University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: ___4___

Date: ___2021-11-27____

Group Number: ___45___

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Kelvin Li	82433020	y9y1b	clay2pws@gmail.com
Jin Niu	73491458	b3w2b	dkiv9570@163.com
Mina Yang	88523360	c7x2b	yangybio@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

0. GitHub Repo Info

url link: https://github.students.cs.ubc.ca/CPSC304-2021W-T1/project_b3w2b_c7x2b_y9y1b

important files:

- project-test-3.php (The final project source code)
- project_m4_ddl.sql (The runnable sql script that can be used to create all tables and data)
- README.md (The readme file that contains a short description for every file in the repo)

1. Short description

This project has created a database application for Translink Company's daily operation. We have created **16** tables along with **85** meaningful tuples and accomplished **10** queries with a graphical user interface. The ten queries that we have made include:

1. Allowing to add a bus route & fee into the busFee table (insert),

2. execute selection query in the busFee table or the vehicleOwns table (select),

3. find the [vehicleID, BusRoutes, BusFee] information according to clauses that users like (join),

4. find drivers who have driven every vehicle (division),

5. delete a vehicle from the vehicleOwns table (delete),

6. chose some of the attributes from the passengerTake table (projection),

7. Find the smallest BranchID from translinkCompany while grouping by their names (Aggregation with Group By),

8. update the Balance according to a combination of TransactionID and CardID in the NormalCard table (update),

9. Display the passengerID & BranchID from the compassCardIssueAndHold table, while grouping by BranchID & PassengerID and having BranchID greater than 10 (Aggregation with Having),

10. Find the accountNumber which has the smallest normalCardID per addedAmount from the LoadMoney table (Nested Aggregation with Grouping By).

2. How final schema differed from the previous ones

Changes in Milestone 4 (comparing to milestone 2)

1. combined the table "Buy" and the table "TicketSell" into a table called ticketSellBuy, since this will make more sense and reduce redundancy

2. corrected some reference to the table "passengerTake", since some of them were mistakenly written as "passenger" in milestone 2

3. removed all “ON UPDATE CASCADE” since Oracle do not support them
4. added a few lines of “NOT NULL” constraints in the DDL where we have forgotten to do so for foreign key references in milestone 2

3. List of all SQL queries

1. insert:
   - insert into busFee values (:bind1, :bind2)
2. select:
   - SELECT $column FROM $table
   - SELECT $column FROM $table WHERE $where
   - SELECT $column FROM $table WHERE $where AND $where2
3. join:
   - SELECT * FROM busRoutes, busFee WHERE busRoutes.BusRoutes=busFee.BusRoutes
   - SELECT * FROM busRoutes, busFee WHERE busRoutes.BusRoutes=busFee.BusRoutes AND $where
   - SELECT * FROM busRoutes, busFee WHERE busRoutes.BusRoutes=busFee.BusRoutes AND $where AND $where2
4. division:
   - SELECT DISTINCT d1.DriverID FROM drive d1 WHERE NOT EXISTS(

     (SELECT VehicleID FROM drive)

     MINUS

     (SELECT d2.VehicleID FROM drive d2 WHERE d1.DriverID=d2.DriverID))
5. delete:
   - delete from vehicleOwns WHERE VehicleID = $name
6. projection:
   - SELECT $a1 from passengerTake
   - SELECT $a1,$a2 from passengerTake
   - SELECT $a1, $a2, $a3 from passengerTake
   - SELECT $a1, $a2, $a3, $a4 from passengerTake
7. Aggregation with Group By:

- SELECT MIN (BranchID), name FROM translinkCompany GROUP BY name

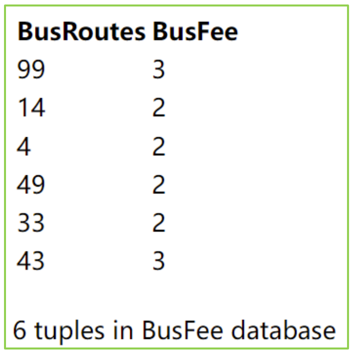
8. update:
    - UPDATE normalCard SET balance = $balance WHERE transactionID = $tID AND cardID = $cID
9. Aggregation with Having:
    - SELECT branchID, passengerID FROM compassCardIssueAndHold GROUP BY branchID, passengerID HAVING branchID>10
10. Nested Aggregation with Grouping By:
    - SELECT accountNumber, addedAmount FROM LoadMoney WHERE normalCardID in (

      SELECT MIN(normalCardID) FROM LoadMoney GROUP BY addedAmount)

4. Screenshots of queries using GUI

1. Insert

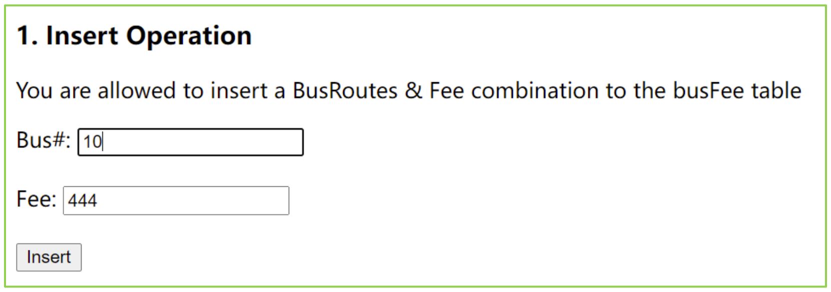
Before Insert:



BusRoutes	BusFee
99	3
14	2
4	2
49	2
33	2
43	3

6 tuples in BusFee database

Insert GUI:



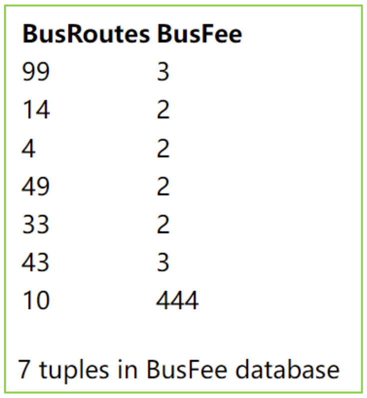
1. Insert Operation

You are allowed to insert a BusRoutes & Fee combination to the busFee table

Bus#: 10

Fee: 444

After Insert:

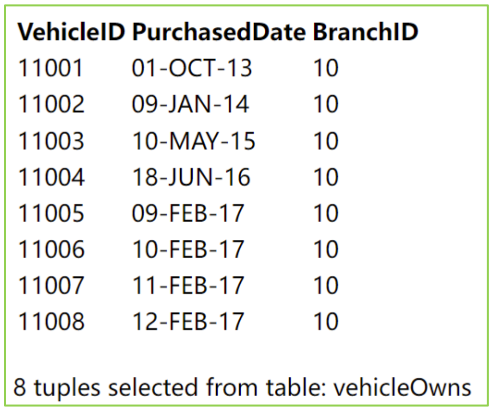


BusRoutes	BusFee
99	3
14	2
4	2
49	2
33	2
43	3
10	444

7 tuples in BusFee database

2. Select

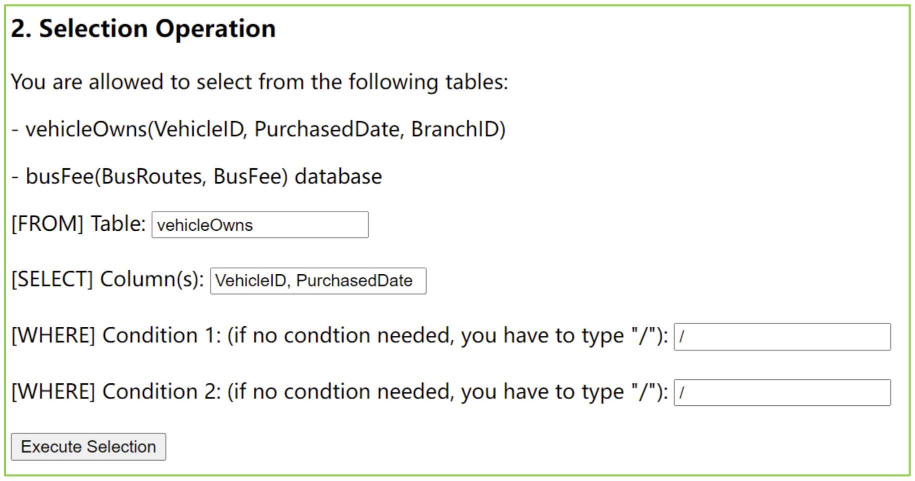
Before select (look at the vehicleOwns table):



VehicleID	PurchasedDate	BranchID
11001	01-OCT-13	10
11002	09-JAN-14	10
11003	10-MAY-15	10
11004	18-JUN-16	10
11005	09-FEB-17	10
11006	10-FEB-17	10
11007	11-FEB-17	10
11008	12-FEB-17	10

8 tuples selected from table: vehicleOwns

Select GUI:



2. Selection Operation

You are allowed to select from the following tables:

- vehicleOwns(VehicleID, PurchasedDate, BranchID)

- busFee(BusRoutes, BusFee) database

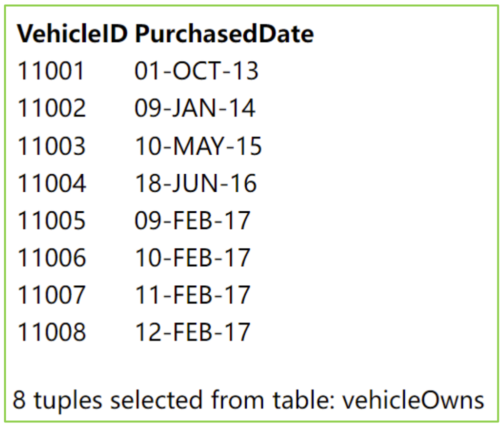
[FROM] Table: vehicleOwns

[SELECT] Column(s): VehicleID, PurchasedDate

[WHERE] Condition 1: (if no condtion needed, you have to type "/"): /

[WHERE] Condition 2: (if no condtion needed, you have to type "/"): /

After Select:

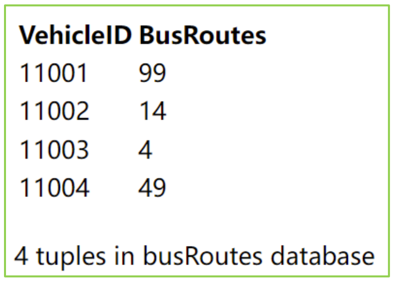


VehicleID	PurchasedDate
11001	01-OCT-13
11002	09-JAN-14
11003	10-MAY-15
11004	18-JUN-16
11005	09-FEB-17
11006	10-FEB-17
11007	11-FEB-17
11008	12-FEB-17

8 tuples selected from table: vehicleOwns

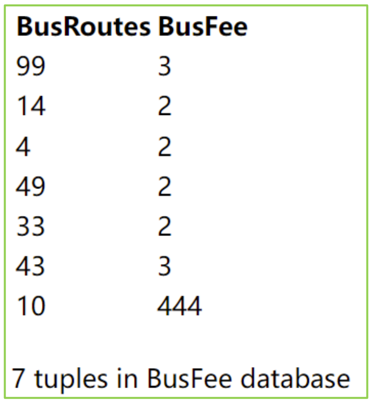
3. Join

Before Join (look at the busRoutes and busFee tables seprately):



VehicleID	BusRoutes
11001	99
11002	14
11003	4
11004	49

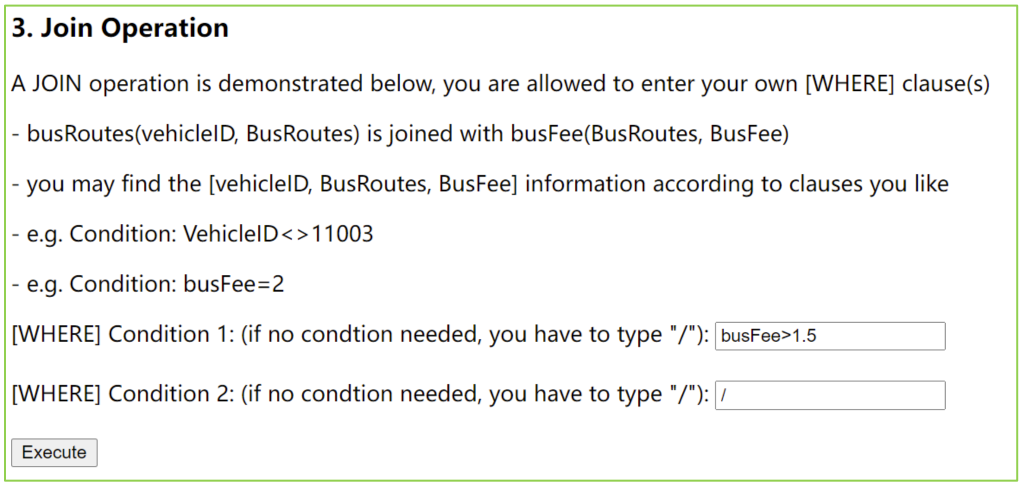
4 tuples in busRoutes database



BusRoutes	BusFee
99	3
14	2
4	2
49	2
33	2
43	3
10	444

7 tuples in BusFee database

Join GUI:



3. Join Operation

A JOIN operation is demonstrated below, you are allowed to enter your own [WHERE] clause(s)

- busRoutes(vehicleID, BusRoutes) is joined with busFee(BusRoutes, BusFee)

- you may find the [vehicleID, BusRoutes, BusFee] information according to clauses you like

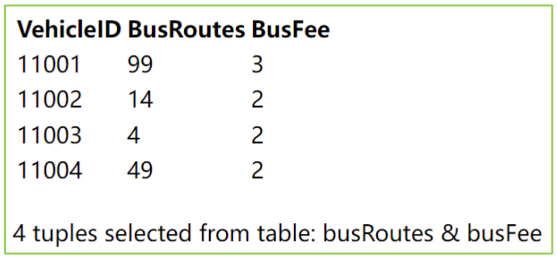
- e.g. Condition: VehicleID<>11003

- e.g. Condition: busFee=2

[WHERE] Condition 1: (if no condtion needed, you have to type "/"): busFee>1.5

[WHERE] Condition 2: (if no condtion needed, you have to type "/"): /

After Join:

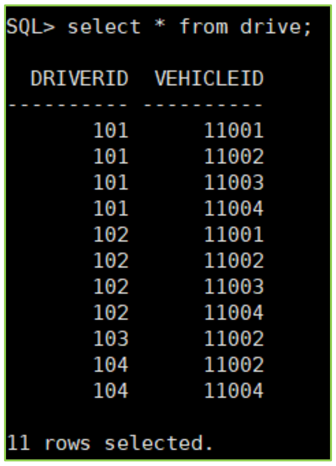


VehicleID	BusRoutes	BusFee
11001	99	3
11002	14	2
11003	4	2
11004	49	2

4 tuples selected from table: busRoutes & busFee

4. Division

Before division (look at the drive table):

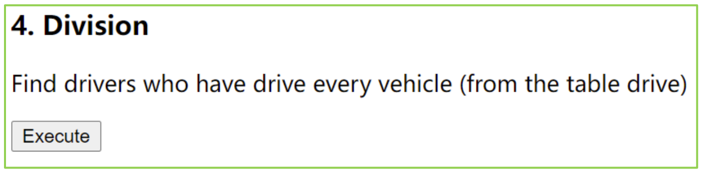


```
SQL> select * from drive;

  DRIVERID  VEHICLEID
---------- ----------
       101      11001
       101      11002
       101      11003
       101      11004
       102      11001
       102      11002
       102      11003
       102      11004
       103      11002
       104      11002
       104      11004

11 rows selected.
```

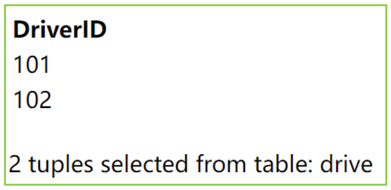
Division GUI:



4. Division

Find drivers who have drive every vehicle (from the table drive)

After Division:

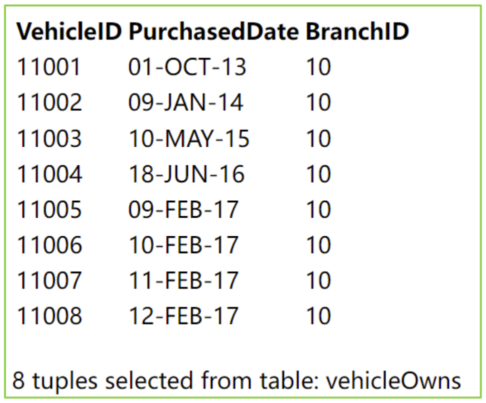


DriverID
101
102

2 tuples selected from table: drive

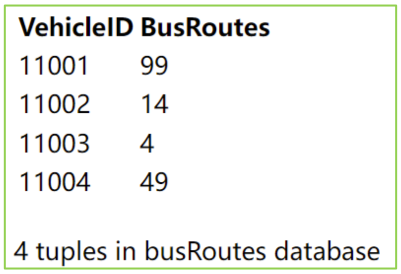
5. Deletion

Before deletion (look at the vehicleOwns, and the busRoutes which references vehicleOwns)):



VehicleID	PurchasedDate	BranchID
11001	01-OCT-13	10
11002	09-JAN-14	10
11003	10-MAY-15	10
11004	18-JUN-16	10
11005	09-FEB-17	10
11006	10-FEB-17	10
11007	11-FEB-17	10
11008	12-FEB-17	10

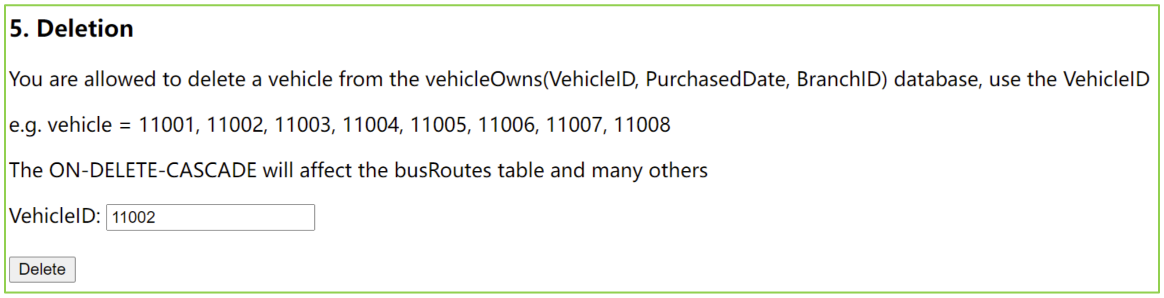
8 tuples selected from table: vehicleOwns



VehicleID	BusRoutes
11001	99
11002	14
11003	4
11004	49

4 tuples in busRoutes database

Deletion GUI:



5. Deletion

You are allowed to delete a vehicle from the vehicleOwns(VehicleID, PurchasedDate, BranchID) database, use the VehicleID

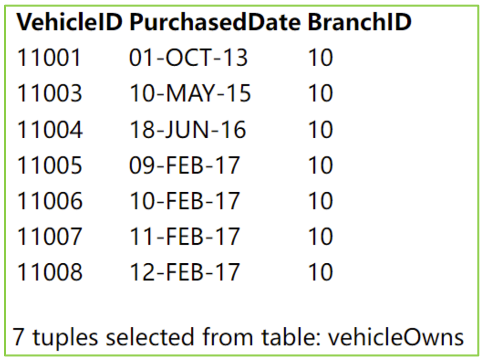
e.g. vehicle = 11001, 11002, 11003, 11004, 11005, 11006, 11007, 11008

The ON-DELETE-CASCADE will affect the busRoutes table and many others

VehicleID: 11002

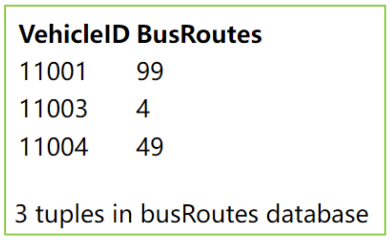
Delete

After Deletion (look at the vehicleOwns, and the busRoutes to check the CASCADE deletion):



VehicleID	PurchasedDate	BranchID
11001	01-OCT-13	10
11003	10-MAY-15	10
11004	18-JUN-16	10
11005	09-FEB-17	10
11006	10-FEB-17	10
11007	11-FEB-17	10
11008	12-FEB-17	10

7 tuples selected from table: vehicleOwns

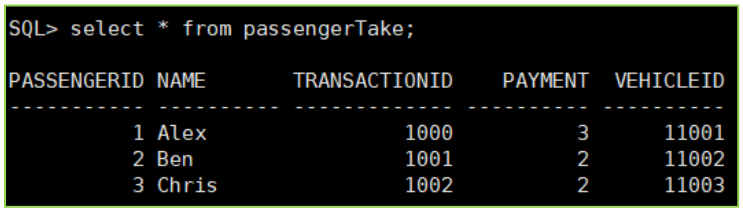


VehicleID	BusRoutes
11001	99
11003	4
11004	49

3 tuples in busRoutes database

6. Projection Operation

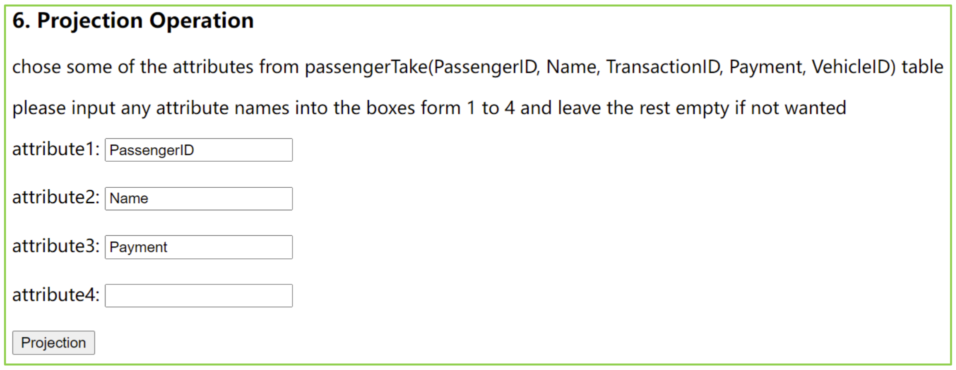
Before projection (look at the passengerTake table):



```
SQL> select * from passengerTake;

PASSENGERID NAME       TRANSACTIONID    PAYMENT  VEHICLEID
----------- ---------- ------------- ---------- ----------
          1 Alex                1000          3      11001
          2 Ben                 1001          2      11002
          3 Chris               1002          2      11003
```

Projection GUI:


6. Projection Operation

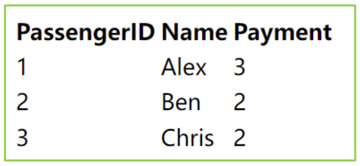
chose some of the attributes from passengerTake(PassengerID, Name, TransactionID, Payment, VehicleID) table

please input any attribute names into the boxes form 1 to 4 and leave the rest empty if not wanted

attribute1:	PassengerID
attribute2:	Name
attribute3:	Payment
attribute4:	

Projection

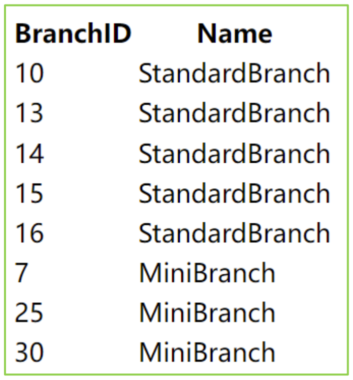
After projection:



PassengerID	Name	Payment
1	Alex	3
2	Ben	2
3	Chris	2

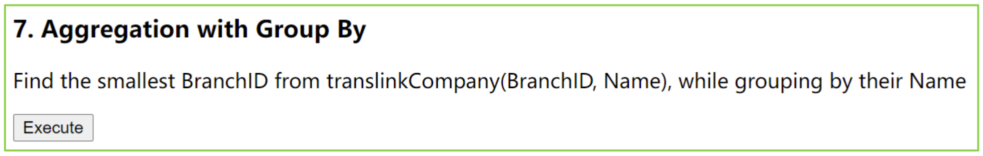
7. Aggregation with Group By

Before execute (look at the translinkCompany table):



BranchID	Name
10	StandardBranch
13	StandardBranch
14	StandardBranch
15	StandardBranch
16	StandardBranch
7	MiniBranch
25	MiniBranch
30	MiniBranch

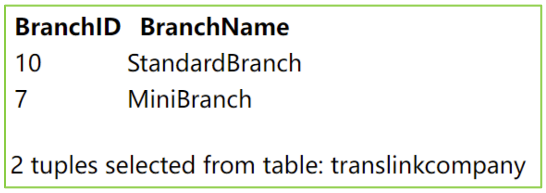
Aggregation with Group By GUI:


7. Aggregation with Group By

Find the smallest BranchID from translinkCompany(BranchID, Name), while grouping by their Name

Execute

After executed:

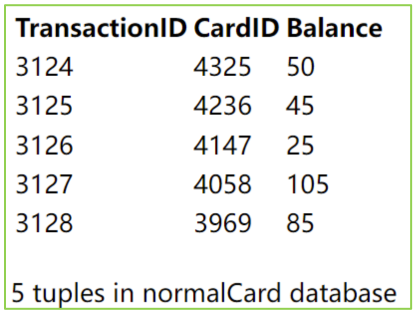


BranchID	BranchName
10	StandardBranch
7	MiniBranch

2 tuples selected from table: translinkcompany

8. Update

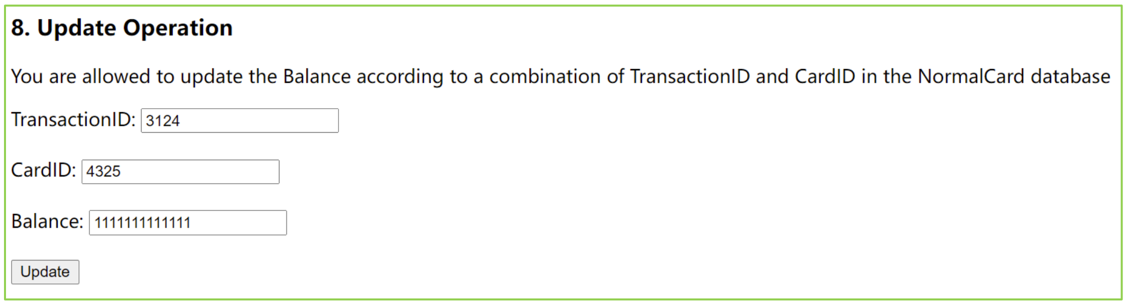
Before update (look at the normalCard table):



TransactionID	CardID	Balance
3124	4325	50
3125	4236	45
3126	4147	25
3127	4058	105
3128	3969	85

5 tuples in normalCard database

Update GUI:



8. Update Operation

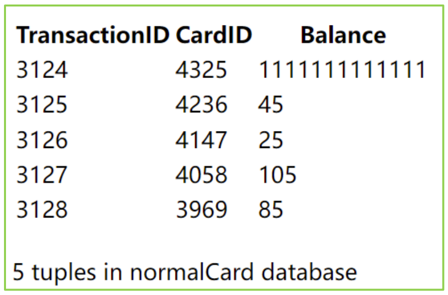
You are allowed to update the Balance according to a combination of TransactionID and CardID in the NormalCard database

TransactionID: 3124

CardID: 4325

Balance: 1111111111111

After Update:

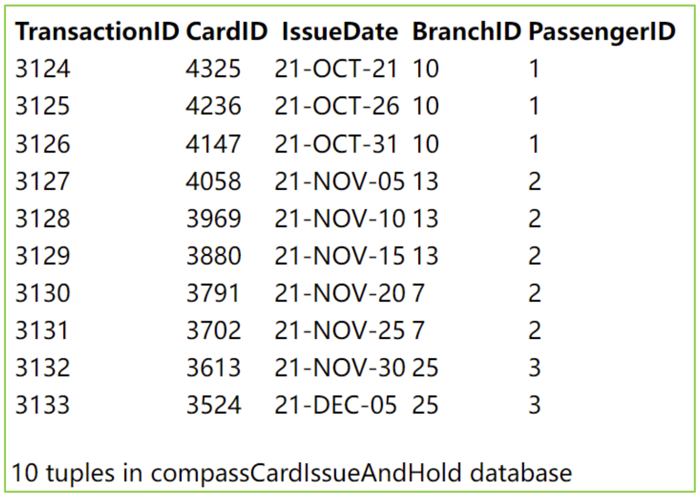


TransactionID	CardID	Balance
3124	4325	1111111111111
3125	4236	45
3126	4147	25
3127	4058	105
3128	3969	85

5 tuples in normalCard database

9. Aggregation with Having

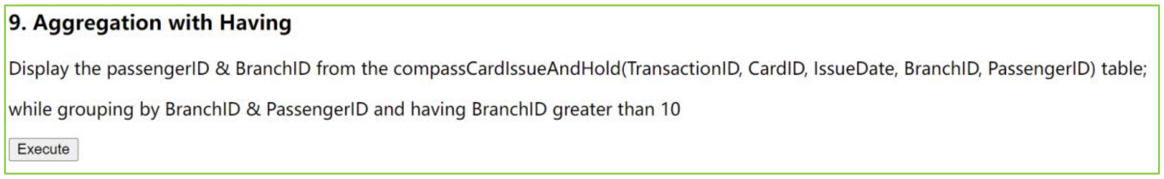
Before execute (look at the compassCardIssueAndHold table):



TransactionID	CardID	IssueDate	BranchID	PassengerID
3124	4325	21-OCT-21	10	1
3125	4236	21-OCT-26	10	1
3126	4147	21-OCT-31	10	1
3127	4058	21-NOV-05	13	2
3128	3969	21-NOV-10	13	2
3129	3880	21-NOV-15	13	2
3130	3791	21-NOV-20	7	2
3131	3702	21-NOV-25	7	2
3132	3613	21-NOV-30	25	3
3133	3524	21-DEC-05	25	3

10 tuples in compassCardIssueAndHold database

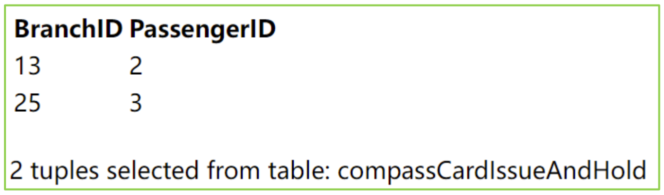
Aggregation with having GUI:



9. Aggregation with Having

Display the passengerID & BranchID from the compassCardIssueAndHold(TransactionID, CardID, IssueDate, BranchID, PassengerID) table; while grouping by BranchID & PassengerID and having BranchID greater than 10

After execute:

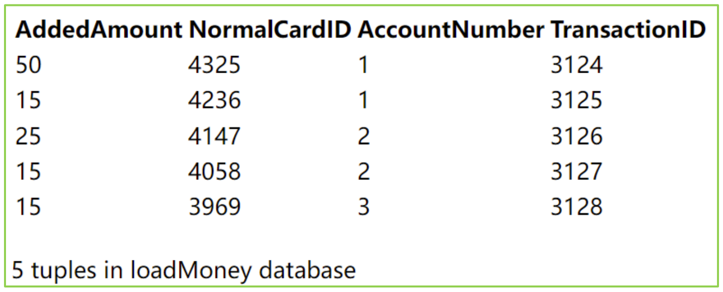


BranchID	PassengerID
13	2
25	3

2 tuples selected from table: compassCardIssueAndHold

10. Nested Aggregation with Grouping By

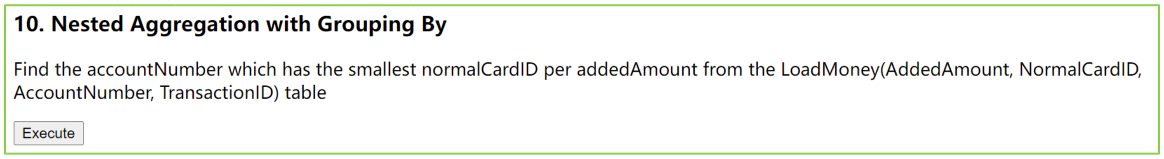
Before execute (look at the loadMoney table):



AddedAmount	NormalCardID	AccountNumber	TransactionID
50	4325	1	3124
15	4236	1	3125
25	4147	2	3126
15	4058	2	3127
15	3969	3	3128

5 tuples in loadMoney database

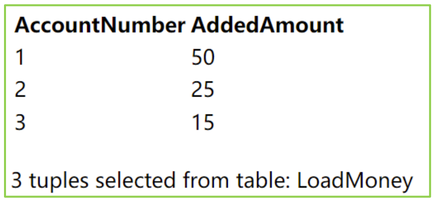
Nested Aggregation with Grouping By GUI:



10. Nested Aggregation with Grouping By

Find the accountNumber which has the smallest normalCardID per addedAmount from the LoadMoney(AddedAmount, NormalCardID, AccountNumber, TransactionID) table

After execute:



```
AccountNumber AddedAmount
1             50
2             25
3             15

3 tuples selected from table: LoadMoney
```