Contents

Question1 – Online Reviews Data Collection

1 Code Design and Workflow

- **Initialize Selenium**
  - Open Chrome browser and navigate to the IMDb movie review page.
- **Load Additional Reviews（the page only displays 25 reviews by default）**
  - Locate the "25 More" button.
  - Scroll to the position of it.
  - Click it to load more reviews dynamically.
- **Extract Review Data**
  - Select the top 30 review elements.
  - For each review, extract:
    - **Username**
    - **Review date**
    - **Review text**
    - **Helpful and non-helpful counts**
    - **Rating score (if exist: sometimes replaced by the user's expression)**
- **Store Data**
  - Save all extracted information in a list of dictionaries.
- **Export Data**
  - Convert the list into CSV and JSON formats.

2 Results Analysis

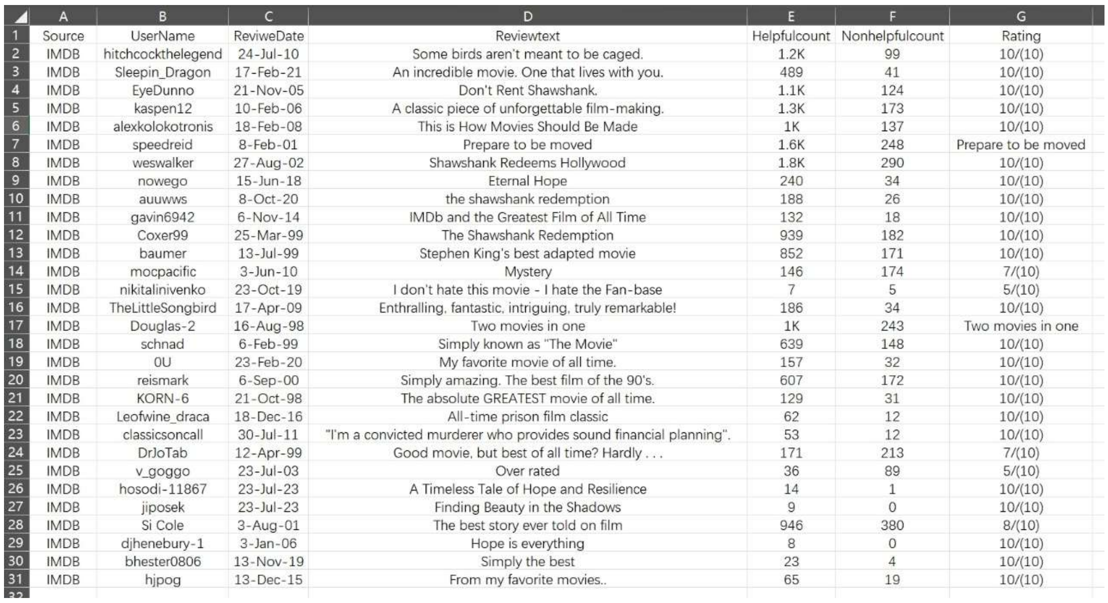
- **Reviews Collected**
  - Successfully collected 30 reviews as required.
- **Completeness**
  - All reviews include text, user, date, and helpfulness votes; some ratings are missing but handled appropriately.
  - All reviews include username, review date, review text, rating and helpful/non-helpful votes.
  - Some reviews do not have a numeric rating. The short text provided by the user is stored instead.
- **Data Quality**
  - Textual and numerical data are clean and structured.
  - multimedia elements are ignored.

3 Code

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
import pandas as pd
import json
import time

browser = webdriver.Chrome()
browser.get("https://www.imdb.com/title/tt0111161/reviews")
button = browser.find_element(By.CSS_SELECTOR, "button[class='ipc-btn ipc-btn--single-padding ipc-btn--center-align-content ipc-btn--default-height ipc-btn--core-base ipc-btn--theme-base ipc-btn--button-radius ipc-btn--on-accent2 ipc-text-button ipc-see-more__button']")
browser.execute_script('arguments[0].scrollIntoView(false);', button)
time.sleep(1)
button.click()
StoreData = []
amount = 0

time.sleep(2)
reviews = browser.find_elements(By.CSS_SELECTOR, "article.user-review-item")[:30]
for el in reviews:
    amount=amount+1
    username = el.find_element(By.CSS_SELECTOR, "a[data-testid='author-link']").text
    reviewdate = el.find_element(By.CSS_SELECTOR, "li[class='ipc-inline-list__item review-date']").text
    reviewtxt = el.find_element(By.CSS_SELECTOR, "h3[class='ipc-title__text ipc-title__text--reduced']").text
    reviewhelpfulcount = el.find_element(By.CSS_SELECTOR, "span[class='ipc-voting__label__count ipc-voting__label__count--up']").text
    reviewnonhelpfulcount = el.find_element(By.CSS_SELECTOR, "span[class='ipc-voting__label__count ipc-voting__label__count--down']").text
    rating = el.find_elements(By.CSS_SELECTOR, "span[class='ipc-rating-star--rating']")
    if rating:
        rating = rating[0].text + '/(10)'
    else:
        rating = el.find_element(By.CSS_SELECTOR, "h3[class='ipc-title__text ipc-title__text--reduced']").text

    StoreData.append({
        "Source": "IMDB",
        "UserName": username,
        "ReviweDate": reviewdate,
        "Reviewtext": reviewtxt,
        "Helpfulcount": reviewhelpfulcount,
        "Nonhelpfulcount": reviewnonhelpfulcount,
        "Rating": rating
        })

browser.quit()

filename = 'reviewIMDB.csv'
df = pd.DataFrame(StoreData)
df.to_csv(filename, index=False)

StoreData.append({
        "Amount": amount
        })
with open("reviewIMDB.json",'w',encoding='utf-8') as json_file:
    json.dump(StoreData,json_file,indent = 4, ensure_ascii=False)
```

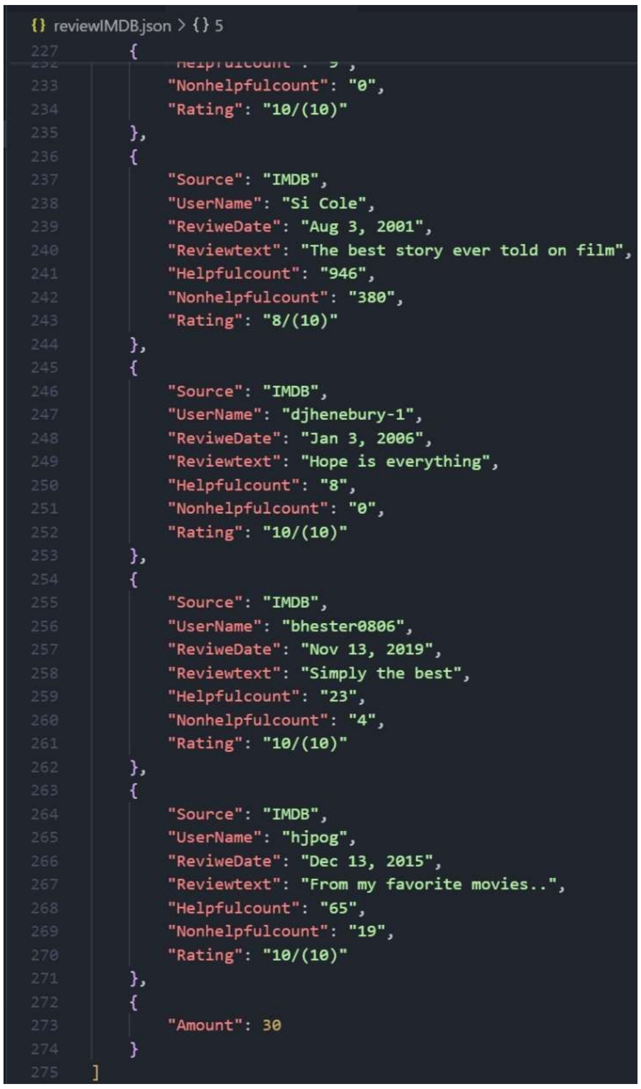
4 Results

- **CSV**



	A	B	C	D	E	F	G
1	Source	UserName	ReviweDate	Reviewtext	Helpfulcount	Nonhelpfulcount	Rating
2	IMDB	hitchcockthelegend	24-Jul-10	Some birds aren't meant to be caged.	1.2K	99	10/(10)
3	IMDB	Sleepin_Dragon	17-Feb-21	An incredible movie. One that lives with you.	489	41	10/(10)
4	IMDB	EyeDunno	21-Nov-05	Don't Rent Shawshank.	1.1K	124	10/(10)
5	IMDB	kaspen12	10-Feb-06	A classic piece of unforgettable film-making.	1.3K	173	10/(10)
6	IMDB	alexkolokotronis	18-Feb-08	This is How Movies Should Be Made	1K	137	10/(10)
7	IMDB	speedreid	8-Feb-01	Prepare to be moved	1.6K	248	Prepare to be moved
8	IMDB	weswalker	27-Aug-02	Shawshank Redeems Hollywood	1.8K	290	10/(10)
9	IMDB	nowego	15-Jun-18	Eternal Hope	240	34	10/(10)
10	IMDB	auuwws	8-Oct-20	the shawshank redemption	188	26	10/(10)
11	IMDB	gavin6942	6-Nov-14	IMDb and the Greatest Film of All Time	132	18	10/(10)
12	IMDB	Coxer99	25-Mar-99	The Shawshank Redemption	939	182	10/(10)
13	IMDB	baumer	13-Jul-99	Stephen King's best adapted movie	852	171	10/(10)
14	IMDB	mocpacific	3-Jun-10	Mystery	146	174	7/(10)
15	IMDB	nikitalinivenko	23-Oct-19	I don't hate this movie - I hate the Fan-base	7	5	5/(10)
16	IMDB	TheLittleSongbird	17-Apr-09	Enthralling, fantastic, intriguing, truly remarkable!	186	34	10/(10)
17	IMDB	Douglas-2	16-Aug-98	Two movies in one	1K	243	Two movies in one
18	IMDB	schnad	6-Feb-99	Simply known as "The Movie"	639	148	10/(10)
19	IMDB	0U	23-Feb-20	My favorite movie of all time.	157	32	10/(10)
20	IMDB	reismark	6-Sep-00	Simply amazing. The best film of the 90's.	607	172	10/(10)
21	IMDB	KORN-6	21-Oct-98	The absolute GREATEST movie of all time.	129	31	10/(10)
22	IMDB	Leofwine_draca	18-Dec-16	All-time prison film classic	62	12	10/(10)
23	IMDB	classicsoncall	30-Jul-11	"I'm a convicted murderer who provides sound financial planning".	53	12	10/(10)
24	IMDB	DrJoTab	12-Apr-99	Good movie, but best of all time? Hardly . . .	171	213	7/(10)
25	IMDB	v_goggo	23-Jul-03	Over rated	36	89	5/(10)
26	IMDB	hosodi-11867	23-Jul-23	A Timeless Tale of Hope and Resilience	14	1	10/(10)
27	IMDB	jiposek	23-Jul-23	Finding Beauty in the Shadows	9	0	10/(10)
28	IMDB	Si Cole	3-Aug-01	The best story ever told on film	946	380	8/(10)
29	IMDB	djhenebury-1	3-Jan-06	Hope is everything	8	0	10/(10)
30	IMDB	bhester0806	13-Nov-19	Simply the best	23	4	10/(10)
31	IMDB	hjpog	13-Dec-15	From my favorite movies..	65	19	10/(10)

- **JSON**



```
        {
            "Nonhelpfulcount": "0",
            "Rating": "10/(10)"
        },
        {
            "Source": "IMDB",
            "UserName": "Si Cole",
            "ReviweDate": "Aug 3, 2001",
            "Reviewtext": "The best story ever told on film",
            "Helpfulcount": "946",
            "Nonhelpfulcount": "380",
            "Rating": "8/(10)"
        },
        {
            "Source": "IMDB",
            "UserName": "djhenebury-1",
            "ReviweDate": "Jan 3, 2006",
            "Reviewtext": "Hope is everything",
            "Helpfulcount": "8",
            "Nonhelpfulcount": "0",
            "Rating": "10/(10)"
        },
        {
            "Source": "IMDB",
            "UserName": "bhester0806",
            "ReviweDate": "Nov 13, 2019",
            "Reviewtext": "Simply the best",
            "Helpfulcount": "23",
            "Nonhelpfulcount": "4",
            "Rating": "10/(10)"
        },
        {
            "Source": "IMDB",
            "UserName": "hjpog",
            "ReviweDate": "Dec 13, 2015",
            "Reviewtext": "From my favorite movies..",
            "Helpfulcount": "65",
            "Nonhelpfulcount": "19",
            "Rating": "10/(10)"
        },
        {
            "Amount": 30
        }
    ]
```

Question2 – Data Cleaning with Regular Expressions

1 Code Design and Workflow

- **Importing RE Module**
- **Task-wise Execution**
    - For each task, a dedicated regular expression is defined.
    - A test cases specified in the assignment is defined (test_cases1/2/…).
- **Matching**
    - Applying the regular expressions to each test string.

2 Results Analysis

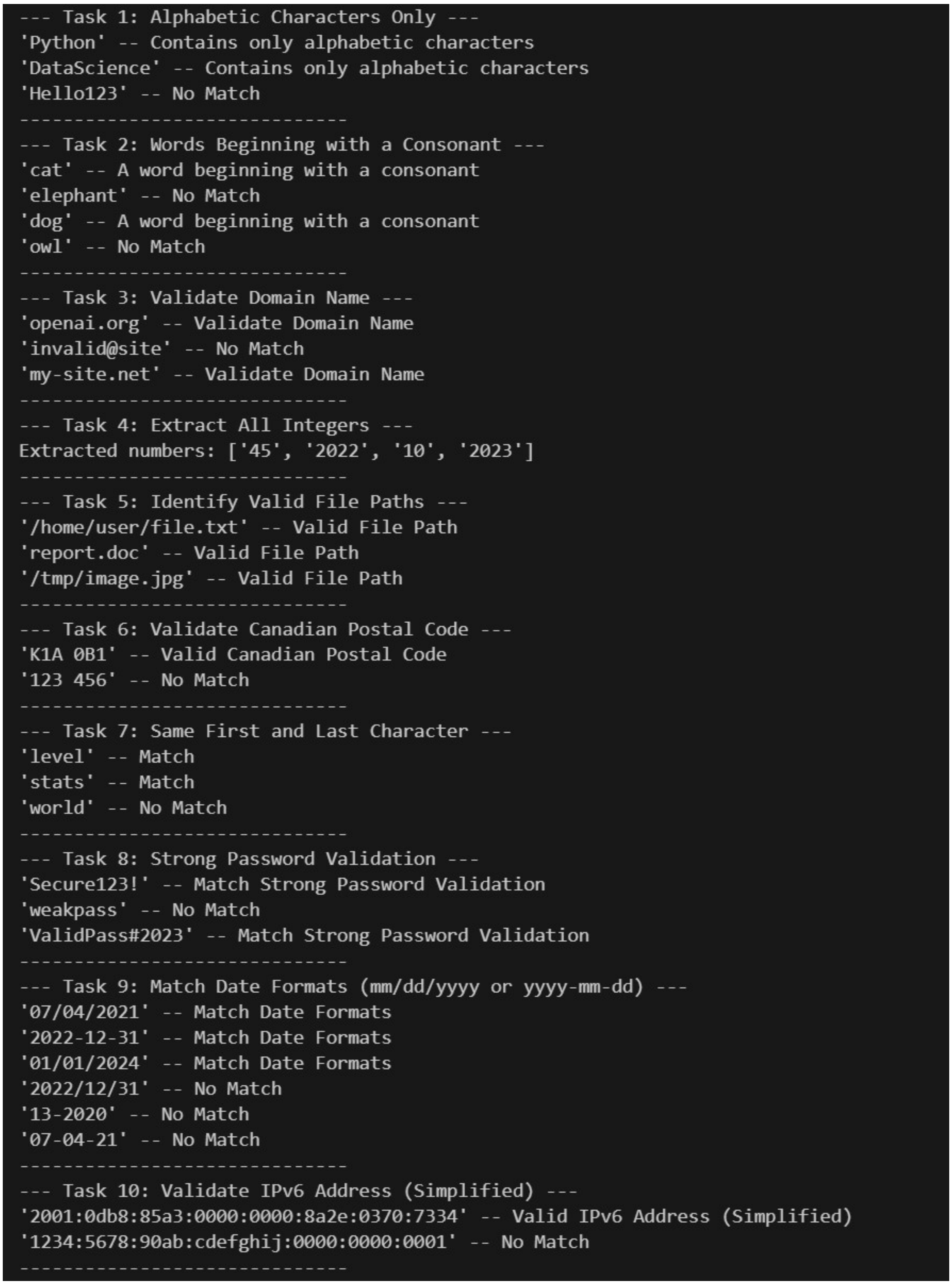
- **Comprehensive Coverage**
    - Successfully implements the corresponding regular expressions for all 10 tasks required by the assignment.
- **Accuracy**
    - All test results meet expectations.
- **Clear Outcome**
    - The code structure is clear.
    - The output clearly demonstrates the performance of each regular expression on its respective test cases.

3 Code

```
import re

print("--- Task 1: Alphabetic Characters Only ---")
REpattern1 = r'^[a-zA-Z]+$'
test_cases1 = ['Python', 'DataScience', 'Hello123']
for text in test_cases1:
    if re.fullmatch(REpattern1, text):
        print(f"'{text}' -- Contains only alphabetic characters")
    else:
        print(f"'{text}' -- No Match")
print("-" * 30)
print("--- Task 2: Words Beginning with a Consonant ---")
REpattern2 = r'\b[b-df-hj-np-tv-z]\w*'
test_cases2 = ["cat", "elephant", "dog", "owl"]
for text in test_cases2:
    if re.match(REpattern2, text, re.IGNORECASE):
        print(f"'{text}' -- A word beginning with a consonant")
    else:
        print(f"'{text}' -- No Match")
print("-" * 30)
print("--- Task 3: Validate Domain Name ---")
REpattern3 = r'^([a-zA-Z0-9]([a-zA-Z0-9-]*[a-zA-Z0-9])?\.)+[a-zA-Z]{2,}$'
test_cases3    = ['openai.org', 'invalid@site', 'my-site.net']
for text in test_cases3:
    if re.fullmatch(REpattern3, text):
        print(f"'{text}' -- Validate Domain Name")
    else:
        print(f"'{text}' -- No Match")
print("-" * 30)

print("--- Task 4: Extract All Integers ---")
REpattern4 = r'\d+'
test_case4 = 'He scored 45 goals in 2022 and 10 goals in 2023.'
numbers = re.findall(REpattern4, test_case4)
print(f"Extracted numbers: {numbers}")
print("-" * 30)

print("--- Task 5: Identify Valid File Paths ---")
REpattern5 = r'^(?:[a-zA-Z]:\\|/)?(?:[\w.-]*[a-zA-Z0-
9]+/)*[\w.-]+\.(txt|csv|jpg|doc)$'
test_cases5 = ['/home/user/file.txt', 'report.doc', '/tmp/image.jpg']
for text in test_cases5:
    if re.search(REpattern5, text):
        print(f"'{text}' -- Valid File Path")
    else:
        print(f"'{text}' -- No Match")
print("-" * 30)
print("--- Task 6: Validate Canadian Postal Code ---")
REpattern6 = r'^[A-Z]\d[A-Z]\s\d[A-Z]\d$'
test_cases6 = ['K1A 0B1', '123 456']
for test in test_cases6:
    if re.fullmatch(REpattern6, test):
        print(f"'{test}' -- Valid Canadian Postal Code")
    else:
        print(f"'{test}' -- No Match")
print("-" * 30)
print("--- Task 7: Same First and Last Character ---")
REpattern7 = r'^(\w).*\1$'
```

```
test_cases7 = ['level', 'stats', 'world']
for text in test_cases7:
    if re.fullmatch(REpattern7, text):
        print(f"'{text}' -- Match")
    else:
        print(f"'{text}' -- No Match")
print("-" * 30)
print("--- Task 8: Strong Password Validation ---")
REpattern8 = r'^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&#])[A-Za-
z\d@$!%*?&#]{10,}$'
test_cases8 = ['Secure123!', 'weakpass', 'ValidPass#2023']
for text in test_cases8:
    if re.fullmatch(REpattern8, text):
        print(f"'{text}' -- Match Strong Password Validation")
    else:
        print(f"'{text}' -- No Match")
print("-" * 30)
print("--- Task 9: Match Date Formats (mm/dd/yyyy or yyyy-mm-dd) ---")
REpattern9 = r'(\d{2}/\d{2}/\d{4}|\d{4}-\d{2}-\d{2})'
test_cases9 = ['07/04/2021', '2022-12-31', '01/01/2024', '2022/12/31', '13-2020', '07-04-
21']
for text in test_cases9:
    match = re.fullmatch(REpattern9, text)
    if match:
        print(f"'{text}' -- Match Date Formats")
    else:
        print(f"'{text}' -- No Match")
print("-" * 30)
print("--- Task 10: Validate IPv6 Address (Simplified) ---")
REpattern10 = r'^([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}$'
test_cases10 = ['2001:0db8:85a3:0000:0000:8a2e:0370:7334',
'1234:5678:90ab:cdefghij:0000:0000:0001']
for text in test_cases10:
    if re.fullmatch(REpattern10, text):
        print(f"'{text}' -- Valid IPv6 Address (Simplified)")
    else:
        print(f"'{text}' -- No Match")
print("-" * 30)
```

4 Results



```
--- Task 1: Alphabetic Characters Only ---
'Python' -- Contains only alphabetic characters
'DataScience' -- Contains only alphabetic characters
'Hello123' -- No Match
------------------------------
--- Task 2: Words Beginning with a Consonant ---
'cat' -- A word beginning with a consonant
'elephant' -- No Match
'dog' -- A word beginning with a consonant
'owl' -- No Match
------------------------------
--- Task 3: Validate Domain Name ---
'openai.org' -- Validate Domain Name
'invalid@site' -- No Match
'my-site.net' -- Validate Domain Name
------------------------------
--- Task 4: Extract All Integers ---
Extracted numbers: ['45', '2022', '10', '2023']
------------------------------
--- Task 5: Identify Valid File Paths ---
'/home/user/file.txt' -- Valid File Path
'report.doc' -- Valid File Path
'/tmp/image.jpg' -- Valid File Path
------------------------------
--- Task 6: Validate Canadian Postal Code ---
'K1A 0B1' -- Valid Canadian Postal Code
'123 456' -- No Match
------------------------------
--- Task 7: Same First and Last Character ---
'level' -- Match
'stats' -- Match
'world' -- No Match
------------------------------
--- Task 8: Strong Password Validation ---
'Secure123!' -- Match Strong Password Validation
'weakpass' -- No Match
'ValidPass#2023' -- Match Strong Password Validation
------------------------------
--- Task 9: Match Date Formats (mm/dd/yyyy or yyyy-mm-dd) ---
'07/04/2021' -- Match Date Formats
'2022-12-31' -- Match Date Formats
'01/01/2024' -- Match Date Formats
'2022/12/31' -- No Match
'13-2020' -- No Match
'07-04-21' -- No Match
------------------------------
--- Task 10: Validate IPv6 Address (Simplified) ---
'2001:0db8:85a3:0000:0000:8a2e:0370:7334' -- Valid IPv6 Address (Simplified)
'1234:5678:90ab:cdefghij:0000:0000:0001' -- No Match
------------------------------
```

Question3 – Data Processing

1 Code Design and Workflow

- **XML to Line-Based TXT**
  - XML Parsing[xml.etree.ElementTree module is used to parse the XML content].
  - Text Extraction[The XML tree is traversed to locate and extract all text contained within 'seg' tag].
  - Text Cleaning.
    - Convert all text to lowercase using the .lower() method.
    - Remove all punctuation using string.punctuation and str.translate(), then use re.sub() to clean up extra spaces.
  - Saving Results
- **BPE Vocabulary Creation**
  - Tool Preparation.
  - learn_bpe.py Execution.
    - Set the number of BPE merge operations.
    - Set the standard output to the vocabulary.bpe file.

2 Results Analysis

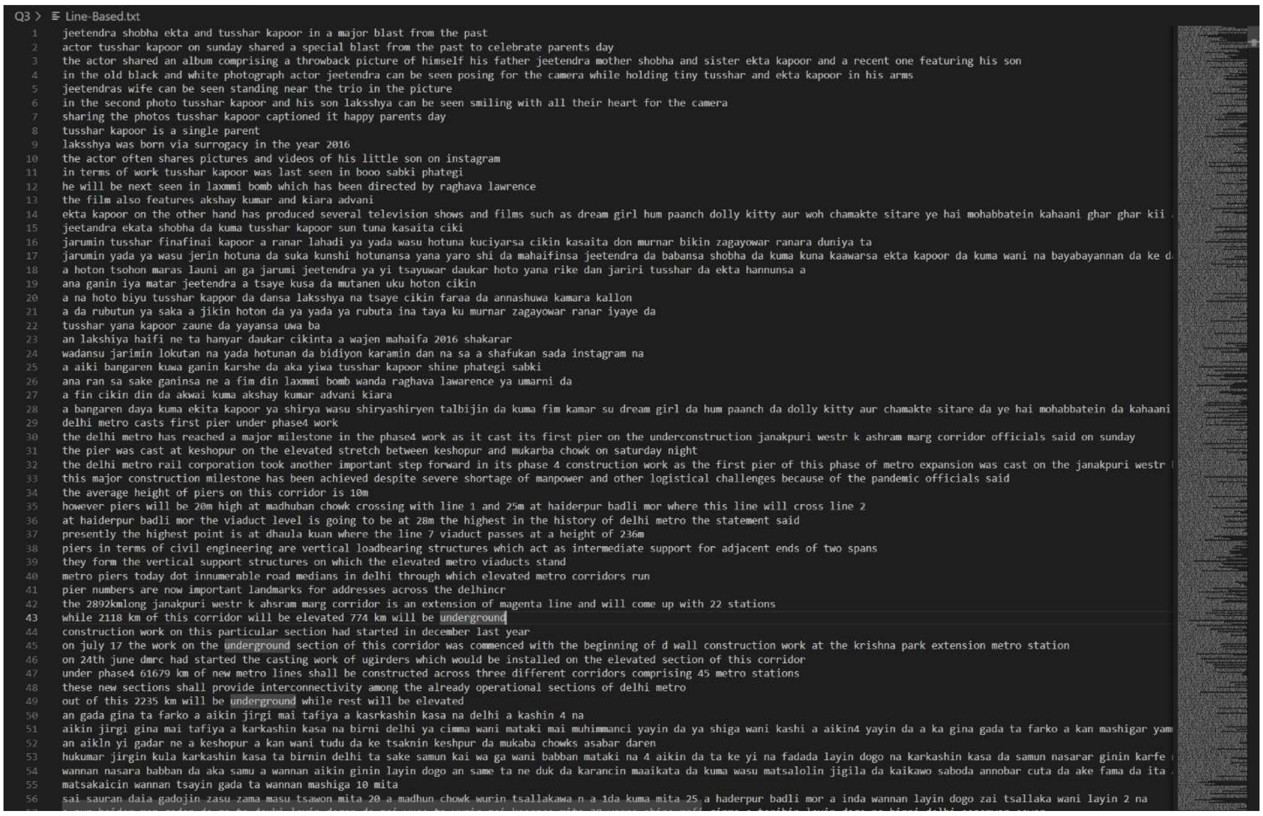
- **XML Conversion**
  - Successfully converts structured XML data into line-based text. All tags are removed, and the text is converted to lowercase without punctuation.
- **BPE Vocabulary Generation**
  - Successfully generates a BPE vocabulary from the cleaned text.

3 Code

```
import xml.etree.ElementTree as ET
import re
import string

XmlFile = 'Q3/samplehyp.xml'
LineBased = 'Q3/Line-Based.txt'
tree = ET.parse(XmlFile)
root = tree.getroot()

CleanedLines = []
for seg in root.findall('.//seg'):
    if seg.text:
        text = seg.text
        text = text.lower()
        text = text.translate(str.maketrans('', '', string.punctuation))
        text = re.sub(r'\s+', ' ', text).strip()
        if text:
            CleanedLines.append(text)

with open(LineBased, 'w', encoding='utf-8') as f:
    for line in CleanedLines:
        f.write(line + '\n')

#Shell commands to create BPE vocabulary
git clone https://github.com/rsennrich/subword-nmt.git
python subword-nmt/subword_nmt/learn_bpe.py -s 2000 < Line-Based.txt >
vocabulary.bpe
```

4 Results

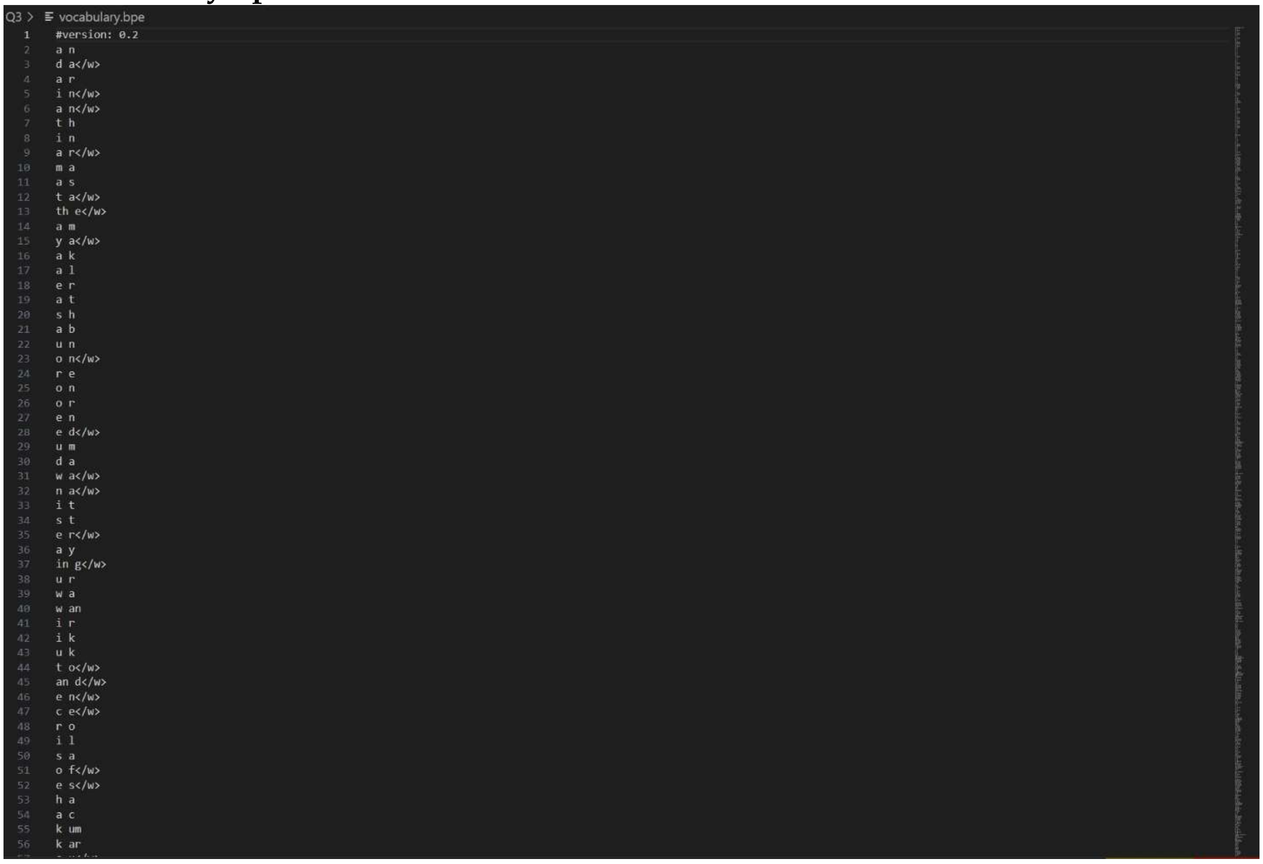
- **Line-Based.txt**



Q3 > Line-Based.txt

```
jeetendra shobha ekta and tusshar kapoor in a major blast from the past
actor tusshar kapoor on sunday shared a special blast from the past to celebrate parents day
the actor shared an album comprising a throwback picture of himself his father jeetendra mother shobha and sister ekta kapoor and a recent one featuring his son
in the old black and white photograph actor jeetendra can be seen posing for the camera while holding tiny tusshar and ekta kapoor in his arms
jeetendras wife can be seen standing near the trio in the picture
in the second photo tusshar kapoor and his son laksshya can be seen smiling with all their heart for the camera
sharing the photos tusshar kapoor captioned it happy parents day
tusshar kapoor is a single parent
laksshya was born via surrogacy in the year 2016
the actor often shares pictures and videos of his little son on instagram
in terms of work tusshar kapoor was last seen in booo sabki phategi
he will be next seen in laxmmi bomb which has been directed by raghava lawrence
the film also features akshay kumar and kiara advani
ekta kapoor on the other hand has produced several television shows and films such as dream girl hum paanch dolly kitty aur woh chamakte sitare ye hai mohabbatein kahaani ghar ghar kii
jeetandra ekata shobha da kuma tusshar kapoor sun tuna kasaita ciki
jarumin tusshar finafinai kapoor a ranar lahadi ya yada wasu hotuna kuciyarsa cikin kasaita don murnar bikin zagayowar ranara duniya ta
jarumin yada ya wasu jerin hotuna da suka kunshi hotunansa yana yaro shi da mahaifinsa jeetendra da babansa shobha da kuma kuna kaawarsa ekta kapoor da kuma wani na bayabayannan da ke d
a hoton tsohon maras launi an ga jarumi jeetendra ya yi tsayuwar daukar hoto yana rike dan jariri tusshar da ekta hannunsa a
ana ganin iya matar jeetendra a tsaye kusa da mutanen uku hoton cikin
a na hoto biyu tusshar kappor da dansa laksshya na tsaye cikin faraa da annashuwa kamara kallon
a da rubutun ya saka a jikin hoton da ya yada ya rubuta ina taya ku murnar zagayowar ranar iyaye da
tusshar yana kapoor zaune da yayansa uwa ba
an lakshiya haifi ne ta hanyar daukar cikinta a wajen mahaifa 2016 shakarar
wadansu jarimin lokutan na yada hotunan da bidiyon karamin dan na sa a shafukan sada instagram na
a aiki bangaren kuwa ganin karshe da aka yiwa tusshar kapoor shine phategi sabki
ana ran sa sake ganinsa ne a fim din laxmmi bomb wanda raghava lawarence ya umarni da
a fin cikin din da akwai kuma akshay kumar advani kiara
a bangaren daya kuma ekita kapoor ya shirya wasu shiryashiryen talbijin da kuma fim kamar su dream girl da hum paanch da dolly kitty aur chamakte sitare da ye hai mohabbatein da kahaani
delhi metro casts first pier under phase4 work
the delhi metro has reached a major milestone in the phase4 work as it cast its first pier on the underconstruction janakpuri westr k ashram marg corridor officials said on sunday
the pier was cast at keshopur on the elevated stretch between keshopur and mukarba chowk on saturday night
the delhi metro rail corporation took another important step forward in its phase 4 construction work as the first pier of this phase of metro expansion was cast on the janakpuri westr
this major construction milestone has been achieved despite severe shortage of manpower and other logistical challenges because of the pandemic officials said
the average height of piers on this corridor is 10m
however piers will be 20m high at madhuban chowk crossing with line 1 and 25m at haiderpur badli mor where this line will cross line 2
at haiderpur badli mor the viaduct level is going to be at 28m the highest in the history of delhi metro the statement said
presently the highest point is at dhaula kuan where the line 7 viaduct passes at a height of 236m
piers in terms of civil engineering are vertical loadbearing structures which act as intermediate support for adjacent ends of two spans
they form the vertical support structures on which the elevated metro viaducts stand
metro piers today dot innumerable road medians in delhi through which elevated metro corridors run
pier numbers are now important landmarks for addresses across the delhincr
the 2892kmlong janakpuri westr k ahsram marg corridor is an extension of magenta line and will come up with 22 stations
while 2118 km of this corridor will be elevated 774 km will be underground
construction work on this particular section had started in december last year
on july 17 the work on the underground section of this corridor was commenced with the beginning of d wall construction work at the krishna park extension metro station
on 24th june dmrc had started the casting work of ugirders which would be installed on the elevated section of this corridor
under phase4 61679 km of new metro lines shall be constructed across three different corridors comprising 45 metro stations
these new sections shall provide interconnectivity among the already operational sections of delhi metro
out of this 2235 km will be underground while rest will be elevated
an gada gina ta farko a aikin jirgi mai tafiya a kasrkashin kasa na delhi a kashin 4 na
aikin jirgi gina mai tafiya a karkashin kasa na birni delhi ya cimma wani mataki mai muhimmanci yayin da ya shiga wani kashi a aikin4 yayin da a ka gina gada ta farko a kan mashigar yam
an aikln yi gadar ne a keshopur a kan wani tudu da ke tsaknin keshpur da mukaba chowks asabar daren
hukumar jirgin kula karkashin kasa ta birnin delhi ta sake samun kai wa ga wani babban mataki na 4 aikin da ta ke yi na fadada layin dogo na karkashin kasa da samun nasarar ginin karfe
wannan nasara babban da aka samu a wannan aikin ginin layin dogo an same ta ne duk da karancin maaikata da kuma wasu matsalolin jigila da kaikawo saboda annobar cuta da ake fama da ita
matsakaicin wannan tsayin gada ta wannan mashiga 10 mita
sai sauran daia gadojin zasu zama masu tsawon mita 20 a madhun chowk wurin tsallakawa n a 1da kuma mita 25 a haderpur badli mor a inda wannan layin dogo zai tsallaka wani layin 2 na
```

- **vocabulary.bpe**



Q3 > vocabulary.bpe

```
#version: 0.2
a n
d a</w>
a r
i n</w>
a n</w>
t h
i n
a r</w>
m a
a s
t a</w>
th e</w>
a m
y a</w>
a k
a l
e r
a t
s h
a b
u n
o n</w>
r e
o n
o r
e n
e d</w>
u m
d a
w a</w>
n a</w>
i t
s t
e r</w>
a y
in g</w>
u r
w a
w an
i r
i k
u k
t o</w>
an d</w>
e n</w>
c e</w>
r o
i l
s a
o f</w>
e s</w>
h a
a c
k um
k ar
```

Question4 – Data Visualization

1 Appropriate Visualization Methods (Answer to Q4--->1)

- **Explore the distribution of each attribute**
  - Major/Gender(categorical variable)[Bar charts or pie charts].
  - GPA (continuous variable)[Histograms].
  - ID(Integer) [unique identifiers || do not have statistical distribution significance, so they are generally not visualized].
- **Explore the distribution of relationships between attributes**
  - GPA vs. Major/ GPA vs. Gender (continuous vs. categorical)[Grouped box plots].
  - Major vs. Gender(categorical vs. categorical)[Grouped bar chart or Heatmap].

2 Code Design and Workflow

- **Importing libraries**
  - pandas
  - numpy
  - matplotlib.pyplot
  - seaborn
- **Generating student data**
  - Create 500 records[use my real studentId ' 59790558' as random seed].
  - Student ID:Integers from 1 to 500.
  - Major: Randomly chosen from 'Computer Science', 'Mathematics', 'Physics'.
  - Gender: Randomly chosen from 'Male' and 'Female'.
  - GPA: Random numbers generated from a normal distribution within the range 0.0–4.0.
- **Visualization**
  - Use seaborn.countplot to create bar charts showing the number of students per major and per gender.
  - Use plt.pie to create pie charts showing the proportion of students' number per major and per gender.
  - Use seaborn.histplot to create a histogram showing the overall distribution of GPA.
  - Use seaborn.boxplot to create grouped box plots to compare GPA distributions across majors and genders.

  - Use other functions to compare distributions.
  - All charts include titles and labels for better readability and are saved as PNG images.
- **Compute and Visualize the Similarity Matrix(Q4-→4)**
  - Use numpy.random.rand to initialize two 5×8 matrices: U and V.
  - Compute the similarity matrix according to the formula: Similarity(U,V).
  - Write a softmax function to perform the calculation
  - Use seaborn.heatmap to visualize the final similarity score matrix

3 Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os

np.random.seed(59790558) # for reproducibility
majors = ['Computer Science', 'Mathematics', 'Physics']
genders = ['Male', 'Female']

student_data = {
	'Student ID': range(1, 501),
	'Major': np.random.choice(majors, 500, p=[0.4, 0.3, 0.3]),
	'Gender': np.random.choice(genders, 500),
	'GPA': np.clip(np.random.normal(loc=3.2, scale=0.5, size=500), 0.0, 4.0)
}
students = pd.DataFrame(student_data)

plt.figure(figsize=(10, 6))
sns.countplot(x='Major', data=students,
order=students['Major'].value_counts().index)
plt.title('Distribution of Students by Major')
plt.xlabel('Major')
plt.ylabel('Count')
plt.savefig('q4_2_by_major_bar.png')
plt.show()

plt.figure(figsize=(8, 8))
major_counts = students['Major'].value_counts()
```

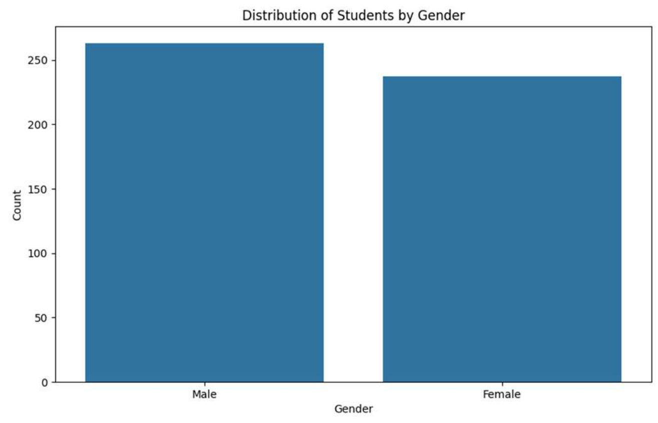
```
plt.pie(major_counts, labels=major_counts.index, autopct='%1.1f%%',
startangle=140)
plt.title('Distribution of Students by Major')
plt.ylabel('')
plt.savefig('q4_2_by_major_pie.png')
plt.show()

plt.figure(figsize=(10, 6))
sns.countplot(x='Gender', data=students,
order=students['Gender'].value_counts().index)
plt.title('Distribution of Students by Gender')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.savefig('q4_2_by_gender_bar.png')
plt.show()

plt.figure(figsize=(8, 8))
gender_counts = students['Gender'].value_counts()
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%',
startangle=140)
plt.title('Distribution of Students by Gender')
plt.ylabel('')
plt.savefig('q4_2_by_gender_pie.png')
plt.show()

plt.figure(figsize=(10, 6))
sns.histplot(students['GPA'], bins=20, kde=True)
plt.title('Distribution of Students by GPA')
plt.xlabel('GPA')
plt.ylabel('Frequency')
plt.savefig('q4_2_by_gpa_histogram.png')
plt.show()

plt.figure(figsize=(10, 6))
sns.boxplot(x='Major', y='GPA', data=students)
plt.title('GPA_vs_Major')
plt.xlabel('Major')
plt.ylabel('GPA')
plt.savefig('q4_GPA_vs_Major.png')
plt.show()

plt.figure(figsize=(10, 6))
sns.boxplot(x='Gender', y='GPA', data=students)
```

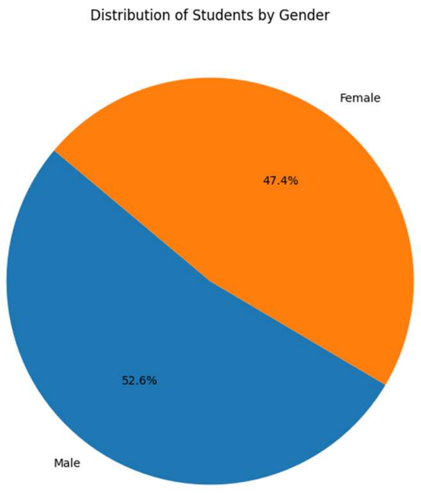
```
plt.title('GPA_vs_Gender')
plt.xlabel('Gender')
plt.ylabel('GPA')
plt.savefig('q4_GPA_vs_Gender.png')
plt.show()

plt.figure(figsize=(12, 7))
sns.countplot(x='Major', hue='Gender', data=students)
plt.title('Major_vs_Gender')
plt.xlabel('Major')
plt.ylabel('Count')
plt.legend(title='Gender')
plt.savefig('q4_Major_vs_Gender.png')
plt.show()

major_gender_crosstab = pd.crosstab(students['Major'], students['Gender'])
plt.figure(figsize=(10, 7))
sns.heatmap(major_gender_crosstab, annot=True, fmt='d', cmap='YlGnBu')
plt.title('Major_vs_Gender Heatmap')
plt.xlabel('Gender')
plt.ylabel('Major')
plt.savefig('q4_Major_vs_Gender_heatmap.png')
plt.show()

plt.figure(figsize=(10, 6))
labels = sns.countplot(x='Major', data=students,
order=students['Major'].value_counts().index)
for count in labels.containers:
    labels.bar_label(count, fmt='%d', label_type='edge', padding=3)
plt.title('Compute the number of students in each major')
plt.xlabel('Major')
plt.ylabel('Count')
plt.savefig('q4_3_by_major_bar.png')
plt.show()
def softmax(x):
    e_x = np.exp(x - np.max(x, axis=1, keepdims=True))
    return e_x / e_x.sum(axis=1, keepdims=True)

d = 8
U = np.random.rand(5, d)
V = np.random.rand(5, d)
similarity_matrix = softmax(np.dot(U, V.T) / np.sqrt(d))
```

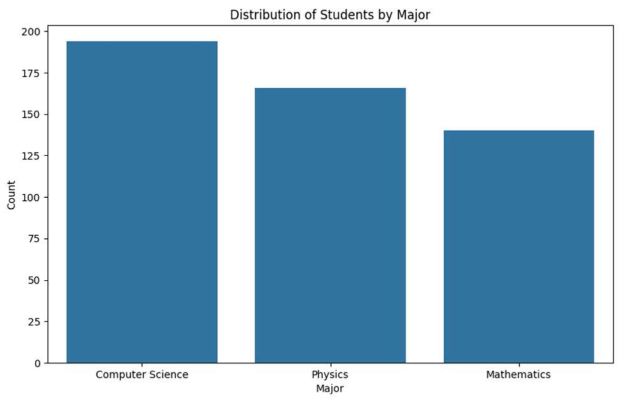
```
plt.figure(figsize=(10, 8))
sns.heatmap(similarity_matrix, annot=True, cmap='viridis', fmt=".2f")
plt.title('U V Correlation Heatmap')
plt.xlabel('Item')
plt.ylabel('User')
plt.savefig('q4_similarity_heatmap.png')
plt.show()
```

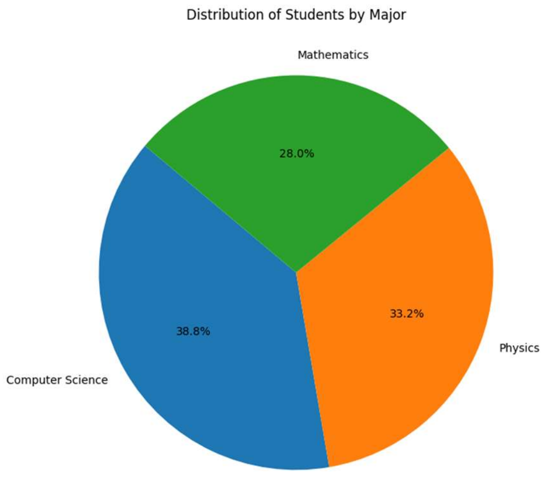
4 Results

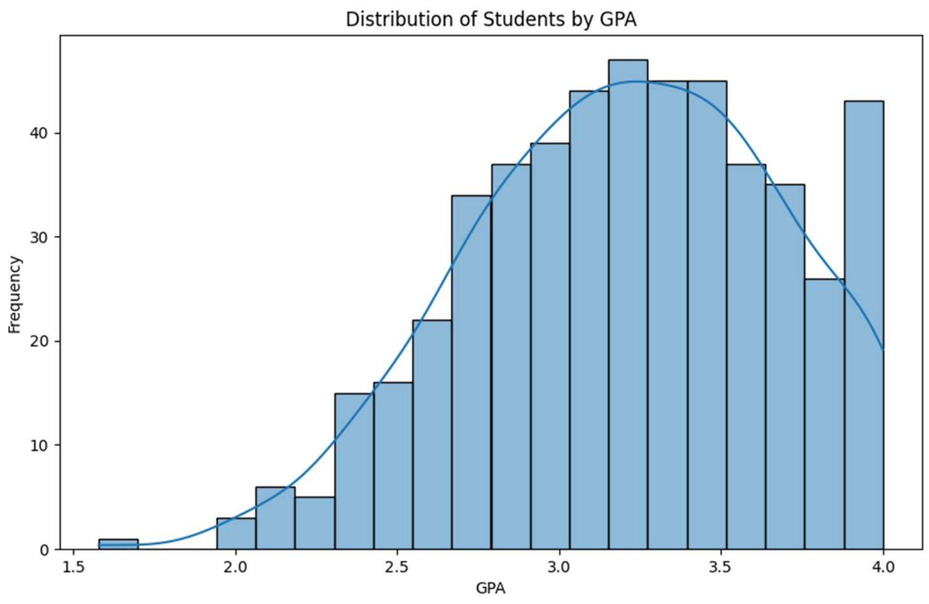
- **4.2**

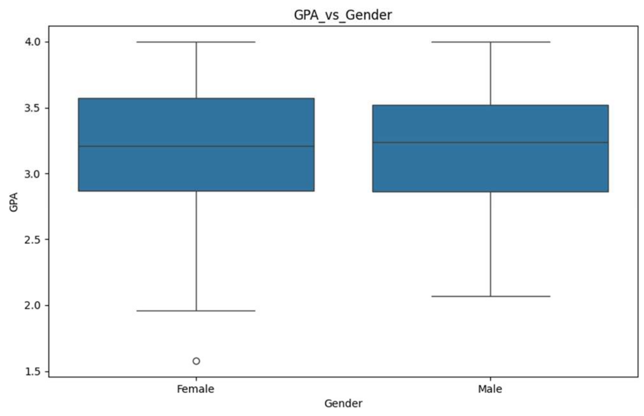


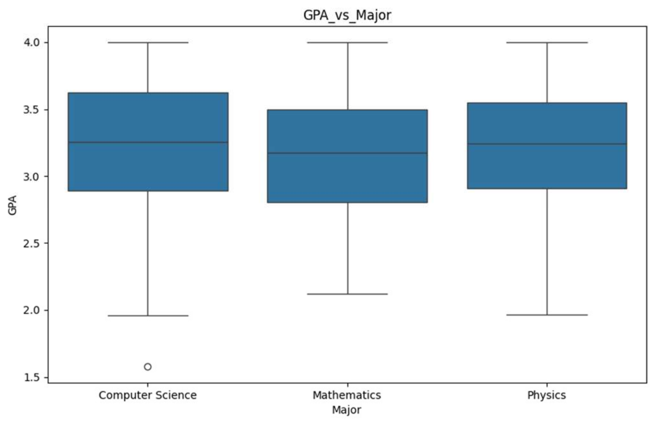


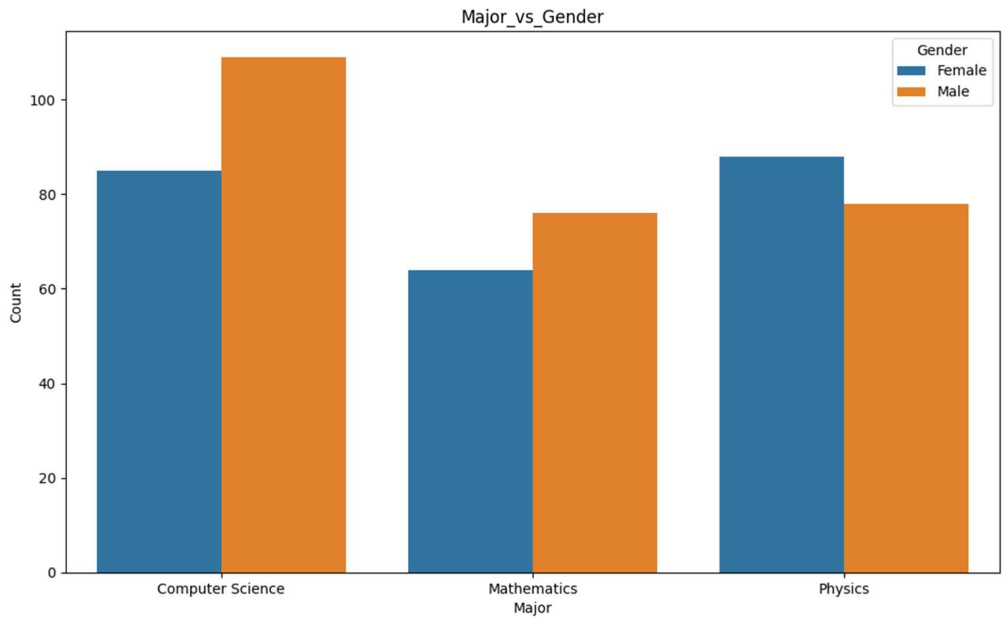


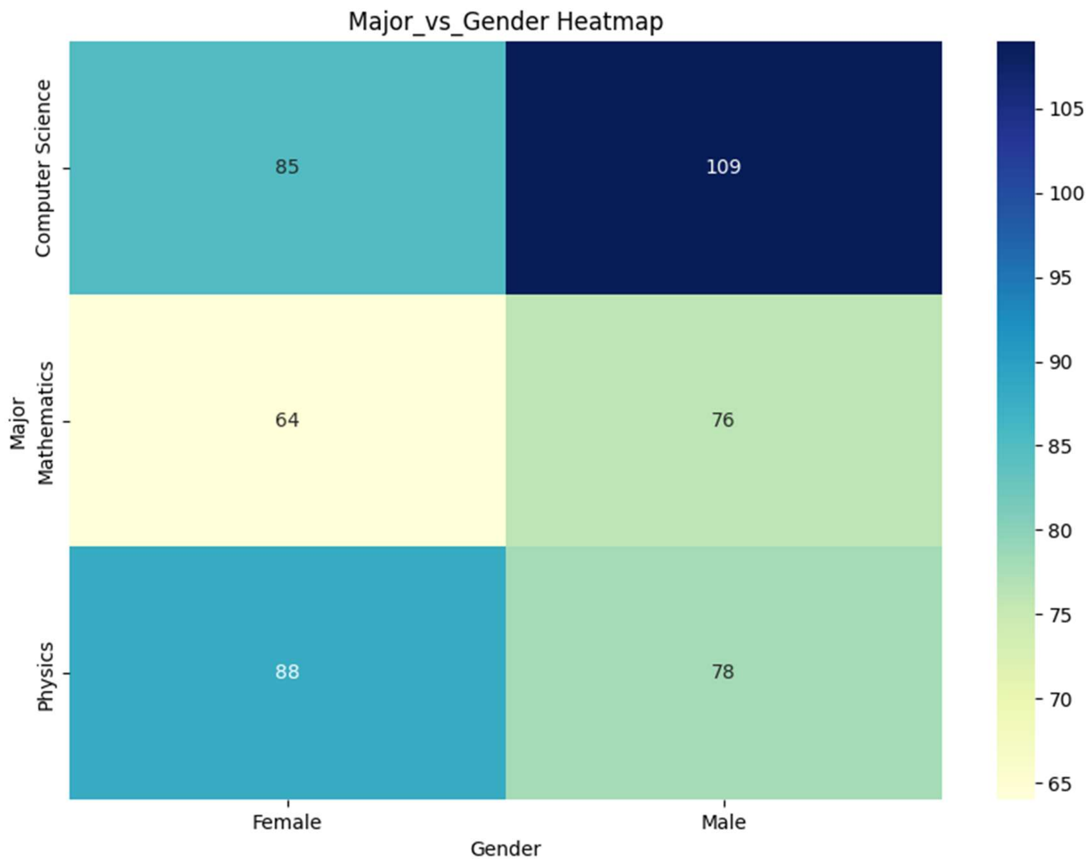



Distribution of Students by GPA
Frequency
GPA
0
10
20
30
40
1.5
2.0
2.5
3.0
3.5
4.0

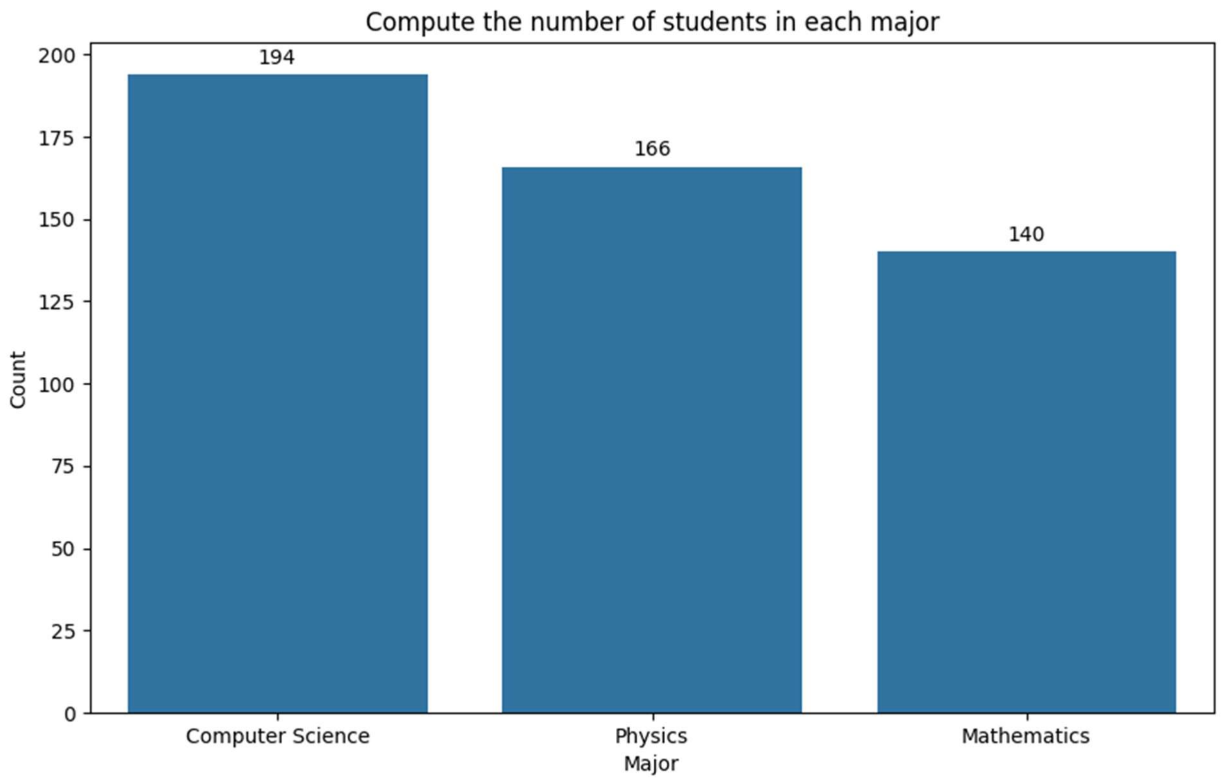

GPA_vs_Gender
GPA
Gender
Female
Male
1.5
2.0
2.5
3.0
3.5
4.0


GPA_vs_Major
GPA
Major
Computer Science
Mathematics
Physics
1.5
2.0
2.5
3.0
3.5
4.0

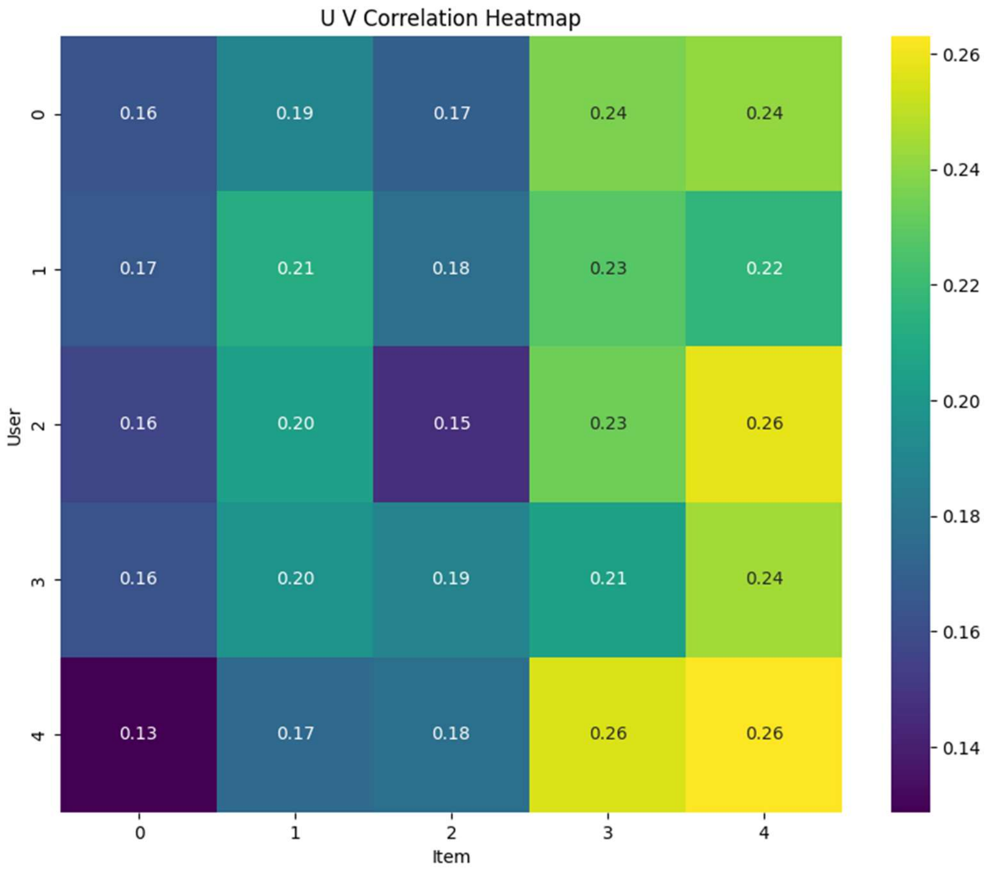

Major_vs_Gender
Count
Major
Gender
Female
Male
Computer Science
Mathematics
Physics
0
20
40
60
80
100


Major_vs_Gender Heatmap
Computer Science
Mathematics
Physics
Major
Female
Male
Gender
85
109
64
76
88
78
105
100
95
90
85
80
75
70
65

- **4.3**


Compute the number of students in each major
194
166
140
Count
Computer Science
Physics
Mathematics
Major
0
25
50
75
100
125
150
175
200

- **4.4**


U V Correlation Heatmap
User
Item
0
1
2
3
4
0.16
0.19
0.17
0.24
0.24
0.17
0.21
0.18
0.23
0.22
0.16
0.20
0.15
0.23
0.26
0.16
0.20
0.19
0.21
0.24
0.13
0.17
0.18
0.26
0.26
0.14
0.16
0.18
0.20
0.22
0.24
0.26