**PIYUSH KUMAR MISHRA**

**230957212**

**ROLL N0:70**

**WEEK 8:**

Exercise 1:

File Handling in Python with Error and Exception Handling

You are tasked with building a simple data processing system for an e-commerce company

that handles product inventories and customer reviews. The company uses two types of files to

store data:

1. CSV Files: The product inventory is stored in a CSV file with the following columns:

Product IL), Product Name, Category, Price, and Stock Quantity.

2, JSON Files: Customer reviews for each product are stored in separate JSON files. Each

review file contains a list of reviews for a product, where each review has the following

fields: Review IL), User ID, Rating, Comment, and Date.

Your system needs to process these files to provide insights and updates on product availability,

pricing trends, and customer feedback. Additionally, the system should allow adding new

products and updating product information in the CSV file, while handling errors and

exceptions gracefully.

System Requirements:

1. Product Management (CSV File):

• Add New Products: Write a function to add a new product to the inventory CSV file.

• Update Stock and Price: Implement a function to update the stock quantity and price

of an existing product in the CSV file.

Check Product Availability: Write a function that reads the CSV file to check if a

product is in stock and, if so, how many units are available.

2. Customer Review Management (JSON Files):

Add Customer Review: Implement a function that adds a new customer review to the

corresponding JSON file for a product.

Average Rating Calculation: Write a function that reads a product's JSON review file

and calculates its average rating.

Review Search: Implement a search function that reads the JSON file and allows users

to search for reviews containing specific keywords in the comment section.

3. Data Analysis:

• Top Rated Products: Create a function that reads all product review JSON files,

calculates the average rating for each product, and lists the top 5 highest-rated products.

• Out of Stock Products: Write a function that reads the CSV inventory file and lists all

products that are out of stock.

• Price Trends: Implement a function to read the CSV file and identify products whose

prices have increased or decreased in the past month.

4. Error and Exception Handling:

• Implement error handling for situations such as:

File Not Found: When the CSV or JSON file is missing.

Invalid Data Format: If the data in the files does not match the expected

structure.

File Corruption: When a file contains corrupted data.

Read/Write Permissions: When the system does not have permission to read

or write to a file.

import csv

import json

import os

def add\_product(file\_path):

product\_id = input("Enter Product ID: ")

product\_name = input("Enter Product Name: ")

category = input("Enter Category: ")

price = float(input("Enter Price: "))

stock\_quantity = int(input("Enter Stock Quantity: "))

try:

with open(file\_path, mode='a', newline='') as file:

writer = csv.writer(file)

writer.writerow([product\_id, product\_name, category, price, stock\_quantity])

print("Product added successfully.")

except PermissionError:

print("Error: Permission denied when trying to write to the file.")

except Exception as e:

print(f"An unexpected error occurred: {e}")

def update\_product(file\_path):

product\_id = input("Enter Product ID to update: ")

new\_price = float(input("Enter New Price: "))

new\_stock\_quantity = int(input("Enter New Stock Quantity: "))

try:

updated\_rows = []

with open(file\_path, mode='r') as file:

reader = csv.reader(file)

for row in reader:

if row[0] == product\_id:

row[3] = str(new\_price)

row[4] = str(new\_stock\_quantity)

updated\_rows.append(row)

with open(file\_path, mode='w', newline='') as file:

writer = csv.writer(file)

writer.writerows(updated\_rows)

print("Product updated successfully.")

except FileNotFoundError:

print("Error: The CSV file was not found.")

except Exception as e:

print(f"An unexpected error occurred: {e}")

def check\_availability(file\_path):

product\_id = input("Enter Product ID to check availability: ")

try:

with open(file\_path, mode='r') as file:

reader = csv.reader(file)

for row in reader:

if row[0] == product\_id:

return f"Available stock: {row[4]}"

return "Product not found."

except FileNotFoundError:

print("Error: The CSV file was not found.")

def add\_review():

product\_id = input("Enter Product ID for the review: ")

review\_id = input("Enter Review ID: ")

user\_id = input("Enter User ID: ")

rating = int(input("Enter Rating (1-5): "))

comment = input("Enter Comment: ")

date = input("Enter Date (YYYY-MM-DD): ")

review = {

"Review ID": review\_id,

"User ID": user\_id,

"Rating": rating,

"Comment": comment,

"Date": date

}

file\_name = f"{product\_id}\_reviews.json"

try:

if not os.path.exists(file\_name):

with open(file\_name, 'w') as file:

json.dump([], file)

with open(file\_name, 'r+') as file:

reviews = json.load(file)

reviews.append(review)

file.seek(0)

json.dump(reviews, file, indent=4)

print("Review added successfully.")

except Exception as e:

print(f"An error occurred while adding the review: {e}")

def average\_rating(product\_id):

file\_name = f"{product\_id}\_reviews.json"

try:

with open(file\_name) as file:

reviews = json.load(file)

total\_rating = sum(review['Rating'] for review in reviews)

average = total\_rating / len(reviews) if reviews else 0

return f"Average Rating: {average:.2f}"

except FileNotFoundError:

return "Review file not found."

except json.JSONDecodeError:

return "Error: Invalid JSON format."

def search\_reviews():

product\_id = input("Enter Product ID to search reviews: ")

keyword = input("Enter keyword to search in comments: ")

file\_name = f"{product\_id}\_reviews.json"

try:

with open(file\_name) as file:

reviews = json.load(file)

matching\_reviews = [review for review in reviews if keyword.lower() in review['Comment'].lower()]

return matching\_reviews if matching\_reviews else "No matching reviews found."

except FileNotFoundError:

return "Review file not found."

def top\_rated\_products(product\_ids):

ratings = {}

for product\_id in product\_ids:

avg\_rating\_result = average\_rating(product\_id)

if isinstance(avg\_rating\_result, str) and "Average Rating" in avg\_rating\_result:

ratings[product\_id] = float(avg\_rating\_result.split(": ")[1])

top\_products = sorted(ratings.items(), key=lambda x: x[1], reverse=True)[:5]

return top\_products

def out\_of\_stock\_products(file\_path):

out\_of\_stock = []

try:

with open(file\_path) as file:

reader = csv.reader(file)

for row in reader:

if int(row[4]) == 0:

out\_of\_stock.append(row[1])

return out\_of\_stock if out\_of\_stock else ["No products are out of stock."]

except FileNotFoundError:

return "Inventory file not found."

def price\_trends(current\_file\_path, previous\_file\_path):

current\_prices = {}

previous\_prices = {}

try:

with open(current\_file\_path) as current\_file:

reader = csv.reader(current\_file)

for row in reader:

current\_prices[row[0]] = float(row[3])

with open(previous\_file\_path) as previous\_file:

reader = csv.reader(previous\_file)

for row in reader:

previous\_prices[row[0]] = float(row[3])

trends = {}

for product\_id in current\_prices.keys():

if product\_id in previous\_prices:

change = current\_prices[product\_id] - previous\_prices[product\_id]

trends[product\_id] = change

return trends

except FileNotFoundError as e:

return f"File not found: {e.filename}"

def main():

inventory\_file\_path = 'inventory.csv'

while True:

print("\nE-commerce Data Processing System")

print("1. Add Product")

print("2. Update Product")

print("3. Check Product Availability")

print("4. Add Customer Review")

print("5. Calculate Average Rating")

print("6. Search Reviews")

print("7. List Out of Stock Products")

print("8. Show Top Rated Products")

choice = input("\nSelect an option (or 'q' to quit): ")

if choice == '1':

add\_product(inventory\_file\_path)

elif choice == '2':

update\_product(inventory\_file\_path)

elif choice == '3':

print(check\_availability(inventory\_file\_path))

elif choice == '4':

add\_review()

elif choice == '5':

product\_id = input("Enter Product ID to calculate average rating: ")

print(average\_rating(product\_id))

elif choice == '6':

results = search\_reviews()

if isinstance(results, list):

for review in results:

print(review)

else:

print(results)

elif choice == '7':

out\_of\_stock\_list = out\_of\_stock\_products(inventory\_file\_path)

for item in out\_of\_stock\_list:

print(item)

elif choice == '8':

example\_product\_ids = ['P001', 'P002', 'P003', 'P004', 'P005']

top\_products\_list = top\_rated\_products(example\_product\_ids)

for product in top\_products\_list:

print(product)

elif choice.lower() == 'q':

break

else:

print("Invalid option. Please try again.")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**OUPUT:**

E-commerce Data Processing System

1. Add Product

2. Update Product

3. Check Product Availability

4. Add Customer Review

5. Calculate Average Rating

6. Search Reviews

7. List Out of Stock Products

8. Show Top Rated Products

Select an option (or 'q' to quit): 1

Enter Product ID: 123

Enter Product Name: XYZ

Enter Category: GROCERY

Enter Price: 1000

Enter Stock Quantity: 9

Product added successfully.

E-commerce Data Processing System

1. Add Product

2. Update Product

3. Check Product Availability

4. Add Customer Review

5. Calculate Average Rating

6. Search Reviews

7. List Out of Stock Products

8. Show Top Rated Products

Select an option (or 'q' to quit): q