Expt No 1

ER diagram and Relational Schema

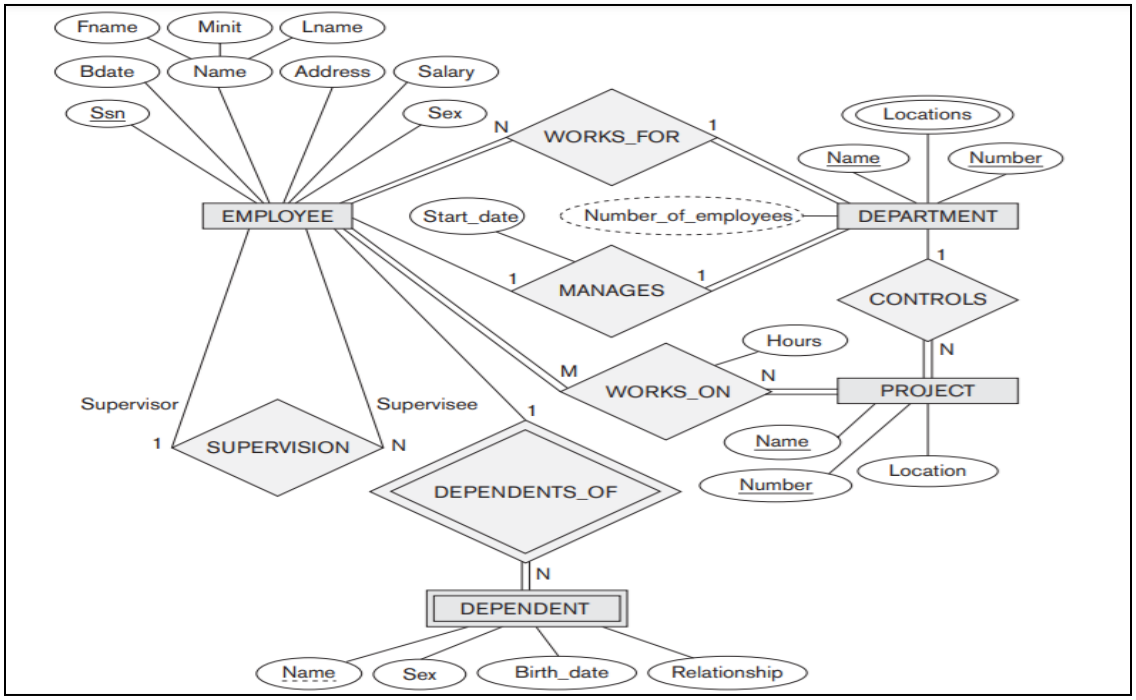
AIM:Design a database schema for an application with ER diagram from a problem description

Problem Description:

The COMPANY database keeps track of a company's employees, departments, and projects. Suppose that after the requirements collection and analysis phase, the database designers provide the following description of the miniworld—the part of the company that will be represented in the database.

- The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.
- A department controls a number of projects, each of which has a unique name, a unique number, and a single location.
- We store each employee's name, Social Security number,address, salary, sex (gender), and birth date. An employee is assigned to one department, but may work on several projects, which are not necessarily controlled by the same department. We keep track of the current number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee (who is another employee).
- We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent's first name, sex, birth date, and relationship to the employee.

ER Diagram:



Entities:

- An entity type DEPARTMENT with attributes Name, Number, Locations, Manager, and Manager_start_date. Locations is the only multivalued attribute. We can specify that both Name and Number are (separate) key attributes because each was specified to be unique.
- An entity type PROJECT with attributes Name, Number, Location, and Controlling_department. Both Name and Number are (separate) key attributes.
- An entity type EMPLOYEE with attributes Name, Ssn, Sex, Address, Salary, Birth_date, Department, and Supervisor. Both Name and Address may be composite attributes; however, this was not specified in the requirements.
- An entity type DEPENDENT with attributes Employee, Dependent_name, Sex, Birth_date, and Relationship (to the employee).

Relationship types:

- MANAGES, a 1:1 relationship type between EMPLOYEE and DEPARTMENT. EMPLOYEE participation is partial. DEPARTMENT participation is not clear from the requirements. We assume that a department must have a manager at all times, which implies total participation.The attribute Start_date is assigned to this relationship type.
- WORKS_FOR, a 1:N relationship type between DEPARTMENT and EMPLOYEE. Both participations are total.
- CONTROLS, a 1:N relationship type between DEPARTMENT and PROJECT. The participation of PROJECT is total, whereas that of DEPARTMENT is determined to be partial, assuming that some departments may control no projects.
- SUPERVISION, a 1:N relationship type between EMPLOYEE (in the supervisor role) and EMPLOYEE (in the supervisee role). Both participations are determined to be partial, assuming that not every employee is a supervisor and not every employee has a supervisor.
- WORKS_ON, determined to be an M:N relationship type with attribute Hours, after the users indicate that a project can have several employees working on it. Both participations are determined to be total. DEPENDENTS_OF, a 1:N relationship type between EMPLOYEE and DEPENDENT, which is also the identifying relationship for the weak entity type DEPENDENT. The participation of EMPLOYEE is partial, whereas that of DEPENDENT is total

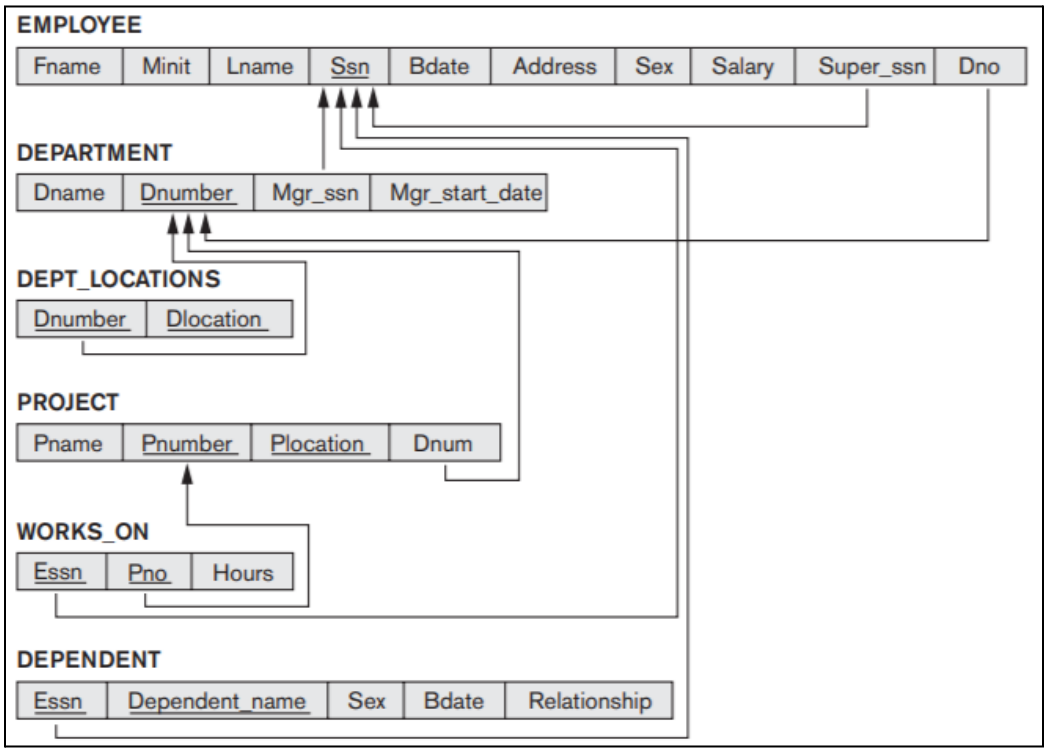
Summary :

Figure above displays the COMPANY ER database schema as an ER diagram. We now review the full ER diagram notation.

- Entity types such as EMPLOYEE, DEPARTMENT, and PROJECT are shown in rectangular boxes. Relationship types such as WORKS_FOR, MANAGES, CONTROLS, and WORKS_ON are shown in diamond-shaped boxes attached to the participating entity types with straight lines.
- Attributes are shown in ovals, and each attribute is attached by a straight line to its entity type or relationship type.
- Component attributes of a composite attribute are attached to the oval representing the composite attribute, as illustrated by the Name attribute of EMPLOYEE.
- Multivalued attributes are shown in double ovals, as illustrated by the Locations attribute of DEPARTMENT.

- Key attributes have their names underlined.
- Derived attributes are shown in dotted ovals, as illustrated by the Number_of_employees attribute of DEPARTMENT.
- Weak entity types are distinguished by being placed in double rectangles and by having their identifying relationship placed in double diamonds, as illustrated by the DEPENDENT entity type and the DEPENDENTS_OF identifying relationship type. The partial key of the weak entity type is underlined with a dotted line.
- The cardinality ratio of each binary relationship type is specified by attaching a 1, M, or N on each participating edge. The cardinality ratio of DEPARTMENT:EMPLOYEE in MANAGES is 1:1, whereas it is 1:N for DEPARTMENT: EMPLOYEE in WORKS_FOR, and M:N for WORKS_ON.
- The participation constraint is specified by a single line for partial participation and by double lines for total participation (existence dependency).
- In Figure we show the role names for the SUPERVISION relationship type because the same EMPLOYEE entity type plays two distinct roles in that relationship. Notice that the cardinality ratio is 1:N from supervisor to supervisee because each employee in the role of supervisee has at most one direct supervisor, whereas an employee in the role of supervisor can supervise zero or more employees

Relational Schema:



ER-to-Relational Schema Mapping:

Step 1: Mapping of Regular Entity Types

We create the relations EMPLOYEE, DEPARTMENT, and PROJECT in Figure to correspond to the regular entity types EMPLOYEE, DEPARTMENT, and PROJECT. We choose Ssn, Dnumber, and Pnumber as primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT, respectively

Step 2: Mapping of Weak Entity Types

We create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT. We include the primary key Ssn of the EMPLOYEE relation—which corresponds to the owner entity type—as a foreign key attribute of DEPENDENT; We rename it Essn. The primary key of the DEPENDENT relation is the combination {Essn, Dependent_name}, because Dependent_name is the partial key of DEPENDENT

Step 3: Mapping of Binary 1:1 Relationship Types.

We map the 1:1 relationship type MANAGES from Figure by choosing the participating entity type DEPARTMENT to serve in the role of S because its participation in the MANAGES relationship type is total (every department has a manager). We include the primary key of the EMPLOYEE relation as foreign key in the DEPARTMENT relation and rename it Mgr_ssn. We also include the simple attribute Start_date of the MANAGES relationship type in the DEPARTMENT relation and rename it Mgr_start_date

Step 4: Mapping of Binary 1:N Relationship Types

We now map the 1:N relationship types WORKS_FOR, CONTROLS, and SUPERVISION from Figure. For WORKS_FOR we include the primary key Dnumber of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it Dno. For SUPERVISION we include the primary key of the EMPLOYEE relation as foreign key in the EMPLOYEE relation itself—because the relationship is recursive—and call it Super_ssn. The CONTROLS relationship is mapped to the foreign key attribute Dnum of PROJECT, which references the primary key Dnumber of the DEPARTMENT relation.

Step 5: Mapping of Binary M:N Relationship Types

We map the M:N relationship type WORKS_ON from the ER diagram by creating the relation WORKS_ON in relational schema. We include the primary keys of the PROJECT and EMPLOYEE relations as foreign keys in WORKS_ON and rename them Pno and Essn, respectively. We also include an attribute Hours in WORKS_ON to represent the Hours attribute of the relationship type. The primary key of the WORKS_ON relation is the combination of the foreign key attributes {Essn, Pno}.

Step 6: Mapping of Multivalued Attributes

We create a relation DEPT_LOCATIONS. The attribute Dlocation represents the multivalued attribute LOCATIONS of DEPARTMENT, while Dnumber—as foreign key—represents the primary key of the DEPARTMENT relation. The primary key of DEPT_LOCATIONS is the combination of {Dnumber, Dlocation}. A separate tuple will exist in DEPT_LOCATIONS for each location that a department has.

RESULT:

We successfully created an ER diagram from the problem description given. We were also able to extract the relational schema from the ER diagram.

Expt No 2

APPLICATION OF DDL COMMANDS USING UI AND SQL

AIM : Creation, modification, configuration, and deletion of databases using UI and SQL Commands

QUERY

Create database students and execute various commands on it.

```
create database students;

use students;

show databases;

+--------------------+

| Database           |

+--------------------+

| information_schema |

| employees          |

| mysql              |

| students           |

| test               |

+--------------------+

use students;
```

Create a table students with the fields student id, name, email and phone number.

```
create table Student(stud_id INT AUTO_INCREMENT PRIMARY KEY,stud_fname
VARCHAR(20),stud_lname VARCHAR(20),stud_email VARCHAR(20),stud_ph
VARCHAR(10));
```

Create a table subject to store the list of subjects.

create table Subject(sub_id INT AUTO_INCREMENT PRIMARY KEY,sub_name VARCHAR(20));

Create a table marks to store marks of students for various subjects.

create table Marks(sub_id INT,stud_id INT,marks INT, PRIMARY KEY(sub_id,stud_id));

show tables;

```
+--------------------+
| Tables_in_students |
+--------------------+
| marks              |
| student            |
| subject            |
+--------------------+
```

Display the details of the tables created.

desc student;

```
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| stud_id    | int(11)     | NO   | PRI | NULL    | auto_increment |
| stud_fname | varchar(20) | YES  |     | NULL    |                |
| stud_lname | varchar(20) | YES  |     | NULL    |                |
| stud_email | varchar(20) | YES  |     | NULL    |                |
| stud_ph    | varchar(10) | YES  |     | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
```

desc subject;

```
+----------+-------------+------+-----+---------+----------------+
```

| Field | Type | Null | Key | Default | Extra |

+----------+-------------+------+-----+---------+----------------+

| sub_id | int(11) | NO | PRI | NULL | auto_increment |

| sub_name | varchar(20) | YES | | NULL | |

+----------+-------------+------+-----+---------+----------------+

desc marks;

+---------+---------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+---------+---------+------+-----+---------+-------+

| sub_id | int(11) | NO | PRI | 0 | |

| stud_id | int(11) | NO | PRI | 0 | |

| marks | int(11) | YES | | NULL | |

+---------+---------+------+-----+---------+-------+

Alter the tables to include foreign keys.

alter table marks ADD FOREIGN KEY (stud_id) REFERENCES student(stud_id);

alter table marks ADD FOREIGN KEY (sub_id) REFERENCES subject(sub_id);

Drop the tables created.

drop table student;

drop table subject;

drop table marks;

drop database students;

alter table student AUTO_INCREMENT=100;

alter table subject AUTO_INCREMENT=200;

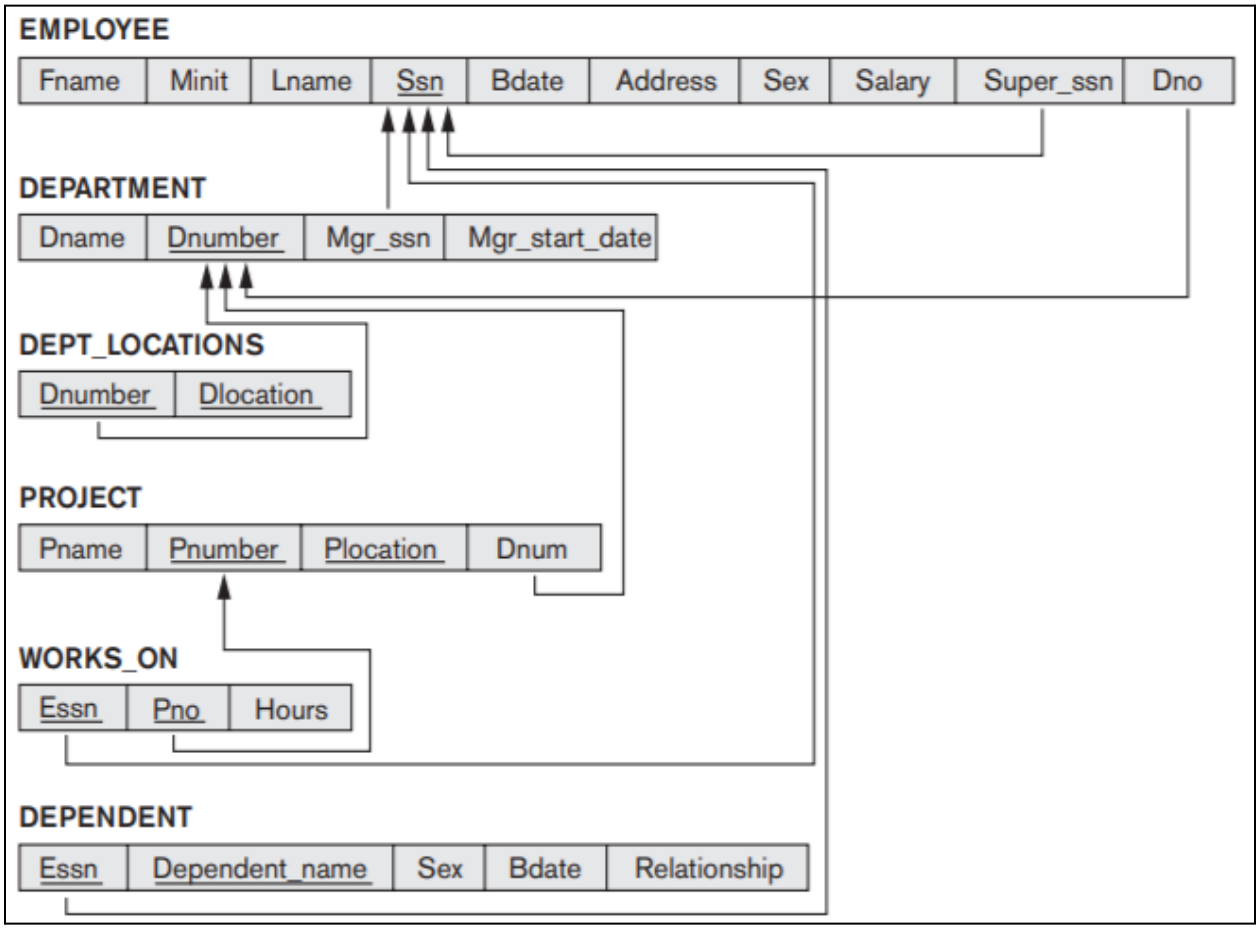
RESULT: Successfully executed the queries using MySQL Workbench.

Expt No 3

CREATION OF DATABASE SCHEMA AND EXTRACTION OF ER DIAGRAM

AIM: Creation of database schema - DDL (create tables, set constraints, enforce relationships, create indices, delete and modify tables). Export ER diagram from the database and verify relationships (with the ER diagram designed in step 1).

Create a database schema for the below diagram.



QUERY:

EER diagram: <SCREENSHOT>

RESULT:

We successfully created the Database Schema from the problem description given. We were also able to export the ER diagram from the database and relationships are verified.

Expt No 4

APPLICATION OF DML COMMANDS USING SQL

AIM :Insertion, updation, deletion, and selection of databases using SQL Commands

Insert data into the students table.

```
insert into student(stud_fname,stud_lname,stud_email,stud_ph)
values('shanti','vasan','shantiv@gmail.com',9677483824);
```

```
mysql> insert into student(stud_fname,stud_lname,stud_email,stud_ph)
values('anjitha','k','anjithak@gmail.com',9574884993);
```

```
mysql> insert into student(stud_fname,stud_lname,stud_email,stud_ph)
values('riya','khan','riyakhan@gmail.com',9637833993);
```

Insert data into the subject table.

```
insert into subject(sub_name) values('chemistry');
```

```
insert into subject(sub_name) values('physics');
```

```
insert into subject(sub_name) values('maths');
```

Insert data into marks table.

```
insert into marks(sub_id,stud_id,marks) values(200,100,75);
```

```
insert into marks(sub_id,stud_id,marks) values(200,101,94);
```

```
insert into marks(sub_id,stud_id,marks) values(200,102,60);
```

```
insert into marks(sub_id,stud_id,marks) values(201,100,85);
```

```
insert into marks(sub_id,stud_id,marks) values(201,101,98);
```

```
insert into marks(sub_id,stud_id,marks) values(201,102,70);
```

```
insert into marks(sub_id,stud_id,marks) values(202,100,50);
```

```
insert into marks(sub_id,stud_id,marks) values(202,101,96);
```

```
insert into marks(sub_id,stud_id,marks) values(202,102,45);
```

Update the lname to ‘rajan’ of the student having student id 100.

update student set stud_lname='rajan' where stud_id=100;

Update the subject name to ‘mathematics’ where subject id is 202.

update subject set sub_name='mathematics' where sub_id=202;

Update the marks of subject 200 and the student having student id 102 to 83.

update marks set marks=83 where sub_id=200 AND stud_id=102;

delete from student where stud_id=102;

SELECT QUERIES:

select * from student;

```
+---------+------------+------------+--------------------+------------+
| stud_id | stud_fname | stud_lname | stud_email         | stud_ph    |
+---------+------------+------------+--------------------+------------+
|     100 | shanti     | rajan      | shantiv@gmail.com  | 9677483824 |
|     101 | anjitha    | k          | anjithak@gmail.com | 9574884993 |
|     102 | riya       | khan       | riyakhan@gmail.com | 9637833993 |
+---------+------------+------------+--------------------+------------+
```

select * from subject;

```
+--------+-------------+
| sub_id | sub_name    |
+--------+-------------+
|    200 | chemistry   |
|    201 | physics     |
|    202 | mathematics |
+--------+-------------+
```

select * from marks;

+--------+---------+-------+

| sub_id | stud_id | marks |

+--------+---------+-------+

| 200 | 100 | 75 |

| 200 | 101 | 94 |

| 200 | 102 | 83 |

| 201 | 100 | 85 |

| 201 | 101 | 98 |

| 201 | 102 | 70 |

| 202 | 100 | 50 |

| 202 | 101 | 96 |

| 202 | 102 | 45 |

+--------+---------+-------+

1.Write a query to get the details of a student whoseSTUD_ID is102.

select *from student where stud_id=102;

+---------+------------+------------+--------------------+------------+

| stud_id | stud_fname | stud_lname | stud_email | stud_ph |

+---------+------------+------------+--------------------+------------+

| 102 | riya | khan | riyakhan@gmail.com | 9637833993 |

+---------+------------+------------+--------------------+------------+

2.Write a query to display student name, subject and marks of student ordered by marks.

select stud_fname,sub_name,marks from student,subject,marks where student

.stud_id=marks.stud_id and subject.sub_id=marks.sub_id order by marks;

+------------+-------------+-------+

| stud_fname | sub_name | marks |

+------------+-------------+-------+

| riya | mathematics | 45 |

| shanti | mathematics | 50 |

| riya | physics | 70 |

| shanti | chemistry | 75 |

| riya | chemistry | 83 |

| shanti | physics | 85 |

| anjitha | chemistry | 94 |

| anjitha | mathematics | 96 |

| anjitha | physics | 98 |

+------------+-------------+-------+

3.Write a query to display student name, subject and marks of students who have marks greater than 70 in physics.

select stud_fname,sub_name,marks from student,subject,marks where student

.stud_id=marks.stud_id and subject.sub_id=marks.sub_id and marks>70 and sub_name

='physics';

+------------+----------+-------+

| stud_fname | sub_name | marks |

+------------+----------+-------+

| shanti | physics | 85 |

| anjitha | physics | 98 |

+------------+----------+-------+

4.Write a query to display average marks in each subject.

select sub_name,avg(marks)from student,subject,marks where student.stud_i

d=marks.stud_id and subject.sub_id=marks.sub_id group by sub_name;

```
+-------------+------------+
| sub_name    | avg(marks) |
+-------------+------------+
| chemistry   |    84.0000 |
| mathematics |    63.6667 |
| physics     |    84.3333 |
+-------------+------------+
```

5.Write a query to display the number of students.

select count(stud_id) from student;

```
+----------------+
| count(stud_id) |
+----------------+
|              3 |
+----------------+
```

6.Write a query to display the maximum and minimum marks obtained by students in each subject.

select sub_name,max(marks),min(marks) from student,subject,marks where student.stud_id=marks.stud_id and subject.sub_id=marks.sub_id group by sub_name;

```
+-------------+------------+------------+
| sub_name    | max(marks) | min(marks) |
+-------------+------------+------------+
```

| chemistry | 94 | 75 |

| mathematics | 96 | 45 |

| physics | 98 | 70 |

+-------------+------------+------------+

7.Write a query to display the details of a student whose name begins with S.

select * from student where stud_fname like 's%';

+---------+------------+------------+------------------+------------+

| stud_id | stud_fname | stud_lname | stud_email | stud_ph |

+---------+------------+------------+------------------+------------+

| 100 | shanti | rajan | shantiv@gmail.com | 9677483824 |

+---------+------------+------------+------------------+------------+

8.Write a query to display the details of a student whose first name contains a in the fourth place.

select * from student where stud_fname like '___a';

+---------+------------+------------+--------------------+------------+

| stud_id | stud_fname | stud_lname | stud_email | stud_ph |

+---------+------------+------------+--------------------+------------+

| 102 | riya | khan | riyakhan@gmail.com | 9637833993 |

+---------+------------+------------+--------------------+------------+

9.Write a query to display the name,subject and marks of students having marks between 50 and 75.

select stud_fname,sub_name,marks from student,subject,marks where student.stud_id=marks.stud_id and subject.sub_id=marks.sub_id and marks between 50 and75;

+------------+-------------+-------+

| stud_fname | sub_name | marks |

+------------+-------------+-------+

| shanti | chemistry | 75 |

| riya | physics | 70 |

| shanti | mathematics | 50 |

+------------+-------------+-------+

10. Create a view to display student name and marks

CREATE VIEW stud1 AS SELECT stud_fname,stud_lname,sub_name,marks from stu

dent,subject,marks where student.stud_id=marks.stud_id and subject.sub_id=marks.

sub_id;

select * from stud1;

+------------+------------+-------------+-------+

| stud_fname | stud_lname | sub_name | marks |

+------------+------------+-------------+-------+

| shanti | rajan | chemistry | 75 |

| anjitha | k | chemistry | 94 |

| riya | khan | chemistry | 83 |

| shanti | rajan | physics | 85 |

| anjitha | k | physics | 98 |

| riya | khan | physics | 70 |

| shanti | rajan | mathematics | 50 |

| anjitha | k | mathematics | 96 |

| riya | khan | mathematics | 45 |

+------------+------------+-------------+-------+

RESULT: Successfully executed the queries using SQL DML Commands.

Expt No 5

IMPLEMENTATION OF BUILT IN FUNCTIONS

AIM: Implementation of built in functions in RDBMS

A. Create a table store. Fields are order no, code, item, quantity, price, discount, mrp

QUERY

```
Create table store (order_no int primary key, code int, item char(15), quantity varchar(8), price int,
discount varchar(7), mrp int);
Insert into store values(‘1’, ‘1’, ‘soap’, ‘5’, ‘75’, ‘2%’, ‘72’,);
1 row created;
Insert into store values(‘2’, ‘2’, ‘chilly powder’, ‘2’, ‘24’, ‘3%’, ‘20’,);
1 row created;
Insert into store values(‘3’, ‘3’, ‘atta’, ‘2’, ‘70’, ‘3%’, ‘78’,);
1 row created;
Insert into store values(‘4’, ‘4’, ‘pepper’, ‘5’, ‘524’, ‘5%’, ‘520’,);
1 row created;
Insert into store values(‘5’, ‘5’, ‘salt’, ‘4’, ‘40’, ‘2%’, ‘39’,);
1 row created;
```

B. Display the table;

QUERY

```
Select * from store;
```

OUTPUT

Oder_no	code	item	quantity	price	discount	mrp
1	1	soap	5	75	2%	72
2	2	chilly powder	2	24	3%	20
3	3	atta	2	70	3%	78
4	4	pepper	5	524	5%	520
5	5	salt	4	40	2%	39

c. Write an SQL query to display the reminder, if the amount of an each item in store is

divided by 9.

QUERY
Select item, mod(mrp,9) from store;

OUTPUT

Item	mod(mrp)
Soap	0
chilly powder	2
atta	6
pepper	7
salt	3

d. Write SQL query to display the amount in store and its square.

QUERY
Select price, power(price,2) as power from store;

OUTPUT

Price	Power
75	5625
24	576
70	49000
524	274576
40	1600

e. Program to divide the amount in stock of each item by 7 in store table and display the resut round to the nearest integer.

QUERY
Select price, round(price/7,0) as round from store;

OUTPUT

Price	Round
75	11
24	3
70	10
524	75
40	6

RESULT: Successfully executed the queries using SQL DML Commands.

EXPT NO:-6

AGGREGATE FUNCTIONS

AIM:-Implementation of various aggregate functions in SQL

1)**Find the sum of the salaries of all employees, the maximum salary,the minimum salary, and the average salary.**

```
SELECT SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)
FROM EMPLOYEE;
```

2)**Find the sum of the salaries of all employees of the 'Research'department, as well as the maximum salary, the minimum salary, and the average salary in this department.**

```
SELECT SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)
FROM (EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber)
WHERE Dname='Research';
```

3)**Retrieve the total number of employees in the company and the number of employees in the 'Research' department**

```
SELECT COUNT (*)FROM EMPLOYEE;
SELECT COUNT (*)FROM EMPLOYEE, DEPARTMENT WHERE DNO=DNUMBER AND
DNAME='Research';
```

4)**Count the number of distinct salary values in the database.**

```
SELECT COUNT (DISTINCT Salary) FROM EMPLOYEE;
```

RESULT: Successfully executed the queries using SQL DML Commands.

EXPT NO:-7

ORDER BY,GROUP BY AND HAVING CLAUSE

AIM:-Implementation of order by,group by and having clause

1)**For each department, retrieve the department number, the number of employees in the department, and their average salary.**

```
SELECT Dno, COUNT (*), AVG (Salary) FROM EMPLOYEE GROUP BY Dno;
```

2)**For each project, retrieve the project number, the project name, and the number of employees who work on that project.**

```
SELECT Pnumber, Pname, COUNT (*)
FROM PROJECT, WORKS_ON
WHERE Pnumber=Pno
GROUP BY Pnumber, Pname;
```

3)**For each project on which more than two employees work, retrieve the project number, the project name, and the number of employees who work on the project.**

```
SELECT Pnumber, Pname, COUNT (*)
FROM PROJECT, WORKS_ON
WHERE Pnumber=Pno
GROUP BY Pnumber, Pname
HAVING COUNT (*) > 2;
```

4)**For each project, retrieve the project number, the project name, and the number of employees from department 5 who work on the project.**

```
SELECT Pnumber, Pname, COUNT (*)
FROM PROJECT, WORKS_ON, EMPLOYEE
WHERE Pnumber=Pno AND Ssn=Essn AND Dno=5
GROUP BY Pnumber, Pname;
```

5)**For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than $40,000.**

```
SELECT Dnumber, COUNT (*)
FROM DEPARTMENT, EMPLOYEE
WHERE Dnumber=Dno AND Salary>40000 AND
( SELECT Dno
FROM EMPLOYEE
GROUP BY Dno
HAVING COUNT (*) > 5)
```

RESULT: Successfully executed the queries using SQL DML Commands.

ExpNo:8

NESTED QUERIES , JOIN QUERIES AND SET OPERATORS

AIM: To perform nested Queries , joining Queries and set operations using DML command

QUERIES

1. Display all employee names and salary whose salary is greater than minimum salary of the company

SQL>select Fname,Lname,Salary from Employee where Salary>(select min(Salary) from Employee);

2. Issue a query to display information about employees who earn more than any employee in dept no 5

SQL> select * from Employee where Salary>(select max(Salary) from Employee where Dno=5);

3. Display the details of those who draw the salary greater than the average salary.

SQL> select distinct * from Employee x where x.Salary >= (select avg(Salary) from Employee);

4. Write SQL Query which retrieves the name and address of every employee who works for the Research Department

Select Fname,Lname,Address from(Employee JOIN DEPARTMENT ON Dno=Dnumber) where Dname='Research'

5 Retrieve the name of each employee who has a dependent with the same first name and is the same sex as the employee.

SQL>Select E.Fname, E.Lname From Employee as E where E.Ssn in (Select Essn From Dependent as D where E.Fname=D.Dependent_Name and E.Sex=D.Sex);

6. Make a list of all project numbers for projects that involve an employee whose last name is 'Smith', either as a worker or as a manager of the department that controls the project.

SQL> (Select distinct Pnumber from Project, Department, Employee where Dnum=Dnumber and Mgr_Ssn=Ssn And Lname='Smith') union (Select Distinct Pnumber From Project, Works_On, Employee Where Pnumber=Pno and Essn=Ssn And Lname='Smith');

7. Write a query to display the name for all employees who work in a department with any employee whose Fname contains h

Select Fname from Employee where Dno IN (Select Dno from Employee where Fname LIKE '%h%');

8. Write a SQL query to find those employees who work in the same department where 'Ramesh' works. Exclude all those records where first name is 'Ramesh'. Return first name, last name

SELECT first_name, last_name FROM employees WHERE DNO = (SELECT

DNO FROM employees WHERE first_name = 'Ramesh') AND first_name <>

'Ramesh';

9 Display all the dept numbers available in emp and not in dept tables and vice versa.

SQL> Select Dno from Employee minus select Dnumber from

Department; SQL> Select Dnumber from Department minus select Dno

from Employee;

RESULT

The query was executed and output was successfully obtained.

Exp No-9
TCL COMMANDS

AIM:Implementation of SQL TCL commands Rollback,Commit,Savepoint.

a) Create a table Student having field id,name,mark

QUERY

Create table Student(id int primary key,name varchar(20),mark int);

Insert into Student values(1,'Anu',45);

1 row created;

Insert into Student values(2,'Manu',43);

1 row created;

Insert into Student values(3,'Veena',47);

1 row created;

Insert into Student values(4,'Seena',40);

1 row created;

B. Display the table;

QUERY

Select * from store;

OUTPUT

id	name	mark
1	Anu	45
2	Manu	43

3	Veena	47
4	Seena	40

C. Implement Savepoint

QUERY

Update Student mark=49 where id=4;

Savepoint a;

//Display table

Select * from Student ;

id	name	mark
1	Anu	45
2	Manu	43
3	Veena	47
4	Seena	49

Delete from Student where id=2;

Insert into Student values(2,'Neethu',42);

Savepoint b;

//Display table

Select * from Student ;

id	name	mark
1	Anu	45
2	Neethu	42
3	Veena	47
4	Seena	49

D.Implementation of Rollback

QUERY

Rollback to a;

//Display table

Select * from Student ;

id	name	mark
1	Anu	45
2	Manu	43
3	Veena	47
4	Seena	49

E. Implementation of Commit

//Display table

Select * from Student ;

id	name	mark
1	Anu	45
2	Manu	43

3	Veena	47
4	Seena	49

QUERY

Commit;

Rollback to b;

OUTPUT

Savepoint b does not exist

//commit command saves the data permanently to the database.All the savepoints are deleted and cannot rollback the same.

RESULT

The query was executed and output was successfully obtained.

Exp No-10
DCL COMMANDS FOR GRANT AND REVOKE

AIM:Implementation of DCL commands Grant and Revoke

QUERY

Create Database

```
mysql> create database db;
Query OK, 1 row affected (0.03 sec)
```

Use Database

```
mysql> use db;
Database changed
```

Create table student

```
mysql> create table student(student_id INT , name varchar (10), age INT);
Query OK, 0 rows affected (0.24 sec)
```

Select Current User

```
mysql> select current_user;
+----------------+
| current_user   |
+----------------+
| root@localhost |
+----------------+
1 row in set (0.00 sec)
```

Granting Permissions to student table

```
mysql> grant insert,delete on student to root@localhost;
Query OK, 0 rows affected (0.02 sec)
```

View Permissions on the table

```
mysql> show grants for root@localhost;
+-----------------------------------------------------------------------+
| Grants for root@localhost                                             |
```

```
+---------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION |
| GRANT DELETE ON `emp`.`employee` TO 'root'@'localhost'              |
| GRANT INSERT, DELETE ON `db`.`student` TO 'root'@'localhost'        |
| GRANT PROXY ON "@" TO 'root'@'localhost' WITH GRANT OPTION          |
+---------------------------------------------------------------------+
4 rows in set (0.00 sec)
```

Revoking Permissions

```
mysql> revoke insert on student from root@localhost;
Query OK, 0 rows affected (0.00 sec)

mysql> show grants for root@localhost;
+---------------------------------------------------------------------+
| Grants for root@localhost                                           |
+---------------------------------------------------------------------+
| GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT OPTION |
| GRANT DELETE ON `emp`.`employee` TO 'root'@'localhost'              |
| GRANT DELETE ON `db`.`student` TO 'root'@'localhost'                |
| GRANT PROXY ON "@" TO 'root'@'localhost' WITH GRANT OPTION          |
+---------------------------------------------------------------------+
4 rows in set (0.00 sec)
```

RESULT

The query was executed and output was successfully obtained.

Exp No-11

Views in SQL

AIM: Demonstrate View in SQL

a. Create a table store. Fields are order no, code, item, quantity, price, discount, mrp

QUERY

Create table store (order_no int primary key, code int, item char(15), quantity varchar(8), price int, discount varchar(7), mrp int);

Insert into store values(‘1’, ‘1’, ‘soap’, ‘5’, ‘75’, ‘2%’, ‘72’,);

1 row created;

Insert into store values(‘2’, ‘2’, ‘chilly powder’, ‘2’, ‘24’, ‘3%’, ‘20’,);

1 row created;

Insert into store values(‘3’, ‘3’, ‘atta’, ‘2’, ‘70’, ‘3%’, ‘78’,);

1 row created;

Insert into store values(‘4’, ‘4’, ‘pepper’, ‘5’, ‘524’, ‘5%’, ‘520’,);

1 row created;

Insert into store values(‘5’, ‘5’, ‘salt’, ‘4’, ‘40’, ‘2%’, ‘39’,);

1 row created;

b. Display the table;

QUERY

Select * from store;

OUTPUT

Oder_no	code	item	quantity	price	discount	mrp
1	1	soap	5	75	2%	72
2	2	chilly powder	2	24	3%	20
3	3	atta	2	70	3%	78
4	4	pepper	5	524	5%	520
5	5	salt	4	40	2%	39

c. Create a view with item_name and quantity for the above table

SYNTAX

Create view name as select field_name1, field name2,..... from table_name;

QUERY

Create view v as select item, quantity from store;

View created;

Select * from v;

OUTPUT

Item	quantity
Soap	5

Chilli powder	2
Atta	2
Pepper	5
Salt	4

d. Write a query to check whether any changes made to present table is reflected in view.

QUERY

```
Update store set quantity='50kg' where item='atta';
Select * from v;
```

OUTPUT

Item	quantity
Soap	5
Chilli powder	2
Atta	50kg
Pepper	5
Salt	4

e. Write a query to delete a view

QUERY

```
Drop view v;
View dropped.
```

RESULT

The query was executed and output was successfully obtained.

ExpNo:12

Implementation of various control structures like IF-THEN,IF-THEN-ELSE,IF-THENELSEIF,CASE ,WHILE USING PL/SQL

AIM: To implement various various control structures like IF-THEN,IF-THEN-ELSE,IF-THEN ELSIF,CASE ,WHILE USING PL/SQL

a)Write a plsql program to check whether a given number is ODD or EVEN

PROGRAM

```
1  declare
2  n number:=&n;
3   begin
4  if mod(n,2)=0
5  then
6  dbms_output.put_line('number is even');
7  else
8  dbms_output.put_line('number is odd');
9  end if;
10 end;
11 /
```

INPUT

```
Enter value for n: 6

old 7: n:=&n;

new 7: n:=6;
```

OUTPUT

Number is even

PL/SQL procedure successfully completed.

b) Write a PL/SQL block to find the maximum number from given three numbers.

PROGRAM:

```
declare
a number;
b number;
c number;
 begin
a:=&a;
b:=&b;
c:=&c;
if (a>b and a>c) then
dbms_output.put_line('a is maximum ' || a);
elsif (b>a and b>c) then
 dbms_output.put_line('b is maximum ' || b);
else
dbms_output.put_line('c is maximum ' || c);
end if;
end;
 /
```

INPUT

Enter value for a: 6

old 7: a:=&a;

new 7: a:=6;

Enter value for b: 9

old 8: b:=&b;

new 8: b:=6;

Enter value for c: 3

old 9: c:=&c;

new 9: c:=6;

OUTPUT:

B is maximum

PL/SQL procedure successfully completed.

C. Write a program to accept a number and find the sum of the digits

PROGRAM

```
declare
n number(5):=&n;
s number:=0;
 r number(2):=0;
begin
while n !=0
```

```
loop
r:=mod(n,10);
s:=s+r;
n:=trunc(n/10);
end loop;
dbms_output.put_line('sum of digits of given number is '||s);
end;
/
```

INPUT

```
Enter value for n: 146
old 7: n:=&n;
new 7: n:=146;
```

OUTPUT

```
Sum of digits of given number is 11
PL/SQL procedure successfully completed.
```

d)PL/SQL Program to accept a number from user and print number in reverse order.

PROGRAM

```
declare
num1 number(5);
num2 number(5);
rev number(5);
begin
```

```
num1:=&num1;

rev:=0;

while num1>0

loop

num2:=num1 mod 10;

rev:=num2+(rev*10);

num1:=floor(num1/10);

end loop;

dbms_output.put_line('Reverse number is: '||rev);

end;

 /
```

INPUT

```
Enter value for num1: 146

old 7: num1:=&num1;

new 7: num1:=146;
```

OUTPUT

```
Reverse number is 641

PL/SQL procedure successfully completed.
```

e)Program to print the days names in the week.

PROGRAM

```
declare
d number:=&num1;
begin
```

```
case d
  when 1 then
    dbms_output.put_line('sunday');
  when 2 then
    dbms_output.put_line('monday');
  when 3 then
    dbms_output.put_line('tuesday');
  when 4 then
    dbms_output.put_line('wednesday');
  when 5 then
    dbms_output.put_line('thursday');
  when 6 then
    dbms_output.put_line('friday');
  when 7 then
    dbms_output.put_line('saturday');
  else
    dbms_output.put_line('invalid day');
end case;

End;
```

INPUT

Enter value for num1: 5

old 7: num1:=&num1;

new 7: num1:=5;

OUTPUT

Thursday

PL/SQL procedure successfully completed.

RESULT

The plsql program was executed and output was successfully obtained.

Exp No-13

Creation of Procedures, Triggers and Functions

AIM: To implement programs using procedures.

Just as you can in other languages, you can create your own procedures in Oracle.

Syntax

The syntax to create a procedure in Oracle is:

CREATE [OR REPLACE] PROCEDURE procedure_name [

(parameter [,parameter])]

IS

[declaration_section]

BEGIN

executable_section

[EXCEPTION

exception_section]

END [procedure_name];

There are three types of parameters that can be declared:

1. **IN** - The parameter can be referenced by the procedure or function. The value of the parameter can not be overwritten by the procedure or function.
2. **OUT** - The parameter can not be referenced by the procedure or function, but the value of the parameter can be overwritten by the procedure or function.
3. **IN OUT** - The parameter can be referenced by the procedure or function and the value of the parameter can be overwritten by the procedure or function.

Drop Procedure

Once you have created your procedure in Oracle, you might find that you need to remove it from the database.

Syntax

The syntax to a drop a procedure in Oracle is:
DROP PROCEDURE procedure_name;

procedure_name -The name of the procedure that you wish to drop.

QUESTION

1) write a PL/SQL Procedure to find largest of two Numbers

2) create a table emplyee(empid,empname,salary,dept,wef)
write a procedure to accept two arguments empid and salary increment(in %).update the employee table with the salary increment also record the effective date.

PROGRAM

```
1. create procedure largest (a1 in number,b1 in number,c1 out number) as
begin
if (a1>b1) then
c1:=a1;
else
c1:=b1;
end if;
end largest;

set serveroutput on;
declare
a2 number:=&a2;
b2 number:=&b2;
c2 number:=0;
begin
largest(a2,b2,c2);
dbms_output.put_line('Largest: '||c2);
end;
```

OUTPUT

anonymous block completed

Enter the value of a2 : 12

Enter the value of b2 : 25

Largest: 25

2. create table employ(eid number(5),nam varchar(6),sal number(6),dep varchar(8),wef date);

```
insert into employ values(1,'jose',1000,'finance','03-05-16');
insert into employ values(2,'vimal',2000,'hr','11-06-16'); insert

into employ values(3,'yadhu',3000,'sales','17-08-16'); insert into

employ values(4,'robin',4000,'finance','09-01-16'); insert into

employ values(5,'roshin',5000,'hr','07-04-16');


create procedure upda(a1 in out number,b1 in out number) as

d date;

begin

select sysdate into d from DUAL;

update employ set sal=sal+((sal*b1)/100) where eid=a1;

update employ set wef=d where eid=a1;

end upda;


set serveroutput on;

declare

 a2 number:=&a2;

 a3 number:=&a3;

begin

 upda(a2,a3);

end;
```

OUTPUT

select * from employ;

	EID	NAM	SAL	DEP	WEF
1	1	jose	1000	finance	03-05-16
2	2	vimal	2000	hr	11-06-16
3	3	yadhu	3000	sales	17-08-16
4	4	robin	4000	finance	09-01-16
5	5	roshin	5000	hr	07-04-16

anonymous block completed

	EID	NAM	SAL	DEP	WEF
1	1	jose	1000	finance	03-05-16
2	2	vimal	2400	hr	29-11-17
3	3	yadhu	3000	sales	17-08-16
4	4	robin	4000	finance	09-01-16
5	5	roshin	5000	hr	07-04-16

AIM:To implement programs using triggers

TRIGGER

QUESTIONS

1)create a trigger which shows the salary difference of a particular employee whenever salary is getting updated

Worker(id,name,salary)

PROGRAM

create table worker(id number(5),name varchar(10),sal number(10));

insert into worker values(1,'riya',25000);

insert into worker values(2,'febi',30000);

insert into worker values(3,'varsha',20000);

insert into worker values(4,'anjali',50000);

```
insert into worker values(5,'athira',35000);

create trigger t after update of sal on worker for each row
declare
 diff number(10);
begin
 diff:=:new.sal-:old.sal;
 dbms_output.put_line('Difference in salary is '||diff);
 end;

 set serveroutput on;
 declare
 n number(3):=&workerid;
 nwsal number(5):=&nwsal;
 begin
 update worker set sal=nwsal where id=n;
 end;
```

OUTPUT

select * from worker;

	ID	NAME	SAL
1	1	riya	25000
2	2	febi	30000
3	3	varsha	20000
4	4	anjali	50000
5	5	athira	35000

TRIGGER T compiled

anonymous block completed

Enter the worker id =1

Enter the new salary = 28000

anonymous block completed

Difference in salary is 3000

select * from worker;

		NAME	SAL
1	1	riya	28000
2	2	febi	30000
3	3	varsha	20000
4	4	anjali	50000
5	5	athira	35000

2) create a table theater (movie id, movie name,language,review_ratings).Whenever rating goes below 5 , the movies has to be removed from theater table and add to table outdated movies with attributes movie id and movie name

PROGRAM

```
create table theatre(mov_id number(10),mov_name varchar(20),lang varchar(20),revw
number(10));

insert into theatre values(101,'Junglebook','English',6);
insert into theatre values(501,'Parava','Malayalam',8);
insert into theatre values(601,'OSO','Hindi',9);
insert into theatre values(701,'Avengers','English',9);
insert into theatre values(801,'Hobbit','English',7);
insert into theatre values(901,'Don','Hindi',6);

create table outdat(mov_id number(10),mov_name varchar(20));

create trigger trii
after delete on theatre
for each row
begin
  insert into outdat values(:old.mov_id,:old.mov_name);
end;

set serveroutput on;
declare
a1 number:=&movie_id;
a2 number:=&new_rating;
begin
  if(a2<5) then
  delete from theatre where mov_id=a1;
  end if;
```

end;

OUTPUT

select * from theatre;

	MOV_ID	MOV_NAME	LANG	REVW
1	101	Junglebook	English	6
2	501	Parava	Malayalam	8
3	601	OSO	Hindi	9
4	701	Avengers	English	9
5	801	Hobbit	English	7
6	901	Don	Hindi	6

anonymous block completed

Enter the movie_id : 801
Enter the new_rating : 4

select * from theatre;

	MOV_ID	MOV_NAME	LANG	REVW
1	101	Junglebook	English	6
2	501	Parava	Malayalam	8
3	601	OSO	Hindi	9
4	701	Avengers	English	9
5	901	Don	Hindi	6

select * from outdat;

	MOV_ID	MOV_NAME
1	801	Hobbit

Aim: To implement programs using functions.

A standalone function is created using the **CREATE FUNCTION** statement. The syntax is given by

CREATE [OR REPLACE] FUNCTION function_name ((parameter_name {IN} type {, ...}))

RETURN return_datatype

{IS | AS}

<declaration section>

BEGIN

< function_body >

END;

Where,

- *function-name* specifies the name of the function.
- [OR REPLACE] option allows the modification of an existing function.
- The optional parameter list contains name, mode and types of the parameters. IN represents the value that will be passed from outside The function must contain a **return** statement.
- The *RETURN* clause specifies the data type you are going to return from the function. ·

Function-body contains the executable part.

- The AS keyword is used instead of the IS keyword for creating a standalone function.

While creating a function, you give a definition of what the function has to do. To use a function, you will have to call that function to perform the defined task. When a program calls a function, the program control is transferred to the called function.

To call a function, you simply need to pass the required parameters along with the function name and if the function returns a value, then you can store the returned value.

QUESTIONS

1)write a PL/SQL function to find factorial of a Number.
2)write a PL/SQL function to find sum of 1st N even Numbers (additional question)
3)sales of different products in one week is recorded

Product (productid,productname,grade)
Sales (prdctname,salesamount,salesday)

Do the following

a) write a function that displays the product name and grade of the given product b)whenever the product sales is grater than the target value it is given a A grade, if there is no sale for a product an exception to be raised

PROGRAM

```
1) create function factorial (a1 in number)
return number
as
 f number:=1;
 i number:=1;
begin
 while(i<=a1)
 loop
 f:=f*i;
 i:=i+1;
 end loop;
 return (f);
end;

set serveroutput on;
declare
 a2 number:=&a2;
 c2 number:=0;
begin
 c2:=factorial(a2);
 dbms_output.put_line('Factorial: '||c2);
end;
```

OUTPUT

```
anonymous block completed
Enter the value of a2: 6
Factorial: 720
```

2) create table product(pid number(5),pname varchar(10),pgrade varchar(3));

```
insert into product values(1,'HDD','a');
insert into product values(2,'GoPro','b');
insert into product values(3,'laptop','b');
insert into product values(4,'mobile','b');
insert into product values(5,'DVD','c');
create table sales(pid number(5),samount number(10),sdate date,sday varchar(15));
insert into sales values(1,1000,'2-10-2017','Monday');
insert into sales values(3,1500,'4-10-2017','Wednesday');
insert into sales values(3,2000,'5-10-2017','Thursday');
insert into sales values(1,3500,'7-10-2017','Saturday');
insert into sales values(3,4000,'3-10-2017','Tuesday');

create function funct(a1 in number,a3 in number)
 return number
 as
 q number;
 z number;
 r number;
 x varchar(10);
 y varchar(10);
 begin
 select pname into x from product where pid=a1;
 select pgrade into y from product where pid=a1;

 dbms_output.put_line('name of product :'||x);
 dbms_output.put_line('grade of product :'||y);

 select sum(samount) into q from sales where pid=a1;
 dbms_output.put_line('sum of sales of '||a1||' is'||q);
 if(q>a3) then
 update product set pgrade='a' where pid=a1;
 end if;
 select count(pid) into z from sales where pid=a1;
 if(z<1) then
 r:=0;
 else
 r:=1;
 end if;
 return(r);
end;

set serveroutput on;
declare
 d1 number;
```

```
 e1 number;
 f1 number;
 pnull exception;
begin
d1:=&d1;
e1:=&e1;
f1:=funct(d1,e1);
if f1=0 then
 raise pnull;
end if;
exception
 when pnull then
 dbms_output.put_line('no sales corresponding to this pid ');
when no_data_found then
 dbms_output.put_line('no data found:');
end;
```

OUTPUT

select * from sales;

	PID	SAMOUNT	SDATE	SDAY
1	1	1000	02-10-17	Monday
2	3	1500	04-10-17	Wednesday
3	3	2000	05-10-17	Thursday
4	1	3500	07-10-17	Saturday
5	3	4000	03-10-17	Tuesday

select * from product;

	PID	PNAME	PGRADE
1	1	HDD	a
2	2	GoPro	b
3	3	laptop	b
4	4	mobile	b
5	5	DVD	c

anonymous block completed

Enter the value of d1 : 3

Enter the value of e1 : 5000

name of product :laptop
grade of product :b sum of sales of 3 is

7500 table updated.

select * from product;

	PID	PNAME	PGRADE
1	1	HDD	a
2	2	GoPro	b
3	3	laptop	a
4	4	mobile	b
5	5	DVD	c

anonymous block completed Enter the
value of d1 : 2
Enter the value of e1 : 1000 name of

product :GoPro grade of product :b sum of

sales of 2 is

no sales corresponding to this pid

Exp No-14

Creation of Packages

A package is a schema object that groups logically related PL/SQL types, variables, and

subprograms. Packages usually have two parts, a specification (spec) and a body.

The specification is the interface to the package. It declares the types, variables, constants, exceptions, cursors, and subprograms that can be referenced from outside the package. The body defines the queries for the cursors and the code for the subprograms.

The package body contains the implementation of every cursor and subprogram declared in the package spec. Subprograms defined in a package body are accessible outside the package only if their specs also appear in the package spec. If a subprogram spec is not included in the package spec, that subprogram can only be called by other subprograms in the same package. A package body must be in the same schema as the package spec.

The following is contained in a PL/SQL package

1. Get and Set methods for the package variables
2. Cursor declarations with the text of SQL queries.
3. Declarations for exceptions.
4. Declarations for procedures and functions that call each other.
5. Declarations for overloaded procedures and functions
6. Variables that you want to remain available between procedure calls in the same session

Advantages in using packages

Modularity

Easier Application Design

Information Hiding

Added Functionality

Better Performance

Syntax

CREATE [OR REPLACE] PACKAGE [schema.] package

[invoker_rights_clause]

{ IS | AS } [item_list_1] END [package_name] ;

Where

Schema- Specify the schema to contain the package. If you omit *schema*, then the database creates the package in your own schema.

Item list 1 : Declares package elements.

invoker_rights_clause : Specifies the AUTHID property of the member functions and procedures of the object type. The AUTHID clause determines whether all the packaged subprograms execute with the privileges of their definer (the default) or invoker, and whether their unqualified references to schema objects are resolved in the schema of the definer or invoker.

CREATE [OR REPLACE] PACKAGE BODY

[schema.] package

{ IS | AS } [declare_section] { body | END package_name } ;

Referencing Package Contents

To reference the types, items, subprograms, and call specs declared within a package spec, use dot notation:

package_name.type_name

package_name.item_name

package_name.subprogram_name

package_name.call_spec_name

QUESTION

1) create the table supplier(supplier id,suppliername,commission and city).

create a package comprising of procedure and function .

1. The procedure will accept a supplier id and display the corresponding supplier name,

2. The function will accept the supplier id and calculate the annual commission and finally return the amount

Algorithm

1. Create a package with declaration for a procedure and a package.
2. Cretae the package body including the definitions for procedure and function

PROGRAM

```
create table supplier(suid number(3),suname varchar(10),sucom number(10),sucity varchar(10));
insert into supplier values(1,'adam',1000,'pune');

insert into supplier values(2,'john',2000,'mumbai');

insert into supplier values(3,'adhya',1500,'kochi');

insert into supplier values(4,'farhaan',3000,'kochi');

insert into supplier

values(5,'merina',2500,'banglore');


select * from supplier;


create package pack_age as

  procedure alpha(a1 in number,c1 out

varchar);  function beta(a1 in number) return

number; end pack_age;


create package body pack_age as

     procedure alpha(a1 in number,c1 out

     varchar)  is

     begin

        select suname into c1 from supplier where

      suid=a1;  end alpha;


     function beta(a1 in number)

     return number
```

```
    is
    s1 number:=0;
    s2 number;
    begin
        select sucom into s2 from supplier where
        suid=a1;  s1:=s2*12;
    return(s1);
    end;

end pack_age;

    set serveroutput on;
    declare
     a1 number:=&a1;
    c1 varchar(10);
    bb1 number:=0;
    begin
        pack_age.alpha(a1,c1);
        dbms_output.put_line('supplier correspoding to '||a1 ||' is
    '||c1); end;

    set serveroutput on;
    declare

     b1 number:=&b1;
    c1 varchar(10);
    bb1 number:=0;
```

```
begin
    bb1 :=pack_age.beta(b1);
    dbms_output.put_line('annual commission for '||b1||' is '||bb1);
end;
```

OUTPUT

```
create table supplier(suid number(3),suname varchar(10),sucom number(10),sucity varchar(10));  insert into supplier values(1,'adam',1000,'pune');
insert into supplier values(2,'john',2000,'mumbai');

insert into supplier values(3,'adhya',1500,'kochi');

insert into supplier values(4,'farhaan',3000,'kochi');

insert into supplier values(5,'merina',2500,'banglore');

select * from supplier;
```

	SUID	SUNAME	SUCOM	SUCITY
1	1	adam	1000	pune
2	2	john	2000	mumbai
3	3	adhya	1500	kochi
4	4	farhaan	3000	kochi
5	5	merina	2500	banglore

on Executing package and package body(do it as step 1)

PACKAGE PACK_AGE compiled

PACKAGE BODY PACK_AGE compiled

On Executing plsql containing procedure

input: Enter the value of a1 : 3

output: supplier correspoding to 3 is adhya

On Executing plsql containing function
input: Enter the value of b1 : 5 output:

annual commission for 5 is 30000

Exp No-15

Creation of Cursors

AIM:To implement programs using Cursors

QUESTIONS

1) Consider the table Customer (accout no, customer name,balance amount,date of join).

Implement a PL/SQL block to insert those customers who have current balance greater than 1 Lakh and date of join before 1 january 2010 into the table premium customer who doesnt meet above criteria are to be inserted into table nonpremium customer .

PROGRAM

```
create table customers(acc_no number(16),c_name varchar(15),bal_amt number(10),DOJ date);

create table premium_customers(acc_no number(16),c_name varchar(15),bal_amt number(10),DOF date);

create table nonpremium_customers(acc_no number(16),c_name varchar(15),bal_amt number(10),DOF date);


insert into customers values(101,'anu',150000,'12-12-1998');

insert into customers values(102,'anjana',200000,'19-08-1997');

insert into customers values(208,'achu',50000,'8-09-2003');

insert into customers values(305,'gopika',95000,'10-10-2010');

insert into customers values(409,'irene',25000,'4-03-2011');

insert into customers values(111,'dany',100000,'11-05-2010');


set serveroutput on;

declare

  cursor s is select * from customers;

begin

 for r in s

 loop

  if r.bal_amt>100000 and r.doj<'01-01-2010' then

   insert into premium_customers values(r.acc_no,r.c_name,r.bal_amt,r.doj);

  else
```

```
   insert into nonpremium_customers values(r.acc_no,r.c_name,r.bal_amt,r.doj);
  end if;
 end loop;
end;
```

OUTPUT

select * from customers;

	ACC_NO	C_NAME	BAL_AMT	DOJ
1	101	anu	150000	12-12-98
2	102	anjana	200000	19-08-97
3	208	achu	50000	08-09-03
4	305	gopika	95000	10-10-10
5	409	irene	25000	04-03-11
6	111	dany	100000	11-05-10

select * from premium_customers;

	ACC_NO	C_NAME	BAL_AMT	DOF
1	101	anu	150000	12-12-98
2	102	anjana	200000	19-08-97

select * from nonpremium_customers;

	ACC_NO	C_NAME	BAL...	DOF
1	208	achu	50000	08-09-03
2	305	gopika	95000	10-10-10
3	409	irene	25000	04-03-11
4	111	dany	100000	11-05-10

2) Consider the table Account(Customer name,account number,date_last transaction,amount). Implement a PL/SQL block to perform the following action on the table .
Calculate the interest of each person if it satisfies the condition
a)if the last transaction is not on the current month insert the records into inactive customer
b)otherwise check the balance amount and display the interest amount
i)if the balance amount is less than 50000 interest rate is 5% of the amount
ii)if it is between 250000 and 5 Lakhs interest rate is 10%
iii)if the amount is greater than 5 lakh interest rate is 15%

PROGRAM

```
set serveroutput on;

create table accdetails(accno number(10),cname varchar(20),lastdate date,amount number(7));

insert into accdetails values(101,'anu','08-11-2017',50000);
insert into accdetails values(102,'anagha','10-10-2017',100000);
insert into accdetails values(103,'elizabeth','17-11-2017',25000);
insert into accdetails values(104,'george','06-10-2017',300000);
insert into accdetails values(105,'albyn','15-11-2017',650000);

create table inactive_customer(accno number(10),cname varchar(20));

 declare
 cursor c1 is select * from accdetails;
 a accdetails %rowtype;
 n number;
 i number;
 begin
 open c1;
 loop
        fetch c1 into a.accno,a.cname,a.lastdate,a.amount;
        exit when c1 %notfound;
        n:=months_between(sysdate,a.lastdate);

        if n>=1 then
                insert into inactive_customer values(a.accno,a.cname);

        else
                if a.amount<250000 then
                i:=a.amount*0.05;
dbms_output.put_line('interest of '||a.cname|| ' is :'||i);
elsif a.amount>250000 and a.amount<500000 then
i:=a.amount*0.1;
```

```
 dbms_output.put_line('interest of '||a.cname|| ' is :'||i);
elsif a.amount>500000 then
 i:=a.amount*0.15;
  dbms_output.put_line('interest of '||a.cname|| ' is :'||i);
else
dbms_output.put_line('error');
 end if;
 end if;
 end loop;
 end;
```

OUTPUT

select * from accdetails;

	ACCNO	CNAME	LASTDATE	AMOUNT
1	101	anu	08-11-17	50000
2	102	anagha	10-10-17	100000
3	103	elizabeth	17-11-17	25000
4	104	george	06-10-17	300000
5	105	albyn	15-11-17	650000

anonymous block completed

interest of anu is :2500
interest of elizabeth is :1250
interest of albyn is :97500

select * from inactive_customer;

	ACCNO	CNAME
1	102	anagha
2	104	george

Exp No:16

Creation of plsql blocks for Exception Handling

AIM:To implement plsql blocks for exception handling

QUESTION

Sales of different products in one week is recorded
Product (productid,productname,grade)
Sales (prdctname,salesamount,salesday)

Do the following
a) write a function that displays the product name and grade of the given product
b)whenever the product sales is grater than the target value it is given a A grade, if there is no sale for a product an exception to be raised

Query

```
create table product(pid number(5),pname varchar(10),pgrade varchar(3));

insert into product values(1,'HDD','a');
insert into product values(2,'GoPro','b');
insert into product values(3,'laptop','b');
insert into product values(4,'mobile','b');
insert into product values(5,'DVD','c');
create table sales(pid number(5),samount number(10),sdate date,sday varchar(15));
insert into sales values(1,1000,'2-10-2017','Monday');
insert into sales values(3,1500,'4-10-2017','Wednesday');
insert into sales values(3,2000,'5-10-2017','Thursday');
insert into sales values(1,3500,'7-10-2017','Saturday');
insert into sales values(3,4000,'3-10-2017','Tuesday');

create function funct(a1 in number,a3 in number)
 return number
 as
 q number;
 z number;
 r number;
 x varchar(10);
 y varchar(10);
```

```
 begin
 select pname into x from product where pid=a1;
 select pgrade into y from product where pid=a1;

 dbms_output.put_line('name of product :'||x);
 dbms_output.put_line('grade of product :'||y);

 select sum(samount) into q from sales where pid=a1;
 dbms_output.put_line('sum of sales of '||a1||' is'||q);
 if(q>a3) then
 update product set pgrade='a' where pid=a1;
 end if;
 select count(pid) into z from sales where pid=a1;
 if(z<1) then
 r:=0;
 else
 r:=1;
 end if;
 return(r);
end;

set serveroutput on;
declare
 d1 number;
 e1 number;
 f1 number;
 pnull exception;
begin
d1:=&d1;
e1:=&e1;
f1:=funct(d1,e1);
if f1=0 then
 raise pnull;
end if;
exception
 when pnull then
 dbms_output.put_line('no sales corresponding to this pid ');
when no_data_found then
```

```
 dbms_output.put_line('no data found:');
end;
```

OUTPUT

select * from sales;

	PID	SAMOUNT	SDATE	SDAY
1	1	1000	02-10-17	Monday
2	3	1500	04-10-17	Wednesday
3	3	2000	05-10-17	Thursday
4	1	3500	07-10-17	Saturday
5	3	4000	03-10-17	Tuesday

select * from product;

	PID	PNAME	PGRADE
1	1	HDD	a
2	2	GoPro	b
3	3	laptop	b
4	4	mobile	b
5	5	DVD	c

anonymous block completed

Enter the value of d1 : 3

Enter the value of e1 : 5000

name of product :laptop

grade of product :b sum of sales of 3 is

7500 table updated.

select * from product;

	PID	PNAME	PGRADE
1	1	HDD	a
2	2	GoPro	b
3	3	laptop	a
4	4	mobile	b
5	5	DVD	c

anonymous block completed Enter the
value of d1 : 2
Enter the value of e1 : 1000 name of

product :GoPro grade of product :b sum of

sales of 2 is

no sales corresponding to this pid

Exp No:17

Familioarisation of NoSQL Database and CRUD Operations

AIM: Familioarisation of NoSQL Database and CRUD Operations(MONGODB)

Create the tables mentioned below using crude operation - MongoDB.

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores student and course information.

Starting mongodb:

```
user@user-HP-Notebook:~$ mongo
MongoDB shell version: 3.0.14
connecting to: test
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2017-02-26T21:38:59.684+0530 I CONTROL  [initandlisten]
2017-02-26T21:38:59.684+0530 I CONTROL  [initandlisten] ** WARNING: /sys/kernel/
mm/transparent_hugepage/enabled is 'always'.
2017-02-26T21:38:59.684+0530 I CONTROL  [initandlisten] **        We suggest set
ting it to 'never'
2017-02-26T21:38:59.684+0530 I CONTROL  [initandlisten]
2017-02-26T21:38:59.684+0530 I CONTROL  [initandlisten] ** WARNING: /sys/kernel/
mm/transparent_hugepage/defrag is 'always'.
2017-02-26T21:38:59.684+0530 I CONTROL  [initandlisten] **        We suggest set
ting it to 'never'
2017-02-26T21:38:59.684+0530 I CONTROL  [initandlisten]
```

To create a database “shyama” using “use” command:

```
> use shyama
switched to db shyama
> db
shyama
> show dbs
local  0.078GB
```

1.To create a collection “Student” in the database “Shyama”:

```
> show dbs
local  0.078GB
> db
shyama
> db.Student
shyama.Student
```

To insert multiple documents at a time :

Step 1) Create a JavaScript variable called myStudent to hold the array of documents

Step 2) Add the required documents with the Field Name and values to the variable

Step 3) Use the insert command to insert the array of documents into the collection

```
> var myStudent=
... [
... {
... "Name" : "Smith",
... "Student_number" : 17,
... "Class" : 1,
... "Major" : "CS"
... },
... {
...
... "Name" : "Brown",
... "Student_number" : 8,
... "Class" : 2,
... "Major" : "CS"
... },
... ];
> db.Student.insert(myStudent);
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

The output shows that those 3 documents were added to the collection.

Use the JSON print functionality to see the output in a better format.

```
> db.Student.find().forEach(printjson)
{
        "_id" : ObjectId("58b30cf73062a399bc306c70"),
        "Name" : "Smith",
        "Student_number" : 17,
        "Class" : 1,
        "Major" : "CS"
}
{
        "_id" : ObjectId("58b30cf73062a399bc306c71"),
        "Name" : "Brown",
        "Student_number" : 8,
        "Class" : 2,
  ];
```

2.To create a collection “Course” and to insert fields:

```
> db.Course
shyama.Course
> var myCourse=
... [
... {
... "Course_Name" : "Intro to Computer Science",
... "Course_number" : "CS1310",
... "Credit_hours" : 4,
... "Department" : "CS"
... },
... {
...
... "Course_Name" : "Data Structures",
... "Course_number" : "CS3320",
... "Credit_hours" : 4,
... "Department" : "CS"
... },
... {
...
... "Course_Name" : "Discrete Mathematics",
... "Course_number" : "MATH2410",
... "Credit_hours" : 3,
... "Department" : "MATH"
... },
... {
...
... "Course_Name" : "Database",
... "Course_number" : "CS3380",
... "Credit_hours" : 3,
... "Department" : "CS"
... },
... ];
```

```
>  db.Course.insert(myCourse);
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 4,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

```
> db.Course.find().forEach(printjson)
{
        "_id" : ObjectId("58b314523062a399bc306c76"),
        "Course_Name" : "Intro to Computer Science",
        "Course_number" : "CS1310",
        "Credit_hours" : 4,
        "Department" : "CS"
}
{
        "_id" : ObjectId("58b314523062a399bc306c77"),
        "Course_Name" : "Data Structures",
        "Course_number" : "CS3320",
        "Credit_hours" : 4,
        "Department" : "CS"
}
{
        "_id" : ObjectId("58b314523062a399bc306c78"),
        "Course_Name" : "Discrete Mathematics",
        "Course_number" : "MATH2410",
        "Credit_hours" : 3,
        "Department" : "MATH"
}
{
        "_id" : ObjectId("58b314523062a399bc306c79"),
        "Course_Name" : "Database",
        "Course_number" : "CS3380",
        "Credit_hours" : 3,
        "Department" : "CS"
}
```

3.To create collection “Section” and to insert fields:

```
> db.Section
shyama.Section
```

```
> var mySection=
... [
... {
... "Section_identifier" : 85,
... "Course_number" : "MATH2410",
... "Semester" : "Fall",
... "Year" : 07,
... "Instructor" : "King"
... },
... {
...
... "Section_identifier" : 92,
... "Course_number" : "CS1310",
... "Semester" : "Fall",
... "Year" : 07,
... "Instructor" : "Anderson"
... },
... {
...
... "Section_identifier" : 102,
... "Course_number" : "CS3320",
... "Semester" : "Spring",
... "Year" : 08,
... "Instructor" : "Knuth"
... },
... {
...
... "Section_identifier" : 112,
... "Course_number" : "MATH2410",
... "Semester" : "Fall",
... "Year" : 08,
... "Instructor" : "Chang"
... },
... {
```

```
...
... "Section_identifier" : 119,
... "Course_number" : "CS1310",
... "Semester" : "Fall",
... "Year" : 08,
... "Instructor" : "Anderson"
... },
... {
...
... "Section_identifier" : 135,
... "Course_number" : "CS3380",
... "Semester" : "Fall",
... "Year" : 08,
... "Instructor" : "Slone"
... },
... ];
> db.Section.insert(mySection);
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 6,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

```
> db.Section.find().forEach(printjson)
{
        "_id" : ObjectId("58b314ce3062a399bc306c7a"),
        "Section_identifier" : 85,
        "Course_number" : "MATH2410",
        "Semester" : "Fall",
        "Year" : 7,
        "Instructor" : "King"
}
{
        "_id" : ObjectId("58b314ce3062a399bc306c7b"),
        "Section_identifier" : 92,
        "Course_number" : "CS1310",
        "Semester" : "Fall",
        "Year" : 7,
        "Instructor" : "Anderson"
}
{
        "_id" : ObjectId("58b314ce3062a399bc306c7c"),
        "Section_identifier" : 102,
        "Course_number" : "CS3320",
        "Semester" : "Spring",
        "Year" : 8,
        "Instructor" : "Knuth"
}
{
        "_id" : ObjectId("58b314ce3062a399bc306c7d"),
        "Section_identifier" : 112,
        "Course_number" : "MATH2410",
        "Semester" : "Fall",
        "Year" : 8,
        "Instructor" : "Chang"
}
```

```
{
        "_id" : ObjectId("58b314ce3062a399bc306c7d"),
        "Section_identifier" : 112,
        "Course_number" : "MATH2410",
        "Semester" : "Fall",
        "Year" : 8,
        "Instructor" : "Chang"
}
{
        "_id" : ObjectId("58b314ce3062a399bc306c7e"),
        "Section_identifier" : 119,
        "Course_number" : "CS1310",
        "Semester" : "Fall",
        "Year" : 8,
        "Instructor" : "Anderson"
}
{
        "_id" : ObjectId("58b314ce3062a399bc306c7f"),
        "Section_identifier" : 135,
        "Course_number" : "CS3380",
        "Semester" : "Fall",
        "Year" : 8,
        "Instructor" : "Slone"
}
```

4.To create collection "GradeReport " and to insert the fields:

```
> db.GradeReport
shyama.GradeReport
> var myGradeReport=
... [
... {
... "Student_Number" : 17,
... "Section_Identifier" : 112,
... "Grade" : "B",
...
... },
... {
...
... "Student_Number" : 17,
... "Section_Identifier" : 119,
... "Grade" : "C",
... },
... {
... "Student_Number" : 8,
... "Section_Identifier" : 85,
... "Grade" : "A",
... },
... {
...
... "Student_Number" : 8,
... "Section_Identifier" : 92,
... "Grade" : "A",
... },
... {
...
... "Student_Number" : 8,
... "Section_Identifier" : 102,
... "Grade" : "B",
... },
```

```
... {
...
... "Student_Number" : 8,
... "Section_Identifier" : 102,
... "Grade" : "B",
... },
... {
...
... "Student_Number" : 8,
... "Section_Identifier" : 135,
... "Grade" : "A",
... },
... ];
> db.GradeReport.insert(myGradeReport);
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 6,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

```
> db.GradeReport.find().forEach(printjson)
{
        "_id" : ObjectId("58b316a93062a399bc306c80"),
        "Student_Number" : 17,
        "Section_Identifier" : 112,
        "Grade" : "B"
}
{
        "_id" : ObjectId("58b316a93062a399bc306c81"),
        "Student_Number" : 17,
        "Section_Identifier" : 119,
        "Grade" : "C"
}
{
        "_id" : ObjectId("58b316a93062a399bc306c82"),
        "Student_Number" : 8,
        "Section_Identifier" : 85,
        "Grade" : "A"
}
{
        "_id" : ObjectId("58b316a93062a399bc306c83"),
        "Student_Number" : 8,
        "Section_Identifier" : 92,
        "Grade" : "A"
}
{
        "_id" : ObjectId("58b316a93062a399bc306c84"),
        "Student_Number" : 8,
        "Section_Identifier" : 102,
        "Grade" : "B"
}
{
        "_id" : ObjectId("58b316a93062a399bc306c85"),
        "Student_Number" : 8,
        "Section_Identifier" : 135,
        "Grade" : "A"
}
```

5.To insert collection “Prerequisite” and to enter the fields:

```
> db.Prerequisite
shyama.Prerequisite
```

```
> var myPrerequisite=
... [
... {
... "Course_Number" : "CS3380",
... "Prerequisite_Number" : "CS3320",
... },
... {
...
... "Course_Number" : "CS3380",
... "Prerequisite_Number" : "MATH2410",
... },
... {
... "Course_Number" : "CS3320",
... "Prerequisite_Number" : "CS1310",
... },
...
... ];
> db.Prerequisite.insert(myPrerequisite);
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 3,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

```
> db.Prerequisite.find().forEach(printjson)
{
        "_id" : ObjectId("58b318813062a399bc306c86"),
        "Course_Number" : "CS3380",
        "Prerequisite_Number" : "CS3320"
}
{
        "_id" : ObjectId("58b318813062a399bc306c87"),
        "Course_Number" : "CS3380",
        "Prerequisite_Number" : "MATH2410"
}
{
        "_id" : ObjectId("58b318813062a399bc306c88"),
        "Course_Number" : "CS3320",
        "Prerequisite_Number" : "CS1310"
}
```