Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

Analiza popularnosti i žanrova u skupu podataka

"Top 15 000 rangiranih animea"

Rahela Štos

Kolegij Skladišta i rudarenja podataka

Sadržaj

Uvod

Danas u ovom digitalnom okruženju, prikupljanje, obrada i intrepetacija podataka postala je velika vještina kod poslovne organizacije, kao i za kreativnu stranu industrije. Anime industrija koja globalno bilježi svoj rast, uspjeh joj se temelji na razumijevanju gledatelja kroz analizu podataka o ocjenama, žanrovima, popularnosti i produkcijskim studijima.

Obrada velikih skupova podataka kao što su “Top 15000 rangiranih anime-a" predstavlja izazove zbog raznolikosti podataka, pa se ne primjenjuje ručna analiza nego se primjenjuju napredni alati poslovne inteligencije kako bi transformirali sirove podatke u djelotvorne uvide.

Ovaj projekt radila sam pomoću Python-a(pandas, sqlalchemy) te MySQL. Python sam koristila za ekstrakciju, transformaciju i učitavanje ETL podataka, a MySQL sam koristila za izgradnju relacijskog i dimenzijskog modela.

Cilj ovog projekta je bio da stvorim skladište podataka gdje se analiziraju trendovi popularnosti, uspoređuju studiji i žanrovi, te se gledaju koji čimbenici doprinose velikim ocjenama.

Osnovna analiza podataka

U ovom dijelu opisat ću osnovnu analizu podataka koju sam provela nad skupom podataka o anime serijama i filmovima. CSV datoteku sam preuzela sa kaggle stranice preko linka: https://www.kaggle.com/datasets/quanthan/top-15000-ranked-anime-dataset-update-to-32025?resource=download

Analizu sam provela u Pythonu koristeći pandas biblioteku.

1.1 Učitavanje podataka

Podatke sam učitala iz CSV datoteke gdje sam koristila pd.read_csv() funkciju. Ovaj način omogućuje jednostavno učitavanje velikih skupova podataka u Pandas DataFrame. Prije svega sam u CMD-u trebala instalirati potrebne biblioteke:

Pip install pandas sqlalchemy mysql-connector-python

Ova naredba instalira pandas potreban za analizu podataka, SQLAlchemy za rad s bazom te MySQL konektor.

Zatim sam koristila kod za učitavanje CSV datoteke:

```
import pandas as pd
PATH = r"C:\Users\rahel\Downloads\archive\top_anime_dataset.csv"
data_first_2000 = pd.read_csv(PATH, nrows=2000)
```

Nakon toga podaci su učitani i spremi za analizu.

1.2 Pregled prvih 5 redaka

Koristila sam metodu head() jer mi ona daje brz uvid u strukturu i sadržaj podataka i prikazuje prvih nekoliko redova DataFrame-a. Specificno pokazuje prvih 2000 redova.

```
print(data_first_2000.head())
```

Kada se taj dio koda pokrene, on prikazuje tipične vrijednosti za svaki stupac, što uključuje naziv anime-a, žanr, ocjenu, broj članova, studio i druge bitne podatke.

1.3 Veličina skupa podataka

```
print(data.shape)
```

Ovaj dio koda prikazuje broj redaka i stupaca u dataframe-u. Rezultat prikazuje dva broja, gdje prvi broj označava ukupan broj redaka, a drugi broj označava broj stupaca.

Rezultat (15000, 21) znači da imamo 15 000 anime naslova, a broj 21 znači da za svaki anime naslov ima 21 atribut.

1.4 Naziv stupaca

```
print(data.columns.values)
```

Ovaj dio koda sam koristila jer mi ispisuje sve stupce. Uključuje razne atributte anime-a, kao što su anime_id, naziv, score, members, genres, studios, episodes, type. Što mi pomaže kod preopznavanja svih dostupnih varijabli za analizu.

1.5 Broj nedostajućih vrijednosti po stupcu

Pri provjeri kvalitete podataka koristila sam kod:

```
print(data.isna().sum())
```

koji mi prikazuje broj nedostajućih vrijednosti za svaki stupac, što je važno za planiranje čišćenja i pripreme podataka za analizu.

1.6 Broj jedinstvenih vrijednosti po stupcu

```
print(data.nunique())
```

Ovaj kod koristila sam za ispis broja različitih vrijednosti u svakom stupcu.

Pomaže mi kod otkrivanja raznolikosti podataka u svakom atributu, gdje anime_id ima 15000 različitih vrijednosti, type ima vrijednosti TV, movie, OVA (original video animation), genres ima više različitih vrijednosti comedy, romance, drama, adventure, fantasy... kao i studios gdje imamo K-Factory, EOTA, Egg Firm itd.

1.7 Tipovi podataka po stupcima

```
print(data.dtypes)
```

Ovaj kod koristila sam za ispis tipova podataka za svaki stupac.

Većina stupaca su string ili brojčanog tipa, što nam pomaže kod korištenja različitih matematičkih i statističkih metoda bez da trebamo dodatno mijenjati tip podatka.

1.8 Frekvencije vrijednosti po stupcu

```
for column in data:
    print(f"\n--- {column} ---")
    print(data[column].value_counts())
```

Ovaj kod koristila sam za detaljnu analizu distribucije vrijednosti.

Što mi omogućuje uvid u raspodjelu vrijednosti unutar svakog atributa, ovo je korisno za otkrivanje potencijalnih problema u podacima

Odgovor na ključna pitanja

Skup podataka odgovara traženom skupu podataka gdje ima 15000 redaka i 22 stupaca. Skup je dovoljno raznolik jer sadrži velik broj jedinstvenih vrijednosti po stupcima kao što su tip anime-a, žanr, studio, produceri, ocjena... Većina stupaca su kvantitativni odnosno numerički, dok su neki kategorijski, to su žanr, tip, studio.. Postoje nedostajuće vrijednosti ali su otkrivene analizom, što je važno za kvalitetu analize.

Zaključak analize

Osnovna analiza pokazuje da je anime skup podataka kvalitetan za daljnju obradu podataka i za različite vrste analitičkih i istraživačkih podataka, Velik broj zapisa i raznovrsnih atributa omogućuje detaljno proučavanje trendova, popularnosti, utjecaja žanrova i studija , kao i ponašanje korisnika kroz ocjene i broja članova. Postoje određeni nedostaci, ali ne utječu na kvalitetu analize, zato što postoji puno ključnih elemenata kao što su raznolikost žanrova, veliki broj producijskih studija i različite varijacije u broju epizoda i ocjena.

Zaključak je da ovaj skup pruža odličnu osnovu za naprednije analize.

2. Izrada relacijskog modela i baze podataka

Nakon osnovne analize slijedi izrada relacijske baze podataka. Ovdje ću detaljno opisati kako sam to napravila. Proces izrade relacijske baze podataka uključuje definiciju entiteta, veza, implementaciju modela u MySQL, kao i punjenje baze podacima iz CSV datoteke.

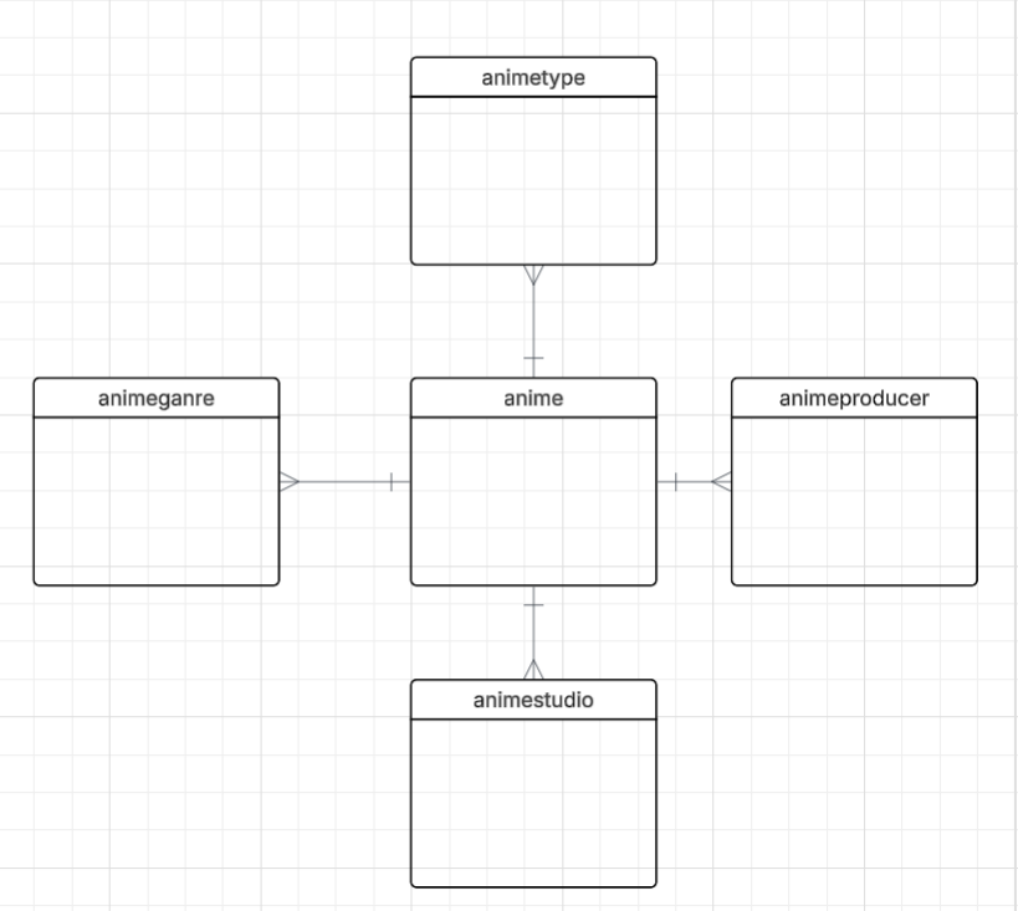
Cilj mi je bio stvoriti strukturirani okvir koji mi omogućava efikasno upravljanje podacima o ključnim atributima kao što su ocjene, žanrovi, studiji...

2.1 Identifikacija entiteta i definicija veza

Identificirala sam sljedeće entitete na temelju analize podataka. Anime entiet predstavlja glavu tablicu te sadrži osnovne informacije kao što su naslov, anime_id, type, epizode, duration, rank, popularity, englesko i japansko ime...Sljedeći entitet je anime_urls koji označava URL adrese povezane s anime-om, uključuje anime_url i image_url. Veza entiteta anime_url i anime je jedan na jedan jer jedan URL pokazuje na jedan anime. Nakon toga imamo anime_genre i ovaj entitet sadrži informacije o žanrovima povezane s pojedinim animeom, veza između anime i anime_genre entiteta je jedan na više. Sljedeći entitet je anime_producer u kojemu su informacije koji sve producenti su sudjelovali u produkciji anime-a. Veza entiteta anime i anime_producer je jedan na više. Entitet anime studio bilježi studije odgovorne za produkciju animea gdje je veza između entiteta anime i anime_studio jedan na više jer jedan studio može proizvesti više animea, a jedan anime se proizvodi u jednom studiju, a anime type kategorizira anime prema tipu što mogu biti TV serija, film ili OVA (original video animation) i veza je jedan na jedan. I imamo entitet anime_stats koji sadrži detaljne statistike o popularnosti, ocjenama i korisničkim preferencijama.

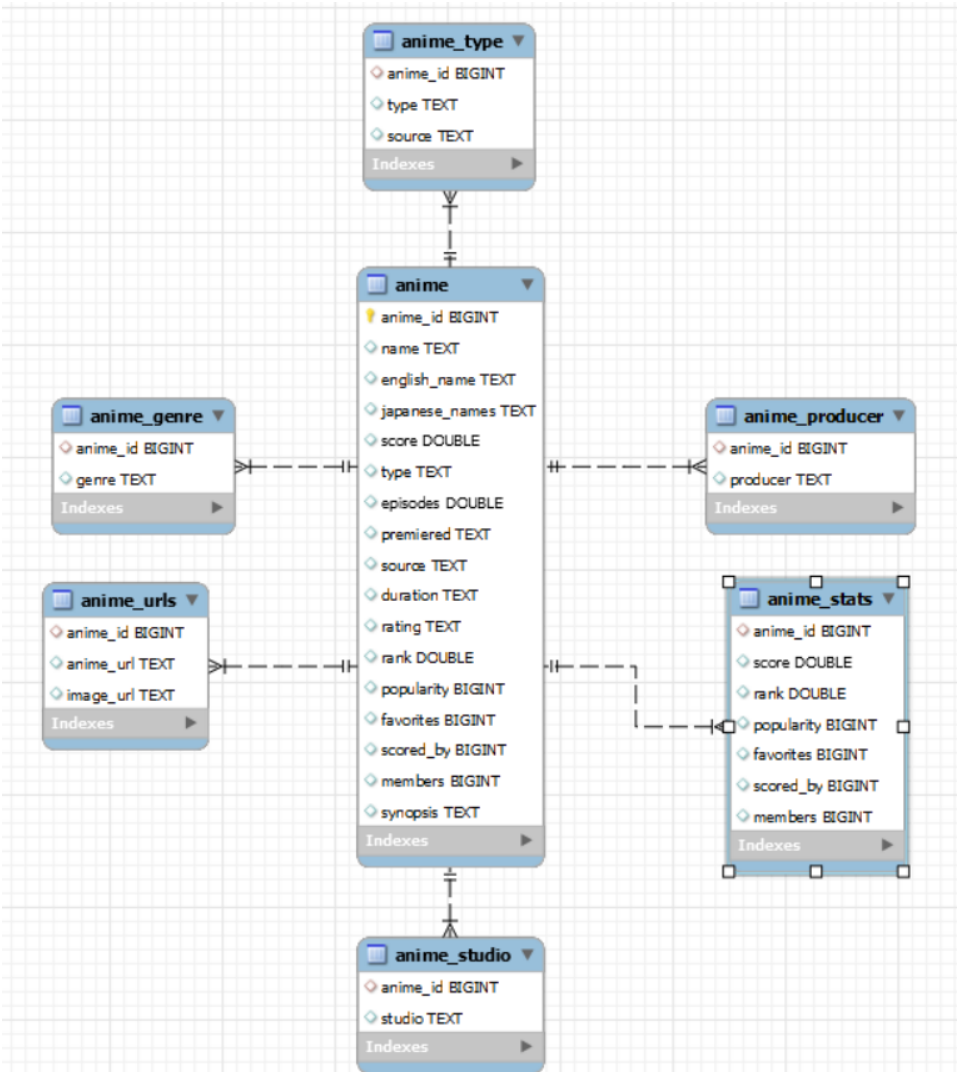
2.2 ER dijagram (konceptualni model)

Sljedeća slika prikazuje ER dijagram odnosno konceptualni model koji prikazuje entitete i njihove međusobne veze. U sredini je glavi entitet *anime* oko kojeg su ostali entiteti spojeni vezom 1:N



2.3 EER dijagram

Za detaljniju obradu modela koristila sam EER dijagram koji uključuje dodatne atribute i prikazuje odnose između tablica. Detaljnije prikazuje strukturu baze podataka s precizno definiranim atributima, tipovima podataka i ogračenjima referencijalnog integriteta, također sve strani ključ veze su vidljive i osiguravaju konzistentnost podataka.



2.4 Implementacija podataka

Koristila sam MySQL DBMS za implementaciju baze podataka. Tablice sam kreirala prema modelu, a podatke učitala tako što sam koristila Python biblioteke pandas i SQLAlchemy. Prije same implementacije napravila sam potpuno resetiranje baze podataka da bi se osiguralo čisto okruženje.

```
df = pd.read_csv(CSV_PATH)
df = df.drop_duplicates(subset=['anime_id'])
print("CSV učitan i očišćen")
```

Dio koda koji služi za učitavanje podataka

```
engine = create_engine(DB_CONNECTION)
print("Spojeno na bazu")
```
Dio koda koji služi za spajanje na bazu podataka, i ispisuje da je spojeno na bazu.

```
with engine.connect() as conn:
    conn.execute(text("SET FOREIGN_KEY_CHECKS = 0;"))
    conn.execute(text("DROP DATABASE IF EXISTS anime_db;"))
    conn.execute(text("CREATE DATABASE anime_db;"))
    conn.execute(text("USE anime_db;"))
    conn.execute(text("SET FOREIGN_KEY_CHECKS = 1;"))
```
Dio koda koji resetira bazu da bi ju očistio za daljnje upotrebljivanje i osigurao čisto okruženje

2.5 Implementacija referencijalnog integriteta

Sustav foreign key (strani ključ) osigurava konzistentnost podataka kroz CASCADE DELETE operacije, za to mi služi sljedeći dio koda:

```
with engine.connect() as conn:
    # URLs
    conn.execute(text("""
        ALTER TABLE anime_urls
        ADD CONSTRAINT fk_urls_anime
        FOREIGN KEY (anime_id) REFERENCES anime(anime_id)
        ON DELETE CASCADE
    """))
```
ON DELETE CASCADE automatski briše sve povezane zapise ako se briše anime.

Za provjeru stranih veza služi ovaj dio koda:

```
with engine.connect() as conn:
    result = conn.execute(text("""
        SELECT
            TABLE_NAME,
            CONSTRAINT_NAME,
            REFERENCED_TABLE_NAME
        FROM
            information_schema.KEY_COLUMN_USAGE
```

```
        WHERE
            REFERENCED_TABLE_SCHEMA = 'anime_db'
            AND REFERENCED_TABLE_NAME IS NOT NULL;
    """))
```

Prikazuje sve tablice koje sadržavaju foreign key vezu, odnosno tablice koje su povezane s drugim tablicama preko stranog ključa.

Zaključak analize

Izradila sam relacijski model i implementirala bazu podataka za anime dataset, time sam postavila čvrst i strukturiran temelj za daljnu analizu i za daljnje upravljanje podacima. Identifikacijom entiteta, definiranjem veza i normalizacijom podataka osigurala sam da baza bude pregledna, efikasna i jednostavna za proširivanje. Koristila sam ON DELETE CASCADE koji je dodatno povećao sigurnost i konzistentnost podataka, kao i implementacija referencijalnog integriteta.

3. Kreiranje dimenzijskog modela i star sheme

U ovom dijelu prikazat ću kako sam za “Top 15 000 ranked anime” napravila dimenzijski model, i izradila star shemu.

3.1 Mjere u tablici činjenica

```
class FactAnimeStats(Base):
    __tablename__ = 'fact_anime_stats'
    fact_id = Column(BigInteger, primary_key=True, autoincrement=True)
```

Ovaj dio koda prikazuje mjere koje se analiziraju i agregiraju u tablici činjenica, mjere u ovom dataframe-u su score koji označuje prosječnu ocjenu anime-a od strane korisnika, members što je broj članova koji gledaju i prate anime, popularity_rank je rang koliko je pojedini anime popularan, favourites označava broj korisnika koji su neki anime dodali u *omiljene* scored_by je broj korisnika koji su ocijenili anime te rank koji označava ukupan rang svakog anime-a na platformi.

3.2. Dimenzije

Ovaj model također sadrži *dimenzije* i one opisuju kontekst mjera u tablici činjenica, ovaj model sadrži šest glavnih dimenzija, dim_anime opisuje osnovne karakteristike pojedinog anime-a

```
class DimAnime(Base):
    __tablename__ = 'dim_anime'
    anime_id = Column(BigInteger, primary_key=True)  # ID iz CSV-a
    name = Column(String(255))
    type = Column(String(50))  # TV, Movie, OVA
```

```
    episodes = Column(Integer)
    score = Column(Float)
    members = Column(Integer)
```

Sljedeća dimenzija je dim_genre i ona opisuje žanr anime sadržaja

```
class DimGenre(Base):
    __tablename__ = 'dim_genre'
    genre_id = Column(Integer, primary_key=True, autoincrement=True)
    genre_name = Column(String(100), unique=True)
```

Dim_studio je dimenzija koja opisuje animacijske studije

```
class DimStudio(Base):
    __tablename__ = 'dim_studio'
    studio_id = Column(Integer, primary_key=True, autoincrement=True)
    studio_name = Column(String(255), unique=True)
```

Dim_producer dimenzija koja opisuje producente anime-a.

```
class DimProducer(Base):
    __tablename__ = 'dim_producer'
    producer_id = Column(Integer, primary_key=True, autoincrement=True)
    producer_name = Column(String(255), unique=True)
```

Dim_rating sam koristila za kategoriziranje anime-a po ocjenama

```
class DimRating(Base):
    __tablename__ = 'dim_rating'
    rating_id = Column(Integer, primary_key=True)
    score_range = Column(String(20))
    category = Column(String(30))
```

I na kraju dim_time koji mi je služio za vremensko analiziranje

```
class DimTime(Base):
    __tablename__ = 'dim_time'
    time_id = Column(Integer, primary_key=True, autoincrement=True)
    year = Column(Integer)
    decade = Column(String(10))
    era = Column(String(20))
```

3.3. Definiranje hijerarhija u dimenzijama

Neke dimenzije imaju svoju hijararhiju koja omogućava analizu podataka na različitim razinama, hijerarhija u dimenziji anime omogućuje analizu od općeg tipa sadržaja kao što su TV, Movie, OVA do detaljnih statistika. Hijerarhija u dimeziji popularnosti omogućuje analizu popularnosti na različitim razinama mjerenja

3.4. Many – to – many veze

Jedan anime može biti povezan s više žanrova, studija ili producenata, a istovremeno jedan žanr ili studio ili producent može biti povezan s više anime-a, i zbog toga je rješavanje složenih veza između entiteta jedan od ključnih izazova. Zato radim dodatne tablice zvane bridge tablicama.

Postoji many-to-many veza između veze anime i žanrovi, gdje jedan anime može imati više žanrova (“Attack on Titan” je istovremeno i akcija i drama i fantazija). Također jedan žanr može pripadati na više anime-a. Veza između anime-a i studija također može biti many-to-many gdje jedan anime može biti produciran od strane više studija i jedan studio može producirati više anime-a. I zadnje što ima many-to-many vezu je anime i producenti gdje jedan anime može imati više producenata i jedan producent može raditi na više anime-a.

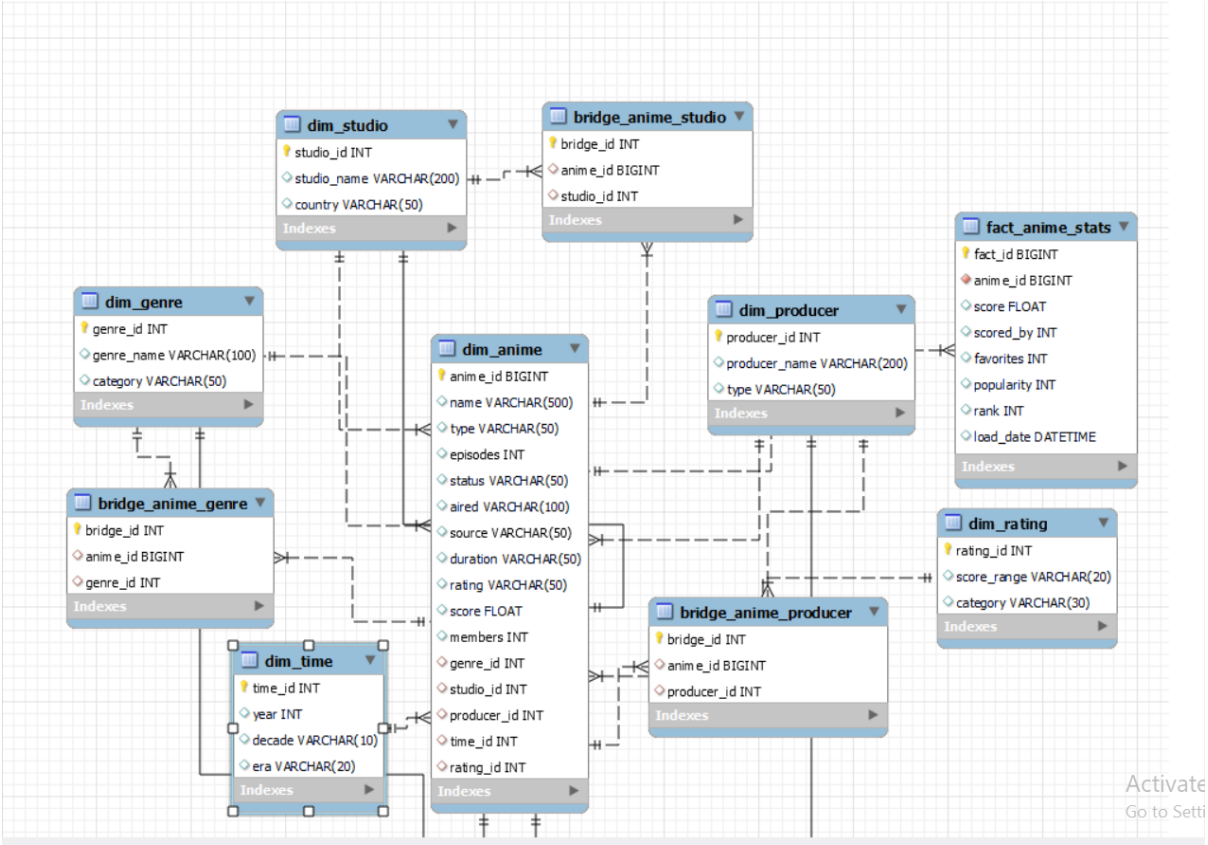
```
class AnimeGenre(Base):
    __tablename__ = 'anime_genre'
    anime_id = Column(BigInteger, ForeignKey('dim_anime.anime_id'),
primary_key=True)
    genre_id = Column(Integer, ForeignKey('dim_genre.genre_id'),
primary_key=True)

class AnimeStudio(Base):
    __tablename__ = 'anime_studio'
    anime_id = Column(BigInteger, ForeignKey('dim_anime.anime_id'),
primary_key=True)
```

```
    studio_id = Column(Integer, ForeignKey('dim_studio.studio_id'),
primary_key=True)

class AnimeProducer(Base):
    __tablename__ = 'anime_producer'
    anime_id = Column(BigInteger, ForeignKey('dim_anime.anime_id'),
primary_key=True)
    producer_id = Column(Integer, ForeignKey('dim_producer.producer_id'),
primary_key=True)
```

Također svaka tablica sadržava kompozitni primary key koji je sastavljen od dva strana ključa, što osigurava jedinstvenost kombinacije (1anime + 1žanr = 1zapis).

3.5. Star shema

Star shema prikazuje dimenzijski model, u središtu se nalazi fact_anime_stats i dimenzije su povezane izravno na nju preko stranih ključeva.



Slika prikazuje vizualni prikaz (EER dijagram).

Činjenica, šest dimenzijskih tablica dim_anime, dim_genre, dim_studio, dim_producer ,dim rating, dim_time su vidljive, kao i tablice koje rješavaju many-to-many veze između glavne i ostalih dimenzija odnosno bridge tablice.

Prednosti star sheme su bolje performanse, skalabilnost te analitičke mogućnosti.

3.6. Implementacija modela SQLAlchemy/Python

Kako bih izradila bazu i tablice koristila sam Python i SQLAlchemy.

```
from sqlalchemy import create_engine, Column, Integer, BigInteger, String,
ForeignKey, Float, Boolean, text
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker
from datetime import datetime



temp_engine = create_engine('mysql+pymysql://root:root@localhost:3306/',
echo=True)
with temp_engine.connect() as conn:
    conn.execute(text("CREATE DATABASE IF NOT EXISTS anime_warehouse"))
    conn.commit()



engine =
create_engine('mysql+pymysql://root:root@localhost:3306/anime_warehouse',
echo=True)
Session = sessionmaker(bind=engine)
session = Session()
Base = declarative_base()



class DimAnime(Base):
    __tablename__ = 'dim_anime'
    anime_id = Column(BigInteger, primary_key=True)
    name = Column(String(255))
    type = Column(String(50))
    episodes = Column(Integer)
    score = Column(Float)
    members = Column(Integer)

class DimGenre(Base):
    __tablename__ = 'dim_genre'
    genre_id = Column(Integer, primary_key=True, autoincrement=True)
    genre_name = Column(String(100), unique=True)
```

```python
class DimStudio(Base):
    __tablename__ = 'dim_studio'
    studio_id = Column(Integer, primary_key=True, autoincrement=True)
    studio_name = Column(String(255), unique=True)

class DimProducer(Base):
    __tablename__ = 'dim_producer'
    producer_id = Column(Integer, primary_key=True, autoincrement=True)
    producer_name = Column(String(255), unique=True)

class DimRating(Base):
    __tablename__ = 'dim_rating'
    rating_id = Column(Integer, primary_key=True)
    score_range = Column(String(20))
    category = Column(String(30))

class DimTime(Base):
    __tablename__ = 'dim_time'
    time_id = Column(Integer, primary_key=True, autoincrement=True)
    year = Column(Integer)
    decade = Column(String(10))
    era = Column(String(20))



class AnimeGenre(Base):
    __tablename__ = 'anime_genre'
    anime_id = Column(BigInteger, ForeignKey('dim_anime.anime_id'),
primary_key=True)
    genre_id = Column(Integer, ForeignKey('dim_genre.genre_id'),
primary_key=True)

class AnimeStudio(Base):
    __tablename__ = 'anime_studio'
    anime_id = Column(BigInteger, ForeignKey('dim_anime.anime_id'),
primary_key=True)
    studio_id = Column(Integer, ForeignKey('dim_studio.studio_id'),
primary_key=True)

class AnimeProducer(Base):
    __tablename__ = 'anime_producer'
    anime_id = Column(BigInteger, ForeignKey('dim_anime.anime_id'),
primary_key=True)
    producer_id = Column(Integer, ForeignKey('dim_producer.producer_id'),
primary_key=True)
```

```
class FactAnimeStats(Base):
    __tablename__ = 'fact_anime_stats'
    fact_id = Column(BigInteger, primary_key=True, autoincrement=True)


    anime_id = Column(BigInteger, ForeignKey('dim_anime.anime_id'),
nullable=False)
    genre_id = Column(Integer, ForeignKey('dim_genre.genre_id'),
nullable=True)
    studio_id = Column(Integer, ForeignKey('dim_studio.studio_id'),
nullable=True)
    producer_id = Column(Integer, ForeignKey('dim_producer.producer_id'),
nullable=True)
    rating_id = Column(Integer, ForeignKey('dim_rating.rating_id'),
nullable=True)
    time_id = Column(Integer, ForeignKey('dim_time.time_id'), nullable=True)


    score = Column(Float)
    members = Column(Integer)
    popularity_rank = Column(Integer)
    favorites = Column(Integer)
    scored_by = Column(Integer)
    rank = Column(Integer)
```

Zaključak

Implementirala sam tablice, šest dimenzijskih tablica anime, genre, studio, time, rating, producer i tri bridge tablice producer, genre i studio koje mi omogućuju many-to-many vezu i jednostavno povezivanje sa anime-a sa više žanrova, producenata i/ili studija što mi pomaže kod skalabilnosti i normalizacije baze. Također koristila sam i fact_anime_stats tablicu koja sadrži ključne mjere kao što su ocjene, popularnost, omiljeno... Pravljenje star sheme mi je omogućilo da koristim brze upite i jednostavno širenje modela ko i analizu trendova u anime industriji

4. Punjenje tablica i ETL

U ovom dijelu ću prikazati kako sam popunila tablice iz dva izvora, također sam provela ETL proces i provjeru unosa podataka za anime skup podataka. Koristila sam Python sa bibliotekama SQLAlchemy i Pandas i koristila sam PySpark.

4.1. Punjenje tablica

koristila sam dva izvora za ovaj checkpoint, a to cu CSV datoteka koju sam preuzela sa linka: https://www.kaggle.com/datasets/quanthan/top-15000-ranked-anime-dataset-update-to-32025?resource=download . I dimenzijski model koji sam kreirala u trećem checkpointu koristeći Python sa SQLAlchemy bibliotekom. Također sam koristila i dva alata Python u kojem sam uključila Pandas biblioteku, koji mi pomaže u detaljnoj obradi i kod kreiranja dimenzijskog modela. Drugi alat koji sam koristila je Apache Spark za obradu velike količine podataka.

4.2. ETL proces

ETL proces ima više koraka, te nam omogućava da spajamo podatke iz različitih izvora u jednu bazu podataka. Postoje tri koraka, prvi korak je *extract* i koristila sam

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import split, explode, col, when, isnan, isnull
```

Ovaj dio koda koji kreira spark aplikaciju i upravlja sparkom. Također omogućuje čitanje podataka iz različitih izvora kao što su CSV, json i baze podataka.

```
spark = SparkSession.builder \
    .appName("Anime ETL") \
    .config("spark.jars", "C:\Users\rahel\Downloads\mysql-connector-j-
9.3.0\mysql-connector-j-9.3.0\mysql-connector-j-9.3.0.jar") \
    .getOrCreate()
```

Ovaj dio koda pokreće spark sesiju sa MySQL connectorom.

```
csv_path = r"C:\Users\rahel\Downloads\archive\top_anime_dataset.csv"
    print(f"Učitavam CSV iz: {csv_path}")
    df = spark.read.option("header", True).option("inferSchema",
True).csv(csv_path)
    print("CSV učitan. Kolone:", df.columns)
```

I ovaj dio učitava CSV datoteku , preko path-a
"C:\Users\rahel\Downloads\archive\top_anime_dataset.csv"

Spark koristimo jer automatski prepozna tip podatka i omogućuje efikasnu obradu velikih količina podataka tako što koristi distribuirano računanje. Nakon *extract-a* slijedi *transformacija,* u ovom dijelu radi se transformacija u odgovarajući format za skladište podataka. Prvi dio koda kreira glavnu anime tablicu

```
print("\nKreiram Anime tabelu...")
    anime_df = df.select("anime_id", "Name", "Score", "Genres", "Episodes",
"Type", "Studios") \
                 .dropDuplicates() \
                 .filter(col("anime_id").isNotNull())
    anime_df = anime_df.fillna({"Episodes": 0, "Score": 0.0})
    anime_df.show(5)
```

Nakon toga napisala sam kod za normalizaciju žanrova

```
genre_df = df.select("anime_id", "Genres") \
                 .filter(col("Genres").isNotNull()) \
                 .withColumn("genre", explode(split(col("Genres"), ","))) \
                 .select("anime_id", col("genre").alias("genre_name")) \
                 .dropDuplicates()
```

Stupac *žanr* sadrži više žanrova pa se sa `explode(split` rastavljaju u zasebne redove. I to omogućava many-to-many vezu između animea i žanrova. Nakon toga sam radila normalizaciju studija

```
studio_df = df.select("anime_id", "Studios") \
                  .filter(col("Studios").isNotNull()) \
                  .withColumn("studio", explode(split(col("Studios"), ","))) \
                  .select("anime_id", col("studio").alias("studio_name")) \
```

```
                    .dropDuplicates()
```

Ovdje se također radi poseban red za svaki zapis, jer postoji više studija koji su odvojeni zarezom, isto kao i za žanr dijeljenje zapisa u poseban stupac radi se sa `explode(split(.` Nakon toga kreira se rating tablica

```
  rating_df = df.select("anime_id", "Score", "Scored By", "Popularity",
"Rank") \
                    .filter(col("anime_id").isNotNull())
```

Ovaj dio koda će izdvojiti sve stupce koji su povezani sa ocjenama i popularnosti anime-a u zasebnu tablicu, a to su score, scored by, popularity, rank.

Zatim slijedi

```
type_df = df.select("anime_id", "Type") \
                .filter(col("Type").isNotNull()) \
                .dropDuplicates()
```

Ovaj dio koda kreira tablicu za tipove anime-a i ona povezuje svaki anime s njegovim tipom, tipovi su TV, Movie i OVA. `.dropDuplicates()` kao i kod prošlih dijelova kodova ovaj dio koda osigurava da se svaki anime_id pojavljuje jednom s odgovarajućim tipom. I zadnji korak ETL procesa je učitavanje, odnosno *load* , gdje se transformirani podaci učitavaju u MySQL bazu koristeći JDBC.

```
    jdbc_url = "jdbc:mysql://localhost:3306/anime_db"
    properties = {
        "user": "root",
        "password": "root",
        "driver": "com.mysql.cj.jdbc.Driver"
    }
```

U ovom dijelu koda se koristio JDBC za povezivanje MySql baze podataka. Zatim

```
   print("\nAnime")
    anime_df.write.jdbc(url=jdbc_url, table="anime", mode="append",
properties=properties)
    print("anime upisana")

    print("žanr")
    genre_df.write.jdbc(url=jdbc_url, table="genre", mode="append",
properties=properties)
    print("žanr upisana")

    print("studio")
    studio_df.write.jdbc(url=jdbc_url, table="studio", mode="append",
properties=properties)
    print("studio upisana")

    print("rating")
```

```
    rating_df.write.jdbc(url=jdbc_url, table="rating", mode="append",
properties=properties)
    print("rating upisana")

    print("tip")
    type_df.write.jdbc(url=jdbc_url, table="type", mode="append",
properties=properties)
    print("tip upisana")
```

U ovom dijelu koda kreiraju se tablice, glavna tablica “anime”, te tablice “genre”, ”studio”, ”rating” i tablica ”tip”.

4.3. Data Validation

Kada su se podaci učitali u skladište, provodi se provjera da se osigura točnost podataka kao i kvaliteta. Prvo radim brojanje redaka, i to radim sa select funkcijom u MySQL

```
select 'dim_anime' as tablica, count() as broj_zapisa from dim_anime
union all select 'dim_genre', count() from dim_genre union all
select 'dim_studio', count() from dim_studio union all select
'dim_producer', count() from dim_producer union all select
'dim_rating', count() from dim_rating union all select 'dim_time',
count() from dim_time union all select 'anime_genre', count() from
anime_genre union all select 'anime_studio', count() from
anime_studio union all select 'anime_producer', count() from
anime_producer union all select 'fact_anime_stats', count() from
fact_anime_stats;
```

	tablica	broj_zapisa
▶	dim_anime	14965
	dim_genre	21
	dim_studio	836
	dim_producer	774
	dim_rating	5
	dim_time	65
	anime_genre	0
	anime_studio	0
	anime_producer	0
	fact_anime_stats	15000

Ova slika prikazuje svaku tablicu i koliko zapisa ima.

Ova select funkcija mi je pomogla da vidim koliko tablica imam i koliko je zapisa zapisano u svakoj tablici. Vidim da imam 4 tablice, a dim_anime sa 14965 zapisa, dim_genre sa 21 zapisom, dim_studio sa 938 zapisa i dim_producer sa 1021 zapisom.

Nakon što sam izbrojala redke provjeravam strukturu tablice sa *describe* funkcijom

```
DESCRIBE dim_anime;
DESCRIBE dim_genre;
DESCRIBE dim_studio;
DESCRIBE dim_producer;
```

	Field	Type	Null	Key	Default	Extra
▶	anime_id	bigint	NO	PRI	NULL	auto_increment
	name	varchar(255)	YES		NULL	
	type	varchar(50)	YES		NULL	
	episodes	int	YES		NULL	
	score	float	YES		NULL	
	members	int	YES		NULL	

Slika prikazuje rezultat DESCRIBE dim_anime funkcije.

Rezultat prikazuje da je tablica ispravno kreirana i sadrži osnovne redove, sadrži anime_id što označava glavni ključ odnosno da svaki anime sadrži primary key. Svi ostali redovi mogu biti NULL i podaci se mogu unositi.

	Field	Type	Null	Key	Default	Extra
▶	genre_id	int	NO	PRI	NULL	auto_increment
	genre_name	varchar(100)	YES	UNI	NULL	

Slika prikazuje rezultat DESCRIBE dim_genre funkcije

	Field	Type	Null	Key	Default	Extra
▶	studio_id	int	NO	PRI	NULL	auto_increment
	studio_name	varchar(255)	YES	UNI	NULL	

Slika prikazuje rezultat DESCRIBE dim_studio funkcije

	Field	Type	Null	Key	Default	Extra
▶	producer_id	int	NO	PRI	NULL	auto_increment
	producer_name	varchar(255)	YES	UNI	NULL	

Slika prikazuje rezultat DESCRIBE dim_producer funkcije

Nakon što sam provjerila ispravnost tablica, provjeravam jedinostvenost i referencijalni integritet. Provjeru jedinstvenosti radim sa

```
select count(distinct anime_id) as jedinstveni_anime from dim_anime;
```

	jedinstveni_anime
▶	14965

Slika prikazuje rezultat SELECT COUNT DISTINCT funkcije

Ova funkcija mi pomaže vidjeti koliko točno ima jedinstvenih anime-a, odnosno prebrojava anime_id.

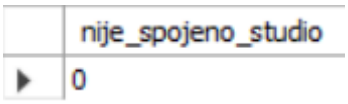
Nakon toga sam napisala funkciju koja provjerava referencijalni integritet između tablica

```
select count(*) as nije_spojeno_zanr

from dim_anime a

where a.genre_id is not null

and a.genre_id not in (select genre_id from dim_genre);
```

	nije_spojeno_zanr
▶	0

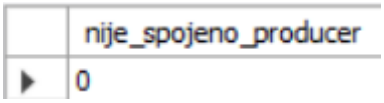
Slika prikazuje koliko stranih ključeva se nije uspjelo spojiti.

Ovu funkciju sam koristila na sve tablice, da provjerim postoje li nespojeni strani ključevi.



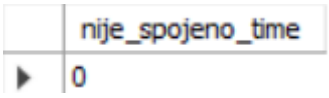
	nije_spojeno_studio
▶	0

Slika prikazuje koliko stranih ključeva nije spojeno između tablica anime i anime_studio



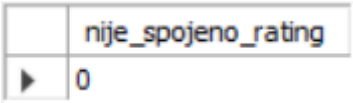
	nije_spojeno_producer
▶	0

Slika prikazuje koliko stranih ključeva nije spojeno između tablica anime i producer



	nije_spojeno_time
▶	0

Slika prikazuje koliko stranih ključeva nije spojeno između tablica anime i time



	nije_spojeno_rating
▶	0

Slika prikazuje koliko stranih ključeva nije spojeno između tablica anime i rating

Kada sam provjerila postoji li jedinstveni ključ i strani ključevi, napisala sam *select* funkcije da provjerim jesu li ostali podaci upisani.

```
select min(score) as min_ocjena, max(score) as max_ocjena,
avg(score) as prosjek from dim_anime;
```

Ova select funkcija prikazuje najmanju i najvišu danu ocjenu, kao i sveukupan prosjek ocjena.

	min_ocjena	max_ocjena	prosjek
▶	5.57	9.31	6.673033746795196

Slika prikazuje rezultat select funkcije za računanje minimalne, maksimalne i prosječne ocjene za anime.

Zatim sam napisala select funkciju koja prikazuje prvih deset najpopularnijih žanrova i broj koliko anime-a je napravljeno u tom žanru.

```
select g.genre_name, count(a.anime_id) as broj_animea

from dim_genre as g

left join dim_anime as a on g.genre_id = a.genre_id group by
g.genre_name, g.genre_id

order by broj_animea desc limit 10;
```

	genre_name	broj_animea
▶	Action	4124
	Comedy	3164
	Adventure	1306
	Hentai	1007
	Drama	989
	Fantasy	658
	Slice of Life	448
	Sports	308
	Romance	243
	Sci-Fi	230

Slika prikazuje rezultat select funkcije za prikazivanje top deset anime žanra sa brojem anime-a koji su na tu temu.

Zatim sam napisala select funkciju za prikaz prvih deset anime studija sa brojem anime-a kojeg su napravili.

```
select s.studio_name, count(a.anime_id) as broj_animea

from dim_studio as s

left join dim_anime as a on s.studio_id = a.studio_id
```

```
group by s.studio_name

order by broj_animea desc limit 10;
```

	studio_name	broj_animea
▶	Toei Animation	716
	Sunrise	531
	J.C.Staff	382
	Madhouse	342
	Production I.G	317
	TMS Entertainment	299
	Studio Deen	268
	Pierrot	258
	OLM	257
	A-1 Pictures	238

Slika prikazuje rezultat select funkcije za prikaz prvih deset studija i broj anime-a kojeg su napravili.

Sljedeća select funkcija mi prikazuje prvih petnaest anime serija/filmova po ocjeni

```
Select anime_id, name, round(score, 2) as ocjena, members, type
from dim_anime
where score > 0
order by score desc, members desc
limit 15;
```

	anime_id	name	ocjena	type
▶	52991	Sousou no Frieren	9.31	1035677
	5114	Fullmetal Alchemist: Brotherhood	9.1	3483268
	9253	Steins;Gate	9.07	2667979
	60022	One Piece Fan Letter	9.06	91278
	38524	Shingeki no Kyojin Season 3 Part 2	9.05	2407861
	28977	Gintama°	9.05	656687
	39486	Gintama: The Final	9.04	163478
	11061	Hunter x Hunter (2011)	9.03	2981476
	9969	Gintama'	9.02	580364
	15417	Gintama': Enchousen	9.02	338134
	820	Ginga Eiyuu Densetsu	9.01	337614
	41467	Bleach: Sennen Kessen-hen	9	616144
	43608	Kaguya-sama wa Kokurasetai: Ult...	8.99	1008941
	34096	Gintama.	8.98	328137
	42938	Fruits Basket: The Final	8.96	515201

Slika prikazuje tablicu najboljih 15 anime filmova/serija po ocjeni

Zaključak

U ovom dijelu uspjela sam implementirati ETL proces, odnosno dodala sam i obradila podatke za anime skup podataka. Koristila sam PySpark za distribuiranu obradu, kao i sami Python u koji sam implementirala SQLAlchemy biblioteku što mi je pomoglo za detaljniju analizu. Korištenje PySparka omogućilo mi je brzo i efikasno ubacivanje zapisa, i ubačeno je 14.965 anime zapisa, a korištenje SQLAlchemy mi je pomoglo kod kreiranja dimenzijskog modela sa stranim ključ vezama. Također sam koristila MySQL za prikazivanje zapisanih zapisa, provjerila sam jedinstvenost ključa odnosno ima li svaki anime jedinstven ID, i odgovor je da, ima. Isto tako provjerila sam povezanost stranih ključeva i svi koji trebaju biti spojeni, su spojeni. Iz ostalih select funkcija koje sam napisala mogu zaključiti da je komedija najpopularniji žanr, Sousou no Frieren je najbolje ocjenjeni anime, te da Toei Animation najpoznatiji anime studio.

5. Grafički prikaz podataka u Pythonu

Za grafički prikaz koristila sam Python kod koji koristi Pandas biblioteku za obradu podataka. Za izradu grafova i generiranje grafova u PDF slike koristi se Matplotlib i Seaborn.

5.1. Učitavanje podataka

Prvo sam učitala potrebne biblioteke

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
```

Pandas služi za manipulaciju podataka i njega sam uvijek koristila, dok MAatplotlib služi za osnovne grafove, a Seaborn za napredne vizualizacije. Uključila sam i os biblioteku koja upravlja direktorijima.

```
plot_dir = './anime_plots'
os.makedirs(plot_dir, exist_ok=True)
```

Ovaj dio koda se koristi kod pripreme direktorija za grafove. `os.makedirs(` ovaj dio koda kreira direktoriji anime_plots i tamo mi se učitavaju PDF-ovi grafova. Nakon toga sam napisala

```
df = pd.read_csv('C:/Users/rahel/Downloads/archive/top_anime_dataset.csv')
```

Kako bih učitala anime skup podataka.

5.2 OLAP operacija

```
plt.figure(figsize=(8,5))
df['score'] = pd.to_numeric(df['score'], errors='coerce')
df['score'].dropna().plot(kind='hist', bins=30, color='skyblue',
edgecolor='black')
```

```
plt.title('Distribucija ocjena animea')
plt.xlabel('Ocjena')
plt.ylabel('Broj animea')
plt.tight_layout()
plt.savefig(f"{plot_dir}/distribucija_ocjena.pdf")
plt.close()
```

U ovom dijelu koda radi se distribucija ocjena, ovdje se izdvaja jedna dimenzija a to je dimenzija ocjena. To se naziva OLAP slice operacija i koristi `pd.to_numeric` s parametrom `errors='coerce'` i on prevodi tekstualne vrijednosti u numeričke, i uklanja nedostajuće vrijednosti prije crtanja grafa.

5.3 Drill down operacija

```
plt.figure(figsize=(10,5))
all_genres = df['genres'].dropna().str.split(',').explode().str.strip()
genre_counts = all_genres.value_counts()
genre_counts.plot(kind='bar', color='orchid')
plt.title('Broj animea po žanru')
plt.xlabel('Žanr')
plt.ylabel('Broj animea')
plt.tight_layout()
plt.savefig(f"{plot_dir}/animea_po_zanru.pdf")
plt.close()
```

U ovom dijelu koda radi se detaljna analiza na razini svakog pojedinačnog žanra, odnosno radi se drill-down OLAP operacija. Koristi se `str.split(','` koji dijeli string žanrove na pojedinačne elemente, `explode()` pretvara listu u odvojene redove, a `str.strip()` miče razmake. Zatim dio koda koji automatski broji pojavljivanje svakog žanra je `value_counts()` te ih također sortira silaznim redoslijedom.

5.5. Roll up operacija

```
plt.figure(figsize=(12,6))
genre_score = df.dropna(subset=['genres', 'score']).copy()
genre_score.loc[:, 'score'] = pd.to_numeric(genre_score['score'],
errors='coerce')
```

```
genre_score =
genre_score.assign(genre=genre_score['genres'].str.split(',')).explode('genre'
)
genre_score['genre'] = genre_score['genre'].str.strip()
mean_score_by_genre =
genre_score.groupby('genre')['score'].mean().sort_values(ascending=False)
mean_score_by_genre.plot(kind='bar', color='teal')
plt.title('Prosječna ocjena po žanru')
plt.xlabel('Žanr')
plt.ylabel('Prosječna ocjena')
plt.tight_layout()
plt.savefig(f"{plot_dir}/prosjek_ocjena_po_zanru.pdf")
plt.close()
```

Ovaj dio koda radi agregaciju podataka na višoj razini odnosno roll up OLAP operaciju. `Df.dropna =['genres', 'score'])` dio koda koji uklanja redove gdje nedostaju ključni podaci, a `assign()` metoda kreira novu kolonu, dok `explode()` ispočetka dijeli žanrove u posebne redove. Na kraju `groupby('genre')['score'].mean(),` izračuna prosječnu ocjenu za svaki žanr, što omogućava usporedbu kvalitete između žanrova.

5.5 Slice i dice operacija

```
plt.figure(figsize=(10,5))
top10 = df[['name', 'members']].dropna().copy()
top10.loc[:, 'members'] = pd.to_numeric(top10['members'], errors='coerce')
top10 = top10.sort_values('members', ascending=False).head(10)
sns.barplot(
    x='members',
    y='name',
    data=top10,
    hue='name',
    palette='viridis',
    legend=False
)
plt.title('Top 10 najpopularnijih animea')
plt.xlabel('Broj članova')
plt.ylabel('Anime')
plt.tight_layout()
plt.savefig(f"{plot_dir}/top10_popularnih.pdf")
plt.close()
```

Ovaj dio koda izdvaja “Top 10 anime-a" korištenjem *slice* operacije, a *dice* operaciju koristi za filtriranje po kriterijima. `head(10)` dio koda koji ograničava rezultate, u ovom slučaju ograničava ih na 10 zapisa.

5.6. Distribucija broja epizoda

```
plt.figure(figsize=(8,5))
df['episodes'] = pd.to_numeric(df['episodes'], errors='coerce')
df['episodes'].dropna().plot(kind='hist', bins=30, color='orange',
edgecolor='black')
plt.title('Distribucija broja epizoda')
plt.xlabel('Broj epizoda')
plt.ylabel('Broj animea')
plt.tight_layout()
plt.savefig(f"{plot_dir}/distribucija_epizoda.pdf")
plt.close()
```

Ovaj dio također koristi OLAP slice metodologiju kao i dio za ocjene.

5.7. Kombinirana OLAP operacija

```
plt.figure(figsize=(12,6))
studio_score = df.dropna(subset=['studios', 'score']).copy()
studio_score.loc[:, 'score'] = pd.to_numeric(studio_score['score'],
errors='coerce')
studio_score =
studio_score.assign(studio=studio_score['studios'].str.split(',')).explode('st
udio')
studio_score['studio'] = studio_score['studio'].str.strip()
mean_score_by_studio =
studio_score.groupby('studio')['score'].mean().sort_values(ascending=False).he
ad(15)
mean_score_by_studio.plot(kind='bar', color='salmon')
plt.title('Prosječna ocjena po studiju (top 15)')
plt.xlabel('Studio')
plt.ylabel('Prosječna ocjena')
plt.tight_layout()
plt.savefig(f"{plot_dir}/prosjek_ocjena_po_studiju.pdf")
```

```
plt.close()
```

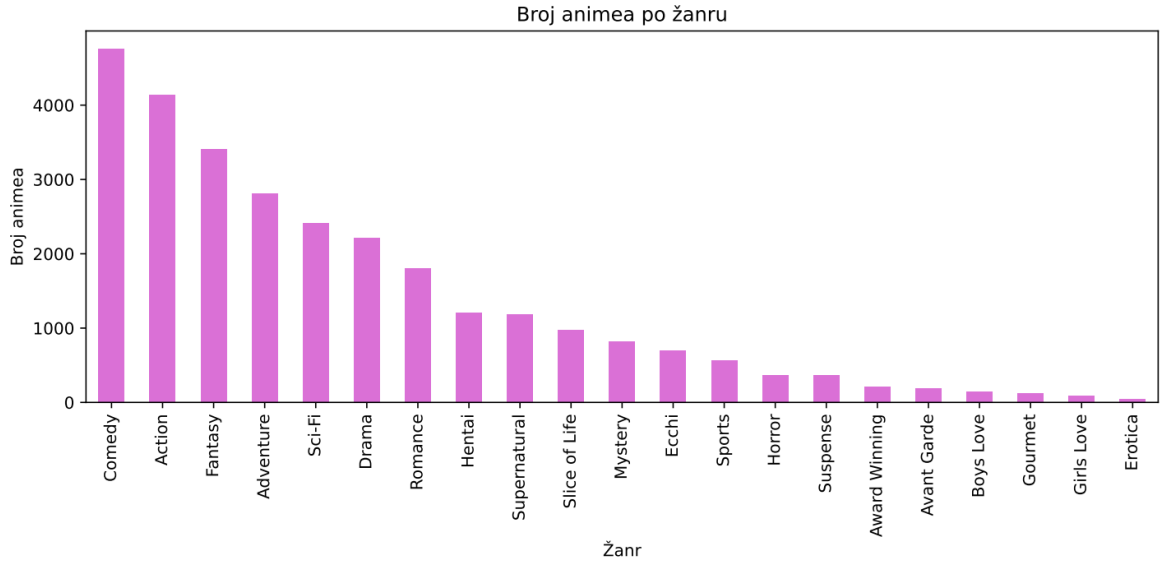
I na kraju imam ovaj dio koda koji kombinira sve OLAP operacije, a to su drill down, agregaciju, slice i dice, gdje se drill down koristi za analizu po studijima, agregaciju sam koristila za izračunavanje prosječne ocjene, slice za ograničenje samo na prvih 15 studija, i na kraju dice koji filtrira po kriterijima kvalitete.

Zaključak

U ovom checkpointu uspješno sam provela analizu grafičkog prikaza koristeći OLAP sustav, odnosno Online Analytical Processing sustav, koji mi je analizirao anime sustav podataka i pokrio sve bitne i ključne dimenzije analize. Koristila sam sve ključne OLAP operacije, a to su slice operacija koja mi je izdvojila pojedinačne dimenzije kod distribucije broja epizoda i distribucije ocjena. Zatim sam koristila drill down operaciju koja mi je poslužila kod detaljne analize kod analize žanrova i studija. Roll up operaciju sam koristila za agregiranje podataka koji su mi služili za izračun prosječnih ocjena. I na kraju dice operacija koju sam koristila za filtriranje podataka po nekim specifičnim kriterijima, i dice operaciju sam koristila kod grafa top 10 animea i top 15 anime studija.

6. Grafički prikaz podataka

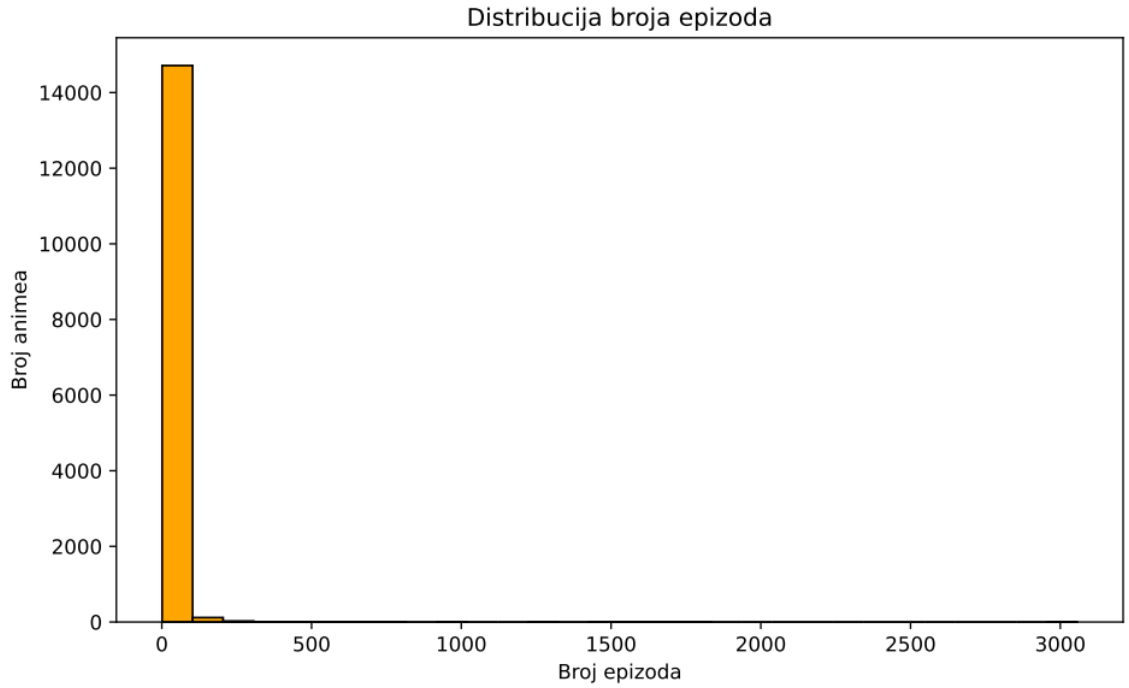
6.1 Broj anime-a po žanru



Slika prikazuje graf distribucije anime-a po svakom žanru poredane silazno, od najzastupljenijih do najrjeđih žanrova.

Prema ovoj slici možemo vidjeti da je *Comedy* odnosno komedija najzastupljeniji žanr među anime-ima i postoji preko 4500 komičnih anime-a. Zatim dolaze akcijski sa 4100 i fantazijski anime sa 3400 anime-a. Adventure, Sci-Fi i Drama također su popularni. Srednju popularnost zauzimaju Ecchi (komedija pomješana sa erotikom), sportski anime, horror, suspense i winning. Najmanje zastupljeni žanrovi anime-a su erotični anime sa manje od 100 anime filmova/serija.

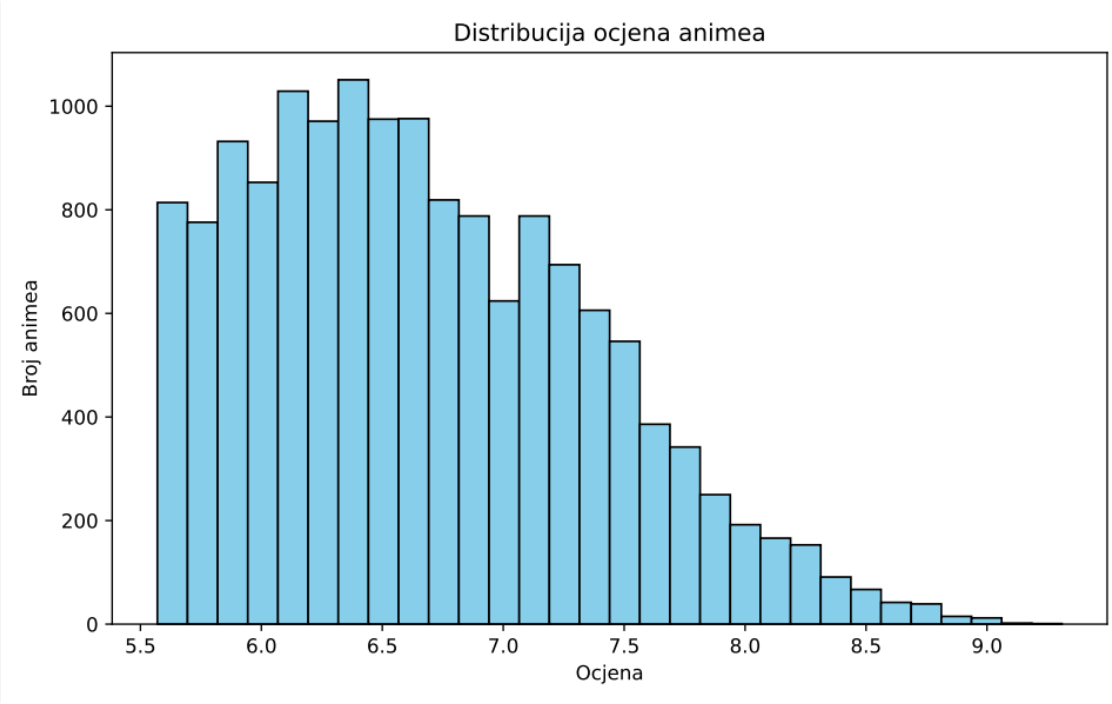
6.2 Distribucija broja epizoda



Slika prikazuje graf distribucije broja epizoda u anime serijama i filmovima.

Više od 14000 anime-a ima vrlo mali broj epizoda, do 100 epizoda, što nam pokazuje da su anime-i zapravo kratke serije. Mali broj anime-a ima više epizoda, to su dugotrajne serije koje nemaju kraja, kao što su One Piece ili Naruto. Ova distribucija prikazuje da je u anime industriji uobičajena praksa proizvoditi kratke serije od 12-26 epizoda.

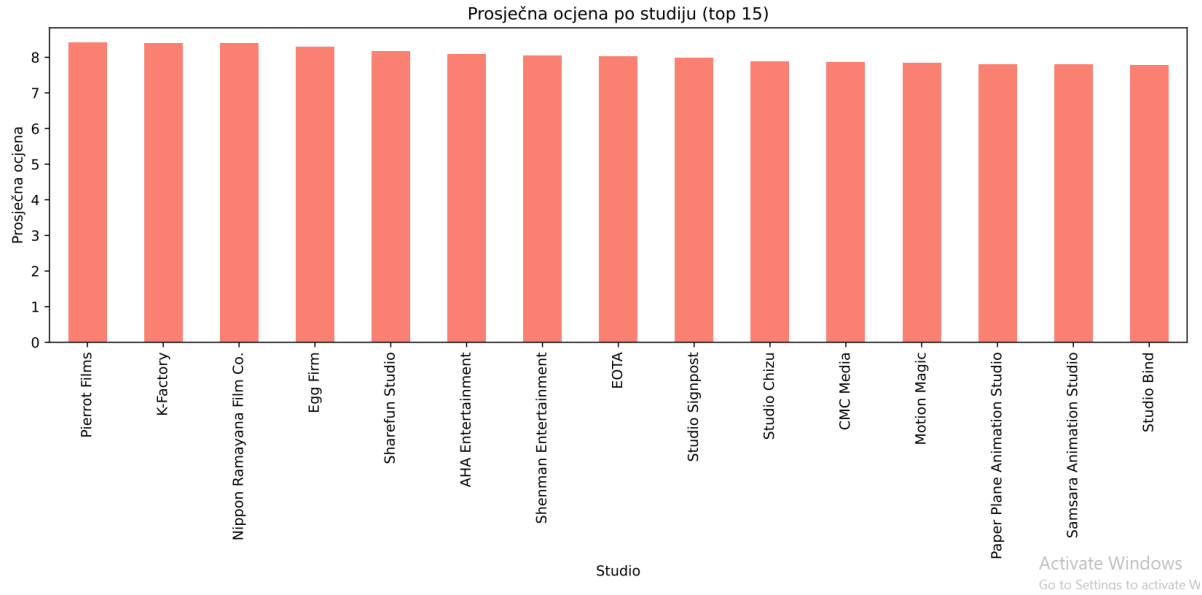
6.3 Distribucija ocjena anime-a



Slika prikazuje distribuciju ocjena anime serija i filmova

Većina anime-a kao što možemo vidjeti sa slike dobija dobre ocjene od 6.0 do 7.0. Mali broj anime-a ima visoke ocjene odnosno ocjene iznad 8.5 ili niske ocjene ispod 5.5, što znači da gledatelji pozitivno reagiraju i ocjenjuju anime, gdje većinom daju normalne i prosječne ocjene.

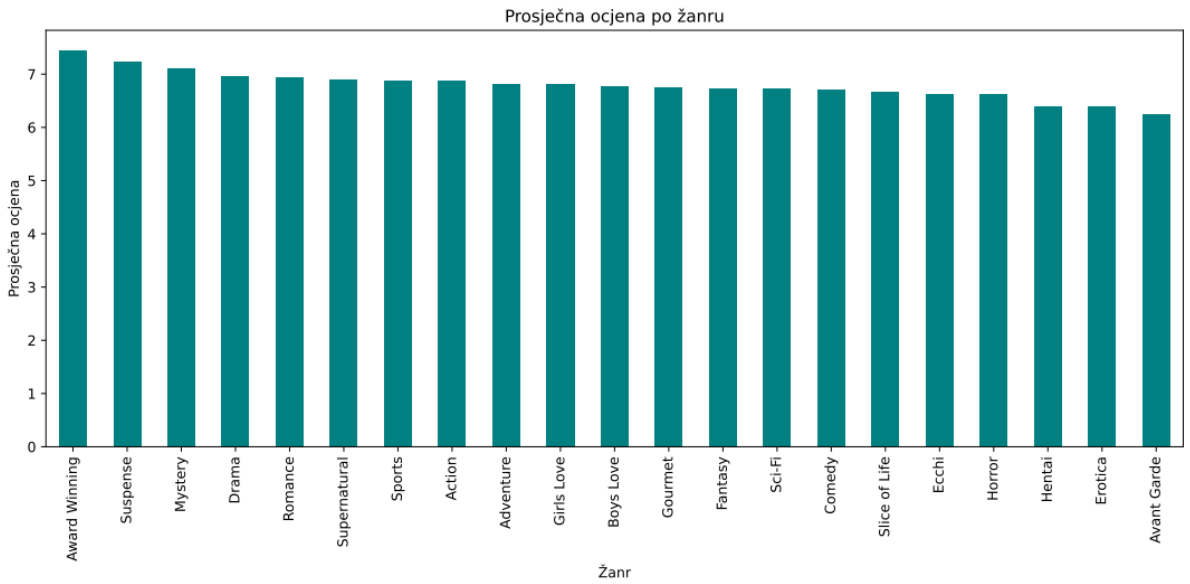
6.4 Top 15 anime studija



Slika prikazuje graf sa prvih petnaest studija.

Svih prikazanih 15 studija ima slične prosječne ocjene, a to su ocjene između 7.6 i 8.4, što prikazuje visoku kvalitetu produkcije među ovim anime studijima. Najpopularniji studiji su Pierrot Films, K-Factory i Nippon Ramayana Film Co koji imaju najvišu prosječnu ocjenu. Iz ovog grafa također vidimo da je razlika između najpoznatijih i one manje poznatih studija razlika u prosječnoj ocjeni vrlo mala, što znači da većina anime studija proizvodi kvalitetan sadržaj. Ova analiza pomaže da identificiramo studije koji proizvode visoko ocjenjene anime filmove ili serije, što može pomoći kod preporuke ili investicijske odluke u industriji.

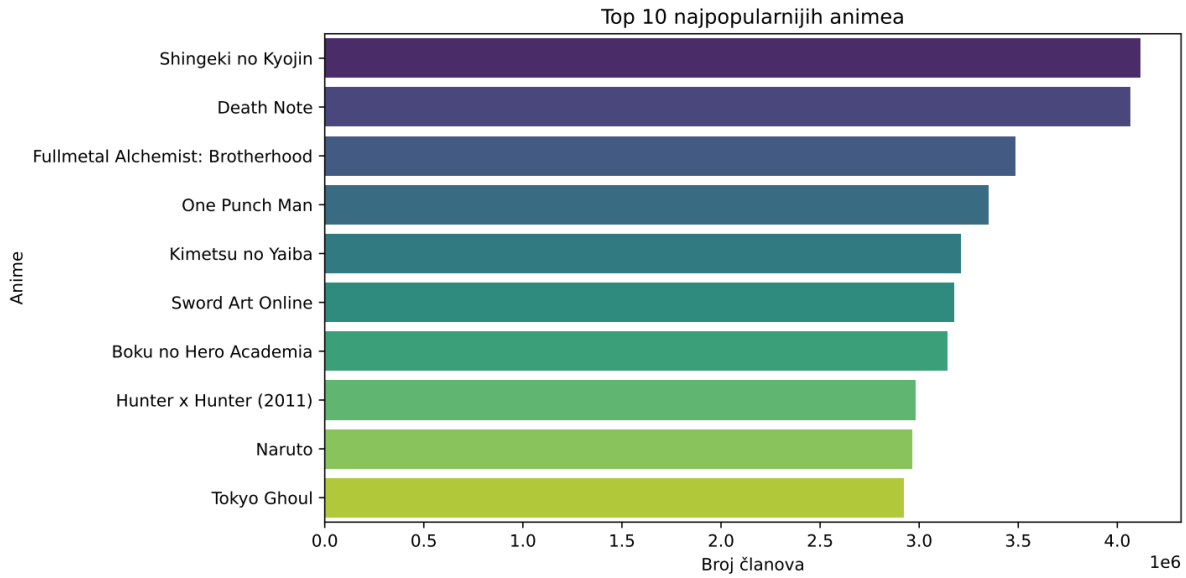
6.5. Prosječna ocjena po žanru



Slika prikazuje prosječne ocjene pojedinačnih anime žanrova, prikazane silazno.

Award Winning žanr ima najvišu prosječnu ocjenu, oko 7.4, što smo mogli i pretpostaviti pošto su to anime djela koja su nagrađivana i poznata po svojoj kvaliteti. Nakon toga ide Suspense i Mystery koji također imaju ocjenu 7 i više, iz toga zaključujemo da anime gledatelji vole misteriju i napete sadržaje. Drama, Romance i Supernatural također su blizu prosječne ocjene 7, a na samom dnu se nalazi Avan Garde i Erotica što smo mogli zaključiti iz broja animea po žanru gdje mi je isto Erotica bila na zadnjem mjestu odnosno najmanje postoji Erotica anime filmova i/ili serija. Ovaj graf pokazuje da anime gledatelji preferiraju emocionalne i dramatične serije i filmove, prije erotike i anime-a koji izlaze iz tradicionalnog crtanja i prikazivanja (avant garde).

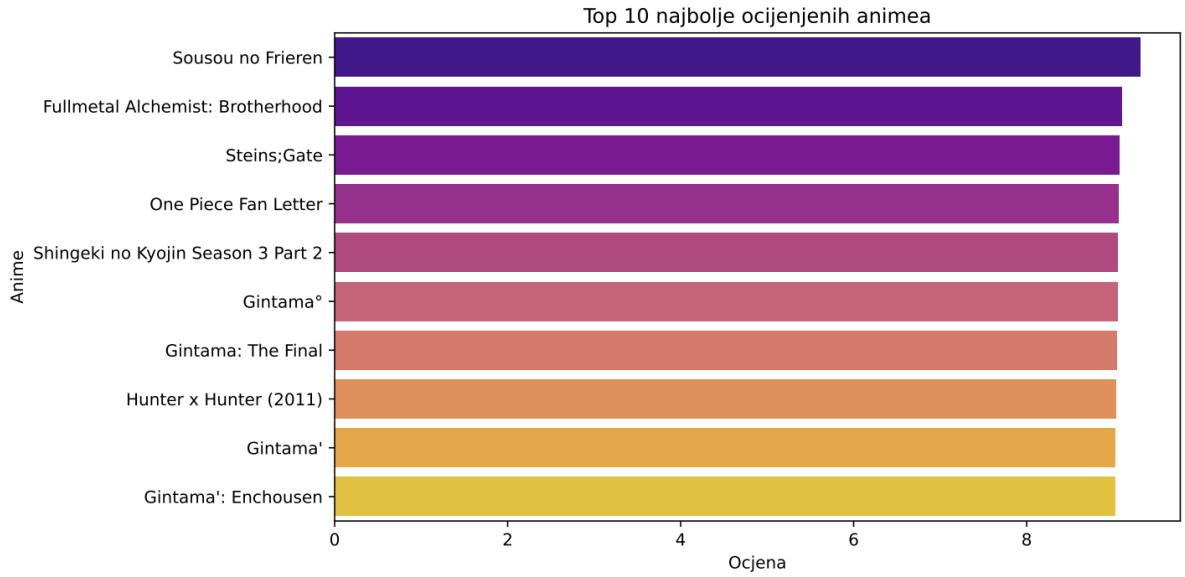
6.6 Top 10 najpopularnijih anime-a



Slika prikazuje deset najpopularnijih anime filmova/serija

Shingeki no Kyojin odnosno Attack on Titan sadrži preko 4 milijuna članova , što potvrđuje da je Attack on Titan globalno najbolji anime. Death Note stoji na drugom mjestu s nešto malo manje od 4 milijuna članova, što opet pokazuje da gledatelji vole psihološke i misteriozne anime serije. Na trećem mjestu se nalazi Fullmetal Alchemist: Brotherhood i smatra se jednim od najboljih anime-a svih vremena sa oko 3.5 milijuna članova. Ostali anime-i u grafu “Top 10 najpopularnijih anime” su One Punch Man, Naruto, KImetsu no Yaiba.. Ova analiza kao i ostale prikazuje kako anime gledatelji vole misteriozne i psihološke anime serije i filmove, te prikazuje kako su zapravo anime serije/filmovi globalno poznati.

6.7. Top 10 ocjenjenih anime-a



Slika prikazuje prvih deset najbolje ocjenjenih anime serija/filmova

Zaključak

Nakon što sam provela ovu analizu, mogu zaključiti da je anime industrija kvalitetna i stabilna jer većina animea dobiva dobre ocjene, standard anime serija su kratke anime serije od 12-26 epizode, gdje preko 14 000 anime-a ima manje od 100 epizoda. Gledatelji više vole komediju, akciju i fantaziju gdje se stalno nešto događa, a manje vole erotiku i romantične anime- serije i/ili filmove. Također iz ove analize mogu zaključiti da su anime- serije na globalnoj popularnosti jer najpopularnija anime serija “Attack on Titan” ima preko 4 milijuna gledatelja. Iako je “Attack on Titan” najpopularnija serija trenutno prvo mjesto na ljestvici sa ocjenjenih 9.31 stoji SouSou no Frieren anime i proglašen je za najbolji anime 2024.

Zaključak

U ovom poglavlju pišem zaključak cijele analize “Top 15 000 ranked anime” . Napravila sam osnovnu analizu ovog skupa podaaka, EER dijagram, koristeći PySpark napunila sam sve tablice, odradila sam OLAP operaciju i vizualizaciju. Koristila sam Python sa bibliotekama Pandas i SQLAlchemy koji su mi služili za detaljnu analizu podataka, Spark sam koristila za distribuiranu obradu anime zapisa, odnosno pomoću njega sam ubacila 14 965 zapisa u tablice. Za ovaj projekt napravila sam i ETL postupak koji mi je služio za pretvaranje CSV datoteke u skladište. Skladište sadrži pet dimenzijskih tablica kao i jednu fact tablicu koja mi služi za spajanje stranih ključeva kao i realiziranje join operacija među tih pet dimenzija. Kvalitetu podataka provjerila sam pomogu MySQL select funkcija gdje sam provjerila sadrži li svaki anime svoj anime_id jedinstveni ključ, i jesu li dimenzije spojne sa stranim ključevima. Napravila sam i star shemu koja ima many-to-many odnose preko bridge tablica. Analizirajući podatke mogu zaključiti da je komedija najpoznatiji žanr u anime-ima sa više od 4500 naslova, ocjene su srednje sa najmanjom ocjenom 5.57, a najvišom ocjenom 9.31, prosječna ocjena koju su gledatelji dali je 6.67. Anime serije su kratke, sa između 12-26 epizoda, osim One Piece i Naruto. Attack on Titan je najpoznatiji anime, te je globalno gledan i ima preko 4 milijuna gledatelja. Sve ove činjenice uspjela sam vidjeti pomoću OLAP funkcionalnosti, a to su slice, dice, roll up, drill down, OLAP funkcionalnosti napravila sam pomoću Python-a koji mi je omogućio višedimenzionalnu analizu koja pomaže kod strateškog planiranja u industriji.

Ovaj rad pokazuje da korištenje naprednih metoda za obradu podataka može pomoći kod odabira investitora, korisničkih potreba te pruža detaljan uvid u tržišne trendove što je dobro kod donošenja poslovnih odluka, novih inovacija ili razvoja.

Literatura

Edureka. (2024). Python Seaborn Tutorial | Data Visualization Using Seaborn.

Anime Recommendation System Data Pipeline

Anime DataDive - A Data Driven Exploration of Anime

Tutorialspoint. ETL in Action: Real-world Examples of Extract, Transform, Load.