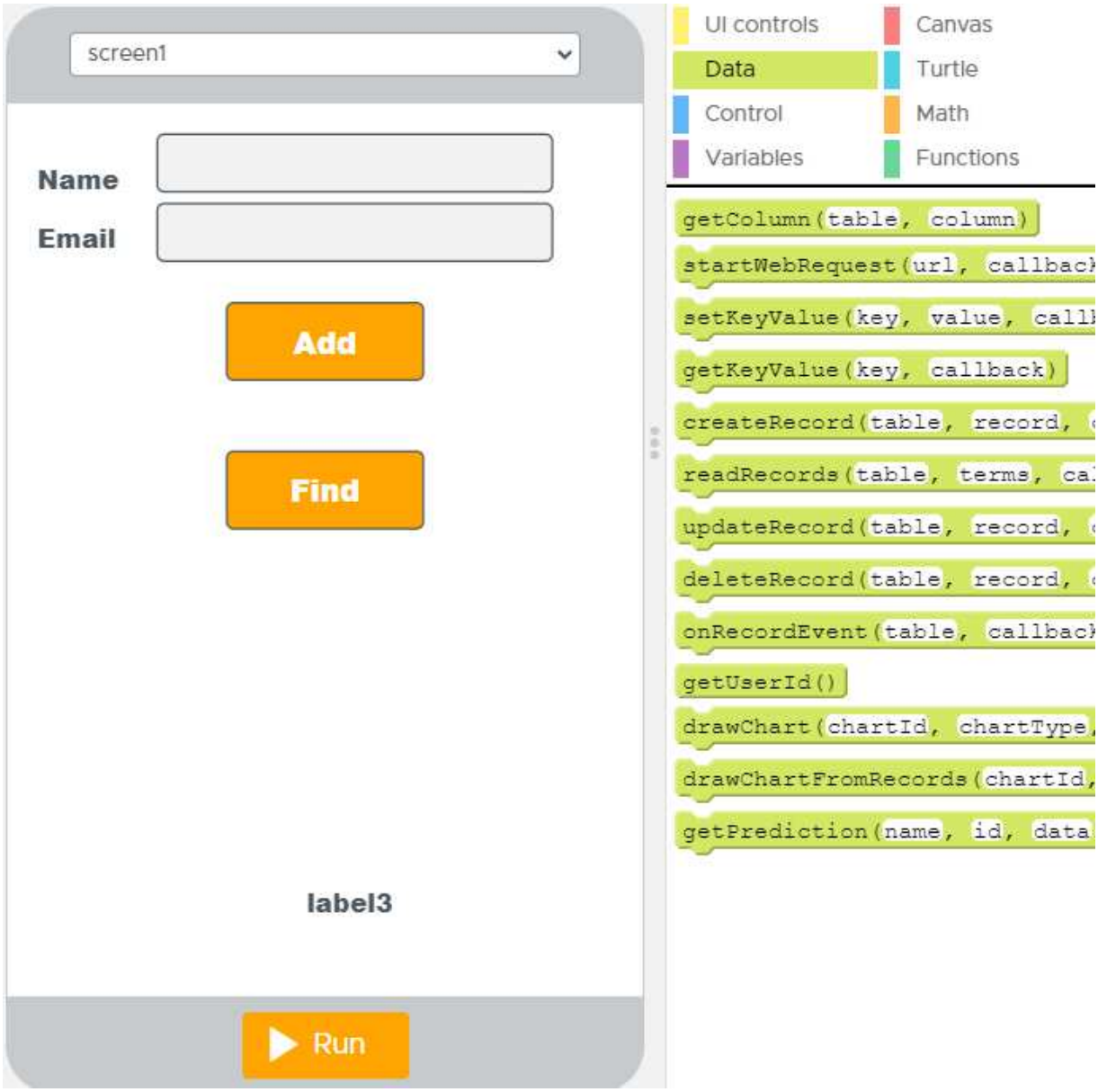
Database

1. Introduction

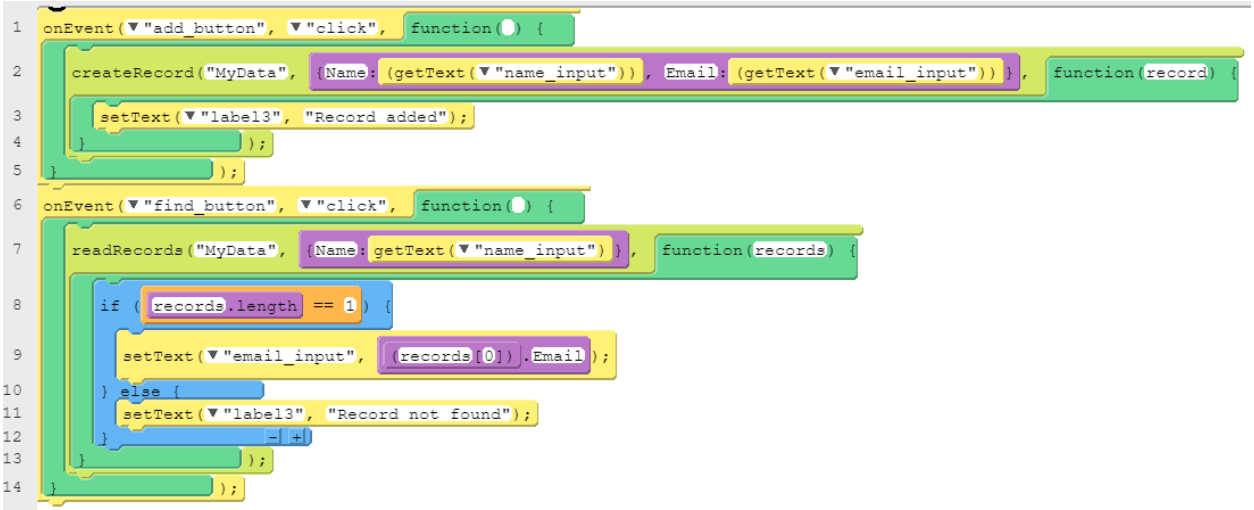
Unlike variables, databases allow a program to store and retain lots of information. The information is stored indefinitely until it is specifically deleted.

2. User Interface

Create the following user interface with three labels, two text input boxes, and two buttons. The ID names for the two text input boxes and buttons are *name_input*, *email_input*, *add_but* and *find_but*.

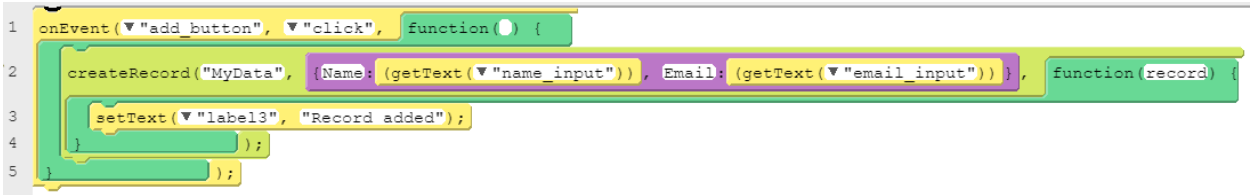


Write the following code. There are two **onEvents**, one for each of the button click.

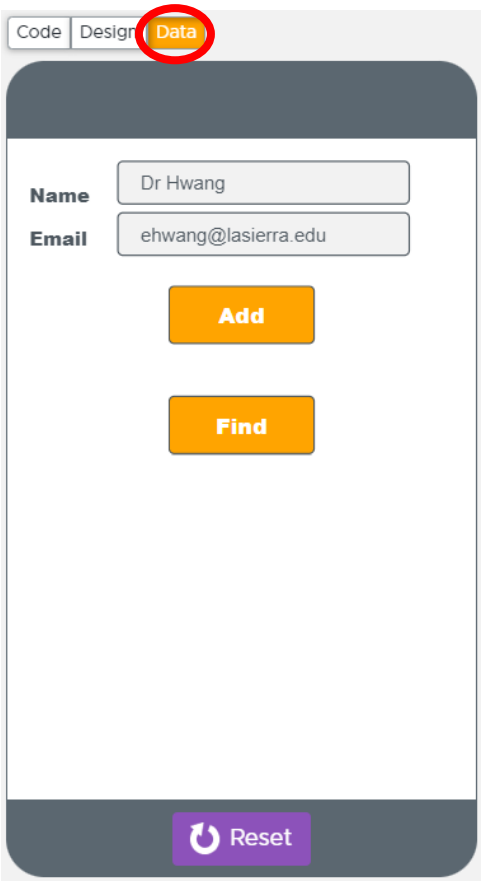


3. Add Record

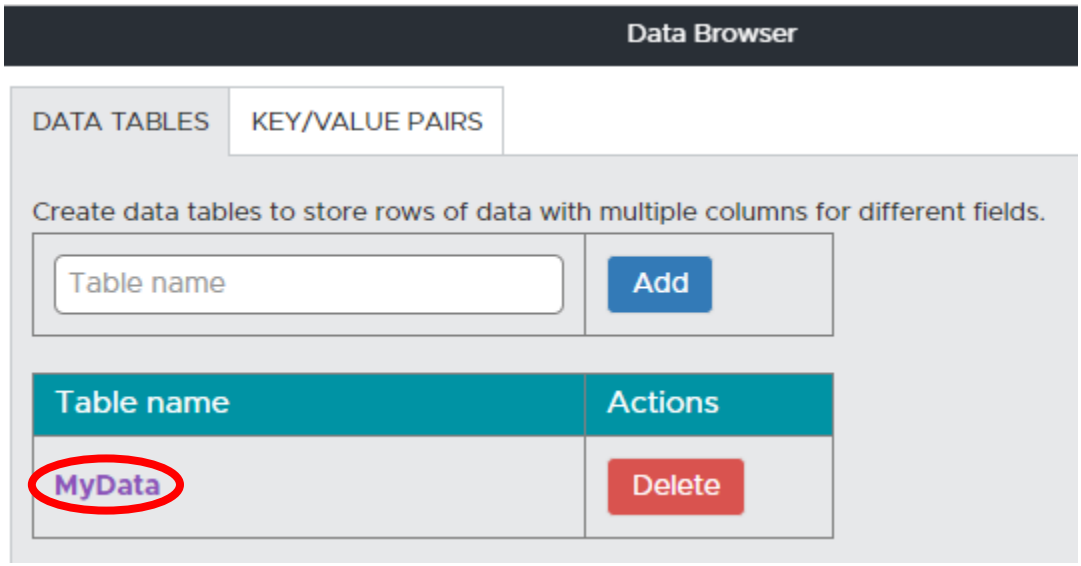
When the add button is clicked, it will execute the **createRecord** block inside the onEvents for the add button click. The **createRecord** command creates and adds a new record into the database table named *MyData* with the two fields *Name* and *Email*. The data for the *Name* and *Email* fields are obtained from the *name_input* and *email_input* text input boxes using the **getText** blocks.



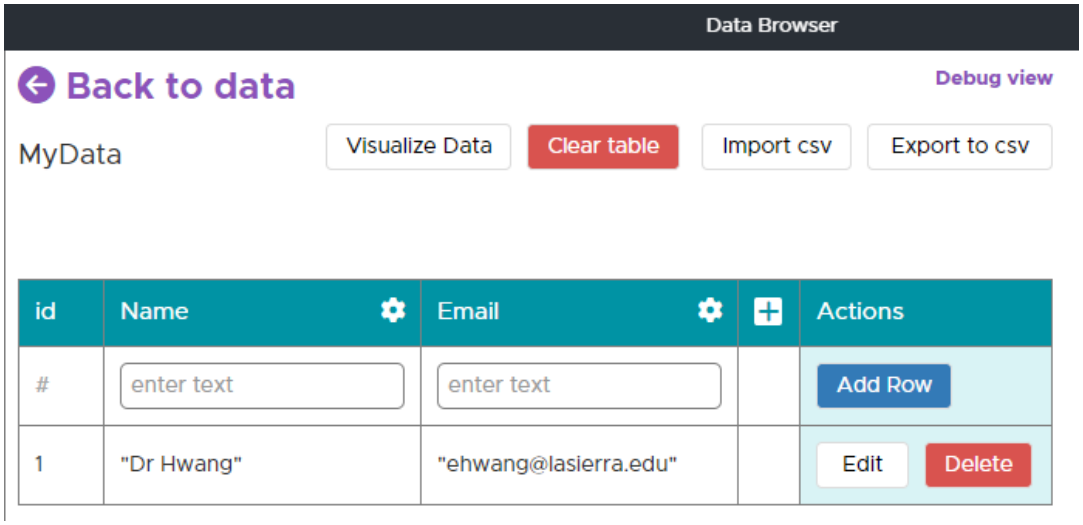

```
createRecord("MyData", {Name:(getText("name_input")), Email:(getText("email_input"))},
    function(record) {

    });
```



Select the Data tab. In the Data Browser window pane you will see the table *MyData*. Click on the *MyData* table name to see the data stored inside that database table.

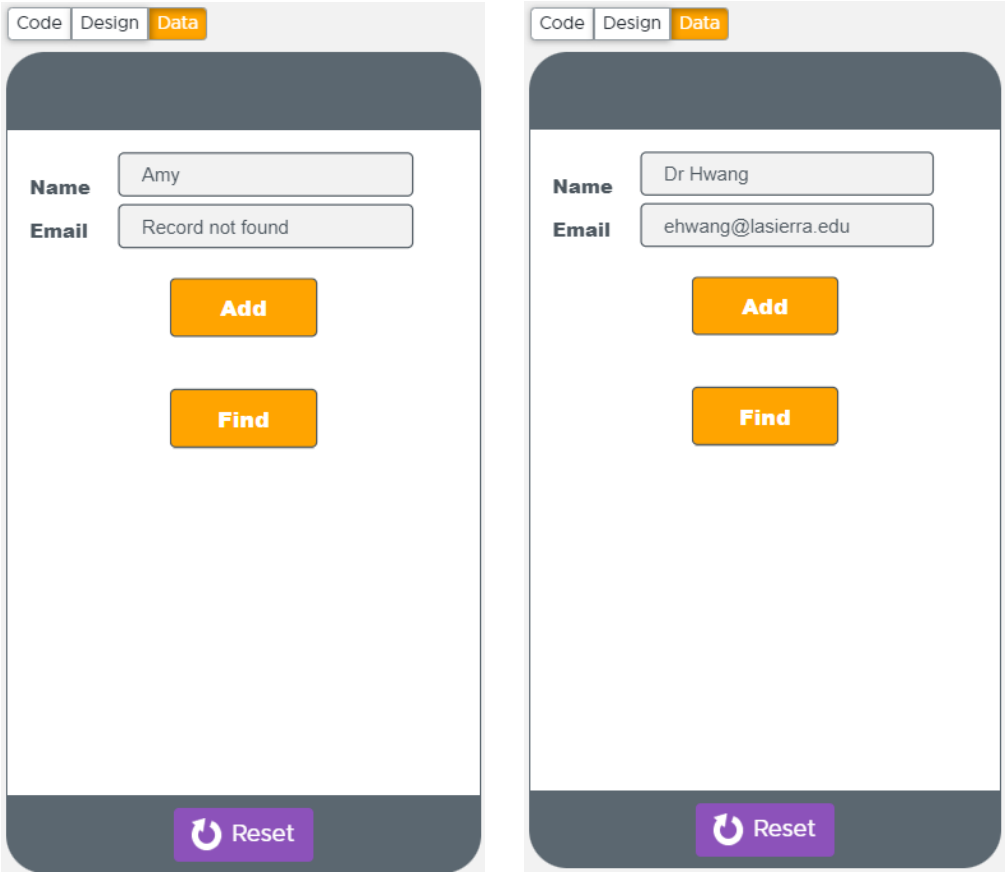


You should see one record with the name and email address that you entered. You can add more records as you like.

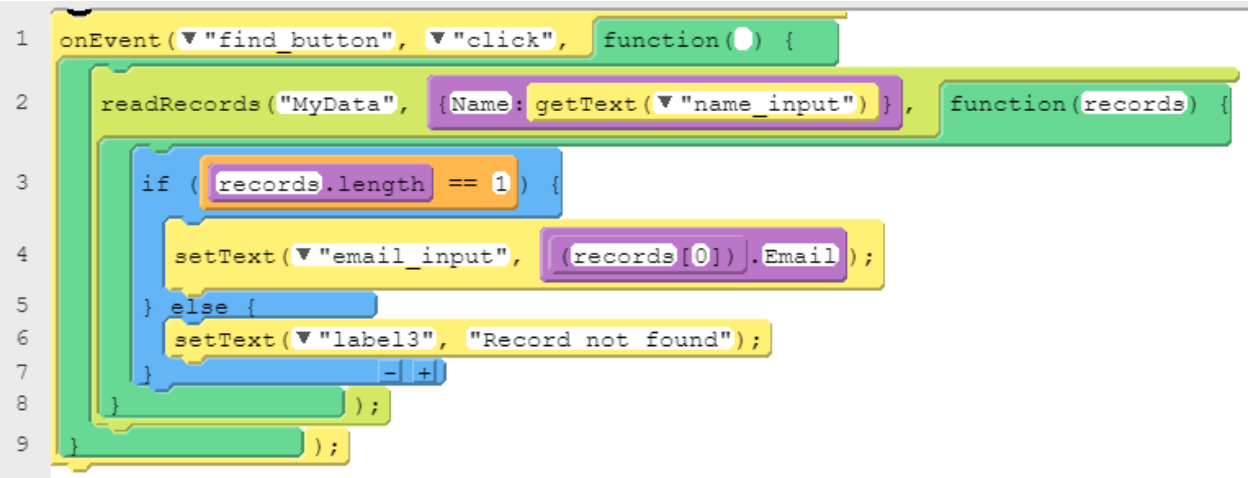


4. Find Record

The second onEvent is triggered when the find button is clicked. Before clicking on this button you need to first enter a name in the *name_input* text input box. If you type a name that doesn't exist in the database table, then it will display the message "Record not found," otherwise, it will display the email address for that person.



Here's the code for doing the find.



When the find button is clicked, it will execute the **readRecords** block inside the onEvents for the find button click. The **readRecords** command will search inside the given table for the given search criteria in the purple block. Our search criteria is

{Name:getText("name_input")}

In the example, it searches inside the *MyData* table in the *Name* column for the name obtained from the **getText** command in the *name_input* text box.

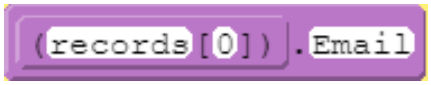
The **readRecords** command returns the data found, if any, in the *records* variable.

The purple block command

returns the number of records actually found matching the search criteria.

You can first check the length of this *records* variable to see how many matching records are found. In our example, we test if there is one record found. If there is then we display the email address found in the email_input text box, otherwise, display the "Record not found" message.

The purple block command



records[0].Email

retrieves the Email address stored in the first record found matching the search criteria. Note that this might not be the first record in the table. In computer programming, counting usually starts at 0, so record zero (denoted as records[0]) is the first record.

5. Delete Record

To delete a record you must first find the record to delete. If the record is found then you can delete it with the green deleteRecord block providing the id of the record to delete. Recall that every record has a unique id number. In the example below we use the id of the first record found.

```
records[0].id
```

```
onEvent("delete_button", "click", function() {
  readRecords("MyData", {Name: getText("name_input")}, function(records) {
    if (records.length == 1) {
      deleteRecord("MyData", {id: records[0].id}, function(success) {
        setText("label3", "Record deleted");
      });
    } else {
      setText("label3", "Record not found");
    }
  });
});
```

6. Change Record

Just like with the delete record, you must first find the record to change. If the record is found then you can change the information with the green updateRecord block providing the id of the record to change and all the data fields. Recall that every record has a unique id number. In the example below we use the id of the first record found.

```
onEvent("change_button", "click", function() {
  readRecords("MyData", {Name: getText("name_input")}, function(records) {
    if (records.length == 1) {
      updateRecord("MyData", {id: records[0].id, Name: getText("name_input"), Email: (getText("email_input"))}, function(record, success) {
        setText("label3", "Record changed");
      });
    } else {
      setText("label3", "Record not found");
    }
  });
});
```

Problems (Questions with an * are more difficult)

1) Display the total number of records in the database table.

2) Display the number of records found that matches a given search criteria.

3) Implement the Delete Record operation. Make sure that it works and that you understand the code for it.

4) Implement the Change Record operation. Make sure that it works and that you understand the code for it.

5) * Add another field for storing the Phone number in the database table. Modify your UI and code accordingly so that every record will have the Name, Email, and Phone fields.

6) * Recall the radio buttons used for selecting the year level. Add the radio buttons to your program so that the year level is also stored in the database table for each record added.

7) * Modify your Find Record operation so that it also displays the year level correctly in the radio buttons.

8) * Recall the checkboxes for selecting the fruits. Add the checkboxes to your program so that all the fruits selected are also stored in the database table for each record added.

9) * Modify your Find Record operation so that it also displays the selected fruits correctly in the checkboxes.

