DBMS

MODULE 4
Structured Query Language(SQL)

Written & Compiled by,
Prof.Priyanka Bhilare

3. Structured Query Language.

* Overview of SQL :-

SQL is a database computer language designed for the retrieval and management of data in relational database.

SQL can perform following operations :-

1) SQL can retrieve data from a database.
2) SQL can insert records in a database.
3) SQL can update records in database.
4) SQL can create new database.
5) SQL can set permissions on tables.

* SQL consists of following languages :-

i) Data Definition Language (DDL).
ii) Data Manipulation Language (DML).
iii) Data Control language (DCL).
iv) Transaction Control Language (TCL).

i) Data definition language (DDL).

Before DB can be used it must be created. DDL is used for defining a language of DB to the computer. So, SQL DDL provides commands for ~~defiining~~ defining the relation, deleting the relations and modifying existing relation schema.

Scanned with CamScanner

The most important DDL statements in SQL are :-

1) Create :- It helps to create new table.

syntax:-

Create table tablename (Columnname 1 datatype(size),
Columnname 2 datatype (size),
:
Columnname n datatype (size));

eg:-

create table student (sid int(9), sname varchar(10), smarks int(10), DOB date);

2) Drop :- A table may be dropped or destroyed using Drop table command.

syntax:-
Drop table tablename;

eg:- drop table student;

3) Alter :- It is used to modify an existing table. It manages changes like adding or dropping a column, changing a column ~~to~~ definition as well as adding or dropping a table constraint.

Scanned with CamScanner

i) To add a column :-

syntax :-
alter table tablename add columnname datatype(size);

eg:- alter table student add grade int(9);

ii) Add multiple columns :-

syntax :-
alter table tablename add columname1 datatype(size), add columnname2 datatype(size), add columnname3 datatype(size), ... add columnnameN datatype(size);

eg:-
alter table student add emailid varchar(15), add deptname varchar(20);

iii) To add column at the first :-
eg:- alter table student add branch_name varchar(9) first;

iv) To add column in between :-
eg:- alter table student add contactno int(10) after sname;

v) To change datatype of column :-
syntax :- alter table tablename ~~alter~~ modify column columnname datatype(size);

Scanned with CamScanner

OR

alter table tablename modify column columnname1 datatype (size), modify column columnname2 datatype (size);

vi) To drop column :-

Syntax:-

alter table tablename drop column columnname;

OR

alter table tablename drop column columnname 1, columnname 2, drop column

4) Rename :-

Syntax:- (for table)

rename table oldtablename to newname;

eg:-

rename table student to student_info;

Syntax (for column)

alter table tablename ~~rename~~ change column oldcolumnname newcolumnname datatype (size);

eg:- alter table student_info change column sname student_name varchar (15);

5) Describe :-

Syntax:- desc tablename;

eg:- desc student_info;

Scanned with CamScanner

2) Data Manipulation Language (DML):-

These commands are used for storing, retrieving, updating and deleting data. These commands basically deals with manipulation of data present in the DB.

1) Insert Command:-

Insert command is basically used to insert new records in the table.

Syntax:-

insert into tablename values (value1, value2, value3, ... valuen);

eg:- insert into student values (1, 'priyanka', 75, '1991-02-06');
YYYY-MM-DD

- Insert values for specified columns:-

syntax:- insert into tablename (columnname1, columnname2) values (v1, v2);

eg:- insert into student (id, name, DoB) values (2, 'Tejas', '1987-13-04');

- To insert multiple values for specified column:-

eg:- insert into student (id, name, DOB) values (3, 'Snehal', '2004-13-05'), (4, 'vipul', 1990-02-06);

Scanned with CamScanner

2) Update Command :-

Update statement is used to modify data in a table.

Syntax :- update tablename set columnname = newvalue where columnname = oldvalue;

eg :- update student set id = 6 where name = 'priyanka';

- multiple updates :-

eg :- update student set id = (7, 8) where name = ('Vipul', 'Snehal');

- update several columns in a row :-

eg :- update student set id = 7, DOB = '2005-18-12' where name = 'priyanka';

eg :- update student set id = (4, 5) where sname = ('priyanka', 'Snehal');

3) Delete Command :-

It is used to delete rows from a table.

Syntax :- delete from table where condition;

eg :- delete from student where id = 1;

Scanned with CamScanner

4) Select Command :-

It is used to retrieve data from the DB.

Syntax :- select columnname(s) from tablename where condition;

eg :- select sid, name from student where sid=1;

eg :- select * from student;

(*) Constraints in DBMS :-

Constraints are placed on columns or tables. They limit the data that can be inserted into tables.

- **Integrity constraints :-**

Integrity constraints are rules that are to be applied on DB columns to ensure the validity of data. Every time data is evaluated or checked against the constraint and only if result comes out to be true; then only data inserted into the column. Integrity constraints prevent ~~da~~ invalid data entry into the table.

Scanned with CamScanner

Integrity constraints can be divided into following

A] Domain Integrity Constraints :-

There are constraints on domain value, therefore these constraints are used to impose restrictions on some particular column.

Types of domain Integrity constraints :-

i) Not null constraint :-

By default all columns in a table allows null values. When a Not null constraint is enforced on a column or set of columns, it will not allow null values.

⊙ syntax :-

create table tablename (columnname1 datatype(size), columnname2 datatype(size) not null, columnname3 datatype(size));

eg :- create table student (sid int(4), sname varchar(10) not null, DoB date);

After creating table not null constraint can be added as,

⊙ syntax :-

Alter table tablename modify

columnname datatype (size) not null;

eg:- alter table student modify DOB date not null;

- Dropping not null constraint :-

syntax :-

Alter table tablename modify columnname datatype(size) null;

eg:- alter table student modify DOB date null

~~ii) Unique constraint :-~~

Not null constraint cannot take null values but can take duplicate values.

① insert into student values (1, 'priyanka', null);
→ error will be displaced on the screen.

② insert into student values (2, 'sneha', '2019-12-03');
— record added to the table.

③ insert into student values (3, 'Aakash', '2019-12-03');
— record added to the table ~~as~~ although we have duplicate entry for DOB.

Scanned with CamScanner

ii) Default constraint :-

Default constraint is used to provide default value for a column. Default value is added to all new records if no other value is specified.

⊙ Syntax :-

Create table tablename (columnname 1 datatype(size), columnname 2 datatype(size) default 'value');

eg:- create table student (sid int(9), sname varchar(10) default 'priyanka', D.B date);

① insert into student values (1, 'Rajesh', '1991-12-07');
— record inserted in the table

② insert into student (sid, dob) values (2, '1987-13-02');
— record inserted with default student name as priyanka.

⊙ After creating table default constraint can be added as:

Syntax :-
alter table tablename alter columname set default 'value';

Scanned with CamScanner

eg:- alter table student alter ~~columnname~~ sname set default 'priyanka';

⊙ Droping default

syntax:- alter table tablename alter column columnname drop default;

eg:- alter table student alter column sname drop default;

iii) Check constraint:-

Check constraints define a condition that each column must satisfy.

⊙ syntax:-

create table tablename (columnname 1 datatype(size), columnname 2 datatype (size), columnname 3 datatype(size), check (condition));

eg:- create table student (id int(9), age int(10), marks int(9), check (age >= 18));

① insert into student values (1, 20, 85);
— record inserted successfully.

② insert into student values (2, 15, 87);
— check constraint violeted error as age is not >= 18.

Scanned with CamScanner

show index from tablename; use select constraint_name from ... where table_name = tablename

- After creating table check constraint can be added as follows:-

Syntax:-
alter table tablename add constraint constraintname check (condition);

eg:-
alter table student add constraint chk1 check (age >=18);

- To drop check constraint:-

syntax:- alter table tablename drop constraint constraintname;

eg:- alter table student drop constraint chk1;

B] Entity - Integrity Constraints :-

As the name implies, it is applied on each entity i.e. on individual row. It specifies that each row should have uniqueness.

There are two types of Entity Integrity constraints:-

Scanned with CamScanner

1) Unique constraint:-

Unique key constraint does not allow duplicate values but it allows null values.

⊙ Syntax:-

create table tablename (columnname1 datatype(size), columnname2 datatype(size), columnname3 datatype(size) unique);

eg:- create table student (sid int(9), sname varchar(9), smarks int(9) unique);

① Insert into student values (1, 'priyanka', 75);
— record inserted successfully.

② insert into student values (2, 'Aakash', 75);
— Error : duplicate value for 'smarks.

③ insert into student values (2, 'Aakash', null);
— record inserted successfully.

⊙ After table created if you want to add unique constraint then the syntax is given below:-

syntax:- alter table tablename add constraint constraintname unique (columnname);

eg:- alter table student add constraint u1 unique (smarks);

Scanned with CamScanner

Date
Page

⊙ Syntax to drop unique constraint :-

alter table tablename drop index constraintname

eg:- alter table student drop index u1;

Note:- While creating table if unique constraint is added then name of the unique constraintname is not specified, in this case if you want to drop a unique constraint then use following syntax to see unique constraintname ~~as~~ and drop the unique constraint accordingly.

show index from tablename;

2) Primary key constraint :-

Primary key is used to prevent the duplication of values within rows of specified column or set of columns in a given table. It doesnot allow null values.

Primary key = unique + not null.

⊙ Syntax:- create table tablename (columnname1 datatype(size), columnname2 datatype(size) primary key, columnname3 datatype(size));

eg:- create table student (sid int(9) primary key, sname varchar(15), smarks int(10));

Scanned with CamScanner

- After creating table primary key is added as below:-

Syntax:- alter table tablename add constraint constraintname primary key(columnname);

eg:- alter table student add constraint pk1 primary key (sid);

- To drop primary key syntax is,

Syntax:- alter table tablename drop primary key;

eg:- alter table student drop primary key;

1. insert into student values (1, 'priyanka', 75);
→ record added successfully.

2. insert into student values (1, 'Riya', 90);
→ record doesnot added as duplicate value for sid.

3. insert into student values(null, 'Atul', 80);
→ error; primary key cannot be null.

Scanned with CamScanner

c] Referential Integrity constraint :-

It is used to establish parent-child relationship (primary - foreign key relationship) between two tables. Primary key is defined in parent table and foreign key is defined in child table.

P.K. F.K.

sid	name	marks	deptid
1	Riya	70	10
2	Uttraj	75	20
3	Shweta	80	10
4	Atul	90	30

child table (student table)

deptid	deptname
10	Comp
20	I.T.
30	EXTC

P.K.

Parent table (department table)

⊙ **syntax :-**

⇒ Create parent table first.

create table department (deptid int(3) primary key, deptname varchar(20));

⇒ Create child table

create table student (sid int(10) primary key, name varchar(15), marks int(20), deptid

int(10), constraint fk1 foreign key(deptid) references department(deptid));

- insert into student (5, 'Tejas', 95, 30);
record added successfully.

- insert into student (6, 'Vedika', 87, 40);
record doesnot added as deptid 40 is not present in department table.

⊙ After creating tables foreign key can be added as follows:-

alter table student add constraint fk1 foreign key (deptid) references department (deptid);

⊙ To drop foreign key

- alter table student drop foreign key fk1;
- desc student;
after dropping still mul index is there therefore use foll. syntax.
- alter table student drop index ~~foreign key~~ fk1;

Scanned with CamScanner

• Transaction is a collⁿ of statements betⁿ specific client & server. These transactions can be controlled efficiently by using mysql TCL.

* TCL Commands :-

Transaction Control language commands are used to manage transactions in the Database. These commands are used to manage the changes made to the data in a table by DML statements.

1) Commit Command :-

Commit Command is used to save any changes of any transaction permanently into the DB.

When we use any DML commands like insert, update, delete the changes made by these commands are not permanent, until the current session is closed. The changes made by these commands can be rolled back. To avoid that we use commit command to mark the changes as permanent.

The syntax for commit command is as follows :-

```
commit;
```

2) Rollback :-

This command restores the DB to last committed state. It is also used with savepoint command to jump to a savepoint in an ongoing transaction.

If we have used the update command to make changes into the DB, and realize that those changes were not required,

Scanned with CamScanner

• whenever a client gets connection with the server a new transaction will be started and the transaction ends by executing commit or rollback.

then we can use the rollback command to rollback those changes, if they were not committed using the commit command.

Syntax for rollback :-

rollback;

Syntax for rollback using savepoint,

rollback to savepoint_name;

3) Savepoint Command :-

Savepoint command is used to save a transaction temporarily, so that you can rollback to the point whenever required.

Syntax for savepoint,

savepoint savepoint_name;

Consider following eg. for TCL Commands.

1) Select * from emp;

empid	lastname	fname	sal
3	kale	Ramesh	6000
7369	Sharma	Vijay	9000
7499	Patil	Sanjay	65000
7521	Patel	Mukesh	4250

2) begin;

3) update emp set sal = sal+400 where empid = 3;

Scanned with CamScanner

4) select * from emp;

empid	lname	fname	sal
3	kale	Ramesh	6400
7369	sharma	Vijay	9000
7499	Patil	Sanjay	65000
7521	Patel	Mukesh	4250

5) rollback;

6) select * from emp;

empid	lname	fname	sal
3	kale	Ramesh	6000
7369	sharma	Vijay	9000
7499	Patil	Sanjay	65000
7521	Patel	Mukesh	4250

7) update emp set sal=sal+400 where empid=3;

8) select * from emp;

empid	lname	fname	sal
3	kale	Ramesh	6400
7369	Sharma	Vijay	9000
7499	Patil	Sanjay	65000
7521	Patel	Mukesh	4250

9) commit;

Scanned with CamScanner

10) select * from emp;

empid	lname	fname	sal
3	kale	Ramesh	6400
7369	Sharma	Vijay	9000
7499	Patil	Sanjay	65000
7521	Patel	Mukesh	4250

11) rollback;

12) select * from emp;

empid	lname	fname	sal
3	kale	Ramesh	6400
7369	Sharma	Vijay	9000
7499	Patil	Sanjay	65000
7521	Patel	Mukesh	4250

13) begin;

14) update emp set sal=sal+400 where empid=3;

15) select * from emp;

empid	lname	fname	sal
3	kale	Ramesh	6800
7369	Sharma	Vijay	9000
7499	Patil	Sanjay	65000
7521	Patel	Mukesh	4250

Scanned with CamScanner

16) update emp set sal = sal+400 where empid = 7369;

17) Select * from emp;

empid	lname	fname	sal
3	kale	Ramesh	6800
7369	sharma	vijay	9400
7499	Patil	Sanjay	65000
7521	Patel	Mukesh	4250

18) rollback;

19) select * from emp;

empid	lname	fname	sal
3	kale	Ramesh	6400
7369	sharma	vijay	9000
7499	Patil	Sanjay	65000
7521	Patel	mukesh	4250

20) start transaction;

21) Savepoint sp1;

22) delete from emp where empid=3;

23) select * from emp;

empid	lname	fname	sal
7369	sharma	vijay	9000
7499	Patil	Sanjay	65000
7521	Patel	mukesh	4250

Scanned with CamScanner

24) savepoint sp2;

25) delete from emp where empid = 7499;

26) select * from emp;

empid	lname	fname	sal
7369	Sharma	Vijay	9000
7521	Patel	Mukesh	4250

27) savepoint sp3;

28) delete from emp where empid = 7369;

29) select * from emp;

empid	lname	fname	sal
7521	Patel	Mukesh	4250

30) rollback to sp3;

31) select * from emp;

empid	lname	fname	sal
7369	Sharma	Vijay	9000
7521	Patel	mukesh	4250

32) Rollback to sp1;

33) select * from emp;

34)

empid	lname	fname	sal
3	Kale	Ramesh	6400
7369	Sharma	Vijay	9000
7499	Patil	sanjay	65000
7521	Patel	mukesh	4250

Scanned with CamScanner

34) rollback to sp2;
error (as it is already rollbacked by sp1)

(*) SQL operators :-

An operator is a reserved word or character primarily in an SQL statements to perform operations such as comparisons or arithmetic operations in where clause.

SQL supports following types of operators:-

i) Arithmetic operator
ii) Comparison operator
iii) Logical operator
iv) Set operators
v) range searching operator
vi) Pattern matching operator.

1) Arithmetic Operators :-

$X=20$; $Y=10$

operator	description	eg.
i) + (addition)	adds value in either side of the operator	$X+Y=30$
ii) - (subtraction)	subtracts right hand side from left hand operand	$X-Y=10$
iii) * (multiplication)	multiplies values on either side	$x*y=200$

Scanned with CamScanner

2) Display details of the employees by giving 5% raise in the salary.

⇒ select id, sal as old_sal, sal + sal * 0.05 as new_sal from emp;

o/p

empid	old_sal	new_sal
1	5000	7500
2	10000	15000
3	7000	10500
4	2000	3000

3) Display id and salary of the employee by dividing 2.

⇒ select empid, sal as old_sal, sal/2 as new_sal from emp;

o/p

empid	old_sal	new_sal
1	5000	2500
2	10000	5000
3	7000	3500
4	2000	1000

Scanned with CamScanner

2) Comparison Operator :-

$x=20 ; y=10$

	operator	Description	eg.
i)	= (equal to)	checks value of two operands is equal or not. If yes then returns true.	(x=y) is not true.
ii)	!= / <> (not equal to)	checks the value of two operands is equal or not, If not equal then returns true.	(x!=y) or (x<>y) is true.
iii)	> (greater than) (>= greater than equal to)	checks the value of the left (greater than equal to) operand is greater than the value of right operand, if yes then returns true.	(x>y) is true
iv)	< (less than) (<= less than equal to)	checks the value of the left operand (less than equal to) is less than the value of right operand, if yes returns true.	(x<y) is not true

Scanned with CamScanner

Queries :-

1) Find the employee whose salary is 7000.

→ Select * from emp where sal=7000;

O/P

empid	fname	lname	city	salary
3	Sanjay	Patil	mum	7000

2) Display all the employees where sal is greater than equal to 7000.

→ Select * from emp where sal >= 7000;

O/P

empid	fname	lname	city	sal
2	Vijay	mehta	Pune	10000
3	Sanjay	Patil	Mum	7000

3) Display record of all the employees by adding 2000 to it only when salary is less than 5000.

→ Select empid, fname, city, sal as old_sal, sal+2000 as new_sal where sal <5000;

O/P

empid	fname	city	old_sal	new_sal
4	Jayesh	Pune	2000	4000

Scanned with CamScanner

3) Logical operators :-

Not, And, or operators are logical operators. These three operators join two or more conditions in a where clause.

The AND operator displays a row if all conditions listed are true; The OR operator displays a row if any of the conditions listed are true. Not returns true if condition is false.

Consider foll. table :-

id	fname	lname	city	Sal.
1	Ajay	Sharma	mum	5000
2	Vijay	Mehta	pune	10000
3	Sanjay	Patil	mum	7000
4	Jayesh	Sharma	Pune	2000
5	Mangesh	Patil	kolkata	15000
6	Ajay	Patil	mum	7500

Queries :-

1) Display the info. of all the employees with first name 'sanjay' and lastname 'patil'.

→ Select * from emp where fname='sanjay' and lname='patil';

O/P

id	fname	lname	city	sal
3	Sanjay	Patil	mum	7000

Scanned with CamScanner

2) Display the information of all the employees with fname 'sanjay' or last name 'sharma'.

→ Select * from emp where fname='Sanjay' or lname='Sharma';

O/P

id	fname	lname	city	sal
1	Ajay	Sharma	mum	5000
3	Sanjay	Patil	mum	7000
4	Jayesh	Sharma	Pune	2000

3) Display the info. of all the employees with firstname Jayesh or Ajay and lname Sharm

→ Select * from emp where (fname = 'Jayesh' or fname='Ajay') and lname = 'sharma';

O/P

if or operator o/p will 1,4,6

id	fname	lname	city	sal
1	Ajay	Sharma	mum	5000
4	Jayesh	Sharma	Pune	2000

4) List all the employees having Sal less than equal to 7000 and having city 'mumbai

→ select * from emp where sal <=7000 and city='mumbai';

O/P

id	fname	lname	city	sal
1	Ajay	Sharma	mum	5000
3	Sanjay	Patil	mum	7000

Scanned with CamScanner

4) Range Searching operators :-

In, Between, Not In, Not Between these are range searching operators.

- In operator :-

It is used to check the value within a set. The IN operator is used if you know the exact value you want to return for atleast one of the columns.

Syntax :-

select columnname(s) from tablename where columnname in (value1, value2 ... value n);

not in

- Between operator :-

It selects a range of data between two values (min, max) and not between two values. These values can be number, text or dates.

Syntax :-

Select columnname(s) from tablename where columnname between value1 and value2;

not between

Scanned with CamScanner

id	fname	lname	city	sal
1	ajay	sharma	mum	5000
2	vijay	Mehta	pune	10000
3	sanjay	patil	mum	7000
4	Jayesh	Sharma	pune	2000

Queries:-

1) Display details of the employees having last name in sharma and patil.

→ select * from emp where lname in ('sharma', 'patil');

o/p

id	fname	lname	city	Sal
1	Ajay	Sharma	mum	5000
3	Sanjay	Patil	mum	7000
4	Jayesh	Sharma	pune	2000

Note:- above query can also be written as follows

select * from emp where lname='sharma' or lname='patil';

↗ if we use and operator o/p will be empty set.

2) Display details of the employees where lastname not in sharma and patil.

→ Select * from emp where lname not in ('sharma', 'patil');

alternate way to write above query is,

select * from emp where ~~fname~~ lname != 'sharma' and lname != 'patil';

↳ if we use instead of and o/p will be [1,2,3,4]

Scanned with CamScanner

O/P

id	fname	lname	city	sal
2	Vijay	Mehta	Pune	10.000

3) Display all the employees having sal between 2000 and 7000.

→ Select * from emp where sal between 2000 and 7000.

O/P

id	fname	lname	city	sal
1	Ajay	sharma	mum	5000
3	Sanjay	Patil	pune	7000
4	Jayesh	Sharma	mum	2000

alternate way to write above query :-

Select * from emp where sal >=2000 and sal <=7000;

4) Display all the employees whose salary is not between 2000 and 7000.

→ select * from emp where sal not between 2000 and 7000;

alternate way to write this query,
select * from emp where sal <2000 OR sal >7000

O/P :-

id	fname	lname	city	sal
2	vijay	Mehta	Pune	10000

Scanned with CamScanner

5) Select * from emp where fname between 'ajay' and 'sanjay';

O/P

id	fname	lname	city	sal
1	Ajay	sharma	mum	5000
3	Sanjay	Patil	mum	7000
4	Jayesh	sharma	Pune	2000

6) Select * from emp where fname between 'a' and 's';

O/P

id	fname	lname	city	sal
1	Ajay	sharma	mum	5000
4	Jayesh	sharma	Pune	2000

7) select * from emp where fname between 'a' and 'sv';

O/P

id	fname	lname	city	sal
1	Ajay	Sharma	mum	5000
3	Sanjay	Patil	mum	7000
4	Jayesh	Sharma	pune	2000

8) select * from emp where fname not between 'a' and 's'; $\boxed{<a \quad >s}$

O/P

id	fname	lname	city	sal
2	Vijay	Mehta	Pune	10000
3	Sanjay	Patil	mum	7000

Scanned with CamScanner

9) select * from emp where fname not between 'a' and 'sN';

o/p

id	fname	lname	city	sal
2	Vijay	Mehta	Pune	10000

10) Select all the employees whose lname is not in mehta and patil and having salary between 2000 & 4000.

→ select * from emp where lname not in ('mehta', 'patil') and sal between 2000 and 4000;

o/p

id	fname	lname	city	sal
4	Jayesh	sharma	Pune	2000

Scanned with CamScanner

Pattern Matching Operator (like/Not like/is null/is not null)

Syntax :- (like operator)

```
select * from tablename where columnname like 'condition';
```

↳ not like

Note :- for digit also condition will be in ' '

Two wild card characters used :-

1) % → specifies zero or more seq. of characters
2) _ → specifies only a single character

id	fname	lname	sal
1	vijay	mehta	10000
2	ajay	sharma	5000
3	sanjay	Patil	7000
4	Jayesh	sharma	2000
5	mukesh	Patel	null

emp table.

queries

① select all the employees whose firstname ends with y.

→ select * from emp where fname like '%y';

1,2,3

② select all the emp. whose fname starts with a and lname ends with a.

→ select * from emp where fname like 'a%' and lname like '%a';

2

Scanned with CamScanner

③ select all the emp. whose fname contains ja in it.

→ select * from emp where fname like '%ja%';

1,2,3,4

④ select all the emp. whose fname contain ja but doesnot start with ja.

→ select * from emp where fname like '%ja%' and fname not like 'ja%'; | 1,2,3. |

⑤ select all the emp. having firstname second letter as a.

→ select * from emp where fname '_a%'; | 3,4 |

⑥ select all the emp. where length of firstname is exactly 6.

→ select * from emp where fname like '_____';

3,4.

⑦ find fname of all the employees that have a in second position and end with y or h.

→ select * from emp where fname ~~in~~ like '_a%y' ~~and~~ or fname like '_a%h';

⑧ select * from emp where sal like '%';

1,2,3,4

Scanned with CamScanner

(9) Display emp having sal as null.

→ select * from emp where sal is null.

(10) —— Display emp having sal is not null.

→ select * from emp where sal is not null.

(*) Set operators :-

Set operators combine the result of two select statements into a single statement (table).

Queries containing set operators are called compound queries.

The set operators are :-

i) Union / union all
ii) intersect
iii) minus.

Rules :-

1) Each select statement must have same no. of columns ~~in~~ with same datatype.
2) ~~The~~ Columns should be in same order in select statements

Scanned with CamScanner

Union / Union all ⟹ All the values from both the tables will be selected.

only diff^n is union will remove duplicate records.

Syntax:-

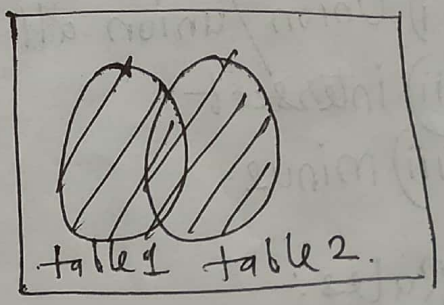
```
select columnname(s) from table 1
union all
select columnname(s) from table 2 ;
```

cid	cname	ccity
1	priyanka	mum
2	vipul	Delhi
3	vijay	Berlin

cust. table

eid	ename	ecity
1	kalpana	London
2	vijay	Berlin
3	sandhya	newyork
4	varsha	texas
5	vijay	Delhas.

emp table.



① select cname from cust
union all
select ename from emp;

cname
priyanka
vipul
vijay
kalpana
vijay
sandhya
varsha
vijay

② select cname from cust union
select ename from emp;

cname
priyanka
vipul
vijay
kalpana
sandhya
varsha

Scanned with CamScanner

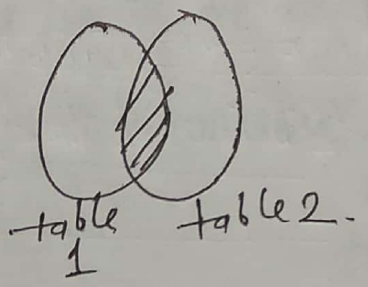
3. select ~~cid~~ ccity, cname from cust where cid>=2
union select eid, ~~ename~~ ecity from emp where eid>=3;

o/p

cid	cname
2	vipul
3	vijay
3	sandhya
4	varsha
5	vijay.

Intersect

Values which are common in both the queries will be returned.



eg:

1. select cname from cust intersect select ename from emp;

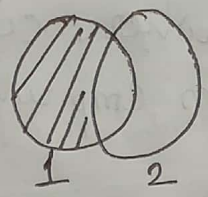
cname
vijay

2. select cname, ccity from cust intersect select ename, ecity from emp;

cname	ccity
vijay	Berlin.

Scanned with CamScanner

Minus :-



Combines two select statements and returns only those values which belong to first select statement.

eg:-

① select cname from cust minus select ename from emp;

o/p

cname
~~Vipul~~ ← Priyanka
vipul

Scanned with CamScanner

* String Manipulations functions :-

⇓

set of characters.

1. upper (columnname) - converts string into uppercase.
2. lower (columnname) - converts string into lowercase.
3. ~~length~~ replace (columnname, 'oldvalue', 'newvalue') - replaces all occurences of substring with a new substring.

   case sensitive
4. length (columnname) - returns length of the string. starts with 1
5. concat (firstcolumnname, secondcolumnname) - add two or more string together.
6. concat_ws ('separator', columnname 1, columnname 2) - add 2 strings with separator.
7. substring (string, start, length) - extracts some characters from a string.

   string from which characters are extracted

   start position (from left to right)

   no. of characters to extract (always positive)

substring (string, -(start), length)

starting from right

Scanned with CamScanner

(8) instr (string, 'substring') - returns the position of the first occurence of substring in string. (not case sensitive) starts with 1.

(9) rtrim (columnname) - removes white spaces from right side.

ltrim (columnname) - removes white spaces from left side.

trim (columnname) - removes white spaces from both the sides.

(10) right (columnname, no_of_characters) - returns the specified rightmost no. of characters.

(11) left (columnname, no_of_characters) - returns specified leftmost no. of characters.

(12) bin (value) - converts string to binary.

(13) hex (value) - converts string to hex.

(14) oct (value) - converts string to octal.

(15) bit_length (columnname) - returns length of the string in bits (each character is of 8 bits).

Scanned with CamScanner

id	fname	lname	sal	dept	city
1	VIJAY	mehta	10000	sales	pune
2	AJAY	patil	5000	sales	mumbai
3	SANJAY	sharma	7000	HR	mumbai
4	JAYESH	Patel	2000	HR	kolkata
5	MUKESH	Patil	NULL	acc	Pune

emp table.

① Write an SQL query to fetch firstname of the emp in small letters and Lastname in capital letters.

→ select upper (lname), lower (fname) from emp;

o/p

upper (lname)	lower (fname
~~Mehta~~ MEHTA	vijay
PATIL	ajay
SHARMA	sanjay
PATEL	jayesh
PATIL	mukesh

1

② find position of a in firstname.

→ select instr (fname, 'a') from emp;

7

o/p

instr (fname, 'a')
4
1
2
2
0

Scanned with CamScanner

③ find position of a in vijay.

→ select instr (fname, 'a') from emp where fname = 'vijay';

o/p

instr (fname, 'a')
4

④ replace each 'a' of lname with 'aa'. ②

→ select replace (lname, 'a', 'aa') from emp;

o/p

replace (lname, 'a', 'aa')
mehtaa
Paatil
shaarmaa
Paatel
Paatil

old value new value

⑤ display length of firstname. ③

→ select length (fname) from emp;

o/p

length (fname)
5
4
6
6
6

Scanned with CamScanner

6) Display length of frame in bits.

→ select bit_length (frame) from emp;

o/p

bit_length (frame)
~~32~~ 40
32
48
48
48

7) Display salary in reverse order.

→ select reverse (sal) from emp;

o/p

reverse (sal)
00001
0005
0007
0002
null

8) Display cities of emp after removing white spaces from left, right, both the sides.

→ select ltrim (city), rtrim (city), trim (city) from emp;

ltrim (city)	rtrim (city)	trim (city)
Pune	Pune	Pune
mumbai	mumbai	mumbai
mumbai	mumbai	mumbai
kolkata	kolkata	kolkata
Pune	Pune	Pune

Scanned with CamScanner

(9) Write an sql query to print first 3 characters of fname.

→ select substring(fname,1,3) from emp;

o/p

substring(fname,1,3)
VIJ
AJA
SAN
JAY
MUK

(10) select substring(fname,2,4) from emp;

→

substring(fname,2,4)
IJAY
JAY
ANJA
AYES
UKES

(11) select substring(fname,-6,2) from emp;

o/p

substring(fname,-6,2)
SA
JA
MU

Scanned with CamScanner

(12) select lower (reverse (fname)) from emp;

o/p

lower (reverse (fname))
yajiv
yajay
yajnas
hseyaj
hsekum

(13) ~~select~~ Display first 4 characters from right & left.

→ select right (fname, 4), left (fname, 4) from emp;

o/p

~~right~~ left (fname, 4)	~~left~~ right (fname, 4)
~~vijay~~ VIJA	IJAY
AJAY	AJAY
SANJ	NJAY
JAYE	YESH
MUKE	KESH

(14) Display id in binary.

→ select bin(id) from emp;

o/p

bin (id)
0001
0010
0011
0100
0101

Scanned with CamScanner

(15) combine & display firstname & lastname of the employees.

→ select concat (fname, lname) from emp;

o/p

Concat (fname, lname)
Vijaymehta
AJAYpatil
SANJAYsharma
JAYESHpatel
MUKESHpatil

(16) select concat_ws ('_', fname, lname) from emp;

Concat (fname, lname)
VIJAY_mehta
AJAY_patil
SANJAY_sharma
JAYESH_patel
MUKESH_patil

lower (fname)

(17) Create new column and add concatenation of fname & lnam separated by space.

→ alter table emp add column add_column varchar(40);

→ update ~~table~~ emp set add_column = concat_ws (' ', fname, lname);

Scanned with CamScanner

Group By clause :-

id	fname	lname	sal.	Deptname
1	ramesh	kale	6000	sales
2	vijay	sharma	9000	acc
3	ramesh	Patil	6000	sales
4	mukesh	Patel	4000	HR
5	ajay	Bhave	10000	marketing
6	manoj	~~Bhoste~~ sharma.	15000	acc
7	Pramod	Patil	null	HR
8	null	null	null	HR

emp. table.

Query 1 :- Find average salary of the employees.

→ select avg(sal) from emp;

o/p

avg(sal)
8333.33

Query 2 :- Find avg. salary of the employees according to the dept.

Note :- Group by clause was added to Sql because aggregate returns aggregate of all the columns, if we want to find out calculation group wise, it is not possible with aggregate functions, therefore group by clause is used.

Scanned with CamScanner

Syntax :-

```
select columnname, aggregate_fun_name (columnname)
from tablename where condition group by columnname;
```

query :- find avg. salary of the employees according to the dept.

ans :- select deptname, avg(sal) from emp group by deptname;

o/p

deptname	avg.(sal)
Sales	6000
acc	12000
HR	4000
marketing	10000

③ Display total salary of the employees according to the dept. when salary is greater than 5000.

ans :- select sum (sal), deptname from emp where sal>5000 group by deptname;

o/p

sum (sal)	deptname
12000	sales
24000	acc
10000	marketing

Scanned with CamScanner

7) select total salary of the employees according to the dept name and lastname.

ans:- select deptname, lname, sum(sal) from emp group by deptname, lname;

o/p

deptname	lname	sum(sal)
sales	kale	6000
sales	patil	6000
acc	sharma	24000
HR	patel	4000
HR	patil	null
HR	null	null
marketing	Bhave	10000.

Having clause :-

In DBMS, for putting condition on aggregate functions having clause is used instead of where clause.

Syntax:-

select columnname, aggregate_fun_name(columnname) from tablename where condition group by columnname having aggregate_fun_name (columnname) condition;

Scanned with CamScanner

eg:-

Display total salary of the employees according to the dept name where total salary is greater than 10,000.

o/p

deptname	sum(sal)
sales	12000
acc	24000

ans:- select deptname, sum(sal) from emp group by deptname having sum(sal)>10000;

Order By clause :-

order by clause is used to sort the result. It sorts result in ascending or descending order.

syntax :-

```
select columname(s) from tablename where condition
order by columnname asc/desc;
```

OR

```
select columnname, aggregate-fun-name (columnname) from
tablename where condition group by columnname having
condition order by columnname asc/desc;
```

Scanned with CamScanner

1. Display id, firstname & salary of the emp according to the descending order of firstname.

id	fname	sal
2	~~vijay~~ ramesh	9000 6000
3	ramesh	6000
7	Pramod	null
4	mukesh	4000
6	manoj	15000
5	ajay	10000
8	null	null

select id, fname, sal from emp order by fname desc;

note:- in asceding null will come first

2. Display id, deptname, fname, sal according to the ascending order of deptname & descending order of firstname.

ans:- select id, deptname, fname, sal from emp order by deptname asc, fname desc;

o/p

id	deptname	fname	sal.
2	acc	vijay	9000
6	acc	manoj	15000
4	HR	~~Patel~~ mukesh	4000
7	HR	~~Patl~~ Pramod	null
8	HR	null	null
5	marketing	ajay	10000
1	sales	ramesh	6000
3	sales	ramesh	6000

Scanned with CamScanner

③ select total salary of the emp. group by dept having total salary greather than 10000 according to the descending order of deptname.

ans:- select sum(sal), deptname from emp group by deptname having sum(sal) >10000 order by deptname desc;

o/p

sum(sal)	deptname
sales	12000
acc	24000

deptname asc;

sum(sal)	deptname
acc	24000
sales	12000

Scanned with CamScanner

Views in SQL

View is a virtual table through which only the selective portion of the data from one or more tables can be seen.

Basically view is a subset of the table created by the query. To the normal user (DB user) view is just a normal table with no. of columns and rows of data.

Views are basically used for security purposes. Views restrict users from viewing certain columns and rows. Views display only those records which are mentioned by the query.

eg:- (1) We have table emp (eid, name, loginid, pswd, sal, DOJ)

Now suppose that the administrator do not want users to access the whole data of employee details which contains some critical information (loginid, pswd) of the emp. So, he can create a view which includes (eid, name, sal, doj) as the output and allows users to view only this information.

(2) Product (pid, pname, price, brand, colour)

customer (cid, cname, address, no_of_prod_bought, location, email_id)

Now salesperson is only interested in knowing details about (pid, pname, brand, cid, no_of_prod_bought)

Scanned with CamScanner

So, here also separate view is created.

Syntax for creating View :-

```
create or replace view view_name as
select columnname(s) from tablename where condition;
```

id	lname	fname	sal	dept	city
9	Shah	Atul	7000	marketing	mum
10	Jadhav	Amol	5000	hr	Pune
11	Patel	Sayli	9000	sales	mum
12	Kadam	Tanvi	8000	Sales	mum
13	Patel	Nayan	9000	hr	kolkata
14	Patil	Vishal	8000	sales	kolkata
15	Bhave	Viraj	6000	acc	Pune

emp table.

① create view v1 as select id, fname, sal, city from emp where city='mumbai' or city='pune';

o/p ② select * from v1;

③ o/p

id	fname	sal	city
9	Atul	7k	mum
10	Amol	5k	pune
11	Sayli	9k	mum
12	Tanvi	8k	mum
15	Viraj	6k	Pune

Scanned with CamScanner

(4) Create or replace view V1 as select id, fname, sal, city from emp where (city='mum' or city='pune') and sal > 5000;

(5) select * from ~~emp;~~ V1;

(6)

id	fname	sal	city
9	Atul	7k	mum
11	sayli	9k	mum
12	Tanvi	8k	mum
15	Viraj	6k	Pune

(7) To add 1 column to view :-

alter view V1 as select id, fname, lname, city, sal from emp where city='mumbai';

(8) Update View :-

update ~~view~~ V1 set ~~so~~ id=99 where id=9;

select * from V1;

id	fname	sal	city	lname
99	Atul	7k	mum	Shah
11	sayli	9k	mum	Patel
12	Tanvi	8k	mum	Kadam
15	Viraj	6k	pune	Bhave

Note :- In emp table also, id=9 is updated by id=99, likewise if you change id=11 to id=91 in emp table same will be reflected in V1.

Scanned with CamScanner

9 update emp set id=91 where id=11;

o/p

empid	fname	sal	city	lname
99	Atul	7K	mum	shah
91	sayli	9K	mum	Patel
12	Tanvi	8K	mum	Kadam
15	Viraj	6K	Pune	Bhave

10 (i) If we add 1 record to V1.

↳ (16, 'Tejas', 19K, 'pune', 'Kadam')

In emp table respective values are added and for other columns values will be null. so dept column will have value as null.

(ii) If we add one more record to V1 as

↳ (17, 'snehal', 3K, 'mum', 'Gawli')

It gives an error as in view salary > 5000.

11 To see all the views in DB.

— Show full tables in database_name where table_type like 'VIEW';

eg:- show full tables in priyanka where talle_type like 'VIEW';

12 To drop view:-

Drop view view_name;

Scanned with CamScanner

Note:- Views are not updatable or insertable if it contains following.

1. aggregate function
2. having clause
3. group by
4. union/union all
5. subquery in where clause
6. Distinct.

You can also create view on existing view. (nested view)

query on step 9:-

create view v2 as select empid, fname, sal from v1 where sal > 7000;

empid	fname	sal
91	Sayli	9k
12	Tanvi	8k

Scanned with CamScanner

SQL JOINS

Sometimes, we have to select data from two or more tables to make our result complete. Joins are used for combining fields from two or more tables by using values common to each.

eid	fname	lname
7369	Vijay	sharma
7499	Sanjay	Patil
7521	mukesh	Patel
04	Pramod	Bhave

emp

pid	Pname	eid
234	shirt	7369
657	Top	7521
865	Tshirt	7521
878	Jeans	—

Product

① Find which employee ordered which product.

→ select emp.eid, emp.fname, emp.lname, product.pname from emp, product where emp.eid = product.eid;

o/p

eid	fname	lname	pname
7369	Vijay	sharma	shirt
7521	mukesh	Patel	Top
7521	mukesh	Patel	Tshirt

Scanned with CamScanner

② Find out the employees who have ordered shirt.

→ select emp. fname, emp. lname, product. pname, emp. eid from emp, product where emp. eid = product. eid and product. pname = 'shirt';

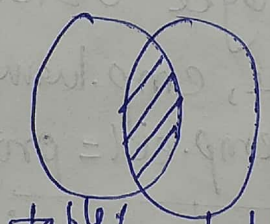
o/p

fname	lname	pname	eid
vijay	sharma	shirt	7369

Types of joins :-

① Inner Join (sometimes called as simple join)

It returns rows when there is a match in both the tables.



Syntax :-

select columnnames (from 1 & 2 table) from first_tablename inner join second_tablename on ~~first_table. keyfield = second_table. foreignkey field~~; Join_Condition;

Inner join creates a new table by combining values from both the tables based on join condition. (2)

The query compares each row of table 1 with each row of table 2, to find out all the pairs of rows which satisfies join condition.

① Find which employee ordered which product.

eg:- select emp.eid, emp.fname, emp.lname, product.pname from emp inner join product on emp.eid = ~~product.pid~~. product.eid;

o/p

empid	fname	lname	Product
7369	Vijay	sharma	shirt
7521	mukesh	Patel	Top
7521	mukesh	Patel	Tshirt

② Find which employee ordered shirt.

→ select emp.eid, emp.fname, emp.lname, product.pname from emp inner join product on emp.eid = product.pid and product.pname = 'shirt';

o/p

empid	fname	lname	product
7369	Vijay	sharma	shirt

Scanned with CamScanner

③ select emp.eid, ~~produc~~ emp.fname, emp.lname, product.pname from emp inner join product on emp.eid > product.eid;

o/p

eid	fname	lname	pname	pid
7499	Sanjay	Patil	shirt	234
7521	mukesh	Patel	Shirt	234

Equi-Join :-

The difference between equi join & inner join is that equi-join uses only equality operator in condition; but inner join can use any type ~~op.~~ of operator.

Natural Join :-

eg:- select emp.fname, emp.lname, emp.eid, product.pname, product.pid from emp natural join product;

Note:- In natural join on condition is missing as natural join automatically finds the matching columnname.

Scanned with CamScanner

Cross join :- (Cartesian join)

It performs cartesian product of rows from multiple tables. Each row of first table is mapped with each row of second table.

eg:- select emp.eid, emp.fname, emp.lname, product.pid, product.pname from emp cross join product;

eid	fname	lname	pid	pname
7369			234	
7499			234	
7521			234	
04			234	
7369			657	
7499			657	
7521			657	
04			657	
⋮				

Scanned with CamScanner

Left Outer Join :-

It returns all the rows from the table on the left even if no matching rows have been found in the table on the right, where no matches have been found in the table on the right null is returned.

table1 table2

eg:- select emp.eid, emp.fname, emp.lname, product.pid, product.pname from emp left join product on emp.eid = product.eid;

eid	fname	lname	pid	pname
7369	vijay	sharma	234	shirt
7499	sanjay	patil	null	null
7521	mukesh	Patel	657	Top
7521	mukesh	Patel	865	Tshirt
04	Pramod	bhave	null	null

Scanned with CamScanner

Right outer join :-

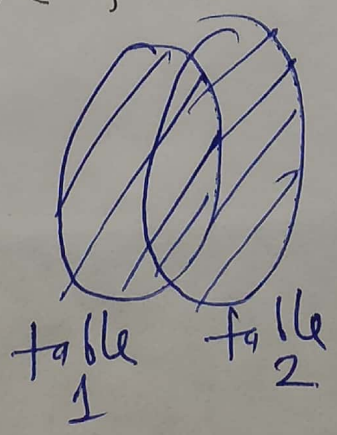
Right O.J. is opposite of left join. The right join returns the columns from the table on the right even if no matching rows found in the table on the left, where no matches have been found in the table on the left null is returned.

eg:- select emp.eid, emp.fname, emp.lname, ~~emp~~ product.pid, product.pname from emp right join product on emp.eid = product.eid;

eid	fname	lname	pid	pname
7369	vijay	sharma	234	Shirt
7521	mukesh	Patel	657	Top
7521	mukesh	Patel	865	Tshirt
–	null	null	878	Jeans

Full outer join

Full outer join returns all the rows from both the tables. If there are rows in left table that do not match with right table & vice versa, null will be returned.



Scanned with CamScanner

eid	fname	lname	pid	pname
7369	vijay	sharma	234	shirt
7499	sanjay	patil	null	null
7521	mukesh	patel	657	Top
7521	mukesh	patel	865	Tshirt
4	Pramod	Bhave	null	null
null	null	null	878	Jeans

O/P

select emp.eid, emp.fname, emp.lname, product.pid, product.pname from emp left join product on emp.eid = product.pid

union

select emp.eid, ~~emp~~, emp.fname, emp.lname, product.pid product.pname, from emp right join product on emp.eid = product.pid;

Scanned with CamScanner

Nested Query.

A subquery or nested query is a SQL query written (nested) inside a larger query.

A subquery may occur in a select clause, a from clause, a where clause.

Inner query executes first before its parent query, so that the result of the inner query can be passed to the outer query.

Syntax :-

```
Select columnname(s) from tablename where
Columnname operator (Select Columnname from tablename
where Condition);
```

Scanned with CamScanner

eid	fname	sal
1	ajay	5000
2	vijay	7000
3	sanjay	9000
4	jayesh	10000
5	viraj	2000

emp table

pid	name	eid
11	shirt	1
12	Tshirt	2
13	Top	2
14	Jeans	3

product table

① List name of the employees paid more than vijay.

→ select eid, ename from emp where sal > (select sal from emp where fname='vijay');

O/P

eid	ename
3	Sanjay
4	Jayesh

② Find out all the employees whose salary is higher than average salary.

→ select eid, fname, sal from emp where sal > (select avg(sal) from the emp);

O/P

eid	fname	Sal
2	Vijay	7000
3	Sanjay	9000
4	jayesh	10000

Scanned with CamScanner

(3) find out the employee having highest salary.

→ Select eid, fname, sal from emp where sal=(select max(sal) from emp);

O/P

eid	fname	sal
4	jayesh	10000

(4) Find second highest salary from emp.

→ Select max(sal) from emp where sal != (select max(sal) from emp);

OR

select max(sal) from emp where sal not in (select max(sal) from emp;

O/P

max(sal)
9000

(5) Find the name of the employee having second highest salary.

(can use in also)

→ select * from emp where sal = (select max(sal) from emp where sal != (select max(sal) from emp));

O/P

id	fname	sal
3	sanjay	9000

Scanned with CamScanner

⑥ Find out 3rd highest / Nth highest salary from

→ Select * from employee emp1 where (n-1) = (select count(emp2.sal) from emp2 where emp2.sal > emp1.sal);

sal
5000
7000
9000
10000
2000

emp2

sal
5000
7000
9000
10000
2000

emp1

emp2.sal > emp1.sal
5000 — 3
7000 — 2
9000 — 1
10000 — 0
2000 — 4

eg:- entire emp2 table is compared with every value of emp1.

So, emp2 is compared with 5000 and the values of emp2 which are greater than 5000 will be returned as count. So 3 values are greater than 5000 hence count is 3, and so on....

Now (n-1) = (3-1) = 2 so the salary having count 2 will be returned as output.

o/p

id	name	sal
3	Sanjay	9000

Note:- in case of lowest value condition will be (emp2.sal < emp1.sal)

Scanned with CamScanner

(7) Find details of the employees who are not related to any product.

→ select eid, fname, sal from employee where eid not in (select ~~eid~~ distinct(eid) from product);

o/p

id	fname	sal
4	jayesh	10000
5	viraj	2000

(8) Find names of the employees who have purchased shirts or tshirts.

→ select * from emp where id in (select eid from product where name='shirt' or name='tshirt');

o/p

id	fname	sal
1	ajay	5000
2	vijay	7000

(9) Increase salary of employee by 20% who are related to the product jeans or tshirts.

→ update emp set sal=sal+(sal*0.2) where eid in (select eid from product where name='jeans' or name='tshirt');

o/p

id	fname	sal
1	ajay	5000
2	vijay	8400
3	sanjay	10800
4	jayesh	12000
5	viraj	2400

Scanned with CamScanner

(10) Find all the employees who earn more than ajay and less than sanjay.

→ select id,fname from employee where sal > (select sal from emp where fname = 'ajay') and sal < (select sal from emp where fname = '~~Vinod~~ sanjay');

o/p

id	name
2	vijay
4	Jayesh

Scanned with CamScanner

Triggers :-

Prof. Priyanka Bhilare

Trigger is a special type of stored Procedure that automatically executes when an event occurs in the DB. It executes when a user tries to modify data through a DML.

Syntax of trigger (for single statement)

```
Create trigger trigger_name trigger_time
trigger_event on table_name for each row
trigger_statement;
```

Where,

trigger_name = name of the trigger
trigger_time = Before or after
trigger_event = insert, update or delete
trigger_statement = mysql statement

Syntax for multiple statements

```
mysql> delimiter //
mysql> Create trigger trigger_name trigger_time
       trigger_event on table_name for each row
        Begin
        sql statements;
        end; //
mysql> delimiter ;
```

Scanned with CamScanner

(1) Define a trigger to update a separate table whenever value in employee table has been inserted.

eid	lname	fname	sal
1			
7369			5k
7499			6k
7521			7k
			8k

emp

fname	sal

emp_insert

a) create table emp
b) create table emp_insert
c) create trigger insert_attempt after insert on emp for each row
    insert into emp_insert set fname=new.fname,
    sal=new.sal;
d) select * from emp_insert; //empty set
e) insert into emp values (2,'P', 'b', 15000);
f) select * from emp_insert;
g)

fname	sal
b	15000

Scanned with CamScanner

eg:- (2)

sid	name	marks
1	PQR	75

stud

name	marks
ABC	60

stud 1

a) create table stud;

b) create table stud1.

c) delimiter //
create trigger stud_insert after insert on stud
for each row
begin
insert into stud1 set name=new.name, marks=new.marks;

delete from stud1 where marks<60;

end; //

d) delimiter ;

e) insert into stud values (2, 'xyz', 55);
insert into stud values (3, 'LMN', 90);

f) select * from stud1;

o/p

name	marks
ABC	60
LMN	90

Scanned with CamScanner

(3)

id	name	sal
1	p	200
1	a	2000

Person

name	sal
q1	0

Person 1

a) create tables person & person1.

```
delimiter //
b) Create trigger t2 after update on person for
each row
begin
insert into person1 set name=new.name, sal=new.sal;
delete from person1 where sal<5000;
update person1 set sal=0 where person1.sal=9000;
end; //
delimiter ;
```

c) insert into person values (2,'a',5500);

o/p

id	name	sal
1	p	200
1	a	2000
2	a	5500

Person

name	sal
q1	0

person 1

d) select from both the tables.

e) update person set name='t' where name='a';

f) select from both the tables.

o/p

id	name	sal
1	p	200
1	t	2000
2	t	5500

Person

name	sal
t	5500

Scanned with CamScanner

4 (1) Create table sample (id, age, adult)

delimiter //

(2) Create trigger adult_check before insert on sample

for each row begin

if new.age > 18 then set new.adult = 1;

else set new.adult = 0;

end if ;

end //

delimiter ;

(3) insert into sample (id, age) values (2, 22);

insert into sample (id, age) where (1, 12);

(4) select * from sample;

O/P

id	age	adult
2	22	1
1	12	0

Scanned with CamScanner