Task 1: Python Fundamentals & Data Manipulation Challenge

Environment: Python 3.9+, pandas, numpy, scikit-learn
Submission: Jupyter Notebook (`.ipynb`) or Python Script (`.py`)

Overview

You will work with a dataset of customer transactions to demonstrate your proficiency in **Python fundamentals** and **data manipulation** using essential data science libraries.
This task evaluates your ability to write **clean, efficient, and well-documented code** while solving practical problems.

Dataset Description

You are provided with three CSV files:

1. **`Customers.csv:`** `customer_id, customer_name, email, signup_date, country, age`
2. **`Products.csv:`** `product_id, product_name, category, supplier_id, cost_price`
3. **`Transactions.csv:`** `transaction_id, customer_id, product_id, quantity, price, timestamp, payment_method`

Download Link: INIT Task

Part A: Data Loading & Basic Operations

Basic Exploration

- Load all three datasets and display:
    - Shape of each dataframe
    - Data types of all columns
    - First 3 rows of each dataset

- Number of missing values in each column
- Number of duplicate `transaction_id` entries
- Date range of transactions (earliest and latest)

Datetime Transformation

- Convert `timestamp` to datetime format
- Extract:
    - Hour of day
    - Day of week
    - Month

Part B: Data Cleaning & Transformation

Handle Missing Values

- Numeric columns → fill with **median**
- Categorical columns → fill with **mode**
- Add comments explaining your approach

Revenue Calculation

`revenue = quantity × price`

Merge Datasets

Create a single dataframe `full_data` that includes:

- All transaction details
- Customer information
- Product information

Ensure **no data loss** from the `transactions` table.

Profit Margin

`profit_margin = (price - cost_price) / price × 100`

Part C: Aggregation & Analysis

C1. Customer Metrics

Calculate per customer:

- Total revenue generated
- Number of transactions
- Average transaction value
- Most frequently purchased category

Output: DataFrame sorted by total revenue (descending)

C2. Time-Based Analysis

Compute per **month**:

- Total revenue
- Number of unique customers
- Average order value
- Month-over-month growth rate (%)

C3. Product Performance

Find **Top 10 Products** by:

- Total revenue
- Total quantity sold
- Profit margin

Display results in a clear, formatted table.

C4. Customer Segmentation (RFM)

Use **Recency, Frequency, Monetary (RFM)** logic:

- **Recency:** Days since last purchase (from latest transaction date)
- **Frequency:** Total number of transactions

- **Monetary:** Total revenue generated

Create bins for each metric (Low, Medium, High) and assign segment labels to each customer.

Part D: Advanced Operations

Efficient Operations Challenge

Identify **suspicious transactions** defined as:

- `quantity > 100` **AND** `price < 10`,
  OR Customers with more than 3 purchases in a single hour.

Requirements:

- Must use **vectorized operations** (no loops)

Return DataFrame with:

`[transaction_id, customer_name, reason]`

```
def find_suspicious_transactions(full_data):
    """
    Identify suspicious transactions using vectorized operations.
    Returns: DataFrame with columns [transaction_id, customer_name,
reason]
    """
    # Your code here
    pass
```

Rolling Window Analysis

Calculate the **7-day moving average** of daily revenue.
Handle edge cases appropriately.

Feature Engineering

For each customer, create:

- `days_since_signup`: Days between signup date and first transaction
- `purchase_frequency`: Average days between purchases
- `category_diversity`: Number of unique product categories purchased
- `preferred_payment`: Most used payment method

Part E: Code Quality & Efficiency

E1. Optimization

Rewrite this **slow loop** using vectorized operations and benchmark performance.

```
# Slow Code (DO NOT USE!)
result = []
for idx, row in df.iterrows():
    if row['age'] > 25 and row['country'] == 'USA':
        result.append(row['customer_id'])
```

Benchmark using `%timeit` or `time.time()`.

E2. Reusable Function

Write a reusable, documented function for **Customer Lifetime Value (CLV)**:

```
def customer_lifetime_value(transactions_df, customer_id,
discount_rate=0.1):
    """
    Calculate customer lifetime value (CLV) with time-discounting.

    CLV = Σ(revenue × discount_factor^month)
    where month is indexed from the first purchase

    Parameters
    ----------
    transactions_df : pd.DataFrame
        DataFrame containing transaction data
    customer_id : int
        Customer ID to calculate CLV for
    discount_rate : float
        Monthly discount rate (default 0.1)

    Returns
    -------
    float : Calculated CLV
    """
    # Your code here
    pass
```

Evaluation Criteria

Criterion	Points	Description
Correctness	40	Accurate solutions, edge case handling
Code Efficiency	25	Vectorized operations, appropriate data structures
Code Quality	20	Clean, readable, well-commented code
Problem Solving	15	Logical thinking, creative approaches, robust handling

---

Bonus Challenge

Detect **potential duplicate customers** based on:

- Similar names (Levenshtein distance < 3)
- Same email domain
- Similar transaction patterns (same products within 7 days)

```
def find_duplicate_customers(customers_df, transactions_df):
    """
    Returns: DataFrame of potential duplicate customer pairs with
confidence score
    """
    pass
```

Use `scikit-learn`, `fuzzywuzzy`, or other string-matching libraries.

Submission Guidelines

- **Repository Name:** `INIT_TASK`
- **Platform:** GitHub
- **Instructions:**
    1. Create a **new public GitHub repository** named **`INIT_TASK`**
    2. Add your notebook or `.py` file and dataset (if needed)
    3. Include a short **README.md** with:
        - Your name
        - Short approach/summary
        - Dependencies/libraries used
    4. Push all your work to the repository
    5. Submit the **GitHub repository link** as your final submission

---

Tips

- Use `.loc` / `.iloc` instead of chained indexing
- Use `groupby()`, `agg()`, and `transform()` for aggregation
- Use `pd.merge()` carefully (verify join types)
- Validate your results (check nulls, types, logical values)
- Avoid explicit loops
- Don't ignore warnings, they often indicate real issues

References

- ▶ *Python Full Course for Beginners*
- ▶ *Data Analysis with Python Course - Numpy, Pandas, Data Visualization*

Task 2: Research Evaluation Task

Objective

This task evaluates your ability to **read, comprehend, and critically analyze** a research paper

Research Paper Link:
https://drive.google.com/file/d/1BZfzOV-IkvvChjkdXYQw6cGT62FQb8uK/view?usp=drive_link

---

Instructions

1. Paper Review (Mandatory)

- Read the assigned research paper carefully.
- Write a **summary (~150 words)** covering:
  - The **core idea / problem statement**
  - The **methodology / approach used**
  - The **key results or findings**
- Focus on **clarity and precision** - describe what the paper solves, how it solves it, and what outcomes it achieves.

---

2. Reflection (Mandatory)

- In **~200 words**, provide your personal reflection on the paper.
- Discuss:
  - One **key limitation** or challenge in the paper
  - A **potential improvement** or **future direction** you would propose

---

3. Implementation (Bonus Task)

- Attempt a **simplified implementation** or **small-scale experiment** inspired by the paper.
- This can be:
  - A minimal reproduction of part of the method
  - A variation or an applied demonstration of the concept

---

4. Results Presentation (Bonus Task)

- If you complete the implementation, include:
    - Short **results summary** (plots, tables, or screenshots)
    - A **2–3 sentence explanation** of your findings

Submission Guidelines

- **Repository Name:** `INIT_TASK` *(same repository as Task 1)*
- Create a separate folder named **`Task_2_Research_Evaluation`**
- Include:
    - Your written responses (`summary.md` or `.pdf`)
    - Any code, data, or visuals used in your implementation

- Update your `README.md` with:
    - The paper title
    - Authors and publication source (add a link to the paper)
    - A short note about your reflection or implementation