KENDRIYA VIDYALAYA AMBARNATH, (SHIFT –I)

COMPUTER SCIENCE PRACTICAL RECORD (TERM-2)

CLASS : XII

2021-2022

ACKNOWLEDGEMENT:

I, ...OF CLASS XII –
SECTION ..... WOULD LIKE TO EXPRESS OUR SINCERE GRATITUDE TO OUR
COMPUTER SCIENCE TEACHER
MS.APARNA DHIRDE,
PGT (COMPUTER SCIENCE), FOR HER VITAL SUPPORT, GUIDANCE AND ENCOURAGEMENT –
WITHOUT WHICH

THIS PRACTICAL(TERM-2) WOULD NOT HAVE COME FORTH.

WE WOULD ALSO LIKE TO EXPRESS OUR GRATITUDE TO OUR SCHOOL KENDRIYA VIDYALAYA
AMBARNATH .

CERTIFICATE

This is to certify that, Practical on Computer Science (Term-2) is successfully completed by

…… of

Class: XII, Division: …….

Roll no. :……..

for the academic year

2021-2022.

Signature:

(Subject Teacher) Internal Examiner

External Examiner

Principal

Date: / / 2022

INDEX

XII CS – Practical Assignments for TERM 2

S.No	Description of Assignment	Sign
1	Write a Python program to implement a stack using list (PUSH & POP Operation on Stack).	
2	Write a python program using function PUSH(Arr), where Arr is a list of numbers. From this list push all numbers divisible by 5 into a stack implemented by using a list. Display the stack if it has at least one element, otherwise display appropriate error message.	
3	Write a python program using function POP(Arr), where Arr is a stack implemented by a list of numbers. The function returns the value deleted from the stack.	
4	Write a python program to check whether a string is a palindrome or not using stack.	
5	Create a table and insert data. Implement all SQL commands on the table	
6	Integrate MySQL with Python by importing the MySQL module and add records of student and display all the record.	
7	Integrate MySQL with Python by importing the MySQL module to search student using rollno, name, age, class and if present in table display the record, if not display appropriate method.	
8	Integrate SQL with Python by importing the MySQL module to search a student using rollno, delete the record.	
9	Integrate SQL with Python by importing the MySQL module to search a student using rollno, update the record.	
10	SQL Queries	
11	Project - BANK OF GRINGOTTS ONLINE PORTAL	

Assignment 10: **SQL QUERIES**

Queries Set 1 (Database Fetching records)

[1] Consider the following MOVIE table and write the SQL queries based on it.

Movie_ID	MovieName	Type	ReleaseDate	ProductionCost	BusinessCost
M001	The Kashmir Files	Action	2022/01/26	1245000	1300000
M002	Attack	Action	2022/01/28	1120000	1250000
M003	Looop Lapeta	Thriller	2022/02/01	250000	300000
M004	Badhai Do	Drama	2022/02/04	720000	68000
M005	Shabaash Mithu	Biography	2022/02/04	1000000	800000
M006	Gehraiyaan	Romance	2022/02/11	150000	120000

1. Display all information from the movie.
2. Display the type of movies.
3. Display movieid, movie name, total_eraning by showing the business done by the movies. Calculate the business done by movie using the sum of production cost and business cost.
4. Display movie id, movie name and production cost for all movies with production cost greater than 150000 and less than 1000000.
5. Display the movie of type action and romance.
6. Display the list of movies which are going to release in February, 2022.

Answers:

[1] select * from movie ;

Output:

```
mysql> select * from movie;
+----------+------------------+----------+-------------+---------------+--------------+
| Movie_id | moviename        | type     | releasedate | productioncost | businesscost |
+----------+------------------+----------+-------------+---------------+--------------+
| M001     | The Kashmir Files | Action   | 2022-01-26  |        1245000 |      1300000 |
| M002     | Attack           | Action   | 2022-01-28  |        1120000 |      1250000 |
| M003     | Loop Lapeta      | Thriller | 2022-02-01  |         250000 |       300000 |
| M004     | Bdhai Do         | Drama    | 2022-02-04  |         720000 |      4800000 |
| M005     | Shabaash Mothu   | Biography | 2022-02-04 |        1000000 |       800000 |
| M006     | Gehriyaan        | Romance  | 2022-02-11  |         150000 |       120000 |
+----------+------------------+----------+-------------+---------------+--------------+
6 rows in set (0.01 sec)
```

2. select distinct from a movie ;

```
mysql> select distinct type from movie;
+------------+
| type       |
+------------+
| Action     |
| Thriller   |
| Drama      |
| Biography  |
| Romance    |
+------------+
5 rows in set (0.00 sec)
```

3. select movieid, movie name, production cost + business cost “total earning” from movie ;

```
mysql> select movie_id, moviename, productioncost + businesscost "total earning" from movie;
+----------+------------------+--------------+
| movie_id | moviename        | total earning |
+----------+------------------+--------------+
| M001     | The Kashmir Files |       2545000 |
| M002     | Attack           |       2370000 |
| M003     | Loop Lapeta      |        550000 |
| M004     | Bdhai Do         |       5520000 |
| M005     | Shabaash Mothu   |       1800000 |
| M006     | Gehriyaan        |        270000 |
+----------+------------------+--------------+
6 rows in set (0.03 sec)
```

4. select movie_id, movie name, production cost from movie where product is >150000 and <1000000 ;

```
mysql> select movie_id,moviename, productioncost from movie where productioncost >150000 and productioncost<100000
+----------+------------+---------------+
| movie_id | moviename  | productioncost |
+----------+------------+---------------+
| M003     | Loop Lapeta |        250000 |
| M004     | Bdhai Do    |        720000 |
+----------+------------+---------------+
2 rows in set (0.03 sec)
```

5. select movie name from movie where type =’action’ or type=’romance’ ;

```
mysql>  select moviename from movie where type ='action' or type='romance'
+-------------------+
| moviename         |
+-------------------+
| The Kashmir Files |
| Attack            |
| Gehriyaan         |
+-------------------+
3 rows in set (0.00 sec)
```

6. select movie name from movie where month(release date)=2 ;

```
mysql> select moviename from movie where month(releasedate)=2;
+-----------------+
| moviename       |
+-----------------+
| Loop Lapeta     |
| Bdhai Do        |
| Shabaash Mothu  |
| Gehriyaan       |
+-----------------+
4 rows in set (0.00 sec)
```

Queries Set 2 (Based on Functions)

1. Write a query to display a cube of 5.
2. Write a query to display the number 563.854741 rounding off to the next hundred.
3. Write a query to display “put” from the word “Computer”.

4. Write a query to display today's date into DD.MM.YYYY format.
5. Write a query to display 'DIA' from the word "MEDIA".
6. Write a query to display movie name – type from the table movie.
7. Write a query to display the first four digits of production cost.
8. Write a query to display the last four digits of business cost.
9. Write a query to display weekdays of release dates.
10. Write a query to display the dayname on which movies are going to be released.

Answers:

[1] select pow(5,3) ;

```
mysql> select pow(5,3);
+----------+
| pow(5,3) |
+----------+
|      125 |
+----------+
1 row in set (0.01 sec)
```

[2] select round(563.854741,-2) ;

```
mysql> select round(563.854741,-2);
+---------------------+
| round(563.854741,-2) |
+---------------------+
|                 600 |
+---------------------+
1 row in set (0.00 sec)
```

[3] select mid("Computer",4,3) ;

```
mysql> select mid("Computer",4,3);
+----------------------+
| mid("Computer",4,3) |
+----------------------+
| put                  |
+----------------------+
1 row in set (0.00 sec)
```

[4] select concat(day(now()),concat('.',month(now()),concat('.',year(now())))) "Date" ;

```
mysql> select concat(day(now()),concat('.',month(now()),concat('.',year(now())))) "Date";
+----------+
| Date     |
+----------+
| 9.1.2022 |
+----------+
1 row in set (0.00 sec)
```

[5] select right("Media",3) ;

```
mysql> select right("Media",3);
+------------------+
| right("Media",3) |
+------------------+
| dia              |
+------------------+
1 row in set (0.00 sec)
```

[6] select concat(movie name,concat(' – ',type)) from movie ;

```
mysql> select concat(moviename,concat(' - ',type)) from movie;
+-------------------------------------+
| concat(moviename,concat(' - ',type)) |
+-------------------------------------+
| The Kashmir Files - Action          |
| Attack - Action                     |
| Loop Lapeta - Thriller              |
| Bdhai Do - Drama                    |
| Shabaash Mothu - Biography          |
| Gehriyaan - Romance                 |
+-------------------------------------+
6 rows in set (0.00 sec)
```

[7] select left (production cost,4) from movie ;

```
mysql> select left(productioncost,4) from movie;
+-------------------------+
| left(productioncost,4) |
+-------------------------+
| 1245                    |
| 1120                    |
| 2500                    |
| 7200                    |
| 1000                    |
| 1500                    |
+-------------------------+
6 rows in set (0.00 sec)
```

[8] select right (business cost,4) from movie ;

```
mysql> select right(businesscost,4) from movie;
+-------------------------+
| right(businesscost,4) |
+-------------------------+
| 0000                    |
| 0000                    |
| 0000                    |
| 0000                    |
| 0000                    |
| 0000                    |
+-------------------------+
6 rows in set (0.00 sec)
```

[9] select weekday (release date) from movie ;

```
mysql> select weekday(releasedate) from movie;
+-----------------------+
| weekday(releasedate) |
+-----------------------+
|                     2 |
|                     4 |
|                     1 |
|                     4 |
|                     4 |
|                     4 |
+-----------------------+
6 rows in set (0.00 sec)
```

[10] select dayname (release date) from movie ;

```
mysql> select dayname(releasedate) from movie;
+-----------------------+
| dayname(releasedate) |
+-----------------------+
| Wednesday             |
| Friday                |
| Tuesday               |
| Friday                |
| Friday                |
| Friday                |
+-----------------------+
6 rows in set (0.01 sec)
```

Queries Set 3 (DDL Commands)

Suppose your school management has decided to conduct cricket matches between students of Class XI and Class XII. Students of each class are asked to join any one of the four teams – Team Titan, Team Rockers, Team Magnet and Team Hurricane. During summer vacations, various matches will be conducted between these teams. Help your sports teacher to do the following:

1. Create a database “Sports”.
2. Create a table “TEAM” with following considerations:
    - It should have a column TeamID for storing an integer value between 1 to 9, which refers to unique identification of a team.
    - Each TeamID should have its associated name (TeamName), which should be a string of length not less than 10 characters.
    - Using table level constraint, make TeamID as the primary key.
    - Show the structure of the table TEAM using a SQL statement.
    - As per the preferences of the students four teams were formed as given below. Insert these four rows in TEAM table:
        - Row 1: (1, Tehlka)
        - Row 2: (2, Toofan)
        - Row 3: (3, Aandhi)
        - Row 3: (4, Shailab)
    - Show the contents of the table TEAM using a DML statement.
3. Now create another table MATCH_DETAILS and insert data as shown below. Choose appropriate data types and constraints for each attribute.

MatchID	MatchDate	FirstTeamID	SecondTeamID	FirstTeamScore	SecondTeamScore
M1	2021/12/20	1	2	107	93
M2	2021/12/21	3	4	156	158
M3	2021/12/22	1	3	86	81
M4	2021/12/23	2	4	65	67
M5	2021/12/24	1	4	52	88
M6	2021/12/25	2	3	97	68

Answers:

[1] create database sports

```
mysql> create database sports
    -> ;
Query OK, 1 row affected (0.01 sec)

mysql> use sports;
Database changed
mysql>
```

[2] Creating table with the given specification

```
create table team
```

```
-> (teamid int(1),
```

```
-> teamname varchar(10), primary key(teamid));
```

Showing the structure of table using SQL statement:

```
desc team;
```

```
mysql> desc team;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| teamid   | int         | NO   | PRI | NULL    |       |
| teamname | varchar(10) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

Inserting data:

```
mqsql> insert into team
```

```
    -> values(1,'Tehlka');
```

```
mysql> insert into team
    -> values(1,'Tehlka');
Query OK, 1 row affected (0.01 sec)

mysql> insert into team
    -> values(2,'Toofan');
Query OK, 1 row affected (0.01 sec)

mysql> insert into team
    -> values(3,'Aandhi');
Query OK, 1 row affected (0.03 sec)

mysql> insert into team
    -> values(4,'Shailab');
Query OK, 1 row affected (0.03 sec)
```

Show the content of table – team:

```
select * from team;
```

```
mysql> select * from team;
+--------+----------+
| teamid | teamname |
+--------+----------+
|      1 | Tehlka   |
|      2 | Toofan   |
|      3 | Aandhi   |
|      4 | Shailab  |
+--------+----------+
4 rows in set (0.00 sec)
```

Creating another table:

```
create table match_details
    -> (matchid varchar(2) primary key,
    -> matchdate date,
    -> firstteamid int(1) references team(teamid),
    -> secondteamid int(1) references team(teamid),
    -> firstteamscore int(3),
    -> secondteamscore int(3));
```

```
mysql> create table match_details
    -> (matchid varchar(2) primary key,
    -> matchdate date,
    -> firstteamid int(1) references team(teamid),
    -> secondteamid int(1) references team(teamid),
    -> firstteamscore int(3),
    -> secondteamscore int(3));
Query OK, 0 rows affected, 4 warnings (0.03 sec)
```

```
mysql> select * from match_details;
+---------+------------+-------------+--------------+----------------+-----------------+
| matchid | matchdate  | firstteamid | secondteamid | firstteamscore | secondteamscore |
+---------+------------+-------------+--------------+----------------+-----------------+
| M1      | 2021-12-20 |           1 |            2 |            107 |              93 |
| M2      | 2021-12-21 |           3 |            4 |            156 |             158 |
| M3      | 2021-12-22 |           1 |            3 |             86 |              81 |
| M4      | 2021-12-23 |           2 |            4 |             65 |              67 |
| M5      | 2021-12-24 |           1 |            4 |             52 |              88 |
| M6      | 2021-12-25 |           2 |            3 |             97 |              68 |
+---------+------------+-------------+--------------+----------------+-----------------+
6 rows in set (0.01 sec)
```

Queries set 4 (Based on Two Tables)

1. Display the matchid, teamid, team score who scored more than 70 in first inning along with team name.
2. Display matchid, team name and second team score between 100 to 160.
3. Display matchid, team names along with matchdates.
4. Display unique team names
5. Display matchid and match date played by Anadhi and Shailab.

Answers:

[1] select match_details.matchid, match_details.firstteamid, team.team name,match_details.firstteamscore from match_details, team where match_details.firstteamid=team.teamid and match_details.firstteamscore>70;

```
mysql> select * from match_details;
+---------+------------+-------------+-------------+---------------+----------------+
| matchid | matchdate  | firstteamid | secondteamid | firstteamscore | secondteamscore |
+---------+------------+-------------+-------------+---------------+----------------+
| M1      | 2021-12-20 |           1 |           2 |           107 |             93 |
| M2      | 2021-12-21 |           3 |           4 |           156 |            158 |
| M3      | 2021-12-22 |           1 |           3 |            86 |             81 |
| M4      | 2021-12-23 |           2 |           4 |            65 |             67 |
| M5      | 2021-12-24 |           1 |           4 |            52 |             88 |
| M6      | 2021-12-25 |           2 |           3 |            97 |             68 |
+---------+------------+-------------+-------------+---------------+----------------+
6 rows in set (0.01 sec)
```

[2] select matchid, team name, second team score from match_details, team were match_details.secondteamid=team.teamid and match_details.secondteamscore between 100 and 160;

```
mysql> select matchid, teamname, secondteamscore from match_details, team where match_details.secondteamid=team.teamid
 and match_details.secondteamscore between 100 and 160;
+---------+---------+----------------+
| matchid | teamname | secondteamscore |
+---------+---------+----------------+
| M2      | Shailab  |            158 |
+---------+---------+----------------+
1 row in set (0.00 sec)
```

[3] select matchid,team name,first teamid,secondteamid,match date from match_details, team were match_details.firstteamid=team.team id;

```
mysql> select matchid,teamname,firstteamid,secondteamid,matchdate from match_details, team where match_details.firstte
amid=team.teamid;
+---------+---------+-------------+-------------+------------+
| matchid | teamname | firstteamid | secondteamid | matchdate  |
+---------+---------+-------------+-------------+------------+
| M1      | Tehlka   |           1 |           2 | 2021-12-20 |
| M2      | Aandhi   |           3 |           4 | 2021-12-21 |
| M3      | Tehlka   |           1 |           3 | 2021-12-22 |
| M4      | Toofan   |           2 |           4 | 2021-12-23 |
| M5      | Tehlka   |           1 |           4 | 2021-12-24 |
| M6      | Toofan   |           2 |           3 | 2021-12-25 |
+---------+---------+-------------+-------------+------------+
6 rows in set (0.00 sec)
```

[4] select distinct(team name) from match_details, team were match_details.firstteamid=team.team id;

```
mysql> select distinct(teamname) from match_details, team where match_details.firstteamid=team.teamid;
+----------+
| teamname |
+----------+
| Tehlka   |
| Aandhi   |
| Shailab  |
| Toofan   |
+----------+
4 rows in set (0.03 sec)
```

[5] select matchid,matchdate from match_details, team were match_details.firstteamid=team.teamid and team.team name in ('Aandhi','Shailab');

```
mysql> select matchid,matchdate from match_details, team where match_details.firstteamid=team.teamid and team.teamname
 in ('Aandhi','Shailab');
+---------+------------+
| matchid | matchdate  |
+---------+------------+
| M2      | 2021-12-21 |
| M4      | 2021-12-23 |
+---------+------------+
2 rows in set (0.00 sec)
```

Queries Set 5 (Group by , Order By)

Consider the following table stock table to answer the queries:

item no	item	dcode	qty	unit price	stockdate
S005	Ballpen	102	100	10	2018/04/22
S003	Gel Pen	101	150	15	2018/03/18
S002	Pencil	102	125	5	2018/02/25
S006	Eraser	101	200	3	2018/01/12
S001	Sharpner	103	210	5	2018/06/11
S004	Compass	102	60	35	2018/05/10
S009	A4 Papers	102	160	5	2018/07/17

1. Display all the items in the ascending order of stockdate.
2. Display maximum price of items for each dealer individually as per dcode from stock.
3. Display all the items in descending orders of item names.
4. Display average price of items for each dealer individually as per doce from stock which average price is more than 5.
5. Display the sum of quantity for each dcode.

[1] select * from stock order by stockdale;

```
mysql> select * from stock order by stockdate;
+--------+-----------+-------+------+-----------+------------+
| itemno | item      | dcode | qty  | unitprice | stockdate  |
+--------+-----------+-------+------+-----------+------------+
| S006   | Eraser    |   101 |  200 |         3 | 2018-01-12 |
| S002   | Pencil    |   102 |  125 |         5 | 2018-02-25 |
| S003   | Gel Pen   |   101 |  150 |        15 | 2018-03-18 |
| S005   | BallPen   |   102 |  100 |        10 | 2018-04-22 |
| S004   | Compass   |   102 |   60 |        35 | 2018-05-10 |
| S009   | A4 Papers |   102 |  160 |         5 | 2018-07-17 |
| S001   | Sharpner  |   103 |  210 |         5 | 2018-11-06 |
+--------+-----------+-------+------+-----------+------------+
7 rows in set (0.00 sec)
```

[2] select dcode,max(unit price) from stock group by code;

```
mysql> select dcode,max(unitprice) from stock group by dcode;
+-------+---------------+
| dcode | max(unitprice) |
+-------+---------------+
|   103 |             5 |
|   102 |            35 |
|   101 |            15 |
+-------+---------------+
3 rows in set (0.01 sec)
```

[3] select * from stock order by item desc;

```
mysql> select * from stock order by item desc;
+--------+-----------+-------+------+-----------+------------+
| itemno | item      | dcode | qty  | unitprice | stockdate  |
+--------+-----------+-------+------+-----------+------------+
| S001   | Sharpner  |   103 |  210 |         5 | 2018-11-06 |
| S002   | Pencil    |   102 |  125 |         5 | 2018-02-25 |
| S003   | Gel Pen   |   101 |  150 |        15 | 2018-03-18 |
| S006   | Eraser    |   101 |  200 |         3 | 2018-01-12 |
| S004   | Compass   |   102 |   60 |        35 | 2018-05-10 |
| S005   | BallPen   |   102 |  100 |        10 | 2018-04-22 |
| S009   | A4 Papers |   102 |  160 |         5 | 2018-07-17 |
+--------+-----------+-------+------+-----------+------------+
7 rows in set (0.01 sec)
```

[4] select dcode,avg(unit price) from stock group by dcode having avg(unit price)>5;

```
mysql> select dcode,avg(unitprice) from stock group by dcode having avg(unitprice)>5;
+-------+---------------+
| dcode | avg(unitprice) |
+-------+---------------+
|   102 |       13.7500 |
|   101 |        9.0000 |
+-------+---------------+
2 rows in set (0.00 sec)
```

[5] select dcode,sum(qty) from stock group by dcode;

```
mysql> select dcode,sum(qty) from stock group by dcode;
+-------+----------+
| dcode | sum(qty) |
+-------+----------+
|   103 |      210 |
|   102 |      445 |
|   101 |      350 |
+-------+----------+
3 rows in set (0.03 sec)
```

Output is:

```
**********************************************************
Enter string to check : nitin
Yes, the string is a palindrome

**********************************************************

Enter string to check : palindrome
No, the string is not a palindrome

**********************************************************

Enter string to check : nursesrun
Yes, the string is a palindrome
```

Write a python program using function PUSH(Arr), where Arr is a list of numbers. From this list push all numbers divisible by 5 into a stack implemented by using a list. Display the stack if it has at least one element, otherwise display appropriate error message.

Ans:

```
def isEmpty(Arr):
    if len(Arr)==0:
        return True
    else:
        return False

def push(Arr,item):
    if item%5==0:
        Arr.append(item)
        top=len(Arr)-1

def show(Arr):
    if isEmpty(Arr):
        print('No item found')
    else:
        t=len(Arr)-1
        print('(TOP)',end='')
        while(t>=0):
            print(Arr[t],'<==',end='')
            t=t-1
        print()

Arr=[]
top=None
while True:
    print('****** STACK IMPLEMENTATION USING LIST ******')
    print('1: PUSH')
    print('2: Show')
    print('0: Exit')
    ch=int(input('Enter choice:'))
    if ch==1:
        val=int(input('Enter no to push:'))
        push(Arr,val)
    elif ch==2:
        show(Arr)
    elif ch==0:
        print('Bye')
        break
```

Output is:

```
****** STACK IMPLEMENTATION USING LIST ******
1: PUSH
2: Show
0: Exit
Enter choice:1
Enter no to push:23
****** STACK IMPLEMENTATION USING LIST ******
1: PUSH
2: Show
0: Exit
Enter choice:1
Enter no to push:20
****** STACK IMPLEMENTATION USING LIST ******
1: PUSH
2: Show
0: Exit
Enter choice:2
(TOP)20 <==
****** STACK IMPLEMENTATION USING LIST ******
1: PUSH
2: Show
0: Exit
Enter choice:0
Bye
>>>
```

Assignment 3:

Write a python program using function POP(Arr), where Arr is a stack implemented by a list of numbers. The function returns the value deleted from the stack.

Ans:

```
def isEmpty(Arr):
    if len(Arr)==0:
        return True
    else:
        return False

def push(Arr,item):
    Arr.append(item)
    top=len(Arr)-1

def pop(Arr):
    if isEmpty(Arr):
        return 'Underflow occurs'
    else:
        val=Arr.pop()
        if len(Arr)==0:
            top=None
        else:
            top=len(Arr)-1
        return val

def show(Arr):
    if isEmpty(Arr):
        print('no item found')
    else:
        t=len(Arr)-1
        print('(TOP)',end='')
        while(t>=0):
            print(Arr[t],'<==',end='')
            t=t-1
        print()

Arr=[]
top=None
while True:
    print('****** STACK IMPLEMENTATION USING LIST ******')
    print('1: PUSH')
    print('2: POP')
    print('3: Show')
    print('0: Exit')
    ch=int(input('Enter choice:'))
    if ch==1:
        val=int(input('Enter no to push:'))
        push(Arr,val)
    elif ch==2:

        val=pop(Arr)
        if val=='Underflow':
            print('Stack is empty')
        else:
```

```
        print('\nDeleted item is:',val)

    elif ch==3:
        show(Arr)
    elif ch==0:
        print('Bye')
        break
```

Output is:

```
****** STACK IMPLEMENTATION USING LIST ******
1: PUSH
```

```
2: POP
3: Show
0: Exit
Enter choice:1
Enter no to push:34
****** STACK IMPLEMENTATION USING LIST ******
1: PUSH
2: POP
3: Show
0: Exit
Enter choice:1
Enter no to push:3
****** STACK IMPLEMENTATION USING LIST ******
1: PUSH
2: POP
3: Show
0: Exit
Enter choice:3
(TOP)3 <==34 <==
****** STACK IMPLEMENTATION USING LIST ******
1: PUSH
2: POP
3: Show
0: Exit
Enter choice:2

Deleted item is: 3
****** STACK IMPLEMENTATION USING LIST ******
1: PUSH
2: POP
3: Show
0: Exit
Enter choice:3
(TOP)34 <==
****** STACK IMPLEMENTATION USING LIST ******
1: PUSH
2: POP
3: Show
0: Exit
Enter choice:0
Bye
>>>
```

Assignment 5: SQL Commands on EMPLOYEE table

MySQL Practical

1. CREATING TABLES IN MYSQL

E.g. in order to create table EMPLOYEE given below :

ECODE	ENAME	GENDER	GRADE	GROSS

We write the following command :

```
CREATE TABLE employee (
ECODE integer ,
    ENAME varchar(20) ,
    GENDER char(1) ,
    GRADE char(2) ,
    GROSS integer

    ) ;
```

2. INSERTING DATA INTO TABLE

- e.g. to enter a row into EMPLOYEE table (created above), we write command as :
  INSERT INTO employee VALUES(1001 , 'Ravi' , 'M' , 'E4' , 50000);

OR

INSERT INTO employee (ECODE , ENAME , GENDER , GRADE , GROSS) VALUES(1001 , 'Ravi' , 'M' , 'E4' , 50000);

ECODE	ENAME	GENDER	GRADE	GROSS
1001	Ravi	M	E4	50000

In order to insert another row in EMPLOYEE table , we write again INSERT command : INSERT INTO employee
VALUES(1002 , 'Akash' , 'M' , 'A1' , 35000);

ECODE	ENAME	GENDER	GRADE	GROSS
1001	Ravi	M	E4	50000
1002	Akash	M	A1	35000

3. INSERTING NULL VALUES

- To insert value NULL in a specific column, we can type NULL without quotes and NULL will be inserted in that column. E.g. in order to insert NULL value in ENAME column of above table, we write INSERT command as :

```
INSERT INTO EMPLOYEE
```

VALUES (1004 , NULL , 'M' , 'B2' , 38965) ;

ECODE	ENAME	GENDER	GRADE	GROSS
1001	Ravi	M	E4	50000
1002	Akash	M	A1	35000
1004	NULL	M	B2	38965

4. SIMPLE QUERY USING SELECT COMMAND

- The SELECT command is used to pull information from a table.
- In order to retrieve everything (all columns) from a table, SELECTcommand is used as :

SELECT * FROM <tablename> ;

e.g.In order to retrieve everything from **Employee** table, we write SELECT command as :

EMPLOYEE

ECODE	ENAME	GENDER	GRADE	GROSS
1001	Ravi	M	E4	50000
1002	Akash	M	A1	35000
1004	NULL	M	B2	38965

SELECT * FROM Employee ;

5. SELECTING PARTICULAR COLUMNS

EMPLOYEE

ECODE	ENAME	GENDER	GRADE	GROSS
1001	Ravi	M	E4	50000
1002	Akash	M	A1	35000
1004	Neela	F	B2	38965
1005	Sunny	M	A2	30000
1006	Ruby	F	A1	45000
1009	Neema	F	A2	52000

- A particular column from a table can be selected by specifying column-names with SELECTcommand.

E.g. in above table, if we want to select ECODE and ENAME column, then command is :

SELECT ECODE , ENAME

FROM EMPLOYEE ;

E.g.2 in order to select only ENAME, GRADE and GROSS column, the command is :

SELECT ENAME , GRADE , GROSS

FROM EMPLOYEE ;

6. SELECTING PARTICULAR ROWS

We can select particular rows from a table by specifying a condition through **WHERE clause** along with SELECT statement. **E.g.** In employee table if we want to select rows where Gender is female, then command is :

SELECT * FROM **EMPLOYEE**

WHERE **GENDER = 'F' ;**

E.g.2. in order to select rows where salary is greater than 48000, then

command is : SELECT * FROM **EMPLOYEE**

WHERE **GROSS > 48000 ;**

7. ELIMINATING REDUNDANT DATA

The **DISTINCT** keyword eliminates duplicate rows from the results of a SELECT statement. For example ,

SELECT GENDER **FROM** EMPLOYEE ;

GENDER
M
M
F
M
F
F

SELECT DISTINCT(GENDER) **FROM** EMPLOYEE ;

DISTINCT(GENDER)
M
F

8. VIEWING STRUCTURE OF A TABLE

- If we want to know the structure of a table, we can use DESCRIBE or DESC command, as per following syntax :

DESCRIBE | **DESC** <tablename> ;

e.g. to view the structure of table **EMPLOYEE**, commandis : **DESCRIBE** EMPLOYEE ; **OR DESC** EMPLOYEE ;

9. USING COLUMN ALIASES

- The columns that we select in a query can be given a different name, i.e. column alias name for output purpose.

e.g. In output, suppose we want to display ECODE column as EMPLOYEE_CODE in output , then

command is : SELECT **ECODE AS "EMPLOYEE_CODE"**

FROM **EMPLOYEE ;**

10. CONDITION BASED ON A RANGE

- The **BETWEEN** operator defines a range of values that the column values must fall in to make the condition true. The range include both lowervalue and uppervalue.

e.g. to display ECODE, ENAME and GRADE of those employees whose salary is between 40000 and 50000, command is:

```
SELECT ECODE , ENAME
,GRADE FROM
EMPLOYEE
WHERE GROSS BETWEEN 40000 AND 50000 ;
```

Output will be :

ECODE	ENAME	GRADE
1001	Ravi	E4
1006	Ruby	A1

11. CONDITION BASED ON A LIST

- To specify a list of values, IN operator is used. The IN operator selects value that match any value in a given list of values. E.g.

```
SELECT * FROM
EMPLOYEE WHERE
GRADE IN ('A1' , 'A2');
```

Output will be :

ECODE	ENAME	GENDER	GRADE	GROSS
1002	Akash	M	A1	35000
1006	Ruby	F	A1	45000
1005	Sunny	M	A2	30000
1009	Neema	F	A2	52000

- The **NOT IN** operatorfinds rows that do not match in the list. E.g.

```
SELECT * FROM EMPLOYEE
WHERE GRADE NOT IN ('A1' , 'A2');
```

Output will be :

ECODE	ENAME	GENDER	GRADE	GROSS
1001	Ravi	M	E4	50000
1004	Neela	F	B2	38965

12. CONDITION BASED ON PATTERN MATCHES

- LIKE operator is used for pattern matching in SQL. Patterns are described using two special wildcard characters:

1. percent(%) – The % character matches any substring.
2. underscore(_) – The _ character matches any character.

e.g. to display names of employee whose name starts with R in EMPLOYEE table, the command is :

```
SELECT ENAME FROM EMPLOYEE

WHERE ENAME LIKE 'R%' ;
```

Output will be :

ENAME
Ravi
Ruby

e.g. to display details of employee whose second character in name is 'e'. SELECT *
FROM **EMPLOYEE**

WHERE **ENAME LIKE '_e%' ;**

Output will be :

ECODE	ENAME	GENDER	GRADE	GROSS
1004	Neela	F	B2	38965
1009	Neema	F	A2	52000

e.g. to display details of employee whose name ends with 'y'. SELECT *
FROM **EMPLOYEE**

WHERE **ENAME LIKE '%y' ;**

Output will be :

ECODE	ENAME	GENDER	GRADE	GROSS
1005	Sunny	M	A2	30000
1006	Ruby	F	A1	45000

13. SEARCHING FOR NULL

- The NULL value in a column can be searched for in a table using IS NULL in the WHERE clause. E.g. to list employee details whose salary contain NULL, we use the command :

SELECT *****
FROM **EMPLOYEE**

WHERE GROSS **IS NULL ;**

e.g.

STUDENT

Roll_No	Name	Marks
1	ARUN	NULL
2	RAVI	56
4	SANJAY	NULL

to display the names of those students whose marks is NULL, we use the command : SELECT **Name**
FROM **EMPLOYEE**

WHERE Marks **IS NULL ;**

Output will be :

Name
ARUN
SANJAY

14. SORTING RESULTS

Whenever the SELECT query is executed , the resulting rows appear in a predecided order. The **ORDER BY clause** allow sorting of query result. The sorting can be done either in ascending or descending order, the default is ascending.

The ORDER BY clause is used as :

SELECT <column name> , <column name>…. FROM <tablename>

WHERE <condition>
ORDER BY <column name> ;

e.g. to display the details of employees in EMPLOYEE table in alphabetical order, we use command :

```
SELECT *
FROM
EMPLOYEE
ORDER BY
ENAME ;
```

Output will be :

ECODE	ENAME	GENDER	GRADE	GROSS
1002	Akash	M	A1	35000
1004	Neela	F	B2	38965
1009	Neema	F	A2	52000
1001	Ravi	M	E4	50000
1006	Ruby	F	A1	45000
1005	Sunny	M	A2	30000

e.g. display list of employee in descending alphabetical order whose salary is greater than 40000.

```
SELECT ENAME

FROM
EMPLOYEE
WHERE GROSS >
40000 ORDER BY
ENAME desc ;
```

Output will be :

ENAME
Ravi
Ruby
Neema

15. MODIFYING DATA IN TABLES

you can modify data in tables using UPDATE command of SQL. The UPDATE command specifies the rows to be changed using the WHERE clause, and the new data using the SET keyword.

e.g. to change the salary of employee of those in EMPLOYEE table having employee code 1009 to 55000.

UPDATE EMPLOYEE **SET** GROSS = 55000 **WHERE** ECODE = 1009 ;

16. UPDATING MORE THAN ONE COLUMNS

e.g. to update the salary to 58000 and grade to B2 for those employee whose employee code is 1001.

```
UPDATE EMPLOYEE

SET GROSS = 58000, GRADE='B2'
WHERE ECODE = 1009 ;
```

OTHER EXAMPLES

Increase the salary of each employee by 1000 in the EMPLOYEE table.

```
UPDATE EMPLOYEE
```

SET GROSS = GROSS +100 ;

Double the salary of employees having grade as 'A1' or 'A2' .

UPDATE EMPLOYEE

SET GROSS = GROSS * 2 ;

WHERE GRADE='A1' **OR** GRADE='A2' ;

Change the grade to 'A2' for those employees whose employee code is 1004 and name is Neela.

UPDATE EMPLOYEE

SET GRADE='A2'

WHERE ECODE=1004 **AND** GRADE='NEELA' ;

17. DELETING DATA FROM TABLES

To delete some data from tables, DELETE command is used. **The DELETE command removes rows from a table.** The syntax of DELETE command is:

DELETE FROM <tablename>

WHERE <condition> ;

For example, to remove the details of those employee from EMPLOYEE table whose grade is A1.

DELETE FROM EMPLOYEE

WHERE GRADE ='A1' ;

18. TO DELETE ALL THE CONTENTS FROM A TABLE

DELETE FROM EMPLOYEE ;

So if we do not specify any condition with WHERE clause, then all the rows of the table will be deleted. Thus above line will delete all rows from employee table.

19. DROPPING TABLES

The DROP TABLE command lets you drop a table from the database.

e.g. to drop a table employee, we need to write :

DROP TABLE employee ;

Once this command is given, the table name is no longer recognized and no more commands can be given on that table. After this command is executed, all the data in the table along with table structure will be deleted.

20. ALTER TABLE COMMAND

The ALTER TABLE command is used to change definitions of existing tables.(adding columns,deleting columns etc.). The ALTER TABLE command is used for :

1. adding columns to a table
2. Modifying column-definitions of a table.
3. Deleting columns of a table.
4. Adding constraints to table.
5. Enabling/Disabling constraints.

21. ADDING COLUMNS TO TABLE

To add a column to a table this command is used,

e.g. to add a new column ADDRESS to the EMPLOYEE table, we can write command as :

ALTER TABLE EMPLOYEE

ADD ADDRESS VARCHAR(50);

A new column by the name ADDRESS will be added to the table, where each row will contain NULL value for the new column.

ECODE	ENAME	GENDER	GRADE	GROSS	ADDRESS
1001	Ravi	M	E4	50000	NULL
1002	Akash	M	A1	35000	NULL
1004	Neela	F	B2	38965	NULL
1005	Sunny	M	A2	30000	NULL
1006	Ruby	F	A1	45000	NULL
1009	Neema	F	A2	52000	NULL

However **if you specify NOT NULL constraint while adding a new column**, MySQL adds the new column with the default value of that datatype e.g. for INT type it will add 0 , for CHAR types, it will add a space, and so on.

e.g. Given a table namely Testt with the following data in it.

Col1	Col2
1	A
2	G

Now following commands are given for the table. Predict the table contents after each of the following statements:

(i) ALTER TABLE testt ADD col3 INT ;

(ii) ALTER TABLE testt ADD col4 INT NOT NULL ;

(iii) ALTER TABLE testt ADD col5 CHAR(3) NOT NULL ;

(iv) ALTER TABLE testtADD col6 VARCHAR(3);

22.MODIFYING COLUMNS

In table EMPLOYEE, change the column GROSS to SALARY.

ALTER TABLE EMPLOYEE

CHANGE GROSS SALARY INTEGER;

In table EMPLOYEE , change the column ENAME to EM_NAME and data type from VARCHAR(20) to VARCHAR(30).

ALTER TABLE EMPLOYEE

CHANGE ENAME EM_NAME VARCHAR(30);

In table EMPLOYEE , change the datatype of GRADE column from CHAR(2) to VARCHAR(2).

ALTER TABLE EMPLOYEE

MODIFY GRADE VARCHAR(2);

23. DELETING COLUMNS

To delete a column from a table, the ALTER TABLE command takes the following form :

ALTER TABLE <table name>

DROP <column name>;

e.g. to delete column GRADE from table EMPLOYEE, we will write :

ALTER TABLE EMPLOYEE

DROP GRADE ;

24. ADDING/REMOVING CONSTRAINTS TO A TABLE

ALTER TABLE statement can be used to add constraints to your existing table.
e.g. to add PRIMARY KEY constraint on column ECODE of table EMPLOYEE , the command is :

ALTER TABLE EMPLOYEE

ADD PRIMARY KEY (ECODE) ;

25. REMOVING CONSTRAINTS

- To remove foreign key constraint from atable, we use ALTER TABLE command as :

ALTER TABLE <table name>

DROP FOREIGN KEY ;

AGGREGATE / GROUP FUNCTIONS

Aggregate / Group functions work upon groups of rows , rather than on single row, and return one single output. Different aggregate functions are : COUNT( ) , AVG( ) , MIN( ) , MAX( ) , SUM ( )

Table : EMPL

EMPNO	ENAME	JOB	SAL	DEPTNO
8369	SMITH	CLERK	2985	10
8499	ANYA	SALESMAN	9870	20
8566	AMIR	SALESMAN	8760	30
8698	BINA	MANAGER	5643	20
8912	SUR	NULL	3000	10

1. AVG( )

This function computes the average of given data.

e.g. SELECT AVG(SAL)
FROM EMPL ;

Output

AVG(SAL)
6051.6

2. COUNT( )

This function counts the number of rows in a given column.

If you specify the COLUMN name in parenthesis of function, then this function returns rows where COLUMN is not null.

If you specify the asterisk (*), this function returns all rows, including duplicates and nulls.

e.g. SELECT COUNT(*) FROM EMPL ;

Output

COUNT(*)
5

e.g.2 SELECT COUNT(JOB) FROM EMPL ;

Output

COUNT(JOB)
4

3. MAX( )

This function returns the maximum value from a given column or expression.

e.g. SELECT MAX(SAL) FROM EMPL ;

Output

MAX(SAL)
9870

4. MIN( )

This function returns the minimum value from a given column or expression.

e.g. SELECT MIN(SAL) FROM EMPL ;

Output

MIN(SAL)
2985

5. SUM( )

This function returns the sum of values in given column or expression.

e.g. SELECT SUM(SAL) FROM EMPL ;

Output

SUM(SAL)
30258

GROUPING RESULT – GROUP BY

The GROUP BY clause combines all those records(row) that have identical values in a particular field(column) or a group of fields(columns).
GROUPING can be done by a column name, or with aggregate functions in which case the aggregate produces a value for each group.

Table : EMPL

EMPNO	ENAME	JOB	SAL	DEPTNO
8369	SMITH	CLERK	2985	10
8499	ANYA	SALESMAN	9870	20
8566	AMIR	SALESMAN	8760	30
8698	BINA	MANAGER	5643	20

e.g. Calculate the number of employees in each grade.

```
SELECT JOB, COUNT(*) FROM EMPL
GROUP BY JOB ;
```

Output

JOB	COUNT(*)
CLERK	1
SALESMAN	2
MANAGER	1

e.g.2. Calculate the sum of salary for each department.

```
SELECT DEPTNO ,
SUM(SAL) FROM
EMPL
GROUP BY DEPTNO ;
```

Output

DEPTNO	SUM(SAL)
10	2985
20	15513
30	8760

e.g.3. find the average salary of each department.

Sol: select avg(sal) FROM EMPL

GROUP BY DEPTNO ;

NESTED GROUP

- To create a group within a group i.e., nested group, you need to specify multiplefields in the GROUP BY expression.

e.g. To group records **job wise** within **Deptno wise**, you need to issue a query statement like :

SELECT DEPTNO , JOB ,
COUNT(EMPNO) FROM EMPL
GROUP BY DEPTNO , JOB ;

Output

DEPTNO	JOB	COUNT(EMPNO)
10	CLERK	1
20	SALESMAN	1
20	MANAGER	1
30	SALESMAN	1

PLACING CONDITION ON GROUPS – HAVING CLAUSE

- The **HAVING clause places conditions on groups** in contrast to WHERE clause that places condition on individual rows. While **WHERE conditions cannot include aggregate functions, HAVING conditions can do so.**
- e.g. To display the jobs where the number of employees is less than 2,
- SELECT JOB, COUNT(*) FROM EMPL GROUP BY JOB
  HAVING COUNT(*) < 2 ;

Output

JOB	COUNT(*)
CLERK	1
MANAGER	1

MySQL FUNCTIONS

Types of MySQL functions : String Functions , Maths Functions and Date & Time Functions.

Table : EMPL

EMPNO	ENAME	JOB	SAL	DEPTNO
8369	SMITH	CLERK	2985	10
8499	ANYA	SALESMAN	9870	20
8566	AMIR	SALESMAN	8760	30
8698	BINA	MANAGER	5643	20
8912	SUR	NULL	3000	10

STRING FUNCTIONS

1. **CONCAT( )** - Returns the Concatenated String.
   Syntax : CONCAT(Column1 , Column2 , Column3, …….)
   e.g. SELECT CONCAT(EMPNO , ENAME) FROM EMPL WHEREDEPTNO=10;

Output

CONCAT(EMPNO , ENAME)
8369SMITH
8912SUR

2. **LOWER( ) / LCASE( )** - Returns the argument in lowercase. Syntax : LOWER(Column name)

   e.g.

   SELECT LOWER(ENAME) FROM EMPL ;

Output

LOWER(ENAME)
smith
anya
amir
bina
sur

3. **UPPER( ) / UCASE( )** - Returns the argument in uppercase. Syntax : UPPER(Column name)

   e.g.

   SELECT UPPER(ENAME) FROM EMPL ;

Output

UPPER(ENAME)
SMITH
ANYA
AMIR
BINA
SUR

4. **SUBSTRING( ) / SUBSTR( )** – Returns the substring as specified.

Syntax : SUBSTR(Column name, m , n), where **m specifies starting index** and **n specifies number of characters from the starting index m.**

e.g.

SELECT SUBSTR(ENAME,2,2) FROM EMPL WHERE DEPTNO=20;

Output

SUBSTR(ENAME,2,2)
NY
IN

SELECT SUBSTR(JOB,-2,2) FROMEMPL WHERE DEPTNO=20;

Output

SUBSTR(JOB,-4,2)
SM
AG

5. **LTRIM( )** – Removes leadingspaces.

e.g. SELECT LTRIM(‘ RDBMS MySQL’) ;

Output

LTRIM(‘ RDBMS MySQL’)
RDBMS MySQL

6. **RTRIM( )** – Removes trailing spaces.

e.g. SELECT RTRIM(‘ RDBMS MySQL ’) ;

Output

RTRIM(‘ RDBMS MySQL’)
RDBMS MySQL

7. **TRIM( )** – Removes trailing and leading spaces.

e.g. SELECT TRIM(‘ RDBMS MySQL ’) ;

Output

TRIM(‘ RDBMS MySQL’)
RDBMS MySQL

8. **LENGTH( )** – Returns the length of a string. e.g.

SELECT LENGTH(“CANDID”) ;

Output

LENGTH(“CANDID”)
6

e.g.2. Output

LENGTH(ENAME)
5
4
4
4
3

```
SELECT LENGTH(ENAME) FROM EMPL;
```

9. **LEFT( )** – Returns the leftmost numberof characters as specified.

e.g. SELECT LEFT('CORPORATE FLOOR' , 3) ;

Output

LEFT('CORPORATE FLOOR', 3)
COR

10. **RIGHT( )** – Returns the rightmost numberof characters as specified.

e.g. SELECT RIGHT('CORPORATE FLOOR' , 3) ;

Output

RIGHT('CORPORATE FLOOR', 3)
OOR

11. **MID( )** – This function is same as SUBSTRING( ) / SUBSTR( ) function. E.g. SELECT MID("ABCDEF" , 2 , 4) ;

Output

MID("ABCDEF" , 2 , 4)
BCDE

NUMERIC FUNCTIONS

These functions accept numeric values and after performing the operation, return numeric value.

1. **MOD( )** – Returns the remainder of given two numbers. e.g. SELECT MOD(11 , 4);

Output

MOD(11, 4)
3

2. **POW( ) / POWER( )** - This function returns m^n i.e , a numberm raised to the n^{th} power.

e.g. SELECT POWER(3,2) ;

Output

POWER(3, 2)
9

3. **ROUND( )** – This function returns a numberroundedoff as per given specifications.

e.g. ROUND(15.193 , 1) ;

Output

ROUND(15.193 , 1)
15.2

e.g. 2. SELECT ROUND(15.193 , -1); - This will convert the number to nearest ten's .

Output

ROUND(15.193 , -1)
20

4. **SIGN( )** – This function returns sign of a given number. If number is negative, the function returns -1. If number is positive, the function returns 1. If number is zero, the function returns 0.

e.g. SELECT SIGN(-15) ;

Output

SIGN(-15)
-1

e.g.2 SELECT SIGN(20) ;

Output

SIGN(20)
1

5. **SQRT( )** – This function returns the square root of a given number. E.g. SELECT SQRT(25) ;

Output

SQRT(25)
5

6. **TRUNCATE( )** – This function returns a number with some digits truncated. E.g. SELECT TRUNCATE(15.79 , 1) ;

Output

TRUNCATE(15.79 , 1)
15.7

E.g. 2. SELECT TRUNCATE(15.79 , -1); - This command truncate value 15.79 to nearest ten's place.

Output

TRUNCATE(15.79 , -1)
10

DATE AND TIME FUNCTIONS

Date functions operate on values of the DATE datatype.

1. **CURDATE( ) / CURRENT_DATE( )** – This function returns the current date. E.g.

SELECT CURDATE( ) ;

Output

CURDATE( )
2016-12-13

2. **DATE( )** – This function extracts the date part from a date. E.g.

SELECT DATE('2016-02-09') ;

Output

DATE('2016-02-09')
09

3. **MONTH( )** – This function returns the month from the date passed. E.g. SELECT MONTH('2016-02-09') ;

Output

MONTH('2016-02-09')
02

4. **YEAR( )** – This function returns the yearpart of a date. E.g.

SELECT YEAR('2016-02-09') ;

Output

YEAR('2016-02-09')
2016

5. **DAYNAME( )** – This function returns the name of weekday. E.g.

SELECT DAYNAME('2016-02-09') ;

Output

DAYNAME('2016-12-14')
Wednesday

6. **DAYOFMONTH( )** – This function returns the day of month. Returns value in range of 1 to 31.

E.g. SELECT DAYOFMONTH('2016-12-14') ;

Output

DAYOFMONTH('2016-12-14')
14

7. **DAYOFWEEK( )** – This function returns the day of week. Return the weekdayindex for date. (1=Sunday, 2=Monday,……., 7=Saturday)

SELECT DAYOFWEEK('2016-12-14') ;

Output

DAYOFWEEK('2016-12-14')
4

8. **DAYOFYEAR( )** – This function returns the day of the year. Returns the value between 1 and 366. E.g. SELECT DAYOFYEAR('2016-02-04) ;

Output

DAYOFYEAR('2016-02-04')
35

9. **NOW( ) –** This function returns the currentdate and time.
It returns a constant time that indicates the time at which the statement began to execute.

e.g. SELECT NOW( );

10. **SYSDATE( ) –** It also returns the current date but it return the time at which SYSDATE( ) executes. It differs from the behavior for NOW( ), which returns a constant time that indicates the time at which the statement began to execute.

e.g. SELECT SYSDATE( ) ;

JOINS

- A join is a query that combines rows from two or more tables. In a join- query, more than one table are listed in FROM clause.

Table : empl

EMPNO	ENAME	JOB	SAL	DEPTNO
8369	SMITH	CLERK	2985	10
8499	ANYA	SALESMAN	9870	20
8566	AMIR	SALESMAN	8760	30
8698	BINA	MANAGER	5643	20

Table : dept

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW DELHI
20	RESEARCH	CHENNAI
30	SALES	KOLKATA
40	OPERATIONS	MUMBAI

CARTESIAN PRODUCT/UNRESTRICTED JOIN/CROSS JOIN

- Consider the following query :

SELECT * FROM EMPL, DEPT ;

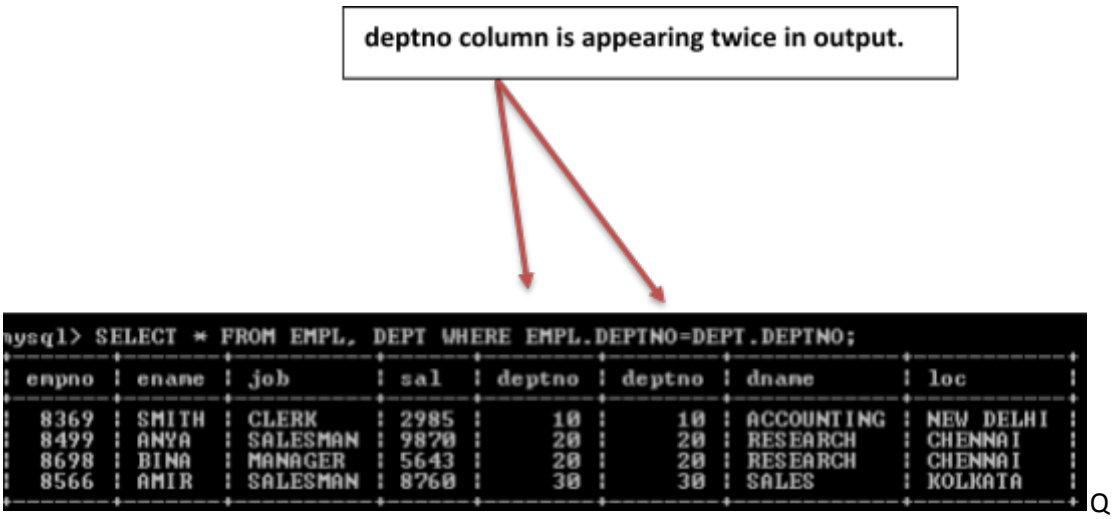
empno	ename	job	sal	deptno	deptno	dname	loc
8369	SMITH	CLERK	2985	10	10	ACCOUNTING	NEW DELHI
8499	ANYA	SALESMAN	9870	20	10	ACCOUNTING	NEW DELHI
8566	AMIR	SALESMAN	8760	30	10	ACCOUNTING	NEW DELHI
8698	BINA	MANAGER	5643	20	10	ACCOUNTING	NEW DELHI
8369	SMITH	CLERK	2985	10	20	RESEARCH	CHENNAI
8499	ANYA	SALESMAN	9870	20	20	RESEARCH	CHENNAI
8566	AMIR	SALESMAN	8760	30	20	RESEARCH	CHENNAI
8698	BINA	MANAGER	5643	20	20	RESEARCH	CHENNAI
8369	SMITH	CLERK	2985	10	30	SALES	KOLKATA
8499	ANYA	SALESMAN	9870	20	30	SALES	KOLKATA
8566	AMIR	SALESMAN	8760	30	30	SALES	KOLKATA
8698	BINA	MANAGER	5643	20	30	SALES	KOLKATA
8369	SMITH	CLERK	2985	10	40	OPERATIONS	MUMBAI
8499	ANYA	SALESMAN	9870	20	40	OPERATIONS	MUMBAI
8566	AMIR	SALESMAN	8760	30	40	OPERATIONS	MUMBAI
8698	BINA	MANAGER	5643	20	40	OPERATIONS	MUMBAI

This query will give you the Cartesian product i.e. all possible concatenations are formed of all rows of both the tables EMPL and DEPT. Such an operation is also known as **Unrestricted Join.** It returns n1 x n2 rows where n1 is number of rows in first table and n2 is number of rows in second table.

EQUI-JOIN

- The join in which columns are compared for equality, is called Equi- Join. In equi-join, all the columns from joining table appear in the output even if they are identical.

  e.g.
  ```
  SELECT * FROM empl, dept
  WHERE empl.deptno = dept.deptno ;
  ```


Q

Q: with reference to empl and dept table, find the location of employee SMITH.

ename column is present in empl and loc column is present in dept. In order to obtain the result, we have to join two tables.

```
SELECT ENAME, LOC
FROM EMPL, DEPT
WHERE EMPL.DEPTNO = DEPT.DEPTNO AND ENAME='SMITH';
```

ENAME	LOC
SMITH	NEW DELHI

Q: Display details like department number, department name, employee number, employee name, job and salary. And order the rows by department number.

```
SELECT EMPL.deptno, dname,empno,ename,job,sal
FROM EMPL,DEPT
WHERE EMPL.DEPTNO=DEPT.DEPTNO
ORDER BY EMPL.DEPTNO;
```

deptno	dname	empno	ename	job	sal
10	ACCOUNTING	8369	SMITH	CLERK	2985
20	RESEARCH	8698	BINA	MANAGER	5643
20	RESEARCH	8499	ANYA	SALESMAN	9870
30	SALES	8566	AMIR	SALESMAN	8760

QUALIFIED NAMES

Did you notice that in all the WHERE conditions of join queries given so far, the field(column) names are given as: **<tablename>.<columnname>**

This type of field names are called qualified field names. Qualified field names are very useful in identifying a field if the two joining tables have fields with same time. For example, if we say deptno field from joining tables empl and dept, you'll definitely ask- **deptno** field of which table ? To avoid such an ambiguity, the qualified field names are used.

TABLE ALIAS

- A table alias is a temporary label given along with table name in FROM clause.

e.g.

```
SELECT E.DEPTNO,
DNAME,EMPNO,ENAME,JOB,SAL FROM
EMPL E, DEPT D
WHERE E.DEPTNO
= D.DEPTNO
ORDER BY
E.DEPTNO;
```

In above command table alias for EMPL table is E and for DEPT table , alias is D.

Q: Display details like department number, department name, employee number, employee name, job and salary. And order the rows by employee number with department number. These details should be only for employees earning atleast Rs. 6000 and of SALESdepartment.

```
SELECT E.DEPTNO, DNAME,EMPNO, ENAME, JOB, SAL FROM EMPL E, DEPT D
WHERE E.DEPTNO = D.DEPTNO AND DNAME='SALES' AND SAL>=6000
ORDER BY E.DEPTNO;
```

DEPTNO	DNAME	EMPNO	ENAME	JOB	SAL
30	SALES	8566	AMIR	SALESMAN	8760

NATURAL JOIN

By default, the results of an equijoin contain two identical columns. One of the two identical columns can be eliminated by restating the query. This result is called a Natural join.

e.g.
```
SELECT empl.*, dname, loc FROM empl,dept
WHERE empl.deptno = dept.deptno ;
```

empno	ename	job	sal	deptno	dname	loc
8369	SMITH	CLERK	2985	10	ACCOUNTING	NEW DELHI
8499	ANYA	SALESMAN	9870	20	RESEARCH	CHENNAI
8698	BINA	MANAGER	5643	20	RESEARCH	CHENNAI
8566	AMIR	SALESMAN	8760	30	SALES	KOLKATA

empl.* means select all columns from empl table. This thing can be used with any table.

LEFT JOIN

- You can use LEFT JOIN clause in SELECT to produce left join i.e.

- When using LEFT JOIN all rows from the first table will be returned whether there are matches in the second table or not. For unmatched rows of first table, NULL is shown in columns of second table.

S1

Roll_no	Name
1	A
2	B
3	C
4	D
5	E
6	F

S2

Roll_no	Class
2	III
4	IX
1	IV
3	V
7	I
8	II

```
SELECT S1.ROLL_NO, NAME,CLASS
FROM S1 LEFT JOIN S2 ON S1.ROLL_NO=S2.ROLL_NO;
```

```
+---------+------+-------+
| ROLL_NO | NAME | CLASS |
+---------+------+-------+
|       1 | A    | IV    |
|       2 | B    | III   |
|       3 | C    | V     |
|       4 | D    | IX    |
|       5 | E    | NULL  |
|       6 | F    | NULL  |
+---------+------+-------+
6 rows in set (0.00 sec)
```

RIGHT JOIN

- It works just like LEFT JOIN but with table order reversed. All rows from the second table are going to be returned whether or not there are matches in the first table.
- You can use RIGHT JOIN in SELECT to produce right join i.e.

e.g

```
SELECT S1.ROLL_NO, NAME,CLASS
FROM S1 RIGHT JOIN S2 ON S1.ROLL_NO=S2.ROLL_NO;
```

```
+---------+------+-------+
| ROLL_NO | NAME | CLASS |
+---------+------+-------+
|       2 | B    | III   |
|       4 | D    | IX    |
|       1 | A    | IV    |
|       3 | C    | V     |
|    NULL | NULL | I     |
|    NULL | NULL | II    |
+---------+------+-------+
```

Assignment 6:

Integrate MySQL with Python by importing the MySQL module and add records of student and display all the record.

Ans:

```
import os
import platform
import mysql.connector

mydb=mysql.connector.connect(host="localhost",user="root",passwd="root",database="student",
charset="utf8")
print(mydb)
mycursor=mydb.cursor()

def stuInsert():
    L=[]
    roll=int(input("Enter the roll number : "))
    L.append(roll)
    name=input("Enter the Name: ")
    L.append(name)
    age=int(input("Enter Age of Student : "))
    L.append(age)
    clas=input("Enter the Class : ")
    L.append(clas)

    stud=(L)
    sql="insert into stud (roll,name,age,clas) values (%s,%s,%s,%s)"
    mycursor.execute(sql,stud)
    mydb.commit()
def stuview():
    mycursor.execute("select * from stud")
    myrus=mycursor.fetchall()
    for x in myrus:
        print(x)

def MenuSet(): #Function For The Student Management System
    print("Enter 1 : To Add Student")
    print("Enter 2 : To View Students")
    userInput = int(input("Please Select An Above Option: ")) #Will Take Input From User
    if(userInput == 1):
        stuInsert()
    if(userInput == 2):
        stuview()
MenuSet()
def runAgain():
    runAgn = input("\nwant To Run Again Y/n: ")
    while(runAgn.lower() == 'y'):
        if(platform.system() == "Windows"):
```

```
            print(os.system('cls'))
        else:
            print(os.system('clear'))
        MenuSet()
        runAgn = input("\nwant To Run Again y/n: ")

runAgain()
```

Output is:

```
<mysql.connector.connection.MySQLConnection object at 0x02272110>
Enter 1 : To Add Student
Enter 2 : To View Students
Please Select An Above Option: 2
(1, 'ANJU JHA', 17, 12)
(2, 'YASH', 16, 11)
(3, 'ANIKET JAISWAR', 16, 12)
(4, 'SANGEETA', 15, 10)
(5, 'SANGEETA SETH', 15, 10)
(6, 'YAMINI', 16, 11)
(7, 'ANJU', 15, 10)
(8, 'OM', 16, 12)
(9, 'MANGALA', 16, 11)

want To Run Again Y/n: Y
0
Enter 1 : To Add Student
Enter 2 : To View Students
Please Select An Above Option: 1
Enter the roll number : 10
Enter the Name: MALINI
Enter Age of Student : 17
Enter the Class : 12

want To Run Again y/n: Y
0
Enter 1 : To Add Student
Enter 2 : To View Students
Please Select An Above Option: 2
(1, 'ANJU JHA', 17, 12)
(2, 'YASH', 16, 11)
(3, 'ANIKET JAISWAR', 16, 12)
(4, 'SANGEETA', 15, 10)
(5, 'SANGEETA SETH', 15, 10)
(6, 'YAMINI', 16, 11)
(7, 'ANJU', 15, 10)
(8, 'OM', 16, 12)
(9, 'MANGALA', 16, 11)
(10, 'MALINI', 17, 12)
```

want To Run Again y/n:

Assignment 7:

Integrate MySQL with Python by importing the MySQL module to search student using rollno, name, age, class and if present in table display the record, if not display appropriate method.

Ans:

```
import os
import platform
import mysql.connector

mydb=mysql.connector.connect(host="localhost",\
                    user="root",\
                    passwd="root",\
                    database="student",charset="utf8")
print(mydb)
mycursor=mydb.cursor()

def stuview():
   print("Select the search criteria : ")
   print("1. Roll")
   print("2. Name")
   print("3. Age")
   print("4. Class")
   print("5. All")
   ch=int(input("Enter the choice : "))
   if ch==1:
      s=int(input("Enter roll no : "))
      rl=(s,)
      sql="select * from stud where roll=%s"
      mycursor.execute(sql,rl)
   elif ch==2:
      s=input("Enter Name : ")
      rl=(s,)
      sql="select * from stud where name=%s"
      mycursor.execute(sql,rl)
   elif ch==3:
      s=int(input("Enter age : "))
      rl=(s,)
      sql="select * from stud where age=%s"
      mycursor.execute(sql,rl)
   elif ch==4:
      s=input("Enter Class : ")
      rl=(s,)
      sql="select * from stud where clas=%s"
      mycursor.execute(sql,rl)
   elif ch==5:
      sql="select * from stud"
      mycursor.execute(sql)
   res=mycursor.fetchall()
```

```
    print("The Students details are as follows : ")
    print("(ROll, Name, Age, Class)")
    for x in res:
        print(x)

def MenuSet(): #Function For The Student Management System
    print("Enter 1 : To Search Student")
    userInput = int(input("Please Select An Above Option: ")) #Will Take Input From User
    if(userInput == 1):
        stuview()
MenuSet()
def runAgain():
    runAgn = input("\nwant To Run Again Y/n: ")
    while(runAgn.lower() == 'y'):
        if(platform.system() == "Windows"):
            print(os.system('cls'))
        else:
            print(os.system('clear'))
        MenuSet()
        runAgn = input("\nwant To Run Again y/n: ")

runAgain()
```

Output is:

```
<mysql.connector.connection.MySQLConnection object at 0x022720F0>
Enter 1 : To Search Student
Please Select An Above Option: 1
Select the search criteria :
1. Roll
2. Name
3. Age
4. Class
5. All
Enter the choice : 1
Enter roll no : 8
The Students details are as follows :
(ROll, Name, Age, Class)
(8, 'OM', 16, 12)

want To Run Again Y/n: y
0
Enter 1 : To Search Student
Please Select An Above Option: 1
Select the search criteria :
1. Roll
2. Name
3. Age
4. Class
5. All
```

```
Enter the choice : 2
Enter Name : MALINI
The Students details are as follows :
(ROll, Name, Age, Class)
(10, 'MALINI', 17, 12)

want To Run Again y/n: Y
0
Enter 1 : To Search Student
Please Select An Above Option: 1
Select the search criteria :
1. Roll
2. Name
3. Age
4. Class
5. All
Enter the choice : 3
Enter age : 15
The Students details are as follows :
(ROll, Name, Age, Class)
(4, 'SANGEETA', 15, 10)
(5, 'SANGEETA SETH', 15, 10)
(7, 'ANJU', 15, 10)

want To Run Again y/n: Y
0
Enter 1 : To Search Student
Please Select An Above Option: 1
Select the search criteria :
1. Roll
2. Name
3. Age
4. Class
5. All
Enter the choice : 4
Enter Class : 12
The Students details are as follows :
(ROll, Name, Age, Class)
(1, 'ANJU JHA', 17, 12)
(3, 'ANIKET JAISWAR', 16, 12)
(8, 'OM', 16, 12)
(10, 'MALINI', 17, 12)

want To Run Again y/n: Y
0
Enter 1 : To Search Student
Please Select An Above Option: 1
Select the search criteria :
1. Roll
2. Name
```

3. Age
4. Class
5. All
Enter the choice : 5
The Students details are as follows :
(ROll, Name, Age, Class)
(1, 'ANJU JHA', 17, 12)
(2, 'YASH', 16, 11)
(3, 'ANIKET JAISWAR', 16, 12)
(4, 'SANGEETA', 15, 10)
(5, 'SANGEETA SETH', 15, 10)
(6, 'YAMINI', 16, 11)
(7, 'ANJU', 15, 10)
(8, 'OM', 16, 12)
(9, 'MANGALA', 16, 11)
(10, 'MALINI', 17, 12)

want To Run Again y/n: N
>>>

Assignment 8:

Integrate MySQL with Python by importing the MySQL module to search a student using rollno, delete the record.

Ans:

```
import os
import platform
import mysql.connector

mydb=mysql.connector.connect(host="localhost",\
                    user="root",\
                    passwd="root",\
                    database="student",charset="utf8")
print(mydb)
mycursor=mydb.cursor()

def removeStu():
   roll=int(input("Enter the roll number of the student to be deleted : "))
   rl=(roll,)
   sql="Delete from stud where roll=%s"
   mycursor.execute(sql,rl)
   print('Record deleted!!!')
   mydb.commit()

def stuview():
   mycursor.execute("select * from stud")
   myrus=mycursor.fetchall()
   for x in myrus:
      print(x)

def MenuSet(): #Function For The Student Management System
   print("Enter 1 : To Delete Student")
   print("Enter 2 : To View Students")
   userInput = int(input("Please Select An Above Option: ")) #Will Take Input From User
   if(userInput == 1):
      removeStu()
   if(userInput == 2):
      stuview()
MenuSet()
def runAgain():
   runAgn = input("\nwant To Run Again Y/n: ")
   while(runAgn.lower() == 'y'):
      if(platform.system() == "Windows"):
         print(os.system('cls'))
      else:
         print(os.system('clear'))
      MenuSet()
      runAgn = input("\nwant To Run Again y/n: ")
```

runAgain()

Output is:

```
<mysql.connector.connection.MySQLConnection object at 0x02272050>
Enter 1 : To Delete Student
Enter 2 : To View Students
Please Select An Above Option: 2
(1, 'ANJU JHA', 17, 12)
(2, 'YASH', 16, 11)
(3, 'ANIKET JAISWAR', 16, 12)
(4, 'SANGEETA', 15, 10)
(5, 'SANGEETA SETH', 15, 10)
(6, 'YAMINI', 16, 11)
(7, 'ANJU', 15, 10)
(8, 'OM', 16, 12)
(9, 'MANGALA', 16, 11)
(10, 'MALINI', 17, 12)

want To Run Again Y/n: y
0
Enter 1 : To Delete Student
Enter 2 : To View Students
Please Select An Above Option: 1
Enter the roll number of the student to be deleted : 8
Record deleted!!!

want To Run Again y/n: y
0
Enter 1 : To Delete Student
Enter 2 : To View Students
Please Select An Above Option: 2
(1, 'ANJU JHA', 17, 12)
(2, 'YASH', 16, 11)
(3, 'ANIKET JAISWAR', 16, 12)
(4, 'SANGEETA', 15, 10)
(5, 'SANGEETA SETH', 15, 10)
(6, 'YAMINI', 16, 11)
(7, 'ANJU', 15, 10)
(9, 'MANGALA', 16, 11)
(10, 'MALINI', 17, 12)

want To Run Again y/n: n
```

Assignment 9:

Integrate SQL with Python by importing the MySQL module to search a student using rollno, update the record.

Ans:

```
import mysql.connector as mycon
cn =
mycon.connect(host='localhost',user='root',password="root",database="student",charset="utf8")
cur = cn.cursor()
print('Welcome to student Details Updation screen... ')

print("******************EDIT STUDENT DETAILS *************************")
ro = int(input("Enter Student's roll number to edit :"))
query="select * from stud where roll="+str(ro)
cur.execute(query)
results = cur.fetchall()
if cur.rowcount<=0:
   print("\## SORRY! NO MATCHING DETAILS AVAILABLE ##")
else:
   print("************************************************")
   print('%5s'%"ROLL NO",'%15s'%'NAME','%12s'%'AGE','%10s'%'CLASS')
   print("************************************************")
   for row in results:
      print('%5s' % row[0],'%15s'%row[1],'%12s'%row[2],'%10s'%row[3])
print("-"*50)
ans = input("Are you sure to update ? (y/n)")
if ans=="y" or ans=="Y":
   d = input("Enter new name to update (enter old value if not to update) :")
   s = int(input("Enter new age to update (enter old value if not to update) :"))

   query="update stud set name='"+d+"',age="+str(s) + " where roll="+str(ro)

   cur.execute(query)
   cn.commit()
   print("\n## RECORD UPDATED  ##")
```

Output is:

```
Welcome to student Details Updation screen...
******************EDIT STUDENT DETAILS *************************
Enter Student's roll number to edit :1
************************************************
ROLL NO          NAME         AGE      CLASS
************************************************
   1       ANJU JHA          17        12
------------------------------------------------
Are you sure to update ? (y/n)y
Enter new name to update (enter old value if not to update) :ANJU
Enter new age to update (enter old value if not to update) :16

## RECORD UPDATED  ##
```