About Project: IMDB Movie Analysis

Objective:

As a data analyst intern at IMDB, you have been tasked with exploring and analyzing the IMDB Movies dataset. Your goal is to answer specific business questions, gain insights into movie trends, and deliver actionable recommrndations. Using Python and libraries such as Pandas, Numpy, Seaborn, and Matplotlib, perform analysis to help IMDB better understand genre popularity, rating trends, and factors influencing movie success.

Tools and Libraries Used

- **Python**
- **Pandas:** Data Manipulation and analysis
- **Numpy:** Numerical Computations
- **Matplotlib:** Data Visualization
- **Seaborn:** Advanced Visualization

About Company

IMDB(Internet Movie Database) is a comprehensive online database of information about films, television shows, video games, and online streaming content. It includes details such as cast and crew, plot summaries, user reviews, trivia and ratings. Establishedi in 1990, IMDB has become one of the most popular platforms for movie enthusiasts and industry professionals alike. It features user-generated content, professional critiques, and a proprietary rating system based on user votes. Owned by Amazon since 1998, IMDB also offers a subscription service, IMDBPro, providing industry-focuesed features like contact information and production updates.

Dataset Overview

The dataset includes the following columns:

- **names:** Movie Titles
- **date_x:** Release Dates
- **score:** IMDB Ratings
- **genre:** Genres
- **overview** Movie Summaries
- **crew:** Cast and Crew Information
- **orig_title:** Original Titles
- **status:** Release status(e.g. released,post-production)
- **orig_lang:** Original Language
- **budget_x:** Production Budgets
- **revenue:** Box Office Revenues
- **country:** Production Country

Loading the dataset and Perform initial setup

Task: Load the dataset and perform initial setup

```
# Importing necessary libraries for the project
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Loading the dataset
Data = "E://CODING//WSCUBE TECH//project
datasets//w14//imdb_movies.csv"
df = pd.read_csv(Data)

# Display the top 5 rows of datasets
df.head()

----------------------------------------------------------------------
-----
PermissionError                           Traceback (most recent call
last)
File C:\ProgramData\anaconda3\Lib\site-packages\pandas\io\
common.py:416, in _get_filepath_or_buffer(filepath_or_buffer,
encoding, compression, mode, storage_options)
    413 try:
    414     file_obj = fsspec.open(
    415         filepath_or_buffer, mode=fsspec_mode,
**(storage_options or {})
--> 416     ).open()
    417 # GH 34626 Reads from Public Buckets without Credentials needs
anon=True

File C:\ProgramData\anaconda3\Lib\site-packages\fsspec\core.py:134, in
OpenFile.open(self)
    128 """Materialise this as a real open file without context
    129 
    130 The OpenFile object should be explicitly closed to avoid
enclosed file
    131 instances persisting. You must, therefore, keep a reference to
the OpenFile
    132 during the life of the file-like it generates.
    133 """
--> 134 return self.__enter__()

File C:\ProgramData\anaconda3\Lib\site-packages\fsspec\core.py:102, in
OpenFile.__enter__(self)
    100 mode = self.mode.replace("t", "").replace("b", "") + "b"
--> 102 f = self.fs.open(self.path, mode=mode)
    104 self.fobjects = [f]

File C:\ProgramData\anaconda3\Lib\site-packages\fsspec\spec.py:1154,
in AbstractFileSystem.open(self, path, mode, block_size,
cache_options, compression, **kwargs)
   1153 ac = kwargs.pop("autocommit", not self._intrans)
-> 1154 f = self._open(
   1155     path,
   1156     mode=mode,
   1157     block_size=block_size,
   1158     autocommit=ac,
   1159     cache_options=cache_options,
   1160     **kwargs,
   1161 )
   1162 if compression is not None:

File C:\ProgramData\anaconda3\Lib\site-packages\fsspec\
implementations\local.py:183, in LocalFileSystem._open(self, path,
mode, block_size, **kwargs)
    182     self.makedirs(self._parent(path), exist_ok=True)
--> 183 return LocalFileOpener(path, mode, fs=self, **kwargs)

File C:\ProgramData\anaconda3\Lib\site-packages\fsspec\
implementations\local.py:287, in LocalFileOpener.__init__(self, path,
mode, autocommit, fs, compression, **kwargs)
    286 self.blocksize = io.DEFAULT_BUFFER_SIZE
--> 287 self._open()

File C:\ProgramData\anaconda3\Lib\site-packages\fsspec\
implementations\local.py:292, in LocalFileOpener._open(self)
    291 if self.autocommit or "w" not in self.mode:
--> 292     self.f = open(self.path, mode=self.mode)
    293     if self.compression:

PermissionError: [Errno 13] Permission denied: 'E:/CODING/WSCUBE
TECH/project datasets/w14/imdb_movies.csv'

During handling of the above exception, another exception occurred:

PermissionError                           Traceback (most recent call
last)
Cell In[6], line 3
      1 # Loading the dataset
      2 Data = "E://CODING//WSCUBE TECH//project
datasets//w14//imdb_movies.csv"
----> 3 df = pd.read_csv(Data)
      5 # Display the top 5 rows of datasets
      6 df.head()

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\io\parsers\
readers.py:912, in read_csv(filepath_or_buffer, sep, delimiter,
header, names, index_col, usecols, dtype, engine, converters,
true_values, false_values, skipinitialspace, skiprows, skipfooter,
nrows, na_values, keep_default_na, na_filter, verbose,
skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col,
date_parser, date_format, dayfirst, cache_dates, iterator, chunksize,
compression, thousands, decimal, lineterminator, quotechar, quoting,
doublequote, escapechar, comment, encoding, encoding_errors, dialect,
on_bad_lines, delim_whitespace, low_memory, memory_map,
float_precision, storage_options, dtype_backend)
    899 kwds_defaults = _refine_defaults_read(
    900     dialect,
    901     delimiter,
   (...)
    908     dtype_backend=dtype_backend,
    909 )
    910 kwds.update(kwds_defaults)
--> 912 return _read(filepath_or_buffer, kwds)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\io\parsers\
readers.py:577, in _read(filepath_or_buffer, kwds)
    574 _validate_names(kwds.get("names", None))
    576 # Create the parser.
--> 577 parser = TextFileReader(filepath_or_buffer, **kwds)
    579 if chunksize or iterator:
    580     return parser

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\io\parsers\
readers.py:1407, in TextFileReader.__init__(self, f, engine, **kwds)
   1404     self.options["has_index_names"] = kwds["has_index_names"]
   1406 self.handles: IOHandles | None = None
-> 1407 self._engine = self._make_engine(f, self.engine)

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\io\parsers\
readers.py:1661, in TextFileReader._make_engine(self, f, engine)
   1659     if "b" not in mode:
   1660         mode += "b"
-> 1661 self.handles = get_handle(
   1662     f,
   1663     mode,
   1664     encoding=self.options.get("encoding", None),
   1665     compression=self.options.get("compression", None),
   1666     memory_map=self.options.get("memory_map", False),
   1667     is_text=is_text,
   1668     errors=self.options.get("encoding_errors", "strict"),
   1669     storage_options=self.options.get("storage_options", None),
   1670 )
   1671 assert self.handles is not None
   1672 f = self.handles.handle

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\io\
common.py:716, in get_handle(path_or_buf, mode, encoding, compression,
memory_map, is_text, errors, storage_options)
```

```
    713     codecs.lookup_error(errors)
    715 # open URLs
--> 716 ioargs = _get_filepath_or_buffer(
    717     path_or_buf,
    718     encoding=encoding,
    719     compression=compression,
    720     mode=mode,
    721     storage_options=storage_options,
    722 )
    724 handle = ioargs.filepath_or_buffer
    725 handles: list[BaseBuffer]

File C:\ProgramData\anaconda3\Lib\site-packages\pandas\io\
common.py:427, in _get_filepath_or_buffer(filepath_or_buffer,
encoding, compression, mode, storage_options)
    423             storage_options = dict(storage_options)
    424             storage_options["anon"] = True
    425         file_obj = fsspec.open(
    426             filepath_or_buffer, mode=fsspec_mode,
**(storage_options or {})
--> 427         ).open()
    429     return IOArgs(
    430         filepath_or_buffer=file_obj,
    431         encoding=encoding,
   (...)
    434         mode=fsspec_mode,
    435     )
    436 elif storage_options:

File C:\ProgramData\anaconda3\Lib\site-packages\fsspec\core.py:134, in
OpenFile.open(self)
    127 def open(self):
    128     """Materialise this as a real open file without context
    129 
    130     The OpenFile object should be explicitly closed to avoid
enclosed file
    131     instances persisting. You must, therefore, keep a
reference to the OpenFile
    132     during the life of the file-like it generates.
    133     """
--> 134     return self.__enter__()

File C:\ProgramData\anaconda3\Lib\site-packages\fsspec\core.py:102, in
OpenFile.__enter__(self)
     99 def __enter__(self):
    100     mode = self.mode.replace("t", "").replace("b", "") + "b"
--> 102     f = self.fs.open(self.path, mode=mode)
    104     self.fobjects = [f]
    106     if self.compression is not None:
```

```
File C:\ProgramData\anaconda3\Lib\site-packages\fsspec\spec.py:1154,
in AbstractFileSystem.open(self, path, mode, block_size,
cache_options, compression, **kwargs)
   1152 else:
   1153     ac = kwargs.pop("autocommit", not self._intrans)
-> 1154     f = self._open(
   1155         path,
   1156         mode=mode,
   1157         block_size=block_size,
   1158         autocommit=ac,
   1159         cache_options=cache_options,
   1160         **kwargs,
   1161     )
   1162     if compression is not None:
   1163         from fsspec.compression import compr

File C:\ProgramData\anaconda3\Lib\site-packages\fsspec\
implementations\local.py:183, in LocalFileSystem._open(self, path,
mode, block_size, **kwargs)
    181 if self.auto_mkdir and "w" in mode:
    182     self.makedirs(self._parent(path), exist_ok=True)
--> 183 return LocalFileOpener(path, mode, fs=self, **kwargs)

File C:\ProgramData\anaconda3\Lib\site-packages\fsspec\
implementations\local.py:287, in LocalFileOpener.__init__(self, path,
mode, autocommit, fs, compression, **kwargs)
    285 self.compression = get_compression(path, compression)
    286 self.blocksize = io.DEFAULT_BUFFER_SIZE
--> 287 self._open()

File C:\ProgramData\anaconda3\Lib\site-packages\fsspec\
implementations\local.py:292, in LocalFileOpener._open(self)
    290 if self.f is None or self.f.closed:
    291     if self.autocommit or "w" not in self.mode:
--> 292         self.f = open(self.path, mode=self.mode)
    293         if self.compression:
    294             compress = compr[self.compression]

PermissionError: [Errno 13] Permission denied: 'E:/CODING/WSCUBE
TECH/project datasets/w14/imdb_movies.csv'
```