1. Create tables with all constraints

2. create views on any two tables using conditions

3. create an index called EmployeeInd for the department table. Entries should be in ascending order by department id and then by employee id within each department.

4. create sequence on Employee id

```
create database chit1;
 use chit1;
create table employee(
  emp_id int primary key not null,
  emp_name varchar(40),
  emp_age int not null);

 insert into employee values(
  1001,"brim",30);

 Insert into employee value(1002,"fade",24);

 Insert into employee value(1003,"yoru",26);

 Insert into employee value(1004,"sova",41);

 Create view view_1 as
  Select emp_id,emp_name
  From    employee
  whereemp_id=1001&emp_name='brim';

 create table department(
  dept_id  int primary key not null,
  dept_name varchar(20) not null,
  dept_floor int not null);
 insert into department value(3101,"Comp",4);

insert into department value(3202,"IT",3);

insert into department value(3303,"ETC",2);

insert into department value(3304,"FE",1);;
insert into department value(3305,"FE",1);

 createviewview_2as
  select dept_id,dept_name
  from department
  where dept_id=3202 & dept_name='IT';

 select*from employee;

 select*from view_1;

 select*from department;

 select*fromview_2;

 create index EmployeeInd on department(dept_id);

 select*from employee
  order by emp_id ASC;

 select*from department
  order by dept_id ASC;


 alter table employee modify emp_id int not null auto_increment;
```

```
select*fromemployee;

insert into employee(emp_name,emp_age)values('johnny',69);

insert into employee(emp_name,emp_age)values('Malkova',699);

select*from employee;
```

Write Exception handling code with the Use of Control structure.

Consider Tables:

1. Borrower(Roll_no, Name, Date of Issue, Name of Book, Status)

2. Fine(Roll_no, Date, Amt)

• Accept Roll_no and Name of Book from user.

• Check the number of days (from date of issue).

• If days are between 15 to 30 then fine amount will be Rs 5per day.

• If no. of days>30, per day fine will be Rs 50 per day and for days less than 30, Rs. 5 per Day.

- **After submitting the book, status will change from I to R.**

• If condition of fine is true, then details will be stored into fine table.

• Also handles the exception by named exception handler or user define exception handler.

```
create database sits;

use sits;

CREATE TABLE borrower(roll_no int , name VARCHAR(25), dateofissue
DATE,name_of_book VARCHAR(25), status VARCHAR(20));

CREATE TABLE fine(roll_no int,date_of_return DATE,amt int);

INSERT INTO borrower VALUES(45,'ASHUTOSH',2020/07/19,'HARRY
POTTER','PENDING');

INSERT INTO borrower VALUES(46,'TOSH','2003/1/1','HARRY POTTER','PENDING');

INSERT INTO borrower VALUES(47,'SH','2004/1/1','POTTER','PENDING');

Delimiter //
```

```
create procedure B(roll_new int,book_name varchar(20))

begin

declare X integer;

declare continue handler for not found

begin

select 'NOT FOUND';

end;

select datediff(curdate(),dateofissue) into X from borrower where roll_no=roll_new;

if(X>15&&X<30)

then

insert into fine values(roll_new,curdate(),(X*5));

end if;

if (X>30)

then

insert into fine values(roll_new,curdate(),(X*50));

end if;

update borrower set status='returned' where roll_no=roll_new;

end;

call B(45,'HARRY POTTER');//

select * from fine;//
```

Write a Stored Procedure namely proc_Grade for the categorization of student. If marks scored by students in examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored are between 989 and900 category is first class, if marks 899 and 825 category is Higher Second Class Write a PL/SQL block for using procedure created with above requirement. Stud_Marks(name, total_marks)

```
show databases;

use sits;

 create database marks;

  use marks;

create table studmarks(name varchar(20),total_marks integer);

create table result(roll_no integer,name varchar(20),class varchar(25));


DELIMITER $
```

```
create procedure proc_grade(in rollno tinyint,in name varchar(15),in marks int)
begin
declare class varchar(25);
 if marks>=990 and marks<=1500 then set class="Distinction";
 elseif marks<=989 and marks>=900 then set class="First Class";
 elseif marks<=899 and marks>=825 then set class="Second Class";
 elseif marks<=824 and marks>=700 then set class="Pass";
 else
 set class="Fail";
 end if;
 insert into studmarks values(name,marks);
 insert into result values(rollno,name,class);
 end$


     DELIMITER $
    call proc_grade(1,"Aryan",850);
  DELIMITER $
    call proc_grade(2,"Peter",1000);
  DELIMITER $
    call proc_grade(3,"Smith",834);
  DELIMITER $
    call proc_grade(4,"Carol",750);
  DELIMITER $
call proc_grade(5,"Bob",950);
  DELIMITER $
    select * from result;
  DELIMITER $
    select * from studmarks;
  DELIMITER $



create function tot_stud(classname varchar(25))
```

```
  returns int
  begin
  declare total int(20);
  select distinct count(*) into total from result where class=classname;
  return total;
  end $
 select tot_stud("Second Class");
  DELIMITER $
 select tot_stud("Pass");
  DELIMITER $
 select tot_stud("First Class");
  DELIMITER $
```

Write a Function namely proc_Grade for the categorization of student. If marks scored by students in examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored are between 989 and900 category is first class, if marks899and 825category is Higher Second Class. Write a PL/SQLblock to use procedure created with above requirement.Stud_Marks(name, total_marks) Result(Roll,Name, Class)

```
create database sits;
use sits
create table marks(roll_no int,name varchar(20),total_marks varchar(20));
create table result(roll_no int,name varchar(20),class varchar(20));
insert into marks values('1','Abhi','1400');
 insert into marks values('2','piyush','980');
 insert into marks values('3','hitesh','880');
 insert into marks values('4','ashley','820');
 insert into marks values('5','partik','740');
 insert into marks values('6','patil','640');
delimiter //
create procedure proc_result(in marks int,out class char(20))
     begin
    if(marks<1500&&marks>990)
    then
```

```
        set class="Distinction";
        end if;
        if(marks<989&&marks>890)
        then
        set class='First Class';
        end if;
        if(marks<889&&marks>825)
        then
        set class='Higher Second Class';
        end if;
        if(marks<824&&marks>750)
        then
        set class='Second Class';
        end if;
        if(marks<749&&marks>650)
        then
        set class='Passed';
        end if;
        if(marks<649)
        then
         set class='Fail';
        end if;
        end;
        //
SET GLOBAL log_bin_trust_function_creators = 1;
//
create function final_result3(R1 int)
      returns int
      begin
      declare fmarks integer;
      declare grade varchar(20);
      declare stud_name varchar(20);
```

```
    select marks.total_marks,marks.name into fmarks,stud_name from marks where
marks.roll_no=R1;

    call proc_result(fmarks,@grade);

    insert into result values(R1,stud_name,@grade);

    return R1;

    end;

    //

select final_result3(2); //
```

6	Write a database trigger on the Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in the Library_Audit table.

```
create database lib;

use lib;

create table library(accession integer primary key, title varchar(60),author
varchar(80),publisher varchar(50));

create table library_audit(accession integer, title varchar(60),author varchar(80),publisher
varchar(50),operation varchar(50),constraint ad foreign key(accession)references
library(accession));

describe library;

describe library_audit;

insert into library values(1,'Let us C','kanitakar','Tata');

insert into library values(2,'C programming','balagi','technical');

insert into library values(3,'Distributed Systems','AAA','TecMax');

create trigger update_lib before update on library for each row insert into
library_audit(accession,title,author,publisher,operation)values(old.accession,old.title,old.auth
or,old.publisher,'update');

create trigger delete_lib before delete on library for each row insert into
library_audit(accession,title,author,publisher,operation)values(old.accession,old.title,old.auth
or,old.publisher,'delete');

SET foreign_key_checks=0;

select * from library;

select * from library_audit;

update library set publisher='tata' where accession=2;

delete from library where accession=1;
```

select * from library_audit;

7	Write a database trigger on the student table.Perform all types of triggers on the created table.

create database lib;

use lib;

create table library(accession integer primary key, title varchar(60),author varchar(80),publisher varchar(50));

describe library;

describe library_audit;

insert into library values(1,'Let us C','kanitakar','Tata');

insert into library values(2,'C programming','balagi','technical');

insert into library values(3,'Distributed Systems','AAA','TecMax');

create trigger update_lib after update on library for each row insert into library_audit(accession,title,author,publisher,operation)values(old.accession,old.title,old.author,old.publisher,'update');

select * from library_audit;

select * from library;

update library set publisher='tata' where accession=2;

select * from library_audit;

create trigger delete_lib after delete on library for each row insert into library_audit(accession,title,author,publisher,operation)values(old.accession,old.title,old.author,old.publisher,'delete');

SET foreign_key_checks=0;

select * from library_audit;

create trigger insert_lib after insert on library for each row insert into library_audit(accession,title,author,publisher,operation)values(new.accession,new.title,new.author,new.publisher,'insert');

insert into library values(4,'Java Programming','BBB','technical');

select * from library_audit;

Create table employee.Write database cursor with implicit and explicit cursor types.

```
create database Employee;
use Employee;
create table emp_dat (empid int,name
varchar(20),address varchar(20));
create table emp_update (empid int,name
varchar(20),addressvarchar(20));
insert into emp_dat values(1011,'Sunny','Canada');
insert into emp_dat values(6969,'Lana','China');
insert into emp_dat values(6666,'Alexa','USA');
insert into emp_dat values(9999,'CinKy','Russia');
insert into emp_dat values(6699,'Dekhmat','INDIA');
delimiter //
create procedure p3(in r1 int) begin declare r2 int;
  declare exit_loop boolean;
  declare c1 cursor for select empid from emp_dat
where empid>r1;
  declare continue handler for not found set
exit_loop=true;
  open c1;
  e_loop:loop
  fetch c1 into r2;
  if not exists(select * from emp_update where
empid=r2)
  then
  insert into emp_update select * from emp_dat where
empid=r2;
  end if;
  if exit_loop
  then
  close c1;
  leave e_loop;
  end if;
  end loop e_loop;
  end
  //
call p3(3);//
select * from emp_update;
//
call p3(0);//
select * from emp_update;//
insert into emp_dat values(9696,'Jordi','USA'); //
select * from emp_update;
```

Write a PL/SQL block of code using parameterized Cursor.also execute all types of cursor

NOT CONIRMED

```
Create table cnew(sname varchar2(20));
Create table cold(sname varchar2(20));
insert into cnew values('ABC');
insert into cold values('ABC');
insert into cold values('PQR');
insert into cold values('XYZ');

declare
   cursor cn is
   select sname from cnew;

   cursor co is
   select sname from cold;

   newv varchar2(20);
   oldv varchar2(20);

begin
   open cn;
   open co;

   loop
      fetch cn into newv;
```

```
        fetch co into oldv;


        exit when co%found=false;

        if(newv <> oldv) then

           insert into cnew values(oldv);

        end if;

    end loop;


end;

/


select * from cnew
```

Write a program to implement MySQL/Oracle database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.)

ASK FOR DIFFERENT P   ROBLEM STATEMENT…

THIS CODE NEEDS DEPENDANCIES

```
import java.net.UnknownHostException;

import java.util.Scanner;

import com.mongodb.*;

public class DatabaseConnectivity {

private static void choice_input(){

System.out.println("\n1.insert data into database\n2.update database

documents\n3.delete database documents\n4.show database

collections\n5.Exit");

}

public static void main(String[] args) {

String key, value;

Scanner scanner = new Scanner(System.in);

int choice;

try {
```

```
Mongo mongo = new Mongo("localhost", 27017);
DB db = mongo.getDB("myDb");
DBCollection collection = db.getCollection("dummyColl");
do{
choice_input();
System.out.println("Enter your choice: ");
choice = scanner.nextInt();
switch (choice){
case 1:
BasicDBObject document = new BasicDBObject();
String ch;
do{
System.out.println("Enter key: ");
key = scanner.next();
System.out.println("Enter value: ");
value = scanner.next();
document.put(key, value);
System.out.println("Do you want to enter more(y/n)? ");
ch = scanner.next();
} while (!ch.equals("n"));
collection.insert(document);
break;
case 2:
BasicDBObject searchObj = new BasicDBObject();
System.out.println("Enter searched key: ");
key = scanner.next();
System.out.println("Enter searched value: ");
value = scanner.next();
searchObj.put(key, value);
BasicDBObject newObj = new BasicDBObject();
System.out.println("Enter new key: ");
key = scanner.next();
System.out.println("Enter new value: ");
value = scanner.next();
newObj.put(key, value);
collection.update(searchObj, newObj);
break;
case 3:
System.out.println("Enter removable key: ");
key = scanner.next();
System.out.println("Enter removable value: ");
value = scanner.next();
BasicDBObject removableObj = new BasicDBObject();
removableObj.put(key, value);
collection.remove(removableObj);
break;
case 4:
DBCursor cursorDoc = collection.find();
while (cursorDoc.hasNext()) {
System.out.println(cursorDoc.next());
}
break;
case 5:
System.exit(0);
break;
}
} while(choice != 6);
} catch (UnknownHostException | MongoException e) {
e.printStackTrace();
}
}
}
```

Design and Develop MongoDBQueries using CRUD operations.(Use CRUDoperations,SAVE method, logical operators etc.).

```
use mydb

db.createCollection('stud');

db.stud.insert({rollno:1, name:'ZZZ',marks:91});

db.stud.find().pretty()

db.stud.insert([{rollno:2, name:'AAA', marks:90}, {rollno:3, name:'BBB', marks:80},
{rollno:4, name:'CCC', marks:70}, {rollno:5, name:'DDD', marks:85}, {rollno:6, name:'EEE',
marks:90}, {rollno:7, name:'FFF', marks:95}, {rollno:8, name:'GGG', marks:70}, {rollno:9,
name:'HHH', marks:60}, {rollno:10, name:'III', marks:98}]);

db.stud.find().pretty()

db.stud.find({name:'AAA'}).pretty();

db.stud.find({},{name:1}).pretty();

db.stud.find({},{name:1,rollno:1}).pretty();

db.stud.find({name:'ZZZ',rollno:1}).pretty();

db.stud.find({marks:{$lt:70}}).pretty();

db.stud.find({marks:{$gt:70}}).pretty();

db.stud.find({marks:{$gte:90}}).pretty();

db.stud.find({marks:{$lte:80}}).pretty();

db.stud.update({rollno:2},{$set:{name:'AAA'}});

db.stud.find().pretty()


db.stud.remove({name:'xyz',rollno:1,marks:90});

db.stud.find({name:{$in:['FFF','III','JJJ']}});

db.stud.ensureIndex({name:1});

db.stud.getIndexes();

db.stud.dropIndex({name:1});

db.stud.getIndexes();
```

Implement Map reduce operation with suitable examples using MongoDB.

```
db.createCollection("orders")

use orders

db.orders.insertMany([{ _id: 1, cust_id: "AAAAA", price: 25, items: [ { sku: "oranges", qty: 5, price: 2.5 }, { sku: "apples", qty: 5, price: 2.5 } ]}, { _id: 2, cust_id: "AAAAA", price: 70, items: [ { sku: "oranges", qty: 8, price: 2.5 }, { sku: "chocolates", qty: 5, price: 10 } ]},{ _id: 3, cust_id: "BBBBB", price: 50, items: [ { sku: "oranges", qty: 10, price: 2.5 }, { sku: "pears", qty: 10, price: 2.5 } ]}, { _id: 4, cust_id: "BBBBB", price: 25, items: [ { sku: "oranges", qty: 10, price: 2.5 } ]}, { _id: 5, cust_id: "BBBBB", price: 50, items: [ { sku: "chocolates", qty: 5, price: 10 } ]}, { _id: 6, cust_id: "CCCCC", price: 35, items: [ { sku: "carrots", qty: 10, price: 1.0 }, { sku: "apples", qty: 10, price: 2.5 } ]}, { _id: 8, cust_id: "DDDDD", price: 75, items: [ { sku: "chocolates", qty: 5, price: 10 }, { sku: "apples", qty: 10, price: 2.5 } ]},{ _id: 9, cust_id: "DDDDD", price: 55, items: [ { sku: "carrots", qty: 5, price: 1.0 }, { sku: "apples", qty: 10, price: 2.5 }, { sku: "oranges", qtyqty: 10, price: 2.5 } ]},{ _id: 7, cust_id: "CCCCC", price: 25, items: [ { sku: "oranges", qty: 10, price: 2.5 } ]},{ _id: 10, cust_id: "DDDDD", price: 25, items: [ { sku: "oranges", qty: 10, price: 2.5 } ]} ])

var mapFunction1 = function() {emit(this.cust_id, this.price);};

var reduceFunction1 = function(keyCustId, valuesPrices) {return Array.sum(valuesPrices);};

db.orders.mapReduce(mapFunction1,reduceFunction1,{ out: "map_reduce_example" })

db.map_reduce_example.find().sort( { _id: 1 } )
```

Write a program to implement MongoDB database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.)

ASK FOR DIFFERENT…THIS CODE NEEDS DEPENDANCIES

```
import com.mongodb.client.MongoDatabase;

import com.mongodb.MongoClient;

import com.mongodb.MongoCredential;

public class ConnectToDB {


  public static void main( String args[] ) {

    // Creating a Mongo client
}


    MongoClient mongo = new MongoClient( "localhost" , 27017 );


    // Creating Credentials
    MongoCredential credential;
    credential = MongoCredential.createCredential("sampleUser", "myDb",
      "password".toCharArray());
    System.out.println("Connected to the database successfully");


    // Accessing the database
    MongoDatabase database = mongo.getDatabase("myDb");
    System.out.println("Credentials ::"+ credential);


database.createCollection("sampleCollection");
    System.out.println("Collection created successfully");


Document document = new Document("title", "MongoDB")
        .append("description", "database")
        .append("likes", 100)
        .append("url", "http://www.tutorialspoint.com/mongodb/")
        .append("by", "tutorials point");


        //Inserting document into the collection
        collection.insertOne(document);
        System.out.println("Document inserted successfully");
collection.deleteOne(Filters.eq("title", "MongoDB"));
    System.out.println("Document deleted successfully...");


}


  }
```

}

Create table employee with attributes emp_id, emp_name,dept, city etc. implement group functions like max,min, count,sum on employee table.Find top two highest salaries from salary attributes.

```
create database dbms;
create table a1;
use dbms;
create table  employee(emp_id int primary key,emp_name
varchar (50),dept varchar(20),city varchar(20));

//INSERT DATA QUERIES in the following

+--------+----------+------+------+---------+
| emp_id | emp_name | dept | city | emp_sal |
+--------+----------+------+------+---------+
|      1 | Rush     | comp | beed |   60000 |
|      2 | Vin      | comp | pune |   67000 |
|      3 | Kya      | comp | pune |    7000 |
|      4 | Dekh     | comp | pune |  100000 |
|      5 | Rha      | comp | pune |  150000 |
|      6 | He       | comp | pune |  151244 |
|      7 | Lo     | comp | pune |  120000 |

select max(emp_sal) from employee;
select min(emp_sal) from employee;
select count(*) from employee;
select sum(emp_sal) from employee;
select max(emp_sal) from employee union select max(emp_sal) from employee where
emp_sal not in(select max(emp_sal) from employee) ;
```

Create Student and Branch tables. In the students table create attributes like Rollno, Name,Class, Div,City, DOB. Branch table with Branch_Id,Branch_name,Rollno. Build relationship between the table using primary key and Foreign key.

```
create database SITS;
 use SITS;
create table student(roll int primary key, name varchar(30),class varchar(30),division int, city
 varchar(30),dob DATE);

create table branch(bid int primary key,bname varchar(30),roll int references student(roll));

update branch set bname="Bas Kar Bhai" where roll=4;
```