IMDb Movies Dataset with MongoDB & MatplotLib

Project Objective: To demonstrate ingestion, storage, cleaning, aggregation, and visualization of large-scaled movie metadata using MongoDB, a distributed NoSQL database. This project uses the publicly available IMDb dataset and is designed to showcase practical Big Data handling in a professional setting.

Why MongoDB?

- MongoDB is a schema-less, document-oriented NoSQL database.
- Ideal for semi-structured or hierarchical datasets like JSON or nested movie data.
- Supports horizontal scaling and fast aggregation pipelines.
- Unlike relational databases, MongoDb allows flexibility in data modeling and data iterative development.

Dataset Used:

- Source: [IMDb Datasets] (https://www.imdb.com/interfaces/)
- Files:
    - 'title.basics.tsv': contains metadata like movie title, year, runtime, and genres.
    - 'title.ratings.tsv': contains average ratings and vote counts.
- Format: TSV (Tab-Separated Values)
- Size: 300, 000+ rows, 7 relevant columns after cleaning

Tools & Technologies:

- Language: Python 3
- Database: MongoDB Community Edition (local)
- Libraries:
    - 'pandas': data preprocessing
    - 'pymongo': MongoDB interaction
    - 'matplotlib': visualization

#1 - Environment Setup:

bash

```
brew tap mongodb/brew
brew install mongodb-community@7.0
brew services start mongodb-community@7.0
```

#2 - Download & Extract Dataset:
bash
```
curl -O https://datasets.imdbws.com/title.basics.tsv.gz
curl -O https://datasets.imdbws.com/title.ratings.tsv.gz
gunzip title.basics.tsv.gz
gunzip title.ratings.tsv.gz
```

#3 - Python Script to Load and Process Data:
bash
~~~
Saved as `imdb_mongodb_project.py`
```python
import pandas as pd
from pymongo import MongoClient
import matplotlib.pyplot as plt
~~~

#4 - Load IMDb datasets:
```
print("Loading data...")
movies = pd.read_csv("title.basics.tsv", sep='\t', na_values='\\N')
ratings = pd.read_csv("title.ratings.tsv", sep='\t', na_values='\\N')
```

#5 - Filter for movie records only:
```
print("Filtering movie records...")
movies = movies[movies["titleType"] == "movie"]
```

#6 - Merge ratings:
```
print("Merging data...")
df = pd.merge(movies, ratings, on="tconst")
df = df[["tconst", "primaryTitle", "startYear", "runtimeMinutes",
"genres", "averageRating", "numVotes"]]
```

#7 - Clean the data:
```
print("Cleaning data...")
df.dropna(inplace=True)
df.drop_duplicates(subset="tconst", inplace=True)
df["startYear"] = df["startYear"].astype(int)
df["runtimeMinutes"] = df["runtimeMinutes"].astype(int)
df["genres"] = df["genres"].apply(lambda x: x.split(","))
```

#8 - Connect to MongoDB:
```
print("Connecting to MongoDB...")
client = MongoClient("mongodb://localhost:27017/")
db = client["imdb"]
collection = db["movies"]
collection.delete_many({})
print(f"🚀 Inserting {len(df)} records into MongoDB...")
collection.insert_many(df.to_dict("records"))
```

#9 - Display number of rows and columns:
```
print("\nMongoDB Document Count:", collection.count_documents({}))
print("Sample Document Keys:", list(collection.find_one().keys()))
```

#10 - Aggregation query - Top 10 genres:
```
print("\nAggregating top genres...")
pipeline = [
    {"$unwind": "$genres"},
    {"$group": {"_id": "$genres", "count": {"$sum": 1}}},
    {"$sort": {"count": -1}},
    {"$limit": 10}
]
top_genres = list(collection.aggregate(pipeline))
```

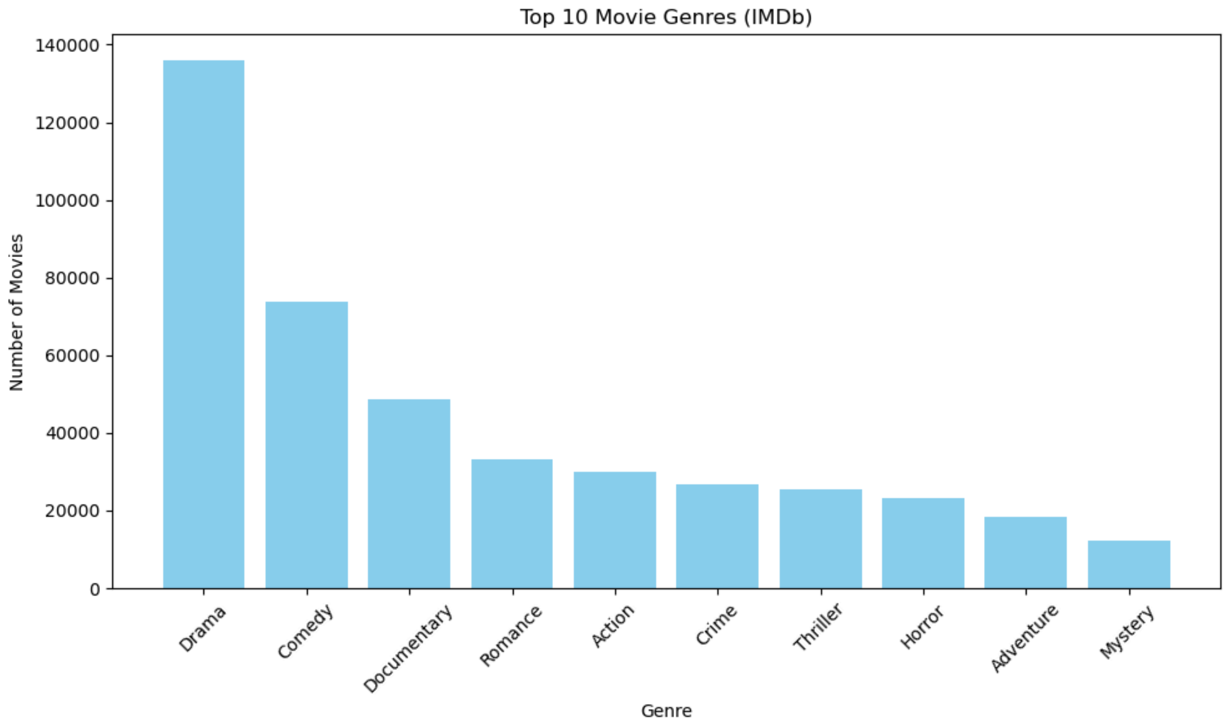
#11 - Visualize:
```
print("Generating visualization...")
labels = [g['_id'] for g in top_genres]
counts = [g['count'] for g in top_genres]

plt.figure(figsize=(10, 6))
plt.bar(labels, counts, color='skyblue')
plt.title("Top 10 Movie Genres (IMDb)")
plt.xlabel("Genre")
plt.ylabel("Number of Movies")
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig("top_genres.png")
plt.show()

print("\nCompleted. Chart saved as 'top_genres.png'.")
```

Output:

- Document Count in MongoDB: >130, 000 (filtered to movies only)
- Sample Document Keys:
  ```
  {'_id', 'tconst', 'primaryTitle', 'startYear',
  'runtimeMinutes', 'genres', 'averageRating', 'numVotes'}
  ```



Presentation Components:

- Script walkthrough: Loading, cleaning, inserting, querying
- MongoDB usage: Count documents, view sample
- Aggregation logic: ‘$unwind’, ‘$group’, ‘$sort’
- Chart explanation: Movie genres ranked

Why MongoDB?

- Schema flexibility
- Native JSON support
- Real-time querying with aggregation pipeline

imdb_mongodb_project.py

```
# IMDb MongoDB Big Data Project

import pandas as pd
from pymongo import MongoClient
import matplotlib.pyplot as plt

# 1. Load IMDb datasets
print("📥 Loading data...")
movies = pd.read_csv("title.basics.tsv", sep='\t', na_values='\\N')
ratings = pd.read_csv("title.ratings.tsv", sep='\t', na_values='\\N')

# 2. Filter only 'movie' entries
print("🎬 Filtering movie records...")
movies = movies[movies["titleType"] == "movie"]

# 3. Merge ratings with movie metadata
print("🔗 Merging data...")
df = pd.merge(movies, ratings, on="tconst")

# 4. Keep only relevant columns
df = df[[
    "tconst", "primaryTitle", "startYear",
    "runtimeMinutes", "genres", "averageRating", "numVotes"
]]

# 5. Clean the data
print("🧹 Cleaning data...")
df.dropna(inplace=True)
df.drop_duplicates(subset="tconst", inplace=True)
df["startYear"] = df["startYear"].astype(int)
df["runtimeMinutes"] = df["runtimeMinutes"].astype(int)
df["genres"] = df["genres"].apply(lambda x: x.split(","))

# 6. Connect to MongoDB
print("🧠 Connecting to MongoDB...")
client = MongoClient("mongodb://localhost:27017/")
db = client["imdb"]
collection = db["movies"]

# 7. Insert into MongoDB
print(f"🚀 Inserting {len(df)} records into MongoDB...")
collection.delete_many({})  # Optional: clear existing data
collection.insert_many(df.to_dict("records"))

# 8. Verify data
print("\n📊 Record count in MongoDB:", collection.count_documents({}))
print("📄 Sample document keys:", list(collection.find_one().keys()))

# 9. Aggregate genre counts
print("\n📈 Aggregating top 10 genres...")
pipeline = [
   {"$unwind": "$genres"},
   {"$group": {"_id": "$genres", "count": {"$sum": 1}}},
   {"$sort": {"count": -1}},
   {"$limit": 10}
]
top_genres = list(collection.aggregate(pipeline))

# 10. Plot genre distribution
print("📉 Generating bar chart...")
labels = [g["_id"] for g in top_genres]
counts = [g["count"] for g in top_genres]

plt.figure(figsize=(10, 6))
plt.bar(labels, counts, color='skyblue')
plt.title("Top 10 Movie Genres (IMDb)")
plt.xlabel("Genre")
plt.ylabel("Number of Movies")
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig("top_genres.png")
plt.show()

print("\n✅ Done! Chart saved as 'top_genres.png'.")
```