My SQL.

My SQL is relational database management system (RDBMS) developed by oracle that is based on structured query language (SQL)

Database. A database is an organized collection of structured information or data.

DBMS. Database Management System (DBMS) are software system used to store, retrieve and run queries on data.

— : C R U D :—
Create Read Updata Delete.

DBMS.

Relational. (SQL)

Non-relational (No-SQL)

kanneCt
Get plugged in

A relation database is a collection of data items with predefined relationships between them, stored in the form of table, row and column

List of SQL Database :

1. MySQL 2. MariaDB 3. Oracle
4 PostgreSQL 5. MSSQL

Installation MySQL

1. Install a server
   mySQL community server
2. Install mySQL workbench.

How To Create Table in MySQL.

kanneCt
Get plugged in

Database Table.

Database table is collection of rows and columns that contain Relational Data

for eg.

Columns

schema

rows	Id	Name	Email	DoB
	1	Ramesh	abc@gmail	10/01/1999
	2	Johan	xyz@gmail	5/02/1997
	3	Reena	shiv@gmail	15/5/1997

Categories of DataTypes

1. String
2. Numeric
3. Date & Time.

Create Table Syntax.

```
CREATE TABLE table_name
(
    column1  datatype,
    column2  datatype,
    column3  datatype,
    ...
);
```

kanneCt
Get plugged in

String Data Types :-

1. CHAR (size) 0 to 255
2. VARCHAR (size) 0 to 65535
3. BINARY (size) Data in 0 1
4. VARBINARY (size)
5. TINYTEXT 255 character
6. TEXT (size) 65,535 bytes
7. MEDIUMTEXT 16,777,215 character
8. LONGTEXT 4,294,967,295 character
9. TINYBLOB 255 bytes
10. BLOB (size) 65,535 bytes
11. MEDIUMBLOB 16,777,215 bytes
12. LONGBLOB 4,294,967,295 bytes
13. ENUM (val1, val2, val3 ....) list upto 65535 value
14. SET (val1, val2, val3 ...) list upto 64 values

Number Data Types :-

			Both +ve -ve signed	only +ve unsigned
1.	BIT (size)	1 to 64		
2.	TINYINT (size)	-128 to 127	✓	
3.	INT (size)	-2147483648 to 2147483647		
4.	INTEGER (size)			
5.	SMALLINT (size)	-32768 to 32767		
6	BIGINT (size)	-8388608 to 8388607		
7.	MEDIUMINT (size)	-9223372036854775808 to 9223372036854775807		

8 BOOL

9. BOOLEAN 0/1

10. FLOAT (P)

11 DOUBLE (size, d) 255.568

12 DECIMAL (size, d) size = 60, d = 30

13 DEC (size, d)

Date Data types.

1. DATE '1000-01-01' to '9999-12-31'

2. DATETIME (FSP) YYYY-MM-DD hh:mm:ss

3. TIMESTAMP (FSP)

4. TIME (FSP) hh:mm:ss

8. YEAR. four digit format : 1901.

kanneGt
Get plugged in

4.

Add Data Into Tables using the INSERT Query.

1. Insert Single row

a. INSERT INTO table_name
(column1, column2, column3, ...)
VALUES
(value1, value2, value3, ...);

2. Insert Multiple rows

b. INSERT INTO table_name
(colum1, column2, colum3, ...)
VALUES
(value1, value2, value3, ...),
(value1, value2, value3, ...),
(value1, value2, value3, ...),

* We do not have need to mention column name, If we are adding values for all the columns of the table
* Make sure the order of the values is in the same order as the columns in table.

c. INSERT INTO table_name
VALUES (value1, value2, value3, ...)

Select Query With where Clause.

kannect
Get plugged in

Select query Syntax

- Select some columns.

```
SELECT column1, column2, ...
FROM table_name;
```

x set Alias.

```
SELECT column1 AS "First Colum", column2, ...
```

- Select All columns using * astrick

```
SELECT * FROM table_name;
```

Where Clause.

- The WHERE clause is used to filter records.
- It is used to extract only those records that fulfil a specified condition.

kanneCt
Get plugged in

Syntax.

astrick
*

SELECT column1, column2, ...
FROM table_name.
WHERE condition.

condition.

= equals
!= not equals.
>= Greater than or equal to
<= Less than or equal to
> greater than
< less than

for eg.

SELECT AGE FROM Student WHERE AGE> 18

Table Constraints Tutorial with Example.

- NOT NULL - Ensures that a column can not have a null value.
- UNIQUE - Ensures that all values in column are different.
- DEFAULT - Sets a default value for a column if no value is specify
- CHECK - Ensures that the value in column if no value is specified.
- FOREGIN KEY - Combination of NOT NULL and Unique. Uniquely identifies each row in table
- PRIMARY KEY. - Prevents actions that would destroy link b/w table.

SQL Constraints are used to specify rules for the data in a table

kanneCt
Get plugged in

7.

Commands: And, OR & NOT operator.

- **AND operator** : MySQL Logical AND operator compares two expression and returns true if both of the expression are true.

- **OR operator** : MySQL OR operator compares two expressions and return TRUE if either of the expressions is TRUE.

- **NOT Operator** : MySQL NOT operator reverses or negates the inputs.

Syntax.

SELECT * From table_name
WHERE

1. condition1 AND condition2.
2. condition1 OR condition2
3. NOT condition1

for eg: SELECT * FROM student
WHERE NOT Gender="F" AND NOT Gender="M"

* Select those student which are neither femal ~~and~~ NOR male.

kannect
Get plugged in

8.

IN Operator on Database Table.

* IN operator :- The IN operator allows you to specify multiple values in WHERE clause

* The IN operator is a shorthand for multiple OR conditions

Syntax.

```
SELECT * FROM table_name.
WHERE column_name IN
(value1, value2, ...);
```

for example-

```
SELECT * FROM student WHERE
age = 19 OR age = 20 OR age = 21 OR
age = 22.
```

→ Instead of using OR multiple time we use IN

(*) SELECT * FROM student WHERE age IN (19, 20, 21, 22)

LIKE Operator and Wildcard.

SR. No	Name	Email.
1	Bhagirath	bhagirat@gmail.com
2	Bhavesh	bhavesh@gmail.com
3	Sunil	sunil@——
4	Ramesh	ramesh@—
5	Ravi	ravi@—
6	Suresh.	suresh@—

Task. We have to select those student whose name is started with 'B'

- Here we us Like operator

```
SELECT * FROM users WHERE
name Like "b%")
```

↓

Result

SR	Name	Email.
1	Bhagirath	bhagirath@gmail.com
2	Bhavesh	bhavesh@gmail.com.

MySQL LIKE

The LIKE operator is used in WHERE clause to search for a specified pattern in column

- The percent sing (%) represents zero, one, or multiple characters

- The underscore siagn (_) represent one, single character.

LIKE Operator	Description.
LIKE 'a%'	starts with "a"
LIKE '%a'	End with "a"
LIKE '%or%'	Have "or" in any position
LIKE '_r%'	Have "r" in 2nd position
LIKE 'a_%'	starts with "a" and are at least 2 characters in length
LIKE 'a__%'	Starts with "a" and are at least 3 characters in length.
LIKE 'a%o'	Start with "a" and end with "o"

10.

kanneCt
Get plugged in

Between and Not Between Operator.

→ The BETWEEN operator selects values within a given range. The value can be number, text or dates.

→ The ~~Between~~ BETWEEN operator is inclusive: begin and end value are included.

BETWEEN Syntax.

```
SELECT column_name
FROM table_name
WHERE column_name BETWEEN
value1 AND value2.
```

for eg

1. SELECT * From users WHERE age BETWEEN 20 AND 25

2. SELECT * FROM users WHERE age NOT BETWEEN val1 and val2

11

Order By and Distinct

The ORDER BY keyword is used to sort the result-set in asscending or descending order

→ The ORDER BY keyword sorts the records in ascending order by default

→ To sort the records in descending order, use the DESC keyword.

ORDER BY syntax.

```
SELECT column1, column2, ....
FROM table_name
ORDER BY column1, column2, ... ASC/DESC;
```

OR.

```
SELECT * FROM table_name ORDER BY
col1 ASC|DESC
```

for eg.

```
SELECT * FROM student ORDER BY
name DESC.
```

* DISTINCT Statement

The SELECT DISTINCT statement is used to return only distinct (different) values.

- Inside a table, a column often contain many duplicate values; and sometime

Syntax :

```
SELECT DISTINCT column1, column2,...
FROM table_name
```

* ORDER BY + DISTINCT.

for exp.

```
SELECT DITINCT age FROM
student ORDER BY age DES
```

IS NULL and IS NOT NULL Operator

→ The IS NULL operator is used to test for empty values (NULL values)

→ The IS NOT NULL operator is used to test for non-empty values (NOT NULL values).

* What is NULL value.

- A field with a NULL value is a field with no value.

- If a field in a table is optional, it is possible to insert a new record or update a record without adding value to this field, Then the field will be saved with a NULL value.

Syntax.

```
SELECT * FROM table-name
WHERE col-name
IS NULL | IS NOT NULL
```

Tip:- Always use IS NULL to look for NULL values.

LIMIT and Offset

→ The LIMIT clause is used to specify the number of records to return.

→ If want to LIMIT, the number of results that are returned you can simply use the limit command with several ways to LIMIT by.

→ The LIMIT clause is on large tables with thousands of records Returning a large number of records can impact performance.

Syntax

```
SELECT * FROM table-name
LIMIT number
```

→ with ORDER BY

```
SELECT * FROM table_name
ORDER BY col-name DESC
LIMIT number
```

3. Using WHERE clause.

SELECT * FROM table-name.
WHERE age > 18 LIMIT number

OFFSET

You can also specify an ~~off~~ OFFSET from where to start returning data

Syntax:

SELECT * FROM table-name
LIMIT [Number to LIMIT By].

for example: SELECT * FROM student
LIMIT 5 OFFSET 7

⇒ This will return Five student from 7th position.

Aggregate Function's :- SUM, MIN, MAX, AVERAGE (AVG)

Function's

1. COUNT(): Returns the number of rows in a database table.
2. SUM() :- Returns the total sum of a numeric column
3. AVG() :- Calculates the average of a set of values
4. MIN() :- Returns the lowest value (minimum) in a set of non-null values.
5. MAX() : Returns the greatest value (maximum) in a set of non-NULL values.

Syntax: 1. COUNT()

```
SELECT  COUNT (col-name),
FROM    table-name
WHERE   condition
```

2. SUM()

```
SELECT  SUM (column-name)
FROM    table-name
```

3. AVG()

```
SELECT  AVG (col-name)
FROM    table-name
```

4. MIN():

```
SELECT  MIN (col-name)
FROM    table-name
```

5. MAX()

```
SELECT  MAX (col-name)
FROM    table-name.
```

UPDATE Statement

The UPDATE statement is used to modify the existing records in a table.

NOTE :- Be careful when updating records in a table! NOTICE the WHERE clause in the UPDTE statement The WHERE clause specifies which record(s) that should be updates

→ If you omit the WHERE clause, all records in table will be updated

Syntax

```
UPDATE table-name
SET column1 = newValue1,
column2 = newValue2, ....
WHERE condition.
```

for ex

```
UPDATE user SET city = 'Chandigarh',
email = 'abc@gmail.com'
WHERE id = 2
```

DELETE Query.

The DELETE statement is used to delete existing records in a table

Syntax

```
DELETE FROM table-name
WHERE Condition.
```

Note:- If you omit the WHERE clause, all records in the table will be deleted.

For Example.

1.
```
DELETE FROM user
WHERE id IN (5,6);
```

2.
```
DELETE FROM user
WHERE id = 5;
```

COMMIT and ROLLBACK

COMMIT :- The commit statement lets a user save any changes or alteration. These changes will remain permanent

→ Once you use the commit command to (completely) execute the current transaction, then it cannot undo and get back to its previous state in any way

2. **ROLLBACK** :- The ROLLBACK statement lets a user undo all the alteration and changes that occured on the current transaction after the last COMMIT

→ On the otherhand, the ROLLBACK command assists a user to get the command back to its previous state. It lets them undo the current transaction.

For example: Here Initial fees = 5000

```
SELECT * from ~~shivam~~ user ;
Update user SET fees=15000 WHERE id=1 ;
ROLLBACK ;
COMMIT ;
```

// If we run update command by pressing ⇧I, this will update the fees = 15000

→ NOW if we run ROLLBACK command it will get back to initial fees = 5000

→ If we run COMMIT command first then the changes will be permanently saved, ROLLBACK can not change the value.

Transaction can be INSERT, UPDATE, DELETE

Data Transaction

Primary key and Foreign key.

Primary key.

1. Primary key contains always unique data.
2. It can not be null.
3. There must be a single primary key.

Primary key Syntax.

```
CREATE TABLE students (

id INT NOT NULL AUTO-INCREMENT,
name VARCHAR(50) NOT NULL,
age INT NOT NULL ,
city VARCHAR(10) NOT NULL ,

PRIMARY KEY (id)
```

Foreign key

1. The foreign key is used to link two tables.
2. A foreign key in one table (child table) is used to point PRIMARY KEY in another table (parent table)

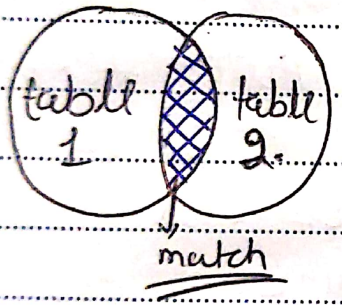
Syntax

```
CREATE TABLE students (
id INT NOT NULL AUTO_INCREMENT,
name VARCHAR(50) NOT NULL,
age INT NOT NULL,
city INT NOT NULL,
PRIMARY KEY (id)
FOREIGN KEY (city) REFERENCES city (id)
```

INNER JOIN COMMAND

→ The MySQL Inner Join is used to returns only those results from the tables that match the specified condition and hides other rows and columns.

→ MySQL assumes it as a default Join, so it is optional to use Inner Join keyword with the query



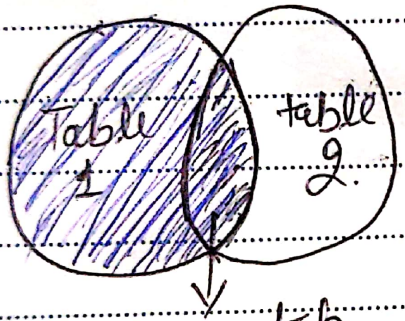
Syntax.

```
SELECT columns
FROM table1
INNER JOIN table2 ON condition1
```

LEFT JOIN & Right Join

1. **LEFT JOIN** :- The MySQL LEFT JOIN keyword returns all records from the left table (table1), and and the matched records from right table (table2)

LEFT JOIN.

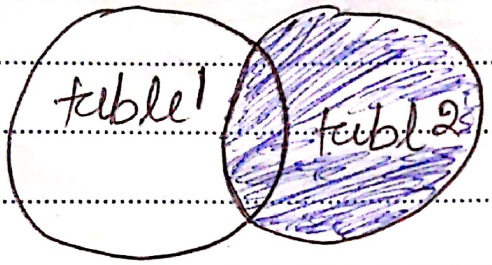


Syntax.

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON
table1.col-name = table2.col-name;
```

MySQL RIGHT JOIN

- The RIGHT JOIN keyword returns all records from the right table (table2), and the matched records from the left table (table1)



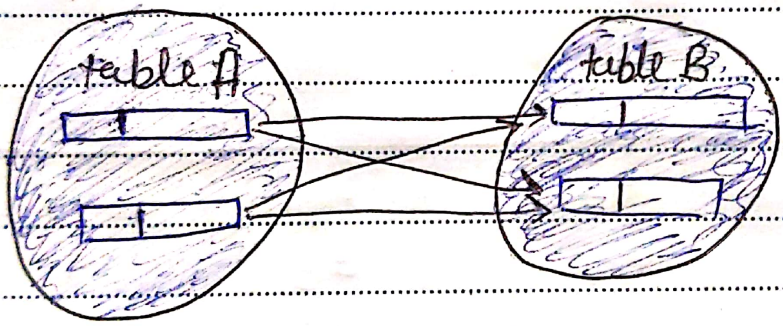
Syntax,

```
SELECT  col-name(s)
FROM    table1
RIGHT JOIN  table2
 ON
table1.col-name = table2.col-name
```

Cross Join in MySQL

- The MySQL CROSS JOIN produces a result set which is the number of rows in the first table multiplied by the number of rows in the second table if no WHERE clause is used along with CROSS JOIN

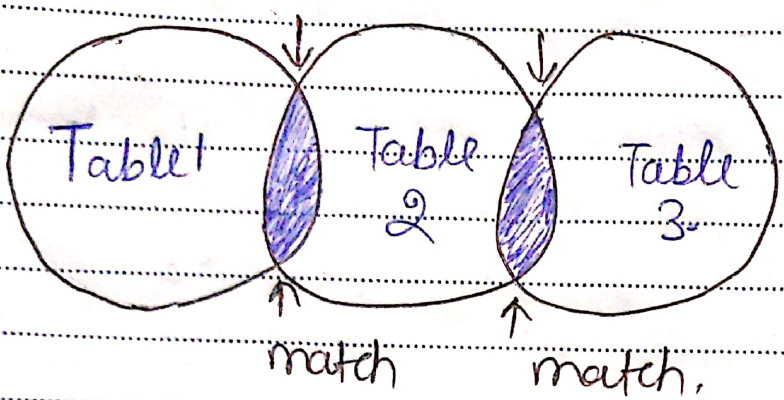
- This kind of result is called as Cartesian product.




```
SELECT    col-name
FROM      table 1
CROSS JOIN  table 2
  ON  table1.col-name = table2.col-name
```

Get plugged in

Join Multiple Tables using MySQL Command.



→ Gives data which is common among all three tables

table 1 Students

Id	Name	Age	Course	City
1	Ravi	23	1	1
2	Shivam	25	1	2
3	Rahul	20	2	1
4	Kapil	19	3	3

table 2. cities

cid	Name
1	Agra
2	Bhopal
3	Delhi
4	Noid

table 3 Courses

cid	Name
1	PHP
2	Java
3	C++

Syntax.

```
SELECT  col-name(s)
FROM    table 1
INNER JOIN  table 2
 ON  table 1 . col-name = table 2 . col-name;
INNER JOIN
 ON
```

for eg.

```
SELECT * From students
Inner JOIN cities
on students.cid = cities.id
INNER JOIN courses
on students.courseid = courses.id;
```

kahneCt
Get plugged in

Group By & Having Clause.

1. Group By

Syntax.

```
SELECT col-name(s)
FROM table 1
WHERE condition
GROUP BY col-name (s)
```

The GROUP By statement groups rows that have the same values into summary rows, like "find the number of customer in each country"

→ The GROUP BY statement is often used with aggregate function COUNT(), MAX(), MIN(), SUM(), AVG() to group the results-set by one or more column

for example

```
select cid, cities.name, COUNT(cid) as Total
FROM students
inner join cities
on students.cid = cities.id
group by (cid)
```

2. Having Clause

Syntax

```
SELECT col-name
FROM table1
WHERE condition
Group BY col-name
HAVING condition
```

The HAVING clause was added to SQL because the WHERE keyword can not be used with aggregate functions.

for example

```
select cid, cities.name, COUNT as Total
FROM students
inner joint cities
on students.cid = cities.id
group by (cid)
having count(cid) >= 2.
```

kanneCt
Get plugged in

SubQuery With EXISTS & NOT EXISTS

Sub Query

A MySQL subquery is a query nested within another query such as SELECT, INSERT, UPDATE and DELETE

- Also, a subquery can be nested within another subquery

Syntax

```
SELECT col_name(s)
FROM table1
WHERE
(SELECT col_name FROM table2)
```

For Example :-

```
select name for students
where cid = (select id from cities where
                name = "Jodhpur");
```

Exists : The Exists operator is used used to test for the existence of any record in a subquery.

The EXISTS operator return TRUE if the subquery returns one or more records.

Syntax

```
SELECT col-name(s)
FROM table1
WHERE EXISTS
(SELECT col-name FROM table2)
```

for example

```
Select name from students
WHERE (select id from cities where
        name = 'Agra');
```

NOT Exists

NOT EXISTS is the vice-versa of EXISTS