Data 225 - DB Systems for Analytics
Lab 1

GitHub Link - https://github.com/Parag000/Data-225-Lab-1

1. Problem Statement:

The proposed application system is a movie catalog system aimed at providing users with information about movie-related data, including details such as movie titles, release dates, ratings, links, genres, cast, crew, and production companies.

Limitation:

- Data Format: A portion of the TMDB dataset is sourced from the TMDB website via an API, provided in JSON format. However, this format poses a challenge as it cannot be integrated into MySQL Workbench for further processing and analysis.

- Usage of No Cloud Services Like MySQL: For testing, data has been loaded into MySQL Workbench. However, MySQL's constraints in resource management and scalability may hinder its ability to accommodate growing data and user demands. To address this, we will migrate the data to AWS after testing on MySQL, ensuring scalability and reliability.

A database is crucial for the system as it offers organized storage and management of movie-related data. It facilitates efficient data retrieval through complex queries, enabling users to quickly access specific information.

Migrating the TMDB database to Amazon Web Services (AWS) is crucial for scalability, reliability, and performance. AWS offers scalable solutions, ensuring the system can handle growing data volumes and user demands without sacrificing performance.

2. Solution Requirements:

- Data Migration to AWS: Performance, scalability, and dependability require an effective data migration from MySQL to Amazon Web Services (AWS).
- Scalability: The system must be able to accommodate expanding user needs and data quantities without compromising performance.
- Efficient Data Retrieval: Users should be able to quickly obtain specific information by utilizing sophisticated queries to retrieve movie-related data.

System Functionalities:

- Data Retrieval: Through searches, users can obtain movie details based on a variety of characteristics, including the title, genre, cast, crew, and production firm.
- Security Measures: Security procedures are adopted by granting specific type of access privileges to various users among the database

System Limitations:

- Additional Cost of AWS: The scalability and dependability of the system rely on AWS, which could result in additional costs and continuing maintenance.

User Interaction with the System:

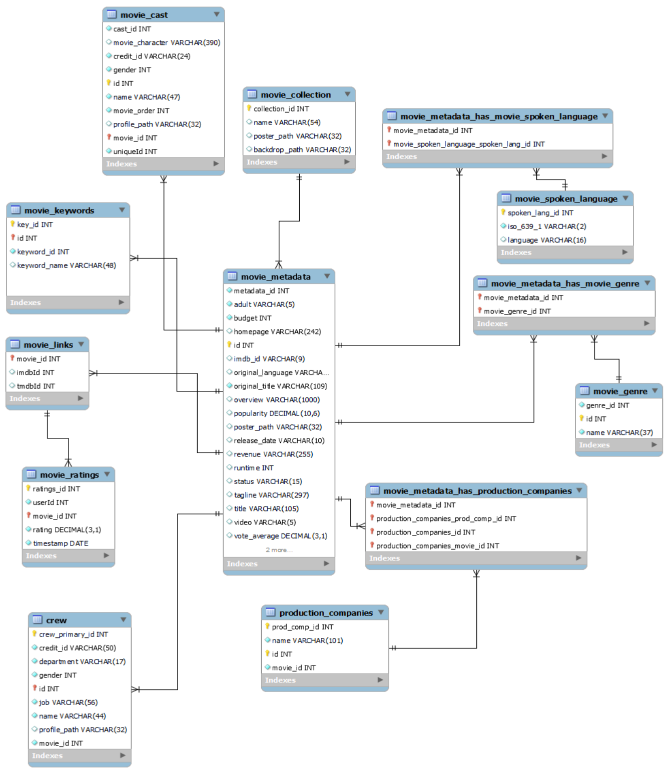
- Searching & Browsing: Users have the option to look up certain titles, genres, cast members, crew members, or production firms in a collection of movies.
- Viewing Movie Details: A user can view a movie's synopsis, release date, ratings, cast, crew, and associated metadata, among other extensive details.
- Rating and Reviewing: Users can be able to rate and review films, adding to the body of feedback from the community.

3. Conceptual Database Design:

Database Requirements:

The system needs a database to manage movie data effectively, including various data types for movie titles, genres, cast details, and ratings. Security, reliability, data integrity, performance, and scalability are key considerations for uninterrupted service and user protection.

ER Diagram:



Normalization Process:

In Phase 1, our initial task was to analyze the structure of the data and identify columns that could be separated into distinct tables. This involved examining the dataset to identify entities and relationships between them. For example, in the context of a movie database, we recognized that information about cast members, crew members, genres, languages, production companies, and countries could be logically separated into individual tables from their respective parent files.

In Phase 2, we utilized Python for preprocessing the data and converting it from JSON format into a normalized form suitable for database storage. We had to make sure the foreign keys referenced from movie_metadata.csv, and credits.csv were mapped correctly with their respective table as this is used to identify the relationships in the database

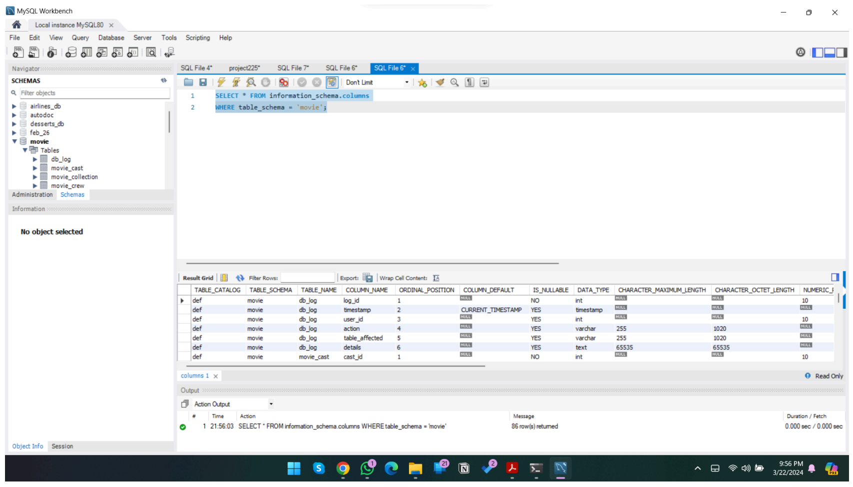
Phase 3 - This step involved writing the transformed data from Python into CSV files using appropriate libraries. Each CSV file represented a distinct entity in the database schema.

To convert the data to the Third Normal Form (3NF), we followed the 3 phases, and each entity was then separated into its own table to eliminate redundancy and ensure data integrity. Relationships between these tables were established using foreign keys, linking related information, and maintaining referential integrity. Finally, we ensured that each table satisfied the requirements of 3NF by eliminating transitive dependencies and ensuring that each non-key attribute was functionally dependent on the primary key.

4. Functional Analysis:

- The database interactions for functional components are provided in the SQL Queries file which is attached to the GitHub link.
- Link to all the SQL questions, code, and output screenshots
  https://github.com/Parag000/Data-225-Lab-1/blob/main/SQL%20Queries.pdf
  Or
  https://docs.google.com/document/d/1vlc8NkfEaZro40cw8Jt2p0ubJPs-gNXyASY-HWgqp3Y/edit?usp=sharing

- Table Structure:



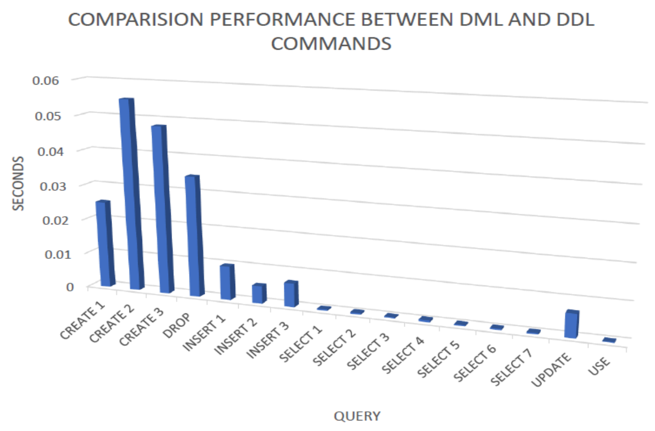
```sql
SELECT * FROM information_schema.columns
WHERE table_schema = 'movie';
```

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_NULLABLE	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH	CHARACTER_OCTET_LENGTH	NUMERIC_I
def	movie	db_log	log_id	1	NULL	NO	int	NULL	NULL	10
def	movie	db_log	timestamp	2	CURRENT_TIMESTAMP	YES	timestamp	NULL	NULL	NULL
def	movie	db_log	user_id	3	NULL	YES	int	NULL	NULL	10
def	movie	db_log	action	4	NULL	YES	varchar	255	1020	NULL
def	movie	db_log	table_affected	5	NULL	YES	varchar	255	1020	NULL
def	movie	db_log	details	6	NULL	YES	text	65535	65535	NULL
def	movie	movie_cast	cast_id	1	NULL	NO	int	NULL	NULL	10

```
#  Time      Action                                                        Message             Duration / Fetch
1  21:56:03  SELECT * FROM information_schema.columns WHERE table_schema = 'movie'  86 row(s) returned  0.000 sec / 0.000 sec
```

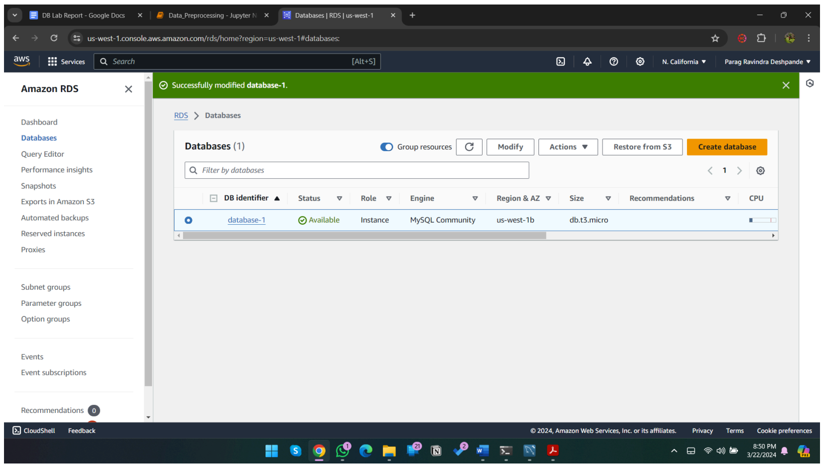
- Some of the concepts we have addressed are
  A. Created 3 users and granted various types of access privileges to each of the users.
  B. Generated 10 SQL queries that make use of various concepts such as window functions, sub-querying, aggregation function with group by and having clause, and other basic operations.
  C. Generated triggers and stored procedures when a user performs insert and update operations.
  D. Created tables for logging movie ratings, and movie links when users performs update or insert operation

- SQL Performance:
  - Not used SELECT * as it increases querying runtime. Instead the * is replaced with column names in the queries that are generated
  - We have maximized the usage of INNER JOIN as it eliminates the mismatch rows
  - Used WHERE and GROUP BY statement in queries to filter out unnecessary information
  - We have used aliases on queries that perform aggregation or any type of mathematical operation
  - Performance measures for DDL commands and DML commands for this database have been plotted using a pivot table using the SHOW PROFILES command.



5. Database Migration to AWS:

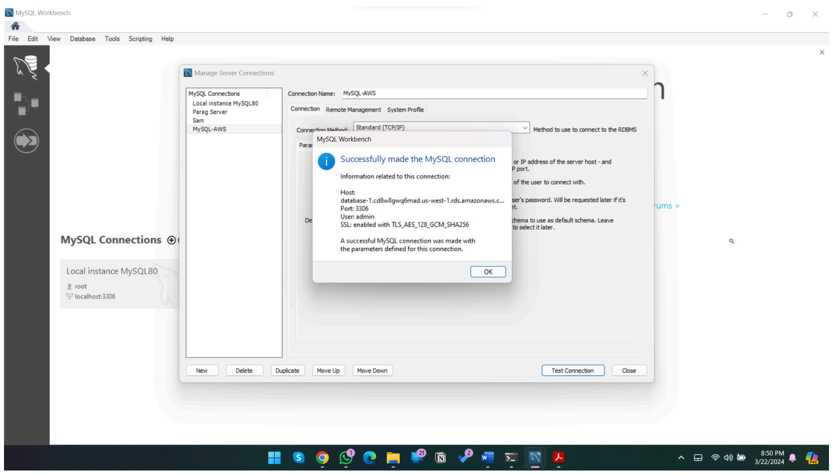
AWS Database creation



Successfully modified **database-1**.

Databases (1)

DB identifier	Status	Role	Engine	Region & AZ	Size	Recommendations	CPU
database-1	Available	Instance	MySQL Community	us-west-1b	db.t3.micro		


MySQL Workbench
File Edit View Database Tools Scripting Help
Manage Server Connections
MySQL Connections
Local instance MySQL80
Parag Server
Sam
MySQL-AWS
Connection Remote Management System Profile
Method to use to connect to the RDBMS
New Delete Duplicate Move Up Move Down Test Connection Close
MySQL Connections
Local instance MySQL80
root
localhost:3306

Connection Name: MySQL-AWS

Connection Method: Standard (TCP/IP)

MySQL Workbench

Successfully made the MySQL connection

Information related to this connection:

Host:
database-1.cd8w8gwq6mad.us-west-1.rds.amazonaws.c...
Port: 3306
User: admin
SSL: enabled with TLS_AES_128_GCM_SHA256

A successful MySQL connection was made with
the parameters defined for this connection.

OK

:

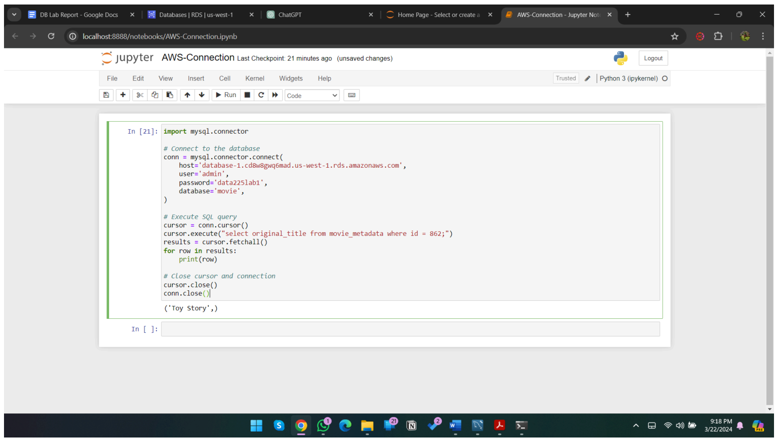
Process:

We have set up an AWS database named "database-1" and created a corresponding security group called "rds-python." This security group is configured to reject inbound connections by default, ensuring a secure environment.

To access the database, we utilize the provided Endpoint, Username, and Password credentials. Leveraging MySQL Workbench, we establish connectivity to the database instance by specifying the hostname and username of the AWS database endpoint.

This configuration enables us to successfully connect to the MySQL database, facilitating efficient management and querying of data within the AWS environment.

Connectivity to AWS Using Python


DB Lab Report - Google Docs
Databases | RDS | us-west-1
ChatGPT
Home Page - Select or create a
AWS-Connection - Jupyter Not
localhost:8888/notebooks/AWS-Connection.ipynb
jupyter
AWS-Connection Last Checkpoint: 21 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [21]: import mysql.connector

         # Connect to the database
         conn = mysql.connector.connect(
             host='database-1.cd8w8gwq6mad.us-west-1.rds.amazonaws.com',
             user='admin',
             password='data225lab1',
             database='movie',
         )

         # Execute SQL query
         cursor = conn.cursor()
         cursor.execute("select original_title from movie_metadata where id = 862;")
         results = cursor.fetchall()
         for row in results:
             print(row)

         # Close cursor and connection
         cursor.close()
         conn.close()

         ('Toy Story',)

In [ ]:
```

Code Snippet:

```
import mysql.connector

# Connect to the database
conn = mysql.connector.connect(
    host='database-1.cd8w8gwq6mad.us-west-1.rds.amazonaws.com',
    user='admin',
    password='data225lab1',
    database='movie',
)
cursor = conn.cursor()
cursor.execute("select original_title from movie_metadata where id = 862;")
results = cursor.fetchall()
for row in results:
    print(row)
cursor.close()
conn.close()
```