Github Link:
https://github.students.cs.ubc.ca/CPSC304-2021S-T1/project_i6d2d_j7v2b_y5c2b

Links to the web pages:
(1) Login: https://www.students.cs.ubc.ca/~ywang237/login.php
(2) Main Page (Vehicle Rental Service):
https://www.students.cs.ubc.ca/~ywang237/oracle-test.php
(3) Vehicle Rental Transaction:
https://www.students.cs.ubc.ca/~ywang237/vehicleTransaction.php
(4) Employees:
https://www.students.cs.ubc.ca/~ywang237/employees.php

Project Description:

We have successfully made a working version of a Vehicle Rental Service management system. The target users are Vehicle Rental Service employees who help the customers rent vehicles. It incorporates various features such as :-
- Inserting the details of a customer, a vehicle, the Rental Service Agency, the Agency branch location, Employee details, and even the benefits received by an Employee.
- The application system is able to display all the details that are inserted via the user (the operator/admin).
- It includes the feature of projecting certain details(attributes) based on user choice.
- In case, the user wants to fire an employee or remove a customer or a vehicle from the list, the application has the capacity to delete those specific data as well.
- The application can also update certain details of a registered customer like the email or the details of a vehicle if change is required.

Other than the above features, the application includes the working examples of queries dealing with Aggregation, Nested Aggregation with Group By, Join, Select, and Division and thus our application meets all the query requirements. The application has a very basic GUI interface, easy to follow and pretty to look at. With all that said, the application has a huge practical use in the real world and can be taken advantage of by the Vehicle Rental Service Management companies.

The domain that we modeled is Rental cars. We focused on the data stored in the inventory. The inventory aspect of a car dealership is modeled. This includes which cars(models) have been rented out to which customers at what price for how long, which cars are available to rent out, which cars need to be sent to the repair worksop, which new cars need to be

bought from the dealerships, which rental location has what car, car insurance payments and customer reviews. Additionally, a customer can rent a car at a daily rate or for a week, month or year.

The main users of the system are the employees of vehicle rental agency. The objective is to provide a system for the employees to manage the customers, vehicles, dealerships, locations information to optimize the efficiency in vehicle renting and inventory management. The employees are able to access the data of different rental locations and vehicles. Staffs are able to submit various types of rental requests of vehicles of customers' choice to their preferred car rental branch.

This project will be done using the CPSC department's Oracle database system, using PHP, HTML and CSS. We do not anticipate using any special software or hardware.

Final Schema Description:

We made a few minor changes to our schema and project plan. Firstly we added an attribute in the Vehicle table called DriverLicenseID that is a foreign key to the DriverLicenseID Primary Key in the Customer table. This was so we can track if a vehicle is currently occupied. For example, to display all available vehicles at a particular location we searched for vehicles who's DriverLicenseID attribute was NULL as this represents a vehicle that is available to be rented out.

Secondly, we changed the employee entity completely and added attributes like UniqueID, Name, Role and Wage. We also added an entity called Insurance_Benefit that has a benefitID primary key and name. We also introduced a many-to-many relationship between Employee and Insurance_Benefit, this relationship is captured by creating another table called Employee_Benefits that holds a UniqueID as a Foreign Key to the Employee table and Benefit ID which is a Foreign Key to the Insurance_Benefit table. Both attributes are primary keys of the table as it is a many-to-many relationship and both foreign keys are set with an On Delete Cascade statement. We made this change to track more relevant employee information.

Lastly, we deleted some tables/relationships because we realized that they wre redundant or did not add much value to our project if we kept them. Having too many unnecessary relationships on a database is not efficient so we removed the tables such as Bike_Sell, Have, Car_Buy and Car_Sell. The information these tables represent can be easily represented in other tables without violating BCNF.

We also deleted the Customers_phoneNum_email table as we realized our FD for this table was wrong and did not need to use an FD, so we deleted this table as it was a result of BCNF normalization.

Insert Operation

This insert operation takes place when a new customer needs to be added to the database.

Code to send query: executeBoundSQL("insert into Customers values (:bind1, :bind2, :bind3, :bind4, :bind5)", $alltuples);

All Insert Queries used:

insert into Customers values (:bind1, :bind2, :bind3, :bind4, :bind5)
:bind1, :bind2, :bind3, :bind4, :bind5 **refer to** drivingLicenseNumber, Name, phoneNumber, address and emailID **that the customer inputs.**

insert into Rental_Locations_Have values (:bind1, :bind2, :bind3)
bind1: Rental location ID, bind 2: rental service, bind3: rental location name

insert into Rent_LicenseNum values (:bind1, :bind2, :bind3)
bind1: drivingLicenseNumber, bind2: vehicleNumber, bind3: rentalLocationID

insert into Dealership values (:bind1, :bind2, :bind3)
bind1: dealershipID, bind2: dealershipName, bind3: deliveryFees

insert into Rental_Service_Agency values (:bind2, :bind1)
bind1: rentalServiceAgencyName, bind2: budget

insert into TypeOfRental values (:bind1)
bind1: TypeOfRentalName

insert into Vehicles values (:bind1, :bind2, :bind3, :bind4, :bind5, :bind6, :bind7)
bind1: VehicleNumber, bind2: drivingLicenseNumber, bind3: countryOfmanufatcure, bind4: company, bind5: dealershipID, bind6: dealershipID, bind7: category

insert into Rent_type values (:bind1, :bind2)
Bind1: TypeOfRentalName, bind2: DeliveryFees

insert into Rent values (:bind1, :bind3, :bind4)
Bind1: drivingLicenseNumber, bind3: typeOfRentalName, bind4: rentalLocationID

insert into Employee values (:bind1, :bind2, :bind3, :bind4)
Bind1: UniqueID, bind2: EmployeeName, bind3: Role, bind4: wage

insert into Insurance_Benefit values (:bind1, :bind2)
Bind1: BenefitID, bind2: BenefitName

insert into Employee_Benefits values (:bind1, :bind2)
Bind1: EmployeeUniqueID, bind2: BenefitName

One example(Customers):

Select **Display Customers** to view Customer Table before inputting customers with

drivingLicenseNumber	Name	Number	Address	Email
420	Lakshya	420	ubc	l@gmail.com
122334	Amine	123298208	Portland	amine@clbn.com
356356	james	3245245	245245	245245
6969	jac	666	hi	jack123@gmail.com
123678	Jermaine	7787787788	Frankfurt	jermaine@gmail.com
7708630	Lamarr	6046789087	Compton	kdot@gmail.com
9836920	aubery	7788907656	toronto	october@gmail.com
56356	Ayan	6073545245	UBC	ayandas
65356	Ayan Das	6049705013	3545 W34 Ave	ayandas99@gmail.com
123456	jay	12234	1234 fjksjfs	16373@gmail.com
9	lolo	8	u	u
11111	jasasd	3563456	dubai uae	jasasa
69696969	Mcdonalds	6969	Uni village	mc@gmail.com
123	q	123	abc	q@gmail.com

Retrieved data from table Customers:

drivingLicenseNumber 3009004.

Customers

LicenseNumber: 3009004

Name: Ross Geller

Phone: 6049991344

Address: CentralPerk

Email: ross@gmail.com

Insert
vehicleTransaction
Display Customers

Add Customer 3009004 by clicking **insert**

After Adding Customer 3009004 click **Display Customers**

Retrieved data from table Customers:

drivingLicenseNumber	Name	Number	Address	Email
420	Lakshya	420	ubc	l@gmail.com
122334	Amine	123298208	Portland	amine@clbn.com
356356	james	3245245	245245	245245
6969	jac	666	hi	jack123@gmail.com
123678	Jermaine	7787787788	Frankfurt	jermaine@gmail.com
7708630	Lamarr	6046789087	Compton	kdot@gmail.com
9836920	aubery	7788907656	toronto	october@gmail.com
56356	Ayan	6073545245	UBC	ayandas
65356	Ayan Das	6049705013	3545 W34 Ave	ayandas99@gmail.c
123456	jay	12234	1234 fjksjfs	16373@gmail.com
9	lolo	8	u	u
11111	jasasd	3563456	dubai uae	jasasa
69696969	Mcdonalds	6969	Uni village	mc@gmail.com
123	q	123	abc	q@gmail.com
3009004	Ross Geller	6049991344	CentralPerk	ross@gmail.com

Customer 3009004 has been added (bottom of the table)

Update Operation

Though we have other update operation examples for other tables, assigning a customer with a vehicle is the most important update operation.

Query:
UPDATE Vehicles SET drivingLicenseNumber='" . $DID . "' WHERE VehicleNumber='" . $VNUM . "'

UPDATE Customers SET email='" . $new_email . "' WHERE email='" . $old_email . "'

Example:
1) Select VehicleTransaction from customer table after clicking insert and adding a new customer, a new page should appear.

Customers
LicenseNumber:

Name:

Phone:

Address:

Email:

Insert
vehicleTransaction
Display Customers

2)

Display Available Vehicles
Submit

Then Click submit to display available vehicles on the new page

3) This shows all available vehicles, pick any vehicle number

VehicleNumber	drivingLicenseNumber	countryOfManufacture	company	dealership_ID	rentalLocation_ID	category
22		India	IndianCar		134	Car
59		Japan	Mitsubishi 1		134	Car

All Available Vehicles

4) Add the newly inserted driving license number, vehicle number from the available vehicle list, rental type and location of the vehicle and click update.

Assign Vehicle
DrivingLicenseNumber: 3009004

VehicleNumber: 59

RentalType: 1 Week

RentalLocationID: 134

Update

5)After clicking display available vehicles, notice Vehicle 59 does not show up.

All Available Vehicles

VehicleNumber	drivingLicenseNumber	countryOfManufacture	company	dealership_ID	rentalLocation_ID	category
22		India	IndianCar		134	Car

Clicking the display customers button also shows that Ross Geller has vehicle 59 as the vehicle tuple with ID = 59 in the Vehicles table now has the DriverID 3009004 associated with it.

Retrieved data from table Customers:

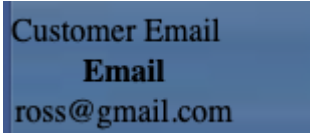| ID	Name	Number	Address	Email	VehicleNumber
420	Lakshya	420	ubc	1@gmail.com	21
11111	jasasd	3563456	dubai uae	jasasa	55
123456	jay	12234	1234 fjksjfs	16373@gmail.com	1
3009004	Ross Geller	6049991344	CentralPerk	ross@gmail.com	59

Selection Query
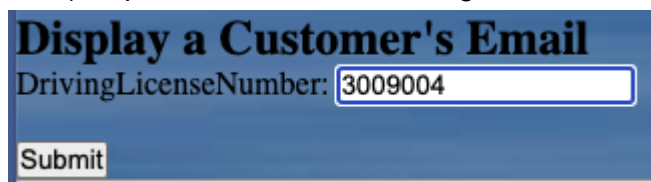
Query:
SELECT email from Customers WHERE drivingLicenseNumber = {$var}
Where $var is the drivingLicenseNumber as an input from the user

SELECT {$insertString} from Vehicles

1) Input 3009004 as the DrivingLicenseNumber

Display a Customer's Email
DrivingLicenseNumber: 3009004

Submit

Customer Email

Email
ross@gmail.com

2) At the bottom the associated email is selected and returned from the database.
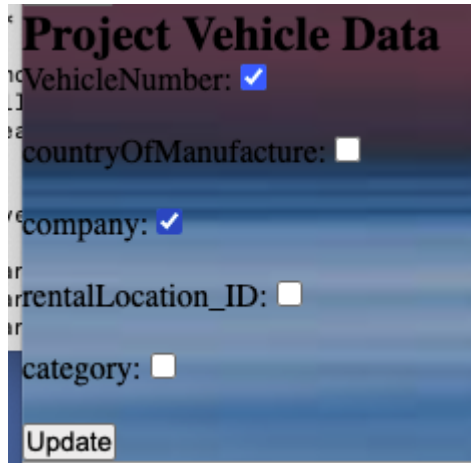
Projection Query

Query:

SELECT {$insertString} from Vehicles
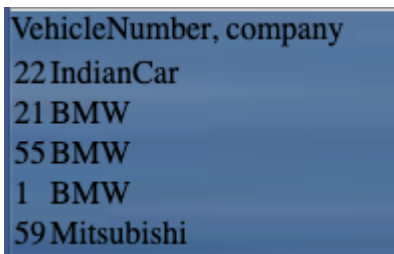$insertString is the list of all columns the user wants to project from the view table.

1) Select which attributes from the vehicle table that you want to project. The checkboxes will be converted into SQL format and inserted to the SQL query above. Both single and multiple columns can be projected.

Multiple

2) The vehicleNumber and VehicleCompany tuples are displayed
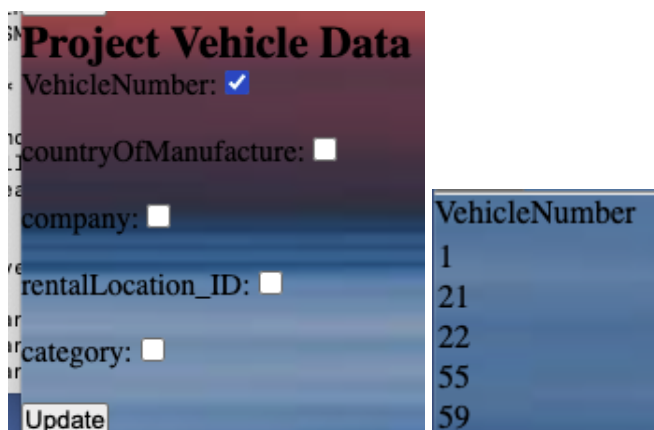
Single
1) Select only vehicleNumber

  2) only vehicle numbers are projected.

JoinQuery

Query:

1) Click DisplayCustomers in the Vehicle Transactions Page

Display Customers
Submit

2)

Retrieved data from table Customers:

ID	Name	Number	Address	Email	VehicleNumber
420	Lakshya	420	ubc	1@gmail.com	21
11111	jasasd	3563456	dubai uae	jasasa	55
123456	jay	12234	1234 fjksjfs	16373@gmail.com	1
3009004	Ross Geller	6049991344	CentralPerk	ross@gmail.com	59

Here the Customers and Vehicles tables are joined as all customer data is displayed along with the vehicle number data from the separate vehicle table.

Delete(cascade) Operation

Query:
DELETE from Employee where UniqueID=$UniqueId

1) Select display employees before delete.

Retrieved data from table Employees:

ID	Name	Role	Wage
14	Laksh	Cashier	20
13	Sylvia	Cashier	20
15	Ayan	CarWash	10

2) Select display benefits to view the different insurance options employees can choose.

Retrieved data from table Insurance_Benefit:
<

BenefitID	Name
1	Insurance1
13	2
2	AXA
3	AXAPlus

3) Select display employee benefits to see all the different benefits each employee has chosen.

Retrieved data from table Employee_Benefits:

EmployeeID	BenefitID
13	1
13	2
13	3
13	13
14	3

4) Type the id of the employee you wish to delete

Delete Employees

Unique Id: 14

Delete

5) Notice employee 14 is no longer in the Employee table and is no longer in the Employee_Benefits table because of the on Delete Cascade statement that is written when the Employee_Benefits table was created.

Retrieved data from table Employees:

ID	Name	Role	Wage
13	Sylvia	Cashier	20
15	Ayan	CarWash	10

Retrieved data from table Employee_Benefits:

EmployeeID	BenefitID
13	1
13	2
13	3
13	13

Aggregation Query

Query:
SELECT {$col1}(DeliveryFees)
 from Dealership
{$col1} is the user input which can be min/max/avg - aggregate function

(1). Type the aggregate function the user wishes to use, click "Check Fees" to see the result

Delivery Fees

Enter max/min/avg: [   ]

Check Fees

(2) before query (dealership table)

All Dealerships

Unique_ID	Name	DeliveryFees
2	2	100
1	1	100
3	UBC Dealership	100
4	Kits Dealership	800

(3) after query:

    i) Min delivery fees:

min DeliveryFees

Fees
100

ii) max delivery fees:

max DeliveryFees

Fees
800

iii) avg delivery fees:

avg DeliveryFees

Fees
275

Nested Aggregation Query

Query:

SELECT company, count(*)
from Vehicles, Customers
where Customers.drivingLicenseNumber = Vehicles.drivingLicenseNumber
group by company

(1) Output the number of **rented** vehicles of different companies

Display the Number of Rented Vehicles of different companies

Submit

(2) Before query (Vehicle Table)

All Vehicles						
VehicleNumber	drivingLicenseNumber	countryOfManufacture	company	dealership_ID	rentalLocation_ID	category
22		India	IndianCar		134	Car
21	420	Germany	BMW		134	Car
55	11111	Germany	BMW		134	Car
1	123456	Germany	BMW	1	134	Car
59	3009004	Japan	Mitsubishi	1	134	Car

Only the Car with vehicleNumber 22 is not rented. Thus, there won't be any tuples with company "IndianCar"

(3)After query:

Company Number of rented vehicles

BMW   3

Mitsubishi 1

Division Query:

Query:

SELECT e.UniqueID, e.Name
from Employee e
where not exists (select * from Insurance_Benefit i
where not exists (select b.UniqueID from Employee_Benefits b

where e.UniqueID = b.UniqueID AND i.BenefitID = b.BenefitID))

(1)Output the ID and Name of the employees who have all the benefits

Display Employees who have all the benefits

Display Employees who have all the benefits

(2)Before query:

    i) Employee Table:

Retrieved data from table Employees:

ID	Name	Role	Wage
13	Sylvia	Cashier	20
15	Ayan	CarWash	10

    ii) Benefits Table:

Retrieved data from table Insurance_Benefit:
<

BenefitID	Name
1	Insurance1
13	2
2	AXA
3	AXAPlus

iii) Employee_Benefit Table:

Retrieved data from table Employee_Benefits:

EmployeeID	BenefitID
13	1
13	2
13	3
13	13

iv) After query:

Display all the employees who have all the benefits:

EmployeeID
13

Other GUI Functionalities:

(1) A Login Page:
Description: after the users input their user username and password, by clicking on "Sign In", they will be directed to the main page - Vehicle Rental Service

Login

👤 Adminname
_____

🔒 Password
_____

Sign In

(2) Users can choose to visit different pages
Description: In the main page (Vehicle Rental Service), users can choose whether they want to visit Vehicle Rental Transaction page or Employees Page. In those pages, users can also choose to go back to the main page by clicking on the links.

i)Main page -> vehicle Transaction page / employee page

Vehicle Rental Service
Go to Vehicle Transcation Page
Go to Employees Page

ii) vehicle Transaction page / employee page -> Main page

Vehicle Rental Transaction

Go Back to Main Page

Employees

Go Back to Main Page