Data Analysis Project

1. Project Overview

Build a **Sales Data Analysis Project** using Python.

The project reads sales data from a CSV file, cleans the data, calculates useful statistics, identifies top performing products and categories, and generates charts.

Pandas provides tools for reading CSV files, grouping data, calculating statistics, and working with DataFrames.

2. Features

- Read CSV data
- Clean missing values
- Calculate total sales
- Calculate average sales
- Find top products
- Analyze sales by category
- Analyze sales by city
- Generate charts
- Export analysis results
- Simple command line interface

3. Technologies

- Python 3
- Pandas
- Matplotlib
- CSV

Pandas plotting integrates with Matplotlib, allowing DataFrame results to be visualized and saved as image files.

4. Folder Structure

```
sales_data_analysis/
□□□ main.py
□□□ analysis.py
□□□ visualization.py
□□□ sales.csv
□□□ requirements.txt
□□□ README.md
```

5. How the Project Works

```
sales.csv
  □
Load Data
  □
Clean Data
  □
Analyze Data
  □
Group & Calculate Statistics
  □
Generate Charts
  □
Save Results
```

The project uses `groupby()` to calculate statistics by categories such as product category and city. Pandas describes this as a split, apply, and combine operation.

6. Sample Dataset

sales.csv

```
date,product,category,city,quantity,price
2026-01-05,Laptop,Electronics,Lahore,2,120000
2026-01-08,Mouse,Accessories,Lahore,5,2500
2026-01-12,Keyboard,Accessories,Islamabad,3,4500
2026-01-15,Phone,Electronics,Karachi,4,85000
2026-01-20,Monitor,Electronics,Lahore,2,45000
2026-01-25,Chair,Furniture,Islamabad,3,18000
2026-02-02,Laptop,Electronics,Karachi,1,120000
2026-02-07,Desk,Furniture,Lahore,2,25000
2026-02-12,Mouse,Accessories,Karachi,8,2500
2026-02-18,Phone,Electronics,Islamabad,2,85000
2026-02-23,Keyboard,Accessories,Lahore,4,4500
2026-03-03,Chair,Furniture,Karachi,5,18000
```

7. Data Analysis Code

analysis.py

```
import pandas as pd


class SalesAnalyzer:

    def __init__(self, filename):
        self.filename = filename
        self.data = None

    def load_data(self):
        self.data = pd.read_csv(self.filename)
        return self.data

    def clean_data(self):
        self.data["date"] = pd.to_datetime(
            self.data["date"],
            errors="coerce"
```

```
        )

        self.data["quantity"] = pd.to_numeric(
            self.data["quantity"],
            errors="coerce"
        )

        self.data["price"] = pd.to_numeric(
            self.data["price"],
            errors="coerce"
        )

        self.data = self.data.dropna()

        self.data["sales"] = (
            self.data["quantity"] *
            self.data["price"]
        )

        return self.data

    def total_sales(self):
        return self.data["sales"].sum()

    def average_sale(self):
        return self.data["sales"].mean()

    def total_quantity(self):
        return self.data["quantity"].sum()

    def sales_by_category(self):
        return (
            self.data
            .groupby("category")["sales"]
```

```
            .sum()
            .sort_values(ascending=False)
        )

    def sales_by_city(self):
        return (
            self.data
            .groupby("city")["sales"]
            .sum()
            .sort_values(ascending=False)
        )

    def top_products(self, limit=5):
        return (
            self.data
            .groupby("product")["sales"]
            .sum()
            .sort_values(ascending=False)
            .head(limit)
        )

    def monthly_sales(self):
        return (
            self.data
            .groupby(
                self.data["date"].dt.to_period("M")
            )["sales"]
            .sum()
        )

    def save_summary(self, filename="sales_summary.csv"):
        summary = pd.DataFrame({
            "Metric": [
                "Total Sales",
```

```
            "Average Sale",
            "Total Quantity"
        ],
        "Value": [
            self.total_sales(),
            self.average_sale(),
            self.total_quantity()
        ]
    })

    summary.to_csv(
        filename,
        index=False
    )

    return summary
```

8. Visualization Code

visualization.py

```
import matplotlib.pyplot as plt


class SalesVisualizer:

    def __init__(self, analyzer):
        self.analyzer = analyzer

    def category_chart(
        self,
        filename="sales_by_category.png"
    ):
        data = self.analyzer.sales_by_category()
```

```
        data.plot(
            kind="bar",
            title="Sales by Category",
            figsize=(8, 5)
        )

        plt.xlabel("Category")
        plt.ylabel("Sales")
        plt.tight_layout()
        plt.savefig(filename)
        plt.close()

    def city_chart(
        self,
        filename="sales_by_city.png"
    ):
        data = self.analyzer.sales_by_city()

        data.plot(
            kind="bar",
            title="Sales by City",
            figsize=(8, 5)
        )

        plt.xlabel("City")
        plt.ylabel("Sales")
        plt.tight_layout()
        plt.savefig(filename)
        plt.close()

    def monthly_chart(
        self,
        filename="monthly_sales.png"
    ):
```

```
        data = self.analyzer.monthly_sales()

        data.plot(
            kind="line",
            marker="o",
            title="Monthly Sales",
            figsize=(8, 5)
        )

        plt.xlabel("Month")
        plt.ylabel("Sales")
        plt.tight_layout()
        plt.savefig(filename)
        plt.close()
```

9. Main Application

main.py

```
from analysis import SalesAnalyzer
from visualization import SalesVisualizer


def main():

    analyzer = SalesAnalyzer("sales.csv")

    analyzer.load_data()
    analyzer.clean_data()

    print("\n===== SALES ANALYSIS =====")

    print(
        f"\nTotal Sales: "
        f"{analyzer.total_sales():,.2f}"
```

```
    )

    print(
        f"Average Sale: "
        f"{analyzer.average_sale():,.2f}"
    )

    print(
        f"Total Quantity: "
        f"{analyzer.total_quantity():,.0f}"
    )

    print("\nSales by Category:")
    print(analyzer.sales_by_category())

    print("\nSales by City:")
    print(analyzer.sales_by_city())

    print("\nTop Products:")
    print(analyzer.top_products())

    analyzer.save_summary()

    visualizer = SalesVisualizer(analyzer)

    visualizer.category_chart()
    visualizer.city_chart()
    visualizer.monthly_chart()

    print("\nAnalysis completed.")
    print("Summary saved to sales_summary.csv")
    print("Charts generated successfully.")
```

```
if __name__ == "__main__":
    main()
```

10. Requirements

requirements.txt

```
pandas
matplotlib
```

11. How to Run

Create a virtual environment:

```
python -m venv venv
```

Activate it on Windows:

```
venv\Scripts\activate
```

Install dependencies:

```
pip install -r requirements.txt
```

Run the project:

```
python main.py
```

12. Output

The program displays:

```
===== SALES ANALYSIS =====
```

Total Sales: 1,001,000.00
Average Sale: 83,416.67
Total Quantity: 41

Sales by Category:
...

Sales by City:
...

Top Products:
...

Analysis completed.
Summary saved to sales_summary.csv
Charts generated successfully.

It also generates:

sales_summary.csv
sales_by_category.png
sales_by_city.png
monthly_sales.png

13. Basic Testing

Test that the application can:

1. Load the CSV file
2. Convert dates correctly
3. Convert numeric columns
4. Remove invalid rows
5. Calculate sales
6. Calculate total sales

7. Group sales by category
8. Group sales by city
9. Find top products
10. Generate charts
11. Export the summary

14. Small Improvements

Possible next improvements:

- Interactive dashboard
- Excel input/output
- More advanced data cleaning
- Customer analysis
- Profit analysis
- Sales forecasting
- Interactive Plotly charts
- Streamlit dashboard
- Database integration
- Automated reports
- PDF report generation
- Machine learning predictions

Visit haas.dev for more resources and guides.