ASSIGNMENT NO 2

Title: Design and Develop SQL DDL statements which demonstrate the use of SQL objects such as Table, View, Index, Sequence, Synonym

Objectives: To understand and demonstrate DDL statements on various SQL objects

Outcomes: 1) Students will be able to learn and understand various DDL queries like create, drop, truncate.

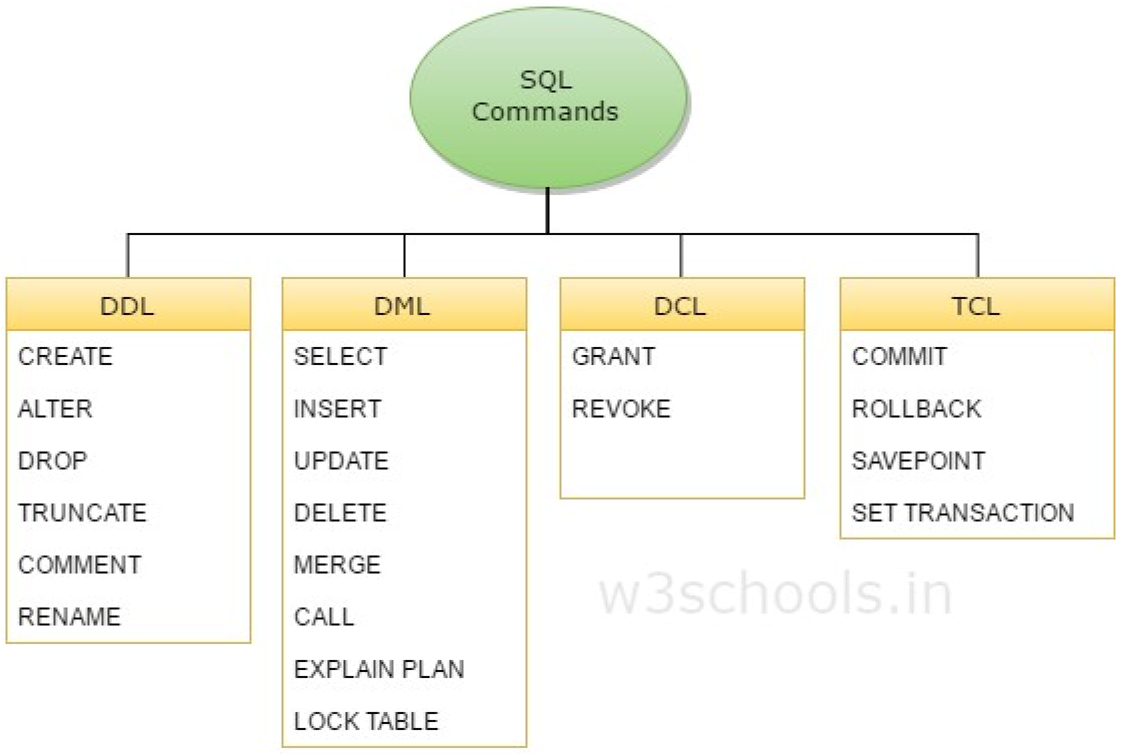
2) Students will be able to demonstrate creating and dropping SQL objects like table, view, sequence, index etc.

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

Software requirements: Ubuntu 14 Operating System, MySQL

Theory:

DATA DEFINITION LANGUAGE (DDL) QUERIES



DDL- Data Definition Language (DDL) statements are used to define the database structure or schema. Data Definition Language understanding with database schemas and describes how the data should consist in the database, therefore language statements like CREATE TABLE or ALTER TABLE belongs to the DDL. DDL is about "metadata".

DDL includes commands such as CREATE, ALTER and DROP statements.DDL is used to CREATE, ALTER OR DROP the database objects (Table, Views, Users).

Data Definition Language (DDL) are used different statements :

- CREATE - to create objects in the database
- ALTER - alters the structure of the database
- DROP - delete objects from the database
- TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
- COMMENT - add comments to the data dictionary
- RENAME - rename an object

Create Database: From the MySQL command line, enter the command CREATE DATABASE <DATABASENAME>;. Replace <DATABASENAMEs> with the name of your database. It cannot include spaces.

- For example, to create a database of all the US states, you might enter CREATE DATABASE pune;
- Note: Commands do not have to be entered in upper-case.
- Note: All MySQL commands must end with ";". If you forgot to include the semicolon, you can enter just ";" on the next line to process the previous command.

Select your database: Once the database has been created, you will need to select it in order to begin editing it. Enter the command

- USE pune;

You will see the message Database changed, letting you know that your active database is now pune.

Display a list of your available databases: Enter the command

- SHOW DATABASES;

to list all of the databases you have stored. Besides the database you just created, you will also see a mysql database and a test database.

Create table: We define an SQL relation by using the **create table** command. The following command creates a relation *department* in the database.

- **create table** *department*(*dept name* **varchar** (20),*building* **varchar** (15),*budget* **numeric** (12,2), **primary key** (*dept name*));

The relation created above has three attributes, *dept name*, which is a character string of maximum length 20, *building*, which is a character string of maximum length 15, and *budget*, which is a number with 12 digits in total, 2 of which are after the decimal point. The **create table** command also specifies that the *dept name* attribute is the primary key of the *department* relation.

Insert values in table: A newly created relation is empty initially. We can use the **insert** command to load data into the relation. For example, if we wish to insert the fact that there is an instructor named Smith in the Biology department with *instructor id* 10211 and a salary of $ 66,000, we write:

- **insert into** *instructor* **values** (10211, 'Smith', 'Biology', 66000);

Drop table: To remove a relation from an SQL database, we use the **drop table** command. The **drop table** command deletes all information about the dropped relation from the database. The command

- **drop table** r

Alter Table : We use the **alter table** command to add attributes to an existing relation. All tuples in the relation are assigned *null* as the value for the new attribute. The form of the **alter table** command is

- **alter table** r **add** AD; where r is the name of an existing relation, A is the name of the attribute to be added, and D is the type of the added attribute. We can drop attributes from a relation by the command

- **alter table** r **drop** A; where r is the name of an existing relation, and A is the name of an attribute of the relation.

View: SQL allows a "virtual relation" to be defined by a query, and the relation conceptually contains the result of the query. The virtual relation is not precomputed and stored, but instead is computed by executing the query whenever the virtual relation is used. Any such relation that is not part of the logical model, but is made visible to a user as a virtual relation, is called a **view**. It is possible to support a large number of views on top of any given set of actual relations.

Create View: We define a view in SQL by using the create view command. To define a view, we must give the view a name and must state the query that computes the view. The form of the create view command is:

- create view *v* as <query expression>; where <query expression> is any legal query expression. The view name is represented by *v.*

  Consider again the clerkwho needs to access all data in the *instructor* relation, except *salary.* The clerk should not be authorized to access the *instructor* relation. Instead, a view relation *faculty* can bemade available to the clerk,with the view defined as follows:

- create view *faculty* as select *ID*, *name*, *dept name* from *instructor*;

  Once we have defined a view, we can use the view name to refer to the virtual relation that the view generates. Using the view *physics fall 2009*, we can find all Physics courses offered in the Fall 2009 semester in the Watson building by writing:

- select *course id* from *physics fall 2009* where *building*= 'Watson';

  Alter View: The CREATE VIEW statement creates a new view, or replaces an existing view if the OR REPLACE clause is given. If the view does not exist, CREATE OR REPLACE VIEW is the same as CREATE VIEW.

  Drop view: Use the DROP VIEW statement to remove a view or an object view from the database. You can change the definition of a view by dropping and re-creating it.

  Syntax:

- Drop view viewname;

Create Index:

A database index is a data structure that improves the speed of operations in a table. Indexes can be created using one or more columns, providing the basis for both rapid random lookups and efficient ordering of access to records.

While creating index, it should be taken into consideration which all columns will be used to make SQL queries and create one or more indexes on those columns.

Practically, indexes are also a type of tables, which keep primary key or index field and a pointer to each record into the actual table.

The users cannot see the indexes, they are just used to speed up queries and will be used by the Database Search Engine to locate records very fast.

The INSERT and UPDATE statements take more time on tables having indexes, whereas the SELECT statements become fast on those tables. The reason is that while doing insert or update, a database needs to insert or update the index values as well.

Simple and Unique Index: You can create a unique index on a table. A unique index means that two rows cannot have the same index value. Here is the syntax to create an Index on a table.

- CREATE UNIQUE INDEX index_name ON table_name (column1, column2,...);

  You can use one or more columns to create an index. For example, we can create an index on table student using column rno

- CREATE UNIQUE INDEX RNO_INDEX ON student (rno);

  You can create a simple index on a table. Just omit the **UNIQUE** keyword from the query to create a simple index. A Simple index allows duplicate values in a table. If you want to index the values in a column in a descending order, you can add the reserved word DESC after the column name.

- CREATE UNIQUE INDEX RNO_INDEX ON student (rno DESC);

  Alter Index:

  There are four types of statements for adding indexes to a table –

- **ALTER TABLE tbl_name ADD PRIMARY KEY (column_list)** – This statement adds a **PRIMARY KEY**, which means that the indexed values must be unique and cannot be NULL.

- **ALTER TABLE tbl_name ADD UNIQUE index_name (column_list)** – This statement creates an index for which the values must be unique (except for the NULL values, which may appear multiple times).

- **ALTER TABLE tbl_name ADD INDEX index_name (column_list)** – This adds an ordinary index in which any value may appear more than once.

- **ALTER TABLE tbl_name ADD FULLTEXT index_name (column_list)** – This creates a special FULLTEXT index that is used for text-searching purposes.

  The following code block is an example to add index in an existing table.

- ALTER TABLE student ADD INDEX (id);

Drop Index:

You can drop any INDEX by using the **DROP** clause along with the ALTER command.

Try out the following example to drop the above-created index.

- ALTER TABLE student DROP INDEX (id);

You can drop any INDEX by using the DROP clause along with the ALTER command.

Sequence: To create a sequence value in SQL query, we can use AUTO_INCREMENT with optional starting value.

1) create table sonali(id int primary key auto_increment, name varchar(20));
2) create table sonali(id int primary key auto_increment = 10, name varchar(20));

Summary of commands :

Creating View, replacing or updating view and dropping view

1) Create view view1 as select ∗ from emp;
2) Create or replace view1 as select ∗ from emp1;
3) Drop view view1;

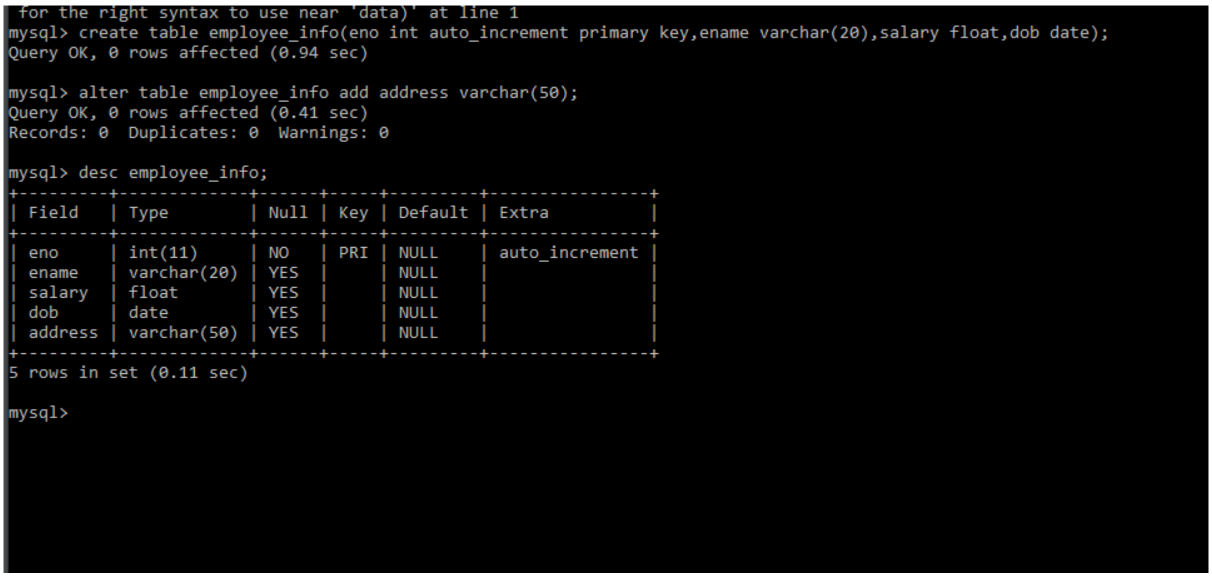
Crating sequence and inserting values

1) create table sonali(id int primary key auto_increment, name varchar(20));
2) create table sonali(id int primary key auto_increment = 10, name varchar(20));
3) insert into sonali(name) values('a'),('b'),('c');
4) insert into sonali values(NULL,'sonali');
5) insert into sonali(name) values('sonali');

Creating index

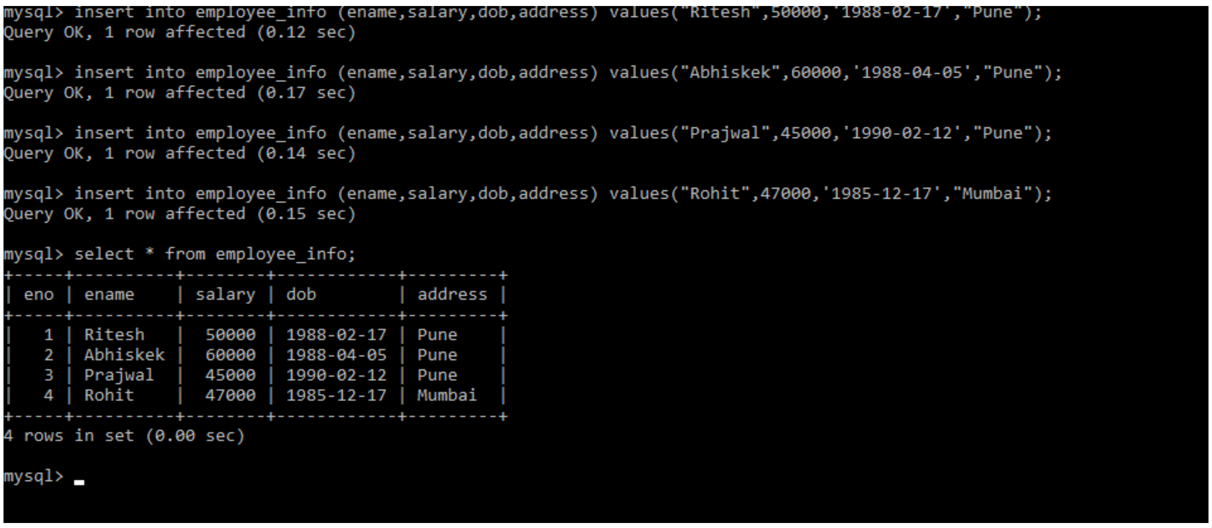
1) create index id1 on sonali(id)
2) create index id1 on sonali(id desc)

1.Create TABLE employee_info and alter table commands



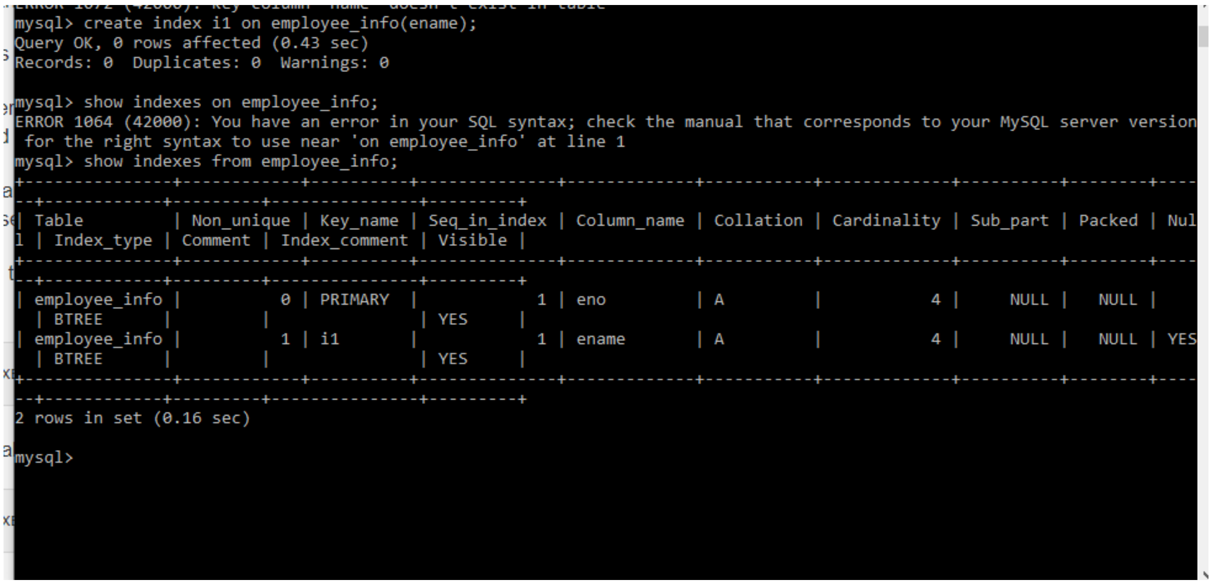
```
 for the right syntax to use near 'data)' at line 1
mysql> create table employee_info(eno int auto_increment primary key,ename varchar(20),salary float,dob date);
Query OK, 0 rows affected (0.94 sec)

mysql> alter table employee_info add address varchar(50);
Query OK, 0 rows affected (0.41 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc employee_info;
+---------+------------+------+-----+---------+----------------+
| Field   | Type       | Null | Key | Default | Extra          |
+---------+------------+------+-----+---------+----------------+
| eno     | int(11)    | NO   | PRI | NULL    | auto_increment |
| ename   | varchar(20)| YES  |     | NULL    |                |
| salary  | float      | YES  |     | NULL    |                |
| dob     | date       | YES  |     | NULL    |                |
| address | varchar(50)| YES  |     | NULL    |                |
+---------+------------+------+-----+---------+----------------+
5 rows in set (0.11 sec)

mysql>
```

2.Inserting values



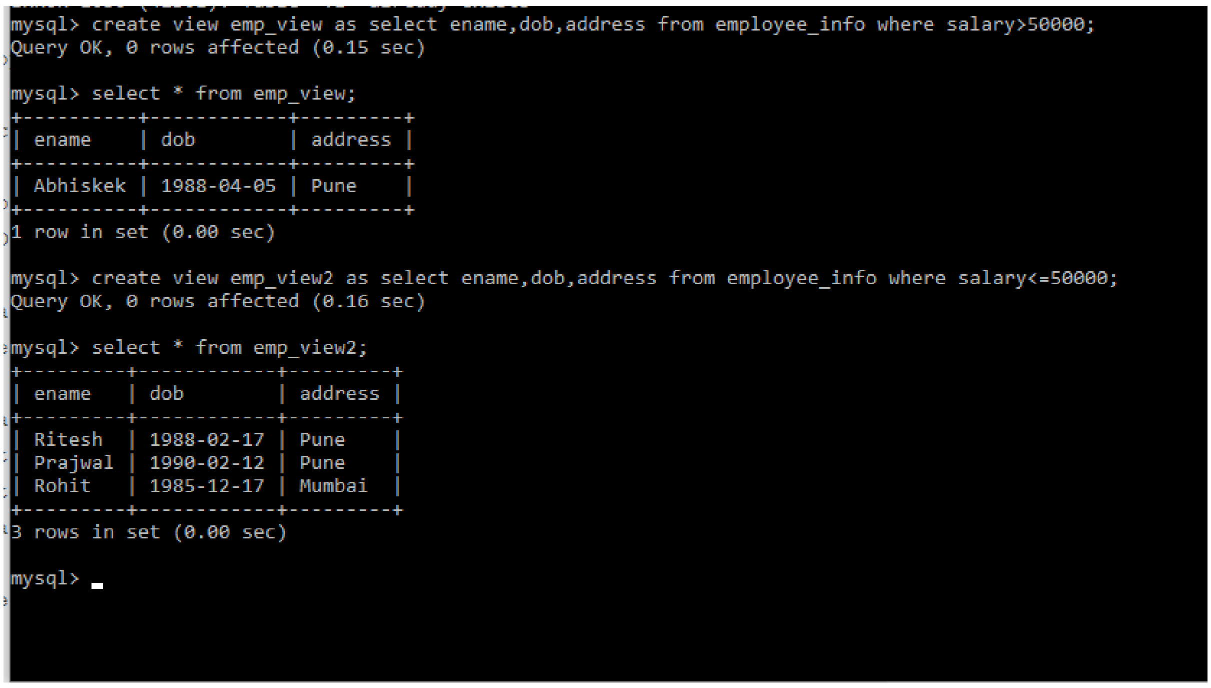
```
mysql> insert into employee_info (ename,salary,dob,address) values("Ritesh",50000,'1988-02-17',"Pune");
Query OK, 1 row affected (0.12 sec)

mysql> insert into employee_info (ename,salary,dob,address) values("Abhiskek",60000,'1988-04-05',"Pune");
Query OK, 1 row affected (0.17 sec)

mysql> insert into employee_info (ename,salary,dob,address) values("Prajwal",45000,'1990-02-12',"Pune");
Query OK, 1 row affected (0.14 sec)

mysql> insert into employee_info (ename,salary,dob,address) values("Rohit",47000,'1985-12-17',"Mumbai");
Query OK, 1 row affected (0.15 sec)

mysql> select * from employee_info;
+-----+----------+--------+------------+---------+
| eno | ename    | salary | dob        | address |
+-----+----------+--------+------------+---------+
|   1 | Ritesh   |  50000 | 1988-02-17 | Pune    |
|   2 | Abhiskek |  60000 | 1988-04-05 | Pune    |
|   3 | Prajwal  |  45000 | 1990-02-12 | Pune    |
|   4 | Rohit    |  47000 | 1985-12-17 | Mumbai  |
+-----+----------+--------+------------+---------+
4 rows in set (0.00 sec)

mysql>
```

3.Creating index on ename



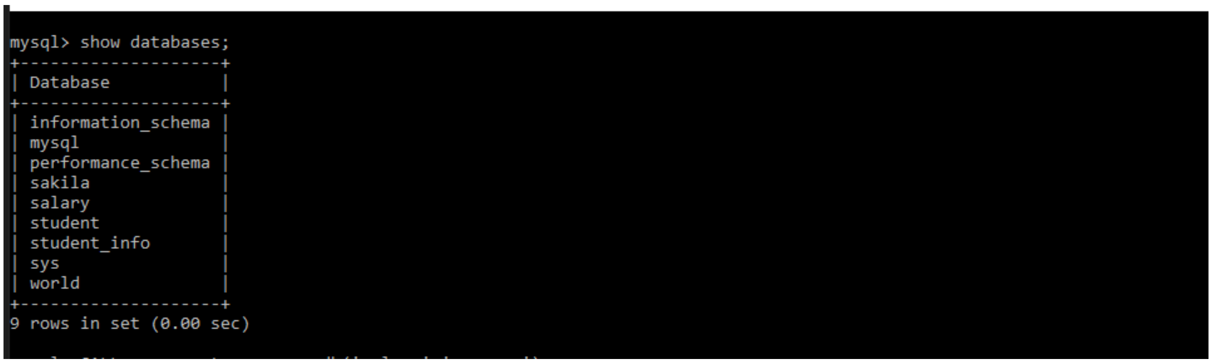
```
mysql> create index i1 on employee_info(ename);
Query OK, 0 rows affected (0.43 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> show indexes on employee_info;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version
 for the right syntax to use near 'on employee_info' at line 1
mysql> show indexes from employee_info;
+---------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+----
--+------------+---------+---------------+---------+
| Table         | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Nul
l | Index_type | Comment | Index_comment | Visible |
+---------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+----
--+------------+---------+---------------+---------+
| employee_info |          0 | PRIMARY  |            1 | eno         | A         |           4 |     NULL |   NULL |
  | BTREE      |         |               | YES     |
| employee_info |          1 | i1       |            1 | ename       | A         |           4 |     NULL |   NULL | YES
  | BTREE      |         |               | YES     |
+---------------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+----
--+------------+---------+---------------+---------+
2 rows in set (0.16 sec)

mysql>
```

Creating view

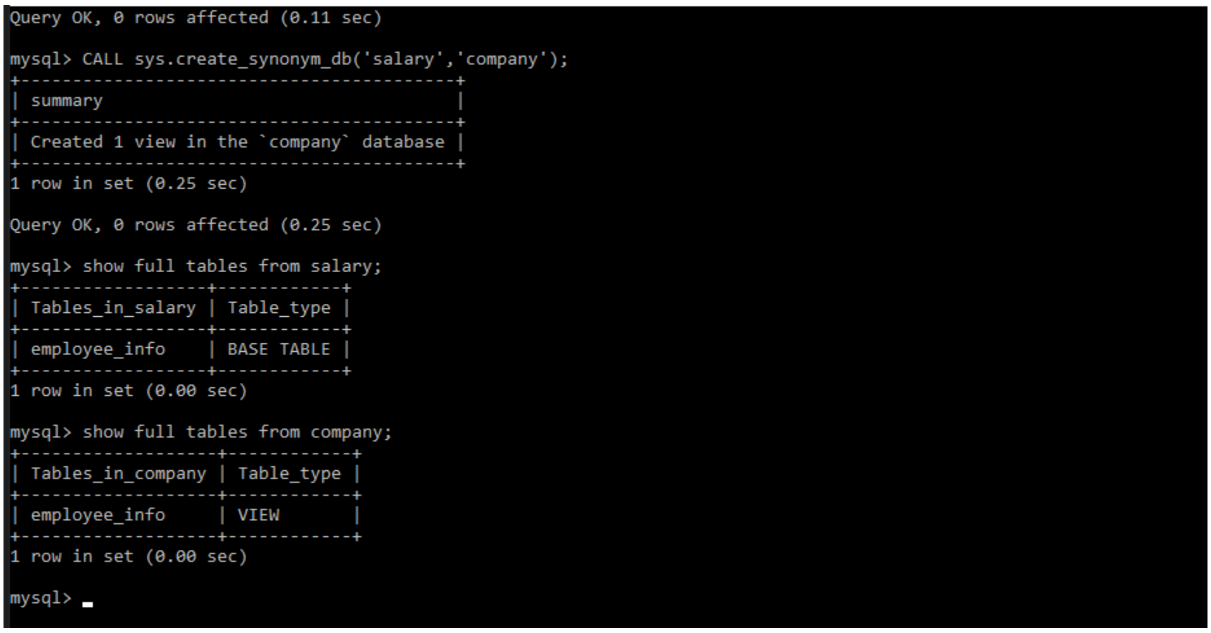


```
mysql> create view emp_view as select ename,dob,address from employee_info where salary>50000;
Query OK, 0 rows affected (0.15 sec)

mysql> select * from emp_view;
+----------+------------+---------+
| ename    | dob        | address |
+----------+------------+---------+
| Abhiskek | 1988-04-05 | Pune    |
+----------+------------+---------+
1 row in set (0.00 sec)

mysql> create view emp_view2 as select ename,dob,address from employee_info where salary<=50000;
Query OK, 0 rows affected (0.16 sec)

mysql> select * from emp_view2;
+---------+------------+---------+
| ename   | dob        | address |
+---------+------------+---------+
| Ritesh  | 1988-02-17 | Pune    |
| Prajwal | 1990-02-12 | Pune    |
| Rohit   | 1985-12-17 | Mumbai  |
+---------+------------+---------+
3 rows in set (0.00 sec)

mysql> _
```

5.Creating Synonyms



```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| salary             |
| student            |
| student_info       |
| sys                |
| world              |
+--------------------+
9 rows in set (0.00 sec)
```



```
Query OK, 0 rows affected (0.11 sec)

mysql> CALL sys.create_synonym_db('salary','company');
+---------------------------------------------+
| summary                                     |
+---------------------------------------------+
| Created 1 view in the `company` database |
+---------------------------------------------+
1 row in set (0.25 sec)

Query OK, 0 rows affected (0.25 sec)

mysql> show full tables from salary;
+------------------+------------+
| Tables_in_salary | Table_type |
+------------------+------------+
| employee_info    | BASE TABLE |
+------------------+------------+
1 row in set (0.00 sec)

mysql> show full tables from company;
+-------------------+------------+
| Tables_in_company | Table_type |
+-------------------+------------+
| employee_info     | VIEW       |
+-------------------+------------+
1 row in set (0.00 sec)

mysql>
```

Conclusion: In this assignment, we have studied and demonstrated various DDL statements in SQL.