**SQL Commands and Functions**

**To Create/Open Database:**

**mysql> CREATE DATABASE <name of database>;**

Now the database with the given name will be created. One must be connected to the database before using it, as below:

**mysql> use <name of database>;**

**Creating Tables**

Tables are defined with the CREATE TABLE command. When tables are created its columns are named, data types and sizes supplied for each column. At least one column must be specified.

Syntax:

**mysql> CREATE TABLE <TableName>**

**(<ColumnName1> <Data Type1>,**

**<ColumnName2> <Data Type2>,**

**…..….,**

**<ColumnNameN> <Data Type N>);**

Example:

mysql> CREATE TABLE Students

( RollNo DECIMAL(3),

Name VARCHAR(25)

);

**Following constraints can be defined on a table in SQL:**

**Constraints name Description**

PRIMARY KEY Used to create a primary key

UNIQUE to create a unique key

NOT NULL to define that column will not accept null values.

FOREIGN KEY/ REFERENCES to define referential integrity with another table.

DEFAULT to define the columns default value.

CHECK to define the custom rule.

Not Null and Default constraints can be applied only at column level rest all constraints can be applied on both column level and table levels.

**Use of constraints**

mysql> CREATE TABLE student

(Srollno integer PRIMARY KEY NOT NULL,

Sclass integer DEFAULT 12 CHECK (Sclass>0),

Sname varchar(30)

);

mysql> CREATE TABLE teacher

(Tid integer NOT NULL, FOREIGN KEY (Studentid ) REFRENCES student(Sid));

**Inserting the record in existing table**

The INSERT INTO command appends a new record to an existing table and initializes it to desired values.

**Syntax:**

**mysql> INSERT INTO table\_name (column\_name [,column\_name])**

**VALUES (value [,value]);**

mysql> INSERT INTO Student (RollNo, Name) VALUES (12333,’Mohan’);

**Inserting valid Date Values in the table:**

mysql> INSERT INTO myTable (idate) VALUES (19970505);

mysql> INSERT INTO myTable (idate) VALUES ('97-05-05');

mysql> INSERT INTO myTable (idate) VALUES ('1997.05.05');

mysql> INSERT INTO my\_table (idate) VALUES ('0000-00-00');

**Deleting Existing records from the table** :

The DELETE command deletes one, many, or even all records in a table, depending on the conditions that you specify.

Syntax:

**mysql> DELETE FROM tablename WHERE search\_conditions;**

for example

**mysql> DELETE FROM Students WHERE RollNo>11255;**

**Note:** The delete command is VERY dangerous. If run without conditions, it will delete ALL records in a table. In addition, SQL has no undo function. For instance,

mysql> DELETE FROM Students;

Will delete all records from Students table. This is not likely to be what you want.

**Modifying the contents of records :**

The UPDATE command changes one, many, or even all records in a table, depending on the conditions that you specify

Syntax:

**mysql> UPDATE tablename**

**SET column\_name = expression [,column\_name = expression..]**

**[WHERE search\_conditions];**

for example(assuming a customer table)

mysql> UPDATE customer

SET f\_name = ‘Thomas’

WHERE l\_name = ‘Smith’ and date\_of\_birth = ‘3/2/1985’;

[An expression can be either a constant value (e.g., ‘Thomas’) or an operation done on another column or columns (see the example below, assuming a loan table with column rate.]

mysql> UPDATE TABLE loan

SET rate = rate + 1.5;

Because there is no condition (i.e., no WHERE ) all records will be updated. All rates will be increased by 1.5.

**Possible Search Conditions**

**DISTINCT**

Returns unique data from any column.

mysql> SELECT DISTINCT city from stud;

It will display unique city names from the table stud.

**Temporarily renaming columns in query results (ALIAS)**

mysql> SELECT column\_heading AS column\_name

[,column\_heading AS column\_name] FROM table\_name;

Example:

mysql> SELECT f\_name AS “Name”

FROM customer;

**Comparison operators (=,<,>,!=.<>,<=,>=)**

mysql> SELECT \* FROM loan WHERE principal > 100000000;

**Ranges (between and not between; inclusive)**

mysql> SELECT \* FROM loan WHERE rate BETWEEN 7.5 AND 8.5;

OR

mysql> SELECT \* FROM loan WHERE rate NOT BETWEEN 7.5 AND 8.5;

**Lists (in and not in)**

mysql> SELECT \* FROM Customer

WHERE city IN (‘Ahmedabad’, ‘Baroda’, ‘Delhi’,’Mumbai’,’Chennai’);

OR

mysql> SELECT \* FROM Customer

WHERE city NOT IN (‘Ahmedabad’, ‘Baroda’, ‘Delhi’,’Mumbai’,’Chennai’);

**Null values**

mysql> SELECT \* FROM Customer WHERE city IS NULL;

OR

mysql> SELECT \* FROM Customer WHERE city IS NOT NULL;

**Character matches (LIKE and NOT LIKE)**

mysql> SELECT f\_name, l\_name FROM customer WHERE l\_name LIKE ‘Fos%’;

mysql> SELECT f\_name, l\_name FROM customer WHERE l\_name LIKE ‘\_oster’;

Note: “%” (matches any string of zero or more characters) and “\_” (matches any one character). In addition to those, brackets can be used to include either ranges or sets of characters.

**Combinations of previous options using logical operators and, or, and not etc.:**

mysql> SELECT f\_name, l\_name FROM customer

WHERE l\_name LIKE ‘San%’ AND City NOT IN (‘Baroda’,‘Delhi’)

**Some more examples:**

‘Am%’ matches any string starting with Am.

‘%Singh%’ matches any string containing ‘Singh’

‘%a’ matches any string ending with ‘a’

‘\_ \_ \_’ matches any string that is exactly 3 characters long.

‘\_ \_ %’ matches any string that has at least 2 characters long.

‘\_ \_ \_g’ matches any string that is 4 characters along with 3 characters in the beginning but ‘g’ as the 4th character.

**Sorting records**

The output of a SELECT query can be sorted in ascending or descending order on one or more columns, the default is ascending. This is important to note that the data in table is not sorted, only the results that appear on the screen are sorted.

Syntax:

**mysql> SELECT <column name> [,<column name>, ….]**

**FROM <table name> [WHERE <condition>]**

**[ORDER BY <column name> [, <column name>…] ASC/DESC];**

Example: (Sorting on single column)

mysql> SELECT \* FROM EMPL ORDER BY ENAME;

Example : (Sorting on Multiple columns)

mysql> SELECT \* FROM EMPL ORDER BY ENAME DESC, JOB;

To display data in descending order, DESC keyword is used in ORDER BY clause. However it is not necessary to specify ASC for ascending order as it is the default order.

**ALTER TABLE command**

The ALTER TABLE command is used to change definitions of existing tables . It can add columns, delete columns or change their size.

**Adding a column:**

**Syntax:**

**mysql> ALTER TABLE <table name>**

**ADD (<column name> <data type with size> <constraints>);**

Example:

mysql> ALTER TABLE Students

ADD ( age NUMBER (2) CHECK (age > 5));

**Modify a column :**

**Syntax :**

**mysql> ALTER TABLE <table name>**

**MODIFY ( column name newdatatype (newsize));**

Example:

mysql> ALTER TABLE Students

MODIFY ( age NUMBER (1));

**Changing a column name:**

**mysql> ALTER TABLE <table name>**

**CHANGE <old\_column\_name> <new\_column\_name> <column definition> ;**

Example:

mysql> ALTER TABLE Students

CHANGE age s\_age NUMBER (2)

**Removing table components**

To remove primary key constraints

mysql> ALTER TABLE Students

DROP primary key;

To remove column from the table

mysql> ALTER TABLE Students

DROP COLUMN age ;

**Drop a table from database:**

mysql> DROP TABLE <table name> ;

Example:

mysql> DROP TABLE Students;

**FUNCTIONS in MySQL**

**Numeric Functions**

**POWER() :**

Returns the argument raised to the specified power. POW () works the same way.

Example:

(i)POW(2,4); :Result:16

(ii)POW(2,-2); :Result:0.25

(iii)POW(-2,3); :Result: -8

**ROUND() :**

ROUND(X) Rounds the argument to the zero decimal place, Where as ROUND(X,d) rounds the argument to d decimal places.

Example :

(i) ROUND(-1.23); Result: -1

(ii) ROUND(-1.58); Result: -2

(iii) ROUND(1.58); Result: 2

(iv) ROUND(3.798, 1); Result: 3.8

(v) ROUND(1.298, 0); Result: 1

(vi) ROUND(23.298, -1); Result: 20

(vii) ROUND( 25.298,-1); Result: 30

**TRUNCATE() :**

Truncates the argument to specified number of decimal places.

Example:

(i) TRUNCATE (7.29,1); Result: 7.2

(ii) TRUNCATE(27.29,-1); Result: 20

**SIGN() :**

Returns sign of a given number.

Example :

(i) SIGN (15); Result : 1 :

(ii) SIGN (-15); Result : -1 :

(iii) SIGN (0); Result : 0.

**SQRT :**

Returns the square root of given number.

Example :

(i) SQRT (25) Result : 5

**Character/String Functions**

**LENGTH() :**

Returns the length of a string in bytes/no.of characters in string.

Example:

LENGTH(‘INFORMATICS’); Result:11

**CHAR() :**

Returns the corresponding ASCII character for each integer passed.

Example :

CHAR(65) ; Result : A

**CONCAT():**

Returns concatenated string i.e. it adds strings.

Example :

CONCAT(‘Informatics’,’ ‘,‘Practices’); Result : ‘Informatics Practices’

**INSTR():**

Returns the index of the first occurrence of substring.

Example :

INSTR(‘Informatics’,’ mat’); Result : 6 (since ‘m’ of ‘mat’ is at 6th place)

**LOWER()/ LCASE():**

Returns the argument after converting it in lowercase.

Example:

LOWER(‘INFORMATICS’); Result : informatics

**UPPER()/ UCASE():**

Returns the argument after converting it in uppercase.

Example:

UCASE(‘informatics’); Result : INFORMATICS

**LEFT() :**

Returns the given number of characters by extracting them from the left side of the given string

Example :

LEFT(‘INFORMATICS’, 3); Result : INF

**RIGHT():**

Returns the given number of characters by extracting them from the right side of the given string

Example :

RIGHT(‘INFORMATICS’,3); Result : ICS

**MID() / SUBSTR() :**

Returns a substring starting from the specified position in a given string.

Example:

MID(‘INFORMATICS’,3,4); Result : FORM

SUBSTR(‘INFORMATICS’,3,4 ); Result : FORM

**LTRIM() :**

Removes all leading spaces.

Example :

LTRIM(' INFORMATICS'); Result: 'INFORMATICS’

**RTRIM():**

Removes all trailing spaces.

Example :

RTRIM('INFORMATICS '); Result: 'INFORMATICS’

**TRIM() :**

Removes all leading and trailing spaces.

Example:

TRIM(' INFORMATICS '); Result: 'INFORMATICS’

**ASCII(str) :**

Returns the ASCII value of the leftmost character of the string str. Returns 0 if str is an empty string. Returns NULL if str is NULL.

Example:

mysql> SELECT ASCII('2'); Result: 50 (ASCII value of character '2')

mysql> SELECT ASCII('dx'); Result: 100 (ASCII value of d)

mysql> SELECT ASCII('A'); Result: 65 (ASCII value of 'A')

**Date/Time Functions**

**CURDATE() : Returns the current date**

Example: CURDATE(); Result: '2012-09-18'

**NOW() : Returns the current date and time**

Example: NOW(); Result : '2010-07-21 13:58:11'

**SYSDATE() : Return the time at which the function executes**

Example: SYSDATE(); Result: '2010-07-21 13:59:23’

**DATE() : Extracts the date part of a date or datetime expression**

Example: DATE('2003-12-31 01:02:03'); Result:: '2003-12-31'

**MONTH() Returns the month from the date passed**

Example: MONTH('2010-07-21'); Result : 7

**YEAR() : Returns the year.**

Example: YEAR('2010-07-21'); Result : 2010

**DAYNAME() : Returns the name of the weekday**

Example: DAYNAME('2010-07-21'); Result : WEDNESDAY

**DAYOFMONTH() : Returns the day of the month (0-31)**

Example: DAYOFMONTH('2010-07-21'); Result: 21

**DAYOFWEEK() : Returns the weekday index of the argument**

Example: DAYOFWEEK('2010-07-21'); Result: 4 (Sunday is counted as 1)

**DAYOFYEAR() : Return the day of the year(1-366)**

Example: DAYOFYEAR('2010-07-21'); Result: 202

**SLEEP(n) : Tells the system to sleep by n Second.**

mysql> SELECT now(), sleep(20), now();

Displays the same date and time without elapsing 20 seconds used by sleep(20).

mysql> SELECT sysdate(), sleep(20), sysdate();

Displays the date and time elapsing 20 seconds used by sleep(20).

**AGGREGATION Functions:**

A single row function works on a single value. SQL also provides us multiple row functions. A multiple row function works on multiple values. These functions are called aggregate functions or group functions. These functions are:

**MAX()** function is used to find the highest value of any column or any expression based on a column. MAX() takes one argument which can be any column name or a valid expression involving a column name. e.g.,

SELECT MAX(cost)FROM shoes;

SELECT MAX(cost+cost\*margin/100)FROM shoes;

**To find the highest selling price of any type of shoe rounded to 2 decimal places.**

SELECT ROUND(MAX(cost+cost\*margin/100),2)AS "Max. SP" FROM shoes;

**MIN()** function is used to find the lowest value of any column or an expression based on a column. MIN() takes one argument which can be any column name or a valid expression involving a column name. e.g.,

**To find the lowest selling price of any type of shoe rounded to 2 decimal places.**

SELECT ROUND(MIN(cost+cost\*margin/100),2)AS "Min. SP" FROM shoes;

**AVG()** function is used to find the average value of any column or an expression based on a column. AVG() takes one argument which can be any column name or a valid expression involving a column name. Here we have a limitation: the argument of AVG() function can be of numeric (int/decimal) type only. Averages of String and Date type data are not defined. E.g.,

**To find the average quantity in stock for the shoes of type Sports.**

SELECT AVG(qty)FROM shoes WHERE type ='Sports';

**SUM()** function is used to find the total value of any column or an expression based on a

column. SUM() also takes one argument which can be any column name or a valid

expression involving a column name. Like AVG(), the argument of SUM() function can be

of numeric (int/decimal) type only. Sums of String and Date type data are not defined.

e.g.,

**To find the the total value (Quanitity x Cost) of Shoes of type 'Office' present in the inventory**

SELECT SUM(cost\*qty)FROM shoes WHERE type ='Office';

**COUNT()** function is used to count the number of values in a column. COUNT() takes one argument which can be any column name, an expression based on a column, or an asterisk(\*). When the argument is a column name or an expression based on a column, COUNT() returns the number of non-NULL values in that column. If the argument is a \*, then COUNT() counts the total number of rows satisfying the condition, if any, in the table.

e.g.,

**To count the different types of shoes that the factory produces**

SELECT COUNT(distinct type)FROM shoes;

**To count the records for which the margin is greater than 2.00**

SELECT COUNT(margin)FROM shoes WHERE margin > 2;

**To count the number of customers in 'A' category**

SELECT COUNT(\*)FROM customers WHERE category ='A';

**GROUP BY Clause:**

In practical applications many times there arises a need to get reports based on some groups of data. These groups are based on some column values. For example, The management of the shoe factory may want to know what is the total quantity of shoes of various types. i.e., what is the total quantity of shoes of type School, Office, and Sports each. The management may also want to know what is the maximum, minimum, and average margin of each type of shoes. It may also be required to find the total number of customers in each category.

There are many such requirements.

SQL provides GROUP BY clause to handle all such requirements. For the above three situations, the statements with GROUP BY clause are given below:

In the first situation we want MySQL to divide all the records of shoes table into different groups based on their type (GROUP BY type) and for each group it should display the type and the corresponding total quantity (SELECT type, SUM(qty)). So the complete statement to do this is:

**SELECT type, SUM(qty) FROM shoes GROUP BY type;**

**SELECT type, MIN(margin), MAX(margin), AVG(margin)**

**FROM shoes GROUP BY type;**

**SELECT category, COUNT(\*) FROM customers GROUP BY category;**

Sometimes we do not want to see the whole output produced by a statement with GROUP BY clause. We want to see the output only for those groups which satisfy some condition. It means we want to put some condition on individual groups (and not on individual records). A condition on groups is applied by HAVING clause. As an example reconsider the statement G1 discussed above. The statement produced three records in the output – one for each group. Suppose, we are interested in viewing only those groups' output for which the total quantity is more than 1500 (SUM(Qty) > 1500). As this condition is applicable to groups and not to individual rows, we use HAVING clause as shown below:

**SELECT type, SUM(qty) FROM shoes**

**GROUP BY type HAVING SUM(qty) > 1500;**

****

**SELECT type, SUM(qty) FROM shoes**

**GROUP BY type HAVING AVG(margin) >2;**

**WHERE & HAVING Clause:**

WHERE and HAVING both filters out records based on one or more conditions. The difference is WHERE clause can only be applied on a static non-aggregated column whereas we will need to use HAVING for aggregated columns.

1. Apart from SELECT queries, we can use WHERE clause with UPDATE and DELETE clause but HAVING clause can only be used with SELECT query.

**UPDATE department SET deptName="NewSales" WHERE deptID=1 ;  // works fine**

**UPDATE department SET deptName="NewSales" HAVING deptID=1 ; ; // error**

1. WHERE clause is used for filtering rows and it applies on each and every row, while HAVING clause is used to filter groups in SQL.

**SELECT \* FROM stud GROUP BY regno WHRER sub1>AVG(sub1); // error**

**SELECT \* FROM stud GROUP BY regno HAVING sub1>AVG(sub1); // works fine**

1. When WHERE and HAVING clause are used together in a SELECT query with aggregate function,  WHERE clause is applied first on individual rows and only rows which pass the condition is included for creating groups. Once group is created, HAVING clause is used to filter groups based upon condition specified.

**SELECT \* FROM stud WHERE sub2>50 GROUP BY regno HAVING sub1>AVG(sub1); // works fine**

Here it will operate the Query with WHERE clause first with GROUP BY clause then HAVING clause will filter groups based upon the condition specified with HAVING.

**JOIN : Displaying Data from Multiple Tables**

We have faced so far that the data was extracted from a single table. There was no need to refer to more than one table in the same statement. But many times, in real applications of databases, it is required to produce reports which need data from more than one table. Extracting data from more than one table is called one or other JOIN. Joining two of more table can be done in different way in SQL:

**Equi Join:**

In the Equi Join, values in the columns of both the tables are compared for equality first then the result of both the tables will be displayed. In the output all columns of both the tables will be displayed even if they are identical.

**SELECT \* FROM empl, dept WHERE empl.deptno = dept.deptno;**

**Non-Equi Join:**

In the non-Equi Join, values in the columns of both the tables are compared without equality check between the columns of both the tables. Comparison can be done using other relationship between the tables to Join and get the result. In the output all columns of both the tables will be displayed even if they are identical.

**SELECT \* FROM empl, dept WHERE salary>12500 AND JOB = ‘Analyst’;**

**Natural Join:**

The result of Equi and non-Equi Join includes two identical columns, where as in the Natural Join eliminates one of the two identical columns from the result.

**SELECT \* FROM empl NATURAL JOIN dept WHERE empl.deptno = dept.deptno;**

Note: In NATURAL JOIN it is not necessary to define the join condition to eliminate the identical fields in the tables.

**Cross Join:**

The Cross Join is the output as Cartesian Product of two tables. It simply matches the all tuples of one table to every tuples of other table. It does not apply any condition with the tables by default but we can use the WHERE clause to restrict the result.

**SELECT \* FROM empl CROSS JOIN dept WHERE empl.deptno = dept.deptno;**

**Left Join:**

The LEFT JOIN keyword returns all rows from the left table (table1), with the matching rows in the right table (table2). The result is NULL in the right side when there is no match.

SELECT *column\_name(s)*  
FROM *table1*  
LEFT JOIN *table2*  
ON *table1.column\_name*=*table2.column\_name*;

SELECT Customers.CustomerName, Orders.OrderID  
FROM Customers  
LEFT JOIN Orders  
ON Customers.CustomerID=Orders.CustomerID;

**Right Join:**

The RIGHT JOIN keyword returns all rows from the right table (table2), with the matching rows in the left table (table1). The result is NULL in the left side when there is no match.

SELECT *column\_name(s)*  
FROM *table1*  
RIGHT JOIN *table2*  
ON *table1.column\_name*=*table2.column\_name*;

SELECT Orders.OrderID, Employees.FirstName  
FROM Orders  
RIGHT JOIN Employees  
ON Orders.EmployeeID=Employees.EmployeeID

**Inner Join**

Inner join is the most common type of Join which is used to combine the rows from two tables and create a result set containing only such records that are present in both the tables based on the joining condition (predicate).

If none of the record matches between two tables, then INNER JOIN will return a NULL set. Below is an example of INNER JOIN and the resulting set.

SELECT dept.name DEPARTMENT, emp.name EMPLOYEE

FROM DEPT dept, EMPLOYEE emp

WHERE emp.dept\_id = dept.id;

**The SQL UNION Operator**

The UNION operator is used to combine the result-set of two or more SELECT statements. Notice that each SELECT statement within the UNION must have the same number of columns. The columns must also have similar data types. Also, the columns in each SELECT statement must be in the same order.

### SQL UNION Syntax

**SELECT *column\_name(s)* FROM *table1* UNION SELECT *column\_name(s)* FROM *table2*;**

## SQL UNION Example

The following SQL statement selects all the **different** cities (only distinct values) from the "Customers" and the "Suppliers" tables:

**SELECT City FROM Customers UNION SELECT City FROM Suppliers ORDER BY City;**

Note: UNION cannot be used to list ALL cities from the two tables. If several customers and suppliers share the same city, each city will only be listed once. UNION selects only distinct values.

**SQL Based Short Answer Questions (1-2 Marks)**

**1. What do you mean by a Database Management System?**

Ans- Database Management is a collection of programs and files that allow a user to define structure of a database, store data into it, modify the structure and manipulate the data.

**2. What do you mean by Relational database?**

Ans-Relational Database is a type of database in which relation is used as its basic element.Row and columns are used to store data.

**3. What is a foreign key?**

Ans If a key is available in a table as a primary key then this key is called foreign key in another table.

**4. What is primary key?**

Ans-Primary key is a unique key in a relation which can uniquely identifies a tuple (row) in a given relation.

**5. What is SQL?**

Ans-SQL is stands for structured query language. This language is used to manipulate data stored in a table.

**6. What is referential integrity?**

Ans-This is a rule which ensures that in DBMS relationships between records in related tables are valid. And that user don’t accidently delete or change related data.

**7. What is MySql?**

Ans-Mysql is an open source RDBMS which uses SQL.

**8. What is DDL?**

Ans- DDL provides commands to define or redefine the schema of a table. Table is created, altered and dropped using DDL.

**9. What are DML commands?**

Ans- DML commands are used to manipulate data stored in a table.Insertion, deletion and modifications are possible using DML commands.

**10. Maximum how many characters can be stored in a (i) text literals (ii) numeric literal**

Ans-(i) Text literals can have 4000 bytes (ii) A numeric literals can store 53 digits.

**11. What is null value in MySql?**

Ans-If a column in a row has no value, then column is said to be null.

**12. Which keyword eliminates redundant data in from a query result?**

Ans- DISTINCT

**13. How would you display system date as the result of a query?**

Ans- CURDATE()

**14. What is NOW() function in MySql?**

Ans- It returns the current date and time.

**15. What is NOT NULL constraint?**

Ans- NOT NULL constraints impose a condition that value of a row cannot be left blank.

**16. What is error in following statement?**

UPDATE EMPL;

Ans- WHERE clause is missing in given query.

**17. Identify the error?**

DELETE ALL FROM TABLE EMP;

Ans-There is no need to write ALL and TABLE word in above query.

Correct form is-DELETE FROM EMP;

**18. Differentiate WHERE and HAVING clause?**

Ans-:- Where clause is used to select particular rows that satisfy condition whereas having clause is used in connection with aggregate function, group by clause.

**19. How SQL commands are classified?**

Ans-SQL Commands are classified into three categories

(i) **Data Definition Language (DDL)**-Commands that allow us to perform tasks related to data definition. E.g. creating, altering and dropping

(ii) **Data Manipulation Language (DML)** - Commands that allows us to perform data manipulation e.g retrieval, insertion, and modification of data stored in a database.

(iii) **Transaction Control Language (TCL)** - Commands that allow us to manage and control the transactions.

**20. What is difference between char and varchar?**

Ans-The difference between char and varchar is that of fixed length and variable length.The CHAR datatypes specifies a fixed length character string.When a column is given datatype as CHAR(n) then MySQL ensures that all values stored in that column have this length.But on other hand when a column is given datatype as VARCHAR(n) ,then the maximum size of a value in this column stores exactly what we specify.

**21. What do you understand by the terms primary key and degree of a relation in relational data base?**

Ans: Primary Key: A primary key is a set of one or more attributes that can uniquely identify tuples within the relations. The number of attributes in a relation is called Degree of a relation in relational data base.

**22. What do you understand by the candidate key and cardinality of a relation in relational data base?**

Candidate Key: All attribute combinations inside a relation that can serve as primary key (uniquely identifies a row in a relation) are Candidate Keys as they are candidates for the primary key position. The number of rows in a relation is known as cardinality of a relation.

**23. Why is it not allowed to give String and Date type arguments for SUM() and AVG() functions? Can we give these types of arguments for other functions?**

Answer: String and dates are not real numbers that we calculate so sum() or avg() functions are not valid for them.

**24. What is default, Autocommit mode in MySQL ?**

Answer : By default, Autocommit mode is on in MySQL.

**25. Can where be added a savepoint in a transaction ?**

Answer : We can add a savepoint anywhere in a transaction.

**26. How are NULL values treated by aggregate functions?**

Answer: None of the aggregate functions takes NULL into consideration. NULL is simply

ignored by all the aggregate functions.

**27. There is a column C1 in a table T1. The following two statements:**

**SELECT COUNT(\*) FROM T1; and**

**SELECT COUNT(C1) from T1; are giving different outputs. What may be the possible reason?**

Answer: There may be a null value.

**28. What is the purpose of GROUP BY clause?**

Answer: GROUP BY: GROUP BY clause is used in a SELECT statement in conjunction with aggregate functions to group the result based on distinct values in a column.

**29. What is the difference between HAVING and WHERE clauses? Explain with the help of an example.**

Answer: WHERE Vs HAVING: WHERE is used to put a condition on individual row of a table whereas HAVING is used to put condition on individual group formed by GROUP BY clause in a SELECT statement.

**30. What is a Foreign key? What is its importance?**

Answer: Foreign Key: It is a column of a table which is the primary key of another table in the same database. It is used to enforce referential integrity of the data.

**31. What are constraints? Are constraints useful or are they hindrances to effective management of databases?**

Answer: These are the rules which are applied on the columns of tables to ensure data integrity and consistency. These play very important role for tables so are not hindrances.

**32. In a database there is a table Cabinet. The data entry operator is not able to put NULL in a column of Cabinet? What may be the possible reason(s)?**

Answer: Not NULL or Primary key constraints used.

**33. In a database there is a table Cabinet. The data entry operator is not able to put**

**duplicate values in a column of Cabinet? What may be the possible reason(s)?**

Answer : Primary key constraint used.

**34. Do Primary Key column(s) of a table accept NULL values?**

Answer: No.

**35. There is a table T1 with combination of columns C1, C2, and C3 as its primary key? Is it possible to enter:**

a. NULL values in any of these columns?

b. Duplicate values in any of these columns?

Answer: No.

**36. What are the differences between DELETE and DROP commands of SQL?**

Answer: Delete is used for row removing while drop is used for removing complete table.

**37. What are Aggregate Functions?**

Answer: A multiple row function works on multiple values. These functions are called aggregate functions or group functions.

**38. For what Data Types aggregate functions : MIN(), MAX(), and COUNT() work?**

Answer : on any type of values - Numeric, Date, or String. AVG(), and SUM() work on only Numeric values (INT and DECIMAL).

**39. What is HAVING clause ?**

Answer : HAVING clause is used in conjunction with GROUP BY clause in a SELECT statement to put condition on groups.

**40. What is Referential Integrity ?**

Answer : The property of a relational database which ensures that no entry in a foreign key column of a table can be made unless it matches a primary key value in the corresponding column of the related table.

**41. What is Union used for ?**

Answer : Union is an operation of combining the output of two SELECT statements.

**42. What is ALTER TABLE ?**

Answer : ALTER TABLE command can be used to Add, Remove, and Modify columns of a table. It can also be used to Add and Remove constraints.

**43. What is DROP TABLE ?**

Answer : DROP TABLE command is used to delete tables.

**44. What function is used whenever a condition involves an aggregate function ?**

Answer : whenever a condition involves an aggregate function, then we use HAVING clause in

conjunction with GROUP BY clause.

**45. What is Difference between GROUP BY’ and Having functions ?**

Answer : WHERE function is used for individual records and HAVING for groups . GROUP BY function is used for getting results based on some groups of data while a condition on groups is applied by HAVING clause.

**46. Why are aggregate functions called so? Name some aggregate functions.**

Answer : A multiple row function works on multiple values. These functions are called aggregate functions or group functions. Some of the most frequently used. Aggregate functions in MySQL are

: MIN(), MAX(), AVG(), SUM(), COUNT().

**47. What is ALTER TABLE command? Wr i t e all the commands that can be applied using alter table.**

Answer: a new column can be added to a table using ALTER TABLE command. ALTER TABLE can be

used:

• to add a constraint

• to remove a constraint

• to remove a column from a table

• to modify a table column

**48. What is the Cartesian product of two table? Is it same as an Equi-join?**

Answer : Cartesian Product (or Cross Join): Cartesian product of two tables is a table obtained by pairing each row of one table with each row of the other. A cartesian product of two tables contains all the columns of both the tables.

Equi-Join: An equi join of two tables is obtained by putting an equality condition on the Cartesian product of two tables. This equality condition is put on the common column of the tables. This common column is, generally, primary key of one table and foreign key of the other.

**49. Does aggregate Functions consider Null values. Does NULLs play any role in actual calculations?**

Answer : None of the aggregate functions takes NULL into consideration. NULL is simply ignored by all the aggregate functions. For example, the statement:

**SELECT COUNT(\*) FROM shoes;**

+--------------+

| COUNT(\*) |

+--------------+

| 13 |

+--------------+

Indicating that there are 13 records in the Shoes table.

Whereas the query:

**SELECT COUNT(margin) FROM shoes; produces the output:**

+----------------------+

| COUNT(margin) |

+----------------------+

| 10 |

+----------------------+

This output indicates that there are 10 values in the margin column of Shoes table. This means there are 3 (13-10) NULLs in the margin column. This feature of aggregate functions ensures that NULLs don't play any role in actual calculations.

**SOLVED QUESTIONS from Transaction Control Language (SQL):**

**1. Define a transaction.**

Ans. -A transaction is a logical unit of a work that must succeed or fail in its entirely. It is an

atomic operation which can be divided unto smaller operations.

**2. What are the two ways in which multiple transactions can be executed?**

Ans. - Multiple transactions can be executed in one of the following two ways:

(i) Serially (ii) Concurrently

**3. What is a savepoint?**

Ans. - Savepoints are special operations that allow you to divide the work of a transaction into different segments. In case of a failure, you can execute rollbacks to the savepoint only, leaving prior changes intact.

**4. What to you understand by a database transaction?**

Ans. - A database transaction is a logical unit of work that must succeed or fail in its entirely.

**5. Why do understand by transaction COMMIT and ROLLBACK?**

Ans- COMMITing a transaction means all the steps of a transaction are carried out successfully and all data changes are made permanent in the database. Transaction ROLLBACK means transaction has not been finished completely and hence all data changes made by the transaction in the database if any, are undone and the database returns to the state as it was before this transaction execution started.

**6. What do you understand by ACID properties of database transaction?**

Ans. -To ensure the data-integrity, the database system maintains the following properties of transaction. The properties given below are termed as ACID properties-an acronym derived from the first letter of each of the properties.

(i) **Atomicity** - This property ensures that either all operations of the transactions

are reflected properly in the database, none are. Atomicity ensures either all-ornone

operations of a transaction are carried out.

(ii) **Consistency** - This property ensures that database remains in a consistent state before the start of transaction and after the transaction is over.

(iii) **Isolation** - Isolation ensures that executing transaction execution in isolation i.e.

is unaware of other transactions executing concurrently in the system.

(iv) **Durability** - This property ensures that after the successful completion of a transaction i. e., when a transaction COMMITs, the changes made by it to the database persist i. e., remain in the database irrespective of other failures.

**7. What the function is of redo and undo logs?**

Ans. -Every database has a set of redo log files. It records all change in data including both committed and uncommitted changes. Undo logs stored roll backed data.

**8. What TCL commands are supported by SQL?**

Ans. -SQL supports following TCL commands

• **BEGIN |START TRANSACTION**-Marks the beginning of a transaction

• **COMMIT**-Ends the current transaction by saving database changes and starts a new

transaction.

• **ROLLBACK**-Ends the current transaction by discarding changes and starts a new

transaction.

• **SAVEPOINT**-Defines breakpoints for the transactions to allow partial rollbacks.

• **SET AUTOCOMMIT**-Enables or disable the default autocommit mode.

**9. Which two statements complete a transaction?**

A. DELETE employees;

B. DESCRIBE employees;

C. ROLLBACK TO SAVEPOINT C;

D. GRANT SELECT ON employees TO SCOTT;

E. ALTER TABLE employees

MODIFY COLUMN sal;

F. Select MAX(sal) FROM employees WHERE department\_id=20;

Ans. - C, E