INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal - 500 043, Hyderabad, Telangana

COURSE CONTENT

DATABASE MANAGEMENT SYSTEMS LABORATORY								
IV Semester: CSE / IT / CSIT / CSE (AI&ML) / CSE (DS) / CSE (CS)								
Course Code	**Category**	**Hours / Week**			**Credits**	**Maximum Marks**		
		L	**T**	**P**	**C**	**CIA**	**SEE**	**Total**
AITD05	**Core**	0	0	2	1	40	60	100
Contact Classes: Nil	**Tutorial Classes: Nil**	**Practical Classes: 45**				**Total Classes: 45**		
Prerequisite: No Prerequisites								

I. COURSE OVERVIEW:

The purpose of this course is to provide a clear understanding of fundamentals with emphasis on their applications to create and manage large data sets. It highlights on technical overview of database software to retrieve data from n database. The course includes database design principles, normalization, concurrent transaction processing, security, recovery and file organization techniques.

II. COURSES OBJECTIVES:

The students will try to learn

I. The SQL commands for data definition, manipulation, control and perform transactions in database systems.
II. The procedural language for implementation of functions, procedures, cursors and triggers using PL/SQL programs.
III. The logical design of a real time database system with the help of Entity Relationship diagrams.

III. COURSE OUTCOMES:

At the end of the course students should be able to:

CO1	Demonstrate database creation and manipulation concepts with the help of SQL queries.
CO2	Make use of inbuilt functions of SQL queries to perform data aggregations, subqueries, embedded queries and views.
CO3	Apply key constraints on database for maintaining integrity and quality of data.
CO4	Demonstrate normalization techniques by using referential key constraints.
CO5	Implement PL/SQL programs on procedures, cursors and triggers for enhancing the features of database system to handle exceptions.
CO6	Design database model with the help of Entity Relationship diagrams for a real time system or scenario.

IV. COURSE CONTENT:

EXERCISES FOR DATABASE MANAGEMENT SYSTEMS LABORATORY

Note: Students are encouraged to bring their own laptops for laboratory practice sessions.

1. Getting Started Exercises

Installation and Accessing Data base

Introduction:

A Database Management System (DBMS) is a software system that is designed to manage and organize data in a structured manner. It allows users to create, modify, and query a database, as well as manage the security and access controls for that database.

We can use different Database Languages and PL/SQL Programming for performing Database management in Convenient manner.

.**Software:**

Oracle

Installation of Oracle Database:

Step 1: Go to Main Database Folder where you'll find Setup. Right click the setup.exe file and choose Run as Administrator.
Step 2: Click Yes to continue. This will start Oracle Universal Installer.
Step 3: Select any of the three different Installation Options according to your needs.
• Option 1 – If you want to Install Oracle Server Software and want to Create Database also.
• Option 2 – If you want to Install Oracle Server only.
• Option 3 – If you want to Upgrade your Existing Database.
Step 4: Choose between Server Class and Desktop Class as per your requirement and click on Next.
Step 5: Configure the basic settings and create a Password for your database. Once the configuration is done click on Next to continue.
Step 6: Oracle Universal Installer (OUI) will check for the Prerequisites such as Hardware compatibility.
Step 7: Click on Finish to start the Installation process. This installation might take some time depending on your Hardware.
Step 8: Click OK to finish the installation.
Step 9: Copy the localhost link provided to open your Enterprise Manager.
Step 10: Click the Close Button and you are done with the Installation Process.
Go to Start Menu and
• Search Oracle Folder
• Click on Database Control – Oracle (Your Global Database Name)
• This will take you to the Login Screen of your Oracle Enterprise Manager.

1.1 Creating and Deleting a Database - *CREATE DATABASE* and *DROP DATABASE*

You can create a new database using SQL command "`CREATE DATABASE databaseName`"; and delete a database using "`DROP DATABASE databaseName`". You could optionally apply condition "`IF EXISTS`" or "`IF NOT EXISTS`" to these commands. For example,

```
mysql> CREATE DATABASE southwind;
Query OK, 1 row affected (0.03 sec)

mysql> DROP DATABASE southwind;
Query OK, 0 rows affected (0.11 sec)

mysql> CREATE DATABASE IF NOT EXISTS southwind;
Query OK, 1 row affected (0.01 sec)

mysql> DROP DATABASE IF EXISTS southwind;
Query OK, 0 rows affected (0.00 sec)
```

IMPORTANT: Use SQL DROP (and DELETE) commands with extreme care, as the deleted entities are irrecoverable. **THERE IS NO UNDO!!!**

SHOW CREATE DATABASE

The CREATE DATABASE commands uses some defaults. You can issue a "SHOW CREATE DATABASE *databaseName*" to display the full command and check these default values. We use \G (instead of ';') to display the results vertically. (Try comparing the outputs produced by ';' and \G.)

```
mysql> CREATE DATABASE IF NOT EXISTS southwind;

mysql> SHOW CREATE DATABASE southwind \G
*************************** 1. row ***************************
       Database: southwind
Create Database: CREATE DATABASE `southwind` /*!40100 DEFAULT CHARACTER SET latin1 */
```

Try:
Create own database for storing all tables

2. Exercises on database definition language queries.

2.1 Create a table called Employee with the following structure.

Name	Type
Empno	Number
Ename	Varchar2(20)
Job	Varchar2(20)
Mgr	Number
Sal	Number

a. Add a column commission with domain to the Employee table.
b. Insert any five records into the table.
c. Update the column details of job
d. Rename the column of Employ table using alter command.

e. Delete the employee whose emp no is 19

```
create table employee (empno number, ename varchar2(10),job varchar2(10),mgr number,sal
number);
Table created.
SQL> desc employee;
Name Null Type
EMPNO NUMBER
ENAME VARCHAR2(10)
JOB VARCHAR2(10)
MGR NUMBER
SAL NUMBER
```

2.2 Add a column commission with domain to the Employee table.

```
SQL> alter table employee add(commission number); Table altered.
SQL> desc employee;
Name Null Type
EMPNO NUMBER
ENAME VARCHAR2(10)
JOB VARCHAR2(10)
MGR NUMBER
SAL NUMBER
COMMISSION NUMBE
```

2.3 Insert any five records into the table.

SQL> insert into employee values(&empno,'&ename','&job',&mgr,&sal,'&commission');

```
Enter value for empno: 101
Enter value for ename: abhi Enter value for job: manager
Enter value for mgr: 1234 Enter value for sal: 10000 Enter value for commission: 70
Page 11
old 1: insert into employee values(&empno,'&ename','&job',&mgr,&sal,'&commission')
new 1: insert into employee values(101,'abhi','manager',1234,10000,'70')
1 row created.
```

2.4 Update the column details of job

SQL> update employee set job='trainee' where empno=103; 1 row updated.
SQL> select * from employee;

```
EMPNO ENAME JOB MGR SAL COMMISSION
101 abhi manager 1234 10000 70
102 rohith analyst 2345 9000 65
103 david trainee 3456 9000 65
104 rahul Clerk 4567 7000 55
105 pramod salesman 5678 5000 5
```

2.5 Rename the column of Employ table using alter command.

SQL> alter table employee rename column mgr to manager_no;
Table altered.
SQL> desc employee;

Name Null Type
EMPNO NUMBER
ENAME VARCHAR2(10)
JOB VARCHAR2(10)
MANAGER_NO NUMBER
SAL NUMBER
COMMISSION NUMBER

Try:
1. List the maximum salary paid to salesman
2. Display name of emp whose name start with "I"
3. List details of emp who have joined before "30-sept-81"
4. Display the emp details in the descending order of their basic salary
5. List of no of emp & avg salary for emp in the dept no „20"

Hint:
Create a database called company consist of the following tables by using data definition languages.
1.Emp (eno, ename, job, hire date, salary, commission, deptno,)
2.dept (deptno, deptname, location)
eno is primary key in emp
deptno is primary key in dept

3. Exercises on database Manipulation language Queries.

3.1 Create a table called department with the following structure.

Name	Type
Deptno	Number
Deptna me location	Varchar2(2 0) Varchar2(2 0)

SQL> create table department (deptno number, deptname varchar2(10), location varchar2(10));

Table created.
SQL> desc department;
DEPTNO NUMBER
DEPTNAME VARCHAR2(10)
LOCATION VARCHAR2(10)

3.2 Add column designation to the department table.

SQL> alter table department add(designation varchar2(10));
Table altered.

```
SQL> desc department;
DEPTNO NUMBER
DEPTNAME VARCHAR2(10)
LOCATION VARCHAR2(10)
DESIGNATION VARCHAR2(10)
```

3.3 Insert values into the table.

SQL> insert into department values(&deptno,'&deptname','&location',&designation');

```
Enter value for deptno: 9
Enter value for deptname: accounting Enter value for location: hyderabad Enter value for designation:
manager
```

old 1: insert into department values(&deptno,'&deptname','&location','&designation')
new 1: insert into department values(9,'accounting','hyderabad','manager')

```
1 row created.
SQL> /
```

Enter value for deptno: 10
Enter value for deptname: research Enter value for location: chennai Enter value for designation:
professor

```
old 1: insert into department values(&deptno,'&deptname','&location','&designation')
new 1: insert into department values(10,'research','chennai','professor')
1 row created
```

3.4 List the records of emp table grouped by deptno.

SQL> select deptno,deptname from department group by deptno,deptname;

```
Page 13
DEPTNO DEPTNAME
9 accounting
12 operations
10 research
11 sales
```

3.5 Update the record designation where deptno is 17.

```
SQL> update department set designation='accountant' where deptno=17;
2 rows updated.
```

Try:

1. List full details of all Rooms.
2. How many Rooms are there.
3. List the price and type of all rooms at the Amrutha Hotel.
4. List the number of guests in each Room
5. Update the price of all rooms by 5%.

Hint:
The following tables form part of a database held in a relational DBMS by using data manipulation languages:

Hotel (HotelNo, Name, City) HotelNo is the primary key
Room (RoomNo, HotelNo, Type, Price)
Booking (HotelNo, GuestNo, DateFrom, DateTo, RoomNo)
Guest (GuestNo, GuestName, GuestAddress) GuestNo is primary key
Room contains room details for each hotel and (HotelNo, RoomNo) forms the primary key.
Booking contains details of the bookings and the primary key comprises (HotelNo, GuestNo and DateFrom)

4. Exercises on both DDL and DML Queries.

4.1 Create a table called Customer with the following structure.

Students can execute fallowing queries based on previous queries executed

Try:
1. Retrieve city, phone, url of author whose name is „CHETAN BHAGAT".
2. Retrieve book title, reviewable id and rating of all books.
3. Retrieve book title, price, author name and url for publishers „MEHTA".
4. In a PUBLISHER relation change the phone number of „MEHTA" to 123456
5. Calculate and display the average, maximum, minimum price of each publisher.
6. Delete details of all books having a page count less than 100.
7. Retrieve details of all authors residing in city Pune and whose name begins with character „C".
8. Retrieve details of authors residing in same city as „Korth".
9. Create a procedure to update the value of page count of a book of given ISBN.
10. Create a function that returns the price of book with a given ISBN.

Hint:
Create the following tables using data definition languages, data manipulation languages.
1)PUBLISHER (PID , PNAME ,ADDRESS ,STATE ,PHONE ,EMAILID);
2)BOOK (ISBN ,BOOK_TITLE , CATEGORY , PRICE , COPYRIGHT_DATE , YEAR ,PAGE_COUNT ,PID);
3) AUTHOR (AID,ANAME,STATE,CITY ,ZIP,PHONE,URL)
4) AUTHOR_BOOK (AID, ISBN);
5) REVIEW (RID, ISBN, RATING);

5. Exercises on tables with different types of constraints.

5.1 Not Null constraints.

CREATE TABLE STUDENT(

```
ROLL_NO INT NOT NULL,
STU_NAME VARCHAR (35) NOT NULL,
STU_AGE INT NOT NULL,
STU_ADDRESS VARCHAR (235),
PRIMARY KEY (ROLL_NO)
);
```

```
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> insert into student values(6201 ,'raj',18,'meerpet');
```

```
Query OK, 1 row affected (0.02 sec)
```

```
mysql> insert into student values(NULL,'raj',18,'meerpet');
```

```
ERROR 1048 (23000): Column 'ROLL_NO' cannot be null
```

5.2 Unique constraints.

```
CREATE TABLE STUDENT1(
ROLL_NO INT NOT NULL,
STU_NAME VARCHAR (35) NOT NULL UNIQUE,
STU_AGE INT NOT NULL,
STU_ADDRESS VARCHAR (35) UNIQUE,
PRIMARY KEY (ROLL_NO)
);
```

```
Query OK, 0 rows affected (0.02 sec)
```

```
mysql> insert into student1 values(6201,'raj',18,'meerpet');
Query OK, 1 row affected (0.01 sec)
```

```
mysql> insert into student1 values(6201,'raj',18,'meerpet');
ERROR 1062 (23000): Duplicate entry '6201' for key 'PRIMARY'
mysql> insert into student1 values(6202,'raj',18,'meerpet');
ERROR 1062 (23000): Duplicate entry 'raj' for key 'STU_NAME'
mysql> insert into student1 values(6202,'ram',18,'meerpet');
ERROR 1062 (23000): Duplicate entry 'meerpet' for key 'STU_ADDRESS'
```

5.3 Default constraints.

```
CREATE TABLE STUDENT2(
ROLL_NO   INT  NOT NULL,
STU_NAME VARCHAR (35) NOT NULL,
STU_AGE INT NOT NULL,
EXAM_FEE INT  DEFAULT 10000,
```

```
STU_ADDRESS VARCHAR (35) ,
PRIMARY KEY (ROLL_NO)
);
insert into
student2(ROLL_NO,STU_NAME,STU_AGE,STU_ADDRESS)values(6202,'sam',21,'lbnagar');
Query OK, 1 row affected (0.00 sec)

mysql>  select * from student2;
+---------+----------+---------+----------+-------------+
| ROLL_NO | STU_NAME | STU_AGE | EXAM_FEE | STU_ADDRESS |
+---------+----------+---------+----------+-------------+
|    6201 | raj      |      18 |    12000 | meerpet     |
|    6202 | sam      |      21 |    10000 | lbnagar     |
+---------+----------+---------+----------+-------------+
2 rows in set (0.00 sec)
```

5.4 Check constraints.

```
CREATE TABLE STUDENT3(
ROLL_NO   INT  NOT NULL CHECK(ROLL_NO >1000) ,
STU_NAME VARCHAR (35)  NOT NULL,
STU_AGE INT  NOT NULL,
EXAM_FEE INT DEFAULT 10000,
STU_ADDRESS VARCHAR (35) ,
PRIMARY KEY (ROLL_NO)
);

 insert into student3 values(112,'ramana',23,12000,'hyderabad');
Query OK, 1 row affected (0.01 sec)

mysql> select * from student3;
+---------+----------+---------+----------+-------------+
| ROLL_NO | STU_NAME | STU_AGE | EXAM_FEE | STU_ADDRESS |
+---------+----------+---------+----------+-------------+
|     112 | ramana   |      23 |    12000 | hyderabad   |
|     800 | raj      |      13 |    12000 | lbnagar     |
|     900 | raj      |      13 |    12000 | lbnagar     |
+---------+----------+---------+----------+-------------+
3 rows in set (0.00 sec)
```

5.5 Auto Increment Constraints.

```
CREATE TABLE Persons1 (
    Personid int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
```

```
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (Personid)
);

INSERT INTO persons1 values(11,'reddy','ramana',23);
Query OK, 1 row affected (0.00 sec)

INSERT INTO persons1(LastName,FirstName,Age) values('reddy','ramana',23);
Query OK, 1 row affected (0.02 sec)

mysql> select * from persons1;
+----------+----------+-----------+------+
| Personid | LastName | FirstName | Age  |
+----------+----------+-----------+------+
|       11 | reddy    | ramana    |   23 |
|       12 | reddy    | ramana    |   23 |
+----------+----------+-----------+------+
2 rows in set (0.00 sec)
```

Try:
Create tables with both Primary key and foreign key constraints

Hint:
Practice all constraints for obtaining syntax on both primary and foreign key constraints that applied on ID.

6. Exercises on Join Operations and Aggregate Functions in Database.

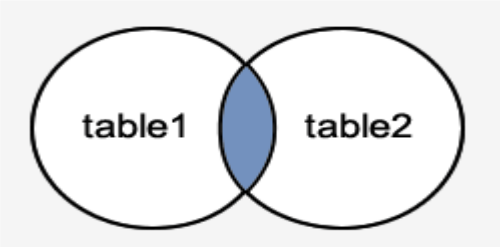
6.1 Create table names called 'students' and 'officers' and insert values as show in below.

```
mysql> create table officers (officer_id integer(2),officer_name char(7),officer_address char(7));
Query OK, 0 rows affected (0.01 sec)

mysql> insert into officers values(1,'ajeet','Mau'),(2,'deepika','luknow'),(3,'vimal','hyderabad');
ERROR 1406 (22001): Data too long for column 'officer_address' at row 3
mysql> insert into officers values(1,'ajeet','Mau');
Query OK, 1 row affected (0.00 sec)

mysql> insert into officers values(2,'deepika','luknow');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> create table student(student_id integer(7),student_name char(10),course_name char(10));
Query OK, 0 rows affected (0.02 sec)

mysql> insert into student values(1,'aryan','java');
Query OK, 1 row affected (0.01 sec)

mysql> insert into student values(2,'rohini','haddop');
Query OK, 1 row affected (0.00 sec)

mysql> insert into student values(3,'lallu','mongoDB');
Query OK, 1 row affected (0.01 sec)
```

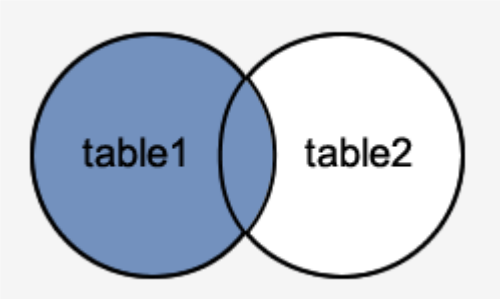
6.2 Inner join

1. SELECT officers.officer_name, officers.officer_address, students.course_name
2. FROM officers
3. INNER JOIN students
4. ON officers.officer_id = students.student_id;



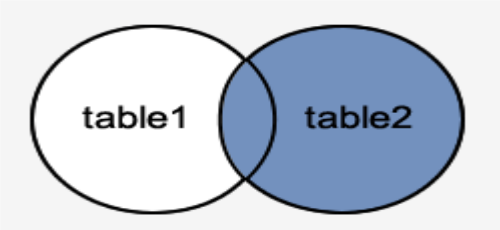
6.3 Left Outer Join

1. SELECT officers.officer_name, officers.officer_address, students.course_name
2. FROM officers
3. LEFT JOIN students
4. ON officers.officer_id = students.student_id;



6.4 Right Outer Join

1. SELECT officers.officer_name, officers.officer_address, students.course_name, students.student_name
2. FROM officers
3. RIGHT JOIN students
4. ON officers.officer_id = students.student_id;



6.5 Cross Join

SSelect officers.officer_name,officers.officer_address,students.course_name from officers CROSS join students;

```
mysql>  select officers.officer_name,officers.officer_address,students.course_name from officers CROSS join students;
+--------------+-----------------+-------------+
| officer_name | officer_address | course_name |
+--------------+-----------------+-------------+
| ajeet        | Mau             | java        |
| ajeet        | Mau             | haddop      |
| ajeet        | Mau             | mongoDB     |
| ajeet        | Mau             | luknow      |
| deepika      | luknow          | java        |
| deepika      | luknow          | haddop      |
| deepika      | luknow          | mongoDB     |
| deepika      | luknow          | luknow      |
| vimal        | hydera          | java        |
| vimal        | hydera          | haddop      |
| vimal        | hydera          | mongoDB     |
| vimal        | hydera          | luknow      |
+--------------+-----------------+-------------+
12 rows in set (0.00 sec)
```

Try:
Perform Self join, Equi join, Natural joins on tables to get the values from multiple tables.

Hint:
Understand and use same syntax applied for all joins can use as same it is for Self-join, Equi join, Natural joins.

6.6 AGGREGATE FUNCTIONS:

1. Count() Function:

```
SELECT COUNT(name) FROM employee;
```

```
mysql> SELECT COUNT(name) FROM employee;
+-------------+
| COUNT(name) |
+-------------+
|           6 |
+-------------+
1 row in set (0.00 sec)
```

2. Sum() Function

```
SELECT SUM(working_hours) AS "Total working hours" FROM employee;
```

3. AVG() Function:

```
SELECT AVG(working_hours) AS "Average working hours" FROM employee;
```

4. MIN() Function:

```
SELECT MIN(working_hours) AS Minimum_working_hours FROM employee;
```

5. MAX() Function:

```
SELECT MAX(working_hours) AS Maximum_working_hours FROM employee;
```

Try:
Execute other aggregate functions to find first and Last employees

Hint:
Use First and Last functions as aggregate functions to execute

7. Exercises on TCL (transaction control language) Queries to perform rollbacking and save point mechanisms.

7.1 Create table names called 'class' and insert values into table.

mysql> create table class (id int(5),Name char(10));

```
Query OK, 0 rows affected (0.01 sec)
```

mysql> insert into class values(1,'one'),(2,'two');

```
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

```
mysql> select * from class;
+------+------+
| id   | Name |
+------+------+
|    1 | one  |
|    2 | two  |
+------+------+
2 rows in set (0.00 sec)
```

Now to perform TCL we need start the transaction:

```
For that START TRANSACTION is a command to do start our transaction.
SAVEPOINT works only before commit operation.
After commit operation SAVEPOINT will not work.
```

So save point saves the current work and displays the information.

7.2 Before commit performs

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO CLASS VALUES(3,'THREE');
Query OK, 1 row affected (0.02 sec)

mysql> SAVEPOINT A;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO CLASS VALUES(4,'FOUR');
Query OK, 1 row affected (0.00 sec)

mysql> SAVEPOINT B;
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE CLASS SET NAME='RAJ' where id=1;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SAVEPOINT C;
Query OK, 0 rows affected (0.00 sec)
```

NOW
If we perform COMMIT operation then we are not rollback the data which happens on save point states;
We can roll back the data which doesn't performed with commit operation.

```
mysql> ROLLBACK TO SAVEPOINT A;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM CLASS;
+------+-------+
| id   | Name  |
+------+-------+
|    1 | one   |
|    2 | two   |
|    3 | THREE |
+------+-------+
3 rows in set (0.00 sec)
```

7.3 After commit performs

```
mysql> insert into class values(5,'five');
Query OK, 1 row affected (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.00 sec)

mysql> ROLLBACK TO SAVEPOINT B;
ERROR 1305 (42000): SAVEPOINT B does not exist
mysql> ROLLBACK TO SAVEPOINT C;
ERROR 1305 (42000): SAVEPOINT C does not exist
mysql> ROLLBACK TO SAVEPOINT A;
ERROR 1305 (42000): SAVEPOINT A does not exist
mysql>
```

So after COMMIT No information will rollback.
All the above commit,
rollback, save point operations can be done after START TRANSACTION only.

Try:
Insert two more values and perform commit operation then roll back again

Hint:

Perform before commit operation to rollback data from various operations performed

8. Exercises on user accounts to grant privileges using Data Control language process.

8.1 Create user.

create user 'username'@'localhost' IDENTIFIED BY 'password';
Ex:

create user 'iare' identified by 'iare';
Query OK, 0 rows affected (0.05 sec)

SEEL LIST OF USERS IN DATABASE:

select user from mysql.user;

TO SEE CURRENT USER:

select current_user();

TO SEE DESCRIPTION OF USER:

desc mysql.user;

TO SEE LIST OF USERS, HOST:

```
select user,host from mysql.user;
```

8.2 To Grant All Privileges, Specific Privileges to The User:

SYNTAX:

```
GRANT ALL PRIVILEGES ON * . * TO 'new_user'@'localhost';
GRANT ALL PRIVILEGES ON * . * TO '2nduser'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

TO GRANT SPECIFIC PRIVILEGES TO THE USER:

```
GRANT CREATE, SELECT ON * . * TO '2nduser'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

SHOW PRIVILEGES TO THE USER:

```
SHOW GRANTS FOR '2nduser'@'localhost';
```

```
mysql> SHOW GRANTS FOR '2nduser'@'localhost';
+-----------------------------------------------------------------------------------------------------------------------------+
| Grants for 2nduser@localhost                                                                                                |
+-----------------------------------------------------------------------------------------------------------------------------+
| GRANT SELECT, CREATE ON *.* TO '2nduser'@'localhost' IDENTIFIED BY PASSWORD '*AB89A321BAE9049CFC0F342CAE0534E391B591FB' |
+-----------------------------------------------------------------------------------------------------------------------------+
```

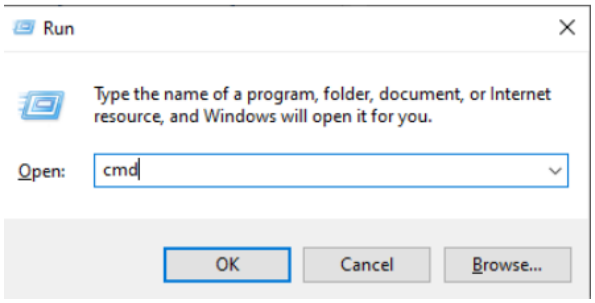
8.3 Revoke permissions from user:

```
mysql> REVOKE ALL PRIVILEGES ON * . * FROM '2nduser'@'localhost';
Query OK, 0 rows affected (0.00 sec)
```

8.4 Login as User:

Before login as a user, the user has to privileged with root user as mentioned above grants option.
That means a new user having permission to login as user.

Now
To login to a different user account, we need to open the command prompt by executing the run command.



After clicking the **OK** button, we can see the command prompt
Now move to the path where my sql has been specified

C drive->program files(x86)->mysql->mysql server5.5->bin

As shown in below

```
C:\>cd Program Files (x86)\MySQL\MySQL Server 5.5\bin

C:\Program Files (x86)\MySQL\MySQL Server 5.5\bin>
```

Next type the fallowing syntax to login as user:

Mysql -u username -p

```
C:\Program Files (x86)\MySQL\MySQL Server 5.5\bin>mysql -u 2nduser -p
Enter password: *******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.5.27 MySQL Community Server (GPL)

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

8.5 DROP USERS FROM DATABASE:

DROP USER ‘user_name’@‘localhost’;

```
mysql> drop user '2nduser'@'localhost';
Query OK, 0 rows affected (0.01 sec)
```

Try:
create any other user and grant specific privileges on create and select permissions

Hint:
create user and perform DCL operations to give privileges and revoke privileges

9. Working with basic PL/SQL programing Examples

To start with pl/sql programming
Open oracle run sql cmd line avialable in oracle
After opening window
Just type CONNECT and hit enter
Now
it is asking you to enter user-name and password;
Enter both and now it is connected as shown in below

```
SQL> CONNECT
Enter user-name: SYSTEM
Enter password:
Connected.
SQL>
```

```
SQL> CONNECT
Enter user-name: SYSTEM
Enter password: admin
Connected.
```

Before starting pl/sql programming we need to set server;
To start server type fallowing command after connected as shown in above.
SET SERVEROUTPUT ON

After that we can start to write program as per the below syntax

```
DECLARE
   <declarations section>
BEGIN
   <executable command(s)>
EXCEPTION
   <exception handling>
END;
```

9.1 Hello world example with programming

```
SQL> CONNECT
Enter user-name: SYSTEM
Enter password:
Connected.
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
  2     message  varchar2(20):= 'Hello, World!';
  3  BEGIN
  4     dbms_output.put_line(message);
  5  END;
  6  /
Hello, World!

PL/SQL procedure successfully completed.

SQL>
```

9.2 Declaration and Initialization of Variables

```
SQL> DECLARE
  2     var1 integer := 20;
  3     var2 integer := 40;
  4     var3 integer;
  5     var4 real;
  6  BEGIN
  7     var3 := var1 + var2;
  8     dbms_output.put_line('Value of var3: ' || var3);
  9     var4 := 50.0/3.0;
 10     dbms_output.put_line('Value of var4: ' || var4);
 11  END;
 12  /
Value of var3: 60
Value of var4: 16.66666666666666666666666666666666666667

PL/SQL procedure successfully completed.
```

9.3 Datatypes and its values

```
SQL> DECLARE
  2       m CHAR(20) := 'softwareTest!';
  3       n VARCHAR2(30) := 'plsql';
  4       o NCHAR(30) := 'plsql datatypes';
  5       p NVARCHAR2(30) := 'plsql literals';
  6       presentDt DATE:= SYSDATE;
  7       a INTEGER := 16;
  8       b NUMBER(20) := 11.2;
  9       c DOUBLE PRECISION := 14.7;
 10  BEGIN
 11      dbms_output.put_line('The char datatype is: ' || m);
 12      dbms_output.put_line('The varchar datatype is: ' || n);
 13      dbms_output.put_line('The nchar datatype is: ' || o);
 14      dbms_output.put_line('The nvarchar2 datatype is: ' || p);
 15      dbms_output.put_line('The current date is: ' || presentDt);
 16      dbms_output.put_line('The number a is: ' || a);
 17      dbms_output.put_line('The number b is: ' || b);
 18      dbms_output.put_line('The number c is: ' || c);
 19  END;
 20  /
The char datatype is: softwareTest!
The varchar datatype is: plsql
The nchar datatype is: plsql datatypes
The nvarchar2 datatype is: plsql literals
The current date is: 19-JUN-23
The number a is: 16
The number b is: 11
The number c is: 14.7

PL/SQL procedure successfully completed.
```

Try:
write program to display both local and global variable values
Hint:
use the concept local and global declaration and its variables

10. Working with PL/SQL Programming Functions.

Syntax to create a function:

```
CREATE [OR REPLACE] FUNCTION function_name [parameters]
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
RETURN return_datatype
{IS | AS}
BEGIN
   < function_body >
END [function_name];
```

10.1 Addition of two numbers

Creating function:

```
create or replace function adder(n1 in number, n2 in number)
return number
is
n3 number(8);
begin
n3 :=n1+n2;
return n3;
end;
/
```

call the function.:

```
DECLARE
  n3 number(2);
BEGIN
  n3 := adder(11,22);
  dbms_output.put_line('Addition is: ' || n3);
END;
/
```

```
SQL> DECLARE
  2     n3 number(2);
  3  BEGIN
  4     n3 := adder(11,22);
  5     dbms_output.put_line('Addition is: ' || n3);
  6  END;
  7  /
Addition is: 33

PL/SQL procedure successfully completed.
```

10.2 compute and return the maximum of two values.

```
DECLARE
  a number;
  b number;
  c number;
FUNCTION findMax(x IN number, y IN number)
RETURN number
IS
   z number;
BEGIN
  IF x > y THEN
     z:= x;
  ELSE
     Z:= y;
  END IF;
 RETURN z;
END;
BEGIN
  a:= 23;
  b:= 45;
  c := findMax(a, b);
  dbms_output.put_line(' Maximum of (23,45): ' || c);
END;
/
```

10.3 creating and calling function to access values from table.

To do this first create a table as shown in below

```
CREATE TABLE customers1
   ( customer_id number(10) NOT NULL,
   customer_name varchar2(50) NOT NULL,
  city varchar2(50)
  );
```

Next insert values into table as shown in below

```
insert into customers1 values(12,'raj','hyderabad');

insert into customers1 values(13,'ram','secunderabad');
```

Now create a function to count the no of customers in table:

Creating function:

```
CREATE OR REPLACE FUNCTION Fcustomers
RETURN number IS
   total number(2) := 0;
BEGIN
   SELECT count(*) into total
   FROM customers1;
    RETURN total;
END;
/
```

Calling function:

```
DECLARE
   c number(2);
BEGIN
   c := Fcustomers();
   dbms_output.put_line('Total no. of Customers: ' || c);
END;
/.
```

Try:
Create a function that updates the value of any column in the table.

Hint:
To update we use dml commands and call function to access values from table.

11. Working with PL/SQL Programming procedures.

Syntax:

```
CREATE [OR REPLACE] PROCEDURE procedure_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
{IS | AS}
BEGIN
  < procedure_body >
END procedure_name;
```

11.1 Create procedure to display hello world.

```
CREATE OR REPLACE PROCEDURE greetings
AS
BEGIN
   dbms_output.put_line('Hello World!');
END;
/
```

TO EXECUTE:

```
EXECUTE greetings;
```

```
SQL> CREATE OR REPLACE PROCEDURE greetings
  2  AS
  3  BEGIN
  4     dbms_output.put_line('Hello World!');
  5  END;
  6  /

Procedure created.

SQL> EXECUTE greetings;
Hello World!

PL/SQL procedure successfully completed.
```

11.2 IN & OUT Mode Example to display square of number

```
DECLARE
   a number;
PROCEDURE squareNum(x IN OUT number) IS
```

```
BEGIN
  x := x * x;
END;
BEGIN
   a:= 23;
   squareNum(a);
   dbms_output.put_line(' Square of (23): ' || a);
END;
/
```

11.3 calculate the net salary and year salary if da is 30% of basic, hra is 10% of basic and pf is 7% if basic salary is less than 8000, pf is 10% if basic sal between 8000 to 160000.

```
declare
ename varchar2(15); basic number;
da number; hra number; pf number;
netsalary number; yearsalary number;
begin
ename:='&ename'; basic:=&basic; da:=basic * (30/100); hra:=basic * (10/100);
if (basic < 8000) then
pf:=basic * (8/100);
elsif (basic >= 8000 and basic <= 16000) then
pf:=basic * (10/100);
end if;
netsalary:=basic + da + hra - pf; yearsalary := netsalary*12;
dbms_output.put_line('Employee name : ' || ename);
dbms_output.put_line('Providend Fund : ' || pf); dbms_output.put_line('Net salary : ' || netsalary);
dbms_output.put_line('Year salary : '|| yearsalary); end;
/
```

```
SQL> declare
  2  ename varchar2(15); basic number;
  3  da number; hra number; pf number;
  4  netsalary number; yearsalary number;
  5  begin
  6  ename:='&ename'; basic:=&basic; da:=basic * (30/100); hra:=basic * (10/100);
  7  if (basic < 8000) then
  8  pf:=basic * (8/100);
  9  elsif (basic >= 8000 and basic <= 16000) then
 10  pf:=basic * (10/100);
 11  end if;
 12  netsalary:=basic + da + hra - pf; yearsalary := netsalary*12;
 13  dbms_output.put_line('Employee name : ' || ename);
 14  dbms_output.put_line('Providend Fund : ' || pf); dbms_output.put_line('Net salary : ' || netsalary);
 15  dbms_output.put_line('Year salary : '|| yearsalary); end;
 16  /
Enter value for ename: ONE
Enter value for basic: 7000
old   6: ename:='&ename'; basic:=&basic; da:=basic * (30/100); hra:=basic * (10/100);
new   6: ename:='ONE'; basic:=7000; da:=basic * (30/100); hra:=basic * (10/100);
Employee name : ONE
Providend Fund : 560
Net salary : 9240
Year salary : 110880

PL/SQL procedure successfully completed.
```

Try:
Write procedure to find maximum and minimum of two numbers

Hint:
create and call procedure, use conditions to find max or minimum numbers

12. Working with PL/SQL Programming Triggers.

Triggers are stored programs, which are automatically executed or fired when some event occurs. Trigger is stored into database and invoked repeatedly, when specific condition match.

12.1 Syntax for creating trigger:

```
CREATE [OR REPLACE ] TRIGGER trigger_name
{BEFORE | AFTER | INSTEAD OF }
{INSERT [OR] | UPDATE [OR] | DELETE}
[OF col_name]
ON table_name
[REFERENCING OLD AS o NEW AS n]
[FOR EACH ROW]
WHEN (condition)
```

```
DECLARE
  Declaration-statements
BEGIN
  Executable-statements
EXCEPTION
  Exception-handling-statements
END;
```

To do this first create a table called emp:

create table emp (id number, name varchar2(50),salary number);

```
SQL> create table emp(
  2  id number,name varchar2(50),salary number);

Table created.
```

Insert any values:

```
SQL> insert into emp values(121,'raj',5000);

1 row created.

SQL> insert into emp values(122,'rani',4000);

1 row created.
```

Check values inserted or not:

```
SQL> select * from emp;

        ID NAME                                               SALARY
---------- -------------------------------------------------- ----------
       121 raj                                                  5000
       122 rani                                                 4000
```

12.1 Creating the trigger for updating salary:

```
CREATE OR REPLACE TRIGGER display_salary_changes
BEFORE DELETE OR INSERT OR UPDATE ON emp
FOR EACH ROW
WHEN (NEW.ID > 0)
DECLARE
  sal_diff number;
BEGIN
  sal_diff := :NEW.salary  - :OLD.salary;
  dbms_output.put_line('Old salary: ' || :OLD.salary);
  dbms_output.put_line('New salary: ' || :NEW.salary);
  dbms_output.put_line('Salary difference: ' || sal_diff);
END;
/
```

12.2 Calling the trigger for updating salary:

```
DECLARE
  total_rows number(2);
BEGIN
  UPDATE  emp
  SET salary = salary + 5000;
  IF sql%notfound THEN
    dbms_output.put_line('no customers updated');
  ELSIF sql%found THEN
    total_rows := sql%rowcount;
    dbms_output.put_line( total_rows || ' customers updated ');
  END IF;
```

```
END;
/
```

12.2 Check the salary is updated or not:

```
SQL> select * from emp;

        ID NAME                                               SALARY
---------- -------------------------------------------------- ----------
       121 raj                                                 15000
       122 rani                                                14000
```

Try:
Create the trigger to delete an employee name 'raj'

Hint:
Use DML and DDL commands, triggers to delete the employee's name 'raj'

13. Working with PL/SQL Programming Cursors.

A cursor is a pointer to this context area. It contains all information needed for processing the statement. In PL/SQL, the context area is controlled by Cursor. A cursor contains information on a select statement and the rows of data accessed by it.
A cursor is used to referred to a program to fetch and process the rows returned by the SQL statement, one at a time. There are two types of cursors:

1. Implicit Cursors
2. Explicit Cursors

13.1 syntax for explicit cursor declaration

```
Declare the cursor:
Syntax for explicit cursor declaration
CURSOR name IS
 SELECT statement;
Open the cursor:
OPEN cursor_name;
Fetch the cursor:
FETCH cursor_name INTO variable_list;
Close the cursor:
Close cursor_name;
```

13.2 Implicit Cursor for customers salary update

```
DECLARE
  total_rows number(2);
BEGIN
  UPDATE  emp
  SET salary = salary + 5000;
  IF sql%notfound THEN
    dbms_output.put_line('no customers updated');
  ELSIF sql%found THEN
    total_rows := sql%rowcount;
    dbms_output.put_line( total_rows || ' customers updated ');
  END IF;
END;
/
```

13.3 Explicit cursor for displaying updated salary

```
DECLARE
  c_id emp.id%type;
  c_name emp.name%type;
  c_salary emp.salary%type;
  CURSOR c_customers is
    SELECT id, name, salary FROM emp;
BEGIN
  OPEN c_customers;
  LOOP
    FETCH c_customers into c_id, c_name, c_salary;
    EXIT WHEN c_customers%notfound;
    dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_salary);
  END LOOP;
  CLOSE c_customers;
END;
/
```

```
SQL> DECLARE
  2     c_id emp.id%type;
  3     c_name emp.name%type;
  4     c_salary emp.salary%type;
  5     CURSOR c_customers is
  6        SELECT id, name, salary FROM emp;
  7  BEGIN
  8     OPEN c_customers;
  9     LOOP
 10        FETCH c_customers into c_id, c_name, c_salary;
 11        EXIT WHEN c_customers%notfound;
 12        dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_salary);
 13     END LOOP;
 14     CLOSE c_customers;
 15  END;
 16  /
121 raj 25000
122 rani 24000

PL/SQL procedure successfully completed.
```

Try:
Create cursor to delete the employee whose salary is more than 24000

Hint:
Use DDL and DML commands with cursors to delete the employee.

V. REFERENCE BOOKS:

1. Abraham Silberschatz, Henry F. Korth, S. Sudarshan, "Database System Concepts", Mc raw-Hill, 4th edition,2002.
2. Ivan Bayross, "SQL, PL/SQL The programming language of oracle", BPB publications, 4th Revised edition, 2010.

VI. WEB REFERENCE:

I. Ramez Elmasri, Shamkant, B. Navathe, "Database Systems", Pearson Education, 6th edition, 2013.
II. Peter Rob, Carles Coronel, "Database System Concepts", Cengage Learning, 7th edition, 2008.
III. M L Gillenson, "Introduction to Database Management", Wiley Student edition,2012.

VIII. MATERIALS ONLINE

1. Course template
2. Lab Manual