University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 4

Date: 4 April 2024

Group Number: 39

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Hubert Wong	82570367	h8f3h	ycwonghubert@gmail.com
Sunny Lau	45195864	g8m3i	lausunny@student.ubc.ca
Veronica Leung	43477207	y0v1e	veronicaxlcw@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

1. Summary of the project

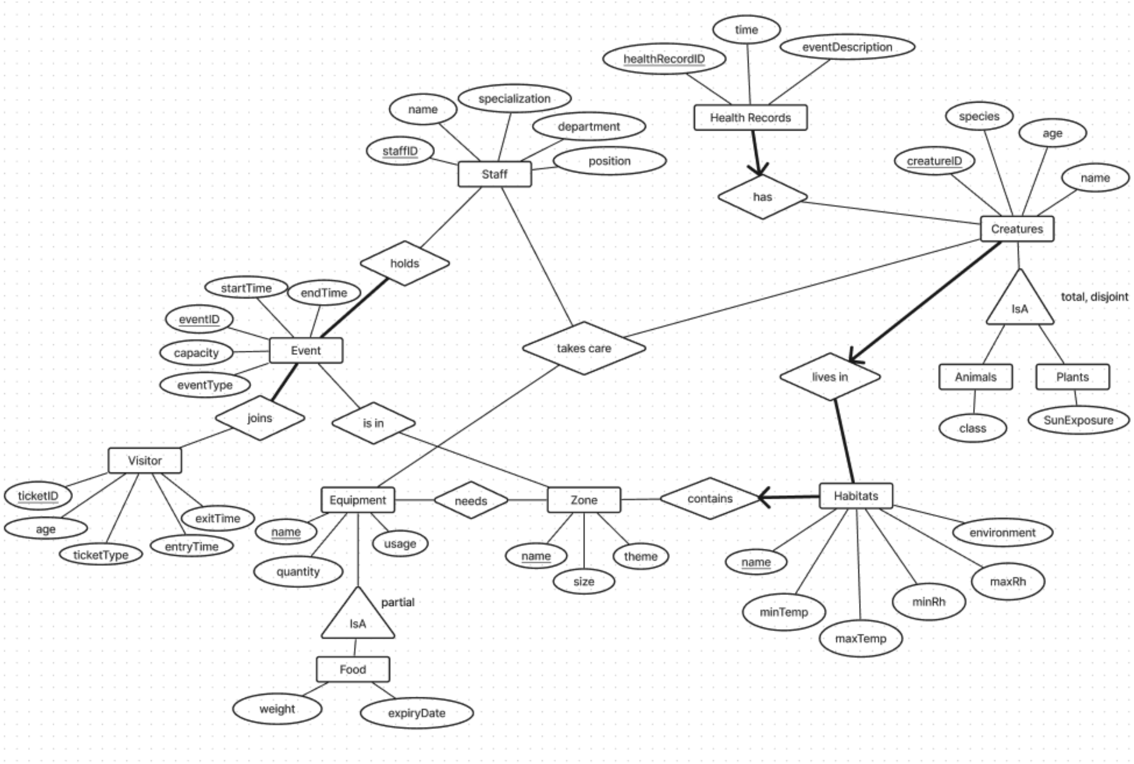
   This is a database for Botanical Park and Zoo Operations, which allows users to retrieve information about creatures in the botanical park and zoo, operations of the park and visitor statistics for data analysis.

2. Repository Link

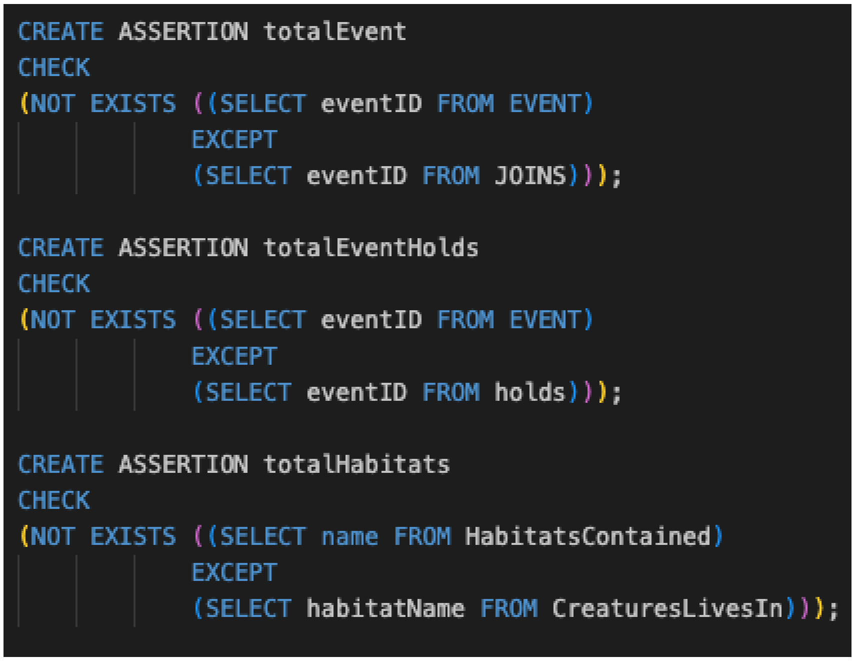
   https://github.students.cs.ubc.ca/CPSC304-2023W-T2/project_g8m3i_h8f3h_y0v1e

3. SQL script to create all tables and data

   - The script can be found in `./src/sql/scripts/create_table.sql`
   - For our ER diagram, the total participation constraint for Visitor in Joins has been removed, as not all visitors are able to join at least 1 event.
   - Updated ER diagram:



- For total participation, the assertions below should be applied:



```
CREATE ASSERTION totalEvent
CHECK
(NOT EXISTS ((SELECT eventID FROM EVENT)
            EXCEPT
            (SELECT eventID FROM JOINS)));

CREATE ASSERTION totalEventHolds
CHECK
(NOT EXISTS ((SELECT eventID FROM EVENT)
            EXCEPT
            (SELECT eventID FROM holds)));

CREATE ASSERTION totalHabitats
CHECK
(NOT EXISTS ((SELECT name FROM HabitatsContained)
            EXCEPT
            (SELECT habitatName FROM CreaturesLivesIn)));
```

4. Differences between the Final Schema and the Normalized Schema Derived from ERD

Normalized Schema derived from ER Diagram

- Staff1(staffID: INTEGER, name: VARCHAR(50), **specialization: VARCHAR(50)**, position: VARCHAR(50))
- Staff2(specialization: VARCHAR(50), department: VARCHAR(50))
- Event1(eventID: INTEGER, startTime: DATETIME, endTime: DATETIME, **eventType: ENUM(‘promotion’, ‘private party’, ‘holiday special’, ‘fundraising’, ‘conference’)**)
- Event2(eventType: ENUM(‘promotion’, ‘private party’, ‘holiday special’, ‘fundraising’, ‘conference’), capacity: INTEGER)
- Holds(**staffID: INTEGER, eventID: INTEGER**)
- Visitor1(ticketID: INTEGER, entryTime: DATETIME, exitTime: DATETIME, **age: INTEGER**)
- Visitor2(age: INTEGER, ticketType: ENUM(“Child”, “Teen”, “Adult”, “Senior”))
- Joins(**ticketID: INTEGER, eventID: INTEGER**)
- Zone(name: VARCHAR(50), size: INTEGER, theme: VARCHAR(50))
- IsIn(**eventID: INTEGER, zoneName: VARCHAR(50)**)
- Equipment(name: VARCHAR(50), quantity: INTEGER, usage: VARCHAR(50))
- Food(**name: VARCHAR(50)**, weight: FLOAT, expiryDate: DATE)
- Needs(**zoneName: VARCHAR(50), equipmentName: VARCHAR(50)**)

- HasHealthRecord(healthRecordID: INTEGER, time: DATETIME, eventDescription: VARCHAR(100), **creatureID: INTEGER**)
- CreaturesLivesIn(creatureID: INTEGER, species: VARCHAR(50), age: INTEGER, name: VARCHAR(50), class: VARCHAR(50), sunExposure: Enum (“FullSun”, “PartSun”, “PartShade”, “FullShade”) , **habitatName: VARCHAR(50)**)
- HabitatsContained(name:VARCHAR(50), minTemp: FLOAT, maxTemp: FLOAT, minRh: INTEGER, maxRh: INTEGER, environment: VARCHAR(50), **zoneName: VARCHAR(50)**)
- TakesCare(**staffID: INTEGER, equipmentName: VARCHAR(50), creatureID: INTEGER**)

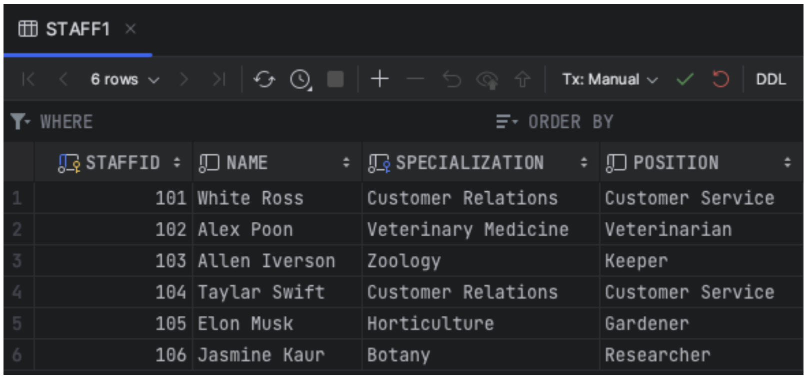
Final Schema

- Mostly remains unchanged from the last milestone, but some of the data types are changed. The changes are highlighted in yellow.

- Staff1(staffID: INTEGER, name: VARCHAR(50), **specialization: VARCHAR(50)**, position: VARCHAR(50))
- Staff2(specialization: VARCHAR(50), department: VARCHAR(50))
- Event1(eventID: INTEGER, startTime: TIMESTAMP, endTime: TIMESTAMP, **eventType: VARCHAR(30)**)
- Event2(eventType: VARCHAR(30), capacity: INTEGER)
- Holds(**staffID: INTEGER, eventID: INTEGER**)
- Visitor1(ticketID: INTEGER, entryTime: TIMESTAMP, exitTime: TIMESTAMP, **age: INTEGER**)
- Visitor2(age: INTEGER, ticketType: VARCHAR(10))
- Joins(**ticketID: INTEGER, eventID: INTEGER**)
- Zone(name: VARCHAR(50), size: INTEGER, theme: VARCHAR(50))
- IsIn(**eventID: INTEGER, zoneName: VARCHAR(50)**)
- Equipment(name: VARCHAR(50), quantity: INTEGER, usage: VARCHAR(50))
- Food(**name: VARCHAR(50)**, weight: FLOAT, expiryDate: DATE)
- Needs(**zoneName: VARCHAR(50), equipmentName: VARCHAR(50)**)
- HasHealthRecord(healthRecordID: INTEGER, time: DATE, eventDescription: VARCHAR(100), **creatureID: INTEGER**)
- CreaturesLivesIn(creatureID: INTEGER, species: VARCHAR(50), age: INTEGER, name: VARCHAR(50), class: VARCHAR(50), sunExposure: VARCHAR(20) , **habitatName: VARCHAR(50)**)
- HabitatsContained(name:VARCHAR(50), minTemp: FLOAT, maxTemp: FLOAT, minRh: INTEGER, maxRh: INTEGER, environment: VARCHAR(50), **zoneName: VARCHAR(50)**)
- TakesCare(**staffID: INTEGER, equipmentName: VARCHAR(50), creatureID: INTEGER**)

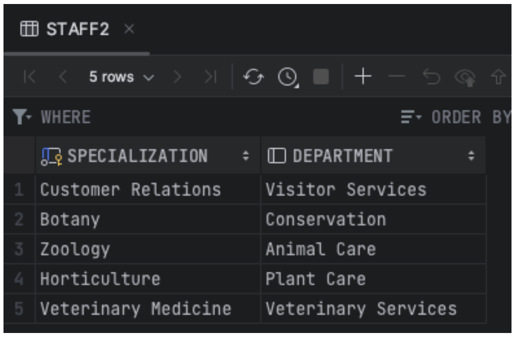
5. Copy of Schema and Data

- Staff1(staffID: INTEGER, name: VARCHAR(50), **specialization: VARCHAR(50)**, position: VARCHAR(50))



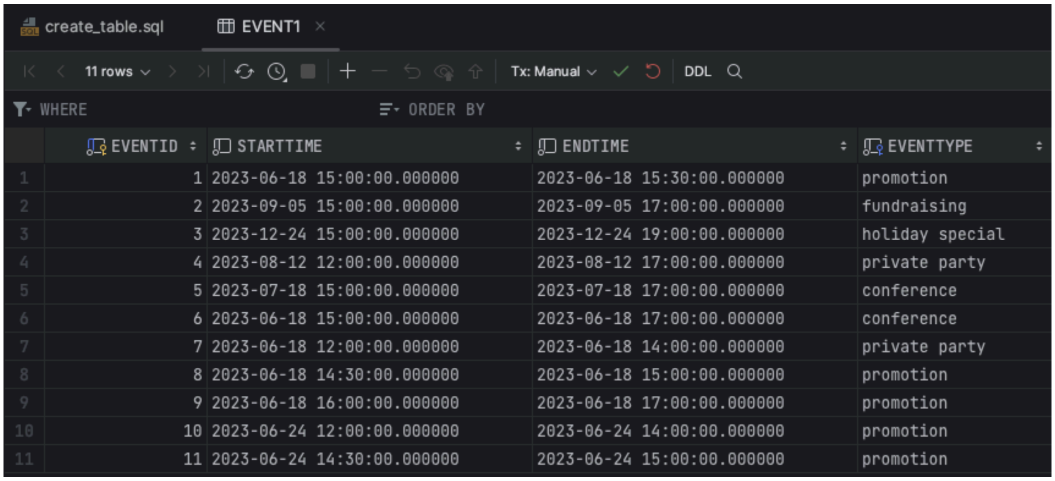
	STAFFID	NAME	SPECIALIZATION	POSITION
1	101	White Ross	Customer Relations	Customer Service
2	102	Alex Poon	Veterinary Medicine	Veterinarian
3	103	Allen Iverson	Zoology	Keeper
4	104	Taylar Swift	Customer Relations	Customer Service
5	105	Elon Musk	Horticulture	Gardener
6	106	Jasmine Kaur	Botany	Researcher

- Staff2(specialization: VARCHAR(50), department: VARCHAR(50))



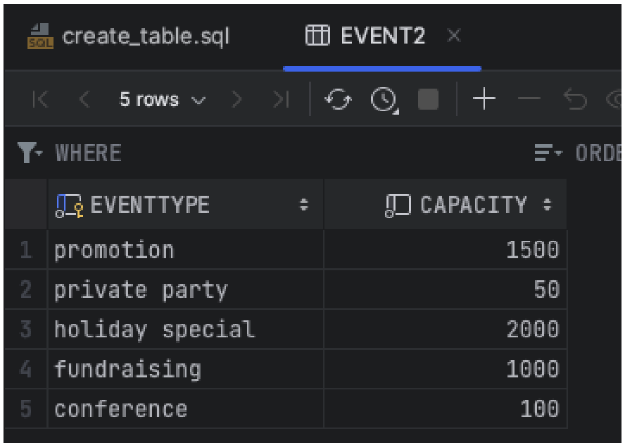
	SPECIALIZATION	DEPARTMENT
1	Customer Relations	Visitor Services
2	Botany	Conservation
3	Zoology	Animal Care
4	Horticulture	Plant Care
5	Veterinary Medicine	Veterinary Services

- Event1(eventID: INTEGER, startTime: TIMESTAMP, endTime: TIMESTAMP, **eventType: VARCHAR(30)**)



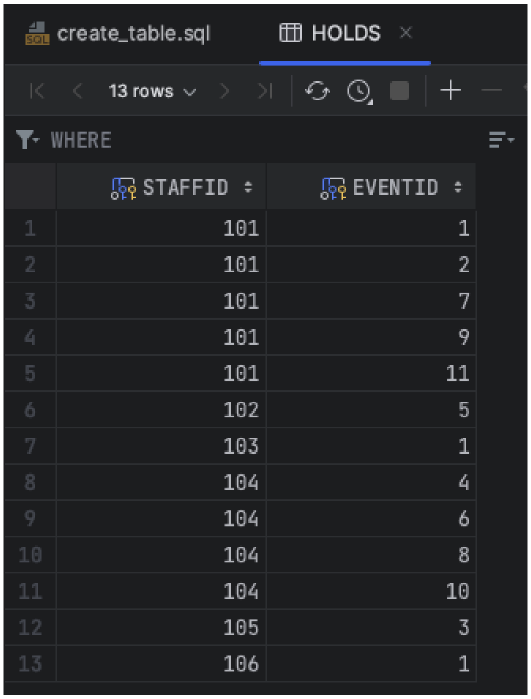
	EVENTID	STARTTIME	ENDTIME	EVENTTYPE
1	1	2023-06-18 15:00:00.000000	2023-06-18 15:30:00.000000	promotion
2	2	2023-09-05 15:00:00.000000	2023-09-05 17:00:00.000000	fundraising
3	3	2023-12-24 15:00:00.000000	2023-12-24 19:00:00.000000	holiday special
4	4	2023-08-12 12:00:00.000000	2023-08-12 17:00:00.000000	private party
5	5	2023-07-18 15:00:00.000000	2023-07-18 17:00:00.000000	conference
6	6	2023-06-18 15:00:00.000000	2023-06-18 17:00:00.000000	conference
7	7	2023-06-18 12:00:00.000000	2023-06-18 14:00:00.000000	private party
8	8	2023-06-18 14:30:00.000000	2023-06-18 15:00:00.000000	promotion
9	9	2023-06-18 16:00:00.000000	2023-06-18 17:00:00.000000	promotion
10	10	2023-06-24 12:00:00.000000	2023-06-24 14:00:00.000000	promotion
11	11	2023-06-24 14:30:00.000000	2023-06-24 15:00:00.000000	promotion

- Event2(eventType: VARCHAR(30), capacity: INTEGER)



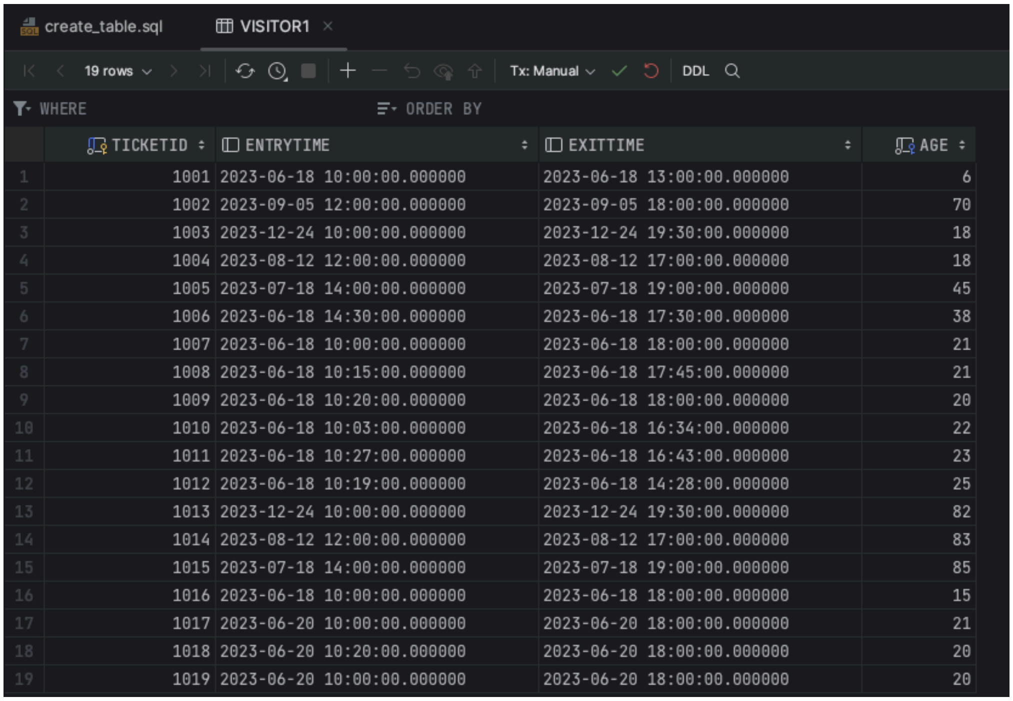
	EVENTTYPE	CAPACITY
1	promotion	1500
2	private party	50
3	holiday special	2000
4	fundraising	1000
5	conference	100

- Holds(**staffID: INTEGER, eventID: INTEGER**)



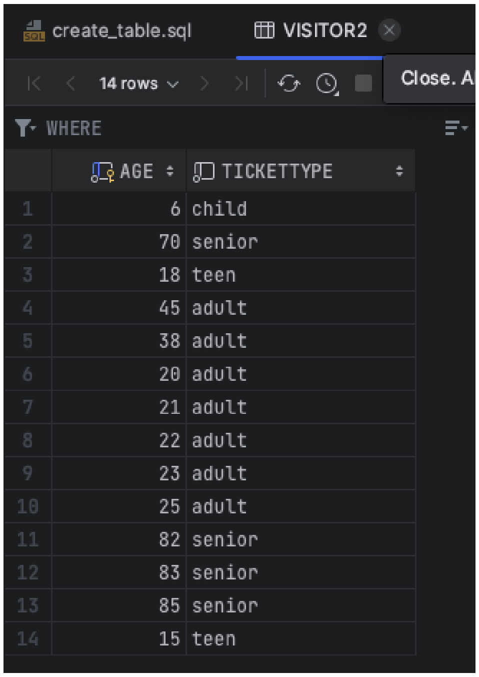
	STAFFID	EVENTID
1	101	1
2	101	2
3	101	7
4	101	9
5	101	11
6	102	5
7	103	1
8	104	4
9	104	6
10	104	8
11	104	10
12	105	3
13	106	1

- Visitor1(<u>ticketID: INTEGER</u>, entryTime: TIMESTAMP, exitTime: TIMESTAMP, **age: INTEGER**)



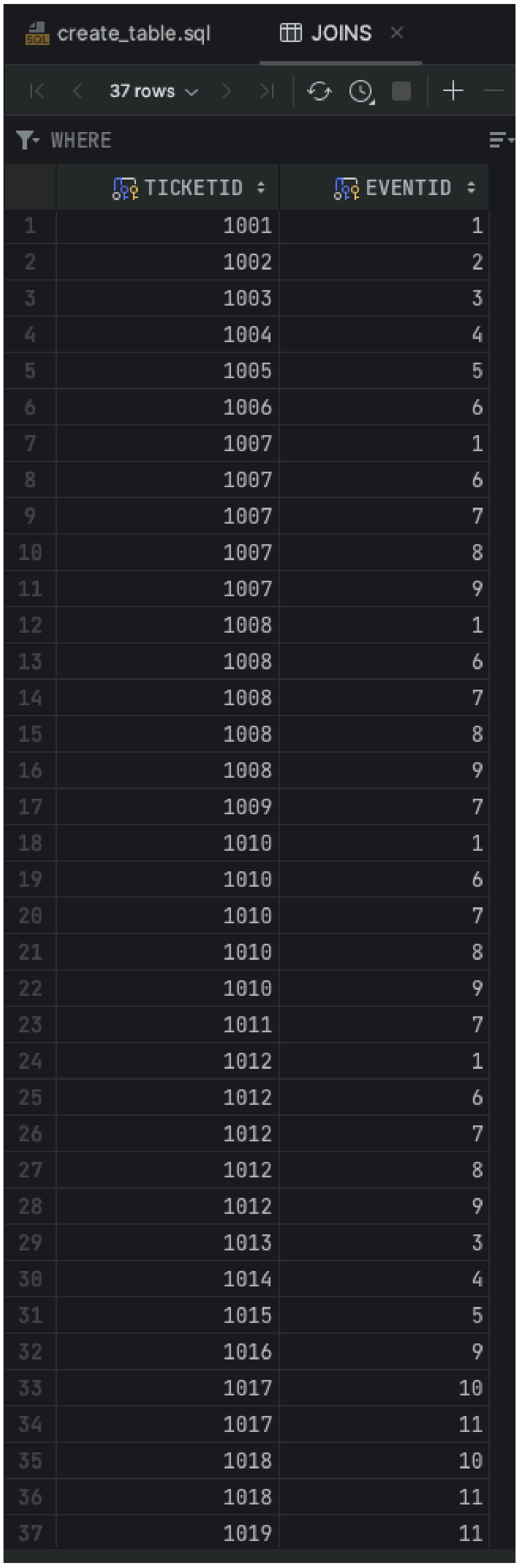
	TICKETID	ENTRYTIME	EXITTIME	AGE
1	1001	2023-06-18 10:00:00.000000	2023-06-18 13:00:00.000000	6
2	1002	2023-09-05 12:00:00.000000	2023-09-05 18:00:00.000000	70
3	1003	2023-12-24 10:00:00.000000	2023-12-24 19:30:00.000000	18
4	1004	2023-08-12 12:00:00.000000	2023-08-12 17:00:00.000000	18
5	1005	2023-07-18 14:00:00.000000	2023-07-18 19:00:00.000000	45
6	1006	2023-06-18 14:30:00.000000	2023-06-18 17:30:00.000000	38
7	1007	2023-06-18 10:00:00.000000	2023-06-18 18:00:00.000000	21
8	1008	2023-06-18 10:15:00.000000	2023-06-18 17:45:00.000000	21
9	1009	2023-06-18 10:20:00.000000	2023-06-18 18:00:00.000000	20
10	1010	2023-06-18 10:03:00.000000	2023-06-18 16:34:00.000000	22
11	1011	2023-06-18 10:27:00.000000	2023-06-18 16:43:00.000000	23
12	1012	2023-06-18 10:19:00.000000	2023-06-18 14:28:00.000000	25
13	1013	2023-12-24 10:00:00.000000	2023-12-24 19:30:00.000000	82
14	1014	2023-08-12 12:00:00.000000	2023-08-12 17:00:00.000000	83
15	1015	2023-07-18 14:00:00.000000	2023-07-18 19:00:00.000000	85
16	1016	2023-06-18 10:00:00.000000	2023-06-18 18:00:00.000000	15
17	1017	2023-06-20 10:00:00.000000	2023-06-20 18:00:00.000000	21
18	1018	2023-06-20 10:20:00.000000	2023-06-20 18:00:00.000000	20
19	1019	2023-06-20 10:00:00.000000	2023-06-20 18:00:00.000000	20

- Visitor2(<u>age: INTEGER</u>, ticketType: VARCHAR(10))



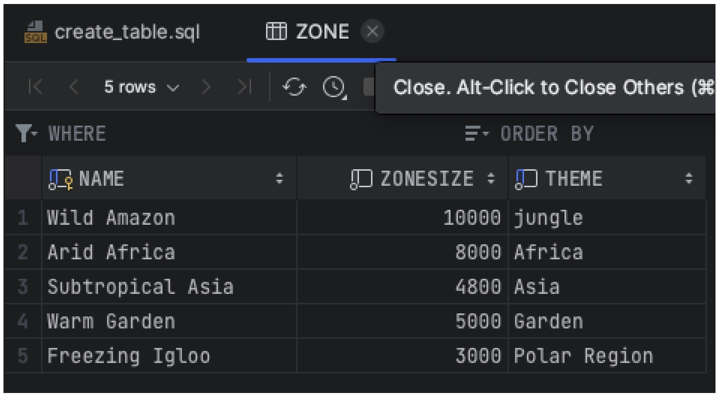
	AGE	TICKETTYPE
1	6	child
2	70	senior
3	18	teen
4	45	adult
5	38	adult
6	20	adult
7	21	adult
8	22	adult
9	23	adult
10	25	adult
11	82	senior
12	83	senior
13	85	senior
14	15	teen

- Joins(**ticketID: INTEGER, eventID: INTEGER**)



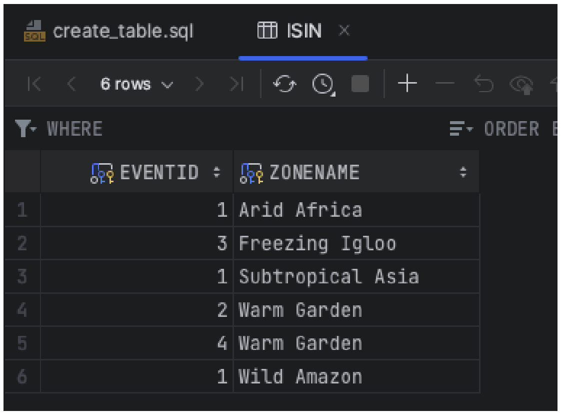
TICKETID	EVENTID
1001	1
1002	2
1003	3
1004	4
1005	5
1006	6
1007	1
1007	6
1007	7
1007	8
1007	9
1008	1
1008	6
1008	7
1008	8
1008	9
1009	7
1010	1
1010	6
1010	7
1010	8
1010	9
1011	7
1012	1
1012	6
1012	7
1012	8
1012	9
1013	3
1014	4
1015	5
1016	9
1017	10
1017	11
1018	10
1018	11
1019	11

- Zone(name: VARCHAR(50), size: INTEGER, theme: VARCHAR(50))



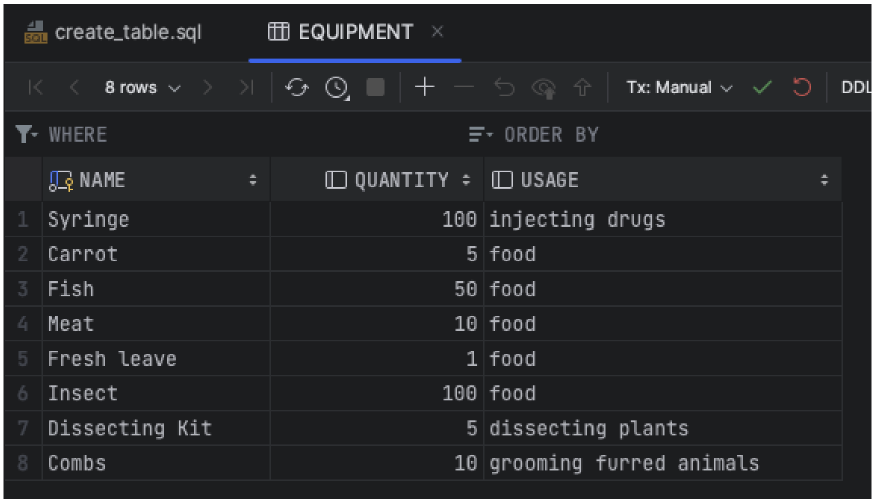
	NAME	ZONESIZE	THEME
1	Wild Amazon	10000	jungle
2	Arid Africa	8000	Africa
3	Subtropical Asia	4800	Asia
4	Warm Garden	5000	Garden
5	Freezing Igloo	3000	Polar Region

- IsIn(**eventID: INTEGER, zoneName: VARCHAR(50)**)



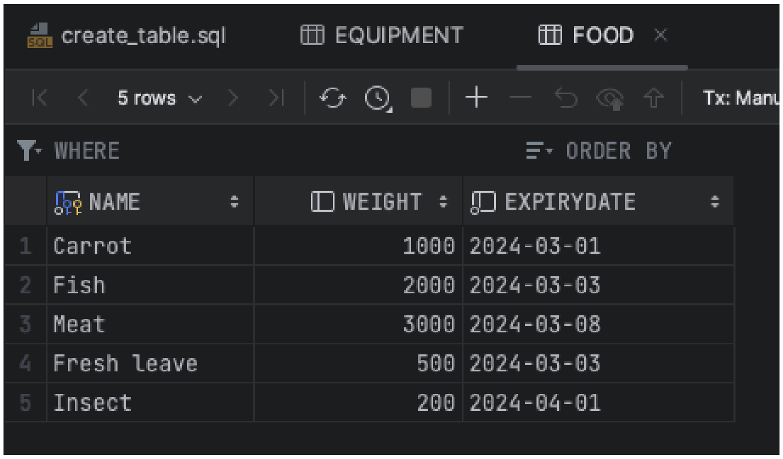
	EVENTID	ZONENAME
1	1	Arid Africa
2	3	Freezing Igloo
3	1	Subtropical Asia
4	2	Warm Garden
5	4	Warm Garden
6	1	Wild Amazon

- Equipment(name: VARCHAR(50), quantity: INTEGER, usage: VARCHAR(50))



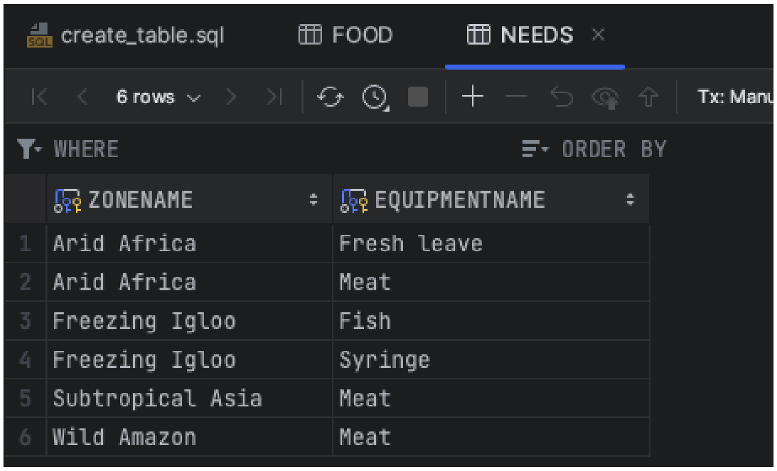
	NAME	QUANTITY	USAGE
1	Syringe	100	injecting drugs
2	Carrot	5	food
3	Fish	50	food
4	Meat	10	food
5	Fresh leave	1	food
6	Insect	100	food
7	Dissecting Kit	5	dissecting plants
8	Combs	10	grooming furred animals

- Food(**name: VARCHAR(50)**, weight: FLOAT, expiryDate: DATE)



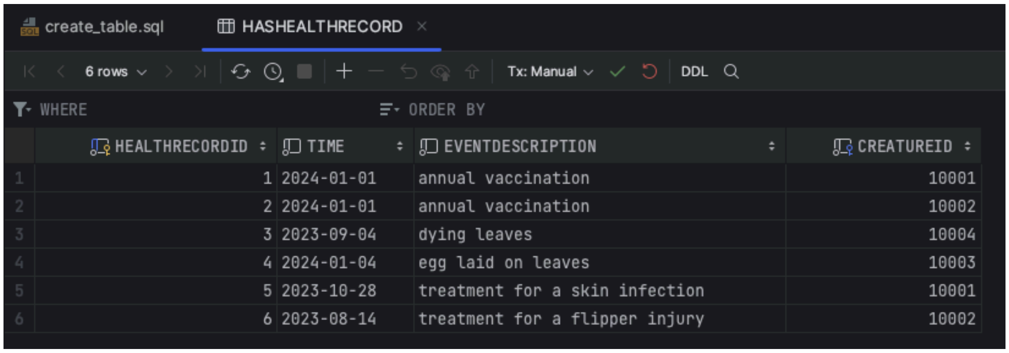
	NAME	WEIGHT	EXPIRYDATE
1	Carrot	1000	2024-03-01
2	Fish	2000	2024-03-03
3	Meat	3000	2024-03-08
4	Fresh leave	500	2024-03-03
5	Insect	200	2024-04-01

- Needs(**zoneName: VARCHAR(50), equipmentName: VARCHAR(50)**)



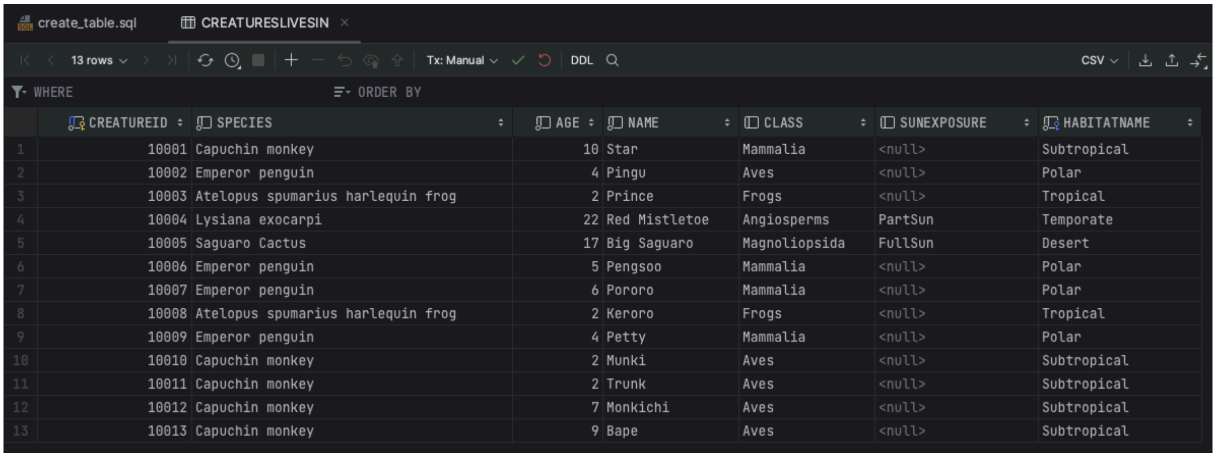
	ZONENAME	EQUIPMENTNAME
1	Arid Africa	Fresh leave
2	Arid Africa	Meat
3	Freezing Igloo	Fish
4	Freezing Igloo	Syringe
5	Subtropical Asia	Meat
6	Wild Amazon	Meat

- HasHealthRecord(healthRecordID: INTEGER, time: DATE, eventDescription: VARCHAR(100), **creatureID: INTEGER**)



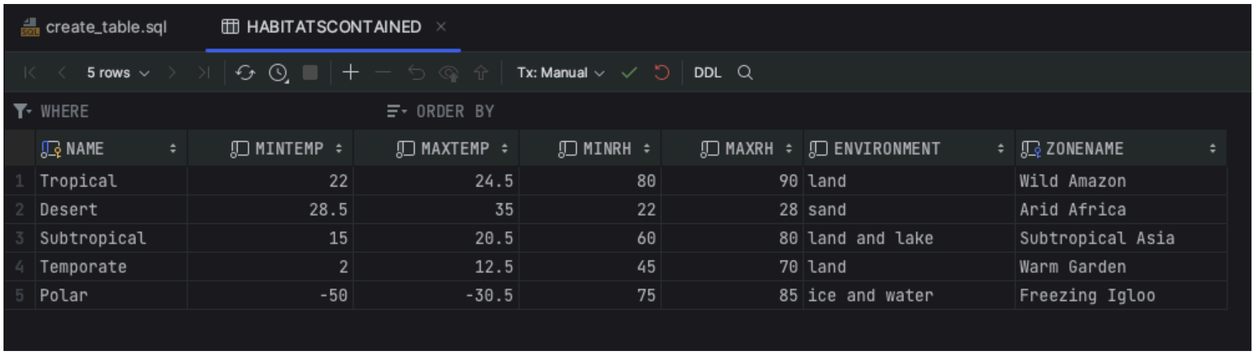
	HEALTHRECORDID	TIME	EVENTDESCRIPTION	CREATUREID
1	1	2024-01-01	annual vaccination	10001
2	2	2024-01-01	annual vaccination	10002
3	3	2023-09-04	dying leaves	10004
4	4	2024-01-04	egg laid on leaves	10003
5	5	2023-10-28	treatment for a skin infection	10001
6	6	2023-08-14	treatment for a flipper injury	10002

- CreaturesLivesIn(creatureID: INTEGER, species: VARCHAR(50), age: INTEGER, name: VARCHAR(50), class: VARCHAR(50), sunExposure: VARCHAR(20) , **habitatName: VARCHAR(50)**)



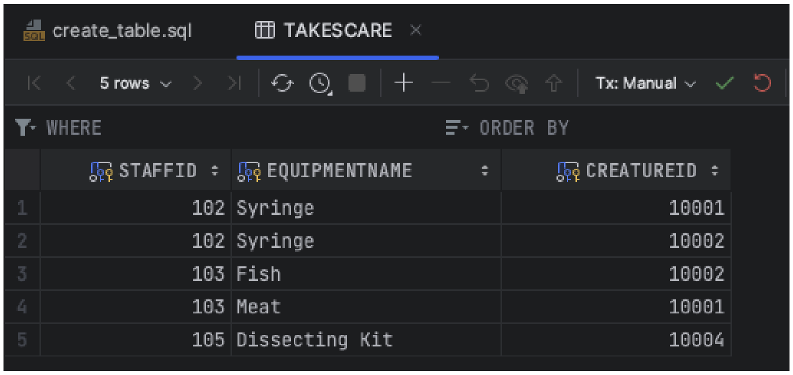
	CREATUREID	SPECIES	AGE	NAME	CLASS	SUNEXPOSURE	HABITATNAME
1	10001	Capuchin monkey	10	Star	Mammalia	<null>	Subtropical
2	10002	Emperor penguin	4	Pingu	Aves	<null>	Polar
3	10003	Atelopus spumarius harlequin frog	2	Prince	Frogs	<null>	Tropical
4	10004	Lysiana exocarpi	22	Red Mistletoe	Angiosperms	PartSun	Temporate
5	10005	Saguaro Cactus	17	Big Saguaro	Magnoliopsida	FullSun	Desert
6	10006	Emperor penguin	5	Pengsoo	Mammalia	<null>	Polar
7	10007	Emperor penguin	6	Pororo	Mammalia	<null>	Polar
8	10008	Atelopus spumarius harlequin frog	2	Keroro	Frogs	<null>	Tropical
9	10009	Emperor penguin	4	Petty	Mammalia	<null>	Polar
10	10010	Capuchin monkey	2	Munki	Aves	<null>	Subtropical
11	10011	Capuchin monkey	2	Trunk	Aves	<null>	Subtropical
12	10012	Capuchin monkey	7	Monkichi	Aves	<null>	Subtropical
13	10013	Capuchin monkey	9	Bape	Aves	<null>	Subtropical

- HabitatsContained(name:VARCHAR(50), minTemp: FLOAT, maxTemp: FLOAT, minRh: INTEGER, maxRh: INTEGER, environment: VARCHAR(50), **zoneName: VARCHAR(50)**)



	NAME	MINTEMP	MAXTEMP	MINRH	MAXRH	ENVIRONMENT	ZONENAME
1	Tropical	22	24.5	80	90	land	Wild Amazon
2	Desert	28.5	35	22	28	sand	Arid Africa
3	Subtropical	15	20.5	60	80	land and lake	Subtropical Asia
4	Temporate	2	12.5	45	70	land	Warm Garden
5	Polar	-50	-30.5	75	85	ice and water	Freezing Igloo

- TakesCare(**staffID: INTEGER, equipmentName: VARCHAR(50), creatureID: INTEGER**)



	STAFFID	EQUIPMENTNAME	CREATUREID
1	102	Syringe	10001
2	102	Syringe	10002
3	103	Fish	10002
4	103	Meat	10001
5	105	Dissecting Kit	10004

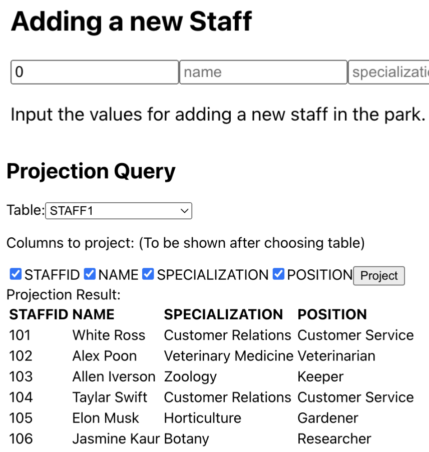
6. Queries

    There are three layers in our project. The frontend component is found in the directory `./web`. Requests are sent to the `server.js` in `./backend` and routed to `appController.js` in the same directory. The APIs defined in the `appController.js` will call functions in `appService.js` in the same directory, which is the only layer that can access the database.

    Remark: For clarity and consistency, the “Before” state is always based on the initial state of the database.

INSERT

- The UI component can be found in `./web/src/InsertInput.js`
- The API can be found on lines 74 - 93 in `./backend/appController.js`
- The service method can be found on lines 114 - 158 in `./backend/appService.js`

<table>
<tr>
<td>Before:
The following tuples are present.</td>
<td>


Adding a new Staff

0 name specialization position insert

Input the values for adding a new staff in the park.

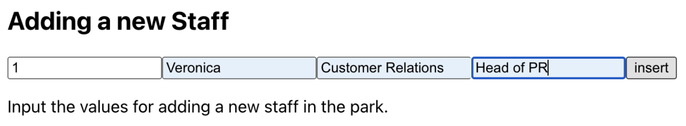
Projection Query

Table: STAFF1

Columns to project: (To be shown after choosing table)

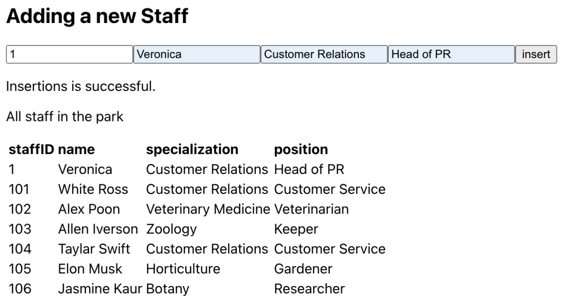
☑ STAFFID ☑ NAME ☑ SPECIALIZATION ☑ POSITION Project

Projection Result:

<table>
<tr><th>STAFFID</th><th>NAME</th><th>SPECIALIZATION</th><th>POSITION</th></tr>
<tr><td>101</td><td>White Ross</td><td>Customer Relations</td><td>Customer Service</td></tr>
<tr><td>102</td><td>Alex Poon</td><td>Veterinary Medicine</td><td>Veterinarian</td></tr>
<tr><td>103</td><td>Allen Iverson</td><td>Zoology</td><td>Keeper</td></tr>
<tr><td>104</td><td>Taylar Swift</td><td>Customer Relations</td><td>Customer Service</td></tr>
<tr><td>105</td><td>Elon Musk</td><td>Horticulture</td><td>Gardener</td></tr>
<tr><td>106</td><td>Jasmine Kaur</td><td>Botany</td><td>Researcher</td></tr>
</table>
</td>
</tr>
<tr>
<td>During:
By inputting the values, then clicking the “insert” button, the operation will be triggered.</td>
<td>


Adding a new Staff

1 Veronica Customer Relations Head of PR insert

Input the values for adding a new staff in the park.
</td>
</tr>
<tr>
<td>After:
After clicking the “insert” button, if the values are valid, the tuple will be inserted.</td>
<td>


Adding a new Staff

1 Veronica Customer Relations Head of PR insert

Insertions is successful.

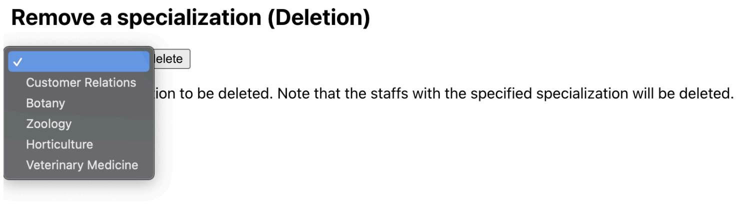
All staff in the park

<table>
<tr><th>staffID</th><th>name</th><th>specialization</th><th>position</th></tr>
<tr><td>1</td><td>Veronica</td><td>Customer Relations</td><td>Head of PR</td></tr>
<tr><td>101</td><td>White Ross</td><td>Customer Relations</td><td>Customer Service</td></tr>
<tr><td>102</td><td>Alex Poon</td><td>Veterinary Medicine</td><td>Veterinarian</td></tr>
<tr><td>103</td><td>Allen Iverson</td><td>Zoology</td><td>Keeper</td></tr>
<tr><td>104</td><td>Taylar Swift</td><td>Customer Relations</td><td>Customer Service</td></tr>
<tr><td>105</td><td>Elon Musk</td><td>Horticulture</td><td>Gardener</td></tr>
<tr><td>106</td><td>Jasmine Kaur</td><td>Botany</td><td>Researcher</td></tr>
</table>
</td>
</tr>
</table>

DELETE

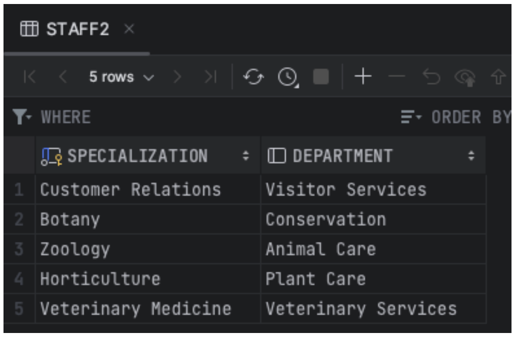
- The UI component can be found in `./web/src/DeleteInput.js`
- The APIs called by the web app can be found on lines 188 - 209 in `./backend/appController.js`
- The service method can be found on lines 160 - 200 in `./backend/appService.js`

Before:
The following tuples are present.



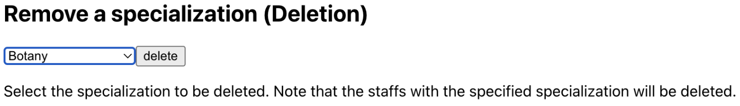
Remove a specialization (Deletion)

ion to be deleted. Note that the staffs with the specified specialization will be deleted.



	SPECIALIZATION	DEPARTMENT
1	Customer Relations	Visitor Services
2	Botany	Conservation
3	Zoology	Animal Care
4	Horticulture	Plant Care
5	Veterinary Medicine	Veterinary Services

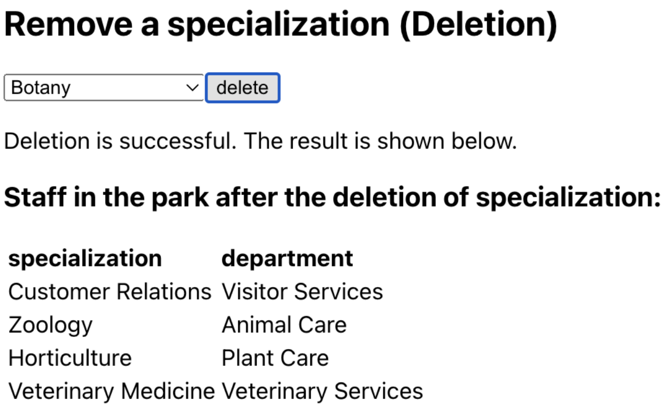
During:
By selecting the specialization to delete and hitting the “delete” button, the operation will be triggered.



Remove a specialization (Deletion)

Select the specialization to be deleted. Note that the staffs with the specified specialization will be deleted.

After:
After clicking the “delete” button, the tuple will be inserted.



Remove a specialization (Deletion)

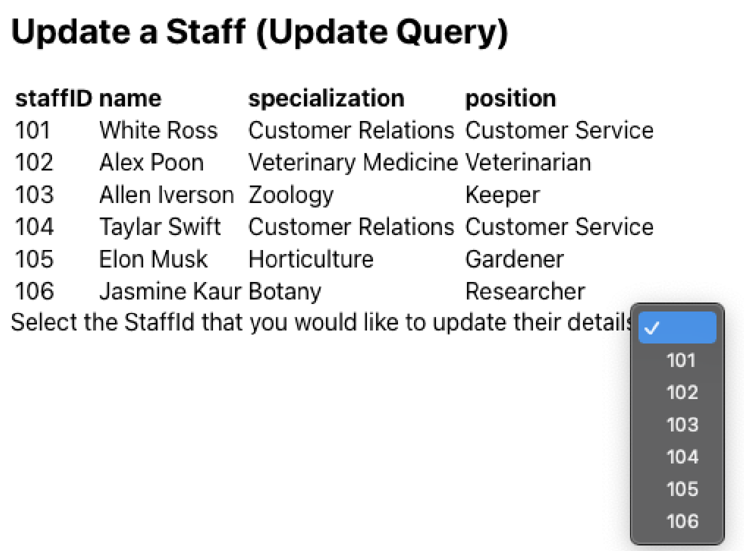
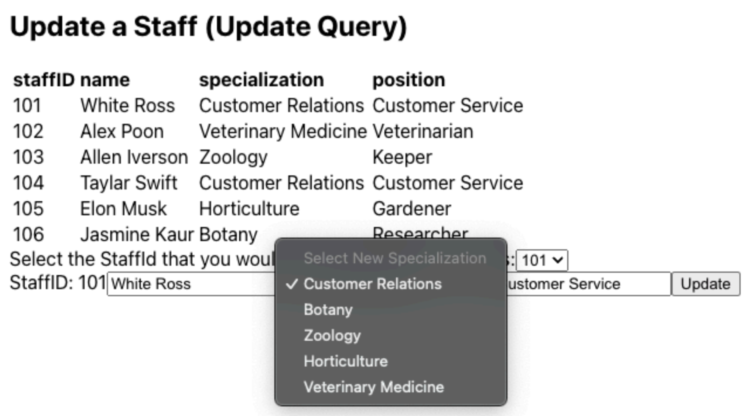
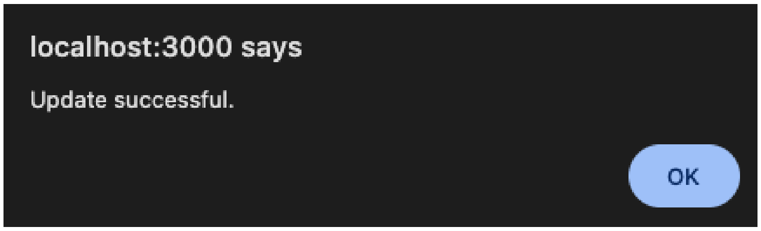
Deletion is successful. The result is shown below.

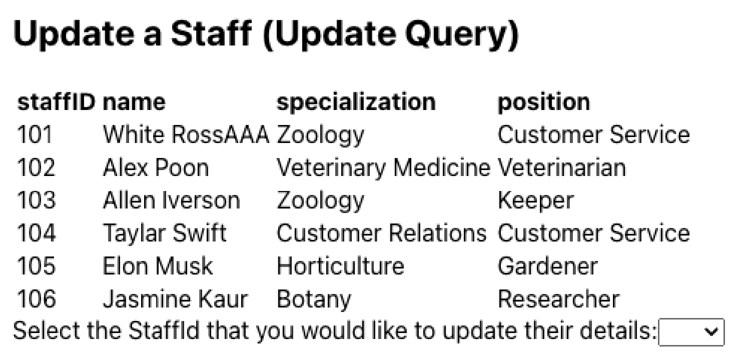
Staff in the park after the deletion of specialization:

specialization	department
Customer Relations	Visitor Services
Zoology	Animal Care
Horticulture	Plant Care
Veterinary Medicine	Veterinary Services

UPDATE

- The UI component can be found in `./web/src/UpdateQuery.js`
- The API can be found on lines 247 - 256 in `./backend/appController.js`
- The service method can be found on lines 272 - 290 in `./backend/appService.js`

Before: All tuples of table Staff1 are shown to user, and user can choose which staff to edit based on staffID.	 **Update a Staff (Update Query)** **staffID name specialization position** 101 White Ross Customer Relations Customer Service 102 Alex Poon Veterinary Medicine Veterinarian 103 Allen Iverson Zoology Keeper 104 Taylar Swift Customer Relations Customer Service 105 Elon Musk Horticulture Gardener 106 Jasmine Kaur Botany Researcher Select the StaffId that you would like to update their details
During: After selecting staffID, textboxes and dropdown lists are prepopulated with the original values, and users are able to modify them.	 **Update a Staff (Update Query)** **staffID name specialization position** 101 White Ross Customer Relations Customer Service 102 Alex Poon Veterinary Medicine Veterinarian 103 Allen Iverson Zoology Keeper 104 Taylar Swift Customer Relations Customer Service 105 Elon Musk Horticulture Gardener 106 Jasmine Kaur Botany Researcher Select the StaffId that you woul s: 101 StaffID: 101 White Ross ustomer Service Update Select New Specialization ✓ Customer Relations Botany Zoology Horticulture Veterinary Medicine
After: After clicking update, there will be a pop-up message indicating the action is successful or not, if yes the page will refresh and load the latest table Staff1 for users to confirm the changes	 localhost:3000 says Update successful.

<table>
<tr>
<td>have been applied to the table.</td>
<td>


Update a Staff (Update Query)

<table>
<tr><th>staffID</th><th>name</th><th>specialization</th><th>position</th></tr>
<tr><td>101</td><td>White RossAAA</td><td>Zoology</td><td>Customer Service</td></tr>
<tr><td>102</td><td>Alex Poon</td><td>Veterinary Medicine</td><td>Veterinarian</td></tr>
<tr><td>103</td><td>Allen Iverson</td><td>Zoology</td><td>Keeper</td></tr>
<tr><td>104</td><td>Taylar Swift</td><td>Customer Relations</td><td>Customer Service</td></tr>
<tr><td>105</td><td>Elon Musk</td><td>Horticulture</td><td>Gardener</td></tr>
<tr><td>106</td><td>Jasmine Kaur</td><td>Botany</td><td>Researcher</td></tr>
</table>

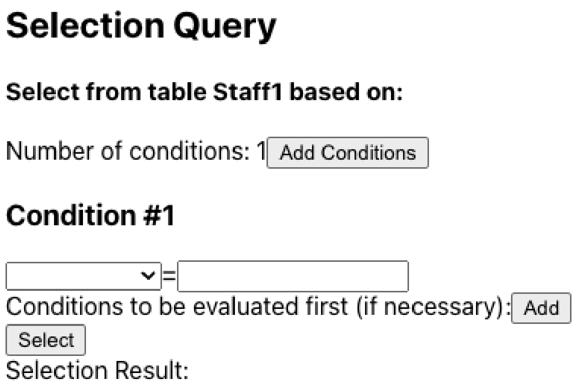
Select the StaffId that you would like to update their details:
</td>
</tr>
</table>

Selection

- The UI component can be found in `./web/src/SelectionQuery.js`
- The API can be found on lines 99 - 113 in `./backend/appController.js`
- The service method can be found on lines 292 - 301 in `./backend/appService.js`

Before:
A page is loaded such that users can add conditions and brackets on conditions that should be evaluated first.

The second picture shows all tuples present before selection.



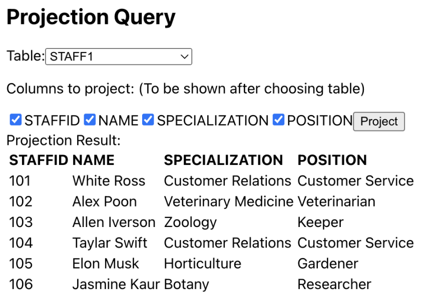
Selection Query

Select from table Staff1 based on:

Number of conditions: 1

Condition #1

Selection Result:



Projection Query

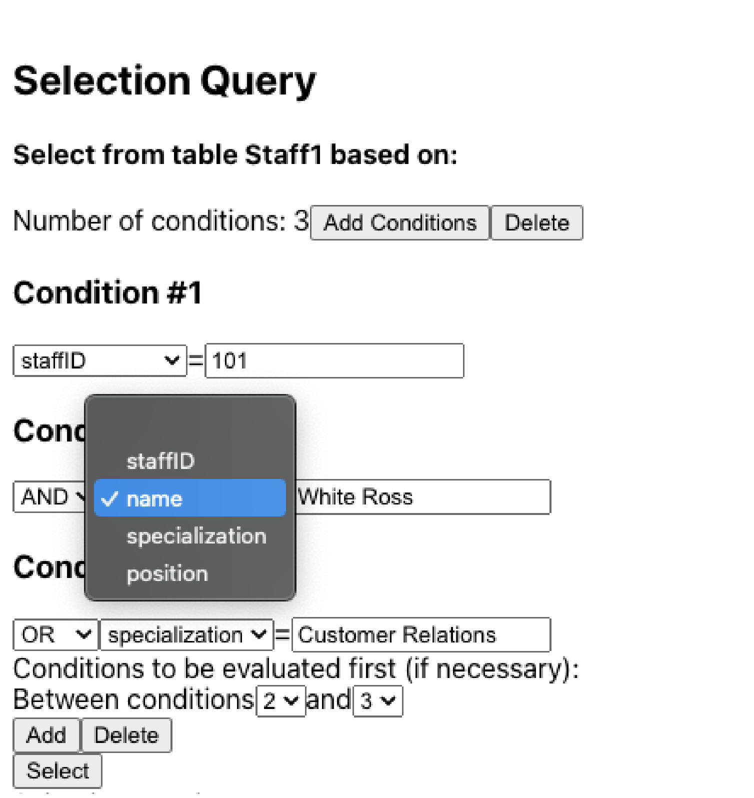
Table: STAFF1

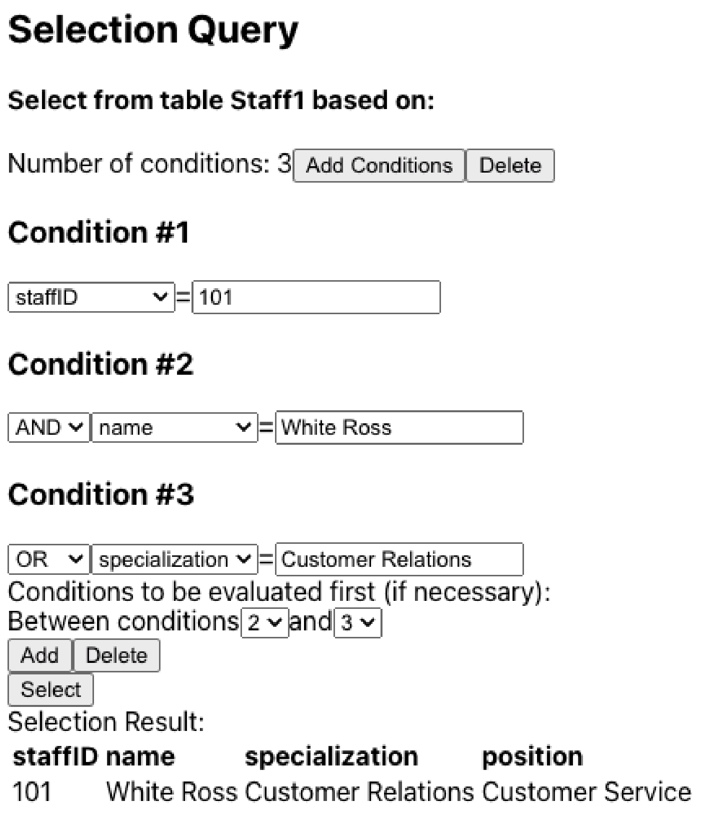
Columns to project: (To be shown after choosing table)

☑ STAFFID ☑ NAME ☑ SPECIALIZATION ☑ POSITION

Projection Result:

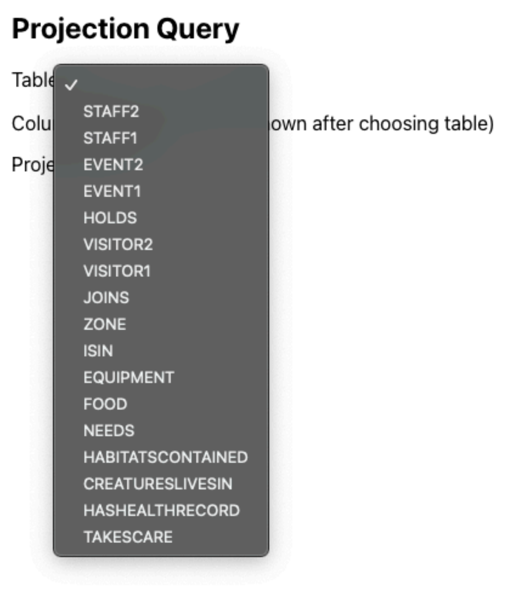
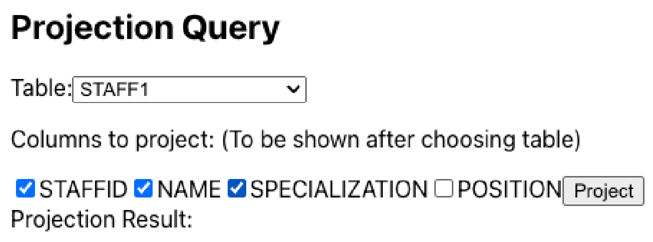
STAFFID	NAME	SPECIALIZATION	POSITION
101	White Ross	Customer Relations	Customer Service
102	Alex Poon	Veterinary Medicine	Veterinarian
103	Allen Iverson	Zoology	Keeper
104	Taylar Swift	Customer Relations	Customer Service
105	Elon Musk	Horticulture	Gardener
106	Jasmine Kaur	Botany	Researcher

During: Users are able to add conditions based on equality, with dropdown lists to choose appropriate columns and operators and textboxes for values, and add brackets on different conditions that should be evaluated first.	 **Selection Query** **Select from table Staff1 based on:** Number of conditions: 3 Add Conditions Delete **Condition #1** staffID = 101 **Cond** AND name = White Ross **Cond** OR specialization = Customer Relations Conditions to be evaluated first (if necessary): Between conditions 2 and 3 Add Delete Select

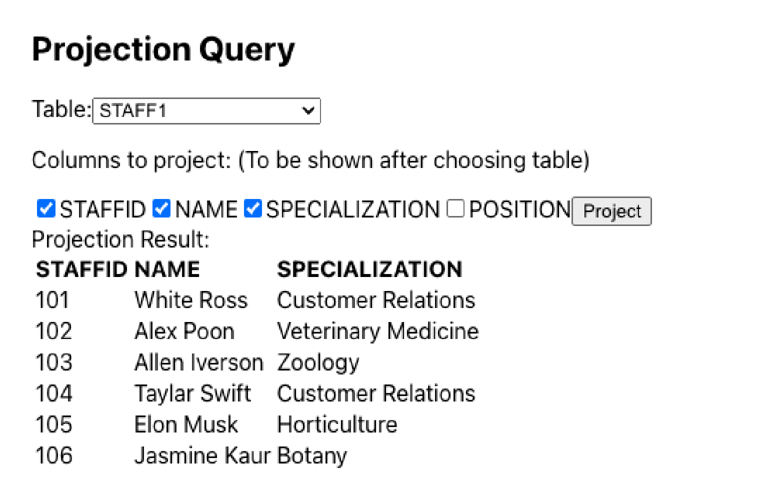
After: After clicking the “Select” button, the required selection will be done and presented in a table.	 **Selection Query** **Select from table Staff1 based on:** Number of conditions: 3 **Condition #1** staffID = 101 **Condition #2** AND name = White Ross **Condition #3** OR specialization = Customer Relations Conditions to be evaluated first (if necessary): Between conditions 2 and 3 Selection Result: **staffID** **name** **specialization** **position** 101 White Ross Customer Relations Customer Service

Projection

- The UI component can be found in `./web/src/ProjectQuery.js`
- The API can be found on lines 20 - 72 in `./backend/appController.js`
- The service method can be found on lines 82 - 112 in `./backend/appService.js`

Before: All table names are available for users to choose from.	 **Projection Query** Table Colu own after choosing table) Proje ✓ STAFF2 STAFF1 EVENT2 EVENT1 HOLDS VISITOR2 VISITOR1 JOINS ZONE ISIN EQUIPMENT FOOD NEEDS HABITATSCONTAINED CREATURESLIVESIN HASHEALTHRECORD TAKESCARE
During: User will be able to choose column(s) to project	 **Projection Query** Table: STAFF1 Columns to project: (To be shown after choosing table) ☑ STAFFID ☑ NAME ☑ SPECIALIZATION ☐ POSITION Projection Result:

After:
After clicking the “Project” button, the required projection will be done and presented in the table.



Projection Query

Table: STAFF1

Columns to project: (To be shown after choosing table)

☑ STAFFID ☑ NAME ☑ SPECIALIZATION □ POSITION

Projection Result:

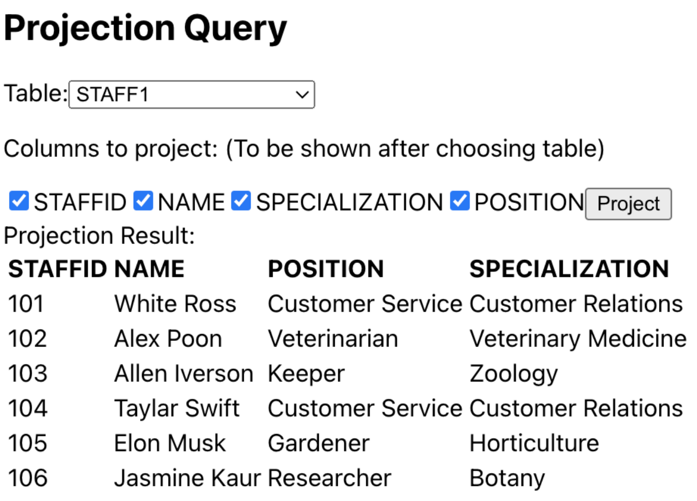
STAFFID	NAME	SPECIALIZATION
101	White Ross	Customer Relations
102	Alex Poon	Veterinary Medicine
103	Allen Iverson	Zoology
104	Taylar Swift	Customer Relations
105	Elon Musk	Horticulture
106	Jasmine Kaur	Botany

Join

- The UI component can be found in `./web/src/StaffForEventQuery.js`
- The API can be found on lines 122 - 137 in `./backend/appController.js`
- The service method can be found on lines 310 - 327 in `./backend/appService.js`

Before:
There are two separated tables, Staff1 and Holds in the database. All the tuples of Staff1 and Holds stored in the database are shown as the images on the right hand side.

A page is loaded and there is a box for eventID entry



Projection Query

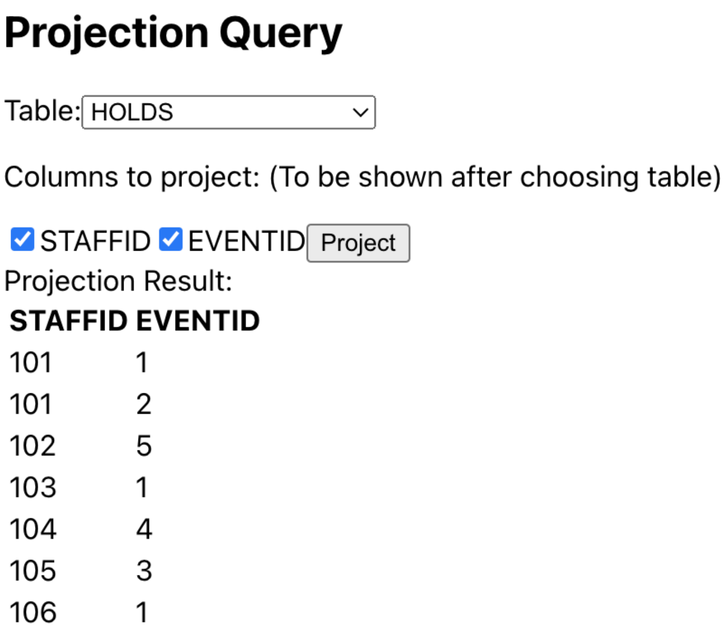
Table: STAFF1

Columns to project: (To be shown after choosing table)

☑ STAFFID ☑ NAME ☑ SPECIALIZATION ☑ POSITION

Projection Result:

STAFFID	NAME	POSITION	SPECIALIZATION
101	White Ross	Customer Service	Customer Relations
102	Alex Poon	Veterinarian	Veterinary Medicine
103	Allen Iverson	Keeper	Zoology
104	Taylar Swift	Customer Service	Customer Relations
105	Elon Musk	Gardener	Horticulture
106	Jasmine Kaur	Researcher	Botany



Projection Query

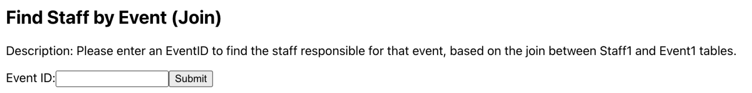
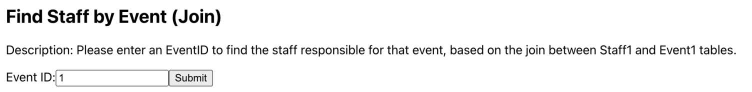
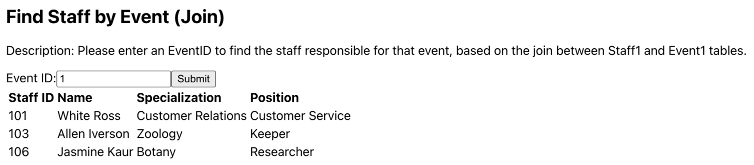
Table: HOLDS

Columns to project: (To be shown after choosing table)

☑ STAFFID ☑ EVENTID

Projection Result:

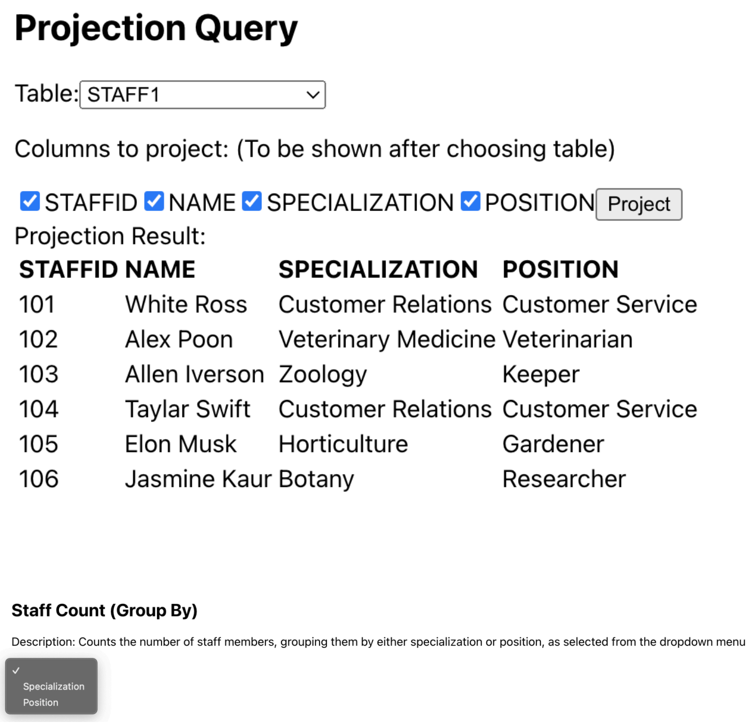
STAFFID	EVENTID
101	1
101	2
102	5
103	1
104	4
105	3
106	1

	 **Find Staff by Event (Join)** Description: Please enter an EventID to find the staff responsible for that event, based on the join between Staff1 and Event1 tables. Event ID: Submit
During: Users will be able to enter any eventID they want	 **Find Staff by Event (Join)** Description: Please enter an EventID to find the staff responsible for that event, based on the join between Staff1 and Event1 tables. Event ID: 1 Submit
After: After clicking the "Submit" button, the required staff1 tuples will be presented in the table.	 **Find Staff by Event (Join)** Description: Please enter an EventID to find the staff responsible for that event, based on the join between Staff1 and Event1 tables. Event ID: 1 Submit **Staff ID Name Specialization Position** 101 White Ross Customer Relations Customer Service 103 Allen Iverson Zoology Keeper 106 Jasmine Kaur Botany Researcher

<u>Aggregation with Group By</u>

- The UI component can be found in `./web/src/GroupCounterQuery.js`
- The API can be found on lines 146 - 161 in `./backend/appController.js`
- The service method can be found on lines 336 - 351 in `./backend/appService.js`

<table>
<tr>
<td>Before:
All the tuples of Staff1 stored in the database are shown as the image on the right hand side.

A page is loaded and there is a dropdown table allowing user to pick either “Specialization” or “Position”</td>
<td>

Projection Query

Table: STAFF1

Columns to project: (To be shown after choosing table)

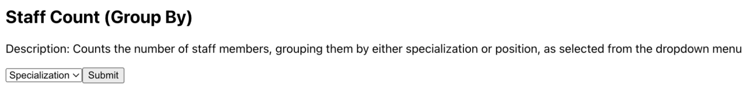
☑ STAFFID ☑ NAME ☑ SPECIALIZATION ☑ POSITION

Projection Result:

<table>
<tr><th>STAFFID</th><th>NAME</th><th>SPECIALIZATION</th><th>POSITION</th></tr>
<tr><td>101</td><td>White Ross</td><td>Customer Relations</td><td>Customer Service</td></tr>
<tr><td>102</td><td>Alex Poon</td><td>Veterinary Medicine</td><td>Veterinarian</td></tr>
<tr><td>103</td><td>Allen Iverson</td><td>Zoology</td><td>Keeper</td></tr>
<tr><td>104</td><td>Taylar Swift</td><td>Customer Relations</td><td>Customer Service</td></tr>
<tr><td>105</td><td>Elon Musk</td><td>Horticulture</td><td>Gardener</td></tr>
<tr><td>106</td><td>Jasmine Kaur</td><td>Botany</td><td>Researcher</td></tr>
</table>

Staff Count (Group By)

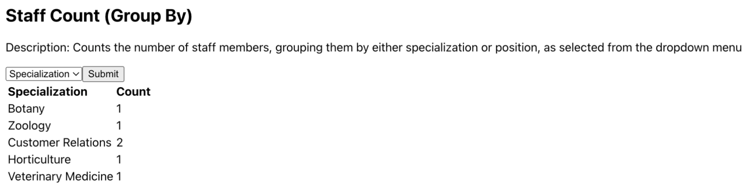
Description: Counts the number of staff members, grouping them by either specialization or position, as selected from the dropdown menu

</td>
</tr>
<tr>
<td>During:
After picking either “Specialization” or “Position”, the user will be able to click the “Submit” button.</td>
<td>

Staff Count (Group By)

Description: Counts the number of staff members, grouping them by either specialization or position, as selected from the dropdown menu

Specialization

</td>
</tr>
<tr>
<td>After:
After clicking the “Submit” button, the number of staff belonging to each group will be presented in the table.</td>
<td>

Staff Count (Group By)

Description: Counts the number of staff members, grouping them by either specialization or position, as selected from the dropdown menu

Specialization

<table>
<tr><th>Specialization</th><th>Count</th></tr>
<tr><td>Botany</td><td>1</td></tr>
<tr><td>Zoology</td><td>1</td></tr>
<tr><td>Customer Relations</td><td>2</td></tr>
<tr><td>Horticulture</td><td>1</td></tr>
<tr><td>Veterinary Medicine</td><td>1</td></tr>
</table>

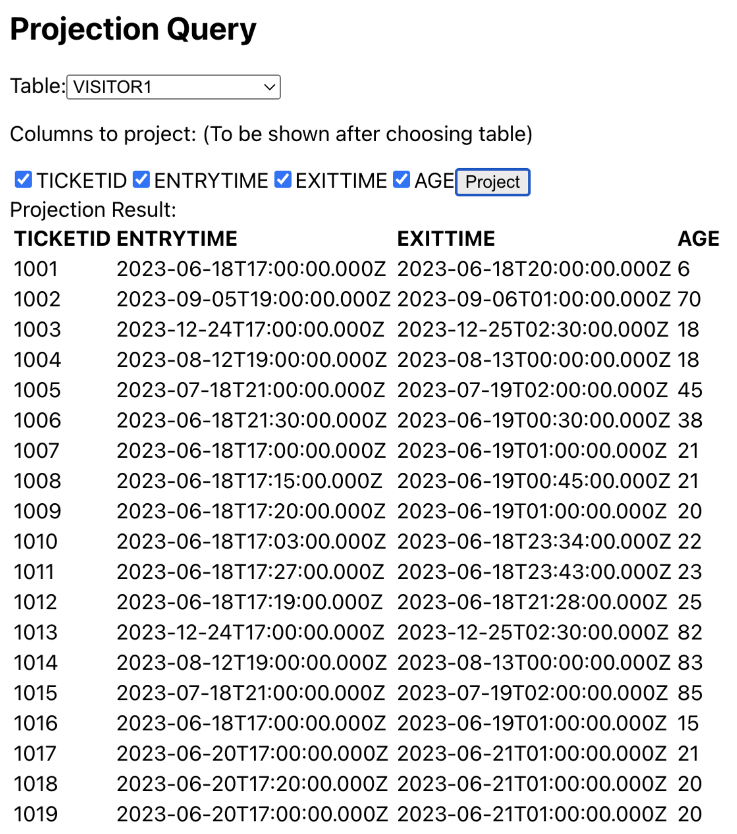
</td>
</tr>
</table>

Aggregation with Having

- The UI component can be found in `./web/src/AgeAggregationQuery.js`
- The API can be found on lines 170 - 186 in `./backend/appController.js`
- The service method can be found on lines 360 - 379 in `./backend/appService.js`

Before:
There are two separated tables, Vistor1 and Vistor2 in the database. All the tuples of Vistor1 and Vistor2 stored in the database are shown as the images on the right hand side.

A page is loaded and there is a dropdown table allowing user to pick either "Maximum", "Minimum" or "Average"



Projection Query

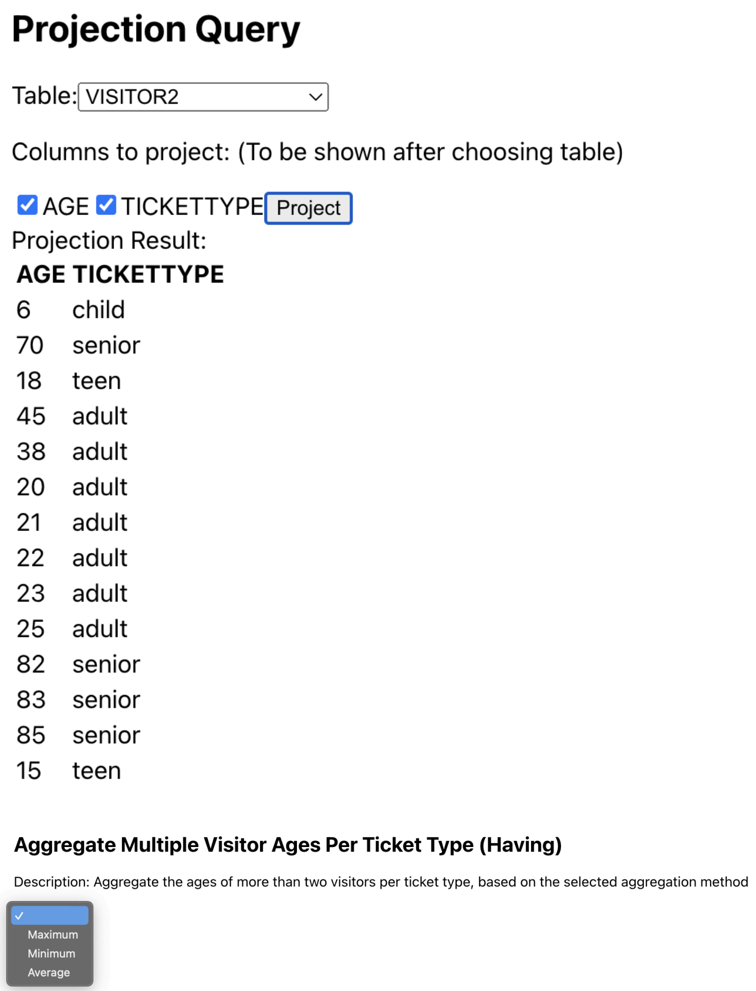
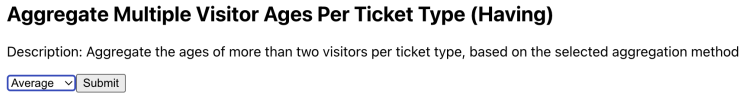
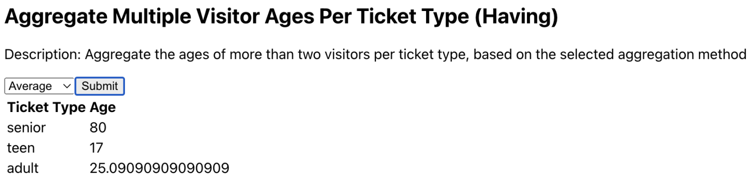
Table: VISITOR1

Columns to project: (To be shown after choosing table)

☑ TICKETID ☑ ENTRYTIME ☑ EXITTIME ☑ AGE

Projection Result:

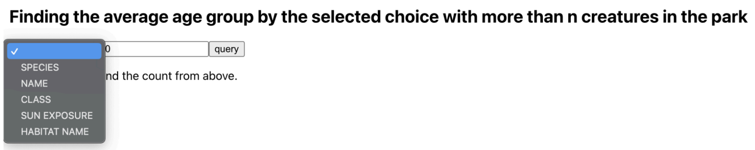
TICKETID	ENTRYTIME	EXITTIME	AGE
1001	2023-06-18T17:00:00.000Z	2023-06-18T20:00:00.000Z	6
1002	2023-09-05T19:00:00.000Z	2023-09-06T01:00:00.000Z	70
1003	2023-12-24T17:00:00.000Z	2023-12-25T02:30:00.000Z	18
1004	2023-08-12T19:00:00.000Z	2023-08-13T00:00:00.000Z	18
1005	2023-07-18T21:00:00.000Z	2023-07-19T02:00:00.000Z	45
1006	2023-06-18T21:30:00.000Z	2023-06-19T00:30:00.000Z	38
1007	2023-06-18T17:00:00.000Z	2023-06-19T01:00:00.000Z	21
1008	2023-06-18T17:15:00.000Z	2023-06-19T00:45:00.000Z	21
1009	2023-06-18T17:20:00.000Z	2023-06-19T01:00:00.000Z	20
1010	2023-06-18T17:03:00.000Z	2023-06-18T23:34:00.000Z	22
1011	2023-06-18T17:27:00.000Z	2023-06-18T23:43:00.000Z	23
1012	2023-06-18T17:19:00.000Z	2023-06-18T21:28:00.000Z	25
1013	2023-12-24T17:00:00.000Z	2023-12-25T02:30:00.000Z	82
1014	2023-08-12T19:00:00.000Z	2023-08-13T00:00:00.000Z	83
1015	2023-07-18T21:00:00.000Z	2023-07-19T02:00:00.000Z	85
1016	2023-06-18T17:00:00.000Z	2023-06-19T01:00:00.000Z	15
1017	2023-06-20T17:00:00.000Z	2023-06-21T01:00:00.000Z	21
1018	2023-06-20T17:20:00.000Z	2023-06-21T01:00:00.000Z	20
1019	2023-06-20T17:00:00.000Z	2023-06-21T01:00:00.000Z	20

	 **Projection Query** Table: VISITOR2 Columns to project: (To be shown after choosing table) ☑ AGE ☑ TICKETTYPE Project Projection Result: **AGE TICKETTYPE** 6 child 70 senior 18 teen 45 adult 38 adult 20 adult 21 adult 22 adult 23 adult 25 adult 82 senior 83 senior 85 senior 15 teen **Aggregate Multiple Visitor Ages Per Ticket Type (Having)** Description: Aggregate the ages of more than two visitors per ticket type, based on the selected aggregation method
During: After picking either “Maximum”, “Minimum” or “Average”, the user will be able to click the “Submit” button.	 **Aggregate Multiple Visitor Ages Per Ticket Type (Having)** Description: Aggregate the ages of more than two visitors per ticket type, based on the selected aggregation method Average Submit
After: After clicking the “Submit” button, a table aggregating the ages of more than two visitors per ticket type, based on the selected aggregation method will be presented	 **Aggregate Multiple Visitor Ages Per Ticket Type (Having)** Description: Aggregate the ages of more than two visitors per ticket type, based on the selected aggregation method Average Submit **Ticket Type Age** senior 80 teen 17 adult 25.09090909090909

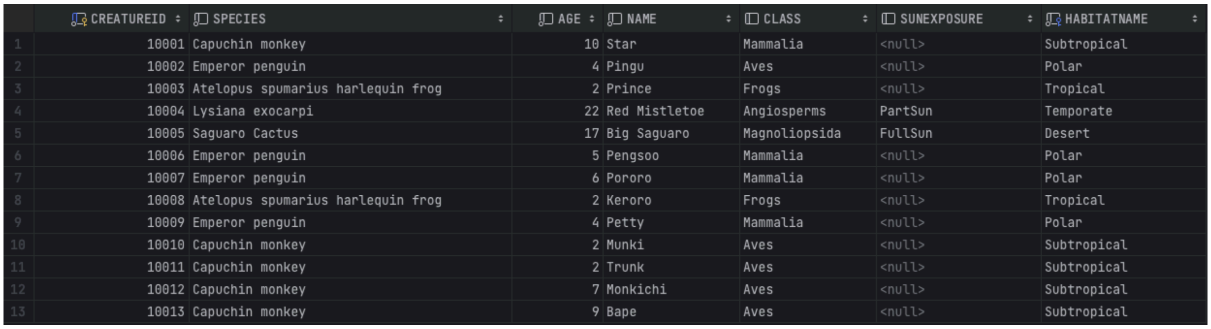
Nested aggregation with Group By

- The UI component can be found in `./web/src/NestedAggregationQuery.js`
- The API can be found on lines 229 - 246 in `./backend/appController.js`
- The service method can be found on lines 241 - 267 in `./backend/appService.js`

Before:
The following tuples are present.

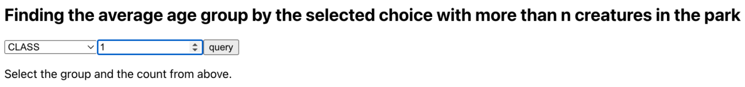


Finding the average age group by the selected choice with more than n creatures in the park



	CREATUREID	SPECIES	AGE	NAME	CLASS	SUNEXPOSURE	HABITATNAME
1	10001	Capuchin monkey	10	Star	Mammalia	<null>	Subtropical
2	10002	Emperor penguin	4	Pingu	Aves	<null>	Polar
3	10003	Atelopus spumarius harlequin frog	2	Prince	Frogs	<null>	Tropical
4	10004	Lysiana exocarpi	22	Red Mistletoe	Angiosperms	PartSun	Temporate
5	10005	Saguaro Cactus	17	Big Saguaro	Magnoliopsida	FullSun	Desert
6	10006	Emperor penguin	5	Pengsoo	Mammalia	<null>	Polar
7	10007	Emperor penguin	6	Pororo	Mammalia	<null>	Polar
8	10008	Atelopus spumarius harlequin frog	2	Keroro	Frogs	<null>	Tropical
9	10009	Emperor penguin	4	Petty	Mammalia	<null>	Polar
10	10010	Capuchin monkey	2	Munki	Aves	<null>	Subtropical
11	10011	Capuchin monkey	2	Trunk	Aves	<null>	Subtropical
12	10012	Capuchin monkey	7	Monkichi	Aves	<null>	Subtropical
13	10013	Capuchin monkey	9	Bape	Aves	<null>	Subtropical

During:
By selecting the group by attribute and the count, then clicking the “query” button, the operation will be triggered.

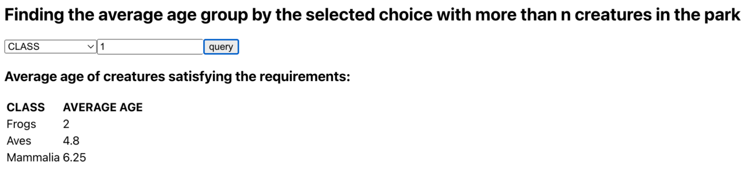


Finding the average age group by the selected choice with more than n creatures in the park

CLASS | 1 | query

Select the group and the count from above.

After:
After clicking the “query” button, the query result is shown.



Finding the average age group by the selected choice with more than n creatures in the park

CLASS | 1 | query

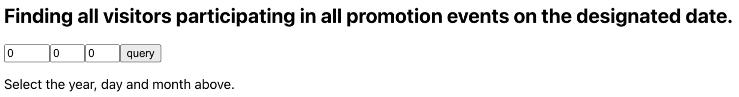
Average age of creatures satisfying the requirements:

CLASS	AVERAGE AGE
Frogs	2
Aves	4.8
Mammalia	6.25

Division

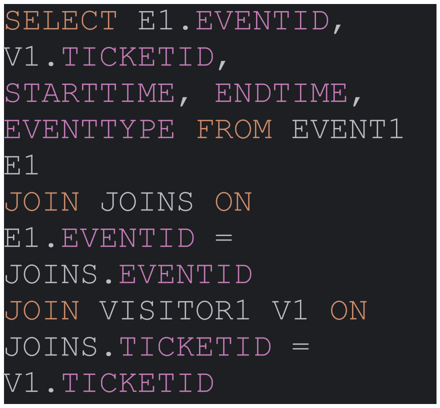
- The UI component can be found in `./web/src/DivisionQuery.js`
- The API can be found on lines 211 - 227 in `./backend/appController.js`
- The service method can be found on lines 206 - 239 in `./backend/appService.js`

Before:
The following tuples are present.

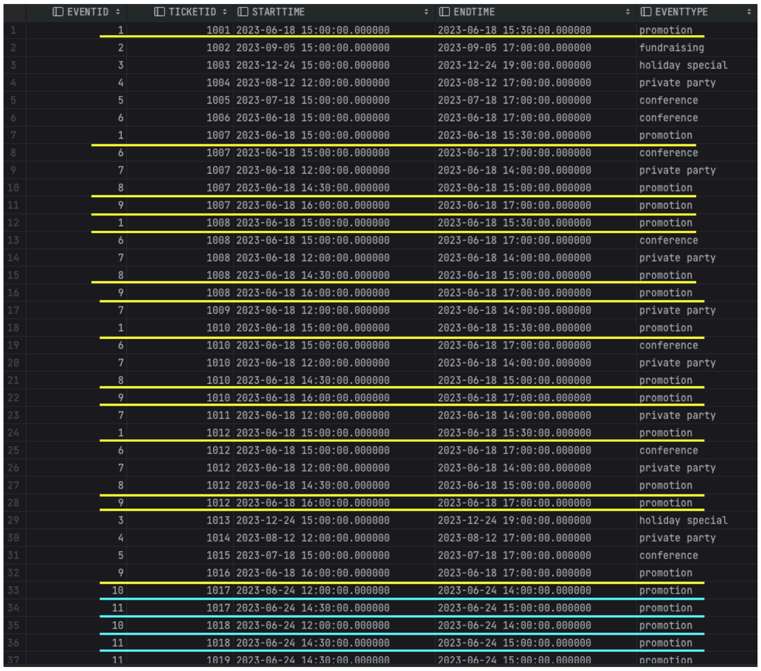


Finding all visitors participating in all promotion events on the designated date.

Select the year, day and month above.



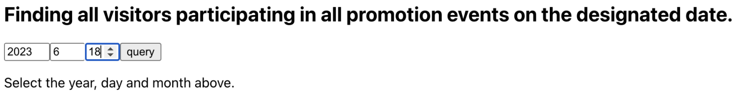
```
SELECT E1.EVENTID,
V1.TICKETID,
STARTTIME, ENDTIME,
EVENTTYPE FROM EVENT1
E1
JOIN JOINS ON
E1.EVENTID =
JOINS.EVENTID
JOIN VISITOR1 V1 ON
JOINS.TICKETID =
V1.TICKETID
```



EVENTID	TICKETID	STARTTIME	ENDTIME	EVENTTYPE
1	1001	2023-06-18 15:00:00.000000	2023-06-18 15:30:00.000000	promotion
2	1002	2023-09-05 15:00:00.000000	2023-09-05 17:00:00.000000	fundraising
3	1003	2023-12-24 15:00:00.000000	2023-12-24 19:00:00.000000	holiday special
4	1004	2023-08-12 12:00:00.000000	2023-08-12 17:00:00.000000	private party
5	1005	2023-07-18 15:00:00.000000	2023-07-18 17:00:00.000000	conference
6	1006	2023-06-18 15:00:00.000000	2023-06-18 17:00:00.000000	conference
1	1007	2023-06-18 15:00:00.000000	2023-06-18 15:30:00.000000	promotion
6	1007	2023-06-18 15:00:00.000000	2023-06-18 17:00:00.000000	conference
7	1007	2023-06-18 12:00:00.000000	2023-06-18 14:00:00.000000	private party
8	1007	2023-06-18 14:30:00.000000	2023-06-18 15:00:00.000000	promotion
9	1007	2023-06-18 16:00:00.000000	2023-06-18 17:00:00.000000	promotion
1	1008	2023-06-18 15:00:00.000000	2023-06-18 15:30:00.000000	promotion
6	1008	2023-06-18 15:00:00.000000	2023-06-18 17:00:00.000000	conference
7	1008	2023-06-18 12:00:00.000000	2023-06-18 14:00:00.000000	private party
8	1008	2023-06-18 14:30:00.000000	2023-06-18 15:00:00.000000	promotion
9	1008	2023-06-18 16:00:00.000000	2023-06-18 17:00:00.000000	promotion
7	1009	2023-06-18 12:00:00.000000	2023-06-18 14:00:00.000000	private party
1	1010	2023-06-18 15:00:00.000000	2023-06-18 15:30:00.000000	promotion
6	1010	2023-06-18 15:00:00.000000	2023-06-18 17:00:00.000000	conference
7	1010	2023-06-18 12:00:00.000000	2023-06-18 14:00:00.000000	private party
8	1010	2023-06-18 14:30:00.000000	2023-06-18 15:00:00.000000	promotion
9	1010	2023-06-18 16:00:00.000000	2023-06-18 17:00:00.000000	promotion
7	1011	2023-06-18 12:00:00.000000	2023-06-18 14:00:00.000000	private party
1	1012	2023-06-18 15:00:00.000000	2023-06-18 15:30:00.000000	promotion
6	1012	2023-06-18 15:00:00.000000	2023-06-18 17:00:00.000000	conference
7	1012	2023-06-18 12:00:00.000000	2023-06-18 14:00:00.000000	private party
8	1012	2023-06-18 14:30:00.000000	2023-06-18 15:00:00.000000	promotion
9	1012	2023-06-18 16:00:00.000000	2023-06-18 17:00:00.000000	promotion
3	1013	2023-12-24 15:00:00.000000	2023-12-24 19:00:00.000000	holiday special
4	1014	2023-08-12 12:00:00.000000	2023-08-12 17:00:00.000000	private party
5	1015	2023-07-18 15:00:00.000000	2023-07-18 17:00:00.000000	conference
9	1016	2023-06-18 16:00:00.000000	2023-06-18 17:00:00.000000	promotion
10	1017	2023-06-24 12:00:00.000000	2023-06-24 14:00:00.000000	promotion
11	1017	2023-06-24 14:30:00.000000	2023-06-24 15:00:00.000000	promotion
10	1018	2023-06-24 12:00:00.000000	2023-06-24 14:00:00.000000	promotion
11	1018	2023-06-24 14:30:00.000000	2023-06-24 15:00:00.000000	promotion
11	1019	2023-06-24 14:30:00.000000	2023-06-24 15:00:00.000000	promotion

Underlined are promotion events in the database.
The yellow ones are tuples of promotional events on 2023-06-18. The blue ones are promotional events on 2023-06-24.

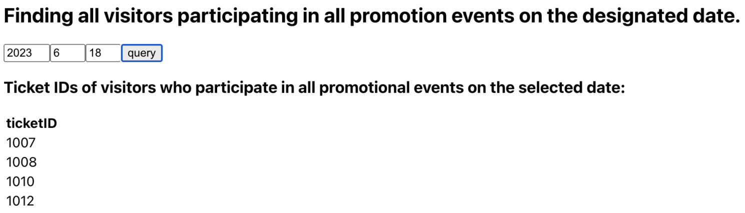
During:
By specifying the date and hitting the “query” button, the operation will be triggered.



Finding all visitors participating in all promotion events on the designated date.

Select the year, day and month above.

After:
After clicking the “query” button, the ticketIDs participating in all promotion events on the specified date are presented.



Finding all visitors participating in all promotion events on the designated date.

Ticket IDs of visitors who participate in all promotional events on the selected date:

ticketID
1007
1008
1010
1012