EASWARI ENGINEERING COLLEGE

(Autonomous)

Bharathi Salai, Ramapuram, Chennai-600 089

Department: _____

Name of the Lab (with code) _____

Name :

Reg No. :

Semester :

Year :

Branch :

EASWARI ENGINEERING COLLEGE

(Autonomous)

Bharathi Salai, Ramapuram, Chennai-600 089

Department: _____

PRACTICAL EXAMINATIONS_____ (Month/Year)

BONAFIDE CERTIFICATE

This is to Certify that this practical work titled_____
(code)

_____ is the bonafide work of
(Name of the Laboratory)

Mr./Miss._____ (Name of the Student)

With Registration Number_____

_____ in semester_____of year in the Department of

_____during the academic year 20____

20____.

Faculty Incharge Head of the Department

Submitted for Practical Examination held on_____/_____/_____at

Easwari Engineering College,Ramapuram,Chennai-89

Internal Examiner External Examiner

S.No	DATE	CONTENT	PAGE NO	SIGNATURE
1		NOSQLExercises. A. MongoDB B. Cassandra C. HIVE D. OrientDB		
2		MySQL Database Creation, Table Creation,Query		
3		MySQL Replication –Distributed Databases		
4		Spatial data storage and retrieval in MySQL		
5		Temporal data storage and retrieval in MySQL		
6		Object storage and retrieval in MySQL		
7		XML Databases , XML table creation, XQuery FLWOR expression		
8		Mobile Database Query Processing using open source DB (MongoDB)		

Ex.No :1	**NOSQL Commands using MONGODB,CASSANDRA,HIVE,ORIENTDB**
Date:	

A) MANGODB

AIM:

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

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 operationsinstead 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"  }
```

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: {
          "_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 MongoDB Queries has been executed successfully.

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;  //Display the Created 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
----+-------+-------+-------+-----------
(0 rows)
```

Alter Table

```
cqlsh:vysya> ALTER TABLE VVT DROP email;
cqlsh:vysya> select * from vvt;

 id | city | fees | name
----+-------+-------+---------
(0 rows)
```

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
----+-----------+-------+---------------
  1 | Namakkal | 5000 | Natarajan S
(1 rows)
```

Update Data

```
cqlsh:vysya>  UPDATE VVT SET fees= 500, Name='Natarajan S'  WHERE id=1;
cqlsh:vysya> select * from vvt;

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
(2 rows)
```

Drop Table

```
cqlsh:vysya> describe columnfamilies
vvt
cqlsh:vysya> DROP TABLE VVT;
cqlsh:vysya> describe columnfamilies
<empty>
cqlsh:vysya>
```

RESULT:

Thus the above Cassandra Queries has been executed successfully.

C) HIVE:

AIM:

To write NOSQLQUERIES 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.

QUERIES WITH EXECUTION:

```
hive> show databases;
OK
Default
Time taken: 0.271 seconds, Fetched: 1 row(s)
hive>  create database demo;
hive> show databases;
OK
Time taken: 0.215 seconds
hive> create a 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
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
id        int
Name  string
salary    float
age
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
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
```

Static Partitioning

```
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           course
course              string
# Partition Information
# col_name                  data_type         comment

Course              string
Time taken: 1.054 seconds, Fetched: 10 row(s)
```

Dynamic Partitioning

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]
hive> insert into student_part partition(course) select id, name, age, institute, course from stud_demo;
-hive >insert student_part  > partition(cinto  ourse) select id, name,  age, institute,  course > from studdemo:
e.ry ID = codegyani_20190801062015_d7649030  -£370 -47a2 -a86d -ff402d3e7de7  otal jobs = 3 nothing
Job 1 out of 3 'umber of reduce tasks is set to 0 since there's no reduce operator •tarting Job = job_
1555046592674 0017, Tracking 171,1 = http://ubuntu64server:808: proxy/application1555046592674  0017/
ill Command = /heie/codegyani/hadoop-2.7.1//bin/hadoop job -kil job_15550465• '6740017 Jadoop job
information for Stage-1: number of mappers:  1: number of reducers:  0 '019-08-01 06:21:50,531 Stage-1 map
```

= 0%, reduce = 0% '019-08-01 06:22:51,598 Stage-1 map = 0%, reduce = 0% '019-08-01 06:23:06,456 Stage-1 map = 100%, reduce = 0%, Cumulative CCU 10.03 MapReduce To cumulative CCU time: 10 seconds 30 cosec nded Job = job1555046592674 0017 Cage-4 is selected by condition resolver. Cage-3 is filtered out by condition resolver. Cage-5 is filtered out by condition resolver. oving data to: hdfs://192.168.56.123:8020/user/hive/warehouse/show.db/student. • • . . . • •

RESULT:

Thus the above Hive Queries has been executed successfully.

D) ORIENTDB:

AIM:

To write NOSQLQUERIES 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.

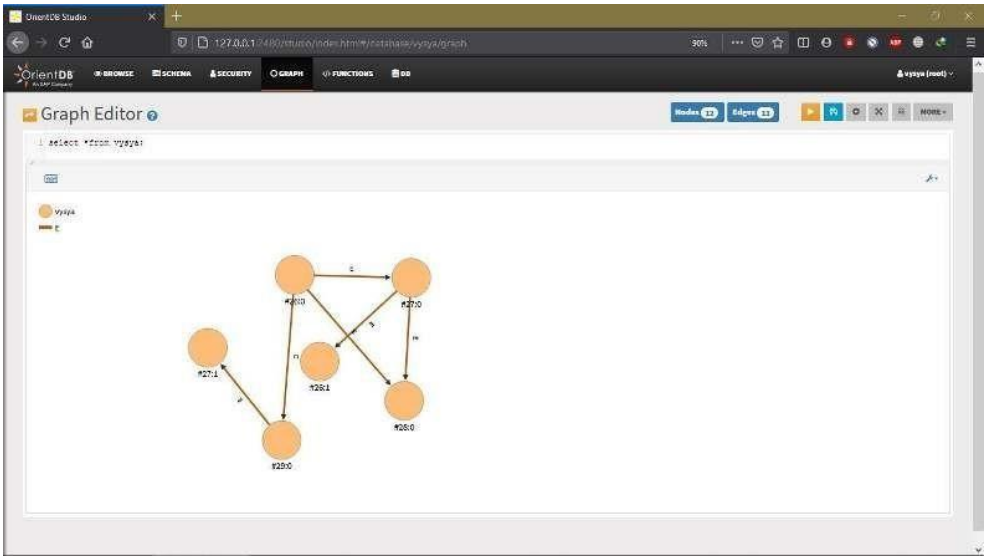
Step6: Logout from the Browser.

Step7: Stop the Server.

QUERIES

```
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;
```

OUTPUT:



RESULT:

Thus the above Orient DB Queries has been executed successfully.

Ex.No : 2	DATABASE AND TABLE CREATION IN MYSQL
Date:	

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:

Create DataBase:

```
mysql> CREATE DATABASE Vysya;
Query OK, 1 row affected (0.00 sec)
.
mysql> SHOW CREATE DATABASE Vysya;
+.....+.....+
| Database | Create Database                                                    |
+.....+.....+
| vysya    | CREATE DATABASE `vysya` /*!40100 DEFAULT CHARACTER SET latin1 */ |
+.....+.....+
1 row in set (0.00 sec)

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

Select DataBase

```
mysql> USE Vysya;
Database changed
```

Drop Database:

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

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

Create Table:

```
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));
ERROR 1046 (3D000): No database selected
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)
```

Alter Table:

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

Insert Record:

```
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)
```

Update Record:

```
mysql> use vysya;
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)
```

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

Drop Table
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

RESULT:

Thus the above MYSQSL Queries has been executed successfully.

Ex.No : 3	**Distributed databse Replication in MYSQL**
Date:	

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.

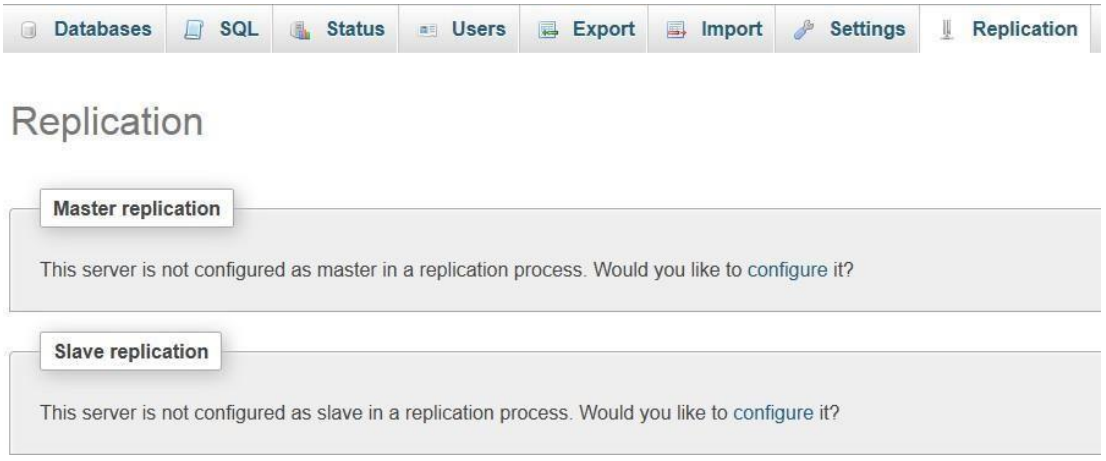
Step 7: Need 2 MYSQL server, each running as a copy of PHPMyAdmin

Step 8: Your primary Moodle Database is referred to as the "master", the replicated server wil 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

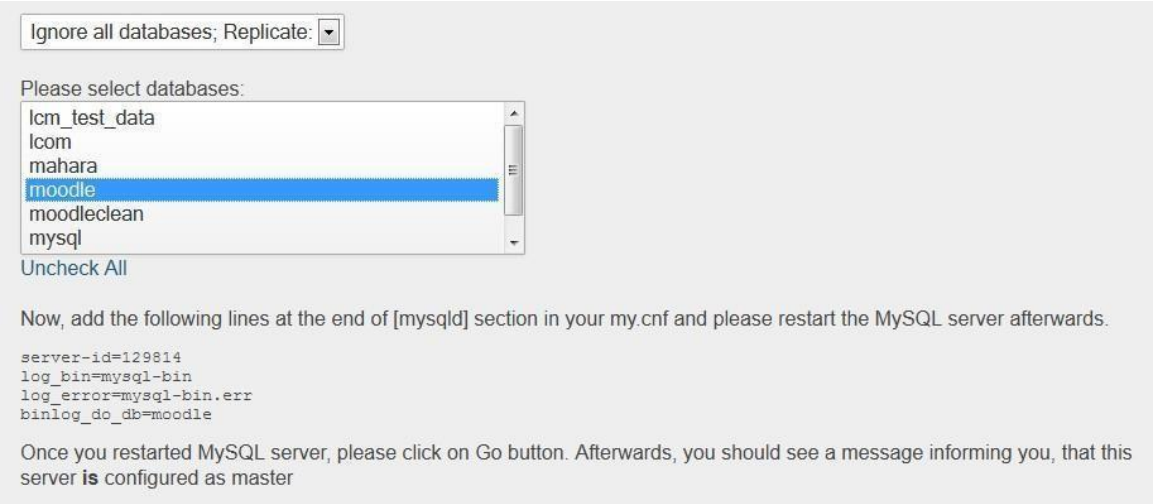
Configure the Master



MySQL Replication **Step 1**

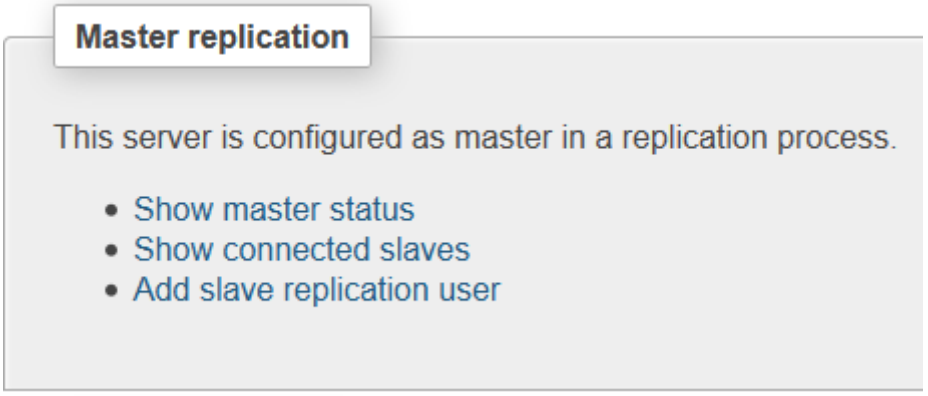
1. Now click the link to configure this as the master server

2. Select “Ignore al databases, replicate:”
3. Select your Moodle database from the available list (see below)



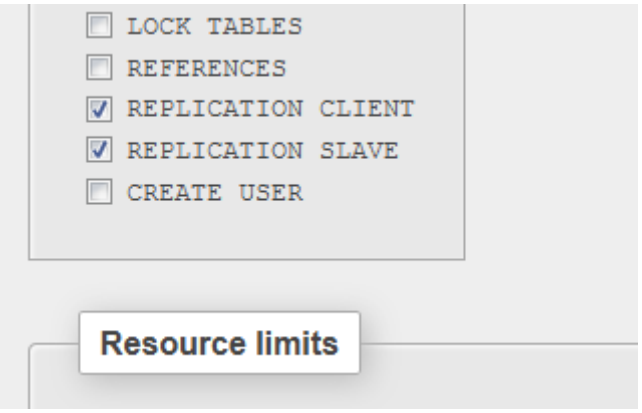
Replication **Step 2**

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



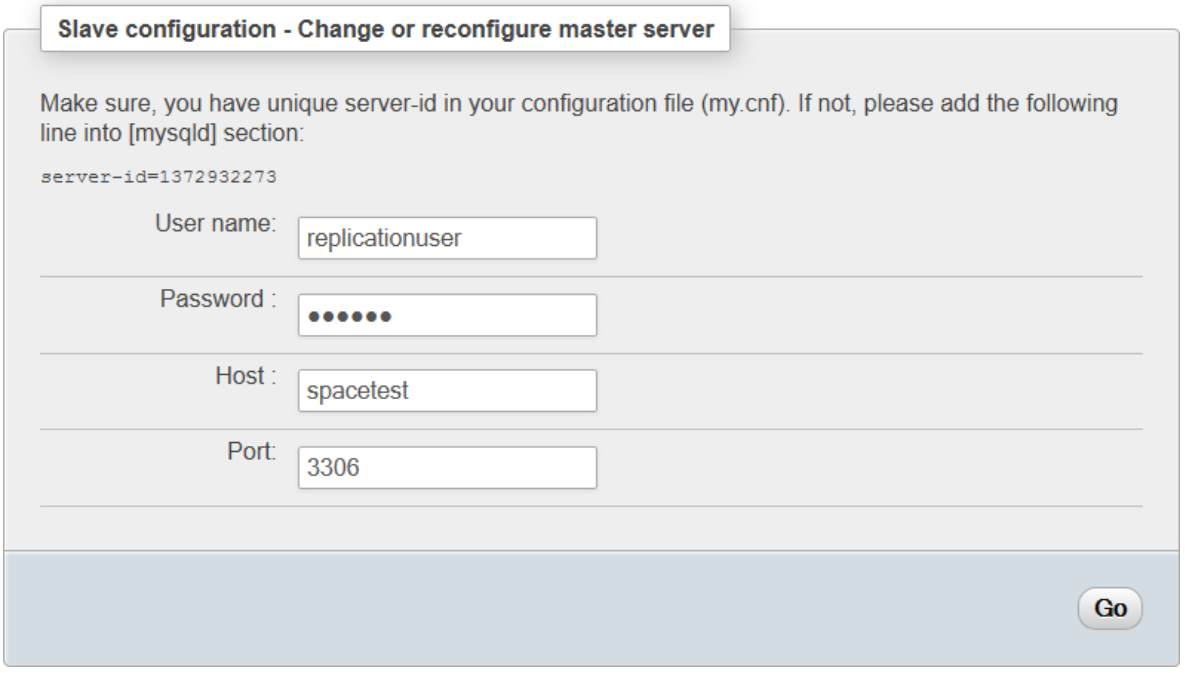
Replication **Step 3**

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. **<u>That's all we need to do on the master server, now lets move over to the slave</u>**
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

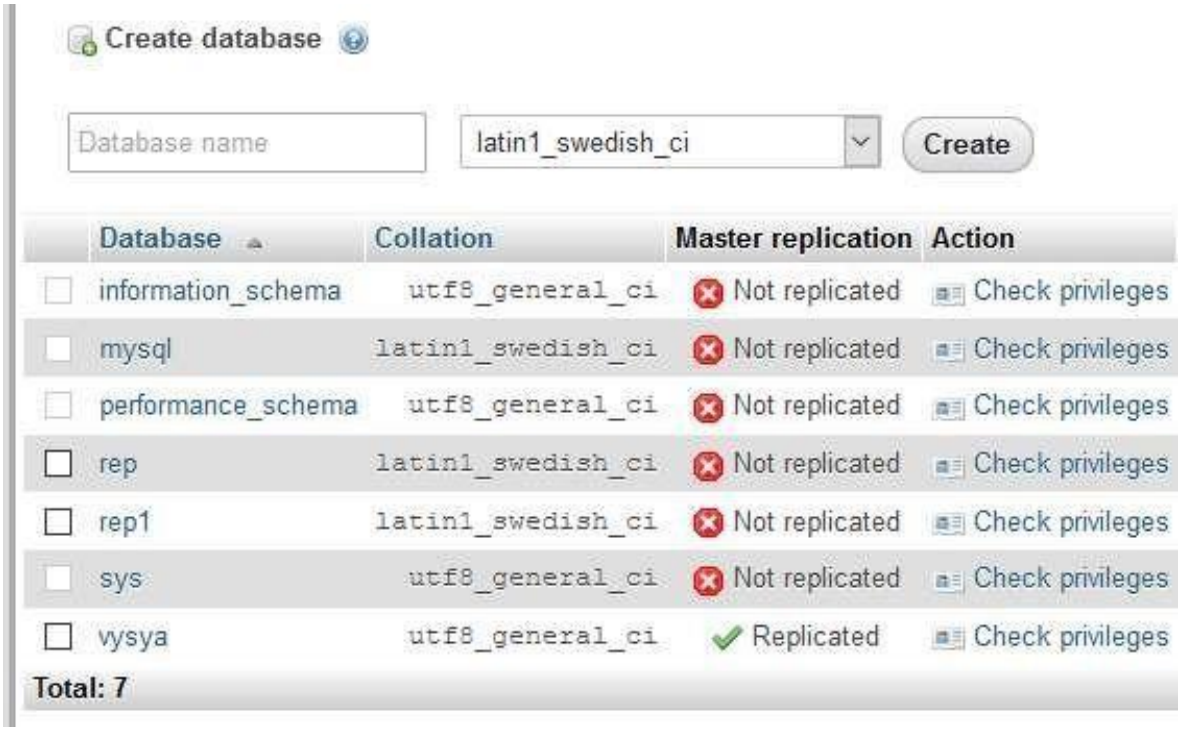
20. It then takes you back to the replication screen and appears as though it‟s not configured but it requires a refresh
21. So refresh the page and you wil see it‟s configured but not running

Replication Step 6

22. Click "Control slave" then click "Ful 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 wi**l** see a message against Slave_IO_State that reads “Waiting for master to send event”



RESULT:

Thus the above program has been successfully executed.

Ex.No :4	**Spatial Data Storage and Retrieval in MYSQL**
Date:	

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.

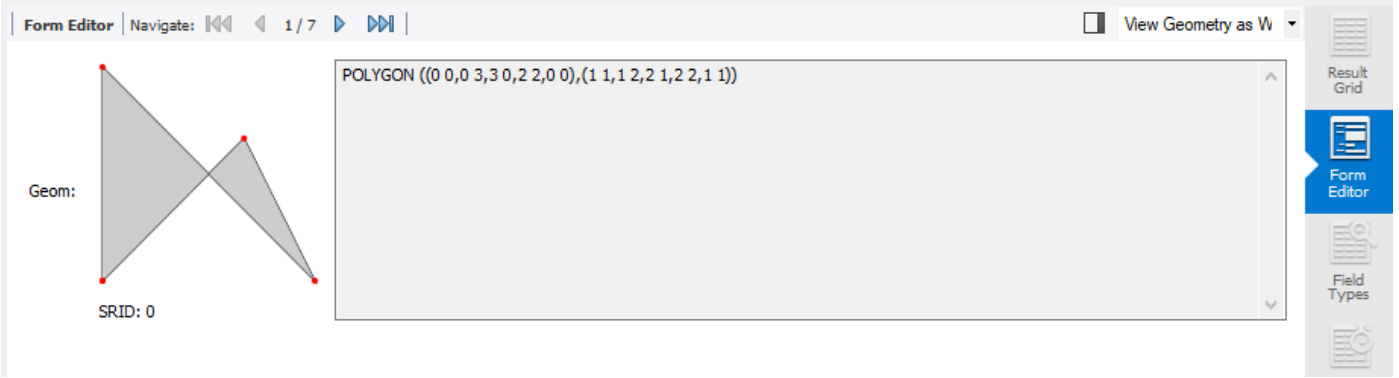
Step 5: Display the output using the select command

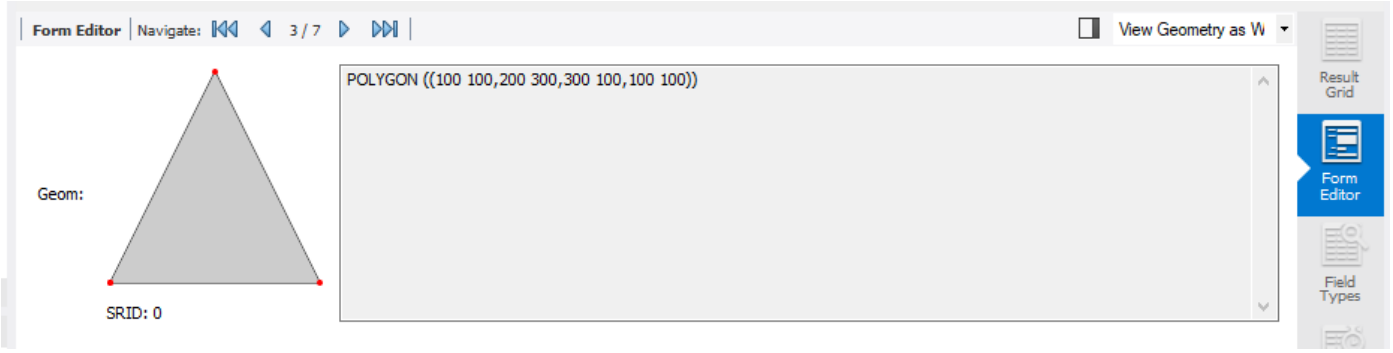
Step 6: In the output screen Select Form Editor output will be displayed.

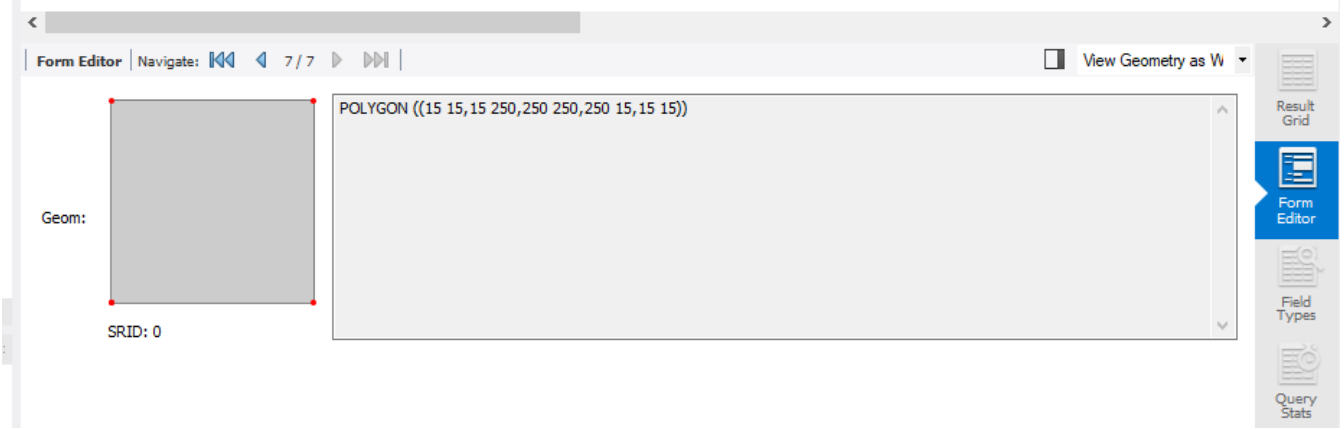
QUERIES WITH EXECUTION:

```
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;
```

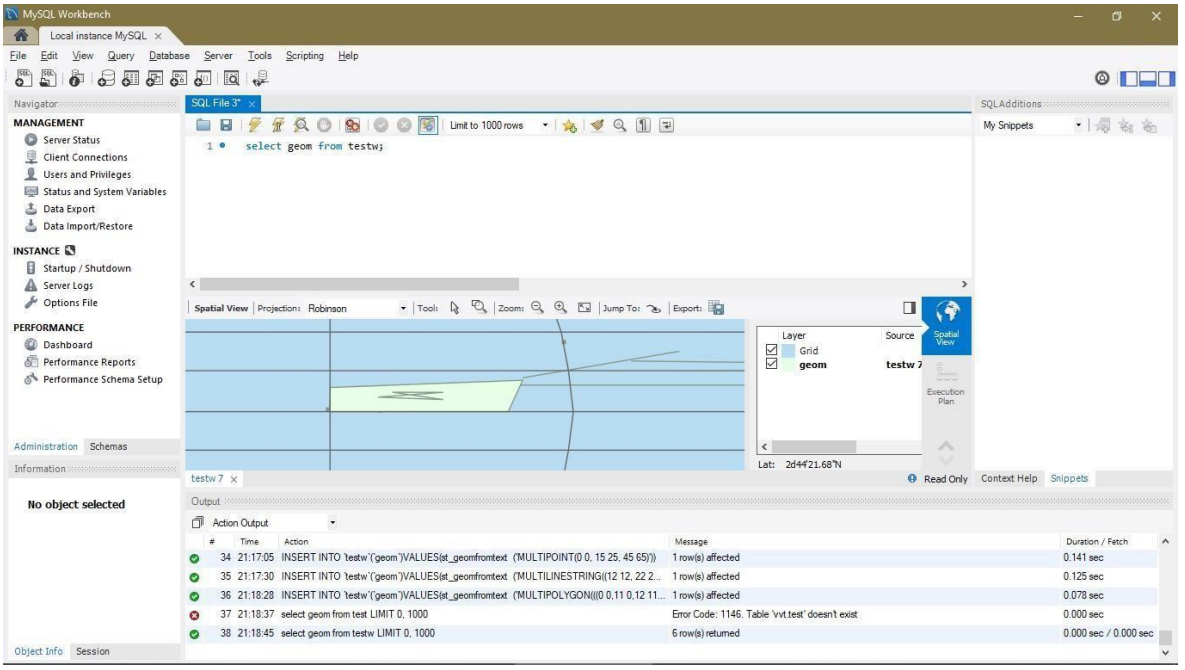
Output:


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:
3 / 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.

Ex.No :5	TEMPORAL DATA STORAGE AND RETRIEVAL IN MYSQL
Date:	

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.

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>
```

RESULT:

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

Ex.No :6	**OBJECT STORAGE AND RETRIEVAL IN MYSQL**
Date:	

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)
```

OUTPUT:

```
Select C:\Python27\python.exe
(1, 'record inserted.')
(u'MUNIYASAMY', u'AIYOTHIYAPATTANAM')
(u'NATARAJAN', u'RASIPURAM')
(u'SUNDAR', u'SALEM')
(u'Partha', u'salem')
(u'Uday', u'salem')
>>>
```

```
mysql> select * from mca;
+------------+-------------------+
| Name       | College           |
+------------+-------------------+
| MUNIYASAMY | AIYOTHIYAPATTANAM |
| NATARAJAN  | RASIPURAM         |
| SUNDAR     | SALEM             |
| Partha     | salem             |
+------------+-------------------+
4 rows in set (0.00 sec)

mysql> select * from mca;
+------------+-------------------+
| Name       | College           |
+------------+-------------------+
| MUNIYASAMY | AIYOTHIYAPATTANAM |
| NATARAJAN  | RASIPURAM         |
| SUNDAR     | SALEM             |
| Partha     | salem             |
| Uday       | salem             |
+------------+-------------------+
5 rows in set (0.26 sec)
```

RESULT:

Thus the above program for object creation and retrieval in MYSQL executed successfully using python programming language

Ex.No :7	XML DATABASE CREATION AND XQUERY FLWOR EXPRESSION
Date:	

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)

```
<?xml version="1.0" encoding="UTF-8"?>
<books>

  <book category="ADT">
    <title lang="en">Learn ADT in 24 Hours</title>
    <author>Robert</author>
    <college>Vysya</college>
    <year>2005</year>
    <price>30.00</price>
  </book>

  <book category="DOTNET">
    <title lang="en">Learn .Net in 24 hours</title>
    <author>Peter</author>
    <college>Vysya</college>
    <year>2011</year>
    <price>40.50</price>
  </book>

  <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(books.xqy)

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

OUTPUT:

Learn .Net in 24 hours
Learn XQuery in 24 hours

RESULT:

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

Ex.No :8	MOBILE DATABASE QUERY PROCESSING USING OPENSOURCE DB
Date:	

AIM:

The goal of this task is to connect to a MongoDB database using Python, insert some sample data into a collection, and then retrieve and display the inserted data.

PROCEDURE:

1. **Connect to MongoDB**:
   - First, we connect to the MongoDB server running on your computer (localhost).
2. **Select Database and Collection**:
   - After connecting, we choose the database where we want to store data.
   - Within the database, we choose a collection where we will insert and query the data.
3. **Insert Data**:
   - We create some sample data (like names and ages) in a list and insert it into the collection.
4. **Retrieve Data**:
   - We then query the collection to get all the documents (data) we have inserted.
5. **Display Data**:
   - Finally, we print out the retrieved data to see the results.

PROGRAM:

```
from pymongo import MongoClient

# Step 1: Connect to the MongoDB server on localhost at port 27017
client = MongoClient('localhost', 27017)

# Step 2: Access a specific database (db_name) within the MongoDB instance
db = client['db_name']

# Step 3: Access a specific collection (collection_name) within the database
collection = db['collection_name']

# Sample data to insert into the collection
sample_data = [
    {"name": 'john doe', 'age': 25},  # Document 1
    {"name": 'jame', 'age': 44}       # Document 2
]

# Step 4: Insert multiple records into the collection using insert_many
collection.insert_many(sample_data)

# Step 5: Query all records in the collection
records = collection.find()

# Step 6: Iterate through the query result and print each document
for record in records:
    print(record)
```

OUTPUT:

```
{'_id': ObjectId('...'), 'name': 'john doe', 'age': 25}
{'_id': ObjectId('...'), 'name': 'jame', 'age': 44
```

RESULT:

Thus the above program has been successfully executed.