CPSC 304 Project Cover Page

Milestone #4

Date: August 11th, 2023

Group Number: 9

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Hailey Wu	72671456	c2v9l	haileyyiyu@gmail.com
Madeline Dow	75243949	a4d8w	mdow@student.ubc.ca
Sophia Zhou	55661094	r8a0q	realsophiazhou@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

A short description of the final project and what it accomplished:

Our final project was an amusement park system. It accomplishes the task of gathering critical information for the administrators of an amusement park to analyze the ongoings of the park in order to make adjustments according to presented numbers and data, and managing some parts of the system. For example, the project is able to add new entries to the list of all restaurants, viewing the total list with drinks associated, if the amusement park decides to open up new restaurants, and delete restaurants as well. Also, a user can check the amusement park's restaurant list to see which ones fall under a specific capacity range if there will be certain events that require a select number of seats.

How our final schema differed from the schema we turned in:

Our final schema differed from the schema initially submitted in that it only has a GUI for the administrators in the amusement park, and not also the visitors as we proposed. When we discussed our intended purposes of the schema, we realized that our queries and ideas would be more suited for an administrative-use GUI, and a visitor GUI would not serve any meaningful purpose or have any meaningful queries.

All SQL queries used:

```
CREATE TABLE Groups (GroupName char(50) PRIMARY KEY);
INSERT INTO Groups VALUES ('Happy Group');
INSERT INTO Groups VALUES ('Avengers');
INSERT INTO Groups VALUES ('Transformers');
INSERT INTO Groups VALUES ('Guardians');
INSERT INTO Groups VALUES ('Minions');

CREATE TABLE PartOfActor(StageName char(50) PRIMARY KEY, GroupName
char(50) NOT NULL, FOREIGN KEY (GroupName)
REFERENCES Groups);
INSERT INTO PartOfActor VALUES ('Bob', 'Happy Group');
INSERT INTO PartOfActor VALUES ('Sara', 'Happy Group');
INSERT INTO PartOfActor VALUES ('Steve Rogers', 'Avengers');
INSERT INTO PartOfActor VALUES ('Loki', 'Avengers');
INSERT INTO PartOfActor VALUES ('Rocket', 'Guardians');
INSERT INTO PartOfActor VALUES ('Groot', 'Guardians');
INSERT INTO PartOfActor VALUES ('Kevin', 'Minions');
INSERT INTO PartOfActor VALUES ('Stuart', 'Minions');

CREATE TABLE Performs_Show_R1 (Title char(50) PRIMARY KEY, Genre
char(50));
CREATE TABLE Performs_Show_R2 (StartTime integer, Seats integer, Title
char(50), GroupName char(50), PRIMARY KEY (StartTime, Title), FOREIGN KEY
(GroupName) REFERENCES Groups);

INSERT INTO Performs_Show_R1 VALUES('The Happy Show', 'Action');
INSERT INTO Performs_Show_R1 VALUES('The Happy Show Continued', 'Comedy');
INSERT INTO Performs_Show_R1 VALUES('Lokis Adventures', 'Action');
INSERT INTO Performs_Show_R1 VALUES('Groot Growing Up', 'Action');
INSERT INTO Performs_Show_R1 VALUES('Steves Sorrows', 'Tragedy');
INSERT INTO Performs_Show_R1 VALUES('Minions 1', 'Comedy');
INSERT INTO Performs_Show_R1 VALUES('Minions 2', 'Comedy');

INSERT INTO Performs_Show_R2 VALUES(0800, 200, 'Lokis Adventures',
'Avengers');
INSERT INTO Performs_Show_R2 VALUES(0900, 200, 'Lokis Adventures',
'Avengers');
INSERT INTO Performs_Show_R2 VALUES(1100, 100, 'Steves Sorrows',
'Avengers');
INSERT INTO Performs_Show_R2 VALUES(1200, 100, 'The Happy Show', 'Happy
Group');
INSERT INTO Performs_Show_R2 VALUES(1400, 100, 'The Happy Show Continued',
'Happy Group');
INSERT INTO Performs_Show_R2 VALUES(1300, 200, 'Groot Growing Up',
'Guardians');
INSERT INTO Performs_Show_R2 VALUES (0900, 100, 'Minions 1', 'Minions');
INSERT INTO Performs_Show_R2 VALUES (1100, 100, 'Minions 2', 'Minions');

CREATE TABLE Visitor (TicketNumber Integer PRIMARY KEY, VisitorName
char(50));
INSERT INTO Visitor VALUES (10001, 'Bob Jones');
INSERT INTO Visitor VALUES (10002, 'Maria Jones');
INSERT INTO Visitor VALUES (10003, 'Bob Jr Jones');
INSERT INTO Visitor VALUES (10004, 'Mary Sue');
INSERT INTO Visitor VALUES (10005, 'Ken Sue');
INSERT INTO Visitor VALUES (10006, 'Barbie Alex');
INSERT INTO Visitor VALUES (10007, 'Jenna Sue');
INSERT INTO Visitor VALUES (10008, 'Hailey Wu');
INSERT INTO Visitor VALUES (10009, 'Madeline Dow');
INSERT INTO Visitor VALUES (10010, 'Sophia Zhou');

CREATE TABLE Watches(StartTime Integer, Title char(50), TicketNumber
Integer,
FOREIGN KEY (StartTime, Title) REFERENCES Performs_Show_R2, FOREIGN KEY
(TicketNumber) REFERENCES Visitor,
PRIMARY KEY (StartTime, Title, TicketNumber));
INSERT INTO Watches VALUES (1200, 'The Happy Show', 10001);
INSERT INTO Watches VALUES (1200, 'The Happy Show', 10002);
INSERT INTO Watches VALUES (0800, 'Lokis Adventures', 10001);
INSERT INTO Watches VALUES (1100, 'Steves Sorrows', 10004);
INSERT INTO Watches VALUES (1100, 'Steves Sorrows', 10005);
INSERT INTO Watches VALUES (1300, 'Groot Growing Up', 10002);

CREATE TABLE Child (Height INTEGER, TicketNumber INTEGER PRIMARY KEY,
FOREIGN KEY (TicketNumber) REFERENCES Visitor);
INSERT INTO Child VALUES (140, 10006);
INSERT INTO Child VALUES (130, 10007);
INSERT INTO Child VALUES (129, 10008);
INSERT INTO Child VALUES (131, 10009);
INSERT INTO Child VALUES (160, 10010);

CREATE TABLE Adult (Age INTEGER, TicketNumber INTEGER PRIMARY KEY, FOREIGN
KEY (TicketNumber) REFERENCES Visitor);
INSERT INTO Adult VALUES (40, 10001);
INSERT INTO Adult VALUES (35, 10002);
INSERT INTO Adult VALUES (18, 10003);
INSERT INTO Adult VALUES (30, 10004);
INSERT INTO Adult VALUES (55, 10005);

CREATE TABLE Staff(StaffID Integer PRIMARY KEY, StaffName char(50));
INSERT INTO Staff VALUES(1, 'Anna');
INSERT INTO Staff VALUES(2, 'Ben');
INSERT INTO Staff VALUES (3, 'Charlie');
```

```sql
INSERT INTO Staff VALUES(4, 'Drew');
INSERT INTO Staff VALUES(5, 'Elsa');

Create Table Operates_Ride_R1 (RideType char(50) PRIMARY KEY,
HeightRestriction integer);
Create Table Operates_Ride_R2 (RideName char(50) PRIMARY KEY, Capacity
integer, RideType char(50), StaffID integer, FOREIGN KEY (StaffID)
REFERENCES Staff);

INSERT INTO Operates_Ride_R1 VALUES('Roller Coaster', 130);
INSERT INTO Operates_Ride_R1 VALUES('Drop', 120);
INSERT INTO Operates_Ride_R1 VALUES('Wheel', 0);
INSERT INTO Operates_Ride_R1 VALUES('Carousel', 0);
INSERT INTO Operates_Ride_R1 VALUES('Cars', 100);

INSERT INTO Operates_Ride_R2 VALUES('Splash Mountain', 6, 'Roller
Coaster', 1);
INSERT INTO Operates_Ride_R2 VALUES('Tower of Terror', 24, 'Drop', 2);
INSERT INTO Operates_Ride_R2 VALUES('Ferris Wheel', 4, 'Wheel', 3);
INSERT INTO Operates_Ride_R2 VALUES('Happy Carousel', 30, 'Carousel', 4);
INSERT INTO Operates_Ride_R2 VALUES('Bumper Cars', 12, 'Cars', 4);

CREATE TABLE GoesOn(RideName char(50), TicketNumber Integer,
PRIMARY KEY (RideName, TicketNumber),
FOREIGN KEY (RideName) REFERENCES Operates_Ride_R2,
FOREIGN KEY (TicketNumber) REFERENCES Visitor);
INSERT INTO GoesOn VALUES ('Splash Mountain', 10001);
INSERT INTO GoesOn VALUES ('Splash Mountain', 10002);
INSERT INTO GoesOn VALUES ('Tower of Terror', 10001);
INSERT INTO GoesOn VALUES ('Happy Carousel', 10006);
INSERT INTO GoesOn VALUES ('Bumper Cars', 10005);
INSERT INTO GoesOn VALUES ('Ferris Wheel', 10001);
INSERT INTO GoesOn VALUES ('Happy Carousel', 10001);
INSERT INTO GoesOn VALUES ('Bumper Cars', 10001);


CREATE TABLE Restaurant(RestaurantName char(50) PRIMARY KEY, Capacity
Integer);
INSERT INTO Restaurant VALUES('Princess Tea Party', 50);
INSERT INTO Restaurant VALUES('Death Eater Bar', 20);
INSERT INTO Restaurant VALUES('Marios Buffet', 100);
INSERT INTO Restaurant VALUES('Asgardian Feast', 100);
INSERT INTO Restaurant VALUES('Bobs Burgers', 50);

CREATE TABLE Provides_AlcoholicDrink(RestaurantName char(50), DrinkName
char(50), Price Integer,
PRIMARY KEY (RestaurantName, DrinkName),
FOREIGN KEY (RestaurantName) REFERENCES Restaurant
ON DELETE CASCADE);
INSERT INTO Provides_AlcoholicDrink VALUES('Death Eater Bar', 'Avada
Kevodka', 6.99);
INSERT INTO Provides_AlcoholicDrink VALUES('Death Eater Bar', 'Thunder
Beer', 10.99);
INSERT INTO Provides_AlcoholicDrink VALUES('Asgardian Feast', 'Thunder
Beer', 5.99);
INSERT INTO Provides_AlcoholicDrink VALUES('Marios Buffet', 'Super
Margarita', 10.99);
INSERT INTO Provides_AlcoholicDrink VALUES('Death Eater Bar', 'Winegardian
Leviosa', 6.99);

CREATE TABLE DinesAt (TicketNumber INTEGER, RestaurantName char(50),
PRIMARY KEY (TicketNumber, RestaurantName),
FOREIGN KEY (TicketNumber) REFERENCES Visitor, FOREIGN KEY
(RestaurantName) REFERENCES Restaurant ON DELETE CASCADE);
INSERT INTO DinesAt VALUES (10001, 'Princess Tea Party');
INSERT INTO DinesAt VALUES (10001, 'Death Eater Bar');
INSERT INTO DinesAt VALUES (10002, 'Death Eater Bar');
INSERT INTO DinesAt VALUES (10003, 'Marios Buffet');
INSERT INTO DinesAt VALUES (10004, 'Asgardian Feast');

CREATE TABLE Purchases (TicketNumber Integer,
                        RestaurantName char(50),
                        DrinkName char(50),
                        PRIMARY KEY (TicketNumber, RestaurantName,
DrinkName),
                        FOREIGN KEY (TicketNumber) REFERENCES Adult,
                        FOREIGN KEY (RestaurantName, DrinkName) REFERENCES
Provides_AlcoholicDrink ON DELETE CASCADE);
```

```
INSERT INTO Purchases VALUES(10001, 'Death Eater Bar', 'Thunder Beer');
INSERT INTO Purchases VALUES(10004, 'Asgardian Feast', 'Thunder Beer');
INSERT INTO Purchases VALUES(10004, 'Death Eater Bar', 'Thunder Beer');
INSERT INTO Purchases VALUES(10003, 'Marios Buffet', 'Super Margarita');
INSERT INTO Purchases VALUES(10002, 'Death Eater Bar', 'Avada Kevodka');
```

```
-- Selection
-- args (c1, c2)
SELECT RestaurantName
FROM Restaurant
WHERE Capacity > c1 AND Capacity < c2

-- Aggregation with group by
SELECT RestaurantName, MIN(Price)
FROM Provides_AlcoholicDrink
GROUP BY RestaurantName;

-- Division
-- Find visitors who have gone to all rides
SELECT VisitorName
FROM Visitor V
WHERE NOT EXISTS ((SELECT R.RideName
                FROM Operates_Ride_R2 R)
                MINUS
                (SELECT S.RideName
                FROM GoesOn S
                WHERE S.TicketNumber = V.TicketNumber));


-- Insert
-- arg(rname, capacity)
INSERT INTO RESTAURANT
VALUES (rname, capacity);

-- Delete
-- arg(rname)
DELETE FROM RESTAURANT
```

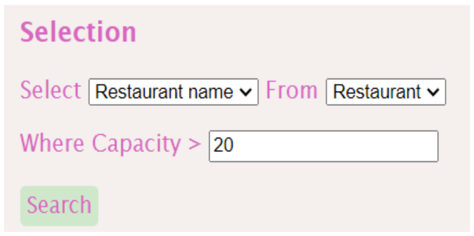
```
WHERE RESTAURANTNAME = rname;

-- Show the RESTAURANT and PROVIDES_ALCOHOLICDRINK Tables
SELECT *
FROM RESTAURANT;

SELECT Count(*)
FROM PROVIDES_ALCOHOLICDRINK;

SELECT *
FROM PROVIDES_ALCOHOLICDRINK;

-- Projection
-- arg(column1, column2, …)
SELECT column1, column2, …
FROM PERFORMS_SHOW_R2;

-- Having
-- arg(minShows)
SELECT GENRE, COUNT(*)
FROM PERFORMS_SHOW_R1 r1, PERFORMS_SHOW_R2 r2
WHERE r1.TITLE = r2.TITLE
GROUP BY GENRE
HAVING COUNT(*) >= minShows;

-- Display the Show Schedule
SELECT STARTTIME, r1.TITLE, GENRE, SEATS, GROUPNAME
FROM PERFORMS_SHOW_R1 r1, PERFORMS_SHOW_R2 r2
WHERE r1.TITLE = r2.TITLE
ORDER BY STARTTIME;



--Find the types of rides with capacity thats greater than the average
capacity of all the ride types


SELECT RideType
FROM Operates_Ride_R2 r2
```

```sql
GROUP BY RideType
HAVING avg(Capacity) > (SELECT avg(Capacity)
                        FROM Operates_Ride_R2);

-- Find the name of all visitors who have been on a ride (Join)
SELECT VisitorName
FROM Visitor v, GoesOn g
WHERE v.TicketNumber = g.TicketNumber;


-- Update RideName


UPDATE Operates_Ride_R2
SET RideName = 'Splasher'
WHERE RideName = 'Splash Mountain';
```

Screenshots of the sample output of the queries using the GUI:

(BEFORE) shows what data is in our table before running the query, (AFTER) shows another screenshot after running the query, with GUI input included.

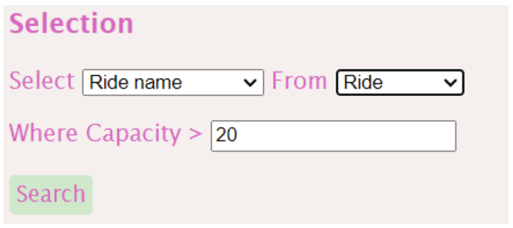
SELECTION: after clicking the search button, shows the names of the restaurants in the given range of capacity
(Before):



Selection

Select Restaurant name From Restaurant

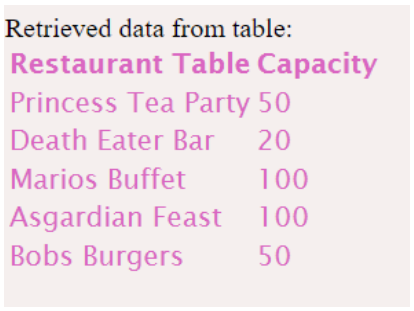
Where Capacity > 20



Selection

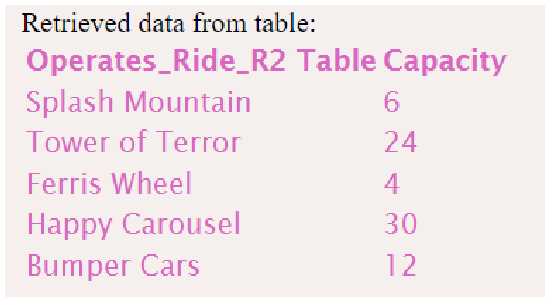
Select Ride name From Ride

Where Capacity > 20



Retrieved data from table:

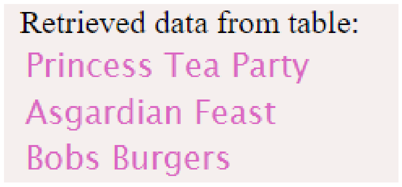
Restaurant	Table Capacity
Princess Tea Party	50
Death Eater Bar	20
Marios Buffet	100
Asgardian Feast	100
Bobs Burgers	50



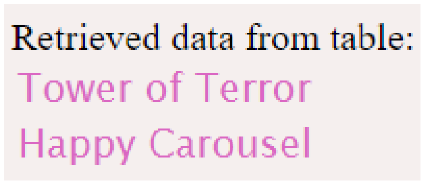
Retrieved data from table:

Operates_Ride_R2	Table Capacity
Splash Mountain	6
Tower of Terror	24
Ferris Wheel	4
Happy Carousel	30
Bumper Cars	12

(After):



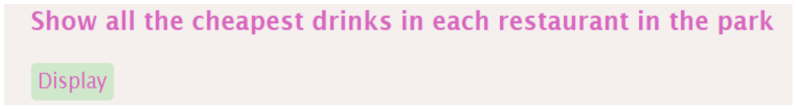
Retrieved data from table:
Princess Tea Party
Asgardian Feast
Bobs Burgers



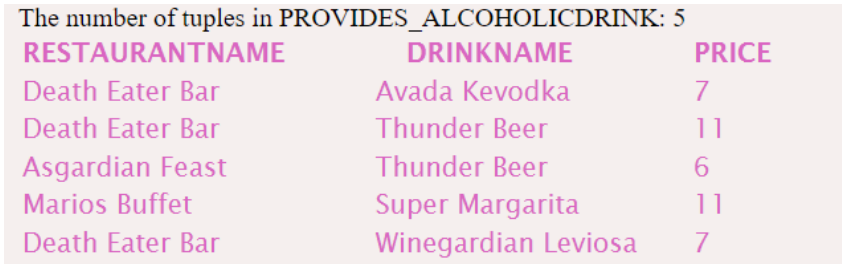
Retrieved data from table:
Tower of Terror
Happy Carousel

<u>AGGREGATION WITH GROUP BY</u>: after clicking the display button, shows the cheapest drink in each restaurant

(Before):



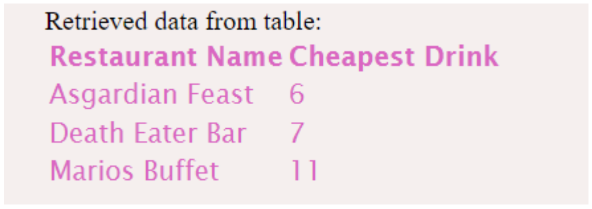
Show all the cheapest drinks in each restaurant in the park



The number of tuples in PROVIDES_ALCOHOLICDRINK: 5

RESTAURANTNAME	DRINKNAME	PRICE
Death Eater Bar	Avada Kevodka	7
Death Eater Bar	Thunder Beer	11
Asgardian Feast	Thunder Beer	6
Marios Buffet	Super Margarita	11
Death Eater Bar	Winegardian Leviosa	7

(After):

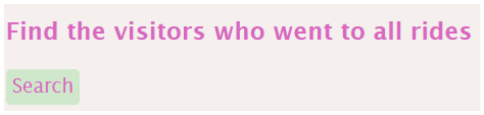


Retrieved data from table:

Restaurant Name	Cheapest Drink
Asgardian Feast	6
Death Eater Bar	7
Marios Buffet	11

<u>DIVISION</u>: after clicking the search button, shows the visitors who went to all rides

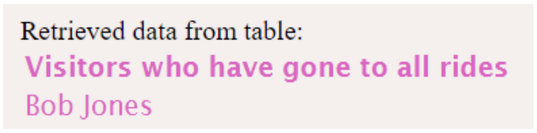
(Before):



Find the visitors who went to all rides

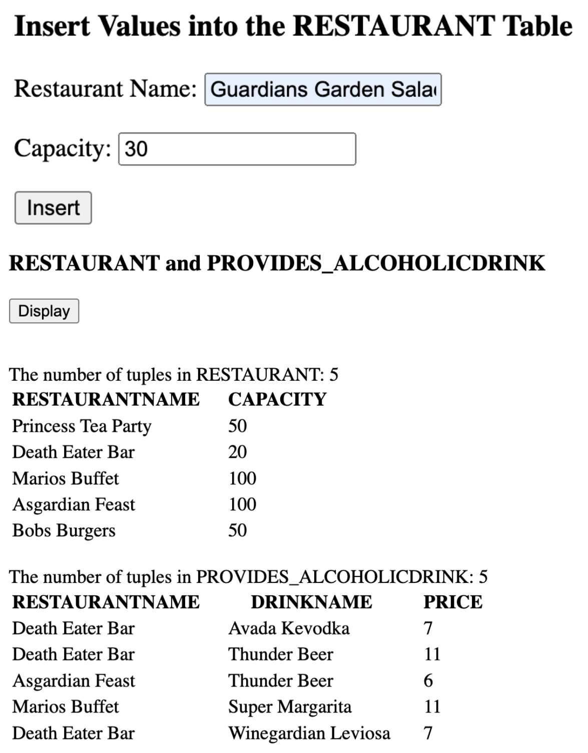
```
SQL> SELECT * FROM Visitor;

TICKETNUMBER VISITORNAME
------------ --------------------------------------------------
       10001 Bob Jones
       10002 Maria Jones
       10003 Bob Jr Jones
       10004 Mary Sue
       10005 Ken Sue
       10006 Barbie Alex
       10007 Jenna Sue
       10008 Hailey Wu
       10009 Madeline Dow
       10010 Sophia Zhou

SQL> SELECT * FROM GoesON;

RIDENAME                                           TICKETNUMBER
-------------------------------------------------- ------------
Bumper Cars                                               10001
Bumper Cars                                               10005
Ferris Wheel                                              10001
Happy Carousel                                            10001
Happy Carousel                                            10006
Splash Mountain                                           10001
Splash Mountain                                           10002
Tower of Terror                                           10001


RIDENAME                                             CAPACITY
-------------------------------------------------- ----------
RIDETYPE                                              STAFFID
-------------------------------------------------- ----------
Splash Mountain                                             6
Roller Coaster                                              1

Tower of Terror                                            24
Drop                                                        2

Ferris Wheel                                                4
Wheel                                                       3

RIDENAME                                             CAPACITY
-------------------------------------------------- ----------
RIDETYPE                                              STAFFID
-------------------------------------------------- ----------
Happy Carousel                                             30
Carousel                                                    4

Bumper Cars                                                12
Cars                                                        4
```

(After):


Retrieved data from table:
Visitors who have gone to all rides
Bob Jones

INSERT: Takes values for a new restaurant name and capacity, and insert a new restaurant

(BEFORE):



Insert Values into the RESTAURANT Table

Restaurant Name: Guardians Garden Sala

Capacity: 30

RESTAURANT and PROVIDES_ALCOHOLICDRINK

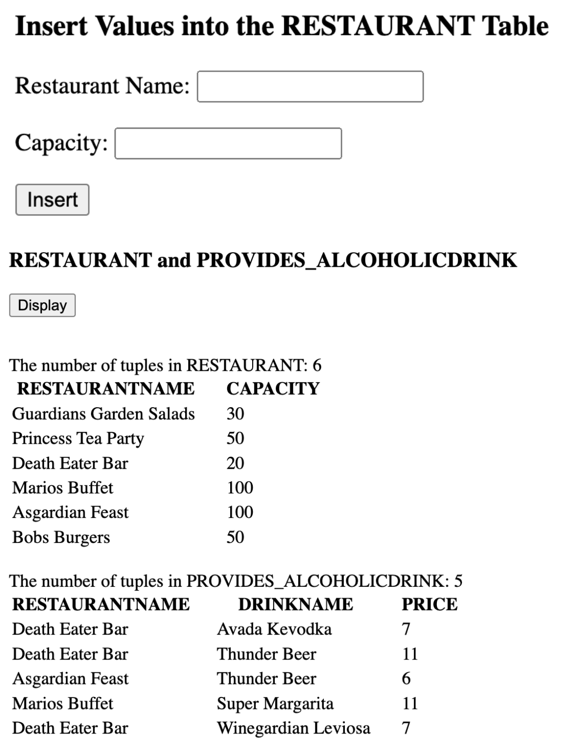
The number of tuples in RESTAURANT: 5

RESTAURANTNAME	CAPACITY
Princess Tea Party	50
Death Eater Bar	20
Marios Buffet	100
Asgardian Feast	100
Bobs Burgers	50

The number of tuples in PROVIDES_ALCOHOLICDRINK: 5

RESTAURANTNAME	DRINKNAME	PRICE
Death Eater Bar	Avada Kevodka	7
Death Eater Bar	Thunder Beer	11
Asgardian Feast	Thunder Beer	6
Marios Buffet	Super Margarita	11
Death Eater Bar	Winegardian Leviosa	7

(AFTER):



Insert Values into the RESTAURANT Table

Restaurant Name:

Capacity:

RESTAURANT and PROVIDES_ALCOHOLICDRINK

The number of tuples in RESTAURANT: 6

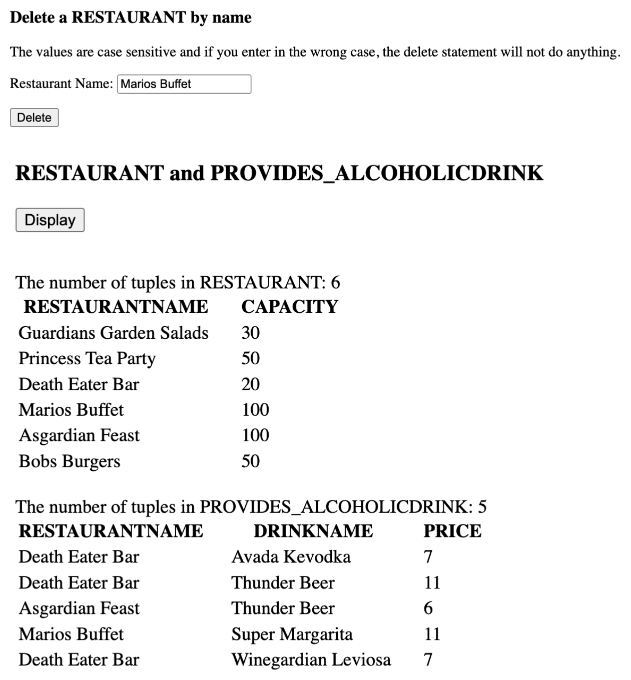
RESTAURANTNAME	CAPACITY
Guardians Garden Salads	30
Princess Tea Party	50
Death Eater Bar	20
Marios Buffet	100
Asgardian Feast	100
Bobs Burgers	50

The number of tuples in PROVIDES_ALCOHOLICDRINK: 5

RESTAURANTNAME	DRINKNAME	PRICE
Death Eater Bar	Avada Kevodka	7
Death Eater Bar	Thunder Beer	11
Asgardian Feast	Thunder Beer	6
Marios Buffet	Super Margarita	11
Death Eater Bar	Winegardian Leviosa	7

DELETE: Takes the name for the restaurant to be deleted, and then deletes (with on-cascade function)

(BEFORE):



Delete a RESTAURANT by name

The values are case sensitive and if you enter in the wrong case, the delete statement will not do anything.

Restaurant Name: Marios Buffet

RESTAURANT and PROVIDES_ALCOHOLICDRINK

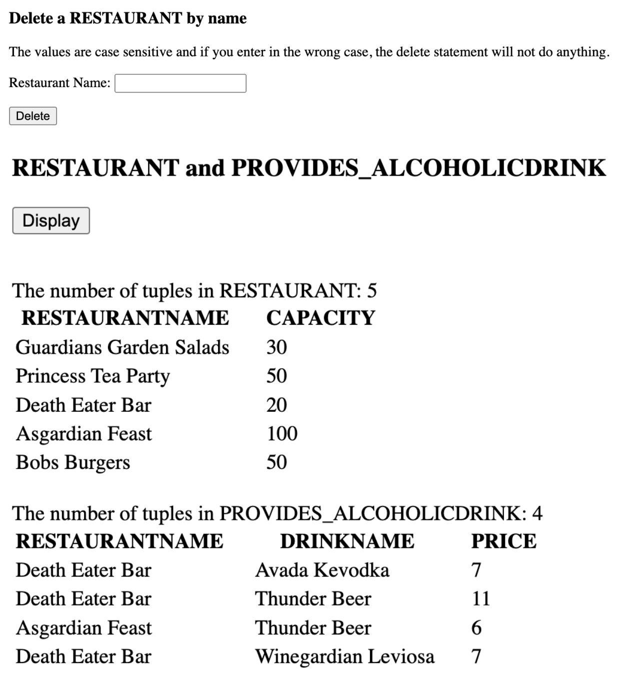
The number of tuples in RESTAURANT: 6

RESTAURANTNAME	CAPACITY
Guardians Garden Salads	30
Princess Tea Party	50
Death Eater Bar	20
Marios Buffet	100
Asgardian Feast	100
Bobs Burgers	50

The number of tuples in PROVIDES_ALCOHOLICDRINK: 5

RESTAURANTNAME	DRINKNAME	PRICE
Death Eater Bar	Avada Kevodka	7
Death Eater Bar	Thunder Beer	11
Asgardian Feast	Thunder Beer	6
Marios Buffet	Super Margarita	11
Death Eater Bar	Winegardian Leviosa	7

(AFTER):



Delete a RESTAURANT by name

The values are case sensitive and if you enter in the wrong case, the delete statement will not do anything.

Restaurant Name:

RESTAURANT and PROVIDES_ALCOHOLICDRINK

The number of tuples in RESTAURANT: 5

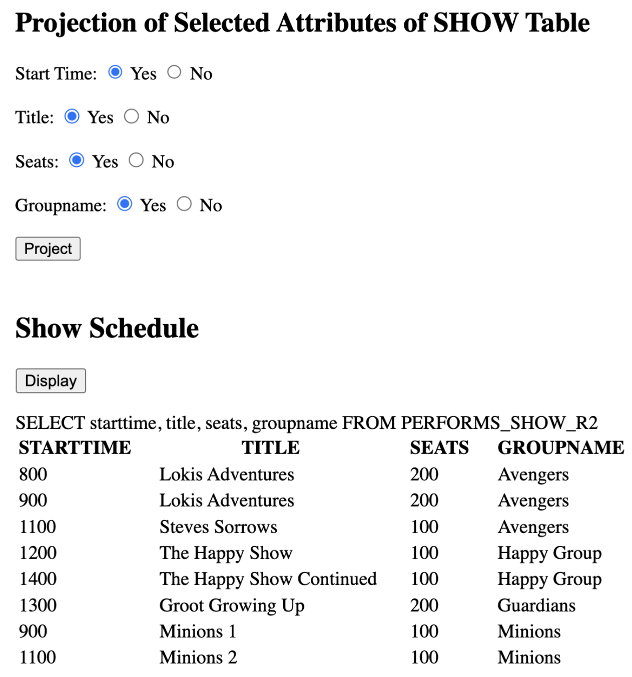
RESTAURANTNAME	CAPACITY
Guardians Garden Salads	30
Princess Tea Party	50
Death Eater Bar	20
Asgardian Feast	100
Bobs Burgers	50

The number of tuples in PROVIDES_ALCOHOLICDRINK: 4

RESTAURANTNAME	DRINKNAME	PRICE
Death Eater Bar	Avada Kevodka	7
Death Eater Bar	Thunder Beer	11
Asgardian Feast	Thunder Beer	6
Death Eater Bar	Winegardian Leviosa	7

PROJECTION: Using radio buttons, the user selects the desired attributes from Show and upon clicking Project, Show Schedule will be displayed with only selected attributes

(BEFORE):



Projection of Selected Attributes of SHOW Table

Start Time: [✓ Yes] No

Title: [✓ Yes] No

Seats: [✓ Yes] No

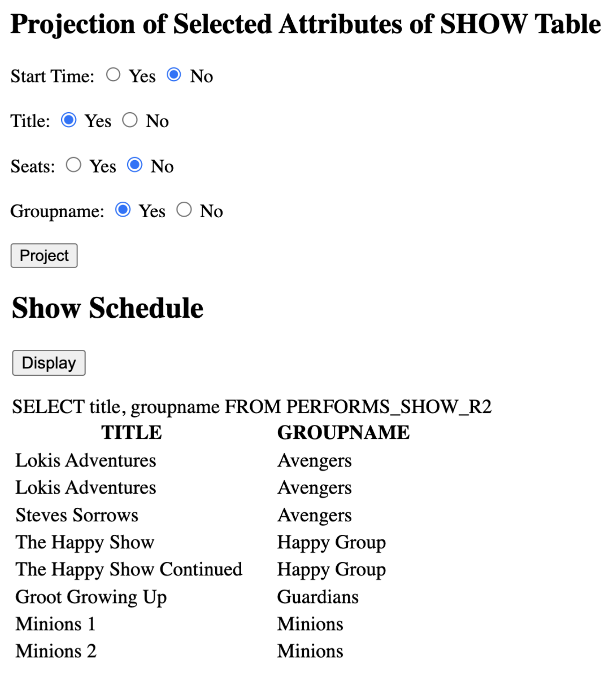
Groupname: [✓ Yes] No

Show Schedule

```
SELECT starttime, title, seats, groupname FROM PERFORMS_SHOW_R2
```

STARTTIME	TITLE	SEATS	GROUPNAME
800	Lokis Adventures	200	Avengers
900	Lokis Adventures	200	Avengers
1100	Steves Sorrows	100	Avengers
1200	The Happy Show	100	Happy Group
1400	The Happy Show Continued	100	Happy Group
1300	Groot Growing Up	200	Guardians
900	Minions 1	100	Minions
1100	Minions 2	100	Minions

(AFTER):



Projection of Selected Attributes of SHOW Table

Start Time: Yes [✓ No]

Title: [✓ Yes] No

Seats: Yes [✓ No]

Groupname: [✓ Yes] No

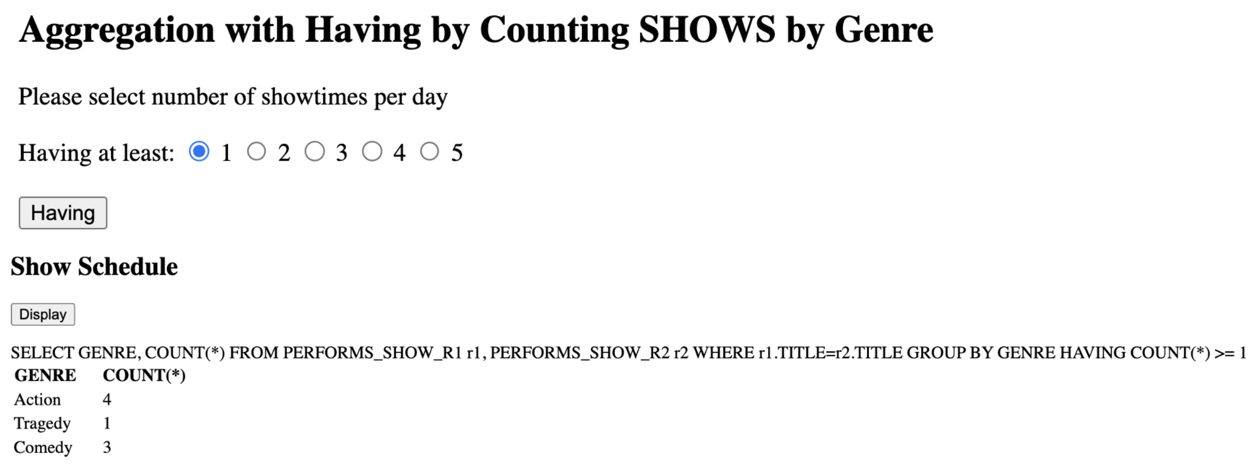
Show Schedule

```
SELECT title, groupname FROM PERFORMS_SHOW_R2
```

TITLE	GROUPNAME
Lokis Adventures	Avengers
Lokis Adventures	Avengers
Steves Sorrows	Avengers
The Happy Show	Happy Group
The Happy Show Continued	Happy Group
Groot Growing Up	Guardians
Minions 1	Minions
Minions 2	Minions

Aggregation with Having: Displaying the genres of Shows that have at least [select] showtimes

(BEFORE):



Aggregation with Having by Counting SHOWS by Genre

Please select number of showtimes per day

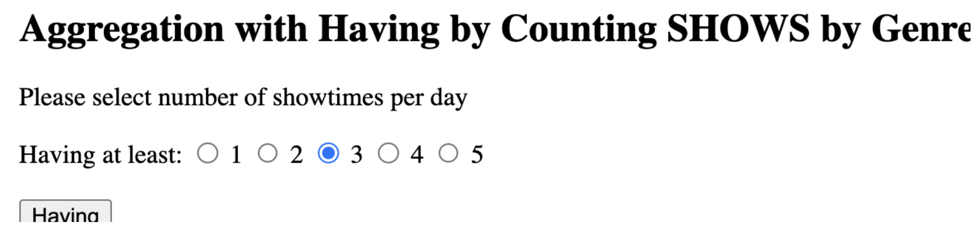
Having at least: [✓ 1] 2 3 4 5

Show Schedule

```
SELECT GENRE, COUNT(*) FROM PERFORMS_SHOW_R1 r1, PERFORMS_SHOW_R2 r2 WHERE r1.TITLE=r2.TITLE GROUP BY GENRE HAVING COUNT(*) >= 1
```

GENRE	COUNT(*)
Action	4
Tragedy	1
Comedy	3

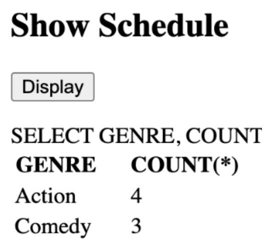
(AFTER):



Aggregation with Having by Counting SHOWS by Genre

Please select number of showtimes per day

Having at least: 1 2 [✓ 3] 4 5

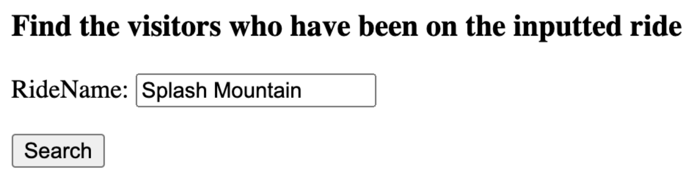


Show Schedule

```
SELECT GENRE, COUNT
```

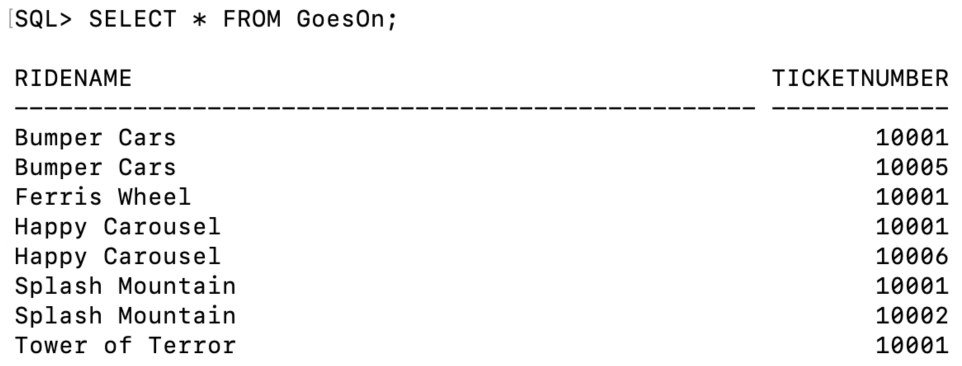
GENRE	COUNT(*)
Action	4
Comedy	3

<u>JOIN</u>: Uses user input to find the visitors who have been on the inputted ride by joining the Visitor and GoesOn tables
(Before):



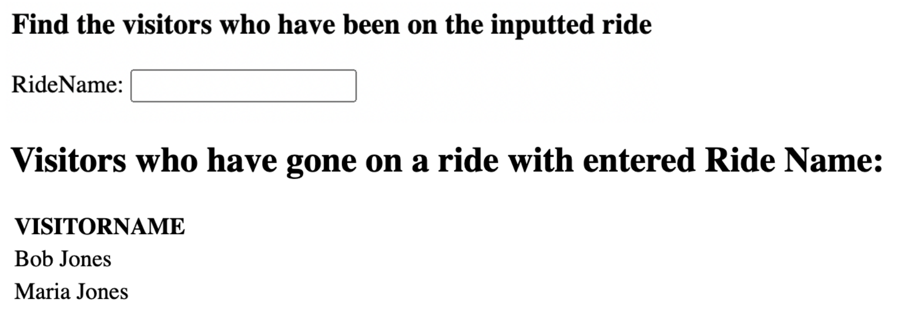
Find the visitors who have been on the inputted ride

RideName: Splash Mountain



```
SQL> SELECT * FROM GoesOn;

RIDENAME                                            TICKETNUMBER
--------------------------------------------------- ------------
Bumper Cars                                                10001
Bumper Cars                                                10005
Ferris Wheel                                               10001
Happy Carousel                                             10001
Happy Carousel                                             10006
Splash Mountain                                            10001
Splash Mountain                                            10002
Tower of Terror                                            10001
```

(After):



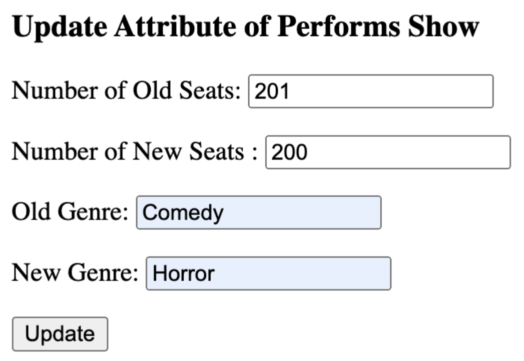
Find the visitors who have been on the inputted ride

RideName:

Visitors who have gone on a ride with entered Ride Name:

VISITORNAME
Bob Jones
Maria Jones

<u>UPDAT:</u> Update attributes of the Performs Show table by inputting number of old seats + number of new seats and/or old genre + new genre
(Before):



Update Attribute of Performs Show

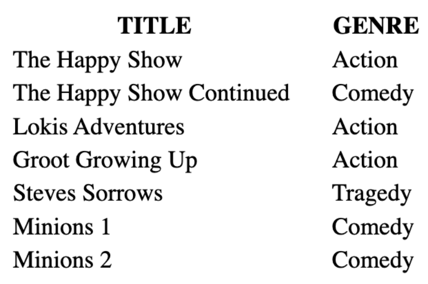
Number of Old Seats: 201

Number of New Seats : 200

Old Genre: Comedy

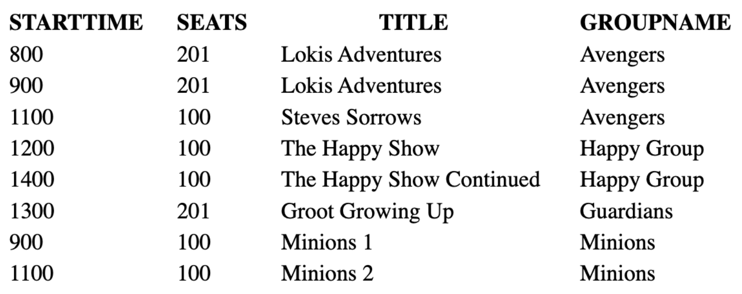
New Genre: Horror

Performs_Show_R1 table



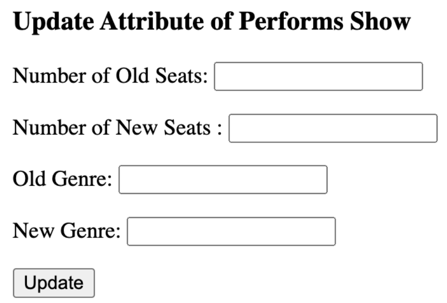
TITLE	GENRE
The Happy Show	Action
The Happy Show Continued	Comedy
Lokis Adventures	Action
Groot Growing Up	Action
Steves Sorrows	Tragedy
Minions 1	Comedy
Minions 2	Comedy

Performs_Show_R2 table



STARTTIME	SEATS	TITLE	GROUPNAME
800	201	Lokis Adventures	Avengers
900	201	Lokis Adventures	Avengers
1100	100	Steves Sorrows	Avengers
1200	100	The Happy Show	Happy Group
1400	100	The Happy Show Continued	Happy Group
1300	201	Groot Growing Up	Guardians
900	100	Minions 1	Minions
1100	100	Minions 2	Minions

(After):



Update Attribute of Performs Show

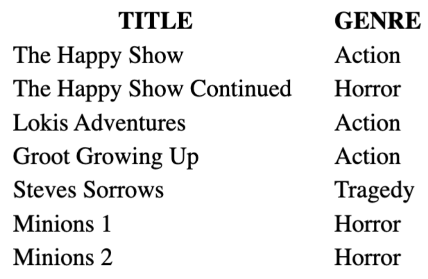
Number of Old Seats:

Number of New Seats :

Old Genre:

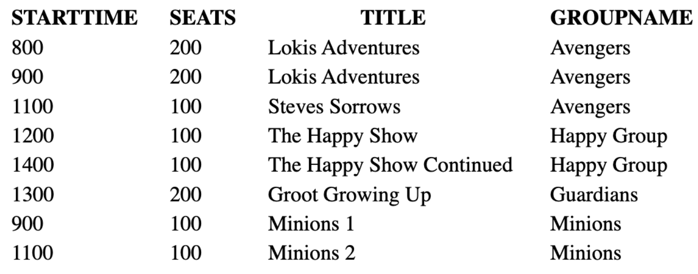
New Genre:

Performs_Show_R1 table



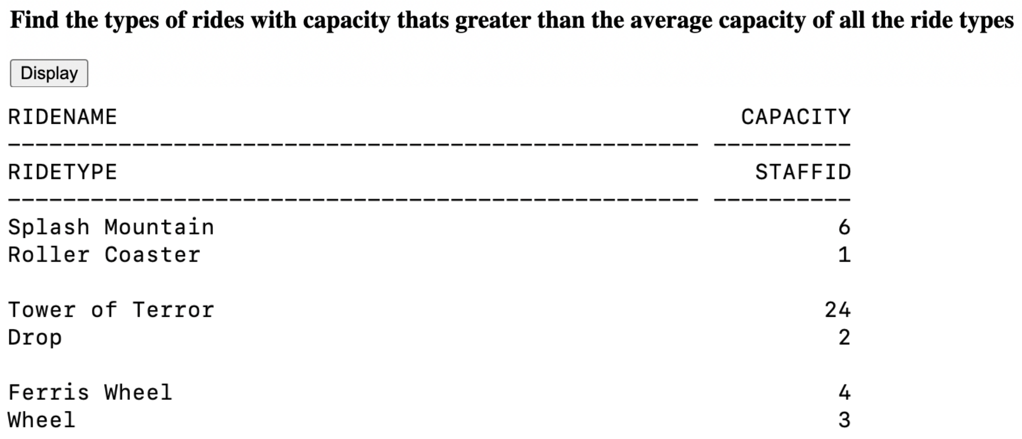
TITLE	GENRE
The Happy Show	Action
The Happy Show Continued	Horror
Lokis Adventures	Action
Groot Growing Up	Action
Steves Sorrows	Tragedy
Minions 1	Horror
Minions 2	Horror

Performs_Show_R2 table



STARTTIME	SEATS	TITLE	GROUPNAME
800	200	Lokis Adventures	Avengers
900	200	Lokis Adventures	Avengers
1100	100	Steves Sorrows	Avengers
1200	100	The Happy Show	Happy Group
1400	100	The Happy Show Continued	Happy Group
1300	200	Groot Growing Up	Guardians
900	100	Minions 1	Minions
1100	100	Minions 2	Minions

Nested Aggregation with Group By: Find the types of rides that have a capacity that's greater than the average capacity of all the ride types
(Before):



Find the types of rides with capacity thats greater than the average capacity of all the ride types

```
RIDENAME                                              CAPACITY
--------------------------------------------------- ----------
RIDETYPE                                               STAFFID
--------------------------------------------------- ----------
Splash Mountain                                              6
Roller Coaster                                               1

Tower of Terror                                             24
Drop                                                         2

Ferris Wheel                                                 4
Wheel                                                        3
```

```
RIDENAME                                                   CAPACITY
-------------------------------------------------- ----------
RIDETYPE                                                    STAFFID
-------------------------------------------------- ----------
Happy Carousel                                                   30
Carousel                                                          4

Bumper Cars                                                      12
Cars                                                              4
```

(After):

```
RIDETYPE
Drop
Carousel
```