Miembros del equipo:

- Sebastian Mesa Montoya
- Juan Fernando Quintero Perez

Punto 1

In [1]:
```python
import re


def encontrar_codigos(texto:str) -> int:
    """De un string dado encontramos todos los codigos
    coincidentes.

    Parametros:
        - texto (str): Cadena inicial de donde se
                       buscaran los codigos.

    Retorna:
        - int: Con el total de codigos encontrados.
    """
    # Patron para identificar los codigos
    patron = '^REF-[98765]{2}[PQRWXYZ]{4}[_#&%]{1}$'
    # Podemos usar MULTILINE para hacer la coincidencia en cada linea
    codigos = re.findall(patron, texto, re.MULTILINE)

    print('Estos son los codigos encontrados:\n', codigos, sep='')
    return len(codigos)


ejemplo = """
REF-86PQXY%
REF-99ZZWW#
REF-34ABC1&
REF-75QRWP&
REF-123ABC#
REF-68XYQR#
REF-57PWRX%
REF-88WQXY_
REF-XYZ987%
"""
c = encontrar_codigos(ejemplo)
print('Total de codigos encontrados en la cadena:', c)
```

```
Estos son los codigos encontrados:
['REF-86PQXY%', 'REF-99ZZWW#', 'REF-75QRWP&', 'REF-68XYQR#', 'REF-57PWRX%', 'REF-88WQXY_']
Total de codigos encontrados en la cadena: 6
```

Punto 2

In [4]:
```python
def es_primo(numero):
    # Verificamos si un numero dado es primo
    if numero < 2:
        return False
    for i in range(2, int(numero**0.5) + 1):
        if numero % i == 0:
            return False
    return True


def generador_primos():
    numero = 2
    while True:
        if es_primo(numero):
            # Pausa y devuelve 1 primo
            yield numero
        # Seguimos iterando para buscar otro primo
        numero += 1


# Iterador externo para controlar cuantos primos se desean
def obtener_n_primos(n):
    contador = 0
    # Reanuda el generador cada vez que se solicita
    for primo in generador_primos():
        if contador == n:
            # Detenemos despues de obtener 'n' primos
            break
        print(primo)
        contador += 1

# Ejemplo con los primeros 15 primos
obtener_n_primos(15)
```

```
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
```

Punto 3

- Importar el dataset

- Imprimir los primeros 5 datos
- Encontrar la longitud del dataset
- Imprimir los encabezados
- Obtener un sub dataFrame con películas desde 1980 hasta la actualidad
- Dado el sub data Frame anterior: encontrar la película más corta, la menos rentable y el promedio de duración
- Encontrar la desviación estándar de las calificaciones de la película en el sub dataFrame
- Encontrar el promedio de votos agrupados género
- Encontrar los directores y el número de ocurrencias en el dataset, ordenados descendentemente
- Encontrar la películas mejor calificadas y que pertenezcan incluyan estos 3 géneros (Horror, Mystery, Sci-Fi)

In [5]:
```
import numpy as np
import pandas as pd
```

In [6]:
```
# Importar el dataset
df = pd.read_csv('./Horror Movies IMDb.csv')
```

In [7]:
```
# Imprimir los primeros 5 datos
df.head(5)
```

Out[7]:

	Movie Title	Movie Year	Runtime	Genre	Rating	Director	Votes	Gross
0	Alien	1979	117	Horror, Sci-Fi	8.5	Ridley Scott	9,05,275	$78.90M
1	Psycho	1960	109	Horror, Mystery, Thriller	8.5	Alfred Hitchcock	6,89,068	$32.00M
2	The Shining	1980	146	Drama, Horror	8.4	Stanley Kubrick	10,51,582	$44.02M
3	The Thing	1982	109	Horror, Mystery, Sci-Fi	8.2	John Carpenter	4,39,793	$13.78M
4	Tumbbad	2018	104	Drama, Fantasy, Horror	8.2	Rahi Anil Barve	53,297	NaN

In [8]:
```
# Encontrar la longitud del dataset

filas, columnas = df.shape
print(f'Filas: {filas} \nColumnas: {columnas}')
```

```
Filas: 836
Columnas: 8
```

In [9]:
```
# Imprimir los encabezados
df.columns
```

Out[9]:
```
Index(['Movie Title', 'Movie Year', 'Runtime', 'Genre', 'Rating', 'Director',
       'Votes', 'Gross'],
      dtype='object')
```

In [24]:
```
# sub dataFrame con películas desde 1980 hasta la actualidad
sub_df = df[df["Movie Year"]>=1980].copy()
sub_df
```

Out[24]:

	Movie Title	Movie Year	Runtime	Genre	Rating	Director	Votes	Gross
2	The Shining	1980	146	Drama, Horror	8.4	Stanley Kubrick	1051582	$44.02M
3	The Thing	1982	109	Horror, Mystery, Sci-Fi	8.2	John Carpenter	439793	$13.78M
4	Tumbbad	2018	104	Drama, Fantasy, Horror	8.2	Rahi Anil Barve	53297	NaN
10	The Blue Elephant	2014	170	Drama, Horror, Mystery	8.0	Marwan Hamed	29151	NaN
11	Shaun of the Dead	2004	99	Comedy, Horror	7.9	Edgar Wright	572237	$13.54M
...	...	...	...	...	...	...	...	...
830	BloodRayne	2005	95	Action, Fantasy, Horror	3.0	Uwe Boll	36527	$2.41M
831	Troll 2	1990	95	Comedy, Fantasy, Horror	2.9	Claudio Fragasso	33908	NaN
832	Laxmii	2020	141	Action, Comedy, Horror	2.5	Raghava Lawrence	58053	NaN
833	Alone in the Dark	2005	96	Action, Horror, Sci-Fi	2.4	Uwe Boll	46403	$5.18M
834	House of the Dead	2003	90	Action, Adventure, Horror	2.1	Uwe Boll	38041	$10.25M

776 rows × 8 columns

In [23]:
```
# Encontrar la película más corta y su fila completa
mas_corta_runtime = sub_df["Runtime"].min()
fila_mas_corta = sub_df.loc[sub_df["Runtime"].idxmin()]

# Limpiar la columna 'Gross' y convertirla a valores numéricos
sub_df['Gross'] = sub_df['Gross'].replace({'\$': '', 'M': ''},
                                          regex=True).astype(float)
sub_df['Gross'] = sub_df['Gross'] * 1e6

# Encontrar la película menos rentable y su fila completa
menos_rentable_gross = sub_df["Gross"].min()
fila_menos_rentable = sub_df.loc[sub_df["Gross"].idxmin()]

# Calcular el promedio de duración
prom_duracion = sub_df["Runtime"].mean()
```

```
# Imprimir resultados
print(f"1. Película más corta: \n{fila_mas_corta}\n\n",
      f"2. Película menos rentable: \n{fila_menos_rentable} \n\n",
      f"3. Promedio de duración: {prom_duracion:.2f} minutos",
      sep="")
```

```
1. Película más corta:
Movie Title            Host II
Movie Year                2020
Runtime                     57
Genre          Horror, Mystery
Rating                      6.5
Director            Rob Savage
Votes                   34,122
Gross                      NaN
Name: 297, dtype: object

2. Película menos rentable:
Movie Title              Ginger Snaps
Movie Year                       2000
Runtime                           108
Genre          Drama, Fantasy, Horror
Rating                             6.8
Director                 John Fawcett
Votes                          49,563
Gross                             0.0
Name: 202, dtype: object

3. Promedio de duración: 101.11 minutos
```

In [16]:
```
# Encontrar la desviación estándar de las calificaciones
# de la película en el sub dataFrame

std = sub_df['Rating'].std()
f"La desviacion estadar de las calificaciones es: {std}"
```

Out[16]: 'La desviacion estadar de las calificaciones es: 0.8838861854481449'

In [13]:
```
# Encontrar el promedio de votos agrupados género

# Eliminar las comas en la columna 'Votes' y convertir a int
df['Votes'] = df['Votes'].str.replace(',', '').astype(int)

# Separar géneros múltiples y expandir el DataFrame
df_generos = df.assign(Genre=df['Genre'].str.split(', ')).explode('Genre')

# Agrupar por género y calcular el promedio de votos
promedio_votos_por_genero = df_generos.groupby('Genre')['Votes'].mean()

# Mostrar el resultado
promedio_votos_por_genero
```

Out[13]:
```
Genre
Action       125089.275862
Adventure    110927.962025
Animation     91892.625000
Biography     37177.500000
Comedy        86034.584000
Crime         86731.545455
Drama         95490.457778
Family        62241.777778
Fantasy       93241.361538
History       49744.000000
Horror       100878.299486
Music         60236.000000
Musical      160347.750000
Mystery      104994.643678
Romance       70117.857143
Sci-Fi       110156.379845
Thriller      99868.928854
War           28700.000000
Western      105501.000000
Name: Votes, dtype: float64
```

In [14]:
```python
# Encontrar los directores y el número de ocurrencias en el dataset

# Contar las ocurrencias de cada director
directores_ocurrencias = df['Director'].value_counts()

# Mostrar el resultado ordenado
directores_ocurrencias.head(20)
```

Out[14]:
```
Director
John Carpenter         11
Wes Craven             11
David Cronenberg        8
James Wan               7
Guillermo del Toro      7
Paul W.S. Anderson      6
George A. Romero        6
Alexandre Aja           6
Sam Raimi               6
Christopher Landon      6
Mike Flanagan           6
M. Night Shyamalan      5
Rob Zombie              5
Eli Roth                5
Steve Miner             5
Darren Lynn Bousman     5
Renny Harlin            5
Tobe Hooper             4
Tim Burton              4
Joe Dante               4
Name: count, dtype: int64
```

In [15]:
```python
# Encontrar la películas mejor calificadas y que pertenezcan incluyan
# estos 3 géneros (Horror, Mystery, Sci-Fi)
peliculas_filtradas = df[df['Genre'].str.contains('Horror') &
```

```
                          df['Genre'].str.contains('Mystery') &
                          df['Genre'].str.contains('Sci-Fi')]

# Ordenar por la calificación 'Rating' en orden descendente
peliculas_mejor_calificadas = peliculas_filtradas.sort_values(
    by='Rating', ascending=False)

# Mostrar el resultado
peliculas_mejor_calificadas[['Movie Title', 'Rating', 'Genre']]
```

Out[15]:

	Movie Title	Rating	Genre
3	The Thing	8.2	Horror, Mystery, Sci-Fi
136	Timecrimes	7.1	Horror, Mystery, Sci-Fi
184	Nope	6.8	Horror, Mystery, Sci-Fi
204	eXistenZ	6.8	Horror, Mystery, Sci-Fi
253	Bird Box	6.6	Horror, Mystery, Sci-Fi
260	The Faculty	6.6	Horror, Mystery, Sci-Fi
307	Possessor	6.5	Horror, Mystery, Sci-Fi
323	Time Lapse	6.5	Horror, Mystery, Sci-Fi
433	Color Out of Space	6.2	Horror, Mystery, Sci-Fi
459	Open Grave	6.2	Horror, Mystery, Sci-Fi
419	The Thing I	6.2	Horror, Mystery, Sci-Fi
525	Vivarium	5.9	Horror, Mystery, Sci-Fi
534	The Void I	5.9	Horror, Mystery, Sci-Fi
536	The Fourth Kind	5.9	Horror, Mystery, Sci-Fi
580	The Relic	5.8	Horror, Mystery, Sci-Fi
668	Cube^2: Hypercube	5.5	Horror, Mystery, Sci-Fi
680	In the Tall Grass	5.4	Horror, Mystery, Sci-Fi
728	Apollo 18	5.2	Horror, Mystery, Sci-Fi
729	The Lazarus Effect	5.2	Horror, Mystery, Sci-Fi
739	Halloween III: Season of the Witch	5.1	Horror, Mystery, Sci-Fi