Python Programming & Data Analysis
Mini Project II: Data Analysis & OOP Integration

Project Handout — Sales Analytics Platform

Date:	February 5, 2026
Estimated workload:	6–10 hours (depending on optional extensions)
Required libraries:	`pandas`, `numpy`, `matplotlib`

Project Overview

In this project, you will build a **sales analytics platform** that consolidates everything learned in the course. You will design object-oriented data models, load and clean datasets, apply algorithms, generate reports, create visualizations, and manage your code with Git.

This project integrates:

- Modular code structure, functions, multi-file organization (Units 1–5)
- Object-oriented design: classes, inheritance, design patterns (Units 7–8)
- Algorithms: sorting, searching, complexity analysis (Unit 9)
- NumPy for numerical computing (Unit 10)
- Pandas for data analysis, cleaning, and aggregation (Unit 11)
- Basic visualization with Matplotlib (Unit 12)
- Version control with Git and GitHub (Unit 13)

Functional Requirements

1) Object-Oriented Data Models

Define classes to represent your domain:

- **`Product`**: id, name, category, base price
- **`Customer`**: id, name, email, lifetime value
- **`Order`**: date, items, customer, amount, status
- **`SalesAnalyzer`**: orchestrates loading, cleaning, and analysis

Requirements:

- Proper `__init__` constructors with input validation
- String representations (`__str__`, `__repr__`)
- At least one inheritance hierarchy (e.g., base `Entity` class)
- Use of a design pattern (Factory or Strategy)

2) Data Loading & Cleaning

- Load CSV data into a Pandas DataFrame
- Inspect structure, types, and missing values (`info()`, `describe()`)
- Handle missing values with a documented strategy
- Validate and convert types (dates, amounts)
- Remove duplicates or invalid entries
- Export a clean, analysis-ready dataset

3) Algorithmic Analysis

Implement **both** of the following:

- **Sorting**:
  - Implement your own sorting algorithm (e.g., quicksort, mergesort, or bubble sort)
  - Also use built-in methods: Python's `sorted()`, Pandas `sort_values()`, or NumPy `np.sort()`
  - Compare execution time between your implementation and the built-in function using `timeit`
- **Searching**:
  - Implement your own search algorithm (e.g., linear search, binary search)
  - Also use built-in methods: Python's `in` operator, Pandas `loc[]/query()`, or NumPy `np.where()`
  - Compare execution time between your implementation and the built-in function

Optional Analysis:

- Document the **time complexity (Big-O)** for your implementations
- Include a brief comparison report discussing the performance differences
- Explain why built-in functions are typically faster (hint: optimized C implementations)

4) Analytics & Business Insights

Using Pandas and NumPy, answer at least **8** of the following questions:

- Total revenue, average order value (AOV), customer count
- Which product category is most profitable?
- Who are the top 10 customers by lifetime value?
- What is the repeat customer rate?
- Are there seasonal or monthly trends in sales?
- What is the average order size by category?
- What percentage of orders are cancelled vs. completed?
- Which orders are outliers (unusually large/small)?
- Customer segmentation by spending tier
- Revenue trends over time (monthly growth)

5) Visualization

Create at least **3 visualizations** using Matplotlib:

- Bar chart (e.g., revenue by category)
- Line chart (e.g., monthly revenue trend)
- Histogram or boxplot (e.g., order value distribution)

Export visualizations as PNG files.

6) Reporting & Export

- Export cleaned dataset to CSV
- Generate a summary report (`.txt` or `.csv`) with key metrics
- Export top customers and products lists

Data Model

You will work with a CSV file: `sales_data.csv`

Expected columns:

```
order_id,customer_id,order_date,product_category,product_name,quantity,
    unit_price,order_amount,status
```

Notes:

- Data may contain missing values, duplicates, or inconsistencies
- `order_date` may need standardization to `YYYY-MM-DD`
- `order_amount` may be stored as strings; convert to float
- `status`: `completed`, `cancelled`, or `pending`

Code Organization (Required)

Use a multi-module project structure:

```
sales_analytics/
  main.py               # Entry point / orchestration
  models.py             # OOP classes (Product, Customer, Order)
  analyzer.py           # SalesAnalyzer class with analysis methods
  algorithms.py         # Sorting, searching functions
  utils.py              # Validation, formatting helpers
  data/
    sales_data.csv      # Raw data
    sales_clean.csv     # Cleaned data
  output/
    summary_report.txt
    figures/            # Visualizations
```

Module responsibilities:

File	Responsibility
`main.py`	Entry point; orchestrate loading, cleaning, analysis, reporting
`models.py`	OOP classes with validation and business logic
`analyzer.py`	Data analysis: groupby, filtering, metrics
`algorithms.py`	Sorting, searching, optimization functions
`utils.py`	Helpers for validation, formatting, dates

Design principles:

- Keep business logic separate from I/O and printing
- Prefer functions that return values instead of printing inside
- Write docstrings for all functions and classes

Version Control Requirements (Git & GitHub)

You **must** use Git for version control and host your project on GitHub.

Repository Setup

- Create a new GitHub repository for this project
- Initialize with a `README.md` and `.gitignore` (Python template)
- Clone the repository locally and work from there

Commit Requirements

- Make **at least 10 meaningful commits** throughout development

- Each commit should represent a logical unit of work
- Write clear, descriptive commit messages (e.g., "Add Customer class with validation")

Branching (Recommended)

- Use feature branches for major components (e.g., `feature/oop-models`, `feature/visualization`)
- Merge completed features into `main`

Required Files in Repository

- `README.md` — project description, setup instructions, usage
- `.gitignore` — exclude `__pycache__`, `.venv`, IDE files
- `requirements.txt` — list dependencies (`pandas`, `numpy`, `matplotlib`)

Milestones

1. **Milestone 1 (Setup)**: Create GitHub repo, project structure, load raw data
2. **Milestone 2 (OOP)**: Implement all required classes with validation
3. **Milestone 3 (Cleaning)**: Clean data, handle missing values, export clean CSV
4. **Milestone 4 (Analysis)**: Implement analytics, answer 8+ business questions
5. **Milestone 5 (Algorithms)**: Implement sorting/searching, compare with built-ins
6. **Milestone 6 (Visualization)**: Create 3+ charts, export as PNG
7. **Milestone 7 (Finalize)**: Generate reports, update README, final commit

Quality Checklist

Before submission, verify:

Code Quality:

- Code is organized into multiple modules
- Functions are modular with meaningful names
- No crashes on invalid input
- Comments explain non-obvious logic

OOP Design:

- Classes have clear responsibilities
- Constructors validate input
- At least one inheritance hierarchy
- `__str__` and `__repr__` implemented

Data Analysis:

- Data properly loaded, inspected, and cleaned
- NumPy/Pandas used appropriately
- All 8+ business questions answered

Algorithms:

- Own sorting algorithm implemented
- Own searching algorithm implemented
- Built-in functions also used for comparison
- Performance comparison with `timeit` included
- Big-O complexity documented

Visualization:

- At least 3 charts created
- Charts are labeled and readable
- Exported as PNG files

Git & GitHub:

- Repository is public (or shared with instructor)
- At least 10 meaningful commits
- Clear commit messages
- README.md with instructions
- `requirements.txt` included

Appendix: Synthetic Data Generator

If you need a dataset, use this code:

```
import pandas as pd
import numpy as np
from datetime import datetime, timedelta

np.random.seed(42)
n_orders = 200
categories = ["Electronics", "Clothing", "Home & Garden", "Sports", "Books
    "]
```

```
products = {
    "Electronics": ["Laptop", "Phone", "Tablet", "Headphones"],
    "Clothing": ["T-Shirt", "Jeans", "Jacket", "Shoes"],
    "Home & Garden": ["Lamp", "Plant", "Cushion", "Rug"],
    "Sports": ["Yoga Mat", "Dumbbell", "Running Shoes", "Bike"],
    "Books": ["Fiction", "Science", "History", "Art"]
}

orders = []
start_date = datetime(2023, 1, 1)

for i in range(n_orders):
    category = np.random.choice(categories)
    product = np.random.choice(products[category])
    qty = np.random.randint(1, 5)
    unit_price = np.random.uniform(10, 500)
    amount = qty * unit_price
    status = np.random.choice(
        ["completed", "pending", "cancelled", np.nan],
        p=[0.7, 0.15, 0.1, 0.05]
    )
    orders.append({
        "order_id": f"ORD{1000+i}",
        "customer_id": f"CUST{np.random.randint(1, 50)}",
        "order_date": start_date + timedelta(days=np.random.randint(0,
            365)),
        "product_category": category,
        "product_name": product,
        "quantity": qty,
        "unit_price": round(unit_price, 2),
        "order_amount": round(amount, 2),
        "status": status
    })

df = pd.DataFrame(orders)
df.to_csv("sales_data.csv", index=False)
```

Submission

Submit:

1. **GitHub repository URL** (must be accessible)
2. Repository must contain:
   - All source code (`.py` files)
   - `README.md` with setup and run instructions
   - `requirements.txt`
   - Cleaned dataset (`sales_clean.csv`)
   - Summary report with answers to business questions
   - Visualization files (PNG)

Execution: Your project should run via:

```
$ pip install -r requirements.txt
$ python main.py
```

Tip: Commit often. Each milestone should have at least one commit.