Джерела даних у програмних проєктах

1. ***1 бал*** ***(JSON)*** Підготовка даних у JSON-форматі для візуалізації проходить у декілька етапів. Візьмемо за приклад датасет порушень правил дорожнього руху в окрузі Монтгомері штату Меріленд обсягом близько 1.2Гб. Основні запитання, на які може дати відповіді цей набір даних:
   - Які види автомобілів найчастіше штрафуються за перевищення швидкості?
   - У який період часу поліція найбільш активна?
   - Наскільки поширені частини дороги, оснащені радарами? Чи штрафні квитанції більш-менш рівномірно розподілені географічно?
   - За що найчастіше штрафують людей?

   Поки що структура файлу невідома, тому першим кроком стане дослідження перших кількох рядків файлу з метою виявлення структурованості інформації.

```
{ "meta" : {
"view" : {
"id" : "4mse-ku6q",
"name" : "Traffic Violations",
"averageRating" : 0,
"category" : "Public Safety",
"createdAt" : 1403103517,
"description" : "This dataset contains traffic violation information from
all electronic traffic violations issued in the County. Any information that
can be used to uniquely identify the vehicle, the vehicle owner or the
officer issuing the violation will not be published.\r\n\r\nUpdate
Frequency: Daily",
"displayType" : "table",
```

   Бачимо, що JSON-дані є добре відформатованим словником. На найвищому рівні розташовується ключ meta з відступом у 2 пробіли. Скористаємось цим для знаходження всіх високорівневих ключів шляхом пошуку рядків, які мають спереду мають два пробільних символи та відкриваючі лапки. В ОС Windows це можна зробити в консолі CMD за допомогою команди

   findstr /R /C:"^ \"" rows.json

   Результат може виглядати так:

```
E:\>findstr /R /C:"^  \"" rows.json
  "meta" : {
  "data" : [ [ "row-aqx6~tzah-4a2i", "00000000-0000-0000-908C-A6A938AE6B2E", 0, 1565791383, null
, 1565792190, null, "{ }", "02ccadf1-7ebd-48d8-a793-913e3198f52a", "2019-07-30T00:00:00", "22:15
:00", "MCP", "2nd District, Bethesda", "EXCEEDING THE POSTED SPEED LIMIT OF 35 MPH", "RIVER RD/R
OYAL DOMINION DR", "38.9901016666667", "-77.151645", "No", "No", "No", "No", "No", "No", "No", "
No", "No", "No", null, null, null, null, null, null, null, "VA", "02 - Automobile", "2014", "HON
DA", "CIVIC", "BLACK", "Warning", "21-801.1", "Transportation Article", "False", "ASIAN", "F", "
FAIRFAX", "VA", "VA", "Q - Marked Laser", "1", [ "{\"address\": \"\", \"city\": \"\", \"state\":
 \"\", \"zip\": \"\"}", "38.9901016666667", "-77.151645", null, false ], "1", "12", "103", "1" ]
```

   Для bash команда виглядатиме так:

```
grep -E '^ {2}"' rows.json
```

Він показує, що meta і data – високорівневі ключі в файлі rows.json. *Зробіть аналогічний скриншот для своєї системи*.

Повну структуру ключів можна вивести, якщо задати пошук за регулярним виразом на 2-6 пробілів та символ відкриваючих лапок.

```
grep -E '^ {2,6}"' rows.json
```

На жаль, findstr не підтримує такий запис. Проте загальну структуру ключів добре видно і при простому відкритті документу:

```
{
  "meta" : {
    "view" : {
      "id" : "4mse-ku6q",
      "name" : "Traffic Violations",
      "averageRating" : 0,
      "category" : "Public Safety",
      "createdAt" : 1403103517,
      "description" : "This dataset contains traffic violation information from all electronic traffic violations issu
be used to uniquely identify the vehicle, the vehicle owner or the officer issuing the violation will not be published
      "displayType" : "table",
      "downloadCount" : 105578,
      "hideFromCatalog" : false,
      "hideFromDataJson" : false,
      "indexUpdatedAt" : 1565791377,
      "newBackend" : true,
      "numberOfComments" : 0,
      "oid" : 32145791,
      "provenance" : "official",
      "publicationAppendEnabled" : false,
      "publicationDate" : 1565795260,
      "publicationGroup" : 1620779,
      "publicationStage" : "published",
      "rowsUpdatedAt" : 1591607415,
      "rowsUpdatedBy" : "ajn4-zy65",
      "tableId" : 16433971,
      "totalTimesRated" : 0,
      "viewCount" : 46994,
      "viewLastModified" : 1565797518,
      "viewType" : "tabular",
      "approvals" : [ {
        "reviewedAt" : 1554399267,
        "reviewedAutomatically" : true,
```

Бібліотека ijson дозволяє зручно обробляти вміст json-файлів, зокрема метод items() дає змогу отримати список елементів з файлу. Кожний окремий елемент є, в даному випадку, елементом списку meta.view.columns. Виклик items() повертає генератор, тому зведемо його до вигляду списку та виведемо перший елемент json-структури:

```
import ijson
filename = "E:\\rows.json"
with open(filename, 'r') as f:
    objects = ijson.items(f, 'meta.view.columns.item')
    columns = list(objects)
    print(columns[0])
    column_names = [col["fieldName"] for col in columns]
    print(column_names)
```

Спробуйте запустити такий код на своїй машині для своєї копії файлу. Додайте код та скриншот результату його роботи до звіту. Вивід має бути таким:
{'id': -1, 'name': 'sid', 'dataTypeName': 'meta_data', 'fieldName': ':sid', 'position': 0, 'renderTypeName': 'meta_data', 'format': {}, 'flags': ['hidden']}

Бачимо, що отримали словники в якості елементів columns. Тому для отримання назв стовпців з даними будемо виділяти назву за допомогою релевантного ключа fieldName для кожного елементу списку columns:

[':sid', ':id', ':position', ':created_at', ':created_meta', ':updated_at', ':updated_meta', ':meta', 'seq_id', 'date_of_stop', 'time_of_stop', 'agency', 'subagency', 'description', 'location', 'latitude', 'longitude', 'accident', 'belts', 'personal_injury', 'property_damage', 'fatal', 'commercial_license', 'hazmat', 'commercial_vehicle', 'alcohol', 'work_zone', 'search_conducted', 'search_disposition', 'search_outcome', 'search_reason', 'search_reason_for_stop', 'search_type', 'search_arrest_reason', 'state', 'vehicle_type', 'year', 'make', 'model', 'color', 'violation_type', 'charge', 'article', 'contributed_to_accident', 'race', 'gender', 'driver_city', 'driver_state', 'dl_state', 'arrest_type', ':@computed_region_vu5j_pcmz', 'geolocation', ':@computed_region_tx5f_5em3', ':@computed_region_kbsp_ykn9', ':@computed_region_d7bw_bq6x', ':@computed_region_rbt8_3x7n']

Чудово! Звідси можемо обрати потрібні для візуалізації стовпчики! Продовжуємо попередній код з відібраними стовпцями:

```
good_columns = [
    "date_of_stop",
    "time_of_stop",
    "agency",
    "subagency",
    "description",
    "location",
    "latitude",
    "longitude",
    "vehicle_type",
    "year",
    "make",
    "model",
    "color",
    "violation_type",
    "race",
    "gender",
    "driver_state",
    "driver_city",
    "dl_state",
```

```
    "arrest_type"]
data = []
with open(filename, 'r') as f:
    objects = ijson.items(f, 'data.item')
    for row in objects:
        selected_row = []
        for item in good_columns:
            selected_row.append(row[column_names.index(item)])
            data.append(selected_row)
print(data[0])
```

Вивід програми буде наступним:
['2019-07-30T00:00:00', '22:15:00', 'MCP', '2nd District, Bethesda', 'EXCEEDING THE POSTED SPEED LIMIT OF 35 MPH', 'RIVER RD/ROYAL DOMINION DR', '38.9901016666667', '-77.151645', '02 - Automobile', '2014', 'HONDA', 'CIVIC', 'BLACK', 'Warning', 'ASIAN', 'F', 'VA', 'FAIRFAX', 'VA', 'Q - Marked Laser']

Якби датасет був набагато більшим, можна було б розбивати дані на пакети по 1000000 рядків, щоб обмежити споживання оперативної пам’яті та прискорити обробку.

Наступний крок – зчитування даних у Pandas для більш ефективних зберігання та обробки.

```
import pandas as pd
stops = pd.DataFrame(data, columns=good_columns)
print(stops["color"].value_counts())
print(stops["arrest_type"].value_counts())
```

У коді продемонстровано простий аналіз – підрахунок автомобілів порушників за кольором та типом арешту.

```
BLACK          1716585
SILVER         1524630
WHITE          1312425
GRAY            958140
RED             646545
BLUE            609575
GREEN           287340
GOLD            251050
BLUE, DARK      175985
TAN             161805
MAROON          140450
BLUE, LIGHT     105370
BEIGE            90610
N/A              87900
GREEN, DK        83960
GREEN, LGT       44785
BROWN            37170
YELLOW           30660
ORANGE           29255
BRONZE           18800
PURPLE           15500
MULTICOLOR        6285
CREAM             4870
COPPER            2510
PINK              1155
CHROME             205
CAMOUFLAGE         170
Name: color, dtype: int64
```

```
A - Marked Patrol                       6916605
Q - Marked Laser                         757105
B - Unmarked Patrol                      269765
S - License Plate Recognition             77780
O - Foot Patrol                           72665
L - Motorcycle                            69280
E - Marked Stationary Radar               53265
G - Marked Moving Radar (Stationary)      49280
R - Unmarked Laser                        33490
M - Marked (Off-Duty)                     16030
I - Marked Moving Radar (Moving)          13610
H - Unmarked Moving Radar (Stationary)     7615
J - Unmarked Moving Radar (Moving)         4500
F - Unmarked Stationary Radar              4230
C - Marked VASCAR                          2555
P - Mounted Patrol                         1720
D - Unmarked VASCAR                        1525
N - Unmarked (Off-Duty)                    1165
K - Aircraft Assist                         280
Name: arrest_type, dtype: int64
```

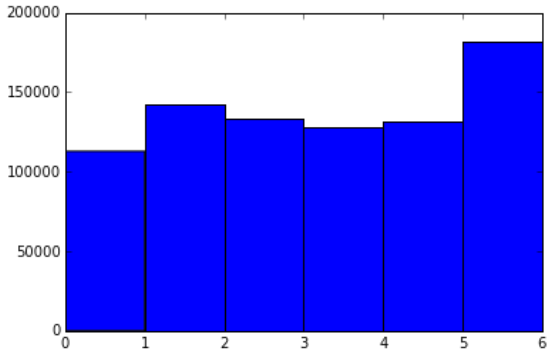
Якщо подібна статистика не виводиться через нестачу оперативної пам'яті, зменшуйте кількість заголовків стовпців у good_columns, залишивши тільки найбільш потрібні. *Додайте аналогічний код та скриншоти до звіту.*

Тепер спробуємо обробити числові дані, зокрема проаналізувати інформацію стосовно геолокації (стовпці longitude, latitude та date). З цією метою потрібно реалізувати зведення широти та довготи до типу float, оскільки всі дані зчитувались як рядки. Тому напишемо прості конвертнери типів та побудуємо залежність кількості штрафів від дня тижня:

```
import numpy as np
def parse_float(x):
    try:
        x = float(x)
    except Exception:
        x = 0
    return x
stops["longitude"] = stops["longitude"].apply(parse_float)
stops["latitude"] = stops["latitude"].apply(parse_float)

import datetime
def parse_full_date(row):
```

```
    date = datetime.datetime.strptime(row["date_of_stop"], "%Y-%m-
%dT%H:%M:%S")
    time = row["time_of_stop"].split(":")
    date = date.replace(hour=int(time[0]), minute = int(time[1]), second =
int(time[2]))
    return date
stops["date"] = stops.apply(parse_full_date, axis=1)

import matplotlib.pyplot as plt
plt.hist(stops["date"].dt.weekday, bins=6)
```

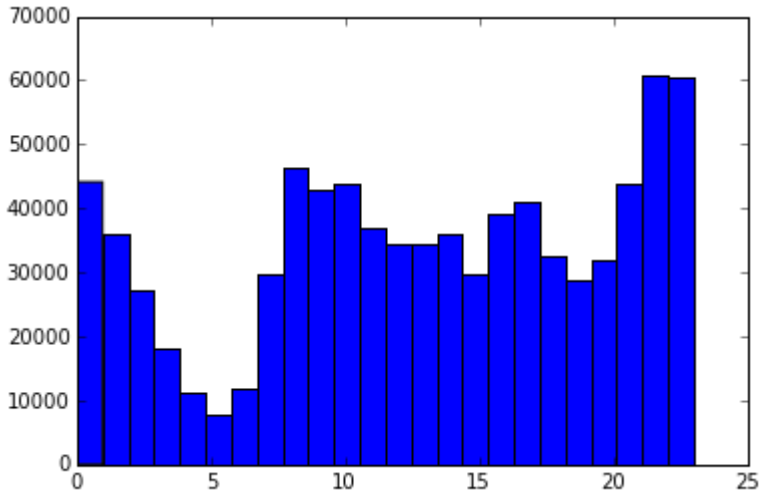
Отримаємо результат на зразок такого:



На даному графіку понеділок = 0, а неділя – 6. Для такого графіку видається очевидним, що найбільше зупинок транспортних засобів відбувається по неділях, а найменше – по понеділках.

Можемо «копнути» ще глибше та розглянути погодинно кількість арештів:

```
plt.hist(stops["date"].dt.hour, bins=24)
```



Візуалізація показує наростання порушень до опівночі та їх спадання з мінімумом близько 5-ї години ранку. *Виконайте аналогічну роботу та отримайте відповідні графіки на більш свіжих даних. Додайте програмні коди та скриншоти з візуалізаціями до звіту.*

2. ***1 бал*** ***(Excel)*** Бібліотека pandas має чудові інструменти для зчитування даних з Excel-файлів. Також доступний експорт результатів з pandas назад в Excel. Для виконання даного завдання потрібно використовувати наступні бібліотеки Python:
    a. matplotlib – для візуалізації даних
    b. NumPy – функціональність для обробки числових даних
    c. OpenPyXL – зчитування/запис файлів xlsx/xlsm
    d. pandas – імпорт даних, їх очистка, дослідження та аналіз
    e. xlrd – зчитування Excel-даних
    f. xlwt – запис в Excel
    g. XlsxWriter – запис у файли Excel (xlsx)

    В якості основного датасету будемо брати IMDB Scores. Він має 3 листа: 1900s, 2000s, 2010s, кожний з яких містить дані щодо фільмів, випущених в той період часу. Використаємо дані для визначення рейтингового розподілу фільмів, візуалізації фільмів з найвищим рейтингом та чистою виручкою, обчислимо статистичну інформацію про фільми.

    Для зчитування датасету буде використано метод read_excel():

```
import pandas as pd

excel_file = 'movies.xls'
movies = pd.read_excel(excel_file)
```

    Отриманий датафрейм зберігаємо в змінну movies. *Виведіть перші рядки датасету, викликавши метод DataFrame.head() та відобразіть код і скриншот у звіті*.

    Для врахування листа Excel, який потрібно обробляти, метод read_excel() передбачає спеціальні параметри:

```
movies_sheet1 = pd.read_excel(excel_file, sheet_name=0, index_col=0)
movies_sheet2 = pd.read_excel(excel_file, sheet_name=1, index_col=0)
movies_sheet3 = pd.read_excel(excel_file, sheet_name=2, index_col=0)
```

    Параметр sheet_name визначає листок, який потрібно зчитати, за його номером (з нуля) або назвою. Індексація рядків та стовпців листа розпочинається з нуля, а конкретний стовпчик задається за допомогою параметра index_col.

    Оскільки всі три листи мають подібні дані, проте різні записи, створимо єдиний датафрейм:

```
movies = pd.concat([movies_sheet1, movies_sheet2, movies_sheet3])
```

    Для зчитування кількох листів застосуємо клас ExcelFile:

```
xlsx = pd.ExcelFile(excel_file)
movies_sheets = []
for sheet in xlsx.sheet_names:
   movies_sheets.append(xlsx.parse(sheet))
movies = pd.concat(movies_sheets)
```

Тут використовується метод concat() для конкатенації об’єктів на зразок DataFrame чи Series. *Дослідіть сформовані дані: виведіть розмір та останні кілька рядків (метод tail()). Додайте до звіту відповідний код та скриншоти.*

Спробуємо відсортувати значення за одним або кількома стовпчиками, як це доступно в Excel. Для цього призначений спеціальний метод sort_values():

```
sorted_by_gross = movies.sort_values(['Gross Earnings'], ascending=False)
```

Таким чином, можна буде вивести, наприклад, топ 10 фільмів за валовою виручкою (Gross Earnings).

```
sorted_by_gross["Gross Earnings"].head(10)
```

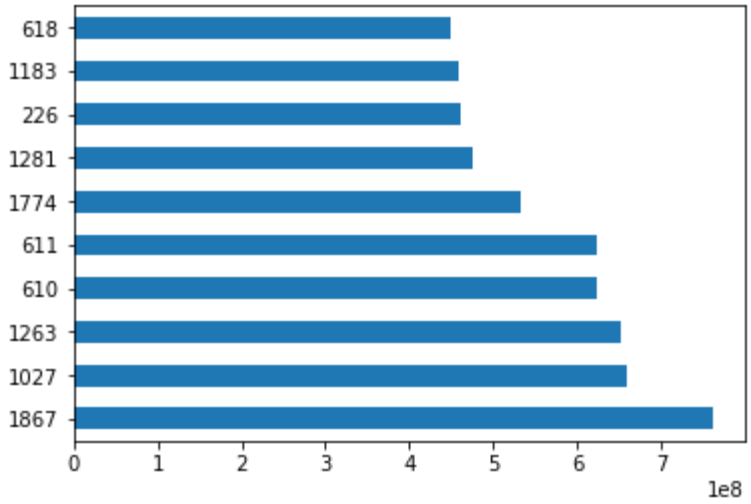
Вивід має бути в формі

```
1867 760505847.0

1027 658672302.0

1263 652177271.0

610 623279547.0

611 623279547.0

1774 533316061.0

1281 474544677.0

226 460935665.0

1183 458991599.0

618 448130642.0

Name: Gross Earnings, dtype: float64
```

Відповідні дані можна швидко візуалізувати за допомогою бібліотеки matplotlib:

```
import matplotlib.pyplot as plt
sorted_by_gross['Gross Earnings'].head(10).plot(kind="barh")

plt.show()
```

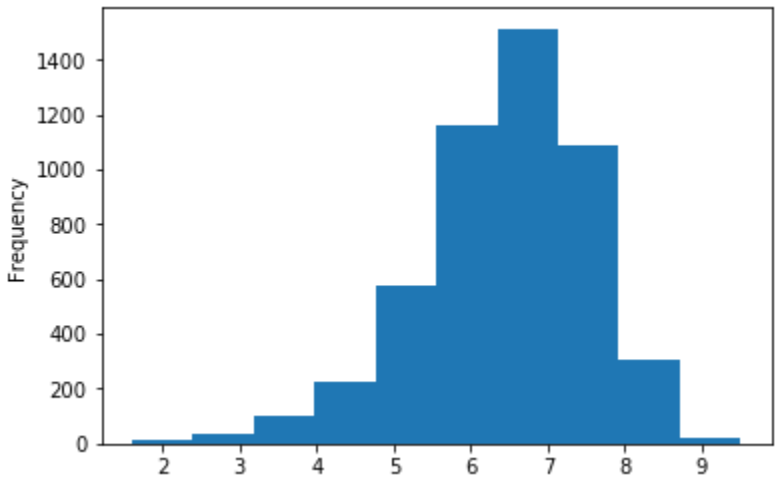
Вигляд графіку буде таким:



Також візуалізуйте розподіл рейтингових оцінок фільмів у вигляді гістограми. Для цього можна використати наступний код:

```
movies['IMDB Score'].plot(kind="hist")

plt.show()
```

Додайте відповідні коди та скриншоти до звіту. Орієнтуйтесь на такий вигляд:



Такий результат показує, що більшість фільмів потрапляє в категорію рейтингу між 6 та 8 балами з 10.

Пропуск деяких рядків у даних з Excel-файлу також часто є важливим. Наприклад, при такому форматуванні

	A	B	C	D	E	F	G	H
1								
2	**IMDB Ratings data**							
3	**Available from Kaggle.com**							
4								
5	Metropolis	1927	Drama|Sci-Fi	German	Germany	Not Rated	145	1.33
6	Pandora's Box	1929	Crime|Drama|Romance	German	Germany	Not Rated	110	1.33
7	The Broadway Melody	1929	Musical|Romance	English	USA	Passed	100	1.37
8	Hell's Angels	1930	Drama|War	English	USA	Passed	96	1.2
9	A Farewell to Arms	1932	Drama|Romance|War	English	USA	Unrated	79	1.37
10	42nd Street	1933	Comedy|Musical|Romance	English	USA	Unrated	89	1.37
11	She Done Him Wrong	1933	Comedy|Drama|History|Musical|Romance	English	USA	Approved	66	1.37
12	It Happened One Night	1934	Comedy|Romance	English	USA	Unrated	65	1.37
13	Top Hat	1935	Comedy|Musical|Romance	English	USA	Approved	81	1.37
14	Modern Times	1936	Comedy|Drama|Family	English	USA	G	87	1.37

Тоді можна застосувати спеціальний параметр skiprows

```
movies_skip_rows = pd.read_excel("movies-no-header-skip-rows.xls",
header=None, skiprows=4)
```

Також тут показано, що немає рядка-заголовку з підписами стовпців.

Якщо потрібно зчитувати не всі стовпці, можна обмежити їх кількість для розбору:

```
movies_subset_columns = pd.read_excel(excel_file, parse_cols=6)
```

Параметр parse_cols задає індекс останнього стовпця від початку, тобто всього в обробку тут будуть доступні 7 стовпців.

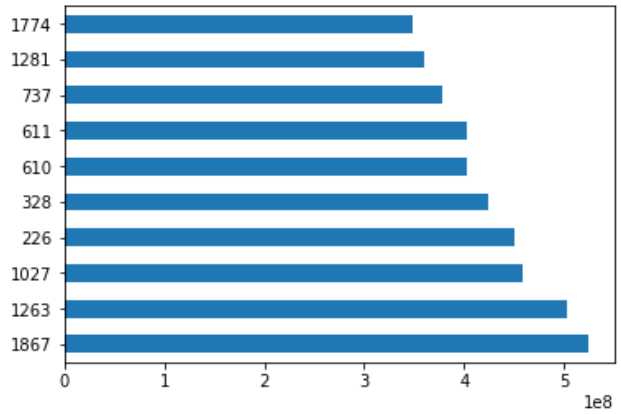
Маючи доступ до стовпців, можна виконувати математичні обчислення на основі їх даних. Зокрема, можемо визначити чисту виручку (Net Earnings) як загальну виручку (Gross Earnings) – бюджет фільму (Budget):

```
movies["Net Earnings"] = movies["Gross Earnings"] - movies["Budget"]
```

Відсортуйте отримані значення та покажіть 10 найуспішніших за виручкою фільмів:

```
sorted_movies = movies[['Net Earnings']].sort_values(['Net Earnings'],
ascending=[False])sorted_movies.head(10)['Net Earnings'].plot.barh()

plt.show()
```

Додайте до звіту відповідний код та гістограму, як показано нижче:



В Excel популярні зведені таблиці (pivot table), які підсумовують дані з інших таблиць, групуючи дані за індексом та застосовуючи операції

сортування, підсумовування або усереднення. Це ж можна зробити і в pandas:

```
movies_subset = movies[['Year', 'Gross Earnings']]
```

Виведіть перші рядки з цими даними.

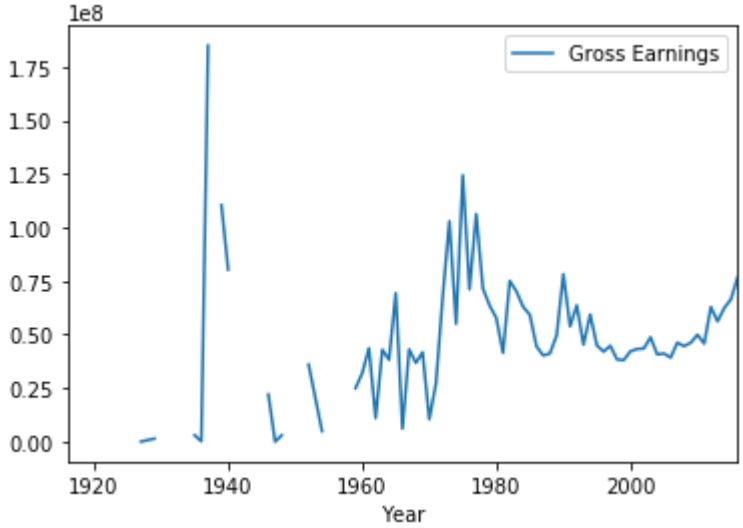
Можемо викликати спеціальний метод pivot_table() для підмножини даних:

```
earnings_by_year = movies_subset.pivot_table(index=['Year'])
```

Виведіть перші рядки з цими даними та візуалізуйте дані:

```
earnings_by_year.plot()

plt.show()
```

Матимемо приблизно такий вигляд:



Експорт результатів в Excel-файл часто здійснюється за допомогою методу to_excel():

```
movies.to_excel('output.xlsx', index=False)
```

Бібліотека Pandas використовує модуль xlwt для реалізації запису даних у файл. Параметр index можна пропускати.

Додаткова опція – об’єкт ExcelWriter

```
writer = pd.ExcelWriter('output.xlsx', engine='xlsxwriter')
movies.to_excel(writer, index=False, sheet_name='report')
workbook = writer.book
worksheet = writer.sheets['report']
```

Налаштування форматування даних можна здійснити за допомогою методу add_format(). Наприклад, виділимо рядок із заголовками жирним:

```
header_fmt = workbook.add_format({'bold': True})

worksheet.set_row(0, None, header_fmt)
```

Наприкінці викличте метод save() та запишіть результат:

```
writer.save()
```

Відобразіть код та структуру утвореного Excel-файлу у звіті. Структура має набути подібного вигляду:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Title	Year	Genres	Language	Country	ntent Rati	Duration	spect Rati	Budget	oss Earnin	Director	Actor 1	Actor 2	Actor 3
2	Intoleranc	1916	Drama|History|War		USA	Not Rated	123	1.33	385907		D.W. Griff	Lillian Gis	Mae Mars	Walter Lo
3	Over the H	1920	Crime|Drama		USA		110	1.33	100000	3000000	Harry F. M	Stephen C	Johnnie W	Mary Carr
4	The Big Pa	1925	Drama|Romance|W		USA	Not Rated	151	1.33	245000		King Vidor	John Gilbe	Renée Ad	Claire Ada
5	Metropoli	1927	Drama|Sc	German	Germany	Not Rated	145	1.33	6000000	26435	Fritz Lang	Brigitte He	Gustav Fr	Rudolf Kle
6	Pandora's	1929	Crime|Dra	German	Germany	Not Rated	110	1.33		9950	Georg Wil	Louise Bro	Francis Le	Fritz Kortn
7	The Broad	1929	Musical|R	English	USA	Passed	100	1.37	379000	2808000	Harry Bea	Anita Page	Bessie Lov	Charles Ki
8	Hell's Ang	1930	Drama|W	English	USA	Passed	96	1.2	3950000		Howard H	Jean Harlo	Marian Ma	James Hal
9	A Farewel	1932	Drama|Ro	English	USA	Unrated	79	1.37	800000		Frank Bor	Gary Coop	Helen Hay	Adolphe M
10	42nd Stree	1933	Comedy|M	English	USA	Unrated	89	1.37	439000	2300000	Lloyd Bac	Ginger Ro	Dick Powe	George Br
11	She Done	1933	Comedy|M	English	USA	Approved	66	1.37	200000		Lowell Sh	Mae West	Gilbert Ro	Louise Be