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

Write SQL statements to answer the following questions:

1. Find the total number of attendees for each day there was a show. Don't forget that a single day can have multiple shows!

   ```
   SELECT year, month, day, sum(attendanceNumber)
   FROM Show
   GROUP BY year, month, day
   ```

2. Find the total number of Canadian musicians per instrument.

   ```
   SELECT instrument, count(*)
   FROM Musician
   WHERE nationality = 'Canadian'
   GROUP BY Instrument
   ```

3. For all the shows where the audience's average age is greater than the average age of our mailing list, what songs were performed?

```
SELECT DISTINCT sp.composer, sp.title
FROM SongsPerformed sp
WHERE sp.showID in (SELECT pur.showID
                    FROM Person p, Purchase pur
                    WHERE pur.email = p.email
                    GROUP BY pur.showID
                    HAVING AVG(p.age) > (SELECT AVG(p1.age) FROM Person p1))
```

Alternatively:

```
CREATE VIEW ShowsWithHigherAvgAge(showID) AS
SELECT sp.showID
FROM SongsPerformed sp, Person p, Purchase pur
WHERE sp.showID = pur.showID AND p.email = pur.email
GROUP BY sp.showID
HAVING AVG(p.age) > (SELECT AVG(p1.age) FROM Person p1)
```

```
SELECT DISTINCT sp.composer, sp.title
FROM SongsPerformed sp, ShowsWithHigherAvgAge sha
WHERE sp.showID = sha.showID
```