CS-GY 6083 - B Principles of Database Systems Spring 2022
Zichen Yang: zy2486, Ziyu Xiang: zx1356, Zhiying Cui: zc2191
Github: https://github.com/Ethan-zc/WOW_Renting/tree/master
Submission Date: May 12, 2022

Database System Final Project – Part II

Table of Content

1. Introduction

1.1. Business case

WOW (World On Wheels), a car rental company, is looking for a solution to convert file system-based isolated data management to a sophisticated centralized database system. WOW has many car rental offices at various places in the United States. With the growth of its business scale, it is crucial to efficiently manage the business. The business team of WOW has provided the following details and business rules:

- Each office location of WOW maintains various classes of rental vehicles, and each class has its own rental price.
- WOW has customers of types Individual or Corporate. Information on different types of customers should be stored separately.
- Customers use different types of discounts in terms of their customer type. For Individual customers, the coupons that they might receive have a validation period. For Corporate customers, the discount percentage depends on the corporation they are working at.
- WOW records pickup location, dropoff location, etc. for a specific order. Some rental services are with unlimited mileage.
- Invoice automatically generated when the rental service ends.
- Customers should provide card information when making the payment.

1.2. Model Design and Assumptions

Considering the business rules that WOW provides, we are going to use the following major parts in our database design for high efficient business management:

- ZZZ_ADDRESS: Address information for all locations. It prevents duplicate addresses.
- ZZZ_OFFICE: Car rental offices of WOW.
- ZZZ_CAR, ZZZ_VEHICLE: Information for each car.
- ZZZ_ORDER, ZZZ_DISCOUNT(ZZZ_DISCINDI, ZZZ_DISCCORP): Order system which stores details of each order record. It also triggers the creation of a new invoice.
- ZZZ_CUSTOMER(ZZZ_INDIVIDUAL, ZZZ_CORPORATE): Customer system which stores details of each customer.
- ZZZ_ACCOUNT, ZZZ_ACC_ROLE, ZZZ_ROLE: Accounts for logging in car rental website. In our design, we provide two types of roles for each account: customer and administrator. Customers can register accounts through the website while administrator users are assigned through the backend. Different roles have different privileges to access the database.
- ZZZ_INVOICE, ZZZ_PAYMENT: Billing system. It records invoices and each transaction the customer makes.

In the following sections, we are going to describe details about how the model works and how we developed a demo website named “ZZZ Car Rental” website to not only make our product

user-friendly but also maintain its robustness and security. The assumptions that we make when designing our model are as follows:

- The rental office may or may not have rental cars. The M:N relationship is optional.
- The car is a specific type of vehicle identified by a unique VIN(Vehicle Identification Number). This relationship is 1:1.
- The entity customer is the supertype of the individual customer and the corporate customer. The discriminator is the attribute CustType in table customer.
- A customer can make many or zero orders. The M:N relationship is optional.
- One order is associated with only one invoice.
- One invoice can be paid with multiple payments.
- The rental office has its own address. It's a 1:1 relationship. Same as the customer's address.
- An address can be the pickup or dropoff location for many orders.
- When the customer places an order, it needs to declare where the car would be returned, which is the dropoff location in the order.
- One customer must have one corresponding address.
- One office must have one corresponding address, one address could only have one office.
- One car may not be ready to be assigned to any office, therefore the relation between car and office is optional.
- One role can have many accounts. For consideration of security, it is better to use an intersect entity to handle the relationship between role and accounts.

1.3. Logical model

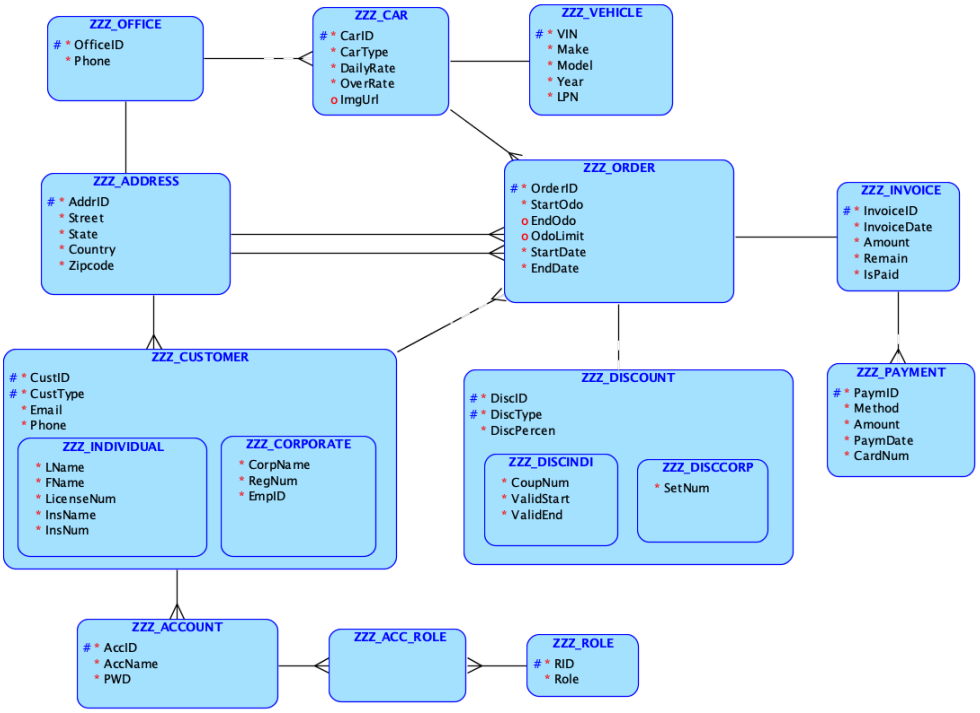


Figure 1: Logical model for WOW.

1.4. Relational model

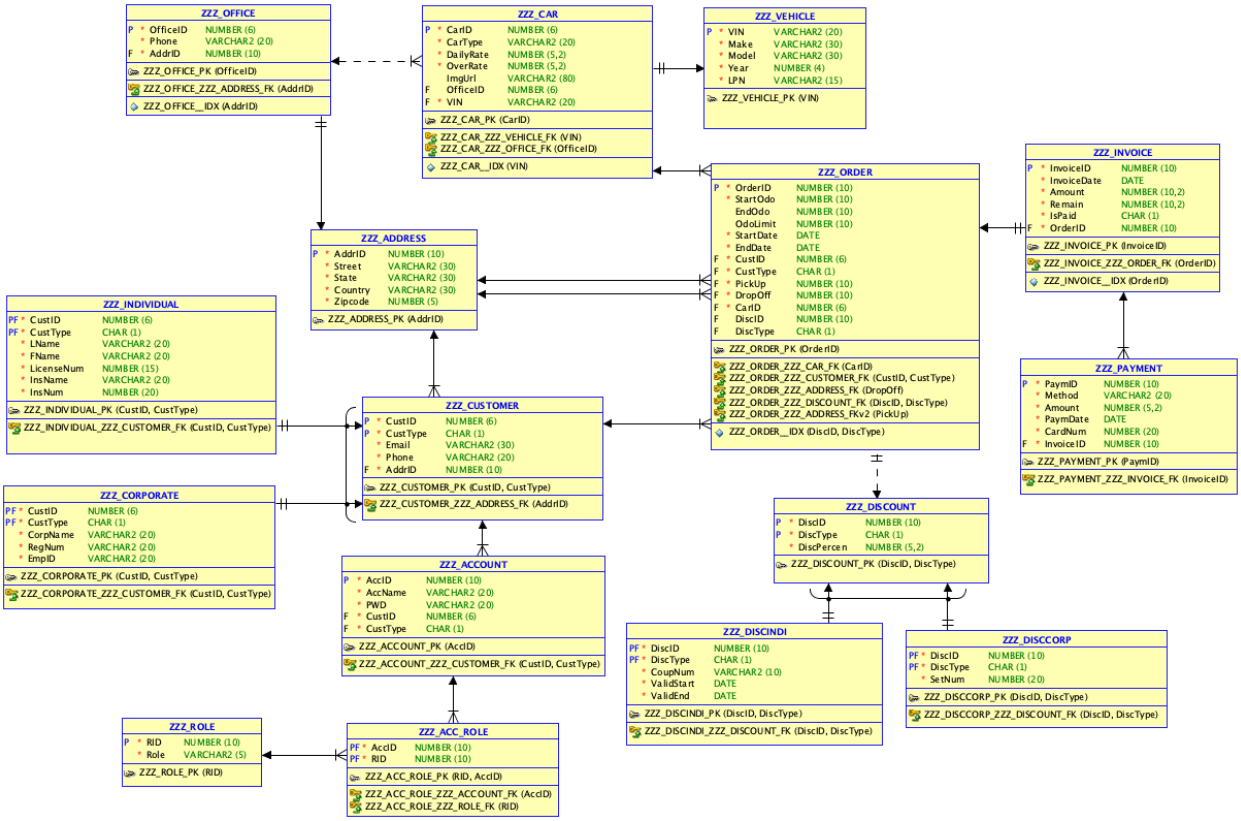


Figure 2: Relational model for WOW.

1.5. Model Design notes

In order for the database to work better with our business rules, we've updated ZZZ_ACCOUNT, ZZZ_ACC_ROLE and ZZZ_ROLE three tables. Which can improve our model design of project part I. ZZZ_ACCOUNT table including basic information such as account name and password according to an account. ZZZ_ROLE table is an intersection table between ZZZ_ACCOUNT and ZZZ_ROLE. The ZZZ_ROLE table represents the role for each account, it indicates the permission of an account in our system. For example, a user account can only check its order history but an administrator account can check all history orders and edit them.

2. Development Environment

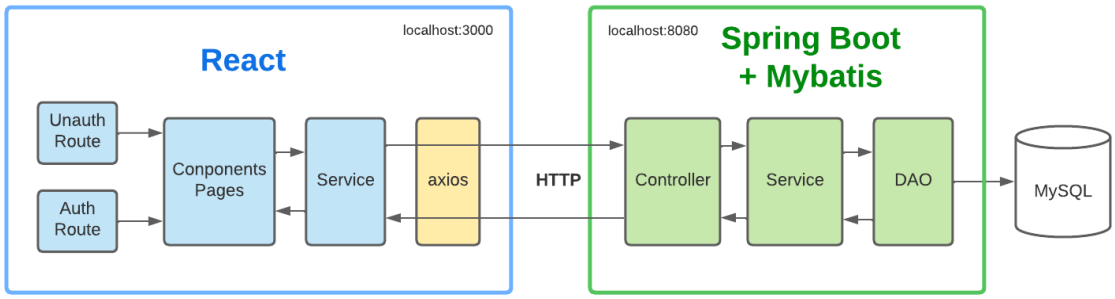
When a guest visits our website, he/she is labeled as an unauthenticated user having limited access to pages until registers an account. The users who sign in with valid accounts can further access more pages and actions. Considering the different roles of authenticated users, they have different privileges to make different kinds of moves. For example, a customer user can access the profile, create an order, and pay his/her bill while for administrator users, they

are able to end an ongoing car rental service, and automatically send invoices to a particular customer.

The architecture for our ZZZ Car Rental demo is given below. Please find the source codes in our Github repository.

Figure 3: Architecture.

2.1. Database



Language: MySQL

Platform: MySQL Workbench

Directory:

```
db
├── DDL.ddl              <-- DDL generated from Data Modeler
├── DDL.sql              <-- MySQL DDL codes converted by sqlines
├── DeleteData.sql       <-- Delete all data in all tables
├── DropAllTables.sql    <-- Drop all tables
├── InitTables.sql       <-- Initialize and create tables
├── InsertData.sql       <-- Insert records
├── Oracle               <-- Logical and relational models
│   ├── WOW
│   └── WOW.dmd
├── README.md
├── TestCase.sql         <-- Worksheet
├── logicalModel.png
└── relationalModel.png
```

2.2. Frontend

Language: JavaScript

Framework: React.js

Material kits: Material Kit Material-UI v4 and Material Dashboard 2 React

Platform: Visual Studio Code

Directory:

```
material-kit-react-main
├── CHANGELOG.md
├── ISSUE_TEMPLATE.md
├── README.md
├── jsconfig.json
├── package-lock.json
├── package.json              <-- Node module dependencies
├── public
└── src                       <-- Source codes
    ├── App.js                <-- Main
    ├── README.md
    ├── assets                <-- Images and theme
    ├── auth-App.js           <-- Entry for registered users
    ├── components            <-- Basic components
    ├── context               <-- Material UI provider
    ├── http-common.js        <-- Config of backend url
    ├── index.js              <-- Main
    ├── pages
    │   ├── account      <-- Account page including profile, order,
billing
    │   ├── authentication    <-- Sign in and sign up pages
    │   ├── trip              <-- Trip page
    │   └── welcome           <-- Welcome page
    ├── routes                <-- Routes
    ├── sections              <-- Worksheet
    ├── services              <-- Methods sending requests to backend API
    └── unauth-App.js         <-- Entry for guests
```

2.3. Backend

Language: Java

Framework: Spring Boot + MyBatis + Swagger-UI

Platform: IntelliJ IDEA

Source codes directory:

```
WOW_Renting/src/main/java/com/wow/rent/
```

```
├── WowRentingApplication.java
├── config                          <-- Configuration
│   ├── CorsConfig.java
│   ├── InterceptorRegister.java
│   ├── SessionConfig.java
│   ├── SqlFilter.java
│   └── UserInterceptor.java
├── controller                      <-- Processing incoming REST requests
│   ├── AccountController.java
│   ├── AddressController.java
│   ├── CarController.java
│   ├── CustomerController.java
│   ├── InvoiceController.java
│   ├── OfficeController.java
│   ├── OrderController.java
│   └── PaymentController.java
├── dao                             <-- Data access objects
│   ├── AccountMapper.java
│   ├── AddressMapper.java
│   ├── CarMapper.java
│   ├── CustomerMapper.java
│   ├── InvoiceMapper.java
│   ├── OfficeMapper.java
│   ├── OrderMapper.java
│   └── PaymentMapper.java
├── demo
│   └── demo.java
├── entry                            <-- Entry classes
│   ├── AccountEntry.java
│   ├── AddressEntry.java
│   ├── CarEntry.java
│   ├── CorpCustEntry.java
│   ├── CorpDiscountEntry.java
│   ├── CorpRegisterRequestEntry.java
│   ├── CustomerEntry.java
│   ├── IndiDiscountEntry.java
│   ├── IndiRegisterRequestEntry.java
│   ├── IndividualCustEntry.java
│   ├── InvoiceEntry.java
│   ├── LoginRequestEntry.java
│   ├── OfficeEntry.java
│   ├── OrderCreateRequestEntry.java
```

```
│   ├── OrderEntry.java
│   ├── OrderUpdateRequestEntry.java
│   ├── PaymentEntry.java
│   ├── PaymentRequestEntry.java
│   ├── ProfileEntry.java
│   ├── TripRequestEntry.java
│   └── VehicleEntry.java
├── model       <-- Result class for sending formatted messages to
frontend
│   └── Result.java
└── service                         <-- Service implementations
    ├── AccountServie.java
    ├── AddressService.java
    ├── CarService.java
    ├── CustomerService.java
    ├── Impl
    │   ├── AccountServiceImpl.java
    │   ├── AddressServiceImpl.java
    │   ├── CarServiceImpl.java
    │   ├── CustomerServiceImpl.java
    │   ├── InvoiceServiceImpl.java
    │   ├── OfficeServiceImpl.java
    │   ├── OrderServiceImpl.java
    │   └── PaymentServiceImpl.java
    ├── InvoiceService.java
    ├── OfficeService.java
    ├── OrderService.java
    └── PaymentService.java
```

API list:

Account Account Controller

- POST `/account/corp/register` corpRegister
- GET `/account/getAccName` getCookieAccName
- GET `/account/getRole` getCookieRole
- POST `/account/indi/register` indiRegister
- GET `/account/islogin` isLogin
- POST `/account/login` login
- GET `/account/logout` logout
- GET `/account/profile` getAccountProfile
- GET `/account/type` getAccountType

Address API Address Controller

- GET `/address/add` addUser
- GET `/address/addrlist` Get all address entries.
- GET `/address/delete` deleteUser

Car Car Controller

- GET `/cars/detailed` getDetailedInfo
- POST `/cars/list` findCars

Customer Customer Controller

- POST `/customer/addcorp` addNewCorpCustomer
- POST `/customer/addindi` addNewIndiCustomer

Invoice Invoice Controller

- GET `/invoice/list` getInvoiceList

Office Office Controller

- GET `/office/carlist` findCarList
- GET `/office/list` findOfficeList
- GET `/office/xss` xssGet
- POST `/office/xss` xssPost

Payment Payment Controller

- GET `/payment/list` getPaymentList
- POST `/payment/pay` makePayment

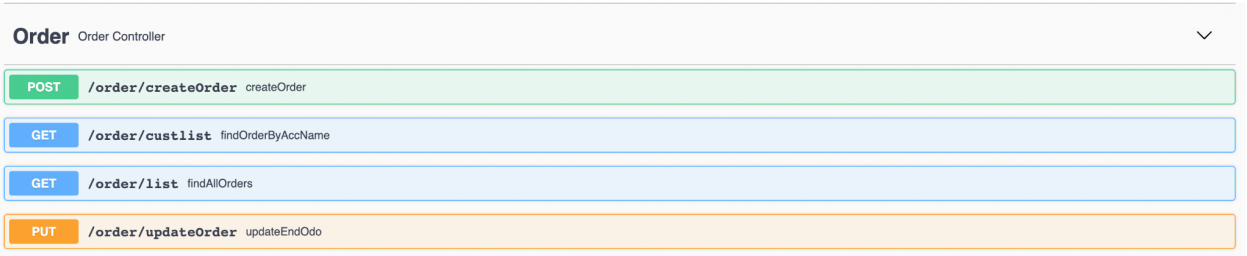


Figure 4: API list.

3. Database Design

Please find the DDL and DML codes in Appendix. Tables and the total number of records in each table are given in Table 1, and the records in each table are listed below.

Table 1: Tables and records.

Table Name	Total Number of Records
zzz_acc_role	3
zzz_account	3
zzz_address	24
zzz_car	11
zzz_corporate	5
zzz_customer	15
zzz_disccorp	7
zzz_discindi	7
zzz_discount	14
zzz_individual	5
zzz_invoice	2
zzz_office	12
zzz_order	6
zzz_payment	0
zzz_role	2
zzz_vehicle	11

	accid	rid
▶	2	1
	3	1
	1	2
*	NULL	NULL

Figure 5: zzz_acc_role.

	accid	accname	pwd	custid	custtype
▶	1	admin	6512bd43d9caa6e02c990b0a82652dca	2	I
	2	aa	6512bd43d9caa6e02c990b0a82652dca	1	I
	3	bb	6512bd43d9caa6e02c990b0a82652dca	7	C
*	NULL	NULL	NULL	NULL	NULL

Figure 6: zzz_account.

addrid	street	state	country	zipcode
1	A St	NY	USA	10001
2	B St	NY	USA	10002
3	C St	PA	USA	15003
4	D St	PA	USA	15004
5	E St	CA	USA	94043
6	F St	CA	USA	94086
7	G St	CA	USA	94089
8	H St	MI	USA	48228
9	I St	MI	USA	48103
10	J St	NJ	USA	7097
11	K St	MA	USA	2139
12	L St	MA	USA	2138
13	M St	MA	USA	2137
14	N St	MA	USA	1604
15	O St	FL	USA	33125
16	P St	FL	USA	33027
17	Q St	FL	USA	32244
18	R St	FL	USA	32808
19	S St	WA	USA	98101
20	T St	WA	USA	98102
21	O St	WA	USA	98004
22	P St	WA	USA	98005
23	Q St	WA	USA	98006
24	R St	WA	USA	98007
NULL	NULL	NULL	NULL	NULL

Figure 7: zzz_address.

carid	cartype	dailyrate	overrate	officeid	vin	imgurl
1	Small car	1.00	50.00	1	299J98JSJIW19923I	https://media.zipcar.com/images/model-image?...
2	Mid-size car	78.28	10.29	2	28WHDJW92U939282	https://media.zipcar.com/images/model-image?...
3	Luxury car	28.17	29.29	3	894739JUHE2932	https://media.zipcar.com/images/model-image?...
4	SUV	92.12	82.23	4	283DWJEHBDU12	https://media.zipcar.com/images/model-image?...
5	Premium SUV	19.29	28.39	5	1231283DHUIHU	https://media.zipcar.com/images/model-image?...
6	Mini Van	12.23	34.72	6	HIH289389223	https://media.zipcar.com/images/model-image?...
7	Station Wagon	49.38	19.30	7	PEI893904212	https://media.zipcar.com/images/model-image?...
8	Small car	28.34	20.00	8	838HIDN8929321	https://media.zipcar.com/images/model-image?...
9	Premium SUV	72.83	22.92	9	EIWDODO123871	https://media.zipcar.com/images/model-image?...
10	Station Wagon	48.20	29.32	10	19HDHJ929391	https://media.zipcar.com/images/model-image?...
11	SUV	23.12	28.23	11	BDU12378492112	https://media.zipcar.com/images/model-image?...
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 8: zzz_car.

custid	custtype	corpname	regnum	empid
6	C	Amazon	amz123	amz456
7	C	Google	gg123	gg456
8	C	Meta	fb123	fb456
9	C	Apple	apple123	apple456
10	C	Linkedin	ln123	ln456
NULL	NULL	NULL	NULL	NULL

Figure 9: zzz_corporate.

custid	custtype	email	phone	addrid
1	I	erte@nyu.edu	(162)-472-844	13
2	I	cxbd@nyu.edu	(162)-472-845	13
3	I	yijj@nyu.edu	(162)-472-846	14
4	I	cvfh@nyu.edu	(162)-472-847	15
5	I	nyif@nyu.edu	(162)-472-848	16
6	C	xcxg@nyu.edu	(162)-472-849	18
7	C	iygj@nyu.edu	(162)-472-850	19
8	C	xzzz@nyu.edu	(162)-472-851	20
9	C	oyou@nyu.edu	(162)-472-852	21
10	C	bnfg@nyu.edu	(162)-472-853	22
NULL	NULL	NULL	NULL	NULL

Figure 10: zzz_customer.

discid	disctype	setnum
8	C	146731
9	C	146757
10	C	158931
11	C	187190
12	C	167880
13	C	778906
14	C	167890
NULL	NULL	NULL

Figure 11: zzz_disccorp.

discid	disctype	coupnum	validstart	validend
1	I	9675731	2022-03-03 00:00:00	2023-03-31 00:00:00
2	I	5760897	2022-02-01 00:00:00	2024-03-01 00:00:00
3	I	903232	2019-05-31 00:00:00	2019-06-30 00:00:00
4	I	5881648	2021-07-01 00:00:00	2021-07-31 00:00:00
5	I	5891678	2021-11-05 00:00:00	2021-12-01 00:00:00
6	I	7689352	2022-03-05 00:00:00	2025-03-15 00:00:00
7	I	7908112	2022-03-05 00:00:00	2024-03-15 00:00:00
NULL	NULL	NULL	NULL	NULL

Figure 12: zzz_discindi.

discid	disctype	discpercen
1	I	30.00
2	I	34.00
3	I	38.00
4	I	20.00
5	I	21.00
6	I	30.00
7	I	39.00
8	C	40.00
9	C	44.00
10	C	48.00
11	C	30.00
12	C	31.00
13	C	40.00
14	C	49.00
NULL	NULL	NULL

Figure 13: zzz_discount.

custid	custtype	lname	fname	licensenum	insname	insnum
1	I	Smith	Jack	218984751	Good Insurance	36571
2	I	Will	Lily	218984752	Better Insurance	36572
3	I	Wade	Micheal	218984753	Awesome Insurance	36573
4	I	Green	Tom	218984754	Cool Insurance	36574
5	I	Gates	Jessie	218984755	Nice Insurance	36575
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 14: zzz_individual.

	invoiceid	invoicedate	amount	ispaid	orderid
▶	1	2022-05-09 00:00:00	4.00	N	1
	2	2022-05-09 00:00:00	17004.00	N	2
*	NULL	NULL	NULL	NULL	NULL

Figure 15: zzz_invoice.

	officeid	phone	addrid
▶	1	(123)-456-789	1
	2	(677)-142-124	2
	3	(456)-634-565	3
	4	(535)-645-089	4
	5	(543)-579-019	5
	6	(082)-977-984	6
	7	(973)-234-893	7
	8	(279)-937-761	8
	9	(769)-733-173	9
	10	(692)-763-182	10
	11	(738)-123-918	11
	12	(162)-472-867	12
*	NULL	NULL	NULL

Figure 16: zzz_office.

	orderid	startodo	endodo	odolimit	startdate	enddate	custid	custtype	pickup	dropoff	carid	disctype	discid
▶	1	1000	1100	40	2020-03-01 00:00:00	2020-03-05 00:00:00	1	I	2	3	1	NULL	NULL
	2	1000	1500	40	2020-03-01 00:00:00	2020-03-05 00:00:00	1	I	2	3	1	NULL	NULL
	3	1000	NULL	NULL	2020-03-01 00:00:00	2020-03-05 00:00:00	1	I	2	3	1	NULL	NULL
	4	1000	NULL	40	2020-03-01 00:00:00	2020-03-05 00:00:00	1	I	2	3	1	I	2
	5	1000	NULL	40	2021-07-01 00:00:00	2021-07-05 00:00:00	1	I	2	3	1	I	4
	6	1000	NULL	40	2021-07-01 00:00:00	2021-07-05 00:00:00	6	C	2	3	1	C	8
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 17: zzz_order.

	paymid	method	amount	paymdate	cardnum	invoiceid
*	NULL	NULL	NULL	NULL	NULL	NULL

Figure 18: zzz_payment.

	rid	role
▶	1	customer
	2	admin
*	NULL	NULL

Figure 19: zzz_role.

	vin	make	model	year	lpn
▶	1231283DHUIHU	Ford	Lincoln	2012	UH198929
	19HDHJ929391	Subaru	Subaru	2014	DUU1738HJD
	283DWJEHBDU12	Toyota	Lexus	2010	12389SH23
	28WHDJW92U939282	Honda	Acura	2020	123899233
	299J98JSJIW19923I	Volvo	Golf	2001	123989828
	838HIDN8929321	Stellantis	Ram	2000	GHJDU288
	894739JUHE2932	General	Buick	2003	123719232
	BDU12378492112	Tesla	Tesla	2020	89273HJHA
	EIWDODO123871	BMW	Rolls-Royce	2012	UD72JDS
	HIH289389223	Mazda	Mazda	2015	HUUE97283
	PEI893904212	BMW	Mini	2013	JIH172HU1
*	NULL	NULL	NULL	NULL	NULL

Figure 20: zzz_vehicle.

4. Web Application

4.1. Welcome

Figure 21 is the welcome page of the website. On top of the page locates the navigation bar, and the car rental logo and welcome messages are shown below it. The next session contains the reservation button which navigates to the trip page. The following is the display section and the last one is a placeholder for other desired components.

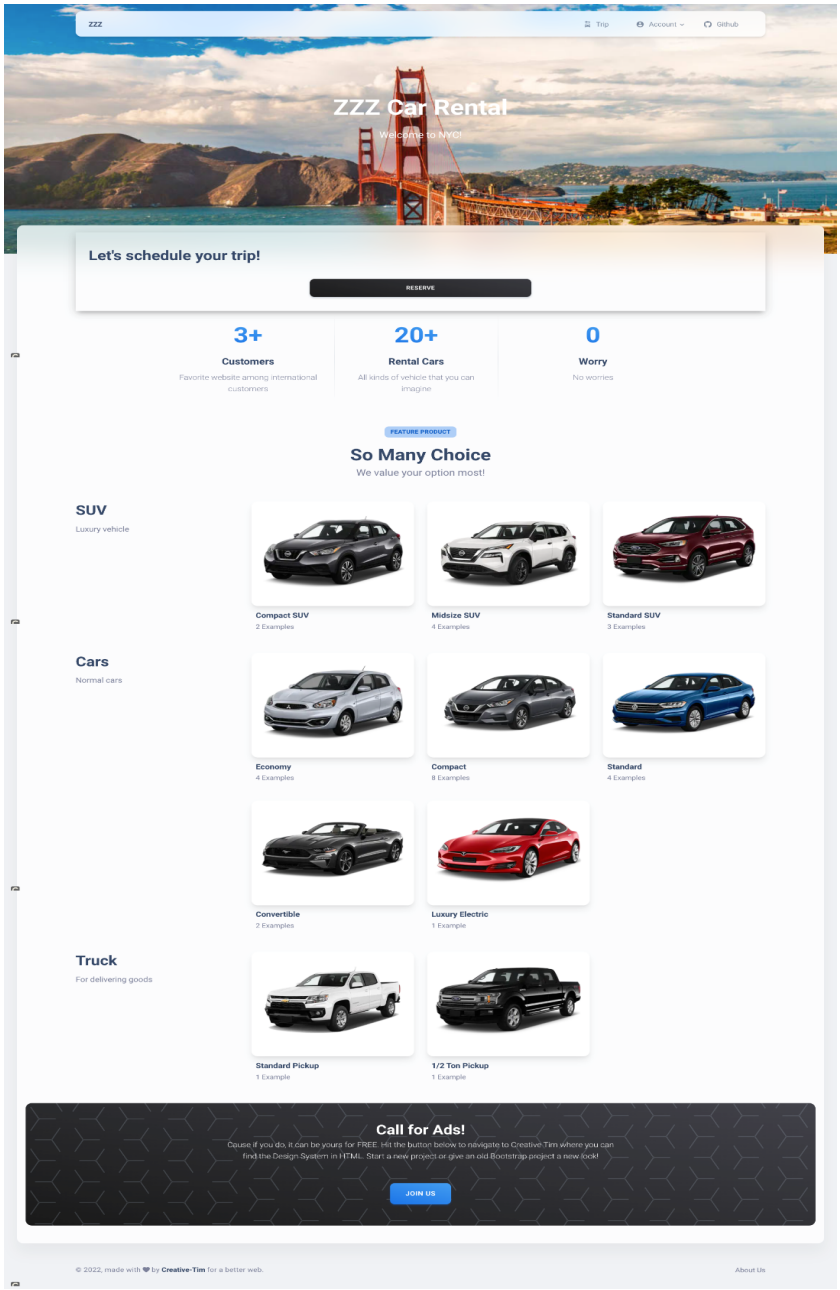


Figure 22: Welcome page.

4.2. Trip

The trip page consists of two major sections. The middle of the page shows the search form, which is used to filter cars that users are looking for. The car list section lies below it, consisting of a car list table and one panel showing the image of the selected car. The frontend sends a user's requirements to the backend. The backend then responds with the recommended car list as shown in Figure 23. An image of a rental car and its brief info would appear on the right panel when the user clicks on the table row.

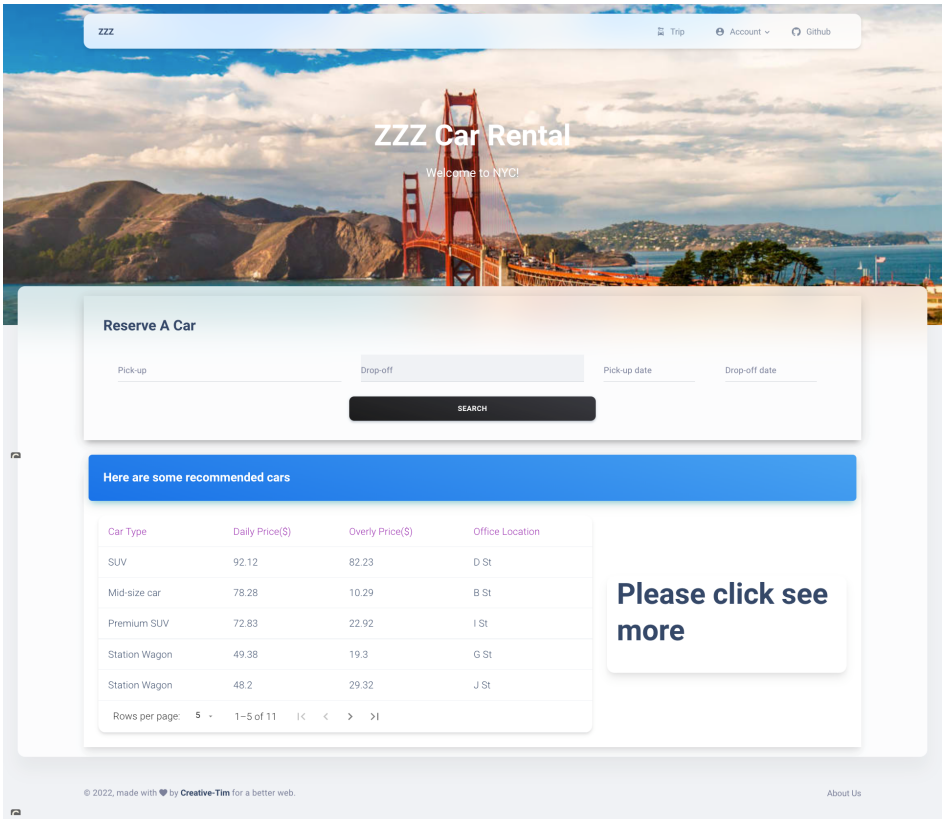


Figure 22: Trip page.

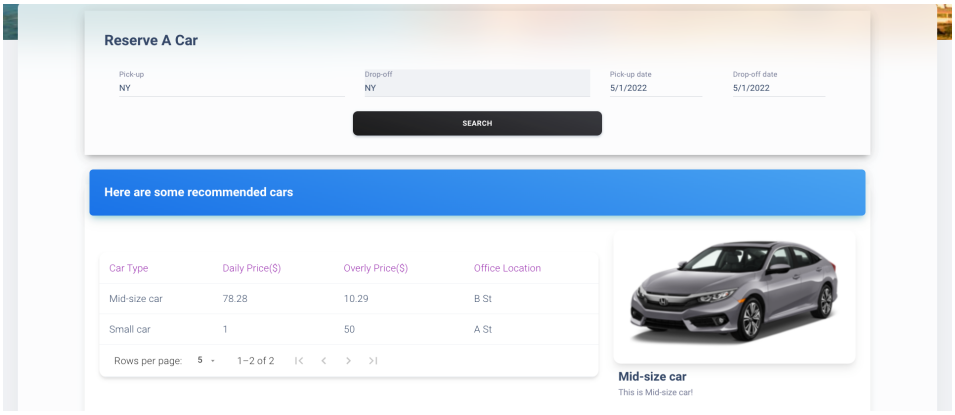


Figure 23: Car results display.

4.3. Authentication

4.3.1. Sign up

Figure 24 to 26 are sign up pages for the website. When the individual user checkbox is checked, this registration form will show up. There are several validations in both frontend and backend.

In frontend:

1) Check if all blanks are filled.
2) Check if the password is greater than 6 characters including at least one lowercase letter, one uppercase letter, one number, and one special character.
3) Check if the license number, insurance number, and zipcodes are all numbers.

In backend:

1) Check if the account or email is used. Error messages would appear if any information is invalid and the corresponding input would be highlighted.

We use another layout of forms for corporation users as given in Figure 25 since this group of users should prove different information from individual users. Messages modal pop out if the registration is successfully submitted, see in Fugure 26.

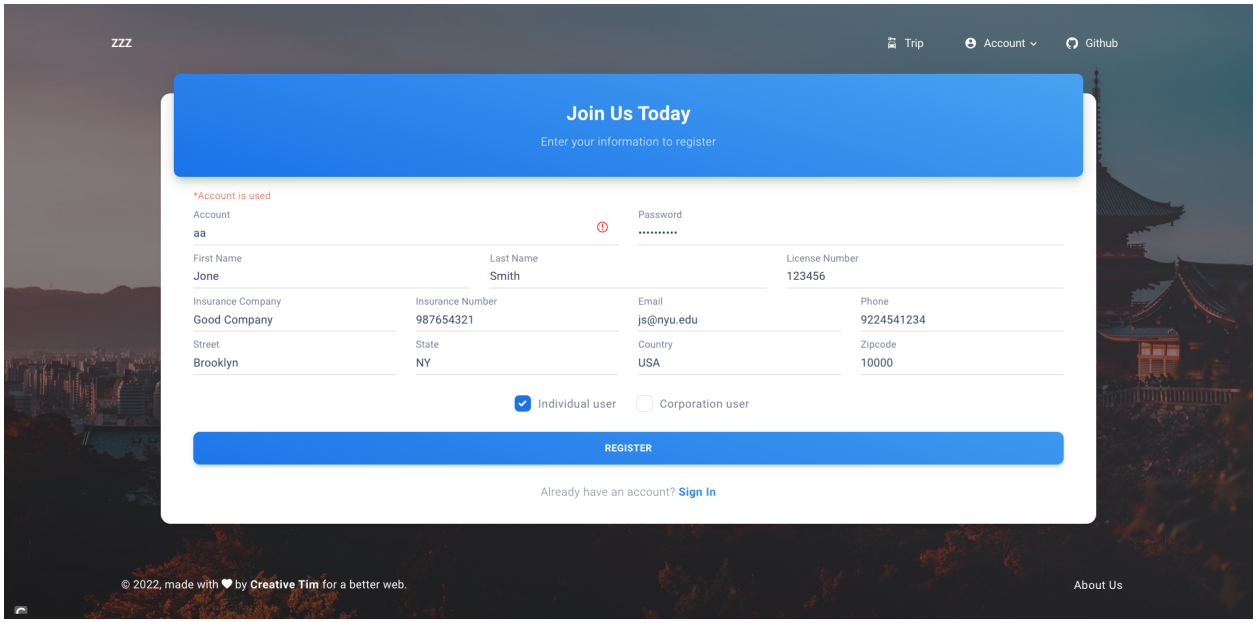


Figure 24: Sign up page for individual user registration.

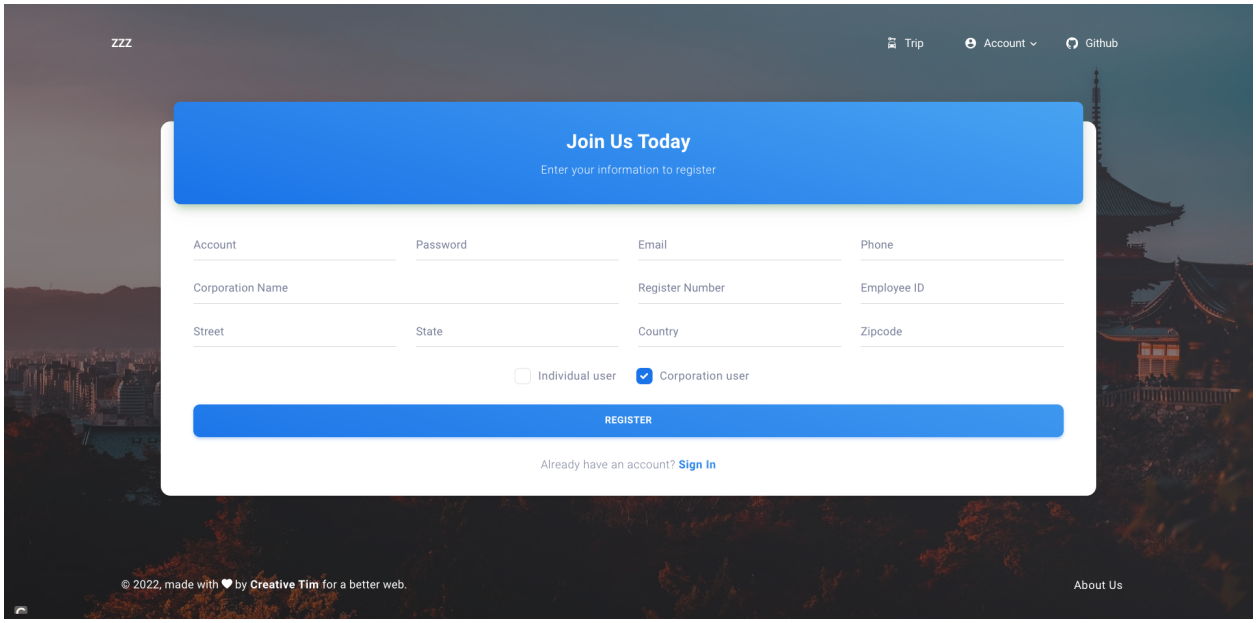


Figure 25: Sign up page for corporation user registration.

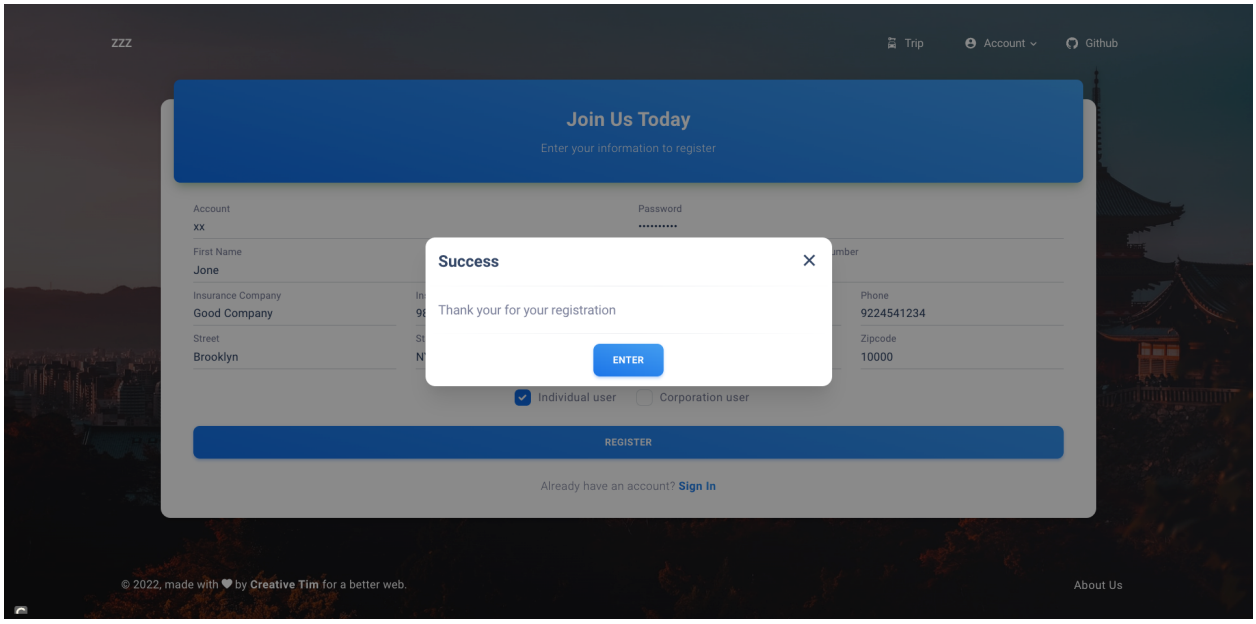


Figure 26: Sign up page when a registration succeeds.

4.3.2. Sign in

Once a user has a registrated account, he/she can sign in on the sign in page. Fill out the account and password, click the sign in button, and then the frontend will send the authentication request to check the user's identification. The backend will send back the success message along with cookies if the account is valid, otherwise, the sign in would fail.

The backend sends back two kinds of error messages:

1) “Account does not exist!” if the account is not in our database.
2) “Account name or password error!” if the password is wrong.

If the remember me switch is on, it enables the browser to remember the account name by caching it to the localStorage. This option will assist the users to remember complex account names. It boosts the user experience.

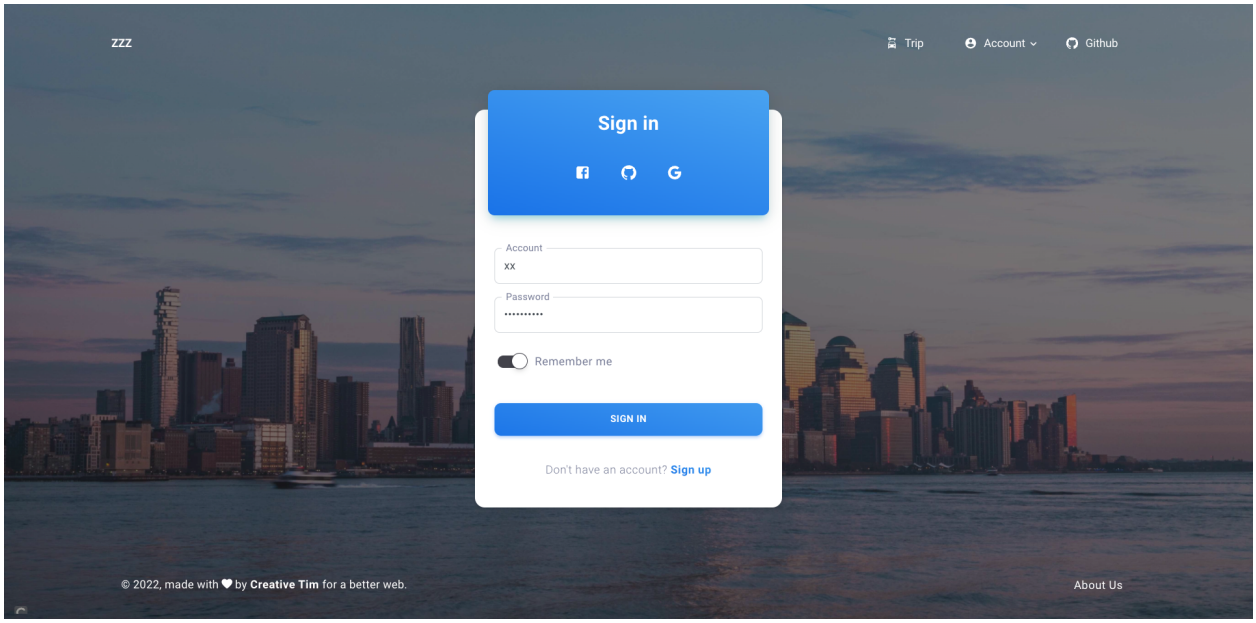


Figure 27: Sign in page.

Once the users sign in, they will see some changes on the website. Since we have implicitly preset different routes for the unauthenticated and the authenticated users, one change can be obviously found in the navigation bar. Another change is the pages users can access depending on the different roles of authenticated users(customer, administrator). We also assign different privileges to each of them.

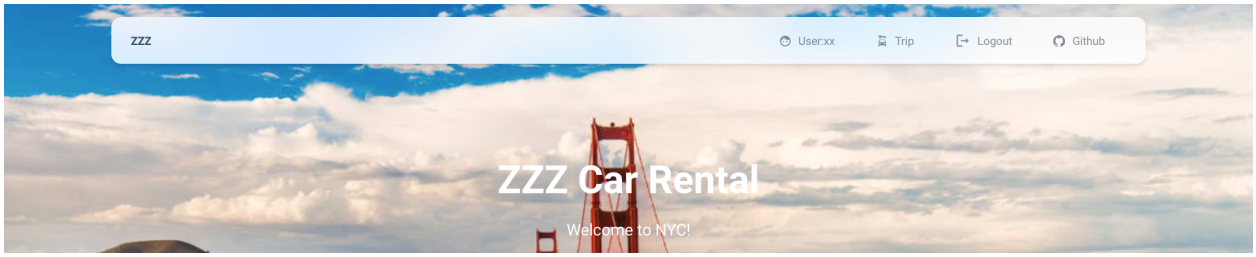


Figure 28: Changes in the navigation bar when the user is logged in.

4.4. Account - Customer (Individual and Corporate users)

4.4.1. Profile

Once users enter their account, the first page they will see is the profile page. The side navigation bar locates on the left of the page, which is automatically hidden when the window size is getting smaller. Different information will appear on the profile page in terms of the user's customer type, see Figures 29 and 30.

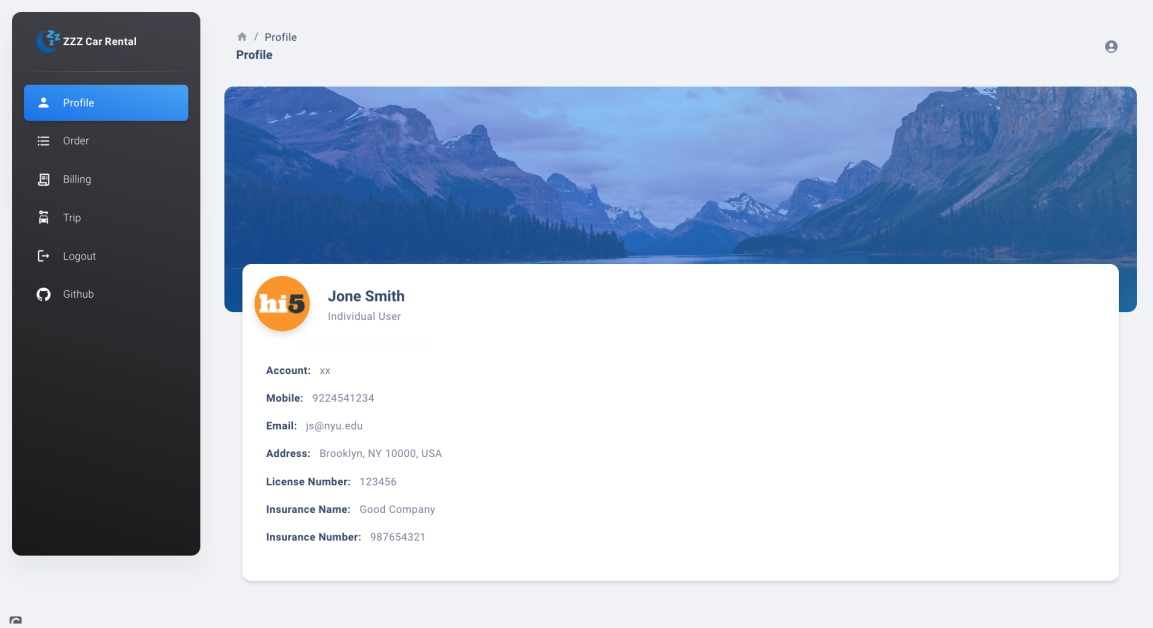


Figure 29: Profile page for the individual user.

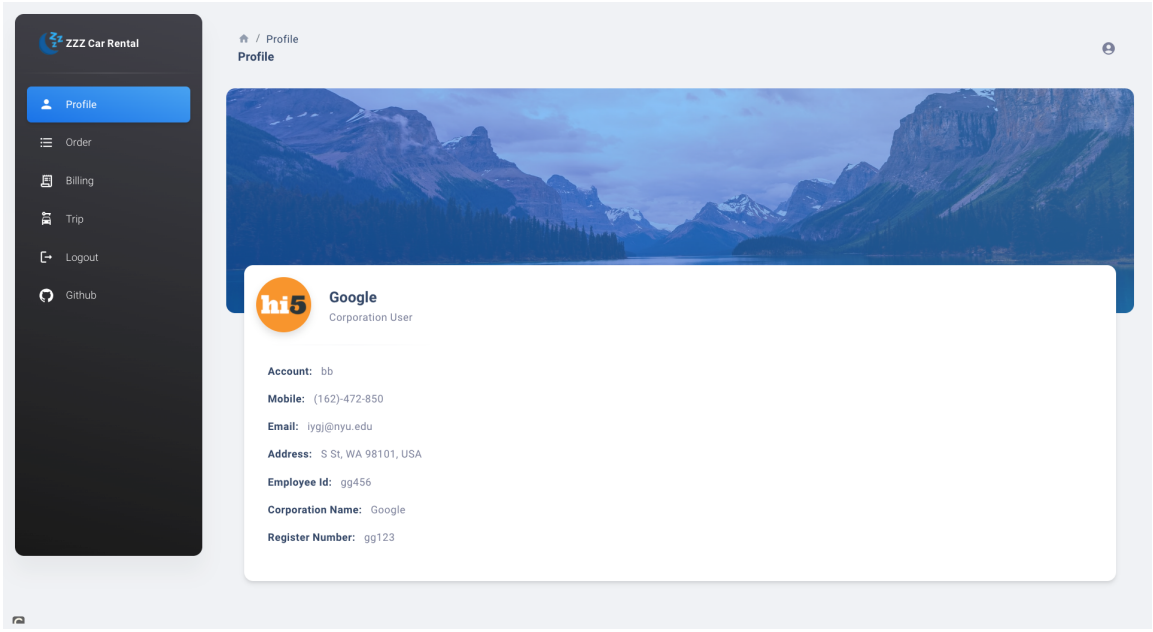


Figure 30: Profile page for the corporation user.

4.4.2. Order

“New order” panel occurs on the order page when the user wants to start a new rental service. There are several ways to create a new order:

1) A guest selects a car on the trip page → Redirect to sign in page → The guest logs in a valid account → New order occurs on the order page.
2) A guest selects a car on the trip page → Redirect to sign in page → The guest signs up for a new account → The guest signs in with the new account → New order occurs on the order page.
3) An authenticated user selects a car on the trip page → Redirect to order page and new order exists.

The story behind it is using sessionStorage to temporarily save the new order information. When a user selects a desired car service, the frontend will first save the information to the browser’s sessionStorage. Then, check the user’s identification by asking the backend. If the backend says it is an authenticated user, the frontend will redirect to the order page, otherwise, redirect to sign in page. The new order information will be kept in the storage until the user signs in and be removed when the new order is submitted, the user logs out or tries a new session.

The users can manually enter a coupon or corporate discount they might have when creating a new order. The backend will check its validation as usual.

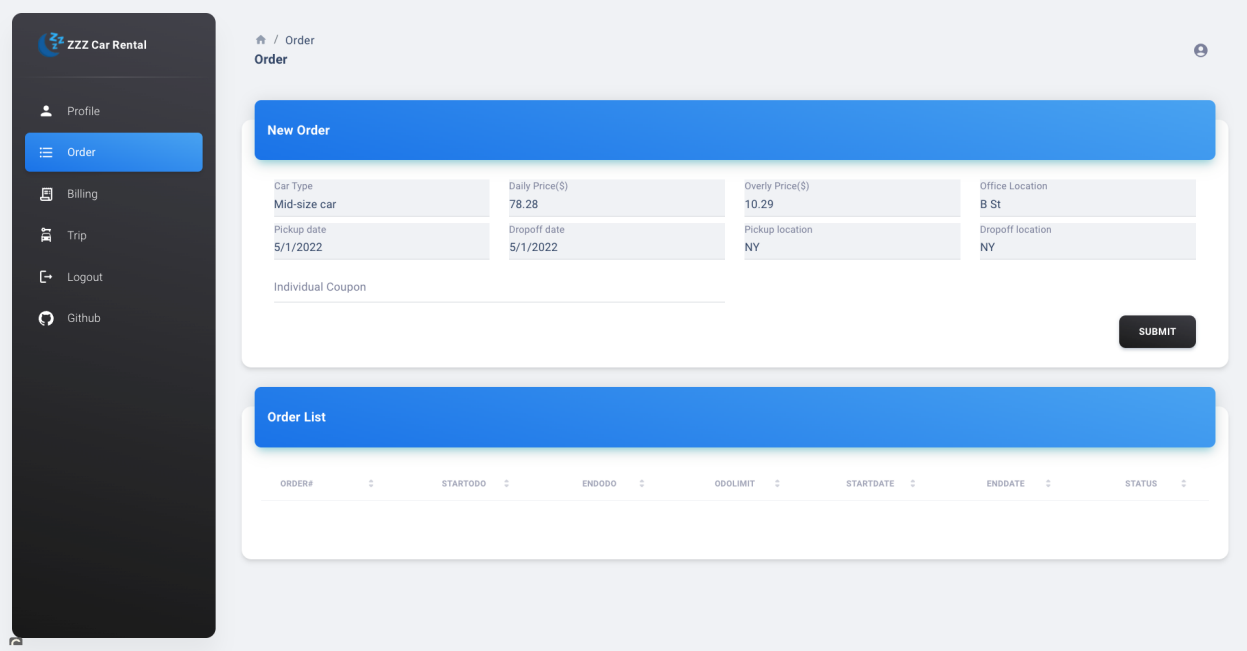


Figure 31: Order page for creating a new order.

Figure 32 gives the order list table. The ongoing order is labeled as IN PROCESS in a blue box, while the completed order is marked as COMPLETED.

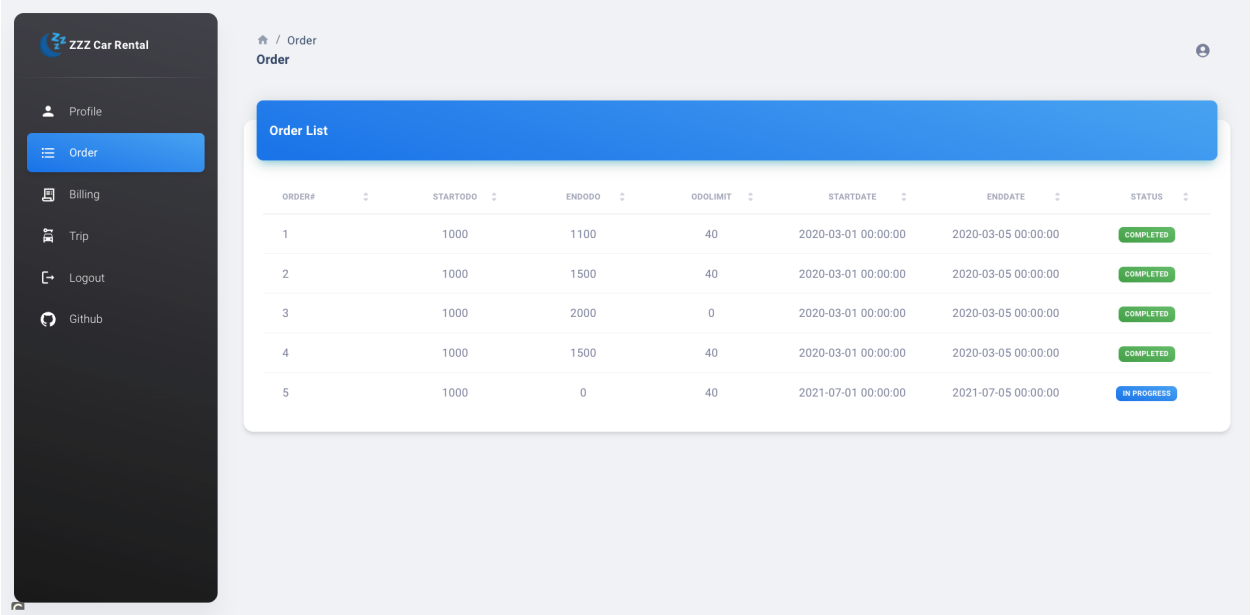


Figure 32: Order page for listing ongoing and completed orders.

4.4.3. Billing

The users can find their invoices and transaction history on the billing page. A user can make a payment by clicking the PAY button and a modal as shown in Figure 34 will pop out. As we know, every bank has its own identity code in the card number, here we simulated the scenario with a simple identification. The card starting with 1234 is identified as a credit card, and the one with 5678 is identified as a debit card. When the total amount is all paid, the PAY button will be disabled.

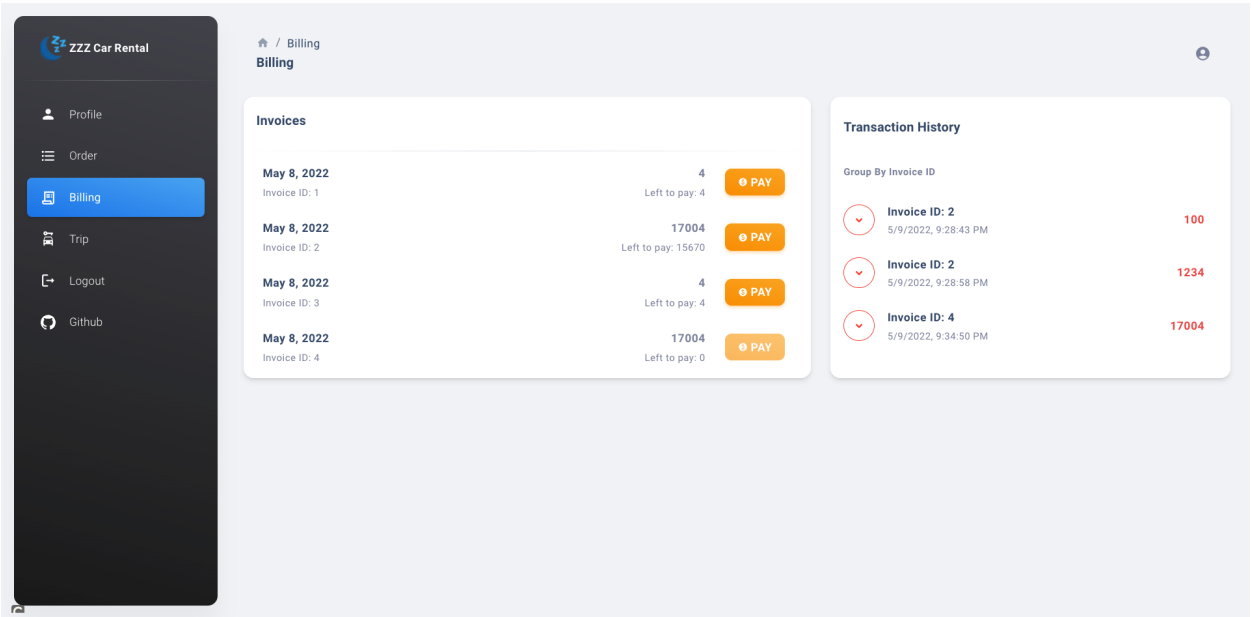


Figure 33: Billing page.

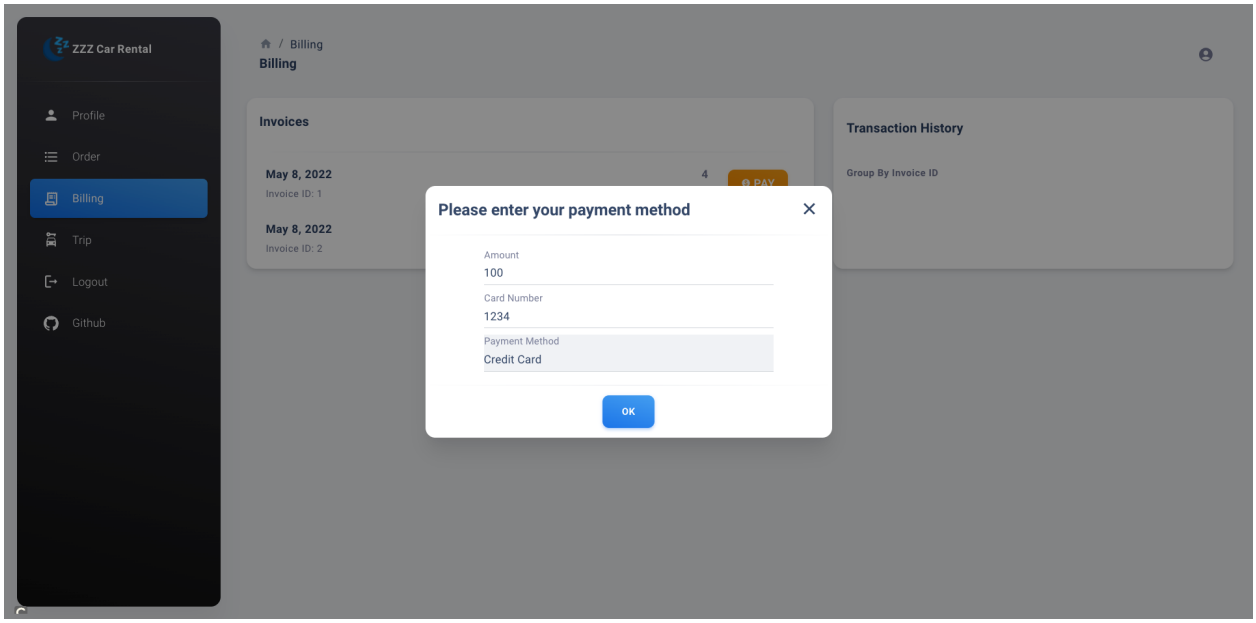


Figure 34: Payment form.

4.5. Account - Administrator

4.5.1. Profile

The profile page for administrator users is similar to the customer user.

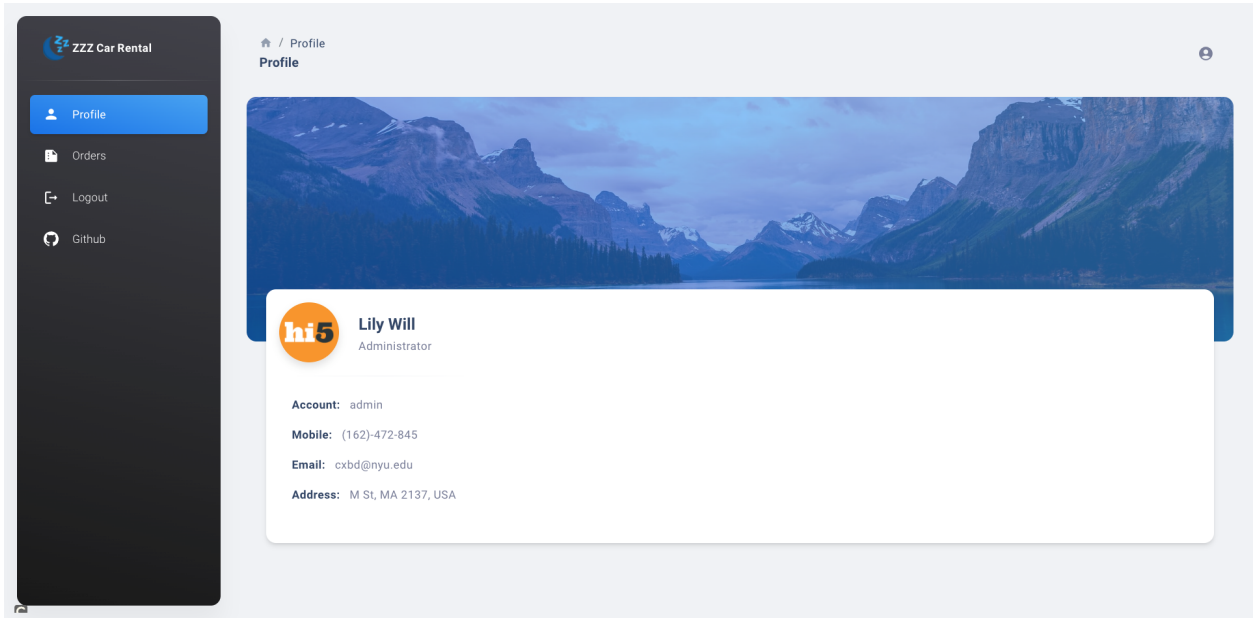


Figure 35: Profile page for the administrator.

4.5.2. Order

Administrator users can see all orders in the database. They have the privilege to finish orders and send the invoice to the corresponding user. An administrator enters the end odometer when the customer ends the service, and submits it. The completed order is marked COMPLETED and the ongoing order is marked IN PROCESS.

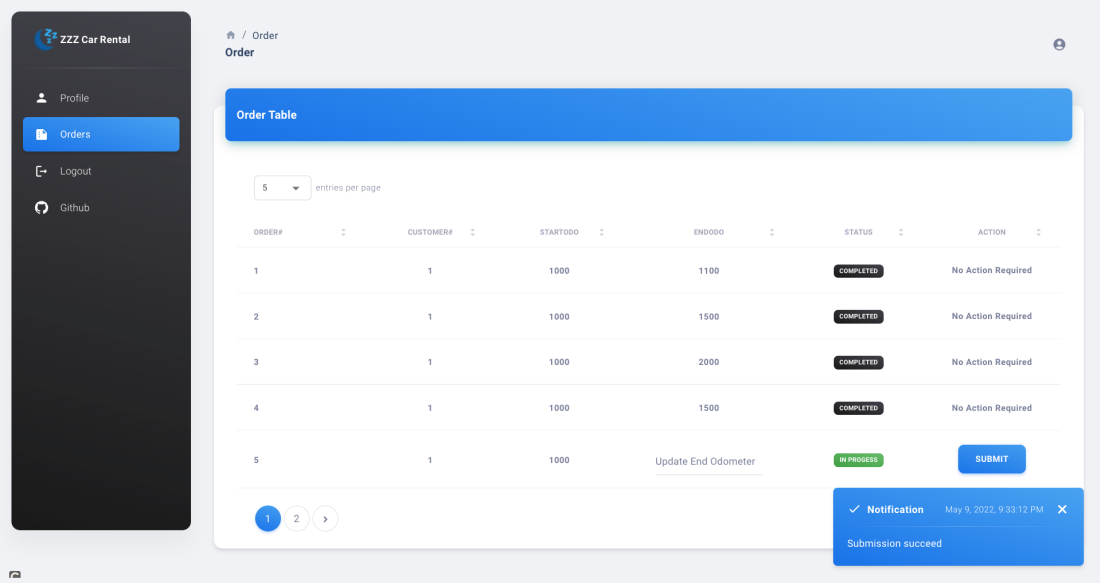


Figure 36: Administrator's order page.

5. Security Features

5.1. SQL injection

SQL Injection is one of the most common web hacking techniques, SQL Injection usually occurs when user input is prompted such as user name and password, and instead of normal input, user input is a sql statement that would be run on the database system. SQL Injection attacks allow attackers to spoof identity, tamper with existing data, cause reduplication issues.

Here in order to avoid SQL Injection, PreparedStatement in Mybatis is used.

For example, if we want to create a mapper that find the user entry from table account with input user name, we could write the following mybatis mapper:

```
@Select("SELECT * FROM zzz_account WHERE accname = ${accName}")
AccountEntry findAccByAccName(String accName);
```

Here when the ${} is used, the user input accName would be directly used as a part of the SQL query statement, which would have the risk of SQL Injection. In this situation, if user input is **'' or '1=1'**, all data in table zzz_account would be printed out, since the SQL would become:

```
SELECT * FROM zzz_account WHERE accname = '' or 1=1
```

Then the SQL statement would be executed. In order to prevent the SQL injection attack,we use **#{}** to put user input into the SQL statement. When #{} is used, instead of directly putting the

user input as a part of SQL statement and executing, PreparedStatement would be used. MyBatis in this way would use JDBC's PreparedStatement. As for the example above, when #{} is used:

```
@Select("SELECT * FROM zzz_account WHERE accname = #{accName}")
AccountEntry findAccByAccName(String accName);
```

The SQL statement would become:

```
SELECT * FROM zzz_account WHERE accname = ?
```

This SQL statement would be handled as PreparedStatement, which would helps to prevent SQL Injection.

5.2. XSS

As for XSS Prevention, we use package called mica-xss to do the filtering for us:

```
<!-- XSS attack filtering -->
<dependency>
    <groupId>net.dreamlu</groupId>
    <artifactId>mica-xss</artifactId>
    <version>2.0.9-GA</version>
</dependency>
<dependency>
    <groupId>net.dreamlu</groupId>
    <artifactId>mica-core</artifactId>
    <version>2.1.0-GA</version>
</dependency>
```

Figure 37: Mica xss.

By using this package, all the illegal user input would be filtered by mica-xss in this process:

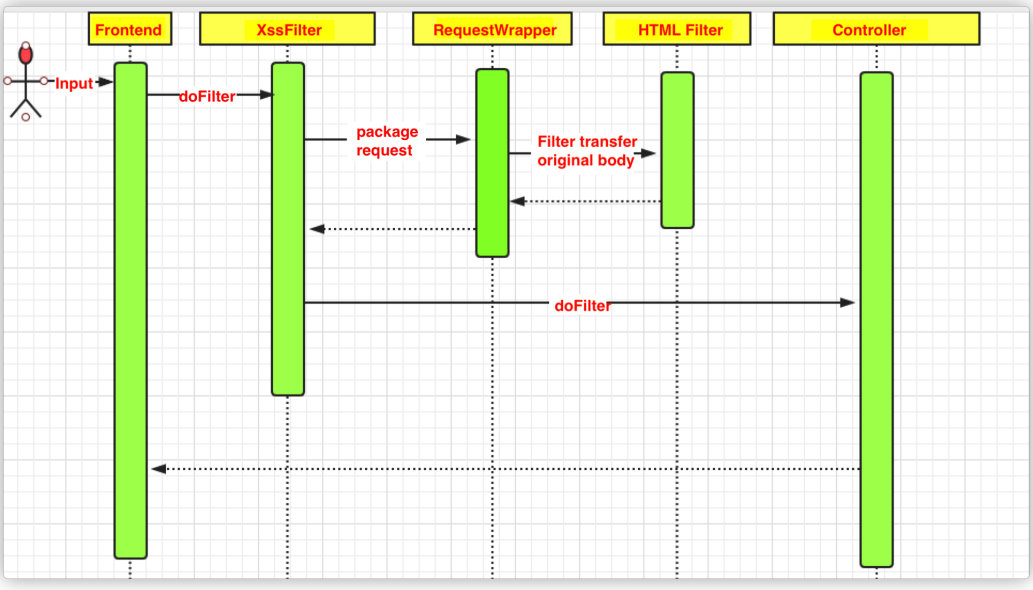


Figure 38: Process against xss.

For example, when we ignore the office xss testing API, if we want to attack the API with data input as: **data=<script>alert(1)</script>:**

```java
@CrossOrigin
@RestController
@RequestMapping("/office")
@Api(tags = "Office")
@XssCleanIgnore
public class OfficeController {

    @Autowired
    private OfficeService officeService;

    @RequestMapping(value = "/list", method = RequestMethod.GET)
    public List<OfficeEntry> findOfficeList(){
        return officeService.findAllOffice();
    }

    @RequestMapping(value = "/carlist", method = RequestMethod.GET)
    public List<CarEntry> findCarList(@RequestParam(value = "officeid") int officeId){
        return officeService.findCarList(officeId);
    }

    @GetMapping("/xss")
    public String xssGet(String data){
        System.out.println(data);
        return data;
    }

    @PostMapping("/xss")
    public String xssPost(String data){
        System.out.println(data);
        return data;
    }

}
```

Figure 39: Xss example.

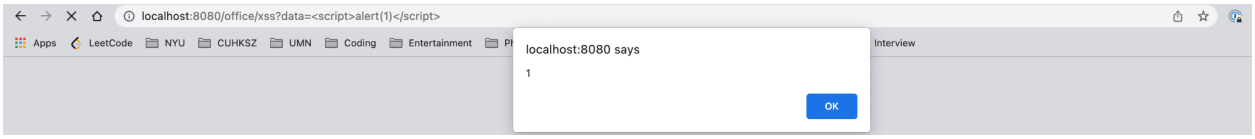


Figure 40: Results of being attacked.

We could see that the attack would be successful since we turned off the XSS on the /office APIs.

When when turn it on, We could see that the input would be filtered from Figure 41. By using this package, our project would prevent XSS attacks.

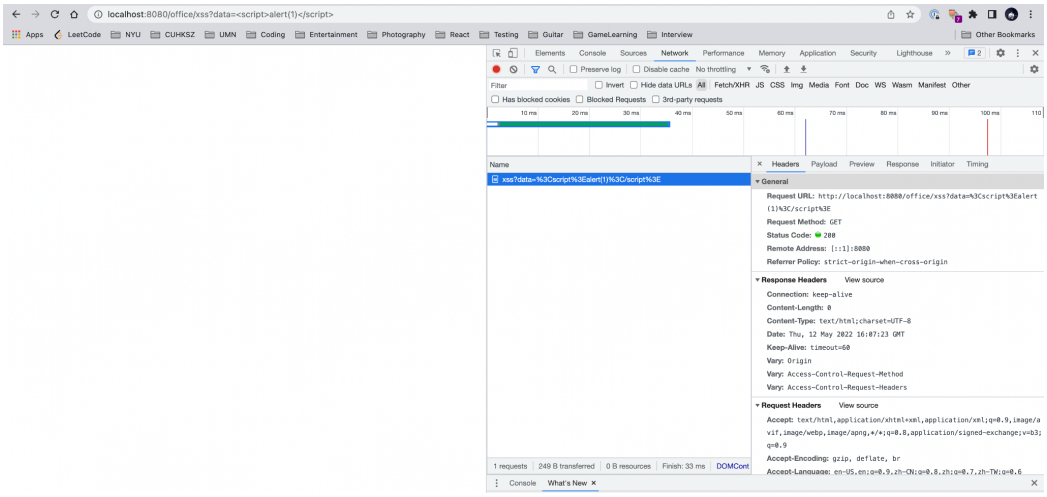


Figure 41: Successfully filtering the xss attack.

5.3. Authentication flow

The authentication flow and the classification of users in our demo are given in Figure 21. There are three types of roles for a user: guest, customer and administrator. The information of the customer and administrator is stored in our database.

When an unauthenticated user tries to sign in, the backend will send back responses with Set-Cookie header if the user uses a valid account. Every request the user makes in the following will carry the cookie so that the server can recognize the user.

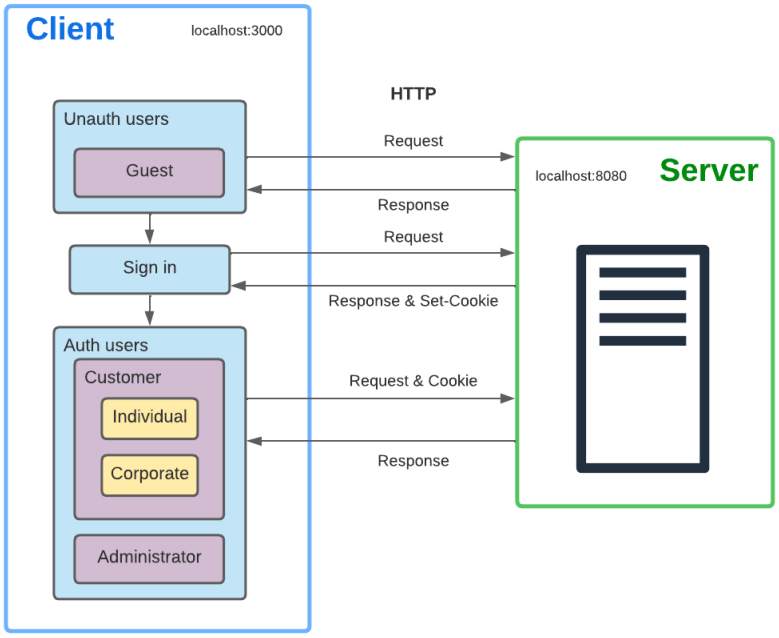


Figure 42: Authentication flow.

6. Business Analysis

6.1. Table joins with at least 3 tables in join

```
select c.CustId, c.CustType, c.Email, c.Phone, c.AccId, c.AccName, d.LName, d.FName
from (
    select a.CustId, a.CustType, a.Email, a.Phone, b.accid, b.accname
    from zzz_customer a join zzz_account b
    on a.custid=b.custid
) c
join zzz_individual d on c.CustId=d.CustId;
```

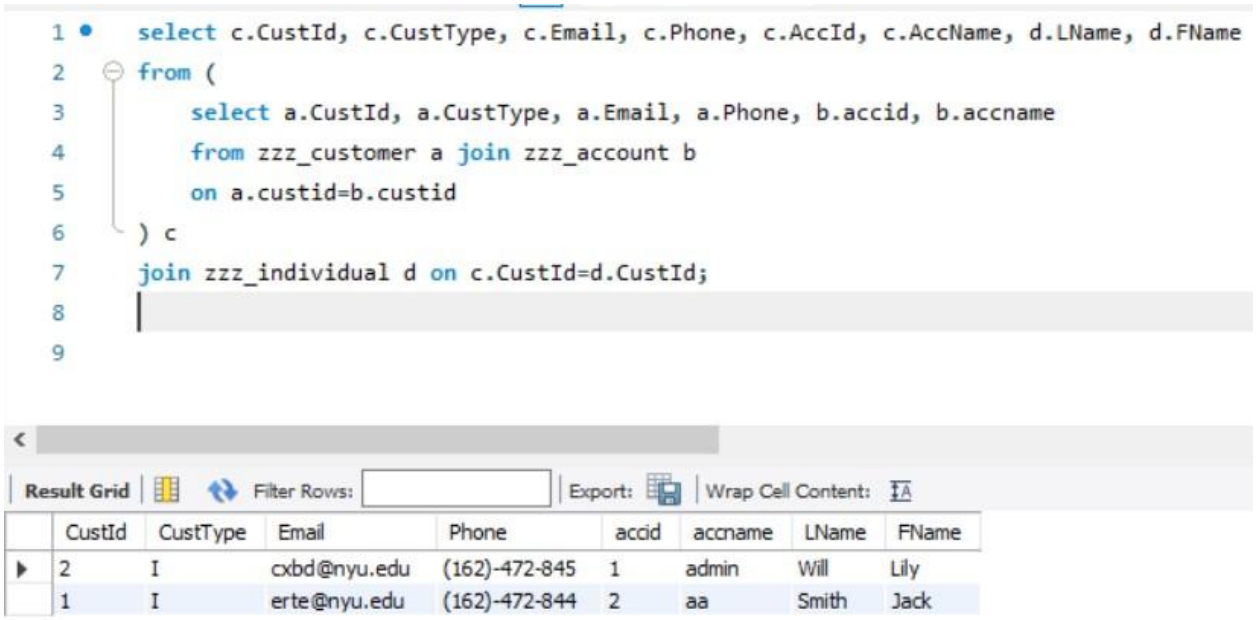


Figure 43: Solution for Q1.

This query is used to ask for showing all individual customers information about their full name, phone number, email address, account number, account name and account ID.

6.2. Multi-row subquery

```
select invoiceid, amount, invoicedate, ispaid
from zzz_invoice
where amount>= all (select avg(amount) from zzz_invoice group by orderid);
```

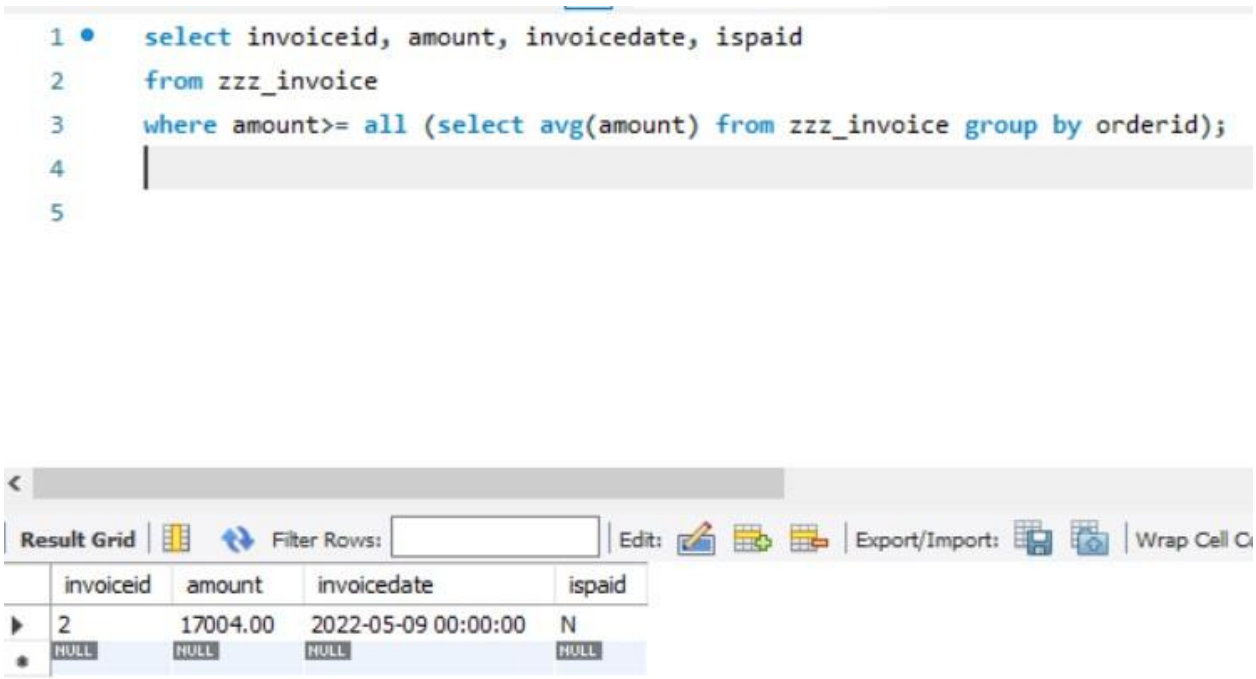
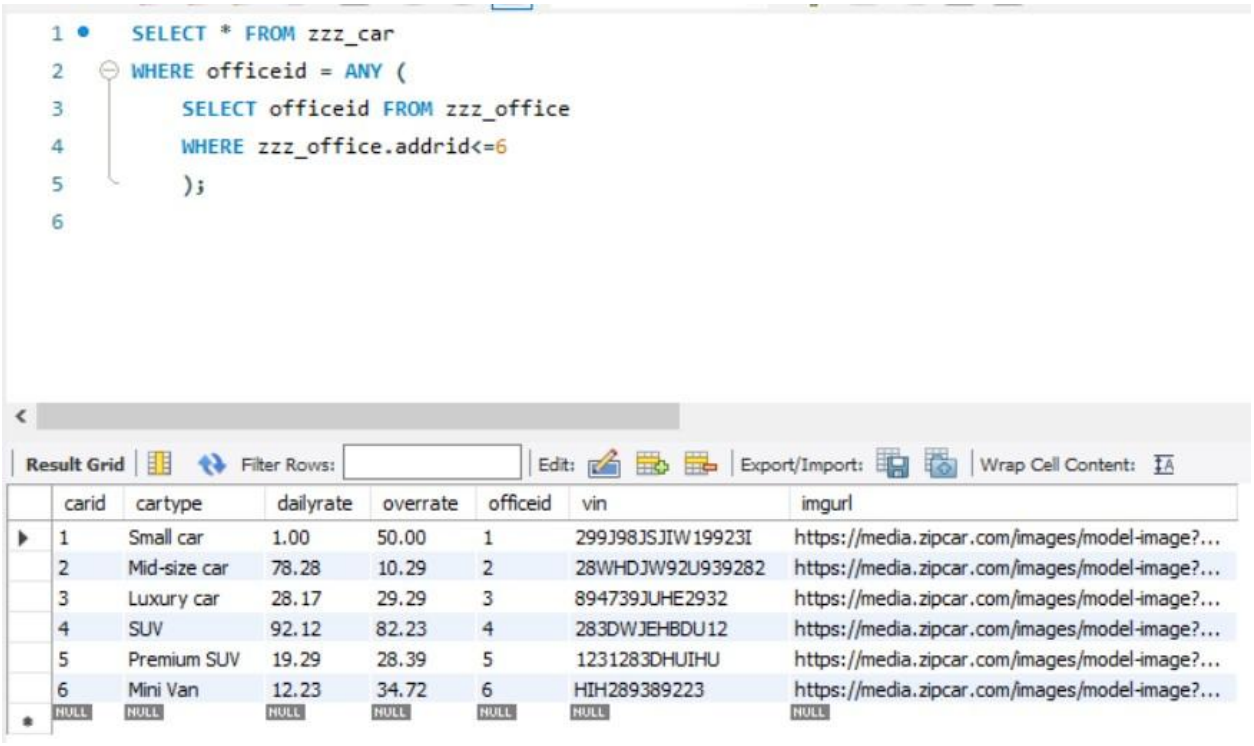


Figure 44: Solution for Q2.

This query is used to show the invoices whose amount is greater or equal than the highest average invoice amount of an order.

6.3. Correlated subquery

```
SELECT * FROM zzz_car
WHERE officeid = ANY (
       SELECT officeid FROM zzz_office
       WHERE zzz_office.addrid<=6
    );
```



```
1  SELECT * FROM zzz_car
2  WHERE officeid = ANY (
3      SELECT officeid FROM zzz_office
4      WHERE zzz_office.addrid<=6
5      );
6
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

carid	cartype	dailyrate	overrate	officeid	vin	imgurl
1	Small car	1.00	50.00	1	299J98JSJIW19923I	https://media.zipcar.com/images/model-image?...
2	Mid-size car	78.28	10.29	2	28WHDJW92U939282	https://media.zipcar.com/images/model-image?...
3	Luxury car	28.17	29.29	3	894739JUHE2932	https://media.zipcar.com/images/model-image?...
4	SUV	92.12	82.23	4	283DWJEHBDU12	https://media.zipcar.com/images/model-image?...
5	Premium SUV	19.29	28.39	5	1231283DHUIHU	https://media.zipcar.com/images/model-image?...
6	Mini Van	12.23	34.72	6	HIH289389223	https://media.zipcar.com/images/model-image?...
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Figure 45: Solution for Q3.

This correlated query is used to list car information for which car's office address id is smaller than 6.

6.4. SET operator query

```
select * from zzz_customer
where custid= '1'
union
select * from zzz_customer
where addrid= '13'
order by custid;
```

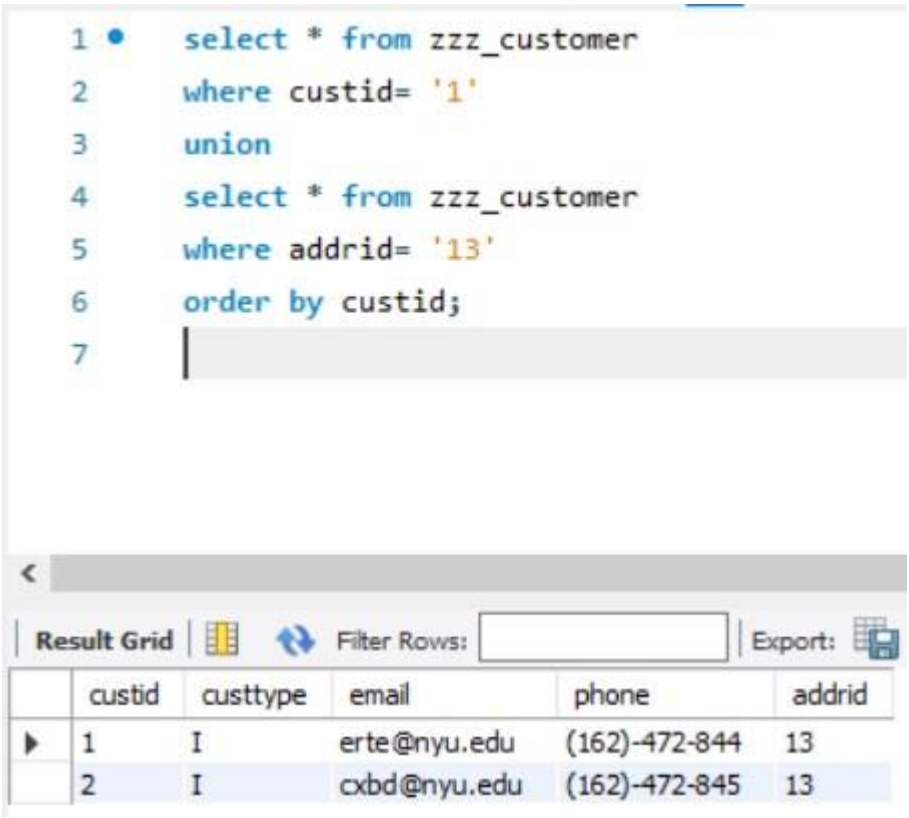


Figure 46: Solution for Q4.

This union query is used to show the customer information in which customer id equals to 1 or the address id equals to 13.

6.5. Query with in line view or WITH clause

```
SELECT * FROM(
SELECT orderid,startodo,endodo,startdate,enddate,custid,custtype FROM zzz_order
WHERE endodo>= 1100) a
WHERE a.endodo <= 1500;
```

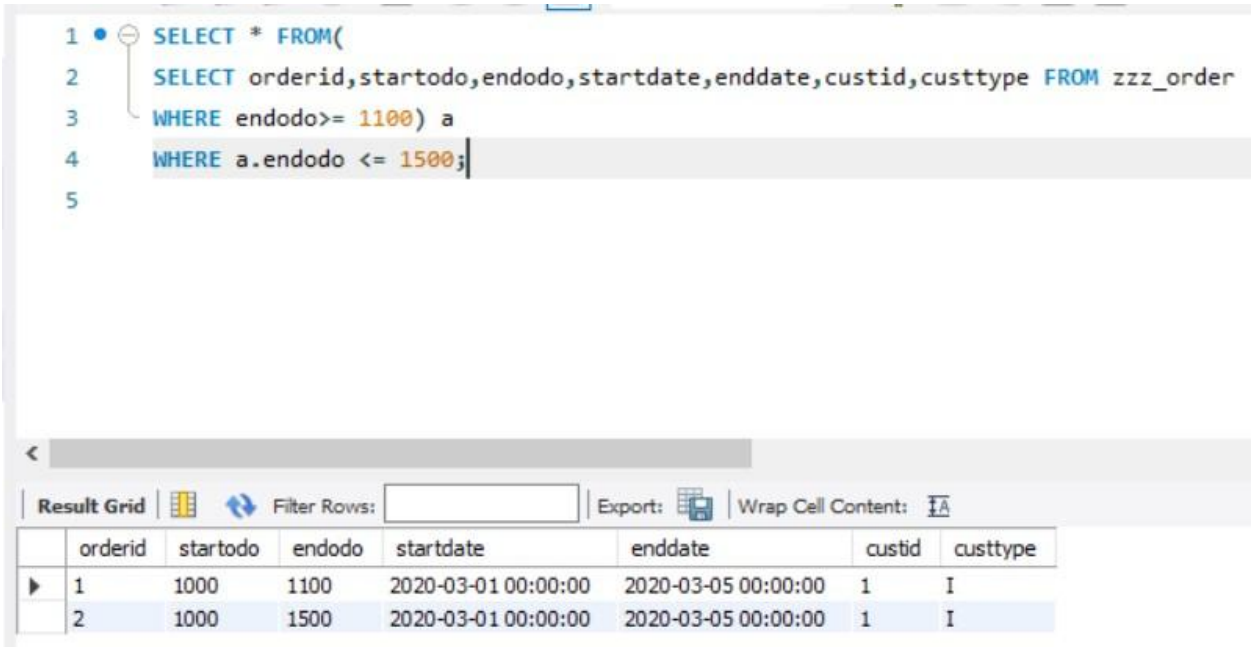


Figure 47: Solution for Q5.

This in-line query is used to show the information about the order in which the end odometer is greater than 1100 but smaller than 1500.

6.6. TOP-N query

```
select * from zzz_discount
order by discpercen asc limit 6;
```

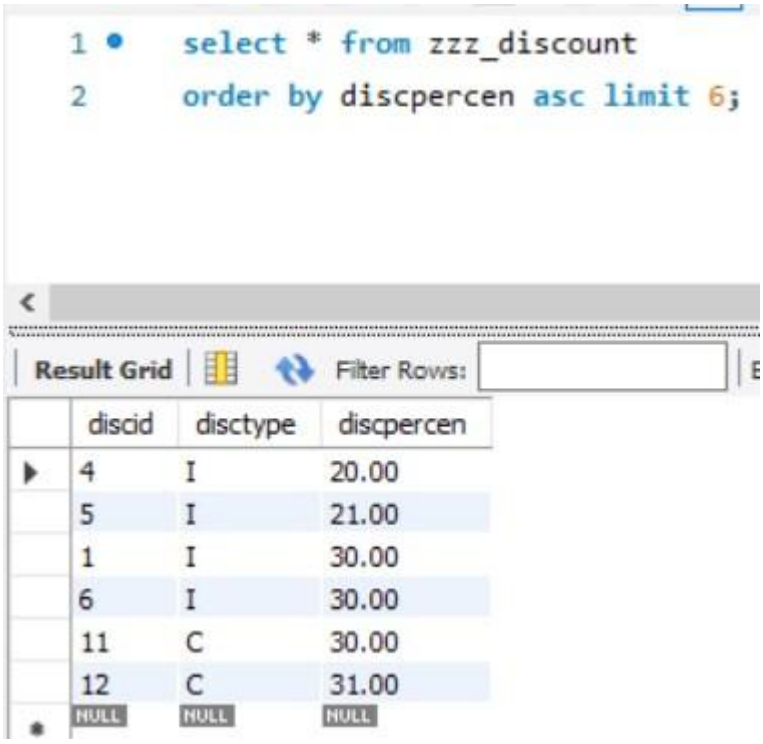


Figure 48: Solution for Q6.

This query is used to list discount information which has the top-6 minimum discount percent.

7. Summary and Reflection

7.1. Summary

For a car rental company, which contains a lot of customers, the most important business is to provide a convenient website for different types of customers to process transactions simply and efficiently along with recording transactions and invoices. This is what we try to resolve in this project. Finally, the WOW car rental website application is coming up and used for addressing different business scenarios.

WOW web enables customers to register their own accounts and log in online to operate their business. We invent two different customers, one is for individuals and the other is for corporations. Both of them should register on our website but on two different register pages. Different customers can hold different discount types, these discounts can be applied in the payment stage. Customers can place orders depending on their needs(Time, Location, Type of vehicle). Once customers place their order, then order information will be updated to the order table. Only the administrator can decide if the order is ending or not, by updating the end odometer for each order. The payment page will add the new payment info after the order ends. Then customers can pay the bill at this page by installment or full payment. After each payment, our database trigger will automatically generate the invoice list according to each payment. Meanwhile, we will show the invoice list on the payment page. Customers can check it out and learn about it if they have finished the order.

By following this logical functionality, WOW car rental website will run properly. The concise user interface, simple operation of account and secure payment method will attract bunches of customers to use our website.

7.2. Reflection

One thing we learned from this project is that data management is not as simple as it seems. It takes a lot of time and effort to manage data in the right way. For example, database design. We have to consider how to design the data model (Logical model and relational model), and how to apply this database with business analysis before we start programming. This is the most basic and significant part of implementing a Database-Web application. The process of organizing data into a logical structure can help us to store, manage, and analyze data efficiently. It is the critical part of our project.

Another lesson we learned is how significant security is for web-based applications. Such as the protection of services from unauthorized access, the prevention of SQL injection, and the resistance to XSS attacks. All of these security features will bring a huge impact on preventing vulnerabilities and building a safe website. These security issues must be considered in the development of a good and complete web application.

In the end, we want to emphasize that this project has been a fascinating experience and also a challenge for all of us. We learned how to build a web application from scratch, and did lots of research on reasonably connecting the front-end and the back-end. Thanks to our team

members' skill sets in various areas and work experience in the industry. This project was able to go smoothly. Cooperation is always the key to success in the business rule. We spent a lot of time discussing on Zoom, which was quite efficient than expected. Every team member took the initiative to share their perspective of view on how to resolve an issue. It is helpful to not only make great progress on the project but also to improve our own technical skills.

8. Appendix

8.1. DDL Codes (Oracle Data Modeler)

```
-- Generated by Oracle SQL Developer Data Modeler 21.4.1.349.1605
--   at:        2022-05-09 15:45:23 EDT
--   site:      Oracle Database 21c
--   type:      Oracle Database 21c



-- predefined type, no DDL - MDSYS.SDO_GEOMETRY

-- predefined type, no DDL - XMLTYPE

CREATE TABLE zzz_acc_role (
   accid NUMBER(10) NOT NULL,
   rid   NUMBER(10) NOT NULL
);

ALTER TABLE zzz_acc_role ADD CONSTRAINT zzz_acc_role_pk PRIMARY KEY ( rid,
                                                                      accid );

CREATE TABLE zzz_account (
   accid    NUMBER(10) NOT NULL,
   accname  VARCHAR2(20) NOT NULL,
   pwd      VARCHAR2(20) NOT NULL,
   custid   NUMBER(6) NOT NULL,
   custtype CHAR(1) NOT NULL
);

COMMENT ON COLUMN zzz_account.accid IS
   'Unique ID for account.';

COMMENT ON COLUMN zzz_account.accname IS
   'Account name.';

COMMENT ON COLUMN zzz_account.pwd IS
   'Password.';

ALTER TABLE zzz_account ADD CONSTRAINT zzz_account_pk PRIMARY KEY ( accid );

CREATE TABLE zzz_address (
   addrid  NUMBER(10) NOT NULL,
```

```
   street  VARCHAR2(30) NOT NULL,
   state   VARCHAR2(30) NOT NULL,
   country VARCHAR2(30) NOT NULL,
   zipcode NUMBER(5) NOT NULL
);

COMMENT ON COLUMN zzz_address.addrid IS
   'Unique ID for address. ';

COMMENT ON COLUMN zzz_address.street IS
   'Street info for address. ';

COMMENT ON COLUMN zzz_address.state IS
   'State info for address. ';

COMMENT ON COLUMN zzz_address.country IS
   'Country info for address. ';

COMMENT ON COLUMN zzz_address.zipcode IS
   'Zipcode for address. ';

ALTER TABLE zzz_address ADD CONSTRAINT zzz_address_pk PRIMARY KEY ( addrid );

CREATE TABLE zzz_car (
   carid     NUMBER(6) NOT NULL,
   cartype   VARCHAR2(20) NOT NULL,
   dailyrate NUMBER(5, 2) NOT NULL,
   overrate  NUMBER(5, 2) NOT NULL,
   imgurl    VARCHAR2(80),
   officeid  NUMBER(6),
   vin       VARCHAR2(20) NOT NULL
);

COMMENT ON COLUMN zzz_car.carid IS
   'Unique ID for each car. ';

COMMENT ON COLUMN zzz_car.cartype IS
   'Class of the car.';

COMMENT ON COLUMN zzz_car.dailyrate IS
   'Regular rental rate per day of the rental service for the car.';

COMMENT ON COLUMN zzz_car.overrate IS
   'Extra fees per mile that exceeds the limit.';
```

```
COMMENT ON COLUMN zzz_car.imgurl IS
   'Url for Images of car.';

CREATE UNIQUE INDEX zzz_car__idx ON
   zzz_car (
       vin
   ASC );

ALTER TABLE zzz_car ADD CONSTRAINT zzz_car_pk PRIMARY KEY ( carid );

CREATE TABLE zzz_corporate (
   custid   NUMBER(6) NOT NULL,
   custtype CHAR(1) NOT NULL,
   corpname VARCHAR2(20) NOT NULL,
   regnum   VARCHAR2(20) NOT NULL,
   empid    VARCHAR2(20) NOT NULL
);

COMMENT ON COLUMN zzz_corporate.custid IS
   'Unique ID for customer.';

COMMENT ON COLUMN zzz_corporate.custtype IS
   'Customer type.';

COMMENT ON COLUMN zzz_corporate.corpname IS
   'Corporation''s name.';

COMMENT ON COLUMN zzz_corporate.regnum IS
   'Registration number of the corporation.';

COMMENT ON COLUMN zzz_corporate.empid IS
   'Employee ID of the customer who rents the car on a corporate account.';

ALTER TABLE zzz_corporate ADD CONSTRAINT zzz_corporate_pk PRIMARY KEY ( custid,
                                                                        custtype
);

CREATE TABLE zzz_customer (
   custid   NUMBER(6) NOT NULL,
   custtype CHAR(1) NOT NULL,
   email    VARCHAR2(30) NOT NULL,
   phone    VARCHAR2(20) NOT NULL,
   addrid   NUMBER(10) NOT NULL
```

```
);

ALTER TABLE zzz_customer
   ADD CONSTRAINT ch_inh_zzz_customer CHECK ( custtype IN ( 'C', 'I' ) );

COMMENT ON COLUMN zzz_customer.custid IS
   'Unique ID for customer.';

COMMENT ON COLUMN zzz_customer.custtype IS
   'Customer type.';

COMMENT ON COLUMN zzz_customer.email IS
   'Email address for customer. ';

COMMENT ON COLUMN zzz_customer.phone IS
   'Phone number for customers. ';

ALTER TABLE zzz_customer ADD CONSTRAINT zzz_customer_pk PRIMARY KEY ( custid,
                                                                      custtype
);

CREATE TABLE zzz_disccorp (
   discid   NUMBER(10) NOT NULL,
   disctype CHAR(1) NOT NULL,
   setnum   NUMBER(20) NOT NULL
);

COMMENT ON COLUMN zzz_disccorp.discid IS
   'Uinque ID for discount.';

COMMENT ON COLUMN zzz_disccorp.disctype IS
   'Discriminator of discount type.';

COMMENT ON COLUMN zzz_disccorp.setnum IS
   'Number for indentifying corporation';

ALTER TABLE zzz_disccorp ADD CONSTRAINT zzz_disccorp_pk PRIMARY KEY ( discid,
                                                                      disctype
);

CREATE TABLE zzz_discindi (
   discid     NUMBER(10) NOT NULL,
   disctype   CHAR(1) NOT NULL,
   coupnum    VARCHAR2(10) NOT NULL,
```

```
   validstart DATE NOT NULL,
   validend   DATE NOT NULL
);

COMMENT ON COLUMN zzz_discindi.discid IS
   'Uinque ID for discount.';

COMMENT ON COLUMN zzz_discindi.disctype IS
   'Discriminator of discount type.';

COMMENT ON COLUMN zzz_discindi.coupnum IS
   'Coupon number.';

COMMENT ON COLUMN zzz_discindi.validstart IS
   'Coupon valid start date.';

COMMENT ON COLUMN zzz_discindi.validend IS
   'Coupon valid end date.';

ALTER TABLE zzz_discindi ADD CONSTRAINT zzz_discindi_pk PRIMARY KEY ( discid,
                                                                      disctype
);

CREATE TABLE zzz_discount (
   discid     NUMBER(10) NOT NULL,
   disctype   CHAR(1) NOT NULL,
   discpercen NUMBER(5, 2) NOT NULL
);

ALTER TABLE zzz_discount
   ADD CONSTRAINT ch_inh_zzz_discount CHECK ( disctype IN ( 'C', 'I' ) );

COMMENT ON COLUMN zzz_discount.discid IS
   'Uinque ID for discount.';

COMMENT ON COLUMN zzz_discount.disctype IS
   'Discriminator of discount type.';

COMMENT ON COLUMN zzz_discount.discpercen IS
   'Discount percentage %.';

ALTER TABLE zzz_discount ADD CONSTRAINT zzz_discount_pk PRIMARY KEY ( discid,
                                                                      disctype
);
```

```
CREATE TABLE zzz_individual (
   custid     NUMBER(6) NOT NULL,
   custtype   CHAR(1) NOT NULL,
   lname      VARCHAR2(20) NOT NULL,
   fname      VARCHAR2(20) NOT NULL,
   licensenum NUMBER(15) NOT NULL,
   insname    VARCHAR2(20) NOT NULL,
   insnum     NUMBER(20) NOT NULL
);

COMMENT ON COLUMN zzz_individual.custid IS
   'Unique ID for customer.';

COMMENT ON COLUMN zzz_individual.custtype IS
   'Customer type.';

COMMENT ON COLUMN zzz_individual.lname IS
   'Last name for individual customer. ';

COMMENT ON COLUMN zzz_individual.fname IS
   'First Name for individual customer. ';

COMMENT ON COLUMN zzz_individual.licensenum IS
   'Drive license number.';

COMMENT ON COLUMN zzz_individual.insname IS
   'Insurance company name.';

COMMENT ON COLUMN zzz_individual.insnum IS
   'Insurance policy number.';

ALTER TABLE zzz_individual ADD CONSTRAINT zzz_individual_pk PRIMARY KEY (
custid,

custtype );

CREATE TABLE zzz_invoice (
   invoiceid   NUMBER(10) NOT NULL,
   invoicedate DATE NOT NULL,
   amount      NUMBER(10, 2) NOT NULL,
   remain      NUMBER(10, 2) NOT NULL,
   ispaid      CHAR(1) NOT NULL,
   orderid     NUMBER(10) NOT NULL
```

```
);

COMMENT ON COLUMN zzz_invoice.invoiceid IS
   'Unique ID for invoice.';

COMMENT ON COLUMN zzz_invoice.invoicedate IS
   'Invoice data.';

COMMENT ON COLUMN zzz_invoice.amount IS
   'Invoice amount.';

COMMENT ON COLUMN zzz_invoice.remain IS
   'Amount remained to pay.';

COMMENT ON COLUMN zzz_invoice.ispaid IS
   'Boolean to check if the invoice is paid.';

CREATE UNIQUE INDEX zzz_invoice__idx ON
   zzz_invoice (
       orderid
   ASC );

ALTER TABLE zzz_invoice ADD CONSTRAINT zzz_invoice_pk PRIMARY KEY ( invoiceid
);

CREATE TABLE zzz_office (
   officeid NUMBER(6) NOT NULL,
   phone    VARCHAR2(20) NOT NULL,
   addrid   NUMBER(10) NOT NULL
);

COMMENT ON COLUMN zzz_office.officeid IS
   'Unique ID for office.';

COMMENT ON COLUMN zzz_office.phone IS
   'Office''s phone number.';

CREATE UNIQUE INDEX zzz_office__idx ON
   zzz_office (
       addrid
   ASC );

ALTER TABLE zzz_office ADD CONSTRAINT zzz_office_pk PRIMARY KEY ( officeid );
```

```
CREATE TABLE zzz_order (
   orderid   NUMBER(10) NOT NULL,
   startodo  NUMBER(10) NOT NULL,
   endodo    NUMBER(10),
   odolimit  NUMBER(10),
   startdate DATE NOT NULL,
   enddate   DATE NOT NULL,
   custid    NUMBER(6) NOT NULL,
   custtype  CHAR(1) NOT NULL,
   pickup    NUMBER(10) NOT NULL,
   dropoff   NUMBER(10) NOT NULL,
   carid     NUMBER(6) NOT NULL,
   discid    NUMBER(10),
   disctype  CHAR(1)
);

COMMENT ON COLUMN zzz_order.orderid IS
   'Unique ID for each order.';

COMMENT ON COLUMN zzz_order.startodo IS
   'Start odometer.';

COMMENT ON COLUMN zzz_order.endodo IS
   'End Odometer.';

COMMENT ON COLUMN zzz_order.odolimit IS
   'Daily odometer limit for the rental service.';

COMMENT ON COLUMN zzz_order.startdate IS
   'Date when the customer starts the service.';

COMMENT ON COLUMN zzz_order.enddate IS
   'Date when the customer ends the service.';

CREATE UNIQUE INDEX zzz_order__idx ON
   zzz_order (
       discid
   ASC,
       disctype
   ASC );

ALTER TABLE zzz_order ADD CONSTRAINT zzz_order_pk PRIMARY KEY ( orderid );

CREATE TABLE zzz_payment (
```

```
   paymid    NUMBER(10) NOT NULL,
   method    VARCHAR2(20) NOT NULL,
   amount    NUMBER(5, 2) NOT NULL,
   paymdate  DATE NOT NULL,
   cardnum   NUMBER(20) NOT NULL,
   invoiceid NUMBER(10) NOT NULL
);

COMMENT ON COLUMN zzz_payment.paymid IS
   'Unique ID for payment.';

COMMENT ON COLUMN zzz_payment.method IS
   'Payment method.';

COMMENT ON COLUMN zzz_payment.amount IS
   'Payment amount.';

COMMENT ON COLUMN zzz_payment.paymdate IS
   'Payment date.';

COMMENT ON COLUMN zzz_payment.cardnum IS
   'Card number.';

ALTER TABLE zzz_payment ADD CONSTRAINT zzz_payment_pk PRIMARY KEY ( paymid );

CREATE TABLE zzz_role (
   rid  NUMBER(10) NOT NULL,
   role VARCHAR2(5) NOT NULL
);

COMMENT ON COLUMN zzz_role.rid IS
   'Unique ID for role.';

COMMENT ON COLUMN zzz_role.role IS
   'Role name.';

ALTER TABLE zzz_role ADD CONSTRAINT zzz_role_pk PRIMARY KEY ( rid );

CREATE TABLE zzz_vehicle (
   vin   VARCHAR2(20) NOT NULL,
   make  VARCHAR2(30) NOT NULL,
   model VARCHAR2(30) NOT NULL,
   year  NUMBER(4) NOT NULL,
   lpn   VARCHAR2(15) NOT NULL
```

```
);

COMMENT ON COLUMN zzz_vehicle.vin IS
   'Vehicle identification number.';

COMMENT ON COLUMN zzz_vehicle.make IS
   'Brand of the vehicle.';

COMMENT ON COLUMN zzz_vehicle.model IS
   'Name of a product or a range of products.';

COMMENT ON COLUMN zzz_vehicle.year IS
   'Manufacture year of the vehicle. ';

COMMENT ON COLUMN zzz_vehicle.lpn IS
   'The registration identifier is a numeric or alphanumeric ID that uniquely
identifies the vehicle or vehicle owner within the issuing region''s vehicle
register.';

ALTER TABLE zzz_vehicle ADD CONSTRAINT zzz_vehicle_pk PRIMARY KEY ( vin );

ALTER TABLE zzz_acc_role
   ADD CONSTRAINT zzz_acc_role_zzz_account_fk FOREIGN KEY ( accid )
       REFERENCES zzz_account ( accid );

ALTER TABLE zzz_acc_role
   ADD CONSTRAINT zzz_acc_role_zzz_role_fk FOREIGN KEY ( rid )
       REFERENCES zzz_role ( rid );

ALTER TABLE zzz_account
   ADD CONSTRAINT zzz_account_zzz_customer_fk FOREIGN KEY ( custid,
                                                            custtype )
       REFERENCES zzz_customer ( custid,
                                 custtype );

ALTER TABLE zzz_car
   ADD CONSTRAINT zzz_car_zzz_office_fk FOREIGN KEY ( officeid )
       REFERENCES zzz_office ( officeid );

ALTER TABLE zzz_car
   ADD CONSTRAINT zzz_car_zzz_vehicle_fk FOREIGN KEY ( vin )
       REFERENCES zzz_vehicle ( vin );

ALTER TABLE zzz_corporate
```

```
   ADD CONSTRAINT zzz_corporate_zzz_customer_fk FOREIGN KEY ( custid,
                                                              custtype )
       REFERENCES zzz_customer ( custid,
                                 custtype );

ALTER TABLE zzz_customer
   ADD CONSTRAINT zzz_customer_zzz_address_fk FOREIGN KEY ( addrid )
       REFERENCES zzz_address ( addrid );

ALTER TABLE zzz_disccorp
   ADD CONSTRAINT zzz_disccorp_zzz_discount_fk FOREIGN KEY ( discid,
                                                             disctype )
       REFERENCES zzz_discount ( discid,
                                 disctype );

ALTER TABLE zzz_discindi
   ADD CONSTRAINT zzz_discindi_zzz_discount_fk FOREIGN KEY ( discid,
                                                             disctype )
       REFERENCES zzz_discount ( discid,
                                 disctype );

ALTER TABLE zzz_individual
   ADD CONSTRAINT zzz_individual_zzz_customer_fk FOREIGN KEY ( custid,
                                                               custtype )
       REFERENCES zzz_customer ( custid,
                                 custtype );

ALTER TABLE zzz_invoice
   ADD CONSTRAINT zzz_invoice_zzz_order_fk FOREIGN KEY ( orderid )
       REFERENCES zzz_order ( orderid );

ALTER TABLE zzz_office
   ADD CONSTRAINT zzz_office_zzz_address_fk FOREIGN KEY ( addrid )
       REFERENCES zzz_address ( addrid );

ALTER TABLE zzz_order
   ADD CONSTRAINT zzz_order_zzz_address_fk FOREIGN KEY ( dropoff )
       REFERENCES zzz_address ( addrid );

ALTER TABLE zzz_order
   ADD CONSTRAINT zzz_order_zzz_address_fkv2 FOREIGN KEY ( pickup )
       REFERENCES zzz_address ( addrid );

ALTER TABLE zzz_order
```

```
   ADD CONSTRAINT zzz_order_zzz_car_fk FOREIGN KEY ( carid )
       REFERENCES zzz_car ( carid );

ALTER TABLE zzz_order
   ADD CONSTRAINT zzz_order_zzz_customer_fk FOREIGN KEY ( custid,
                                                          custtype )
       REFERENCES zzz_customer ( custid,
                                 custtype );

ALTER TABLE zzz_order
   ADD CONSTRAINT zzz_order_zzz_discount_fk FOREIGN KEY ( discid,
                                                          disctype )
       REFERENCES zzz_discount ( discid,
                                 disctype );

ALTER TABLE zzz_payment
   ADD CONSTRAINT zzz_payment_zzz_invoice_fk FOREIGN KEY ( invoiceid )
       REFERENCES zzz_invoice ( invoiceid );

CREATE OR REPLACE TRIGGER arc_fkarc_12_zzz_disccorp BEFORE
   INSERT OR UPDATE OF discid, disctype ON zzz_disccorp
   FOR EACH ROW
DECLARE
   d CHAR(1);
BEGIN
   SELECT
       a.disctype
   INTO d
   FROM
       zzz_discount a
   WHERE
       a.discid = :new.discid
       AND a.disctype = :new.disctype;

   IF ( d IS NULL OR d <> 'C' ) THEN
       raise_application_error(
                             -20223,
                             'FK ZZZ_DISCCORP_ZZZ_DISCOUNT_FK in Table
ZZZ_DISCCORP violates Arc constraint on Table ZZZ_DISCOUNT - discriminator
column DiscType doesn''t have value ''C'''
       );
   END IF;

EXCEPTION
```

```
   WHEN no_data_found THEN
       NULL;
   WHEN OTHERS THEN
       RAISE;
END;
/

CREATE OR REPLACE TRIGGER arc_fkarc_12_zzz_discindi BEFORE
   INSERT OR UPDATE OF discid, disctype ON zzz_discindi
   FOR EACH ROW
DECLARE
   d CHAR(1);
BEGIN
   SELECT
       a.disctype
   INTO d
   FROM
       zzz_discount a
   WHERE
       a.discid = :new.discid
       AND a.disctype = :new.disctype;

   IF ( d IS NULL OR d <> 'I' ) THEN
       raise_application_error(
                               -20223,
                               'FK ZZZ_DISCINDI_ZZZ_DISCOUNT_FK in Table
ZZZ_DISCINDI violates Arc constraint on Table ZZZ_DISCOUNT - discriminator
column DiscType doesn''t have value ''I'''
       );
   END IF;

EXCEPTION
   WHEN no_data_found THEN
       NULL;
   WHEN OTHERS THEN
       RAISE;
END;
/

CREATE OR REPLACE TRIGGER arc_fkarc_11_zzz_individual BEFORE
   INSERT OR UPDATE OF custid, custtype ON zzz_individual
   FOR EACH ROW
DECLARE
   d CHAR(1);
```

```
BEGIN
   SELECT
       a.custtype
   INTO d
   FROM
       zzz_customer a
   WHERE
       a.custid = :new.custid
       AND a.custtype = :new.custtype;

   IF ( d IS NULL OR d <> 'I' ) THEN
       raise_application_error(
                               -20223,
                               'FK ZZZ_INDIVIDUAL_ZZZ_CUSTOMER_FK in Table
ZZZ_INDIVIDUAL violates Arc constraint on Table ZZZ_CUSTOMER - discriminator
column CustType doesn''t have value ''I'''
       );
   END IF;

EXCEPTION
   WHEN no_data_found THEN
       NULL;
   WHEN OTHERS THEN
       RAISE;
END;
/

CREATE OR REPLACE TRIGGER arc_fkarc_11_zzz_corporate BEFORE
   INSERT OR UPDATE OF custid, custtype ON zzz_corporate
   FOR EACH ROW
DECLARE
   d CHAR(1);
BEGIN
   SELECT
       a.custtype
   INTO d
   FROM
       zzz_customer a
   WHERE
       a.custid = :new.custid
       AND a.custtype = :new.custtype;

   IF ( d IS NULL OR d <> 'C' ) THEN
       raise_application_error(
```

```
                              -20223,
                              'FK ZZZ_CORPORATE_ZZZ_CUSTOMER_FK in Table
ZZZ_CORPORATE violates Arc constraint on Table ZZZ_CUSTOMER - discriminator
column CustType doesn''t have value ''C'''
       );
   END IF;

EXCEPTION
   WHEN no_data_found THEN
       NULL;
   WHEN OTHERS THEN
       RAISE;
END;
/



-- Oracle SQL Developer Data Modeler Summary Report:
--
-- CREATE TABLE                            16
-- CREATE INDEX                             4
-- ALTER TABLE                             36
-- CREATE VIEW                              0
-- ALTER VIEW                               0
-- CREATE PACKAGE                           0
-- CREATE PACKAGE BODY                      0
-- CREATE PROCEDURE                         0
-- CREATE FUNCTION                          0
-- CREATE TRIGGER                           4
-- ALTER TRIGGER                            0
-- CREATE COLLECTION TYPE                   0
-- CREATE STRUCTURED TYPE                   0
-- CREATE STRUCTURED TYPE BODY              0
-- CREATE CLUSTER                           0
-- CREATE CONTEXT                           0
-- CREATE DATABASE                          0
-- CREATE DIMENSION                         0
-- CREATE DIRECTORY                         0
-- CREATE DISK GROUP                        0
-- CREATE ROLE                              0
-- CREATE ROLLBACK SEGMENT                  0
-- CREATE SEQUENCE                          0
-- CREATE MATERIALIZED VIEW                 0
-- CREATE MATERIALIZED VIEW LOG             0
```

```
-- CREATE SYNONYM                          0
-- CREATE TABLESPACE                       0
-- CREATE USER                             0
--
-- DROP TABLESPACE                         0
-- DROP DATABASE                           0
--
-- REDACTION POLICY                        0
--
-- ORDS DROP SCHEMA                        0
-- ORDS ENABLE SCHEMA                      0
-- ORDS ENABLE OBJECT                      0
--
-- ERRORS                                  0
-- WARNINGS                                0
```

8.2. DDL Codes (MySQL)

```
-- Table address
CREATE TABLE zzz_address (
   addrid  BIGINT NOT NULL COMMENT 'Unique ID for address.',
   street  VARCHAR(30) NOT NULL COMMENT 'Street info for address.',
   state   VARCHAR(30) NOT NULL COMMENT 'State info for address.',
   country VARCHAR(30) NOT NULL COMMENT 'Country info for address.',
   zipcode INT NOT NULL COMMENT 'Zipcode for address.'
);
ALTER TABLE zzz_address ADD CONSTRAINT zzz_address_pk PRIMARY KEY ( addrid );
ALTER TABLE zzz_address MODIFY addrid BIGINT AUTO_INCREMENT;

-- Table office
CREATE TABLE zzz_office (
   officeid INT NOT NULL COMMENT 'Unique ID for office.',
   phone    VARCHAR(20) NOT NULL COMMENT 'Office''s phone number.',
   addrid   BIGINT NOT NULL
);
CREATE UNIQUE INDEX zzz_office__idx ON
   zzz_office (
       addrid
   ASC );
ALTER TABLE zzz_office ADD CONSTRAINT zzz_office_pk PRIMARY KEY ( officeid );
ALTER TABLE zzz_office MODIFY officeid INT AUTO_INCREMENT;

-- Table car
CREATE TABLE zzz_car (
```

```
   carid     INT NOT NULL COMMENT 'Unique ID for each car.',
   cartype   VARCHAR(20) NOT NULL COMMENT 'Class of the car.',
   dailyrate DECIMAL(5, 2) NOT NULL COMMENT 'Regular rental rate per day of the
rental service for the car.',
   overrate  DECIMAL(5, 2) NOT NULL COMMENT 'Extra fees per mile that exceeds
the limit.',
   officeid  INT,
   vin       VARCHAR(20) NOT NULL,
   imgurl    VARCHAR(80)
);
CREATE UNIQUE INDEX zzz_car__idx ON
   zzz_car (
       vin
   ASC );
ALTER TABLE zzz_car ADD CONSTRAINT zzz_car_pk PRIMARY KEY ( carid );
ALTER TABLE zzz_car MODIFY carid INT AUTO_INCREMENT;

-- Table vehicle
CREATE TABLE zzz_vehicle (
   vin   VARCHAR(20) NOT NULL COMMENT 'Vehicle identification number.',
   make  VARCHAR(30) NOT NULL COMMENT 'Brand of the vehicle.',
   model VARCHAR(30) NOT NULL COMMENT 'Name of a product or a range of
products.',
   year  SMALLINT NOT NULL COMMENT 'Manufacture year of the vehicle. ',
   lpn   VARCHAR(15) NOT NULL COMMENT 'The registration identifier is a numeric
or alphanumeric ID that uniquely identifies the vehicle or vehicle owner within
the issuing region''s vehicle register.'
);
ALTER TABLE zzz_vehicle ADD CONSTRAINT zzz_vehicle_pk PRIMARY KEY ( vin );

-- Table account
CREATE TABLE zzz_account (
   accid    BIGINT NOT NULL COMMENT 'Unique ID for account.',
   accname  VARCHAR(20) NOT NULL COMMENT 'Account name.',
   pwd      VARCHAR(50) NOT NULL COMMENT 'Password.',
   custid   INT NOT NULL,
   custtype CHAR(1) NOT NULL
);
ALTER TABLE zzz_account ADD CONSTRAINT zzz_account_pk PRIMARY KEY ( accid );
ALTER TABLE zzz_account MODIFY accid BIGINT AUTO_INCREMENT;

-- Table role
CREATE TABLE zzz_role (
   rid  BIGINT NOT NULL COMMENT 'Unique ID for role.',
```

```
   role VARCHAR(10) NOT NULL COMMENT 'Role name.'
);
ALTER TABLE zzz_role ADD CONSTRAINT zzz_role_pk PRIMARY KEY ( rid );

-- Table acc_role
CREATE TABLE zzz_acc_role (
   accid BIGINT NOT NULL,
   rid   BIGINT NOT NULL
);
ALTER TABLE zzz_acc_role ADD CONSTRAINT zzz_acc_role_pk PRIMARY KEY ( rid,
                                                                      accid );

-- Table customer
CREATE TABLE zzz_customer (
   custid   INT NOT NULL COMMENT 'Unique ID for customer.',
   custtype CHAR(1) NOT NULL COMMENT 'Customer type.',
   email    VARCHAR(30) NOT NULL COMMENT 'Email address for customer.',
   phone    VARCHAR(20) NOT NULL COMMENT 'Phone number for customers.',
   addrid   BIGINT NOT NULL
);
ALTER TABLE zzz_customer
   ADD CONSTRAINT ch_inh_zzz_customer CHECK ( custtype IN ( 'C', 'I' ) );
ALTER TABLE zzz_customer ADD CONSTRAINT zzz_customer_pk PRIMARY KEY ( custid,
                                                                      custtype
);
ALTER TABLE zzz_customer MODIFY custid INT AUTO_INCREMENT;

-- Table corporate
CREATE TABLE zzz_corporate (
   custid   INT NOT NULL COMMENT 'Unique ID for customer.',
   custtype CHAR(1) NOT NULL COMMENT 'Customer type.',
   corpname VARCHAR(20) NOT NULL COMMENT 'Corporation''s name.',
   regnum   VARCHAR(20) NOT NULL COMMENT 'Registration number of the
corporation.',
   empid    VARCHAR(20) NOT NULL COMMENT 'Employee ID of the customer who rents
the car on a corporate account.'
);
ALTER TABLE zzz_corporate ADD CONSTRAINT zzz_corporate_pk PRIMARY KEY ( custid,
                                                                        custtype
);

-- Table individual
CREATE TABLE zzz_individual (
   custid     INT NOT NULL COMMENT 'Unique ID for customer.',
```

```
   custtype   CHAR(1) NOT NULL COMMENT 'Customer type.',
   lname      VARCHAR(20) NOT NULL COMMENT 'Last name for individual
customer.',
   fname      VARCHAR(20) NOT NULL COMMENT 'First Name for individual
customer.',
   licensenum BIGINT NOT NULL COMMENT 'Drive license number.',
   insname    VARCHAR(20) NOT NULL COMMENT 'Insurance company name.',
   insnum     DECIMAL(20) NOT NULL COMMENT 'Insurance policy number.'
);
ALTER TABLE zzz_individual ADD CONSTRAINT zzz_individual_pk PRIMARY KEY (
custid,

custtype );

-- Table discount
CREATE TABLE zzz_discount (
   discid     BIGINT NOT NULL COMMENT 'Uinque ID for discount.',
   disctype   CHAR(1) NOT NULL COMMENT 'Discriminator of discount type.',
   discpercen DECIMAL(5, 2) NOT NULL COMMENT 'Discount percentage %.'
);
ALTER TABLE zzz_discount
   ADD CONSTRAINT ch_inh_zzz_discount CHECK ( disctype IN ( 'C', 'I' ) );
ALTER TABLE zzz_discount ADD CONSTRAINT zzz_discount_pk PRIMARY KEY ( discid,
                                                                      disctype
);
ALTER TABLE zzz_discount MODIFY discid BIGINT AUTO_INCREMENT;

-- Table disccorp
CREATE TABLE zzz_disccorp (
   discid   BIGINT NOT NULL COMMENT 'Uinque ID for discount.',
   disctype CHAR(1) NOT NULL COMMENT 'Discriminator of discount type.',
   setnum   DECIMAL(20) NOT NULL COMMENT 'Number for indentifying corporation'
);
ALTER TABLE zzz_disccorp ADD CONSTRAINT zzz_disccorp_pk PRIMARY KEY ( discid,
                                                                      disctype
);

-- Table discindi
CREATE TABLE zzz_discindi (
   discid     BIGINT NOT NULL COMMENT 'Uinque ID for discount.',
   disctype   CHAR(1) NOT NULL COMMENT 'Discriminator of discount type.',
   coupnum    VARCHAR(10) NOT NULL COMMENT 'Coupon number.',
   validstart DATETIME NOT NULL COMMENT 'Coupon valid start date.',
   validend   DATETIME NOT NULL COMMENT 'Coupon valid end date.'
```

```
);
ALTER TABLE zzz_discindi ADD CONSTRAINT zzz_discindi_pk PRIMARY KEY ( discid,
                                                                      disctype
);

-- Table order
CREATE TABLE zzz_order (
   orderid   BIGINT NOT NULL COMMENT 'Unique ID for each order.',
   startodo  BIGINT NOT NULL COMMENT 'Start odometer.',
   endodo    BIGINT COMMENT 'End Odometer.',
   odolimit  BIGINT COMMENT 'Daily odometer limit for the rental service.',
   startdate DATETIME NOT NULL COMMENT 'Date when the customer starts the
service.',
   enddate   DATETIME NOT NULL COMMENT 'Date when the customer ends the
service.',
   custid    INT NOT NULL,
   custtype  CHAR(1) NOT NULL,
   pickup    BIGINT NOT NULL,
   dropoff   BIGINT NOT NULL,
   carid     INT NOT NULL,
   disctype  CHAR(1),
   discid    BIGINT
);
CREATE UNIQUE INDEX zzz_order__idx ON
   zzz_order (
       disctype
   ASC,
       discid
   ASC );
ALTER TABLE zzz_order ADD CONSTRAINT zzz_order_pk PRIMARY KEY ( orderid );
ALTER TABLE zzz_order MODIFY orderid BIGINT AUTO_INCREMENT;

-- Table invoice
CREATE TABLE zzz_invoice (
   invoiceid   BIGINT NOT NULL COMMENT 'Unique ID for invoice.',
   invoicedate DATETIME NOT NULL COMMENT 'Invoice data.',
   amount      DECIMAL(10, 2) NOT NULL COMMENT 'Invoice amount.',
   remain      DECIMAL(10, 2) NOT NULL COMMENT 'Remain amount.',
   ispaid      CHAR(1) NOT NULL DEFAULT 'N' COMMENT 'Boolean to check if the
invoice is paid.',
   orderid     BIGINT NOT NULL
);
CREATE UNIQUE INDEX zzz_invoice__idx ON
   zzz_invoice (
```

```
        orderid
    ASC );
ALTER TABLE zzz_invoice ADD CONSTRAINT zzz_invoice_pk PRIMARY KEY ( invoiceid
);

-- Table payment
CREATE TABLE zzz_payment (
    paymid    BIGINT NOT NULL COMMENT 'Unique ID for payment.',
    method    VARCHAR(20) NOT NULL COMMENT 'Payment method.',
    amount    DECIMAL(10, 2) NOT NULL COMMENT 'Payment amount.',
    paymdate  DATETIME NOT NULL COMMENT 'Payment date.',
    cardnum   VARCHAR(50) NOT NULL COMMENT 'Card number.',
    invoiceid BIGINT NOT NULL
);
ALTER TABLE zzz_payment ADD CONSTRAINT zzz_payment_pk PRIMARY KEY ( paymid );
ALTER TABLE zzz_payment MODIFY paymid BIGINT AUTO_INCREMENT;

-- Set FK
ALTER TABLE zzz_acc_role
    ADD CONSTRAINT zzz_acc_role_zzz_account_fk FOREIGN KEY ( accid )
        REFERENCES zzz_account ( accid );

ALTER TABLE zzz_acc_role
    ADD CONSTRAINT zzz_acc_role_zzz_role_fk FOREIGN KEY ( rid )
        REFERENCES zzz_role ( rid );

ALTER TABLE zzz_account
    ADD CONSTRAINT zzz_account_zzz_customer_fk FOREIGN KEY ( custid,
                                                             custtype )
        REFERENCES zzz_customer ( custid,
                                  custtype );

ALTER TABLE zzz_car
    ADD CONSTRAINT zzz_car_zzz_office_fk FOREIGN KEY ( officeid )
        REFERENCES zzz_office ( officeid );

ALTER TABLE zzz_car
    ADD CONSTRAINT zzz_car_zzz_vehicle_fk FOREIGN KEY ( vin )
        REFERENCES zzz_vehicle ( vin );

ALTER TABLE zzz_corporate
    ADD CONSTRAINT zzz_corporate_zzz_customer_fk FOREIGN KEY ( custid,
                                                               custtype )
        REFERENCES zzz_customer ( custid,
```

```
                                custtype );

ALTER TABLE zzz_customer
   ADD CONSTRAINT zzz_customer_zzz_address_fk FOREIGN KEY ( addrid )
       REFERENCES zzz_address ( addrid );

ALTER TABLE zzz_disccorp
   ADD CONSTRAINT zzz_disccorp_zzz_discount_fk FOREIGN KEY ( discid,
                                                             disctype )
       REFERENCES zzz_discount ( discid,
                                 disctype );

ALTER TABLE zzz_discindi
   ADD CONSTRAINT zzz_discindi_zzz_discount_fk FOREIGN KEY ( discid,
                                                             disctype )
       REFERENCES zzz_discount ( discid,
                                 disctype );

ALTER TABLE zzz_individual
   ADD CONSTRAINT zzz_individual_zzz_customer_fk FOREIGN KEY ( custid,
                                                               custtype )
       REFERENCES zzz_customer ( custid,
                                 custtype );

ALTER TABLE zzz_invoice
   ADD CONSTRAINT zzz_invoice_zzz_order_fk FOREIGN KEY ( orderid )
       REFERENCES zzz_order ( orderid );

ALTER TABLE zzz_office
   ADD CONSTRAINT zzz_office_zzz_address_fk FOREIGN KEY ( addrid )
       REFERENCES zzz_address ( addrid );

ALTER TABLE zzz_order
   ADD CONSTRAINT zzz_order_zzz_address_fk FOREIGN KEY ( dropoff )
       REFERENCES zzz_address ( addrid );

ALTER TABLE zzz_order
   ADD CONSTRAINT zzz_order_zzz_address_fkv2 FOREIGN KEY ( pickup )
       REFERENCES zzz_address ( addrid );

ALTER TABLE zzz_order
   ADD CONSTRAINT zzz_order_zzz_car_fk FOREIGN KEY ( carid )
       REFERENCES zzz_car ( carid );
```

```
ALTER TABLE zzz_order
   ADD CONSTRAINT zzz_order_zzz_customer_fk FOREIGN KEY ( custid,
                                                          custtype )
       REFERENCES zzz_customer ( custid,
                                 custtype );

ALTER TABLE zzz_order
   ADD CONSTRAINT zzz_order_zzz_discount_fk FOREIGN KEY ( discid,
                                                          disctype )
       REFERENCES zzz_discount ( discid,
                                 disctype );

ALTER TABLE zzz_payment
   ADD CONSTRAINT zzz_payment_zzz_invoice_fk FOREIGN KEY ( invoiceid )
       REFERENCES zzz_invoice ( invoiceid );

-- Trigger used to generate invoice
delimiter |
CREATE TRIGGER UpdateInvoice AFTER UPDATE ON zzz_order
   FOR EACH ROW
BEGIN
   IF NOT(NEW.endOdo <=> OLD.endOdo) THEN
       -- Calcuate amount before discount
       Set @RegularAmount := DATEDIFF(new.endDate, new.startDate) * (select
dailyrate from zzz_car where new.carid = zzz_car.carid);
       Set @OverAmount := 0;
       Set @LimitOdo := DATEDIFF(new.endDate, new.startDate) * new.OdoLimit;
       IF new.endOdo - new.startOdo > @LimitOdo THEN
           Set @OverAmount := (new.endOdo - new.startOdo - @LimitOdo)
                              * (select overrate from zzz_car where new.carid
= zzz_car.carid);
       END IF;

       -- Calcuate discount
       Set @DiscountPerc = 1;
       IF NOT(NEW.disctype <=> NULL) THEN
           IF (NEW.disctype = 'I') THEN
               Set @ValidStart := (select validstart from zzz_discindi where
new.discid = zzz_discindi.discid);
               Set @ValidEnd := (select validend from zzz_discindi where
new.discid = zzz_discindi.discid);
               IF (@ValidStart < new.enddate AND @ValidEnd > new.endDate) THEN
                   Set @DiscountPerc = 1 - 0.01 * (select discpercen from
zzz_discount where new.discid = zzz_discount.discid);
```

```
            END IF;
        ELSEIF (NEW.disctype = 'C') THEN
            Set @DiscountPerc = 1 - 0.01 * (select discpercen from
zzz_discount where new.discid = zzz_discount.discid);
        END IF;
    END IF;

    -- Insert the new record to invoice
    IF (NEW.OdoLimit <=> NULL) THEN
        INSERT INTO zzz_invoice(invoiceid, invoicedate, amount, remain,
orderid) values
            (NEW.orderid, CURDATE(), @RegularAmount, @RegularAmount,
NEW.orderid);
    ELSE
        INSERT INTO zzz_invoice(invoiceid, invoicedate, amount, remain,
orderid) values
            (NEW.orderid, CURDATE(), (@RegularAmount + @OverAmount) *
@DiscountPerc, (@RegularAmount + @OverAmount) * @DiscountPerc, NEW.orderid);
    END IF;

  END IF;
END;
|
delimiter ;
```

8.3. DML Codes (MySQL)

```
-- insert data to address
insert into zzz_address (street, state, country, zipcode)
values
('A St', 'NY', 'USA', '10001'),
('B St', 'NY', 'USA', '10002'),
('C St', 'PA', 'USA', '15003'),
('D St', 'PA', 'USA', '15004'),
('E St', 'CA', 'USA', '94043'),
('F St', 'CA', 'USA', '94086'),
('G St', 'CA', 'USA', '94089'),
('H St', 'MI', 'USA', '48228'),
('I St', 'MI', 'USA', '48103'),
('J St', 'NJ', 'USA', '07097'),
('K St', 'MA', 'USA', '02139'),
('L St', 'MA', 'USA', '02138'),
('M St', 'MA', 'USA', '02137'),
```

```
('N St', 'MA', 'USA', '01604'),
('O St', 'FL', 'USA', '33125'),
('P St', 'FL', 'USA', '33027'),
('Q St', 'FL', 'USA', '32244'),
('R St', 'FL', 'USA', '32808'),
('S St', 'WA', 'USA', '98101'),
('T St', 'WA', 'USA', '98102'),
('O St', 'WA', 'USA', '98004'),
('P St', 'WA', 'USA', '98005'),
('Q St', 'WA', 'USA', '98006'),
('R St', 'WA', 'USA', '98007');

-- insert data to office
insert into zzz_office (phone, addrid)
values
('(123)-456-789', 1),
('(677)-142-124', 2),
('(456)-634-565', 3),
('(535)-645-089', 4),
('(543)-579-019', 5),
('(082)-977-984', 6),
('(973)-234-893', 7),
('(279)-937-761', 8),
('(769)-733-173', 9),
('(692)-763-182', 10),
('(738)-123-918', 11),
('(162)-472-867', 12);

-- insert data to vehicle
insert into zzz_vehicle (vin, make, model, year, lpn)
values
('299J98JSJIW19923I', 'Volvo', 'Golf', 2001, '123989828'),
('28WHDJW92U939282', 'Honda', 'Acura', 2020, '123899233'),
('894739JUHE2932', 'General', 'Buick', 2003, '123719232'),
('283DWJEHBDU12', 'Toyota', 'Lexus', 2010, '12389SH23'),
('1231283DHUIHU', 'Ford', 'Lincoln', 2012, 'UH198929'),
('HIH289389223', 'Mazda', 'Mazda', 2015, 'HUUE97283'),
('PEI893904212', 'BMW', 'Mini', 2013, 'JIH172HU1'),
('838HIDN8929321', 'Stellantis', 'Ram', 2000, 'GHJDU288'),
('EIWDODO123871', 'BMW', 'Rolls-Royce', 2012, 'UD72JDS'),
('19HDHJ929391', 'Subaru', 'Subaru', 2014, 'DUU1738HJD'),
('BDU12378492112', 'Tesla', 'Tesla', 2020, '89273HJHA');

-- insert data to car
```

```
insert into zzz_car (cartype, dailyrate, overrate, officeid, vin, imgurl)
values
('Small car', 1.0, 50.0, 1, '299J98JSJIW19923I',
'https://media.zipcar.com/images/model-image?model_id=64790058'),
('Mid-size car', 78.28, 10.29, 2, '28WHDJW92U939282',
'https://media.zipcar.com/images/model-image?model_id=94567'),
('Luxury car', 28.17, 29.29, 3,
'894739JUHE2932','https://media.zipcar.com/images/model-image?model_id=1651832377'),
('SUV', 92.12, 82.23, 4,
'283DWJEHBDU12','https://media.zipcar.com/images/model-image?model_id=22973230115'),
('Premium SUV', 19.29, 28.39, 5,
'1231283DHUIHU','https://media.zipcar.com/images/model-image?model_id=475722000'),
('Mini Van', 12.23, 34.72, 6,
'HIH289389223','https://media.zipcar.com/images/model-image?model_id=64790058'),
('Station Wagon', 49.38, 19.3, 7,
'PEI893904212','https://media.zipcar.com/images/model-image?model_id=94567'),
('Small car',28.34,20.0, 8,
'838HIDN8929321','https://media.zipcar.com/images/model-image?model_id=94567'),
('Premium SUV', 72.83, 22.92, 9,
'EIWDODO123871','https://media.zipcar.com/images/model-image?model_id=1651832377'),
('Station Wagon', 48.20, 29.32,
10,'19HDHJ929391','https://media.zipcar.com/images/model-image?model_id=475722000'),
('SUV', 23.12, 28.23, 11,
'BDU12378492112','https://media.zipcar.com/images/model-image?model_id=64790058');

-- insert data to customer
insert into zzz_customer (custtype, email, phone, addrid)
values
('I', 'erte@nyu.edu', '(162)-472-844', 13),
('I', 'cxbd@nyu.edu', '(162)-472-845', 13),
('I', 'yijj@nyu.edu', '(162)-472-846', 14),
('I', 'cvfh@nyu.edu', '(162)-472-847', 15),
('I', 'nyif@nyu.edu', '(162)-472-848', 16),
('C', 'xcxg@nyu.edu', '(162)-472-849', 18),
('C', 'iygj@nyu.edu', '(162)-472-850', 19),
('C', 'xzzz@nyu.edu', '(162)-472-851', 20),
('C', 'oyou@nyu.edu', '(162)-472-852', 21),
```

```
('C', 'bnfg@nyu.edu', '(162)-472-853', 22);

-- insert data to individual
insert into zzz_individual (custid, custtype, lname, fname, licensenum,
insname, insnum)
values
(1, 'I', 'Smith', 'Jack', '218984751', 'Good Insurance', 36571),
(2, 'I', 'Will', 'Lily', '218984752', 'Better Insurance', 36572),
(3, 'I', 'Wade', 'Micheal', '218984753', 'Awesome Insurance', 36573),
(4, 'I', 'Green', 'Tom', '218984754', 'Cool Insurance', 36574),
(5, 'I', 'Gates', 'Jessie', '218984755', 'Nice Insurance', 36575);

-- insert data to corporate
insert into zzz_corporate (custid, custtype, corpname, regnum, empid)
values
(6, 'C', 'Amazon', 'amz123', 'amz456'),
(7, 'C', 'Google', 'gg123', 'gg456'),
(8, 'C', 'Meta', 'fb123', 'fb456'),
(9, 'C', 'Apple', 'apple123', 'apple456'),
(10, 'C', 'Linkedin', 'ln123', 'ln456');

-- insert data to discounts
insert into zzz_discount (disctype, discpercen)
values
('I', 30),
('I', 34),
('I', 38),
('I', 20),
('I', 21),
('I', 30),
('I', 39),
('C', 40),
('C', 44),
('C', 48),
('C', 30),
('C', 31),
('C', 40),
('C', 49);

-- insert data to discindi
insert into zzz_discindi (discid, disctype, validstart, validend, coupnum)
values
(1, 'I', '2022-03-03', '2023-03-31', 9675731),
(2, 'I', '2022-02-01', '2024-03-01', 5760897),
```

```
(3, 'I', '2019-05-31', '2019-06-30', 0903232),
(4, 'I', '2021-07-01', '2021-07-31', 5881648),
(5, 'I', '2021-11-05', '2021-12-01', 5891678),
(6, 'I', '2022-03-05', '2025-03-15', 7689352),
(7, 'I', '2022-03-05', '2024-03-15', 7908112);

-- insert data to disccorp
insert into zzz_disccorp (discid, disctype, setnum)
values
(8, 'C', 0146731),
(9, 'C', 0146757),
(10, 'C', 0158931),
(11, 'C', 0187190),
(12, 'C', 0167880),
(13, 'C', 0778906),
(14, 'C', 0167890);

-- insert data to zzz_orders
insert into zzz_order (startodo, endodo, odolimit, startdate, enddate, custid,
custtype, pickup, dropoff, carid, discid, disctype)
values
(1000, null, 40, '2020-03-01', '2020-03-05', 1, 'I', 2, 3, 1, null, null),
(1000, null, 40, '2020-03-01', '2020-03-05', 1, 'I', 2, 3, 1, null, null),
(1000, null, null, '2020-03-01', '2020-03-05', 1, 'I', 2, 3, 1, null, null),
(1000, null, 40, '2020-03-01', '2020-03-05', 1, 'I', 2, 3, 1, 2, 'I'),
(1000, null, 40, '2021-07-01', '2021-07-05', 1, 'I', 2, 3, 1, 4, 'I'),
(1000, null, 40, '2021-07-01', '2021-07-05', 6, 'C', 2, 3, 1, 8, 'C');

-- insert data to zzz_account
insert into zzz_account (accname, pwd, custid, custtype)
values
('admin', '6512bd43d9caa6e02c990b0a82652dca', 2, 'I'),
('aa', '6512bd43d9caa6e02c990b0a82652dca', 1, 'I'),
('bb', '6512bd43d9caa6e02c990b0a82652dca', 7, 'C');

insert into zzz_role (rid, role)
values
(1, 'customer'),
(2, 'admin');

insert into zzz_acc_role (accid, rid)
values
(1, 2),
(2, 1),
```

```
(3, 1);

-- add an invoice
update zzz_order set endodo=1100 where orderid=1; -- 4
update zzz_order set endodo=1500 where orderid=2; -- 17004

commit;
```