IMDB Movies SQL Queries

```
use imdb_movies;


select * from movies;


select count(*) from movies where gross is null;


alter table movies rename column Series_Title to Title;
alter table movies rename column No_of_Votes to Votes;
alter table movies rename column IMDB_Rating to Rating;
alter table movies rename column Meta_score to Score;
alter table movies modify column Released_Year int;

SET SQL_SAFE_UPDATES = 0;

update movies
set gross = replace(gross, ',', '');

update movies
set gross = null
where gross = '';

update movies
set Certificate = 'Not Rated'
where Certificate is null;

select avg(score) from movies where score is not null;

update movies
set score = 82.3812
where score is null;

update movies
set Runtime = replace(Runtime, ' min', '');

alter table movies modify column Runtime int;
alter table movies modify column gross int;

# Average Score
select avg(score) from movies where score is not null;

# Average Gross
select avg(gross) from movies;

# Movies with Rating Above Average
select count(*) from movies
where rating > ( select avg(rating) from movies );
```

```
# Movie/TV show with the highest gross

select title, gross from movies
order by gross desc
limit 1;

select title, gross from movies
where gross = (select max(gross) from movies);

# Find how many movies have rating grater than avg of all the movie ratings

select count(*) from movies
where rating > (select avg(rating) from movies);

# Find the highest rated movie of each year

select title ,released_year, rating , max_rating
from (select *,
      max(rating) over(partition by released_year order by released_year ) as max_rating
   from movies) t
      where t.rating = t. max_rating
      order by released_year desc ;

# Select the highest rated movie among all movies whoes number of votes are greater the
dataset avg votes

select title, rating, votes, max_rating, avg_votes
from (select *,
 max(rating) over() as max_rating,
 avg(votes) over() as avg_votes
 from movies) t
    where t.rating = max_rating and t.votes > avg_votes;

# Find all movies made by top 3 directors (in terms of total gross income)

with top_directors as (
  select director
  from movies
  group by director
  order by sum(gross) desc
  limit 3
)
select title, director
from movies
where director in (select * from top_directors);

# Find movies of all those actors whoes filmography avg rating > 8.5 (take 25000 votes
as cutoff)

select title , star1, rating, votes
from movies
where star1 in (select star1
    from movies
    where votes > 25000
```

```
    group by star1
                having avg(rating) > 8.5 );


# Find the highest gross movie of each year

select title , Released_Year, gross
from movies
where (released_year, gross) in (select released_year, max(gross)
      from movies
      group by released_year)
                        order by released_year desc;

select title, released_year, gross
from (select *,
 max(gross) over(partition by released_year order by  released_year) as max_gross
 from movies) t
    where t.gross = max_gross
    order by released_year desc ;


# Find the highest grossing movies of top 5 actor/director combo in terms of total gross
income

with top_dous as ( select director, star1, max(gross)
   from movies
   group by director, star1
   order by sum(gross) desc
            limit 5)

select title, star1, director
from movies
where (director, star1, gross) in (select * from top_dous);

# Find all the movies that have a rating higher than the average rating of movies in the
same genre.

select title, genre, rating, avg_rating
from (select *,
 round(avg(rating) over(partition by genre order by genre) , 2) as avg_rating
 from movies) t
    where t.rating > t.avg_rating ;

# Get the percentage of votes for each movie compared to the total number of votes.

select title, round(((votes)/ t.total_votes) * 100 ,2) as vote_percentage
from (select *,
 sum(votes) over() as total_votes
 from movies) t;


select title, (votes/(select sum(votes) from movies)) * 100 as vote_percentage
from movies ;
```

```
# Find genre having avg score > avg score of all movies.

select genre, avg(score)
from movies
group by genre
having avg(score) > (select avg(score) from movies);

# Rank top 3 Movies by Rating within Each Year

select title, released_year, rating, movie_rank
from (select *,
 dense_rank() over(partition by released_year order by rating desc) as movie_rank
 from movies) t
    where t.movie_rank < 4
    order by released_year desc;

# Top 3 Movies with the Highest Gross in Each Year (Using ROW_NUMBER())

select title, released_year, movie_rank
from (select *,
 row_number() over(partition by released_year order by gross desc) as movie_rank
 from movies) t
    where t.movie_rank < 4
    order by released_year desc;

#
_____________________________________________________________________________________
_____________________________________________________________________________

# Find the Average Gross Revenue for Movies Released in Each Genre

select genre, avg(gross)
from movies
group by genre;

# Find the Top 5 Movies with the Most Votes

select title, votes
from movies
order by votes desc
limit 5;

# Find the Year with the Highest Average Rating.

select released_year , avg(rating)
from movies
group by released_year
order by avg(rating) desc
limit 1;

# Find the Number of Movies Released Each Year.

select released_year , count(*) as total_movies
from movies
```

```
group by released_year
order by released_year desc;

# Top 5 Movies with the Longest Runtime

select title, concat(round((runtime/60),2),' ','hrs') as runtime_Hrs
from movies
order by runtime desc
limit 5;

# Top 5 Movies with the Lowest Runtime

select title, concat(round((runtime/60),2),' ','hrs') as runtime
from movies
order by runtime
limit 5;

#  Movies with the Most Grossing Revenue by Genre
# Purpose: Identify which genre has the highest total gross revenue.

select genre, sum(gross) as total_gross
from movies
group by genre
order by total_gross desc
limit 1 ;

# Average Rating by Certificate
# Purpose: Calculate the average rating for movies grouped by their certificate type
(e.g., U/A, A).

select certificate, round(avg(rating),1) as avg_rating
from movies
group by certificate
order by avg_rating desc ;

update movies set certificate = "UA" where certificate = "U/A";
update movies set certificate = "UA" where certificate = "";

# Movies Released After 2010 with Rating Above 8
# Purpose: Identify movies released after 2010 that have a rating above 8.

select title, released_year, rating
from movies
where released_year > 2010
order by released_year desc;


# Top 3 Directors with the Most Votes
# Purpose: Identify the top 3 directors with the most number of votes.

select director, sum(votes) as total_votes
from movies
group by director
order by total_votes desc
```

```
limit 3;

# Movies with Rating Below Average but High Gross Revenue
# Purpose: Identify movies that have a rating below the average but still have high gross revenue.

select title, rating, gross
from movies
where rating < (select avg(rating) from movies) and
gross > (select avg(gross) from movies)
order by gross desc;

# Top 5 Highest Grossing Movies of the Last Decade (2010-2019)
# Purpose: Find the top 5 highest-grossing movies released between 2010 and 2019.

select title, gross, released_year
from movies
where released_year between 2010 and 2019
order by gross desc
limit 5;

# Find Movies with No Gross Revenue but Have a Rating Above 8
# Purpose: Find movies that have no recorded gross revenue but have a rating above 8.

select title, rating, gross
from movies
where rating > 8 and gross is null;

# Find the Number of Movies in Each Genre with a Rating Above 7
# Purpose: Count the number of movies for each genre where the rating is above 7.4

select genre, count(*) total_count
from movies
where rating > 7
group by genre
order by total_count desc;

# Find Movies with the Highest Number of Votes in Each Year
# Purpose: Get the movie with the highest number of votes in each year.

select title, released_year, votes
from (select *,
 rank() over(partition by released_year order by votes desc) as movie_rank
 from movies) t
    where t.movie_rank < 2
    order by released_year desc ;

# Find the Most Successful Director in Terms of Total Gross
# Purpose: Identify the director who has the highest total gross revenue.

select director, sum(gross) as total_gross
from movies
group by director
order by total_gross desc
```

```
limit 1;

# Movies Released Before 2000 with a Rating Above 8
# Purpose: Find all movies released before the year 2000 that have a rating above 8.

select title, released_year, rating
from movies
where released_year < 2000 and rating > 8
order by released_year desc;
```