Difference between count(\*) & count(column\_name)

* Count(\*) include all rows with null values
* Count(column\_name) include only those rows those holds data

Sum(column\_name)

* Returns the added sum of values for a group of rows.

Avg, sum, max, min - These only work if columns contain numbers! & returns only one row.

Group By

* GROUP BY will group together the cost for all movies with similar genres.
* We can use the GROUP BY clause to condense a group of columns into a single row.
* SELECT column\_name, aggregate\_function(column\_name)

FROM table\_name

GROUP BY column\_name;

* SELECT genre, sum(cost)

FROM Movies

GROUP BY genre;

Using the HAVING Clause

* We can use the HAVING clause to only include genres that have more than 1 movie.
* SELECT genre, sum(cost)

FROM Movies

GROUP BY genre

HAVING COUNT(\*) > 1;

* More than 1 row per group
* Having will use after group by clause & it will be applied after grouping the rows.
* The HAVING clause restricts the groups of rows to only those who meet the specified condition.
* SELECT column\_name, aggregate\_function(column\_name)

FROM table\_name

WHERE column\_name operator value (optional)

GROUP BY column\_name

HAVING aggregate\_function (column\_name) operator value;

Using WHERE With the HAVING Clause

* We can use WHERE to filter individual rows before the application of GROUP BY.
* SELECT genre, sum(cost)

FROM Movies

WHERE cost >= 1000000

GROUP BY genre

HAVING COUNT(\*) > 1;



Why Use Constraints?

* The default behavior of database tables can be too permissive.
* Constraints can help with these shortcomings!
* Prevent NULL values
* Ensure column values are unique
* Provide additional validations

Adding the UNIQUE Column Constraint

* The UNIQUE constraint uniquely identifies each field in a table.
* CREATE TABLE Promotions

(

id int

name varchar(50)

category varchar(15)

NOT NULL UNIQUE,

);

* More than 1 constraint can be used on a column!
* Cannot insert duplicate values with UNIQUE constraint.

Making a Column a Primary Key

* Adding a PRIMARY KEY constraint means that column cannot be NULL and must be UNIQUE.
* A Primary Key Prevents Duplicate Entries
* A Primary Key Prevents NULL Values

Difference between PK and NOT NULL + UNIQUE

* A PRIMARY KEY constraint automatically accomplishes the same goals of both the UNIQUE and the NOT NULL constraint. However, it’s not the same thing.
* Primary key can only be defined once per table
* NOT NULL + UNIQUE can be used multiple Can only be defined once per table times per table

Foreign Key

* A foreign key is a column in 1 table that references the primary key column of another table.

Creating a FOREIGN KEY Constraint

* The REFERENCES keyword can be used to make a FOREIGN KEY constraint.
* The table being referenced must be created first.

Preventing Inconsistent Relationships

* The foreign key in the second table must match a primary key in the table being referenced.

Querying for Relationship Data

* A sample query which will collects data from promotions via relationship by foreign key which is the primary key of movies table

SELECT NAME,category

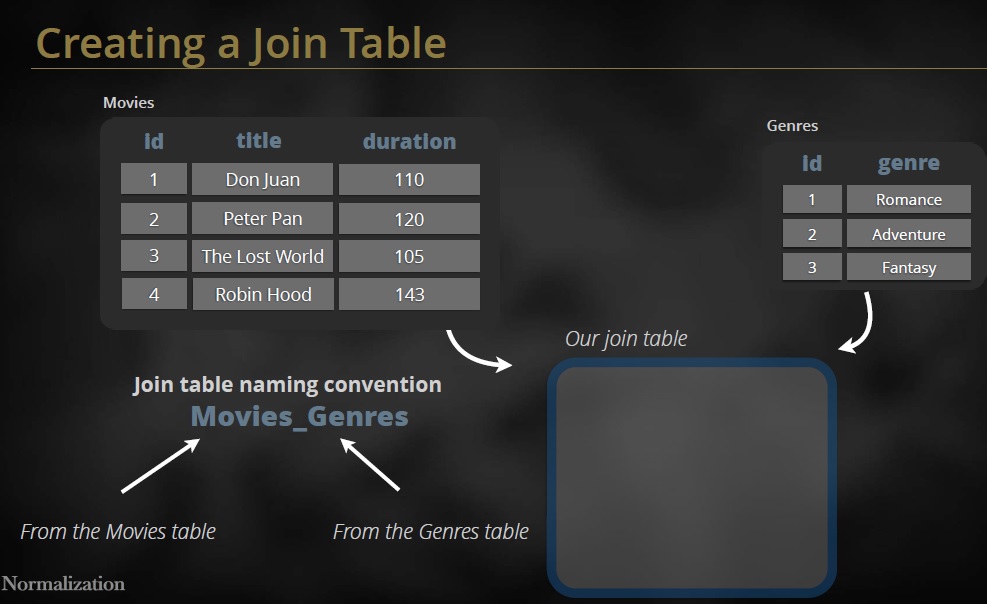
FROM promotions

WHERE movie\_id = 2;

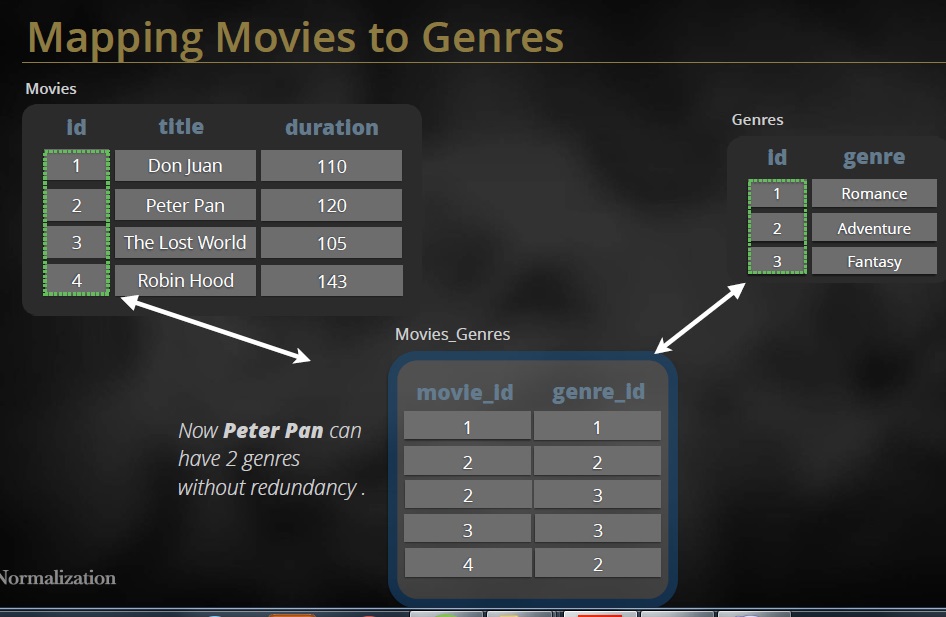
Normalization

* Normalization is the process of reducing duplication in database tables

Join Table



* Join Table naming convention Firsttable\_SecondTable like Movies\_Genres

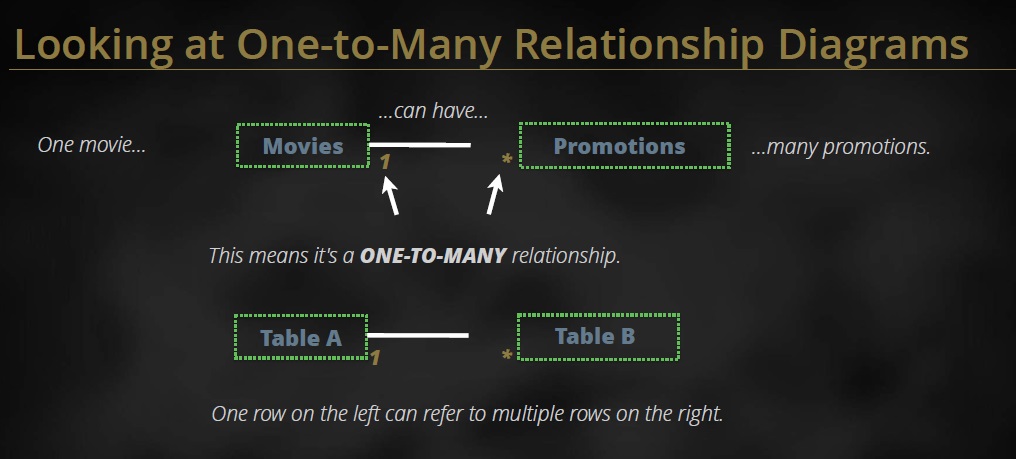


* Here movie\_id and genre\_id are the foreign of their corresponding table
* Both are references primary key of their corresponding table.

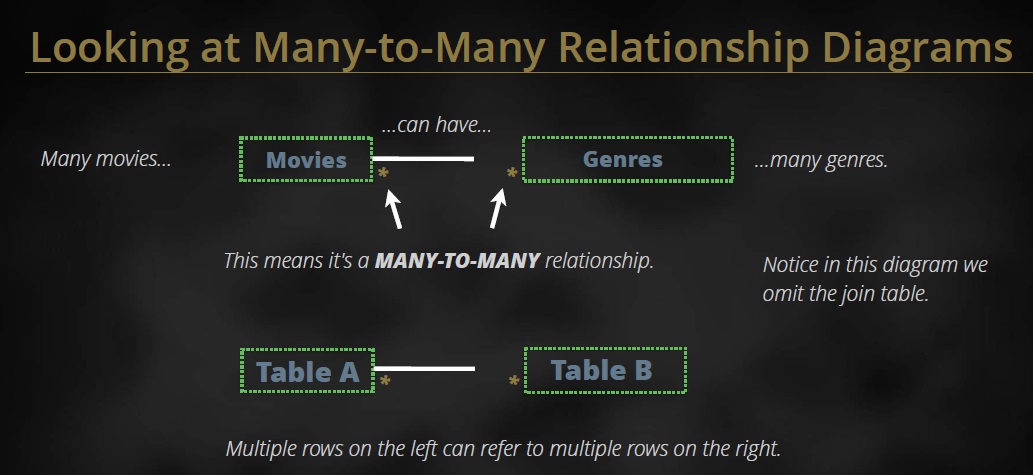
3 Different Types of Table Relationships

* One-to-one
* One-to-many
* Many-to-many

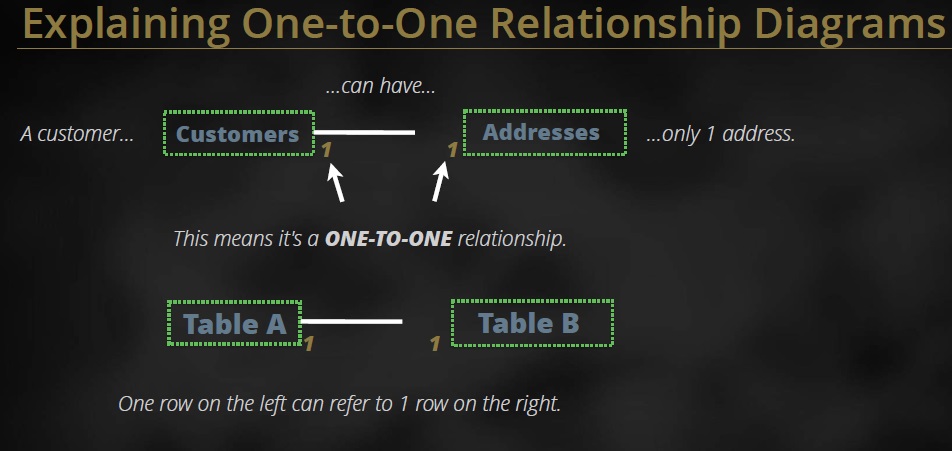
One-to-Many relationships



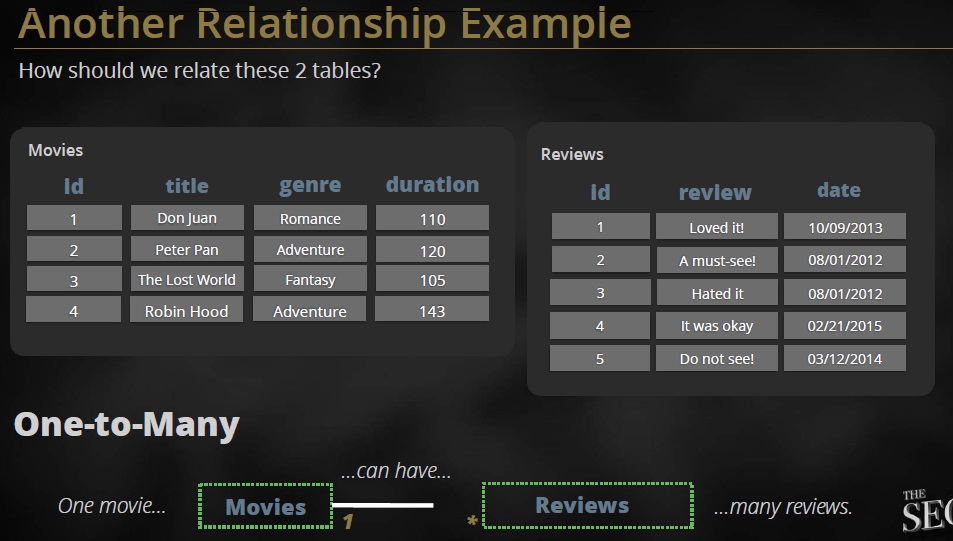
Many-to-Many Relationships



One-to-One Relationship



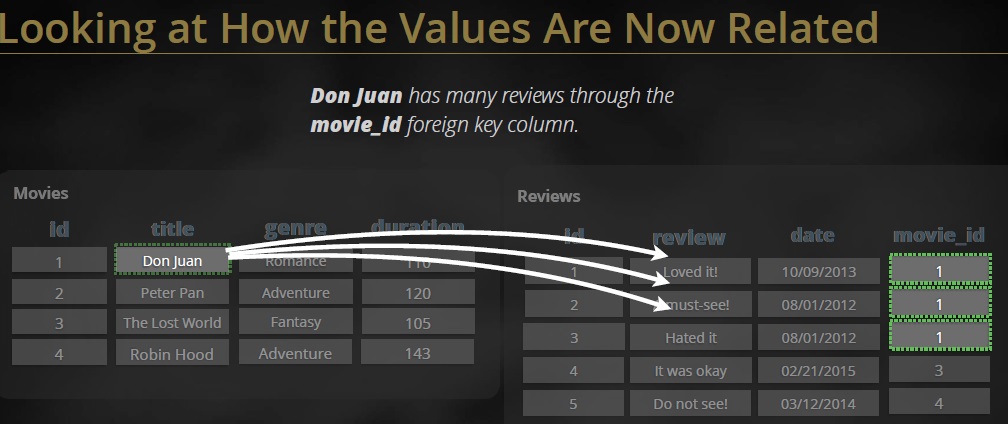
One-to-Many Relationships



Adding a Linking column



Looking at how the values are now related

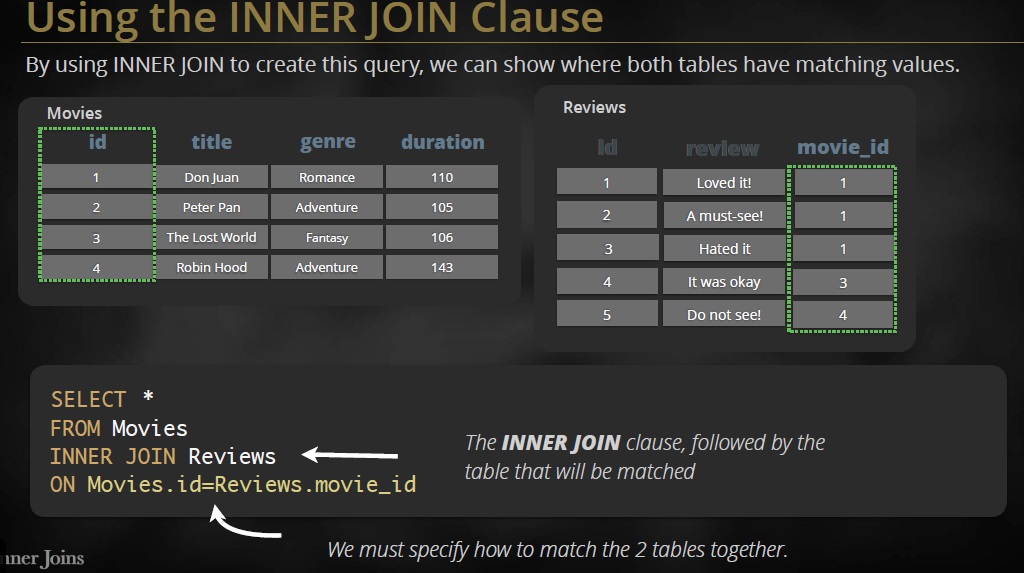


Using the Primary and Foreign key to join

* Two tables can be joined by the primary and a foreign key.

Using the Inner Join Clause

* By using inner join to create this query, we can show where both tables have matching values.



Following Query

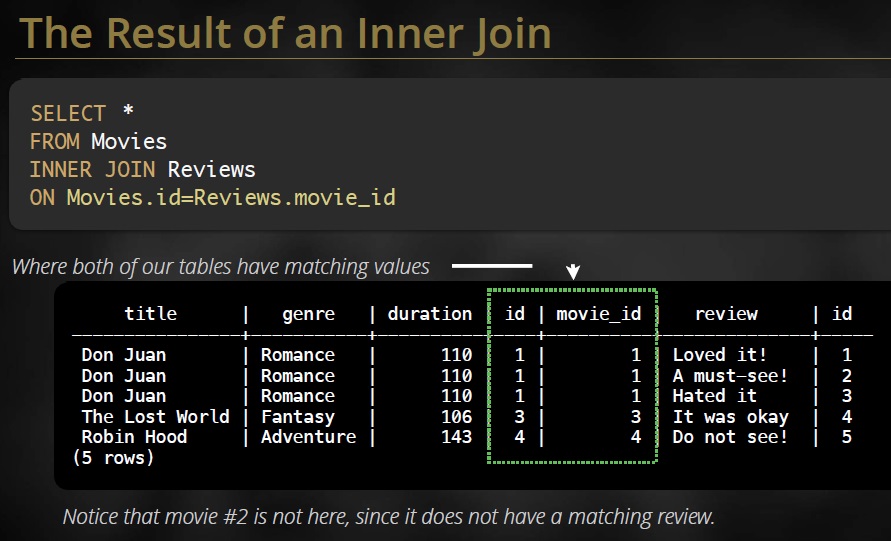
SELECT \*

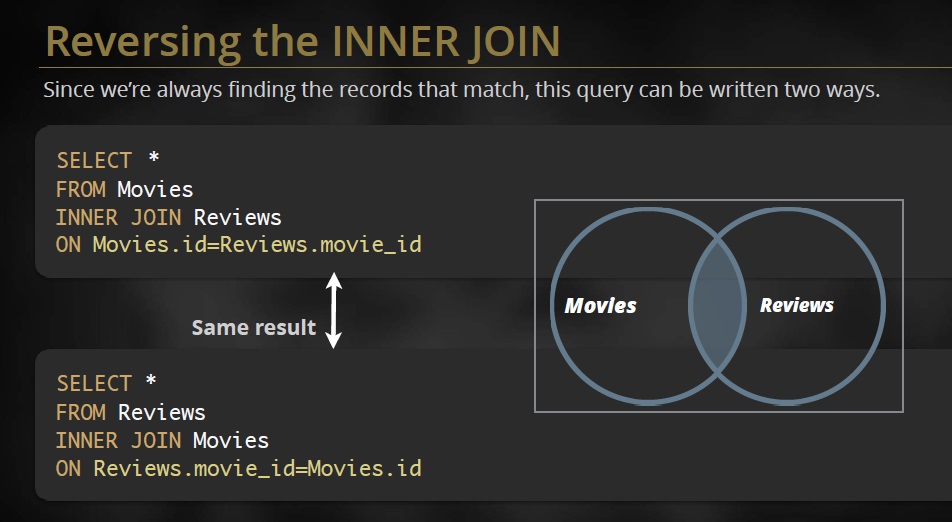
FROM movies

INNER JOIN Reviews

ON movies.id=Reviews.movie\_id;

* Return all columns from both table those are matching in both tables by on condition



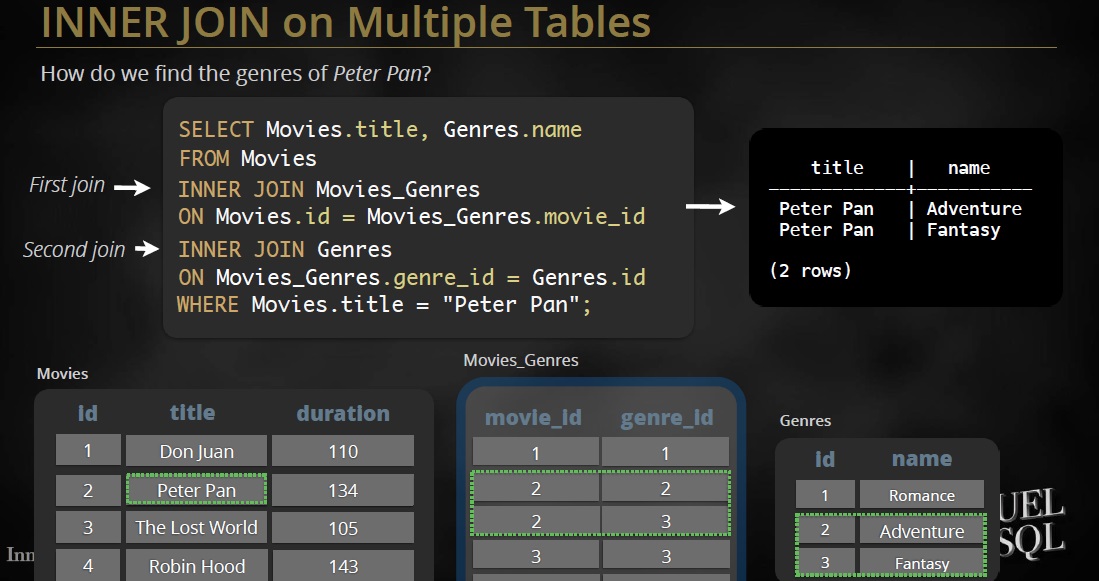


* Reversing the inner join produce same results difference is just that from table column values show first & inner table columns show after first table.



* For selective columns using following queries





**SQL JOIN**

An SQL JOIN clause is used to combine rows from two or more tables, based on a common field between them.

INNER JOIN

* Conditions: categories.categoryID = products.categoryID
* For each row in the products table, the query finds a corresponding row in the categories table that has the same categoryid.If there is a match between two rows in both tables, it returns a row that contains columns specified in the SELECT clause i.e., product id, product name and category name; otherwise,it checks the next row in products table to find the matching row in the categories table. This process continues until the last row of the products table is examined.

INNER JOIN SQL Format

* SELECT

productID,

productName,

categoryName,

companyName AS supplier

FROM

products

INNER JOIN

categories ON categories.categoryID = products.categoryID

INNER JOIN

suppliers ON suppliers.supplierID = products.supplierID

INNER JOIN Example Movies & Genres

SELECT movies.title, Genres.name

FROM Movies

INNER JOIN

Movies\_Genres ON Movies.id = Movies\_Genres.movie\_id

INNER JOIN

Genres ON Movies\_Genres.genre\_id = Genres.id

WHERE movies.title = "Peter Pan";

* Inner join implicitly mean that applied and operation