Chit 1:

```
1. select Cname, Acc_Type, Amount from Customer c, Account a where c.C_Id = a.C_id
and Acc_Type = "Saving";

2. select * from Customer natural join Account;
   select * from Customer c left join Account a on c.C_Id = a.C_Id;
   select * from Customer c right join Account a on c.C_Id = a.C_Id;

3. select * from Customer where City = (select City from Customer where Cname =
"Pooja");

4. select * from Account where Amount < (select avg(Amount) from Account);

5. select C_Id from Account where Amount = (select max(Amount) from Account);

6. select Acc_Type, min(Amount) from Account group by Acc_Type;

7. select Amount from Account where Amount > (select min(Amount) from Account where
Acc_Type = "Saving");
```

Chit 2:

```
SET SERVEROUTPUT ON;

DECLARE
RNO INT;
NOD INT;
NOB BORROWER.NAMEOFBOOK%TYPE;
AMOUNT INT;
DOI DATE;

BEGIN
RNO := &RNO;
NOB := &NOB;

SELECT ISSUEDATE INTO DOI FROM BORROWER WHERE ROLL_NO = RNO AND NAMEOFBOOK = NOB;
NOD := SYSDATE - DOI;

IF (NOD > 30) THEN
AMOUNT := 50 * NOD;
UPDATE BORROWER SET STATUS = 'R' WHERE ROLL_NO = RNO;
INSERT INTO FINE VALUES(RNO, SYSDATE, AMOUNT);
ELSIF (NOD < 30 AND NOD >= 15) THEN
AMOUNT := 5 * NOD;
UPDATE BORROWER SET STATUS = 'R' WHERE ROLL_NO = RNO;
INSERT INTO FINE VALUES(RNO, SYSDATE, AMOUNT);
ELSE
UPDATE BORROWER SET STATUS = 'R' WHERE ROLL_NO = RNO;
DBMS_OUTPUT.PUT_LINE('NO FINE');
END IF;

EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('NO DATA AVAILABLE');
END;
```

Chit 3:

```
SET SERVEROUTPUT ON;

DECLARE
RNO INT;
ATT INT;

BEGIN
RNO := &RNO;
SELECT ATTENDANCE INTO ATT FROM STUD WHERE ROLL = RNO;

IF (ATT < 75) THEN
UPDATE STUD SET STATUS = 'D' WHERE ROLL = RNO;
DBMS_OUTPUT.PUT_LINE('TERM NOT GRANTED');
ELSE
UPDATE STUD SET STATUS = 'ND' WHERE ROLL = RNO;
DBMS_OUTPUT.PUT_LINE('TERM GRANTED');
END IF;

EXCEPTION
WHEN NO_DATA_FOUND THEN
DBMS_OUTPUT.PUT_LINE('NO DATA AVAILABLE');
END;
```

Chit 4:

1.
```
chit4> var map = function(){emit (this.state, this.population)};
chit4> var reduce = function(k, v) {return Array.sum(v)};
chit4> db.city.mapReduce(map, reduce, {out : 'statewise'});
chit4> db.statewise.find()
```

2.
```
chit4> var map = function(){emit (this.city, this.population)};
chit4> var reduce = function(k, v) {return Array.sum(v)};
chit4> db.city.mapReduce(map, reduce, {out : 'citywise'});
chit4> db.citywise.find()
```

3.
```
chit4> var map = function(){emit (this.type, this.population)};
chit4> var reduce = function(k, v) {return Array.sum(v)};
chit4> db.city.mapReduce(map, reduce, {out : 'typewise'});
chit4> db.typewise.find()
```

4.
```
chit4> var map = function(){emit (this.city, 1)};
chit4> var reduce = function(k, v) {return Array.sum(v)};
chit4> db.city.mapReduce(map, reduce, {out : 'citywisecount'});
chit4> db.citywisecount.find()
```

5.
```
chit4> var map = function(){emit (this.state, 1)};
chit4> var reduce = function(k, v) {return Array.sum(v)};
chit4> db.city.mapReduce(map, reduce, {out : 'statewisecount'});
chit4> db.statewisecount.find()
```

Chit 5:

Chit 6:

```
create or replace NONEDITIONABLE FUNCTION FUNC_1
(
  R IN NUMBER
, N IN VARCHAR2
, M IN NUMBER
) RETURN VARCHAR2 AS
BEGIN
  PROCEDURE_1(R,N,M);
  RETURN 'SUCCESSFULL';
END FUNC_1;

create or replace NONEDITIONABLE PROCEDURE PROCEDURE_1
(
  ROLL_NO IN NUMBER
, NAME IN VARCHAR2
, MARKS IN NUMBER
) AS
BEGIN
IF (MARKS <= 1500 AND MARKS >= 990) THEN
DBMS_OUTPUT.PUT_LINE('DISTINCTION');
INSERT INTO RESULT VALUES(ROLL_NO, NAME, 'DISTINCTION');
ELSIF (MARKS <= 989 AND MARKS >= 900) THEN
DBMS_OUTPUT.PUT_LINE('FIRST CLASS');
INSERT INTO RESULT VALUES(ROLL_NO, NAME, 'FIRST CLASS');
ELSIF (MARKS <= 899 AND MARKS >= 825) THEN
DBMS_OUTPUT.PUT_LINE('HIGHER SECOND CLASS');
INSERT INTO RESULT VALUES(ROLL_NO, NAME, 'HIGHER SECOND CLASS');
ELSE
DBMS_OUTPUT.PUT_LINE('FAIL');
INSERT INTO RESULT VALUES(ROLL_NO, NAME, 'FAIL');
END IF;

INSERT INTO STUD_MARKS VALUES(NAME, MARKS);
END PROCEDURE_1;

SET SERVEROUTPUT ON;

DECLARE
NAME_1 VARCHAR2(20);
ROLL_NO_1 NUMBER;
MARKS_1 NUMBER;
CLASS VARCHAR2(20);

BEGIN
ROLL_NO_1 := &ROLL_NO_1;
NAME_1 := &NAME_1;
MARKS_1 := &MARKS_1;
CLASS := FUNC_1(ROLL_NO_1, NAME_1, MARKS_1);
DBMS_OUTPUT.PUT_LINE(CLASS);

END;
/
```

Chit 7:

```
create or replace NONEDITIONABLE FUNCTION FUNCTION1
(
```

```
  ROLL_NO IN NUMBER
, NAME IN VARCHAR2
, MARKS IN NUMBER
) RETURN VARCHAR2 AS
BEGIN
IF (MARKS <= 1500 AND MARKS >= 990) THEN
DBMS_OUTPUT.PUT_LINE('DISTINCTION');
INSERT INTO RESULT VALUES(ROLL_NO, NAME, 'DISTINCTION');
ELSIF (MARKS <= 989 AND MARKS >= 900) THEN
DBMS_OUTPUT.PUT_LINE('FIRST CLASS');
INSERT INTO RESULT VALUES(ROLL_NO, NAME, 'FIRST CLASS');
ELSIF (MARKS <= 899 AND MARKS >= 825) THEN
DBMS_OUTPUT.PUT_LINE('HIGHER SECOND CLASS');
INSERT INTO RESULT VALUES(ROLL_NO, NAME, 'HIGHER SECOND CLASS');
ELSE
DBMS_OUTPUT.PUT_LINE('FAIL');
INSERT INTO RESULT VALUES(ROLL_NO, NAME, 'FAIL');
END IF;

INSERT INTO STUD_MARKS VALUES(NAME, MARKS);

RETURN 'SUCCESSFULL';
END FUNCTION1;

SET SERVEROUTPUT ON;

DECLARE
NAME_1 VARCHAR2(20);
ROLL_NO_1 NUMBER;
MARKS_1 NUMBER;
CLASS VARCHAR2(20);

BEGIN
ROLL_NO_1 := &ROLL_NO_1;
NAME_1 := &NAME_1;
MARKS_1 := &MARKS_1;
CLASS := FUNCTION1(ROLL_NO_1, NAME_1, MARKS_1);
DBMS_OUTPUT.PUT_LINE(CLASS);

END;
/
```

Chit 8:

1.
```
chit8> var map = function(){emit (this.class, 1)}
chit8> var reduce = function(k, v) {return Array.sum(v)}
chit8> db.Student.mapReduce(map, reduce, {out:'classwisestudents'})
chit8> db.classwisestudents.find()
```

2.
```
chit8> var map = function(){emit (this.class, this.fees)}
chit8> var reduce = function(k, v) {return Array.sum(v)}
chit8> db.Student.mapReduce(map, reduce, {out:'classwisefees'})
chit8> db.classwisefees.find()
```

3.
```
chit8> var map = function(){emit (this.subject, this.marks)}
chit8> var reduce = function(k, v) {return Array.sum(v)}
chit8> db.Student.mapReduce(map, reduce, {out:'subjectwisemarks'})
chit8> db.subjectwisemarks.find()
```

4. chit8> var map = function(){emit (this.subject, 1)}
   chit8> var reduce = function(k, v) {return Array.sum(v)}
   chit8> db.Student.mapReduce(map, reduce, {out:'subjectwisestudents'})
   chit8> db.subjectwisestudents.find()

Chit 9:

1. db.orderinfo.aggregate([{$group:{_id:"$name", total_price:
{$sum:"$price"}}}]).sort({total_price:1})

2. db.orderinfo.aggregate([{$group:{_id:"$name"}}])

3. db.orderinfo.find({status:'A'}, {_id:1, price:1})

4. db.orderinfo.deleteMany({status:'A'})

Chit 10:

```
import com.mongodb.*;

import java.util.Scanner;

public class Chit10 {
    public static void main(String[] args) {
        MongoClient client = new MongoClient("localhost", 27017);
        DB db1 = client.getDB("chit10");

        // Creating a collection
        DBCollection col = db1.createCollection("Students",
                null);
        System.out.println("Collection created");

        // Inserting a document
        BasicDBObject obj1 = new BasicDBObject();
        obj1.append("Roll", 65);
        obj1.append("Name", "Siddhesh");
        col.save(obj1);
        System.out.println("Document inserted");

        // Displaying a document
        System.out.println("Displaying the document");
        DBCursor cur1 = col.find();
        while(cur1.hasNext()) {
            System.out.println(cur1.next());
        }

        // Removing a document
        BasicDBObject obj2 = new BasicDBObject();
        obj2.append("Roll", 1);
        obj2.append("Name", "A");
        col.save(obj2);

        System.out.println("Before removing the document");
        DBCursor cur2 = col.find();
        while(cur2.hasNext()) {
            System.out.println(cur2.next());
```

```
        }

        col.remove(obj2);

        System.out.println("After removing the document");
        DBCursor cur3 = col.find();
        while(cur3.hasNext()) {
            System.out.println(cur3.next());
        }

        client.close();
    }
}
```

Chit 11:

```
CREATE OR REPLACE TRIGGER TRIGGER1
BEFORE DELETE OR INSERT OR UPDATE ON LIB_TAB
REFERENCING OLD AS OLD NEW AS NEW
FOR EACH ROW
DECLARE
TRIGGER1 CHAR;
BEGIN
IF UPDATING THEN
DBMS_OUTPUT.PUT_LINE(:OLD.STATUS);
INSERT INTO LIB_AUDIT VALUES(SYSDATE, :OLD.BOOK_NAME, :OLD.STATUS, :NEW.STATUS,
'UPDATE');
ELSIF INSERTING THEN
DBMS_OUTPUT.PUT_LINE(:NEW.STATUS);
INSERT INTO LIB_AUDIT VALUES(SYSDATE, :NEW.BOOK_NAME, :OLD.STATUS, :NEW.STATUS,
'INSERT');
ELSE
DBMS_OUTPUT.PUT_LINE(:OLD.BOOK_NAME || 'DELETING');
INSERT INTO LIB_AUDIT VALUES(SYSDATE, :OLD.BOOK_NAME, :OLD.STATUS, :NEW.STATUS,
'DELETE');
END IF;
END;
```

Chit 12:

```
create table account(acc_no int primary key, branch_name varchar(20) not null,
balance int not null, check(balance >= 0));

create table branch(branch_name varchar(20) not null, branch_city varchar(20) not
null, assets int not null check(assets >= 0));

create table customer(cust_name varchar(20) not null, cust_street int not null,
cust_city varchar(20) not null);

create table depositor(cust_name varchar(20) not null, acc_no int, foreign key
(acc_no) references account(acc_no));

create table loan(loan_no int primary key, branch_name varchar(20) not null, amount
int not null, check(amount >= 0));

create table borrower(cust_name varchar(20) not null, loan_no int, foreign key
(loan_no) references loan(loan_no) on delete cascade);
```

```
1. select distinct branch_name from loan;

2. select loan_no from loan where branch_name = 'Akurdi' and amount > 12000;

3. select cust_name, loan.loan_no, amount from loan inner join borrower on
loan.loan_no = borrower.loan_no;

4. select cust_name from loan l, borrower b where l.loan_no = b.loan_no and
branch_name = 'Akurdi' order by cust_name;

5. select cust_name from account a, depositor d where a.acc_no = d.acc_no union
select cust_name from loan l, borrower b where l.loan_no = b.loan_no;

6. select avg(balance) from account where branch_name = 'Akurdi';

7. select branch_name, avg(balance) from account group by branch_name;

8. select branch_name, count(*) from depositor d, account a where a.acc_no =
d.acc_no group by branch_name;

9. select branch_name, avg(balance) from account group by branch_name having
avg(balance) > 12000;

10. select count(*) from customer;

11. delete from loan where amount between 1300 and 1500;



Chit 13:
1. create table jobs(job_id int primary key, job_desc varchar(20));

2. create table employee(employee_id int primary key, first_name varchar(20),
last_name varchar(20), job_id int, salary int,
   foreign key (job_id) references jobs(job_id) on delete no action on update no
action);

Try some update and delete queries



Chit 14:
1. chit14> db.createCollection('Teachers')

   chit14> db.createCollection('Department')

   chit14> db.createCollection('Students')

   chit14> db.Teachers.insertMany([{Tname:'T1', Dno:1, Experience:13, Salary:17000,
Date_of_joining:new Date(2012-07-21)},
           {Tname:'T2', Dno:2, Experience:17, Salary:9000, Date_of_joining:new
Date(2009-02-15)},
           {Tname:'T3', Dno:1, Experience:8, Salary:21000, Date_of_joining:new
Date(2014-11-29)},
           {Tname:'T4', Dno:2, Experience:19, Salary:7000, Date_of_joining:new
Date(2010-04-19)}])

   chit14> db.Department.insertMany([{Dno:1, Dname:'COMP'}, {Dno:2, Dname:'ENTC'}])
```

```
   chit14> db.Students.insertMany([{Sname:'abc', Roll_no:1, Class:'FE'},
{Sname:'def', Roll_no:2, Class:'SE'},
           {Sname:'xyz', Roll_no:3, Class:'TE'}, {Sname:'ghi', Roll_no:4,
Class:'BE'}, {Sname:'jkl', Roll_no:5, Class:'FE'}])

2. db.Teachers.find({$and:[{Dno:2}, {Salary:{$gte:10000}}]})

3. db.Students.find({$or:[{Roll_no:2}, {Sname:'xyz'}]})

4. db.Students.update({Roll_no:5}, {$set:{Sname:'Sid'}})

5. db.Students.deleteMany({Class:'FE'})

6. db.Teachers.find({Experience:{$gt:10}})

7. db.Students.createIndex({Roll_no:1})
```

Chit 15:

```
1. use Institute

2. db.createCollection('Student')

3. db.Student.insertMany([{RollNo:1, Name:'Yuvraj', Age:16, Branch:'Computer',
Address:{City:'Mumbai', State:'Maharashtra'}, Hobbies:['H1', 'H3']},
   {RollNo:2, Name:'Aman', Age:14, Branch:'Computer', Address:{City:'Bangalore',
State:'Karnataka'}, Hobbies:['H1', 'H4', 'H5']},
   {RollNo:3, Name:'Amit', Age:12, Branch:'ENTC', Address:{City:'Nagar',
State:'Maharashtra'}, Hobbies:['H2', 'H4']},
   {RollNo:4, Name:'John', Age:19, Branch:'Civil', Address:{City:'Ahmedabad',
State:'Gujarat'}, Hobbies:['H3', 'H5']},
   {RollNo:5, Name:'Simon', Age:20, Branch:'Mech', Address:{City:'Delhi',
State:'Delhi'}, Hobbies:['H2', 'H4', 'H5']},
   {RollNo:6, Name:'Paul', Age:14, Branch:'ENTC', Address:{City:'Indore',
State:'Madhya Pradesh'}, Hobbies:['H2', 'H3']},
   {RollNo:7, Name:'Toby', Age:21, Branch:'Civil', Address:{City:'Panaji',
State:'Goa'}, Hobbies:['H4', 'H5']},
   {RollNo:8, Name:'Peter', Age:15, Branch:'Mech', Address:{City:'Guwahati',
State:'Assam'}, Hobbies:['H2', 'H4']},
   {RollNo:9, Name:'Sunil', Age:11, Branch:'Computer', Address:{City:'Pune',
State:'Maharashtra'}, Hobbies:['H1', 'H5']},
   {RollNo:10, Name:'Deepak', Age:22, Branch:'ENTC', Address:{City:'Mumbai',
State:'Maharashtra'}, Hobbies:['H1', 'H3']}])

4. db.Student.find()

5. db.Student.find({Age:{$gt:15}})

6. db.Student.find().sort({Name:1})

7. db.Student.update({RollNo:3}, {$set:{Branch:'Computer'}})

8. db.Student.deleteOne({RollNo:1})

9. db.Student.find({Name:/^A/})

10. db.Student.find().count()
```

```
11. db.Student.find().limit(5)

12. db.Student.find().skip(3)

13. db.Student.find({Address:{City:'Pune', State:'Maharashtra'}})

14. db.Student.distinct("Address.City")

15. var map = function(){emit(this.Address.City, 1)}
    var reduce = function(k, v) {return Array.sum(v)}
    db.Student.mapReduce(map, reduce, {out:'citycount'})
    db.citycount.find()

16. db.Student.find({}, {Name:1})

17. db.Student.find({}, {Name:1, Hobbies:1})

18. db.Student.drop()
```

Chit 16:

```
1. use department

2. db.createCollection('teacher')
   db.teacher.insertMany([{name:'A', department:'Computer', experience:13,
salary:17000}, {name:'B', department:'ENTC', experience:11, salary:12000},
   {name:'C', department:'Computer', experience:15, salary:18300}, {name:'D',
department:'ENTC', experience:15, salary:22400}])

3. db.teacher.aggregate([{$group:{_id:"$department", average_salary:
{$avg:"$salary"}}}])

4. db.teacher.aggregate([{$group:{_id:"$department", no_of_employees:{$sum:1}}}])

5. db.teacher.aggregate([{$group:{_id:"$department", min_salary:
{$min:"$salary"}}}])

6. db.teacher.createIndex({name:1})
   db.teacher.dropIndex({name:1})
```

Chit 17:

```
1. select book.book_id, title, publisher_name, author_name, branch_name,
no_of_copies from book
   join book_authors on book.book_id = book_authors.book_id join book_copies on
book.book_id = book_copies.book_id
   join library_branch on book_copies.branch_id = library_branch.branch_id;

2. select title, branch_name, card_no, date_out, due_date from book_lending
   join book on book_lending.book_id = book.book_id join library_branch on
book_lending.branch_id = library_branch.branch_id
   where date_out between '2017-01-01' and '2017-06-30';

3. delete from book where title = 'Databases';

4. select pub_year, count(*) as total_no_of_books from book group by pub_year order
```

```
by pub_year;

5. create view books as select title, sum(no_of_copies) as copies from book
   join book_copies on book.book_id = book_copies.book_id group by book.book_id;



Chit 18:
1. select distinct address from employee;

2. select max(salary), min(salary) from employee;

3. select * from employee order by salary;

4. select ename from employee where address = 'Nasik' or address = 'Pune';

5. select ename from employee where commission is null;

6. update employee set address = 'Nasik' where ename = 'Amit';

7. select * from employee where ename like 'A%';

8. select count(*) from employee where address = 'Mumbai';

9. select address, count(*) from employee group by address;

10. select distinct address from employee, project where address = addr;

11. select address, min(salary) from employee group by address;

12. select address, max(salary) from employee group by address having max(salary) >
26000;

13. delete from employee where salary > 30000;



Chit 19:
1. create table emp(eno int auto_increment primary key, ename varchar(20) not null,
address varchar(20) default 'Nashik', joindate date);
   alter table emp auto_increment = 101;

2. alter table emp add post varchar(20);

3. insert into emp(ename, address, salary, joindate, post) values('Amit', 'Pune',
25000, '2017-05-23', 'HR'),
   ('Sneha', 'Pune', 35000, '2018-11-17', 'Manager'), ('Savita', 'Nashik', 28000,
'2019-05-21', 'Tech Head'),
   ('Pooja', 'Mummbai', 19000, '2020-01-27', 'PR Head'), ('Sagar', 'Mumbai', 25000,
'2021-09-12', 'Marketing Head');

   create index emp_name on emp(ename);

4. insert into emp(ename, salary, joindate, post) values('Siddhesh', 21000, '2022-
03-27', 'Head');

5. create view emp_data as select ename, salary from emp;
   select * from emp_data;
```

Chit 20:

Indexing:

```
1. db.createCollection('Student')

2. db.Student.insertMany([{rollno:1, name:'navin', subject:'DMSA', marks:78},
{rollno:2, name:'anusha', subject:'OSD', marks:75},
   {rollno:3, name:'ravi', subject:'TOC', marks:69}, {rollno:4, name:'veena',
subject:'TOC', marks:70},
   {rollno:5, name:'pravini', subject:'OSD', marks:80}, {rollno:6, name:'reena',
subject:'DMSA', marks:50},
   {rollno:7, name:'geeta', subject:'CN', marks:90}, {rollno:8, name:'akash',
subject:'CN', marks:85}])

3. db.Student.createIndex({rollno:1})

4. db.Student.createIndex({rollno:1, name:1})

5. db.Student.createIndex({name:1}, {unique:true})

6. db.Student.getIndexes()

7. db.Student.dropIndex('name_1')
```

Aggregation:

```
1. db.Student.aggregate([{$group:{_id:"$subject", max_marks:{$max:"$marks"}}}])

2. db.Student.aggregate([{$group:{_id:"$subject", min_marks:{$min:"$marks"}}}])

3. db.Student.aggregate([{$group:{_id:"$subject", total_marks:{$sum:"$marks"}}}])

4. db.Student.aggregate([{$group:{_id:"$subject", average_marks:{$avg:"$marks"}}}])

5. db.Student.aggregate([{$group:{_id:"$subject", rollno:{$first:"$rollno"}, name:
{$first:"$name"},
   subject:{$first:"$subject"}, marks:{$first:"$marks"}}}])

6. db.Student.aggregate([{$group:{_id:"$subject", rollno:{$last:"$rollno"}, name:
{$last:"$name"},
   subject:{$last:"$subject"}, marks:{$last:"$marks"}}}])

7. db.Student.aggregate([{$group:{_id:"$subject", count:{$sum:1}}}])

8. db.Student.aggregate([{$group:{_id:"$subject", count:{$sum:1}}}])
```

Chit 21:

```
import com.mongodb.*;

import java.util.Scanner;

public class Chit21 {
    public static void main(String[] args) {
        int ch, i = 0;
        String str;
```

```
        MongoClient client = new MongoClient("localhost", 27017);
        DB db1 = client.getDB("chit21");
        DBCollection col = db1.createCollection("Students", null);

        Scanner sc = new Scanner(System.in);

        do {
            System.out.println("1.Insert\n2.Display\n3.Update\n4.Delete\n5.Drop\
n6.Exit");
            System.out.println("Enter a choice:");

            ch = Integer.parseInt(sc.nextLine());

            switch(ch) {
                case 1:
                    System.out.println("Enter name:");
                    str = sc.nextLine();
                    BasicDBObject obj1 = new BasicDBObject().append("Name", str);
                    col.save(obj1);
                    System.out.println("Inserted");
                    break;

                case 2:
                    DBCursor cur = col.find();
                    while(cur.hasNext()) {
                        System.out.println(cur.next());
                    }
                    break;

                case 3:
                    System.out.println("Enter name to be replaced:");
                    str = sc.nextLine();
                    BasicDBObject obj2 = new BasicDBObject().append("Name", str);
                    System.out.println("Enter new name:");
                    str = sc.nextLine();
                    BasicDBObject obj3 = new BasicDBObject().append("Name", str);
                    col.update(obj2, obj3);
                    System.out.println("Updated");
                    break;

                case 4:
                    System.out.println("Enter name:");
                    str = sc.nextLine();
                    BasicDBObject obj4 = new BasicDBObject().append("Name", str);
                    col.remove(obj4);
                    System.out.println("Deleted");
                    break;

                case 5:
                    col.drop();
                    System.out.println("Collection Dropped");
                    break;

                case 6:
                    System.exit(0);
            }

            System.out.println("Enter 1 to continue or 0 to exit");
```

```
            i = Integer.parseInt(sc.nextLine());
        }while(i == 1);

        client.close();
    }
}
```

Chit 22:

```
import java.sql.*;
import java.util.Scanner;

public class Chit22 {
    public static void main(String[] args) throws Exception{
        Class.forName("com.mysql.cj.jdbc.Driver");

        Scanner sc = new Scanner(System.in);

        String url = "jdbc:mysql://localhost:3306/chit22";
        String uname = "root";
        String pass = "2707";

        Connection con = DriverManager.getConnection(url, uname, pass);

        System.out.println("Connection established successfully");

        Statement st = con.createStatement();
        String query;

        int ch, i = 0;

        do {
            System.out.println("1.DML Queries\n2.DQL Queries\n3.DDL Queries");
            System.out.println("Enter a choice:");

            ch = Integer.parseInt(sc.nextLine());

            switch(ch) {
                case 1:
                    query = sc.nextLine();
                    st.execute(query);
                    System.out.println("DML Query executed successfully");
                    break;

                case 2:
                    query = sc.nextLine();
                    ResultSet rs = st.executeQuery(query);
                    while(rs.next()) {
                        System.out.println(rs.getInt(1) + "\t" + rs.getString(2));
                    }
                    System.out.println("DQL Query executed successfully");
                    break;

                case 3:
                    query = sc.nextLine();
                    st.executeUpdate(query);
                    System.out.println("DDL Query executed successfully");
                    break;
```

```
            }

            System.out.println("Enter 1 to continue or 0 to exit");
            i = Integer.parseInt(sc.nextLine());
        }while(i == 1);


        con.close();
    }
}
```

Chit 23:

1. select p.p_name, city from patient p, visit v where date_of_visit = '2017-07-13' and p.p_name = v.p_name;

2. select name, count(*) from physician p, visit v where p.reg_no = v.reg_no group by v.reg_no;
   select date_of_visit, sum(fee) from visit group by date_of_visit;

3. select v.p_name, street, city from patient p, visit v where p.p_name = v.p_name;

4. create view visitors as select * from visit where date_of_visit between '2021-01-01' and '2022-12-31';
   select * from visitors;

5. create index patient_name on patient(p_name);

Chit 24:

db.movies.insertOne({name:'Movie1', type:'action', budget:1000000, producer: {name:'producer1', address:'Pune'}})

1. db.movies.find({budget:{$gt:100000}}, {name:1})

2. db.movies.find({"producer.address":'Pune'}, {"producer.name":1})

3. db.movies.update({type:'action'}, {$set:{type:'horror'}})

4. db.movies.find({"producer.name":'producer1'})

5. db.movies.aggregate([{$group:{_id:'$name'}}])

Chit 25:

1. select * from loan order by amount desc, loan_no;

2. N/A

3. select c_name, c_city, street from customer c join depositor d on c.c_no = d.c_no join borrower b on c.c_no = b.c_no
   join account a on d.acc_no = a.acc_no join loan l on b.loan_no = l.loan_no where l.amount > (3 * a.balance);