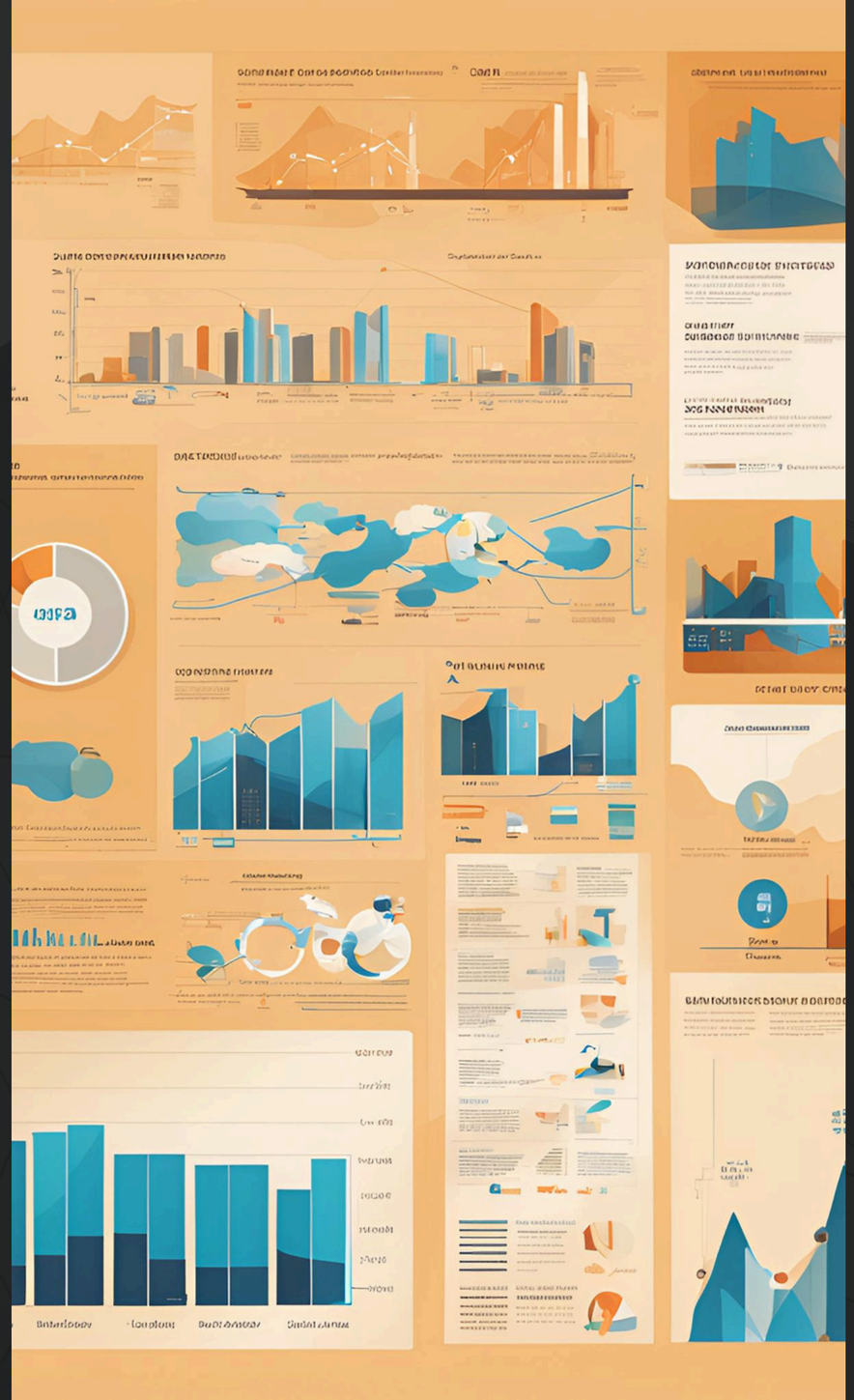


Leveraging Data Analysis with Pandas for Business Insights

PRESENTED BY: Rishita Giri

Introduction

Objective:

Provide a comprehensive analysis of sales data to derive actionable insights for informed business decisions.

Tools Used:

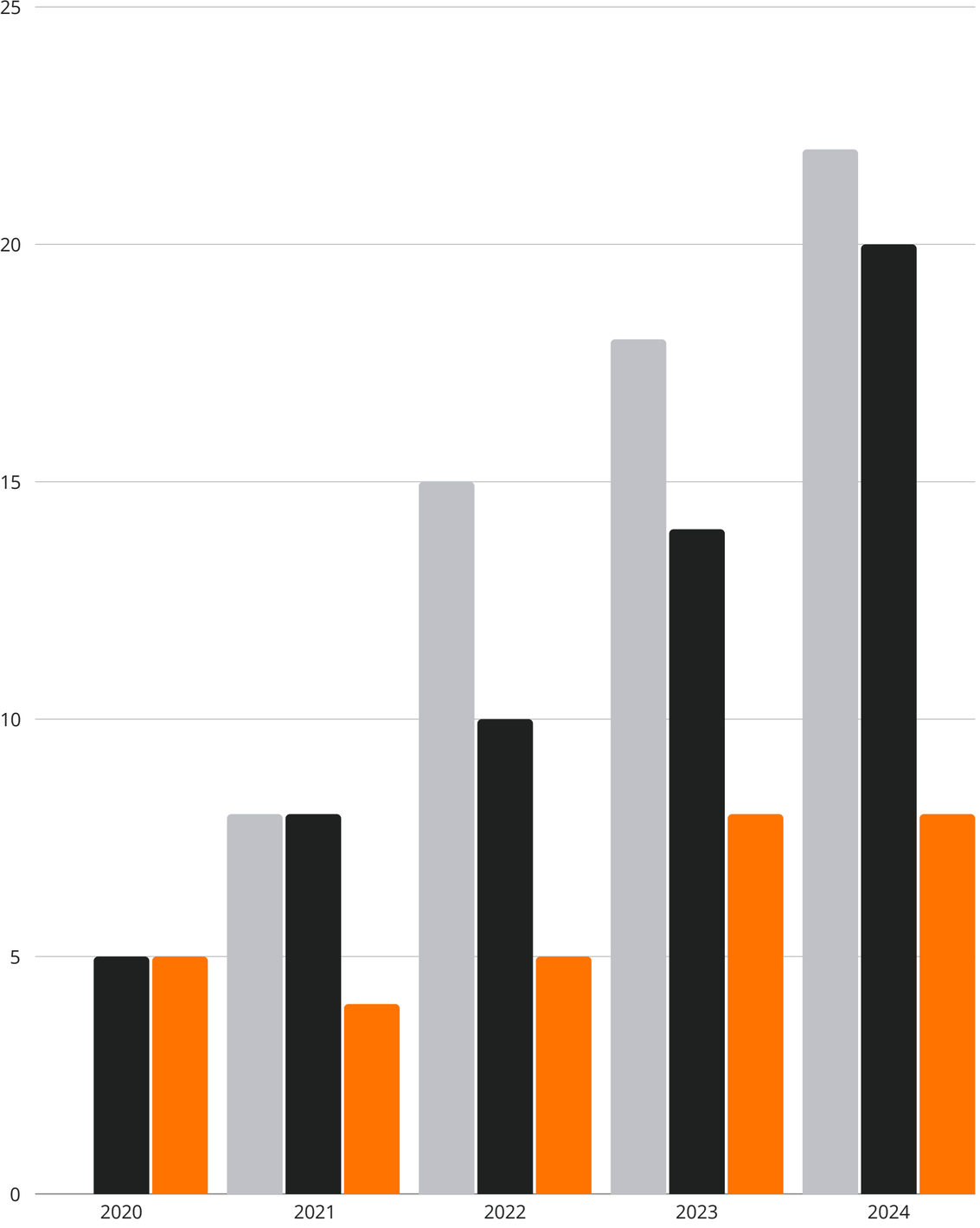
Python libraries – Pandas

Key Areas:

1. Data cleaning and preparation

2. Exploratory data analysis (EDA)

3 . Insight generation with advanced analysis techniques



Data Import and Initial Setup

1.Imported essential libraries (Pandas and Numpy)

2.Loaded sales data (sales_data.csv)

3.Previewed the dataset to understand its structure and initial values

Load the CSV Data

python

```python
import pandas as pd


# Load the transactions dataset
transactions = pd.read_csv('transactions.csv')
customers = pd.read_csv('customers.csv')
products = pd.read_csv('products.csv')
```

Preview the Data:

To quickly check if the data loaded correctly

```python
data.head()  # Shows the first 5 rows
```

Data Cleaning and Preparation

Key Actions

1. Checked for missing values and filled/dropped as necessary.
2. Removed duplicate entries to ensure data integrity
3. Verified and adjusted data types (e.g., converting date columns).
4. Methods Used: isnull(),sum(), dropna(), fillna(), drop_duplicates(), and dtypes().

Check for Missing Values

- It’s essential to understand where data may be incomplete

```python
data.isnull().sum()
```

- data.isnull().sum() provides a quick summary of the missing values in each column, making it easy to identify columns with missing data.

Handle Missing Values:

- Option 1: Drop rows with any missing values

```
python


data.dropna(inplace=True)
```

- Option 2: Fill missing values with a specific value

```
data.fillna(0, inplace=True)  # Replace with 0
# Or, replace with mean of the column
data['column_name'].fillna(data['column_name'].mean(), inplace=True
```

Remove Duplicates:

- Duplicate records can impact analysis accuracy

```python
data.drop_duplicates(inplace=True)
```

View Data Types

- To understand what types of data you're working with, check the data types:

```python
data.dtypes
```

Data Exploration

- Objective: Understand data distributions and relationships

Techniques:

- Descriptive statistics to get an overview of numerical columns
- Unique values and value counts to explore categorical columns
- Data type checks to ensure accurate data formats

- Descriptive Statistics: Quickly get insights into numeric columns

```
data.describe()
```

- Unique Values in a Column: Useful for categorical columns

```
data['column_name'].unique()
```

- **SELECTING COLUMNS**

```
transactions[['sales_date', 'sales_amount', 'product_code']].head()
```

- **FILTERING ROWS WHERE SOME CONDITIONS ARE APPLIED.**

```
transactions[transactions['sales_amount'] > 1000]
```

Filtered Sales Data:

- Filtered data by year and product for focused analysis
- Insight: Enables targeted analysis for specific periods or products

```
# Example: Filter for sales in 2023
data_2023 = data[data['year'] == 2023]


# Example: Filter for a specific product
product_sales = data[data['product_name'] == 'Product A']
```

Sales by Category

- Used groupby to aggregate sales by product category
- Insight: Helps identify high-performing categories

```
sales_by_category = data.groupby('product_category')['sales'].sum
sales_by_category
```

Key Insights

- High-Performing Categories: Identified top product categories based on sales totals.
- Sales Performance: Categorized products based on sales thresholds to highlight strong performers.

Summary

- Completed end-to-end data handling, from loading to cleaning and analysis

- Key insights derived from sales data can inform business strategy


THANK YOU