CS315: Principles of Database Systems, IIT Kanpur				**Midsem** (17 Sep 2024)
Name	DEEBO			**40 marks** Page **1** of **4**
Roll No	240007	Dept.	AWSM	

Instructions:

1. This question paper contains 2 pages (4 sides of paper). Please verify.
2. Write your name, roll number, department above in **block letters neatly with ink**.
3. Write your final answers neatly **with a blue/black pen**. Pencil marks may get smudged.
4. Don't overwrite/scratch answers especially in MCQ – ambiguous cases may get 0 marks.
5. **Hardcoding attempts will not get any credit**.
6. **Be extremely precise in your answers and be careful not to make spelling or punctuation mistakes**. We may type your answers as SQLite queries to actual DB and give marks based on how correct the retrieved results are.

(DBs can do Math!) Deebo has an SQLite table `mth` with 4096 rows. The first column `num` contains integers between 1 and 4096 (both included). Each number occurs exactly once – no duplicates or missing numbers – but the numbers are not in sorted order. The second column `fav` has integers that are 0 or 1 indicating if that number is Deebo's favourite or not (1 = favourite, 0 = not). **Note: SQLite supports modular arithmetic** – if a, b are integers, then the expression $a \% b$ (or $\mathrm{mod}(a, b)$) will give the remainder of a when divided by b.

mth

num	fav
1	0
1729	1
42	1
…	
2607	0

Q1. Write an SQLite query to retrieve all even numbers from `num` sorted in ascending order. Your result should have a single column. **Using the mod operator will incur a 1 mark penalty**.(**3 marks**)

Without Penalty (two options)

```
SELECT m1.num
FROM mth AS m1, mth AS m2
WHERE m1.num = 2*m2.num
ORDER BY m1.num ASC;
```

```
SELECT 2 * num AS n
FROM mth
WHERE 2 * num IN (
   SELECT num from mth
)
ORDER BY n ASC;
```

With Penalty

```
SELECT num
FROM mth
WHERE num % 2 = 0
ORDER BY num ASC;
```

Q2. Write a query to retrieve all primes from `num` sorted in descending order (2 is a prime 1 is not). Your result should have a single column. **Using mod operator will incur 1 mark penalty**.(**4 marks**)

Without Penalty

```
SELECT m0.num FROM mth AS m0
WHERE m0.num <> 1
EXCEPT
SELECT m1.num * m2.num
FROM mth AS m1, mth AS m2
WHERE m1.num > 1 AND m2.num > 1
ORDER BY m0.num DESC;
```

With Penalty

```
SELECT m0.num FROM mth AS m0
WHERE m0.num <> 1
  AND NOT EXISTS (
    SELECT m1.num
    FROM mth AS m1
    WHERE m1.num > 1
      AND m0.num > m1.num
      AND m0.num % m1.num = 0
)
ORDER BY m0.num DESC;
```

Q3. For each value n in the `num` column, count how many numbers $\leq n$ are Deebo's favourite using an SQLite query. Your result should have two columns – the first containing values from the `num` column sorted in descending order and the second containing the favourite counts.(**5 marks**)

```
SELECT m1.num, SUM( m2.fav )
FROM mth AS m1, mth AS m2
WHERE m1.num >= m2.num
GROUP BY m1.num
ORDER BY m1.num DESC;
```

There is a more elegant (and faster) solution by using the *window function* feature supported by recent versions of SQLite: https://www.sqlite.org/windowfunctions.html. See this thread too https://stackoverflow.com/questions/5606560/how-do-i-calculate-a-running-sum

```
WITH tmp AS (
        SELECT * FROM mth
        ORDER BY num ASC
)
SELECT num, SUM( fav ) OVER ( ROWS UNBOUNDED PRECEDING )
FROM tmp ORDER BY num DESC;
```

For Q4,5,6,7, assume that the results of Q1, Q2 are available in views named `even` ***and*** `prime`***. Both views contain a single column containing all even numbers and primes respectively, sorted in ascending and descending order respectively. You may use these views to shorten your queries.***

Q4. Let's verify Goldbach's conjecture – *Every even number greater than 2 is the sum of two primes*. Write a query to retrieve 3 columns n, p, q. $n > 2$ should take even values from `num`, p, q must be primes with $p \leq q, n = p + q$. If n is a prime sum in multiple ways e.g. $14 = 3 + 11 = 7 + 7$, then there should be those many rows for n. If $p = q$, don't create cloned rows e.g. for 14, there should be only 2 rows $(14,3,11), (14,7,7)$, not 3 rows $(14,3,11), (14,7,7), (14,7,7)$. Sort results by n asc. If n has many rows then sort those by p asc e.g. $(14,3,11)$ comes just before $(14,7,7)$. (**5 marks**)

```
SELECT p1.num + p2.num AS n, p1.num AS p, p2.num AS q
FROM prime AS p1, prime AS p2
WHERE p1.num <= p2.num
AND n IN even
ORDER BY n ASC, p ASC;
```

For Q4,5,6,7 it is implicit that the name of the (only) column of the views `even` and `prime` is `num`. However, the use of any other name to refer to the column is allowed too. Also, no marks would be deducted if the final response has column names other than n, p, q.

For Q4,5,6,7, no marks would be deducted for the use of % operator as the penalty was mentioned only for Q1 and Q2.

Warning: *joining views can be super slow if caching is not proper or if the system is running low on memory or disk space. In such cases, converting the view to an actual table really helps*.

Q5. Create a view `succ` with 3 columns – the first with values n from `num` in ascending order, the second with the successor of n if it exists in `num` and `null` otherwise and the third containing the successor of n if it exists in `num` and is also Deebo's favourite and `null` otherwise. **(6 marks)**

```
WITH
  fvrt AS ( SELECT num FROM mth WHERE fav = 1 ),
  valid AS ( SELECT num from mth )

SELECT num AS n, num + 1 AS s, num + 1 AS f
FROM mth WHERE num + 1 IN fvrt

UNION

SELECT num AS n, num + 1 AS s, NULL AS f
FROM mth WHERE num + 1 IN valid AND num + 1 NOT IN fvrt

UNION

SELECT num AS n, NULL AS s, NULL AS f
FROM mth WHERE num + 1 NOT IN valid

ORDER BY n ASC;
```

Q6. Fill one box, give brief justification. Assume `succ` is a table, not a view. Deebo dislikes at least the numbers 1 and 2607, maybe others too. PK ≡ PRIMARY KEY, U ≡ UNIQUE **(3 x (1+1) = 6 marks)**

Can the first column of the `succ` table become a PRIMARY KEY or satisfy UNIQUE constraint?

- [ ] Only PK (not U)
- [ ] Only U (not PK)
- [x] Both PK and U
- [ ] Neither PK nor U

Give justification here

The first column is never NULL and takes unique values hence is eligible to be both PK and U. Note that SQLite allows a column to be both PK and U simultaneously (although such a specification is redundant).

Can the second column of the `succ` table become a PRIMARY KEY or satisfy UNIQUE constraint?

- [ ] Only PK (not U)
- [x] Only U (not PK)
- [ ] Both PK and U
- [ ] Neither PK nor U

Give justification here

The second column will have a single NULL value so it cannot be PK but can be U. Note that SQLite allows PK to be NULL in special circumstances https://www.sqlite.org/lang_createtable.html#the_primary_key

Can the third column of the `succ` table become a PRIMARY KEY or satisfy UNIQUE constraint?

- [ ] Only PK (not U)
- [x] Only U (not PK)
- [ ] Both PK and U
- [x] Neither PK nor U

The third column will have multiple NULL values so it cannot be PK. It shouldn't have been U either but SQLite is weird: it considers NULLs to be distinct. Thus, marks will be given for either option 2 or option 4. https://www.sqlite.org/lang_createtable.html#unique_constraints

Q7. Write an SQLite query to retrieve a bitmap index for primes. Your result should have 2 columns, the first having values from `num` sorted in descending order and the second containing only 0 or 1 depending on whether the number is prime or not (prime ⇒ 1, not prime ⇒ 0). **(5 marks)**

```
SELECT num AS n, 0
FROM mth
WHERE num NOT IN prime

UNION

SELECT num AS n, 1
FROM mth
WHERE num IN prime

ORDER BY n DESC;
```

Q8. Dooba has written a relational expression to find Deebo's favourite perfect squares from `num` i.e. n s.t. $n = m^2$ for some m and n is a favourite. ⋈ without a θ expression does a natural join.

$$\pi_{\text{M1.num}}\left(\sigma_{(\text{M1.num}=\text{M2.num}*\text{M2.num})\vee(\text{M2.fav}=1)}\left(\rho_{\text{M1}}(\text{mth}) \bowtie \rho_{\text{M2}}(\text{mth})\right)\right)$$

Deebo suspects that Dooba's expression will not give the output as intended. Help Deebo make all corrections to the expression by filling the dashed boxes. Using your corrected expression, write an SQLite query to retrieve all favourite perfect squares sorted in ascending order. (**4+2=6 marks**)

$$\pi_{\text{M1.num}}\left(\sigma_{(\quad (\text{M1.num}=\text{M2.num}*\text{M2.num})\wedge(\text{M1.fav}=1) \quad)}\left(\rho_{\text{M1}}(\text{mth}) \quad \times \quad \rho_{\text{M2}}(\text{mth})\right)\right)$$

SQLite query

There may be other ways to correct the expression, but one set of corrections is the following:

1. Change the OR operator ∨ to an AND operator ∧ as we need favorite **and** perfect square
2. Change the second clause to M1.fav=1 since we need n to be favorite and not m
3. Change the natural join ⋈ to a cross join × to allow all pairs to be compared

```
SELECT M1.num AS n
FROM mth AS M1, mth AS M2
WHERE M1.num = M2.num * M2.num AND M1.fav = 1
ORDER BY n ASC;
```

Doing an explicit CROSS JOIN will give the same result but may be slower at execution.