DATA TYPES IN MySQL

Class	*Data Type*	*Description*	*Format*	*Example*
Text	CHAR(size)	A fixed-length string between 1 and 255 characters in length right-padded with spaces to the specified length when stored. Values must be enclosed in single quotes or double quotes.	CHAR(size)	**'COMPUTE R' 'CBSE'**
	VARCHAR(size)	A variable-length string between 1 and 255 characters in length; for example VARCHAR(20).	VARCHAR (size)	**'SCIENCE' 'Informatics'**
NUMERIC	DECIMAL(p,s)	It can represent number with or without the fractional part. The size argument has two parts : *precision* and *scale. Precision (p)* indicates the number of significant digits and *scale* (s)maximum number of digits to the right of the decimal point.	Number(p,s)	**58.63**
	INT	It is used for storing integer values	INT	**164**
Date	**DATE**	**It represents the date including day, month and year between 1000-01-01 and 9999-12-31**	**YYYY-MM-DD**	**2014-08-27**

Difference between CHAR and VARCHAR

The CHAR data-type stores fixed length strings such that strings having length smaller than the field size are padded on the right with spaces before being stored. The VARCHAR on the other hand supports variable length strings and therefore stores strings smaller than the field size without modification.

SQL Constraints/ Integrity Constraints

1-SQL Constraint is a condition or check applicable on a field or set of fields.
2- They can also be defined or modified after creating the tables.
3- When constraints are defined any data entering in the table is first checked to satisfy the condition specified in particular constraint if it is, only then table data set can be updated. If data updation/ insertion is violating the defined constraints, database rejects the data (entire record is rejected).
4- When a constraint is applied to a single column, it is called a column level constraint but if a constraint is applied on a combination of columns it is called a table constraint. 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.

ACCESSING DATABASE IN MYSQL :

Through USE keyword we can start any database Syntax:
USE <database Name>;
Example: USE ADDRESS;

CREATING TABLE IN MYSQL

Through Create table command we can define any table.
CREATE TABLE <tablename>
(<columnname><datatype>[(<Size>)], ...);
CREATE TABLE ADDRESS(SNo integer, City char(25));

INSERTING DATA INTO TABLE

The rows are added to relations using INSERT command.
INSERT INTO <tablename>[<columnname>]
VALUES (<value>, <value>...);
INSERT INTO ADDRESS (SNo, City)
VALUES (100,'JAIPUR');

SELECT COMMAND:

The SELECT command is used to make queries on the database. A query is a command that is given to produce certain specified information from the database table(s). The SELECT command can be used to retrieve a subset of rows or columns from one or more tables. The syntax of Select Command is:
SELECT <Column-list>
FROM <table_name>
[Where <condition>]
[GROUP BY <column_list>]
[Having <condition>]
[ORDER BY <column_list [ASC|DESC]>]

Example:

SELECT * FROM ADDRESS WHERE SNo=100;

Eliminating Redundant Data

- DISTINCT keyword eliminates redundant data

SELECT DISTINCT City FROM ADDRESS;

Selecting from all the rows

SELECT * FROM ADDRESS;

Viewing structure of table:

DESCRIBE/DESC <tablename>;

DESCRIBE ADDRESS;

Using column aliases:

SELECT <column name> AS [columnalias][,...]
FROM <tablename>;
SELECT SNo, City AS "STUDENTCITY"
FROM ADDRESS;

■ Condition based on a range:

Keyword BETWEEN used for making range checks in queries.

SELECT SNo, CITY FROM ADDRESS WHERE SNo BETWEEN 10 AND 20;

■ Condition based on a list:

Keyword IN used for selecting values from a list of values.

SELECT rno, sname FROM student WHERE rno IN (10, 20, 60);

■ Condition based on a pattern matches:

Keyword LIKE used for making character comparison using strings percent(%)
matches any substring underscore(_) matches any character
SELECT SNo, City FROM ADDRESS WHERE City LIKE '%ri';

■ Searching for NULL

The NULL value in a column is searched for in a table using IS NULL in the WHERE clause (Relational Operators like =,<> etc cannot be used with NULL).

For example, to list details of all employees whose departments contain NULL (i.e., novalue), you use the command:

```
SELECT empno, ename
FROM emp
Where Deptno IS NULL;
```

■ ORDER BY clause:

It is used to sort the results of a query.

SELECT <column name> [, <column name>, .]
FROM <table name>
[WHERE <condition>] [ORDER BY <column name>];

SELECT * FROM ADDRESS WHERE SNo>50 ORDER BY City;

■ Creating tables with SQL Constraint :

CREATE TABLE command is used to CREATE tables , the syntax is:

```
CREATE TABLE <Table_name>
  ( column_name 1    data_type1 [(size) column_constraints],
    column_name 1    data_type1 [(size)  column_constraints],
     :
     :
     [<table_constraint> (column_names)] );
```

■ SQL Constraint:

A Constraint is a condition or check applicable on a field or set of fields.

■ NOT NULL/UNIQUE/DEFAULT/CHECK/PRIMARY KEY/FOREIGN KEY Constraint:

```
CREATE TABLE student (rollno integer NOT NULL );

CREATE TABLE student (rollno integer UNIQUE );

CREATE TABLE student (rollno integer NOT NULL, Sclass integer, Sname varchar(30),
Sclass DEFAULT 12 );

CREATE TABLE student (rollno integer CHECK (rollno>0), Sclass integer, Sname
varchar(30));

CREATE TABLE student (rollno integer NOT NULL PRIMARY KEY, Sclass integer,
Sname varchar(30));

CREATE TABLE teacher (Tid integer NOT NULL, FOREIGN KEY (Studentid )
REFRENCES student (Sid));
```

■ Modifying data in tables:

Existing data in tables can be changed with UPDATE command.

The Update command is use to change the value in a table. The syntax of this command is:

```
    UPDATE <table_name>
    SET column_name1=new_value1 [,column_name2=new_value2,……]
    WHERE <condition>;

UPDATE student SET Sclass=12 WHERE Sname='Rohan';
```

■ Deleting data from tables:

The DELETE command removes rows from a table. This removes the entire rows, not individual field values. The syntax of this command is

```
    DELETE FROM <table_name>
    [WHERE <condition>];
```

e.g., to delete the tuples from EMP that have salary less than 2000, the following command is used:

```
    DELETE FROM emp WHERE sal<2000;
```

To delete all tuples from emp table:

DELETE FROM emp;

MySQL functions:

A function is a special type of predefined command set that performs some operation and returns a single value.

Single-row functions return a single result row for every row of a queried table. They are categorized into: Numeric functions, String functions, and Date and Time functions.

1) 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

- **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

2) 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 PRACTICES', 3); Result : INF

- **MID( )/SUBSTR( )** : Returns a substring starting from the specified position in a given string.

Example: MID('INFORMATICS PRACTICES',3,4); Result : FORM

- **LTRIM( ) :** Removes leading spaces.

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

- **RTRIM( ):** Removes trailing spaces.

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

- **TRIM( )** : Removes leading and trailing spaces.

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

3) Date/Time Functions

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

  Example: CURDATE( ); Result:'2014-07-21'

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

  Example: NOW( ); Result: '2014-07-21 13:58:11'

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

  Example: SYSDATE( ); Result:'2014-07-21 13:59:23'

- **DATE( ):** Extracts the date part of a date or date time 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

- **Aggregate or Group functions:** MySQL provides Aggregate or Group functions which work on a number of values of a column/expression and return a single value as the result.

Some of the most frequently used Aggregate functions in MySQL are:

S.No	Name of the Function	Purpose
1	MAX()	**Returns the MAXIMUM of the values under the specified column/expression.**
2	MIN()	**Returns the MINIMUM of the values under the specified column/expression.**
3	AVG()	**Returns the AVERAGE of the values under the specified column/expression.**
4	SUM()	**Returns the SUM of the values under the specified column/expression.**
5	**COUNT()**	**Returns the COUNT of the number of values under the specified column/expression.**

- The **GROUP BY** clause groups the rows in the result by columns that have the same values. Grouping can be done by column name, or with aggregate functions in which case the aggregate produces a value for each group.
- The HAVING clause place conditions on groups in contrast to WHERE clause that place conditions on individual rows. While WHERE condition cannot include aggregate functions, HAVING conditions can do so.

■ ALTER TABLE COMMAND:-

The ALTER Table command is used to change the definition (structure) of existing table. Usually , it can:

- Add columns to a table
- Delete columns
- Modify a column

The syntax of command is:

For Add or modify column:

ALTER TABLE <Table_name> ADD/MODIFY <Column_defnition>;

For Delete column

ALTER TABLE <Table_name> DROP COLUMN <Column_name>;

Example :

To add a new column address in EMP table command will be :

ALTER TABLE EMP ADD (address char (30));

To modify the size of sal column in EMP table, command will be: ALTER

TABLE EMP MODIFY (sal number(9,2));

To delete column Address from Table EMP the command will be: ALTER

TABLE EMP DROP COLUMN address;

• **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.
• **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.
• **Referential Integrity:** 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.
• **Union:** Union is an operation of combining the output of two SELECT statements.

Display data from multiple Tables :-

It does no good to put records in a database unless you retrieve them eventually and do something with them.

Creating Joins on tables :-

If a SELECT statement names multiple tables in the FROM clause with the names separated by commas, MySQL performs a full join. For example, if you join t1 and t2 as follows, each row in t1 is combined with each row in t2:

```
mysql> SELECT t1.*, t2.* FROM t1, t2;
+----+----+----+----+
| i1 | c1 | i2 | c2 |
+----+----+----+----+
| 1  | a  | 2  | c  |
| 2  | b  | 2  | c  |
| 3  | c  | 2  | c  |
| 1  | a  | 3  | b  |
| 2  | b  | 3  | b  |
| 3  | c  | 3  | b  |
| 1  | a  | 4  | a  |
| 2  | b  | 4  | a  |
| 3  | c  | 4  | a  |
+----+----+----+----+
```

A full join is also called a *cross join* because each row of each table is crossed with each row in every other table to produce all possible combinations. This is also known as the *cartesian product*. Joining tables this way has the potential to produce a very large number of rows.

If you add a WHERE clause causing tables to be matched on the values of certain columns, the join becomes what is known as an ***equi-join*** because you're selecting only rows with equal values in the specified columns:

```
mysql> SELECT t1.*, t2.* FROM t1, t2 WHERE t1.i1 = t2.i2;
+----+----+----+----+
| i1 | c1 | i2 | c2 |
+----+----+----+----+
| 2  | b  | 2  | c  |
| 3  | c  | 3  | b  |
```

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

The JOIN and CROSS JOIN join types are equivalent to the ',' (comma) join operator.

Solved Questions :-

Q1. Consider the following tables ACTIVITY and COACH. Write SQL commands for the statements (i) to (iv) and give outputs for SQL queries (v) to (viii).

Table: ACTIVITY

ACode	ActivityName	ParticipantsNum	PrizeMoney	ScheduleDate
1001	Relay 100x4	16	10000	23-Jan-2004
1002	High jump	10	12000	12-Dec-2003
1003	Shot Put	12	8000	14-Feb-2004
1005	Long Jump	12	9000	01-Jan-2004
1008	Discuss Throw	10	15000	19-Mar-2004

Table: COACH

PCode	Name	ACode
1	Ahmad Hussain	1001
2	Ravinder	1008
3	Janila	1001
4	Naaz	1003

(i) To display the name of all activities with their Acodes in descending order.

(ii) To display sum of PrizeMoney for each of the Number of participants groupings (as shown in column ParticipantsNum 10,12,16).

(iii) To display the coach's name and ACodes in ascending order of ACode from the table COACH.

(iv) To display the content of the GAMES table whose ScheduleDate earlier than 01/01/2004 in ascending order of ParticipantNum.

(v) SELECT COUNT(DISTINCT ParticipantsNum) FROM ACTIVITY;

(vi) SELECT MAX(ScheduleDate),MIN(ScheduleDate) FROM ACTIVITY;

(vii) SELECT SUM(PrizeMoney) FROM ACTIVITY;

(viii) SELECT DISTINCT ParticipantNum FROM COACH;

Ans :

I. SELECT ActivityName, ACode FROM ACTIVITY ORDER BY Acode DESC;

II. SELECT SUM(PrizeMoney),ParticipantsNum FROM ACTIVITY GROUP BY ParticipantsNum;

III. SELECT Name, ACode FROM COACH ORDER BY ACode;

IV. SELECT * FROM ACTIVITY WHERE ScheduleDate<'01-Jan-2004' ORDER BY ParticipantsNum;

(v) 3

(vi) 19-Mar-2004 12-Dec-2003

(vii) 54000

(viii) 16
10
12

Q2. Consider the following tables GAMES and PLAYER. Write SQL commands for the statements (i) to (iv) and give outputs for SQL queries (v) to (viii).

Table: GAMES

GCode	GameName	Number	PrizeMoney	ScheduleDate
101	Carom Board	2	5000	23-Jan-2004

102	Badminton	2	12000	12-Dec-2003
103	Table Tennis	4	8000	14-Feb-2004
105	Chess	2	9000	01-Jan-2004
108	Lawn Tennis	4	25000	19-Mar-2004

Table: PLAYER

PCode	Name	Gcode
1	Nabi Ahmad	101
2	Ravi Sahai	108
3	Jatin	101
4	Nazneen	103

(i) To display the name of all Games with their Gcodes.
(ii) To display details of those games which are having PrizeMoney more than 7000.
(iii)To display the content of the GAMES table in ascending order of ScheduleDate.
(iv) To display sum of PrizeMoney for each of the Number of participation groupings (as shown in column Number 2 or 4).
(v) SELECT COUNT(DISTINCT Number) FROM GAMES;
(vi)SELECT MAX(ScheduleDate),MIN(ScheduleDate) FROM GAMES;
(vii) SELECT SUM(PrizeMoney) FROM GAMES;
(viii) SELECT DISTINCT Gcode FROM PLAYER;

Ans : (i) SELECT GameName,Gcode FROM GAMES;

(ii) SELECT * FROM GAMES WHERE PrizeMoney>7000;

(iii) SELECT * FROM GAMES ORDER BY ScheduleDate;

(iv) SELECT SUM(PrizeMoney),Number FROM GAMES GROUP BY Number;

(v)2

(vi) 19-Mar-2004 12-Dec-2003
(vii) 59000
(viii)101
103
108

Q3. Consider the following tables HOSPITAL. Give outputs for SQL queries (i) to (iv) and write SQL commands for the statements (v) to (viii).

No	Name	Age	Department	Dateofadmin	Charge	Sex
1	Arpit	62	Surgery	21/01/06	300	M
2	Zayana	18	ENT	12/12/05	250	F
3	Kareem	68	Orthopedic	19/02/06	450	M
4	Abhilash	26	Surgery	24/11/06	300	M
5	Dhanya	24	ENT	20/10/06	350	F
6	Siju	23	Cardiology	10/10/06	800	M
7	Ankita	16	ENT	13/04/06	100	F
8	Divya	20	Cardiology	10/11/06	500	F
9	Nidhin	25	Orthopedic	12/05/06	700	M
10	Hari	28	Surgery	19/03/06	450	M

(i) Select SUM(Charge) from HOSPITAL where Sex='F';
(ii) Select COUNT(DISTINCT Department) from HOSPITAL;
(iii) Select SUM(Charge) from HOSPITAL group by Department;
(iv) Select Name from HOSPITAL where Sex='F' AND Age > 20;
(v) To show all information about the patients whose names are having four characters only.
(vi) To reduce Rs 200 from the charge of female patients who are in Cardiology department.
(vii) To insert a new row in the above table with the following data :
11, 'Rakesh', 45, 'ENT', {08/08/08}, 1200, 'M'
(viii) To remove the rows from the above table where age of the patient > 60.

Ans : (i) 1200
(ii) 4
(iii) 1050
700
1150
1300
(iv) Dhanya
(v) SELECT * FROM HOSPITAL WHERE NAME LIKE "_____ ";
(vi) UPDATEHOSPITAL SET CHARGE = CHARGE – 200 WHERE (DEPARTMENT = 'CARDIOLOGY' AND SEX = 'F');
(vii) INSERT INTO HOSPITAL VALUES(11,'Rakesh',45,'ENT',{08/08/08}, 1200, 'M');
(viii) DELETE FROM HOSPITAL WHERE AGE > 60;

Q4. Consider the following tables BOOKS. Write SQL commands for the statements (i) to (iv) and give outputs for SQL queries (v) to (viii).

Table : BOOKS

B_Id	Book_Name	Author_Name	Publisher	Price	Type	Quantity
C01	Fast Cook	Lata Kapoor	EPB	355	Cookery	5
F01	The Tears	William Hopkins	First	650	Fiction	20

T01	My C++	Brain & Brooke	FPB	350	Text	10
T02	C++ Brain	A.W.Rossaine	TDH	350	Text	15
F02	Thuderbolts	Anna Roberts	First	750	Fiction	50

i). To list the names from books of Text type.
ii). To display the names and price from books in ascending order of their price.
iii). To increase the price of all books of EPB publishers by 50.
iv). To display the Book_Name, Quantity and Price for all C++ books.
v). Select max(price) from books;
vi). Select count(DISTINCT Publishers) from books where Price >=400;
vii).Select Book_Name, Author_Name from books where Publishers = 'First';
viii).Select min(Price) from books where type = 'Text';

Ans : (i) SELECT Book_Name FROM BOOKS WHERE Type = 'Text';
(ii) SELECT Book_Name, Price FROM BOOKS ORDER BY Price;
(iii) UPDATE BOOKS SET Price = Price + 50 WHERE Publisher = 'EPB';
(iv) SELECT Book_Name, Quantity, Price FROM BOOKS WHERE Book_Name LIKE '%C++%';
(v) 750
(vi) 2
(vii) The Tears William Hopkins
Thuderbolts Anna Roberts
(viii) 350

Q5. Consider the tables ITEMS & COMPANY. Write SQL commands for the statements (i) to (iv) and give the outputs for SQL queries (v) to (viii).

Table : ITEMS

ID	PNAME	PRICE	MDATE	QTY
T001	Soap	12.00	11/03/2007	200
T002	Paste	39.50	23/12/2006	55
T003	Deodorant	125.00	12/06/2007	46
T004	Hair Oil	28.75	25/09/2007	325
T005	Cold Cream	66.00	09/10/2007	144
T006	Tooth Brush	25.00	17/02/2006	455

Table : COMPANY

ID	COMP	City
T001	HLL	Mumbai
T008	Colgate	Delhi
T003	HLL	Mumbai
T004	Paras	Haryana
T009	Ponds	Noida
T006	Wipro	Ahmedabad

i). To display PNAME, PRICE * QTY only for the city Mumbai.

ii). To display product name, company name & price for those items which IDs are equal to the IDs of company.

iii). To delete the items produced before 2007.

iv). To increase the quantity by 20 for soap and paste.

v). SELECT COUNT(*) FROM ITEMS WHERE ITEMS.ID=COMPANY.ID;

vi). SELECT PNAME FROM ITEMS WHERE PRICE=SELECT MIN(PRICE) FROM ITEMS;

vii). SELECT COUNT(*) FROM COMPANY WHERE COMP LIKE "P_____ ";

viii). SELECT PNAME FROM ITEMS WHERE QTY<100;

Ans :

(i) SELECT PNAME, QTY*PRICE FROM ITEMS
WHERE ITEMS.ID = COMPANY.ID AND COMPANY.City='Mumbai';

(ii) SELECT PNAME, COMP, PRICE FROM ITEMS, COMPANY
WHERE ITEMS.ID = COMPANY.ID;

(iii) DELETE FROM ITEMS WHERE MDATE < {01/01/2007};

(iv) UPDATE ITEMS SET QTY = QTY + 20
WHERE PNAME = 'Soap' OR PNAME = 'Paste';

(v)4

(vi) Soap

(vii)2

(viii) Paste
Deodorant

****