7. Inserting and deleting rows

You'll need to be able to write data in order to take full advantage of persisting data on Android with Room. In addition to querying a database, there are also SQL statements for inserting, updating, and deleting rows. You'll need a basic knowledge of these when you learn to write data with Room later in Pathway 2.

INSERT statement

To add a new row, you use the `INSERT` statement. The `INSERT` statement is followed by the `INTO` keyword and the name of the table in which you'd like to add a row. After the VALUES keyword, you provide the value for each column (in order) in parentheses, with each one separated by a comma. The format of an `INSERT` statement is as follows.

```
INSERT INTO table_name
VALUES (column1, column2, ...)
```

To add a row to the `park` table, the `INSERT` statement would look something like this. The values match the order in which the columns are defined for the `park` table. Notice that some of the data is not specified. That's OK for now, as you can always update a row after it's been inserted.

```
INSERT INTO park
VALUES (null, 'Googleplex', 'Mountain View', 12, null, 0, '')
```

Also notice that you pass in `null` for the ID. While you can provide a specific number, this isn't exactly convenient as your app would have to keep track of the latest ID to make sure there are no duplicates. You can, however, configure your database so that the primary key is automatically incremented, which was done here. That way you can pass in `null`, and the next ID is chosen automatically.

Verify that the entry was created, using a `WHERE` clause to specify the park named `"Googleplex"`.

```
SELECT * FROM park
WHERE name = 'Googleplex'
```

UPDATE statement

After a row has been created, you can change its contents at any time. You can do so using an `UPDATE` statement. Like all the other SQL statements you've seen, you first need to specify the table name. In the `SET` clause, simply set each column you want to change to its new value.

```
UPDATE table_name
SET column1 = ...,
column2 = ...,
...
WHERE column_name = ...
...
```

For the Googleplex entry, one existing property is updated, and some other fields are filled in (these fields previously had a value but it was an empty string, `""`). You can update multiple (or all) fields at once with an `UPDATE` statement.

```
UPDATE park
SET area_acres = 46,
established = 1088640000,
type = 'office'
WHERE name = 'Googleplex'
```

See the updates reflected in the query results

```
SELECT * FROM park
WHERE name = 'Googleplex'
```

DELETE Statement

Finally, you can also use a SQL command to delete rows from the database. Again, specify the table name, and just like you did with `SELECT` statements, you use a `WHERE` clause to provide criteria for the rows you want to delete. Since a `WHERE` clause can match multiple rows, you can delete multiple rows with a single command.

```
DELETE FROM table_name
WHERE <column_name> = ...
```

Because the Googleplex isn't a national park, try using a `DELETE` statement to remove this entry from the database.

```
DELETE FROM park
WHERE name = 'Googleplex'
```

Verify to make sure the row is deleted using a `SELECT` statement. The query should return no results, meaning all the rows that had the name "Googleplex" were successfully deleted.

```
SELECT * FROM park
WHERE name = 'Googleplex'
```

That's all there is to inserting, updating, and deleting data. All you need to know is the format for the SQL command you want to perform, and specify values that match the