**MySQL**

**CHAPTER 2:**

1. **What is database?**

A database is nothing but a collection of tabular data built across rows and columns.

As of excel files there are also rows and columns with different sheets so each excel sheet in SQl is called a table like if there are 5 excel sheets then that file in sql will have 5 tables.

As a data scientist most of the time we will be dealing with querying the data instead of creating or updating rows and columns.

**Queries**:

* Select \* from moviesdb.movies;

This query returns all the records for that database table.

**Select, where, distinct, like, order by.**

* Select \* from \_\_\_;

The \* or asterisk here shows the selection of all the columns.

But if we are interested in some specific columns like if there are 10 columns and we want to select 2 of them then the query for this is:

* SELECT title, industry FROM moviesdb.movies;

This part of query **FROM moviesdb.movies** has database and table and sometimes it can be confusing so it can be replaced with:

* USE moviesdb;
* SELECT title, industry FROM movies;

This use query will tell SQL that this moviesdb is my default database so whatever table I’m referring to I mean that particular database.

This is used when our database is not set as default like if moviesdb is not the default database then by using the USE query it is set default for that particular query.

We can also make moviesdb or any other dB as our default database by selecting the specific dB in setting and set it to default, what it will do is, after this we will not need the USE query to call our database we can directly run the select query without calling the database:

* Select title, industry from movies;

In our excel sheet we have two type of columns:

* One is text column which contains only text.
* Other is numerical columns which contains number etc.

Now we will query dB using text query

* SELECT \* FROM movies WHERE industry=”Bollywood”;
* SELECT COUNT(\*) FROM movies WHERE industry="Bollywood";

The 1st query will return a specific text from the table of dB while the 2nd query will count the number of that text in the entire column.

SQL queries are not case sensitive for example if our table has a row name in capital letters like “Hollywood” and then we call it from select function with lower case letters like “hollywood” then it will not get errors it still will work.

To find out unique values in the database we use distinct query:

* SELECT distinct industry FROM movies;

It will show unique industries in movies Table. After this we need to specify the column for which we want to get unique values.

**Example:**

Let’s say you are working in a company called IMDb website which is the movies website and this website has Advance search option where you can search movies by their title rating and so on. I want to know all the THOR movies which are released so for so I can type in title THOR and then search and will see all the Thor movies, so imagine you are working in IMDb as Database Engineer or Data analyst and when on the website some user is making this query from website it will make a call to a backend that backend could be written in JAVA, PYTHON, C++, Kotlin, whatever that language is and when in backend you know that user is querying all the movies which has THOR in the title somewhere it will make an SQL query into a database and it will retrieve all the records like all THOR movies and the SQL query will look something like this:

* SELECT \* FROM movies WHERE title LIKE “%THOR%”

%% means before THOR there could be any string and after that there could be any string, in the title if THOR appears somewhere in the string give me those movies and we will see that we get all the movies with THOR name.

This query is called the **wild card** search. Since all the movies name start from THOR so if we did not give “THOR%” starting % in this query with THOR it will still going to work.

“THOR%” this query means that all the movies that starts with THOR followed by any character any numbers of times, percentage means that. The end % with THOR is important because without this it will not show any result.

For Captain America the THOR like query will not work if we search only “America%” because this movie starts with Captain so for that we should put starting % like “%America%”.

Now if we go back to our original query:

* SELECT \* FROM movies;

There we can see that some of our cells in a column or row are empty and to print all those columns with one empty cells like in our database the table movies have a column studio which have some empty cells or values and to print all the movies with no studio defined so for that:

* SELECT \* FROM movies WHERE studio=””;

**SAVING QUERIES FOR FUTURE REFERENCES:**

* Select \* from moviesdb.movies;
* SELECT \* from movies where industry=”Bollywood”;
* SELECT DISTINCT industry from movies;
* SELECT \* from movies where title LIKE “THOR%;
* SELECT \* from movies where title LIKE “%America%;

**TAKE AWAYS:**

* **SELECT**, **FROM** and **WHERE** are the basics SQL functions.
* **“\*”** means all the columns. Using **“\*”** after select query will select all the columns of a db.
* With the help of the **USE** function, you can indicate the query to use a particular database, especially when there are multiple databases.
* The **COUNT** function will provide the numerical count of rows.
* The **DISTINCT** function will help you see the UNIQUE values present in a given column.
* **“%”** is a **wild** card search.
* Use **LIKE** function and **“%”** to filter the rows based on a text value.

**EXERCISE:**

Write SQL queries for the following:

1. Print all movie titles and release year for all Marvel Studios movies.  
2. Print all movies that have Avenger in their name.  
3. Print the year when the movie "The Godfather" was released.  
4. Print all distinct movie studios in the Bollywood industry.

**CHAPTER: 2.1 (Numerical queries)**

Once again we are visiting IMDb website, where we want to see all the movies which has rating greater than 9 or equal to 9 so click on rating column and select 9 and then search we will see all the movies with rating 9 or more than 9, so we do the exact function in SQL dB as:

* SELECT \* from movies WHERE imdb\_rating>=9;
* SELECT \* FROM movies WHERE imdb\_rating>=9;
* SELECT \* FROM movies WHERE imdb\_rating>9;
* SELECT \* FROM movies WHERE imdb\_rating<7;
* SELECT \* FROM movies WHERE imdb\_rating>=6 AND imdb\_rating<=8;
* # same function can be used as
* SELECT \* FROM movies WHERE imdb\_rating BETWEEN 6 AND 8;

The END function is used to check the values between ranges like 6-8 so in between something we used AND but there is another function called OR operator which includes different values like for imdb rating release year we use:

* SELECT \* FROM movies WHERE release\_year=2022 or release\_year=2019 or release\_year=2018;

Now we can have 5 such year’s distinct years and we want to see all the movies the better idea is to use IN operator, so now we will do the exactly same thing using IN operator so to include 2022 2019 2018 such as:

* SELECT \* FROM movies WHERE release\_year IN (2022, 2019, 2018);

We can use this IN operator for text query as well basically we want to select CLAUSE WITH WHERE condition and we are having couple of OR conditions so there are OR (2022,2019,2022).

So for text queries we can do the same as:

* SELECT \* FROM movies WHERE studio IN (“Marvel Studios”, “Zee Studios”);

Now let’s get back to the movies table we can see that imdb rating has some missing null values so to retrieve null values:

* SELECT \* FROM movies WHERE imdb\_rating is NULL;

We can exactly do the opposite by using:

* SELECT \* FROM movies WHERE imdb\_rating is NOT NULL;

Let’s say we want to print all the imdb rating of Bollywood movies so for that:

* SELECT \* FROM movies WHERE industry=”Bollywood”

Now we can see all the bollywood movies with different rating but we want to order them from lowest to greatest or greatest or lowest. So for that:

* SELECT \* FROM movies WHERE industry=”Bollywood” ORDER BY imdb\_rating

So now all the values of imdb rating are set by order from lowest to greatest.

By default the order is ascending but we can set it to descending order as:

* SELECT \* FROM movies WHERE industry=”Bollywood” ORDER BY imdb\_rating DESC

What it does is it set the values according to descending order starting from greatest to lowest.

By default it is always in ascending order but we can use ascending order command

* SELECT \* FROM movies Where industry=’bollywood’ ORDER BY imdb\_rating asc;

We can also set to see only the first 5 highest rated movies as:

* SELECT \* FROM movies WHERE industry=”Bollywood” ORDER BY imdb\_rating DESC LIMIT 5;

We can save these records from MYSQL workbench to an external file from the result section.

* SELECT \* FROM movies WHERE industry=”Bollywood” ORDER BY imdb\_rating DESC LIMIT 5;

Let’s say we want to start from the 2nd cell or 2nd top rated movie so for that:

* SELECT \* FROM movies WHERE industry=”Bollywood” ORDER BY imdb\_rating DESC LIMIT 5 OFFSET 1;

**EXERCISE:**

Write SQL queries for the following:

1. Print all movies in the order of their release year (latest first)  
2. All movies released in the year 2022  
3. Now all the movies released after 2020  
4. All movies after the year 2020 that have more than 8 rating  
5. Select all movies that are by Marvel studios and Hombale Films  
6. Select all THOR movies by their release year  
7. Select all movies that are not from Marvel Studios

**Key Takeaways:**

* <, <=, >, >= are the basic numerical operators used in SQL.
* You can also use AND, OR, BETWEEN, IN to perform numerical queries.
* You can sort the table by using ORDER BY clause.
* By default, it sorts the data in ascending order but you can specify the sorting order.
* LIMIT clause can be used to fetch the top N or bottom N amount of records. N can be any numerical value.
* OFFSET clause will help you skip a certain number of rows in your final results.

**Chapter: 3**

**Summary Analytics**

(Min, Max, AVG, Group By)

In previous lectures we look at count (\*) star function which can give you the total record count in a given table or for a given condition.

* SELECT COUNT(\*) from movies where industry =’bollywood’;

**COUNT STAR:**

Count star is an inbuilt SQL function, now you may wonder what exactly a function is so let’s understand it by an example of a computer.

On computer we have various keys like power key, led key, USB input or output so these are the inbuilt features of a computer i-e we can turn on or off the monitor so these are inbuilt features of a computer and these features are turning on or off a computer etc. Similarly in SQL there are various inbuilt features COUNT star is one of them and the way you call this inbuilt feature also knows as function is that you will write the name of the function then you will have bracket inside the bracket here we have star so this is called an argument in a function:

**COUNT (\*)**

Now we will look into more of these like functions which allows you to do basic analytics

1. **MAX**

The first one we will be covering is MAX. Let’s say we want to get maximum imdb rating for bollywood movies so the query for this is:

* Select MAX(imdb\_rating) from movies where industry=”bollywood”

1. **MIN**

Similarly there is a MIN function which will show you the minimum rating among all the bollywood movies, the query is:

* Select MIN(imdb\_rating) from movies where industry=”bollywood”

1. **AVERAGE**

Let’s say MARVEL studios is releasing all the awesome movies and we want to know the average rating of marvel movies we can use this function called AVG:

* Select **AVG**(imdb\_rating) from movies where studio=”Marvel Studios”

By doing this it will show us the average rating of marvel movies. It shows too many decimal points like 7.5000 so we can restrict it to 2 decimal points like 7.50 by:

* Select **ROUND**(AVG(imdb\_rating),2) from movies where studio=”Marvel Studios”

It gives us 2 decimal points as describe earlier 7.50 but now if we look at our header column in the result section i-e **round(avg(imdb\_rating),2)** which is too big so we can define our own custom header column or name as:

* Select ROUND(AVG(imdb\_rating),2) **as avg\_rating** from movies where studio=”Marvel Studios”

Now this is useful in scenarios like let’s say you want to print min max and average rating everything for marvel studio so the way you do that is:

* Select MIN(imdb\_rating) as **min\_rating**, MAX(imdb\_rating) as **max\_rating**, ROUND(AVG(imdb\_rating),2) **as avg\_rating** from movies where studio=”Marvel Studios”

By running this it will gives us all the rating with our custom names.

Now getting back to the original query:

* SELECT COUNT(\*) from movies where industry =’bollywood’;

Where we printed number of movies in bollywood, if we want to print the number of movies for Hollywood we can do exactly the same:

* SELECT COUNT(\*) from movies where industry =’hollywood’;

But what if we want to print both the numbers at the same time, I can have say five different industries in my movies table and I want to have industry name and the count something like this:

Hollywood: 18

Bollywood: 20

So how do we do that, so for this we have to use something called **group by** and the way group by works is:

* Select industry, count(\*) from movies Group By industry

So the group by clause will allow you to specify on what criteria you are grouping things.

Let’s say you want to group things by studio:

* Select studio, count(\*) from movies Group By studio

If we can see the result is in random order like 2, 5, 1 etc. But let’s say we want to sort the result based on this count 2, 5, 1 what we do is:

* Select studio, count(\*) as cnt from movies Group By studio order by cnt DESC

Cnt is just some name we can give any name to it.

Now let’s say we want to print the industry, the movie count and the average rating in that industry we can do that as:

* Select industry, count(industry) as cnt, avg(imdb\_rating) as avg\_rating, from movies group by industry;

We can round it to decrease the decimal points upto 2 as:

* Select industry, count(industry) as cnt, round(avg(imdb\_rating),2) as avg\_rating, from movies group by industry;

Let’s say we want to do the same thing for studio as earlier:

* Select studio, count(studio) as cnt, round(avg(imdb\_rating),2) as avg\_rating, from movies group by studio order by avg\_rating DESC;

In our csv file there are many data errors like we have some studio with null values and we want to skip that as:

* Select studio, count(studio) as cnt, round(avg(imdb\_rating),2) as avg\_rating, from movies where studio!=’’ group by studio order by avg\_rating DESC;

We do not need to memorize all these functions as there are so many functions what we need to learn is the art of googling. Googling is the skill that will help you rise in your career so let’s say we want to know all the SQL functions we can search in google search engine as: [MySQL Functions](https://www.w3schools.com/mysql/mysql_ref_functions.asp).

Art of googling is the key to success.

**KEY TAKEAWAYS**:

* Knowing summary analytics in SQL will enable you to perform AD HOC analysis which is an important business use case.
* MAX, MIN and average are the common summary analytics function of SQL.
* You can define a custom column header name by using AS clause.
* GROUP BY clause will help you to create a summary of metrics such as average, count etc. for selected columns.

**EXERCISE**:

**Summary analytics (MIN, MAX, AVG, GROUP BY)**

1. How many movies were released between 2015 and 2022  
2. Print the max and min movie release year  
3. Print a year and how many movies were released in that year starting with the latest year

**Chapter: 4**

**Having Clause**

Let’s say you want to print all the years where more than 2 movies were released how you would do that so this is done clearly by the group-by operation because you are grouping your rows based on the release year and you are getting a count so let’s write the query:

* SELECT release year, count(\*) from movies group by release\_year

By executing this we will get the count pretty straightforward we can also sort this by saying that:

* SELECT release year, count(\*) as movies\_count from movies group by release\_year order by movies\_count desc;

So in the result the 1st four rows shows where the movies count is greater than 2 but this query is showing us all the results let’s say you want to filter all the results so we should write that where my movies count is greater than 2 so we can probably use where i-e:

* SELECT release year, count(\*) as movies\_count from movies where movies\_count>2 group by release\_year order by movies\_count desc;

But when we run this query it’s not going to work it will through an error and the reason this is happening is the way SQL will execute all these operations is it will 1st execute from operation i-e from movies, then it will execute where then it will execute group-by

FROM🡪 WHERE🡪 GROUPBY

So now movies count is not available until group-by is executed but where is executed before Group By so when where is executed it doesn’t know what is movies count that is the reason it doesn’t work.

Luckily there is something very similar to where which get executed after group by and it is called having and in the end you execute order by:

FROM🡪 WHERE🡪 GROUPBY🡪Having🡪OrderBy

So let’s remake the query

* SELECT release year, count(\*) as movies\_count from movies group by release\_year **having** movies\_count>2 order by movies\_count desc;

So we need to remember this order:

FROM🡪 WHERE🡪 GROUPBY🡪Having🡪OrderBy

Having is used mainly with group-by operations so when you go and work in the industry remember this thing having is mainly used with group-by but it can be used without group-by too but main use-case is with group-by and another thing that is different with having and where is that when you have where let’s say I want to get all the movies whose imdb rating is greater than 7 in the where clause if I’m using a column that column doesn’t have to be in my select statement so let’s write the query:

* SELECT release year, count(\*) as movies\_count from movies **where** imdb\_rating>6 group by release\_year having movies\_count>2 order by movies\_count desc;

So it still works but in having clause whatever column we are referring to has to be there in the select statement.

For example:

* SELECT release year, count(\*) as **movies\_count** from movies group by release\_year **having** imdb\_rating>2 order by movies\_count desc;

So it will not work like this because imdb\_rating is not available in select statement.

Most of the time we will find that the column we are using in having clause will be an aggregated column for example:

* SELECT release year, count(\*) as **movies\_count** from movies group by release\_year **having** **movies\_count** >2 order by movies\_count desc;

Here the movies count is kind of aggregation or a derived column so these are the use-cases that we will see in the industry.

**KEY TAKEAWAYS**:

* The order of query execution in –SQL is FROM🡪 WHERE🡪 GROUPBY🡪Having🡪OrderBy.
* Group-By and HAVING clauses are often used together.
* The column you use in HAVING should be present in SELECT clause whereas WHERE clause can use columns that is not present in SELECT clause as well.

**Chapter: 4.1**

Calculated Columns

So far we looked into movies table, now we are moving to different table columns called actors and we will discuss calculated columns or derived columns when you look at the data in csv file actor’s sheet where we have actor name and birth year and for our analytical purpose we may want to get an age for each of these actors let say we want to do an analytics such as what an average age of an actor in bollywood or Hollywood or who is the youngest actor in these industries etc. Having age will be very helpful in that case and the way we derived the age is we will get a current year and then subtract birth year from that now to get current year in EXCEL there has to be some formula and I don’t know what that formula is so I can go and ask the google so we can just search excel get current year and you will find that formula and you can just use that.

Link: [excel get current year - Google Search](https://www.google.com/search?q=excel+get+current+year&sca_esv=f247609a43e21a90&sxsrf=ADLYWIJ-x5-cjic77PS-qN4rsuBKerNsEw%3A1732770862770&source=hp&ei=LvxHZ9ruLNethbIPyuOpyAs&iflsig=AL9hbdgAAAAAZ0gKPmwQ9eXAPtVqDv3pkeAgukXL06SC&ved=0ahUKEwjavYC-ov6JAxXXVkEAHcpxCrkQ4dUDCBc&uact=5&oq=excel+get+current+year&gs_lp=Egdnd3Mtd2l6IhZleGNlbCBnZXQgY3VycmVudCB5ZWFyMgUQABiABDIGEAAYFhgeMgYQABgWGB4yBhAAGBYYHjIGEAAYFhgeMgYQABgWGB4yBhAAGBYYHjIGEAAYFhgeMgYQABgWGB4yBhAAGBYYHkj5KVCAAliAKHABeACQAQCYAcsDoAG9PKoBBjMtMTYuNbgBA8gBAPgBAZgCFqAC8D2oAgrCAgcQLhgnGOoCwgIHECMYJxjqAsICChAjGIAEGCcYigXCAg4QABiABBiRAhixAxiKBcICERAAGIAEGJECGLEDGIMBGIoFwgILEAAYgAQYsQMYgwHCAggQABiABBixA8ICERAuGIAEGLEDGNEDGIMBGMcBwgILEC4YgAQYxwEYrwGYAxKSBwgxLjMtMTUuNqAH2YIB&sclient=gws-wiz)

=YEAR (TODAY ())-C2 by using this formula we will see the age of the actor’s.

Can we something similar in MySQL so now we will replace excel to sql in google search. See what you type in google is a very important skill, this is a skill that can give you super power so do not underestimate that.

Good programmer or data scientist or analyst they don’t remember the syntax, they know the art of how to search for the answers.

LINK: [MySQL YEAR () Function](https://www.w3schools.com/sql/func_mysql_year.asp)

Let say I get this article where there is a function called YEAR () which take data as an input and it will give you the year for example YEAR (“2024-11-24”) but I do not have I date, I want to give it the current data so how we could do that well it can be done like YEAR (CURDATE ()); and by using this function it will give us the current date.

Let try an SQL query for this:

* SELECT CURDATE()

When we rap year around it as:

* SELECT YEAR (CURDATE())

It gives us only the current year. Once we confirm that this returns the current year we should write the query:

* SELECT \* from actors;

What it does that it will return all the actor’s and in addition to that we want to do is:

* SELECT \*, YEAR (CURDATE()) from actors;

What this query does that it create a new column of current date and we can now find the age of an actor by:

* SELECT \*, YEAR (CURDATE())-birth\_year as age from actors;

Now this query will return the age of an actor as well as it will name the column header as age.

Getting age in SQL is easy as getting age in Excel by just writing as appropriate formula. Now let’s go to a different table and that table is financials.

Now we are going to use some more calculated columns for financials tables when we look at this table:

* SELECT \* from financials;

This table has movie\_id for example 101 index is KGF and in the future chapters we will look into how we can join 2 different tables using left join right join etc. But for now using ID we can figure out which movie is that like 102 if we can find in other tables so at number 2 is Doctor Strange and this movie made 954 million dollars at box office as of the date that data was captured and the budget for making this movie is 200 million dollars, so let’s say we want to calculate the profit for these different movies, so how do we do that, it is simple in excel as:

* =C2-B2

So we can do the same thing in SQL by using some formula so what we will say in MYSQL is:

* SELECT \*, (revenue-budget) as profit FROM financials;

So here \* means all the columns and if you want to print the selected columns you can specify the name separated by comma.

Now let’s say you want to print the revenue of all the movies into a single currency because right now these numbers that is in the profit column have multiple currencies like INR, Dollars etc. so we can’t compare these 2 currencies like for KGF and doctor strange in which one has INR and other has dollar currency and we cannot say that KGF made more profit than doctor strange. So if we want to do apple to apple conversion you have to convert everything into a single currency and everything into a single unit but that is a separate discussion for now, let’s say our purpose is to convert everything into INR, so how can we do that, well USD to INR currency conversion ratio right now is 84.50, it is dynamic that is it is changing every day. If you are working in an organization and if you are doing currency conversion you will use the live spot rate, the currency conversion rate as of the date when this data was captured.

So if we want to convert 954 million dollars into INR, we will say that 954 into 77. So how do we have a new column? That is revenue INR so I want to have a new column called revenue\_inr and have every revenue is in INR. So we will use something called the IF condition. The way it works is this:

* SELECT \*,
* IF (Currency=”USD”, revenue\*84.50,revenue) as revenue\_inr
* From financials;

In the IF section it means that in if take currency USD revenue and multiply it with 84 and if it is not USD then just print the revenue called INR.

So what happened is if the number currency is INR then it is keeping its same but if it is USD then it is multiplying it by 84.5.

So the way this IF condition works is in IF the first thing we have is condition, the second condition we have is what we want to do if the condition is TRUE and the third thing is what we want to do if the condition is FALSE.

* IF (condition, True, False)

Now let’s print all the revenue numbers into millions currency, ideally if you want to find out which movie made the maximum money at box office worldwide and you want to compare like which movie made the most money in that case you have to convert revenue into a single currency and then into a single unit and that will make our query little complicated since we have learned currency conversion, now we will do unit conversion so for unit conversion what we have to do is to find out how many units do we have. So for that the query is:

* Select distinct unit from financials;

It will return the number of unique values that this column has so it has returned 3 numbers billion, million, thousands, now millions is a very popular currency so what we will do is to convert everything into a million so we will have some condition and we will print revenue in millions because that is very popular. How do we convert billions into millions, so to convert billions into millions? Whatever number we have let’s say we have 12 billion what will be the millions? It will be 12000 which means you are multiplying 12 with 1000. So the condition is if the value is in billions, you multiply your revenue with thousand but if it is thousands let’s say you have these many thousands you divide that by this many thousands mean this is the actual number so these many thousands is actually 4.567.

* Billions🡪12🡪12000🡪12\*1000
* Billions🡪rev\*1000
* Thousands🡪4567🡪4.567mln

So the formula that we will use here is:

* Thousands🡪 revenue/1000
* Million🡪 revenue

This is like a simple math, so if condition is useful when you have binary condition previously we had IF USD then do this otherwise do that but here we have more than 2 conditions to check if billions then do this, if thousands do this and if the unit is in million then just keep things as it is so how do we express that, we can do this using something called as case statement now how does case statement works.

Link: [MySQL CASE Function](https://www.w3schools.com/sql/func_mysql_case.asp)

So the way case statement works is for example:

SELECT OrderID, Quantity,  
CASE  
    WHEN Quantity > 30 THEN "The quantity is greater than 30"  
    WHEN Quantity = 30 THEN "The quantity is 30"  
    ELSE "The quantity is under 30"  
END  
FROM OrderDetails;

So let’s write our query:

* Select \*,
* **Case**
* WHEN unit=’Thousands’ THEN revenue/1000
* WHEN unit=’Billions’ THEN revenue\*1000
* WHEN currency=millions THEN revenue or ELSE revenue ( use either ELSE or When)
* **END** as revenue\_mln
* From financials;

**KEY TAKEAWAYS**:

* You can derive new columns from existing columns in a table.
* As a data analyst, Revenue and Profit are the most common metrics that you will calculate in any industry.
* Currency conversion and unit conversion are important business use cases of SQL.
* IF function is often used in SQL queries.
* When you have more than 2 conditions, you need to use CASE and END function instead of IF function.

EXERCISE:

Write SQL queries for the following,

1. Print profit % for all the movies

**Chapter: 5**

**SQL Joins**

(Inner, left, right, full-outer join)

(Theoretical part)

In our movies database we had multiple tables and in this chapter we will be discussing why we need multiple tables in first place. We will be using a different dataset of grocery store customer transactions so when you go to a grocery store and buy things they have a POS system and in that system you could have transactions and in a csv file it might have multiple columns like transaction date, item name, category, price, name etc. Now you can have all the data in one place just like in CSV file where every single information that is needed for a given transaction is present in one record and this work fine but this has some disadvantages:

* **DISADVANTAGES:**
* **1: DATA DUPLICATION or REDUNDANCY**: Information is repeated like name of customer etc. Like for Hassam if he has 1000 records, it will repeating that information like repeated Hassam for 1000 times.
* **2: Updates are difficult**: Tomorrow let say potato price is go up and now you want to change it from 60-100 and if you have another records you have to update all of them for example broccoli price change from 120 to 150 tomorrow and we cannot change it for per transaction.
* **3**: **Data is not organized properly**: So let’s look at how it will look if the data is organized into multiple tables so I will first take customer information and put it in a customer table so there is customer name, email, address etc. in customer sheet or table and we will give a unique id for that record example 401,402 etc.
* We will do the similar thing for items table too like for item name, category, price we will have item id example 21, 22 etc.
* Once we will have these things our sales table will look very clean, it will have item id, customer id, weight etc. so if we look into item id like 21 so what is that record, so 21 id is white potatoes etc. and 401 is Hassam.
* So by combining all these information we can save a lot of space in our database also the data organization is much better.
* When you want to check prices for items you just go to items table and that is it.
* Let’s say a customer came and say Oh! I changed my email, so then we will go to customer table and this table will have very less records so you can just change email id but if this customer information is in mail sales table which can have thousands of records changing email id or any record in that table might be painful process.

When we talk about our movies dB exactly same principles are applied, let say I want to have all the information in one table, one of the information I have is actors for example in movies KGF there are 2 actors YASH and Sanjay Dut and I found I want to put that information there in that table so how do we put that, so we can do put one actor in excel in each row and column like YASH and his birth year but we can’t put 2nd actor information in that same column for the same movie so should we create another column for 2nd actor but that is not good because we have more actors, so then by doing such steps information organization becomes difficult, if everything is in one table for that reason we have that concept of multiple tables and specially for actors what we do is actor to movies relation is many to many which means one movie can have multiple actors and one actor can participate in multiple movies therefor we created a separate table called movie actor this is like a mapping i-e movie id and actor id and for movie id 101 we can have then multiple actors like 50 and 51 so this table will allow you to do the mapping and we will see like in query section that print me all the movies along with their actor names and in order to do that we will be using something called **joins**.

**KEY TAKEAWAYS**:

* Companies use multiple tables to store data.
  + To save space by avoiding repetition.
  + Organize data better.
  + Make updates easily.
* In SQL you will be able to map multiple tables using JOIN clause

**Chapter: 5.1**

**SQL Joins**

(Inner, left, right, full-outer join)

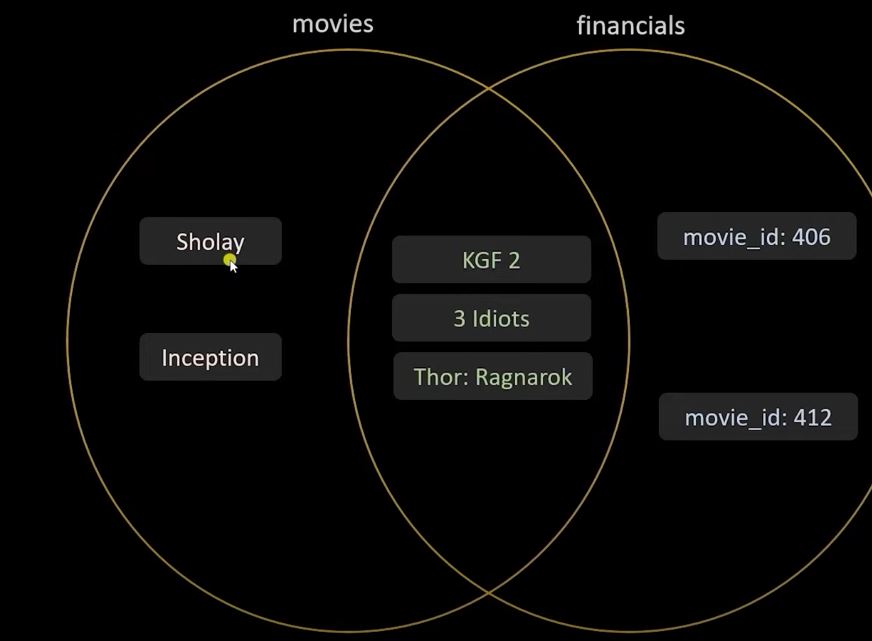
(Practical part)

Let’s say a business manager comes and ask you to generate a report where there is a movie title and there is a budget revenue unit etc. columns. Now this requires joining 2 tables.

We have movies table which has movies titles and we have financial tables which has financial details and the common link which joined these 2 is movie id for example movie id 101 which is KGF2 so using this movie id we are going to join these 2 tables and generate a report and we will be looking into different types of join functions so let’s start:

* Inner join:
* Select
* select
* movies.movie\_id, title, budget, revenue,currency,unit
* from movies
* join financials
* on movies.movie\_id=financials.movie\_id;

The join here we perform is by default inner join. So what is an inner join, by for example:



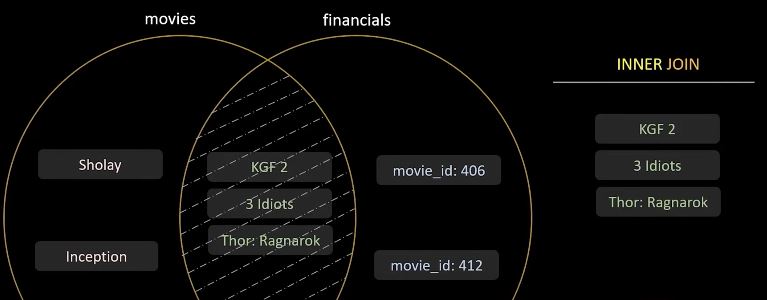
Let say you have these 2 tables and these movies Sholay and inception, they are not available in financial table so they have some movie IDs but those movie IDs are not available in financial table similarly financial table has some records like 406 412 etc. these are not available in our movie table and you will actually this is a case, let see movies table:

* select \* from movies;

Here we have this movie 112 inception and if you look at financial table:

* select \* from financials;

We do not find record called 112 maybe because of data collection error or some other data problem due to which that record is not available and this can happen in real life dB similarly the other record 106 is also not available, so there are some records in the movie table that are not available in the financial table and there are some records in financial table that are not available in the movie table.



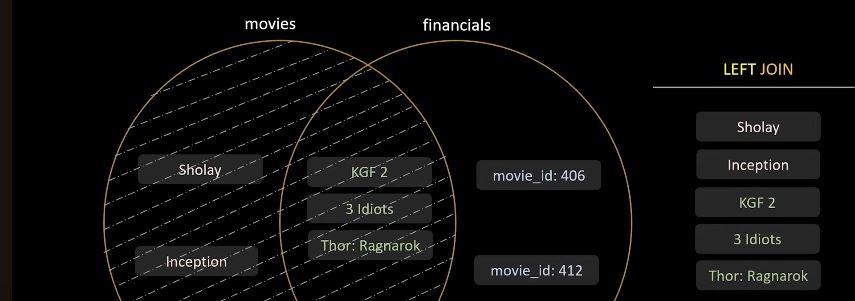
So now when you do inner join what happens is it will only take into account the common records, it is like when your Venn diagrams, it is an intersection and that is what exactly happens when we run the inner join query. By running this we will not be able to see any movies like Sholay or inception because it is present in the left table and not in the right table similarly in financials table which is my right table and I have 406 and 412, which are also unavailable due to the left table.

What if I want to see all the titles and I want to see Sholay and inception even if doesn’t have budget or revenue and if you want to do that you will do something called left join.

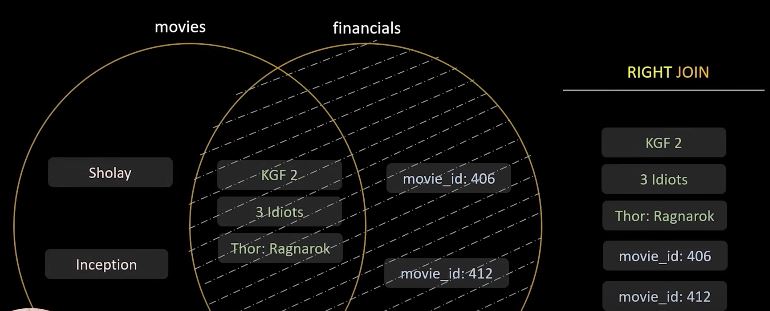
* left join
* select
* m.movie\_id, title, budget, revenue,currency,unit
* from movies m
* left join financials f # it is by default inner join
* on m.movie\_id=f.movie\_id;

Left means when you say from something that becomes your left table and the 2nd one which is financials has become the right table and when you run that query you will notice that there is Sholay and inception.

So pictorial representation for left join is:



Similarly in right join:



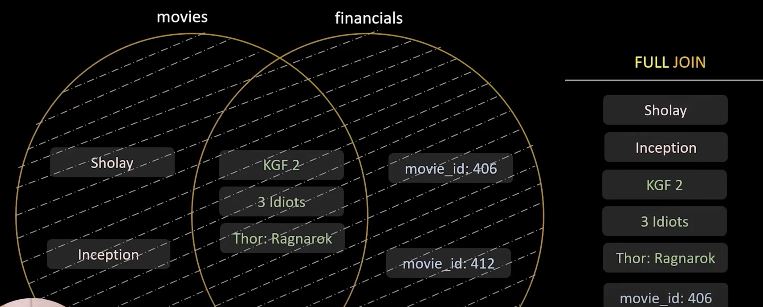
We might guess what will be happened if we do a right join, so in right join we will see the records from the right table which is financials and not see the records from left table which is movie table:

* right join
* select
* m.movie\_id, title, budget, revenue,currency,unit
* from movies m
* right join financials f # it is by default inner join
* on m.movie\_id=f.movie\_id;

It printed 406, 412 but didn’t print movie id because here we are saying m.movie id and we have to get movie id from movies table so for that we should just change m to f like:

* right join
* select
* f.movie\_id, title, budget, revenue,currency,unit
* from movies m
* right join financials f # it is by default inner join
* on m.movie\_id=f.movie\_id;

The final join we have is the outer join.



In final join we will get all the records and the way we do that in MYSQL is that there is nothing like OUTER join, so if we do write outer in the query it will show an error and by the, this is the syntax for MYSQL but if you are using oracle or SQL server the syntax might be different. SQL is the standard language that we use for different databases and these different databases have a slight tweak in terms of syntax so we need to keep that in mind. So what do we do now, so when we did this?

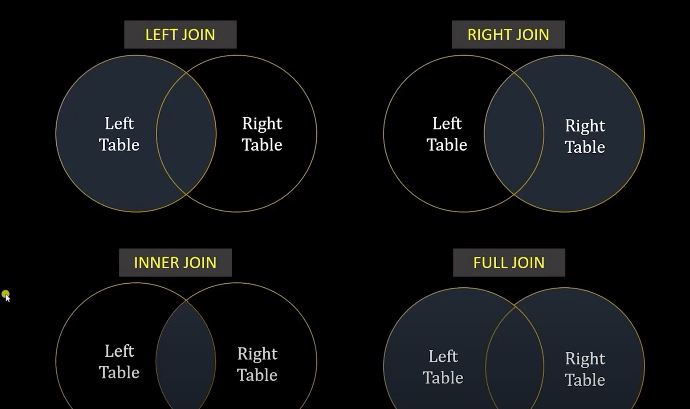
* left join
* select
* m.movie\_id, title, budget, revenue,currency,unit
* from movies m
* left join financials f # it is by default inner join
* on m.movie\_id=f.movie\_id;
* right join
* select
* f.movie\_id, title, budget, revenue,currency,unit
* from movies m
* right join financials f # it is by default inner join
* on m.movie\_id=f.movie\_id;

In left join we see certain records are available but that are not available for right join so now let’s call UNION on these 2 so we had these 2 queries in between the query and when we write this keyword called union.

Union affairs set theory Union is just the union of those two rows. When we use union keyword we have to make sure that the number of columns and their column names are same and if not then it will give us an error.

* # full join
* 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;

When we run this what we see that it shows all the records from left and right hand side of the tables and this is how in MYSQL we perform the full join.



The combination of these 3 Full, Left and Right join is also called the outer join, so whatever is not the inner join is also called as the outer join. So we can say that left outer or right outer or full outer join.

And MYSQL support these keywords like right outer join or left outer join and that will give you the same result as left and right join.

One last thing is something called USING clause so here most of the time we will be using on because in many use cases this column name let’s say f.movies\_id can be f.mid which can explicitly use when the column names are not defined but in our case both the column names are defined like movie\_id and financial\_id so we can do that as USING (movie\_id).

Exercise:

Write SQL queries for the following,

1. Show all the movies with their language names  
2. Show all Telugu movie names (assuming you don't know the language  
id for Telugu)  
3. Show the language and number of movies released in that language