GLOBAL STANDARDS GIET

An Autonomous Institution
NBA Accredited & NAAC A+

# GODAVARI INSTITUTE OF ENGINEERING ANDTECHNOLOGY

Department of **COMPUTER SCIENCE AND ENGINEERING**

Name:__________________________________ PinNo:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

## CERTIFICATE

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

Mr./Ms.__________________________________________________________

a Student of_____________________________________with PinNo:____________________

in the______________________________Laboratory during the Academic year_____________

No. of Experiments Conducted: ☐☐ No. of Experiments attended: ☐☐

Faculty In-charge Head of the Department

Submitted for Practical Examination Conducted on___________

Examiner–1 Examiner – 2

# INDEX

**WEEK-1 EXPERIMENT:**Creation, altering and dropping of tables and inserting rows into a table (use constraints while creating tables) examples using SELECT command.

**AIM:** Aim of the experiment is to CREATE, ALTER and DROP the tables and INSERT rows into a table.

**CREATE SYNTAX:**

CREATE TABLE TABLE_NAME (column1datatype,column2 datatype,column3 datatype,....);

**CREATE EXAMPLE:**

CREATE TABLE STUDENTS(pin int not null,lastname varchar(25),firstname varchar(25), phone_no int not null, email varchar(25) not null);

**DROP SYNTAX:**

DROP TABLE TABLE_NAME;

**DROP EXAMPLE:**

DROP TABLE STUDENTS;

CREATE TABLE STUDENTS(pin int not null, lastname varchar(25), firstname varchar(25), phone_no int not null, email varchar(25) not null);

**INSERT SYNTAX:**

INSERT INTO TABLE_NAME VALUES (column1 data,column2 data,.....columnN data);

**INSERT EXAMPLE:**

INSERT INTO STUDENTS VALUES (1001,'xyz','abc',12346789,'google@gmail.com');

INSERT INTO STUDENTS VALUES (1002,'yxz','bac',21346789,'ogogle@gmail.com');

**ALTER SYNTAX:**

ALTER TABLE TABLE_NAME ADD COLUMN_NAME DATATYPE;

**ALTER EXAMPLE:**

ALTER TABLE STUDENTS ADD GRADE VARCHAR(20);

**SELECT SYNTAX:**

SELECT * FROM table_name;

**SELECT EXAMPLE:**

SELECT * FROM STUDENTS;

Every time you can check the data in the database table using the query**SELECT * FROM TABLE_NAME;** and the structure/description of the table can be seen using **DESC TABLE_NAME;**

**WEEK-2 EXPERIMENTS:**QUERIES (ALONG WITH SUB QUERIES) USING ANY, ALL, IN, EXISTS, NOT-EXISTS, UNION, INTERSET, CONSTRAINTS.

**AIM:** Aim of the experiment is to illustrate ANY, ALL, IN, EXISTS, NOTEXISTS, UNION, INTERSET, Constraints along with Sub-Queries

**DESCRIPTION OF THE OPERATORS:**

**ALL:** The ALL operator is used to compare a value to all values in another value set.

**AND:** The AND operator allows the existence of multiple conditions in ansql statement's where clause.

**ANY:** The ANY operator is used to compare a value to any applicable value in the list as per the condition.

**EXISTS:**The EXISTS operator is used to search for the presence of a row in a specified table that meets a certain criterion.

**IN:** The IN operator is used to compare a value to a list of literal values that have been specified.

**UNION:** The UNION is a binary set operator in DBMS. It is used to combine the result set of two select queries.

**INTERSECT:**INTERSECT is a binary set operator in DBMS. The intersection operation between two selections returns only the common data sets or rows between them.

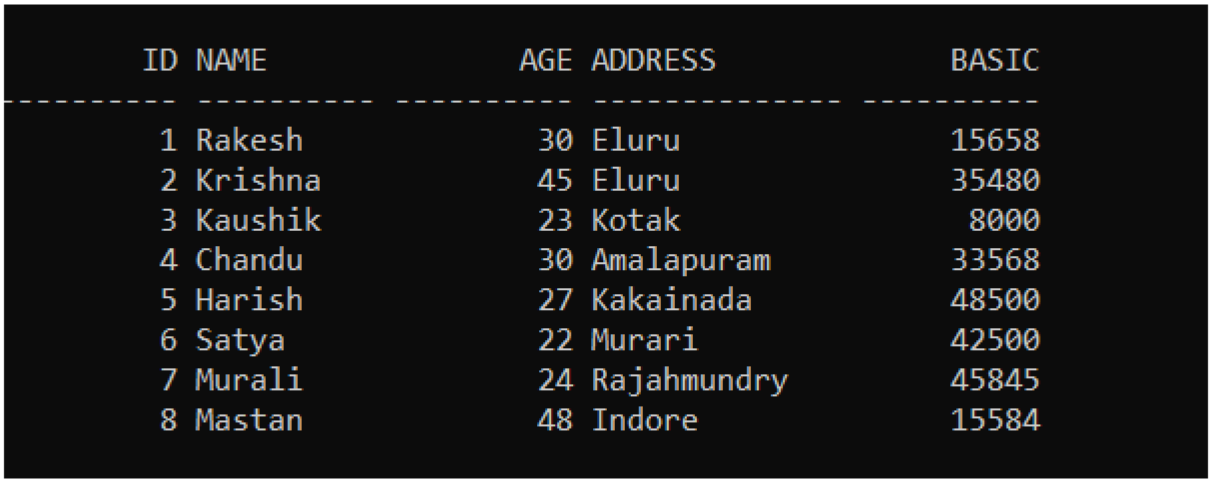
**COONSTARINTS:**CONSTRAINTS enforce limits to the data or type of data that can beinserted/updated/deleted from a table.

**EMPLOYEES TABLE:**

CREATE TABLEEMPLOYEES(idint,namevarchar(100),ageint,address varchar(100),basic int,primary key(id));

**Insertion of rows/data in to the created table is as follows:**

```
INSERT INTO EMPLOYEES VALUES (1,'Rakesh',30,'Eluru',15658.00);
INSERT INTO EMPLOYEES VALUES (2,'Krishna',45,'Eluru ',35480.00);
INSERT INTO EMPLOYEES VALUES (3,'Kaushik',23,'Kotak ',8000.00);
INSERT INTO EMPLOYEES VALUES (4,'Chandu',30,'Amalapuram ',33568.00);
INSERT INTO EMPLOYEES VALUES (5,'Harish',27,'Kakinada',48500.00);
INSERT INTO EMPLOYEES VALUES (6,'Satya',22,'Murari',42500.00);
INSERT INTO EMPLOYEES VALUES (7,'Murali',24,'Rajamahendravaram',45845.00);
INSERT INTO EMPLOYEES VALUES (8,'Mastan',48,'Indore',15584.00);
```



```
        ID NAME              AGE ADDRESS            BASIC
---------- ---------- ---------- ------------- ----------
         1 Rakesh             30 Eluru              15658
         2 Krishna            45 Eluru              35480
         3 Kaushik            23 Kotak               8000
         4 Chandu             30 Amalapuram         33568
         5 Harish             27 Kakainada          48500
         6 Satya              22 Murari             42500
         7 Murali             24 Rajahmundry        45845
         8 Mastan             48 Indore             15584
```

**CUSTOMERS TABLE:**
CREATE TABLE CUSTOMERS(ID INT,NAME VARCHAR(100),AGE INT,ADDRESS VARCHAR(100),SALARY INT);

**Insertion of rows/data in to the created table is as follows:**
INSERT INTO CUSTOMERS VALUES (1,'Ramesh',32,'Ahmedabad',2000.00);

INSERT INTO CUSTOMERS VALUES (2,'Khilan',25,'Delhi ',1500.00);

INSERT INTO CUSTOMERS VALUES (3,'kaushik',23,'Kota ',2000.00);

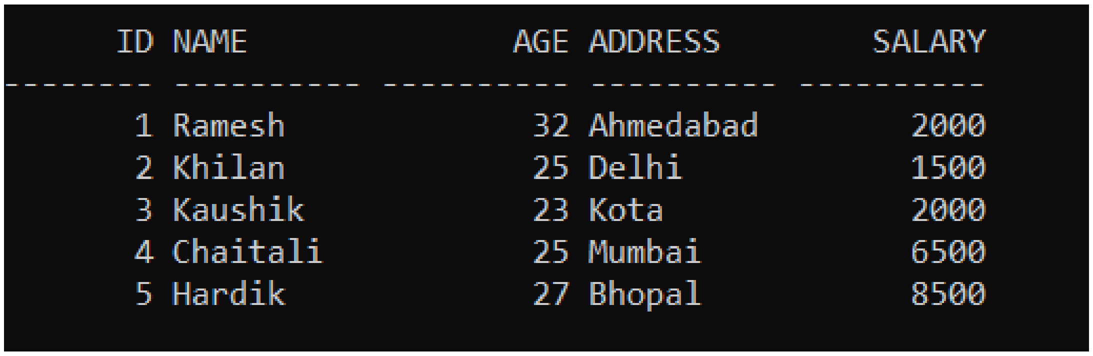
INSERT INTO CUSTOMERS VALUES (4,'Chaitali',25,'Mumbai ',6500.00);

INSERT INTO CUSTOMERS VALUES (5,'Hardik',27,'Bhopal',8500.00);

INSERT INTO CUSTOMERS VALUES (6,'Komal',22,'MP',4500.00);

INSERT INTO CUSTOMERS VALUES (7,'Muffy',24,'Indore',10000.00);

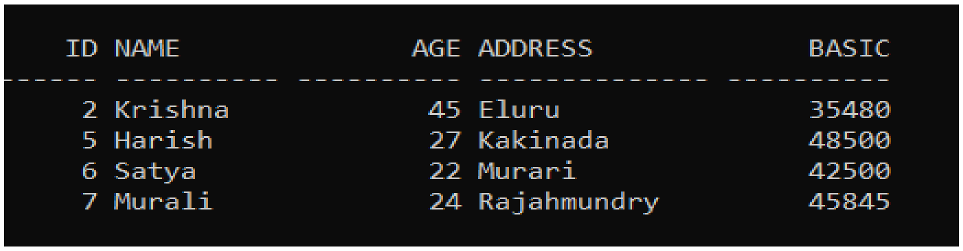
INSERT INTO CUSTOMERS VALUES (8,'Mastan',28,'Indore',20000.00);


```
      ID NAME              AGE ADDRESS        SALARY
-------- ---------- ---------- ---------- ----------
       1 Ramesh             32 Ahmedabad        2000
       2 Khilan             25 Delhi            1500
       3 Kaushik            23 Kota             2000
       4 Chaitali           25 Mumbai           6500
       5 Hardik             27 Bhopal           8500
```

## I. AT FIRST LET US ILLUSTRATE SAMPLE SUB-QUERIES

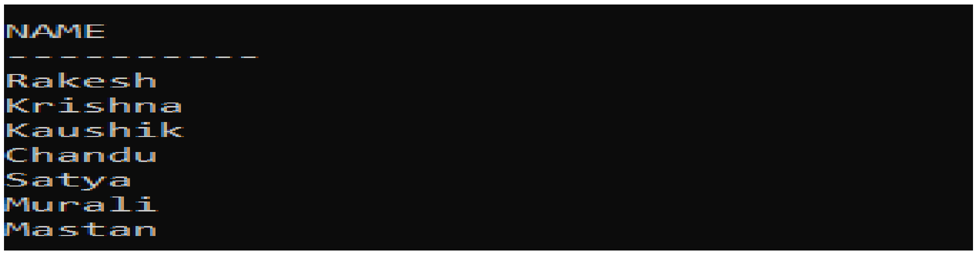
**Q:DISPLAY ALL THE DETAILS OF THE EMPLOYEES WHOSE BASIC IS GREATER THAN HARISH'S BASIC**

**A:**SELECT * FROM EMPLOYEES
WHERE BASIC>=(SELECT BASIC FROM EMPLOYEES WHERE NAME='HARISH');


```
    ID NAME              AGE ADDRESS             BASIC
------ ---------- ---------- -------------- ----------
     2 Krishna            45 Eluru               35480
     5 Harish             27 Kakinada            48500
     6 Satya              22 Murari              42500
     7 Murali             24 Rajahmundry         45845
```

**Q:DISPLAY THE NAMES OF THE EMPLOYEES WHO ARE NOT FROM HARISH'S WORK PLACE;**

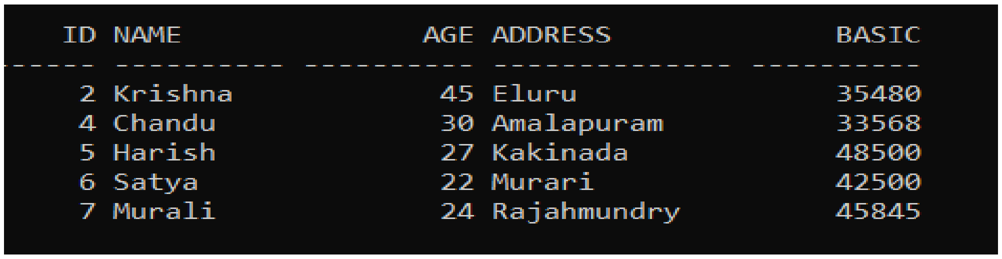
**A:**SELECT NAME FROM EMPLOYEES
WHERE ADDRESS! = (SELECT ADDRESS FROM EMPLOYEES WHERE NAME = 'Harish');



```
NAME
----------
Rakesh
Krishna
Kaushik
Chandu
Satya
Murali
Mastan
```

**Q:DISPLAY ALL THE DETAILS OF THE EMPLOYEES WHOSE SALARY IS GREATER THAN AVERAGE SALARY OF ALL EMPLOYEES**

**A:**SELECT * FROM EMPLOYEES
sWHERE BASIC> (SELECT AVG(BASIC) FROM EMPLOYEES);

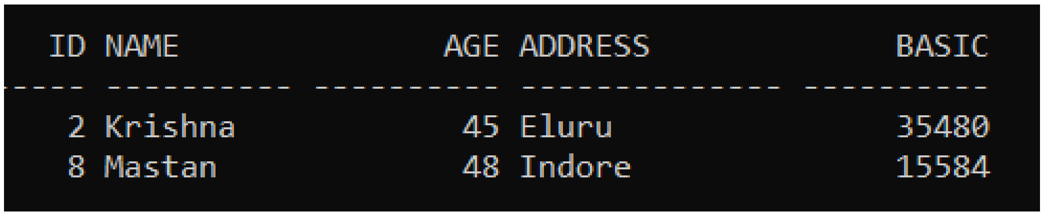


```
   ID NAME              AGE ADDRESS            BASIC
----- ---------- ---------- ------------- ----------
    2 Krishna            45 Eluru              35480
    4 Chandu             30 Amalapuram         33568
    5 Harish             27 Kakinada           48500
    6 Satya              22 Murari             42500
    7 Murali             24 Rajahmundry        45845
```

## II. ALL:

**Q:DISPLAY ALL THE DETAILS OF THE EMPLOYEES WHERE AGE OF AN EMPLOYEE IN EMPLOYEE TABLE IS GREATER THAN AGE OF ALL CUSTOMERS IN CUSTOMERS TABLE**

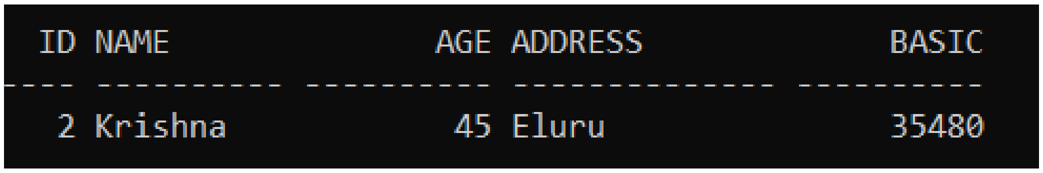
**A:**SELECT * FROM EMPLOYEES
WHERE AGE > ALL (SELECT AGE FROM CUSTOMERS);



```
  ID NAME              AGE ADDRESS            BASIC
---- ---------- ---------- ------------- ----------
   2 Krishna            45 Eluru              35480
   8 Mastan             48 Indore             15584
```

**Q:DISPLAY ALL THE DETAILS OF THE EMPLOYEES WHERE AGE OF AN EMPLOYEE IN EMPLOYEE TABLE IS GREATER THAN AGE OF ALL CUSTOMERS IN CUSTOMERS TABLE AND BASIC OF AN EMPLOYEE SHOULD BE GREATER THAN 50000;**

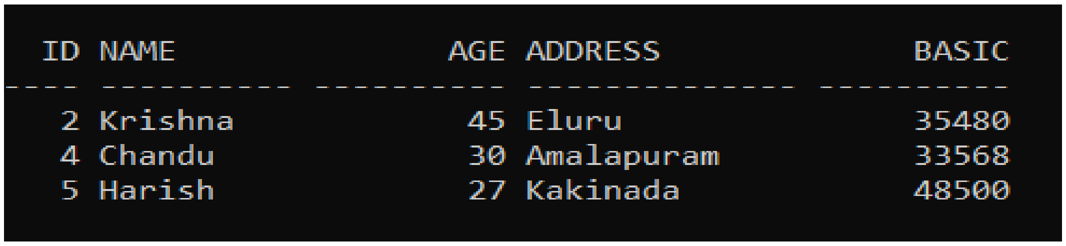
**A:**SELECT * FROM EMPLOYEES
WHERE BASIC > 50000 AND
AGE > ALL (SELECT AGE FROM CUSTOMERS);



```
  ID NAME              AGE ADDRESS            BASIC
---- ---------- ---------- ------------- ----------
   2 Krishna            45 Eluru              35480
```

**Q:DISPLAY THE NAMES OF THE EMPLOYEES WHOSE AGE IS GREATER THAN 25 AND BASIC IS GREATER THAN 25000;**

**A:**SELECT * FROM EMPLOYEES
WHEREBASIC >25000 AND AGE>25;

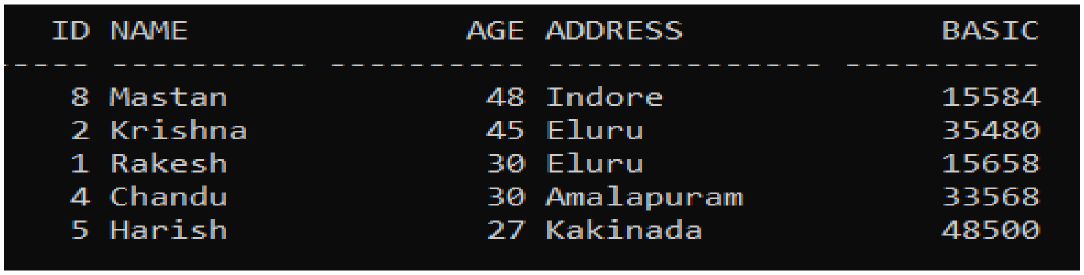


```
 ID NAME                  AGE ADDRESS               BASIC
---- ---------- ---------- -------------- ----------
  2 Krishna               45 Eluru                 35480
  4 Chandu                30 Amalapuram            33568
  5 Harish                27 Kakinada              48500
```

## III. ANY:

**Q:DISPLAY ALL THE DETAILS OF THE EMPLOYEES WHERE AGE OF AN EMPLOYEE IN EMPLOYEE TABLE IS GREATER THAN AGE OF ANY CUSTOMERS IN CUSTOMERS TABLE AND SALARY OF AN CUSTOMERS SHOULD BE GREATER THAN 5000;**

**A:**SELECT * FROM EMPLOYEES
WHERE AGE > ANY (SELECT AGE FROM CUSTOMERS WHERE SALARY>5000);



```
 ID NAME                  AGE ADDRESS               BASIC
---- ---------- ---------- -------------- ----------
  8 Mastan                48 Indore                15584
  2 Krishna               45 Eluru                 35480
  1 Rakesh                30 Eluru                 15658
  4 Chandu                30 Amalapuram            33568
  5 Harish                27 Kakinada              48500
```

## IV. EXISTS:

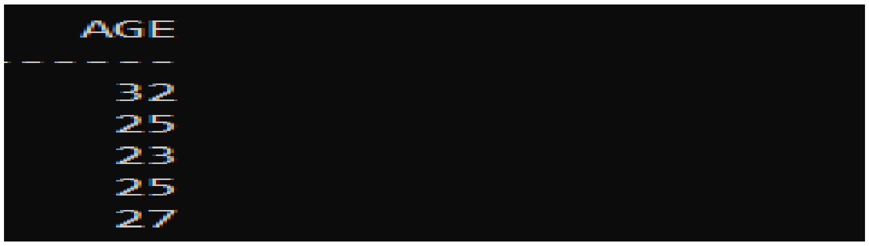
**Q:CHECK WHETHER THE AGE OF AN EMPLOYEE IN EMPLOYEE TABLE IS PRESENT IN THE AGE COLUMN OF CUSTOMERS TABLE OR NOT, IF YES THEN DISPLAY THAT EMPLOYEE AGE DETAILS.**

**A:**SELECT AGE FROM EMPLOYEES
WHERE NOT EXISTS (SELECT AGE FROM CUSTOMERS);

**NO ROWS SELECTED.**

**Q:CHECK WHETHER THE AGE OF AN EMPLOYEE IN EMPLOYEE TABLE IS PRESENT IN THE AGE COLUMN OF CUSTOMERS TABLE OR NOT, IF YES THEN DISPLAY THAT EMPLOYEE AGE DETAILS WHERE SALARY OF A CUSTOMER IS GREATER THAN 6500.**

**A:**SELECT AGE FROM CUSTOMERS
WHERE EXISTS (SELECT AGE FROM CUSTOMERS WHERE SALARY > 6500);

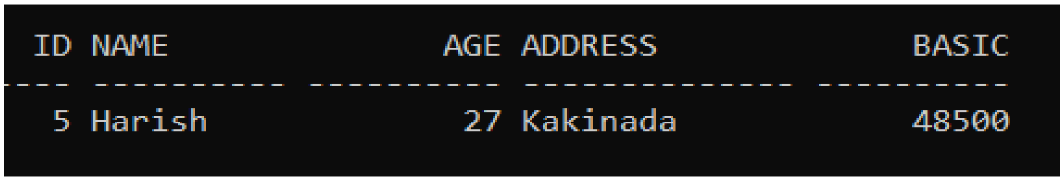


```
       AGE
----------
        32
        25
        23
        25
        27
```

## V. IN:

**Q:DISPLAY ALL THE DETAILS OF THE EMPLOYEES WHOSE AGE IS IN BETWEEN 25 AND 27**

**A:**SELECT * FROM EMPLOYEES WHERE AGE IN ( 25,27 );



```
 ID NAME                  AGE ADDRESS             BASIC
---- ---------- ---------- -------------- ----------
  5 Harish                 27 Kakinada            48500
```

**Note that NOT operator can be used as negation to the existing operator like EXISTS, IN, LIKE, NULL.**

## VI. UNION:

**SQL> select salary from  customers union select salary from employees;**

```
    SALARY
----------
      1500
      2000
      4500
      6500
      8000
      8500
     10000
     15584
     15658
     20000
     33568

    SALARY
----------
     35480
     42500
     45845
     48500
```

**15 rows selected.**

## VII. INTERSECTION

**SQL> select address from customers intersect select address from employees;**

```
ADDRESS
-------------------------
Indore
```

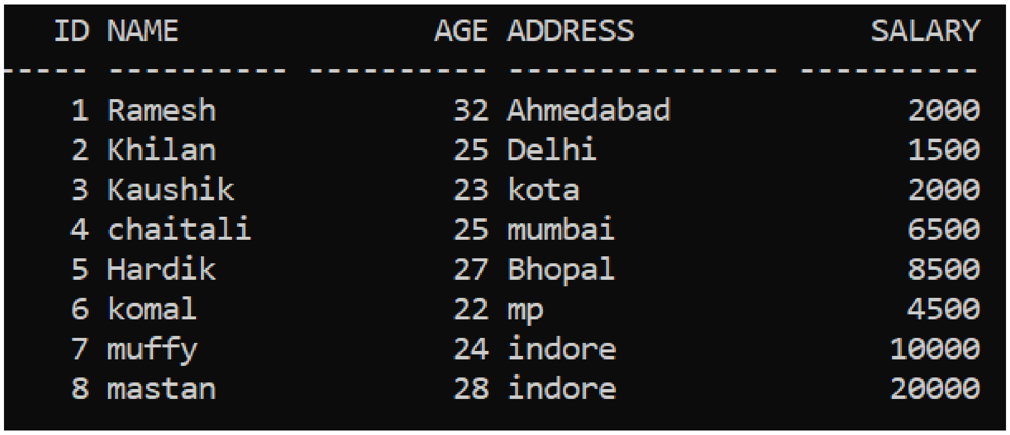
**WEEK-3 EXPERIMENTS:**QUERIES USING AGGREGATE FUNCTIONS (COUNT, SUM, AVG, MAX AND MIN), GROUP BY, HAVING AND CREATION AND DROPPING OF VIEWS.

**AIM**: Aim of the experiment is to illustrate the Aggregate functions -COUNT, SUM, AVG, MAX and MIN, GROUP BY, HAVING and Creation and dropping of Views.

**CUSTOMERS TABLE :**
CREATE TABLE CUSTOMER_S(ID INT,NAME VARCHAR(100),AGE INT,ADDRESS VARCHAR(100),SALARY INT);

INSERT INTO CUSTOMER_S VALUES (1,'Ramesh',32,'Ahmedabad',2000.00);
INSERT INTO CUSTOMER_S VALUES (2,'Khilan',25,'Delhi ',1500.00);
INSERT INTO CUSTOMER_S VALUES (3,'kaushik',23,'Kota ',2000.00);
INSERT INTO CUSTOMER_S VALUES (4,'Chaitali',25,'Mumbai ',6500.00);
INSERT INTO CUSTOMER_S VALUES (5,'Hardik',27,'Bhopal',8500.00);
INSERT INTO CUSTOMER_S VALUES (6,'Komal',22,'MP',4500.00);
INSERT INTO CUSTOMER_S VALUES (7,'Muffy',24,'Indore',10000.00);
INSERT INTO CUSTOMER_S VALUES (8,'Mastan',28,'Indore',20000.00);

Every time you can check the data in the database table using the query SELECT * FROM TABLE_NAME; i.e., SELECT * FROM CUSTOMER_S VALUES

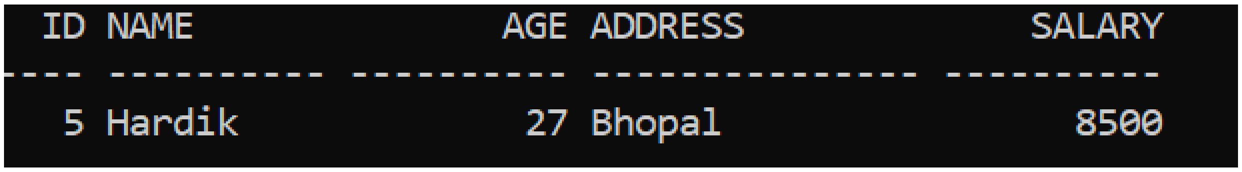

```
   ID NAME                 AGE ADDRESS             SALARY
----- ---------- ---------- --------------- ----------
    1 Ramesh                32 Ahmedabad             2000
    2 Khilan                25 Delhi                 1500
    3 Kaushik               23 kota                  2000
    4 chaitali              25 mumbai                6500
    5 Hardik                27 Bhopal                8500
    6 komal                 22 mp                    4500
    7 muffy                 24 indore               10000
    8 mastan                28 indore               20000
```

**Q: DISPLAY ALL THE DETAILS OF THE PERSON WHO IS IS FROM Bhopal.**
**A**: SELECT * FROM CUSTOMER_S WHERE ADDRESS='BHOPAL';

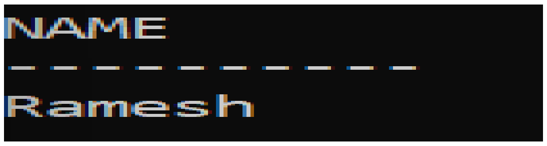
**NO ROWS SELECTED.**

**[NOTE: THAT SQL IS NOT CASE-SENSITIVE, BUT THE DATA IS CASE-SENSITIVE FOR THAT REASON THE ABOVE QUERY WONT WORK.]**

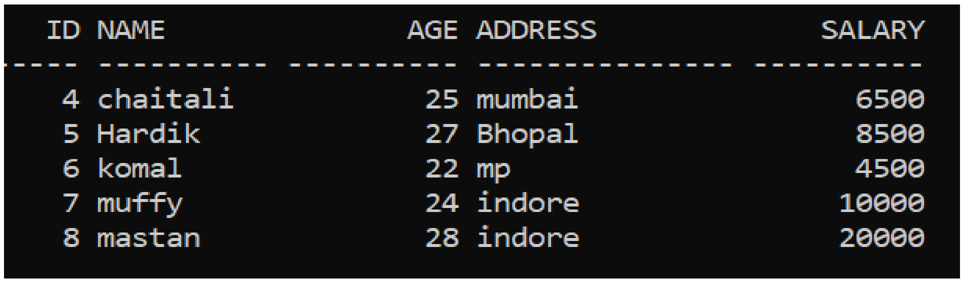
**A**: SELECT * FROM CUSTOMER_S WHERE ADDRESS='Bhopal';


```
  ID NAME                 AGE ADDRESS             SALARY
---- ---------- ---------- --------------- ----------
   5 Hardik                27 Bhopal                8500
```

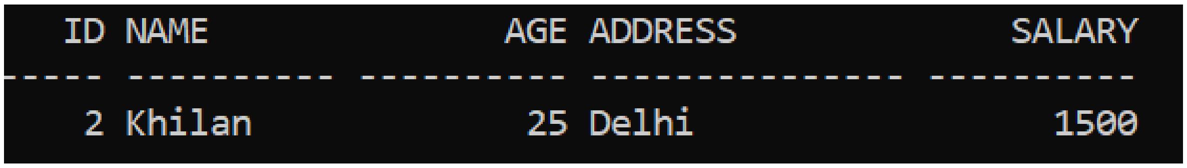
**Q: DISPLAY NAMES OF THE CUSTOMER_S WHO ARE FROM AHMEDABAD.**
**A:** SELECT NAME FROM CUSTOMER_S WHERE ADDRESS ='Ahmedabad';


```
NAME
-----------
Ramesh
```

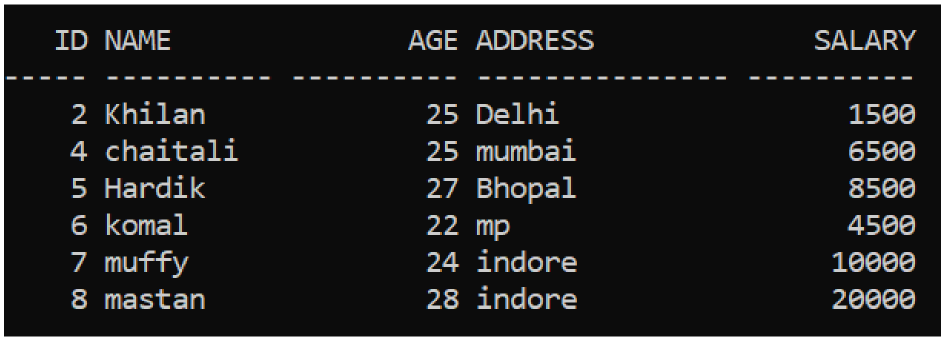
**Q: DISPLAY ALL THE DETAILS OF THE CUSTOMERS WHOSE SALARY IS GREATER THAN 2000**
**A**: SELECT * FROM CUSTOMER_S WHERE SALARY>2000;


```
  ID NAME              AGE ADDRESS            SALARY
---- ---------- ---------- -------------- ----------
   4 chaitali           25 mumbai               6500
   5 Hardik             27 Bhopal               8500
   6 komal              22 mp                   4500
   7 muffy              24 indore              10000
   8 mastan             28 indore              20000
```

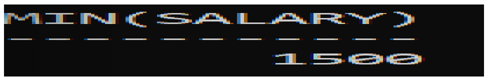
**Q: DISPLAY ALL THE DETAILS OF THE CUSTOMER_S WHOSE SALARY IS LESS THAN 2000**
**A**: SELECT * FROM CUSTOMER_S WHERE SALARY<2000;


```
   ID NAME              AGE ADDRESS            SALARY
----- ---------- ---------- -------------- ----------
    2 Khilan             25 Delhi                1500
```

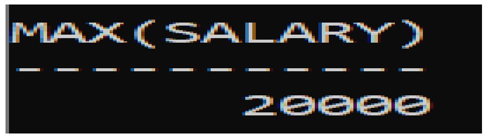
**Q: DISPLAY ALL THE DETAILS OF THE CUSTOMER_S WHOSE SALARY IS NOT EQUAL TO 2000**
**A**: SELECT * FROM CUSTOMER_S WHERE SALARY!=2000;


```
   ID NAME              AGE ADDRESS            SALARY
----- ---------- ---------- -------------- ----------
    2 Khilan             25 Delhi                1500
    4 chaitali           25 mumbai               6500
    5 Hardik             27 Bhopal               8500
    6 komal              22 mp                   4500
    7 muffy              24 indore              10000
    8 mastan             28 indore              20000
```

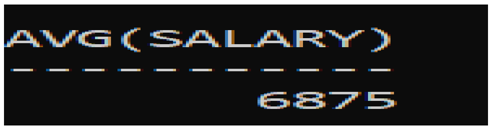
**Q: DISPLAY MINIMUM SALARY FROM CUSTOMER_S TABLE**
**A**: SELECT MIN(SALARY) FROM CUSTOMER_S;


```
MIN(SALARY)
-----------
       1500
```

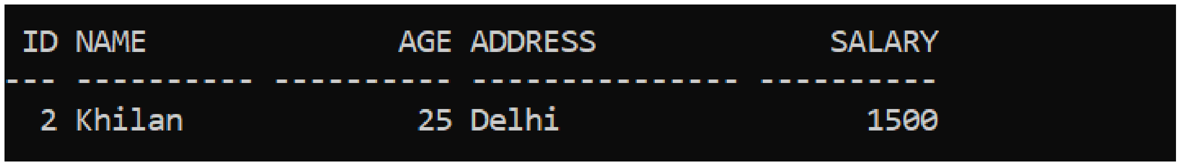
**Q: DISPLAY MAXIMUM SALARY FROM CUSTOMER_S TABLE**
**A:** SELECT MAX(SALARY) FROM CUSTOMER_S;


```
MAX(SALARY)
-----------
      20000
```

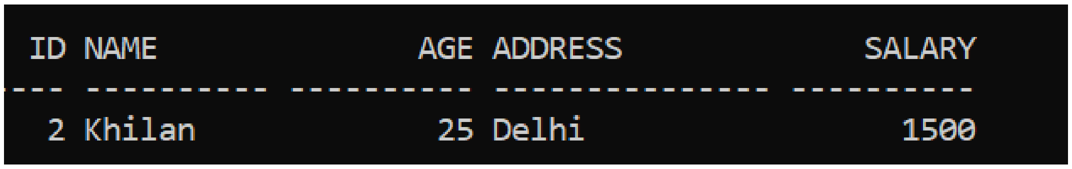
**Q: DISPLAY AVERAGE SALARY FROM CUSTOMER_S TABLE**
**A:** SELECT AVG(SALARY) FROM CUSTOMER_S;


```
AVG(SALARY)
-----------
       6875
```

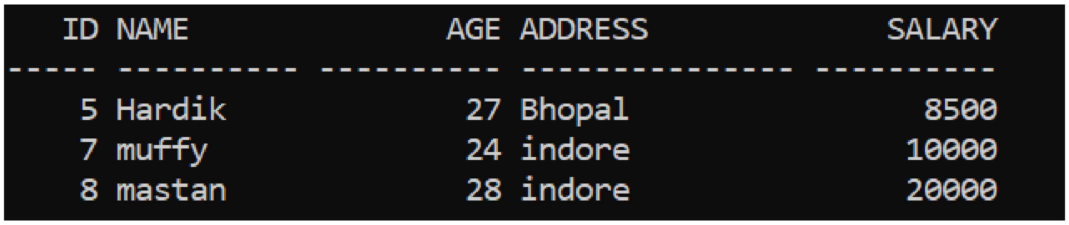
**Q: DISPLAY ALL THE DETAILS OF THE CUSTOMER_S WHOSE SALARY IS MINIMUM.**
**A:** SELECT * FROM CUSTOMER_S WHERE SALARY = (SELECT MIN(SALARY) FROM CUSTOMERS);


```
 ID NAME               AGE ADDRESS              SALARY
--- ---------- ---------- --------------- ----------
  2 Khilan              25 Delhi                  1500
```

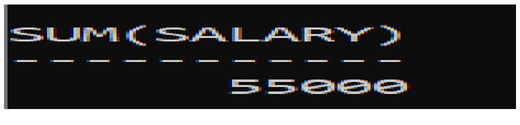
**A:** SELECT * FROM CUSTOMERS WHERE SALARY IN (SELECT MIN(SALARY) FROM CUSTOMER_S);


```
 ID NAME               AGE ADDRESS              SALARY
--- ---------- ---------- --------------- ----------
  2 Khilan              25 Delhi                  1500
```

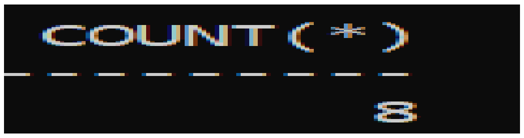
**Q: DISPLAY ALL THE DETAILS OF THE CUSTOMER_S WHOSE SALARY IS GREATER THAN AVERAGE SALARY AMONG ALL.**
**A:** SELECT * FROM CUSTOMER_S WHERE SALARY > (SELECT AVG(SALARY) FROM CUSTOMER_S);


```
   ID NAME               AGE ADDRESS              SALARY
----- ---------- ---------- --------------- ----------
    5 Hardik              27 Bhopal                 8500
    7 muffy               24 indore                10000
    8 mastan              28 indore                20000
```

**Q: DISPLAY THE TOTAL SALARY FROM THE CUSTOMER_S TABLE**
**A:** SELECT SUM(SALARY) FROM CUSTOMER_S;

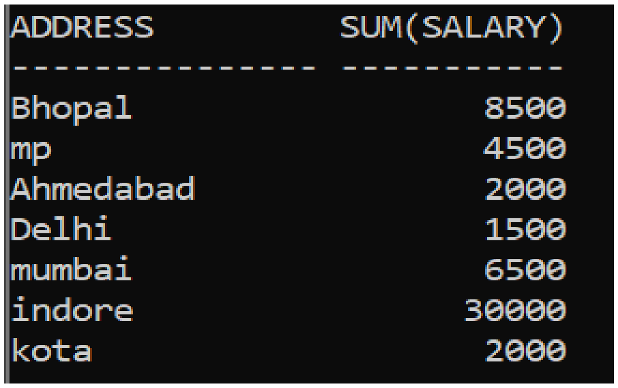

```
SUM(SALARY)
-----------
      55000
```

**Q: DISPLAY THE NUMBER OF ROWS IN THE CUSTOMER_S TABLE**
**A:** SELECT COUNT(*) FROM CUSTOMER_S;



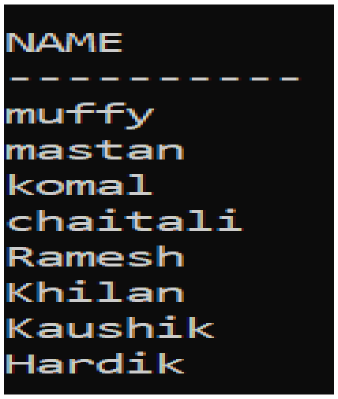
```
  COUNT(*)
----------
         8
```

**Q: DISPLAY THE TOTAL SALARY OF THE CUSTOMER_S ADDRESS WISE**
**A:** SELECT ADDRESS,SUM(SALARY) FROM CUSTOMER_S GROUP BY ADDRESS;



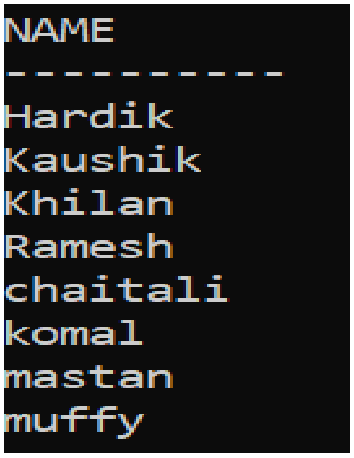
```
ADDRESS         SUM(SALARY)
--------------- -----------
Bhopal                 8500
mp                     4500
Ahmedabad              2000
Delhi                  1500
mumbai                 6500
indore                30000
kota                   2000
```

**Q: DISPLAY THE NAMES OF THE CUSTOMER_S IN DESCENDING ORDER**
**A:** SELECT NAME FROM CUSTOMER_S ORDER BY NAME DESC;



```
NAME
----------
muffy
mastan
komal
chaitali
Ramesh
Khilan
Kaushik
Hardik
```

**A:** SELECT NAME FROM CUSTOMER_S ORDER BY NAME;



```
NAME
----------
Hardik
Kaushik
Khilan
Ramesh
chaitali
komal
mastan
muffy
```

## CREATION AND DROPPING OF VIEWS.

**Creation of view Syntax :**

CREATE VIEW view_name AS
SELECT *column1*, *column2*,…
FROM *table_name*
WHERE *condition*;

**Q :CREATE A VIEW THAT SHOWS ALL CUSTOMERS WHOSE SALARY IS ABOVE 5000**

**A :** CREATE VIEW VIEW1 AS

```
     SELECT ID,NAME,SALARY
      FROM CUSTOMER_S
      WHERE salary>5000;
```

SELECT * FROM VIEW1;

**Dropping of view Syntax :**

DROP VIEW *view_name*;

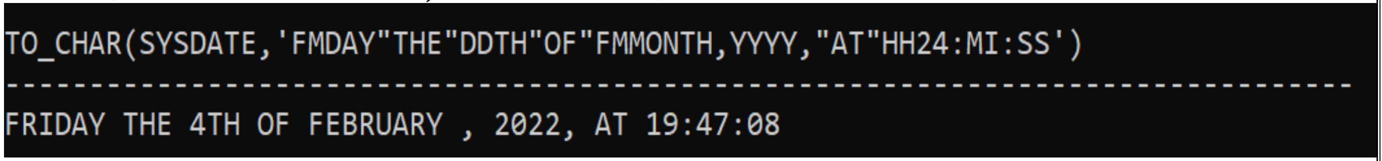
**Q :DROPS THE VIEW1 VIEW OF CUSTOMER_S TABLE**

**A :** DROP VIEW VIEW;

**WEEK-4 EXPERIMENTS:** QUERIES USING CONVERSION FUNCTIONS (TO_CHAR, TO_NUMBER AND TO_DATE), STRING FUNCTIONS (CONCATENATION, LPAD, RPAD, LTRIM, RTRIM, LOWER, UPPER, INITCAP, LENGTH, SUBSTR AND INSTR), DATE FUNCTIONS (SYSDATE, NEXT_DAY, ADD_MONTHS, LAST_DAY, MONTHS_BETWEEN, LEAST, GREATEST, TRUNC, ROUND, TO_CHAR, TO_DATE)

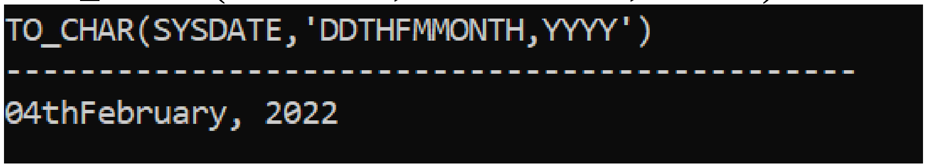
**AIM:**AIMOF THE EXPERIMENT IS TO ILLUSTRATE CHARACTER FUNCTIONS, STRING FUNCTIONS AND DATE FUNCTIONS.

**Q:DISPLAY SYSTEM DATE IN THE FOLLOWING FORMAT FRIDAY THE 20TH OF JULY 2018 AT 11:00:00**
**A:** SELECT TO_CHAR(SYSDATE,'FMDAY "THE" DDTH "OF" FMMONTH, YYYY, "AT" HH24:MI:SS')FROM DUAL;



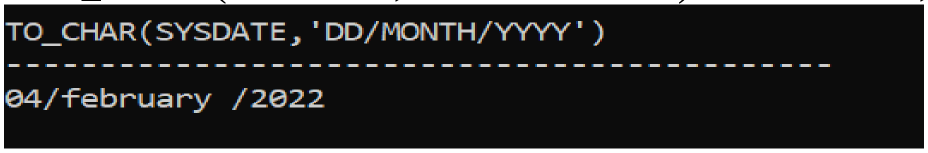
```
TO_CHAR(SYSDATE,'FMDAY"THE"DDTH"OF"FMMONTH,YYYY,"AT"HH24:MI:SS')
--------------------------------------------------------------------------------
FRIDAY THE 4TH OF FEBRUARY , 2022, AT 19:47:08
```

**Q: DISPLAY SYSTEM DATE IN THE FORMAT 20$^{TH}$JULY 2018**
**A:** SELECT TO_CHAR (SYSDATE,'ddthFMMonth, YYYY')FROM DUAL;



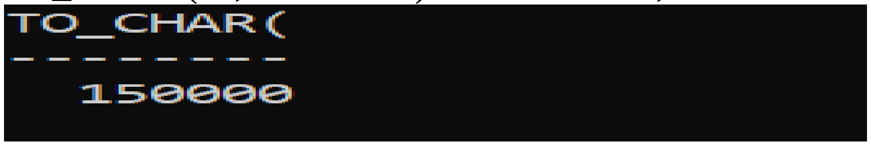
```
TO_CHAR(SYSDATE,'DDTHFMMONTH,YYYY')
--------------------------------------------
04thFebruary, 2022
```

**Q:DISPLAY SYSTEM IN THE FORMAT 20/07/2018**
**A:** SELECT TO_CHAR (SYSDATE,'dd/month/YYYY') FROM DUAL;



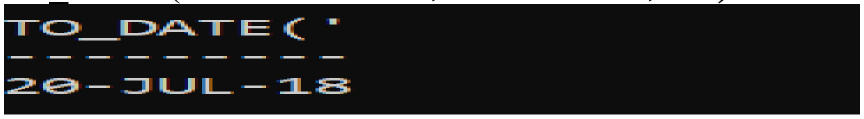
```
TO_CHAR(SYSDATE,'DD/MONTH/YYYY')
-------------------------------------------
04/february /2022
```

**Q: CONVERT 15 TO 15000 WITHOUT USING ANY ARTHIMATIC OPERATORS**
**A:** SELECT TO_CHAR (15,'999V9999') FROM DUAL;



```
TO_CHAR(
---------
   150000
```

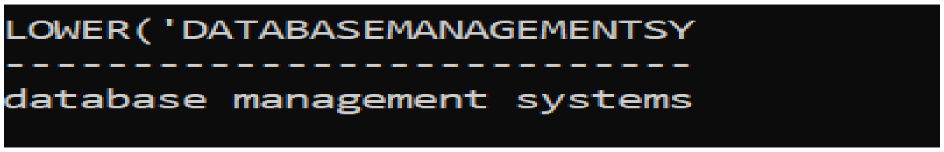
**Q: DISPLAY NUMERIC FORM OF THE DATE WHEN CHARACTER DATE AS AN INPUT**
**A:** SELECT TO_DATE(‘JULY  20 2018','MONTH DD,YY') FROM DUAL;



```
TO_DATE('
---------
20-JUL-18
```

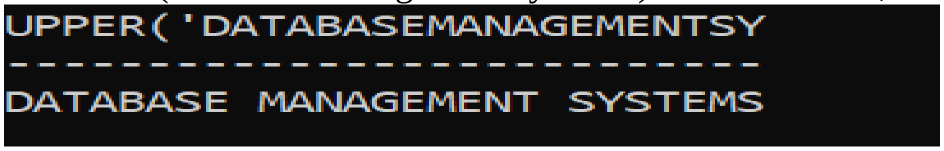
**Q: WRITE A QUERY TO CONVERT UPPER CASE LETTERS TO LOWER CASE.**
**EX: DBMS TO DBMS.**

**A:** SELECT LOWER('DATABASE MANAGEMENT SYSTEMS') FROM DUAL;



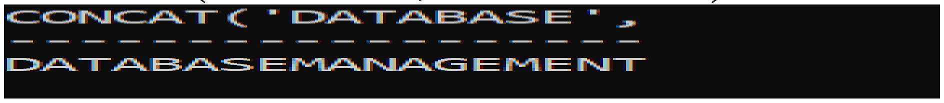
```
LOWER('DATABASEMANAGEMENTSY
---------------------------
database management systems
```

**Q: WRITE A QUERY TO CONVERT LOWER CASE LETTERS TO UPPER CASE.**
**A:** SELECT UPPER('database management systems') FROM DUAL;



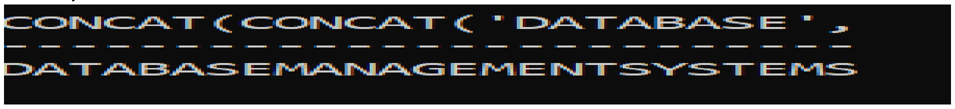
```
UPPER('DATABASEMANAGEMENTSY
---------------------------
DATABASE MANAGEMENT SYSTEMS
```

**Q: WRITE A QUERY TO CONCATENATE TWO STRINGS**
**A:** SELECT CONCAT('DATABASE','MANAGEMENT') FROM DUAL;



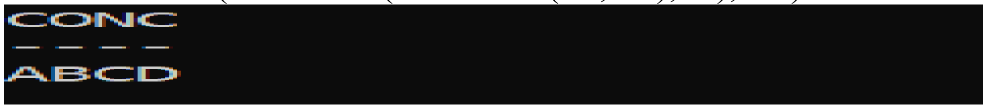
```
CONCAT('DATABASE',
------------------
DATABASEMANAGEMENT
```

**Q: WRITE A QUERY TO CONCATENATE THREE STRINGS**
**A:**SELECTCONCAT(CONCAT('DATABASE','MANAGEMENT'),'SYSTEMS') FROM DUAL;



```
CONCAT(CONCAT('DATABASE',
-------------------------
DATABASEMANAGEMENTSYSTEMS
```

**Q: WRITE A QUERY TO CONCATENATE ‘N’ STRINGS**
**A:** SELECT CONCAT(CONCAT(CONCAT('A', 'B'),'C'),’D’)FROM DUAL;

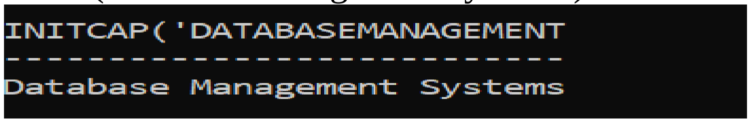


```
CONC
----
ABCD
```

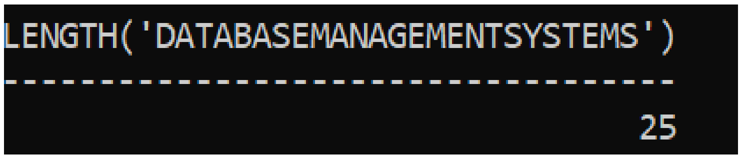
**Q: IDENTIFY THE NO. OF CONCAT COMMANDS USED TO CONCATENATE ‘N’ STRINGS.**
**A:**FROM THE ABOVE SELECT COMMANDS CONCATENATION OF TWO STRINGS NEED 1 CONCAT COMMAND, CONCATENATION OF THREE STRINGS NEED 2 CONCAT COMMANDS, CONCATENATION OF FOUR STRINGS NEED 3 CONCAT COMMANDS, LIKE WISE CONCATENATION OF ‘N’ STRINGS NEEDS ‘N-1’ CONCAT COMMANDS.

**Q: WRITE A QUERY TO CONVERT STARTING LETTER OF THE WORD IN A SENTENCE TO UPPER CASE**

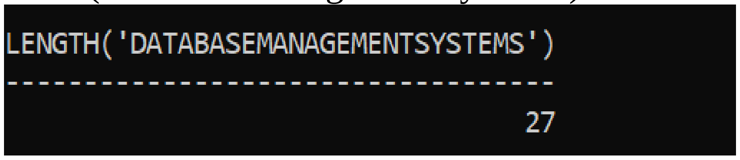
**A:** SELECT INITCAP('database management systems') FROM DUAL;


```
INITCAP('DATABASEMANAGEMENT
---------------------------
Database Management Systems
```

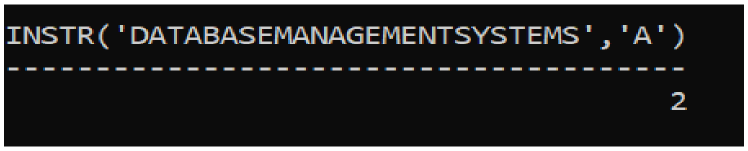
**Q: WRITE A SQL QUERY TO FIND THE LENGTH OF THE GIVEN STRING AND IDENTIFY WHETHER THE SPACES ARE BEING COUNTED OR NOT.**
**A:** SELECT LENGTH('databasemanagementsystems') FROM DUAL;


```
LENGTH('DATABASEMANAGEMENTSYSTEMS')
-----------------------------------
                                 25
```

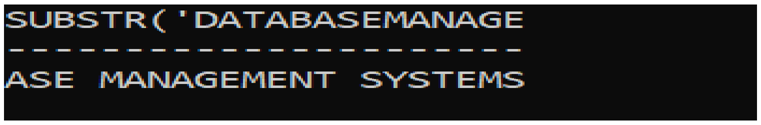
**A:** SELECT LENGTH('database management systems') FROM DUAL;


```
LENGTH('DATABASEMANAGEMENTSYSTEMS')
-----------------------------------
                                 27
```

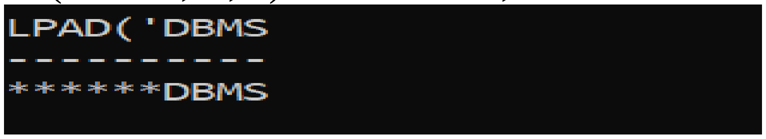
**Q: WRITE A SQL QUERY TO ILLUSTRATE INSTR**
**A:** SELECT INSTR('DATABASE MANAGEMENT SYSTEMS','A') FROM DUAL;


```
INSTR('DATABASEMANAGEMENTSYSTEMS','A')
--------------------------------------
                                     2
```

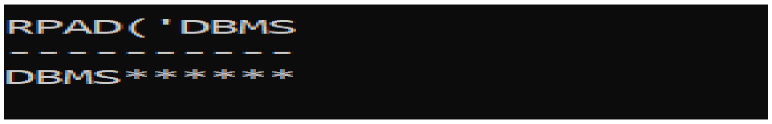
**Q: WRITE A SQL QUERY TO ILLUSTRATE SUBSTR**
**A:** SELECT SUBSTR('DATABASE MANAGEMENT SYSTEMS','6') FROM DUAL;


```
SUBSTR('DATABASEMANAGE
----------------------
ASE MANAGEMENT SYSTEMS
```

**Q: WRITE A SQL QUERY TO CONVERT DBMS AS **********DBMS USING LPAD**
**A:** SELECT LPAD('DBMS',10,'*')FROM DUAL;

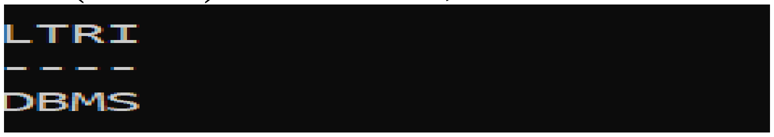

```
LPAD('DBMS
----------
******DBMS
```

**Q: WRITE A SQL QUERY TO CONVERT DBMS AS DBMS********** USING RPAD**
**A:** SELECT RPAD('DBMS',10,'*')FROM DUAL;



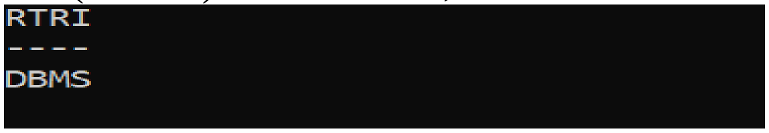
```
RPAD('DBMS
----------
DBMS******
```

**Q: WRITE A SQL QUERY TO REMOVE NULL CHARACTERS ON LEFT SIDE OF THE GIVEN STRING**
**A:** SELECT LTRIM('DBMS') FROM DUAL;



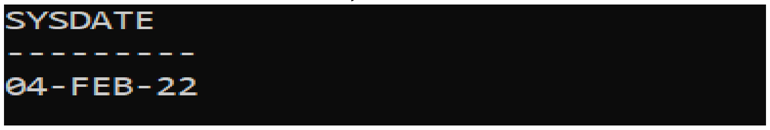
```
LTRI
----
DBMS
```

**Q: WRITE A SQL QUERY TO REMOVE NULL CHARACTERS ON RIGHT SIDE OF THE GIVEN STRING**
**A:** SELECT RTRIM('DBMS') FROM DUAL;



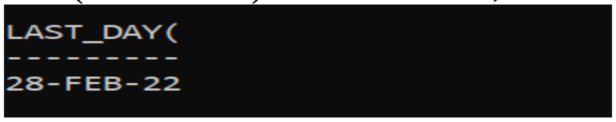
```
RTRI
----
DBMS
```

**Q: WRITE A SQL QUERIES TO ILLUSTRATE SYSDATE.**
**A:** SELECT SYSDATE FROM DUAL;



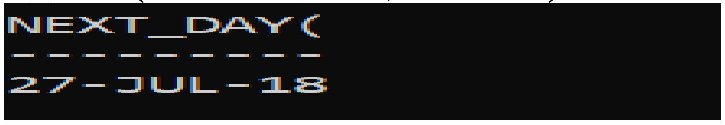
```
SYSDATE
---------
04-FEB-22
```

**Q: WRITE A SQL QUERIES TO DISPLAY LAST_DAY OF THE SYSDATE.**
**A:** SELECT LAST_DAY(SYSDATE) FROM DUAL;



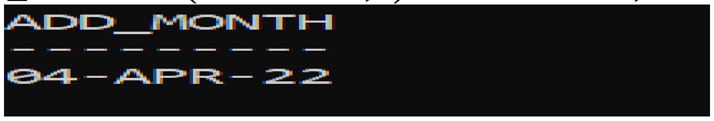
```
LAST_DAY(
---------
28-FEB-22
```

**Q: WRITE A SQL QUERIES TO DISPLAY NEXT_DAY OF THE SYSDATE.**
**A:** SELECT NEXT_DAY('20-JULY-2018','FRIDAY') FROM DUAL;



```
NEXT_DAY(
---------
27-JUL-18
```

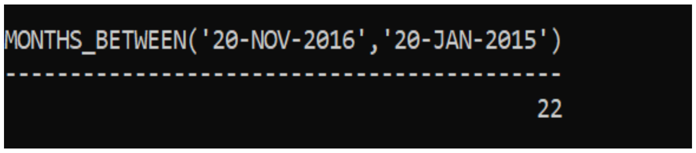
**Q: WRITE A SQL QUERIES TO DISPLAY DATE AFTER TWO MONTHS OF THE SYSDATE.**
**A:** SELECT ADD_MONTHS(SYSDATE,2) FROM DUAL;



```
ADD_MONTH
---------
04-APR-22
```

**Q: WRITEa A SQL QUERIES TO DISPLAY MONTHS BETWEEN TWO GIVEN.**

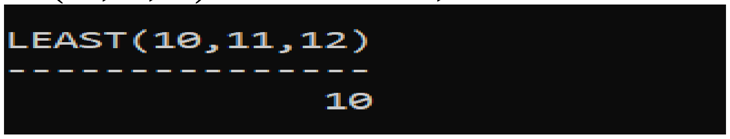
**A:**SELECT MONTHS_BETWEEN('20-NOV-2016','20-JAN-2015') FROM DUAL;



```
MONTHS_BETWEEN('20-NOV-2016','20-JAN-2015')
-------------------------------------------
                                         22
```

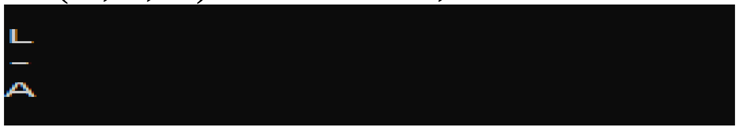
**Q: WRITE A SQL QUERY TO ILLUSTRATE LEAST, GREATEST, ROUND FUNCTIONS.**

**A:** SELECT LEAST(10,11,12) FROM DUAL;



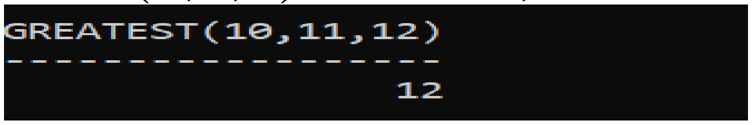
```
LEAST(10,11,12)
---------------
             10
```

**A:** SELECT LEAST('S','F','A') FROM DUAL;



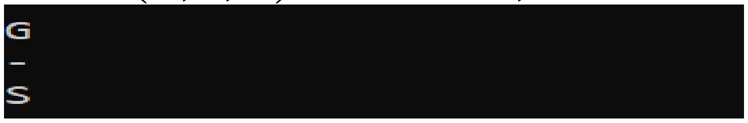
```
L
-
A
```

**A:** SELECT GREATEST(10,11,12) FROM DUAL;



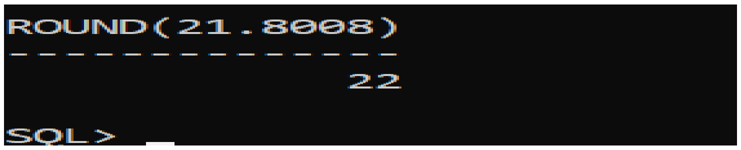
```
GREATEST(10,11,12)
------------------
                12
```

**A:** SELECT GREATEST('S','F','A') FROM DUAL;



```
G
-
S
```

**A:** SELECT ROUND(21.8008) FROM DUAL;



```
ROUND(21.8008)
--------------
            22

SQL> _
```

**WEEK-5 EXPERIMENTS:** CREATION OF SIMPLE PL/SQL PROGRAM WHICH INCLUDES DECLARATION SECTION, EXECUTABLE SECTION AND EXCEPTION – HANDLING SECTION (EX. STUDENT MARKS CAN BE SELECTED FROM THE TABLE AND PRINTED FOR THOSE WHO SECURED FIRST CLASS AND AN EXCEPTION CAN BE RAISED IF NO RECORDS WERE FOUND)

```
SQL> select * from student;
NAME                 PIN             ID     MARKS
-------------------- ---------- ---------- ----------
Sampath          20551A4260        60         8
Shariff           20551A4231        31         8
Aditya            20551A4202         2         8
Rohith            20551A4209         9         9
Nithin            20551A4216        16         8
```

```
SQL> declare
2  top varchar2(25);
3  begin
4  select name into top from student where marks=10;
5  dbms_output.put_line('The NAMES ARE : '||top);
6  exception
7   whenno_data_found then dbms_output.put_line('NO ONE SCORED FIRST CLASS');
8  end;
9  /
NO ONE SCORED FIRST CLASS
```

**PL/SQL procedure successfully completed.**

**WEEK-6 EXPERIMENTS:**INSERT DATA INTO STUDENT TABLE AND USE COMMIT, ROLLBACK AND SAVEPOINT IN PL/SQL BLOCK.

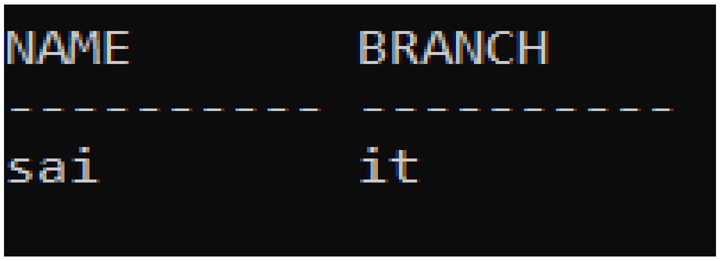
Create table stu(name varchar(10),branch varchar(10));
Insert into stu values('sai','it');
SAVEPOINT h;
**Savepoint created**

**SQL>** set serveroutput on;

**PROGRAM:**

```
SQL> begin
savepoint g;
insert into stu values('mahi','cse');
exception
when dup_val_on_index then
rollback to g;
commit;
end;
/
SQL> Rollback to h;
Rollback completed

SQL> SELECT * FROM stu;
```

**OUTPUT:**



```
NAME       BRANCH
---------- ----------
sai        it
```

**WEEK-7 EXPERIMENTS:** DEVELOP A PROGRAM THAT INCLUDES THE FEATURES NESTED IF, CASE AND CASE EXPRESSION. THE PROGRAM CAN BE EXTENDED USING THE NULLIF AND COALESCE FUNCTIONS.

**Syntax:**

```
DECLARE
        <declare section>
BEGIN
        <executable section>
EXCEPTION
<exception handling>
END;
```

**NESTED IF:**

```
SQL> declare --NESTED IF(2nd internal - 1st Qn)
2  a number:=&a;
3  b number:=&b;
4  begin
5  if a!=b then
6  if a>b then
7  dbms_output.put_line('A is the greatest');
8  else
9  dbms_output.put_line('B is the greatest');
10  end if;
11  end if;
12  end;
13  /
Enter value for a: 10
old   2: a number:=&a;
new   2: a number:=10;
Enter value for b: 20
old   3: b number:=&b;
new   3: b number:=20;
B is the greatest

PL/SQL procedure successfully completed.
```

**CASE EXPRESSION:**

```
SQL> declare --CASE EXPRESSION(2nd internal - 1st Qn)
2  grade varchar2(1):='&grade';
3  begin
4  case grade
5  when 'A' then dbms_output.put_line('Excellent');
6  when 'B' then dbms_output.put_line('GOOD');
7  when 'C' then dbms_output.put_line('Average');
8  when 'F' then dbms_output.put_line('Fail');
9  else
```

```
10  dbms_output.put_line('Invalid Grade Entered');
11  end case;
12  end;
13  /
Enter value for grade: A
old   2: grade varchar2(1):='&grade';
new   2: grade varchar2(1):='A';
Excellent

PL/SQL procedure successfully completed.
```

**CASE EXPRESSION:**

```
SQL>declare  --SEARCHED CASE EXPRESSION(2nd internal - 1st Qn)
2  grade varchar2(1):='&grade';
3  begin
4  case
5  when grade='A' then dbms_output.put_line('Excellent');
6  when grade='B' then dbms_output.put_line('GOOD');
7  when grade='C' then dbms_output.put_line('Average');
8  when grade='F' then dbms_output.put_line('Fail');
9  else
10  dbms_output.put_line('Invalid Grade Entered');
11  end case;
12  end;
13  /
Enter value for grade: A
old   2: grade varchar2(1):='&grade';
new   2: grade varchar2(1):='A';
Excellent

PL/SQL procedure successfully completed.
```

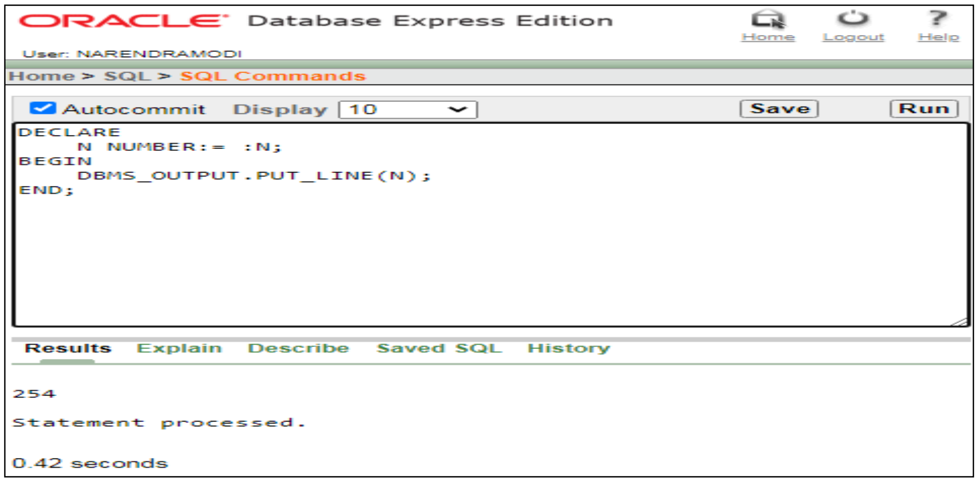
**NULLIF AND COALESCE FUNCTION:**

```
SQL> declare --NULLIF AND COALESCE FUNCTION(2nd internal - 1st Qn)
2  a number;
3  b number:=&b;
4  begin
5  ifnullif(b,a) is not null then
6  dbms_output.put_line(coalesce(a,b));
7  end if;
8  end;
9  /
Enter value for b: 2
old   3: b number:=&b;
new   3: b number:=2;
2
PL/SQL procedure successfully completed.
```

**Q1: TO READ A NUMBER FORM KEY BOARD AND DISPLAY THE SAME ON THE SCREEN**

```
DECLARE
   N NUMBER:= :N;
BEGIN
   DBMS_OUTPUT.PUT_LINE(N);
END;
/
```

**Output:**


ORACLE Database Express Edition
Home
Logout
Help
User: NARENDRAMODI
Home > SQL > SQL Commands
Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History

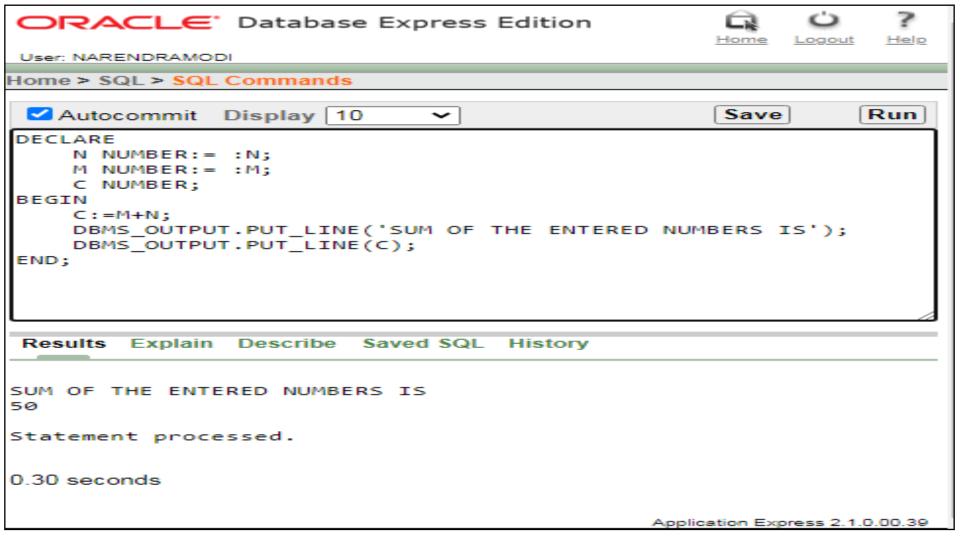
```
DECLARE
     N NUMBER:= :N;
BEGIN
     DBMS_OUTPUT.PUT_LINE(N);
END;
```

```
254

Statement processed.

0.42 seconds
```

**Q2: ADDITION OF TWO NUMBERS**

```
DECLARE
   N NUMBER:= :N;
   M NUMBER:= :M;
   C NUMBER;
BEGIN
C:=M+N;
   DBMS_OUTPUT.PUT_LINE('SUM OF THE ENTERED NUMBERS IS');
   DBMS_OUTPUT.PUT_LINE(C);
END;
/
```

**Output:**


ORACLE Database Express Edition
Home
Logout
Help
User: NARENDRAMODI
Home > SQL > SQL Commands
Autocommit Display 10
Save
Run
Results Explain Describe Saved SQL History
Application Express 2.1.0.00.39

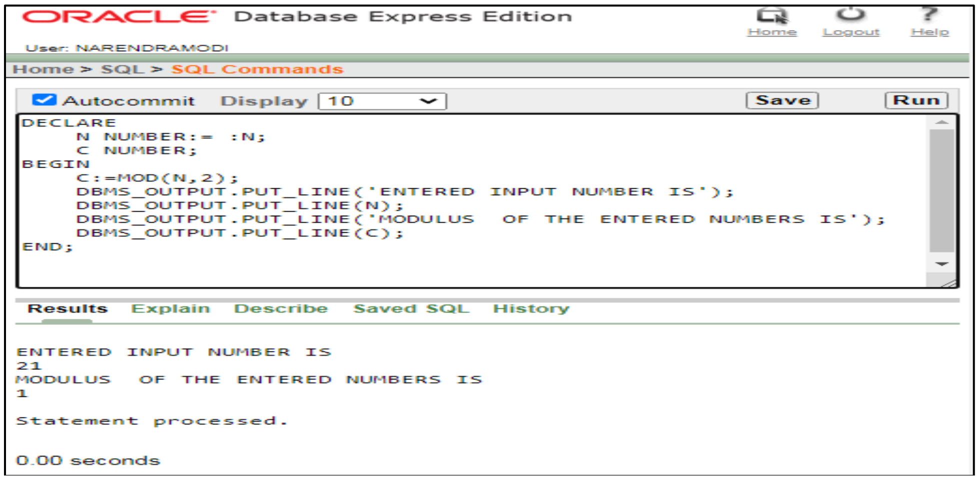
```
DECLARE
     N NUMBER:= :N;
     M NUMBER:= :M;
     C NUMBER;
BEGIN
     C:=M+N;
     DBMS_OUTPUT.PUT_LINE('SUM OF THE ENTERED NUMBERS IS');
     DBMS_OUTPUT.PUT_LINE(C);
END;
```

```
SUM OF THE ENTERED NUMBERS IS
50

Statement processed.

0.30 seconds
```

## Q3: MODULUS OPERATOR ILLUSTRATION

```
DECLARE
   N NUMBER:= :N;
   C NUMBER;
BEGIN
C:=MOD(N,2);
   DBMS_OUTPUT.PUT_LINE('ENTERED INPUT NUMBER IS');
   DBMS_OUTPUT.PUT_LINE(N);
   DBMS_OUTPUT.PUT_LINE('MODULUS  OF THE ENTERED NUMBERS IS');
   DBMS_OUTPUT.PUT_LINE(C);
END;
/
```

**Output:**


ORACLE Database Express Edition
Home Logout Help
User: NARENDRAMODI
Home > SQL > SQL Commands
Autocommit Display 10
Save Run
Results Explain Describe Saved SQL History

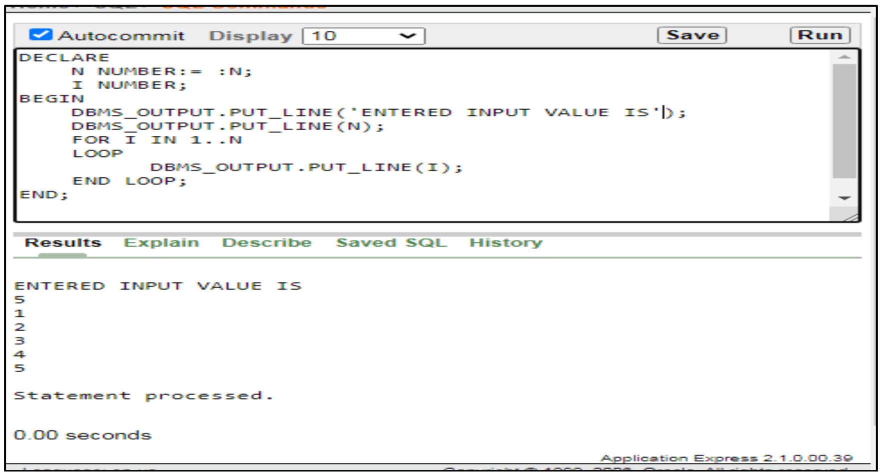
```
DECLARE
     N NUMBER:= :N;
     C NUMBER;
BEGIN
     C:=MOD(N,2);
     DBMS_OUTPUT.PUT_LINE('ENTERED INPUT NUMBER IS');
     DBMS_OUTPUT.PUT_LINE(N);
     DBMS_OUTPUT.PUT_LINE('MODULUS  OF THE ENTERED NUMBERS IS');
     DBMS_OUTPUT.PUT_LINE(C);
END;
```

```
ENTERED INPUT NUMBER IS
21
MODULUS  OF THE ENTERED NUMBERS IS
1

Statement processed.

0.00 seconds
```

## Q4: TO DISPLAY FIRST N NATURAL NUMBERS

```
DECLARE
   N NUMBER:= :N;
   I NUMBER;
BEGIN
   FOR I IN 1..N
   LOOP
   DBMS_OUTPUT.PUT_LINE(I);
   END LOOP;
END;   /
```

**Output:**


Autocommit Display 10
Save Run
Results Explain Describe Saved SQL History
Application Express 2.1.0.00.39

```
DECLARE
     N NUMBER:= :N;
     I NUMBER;
BEGIN
     DBMS_OUTPUT.PUT_LINE('ENTERED INPUT VALUE IS');
     DBMS_OUTPUT.PUT_LINE(N);
     FOR I IN 1..N
     LOOP
           DBMS_OUTPUT.PUT_LINE(I);
     END LOOP;
END;
```

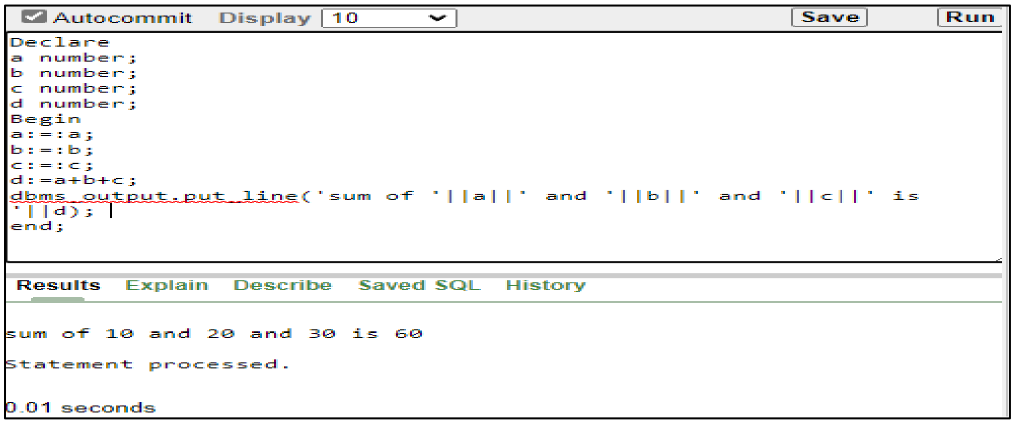
```
ENTERED INPUT VALUE IS
5
1
2
3
4
5

Statement processed.

0.00 seconds
```

**WEEK-8 EXPERIMENTS:** PROGRAM DEVELOPMENT USING WHILE LOOPS, NUMERIC FOR LOOPS, NESTED LOOPS USING ERROR HANDLING, BUILT –IN EXCEPTIONS, USE DEFINED EXCEPTIONS, RAISE- APPLICATION ERROR.

## 1) AIM: ADDITION AT RUN TIME

```
DECLARE
A NUMBER;
B NUMBER;
C NUMBER;
D NUMBER;
BEGIN
A:=:A;
B:=:B;
C:=:C;
D:= A+B+C;
DBMS_OUTPUT.PUT_LINE('SUM OF' || A || 'AND' || B || ' AND ' || C|| 'IS' || D);
END;
/
```

**Output:**



```
Declare
a number;
b number;
c number;
d number;
Begin
a:=:a;
b:=:b;
c:=:c;
d:=a+b+c;
dbms_output.put_line('sum of '||a||' and '||b||' and '||c||' is
'||d);
end;
```

```
sum of 10 and 20 and 30 is 60

Statement processed.

0.01 seconds
```

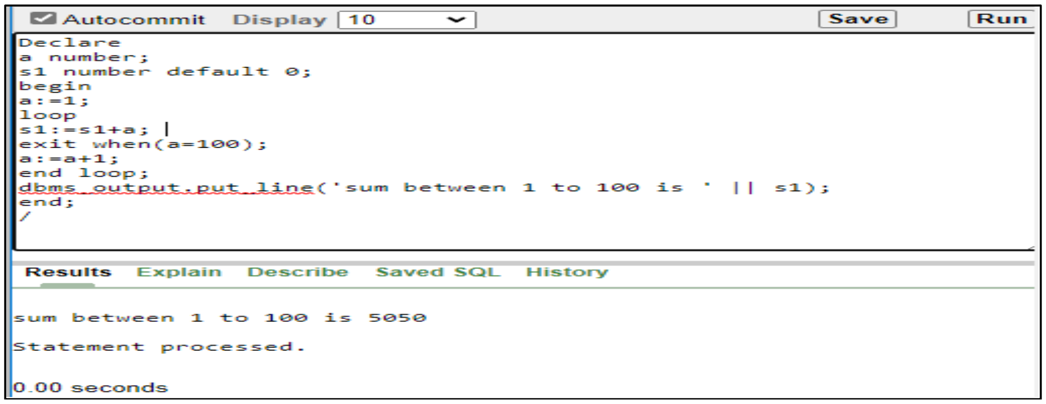
**PL/SQL procedure successfully completed.**

## 2) AIM: SIMPLE LOOP TO GET SUM OF 100 NUMBERS.

**PROGRAM:**

```
DECLARE
A NUMBER;
S1 NUMBER DEFAULT 0;
BEGIN
A:=1;
LOOP
S1:=S1+A;
EXIT WHEN(A=100);
A:=A+1;
END LOOP;
DBMS_OUTPUT.PUT_LINE('SUM BETWEEN 1 TO 100 IS' || S1);
END;
/
```

**Output:**



```
Declare
a number;
s1 number default 0;
begin
a:=1;
loop
s1:=s1+a;
exit when(a=100);
a:=a+1;
end loop;
dbms_output.put_line('sum between 1 to 100 is ' || s1);
end;
/
```

```
sum between 1 to 100 is 5050

Statement processed.

0.00 seconds
```

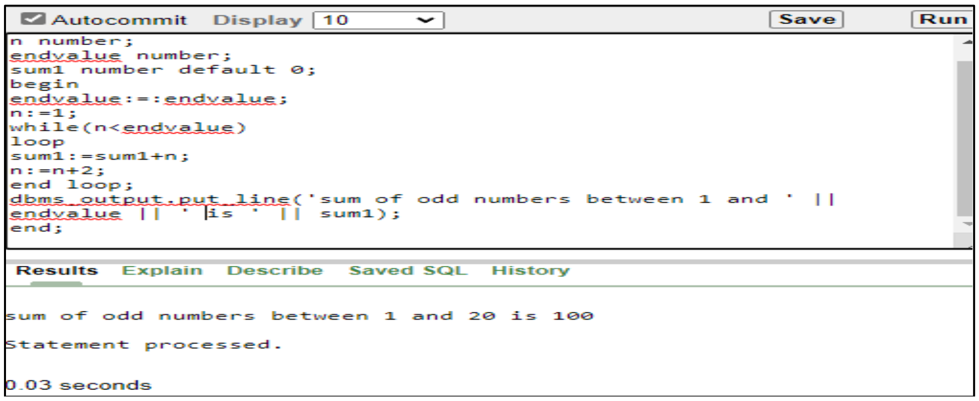
**PL/SQL procedure successfully completed.**

**3) AIM: While Loop for sum of 100 odd numbers.**

**PROGRAM:**

```
DECLARE
N NUMBER;
ENDVALUE NUMBER;
SUM1 NUMBER DEFAULT 0;
BEGIN
ENDVALUE:=:ENDVALUE;
N:=1;
WHILE(N<ENDVALUE)
LOOP
SUM1:=SUM1+N;
N:=N+2;
END LOOP;
DBMS_OUTPUT.PUT_LINE('SUM OF ODD NUMBERS BETWEEN 1 AND ' || ENDVALUE
|| 'IS' || SUM1);
END;
/
```

**Output:**



```
n number;
endvalue number;
sum1 number default 0;
begin
endvalue:=:endvalue;
n:=1;
while(n<endvalue)
loop
sum1:=sum1+n;
n:=n+2;
end loop;
dbms_output.put_line('sum of odd numbers between 1 and ' ||
endvalue || ' is ' || sum1);
end;
```

```
sum of odd numbers between 1 and 20 is 100

Statement processed.

0.03 seconds
```

**PL/SQL procedure successfully completed.**

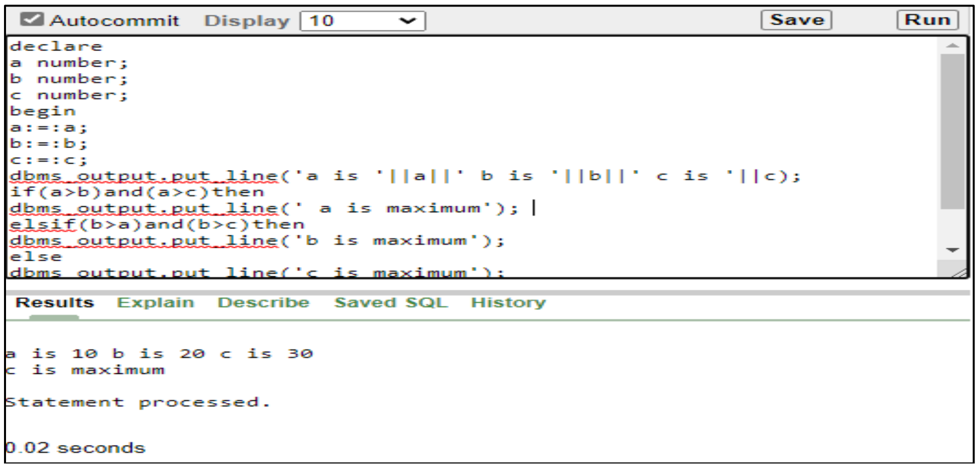
**4) AIM: if else for finding maximum of three numbers**

**Program:**

```
DECLARE
A NUMBER;
```

```
B NUMBER;
C NUMBER;
BEGIN
A:=:A;
B:=:B;
C:=:C;
DBMS_OUTPUT.PUT_LINE('A IS '||A||' B IS '||B||' C IS '||C);
IF(A>B)AND(A>C)THEN
DBMS_OUTPUT.PUT_LINE(' A IS MAXIMUM');
ELSIF(B>A)AND(B>C)THEN
DBMS_OUTPUT.PUT_LINE('B IS MAXIMUM');
ELSE
DBMS_OUTPUT.PUT_LINE('C IS MAXIMUM');
END IF;
END;
/
```

**Output:**



```
declare
a number;
b number;
c number;
begin
a:=:a;
b:=:b;
c:=:c;
dbms_output.put_line('a is '||a||' b is '||b||' c is '||c);
if(a>b)and(a>c)then
dbms_output.put_line(' a is maximum');
elsif(b>a)and(b>c)then
dbms_output.put_line('b is maximum');
else
dbms_output.put_line('c is maximum');
```

```
a is 10 b is 20 c is 30
c is maximum

Statement processed.

0.02 seconds
```

**PL/SQL procedure successfully completed.**

**5) AIM: select column from table employ by using memory variable**

**Program:**

```
DECLARE
MVSALARY NUMBER(10,2);
BEGIN
SELECT SALARY INTO MVSALARY
FROM
EMPLOY
WHERE ENAME='SAI';
DBMS_OUTPUT.PUT_LINE('THE SALARY OF EMPLOY IS' || TO_CHAR9MVSALARY));
END;
/
```

**Output:**

```
The salary of employ is 50000
```

**PL/SQL procedure successfully completed.**

**WEEK-9 EXPERIMENTS:** PROGRAMS DEVELOPMENT USING CREATION OF PROCEDURES, PASSING PARAMETERS IN AND OUT OF PROCEDURES.

**Syntax:**

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

->create table enquiry(enqno1 number,fname varchar2(30));
->insert into enquiry values(111,'sai');
->insert into enquiry values(112,'sindhu');
```

**Program:**

```
create procedure findname(enquiryno1 IN number,fname1 OUT varchar2)
is
fname2 varchar2(30);
begin
select fname into fname2
from enquiry
where enqno1=enquiryno1;
fname1:=fname2;
exception
when no_data-found then
raise_application_error(-20100,'The given number is not present');
end;
/
/*calling procedure*/
declare
enqno2 number(5);
fname2 varchar2(30);
begin
enqno2:=111;
findname(enqno2,fname2);
dbms_output.put_line(fname2);
end;
/
```

**Output:**sai

**WEEK-10 EXPERIMENTS:** PROGRAM DEVELOPMENT USING CREATION OF STORED FUNCTIONS, INVOKE FUNCTIONS IN SQL STATEMENTS AND WRITE COMPLEX FUNCTIONS.

```
->create table dept(deptno int,dname varchar(10));
->insert into dept values(1219,'sai');
```

**Program:**

```
create or replace function getname(dno number)
return varchar2 as
fname1 varchar2(30);
begin
select dname
into fname1
from dept
where deptno=dno;
return(fname1);
exception
whenno_data_found then
raise_application_error(-20100,'The dno is present');
end;
```

**/*calling function*/**

```
declare
fname2 varchar2(30);
deptno2 number(5);
begin
deptno:=1219;
fname2:=getname(dno);
dbms_output.put_line(fname2);
end; /
```

**output:**sai

**WEEK-11 EXPERIMENTS:** PROGRAM DEVELOPMENT USING CREATION OF PACKAGE SPECIFICATION, PACKAGE BODIES, PRIVATE OBJECTS, PACKAGE VARIABLES AND CURSORS AND CALLING STORED PACKAGES.

**(1)CREATE A TABLE DEPT1**

```
->create table dept1(dname varchar2(10),deptno number);
->insert into dept values('accounting',10);
->insert into dept values('hr',20);
```

**(2)CREATE A TABLE DEPT**

```
->create table dept(dno number,vt varchar2(10),dloc varcar2(20));
```

**(3)CREATING PACKAGE HEADER**

```
create or replace package test
is
procedure savedept
(dno in number,dloc in varchar);
end; /
```

**(4)CREATING PACKAGE BODY**

```
create or replace package body test
is
function getdno(dno in number)
return varchar
is
dnum varchar(20);
begin
select dname into dnum from dept
where deptno=dno;
return dnum;
end;
procedure savedept
(dno in number,dloc in varchar)
is
vt varchar(20)
begin
vt:=getno(dno);
insert into dept values(dno,vt,dloc);
exception
when dup_val_on_index then
raise_application_error(-2007,'duplicate');
end;
end; /
```

**(5)EXECUTING PROCEDURE**

```
exectest.savedept(10,'vijayawada');
```

**(6)DISPLAY THE TABLE**

```
->select * from dept;
```

**WEEK-12 EXPERIMENTS:**DEVELOP PROGRAMS USING FEATURES PARAMETERS IN A CURSOR, FOR UPDATE CURSOR, WHERE CURRENT OF CLAUSE AND CURSOR VARIABLES.

**Program:**

```
Create table employ( EID number(38),ENAME varchar2(20) SALARY
number(38));
SQL> insert into employ values(1,'sindhu',26000);
1 row created.
SQL> insert into employ values(3,'satya',26000);
1 row created.
```

**Cursor program:**

```
declare
emp_recvarchar(30);
cursor emp_cur is
select ename
from employ where salary > 25000;
Begin
Open emp_cur;
Loop
fetch emp_cur into emp_rec;
exit when emp_cur % notfound;
dbms_output.put_line(emp_rec);
end loop;
close emp_cur;
end;
/
```

**Output:**

```
sindhu
satya

PL/SQL procedure successfully completed.
```

**WEEK-13 EXPERIMENTS:** DEVELOP PROGRAMS USING BEFORE AND AFTER TRIGGERS, ROW AND STATEMENT TRIGGERS AND INSTEAD OF TRIGGERS.

## (1)CREATE A TRIGGER

```
create or replace trigger trg2
after insert or delete or update
on dept1
for each row
when(new.deptno>0)
begin
dbms_output.put_line('trigger fired');
end;
/
```

## (2)INSERT

```
->insert into deptvalues('sindhu',30);
*trigger fired*
*1 row created*
```

## (3)UPADTE

```
->udpate dept1 set deptno=19 where dname='sindhu';
*trigger fired*
*1 row updated*
```

## (4)DELETE

```
->delete from dept where deptno=30;
*trigger fired*
*1 row deleted*
```