

TRABALHO PRÁTICO FUNDAMENTOS DE BANCO DE DADOS 2023/1

Ana Clara, Felipe Colpo e Guilherme Garcia

Definição da fonte

≡ kaggle

+ Create

🏠 Home

🏆 Competitions

📁 Datasets

🤖 Models

<> Code

💬 Discussions

🎓 Learn

✓ More

📅 View Active Events

🔍 Search



ROUNAK BANIK · UPDATED 6 YEARS AGO

Sign In

Register

The Movies Dataset

Metadata on over 45,000 movies. 26 million ratings from over 270,000 users.



Data Card Code (464) Discussion (34)

About Dataset

Context

These files contain metadata for all 45,000 movies listed in the Full MovieLens Dataset. The dataset consists of movies released on or before July 2017. Data points include cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages.

This dataset also has files containing 26 million ratings from 270,000 users for all 45,000 movies. Ratings are on a scale of 1-5 and have been obtained from the official GroupLens website.

Content

This dataset consists of the following files:

movies_metadata.csv: The main Movies Metadata file. Contains information on 45,000 movies featured in the Full MovieLens dataset. Features include posters, backdrops, budget, revenue, release dates, languages, production countries and companies.

keywords.csv: Contains the movie plot keywords for our MovieLens movies. Available in the form of a stringified JSON Object.

credits.csv: Consists of Cast and Crew Information for all our movies. Available in the form of a stringified JSON Object.

Usability ⓘ

8.24

License

CC0: Public Domain

Expected update frequency

Not specified


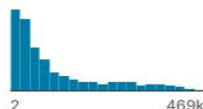
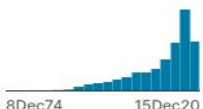


Tags

Earth and Nature

Movies and TV Shows

Popular Culture

Detail Compact Column

belongs_to_collection	# budget	genres	id	release_date	# revenue	# runtime	title
<div> <div>[null] 90%</div> <div>{'id': 415931, 'name': ...} 0%</div> <div>Other (4465) 10%</div> </div>		<div> <div>[{'id': 18, 'name': 'Dr...'}] 11%</div> <div>[{'id': 35, 'name': 'C...'}] 8%</div> <div>Other (36845) 81%</div> </div>					<div>42278</div> <div>unique values</div>
{'id': 10194, 'name': 'Toy Story Collection', 'poster_path': '/7G9915LfUQ2lVfwMEEhDsn3kT4B.jpg', 'ba...	30000000	[{'id': 16, 'name': 'Animation'}, {'id': 35, 'name': 'Comedy'}, {'id': 10751, 'name': 'Family'}]	862	1995-10-30	373554033	81.0	Toy Story
	65000000	[{'id': 12, 'name': 'Adventure'}, {'id': 14, 'name': 'Fantasy'}, {'id': 10751, 'name': 'Family'}]	8844	1995-12-15	262797249	104.0	Jumanji
{'id': 119050, 'name': 'Grumpy Old Men Collection', 'poster_path': '/nLvUdggPgm3F85NMCi19gVFUcet.jpg...	0	[{'id': 10749, 'name': 'Romance'}, {'id': 35, 'name': 'Comedy'}]	15602	1995-12-22	0	101.0	Grumpier Old Men
	16000000	[{'id': 35, 'name': 'Comedy'}, {'id': 18, 'name': 'Drama'}, {'id': 10749, 'name': 'Romance'}]	31357	1995-12-22	81452156	127.0	Waiting to Exhale
{'id': 96871, 'name': 'Father of the Bride Collection', 'poster_path': '/nts4i0mNnq7GNicycMJ9pSan204...	0	[{'id': 35, 'name': 'Comedy'}]	11862	1995-02-10	76578911	106.0	Father of the Bride Part II

Etapa 1

Categoria definida pelo grupo

- Filmes de comédia romântica.

```
df_filtered = df[df['genres'].apply(lambda x: "Comedy" in x and "Romance" in x)]  
df_sort = df_filtered.sort_values(by="revenue", ascending=False).head(1000)
```

Script em python que utiliza o pandas para filtrar quais filmes possuem comédia e romance dentro da sua classe de gêneros ('genres', na imagem).

Etapa 2

- **Normalização**

movies(belongs_to_collection(idCollecton,name), budget,
idMovie, release_date, revenue, runtime, title,
cast(idActor,name,genre), crew(idDirector,name,genre))

Etapa 2

- **Normalização**

filme(colecao(idColecao, nome), bilheteria, idFilme, titulo, lancamento, custo, duracao, ator(idAtor, nome, genero), diretor(idDiretor, nome, genero))

Etapa 2

- **Normalização**

Ator(idAtor, nome, genero)

Diretor(idDiretor, nome, genero)

Colecao(idColeção, nome)

Filme(idFilme, idColecao, titulo, bilheteria, lancamento, nota, custo, duracao)

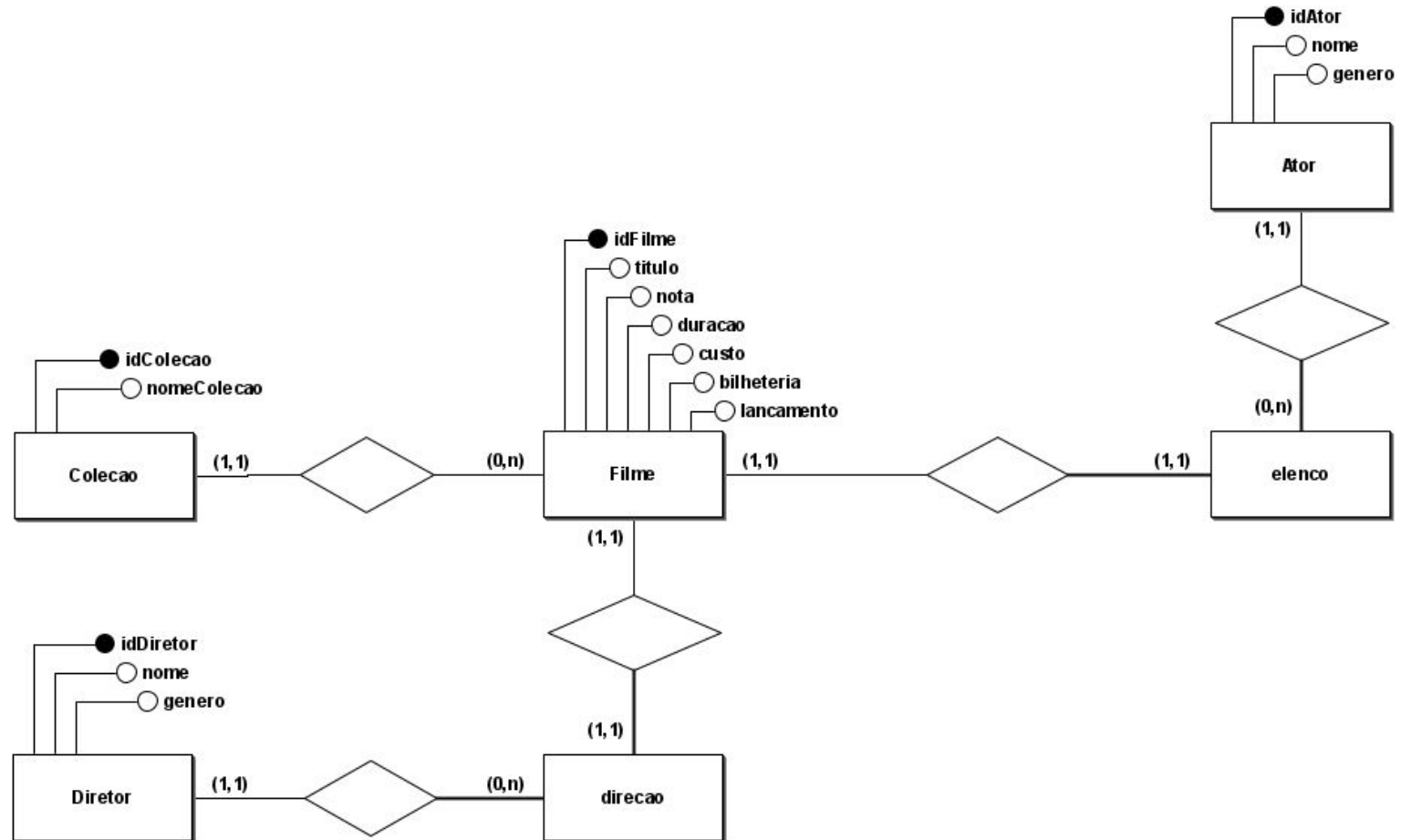
idColecao referencia colecao

Elenco(idFilme, idAtor)

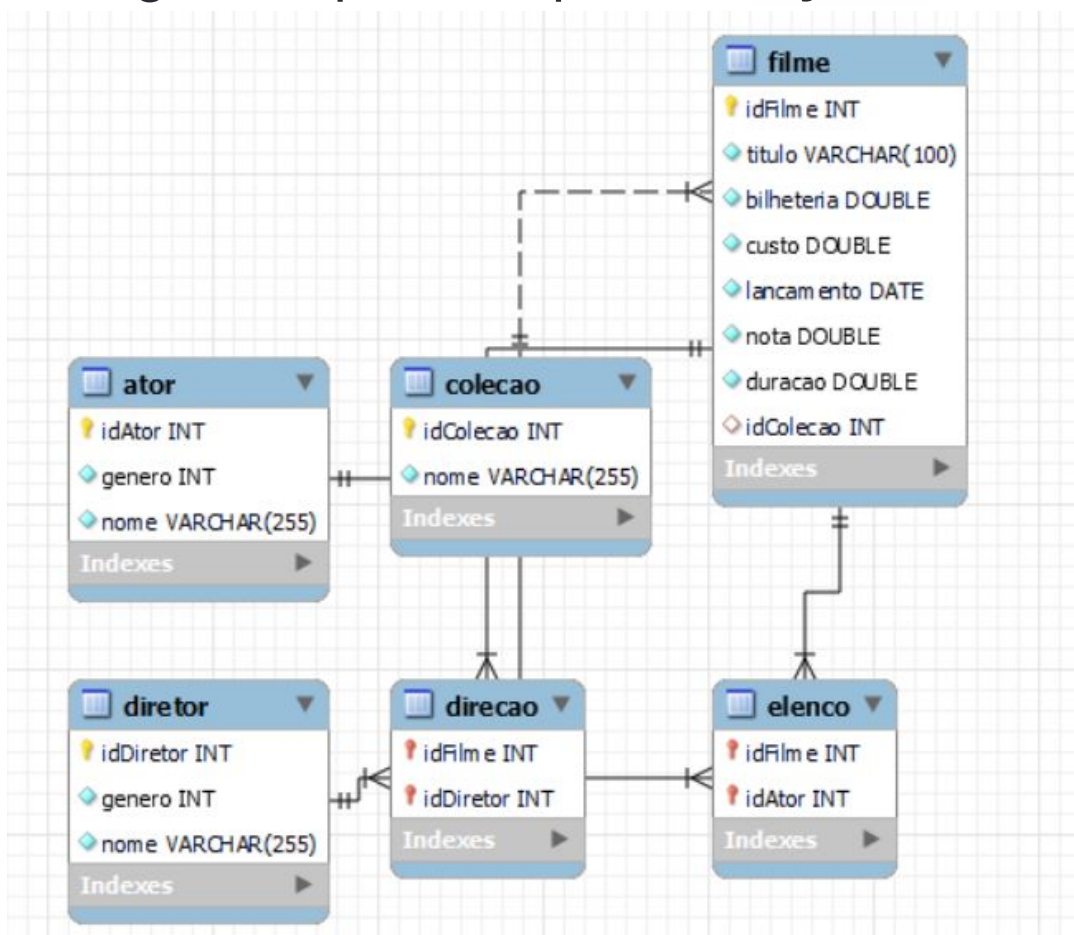
idFilme referencia Filme

idAtor referencia Ator

Diagrama entidade-relacionamento



- Modelo gráfico gerado pelo script de criação no Workbench



Normalização dos dados

Transforma os dados brutos do the movies dataset em tabelas .csv apenas com os dados necessários

```
df_filtered = df[df['genres'].apply(lambda x: "Comedy" in x and "Romance" in x)]
df_sort = df_filtered.sort_values(by="revenue",ascending=False).head(1000)

drop_df = ["homepage", "poster_path", "video", "imdb_id", "overview", "original_title",
           "spoken_languages", "tagline","original_language","status","genres","spoken_languages","adult",
           ,"production_companies","production_countries","vote_count","popularity"]
df = df_sort.drop(drop_df, axis=1)

df["id"] = pd.to_numeric(df['id'], errors='coerce', downcast="integer")
df["budget"] =pd.to_numeric(df['budget'], errors='coerce', downcast="float")
df['release_date'] = pd.to_datetime(df['release_date'])
df = df[df["budget"] > 0]
df["belongs_to_collection"] = df["belongs_to_collection"].apply(lambda x: ast.literal_eval(x) if pd.notna(x) else x)
df["belongs_to_collection"] = df["belongs_to_collection"].apply(lambda x: int(x["id"]) if pd.notna(x) else x)

df=df.rename(columns={"belongs_to_collection":"idColecao","budget":"custo","id":"idFilme","release_date":"lançamento",
                    "revenue":"bilheteria","runtime":"duracao","title":"titulo","vote_average":"nota"})
df.to_csv("../dados_normalizados/movies_base.csv",index=False)
```

Normalização dos dados

Transforma os dados brutos do the movies dataset em tabelas .csv apenas com os dados necessários

```
credits = pd.read_csv("../dados_brutos/credits.csv")
credits["crew"] = credits["crew"].apply(lambda x: ast.literal_eval(x))
credits["cast"] = credits["cast"].apply(lambda x: ast.literal_eval(x))
test = credits[credits["id"].isin(df["idFilme"])]
diretores = test.loc[:,["crew","id"]]
diretores["crew"] = diretores["crew"].apply(lambda x: [i for i in x if i["job"] == "Director"])
```

Normalização dos dados

Transforma os dados brutos do the movies dataset em tabelas .csv apenas com os dados necessários

```
data = []
for i in diretores["crew"]:
    for j in i:
        data.append([j["id"],j["name"],j["gender"]])
dire = pd.DataFrame(data, columns=["id","name","gender"])
dire = dire.drop_duplicates(subset="id")

atores = test.loc[:,["cast","id"]]
data = []
for i in atores["cast"]:
    for j in i:
        if "(Voice)" in j["name"]:
            j["name"].replace("(Voice)","")

        data.append([j["id"],j["name"],j["gender"]])

at = pd.DataFrame(data,columns=["id","name","gender"])
at = at.drop_duplicates(subset="id")
dire.to_csv("../dados_normalizados/diretores.csv",index=False)
at.to_csv("../dados_normalizados/atores.csv",index=False)
```

Inserção dos dados no MYSQL

Cria a conexão com o banco de dados usando o conector do mysql para python

```
config = {  
    'user': os.getenv("user"),  
    'password': os.getenv("password"),  
    'host': os.getenv("host"),  
}  
  
try:  
    conn = mysql.connector.connect(**config)  
    print("Conexão estabelecida com sucesso!")  
  
except mysql.connector.Error as error:  
    # Erro ao estabelecer a conexão  
    print(f"Erro ao conectar ao banco de dados: {error}")  
  
cursor = conn.cursor()
```

Conexão estabelecida com sucesso!

Inserção dos dados no MYSQL

Executa o script SQL de criação do banco de dados

Criacao do banco de dados

```
with open('../script_criacao.sql', 'r') as file:  
    sql_script = file.read()  
  
for result in cursor.execute(sql_script, multi=True):  
    pass  
conn.commit()
```

Inserção dos dados no MYSQL

Exemplo da inserção de dados na tabela diretor, usando um dataframe do pandas

```
for _, row in df_diretores.iterrows():
    idDiretor = row["id"]
    nome = row["name"]
    genero = row["gender"]

    try:
        query = "INSERT INTO diretor (idDiretor, genero, nome) VALUES (%s, %s, %s)"
        values = (idDiretor, genero, nome)
        cursor.execute(query, values)
        conn.commit()
    except IntegrityError as e:
        print(f"Erro de ID duplicado: {e}")
        conn.rollback()
    except Error as e:
        print(f"Erro ao inserir dados: {e}")
        conn.rollback()
```

Etapa 5

Visualização dos dados

- A ferramenta escolhida para a geração do dashboard foi a Microsoft Power BI.

Gráficos relativos ao gênero dos atores

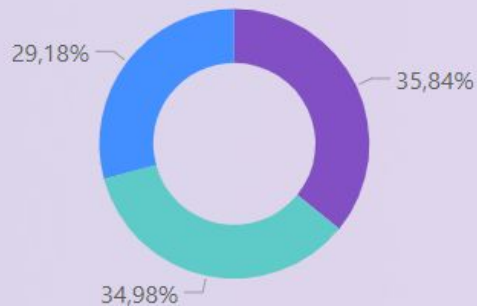
Top 10 filmes com mais homens no elenco



Top 10 filmes com mais mulheres no elenco



Gêneros de atores



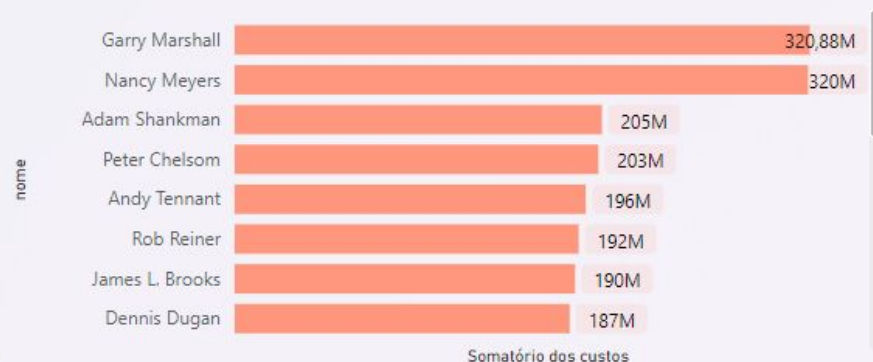
Gênero 2 0 1

Relações de filmes

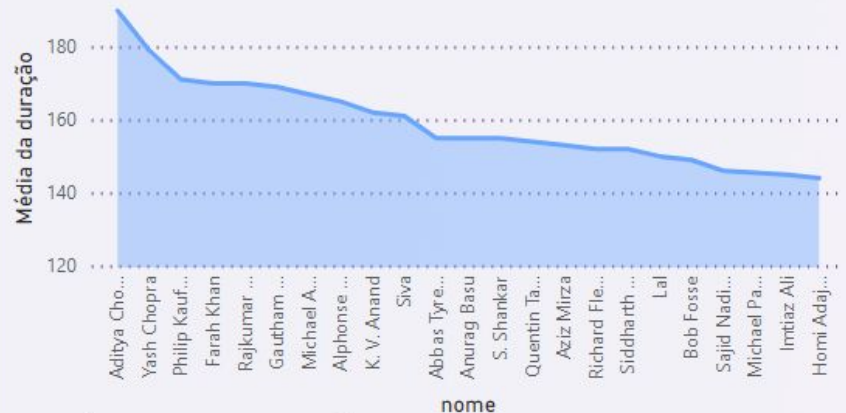
Soma de bilheteria por Ano desde 1990



Soma dos custos por diretores



Diretores cuja média de duração de filmes passou de 2 horas



Média das notas por coleções

