Quiz 7

Question # 1

Consider the following two tables, that are related by the column MovieID which is unique. Additionally, assume there are no duplicate rows in either of the tables.

Movie Table

MovieID	MovieName
1	Star Wars
2	Sholay
3	The Italian Job

Cinema Table

MovieID	CinemaName	RunForDays
2	Naz Cinema	101
5	Apollo Theater	45

Can you write a query that emulates a full join between the two tables?

Recall that a full join includes all the rows from the two tables. Furthermore, we are assured that neither of the tables has a duplicate row. We'll relax this condition in the next question and see how it affects the outcome. For this question, we can **UNION** the result of the left join of the movie table with the right join of the cinema table to get a full join between the two tables. Note that the **UNION** operator will include the intersecting part/rows between the two tables only once.

```
SELECT MovieID, MovieName FROM Movie
LEFT JOIN Cinema
```

```
WHERE MovieID = MovieID;

UNION

SELECT MovieID, CinemaName FROM Movie
RIGHT JOIN Cinema
WHERE MovieID = MovieID;
```

Question # 2

Consider the following two tables that have duplicate rows. What would be the outcome of the full join?

Table A

aID
3
5
5
7

Table B

bID
1
5
8
8

Theoretically, the correct answer is as follows:

aID	bID
3	NULL
5	5
5	5
7	NULL
NULL	1
NULL	8

NULL	8

Each 5 from Table A matches with the only 5 in Table B and we get two rows of (5, 5) in the resulting table. Also, notice the two 8s from Table B appear in the final result. If we follow the same query pattern for a full join from the previous question, we'll get the following result:

```
SELECT * FROM A
LEFT JOIN B
ON aID = bID

UNION

SELECT * FROM A
RIGHT JOIN B
ON aID = bID;
```

```
mysql> SELECT * FROM A
    -> LEFT JOIN B
    -> ON aID=bID
    ->
    -> UNION
    ->
    -> SELECT * FROM A
    -> RIGHT JOIN B
    -> ON aID=bID;
+------+------+
| aID  | bID  |
+------+------+
|    5 |    5 |
|    3 | NULL |
|    7 | NULL |
| NULL |    1 |
| NULL |    8 |
+------+------+
5 rows in set (0.00 sec)
```

The problem with the above approach is that the **UNION** operator removes duplicates as part of its behavior. Can we use **UNION ALL**? Let's find out

```
SELECT * FROM A
LEFT JOIN B
ON aID = bID

UNION ALL

SELECT * FROM A
RIGHT JOIN B
ON aID = bID;
```

```
mysql> SELECT * FROM A
    -> LEFT JOIN B
    -> ON aID=bID
    ->
    -> UNION ALL
    ->
    -> SELECT * FROM A
    -> RIGHT JOIN B
    -> ON aID=bID;
+------+------+
| aID  | bID  |
+------+------+
|    5 |    5 |
|    5 |    5 |
|    3 | NULL |
|    7 | NULL |
|    5 |    5 |
|    5 |    5 |
| NULL |    1 |
| NULL |    8 |
| NULL |    8 |
+------+------+
9 rows in set (0.00 sec)
```

The problem with using **UNION ALL** is that the intersecting portion between the two tables is printed twice. Once when the two 5s from Table A match the single 5 in Table B and once when the single 5 matches the two 5s in Table A. The right solution is to minus this intersecting portion once from the resulting set. One approach can be to take the left join for Table A and then only take those values from the right join for Table B that have a corresponding NULL entry for the aID column, meaning that such rows are only part of Table B. The complete query is shown below:

```sql
SELECT * FROM A
LEFT JOIN B
ON aID = bID

UNION ALL

SELECT * FROM A
RIGHT JOIN B
ON aID = bID
WHERE aID IS NULL;
```

```
mysql> SELECT * FROM A
    -> LEFT JOIN B
    -> ON aID=bID
    ->
    -> UNION ALL
    ->
    -> SELECT * FROM A
    -> RIGHT JOIN B
    -> ON aID=bID
    -> WHERE aID IS NULL;
+------+------+
| aID  | bID  |
+------+------+
|    5 |    5 |
|    5 |    5 |
|    3 | NULL |
|    7 | NULL |
| NULL |    1 |
| NULL |    8 |
| NULL |    8 |
+------+------+
7 rows in set (0.00 sec)
```

Now, you can observe the practical result matches our theoretical result. The astute reader would realize we can achieve the same result if we used the WHERE clause with the left join and took the entirety of the right join. The query is shown below:

```
SELECT * FROM A
LEFT JOIN B
ON aID = bID
WHERE bID IS NULL

UNION ALL

SELECT * FROM A
RIGHT JOIN B
ON aID = bID;
```

```
mysql> SELECT * FROM A
    -> LEFT JOIN B
    -> ON aID = bID
    -> WHERE bID IS NULL
    ->
    -> UNION ALL
    ->
    -> SELECT * FROM A
    -> RIGHT JOIN B
    -> ON aID = bID;
+------+------+
| aID  | bID  |
+------+------+
|    3 | NULL |
|    7 | NULL |
|    5 |    5 |
|    5 |    5 |
| NULL |    1 |
| NULL |    8 |
| NULL |    8 |
+------+------+
```

```
7 rows in set (0.00 sec)
```

Question # 3

Q

Can a column which isn’t unique or the primary key in a parent act as a foreign key in a child table?

Question # 4

Q

Which of the following is a disadvantage of defining foreign key constraints?

COMPLETED 0%

1 of 1

Question # 5

Q

Which of the following is true about foreign keys?

COMPLETED 0%

1 of 1