**Project Environment**

This project will be developed in NetBeans IDE, using Apache Tomcat as web server and MySQL as database management system. NetBeans IDE is a free, open source which helps in developing web applications quickly and easy. Apache Tomcat is an open source web server and servlet container. MySQL is an open source relational database management system. It is mostly used for its quick processing, proven reliability, ease and flexibility of use.

**Project Requirement**

In our project we would like to implement a **car rental system**, through which **customers** can rent **cars** online. Customers can be registered or non-registered. Registered members have to obtain a **membership ID** and must provide **credit card details** and **license** as a form of **security**. We want our customers to select pickup date, time and location to browse through cars **available** then. Once they select the date/time, they will choose the type of car they want like sedan, economy, hatchback, SUV etc. After selecting the type of car, they will be prompted to a vehicle details page. On this page, they will be given estimated **amount** they have to pay depending the time and car they rent. Along with the rent, all the taxes, coupons (if any) will be added and the estimate will be shown. The **registered customers** are entitled to **membership discounts** depending upon the criteria. If they have used our service for 5 times or more, then they are entitled to have a 10% discount. Then, the customer confirms his request by paying through his valid credit card. Once he confirms his order, an invoice will be emailed to him, confirming his pickup date, location, time. When the customer walks in to pick up his car, he then have to provide his driver's license as additional form of security along with card details.

The customers who are not yet members, can sign up and have their membership ID to enjoy the membership discounts and **coupons**. In this case, when they click the sign up option, they will be directed to registration page, where they have to provide all the details about them like name, email address, mailing address and credit card details. There they can set up their username and password for future use.

The **admin** of this system will be having access to all the car rental database and the customer details. He is responsible for revising the car rental prices, adding or removing any car, coupons and discounts to be given. He will generate monthly reports, finalize the orders and review them. He maintains contact with customers through emails. When a customer checks out a car he requested, the admin will update his **order page** by adding the **license plate** of the car given to him and his license number. This page will be updated finally when the car is returned. This page is not accessible to the customers but admin.

The customer, registered or non-registered can access only the available cars according to the given date/ time and location. They can even cancel their orders if they want to and it should be at least 24 hours prior to the pickup time. If the customer fails to do that he will be charged for that particular day. If the customer fails to return the car on decided return date, they will be charged **overdue rent** of 35$ per day.

**Conceptual Design**

**A: Noun-Verb Identification**

**List of Nouns**

customer

admin

car

membership id

price

type

availability

membership discounts

registered customers

amount

credit card details

license

overdue rent

security

license plate

coupons

order page

**List of Verbs**

implement

rent

fails

charged

update

responsible

checks out

returned

access

sign up

confirm

browse

provide

obtain

select

pay

**B: Entities, Relationships and Attributes**

Strong Entity **Customer** (Generalization) with non-composite single-valued attributes

**Entity: Customer**

email\_id

first\_name

middle\_name

last\_name

phone\_number

registration\_time

Strong Entity **RegisteredCustomer** (Specialization) with non-composite single-valued attributes

**Entity: RegisteredCustomer**

email\_id

membership\_id

password

**Foreign Key:** email\_id (From Customer)

Derived attribute of the strong entity **RegisteredCustomer** - num\_rentals

Strong Entity **CarCategory** with non-composite single-valued attributes

**Entity: CarCategory**

category\_id

category\_name

renting\_price

Derived attribute of the strong entity **CarCategory** - availability

Strong Entity **Reservation** with non-composite single-valued attributes, Binary Relationship (one-to-many) between the strong entities **Customer** and **Reservation** (Customer **Makes** Reservation) and Binary Relationship (one-to-one) between the strong entities **Reservation** and **CarCategory** (Reservation **For** CarCategory)

**Entity: Reservation**

reservation\_id

email\_id

category\_id

pick\_up\_date

return\_date

Derived attributes of the strong entity **Reservation** - discount\_amount, transaction\_amount

**Foreign Key Approach:** email\_id (Primary Key of **Customer**), category\_id (Primary Key of **CarCategory**) are included in the strong entity **Reservation** as Foreign Keys

Weak entity **Driver** with non-composite single-valued attributes (Strong entity: **Reservation**) and Weak Binary Relationship (one-to-one) between the strong entity **Reservation** and the weak entity **Driver** (Reservation **UsedBy** Driver)

**Entity: Driver**

reservation\_id

license\_id

first\_name

middle\_name

last\_name

**Foreign Key Approach:** reservation\_id (Primary Key of **Reservation**) is included in the weak entity **Driver** as a Foreign Key

Strong entity **Car** with non-composite single-valued attributes, Binary Relationship (one-to-many) between the strong entities **CarCategory** and **Car** (CarCategory **Has** Car)

**Entity: Car**

VIN

category\_id

license\_plate

availability

**Foreign Key Approach:** category\_id (Primary Key of **CarCategory**) is included in the strong entity **Car** as a Foreign Key

Weak Binary Relationship (one-to-one) between the weak entity **Driver** and the strong entity **Car**

Driver **PicksUp/Returns** a Car

**PicksUpReturns**

reservation\_id

license\_id

VIN

pick\_up\_time

return\_time

cc\_name

cc\_number

cc\_cvv

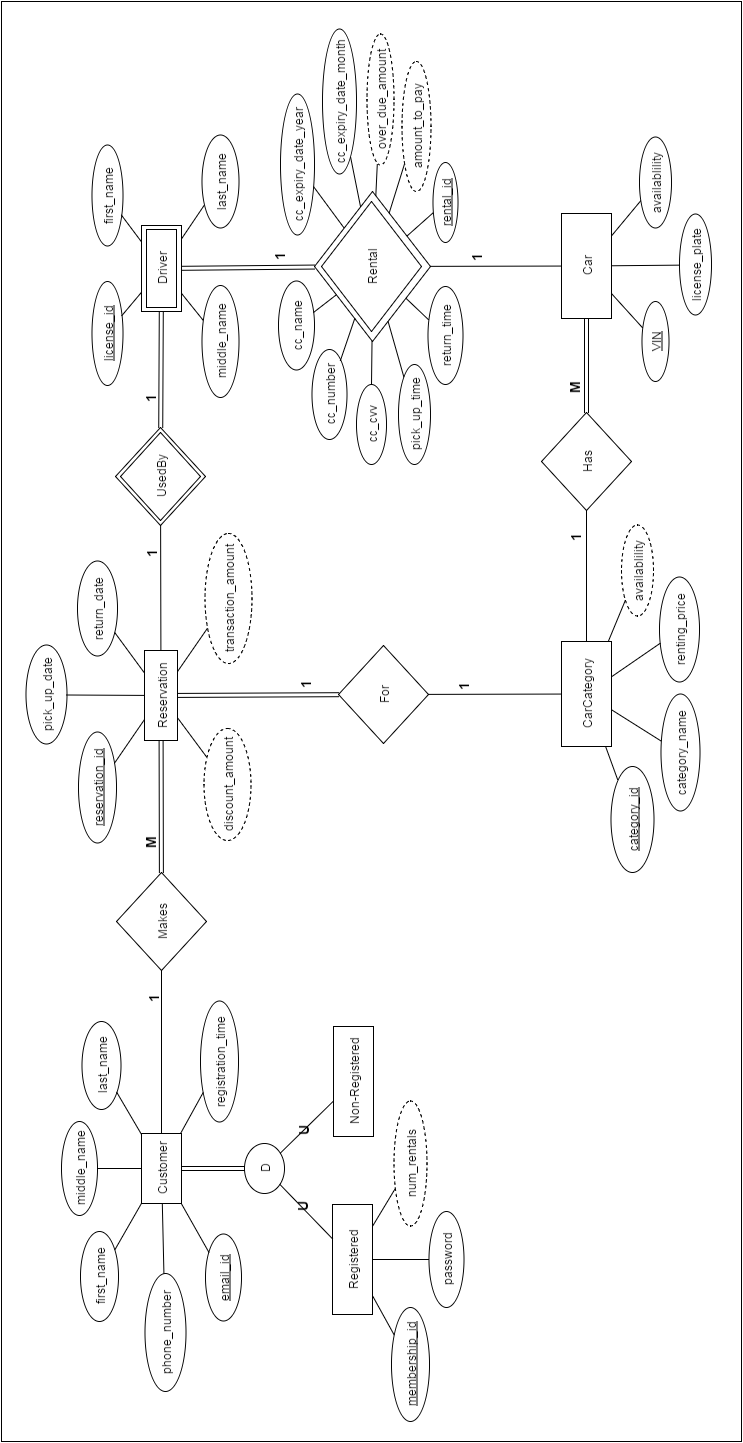
cc\_expiry\_date\_year

cc\_expiry\_date\_month

Derived attributes of the relationship **PicksUpReturns** - over\_due\_amount, amount\_to\_pay

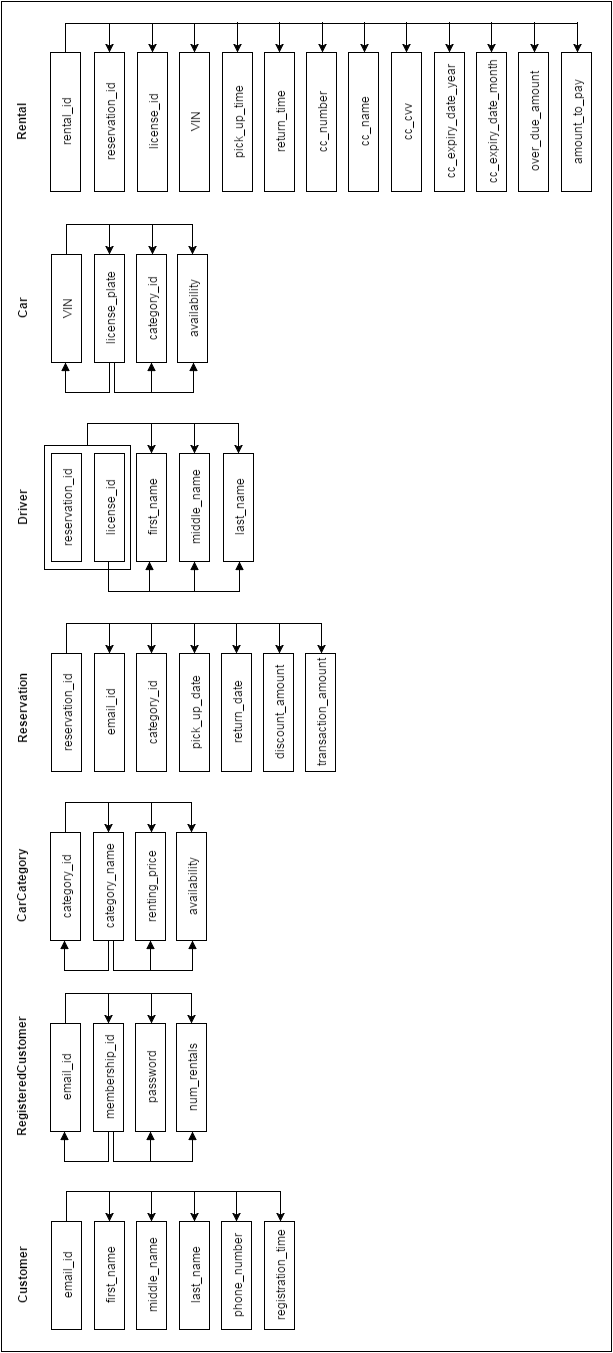
**Foreign Keys:** reservation\_id, license\_id (From Driver), VIN (From Car)

**C: Entity-Relationship Diagram**



**D: Logical Design**

**Functional Dependencies**

****

**Table: Customer**

email\_id VARCHAR(30)

first\_name VARCHAR(40) NOT NULL

middle\_name VARCHAR(20)

last\_name VARCHAR(20) NOT NULL

phone\_number CHAR(10) NOT NULL

registration\_time TIMESTAMP

**Primary Key:** email\_id

**Highest Normal Form:** BCNF

**Justification:** We are not checking for 4NF

**Table: RegisteredCustomer**

email\_id VARCHAR(30)

membership\_id INT(4) NOT NULL

password VARCHAR(20) NOT NULL

num\_rentals INT DEFAULT 0

**Primary Key:** email\_id

**Foreign Key:** email\_id REFERENCES Customer (email\_id)

**Highest Normal Form:** BCNF

**Justification:** We are not checking for 4NF

**Table: CarCategory**

category\_id INT(4)

category\_name VARCHAR(20) NOT NULL

renting\_price DOUBLE NOT NULL

availability BOOLEAN DEFAULT 0

**Primary Key:** category\_id

**Highest Normal Form:** BCNF

**Justification:** We are not checking for 4NF

**Table: Reservation**

reservation\_id INT(4)

email\_id VARCHAR(30)

category\_id INT(4)

pick\_up\_date DATE NOT NULL

return\_date DATE NOT NULL

discount\_amount DOUBLE DEFAULT 0.00

transaction\_amount DOUBLE DEFAULT 0.00

**Primary Key:** reservation\_id

**Foreign Key:** email\_id REFERENCES Customer (email\_id)

**Foreign Key:** category\_id REFERENCES CarCategory (category\_id)

**Highest Normal Form:** BCNF

**Justification:** We are not checking for 4NF

**Table: Driver**

reservation\_id INT(4)

license\_id VARCHAR(15)

first\_name VARCHAR(40) NOT NULL

middle\_name VARCHAR(20)

last\_name VARCHAR(20) NOT NULL

**Primary Key:** reservation\_id, license\_id

**Foreign Key:** reservation\_id REFERENCES Reservation (reservation\_id)

**Highest Normal Form:** 1NF

**Justification:** reservation and license id together form candidate key. first\_name, last\_name and middle\_name do not fully depend on candidate key, violating 2NF.

**Table: Car**

VIN VARCHAR(18)

category\_id INT(4)

license\_plate CHAR(8) NOT NULL

availability BOOLEAN DEFAULT 0

**Primary Key:** VIN

**Foreign Key:** category\_id REFERENCES CarCategory (category\_id)

**Highest Normal Form:** BCNF

**Justification:** We are not checking for 4NF

**Table: Rental**

rental\_id INT(4)

reservation\_id INT(4)

license\_id VARCHAR(15)

VIN VARCHAR(18)

pick\_up\_time TIMESTAMP NOT NULL

return\_time TIMESTAMP

cc\_name VARCHAR(30) NOT NULL

cc\_number CHAR(16) NOT NULL

cc\_cvv CHAR(3) NOT NULL

cc\_expiry\_date\_year CHAR(4) NOT NULL

cc\_expiry\_date\_month CHAR(2) NOT NULL

over\_due\_amount DOUBLE DEFAULT 0.00

amount\_to\_pay DOUBLE DEFAULT 0.00

**Primary Key:** rental\_id

**Foreign Key:** reservation\_id, license\_id REFERENCES Driver (reservation\_id, license\_id)

**Foreign Key:** VIN REFERENCES Car (VIN)

**Highest Normal Form:** 1NF

**Justification:** reservation\_id, license\_id, VIN, check\_out\_date form candidate key and non-key

attributes do not fully depend on candidate key, violating 2NF.

**E: Creation of Database and Tables (SQL Dump)**

CREATE DATABASE CarRentalSystem;

USE CarRentalSystem;

CREATE TABLE Customer

(

email\_id VARCHAR(30),

first\_name VARCHAR(40) NOT NULL,

middle\_name VARCHAR(20),

last\_name VARCHAR(20) NOT NULL,

phone\_number CHAR(10) NOT NULL,

registration\_time TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

PRIMARY KEY (email\_id),

INDEX index\_customer\_email (email\_id),

INDEX index\_customer\_phone (phone\_number),

INDEX index\_customer\_name (last\_name, first\_name)

);

CREATE TABLE RegisteredCustomer

(

membership\_id INT(4) UNSIGNED AUTO\_INCREMENT,

email\_id VARCHAR(30) UNIQUE NOT NULL,

password VARCHAR(20) NOT NULL,

num\_rentals INT DEFAULT 0,

PRIMARY KEY (membership\_id),

FOREIGN KEY (email\_id) REFERENCES Customer (email\_id) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT uc\_member UNIQUE (email\_id, membership\_id),

CONSTRAINT uc\_login UNIQUE (email\_id, password),

INDEX index\_registeredcustomer\_membership (membership\_id),

INDEX index\_registeredcustomer\_email (email\_id)

) AUTO\_INCREMENT=1000;

CREATE TABLE CarCategory

(

category\_id INT(4) UNSIGNED AUTO\_INCREMENT,

category\_name VARCHAR(20) UNIQUE NOT NULL,

renting\_price DOUBLE NOT NULL DEFAULT 0.00,

availability BOOLEAN DEFAULT 1,

PRIMARY KEY (category\_id),

CONSTRAINT uc\_member UNIQUE (category\_name),

INDEX index\_carcategory\_id (category\_id),

INDEX index\_carcategory\_name (category\_name),

INDEX index\_carcategory\_price (renting\_price)

) AUTO\_INCREMENT=6001;

CREATE TABLEReservation

(

reservation\_id INT(4) UNSIGNED AUTO\_INCREMENT,

email\_id VARCHAR(30),

category\_id INT(4) UNSIGNED,

pick\_up\_date DATE NOT NULL,

return\_date DATE NOT NULL,

discount\_amount DOUBLE DEFAULT 0.00,

transaction\_amount DOUBLE DEFAULT 0.00,

PRIMARY KEY (reservation\_id),

FOREIGN KEY (email\_id) REFERENCES Customer (email\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (category\_id) REFERENCES CarCategory (category\_id) ON DELETE CASCADE ON UPDATE CASCADE,

INDEX index\_reservation\_id (reservation\_id),

INDEX index\_reservation\_email (email\_id),

INDEX index\_reservation\_pickupdate (pick\_up\_date),

INDEX index\_reservation\_returndate (return\_date)

) AUTO\_INCREMENT=2001;

CREATE TABLE Driver

(

reservation\_id INT(4) UNSIGNED,

license\_id VARCHAR(15),

first\_name VARCHAR(40) NOT NULL,

middle\_name VARCHAR(20),

last\_name VARCHAR(20) NOT NULL,

PRIMARY KEY (reservation\_id, license\_id),

FOREIGN KEY (reservation\_id) REFERENCES Reservation (reservation\_id) ON DELETE CASCADE ON UPDATE CASCADE,

INDEX index\_driver\_reservation (reservation\_id, license\_id),

INDEX index\_driver\_name (last\_name, first\_name)

);

CREATE TABLE Car

(

VIN VARCHAR(18),

category\_id INT(4) UNSIGNED,

license\_plate CHAR(8) NOT NULL,

availability BOOLEAN DEFAULT 1,

PRIMARY KEY (VIN),

FOREIGN KEY (category\_id) REFERENCES CarCategory (category\_id) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT uc\_licenseplate UNIQUE (license\_plate),

INDEX index\_car\_VIN (VIN),

INDEX index\_car\_category (category\_id),

INDEX index\_car\_licenseplate (license\_plate)

);

CREATE TABLE Rental

(

rental\_id INT(4) UNSIGNED AUTO\_INCREMENT,

reservation\_id INT(4) UNSIGNED,

license\_id VARCHAR(15),

VIN CHAR(18),

pick\_up\_time TIMESTAMP NOT NULL DEFAULT CURRENT\_TIMESTAMP,

return\_time TIMESTAMP,

cc\_name VARCHAR(30) NOT NULL,

cc\_number CHAR(16) NOT NULL,

cc\_cvv CHAR(3) NOT NULL,

cc\_expiry\_date\_year CHAR(4) NOT NULL,

cc\_expiry\_date\_month CHAR(2) NOT NULL,

over\_due\_amount DOUBLE,

amount\_to\_pay DOUBLE,

PRIMARY KEY (rental\_id),

FOREIGN KEY (reservation\_id, license\_id) REFERENCES Driver (reservation\_id, license\_id) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (VIN) REFERENCES Car (VIN) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT uc\_rental UNIQUE (reservation\_id),

INDEX index\_rental\_id (rental\_id),

INDEX index\_rental\_reservation (reservation\_id),

INDEX index\_rental\_license (license\_id),

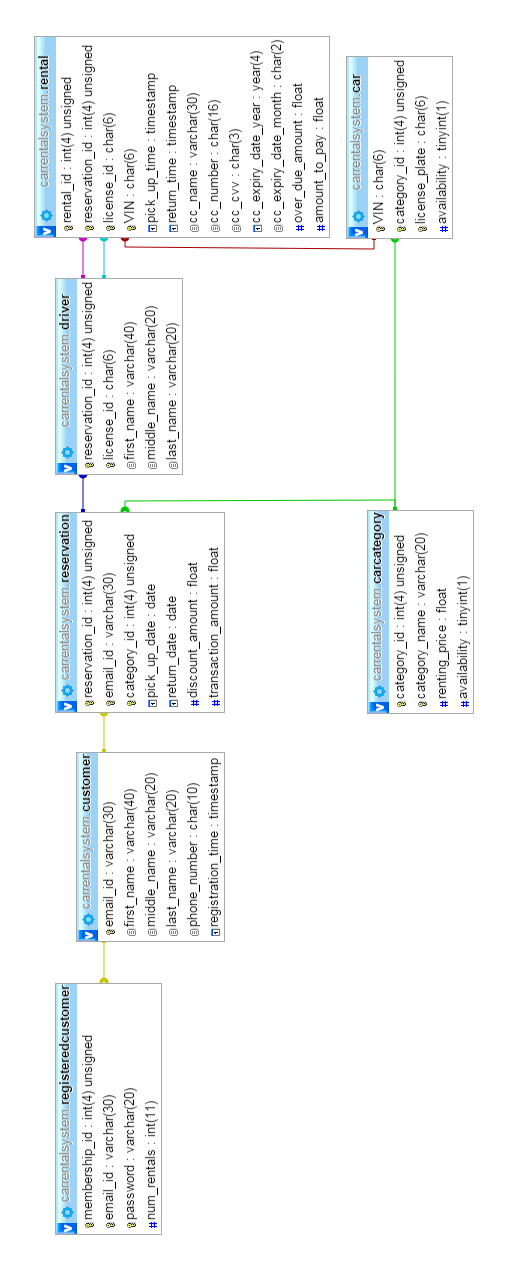
INDEX index\_rental\_VIN (VIN),

INDEX index\_rental\_ccname (cc\_name),

INDEX index\_rental\_ccnumber (cc\_number)

) AUTO\_INCREMENT=7001;

**Tables Design:**

****

**Functions:**

CREATE FUNCTION f\_validate\_user (eid VARCHAR(30), pwd VARCHAR(30)) RETURNS TINYINT(1)

BEGIN

DECLARE exist BOOLEAN;

SELECT EXISTS (SELECT \* FROM RegisteredCustomer WHERE email\_id=eid AND password=pwd) INTO @exist;

RETURN @exist;

END

CREATE FUNCTION f\_validate\_new\_user (eid VARCHAR(30)) RETURNS TINYINT(1)

BEGIN

DECLARE exist BOOLEAN;

SELECT EXISTS (SELECT \* FROM RegisteredCustomer WHERE email\_id=eid) INTO @exist;

IF @exist = 1

THEN

RETURN 0;

ELSE

RETURN 1;

END IF;

END

CREATE FUNCTION f\_validate\_guest (eid VARCHAR(30)) RETURNS TINYINT(1)

BEGIN

DECLARE exist BOOLEAN;

SELECT EXISTS (SELECT \* FROM Customer WHERE email\_id=eid) INTO @exist;

IF @exist = 1

THEN

RETURN 0;

ELSE

RETURN 1;

END IF;

END

CREATE FUNCTION f\_validate\_car\_category (cid INT(4)) RETURNS TINYINT(1)

BEGIN

DECLARE exist BOOLEAN;

SELECT EXISTS (SELECT \* FROM Carcategory WHERE category\_id=cid) INTO @exist;

RETURN @exist;

END

CREATE FUNCTION f\_validate\_car\_category\_availability (cid INT(4), pdate DATE) RETURNS TINYINT(1)

BEGIN

DECLARE exist BOOLEAN;

SELECT EXISTS (SELECT \* FROM v\_category\_available\_carcount WHERE CategoryID=cid) INTO @exist;

RETURN @exist;

END

CREATE FUNCTION f\_return\_actual\_days\_count (rid INT(4)) RETURNS INT(11)

BEGIN

DECLARE dayscount INT;

SELECT DATEDIFF(DATE(return\_time), DATE(pick\_up\_time)) INTO @dayscount FROM Rental WHERE rental\_id=rid;

RETURN @dayscount;

END

CREATE FUNCTION f\_return\_car\_category\_id (rid INT(4)) RETURNS INT(4)

BEGIN

DECLARE cid INT(4);

SELECT category\_id INTO @cid FROM Reservation WHERE reservation\_id=rid;

RETURN @cid;

END

CREATE FUNCTION f\_return\_car\_category\_name (cid INT(4)) RETURNS VARCHAR(20)

BEGIN

DECLARE name VARCHAR(60);

SELECT category\_name INTO @name FROM CarCategory WHERE category\_id=cid;

RETURN @name;

END

CREATE FUNCTION f\_return\_cc\_name (rid INT(4)) RETURNS VARCHAR(30)

BEGIN

DECLARE ccname VARCHAR(30);

SELECT cc\_name INTO @ccname FROM Rental WHERE rental\_id=rid;

RETURN @ccname;

END

CREATE FUNCTION f\_return\_cc\_number (rid INT(4)) RETURNS CHAR(16)

BEGIN

DECLARE ccnumber CHAR(16);

SELECT cc\_number INTO @ccnumber FROM Rental WHERE rental\_id=rid;

RETURN @ccnumber;

END

CREATE FUNCTION f\_return\_days\_count (pdate DATE, rdate DATE) RETURNS INT(11)

BEGIN

RETURN DATEDIFF(rdate, pdate);

END

CREATE FUNCTION f\_return\_discount (rid INT(4)) RETURNS DOUBLE

BEGIN

DECLARE discount DOUBLE;

SELECT discount\_amount INTO @discount FROM Reservation WHERE reservation\_id=rid;

RETURN @discount;

END

CREATE FUNCTION f\_return\_discount\_percentage (mid INT(4)) RETURNS INT(11)

BEGIN

DECLARE discount INT;

DECLARE count INT;

SET @discount=0;

SELECT num\_rentals INTO @count FROM RegisteredCustomer WHERE membership\_id=mid;

IF @count<10

THEN

SET @discount=0;

ELSEIF (@count>=10 & @count<=30)

THEN

SET @discount=5;

ELSEIF (@count>30 & @count<50)

THEN

SET @discount=10;

ELSEIF (@count>50)

THEN

SET @discount=15;

END IF;

RETURN @discount;

END

CREATE FUNCTION f\_return\_expected\_days\_count (rid INT(4)) RETURNS INT(11)

BEGIN

DECLARE dayscount INT;

SELECT DATEDIFF(return\_date, pick\_up\_date) INTO @dayscount FROM Reservation WHERE reservation\_id=rid;

RETURN @dayscount;

END

CREATE FUNCTION f\_return\_membership\_id (eid VARCHAR(30)) RETURNS INT(4)

BEGIN

DECLARE mid INT(4);

SELECT membership\_id INTO @mid FROM RegisteredCustomer WHERE email\_id=eid;

RETURN @mid;

END

CREATE FUNCTION f\_return\_name (eid VARCHAR(30)) RETURNS VARCHAR(60)

BEGIN

DECLARE name VARCHAR(60);

SELECT CONCAT(first\_name, ' ', last\_name) INTO @name FROM Customer WHERE email\_id=eid;

RETURN @name;

END

CREATE FUNCTION f\_return\_pick\_up\_time (rid INT(4)) RETURNS TIMESTAMP

BEGIN

DECLARE ptime TIMESTAMP;

SELECT pick\_up\_time INTO @ptime FROM Rental WHERE rental\_id=rid;

RETURN @ptime;

END

CREATE FUNCTION f\_return\_rental\_id (rid INT(4)) RETURNS INT(4)

BEGIN

DECLARE rentalid INT(4);

SELECT rental\_id INTO @rentalid FROM Rental WHERE reservation\_id=rid;

RETURN @rentalid;

END

CREATE FUNCTION f\_return\_renting\_price (cid INT(4)) RETURNS DOUBLE

BEGIN

DECLARE price DOUBLE;

SELECT renting\_price INTO @price FROM CarCategory WHERE category\_id=cid;

RETURN @price;

END

CREATE FUNCTION f\_return\_reservation\_id (eid VARCHAR(30), cid INT(4), pdate DATE) RETURNS INT(4)

BEGIN

DECLARE rid INT(4);

SELECT reservation\_id INTO @rid FROM Reservation WHERE email\_id=eid AND category\_id=cid AND pick\_up\_date=pdate;

RETURN @rid;

END

CREATE FUNCTION f\_return\_return\_date (rid INT(4)) RETURNS DATE

BEGIN

DECLARE rdate INT(4);

SELECT return\_date INTO @rdate FROM Reservation WHERE reservation\_id=rid;

RETURN @rdate;

END

CREATE FUNCTION f\_return\_return\_time (rid INT(4)) RETURNS TIMESTAMP

BEGIN

DECLARE rtime TIMESTAMP;

SELECT return\_time INTO @rtime FROM Rental WHERE rental\_id=rid;

RETURN @rtime;

END

CREATE FUNCTION f\_return\_vin (rid INT(4)) RETURNS VARCHAR(18)

BEGIN

DECLARE carvin VARCHAR(18);

SELECT VIN INTO @carvin FROM Rental WHERE rental\_id=rid;

RETURN @carvin;

END

**Procedures:**

CREATE PROCEDURE sp\_add\_customer (fname VARCHAR(40), mname VARCHAR(20), lname VARCHAR(20), pnum CHAR(10), eid VARCHAR(30))

BEGIN

INSERT INTO Customer (first\_name, middle\_name, last\_name, phone\_number, email\_id) VALUES (fname, mname, lname, pnum, eid);

END

CREATE PROCEDURE sp\_add\_registeredcustomer (eid VARCHAR(30), pwd VARCHAR(20))

BEGIN

INSERT INTO RegisteredCustomer (email\_id, password) VALUES (eid, pwd);

END

CREATE PROCEDURE sp\_add\_rental (rid INT(4), lid VARCHAR(15), ccname VARCHAR(30), ccnum CHAR(16), year CHAR(4), month CHAR(2), cvv CHAR(3))

BEGIN

DECLARE carvin CHAR(18);

DECLARE valid BOOLEAN;

DECLARE cid INT(4);

SELECT category\_id INTO @cid FROM Reservation WHERE reservation\_id=rid;

SELECT VIN INTO @carvin FROM Car WHERE category\_id=@cid AND availability=1 LIMIT 1;

UPDATE Car SET availability=0 WHERE VIN=@carvin;

SELECT EXISTS (SELECT CategoryID FROM v\_category\_available\_carcount V WHERE CategoryID=@cid) INTO @valid;

IF @valid = 0

THEN

UPDATE CarCategory SET availability=0 WHERE category\_id=@cid;

END IF;

INSERT INTO Rental (reservation\_id, license\_id, cc\_name, cc\_number, cc\_expiry\_date\_year, cc\_expiry\_date\_month, cc\_cvv, pick\_up\_time, VIN) VALUES (rid, lid, ccname, ccnum, expyear, month, cvv, NOW(), @carvin);

END

CREATE PROCEDURE sp\_add\_reservation\_driver (eid VARCHAR(30), cid INT(4), pdate DATE, rdate DATE, discount DOUBLE, amount DOUBLE, lid VARCHAR(15), fname VARCHAR(40), mname VARCHAR(20), lname VARCHAR(20))

BEGIN

DECLARE rid INT(4);

INSERT INTO Reservation (email\_id, category\_id, pick\_up\_date, return\_date, discount\_amount, transaction\_amount) VALUES (eid, cid, pdate, rdate, discount, amount);

SELECT reservation\_id INTO @rid FROM Reservation WHERE email\_id=eid AND category\_id=cid AND pick\_up\_date=pdate;

INSERT INTO Driver (reservation\_id, license\_id, first\_name, middle\_name, last\_name) VALUES (@rid, lid, fname, mname, lname);

END

CREATE PROCEDURE sp\_update\_customer (fname VARCHAR(40), mname VARCHAR(20), lname VARCHAR(20), pnum CHAR(10), eid VARCHAR(30))

BEGIN

UPDATE Customer SET first\_name=fname, middle\_name=mname, last\_name=lname, phone\_number=pnum WHERE email\_id=eid;

END

CREATE PROCEDURE sp\_update\_rental (rid INT(4), lid VARCHAR(15), carvin CHAR(18))

BEGIN

DECLARE due DOUBLE;

DECLARE pay DOUBLE;

DECLARE rentperday DOUBLE;

DECLARE discount DOUBLE;

DECLARE basic DOUBLE;

DECLARE tax DOUBLE;

DECLARE daycount INT;

DECLARE actualdaycount INT;

DECLARE eid VARCHAR(30);

DECLARE valid BOOLEAN;

UPDATE Car SET availability=1 WHERE VIN=carvin;

UPDATE CarCategory SET availability=1 WHERE category\_id=(SELECT category\_id FROM Car WHERE VIN=carvin);

SELECT email\_id INTO @eid FROM Reservation WHERE reservation\_id=rid;

SELECT DATEDIFF(return\_date, pick\_up\_date) INTO @daycount FROM Reservation WHERE reservation\_id=rid;

SELECT DATEDIFF(CURDATE(), pick\_up\_date) INTO @actualdaycount FROM Reservation WHERE reservation\_id=rid;

SELECT CC.renting\_price INTO @rentperday FROM CarCategory CC JOIN Reservation R ON CC.category\_id=R.category\_id AND reservation\_id=rid;

IF @actualdaycount>@daycount

THEN

SET @due = ((@actualdaycount - @daycount)\*@rentperday);

ELSE

SET @due=0;

END IF;

SELECT discount\_amount INTO @discount FROM Reservation WHERE reservation\_id=rid;

SET @basic = @actualdaycount\*@rentperday;

SET @tax = (8.25\*@basic)/100;

SET @pay = @basic+@tax-@discount;

UPDATE Rental SET return\_time=NOW(), over\_due\_amount=@due, amount\_to\_pay=@pay WHERE reservation\_id=rid;

UPDATE RegisteredCustomer SET num\_rentals=(num\_rentals+1) WHERE email\_id=@eid;

END

**Not Used:**

CREATE PROCEDURE sp\_generate\_membership\_id ()

BEGIN

UPDATE RegisteredCustomer SET membership\_id = (SELECT FLOOR(1000.0 + RAND()\*999.0) AS id FROM RegisteredCustomer WHERE membership\_id = NULL AND id NOT IN (SELECT membership\_id FROM RegisteredCustomer) LIMIT 1);

END

CREATE PROCEDURE sp\_generate\_category\_id ()

BEGIN

UPDATE CarCategory SET category\_id = (SELECT FLOOR(6000.0 + RAND()\*999.0) AS id FROM CarCategory WHERE category\_id = NULL AND id NOT IN (SELECT category\_id FROM CarCategory) LIMIT 1);

END

CREATE PROCEDURE sp\_generate\_reservation\_id ()

BEGIN

UPDATE Reservation SET reservation\_id = (SELECT FLOOR(2000.0 + RAND()\*999.0) AS id FROM Reservation WHERE reservation\_id = NULL AND id NOT IN (SELECT reservation\_id FROM Reservation) LIMIT 1);

END

CREATE PROCEDURE sp\_generate\_rental\_id ()

BEGIN

UPDATE Rental SET rental\_id = (SELECT FLOOR(7000.0 + RAND()\*999.0) AS id FROM Rental WHERE rental\_id = NULL AND id NOT IN (SELECT rental\_id FROM Rental) LIMIT 1);

END

**Triggers:**

**Not Used:**

CREATE TRIGGER tr\_membership\_id AFTER INSERT ON RegisteredCustomer FOR EACH ROW

BEGIN

UPDATE RegisteredCustomer SET membership\_id = (SELECT FLOOR(1000.0 + RAND()\*999.0) AS id FROM RegisteredCustomer WHERE membership\_id = NULL AND id NOT IN (SELECT membership\_id FROM RegisteredCustomer) LIMIT 1);

END

CREATE TRIGGER tr\_category\_id AFTER INSERT ON CarCategory FOR EACH ROW

BEGIN

UPDATE CarCategory SET category\_id = (SELECT FLOOR(6000.0 + RAND()\*999.0) AS id FROM CarCategory WHERE category\_id = NULL AND id NOT IN (SELECT category\_id FROM CarCategory) LIMIT 1);

END

CREATE TRIGGER tr\_reservation\_id AFTER INSERT ON Reservation FOR EACH ROW

BEGIN

UPDATE Reservation SET reservation\_id = (SELECT FLOOR(2000.0 + RAND()\*999.0) AS id FROM Reservation WHERE reservation\_id = NULL AND id NOT IN (SELECT reservation\_id FROM Reservation) LIMIT 1);

END

CREATE TRIGGER tr\_rental\_id AFTER INSERT ON Rental FOR EACH ROW

BEGIN

UPDATE Rental SET rental\_id = (SELECT FLOOR(7000.0 + RAND()\*999.0) AS id FROM Rental WHERE rental\_id = NULL AND id NOT IN (SELECT rental\_id FROM Rental) LIMIT 1);

END

**Views:**

CREATE VIEW v\_category\_total\_carcount AS

SELECT CC.category\_id AS CategoryID, COUNT(C.VIN) AS TotalCarCount FROM CarCategory CC LEFT OUTER JOIN Car C ON CC.category\_id=C.category\_id GROUP BY CC.category\_id;

CREATE VIEW v\_category\_available\_carcount AS

SELECT CC.category\_id AS CategoryID, COUNT(C.VIN) AS AvailableCarCount FROM CarCategory CC LEFT OUTER JOIN Car C ON CC.category\_id=C.category\_id WHERE C.availability=1 GROUP BY CC.category\_id;

CREATE VIEW v\_category\_carcount AS

SELECT T.CategoryID, T.TotalCarCount, A.AvailableCarCount, T.TotalCarCount- A.AvailableCarCount AS RentedCarCount FROM v\_category\_total\_carcount T JOIN v\_category\_available\_carcount A ON T.CategoryID=A.CategoryID;

**Inserting Data (manually):**

INSERT INTO CarCategory (category\_name, renting\_price) VALUES

('Economy', 25.00),

('Intermediate', 24.00),

('SUV', 28.00),

('Luxury', 35.00),

('Compact', 23.00),

('Family', 26.00),

('Convertible', 28.00),

('Cargo Van', 34.00),

('Mid-Level', 24.00),

('Roadster', 34.00);

INSERT INTO Car (VIN, category\_id, license\_plate) VALUES

('2KJPDDJH51K971089', 6001, 'LCH-3210'),

('2BNPDDJH51K97100', 6001, 'ABD-768'),

('2WKPDDJH51K970989', 6001, 'BGD-4321'),

('KMHHU6KJ1DU186741', 6001, 'CVF-7865'),

('5GAET13M262340847', 6001, 'NJH-8765'),

('2HGES16535HG81755', 6001, 'SDF-8765'),

('1HD1KAV106Y69592', 6001, 'GHB-7865'),

('5GAET13M262340848', 6001, 'DVG-7653'),

('1HD1KAV106Y695921', 6002, 'BHG-5678'),

('3FRWW6FC3DV072954', 6002, 'BJR-8765'),

('JS1GL52A4D2145863', 6002, 'KKF-7654'),

('2WKPDDJH51K970990', 6002, 'CJG-5678'),

('KMHHU6KJ1DU186742', 6002, 'LKJ-9879'),

('5GAET13M262341848', 6002, 'JHF-5678'),

('2HGES16535HG81756', 6002, 'HHJ-4567'),

('1HD1KAV106Y695922', 6002, 'VGJ-5678'),

('1LNLM9842KY774961', 6003, 'DFG-3456'),

('JS1GL52A4D2145864', 6003, 'CFF-5567'),

('2HGES16535HG81759', 6003, 'CGH-8765'),

('5GAET13M262340851', 6003, 'XCF-4322'),

('2WKPDDJH51K97099', 6003, 'FGB-5676'),

('3FRWW6FC3DV072957', 6003, 'CKK-6787'),

('2HGES16535HG81758', 6004, 'SDF-4456'),

('KMHHU6KJ1DU186744', 6004, 'CFF-5435'),

('KMHHU6KJ1DU186749', 6004, 'VBH-6645'),

('5GAET13M262340852', 6004, 'BGG-8759'),

('2HGES16535HG81760', 6004, 'SDF-8762'),

('2WKPDDJH51K970995', 6004, 'VGH-4221'),

('KMHHU6KJ1DU186747', 6005, 'VHJ-7656'),

('5GAET13M262340853', 6005, 'BHH-5678'),

('2HGES16535HG81761', 6005, 'BHJK-4567'),

('5GAET13M262340854', 6005, 'VHH-5567'),

('2WKPDDJH51K970997', 6005, 'VBJ-6778'),

('5GAET13M262340855', 6005, 'NGG-4567'),

('2HGES16535HG81763', 6006, 'BVF-5679');