**SRM DBMS ELAB**

**1ST SESSION :**

QUESTION

* **SESSION NAME:**Basic Select
* **QUESTION NAME:** Query NAME, COUNTRY GREATHER 4000
* **TABLE DESCRIPTION**

Query the NAME and COUNTRY CODE of CITY Table where ID is greater than 4000

**QUERY:**

select NAME,COUNTRYCODE from city where id>4000;

QUESTION

* **SESSION NAME:**Basic Select
* **QUESTION NAME:** Display the Population more than 100000
* **TABLE DESCRIPTION**

Query all columns in CITY with populations larger than 100000.

**QUERY:**

select \* from CITY where POPULATION>100000;

QUESTION

* **SESSION NAME:**Basic Select
* **QUESTION NAME:** Query India
* **TABLE DESCRIPTION**

Query all attributes of every Indian city in the CITY table. The COUNTRYCODE for India is IND.

**QUERY:**

select \* from CITY where COUNTRYCODE='IND';

QUESTION

* **SESSION NAME:**Basic Select
* **QUESTION NAME:** Query SRM CARE - Name
* **TABLE DESCRIPTION**

Query all attributes CITY table with name as "SRMCARE".

**QUERY:**

select \* from CITY where name='SRMCARE';

QUESTION

* **SESSION NAME:**Basic Select
* **QUESTION NAME:** Weather Observation - Query 1
* **TABLE DESCRIPTION**

Query a list of CITY and STATE from the STATION table.  
  
Input Format  
  
The STATION table is described as follows:

**QUERY:**

select city,state from station ;

QUESTION

* **SESSION NAME:**Basic Select
* **QUESTION NAME:** Query ALL DATA 3000
* **TABLE DESCRIPTION**

Query all data of the CITY Table where ID is greater than 3000

**QUERY:**

select \* from city where id >3000;

QUESTION

* **SESSION NAME:**Basic Select
* **QUESTION NAME:** Display the complete table
* **TABLE DESCRIPTION**

Query all columns of the table CITY

**QUERY:**

select \* from city;

QUESTION

* **SESSION NAME:**Basic Select
* **QUESTION NAME:** Query ALL DATA 4000
* **TABLE DESCRIPTION**

Query all data of the CITY Table where ID is greater than 4000

**QUERY:**

select \* from city where id > 4000;

QUESTION

* **SESSION NAME:**Basic Select
* **QUESTION NAME:** Query NAME, COUNTRY CODE GREATER THAN 3000
* **TABLE DESCRIPTION**

Query the NAME and COUNTRY CODE of CITY Table where ID is greater than 3000

**QUERY:**

select NAME,COUNTRYCODE from city where id>3000;

QUESTION

* **SESSION NAME:**Basic Select
* **QUESTION NAME:** Query TAMIILNADU
* **TABLE DESCRIPTION**

Query all attributes of Tamil Nadu district in the CITY table. The district code for Tamil Nadu is TN.

**QUERY:**

select \* from city where district = 'tn';

**SESSION 2:**

QUESTION

* **SESSION NAME:**Advance Select - 1
* **QUESTION NAME:** Weather Observation - Vowels No Start
* **TABLE DESCRIPTION**

Query the list of CITY names from STATION that do not start with vowels. Your result cannot contain duplicates.  
  
Input Format  
  
The STATION table is described as follows:

**QUERY:**

select DISTINCT CITY from STATION where CITY NOT RLIKE '^[aeiouAEIOU].\*$';

QUESTION

* **SESSION NAME:**Advance Select - 1
* **QUESTION NAME:** Weather Observation - Vowels No End
* **TABLE DESCRIPTION**

Query the list of CITY names from STATION that do not end with vowels. Your result cannot contain duplicates.  
  
Input Format  
  
The STATION table is described as follows:

**QUERY:**

select distinct CITY from STATION where CITY RLIKE '[aeiou]\*[^aeiou]$';

QUESTION

* **SESSION NAME:**Advance Select - 1
* **QUESTION NAME:** Weather Observation - Vowels Start and End
* **TABLE DESCRIPTION**

Query the list of CITY names from STATION which have vowels (i.e., a, e, i, o, and u) as both their first and last characters. Your result cannot contain duplicates.  
  
Input Format  
  
The STATION table is described as follows:  
  
  
Input Format  
  
The STATION table is described as follows:

**QUERY:**

select DISTINCT CITY FROM STATION where CITY RLIKE '^[AEIOUaeiou].\*[aeiouAEIOU]$';

QUESTION

* **SESSION NAME:**Advance Select - 1
* **QUESTION NAME:** Weather Observation - Query 3
* **TABLE DESCRIPTION**

Let N be the number of CITY entries in STATION, and let N' be the number of distinct CITY names in STATION; query the value of N - N' from STATION. In other words, find the difference between the total number of CITY entries in the table and the number of distinct CITY entries in the table.  
  
Input Format  
  
The STATION table is described as follows:

**QUERY:**

select COUNT(CITY)-COUNT(DISTINCT CITY) from STATION;

QUESTION

* **SESSION NAME:**Advance Select - 1
* **QUESTION NAME:** Weather Observation - Vowels Ending
* **TABLE DESCRIPTION**

Query the list of CITY names ending with vowels (i.e., a, e, i, o, or u) from STATION. Your result cannot contain duplicates.  
  
Input Format  
  
The STATION table is described as follows:

**QUERY:**

select DISTINCT CITY from STATION where CITY RLIKE '[aeiou]$';

QUESTION

* **SESSION NAME:**Advance Select - 1
* **QUESTION NAME:** Query Population - 12000 in USA
* **TABLE DESCRIPTION**

Query the table of all American cities in CITY with populations larger than 120000. The CountryCode for America is USA.

**QUERY:**

select \* from CITY where COUNTRYCODE='USA' AND POPULATION>120000;

QUESTION

* **SESSION NAME:**Advance Select - 1
* **QUESTION NAME:** Weather Observation Station 11
* **TABLE DESCRIPTION**

Query the list of CITY names from STATION that either do not start with vowels or do not end with vowels. Your result cannot contain duplicates.  
  
Input Format  
  
The STATION table is described as follows:

**QUERY:**

SELECT DISTINCT CITY FROM STATION WHERE CITY RLIKE '^[^AEIOUaeiou]|[^aeiouAEIOU]$' ;

QUESTION

* **SESSION NAME:**Advance Select - 1
* **QUESTION NAME:** Query NAME, COUNTRY CODE LESS THAN 4000
* **TABLE DESCRIPTION**

Query the NAME and COUNTRY CODE of CITY Table where ID is greater than 3000 and Less than 4000

**QUERY:**

select NAME,COUNTRYCODE from CITY where ID>3000 and ID<4000;

QUESTION

* **SESSION NAME:**Advance Select - 1
* **QUESTION NAME:** Query ID Less than 4000 and Greater than 3000
* **TABLE DESCRIPTION**

Query the ID, NAME and COUNTRY CODE of CITY Table where ID is greater than 3000 and less than 4000

**QUERY:**

select ID,NAME,COUNTRYCODE from CITY where ID>3000 and ID<4000;

QUESTION

* **SESSION NAME:**Advance Select - 1
* **QUESTION NAME:** Weather Observation -  Start
* **TABLE DESCRIPTION**

Query the list of CITY names from STATION which have vowels (i.e., a, e, i, o, and u) as first characters. Your result cannot contain duplicates.  
  
Input Format  
  
The STATION table is described as follows:  
  
  
Input Format  
  
The STATION table is described as follows:

**QUERY:**

select DISTINCT CITY from STATION where CITY RLIKE '^[AEIOUaeiou]';

**SESSION 3**

QUESTION

* **SESSION NAME:**Advance Select - 2
* **QUESTION NAME:** Weather Observation - Longest
* **TABLE DESCRIPTION**

Query the two cities in STATION with the longest, as well as their lengths (i.e.: number of characters in the name).  
  
Input Format  
  
The STATION table is described as follows:  
Input Format  
  
The STATION table is described as follows:

**Query:**

**select city, length(city) from station order by length(city) DESC,city ASC LIMIT 1;**

QUESTION

* **SESSION NAME:**Advance Select - 2
* **QUESTION NAME:** Weather Observation - Even Numbers
* **TABLE DESCRIPTION**

Query a list of CITY names from STATION with even ID numbers only. You may print the results in any order, but must exclude duplicates from your answer.  
  
Input Format  
  
The STATION table is described as follows:

**Query:**

**select distinct CITY from STATION where ID % 2 = 0;**

QUESTION

* **SESSION NAME:**Advance Select - 2
* **QUESTION NAME:** The PADS
* **TABLE DESCRIPTION**

Generate the following two result sets:  
  
Query an alphabetically ordered list of all names in OCCUPATIONS, immediately followed by the first letter of each profession as a parenthetical (i.e.: enclosed in parentheses). For example: AnActorName(A), ADoctorName(D), AProfessorName(P), and ASingerName(S).  
Query the number of ocurrences of each occupation in OCCUPATIONS. Sort the occurrences in ascending order, and output them in the following format:  
  
There are a total of [occupation\_count] [occupation]s.  
where [occupation\_count] is the number of occurrences of an occupation in OCCUPATIONS and [occupation] is the lowercase occupation name. If more than one Occupation has the same [occupation\_count], they should be ordered alphabetically.  
  
Note: There will be at least two entries in the table for each type of occupation.

**Query:**

**SELECT CONCAT(NAME,'(',SUBSTR(OCCUPATION,1,1),')') AS N**

**FROM OCCUPATIONS**

**ORDER BY N;**

**SELECT CONCAT('There are a total of ',COUNT(OCCUPATION),' ',LOWER(OCCUPATION),'s.')**

**FROM OCCUPATIONS**

**GROUP BY OCCUPATION**

**ORDER BY COUNT(OCCUPATION), OCCUPATION;**

QUESTION

* **SESSION NAME:**Advance Select - 2
* **QUESTION NAME:** Higher than 75 Marks
* **TABLE DESCRIPTION**

Query the Name of any student in STUDENTS who scored higher than 75 Marks. Order your output by the last three characters of each name. If two or more students both have names ending in the same last three characters (i.e.: Bobby, Robby, etc.), secondary sort them by ascending ID.

**Query:**

**select Name from STUDENTS**

**where marks > 75**

**ORDER BY SUBSTR(Name,LENGTH(Name)-2,3),ID;**

QUESTION

* **SESSION NAME:**Advance Select - 2
* **QUESTION NAME:** Weather Observation - Vowels
* **TABLE DESCRIPTION**

Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from STATION. Your result cannot contain duplicates.  
  
Input Format  
  
The STATION table is described as follows:

**Query:**

**select distinct CITY from STATION where CITY RLIKE '^[AEIOUaeiou]' ;**

QUESTION

* **SESSION NAME:**Advance Select - 2
* **QUESTION NAME:** Employee Salaries
* **TABLE DESCRIPTION**

Write a query that prints a list of employee names (i.e.: the name attribute) for employees in Employee having a salary greater than $2000 per month who have been employees for less than 10 months. Sort your result by ascending employee\_id.  
  
Input Format  
  
The Employee table containing employee data for a company is described as follows:

**Query:**

**select NAME from EMPLOYEE where salary > 2000 and months < 10 order by employee\_id;**

QUESTION

* **SESSION NAME:**Advance Select - 2
* **QUESTION NAME:** Employee Salaries
* **TABLE DESCRIPTION**

Write a query that prints a list of employee names (i.e.: the name attribute) for employees in Employee having a salary greater than $2000 per month who have been employees for less than 10 months. Sort your result by ascending employee\_id.  
  
Input Format  
  
The Employee table containing employee data for a company is described as follows:

**Query:**

**select NAME from EMPLOYEE where salary > 2000 and months < 10 order by employee\_id;**

QUESTION

* **SESSION NAME:**Advance Select - 2
* **QUESTION NAME:** Weather Observation Station 12
* **TABLE DESCRIPTION**

Query the list of CITY names from STATION that do not start with vowels and do not end with vowels. Your result cannot contain duplicates.  
  
Input Format  
  
The STATION table is described as follows:

**Query:**

**select distinct CITY from STATION where CITY RLIKE '^[^AEIOUaeiou].+[^AEIOUaeiou]$';**

QUESTION

* **SESSION NAME:**Advance Select - 2
* **QUESTION NAME:** Triangles
* **TABLE DESCRIPTION**

Write a query identifying the type of each record in the TRIANGLES table using its three side lengths. Output one of the following statements for each record in the table:  
  
Equilateral: It's a triangle with 3 sides of equal length.  
Isosceles: It's a triangle with 2 sides of equal length.  
Scalene: It's a triangle with 3 sides of differing lengths.  
Not A Triangle: The given values of A, B, and C don't form a triangle.  
  
Input Format  
  
The TRIANGLES table is described as follows:

**Query:**

**select**

**CASE**

**WHEN (A + B <= C OR A + C <= B OR B + C <= A) THEN**

**'Not A Triangle'**

**WHEN (A = B AND B = C) THEN**

**'Equilateral'**

**WHEN (A = B OR A = C OR B = C) THEN**

**'Isosceles'**

**WHEN (A <> B AND B <> C) THEN**

**'Scalene'**

**END TRIANGLES**

**from TRIANGLES;**

QUESTION

* **SESSION NAME:**Advance Select - 2
* **QUESTION NAME:** Employee Names
* **TABLE DESCRIPTION**

Write a query that prints a list of employee names (i.e.: the name attribute) from the Employee table in alphabetical order.  
  
Input Format  
  
The Employee table containing employee data for a company is described as follows:

**Query:**

**select NAME from EMPLOYEE order by NAME ASC;**

**SESSION -4**

QUESTION

* **SESSION NAME:**SQL Functions - 1
* **QUESTION NAME:** Average Function
* **TABLE DESCRIPTION**

Write a query that prints the average of the price column.  
  
Note: Use SQL in-built Function to calculate the average of the price column  
  
The statement has to return a single result which contains the average value of everything returned in the price column from the items\_ordered table.  
  
Input Format  
  
The items\_ordered table containing data for a company is described as follows:

**Query:**

**select avg(price) from items\_ordered;**

QUESTION

* **SESSION NAME:**SQL Functions - 1
* **QUESTION NAME:** Count all Columns
* **TABLE DESCRIPTION**

Write a query to display number of rows(Only count ) in the items\_ordered table  
  
  
Note: Use SQL in-built Function to find the count  
  
Input Format  
  
The items\_ordered table containing data for a company is described as follows:

**Query:**

**select count(\*)from items\_ordered;**

QUESTION

* **SESSION NAME:**SQL Functions - 1
* **QUESTION NAME:** Minimum Tents
* **TABLE DESCRIPTION**

For all of the tents that were ordered in the items\_ordered table, what is the price of the lowest tent?  
  
Hint: Your query should return the price only.  
  
Note: Use SQL in-built Function to calculate minimum value in tents  
  
Input Format  
  
The items\_ordered table containing data for a company is described as follows:

**Query:**

**select min(price) from items\_ordered where item='tent';**

QUESTION

* **SESSION NAME:**SQL Functions - 1
* **QUESTION NAME:** Absolute Mathematical Functions
* **TABLE DESCRIPTION**

Display all the details from the items\_ordered table by using (select \* from items\_ordered table).  
  
Perform Absolute function (abs) for the columns customerid, price and quantity respectively.  
  
Note: The final result should have only customerid, price and quantity  
  
Input Format  
  
The items\_ordered table containing data for a company is described as follows:

**Query:**

**select abs(customerid), abs(price), abs(quantity) from items\_ordered;**

QUESTION

* **SESSION NAME:**SQL Functions - 1
* **QUESTION NAME:** Minuimum of Items ordered
* **TABLE DESCRIPTION**

Write a query that prints the minimum of customerid, quantity and price in items\_ordered table.  
  
Note: Use SQL in-built Function to calculate minimum of the tuples in items\_ordered table.  
  
The statement has to return a single result which contains the minimum value of customerid, quantity and price columns of everything returned in the items\_ordered table.  
  
Input Format  
  
The items\_ordered table containing data for a company is described as follows:

**Query:**

**select min(customerid),min(quantity),min(price) from items\_ordered;**

QUESTION

* **SESSION NAME:**SQL Functions - 1
* **QUESTION NAME:** Maximum of Items ordered
* **TABLE DESCRIPTION**

Write a query that prints the maximum of customerid, quantity and price in items\_ordered table.  
  
Note: Use SQL in-built Function to calculate maximum of the tuples in items\_ordered table.  
  
The statement has to return a single result which contains the maximum value of customerid, quantity and price columns of everything returned in the items\_ordered table.  
  
Input Format  
  
The items\_ordered table containing data for a company is described as follows:

**Query:**

**select max(customerid), max(quantity), max(price) from items\_ordered ;**

QUESTION

* **SESSION NAME:**SQL Functions - 1
* **QUESTION NAME:** Maximum of Items ordered
* **TABLE DESCRIPTION**

Write a query that prints the maximum of customerid, quantity and price in items\_ordered table.  
  
Note: Use SQL in-built Function to calculate maximum of the tuples in items\_ordered table.  
  
The statement has to return a single result which contains the maximum value of customerid, quantity and price columns of everything returned in the items\_ordered table.  
  
Input Format  
  
The items\_ordered table containing data for a company is described as follows:

**Query:**

**select max(customerid), max(quantity), max(price) from items\_ordered ;**

QUESTION

* **SESSION NAME:**SQL Functions - 1
* **QUESTION NAME:** Maximum and Minimum of Items ordered
* **TABLE DESCRIPTION**

Write a query that prints the minimum and maximum of customerid, quantity and price in items\_ordered table.  
  
Note: Use SQL in-built Function to calculate minimum and maximum of the tuples in items\_ordered table.  
  
The statement has to return a single result which contains the minimum and maximum value of customerid, quantity and price columns of everything returned in the items\_ordered table.  
  
Input Format  
  
The items\_ordered table containing data for a company is described as follows:

**Query:**

**select min(customerid),min(quantity), min(price), max(customerid), max(quantity),max(price) from items\_ordered;**

QUESTION

* **SESSION NAME:**SQL Functions - 1
* **QUESTION NAME:** Sum Function
* **TABLE DESCRIPTION**

Write a query that prints the sum of the customerid, quantity and price.  
  
Note: Use SQL in-built Function to calculate the sum of the tuples in items\_ordered table.  
  
The statement has to return a single result which contains the sum value customerid, quantity and price columns of everything returned in the items\_ordered table.  
  
Input Format  
  
The items\_ordered table containing data for a company is described as follows:

**Query:**

**select sum(customerid),sum(quantity),sum(price) from items\_ordered;**

QUESTION

* **SESSION NAME:**SQL Functions - 1
* **QUESTION NAME:** Sum and Average of Items ordered
* **TABLE DESCRIPTION**

Write a query that prints the sum and average of the customerid, quantity and price.  
  
Note: Use SQL in-built Function to calculate the sum and average of the tuples in items\_ordered table.  
  
The statement has to return a single result which contains the sum and average value of customerid, quantity and price columns of everything returned in the items\_ordered table.  
  
Input Format  
  
The items\_ordered table containing data for a company is described as follows:

**Query:**

**select sum(customerid),sum(quantity),sum(price),avg(customerid),avg(quantity),avg(price) from items\_ordered;**

QUESTION

* **SESSION NAME:**SQL Functions - 1
* **QUESTION NAME:** Arithmetic Operations
* **TABLE DESCRIPTION**

Perform Arithmetic Operations.  
  
Display all the details from the items\_ordered table by using (select \* from items\_ordered table).  
  
Perform arithmetic query by adding 113 in customer id filed, multiply 50 in quantity, reduce the price by 50 and divide the customer id by 100 respectively and duly.  
  
customerid+113  
quantity\*50  
price-50  
customerid/100  
  
Input Format  
  
The items\_ordered table containing data for a company is described as follows:

**Query:**

**select customerid+113,quantity\*50,**

**price-50,**

**customerid/100 from items\_ordered;**

**SESSION 5**

QUESTION

* **SESSION NAME:**SQL Functions - 2
* **QUESTION NAME:** MID FUNCTION - SCALAR FUNCTION - NAME column
* **TABLE DESCRIPTION**

MID Function:  
  
MID Function has three parameters:  
  
First Parameter = Column Name  
Second Parameter = Start Length  
Third Parameter = Length of characters to be displayed  
Note: Index starts from 1.  
  
Example:  
  
MID(column\_name, start, length) from table-name;  
  
Write a FUNCTION to display the name column of employee table using mid function.  
  
The start length (index) is 1 and display total of 3 characters in name column.  
  
Table Name : emp  
Columns  
  
eid - Number  
name - varchar  
age - Number  
salary - Number

**Query:**

**select mid(name,1,3) from emp;**

QUESTION

* **SESSION NAME:**SQL Functions - 2
* **QUESTION NAME:** Round, Power and Square root Function
* **TABLE DESCRIPTION**

In this task, the user needs to write a query to implement Round, Power and Square root function.  
  
Note: Use SQL in-built Function to calculate round, power and square root function for station table.  
  
The user needs to implement round, power and square root function for the column lat\_n and long\_w and ID in customers table  
  
Round the lat\_n method to TWO digits.  
  
1. Hint: round(lat\_n , 2).  
  
2. For power function :  
  
Use lat\_n value as one parameter "x" value and the "y" as 5.  
Note: power(x,y) X= value denoted , lat\_n and Y denotes="5"  
  
Use long\_w value as one parameter "x" value and the "y" as 7.  
Note: power(x,y) X= value denoted , long\_w and Y denotes="7"  
  
3. Square root function for column ID.  
  
Input Format  
  
The station table is described as follow

**Query:**

**select ROUND(LAT\_N,2), POWER(LAT\_N, 5), POWER(LONG\_W,7),SQRT(ID) from STATION;**

QUESTION

* **SESSION NAME:**SQL Functions - 2
* **QUESTION NAME:** LCASE - LOWER
* **TABLE DESCRIPTION**

Write a function to convert all the names in the column to lower case.  
  
Table Name : emp  
Columns  
  
eid - Number  
name - varchar  
age - Number  
salary - Number  
  
Note: LCASE function is similar to LOWER function.  
Use LOWER Function

**Query:**

**select lower(name) from emp;**

QUESTION

* **SESSION NAME:**SQL Functions - 2
* **QUESTION NAME:** Round and Square root Function
* **TABLE DESCRIPTION**

In this task, the user needs to write a query to implement Round and Square root function.  
  
Note: Use SQL in-built Function to calculate round and square root function for items\_ordered table.  
  
The user needs to implement round and square root function for the column price in items\_ordered table.  
  
Round the price method TWO digits.  
  
Hint: round(price,2)  
  
Input Format  
  
The items\_ordered table containing data for a company is described as follows:

**Query:**

**select round(price,2), sqrt(price) from items\_ordered;**

QUESTION

* **SESSION NAME:**SQL Functions - 2
* **QUESTION NAME:** UCASE - UPPER
* **TABLE DESCRIPTION**

Write a function to convert all the names in the column to upper case.  
  
Table Name : emp  
Columns  
  
eid - Number  
name - varchar  
age - Number  
salary - Number  
  
Note: UCASE function is similar to UPPER function.  
Use UPPER Function

**Query:**

**select upper(name) from emp;**

QUESTION

* **SESSION NAME:**SQL Functions - 2
* **QUESTION NAME:** Sign Mathematical Functions
* **TABLE DESCRIPTION**

Write an query to execute the sign function in SQL.  
  
Table Name = Customers and the description for customer table is as follows:

**Query:**

**select sign(customerid) from customers;**

QUESTION

* **SESSION NAME:**SQL Functions - 2
* **QUESTION NAME:** MID FUNCTION - SCALAR FUNCTION - Column Salary
* **TABLE DESCRIPTION**

MID Function:  
  
MID Function has three parameters:  
  
First Parameter = Column Name  
Second Parameter = Start Length  
Third Parameter = Length of characters to be displayed  
Note: Index starts from 1.  
  
Example:  
  
MID(column\_name, start, length) from table-name;  
  
Write a FUNCTION to display the salary column of employee table using mid function.  
  
The start length (index) is 1 and display total of 4 characters in salary column.  
  
Table Name : emp  
Columns  
  
eid - Number  
name - varchar  
age - Number  
salary - Number

**Query:**

**select mid(salary,1,4) from emp;**

QUESTION

* **SESSION NAME:**SQL Functions - 2
* **QUESTION NAME:** ASCII - Value
* **TABLE DESCRIPTION**

Usage of ASCII Value = Returns the ASCII value for the specific character.  
  
ASCII(character or column name)  
  
character = The character to return the ASCII value for. If more than one character is entered, it will only return the value for the first character  
  
Display the name and the ascii value column (ASCII(name)) from students table

**Query:**

**select name, ASCII(name) from students;**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**

**Query:**