```
select * from movies;

select title,industry from movies

select title,industry from movies where industry="Bollywood"; -- we can search a particular thing like this ,by where clauses

SELECT * FROM movies where industry ="bollywood";

SELECT count(*) FROM movies where industry ="bollywood"; -- count() is used to get the number of elements

select  distinct industry from movies; -- distinct for having the distinct values or elements

select * from movies where title like "Thor%"; -- here % is used for filling the previous or past by anything,like if the data
                                                is I love you ,we can find it by %love%

Select * from movies where title like "% America%";

select * from movies where studio="";          -- Searching for movies that don't have any studio name.

select * from moviesbd.movies;

SELECT * FROM movies;

select max(imdb_rating) from movies;

select min(imdb_rating) from movies where industry="bollywood";

select avg(imdb_rating) from movies where industry="bollywood";

select round (avg (imdb_rating),2)from movies where industry="bollywood";
```

we can make a table like that

```
select max(imdb_rating) as max_rating,
      min(imdb_rating) as min_rating,
       avg(imdb_rating) as avg_rating
from movies
where industry="bollywood"; -- have to select the whole 4 lines before executing.

/* The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

  GROUP BY Syntax

SELECT column_name(s)

FROM table_name
```

```
WHERE condition

GROUP BY column_name(s)

ORDER BY column_name(s);

*/


select industry,count(*) from movies group by industry; -- here industry is the coloumn name.

select studio,count(*) from movies group by studio;


select

   studio,count(*) as cnt

   from movies

   group by studio

   order by cnt desc; -- can be written like this.


select

  studio, -- this is the main column

  count(studio) as cnt, -- these are the secondary column

  avg(imdb_rating) as avg_rating,   -- these are the secondary column

  round(avg(imdb_rating),2) as round -- these are the secondary column

from movies

group by studio -- group by the main column

order by round desc;

-- now here we will work with the actor table.

select * from actors;

-- now if we want a column named as age,we need the current year for that.we can get the formula for implementing the current year from google(smart way)

SELECT * , YEAR(CURDATE())- birth_year AS age from actors order by age;

/* here The comma (,) is used in SQL to separate multiple columns that you want to select in a SELECT statement.

 In your query, the comma is necessary because you want to select all columns from the "actors" table in
```

```
 addition to the calculated "current_year" column. */


-- -- now here we will work with the financial table.

select * from financials;

select *,revenue-budget as profit from financials;

select movie_id,budget,revenue,(revenue-budget) as profit from financials; -- i can just print profit along with some selected comumns


-- if i wanna convrt all USD currency into INR into a new column

select *,if(currency="USD",revenue*77,revenue) as revenue_in_INR from financials;

-- if conditions work as IF(condition,what if True,what if False)


select  count(distinct(unit)) from financials;


-- now if we wann work with multiple conditions then we cannot use if function ,here we have to work with " case-- end " function

-- converting all units into millions


select *,

case

   when unit="Billions" then revenue*1000

  when unit="Thousands" then revenue/1000

  else revenue

 end as unit_million -- -- NB if i wanna name the new column as unit(million) i have to do it like "unit(million)" this

 from financials

-- multiple table, company uses multiple table for 1. save space by avoiding repeatation 2. organize data better 3. make updates easily

-- JOING TWO TABLE--

-- in movies and financials table we have a common column named movie_id
```

```
Select
  movies.movie_id,title,budget, revenue ,currency ,unit -- which columns i want in my new table
from movies
join financials -- if we will write join by default it will represent inner join
on movies.movie_id=financials.movie_id; -- on from where.common column= from where . common column

-- for shortcut
Select
  m.movie_id,title,budget, revenue ,currency ,unit
from movies m  -- here m is representing movies
join financials f -- here f is representing financials
on m.movie_id=f.movie_id; -- on from where.common column= from where . common column

/* Inner Join: Retrieves matching rows from both tables, excluding non-matching rows.

Left Join (Left Outer Join): Includes all rows from the left table and matching rows from the right, filling unmatched columns with NULL values.

Right Join (Right Outer Join): Includes all rows from the right table and matching rows from the left, filling unmatched columns with NULL values.

Full Join (Full Outer Join): Retrieves all rows from both tables, filling unmatched columns with NULL values where necessary. */

Select
  m.movie_id,title,budget, revenue ,currency ,unit
from movies m
left join financials f -- it will take all movies_id from left table
```

```
on m.movie_id=f.movie_id ;



Select

  f.movie_id,title,budget, revenue ,currency ,unit -- here we have to write f.movie_id because we are taking all from right table

from movies m

right join financials f -- it will take all movies_id from left table

on m.movie_id=f.movie_id;


-- here we cannot use outer join directly ,for outer join we have to add left and right join by union function



Select

  m.movie_id,title,budget, revenue ,currency ,unit from movies m  left join financials f on m.movie_id=f.movie_id

union

Select

  f.movie_id,title,budget, revenue ,currency ,unit from movies m  right join financials f on m.movie_id=f.movie_id;


  /* using can be used instead of "on" in joining, if there is more then one common column we can use " using(x,y) ".we can use this  for single cases also and

  In that case, we don't need to write movies.movie_id or this kind of staff" */


  select

   movie_id,title,budget, revenue ,currency ,unit -- no need of m.movie_id

   from movies

   join financials

   using (movie_id) -- this one is easy and convenient
```