MySQL
Structured Query Language (1970's IBM)

- Core idea on Relational Algebra and hence Relational Database
- Industry standard way to query/obtain/add/delete/modify a data

SQL is not a general purpose programming language (C/C++/Java/Python)
SQL is a domain specific language.
SQL is a declarative programming language.

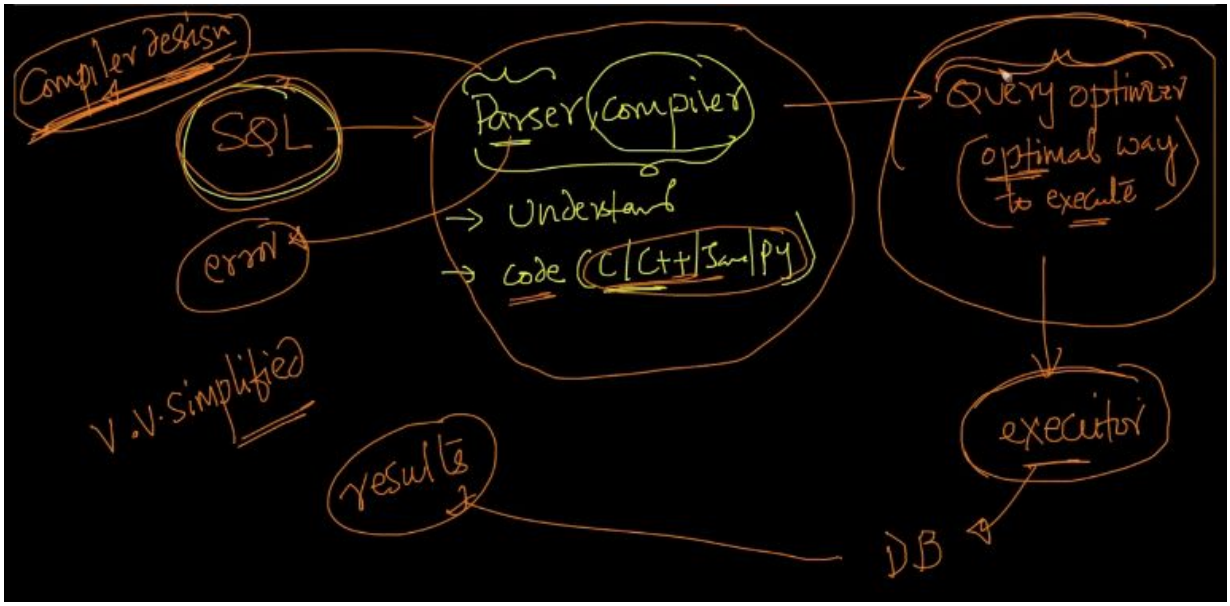
SQL queries describe the desired results without listing commands or steps that must be performed. It is a declarative programming language .

We don't have any if-else and for loops. We exactly command the required result.

SELECT CITY,STATE
FROM STATION

<u>It is simple but a powerful and popular tool.</u>

How does SQL statement get executed to get the solution that we desired?
Use of Parser and Compiler
Parser and Compiler take the SQL query and understand what is there in the SQL query and compiles into a procedural program.Query optimizer = optimal way to execute the query



To run database in terminal:

my(base) Mys-MacBook-Air:~ Royal$ mysql -u root -p
Enter password:

USE imdb; ----> Use imdb database
SHOW TABLES; ---> shows the table of database
DESCRIBE movies; ----> DESCRIBE keyword is more often used to obtain information about table structure.

ROW ORDER PRESERVATION = SIMPLE SELECT QUERY

```
SELECT name,year FROM movies;
SELECT rankscore,name FROM movies;
```

Data Sorted
By descending order

```
SELECT name,rankscore,year FROM movies ORDER BY year DESC LIMIT 10;
```

NULL => doesnot-exist/unknown/missing

```
# "=" does not work with NULL, will give you an empty result-set.
SELECT name,year,rankscore FROM movies WHERE rankscore = NULL;


SELECT name,year,rankscore FROM movies WHERE rankscore IS NULL LIMIT 20;

SELECT name,year,rankscore FROM movies WHERE rankscore IS NOT NULL LIMIT 20;
```

LOGICAL OPERATORS:

AND, OR, NOT, ALL, ANY, BETWEEN, EXISTS, IN, LIKE, SOME

```
SELECT name,year,rankscore FROM movies WHERE rankscore>9 AND year>2000;


SELECT name,year,rankscore FROM movies WHERE NOT year<=2000 LIMIT 20;


SELECT name,year,rankscore FROM movies WHERE rankscore>9 OR year>2007;
```

ANY and ALL requires sub-queries

#inclusive: year>=1999 and year<=2000
SELECT name,year,rankscore FROM movies WHERE year BETWEEN 1999 AND 2000;

```
mysql> SELECT name,year,rankscore FROM movies WHERE year BETWEEN 2001 AND 2000;
Empty set (0.12 sec)
```

When low value > high value, you get an empty set.

#lowvalue <= highvalue else you will get an empty result set
SELECT name,year,rankscore FROM movies WHERE year BETWEEN 2000 AND 1999;

 # same as genre='Comedy' OR genre='Horror'
SELECT director_id, genre FROM directors_genres WHERE genre IN ('Comedy','Horror');

****

HAVING:

Print years which have >1000 movies in our DB [Data Scientist for Analysis]

SELECT year, COUNT(year) year_count FROM movies GROUP BY year HAVING year_count>1000;
specify a condition on groups using HAVING.

Order of execution:
1. GROUP BY to create groups
2. apply the AGGREGATE FUNCTION
3. Apply HAVING condition.

often used along with GROUP BY. Not Mandatory.

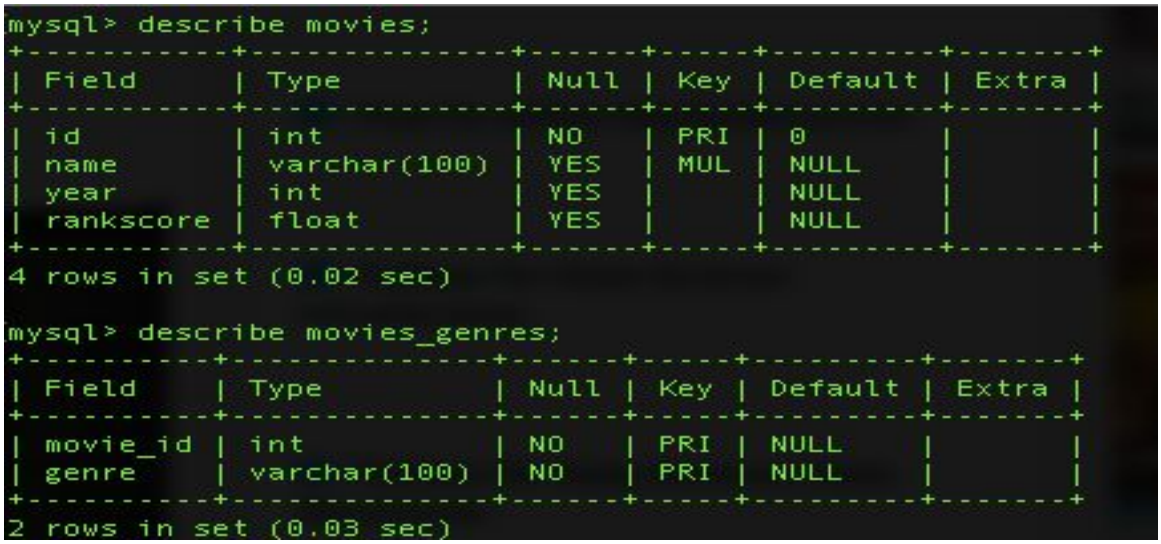
SELECT name, year FROM movies HAVING year>2000;
HAVING without GROUP BY is same as WHERE

SELECT year, COUNT(year) year_count FROM movies WHERE rankscore>9 GROUP BY year HAVING year_count>20;

HAVING vs WHERE
WHERE is applied on individual rows while HAVING is applied on groups.
HAVING is applied after grouping while WHERE is used before grouping.

****

JOINS



```
mysql> describe movies;
+-----------+---------------+------+-----+---------+-------+
| Field     | Type          | Null | Key | Default | Extra |
+-----------+---------------+------+-----+---------+-------+
| id        | int           | NO   | PRI | 0       |       |
| name      | varchar(100)  | YES  | MUL | NULL    |       |
| year      | int           | YES  |     | NULL    |       |
| rankscore | float         | YES  |     | NULL    |       |
+-----------+---------------+------+-----+---------+-------+
4 rows in set (0.02 sec)

mysql> describe movies_genres;
+----------+---------------+------+-----+---------+-------+
| Field    | Type          | Null | Key | Default | Extra |
+----------+---------------+------+-----+---------+-------+
| movie_id | int           | NO   | PRI | NULL    |       |
| genre    | varchar(100)  | NO   | PRI | NULL    |       |
+----------+---------------+------+-----+---------+-------+
2 rows in set (0.03 sec)
```

Join :combine data in multiple tables

For each movie, print name and the genres
SELECT m.name, g.genre from movies m JOIN movies_genres g ON m.id=g.movie_id LIMIT 20;

```
mysql> SELECT m.name, g.genre FROM movies m JOIN movies_genres g ON m.id=g.movie_id LIMIT 20;
+-----------------------------------+-------------+
| name                              | genre       |
+-----------------------------------+-------------+
| #7 Train: An Immigrant Journey, The | Documentary |
| #7 Train: An Immigrant Journey, The | Short       |
| $                                 | Comedy      |
| $                                 | Crime       |
| $1,000 Reward                     | Western     |
| $1,000,000 Duck                   | Comedy      |
| $1,000,000 Duck                   | Family      |
| $10,000 Under a Pillow            | Animation   |
| $10,000 Under a Pillow            | Comedy      |
| $10,000 Under a Pillow            | Short       |
| $100,000                          | Drama       |
| $100,000 Pyramid, The             | Family      |
| $1000 a Touchdown                 | Comedy      |
| $20,000 Carat, The                | Crime       |
| $20,000 Carat, The                | Drama       |
| $20,000 Carat, The                | Short       |
| $21 a Day Once a Month            | Animation   |
| $21 a Day Once a Month            | Short       |
| $2500 Bride, The                  | Drama       |
| $2500 Bride, The                  | Romance     |
+-----------------------------------+-------------+
20 rows in set (0.00 sec)
```

 # table aliases: m and g

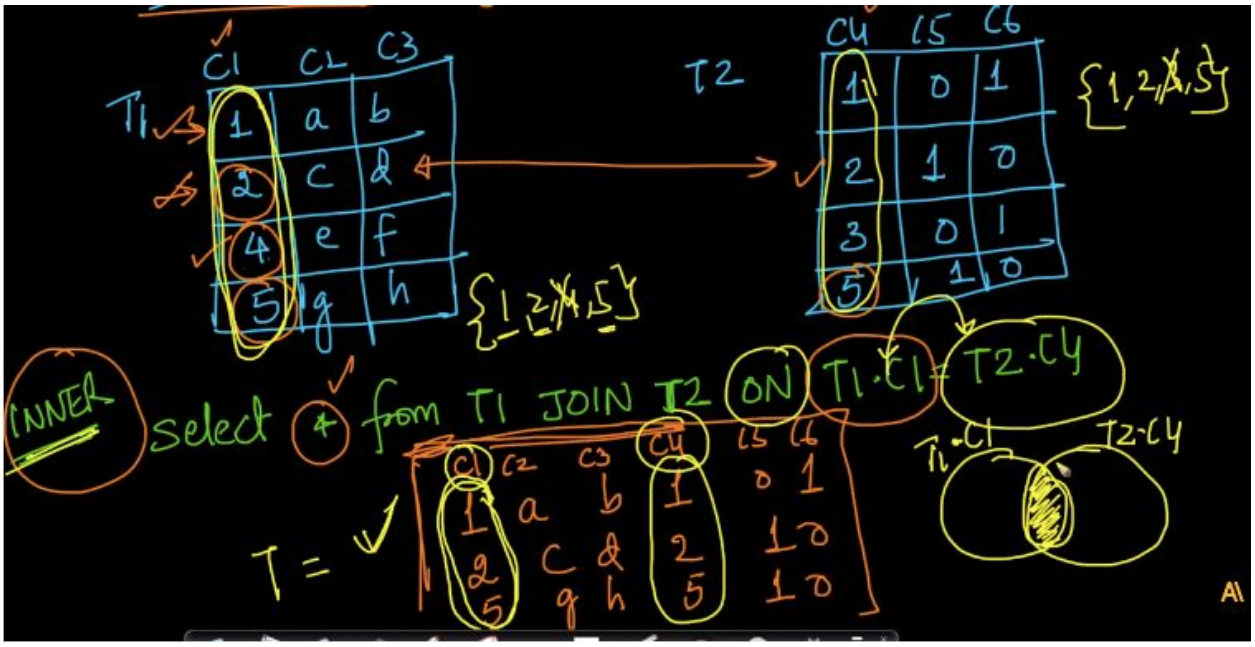
natural join: a join where we have the same column-names across two tables.
#T1: C1, C2
#T2: C1, C3, C4

SELECT * FROM T1 JOIN T2;

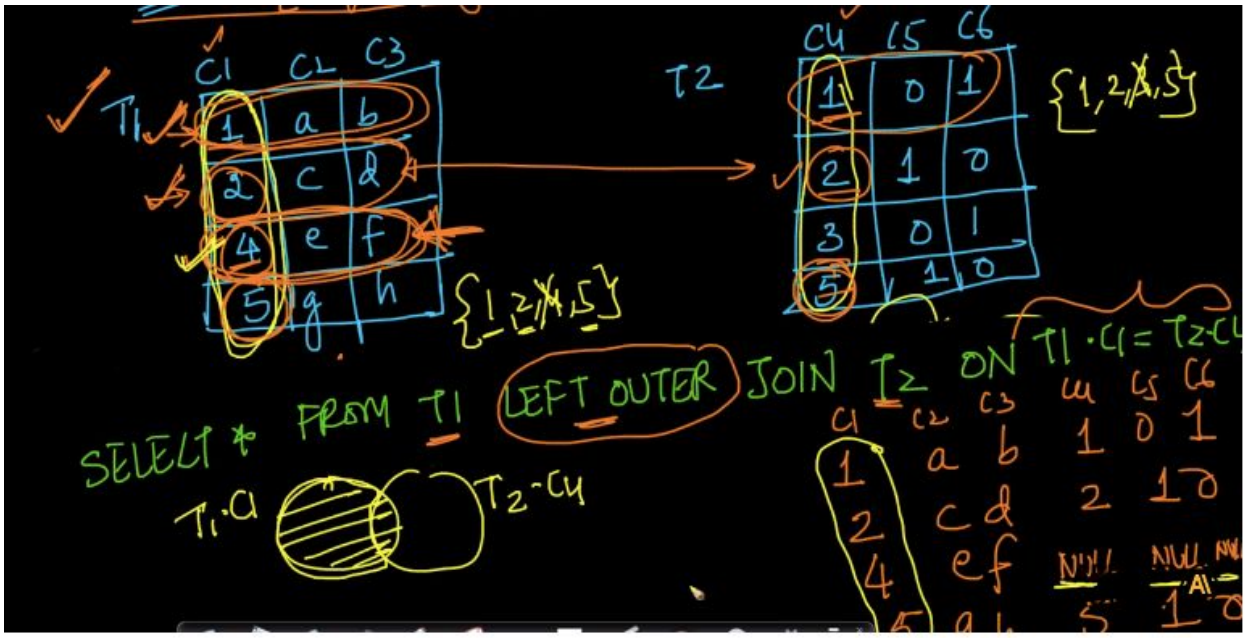
SELECT * FROM T1 JOIN T2 USING (C1);

returns C1,C2,C3,C4
no need to use the keyword "ON"

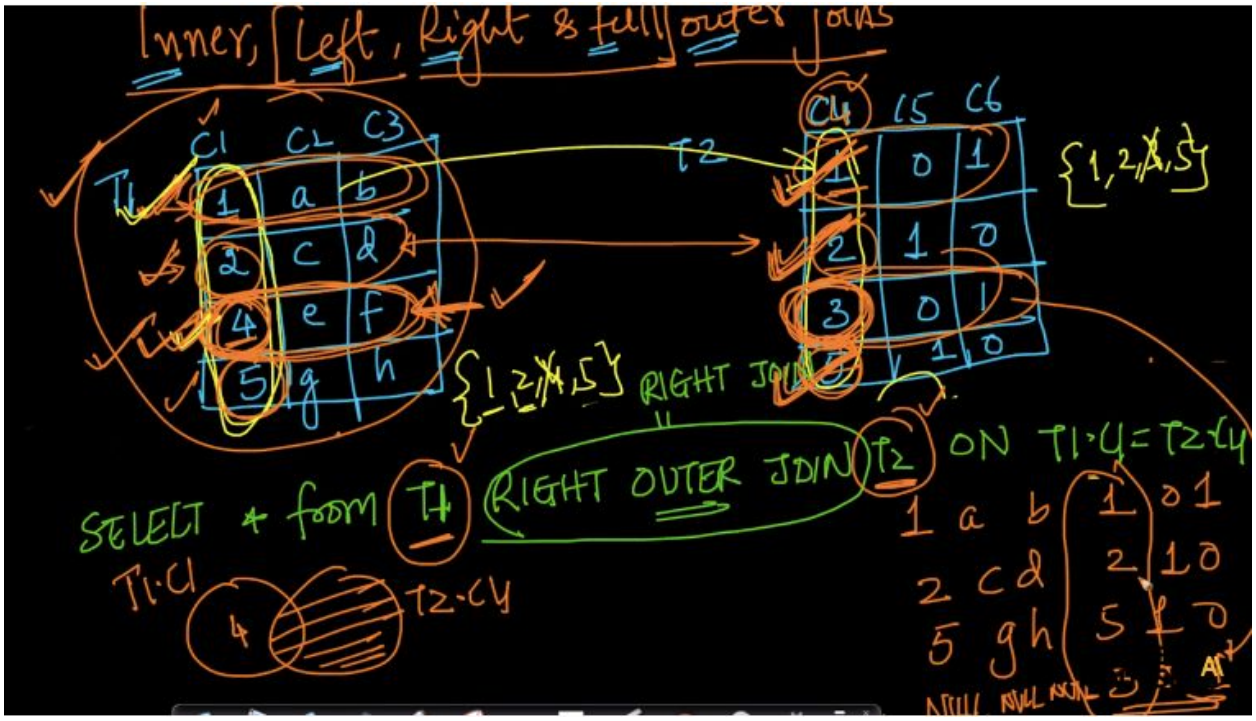
INNER JOIN



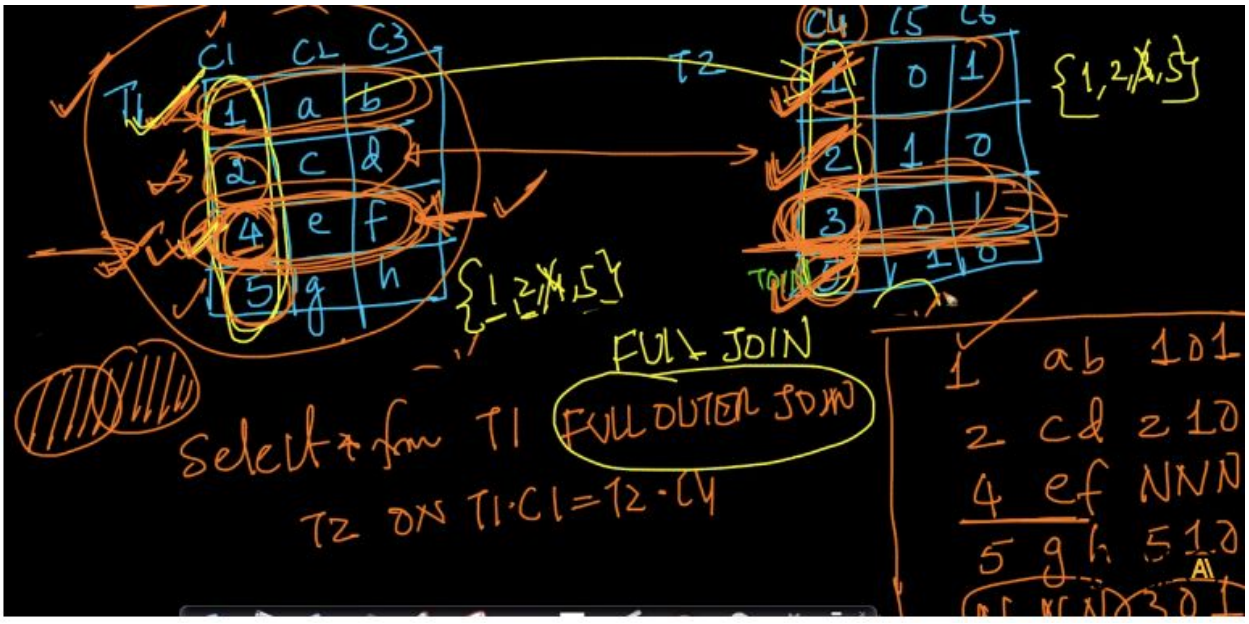
LEFT OUTER JOIN



RIGHT OUTER JOIN



Full join/full outer join



Incase you use mysql

```
SQL> SELECT  ID, NAME, AMOUNT, DATE
   FROM CUSTOMERS
   LEFT JOIN ORDERS
   ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID
UNION ALL
   SELECT  ID, NAME, AMOUNT, DATE
   FROM CUSTOMERS
   RIGHT JOIN ORDERS
   ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID
```

DATA Manipulation Language

SQL- DML,DDL,DCL

DML helps to manipulate data: SELECT,INSERT,UPDATE,DELETE

INSERT INTO movies(id, name, year, rankscore) VALUES (412321, 'Thor', 2011, 7);

INSERT INTO movies(id, name, year, rankscore) VALUES (412321, 'Thor', 2011, 7), (412322, 'Iron Man', 2008, 7.9), (412323, 'Iron Man 2', 2010, 7);

INSERT FROM one table to annother using nnested sub query:
https://en.wikipedia.org/wiki/Insert_(SQL)#Copying_rows_from_other_tables

 # UPDATE Command
UPDATE <TableName> SET col1=val1, col2=val2 WHERE condition

UPDATE movies SET rankscore=9 where id=412321;

Update multiple rows also.
Can be used along with Sub-queries.

#DELETE

DELETE FROM movies WHERE id=412321;

Remove all rows: TRUNCATE TABLE TableName;
Same as selete without a WHERE Clause.

_

Data Definition Language

CREATE TABLE language (id INT PRIMARY, lang VARCHAR(50) NOT NULL);

Datatypes: https://www.journaldev.com/16774/sql-data-types

Constraints: https://www.w3schools.com/sql/sql_constraints.asp

NOT NULL - Ensures that a column cannot have a NULL value
UNIQUE - Ensures that all values in a column are different
PRIMARY KEY - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
FOREIGN KEY - Uniquely identifies a row/record in another table
CHECK - Ensures that all values in a column satisfies a specific condition
DEFAULT - Sets a default value for a column when no value is specified
INDEX - Used to create and retrieve data from the database very quickly

****
****

ALTER: ADD, MODIFY, DROP

ALTER TABLE language ADD country VARCHAR(50);

ALTER TABLE language MODIFY country VARCHAR(60);

ALTER TABLE langauge DROP country;

****

Removes both the table and all of the data permanently.
DROP TABLE Tablename;

DROP TABLE TableName IF EXISTS;

#https://dev.mysql.com/doc/refman/8.0/en/drop-table.html

TRUNCATE TABLE TableName;

as discussed earlier same as DELETE FROM TableName;

****

Data Control Language for DB Admins.

https://en.wikipedia.org/wiki/Data_control_language
https://dev.mysql.com/doc/refman/8.0/en/grant.html
https://dev.mysql.com/doc/refman/8.0/en/revoke.html