TABLE OF CONTENTS

1. Introduction

This project demonstrates a basic car insurance system that allows users to do a variety of car insurance-related tasks, such as entering owner and vehicle information, setting insurance policies, reporting accidents, submitting claims, and creating reports.

This project involves connecting to a database and using stored procedures to define the function of each activity, such as adding owner information, generating insurance policies. It also utilizes triggers to manage insurance status when a claim is filed, approved, or rejected. The database has views that are created to generate reports for car owners, accidents, claims, insurances, etc.

The program used to provide a user interface is written in python as a CLI application. Users can choose their preferred action from a menu, and the appropriate function is executed based on their selection.

The project repository can be found here: [suparthghimire/car-insurance-project (github.com)]

2. Scope

a. Allow users to add information about a car owner and car details.
b. Implement a feature to create insurance for cars.
c. Allow users to report car accidents, providing details like date, time, location, and cause.
d. Enable car owners to file insurance claims for accidents or damages.
e. Provide a mechanism for managing insurance claims.
f. Generate reports that provide information about car owners.
g. Create reports detailing car accidents, including involved vehicles.
h. Generate reports specific to insurance claims and accidents.
i. Produce reports summarizing successful insurance claims.
j. Generate reports that show successful insurance claims grouped by the agent responsible.
k. Automatic updates on insurance status when a claim is filed, approved, or rejected.

3. Creation of ER Diagram

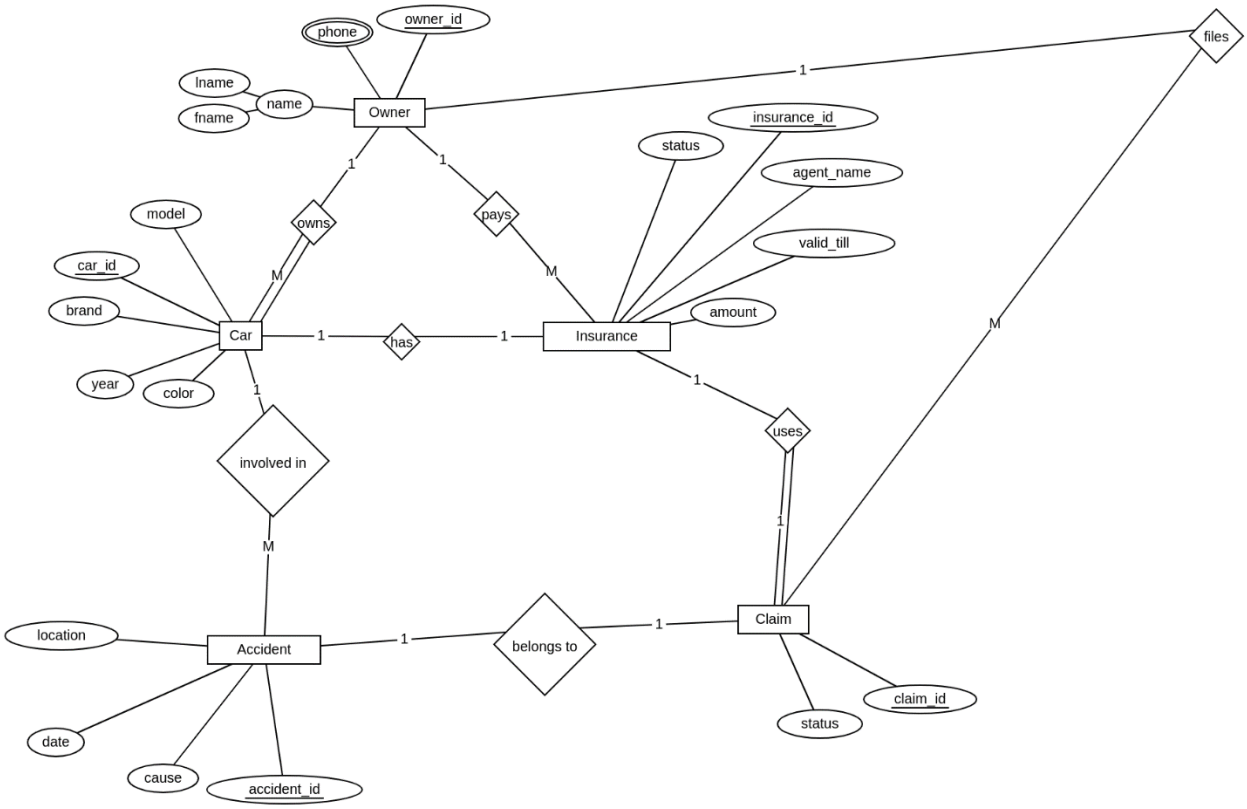


Figure 1 ER Diagram of Car Insurance System

4. Conversion of ER Diagram to Relation

a. Conversion of strong entities: For each strong entity create a separate table with the same name. Include all attributes as relation columns, if there is any composite attribute divide them into simple attributes add them as relation columns. Multivalued attributes are ignored at this stage. A primary key is also selected in this stage.

b. Conversion of weak entity: For each weak entity create a separate table with the same name. Include all attributes as table columns. Include the Primary key of a strong entity as foreign key of the weak entity. Declare the combination of foreign key and decimator attribute as Primary key from the weak entity.

c. Conversion of one-to-one relationship: For each one-to-one relationships take primary key of either relation and include it in another. In case of full participation, entity with full participation must contain the primary key of other relation as foreign key.

d. **Conversion of one-to-many relationship:** For each one-to-many relationships take primary key of relation with relationship cardinality 1 and include it to relation with relation cardinality M as foreign key.

e. **Conversion of many-many relationships:** For each many-to-many relationships make a new table that includes primary key of both tables as its foreign key.

f. **Conversion of multivalued attributes:** For each multivalued attribute create a separate table and include the primary key of the parent relation as foreign key.

g. **Conversion of n-ary relationship:** For each n-ary relationships create a separate table and include the primary key of all participating entities as foreign key. The combination of foreign keys is the primary key of the new relation.

The relations created after conversion are as follows.

Owner

owner_id	fname	lname	phone

Insurance

insurance_id	status	amount	valid_till	agent_name	owner_id

Car

car_id	model	year	brand	color	owner_id	insurance_id

Claim

claim_id	status	owner_id	accident_id	insurance_id

Accident

accident_id	cause	date	location	car_id

5. Normalization

Normalization of relation is done based on normalization degree.

a. **First Normal Form (1NF):**
    i. A relation will be 1NF if it contains an atomic value.
    ii. It states that an attribute of a table cannot hold multiple values. It must hold only single-valued attributes.
    iii. First normal form disallows the multi-valued attribute, composite attribute, and their combinations.
b. **Second Normal Form (2NF):**
    i. In the 2NF, relation must be in 1NF.
    ii. In the second normal form, all non-key attributes are fully functional dependent on the primary-key.
c. **Third Normal Form (3NF):**
    i. It satisfies the First Normal Form and the Second Normal form.
    ii. And, it doesn't have Transitive Dependency.
d. **Boyce Codd Normal Form (BCNF):**
    i. BCNF is the advanced version of 3NF. It is stricter than 3NF.
    ii. A table is in BCNF if every functional dependency X → Y, X is the super key of the table.
    iii. For BCNF, the table should be in 3NF, and for every FD, LHS is a super key.

After normalization for each normal form, the relations are normalized upto BCNF.

Tables after normalization are as follows:

Owner

owner_id	fname	lname

Owner_phone

owner_id	phone

Insurance

insurance_id	status	amount	valid_till	agent_name	paid_by

Car

car_id	model	year	brand	color	owner_id	insurance_id

Accident

accident_id	cause	date	location	car_id

Claim

claim_id	status	filed_by	insurance_id	accident_id

6. Implementation

a. Database

i. Code for Database creation:

```
DROP DATABASE IF EXISTS insurance;
CREATE DATABASE insurance;
USE insurance;
```

ii. Code to create tables:

```
CREATE TABLE tbl_owner(
   owner_id INT AUTO_INCREMENT,
   fname VARCHAR (255),
   lname VARCHAR (255),
   PRIMARY KEY(owner_id)
);
```

```
CREATE TABLE tbl_owner_phone(
   phone BIGINT,
   owner_id INT,
   FOREIGN KEY(owner_id) REFERENCES tbl_owner (owner_id)
);
CREATE TABLE tbl_insurance(
   insurance_id INT AUTO_INCREMENT,
   status ENUM(
      'OPEN',
      'CLAIMED',
      'PENDING',
      'REJECTED'
   ) DEFAULT 'OPEN',
   valid_till DATETIME,
   amount INT,
   agent_name VARCHAR(255),
   paid_by INT,
   FOREIGN KEY(paid_by) REFERENCES tbl_owner (owner_id),
   PRIMARY KEY (insurance_id)
);
CREATE TABLE tbl_car(
   car_id INT AUTO_INCREMENT,
   model VARCHAR(255),
   color VARCHAR(255),
   brand VARCHAR(255),
   year YEAR,
   owner_id INT,
   FOREIGN KEY(owner_id) REFERENCES tbl_owner (owner_id),
   insurance_id INT,
   FOREIGN KEY(insurance_id) REFERENCES tbl_insurance (insurance_id),
   PRIMARY KEY(car_id)
);
CREATE TABLE tbl_accident(
   accident_id INT AUTO_INCREMENT,
   cause TEXT,
   date DATETIME,
   location VARCHAR(255),
   car_id INT,
   FOREIGN KEY(car_id) REFERENCES tbl_car (car_id),
   PRIMARY KEY (accident_id)
);
CREATE TABLE tbl_claim(
   claim_id INT AUTO_INCREMENT,
   status ENUM('PENDING', 'APPROVED','REJECTED') DEFAULT 'PENDING',
   filed_by INT NOT NULL,
   FOREIGN KEY(filed_by) REFERENCES tbl_owner (owner_id),
   insurance_id INT NOT NULL,
```

```
    FOREIGN KEY(insurance_id) REFERENCES tbl_insurance (insurance_id),
    accident_id INT NOT NULL,
    FOREIGN KEY(accident_id) REFERENCES tbl_accident (accident_id),
    PRIMARY KEY (claim_id),
    UNIQUE KEY (insurance_id, accident_id, filed_by)
);
```

iii. Code for Stored Procedures

```
DELIMITER //

DROP PROCEDURE IF EXISTS insert_owner;
CREATE PROCEDURE insert_owner(IN fname VARCHAR(255), IN lname VARCHAR(255))
BEGIN
INSERT INTO tbl_owner(fname, lname)
VALUES(fname, lname);

DROP PROCEDURE IF EXISTS insert_car;
CREATE PROCEDURE insert_car(IN model VARCHAR(255), IN color VARCHAR(255), IN brand VARCHAR(255), IN
year YEAR, IN owner_id INT)
BEGIN
INSERT INTO tbl_car(model, color, brand, year, owner_id)
VALUES(model, color, brand, year, owner_id);

DROP PROCEDURE IF EXISTS report_accident;
CREATE PROCEDURE report_accident(IN cause TEXT, IN date DATETIME, IN location VARCHAR(255), IN car_id
INT)
BEGIN
INSERT INTO tbl_accident(cause, date, location, car_id)
VALUES(cause, date, location, car_id);

DROP PROCEDURE IF EXISTS file_claim;
CREATE PROCEDURE file_claim(IN filed_by INT, IN p_insurance_id INT, IN accident_id INT)
BEGIN
IF (SELECT (SELECT valid_till from tbl_insurance WHERE insurance_id =p_insurance_id ) > NOW()) THEN
INSERT INTO tbl_claim(filed_by, insurance_id, accident_id)
VALUES( filed_by, p_insurance_id, accident_id);
ELSE
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Insurance is not valid';
END IF;

DROP PROCEDURE IF EXISTS approve_claim;
CREATE PROCEDURE approve_claim(IN p_claim_id INT)
BEGIN
UPDATE tbl_claim
SET status = 'APPROVED'
WHERE claim_id = p_claim_id;
```

```
DROP PROCEDURE IF EXISTS reject_claim;
CREATE PROCEDURE reject_claim(IN p_claim_id INT)
BEGIN
UPDATE tbl_claim
SET status = 'REJECTED'
WHERE claim_id = p_claim_id;

DROP PROCEDURE IF EXISTS insert_phone;
CREATE PROCEDURE insert_phone(IN phone BIGINT, IN owner_id INT)
BEGIN
INSERT INTO tbl_owner_phone(phone, owner_id)
VALUES(phone, owner_id);

DROP PROCEDURE IF EXISTS delete_phone;
CREATE PROCEDURE delete_phone(IN phone BIGINT, IN owner_id INT)
BEGIN
DELETE FROM tbl_owner_phone
WHERE phone = phone AND owner_id = owner_id;

END // DELIMITER;
```

iv. Code for Triggers

```
DELIMITER //

DROP TRIGGER IF EXISTS update_insurance_status;
CREATE TRIGGER update_insurance_status
AFTER INSERT ON tbl_claim
FOR EACH ROW
BEGIN
UPDATE tbl_insurance
SET status = 'PENDING'
WHERE insurance_id = NEW.insurance_id;

DROP TRIGGER IF EXISTS update_insurance_status_claimed;
CREATE TRIGGER update_insurance_status_claimed
AFTER UPDATE ON tbl_claim
FOR EACH ROW
BEGIN
IF NEW.status = 'APPROVED' THEN
UPDATE tbl_insurance
SET status = 'CLAIMED'
WHERE insurance_id = NEW.insurance_id;
END IF;

DROP TRIGGER IF EXISTS update_insurance_status_rejected;
```

```
CREATE TRIGGER update_insurance_status_rejected
AFTER UPDATE ON tbl_claim
FOR EACH ROW
BEGIN
IF NEW.status = 'REJECTED' THEN
UPDATE tbl_insurance
SET status = 'REJECTED'
WHERE insurance_id = NEW.insurance_id;
END IF;

END // DELIMITER;
```

v. Code for Views

```
CREATE VIEW vw_cars_owners AS
SELECT c.car_id, c.model, c.color, c.brand, c.year, o.fname, o.lname
FROM tbl_car c
INNER JOIN tbl_owner o
ON c.owner_id = o.owner_id;

CREATE VIEW vw_accidents_cars AS
SELECT a.accident_id, a.cause, a.date, a.location, c.model, c.color, c.brand, c.year
FROM tbl_accident a
INNER JOIN tbl_car c
ON a.car_id = c.car_id;

CREATE VIEW vw_claims_accidents AS
SELECT cl.claim_id, cl.status, cl.filed_by, cl.insurance_id, cl.accident_id, a.cause, a.date, a.location
FROM tbl_claim cl
INNER JOIN tbl_accident a
ON cl.accident_id = a.accident_id;

CREATE VIEW vw_total_claims AS
SELECT i.insurance_id, i.agent_name, COUNT(c.claim_id) AS total_claims
FROM tbl_insurance i
INNER JOIN tbl_claim c
ON i.insurance_id = c.insurance_id
GROUP BY i.insurance_id;

CREATE VIEW vw_successful_claims AS
SELECT i.agent_name, COUNT(
   CASE WHEN i.status = 'CLAIMED' THEN 1 ELSE NULL END
) AS total_claims
FROM tbl_insurance i
GROUP BY i.agent_name;
```

b. Python App

```
import mysql.connector
import os
from dotenv import load_dotenv
import matplotlib.pyplot as plt
# functions
def _exit(_):
    print("Exiting... ")
    exit()
def insertOwner(cursor):
    fname = input("Enter first name: ")
    lname = input("Enter last name: ")
    sql = "CALL insert_owner(%s, %s)"
    val = (fname, lname)
    cursor.execute(sql, val)
    print("Owner inserted")
def insertOwnerPhoneNumber(cursor):
    phone = input("Enter phone number: ")
    owner_id = input("Enter owner id: ")
    sql = "CALL insert_phone(%s, %s)"
    val = (phone, owner_id)
    cursor.execute(sql, val)
    print("Owner phone number inserted")
def deleteOwnerPhoneNumber(cursor):
    phone = input("Enter phone number: ")
    owner_id = input("Enter owner id: ")
    sql = "CALL delete_phone(%s, %s)"
    val = (phone, owner_id)
    cursor.execute(sql, val)
    print("Owner phone number deleted")


def insertCar(cursor):
    model = input("Enter car model: ")
    color = input("Enter car color: ")
    brand = input("Enter car brand: ")
    year = input("Enter car year: ")
    owner_id = input("Enter owner id: ")
    sql = "CALL insert_car(%s, %s, %s, %s, %s)"
    val = (model, color, brand, year, owner_id)
    cursor.execute(sql, val)
    print("Car inserted")
def createInsurance(cursor):
    valid_till = input("Enter valid till date (YYYY-MM-
DD HH:MM:SS): ")
    amount = input("Enter amount: ")
    agent_name = input("Enter agent name: ")
    paid_by = input("Enter paid by: ")
    sql = "CALL create_insurance(%s, %s, %s, %s)"
    val = (valid_till, amount, agent_name, paid_by)
    cursor.execute(sql, val)
    print("Insurance created")
def reportAccident(cursor):
    cause = input("Enter cause: ")
    date = input("Enter date (YYYY-MM-DD
HH:MM:SS): ")
    location = input("Enter location: ")
    car_id = input("Enter car id: ")
    sql = "CALL report_accident(%s, %s, %s, %s)"
    val = (cause, date, location, car_id)
    cursor.execute(sql, val)
    print("Accident reported")
def fileClaim(cursor):
    filed_by = input("Enter filed by: ")
    insurance_id = input("Enter insurance id: ")
    accident_id = input("Enter accident id: ")
    sql = "CALL file_claim(%s, %s, %s)"
    val = (filed_by, insurance_id, accident_id)
    cursor.execute(sql, val)
    print("Claim filed")
def approveClaim(cursor):
    claim_id = input("Enter claim id: ")
    sql = "CALL approve_claim(%s)"
    val = (claim_id,)
    cursor.execute(sql, val)
    print("Claim approved")
def rejectClaim(cursor):
    claim_id = input("Enter claim id: ")
    sql = "CALL reject_claim(%s)"
    val = (claim_id,)
    cursor.execute(sql, val)
def generateCarOwnerReport(cursor):
    sql = "SELECT * FROM vw_cars_owners"
    cursor.execute(sql)
    result = cursor.fetchall()
    print(result)
def generateCarAccidentReport(cursor):
    sql = "SELECT * FROM vw_accidents_cars"
    cursor.execute(sql)
    result = cursor.fetchall()
    print(result)
def generateClaimAccidentReport(cursor):
```

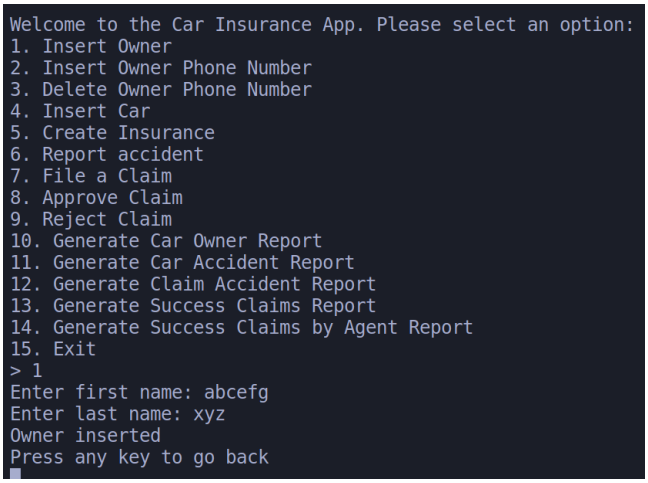
```
    sql = "SELECT * FROM vw_claims_accidents"
    cursor.execute(sql)
    result = cursor.fetchall()
    print(result)
def generateSuccessClaimsReport(cursor):
    sql = "SELECT * FROM vw_total_claims"
    cursor.execute(sql)
    result = cursor.fetchall()
    print(result)
def generateSuccessClaimsByAgentReport(cursor):
    sql = "SELECT * FROM vw_successful_claims"
    cursor.execute(sql)
    result = cursor.fetchall()
    print(result)
# app
menus = [
    {"label": "Insert Owner", "action": insertOwner},
    {"label": "Insert Owner Phone Number", "action":
insertOwnerPhoneNumber},
    {"label": "Delete Owner Phone Number", "action":
deleteOwnerPhoneNumber},
    {"label": "Insert Car", "action": insertCar},
    {"label": "Create Insurance", "action":
createInsurance},
    {"label": "Report accident", "action":
reportAccident},
    {"label": "File a Claim", "action": fileClaim},
    {"label": "Approve Claim", "action":
approveClaim},
    {"label": "Reject Claim", "action": rejectClaim},
    {"label": "Generate Car Owner Report", "action":
generateCarOwnerReport},
    {"label": "Generate Car Accident Report", "action":
generateCarAccidentReport},
    {"label": "Generate Claim Accident Report",
"action": generateClaimAccidentReport},
    {"label": "Generate Success Claims Report",
"action": generateSuccessClaimsReport},
    {"label": "Generate Success Claims by Agent
Report", "action":
generateSuccessClaimsByAgentReport},
    {"label": "Exit", "action": _exit},
]
def menu():
    print ("Welcome to the Car Insurance App. Please
select an option:")
    for i in range(len(menus)):
        print(str(i+1) + ". " + menus[i]["label"])
    option = input("> ")
    return int(option)
def back():
    print("Press any key to go back")
    input()
    return
def main():
    load_dotenv()
    # Connect to database
    db_host = os.environ.get('DB_HOST')
    db_username = os.environ.get('DB_USERNAME')
    db_password = os.environ.get('DB_PASSWORD')
    db_name = os.environ.get('DB_NAME')
    try:
        db = mysql.connector.connect(
            host=db_host,
            user=db_username,
            password=db_password,
            database=db_name
        )
        cursor = db.cursor()
        print("DATABASE CONNECTED " +
str(db.is_connected()))
        while(True):
            try:
            # clear screen and show menu
                os.system('cls' if os.name == 'nt' else 'clear')
                option = menu()
                # see if option is valid
                if len(menus) < option or option < 1:
                    print("Invalid option")
                    back()
                    continue
                selected_menu = menus[option-1]
                #execute the action
                selected_menu["action"](cursor)
                db.commit()
                back()
            except Exception as e:
                print("Error: " + str(e))
                back()
                continue
    except Exception as e:
        print("Error: " + str(e))
main(
```

7. Conclusion

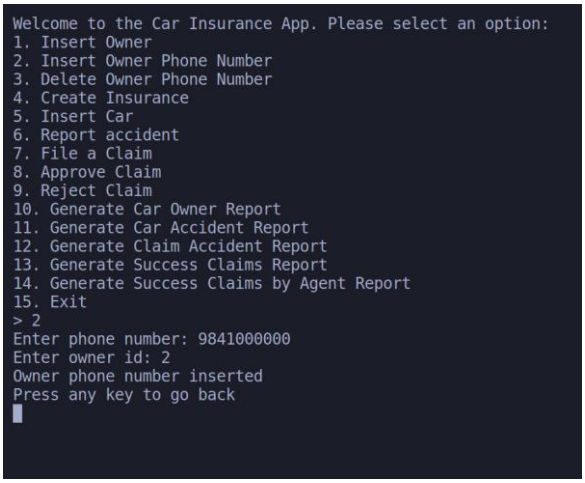
In conclusion, this project represents a successful integration of database management, stored procedures, triggers, and a user interface to create a functional car Users can easily perform essential tasks, such as inputting owner and vehicle details, reporting accidents, submitting claims and generating informative reports. Additionally, it showcases the practical application of database normalization principles to ensure data integrity reduction of redundancy. This project acts as a solid foundation that can be further enhanced and customized to meet the specific needs of insurance providers.

8. Result

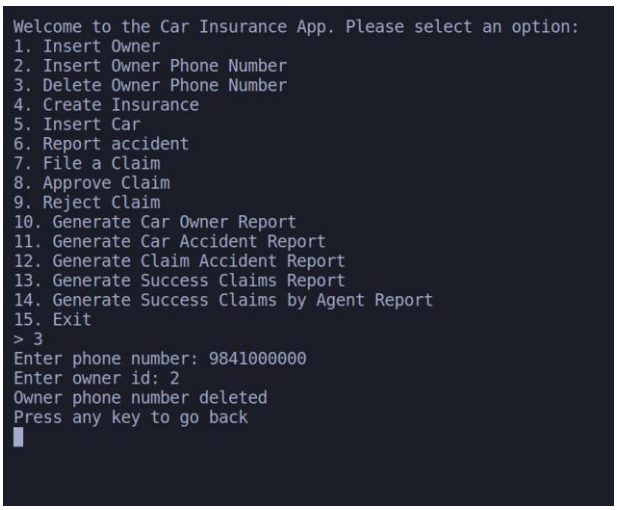
1. Insert Owner



2. Insert Owner Phone



3. Delete Owner Phone



4. Create Insurance

5. Insert Car

```
Welcome to the Car Insurance App. Please select an option:
1. Insert Owner
2. Insert Owner Phone Number
3. Delete Owner Phone Number
4. Create Insurance
5. Insert Car
6. Report accident
7. File a Claim
8. Approve Claim
9. Reject Claim
10. Generate Car Owner Report
11. Generate Car Accident Report
12. Generate Claim Accident Report
13. Generate Success Claims Report
14. Generate Success Claims by Agent Report
15. Exit
> 5
Enter car model: racing
Enter car color: black
Enter car brand: bmw
Enter car year: 2023
Enter owner id: 2
Enter insurance id: 2
Car inserted
Press any key to go back
```

6. Report Accident

```
Welcome to the Car Insurance App. Please select an option:
1. Insert Owner
2. Insert Owner Phone Number
3. Delete Owner Phone Number
4. Create Insurance
5. Insert Car
6. Report accident
7. File a Claim
8. Approve Claim
9. Reject Claim
10. Generate Car Owner Report
11. Generate Car Accident Report
12. Generate Claim Accident Report
13. Generate Success Claims Report
14. Generate Success Claims by Agent Report
15. Exit
> 6
Enter cause: Drink and Drive
Enter date (YYYY-MM-DD HH:MM:SS): 2023-09-12 00:13:00
Enter location: Kalanki
Enter car id: 2
Accident reported
Press any key to go back
```

7. File a Claim

```
Welcome to the Car Insurance App. Please select an option:
1. Insert Owner
2. Insert Owner Phone Number
3. Delete Owner Phone Number
4. Create Insurance
5. Insert Car
6. Report accident
7. File a Claim
8. Approve Claim
9. Reject Claim
10. Generate Car Owner Report
11. Generate Car Accident Report
12. Generate Claim Accident Report
13. Generate Success Claims Report
14. Generate Success Claims by Agent Report
15. Exit
> 7
Enter filed by: 2
Enter insurance id: 2
Enter accident id: 2
Claim filed
Press any key to go back
```

8. Approve Claim

```
Welcome to the Car Insurance App. Please select an option:
1. Insert Owner
2. Insert Owner Phone Number
3. Delete Owner Phone Number
4. Create Insurance
5. Insert Car
6. Report accident
7. File a Claim
8. Approve Claim
9. Reject Claim
10. Generate Car Owner Report
11. Generate Car Accident Report
12. Generate Claim Accident Report
13. Generate Success Claims Report
14. Generate Success Claims by Agent Report
15. Exit
> 8
Enter claim id: 2
Claim approved
Press any key to go back
```

9. Reject Claim

```
Welcome to the Car Insurance App. Please select an option:
1. Insert Owner
2. Insert Owner Phone Number
3. Delete Owner Phone Number
4. Create Insurance
5. Insert Car
6. Report accident
7. File a Claim
8. Approve Claim
9. Reject Claim
10. Generate Car Owner Report
11. Generate Car Accident Report
12. Generate Claim Accident Report
13. Generate Success Claims Report
14. Generate Success Claims by Agent Report
15. Exit
> 9
Enter claim id: 1
Press any key to go back
```

10. Car Owner Report

```
Welcome to the Car Insurance App. Please select an optior
1. Insert Owner
2. Insert Owner Phone Number
3. Delete Owner Phone Number
4. Create Insurance
5. Insert Car
6. Report accident
7. File a Claim
8. Approve Claim
9. Reject Claim
10. Generate Car Owner Report
11. Generate Car Accident Report
12. Generate Claim Accident Report
13. Generate Success Claims Report
14. Generate Success Claims by Agent Report
15. Exit
> 10
(1, 'Model 3', 'Black', 'Tesla', 2021, 'John', 'Doe')
(2, 'racing', 'black', 'bmw', 2023, 'Peter', 'Parker')
Press any key to go back
```

11. Car Accident Report

```
Welcome to the Car Insurance App. Please select an option:
1. Insert Owner
2. Insert Owner Phone Number
3. Delete Owner Phone Number
4. Create Insurance
5. Insert Car
6. Report accident
7. File a Claim
8. Approve Claim
9. Reject Claim
10. Generate Car Owner Report
11. Generate Car Accident Report
12. Generate Claim Accident Report
13. Generate Success Claims Report
14. Generate Success Claims by Agent Report
15. Exit
> 11
(1, 'Rear Ended', datetime.datetime(2021, 9, 1, 0, 0), 'New York', 'Model 3', 'Black', 'Tesla', 2021)
(2, 'Drink and Drive', datetime.datetime(2023, 9, 12, 0, 13), 'Kalanki', 'racing', 'black', 'bmw', 2023)
Press any key to go back
```

12. Total Claim Report

```
Welcome to the Car Insurance App. Please select an option:
1. Insert Owner
2. Insert Owner Phone Number
3. Delete Owner Phone Number
4. Create Insurance
5. Insert Car
6. Report accident
7. File a Claim
8. Approve Claim
9. Reject Claim
10. Generate Car Owner Report
11. Generate Car Accident Report
12. Generate Claim Accident Report
13. Generate Success Claims Report
14. Generate Success Claims by Agent Report
15. Exit
> 13
(1, 'John Doe', 1)
(2, 'Mary Jane', 1)
Press any key to go back
```

13. Success Claim Report by Agent

```
Welcome to the Car Insurance App. Please select an option:
1. Insert Owner
2. Insert Owner Phone Number
3. Delete Owner Phone Number
4. Create Insurance
5. Insert Car
6. Report accident
7. File a Claim
8. Approve Claim
9. Reject Claim
10. Generate Car Owner Report
11. Generate Car Accident Report
12. Generate Claim Accident Report
13. Generate Success Claims Report
14. Generate Success Claims by Agent Report
15. Exit
> 14
('John Doe', 0)
('Mary Jane', 1)
Press any key to go back
```