SQL - Advanced - Part 2

Print Prime Numbers

https://www.hackerrank.com/challenges/print-prime-numbers/problem?isFullScreen=true

Query to print 2 & 3 & 5 & 7 ...

To check whether number is prime or not :- python logics

Iteration from 2 to n/2 : → also can check upto $\sqrt{n}$

Time Complexity: O(sqrt(n))
Auxiliary Space: O(1)

```
if num >2:
    for i in range (2, int (num /2) +1):
        if (num % i) == 0:
            print (num, " is not a prime number")
            break
    else:
        print (num , " is a prime number")
else:
    print ( num, " is not a prime number")
```

Through Recursion

int (sqrt(num)+1)

Time Complexity: O(sqrt(n))
Auxiliary space O(sqrt(n))

```
def Prime (number, itr):
    if itr == 1:          # base condn
        return True
    if number % itr == 0:
        return false
    if Prime (number, itr-1) == false):
        return False
    return True
```

CREATE PROCEDURE procedure_name
AS
sql- statement
GO;

→ Stored procedure you can save code and use it again

EXEC procedure_name; → used to execute this code.

Note: that you may also need to store multiple parameters in one code. just list them down as in example shown below.

SAN:-

```
1  DELIMITER //
2
3  CREATE PROCEDURE find_primes(IN upper_limit INT)
4  BEGIN
5      DECLARE num INT DEFAULT 2;
6      DECLARE i INT;
7      DECLARE is_prime TINYINT;
8      DECLARE primes_list VARCHAR(16383) DEFAULT '';
9
10     prime_loop: WHILE num <= upper_limit DO
11         SET i = 2;
12         SET is_prime = 1; -- TRUE
13
14         divisor_loop: WHILE i <= SQRT(num) DO
15             IF num % i = 0 THEN
16                 SET is_prime = 0; -- FALSE
17                 LEAVE divisor_loop;
18             END IF;
19             SET i = i + 1;
20         END WHILE;
21
22         IF is_prime = 1 THEN
23             IF LENGTH(primes_list) > 0 THEN
24                 SET primes_list = CONCAT(primes_list, '&', num);
25             ELSE
26                 SET primes_list = CAST(num AS CHAR);
27             END IF;
28         END IF;
29
30         SET num = num + 1;
31     END WHILE;
32
33     SELECT primes_list;
34 END //
35
36 DELIMITER ;
37
38 CALL find_primes(1000); -- This will find primes between 2 and 1000
```

input
procedure name
as type int

→ we would need ; in procedure so we need to change delimiter from ";" to "//"

→ starting num

→ flag whether to indicate num prime or not

→ list of concatened prime numbers

→ 2 to upper limit

→ This loop checks if num is prime number or not

→ if num is divisible by i then it is not prime → flag=0
break

→ counter increase

→ while loop finishes.

→ now if it is prime

→ (List >0)

→ Concat num to primes-list

→ if its a first one then change type of number to the character.

→ next number

→ list down all the prime numbers.

→ call the procedure.

15 days of Learning SQL

https://www.hackerrank.com/challenges/15-days-of-learning-sql/problem?isFullScreen=true

condns:- [There is some problem with this question]

* Total number of unique hackers who made atleast one submission.
  - hacker_id and hacker_name of the maximum submissions each day.
    - If more than one such hackers → print lowest hacker_id
  - Is sorted by date

Hackers:
hacker_id
name

Submissions:-
submission_date
submission_id
hacker_id
score

```sql
SELECT
  s1.submission_date,
  (
    SELECT COUNT(DISTINCT s2.hacker_id)
    FROM Submissions s2
    WHERE s2.submission_date = s1.submission_date
    AND (
      SELECT COUNT(DISTINCT s3.submission_date)
      FROM Submissions s3
      WHERE s3.hacker_id = s2.hacker_id
      AND s3.submission_date < s1.submission_date
    ) = DATEDIFF(s1.submission_date, '2016-03-01')
  ) AS total_unique_hackers,
  (
    SELECT s2.hacker_id
    FROM Submissions s2
    WHERE s2.submission_date = s1.submission_date
    GROUP BY s2.hacker_id
    ORDER BY COUNT(s2.submission_id) DESC, s2.hacker_id ASC
    LIMIT 1
  ) AS hacker_id,
  (
    SELECT h.name
    FROM Hackers h
    WHERE h.hacker_id = (
      SELECT s2.hacker_id
      FROM Submissions s2
      WHERE s2.submission_date = s1.submission_date
      GROUP BY s2.hacker_id
      ORDER BY COUNT(s2.submission_id) DESC, s2.hacker_id ASC
      LIMIT 1
    )
  ) AS hacker_name
FROM (
  SELECT DISTINCT submission_date
  FROM Submissions
) s1
GROUP BY s1.submission_date
ORDER BY s1.submission_date;
```

→ total unique hackers.

→ Second subquery for hackers with max submissions.

→ Hackers name

→ s1 subquery for unique submission dates