**DATABASE MANAGEMENT SYSTEM PROJECT**

**MOVIE TICKET BOOKING SYSTEM.**

**AIM**

The **Movie Ticket Booking System** Database Design is basically aimed to provide complete information of the movie and schedule to the customer, according to which he/she can easily get a ticket instantly and can book a ticket on his/her favorite movies. Admin can use Movie Ticketing System to insert and delete data such as movie descriptions, movie schedules which will update the related webpage and will be accessible by the customers. Admin can update the webpage changing according to the data in the database also admin can check the statistics information from the system. Operate your cinema with better efficiency by automating reservation and ticketing process improve profitability and manage your cinema better by having access to key data in a centralized and systematic view and increase customer satisfaction by giving your customers what they want when it comes to the seat preference.

Based on seat and ticket reservation system allowing booking in a few easy steps. This powerful movie ticketing system can be deployed on any website offering tickets for movies, theater, and other scheduled performances.

**PROPOSED WORK DETAILS**

The proposed Database consists of 6 tables that are interconnected. The team members work on tables and keep updating them by implementing queries.

**DDL Statements:**

In the context of SQL, data definition or data description language (DDL) is a syntax for creating and modifying database objects such as tables, indices, and users.

* CREATE to create a new table or database.
* ALTER for alteration.
* Truncate to delete data from the table.
* DROP to drop a table
* RENAME to rename a table.

**1.Halls Table :-**

create table hall(hall\_id int,class varchar2(20),no\_of\_seats int)

**Table created.**

**2.Movies Table:**

create table movies(movie\_id int,language varchar2(20),movie\_name varchar2(20),type varchar2(20))

**Table created.**

**3.** **Shows Table:-**

create table shows(show\_id int,movie\_id int,day varchar2(20),duration int)

**Table created.**

**4.Prices Table:-**

create table prices(price\_id int,day varchar2(20),amount int)

**Table created.**

**5.** **Tickets Table:-**

create table prices(price\_id int,day varchar2(20),amount int)

**Table created.**

alter table tickets add movie\_name varchar(20)

**Table altered.**

**DML Statements**

A data manipulation language (DML) is a computer programming language used for adding (inserting), deleting, and modifying (updating) data in a database.

[INSERT](https://www.geeksforgeeks.org/sql-insert-statement/) – is used to insert data into a table.

[UPDATE](https://www.geeksforgeeks.org/sql-update-statement/) – is used to update existing data within a table.

[DELETE](https://www.geeksforgeeks.org/sql-delete-statement/) – is used to delete records from a database table.

insert into hall values(1,'gold',50)

**1 row(s) inserted**

insert into hall values(2,'silver',75)

**1 row(s) inserted**

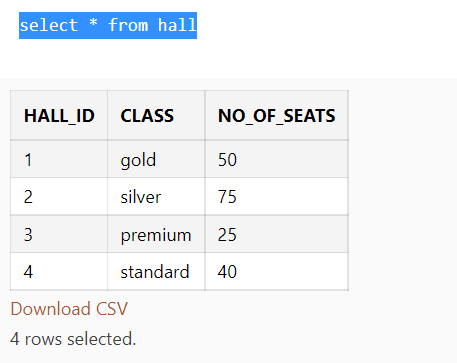
insert into hall values(3,'premium',25)

**1 row(s) inserted.**

insert into hall values(4,'standard',40)

**1 row(s) inserted.**

select \* from hall

****

insert into movies values(101,'Telugu','RRR','Action')

**1 row(s) inserted.**

insert into movies values(102,'Hindi','PK','Comedy')

**1 row(s) inserted.**

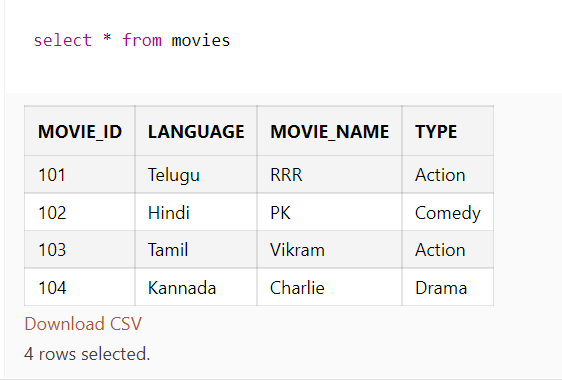
insert into movies values(103,'Tamil','Vikram','Action')

**1 row(s) inserted**.

insert into movies values(104,'Kannada','Charlie','Drama')

**1 row(s) inserted.**

select \* from movies

****

insert into shows values(501,102,'Thursday',3)

**1 row(s) inserted.**

insert into shows values(502,101,'Friday',3)

**1 row(s) inserted.**

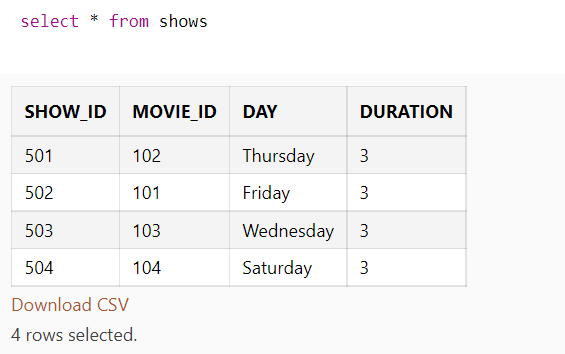
insert into shows values(503,103,'Wednesday',3)

**1 row(s) inserted.**

insert into shows Values(504,104,'Saturday',3)

**1 row(s) inserted**

select \* from shows



insert into prices values(701,'Thursday',500)

**1 row(s) inserted.**

insert into prices values(702,'Friday',350)

**1 row(s) inserted.**

insert into prices values(703,'Wednesday',200)

**1 row(s) inserted.**

insert into prices values(704,'Sunday',750)

**1 row(s) inserted.**

insert into prices values(705,'M0nday',250)

**1 row(s) inserted.**

select \* from prices

insert into tickets values(1001,5,45,'RRR')

**1 row(s) inserted.**

insert into tickets values(1002,7,39,'PK')

**1 row(s) inserted.**

insert into tickets values(1003,2,28,'Vikram')

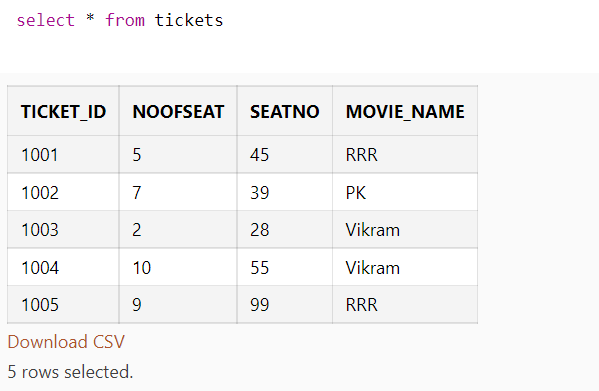
**1 row(s) inserted.**

insert into tickets values(1004,10,55,'Vikram')

**1 row(s) inserted.**

insert into tickets values(1005,9,99,'RRR')

**1 row(s) inserted.**



update prices set amount=300 where price\_id=703

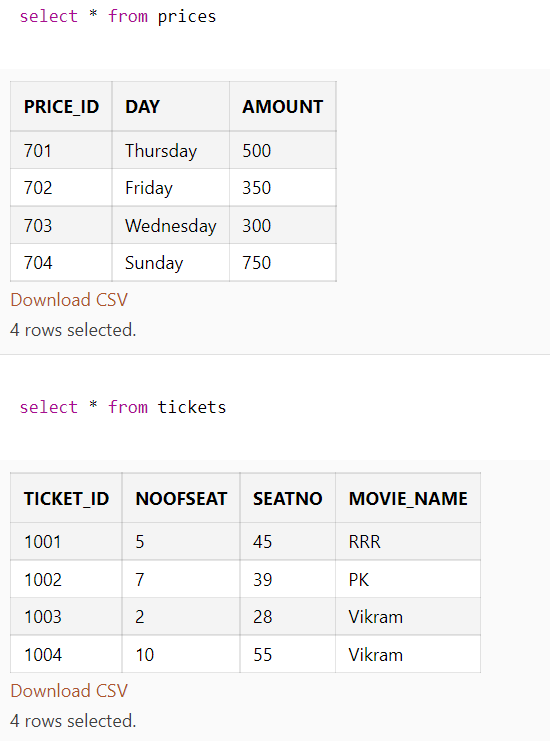
**1 row(s) updated.**

delete from prices where amount=250

**1 row(s) deleted.**

delete from tickets where seatno=99

**1 row(s) deleted.**

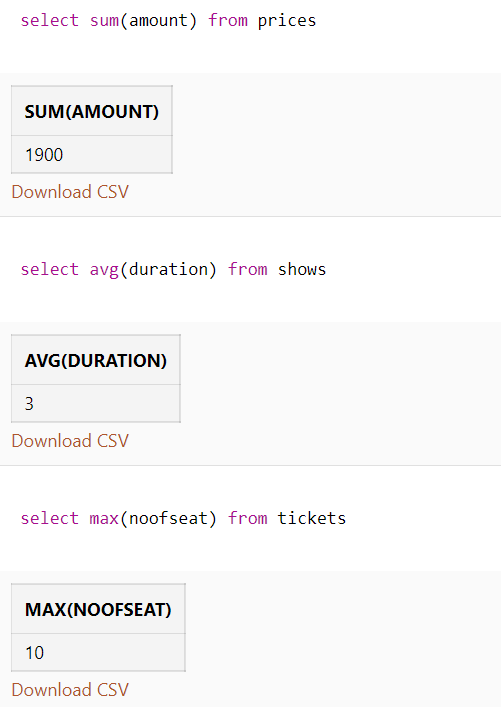


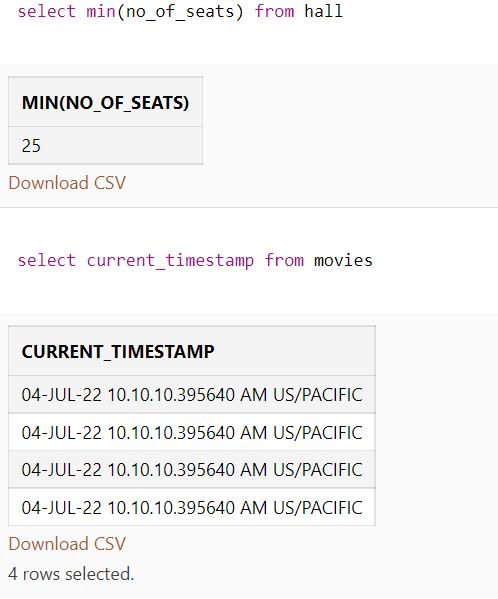
commit

**Statement processed.**

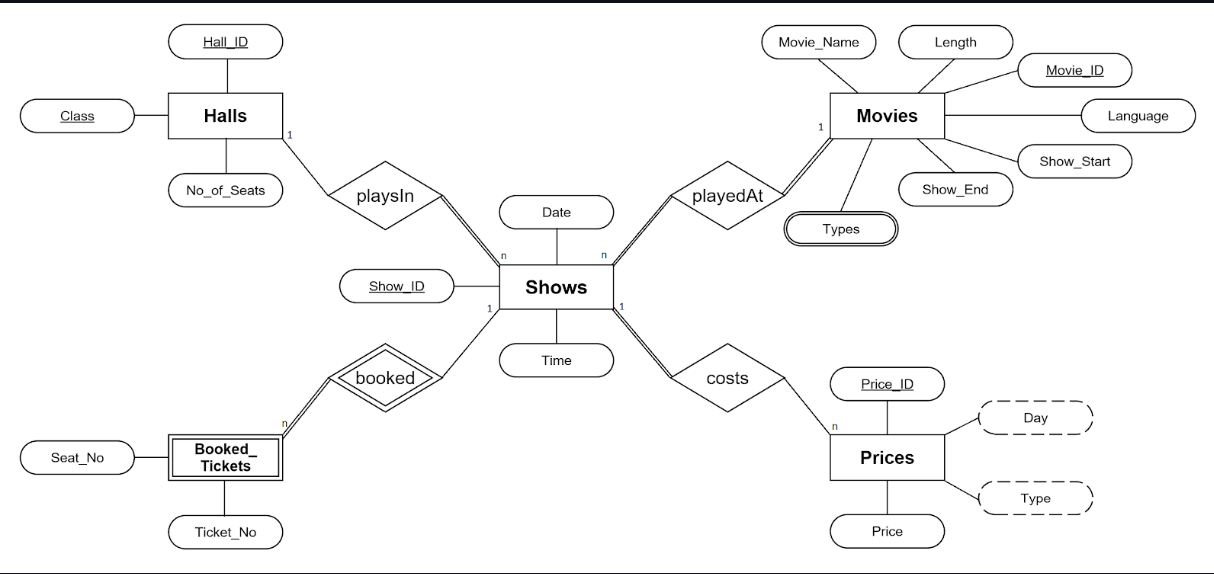
**INBUILT FUNCTIONS:**

A built-in function is a function that is already available in a programming language, application, or another tool that can be accessed by end users.





**ER-Diagram:-**



**JOINS:-**

A joins  clause is used to combine rows from two or more tables, based on a related column between them.

Different Types of SQL Joins:

(INNER) JOIN: Returns records that have matching values in both tables

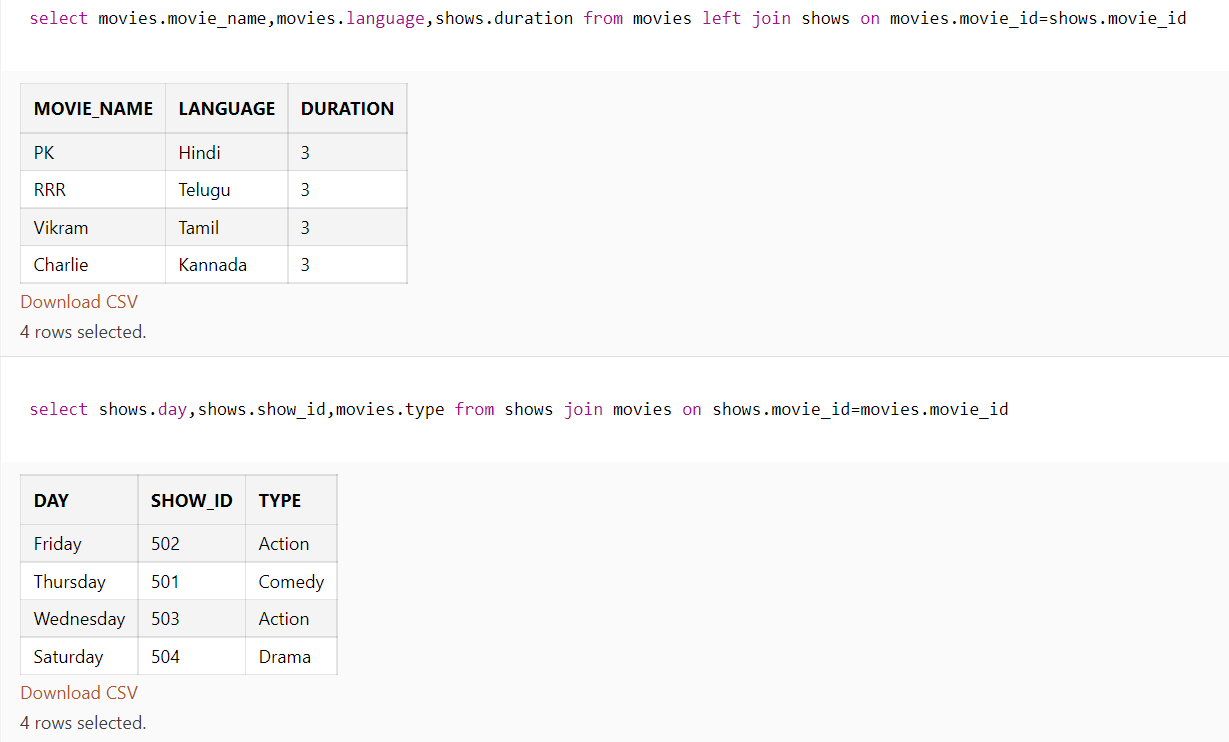
LEFT (OUTER) JOIN: Returns all records from the left table, and the matched records from the right table

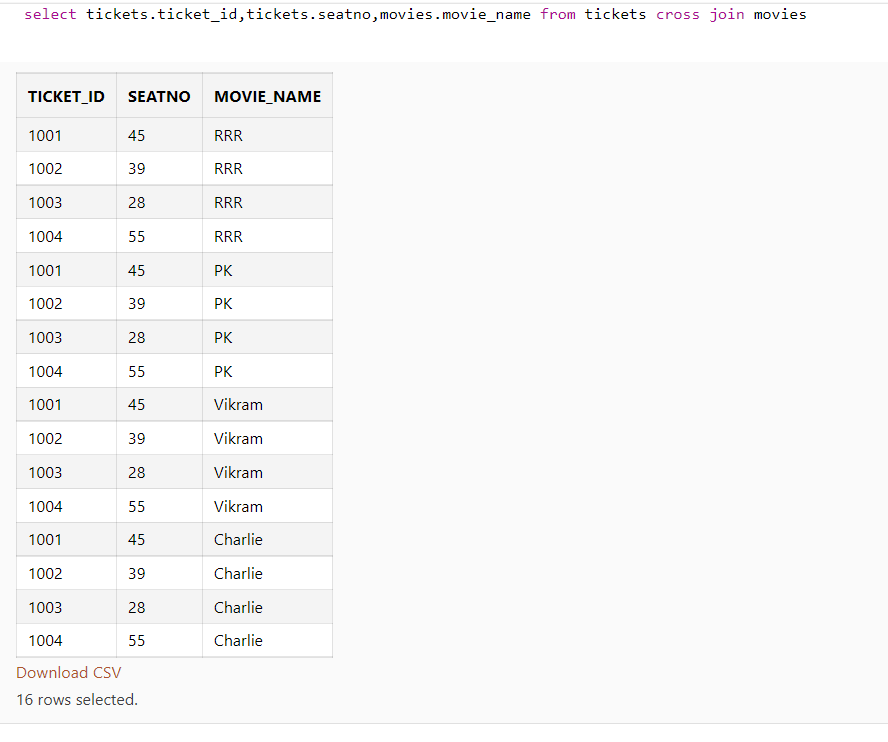
RIGHT (OUTER) JOIN: Returns all records from the right table, and the matched records from the left table

FULL (OUTER) JOIN: Returns all records when there is a match in either left or right table.

NATURAL JOIN: Returns all records based on the common columns in the two tables being joined.

SELF JOIN: A table is joined with itself.





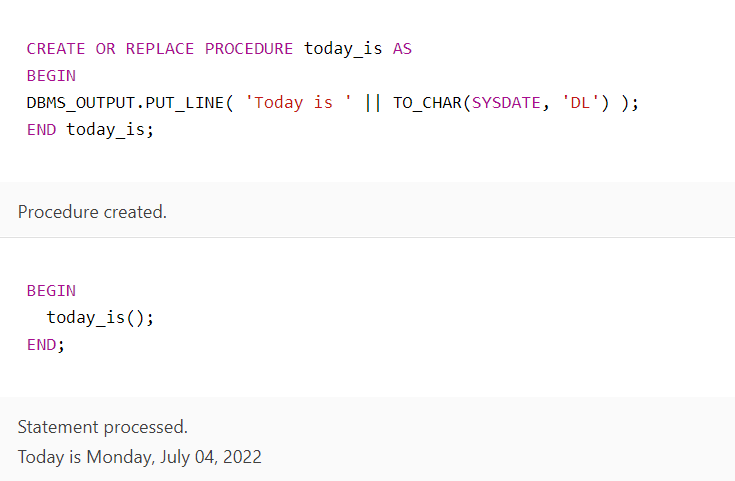
**SUBQUERIES:-**

A query within a query is known as subquery**.**

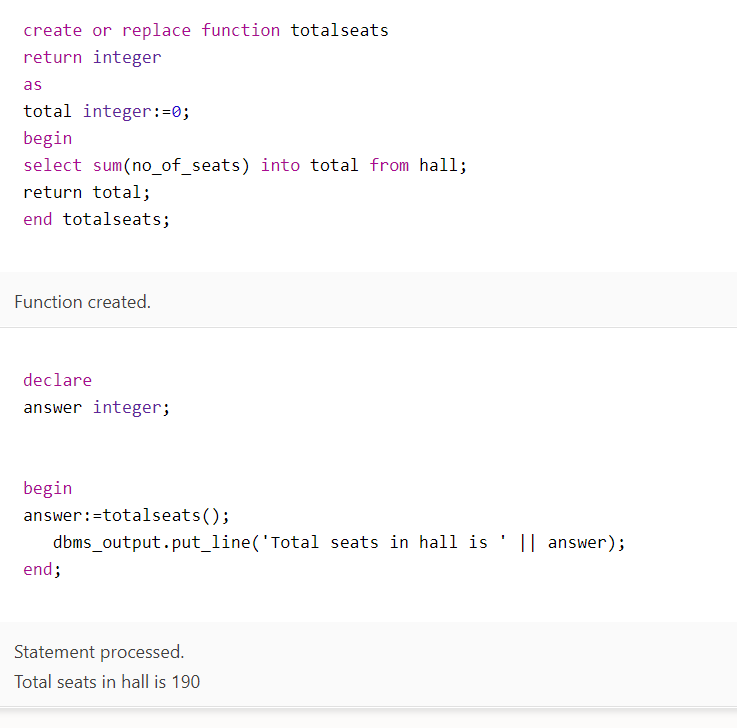
****

**PL/SQL PROGRAMS:**

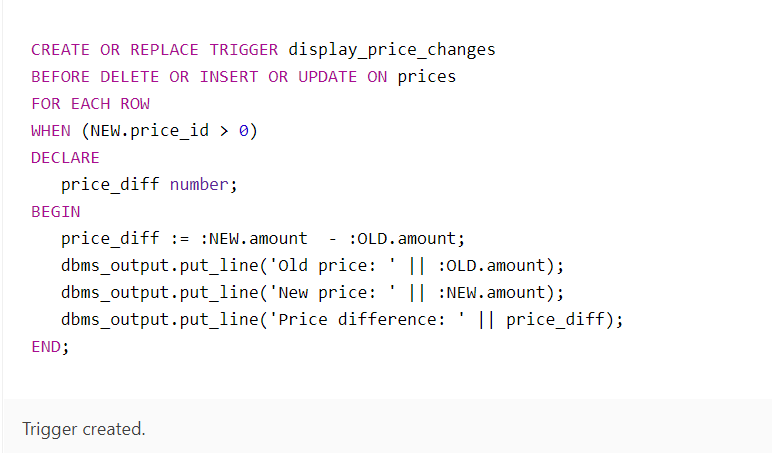
**PROCEDURES-**

****

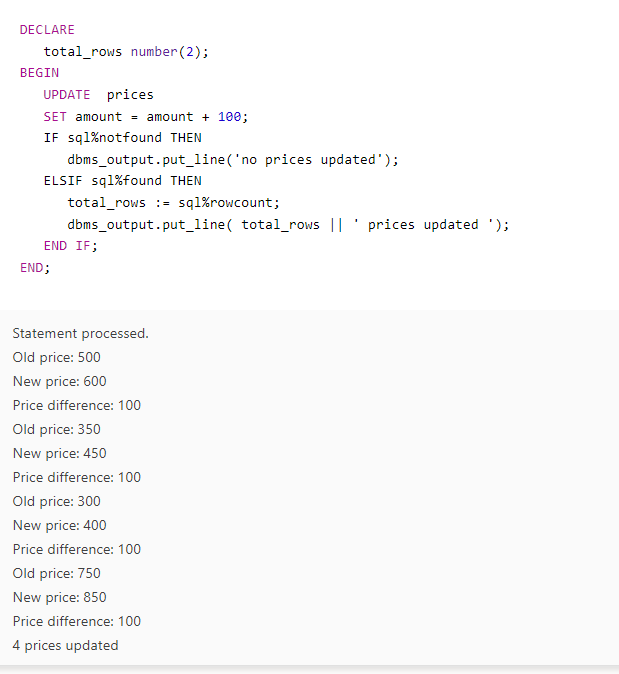
**FUNCTIONS:**

****

**TRIGGERS:**

****

**CURSORS:-**

****

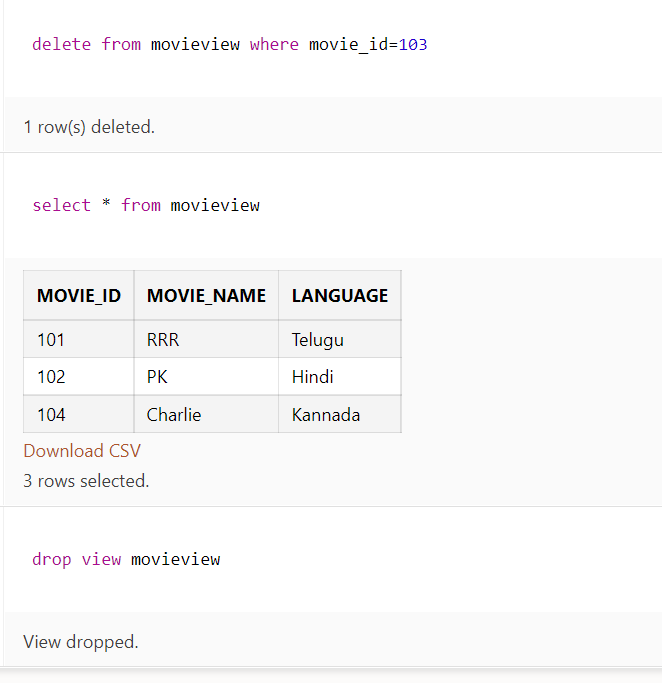
**SET OPERATORS:-**

****

****

**VIEWS:-**

****

****