DBMS
SQL-DOCUMENTS

Table of content:

Intro→
SQL-Structured Query Language
Mysql ,oracle,ms access etc are the RDBMS
RDMS–faciliate us to interact with database.

Query →RDBMS → DB
Note: order of execution→ Right to Left
Note: - - this is comment

MYSQL VS SQL →MYSQL is the software who uses SQL language to interact with database.

1. CREATE DATABASE db ;

2. SHOW DATABASES ;

3. USE db;

4. CREATE TABLE Worker(worker_id INT NOT NULL PRIMARY KEY AUTO_INCREMENT , first_name CHAR(25) , last_name CHAR(25),
salary INT(15),
joining_date DATETIME,
department CHAR(25)
);

5. INSERT
INTO Worker(worker_id,first_name,last_name,salary,joining_date,department)
VALUES(001,'chail','singh',10000,'12-02-22 09.00.00','HR'),
(002,'gaju','singh',20000,'13-03-22 08.00.00','ADMIN'),
(003,'kismat','singh',30000,'14-04-22 06.00.00','DEVELOPER'),
(004,'rajvi','singh',40000,'15-05-22 03.00.00','ACCOUNT'),
(005,'komal','singh',50000,'16-06-22 02.00.00','ADMIN'),
(006,'rinku','singh',60000,'17-07-22 01.00.00','HR');

6. SELECT * FROM Worker;

7. SELECT salary FROM Worker;

8. SELECT first_name,salary FROM Worker;

9. TRUNCATE TABLE Worker;

10. CREATE TABLE Bonus(
worker_ref_id INT,
bonus_amount INT(10),
bonus_date DATETIME,
FOREIGN KEY (worker_ref_id)
REFERENCES Worker(worker_id)
ON DELETE CASCADE);

11. INSERT INTO Bonus
(worker_ref_id,bonus_amount,bonus_date)
VALUES (001,1000,'13-02-22'),
(002,2000,'14-02-22'),
(003,3000,'15-02-22'),
(004,4000,'16-02-22'),
(005,5000,'17-02-22');

12. SELECT * FROM Bonus;

13. CREATE TABLE Title(
worker_ref_id INT,
worker_title CHAR(25),
affected_from DATETIME,
FOREIGN KEY (worker_ref_id)
REFERENCES Worker(worker_id)
ON DELETE CASCADE
);

14. INSERT INTO Title
(worker_ref_id,worker_title,affected_from)
VALUES (001,'Manager','12-02-22 09.00.00'),
(002,'Executive','15-02-22 09.00.00'),
(003,'Assistant','17-02-22 09.00.00'),
(004,'Manager','19-02-22 09.00.00'),
(005,'Executive','20-02-22 09.00.00');

15. SELECT * FROM Title;

16. SELECT * FROM Worker WHERE SALARY>20000;

17. SELECT * FROM Worker WHERE DEPARTMENT='ADMIN';

18. SELECT * FROM Worker WHERE SALARY BETWEEN 20000 AND 30000;
NOTE: 0 and 1 are inclusive

Note :Can we use SELECT keyword without using FROM clause?
yes,using dual table
Dual table are dummy table created by mysql ,help users to do certain obvious actions without reffering to user defined tables.
SELECT 55+22;
SELECT now();
SELECT ucase('chailsingh');
SELECT lcase('chailsingh');

19. **IN** → use to reduce **OR** condition
without IN → SELECT * FROM Worker WHERE DEPARTMENT='HR' **OR** DEPARTMENT='ADMIN' **OR** DEPARTMENT='ACCOUNT';

with IN (BETTER WAY) → SELECT * FROM Worker WHERE DEPARTMENT **IN**('HR','ADMIN','ACCOUNT');

20. SELECT * FROM Title WHERE WORKER_TITLE is NULL;

21. Pattern Searching/wild card

a) **_%_** = any no.of charactor
b) _ = only one character
%pa% -->possible ans=> abcpadb , pabcc , pabc
_pa_ -->possible ans=> apab , bpac

SELECT * FROM Worker WHERE first_name LIKE '%i%';
SELECT * FROM Worker WHERE last_name LIKE '_i_';
SELECT * FROM Worker WHERE first_name LIKE '%_i%';

22. **Sorting**

SELECT * FROM Worker ORDER BY salary;
SELECT * FROM Worker ORDER BY salary DESC;
SELECT * FROM Worker ORDER BY salary ASC;

23. Distinct

SELECT department FROM Worker ;
SELECT DISTINCT department FROM Worker ;

24. Data-Grouping / Data-Aggregation

Note:GROUP BY--> Aggregation function(**SUM , COUNT , AVG , MIN , MAX**)
Q.1) find no.of employee working in different department/how many customer are in india in facebook database
aggregation(count)

SELECT department FROM Worker GROUP BY department ; --> this will work same as distinct because we did not use aggregation function
SELECT department , **COUNT**(*) FROM Worker GROUP BY department ; --> this will returns department with count
SELECT department , **COUNT**(department) FROM Worker GROUP BY department; -->better way , use column name instead of *
Result
HR--2
ACCOUNT--2
ADMIN--3

Q.2) find average salary per department
SELECT department,**AVG(**salary**)** FROM Worker GROUP BY department**;**
SELECT department,**MIN(**salary**)** FROM Worker GROUP BY department**;**
SELECT department,**MAX(**salary**)** FROM Worker GROUP BY department**;**

SELECT department,**SUM(**salary**)** FROM Worker GROUP BY department**;**

NOTE:HAVING used to filter in **GROUP BY**
→ HAVING is similar as WHERE
Q.1) find department count having more than 2 worker?
SELECT department , COUNT(department) FROM Worker GROUP BY department HAVING COUNT(department) >1;

WHERE VS HAVING?
→ we can't use count with where whereas having does it.

CONSTRAINTS

1)**PRIMARY KEY**→ must NOT NULL , UNIQUE, only one PK per table
→should be INT
2)**FOREIGN KEY** → FK refers PK of other table
→ each relation can have many FK

CREATE DATABASE temp; → new database

USE temp;

```
CREATE TABLE Customer(
id INTEGER  PRIMARY KEY,
Cname VARCHAR(255),
Address VARCHAR(255),
Gender CHAR(2),
City VARCHAR(255),
Pincode INTEGER
);

INSERT INTO Customer VALUES
(121,'Ram','vihan vihar','M','delhi',1414),
(122,'Ram',anand vihar','M','delhi',1415),
(123,'Ram','shastri vihar','M','delhi',1416),
(124,'Ram','janta vihar','M','delhi',1417);

CREATE TABLE order_details(
order_id INT PRIMARY KEY,
delivery_date DATE,
cust_id INT,
FOREIGN KEY(cust_id)REFERENCES customer(id)
);

INSERT INTO order_details VALUES(1,'20-02-22',124);
```

INSERT INTO customer VALUES(123,'lakshaman','delhibagh','F','jaislamer',NULL);

3)**UNIQUE AND CHECK**

CREATE TABLE account(
id INT PRIMARY KEY,
name VARCHAR(255) **UNIQUE**,
balance INT,
CONSTRAINT acc_balance_chk **CHECK**(balance>1000)
);

INSERT INTO account(id,name,balance)
VALUES(1,'A',1000);
Note: this query gives an error because we use CHECK constraint and it failed to pass the condition

INSERT INTO account(id,name,balance)
VALUES(1,'A',2000);
Note:this will be executed because it passed the condition.

INSERT INTO account(id,name,balance)
VALUES(1,'A',4000);
Note:Gives an error because **duplicate** entry of 'A' (**UNIQUE** CONSTRAINT)

INSERT INTO account(id,name,balance)
VALUES(1,'B',4000);
Note:this will be executed because it passed the condition.

4)**DEFAULT CONSTRAINTS**

CREATE TABLE account_with_default_balance(
id INT PRIMARY KEY,
name VARCHAR(255) UNIQUE,
balance INT NOT NULL DEFAULT 0
);

INSERT INTO account_with_default_balance(id,name)
VALUES(2,'B');
Note: value of balance will be default set by 0

ALTER OPERATION

ADD-->add new column
eg. ALTER TABLE account ADD interest FLOAT NOT NULL DEFAULT 0;

MODIFY-->change datatype of an attribute
eg.ALTER TABLE account MODIFY interest DOUBLE NOT NULL DEFAULT 0;

CHANGE COLUMN-->rename column name
eg.ALTER TABLE account CHANGE COLUMN interest saving_interest FLOAT NOT NULL DEFAULT 0;

DROP COLUMN-->drop a column completely
eg.ALTER TABLE account DROP COLUMN saving_interest;

RENAME-->rename table name itself
eg.ALTER TABLE account RENAME TO account_details;

MODIFY OPERATION

INSERT
→ *single entry*

INSERT INTO customer(id,cname,Address,Gender,City,Pincode)VALUES
(120,'Ram','vihan vihar','M','delhi',1414);

INSERT INTO customer VALUES
(120,'Ram','vihan vihar','M','delhi',1414);

INSERT INTO customer VALUES
(120,'Ram');

→*multiple entry*

INSERT INTO customer VALUES
(120,'Ram');

INSERT INTO customer(id,cname,Address,Gender,City,Pincode) VALUES
(141,'Ram','vihan vihar','M','delhi',1414),
(144,'Ram','vihan vihar'CREATE TABLE customer(
id integer PRIMARY KEY,
cname VARCHAR(255),
Address VARCHAR(255),

```
Gender CHAR(2),
City VARCHAR(255),
Pincode integer
);

INSERT INTO customer VALUES
(121,'Ram','vihan vihar','M','delhi',1414),
(123,'Ram','vihan vihar','M','delhi',1414),
(124,'Ram','vihan vihar','M','delhi',1414),
(125,'Ram','vihan vihar','M','delhi',1414);

CREATE TABLE order_details(
order_id INT PRIMARY KEY,
delivery_date DATE,
cust_id INT,
FOREIGN KEY(cust_id)REFERENCES customer(id)
);

INSERT INTO order_details
VALUES(1,'20-02-22',124);,'M','delhi',1414),
(155,'Ram','vihan vihar','M','delhi',1414),
(166,'Ram','vihan vihar','M','delhi',1414);

INSERT INTO customer VALUES
(141,'Ram','vihan vihar','M','delhi',1414),
(144,'Ram','vihan vihar','M','delhi',1414),
(155,'Ram','vihan vihar','M','delhi',1414),
(166,'Ram','vihan vihar','M','delhi',1414);
```

UPDATE

→ single entry

```
UPDATE customer SET Address='mumbai',Gender='F' WHERE id=141;
```

→ update all rows

```
UPDATE customer SET Address='mumbai';
```

Note: This command should replace all address with mumbai but this may be a virus who is trying to modify the all rows so for security purpose **mysql itself enable the SQL_SAFE_UPDATES** so this command will not run,to run this command we need to set the **SQL_SAFE_UPDATE to ZERO**

SET SQL_SAFE_UPDATES=0;
UPDATE customer SET Address='mumbai';
UPDATE customer SET Pincode=Pincode+1;
Note: now these command will runs.

DELETE

→ single entry delete
DELETE FROM customer WHERE id=121;

→ whole rows in table delete
DELETE FROM customer;
Note: this command should delete all customer but this may be a virus who is trying to delete whole table so for security purpose **mysql itself enable the SQL_SAFE_UPDATES** so this command will not run,to run this command we need to set the **SQL_SAFE_UPDATE to ZERO**

SET SQL_SAFE_UPDATES=0;
DELETE FROM customer;
Note: now these command will runs.

REFERENTIAL CONSTRAINTS

1) INSERT CONSTRAINT
You can't insert value in child untill the value is not lying in the parent table.

2) DELETE CONSTRAINT
You can't delete value from parent if the corrosponding value is present in child

To overcome Delete constraint→
method 1)*ON DELETE CASCADE*-->value from parent and child both delete
method 2)*ON DELETE SET NULL*-->value from parent delete and child fk set null.

ON DELETE CASCADE

CREATE TABLE customer(
id integer PRIMARY KEY,
cname VARCHAR(255),
Address VARCHAR(255),

```
Gender CHAR(2),
City VARCHAR(255),
Pincode integer
);

INSERT INTO customer VALUES
(121,'Ram','vihan vihar','M','delhi',1414),
(123,'Ram','vihan vihar','M','delhi',1414),
(124,'Ram','vihan vihar','M','delhi',1414),
(125,'Ram','vihan vihar','M','delhi',1414);

CREATE TABLE order_details(
order_id INT PRIMARY KEY,
delivery_date DATE,
cust_id INT,
FOREIGN KEY(cust_id)REFERENCES customer(id) ON DELETE CASCADE
);

INSERT INTO order_details
VALUES(1,'20-02-22',121),
(1,'20-02-22',121);

DELETE  FROM customer WHERE id=121;
```

ON DELETE SET NULL

```
CREATE TABLE customer(
id integer PRIMARY KEY,
cname VARCHAR(255),
Address VARCHAR(255),
Gender CHAR(2),
City VARCHAR(255),
Pincode integer
);

INSERT INTO customer VALUES
(121,'Ram','vihan vihar','M','delhi',1414),
(123,'Ram','vihan vihar','M','delhi',1414),
(124,'Ram','vihan vihar','M','delhi',1414),
(125,'Ram','vihan vihar','M','delhi',1414);

CREATE TABLE order_details(
order_id INT PRIMARY KEY,
delivery_date DATE,
```

```
cust_id INT,
FOREIGN KEY(cust_id)REFERENCES customer(id) ON DELETE SET NULL
);

INSERT INTO order_details
VALUES(1,'20-02-22',121),
(1,'20-02-22',121);

DELETE  FROM customer WHERE id=121;
```

REPLACE VS INSERT

REPLACE → *data already present then replace ,if data not present then this command act as an* **INSERT**

Behave as REPLACE →

```
REPLACE INTO customer(id,City)
VALUES(120,'JAPAN');
```

behave as an **INSERT** →

```
REPLACE INTO customer(id,City)
VALUES(12000,'orrisa');
```

other way--> REPLACE INTO customer SET id=12,Name='mac',City='banglore';

```
other way--> REPLACE INTO customer(id,cname,City)
                SELECT cname,City
                FROM customer WHERE id=500;
```

REPLACE VS UPDATE → if row is not present then replace will add new row while update will do nothing.

.

JOINS

we establish relation with the help of **FK**,now to fetch the relational data we use **JOINS**

To apply JOINS there should be a common attribute.

1) INNER JOIN → matching data from both table and shows into resultant table
SELECT c.*,o.* FROM customer as c INNER JOIN order_details as o ON c.id=o.cust_id;

2) OUTER JOIN

A)LEFT JOIN → return resultant table that all the data from **LEFT** table and **matched** data from **RIGHT** table(unmathced corrosponding data in *RIGHT* table set as *NULL*)
SELECT c.*,o.* FROM customer as c **LEFT JOIN** order_details as o **ON** c.id=o.cust_id;

B) RIGHT JOIN → return resultant table that all the data from **RIGHT** table and **matched** data from **LEFT** table(unmathced corrosponding data in *LEFT* table set as *NULL*)
SELECT c.*,o.* FROM customer as c RIGHT JOIN order_details as o ON c.id=o.cust_id;

C) FULL JOIN → ***UNION OF LEFT AND RIGHT JOIN***
Note:there is no full join named keyword available in mysql so we need to emulate→*LEFT JOIN U RIGHT JOIN*
SELECT * FROM LEFT TABLE as l LEFT JOIN RIGH TTABLE as r ON l.key=r.key
UNION
SELECT * FROM LEFT TABLE as l RIGHT JOIN RIGHT TABLE as r ON l.key=r.key

D) CROSS JOIN → ***CARTESIAN PRODUCT OF RIGHT AND LEFT TABLE***
let 5 row in left and 10 row in right table then resultant table gives all possible matches → 5*10=50 rows

E) SELF JOIN
Note:there is no full join named keyword available in mysql so we need to emulate → *INNER JOIN ,ALIAS 'AS'*
Eg.
SELECT e1.id,e2.id,e2.name FROM employee as e1 INNER JOIN employee as e2 ON e1.id=e2.id;

Practical on workbench of all JOINS method
left table → employee,right table → project

INNER JOIN →
Q.1) enlist all the employee id's,names along with the project allocated to them
→*selected column*

SELECT e.id,e.fname,e.lname,p.id,p.name from employee as e INNER JOIN project as p ON e.id=p.empID;

→All column
SELECT * from employee as e INNER JOIN project as p ON e.id=p.empID;

Q.2) fetch all the employees'id and their contact details who have been working from jaipur with their client name working in hydrabad.

SELECT e.id,e.emailID,e.phooneNO,c.first_name,c.last_name from Employee as e INNER JOIN Client as c ON e.id=c.empID WHERE c.City='jaipur' AND c.City='Hydrabad';

LEFT JOIN →
Q.1)fetch out each project allocated to each employee
SELECT * FROM Employee as e LEFT JOIN Project as p ON e.id=p.empID;

RIGHT JOIN →
Q.1)list out all the project along with employee name and their respective allocated emailid
SELECT p.id,p.name,e.fname,e.lname,e.emailID FROM Employee as e RIGHT JOIN Project as p ON e.id=p.emailID;

CROSS JOIN →
Q.1)list out all the combination possible for employee name and project that can exist
SELECT e.fname,e.lname,p.id,p.name FROM Employee as e CROSS JOIN Project as p;

Note: can we use INNER JOIN without using INNER JOIN KEYWORD?
YES,we can
SELECT * FROM LEFT TABLE , RIGHT TABLE WHERE LEFT TABLE.id=RIGHT TABLE.id

SELECT e.id,e.fname,e.lname,p.id,p.name FROM employee as e,project as p WHERE e.id=P.empID;

SET OPERATION

SET → group of unique items.
TYPES → UNION , INTERSECTION , MINUS

1)UNION → *SELECT * FROM table1 UNION SELECT * FROM table2;*

Note:full join vs union → in full join no. of column increases while in union no.of rows increases.

2)INTERSECTION → SELECT * FROM table1 INTERSECT SELECT * FROM table2;
Note:INTERSECT keyword does not exist in mysql so we need to emulate using inner join.
SELECT DISTINCT id from T1 INNER JOIN T2 using(id);
SELECT DISTINCT id from T1 INNER JOIN T2 t1.id=t2.id;

3)MINUS →T1-T2
Note:MINUS keyword does not exist in mysql so we need to emulate using inner join.
SELECT id FROM T1 LEFT JOIN T2 using(id) WHERE T2.id is NULL;

PRACTICAL ON WORKBENCH OF SET OPERATION

DEPARTMENT 1 AND DEPARTMENT 2 → empid,name,role

UNION

Q1)list out all the employees in the company
*SELECT * FROM dep1 UNION SELECT * FROM dep2;*

Q2)list out all the employees in all department who work as a sales man
*SELECT * FROM dep1 WHERE role='salesman' UNION SELECT * FROM dep2 WHERE role='salesman';*

INTERSECTION →
Q1)list out all the employees who work in both department.
*SELECT * FROM dep1 INNER JOIN dep2 USING(empid);*
SELECT dep1. FROM dep1 INNER JOIN dep2 USING(empid);*

MINUS →
Q1)list out all employee who work in dep1 but not in dep2
SELECT dep1. FROM dep1 LEFT JOIN dep2 USING(empid) WHERE dep2.empid=NULL;*

SUB QUERIES -->ALTERNATIVE METHOD OF JOIN-->GENERALLY OUTER QUERY DEPEND ON INNER QUERY

eg. *SELECT * FROM TABLE WHERE id IN(SELECT id FROM TABLE WHERE name='chail');*

Types

1)WHERE CLAUSE ON SAME TABLE → *employee with age>30*

*SELECT * FROM employee WHERE age IN(SELECT age FROM employee WHERE age>30);*

2)WHERE CLAUSE ON DIFFERENT TABLE → *emp details working in more than 1 project*

*SELECT * FROM employee WHERE id IN(SELECT empID FROM project GROUP BY empID HAVING COUNT(empID)>1);*

3)SINGLE VALUED QUERY → *emp details having age >AVG(age)*

*SELECT * FROM employee WHERE age >(SELECT AVG(age) FROM employee);*

4)FROM CLAUSE → derived table-

→ select max aged person whose first name has contains 'a'

*SELECT MAX(age) FROM (SELECT * FROM employee WHERE fname LIKE '%a%') AS temp;*

5)CORELATED SUBQUERY → *inner query also refers the outer query-->find third oldest employee*

```
SELECT * FROM employee e1
    WHERE 3=(
            SELECT COUNT(e2.age)
            FROM employee e2
            WHERE e2.age>=e1.age
        );
```

SQL VIEWS → template virtual table

let customer has 10 column but we defined only 5 column in view so when we see table using view it will show only 5 col.

```
SELECT * FROM employee;
CREATE VIEW custom_view AS SELECT fname,age FROM employee;
SELECT * FROM custom_view;
ALTER VIEW custom_view AS SELECT fname,lname,age FROM employee;
DROP VIEW IF EXISTS custom_view;
```

What is an Entity Relationship Diagram (ERD)?

An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves. ER diagrams also are often used in conjunction with data flow diagrams (DFDs), which map out the flow of information for processes or systems.

STEPS TO GENERATE ERD→

1) go to mysql workbench , click on database nav button then click on AFE_UPDATES=0 engineering
2) select instance then click on next twice
3)select targeted database and click next twice
4)select on execute then next and close

chail