Shri Gajanan Maharaj Shikshan Prasarak Mandal's

SHARADCHANDRA PAWAR COLLEGE OF ENGINEERING

Otur (Dumbarwadi) , Tal.Junnar,Dist.Pune-412409

CERTIFICATE

This is to certify that,

Mr. ____________________________Student of________**Year** Branch**COMPUTER ENGINEERING** Roll No.___has performed Practical's Work in the Subject **Database Management System** satisfactorily.

It is further certified that she has done this work in the premises of college as per the syllabus of Savitribai Phule Pune University during the Academic Year 2022-2023

Date: / /2022

Place: Otur. **Signature of the**

Teacher **Head of the Department** **Principal**

Sharadchandra Pawar College of Engineering, Pune

Department of Computer Engineering

DATABASE MANAGEMENT SYSTEM LABORATORY MANUAL

A.Y. 2022-23

SEMESTER-I

Subject Code: -310246

TEACHING SCHEME	EXAMINATION SCHEME
Practical: 4Hrs/Week	Practical Assessment: 25 Marks
Term Work: 25 Marks	

Name of Faculty:- Prof. Dumbre R.C

Companion Course: Database Management System (310246)

Course Objectives:

•To develop basic, intermediate and advanced Database programming skills .
•To develop basic Database administration skills.
•To percept transaction processing.

Course Outcomes:

On completion of the course, student will be able to–
•Develop the ability to handle databases of varying complexities
•Use advanced database Programming concepts

Guidelines for Student Journal

The laboratory assignments are to be submitted by student in the form of journal. Journal consists of program, Certificate, table of contents, and handwritten write-upof each assignment (Title, Objectives, Problem Statement, Outcomes, software & Hardware requirements, Date of Completion, Assessment grade/marks and assessor's sign, Theory-Concept in brief, Database design, test cases, conclusion/analysis. Program codes with sample output of all performed assignments are to be submitted as softcopy.As aConscious effort and little contribution towards Green IT and environment awareness, attaching printed papers as part of write-ups and program listing to journal may be avoided. Use of DVD containing students programs maintained by lab In-charge is highly encouraged. For reference one or two journals may be maintained with program prints at Laboratory

Operating System recommended:-64-bit Open source Linux or its derivativeProgramming tools recommended: SQL, PL/SQL, FrontEnd: Java/Perl/PHP/Python/Ruby/.net,
Backend : Monod/MYSQL/Oracle, Database Connectivity : ODBC/JDBC
Books:
References:

1. Ivan Bayross, "SQL, PL/SQL: The
Programming Language of Oracle", BPB Publication, ISBN-10: 8176560723;
ISBN-13: 978-8176560726
2.Kristina Chodorow, Michael Dirolf, "MangoDB: The Definitive Guide", O'Reilly Publications, ISBN: 9781449381561
3.Import, Tidy, Transform," R for DataScience", O'REILLY, ISBN: 13:978-93-5213-497-7
4.http://www.tutorialspoint.com/json/& http://docs.mongodb.org/manual/

Set of suggested assignment list is provided in groups-A and B. Each student must perform at least 13 assignments (8-Mandotory plus 4 from remaining 8 assignments) from group A , 5 from group B and 2 mini projects from Group C.

Vision

“To create a best quality computer engineering professionals with research and innovative skills.”

Mission

- To educate our students to meet high standards of excellence in computer engineering.
- To create and distribute new knowledge through basic and applied research in computer engineering.

- To create computer engineering expertise who serves as a resource at the local, regional and national levels.

- To provide quality technical education to the students through innovative and interactive teaching learning process.

INDEX

SR. NO	NAME OF EXPERIMENT	PAGE NO	DATE OF PERFORMANCE	DATE OF SUBMISSION	SIGN
1.	ER odeling and Normalization				
2.	a. Design and Develop SQLDDL statements which demonstrate the use of SQL objects suchas Table, View, Index, Sequence, Synonym, different constraints etc. b. Write at least 10 SQL queries on the suitable database application using SQL DML statements.				
3.	Write at least10 SQL queries for suitable database application using SQL DML statements. Note: Instructor will design the queries which demonstrate the use of concepts like all types of Join ,Sub-Query and View.				
4.	Unnamed PL/SQLcode block: Use of Control structure and Exception handling ismandatory. Suggested Problem statement: Consider Tables: 1. Borrower(Roll_no, Name, Date of Issue, Name of Book, Status) 2. Fine(Roll_no, Date, Amt) • Accept Roll_no and Name of Book from user. • Check the number of days (from date of issue). • If days are between 15 to 30 then fine amount will be Rs 5per day. If no. of days>30, per day fine will be Rs 50 per day and for days less than 30, Rs. 5 perday.				

5.	Unnamed PL/SQL code block: Use of Control structure and Exception handling is mandatory. • After submitting the book, status will change from I to R. • If condition of fine is true, then details will be stored into fine table. Also handles the exception by named exception handler or user define exception handler				
6.	PL/SQL Stored Procedure and Stored Function. Write a Stored Procedure namely proc_Grade for the categorization of student. If marks scored by students in examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored are between 989 and900 category is first class, if marks 899 and 825 category is Higher Second Class Write a PL/SQL block for using procedure created with above requirement. Stud_Marks(name, total_marks) Result(Roll,Name, Class).				
7.	Cursors: (All types: Implicit, Explicit, Cursor FOR Loop, Parameterized Cursor) Write a PL/SQL block of code using parameterized Cursor, that will merge the data available in the newly created table N_RollCall with the data available in the table O_RollCall. If the data in the first table already exist in the second table then that data should be skipped.				

8.	Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers). Write a database trigger on Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in Library_Audit table				
9.	Database Connectivity: Write a program to implement MySQL/Oracle database connectivity with any front end language to implement Database navigation operations (add, delete, edit etc.)				
10.	Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations, SAVE method, logical operators) .				
11.	Implement aggregation and indexing with suitable example using MongoDB.				
12.	Implement Map reduces operation with suitable example using MongoDB.				
13.	Write a program to implement Mongo DB database connectivity with any front end language to implement Database navigation operations(add, delete, edit etc.)				

Subject Teacher
Prof. Dumbre Sir

Head of the Department
S.S Prof. Khatal S.

Assignment - 1

TITLE :

Conceptual design using ER Model, Reducing ER into tables, normalization

OBJECTIVE :-Propose a Conceptual Design using ER features using tools like ERDplus,ER Win etc. Convert the ER diagram into tables on paper and propose a normalize Relational data model.

PROBLEM STATEMENT:

College Management System

THEORY:

| | All ER Notations

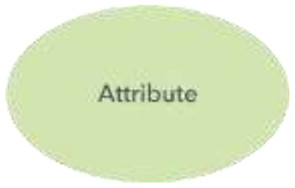
Ent ty: A definable thing—such as a person, object, concept or event—that canhave data storedabout it.

Relat onship: How entities act upon each other or are associated with each other. Think of relationships as verbs. Relationships are typically shown as diamonds orlabels directly on theconnecting lines.

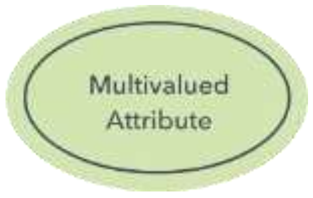
Relationship

Attribute: A property or characteristic of an entity. Often shown as an oval or circle.

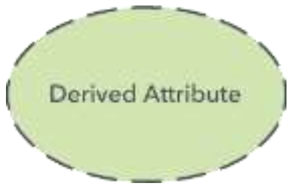
- Simple: Means the attribute value is atomic and can’t be further divided, such as a phonenumber.



- Mu t valued Attribute: More than one attribute value is denoted, suchas multiple phonenumbers for a person.



- Derived Attribute: Attributed is calculated or otherwise derivedfrom another attribute,such as age from a birthdate.



- Composite Attribute: Sub-attributes spring from an attribute.

| | Brief ru es to reduce ER diagram into tables

There are some points for converting the ER diagram to the table:

- o Ent ty type becomes a table.

In the given ER diagram, LECTURE, STUDENT, SUBJECT and COURSE forms individual tables.

- o Al single-valued attribute becomes a column for the table.

In the STUDENT entity, STUDENT_NAME and STUDENT_ID form the column of STUDENT table. Similarly, COURSE_NAME and COURSE_ID form the column of COURSE table and so on.

- A key attribute of the entity type represented by the primary key.

In the given ER diagram, COURSE_ID, STUDENT_ID, SUBJECT_ID, and LECTURE_ID arethe key attribute of the entity.

- The mult valued attribute is represented by a separate table.

In the student table, a hobby is a multivalued attribute. So it is not possible torepresent multiple values in a single column of STUDENT table. Hence we create a table STUD_HOBBY with column name STUDENT_ID and HOBBY. Using both thecolumn, we create a composite key.

- Composite attribute represented by components.

In the given ER diagram, student address is a composite attribute. It contains CITY, PIN, DOOR#, STREET, and STATE. In the STUDENT table, these attributes can merge as an individual column.

- Derived attributes are not considered in the table.

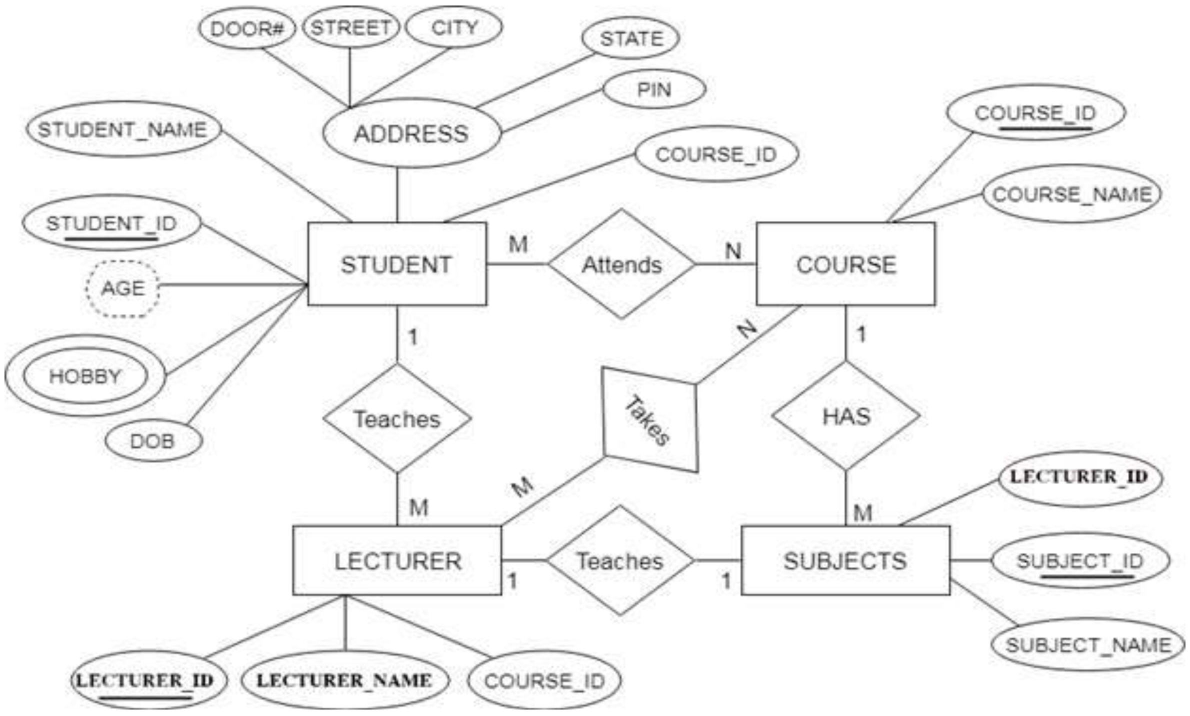
In the STUDENT table, Age is the derived attribute. It can be calculated at any point of time by calculating the difference between current date and Date of Birth.

Using these rules, you can convert the ER diagram to tables and columns and assign the mapping between the tables. Table structure for the given ER diagram is as below:

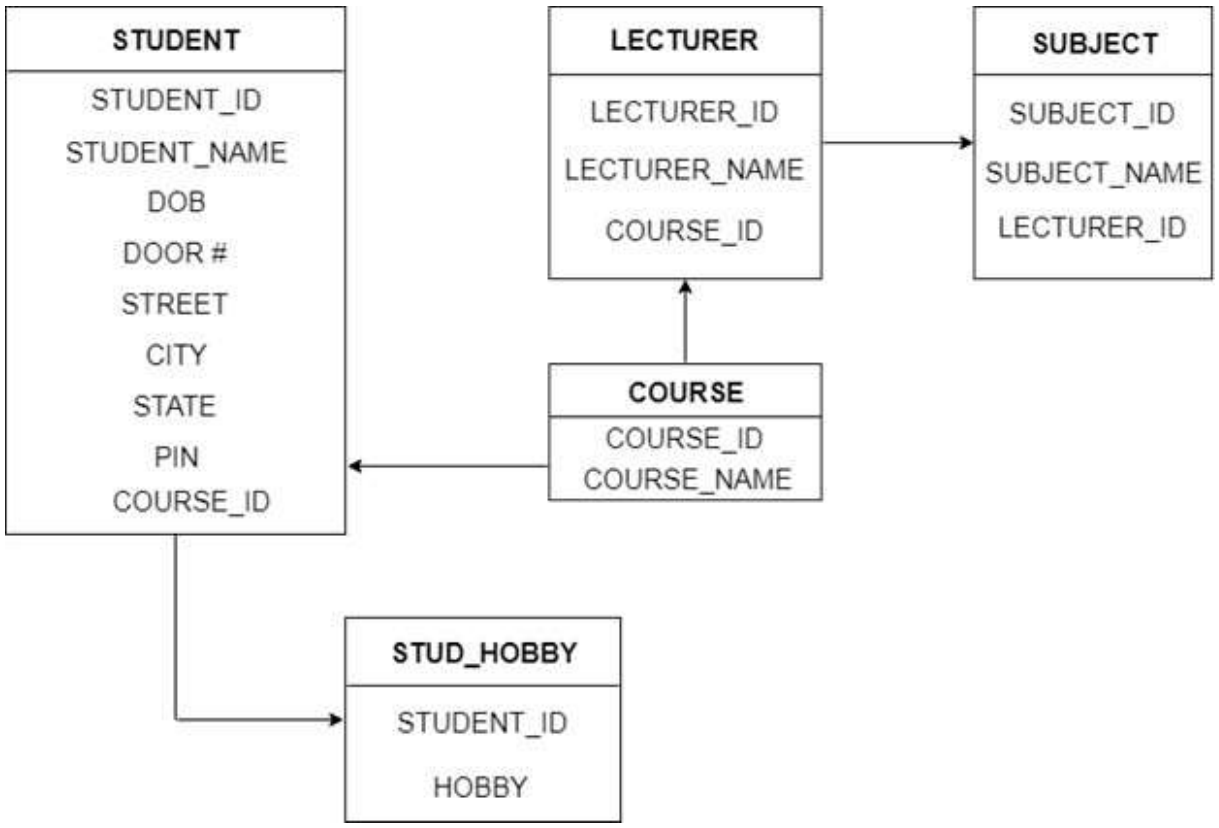
Brief about 1NF, 2NF, 3NF and BCNF

Normal Form	Descript on
1NF	A relation is in 1NF if it contains an atomic value.
2NF	A relation will be in 2NF if it is in 1NF and all non-key attributesare fully functional dependent on the primary key.
3NF	A relation will be in 3NF if it is in 2NF and no transitiondependency exists.
BCNF	A stronger definition of 3NF is known as Boyce Codd's normalform.

E R Diagram (for your Problem Statement):



Reduced Table Structure from above E-R Diagram:



Normalized Tables at least in 3NF:

CONCLUSION:-

Hence, we have successfully design college management system using ER Model,Reducing ER into tables, normalization.

Assignment No:-2

Aim: Design and develop SQL DDL statements which demonstrate the uses of SQL objects such as Table, view, Index,Sequence, Synonym.

Objective:
To learn and understand open the SQL DDL queries to create table,view,index,sequence,synonym.

Outcomes:
Students will be able to learn concepts of DDL commands.

Hardware Requirements:
Any CPU with Pentinum Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.

Software Requirements:
LINUX operating system, MySQL.

Program and Output:

```
mysql> create database vs;
Query OK, 1 row affected (0.00 sec)
mysql> use vs;
Database changed
mysql> create table employee(id int, name varchar(255),city varchar(255),salary int);
Query OK, 0 rows affected (0.10 sec)
mysql> desc employee;
+--------+--------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| id | int(11) | YES | | NULL | |
| name | varchar(255) | YES | | NULL | |
| city | varchar(255) | YES | | NULL | |
| salary | int(11) | YES | | NULL | |
+--------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
mysql> alter table employee add contact int;
Query OK, 0 rows affected (0.27 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql> desc employee;
+---------+--------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+---------+--------------+------+-----+---------+-------+
| id | int(11) | YES | | NULL | |
| name | varchar(255) | YES | | NULL | |
| city | varchar(255) | YES | | NULL | |
| salary | int(11) | YES | | NULL | |
| contact | int(11) | YES | | NULL | |
+---------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)


mysql> alter table employee drop contact;
Query OK, 0 rows affected (0.25 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql> desc employee;
+--------+--------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| id | int(11) | YES | | NULL | |
| name | varchar(255) | YES | | NULL | |
| city | varchar(255) | YES | | NULL | |
| salary | int(11) | YES | | NULL | |
+--------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
mysql> insert into employe values(1,'Prachi','Pune',70000),(2,'Gauri','Dholwad',90000),
(3,'Sonu','Thane',80000),(4,'Vaishnavi','Mumbai',80000),(5,'Renuka','Buldhana',60000),
(6,'Swati','Ozar',60000),(7,'Rutika','Otur',40000),(8,'Nikita','Satara',50000),(9,'Sonal','Narayangoan',80000),
(10,'Akanksha','Kalwadi',40000);
Query OK, 10 rows affected (0.06 sec)
Records: 10 Duplicates: 0 Warnings: 0

mysql> select * from employe;
+------+-----------+--------+--------+
| id | name | city | salary |
+------+-----------+--------+--------+
| 1 | Prachi | Pune | 80000 |
| 2 | Gauri | Dholwad | 90000 |
| 3 | Sonu | Thane | 80000 |
| 4 | Vaishnavi | Mumbai | 80000 |
| 5 | Renuka | Buldhana | 60000 |
| 6 | Swati | Ozar | 60000 |
| 7 | Rutika | Otur | 40000 |
| 8 | Nikita | Satara | 50000 |
| 9 | Sonal | Narayangoan | 80000 |
| 10 | Akanksha | Kalwadi | 40000 |
+------+-----------+--------+--------+
10 rows in set (0.00 sec)

mysql> create table emp(id int, city varchar(255),salary int);Query OK, 0 rows affected (0.11 sec)
mysql> desc emp;
+--------+--------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| id | int(11) | YES | | NULL | |
| city | varchar(255) | YES | | NULL | |
| salary | int(11) | YES | | NULL | |
+--------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
mysql> truncate table emp;
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> select * from emp;
Empty set (0.00 sec)


mysql> create view ss as select city,salary from employee where name='Sonu';
Query OK, 0 rows affected (0.05 sec)
mysql> desc employee;
+--------+--------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| id | int(11) | YES | | NULL | |
| name | varchar(255) | YES | | NULL | |
| city | varchar(255) | YES | | NULL | |
| salary | int(11) | YES | | NULL | |
+--------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)

mysql> select * from ss;
+-------+--------+
| city | salary |
+-------+--------+
| Thane | 80000 |
+-------+--------+
1 row in set (0.00 sec)
mysql> create view sp as select * from employee;
Query OK, 0 rows affected (0.06 sec)
mysql> select * from sp;
+------+-----------+--------+--------+
| id | name | city | salary |
+------+-----------+--------+--------+
| 1 | Prachi | Pune | 80000 |
| 2 | Gauri | Dholwad | 90000 |
| 3 | Sonu | Thane | 80000 |
| 4 | Vaishnavi | Mumbai | 80000 |
| 5 | Renuka | Buldhana | 60000 |
| 6 | Swati | Ozar | 60000 |
| 7 | Rutika | Otur | 40000 |
| 8 | Nikita | Satara | 50000 |
| 9 | Sonal | Narayangoan | 80000 |
| 10 | Akanksha | Kalwadi | 40000 |
+------+-----------+--------+--------+
10 rows in set (0.00 sec)
mysql> use vs;
Database changed

mysql> select * from employee;
+------+-----------+--------+--------+
| id | name | city | salary |
+------+-----------+--------+--------+
| 1 | Prachi | Pune | 80000 |
| 2 | Gauri | Dholwad | 90000 |
| 3 | Sonu | Thane | 80000 |
| 4 | Vaishnavi | Mumbai | 80000 |
| 5 | Renuka | Buldhana | 60000 |
| 6 | Swati | Ozar | 60000 |
| 7 | Rutika | Otur | 40000 |
| 8 | Nikita | Satara | 50000 |
| 9 | Sonal | Narayangoan | 80000 |
| 10 | Akanksha | Kalwadi | 40000 |
+------+-----------+--------+--------+
10 rows in set (0.00 sec)
```

Conclusion:
Thus we have studied the DDL command for creating tables,view etc.

Assignment No:3

Aim: Design at least 10 SQL queries for suitable database application using SQL DML statements:all types of Join,Sub-Query and View.

Objective: To learn and understand SQL DML statements:all types of Join,Sub-Query and View.

Outcomes: Students will be able to learn concepts of all types of Joins,Sub-Query and View.

Hardware Requirements: Any CPU Pentium Processor or similar,256 MB RAM or more,1 GB Hard Disk or more.

Software Requirements: LINUX Operating System,MySQL.

Program and Output:

```
mysql> use Sonu;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed

mysql> create table s1(sid int,sname VARCHAR(20),price int);
Query OK, 0 rows affected (0.09 sec)

mysql> insert into s1 values(1,'Sonal',100);
Query OK, 1 row affected (0.04 sec)

mysql> insert into s1 values(2,'Prachi',200);
Query OK, 1 row affected (0.04 sec)

mysql> insert into s1 values(3,'Gauri',300);
Query OK, 1 row affected (0.05 sec)

mysql> insert into s1 values(4,'Vaishnavi',400);
Query OK, 1 row affected (0.04 sec)

mysql> insert into s1 values(2,'Shital',500);
Query OK, 1 row affected (0.06 sec)

mysql> insert into s1 values(4,'Sonal',600);
Query OK, 1 row affected (0.05 sec)
```

```
mysql> select * from s1;
+------+-----------+-------+
| sid  | sname     | price |
+------+-----------+-------+
|    1 | Sonal     |   100 |
|    2 | Prachi    |   200 |
|    3 | Gauri     |   300 |
|    4 | Vaishnavi |   400 |
|    2 | Shital    |   500 |
|    4 | Sonal     |   600 |
+------+-----------+-------+
6 rows in set (0.00 sec)

mysql> create table v1(did int,eid1 int,quality int,eprice int,cname VARCHAR(20));
Query OK, 0 rows affected (0.09 sec)

mysql> insert into v1 values(101,1,50,300,'Sonu');
Query OK, 1 row affected (0.04 sec)

mysql> insert into v1 values(102,1,50,200,'Shital');
Query OK, 1 row affected (0.05 sec)

mysql> insert into v1 values(103,1,50,200,'Vaishnavi');
Query OK, 1 row affected (0.05 sec)

mysql> insert into v1 values(104,1,20,100,'Prachi');
Query OK, 1 row affected (0.05 sec)

mysql> select * from v1;
+------+------+---------+--------+-----------+
| did  | eid1 | quality | eprice | cname     |
+------+------+---------+--------+-----------+
|  101 |    1 |      50 |    300 | Sonu      |
|  102 |    1 |      50 |    200 | Shital    |
|  103 |    1 |      50 |    200 | Vaishnavi |
|  104 |    1 |      20 |    100 | Prachi    |
+------+------+---------+--------+-----------+
4 rows in set (0.00 sec)


mysql> select sid,price from s1 inner join v1 where s1.sid=v1.eid1;
+------+-------+
| sid  | price |
+------+-------+
|    1 |   100 |
|    1 |   100 |
|    1 |   100 |
|    1 |   100 |
+------+-------+
4 rows in set (0.00 sec)
```

```
mysql> select sid,price ,v1.cname from s1 inner join v1 where s1.sid=v1.eid1;
+------+-------+-----------+
| sid  | price | cname     |
+------+-------+-----------+
|    1 |   100 | Sonu      |
|    1 |   100 | Shital    |
|    1 |   100 | Vaishnavi |
|    1 |   100 | Prachi    |
+------+-------+-----------+
4 rows in set (0.00 sec)

mysql> select sid,price ,v1.cname from s1 right outer join v1 on s1.sid=v1.eid1;
+------+-------+-----------+
| sid  | price | cname     |
+------+-------+-----------+
|    1 |   100 | Sonu      |
|    1 |   100 | Shital    |
|    1 |   100 | Vaishnavi |
|    1 |   100 | Prachi    |
+------+-------+-----------+
4 rows in set (0.00 sec)

mysql> select sid,price ,v1.cname from s1 left outer join v1 on s1.sid=v1.eid1;+------+-------+-----------+
| sid  | price | cname     |
+------+-------+-----------+
|    1 |   100 | Sonu      |
|    1 |   100 | Shital    |
|    1 |   100 | Vaishnavi |
|    1 |   100 | Prachi    |
|    2 |   200 | NULL      |
|    3 |   300 | NULL      |
|    4 |   400 | NULL      |
|    2 |   500 | NULL      |
|    4 |   600 | NULL      |
+------+-------+-----------+
9 rows in set (0.00 sec)
```

```
mysql> select sid,price ,v1.cname from s1 right outer join v1 on s1.sid=v1.eid1 union select sid, price,
v1.cname from s1 left outer join v1 on s1.sid=v1.eid1;
+------+-------+-----------+
| sid  | price | cname     |
+------+-------+-----------+
|    1 |   100 | Sonu      |
|    1 |   100 | Shital    |
|    1 |   100 | Vaishnavi |
|    1 |   100 | Prachi    |
|    2 |   200 | NULL      |
|    3 |   300 | NULL      |
|    4 |   400 | NULL      |
|    2 |   500 | NULL      |
|    4 |   600 | NULL      |
+------+-------+-----------+
9 rows in set (0.00 sec)

mysql> select * from s1 where sid=(select sid from s1 where sname='sonu');
Empty set (0.00 sec)

mysql> select * from s1 where sid=(select sid from s1 where sname='shital');
+------+--------+-------+
| sid  | sname  | price |
+------+--------+-------+
|    2 | Prachi |   200 |
|    2 | Shital |   500 |
+------+--------+-------+
2 rows in set (0.00 sec)
```

Conclusion:
Thus we have studied the concept of view,index,synonym.

Assignment No:-4

Aim:

Design at least 10 SQL queries for suitable database application using SQL DML statements: all types of Join, Sub-Query and View.

Objective:

To learn and understand SQL DML statements: all types of Join, Sub-Query and View.
.

Outcomes:

Students will be able to learn concepts of all types of Join, Sub-Query and View.
.

Hardware Requirements:

Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.

Software Requirements:

LINUX Operating System, MySQL.

Program and Output:

```
Mysql>use mydatabase;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed

Mysql>create table employee1(pro_id int,name varchar(20),prise int);
Query OK, 0 rows affected (0.05 sec)
Mysql>insert into employee1 value(1,"rushikesh",100);
Query OK, 1 row affected (0.02 sec)

Mysql>insert into employee1 values(2,"sonali",200);
Query OK, 1 row affected (0.02 sec)

Mysql>insert into employee1 value(1,"sonali",100);
Query OK, 1 row affected (0.03 sec)

Mysql>insert into employee1 values(2,"abhijit",500);
Query OK, 1 row affected (0.02 sec)

Mysql>insert into employee1 values(3,"abhijit",500);
Query OK, 1 row affected (0.02 sec)

Mysql>insert into employee1 values(4,"saurabh",500);
Query OK, 1 row affected (0.02 sec)

Mysql>insert into employee1 values(5,"sagar",500);
Query OK, 1 row affected (0.02 sec)

Mysql>select*from employee1;
+--------+-----------+-------+
| pro_id | name      | prise |
+--------+-----------+-------+
|      1 | rushikesh | 100 |
|      2 | sonali | 200 |
|      1 | sonali     | 100 |
|      2 | abhijit |   500 |
|      3 | abhijit |   500 |
|      4 | saurabh     | 500 |
|      5 | sagar     |  500 |
+--------+-----------+-------+
7 rows in set (0.00 sec)

Mysql>create table details(sale_id int,Pro_id1 int,quantity int,price int,customer_name varchar(20));
Query OK, 0 rows affected (0.09 sec)
```

```
Mysql>insert into details values(21,1,100,101,"rushikesh");
Query OK, 1 row affected (0.03 sec)

Mysql>insert into details values(31,1,100,101,"abhijit");
Query OK, 1 row affected (0.04 sec)

Mysql>insert into details values(41,1,100,101,"saurabh");
Query OK, 1 row affected (0.03 sec)

Mysql>insert into details values(41,1,100,101,"dipak");
Query OK, 1 row affected (0.03 sec)

Mysql>select*from details;
+.........+.........+..........+.......+..................+
| sale_id | Pro_id1 | quantity | price | customer_name |
+.........+.........+..........+.......+..................+
|      21 |       1 |      100 |   101 | rushikesh       |
|      31 |       1 |      100 |   101 | abhijit     |
|      41 |       1 |      100 |   101 | saurabh       |
|      41 |       1 |      100 |   101 | dipak        |
+.........+.........+..........+.......+..................+
4 rows in set (0.00 sec)
mysql>select Pro_id,Pro_price from employee1 inner join details where product_-
details.pro_id=details.Pro_id1;
+———+————+
| Pro_id | Pro_price |
+———+————+
| 1 | 100 |
| 3 | 500|
| 5 | 500 |
+———+————+
3 rows in set (0.00 sec)

mysql>select Pro_id,Pro_price,sale_details.customer_name from employee1 inner join
details where employee1.Pro_id=sale_details.Pro_id1;
+———+————+——————+
| Pro_id | Pro_price | customer_name |
+———+————+——————+
| 1 | 100 | rushikesh |
| 3 | 500 | abhijit |
| 5 | 500 |sagar |
+———+————+——————+
3 rows in set (0.00 sec)

mysql>select Pro_id,Pro_price,details.customer_name from employee1 inner join
details where employee1.Pro_id=sale_details.Pro_id1;
+———+————+——————+
| Pro_id | Pro_price | customer_name |
+———+————+—————+
| 1 | 100 | rushikesh|
| 3 | 80 | abhijit |
| 5 | 110 | sagar |
+———+————+—————+
3 rows in set (0.00 sec)

mysql>select Pro_id,Pro_price,details.customer_name from employee1 right outer
join details on employee1.Pro_id=details.Pro_id1;
+———+————+—————+
| Pro_id | Pro_price | customer_name |
+———+————+—————+
| 1 | 100 | rushikesh|
| NULL | NULL | sonali|
NULL |NULL|sonali|
| 3 | 500 | abhijit|
| NULL | NULL | saurabh |
| 5 | 500 | sagar |
+———+————+—————+
5 rows in set (0.04 sec)

mysql>select Pro_id,Pro_price,details.customer_name from employee1 left outer
join details on employee1.Pro_id=details.Pro_id1;
+———+————+—————+
| Pro_id | Pro_price | customer_name |
+———+————+—————+
| 1 | 100 | rushikesh |
| 2 | 200| NULL |
| 1 |100 | NULL|
|3 |500|abhijit|
| 4 | 90 | NULL|
| 5 | 110 | sagar |
+———+————+—————+
5 rows in set (0.00 sec)

mysql>select Pro_id,Pro_price,details.customer_name from employee1 left outer
join details on employee1.Pro_id=details.Pro_id1 union select Pro_id,Pro_price,
details.customer_name from employee1 left outer join details on employee
.Pro_id=sale_details.Pro_id1;
+———+————+—————+
| Pro_id | Pro_price | customer_name |
+———+————+—————+
| 1 | 100 | rushikesh |
| 2 | 70 | NULL |
| 3 | 80 | abhijit |
| 4 | 90 | NULL |
| 5 | 110 | sagar |
+———+————+—————+
5 rows in set (0.04 sec)
```

```
mysql>select *from employee1 where Pro_id=(select Pro_id from employee1 where
Pro_name=‘Rushikesh’);
+———+————-+————+
| Pro_id | Pro_name | Pro_price |
+———+————-+————+
| 1 | rushikesh | 100 |
+———+————-+————+
1 row in set (0.02 sec)
```

<u>**Conclusion:**</u>

Thus we have studied the concept of view ,index ,synonym.

Assignment No:-5

Aim:

Unnamed PL/SQL code block: Use of Control structure andException handling is mandatory. Write a PL/SQL block of code for the following requirements:- Schema:
1. Borrower(Rollin, Name, DateofIssue,NameofBook, Status)
2. Fine(Roll_no,Date,Amt) ?

Accept roll_no name of book from user. Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day. If no. of days>30,per day fine will be Rs 50 per day for days less than 30, Rs. 5 per day. After submitting the book, status will change from I to R. If condition of fine is true, then details will be stored into fine table. Frame the problem statement for writing PL/SQL block inline with above statement.

Objective:

To learn and understand PL/SQL code block: Use of Control structure andException.

Outcomes:

Students will be able to learn concepts PL/SQL code block: Use of Control structure andException.

Hardware Requirements:

Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.

Software Requirements:

LINUX Operating System, MySQL

Program and Output:

```
mysql> use mydatabase;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
Database changed

mysql>create table borrower(Roll_no int,Name varchar(10),dateofissue date,bookname
varchar(
20),status varchar(10));
Query OK, 0 rows affected (0.08 sec)
mysql>insert into borrower values(01,”rushikesh”,'2018-2-3’,”java',”I”);
Query OK, 1 row affected (0.04 sec)

mysql>insert into borrower values(02,”abhijit”,’2018-3-23’,”DBMS”,”I”);
Query OK, 1 row affected (0.04 sec)
```

```
mysql>insert into borrower values(03,”saurabh”,’2018-8-23’,”CN”,”I');
Query OK, 1 row affected (0.03 sec)

mysql>insert into borrower values(04,”sachin”,’2018-7-2’,”ISEE”,”I”);
Query OK, 1 row affected (0.04 sec)

mysql>select * from borrower;
+———–+———–+————––-+———–-+———+
| Roll_no | Name | dateofissue | bookname | status |
+———–+———–+————––-+———–-+———+
| 1 | rushikesh | 2018-02-03 | java | I |
| 2 | abhijit | 2018-03-23 | DBMS | I |
| 3 | saurabh | 2018-08-23 | CN | I |
| 4 | sachin | 2018-07-02 | ISEE | I |
+———–+———–+————––-+———–-+———+
4 rows in set (0.00 sec)

mysql>create table fine(Roll_no int,curdate date,fine int);
Query OK, 0 rows affected (0.10 sec)

mysql>delimiter
mysql>Create procedure AA2(In rno1 int(3),name1 varchar(30))
begin
Declare i_date date;
Declare diff int;
select dateofissue into i_date from borrower where Roll_no=rno1;
select datediff(curdate(),i_date)into diff;
select diff;
End;
Query OK, 0 rows affected (0.00 sec)

mysql>delimiter ;
mysql>call AA2(1,’java’);
+——+
| diff |
+——+
| 202 |
+——+
1 row in set (0.01 sec)
Query OK, 0 rows affected (0.01 sec)
mysql>delimiter
mysql>Create procedure SS2(In rno1 int(3),name1 varchar(30))
-> begin
-> Declare i_date date;
-> Declare diff int;
-> select dateofissue into i_date from borrower where Roll_no=rno1 and bookname=name1;
-> select datediff(curdate(), i_date) into diff;
-> select diff;
-> If diff>15 then
-> Update Borrower
-> set status='R'
-> where Roll_no=rno1 and NameofBook=name1;
-> End if;
-> End;
Query OK, 0 rows affected (0.00 sec)

mysql>delimiter ;
mysql>call ss2(4,'ISEE');
+——+
| diff |
+——+
| 53 |
+——+
1 row in set (0.01 sec)

mysql>delimiter
mysql>Create procedure RRR3(In rno1 int(3),name1 varchar(30))
-> begin
-> Declare i_date date;
-> Declare diff int;
-> Declare fine_amt int;
-> select dateofissue into i_date from borrower where Roll_no=rno1 and bookname=name1;
-> select datediff(curdate(),i_date)into diff;
-> select diff;
-> If (diff>=15 and diff<=30)then
-> SET fine_amt=diff*5;
-> insert into Fine values(rno1,CURDATE(),fine_amt);
-> Update Borrower set status='R' where Roll_no=rno1 and NameofBook=name1;
-> End if;
-> End;
Query OK, 0 rows affected (0.00 sec)

mysql>delimiter ;
mysql>call RRR3(1,"java");


+——+
| diff |
+——+
| 202 |
+——+
1 row in set (0.00 sec)
Query OK, 0 rows affected (0.00 sec)

mysql>delimiter
mysql>Create procedure RR3(In rno1 int(3),name1 varchar(30))
-> begin
-> Declare i_date date;
-> Declare diff int;
-> Declare fine_amt int;
-> select dateofissue into i_date from borrower where Roll_no=rno1 and bookname=name1;
-> select datediff(curdate(),i_date)into diff;
-> select diff;
-> If (diff>=15 and diff<=30)then
-> SET fine_amt=diff*5;
-> insert into Fine values(rno1,CURDATE(),fine_amt);
-> Update Borrower set status=’R’ where Roll_no=rno1 and NameofBook=name1;
-> End if;
-> End;
Query OK, 0 rows affected (0.00 sec)

mysql>delimiter ;
mysql>call RR3(4,’ISEE’);
+——+
| diff |
+——+
| 53 |
+——+
1 row in set (0.00 sec)
Query OK, 0 rows affected (0.00 sec)

mysql>select * from borrower;
+———+———+————-+———-+——+
| Roll_no | Name | dateofissue | bookname | status |
+———+———+————-+———-+——+
| 1 | rushikesh | 2018-02-03 | java | I |
| 2 | abhijit | 2018-03-23 | DBMS | I |
| 3 | prakiash | 2018-08-23 | CN | I |
| 4 | sachin | 2018-07-02 | ISEE | I |
+———+———+————-+———-+——+
4 rows in set (0.00 sec)

mysql>insert into borrower values(05,”sonali”,’2018-08-26’,”C”,”I”);
Query OK, 1 row affected (0.04 sec)

mysql>select * from borrower;
+———+———+————-+———-+——+
| Roll_no | Name | dateofissue | bookname | status |
+———+———+————-+———-+——+
| 1 | rushikesh | 2018-02-03 | java | I |
| 2 | abhijit | 2018-03-23 | DBMS | I |
| 3 | saurabh | 2018-08-23 | CN | I |
| 4 | sachin | 2018-07-02 | ISEE | I |
| 5 | sonali | 2018-08-26 | C | I |
+———+———+————-+———-+——+
5 rows in set (0.00 sec)

mysql>call RR3(5,’C’);
+——+
| diff |
+——+
| -2 |
+——+
1 row in set (0.00 sec)
Query OK, 0 rows affected (0.00 sec)

mysql>select * from borrower;
+———+———+————-+———-+———+
| Roll_no | Name | dateofissue | bookname | status |
+———+———+————-+———-+———+
| 1 | rushikesh | 2018-02-03 | java | I |
| 2 | abhijit | 2018-03-23 | DBMS | I |
| 3 | saurabh | 2018-08-23 | CN | I |
| 4 | sachin | 2018-07-02 | ISEE | I |
| 5 | sonali | 2018-08-26 | C | I |
+———+———+————-+———-+———+
5 rows in set (0.00 sec)

mysql>insert into borrower values(06,’saurabhki’,’2018-08-16’,’C++’,’I’);
Query OK, 1 row affected (0.02 sec)
mysql>call RR3(6,’C++’);
+——+
| diff |
+——+
| 8 |
+——+
1 row in set (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
mysql[sonalidumbre]> select * from borrower;
+———+———+————-+———-+———+
| Roll_no | Name | dateofissue | bookname | status |
+———+———+————-+———-+———+
| 1 | rushikesh| 2018-02-03 | java | I |
| 2 | abhijit | 2018-03-23 | DBMS | I |
| 3 | praksh | 2018-08-23 | CN | I |
| 4 | sachin | 2018-07-02 | ISEE | I |
| 5 | sonali | 2018-08-26 | C | I |
| 6 | saurabhki | 2018-08-16 | C++ | I |
+———+———+————-+———-+———+
6 rows in set (0.00 sec)

mysql>insert into borrower values(07,”nikita”,’2018-08-08’,”HTML”,”I”);
Query OK, 1 row affected (0.05 sec)

mysql>call RR3(7,’C++’);
+——+
| diff |
+——+
| 16 |
+——+
1 row in set (0.00 sec)

mysql>select * from borrower;
+———–+———–+—————–+———–+———+
| Roll_no | Name | dateofissue | bookname | status |
+———–+———–+—————–+———–+———+
| 1 | rushikesh | 2018-02-03 | java | I |
| 2 | abhijit | 2018-03-23 | DBMS | I |
| 3 | saurabh | 2018-08-23 | CN | I |
| 4 | sachin | 2018-07-02 | ISEE | I |
| 5 | sonali | 2018-08-26 | C | I |
| 6 | saurabhki | 2018-08-16 | C++ | I |
| 7 | nikita | 2018-08-08 | HTML | I |
+———–+———–+—————–+———–+———+
7 rows in set (0.00 sec)

mysql>delimiter
mysql>Create procedure R3(In rno1 int(3),name1 varchar(30))
-> begin
-> Declare i_date date;
-> Declare diff int;
-> Declare fine_amt int;
-> select dateofissue into i_date from borrower where Roll_no=rno1 and bookname=name1;
-> select datediff(curdate(),i_date)into diff;
-> select diff;
-> If (diff>=15 and diff<=30)then
-> SET fine_amt=diff*5;
-> insert into fine values(rno1,curdate(),fine_amt);
-> Update borrower set status=’R’ where Roll_no=rno1 and bookname=name1;
-> End if;
-> End;
Query OK, 0 rows affected (0.00 sec)

mysql>delimiter ;
mysql>call R3(7,’C++’);
+——+
| diff |
+——+
| 16 |
+——+
1 row in set (0.00 sec)
Query OK, 1 row affected (0.10 sec)

mysql>select * from borrower;
+———–+———–+—————–+———–+———+
| Roll_no | Name | dateofissue | bookname | status |
+———–+———–+—————–+———–+———+
| 1 | rushikesh | 2018-02-03 | java | I |
| 2 | abhijit | 2018-03-23 | DBMS | I |
| 3 | saurabh | 2018-08-23 | CN | I |
| 4 | sachin | 2018-07-02 | ISEE | I |
| 5 | sonali | 2018-08-26 | C | I |
| 6 | saurabhki | 2018-08-16 | C++ | I |
| 7 | nikita | 2018-08-08 | HTML | R |
+———+———+————-+———-+——+
7 rows in set (0.00 sec)

mysql>delimiter
mysql>Create procedure III3(In rno1 int(3),name1 varchar(30))
-> begin
-> Declare i_date date;
-> Declare diff int;
-> Declare fine_amt int;
-> select dateofissue into i_date from borrower where Roll_no=rno1 and bookname=name1;
-> select datediff(curdate(), i_date) into diff;
-> select diff;
-> If (diff>=15 and diff<=30)then
-> SET fine_amt=diff*5;
-> insert into fine values(rno1,curdate(),fine_amt);
-> elseif (diff>30) then
-> SET fine_amt=diff*50;
-> insert into fine values(rno1,curdate(),fine_amt);
-> End if;
-> Update borrower set status=’R’ where Roll_no=rno1 and bookname=name1;
-> End;
Query OK, 0 rows affected (0.00 sec)

mysql>delimiter ;
mysql>call III3(2,’DBMS’);
+——+
| diff |
+——+
| 154 |
+——+
1 row in set (0.00 sec)
Query OK, 1 row affected (0.07 sec)

mysql>select * from borrower;
+———+———+————-+———-+——+
| Roll_no | Name | dateofissue | bookname | status |
+———+———+————-+———-+——+
| 1 | rushikesh| 2018-02-03 | java | I |
| 2 | abhijit | 2018-03-23 | DBMS | R |
| 3 | praksh | 2018-08-23 | CN | I |
| 4 | sachini | 2018-07-02 | ISEE | I |
| 5 | sonali | 2018-08-26 | C | I |
| 6 | saurabhki | 2018-08-16 | C++ | I |
| 7 | nikita | 2018-08-08 | HTML | R |
+———+———+————-+———-+——+
7 rows in set (0.00 sec)
```

```
mysql>select * from fine;
+———–+————–+——+
| Roll_no | curdate | fine |
+———–+————–+——+
| 7 | 2018-08-24 | 80 |
| 2 | 2018-08-24 | 7700 |
+———–+————–+——+
2 rows in set (0.00 sec)
```

Conclusion:

Thus we have studied the concept of PL/SQL code block: Use of Control structure andException.

Assignment No:-6

Aim:

Cursors: Write a PL/SQL block of code using parameterized Cursor, that will merge the data available in the newly created table N_RollCall with the data available in the table O_RollCall. If the data in the first table already exist in the second table then that data should be skipped. .

Objective:

To learn and understand the concept of Cursor.

Outcomes:

Students will be able to learn concepts of Explicate and Implicate cursors.

Hardware Requirements:

Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.

Software Requirements:

LINUX Operating System, MySQL.

Program and Output:

```
Mysql> use mydatabase;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
Mysql> create table O_rollcall (rno int(3) primary key, name varchar(20),addr varchar(30));
Query OK, 0 rows affected (0.11 sec)

Mysql>insert into O_rollcall values(1,"rushikesh","sangamner");
Query OK, 1 row affected (0.03 sec)

Mysql>insert into O_rollcall values(2,"saurabh","pune");
Query OK, 1 row affected (0.02 sec)

Mysql>insert into O_rollcall values(3,"sachin","nagar");
Query OK, 1 row affected (0.03 sec)

Mysql>select * from O_rollcall;
+------+------------+-----------+
| rno | name      | addr     |
+------+------------+-----------+
| 1 | rushikesh | sangamner |
| 2 | saurabh | pune  |
| 3 | sachin | nagar |
+------+------------+-----------+
3 rows in set (0.00 sec)

Mysql> show tables;
+-------------------------+
| Tables_in_mydatabase|
+-------------------------+
| O_rollcall       |
| borrower         |
| emp1             |
| emp1_details     |
| emp2             |
| employee1        |
| fine             |
| product_details  |
| sale_details     |
| v1               |
| v2               |
+-------------------------+
11 rows in set (0.00 sec)

Mysql> insert into fine select * from O_rollcall;
Query OK, 3 rows affected, 6 warnings (0.03 sec)
Records: 3 Duplicates: 0 Warnings: 6

Mysql>select * from fine;
+----------+--------------+-------+
| Roll_no | curdate    | fine |
+----------+--------------+-------+
|       1 | 0000-00-00 |    0 |
|       2 | 0000-00-00 |    0 |
|       3 | 0000-00-00 |    0 |
+----------+--------------+-------+
3 rows in set (0.00 sec)

Mysql>create table n_rollcall (rno int(3) primary key, name varchar(20),addr varchar(30));
Query OK, 0 rows affected (0.05 sec)

Mysql>insert into n_rollcall select * from O_rollcall;
Query OK, 3 rows affected (0.02 sec)
Records: 3 Duplicates: 0 Warnings: 0

Mysql> select * from n_rollcall;
+------+------------+-----------+
| rno | name      | addr    |
+------+------------+-----------+
| 1 | rushikesh | sangamner |
|   2 | saurabh | pune  |
|   3 | sachin   | nagar |
+......+..............+............+
3 rows in set (0.00 sec)

Mysql> delete from  n_rollcall;
Query OK, 3 rows affected (0.02 sec)

Mysql>select * from n_rollcall;
Empty set (0.00 sec)

Mysql>select * from O_rollcall;
+......+..............+............+
| rno | name     | addr    |
+......+..............+............+
| 1 | rushikesh | sangamner |
|   2 | saurabh | pune  |
|   3 | sachin   | nagar |
+......+..............+............+
3 rows in set (0.00 sec)

Mysql> delimiter //
Mysql>CREATE PROCEDURE newcurr1(IN rno1 int(3))
    -> begin
    -> DECLARE rno2 int(3);
    -> DECLARE exit_loop BOOLEAN;
    -> DECLARE c1 CURSOR FOR SELECT rno FROM O_rollcall where rno>rno1;
    -> DECLARE CONTINUE HANDLER FOR NOT FOUND SET exit_loop = TRUE;
    -> OPEN c1;
    -> emp_loop: LOOP
    -> FETCH c1 INTO rno2;
    -> If not exists(select * from n_rollcall where rno=rno2) then
    -> insert into n_rollcall select * from O_rollcall where rno=rno2;
    -> End If;
    -> IF exit_loop THEN
    -> CLOSE c1;
    -> LEAVE emp_loop;
    -> END IF;
    -> END LOOP emp_loop;
    -> END
    -> //
Query OK, 0 rows affected (0.04 sec)

Mysql>call newcurr1(1);
    -> //
Query OK, 0 rows affected, 1 warning (0.04 sec)

Mysql> select * from n_rollcall;
    -> //
+......+..............+............+
| rno | name     | addr    |
+......+..............+............+
|   2 | saurabh | pune  |
|   3 | sachin   | nagar |
+------+------------+----------+
2 rows in set (0.00 sec)

Mysql>select * from O_rollcall;
    -> //
+------+------------+----------+
| rno | name      | addr    |
+------+------------+----------+
| 1 | rushikesh | sangamner |
|   2 | saurabh | pune  |
|   3 | sachin   | nagar |
+------+------------+----------+
3 rows in set (0.00 sec)

Mysql> delimiter //
Mysql> CREATE PROCEDURE newcurr22()
    ->  begin
    -> DECLARE rno1 int(3);
    -> DECLARE exit_loop BOOLEAN;
    -> DECLARE c1 CURSOR FOR SELECT rno FROM O_rollcall ;
    -> DECLARE CONTINUE HANDLER FOR NOT FOUND SET exit_loop = TRUE;
    -> OPEN c1;
    -> emp_loop: LOOP
    -> FETCH c1 INTO rno1;
    -> If not exists(select * from n_rollcall where rno=rno1) then
    -> insert into n_rollcall select * from O_rollcall where rno=rno1;
    -> End If;
    -> IF exit_loop THEN
    -> CLOSE c1;
    -> LEAVE emp_loop;
    -> END IF;
    -> END LOOP emp_loop;
    -> END
    -> //
Query OK, 0 rows affected (0.00 sec)

Mysql>call newcurr22();
    -> //
Query OK, 0 rows affected, 1 warning (0.03 sec)

Mysql>select * from n_rollcall;
    -> //
+------+------------+----------+
| rno | name      | addr    |
+------+------------+----------+
| 1 | rushikesh | sangamner |
|   2 | saurabh | pune  |
|   3 | sachin   | nagar |
+......+..............+............+
3 rows in set (0.00 sec)

Mysql> select * from O_rollcall;
    -> //
+......+..............+............+
| rno | name     | addr    |
+......+..............+............+
| 1 | rushikesh | sangamner |
|   2 | saurabh | pune  |
|   3 | sachin   | nagar |
+......+..............+............+
3 rows in set (0.00 sec)

Mysql> select * from n_rollcall;
    -> //
+......+..............+............+
| rno | name     | addr    |
+......+..............+............+
| 1 | rushikesh | sangamner |
|   2 | saurabh | pune  |
|   3 | sachin   | nagar |
+......+..............+............+
3 rows in set (0.00 sec)

Mysql> delete from n_rollcall;
    -> //
Query OK, 3 rows affected (0.03 sec)

Mysql> select * from n_rollcall;
    -> //
Empty set (0.00 sec)

Mysql>select * from O_rollcall;
    -> //
+......+..............+............+
| rno | name     | addr    |
+......+..............+............+
| 1 | rushikesh | sangamner |
|   2 | saurabh | pune  |
|   3 | sachin   | nagar |
+......+..............+............+
3 rows in set (0.00 sec)

Mysql> delimiter //
Mysql>CREATE PROCEDURE newcurr22()
    -> begin
    -> DECLARE rno1 int(3);
    -> DECLARE exit_loop BOOLEAN;
    -> DECLARE c1 CURSOR FOR SELECT rno FROM O_rollcall ;
    -> DECLARE CONTINUE HANDLER FOR NOT FOUND SET exit_loop = TRUE;
    -> OPEN c1;
    -> emp_loop: LOOP
    -> FETCH c1 INTO rno1;
    -> If not exists(select * from n_rollcall where rno=rno1) then
    -> insert into n_rollcall select * from O_rollcall where rno=rno1
    -> End If;
    -> IF exit_loop THEN
    -> CLOSE c1;
    -> LEAVE emp_loop;
    -> END IF;
    -> END LOOP emp_loop;
    -> END
    -> //
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MariaDB server version for the right syntax to use near 'End If;
IF exit_loop THEN
CLOSE c1;
LEAVE emp_loop;
 END IF;
 END LOOP emp_loop;' at line 12
Mysql>select * from n_rollcall;
    -> //
Empty set (0.00 sec)

Mysql>select * from n_rollcall;
    -> //
Empty set (0.00 sec)

Mysql>select * from O_rollcall;
    -> //
+------+-----------+-----------+
| rno | name      | addr      |
+------+-----------+-----------+
| 1 | rushikesh | sangamner |
|   2 | saurabh | pune  |
|   3 | sachin   | nagar |
+------+-----------+-----------+
3 rows in set (0.00 sec)

Mysql>
```

Conclusion:

Thus in this assignment we have studied the types of cursors.

Assignment No:-7

Aim:

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

Objective:

To learn and understand PL/SQL Stored Procedure and Stored Function.

Outcomes:

Students will be able to learn concept of how to create a Stored Procedure and Stored Function

Hardware Requirements:

Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.

Software Requirements:

LINUX Operating System, MySQL.

Program and Output:

```
Mysql> use mydatabase;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
Mysql>create table stud_marks(roll_no int(3) primary key,name varchar(20),total_marks
    -> int(10));
Query OK, 0 rows affected (0.08 sec)

Mysql>insert into stud_marks values (1,"rushikesh",1100);
Query OK, 1 row affected (0.03 sec)

Mysql> insert into stud_marks values (2,"sonali",902)
    -> ;
Query OK, 1 row affected (0.03 sec)

Mysql> insert into stud_marks values (3,"dipak",902)
    -> ;
Query OK, 1 row affected (0.02 sec)
```

```
Mysql>create table result (rno int(3),name varchar(20),class varchar(30));
Query OK, 0 rows affected (0.05 sec)

Mysql>delimiter //
mysql>Create procedure Fgrade32(rno1 int)
Begin
->Declare grade varchar(20);
->Declare mark1 int(10);
->Declare name1 varchar(50);
->Select total_marks,name into mark1,name1 from stud_marks where roll_no=rno1;
->If (mark1>=990 and mark1<=1500)then
->Set grade='Distinction';
->insert into result values(rno1,name1,grade);
->elseif (mark1>=900 and mark1<=989) then
->Set grade='First Class';
->insert into result values(rno1,name1,grade);
->

->Set grade='Higher Second Class';
->insert into result values(rno1,name1,grade);
->End if;
->select grade;
->End;
->//
Query OK, 0 rows affected (0.00 sec)

mysql>call Fgrade(1);
-> //
+————-+
| grade |
+————-+
| Distinction |
+————-+
1 row in set (0.03 sec)
Query OK, 0 rows affected (0.03 sec)

mysql>select * from result;
-> //

+——+——+————-+
| rno | name | class |
+——+——+————-+
| 1 | rushikesh Distinction |
+——+——+————-+
1 row in set (0.00 sec)

mysql>delimiter //
mysql>Create procedure Fgrade32(rno1 int)
-> Begin
-> Declare grade varchar(20);
-> Declare mark1 int(10);
-> Declare name1 varchar(50);
-> Declare Continue Handler for sqlwarning set grade='NOT FOUND';
-> Select total_marks,name into mark1,name1 from stud_marks where roll_no=rno1;
-> If (mark1>=990 and mark1<=1500)then
-> Set grade='Distinction';
-> insert into result values(rno1,name1,grade);
-> elseif (mark1>=900 and mark1<=989) then
-> Set grade='First Class';
-> insert into result values(rno1,name1,grade);
-> elseif (mark1>=825 and mark1<=899) then
-> Set grade='Higher Second Class';
-> insert into result values(rno1,name1,grade);
-> End if;
-> select grade;
-> End;
-> //
Query OK, 0 rows affected (0.00 sec)

mysql>call Fgrade32(3);
-> //
+—————-+
| grade |
+—————-+
| Distinction |
+—————-+
1 row in set (0.05 sec)
Query OK, 0 rows affected (0.05 sec)

mysql>select * from result;
-> //
+——+——-+—————-+
| rno | name | class |
+——+——-+—————-+
| 1 | rushikesh | Distinction |
| 3 | dipak | Distinction |
+——+——-+—————-+
2 rows in set (0.00 sec)

mysql>call Fgrade32(4);
-> //
+————+
| grade |
+————+
| NOT FOUND |
+————+
1 row in set (0.00 sec)
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

```
mysql>call Fgrade32(2);
-> //
+—————-+
| grade |
+—————-+
| Distinction |
+—————-+
1 row in set (0.05 sec)
Query OK, 0 rows affected (0.05 sec)

mysql>select * from result;
-> //
+——+——-+—————-+
| rno | name | class |
+——+——-+—————-+
| 1 | rushikesh | Distinction |
| 3 | dipak | Distinction |
| 2 | sonali| Distinction |
+——+——-+—————-+
3 rows in set (0.00 sec)

mysql>call Fgrade32(10);
-> //
+———–+
| grade |
+————+
| NOT FOUND |
+————+
1 row in set (0.00 sec)
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql>delimiter //
mysql>Create function Fgrade4(rno1 int) returns varchar(20)
-> DETERMINISTIC
-> Begin
-> Declare grade varchar(20);
-> Declare mark1 int(10);
-> Declare name1 varchar(50);
-> Declare Continue Handler for sqlwarning set grade='NOT FOUND';
-> Select total_marks,name into mark1,name1 from stud_marks where roll_no=rno1;
-> If (mark1>=990 and mark1<=1500)then
-> Set grade='Distinction';
-> insert into result values(rno1,name1,grade);
-> elseif (mark1>=900 and mark1<=989) then
-> Set grade='First Class';
-> insert into result values(rno1,name1,grade);
-> elseif (mark1>=825 and mark1<=899) then
-> Set grade='Higher Second Class';
-> insert into result values(rno1,name1,grade);
-> End if;
-> Return grade;
-> End;
-> //
Query OK, 0 rows affected (0.00 sec)

mysql>select Fgrade4(6);
-> //
+————————+
| Fgrade4(6) |
+————————+
| NOT FOUND |
+————————+
1 row in set, 1 warning (0.00 sec)

mysql>select Fgrade4(1);
-> //
+————————-+
| Fgrade4(1) |
+————————-+
| Distinction |
+————————-+1 row in set (0.03 sec)
```

<u>Conclusion:</u>

Thus we have studied the concept of PL/SQL Stored Procedure and Stored Function.

Assignment No:-8

Aim:

Database Trigger (All Types: Row level and Statement level triggers, Before and After Triggers). Write a database trigger on Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in Library_Audit table.

.

Objective:

To learn and understand database Trigger.

Outcomes:

Students will be able to learn concepts of database Roe Level and Statement Level Trigger.

Hardware Requirements:

Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.

Software Requirements:

LINUX Operating System, MySQL.

Program and Output:

```
Mysql> use mydatabase;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
Mysql>create table libb(lno int(10),bname varchar(10),author varchar(20),allow_days int (10));
Query OK, 0 rows affected (0.05 sec)

Mysql> create table libb_audit(lno int(10),old_all_days int (10),new_all_days int(10));
Query OK, 0 rows affected (0.06 sec)

Mysql>insert into libb values(1,"JAVA","rushikesh",11);
Query OK, 1 row affected (0.03 sec)

Mysql>insert into libb values(1,"VB","sachin",15);
Query OK, 1 row affected (0.03 sec)

Mysql>insert into libb values(3,"DBMS","dipak",12);
Query OK, 1 row affected (0.03 sec)
```

```
Mysql>insert into libb values(2,"C++","sonali",11);
Query OK, 1 row affected (0.03 sec)

Mysql>SELECT * FROM lib;
+------+-------+---------+------------+
| lno | bname | author | allow_days |
+------+-------+---------+------------+
|    1 | java  | rushikesh  |        10 |
|    1 | VB    | sachin  |        15 |
|    1 | DBMS | dipak |         12 |
|    1 | java  | sonali  |        10 |
|    1 | java  | sonali  |        10 |
+------+-------+---------+------------+
5 rows in set (0.00 sec)

Mysql>select*from libb;
+------+-------+---------+------------+
| lno | bname | author | allow_days |
+------+-------+---------+------------+
|    1 | JAVA  | rushikesh  |        11 |
|    1 | VB    | sachin  |        15 |
|    3 | DBMS | dipak |         12 |
|    2 | C++  | sonali  |        11 |
+------+-------+---------+------------+
4 rows in set (0.00 sec)

Mysql>SELECT * FROM libb_audit;
Empty set (0.00 sec)

Mysql> delimiter //
 Mysql> CREATE TRIGGER t22
    -> AFTER UPDATE ON libb
    -> FOR EACH ROW
    -> BEGIN
    -> INSERT INTO lib_audit SET lno = OLD. lno,old_all_days=OLD. allow_days,new_all_-
    -> days=NEW. allow_days;
    -> INSERT INTO libb_audit SET lno = OLD. lno,old_all_days=OLD. allow_days,new_all_-
    -> END
    -> //
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to
your MariaDB server version for the right syntax to use near '-
days=NEW. allow_days;
INSERT INTO libb_audit SET lno = OLD. lno,old_all_days=O' at line 5
Mysql>INSERT INTO libb_audit SET lno = OLD. lno,old_all_days=OLD. allow_days,new_all_-
```

```
   -> days=NEW. allow_days;
   -> END
   -> //
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to
your MariaDB server version for the right syntax to use near '-
days=NEW. allow_days;
END' at line 1
Mysql>INSERT INTO libb_audit SET lno = OLD. lno,old_all_days=OLD. allow_days,new_all days=NEW.
allow_days;
   -> END
   -> //
Query OK, 0 rows affected (0.06 sec)

mysql>delete from libb where lno=4;
-> //
Query OK, 1 row affected (0.02 sec)

mysql>select * from libb_audit;
-> //
+——+—————+—————+
| lno | old_all_days | new_all_days |
+——+—————+—————+
| 1 | 10 | 20 |
| 4 | 11 | NULL |
+——+—————+—————+
2 rows in set (0.00 sec)
```

Conclusion:

Thus we have studied the concept of database trigger.

Assignment No:-9

Aim:

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

.

Objective:

To learn and understand open source NOSQL databases:MongoDB.

Outcomes:

Students will be able to learn concepts of NOSQL databases.

Hardware Requirements:

Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.

Software Requirements:

LINUX Operating System, MongoDB.

Theory:

MongoDB is a NoSQL database intended for storing large amounts of data in document-oriented storage with dynamic schemas. NoSQL refers to a database with a data model other than the tabular format used in relational databases such as MySQL, PostgreSQL, and Microsoft SQL. MongoDB features include: full index support, replication, high availability, and auto-sharding.

Pre-Flight Check

- These instructions are intended for installing MongoDB on a single LINUXnode.
- I'll be working from a Liquid Web Self Managed LINUXserver, and I'll be logged in as root.

Step #1: Add the MongoDB Repository

```
vim /etc/yum.repos.d/mongodb.repo
```

Option A: If you are running a 64-bit system, add the following information to the file you've created, using i to insert:

```
[mongodb]
name=MongoDB
```

```
Repositorybaseurl=http://downloads-distro.mongodb.org/repo/redhat/os/x86_64/
gpgcheck=0
enabled=1
```

Then exit and save the file with the command :wq . You should see an output very similar to the following image:

Option B: If you are running a 32-bit system, add the following information to the file you've created, using i to insert:

```
[mongodb]
name=MongoDB Repository
baseurl=http://downloads-distro.mongodb.org/repo/redhat/os/i686/
gpgcheck=0
enabled=1
```

Then exit and save the file with the command :wq .

Step #2: Install MongoDB

As a matter of best practice we'll update our packages:

```
yum -y update
```

At this point, installing MongoDB is as simple as running just one command:

```
yum -y install mongodb-org mongodb-org-server
```

Step #3: Get MongoDB Running

Start-Up MongoDB

```
systemctl start mongod
```

Check MongoDB Service Status

```
systemctl status mongod
```

Start the MongoDB Service at Boot

```
systemctl enable mongod
```

Summary List of Status Statistics (Continuous)

```
mongostat
```

Summary List of Status Statistics (5 Rows, Summarized Every 2 Seconds)

```
mongostat --rowcount 5 2
```

Enter the MongoDB Command Line

```
mongo
```

By default, running this command will look for a MongoDB server listening on port 27017 on the localhost interface.

If you’d like to connect to a MongoDB server running on a different port, then use the –port option. For example, if you wanted to connect to a local MongoDB server listening on port 22222, then you’d issue the following command:

```
mongo --port 22222
```

Shutdown MongoDB

```
systemctl stop mongod
```

Steps:

1. Download MongoDB.
2. Login as root and open terminal
3. Extract the using commond "tar xvzf filename.tgz"
4. Create a directory in mongodb\bin folder as data\db using commond "mkdir data\db"
5. Now go to Bin directory and run mongod with setting the path as "./mongod --dbpath data\db"
6. open another terminal goto Bin folder and run "./mongo"

Extract the tgz and create data\db in BIN folder

```
[illegible]
```

run Mongod with setting path

```
[root@localhost bin]# ./mongo
MongoDB shell version: 3.2.[illegible]
connecting to: test
>
```

Run mongo

Conclusion:

Thus we have studied the open source NOSQL database:MongoDB.

Assignment No:-10

Aim:

Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations, SAVE method, logical operators)

.

Objective:

To learn and understand MongoDB Queries using CRUD operations.

.

Outcomes:

Students will be able to learn concepts of MongoDB Queries using CRUD operations.like SAVE method, logical operators

.

Hardware Requirements:

Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.

Software Requirements:

LINUX Operating System, MongoDB.

Program and Output:

```
[fedora@localhost ~]$ mongo 192.168.0.78/test
MongoDB shell version: 2.4.6
connecting to: 192.168.0.78/test
> db createCollection("rushikesh")
Mon Oct 8 01:52:37.095 SyntaxError: Unexpected identifier
> db.createCollection("rushikesh")
{ "ok" : 0, "errmsg" : "collection already exists", "code" : 48 }
> db.rushikesh.insert({roll_no:1,name:'rutuja',subject:'TOC',marks:99});
> db.rushikesh.insert({roll_no:2,name:’Rushikesh’,subject:'DBMS',marks:98});
> db.rushikesh.insert({roll_no:3,name:'Saurabh',subject:'ISEE',marks:98});
> db.rushikesh.insert({roll_no:4,name:'abhijit',subject:'ISEE',marks:95});
> db.rushikesh.insert({roll_no:5,name:'Dipak',subject:'CN',marks:94});
> db.rushikesh.insert({roll_no:6,name:'Sachin',subject:'SEPM',marks:99});
> db.rushikesh.insert({roll_no:7,name:'Sachin',subject:'oop',marks:90});
> db.rushikesh.insert({roll_no:8,name:'akshay',subject:'COA',marks:92});
> db.rushikesh.insert({roll_no:9,name:'nikita',subject:'ADS',marks:99});
> db.rushikesh.insert({roll_no:10,name:'viky',subject:'DSA',marks:94});
> db.rushikesh.insert({roll_no:11,name:'Arvind',subject:'DELD',marks:94});
> db.rushikesh.find();
{ "_id" : ObjectId("5bb8699b615a60a2f1c03d2a"), "rollno" : 1, "Fee" : 23000, "status" : "Paid" }
{ "_id" : ObjectId("5bb869dc615a60a2f1c03d2b"), "rollno" : 2, "Fee" : 33000, "status" : "Not Paid" }
{ "_id" : ObjectId("5bb869f5615a60a2f1c03d2c"), "rollno" : 4, "Fee" : 3000, "status" : "Not Paid" }
{ "_id" : ObjectId("5bb86a0e615a60a2f1c03d2d"), "rollno" : 4, "Fee" : 3000, "status" : "Paid" }
{ "_id" : ObjectId("5bb86a22615a60a2f1c03d2e"), "rollno" : 5, "Fee" : 45000, "status" : "Paid" }
{ "_id" : ObjectId("5bb86a36615a60a2f1c03d2f"), "rollno" : 5, "Fee" : 5000, "status" : "Paid" }
{ "_id" : ObjectId("5bb86fd0cf83c92ed1614b46"), "rollno" : 1, "name" : "rushikesh", "city" : "pune" }
{ "_id" : ObjectId("5bb87057cf83c92ed1614b47"), "rollno" : 1, "name" : "rushikesh", "subj" : "TOC",
"marks" : 78 }
{ "_id" : ObjectId("5bb8707acf83c92ed1614b48"), "rollno" : 2, "name" : "pady", "subj" : "TOC",
"marks" : 80 }
{ "_id" : ObjectId("5bb870b4cf83c92ed1614b49"), "rollno" : 32, "name" : "nikita", "subj" : "TOC",
"marks" : 88 }
{ "_id" : ObjectId("5bb870dacf83c92ed1614b4a"), "rollno" : 3, "name" : "saurabh", "subj" : "ISEE",
"marks" : 79 }
{ "_id" : ObjectId("5bb870f8cf83c92ed1614b4b"), "rollno" : 4, "name" : "dipak", "subj" : "CN",
"marks" : 55 }
{ "_id" : ObjectId("5bb8711acf83c92ed1614b4c"), "rollno" : 5, "name" : "raju", "subj" : "DBMS",
"marks" : 85 }
{ "_id" : ObjectId("5bb8712bcf83c92ed1614b4d"), "rollno" : 6, "name" : "sachin", "subj" : "DBMS",
"marks" : 85 }
{ "_id" : ObjectId("5bbaf13461cf1d70b3fe0695"), "roll_no" : 1, "name" : "rutuja", "subject" : "TOC",
"marks" : 99 }
{ "_id" : ObjectId("5bbaf15261cf1d70b3fe0696"), "roll_no" : 2, "name" : "rushikesh", "subject" :
"DBMS", "marks" : 98 }
{ "_id" : ObjectId("5bbaf19361cf1d70b3fe0697"), "roll_no" : 3, "name" : "Saurabh", "subject" :
"ISEE", "marks" : 98 }
{ "_id" : ObjectId("5bbaf1ae61cf1d70b3fe0698"), "roll_no" : 4, "name" : "abhijit", "subject" : "ISEE",
"marks" : 95 }
{ "_id" : ObjectId("5bbaf1d561cf1d70b3fe0699"), "roll_no" : 5, "name" : "Dipak", "subject" : "CN",
"marks" : 94 }
{ "_id" : ObjectId("5bbaf1f461cf1d70b3fe069a"), "roll_no" : 6, "name" : "Sachin", "subject" : "SEPM",
"marks" : 99 }
Type "it" for more
> db.rushikesh.find().pretty();
{
        "_id" : ObjectId("5bb8699b615a60a2f1c03d2a"),
        "rollno" : 1,
        "Fee" : 23000,
        "status" : "Paid"
}
{
        "_id" : ObjectId("5bb869dc615a60a2f1c03d2b"),
        "rollno" : 2,
        "Fee" : 33000,
        "status" : "Not Paid"
}
{
        "_id" : ObjectId("5bb869f5615a60a2f1c03d2c"),
        "rollno" : 4,
        "Fee" : 3000,
        "status" : "Not Paid"
}
{
        "_id" : ObjectId("5bb86a0e615a60a2f1c03d2d"),
        "rollno" : 4,
        "Fee" : 3000,
        "status" : "Paid"
}
{
        "_id" : ObjectId("5bb86a22615a60a2f1c03d2e"),
        "rollno" : 5,
        "Fee" : 45000,
        "status" : "Paid"
}
{
        "_id" : ObjectId("5bb86a36615a60a2f1c03d2f"),
        "rollno" : 5,
        "Fee" : 5000,
        "status" : "Paid"
}
{
        "_id" : ObjectId("5bb86fd0cf83c92ed1614b46"),
        "rollno" : 1,
        "name" : "rushikesh",
        "city" : "pune"
}
{
        "_id" : ObjectId("5bb87057cf83c92ed1614b47"),
        "rollno" : 1,
        "name" : "rushikesh",
        "subj" : "TOC",
        "marks" : 78
}
{
        "_id" : ObjectId("5bb8707acf83c92ed1614b48"),
        "rollno" : 2,
        "name" : "pady",
        "subj" : "TOC",
        "marks" : 80
}
{
        "_id" : ObjectId("5bb870b4cf83c92ed1614b49"),
        "rollno" : 32,
        "name" : "nikita",
        "subj" : "TOC",
        "marks" : 88
}
{
        "_id" : ObjectId("5bb870dacf83c92ed1614b4a"),
        "rollno" : 3,
        "name" : "saurabh",
        "subj" : "ISEE",
        "marks" : 79
}
{
        "_id" : ObjectId("5bb870f8cf83c92ed1614b4b"),
        "rollno" : 4,
        "name" : "dipak",
        "subj" : "CN",
        "marks" : 55
}
{
        "_id" : ObjectId("5bb8711acf83c92ed1614b4c"),
        "rollno" : 5,
        "name" : "raju",
        "subj" : "DBMS",
        "marks" : 85
}
{
        "_id" : ObjectId("5bb8712bcf83c92ed1614b4d"),
        "rollno" : 6,
        "name" : "sachin",
        "subj" : "DBMS",
        "marks" : 85
}
{
        "_id" : ObjectId("5bbaf13461cf1d70b3fe0695"),
        "roll_no" : 1,
        "name" : "rutuja",
        "subject" : "TOC",
        "marks" : 99
```

```
}
{
        "_id" : ObjectId("5bbaf15261cf1d70b3fe0696"),
        "roll_no" : 2,
        "name" : "rushikesh",
        "subject" : "DBMS",
        "marks" : 98
}
{
        "_id" : ObjectId("5bbaf19361cf1d70b3fe0697"),
        "roll_no" : 3,
        "name" : "Saurabh",
        "subject" : "ISEE",
        "marks" : 98
}
{
        "_id" : ObjectId("5bbaf1ae61cf1d70b3fe0698"),
        "roll_no" : 4,
        "name" : "abhijit",
        "subject" : "ISEE",
        "marks" : 95
}
{
        "_id" : ObjectId("5bbaf1d561cf1d70b3fe0699"),
        "roll_no" : 5,
        "name" : "Dipak",
        "subject" : "CN",
        "marks" : 94
}
{
        "_id" : ObjectId("5bbaf1f461cf1d70b3fe069a"),
        "roll_no" : 6,
        "name" : "Sachin",
        "subject" : "SEPM",
        "marks" : 99
}
Type "it" for more
>
```

Conclusion:

Thus we have studied theMongoDB Queries using CRUD operations. And Use CRUD operations, SAVE method, logical operators

Assignment No:-11

Aim:

Implement aggregation and indexing with suitable example using MongoDB.
.

Objective:

To learn and understand aggregation and indexing with suitable example using MongoDB.

Outcomes:

Students will be able to learn concepts of aggregation and indexing with suitable example using MongoDB.

Hardware Requirements:

Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.

Software Requirements:

LINUX Operating System, MongoDB.

Program and Output:

```
[fedora@localhost ~]$ mongo 192.168.0.78/test
MongoDB shell version: 2.4.6
connecting to: 192.168.0.78/test
> db createCollection("rushikesh")
Sat Oct 6 04:17:24.447 SyntaxError: Unexpected identifier
> db.createCollection("rushikesh")
{ "ok" : 0, "errmsg" : "collection already exists", "code" : 48 }
> db.rushikesh.insert({rollno:1,name:"rushikesh",city:"pune"});
>  db.rushikesh.insert({rollno:1,name:"rushikesh",subj:"TOC",marks:78});
> db.rushikesh.insert({rollno:2,name:"pady",subj:"TOC",marks:80});
> db.rushikesh.insert({rollno:32,name:"nikita",subj:"TOC",marks:88});
> db.rushikesh.insert({rollno:3,name:"saurabh",subj:"ISEE",marks:79});
> db.rushikesh.insert({rollno:4,name:"dipak",subj:"CN",marks:55});
> db.rushikesh.insert({rollno:5,name:"raju",subj:"DBMS",marks:85});
> db.rushikesh.insert({rollno:6,name:"sachin",subj:"DBMS",marks:85});
> db.studr.aggregate([{$group:{_id:"$subj",marks:{$min:"$marks"}}}]);
{ "result" : [ ], "ok" : 1 }
> db.rushikesh.aggregate([{$group:{_id:"$subj",marks:{$min:"$marks"}}}]);
```

```
{
        "result" : [
                {
                        "_id" : "DBMS",
                        "marks" : 85
                },
                {
                        "_id" : "ISEE",
                        "marks" : 79
                },
                {
                        "_id" : "CN",
                        "marks" : 55
                },
                {
                        "_id" : "TOC",
                        "marks" : 78
                },
                {
                        "_id" : null,
                        "marks" : null
                }

        ],
        "ok" : 1
}
> db.rushikesh.aggregate([{$group:{_id:"$subj",marks:{$max:"$marks"}}}]);
{
        "result" : [
                {
                        "_id" : "DBMS",
                        "marks" : 85
                },
                {
                        "_id" : "ISEE",
                        "marks" : 79
                },
                {
                        "_id" : "CN",
                        "marks" : 55
                },
                {
                        "_id" : "TOC",
                        "marks" : 88
                },
                {
```

```
                                "_id" : null,
                                "marks" : null
                        }
                ],
                "ok" : 1
        }

        > db.rushikesh.aggregate([{$group:{_id:"$subj",marks:{$sum:"$marks"}}}]);
        {
                "result" : [
                        {
                                "_id" : "DBMS",
                                "marks" : 170
                        },
                        {
                                "_id" : "ISEE",
                                "marks" : 79
                        },
                        {
                                "_id" : "CN",
                                "marks" : 55
                        },
                        {
                                "_id" : "TOC",
                                "marks" : 246
                        },
                        {
                                "_id" : null,
                                "marks" : 0
                        }

                ],
                "ok" : 1
        }
        > db.rushikesh.aggregate([{$group:{_id:"$subj",marks:{$avg:"$marks"}}}]);
        {
                "result" : [
                        {
                                "_id" : "DBMS",
                                "marks" : 85
                        },
                        {
                                "_id" : "ISEE",
                                "marks" : 79
                        },
                        {
                                "_id" : "CN",
                                "marks" : 55
                        },
                        {
                                "_id" : "TOC",
                                "marks" : 82
                        },
                        {
                                "_id" : null,
                                "marks" : 0
                        }
                ],
                "ok" : 1
        }

        > db.rushikesh.aggregate([{$group:{_id:"$subj",marks:{$last:"$marks"}}}]);
        {
                "result" : [
                        {
                                "_id" : "DBMS",
                                "marks" : 85
                        },
                        {
                                "_id" : "ISEE",
                                "marks" : 79
                        },
                        {
                                "_id" : "CN",
                                "marks" : 55
                        },
                        {
                                "_id" : "TOC",
                                "marks" : 88
                        },
                        {
                                "_id" : null,
                                "marks" : null
                        }

                ],
                "ok" : 1
        }
        >  db.rushikesh.aggregate([{$group:{_id:"$subj",marks:{$first:"$marks"}}}]);
        {
                "result" : [
                        {
                                "_id" : "DBMS",
                                "marks" : 85
```

```
                },
                {
                        "_id" : "ISEE",
                        "marks" : 79
                },
                {
                        "_id" : "CN",
                        "marks" : 55
                },
                {
                        "_id" : "TOC",
                        "marks" : 78
                },
                {
                        "_id" : null,
                        "marks" : null
                }
        ],
        "ok" : 1




> db.rushikesh.ensureIndex({rollno:1});
> db.studr.getIndexes();
[ ]
> db.rushikesh.getIndexes();
[
        {
                "v" : 1,
                "key" : {
                        "_id" : 1
                },

                "name" : "_id_",
                "ns" : "test.rushikesh"
        },
        {
                "v" : 1,
                "key" : {
                        "rollno" : 1
                },
                "name" : "rollno_1",
                "ns" : "test.rushikesh"
        }
]
> db.rushikesh.find().min({rollno:4});
```

```
{ "_id" : ObjectId("5bb869f5615a60a2f1c03d2c"), "rollno" : 4, "Fee" : 3000, "status" : "Not Paid" }
{ "_id" : ObjectId("5bb86a0e615a60a2f1c03d2d"), "rollno" : 4, "Fee" : 3000, "status" : "Paid" }
{ "_id" : ObjectId("5bb870f8cf83c92ed1614b4b"), "rollno" : 4, "name" : "dipak", "subj" : "CN",
"marks" : 55 }
{ "_id" : ObjectId("5bb86a22615a60a2f1c03d2e"), "rollno" : 5, "Fee" : 45000, "status" : "Paid" }
{ "_id" : ObjectId("5bb86a36615a60a2f1c03d2f"), "rollno" : 5, "Fee" : 5000, "status" : "Paid" }
{ "_id" : ObjectId("5bb8711acf83c92ed1614b4c"), "rollno" : 5, "name" : "raju", "subj" : "DBMS",
"marks" : 85 }
{ "_id" : ObjectId("5bb8712bcf83c92ed1614b4d"), "rollno" : 6, "name" : "sachin", "subj" : "DBMS",
"marks" : 85 }
{ "_id" : ObjectId("5bb870b4cf83c92ed1614b49"), "rollno" : 32, "name" : "nikita", "subj" : "TOC",
"marks" : 88 }
>  db.rushikesh.getIndexes();
[
        {
                "v" : 1,
                "key" : {
                        "_id" : 1
                },

                "name" : "_id_",
                "ns" : "test.rushikesh"
        },
        {
                "v" : 1,
                "key" : {
                        "rollno" : 1
                },
                "name" : "rollno_1",
                "ns" : "test.rushikesh"
        }
]
>
```

Conclusion:

Thus we have studied the concept of aggregation and indexing in MongoDB

Assignment No:-12

Aim:

Implement Map reduces operation with suitable example using MongoDB.

.

Objective:

To learn and understand Map reduces operation with suitable example using MongoDB.

.

Outcomes:

Students will be able to learn concepts Map reduces operation with suitable example using MongoDB.

.

Hardware Requirements:

Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.

Software Requirements:

LINUX Operating System, MongoDB.

Program and Output:

```
[root@localhost fedora]# mongo 192.168.0.78/test
MongoDB shell version: 2.4.6
connecting to: 192.168.0.78/test
> db.createCollection("rushikesh")
{ "ok" : 0, "errmsg" : "collection already exists", "code" : 48 }
> db.stud1.insert({rollno: 1,Fee:23000,status:'Paid'});
> db.rushikesh.insert({rollno: 1,Fee:23000,status:'Paid'});
> db.rushikesh.insert({rollno: 2,Fee:33000,status:'Not Paid'});
> db.rushikesh.insert({rollno: 4,Fee:3000,status:'Not Paid'});
> db.rushikesh.insert({rollno: 4,Fee:3000,status:'Paid'});
> db.rushikesh.insert({rollno: 5,Fee:45000,status:'Paid'});
> db.rushikesh.insert({rollno: 5,Fee:5000,status:'Paid'});
> db.rushikesh.find().pretty();
{
        "_id" : ObjectId("5bb8699b615a60a2f1c03d2a"),
        "rollno" : 1,
        "Fee" : 23000,
        "status" : "Paid"
```

```
}
{
        "_id" : ObjectId("5bb869dc615a60a2f1c03d2b"),
        "rollno" : 2,
        "Fee" : 33000,
        "status" : "Not Paid"
}
{
        "_id" : ObjectId("5bb869f5615a60a2f1c03d2c"),
        "rollno" : 4,
        "Fee" : 3000,
        "status" : "Not Paid"
}
{
        "_id" : ObjectId("5bb86a0e615a60a2f1c03d2d"),
        "rollno" : 4,
        "Fee" : 3000,
        "status" : "Paid"
}
{
        "_id" : ObjectId("5bb86a22615a60a2f1c03d2e"),
        "rollno" : 5,
        "Fee" : 45000,
        "status" : "Paid"
}
{
        "_id" : ObjectId("5bb86a36615a60a2f1c03d2f"),
        "rollno" : 5,
        "Fee" : 5000,
        "status" : "Paid"
}
>  db.stud1.mapReduce(
... ... function(){emit(this.rollno,1);},
... ... function(key,values){return Array.sum(values)},{
... ... query:{status:"Paid"},
... ... out:"total"
... ... }
... )
{
        "result" : "total",
        "timeMillis" : 529,
        "counts" : {
                "input" : 1,
                "emit" : 1,
                "reduce" : 0,
                    "output" : 1
          },
          "ok" : 1,
}
```

Conclusion:

Thus we have studied the how to Implement Map reduces operation with suitable example using MongoDB.

Assignment No:-13

Aim:

Design and Implement any 5 query using MongoDB

Objective:

To learn and understand the aMongoDB.

Outcomes:

Students will be able to learn concepts MongoDB.

Hardware Requirements:

Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.

Software Requirements:

LINUX Operating System, MySQL,MongoDB.MyConnector-Sql.jar,Myconnector-MongoDB.jar.

Program:

```
[fedora@localhost ~]$ mongo 198.68.0.78/test
MongoDB shell version: 2.4.6
connecting to: 198.68.0.78/test
Wed Oct 10 04:34:46.306 Error: couldn't connect to server 198.68.0.78:27017 at
src/mongo/shell/mongo.js:147
exception: connect failed
[fedora@localhost ~]$ ping 192.168.0.78
PING 192.168.0.78 (192.168.0.78) 56(84) bytes of data.
64 bytes from 192.168.0.78: icmp_seq=1 ttl=128 time=0.895 ms
64 bytes from 192.168.0.78: icmp_seq=2 ttl=128 time=0.581 ms
64 bytes from 192.168.0.78: icmp_seq=3 ttl=128 time=0.573 ms
64 bytes from 192.168.0.78: icmp_seq=4 ttl=128 time=0.578 ms
64 bytes from 192.168.0.78: icmp_seq=5 ttl=128 time=0.574 ms
64 bytes from 192.168.0.78: icmp_seq=6 ttl=128 time=0.561 ms
64 bytes from 192.168.0.78: icmp_seq=7 ttl=128 time=0.572 ms



[1]+  Stopped               ping 192.168.0.78
[fedora@localhost ~]$ mongo 192.168.0.78/test
MongoDB shell version: 2.4.6
```

```
connecting to: 192.168.0.78/test
> db.createCollection("stude")
{ "ok" : 1 }
> db.stude.insert({rollno:1,name:"rushikesh",city:"sangamner"});
> db.stude.find();
{ "_id" : ObjectId("5bbdbb07f865c85492fe485e"), "rollno" : 1, "name" : "rushikesh", "city" :
"sangamner" }
> db.stude.find().pretty()
{
        "_id" : ObjectId("5bbdbb07f865c85492fe485e"),
        "rollno" : 1,
        "name" : "rushikesh",
        "city" : "sangamner"
}
> db.stude.insert({rollno:2,name:"sachin",city:"jambut"});
> db.stude.insert({rollno:3,name:"dipak",city:"jamkhed"});
> db.stude.insert({rollno:4,name:"sonali",city:"otur"});
> db.stude.insert({rollno:4,name:"abhijit",city:"nashik"});
> db.stude.insert({rollno:5,name:"saurabh",city:"nagar"});
> db.stude.insert({rollno:6,name:"nikita",city:"pune"});
> db.stude.insert({rollno:7,name:"sonali",city:"mumbai"});
> db.stude.find().pretty();
{
        "_id" : ObjectId("5bbdbb07f865c85492fe485e"),
        "rollno" : 1,
        "name" : "rushikesh",
        "city" : "sangamner"
}
{
        "_id" : ObjectId("5bbdbb71f865c85492fe485f"),
        "rollno" : 2,
        "name" : "sachin",
        "city" : "jambut"
}
{
        "_id" : ObjectId("5bbdbb99f865c85492fe4860"),
        "rollno" : 3,
        "name" : "dipak",
        "city" : "jamkhed"
}
{
        "_id" : ObjectId("5bbdbbb7f865c85492fe4861"),
        "rollno" : 4,
        "name" : "sonali",
        "city" : "otur"
```

```
}
{
        "_id" : ObjectId("5bbdbbdbf865c85492fe4862"),
        "rollno" : 4,
        "name" : "abhijit",
        "city" : "nashik"
}
{
        "_id" : ObjectId("5bbdbbf5f865c85492fe4863"),
        "rollno" : 5,
        "name" : "saurabh",
        "city" : "nagar"
}
{
        "_id" : ObjectId("5bbdbc14f865c85492fe4864"),
        "rollno" : 6,
        "name" : "nikita",
        "city" : "pune"
}
{
        "_id" : ObjectId("5bbdbc3af865c85492fe4865"),
        "rollno" : 7,
        "name" : "sonali",
        "city" : "mumbai"
}
> db.stude.stats();
{
        "ns" : "test.stude",
        "count" : 8,
        "size" : 896,
        "avgObjSize" : 112,
        "numExtents" : 1,
        "storageSize" : 8192,
        "lastExtentSize" : 8192,
        "paddingFactor" : 1,
        "paddingFactorNote" : "paddingFactor is unused and unmaintained in 3.0. It remains hard
coded to 1.0 for compatibility only.",
        "userFlags" : 1,
        "capped" : false,
        "nindexes" : 1,
        "indexDetails" : {

        },
        "totalIndexSize" : 8176,
        "indexSizes" : {
                "_id_" : 8176
        },
        "ok" : 1
```

Conclusion:

Thus we have studied the concept of connectivity with java to Mysql and java to MongoDB also the jdbc and odbc driver.

Assignment No:-14

Aim:

Create simple objects and array objects using JSON.

Objective:

To learn and understand how Create simple objects and array objects using JSON.
.

Outcomes:

Students will be able to learn concepts of simple objects and array objects using JSON

Hardware Requirements:

Any CPU with Pentium Processor or similar, 256 MB RAM or more, 1 GB Hard Disk or more.

Software Requirements:

LINUX Operating System, MongoDB.

Program:

Using java:

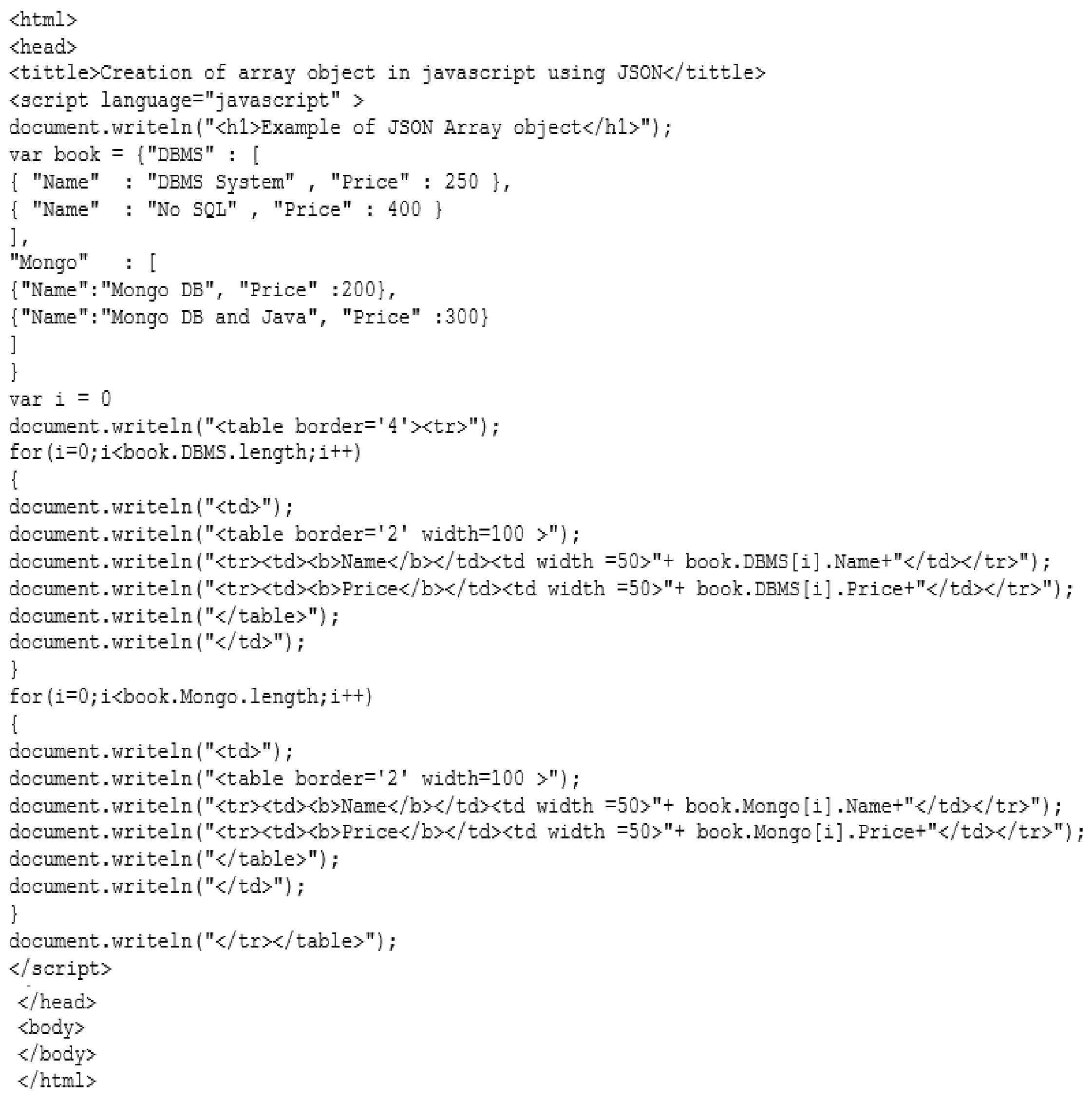
```
package json;
     import org.json.simple.*;
public class JSONOBJ{
    @SuppressWarnings("unchecked")
public static void main(String[] args) {
s JSONObject obj = new JSONObject();
       obj.put("name", "Rushikesh");
       obj.put("age",28);
       obj.put("Designation","Manager");
       obj.put("Exp",5);

JSONArray list = new JSONArray();
       list.add("8975747465");
       list.add("7219498419");
       obj.put("phone", list);

       JSONArray address=new JSONArray();
       address.add("SPCOE,Otur");
       address.add("Pune");
       obj.put("address",address);
       System.out.println("JSON Object Getting Created Successfully");
```

```
        System.out.print(obj);
          }
    }
```

Output:

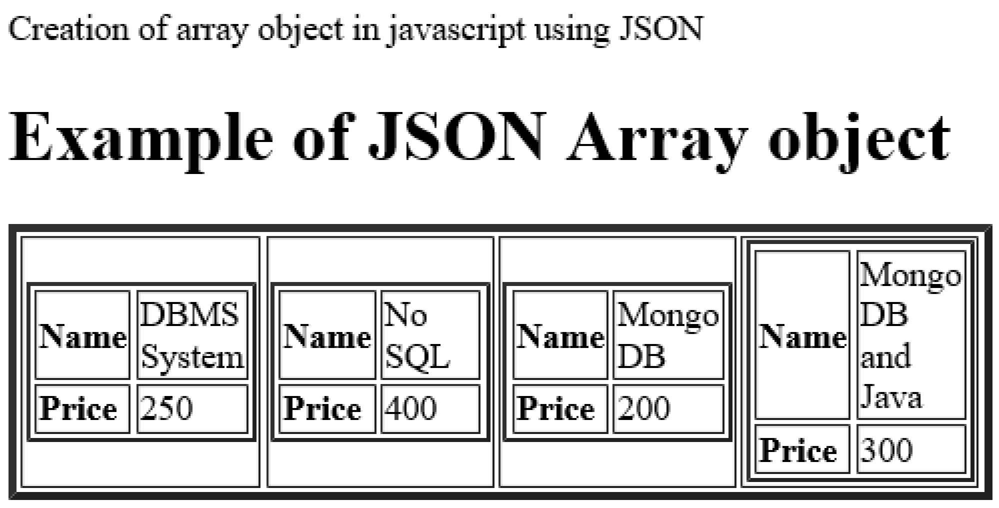
/*JSON Object Getting Created Successfully
{"Exp":5,"phone":["8975747465","9511581268"],"address":["SPCOE,Otur","Pune"],"age":28,"name":"Rushikesh","Designation":"Manager"}*/

Using html:



```
<html>
<head>
<tittle>Creation of array object in javascript using JSON</tittle>
<script language="javascript" >
document.writeln("<h1>Example of JSON Array object</h1>");
var book = {"DBMS" : [
{ "Name"  : "DBMS System" , "Price" : 250 },
{ "Name"  : "No SQL" , "Price" : 400 }
],
"Mongo"   : [
{"Name":"Mongo DB", "Price" :200},
{"Name":"Mongo DB and Java", "Price" :300}
]
}
var i = 0
document.writeln("<table border='4'><tr>");
for(i=0;i<book.DBMS.length;i++)
{
document.writeln("<td>");
document.writeln("<table border='2' width=100 >");
document.writeln("<tr><td><b>Name</b></td><td width =50>"+ book.DBMS[i].Name+"</td></tr>");
document.writeln("<tr><td><b>Price</b></td><td width =50>"+ book.DBMS[i].Price+"</td></tr>");
document.writeln("</table>");
document.writeln("</td>");
}
for(i=0;i<book.Mongo.length;i++)
{
document.writeln("<td>");
document.writeln("<table border='2' width=100 >");
document.writeln("<tr><td><b>Name</b></td><td width =50>"+ book.Mongo[i].Name+"</td></tr>");
document.writeln("<tr><td><b>Price</b></td><td width =50>"+ book.Mongo[i].Price+"</td></tr>");
document.writeln("</table>");
document.writeln("</td>");
}
document.writeln("</tr></table>");
</script>
 </head>
 <body>
 </body>
 </html>
```

Output:



Creation of array object in javascript using JSON

Example of JSON Array object

Name	DBMS System	**Name**	No SQL	**Name**	Mongo DB	**Name**	Mongo DB and Java
Price	250	**Price**	400	**Price**	200	**Price**	300

Conclusion:

Thus we have studied the concept of creatinga simple objects and array objects using JSON.