Here are some relations that exist in a database for an orchestra.

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, day, showing, attendanceNumber)

- Showing describes whether a show was during morning, afternoon, or evening
- {year, month, day, showing} is a candidate key for Show

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

Consider the following query meant to find songs that have been performed by every violinist in the orchestra.

```
SELECT DISTINCT composer, title
FROM SongsPerformed sp1
WHERE NOT EXISTS (SELECT *
                  FROM Musician m
                  WHERE NOT EXISTS (
                        SELECT *
                        FROM PerformedIn pi
                        WHERE pi.id = m.id AND pi.showID = sp1.showID AND
                              m.instrument = 'violin'))
```

Riley believes that this query does not work as intended while Alex believes it does. Provide a relational instance that demonstrates why this query does or does not work. Walk through the execution of the query above using your relational instance to prove your point. Your relational instance does not need to include every attribute (though you certainly can if you wish).