Here are some relations that exist in a database for a symphony.

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

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

Write **Datalog** statements to answer the following questions:

1. Find the year, month, date, and attendance numbers of all shows that had a Canadian musician perform in it.

   Ans_1(Y, M, D, AN) :- Show(SID, Y, M, D, _, AN), Performed(MID,SID),
   Musician(MID, _, _, _, 'Canadian')

2. Find the showIDs of shows where the symphony performed songs by Mozart and Beethoven.

   Ans_2(SID) :- SongsPerformed(SID, 'Mozart', _), SongsPerformed(SID, 'Beethoven, _)

   The different occurrences of SongPerformed are different instances. We are requiring that the SID is the same, but nothing else has to be the same.

3. Find the showIDs of shows where the symphony performed at least one song by Mozart or Beethoven.

   Ans_3(SID) :- SongsPerformed(SID, 'Mozart', _)
   Ans_3(SID) :- SongsPerformed(SID, 'Beethoven, _)

   Unions are denoted by repeating the name of the answer relation. So, by naming both of the returned values "Ans_3", we are saying that the answer is the union of the two values.

4. Find the songs that have been performed by every violinist in the orchestra. You can safely assume that if a musician is performing in a show, they will play every song in that show.

   Witness(C, T, ID) :- SongsPerformed(SID, C, T), PerformedIn(ID, SID), Musician(ID, _, 'violin', _, _) this gets all song violinist combinations
   Bad(C, T) :- Songs(C, T), Musician(ID, _, 'violin', _, _) NOT Witness(C, T, ID) this gets all combinations of songs and violinists and the subtracts the ones we do have in witness therefore getting the songs that are not played by some violinist
   Good(C, T):- Songs(C, T), NOT Bad(C, T) this gets all songs that are not in bad, so all songs that have been played by all violinists

   Note: You can't put a _ in any subgoal that is negated, because otherwise that variable is unbound. Every variable in a negated subgoal must appear in some other subgoal that is not negated. SID must be repeated anyway in order to perform the join with PerformedIn, but this is why we need to include Song(C,T). Otherwise those values are undefined.