

PROGRAM-1

The following are two suggestive databases. The students may use any one or both databases for their core practicals. However, the instructor may provide any other databases for executing these practical.

1. COLLEGE DATABASE:

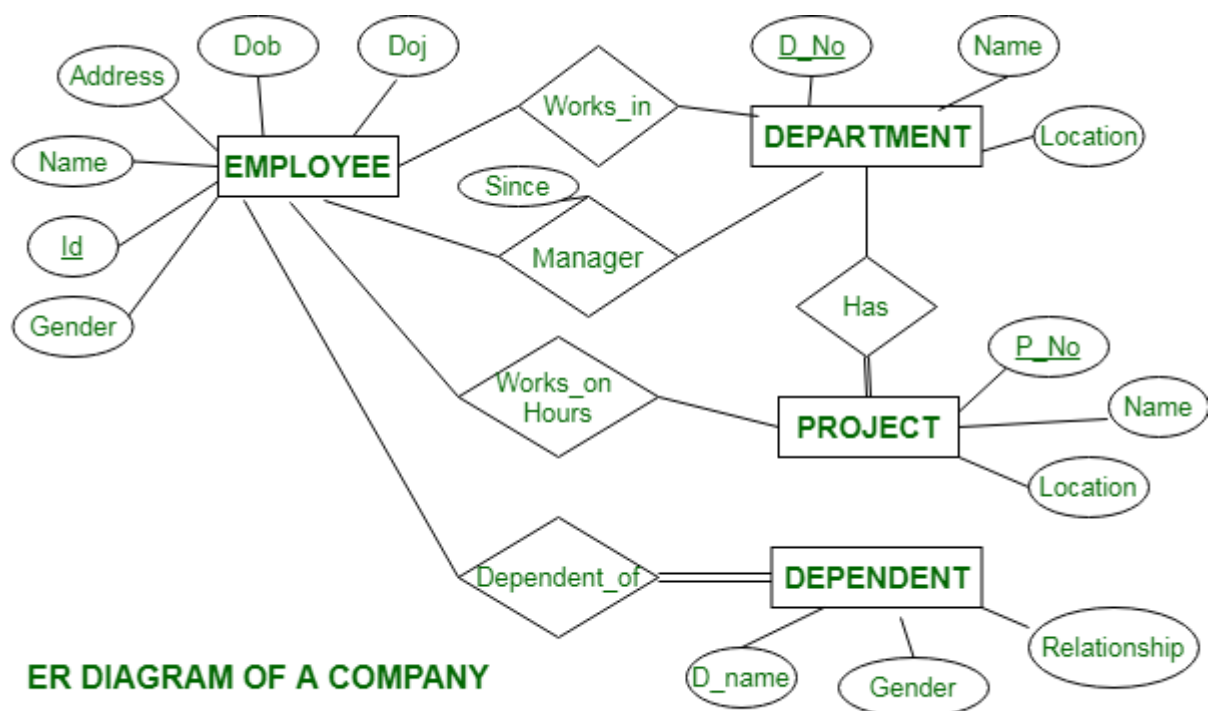
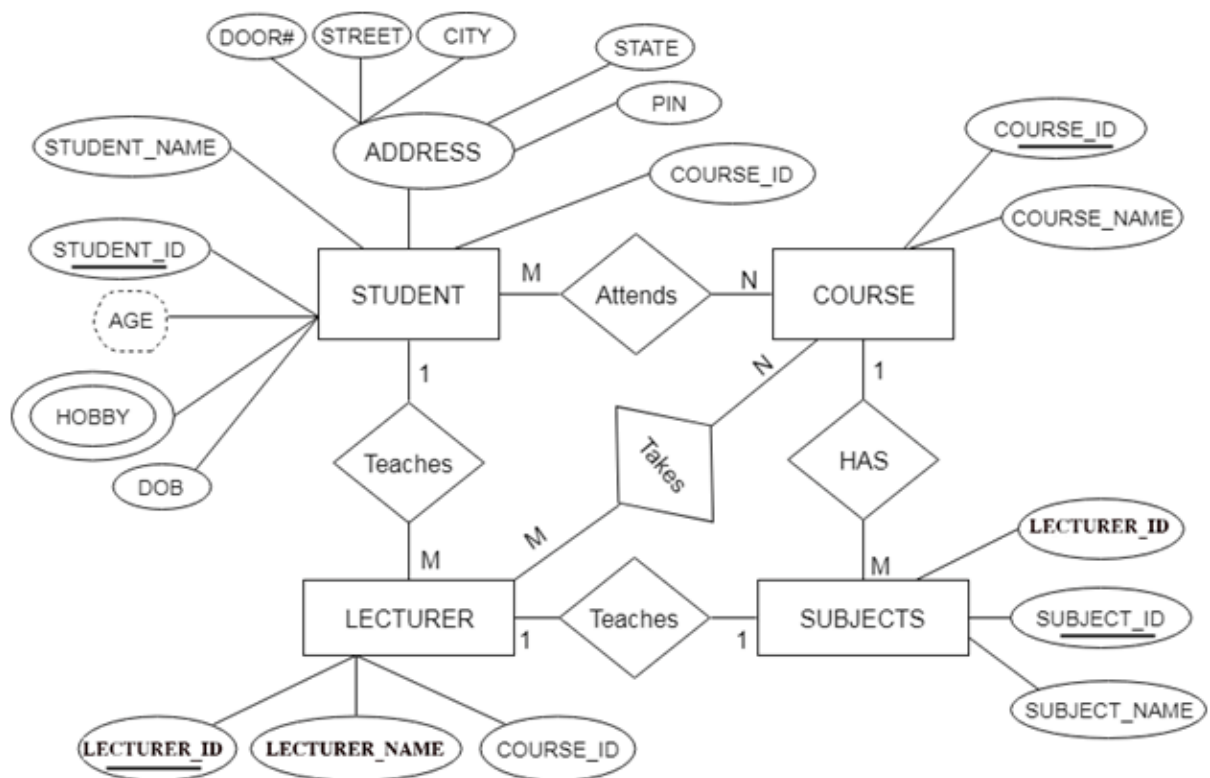
STUDENT (USN, SName, Address, Phone,
Gender)
SEMSEC (SSID, Sem, Sec)
CLASS (USN, SSID)
SUBJECT (Subcode, Title, Sem, Credits)
IAMARKS (USN, Subcode, SSID, Test1, Test2, Test3, FinalIA)

2.COMPANY DATABASE:

EMPLOYEE (SSN, Name, Address, Sex, Salary,
SuperSSN,DNo)
DEPARTMENT (DNo, DName, MgrSSN, MgrStartDate)
DLOCATION (DNo,DLoc)
PROJECT (PNo, PName, PLocation, DNo) WORKS_ON (SSN, PNo, Hours)

Draw an E-R diagram from given entities and their attributes

OUTPUT:



ER DIAGRAM OF A COMPANY

PROGRAM-2

Convert the E-R diagram into a Relational model with proper constraints.

OUTPUT:

ORACLE Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

☒ Autocommit Display 10 Save Run

```
select * from college
```

Results Explain Describe Saved SQL History

| USN | SNAME | ADDRESS | PHONE | GENDER |
|-----|---------|---------------|------------|--------|
| 1 | DEV | GREATER NOIDA | 1342565434 | MALE |
| 2 | VISHAL | GREATER NOIDA | 8765567843 | MALE |
| 3 | MONIKA | GANESH NAGAR | 8765234567 | FEMALE |
| 4 | DEEPIKA | NOIDA | 9345678909 | FEMALE |
| 5 | RIRIKA | VAISHALI | 9876345626 | FEMALE |

5 rows returned in 0.08 seconds [CSV Export](#)

ORACLE Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

☒ Autocommit Display 10 Save Run

```
SELECT*FROM COMPANY
```

Results Explain Describe Saved SQL History

| SSN | NAME | ADDRESS | GENDER | SALARY | DNO |
|-----|--------|---------------|--------|---------|-----|
| 1 | VISHAL | GREATER NOIDA | MALE | 450000 | 1 |
| 2 | DEV | GREATER NOIDA | MALE | 30000 | 5 |
| 3 | RITIKA | VASUNDHARA | FEMALE | 7000000 | 10 |
| 4 | MONIKA | GANESH NAGAR | FEMALE | 45002 | 21 |
| 5 | LIPIKA | KALKAJI | FEMALE | 50000 | 20 |

5 rows returned in 0.00 seconds [CSV Export](#)

PROGRAM-3

Write queries to execute following DDL commands:

CREATE: Create the structure of a table with at least five columns

ALTER: Change the size of a particular column.

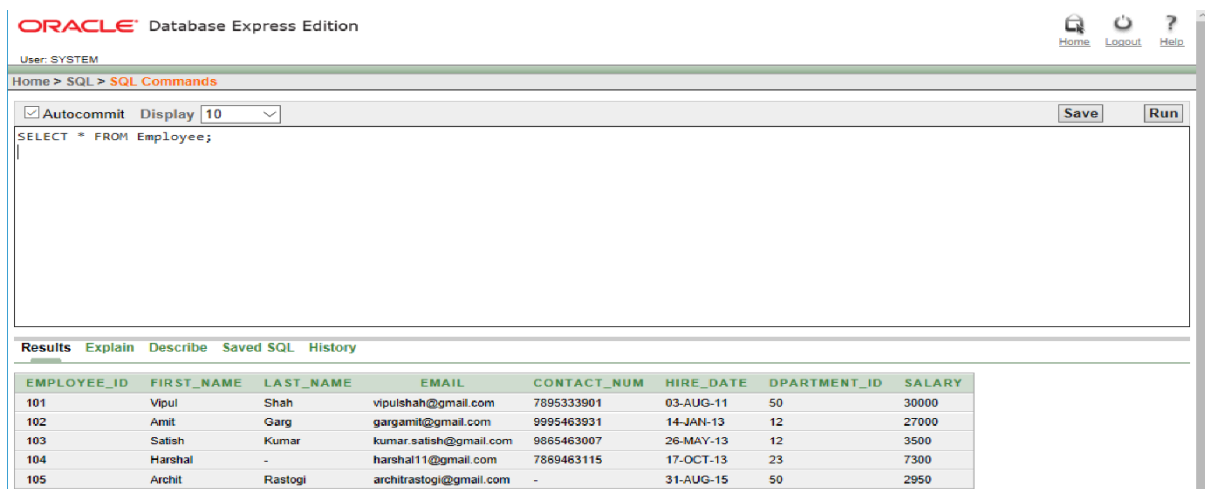
Add a new column to the existing table.

Remove a column from the table.

DROP: Destroy the table along with its data.

OUTPUT:

CREATE-



ORACLE Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

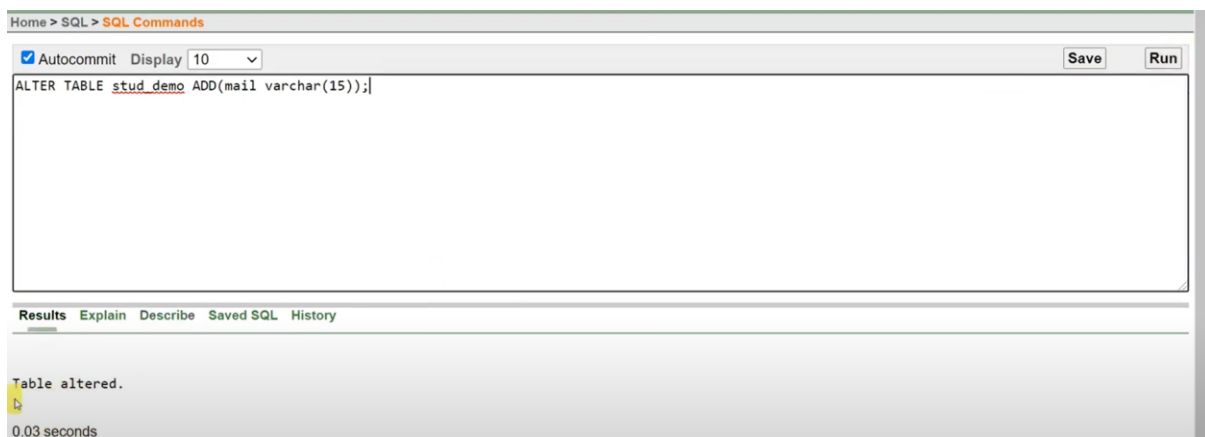
☒ Autocommit Display 10 Save Run

```
SELECT * FROM Employee;
```

Results Explain Describe Saved SQL History

| EMPLOYEE_ID | FIRST_NAME | LAST_NAME | EMAIL | CONTACT_NUM | HIRE_DATE | DEPARTMENT_ID | SALARY |
|-------------|------------|-----------|-------------------------|-------------|-----------|---------------|--------|
| 101 | Vipul | Shah | vipulshah@gmail.com | 7895333901 | 03-AUG-11 | 50 | 30000 |
| 102 | Amit | Garg | gargamit@gmail.com | 9995463931 | 14-JAN-13 | 12 | 27000 |
| 103 | Satish | Kumar | kumar.satish@gmail.com | 9865463007 | 26-MAY-13 | 12 | 3500 |
| 104 | Harshal | - | harshal11@gmail.com | 7869463115 | 17-OCT-13 | 23 | 7300 |
| 105 | Archit | Rastogi | architrastogi@gmail.com | - | 31-AUG-15 | 50 | 2950 |

ALTER-



Home > SQL > SQL Commands

☒ Autocommit Display 10 Save Run

```
ALTER TABLE stud_demo ADD(mail varchar(15));
```

Results Explain Describe Saved SQL History

Table altered.

0.03 seconds

Home > SQL > SQL Commands

☒ Autocommit Display 10 Save Run

```
DESCRIBE stud demo;
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **STUD_DEMO**

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|-----------|---------|-----------|--------|-----------|-------|-------------|----------|---------|---------|
| STUD_DEMO | STUD_ID | Number | - | 3 | 0 | - | ✓ | - | - |
| | NAME | Varchar2 | 15 | - | - | - | ✓ | - | - |
| | ADDRESS | Varchar2 | 30 | - | - | - | ✓ | - | - |
| | MOBILE | Number | - | 10 | 0 | - | ✓ | - | - |
| | MAIL | Varchar2 | 15 | - | - | - | ✓ | - | - |

DROP-

Home > SQL > SQL Commands

☒ Autocommit Display 5000 Save Run

```
desc teacher

alter table teacher rename column dept to department

drop table teacher
```

Results Explain Describe Saved SQL History

Object Type **TABLE** Object **TEACHER**

| Table | Column | Data Type | Length | Precision | Scale | Primary Key | Nullable | Default | Comment |
|---------|-----------------|-----------|--------|-----------|-------|-------------|----------|---------|---------|
| TEACHER | T_ID | Number | - | 10 | 0 | - | ✓ | - | - |
| | T_NAME | Varchar2 | 20 | - | - | - | ✓ | - | - |
| | PHONE | Number | - | 11 | 0 | - | ✓ | - | - |
| | SALARY | Number | - | 7 | 3 | - | ✓ | - | - |
| | BLOOD_GROUP | Varchar2 | 3 | - | - | - | ✓ | - | - |
| | PUBLISHED_PAPER | Number | - | 2 | 0 | - | ✓ | - | - |
| | DESIGNATION | Varchar2 | 15 | - | - | - | ✓ | - | - |

Activate Windows
Go to Settings to activate Windows.

Home > SQL > SQL Commands

☒ Autocommit Display 5000 Save Run

```
rename teachers to teacher

desc teacher

alter table teacher rename column dept to department

drop table teacher
```

Results Explain Describe Saved SQL History

Table dropped.

0.58 seconds

PROGRAM-4

Write queries to execute following DML commands :

INSERT: Insert five records in each table.

UPDATE: Modify data in single and multiple columns in a table

DELETE: Delete selective and all records from a table

OUTPUT:

INSERT-

```
INSERT INTO emp (empno, ename, job, sal, comm, deptno)
VALUES (4160, 'STURDEVIN', 'SECURITY GUARD', 2045, NULL, 30);
```

UPDATE-

```
UPDATE EMPLOYEE
  SET JOB=NULL, SALARY=0, BONUS=0, COMM=0
  WHERE WORKDEPT = 'E21' AND JOB <> 'MANAGER'
```

DELETE-

```
DELETE FROM emp WHERE deptno = depts(j)
  RETURNING empno BULK COLLECT INTO enums;
```

PROGRAM-5

Write queries to execute following DML command :

SELECT: Retrieve the entire contents of the table.

Retrieve the selective contents (based on provided conditions) from a table.

Retrieve contents from a table based on various operators i.e. string operators, logical operators and conditional operators, Boolean operators.

Sort the data in ascending and descending order in a table on the basis of one column or more than one column

OUTPUT:

(I) To get the customer names from the customers table, you use the following statement:

```
SELECT
    name
FROM
    customers;
```

(II) To query data from multiple columns, you specify a list of comma-separated column names. **SELECT**

```
    customer_id,
    name,
    credit_limit
FROM
    customers;
```

(III) To sort the customer data by names alphabetically in ascending order, you use the following statement:

```
SELECT
    name,
    address,
    credit_limit
FROM
    customers
ORDER BY
    name ASC;
```

(IV) You can combine the AND operator with other logical operators such as OR and NOT to form a condition

```
SELECT
    order_id,
    customer_id,
```

```
    status,  
    salesman_id,  
    order_date  
FROM  
    orders  
WHERE  
    (  
        status = 'Canceled'  
        OR status = 'Pending'  
    )  
    AND customer_id = 44  
ORDER BY  
    order_date;
```

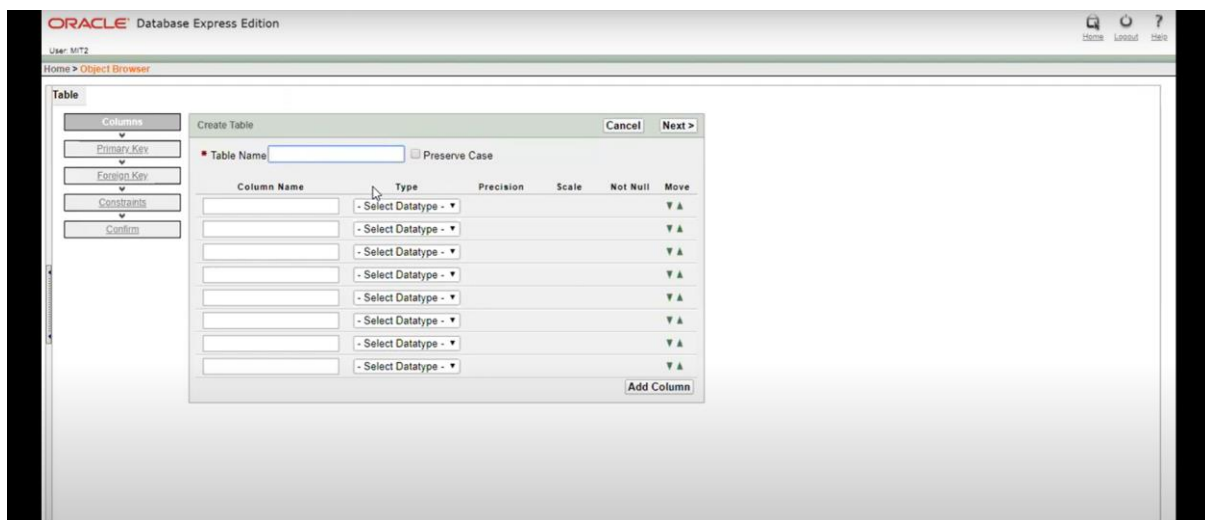

PROGRAM-6

Create table using following integrity constraints:

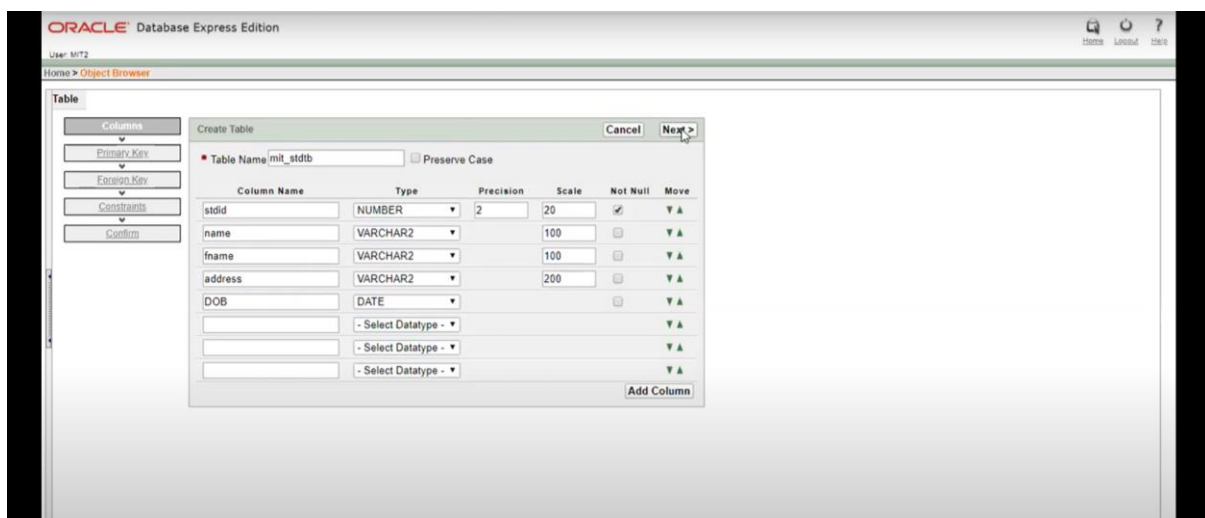
- Primary Key
- Unique Key
- Not Null
- Check Default
- Foreign Key

OUTPUT:

STEP 1:



STEP 2:



STEP 3:

The screenshot shows the Oracle Database Express Edition interface. The 'Table' tab is selected, and the 'Primary Key' sub-tab is active. The table name is 'MIT_STDTB'. The 'Primary Key' section has three radio buttons: 'No Primary Key', 'Populated from a new sequence' (selected), 'Populated from an existing sequence', and 'Not populated'. Below these, the 'Primary Key Constraint Name' is 'MIT_STDTB_PK', the 'Sequence Name' is 'MIT_STDTB_SEQ', and the 'Primary Key' is 'STOID(NUMBER)'. A 'Confirm' button is at the bottom. On the right, a 'Primary Key' help box explains that a primary key allows each row to be uniquely identified and provides instructions on how to populate it from a new or existing sequence.

STEP 4:

The screenshot shows the Oracle Database Express Edition interface. The 'Table' tab is selected, and the 'Foreign Keys' sub-tab is active. The 'Add Foreign Key' dialog is open. The 'Name' is 'MIT_STDTB_fk'. The 'References Table' is 'MIT_STDTB'. The 'Key Column(s)' is 'STOID'. The 'Select Key Column(s)' list contains 'STOID', 'NAME', 'FNAME', 'ADDRESS', and 'DOB'. The 'References Table' list contains 'MIT_STDTB'. The 'Add' button is at the bottom right. On the right, a 'Foreign Key' help box explains that a foreign key establishes a relationship between a column or columns in one table and a primary or unique key in another table, and provides instructions on how to define it.

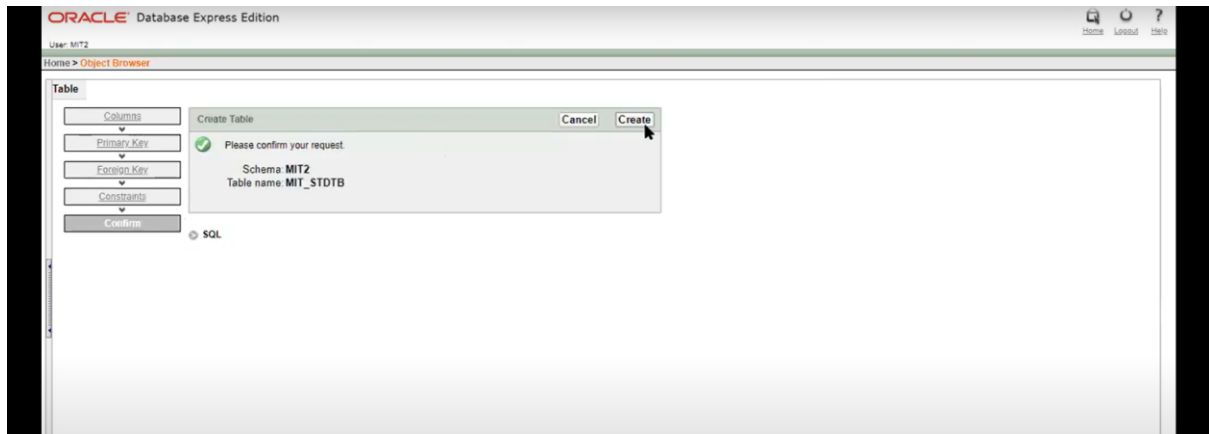
STEP 5:

The screenshot shows the Oracle Database Express Edition interface. On the left, a sidebar contains a 'Table' section with buttons for 'Columns', 'Primary Key', 'Foreign Key', 'Constraints', and 'Confirm'. The 'Constraints' button is selected. The main area displays the 'Add Constraint' dialog. At the top, there are 'Cancel', '< Previous', and 'Finish' buttons. Below them is a table with columns 'Constraint Name', 'Type', and 'Column(s)/Check'. The 'Add Constraint' section has two radio buttons: 'Check' (selected) and 'Unique'. Below the radio buttons is a text input field for the constraint name, which contains 'MIT_STDTB_ck1'. At the bottom, there are two radio buttons: 'Available Columns' (selected) and 'Example Check Constraints'. On the right side, there are three informational boxes: 'Constraints' (general), 'Check Constraint' (definition), and 'Unique Constraint' (definition).

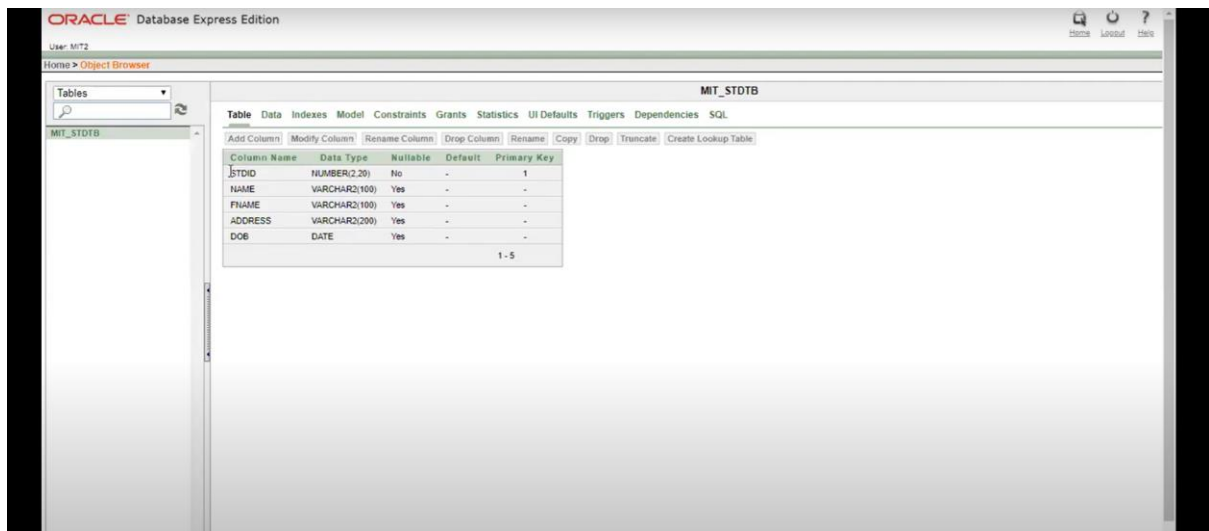
STEP 6:

The screenshot shows the Oracle Database Express Edition interface. On the left, a sidebar contains a 'Table' section with buttons for 'Columns', 'Primary Key', 'Foreign Key', 'Constraints', and 'Confirm'. The 'Constraints' button is selected. The main area displays the 'Add Constraint' dialog. At the top, there are 'Cancel', '< Previous', and 'Finish' buttons. Below them is a table with columns 'Constraint Name', 'Type', and 'Column(s)/Check'. The 'Add Constraint' section has two radio buttons: 'Check' and 'Unique' (selected). Below the radio buttons is a text input field for the constraint name, which contains 'MIT_STDTB_uk1'. Below the name field, there are two columns: 'Select Key Column(s)' and 'Key Column(s)'. The 'Select Key Column(s)' column contains a list of columns: 'STOID', 'NAME', 'FNAME', 'ADDRESS', and 'DOB'. The 'Key Column(s)' column is empty. At the bottom, there are two radio buttons: 'Available Columns' (selected) and 'Example Check Constraints'. On the right side, there are three informational boxes: 'Constraints' (general), 'Check Constraint' (definition), and 'Unique Constraint' (definition).

STEP 7:



STEP 8:



PROGRAM-7

Write queries to execute following Aggregate functions Sum,Avg,Count,Minimum and Maximum value of a numeric column of a table using aggregate function.

OUTPUT:

(I) The COUNT() function returns the number of rows that matches a specified criterion.

```
SELECT COUNT(ProductID)
FROM Products;
```

(II) The AVG() function returns the average value of a numeric column.

```
SELECT AVG(Price)
FROM Products;
```

(III) The SUM() function returns the total sum of a numeric column.

```
SELECT SUM(Quantity)
FROM OrderDetails;
```

(IV)The MIN function returns the smallest value in the specified table field.

```
SELECT MIN(`sales`) FROM `OrderDetails`;
```

(V) It returns the largest value from the specified table field.

```
SELECT MAX(`sales`) FROM `OrderDetails`;
```

PROGRAM-8

Retrieve data from a table using alias names .

OUTPUT:

SQL aliases are used to give a table, or a column in a table, a temporary name.

Aliases are often used to make column names more readable.

An alias only exists for the duration of that query.

An alias is created with the AS keyword.

Alias Column Syntax:

```
SELECT column_name AS alias_name  
FROM table_name;
```

Alias Table Syntax:

```
SELECT column_name(s)  
FROM table_name AS alias_name;
```

Example 1:

```
SELECT CustomerID AS ID, CustomerName AS Customer  
FROM Customers;
```

Example 2:

```
SELECT o.OrderID, o.OrderDate, c.CustomerName  
FROM Customers AS c, Orders AS o  
WHERE c.CustomerName='Around the Horn' AND c.CustomerID=o.CustomerID;
```

PROGRAM-9

Retrieve data from more than one table using inner join, left outer, right outer and full outer joins.

OUTPUT:

CROSS JOIN:

```
select * from employee cross join department ;|
```

Results Explain Describe Saved SQL History

| ENAME | SSN_NO. | ADDRESS | SEX | SALARY | SUPERSSN | DNO | DNO | DNAME | MGRSSN | MGRSTARTDATE |
|---------|---------|-------------|--------|--------|--------------|-----|-----|-------|--------------|--------------|
| RICHA | 1 | SECTOR 55 | FEMALE | 55000 | 201035412365 | 11 | 1 | AI | 564367879098 | 06-JAN-19 |
| RICHA | 1 | SECTOR 55 | FEMALE | 55000 | 201035412365 | 11 | 2 | EC | 324565435689 | 26-FEB-20 |
| RICHA | 1 | SECTOR 55 | FEMALE | 55000 | 201035412365 | 11 | 12 | EC | 457689089767 | 23-FEB-19 |
| RICHA | 1 | SECTOR 55 | FEMALE | 55000 | 201035412365 | 11 | 3 | CIVIL | 201035865409 | 12-DEC-19 |
| RICHA | 1 | SECTOR 55 | FEMALE | 55000 | 201035412365 | 11 | 11 | civil | 543476546789 | 04-JAN-19 |
| KASHISH | 2 | PATEL NAGAR | FEMALE | 60000 | 345470897864 | 12 | 1 | AI | 564367879098 | 06-JAN-19 |
| KASHISH | 2 | PATEL NAGAR | FEMALE | 60000 | 345470897864 | 12 | 2 | EC | 324565435689 | 26-FEB-20 |
| KASHISH | 2 | PATEL NAGAR | FEMALE | 60000 | 345470897864 | 12 | 12 | EC | 457689089767 | 23-FEB-19 |
| KASHISH | 2 | PATEL NAGAR | FEMALE | 60000 | 345470897864 | 12 | 3 | CIVIL | 201035865409 | 12-DEC-19 |
| KASHISH | 2 | PATEL NAGAR | FEMALE | 60000 | 345470897864 | 12 | 11 | civil | 543476546789 | 04-JAN-19 |

More than 10 rows available. Increase rows selector to view more rows.

FULL OUTER JOIN:

Home > SQL > SQL Commands

☒ Autocommit Display 10 Save Run

select employee.superssn , employee.salary , employee.ename, department.mgrssn , department.dname from employee full outer join department on employee.dno = department.dno;

Results Explain Describe Saved SQL History

| SUPERSSN | SALARY | ENAME | MGRSSN | DNAME |
|--------------|--------|---------|--------------|-------|
| 201035412365 | 55000 | RICHA | 543476546789 | civil |
| 345470897864 | 60000 | KASHISH | 457689089767 | EC |
| 564367879098 | 43000 | MIRAZ | - | - |
| 768907652314 | 50000 | SHIVAM | - | - |
| 34567890987 | 55000 | ADITI | - | - |
| - | - | - | 201035865409 | CIVIL |
| - | - | - | 564367879098 | AI |
| - | - | - | 324565435689 | EC |

8 rows returned in 0.00 seconds

CSV Export

INNER JOIN:

The screenshot shows the Oracle Database Express Edition interface. The user is SYSTEM. The breadcrumb navigation is Home > SQL > SQL Commands. The Autocommit checkbox is checked, and the display size is set to 10. The SQL command entered is: `select employee.superssn , employee.salary , employee.ename, department.mgrssn , department.dname from employee inner join department on employee.dno = department.dno;` The results are displayed in a table with 5 columns: SUPERSSN, SALARY, ENAME, MGRSSN, and DNAME. Two rows are returned.

Oracle Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

☒ Autocommit Display 10 Save Run

```
select employee.superssn , employee.salary , employee.ename, department.mgrssn , department.dname from employee inner join department on employee.dno = department.dno;
```

Results Explain Describe Saved SQL History

| SUPERSSN | SALARY | ENAME | MGRSSN | DNAME |
|--------------|--------|---------|--------------|-------|
| 201035412365 | 55000 | RICHA | 543476546789 | civil |
| 345470897864 | 60000 | KASHISH | 457689089767 | EC |

2 rows returned in 0.00 seconds [CSV Export](#)

LEFT OUTER JOIN:

The screenshot shows the Oracle Database Express Edition interface. The user is SYSTEM. The breadcrumb navigation is Home > SQL > SQL Commands. The Autocommit checkbox is checked, and the display size is set to 10. The SQL command entered is: `select employee.superssn , employee.salary , employee.ename, department.mgrssn , department.dname from employee left outer join department on employee.dno = department.dno;` The results are displayed in a table with 5 columns: SUPERSSN, SALARY, ENAME, MGRSSN, and DNAME. Five rows are returned, including rows where the department is null.

Oracle Database Express Edition

User: SYSTEM

Home > SQL > SQL Commands

☒ Autocommit Display 10 Save Run

```
select employee.superssn , employee.salary , employee.ename, department.mgrssn , department.dname from employee left outer join department on employee.dno = department.dno;
```

Results Explain Describe Saved SQL History

| SUPERSSN | SALARY | ENAME | MGRSSN | DNAME |
|--------------|--------|---------|--------------|-------|
| 201035412365 | 55000 | RICHA | 543476546789 | civil |
| 345470897864 | 60000 | KASHISH | 457689089767 | EC |
| 564389076512 | 43000 | MIRAZ | - | - |
| 768907652314 | 50000 | SHIVAM | - | - |
| 345676890987 | 55000 | ADITI | - | - |

5 rows returned in 0.00 seconds [CSV Export](#)

RIGHT OUTER JOIN:

ORACLE Database Express Edition

Home Logout Help

User: SYSTEM

Home > SQL > SQL Commands

☒ Autocommit Display 10 Save Run

```
select employee.superssn , employee.salary , employee.ename, department.mgrssn , department.dname from employee right outer join department
on employee.dno = department.dno;
```

Results Explain Describe Saved SQL History

| SUPERSSN | SALARY | ENAME | MGRSSN | DNAME |
|--------------|--------|---------|--------------|-------|
| 201035412365 | 55000 | RICHA | 543476546789 | civil |
| 345470897864 | 60000 | KASHISH | 457689089767 | EC |
| - | - | - | 201035865409 | CIVIL |
| - | - | - | 564367879098 | AI |
| - | - | - | 324565435689 | EC |

5 rows returned in 0.02 seconds [CSV Export](#)

PROGRAM-10

Create view from one table and more than one table

OUTPUT:

To create a view 'ordersview' by three tables 'orders', 'customer' and 'agents' with following conditions -

1. 'a' and 'b' and 'c' are the aliases of 'orders' and 'customer' and 'agents' table,
2. 'cust_code' of 'orders' and 'customer' table must be same,
3. 'agent_code' of 'orders' and 'agents' table must be same,

The following SQL statement can be used:

```
CREATE VIEW ordersview
AS SELECT ord_num, ord_amount, a.agent_code,
agent_name, cust_name
FROM orders a, customer b, agents c
WHERE a.cust_code=b.cust_code
AND a.agent_code=c.agent_code;
```

PROGRAM-11

Consider the following schema of a library management system. Write the SQL queries for the questions given below;

Student(Stud_no : integer, Stud_name: string)

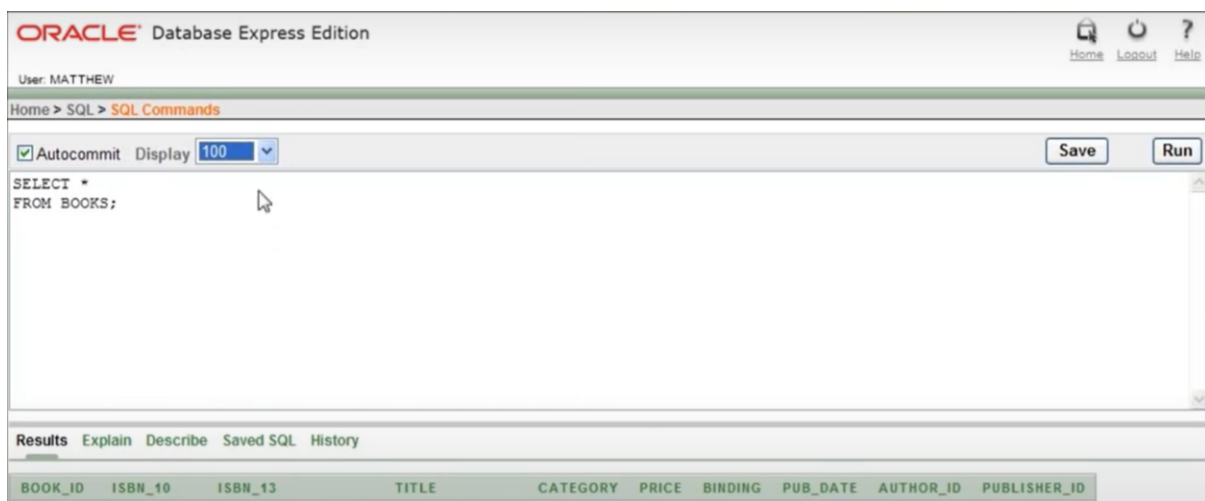
Membership(Mem_no: integer, Stud_no: integer)

Book_(book_no: integer, book_name:string, author: string)

Iss_rec_(iss_no:integer, iss_date: date, Mem_no: integer, book_no: integer)

- I. Create the tables with the appropriate integrity constraints
- II. Insert around 10 records in each of the tables
- III. Display all records for all tables
- IV. List all the student names with their membership numbers
- V. List all the issues for the current date with student and Book names
- VI. List the details of students who borrowed book whose author is Elmarsi & Navathe
- VII. Give a count of how many books have been bought by each student
- VIII. Give a list of books taken by student with stud_no as 1005
- IX. Delete the List of books details which are issued as of today
- X. Create a view which lists out the iss_no, iss _date, stud_name, book name

OUTPUT:



| BOOK_ID | ISBN_10 | ISBN_13 | TITLE | CATEGORY | PRICE | BINDING | PUB_DATE | AUTHOR_ID | PUBLISHER_ID |
|---------|------------|---------------|------------------------------------|----------|-------|---------|----------|-----------|--------------|
| 2 | 140232478 | 9780140232479 | Enemy of God: A Novel of Arthor | Fiction | 14.99 | P | 1997 | 1 | 2 |
| 6 | 7149891 | 9780007149896 | Heretic: Grail Quest | Fiction | 19.99 | P | 2004 | 1 | 1 |
| 7 | 553103547 | 9780553103540 | A Game of Thrones | Fiction | 13.99 | H | 1996 | 2 | 4 |
| 9 | 055357342X | 9780553573428 | A Storm of Swords | Fiction | 4.99 | P | 2003 | 2 | 4 |
| 11 | 553108034 | - | A Clash of Kings | Fiction | - | - | - | 2 | - |
| 1 | 140231862 | 9780140231861 | The Winter King: A Novel of Arthor | Fiction | 14.99 | P | 1996 | 1 | 2 |
| 3 | 140232877 | 9780140232875 | Excalibur: A Novel of Arthor | Fiction | 14.99 | P | 1998 | 1 | 2 |
| 4 | 6513840 | 9780006513841 | Harlequin: Grail Quest | Fiction | 19.99 | P | 2001 | 1 | 1 |
| 5 | 6513859 | 9780006513858 | Vagabond: Grail Quest | Fiction | 19.99 | P | 2003 | 1 | 1 |
| 8 | 553801473 | 9780553801477 | A Dance with Dragons | Fiction | 13.99 | H | 2010 | 2 | 5 |
| 10 | 1596590424 | 9781596590427 | 50 Success Classics | Audio | 17.99 | - | - | 3 | 3 |

11 rows returned in 0.01 seconds [CSV Export](#)

Home > SQL > SQL Commands

☒ Autocommit Display 100 Save Run

```
SELECT *
FROM PUBLISHER;
```

Results Explain Describe Saved SQL History

| PUBLISHER_ID | PUBLISHER_NAME |
|--------------|--|
| 3 | Gildan Media Corp |
| 1 | HarperCollins Publishers Canada, Limited |
| 2 | Penguin Books Canada, Limited |
| 4 | Bantam Dell Pub Group |

4 rows returned in 0.03 seconds [CSV Export](#)

Home > SQL > SQL Commands

☒ Autocommit Display 100 Save Run

```
SELECT *
FROM AUTHOR;
```

Results Explain Describe Saved SQL History

| AUTHOR_ID | AUTHOR_LAST | AUTHOR_FIRST |
|-----------|-------------|--------------|
| 1 | Cornwell | Bernard |
| 2 | Martin | George |
| 3 | Butler | Tom |

3 rows returned in 0.02 seconds [CSV Export](#)

PROGRAM-12

Define store procedure, cursor, trigger with an example.

OUTPUT:

Stored Procedures are created to perform one or more DML operations on Database. It is nothing but the group of SQL statements that accepts some input in the form of parameters and performs some task and may or may not returns a value.

Syntax: Creating a Procedure

```
CREATE or REPLACE PROCEDURE name(parameters)
```

```
IS
```

```
variables;
```

```
BEGIN
```

```
//statements;
```

```
END;
```

The most important part is parameters. Parameters are used to pass values to the Procedure. There are 3 different types of parameters, they are as follows:

IN:

This is the Default Parameter for the procedure. It always receives the values from calling program.

OUT:

This parameter always sends the values to the calling program.

IN

OUT:

This parameter performs both the operations. It Receives value from as well as sends the values to the calling program.

Example:

Imagine a table named with emp_table stored in Database. We are Writing a Procedure to update a Salary of Employee with 1000.

```
CREATE or REPLACE PROCEDURE INC_SAL(eno IN NUMBER, up_sal OUT  
NUMBER)
```

IS

BEGIN

UPDATE emp_table SET salary = salary+1000 WHERE emp_no = eno;

COMMIT;

SELECT sal INTO up_sal FROM emp_table WHERE emp_no = eno;

END;

Cursor is a Temporary Memory or Temporary Work Station. It is Allocated by Database Server at the Time of Performing DML(Data Manipulation Language) operations on Table by User. Cursors are used to store Database Tables. There are 2 types of Cursors: Implicit Cursors, and Explicit Cursors. These are explained as following below.

Implicit

Cursors:

Implicit Cursors are also known as Default Cursors of SQL SERVER. These Cursors are allocated by SQL SERVER when the user performs DML operations.

Explicit Cursors :

Explicit Cursors are Created by Users whenever the user requires them. Explicit Cursors are used for Fetching data from Table in Row-By-Row Manner.

How to create Explicit Cursor:

Declare Cursor Object.

Syntax : DECLARE cursor_name CURSOR FOR SELECT * FROM table_name

DECLARE s1 CURSOR FOR SELECT * FROM studDetails

Open Cursor Connection.

Syntax : OPEN cursor_connection

OPEN s1

Fetch Data from cursor.

There are total 6 methods to access data from cursor. They are as follows :

FIRST is used to fetch only the first row from cursor table.

LAST is used to fetch only last row from cursor table.

NEXT is used to fetch data in forward direction from cursor table.

PRIOR is used to fetch data in backward direction from cursor table.

ABSOLUTE n is used to fetch the exact nth row from cursor table.

RELATIVE n is used to fetch the data in incremental way as well as decremental way.

Syntax : FETCH NEXT/FIRST/LAST/PRIOR/ABSOLUTE n/RELATIVE n FROM
cursor_name

FETCH FIRST FROM s1

FETCH LAST FROM s1

FETCH NEXT FROM s1

FETCH PRIOR FROM s1

FETCH ABSOLUTE 7 FROM s1

FETCH RELATIVE -2 FROM s1

Close cursor connection.

Syntax : CLOSE cursor_name

CLOSE s1

Deallocate cursor memory.

Syntax : DEALLOCATE cursor_name

DEALLOCATE s1

Trigger: A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

Syntax:

create trigger [trigger_name]

[before | after]

{insert | update | delete}

on [table_name]

[for each row]

[trigger_body]

PROGRAM-13

Explain recovery and backup in database.

OUTPUT:

Database backup basically means that a duplicate of the database information and data is created and stored in backup server just to be on the safe side. Transaction logs are also stored in the backup along with the database data because without them, the data would be useless.

Reasons of Failure in a Database

There can be multiple reasons of failure in a database because of which a database backup and recovery plan is required. Some of these reasons are:

User Error - Normally, user error is the biggest reason of data destruction or corruption in a database. To rectify the error, the database needs to be restored to the point in time before the error occurred.

Hardware Failure - This can also lead to loss of data in a database. The database is stored on multiple hard drives across various locations. These hard drives may sometimes malfunction leading to database corruption. So, it is important to periodically change them.

Catastrophic Event - A catastrophic event can be a natural calamity like a flood or earthquake or deliberate sabotage such as hacking of the database. Either way, the database data may be corrupted and backup may be required.

Methods of Backup

The different methods of backup in a database are:

Full Backup - This method takes a lot of time as the full copy of the database is made including the data and the transaction records.

Transaction Log - Only the transaction logs are saved as the backup in this method. To keep the backup file as small as possible, the previous transaction log details are deleted once a new backup record is made.

Differential Backup - This is similar to full backup in that it stores both the data and the transaction records. However only that information is saved in the backup that has changed since the last full backup. Because of this, differential backup leads to smaller files.

Database Recovery

There are two methods that are primarily used for database recovery. These are:

Log based recovery - In log based recovery, logs of all database transactions are stored in a secure area so that in case of a system failure, the database can recover the data. All log information, such as the time of the transaction, its data etc. should be stored before the transaction is executed.

Shadow paging - In shadow paging, after the transaction is completed its data is automatically stored for safekeeping. So, if the system crashes in the middle of a transaction, changes made by it will not be reflected in the database.