Name: Raghav Sharma

Roll No: 25201313

1. An e-commerce platform tracks user activity:
   - User ID
   - Viewed Products (list)
   - Purchased Products (set)
   - Product Categories (dictionary: product → category)

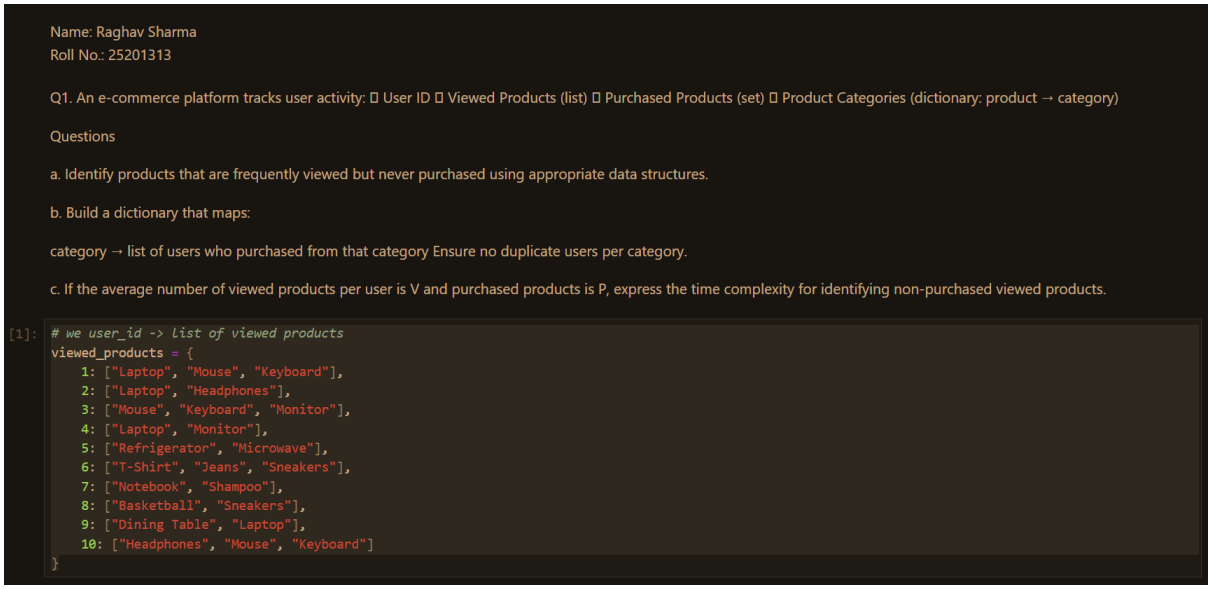
   Questions

   a. Identify products that are frequently viewed but never purchased using appropriate data structures.

   b. Build a dictionary that maps:

   category → list of users who purchased from that category
   Ensure no duplicate users per category.

   c. If the average number of viewed products per user is V and purchased products is P, express the time complexity for identifying non-purchased viewed products.



```
[3]: # we have product -> category
product_categories = {
    "Laptop": "Electronics",
    "Mouse": "Electronics",
    "Keyboard": "Electronics",
    "Headphones": "Electronics",
    "Monitor": "Electronics",
    "Dining Table": "Furniture",
    "T-Shirt": "Clothing",
    "Jeans": "Clothing",
    "Sneakers": "Footwear",
    "Refrigerator": "Appliances",
    "Microwave": "Appliances",
    "Shampoo": "Personal Care",
    "Notebook": "Stationery",
    "Basketball": "Sports"
}
```

```
•[4]: # Number of products viewed
view_count = {}

for products in viewed_products.values():
    for product in products:
        if product in view_count:
            view_count[product] += 1
        else:
            view_count[product] = 1

print(view_count)
```

```
{'Laptop': 4, 'Mouse': 3, 'Keyboard': 3, 'Headphones': 2, 'Monitor': 2, 'Refrigerator': 1, 'Microwave': 1, 'T-Shirt': 1, 'Jeans': 1, 'Sneakers': 2, 'Note
book': 1, 'Shampoo': 1, 'Basketball': 1, 'Dining Table': 1}
```

```
•[5]: # Set of purchased products
all_purchased = set()
for products in purchased_products.values():
    all_purchased.update(products)
```

```
[6]: all_purchased
```

```
[6]: {'Basketball',
 'Dining Table',
 'Headphones',
 'Jeans',
 'Keyboard',
 'Laptop',
 'Mouse',
 'Refrigerator',
 'T-Shirt'}
```

```
never_purchased_but_viewed = []

for product, count in view_count.items():
    if product not in all_purchased:
        never_purchased_but_viewed.append(product)

never_purchased_but_viewed
```

```
['Monitor', 'Microwave', 'Sneakers', 'Notebook', 'Shampoo']
```

```
print("Products viewed but never purchased:")
print(never_purchased_but_viewed)
```

```
Products viewed but never purchased:
['Monitor', 'Microwave', 'Sneakers', 'Notebook', 'Shampoo']
```

Part b:

Build a dictionary that maps:

category → list of users who purchased from that category Ensure no duplicate users per category.

```
category_to_users = {}

for user_id, products in purchased_products.items():
    for product in products:
        category = product_categories.get(product)

        if category not in category_to_users:
            category_to_users[category] = set()  # use set to avoid duplicates

        category_to_users[category].add(user_id)
```

```
category_to_users
```

```
{'Electronics': {1, 2, 4, 10},
 'Appliances': {5},
 'Clothing': {6},
 'Sports': {8},
 'Furniture': {9}}
```

```
# Convert sets to lists for final output
for category in category_to_users:
    category_to_users[category] = list(category_to_users[category])

print("\nCategory to users mapping:")
print(category_to_users)
```

```
Category to users mapping:
{'Electronics': [1, 2, 10, 4], 'Appliances': [5], 'Clothing': [6], 'Sports': [8], 'Furniture': [9]}
```

Part c:

If the average number of viewed products per user is V and purchased products is P,

express the time complexity for identifying non-purchased viewed products.

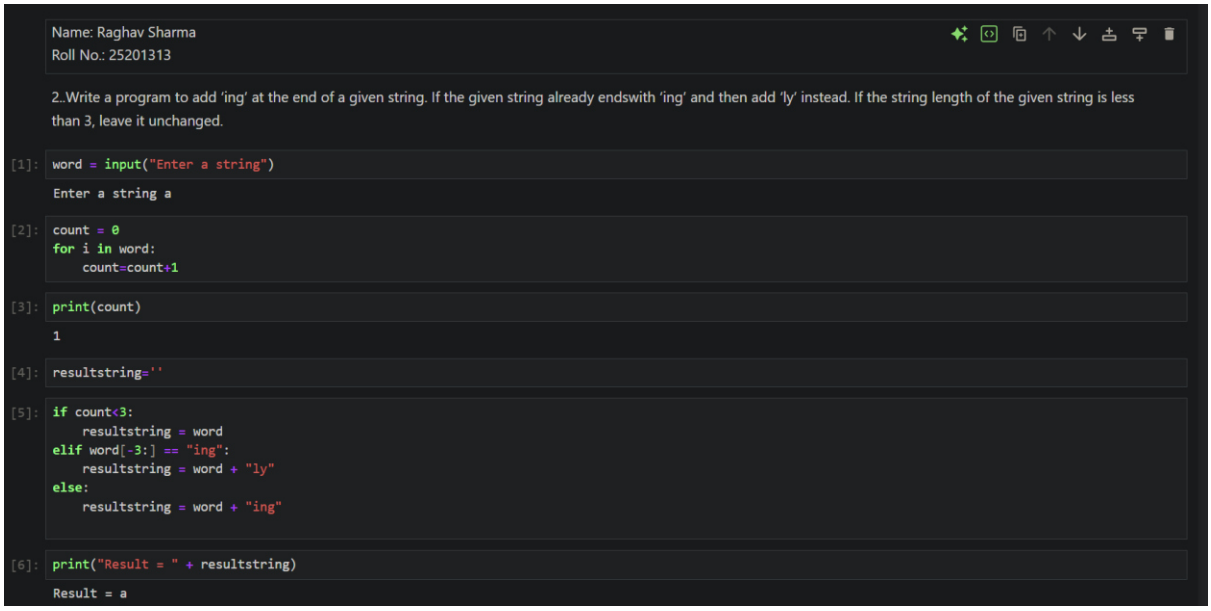
```
# let n be number of users, so checking viewed products will take O(n*V)
# and collecting data of purchased products will take O(n*P)
# so Total time Complexity = O(n*(V+P))
```

Name: Raghav Sharma

Roll No: 25201313

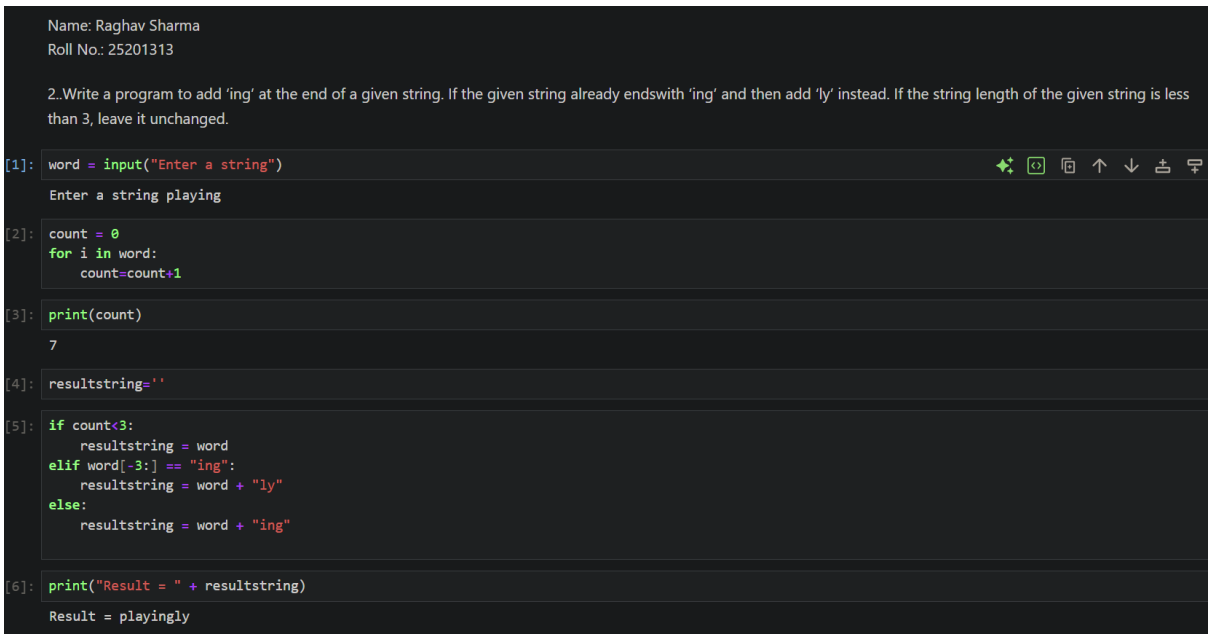
Q2:

Case 1: Less than 3



```
Name: Raghav Sharma
Roll No.: 25201313

2..Write a program to add 'ing' at the end of a given string. If the given string already endswith 'ing' and then add 'ly' instead. If the string length of the given string is less than 3, leave it unchanged.

[1]: word = input("Enter a string")
     Enter a string a

[2]: count = 0
     for i in word:
         count=count+1

[3]: print(count)
     1

[4]: resultstring=''

[5]: if count<3:
         resultstring = word
     elif word[-3:] == "ing":
         resultstring = word + "ly"
     else:
         resultstring = word + "ing"

[6]: print("Result = " + resultstring)
     Result = a
```

Case 2: Ending with “ing”



```
Name: Raghav Sharma
Roll No.: 25201313

2..Write a program to add 'ing' at the end of a given string. If the given string already endswith 'ing' and then add 'ly' instead. If the string length of the given string is less than 3, leave it unchanged.

[1]: word = input("Enter a string")
     Enter a string playing

[2]: count = 0
     for i in word:
         count=count+1

[3]: print(count)
     7

[4]: resultstring=''

[5]: if count<3:
         resultstring = word
     elif word[-3:] == "ing":
         resultstring = word + "ly"
     else:
         resultstring = word + "ing"

[6]: print("Result = " + resultstring)
     Result = playingly
```

Case 3: adding “ing”

Name: Raghav Sharma
Roll No.: 25201313

2..Write a program to add ‘ing’ at the end of a given string. If the given string already endswith ‘ing’ and then add ‘ly’ instead. If the string length of the given string is less than 3, leave it unchanged.

[1]:
```
word = input("Enter a string")
```

```
Enter a string play
```

[2]:
```
count = 0
for i in word:
    count=count+1
```

[3]:
```
print(count)
```

```
4
```

[4]:
```
resultstring=''
```

[5]:
```
if count<3:
    resultstring = word
elif word[-3:] == "ing":
    resultstring = word + "ly"
else:
    resultstring = word + "ing"
```

[6]:
```
print("Result = " + resultstring)
```

```
Result = playing
```

Name: Raghav Sharma

Roll No: 25201313

3.Multiple evaluation servers generate partial results.Problem Statement
a. Read multiple result files with student marks.
b. Validate:
o Marks range (0–100)
o Duplicate student IDs
c. Aggregate:
o Average marks
o Standard deviation
d. Automatically flag:
o Anomalous results (>2σ)
e. Log:
o Source file
o Error type
o Correction applied
f. Must use:
o Dictionaries
o Sets
o File I/O
o Exception chaining

Name: Raghav Sharma
Roll No: 25201313

3.Multiple evaluation servers generate partial results.Problem Statement
a. Read multiple result files with student marks.
b. Validate:
o Marks range (0–100)
o Duplicate student IDs
c. Aggregate:
o Average marks
o Standard deviation
d. Automatically flag:
o Anomalous results (>2σ)
e. Log:
o Source file
o Error type
o Correction applied
f. Must use:
o Dictionaries
o Sets
o File I/O
o Exception chaining

[1]:
```
with open("results1.txt", "w") as f:
    f.write("24201201,85\n")
    f.write("24201202,90\n")
    f.write("24201203,110\n")
    f.write("24201204,70\n")
    f.write("24201205,88\n")

with open("results2.txt", "w") as f:
    f.write("24201206,95\n")
    f.write("24201202,78\n")
    f.write("24201207,-10\n")
    f.write("24201208,60\n")
    f.write("24201209,45\n")
```

[2]:
```python
import math

result_files = ["results1.txt", "results2.txt"]

log_file = "error_log.txt"
```

[3]:
```python
students = {}
seen_ids = set()
all_marks = []
```

[4]:
```python
def log_error(source_file, error_type, correction):
    with open(log_file, "a") as log:
        log.write(f"File: {source_file} | Error: {error_type} | Correction: {correction}\n")
```

[5]:
```python
for file_name in result_files:
    try:
        with open(file_name, "r") as file:
            for line in file:
                try:
                    student_id, marks = line.strip().split(",")
                    marks = int(marks)

                    # Duplicate ID check
                    if student_id in seen_ids:
                        log_error(file_name, "Duplicate Student ID", "Ignored entry")
                        continue

                    # Marks range validation
                    if marks < 0 or marks > 100:
                        original = marks
                        # Correction: clamp marks to valid range
                        marks = max(0, min(100, marks))
                        log_error(
                            file_name,
                            "Invalid Marks Range",
                            f"Corrected from {original} to {marks}"
                        )

                    # Store valid data
                    seen_ids.add(student_id)
                    students[student_id] = marks
                    all_marks.append(marks)

                except ValueError as e:
                    # Error in conversion or format
                    log_error(file_name, "Invalid Line Format", "Line skipped")
                    raise ValueError("Error parsing student record") from e

    except FileNotFoundError as e:
        log_error(file_name, "File Not Found", "File skipped")
        raise FileNotFoundError("Result file missing") from e
```

[6]:
```
mean = sum(all_marks) / len(all_marks)

variance = sum((m - mean) ** 2 for m in all_marks) / len(all_marks)
std_dev = math.sqrt(variance)

print("Average Marks:", round(mean, 2))
print("Standard Deviation:", round(std_dev, 2))
```

```
Average Marks: 70.33
Standard Deviation: 30.0
```

[7]:
```
print("Anomalous Results (> 2σ):")
for student_id, marks in students.items():
    if abs(marks - mean) > 2 * std_dev:
        print(f"Student {student_id} -> Marks: {marks}")
```

```
Anomalous Results (> 2σ):
Student 24201207 -> Marks: 0
```

[8]:
```
#Log  Files

try:
    with open("error_log.txt", "r") as log:
        print("\n===== ERROR LOG CONTENT =====\n")
        for line in log:
            print(line.strip())
        print("\n===== END OF LOG =====")

except FileNotFoundError:
    print("error_log.txt not found. No logs to display.")
```

```
===== ERROR LOG CONTENT =====

File: results1.txt | Error: Invalid Marks Range | Correction: Corrected from 110 to 100
File: results2.txt | Error: Duplicate Student ID | Correction: Ignored entry
File: results2.txt | Error: Invalid Marks Range | Correction: Corrected from -10 to 0

===== END OF LOG =====
```