Project 2

Pandas Data Manipulation

Introduction to Data Science with Python

Computer Science Program

Kutaisi International University

Project Information

Total Weight:	6% of Final Grade
Task Breakdown:	3 Tasks × 2% each
Due Date:	End of Week 6 (Sunday, 23:59 GT)
Format:	Individual Assignment
Deliverable:	Jupyter Notebook (.ipynb) or Python Script (.py)

Contents

1 Project Overview

1.1 Introduction

In this project, you will work with real-world messy data, applying Pandas operations to clean, transform, and prepare datasets for analysis. You'll encounter common data quality issues such as missing values, duplicates, inconsistent formatting, and outliers—all of which you must handle systematically using Pandas' powerful data manipulation capabilities.

This project simulates authentic data science work where raw data is rarely clean and requires significant preprocessing before meaningful analysis can begin.

Important: Data Files

You will be provided with a Python script (`generate_project2_data.py`) that generates three CSV files with realistic e-commerce data and intentional quality issues. Run this script before starting your project work.
Download the data generator script from:

- LMS course materials folder
- Or instructor will provide via email/announcement

Run the script: `python generate_project2_data.py`
This will create: `customers.csv`, `products.csv`, `transactions.csv`

1.2 Learning Objectives

Upon completion of this project, you will be able to:

- **Data Loading:** Import data from various formats (CSV, Excel, JSON) using Pandas
- **Data Exploration:** Examine datasets to understand structure, types, and quality issues
- **Data Cleaning:** Handle missing values, remove duplicates, fix data type issues
- **Data Transformation:** Filter, sort, group, aggregate, and reshape data
- **Feature Engineering:** Create new features from existing columns
- **Data Integration:** Merge and concatenate multiple datasets
- **Documentation:** Maintain detailed records of cleaning decisions and transformations

1.3 Project Structure

Task	Focus Area	Weight
Task 1	Pandas Fundamentals & Exploration	2%
Task 2	Data Cleaning & Quality Control	2%
Task 3	Advanced Transformation & Integration	2%
Total		**6%**

Critical Deadline Information

Academic Integrity Policy:

- All work must be your own individual effort
- You may consult official documentation (Pandas docs) and course materials
- Copying code from online sources, AI tools, or other students will result in zero points
- If you reference any external resources for concepts, cite them in comments or markdown cells

2 Task Specifications

2.1 Task 1: Pandas Fundamentals & Exploration (2%)

Objective

Demonstrate proficiency in loading data, performing exploratory data analysis, and executing basic Pandas operations.

Scenario

You work for an e-commerce company that has provided you with customer transaction data from the past year. Your first task is to load this data, understand its structure, and perform initial exploratory analysis.

Requirements

Part A: Data Loading (20%)
Load the three provided CSV files into Pandas DataFrames:

1. **Load customers.csv:**
   - Contains customer information (205 rows)
   - Columns: customer_id, name, email, registration_date, country, age
2. **Load products.csv:**
   - Contains product catalog (50 rows)
   - Columns: product_id, product_name, category, price, stock
3. **Load transactions.csv:**
   - Contains transaction records (508 rows)
   - Columns: transaction_id, customer_id, product_id, quantity, transaction_date, payment_method

Use appropriate Pandas functions to load the data and verify successful loading.
Part B: Data Exploration (40%)
For each DataFrame, perform the following:

1. **Basic Information:**
   - Display first and last 5 rows
   - Show shape (rows, columns)
   - Display column names and data types
   - Show memory usage
2. **Statistical Summary:**
   - Use `.describe()` for numerical columns
   - Use `.describe(include='object')` for categorical columns
   - Identify unique value counts for categorical columns
3. **Data Quality Check:**

- Check for missing values in each column
- Check for duplicate rows
- Identify any unusual values or patterns

Part C: Basic Analysis (40%)
Using the loaded DataFrames, answer the following:

1. **Customer Analysis:**
   - How many customers from each country?
   - What is the age distribution (min, max, mean, median)?
   - Which month had the most new registrations?
2. **Product Analysis:**
   - How many products in each category?
   - What is the average price per category?
   - Which products are out of stock (stock = 0)?
3. **Transaction Analysis:**
   - How many transactions per payment method?
   - What is the most popular product (most transactions)?
   - Which customer made the most purchases?

Evaluation Criteria

- Correct data loading from provided CSV files (20%)
- Comprehensive exploration using appropriate Pandas methods (40%)
- Accurate analysis and insights (30%)
- Code organization and documentation (10%)

Note on Data Files

The data generator script creates files with intentional issues. Do NOT clean or modify these files before loading them in Task 1. Your cleaning work should happen in Task 2.

Helpful Pandas Methods

- Loading: `pd.read_csv()`, `pd.read_excel()`
- Exploration: `.head()`, `.tail()`, `.info()`, `.describe()`
- Counting: `.value_counts()`, `.nunique()`
- Missing data: `.isnull()`, `.isna()`, `.sum()`
- Selection: `.loc[]`, `.iloc[]`, `[]`

2.2 Task 2: Data Cleaning & Quality Control (2%)

Objective

Apply systematic data cleaning techniques to handle missing values, duplicates, data type issues, and outliers.

Scenario

The provided datasets contain various data quality issues that are common in real-world data. Your manager has asked you to clean the data and document all cleaning decisions for the team.

Requirements

Part A: Identify Data Quality Issues (20%)

Thoroughly examine each dataset to identify all quality issues:

1. **In customers.csv, look for:**
   - Missing values (especially in email column)
   - Duplicate rows
   - Inconsistent data types (e.g., age column)
   - Inconsistent country names (USA, US, United States)
2. **In products.csv, look for:**
   - Missing values in price column
   - Negative prices (data entry errors)
   - Unrealistic stock values
   - Whitespace around product names
   - Inconsistent category naming (mixed case)
3. **In transactions.csv, look for:**
   - Missing quantities
   - Duplicate transaction IDs
   - Invalid customer_id references
   - Future dates
   - Inconsistent payment_method naming

Document what you find using Pandas methods like `.isnull().sum()`, `.duplicated().sum()`, `.describe()`, etc.

Part B: Data Cleaning Implementation (50%)

Clean each dataset systematically:

1. **Handle Missing Values:**
   - For `email`: Drop rows with missing emails
   - For `price`: Fill with median price of same category
   - For `quantity`: Fill with mode (most common value)
   - Document why you chose each strategy

2. **Remove Duplicates:**
   - Identify and remove exact duplicate rows
   - Keep the first occurrence
   - Report how many duplicates were found
3. **Fix Data Types:**
   - Convert `age` to integer (extract numbers from strings)
   - Convert date columns to datetime format
   - Ensure `price` and `quantity` are numeric
4. **Standardize Values:**
   - Standardize country names (replace ”US”, ”USA” with ”United States”)
   - Strip whitespace from text columns
   - Convert email addresses to lowercase
5. **Handle Outliers & Invalid Data:**
   - Remove or fix negative prices
   - Cap unrealistic stock levels at 500
   - Remove transactions with invalid customer_id references
   - Remove future dates from transactions

Part C: Validation & Documentation (35%)

1. **Create Cleaning Report:**
   - Create a DataFrame or dictionary showing:
     - Original row count vs. cleaned row count for each dataset
     - Number of missing values before and after
     - Number of duplicates removed
     - Number of outliers handled
     - Data type corrections made
2. **Verify Data Quality:**
   - Confirm no missing values remain (or document why kept)
   - Confirm no duplicates remain
   - Confirm all data types are correct
   - Confirm value ranges are reasonable
3. **Export Clean Data:**
   - Save cleaned DataFrames to new CSV files:
   - `customers_clean.csv`
   - `products_clean.csv`
   - `transactions_clean.csv`

Evaluation Criteria

- Thorough identification of all data issues (20%)
- Systematic cleaning implementation (50%)
- Complete validation and documentation (20%)
- Code quality and comments explaining decisions (10%)

Essential Pandas Cleaning Methods

- Missing data: `.fillna()`, `.dropna()`, `.interpolate()`
- Duplicates: `.duplicated()`, `.drop_duplicates()`
- Data types: `.astype()`, `pd.to_datetime()`, `pd.to_numeric()`
- String operations: `.str.strip()`, `.str.lower()`, `.str.replace()`
- Filtering: `.between()`, Boolean indexing
- Replacing: `.replace()`, `.map()`

2.3 Task 3: Advanced Transformation & Integration (2%)

Objective

Perform complex data transformations, merge multiple datasets, engineer new features, and conduct comprehensive analysis.

Scenario

Now that your data is clean, you need to integrate the three datasets, create meaningful features, and provide actionable business insights to your manager.

Requirements

Part A: Data Integration (25%)
Merge the three cleaned datasets:

1. **Create Complete Transaction View:**
   - Merge `transactions` with `customers` (on customer_id)
   - Merge result with `products` (on product_id)
   - Result should have all transaction details with customer and product info
   - Use appropriate join type (inner, left, right, outer) and explain why
2. **Handle Merge Issues:**
   - Check for unmatched records
   - Document any data loss during merges
   - Verify merged DataFrame has expected number of rows

Part B: Feature Engineering (30%)
Create new features from the merged dataset:

1. **Financial Features:**
   - `total_amount`: price $\times$ quantity for each transaction
   - `discount`: Apply 10% discount if quantity > 3
   - `final_amount`: total_amount - discount
2. **Temporal Features:**
   - `transaction_month`: Extract month from transaction_date
   - `transaction_day_of_week`: Extract day name (Monday, Tuesday, etc.)
   - `customer_age_at_purchase`: Calculate age at time of purchase
3. **Categorical Features:**
   - `customer_segment`: Based on total spending
     - High: > 1000
     - Medium: 500-1000
     - Low: < 500
   - `age_group`: 18-30, 31-45, 46-60, 61+

- `is_weekend`: True if purchase was on Saturday/Sunday

Part C: Advanced Analysis (45%)
Perform comprehensive analysis using groupby and aggregation:

1. **Revenue Analysis:**
   - Total revenue by product category
   - Monthly revenue trend (show growth/decline)
   - Revenue by country (top 5 countries)
   - Average transaction value by payment method
2. **Customer Behavior:**
   - Number of purchases per customer (top 10 customers)
   - Average spending by age group
   - Most popular product category by country
   - Weekend vs. weekday transaction patterns
3. **Product Performance:**
   - Top 10 products by revenue
   - Top 10 products by quantity sold
   - Category with highest average transaction value
   - Identify slow-moving products (low sales)
4. **Create Summary Tables:**
   - Pivot table: category vs. country showing total revenue
   - Cross-tabulation: age_group vs. customer_segment
   - Summary statistics grouped by multiple dimensions

Evaluation Criteria

- Correct data integration with appropriate join strategy (25%)
- Accurate feature engineering (30%)
- Comprehensive and insightful analysis (35%)
- Code efficiency and organization (10%)

Excellence Indicators: To achieve >95%, demonstrate:

- Strategic use of Pandas methods (chaining operations efficiently)
- Creative feature engineering beyond requirements
- Insightful business interpretations of results
- Professional data presentation (sorted, formatted tables)
- Efficient code without unnecessary loops

Advanced Pandas Operations

- Merging: `.merge()`, `pd.concat()`, `.join()`
- Grouping: `.groupby()`, `.agg()`, `.transform()`
- Pivoting: `.pivot_table()`, `.crosstab()`
- DateTime: `.dt.month`, `.dt.day_name()`, `.dt.year`
- Binning: `pd.cut()`, `pd.qcut()`
- Sorting: `.sort_values()`, `.nlargest()`, `.nsmallest()`

3 Submission Guidelines

3.1 Deliverable Format

Submit **ONE** file in your preferred format:

- Jupyter Notebook: `Project2_FirstName_LastName.ipynb`, OR
- Python Script: `Project2_FirstName_LastName.py`

You may use any IDE (PyCharm, VS Code, Jupyter, Spyder, etc.)

Required File Structure

For Jupyter Notebook (.ipynb):

1. **Title Cell (Markdown):**
   - Project title and number
   - Your full name and Student ID
   - Submission date
   - Honor code: "I certify this work is my own"
2. **For Each Task:**
   - Clear header with task number and title
   - Brief markdown explanation before major code blocks
   - Well-commented code cells
   - All cells executed with visible outputs
3. **Include Generated Files:**
   - Original CSV files (from data generator script)
   - Cleaned CSV files from Task 2
   - Do NOT include the data generator script itself
   - (Submit in a ZIP if needed)

For Python Script (.py):

1. **Header Comment Block:**

```
"""
Project 2: Pandas Data Manipulation
Name: [Your Full Name]
Student ID: [Your ID]
Date: [Submission Date]
Honor Code: I certify this work is my own
"""
```

2. **For Each Task:**
   - Clear section comments (e.g., `# ===== TASK 1 =====`)
   - Function definitions with docstrings
   - Well-commented code

- Print statements to display results

3. **Include Generated Files:**
    - Original CSV files (from data generator script)
    - Cleaned CSV files from Task 2
    - Do NOT include the data generator script itself

3.2 Code Quality Standards

Mandatory Standards:

- **Naming:** Descriptive variable names (`customer_df`, not `df1`)
- **Comments:** Explain cleaning decisions and transformations
- **Formatting:** Consistent indentation, proper spacing
- **Imports:** All imports at the top (`import pandas as pd`, `import numpy as np`)
- **Efficiency:** Use Pandas methods, avoid unnecessary loops
- **Execution:** Code must run completely without errors
- **Output:** Display key results clearly (use `.head()`, not entire DataFrames)

3.3 Submission Methods

Choose **ONE** submission method:

Option 1: Direct LMS Upload (Recommended)

1. Ensure your code runs completely without errors
2. For Jupyter: verify all cells executed with visible outputs
3. Gather all files (script/notebook + CSV files)
4. Create ZIP file: `Project2_FirstName_LastName.zip`
5. Upload ZIP to LMS before deadline
6. Verify upload was successful

Option 2: GitHub Repository Submission

1. Create a **public** GitHub repository named: `KIU-DS-Project2-FirstName-LastName`
2. Repository structure:

```
Project2/
 Project2_FirstName_LastName.ipynb (or .py)
 data/
    original/
       customers.csv
       products.csv
       transactions.csv
    cleaned/
        customers_clean.csv
        products_clean.csv
        transactions_clean.csv
 README.md
 requirements.txt
```

3. `README.md` should include:
   - Project description
   - Your name and student ID
   - Instructions to run your code
   - Brief summary of cleaning decisions
4. `requirements.txt`:

```
pandas==2.0.3
numpy==1.24.3
```

5. Submit repository URL via LMS **before deadline**
6. **CRITICAL:** Do not commit after deadline

Critical Deadline Information

Deadline Enforcement:

- **Due Date:** End of Week 6 - Sunday, 23:59 Georgian Time
- **LMS Submissions:** Timestamp automatically recorded, late submissions not accepted
- **GitHub Submissions:** Last commit timestamp verified
- **Any commits after deadline = Automatic 0 points**
- **Technical Issues:** Submit at least 2 hours early
- **No Extensions:** Start early!

3.4 Pre-Submission Checklist

Before submitting, verify:

☐ All three tasks completed
☐ Code runs successfully from start to finish
☐ For Jupyter: all outputs visible
☐ All CSV files generated and included
☐ Filename follows convention
☐ Header includes name, ID, honor code
☐ Code is well-commented with cleaning decisions explained
☐ No errors when running complete file
☐ DataFrames displayed with `.head()` not entire frames
☐ (If ZIP) all files included in archive
☐ (If GitHub) README and requirements.txt present
☐ Submitting well before deadline

4 Grading Rubric

4.1 Grade Distribution

Component	**Points**	**% of Final Grade**
Task 1: Pandas Fundamentals	2.0	2%
Task 2: Data Cleaning	2.0	2%
Task 3: Advanced Transformation	2.0	2%
Project Total	**6.0**	**6%**

Table 1: Project Point Distribution

4.2 Per-Task Grading Criteria

Criterion	**Excellent (90-100%)**	**Good (70-89%)**	**Needs Work (<70%)**
Correctness	All operations correct, handles edge cases	Minor errors, mostly correct	Significant errors or incomplete
Data Quality	Comprehensive cleaning, well-documented decisions	Adequate cleaning, some documentation	Insufficient cleaning or poor decisions
Pandas Usage	Efficient use of Pandas methods, vectorized operations	Functional but could be more efficient	Inefficient or improper methods
Documentation	Clear explanations of all cleaning decisions	Basic comments present	Minimal or missing documentation
Analysis	Insightful, goes beyond requirements	Adequate, meets requirements	Superficial or incomplete

Table 2: Quality Assessment Rubric

4.3 Deductions

Points will be deducted for:

- **-10%** Missing or incorrect filename
- **-15%** Code doesn't run completely without errors
- **-10%** Missing CSV files or data outputs
- **-10%** Minimal or no comments explaining cleaning decisions
- **-15%** Using loops where Pandas methods should be used
- **-10%** Displaying entire DataFrames instead of samples
- **-5%** Poor code organization
- **-100%** Late submission or commits after deadline
- **-100%** Academic integrity violations

4.4 Bonus Opportunities (Up to +10%)

- **+5%:** Exceptional data cleaning strategy with creative solutions
- **+3%:** Outstanding analysis with business-relevant insights
- **+2%:** Professional-quality documentation and code organization

Note: Bonus points apply to this project only, maximum score capped at 6%

5 Resources & Support

5.1 Official Documentation

- **Pandas Documentation:** `https://pandas.pydata.org/docs/`
- **Pandas User Guide:** `https://pandas.pydata.org/docs/user_guide/index.html`
- **10 Minutes to Pandas:** `https://pandas.pydata.org/docs/user_guide/10min.html`
- **Data Cleaning Guide:** `https://pandas.pydata.org/docs/user_guide/missing_data.html`

5.2 Recommended Study Materials

- Course lecture slides (Weeks 4-6)
- Lab notebooks on Pandas
- Python Data Science Handbook - Chapter 3: `https://jakevdp.github.io/PythonDataScienceHandboo`

5.3 Getting Help

1. **Review Course Materials:** Start with lecture slides and lab notebooks
2. **Email Instructor:** Contact Nika Gagua at `Nika.Gagua@kiu.edu.ge` for clarifications
3. **Study Groups:** Discuss approaches with peers (write your own code)

Success Strategies

- **Start Early:** Begin right after Week 5
- **Run Data Generator First:** Execute the provided script before coding
- **Don't Modify CSV Files:** Keep original files untouched, do cleaning in code
- **Work Incrementally:** Complete one task at a time
- **Test Frequently:** Check your DataFrames after each operation
- **Document Decisions:** Write comments as you make cleaning choices
- **Check Data Quality:** Use `.info()`, `.describe()` frequently
- **Verify Merges:** Always check row counts before and after merging
- **Use Method Chaining:** Combine operations for cleaner code
- **Save Intermediate Results:** Export cleaned data to verify
- **Ask for Help Early:** Don't struggle alone

5.4 Common Pitfalls to Avoid

- Not verifying data types after loading
- Forgetting to check for duplicates

- Dropping missing values without considering impact
- Using wrong merge type (inner vs. left vs. outer)
- Not documenting cleaning decisions
- Displaying entire large DataFrames (use `.head()`)
- Not saving intermediate cleaned files
- Using loops instead of Pandas vectorized operations
- Not handling edge cases (e.g., division by zero)
- Starting too late

6 Frequently Asked Questions

6.1 General Questions

Q: How long should this project take?
A: Plan for 6-8 hours total. Task 1: 1.5-2 hours, Task 2: 2.5-3 hours, Task 3: 2.5-3 hours. This does not include time to run the data generator (1 minute).

Q: Can I use libraries other than Pandas and NumPy?
A: For this project, focus on Pandas. You may use NumPy for numerical operations, but avoid matplotlib/seaborn (not yet covered).

Q: Where do I get the data generator script?
A: Download `generate_project2_data.py` from LMS course materials or contact instructor. Run it before starting your project.

Q: Can I modify the data generator script?
A: No. Use the script as-is to generate your data files. All cleaning should be done in your project code.

Q: Can I work with a partner?
A: No. This is an individual assignment.

Q: What if I can't figure out how to clean a specific issue?
A: Document your attempted approach and ask for help early. Partial credit given for documented attempts.

6.2 Technical Questions

Q: How do I choose between `.fillna()` strategies?
A: Consider the data context. Use mean for normally distributed numerical data, median for skewed data, mode for categorical data. Document your reasoning.

Q: What merge type should I use?
A: Usually inner join for transactions (only keep matching records). Document if you need left/right/outer and why.

Q: My merged DataFrame has more rows than expected. Why?
A: Likely a one-to-many relationship. Check for duplicate keys. Use `.merge(validate='1:1')` to verify.

Q: How do I handle dates in different formats?
A: Use `pd.to_datetime()` with `format` parameter or `infer_datetime_format=True`.

Q: Should I create functions or write inline code?
A: Either is fine, but functions improve reusability. Document your approach.

6.3 Data Questions

Q: What if the data generator script doesn't run?
A: Ensure you have pandas and numpy installed. Contact instructor immediately if issues persist.

Q: Can I generate the data differently?
A: No. All students must use the provided data generator script to ensure consistency for grading.

Q: The data looks random/unrealistic. Is this correct?
A: Yes. The data is simulated for educational purposes. Focus on the cleaning techniques, not data realism.

6.4 Submission Questions

Q: Do I need to submit CSV files?
A: Yes! Include all generated CSV files (original and cleaned) in your submission.
Q: Can I submit as separate files instead of ZIP?
A: For LMS: No, submit as ZIP. For GitHub: Yes, use proper folder structure.
Q: What if my files are too large for LMS?
A: CSV files should be small (<1MB each). If issues persist, use GitHub option.
Q: Can I fix errors after submission?
A: No. Test thoroughly before submitting.

6.5 Grading Questions

Q: Will different cleaning strategies affect my grade?
A: As long as your strategy is reasonable and well-documented, you won't lose points for different approaches.
Q: What if I can't complete Task 3?
A: Each task is graded independently. Submit what you complete.
Q: Do I lose points for creating extra features?
A: No! Extra relevant features can earn bonus points.

Best of Luck!

Data cleaning is a critical skill in data science.
Master these techniques and you'll be prepared for real-world projects.
Take your time, document your decisions, and ask for help when needed.

We look forward to seeing your clean, well-structured data!

- Your Data Science Instructors