Screenex : Database Management System

Screenex: The Revolution

Screenex is an innovative online streaming platform dedicated to offering a 2diverse range of movies for free. Our goal is to provide an exceptional viewing experience by featuring a curated selection of films across three major categories: Bollywood, Hollywood, and Anime. Whether you're in the mood for vibrant Indian cinema, the latest Hollywood blockbusters, or captivating anime series, Screenex is your go-to destination for unlimited entertainment.

Key Features:

1. **Diverse Movie Categories:**

   - **Bollywood: Discover a vibrant collection of Indian cinema, from classics to new hits.**
   - **Hollywood: Watch popular blockbusters and acclaimed films from the heart of the film industry.**
   - **Anime: Dive into unique and imaginative anime series and movies.**

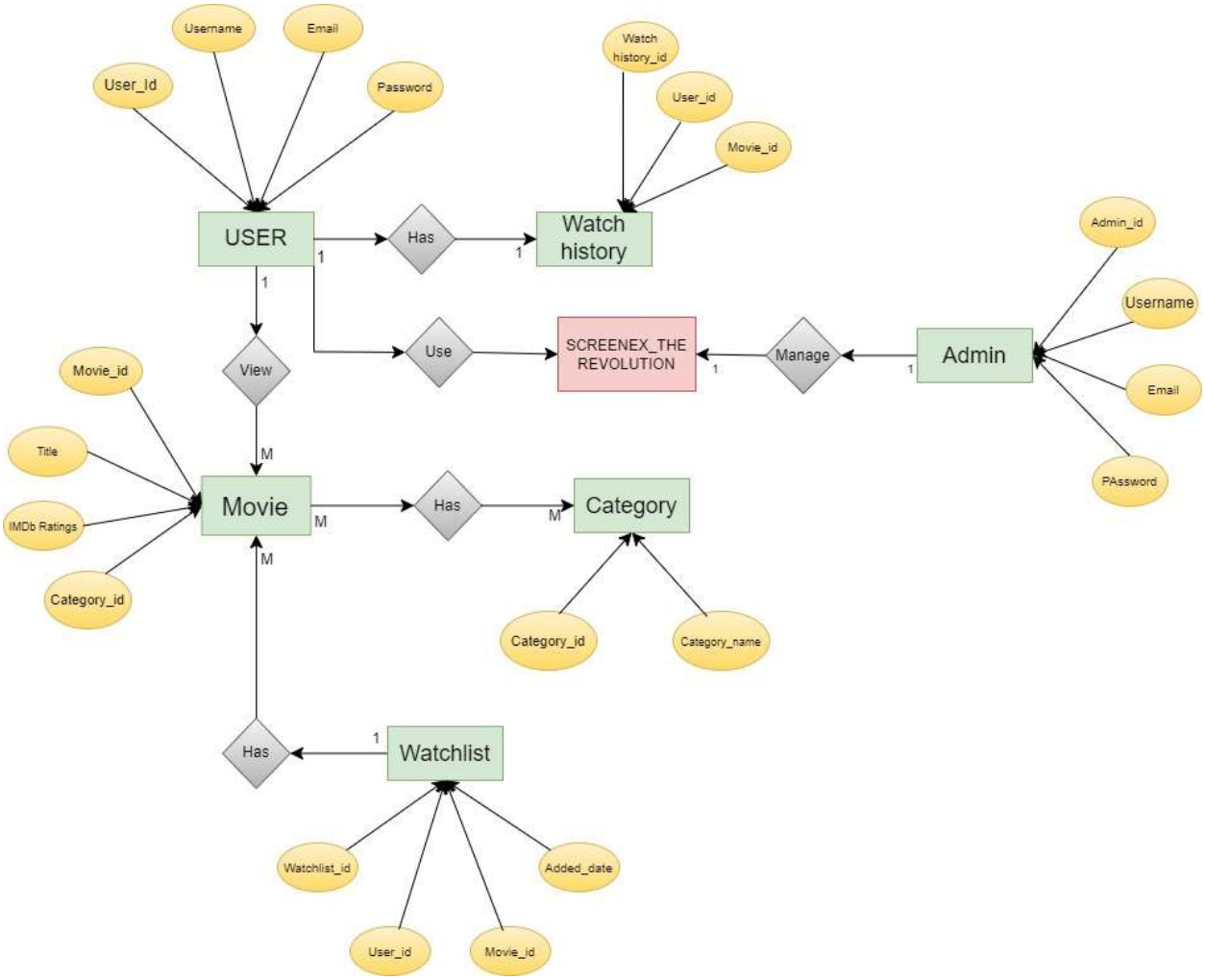
2. **User Friendly Interface:**

   - **User -Friendly: Easy navigation and seamless streaming.**
   - **Free Access: No subscriptions or payments required.**

Vission :

At Screenex, our Vision is to make high-quality entertainment accessible to all. We are committed to providing a platform where movie lovers can discover, enjoy, and share their favorite films, all at no cost. By offering a diverse range of content across Bollywood, Hollywood, and Anime, we aim to cater to varied tastes and preferences, ensuring that everyone finds something to enjoy.

ENTITY RELATIONAL DIAGRAM (ER)



NORMALISATION

DATASET:

Username	Email	Title	IMDb Rating	CategoryNam	Username2	Email3
hayley02	robert80@gmail.com	Jab We Met	9.5	Bollywood	Aashay Pariskar	aashaypariskar1605@gmail.com
ifrancis	vhurst@gmail.com	Jaane Tu Yaa Jaane Na	8.5	Hollywood	Aashay Pariskar	aashaypariskar1605@gmail.com
jenniferreeves	dunnbrandy@gmail.com	Student Of the year	8	Anime	Aashay Pariskar	aashaypariskar1605@gmail.com
romeromaria	nbaker@yahoo.com	Sanam re	7.5	Bollywood	Aashay Pariskar	aashaypariskar1605@gmail.com
heidiross	juanprice@yahoo.com	Ms Dhoni - The Untold Story	5.7	Hollywood	Aashay Pariskar	aashaypariskar1605@gmail.com

1 NF:

Username	Email	Title	IMDb Rating	CategoryNam	Username2	Email3
hayley02	robert80@gmail.com	Jab We Met	9.5	Bollywood	Aashay Pariskar	aashaypariskar1605@gmail.com
ifrancis	vhurst@gmail.com	Jaane Tu Yaa Jaane Na	8.5	Hollywood	Aashay Pariskar	aashaypariskar1605@gmail.com
jenniferreeves	dunnbrandy@gmail.com	Student Of the year	8	Anime	Aashay Pariskar	aashaypariskar1605@gmail.com
romeromaria	nbaker@yahoo.com	Sanam re	7.5	Bollywood	Aashay Pariskar	aashaypariskar1605@gmail.com
heidiross	juanprice@yahoo.com	Ms Dhoni - The Untold Story	5.7	Hollywood	Aashay Pariskar	aashaypariskar1605@gmail.com

2NF:

Username	Email	Title	IMDb Rating	CategoryNam	Username2	Email3
hayley02	robert80@gmail.com	Jab We Met	9.5	Bollywood	Aashay Pariskar	aashaypariskar1605@gmail.com
ifrancis	vhurst@gmail.com	Jaane Tu Yaa Jaane Na	8.5	Hollywood	Aashay Pariskar	aashaypariskar1605@gmail.com
jenniferreeves	dunnbrandy@gmail.com	Student Of the year	8	Anime	Aashay Pariskar	aashaypariskar1605@gmail.com
romeromaria	nbaker@yahoo.com	Sanam re	7.5	Bollywood	Aashay Pariskar	aashaypariskar1605@gmail.com
heidiross	juanprice@yahoo.com	Ms Dhoni - The Untold Story	5.7	Hollywood	Aashay Pariskar	aashaypariskar1605@gmail.com

Admin	
Username	**Email**
Aashay Pariskar	aashaypariskar1605@gmail.com

3 NF:

USER			
UserID(PK)	**Username**	**Password**	**Email**
1	hayley02	#IGBvky@I1	robert80@gmail.com
2	ifrancis	1@P8SjcLNR	vhurst@gmail.com
3	jenniferreeves	2nlVk6zg)0	dunnbrandy@gmail.com
4	romeromaria	n1vzH^d&&P	nbaker@yahoo.com
5	heidiross	!&3HhQ4#yg	juanprice@yahoo.com

CATEGORY	
CategoryID(PK)	**CategoryName**
1001	Bollywood
1002	Holywood
1003	Anime

MOVIE			
MovieID(PK)	**Title**	**IMDb Rating**	**Category id(FK)**
101	Jab We Met	9.5	1001
102	Jaane Tu Yaa Jaane Na	8.5	1001
103	Student Of the year	8	1001
104	Sanam re	7.5	1001
105	Ms Dhoni - The Untold Story	5.7	1001

WATCHLIST		
WatchlistID(PK)	**UserID(FK)**	**MovieID(FK)**
501	1	101
502	2	102
503	3	103
504	4	104
505	5	105

WATCHHISTORY

Watch History id(PI	User_Id(FK)	Movie_Id(FK)
2001	1	101
2002	2	102
2003	3	103
2004	4	104
2005	5	105

ADMIN

Adminid(PK)	Username	Email
10001	Aashay Pariskar	aashaypariskar1605@gmail.com

SQL QUERIES

Data Defination Language (DDL): DDL stands for Data Definition Language, used to define and manage database structures.

- **Create database:**

```
mysql> create database screenex;
Query OK, 1 row affected (0.03 sec)
```

- **Databases:**

```
+--------------------+
| Database           |
+--------------------+
| classicmodels      |
| company            |
| company1           |
| company_details    |
| cursor1            |
| emp_details        |
| employee_details   |
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| screenex           |
| sys                |
+--------------------+
13 rows in set (0.00 sec)

mysql>
```

- **Create table user:** The CREATE TABLE statement in SQL defines a new table. It specifies the table name and its columns with their data types

```
3 •  create table user (user_id int primary key,Username varchar(80),Email varchar(150) not null);
4 •  desc user;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
user_id	int	NO	PRI	NULL	
Username	varchar(80)	YES		NULL	
Email	varchar(150)	NO		NULL	

- **Alter Table:** The ALTER TABLE statement in SQL is used to modify an existing table's structure

- **Alter table for changing table name:**

```
3 •  alter table user rename to User_info;
4 •  desc user_info;
```

Output:

```
show tables;
alter table Us
```

Result Grid | Filter Rows:

Tables_in_screenex
category
movie
user_info
watch_history
watchlist

Result 5

Output

- **Alter table to add column after the specific column:**

```
alter table User_info add Mobile bigint after Email;
desc user_info;
```

Output:

```
desc user_info;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content

Field	Type	Null	Key	Default	Extra
user_id	int	NO	PRI	NULL	
Username	varchar(80)	YES		NULL	
Email	varchar(150)	NO		NULL	
Mobile	bigint	YES		NULL	

Result 6

➢ Alter table to add constraints:

(using another Table) Movie

```
90
91 •  alter table movie add constraint Category_id foreign key (Category_id) references category;
92 •  desc movie;
```

Output:

```
91 •  alter table movie add constraint Category_id
92 •  desc movie;
```

Result Grid | Filter Rows: | Export: | Wrap Ce

Field	Type	Null	Key	Default	Extra
movie_id	int	NO	PRI	NULL	
Title	varchar(120)	YES	UNI	NULL	
IMDb_rating	decimal(10,2)	YES		NULL	
Category_id	int	YES	MUL	NULL	

Result 9 ×

Output

> Drop Query: In SQL, DROP is used to remove objects like tables, columns, or databases from a database

> To Drop specific column

```
alter table User_info drop Mobile;
```

Output:

Result Grid | Filter Rows: | Export: | Wrap Cell

Field	Type	Null	Key	Default	Extra
user_id	int	NO	PRI	NULL	
Username	varchar(80)	YES		NULL	
Email	varchar(150)	NO		NULL	

- **Truncate:** In SQL, TRUNCATE removes all rows from a table but keeps the table structure for future use

(created a dummy table to truncate values)

- **Before truncate:**

Result Grid | Filter Rows:

Watcher_id	first_name	last_name
1	Alice	Johnson
2	Bob	Smith
3	Charlie	Davis

- **Query:**

```
125
126 •   truncate table Binge_Watchers;
```

Output:

Result Grid | Filter Rows:

Watcher_id	first_name	last_name

- **Data Manipulation Language(DML):**
  Data Manipulation Language (DML) refers to SQL commands used to manage and manipulate data within a database.

- **Insert muliple values in user:**

```sql
insert into user values
(1, 'alice_smith', 'alice.smith@example.com'),
(2, 'bob_johnson', 'bob.johnson@example.com'),
(3, 'carol_davis', 'carol.davis@example.com'),
(4, 'dave_martin', 'dave.martin@example.com'),
(5, 'emma_wilson', 'emma.wilson@example.com'),
(6, 'frank_taylor', 'frank.taylor@example.com'),
(7, 'grace_lee', 'grace.lee@example.com'),
(8, 'henry_anderson', 'henry.anderson@example.com'),
(9, 'isabel_thomas', 'isabel.thomas@example.com'),
(10, 'jack_clark', 'jack.clark@example.com'),
(11, 'karen_wright', 'karen.wright@example.com'),
(12, 'luke_moore', 'luke.moore@example.com'),
(13, 'mona_james', 'mona.james@example.com'),
(14, 'nathan_white', 'nathan.white@example.com'),
(15, 'olivia_harris', 'olivia.harris@example.com');
```

Output:

Result Grid | Filter Rows: | Edit:

user_id	Username	Email
1	alice_smith	alice.smith@example.com
2	bob_johnson	bob.johnson@example.com
3	carol_davis	carol.davis@example.com
4	dave_martin	dave.martin@example.com
5	emma_wilson	emma.wilson@example.com
6	frank_taylor	frank.taylor@example.com
7	grace_lee	grace.lee@example.com
8	henry_anderson	henry.anderson@example.com
9	isabel_thomas	isabel.thomas@example.com
10	jack_clark	jack.clark@example.com
11	karen_wright	karen.wright@example.com
12	luke_moore	luke.moore@example.com
13	mona_james	mona.james@example.com
14	nathan_white	nathan.white@example.com
15	olivia_harris	olivia.harris@example.com

user 3

➢ Insert Single Value in table:

```
insert into User_info values (16,'Amar_gupta','guptamar0911@gmail.com');
select * from User_info;
```

Output:

Result Grid | Filter Rows: | Edit:

user_id	Username	Email
1	alice_smith	alice.smith@example.com
2	bob_johnson	bob.johnson@example.com
3	carol_davis	carol.davis@example.com
4	dave_martin	dave.martin@example.com
5	emma_wilson	emma.wilson@example.com
6	frank_taylor	frank.taylor@example.com
7	grace_lee	grace.lee@example.com
8	henry_anderson	henry.anderson@example.com
9	isabel_thomas	isabel.thomas@example.com
10	jack_clark	jack.clark@example.com
11	karen_wright	karen.wright@example.com
12	luke_moore	luke.moore@example.com
13	mona_james	mona.james@example.com
14	nathan_white	nathan.white@example.com
15	olivia_harris	olivia.harris@example.com
16	Amar_gupta	guptamar0911@gmail.com

> Update Query: In SQL, the UPDATE statement is used to modify existing records in a table.

>Update singe value

Before Update Query:

Result Grid | Filter Rows: | Edit:

user_id	Username	Email
4	dave_martin	dave.martin@example.com
5	emma_wilson	emma.wilson@example.com
6	frank_taylor	frank.taylor@example.com
7	grace_lee	grace.lee@example.com
8	henry_anderson	henry.anderson@example.com
9	isabel_thomas	isabel.thomas@example.com
10	jack_clark	jack.clark@example.com
11	karen_wright	karen.wright@example.com
12	luke_moore	luke.moore@example.com
13	mona_james	mona.james@example.com
14	nathan_white	nathan.white@example.com
15	olivia_harris	olivia.harris@example.com
16	Amar_gupta	guptamar0911@gmail.com
NULL	NULL	NULL

➢ Query

```
select * from User_info;
update User_info set Username=Abhishek where user_id=16;
```

Output

Result Grid | Filter Rows: | Edit:

user_id	Username	Email
4	dave_martin	dave.martin@example.com
5	emma_wilson	emma.wilson@example.com
6	frank_taylor	frank.taylor@example.com
7	grace_lee	grace.lee@example.com
8	henry_anderson	henry.anderson@example.com
9	isabel_thomas	isabel.thomas@example.com
10	jack_clark	jack.clark@example.com
11	karen_wright	karen.wright@example.com
12	luke_moore	luke.moore@example.com
13	mona_james	mona.james@example.com
14	nathan_white	nathan.white@example.com
15	olivia_harris	olivia.harris@example.com
16	Abhishek	guptamar0911@gmail.com
NULL	NULL	NULL

➢ Update Multiple values at a time :

Before query:

Result Grid | Filter Rows: | Edit: | Expor

user_id	Username	Email	Country
1	alice_smith	alice.smith@example.com	NULL
2	bob_johnson	bob.johnson@example.com	NULL
3	carol_davis	carol.davis@example.com	NULL
4	dave_martin	dave.martin@example.com	NULL
5	emma_wilson	emma.wilson@example.com	NULL
6	frank_taylor	frank.taylor@example.com	NULL
7	grace_lee	grace.lee@example.com	NULL
8	henry_anderson	henry.anderson@example.com	NULL
9	isabel_thomas	isabel.thomas@example.com	NULL

➢ Query:

```
update User_info set Country="India";
```

Output:

Result Grid | Filter Rows: | Edit: | Expo

user_id	Username	Email	Country
1	alice_smith	alice.smith@example.com	India
2	bob_johnson	bob.johnson@example.com	India
3	carol_davis	carol.davis@example.com	India
4	dave_martin	dave.martin@example.com	India
5	emma_wilson	emma.wilson@example.com	India
6	frank_taylor	frank.taylor@example.com	India
7	grace_lee	grace.lee@example.com	India
8	henry_anderson	henry.anderson@example.com	India

- **Delete:** The DELETE statement in SQL is used to remove rows from a table

Delete specific Row

(Dummy Table)

- **Before:**

Result Grid | Filter Rows:

Watcher_id	first_name	last_name
1	Alice	Johnson
2	Bob	Smith
3	Charlie	Davis
4	Aashay	Pariskar
5	Amar	gupta
6	Abhishek	more

Binge_Watchers 24

- **Query:**

```
delete from Binge_Watchers where Watcher_id=3;
select * from Binge_Watchers;
```

Output:

Result Grid | Filter Rows:

Watcher_id	first_name	last_name
1	Alice	Johnson
2	Bob	Smith
4	Aashay	Pariskar
5	Amar	gupta
6	Abhishek	more

Binge_Watchers 25

- **Delete all rows from table :**

Before:

212

Result Grid | Filter Rows:

Watcher_id	first_name	last_name
1	Alice	Johnson
2	Bob	Smith
4	Aashay	Pariskar
5	Amar	gupta
6	Abhishek	more

Binge_Watchers 25

- **Query:**

```
select * from Binge_Watchers;
delete from Binge_Watchers;
```

Output:

212

Result Grid | Filter Rows: | Export:

	Watcher_id	first_name	last_name

- **Data Query Language (DQL):** Data Query Language (DQL) is a subset of SQL used for querying data from a database

- **Select query(To fetch all record):**

```
select * from User_info;
```

Output:

Result Grid | Filter Rows: | Edit:

	user_id	Username	Email	Country
▶	1	alice_smith	alice.smith@example.com	India
	2	bob_johnson	bob.johnson@example.com	India
	3	carol_davis	carol.davis@example.com	India
	4	dave_martin	dave.martin@example.com	India
	5	emma_wilson	emma.wilson@example.com	India
	6	frank_taylor	frank.taylor@example.com	India
	7	grace_lee	grace.lee@example.com	India
	8	henry_anderson	henry.anderson@example.com	India
	9	isabel_thomas	isabel.thomas@example.com	India
	10	jack_clark	jack.clark@example.com	India
	11	karen_wright	karen.wright@example.com	India
	12	luke_moore	luke.moore@example.com	India
	13	mona_james	mona.james@example.com	India

User_info 27 ×

➢ Select specific Column:

```
select Username from User_info;
```

Output:

Result Grid | Filter

Username
alice_smith
bob_johnson
carol_davis
dave_martin
emma_wilson
frank_taylor
grace_lee
henry_anderson
isabel_thomas
jack_clark
karen_wright
luke_moore
mona_james

User_info 28

➢ Select with 'Where' clause:

```
select Username from User_info;
select Username,Email from User_info where user_id=3;
```

Output:

Result Grid | Filter Rows:

Username	Email
carol_davis	carol.davis@example.com

- ## Select with Multiple Condition:

```
select Username,Email from User_info where User_id=15 And Country='India';
```

Output:

Result Grid | Filter Rows:

Username	Email
olivia_harris	olivia.harris@example.com

- ## Select with 'Or' Condition:

```
select * from User_info where User_id=5 or country='India';
```

Output:

Result Grid | Filter Rows: | Edit: | Export/

user_id	Username	Email	Country
1	alice_smith	alice.smith@example.com	India
2	bob_johnson	bob.johnson@example.com	India
3	carol_davis	carol.davis@example.com	India
4	dave_martin	dave.martin@example.com	India
5	emma_wilson	emma.wilson@example.com	India

User_info 34 ×

Output

- **Fetch record using 'Order by'**

```
select * from User_info order by Username;
```

Output:

```
29 •  select username,email from user_info where user_id=
```

Result Grid | Filter Rows: | Edit: | Exp

user_id	Username	Email	Country
16	Abhishek	guptamar0911@gmail.com	India
1	alice_smith	alice.smith@example.co	India
2	bob_johnson	bob.johnson@example.com	India
3	carol_davis	carol.davis@example.com	India
4	dave_martin	dave.martin@example.com	India
5	emma_wilson	emma.wilson@example.com	India
6	frank_taylor	frank.taylor@example.com	India
7	grace_lee	grace.lee@example.com	India
8	henry_anderson	henry.anderson@example.com	India
9	isabel_thomas	isabel.thomas@example.com	India
10	jack_clark	jack.clark@example.com	India
11	karen_wright	karen.wright@example.com	India
12	luke_moore	luke.moore@example.com	India
13	mona_james	mona.james@example.com	India

alice.smith@example.co

User_info 35 ×

➢ Fetch Records using order by and desc:

```
select * from User_info order by Username desc;
```

Output:

Result Grid | Filter Rows: | Edit: | Expor

user_id	Username	Email	Country
15	olivia_harris	olivia.harris@example.com	India
14	nathan_white	nathan.white@example.com	India
13	mona_james	mona.james@example.com	India
12	luke_moore	luke.moore@example.com	India
11	karen_wright	karen.wright@example.com	India
10	jack_clark	jack.clark@example.com	India
9	isabel_thomas	isabel.thomas@example.com	India
8	henry_anderson	henry.anderson@example.com	India
7	grace_lee	grace.lee@example.com	India
6	frank_taylor	frank.taylor@example.com	India
5	emma_wilson	emma.wilson@example.com	India
4	dave_martin	dave.martin@example.com	India
3	carol_davis	carol.davis@example.com	India

➢ Select using 'between'

```
select * from User_info where User_id between 1 and 8;
```

Output:

Result Grid | Filter Rows: | Edit:

user_id	Username	Email	Country
1	alice_smith	alice.smith@example.com	India
2	bob_johnson	bob.johnson@example.com	India
3	carol_davis	carol.davis@example.com	India
4	dave_martin	dave.martin@example.com	India
5	emma_wilson	emma.wilson@example.com	India
6	frank_taylor	frank.taylor@example.com	India
7	grace_lee	grace.lee@example.com	India
8	henry_anderson	henry.anderson@example.com	India
NULL	NULL	NULL	NULL

- **Select using 'Like' Operator:**

```
select Username from User_info where Username Like "c%";
```

Output:

Result Grid

Username
carol_davis

- **Select statement using last word of the name:**

```
select Username from User_info where Username Like "_____k";
```

Output:

Result Grid

Username
Abhishek

- **Select with 'limit':**

```
select * from User_info limit 6;
```

Output:

Result Grid | Filter Rows: | Edit:

user_id	Username	Email	Country
1	alice_smith	alice.smith@example.com	India
2	bob_johnson	bob.johnson@example.com	India
3	carol_davis	carol.davis@example.com	India
4	dave_martin	dave.martin@example.com	India
5	emma_wilson	emma.wilson@example.com	India
6	frank_taylor	frank.taylor@example.com	India
NULL	NULL	NULL	NULL

- **Aggregate Functions:** Aggregate functions in SQL perform calculations on multiple rows of a table and return a single result

- **Count :**

```
SELECT COUNT(Username) FROM User_info;
```

Output:

Result Grid | Filter

	COUNT(Username)
▶	16

- **Count function with providing a name:**

```
SELECT COUNT(Username) as "Total Users" FROM User_info;
```

Output:

	Total Users
▶	16

- **Sum function (created a dummy table to perform this):**

```
select sum(Subscription_Amt) As "Total_Amount" from Binge_Watchers;
```

Output:

Result Grid | Filter

Total_Amount
2794

Refresh

- **Avg function:**

```
select avg(Subscription_Amt) As "Average_price" from Binge_Watchers;
```

Output:

Result Grid | Filter Ro

Average_price
465.6667

- **Min Function:**

```
select min(Subscription_Amt) As "Minimum_price" from Binge_Watchers;
```

Output:

Result Grid | Filter R

Minimum_price
299

➢ Max Function :

```
select max(Subscription_Amt) As "Minimum_price" from Binge_Watchers;
```

Output:

Result Grid

	Minimum_price
▶	999

➢ Using Aggregate Function with Group by clause :

```
SELECT Subscription_type, min(Subscription_Amt) AS "subscription_amt"
FROM Binge_Watchers
GROUP BY Subscription_type;
```

Output:

Result Grid | Filter Rows:

	Subscription_type	subscription_amt
▶	Silver	299
	Gold	399
	Platinum	499
	Diamond	999

- **String Functions :** String functions in SQL are used to perform operations on text data

- **Concat function:**

```
SELECT CONCAT(first_name, ' ', last_name) AS full_name
FROM Binge_Watchers;
```

Output:

Result Grid | Filter Rov

full_name
Alice Johnson
Bob Smith
Charlie Davis
Aashay Pariskar
Amar gupta

Result 62

- **Uppercase:**

```
SELECT UPPER(first_name) AS upper_First_Name
FROM Binge_Watchers;
```

Output:

Result Grid | Filter Ro

upper_First_Name
ALICE
BOB
CHARLIE
AASHAY
AMAR

Result 63

Output

- **Lower:**

```
SELECT LOWER(last_name) AS lower_last_name
FROM Binge_Watchers;
```

Output:

Result Grid | Filter R

lower_last_name
johnson
smith
davis
pariskar
gupta

Result 64

- **Length:**

```
select first_name,Length(first_name) as length from Binge_Watchers;
```

Output:

Result Grid | Filter Rows

first_name	length
Alice	5
Bob	3
Charlie	7
Aashay	6
Amar	4

Result 65

- **Conditional Statement:** In SQL, conditional statements are used to execute different actions based on certain conditions.

```sql
select first_name,last_name,
Case
when Subscription_Amt>399 then 'Special member'
else 'Normal member'
end as Membership
from Binge_Watchers;
```

Output:

Result Grid | Filter Rows:

first_name	last_name	Membership
Alice	Johnson	Normal member
Bob	Smith	Normal member
Charlie	Davis	Normal member
Aashay	Pariskar	Normal member
Amar	gupta	Special member

Result 66

➢ Comparison Function :

➢ Greater than function

```
SELECT * FROM Binge_Watchers WHERE Subscription_Amt >299;
```

Output:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Watcher_id	first_name	last_name	Subscription_Amt	Subscription_type
4	Aashay	Pariskar	399	Gold
5	Amar	gupta	499	Platinum
6	Abhishek	more	999	Diamond

➢ Less Than Function:

```
SELECT * FROM Binge_Watchers WHERE Subscription_Amt >299;
SELECT * FROM Binge_Watchers WHERE Subscription_Amt <499;
```

Output:

Result Grid | Filter Rows: | Export: | Wrap Cell Conten

Watcher_id	first_name	last_name	Subscription_Amt	Subscription_type
1	Alice	Johnson	299	Silver
2	Bob	Smith	299	Silver
3	Charlie	Davis	299	Silver
4	Aashay	Pariskar	399	Gold

Binge_Watchers 68

- **Equal = Function:**

```
SELECT * FROM Binge_Watchers WHERE Subscription_Amt =499;
```

Output:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Watcher_id	first_name	last_name	Subscription_Amt	Subscription_type
5	Amar	gupta	499	Platinum

- **Not equal:**

```
SELECT * FROM Binge_Watchers WHERE Watcher_id!= 2;
```

Output:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Watcher_id	first_name	last_name	Subscription_Amt	Subscription_type
1	Alice	Johnson	299	Silver
3	Charlie	Davis	299	Silver
4	Aashay	Pariskar	399	Gold
5	Amar	gupta	499	Platinum
6	Abhishek	more	999	Diamond

Binge_Watchers 70

- **SUBQUERY:** A subquery is a query within another query used to provide intermediate results or conditions.

➢ Single row subquery (performing subquery in movie and category table)

Find the Highest-Rated Movie:

```
SELECT Title, IMDb_rating, Category_id
FROM movie
WHERE IMDb_rating = (SELECT MAX(IMDb_rating) FROM Movie);
```

Output:

Result Grid | Filter Rows:

Title	IMDb_rating	Category_id
Jab We Met	9.50	1001

Find Movies in the Same Category as the Highest-Rated Movie:

```
SELECT Title, IMDb_rating
FROM Movie
WHERE Category_id = (SELECT Category_id
                     FROM Movie
                     WHERE IMDb_rating = (SELECT MAX(IMDb_rating) FROM Movie));
```

Output:

301

Result Grid | Filter Rows: | Export

Title	IMDb_rating
Jab We Met	9.50
Jaane Tu Yaa Jaane Na	8.50
Student Of the Year	8.00
Sanam Re	7.50
Ms Dhoni - The Untold Story	5.70
Cuttputli	8.90
12th Fail	6.20
Animal	3.50
Housefull 4	7.70
Shiddat	7.60
Salaar	6.50
Sita Ramam	8.50

Movie 74

➢ Multiple Row Subquery (using IN):

```
SELECT Title, IMDb_rating
FROM Movie
WHERE Category_id IN (SELECT Category_id
                      FROM category
                      WHERE Category_name IN ('Bollywood', 'Anime'));
```

Output:

Result Grid | Filter Rows:

Title	IMDb_rating
Jab We Met	9.50
Jaane Tu Yaa Jaane Na	8.50
Student Of the Year	8.00
Sanam Re	7.50
Ms Dhoni - The Untold Story	5.70
Cuttputli	8.90
12th Fail	6.20
Animal	3.50
Housefull 4	7.70
Shiddat	7.60
Salaar	6.50
Sita Ramam	8.50
Wake Up Sid	7.60
Pushpa	7.60

Movie 81

Output

➢ Multiple Row Subquery (using ANY):

```
select Title,IMDb_rating
From movie
where IMDb_rating > any (
Select IMDb_rating
from movie where IMDb_rating=7.50);
```

Output:

Result Grid | Filter Rows:

Title	IMDb_rating
Jab We Met	9.50
Jaane Tu Yaa Jaane Na	8.50
Student Of the Year	8.00
Cuttputli	8.90
Housefull 4	7.70
Shiddat	7.60
Sita Ramam	8.50
Wake Up Sid	7.60
Pushpa	7.60
Baahubali	8.00
Baahubali 2	8.20

movie 82

Output

➢ Multiple Row Subquery (using ALL):

```
select Title,IMDb_rating
From movie
where IMDb_rating > all (
Select IMDb_rating
from movie where IMDb_rating=7.50);
```

Output:

Result Grid | Filter Rows:

Title	IMDb_rating
Jab We Met	9.50
Jaane Tu Yaa Jaane Na	8.50
Student Of the Year	8.00
Cuttputli	8.90
Housefull 4	7.70
Shiddat	7.60
Sita Ramam	8.50
Wake Up Sid	7.60
Pushpa	7.60
Baahubali	8.00
Baahubali 2	8.20
One piece	8.30

movie 83

- **JOINS:** Joins combine rows from two or more tables based on a related column.
- **Join:**

```
select m.Title,m.movie_id,c.Category_name
 from movie m join category c
 on m.Category_id=c.Category_id;
```

Output:

Result Grid | Filter Rows: | Export:

Title	movie_id	Category_name
Animal	108	Bollywood
Housefull 4	109	Bollywood
Shiddat	113	Bollywood
Salaar	114	Bollywood
Sita Ramam	115	Bollywood
Wake Up Sid	116	Bollywood
Pushpa	117	Bollywood
Baahubali	118	Bolly Bollywood
Baahubali 2	119	Bollywood
Bholaa	120	Bollywood
Thursday	110	Hollywood
Dr. Strange	111	Hollywood
Iron Man 3	112	Hollywood
One piece	121	Anime
Attack on titan	122	Anime

Result 10

➢ Inner Join:

```
select m.Title,m.movie_id,m.IMDb_rating,c.Category_name
from movie m inner join category c
on m.Category_id=c.Category_id;
```

Output:

Result Grid | Filter Rows: | Export: | Wrap C

Title	movie_id	IMDb_rating	Category_name
Jab We Met	101	9.50	Bollywood
Jaane Tu Yaa Jaane Na	102	8.50	Bollywood
Student Of the Year	103	8.00	Bollywood
Sanam Re	104	7.50	Bollywood
Ms Dhoni - The Untold Story	105	5.70	Bollywood
Cuttputli	106	8.90	Bollywood
12th Fail	107	6.20	Bollywood
Animal	108	3.50	Bollywood
Housefull 4	109	7.70	Bollywood
Shiddat	113	7.60	Bollywood
Salaar	114	6.50	Bollywood
Sita Ramam	115	8.50	Bollywood
Wake Up Sid	116	7.60	Bollywood
Pushpa	117	7.60	Bollywood
Baahubali	118	8.00	Bollywood
Baahubali 2	119	8.20	Bollywood
Bholaa	120	6.00	Bollywood
Thursday	110	7.50	Hollywood
Dr. Strange	111	7.10	Hollywood
Iron Man 3	112	7.10	Hollywood

Result 12

➢ Left Join:

```
select m.Title,m.movie_id,m.IMDb_rating,c.Category_name
from movie m left join category c on m.Category_id=c.Category_id;
```

Output:

Result Grid | Filter Rows: | Export: | Wrap Cell Co

Title	movie_id	IMDb_rating	Category_name
Jab We Met	101	9.50	Bollywood
Jaane Tu Yaa Jaane Na	102	8.50	Bollywood
Student Of the Year	103	8.00	Bollywood
Sanam Re	104	7.50	Bollywood
Ms Dhoni - The Untold Story	105	5.70	Bollywood
Cuttputli	106	8.90	Bollywood
12th Fail	107	6.20	Bollywood
Animal	108	3.50	Bollywood
Housefull 4	109	7.70	Bollywood
Thursday	110	7.50	Hollywood
Dr. Strange	111	7.10	Hollywood

Result 41

➢ Right join:

- ```
  select m.Title,m.movie_id,m.IMDb_rating,c.Category_name
  from movie m right join category c on m.Category_id=c.Category_id;
  ```

Output:

Result Grid | Filter Rows: | Export: | Wrap Cell C

Title	movie_id	IMDb_rating	Category_name
Wake Up Sid	116	7.60	Bollywood
Pushpa	117	7.60	Bollywood
Baahubali	118	8.00	Bollywood
Baahubali 2	119	8.20	Bollywood
Bholaa	120	6.00	Bollywood
Thursd Bholaa	110	7.50	Hollywood
Dr. Strange	111	7.10	Hollywood
Iron Man 3	112	7.10	Hollywood
One piece	121	8.30	Anime
Attack on titan	122	9.10	Anime
NULL	NULL	NULL	Tollywood

Result 42

Output
  ```

➢ Cross join:

```
select m.Title,m.movie_id,m.IMDb_rating,c.Category_name
from movie m cross join category c;
```

Output:

Result Grid | Filter Rows: | Export: | Wrap Cell

Title	movie_id	IMDb_rating	Category_name
Jab We Met	101	9.50	Anime
Jab We Met	101	9.50	Hollywood
Jab We Met	101	9.50	Bollywood
Jaane Tu Yaa Jaane Na	102	8.50	Anime
Jaane Tu Yaa Jaane Na	102	8.50	Hollywood
Jaane Tu Yaa Jaane Na	102	8.50	Bollywood
Student Of the Year	103	8.00	Anime
Student Of the Year	103	8.00	Hollywood
Student Of the Year	103	8.00	Bollywood
Sanam Re	104	7.50	Anime
Sanam Re	104	7.50	Hollywood
Sanam Re	104	7.50	Bollywood
Ms Dhoni - The Untold ...	105	5.70	Bollywood
Ms Dhoni - The Untold ...	105	5.70	Hollywood
Ms Dhoni - The Untold ...	105	5.70	Bollywood
Cuttputli	106	8.90	Anime
Cuttputli	106	8.90	Hollywood
Cuttputli	106	8.90	Bollywood

Result 16

➢ Natural join:

```
use screenex;
select * from movie natural join category;
```

Output:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Category_id	movie_id	Title	IMDb_rating	Category_name
1001	101	Jab We Met	9.50	Bollywood
1001	102	Jaane Tu Yaa Jaane Na	8.50	Bollywood
1001	103	Student Of the Year	8.00	Bollywood
1001	104	Sanam Re	7.50	Bollywood
1001	105	Ms Dhoni - The Untold Story	5.70	Bollywood
1001	106	Cuttputli	8.90	Bollywood
1001	107	12th Fail	6.20	Bollywood
1001	108	Animal	3.50	Bollywood
1001	109	Housefull 4	7.70	Bollywood
1002	110	Thursday	7.50	Hollywood
1002	111	Dr. Strange	7.10	Hollywood
1002	112	Iron Man 3	7.10	Hollywood
1001	113	Shiddat	7.60	Bollywood
1001	114	Salaar	6.50	Bollywood
1001	115	Sita Ramam	8.50	Bollywood
1001	116	Wake Up Sid	7.60	Bollywood

Result 1

- **VIEWS:** A view is a virtual table in SQL that provides a way to simplify complex queries or present data from one or more tables in a specific format.

```
Create view Movie_Details As
select movie_id,Title from movie where IMdb_rating>7;
```

```
select * from Movie_Details;
```

Output:

Result Grid | Filter Rows:

movie_id	Title
101	Jab We Met
102	Jaane [illegible] Na
103	Student Of the Year
104	Sanam Re
106	Cuttputli
109	Housefull 4
110	Thursday
111	Dr. Strange
112	Iron Man 3
113	Shiddat
115	Sita Ramam
116	Wake Up Sid
117	Pushpa
118	Baahubali

Movie_Details 2

- **Stored Procedure :** A stored procedure is a precompiled SQL program that can be executed with a single call, allowing for reusable and modular code in the database.
- **Using(IN)**

```
DELIMITER //
create procedure Getmovie_info1(IN G_Category_id int)
begin
    select * from movie
    where Category_id=G_Category_id;
END //
DELIMITER ;

call Getmovie_info1(1001);
```

Output:

Result Grid | Filter Rows: | Export: | Wrap Cell Conten

movie_id	Title	IMDb_rating	Category_id
101	Jab We Met	9.50	1001
102	Jaane Tu Yaa Jaane Na	8.50	1001
103	Student Of the Year	8.00	1001
104	Sanam Re	7.50	1001
105	Ms Dhoni - The Untold Story	5.70	1001
106	Cuttputli	8.90	1001
107	12th Fail	6.20	1001
108	Animal	3.50	1001
109	Housefull 4	7.70	1001
113	Shiddat	7.60	1001
114	Salaar	6.50	1001
115	Sita Ramam	8.50	1001
116	Wake Up Sid	7.60	1001

Cuttputli

Result 5

Output

- **Stored procedure:**
- **Using (Out):**

```
DELIMITER //
create procedure Getmovie_info10(IN G_Category_id int,OUT Total_movie int)
begin
    select count(*) into Total_movie from movie
    where Category_id=G_Category_id;
END //
DELIMITER ;

call Getmovie_info10(1001,@Total_movie);
select @Total_movie;
```

Output:

Result Grid

@Total_movie
17

- **Cursor:** A cursor in SQL is a database object used to retrieve, manipulate, and navigate through rows of a result set one at a time

```
delimiter //
create procedure proc_User1()
begin
        declare eUsername varchar(120);
        declare eEmail varchar(120);
        declare cur cursor for select Username,Email from User_info;
        open cur;
        get_user:LOOP
           fetch cur into eUsername,eEmail;
           select concat(eUsername," ",eEmail);
        END LOOP get_user;
        close cur;
end //
delimiter ;
call proc_User();

select * from User_info;
```

Output:

Result Grid | Filter Rows:

	concat(eUsername,"",eEmail)
▶	Abhishekguptamar0911@gmail.com

Result Grid | Filter Rows: | Export:

	concat(eUsername,"",eEmail)
▶	henry_andersonhenry.anderson@example.com

Result Grid | Filter Rows: | Expo

concat(eUsername,"",eEmail)
karen_wrightkaren.wright@example.com

- **Triggers:** Triggers are special SQL procedures that automatically execute in response to certain events on a table, such as INSERT, UPDATE, or DELETE.
-

```
CREATE TABLE AuditLog1000 (
    ActionType VARCHAR(50),
    ActionTime DATETIME);

DELIMITER $$
CREATE TRIGGER log_admin_insert177
AFTER INSERT
ON movie
FOR EACH ROW
BEGIN
    INSERT INTO AuditLog998 (ActionType, ActionTime)
    VALUES (
        'INSERT',
        NOW());
END $$
DELIMITER ;
```

Output:

456

Result Grid | Filter Rows:

ActionType	ActionTime
Insert	2024-08-09 16:08:19