
MySQL®

SQL Language

1. DDL(Data Definition Language)
2. DML(Data Manipulation Language)
3. DRL/DQL (Data Retrieval Language/Data Query Language)
4. TCL (Transaction Control Language)
5. DCL (Data Control Language)

SQL Commands

- DDL (Data Definition Language):
  - CREATE, ALTER, DROP, TRUNCATE, RENAME
- DML (Data Manipulation Language):
  - INSERT, UPDATE, DELETE
- DRL/DQL (Data Retrieval /Data Query Language):
  - SELECT
- TCL(Transaction Control Language)
  - COMMIT, ROLLBACK, SAVE POINT
- DCL(Data Control Language)
  - GRANT, REVOKE

Create Database/Schema

- CREATE DATABASE mydb;
- DROP DATABASE mydb;

- CREATE SCHEMA mydb;
- DROP SCHEMA mydb;

- CREATE DATABASE IF NOT EXISTS mydb;

Creating Table

- create table <<TABLE NAME>>(col1 datatype,col2 datatype, col3 datatype.....);

- Example:
- USE mydb;
- CREATE TABLE STUDENT(SNO INT (5),SNAME VARCHAR(15),MARKS INT(3));

Inserting data into table

- INSERT INTO <<TABLE NAME>> VALUES(VAL1,AL2,VAL3....);
- Example:
- USE mydb;
- INSERT INTO STUDENT VALUES(101,'KIRAN',80);
- INSERT INTO STUDENT(SNAME,SNO,MARKS) VALUES('RAM',102,60);
- INSERT INTO STUDENT VALUES(103,'KRISHNA',NULL);

Selecting Rows from a table

- USE hr;
- SELECT * FROM EMPLOYEES;SELECT EMPLOYEE_ID,FIRST_NAME,SALARY FROM EMPLOYEES;
- SELECT EMPLOYEE_ID EMPID,FIRST_NAME FNAME,SALARY+300 SAL FROM EMPLOYEES;

SQL Data types

- Numeric
- Text
- Date/time

Numeric Data Types

TINYINT	-128 to 127 normal 0 to 255 UNSIGNED.
SMALLINT	-32768 to 32767 normal 0 to 65535 UNSIGNED.
MEDIUMINT	-8388608 to 8388607 normal 0 to 16777215 UNSIGNED.
INT	-2147483648 to 2147483647 normal 0 to 4294967295 UNSIGNED.
BIGINT	-9223372036854775808 to 9223372036854775807 normal 0 to 18446744073709551615 UNSIGNED.
FLOAT	A small approximate number with a floating decimal point.
DOUBLE(,)	A large number with a floating decimal point.
DECIMAL(,)	A DOUBLE stored as a string , allowing for a fixed decimal point. Choice for storing currency values.

Text Data Types

CHAR( )	A fixed section from 0 to 255 characters long.
VARCHAR( )	A variable section from 0 to 255 characters long.
BLOB	A string with a maximum length of 65535 characters.
MEDIUMTEXT	A string with a maximum length of 16777215 characters.
MEDIUMBLOB	A string with a maximum length of 16777215 characters.
LONGTEXT	A string with a maximum length of 4294967295 characters.
LONGBLOB	A string with a maximum length of 4294967295 characters.

Date & Time

DATE	YYYY-MM-DD
DATETIME	YYYY-MM-DD HH:MM:SS
TIMESTAMP	YYYYMMDDHHMMSS
TIME	HH:MM:SS

Where clause

- Used for selecting the rows based on condition.(Filtering the rows using where condition)
- USE hr;
- SELECT * FROM EMPLOYEES;
- SELECT * FROM EMPLOYEES WHERE SALARY>3000;
- SELECT * FROM EMPLOYEES WHERE SALARY<=3000;
- SELECT * FROM EMPLOYEES WHERE DEPARTMENT_ID=30;
- SELECT * FROM EMPLOYEES WHERE COMMISSION_PCT is null;
- SELECT * FROM EMPLOYEES WHERE FIRST_NAME='Jennifer';
- SELECT DISTINCT DEPARTMENT_ID FROM EMPLOYEES;
- SELECT DISTINCT * FROM EMPLOYEES;

Logical Operators (AND, OR, NOT)

- USE hr;
- SET SQL_SAFE_UPDATES = 0;
- SELECT * From EMPLOYEES;
- SELECT * FROM EMPLOYEES WHERE SALARY>15000 AND JOB_ID='AD_VP’;
- SELECT * FROM EMPLOYEES WHERE SALARY>15000 OR JOB_ID='AD_VP’;
- SELECT * FROM EMPLOYEES WHERE NOT FIRST_NAME='David';

Between & IN Operators

- Between →Used to display the rows which is following in the range of values.
- Not Between
- USE hr;
- SELECT * FROM EMPLOYEES WHERE SALARY BETWEEN 10000 AND 12000;
- SELECT * FROM EMPLOYEES WHERE SALARY NOT BETWEEN 10000 AND 12000;
- IN → IN operator return the rows when the values are matching in the list
- Not In
- SELECT * FROM EMPLOYEES WHERE SALARY=3400 OR SALARY=2500 OR SALARY=3000;
- SELECT * FROM EMPLOYEES WHERE SALARY IN(3400,2500,3000);
- SELECT * FROM EMPLOYEES WHERE SALARY NOT IN(3400,2500,3000);

Pattern Matching operators (whiled card characters)

- % --> many characters
- _ --> single character
- SELECT * FROM EMPLOYEES WHERE FIRST_NAME LIKE 'S%';
- SELECT * FROM EMPLOYEES WHERE FIRST_NAME LIKE '%r';
- SELECT * FROM EMPLOYEES WHERE FIRST_NAME LIKE 'S%r';
- SELECT * FROM EMPLOYEES WHERE FIRST_NAME LIKE '%m%’;
- SELECT * FROM EMPLOYEES WHERE FIRST_NAME NOT LIKE 'S%';
- SELECT * FROM EMPLOYEES WHERE FIRST_NAME LIKE '%e_';
- SELECT * FROM EMPLOYEES WHERE FIRST_NAME LIKE '___';

DDL Commands

DDL Commands(Data Definition Language)

1) CREATE

2) ALTER

3) DROP

4) TRUNCATE

5) RENAME

Create & Alter

- CREATE is used to create database objects(Database, Table, views, synonymes etc...)
- ALTER
  - Adding a new column
  - Dropping the existing column
  - Modifying the existing column (Increase/Decrease size of the column & change the data type of the column)
  - Renaming a column

- use mydb;
- SELECT * FROM STUDENT;
- DESCRIBE STUDENT;
- Adding a new column
- ALTER TABLE STUDENT ADD(grade varchar(2));
- Dropping a column from table
- ALTER TABLE STUDENT DROP COLUMN grade;
- Modifying the existing column
  - We can increase/decrease the size of the column.
  - We can decrease the column size ONLY when existing column values can fit into new size..
  - Column should be empty should be empty to change its data type.
- ALTER TABLE STUDENT MODIFY COLUMN SNAME VARCHAR(20);
- Renaming a column
- ALTER TABLE STUDENT RENAME COLUMN SNAME TO STUNAME;

DROP, TRUNCATE & DELETE

- DROP:
- Used to drop the table definition with data
- DROP TABLE STUDENT;
- TRUNCATE
- Used to remove all the rows from the table.
- TRUNCATE TABLE STUDENT;
- DELETE
- Used for delete all the rows from the table.
- DELETE FROM TABLE;

Differences between Drop, Truncate & delete

- DROP TABLE STUDENT; -- Drops the structure & data
- TRUNCATE TABLE STUDENT; -- Removes the all the rows permanently
- DELETE FROM STUDENT; -- Removes the all the rows temporarily. We can Roll back the rows.
  - To roll back record we need to apply AUTO COMMIT Zero
  - SET SQL_SAFE_UPDATES = 0;
  - SET autocommit=0;

- RENAME TABLE
- Used for changing the name of the table
- RENAME TABLE STUDENT TO STU;

MySQL Functions

MySQL Functions

1) Strings functions - operate on string data types
2) Numeric functions - operate on numeric data types
3) Date functions - operate on date data types
4) Aggregate functions - operate on all of the data types and produce summarized result sets.

String Functions

- Upper(): converts into upper case letters.
- SELECT UPPER(First_name) from employees;
- Lower() : converts into lower case letters
- SELECT LOWER(First_name) from employees;
- Length(): return the length of string.
- SELECT LENGTH('oracle’);
- SELECT * FROM EMPLOYEES WHERE LENGTH(FIRST_NAME)=4;
- TRIM(): Removes the specified characters from both sides.
- SELECT TRIM(' ORACLE ') FROM DUAL;
- SELECT TRIM('z' from 'zzoraclezz') from dual;
- NSTR(): Returns the position of the character within a string
- SELECT INSTR('ORACLE','E');

String Functions…

- SUBSTR() /SUBSTRING(): Returns the substring of the string.
- SELECT SUBSTR('ORACLE',2,3); -- RAC
- SELECT SUBSTR('ORACLE',3,3); -- ACL
- SELECT SUBSTR('ORACLE',4,3); -- CLE
- SELECT SUBSTRING('ORACLE',2,3); -- RAC
- SELECT SUBSTRING('ORACLE',3,3); -- ACL
- SELECT SUBSTRING('ORACLE',4,3); -- CLE
- USE hr;
- SELECT SUBSTR(FIRST_NAME,1,3) FROM EMPLOYEES;
- CONCAT(): To join two strings.
- SELECT CONCAT('ORACLE','TRAINING');
- USE hr;
- SELECT CONCAT(FIRST_NAME,LAST_NAME) ENAME FROM EMPLOYEES;

Ref: https://dev.mysql.com/doc/refman/5.5/en/string-functions.html

Numeric Functions

- SELECT ABS(-40);
- SELECT ABS(40);
- select SQRT(25);
- select MOD(10,3);
- select power(2,5);
- TRUNCATE() function truncates a number to the specified number of decimal places.
- select TRUNCATE(40.1234,3); -- 40.123
- select TRUNCATE(40.1234,2); -- 40.12
- Select TRUNCATE(6876,-1); -- 6870
- Select TRUNCATE(6876,-2); -- 6800
- Select TRUNCATE(68763456,-5); -- 68700000

Numeric Functions…

- GREATEST() & LEAST(): returns greatest, least values in the provided values.
- SELECT GREATEST(100,200,300,400,500);
- SELECT LEAST(100,200,300);

- Ref: https://dev.mysql.com/doc/refman/5.5/en/numeric-functions.html

Date Functions

- CURDATE() /CURRENT_DATE() function returns the current date.
- SELECT CURDATE();
- SELECT CURRENT_DATE();

- CURTIME()/CURRENT_TIME() function returns the current time.
- SELECT CURTIME();
- SELECT CURRENT_TIME();

- NOW() function returns the current date and time.
- SELECT NOW();

- SYSDATE() function returns the current date and time.
- SELECT SYSDATE();

- MONTH() function returns the month part for a given date (a number from 1 to 12).
- SELECT MONTH("2019-05-19"); -- 5

- YEAR() function returns the year part for a given date (a number from 1000 to 9999).
- SELECT YEAR("2019-05-19"); -- 2019

- DAY() function returns the day of the month for a given date (a number from 1 to 31).
- SELECT DAY("2019-05-19");

- Ref https://dev.mysql.com/doc/refman/5.5/en/date-and-time-functions.html

Queries on Date Functions

- Display employees who are joined in 1987.
- SELECT * FROM EMPLOYEES WHERE YEAR(HIRE_DATE)="1987";

- Display employees who are joined in June.
- SELECT * FROM EMPLOYEES WHERE MONTH(HIRE_DATE)="6";
- SELECT * FROM EMPLOYEES WHERE MONTHNAME(HIRE_DATE)="JUNE";

Aggregate Functions

- Aggregate Functions are all about performing calculations on multiple rows of a single column of a table and returning a single value.
- USE hr;
- SELECT AVG(SALARY) FROM EMPLOYEES;
- SELECT SUM(SALARY) FROM EMPLOYEES;
- SELECT MIN(SALARY) FROM EMPLOYEES;
- SELECT MAX(SALARY) FROM EMPLOYEES;
- SELECT COUNT(*) FROM EMPLOYEES;

Group By clause

Group By clause

- Group By clause:
- The GROUP BY clause groups records into summary rows.
- GROUP BY returns one records for each group.
- GROUP BY typically also involves aggregates: COUNT, MAX, SUM, AVG, etc.
- GROUP BY can group by one or more columns.

- SELECT DEPARTMENT_ID,SUM(SALARY) FROM EMPLOYEES GROUP BY DEPARTMENT_ID;
- SELECT DEPARTMENT_ID,AVG(SALARY) FROM EMPLOYEES GROUP BY DEPARTMENT_ID ;
- SELECT DEPARTMENT_ID, MAX(SALARY),MIN(SALARY) FROM EMPLOYEES GROUP BY DEPARTMENT_ID;
- SELECT JOB_ID, COUNT(*) FROM EMPLOYEES GROUP BY JOB_ID;
- All the columns in the select list should include in group by clause.
- SELECT DEPARTMENT_ID, JOB_ID, SUM(SALARY) FROM EMPLOYEES GROUP BY DEPARTMENT_ID, JOB_ID;
- SELECT DEPARTMENT_ID,SUM(SALARY),FIRST_NAME FROM EMPLOYEES GROUP BY DEPARTMENT_ID; // Invalid query

Having & Order by clause

- Having clause: Having clause is used to filter the output from the group by clause.
- SELECT DEPARTMENT_ID, SUM(SALARY) FROM EMPLOYEES GROUP BY DEPARTMENT_ID HAVING SUM(SALARY)>20000;
- SELECT DEPARTMENT_ID, SUM(SALARY) FROM EMPLOYEES WHERE DEPARTMENT_ID<>50 GROUP BY DEPARTMENT_ID;
- Order By clause: Order by clause is used to arrange the rows in a table (ascending or descending order).
- SELECT * FROM EMPLOYEES ORDER BY DEPARTMENT_ID DESC;
- SELECT * FROM EMPLOYEES ORDER BY SALARY;

Order Of Execution

- Where-> Group by->Having-> Order By

```
SELECT column-names
  FROM table-name
    WHERE condition
        GROUP BY column-names
            Having condition
              ORDER BY column-names
```

- SELECT DEPARTMENT_ID,SUM(SALARY) FROM EMPLOYEES GROUP BY DEPARTMENT_ID HAVING SUM(SALARY)>20000 ORDER BY SUM(SALARY);

- SELECT DEPARTMENT_ID,SUM(SALARY) FROM EMPLOYEES WHERE DEPARTMENT_ID<>100 GROUP BY DEPARTMENT_ID HAVING SUM(SALARY)>20000 ORDER BY SUM(SALARY)DESC ;

UNION Operator

UNION & UNION ALL

- The UNION operator is used to combine the result-set of two or more SELECT statements.
- Each SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in each SELECT statement must also be in the same order

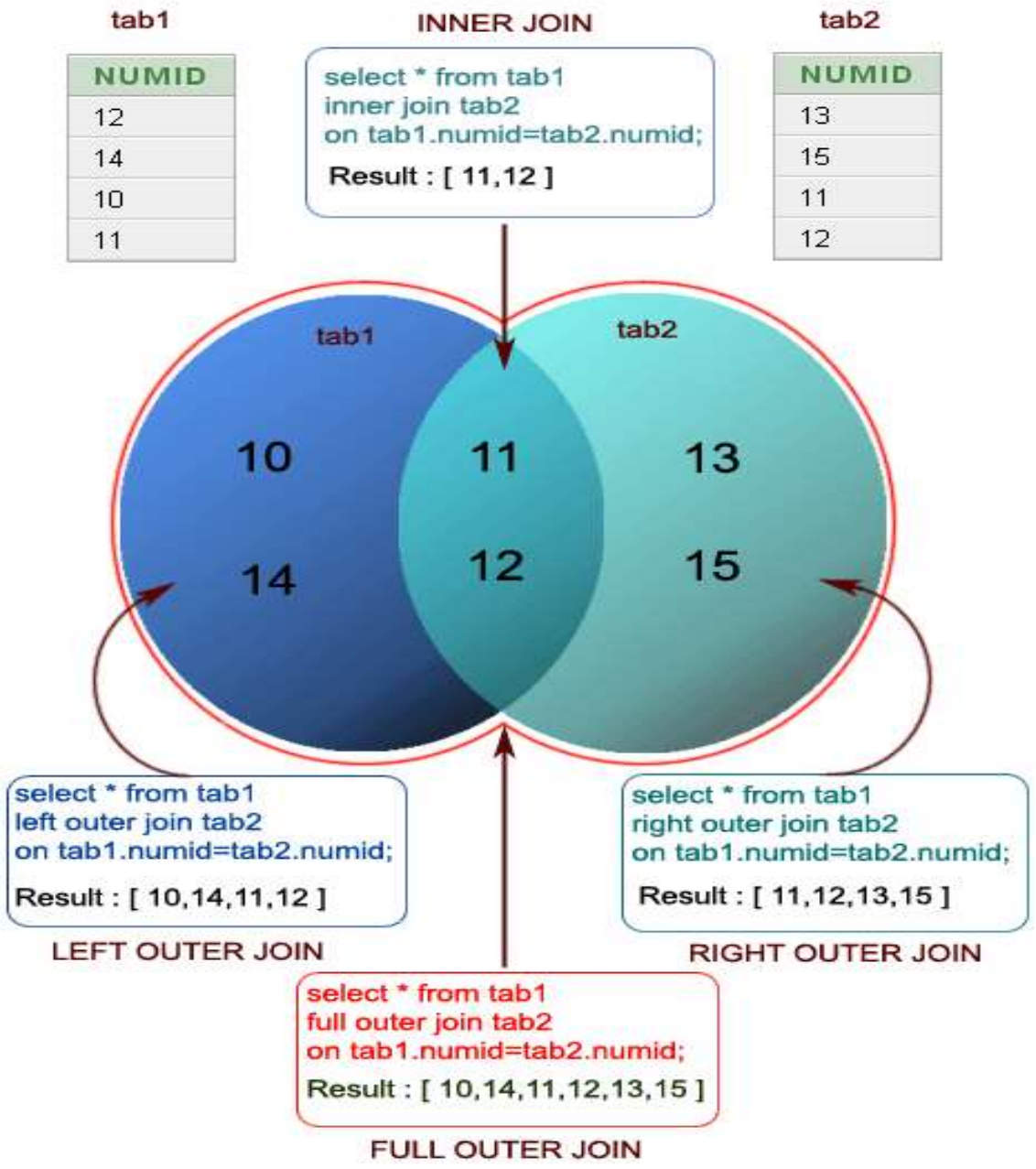
UNION & UNION ALL..

- CREATE TABLE A(SNAME varchar(10), NUM INT(2));
- CREATE TABLE B(NUM INT(2),GRADE VARCHAR(3));
- INSERT INTO A VALUES('ABC',10);
- INSERT INTO A VALUES('XYZ',11);
- INSERT INTO A VALUES('PQR',12);
- INSERT INTO A VALUES('MNO',14);
- COMMIT;
- INSERT INTO B VALUES(11,'A');
- INSERT INTO B VALUES(12,'B');
- INSERT INTO B VALUES(13,'C');
- INSERT INTO B VALUES(15,'B');
- COMMIT

- Displays all the records from multiple tables without duplicates.
- SELECT NUM FROM A UNION SELECT NUM FROM B;
- Displays all the records from multiple tables including duplicates.
- SELECT NUM FROM A UNION ALL SELECT NUM FROM B;

SQL Joins

- Joins help retrieving data from two or more database tables.
- The tables are mutually related using primary and foreign keys.
- Types of Joins
  1. Equi Join/Inner Join/Simple Join
  2. Right Join
  3. Left Join
  4. Full Join
  5. Self Join




```
CREATE TABLE TAB1(NUMID INT(3));
CREATE TABLE TAB2(NUMID INT(3));

INSERT INTO TAB1 VALUES(10);
INSERT INTO TAB1 VALUES(11);
INSERT INTO TAB1 VALUES(12);
INSERT INTO TAB1 VALUES(14);

INSERT INTO TAB2 VALUES(11);
INSERT INTO TAB2 VALUES(12);
INSERT INTO TAB2 VALUES(13);
INSERT INTO TAB2 VALUES(15);

SELECT * FROM TAB1;
SELECT * FROM TAB2;
```

Equi Join/Inner Join

- Inner/Equi join (Returns Only matched records from Tab1 & Tab2)
  - SELECT * FROM TAB1 INNER JOIN TAB2 ON TAB1.NUMID=TAB2.NUMID;
- Right outer join (Returns matched records+ unmatched from right table Tab1)
  - SELECT * FROM TAB1 RIGHT JOIN TAB2 ON TAB1.NUMID=TAB2.NUMID;
- Left outer join (Returns matched records+ unmatched from right table Tab2)
  - SELECT * FROM TAB1 LEFT JOIN TAB2 ON TAB1.NUMID=TAB2.NUMID;
- Full outer join (Returns matched records+ unmatched from both tables) -- not supported in mysql
  - SELECT * FROM TAB1 FULL JOIN TAB2 t2 ON t1.NUMID=t2.NUMID;

Inner Join/Self-Join

- Returns Only matched records from both the tables
- SELECT

  EMPLOYEE_ID,FIRST_NAME,JOB_ID,DEPT.DEPARTMENT_ID,DEPARTMENT_NAME,LOCATION_ID

  FROM EMPLOYEES EMP INNER JOIN DEPARTMENTS DEPT ON

  (EMP.DEPARTMENT_ID=DEPT.DEPARTMENT_ID);

Right Join

- Returns matched records+ unmatched from right table employees
- SELECT

  EMPLOYEE_ID,FIRST_NAME,JOB_ID,DEPT.DEPARTMENT_ID,DEPARTMENT_NAME,LOCATION_ID

  FROM EMPLOYEES EMP RIGHT JOIN DEPARTMENTS DEPT ON

  (EMP.DEPARTMENT_ID=DEPT.DEPARTMENT_ID);

Left Join

- Returns matched records+ unmatched from left table departments
- SELECT

  EMPLOYEE_ID,FIRST_NAME,JOB_ID,DEPT.DEPARTMENT_ID,DEPARTMENT_NAME,LOCATION_ID

  FROM EMPLOYEES EMP LEFT JOIN DEPARTMENTS DEPT ON

  (EMP.DEPARTMENT_ID=DEPT.DEPARTMENT_ID);

Self-Join

- Join with a table with the same table
- Query: Print Employees details who is Manager of other employees.
- SELECT E.EMPLOYEE_ID,M.MANAGER_ID,E.FIRST_NAME,E.JOB_ID,M.FIRST_NAME FROM EMPLOYEES E, EMPLOYEES M WHERE EMPLOYEE_ID=M.MANAGER_ID;

Sub Queries

- Sub Query is a Query within a Query.
- Sub Query contains 2 parts.
  1. Outer Query
  2. Inner Query
- The output of inner query is become input of outer query.
- 2 Types of Sub Queries:
  1. Single row sub query , <=, >=, !=
  2. Multi row Sub Query. IN,ANY,ALL

Single Row Sub Queries

- Display employees whose salary is less than the of Ellen
- SELECT * FROM EMPLOYEES WHERE SALARY<(SELECT SALARY FROM EMPLOYEES WHEREFIRST_NAME='Ellen’);
- 2nd max salary from employee
- SELECT MAX(SALARY) FROM EMPLOYEES WHERE SALARY<(SELECT MAX(SALARY) FROM EMPLOYEES);
- 3rd Maximum salary
- SELECT MAX(SALARY) FROM EMPLOYEES WHERE SALARY<(SELECT MAX(SALARY) FROM EMPLOYEES WHERE SALARY<(SELECT MAX(SALARY) FROM EMPLOYEES));

Single Row Sub Queries…

- Find the salary of employees whose salary is greater than the salary of employee whose EMPLOYEE_ID 150.

- SELECT SALARY FROM EMPLOYEES WHERE SALARY>(SELECT SALARY FROM EMPLOYEES WHERE EMPLOYEE_ID=150);

- Display the employees who all are earning the highest salary.

- SELECT * FROM EMPLOYEES WHERE SALARY=(SELECT MAX(SALARY) FROM EMPLOYEES);

Multi Row Sub Queries

- Display employees whose salary is equal to the salary of the at least one employee in department id 30.
- SELECT * FROM EMPLOYEES WHERE SALARY IN (SELECT SALARY FROM EMPLOYEES WHERE DEPARTMENT_ID=30);
- Display the employees whose salary is greater than the at least on employee in department id 30.
- SELECT * FROM EMPLOYEES WHERE SALARY>ANY(SELECT SALARY FROM EMPLOYEES WHERE DEPARTMENT_ID=30);
- Display the employees whose salary is less than the at least on employee in department id 30.
- SELECT * FROM EMPLOYEES WHERE SALARY<ANY(SELECT SALARY FROM EMPLOYEES WHERE DEPARTMENT_ID=30);

Multi Row Sub Queries…

- Query to get department name of the employee.
- SELECT FIRST_NAME,EMPLOYEE_ID,DEPARTMENT_ID,(SELECT DEPARTMENT_NAME FROM DEPARTMENTS WHEREEMPLOYEES.DEPARTMENT_ID=DEPARTMENTS.DEPARTMENT_ID)DNAME FROM EMPLOYEES;
- List out the employees who are having salary less than the maximum salary and also having hire date greater than the hire date of an employee who is having maximum salary.
- SELECT EMPLOYEE_ID,FIRST_NAME SALARY,HIRE_DATE FROM EMPLOYEES WHERE SALARY<

  (SELECT MAX(SALARY) FROM EMPLOYEES) AND HIRE_DATE>

  (SELECT HIRE_DATE FROM EMPLOYEES WHERE SALARY=

  (SELECT MAX(SALARY) FROM EMPLOYEES));

Integrity Constraints

Integrity Constraints

- SQL constraints are used to specify rules for data in a table.
- Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement.
- SQL Constraints
- **NOT NULL** - Ensures that a column cannot have a NULL value
- **UNIQUE** - Ensures that all values in a column are different
- **PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table
- **FOREIGN KEY** - Uniquely identifies a row/record in another table
- **CHECK** - Ensures that all values in a column satisfies a specific condition
- **DEFAULT** - Sets a default value for a column when no value is specified

Not null

- NOT NULL: This is a constraint will not accept NULL values into the column.
- You can apply NOT NULL on any number of columns

```
create table student1( sno int(3) NOT NULL,sname varchar(10),marks int(3));
```

- insert into student1 values(101,'arun',50); // CORRECT
- insert into student1 values(NULL,'KIRAN',70); // ERROR

UNIQUE

- UNIQUE: this constraint will not accept duplicate values.
- This constraint can apply on both **column and table level**.

- **Column Level**
- create table student1(sno int(3) Unique, sname varchar(10), marks int(3));
- **Table Level**
- create table student1(sno int(3), sname varchar(10), marks int(3), unique(sno));

UNIQUE…

- insert into student1 values(101,'arun',50);
- insert into student1 values(101,'kiram',60); // 101 not allowed

- insert into student1 values(null,'suresh',80);
- insert into student1 values(null,'raj',60);

- * Unique constraint column can accept multiple NULLS.

Primary Key

- PRIMARY KEY : Combination of Unique + Not Null
- primary key column will not allow duplicate values and also null values.
- primary key constraint can create both column level & table level.

```
create table student1(
sno int(3) primary key,
sname varchar(10),
marks int(3));
```

Primary Key...

- insert into student1 values(101,'arun',50); // right
- insert into student1 values(101,'suresh',60); // Invalid
- insert into student1 values(null,'suresh',60); // Invalid

* we can create primary key on combination of two columns called as composite primary key.

* Composite key can be applied only at table level.

Foreign key constraint

- A FOREIGN KEY is a key used to link two tables together.
- A FOREIGN KEY is a field (or collection of fields) in one table that refers to the PRIMARY KEY in another table.
- The table containing the foreign key is called the child table, and the table containing the candidate key is called the referenced or parent table.

Foreign key constraint…

- // parent Table

```
create table school(
sno int(3),
sname varchar(15),
marks int(3),
primary key(sno));
```

- insert into school values(101,'arun',90);
- insert into school values(102,'kiran',70);
- insert into school values(103,'amit',80);
- Select * from school;

- // child Table

```
create table library
(sno int(3),  FOREIGN KEY (sno) REFERENCES
school(sno),
book_name varchar(10));
```

- insert into library values(102,'java');
- insert into library values(108,'c'); //Not valid
- insert into library values(null,'dot net');

ON DELETE CASCADE

- ON DELETE CASCADE:
- Normally, we cannot delete rows from Parent table unless we delete corresponding row from child table.
- We can delete the rows from the parent table & corresponding child table row as well(at same time) by using ONDELETECASCADE option.

ON DELETE CASCADE...

- // parent Table

```
create table school(
sno int(3),
sname varchar(15),
marks int(3),
primary key(sno));
```

- insert into school values(101,'arun',90);
- insert into school values(102,'kiran',70);
- insert into school values(103,'amit',80);
- Select * from school;

- // child Table

```
create table library
(sno int(3),
book_name varchar(10),
FOREIGN KEY (sno) REFERENCES school(sno) ON DELETE CASCADE );
```

- insert into library values(101,'dot net');
- insert into library values(102,'java');

ON DELETE CASCADE...

- SELECT * FROM SCHOOL;
- DELETE FROM SCHOOL WHERE SNO=102; //valid
- SELECT * FROM LIBRARY;

* One row deleted from parent table and one from child table also deleted.

Foreign key constraint at table level

```
CREATE TABLE SCHOOL( SNO int(3),
                     SNAME VARCHAR(15),
                     MARKS NUMBER(3),
                     PRIMARY KEY(SNO));
)
```

- INSERT INTO SCHOOL VALUES(101,'ARUN',50);
- INSERT INTO SCHOOL VALUES(102,'AJAY',60);
- INSERT INTO SCHOOL VALUES(103,'KIRAN',80);

```
 CREATE TABLE LIBRARY
 (ROLLNO int(3),
 BOOK_NAME VARCHAR(10),
FOREIGN KEY (ROLLNO) REFERENCES SCHOOL(SNO) ON DELETE CASCADE);
```

- INSERT INTO LIBRARY VALUES(101,'DOT NET');
- INSERT INTO LIBRARY VALUES(102,'JAVA');

Check Constraint

- Check constraint is used for allowing to user to enter specific values into column.

```
create table student(
sno int(5),
sname varchar(15),
marks int(5) check(marks between 50 and 100));
```

- insert into student values(101,'amith',60); // valid
- insert into student values(101,'amith',45);// in valid
- insert into student values(101,'amith',105); //invalid

```
create table loc
( city varchar(15) check(city in('HYDERABD','CHENNAI','DELHI')),
  country varchar(15),
  pin int(8));
```

- insert into loc values('HYDERABD','INDIA',123456); // VALID
- insert into loc values('MUMBAI','INDIA',644566); // in valid
- insert into loc values('DELHI','INDIA',678445); //valid
- select * from loc;

Default Constraint

- The DEFAULT constraint is used to provide a default value for a column.
- The default value will be added to all new records IF no other value is specified.

```
CREATE TABLE Orders
(ID int(5),
OrderNumber int(5) ,
OrderDate datetime DEFAULT now());
```

- insert into Orders(ID,OrderNumber) values(101,2456);
- insert into Orders(ID,OrderNumber) values(102,2457);
- select * From Orders;

AUTO_INCREMENT

- Auto Increment is a function that operates on numeric data types. It automatically generates sequential numeric values every time that a record is inserted into a table for a field defined as auto increment.

```
create table student
(sno INT(5) primary key AUTO_INCREMENT,
sname varchar(15),
marks int(5));

ALTER TABLE student AUTO_INCREMENT = 100;


insert into student(sname,marks) values('X',60);
insert into student(sname,marks) values('Y',45);
insert into student(sname,marks) values('Z',105);

select * from student;
```

LIMIT

- Limit is used to display limited Rows from a table.
- SELECT * FROM EMPLOYEES LIMIT 10;
- SELECT * FROM EMPLOYEES LIMIT 5, 10;

Views

- A view is a virtual table based on the result-set of an SQL statement.
- A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.
- You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.
- Example:
- use hr;
- SELECT * FROM EMPLOYEES;
- Creating a view
  - CREATE VIEW EMPLOYEES_V1 AS SELECT EMPLOYEE_ID,FIRST_NAME,SALARY FROM EMPLOYEES;
- SELECT * FROM EMPLOYEES_V1;
- Dropping View
  - DROP VIEW EMPLOYEES_V1;

Index

- Indexes are used to retrieve data from the database very fast.
- The users cannot see the indexes, they are just used to speed up searches/queries.

- Creating Index:
  - CREATE INDEX idx_employees ON Employees (First_Name);
- Dropping Index:
  - drop index idx_employees on Employees;

TCL – Commit & Rollback

- SET autocommit = 1;
- drop table student;
- create table student(sid int(3),sname varchar(15));
- insert into student values(101,'abc’);
- insert into student values(102,'abc’);
- insert into student values(103,'abc’);
- Delete from student where sid=103; //Deletes record remporarly
- select * from student;
- Rollback; // Rollback record
- select * from student;
- Commit; //Commited delete
- Rollback;
- select * from student; // cannot get deleted record after commit

Exists

- The EXISTS operator is used to test for the existence of any record in a subquery.
- The EXISTS operator returns true if the subquery returns one or more records.

JDBC – Java Database Connectivity