॥ श्री ॥

Page No.
Date

SQL Datatypes :—

(i) char - characters of fixed length. (0-255)
(ii) VARCHAR - characters upto given length.
(iii) BLOB - binary large object.
(iv) INT - integer.
(v) TINYINT - integer.
(vi) BIGINT - integer.
(vii) FLOAT - decimal numbers with 23 digits
(viii) DOUBLE - —"— 24 to 53 digits.
(ix) BOOLEAN - 0 or 1
(x) DATE
(xi) YEAR

Types of SQL commands :—

(i) DDL (Data Defination language)
— create, alter, rename, truncate, drop.

(ii) DQL (Data Query language)
— select.

(iii) DML (Data Manipulation language)
— insert, update, delete.

(iv) DCL (Data control language)
— grant, revoke.

(v) TCL (Transmission control language)
— start transaction, commit, rollback

Database Related Queries :—

(i) create database db_name ;

(ii) — 11 — if not exists db_name ;

↳ if exits then not created only whe
not exit it will created.

(iii) drop database db_name ;

(iv) d — 11 — if exists db_name ;

(v) show databases ; — all databases

(vi) show tables ; — all tables.

Table related query :—

(i) CREATE :—

```
create table table_name (
    column_name1 datatype constraint,
    column_name2 datatype constraint,
);
```

```
create table student (
    rollno INT primary key,
    name varchar(10)
);
```

(ii) SELECT :— * means → all

```
select * from table_name ;
```

↳ display table.

(iii) INSERT :—

multiple values →

```
insert into table_name (colname1, colname2)
values (col1_v1, col2_v1),
       (col2v2, col2_v2);
```

single value →

```
insert into student values (104, "Jham");
```

```
insert into student (rollno, name)
values (101, "karan");
       (102, "arjun");
```

Keys :—

(i) Primary Key :—
- It is a column (or set of columns) in a table that uniquely identifies each row. (a unique id)
- There is only 1 PK & it should be NOT null.

(ii) Foreign Key :—
- column (or set of columns) in a table that refers to primary key of another table.
- There can be multiple FKs
- can have duplicates & null values

Constraints :—
- used to specify rules for data in a table.

(i) NOT NULL :—
- column cannot have null values
- col1 int NOT NULL.

(ii) UNIQUE :—
- all values in column are different
- col2 int UNIQUE.

(iii) PRIMARY KEY :—
- makes column unique & not null but used only for one.
- id int PRIMARY KEY

(iv) FOREIGN KEY :-

— prevent actions that would destroy links between table.

```
create table temp(
    cust_id int,
    FOREIGN KEY (cust_id) references customer(id)
);
```

table name

(v) DEFAULT :-

— set default value of a column.

```
salary int DEFAULT 25000
```

(vi) CHECK :-

- it can limit the values allowed in a column.

```
create table city (
    id INT PRIMARY KEY,
    city varchar(50),
    age int,
    CONSTRAINT age_check CHECK (age >= 18 AND
                                city = "Delhi")
);
```

constraint name (it can be written or skip)

now we can not add age < 18 & city not contain delhi.

```
create table newTab(
    age int CHECK (age ≥ 18)
);
```

Where Clause :— (condition)

— to define some conditions

```
Select col1, col2 FROM table_name
WHERE conditions ;
```

eg. — (i) select * from student WHERE marks >80 ;

- Using operators in WHERE :—

(i) Arithematic Operators :— (ii) Comparison operators
— (+, -, *, /, %) — (=, !=, >, ≥, <, ≤)
↳ gives remainder.

(iii) Logical operator :— (iv) Bitwise Operator
— (AND, OR, NOT, IN, BETWEEN, — (&, |)
ALL, LIKE, ANY)

Operators :—

(i) AND (to check for both conditions to be true)

```
select * from student WHERE marks >80 AND city =
                                        "Mumbai";
```

(ii) OR (to check for one of the conditions to be true)

```
select * from student WHERE marks >90 OR city="Pune";
```

(iii) BETWEEN (selects for given range)

```
select * from student where marks BETWEEN 80 AND 90;
```

(iv) IN (matches any values in the list)

select * from student WHERE city IN ("Delhi", "Pune");

(v) NOT (to negate given condition)

select *from student WHERE city NOT IN ("Pune", "Delhi");

LIMIT clause :—

- sets upper limit on number of (tuples) rows to be returned.

select * from student LIMIT 3 ; 3 ही student का data आएगा.

select col1, col2 FRom table_name
LIMIT number;

select * from student
where marks >75
limit 3;

ORDER BY clause :—

— to sort in ascending (ASC) or descending order (DESC)

select * from student
ORDER BY city ASC;

select col1, col2 from table_name
ORDER BY col_name(s) ASC;

Q) we want top 3 students rankers ?

select * from student
ORDER BY marks DESC
LIMIT 3;

select city from student
GROUP BY city;

output:- Pune
Mumbai
Delhi

Aggregate Functions :—
- perform calculations on set of values, & return single value.

(i) COUNT ()
(ii) MAX ()
(iii) MIN ()
(iv) SUM ()
(v) AVG ()

(i) Get maximum marks

```
select ~~from~~ max (marks)
from student;
```

(ii) Get average marks.

```
select avg (marks)
from student;
```

GROUP BY clause :—
- group rows that have the same values into summary rows.
- it collects data from multiple records & groups the result by one or more column.

* Generally we use group by with some aggregation function.

Q) count no. of students in each city?—

```
select city, count (rollno)
from student
GROUP BY city;
```

दोन्ही same हव नाहीतर error.

count of student.

Delhi	3
Mumbai	2
Pune	1

HAVING Clause :—
Groups के ऊपर conditions.
- similar to where i.e. applies some condition on rows.
- used when we want to apply any condition after grouping.

SET. SQL-SAFE_UPDATE = 0; → safe mode off.

Page No.
Date

Q) count number of students in each city where max marks cross 90.

```
select count (name), city
from student
GROUP BY city
HAVING max (marks) > 90;
```

General order :—

```
SELECT columns
FROM table_name
WHERE condition
GROUP BY column(s)
HAVING condition
ORDER BY column(s) ASC;
```

rows ← WHERE applies

Group ← HAVING applies.

Table Related Query :—

(i) Update :—

— to update existing rows data.

```
UPDATE table_name
SET col1 = val1, col2 = val2
WHERE condition;
```

eg.

```
update student
SET grade = "O"
where grade = "A";
```

(ii) (a) DELETE :—

— to delete existing rows data.

```
DELETE from table_name
where condition;
```

Foreign key :—

```
create table dept (
    id int primary key,
    name varchar (50)
);
```

id	name
101	x
102	A

parent table

child table.

```
create table teacher (
    id int primary key,
    name varchar(50),
    dept_id int,
    foreign key (dept_id) references dept(id)
);
```

id	name	dept_id
101	A	102
102	B	101
103	C	102

- Cascading for Foreign Keys :—

(i) On ~~Update~~ Delete cascade :—

— when we create foreign key wing this option, it deletes the referencing rows in the child table when the referenced row is deleted in the parent table which has primary key.

(ii) On Update cascade :—

— when we create a foreign key wing UPDATE CASCADE the referencing rows are updated in the child table when the referenced row is updated in the parent table which has primary key.

```
Create table student (
    id int primary key,
    course_id int,
    foreign key (course_id) references course (id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

(iii) Alter :—

- to change the schema (structure)
  ↳ column related info (column, datatype, constraints)
  data.

① ADD column :—

```
ALTER TABLE table_name
Alter ADD COLUMN column_name datatype constraint;
```

② DROP column :—

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

③ RENAME Table :—

```
ALTER TABLE table_name
RENAME TO new_table_name;
```

④ CHANGE column :— (rename)

```
ALTER TABLE table_name
CHANGE COLUMN old_name new_name new_data-
type new_constraint;
```

⑤ MODIFY column (modify datatype & constraint) :—

```
ALTER TABLE table_name
MODIFY col_name new_datatype new_constraint;
```

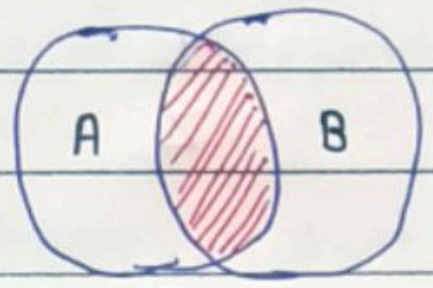
DROP - Table को ही delete कर देगा।
TRUNCATE - Table के data को.

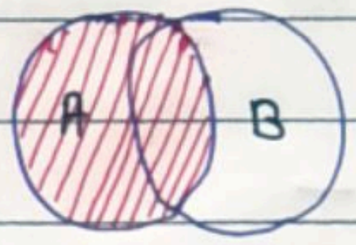
(iv) Truncate :- table के सारे data को delete कर देता है।
- to delete table's data

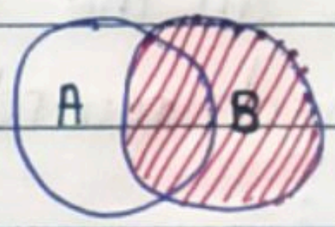
```
TRUNCATE TABLE table_name;

UPDATE student
```

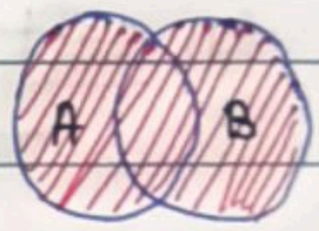
Joins in SQL :-
- used to combine rows from two or more tables, based on related column between them.

• Types :-

(i) Inner Join 

(ii) Outer Join
(a) Left Join → 

(b) Right Join → 

(c) Full Join :- 

(i) Inner Join :-
- returns records that ~~are~~ have matching values in both tables.
- syntax :-

```
SELECT  column(s)
FROM    tableA
INNER JOIN  table B
ON  tableA.col_name = tableB.col_name;
```

Alias :- alternative name

ie. From student as S

- eg.

student

stud_id	name
101	X
102	Y
103	Z

course

stud-id	course
102	E
105	M
103	S
107	CS

select * from student
Inner Join course
ON student.stud_id = course.stud-id;

• Result :-

stud_id	name	course
102	Y	E
103	Z	S

(ii) outer Join :—

(a) left Join :—

— returns all the records from the left table, & the matched records from right table.

— syntax :-

```
SELECT column(s)
FROM tableA
LEFT JOIN tableB
ON tableA.col_name = tableB.col_name;
```

(b) Right Join :—

— return all the records from right table, & the matched records from left table.

— syntax :—

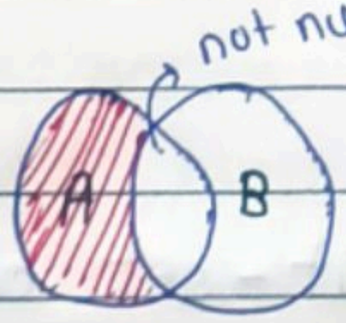
```
SELECT column(s)
FROM tableA
Right Join tableB
ON tableA.col_name = tableB.col_name;
```

(c) Full Join :—

- Returns all records when there is a match in either left or right table.
- syntax :-

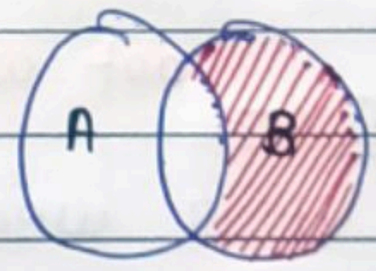
```
SELECT * FROM student as s
LEFT JOIN course as c
ON s.stud_id = c.stud_id
UNION
SELECT * FROM student as a
RIGHT JOIN course as b
ON a.stud_id = b.stud_id;
```

Left Exclusive Join




```
SELECT *
FROM student as a
LEFT JOIN course as b
ON a.id = b.id
WHERE b.id IS NULL;
```

Right Exclusive Join.




```
SELECT *
FROM student as a
RIGHT JOIN course as b
ON a.id = b.id
WHERE b.id IS NULL;
```

Self Join :—

- regular join but table is joined with itself.

• syntax :-

```
SELECT column(s)
FROM table as a
JOIN table as b
ON a.col_name = b.col_name;
```

- eg.

Employee

id	name	manager_id
101	X	103
102	Y	104
103	Z	null
104	A	103

```
select a.name as manager_name,
b.name
FROM employee as a
JOIN employee as b
ON a.id = b.manager_id;
```

* Result :-

manager_name	name
Z	A
Z	X
A	Y

Union :- not contains duplicates.

— it is used to combine the result-set of two or more SELECT statements.

— gives unique records.

• To use it :—

(i) every select should have same no. of column

(ii) columns must have similar datatype

(iii) columns in every select should be in same order.

Page No.
Date

- Syntax :-

```
SELECT column(s) FROM tableA
UNION
SELECT column(s) FROM tableB
```

Union All :- gives duplicated also.

- syntax :-

→ same UNION ALL.

can be written in (i) select (ii) FROM (iii) WHERE

mostly used

SQL sub queries / Nested queries :-

- query within another SQL query.
- involves two SELECT statements.

- Syntax :-

```
SELECT column(s)
FROM table_name
WHERE col_name operator
(subquery);
```

Q) Get names of all students who scored more than class average.

rollno	name	marks
101	anil	78
102	bhumika	98
103	chetan	85
104	dhruv	96
105	emanuel	92
106	Farah	82

Step 1: Find avg of class
Step 2: Find name of students with marks > avg.

```
select name
from student
where marks > (select avg(marks) from student);
```

DROP VIEW view1;

Page No.
Date

MySQL Views :— → temporary table
a) c) version of real table

— virtual table based on the result-set of an sql statement

— always shows up-to-date data. The database engine recreates the view, every time a user queries it.

• Syntax :—

```
CREATE VIEW view1 AS
SELECT rollno, name FROM student;

SELECT * FROM view1;
```