## 1. Print each user's name, along with the number of times they have recorded a location. (1)

```
SELECT User.name, COUNT(latitude)

FROM User LEFT JOIN Location

ON Location.user = User.id

GROUP BY User.id;
```

### NOTE

- "The column selected for display should be named in the GROUP BY clause, or at least functionally-determined by the latter."

### COMMON ERRORS

- INNER instead of LEFT join
- if you use a left join, you have to count non-null values in a column, not rows

## 2. How many cities are in the same state as Melbourne? (Don't count Melbourne in your answer.) (1)

```
SELECT COUNT(*) FROM City

WHERE state =

        (SELECT state FROM City

        WHERE cityName = 'Melbourne')

        AND cityName != 'Melbourne';
```

### COMMON ERRORS

- hard-coding "Vic" instead of using the subquery

## 3. List the names of any members of Academia gym who have been north of Brunswick gym. (1)

```
SELECT User.name
FROM Location JOIN User on Location.user = User.id
JOIN Gym on User.gym = Gym.id
WHERE Gym.name = 'Academia'
AND Location.latitude >
    (SELECT latitude FROM Gym WHERE name = 'Brunswick');
```

### ALTERNATIVES

- can use JOIN instead of subquery

## 4. How many users are registered with gyms in the state of Vic? (1)

```
SELECT count(*) as VicUserCount
FROM User JOIN Gym ON User.gym = Gym.id
JOIN City on Gym.city = City.id
WHERE state = 'Vic';
```

## 5. What percentage of the total number of users are not affiliated with gyms? (1)

```
SELECT
        (SELECT COUNT(*) FROM User WHERE gym IS NULL)
/
        (SELECT COUNT(*) FROM User)
* 100 AS Percent;
```

## 6. How much time elapsed between the first and last recorded locations of the user with id 4? (2)

```
SELECT TIMEDIFF(MAX(whenRecorded), MIN(whenRecorded))
FROM Location
WHERE user = 4;

/*   OR */

SELECT TIMEDIFF(
(SELECT whenRecorded FROM Location
WHERE Location.id =
    (SELECT MAX(id) FROM Location WHERE Location.user = 4)),
(SELECT whenRecorded FROM Location
WHERE Location.id =
    (SELECT MIN(id) FROM Location WHERE Location.user = 4))
) as timediff;
```

### COMMON ERRORS

- joining to User table

7. **Print as two columns: the average number of locations recorded by registered users, and the average number of locations recorded by unregistered users. (3)**

```
SELECT (SELECT AVG(NumLocations) FROM
            (SELECT user, gym, COUNT(*) as NumLocations
            FROM Location JOIN User on Location.user = User.id
            WHERE gym IS NOT NULL
            GROUP BY user) as RegisteredUsers) as RU,
 (SELECT AVG(NumLocations) FROM
            (SELECT user, gym, COUNT(*) as NumLocations
            FROM Location JOIN User on Location.user = User.id
            WHERE gym IS NULL
            GROUP BY user) as UnregisteredUsers) as URU;
```

## ALTERNATIVES

- "average across all users" (INNER JOIN) vs
  "average across those users who have recorded locations" (OUTER JOIN)

## 8. List the names of users who have run within 100m of the Doug McDonell building. (DMD is at longitude 144.9630, latitude -37.7990 .) (3)

```
SELECT DISTINCT User.name

FROM Location JOIN User on Location.user = User.id

WHERE SQRT(POWER(longitude - 144.9630, 2) + POWER(latitude - -37.7990, 2)) * 100 < 0.1;
```

## 9. What is the distance between the northern-most and southern-most locations to which Alice has run? (3)

```
SELECT SQRT( POWER(

        (SELECT longitude FROM Location JOIN User on Location.user = User.id

        WHERE User.name = 'Alice'

        ORDER BY latitude DESC LIMIT 1) -

        (SELECT longitude FROM Location JOIN User on Location.user = User.id

        WHERE User.name = 'Alice'

        ORDER BY latitude ASC LIMIT 1)

        ,2) +

        (SELECT POWER(MAX(latitude) - MIN(latitude),2)

FROM Location JOIN User ON Location.user = User.id

WHERE User.name = 'Alice')) * 100 as Distance;
```

### ALTERNATIVES

- there are TWO northernmost points and TWO southernmost locations – either is fine
- find northern/southern location in a different way, e.g using MAX

### COMMON ERRORS

- choosing northermost / southernmost Latitude and easternmost / westernmost Longitude

## 10. Show the total distance that Alice has run. Calculate this by summing the individual distances between each successive pair of locations. (4)

```
SELECT SUM(SQRT( POW(A.latitude-B.latitude,2) + POW(A.longitude-B.longitude,2) ) *100) AS dist

FROM Location A JOIN Location B ON B.id =

        (SELECT MAX(id) FROM Location

        WHERE id < A.id AND user = A.user)

WHERE A.user =

        (SELECT id FROM User

         WHERE name = 'Alice');
```

### ALTERNATIVES

- using LEAD function and Windowing

### COMMON ERRORS

- hoping that location.id values for Alice are always two integers apart.
  (join condition:    A.id = B.id + 2 )
- hoping that location.whenRecorded timestamps for Alice are always 1 minute apart.
  (join condition:   TIMEDIFF(A.whenRecorded, B.whenRecorded) = '00:01:00')