2021

DATABASE PROJECT FOR CELL PHONE REPAIR STORE

DATABASE

Author: Nwaobiora Faustina Eberechukwu
Reviewer: Prof. Dr. Ulrike Herster
Date: 25th June 2021.

Table Of Contents

Part 1:

Part 2:

Part 3:

DATABASE DESIGN FOR CELL PHONE REPAIR STORE

PART 1

1.1 INTRODUCTION

An organized file for storing data is known as database. A database can be easily accessed and developed through proper modelling, complex design, and techniques. To communicate with the database, we need a Database Management System (DBMS), that creates a bridge between the end user and data. MySql is an example of this DBMS, for this project MySQL was used to design the cell phone repair store.

1.2 SHORT DESCRIPTION

The database is specifically for a particular store. That has several departments, and each departments engage a lot of employees. An employee can work on multiple phones or order. A customer can have multiple phones submitted for repairs.

This is system that helps to track all the phone in the store. This application helps displays all damaged phone together with their brief problem description, due date for repair and allows for adding, removing, editing the phones and helps to keep track of the time.

1.3 USE CASES

1. To add new customer repair
2. To generate invoice.
3. To add, remove or modify damaged phone from the store.
4. To Collect repaired phone from the store.
5. To track all phones in the store and when they are repaired.
6. To pay for services render via any method.
7. To track employees working on a phone

PART 2

2.1 ER DIAGRAM:

The diagram illustrates the relationship between the tables in the database system. The entity relationship diagram depicted in Figure a show the entities (tables), attributes (fields) and their relationships for the cell phone repair database. The database contains eight data tables in total.

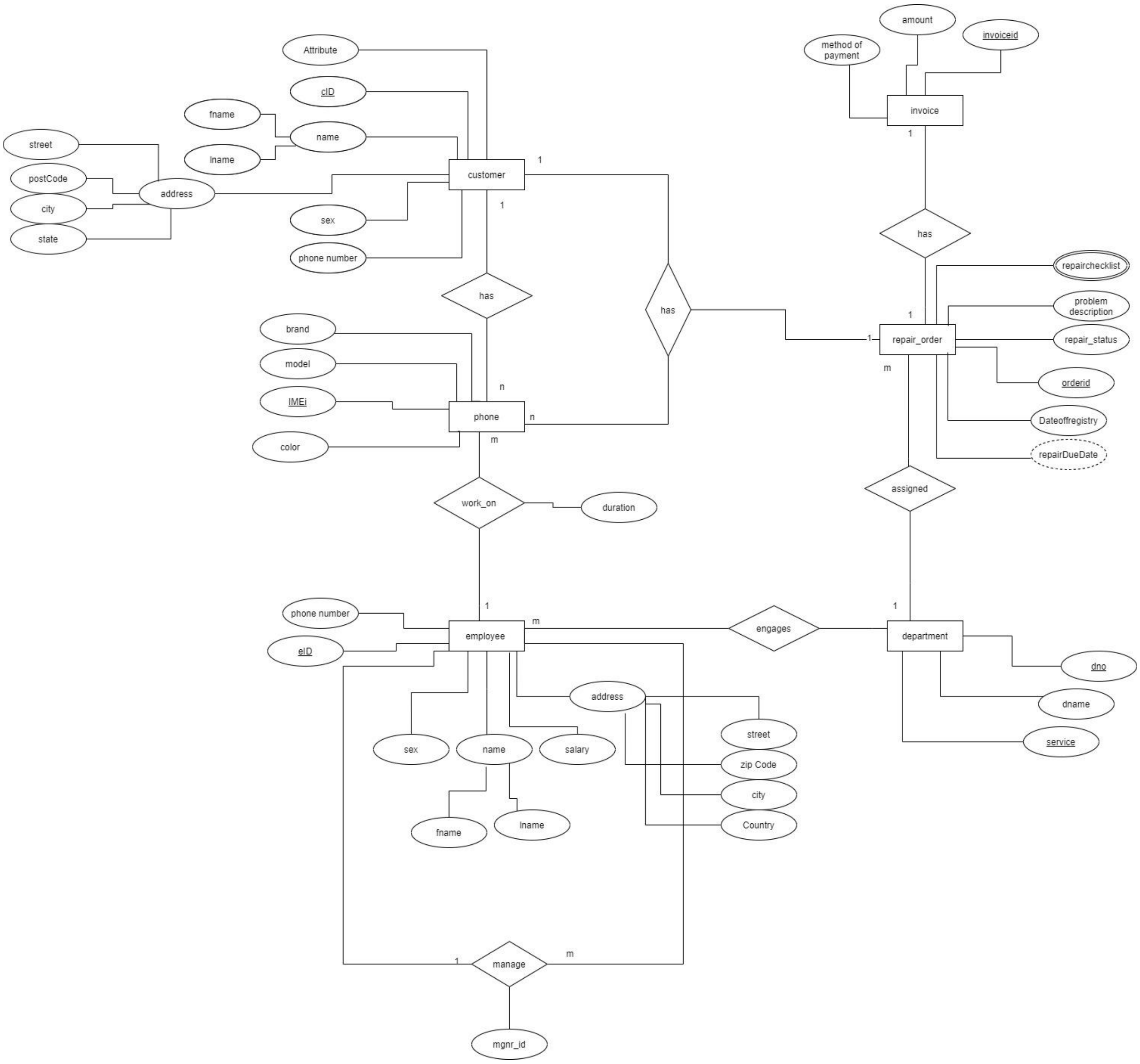


Fig a:ER diagram of cell phone repair store

2.2 MAPPING OF ER DIGRAM TO TABLE

To map the ER diagram, we go through seven steps which includes:

a) Mapping of regular entity types
b) Mapping of weak entity type
c) Mapping of binary 1:1 relationship
d) Mapping of binary 1:m relationship
e) Mapping of binary m: n relationship
f) Mapping of multivalued attributes
g) Mapping of n-ary relationship

CUSTOMER

customerID (PK)	fname	lname	sex	Phonenumber	Streetname	postCode	city	streetno	state

PHONE

IMEi(Pk)	Model	brand	color	empID(FK)	CusID(Fk)	duration

EMPLOYEE

Eid (Pk)	fname	lname	sex	Phone number	salary	street	postcode	city	state	Dno(fK)	MgnrID (FK)

DEPARTMENT

deptID (Pk)	Dname	service

REPAIR_ ORDER

Orderid (PK)	repairStatus	repairDueDate	DateRegister	ProblemDescription	Dnumber(FK)

INVOICE

Invoiceid(PK)	amount	methodOfPayment	repairOrderID(FK)

REPAIR_CHECKLIST

Orderid (PK)(FK)	Parts (PK)

REQUEST

cID(Pk)(Fk)	IMEi(Pk)(Fk)	orderID(Pk)(Fk)

2.3 NORMALIZATION

This is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies. We have the first normal form(1NF), second normal form(2NF), third normal form(3NF). For each of the table created, it will be normalized it to the 3NF.

a) **First Normal Form (1NF):** In 1NF the table must consist of only atomic attributes. That is, no composite attribute is allowed.
b) **Second Normal Form (2NF):** In 2NF the table must not contain any partial dependency and it must be in the 1NF.
c) **Third Normal Form (3NF):** In 3NF the table must not contain any transitive dependency and it must be in the 2NF.

Functional Dependency of Each table

1. Customer
   CustomerID → fname,lname,phone number,sex,post code,street,state,city
   PostCode → state,city

CUSTOMER

customerID (PK)	fname	lname	sex	Phonenumber	Streetname	streetno	postcode (FK)	Email_ Address

ADDRESS

Postcode (PK)	city	state

2. Department
   deptID → deptname, Service

3. Employee
   empID → fname, lname, sex, postcode, phone number, street,city,salary, mgnr_id ,dnumber
   PostCode → state,city

Normalized table:

EMPLOYEE

Eid (Pk)	fname	lname	sex	Phone number	salary	streename	streetno	Postcode(FK)	Dno(fK)	MgnrID (FK)

ADDRESS

Postcode (PK)	city	state

4. Invoice
   invoiceID → amount, method of Payment, dateDelivery, repairOrderID

5. Repair_ order
   Orderid → repair status, repair due date, date of register, problem description,Dnumber.

6. Phone
   IMEi → brand, model, colour,empID,cusID

In summary, all the tables mapped from the ER diagram are all in the 1NF because the tables consist of atomic attributes. From the functional dependency we do not have any partial dependency. This implies that all the tables are in the 2NF.For the customer and employee table we had a transitive dependency, which was normalized. Now all the tables are in 3NF.

3.1 CREATING RELATIONS IN MYSQL WITH DATA DEFINITION LANGUAGE(DDL)

Data Definition Language (DDL): The DDL is a SQL use for creating and specifying database schema example: create, delete, alter, truncate.

CREATED SCHEMA AND TABLES

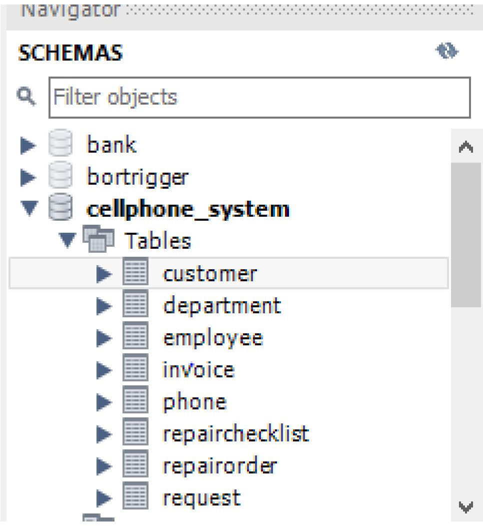
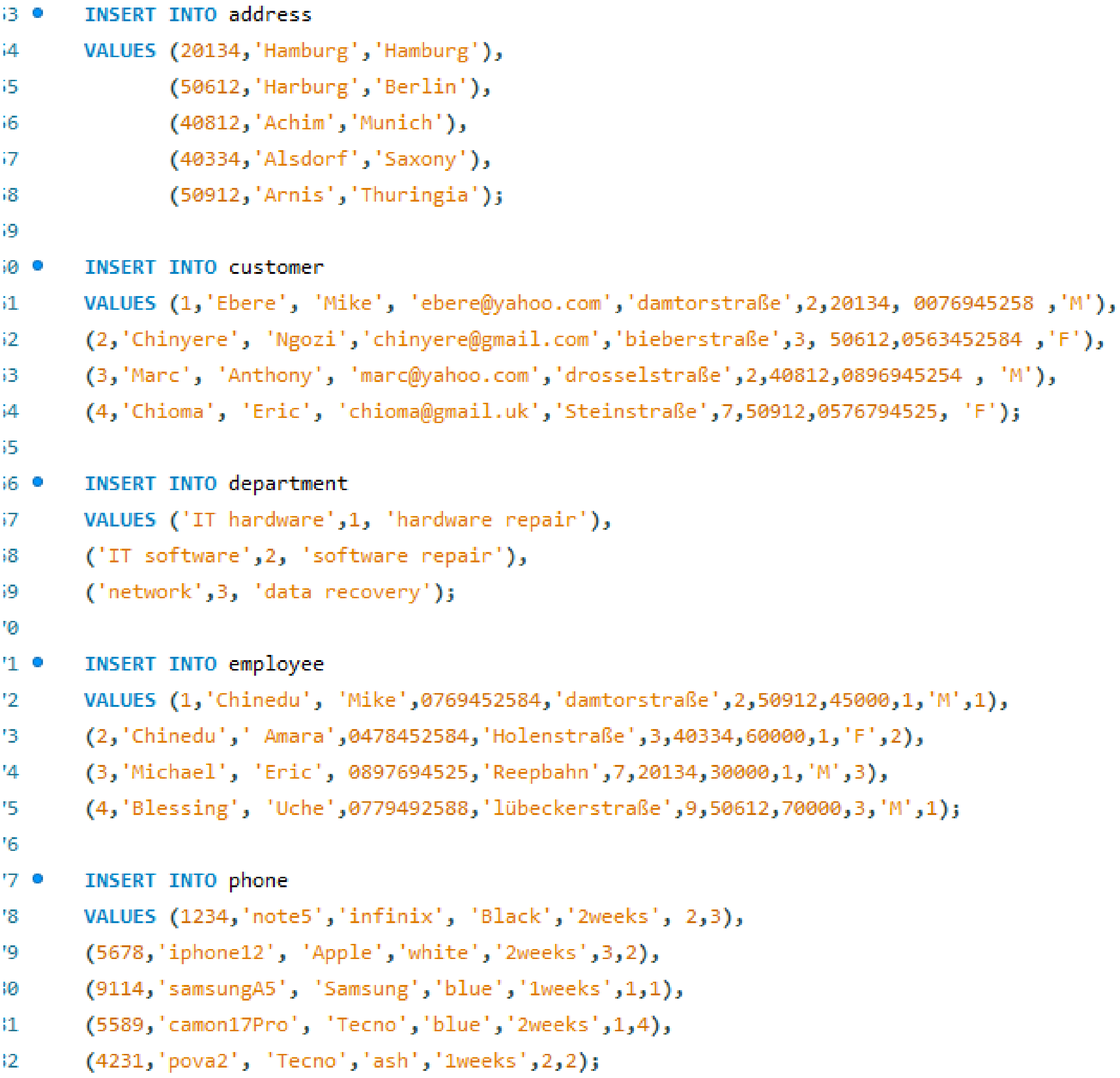


Fig b: Tables in the schema

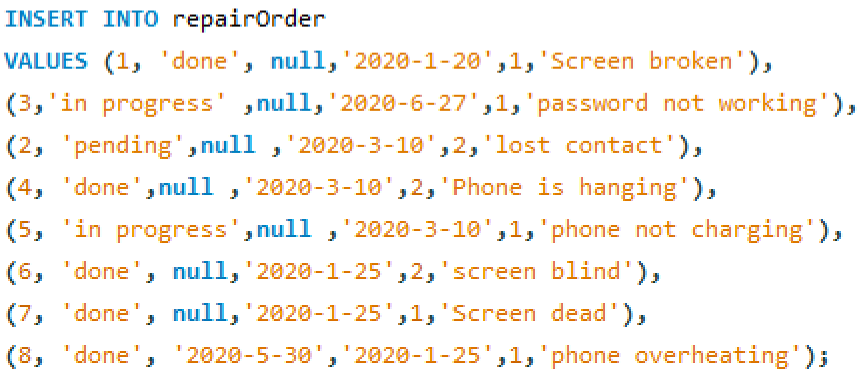
3.2 INSERTING VALUES INTO THE TABLES

Using the insert keyword, I inserted values into the various tables that I have in Schema.

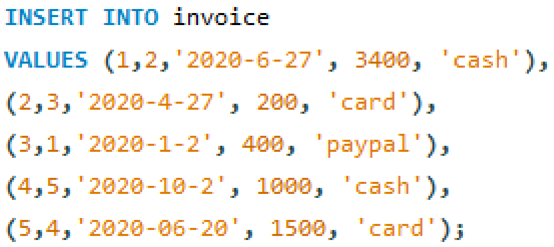


```
INSERT INTO address
VALUES (20134,'Hamburg','Hamburg'),
       (50612,'Harburg','Berlin'),
       (40812,'Achim','Munich'),
       (40334,'Alsdorf','Saxony'),
       (50912,'Arnis','Thuringia');

INSERT INTO customer
VALUES (1,'Ebere', 'Mike', 'ebere@yahoo.com','damtorstraße',2,20134, 0076945258 ,'M'),
(2,'Chinyere', 'Ngozi','chinyere@gmail.com','bieberstraße',3, 50612,0563452584 ,'F'),
(3,'Marc', 'Anthony', 'marc@yahoo.com','drosselstraße',2,40812,0896945234 , 'M'),
(4,'Chioma', 'Eric', 'chioma@gmail.uk','Steinstraße',7,50912,0576794525, 'F');

INSERT INTO department
VALUES ('IT hardware',1, 'hardware repair'),
('IT software',2, 'software repair'),
('network',3, 'data recovery');

INSERT INTO employee
VALUES (1,'Chinedu', 'Mike',0769452584,'damtorstraße',2,50912,45000,1,'M',1),
(2,'Chinedu',' Amara',0478452584,'Holenstraße',3,40334,60000,1,'F',2),
(3,'Michael', 'Eric', 0897694525,'Reepbahn',7,20134,30000,1,'M',3),
(4,'Blessing', 'Uche',0779492588,'lübeckerstraße',9,50612,70000,3,'M',1);

INSERT INTO phone
VALUES (1234,'note5','infinix', 'Black','2weeks', 2,3),
(5678,'iphone12', 'Apple','white','2weeks',3,2),
(9114,'samsungA5', 'Samsung','blue','1weeks',1,1),
(5589,'camon17Pro', 'Tecno','blue','2weeks',1,4),
(4231,'pova2', 'Tecno','ash','1weeks',2,2);
```

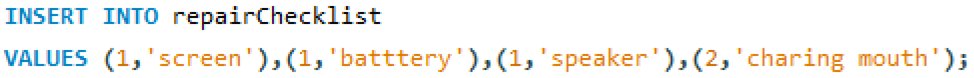
Fig c1: insert query.

- 

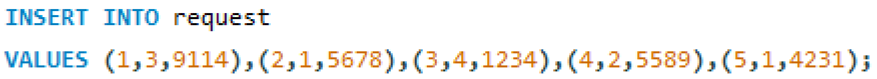
```
INSERT INTO repairOrder
VALUES (1, 'done', null,'2020-1-20',1,'Screen broken'),
(3,'in progress' ,null,'2020-6-27',1,'password not working'),
(2, 'pending',null ,'2020-3-10',2,'lost contact'),
(4, 'done',null ,'2020-3-10',2,'Phone is hanging'),
(5, 'in progress',null ,'2020-3-10',1,'phone not charging'),
(6, 'done', null,'2020-1-25',2,'screen blind'),
(7, 'done', null,'2020-1-25',1,'Screen dead'),
(8, 'done', '2020-5-30','2020-1-25',1,'phone overheating');
```

- 

```
INSERT INTO invoice
VALUES (1,2,'2020-6-27', 3400, 'cash'),
(2,3,'2020-4-27', 200, 'card'),
(3,1,'2020-1-2', 400, 'paypal'),
(4,5,'2020-10-2', 1000, 'cash'),
(5,4,'2020-06-20', 1500, 'card');
```

- 

```
INSERT INTO repairChecklist
VALUES (1,'screen'),(1,'batttery'),(1,'speaker'),(2,'charing mouth');
```

- 

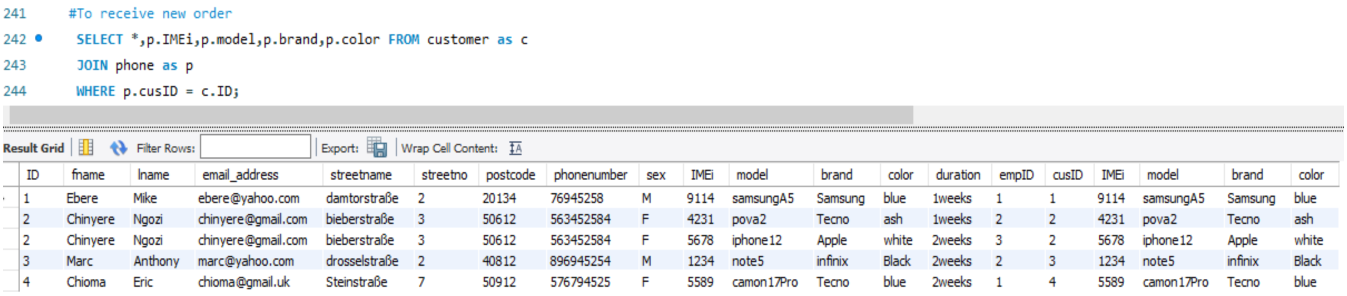
```
INSERT INTO request
VALUES (1,3,9114),(2,1,5678),(3,4,1234),(4,2,5589),(5,1,4231);
```

Fig c2: insert query.

3.3 QUERIES FOR THE USED CASES

The data contained in the tables of the database can be queried to manipulate and process the data into meaningful information. There are several different query operations that can be easily created in access that will manipulate the data in different ways. This section provides basic queries from the cell phone repair database.

To receive new order

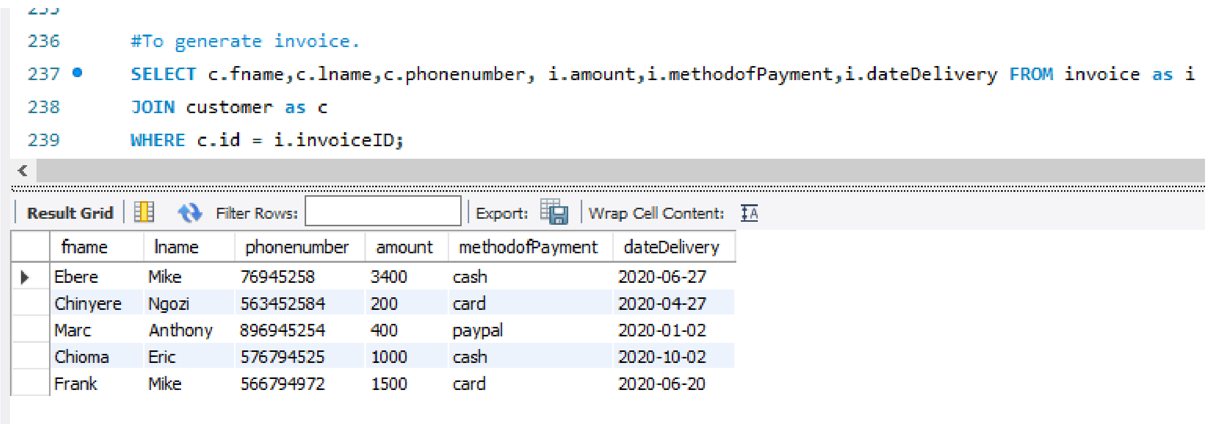


```
#To receive new order
SELECT *,p.IMEi,p.model,p.brand,p.color FROM customer as c
JOIN phone as p
WHERE p.cusID = c.ID;
```

ID	fname	lname	email_address	streetname	streetno	postcode	phonenumber	sex	IMEi	model	brand	color	duration	empID	cusID	IMEi	model	brand	color
1	Ebere	Mike	ebere@yahoo.com	damtorstraße	2	20134	76945258	M	9114	samsungA5	Samsung	blue	1weeks	1	1	9114	samsungA5	Samsung	blue
2	Chinyere	Ngozi	chinyere@gmail.com	bieberstraße	3	50612	563452584	F	4231	pova2	Tecno	ash	1weeks	2	2	4231	pova2	Tecno	ash
2	Chinyere	Ngozi	chinyere@gmail.com	bieberstraße	3	50612	563452584	F	5678	iphone12	Apple	white	2weeks	3	2	5678	iphone12	Apple	white
3	Marc	Anthony	marc@yahoo.com	drosselstraße	2	40812	896945254	M	1234	note5	infinix	Black	2weeks	2	3	1234	note5	infinix	Black
4	Chioma	Eric	chioma@gmail.uk	Steinstraße	7	50912	576794525	F	5589	camon17Pro	Tecno	blue	2weeks	1	4	5589	camon17Pro	Tecno	blue

Fig d: table of all phones and employee working on it.

To generate invoice.

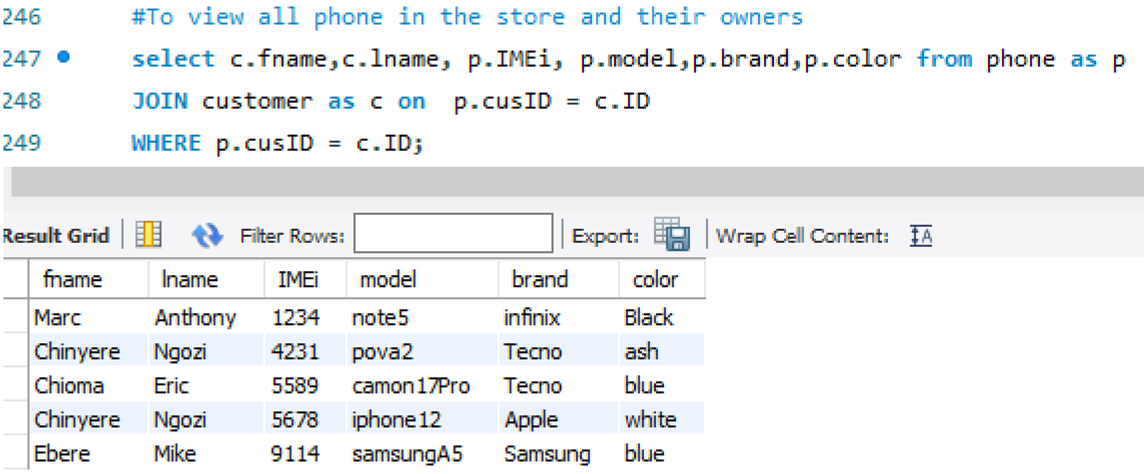


```
#To generate invoice.
SELECT c.fname,c.lname,c.phonenumber, i.amount,i.methodofPayment,i.dateDelivery FROM invoice as i
JOIN customer as c
WHERE c.id = i.invoiceID;
```

fname	lname	phonenumber	amount	methodofPayment	dateDelivery
Ebere	Mike	76945258	3400	cash	2020-06-27
Chinyere	Ngozi	563452584	200	card	2020-04-27
Marc	Anthony	896945254	400	paypal	2020-01-02
Chioma	Eric	576794525	1000	cash	2020-10-02
Frank	Mike	566794972	1500	card	2020-06-20

Fig e: customer table

To view all phones in the store and their owners

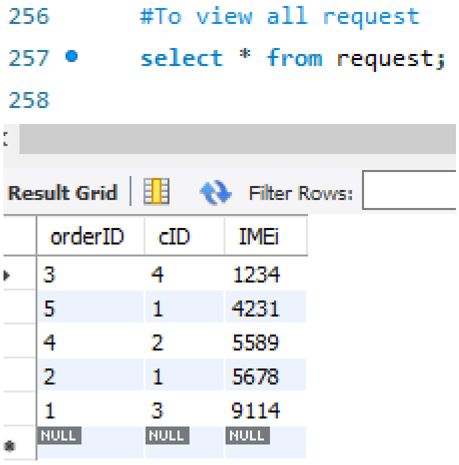


```
#To view all phone in the store and their owners
select c.fname,c.lname, p.IMEi, p.model,p.brand,p.color from phone as p
JOIN customer as c on  p.cusID = c.ID
WHERE p.cusID = c.ID;
```

fname	lname	IMEi	model	brand	color
Marc	Anthony	1234	note5	infinix	Black
Chinyere	Ngozi	4231	pova2	Tecno	ash
Chioma	Eric	5589	camon17Pro	Tecno	blue
Chinyere	Ngozi	5678	iphone12	Apple	white
Ebere	Mike	9114	samsungA5	Samsung	blue

Fig f: all phones and their owners

To View all request.

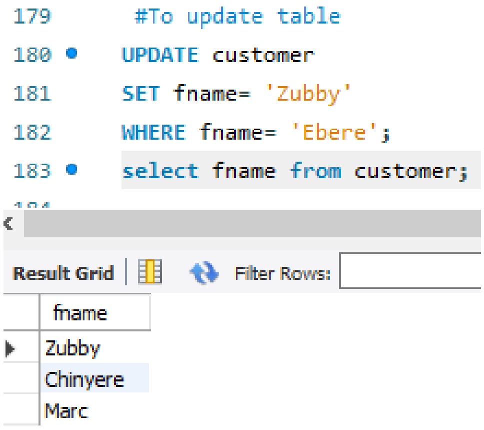


```
#To view all request
select * from request;
```

orderID	cID	IMEi
3	4	1234
5	1	4231
4	2	5589
2	1	5678
1	3	9114
NULL	NULL	NULL

Fig g: all request in the store

To Update



```
#To update table
UPDATE customer
SET fname= 'Zubby'
WHERE fname= 'Ebere';
select fname from customer;
```

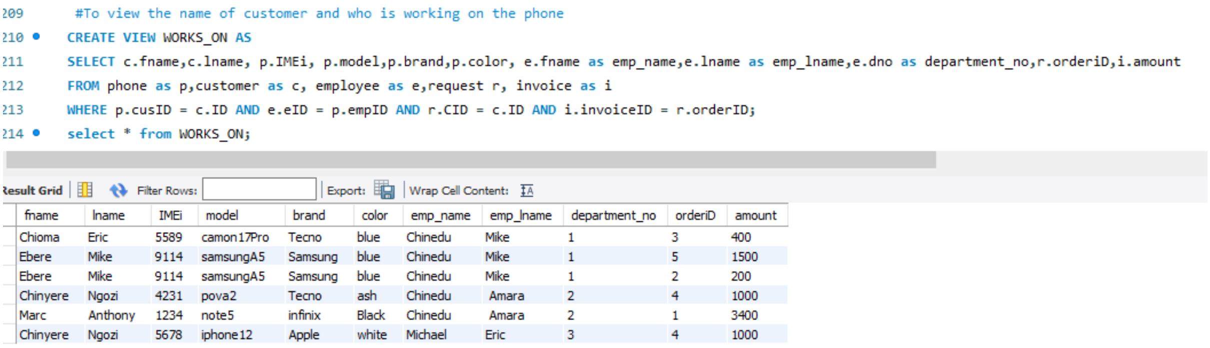
fname
Zubby
Chinyere
Marc

Fig h: updated table

3.4 VIEWS

View can be seen as a virtual table that contains data from one or multiple tables. It does not hold any data and does not exist physically in the database. A view contains rows and columns, just like a real table. I have created some views for easy access.

View for all phone and who is working on it.

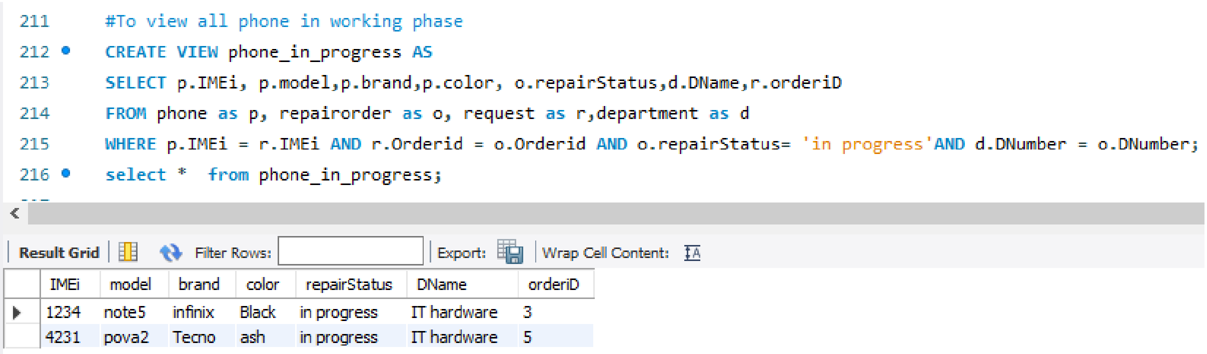


```
 #To view the name of customer and who is working on the phone
CREATE VIEW WORKS_ON AS
SELECT c.fname,c.lname, p.IMEi, p.model,p.brand,p.color, e.fname as emp_name,e.lname as emp_lname,e.dno as department_no,r.orderiD,i.amount
FROM phone as p,customer as c, employee as e,request r, invoice as i
WHERE p.cusID = c.ID AND e.eID = p.empID AND r.CID = c.ID AND i.invoiceID = r.orderID;
select * from WORKS_ON;
```

fname	lname	IMEi	model	brand	color	emp_name	emp_lname	department_no	orderiD	amount
Chioma	Eric	5589	camon17Pro	Tecno	blue	Chinedu	Mike	1	3	400
Ebere	Mike	9114	samsungA5	Samsung	blue	Chinedu	Mike	1	5	1500
Ebere	Mike	9114	samsungA5	Samsung	blue	Chinedu	Mike	1	2	200
Chinyere	Ngozi	4231	pova2	Tecno	ash	Chinedu	Amara	2	4	1000
Marc	Anthony	1234	note5	infinix	Black	Chinedu	Amara	2	1	3400
Chinyere	Ngozi	5678	iphone12	Apple	white	Michael	Eric	3	4	1000

Fig i: works on view.

View for all phone is working phase and department working on it.

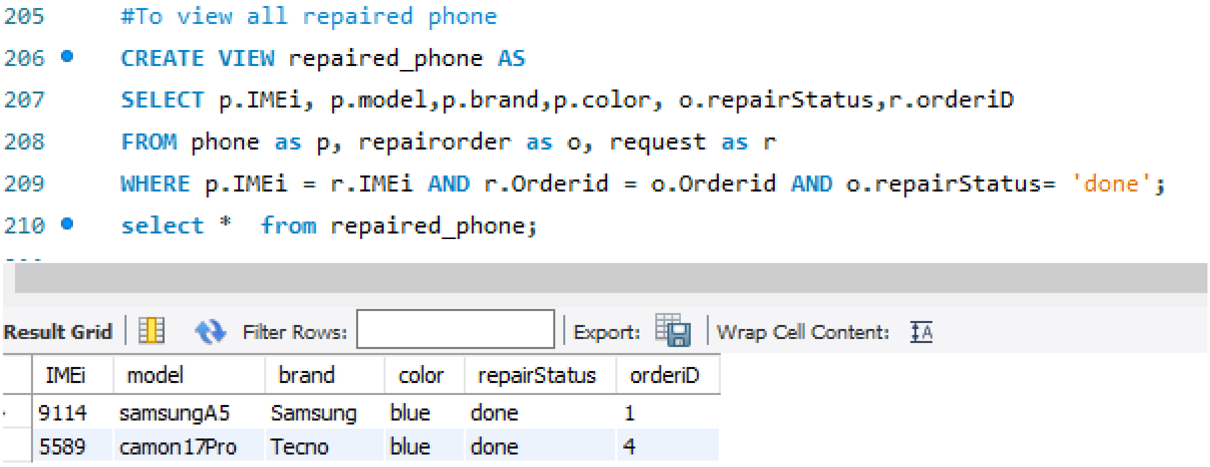


```
#To view all phone in working phase
CREATE VIEW phone_in_progress AS
SELECT p.IMEi, p.model,p.brand,p.color, o.repairStatus,d.DName,r.orderiD
FROM phone as p, repairorder as o, request as r,department as d
WHERE p.IMEi = r.IMEi AND r.Orderid = o.Orderid AND o.repairStatus= 'in progress'AND d.DNumber = o.DNumber;
select *  from phone_in_progress;
```

IMEi	model	brand	color	repairStatus	DName	orderiD
1234	note5	infinix	Black	in progress	IT hardware	3
4231	pova2	Tecno	ash	in progress	IT hardware	5

Fig j: phones in progress view.

View for all phone that are repaired.

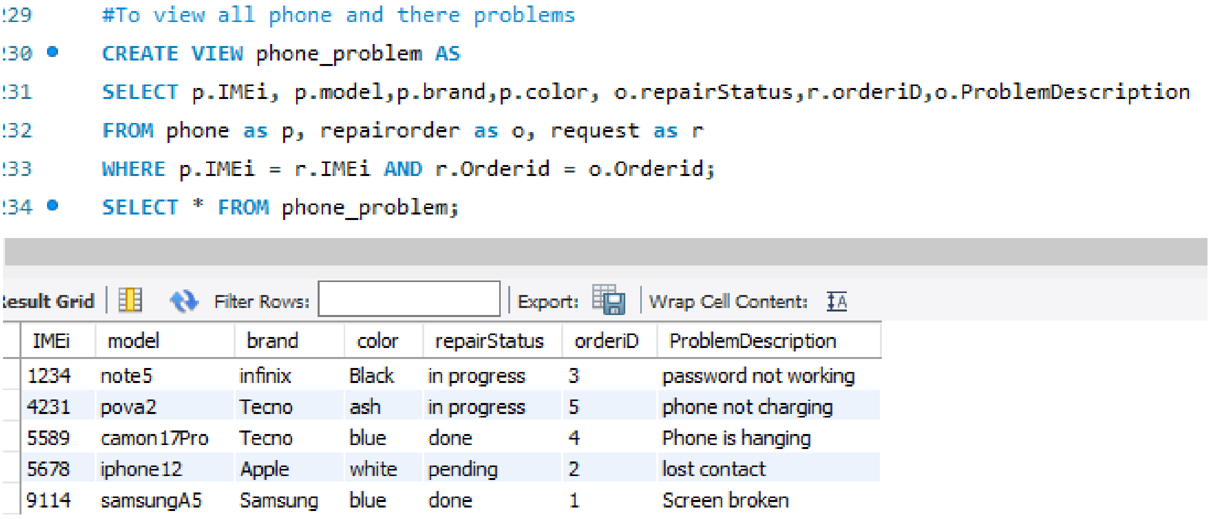


```
#To view all repaired phone
CREATE VIEW repaired_phone AS
SELECT p.IMEi, p.model,p.brand,p.color, o.repairStatus,r.orderiD
FROM phone as p, repairorder as o, request as r
WHERE p.IMEi = r.IMEi AND r.Orderid = o.Orderid AND o.repairStatus= 'done';
select *  from repaired_phone;
```

IMEi	model	brand	color	repairStatus	orderiD
9114	samsungA5	Samsung	blue	done	1
5589	camon17Pro	Tecno	blue	done	4

Fig k: repaired phones view.

View for all phones and their problems



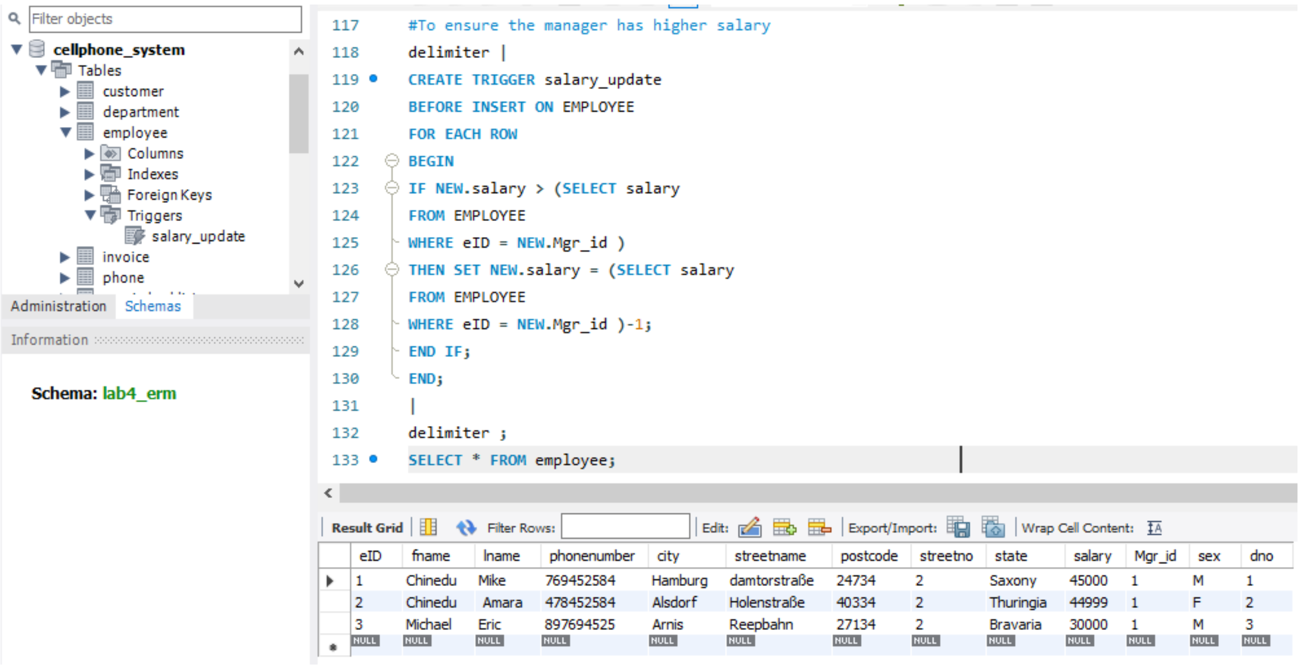
```
#To view all phone and there problems
CREATE VIEW phone_problem AS
SELECT p.IMEi, p.model,p.brand,p.color, o.repairStatus,r.orderiD,o.ProblemDescription
FROM phone as p, repairorder as o, request as r
WHERE p.IMEi = r.IMEi AND r.Orderid = o.Orderid;
SELECT * FROM phone_problem;
```

IMEi	model	brand	color	repairStatus	orderiD	ProblemDescription
1234	note5	infinix	Black	in progress	3	password not working
4231	pova2	Tecno	ash	in progress	5	phone not charging
5589	camon17Pro	Tecno	blue	done	4	Phone is hanging
5678	iphone12	Apple	white	pending	2	lost contact
9114	samsungA5	Samsung	blue	done	1	Screen broken

Fig l: repaired phones view.

3.5 TRIGGERS

This trigger helps ensure that the salary of the manager is always greater.



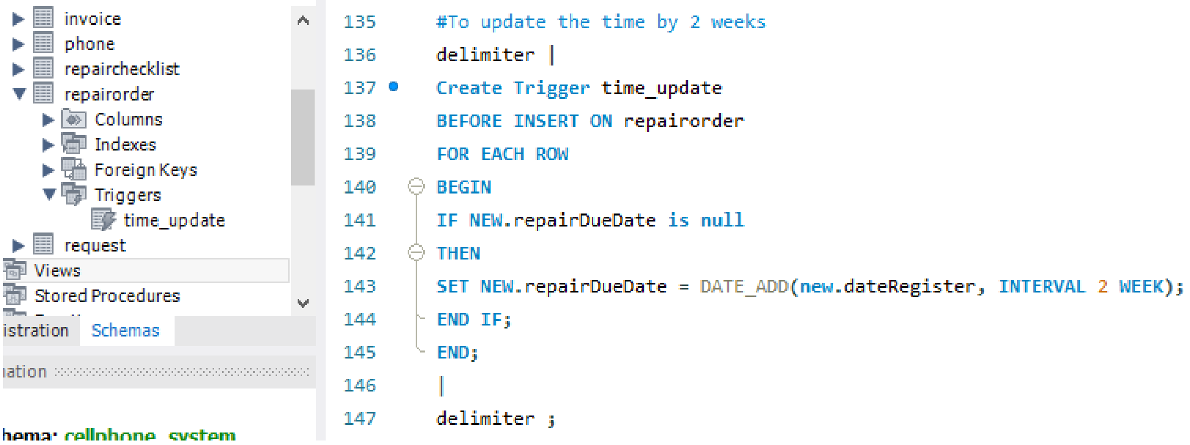
Schema: lab4_erm

```
#To ensure the manager has higher salary
delimiter |
CREATE TRIGGER salary_update
BEFORE INSERT ON EMPLOYEE
FOR EACH ROW
BEGIN
IF NEW.salary > (SELECT salary
FROM EMPLOYEE
WHERE eID = NEW.Mgr_id )
THEN SET NEW.salary = (SELECT salary
FROM EMPLOYEE
WHERE eID = NEW.Mgr_id )-1;
END IF;
END;
|
delimiter ;
SELECT * FROM employee;
```

eID	fname	lname	phonenumber	city	streetname	postcode	streetno	state	salary	Mgr_id	sex	dno
1	Chinedu	Mike	769452584	Hamburg	damtorstraße	24734	2	Saxony	45000	1	M	1
2	Chinedu	Amara	478452584	Alsdorf	Holenstraße	40334	2	Thuringia	44999	1	F	2
3	Michael	Eric	897694525	Arnis	Reepbahn	27134	2	Bravaria	30000	1	M	3
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fig m: salary update trigger

The trigger updates the due date for each repair order by 2weeks for those orders without due date.



hema: cellphone_system

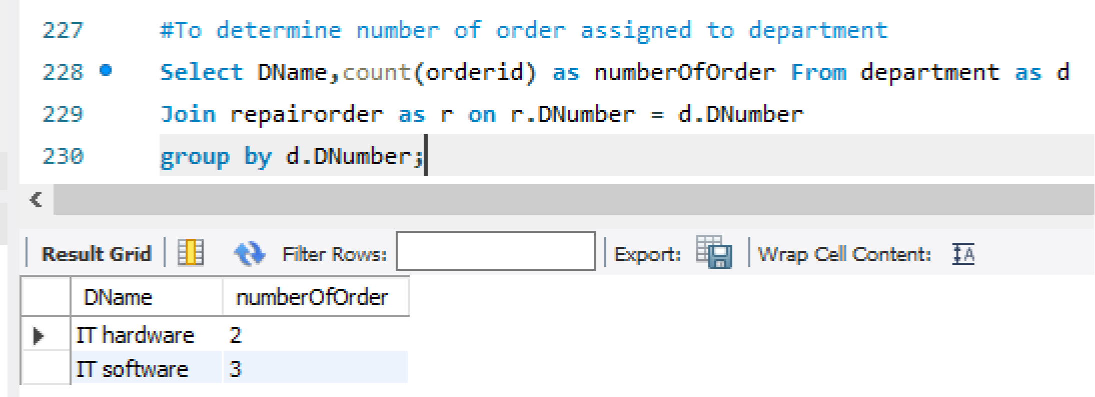
```
#To update the time by 2 weeks
delimiter |
Create Trigger time_update
BEFORE INSERT ON repairorder
FOR EACH ROW
BEGIN
IF NEW.repairDueDate is null
THEN
SET NEW.repairDueDate = DATE_ADD(new.dateRegister, INTERVAL 2 WEEK);
END IF;
END;
|
delimiter ;
```

Fig n: time update trigger

3.6 AGGREGATE FUNCTION

We have a lot of aggregate function but in this project, only count(), min(),max(),AVG() were used.

Count (): To view number of orders assigned to a department.



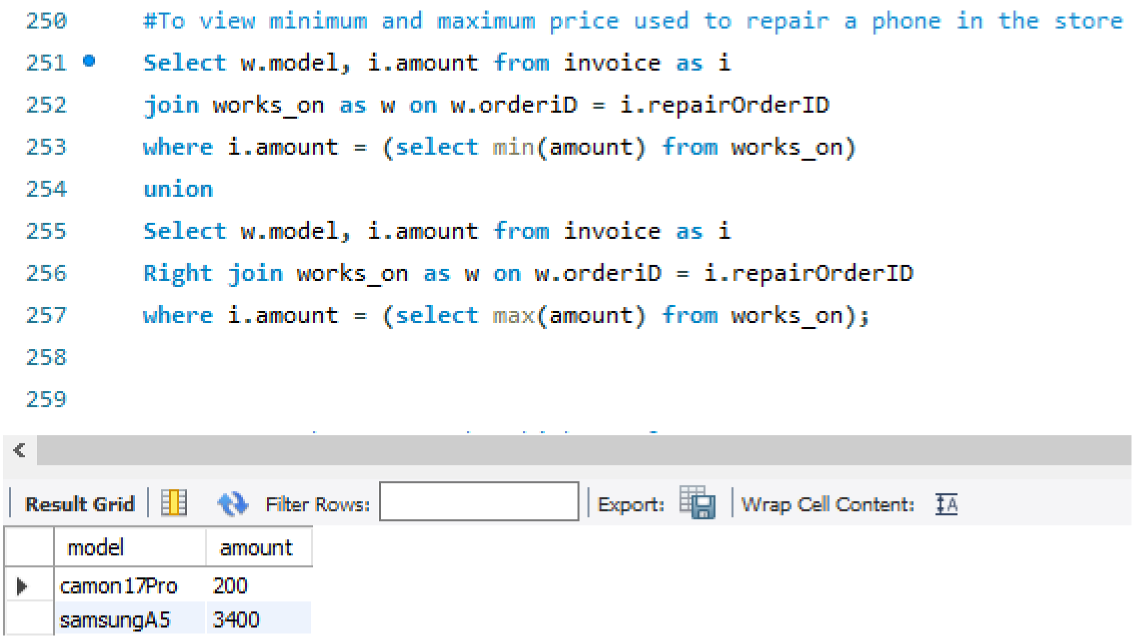
```
#To determine number of order assigned to department
Select DName,count(orderid) as numberOfOrder From department as d
Join repairorder as r on r.DNumber = d.DNumber
group by d.DNumber;
```

DName	numberOfOrder
IT hardware	2
IT software	3

Fig o: number of orders assigned to a department.

MIN (), MAX ()

The query displays the minimum and maximum amount paid to repair a phone.



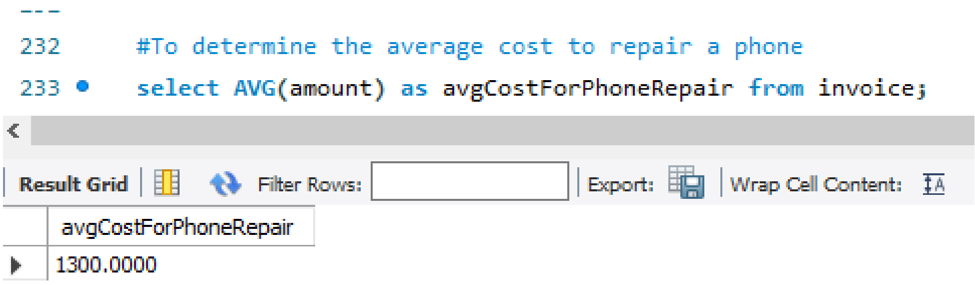
```
#To view minimum and maximum price used to repair a phone in the store
Select w.model, i.amount from invoice as i
join works_on as w on w.orderiD = i.repairOrderID
where i.amount = (select min(amount) from works_on)
union
Select w.model, i.amount from invoice as i
Right join works_on as w on w.orderiD = i.repairOrderID
where i.amount = (select max(amount) from works_on);
```

model	amount
camon17Pro	200
samsungA5	3400

Fig p: maximum and minimum amount to repair a phone.

AVG ()

This query returns the average cost to repair a phone in the store.



```
#To determine the average cost to repair a phone
select AVG(amount) as avgCostForPhoneRepair from invoice;
```

```
avgCostForPhoneRepair
1300.0000
```

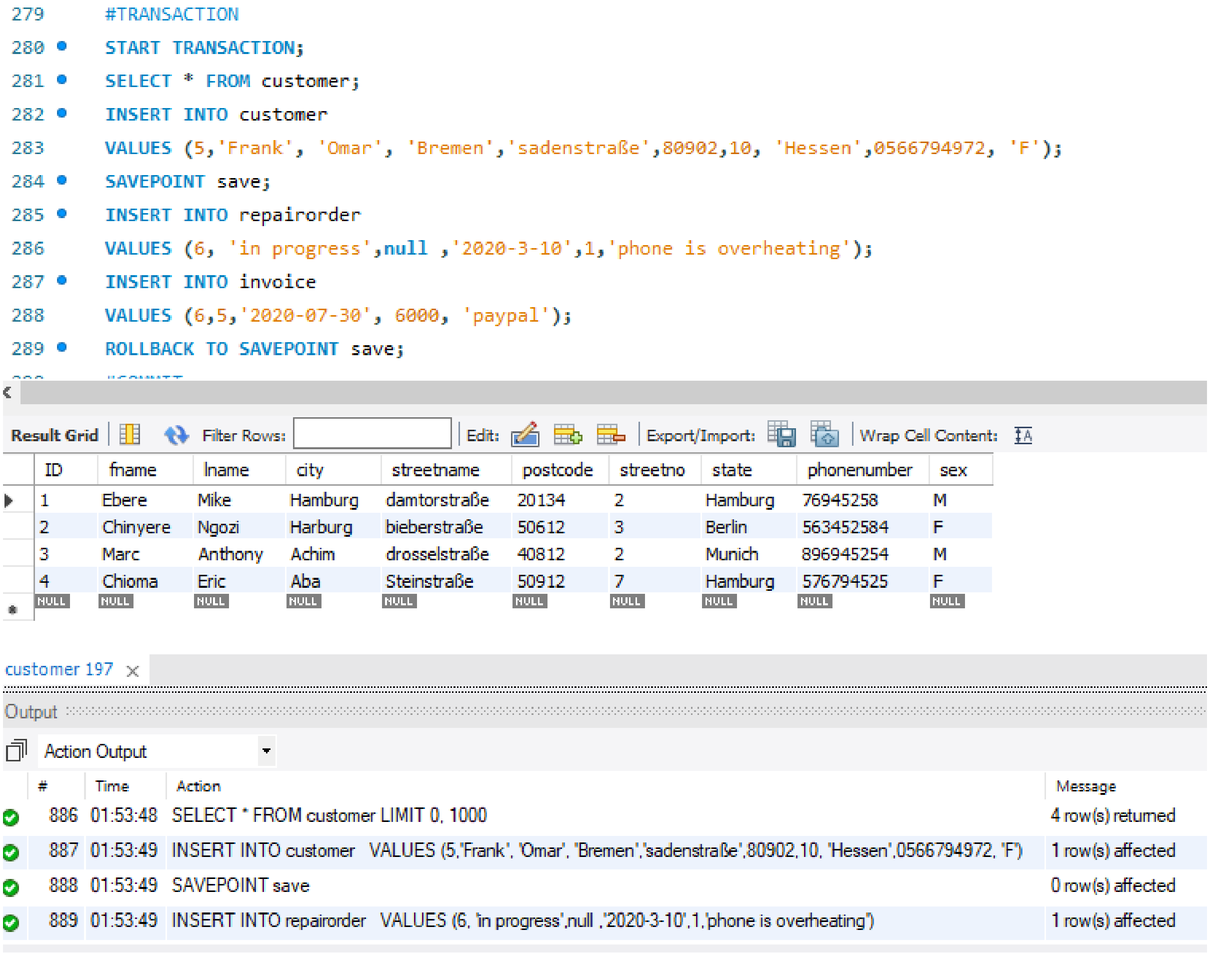
Fig q: average cost to repair a phone.

3.7 TRANSACTION

A transaction contains a group of statement such delete, update, insert as a unit which can be committed or rollback. It works on the principle of all or nothing. That is, a transaction cannot be successful without completing each operation available in the set. It means if any statement fails, the transaction operation cannot produce results.

Savepoint are "fall back" points. This makes it possible to rollback up to a save point and restart transaction execution from this point on.

This transaction consists of selects, insert and can commit or rollback.



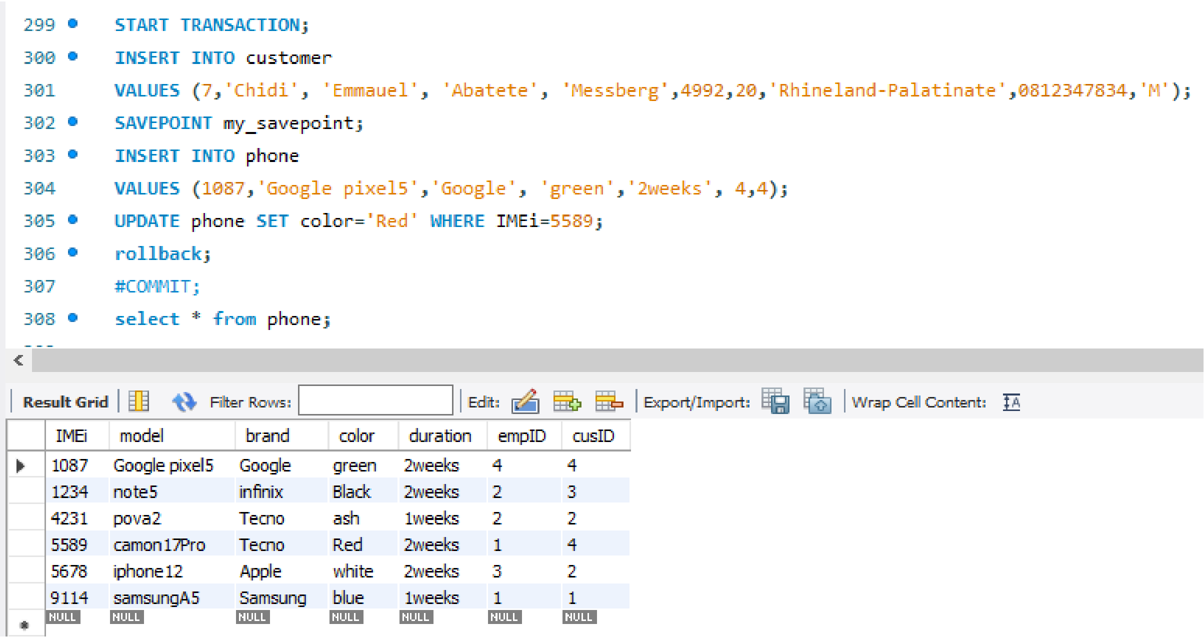
```
#TRANSACTION
START TRANSACTION;
SELECT * FROM customer;
INSERT INTO customer
VALUES (5,'Frank', 'Omar', 'Bremen','sadenstraße',80902,10, 'Hessen',0566794972, 'F');
SAVEPOINT save;
INSERT INTO repairorder
VALUES (6, 'in progress',null ,'2020-3-10',1,'phone is overheating');
INSERT INTO invoice
VALUES (6,5,'2020-07-30', 6000, 'paypal');
ROLLBACK TO SAVEPOINT save;
```

ID	fname	lname	city	streetname	postcode	streetno	state	phonenumber	sex
1	Ebere	Mike	Hamburg	damtorstraße	20134	2	Hamburg	76945258	M
2	Chinyere	Ngozi	Harburg	bieberstraße	50612	3	Berlin	563452584	F
3	Marc	Anthony	Achim	drosselstraße	40812	2	Munich	896945254	M
4	Chioma	Eric	Aba	Steinstraße	50912	7	Hamburg	576794525	F
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

#	Time	Action	Message
886	01:53:48	SELECT * FROM customer LIMIT 0, 1000	4 row(s) returned
887	01:53:49	INSERT INTO customer VALUES (5,'Frank', 'Omar', 'Bremen','sadenstraße',80902,10, 'Hessen',0566794972, 'F')	1 row(s) affected
888	01:53:49	SAVEPOINT save	0 row(s) affected
889	01:53:49	INSERT INTO repairorder VALUES (6, 'in progress',null ,'2020-3-10',1,'phone is overheating')	1 row(s) affected

Fig r: transaction1

This transaction consists of selects, insert, savepoint, update and can commit or rollback. This transaction insert into customer, phone and update the phone because there was an error in the colour of the phone.



```
START TRANSACTION;
INSERT INTO customer
VALUES (7,'Chidi', 'Emmauel', 'Abatete', 'Messberg',4992,20,'Rhineland-Palatinate',0812347834,'M');
SAVEPOINT my_savepoint;
INSERT INTO phone
VALUES (1087,'Google pixel5','Google', 'green','2weeks', 4,4);
UPDATE phone SET color='Red' WHERE IMEi=5589;
rollback;
#COMMIT;
select * from phone;
```

IMEi	model	brand	color	duration	empID	cusID
1087	Google pixel5	Google	green	2weeks	4	4
1234	note5	infinix	Black	2weeks	2	3
4231	pova2	Tecno	ash	1weeks	2	2
5589	camon17Pro	Tecno	Red	2weeks	1	4
5678	iphone12	Apple	white	2weeks	3	2
9114	samsungA5	Samsung	blue	1weeks	1	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fig s: transaction2

Conclusion

Database designing is critical to successfullly implement of a database management system that meets the data requirements of an enterprise system.

Normalization in DBMS is a process which helps produce database systems that have better security models. Functional dependencies are a very important component of the normalize data process.

A primary key uniquely identifies are record in a Table and cannot be null. A foreign key helps connect table and references a primary key.

Views helps to make work easier and faster to work it. Triggers helps to ensure consistency and security.

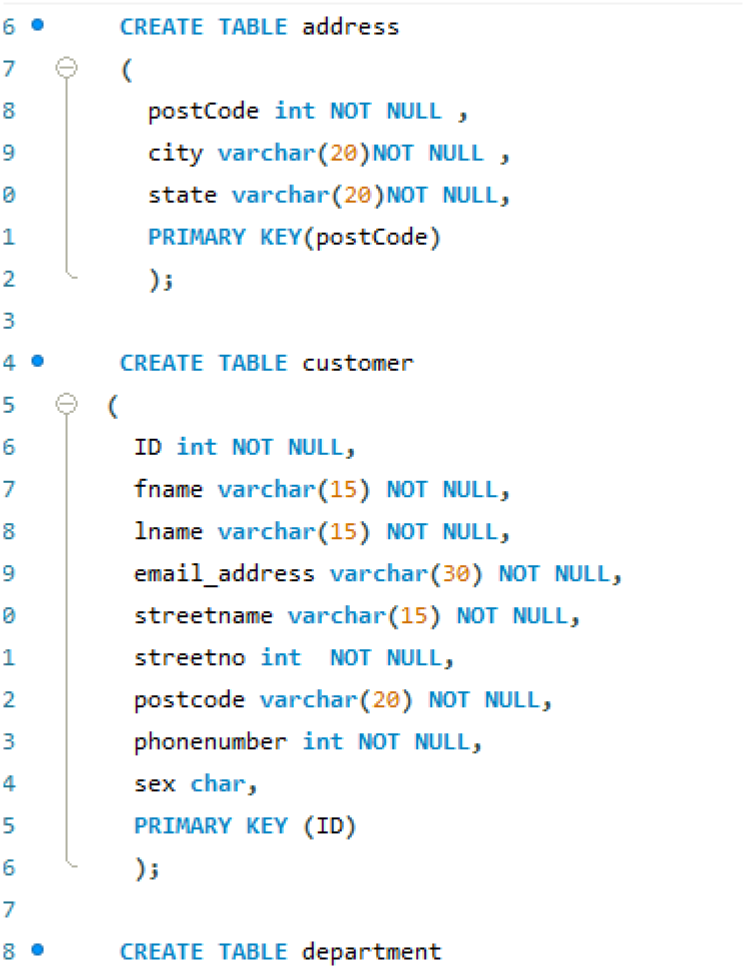
In generally, it was a wonderful and learning experience for me working on this project. As my skill was greatly improved.

References

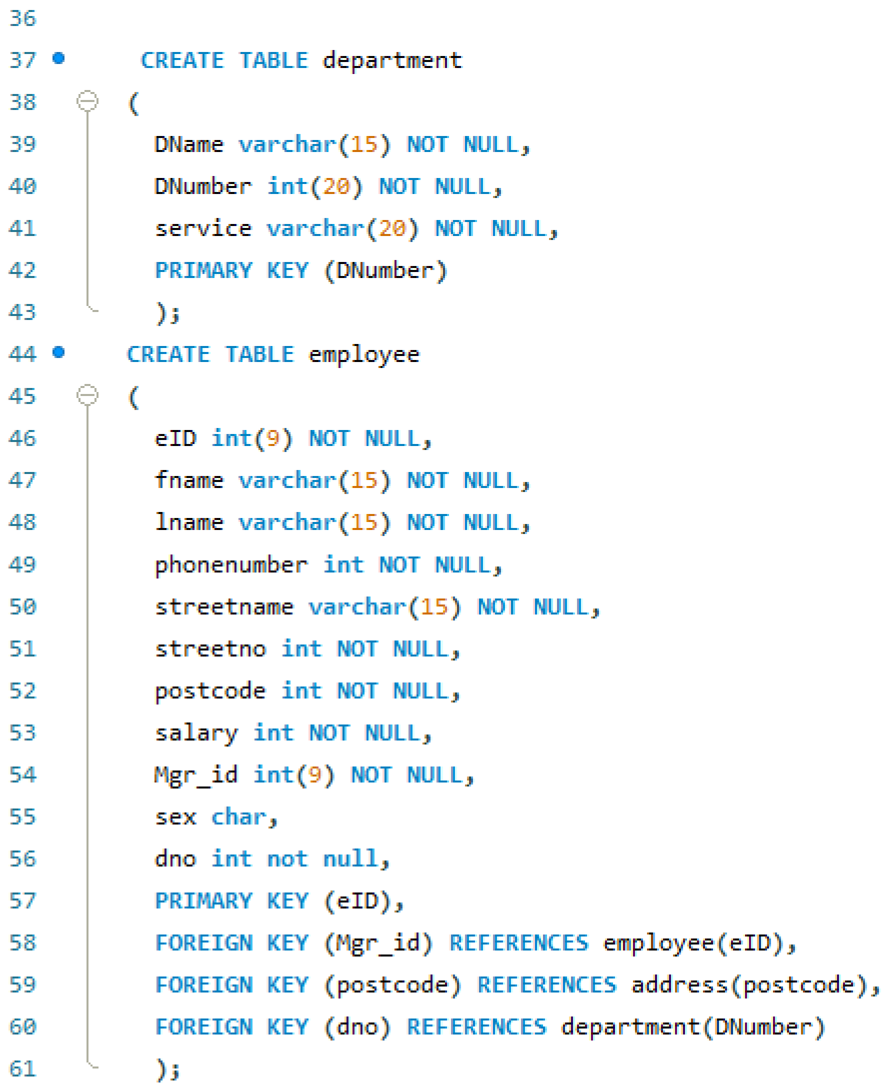
1. Prof. Dr. Ulrike Herster Hamburg University of Applied Sciences Summer Semester 2021, Lecture slides.
2. MySQL 8.0 Reference Manual

Appendix

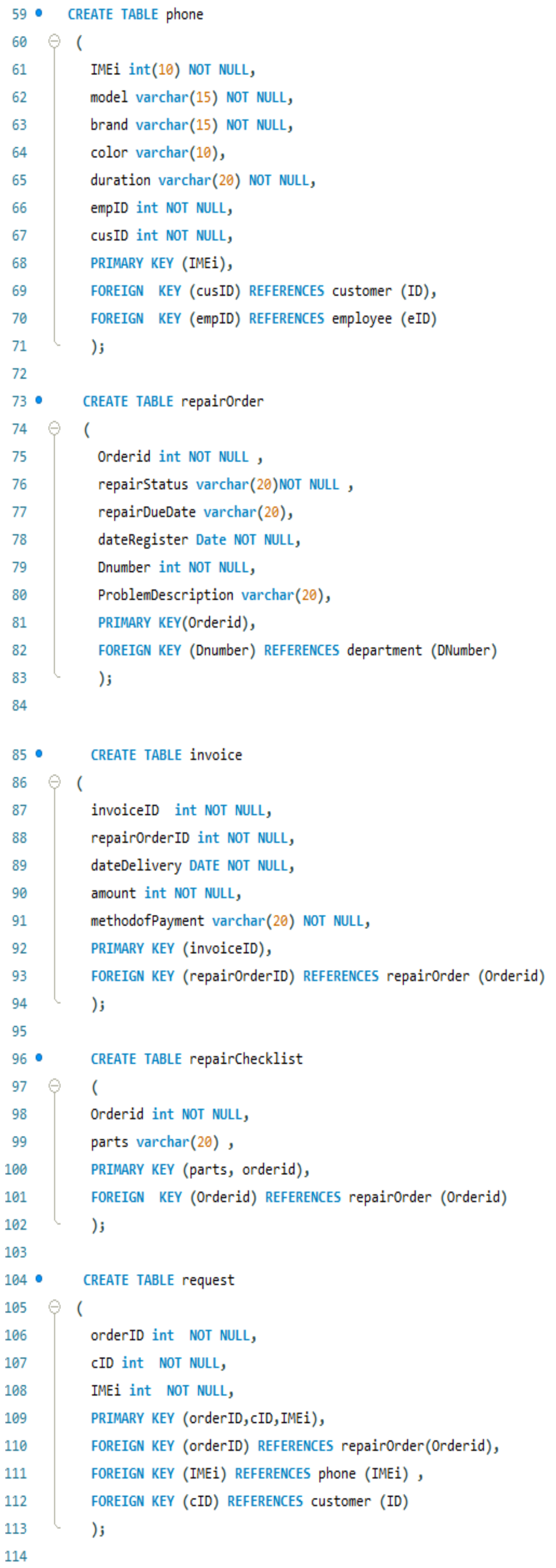
SQL queries for creating tables.



```
CREATE TABLE address
(
  postCode int NOT NULL ,
  city varchar(20)NOT NULL ,
  state varchar(20)NOT NULL,
  PRIMARY KEY(postCode)
  );

CREATE TABLE customer
(
  ID int NOT NULL,
  fname varchar(15) NOT NULL,
  lname varchar(15) NOT NULL,
  email_address varchar(30) NOT NULL,
  streetname varchar(15) NOT NULL,
  streetno int  NOT NULL,
  postcode varchar(20) NOT NULL,
  phonenumber int NOT NULL,
  sex char,
  PRIMARY KEY (ID)
  );

CREATE TABLE department
```



```sql

 CREATE TABLE department
(
  DName varchar(15) NOT NULL,
  DNumber int(20) NOT NULL,
  service varchar(20) NOT NULL,
  PRIMARY KEY (DNumber)
  );
CREATE TABLE employee
(
  eID int(9) NOT NULL,
  fname varchar(15) NOT NULL,
  lname varchar(15) NOT NULL,
  phonenumber int NOT NULL,
  streetname varchar(15) NOT NULL,
  streetno int NOT NULL,
  postcode int NOT NULL,
  salary int NOT NULL,
  Mgr_id int(9) NOT NULL,
  sex char,
  dno int not null,
  PRIMARY KEY (eID),
  FOREIGN KEY (Mgr_id) REFERENCES employee(eID),
  FOREIGN KEY (postcode) REFERENCES address(postcode),
  FOREIGN KEY (dno) REFERENCES department(DNumber)
  );
```



```
CREATE TABLE phone
(
  IMEi int(10) NOT NULL,
  model varchar(15) NOT NULL,
  brand varchar(15) NOT NULL,
  color varchar(10),
  duration varchar(20) NOT NULL,
  empID int NOT NULL,
  cusID int NOT NULL,
  PRIMARY KEY (IMEi),
  FOREIGN  KEY (cusID) REFERENCES customer (ID),
  FOREIGN  KEY (empID) REFERENCES employee (eID)
  );

  CREATE TABLE repairOrder
  (
   Orderid int NOT NULL ,
   repairStatus varchar(20)NOT NULL ,
   repairDueDate varchar(20),
   dateRegister Date NOT NULL,
   Dnumber int NOT NULL,
   ProblemDescription varchar(20),
   PRIMARY KEY(Orderid),
   FOREIGN KEY (Dnumber) REFERENCES department (DNumber)
   );

   CREATE TABLE invoice
(
  invoiceID  int NOT NULL,
  repairOrderID int NOT NULL,
  dateDelivery DATE NOT NULL,
  amount int NOT NULL,
  methodofPayment varchar(20) NOT NULL,
  PRIMARY KEY (invoiceID),
  FOREIGN KEY (repairOrderID) REFERENCES repairOrder (Orderid)
  );

   CREATE TABLE repairChecklist
   (
  Orderid int NOT NULL,
  parts varchar(20) ,
  PRIMARY KEY (parts, orderid),
  FOREIGN  KEY (Orderid) REFERENCES repairOrder (Orderid)
  );

  CREATE TABLE request
(
  orderID int  NOT NULL,
  cID int  NOT NULL,
  IMEi int  NOT NULL,
  PRIMARY KEY (orderID,cID,IMEi),
  FOREIGN KEY (orderID) REFERENCES repairOrder(Orderid),
  FOREIGN KEY (IMEi) REFERENCES phone (IMEi) ,
  FOREIGN KEY (cID) REFERENCES customer (ID)
  );
```