# Chapter - 1
# Querying and SQL Functions

---

**Que 1. Answer the following questions:**

**a) Define RDBMS. Name any two RDBMS software.**

**b) What is the purpose of the following clauses in a select statement?**

**i) ORDER BY**

**ii) HAVING**

**c) Site any two differences between Single row functions and Aggregate functions.**

**d) What do you understand by Cartesian Product?**

**e) Write the name of the functions to perform the following operations:**

**i) To display the day like "Monday", "Tuesday", from the date when India got independence.**

**ii) To display the specified number of characters from a particular position of the given string.**

**iii) To display the name of the month in which you were born.**

**iv) To display your name in capital letters.**

**Ans.**

**(a)** RDBMS stands for Relational Database Management System. An RDBMS is a DBMS designed especially for relational databases, which provide the facility to store and manage large amount of data. It allows to store the data in structured format using rows and columns.

**Two RDBMS Software are –** MySQL, Oracle

**(b) (i) ORDER BY** – ORDER BY clause is used with SELECT statement, to arranges the result of an SQL query in either ascending or descending on the basis of particular columns (fields).

**(ii) HAVING** – HAVING clause used with SELECT statement to apply the condition on the grouped record. It is always use with GROUP BY.

(c) Single Row Functions and Aggregate Functions

- Single Row Function applied on the each row while Aggregate Functions applied on the group of rows.

- Single Row Functions return multiple output i.e. output based on each row while Aggregate function returns only one result i.e. output based on group of rows.

(d) A Cartesian product combines the tuples of one relation with all the tuples of the other relation. It is created when two tables are joined without any join condition.

(e) i) DAYNAME( )

ii) MID( ) or SUBSTRING( )

iii) MONTHNAME( )

iv) UCASE( ) or UPPER( )

Que 2. Write the output produced by the following SQL commands:

a) SELECT POW(2,3);

b) SELECT ROUND(123.2345, 2), ROUND(342.9234,-1);

c) SELECT LENGTH("Informatics Practices");

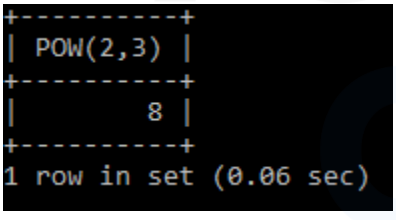
d) SELECT YEAR("1979/11/26"), MONTH("1979/11/26"), DAY("1979/11/26"), MONTHNAME("1979/11/26");

e) SELECT LEFT("INDIA",3), RIGHT("Computer Science",4);

f) SELECT MID("Informatics",3,4), SUBSTR("Practices",3);

Ans.

(a)




(b)

```
+--------------------+--------------------+
| ROUND(123.2345, 2) | ROUND(342.9234,-1) |
+--------------------+--------------------+
|             123.23 |                340 |
+--------------------+--------------------+
1 row in set (0.08 sec)
```

(c)

```
+-----------------------------------+
| LENGTH("Informatics Practices") |
+-----------------------------------+
|                              21 |
+-----------------------------------+
1 row in set (0.07 sec)
```

(d)

```
+--------------------+---------------------+------------------+-------------------------+
| YEAR("1979/11/26") | MONTH("1979/11/26") | DAY("1979/11/26") | MONTHNAME("1979/11/26") |
+--------------------+---------------------+------------------+-------------------------+
|               1979 |                  11 |               26 | November                |
+--------------------+---------------------+------------------+-------------------------+
1 row in set (0.00 sec)
```

(e)

```
+-----------------+-------------------------------+
| LEFT("INDIA",3) | RIGHT ("Computer Science",4) |
+-----------------+-------------------------------+
| IND             | ence                          |
+-----------------+-------------------------------+
1 row in set (0.00 sec)
```

(f)

```
+-----------------------+-----------------------+
| MID("Informatics",3,4) | SUBSTR("Practices",3) |
+-----------------------+-----------------------+
| form                  | actices               |
+-----------------------+-----------------------+
1 row in set (0.00 sec)
```

Que 3. Consider the following table named "Product", showing details of products being sold in a grocery shop.

| PCode | PName | UPrice | Manufacturer |
|---|---|---|---|
| P01 | Washing Powder | 120 | Surf |
| P02 | Tooth Paste | 54 | Colgate |
| P03 | Soap | 25 | Lux |
| P04 | Tooth Paste | 65 | Pepsodant |
| P05 | Soap | 38 | Dove |
| P06 | Shampoo | 245 | Dove |

a) Write SQL queries for the following:

i. Create the table Product with appropriate data types and constraints.

ii. Identify the primary key in Product.

iii. List the Product Code, Product name and price in descending order of their product name. If PName is the same then display the data in ascending order of price.

iv. Add a new column Discount to the table Product.

v. Calculate the value of the discount in the table Product as 10 per cent of the UPrice for all those products where the UPrice is more than 100, otherwise the discount will be 0.

vi. Increase the price by 12 per cent for all the products manufactured by Dove.

vii. Display the total number of products manufactured by each manufacturer.

b) Write the output(s) produced by executing the following queries on the basis of the information given above in the table Product:

i. SELECT PName, Average(UPrice) FROM Product GROUP BY Pname;

ii. SELECT DISTINCT Manufacturer FROM Product;

iii. SELECT COUNT(DISTINCT PName) FROM Product;

iv. SELECT PName, MAX(UPrice), MIN(UPrice) FROM Product GROUP BY PName;

Ans.

(a) i.

```
mysql> CREATE TABLE PRODUCT
    -> (PCode CHAR(4) PRIMARY KEY,
    -> PName VARCHAR(25) NOT NULL,
    -> UPrice DECIMAL(10,2),
    -> Manufacturer VARCHAR(30));
Query OK, 0 rows affected (0.99 sec)
```

**ii. PCode**

**iii. SELECT PCODE, PNAME, UPRICE**
**FROM PRODUCT**
**ORDER BY PNAME DESC, UPRICE ASC;**

```
mysql> SELECT PCODE, PNAME, UPRICE
    -> FROM PRODUCT
    -> ORDER BY PNAME DESC, UPRICE ASC;
+-------+----------------+--------+
| PCODE | PNAME          | UPRICE |
+-------+----------------+--------+
| P01   | Washing Powder | 120.00 |
| P02   | Tooth Paste    |  54.00 |
| P04   | Tooth Paste    |  65.00 |
| P03   | Soap           |  25.00 |
| P05   | Soap           |  38.00 |
| P06   | Shampoo        | 245.00 |
+-------+----------------+--------+
6 rows in set (0.00 sec)
```

**iv. ALTER TABLE PRODUCT ADD DISCOUNT DECIMAL (10,2);**

**v.** mysql> UPDATE PRODUCT SET DISCOUNT = UPRICE * 0.10
-> WHERE UPRICE > 100;

**vi.** mysql> UPDATE PRODUCT
-> SET UPRICE = UPRICE + UPRICE * 0.12
-> WHERE MANUFACTURER = 'DOVE';

**vii.** mysql > SELECT MANUFACTURER, COUNT(*)
-> FROM PRODUCT
-> GROUP BY MANUFACTURER;

**(b) i.**

```
mysql> SELECT PName, Avg(UPrice) FROM Product GROUP BY Pname;
+----------------+-------------+
| PName          | Avg(UPrice) |
+----------------+-------------+
| Washing Powder |  120.000000 |
| Tooth Paste    |   59.500000 |
| Soap           |   33.780000 |
| Shampoo        |  274.400000 |
+----------------+-------------+
4 rows in set (0.00 sec)
```

ii.

```
mysql> SELECT DISTINCT Manufacturer FROM Product;
+--------------+
| Manufacturer |
+--------------+
| Surf         |
| Colgate      |
| Lux          |
| Pepsodant    |
| Dove         |
+--------------+
5 rows in set (0.05 sec)
```

iii.

```
mysql> SELECT COUNT(DISTINCT PName) FROM Product;
+-----------------------+
| COUNT(DISTINCT PName) |
+-----------------------+
|                     4 |
+-----------------------+
1 row in set (0.00 sec)
```

iv.

```
mysql> SELECT PName, MAX(UPrice), MIN(UPrice) FROM Product GROUP BY PName;
+----------------+-------------+-------------+
| PName          | MAX(UPrice) | MIN(UPrice) |
+----------------+-------------+-------------+
| Washing Powder |      120.00 |      120.00 |
| Tooth Paste    |       65.00 |       54.00 |
| Soap           |       42.56 |       25.00 |
| Shampoo        |      274.40 |      274.40 |
+----------------+-------------+-------------+
4 rows in set (0.05 sec)
```

Que 4. Using the CARSHOWROOM database given in the chapter, write the SQL queries for the following:

**Table 1.1 INVENTORY**

```
mysql> SELECT * FROM INVENTORY;
+-------+--------+----------+----------+-----------------+----------+
| CarId | CarName| Price    | Model    | YearManufacture | Fueltype |
+-------+--------+----------+----------+-----------------+----------+
| D001  | Car1   | 582613.00 | LXI      |            2017 | Petrol   |
| D002  | Car1   | 673112.00 | VXI      |            2018 | Petrol   |
| B001  | Car2   | 567031.00 | Sigma1.2 |            2019 | Petrol   |
| B002  | Car2   | 647858.00 | Delta1.2 |            2018 | Petrol   |
| E001  | Car3   | 355205.00 | 5 STR STD|            2017 | CNG      |
| E002  | Car3   | 654914.00 | CARE     |            2018 | CNG      |
| S001  | Car4   | 514000.00 | LXI      |            2017 | Petrol   |
| S002  | Car4   | 614000.00 | VXI      |            2018 | Petrol   |
+-------+--------+----------+----------+-----------------+----------+
```

a) Add a new column Discount in the INVENTORY table.

b) Set appropriate discount values for all cars keeping in mind the following:

(i) No discount is available on the LXI model.

(ii) VXI model gives a 10% discount.

(iii) A 12% discount is given on cars other than LXI model and VXI model.

c) Display the name of the costliest car with fuel type “Petrol”.

d) Calculate the average discount and total discount available on Car4.

e) List the total number of cars having no discount.

Ans.

(a) ALTER TABLE INVENTORY ADD DISCOUNT DECIMAL(10,2);

(b) (i) UPDATE INVENTORY SET DISCOUNT = NULL WHERE MODEL = 'LXI';

(ii) UPDATE INVENTORY SET DISCOUNT = PRICE * 0.10 WHERE MODEL = 'VXI';

(iii) UPDATE INVENTORY SET DISCOUNT = PRICE * 0.12 WHERE MODEL NOT IN ('LXI', 'VXI')

(c) SELECT CARNAME FROM INVENTORY WHERE PRICE = (SELECT MAX(PRICE) FROM INVENTORY WHERE FUELTYPE = 'PETROL' );

(d) SELECT AVG(DISCOUNT), SUM(DISCOUNT) FROM INVENTORY GROUP BY CARNAME HAVING CARNAME = 'CAR4';

(e) SELECT * FROM INVENTORY
WHERE DISCOUNT IS NULL;

Que 5. Consider the following tables Student and Stream in the Streams_of_Students database. The primary key of the Stream table is StCode (stream code) which is the foreign key in the Student table. The primary key of the Student table is AdmNo (admission number).

| AdmNo | Name | StCode |
|---|---|---|
| 211 | Jay | NULL |
| 241 | Aditya | S03 |
| 290 | Diksha | S01 |
| 333 | Jasqueen | S02 |
| 356 | Vedika | S01 |
| 380 | Ashpreet | S03 |

| StCode | Stream |
|---|---|
| S01 | Science |
| S02 | Commerce |
| S03 | Humanities |

Write SQL queries for the following:

a) Create the database Streams_Of_Students.

b) Create the table Student by choosing appropriate data types based on the data given in the table.

c) Identify the Primary keys from tables Student and Stream. Also, identify the foreign key from the table Stream.

d) Jay has now changed his stream to Humanities. Write an appropriate SQL query to reflect this change.

e) Display the names of students whose names end with the character 'a'. Also, arrange the students in alphabetical order.

f) Display the names of students enrolled in Science and Humanities stream, ordered by student name in alphabetical order, then by admission number in ascending order (for duplicating names).

g) List the number of students in each stream having more than 1 student.

h) Display the names of students enrolled in different streams, where students are arranged in descending order of admission number.

i) Show the Cartesian product on the Student and Stream table. Also mention the degree and cardinality produced after applying the Cartesian product.

j) Add a new column 'TeacherIncharge" in the Stream table. Insert appropriate data in each row.

k) List the names of teachers and students.

l) If Cartesian product is again applied on Student and Stream tables, what will be the degree and cardinality of this modified table?

Ans.

(a) mysql> CREATE DATABASES STREAMS_OF_STUDENTS;

(b)
```
mysql> CREATE TABLE STREAM (
            STCODE VARCHAR(4) PRIMARY KEY,
            STREAM VARCHAR(30) );
mysql> CREATE TABLE STUDENT (
            ADMNO INT(4) PRIMARY KEY,
            NAME VARCHAR(30) NOT NULL,
            STCODE VARCHAR(4),
FOREIGN KEY (STCODE) REFERENCES STREAM(STCODE) );
```

(c) TABLE : STUDENT

PRIMARY KEY : ADMNO, FOREIGN KEY: STCODE

TABLE : STREAM
PRIMARY KEY : STCODE

**(d) mysql>** UPDATE STUDENT SET STCODE = 'HUMANITIES' WHERE NAME = 'JAY';

**(e) mysql>** SELECT NAME FROM STUDENT
**WHERE NAME LIKE '%A'**
**ORDER BY NAME ASC;**

**(f)** mysql> SELECT NAME, STREAM FROM STUDENT, STREAM
WHERE STUDENT.STCODE = STREAM.STCODE
AND STREAM.STREAM IN ('SCIENCE', 'HUMANITIES')
ORDER BY NAME, ADMNO

**(g) mysql>** SELECT STREAM, COUNT(STREAM) FROM STUDENT, STREAM
WHERE STUDENT.STCODE = STREAM.STCODE
GROUP BY STREAM
HAVING COUNT(STREAM) > 1

**(h) mysql>** SELECT NAME, STREAM FROM STUDENT, STREAM
WHERE STUDENT.STCODE = STREAM.STCODE
ORDER BY ADMNO DESC;

**(i) mysql>** SELECT * FROM STUDENT, STREAM;
**DEGREE OF CARTESIAN PRDUCT OF STUDENT & STREAM = 5 (3+2)**
**CARDINALITY OF CARTESIAN PRODUCT OF STUDENT & STREAM = 18 (6 X 3)**

**(j)** ALTER TABLE STREAM ADD COLUMN TEACHERINCHAGE VARCHAR(30);

**Insert appropriate data in each row**

a) mysql>UPDATE STREAM SET TEACHERINCHARGE = 'ANJEEV SINGH' WHERE STREAM = 'SCIENCE';

b) mysql>UPDATE STREAM SET TEACHERINCHARGE = 'RASHMI' WHERE STREAM = 'COMMERCE';

c) mysql> UPDATE STREAM SET TEACHERINCHARGE = 'AMRIT SINGH WHERE STREAM = 'HUMANITIES';

**(k)** mysql>SELECT NAME, TEACHERINCHARGE
FROM STUDENT, STREAM
WHERE STUDENT.STCODE = STREAM.STCODE;

(l) => CARDINALITY OF STUDENT & STREAM TABLE : IS 18 (SAME) Because no any new rows are added

=> DEGREE OF STUDENT & STREAM TABLE: 3 + 3= 6,
Because 1 column is added in STREAM table.