CSE 344 Homework 1

Name: Ananya Shreya Soni

Task 1

1. (SQL) This problem gives you practice with the basics of SQL.

   (a) Write a SQL statement that creates a table edges with two columns source and destination that are both integers.

   ```
   CREATE TABLE edges (
       source INT,
       destination INT
   );
   ```

   (b) Write a single SQL statement that adds these four tuples to edges: (11, 6), (2, 26), (2, 4), and (5, 5).

   ```
   INSERT INTO edges VALUES (11, 6), (2, 26),
       (2, 4), (5, 5);
   ```

   (c) Write a SQL query that returns all tuples in edges. What order do the output rows appear to be printed in?

   ```
   SELECT * FROM edges;
   ```

   The output rows appear in the same order they were inserted in.

   (d) Write a SQL query that returns only the source column for each tuple in edges.

   ```
   SELECT source FROM edges;
   ```

   (e) Write a SQL query that returns only the source column for each tuple in edges. Do not include duplicate elements.

```
SELECT DISTINCT source FROM edges;
```

(f) Write a SQL query that returns, for each tuples in edges, the value of source plus the value of destination.

```
SELECT source + destination FROM edges;
```

(g) Write a SQL query that returns all tuples of edges where source is less than destination. Sort the output by destination.

```
SELECT * FROM edges
    WHERE source < destination
    ORDER BY destination;
```

(h) The next few subproblems explore SQLite's dynamic typing, which is very atypical compared to other RDBMSes. You may find the documentation on datatypes in SQLite useful. Write a SQL statement to try to insert the tuple ('17', 'hello') into edges.

```
INSERT INTO edges VALUES ('17', 'hello');
```

i. According to the relational model, your statement should cause an error, because neither field of the to-be-inserted tuple matches the correct type. What happens in SQLite instead?

> Instead of causing an error SQLite allows you to insert the tuple even the though the types are incorrect because SQLite uses something called type affinity which means the type of a column is recommended, not required. So a column can still store any type of data.

ii. Read the documentation for the built-in typeof function in SQLite. Then, run this query:

```
SELECT source, destination, typeof(source),
    typeof(destination) FROM edges;
```

Paste the output of that query into your answers. Then explain the output (especially the output that corresponds the tuple you inserted in the previous part of this problem)

source	destination	typeof(source)	typeof(destination)
11	6	integer	integer
2	26	integer	integer
2	4	integer	integer
5	5	integer	integer
17	hello	integer	text

The last tuple inserted ('17', 'hello') is shown in the last row of the output of the query since it was successfully added to the edges table despite the types of each field not matching the type of the corresponding attributes due to integer affinity. '17' is converted to an integer 17 since it can easily be converted to an integer and 'hello' remains a text string. This is okay in SQLite because the type of a column in SQLite is recommended, not required.

Task 2

1. (SQL) Use booleans and dates in SQLite:

   (a) Write an SQL statement to create a table called `my_restaurants` with the following attributes (you can pick your own names for the attributes, just make sure it is clear which one is for which):

      i. Name of the restaurant: a string
      ii. Type of food they make: a string
      iii. Distance (in minutes) from your house: an int
      iv. Date of your last visit: a string, interpreted as date
      v. Whether you like it or not: an int, interpreted as a boolean

```
CREATE TABLE my_restaurants (
    name TEXT,
    type_of_food TEXT,
    distance_away INT,
    date_last_visited TEXT,
    liked_food INT
);
```

   (b) Write some number of INSERT statements to add at least five restaurants to the table, subject to the following requirements:

      i. you like at least one restaurant
      ii. you dislike at least one restaurant
      iii. at least one restaurant has NULL in the field indicating whether you like it or not

```
INSERT INTO my_restaurants VALUES
    ('Inchin Bamboo Garden', 'Indo-Chinese',
        50, '2024-06-29', 1),
    ('Mayuri', 'Indo-Chinese', 180,
        '2019-04-31', 1),
    ('Chipotle', 'Mexican', 5, '2025-01-10',
        1),
    ('Anjapur', 'South-Indian', 90,
        '2023-05-06', 0),
    ('Bai-Tong', 'Thai', 75, '2024-09-25', 0),
    ('Olive Garden', 'Italian', 55,
        '2025-01-01', NULL);
```

name	type_of_food	distance_away	date_last_visited	liked_food
Inchin Bamboo Garden	Indo-Chinese	50	2024-06-29	1
Mayuri	Indo-Chinese	180	2019-04-31	1
Chipotle	Mexican	5	2025-01-10	1
Anjapur	South-Indian	90	2023-05-06	0
Bai-Tong	Thai	75	2024-09-25	0
Olive Garden	Italian	55	2025-01-01	NULL

(c) Write a SQL query that returns all attributes of all restaurants that you like, but have not visited for more than 3 months (not including exactly 3 months). Your query must work correctly tomorrow. (In other words, do not hard-code the date of "3 months ago". Instead, use the date() function.)

```
SELECT * FROM my_restaurants
    WHERE date('now', '-3 month') >
        date_last_visited AND liked_food = 1;
```

name	type_of_food	distance_away	date_last_visited	liked_food
Inchin Bamboo Garden	Indo-Chinese	50	2024-06-29	1
Mayuri	Indo-Chinese	180	2019-04-31	1

Task 3

Let L_2 be an arbitrary list.
Let $P(L_1)$ be $\text{count}(x, \text{concat}(L_1, L_2)) = \text{count}(x, L_1) + \text{count}(x, L_2)$.
We will prove $P(L_1)$ for all $L_1 \in \text{List}$ by structural induction.

Base Case (nil):

$$\begin{aligned}\text{count}(x, \text{concat}(\text{nil}, L_2)) &= \text{count}(x, L_2) \quad \text{(by definition of concat)} \\ &= 0 + \text{count}(x, L_2) \quad \text{(by algebra)} \\ &= \text{count}(x, \text{nil}) + \text{count}(x, L_2) \quad \text{(by definition of count)}\end{aligned}$$

Inductive Hypothesis (IH):
Assume $P(L_1)$ is true for some arbitrary $L_1 \in \text{List}$, i.e.,

$$\text{count}(x, \text{concat}(L_1, L_2)) = \text{count}(x, L_1) + \text{count}(x, L_2)$$

Inductive Step (IS):
Let $a \in Z$ be arbitrary. Show $P(a :: L_1)$ holds:

$$\text{count}(x, \text{concat}(a :: L_1, L_2)) = \text{count}(x, a :: \text{concat}(L_1, L_2)) \quad \text{(by definition of concat)}$$

Case 1: $a = x$

$$\begin{aligned}&= 1 + \text{count}(x, \text{concat}(L_1, L_2)) \quad \text{(by definition of count)} \\ &= 1 + \text{count}(x, L_1) + \text{count}(x, L_2) \quad \text{(by IH)}\end{aligned}$$

Case 2: $a \neq x$

$$\begin{aligned}&= \text{count}(x, \text{concat}(L_1, L_2)) \quad \text{(by definition of count)} \\ &= \text{count}(x, L_1) + \text{count}(x, L_2) \quad \text{(by IH)}\end{aligned}$$

$$= \text{count}(x, a :: L_1) + \text{count}(x, L_2) \text{ (by cases and the definition of count)}$$

Therefore, by induction, we have shown that the claim holds for all $L_1 \in \text{List}$.
Since L_2 was arbitrary, we have proven the claim.