NAME:AVINASH BHIWALE ROLL NO.:T1851050 PRN NO.:71901284J DIV:A

Assignment No: 6

Aim: Execute the aggregate functions like count, sum, avg etc. on the suitable database. Make use of built in functions according to the need of the database chosen. Retrieve the data from the database based on time and date functions like now (), date (), day (), time () etc. Use group by and having clauses.

Objective:

O To understand and implement various types of function in MYSQL.
O To learn the concept of group functions

NUMBER FUNCTION*:*

Abs(n) :Select abs(-15) from dual; Exp(n):
Select exp(4) from dual; Power(m,n): Select
power(4,2) from dual; Mod(m,n): Select
mod(10,3) from dual; Round(m,n): Select
round(100.256,2) from dual; Trunc(m,n):
;Select trunc(100.256,2) from dual;
Sqrt(m,n);Select sqrt(16) from dual; **Aggregate**
Functions:

1. Count: COUNT following by a column name returns the count of tuple in that column. If DISTINCT keyword is used then it will return only the count of unique tuple in the column. Otherwise, it will return count of all the tuples (including duplicates) count (*) indicates all the tuples of the column.

Syntax: COUNT (Column name)
Example: SELECT COUNT (Sal) FROM emp;

2. SUM: SUM followed by a column name returns the sum of all the values in that column. ***Syntax:*** SUM (Column name)

Example: SELECT SUM (Sal) From emp;

3. AVG: AVG followed by a column name returns the average value of that column values. ***Syntax:*** AVG (n1, n2...)

Example: Select AVG (10, 15, 30) FROM DUAL;

4. MAX: MAX followed by a column name returns the maximum value of that column. ***Syntax:*** MAX (Column name)

Example: SELECT MAX (Sal) FROM emp;

```
mysql> select deptno, max(sal) from emp group by deptno;
```

DEPTNO MAX (SAL)

10 5000
20 3000

30 2850

mysql> select deptno, max (sal) from emp group by deptno having max(sal)<3000;

DEPTNO MAX(SAL)

302850

5. MIN: MIN followed by column name returns the minimum value of that column. ***Syntax:*** MIN (Column name)

Example: SELECT MIN (Sal) FROM emp;

mysql> select deptno,min(sal) from emp group by deptno having min(sal)>1000;

DEPTNO MIN (SAL)

101300

DATE FUNCTIONS:

CURDATE()

Returns the current date as a value in 'YYYY-MM-DD' or YYYYMMDD format, depending on whether the function is used in a string or numeric context.

```
mysql> SELECT CURDATE();
+---------------+
| CURDATE()     |
+---------------+
```

```
+------------+
| 1997-12-15 |
+------------+
```

1 row in set (0.00 sec)

CURTIME()

Returns the current time as a value in 'HH:MM:SS' or HHMMSS format, depending on whether the function is used in a string or numeric context. The value is expressed in the current time zone.

```
mysql> SELECT CURTIME();
+-----------+
| CURTIME() |
+-----------+
| 23:50:26  |
+-----------+
```

1 row in set (0.00 sec)

DATE(expr)

Extracts the date part of the date or datetime expression expr.

```
mysql> SELECT DATE('2003-12-31 01:02:03');
+-----------------------------+
| DATE('2003-12-31 01:02:03') |
+-----------------------------+
| 2003-12-31                  |
+-----------------------------+
```

```
_____
_____
_____
. ..........+
1 row in set (0.00 sec)
```

DAY(date)

DAY() is a synonym for DAYOFMONTH().

DAYNAME(date)

Returns the name of the weekday for date.

```
mysql> SELECT DAYNAME('1998-02-05');
                                    +
   |                                |
_____
_____
_____
          -+                        +
   |       Thursday                 |
          -+                        +
1 row in set (0.00 sec)
```

HOUR(time)

Returns the hour for the time. The range of the return value is 0 to 23 for time-of-day values. However, the range of TIME values actually is much larger, so HOUR can return values greater than 23.

```
mysql> SELECT HOUR('10:05:03');
```

```
---------+                                        +
|              HOUR('10:05:03')                    |
_____________
_____________
_____________
---------+                                        +
|                          10                      |
+---------------------------------------------------+
```

1 row in set (0.00 sec)

LAST_DAY(date)

Takes a date or datetime value and returns the corresponding value for the last day of the month. Returns NULL if the argument is invalid.

```
mysql> SELECT LAST_DAY('2003-02-05');
_____________
_____________
_____________
_____________
             -+                                    +
|            LAST_DAY('2003-02-05')                 |
_____________
_____________
_____________
_____________
             -+                                    +
|                  2003-02-28                       |
_____________
_____________
_____________
_____________
               -+                                  +
```

1 row in set (0.00 sec)

MINUTE(time)

Returns the minute for time, in the range 0 to 59.

```
mysql> SELECT MINUTE('98-02-03 10:05:03');
+-------------------------------------+
| MINUTE('98-02-03 10:05:03') |
+-------------------------------------+
|                                   5 |
+-------------------------------------+
1 row in set (0.00 sec)
```

MONTH(date)

Returns the month for date, in the range 0 to 12.

```
mysql> SELECT MONTH('1998-02-03')
+-----------------------+
| MONTH('1998-02-03') |
+-----------------------+
|                     2 |
+-----------------------+
1 row in set (0.00 sec)
```

MONTHNAME(date)

Returns the full name of the month for date.

```
mysql> SELECT MONTHNAME('1998-02-05');
-----+                                  +
|        MONTHNAME('1998-02-05')        |
-----+                                  +
|        February                       |
-----+                                  +
```

1 row in set (0.00 sec)

NOW()

Returns the current date and time as a value in 'YYYY-MM-DD HH:MM:SS' or YYYYMMDDHHMMSS format, depending on whether the function is used in a string or numeric context. The value is expressed in the current time zone.

```
mysql> SELECT NOW();
-+                                  +
|        NOW()                      |
-+                                  +
```

```
|                    1997-12-15 23:50:26                    |
_____________
_____________
_____________
_____________
-+                                                          +
```

1 row in set (0.00 sec)

GROUP BY: This query is used to group to all the records in a relation together for each and every value of a specific key(s) and then display them for a selected set of fields the relation.

Syntax: SELECT <set of fields> FROM <relation_name>

GROUP BY <field_name>;

Example:SELECT EMPNO, SUM (SALARY) FROM EMP GROUP BY EMPNO;

GROUP BY-HAVING : The HAVING clause was added to SQL because the WHERE keyword could not be used with aggregate functions. The HAVING clause must follow the GROUP BY clause in a query and must also precede the ORDER BY clause if used.

Syntax: SELECT column_name, aggregate_function(column_name) FROM table_name WHERE column_name operator value

GROUP BY column_name

HAVING aggregate_function(column_name) operator value;

Example : SELECT empno,SUM(SALARY) FROM emp,dept

WHERE emp.deptno =20 GROUP BY empno;

ORDER BY: This query is used to display a selected set of fields from a relation in an ordered manner base on some field.

Syntax: SELECT <set of fields> FROM <relation_name>

ORDER BY <field_name>;

Example: SQL> SELECT empno, ename, job FROM emp ORDER BY job;

LAB PRACTICE ASSIGNMENT:

Consider the following table structure for this assignment:

CUSTOMER(Cust_id, C_name, City)
BRANCH(Branch_id, bname, City)
DEPOSIT(Acc_no , Cust_id, Amount, Branch_id, Open_date)
BORROW(Loan_no, Cust_id, Branch_id, Amount)

Perform the following queries on the above table:

1) List total loan.
2) List total deposit.
3) List maximum deposit of customers living in Mumbai.
4) Count total number of branch cities.
5) List branch_id and branch wise deposit.
6) List the branches having sum of deposit more than 4000.
7) List the names of customers having minimum deposit.
8) Count the number of depositors living in **'**nagpur**'**.
9) Find the maximum deposit of the Akurdi branch.
10) Find out number of customers living in Pune.
11) Find out the customers who are not living in Pune or Mumbai.
12) List out Cust_id and C_name in descending order of their C_name.
13) Display the number of depositors in branch wise.
14) Find out the branch which has not borrowers.
15) How many customers have opened deposit after **'**01-01-2016**'**

Conclusion:-

In this assignment, we have learned and executed Aggregate and date functions of MYSQL.

Output:

1) mysql> select * from borrow;

```
+---------+---------+-------------+----------+
| Loan_no | Cust_id | Branch_id | Amount |
+---------+---------+-------------+----------+
| 1       | 1       | 103         | 5000 |
| 2       | 5       | 240         | 4670 |
| 3       | 3       | 652         | 6543 |
| 4       | 7       | 753         | 1654 |
| 5       | 12      | 043         | 1645 |
| 6       | 16      | 128         | 5749 |
| 7       | 18      | 756         | 8979 |
```

```
+---------+----------+----------+----------+
7 rows in set (0.00 sec)

mysql> select sum(amount) from borrow;
+-------------+
| sum(amount) |
+-------------+
|       34240 |
+-------------+
1 row in set (0.00 sec)

2) mysql> select * from deposit;
+-------+---------+--------+-----------+------------+
| Acc_no | Cust_id | Amount | Branch_id | Open_date |
+-------+---------+--------+-----------+------------+
| 4886  | 16      |  75000 | 128       | 2019-04-08 |
| 540   | 12      |  4500 | 103        | 2020-11-04 |
+-------+---------+--------+-----------+------------+
2 rows in set (0.00 sec)

mysql>
mysql> select sum(amount) from deposit;
+-------------+
| sum(amount) |
+-------------+
|       79500 |
+-------------+
1 row in set (0.00 sec)

4) mysql> select count(branch_id) from deposit;
+-----------------+
| count(branch_id) |
+-----------------+
|               2 |
+-----------------+
1 row in set (0.00 sec)

mysql> select count(branch_id) from borrow;
```

```
+ ------------------+
| count(branch_id) |
+ ------------------+
|                7 |
+ ------------------+
1 row in set (0.00 sec)

5) mysql> select * from customer;
+---------  +--------+-------------+
 | Cust_id | C_name | city     |
+---------  +--------+-------------+
| 1       | Akash  | Pune      |
| 12      | Gaurav | Nagpur    |
| 16      | Arya   | Mumbai    |
| 18      | Nupur  | Kolkata   |
| 2       | Raj    | Delhi     |
| 3       | Diksha | Goa       |
| 5       | Ruhi   | Chennai   |
| 7       | Rahul  | Bangalore |
+---------  +--------+------------  +
8 rows in set (0.00 sec)

mysql> select count(city) from customer;
+-------------+
| count(city) |
+-------------          +
|             8 |
+-------------          +
1 row in set (0.00 sec)

8) mysql> select count(city) from customer where city='Nagpur';
+--------------+
| count(city) |
+--------------+
|          1 |
+--------------+
1 row in set (0.00 sec)
```

12) mysql> select Cust_id, C_name from customer order by C_name desc;

```
+-------+-----------+
|Cust_id|C_name|
+-------+-----------+
| 5     | Ruhi      |
| 2     | Raj  |
| 7     | Rahul     |
| 18    | Nupur   |
| 12    | Gaurav |
| 3     | Diksha |
| 16    | Arya      |
| 1     | Akash     |
+-------+-----------+
```

8 rows in set (0.00 sec)

11) mysql> select * from customer where city != 'Pune' or 'Mumbai';

```
+--------+-------+-------------+
 | Cust_id | C_name | city     |
+--------+-------+-------------+
| 12     | Gaurav | Nagpur    |
| 16     | Arya   | Mumbai    |
| 18     | Nupur | Kolkata |
| 2      | Raj    | Delhi  |
| 3      | Diksha | Goa       |
| 5      | Ruhi   | Chennai  |
| 7      | Rahul | Bangalore |
+--------  +-------- +-----------   +
```

7 rows in set, 1 warning (0.00 sec)

10) mysql> select count(C_name) from customer where
city='Pune'; + ----------+

```
| count(C_name) |
+---------------+
|             1 |
+---------------+
```

1 row in set (0.00 sec)

15) mysql> select * from deposit where Open_date > 2016-01-01;

```
+--------+---------+---------+-----------+-------------+
| Acc_no | Cust_id | Amount | Branch_id | Open_date |
+--------+---------+---------+-----------+-------------+
| 4886   | 16      | 75000 | 128       | 2019-04-08 |
| 540    | 12      | 4500 | 103       | 2020-11-04 |
+--------+---------+---------+-----------+-------------+
2 rows in set, 1 warning (0.00 sec)
```