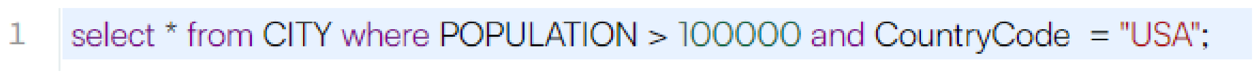
Structured Query Language

SQL Basics Queries:

Question 1:

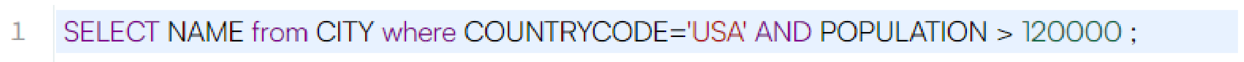
Query all columns for all American cities in the CITY table with populations larger than 100000. The CountryCode for America is USA.



```sql
select * from CITY where POPULATION > 100000 and CountryCode  = "USA";
```

Question 2:

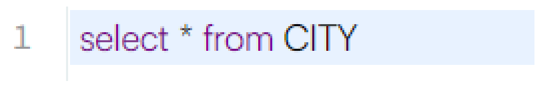
Query the NAME field for all American cities in the CITY table with populations larger than 120000. The CountryCode for America is USA.



```sql
SELECT NAME from CITY where COUNTRYCODE='USA' AND POPULATION > 120000 ;
```

Question 3:

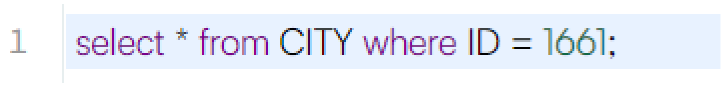
Query all columns (attributes) for every row in the **CITY** table.



```sql
select * from CITY
```

Question 4:

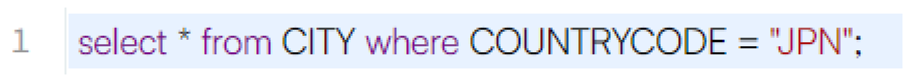
Query all columns for a city in **CITY** with the *ID* 1661.



```sql
select * from CITY where ID = 1661;
```

Question 5:

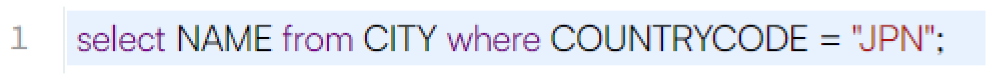
Query all attributes of every Japanese city in the **CITY** table. The **COUNTRYCODE** for Japan is JPN.



```sql
select * from CITY where COUNTRYCODE = "JPN";
```

Question 6:

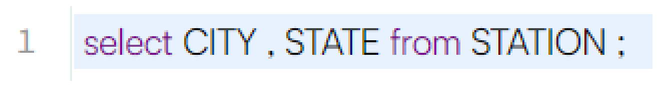
Query the names of all the Japanese cities in the **CITY** table. The **COUNTRYCODE** for Japan is JPN.



```sql
select NAME from CITY where COUNTRYCODE = "JPN";
```

Question 7:

Query a list of CITY and STATE from the STATION table.

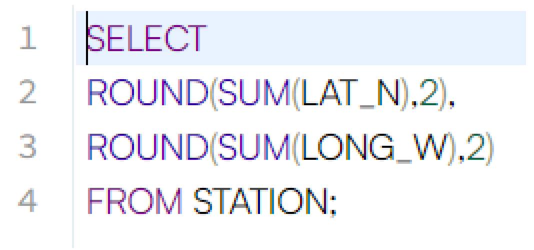


```
select CITY , STATE from STATION ;
```

Question 8:

Query the following two values from the STATION table:

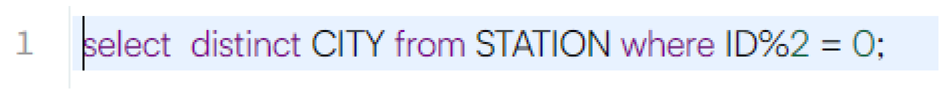
- The sum of all values in LAT_N rounded to a scale of decimal places.
- The sum of all values in LONG_W rounded to a scale of decimal places.



```
SELECT
ROUND(SUM(LAT_N),2),
ROUND(SUM(LONG_W),2)
FROM STATION;
```

Question 9:

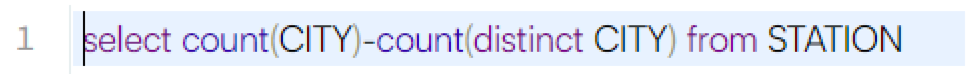
Query a list of CITY names from STATION for cities that have an even ID number. Print the results in any order, but exclude duplicates from the answer.



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

Question 10:

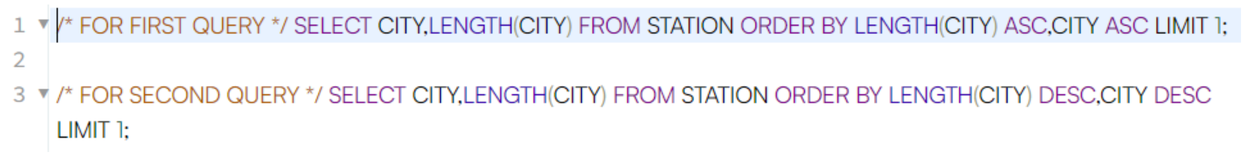
Find the difference between the total number of CITY entries in the table and the number of distinct CITY entries in the table.



```
select count(CITY)-count(distinct CITY) from STATION
```

Question 11:

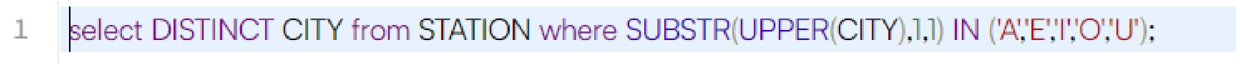
Query the two cities in STATION with the shortest and longest CITY names, as well as their respective lengths (i.e.: number of characters in the name). If there is more than one smallest or largest city, choose the one that comes first when ordered alphabetically.



```
/* FOR FIRST QUERY */ SELECT CITY,LENGTH(CITY) FROM STATION ORDER BY LENGTH(CITY) ASC,CITY ASC LIMIT 1;

/* FOR SECOND QUERY */ SELECT CITY,LENGTH(CITY) FROM STATION ORDER BY LENGTH(CITY) DESC,CITY DESC
LIMIT 1;
```

Question 12:

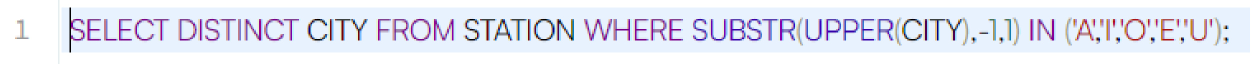
Query the list of CITY names starting with vowels (i.e., a, e, i, o, or u) from STATION. Your result cannot contain duplicates.



```
select DISTINCT CITY from STATION where SUBSTR(UPPER(CITY),1,1) IN ('A','E','I','O','U');
```

Question 13:

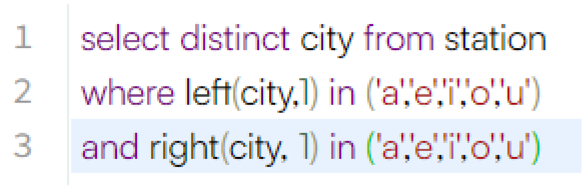
Query the list of CITY names ending with vowels (a, e, i, o, u) from STATION. Your result cannot contain duplicates.



```
SELECT DISTINCT CITY FROM STATION WHERE SUBSTR(UPPER(CITY),-1,1) IN ('A','I','O','E','U');
```

Question 14:

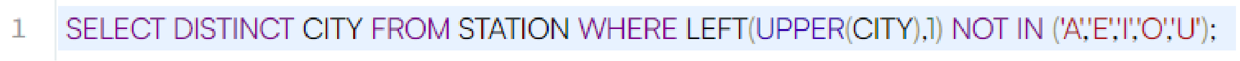
Query the list of CITY names from STATION which have vowels (i.e., a, e, i, o, and u) as both their first and last characters. Your result cannot contain duplicates.



```
select distinct city from station
where left(city,1) in ('a','e','i','o','u')
and right(city, 1) in ('a','e','i','o','u')
```

Question 15:

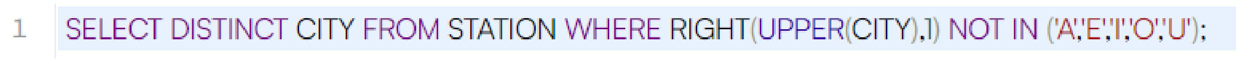
Query the list of CITY names from STATION that do not start with vowels. Your result cannot contain duplicates.



```
SELECT DISTINCT CITY FROM STATION WHERE LEFT(UPPER(CITY),1) NOT IN ('A','E','I','O','U');
```

Question 16:

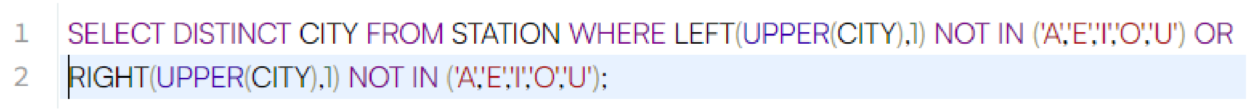
Query the list of CITY names from STATION that do not end with vowels. Your result cannot contain duplicates.



```
SELECT DISTINCT CITY FROM STATION WHERE RIGHT(UPPER(CITY),1) NOT IN ('A','E','I','O','U');
```

Question17:

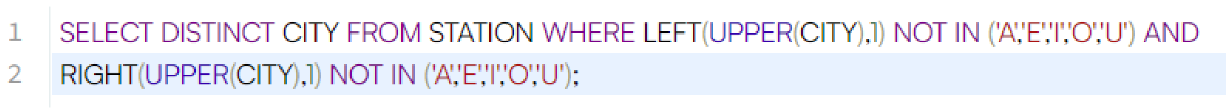
Query the list of CITY names from STATION that either do not start with vowels or do not end with vowels. Your result cannot contain duplicates.



```
SELECT DISTINCT CITY FROM STATION WHERE LEFT(UPPER(CITY),1) NOT IN ('A','E','I','O','U') OR
RIGHT(UPPER(CITY),1) NOT IN ('A','E','I','O','U');
```

Question 18:

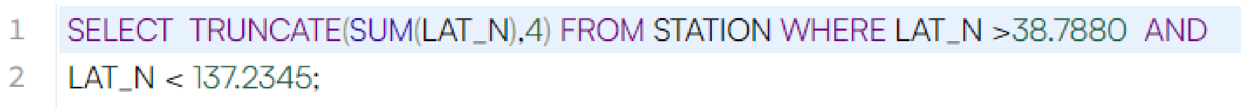
Query the list of CITY names from STATION that do not start with vowels and do not end with vowels. Your result cannot contain duplicates.



```
SELECT DISTINCT CITY FROM STATION WHERE LEFT(UPPER(CITY),1) NOT IN ('A','E','I','O','U') AND
RIGHT(UPPER(CITY),1) NOT IN ('A','E','I','O','U');
```

Question 19:

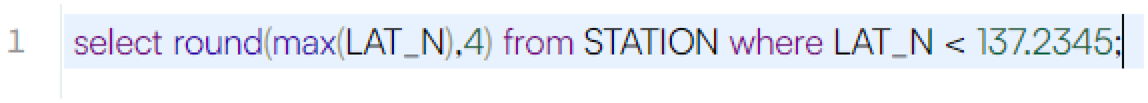
Query the sum of Northern Latitudes (LAT_N) from STATION having values greater than 38.7880 and less than 137.2324 . Truncate your answer to decimal places.



```
SELECT  TRUNCATE(SUM(LAT_N),4) FROM STATION WHERE LAT_N >38.7880  AND
LAT_N < 137.2345;
```

Question 20:

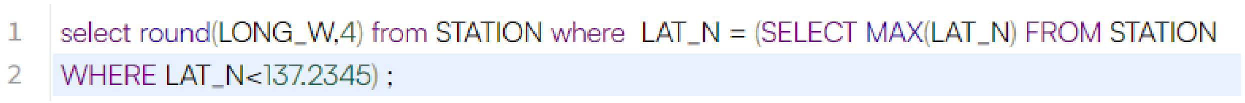
Query the greatest value of the Northern Latitudes (LAT_N) from STATION that is less than 137.2345. Truncate your answer to 4 decimal places.



```
select round(max(LAT_N),4) from STATION where LAT_N < 137.2345;
```

Question 21:

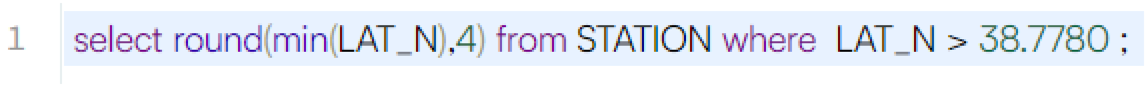
Query the Western Longitude (LONG_W) for the largest Northern Latitude (LAT_N) in STATION that is less than 137.2345. Round your answer to 4 decimal places.



```
select round(LONG_W,4) from STATION where  LAT_N = (SELECT MAX(LAT_N) FROM STATION
WHERE LAT_N<137.2345) ;
```

Question 21:

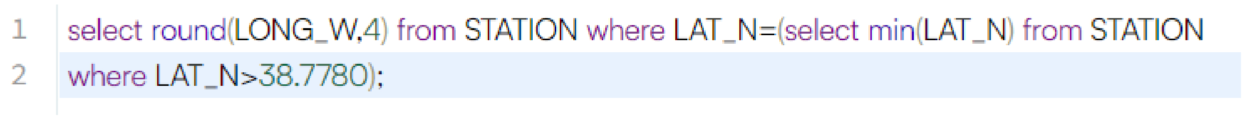
Query the smallest Northern Latitude (LAT_N) from STATION that is greater than 38.7780. Round your answer to 4 decimal places.



```
select round(min(LAT_N),4) from STATION where  LAT_N > 38.7780 ;
```

Question 22:

Query the Western Longitude (LONG_W)where the smallest Northern Latitude (LAT_N) in STATION is greater than 38.7780. Round your answer to 4 decimal places.



```
select round(LONG_W,4) from STATION where LAT_N=(select min(LAT_N) from STATION
where LAT_N>38.7780);
```

Question 23:

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.

```
select Name from STUDENTS where Marks > 75 order by RIGHT(Name,3), ID asc;
```

Question 24:

Write a query that prints a list of employee names (i.e.: the name attribute) from the Employee table in alphabetical order.

```
select name from Employee order by name;
```

Question 25:

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.

```
select name from Employee
where salary > 2000 and months < 10
order by employee_id asc;
```

Question 26:

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 3 with sides of equal length.
- Isosceles: It's a triangle 2 with sides of equal length.
- Scalene: It's a triangle 3 with sides of differing lengths.
- Not A Triangle: The given values of A, B, and C don't form a triangle.

```
SELECT CASE
WHEN A + B > C THEN CASE WHEN A = B AND B = C THEN 'Equilateral' WHEN A = B OR B = C OR A = C THEN 'Isosceles' WHEN A != B OR B != C OR A != C THEN 'Scalene' END
ELSE 'Not A Triangle' END FROM TRIANGLES;
```

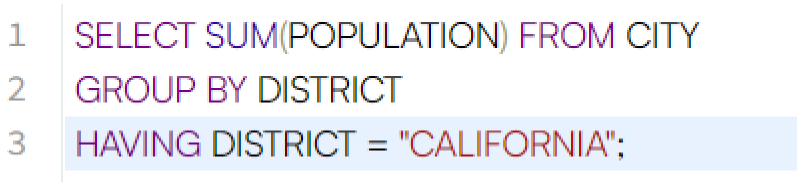
Question 27:

Query the average population for all cities in CITY, rounded down to the nearest integer.

```
SELECT ROUND(AVG(Population)) FROM City
```

Question 28:

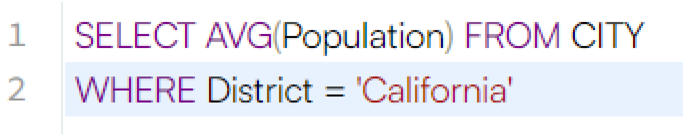
Query the total population of all cities in CITY where District is California.



```
SELECT SUM(POPULATION) FROM CITY
GROUP BY DISTRICT
HAVING DISTRICT = "CALIFORNIA";
```

Question 29:

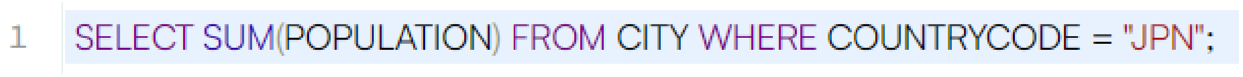
Query the average population of all cities in CITY where District is California.



```
SELECT AVG(Population) FROM CITY
WHERE District = 'California'
```

Question 30:

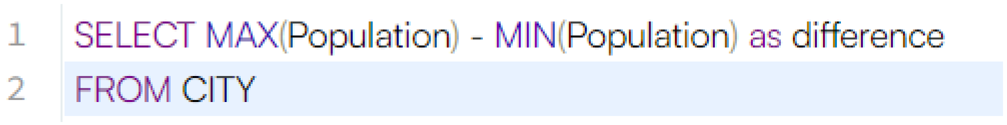
Query the sum of the populations for all Japanese cities in CITY. The COUNTRYCODE for Japan is JPN.



```
SELECT SUM(POPULATION) FROM CITY WHERE COUNTRYCODE = "JPN";
```

Question 31:

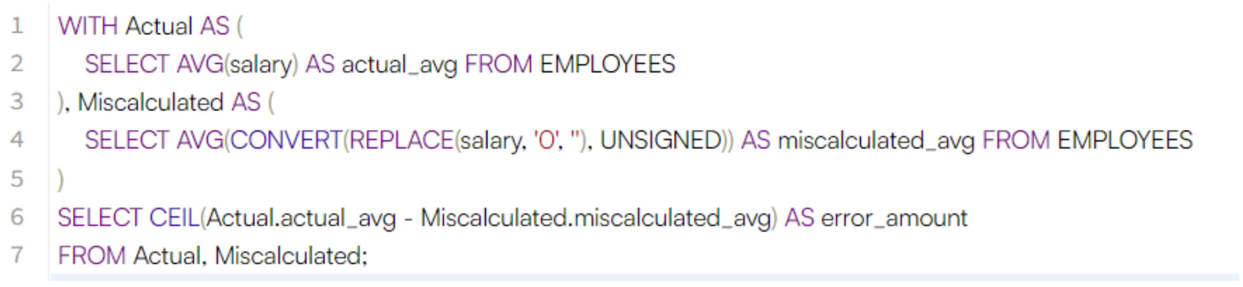
Query the difference between the maximum and minimum populations in CITY.



```
SELECT MAX(Population) - MIN(Population) as difference
FROM CITY
```

Question 32:

Samantha was tasked with calculating the average monthly salaries for all employees in the EMPLOYEES table, but did not realize her keyboard's 0 key was broken until after completing the calculation. She wants your help finding the difference between her miscalculation (using salaries with any zeros removed), and the actual average salary. Write a query calculating the amount of error (i.e.:actual-miscalculated average monthly salaries), and round it up to the next integer.



```
WITH Actual AS (
   SELECT AVG(salary) AS actual_avg FROM EMPLOYEES
), Miscalculated AS (
   SELECT AVG(CONVERT(REPLACE(salary, '0', ''), UNSIGNED)) AS miscalculated_avg FROM EMPLOYEES
)
SELECT CEIL(Actual.actual_avg - Miscalculated.miscalculated_avg) AS error_amount
FROM Actual, Miscalculated;
```

Question 33:

We define an employee's total earnings to be their monthly salary*months worked, and the maximum total earnings to be the maximum total earnings for any employee in the Employee table. Write a query to find the maximum total earnings for all employees as well as the total number of employees who have maximum total earnings. Then print these values as 2 space-separated integers.

```sql
select (salary * months) ,' ',COUNT(name) from Employee where salary * months = (select MAX(salary*months) from
employee) group by (salary * months);
```

Question 34:

Given the CITY and COUNTRY tables, query the sum of the populations of all cities where the CONTINENT is 'Asia'. Note: CITY.CountryCode and COUNTRY.Code are matching key columns.

```sql
SELECT SUM(CITY.POPULATION) AS POPULATION FROM CITY INNER JOIN COUNTRY ON CITY.COUNTRYCODE =
COUNTRY.CODE WHERE CONTINENT = "Asia";
```

Question 35:

Given the CITY and COUNTRY tables, query the names of all cities where the CONTINENT is 'Africa'.

```sql
select city.name from city inner join country on countrycode = country.code where country.continent = 'Africa'
```

Question 36:

Given the CITY and COUNTRY tables, query the names of all the continents (COUNTRY.Continent) and their respective average city populations (CITY.Population) rounded down to the nearest integer.

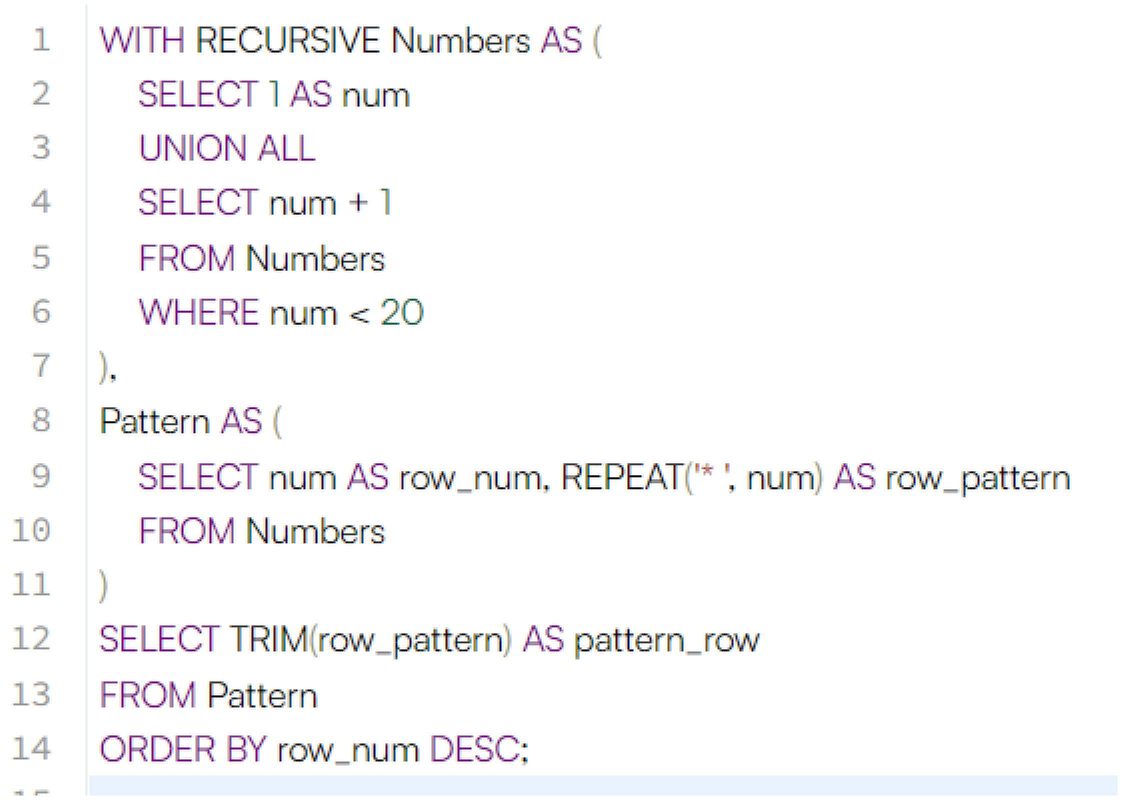
```sql
SELECT COUNTRY.CONTINENT, FLOOR(AVG(CITY.POPULATION)) FROM COUNTRY INNER JOIN CITY ON
CITY.COUNTRYCODE = COUNTRY.CODE GROUP BY COUNTRY.CONTINENT;
```

Question 37:

P(R) represents a pattern drawn by Julia in R rows. The following pattern represents P(5):

```
* * * * *

* * * *

* * *

* *

*
```

Write a query to print the pattern P(20).



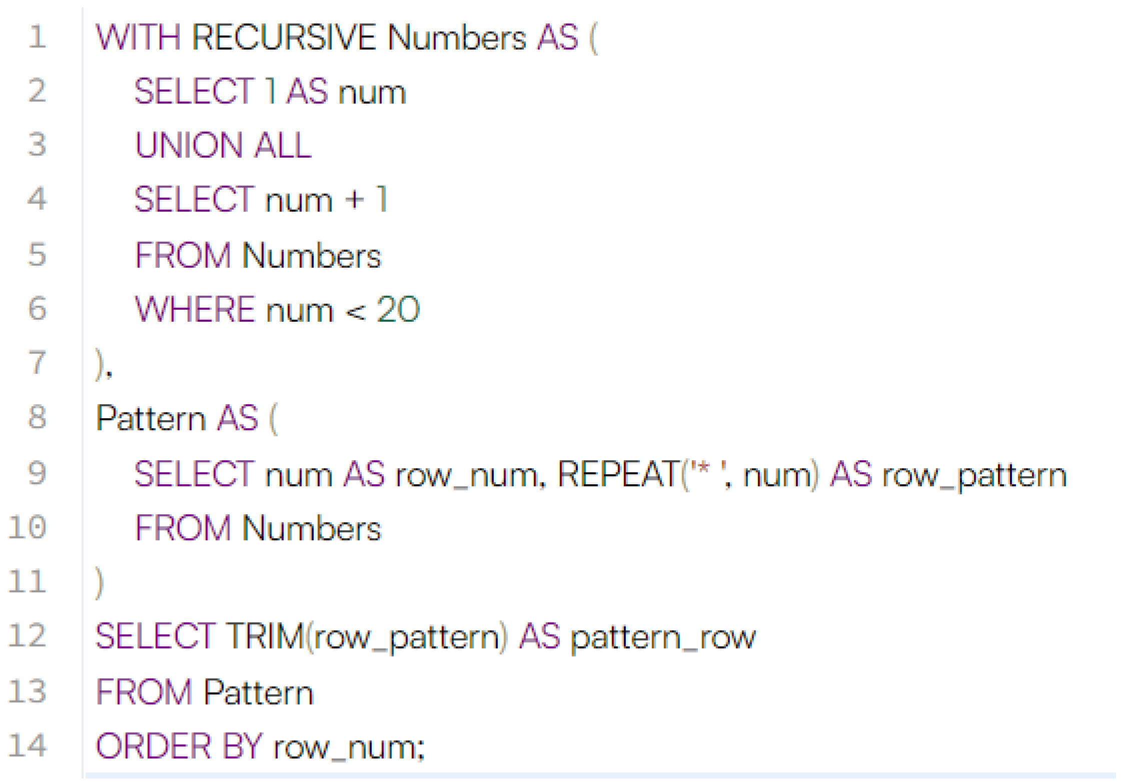
```sql
WITH RECURSIVE Numbers AS (
    SELECT 1 AS num
    UNION ALL
    SELECT num + 1
    FROM Numbers
    WHERE num < 20
),
Pattern AS (
    SELECT num AS row_num, REPEAT('* ', num) AS row_pattern
    FROM Numbers
)
SELECT TRIM(row_pattern) AS pattern_row
FROM Pattern
ORDER BY row_num DESC;
```

Question 38:

P(R) represents a pattern drawn by Julia in R rows. The following pattern represents P(5):

```
*

* *

* * *

* * * *

* * * * *
```

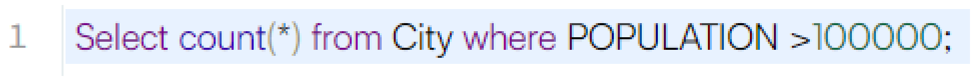
Write a query to print the pattern P(20).



```
WITH RECURSIVE Numbers AS (
    SELECT 1 AS num
    UNION ALL
    SELECT num + 1
    FROM Numbers
    WHERE num < 20
),
Pattern AS (
    SELECT num AS row_num, REPEAT('* ', num) AS row_pattern
    FROM Numbers
)
SELECT TRIM(row_pattern) AS pattern_row
FROM Pattern
ORDER BY row_num;
```

Question 39:

Query a count of the number of cities in CITY having a Population larger than 100,000.



```sql
Select count(*) from City where POPULATION >100000;
```