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 #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 #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?

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 #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”)
```