University of British Columbia, Vancouver

Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: Milestone 4

Date: July 31, 2024

Group Number: Project Group 3

Name	**Student Number**	**CS Alias (userid)**	**Preferred E-mail Address**
Jeffrey Ho	26034066	g1k2b	jho1211@student.ubc.ca
Michelle Lei	11357167	j1r2b	michellejslei@gmail.com
Frederick Sunstrum	42266379	k0l3e	fr.sunstrum@gmail.com

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

University of British Columbia, Vancouver

Department of Computer Science

Project Repository

https://github.students.cs.ubc.ca/CPSC304-2024S-T2/project_g1k2b_j1r2b_k0l3e

SQL Script

The SQL script to initialize all the tables can be found at:
https://github.students.cs.ubc.ca/CPSC304-2024S-T2/project_g1k2b_j1r2b_k0l3e/blob/main/backend/sql/comgarden.sql

Project Description

This database management system will be used by a municipality to store information about various community gardens throughout the city so that they can better organize these community gardens. Each garden will be managed by different community organizations and they will record the plants that are seeded in each plot and the gardeners who are assigned to each plot. There will also be donations of tools and plants which are recorded so that the garden members will know what is available for use.

Changes to Schema

Our CREATE TABLE statements have changed slightly as we realized that there is no ‘ON UPDATE’ option for Oracle databases, so we had to remove those. Also, we moved the ‘availability’ attributes from the Tool relation to the Stores relation as we realized that we couldn’t change the availability of a tool for a particular garden unless we did that. This change has been reflected in our updated ER Diagram below.

We also removed ON DELETE CASCADE from GardenerPlot when GardenNumPlots is deleted. This is because we do not want users to be able to delete gardens once they have been added to the database. We modified the ON DELETE SET NULL from GardenerPlot when Gardener is deleted to ON DELETE CASCADE. This is so that when a gardener is removed from the database, they are no longer assigned to the plot and the plot is cleared such that all plants growing in that plot are removed before the next gardener is potentially assigned.

Final Schema

PRIMARY KEY
FOREIGN KEY

```
GardenInfo(address: VARCHAR(20),
           garden_name: VARCHAR(20),
           num_of_plots: INTEGER)
```

- garden_name is UNIQUE

Donation(donation_id: INTEGER,
donor_name: VARCHAR(20),
don_date: DATE,
item: VARCHAR)

- Candidate key: {donor_name, date, item}

Tool(tool_type: VARCHAR(20))

GardenManager(name: VARCHAR(20),
phone: CHAR(12),
email: VARCHAR(30))

- Candidate key: {name, phone}
- name is NOT NULL

GardenNumPlots(address: VARCHAR(20),
num_of_plots: INTEGER)

Receives(**donation_id**: INTEGER,
garden_address: VARCHAR(20))

Stores(**garden_address**: VARCHAR(20),
tool_type: VARCHAR(20),
availability: CHAR(1))

GardenManages(**garden_name**: VARCHAR(20),
manager_email: VARCHAR(30))

Gardener(email: VARCHAR(30),
phone: CHAR(12))

- Candidate key: {name, phone}
- name is NOT NULL

GardenerPlot(**garden_address**: VARCHAR(20),
gardener_email: VARCHAR(30),
plot_num: INTEGER,
sun_exposure: VARCHAR(20),
plot_size: INTEGER)

```
Compost(bin_id: INTEGER, capacity: INTEGER)

HasCompost(bin_id: INTEGER, garden_address: VARCHAR(20))

PlantColour(common_name: VARCHAR(20), colour: VARCHAR(20))

PlantHarvest(common_name: VARCHAR(20), harvest_time: INTEGER)

PlantInfo(species: VARCHAR(30),
          genus: VARCHAR(20),
          variety: VARCHAR(20),
          common_name: VARCHAR(20))
   ●  common_name is NOT NULL

Grows(species: VARCHAR(30),
      genus: VARCHAR(20),
      variety: VARCHAR(20),
      plot_num: INTEGER,
      garden_address: VARCHAR(20),
      qty: INTEGER, plant_date: DATE)
```

Initialized Tables

The data in each relation after running the initialization script can be found at: https://github.students.cs.ubc.ca/CPSC304-2024S-T2/project_g1k2b_j1r2b_k0l3e/blob/main/milestone_docs/SQL%20Initialization%20Result.pdf

University of British Columbia, Vancouver

Department of Computer Science

INSERT operation

Query

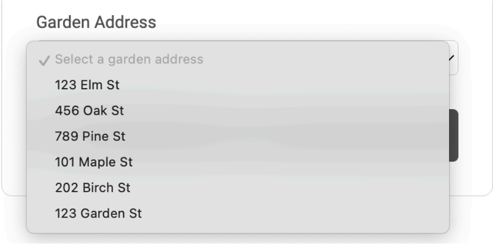
The INSERT operation in our application involves inserting a new donation into the **Donation** table and subsequently inserting a related entry into the **Receives** table.
The INSERT operation handles cases where the foreign key value (**garden_address**) does not exist in the **GardenInfo** table. However, the front end prevents these cases from occurring because garden addresses must be selected from a drop-down of addresses that already exist in the database. If the specified garden address does not exist, an appropriate error message is logged, and the transaction is rolled back.

The implementation of this INSERT operation can be found in backend/models/donation.js in the insertDonation function (line 28) and backend/models/receives.js in the insertReceives function (line 12).

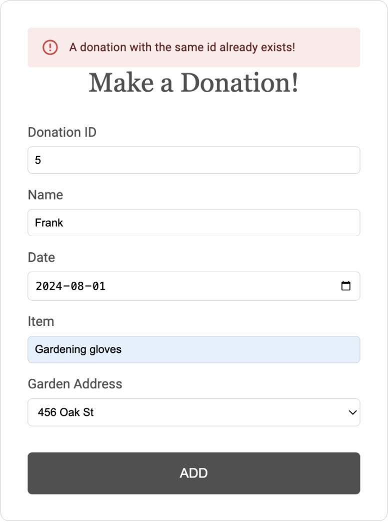
Screenshots

Found at: http://localhost:3000/donations

Garden address selection on new donation form:



Error for existing donation id on new donation form:



A donation with the same id already exists!

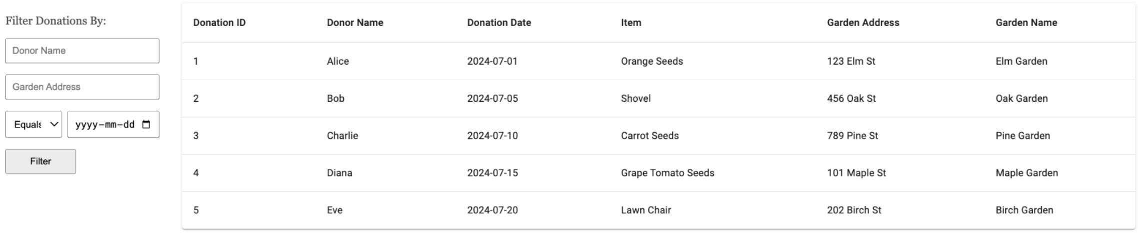
Make a Donation!

Donation ID	5
Name	Frank
Date	2024-08-01
Item	Gardening gloves
Garden Address	456 Oak St

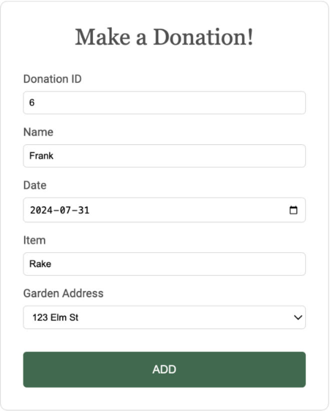
University of British Columbia, Vancouver

Department of Computer Science

Table before insertion:



Donation ID	Donor Name	Donation Date	Item	Garden Address	Garden Name
1	Alice	2024-07-01	Orange Seeds	123 Elm St	Elm Garden
2	Bob	2024-07-05	Shovel	456 Oak St	Oak Garden
3	Charlie	2024-07-10	Carrot Seeds	789 Pine St	Pine Garden
4	Diana	2024-07-15	Grape Tomato Seeds	101 Maple St	Maple Garden
5	Eve	2024-07-20	Lawn Chair	202 Birch St	Birch Garden



Make a Donation!

Donation ID: 6

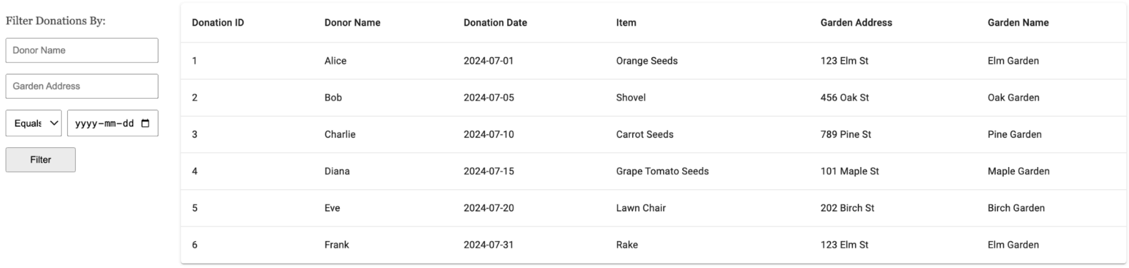
Name: Frank

Date: 2024-07-31

Item: Rake

Garden Address: 123 Elm St

Table after insertion:



Donation ID	Donor Name	Donation Date	Item	Garden Address	Garden Name
1	Alice	2024-07-01	Orange Seeds	123 Elm St	Elm Garden
2	Bob	2024-07-05	Shovel	456 Oak St	Oak Garden
3	Charlie	2024-07-10	Carrot Seeds	789 Pine St	Pine Garden
4	Diana	2024-07-15	Grape Tomato Seeds	101 Maple St	Maple Garden
5	Eve	2024-07-20	Lawn Chair	202 Birch St	Birch Garden
6	Frank	2024-07-31	Rake	123 Elm St	Elm Garden

DELETE operation

Query

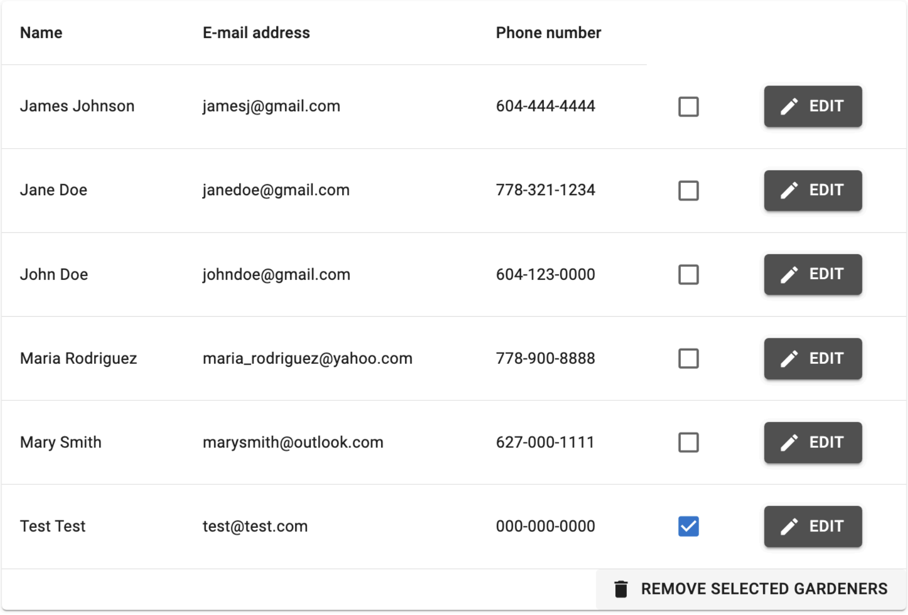
The DELETE operation in our application involves deleting an existing gardener from the **Gardener** table. This subsequently deletes any tuples in the **GardenerPlot** table which have gardeners with the same email as the one deleted due to the ON DELETE CASCADE relationship between the two tables.

The implementation of this DELETE operation can be found in backend/models/gardener.js in the deleteGardenersByEmail function (line 71) and backend/models/gardenerPlot.js in the deleteGardenerPlotsByEmail function (line 141).

Screenshots

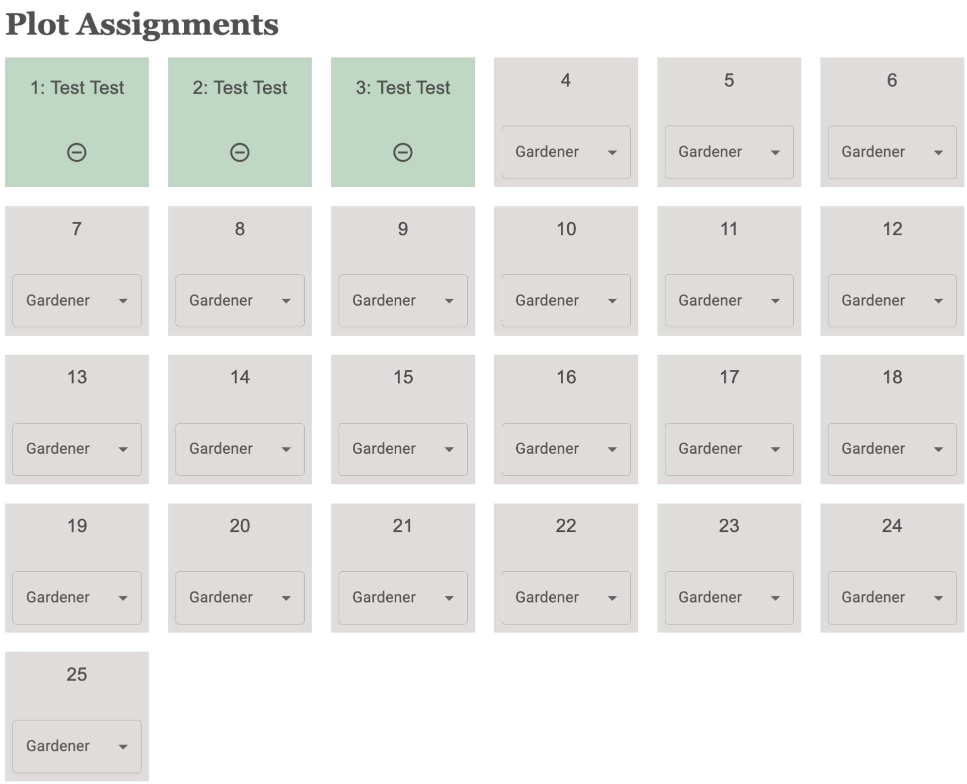
Found at: http://localhost:3000/gardeners

Gardeners table before deleting Test Test gardener:



Name	E-mail address	Phone number		
James Johnson	jamesj@gmail.com	604-444-4444	□	EDIT
Jane Doe	janedoe@gmail.com	778-321-1234	□	EDIT
John Doe	johndoe@gmail.com	604-123-0000	□	EDIT
Maria Rodriguez	maria_rodriguez@yahoo.com	778-900-8888	□	EDIT
Mary Smith	marysmith@outlook.com	627-000-1111	□	EDIT
Test Test	test@test.com	000-000-0000	☑	EDIT

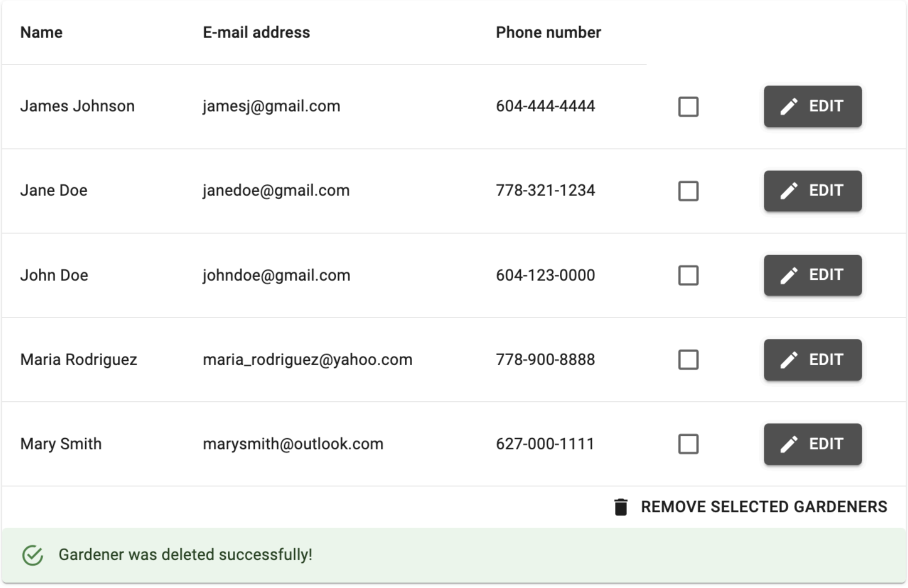
Plot assignments in Garden Garden before deleting Test Test gardener:



Plot Assignments

1: Test Test ⊖	2: Test Test ⊖	3: Test Test ⊖	4 Gardener ▾	5 Gardener ▾	6 Gardener ▾
7 Gardener ▾	8 Gardener ▾	9 Gardener ▾	10 Gardener ▾	11 Gardener ▾	12 Gardener ▾
13 Gardener ▾	14 Gardener ▾	15 Gardener ▾	16 Gardener ▾	17 Gardener ▾	18 Gardener ▾
19 Gardener ▾	20 Gardener ▾	21 Gardener ▾	22 Gardener ▾	23 Gardener ▾	24 Gardener ▾
25 Gardener ▾					

Gardeners table after deleting Test Test gardener:



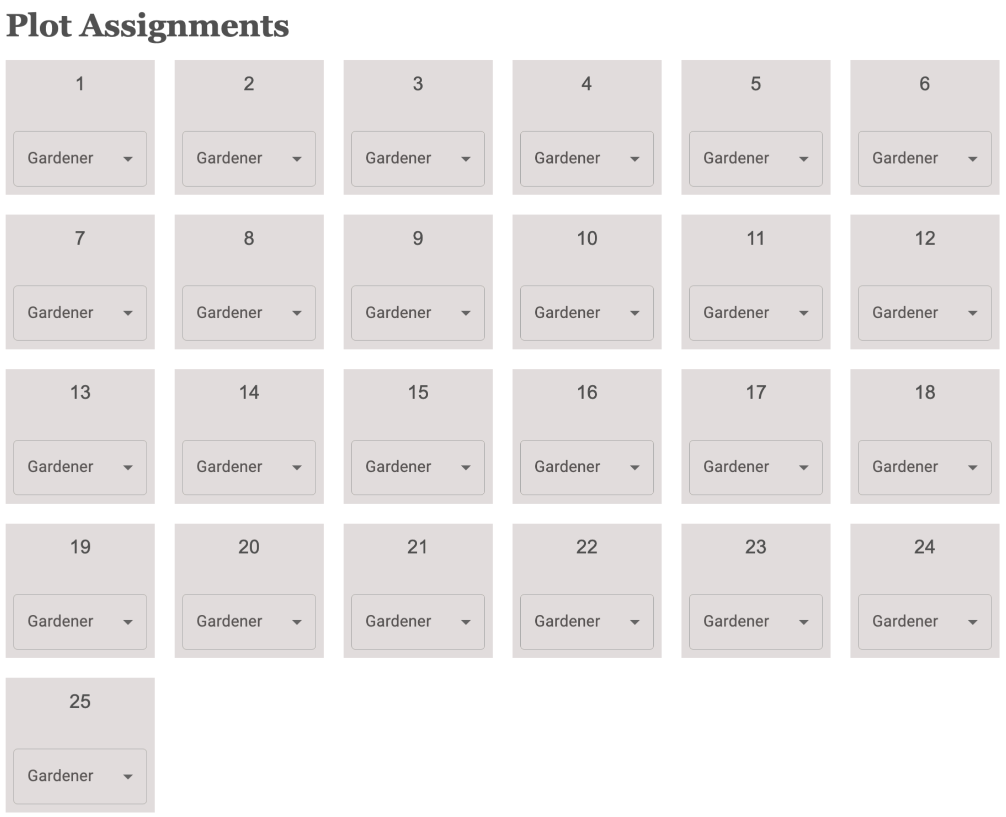
Name	E-mail address	Phone number		
James Johnson	jamesj@gmail.com	604-444-4444	□	EDIT
Jane Doe	janedoe@gmail.com	778-321-1234	□	EDIT
John Doe	johndoe@gmail.com	604-123-0000	□	EDIT
Maria Rodriguez	maria_rodriguez@yahoo.com	778-900-8888	□	EDIT
Mary Smith	marysmith@outlook.com	627-000-1111	□	EDIT

✓ Gardener was deleted successfully!

Plot assignments in Garden Garden after deleting Test Test gardener:

University of British Columbia, Vancouver

Department of Computer Science



Plot Assignments

1 Gardener ▾	2 Gardener ▾	3 Gardener ▾	4 Gardener ▾	5 Gardener ▾	6 Gardener ▾
7 Gardener ▾	8 Gardener ▾	9 Gardener ▾	10 Gardener ▾	11 Gardener ▾	12 Gardener ▾
13 Gardener ▾	14 Gardener ▾	15 Gardener ▾	16 Gardener ▾	17 Gardener ▾	18 Gardener ▾
19 Gardener ▾	20 Gardener ▾	21 Gardener ▾	22 Gardener ▾	23 Gardener ▾	24 Gardener ▾
25 Gardener ▾					

UPDATE operation

Query

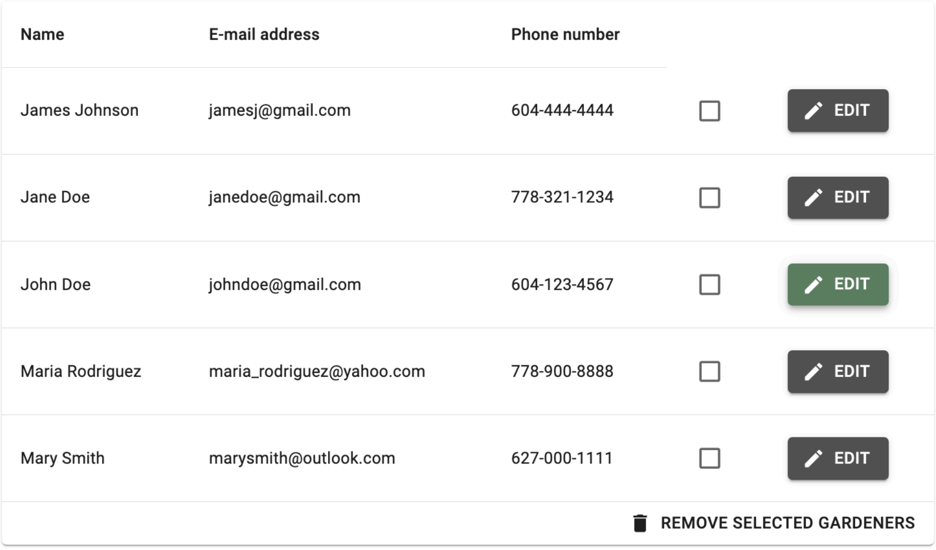
The UPDATE operation in our application involves updating the name or phone number (non-primary key attributes) of an existing gardener in the **Gardener** table. The combination of name and phone number for a gardener is also a candidate key so must fulfill the UNIQUE constraint, so if a user tries to update a gardener's name and phone number and that combination is the same as another gardener's name and phone number in the table, an error will be thrown and the user will be prompted to try again.

The implementation of this UPDATE operation can be found in backend/models/gardener.js in the updateGardener function (line 147).

Screenshots

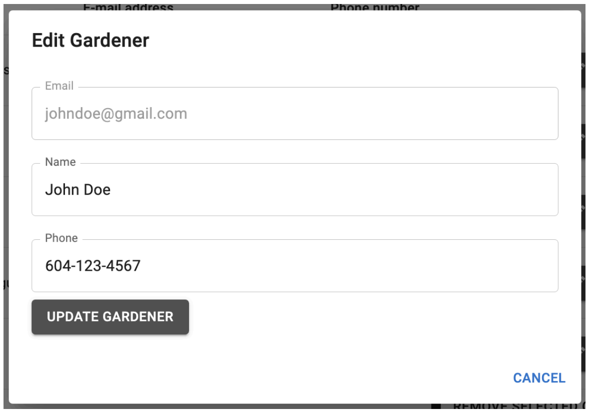
Found at: http://localhost:3000/gardeners

Table before editing gardener 'John Doe':



Name	E-mail address	Phone number		
James Johnson	jamesj@gmail.com	604-444-4444	□	EDIT
Jane Doe	janedoe@gmail.com	778-321-1234	□	EDIT
John Doe	johndoe@gmail.com	604-123-4567	□	EDIT
Maria Rodriguez	maria_rodriguez@yahoo.com	778-900-8888	□	EDIT
Mary Smith	marysmith@outlook.com	627-000-1111	□	EDIT

Edit gardener form before editing:



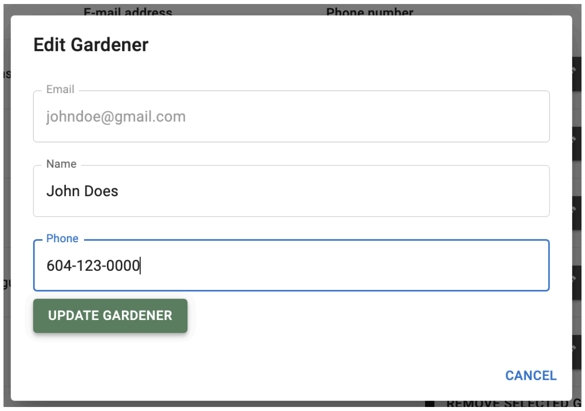
Edit Gardener

Email: johndoe@gmail.com

Name: John Doe

Phone: 604-123-4567

Edit gardener form during editing



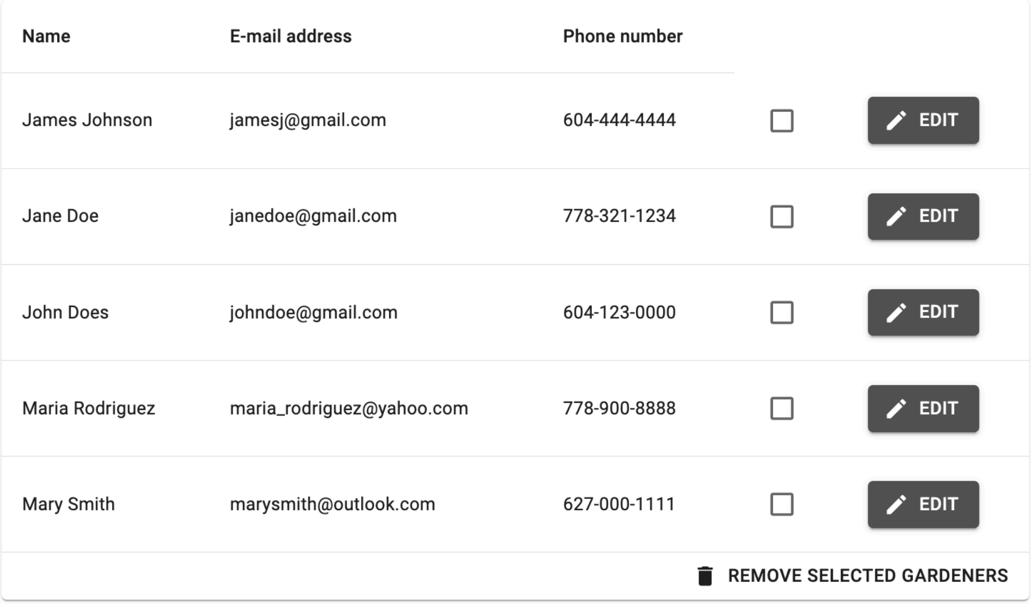
Edit Gardener

Email: johndoe@gmail.com

Name: John Does

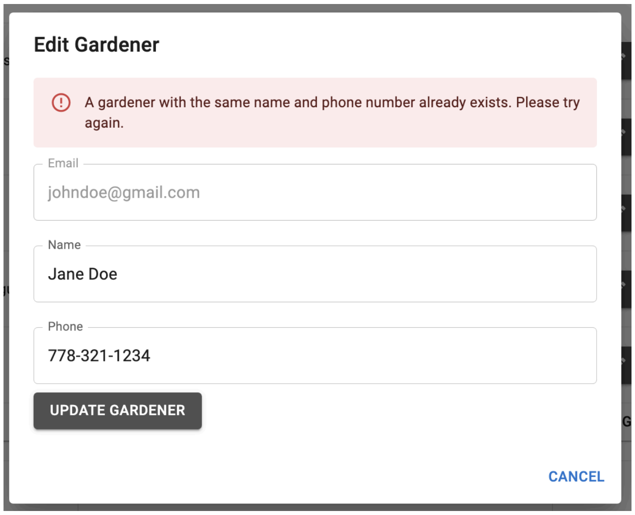
Phone: 604-123-0000

Gardeners table after updating ‘John Doe’ name and phone number:



Name	E-mail address	Phone number		
James Johnson	jamesj@gmail.com	604-444-4444	□	EDIT
Jane Doe	janedoe@gmail.com	778-321-1234	□	EDIT
John Does	johndoe@gmail.com	604-123-0000	□	EDIT
Maria Rodriguez	maria_rodriguez@yahoo.com	778-900-8888	□	EDIT
Mary Smith	marysmith@outlook.com	627-000-1111	□	EDIT

Unique constraint violation error on edit gardener form:



Edit Gardener

A gardener with the same name and phone number already exists. Please try again.

Email: johndoe@gmail.com

Name: Jane Doe

Phone: 778-321-1234

University of British Columbia, Vancouver

Department of Computer Science

Selection

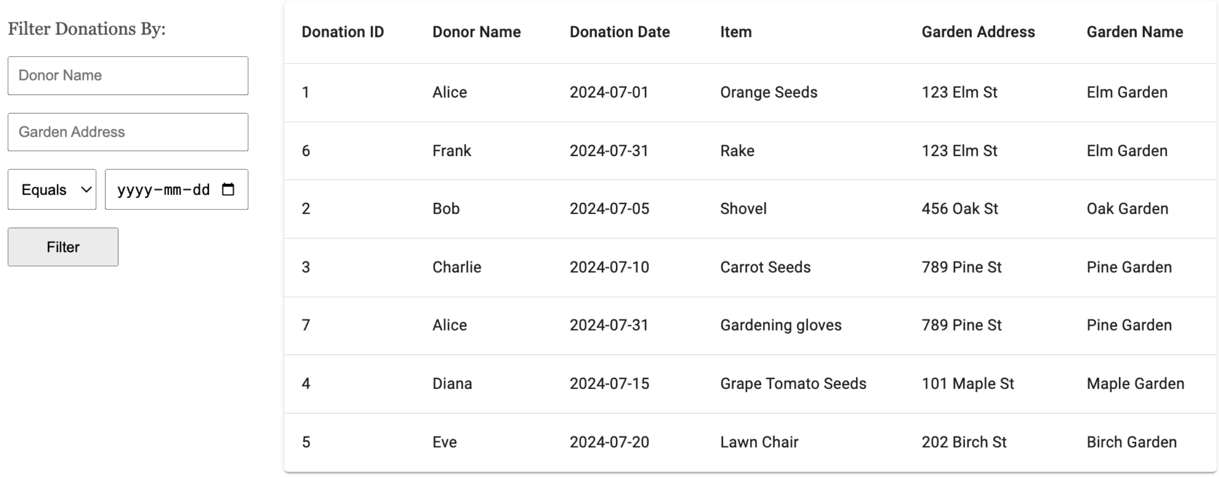
The SELECT query in our application allows users to filter tuples in the **Donations** table by donor name, garden address, and donation date (before, at, or after a given date). These filters can be applied separately or in combination with each other.

The implementation of this SELECT operation can be found in backend/models/donations.js in the getAll function (line 118).

Screenshots

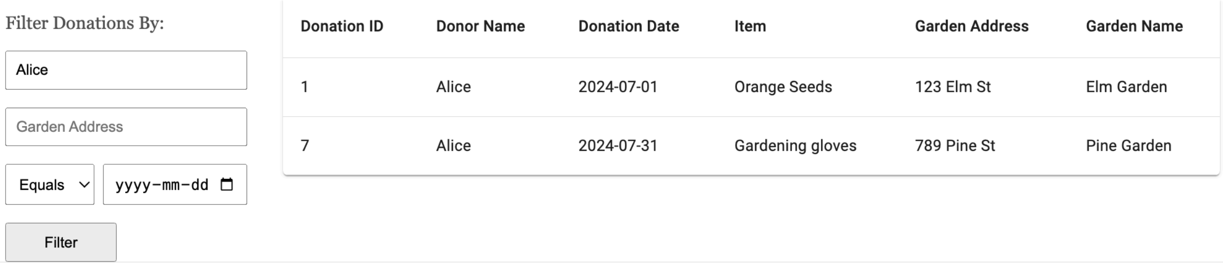
Found at: http://localhost:3000/donations

Table before filtering:



Donation ID	Donor Name	Donation Date	Item	Garden Address	Garden Name
1	Alice	2024-07-01	Orange Seeds	123 Elm St	Elm Garden
6	Frank	2024-07-31	Rake	123 Elm St	Elm Garden
2	Bob	2024-07-05	Shovel	456 Oak St	Oak Garden
3	Charlie	2024-07-10	Carrot Seeds	789 Pine St	Pine Garden
7	Alice	2024-07-31	Gardening gloves	789 Pine St	Pine Garden
4	Diana	2024-07-15	Grape Tomato Seeds	101 Maple St	Maple Garden
5	Eve	2024-07-20	Lawn Chair	202 Birch St	Birch Garden

Table when filtering only by donor name:



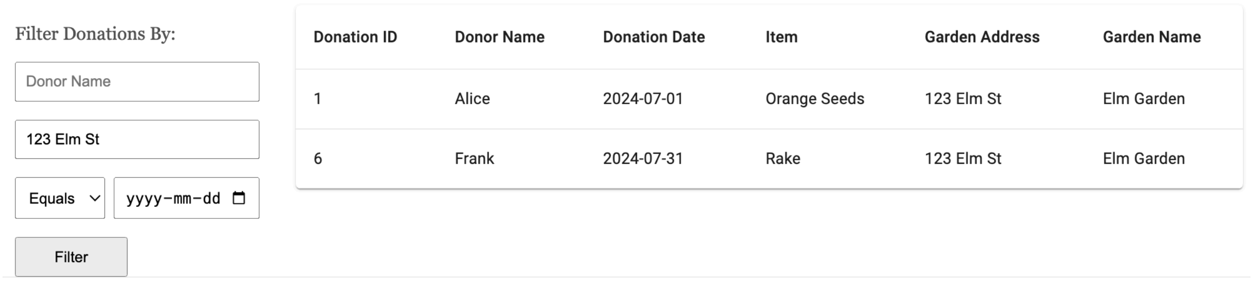
Alice

Donation ID	Donor Name	Donation Date	Item	Garden Address	Garden Name
1	Alice	2024-07-01	Orange Seeds	123 Elm St	Elm Garden
7	Alice	2024-07-31	Gardening gloves	789 Pine St	Pine Garden

Table when filtering only by garden address:

University of British Columbia, Vancouver

Department of Computer Science



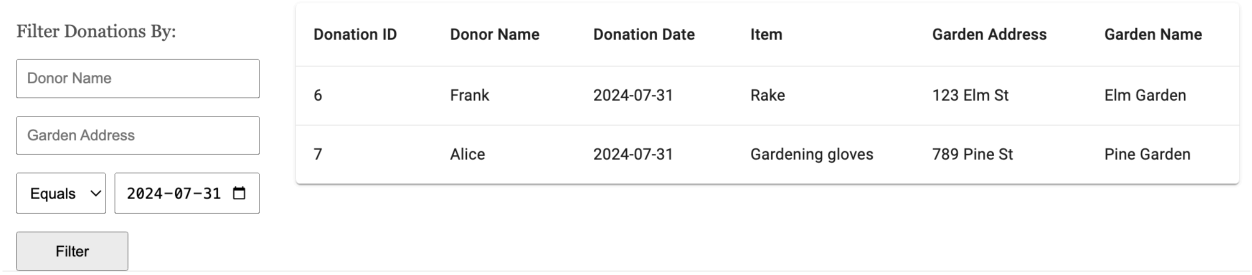
Filter Donations By:

123 Elm St

Equals yyyy-mm-dd

Donation ID	Donor Name	Donation Date	Item	Garden Address	Garden Name
1	Alice	2024-07-01	Orange Seeds	123 Elm St	Elm Garden
6	Frank	2024-07-31	Rake	123 Elm St	Elm Garden

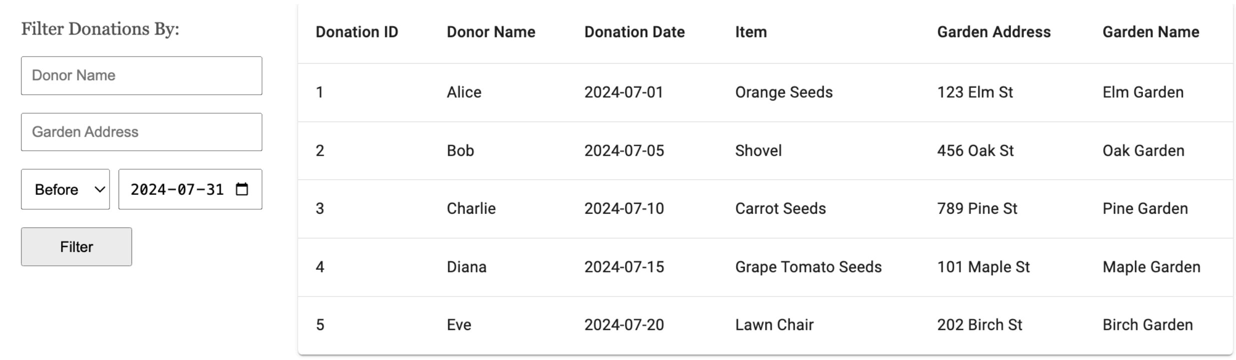
Table when filtering only by dates (before, equals, and after a user-provided given date):



Filter Donations By:

Equals 2024-07-31

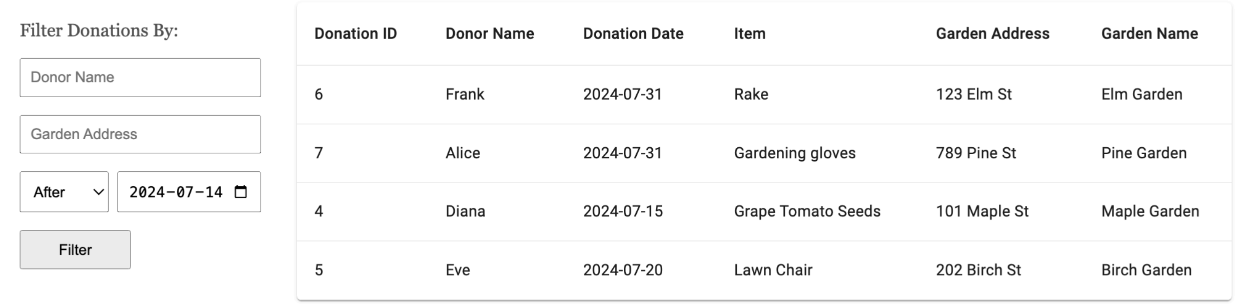
Donation ID	Donor Name	Donation Date	Item	Garden Address	Garden Name
6	Frank	2024-07-31	Rake	123 Elm St	Elm Garden
7	Alice	2024-07-31	Gardening gloves	789 Pine St	Pine Garden



Filter Donations By:

Before 2024-07-31

Donation ID	Donor Name	Donation Date	Item	Garden Address	Garden Name
1	Alice	2024-07-01	Orange Seeds	123 Elm St	Elm Garden
2	Bob	2024-07-05	Shovel	456 Oak St	Oak Garden
3	Charlie	2024-07-10	Carrot Seeds	789 Pine St	Pine Garden
4	Diana	2024-07-15	Grape Tomato Seeds	101 Maple St	Maple Garden
5	Eve	2024-07-20	Lawn Chair	202 Birch St	Birch Garden



Filter Donations By:

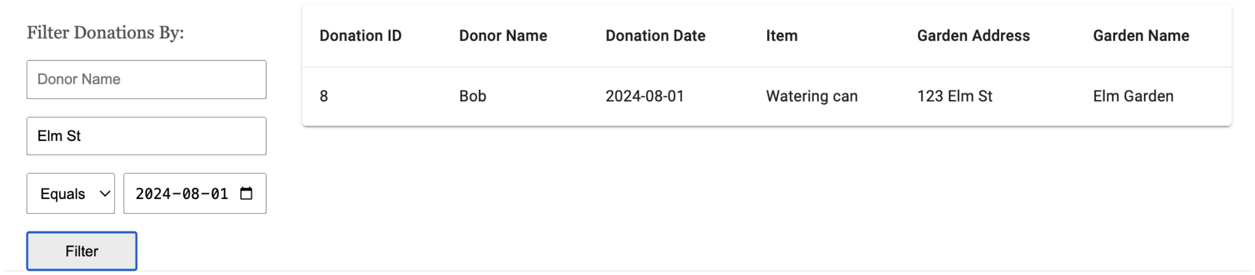
After 2024-07-14

Donation ID	Donor Name	Donation Date	Item	Garden Address	Garden Name
6	Frank	2024-07-31	Rake	123 Elm St	Elm Garden
7	Alice	2024-07-31	Gardening gloves	789 Pine St	Pine Garden
4	Diana	2024-07-15	Grape Tomato Seeds	101 Maple St	Maple Garden
5	Eve	2024-07-20	Lawn Chair	202 Birch St	Birch Garden

Table when applying multiple filters concurrently:

University of British Columbia, Vancouver

Department of Computer Science



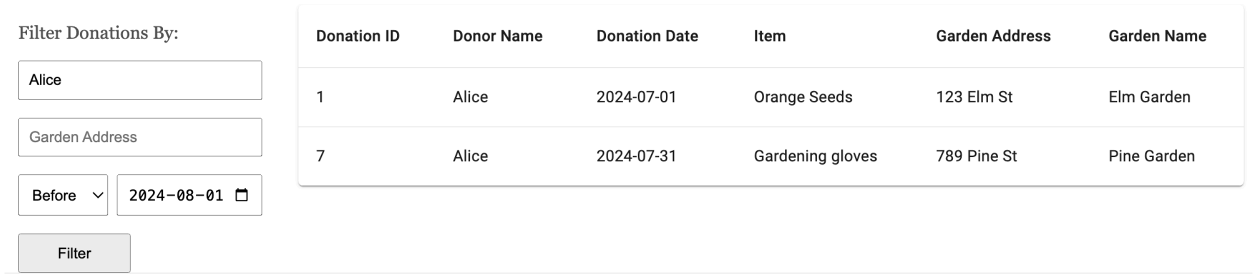
Filter Donations By:

Donor Name

Elm St

Equals 2024-08-01

Donation ID	Donor Name	Donation Date	Item	Garden Address	Garden Name
8	Bob	2024-08-01	Watering can	123 Elm St	Elm Garden



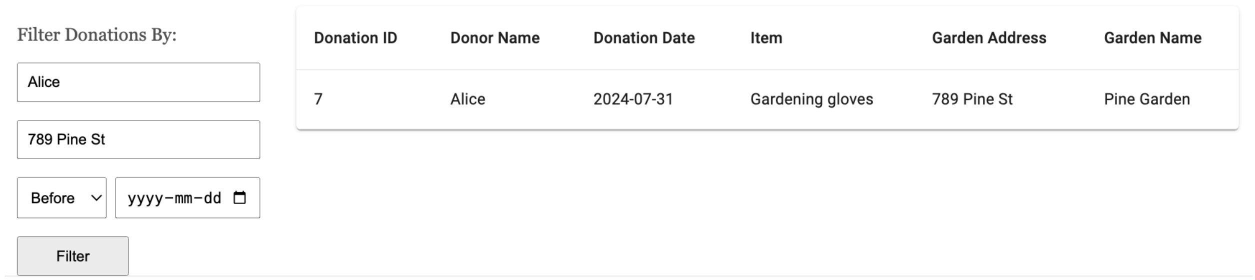
Filter Donations By:

Alice

Garden Address

Before 2024-08-01

Donation ID	Donor Name	Donation Date	Item	Garden Address	Garden Name
1	Alice	2024-07-01	Orange Seeds	123 Elm St	Elm Garden
7	Alice	2024-07-31	Gardening gloves	789 Pine St	Pine Garden



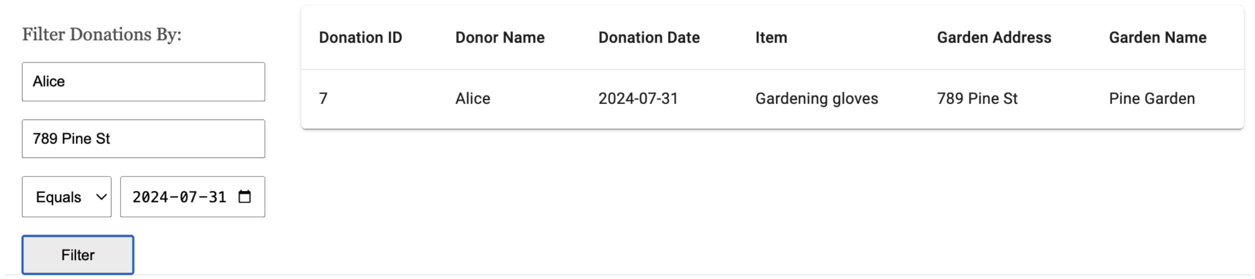
Filter Donations By:

Alice

789 Pine St

Before yyyy-mm-dd

Donation ID	Donor Name	Donation Date	Item	Garden Address	Garden Name
7	Alice	2024-07-31	Gardening gloves	789 Pine St	Pine Garden



Filter Donations By:

Alice

789 Pine St

Equals 2024-07-31

Donation ID	Donor Name	Donation Date	Item	Garden Address	Garden Name
7	Alice	2024-07-31	Gardening gloves	789 Pine St	Pine Garden

University of British Columbia, Vancouver

Department of Computer Science

Projection

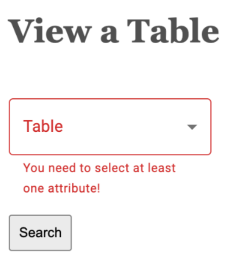
We have an admin page that allows the admin to select whatever table they wish to view. The tables are dynamically fetched from the user_tables tablespace in the OracleDB. After selecting the table they want, they can choose which attributes they want to project onto the table. These attributes are dynamically fetched from the relation to populate the checkboxes. After pressing the search button, a query will be run to get the rows from the relation with the attributes specified.

Frontend

Found at: http://localhost:3000/admin

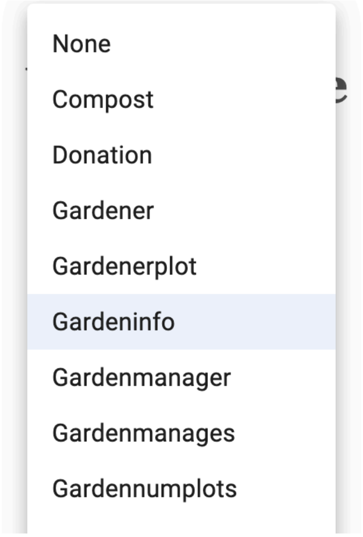
Screenshots

Default View:



You need to select at least one attribute!

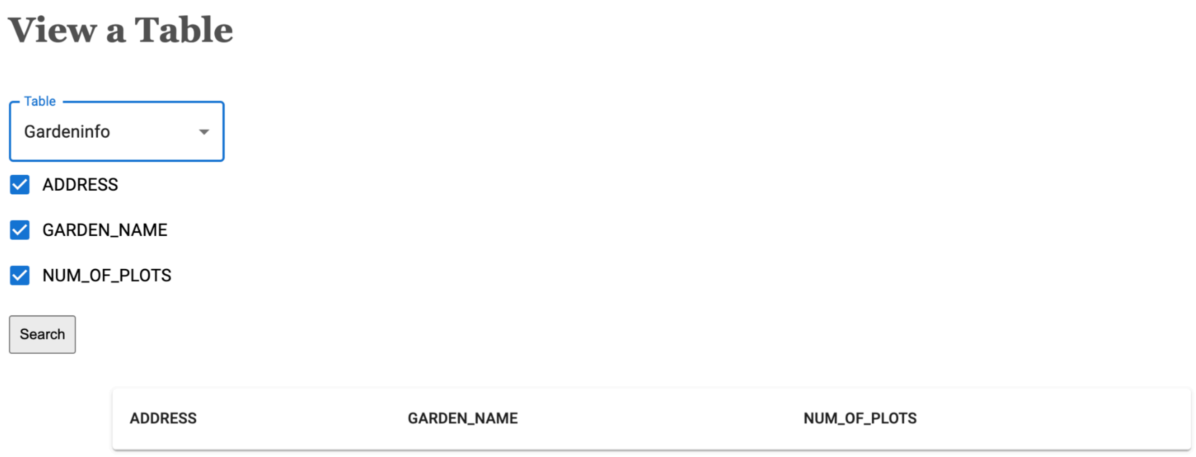
Table Selection:



After choosing a table:

University of British Columbia, Vancouver

Department of Computer Science



View a Table

Table
Gardeninfo

☑ ADDRESS

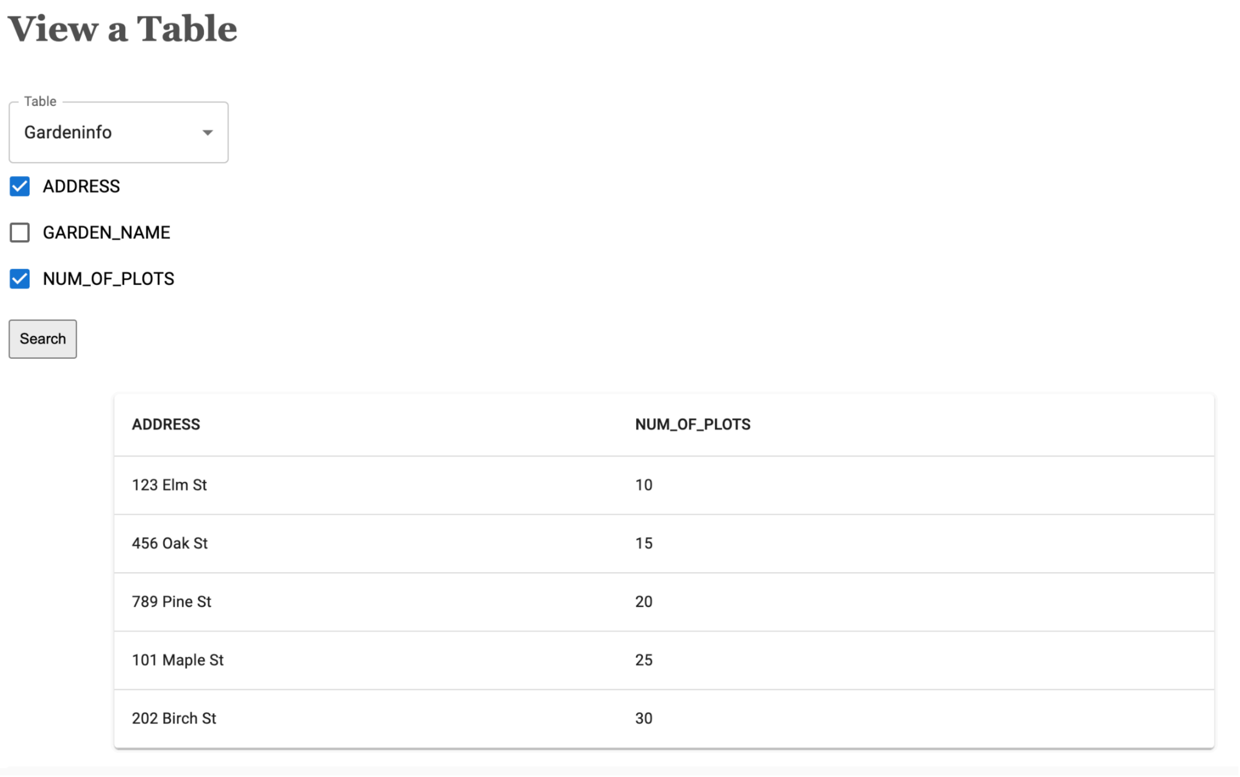
☑ GARDEN_NAME

☑ NUM_OF_PLOTS

Search

ADDRESS	GARDEN_NAME	NUM_OF_PLOTS

After choosing attributes and pressing search:



View a Table

Table
Gardeninfo

☑ ADDRESS

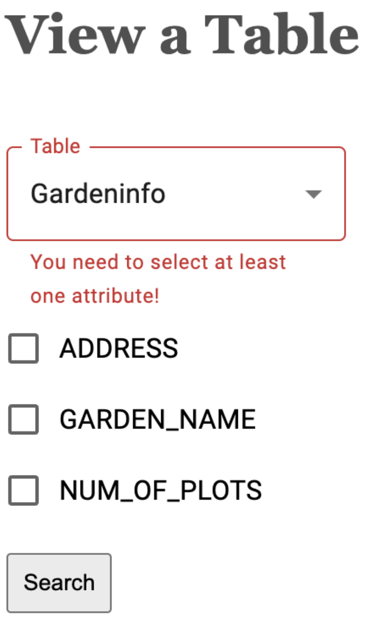
□ GARDEN_NAME

☑ NUM_OF_PLOTS

Search

ADDRESS	NUM_OF_PLOTS
123 Elm St	10
456 Oak St	15
789 Pine St	20
101 Maple St	25
202 Birch St	30

If no attributes chosen:



View a Table

Table: Gardeninfo

You need to select at least one attribute!

□ ADDRESS

□ GARDEN_NAME

□ NUM_OF_PLOTS

Backend

SQL Query Found in:
(backend/models/all.js, getTable(tname, attrs), line 66)

- constructSelectQuery(tname, attrs), line 54)

API Express Route Found in: (backend/controllers/allController.js)

Relevant API Endpoints:
Fetch All Relation Names: http://localhost:65535/api/tables
Fetch All Attributes of a Relation: http://localhost:65535/api/table/attrs
Fetch a Table with Attributes: http://localhost:65535/api/table/:table_name?attrs=attrA,attrB,attrC

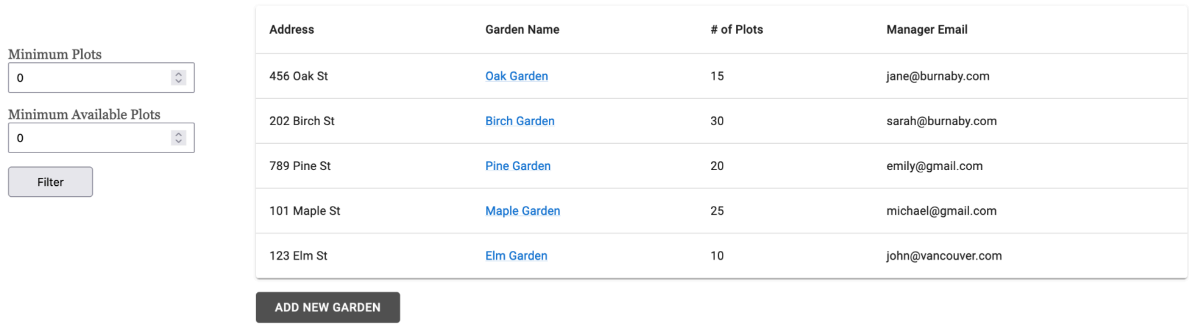
University of British Columbia, Vancouver

Department of Computer Science

Join + Aggregation with Group By + Having

The join, aggregation with Group By, and aggregation with having query retrieves information about gardens. It joins the main garden table with tables for garden managers and individual plots, and the user can filter results by providing a minimum number of plots value.
The same query uses Group By and Having to filter gardens based on the number of available plots. This requires counting the number of plots and selecting those greater than the user provided minimum available plots value.

Frontend Display of Join, Aggregation with Group By/Having
Found at: http://localhost:3000/gardens



Minimum Plots: 0

Minimum Available Plots: 0

Address	Garden Name	# of Plots	Manager Email
456 Oak St	Oak Garden	15	jane@burnaby.com
202 Birch St	Birch Garden	30	sarah@burnaby.com
789 Pine St	Pine Garden	20	emily@gmail.com
101 Maple St	Maple Garden	25	michael@gmail.com
123 Elm St	Elm Garden	10	john@vancouver.com

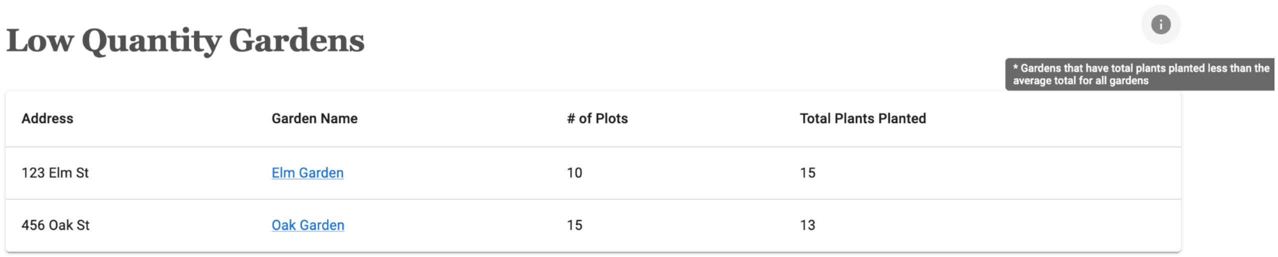
Frontend display of Join, aggregation with Group By/Having
SQL Query Found in: (backend/models/gardenInfo.js, getAllGardens(), lines 312-327)
API Express Route Found in: (backend/controllers/gardensController.js)
API Endpoint: http://localhost:65535/api/gardens

Nested Aggregation with Group By

The nested aggregation with group by query will sum the number of plants planted for each garden and then find all the gardens with a total number of plants planted lower than the average total for all gardens. This can be used to identify gardens that do not have much plot usage.

Frontend Display of Nested Aggregation with Group By Query

Found at: http://localhost:3000/gardens (bottom of the page)



Low Quantity Gardens

* Gardens that have total plants planted less than the average total for all gardens

Address	Garden Name	# of Plots	Total Plants Planted
123 Elm St	Elm Garden	10	15
456 Oak St	Oak Garden	15	13

Backend Query for Nested Aggregation with Group By

SQL Query Found in: (backend/models/grows.js, underAchievingGardens(), line 50-62)
API Express Route Found in: (backend/controllers/growsController.js)
API Endpoint: http://localhost:65535/api/gardens/underachievers

University of British Columbia, Vancouver

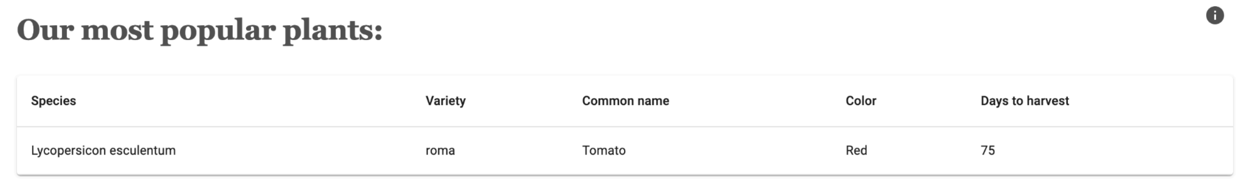
Department of Computer Science

Division

The Division query retrieves plants that are being grown in every garden.

Frontend Display of Division Query

Found at: http://localhost:3000/plants (bottom of the page)



Our most popular plants:

Species	Variety	Common name	Color	Days to harvest
Lycopersicon esculentum	roma	Tomato	Red	75

Backend Query for Division

SQL Query Found in: (backend/models/plantInfo.js, getPopularPlants(), lines 83-99)
API Express Route Found in: (backend/controllers/plantController.js)
API Endpoint: http://localhost:65535/api/plants/popular