

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**CSB4232 – DATABASE MANAGEMENT LABORATORY**

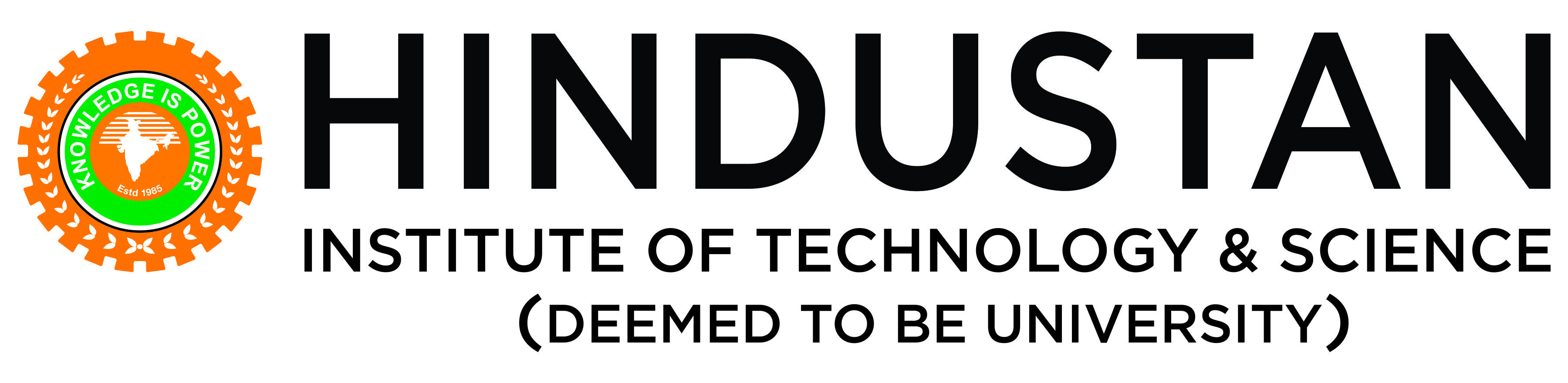
NAME : KONDREDDY SUMANTH KUMAR REDDY

REGISTER NUMBER : 19113105

SEMESTER : **III**

YEAR : **II**

BRANCH/SECTION : **CSE - B**



**LABORATORY RECORD**

19113105

**REG NO:**

Name of the lab **Database Management Systems** in the Department of **CSE**

Certified that this is a bonafide record of the work done by

**K. Sumanth Kumar Reddy** of **CSE 3B**Class in the

**virtual** laboratory during the year **2020 - 21**

……………………………

Signature of

Staff-in-charge

………………………….... …………………………...

Internal Examiner External Examiner

Name of the Examination : **University Practical Lab Examination**

Register No. : **19113105**

Date of the Examination : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EXP NO.** | **DATE** | **NAME OF THE EXPERIMENT** | **PAGE NO.** | **MARKS** |
| 1 | 29/07/20 | Creation of a table | 13-15 |  |
| 2 | 05/08/20 | Select and Update Commands | 16-18 |  |
| 3 | 05/08/20 | Alter and Delete commands | 19-21 |  |
| 4 | 12/08/20 | Command Conditions | 22-23 |  |
| 5 | 12/08/20 | Aggregate Functions | 24-25 |  |
| 6 | 19/08/20 | Grouping commands | 26-28 |  |
| 7 | 19/08/20 | Data constraints | 29-30 |  |
| 8 | 02/09/20 | Alising and Renaming | 31-32 |  |
| 9 | 02/09/20 | Commands for Join | 33-35 |  |
| 10 | 16/09/20 | Different set Operations | 36 |  |
| 11 | 16/09/20 | Scalar and String functions | 37-38 |  |
| 12 | 23/09/20 | Commands for view | 39-40 |  |
| 13 | 23/09/20 | Commands involving index | 41-42 |  |
| 14 | 14/10/20 | Conditional control statements | 43-45 |  |
| 15 | 14/10/20 | Procedural and Trigger | 46-47 |  |
| 16 | 28/10/20 | Case Study:  Hospital Management System | 48-53 |  |

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**VISION**

* To excel in Computer Science and Engineering education, research and project management by empowering the students with strong conceptual knowledge.

**MISSION**

* **M1.** To educate the students with basic foundation blocks of core and allied disciplines of Computer Science and Engineering.
* **M2.** To provide practical skills in the advancements of the Computer Science and Engineering field required for the growing dynamic IT and ITES industries.
* **M3.** To sculpt strong personal, technical, research, entrepreneurial, and leadership skills.
* **M4.** To inculcate knowledge in lifelong learning, professional ethics and contribution to the society.



**PROGRAMME EDUCATIONAL OBJECTIVES**

The B.Tech Programs offered by the Department of Computer Science & Engineering will meet the following objectives:

* **PEO 1**. Excel in his/her professional career and/or pursue higher education including research by applying the knowledge of Computer Science and Engineering
* **PEO 2.** Demonstrate the technical skills to analyze and design appropriate solutions for problems with social consciousness and ethical values.
* **PEO 3.** Adapt themselves to organizational needs by understanding the dynamically changing technologies.

**PROGRAMME SPECIFIC OUTCOMES**

The B.Tech Programs offered by the Department of Computer Science &Engineering will meet the following objectives:

* **PSO 1**. Apply mathematical, conceptual knowledge of computing and analytical skills to solve complex problems
* **PSO 2.** Design and develop computer systems based on the domains of cyber physical Systems, Algorithm design Techniques and Enterprise systems security
* **PSO 3.** Do innovative system design with analytical knowledge by developing modern tools and techniques



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

The B.Tech Programs offered by the Department of Computer Science & Engineering will meet the following outcomes

**PROGRAMME OUTCOMES**

**PO 1: Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

**PO 2: Problem analysis**: Identify, formulate, review research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

**PO 3: Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

**PO 4: Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

**PO 5: Modern tool usage**: Create, select, and apply appropriate techniques, resources, modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

**PO 6: The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

**PO 7: Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

**PO 8: Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

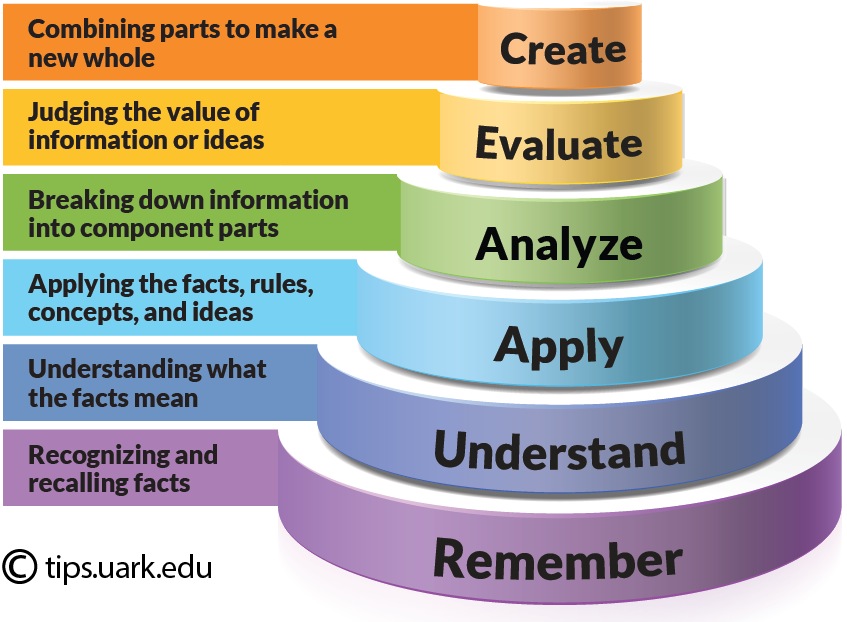
**PO 9: Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

**PO 10: Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

**PO 11: Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**PO 12: Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

**THE BLOOM’S TAXONOMY**



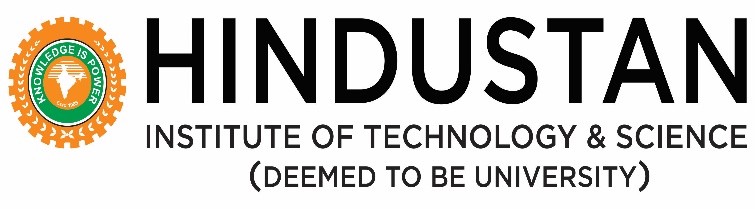
**The Bloom’s Taxonomy** is to be followed in curriculum development, courseware development, planning and delivery of contents, Assessment, Mapping, Data Analysis and CQI (Continuous Quality Improvement)

|  |
| --- |
| **COURSE OUTCOMES (CO)** |
| CO1:Populate and query a database using SQL commands. |
| CO2: .Declare and enforce integrity constraints on a database using a state-of-the-art RDBMS |
| CO3: Implementing Indexing on table |
| CO4:Programming PL/SQL including stored procedures, stored functions, cursors, packages |
| CO5: Solve basic issues of simple database applications and construct a real time database application using current techniques |

**MAPPING OF COs TO POs and PSOs**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S. No** | **COURSE OUTCOMES (CO)** | **PROGRAMME OUTCOMES (PO)** | | | | | | | | | | | | | | |
| **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO 11** | **PO12** | **PSO1** | **PSO2** | **PSO3** |
| 1 | Populate and query a database using SQL commands | **3** | **3** | **2** | **2** | **1** | **-** | **-** | **-** | **-** | **3** | **3** | **1** | **3** | **3** | **1** |
| 2 | Declare and enforce integrity constraints on a database | **3** | **3** | **3** | **2** | **1** | **-** | **-** | **-** | **-** | **3** | **3** | **1** | **3** | **2** | **1** |
| 3 | Implementing Indexing on table | **3** | **2** | **2** | **2** | **1** | **-** | **-** | **-** | **-** | **3** | **3** | **1** | **3** | **2** | **1** |
| 4 | Programming PL/SQL including stored procedures, stored functions, cursors | **3** | **2** | **2** | **2** | **1** | **-** | **-** | **-** | **-** | **3** | **3** | **1** | **3** | **2** | **1** |
| 5 | Solve basic issues of simple database applications and construct a real time database application | **3** | **3** | **2** | **2** | **1** | **-** | **-** | **-** | **-** | **3** | **3** | **1** | **3** | **2** | **1** |

**3 - Strong Correlation 2 - Medium Correlation 1 - Weak Correlation - No correlation**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**LIST OF EXPERIMENTS**

1. To study Basic SQL commands (create table, use , drop, insert) and execute the following queries using these commands:
2. To study the viewing commands (select, update) and execute the following queries using these commands:
3. To study the commands to modify the structure of table (alter, delete) and execute the following queries using these commands:
4. To study the commands that involve compound conditions (and, or, in , not in, between , not between , like , not like) and execute the following queries using these commands:
5. To study the aggregate functions (sum, count, max, min, average) and execute the following queries using these commands:
6. To study the grouping commands (group by, order by) and execute the following queries using these commands:
7. To study the commands involving data constraints and execute the following queries using these commands:
8. To study the commands for joins ( cross join, inner join, outer join) and execute the following queries using these commands:
9. To study the various set operations and execute the following queries using these commands:
10. To study the various scalar functions and string functions ( power, square, substring, reverse, upper, lower, concatenation) and execute the following queries using these commands:
11. To study the commands involving indexes and execute the following queries:
12. To study the conditional controls and case statement in PL-SQL and execute the following queries:
13. To study procedures and triggers in PL-SQL and execute the following queries:
14. Consider the tables given below. The primary keys are made bold and the data types are specified.

PERSON ( driver\_id:string , name:string , address:string )

CAR ( regno:string , model:string , year:int )

ACCIDENT (report\_number:int, accd\_date:date , location:string )

OWNS (driver\_id:string , regno:string )

PARTICIPATED (driver\_id:string , regno:string , report\_number:int , damage\_amount:int)

a. Create the above tables by properly specifying the primary keys and foreign keys.

b. Enter at least five tuples for each relation.

c. Demonstrate how to update the damage amount for the car with specific regno in the accident with report number 12 to 25000.

d. Find the total number of people who owned cars that were involved in accidents in the year 2008. e. Find the number of accidents in which cars belonging to a specific model were involved

1. Case study with scenario for studying the SQL queries in detail.

**Experiment No:** 01  **CREATION OF A TABLE**

**Date:** 29/07/20

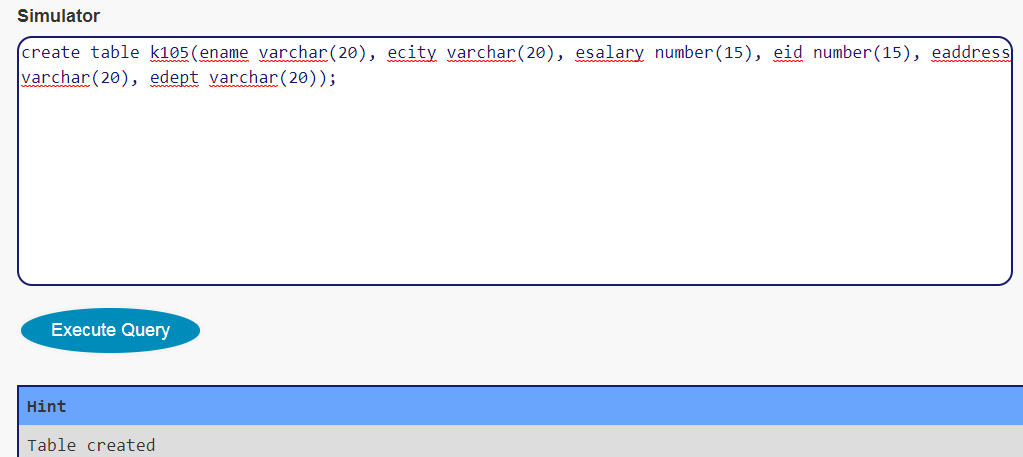
**Aim:** To create the basic SQL commands.

1. Create a basic employee with attributes ename, esalary, eid, eaddress, edept where salary and eid are integer and others are characters.
2. Create another table company with the attributes ename, ecity, eid, eaddress. Insert 5 enteries in each table.

**Queries: Employee table**

**SQL>** Create table k105(ename varchar(20), ecity varchar(20), esalary number(15), eid number(15), eaddress varchar(20), edept varchar(20));

Table created.



**SQL>** Insert into k105 values(‘sumanth’,’chennai’,54000,105,’padur’,’cse’);

1 row created.

**SQL>** Insert into k105 values(‘umesh’,’tirupati’,60000,118,’pakala’,’cse’);

1 row created.

**SQL>** Insert into k105 values(‘sai manoj’,’tirupati’,56000,101,’mr palli’,’cse’);

1 row created.

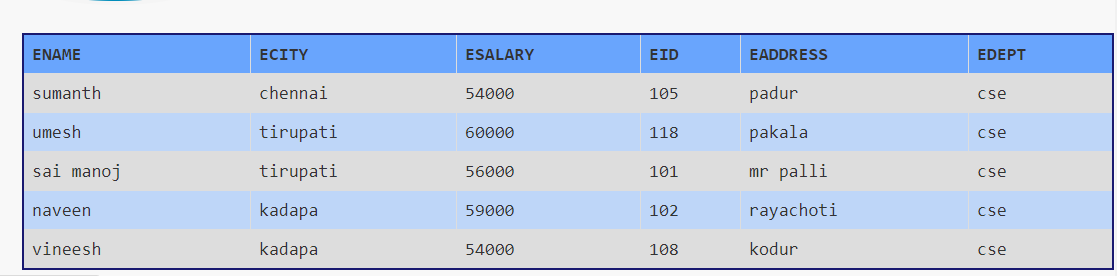
**SQL>** Insert into k105 values(‘naveen’,’kadapa’,’59000’,’102’,’rayachoti’,’cse’);

1 row created.

**SQL>** Insert into k105 values(‘vineesh’,’kadapa’,’54000’,’108’,’kodur’,’cse’);

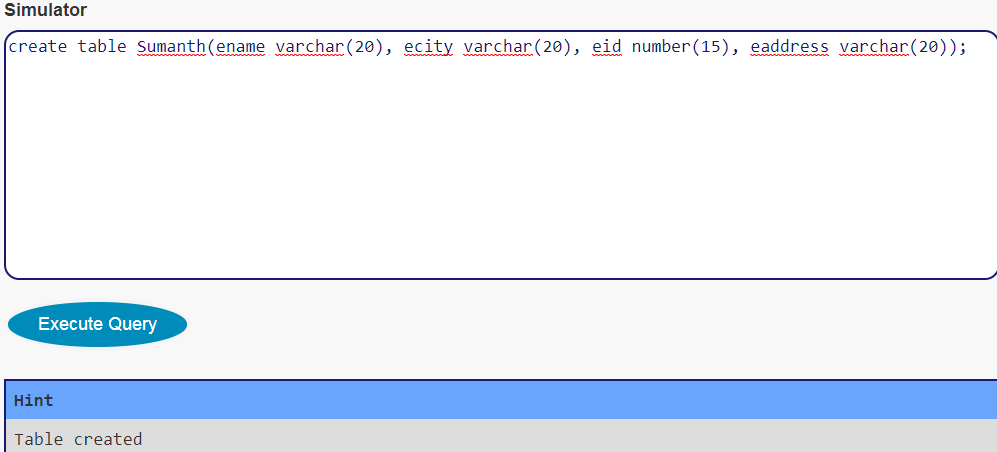
1 row created.

**SQL>** select \* from k105;



**Company Table**

**SQL>** Create table Sumanth(ename varchar(20), ecity varchar(20), eid number(15), eaddress varchar(20));



**SQL>** Insert into Sumanth values(‘milan’,’chennai’,106,’padur’);

1 row created.

**SQL>** Insert into Sumanth values(‘tharun’,’mumbai’,092,’pune’);

1 row created.

**SQL>** Insert into Sumanth values(‘kowshik’,’kolkata’,081,’kolkata’);

1 row created.

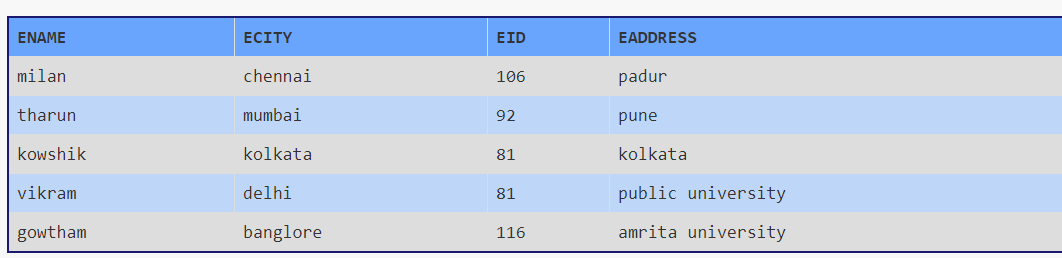
**SQL>** Insert into Sumanth values(‘vikram’,’delhi’,107,’’);

1 row created.

**SQL>** Insert into Sumanth values(‘gowtham’,’banglore’,116,’amrita university’);

1 row created.

**SQL>** select \* from Sumanth;



**Result:** Thus, the creation of table is run & executed.

**Experiment No:** 02 **SELECT AND UPDATE COMMANDS**

**Date:** 05/08/20

**Aim:** To study the view commands such as select, update and execute the queries.

1. Find the names of all employees who live in Chennai.
2. Increase the salary of all employess by 20000.
3. Find the employee name whose salary is greater than 50000.
4. Change the employee city to Chennai where employee name is Sumanth.

**Queries:**

**SQL>** Create table emp(ename varchar(20), esalary number(15), ecity varchar(20));

Table created.

**SQL>** Insert into emp values(‘sumanth’,50000,’pune’);

1 row created.

**SQL>** Insert into emp values(‘mahesh’,54000,’tirupati’);

1 row created.

**SQL>** Insert into emp values(‘sudheer’,51000,’chennai’);

1 row created.

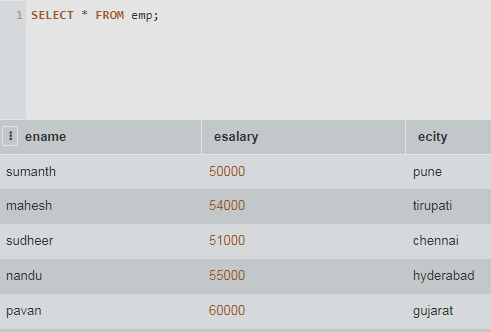
**SQL>** Insert into emp values(‘nandu’,55000,’hyderabad’);

1 row created.

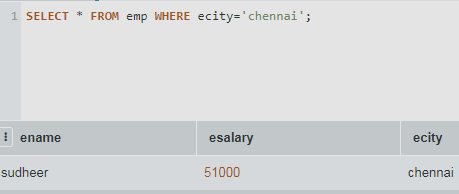
**SQL>** Insert into emp values(‘pavan’,60000,’gujarat’);

1 row created.

**SQL>** select \* from emp;



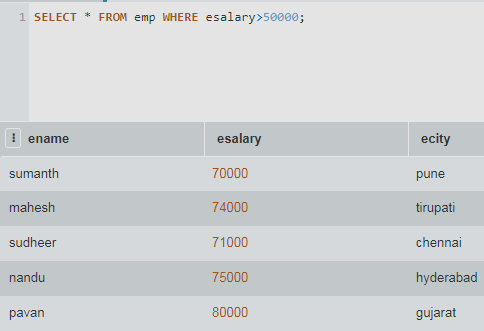
1. Select \* where from emp where ecity=’chennai’;



1. Update emp set esalary=esalary+20000;

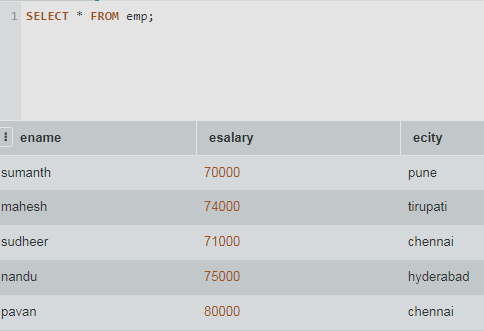


1. Select \* from emp where esalary>50000;



1. Update emp set ecity=’chennai’ where ename=’sudheer’;

Select \* from emp;



**Result:** Thus, select and update commands are executed.

**Experiment No:** 03 **ALTER AND DELETE COMMANDS**

**Date:** 05/08/20

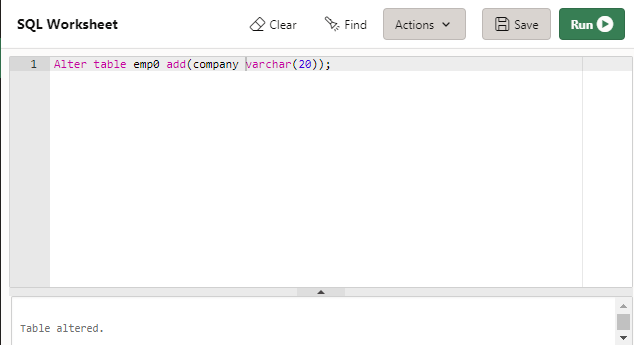
**Aim:** To study the commands to modify the structures of tables and execute queries.

**Queries:**

1. Add an attribute, named designation to the employee table.
2. Modify the employee table and change the data type of ‘salary’ attribute to float.
3. Drop the attribute eid from the employee table.
4. Delete the entities from employee table where salary is less than 20000.

**SQL>** Alter table emp0 add(company varchar(20));

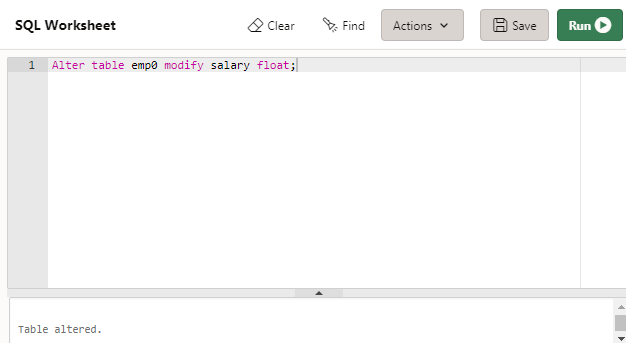
Table altered.



**SQL>** desc emp0;

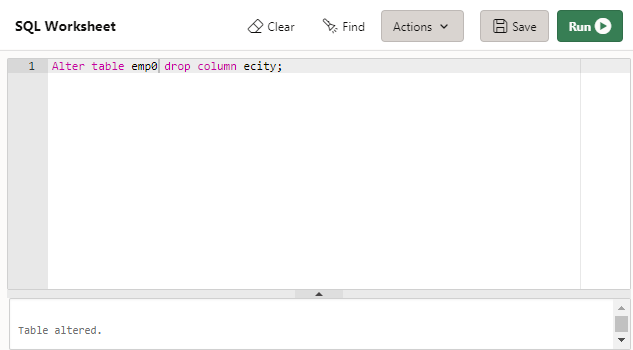
**SQL>** Alter table emp0 modify salary float;

Table altered.



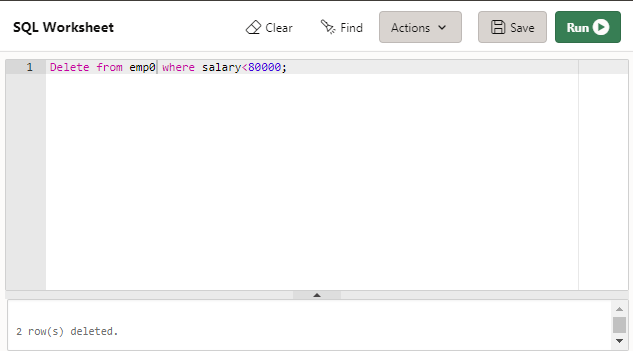
**SQL>** Alter table emp0 drop column eaddress;

Table altered.

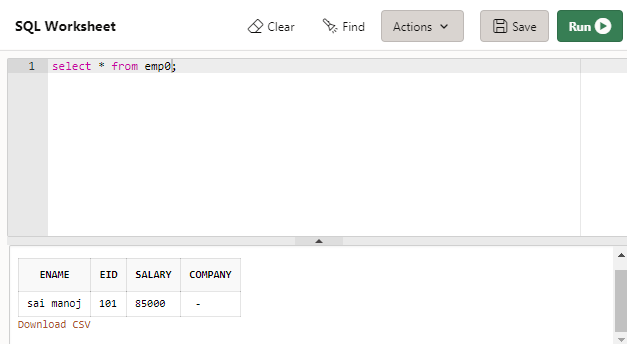


**SQL>** Delete from emp0 where salary<80000;

1 row deleted.



**SQL>** select \* from emp0;



**Result:** Thus, all the queries are verified and executed successfully.

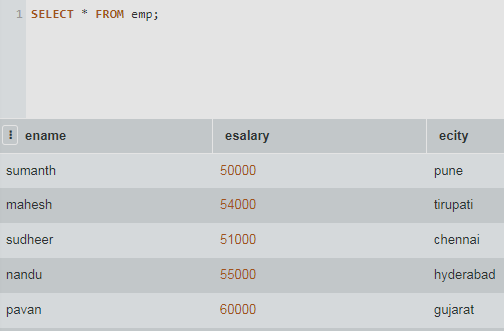
**Experiment No:** 04 **COMMAND CONDITIONS**

**Date:** 12/08/20

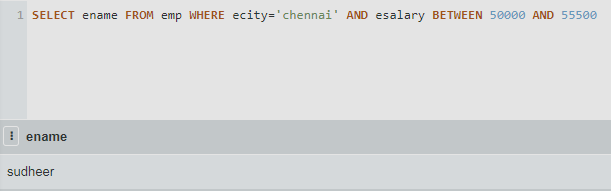
**Aim:** To study the command that involve compound conditions such as AND, OR, IN, NOT, BETWEEN, Like, Not Like, Not in, Not Between.

**Queries:**

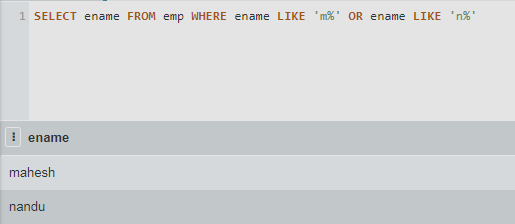
1. Find the enames of employees who like in Chennai and whose salary is in between 50000 and 55500.
2. Find the names od all employees whose names begin in with ‘A’ or ’S’.
3. Find the company name whether the company city is Chennai and salary is not between 55000 and 59500.
4. Find the names of all companies that do not end with with letter ‘A’.



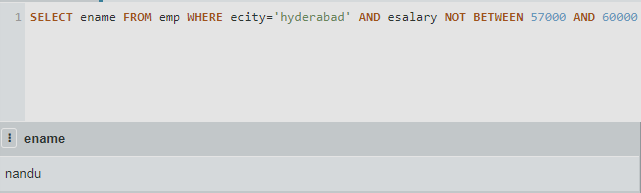
**SQL>** select ename from emp where ecity=’chennai’ and esalary between 50000 and 55500.



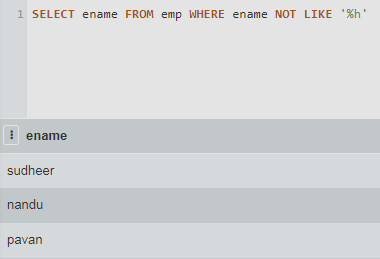
**SQL>** select ename from emp where ename like ‘m%’ or ename like ‘n%’.



**SQL>** select ename from emp where ecity=’hyderabad’ and esalary not between 57000 and 60000.



**SQL>** select ename from emp where ename not like ‘%h’.



**Result:**  Thus, all the queries has been executed successfully.

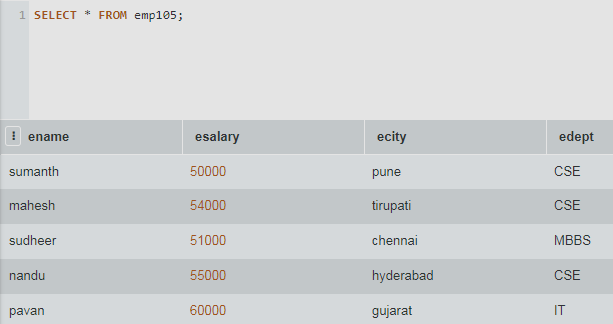
**Experiment No:** 05 **AGGREGATE FUNCTIONS**

**Date:** 12/08/20

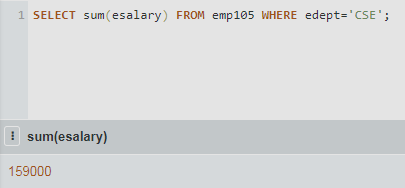
**Aim:** To study the Aggregate functions such as Sum, Count, Max., Min., Avg. And to execute the following queries.

**Query**

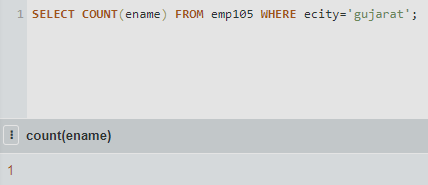
1. Find the sum and average of salaries of all employees in testing department.
2. Find the number of all employees who live Chennai.
3. Find the out the max., min. And salary from CSE department.



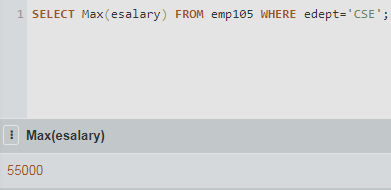
**SQL>** select sum(esalary) from emp105 where edept=’CSE’;



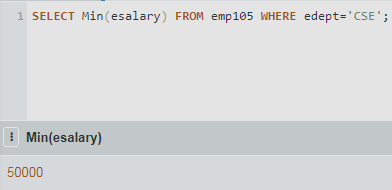
**SQL>** select count(ename) from emp105 where ecity=’gujarat’;



**SQL>** select Max(esalary) from emp105 where edept=’CSE’;



**SQL>** select Min(esalary) from emp105 where edept=’CSE’;



**Result:** Thus, all the queries has been executed successfully.

**Experiment No:** 06 **GROUPING COMMANDS**

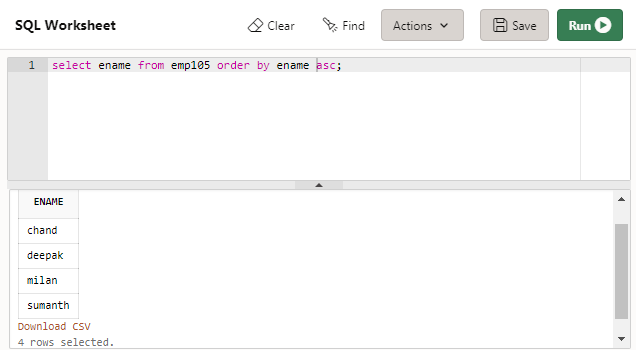
**Date:** 19/08/20

**Aim:** To study grouping commands such as group by and order by execute the queries.

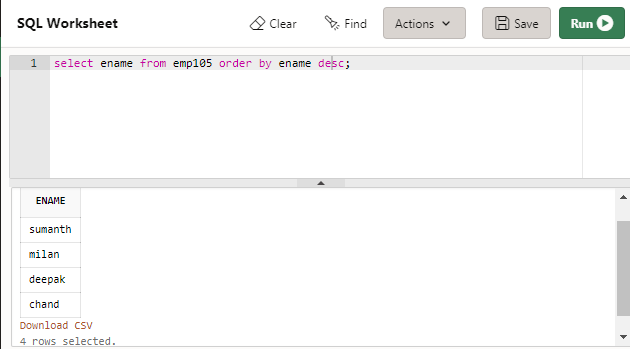
**Queries:**

1. List out all employee names in Ascending or Descending order.
2. Find the no. of employees in each dept. where, the no. of employees is greater than 5.
3. List out all dept names where avg. Salary of dept. Is greater than 10000.
4. List out the no. of employees grouping by dept. In descending order.

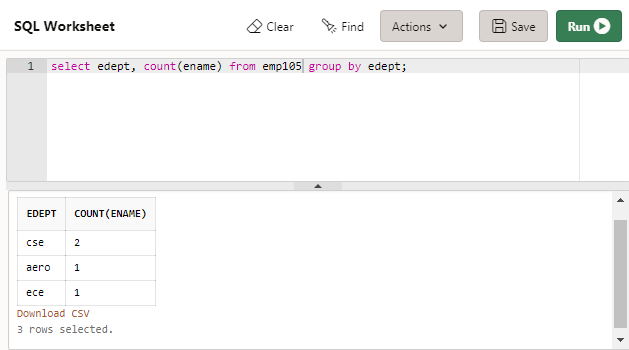
**SQL>** select ename from s105 order by ename asc;



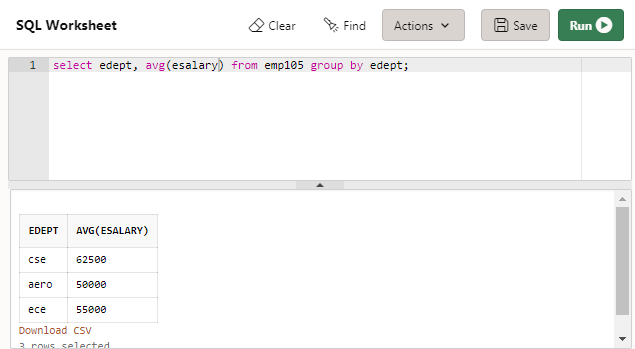
**SQL>** select ename from s105 order by ename desc;



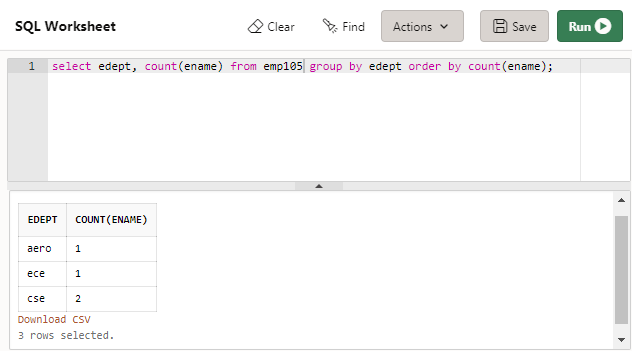
**SQL>** select edept, count(ename) from s105 group by edept;



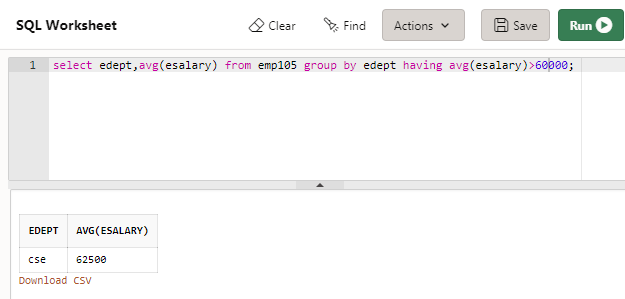
**SQL>** select edept, avg(esalary) from s105 group by edept;



**SQL>** select edept, count(ename) from s105 group by edept order by count(ename);



**SQL>** select edept,avg(esalary) from s105 group by edept having avg(esalary)>60000;



**Result:** Thus, all the queries are executed successfully.

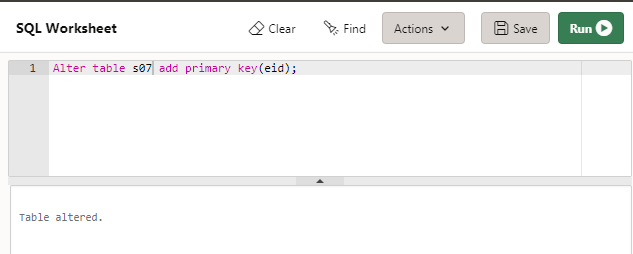
**Experiment No:** 07 **DATA CONSTRAINTS**

**Date:** 19/08/20

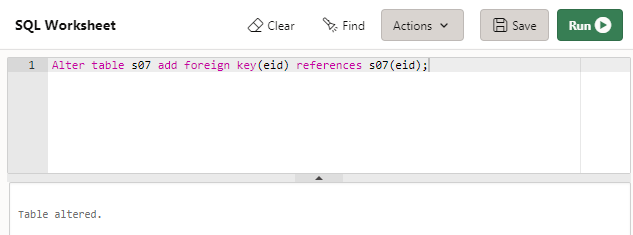
**Aim:** To study the commands that involved data constraints and to execute the following queries.

1. Alter the table employee and make eid as the primary key.
2. Alter the table company and add foreign key.
3. Add a check constrains in the table employee such that salary has the value between 0 to 100000.
4. Alter the table company and add unique constrains to column ename.
5. Add a defult constrains to column ecity(company) chennai.

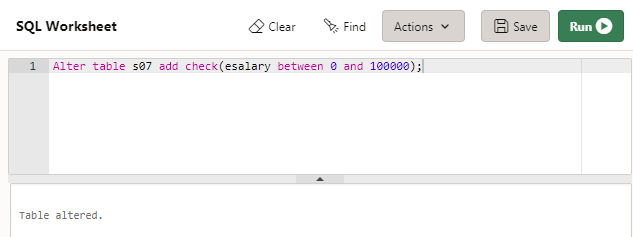
**SQL>** Alter table s07 add primary key(eid);



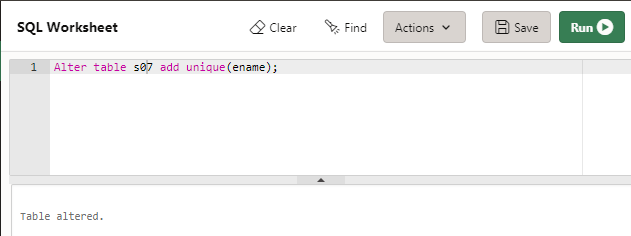
**SQL>** Alter table s07 add foreign key(eid) references s07(eid);



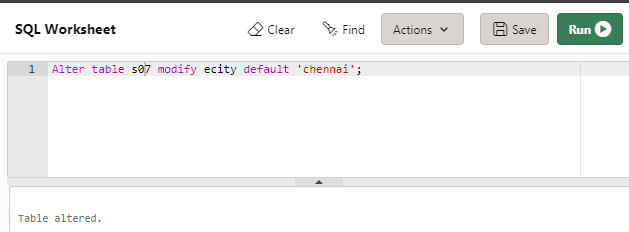
**SQL>** Alter table s07 add check(salary between 0 and 100000);



**SQL>** Alter table s07 add unique(ename);



**SQL>** Alter table so7 modify ecity default ’chennai’;



**Result:** Thus, all the queries are executed successfully.

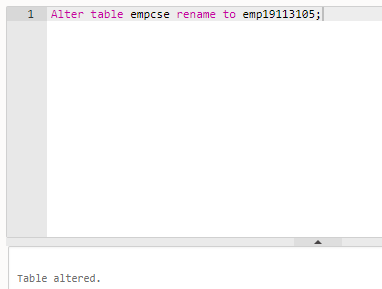
**Experiment No:** 08 **ALISING AND RENAMING**

**Date:** 02/09/20

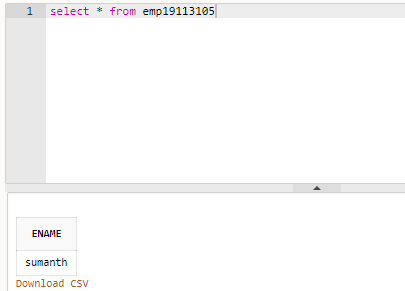
**Aim:** To study the commands for alising and renaming for execute the following the queries.

1. Rename the employee table to emp19113105
2. Change the name of the attribute ename to name

**SQL>** Alter table empcse rename to emp19113105;

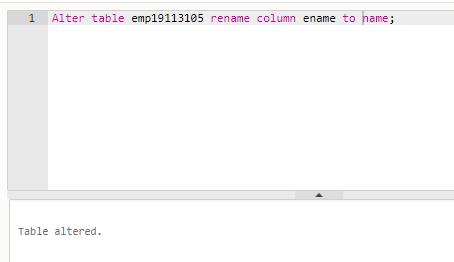


**SQL>** desc emp19113105;



**SQL>** Alter table emp19113105 rename column ename to name;

**SQL>** Table altered.



**Result:** Thus, all the queries has been executed successfully.

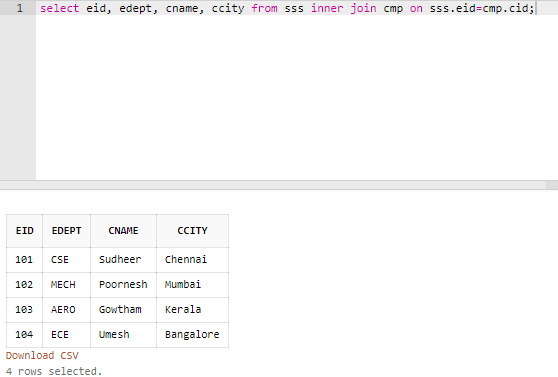
**Experiment No:** 09 **COMMANDS FOR JOIN**

**Date:** 02/09/20

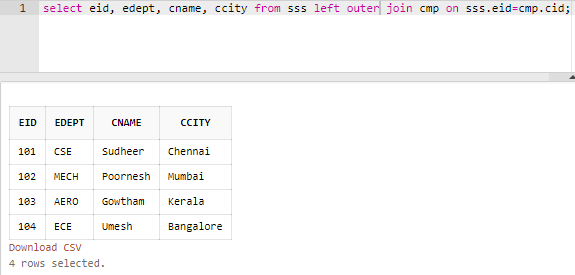
**Aim:** To study the commands for join such as Inner join and Outer join to execute the following queries.

1. Retrieve the complete records of the employees and their companies from the both tables using join.
2. Retrieve the complete records of of the employees and their companies from the both tables using tools.

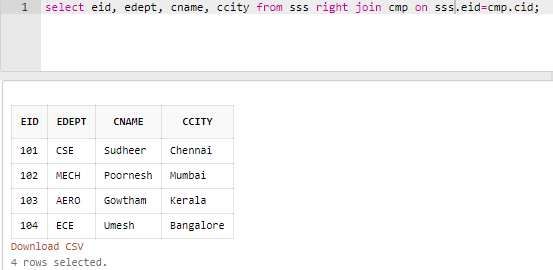
**SQL>** select eid, edept, cname, ccity from sss inner join cmp on sss.eid=cmp.cid;



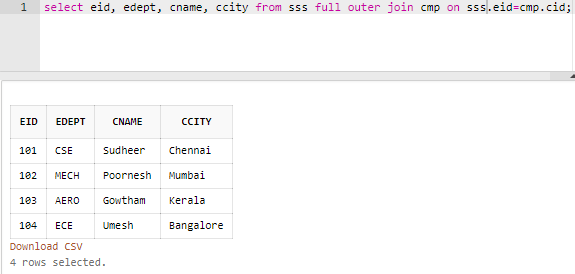
**SQL>** select eid, edept, cname, ccity from sss outer join cmp on sss.eid=cmp.cid;



**SQL>** select eid, edept, cname, ccity from sss left outer join cmp on sss.eid=cmp.cid;



**SQL>** select eid, edept, cname, ccity from sss full outer join cmp on sss.eid=cmp.cid;



**Result:** Thus, all the queries for commands for join has been executed successfully.

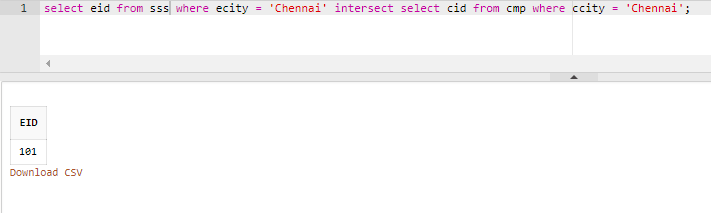
**Experiment No:** 10 **DIFFERENT SET OPERATIONS**

**Date:** 16/09/20

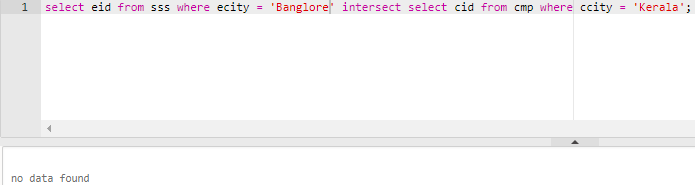
**Aim:** Study the different set of operations to execute the following queries.

1. Find out all the employees who lives in Chennai & company is in Bangalore.
2. List out the number of employees who lives in Kolkata and whose company is not in Bangalore.
3. Select eid from employees where ecity=’chennai’ intersect select cid from company where the ccity=’chennai’.

**SQL>** select eid from employee where ecity=’chennai’ intersect select cid from company where ccity=’bangalore’;



**SQL>** select eid from emp where ecity = 'Chennai' intersect select cid from cmp where ccity = 'Kerala';

****

**Result:** Thus, the study of various set of operations has been executed successfully.

**Experiment No:** 11 **SCALAR AND STRING FUNCTIONS**

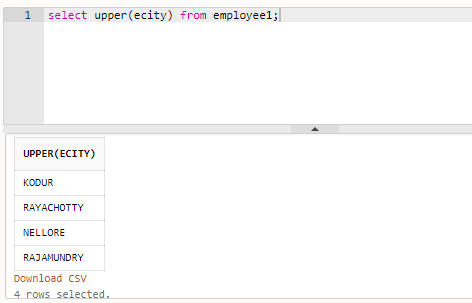
**Date:** 16/09/20

**Aim:** To study the scalar, string functions such as reverse, upper, lower and concat.

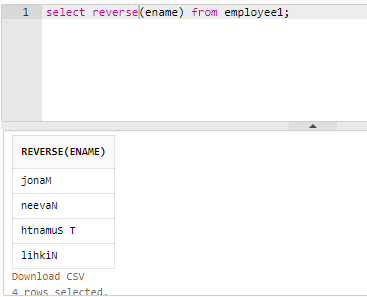
**Queries:**

1. Reverse the names of all employees in the employee table.
2. Display the names of all the cities in the company table in upper case.
3. Concat the names and cities of all the employees.

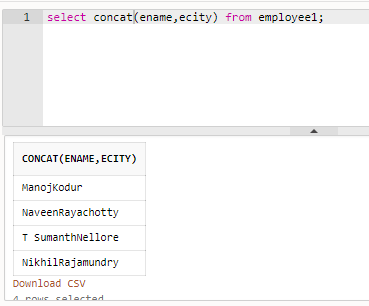
**SQL>** select upper(ecity) from employee1;



**SQL>** select reverse(ename) from employee1;



**SQL>** select concat(ename,ecity) from employee1;



**Result:** Thus, scalar functions and string functions has been executed successfully.

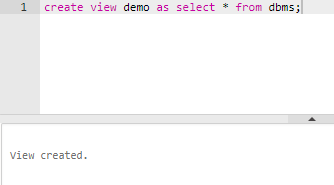
**Experiment No:** 12 **COMMANDS FOR VIEW**

**Date:** 23/09/20

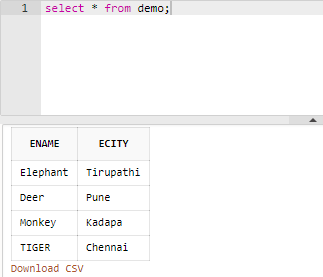
**Aim:** To study the commands for view and to execute the following queries

1. Create a view having ename and ecity.
2. In the above view change the city to Delhi where ename is Tiger.
3. Create a view that includes attributes from both the tables.

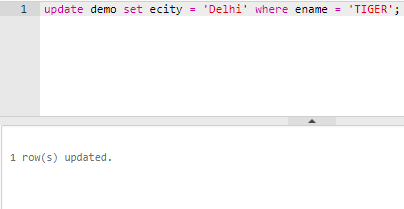
**SQL>** create view demo as select \* from dbms;



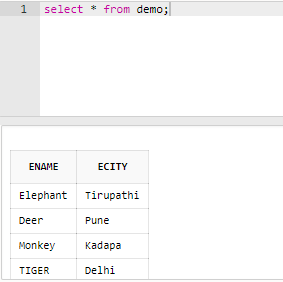
**SQL>** select \* from demo;



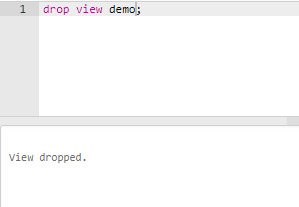
**SQL>** update demo set ecity = 'Delhi' where ename = 'TIGER';



**SQL>** select \* from demo;



**SQL>** drop view demo;



**Result:** Thus, all the queries have been verified and executed successfully.

**Experiment No:** 13 **COMMANDS INVOLVING EDGE**

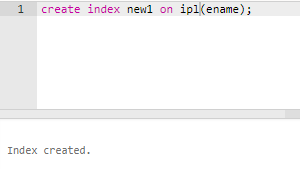
**Date:** 23/09/20

**Aim:** To study the commands involving indexes and execute queries.

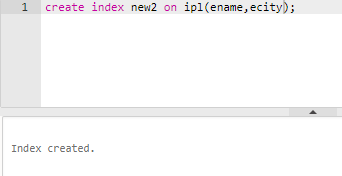
**Queries:**

1. Create index with attribute ename on table employee.
2. Create a composite index with attribute ename and ecity on table company.
3. Drop the index.

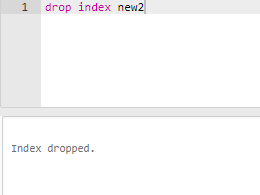
**SQL>** create index new1 on sample1(ename);



**SQL>** **create index new2 on sample1(ename, ecity);**

****

**SQL>** drop index new2



**Result:** The given queries has been executed successfully.

**Experiment No:** 14 **CONDITIONAL CONTROL STATEMENTS**

**Date:** 14/10/20

**Aim:** To Study the CONDITIONAL CONTROL commands statement using PL/SQL and to execute the following queries.

**Queries:**

1. Write a PL/SQL program using if else if and display command on day of the week.
2. Write a PL/SQL program to categorize the first 10 natural numbers as odd or even.

**Program:**

(i) declare

s varchar(10):=’DBMS LAB’;

begin

if (s='Sumanth') then

dbms\_output.put\_line('enjoy');

elsif (s='DBMS LAB') then

dbms\_output.put\_line('New');

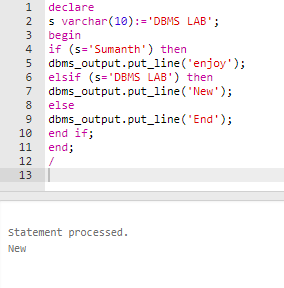
else

dbms\_output.put\_line('End');

end if;

end;

/



(ii) declare

x number := 0;

begin

for x in 1..15 loop

if mod(x,2)=0 then

dbms\_output.put\_line ('Number is even:'|| x);

else

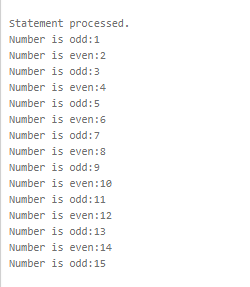
dbms\_output.put\_line ('Number is odd:'|| x);

end if;

end loop;

end;

/



**RESULT:** Thus, the queries are verified and executed successfully.

**Experiment No:** 15 **PROCEDURAL AND TRIGGER**

**Date:** 14/08/20

**Aim:** To Study the PROCEDURAL and TRIGGER in PL/SQL.

**CREATION AND EXECUTION OF PROCEDURE:**

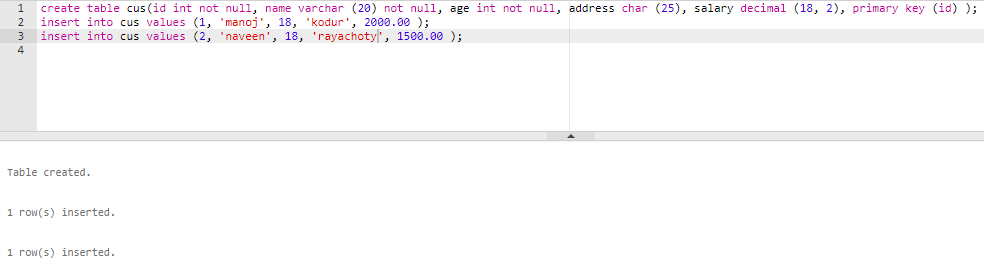
* Create a table Student with rollno, mark1, mark2, total with roll no as the primary key.
* Insert values for 1,2,3 then
* Write a procedure Student to update the total.

**Queries:**

**SQL>** create table cus(id int not null, name varchar (20) not null, age int not null, address char (25), salary decimal (18, 2), primary key (id) );

**SQL>** insert into cus values (1, 'manoj', 18, 'kodur', 2000.00 );

**SQL>** insert into cus values (2, 'naveen', 18, 'rayachoty', 1500.00 );



DECLARE

c\_id cus.id%type := 2;

c\_name cus.name%type;

c\_addr cus.address%type;

c\_sal cus.salary%type;

BEGIN

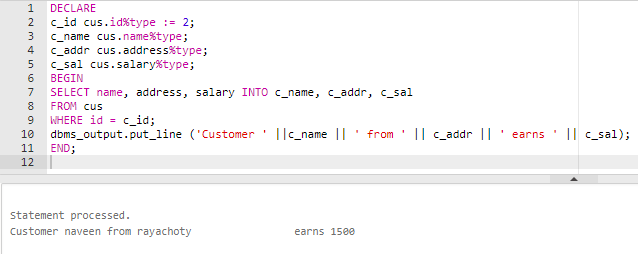
SELECT name, address, salary INTO c\_name, c\_addr, c\_sal

FROM cus

WHERE id = c\_id;

dbms\_output.put\_line ('Customer ' ||c\_name || ' from ' || c\_addr || ' earns ' || c\_sal);

END;



**Result:** Thus, the queries are verified and executed successfully.

**Experiment:** 16 **CASE STUDY (HOSPITAL MANAGEMNT SYSTEM)**

**Date:** 28/10/20

**Aim:** Arogya hospital is a multi specialty hospital that includes a number of departments, rooms, doctors, nurses, compounders, and other staff working in the hospital. Patients having different kinds of ailments come to the hospital and get checkup done from the concerned doctors. If required they are admitted in the hospital and discharged after treatment. There is also special care for corona patients.

The aim of this case study is to design and develop a database for the hospital to maintain the records of various departments, rooms, and doctors in the hospital. It also maintains records of the regular patients, patients admitted in the hospital, the check up of patients done by the doctors, the patients that have been operated, and patients discharged from the hospital.

Description: In hospital, there are many departments like Orthopedic, Pathology, Emergency, Dental, Gynecology, Anesthetics, I.C.U., Blood Bank, Operation Theater, Laboratory, M.R.I., Neurology, Cardiology, Cancer Department, Corpse, etc. There is an OPD where patients come and get a card (that is, entry card of the patient) for check up from the concerned doctor.

After making entry in the card, they go to the concerned doctor’s room and the doctor checks up their ailments. According to the ailments, the doctor either prescribes medicine or admits the patient in the concerned department. The patient may choose either private or general room according to his/her need. But before getting admission in the hospital, the patient has to fulfill certain formalities of the hospital like room charges, etc. After the treatment is completed, the doctor discharges the patient. Before discharging from the hospital, the patient again has to complete certain formalities of the hospital like balance charges, test charges, operation charges (if any), blood charges, doctors’ charges, etc.

Next we talk about the doctors of the hospital. There are two types of the doctors in the hospital, namely, regular doctors and call on doctors. Regular doctors are those doctors who come to the hospital daily. Calls on doctors are those doctors who are called by the hospital if the concerned doctor is not available.

**Queries :**

1 .DEPARTMENT TABLE:

**SQL>** create table DEPARTMENT(D\_NAME varchar2(4000) constraint department primary key, D\_LOCATION varchar2(4000),FACILITIES varchar2(4000));

**SQL>** insert into DEPARTMENT values('Orthopedic','Guindy','Physio Consultation & Treatment Room');

**SQL>** insert into DEPARTMENT values('Dental','OMR','Replacement of missing teeth');

**SQL>** insert into DEPARTMENT values('Emergency','Nungambakkam','Minor Procedure Room');

**SQL>** insert into DEPARTMENT values('OPD','Guindy','Treatment of outpatients');

**SQL>**select \* from DEPARTMENT;

Table

Description automatically generated

2.ALL\_DOCTORS TABLE:

**SQL>** create table ALL\_DOCTORS2(DOC\_NO number constraint all\_doctors primary key, DEPARTMENT varchar(400);

**SQL>** insert into ALL\_DOCTORS2 values(101,'Dental');

**SQL>** insert into ALL\_DOCTORS2 values(223,'Orthopedic');

**SQL>** insert into ALL\_DOCTORS2 values(554,'Emergency');

**SQL>** insert into ALL\_DOCTORS2 values(234,'OPD');

**SQL>** select \* from ALL\_DOCTORS2;

Graphical user interface, application

Description automatically generated

3.DOC\_REG TABLE:

**SQL>** create table DOC\_REG ( DOC\_NO varchar2(4000),SALARY varchar2(4000));

**SQL>** insert into DOC\_REG values('DRIN12','INDRANI','B Sc. & Post Graduate ADA Recognized','65000','8:00 am','7:00 pm','Padur','8741241237','1993-06-23');

**SQL>** insert into DOC\_REG values('DRRA43','Rahul','PDCC Ortho Anaesthesiology','81000','10:00 am','5:00 pm','Roypetta','9231241237','1990-01-31');

**SQL>** insert into DOC\_REG values('DRJA21','James','B Sc. & Post Graduate','68000','8:00 am','2:00 pm','Avadi','7941261231','1991-03-14');

**SQL>** insert into DOC\_REG values('DRSU01','Susan','B Sc. & Post Graduate','50000','6:00 am','12:00 pm','Guindy','7441241237','1995-07-07');

select \* from DOC\_REG;



4.DOC\_ON\_CALL TABLE:

**SQL>** create table DOC\_ON\_CALL( DOC\_NO varchar2(4000),PYMT\_DU varchar2(4000),FS\_PR\_CL varchar2(4000),ADDRESS varchar2(4000));

**SQL>** insert into DOC\_ON\_CALL values('DCDX','Dax','B Sc. & Post Graduate','45000','8000','Avadi','712261231');

**SQL>** insert into DOC\_ON\_CALL values('DCZY','Ziyech','B Sc. & Post Graduate','30000','4000','Guindy','8441241237');

**SQL>** select \* from DOC\_ON\_CALL;



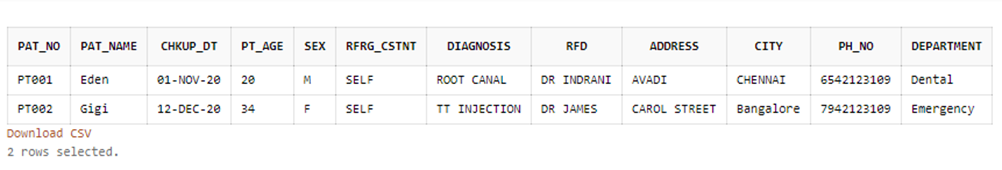
5.PAT\_ENTRY TABLE:

**SQL>** create table PAT\_ENTRY(PAT\_NO varchar2(4000) constraint pat\_entry primary key,PAT\_NAME varchar2(4000),CHKUP\_DT date,PT\_AGE number,SEX varchar2(4000),RFRG\_CSTNT varchar2(4000),DIAGNOSIS varchar2(4000),RFD varchar2(4000),ADDRESS varchar2(4000),CITY varchar2(4000),PH\_NO number,DEPARTMENT varchar2(4000));

**SQL>** insert into PAT\_ENTRY values('PT001','Eden',DATE '2020-11-01',20,'M','SELF','ROOT CANAL','DR INDRANI','AVADI','CHENNAI',6542123109,'Dental');

**SQL>** insert into PAT\_ENTRY values('PT002','Gigi',DATE '2020-12-12',34,'F','SELF','TT INJECTION','DR JAMES','CAROL STREET','Bangalore',7942123109,'Emergency');\

**SQL>** select \* from PAT\_ENTRY;

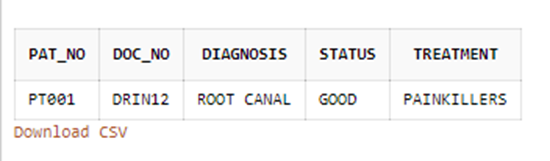


6.PAT\_CHKUP TABLE:

**SQL>** create table PAT\_CHKUP (PAT\_NO varchar2(4000),DOC\_NO varchar2(4000),DIAGNOSIS varchar2(4000),STATUS varchar2(4000),TREATMENT varchar2(4000));

**SQL>** insert into PAT\_CHKUP values('PT001','DRIN12','ROOT CANAL','GOOD','PAINKILLERS');

**SQL>** select \* from PAT\_CHKUP;

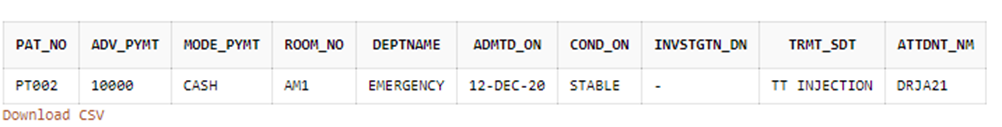


7.PAT\_ADMIT TABLE:

**SQL>** create table PAT\_ADMIT(PAT\_NO varchar2(4000),ADV\_PYMT varchar2(4000),ODE\_PYMT varchar2(4000),ROOM\_NO number, DEPTNAME varchar2(4000),ADMTD\_ON date,COND\_ON varchar2(4000), INVSTGTN\_DN varchar2(4000),TRMT\_SDT varchar2(4000),ATTDNT\_NM number);

**SQL>** insert into PAT\_ADMIT VALUES('PT002','10000','CASH','AM1','EMERGENCY',DATE '2020-12-12','STABLE','-','TT INJECTION','DRJA21');

**SQL>** select \* from PAT\_ADMIT;

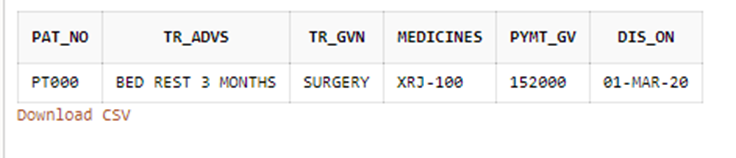


8.PAT\_DIS TABLE:

**SQL>** create table PAT\_DIS (PAT\_NO varchar2(4000),TR\_ADVS varchar2(4000),TR\_GVN varchar2(4000),MEDICINES varchar2(4000),PYMT\_GV number,DIS\_ON date);

**SQL>** insert into PAT\_DIS values('PT000','BED REST 3 MONTHS','SURGERY','XRJ-100','152000',DATE '2020-03-01');

**SQL>** select \* from PAT\_DIS;

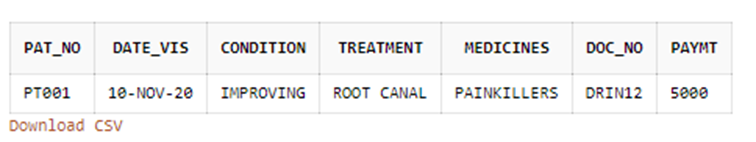


9.PAT\_REG TABLE:

**SQL>** create table PAT\_REG(PAT\_NO varchar2(4000),DATE\_VIS varchar2(4000),CONDITION varchar2(4000),TREATMENT varchar2(4000),MEDICINES varchar2(4000),DOC\_NO varchar2(4000),PAYMT number);

**SQL>** insert into PAT\_REG values('PT001',DATE '2020-11-10','IMPROVING','ROOT CANAL','PAINKILLERS','DRIN12',5000);

**SQL>** select \* from PAT\_REG;

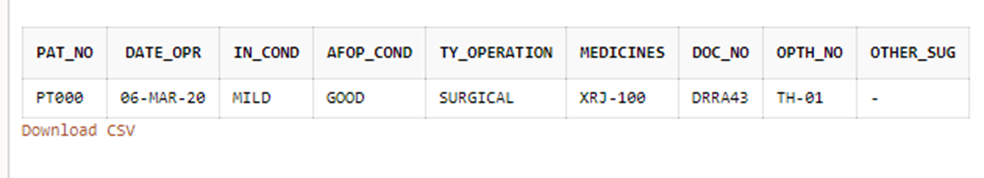


10.PAT\_OPR TABLE:

**SQL>** create table PAT\_OPR (PAT\_NO varchar2(4000),DATE\_OPR date,IN\_COND varchar2(4000),AFOP\_COND varchar2(4000),TY\_OPERATION timestamp,MEDICINES varchar2(4000),DOC\_NO varchar2(4000),OPTH\_NO varchar2(4000),OTHER\_SUG varchar2(4000));

**SQL>** insert into PAT\_OPR values('PT000',DATE '2020-03-06','MILD','GOOD','SURGICAL','XRJ-100','DRRA43','TH-01','-');

**SQL>** select \* from PAT\_OPR;



11.ROOM\_DETAILS TABLE:

**SQL>** create table ROOM\_DETAILS (ROOM\_NO varchar2(4000) constraint room\_details\_pk primary key,TYPE varchar2(4000),STATUS varchar2(4000),RM\_DL\_CRG varchar2(4000),OTHER\_CRG varchar2(4000));

**SQL>** alter table ROOM\_DETAILS add constraint room\_details\_room\_no\_uq unique (ROOM\_NO);

**SQL>** insert into ROOM\_DETAILS values('AM1','P','Y',2500,500);

**SQL>** insert into ROOM\_DETAILS values('AM2','G','N',1500,100);

**SQL>** insert into ROOM\_DETAILS values('AM3','P','N',1500,500);

**SQL>** select \* from ROOM\_DETAILS;

