
Hey Everyone !
I am Bharath

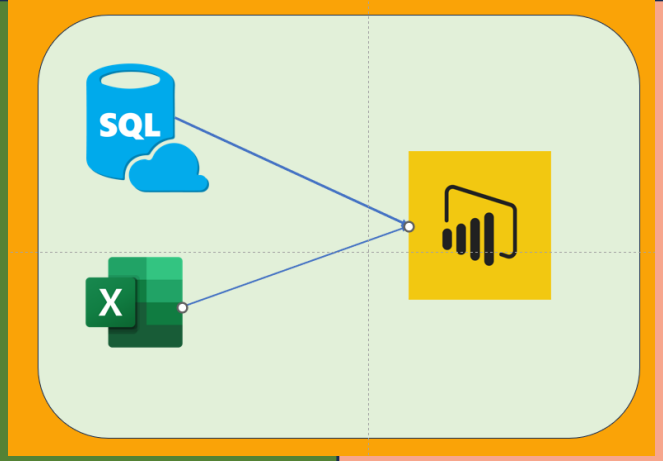
Recently, I have learned Python **Pandas** from **CodeBasics** and worked on a few hands-on **projects**,
I want to share my **insights** with you regarding what I understand about pandas and **key phases** of pandas for data analysis

Oh ! Not this one

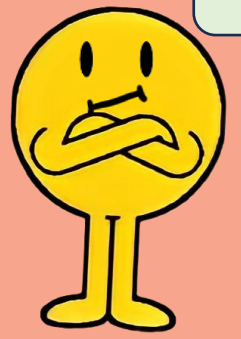
pandas

Python Pandas Library

Before Pandas



But..



Before Pandas, I mostly worked with **Excel, SQL, and Power BI** for data analysis.
They all have their own strengths:

Excel is great for quick analysis, and working with small datasets interactively.

SQL is awesome for extracting data from databases and performing basic aggregations or filters efficiently

Power BI is excellent for building dashboards, creating rich visualizations, and sharing insights with stakeholders.

Despite their strengths, I started noticing some limitations:

Excel struggles with large datasets, and cleaning or transforming data becomes time-consuming and error-prone

SQL is not designed for heavy data wrangling. Complex transformations or custom functions become hard to manage.

Power BI is great for visuals, but the data cleaning process using Power Query feels restrictive

That's when I discovered Pandas!

pandas

Why Pandas stands out:

So… **What is Pandas?**

•Pandas is a powerful **open-source Python library** built specifically for working with structured data like tables and time series.

•It allows you to **load, clean, transform, and analyze data** using simple, readable Python code.

•Think of it as a blend of **Excel's flexibility**, **SQL's power**, and **Power BI's structure**, all in a programmable format.

With Pandas, we can:

•Work with **large datasets** smoothly—millions of rows, no problem.

•**Automate** repetitive cleaning and transformation steps with code.

•Easily **combine data** from multiple sources (CSV, Excel, databases).

•Perform complex operations like **grouping, pivoting, and custom logic** in just a few lines.

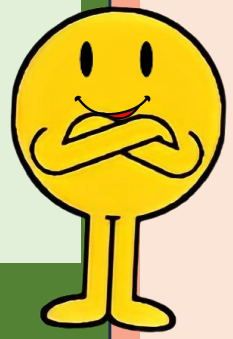
pandas

📦 Pandas Series

A **Series** is like a single column of data in Excel.
It's a **one-dimensional labeled array**—think of it as a column with index labels.

```
score_series = pd.Series([85, 92, 78])
score_series
```

```
0    85
1    92
2    78
dtype: int64
```



📊 Pandas DataFrame

A **DataFrame** is like a whole Excel table or SQL result.
It's a **two-dimensional labeled data structure**—with rows and columns.

```
data = {
    'Name': ['Bharath', 'Asha', 'Ravi'],
    'Score': [85, 92, 78]
}
df = pd.DataFrame(data)
df
```

	Name	Score
0	Bharath	85
1	Asha	92
2	Ravi	78

Alright! Now that we had a quick glimpse about Pandas

Let's talk about how we actually use it in real-life data analysis. Whether you're working on a sales report, customer insights, or predicting trends—data analysis usually goes through these **key phases**.

and we will Understand this through **"Atliq Hotels Project "**

Company Overview – AtliQ Grands

AtliQ Grands is a premium hospitality brand that owns multiple five-star hotels across major Indian cities like Delhi, Mumbai, Hyderabad, and Bangalore.

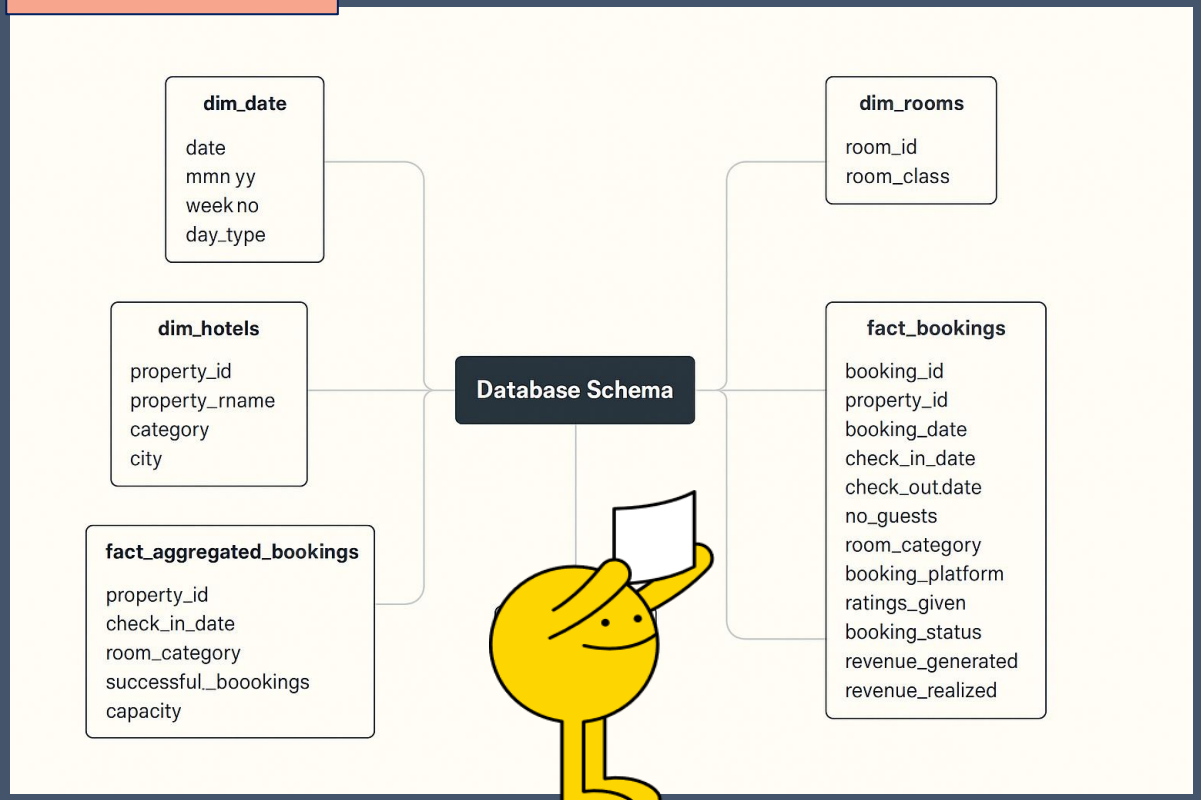
The hotels are categorized into two segments: **Luxury Hotels** and **Business Hotels**, catering to both elite leisure travelers and corporate guests.

Problem Statement :

Due to **strong competitor strategies** and **ineffective internal decision-making**, AtliQ Grands is witnessing a **decline in both market share and revenue**.
To counter this, the leadership team aims to integrate **Business and Data Intelligence** for:

- Smarter decision-making
- Optimized pricing strategies
- Enhanced guest experience

Table Schema



Unique Hotels

Property Name	Category
Atliq Grands, Atliq Exotica, Atliq Blu	Luxury
Atliq City, Atliq Bay, Atliq Palace, Atliq Seasons	Business

Unique Cities

Delhi , Mumbai, Hyderabad, Bangalore

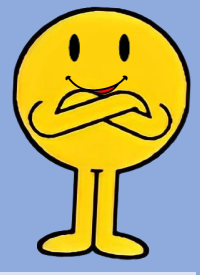
Unique Booking Platform

Direct Online, Logtrip, Tripster, Makeyourtrip, Journey, Direct Offline, Others

Unique Room Categories

Standard, Elite, Premium, Presidential

1. Loading the Data:



What it means:

This is the starting point. You bring your raw data into Python using Pandas—whether it's from a CSV, Excel file, database, even an API.
Or… you can **create your own DataFrame** for testing or small tasks.

Common methods:

pd.read_csv('file.csv')	# Load from CSV
pd.read_excel('file.xlsx')	# Load from Excel
df.head(), df.tail()	# Peek at the data
df.info(), df.shape	# Understand structure

Import pandas once installed

```
import pandas as pd
```

```
df_bookings = pd.read_csv('datasets/fact_bookings.csv')
df_date = pd.read_csv('datasets/dim_date.csv')
df_hotels = pd.read_csv('datasets/dim_hotels.csv')
df_rooms = pd.read_csv('datasets/dim_rooms.csv')
df_agg_bookings = pd.read_csv('datasets/fact_aggregated_bookings.csv')
```

```
df_agg_bookings.head(4)
```

Top 4 rows

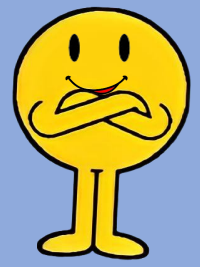
	property_id	check_in_date	room_category	successful_bookings	capacity
0	16559	1-May-22	RT1	25	30.0
1	19562	1-May-22	RT1	28	30.0
2	19563	1-May-22	RT1	23	30.0
3	17558	1-May-22	RT1	30	19.0

```
df_agg_bookings.shape
```

table has 9200 rows and 5 columns

```
(9200, 5)
```

🔍 2. Exploratory Data Analysis (EDA)



What it means:

EDA is all about getting familiar with the data—its structure, patterns, and quirks. You ask questions like:

- What are the column types?
- Are there missing values?
- What are the min/max/avg values?
- Are there outliers or weird entries?

Common methods:

df.describe()	# Summary statistics
df.dtypes	# Data types
df['column'].value_counts()	# Frequency count
df['column'].unique()	# Unique values
df.plot(), df.hist(), df.boxplot()	# Quick visuals

Summary Statistics

```
In [11]: df_agg_bookings.describe()
```

Out[11]:

	property_id	successful_bookings	capacity
count	9200.000000	9200.000000	9198.000000
mean	18040.640000	14.655761	25.280496
std	1099.818325	7.736170	11.442080
min	16558.000000	1.000000	3.000000
25%	17558.000000	9.000000	18.000000
50%	17564.000000	14.000000	25.000000
75%	18563.000000	19.000000	34.000000
max	19563.000000	123.000000	50.000000

Unique Values

```
In [12]: #Unique Property_ids
df_agg_bookings.property_id.unique()
```

```
Out[12]: array([16559, 19562, 19563, 17558, 16558, 17560, 19558, 19560, 17561,
       16560, 16561, 16562, 16563, 17559, 17562, 17563, 18558, 18559,
       18561, 18562, 18563, 19559, 19561, 17564, 18560], dtype=int64)
```

```
In [14]: #number of bookings per room category
df_agg_bookings.groupby("room_category")["successful_bookings"].sum()
```

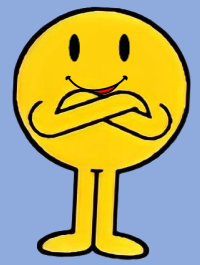
```
Out[14]: room_category
RT1    38651
RT2    49534
RT3    30566
RT4    16082
Name: successful_bookings, dtype: int64
```

```
In [15]: #days on which bookings are greater than capacity
df_agg_bookings[df_agg_bookings.successful_bookings>df_agg_bookings.capacity]
```

Out[15]:

	property_id	check_in_date	room_category	successful_bookings	capacity
3	17558	1-May-22	RT1	30	19.0
12	16563	1-May-22	RT1	100	41.0
4136	19558	11-Jun-22	RT2	50	39.0
6209	19560	2-Jul-22	RT1	123	26.0
8522	19559	25-Jul-22	RT1	35	24.0
9194	18563	31-Jul-22	RT4	20	18.0

3. Data Cleaning:



What it means:

Now you clean up the mess—fixing missing values, correcting data types, renaming columns, and handling duplicates or inconsistent entries.

Common methods:

df.dropna(), df.fillna(), df.drop_duplicates()	# Handle missing data
df['column'] = df['column'].astype('int')	# Convert column to integer type
df.replace({'old': 'new'})	# Replace values

```
'''
In aggregate bookings find columns that have null values.
Fill these null values with whatever you think is the appropriate subtitute
(possible ways is to use mean or median)

'''

df_agg_bookings.isnull().sum()
```

Counting null values column wise

```
property_id            0
check_in_date          0
room_category          0
successful_bookings    0
capacity               2
dtype: int64
```

```
df_agg_bookings[df_agg_bookings.capacity.isna()]
```

	property_id	check_in_date	room_category	successful_bookings	capacity
8	17561	1-May-22	RT1	22	NaN
14	17562	1-May-22	RT1	12	NaN

```
df_agg_bookings.capacity.median()
```

```
25.0
```

Filling null values with median

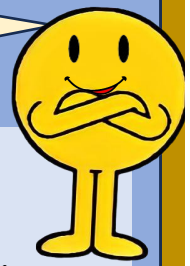
```
df_agg_bookings.capacity.fillna(df_agg_bookings.capacity.median(), inplace=True)
```

```
df_agg_bookings.loc[[8,15]]
```

	property_id	check_in_date	room_category	successful_bookings	capacity
8	17561	1-May-22	RT1	22	25.0
15	17563	1-May-22	RT1	21	25.0

isnull() is an **alias** for **isna()**, They both return a **Boolean** mask of the same shape as the input, where **True** indicates a missing value and **False** indicates a non-missing value

4. Data Transformation



What it means:

Now the data is clean, you mold it to fit your analysis needs—
Creating new columns,
Aggregating,
Reshaping,
Normalization,
Merging datasets, etc.

Common methods:

df['new_col'] = df['col1'] + df['col2']	# Create new columns
df.groupby('category').mean()	# Aggregation
pd.merge(df1, df2, on='id')	# Merge datasets
df.pivot_table(index='A', columns='B', values='C')	# Reshape
df.sort_values('column', ascending=False), df.apply(lambda x: ...)	# Custom transformation

```
#creating new column called occupancy percentage
df_agg_bookings['occ_pct'] =
df_agg_bookings.apply(lambda row: round((row['successful_bookings']/row['capacity'])*100,2), axis=1)


df_agg_bookings.head(5)
```

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct
0	16559	1-May-22	RT1	25	30.0	83.33
1	19562	1-May-22	RT1	28	30.0	93.33
2	19563	1-May-22	RT1	23	30.0	76.67
3	17558	1-May-22	RT1	30	19.0	157.89
4	16558	1-May-22	RT1	18	19.0	94.74

Here you can observe we use lambda function to apply the change row wise

What is a lambda function?

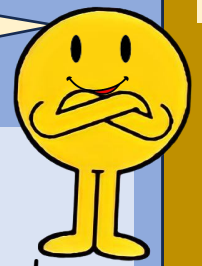
A **lambda function** is a **short, anonymous function** in Python.
You use it when you need a **quick one-liner function**—especially inside methods like .apply().
It's like a shortcut for writing simple functions!

In the above function

```
df_agg_bookings.apply(lambda row: round((row['successful_bookings']/row['capacity'])*100,2), axis=1)
```

lambda row	Takes each row of the DataFrame
The expression inside	Calculates occupancy percentage
axis=1	Tells Pandas to apply this function row by row
round(..., 2)	Rounds the result to 2 decimal places

5. Generating Insights



What it means:

This is where you ask deep questions and find meaningful patterns:

- Which product sells the most?
- What are the top customer segments?
- Is there a trend in sales over months?

You use summary tables, trends, and visualizations to communicate findings.

We will analyse insights with these following mini exercise

1. occupancy percentage by room category

```
# we can group occupancy percentage by room_class because they are from same table df_agg_bookings
df_agg_bookings.groupby("room_category")["occ_pct"].mean()
```

```
room_category
RT1    58.224247
RT2    58.040278
RT3    58.028213
RT4    59.300461
Name: occ_pct, dtype: float64
```

Occupancy % by room type

```
# To group occupancy percentage by room_class which is in df_room
# step 1 : first merge both df_agg_bookings and df_room tables and drop any common tables

new_df = pd.merge(df_agg_bookings, df_rooms, left_on="room_category", right_on="room_id")
new_df.drop("room_id",axis=1, inplace=True)
new_df.head(3)
```

Merge tables (like join in SQL)

	property_id	check_in_date	room_category	successful_bookings	capacity	occ_pct	room_class
0	16559	1-May-22	RT1	25	30.0	83.33	Standard
1	19562	1-May-22	RT1	28	30.0	93.33	Standard
2	19563	1-May-22	RT1	23	30.0	76.67	Standard

```
#step2 : calculate occupancy percentage by room_class
average_occupancy_rate = new_df.groupby("room_class")["occ_pct"].mean()
average_occupancy_rate
```

```
room_class
Elite           58.040278
Premium         58.028213
Presidential    59.300461
Standard        58.224247
Name: occ_pct, dtype: float64
```

Occupancy % by room category

2. Revenue realized per city

```
# Here revenue realized column is in df_bookings table and city column in df_hotels table

# step 1 : first merge both df_bookings and df_hotels tables and drop any common tables

df_merged = pd.merge(df_bookings, df_hotels, on="property_id")
df_merged.head(3)
```

Merge tables (like join in SQL)

checkout_date	no_guests	room_category	booking_platform	ratings_given	booking_status	revenue_generated	revenue_realized	property_name	category	city
2/5/2022	2.0	RT1	others	NaN	Cancelled	9100	3640	Atliq Grands	Luxury	Delhi
2/5/2022	4.0	RT1	direct online	5.0	Checked Out	10920	10920	Atliq Grands	Luxury	Delhi
3/5/2022	2.0	RT1	others	4.0	Checked Out	9100	9100	Atliq Grands	Luxury	Delhi

```
# step 2 : calcuate Revenue realized per city

df_merged.groupby("city")["revenue_realized"].sum()
```

```
city
Bangalore    420383550
Delhi        294404488
Hyderabad    325179310
Mumbai       668569251
Name: revenue_realized, dtype: int64
```

Revenue realized per city

3. Revenue realized per hotel

```
df_merged.groupby("property_name")["revenue_realized"].sum().round(2).sort_values()
```

```
property_name
Atliq Seasons     66086735
Atliq Grands     211462134
Atliq Bay        259996918
Atliq Blu        260851922
Atliq City       285798439
Atliq Palace     304081863
Atliq Exotica    320258588
Name: revenue_realized, dtype: int64
```

Revenue realized by hotel

4. Average rating per city

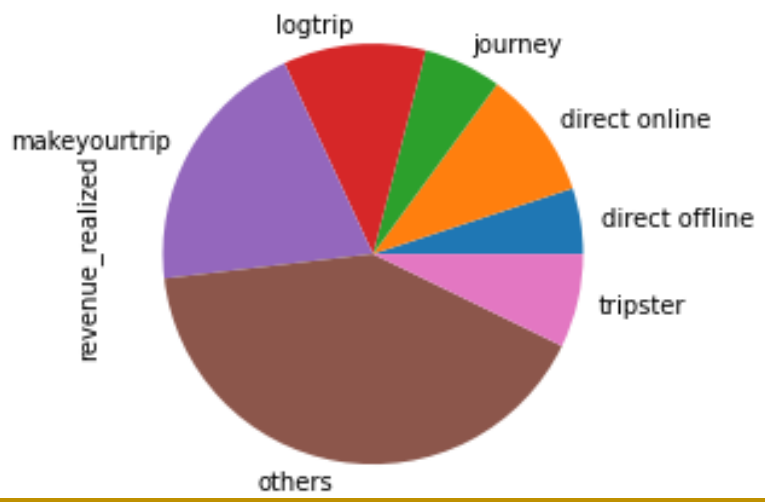
```
df_merged.groupby("city")["ratings_given"].mean().round(2)
```

```
city
Bangalore    3.41
Delhi        3.78
Hyderabad    3.66
Mumbai       3.65
Name: ratings_given, dtype: float64
```

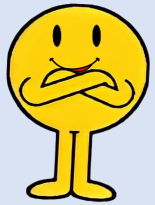
5. Visual representation of revenue realized per booking platform

```
import matplotlib.pyplot as plt

df_merged.groupby("booking_platform")["revenue_realized"].sum().plot(kind="pie")
plt.show()
```



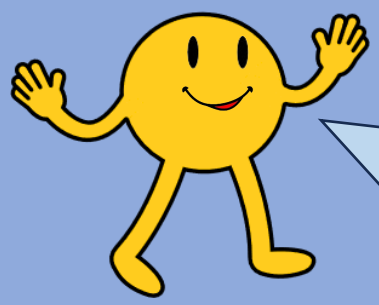
Matplotlib:



What it means:

Matplotlib is a popular Python library for **data visualization**.
It helps you **turn your numbers into visual stories**—like bar charts, line plots, pie charts, and more.
Think of it as the **PowerPoint of Python**—but for data! ✨

We will discuss more about matplotlib and data visualization soon..!



This project is part of code basics python course

Big thanks to **Dhaval Patel Sir, Hemanand Vadivel Sir**, and the **Codebasics** team

Amaresam Sai Bharath Chand