Literals : Values, A text literal can have maximum 4000 bytes in MySQL 5.1 and numeric can store a maximum 0f 53 digits of precision.

Identifier : Variables

Data Types :
Numeric

INT	Width 11 digits
TINYINT	Width 4 digits (127 to -128)
SMALLINT	Width 5 digits (32767 to – 32768)
MIDILINT	Width 9 digits
BIGINT	Width 11 digits (Range more then INT)
FLOAT (M,D)	Length is M and decimal is D (decimal can go 24 place) Floating
DOUBLE (M,D)	Decimal can go 53 place, it is synonym of REAL, Floating
DECIMAL (M,D)	Unpacked Floating-point

Date and Time

DATE	Format yyyy-mm-dd, between 1000-01-01 to 9999-12-31
DATETIME	Combination of yyyy-mm-dd and hh:mm:ss
TIMESTAMP	Same as DATETIME but no separator
TIME	Hh:mm:ss
YEAR(M)	Year in 2-digit or 4-digit

String/Text type

CHAR(M)	Fixed length between 1 to 255, defining length is not required, default is 1
VARCHAR(M)	Variable length between 1 to 255, you must define the length
BLOB or TEXT	Maximum is 65535, BLOB (Binary Large Object), can store large amount of binary data e.g. Images. Text also same. Only difference is while sorting in BLOB is case sensitive but TEXT is not. No need to specify the length.
TINYBLOB or TINYTEXT	Maximum is 255, No need to specify the length.
MEDIUMBLOB or MEDIUMTEXT	Maximum is 16777215, No need to specify the length.
LONGBLOB or LONGTEXT	No need to specify the length.
ENUM	Define ENUM as ENUM('A','B','C') then only values can store A, B, C or NULL

Difference between Char and Varchar

CHAR	VARCHAR
1. Fixed length 2. CHAR(n), all values will store n bytes 3. Blank will store 4. defining length is not required, default is 1	1. Variable length 2. VARCHAR(n), all value will store maximum length of value, not always 'n'. 3. No blank will store 4. Must be define the length

1. Null Values: Any arithmetic expression containing a null, always evaluates to null. E.g. add null to 10 = null

2. Comments: Not executable
   1. /* …… */ Multilane comment
   2. (- -) Single line comment
   3. # Single line comment

3. show databases;
Show all database in MySql

4. use emp;   emp database will open

5. show tables;
Show all tables in that database

6. create table : you all know how to create.

7. insert into <table name> [<column list>] values (<value 1>, <value 2> ..);

e.g.
```
create table emp
        (ecode integer,
        ename char(20),
        sex char(1),
        gross decimal );
```

8. insert into emp values (1001,'Ram','M',4567.20);

9. insert into emp (ecode, ename) values (1002,'Mohan');
If you are inserting values in selected columns only

In an INSERT statement only those columns can be omitted that have either default value defined or they allow NULL values.

10. Insert into emp values (1004, NULL, 'F', 123.00);

11. date is inserted in format of YYYY-MM-DD. E.g. ('2010-3-29');

Select Queries :

*1. select * from emp*
where sex = 'M';

We can use AND & OR between two condition.

2. select ecode, sex from emp;
Select particular column from table

3. Eliminating Redundant Data (with keyword DISTINCT)
select DISTINCT(owner) from pet;

4. Selecting from all the Rows – ALL keyword (It is same as select * from)
select all city from suppliers;

5. Viewing Structure of a table :
DESC <table name>;
DESCRIBE <table name>;

6. Performing simple calculations:
*select 3 * 6 * 4;* *Ans : 72*

select curdate(); *Ans : 2011-03-19*

7. Using column Aliases
select ecode, ename as "Employee Name" from emp;

The column alias name having more then one word, should be enclosed in quotation marks.

8. Condition based on a Range

select ecode, ename, pay
from emp
where pay between 2000 and 5000;

9. Condition based on a list

select ecode, ename, city
from emp
where city in ('DELHI', 'JAIPUR');

select ecode, ename, city
from emp
where city not in ('JAIPUR');

10. Condition based on Pattern Matches

- Percent (%) character matches any substring
- Underscore (_) character matches any character

select ecode, ename from emp *where ename like 'AN%';*	Will give all names start from AN
select ecode, ename from emp *where ename like 'AN_ _';*	Will give all names with starting character is AN and after that only 2 characters

11. Searching for NULL

select ecode, ename from emp where ename IS NULL;	With NULL = sign not come

12. Sorting result – Order by

*select * from emp* *where pay > 10000* *order by ename;*	In order of Increasing
*select * from emp* *where pay > 10000* *order by ename desc;*	In order of Decreasing

13. Relational operator

Use AND (&&), OR (||) and NOT(!) for this

SQL FUNCTIONS

String Functions

Function	**Description**	**Example**	**Output**
CHAR()	The character of each integer	*select char (65,66,67,97);*	ABCa
CONCAT()	Return concatenated string	*select concat ('RAJ', 'EEV');*	RAJEEV

LOWER()
LCASE()
	In lowercase	*select lower('RAJEEV');*	rajeev

UPPER()
UCASE()
	In Uppercase	*select upper ('rajeev');*	RAJEEV

SUBSTRING()
SUBSTR()
	Return the substring, If starting value is in (-) e.g. (-2) it work as read from right side 3 char	*select substr ('rajeev',2,3);*
		select substr ('rajeev',-2,3);
			Aje
			eev

MID()	Same as above	---	---

LEFT()	Read no. of character from L.H.S.	*select left('Rajeev',4);*
		*select left('Parashar,2*3);*
			Raje
			Parash

RIGHT()	Read no. of character from R.H.S.	*select right('Rajeev',4);*
		*select right('Parashar',2*3);*
			jeev
			rashar

LENGTH()	Returns the no of char in a string	*select length('Anil Singh');*	10
INSTR()	Return the first occurrence of substring, Its not case sensitive	*select instr('Rajeev Parashar','r');*	1
LTRIM()	Remove leading space		
RTRIM()	Remove trilling space		
TRIM()	Remove leading & trilling space		

Numeric Functions

Function	**Description**	**Example**	**Output**
MOD()	Remainder	select mod(11,4);	3

POWER()
POW()
	Raised to power	select pow(4,3);	64

ROUND()	Round to an integer or decimal place	select round(123.456,2);
		select round(123.456,-2);
			123.46
			100

SIGN()	Returns the sign of given number	select sign(-15);
		select sign(15);
			-1
			1

SQRT()	Returns the non-negative square root of numeric	select sqrt(16);
		select sqrt(-25);
			4
			NULL

TRUNCATE()	Return truncated numeric	select truncate(123.46,1);
		select truncate(123.46,0);
		select truncate(123.46,-1);
		select truncate(129.45,-1);
			123.4
			123
			120
			120

Date and Time Functions

CURDATE()
CURRENT_DATE()
CURERNT_DATE
	Returns the current date	select curdate();	2020-7-23
			-
			-

DATE()	Extracts the date part	select date('2011-3-23 08:12:15')	2011-3-23
MONTH()		select month('2011-3-23')	3
YEAR()	Extracts the year	select year('2011-3-23');	2011
DAYNAME()	Extracts the name of the week	select dayname('2011-3-23')	Wednesday
DAYOFMONTH()	Return the day of month	select dayofmonth('2011-3-23')	23
DAYOFWEEK()	Return the day of week S-1, M-2, T-3, … S-7	select dayofweek('2011-3-23') select dayofweek('2011-3-20') select dayofweek('2011-3-19')	4 1 7
DAYOFYEAR()	Return the day of the yaer (1-365)	select dayofyear('2011-3-23') select dayofyear('2011-1-1') select dayofyear('2011-12-31')	78 1 365
NOW()	Returns the current date and time	select now();	2011-3-23 11:18:38
SYSDATE()	Returns the time at which function executes	select now();	2011-3-23 11:18:38

SQL Constraints :

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

S.No.	Constraints
1	NOT NULL
2	DEFAULT
3	UNIQUE
4	CHECK
5	Primary Key
6	Foreign Key

A constraint applicable to just one column, it is known as **Column constraint**, a constraint that affects multiple rows or columns of a table is known as **table constraint**.
(i) In the line of column – (inline definition) – column level constraint
(ii) In a separate line after all the column definition (out-of-line definition) – table level constraint

Integrity constraints (or constraints) are the rules that a database must comply at all times. Constraints determine what all changes are permissible to a database.

How MySQL maintains data integrity?

MySQL maintains data integrity through the constraints that are defined. After each change in the database, MySQL first tests whether the database change is permissible with relevant integrity constraints. If it dose, then only change else rejects the changes.

1. SQL NOT NULL Constraints

```
create table r1
(code int(4) not null,
name varchar(20),
pay decimal );

insert into r1 values (1,'Rajeev',25000);                1  :   Rajeev  :  25000
insert into r1 (code, name) values (2,'Anil');           2  :   Anil    :  NULL
insert into r1 (name, pay) values ('Mohan',20000);       Error
```

NOT NULL constraint can be only Column level constraint.

2. SQL DEFAULT Constraints

```
create table r1
(code int(4) not null,
name varchar(20),
pay decimal default 10000);

insert into r1 values (1,'Rajeev',25000);                1  :   Rajeev  :  25000
insert into r1 (code, name) values (2,'Anil');           2  :   Anil    :  10000
```

(Because pay default value is 10000)

3. SQL UNIQUE constraints

```
create table r1
(code int(4) unique,
name varchar(20),
pay decimal);

insert into r1 values (1,'Rajeev',25000);                1  :   Rajeev  :  25000
insert into r1 (code, name) values (2,'Anil');           2  :   Anil    :  NULL
insert into r1 values (1,'Mohan',20000);                 Error
```

(Because code is same)

4. SQL CHECK Constraint

```
create table r1
(code int(4) check (code>0),
name varchar(20),
pay decimal);
```

```
insert into r1 values (1,'Rajeev',25000);          1  :  Rajeev  :  25000
insert into r1 values (0,'Anil',2000);             0  :  Anil    :  20000
insert into r1 values (-2,'Mohan',2200);          -2  :  Mohan   :  22000
```

(MySQL ignores CHECK constraint internally)

5. PRIMARY KEY Constraint

Primary key constraint can be defined as column level constraint and table level constraint.
<u>Table 1</u>

```
create table r1
(code int(4) primary key,
name varchar(20),
pay decimal);
```

<u>Table 2</u>

```
create table r1
(code int(4) not null,
name varchar(20),
pay decimal,
primary key (code) );
```

<u>Table 3</u>

```
create table r1
(code int(4) not null,
name varchar(20) not null,
pay decimal,
primary key (code, name) );
```

(Combination of fields Code and Name is the composite primary key)

In simple primary key constraint save vale for column can not repeat. But in composite key the condition is different. E.g.

```
insert into r1 values (1,'Rajeev',25000);          1  :  Rajeev  :  25000
insert into r1 values (2,’Anil’,20000);          2  :  Anil    :  20000
insert into r1 values (3,’Anil’,25000);          3  :  Anil    :  25000
insert into r1 values (1,’Mohan’,21000);         1  :  Mohan  :  21000

insert into r1 values (1,’Rajeev’,24000);        Error 1-Rajeev is Duplicate entry
```

Difference between Primary Key and Unique :

Primary Key	**Unique**
A column defined as Primary key also Unique	A column defined as Unique may or may not be a primary key
Primary key constraint can be only one in a table (Can be composite)	Unique constraint can be defined more then one

6. Foreign Key Constraint

A foreign key is a non-key column of a table (child table) that draws its values from primary key (or unique key) of another table (called parent table).

When two tables are related by a common column (or set of columns), then the related column in the **parent table** should be declared a PRIMARY KEY or UNIQUE key and the related column in the **child table** should have a FOREIGN KEY constraint.

Important for Foreign Key Implementation in MySQL
Storage engine must be InnoDb. To check your engine tyoe the command:

show create table <table name>;

This command will give full structure with engine. To change engine while time of creating table the command like this:

```
create table <table name>
(
        :: column names
) engine = innodb;
```

```
create table rate
  (code integer primary key,
  name varchar(20),
  rate decimal) ENGINE = INNODB;

create table bill
  (code integer,
  qty integer,
  FOREIGN KEY (code) REFERENCES rate(code) ) ENGINE = INNODB;
```

In this above example, last line of table-creation defines a foreign key constraint for **code** that refers to **code** column of **rate** table.

In above example last line we can write in this way also:
FOREIGN KEY (code) REFERENCES rate(code) ON DELETE set null) ENGINE = INNODB;

Foreign Key Specification: ON DELETE <option>

- CASCADE : If delete or update command is used on the parent table, then automatically delete of update the matching rows in the child table.
- SET NULL : Set the foreign key column or columns in the child table to NULL.
- NO ACTION : If there is any related vale in child table then InnoDB rejects the delete or update operation for the parent table.
- RESTRICT : This action rejects the delete or update operation for the parent table.

The following rules apply when a foreign key is specified :

Referenced table – Parent table
Referencing table – Child table

- The parent table must be created. In the later case child table.
- A primary key must be defined for the parent table.
- The data type of the column in the foreign key must be same.

Self-referencing Tables :

The parent and child table associated with a foreign key may be the same. Such a table is called a *self-referencing table*, and this is known as *self-referential integrity*.

```
create table r1
  (code integer(4) primary key,
  name char(10),
  pay decimal,
  i_code integer(4),
  FOREIGN KEY (i_code) REFERENCES r1 (code) ) ENGINE = INNODB;
```

ALTER (Change) TABLE WITH CONSTRAINTS

(i) adding column to a table
(ii) modifying column-definitions of a table
(iii) deleting column of a table
(iv) adding constraints to a table
(v) enable/disable constraints

```
create table r1
   (col1 int(4),
   name char(15) );
```

```
insert into r1 values (1,'Rajeev');
insert into r1 values (2,'Anil');
```

```
select * from r1;
```

Col1	name
1	Rajeev
2	Anil

1. Adding column to a table

```
alter table r1 add col3 int;
select * from r1;
```

Col1	name	col3
1	Rajeev	NULL
2	Anil	NULL

```
alter table r1 add col4 int not null;
select * from r1;
```

Col1	name	col3	col4
1	Rajeev	NULL	0
2	Anil	NULL	0

```
alter table r1 add col5 char(10) not null;
select * from r1;
```

Col1	name	col3	col4	col5
1	Rajeev	NULL	0	
2	Anil	NULL	0	

BLANK

alter table r1 add col6 char(10);
select * from r1;

Col1	name	col3	col4	col5	col6
1	Rajeev	NULL	0		NULL
2	Anil	NULL	0		NULL

NOTE : Adding a new column with NOT NULL constraint for date & time will store 0000-00-00 & 00:00:00 respectively.

2. Modifying column-definitions of a table

It is use to change column name or column length

For this we can use with alter (i) change and (ii) modify. In change clause the name of the column will come twice. With Modify you cannot change the name of a column, but with change you can do so.

e.g.

create table r1
   (col1 int(4),
   name char(15));

insert into r1 values (1,'Rajeev');
insert into r1 values (2,'Anil');

select * from r1;

Col1	name
1	Rajeev
2	Anil

alter table r1 change name name char(20);
(now if you will see structure of r1 table name column have char(20) width)

alter table r1 modify name char(10);
(now if you will see structure of r1 table name column have char(10) width)

alter table r1 change name name1 char(20);

Col1	Name1
1	Rajeev
2	Anil

(Means column name is changed from name to name1 by change clause)

3. Deleting column of a table

```
alter table r1 drop name;
                or
alter table r1 drop column name;
```

(Word COLUMN is optional)

4. Adding constraints to a table

e.g**. PRIMARY KEY**

```
create table r1
   (col1 int(4),
   name char(15) );

insert into r1 values (1,’Rajeev’);
insert into r1 values (2,’Anil’);

select * from r1;
```

col1	name
1	Rajeev
2	Anil

```
alter table r1 add primary key (col1);
```

e.g. FOREIGN KEY

```
create table r1
(
          A integer primary key,
          B integer not null
);

create table r2
(
          A integer primary key,
          B integer not null
);

alter table r1
add foreign key(A) references r2 (A) ;
```

Removing Constraint

```
alter table r1 drop primary key;
alter table r1 drop foreign key;
```

5. Enable/disable constraints

In MySQL, you cannot disable a PRIMARY KEY constraint but you can drop.

To disable foreign keys

```
SET FOREIGN_KEY_CHECKS = 0;
```

To enable foreign keys

```
SET FOREIGN_KEY_CHECKS = 0;
```

DROP TABLE will delete table

```
drop table item;
```

```
drop table if exists item;
```

DELETEING DATA from table

```
delete from r1 where code = 2;
```

MODIFY DATA in a table

```
update r1 set code = 3
where code = 2;
```

DATABSE TRANSACTIONS

A transaction is a logical unit of work (LUW) that must succeed or fail in the entirety. A transaction is an atomic operation which may not me divided into smaller operations.

ROLLBACK : undo changes
COMMIT : changes are permanent

Transaction properties:

ACID Properties

- AUTOMICITY (all-or-none concept) : This property ensures that either all operations of the transactions are reflected properly in the database, or none are.
- CONSISTENCY : This property say that if databases was in a consistent state before the state of transaction-execution, then after termination of transaction, the databases will also consistent state.

- ISOLATION : Means one transaction is not effect on other transactions until it is not complete.
- DURABILITY : This property of a transaction ensures that after the successful completion of a transaction (COMMIT) the changes made bu this is permanent.

See Example 15.3 at page 547.

QUESTION PAPER -1

Q.1 select substr (‘rajeev’,2,3);

Q.2 select substr (‘rajeev’,-2,3);

Q.3 select round(123.456,2);

Q.4. select round(123.456,-2);

Q.5 How to change in a table that it add Foreign Key after create Table? (524)

Q.6 How to modify data in a table?

Q.7 How to insert data from another table?

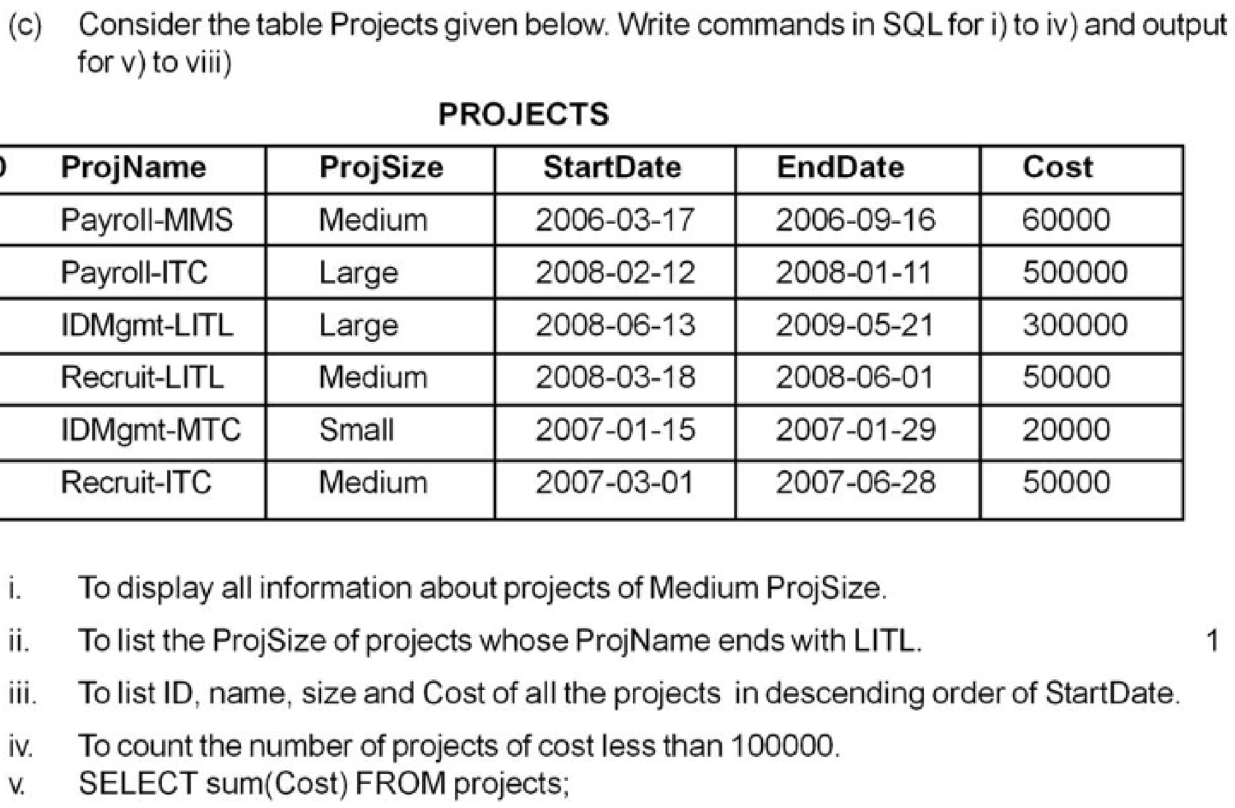
QUESTION PAPER -2

1. “Name” is a column in a table “school”. The SQL quire

```
select count(*) from school;            Output : 27
select count(Name) from school;         Output : 28
```

   What may be the possible reason for this?

2.



(c) Consider the table Projects given below. Write commands in SQL for i) to iv) and output for v) to viii)

PROJECTS

ID	ProjName	ProjSize	StartDate	EndDate	Cost
1	Payroll-MMS	Medium	2006-03-17	2006-09-16	60000
2	Payroll-ITC	Large	2008-02-12	2008-01-11	500000
3	IDMgmt-LITL	Large	2008-06-13	2009-05-21	300000
4	Recruit-LITL	Medium	2008-03-18	2008-06-01	50000
5	IDMgmt-MTC	Small	2007-01-15	2007-01-29	20000
6	Recruit-ITC	Medium	2007-03-01	2007-06-28	50000

i. To display all information about projects of Medium ProjSize.

ii. To list the ProjSize of projects whose ProjName ends with LITL. 1

iii. To list ID, name, size and Cost of all the projects in descending order of StartDate.

iv. To count the number of projects of cost less than 100000.

v. SELECT sum(Cost) FROM projects;

3. **ALTER TABLE WITH CONSTRAINTS**

    (i) adding column to a table
    (ii) modifying column-definitions of a table
    (iii) deleting column of a table
    (iv) adding constraints to a table
    (v) enable/disable constraints

4. If we use DAYOFWEEK the what will out for S, M, T, W, T, F and S.

5. How MySQL maintains data integrity?

6. What will be the output in this Insert Statements ?

```
create table r1
(code int(4) not null,
name varchar(20) not null,
pay decimal,
primary key (code, name) );

insert into r1 values (1,’Rajeev’,25000);
insert into r1 values (2,’Anil’,20000);
insert into r1 values (3,’Anil’,25000);
insert into r1 values (1,’Mohan’,21000);
```

7. What is inline definition and out-of-line definition?
8. What are Transaction properties?