SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)
SRM Nagar, Kattankulathur-603203.

DEPARTMENT OF COMPUTER APPLICATIONS

LAB MANUAL

MC4166 DATABASE TECHNOLOGIES LABORATORY

ACADEMIC YEAR 2026-2027

(ODD SEMESTER)

FIRST SEMESTER

Prepared By

Mrs.G.S.Gayathri, Assistant Professor [O.G] / MCA

Mrs.N.Priyadharshini, Assistant Professor [O.G] / MCA

MC4166	DATABASE TECHNOLOGIES LABORATORY	L	T	P	C
		0	0	4	2

COURSE OBJECTIVES:

- To understand the process of distributing tables across multiple systems.
- To understand the process of storing, retrieving spatial and temporal data.
- To understand the process of storing, retrieving objects in a database.
- To understand the process of storing and retrieving data from a XML Database.
- To use the open source database for building a mobile application.

LIST OF EXPERIMENTS:

1. NOSQL Exercises
    a) MongoDB – CRUD operations, Indexing, Sharding
    b) Cassandra: Table Operations, CRUD Operations, CQL Types
    c) HIVE: Data types, Database Operations, Partitioning – HiveQL
2. OrientDB Graph database – OrientDB Features
3. MySQL Database Creation, Table Creation, Query
4. MySQL Replication – Distributed Databases
5. Spatial data storage and retrieval in MySQL
6. Temporal data storage and retrieval in MySQL
7. Object storage and retrieval in MySQL
8. XML Databases, XML table creation, XQuery FLWOR expression
9. Mobile Database Query Processing using open source DB (MongoDB/MySQL etc)

TOTAL: 60 PERIODS

SOFTWARE REQUIREMENTS

1. Java / Python / R / Scala
2. Oracle, MySQL, MongoDB, Casandra, Hive, OrientDB

COURSE OUTCOMES (COs):

At the end of the course, students will be able to

CO1: Design and implement advanced databases.
CO2: Use big data frameworks and tools.
CO3: Formulate complex queries using SQL.
CO4: Create an XML document and perform Xquery.
CO5: Query processing in Mobile databases using open-source tools.

CO's- PO's MAPPING

Course Outcomes	PROGRAM OUTCOMES					
	1	**2**	**3**	**4**	**5**	**6**
CO1	2	1	2	2	2	2
CO2	2	1	2	2	2	2
CO3	2	1	2	2	2	2
CO4	2	1	2	2	2	2
CO5	2	1	2	2	2	2
AVG	2	1	2	2	2	2

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

PEO1: To prepare students with comprehensive knowledge of database technologies, enabling them to analyze, design, implement, and manage database solutions for real-world applications.

PEO2: To provide students with strong foundations in relational, NoSQL, distributed, graph, spatial, temporal, object, and XML databases and equip them with the skills required for higher studies, research, and professional practice.

PEO3: To inculcate professional, ethical, and responsible practices in students while designing and deploying database systems for diverse application domains.

PEO4: To empower students with practical skills in modern database tools and open-source technologies, enabling them to work effectively as individuals or members/leaders of multidisciplinary teams with continuous learning ability.

PROGRAM OUTCOMES (POs)

After going through the two years of study, our master's in computer applications Graduates will exhibit ability to:

PO#	Programme Outcomes
PO1	**An ability to independently carry out research, investigation, and development work** to identify, analyze, and solve practical computing problems.
PO2	**An ability to write, document, and present technical information effectively** through reports, presentations, and other professional communication methods.
PO3	**An ability to demonstrate mastery in the design and development of computer applications** using appropriate programming, database, and computing techniques.
PO4	**An ability to select, adapt, and apply appropriate modern computing tools, techniques, and resources** to solve complex computing problems while understanding their limitations.
PO5	**An ability to recognize the need for continuous learning** and independently acquire new knowledge, technologies, and skills for professional development.
PO6	**An ability to function effectively as an individual and as a member or leader of a team** in diverse technical and multidisciplinary environments.

COURSE OUTCOMES:

Course Name: MC4166 DATABASE TECHNOLOGIES LABORATORY
Year of study: 2026 –2027

MC4165.1	Design and implement advanced databases
MC4165.2	Use big data frameworks and tools
MC4165.3	Formulate complex queries using SQL
MC4165.4	Create an XML document and perform Xquery
MC4165.5	Query processing in Mobile databases using open-source tools.

CO-PO Matrix:

1	2	1	2	2	2	2
2	2	1	2	2	2	2
3	2	1	2	2	2	2
4	2	1	2	2	2	2
5	2	1	2	2	2	2
AVG	2	1	2	2	2	2

Justification:

Course Outcome	Program Outcome	Value	Justification
MC4166.1	PO1	3	Students implement NoSQL database operations using MongoDB, Cassandra, and Hive and solve practical database problems through experimentation.
	PO4	3	Students use modern NoSQL tools and database technologies for CRUD operations, indexing, sharding, partitioning, and data management.
	PO5	3	Students gain practical experience with contemporary database platforms and tools such as MongoDB, Cassandra, and Hive.
MC4166.2	PO1	3	Students design and implement graph and relational database solutions using OrientDB and MySQL.
	PO3	3	Students demonstrate application-oriented database design through graph modelling, table creation, and SQL query formulation.
	PO4	3	Students apply appropriate database management tools and techniques for solving database-related computational problems.
MC4166.3	PO1	3	Students configure and implement database replication concepts to address distributed data management requirements.

	PO4	3	Students use modern database administration and replication mechanisms for distributed database environments.
	PO6	3	Replication experiments help students understand database deployment in distributed and collaborative technical environments.
MC4166.4	PO1	3	Students implement storage and retrieval mechanisms for spatial, temporal, and object data using MySQL.
	PO3	3	Students develop database solutions that handle specialized data models and application requirements.
	PO4	3	Students use appropriate MySQL features and tools to manage advanced data types and retrieval operations.
MC4166.5	PO2	2	Students develop the ability to formulate structured queries for XML data and analyze XML database requirements.
	PO3	3	Students create XML databases and formulate XQuery FLWOR expressions for structured data retrieval.
	PO4	3	Students apply XML technologies and query tools for handling semi-structured data.
MC4166.6	PO1	3	Students apply database query-processing concepts to practical mobile application requirements.
	PO3	3	Students design database-backed solutions using open-source database technologies for mobile applications.
	PO4	3	Students select and apply suitable open-source database tools for mobile database query processing.
	PO5	3	Students gain hands-on experience with open-source database technologies such as MongoDB and MySQL.
	PO6	3	Mobile database experiments encourage students to work with database technologies in application-oriented and collaborative environments.

CO-PO Average:

CO	PO1	PO2	PO3	PO4	PO5	PO6
MC4165	2	1	2	2	2	2

ASSESSMENT METHOD

i) EVALUATION PROCEDURE FOR EACH EXPERIMENTS

S.No.	Description	Mark
1.	Aim & Pre-Lab Discussion	20
2.	Observation	20
3.	Conduction and Execution	30
4.	Output & Result	10
5.	Viva	20
Total		100

INTERNAL ASSESSMENT FOR LABORATORY

S.No	Description	Mark
1.	Conduction & Execution of Experiment	25
2.	Record	10
3.	Model Test	15
	Total	**50**

TABLE OF CONTENTS

EXP NO	NAME OF THE EXPERIMENT	PAGE NO
1 A	NoSQL – MongoDB: Perform CRUD operations, create and manage indexes, and demonstrate sharding.	
1 B	NoSQL – Cassandra: Create tables and perform CRUD operations using CQL data types.	
1 C	NoSQL – Hive: Work with Hive data types, database operations, and partitioning using HiveQL.	
2	OrientDB Graph Database: Study and implement the major features of OrientDB for graph-based data management.	
3	MySQL: Create databases and tables and execute basic and advanced SQL queries.	
4	MySQL Replication: Configure and demonstrate database replication concepts for distributed database environments.	
5	Spatial Data in MySQL: Store, manipulate, and retrieve spatial data using MySQL spatial data types and functions.	
6	Temporal Data in MySQL: Store and retrieve temporal data using appropriate date, time, and temporal functions.	
7	Object Storage in MySQL: Implement storage and retrieval of object-related data using MySQL.	
8	XML Databases: Create XML-based database structures and perform data retrieval using XQuery FLWOR expressions.	
9	Mobile Database Query Processing: Implement database query processing for mobile applications using open-source databases such as MongoDB or MySQL.	

***Topic beyond the syllabus**

Ex. No:01	NOSQL Commands using MONGO DB, CASSANDRA, HIVE

A) MONGODB

AIM:

To write NOSQL QUERIES to understand the concept of Open Source Database Management System such as Mongo DB.

PROCEDURE:

Step 1: Start the MongoDB Server (Mongos). Step 2:

Start the Client (Mongod)

Step 3: Perform the MongoDB Curd Operations such as (Create, Update,Read,Delete).

Syntax: To Create/Select a Collection: USE DATABASE_NAME.

Syntax: To Insert: DB.COLLECTION_NAME.INSERT(DOCUMENT).

Syntax: To Update: DB.COLLECTION_NAME.UPDATE(<FILTER>, <UPDATE>)

Syntax: To Display/Search: DB.COLLECTION_NAME.FIND ()

Syntax: To Delete: DB.COLLECTION_NAME.REMOVE (DELETION_CRITERIA)

Step 4: Perform the MongoDB Indexing Operations is used to improve the speed of search

operations instead of searching the whole document.

Syntax: To Create Index: DB.COLLECTION_NAME.CREATE_INDEX({FIELD : VALUE })

Step 5: Perform the MongoDB Sharding Operations

Syntax TO SHARDING DB.COLLECTION_NAME.GETSHARDDISTRIBUTION()

Step 6: Perform the Deployment Operation in MongoDB

Syntax To Deployment: RS.INITIATE() to connect the other Replica machines.

QUERIES WITH EXECUTION:

Create:

> use vysya; switched
to db vysya

Insert:

```
> db.vysya.insert({
  course: "ADT",
  details: {
    lab: "6 months",
    Trainer: "Natarajan"
  },
  category: "Programming language"
})
WriteResult({ "nInserted" : 1 })
```

Update:

```
> db.vysya.update({'course':'ADT'},{$set:{'course':'Advance DataBase Technology'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

Find:

```
> db.vysya.find()
{
  "_id" : ObjectId("603fab366a8c1de1c3b4b7a1"),
  "course" : "Advance DataBase Technology",
  "details" : {
    "lab" : "6 months",
    "Trainer" : "Natarajan"
  },
  "category" : "Programming language"
}
```

SRM-VEC

Delete:

```
> db.vysya.remove({ })
WriteResult({ "nRemoved" : 1 })
```

Indexing:

```
> db.vysya.createIndex({regNo : 1})
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "ok" : 1
}
```

Sharding:

```
> use vysya switched
to db vysya
> show collections
vysya
> db.createCollection("movie");
{ "ok" : 1 }
> db.createCollection("movie1");
{ "ok" : 1 }
> db.movie1.getShardDistribution()
Collection vysya.movie1 is not sharded.
> sh.enableSharding("vysya")
{ "ok" : 1 }
> sh.shardCollection("vysya.movie1",  {"title":"Vis"})
{ "ok" : 1 }
> db.movie1.getShardDistribution()
Collection vysya.movie1 is sharded.
> sh.status()
Sharding Status
--- sharding version:
```

SRM-VEC

```
{
  "_id" : 1,
  "version" : 4,
  "minCompatibleVersion" : 4,
  "currentVersion" : 5,
  "clusterId" : ObjectId("536c30b36eaf84a8336da659")
}
shards:
{ "_id" : "shard0000", "host" : "Natz:27000" }
{ "_id" : "shard0001", "host" : "Natz:27001" }
databases:
{
  "id" : "admin",
  "partitioned" : false,
  "primary" : "config"
}
Deployment:
> rs.status();
{ "ok" : 0, "errmsg" : "not running with --replSet", "code" : 76, "codeName" :
"NoReplicationEnabled" }
> rs.initiate({
  _id: "rs0",
  members: [
    { _id: 0, host: "mongodb0:27017" },
    { _id: 1, host: "mongodb1:27017" },
    { _id: 2, host: "mongodb2:27017" }
  ]
})
> exit
> mongo mongodb://mongodb0:27017,mongodb1:27017,mongodb2:27017
> mongo 'mongodb://mongodb0,mongodb1,mongodb2/?replicaSet=rs0'
> rs.printReplicationInfo()
```

RESULT:

Thus the above Mongo DB Queries has been executed successfully.

SRM-VEC

B) CASSANDRA

AIM:

To write NOSQL QUERIES to understand the concept of Open Source Database Management System such as CASSANDRA.

PROCEDURE:

Step 1: Start the CASSANDRA Server (Cassandra) using CMD. Step
2: Start the Client (CQLSH.py) using CMD.
Step 3: Perform the Cassandra Table Operation, Curd Operation and CQL Types.
Cassandra Table Operations:

1. Create Key Space in Cassandra. CREATE KEYSPACE <identifier> WITH <properties>
2. To Create Cassandra Table, Using Create Command.
3. To Change the structure of the table, Using Alter Command.
4. To delete the existing table in Cassandra, Using Truncate Command.
5. To Insert the values in CQL, use insert command
6. The SELECT command is used to read data from Cassandra table
7. The UPDATE command is used to update the existing data in a Cassandra.
8. The DELETE command is used to delete data from Cassandra table Step 4

: Close the command prompt
Step 5 : Stop the Server

QUERIES WITH EXECUTION:

Create KeySpace:

cqlsh> CREATE KEYSPACE vysya WITH replication = {'class':'SimpleStrategy',
'replication_factor': 3};

cqlsh> describe keyspaces;

vysya1 system_auth system_distributed vysya system_schema system system_traces

cqlsh> use vysya;

Create Table:

cqlsh:vysya> CREATE TABLE VVT(Id int PRIMARY KEY, name text, city text, fees varint);

Alter Table:

cqlsh:vysya> ALTER TABLE VVT ADD email text;

View Table:

cqlsh:vysya> select * from vvt; id

| city | email | fees | name

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

cqlsh:vysya> ALTER TABLE VVT DROP email;

cqlsh:vysya> select * from vvt;

id | city | fees | name

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

Truncate Data:

cqlsh:vysya> TRUNCATE VVT;

cqlsh:vysya> INSERT INTO VVT(Id, fees, name, city) VALUES(1, 5000, 'Natarajan S', 'Namakkal');

cqlsh:vysya> select * from vvt;

id | city | fees | name

SRM-VEC

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

1 | Namakkal | 5000 | Natarajan S

Update Data:

cqlsh:vysya> UPDATE VVT SET fees = 500, name = 'Natarajan S' WHERE id = 1;

cqlsh:vysya> select * from vvt;

id | city | fees | name

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

1 | Namakkal | 500 | Natarajan S

2 | Aathur | 5000 | Rahul

3 | Salem | 5000 | Partha

Delete Data:

cqlsh:vysya> DELETE FROM VVT WHERE id = 3;

cqlsh:vysya> select * from vvt;

id | city | fees | name

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

1 | Namakkal | 500 | Natarajan S

2 | Aathur | 5000 | Rahul

Drop Table:

cqlsh:vysya> describe columnfamilies vvt

cqlsh:vysya> DROP TABLE VVT;

cqlsh:vysya> describe columnfamilies

<empty>

RESULT:

Thus the above Hive Queries has been executed successfully

SRM-VEC

C) HIVE:

AIM:

To write NOSQL QUERIES to understand the concept of Open Source Database Management System such as HIVE.

PROCEDURE:

Step 1: Start the Hadoop Cluster from sbin Folder (Run start-dfs, start-yarn).

Step 2: Start the derby node using the command(StartNetworkServer -h 0.0.0.0)

Step 3: Then Start the Hive.(Hive command)

Step 4: Hive data types are categorized in numeric types, string types, misc types, and complex types.

Step 5: Syntax to Create a Database.first we have to check weather the DB is Already Exist or Not for that

show database;

Step 6: if Not Exist Create Database Database_Name;

Step 7: Perform Some Table Operations in Hive such as Create, Alter, Drop Table. Step 8: Finally Partitioning the Hive

SRM

QUERIES WITH EXECUTION:

hive> show databases;

OK

Default

Time taken: 0.271 seconds, Fetched: 1 row(s)

hive> create database demo; OK

Time taken: 0.215 seconds

hive> create database if not exists demo; OK

Time taken: 0.107 seconds

hive> create database demo WITH DBPROPERTIES('creator' = 'Natz', 'date' = '2019-06-03');

OK

Time taken: 2.389 seconds

hive> create table demo.employee (Id int, Name string, Salary float); OK

Time taken: 0.461 seconds

hive> select * from demo.employee; OK

1 "NATARAJAN S" 30000.0

2 "SUNDAR S" 40000.0

3 "SURESH C" 50000.0

4 "MUNISH" 90000.0

SRM-VEC

```
hive> describe demo.employee; OK
id      int  Name
        string
Salary  float
Time taken: 0.215 seconds
hive> alter table employee rename to employee_data; OK
Time taken: 6.06 seconds hive>
describe employee_data; OK
id      int  Name
        string
Salary  float
Time taken: 0.275 seconds hive>
show tables;
OK
Employee Employee_data
Time taken: 0.098 seconds, Fetched: 2 row(s)
hive> alter table employee_data add columns (age int); OK
Time taken: 0.275 seconds hive>
show tables;
OK
employee employee_data
Time taken: 0.098 seconds, Fetched: 2 row(s)

hive> drop table new_employee; OK
```

SRM-VEC

Time taken: 17.5 seconds hive>
show tables;
OK
emp
employee
Time taken: 0.098 seconds

hive> drop database demo; OK
Time taken: 2.354 seconds hive>
use test;
hive> create table student (id int, name string, age int, institute string) partitioned by (course string);
OK
Time taken: 3.054 seconds
hive> describe student; OK
id int name
  string
age int
institute string
course string
Time taken: 1.054 seconds, Fetched: 10 row(s) hive>
use show;
hive> set hive.exec.dynamic.partition=true;
hive> set hive.exec.dynamic.partition.mode=nonstrict;
hive> create table stud_demo(id int, name string, age int, institute string, course string); OK

hive> load data local inpath 'd:/student_details' into table stud_demo; Loading
data to table show.stud_demo
Table show.stud_demo status: [numFiles=1, totalSize=152]

SRM-VEC

hive> insert into student_part partition(course) select id, name, age, institute, course from stud_demo;
Job ID = codegyani_20190801062015_d7649030 -£370 -47a2 -a86d -ff402d3e7de7 Total jobs = 3
Number of reduce tasks is set to 0 since there's no reduce operator

Starting Job = job_1555046592674_0017, Tracking URL = http://ubuntu64server:8088/proxy/application_1555046592674_0017/
Command = /home/codegyani/hadoop-2.7.1/bin/hadoop job -kill job_1555046592674_0017

MapReduce Job Information:
- Stage-1: number of mappers: 1, number of reducers: 0
- 2019-08-01 06:21:50,531 Stage-1 map = 0%, reduce = 0%
- 2019-08-01 06:22:51,598 Stage-1 map = 0%, reduce = 0%
- 2019-08-01 06:23:06,456 Stage-1 map = 100%, reduce = 0%, Cumulative CCU time: 10 seconds 30 cosec
- Ended Job = job_1555046592674_001

RESULT:

Thus the above Hive Queries has been executed successfully

SRM-VEC

Ex. No:02	OrientDB Graph database – OrientDB Features

AIM:

To write NOSQL QUERIES to understand the concept of Open Source Database Management System such as OrientDB Graph.

PROCEDURE:

Step1: Start the Server form the Orientdb Bin folder.

Step2: Use the respective url http://192.168.43.111:2480/studio/index.html to login to OrientDb browser.

Step3: Choose the Database and enter the username and password for the OrientDB Server.
Step4: In the Menu choose the Graph Tab.
Step5: Create a Class, Node, Edges and Insert the fields Finally connect the graph display the output.
Step6: Logout from the Browser.
Step7: Stop the Server

QUERIES WITH EXECUTION:

create class vysya extends v

create vertex vysya set name='ahmed',class='I_MCA' create

vertex vysya set name='bala',class='I_MCA' create vertex

vysya set name='partha',class='I_MCA' create vertex vysya

set name='praveen',class='I_MCA' create vertex vysya set

name='tamil',class='I_MCA' create vertex vysya set

name='ragul',class='I_MCA' create edge from #43:0 to #44:0

create edge from #43:0 to #45:0

create edge from #46:0 to #46:0

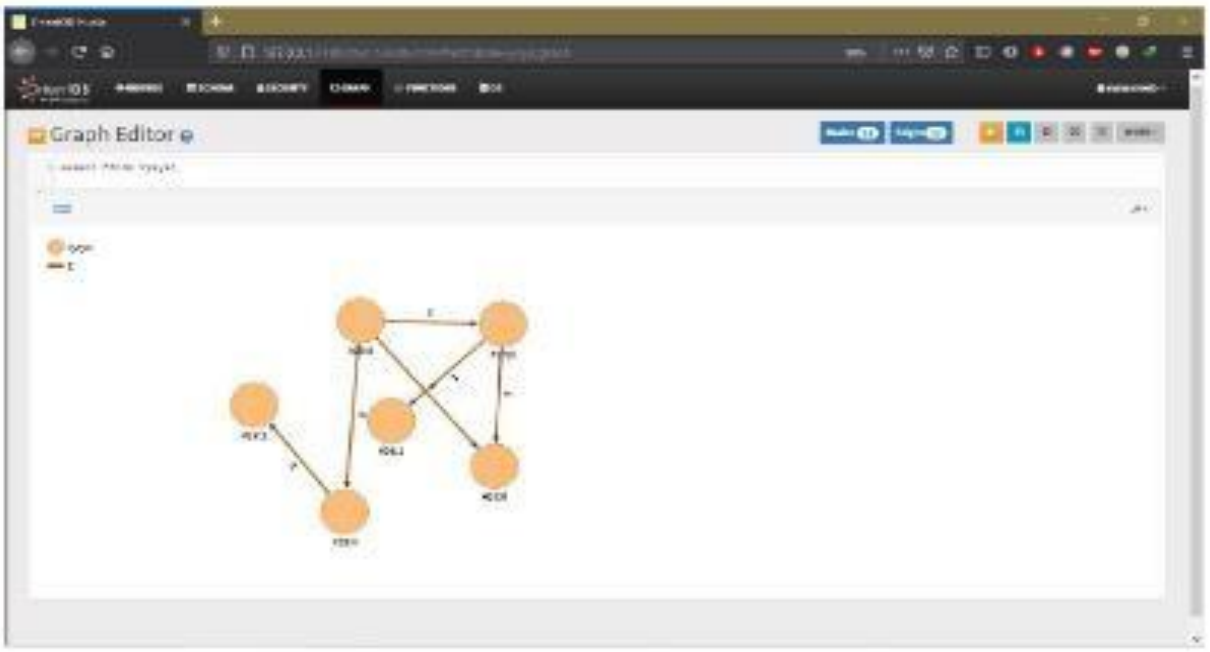
create edge from #43:1 to #44:0

create edge from #44:1 to #45:0 select

*from vysya;

SRM-

OUTPUT:



RESULT:

Thus the above Orient DB Queries has been executed successfully.

SRM-

Ex. No:03	Database and Table Creation in MYSQL

AIM:

To write a Query for MySQL Database Creation, Table Creation and Some other Queries Execution.

PROCEDURE:

Step 1: Start the MYSQL Server.
Step 2: Open MySQL Command Line Client
Step 3: Write and Execute Queries for Database Creation Step 4:
Write and Execute Queries for Table Creation
Step 5: Verify with Show Command to Check whether the Database and Table Created. Step 6: Do some other Queries Execution like insert, update, delete a record.
Step7: Close MySQL Command Line Client Step
8:Stop the MYSQL Server.

QUERIES WITH EXECUTION:

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+
3 rows in set (0.00 sec)

mysql> CREATE DATABASE Vysya;
Query OK, 1 row affected (0.00 sec)

mysql> USE Vysya;
Database changed

mysql> CREATE TABLE MCA(
   id int NOT NULL AUTO_INCREMENT,
   name varchar(45) NOT NULL,
   Dept varchar(35) NOT NULL,
   age int NOT NULL,
   PRIMARY KEY (id)
);
Query OK, 0 rows affected (0.01 sec)

mysql> SHOW TABLES;
+--------------------+
| Tables_in_vysya    |
+--------------------+
| mca                |
```

```
+----------+
1 row in set (0.00 sec)

mysql> DESCRIBE mca;
+-------+-------------+------+-----+---------+----------------+
| Field  | Type          | Null | Key | Default | Extra          |
+-------+-------------+------+-----+---------+----------------+
| id     | int(11)      | NO   | PRI | NULL    | auto_increment |
| name   | varchar(45)  | NO   |     | NULL    |                |
| Dept   | varchar(35)  | NO   |     | NULL    |                |
| age    | int(11)      | NO   |     | NULL    |                |
+-------+-------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)

mysql> ALTER TABLE MCA ADD Coll varchar(40) NOT NULL;
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> INSERT INTO mca VALUES (102, 'Natarajan', 'cs', 30, 'vysya');
Query OK, 1 row affected (0.01 sec)

mysql> INSERT INTO mca VALUES (101, 'Munish', 'cs', 32, 'vysya');
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM mca;
+---+--------+------+-----+-------+
| id  | name     | Dept | age | Coll  |

+---+--------+------+-----+-------+
| 101 | Munish   | cs   | 32  | vysya |
| 102 | Natarajan| cs   | 30  | vysya |
+---+--------+------+-----+-------+
2 rows in set (0.00 sec)
```

```
mysql> UPDATE MCA SET name = 'Munish', age = 36 WHERE id = 102; Query
OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM MCA;
+---+--------+------+-----+--------+
| id  | name       | Dept | age | Coll  |
+---+--------+------+-----+--------+
| 101 | Munish   | cs   | 32  | vysya |
| 102 | Munish   | cs   | 36  | vysya |
+---+--------+------+-----+--------+
2 rows in set (0.00 sec)

mysql> DELETE FROM MCA WHERE id=102;
Query OK, 1 row affected (0.01 sec)

mysql> DROP TABLE mca;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM MCA;
ERROR 1146 (42S02): Table 'vysya.mca' doesn't exist

mysql> CREATE DATABASE Vysya;
Query OK, 1 row affected (0.00 sec)

mysql> SHOW CREATE DATABASE Vysya;
+-------+--------------------------------+

| Database | Create Database                                  |
+-------+--------------------------------+
| vysya    | CREATE DATABASE `vysya`  |
+-------+--------------------------------+
1 row in set (0.00 sec)
```

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| test               |
| vysya              |
+--------------------+
4 rows in set (0.00 sec)

mysql> USE Vysya;
Database changed

mysql> DROP DATABASE Vysya;
Query OK, 0 rows affected (0.01 sec)
```

RESULT:

Thus the above MYSQSL Queries has been executed successfully.

Ex. No:04	Distributed Database Replication in MYSQL

AIM:

To apply MYSQL replication technique in Distributed database.

PROCEDURE:

Step 1: Software to be need to run this Replication Java
version 1.8 and above
MYSQL Server 8.05
WAMP Server
Step 2: Start the MYSQL Server and WAMP Server.
Step 3: Go to Browser in that type http://127.0.0.1/ Wamp Server Page will open. Step 4: In the bottom select tools PHPMyAdmin it will redirect to http://127.0.0.1/phpmyadmin/
Step 5: Type User Name as „Root" and Password „" and hit enter
Step 6: Create a Database, Table in MYSQL using PHPMYADMIN page. Step 7:
Need 2 MYSQL server, each running as a copy of PHPMyAdmin
Step 8: Your primary Moodle Database is referred to as the "master", the replicated server will be referred to
as the "slave"
Step 9: Take an SQL dump of your master database and restore it on your slave database Step 10: Now, on your master database log in to PHPMyAdmin and click the Replication Tab

Replication Configuration:



Replication Step 1

1. Now click the link to configure this as the master server.
2. Select "Ignore all databases, replicate:"
3. Select your Moodle database from the available list (see below).



Replication Step 2

4. Copy and paste the code this screen provides into the very bottom of your MySQL config file (my.ini on Windows) AND add a line that says `binlog_format=ROW` (this fixes an error when running an external DB enrollment sync with replication).

5. Restart the MySQL services on the master server, leave PHPMyAdmin open though.

6. Once the service has restarted, click on "Go" on the PHPMyAdmin screen.

7. You will be redirected to the Replication screen which now looks like this.



Replication Step 3

8. Lastly, we need to create a replication user so click on the link that says "Add slave replication user."

9. Create a user and password, set the host to "Any" and click "Go."

10. On the privileges screen ensure the new user has both replication permissions checked and click "Go."



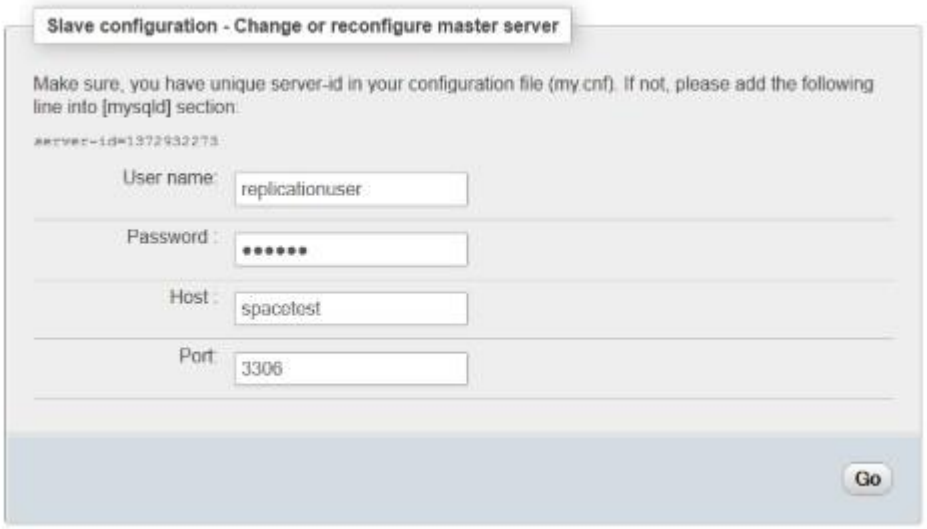
Replication Step 4

11. That's all we need to do on the master server, now let's move over to the slave.

12. From within PHPMyAdmin on the slave, click the replication tab.

13. Then click the link to configure this server as slave replication.

14. Copy the line of code that shows your new server id and paste this entire line into the MySQL config file on your slave database server.
15. Stop and start your MySQL service on the slave server.
16. Now in PHPMyAdmin enter the username of the replication user you created in step 13.
17. Enter the password and the host (the hostname of the master server or its IP Address).
18. If your default port is not 3306 then change it, chances are it uses the default port.
19. Click "Go."



Replication Step 5

20. It then takes you back to the replication screen and appears as though it's not configured but it requires a refresh.
21. So refresh the page and you will see it's configured but not running.



Replication Step 6

22. Click “Control slave” then click “Full Start.”

23. PHPMyAdmin now sits there with a Loading window, after a while this will time out, if this happens or indeed after 5 minutes nothing happens then reload the replication page again (refresh it).

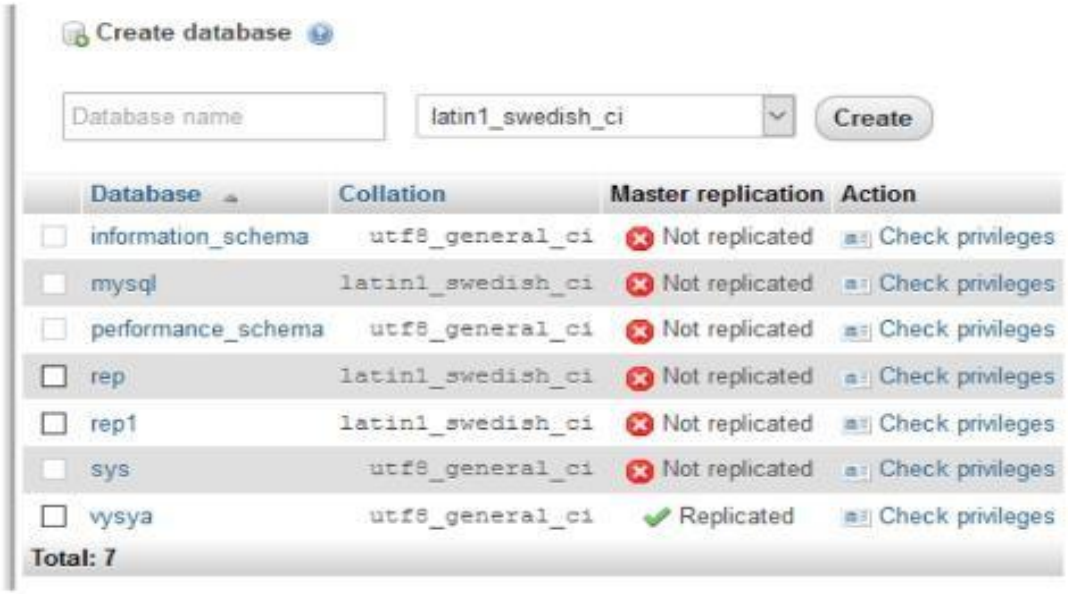
24. When the page reloads (either automatically or manually forced by you), you will see no error warnings and a message that says “Server is configured as slave.”

25. Now to check it’s working, click on the link that says “See slave status table.”



Replication Step 7

26. If everything is working then you will see a message against `Slave_IO_State` that reads “Waiting for master to send event.”



SRM-VEC

RESULT:

Thus the above program has been successfully executed

Ex.No:05	Spatial data storage and retrieval in MYSQL

AIM:

To implement Partial data storage and retrieval in MYSQL.

PROCEDURE:

Step 1: Start the MYSQL Server. Step 2:

Create a Database.

Step 3: Create a table with spatial data type. Step 4:

Insert the spatial values in the table.

Step 5: Display the output using the select command

Step 6: In the output screen Select Form Editor output will be displayed.

SRM

QUERIES WITH EXECUTION:

```
CREATE TABLE `test` (`id` INT NOT NULL AUTO_INCREMENT,`geom` GEOMETRY NULL,
PRIMARY KEY (`id`));
INSERT INTO `test`(`geom`)VALUES(st_geomfromtext ('polygon((0
0,0 3,3 0, 2 2,0 0),(1 1,1 2,2 1,2 2, 1 1))'));

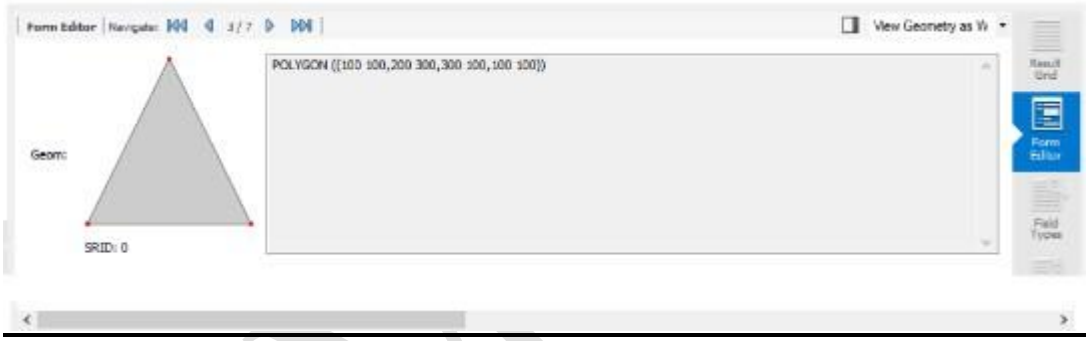
select geom from test;

INSERT INTO `test`(`geom`)VALUES(st_geomfromtext
('POLYGON((100 100,200 300,300 100, 100 100))', 0));
INSERT INTO `testw`(`geom`)VALUES(st_geomfromtext
(MULTILINESTRING('LINESTRING(0  0,1 1,2 2,3 3,4 4)')));
INSERT INTO `testw`(`geom`)VALUES(st_geomfromtext
('MULTILINESTRING((12  12, 22 22), (19 19, 32 18))'));

select geom from test;
```

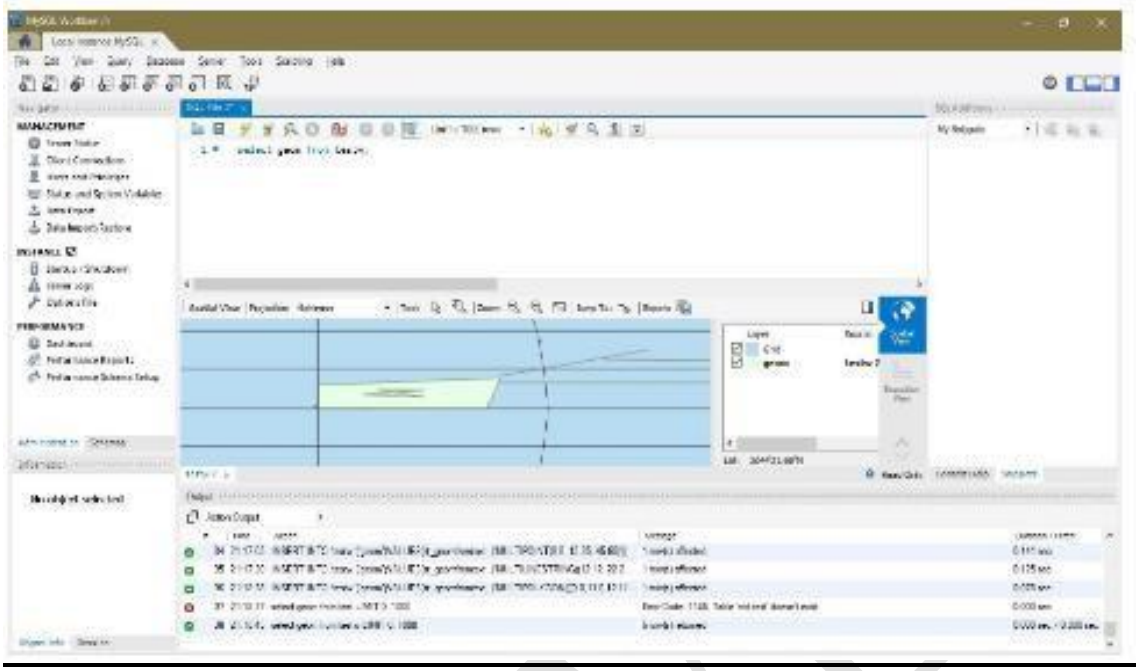
OUTPUT:


Form Editor
Navigate:
1 / 7
View Geometry as W
Geom:
SRID: 0
POLYGON ((0 0,0 3,3 0,2 2,0 0),(1 1,1 2,2 1,2 2,1 1))
Result Grid
Form Editor
Field Types


Form Editor
Navigate:
1 / 7
View Geometry as W
Geom:
SRID: 0
POLYGON ((100 100,200 300,300 100,100 100))
Result Grid
Form Editor
Field Types


Form Editor
Navigate:
7 / 7
View Geometry as W
Geom:
SRID: 0
POLYGON ((15 15,15 250,250 250,250 15,15 15))
Result Grid
Form Editor
Field Types
Query Stats

SPATIAL VIEW:



RESULT:

Thus the above program for Partial data storage and retrieval in MYSQL executed successfully.

SRM-VDP

Ex.No:06	Temporal data storage and retrieval in MYSQL

AIM:

To Create a Temporal data storage and retrieve that data in mysql.

PROCEDURE:

Step 1: Start the MYSQL Server. Step

2: Create a Database.

Step 3: Create Two Tables with the name of Customers and Orders. Step 4: Insert Some record in that tables.

Step 5: Create a Temporary Table by using the object we can create it Step 6: Show the temporary table.

SRM

QUERIES WITH EXECUTION:

```
mysql> create table customer(Cust_id  int PRIMARY KEY,cust_name  text,city
text,occupation text);
Query OK, 0 rows affected  (0.84 sec)
mysql> create table order1(order_id int PRIMARY KEY,prod_name text,price text);
Query OK, 0 rows affected  (0.89 sec)
mysql> show tables;
+- - -  -+
| Tables_in_vvt  |
+- - -  -+
| customer  |
| mca  |
| mcadet  |
| order1  |
+- - -  -+
4 rows in set (0.10 sec)
mysql> insert into customer values(1001,'NATARAJAN S','RASIPURAM','ASST_PROF');
Query OK, 1 row affected  (0.28 sec)
mysql> insert into customer values(1002,'SUNDAR','SALEM','ASST_PROF');
Query OK, 1 row affected  (0.17 sec)
mysql> insert into customer values(1003,'SURESH','ATHUR','ASST_PROF');
Query OK, 1 row affected  (0.14 sec)
mysql> insert into customer values(1004,'UDAY','SALEM','ASST_PROF');
Query OK, 1 row affected (0.34 sec)
```

mysql> insert into customer values(1005,'MUNIYASAMY','AIYOTHIYA','ASST_PROF'); Query OK, 1 row affected (0.09 sec)

```
mysql>  SELECT * FROM CUSTOMER;
+-----    -+  +-  -+  -+
| Cust_id  | cust_name  | city  | occupation  |
+-----    -+  +-  -+  -+
|  1001 | NATARAJAN  S | RASIPURAM | ASST_PROF  |
|  1002 | SUNDAR  | SALEM  | ASST_PROF |
|  1003 | SURESH  | ATHUR  | ASST_PROF |
|  1004 | UDAY  | SALEM  | ASST_PROF |
|  1005 | MUNIYASAMY  | AIYOTHIYA  | ASST_PROF  |
+-----    -+  +-  -+  -+
5 rows in set (0.00 sec)
mysql> insert into order1 values(1001,'LAPTOP','1000000'); Query
OK, 1 row affected (0.14 sec)

mysql> insert  into order1 values(1002,'MOBILE','10000'); Query
OK, 1 row affected (0.12 sec)

mysql> insert  into  order1 values(1003,'TV','15000'); Query
OK, 1 row affected (0.08 sec)

mysql> insert  into order1 values(1004,'RECHARGE','1500'); Query
OK, 1 row affected (0.15 sec)
.
mysql> insert  into order1 values(1005,'BAG','4500'); Query
OK, 1 row affected (0.08 sec)
```

```
mysql> show tables;
+- - -  -+
| Tables_in_vvt  |
+- - -  -+
| customer  |
| mca  |
| mcadet  |
| order1  |
+- - -  -+
4 rows in set (0.00 sec)

mysql> DESCRIBE customer;
+-----  -+  -+  -+  +  +  -+
| Field  | Type | Null | Key | Default  | Extra |
+-----  -+  -+  -+  +  +  -+
| Cust_id  | int  | NO  | PRI | NULL  |  |
| cust_name  | text | YES  |  | NULL  |  |
| city  | text | YES  |  | NULL  |  |
| occupation  | text | YES  |  | NULL  |  |
+-----  -+  -+  -+  +  +  -+
4 rows in set (0.00 sec)

mysql> DESCRIBE ORDER1;
+-----  +  +  +  +-  +-  +

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

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


| order_id  | int  | NO  | PRI | NULL  |  |

| prod_name  | text | YES  |  | NULL  |  |

| price  | text | YES  |  | NULL  |  |

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

```
3 rows in  set (0.04 sec)

mysql>  CREATE TEMPORARY TABLE temp_customers
->  SELECT c.cust_name,  c.city, o.prod_name, o.price
->  FROM order1 o
->  INNER JOIN customer  c ON c.cust_id  = o.order_id
->  ORDER BY o.price DESC;
Query OK, 5 rows affected (0.11 sec)
Records: 5 Duplicates:  0  Warnings:  0

mysql> show tables;
+- - -  -+
| Tables_in_vvt  |
+- - -  -+
| customer  |
| mca  |
| mcadet  |
| order1  |

+- - -  -+
4 rows in set (0.07 sec)
```

```
mysql>  select * from temp_customers;

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

| cust_name  | city  | prod_name | price  |

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


| MUNIYASAMY  | AIYOTHIYA  | BAG  | 4500  |

| SURESH  | ATHUR  | TV  | 15000  |

| UDAY  | SALEM  | RECHARGE  | 1500  |

| NATARAJAN  S | RASIPURAM  | LAPTOP  | 1000000 |

| SUNDAR  | SALEM  | MOBILE  | 10000  |

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

5 rows in set (0.00 sec)

mysql>
```

SRM

RESULT:

Thus the above program Temporary data storage and retrieval has been executed successfully

Ex.No:07	Object storage and retrieval in MYSQL

AIM:

To implement object storage and retrieval in MYSQL using Python programming language.

PROCEDURE:

Step 1: Start the SQL Server.

Step 2: Create a Database “vysya” and Table “mca” in MYSQL

Step 2: In Python import mysql.connector to connect MYSQL with Python Step 4:

Step 3: Create an object in python and insert that object in table Step 4:

Execute the SQL query.

Step 5: Fetch records from the result.

Step 6: Show the table after you make any changes in the table.

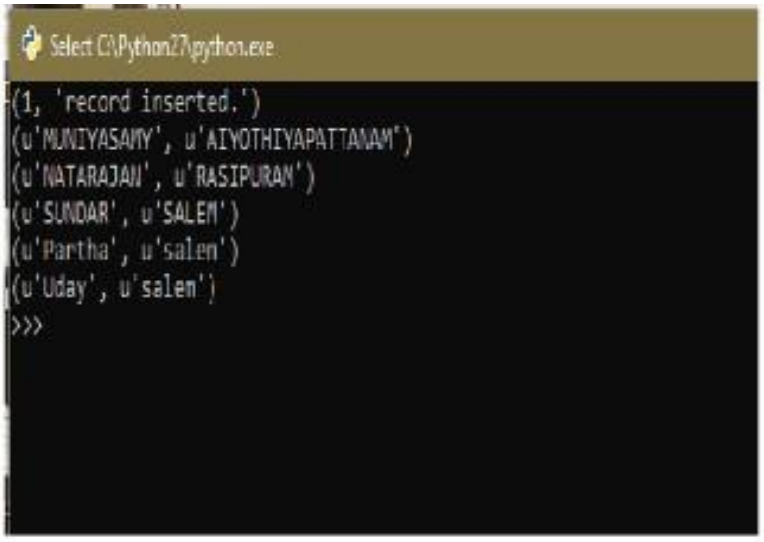
PROGRAM:

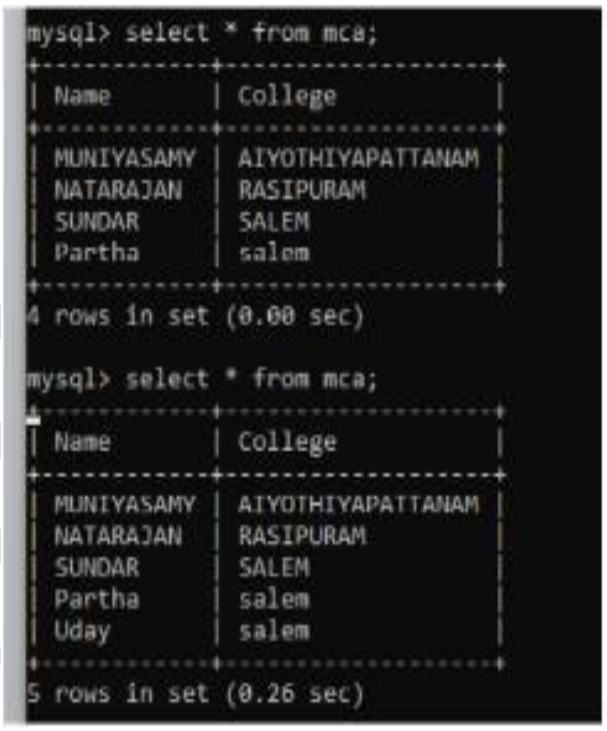
```
import mysql.connector

mydb = mysql.connector.connect( host="localhost",
    user="root",
    password="Vysya@123",
    database="vysya"
)
mycursor = mydb.cursor()
sql = "INSERT INTO mca (Name, College) VALUES (%s, %s)" val =
("Uday", "salem")
mycursor.execute(sql, val)
mydb.commit()
print(mycursor.rowcount,  "record  inserted.")
mycursor.execute("SELECT  *  FROM  mca")
myresult = mycursor.fetchall()
for x in myresult:
    print(x)
```

SRM-VEC

OUTPUT:





RESULT:

Thus the program to implement object storage and retrieval in MYSQL using Python programming language has been executed successfully

Ex. No:08	XML DATABASE CREATION AND XQUERY FLWOR EXPRESSION

AIM:

To Create a XML Databases in that create a XML Table and process the XQuery FLOWR Expressions.

PROCEDURE:

Step1: Create a XML Database in Notepad and save with an Extension filename.xml.

Step2: Open Notepad.

```
<?xml version="1.0" encoding="UTF-8"?>
<Test>
<Name>Natarajan S</Name>
<Dept>Computer Science</Dept>
<College>Vysya College</College>
<City>Salem</City>
</Test>
```

Step3: Create multiple records in a table.

Step4: Create a XML Table in that use XQuery FLWOR expression filter the record.

PROGRAM 1: (Book.xml)

```
<books>
  <book category="XML">
    <title lang="en">Learn XQuery in 24 hours</title>
    <author>Robert</author>
    <author>Peter</author>
    <college>Vysya</college>
    <year>2013</year>
    <price>50.00</price>
  </book>
  <book category="XML">
    <title lang="en">Learn XPath in 24 hours</title>
    <author>Jay Ban</author>
    <college>Vysya</college>
    <year>2010</year>
    <price>16.50</price>
  </book>
</books>
```

PROGRAM 2: (Book.xqy)

```
for $x in doc("books.xml")/books/book where
$x/price > 30
return string($x/title)
```

SRM-VEC

OUTPUT:

Learn .Net in 24 hours

Learn XQuery in 24 hour

RESULT:

Thus the above program has XML FLOWR Query has been executed successfully.

Ex. No:09	Mobile Database Query Processing using open source DB

Aim:

To write a program for Mobile Database Query Processing using open source DB using MongoDB

PROCEDURE:

Step 1: Open Eclipse and Import Java ME SDK 3.0.

Step 2: In Device Management Manually install the Java ME SDK Files.

Step 3: Open a New File JAVAME MIDLET Project, Create a project Name and configure below Emulator name.

Step 4: To establish a connection between Java ME to Mongo DB Download the Jar file Mongo DB-driver-

3.2.2. Jar and Import to Project Library Step 5:

Import the Necessary header files.

Step 6: use MONGOCLIENT to establish connection between MONGODB SERVER TO JAVA ME.

Step 7: create Collection in Mongo DB Console.

Step 8: By using find () display all the records in collections.

Program:

```
package xyz;


import     com.mongodb.MongoClient;
import com.mongodb.DB;
import     com.mongodb.DBCollection;
import com.mongodb.DBCursor;
import     javax.microedition.midlet.MIDlet;
import      javax.microedition.lcdui.Display;
import javax.microedition.lcdui.TextBox;


public class Main extends MIDlet {


    private Display display;


    public void startApp() {
        display = Display.getDisplay(this);


        // Establish connection to MongoDB
        MongoClient mongoClient = new MongoClient("localhost", 27017); DB db
        = mongoClient.getDB("test");
        DBCollection collection = db.getCollection("SSSIT");


        // Query the collection
        DBCursor cursor = collection.find();
```

SRM-VEC

```
        StringBuilder results = new StringBuilder();

        // Display the records while
        (cursor.hasNext()) {
            results.append(cursor.next().toString()).append("\n");
        }

 // Show results on the screen
        TextBox textBox = new TextBox("MongoDB Records", results.toString(), 1024,
TextBox.NO_INPUT);
        display.setCurrent(textBox);
    }

    public void pauseApp() {
        // Not used
    }

    public void destroyApp(boolean unconditional) {
        // Clean up resources
        notifyDestroyed();
    }
}
```

SRM-VEC

OUTPUT:



Result:

Thus, the above program has been successfully executed.

SRM-VDP

ADT LAB VIVA QUESTIONS

1. What is database?
2. What is DBMS?
3. What is a Database system?
4. What are the advantages of DDBMS?
5. What is the disadvantage in File Processing System?
6. Describe the three levels of data abstraction?
7. Define the "integrity rules"?
8. What is extension and intension?
9. What is System R? What are its two major subsystems?
10. How is the data structure of System R different from the relational structure?
11. What is Data Independence?
12. What is a view? How it is related to data independence?
13. What is Data Model?
14. What is E-R model?
15. What is Object Oriented model?
16. Define Replication.
17. What is an Entity type?
18. What is an Entity set?
19. What is an Extension of entity type?
20. What is Weak Entity set?
21. What is an attribute?
22. What is a Relation Schema and a Relation?
23. What is degree of a Relation?
24. What is Temporal data storage?
25. What is Relationship set?
26. What is Relationship type?
27. What is degree of Relationship type?
28. What is DDL (Data Definition Language)?
29. What is VDL (View Definition Language)?
30. What is SDL (Storage Definition Language)?
31. What is Data Storage - Definition Language?
32. What is DML (Data Manipulation Language)?
33. What is DML Compiler?
34. How Object storage and retrieval in MySQL?
35. What is DDL Interpreter?
36. What is XML Databases?
37. What is Set-at-a-time or Set-oriented?
38. What is HIVE?
39. What is Spatial DB?
40. How does Tuple-oriented relational calculus differ from domain-oriented relational calculus?
41. What is Mobile Database Query Processing?
42. What is Functional Dependency?
43. What is Lossless join property?

SRM-VEC

44. What is 1 NF (Normal Form)?
45. What is Fully Functional dependency?
46. What is 2NF?
47. What is 3NF?
48. What is BCNF (Boyce-Codd Normal Form)?
49. What is 4NF?
50. What is 5NF?
51. What is Domain-Key Normal Form?
52. What are partial, alternate,, artificial, compound and natural key?
53. What is indexing and what are the different kinds of indexing?
54. What is system catalog or catalog relation? How is better known as?
55. What is meant by query optimization?
56. What is durability in DBMS?
57. What do you mean by atomicity and aggregation?
58. What is a Phantom Deadlock?
59. What is a checkpoint and When does it occur?
60. What are the different phases of transaction?
61. What do you mean by flat file database?
62. What is "transparent DBMS"?
63. What is a query?
64. What do you mean by Correlated subquery?
65. What are the primitive operations common to all record management systems?
66. Name the buffer in which all the commands that are typed in are stored?
67. What are the unary operations in Relational Algebra?
68. Are the resulting relations of PRODUCT and JOIN operation the same?
69. What is RDBMS KERNEL?
70. Name the sub-systems of a RDBMS.
71. Which part of the RDBMS takes care of the data dictionary? How?
72. What is the job of the information stored in data-dictionary?
73. How do you communicate with an RDBMS?
74. Define SQL and state the differences between SQL and other conventional programming Languages.
75. Name the three major set of files on disk that compose a database in Oracle.
76. What is database Trigger?
77. What are stored-procedures? And what are the advantages of using them?
78. What is Storage Manager?
79. What is Buffer Manager?
80. What is Transaction Manager?
81. What is File Manager?
82. What is Authorization and Integrity manager?
83. What are stand-alone procedures?
84. What are cursors give different types of cursors?
85. What is cold backup and hot backup (in case of Oracle)?
86. What is meant by Proactive, Retroactive and Simultaneous Update.

SRM VEC