1)Implementation of DDL commands(Create table , Alter table , Drop Table) of SQL on Student Database

Create:

create table student(roll\_no int primary key, name varchar(50), address varchar(50) , class varchar(50));  
  
  
Alter:

ALTER TABLE student ADD pn\_number varchar(10);  >> add column

alter table student drop column pn\_number; >> Delete column

alter table student rename column roll\_no to rno; >> rename / modify

Drop:

drop table student;

2)Implementation of DML(Insert, Update ,Delete) commands of SQL on Employee Database

Insert:

insert into student values(1,'Bhagwatilal','Navapur','FYMCA');

insert into student values(2,'Harshal', ' pune ', 'FYMCA');

insert into student values(3,'Abhishek', 'nashik', 'FYMCA');

insert into student values(4,'Prafful', 'Mumbai', 'FYMCA');

insert into student values(5,'Akash', 'satara', 'FYMCA');

Display:  
select \* from student;

Update:  
  
update student set address = 'pune' where rno = 3;

select \* from student;

Delete:

delete from student  where roll\_no = 5;

select \* from student;

3)Implementation of different types of function (Number function, Aggregate Function ,Character Function ,Conversion Function,& Date Function ) on student Database.

create table student1(roll\_no int primary key, name varchar(50), address varchar(50) , class varchar(50),marks int);

insert into student values(1,'Bhagwat','Navapur','FYMCA',80);

insert into student1 values(2,'Harshal', ' pune ', 'FYMCA',82);

insert into student1 values(3,'Abhishek', 'nashik', 'FYMCA',76);

insert into student1 values(4,'Prafful', 'Mumbai', 'FYMCA',83);

insert into student1 values(5,'Akash', 'satara', 'FYMCA',86);  
  
select \* from student1;  
  
  
Aggregate functions:  
  
Count: select count(roll\_no) from student1;   
  
Sum: select sum(marks) from student1;  
  
AVG: select avg(marks) from student;  
  
  
  
Max: select max(marks) from student;  
  
Min: select min(marks) from student;

**Number Functions:**

Power: select power(4,2) "power" from dual;  
  
Absolute: select abs(-2344) ) "abs" from dual;;  
  
Floor: select floor(3.456) “Floor” from dual;  
  
Ciel: select ciel(3.456) “Ciel” from dual;

Character Functions:

Lower: select lower(name) from student1;  
  
Upper: select upper(name) from student1;  
  
Concat: select concat(name, '..Hi') from student1;  
  
Length: select length(name) from student1 where roll\_no=1;  
  
  
**Conversion Functions:**

SELECT TO\_CHAR(123.45) AS converted\_string FROM DUAL;

SELECT TO\_NUMBER('123.45') AS converted\_number FROM DUAL;

SELECT TO\_DATE('2023-07-20', 'YYYY-MM-DD') AS converted\_date FROM DUAL;

**Date Functions:**

select sysdate as current\_date from dual;

select extract(year from sysdate) as year from dual;

4)Implementation of different types of operators(Arithmetic Operators ,Logical Operators , Comparison Operator ,Special Operator, Set Operation) in SQL by using suitable examples.

CREATE TABLE department1(dept\_id int primary Key,dname varchar(20),location varchar(25));

insert into department1 VALUES(1,'developement','nashik');

insert into department1 VALUES(2,'Sales','pune');

insert into department1 VALUES(3,'QA','mumbai');

insert into department1 VALUES(4,'Hr','satara');

SELECT \* from department;

CREATE TABLE employee1(eId int primary key, name varchar(20), designation varchar(30), salary int);

insert into employee1 values(101,'Bhagwatilal', 'developer',10000);

INSERT INTO employee1 VALUES (102, 'Prafful', 'inside sales',23000);

INSERT INTO employee1 VALUES (103, 'Abhishek', 'QA engineer',7000);

INSERT INTO employee1 VALUES (104, 'Harshal', 'Jr.Hr',15000);

SELECT \* FROM employee1;  
  
**Arithmatic operators:**  
  
+ operator: select salary+1000 from employee1;  
  
- operator: select salary-1000 from employee1;   
  
/ operator: select salary/2 from employee1;  
  
\* operator: select salary\*3 from employee1;

Set operators:

For performing set operations we need a similar table;

CREATE TABLE employee2(eId int primary key, name varchar(20), designation varchar(30), salary int);  
  
insert into employee2 values(101,'Parth', 'CS Manager',60000);

INSERT INTO employee2 VALUES (102, 'Aditya', 'DB Manager',53000);

INSERT INTO employee2 VALUES (103, 'Mandar', 'Tester',50000);

INSERT INTO employee2 VALUES (104, 'Priyal', 'Hr',45000);

Union:

select \* from employee1

union

select \* from employee2;

Intersect:

select \* from employee1

intersect

select \* from employee2;

Minus:

select \* from employee1

minus

select \* from employee2;  
  
Comparison operators:  
  
= operator: SELECT \* FROM employee1 WHERE name = 'Prafful';  
  
> operator: SELECT \* FROM employee1 WHERE salary > 10000;

>= operator: SELECT \* FROM employee1 WHERE salary >= 10000;

< operator: SELECT \* FROM employee1 WHERE salary < 10000;

<= operator: SELECT \* FROM employee1 WHERE salary <= 10000;  
  
!= operator: SELECT \* FROM employee WHERE salary != 10000;  
  
  
  
  
**Logical operators:**  
  
SELECT \* FROM employee1 WHERE name = 'Abhishek' AND designation = 'QA engineer';  
  
SELECT \* FROM employee1 WHERE name = 'Abhishek' or designation = 'QA engineer';  
  
SELECT \* FROM employee2 WHERE designation NOT in ('Hr');  
  
**Special operators:**  
  
SELECT \* FROM employee1 WHERE designation IN ('QA engineer','developer');  
  
SELECT \* FROM employee1 WHERE salary BETWEEN 10000 AND 25000;  
  
SELECT \* FROM employee2 WHERE name LIKE 'P%';

5)Implementation of Different types of Joins(Inner Join ,Outer Join Natural Join ) in SQL on Employee Database.

**create table student2(rolln int primary key , name varchar(30), address varchar(50) , class varchar(255),marks int);**

insert into student values(1,'Bhagwatilal', 'Navapur','FYMCA',79);

insert into student values(2,'Harshal', 'Navapur', 'FYMCA',80);

insert into student values(3,'Abhishek', 'nashik', 'FYMCA',67);

insert into student values(4,'Prafful', 'nashik', 'FYMCA',77);

insert into student values(5,'Akash', 'nashik', 'FYMCA',70);

select \* from student2;

**create table student3(rolln int primary key,name varchar(20),subjects varchar(50));**

**insert into student3 values(1,'Raghav','Ethics');**

**insert into student3 values(2,'Sanjana','WE');**

**insert into student3 values(3,'Jack','physiology');**

**insert into student3 values(4,'Parth','CCNA');**

**insert into student3 values(5,'Kartik','kotlin');**

**insert into student3 values(6,'gogi','masti');**

select \* from student3;

**Inner join :**

select address,class,marks,subjects from student3 inner join student2 on student3.rolln=student2.rolln;

**Outer join:**

select \* from student3 right outer join student2 on student3.rolln=student2.rolln;

**Natural join :**

select \* from student2 natural join student3;

6) Study and Implementation of Clauses (Group By & having clause, Order by clause, Indexing) in SQL on Suitable Database.

select \* from student1;

**select roll\_no from student1 group by roll\_no having min(marks)>70;**

**order by**

**select \* from student1 order by roll\_no asc;**

**select \* from student1 order by roll\_no desc;**

**create index binod on student1(name);  
binod**

7) Study & Implementation of Sub queries and Views on Employee Database .

select \*

from student1

where roll\_no >(select rolln from student3 where name='Jack');

create view education as

select marks,name,roll\_no

from student1

where marks>70;

select marks,name

from education

order by roll\_no;

8. study and implementation of SQL Cursor

Implicit:

DECLARE

total\_rows number(2);

BEGIN

UPDATE employee1

SET salary = salary + 500;

IF sql%notfound THEN

dbms\_output.put\_line('no employee selected');

ELSIF sql%found THEN

total\_rows := sql%rowcount;

dbms\_output.put\_line( total\_rows || ' employee selected ');

END IF;

END;

**Explicit:**

DECLARE

eid employee1.eid%type;

name employee1.name%type;

salary employee1.salary%type;

CURSOR cemp is

SELECT eid, name, salary FROM employee1;

BEGIN

OPEN cemp;

LOOP

FETCH cemp into eid, name, salary;

EXIT WHEN cemp%notfound;

dbms\_output.put\_line(eid || ' ' ||name || ' ' || salary);

END LOOP;

CLOSE cemp;

END;

9. study and implementation of SQL function and procedure

**Procedure:**

create or replace procedure "INSERTUSER"

(rno in number,

name in varchar2,

address in varchar2,

class in varchar2)

is

begin

insert into student values(rno,name,address,class);

end;

BEGIN

insertuser(102,'Rahul','surat','kk');

dbms\_output.put\_line('record inserted successfully');

END;

**Function:**

CREATE OR REPLACE FUNCTION GTS(pid NUMBER) RETURN NUMBER AS

total\_salary NUMBER := 0;

BEGIN

SELECT SUM(salary) INTO total\_salary FROM employee1 WHERE eid = pid;

RETURN total\_salary;

END;

SELECT eid, GTS(103) AS total\_salary

FROM employee1;

12).Study of Open Source NOSQL Database: MongoDB (Installation, Basic CRUD operations, Execution)

ANS:-

1. Database Commands

View all databases

>show dbs

2.Create a new or switch databases

>use student

3.View current Database

>db

4,=.Delete Database

>db.dropDatabase()

5.Collection Commands

>Show Collections

6.Create a collection named 'student'

>db.createCollection('student')

7.Drop a collection named 'stud'

db.stud.drop()

Show all Rows in a Collection

db.student.find()

Show all Rows in a Collection (Prettified)

db.comments.find().pretty()

Find the first row matching the object

> db.student.find({name:"Jay"}).pretty()

Insert One Row

> db.student.insertOne({name:"Jay",class:"Symca",age:22})

Insert many Rows

db.student.insertOne([{name:"Mohit",class:"Symca",age:23},{name:"Sumit",class:"Symca",age:26}])

Limit the number of rows in output

db.comments.find().limit(2)

Count the number of rows in the output

db.comments.find().count()

Update a row

> db.student.update({"name":"Abc"},{$set:{"class":32}})

Mongodb Increment Operator

db.student.update({name: 'Aarti'},

{$inc:{

age: 1

}})

Less than/Greater than/ Less than or Eq/Greater than or Eq

db.student.find({age: {$lt: 90}})

db.student.find({age: {$lte: 90}})

db.student.find({age: {$gt: 90}})

db.student.find({age: {$gte: 90}})

**Practical 3**

**Queries in mongoDB**

db.student.insert({ "stud\_name": "Maya", "class": "TYBBA\_CA", "percentage":80})

db.student.insert({ "stud\_name": "Pooja", "class": "TYBBA\_CA", "percentage":50}) db.student.insert({ "stud\_name": "Ram", "class": "SYBBA\_CA", "percentage":90}) db.student.insert({"stud\_name": "Vedant", "class": "SYBBA\_CA", "percentage":85})

db.student.insert({ "stud\_name": "Supriya", "class": "TYBBA\_CA", "percentage":56})

db.student.insert({"stud\_name": "Smita", "class": "TYBBA\_CA", "percentage":90, "contact":9876543210})

Display the students from TYBBA\_CA class.

>db.student.find({"class":{$eq:"TYBBA\_CA"}})

Display Name of students not having TYBBA\_CA class.

> db.student.find({"class":{$ne:"TYBBA\_CA"}},{"\_id":0, "stud\_name":1})

Display the students having percentage greater than 80

>db.student.find({"percentage":{"$gt":80}})

Display the students having percentage either 80 or 90.

>db.student.find({"percentage":{$in:[80,90]}})

Display students having percentage less than 80 and class TYBBA\_CA

>db.student.find({$and:[{"percentage":{"$lt":80}},{"class":"TYBBA\_CA"}]})

Display student details whose name start with “Ra” pattern.

>db.student.find({"stud\_name": /^Ra/},{“\_id”:0})

Display student details whose name end with “ya” pattern.

>db.student.find({"stud\_name": /ya$/},{“\_id”:0})

Counting the total numbers of documents

>db.student.count()

Update the percentage of the student whose name is “Maya”

>db.student.updateOne({stud\_name: "Pooja"}, {$set:{percentage:60}})

delete the record of the student whose name is “Pooja”

> db.student.deleteOne({name:"Mohit"})

{ "acknowledged" : true, "deletedCount" : 1 }

Update

> db.student.updateOne({"name":"Abhishek"},{$se{name:"Abhishek",class:"Fymca",age:23}},{upsert:true})