SQL - JournalDev Good Site for SQL

1. Create A Table

```
CREATE TABLE personal (
    id INT ,
    name VARCHAR(50) ,
    birth_date DATE,
    phone VARCHAR(15),
    gender VARCHAR(1)
);
```

2. Insert data into table

Create Table

Example

```
CREATE TABLE personal (
    id INT ,
    name VARCHAR(50) ,
    birth_date DATE,
    phone VARCHAR(15),
    gender VARCHAR(1)
);
```

Insert Data in personal table

Example

```
INSERT INTO personal ( id, name, birth_date, phone, gender)
VALUES ( 1,"Ram Kumar", "1990-07-15", "9977664422", "M" );

INSERT INTO personal ( id, name, birth_date, phone, gender)
VALUES ( 2,"Meera Khan", "1991-02-10", "9988552211", "F" );

INSERT INTO personal ( id, name, birth_date, phone, gender)
VALUES ( 1,"Anil Kapoor", "1993-10-05", "9484542414", "M" );
```

3. Make Some constraints on our table

```sql
CREATE TABLE personal(
    id INT NOT NULL UNIQUE,
    name VARCHAR(50) NOT NULL,
    age INT NOT NULL CHECK(age >= 18),
    gender VARCHAR(1) NOT NULL,
    phone VARCHAR(10) NOT NULL UNIQUE,
    city VARCHAR(15) NOT NULL DEFAULT 'Agra'
);
```

Isme Id null ni hgi and and whi data save hga jisme age >=18 h

4. Select and where clause

Select Query Examples

Example

```sql
SELECT * FROM personal;

SELECT id,name,phone FROM personal;

SELECT id AS Id,name AS Student,phone AS Phone FROM personal
```

Select with Where

Example

```sql
SELECT * FROM personal WHERE gender = "F";

SELECT * FROM personal WHERE age<20;

SELECT * FROM personal WHERE city != "Agra";

SELECT * FROM personal WHERE city = "Agra";

SELECT id, name FROM personal WHERE city < > "Agra";
```

5. AND OR NOT

AND, OR, NOT

Example

SQL Copy

```sql
SELECT * FROM personal WHERE age >= 18 AND age <= 21;

SELECT * FROM personal WHERE  age <= 20 AND gender = "M";

SELECT * FROM personal WHERE  age <= 20 OR city = "Agra";

SELECT * FROM personal WHERE  (city = "Bhopal" OR city = "Agra") AND gender = "M";

SELECT * FROM personal WHERE NOT (city = "Bhopal" OR city = "Agra");

SELECT * FROM personal WHERE NOT age >= 20;
```

6. SQL | BETWEEN

BETWEEN

The SQL BETWEEN condition allows you to easily test if an expression is within a range of values (inclusive). The values can be text, date, or numbers. It can be used in a SELECT, INSERT, UPDATE, or DELETE statement. The SQL BETWEEN Condition will return the records where expression is within the range of value1 and value2.

```sql
SELECT Fname, Lname
FROM Employee
WHERE Salary
BETWEEN 30000 AND 45000;
```

7. SQL | IN

IN

IN operator allows you to easily test if the expression matches any value in the list of values. It is used to remove the need of multiple OR condition in SELECT, INSERT, UPDATE or DELETE. You can also use NOT IN to exclude the rows in your list. We should note that any kind of duplicate entry will be retained.

```
SELECT Fname, Lname
FROM Employee
WHERE Salary IN (30000, 40000, 25000);
```

8. SQL | Like Operator

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE CustomerName LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

9. SQL | Order By

The ORDER BY statement in SQL is used to sort the fetched data in either ascending or descending according to one or more columns.

- By default ORDER BY sorts the data in ascending order.
- We can use the keyword DESC to sort the data in descending order and the keyword ASC to sort in ascending order.

Sort By on the basis of more than one column

```
SELECT * FROM table_name ORDER BY column1 ASC|DESC , column2 ASC|DESC
```

10. SQL | Distinct

The SQL SELECT DISTINCT Statement

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

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

SELECT DISTINCT Syntax

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

11. SQL | Limit

SQL | LIMIT Clause

Difficulty Level : Basic • Last Updated : 06 Nov, 2019

If there are a large number of tuples satisfying the query conditions, it might be resourceful to view only a handful of them at a time.

- The LIMIT clause is used to set an upper limit on the number of tuples returned by SQL.
- It is important to note that this clause is not supported by all SQL versions.
- The LIMIT clause can also be specified using the SQL 2008 OFFSET/FETCH FIRST clauses.
- The limit/offset expressions must be a non-negative integer.

```
SELECT *
FROM Student
ORDER BY Grade DESC
LIMIT 3;
```

Output:

12006	Anne	10
12001	Aditya	9
12004	Robin	9

Using LIMIT along with OFFSET

LIMIT x OFFSET y simply means skip the first y entries and then return the next x entries.
OFFSET can only be used with ORDER BY clause. It cannot be used on its own.
OFFSET value must be greater than or equal to zero. It cannot be negative, else returns error.

Queries:

```
SELECT *
FROM Student
LIMIT 5 OFFSET 2
ORDER BY ROLLNO;
```

12. SQL Aggregate Function

Introduction to SQL aggregate functions

An aggregate function allows you to perform a calculation on a set of values to return a single scalar value. We often use aggregate functions with the GROUP BY and HAVING clauses of the SELECT statement.

The following are the most commonly used SQL aggregate functions:

- AVG – calculates the average of a set of values.
- COUNT – counts rows in a specified table or view.
- MIN – gets the minimum value in a set of values.
- MAX – gets the maximum value in a set of values.
- SUM – calculates the sum of values.

COUNT

Example

```
SELECT COUNT(name) FROM personal;

SELECT COUNT(*) FROM personal;

SELECT COUNT(DISTINCT city) FROM personal;

SELECT COUNT(DISTINCT city) AS Count FROM personal;
```

MAX

Example

```
SELECT MAX(percentage) AS Percentage FROM personal;
```

MIN

Example

```
SELECT MIN(percentage) AS Percentage FROM personal;

SELECT MIN(percentage) AS Percentage,name,city FROM personal;
```

SUM

Example

```
SELECT SUM(percentage) AS Total FROM personal;
```

13. SQL | Update

The SQL UPDATE Statement

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

UPDATE Syntax

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

The following SQL statement updates the first customer (CustomerID = 1) with a new contact person *and* a new city.

Example

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

14. SQL | Alter

SQL ALTER TABLE Statement

The `ALTER TABLE` statement is used to add, delete, or modify columns in an existing table.

The `ALTER TABLE` statement is also used to add and drop various constraints on an existing table.

ALTER TABLE - ADD Column

To add a column in a table, use the following syntax:

```
ALTER TABLE table_name
ADD column_name datatype;
```

The following SQL adds an "Email" column to the "Customers" table:

Example

```
ALTER TABLE Customers
ADD Email varchar(255);
```

ALTER TABLE - DROP COLUMN

To delete a column in a table, use the following syntax (notice that some database systems don't allow deleting a column):

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

The following SQL deletes the "Email" column from the "Customers" table:

Example

```
ALTER TABLE Customers
DROP COLUMN Email;
```

15. SQL Commit & Rollback

SQL Commit And Rollback - JournalDev yha se pdh lena acha btaya h with example

16. SQl | Delete

SQL Delete Query is used to remove rows from table in a database. In a database the storage and retrieval of data is the most important aspect. But, there are cases when we have insert some incorrect data by mistake and we have to remove it. Or the data is obsolete now and we can delete it, such as logging information that can be purged after few days.

SQL Delete Syntax

If we want to delete specific rows, then we need to provide delete statement with where clause.

```
DELETE From table_name WHERE condition;
```

,

SQL Delete Row

Let's try to understand the DELETE command through some example. Let's consider the following Customer Table to understand DELETE command.

CustomerId	CustomerName	CustomerAge	CustomerGender
1	James	32	M
2	Diana	26	M
3	Annie	35	F

We want to delete rows with CustomerGender as Female. The delete statement will be;

```
DELETE FROM Customer WHERE CustomerGender = 'F';
```

17. Primary Key

create table "city"

Example

```
CREATE TABLE city(
    cid INT NOT NULL AUTO_INCREMENT,
    cityname VARCHAR(50) NOT NULL,
    PRIMARY KEY (cid)
);
```

18. Foreign Key

Persons Table

PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

Orders Table

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

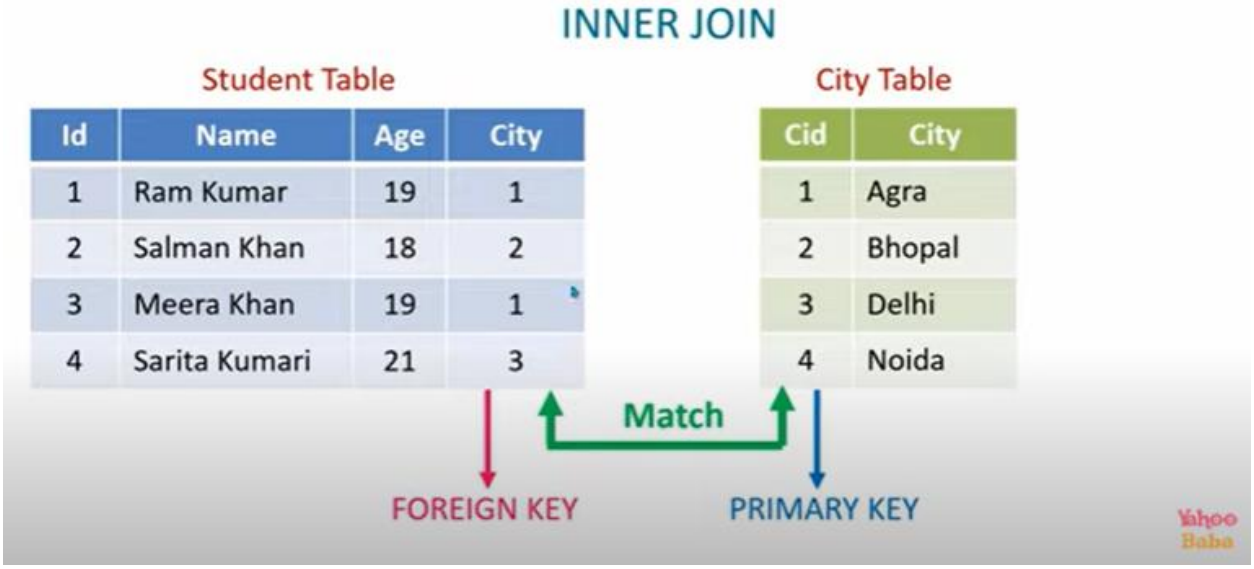
SQL FOREIGN KEY on CREATE TABLE

The following SQL creates a `FOREIGN KEY` on the "PersonID" column when the "Orders" table is created:

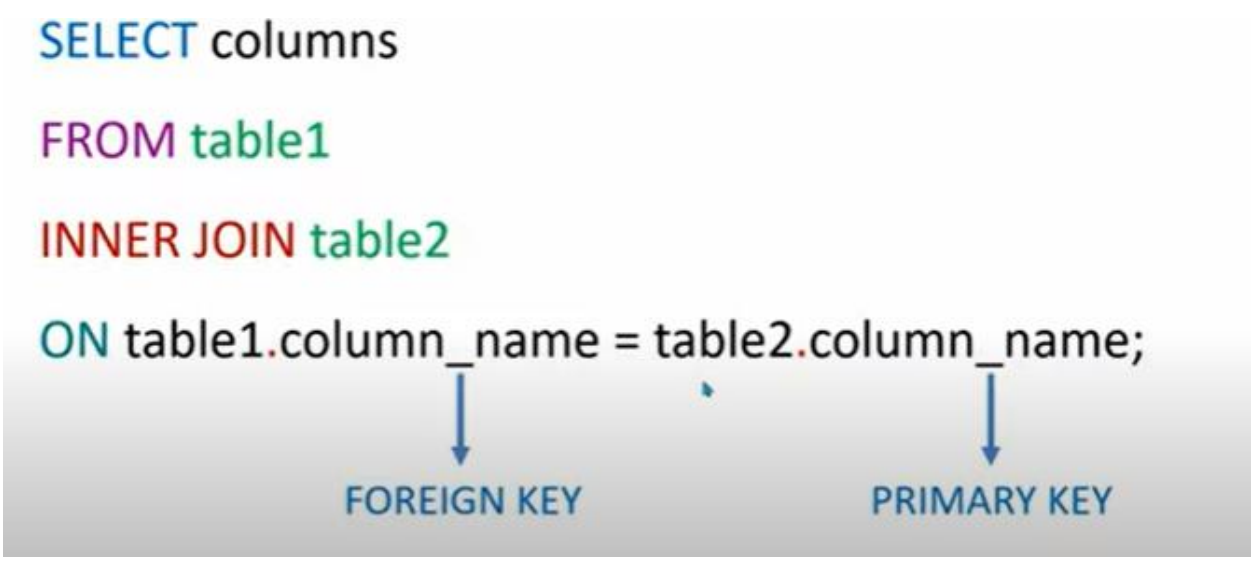
MySQL:

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);
```

19. Inner Join



Syntax



```
1  SELECT p.id,p.name,p.percentage,p.age,p.gender,c.cityname
2  FROM personal p INNER JOIN city c
3  ON p.city = c.cid
4  WHERE c.cityname = "Agra"
5  ORDER BY p.name;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	name	percentage	age	gender	cityname
1	Ram Kumar	45	19	M	Agra
3	Salman Khan	62	20	M	Agra
5	Anil Kapoor	74	22	M	Agra
7	Shahid Kapoor	52	20	M	Agra

20. SQL || Group by

SQL | GROUP BY

Difficulty Level : Easy • Last Updated : 21 Mar, 2018

The GROUP BY Statement in SQL is used to arrange identical data into groups with the help of some functions. i.e if a particular column has same values in different rows then it will arrange these rows in a group.

Important Points:

- GROUP BY clause is used with the SELECT statement.
- In the query, GROUP BY clause is placed after the WHERE clause.
- In the query, GROUP BY clause is placed before ORDER BY clause if used any.

Syntax:

```
SELECT column1, function_name(column2)
FROM table_name
WHERE condition
GROUP BY column1, column2
ORDER BY column1, column2;

function_name: Name of the function used for example, SUM() , AVG().
table_name: Name of the table.
condition: Condition used.
```

21.SQL || Having

SR.NO.	WHERE Clause	HAVING Clause
1.	WHERE Clause is used to filter the records from the table based on the specified condition.	HAVING Clause is used to filter record from the groups based on the specified condition.
2.	WHERE Clause can be used without GROUP BY Clause	HAVING Clause cannot be used without GROUP BY Clause
3.	WHERE Clause implements in row operations	HAVING Clause implements in column operation
4.	WHERE Clause cannot contain aggregate function	HAVING Clause can contain aggregate function
5.	WHERE Clause can be used with SELECT, UPDATE, DELETE statement.	HAVING Clause can only be used with SELECT statement.
6.	WHERE Clause is used before GROUP BY Clause	HAVING Clause is used after GROUP BY Clause
7.	WHERE Clause is used with single row function like UPPER, LOWER etc.	HAVING Clause is used with multiple row function like SUM, COUNT etc.

```
1•  SELECT c.cityname, COUNT(p.city) AS Total
2   FROM personal p INNER JOIN city c
3   ON p.city = c.cid
4   GROUP BY city
5   HAVING COUNT(p.city) > 3
6   ORDER BY COUNT(p.city) DESC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

cityname	Total
Agra	4

22.SQL Subquery

Esme Ek query k andr dsri subquery likhi hoti h.
In MySQL, a Subquery is defined as a SELECT SQL Statement that is used inside another SQL statement to calculate the results of outer queries. Simply in SQL, a Subquery is an inner query which is placed within an outer SQL query using different SQL clauses like WHERE, FROM, HAVING and with statement keywords such as SELECT, INSERT, FROM, UPDATE, DELETE, SET or DO, accompanied with expressional operators or logical operators.

```
SELECT columns
FROM table1
WHERE
column = (SELECT columns FROM table2 WHERE condition);
```

Iska Example Yahoo Baba pr dekh lena.

23. IF and CASE

```
SELECT column1, column2,
IF (Condition, TRUE Result, FALSE Result ) AS alias_name
FROM table_name;
```

```
1 • SELECT id,name, percentage,
2   IF(percentage >= 33,"Pass","Fail") AS Result
3   FROM students;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	name	percentage	Result
1	Ram Kumar	45	Pass
2	Sarita Kumari	85	Pass
3	Salman Khan	29	Fail
4	Juhi Chawla	47	Pass
5	Anil Kapoor	74	Pass
6	John Abraham	64	Pass
7	Shahid Kapoor	120	Pass

Yahoo Baba

CASE

Case statement ka use wha hota h jha multiple condition hme lgani hoti h

```
SELECT column1, column2,
CASE
    WHEN Condition1 THEN result1
    WHEN Condition2 THEN result2
    WHEN Condition3 THEN result3
    ELSE result alias_name
END AS alias_name
FROM table_name;
```

```sql
SELECT id,name, percentage,
CASE
    WHEN percentage >= 80 AND percentage <= 100 THEN "Merit"
    WHEN percentage >= 60 AND percentage < 80 THEN "Ist Division"
    WHEN percentage >= 45 AND percentage < 60 THEN "IInd Division"
    WHEN percentage >= 33 AND percentage < 45 THEN "IIIrd Division"
    WHEN percentage < 33 THEN "Fail"
    ELSE "Not Correct %"
END AS Grade
FROM students;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	name	percentage	Grade
1	Ram Kumar	45	IInd Division
2	Sarita Kumari	85	Merit
3	Salman Khan	29	Fail
4	Juhi Chawla	47	IInd Division
5	Anil Kapoor	74	Ist Division
6	John Abraham	64	Ist Division
7	Shahid Kapoor	120	Not Correct %