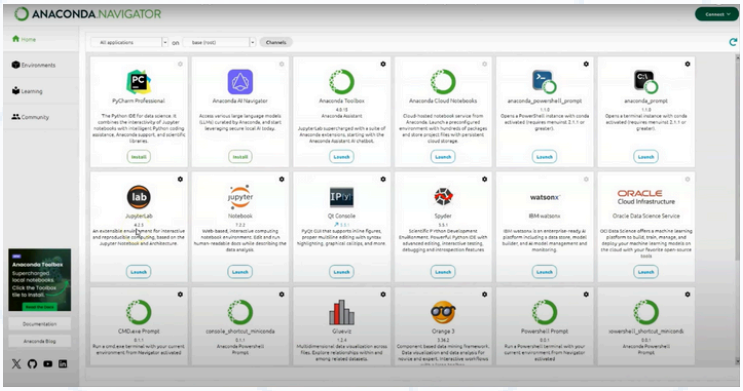
I.) Once installation is complete: Click on the next button.

Chapter 1: Python for Data Cleaning & Visualizations

j.) Continue by clicking next:

k.) Only select the “Launch Anaconda Navigator” check box.

l.) Anaconda Navigator” will launch. If not, you can search Anaconda Navigator in your search bar on your computer

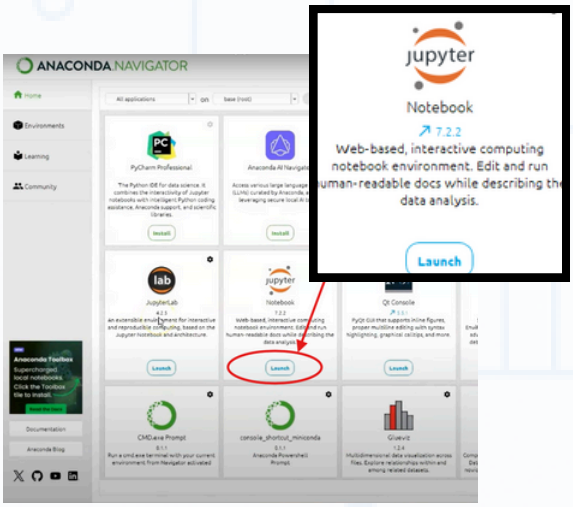


Chapter 1: Python for Data Cleaning & Visualizations

2. Launch Jupyter Notebook

a.) Open Anaconda Navigator

In the Anaconda Navigator, click the Jupyter Notebook tile to launch the application.



3. Get the Dataset and Save to your computer

a.) First You'll Need to Download the Sample Dataset

To follow along with the cleaning exercises, download the sample dataset provided below.

🔗 **Click to open:**
https://docs.google.com/spreadsheets/d/1oKXmVa8hqQeTIk2wsmD744nBhKlixY5Es6Wz0O5quz8/edit?usp=sharing

📥 **How to Download:**

- Click the link above to open the Google Sheet
- In the top menu, go to:
- File → Download → Comma Separated Values (.csv)
- Save the file as: **sample_sales_data.csv**

You will use this file in the next section to practice loading and cleaning data with python.

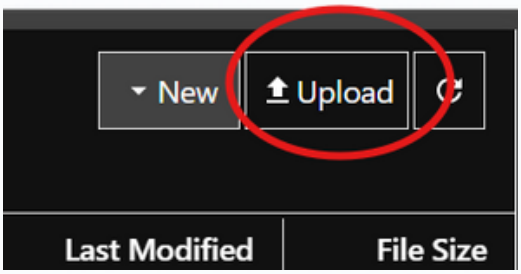
💡 **Tip: Save it in an easy-to-find folder on your computer so you can navigate to it from Jupyter Notebook later.**

4. Upload the dataset (CSV File) into Jupyter Notebook

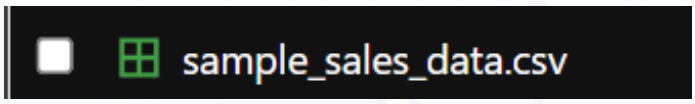
To read the dataset in Jupyter Notebook, the file must be in the **same folder** as your notebook.

Use the Jupyter File Upload Button

- Open Jupyter Notebook
- In the file browser (homepage), click the **Upload** button on the top right
- Choose the `**sample_sales_data.csv**` file from your computer
- Click **Upload** again to confirm



You should now see your CSV file listed next to your notebook file. In Jupyter Notebook Homepage/file location.

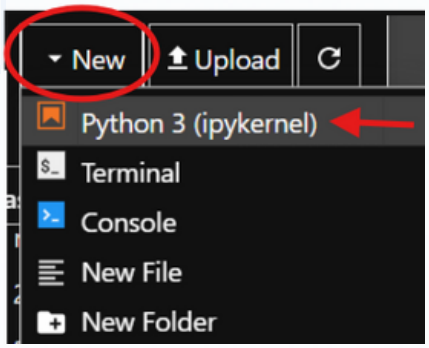


🎥 Want a walkthrough, and learn data cleaning?
Watch my YouTube tutorial where I clean this exact dataset using Pandas in Jupyter Notebook:
🔗 [IClick Here]

5. Open a New Workbook in Jupyter Notebook

a.) Create a New Notebook

From the homepage of Jupyter Notebook, click on the "**New**" button on the top right and select Python 3 (ipykernel)

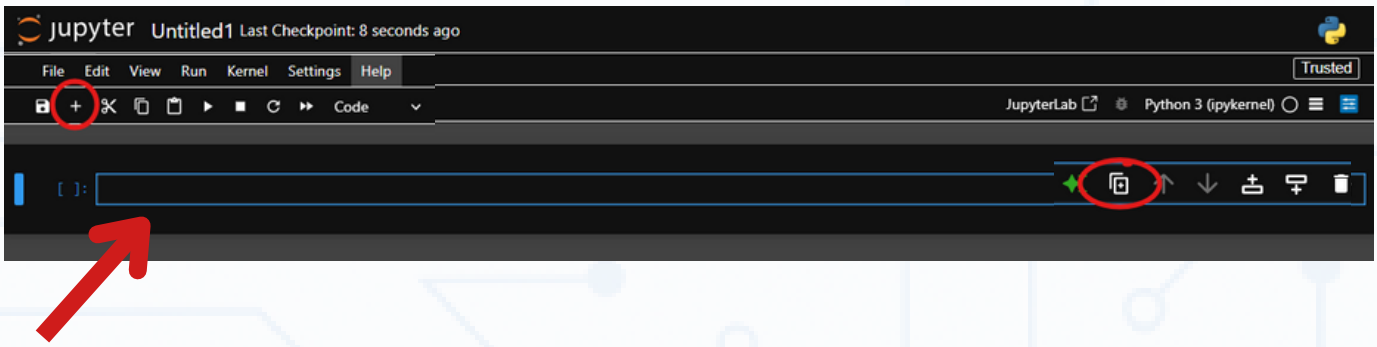


b.) You're now inside a new notebook in Jupyter and ready to start coding in Python.

You'll see a blank box called a "cell" — this is where you'll write your code.

To add more cells:

Click the + button at the top toolbar, or use the icon that appears near an existing cell



This box is a called a "cell", and where you will code

6. Upload the Dataset in Jupyter Notebook

Now that you have the file downloaded, lets load it into Python using Pandas library.

What is Pandas? 

Pandas is a powerful library used for working with **structured data** – like spreadsheets and datasets.

With Pandas, you can:

Load data from CSV or Excel files

- **Clean messy datasets** (remove duplicates, fill missing values)
- Filter, sort, and **analyze** your data

Group and summarize data (like pivot tables!)

Pandas is one of the most important tools in data analytics – and it works hand-in-hand with Jupyter Notebook to help explore and prepare data with just a few lines of code.

You will use Pandas through out this eBook to:

- Import and inspect your dataset
- Clean and transform the data
- Prepare it for SQL queries or Power BI dashboards

a.) Import the Pandas Library

Type this line of code into the first cell in your New Jupyter Notebook.

```
import pandas as pd
```

This imports your pandas libraries. Make sure to hit the run or play button, ▶ or use the short cut in Windows: ***Ctrl + Enter*** at the same time.

NOTE: Make sure the .csv file is saved in the same folder as your Jupyter Notebook & move on to the next step. We did this in step-4.

Chapter 1: Python for Data Cleaning & Visualizations

b.) Load Dataset

Next will will use Python Code to upload the dataset. This will load your dataset into the Notebook to view and clean. Type:

```
df = pd.read_csv("sample_sales_data.csv")
```

Hit the button ▶ to run the code, or ***Ctrl + Enter*** (in windows) at the same time. Continue to hit run every time you enter a new code.

This will load your dataset into the Notebook to view and clean. Here's a breakdown of this code:

df — This stands for dataframe. A dataframeis like a virtual spreadsheet – rows and columns – that lives inside python.

= — This is the assignment operator. It tells Python: "Take what's on the right and store it in the variable on the left.

read_csv() — This is a **Pandas function** that reads a .csv)Comma Separated Values) file and turns it into a dataframe (table)

"sample_sales_data.csv" — This is the **name of your dataset**. It should match exactly with the file name on your computer. If you have a different name for your dataset, make sure it is exactly how it is spelled.

Final Summary:

This line of code:

- Loads the dataset
- Converts it into a usable Python table (dataframe)
- Stores it in a variable called **df** so you can work with it.

7. Preview the Dataset

To follow you've loaded your data, it's always a good idea to take a look at the first few rows of the dataframe. Type this and hit run.

```
df.head()
```

This powerful Pandas command shows you the **first 5 rows** of your dataset by default.

What does 'head() do?

- It's a quick way to inspect the size and shape of your dataframe
- Helps you understand column names, data types, and examples of the data
- It's useful for catching missing values or any format issues

If you want to see more rows, just add a number inside the parentheses, like:

```
df.head(10)
```

8. Clean the Dataset with Pandas

Once your data is loaded, the next step is to clean it.

Data cleaning is one of the most important steps in the data analytics process. Before you can analyze or visualize anything, your dataset must be accurate, consistent, and free from errors. Cleaning ensures that the insights you discover are reliable, your visuals make sense, and your decisions are based on quality data — not noise or mistakes.

a.) Drop Missing Values

```
df.dropna(inplace=True)
```

This removes any rows where one or more cells are empty.

b.) Drop Duplicate Rows

```
df.drop_duplicates(inplace=True)
```

This removes any repeated rows in the dataset to keep it clean

c.) Rename Columns

```
df.rename(columns={"OldName": "NewName"}, inplace=True)
```

This renames any name columns. You can make column names clear and consistent by renaming them.

🧐 9. Inspect the Dataset Structure

While cleaning your data, it's helpful to understand:

- How many rows and columns it has
- What types of data are in each column
- Whether any columns are missing values

a.) View Column Names

```
df.columns
```

This shows a list of all the column names only within your dataset

b.) View Dataset Shape

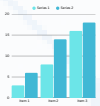
```
df.shape
```

This tells you (number of rows, number of columns) - for example, (120, 10) means 120 rows and 10 columns

Why It Matters

- You can catch any typos or extra spaces in column names
- Let's you identify which columns might need cleaning
- Prepares you for working with SQL, charts, or dashboards

Chapter 1: Python for Data Cleaning & Visualizations



10. Summarize the Data with Pandas

a.) Describe the Dataset

df.describe()

After cleaning your data, it's time to start **exploring** it with some basic statistics.

📈 This function gives you a statistical summary of all numeric columns:

- Count, mean, standard deviation
- Minimum and maximum values
- 25%, 50%, and 75% percentiles

b.) Count Unique Values

df['Product'].value_counts()

🔢 This shows how many times each unique value appears in a column. Great for analyzing categories like products, regions, or customer types. Remember df is what we named the variable to the dataset in step 6b (load the dataset) and ['Product'] is the column name in your dataset.

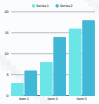
c.) Get Column-Wise Null Value Counts

df.isnull().sum()

🩹 Quickly check on how many missing values are in each column — even after cleaning. It shows each column name with each number of null values within that column.

Pro Tip: Try using the .head() to preview any result after you made changes. This helps you inspect without overwhelming your screen.

Chapter 1: Python for Data Cleaning & Visualizations



11. Visualize Your Data with Pandas

Now that your data is clean, let's use Python to create simple charts directly from your dataframe.

To display plots inside Jupyter Notebook, always run this library first:

```
import matplotlib.pyplot as plt
```

This tells Jupyter to render charts inside the notebook cells.

a). Create a Bar Chart

This creates a bar chart of how many times each product appears. Using our sample_sales_data.csv file. This shows purchases by item.

```
country_sales = df.groupby("Items Purchased")["Purchase Amount"].sum().sort_values()

plt.figure(figsize=(10, 6))
country_sales.plot(kind='barh', color='skyblue')
plt.title("Total Purchase by Items Purchased ")
plt.xlabel("Purchase Amount")
plt.ylabel("Items")
plt.tight_layout()
plt.show()
```

b). Create a Pie Chart

Pie charts are great for showing category proportions — like region breakdowns. Using our sample_sales_data.csv file. Purchases by Season is used:

```
season_counts = df["Season"].value_counts()

plt.figure(figsize=(6, 6))
plt.pie(season_counts, labels=season_counts.index, autopct='%1.1f%%', colors=['#66b3ff','#ff9999','#99ff99'])
plt.title("Season Sales Breakdown") # this is the tile of your visual
plt.axis('equal')  # Keeps the pie chart circular
plt.show()
```

Tip: Explore other column names in these visualizations. To see other types of analysis.

Chapter 1 Wrap-Up: Python for Data Cleaning & Visualizations

Great work! 🎉 You've just taken your first big step into data analytics.

Here's what you accomplished:

- Set up Jupyter Notebook and installed Python using Anaconda
- Loaded and explored your first dataset with Pandas
- Cleaned the data by removing nulls, duplicates, and renaming columns
- Summarized the dataset and created simple visualizations

What's Next?

In the next chapter, we'll switch gears and dive into SQL — a powerful language for working with databases.

You'll learn how to:

- Query data
- Filter results
- Combine multiple tables using JOINs
- And even connect SQL to Python and Power BI later

💡 Tip: Practice your Python steps a couple more times to build muscle memory. You're doing amazing!

Let's move on to Chapter 2: SQL for Beginners →

Chapter 2: SQL for Beginners

Start writing your first database queries with ease!

What is SQL?

SQL (Structured Query Language) is the language used to communicate with **relational databases**.
It allows you to:

- Access and extract specific rows and columns
- Filter results based on conditions
- Combine data from multiple tables using **JOINs**
- Sort, group, and summarize your data

Why Learn SQL?

- It's the **#1 skill** used by analysts, marketers, and data scientists
- It helps you work with real-world business data stored in databases
- It connects seamlessly with **Python, Power BI**, and other tools

In This Chapter, You'll Learn:

- How to set up a free SQL workspace
- How to write your first SQL query
- How to filter, sort, and join data
- How to use real business scenarios to practice

Chapter 2: SQL for Beginners

🛠️ Set Up Your Free SQL Workspace

You don't need to install anything to start practicing SQL.
We'll use an easy, web-based tool called:

👉 SQL Fiddle: https://sqlfiddle.com/
or
👉 Mode SQL Editor: https://mode.com/sql-tutorial/
or
👉 DB Fiddle: https://www.db-fiddle.com/

📄 Steps to Get Started:

1. Visit one of the links above in your browser
2. Choose **MySQL** or **PostgreSQL** as the database engine
3. Paste in the sample SQL script (we'll provide it next)
4. Click "**Build Schema**" or "**Run**" to create your test database
5. Start writing SQL queries in the right-side editor

Note: If you want to download PostgreSQL using PgAdmin. Here is a step-by-step video on how to download it here: https://youtu.be/IcJvVUln-KY?si=9z2yGJ_rERqy03je

💡 Why Use Online Editors?

- No downloads or setup required
- Practice instantly from any computer
- Great for beginners to test and learn without breaking anything

Coming Up Next

We'll give you a ready-to-go SQL script so you can build your first table and run your first query!

Chapter 2: SQL for Beginners

📋 Create a Sample Table and Run Your First SQL Query

Now that your SQL environment is ready, let's create your first table and explore it with a real query.

🏗️ Step 1: Paste this SQL script to build a table named sales_data

```
CREATE TABLE sales_data (
   id INT,
   customer_name VARCHAR(50),
   product VARCHAR(50),
   quantity INT,
   price DECIMAL(10, 2),
   sale_date DATE
);

INSERT INTO sales_data VALUES
(1, 'Alice', 'Laptop', 1, 899.99, '2024-05-12'),
(2, 'Bob', 'Tablet', 2, 299.50, '2024-05-13'),
(3, 'Carlos', 'Laptop', 1, 899.99, '2024-05-14'),
(4, 'Dina', 'Monitor', 2, 199.99, '2024-05-15'),
(5, 'Eli', 'Tablet', 1, 299.50, '2024-05-16');
```

✅ **This script:**

- Creates a table called sales_data
- Adds 5 sample sales records for practice

Chapter 2: SQL for Beginners

Step 2: Run Your First SQL Query

Now that you created your table it is time to view the data. Type or paste this in your editor:

```
SELECT * FROM sales_data;
```

🧠 This tells SQL:
Show me **all (*)** the rows and columns from the **sales_data** table.
You should see all 5 records in a table view once you run the query.

🎉 That's it! You just ran your first SQL query.
Next, we'll show you how to filter data using **WHERE**.

Chapter 2: SQL for Beginners

🔍 Filter Your Data with SQL

The `**WHERE**` clause allows you to filter rows based on a specific condition.

Syntax:

SELECT **column_name**
FROM **table_name**
WHERE **condition**;

✅ Example 1: Filter by Product sql

```
SELECT *
FROM sales_data
WHERE product = 'Tablet';
```

This query returns only the rows where the product is a **Tablet**.

✅ Example 2: Filter by Quantity

```
SELECT *
FROM sales_data
WHERE quantity > 1;
```

This returns all sales where the customer bought **more than 1 item.**

✅ Example 3: Combine Conditions

```
SELECT *
FROM sales_data
WHERE product = 'Tablet'
AND quantity > 1;
```

You can use **AND / OR** to combine multiple filters.

💡 **Tip:** SQL is case-insensitive, but it's good practice to write keywords like SELECT, WHERE, and FROM in uppercase for readability.

Chapter 2: SQL for Beginners

📊 Sort & Limit Your SQL Results

Now that you can filter data, let's organize your results using **ORDER BY** and **LIMIT.**

🔽 Sort Results with `ORDER BY`

```
SELECT *
FROM sales_data
ORDER BY price DESC;
```

This query sorts the data by price from highest to lowest.

- Use **ASC** for ascending order (default)
- Use **DESC** for descending order

🔢 Limit the Number of Results

```
SELECT *
FROM sales_data
LIMIT 3;
```

This shows only the first 3 rows of the result.

🔄 Combine Sorting + Limit

```
SELECT *
FROM sales_data
ORDER BY quantity DESC
LIMIT 2;
```

This gives you the top 2 sales with the highest quantity.

💡 **Tip:** Sorting and limiting your results helps you focus on top-selling products, highest values, or quick previews.

Chapter 2: SQL for Beginners

📊 Combine Tables with SQL JOIN

The **JOIN** clause merges data from multiple tables — allowing you to analyze related information together.

🔗 Let's say you have another table called customers:

```
CREATE TABLE customers (
   id INT,
   customer_name VARCHAR(50),
   country VARCHAR(50)
);

INSERT INTO customers VALUES
(1, 'Alice', 'USA'),
(3, 'Carlos', 'Canada'),
(4, 'Dina', 'USA'),
(5, 'Eli', 'UK');
```

🔗 Combine Tables with an **INNER JOIN**

```
SELECT s.*, c.country
FROM sales_data s
JOIN customers c
ON s.customer_name = c.customer_name;
```

💡 This tells SQL: "Combine rows from both tables where the **customer_name** values match."

- The result will show only records that have a match in both tables
- You can also try other **JOIN** types like:
  - **LEFT JOIN** → keeps all records from the left table
  - **RIGHT JOIN** → keeps all records from the right table

FULL JOIN → keeps all records from both

We will explain the **INNER JOIN** more clearly, line by line on the next page...

Chapter 2: SQL for Beginners

🧠 Explaining the INNER JOIN Line by Line

```
SELECT s.*, c.country
FROM sales_data s
JOIN customers c
ON s.customer_name = c.customer_name;
```

🔹 **sales_data s**
This creates a short alias for the sales_data table.
Instead of typing sales_data each time, we can just use **s**.

🔹 **s.***
This means:
“Select all columns from the sales_data table.”
Because we aliased it as s, we use s.* instead of just *.

🔹 **c.country**
This means:
“Select only the country column from the customers table.”
We’re not pulling all columns from customers, just the country field.

🔹 **ON s.customer_name = c.customer_name**
This is the JOIN condition — it tells SQL:
“Only combine rows where the customer_name in both tables match.”

✅ So the result will show:

- All fields from sales_data (s.*)
- The country field from customers
- Only for customers that appear in both tables

Chapter 2: SQL for Beginners

📊 Summarize Your Data with GROUP BY

SQL lets you group data and apply **aggregate functions** like:

- **SUM()** — total values
- **AVG()** — average values
- **COUNT()** — number of records

Example 1: Total Sales by Product

```
SELECT product, SUM(price) AS total_sales
FROM sales_data
GROUP BY product;
```

🧠 This groups all rows by product, and adds up the total price for each one.

Example 2: Average Sale Price by Product

```
SELECT product, AVG(price) AS average_price
FROM sales_data
GROUP BY product;
```

💡 This helps compare the average sale price for each product type.

Example 3: Count How Many Times Each Product Was Sold

```
SELECT product, COUNT(*) AS sales_count
FROM sales_data
GROUP BY product;
```

This is a quick way to find the most popular items by count.

✅ **GROUP BY** is perfect for:

- Sales by region or product
- Customer activity summaries
- Comparing performance between categories

Chapter 2: SQL for Beginners

🧠 Add Conditional Logic with “CASE” and “WHEN”

The `**CASE**` statement lets you create **if-then logic** directly in your SQL query.

📝 Example: Label Sales as “High” or “Low”

```
SELECT
   customer_name,
   product,
   price,
   CASE
      WHEN price > 500 THEN 'High Sale'
      ELSE 'Low Sale'
   END AS sale_category
FROM sales_data;
```

🔍 How It Works:

- **CASE** checks each row
- If price > 500, it returns 'High Sale'
- If not, it returns 'Low Sale'
- The result appears as a new column called sale_category

💡 Common Uses for CASE:

- Label customer types (e.g., 'New' vs. 'Returning')
- Categorize sales as 'Low', 'Medium', 'High'
- Handle missing or unexpected values

Chapter 2: SQL for Beginners

✅ Chapter 2 Wrap-Up: SQL for Beginners

Great job! 🎉 You now know the fundamentals of writing SQL queries — a skill that powers real-world business analytics.

🔍 Here's What You Learned:

- How to write and run your first SQL query using `SELECT` and `FROM`
- How to filter results with `WHERE`
- How to sort and limit your data using `ORDER BY` and `LIMIT`
- How to combine tables using `JOIN`
- How to summarize with `GROUP BY`, `SUM`, `COUNT`, and `AVG`
- How to apply logic using `CASE WHEN`

🚀 What's Next?

In the next chapter, you'll learn how to:

✔ Import and explore data inside **Power BI**
✔ Build visual dashboards
✔ Connect Power BI to SQL and Python
✔ Tell powerful data stories with real business insights

🧠 *Keep practicing with the sample tables and try new queries — the more you write, the more fluent you'll become in SQL!*

Let's dive into Chapter 3: Power BI for Visualization →

Chapter 3 – Power BI for Visualization

📊 Chapter 3: Power BI for Visualization

Let’s turn cleaned and organized data into stunning dashboards using **Power BI** — one of the most powerful tools in data analytics today.

💡 What is Power BI?

- Power BI is a free business intelligence tool from Microsoft that lets you:
- Import and transform data from Excel, CSV, SQL, or Python
- Create **interactive dashboards** and **visual reports**
- Share insights with teams, clients, or the public

🧠 Why Learn Power BI?

✅ In-demand skill for analysts, business pros, and freelancers
✅ Combines visuals + data modeling without needing code
✅ Easily connects to SQL databases, Excel files, or CSVs
✅ Free to use with **Power BI Desktop**

🟦 In This Chapter, You'll Learn:

- How to install and open Power BI Desktop
- How to import a CSV or Excel dataset
- How to create bar, pie, and line charts
- How to build a real-world dashboard
- How to publish and share your visualizations

Chapter 3 – Power BI for Visualization

🛠️ Install Power BI Desktop

Power BI Desktop is the free app you'll use to create reports and dashboards.

✅ Step-by-Step Setup

- Go to the official site: https://powerbi.microsoft.com/desktop/
- Click the yellow **"Download free"** button
- Choose the **Microsoft Store version** or scroll down to download the .exe file
- Install Power BI Desktop like any other app
- Launch Power BI Desktop from your Start menu or desktop shortcut

⚠️ System Requirements

- Windows 10 or later
- 64-bit system recommended
- Not available for Mac (use Parallels or Bootcamp if needed)

💡 **Tip:** If you're using a **company laptop**, you might need admin rights to install. Use the **web version of Power BI** (limited) at:
🔗 **https://app.powerbi.com**

** **Want to watch a video instead**? Click here on how to download Power BI Desktop for Free Step-by-Step:
https://youtu.be/udXoVb5zACc?si=lBLJBSN7rim06Cf_

Chapter 3 – Power BI for Visualization

📥 Import an Excel File into Power BI

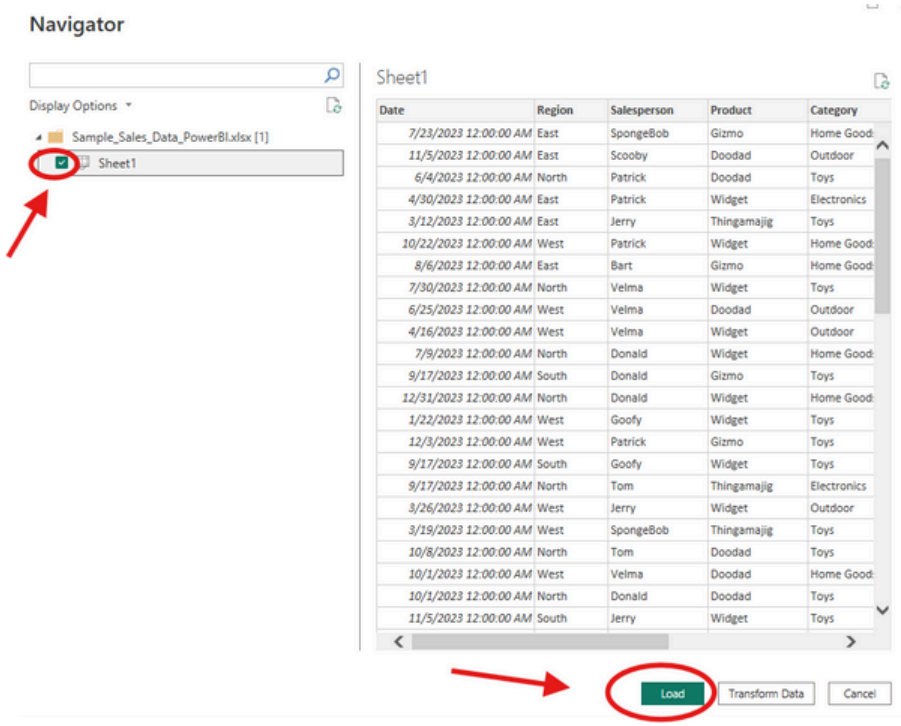
Let's bring in your dataset so you can start creating charts and dashboards.

✅ Step-by-Step Guide:

- Open **Power BI Desktop**
- On the **Home** tab, click **"Get Data"**
- Choose **Excel** from the list of data sources
- Find and select your Excel file (e.g., `sample_sales_data.xlsx`)
- Click "**Load**" to bring the data into Power BI

📊 Load the dataset into Power BI

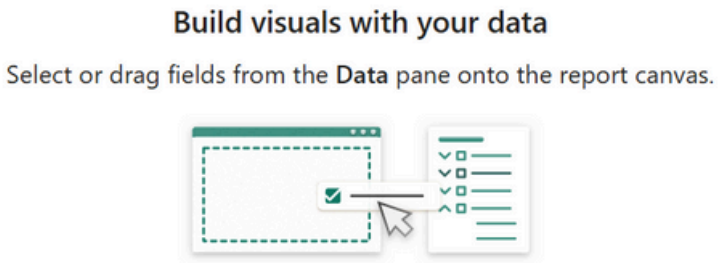
A Navigator window will popup. Here it shows an example of your data. On the left side it shows the dataset (file) name and under it with a box to check is the names of your tabs within your dataset. Select the tab you want to upload and click the load button.



Chapter 3 – Power BI for Visualization

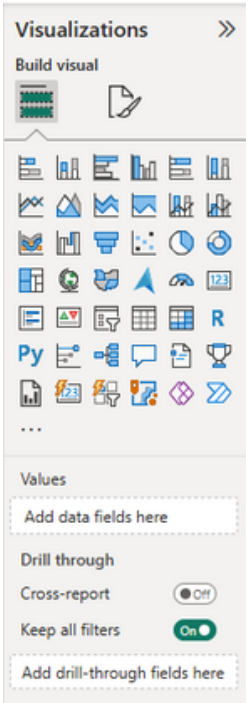
📊 What You'll See:

You see a "Build visuals with your data. This is a blank dashboard and this is where you will start creating your dashboard from your dataset.



📊 Build Visuals

On the "Right Panel" called "Visualizations" these are all your graphs. You can click on one and it will appear on the blank dashboard or you can click and drag to where you want to place it on the board.



Chapter 3 – Power BI for Visualization

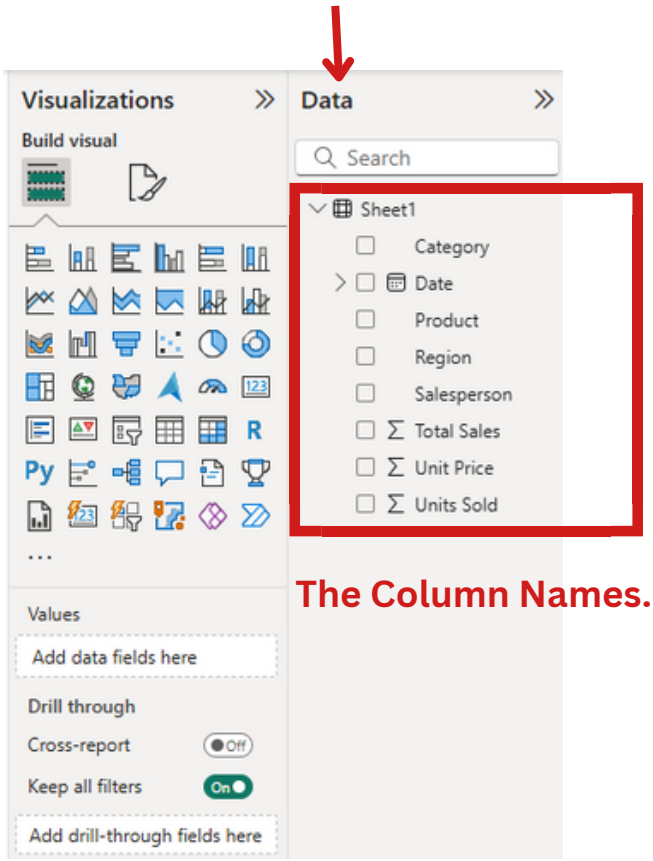
📊 Dataset columns

On the “Right Panel” next to your “Visualizations” you’ll see “Data”. This is your dataset you uploaded. You will see “Sheet 1”. This is the name of the sheet within your dataset.

Next to Sheet1 press the < to expand the name “Sheet1”. You will see all of your column names.

Here is where you can drag to the “Values” and/or Drill through under each Visualization.

We will show that step next.



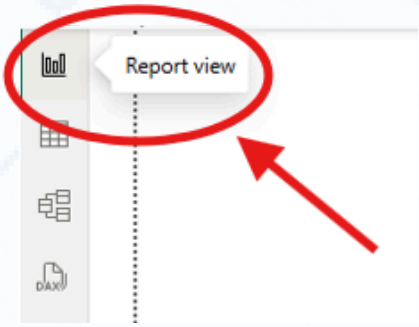
Chapter 3 – Power BI for Visualization

📊 Create Your First Bar Chart in Power BI

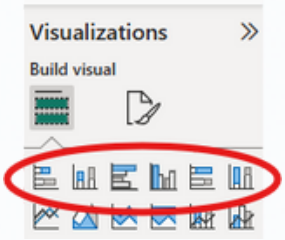
Let’s turn your Excel data into a clear and beautiful bar chart in just a few clicks.

✅ Step-by-Step Instructions:

1. Go to the **Report view** (the icon with a chart on the left sidebar)

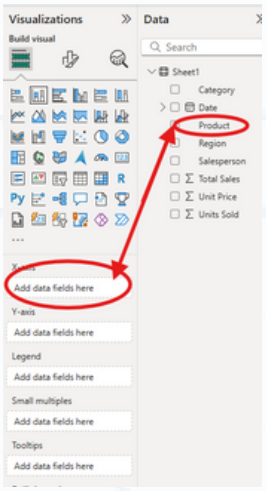


2. Click on the **Bar chart icon** in the visualizations pane. Any bar chart is okay.



3. Drag a **categorical field** (like `Product`) into the **X-axis (Axis)** box

4. Drag a numeric field (like `Amount` or `Price`) into the Y-axis (Values) box



Chapter 3 – Power BI for Visualization

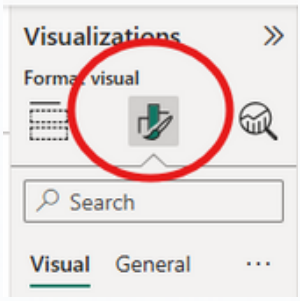
💡 **Example:**

- **Axis** → `Product`
- **Values** → `SUM of Price` or `SUM of Amount`

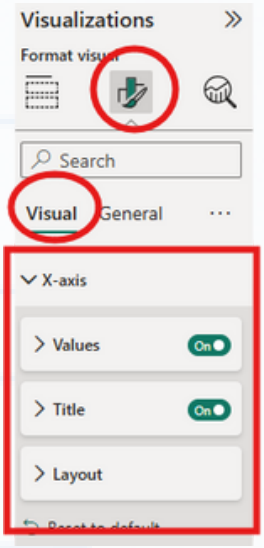
  - This will show how much revenue came from each product.

🎨 **Customize the Chart:**

- Click on the chart →
- go to the **paint roller icon** (Format)



- Change bar colors, labels, and title
- Enable **Data Labels** to show numbers on the bars



🎉 *You've now created your first visual in Power BI!*
Up next: try a Pie Chart, Line Chart, or Card for summary values

Chapter 3 – Power BI for Visualization

🥧 Create a Pie Chart & Summary Card in Power BI

Let's add more visuals to your dashboard by creating a pie chart and a card.

✅ Create a Pie Chart

1. Click the **Pie chart icon** in the Visualizations pane
2. Drag a **category field** (e.g., `Region` or `Product`) into the **Legend** area
3. Drag a **numeric field** (e.g., `Amount` or `Quantity`) into the **Values** area area

💡 Now you'll see a pie chart showing how each category contributes to the total.

📋 Create a Summary Card

1. Click the **Card icon** in the Visualizations pane (looks like a single number box)
2. Drag a **numeric field** (e.g., `Amount`) into the **Fields** area

- ✅ You now have a clean, bold number showing the total sales, revenue, or quantity.

🎨 Customize Your Visuals:

- Change colors, labels, or fonts using the **Format (paint roller)**
- Add a title to each visual using the **General → Title** option
- Turn on **Data Labels** to show percentages or values

🎉 *You've just added two powerful visuals to your report!*
Coming up next: Combine them into a full dashboard layout.

Chapter 3 – Power BI for Visualization

📊 Build & Publish Your First Power BI Dashboard

You've created several visuals — now let's arrange them into a clean, interactive dashboard and publish it online.

🧩 Step 1: Build Your Dashboard Layout

- Click and drag your visuals (bar chart, pie chart, card, etc.) to arrange them on the canvas
- Resize each visual for clarity and balance
- Use the **"+ Text box"** or **"Shapes"** tools to add section labels or headers (Insert tab) at the top and select "Text Box".

🎨 Design Tips:

- Use consistent color themes for a professional look
- Place summary cards at the top
- Group related visuals side by side (e.g., sales by product next to sales by region)

☁️ Step 2: Publish to Power BI Service

1. Click **"File → Publish → Publish to Power BI"**
2. Sign in with your Microsoft account
3. Choose **My Workspace** or create a new one
4. Click **"Select"** to upload

Your report will now be live online — ready to share!

🔗 Step 3: Share Your Dashboard

- Log into **https://app.powerbi.com**
- Open your dashboard
- Click "**Share**" to generate a link or invite collaborators

🎉 *Congratulations —*
you've built your first interactive Power BI dashboard and published it to the web

Chapter 3 – Power BI for Visualization

✅ Chapter 3 Wrap-Up: Power BI for Visualization

Well done! 🎉 You've now built and published your first interactive data dashboard using Power BI.

📌 Here's What You Learned:

- How to install and open Power BI Desktop
- How to import data from Excel
- How to create bar charts, pie charts, and summary cards
- How to format visuals and arrange them into a full dashboard
- How to publish and share your report online

🧠 Why This Matters

Power BI helps you:

- Turn raw data into decision-ready visuals
- Communicate your insights to teams or clients
- Build dashboards you can update monthly or automate with SQL/Python

🚀 What's Next?

In the next chapter or project, you'll discover how to:

- **Combine SQL, Python, and Power BI together**
- Automate reporting with real-time data
- Present your work like a pro analyst

💡 **Take time to explore Power BI's other visuals and features — bookmarks, slicers, maps, and more.**

You're officially dashboard-ready. 👏 Let's move forward to the final chapter!

Chapter 4:
Combine Python + SQL + Power BI for Visualization

🧩 Final Chapter: Combine Python + SQL + Power BI

Build a full data project from start to dashboard.

Now that you've mastered the fundamentals, it's time to combine everything you've learned into a **real-world workflow**.

💡 What You'll Do in This Chapter:

- Use **Python** to clean and prep your data
- Use **SQL** to query and filter cleaned data
- Use **Power BI** to build a dashboard from that query
- Automate the flow between tools

🛠️ Tools You'll Use:

- **Jupyter Notebook** (Python)
- **PostgreSQL** or a cloud SQL tool
- **Power BI Desktop**
- **CSV or Excel dataset**

🟦 What You'll Build:

A dynamic sales dashboard powered by real business data.

You'll:
✔ Clean and summarize the dataset in Python
✔ Load it into SQL and filter key insights
✔ Connect Power BI to your SQL database
✔ Visualize the final output in a clean dashboard

🎯 Let's bring everything together and show how the pros do it!

Chapter 4:
Combine Python + SQL + Power BI for Visualization

🧹 Step 1: Clean Your Data in Python & Load into SQL

To build a real-world workflow, we'll start by preparing the data in Python and moving it into a SQL database for querying.

✅ Part 1: Clean the Data in Python

```
import pandas as pd
from sqlalchemy import create_engine

# Load your dataset
df = pd.read_csv("sample_sales_data.csv")

# Clean it
df.dropna(inplace=True)
df.drop_duplicates(inplace=True)

# Optional: rename columns for clarity
df.rename(columns={
    'Amt': 'Amount',
    'Cust_ID': 'Customer_ID'
}, inplace=True)
```

💾 Part 2: Load into SQL (Using SQLite for Simplicity)

```
i# Create a connection to an in-memory SQLite
database
engine = create_engine('sqlite:///sales_data.db')

# Load dataframe into SQL table
df.to_sql("sales_data", engine, if_exists='replace',
index=False)
```

✅ You now have a SQL-accessible database that contains your cleaned data.

💡 If you're using PostgreSQL or MySQL instead of SQLite:
Change the connection string
Install the correct Python SQL driver (e.g., `psycopg2`)

Chapter 4:
Combine Python + SQL + Power BI for Visualization

Step 2: Query SQL from Python

Now that your cleaned data is loaded into a SQL table, let's use Python to run SQL queries directly and fetch insights.

✅ Query the SQL Table Using Pandas

```
import pandas as pd
from sqlalchemy import create_engine

# Reconnect to the same SQLite database
engine = create_engine('sqlite:///sales_data.db')

# Run a SQL query and store the result in a new
dataframe
query = """
SELECT product, SUM(Amount) AS total_sales
FROM sales_data
GROUP BY product
ORDER BY total_sales DESC
"""
sales_summary = pd.read_sql(query, engine)

# Preview the result
sales_summary.head()
```

🔍 What's Happening Here?

- You're writing SQL inside Python using triple quotes `"""..."""`
- The result is stored in a new dataframe called `sales_summary`
- You can use `.head()` or visualize it with Pandas or Power BI

💡 Pro Tip: You can write any SQL query — filters, joins, aggregations — and combine it with Python's plotting or data export tools.

Chapter 4:
Combine Python + SQL + Power BI for Visualization

Step 3: Connect Power BI to Your SQL Table

Let's bring your SQL-powered data into Power BI and build a visual dashboard directly from it.

✅ Option A: Connect Power BI to SQLite (via CSV)

SQLite is great for beginners, but Power BI doesn't support direct SQLite connections.

Instead, export your query result from Python as a CSV:

```
sales_summary.to_csv("sales_summary.csv", index=False)
```

Then in Power BI:

1. Click Home → Get Data → CSV
2. Load sales_summary.csv
3. Visualize it using bar charts, pie charts, cards, and filters

✅ Option B: Connect to PostgreSQL (Direct SQL)

1. If you're using PostgreSQL:
2. In Power BI, click Home → Get Data → PostgreSQL database
3. Enter your host (e.g., localhost) and database name
4. Input credentials and click Connect
5. Select the table (e.g., sales_data) and click Load

💡 Make sure you have the PostgreSQL ODBC connector installed

📊 Build Your Final Dashboard

- Use bar charts to show sales by product or region
- Use cards for total revenue or quantity sold
- Add slicers (filters) to explore results by category or date
- Format with titles, labels, and themes for a polished result

🎉 Your pipeline is complete: Python → SQL → Power BI
You've now built a fully integrated data workflow like a real analyst!

Chapter 4:
Combine Python + SQL + Power BI for Visualization

🎓 Final Wrap-Up: Your Data Analytics Journey

Congratulations! 🎉 You've just completed a full data analytics project — from raw data to dashboard — using **Python**, **SQL**, and **Power BI**.

✅ Here's What You've Accomplished:

- Cleaned and prepared data using **Python & Pandas**
- Queried and joined datasets using **SQL**
- Built and published dashboards in **Power BI**
- Learned how to connect all 3 tools into one seamless workflow

💡 What This Means for You:

You now have the skills to:
✔️ Apply for analyst roles
✔️ Build automated reporting solutions
✔️ Tackle real-world data challenges
✔️ Start freelancing or consulting

🚀 What's Next?

- Build a **portfolio project** using your own dataset
- Practice using **APIs** and external data sources
- Share your dashboards on **LinkedIn, GitHub, or your website**
- Keep refining your skills — this is just the beginning!

✨ *The best way to become a confident data analyst is to keep solving problems.*
Use this eBook as your blueprint — and build from here.

Stay curious. Keep coding.
— Veronica
aka Data Geek Is My Name

Chapter 4:
Combine Python + SQL + Power BI for Visualization

🎁 Bonus Section

Unlock Extra Value as a Thank You for Reading!

- ✓ 3 Real-World beginner-friendly project ideas,
- ✓ The tools and skills you'll use with step-by-step videos.
- ✓ Pandas and SQL Cheat Sheet - beginner friendly beginners list.

Stay in the loop — I share new tutorials weekly on:
🔗 YouTube

✨The best way to learn is to apply what you've read. Use these templates to practice, customize, and start building your data portfolio today!

3 Real-World Beginner Friendly Data Projects

1. **Sales Data:**
    a. **Source**: Sample CSV or Excel
    b. **Explore**: Total Sales, top products, customer segments
    c. **Story**: “Which product drives the most revenue - and why?”
2. **Movie Ratings**
    a. **Source**: IMDB or Kaggle.com
    b. **Explore**: Ratings by genre/year
    c. **Story**: “How have movie ratings changed over time?”
3. **Fitness Tracker**
    a. **Source**: Daily steps, heart rate
    b. **Explore**: Patterns over weeks/months”
    c. **Story**: “Did my activity imporve in the past 30 days?”

The tools and skills you'll use with step-by-step videos

Take your skills to the next level with these hands-on projects.

1 Clean a Messy Dataset with Python

Tools: Jupyter Notebook, Pandas
Skills: Data cleaning, null handling, renaming columns

Watch these: Install Anaconda for Jupyter Notebook: https://youtu.be/ImhtTEfQafo?si=L9Ptw1ecc-RWg_7q
Clean Data project: https://youtu.be/cWf08xuSqdU?si=pci6RzU1AXEwM8Hf (Use the dataset provided in the description of the video for free!)

2 Build a Sales Dashboard in Excel or Power BI

Tools: Excel or Power BI
Skills: Pivot tables, visualizations, business KPIs

Watch this: how to download Power BI (Free) and build your dashboard. Use the same dataset from Step 1. https://youtu.be/udXoVb5zACc?si=z1piourZtwElZtXq

3 Query a Database with SQL

Tools: pgAdmin or DBeaver
Skills: SELECT, WHERE, GROUP BY, JOIN

Walkthrough: Download PostgreSQL: https://youtu.be/IcJvVUln-KY?si=SS_Y5UkTJVUbrr-P

PYTHON PANDAS CHEAT SHEET

START HERE:

```
import pandas as pd
df = pd.read_csv('yourfile.csv')
```

DATAFRAME BASICS:

```
df.head()       # Preview top rows
df.info()       # Data types & nulls
df.describe()   # Stats summary
```

SELECTION:

```
df['Column']          # Select column
df[['Col1','Col2']]   # Multiple columns
df.loc[0]             # Row by index
```

COMMON METHODS:

```
df.dropna()          # Remove nulls
df.fillna(0)         # Fill nulls
df.sort_values('Col')        # Sort by column
df.groupby('Col').mean()     # Group & aggregate
```

SQL CHEAT SHEET

BASIC QUERY:

SELECT * FROM table;

FILTER DATA:

SELECT name FROM customers WHERE age > 30;

SORT RESULTS:

SELECT * FROM sales ORDER BY total DESC;

GROUP DATA:

SELECT region, SUM(sales) FROM sales GROUP BY region;

JOIN TABLES:

SELECT a.name, b.order_id FROM customers a JOIN orders b ON a.customer_id = b.customer_id;

- ✅ **Python Cheat Cheat**
- ✅ **SQL Queries Practice Guide**
- ✅ **Power BI download**

Next Steps:

Have fun exploring, watch the tutorials on my YouTube channel and continue learning by following a long and use all material I provide in each video.

Questions?

Connect with me here:

reply@datageekismyname.com