Administrative Notes
March 2, 2023

- **March 3**: Assignment 4 is due (individual assignment)
  - The question PDF was updated on both Feb 21 and Feb 28 so please make sure you are looking at the right version
  - Tutorials this week will function as open office hours.
    - Friday tutorial moved online for **this week only**
- **March 3 @ 10PM**: Assignment 5 released (programming assignments)
  - If you are working in a group, sign up here.
- **March 9** (midterm): During class time. Closed book, closed notes, closed neighbor. More details on Piazza (@38).
- Notify Jessica by email ASAP if you have an exam hardship or conflict

CPSC 368
Databases in Data Science

Working with SQL

Learning Goals

- Evaluate the accuracy of given SQL statements for a specified query.
- Determine whether a given query is correctly translated into SQL or not.
- When given a SQL query and a set of relational instances, determine what the SQL query would produce.
- Determine whether two or more SQL queries would produce the same answer for all legal instances of a relation.

General Checklist

- Is the query valid?
  - Are the aggregate operators used properly?
  - Are we selecting values that exist given the relations we are working with?
  - Are the attributes that are being selected in the grouping list?

- Is the query selecting for the right attributes?

- Try to come up with an English description of the query. If you had to use 30 seconds to state what this query is trying to do, what would you say? See if this description matches with the problem statement.

- Test the query with some sample relational instances
  - Boundary cases
  - “Normal” case

Orchestra Database

Person(email, name, age)
- This relation stores anyone who has signed up for our mailing list. Tuples in this relation may not be listed in Purchase.

Show(id, year, month, date, showing, attendanceNumber)
- Showing describes whether a show was during morning, afternoon, or evening

Song(composer, title)

SongsPerformed(**showID**, **composer**, **title**)
- showID is a foreign key referring to Show
- composer and title are foreign keys referring to attributes of the same name in Song

Purchase(**email**, **showID**, price)
- email is a foreign key referring to the email attribute in Person
- showID is a foreign key referring to Show

Musician(id, name, instrument, position, nationality)

PerformedIn(**id**, **showID**)
- id refers to the attribute of the same name in Musician
- showID is a foreign key referring to Show

Task #1

Query: Find the email addresses of everyone who is in the Person table but has not made a purchase before.

Given SQL:

```
SELECT    email
FROM      Purchase
WHERE     EXISTS(
          SELECT email
          FROM Person
);
```

```
SELECT email
FROM   Person
WHERE  NOT EXISTS(
               SELECT email
               FROM Purchase
);
```

Task #1

Query: Find the email addresses of everyone who is in the Person table but has not made a purchase before.

Given SQL:

```
SELECT    email
FROM      Purchase
WHERE     EXISTS(
          SELECT email
          FROM Person
);
```

```
SELECT email
FROM   Person
WHERE  NOT EXISTS(
               SELECT email
               FROM Purchase
);
```

- We need to retrieve information about people from the Person table, not from the Purchase table
- We are looking for those who have NOT made a purchase, so NOT EXISTS should be used instead of EXISTS.

Task #2

Query: Find the total number of shows performed during each year and month. If the query is not valid or correct, fix it.

Given SQL:

```
SELECT DISTINCT year, month, COUNT(date)
FROM Show
```

Task #2

Query: Find the total number of shows performed during each year and month. If it is not valid or correct, fix it.

Given SQL:

```
SELECT DISTINCT year, month, COUNT(date)
FROM Show
```

- The given SQL statement is valid but not correct because it has has an aggregated column (date) while others are not aggregated (year and month)
- In order to count the number of records for each year and month, we need to group the data by those columns, and then count the number of records in each group

```
SELECT      year, month, COUNT(date)
FROM        Show
GROUP BY    year, month
```

Task #3

Query: Find the show IDs of shows where every attendee is older than the average age of all individuals in the Person table.

Given SQL:

```
CREATE VIEW Age(age) AS
    SELECT AVG(age)
    FROM Purchase
    NATURAL JOIN Person
```

```
SELECT showID
FROM Purchase
NATURAL INNER JOIN Person
GROUP BY showID
HAVING AVG(age) >
      (SELECT AVG(age)
       FROM Person)
```

- What are these SQL queries trying to do?

Task #3: Fix

Query: Find the show IDs of shows where every attendee is older than the average age of all individuals in the Person table.

```
SELECT showID
FROM Purchase NATURAL INNER JOIN Person
GROUP BY showID
HAVING MIN(age) > (SELECT AVG(age) FROM Person)
```

Definition of Equivalency

- Two queries are equivalent if they produce the same result across all legal instances

Task #4

- Consider the two queries below meant to find the name of musicians who performed at least 2 songs in each show.
- Are they equivalent?

```
SELECT DISTINCT name
FROM Musician, PerformedIn, SongsPerformed
GROUP BY name
HAVING COUNT(title) > 1
```

```
SELECT name
FROM Musician NATURAL INNER JOIN PerformedIn NATURAL INNER JOIN
SongsPerformed
GROUP BY name, showID
HAVING COUNT(title) > 1
```

Task #4: Take 2!

- Consider the two queries below meant to find the name of musicians who performed at least 2 songs in each show.
- Are they equivalent?

```
SELECT DISTINCT m.name
FROM Musician m, PerformedIn pi, SongsPerformed sp
WHERE m.id = pi.id AND pi.showID = sp.showID
GROUP BY m.name
HAVING COUNT(sp.title) > 1
```

```
SELECT name
FROM Musician NATURAL INNER JOIN PerformedIn NATURAL INNER JOIN
SongsPerformed
GROUP BY name, showID
HAVING COUNT(title) > 1
```

Task #4

Come up with test cases.

- Only performed in 1 show but played 2 songs
- Only performed in 1 show but played 1 song
- Performed in 2 shows and played at least two in each show
- Performed in 2 shows and at least one of the shows didn't meet the song quota

Musician

ID	name
1	Carol
2	Jia
3	Irena
4	Jessica

PerformedIn

ID	showID
1	1
2	2
3	3
3	4
4	3
4	5

SongsPerformed

showID	composer	title
1	A	A
1	B	B
2	C	C
3	D	D
3	E	E
3	F	F
4	A	A
4	B	B
5	C	C

Task #4

```
SELECT DISTINCT m.name
FROM Musician m, PerformedIn pi, SongsPerformed sp
WHERE m.id = pi.id AND pi.showID = sp.showID
GROUP BY m.name
HAVING COUNT(sp.title) > 1
```

Musician

ID	name
1	Carol
2	Jia
3	Irena
4	Jessica

PerformedIn

ID	showID
1	1
2	2
3	3
3	4
4	3
4	5

SongsPerformed

showID	composer	title
1	A	A
1	B	B
2	C	C
3	D	D
3	E	E
3	F	F
4	A	A
4	B	B
5	C	C

Task #4

- Find the name of musicians who performed at least 2 songs in each show.

```
SELECT DISTINCT m.name
FROM Musician m, PerformedIn pi, SongsPerformed sp
WHERE m.id = pi.id AND pi.showID = sp.showID
GROUP BY m.name
HAVING COUNT(sp.title) > 1
```

name
Carol
Irena
Jessica

Task #4

- Find the name of musicians who performed at least 2 songs in each show.

```
SELECT name
FROM Musician NATURAL INNER JOIN PerformedIn NATURAL INNER JOIN
SongsPerformed
GROUP BY name, showID
HAVING COUNT(title) > 1
```

Musician

ID	name
1	Carol
2	Jia
3	Irena
4	Jessica

PerformedIn

ID	showID
1	1
2	2
3	3
3	4
4	3
4	5

SongsPerformed

showID	composer	title
1	A	A
1	B	B
2	C	C
3	D	D
3	E	E
3	F	F
4	A	A
4	B	B
5	C	C

Task #4

- Find the name of musicians who performed at least 2 songs in each show.

```
SELECT name
FROM Musician NATURAL INNER JOIN PerformedIn NATURAL INNER JOIN
SongsPerformed
GROUP BY name, showID
HAVING COUNT(title) > 1
```

name
Carol
Irena
Jessica

Task #4: Fix it!

Pick one of the queries and work with the people around you to fix it such that the query can find the name of musicians who performed at least 2 songs in each show.

Task #4: Fix it! Option 1 (amongst others)

Pick one of the queries and work with the people around you to fix it such that the query can find the name of musicians who performed at least 2 songs in each show.

```
SELECT DISTINCT m.name
FROM Musician m, PerformedIn pi
WHERE 2 <= ALL(SELECT COUNT(*)
               FROM Musician m2, PerformedIn pi2, SongsPerformed sp
               WHERE m2.id = pi2.id AND pi2.showID = sp.showID AND
                     m.id = m2.id
GROUP BY pi.showID, m.name)
```

Task #4: Fix it! Option 2 (amongst others)

Pick one of the queries and work with the people around you to fix it such that the query can find the name of musicians who performed at least 2 songs in each show.

```
CREATE VIEW MusiciansWhoPerformedOneSong(id) AS
SELECT m.id
FROM Musician m, PerformedIn pi, SongsPerformed sp
WHERE m.id = pi.id AND pi.showID = sp.showID
GROUP BY pi.showID, m.name
HAVING COUNT(sp.title) = 1
```

```
SELECT m.name
FROM Musician m NATURAL INNER JOIN PerformedIn pi
WHERE m.id NOT IN (SELECT * FROM MusiciansWhoPerformedOneSong)
```

Task #5

- Consider the query below. If the query is meant to find the title of all songs that have been performed without a pianist, does it accomplish the task?

```
SELECT title
FROM   SongsPerformed
NATURAL JOIN  PerformedIn
WHERE   NOT EXISTS (
       SELECT *
       FROM Musician
       WHERE instrument = “piano”)
```

Task #5

- Consider the query below. If the query is meant to find the title of all songs that have been performed without a pianist, does it accomplish the task?

```
SELECT title
FROM   SongsPerformed
NATURAL JOIN  PerformedIn
WHERE   NOT EXISTS (
       SELECT *
       FROM Musician
       WHERE instrument = “piano”)
```

- The nested query in the NOT EXISTS clause always returns a row unless there are no pianists in the Musician table.
- This query returns the titles of all songs when there are no pianists at all.
- The problem is that nested query is not CORRELATED with the outer query. In other words, it runs independently of the selected song.

Task #5

- Consider the query below. If the query is meant to find the title of all songs that have been performed without a pianist, does it accomplish the task?

```
SELECT DISTINCT title
FROM  SongsPerformed sp NATURAL INNER JOIN PerformedIn pi
WHERE  NOT EXISTS ( SELECT *
                    FROM Musician m
                    WHERE m.id = pi.id AND
                          instrument = “piano”)
```

- When would the inner query be true?
- What relational instances would you use to prove that this query does not work?

Task #5

- Consider the query below. If the query is meant to find the title of all songs that have been performed without a pianist, does it accomplish the task?

```
SELECT DISTINCT title
FROM  SongsPerformed sp NATURAL INNER JOIN PerformedIn pi
WHERE  NOT EXISTS ( SELECT *
                    FROM Musician m
                    WHERE m.id = pi.id AND
                          instrument = “piano”)
```

Musician

ID	name	instrument
1	Carol	violin
2	Jia	cello
3	Jessica	piano

SongsPerformed

showID	composer	title
1	A	A
1	B	B
2	C	C

PerformedIn

ID	showID
1	1
2	1
3	2
1	2

Task #5

- Consider the query below. If the query is meant to find the title of all songs that have been performed without a pianist, does it accomplish the task?

```
SELECT DISTINCT title
FROM  SongsPerformed sp NATURAL INNER JOIN PerformedIn pi
WHERE  NOT EXISTS ( SELECT *
                    FROM Musician m
                    WHERE m.id = pi.id AND
                          instrument = “piano”)
```

Result

title
A
B
C