University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 4

Date: November 25th, 2022

Group Number: 76

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Jason Nguyen	11170081	h1a3b	nguyen.j1305@gmail.com
Andy Lee	13349634	h2j4n	andy2002221@gmail.com
West Liu	40081531	q9i4u	westliu20200101@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

Repository Link:
https://github.students.cs.ubc.ca/CPSC304-2022W-T1/project_h1a3b_h2j4n_q9i4u

SQL Script:

- Included in github under the file name: cruiseship.sql

Project Description:

- Our final project is a database for cruise ships. It holds data for various elements of a cruise ship including ticket information, passenger information, amenities, and crew member information. Our database keeps track of useful information such as which passenger utilized which amenity and which crew member was responsible for maintaining which amenity for each ship. This allows the database administrator to extract key business information such as the average captain salary for each cruise ship, or screen crew members to identify inefficient or slow workers (return crew members who have maintained at least x amount of rooms). As a passenger/public ticket holder, one can also look up information about the cruise by entering just the ticketID to discover their sailing details and available amenities onboard.

Changes to Schema:

- No functional changes were made to our post-normalization schema.
- The minor changes are simple attribute renaming, due to php's reserved keywords such as "name", "start", "end", "from", etc. We simply added a prefix to each of these attributes. (for example, name for passengers renamed to passengerName)
- Due to Oracle not supporting ON UPDATE CASCADE, we replaced these with ON UPDATE NO ACTION.

Final Schema:

Ticket(ticketID : number, ticketClass : string, ticketDate : string, **hullID** : number, **passengerID**: number)

PassengerLocation(postalCode : string, city : string)

Passengers(passengerID : number, passengerName: string, age: number, **postalCode** : string, passengerAddress : string)

Pets(**passengerID**: number, petName: string, breed: string)

CruiseShip(hullID: number, cruiseName: string, fromLocation: string, toLocation: string)

Hospitality(roomNo : string, maxCapacity: number, roomType: string, **hullID** : number)

PassengersStayAt(**passengerID** : number, **roomNo**: number)

Activities(stall : string, actStart : number, actEnd : number, activityName: string, **hullID** : number)

PassengersParticipateIn(**stall** : string, **passengerID** : number)

Restaurants(stall : string, restName : string, restStart : number, restEnd : number, **hullID** : number)

PassengersEatAt(**passengerID** : number, **stall** : string)

Captain(crewID: number, captainName : string, salary: number, licenseNum: number)

Pilots(**crewID** : number, **hullID** : number)

GeneralStaffSalary(staffRole : string, salary : number)

GeneralStaff(crewID : number, staffName : string, **staffRole** : string)

ManageHospitalities(**crewID** : number, **roomNum** : number)

ManageActivities(**crewID** : number, **stall** : string)

ManageRestaurants(**crewID** : number, **stall** : string)

Screenshots to show what data is present in each relation after SQL script is

Ticket

TICKETID	TICKETCLASS	TICKETDATE	HULLID	PASSENGERID
1	luxury	6/23/22	1	1
2	economy	6/23/22	1	2
3	suite	9/20/22	2	3
4	suite	9/20/22	2	4
5	economy	9/25/22	3	5
6	luxury	10/15/22	4	6
7	suite	10/21/22	5	7
8	suite	10/21/22	5	8
15	luxury	6/30/22	1	9
20	economy	6/30/22	1	10

PassengerLocation

POSTALCODE	CITY
v3t	surrey
v1m	surrey
v3j	burnaby
v3n	burnaby
v5z	vancouver

Passengers

PASSENGERID	PASSENGERNAME	AGE	POSTALCODE	PASSENGERADDRESS
1	Jason Smith	21	v3t	31232 140 Street
2	John Legend	35	v1m	87732 128 Street
3	Kim Jane	28	v3j	68232 64 Avenue
4	Daniel Jane	29	v3j	68232 64 Avenue
5	Alex Green	33	v5z	24521 120A Street
6	Adrian Chun	60	v3n	43262 80 Avenue
7	Ashley Campbell	31	v5z	14461 96 Street
8	David Campbell	32	v5z	23232 100 Street
9	Snoopy Binoo	31	v3t	23022 156 Street
10	Joshua Asfa	60	v3n	76493 94 Avenue

GeneralStaff

CREWID	STAFFNAME	STAFFROLE
10	Jason	Housekeeping
11	Andy	Waiter
12	West	Cook
13	Janel	Activity Manager
14	Ariana	Housekeeping
15	Kawhi	Janitor
16	Jinmin	Housekeeping
17	Elon	Activity Manager
29	Johnathon	Housekeeping
30	Rachel	Housekeeping
35	Jasmine	Housekeeping
39	Mary	Housekeeping

Pets

PASSENGERID	PETNAME	BREED
1	Bogey	Golden Retriever
3	Armpit	Pitbull
5	Steve	Siamese Cat
6	Bummy	Burmese Cat
7	Hammy	Syrian Hamster

CruiseShip

HULLID	CRUISENAME	FROMLOCATION	TOLOCATION
1	Princess Cruises	Vancouver, BC	Vancouver Island, BC
2	Disney Cruise Line	Tsawwassen, BC	Tsawwassen, BC
3	Viking Cruises	Vancouver, BC	Vancouver, BC
4	Crystal Cruises	Vancouver, BC	Bowen Island, BC
5	Klondike Cruises	Nanaimo, BC	Vancouver, BC

Hospitality

ROOMNO	MAXCAPACITY	ROOMTYPE	HULLID
PC1-01	4	double kings	1
PC1-02	2	double twin	1
PC1-03	4	double kings	1
PC1-04	2	double twin	1
DCL2-01	2	single queen	2
VC3-01	1	single twin	3
CC4-01	4	double kings	4
KC5-01	2	double queen	5

ManageActivities

CREWID	STALL
13	PC1-DANCE1
15	DCL2-DRAW1
15	VC3-SWIM1
17	KC5-MOVIE1
17	PC1-DANCE2

PassengersStayAt

PASSENGERID	ROOMNO
1	PC1-01
2	PC1-02
3	DCL2-01
4	DCL2-01
5	VC3-01
6	CC4-01
7	KC5-01
8	KC5-01
9	PC1-03
10	PC1-04

Activities

STALL	ACTSTART	ACTEND	ACTIVITYNAME	HULLID
PC1-DANCE1	1200	1500	dancing	1
PC1-DANCE2	1600	1800	dancing	1
DCL2-DRAW1	1400	1900	painting	2
DCL2-SWIM1	1200	2000	swimming	2
VC3-SWIM1	1100	2100	swimming	3
CC4-MNGLF1	1400	2300	mini golf	4
KC5-MOVIE1	1800	2000	movies	5

PassengersEatAt

PASSENGERID	STALL
1	PC1-TACO
3	DCL2-MCD
4	DCL2-MCD
7	KC5-KEG
8	KC5-KEG

PassengersParticipateIn

STALL	PASSENGERID
CC4-MNGLF1	3
CC4-MNGLF1	5
DCL2-DRAW1	3
DCL2-DRAW1	5
DCL2-SWIM1	3
DCL2-SWIM1	4
DCL2-SWIM1	5
KC5-MOVIE1	3
KC5-MOVIE1	5
KC5-MOVIE1	7
KC5-MOVIE1	8
PC1-DANCE1	1
PC1-DANCE1	3
PC1-DANCE1	5
PC1-DANCE2	3
PC1-DANCE2	5
VC3-SWIM1	3
VC3-SWIM1	5

GeneralStaffSalary

STAFFROLE	SALARY
Housekeeping	50000
Waiter	55000
Cook	70000
Activity Manager	52000
Janitor	60000

ManageHospitalities

CREWID	ROOMNUM
10	KC5-01
10	PC1-01
10	PC1-02
11	DCL2-01
11	VC3-01
12	CC4-01
30	PC1-03
35	PC1-04

ManageRestaurants

CREWID	STALL
11	DCL2-MCD
11	KC5-KEG
12	CC4-POKE
12	PC1-TACO
12	VC3-EARLS

Restaurants

STALL	RESTNAME	RESTSTART	RESTEND	HULLID
DCL2-MCD	McDonalds	0	2400	2
KC5-KEG	The Keg	1600	2400	5
PC1-TACO	Tacofina	800	2200	1
CC4-POKE	Pokaye	1000	2100	4
VC3-EARLS	Earls	1200	2400	3

Captain

CREWID	CAPTAINNAME	SALARY	LICENSENUM
3	Masa	120000	1278
4	Brian	160000	2834
5	Alex	170000	7263
6	Mike	200000	4028
7	Chris	110000	3948
18	Bob Marley	230000	1278
19	Tupac	90000	2834
20	Snoop Dogg	163000	7263
21	50 Cent	125000	4028
25	Jay Z	103000	3948

Pilots

CREWID	HULLID
3	1
4	2
5	3
6	4
7	5
18	1
19	2
20	3
21	4
25	5

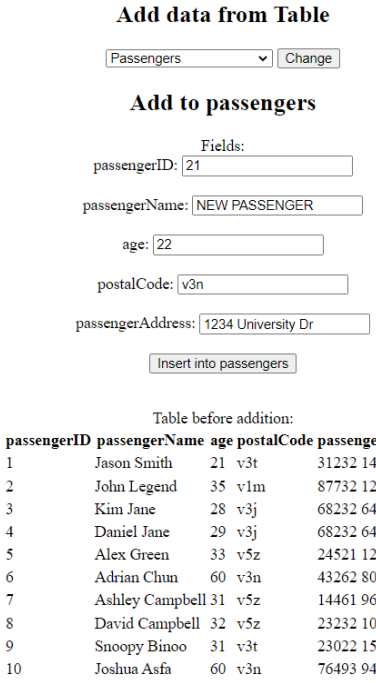
Queries:

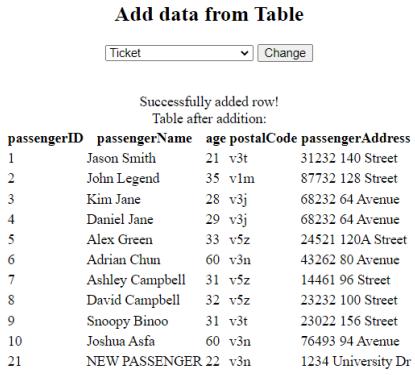
1. INSERT

a. INSERT INTO table_name

VALUES (input1, input2, ….)

b. Implementation can be found in “add-data.php” on github





2. DELETE

a. DELETE FROM table_name

WHERE table_prim_key(s) = user_input(s)

b. Implementation can be found in “remove_data_page.php” on github

Remove data from Table

Ticket Change

Deleting from passengers

CONDITIONS:

passengerID: 21

passengerName:

age:

postalCode:

passengerAddress:

Remove from passengers

Table before deletion:

passengerID	passengerName	age	postalCode	passengerAddress
1	Jason Smith	21	v3t	31232 140 Street
2	John Legend	35	v1m	87732 128 Street
3	Kim Jane	28	v3j	68232 64 Avenue
4	Daniel Jane	29	v3j	68232 64 Avenue
5	Alex Green	33	v5z	24521 120A Street
6	Adrian Chun	60	v3n	43262 80 Avenue
7	Ashley Campbell	31	v5z	14461 96 Street
8	David Campbell	32	v5z	23232 100 Street
9	Snoopy Binoo	31	v3t	23022 156 Street
10	Joshua Asfa	60	v3n	76493 94 Avenue
21	NEW PASSENGER	22	v3n	1234 University Dr

Remove data from Table

Ticket Change

Successfully deleted row(s)!

Table after deletion:

passengerID	passengerName	age	postalCode	passengerAddress
1	Jason Smith	21	v3t	31232 140 Street
2	John Legend	35	v1m	87732 128 Street
3	Kim Jane	28	v3j	68232 64 Avenue
4	Daniel Jane	29	v3j	68232 64 Avenue
5	Alex Green	33	v5z	24521 120A Street
6	Adrian Chun	60	v3n	43262 80 Avenue
7	Ashley Campbell	31	v5z	14461 96 Street
8	David Campbell	32	v5z	23232 100 Street
9	Snoopy Binoo	31	v3t	23022 156 Street
10	Joshua Asfa	60	v3n	76493 94 Avenue

3. UPDATE

a. UPDATE table_name

SET table_att1 = user_input1, table_att2 = user_input2,...

WHERE user_selected_att1 = user_cond1…

b. Implementation can be found in “update_data.php” on github

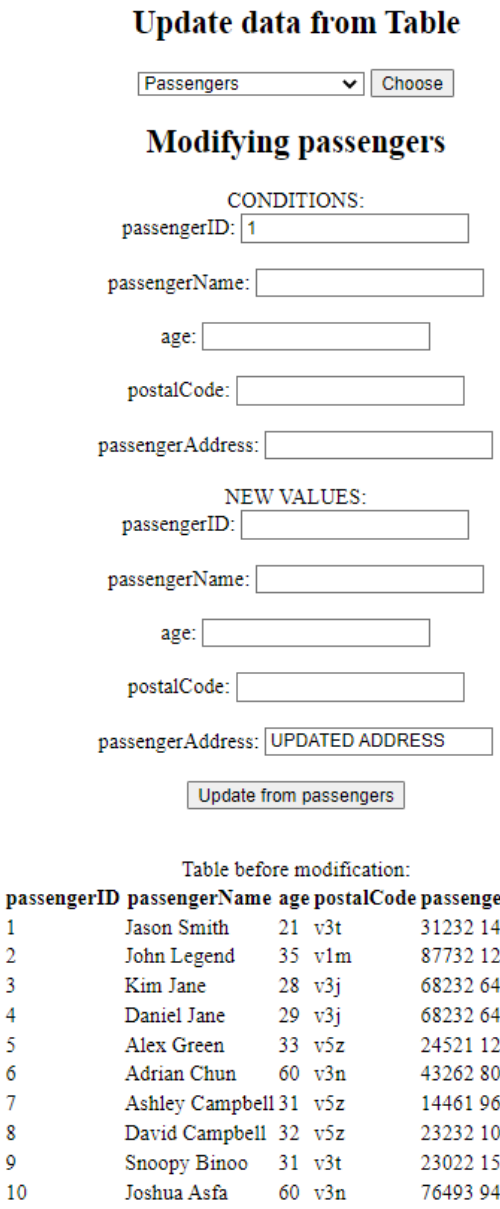
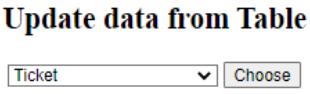


Table before modification:

passengerID	passengerName	age	postalCode	passengerAddress
1	Jason Smith	21	v3t	31232 140 Street
2	John Legend	35	v1m	87732 128 Street
3	Kim Jane	28	v3j	68232 64 Avenue
4	Daniel Jane	29	v3j	68232 64 Avenue
5	Alex Green	33	v5z	24521 120A Street
6	Adrian Chun	60	v3n	43262 80 Avenue
7	Ashley Campbell	31	v5z	14461 96 Street
8	David Campbell	32	v5z	23232 100 Street
9	Snoopy Binoo	31	v3t	23022 156 Street
10	Joshua Asfa	60	v3n	76493 94 Avenue



Successfully modified row(s)!
Table after modification:

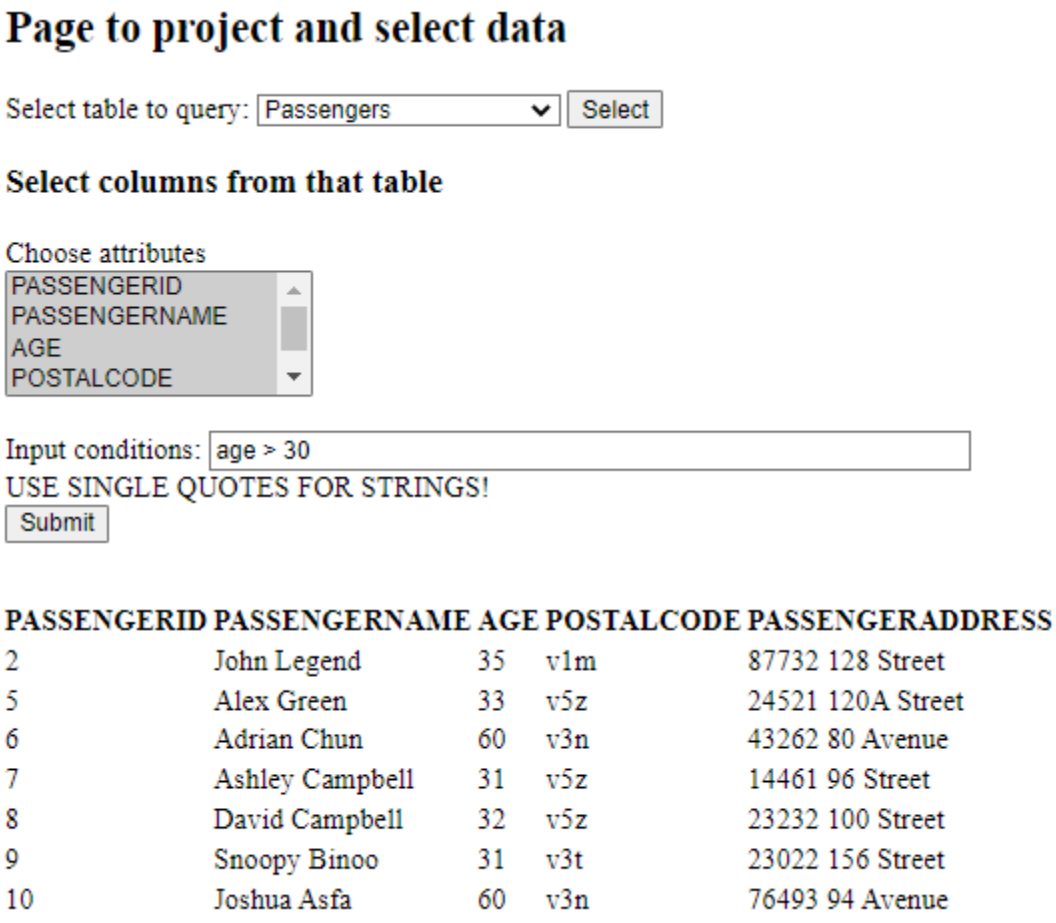
passengerID	passengerName	age	postalCode	passengerAddress
1	Jason Smith	21	v3t	UPDATED ADDRESS
2	John Legend	35	v1m	87732 128 Street
3	Kim Jane	28	v3j	68232 64 Avenue
4	Daniel Jane	29	v3j	68232 64 Avenue
5	Alex Green	33	v5z	24521 120A Street
6	Adrian Chun	60	v3n	43262 80 Avenue
7	Ashley Campbell	31	v5z	14461 96 Street
8	David Campbell	32	v5z	23232 100 Street
9	Snoopy Binoo	31	v3t	23022 156 Street
10	Joshua Asfa	60	v3n	76493 94 Avenue

4. **Selection**
    a. SELECT *

       FROM user_selection

       WHERE user_input_condition

    b. Implementation can be found in “select-project.php” on github



5. **Projection**
    a. SELECT user_selection

       FROM user_selection

       WHERE user_input_condition

    b. Implementation can be found in “select_project.php” on github

Page to project and select data

Select table to query: Passengers Select

Select columns from that table

Choose attributes

PASSENGERNAME
AGE
POSTALCODE
PASSENGERADDRESS

Input conditions: e.g. age > 20 and passengerName = 'Jason'

USE SINGLE QUOTES FOR STRINGS!

Submit

POSTALCODE	PASSENGERADDRESS
v3t	UPDATED ADDRESS
v1m	87732 128 Street
v3j	68232 64 Avenue
v3j	68232 64 Avenue
v5z	24521 120A Street
v3n	43262 80 Avenue
v5z	14461 96 Street
v5z	23232 100 Street
v3t	23022 156 Street
v3n	76493 94 Avenue

6. Join

a. SELECT c.cruiseName, t.ticketClass, p.roomNo
FROM Ticket t, CruiseShip c, PassengerStaysAt p
WHERE t.ticketID = user_input AND
t.hullID = c.hullID AND
t.passengerID - p.passengerID

b. Implementation can be found in “user-view.php” on github

Page to view public data about the cruise ship

2 Submit

Cruise Name: Princess Cruises

Ticket Class: economy

Room Number: PC1-02

Cruise Activities

Activity Name	Start	End
dancing	1200	1500
dancing	1600	1800

Cruise Restaurants

Restaurant Name	Open	Close
Tacofina	800	2200

Page to view public data about the cruise ship

5 Submit

Cruise Name: Viking Cruises

Ticket Class: economy

Room Number: VC3-01

Cruise Activities

Activity Name	Start	End
swimming	1100	2100

Cruise Restaurants

Restaurant Name	Open	Close
Earls	1200	2400

7. **Aggregation with Group By**
    a. SELECT hullID, AVG(Salary)
       FROM Pilots p, Captain c
       WHERE c.crewID = p.crewID
       GROUP BY hullID
    b. Implementation can be found in “avgSalary.php” on github

Average salary of captains for each cruise ship

hullID	Average Salary
1	175000
2	125000
4	162500
5	106500
3	166500

8. **Aggregation with Having**
    a. SELECT crewID, count(*)

       FROM managehospitalities

       GROUP BY crewID

       HAVING count(*) >= user_input

    b. Implementation can be found in “at_least_x_rooms.php” on github

Crew members cleaning at least X rooms

X: 1 Submit

crewID	Name	count
10	Jason	3
11	Andy	2
12	West	1
30	Rachel	1
35	Jasmine	1

Crew members cleaning at least X rooms

X: 2 Submit

crewID	Name	count
11	Andy	2
10	Jason	3

9. **Nested Aggregation with Group By**
    a. SELECT c.hullID, avg(age)
    FROM cruiseship c, passengers p, ticket t
    WHERE c.hullID = t.hullID AND
        t.passengerID = p.passengerID
    GROUP BY c.hullID
    HAVING avg(age) >= ALL (SELECT avg(age)
        FROM cruiseship cr, passengers pa, ticket ti
        WHERE cr.hullID = ti.hullID AND
            pa.passengerID = ti.passengerID
        GROUP BY cr.hullID)
    b. Implementation can be found in “ship_highest_avg_pass_age.php” on github

Ship with highest average passenger age

hullID	Average Age
4	60

10. **Division**

a. SELECT p.passengerID

FROM Passengers p

WHERE NOT EXISTS (SELECT a.stall

FROM Activities a

MINUS

(SELECT pp.stall

FROM PassengersParticipateIn pp

WHERE pp.passengerID = p.passengerID))

b. Implementation can be found in “done-every-activities.php” on github

Passengers that have done every activities

passengerID	PassengerName
3	Kim Jane
5	Alex Green