LAB 1 - SJSU MOVIE DATABASE - REPORT
DATA 225 – DB SYSTEMS FOR ANALYTICS

Submitted by : Gayathri Sundareshwar, Keerthana Gopikrishnan and Deepasha Jenamani

25th March 2022

SJSU MOVIE DATABASE (SMD)

PROBLEM STATEMENT

A well-known streaming service wishes to provide better experience for its customer on their online platform. As they are currently dealing with thousands of movies and customers. Before implementing the changes to platform, they want to conduct a study to identify areas for improvement and obtain an analysis based on demographics, traffic, customer interest, and other factors that can help their business. However, this task is challenging because having to document watch history of all the customers and suggest movies based on similar liking of customers.

The management would like to bring in features such as movie recommendation for the customer. As it will provide better interaction with customer which will give them a unique experience different from all the similar platforms out there. And, when a business wants to expand into new markets or partner with other industries, it is critical to keep the information up to date and efficient. To achieve this goal, a system for documenting and storing resources is required. As a result, a database management system solution would be extremely useful to build this.

BUSINESS REQUIREMENT

The business requirements must be identified in order to build an effective model. The following are the company requirements:

1. Keeping track of each movie and its corresponding information, tagging it to a unique ID field.
2. Keeping track of every movie's title, genre, description, and other pertinent information.
3. Preserving customer details.
4. Keeping an automated monthly billing system in place for all the valid customers.
5. Tracking and handling customer complaints.
6. Maintaining a record of each user's activities such as their watch history.
7. Visualizing data based on factors such as user activities, demographics, and customer complaints.
8. Analyzing and comprehending profit factors.
9. Generating monthly subscription data and movies watched.

SPECIFICATIONS

Software, Packages and Libraries used in this implementation are:

1. AWS RDS Connection with MYSQL databases ,
2. Database - MYSQL Workbench 8.0 CE
3. Jupyter Notebook used for connectivity to Python.
4. Matplotlib and seaborn used for visualizations.

SOLUTION REQUIREMENT

The table below describes the business and solution requirements in greater detail by categorizing them based on the type of users involved in this business.

User	Requirements
Admin	• Has full privilege over the entire database, including creation of new tables and relations or deletion of existing ones as well as adding new users. • Access to movie-related information • Customer and billing information can be accessed • Access to customer activities, such as movies watched, as well as customer complaints • Employee data accessibility • Add, delete, or update any of the above-mentioned information, such as customer details, movie information, customer complaints, or employee information. • Can grant and revoke privileges of other existing users.

Managerial staff	• Access to information about movies • Access to customer information, invoice information, and customer activities • Customer complaints are available for inspection. • Access to employee data • Add any of the information listed above, such as customer details, movie information, customer complaints, or employee information.
Employee	• Has privilege to only view the data, cannot make any changes to existing data or add new ones. • Access to movie-related information • Customer information, invoice information, and customer activities are all accessible. • Customer complaints can be accessed. • Employee data access

Table 1. Database requirements

While designing this model, certain business rules must be followed, such as only one customer owning an account that can be shared by a maximum of five family members. Customers must sign up for a monthly subscription plan of $19.95. A subscription can only be owned by one person.

DATABASE ANALYSIS AND DESIGN

Conceptual Model

Figure 1 depicts the conceptual model of the SJSU movie model. The conceptual model depicts the entities that have been identified as well as the relationships that exist between them.

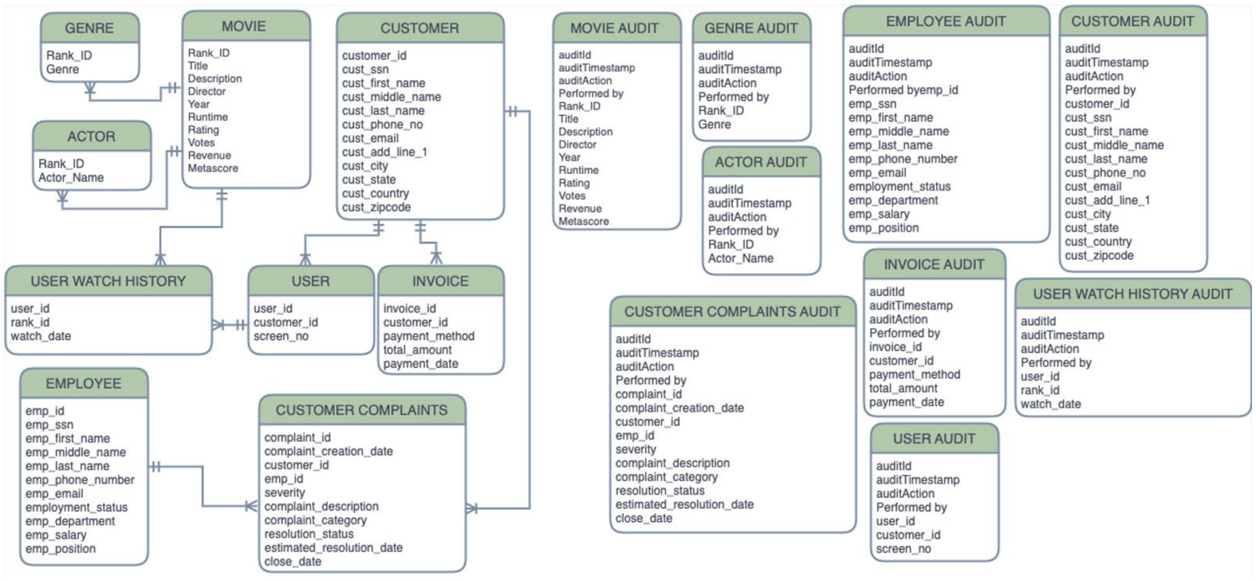


Figure 1. Conceptual Model

Nine primary entities and nine standalone entities have been identified for the model.
The nine primary entities identified are :

1. Customer Details,
2. Customer Complaints,
3. Movie Details,
4. Actor Details,
5. Genre Details,
6. Invoice Details,
7. User Details,
8. User watch history,
9. Employee Details.

The Standalone entities are nothing but the audit tables that had to be created for each of these primary entities. This will help track the changes made to the primary entities.

The first is customer details, which stores customer information such as name, SSN, and address. The second is customer complaints, which stores data relating to any complaints raised by a customer, such as account, video, or audio quality issues. A customer may lodge one or more complaints. It keeps track of each ticket's severity level, creation date, estimated close date, and additional close date if the ticket has been resolved. The third one is employees, which collects information about employees. A trainee, junior or senior support associate, tech support lead, or manager are all instances of employees. Each employee can work on a single or multiple open complaints. Another type is a movie entity, which contains all the information about a movie, such as the title, description, director, and revenue. Each film can feature a variety of actors and genres. Details about actors and genres can be stored in separate entities, namely actor and genre.

Each customer's payment information is stored in the invoice entity. The payment method, payment date, total amount, and corresponding customer ID are all included in each invoice entity instance. The user entity stores the user's information as well as the screen number. One customer may have multiple users, but the number should be limited to five to meet business requirements. A watch history entity is identified to track the user's watch history, which stores the movie rank id, user id, and watch date. In the user watch history entity, a user and a movie can have one or more instances.

The audit features keep track of the user who performs the action, as well as the time stamp and the type of activities performed. More information about the audit activities is provided in the physical model section.

Logical Model

The logical model of the SJSU movie model is depicted in Figure 2. This model illustrates how a unique primary key can be used to identify each entity and how a relationship between two entities can be established using a foreign key.

Each customer can be identified uniquely using the customer ID. A unique employee ID can be used to identify an employee. And the complaint id will be a primary key for the customer complaints entity, and it will be linked to the employee and customer entities via the employee and customer ids, respectively. The invoice entity has a primary identifier, invoice ID, and it is linked to the customer entity via a foreign key, customer ID. Each user has a unique identifier, the user ID, which is connected to the customer entity through a foreign key relationship. The primary identifier for a movie entity is its rank ID. The genre and actor entities are associated to the movie entity through a foreign key relationship based on rank ID. The user watch history is coupled to movie and user entities via a foreign key relationship of rank ID and user ID. A unique audit ID identifies each audit instance. Furthermore, the audit time stamp serves as an additional index for a faster operation on the entity.

Normalization:

Since the movie data table contains columns with multi valued attributes, it is always advisable to normalize it before performing any operations on it. This will help in data cleansing.

Hence, we have Normalized the Movie data table by creating two other tables namely, Actor and Genre. The multi valued attributes are normalized and split into separate rows with respective Rank ID.

The rest of the tables created are also fully normalized.

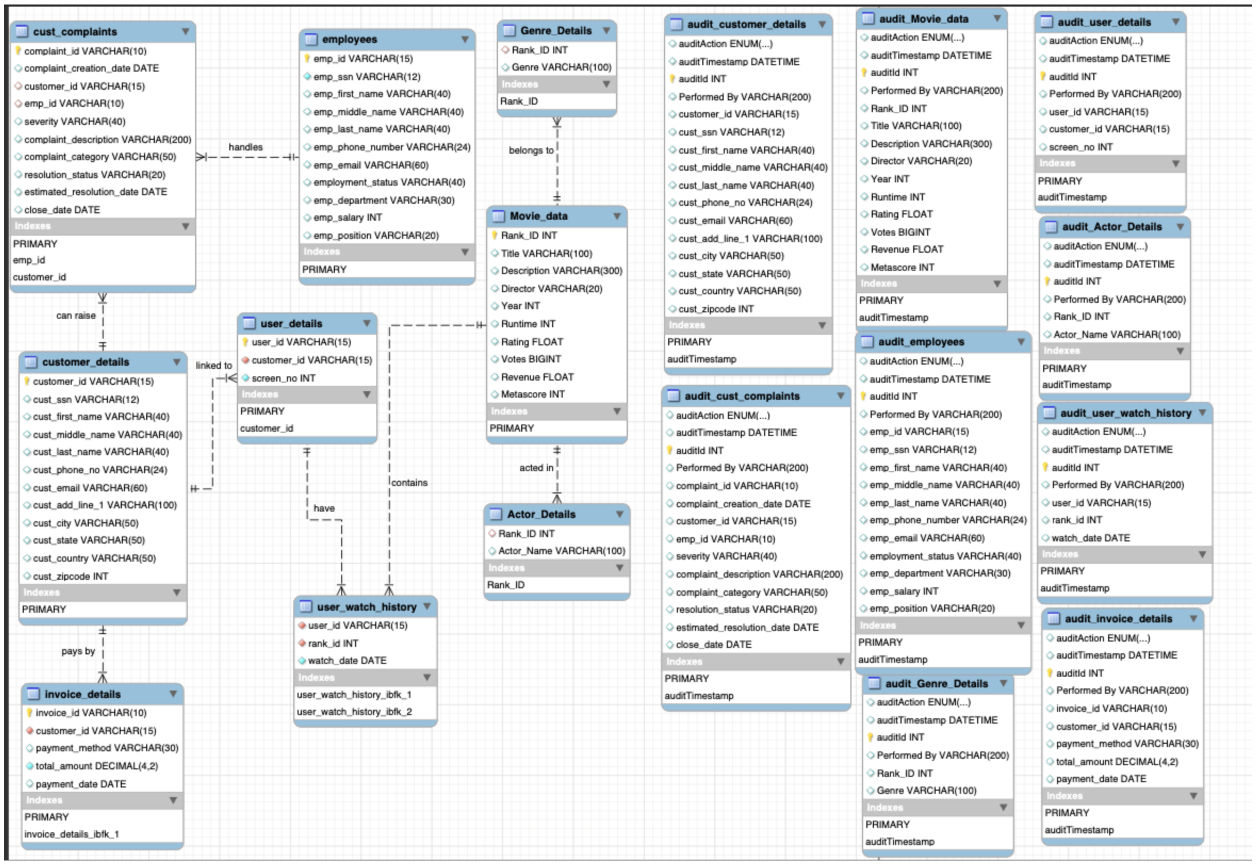
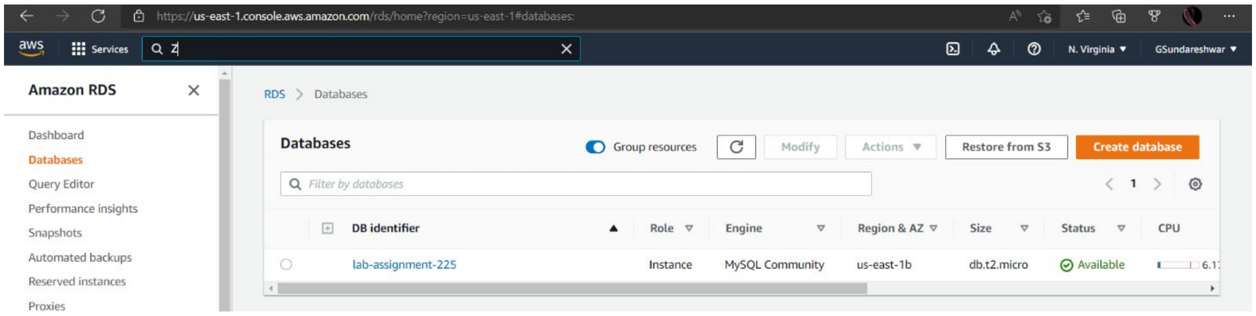


Figure 2. Logical Model

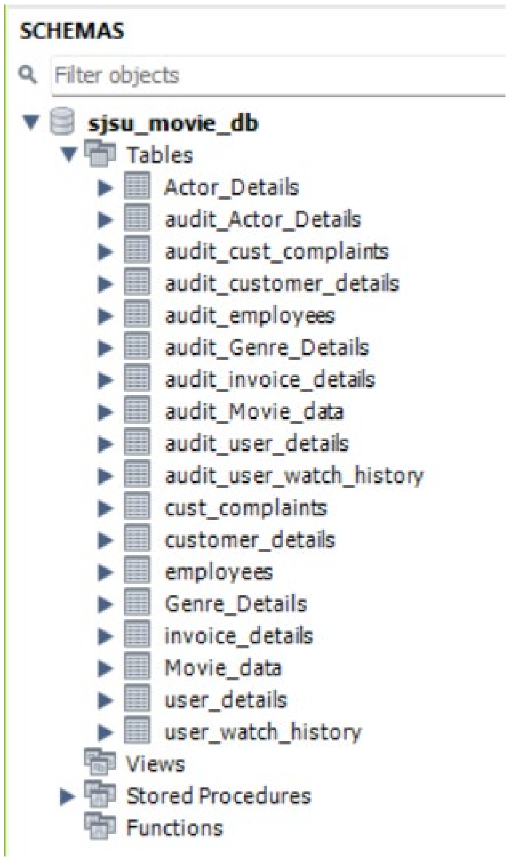
Physical Model

The logical model shown above serves as the foundation for the physical model. Using Amazon Relational Database Service, a database called sjsu_movie_db is created and deployed in the cloud (RDS). Figures 3 and 4 illustrate the database connection in AWS and the database creation in MySQL workbench, respectively. The database contains nine primary tables and audit tables for each of the tables that correspond to the logical model's primary entities. Each table has been designed with business rules in mind.



DB identifier	Role	Engine	Region & AZ	Size	Status	CPU
lab-assignment-225	Instance	MySQL Community	us-east-1b	db.t2.micro	Available	6.1

Figure 3. AWS and MySQL connection


SCHEMAS

Filter objects

- sjsu_movie_db
  - Tables
    - Actor_Details
    - audit_Actor_Details
    - audit_cust_complaints
    - audit_customer_details
    - audit_employees
    - audit_Genre_Details
    - audit_invoice_details
    - audit_Movie_data
    - audit_user_details
    - audit_user_watch_history
    - cust_complaints
    - customer_details
    - employees
    - Genre_Details
    - invoice_details
    - Movie_data
    - user_details
    - user_watch_history
  - Views
  - Stored Procedures
  - Functions

Figure 4. SJSU movie database and tables creation in MySQL workbench

The Actors_Details(Table 2) for each movie is stored in the actor table along with a rank id and actor name. And Genre_details(Table 3) consist of rank id and gerne each movie. Movie_data (Table 4) table contains movie-related information, with the rank id column serving as the primary key because we can identify each movie by its rank id. To keep customer records, a customer_details (Table 5) table is created, and each customer is identified by a unique customer id. To keep track of customer complaints, a separate cust_complaints (Table 6) table is created, with customer id and employee id establishing a relationship with customer details and employee. A user_detailS (Table 7) table stores information about each user, such as the screen number and the associated customer id. To keep track of user watch activities, the user_watch_history(Table 8) table is created, which has a relationship with the movie details via rank id and with user details via user id. Table invoice_details (Table 9) is maintained to keep all payment information for each customer, and each customer is identified by a unique customer id. Employees is the name given to a table that is created for employees (Table 10). This table contains all the records pertaining to the employees.

Key	**Column**	Datatype
FK	**Rank_ID**	int
	Actor_Name	varchar(100)

Table 2. Actor_Details table components

Key	**Column**	Datatype
FK	**Rank_ID**	int
	Genre	varchar(100)

Table 3. Genre_Details table components

Key	**Column**	Datatype
PK	Rank_ID	int AI
	Title	varchar(100)
	Description	varchar(300)
	Director	varchar(20)
	Year	int
	Runtime	int
	Rating	float
	Votes	bigint
	Revenue	float
	Metascore	int

Table 4. Movie_data table components

Key	**Column**	Datatype
PK	**customer_id**	varchar(15)
	cust_ssn	varchar(12)
	cust_first_name	varchar(40)
	cust_middle_name	varchar(40)
	cust_last_name	varchar(40)
	cust_phone_no	varchar(24)
	cust_email	varchar(60)
	cust_add_line_1	varchar(100)
	cust_city	varchar(50)
	cust_state	varchar(50)
	cust_country	varchar(50)
	cust_zipcode	int

Table 5. Customer_details table components

Key	**Column**	Datatype
PK	**complaint_id**	varchar(10)
	complaint_creation_date	date
FK	**customer_id**	varchar(15)
FK	**emp_id**	varchar(10)
	severity	varchar(40)
	complaint_description	varchar(200)
	complaint_category	varchar(50)
	resolution_status	varchar(20)
	estimated_resolution_date	date
	close_date	date

Table 6. Cust_Complaint details table components

Key	Column	Datatype
PK	**user_id**	varchar(15)
FK	customer_id	varchar(15)
	screen_no	int

Table 7. User_details table components

Key	Column	Datatype
FK	**user_id**	varchar(15)
FK	**rank_id**	int
	watch_date	date

Table 8. User_watch_history table component

Key	Column	Datatype
PK	**invoice_id**	varchar(10)
FK	customer_id	varchar(15)
	payment_method	varchar(30)
	total_amount	decimal(4,2)
	payment_date	date

Table 9. Invoice_details table component

Key	Column	Datatype
PK	**emp_id**	varchar(15) PK
	emp_ssn	varchar(12)
	emp_first_name	varchar(40)
	emp_middle_name	varchar(40)
	emp_last_name	varchar(40)
	emp_phone_number	varchar(24)
	emp_email	varchar(60)
	employment_status	varchar(40)
	emp_department	varchar(30)
	emp_salary	int
	emp_position	varchar(20)

Table 10. Employee_details table components

ACCESS PRIVILAGES

According to the requirements, there are three types of access to protect the data and adhere to data security. The first is admin access, which grants full access to all tables. A user with the admin role can view any details, add records, update them, or delete them as needed. The second type is managerial access, which allows a user to view and add records but not update or delete them. There are only a few grants available for the third type of role, which is employee. A user with this role can only view or access the records.

Table 11 depicts the controlled access created for some of the manager and employee roles.

User	Description
Manager 1	The manager with employee ID '543248497' is granted access to select, update, insert records in the database.
Manager 2	The manager with employee ID '407394573' is granted access to select, update, insert records in the database.
Employee 1	The employee with ID '246319689’ is granted access to select records from the database.
Employee 2	The employee with ID '292240662' is granted access to select records from the database.
Employee 3	The employee with ID '512211915' is granted access to select records from the database.

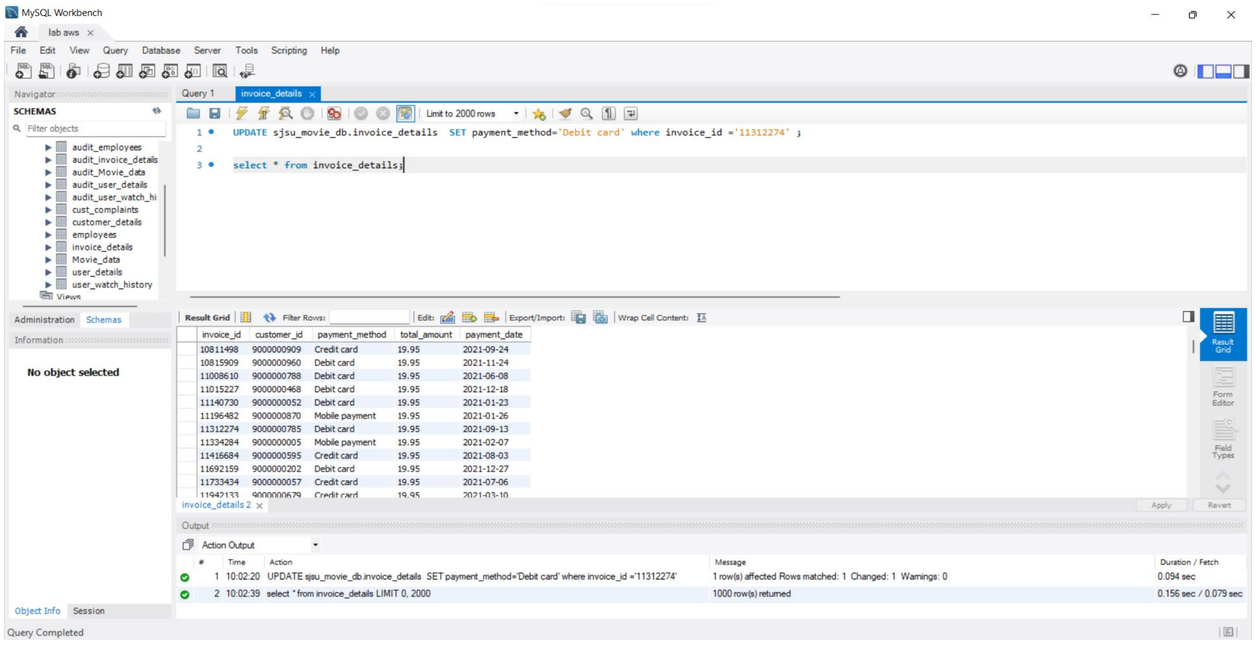
Table 11. Controlled access

The figures 5,6,7 and 8 below demonstrate how the DBMS controlled access works .

ACCESS PROVIDED FOR MANAGER ROLE

Figure 5 shows a manager successfully updating a customer's payment method. Figure 6 shows how a manager can add a new customer.

This shows that the manager has the privileges to select, insert and update the records.

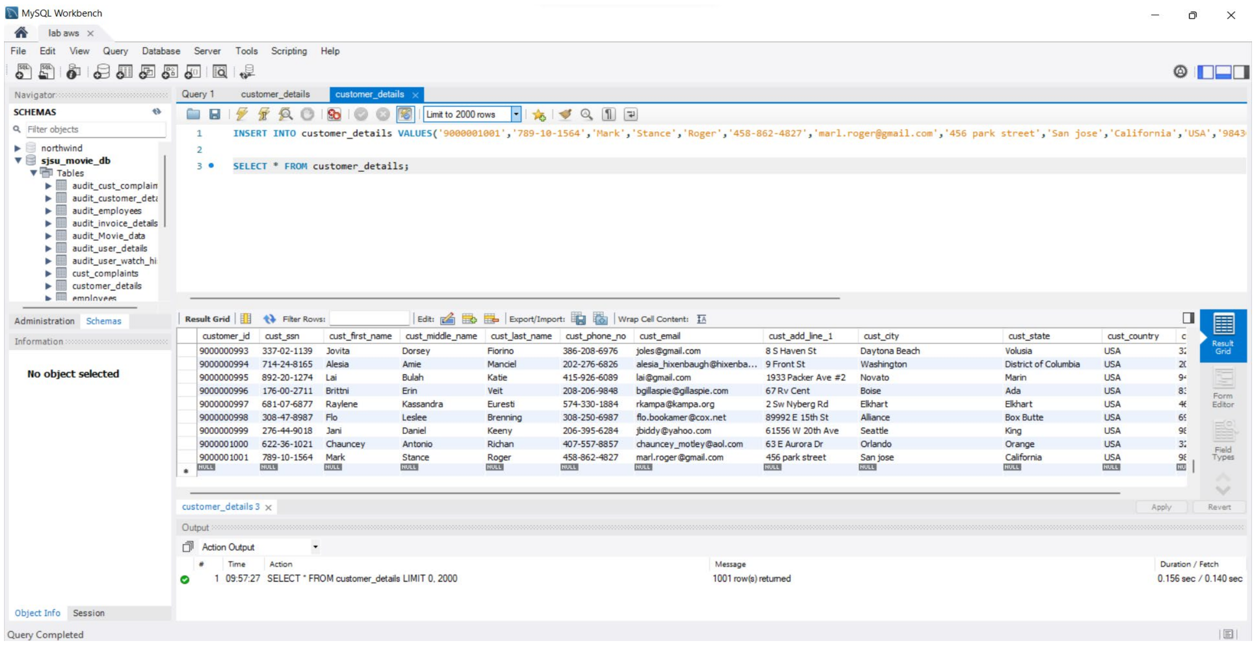


```
UPDATE sjsu_movie_db.invoice_details  SET payment_method='Debit card' where invoice_id ='11312274' ;

select * from invoice_details;
```

invoice_id	customer_id	payment_method	total_amount	payment_date
10811498	9000000909	Credit card	19.95	2021-09-24
10815909	9000000960	Debit card	19.95	2021-11-24
11008610	9000000788	Debit card	19.95	2021-06-08
11015227	9000000468	Debit card	19.95	2021-12-18
11140730	9000000052	Debit card	19.95	2021-01-23
11196482	9000000870	Mobile payment	19.95	2021-01-26
11312274	9000000785	Debit card	19.95	2021-09-13
11334284	9000000005	Mobile payment	19.95	2021-02-07
11416684	9000000595	Credit card	19.95	2021-08-03
11692159	9000000202	Debit card	19.95	2021-12-27
11733434	9000000057	Credit card	19.95	2021-07-06
11942133	9000000679	Credit card	19.95	2021-03-10

```
#  Time      Action                                                                                     Message                                               Duration / Fetch
1  10:02:20  UPDATE sjsu_movie_db.invoice_details  SET payment_method='Debit card' where invoice_id ='11312274'  1 row(s) affected Rows matched: 1  Changed: 1  Warnings: 0  0.094 sec
2  10:02:39  select * from invoice_details LIMIT 0, 2000                                               1000 row(s) returned                                  0.156 sec / 0.079 sec
```

Figure 5. Manager updates invoice details



```sql
INSERT INTO customer_details VALUES('9000001001','789-10-1564','Mark','Stance','Roger','458-862-4827','marl.roger@gmail.com','456 park street','San jose','California','USA','9843

SELECT * FROM customer_details;
```

customer_id	cust_ssn	cust_first_name	cust_middle_name	cust_last_name	cust_phone_no	cust_email	cust_add_line_1	cust_city	cust_state	cust_country
9000000993	337-02-1139	Jovita	Dorsey	Fiorino	386-208-6976	joles@gmail.com	8 S Haven St	Daytona Beach	Volusia	USA
9000000994	714-24-8165	Alesia	Amie	Manciel	202-276-6826	alesia_hixenbaugh@hixenba...	9 Front St	Washington	District of Columbia	USA
9000000995	892-20-1274	Lai	Bulah	Katie	415-926-6089	lai@gmail.com	1933 Packer Ave #2	Novato	Marin	USA
9000000996	176-00-2711	Brittni	Erin	Veit	208-206-9848	bgillaspie@gillaspie.com	67 Rv Cent	Boise	Ada	USA
9000000997	681-07-6877	Raylene	Kassandra	Euresti	574-330-1884	rkampa@kampa.org	2 Sw Nyberg Rd	Elkhart	Elkhart	USA
9000000998	308-47-8987	Flo	Leslee	Brenning	308-250-6987	flo.bookamer@cox.net	89992 E 15th St	Alliance	Box Butte	USA
9000000999	276-44-9018	Jani	Daniel	Keeny	206-395-6284	jbiddy@yahoo.com	61556 W 20th Ave	Seattle	King	USA
9000001000	622-36-1021	Chauncey	Antonio	Richan	407-557-8857	chauncey_motley@aol.com	63 E Aurora Dr	Orlando	Orange	USA
9000001001	789-10-1564	Mark	Stance	Roger	458-862-4827	marl.roger@gmail.com	456 park street	San jose	California	USA
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
#  Time      Action                                          Message                 Duration / Fetch
1  09:57:27  SELECT * FROM customer_details LIMIT 0, 2000   1001 row(s) returned    0.156 sec / 0.140 sec
```

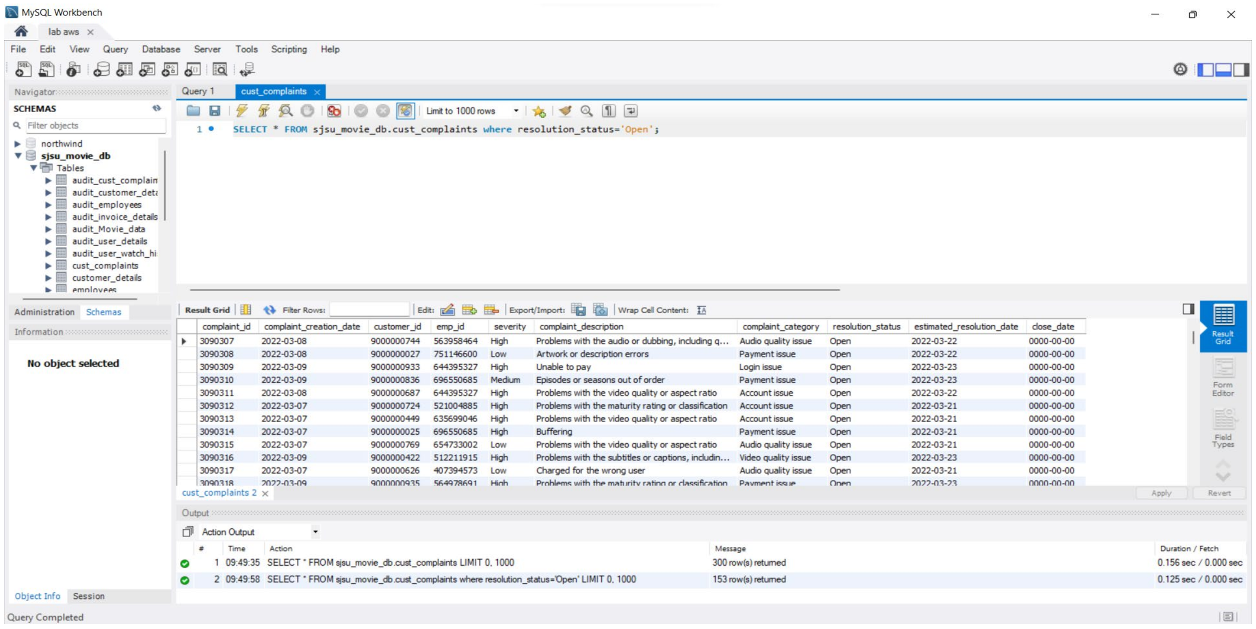
Query Completed

Figure 6. Manager adds a new customer

ACCESS PROVIDED FOR EMPLOYEE ROLE

Figure 7 depicts how an employee can gain access to a customer's complaint. Figure 8 illustrates how access is denied when an employee attempts to insert any record.

This shows that the employee can only view the records, when the employee tries to insert, access is denied.



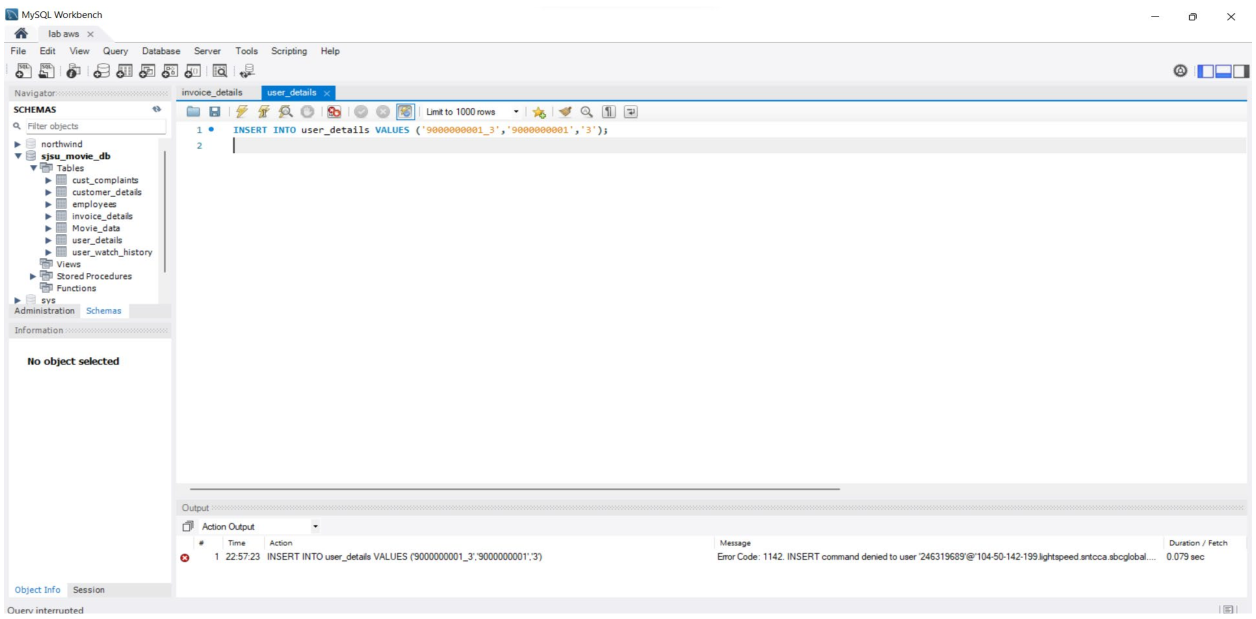
```sql
SELECT * FROM sjsu_movie_db.cust_complaints where resolution_status='Open';
```

complaint_id	complaint_creation_date	customer_id	emp_id	severity	complaint_description	complaint_category	resolution_status	estimated_resolution_date	close_date
3090307	2022-03-08	9000000744	563958464	High	Problems with the audio or dubbing, including q...	Audio quality issue	Open	2022-03-22	0000-00-00
3090308	2022-03-08	9000000027	751146600	Low	Artwork or description errors	Payment issue	Open	2022-03-22	0000-00-00
3090309	2022-03-09	9000000933	644395327	High	Unable to pay	Login issue	Open	2022-03-23	0000-00-00
3090310	2022-03-09	9000000836	696550685	Medium	Episodes or seasons out of order	Payment issue	Open	2022-03-23	0000-00-00
3090311	2022-03-08	9000000687	644395327	High	Problems with the video quality or aspect ratio	Account issue	Open	2022-03-22	0000-00-00
3090312	2022-03-07	9000000724	521004885	High	Problems with the maturity rating or classification	Account issue	Open	2022-03-21	0000-00-00
3090313	2022-03-07	9000000449	635699046	High	Problems with the video quality or aspect ratio	Account issue	Open	2022-03-21	0000-00-00
3090314	2022-03-07	9000000025	696550685	High	Buffering	Payment issue	Open	2022-03-21	0000-00-00
3090315	2022-03-07	9000000769	654733002	Low	Problems with the video quality or aspect ratio	Audio quality issue	Open	2022-03-21	0000-00-00
3090316	2022-03-09	9000000422	512211915	High	Problems with the subtitles or captions, includin...	Video quality issue	Open	2022-03-23	0000-00-00
3090317	2022-03-07	9000000626	407394573	Low	Charged for the wrong user	Audio quality issue	Open	2022-03-21	0000-00-00
3090318	2022-03-09	9000000935	564978691	High	Problems with the maturity rating or classification	Payment issue	Open	2022-03-23	0000-00-00

```
#  Time      Action                                                                            Message                Duration / Fetch
1  09:49:35  SELECT * FROM sjsu_movie_db.cust_complaints LIMIT 0, 1000                         300 row(s) returned    0.156 sec / 0.000 sec
2  09:49:58  SELECT * FROM sjsu_movie_db.cust_complaints where resolution_status='Open' LIMIT 0, 1000   153 row(s) returned    0.125 sec / 0.000 sec
```

Query Completed

Figure 7. Employee accessing customer complaint



```
INSERT INTO user_details VALUES ('9000000001_3','9000000001','3');
```

```
#  Time      Action                                                            Message                                                                                                  Duration / Fetch
1  22:57:23  INSERT INTO user_details VALUES ('9000000001_3','9000000001','3')  Error Code: 1142. INSERT command denied to user '246319689'@'104-50-142-199.lightspeed.sntcca.sbcglobal....  0.079 sec
```

Query interrupted

Figure 8. Employee access denied

AUDIT TABLES

An audit table is maintained to keep track of user activities or any unauthorized access. This table keeps track of all user activities in all tables that have been created. This keeps track of the changes made to each table, such as insertion, modification, or deletion, and records the action, as well as the timestamp and the user who performed the action.

Table 12 depicts the list of audit tables create to monitor the changes made to the primary tables.

No.	Audit Table Name	Description	Execution Logs
1	audit_Movie_data	Monitor changes made to the mentioned tables, such as, inserts, updates and deletes. An entry is created with username, time and what operation has been performed and includes the column values	AuditTables_Triggers_Procedures_QueriesA
2	audit_Actor_Details		
3	audit_Genre_Details		
4	audit_customer_details		
5	audit_cust_complaints		
6	audit_user_details		
7	audit_employees		
8	audit_invoice_Details		
9	audit_user_watch_history		

Table 12. List of Audit tables created

PRE-DEFINED AND UN-DEFINED BUSINESS RULES

There are some rules which were established in the business requirement, and there are others which were not defined explicitly. Some of those are:

- The subscription plan can only be shared among five family members, according to the business rules.
- The subscription plan fee should be a fixed amount of $19.95.
- Any future date should be rejected by the user's watch history.
- If there is an open complaint, the estimated resolution date should be in the future. And if, on the other hand, the ticket is resolved, the estimated resolution date and close date should not be in the future.

To check the data passes through all these rules, triggers were created.

TRIGGERS

Four triggers are created to ensure compliance with these requirements. Trigger before_insert_user is created on the user details table that is triggered before inserting any record into the table to determine if the number of screens is greater than five. Trigger before_insert_invoice is created on the invoice details table that is fired before insertion to ensure that the amount paid is only $19.95.Trigger before_insert_watch_history is created on the user watch history table that runs before any insertion into the table, ensuring that the movie watched date is not in the future. Trigger before_insert_cust_compliant is created on the customer complaints table that is fired prior to insertion and checks the date input constraint for open and closed tickets.

A total of 31 triggers were created , 4 for the pre-defined and undefined business rules. Remaining 27 (3 per table) were created to implement the audit functionality and monitor the changes made to the 9 primary tables.

Sr.No	Trigger	Event	Table	Statement	Timing	Created	Definer
1	Actor_Details_inserts	INSERT	Actor_Details	INSERT INTO `sjsu_movi	AFTER	2022-03-22 02:20:17.77	admin@%
2	Actor_Details_updates	UPDATE	Actor_Details	INSERT INTO `sjsu_movi	AFTER	2022-03-22 02:20:17.97	admin@%
3	Actor_Details_deletes	DELETE	Actor_Details	INSERT INTO `sjsu_movi	AFTER	2022-03-22 02:20:18.15	admin@%
4	Genre_Details_inserts	INSERT	Genre_Details	INSERT INTO `sjsu_movi	AFTER	2022-03-22 03:40:31.40	admin@%
5	Genre_Details_updates	UPDATE	Genre_Details	INSERT INTO `sjsu_movi	AFTER	2022-03-22 03:40:31.59	admin@%
6	Genre_Details_deletes	DELETE	Genre_Details	INSERT INTO `sjsu_movi	AFTER	2022-03-22 03:40:31.78	admin@%
7	Movie_data_inserts	INSERT	Movie_data	INSERT INTO `sjsu_movi	AFTER	2022-03-22 02:18:16.98	admin@%
8	Movie_data_updates	UPDATE	Movie_data	INSERT INTO `sjsu_movi	AFTER	2022-03-22 02:18:17.16	admin@%
9	Movie_data_deletes	DELETE	Movie_data	INSERT INTO `sjsu_movi	AFTER	2022-03-22 02:18:17.34	admin@%
10	before_insert_cust_com	INSERT	cust_complaints	begin declare msg_err va	BEFORE	2022-03-13 02:17:55.89	admin@%
11	cust_complaints_inserts	INSERT	cust_complaints	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:16:39.16	admin@%
12	cust_complaints_update	UPDATE	cust_complaints	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:16:39.34	admin@%
13	cust_complaints_deletes	DELETE	cust_complaints	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:16:39.52	admin@%
14	customer_details_inserts	INSERT	customer_details	INSERT INTO `sjsu_movi	AFTER	2022-03-13 09:57:35.86	admin@%
15	customer_details_updat	UPDATE	customer_details	INSERT INTO `sjsu_movi	AFTER	2022-03-13 09:57:36.04	admin@%
16	customer_details_delete	DELETE	customer_details	INSERT INTO `sjsu_movi	AFTER	2022-03-13 09:57:36.22	admin@%
17	employees_inserts	INSERT	employees	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:40:40.51	admin@%
18	employees_updates	UPDATE	employees	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:40:40.72	admin@%
19	employees_deletes	DELETE	employees	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:40:40.91	admin@%
20	before_insert_invoice	INSERT	invoice_details	begin declare msg_err va	BEFORE	2022-03-09 20:07:12.09	admin@%
21	invoice_details_inserts	INSERT	invoice_details	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:50:49.08	admin@%
22	invoice_details_updates	UPDATE	invoice_details	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:50:49.26	admin@%
23	invoice_details_deletes	DELETE	invoice_details	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:50:49.44	admin@%
24	before_insert_users	INSERT	user_details	begin declare msg_err va	BEFORE	2022-03-09 06:10:02.57	admin@%
25	user_details_inserts	INSERT	user_details	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:27:16.79	admin@%
26	user_details_updates	UPDATE	user_details	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:27:16.97	admin@%
27	user_details_deletes	DELETE	user_details	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:27:17.15	admin@%
28	before_insert_watch_his	INSERT	user_watch_history	begin declare msg_err va	BEFORE	2022-03-09 07:54:50.31	admin@%
29	user_watch_history_inse	INSERT	user_watch_history	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:59:58.05	admin@%
30	user_watch_history_upd	UPDATE	user_watch_history	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:59:58.24	admin@%
31	user_watch_history_dele	DELETE	user_watch_history	INSERT INTO `sjsu_movi	AFTER	2022-03-13 10:59:58.41	admin@%

Table 13. List of Triggers created

SQL QUERIES PERFORMED

Table 14 depicts the list of scenario checks performed to analyze the data

Sr.No	DESCRIPTION	Plot type	Execution Logs
1	Display the number of customers in each country	Bar	Embedded_SQL_Screenshots(1-16).pdf
2	Displaying the number of movies released per year	Pie	
3	Top 5 employees	Bar	
4	Average time taken to close a ticket	Query Result	
5	Top 10 Movies in USA, UK and Canada	Query Result	
6	Average Number of screens used by the customers	Query Result	
7	Total number of complaints per complaint category	Bar	
8	Total number of complaints closed per month	Bar	
9	Total number of payments made per month	Pie	
10	Top 10 customers based on movies watched	Bar	
11	Top 5 directors based on user watch history	Bar	
12	Total Number of complaints based on severity	Pie	
13	Average runtime of movies	Query Result	
14	Total number of complaints registered per month	Pie	
15	Average salary based on designation	Bar	
16	Total no of employees per designation	Query Result	

Table 14: List of queries created to analyze the data

STORED PROCEDURES

Table 15 depicts the list of stored procedures created.

Sr.No	Procedure Name	Description	Execution Logs
1	TopTenMoviesPerYear	When year is given as input, this procedure displays the top 10 movies released that year.	AuditTables_Triggers_Procedures_Queries
2	GetMaximumTrafficInAMonth	When month is given as input, this procedure displays the most watched movies in that specific month	
3	CompResolutionStatus	When resolution date is given as input, this procedure displays the 5 oldest complaints in that status	
4	DetailsOfCustomer	When customer id is given and "personal" or "invoice" is chosen, this procedure displays the respective details for that customer	
5	payment_per_month	This procedure displays the payments made per month	

Table 15. List of stored procedures created

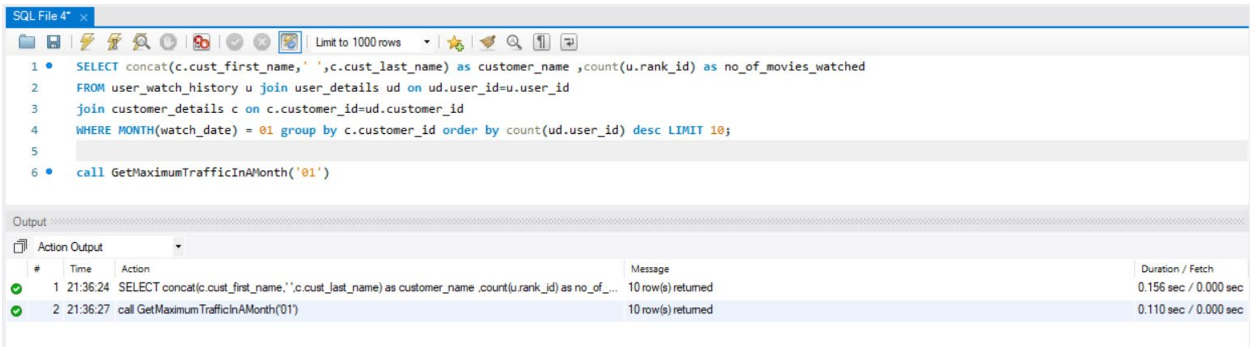
PERFORMANCE MEASUREMENT

Monitoring the performance of our database will help us to preemptively handle the possible problems and bugs that could occur in our application before the release. Aside from helping in fixing the issues, it can also be helpful in finding out which is the optimum way to arrive at a solution. A query can be dealt with in multiple ways, But it is always advisable to take the route which is more efficient and less resource consuming.

We have performed performance monitoring for few sample scenarios and the reason for going with the less time consuming one.

Figure 8 and 9 depicts the execution time taken by 2 query and 2 procedures. Point to be noted that, the procedure contains the same exact query inside it as the one above. We can see that procedure takes lesser time to execute than the query itself.

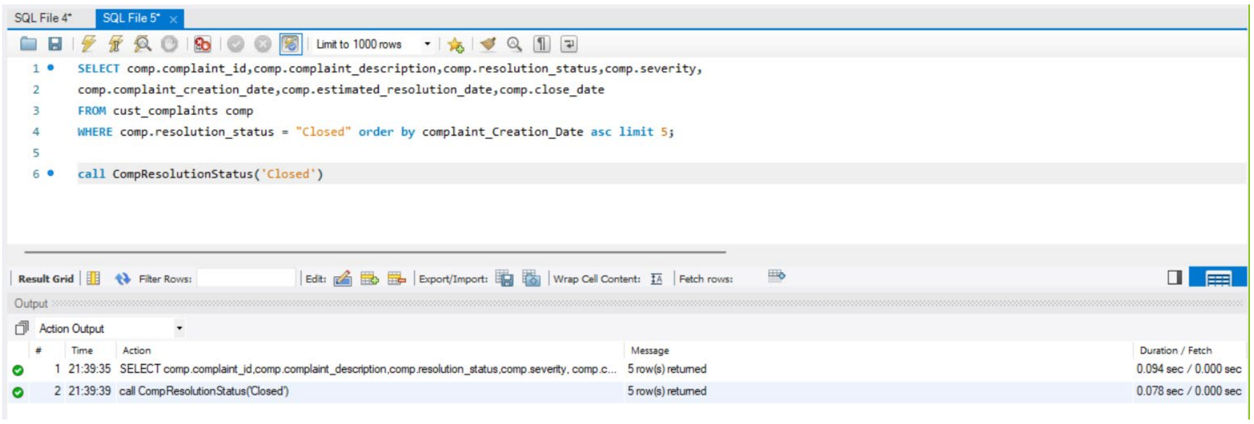
Hence the use of procedures is opted for efficient performance.



```sql
SELECT concat(c.cust_first_name,' ',c.cust_last_name) as customer_name ,count(u.rank_id) as no_of_movies_watched
FROM user_watch_history u join user_details ud on ud.user_id=u.user_id
join customer_details c on c.customer_id=ud.customer_id
WHERE MONTH(watch_date) = 01 group by c.customer_id order by count(ud.user_id) desc LIMIT 10;

call GetMaximumTrafficInAMonth('01')
```

```
#  Time      Action                                                                              Message              Duration / Fetch
1  21:36:24  SELECT concat(c.cust_first_name,' ',c.cust_last_name) as customer_name ,count(u.rank_id) as no_of_...  10 row(s) returned  0.156 sec / 0.000 sec
2  21:36:27  call GetMaximumTrafficInAMonth('01')                                                10 row(s) returned   0.110 sec / 0.000 sec
```

Figure 8. Performance Measurement 1 (procedure 1)



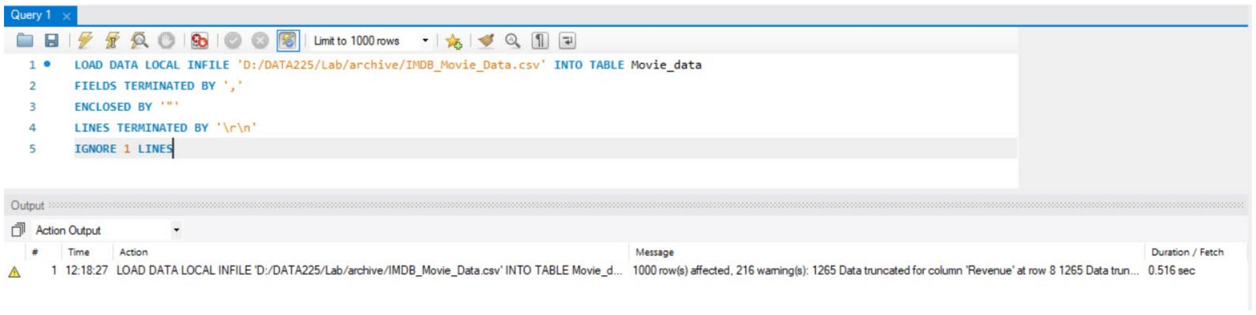
```sql
SELECT comp.complaint_id,comp.complaint_description,comp.resolution_status,comp.severity,
comp.complaint_creation_date,comp.estimated_resolution_date,comp.close_date
FROM cust_complaints comp
WHERE comp.resolution_status = "Closed" order by complaint_Creation_Date asc limit 5;

call CompResolutionStatus('Closed')
```

```
#  Time      Action                                                                              Message             Duration / Fetch
1  21:39:35  SELECT comp.complaint_id,comp.complaint_description,comp.resolution_status,comp.severity, comp.c...  5 row(s) returned  0.094 sec / 0.000 sec
2  21:39:39  call CompResolutionStatus('Closed')                                                 5 row(s) returned   0.078 sec / 0.000 sec
```

Figure 9. Performance Measurement 2 (procedure 2)

TIME TAKEN BY INSERT COMMANDS VS LOAD DATA LOCAL INFILE:

Figure 10 and 11 depicts the difference between the time taken for loading the data into the table using “load data local infile” vs “Insert” commands.
We can see the insert commands are more time consuming than the other. Hence, we have used load data local infile since its less time consuming and more efficient.

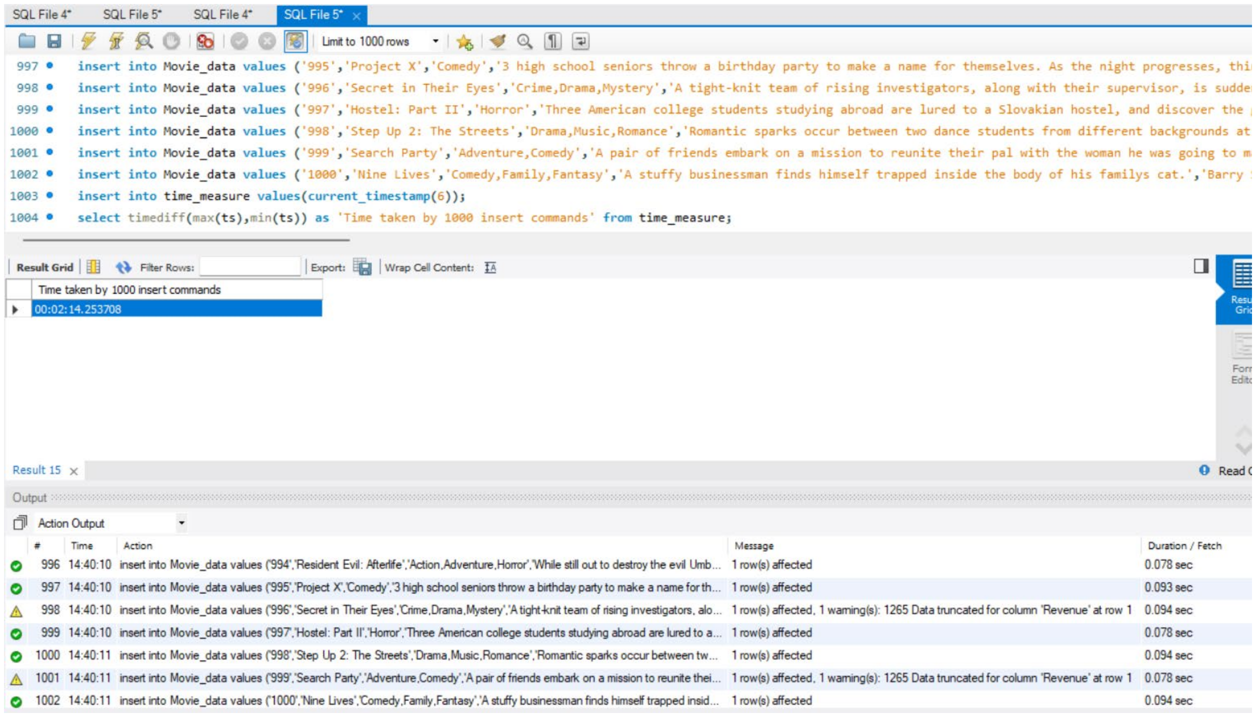


```
LOAD DATA LOCAL INFILE 'D:/DATA225/Lab/archive/IMDB_Movie_Data.csv' INTO TABLE Movie_data
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r\n'
IGNORE 1 LINES
```

```
Action Output
#  Time      Action                                                                                Message                                                                                           Duration / Fetch
1  12:18:27  LOAD DATA LOCAL INFILE 'D:/DATA225/Lab/archive/IMDB_Movie_Data.csv' INTO TABLE Movie_d...  1000 row(s) affected, 216 warning(s): 1265 Data truncated for column 'Revenue' at row 8 1265 Data trun...  0.516 sec
```

Figure 10. Performance Measurement 3 (Load data local infile)

The usage off triggers can also be considered as performance tuning because, it helps eliminate the bad data, thus reducing the time taken to manually cleanse it.



```
insert into Movie_data values ('995','Project X','Comedy','3 high school seniors throw a birthday party to make a name for themselves. As the night progresses, thi
insert into Movie_data values ('996','Secret in Their Eyes','Crime,Drama,Mystery','A tight-knit team of rising investigators, along with their supervisor, is sudde
insert into Movie_data values ('997','Hostel: Part II','Horror','Three American college students studying abroad are lured to a Slovakian hostel, and discover the
insert into Movie_data values ('998','Step Up 2: The Streets','Drama,Music,Romance','Romantic sparks occur between two dance students from different backgrounds at
insert into Movie_data values ('999','Search Party','Adventure,Comedy','A pair of friends embark on a mission to reunite their pal with the woman he was going to m
insert into Movie_data values ('1000','Nine Lives','Comedy,Family,Fantasy','A stuffy businessman finds himself trapped inside the body of his familys cat.','Barry
insert into time_measure values(current_timestamp(6));
select timediff(max(ts),min(ts)) as 'Time taken by 1000 insert commands' from time_measure;
```

```
Time taken by 1000 insert commands
00:02:14.253708
```

```
Action Output
#     Time      Action                                                                                                   Message                                                                          Duration / Fetch
996   14:40:10  insert into Movie_data values ('994','Resident Evil: Afterlife','Action,Adventure,Horror','While still out to destroy the evil Umb...  1 row(s) affected                                                                0.078 sec
997   14:40:10  insert into Movie_data values ('995','Project X','Comedy','3 high school seniors throw a birthday party to make a name for th...  1 row(s) affected                                                                0.093 sec
998   14:40:10  insert into Movie_data values ('996','Secret in Their Eyes','Crime,Drama,Mystery','A tight-knit team of rising investigators, alo...  1 row(s) affected, 1 warning(s): 1265 Data truncated for column 'Revenue' at row 1  0.094 sec
999   14:40:10  insert into Movie_data values ('997','Hostel: Part II','Horror','Three American college students studying abroad are lured to a...  1 row(s) affected                                                                0.078 sec
1000  14:40:11  insert into Movie_data values ('998','Step Up 2: The Streets','Drama,Music,Romance','Romantic sparks occur between tw...  1 row(s) affected                                                                0.094 sec
1001  14:40:11  insert into Movie_data values ('999','Search Party','Adventure,Comedy','A pair of friends embark on a mission to reunite thei...  1 row(s) affected, 1 warning(s): 1265 Data truncated for column 'Revenue' at row 1  0.078 sec
1002  14:40:11  insert into Movie_data values ('1000','Nine Lives','Comedy,Family,Fantasy','A stuffy businessman finds himself trapped insid...  1 row(s) affected                                                                0.094 sec
```

Figure 11. Performance Measurement 4 (Insert commands)

FUNCTIONAL ANALYSIS

The application was designed not only to serve as an archive for the movies, the customer , their watch history and payment methods. But instead, this can help in deriving the knowledge patterns which in turn will provide insights for the business management team to make purchase / management decisions .

The Normalizations performed and the triggers created are helpful in eliminating the redundancy and in the cleansing of the data.

Tracking the user watch history can help us in finding out the wavering interest of people in different time periods, for example, listing the top 10 most watched movies can help in finding the most loved genre during that time period. Tracking this over the years can help the management to decide on future copyright purchases for upcoming movies.

The system can be accessed by a user with appropriate access, such as an administrator, manager, or employee. Administrators have the most access, while employees have the least.

The pictorial representation of data aids in better data analysis and faster decision making. This goal is accomplished using data visualization techniques. Plotting the data derived from the query results as visualizations makes it easier for the management team to understand.

When there is a change in the data, triggers are created to help maintain the data's integrity. The triggers ensure that certain business requirements, such as the number of screens, subscription payment, or any other validations, such as movie watch date, complaint resolution dates, are met. A corresponding audit table for each table is created to track how the data is used and to alert any risks of misuses or breaches.

Monitoring the complaint creation date, estimated closure and closure date will aid in finding out the category of complaints which take more time to resolve than the estimated date. Knowledge attained from this data can be helpful in determining the vertical where employee needs more training on.

LIMITATION

Even though the model is effective in handling the current business scenario and adequately captures all requirements, it does have a few limitations, which are listed below.

1. It does not encrypt a customer's payment information.
2. Any customer's or employee's Social Security Number (SSN), which is sensitive information, is not encrypted and thus not protected from cyberattacks.
3. Very limited payment methods are available currently, more will be added in future as required when new partnership between platforms emerge.
4. The dataset provided for this project has data only from 2006 – 2016, new data can be added.
5. Email notifications for ticket status & payment are not enabled yet in this version.

SCOPE

The plan is to cover the limitations as parts of future scope in each release.

1. Important information such as Employee and Customer SSN, Address and Phone number can be encrypted using field level AES or DES encryptions. As well as using AWS KMS.
2. The application created now is accustomed to only movies , this can be further enhanced with series, documentaries, short films, reality shows etc.
3. Email notifications can be implemented for the complaints , payments. Also, audit tables can be set up in a way that the admin can receive timely audit table updates.
4. Customer input regarding a movie they watched can be recorded as user rating, and recommendations can be formulated using this data.
5. Subscription plans can be enhanced with different tiers each having its own set of functionalities . ex, tier 1 with 1080p video and 5 screens and tier 2 with 4k videos and 7 screens.

REFERENCES

Dataset - https://www.kaggle.com/PromptCloudHQ/imdb-data

PROJECT TEAM

Gayathri Sundareshwar , Keerthana Gopikrishnan, Deepasha Jenamani