PES UNIVERSITY

100 feet Ring Road, BSK 3rd Stage, Bengaluru 560085

Department of Computer Science and Engineering

Jan – May 2020

UE18CS252
Database Management Systems

Project Report

Mobile-Retail DBMS

PES1201801481 Kushal K
4th Sem Sec. [I] Roll No. 36

PROJECT SUMMARY

A mobile_retail database can store information about its workman, its customers, depot, different departments, many categories in a department, items and payments. An ER-Model is developed which consists of 7 entities, 7 relations and 30 attributes, The following ER-Model is then mapped to Relational schema using mapping techniques in DBMS. An extra table orders is generated as a result m:n relationship between payment and items. Hence, the relational schema consists of 8 relations. A trigger is created in order to update the department stock count of an item after a successful purchase by a customer. 5 Queries has been written which generates the output based on the current state of the database. For the effective maintenance of the department system provides a way to assign a best workman. A customer's details can be saved without a mandatory address requirement which helps in easy checkout of the 1-time customers.

Introduction

A mobile retail mini-world stores the following information,

- Workman (identified by ID, with name, phone number, email, gender, address, salary as attributes), No designation to the workman's as they can turn their jobs as assistors or cashiers.

- Customers (identified by an ID, with name, email, address, phone number as attributes),
  A customer need not register his email/address mandatorily.

- Depot (with name as its attribute), Items are stored in depot and depot names can be same.

- Department (with name as its attribute), A department is managed by an employee.

- Categories in a department (identified by an ID, with name as its attribute), A category belongs to a single department, and a given category of items are stored in the same depot

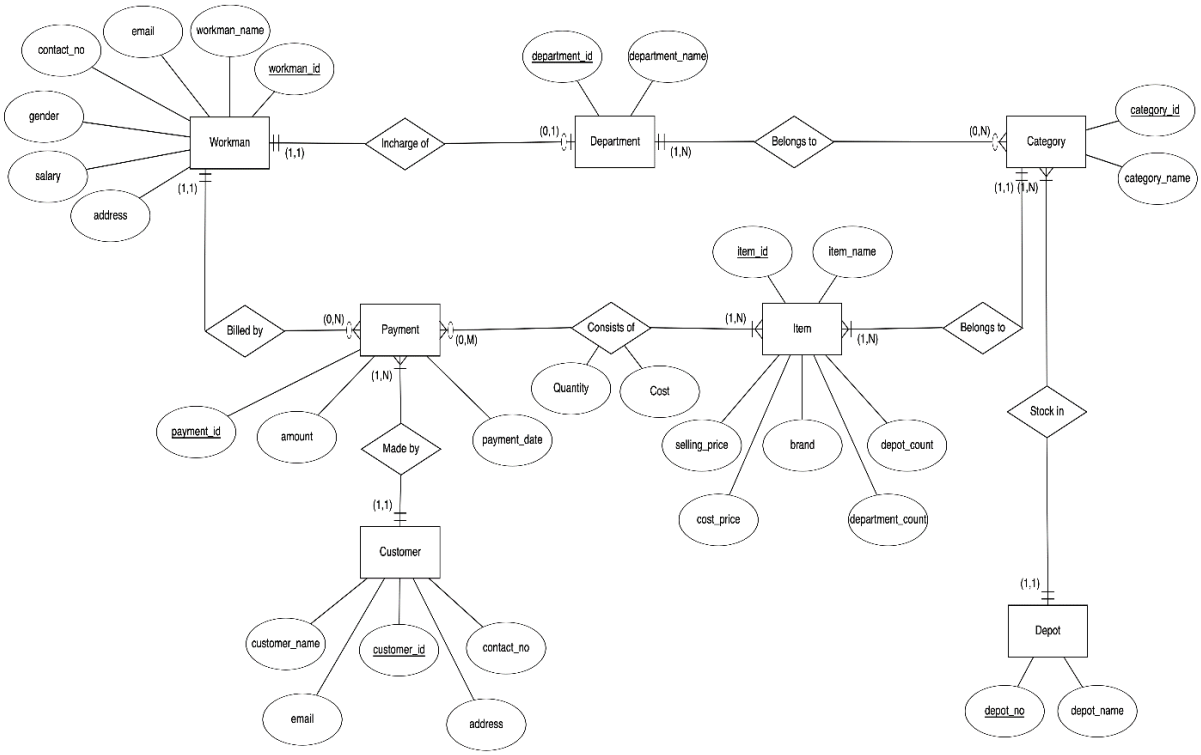
- Items (identified by an ID, with name, brand, selling price, cost price, department count and depot count as its attributes) , An item belongs to a single category.

- Payment (identified by an ID, with amount and date as its attributes), For each purchase by a customer, A payment is generated by the cashier. All the items in the purchase, their respective quantity and cost should be recorded as well.

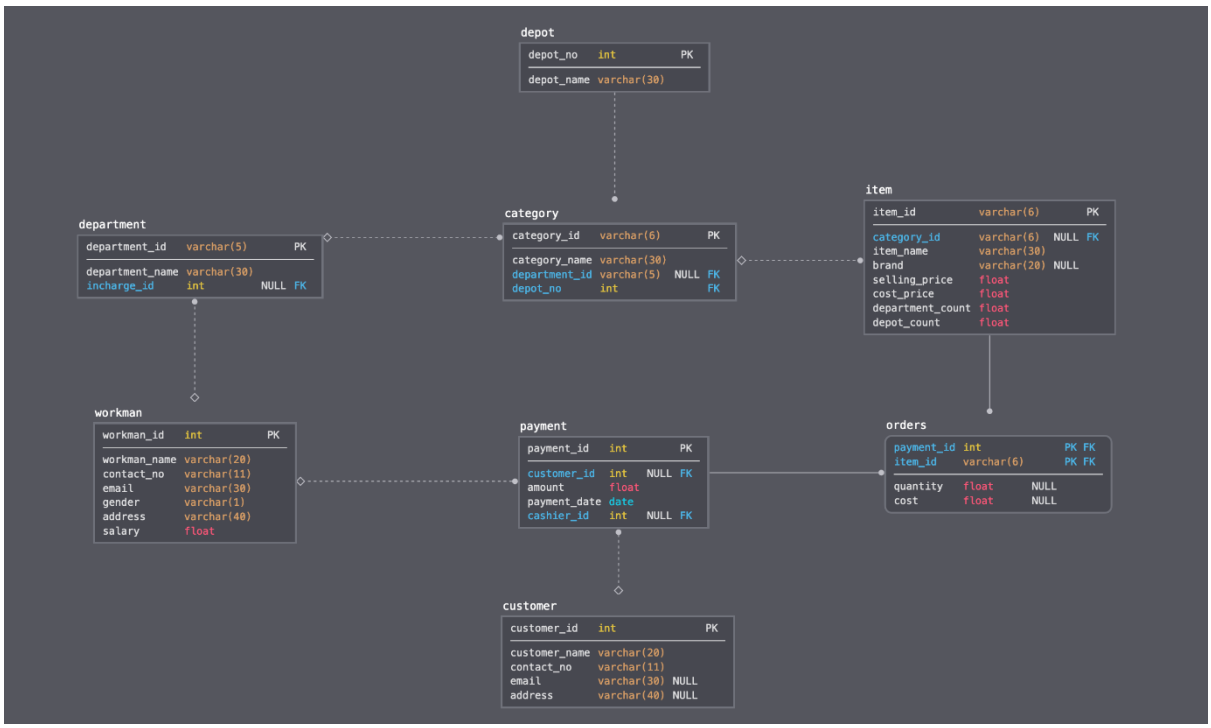
- Transactions of the system include updating the department stock count of an item after a successful purchase by a customer.

Data Model

ER Diagram



DATABASE SCHEMA



FD and Normalization

Functional Dependencies:

- depot_no → depot_name

- workman_id → workman_name,
  workman_id → contact_no,
  workman_id → email,
  workman_id → gender,
  workman_id → address,
  workman_id → salary

- customer_id → customer_name,
  customer_id → contact_no,
  customer_id → email,
  customer_id → address

- department_id → department_name,
  department_id → incharge_id

- category_id → category_name,
  category_id → department_id,
  category_id → depot_no

- item_id → item_name,
  item_id → brand,
  item_id → category_id,
  item_id → selling_price,
  item_id → cost_price,
  item_id → department_count,
  item_id → depot_count

- {brand, item_name} → selling_price, cost_price, department_count, depot_count, product_id

- payment_id → customer_id,
  payment_id → amount,
  payment_id → payment_date,
  payment_id → cashier_id

- payment_id, item_id → quantity, payment_id, item_id → cost

Candidate keys:

1. Depot
   On applying attribute closure,
   { depot_no }+ = { depot_no, depot_name }
   It can be seen that the closure for the attribute depot_no covers all the other attributes.
   Hence, candidate key = { depot_no } .

2. Workman
   On applying attribute closure,
   { workman_id }+ = { workman_id, workman_name, contact_no, email, address, gender, salary } It can be seen that the closure for the attribute workman_id covers all the other attributes. Hence, candidate key = { workman_id } .

3. Customer
   On applying attribute closure,
   { customer_id }+ = { customer_id, customer_name, contact_no, email, address }
   It can be seen that the closure for the attribute customer_id covers all the other attributes.
   Hence, candidate key = { customer_id } .

4. Department
   On applying attribute closure,
   { department_id }+ = { department_id, department_name, incharge_id }
   It can be seen that the closure for the attribute department_id covers all the other attributes. Hence, candidate key = { department_id } .

5. Category
   On applying attribute closure,
   { category_id }+ = { category_id, category_name, department_id, depot_no }
   It can be seen that the closure for the attribute category_id covers all the other attributes.
   Hence, candidate key = { category_id } .

6. Item
   On applying attribute closure,
   { item_id }+ = { item_id, item_name, category_id, brand, selling_price, cost_price, department_count, depot_count }
   { brand, item_name }+ = { item_id, item_name, category_id, brand, selling_price, cost_price, department_count, depot_count }

It can be seen that the closure for the attributes item_id and { brand, item_name} cover all the other attributes. Hence, both of them form the candidate keys. Hence, candidate keys { item_id , (brand, item_name) } The primary key is chosen as { item_id } .

7. Payment
   On applying attribute closure,
   { payment_id }+ = { payment_id, customer_id, amount, date, cashier_id }
   It can be seen that the closure for the attribute payment_id covers all the other attributes. Hence, candidate key = { payment_id } .

8. Orders
   On applying attribute closure,
   { payment_id, item_id }+ = { payment_id, item_id, quantity, cost }
   It can be seen that the closure for the attributes { payment_id, item_id } covers all the other attributes. Hence, candidate key = { (payment_id, item_id) }

Normalization and testing for lossless join property:

- Since the relations have been attained by first creating an ER model and then converting it to a relational schema, the relations are in normalized form of 3NF and BCNF.

- As the relations are already in their normal forms, there are no further decompositions involved. Hence, there is no need to test for the lossless join property.

- One case where it would violate 2NF would be when item_name is added to the orders relation. The primary key is { payment_id, item_id } and the FD being considered is item_id → item_name This would create a partial dependency on the primary key, since item_name can be functionally determined only by item_id. Hence, it would not result in a 2NF.

DDL

```
CREATE DATABASE mobile_retail;

CREATE TABLE depot
(
    depot_no INT,
    depot_name VARCHAR(30) NOT NULL,
    PRIMARY KEY(depot_no));
```

```
INSERT INTO depot VALUES(101,'RR NAGAR-1');
INSERT INTO depot VALUES(102,'COSMO RETAILS');
INSERT INTO depot VALUES(103,'IMPERIAL STOCKS');
INSERT INTO depot VALUES(104,'SHANTHI TRADERS');



CREATE TABLE workman
(
    workman_id INT,
    workman_name VARCHAR(20) NOT NULL,
    contact_no VARCHAR(11) NOT NULL,
    email VARCHAR(30) NOT NULL,
    gender VARCHAR(1) NOT NULL CHECK(gender='M' OR gender='F' OR gender='O'),
    address VARCHAR(40) NOT NULL,
    salary FLOAT NOT NULL,
    PRIMARY KEY(workman_id));

INSERT INTO workman VALUES(1011,'Ramesh','9545341211','ramesh121@gmail.com','M','#201,Rajajinagar, Ban
galore-87',12500);
INSERT INTO workman VALUES(1012,'Shruti','9545341211','shruti101@gmail.com','F','#234,Vijayanagr, Bang
alore- 78',22500);
INSERT INTO workman VALUES(1013,'Ramya','9545341211','ramya_321@gmail.com','F','#45, RR Nagar, Bangalo
re-98',13500);
INSERT INTO workman VALUES(1014,'Harish','9545341211','harish.ak@gmail.com','M','#32,Deepanjali Nagar,
Bangalore-84',8500);
INSERT INTO workman VALUES(1015,'Mary','9545341211','maryjoseph@gmail.com','F','#49, Rajajinagar,Banga
lore-87',11500);
INSERT INTO workman VALUES(1016,'Sumanth','9545341211','sumath_ar@gmail.com','M','#21,Rajajinagar,Bang
al ore-87',12500);
INSERT INTO workman VALUES(1017,'Amar','9545341211','amarrnatheshwar@gmail.com','M','#76,Chandra Layou
t,Bangalore-47',21500);
INSERT INTO workman VALUES(1018,'Dhanush','9545341211','dhanush_k@gmail.com','M','#28, KR Palya,Bangal
ore-81',23500);
INSERT INTO workman VALUES(1019,'Aishwarya','9545341211','aishu121@gmail.com','F','#286, Rajajinagar,B
angalore-87',12500);
INSERT INTO workman VALUES(1020,'Deepak','9545341211','deepakrao@gmail.com','M','#34,RR Nagar,Bangalor
e-98',20500);

CREATE TABLE customer
(
    customer_id INT,
    customer_name VARCHAR(20) NOT NULL,
    contact_no VARCHAR(11) NOT NULL,
    email VARCHAR(30) ,
    address VARCHAR(40),
    PRIMARY KEY(customer_id));
```

```
INSERT INTO customer VALUES(2003,'Chethan R','9876543210','cheth@gmx.com','Aarohi, 6th main, Vijaynaga
r');
INSERT INTO customer VALUES(2004,'Ragahvendra M','9686263452','raghs34@gmail.com','Avani, 7th block, R
ajajinagar');
INSERT INTO customer VALUES(2007,'Yash Purohit','7656453478',NULL);


CREATE TABLE department
(
    department_id VARCHAR(5),
    department_name VARCHAR(30) NOT NULL,
    incharge_id INT,
    PRIMARY KEY(department_id),
    FOREIGN KEY(incharge_id) REFERENCES workman(workman_id) ON DELETE SET NULL ON UPDATE CASCADE);

INSERT INTO department VALUES('PH000','Mobile Phones',1012);
INSERT INTO department VALUES('AC000','Mobile Accessories',1018);
INSERT INTO department VALUES('PR000','Mobile Protection',1020);

CREATE TABLE category
(
    category_id VARCHAR(6),
    category_name VARCHAR(30) NOT NULL,
    department_id VARCHAR(5),
    depot_no INT NOT NULL,
    PRIMARY KEY(category_id),
    FOREIGN KEY(department_id) REFERENCES department(department_id) ON DELETE SET NULL ON UPDATE CASCA
DE,
    FOREIGN KEY(depot_no) REFERENCES depot(depot_no) ON DELETE SET NULL ON UPDATE CASCADE);

INSERT INTO category VALUES('PH101','Flip Phones','PH000',101);
INSERT INTO category VALUES('PH102','Full Screen Phones','PH000',103);
INSERT INTO category VALUES('PH103','Keypad Phones','PH000',103);

INSERT INTO category VALUES('AC101','Earphones','AC000',102);
INSERT INTO category VALUES('AC102','Headphones','AC000',102);
INSERT INTO category VALUES('AC103','Chargers','AC000',102);

INSERT INTO category VALUES('PR101','tempered glass','PR000',103);
INSERT INTO category VALUES('PR102','protection covers','PR000',103);
INSERT INTO category VALUES('PR103','warranty extension','PR000',103);
CREATE TABLE item
(
    item_id VARCHAR(6),
    category_id VARCHAR(6),
    item_name VARCHAR(30) NOT NULL,
    brand VARCHAR(20),
```

```
    --weight VARCHAR(10)  NOT NULL,
    selling_price FLOAT NOT NULL,
    cost_price FLOAT NOT NULL,
    department_count FLOAT NOT NULL,
    depot_count FLOAT NOT NULL,
    PRIMARY KEY(item_id),
    FOREIGN KEY(category_id) REFERENCES category(category_id) ON DELETE SET NULL ON UPDATE CASCADE);

INSERT INTO item VALUES('PT101','PH101','blackberry flip','Blackberry',50000,43000,100,250);
INSERT INTO item VALUES('PT102','PH101','samsung fold','Samsung',140000,110000,100,250);

INSERT INTO item VALUES('PT105','PH102','iphone 11','Apple',70000,64000,40,100);
INSERT INTO item VALUES('PT106','PH102','iphone 11 pro','Apple',118000,106000,40,100);
INSERT INTO item VALUES('PT107','PH102','one plus 8','Oneplus',50000,44000,50,150);
INSERT INTO item VALUES('PT108','PH102','samsung s20','Samsung',90000,81000,50,150);
INSERT INTO item VALUES('PT109','PH102','huawei p30','Huawei',45000,30000,60,100);

INSERT INTO item VALUES('PT110','PH103','nokia 7630','Nokia',2000,1800,45,100);
INSERT INTO item VALUES('PT111','PH103','nokia keypad pro','Nokia',4000,3600,45,100);
INSERT INTO item VALUES('PT112','PH103','samsung s2101 keypad','Samsung',2500,2400,50,100);
INSERT INTO item VALUES('PT113','PH103','blackberry ultra keypad','Blackberry',10000,8500,50,150);


INSERT INTO item VALUES('PT138','AC101','philips lite bass','Philips',1200,1100,30,400);
INSERT INTO item VALUES('PT139','AC101','Airpods 2','Apple',14500,13000,5,20);
INSERT INTO item VALUES('PT140','AC101','sony mdr extra bass','Sony',2200,1950,3,3);
INSERT INTO item VALUES('PT141','AC101','oneplus shots','Oneplus',6000,5500,5,10);

INSERT INTO item VALUES('PT142','AC102','Bose soundtrue','Bose',20000,18000,10,120);
INSERT INTO item VALUES('PT143','AC102','marshaal wireless3','Marshaal',7500,5400,10,100);

INSERT INTO item VALUES('PT144','AC103','Apple fast C-type','Apple',3500,3200,3,70);
INSERT INTO item VALUES('PT145','AC103','samsung d-type','Samsung',1450,1100,10,100);
INSERT INTO item VALUES('PT146','AC103','14w travel charger','Onplus',3200,3000,5,90);
INSERT INTO item VALUES('PT147','AC103','multiple ex charger','Blackberry',900,850,10,80);


INSERT INTO item VALUES('PT152','PR101','A1 quality t-glass','Spigen',1000,800,20,100);
INSERT INTO item VALUES('PT153','PR101','A2 quality t-glass','T-glass company',800,600,20,200);
INSERT INTO item VALUES('PT154','PR101','A3 quality t-glass','Phone Warrior',500,400,50,150);
INSERT INTO item VALUES('PT155','PR101','A4 quality t-glass','Sheild',200,150,15,100);

INSERT INTO item VALUES('PT156','PR102','apple protection cover','Apple',3500,3000,30,400);
INSERT INTO item VALUES('PT157','PR102','samsung protection cover','Samsung',2500,2200,40,350);
INSERT INTO item VALUES('PT158','PR102','oneplus protection cover','Oneplus',1500,1000,15,300);
INSERT INTO item VALUES('PT159','PR102','huawei protection cover','Huawei',500,460,10,80);

INSERT INTO item VALUES('PT164','PR103','Apple warranty extension plan','Apple',8000,7900,50,500);
```

```
INSERT INTO item VALUES('PT165','PR103','samsung warranty ext plan','Samsung',6000,5800,80,600);
INSERT INTO item VALUES('PT166','PR103','oneplus warranty ext plan','Oneplus',4000,3900,50,400);
INSERT INTO item VALUES('PT167','PR103','blackberry warranty ext plan','Blackberry',3500,3400,30,550);
INSERT INTO item VALUES('PT168','PR103','huawei warranty ext plan','Huawei',3300,3200,10,50);




CREATE TABLE payment
(
    payment_id INT,
    customer_id INT,
    amount FLOAT NOT NULL,
    payment_date DATE NOT NULL,
    cashier_id INT,
    PRIMARY KEY(payment_id),
    FOREIGN KEY(customer_id) REFERENCES customer(customer_id) ON DELETE SET NULL ON UPDATE CASCADE,
    FOREIGN KEY(cashier_id) REFERENCES workman(workman_id) ON DELETE SET NULL ON UPDATE CASCADE);

INSERT INTO payment VALUES(1011,2003,93650,'2020-05-16',1013);
INSERT INTO payment VALUES(1012,2004,21000,'2020-05-18',1019);
INSERT INTO payment VALUES(1013,2007,53000,'2020-05-19',1019);

CREATE TABLE orders
(
    payment_id INT,
    item_id VARCHAR(6),
    quantity FLOAT,
    cost FLOAT,
    PRIMARY KEY(payment_id,item_id),
    FOREIGN KEY(payment_id) REFERENCES payment(payment_id) ON DELETE SET NULL ON UPDATE CASCADE,
    FOREIGN KEY(item_id) REFERENCES item(item_id) ON DELETE SET NULL ON UPDATE CASCADE);




INSERT INTO orders VALUES(1011,'PT108',1,90000);
INSERT INTO orders VALUES(1011,'PT140',1,2200);
INSERT INTO orders VALUES(1011,'PT145',1,1450);

 INSERT INTO orders VALUES(1012,'PT164',1,8000);
INSERT INTO orders VALUES(1012,'PT156',4,14000);

INSERT INTO orders VALUES(1013,'PT167',1,3500);
INSERT INTO orders VALUES(1013,'PT101',1,50000);
```

Triggers

A trigger to update the department stock count of a product after a purchase by a customer.

```
CREATE FUNCTION manage_count()
RETURNS TRIGGER AS $count$
BEGIN

UPDATE item SET department_count = department_count - new.quantity
WHERE item_id = new.item_id AND department_count - new.quantity >= 0;
RETURN NEW;

END;
$count$ LANGUAGE plpgsql;

CREATE TRIGGER deprt_count
AFTER INSERT ON orders
FOR EACH ROW EXECUTE PROCEDURE manage_count();
```

```
→  ~ psql
psql (12.3)
Type "help" for help.

kushalkrishnappa333=# \c mobile_retail
You are now connected to database "mobile_retail" as user "kushalkrishnappa333".
mobile_retail=# CREATE FUNCTION manage_count()
mobile_retail-# RETURNS TRIGGER AS $count$
mobile_retail$# BEGIN
mobile_retail$#
mobile_retail$# UPDATE item SET department_count = department_count - new.quantity
mobile_retail$# WHERE item_id = new.item_id AND department_count - new.quantity >= 0;
mobile_retail$# RETURN NEW;
mobile_retail$#
mobile_retail$# END;
mobile_retail$# $count$ LANGUAGE plpgsql;
CREATE FUNCTION
mobile_retail=# CREATE TRIGGER deprt_count
mobile_retail-# AFTER INSERT ON orders
mobile_retail-# FOR EACH ROW EXECUTE PROCEDURE manage_count();
CREATE TRIGGER
mobile_retail=#
```

Figure 1: Trigger executed in mobile_retail db

SQL Queries

1) Query the list of customers who have not registered their address.

```
SELECT customer_name, contact_no
FROM customer
where ADRESS is NULL;
```

```
kushalkrishnappa333=# \c mobile_retail
You are now connected to database "mobile_retail" as user "kushalkrishnappa333".
mobile_retail=# SELECT customer_name, contact_no
mobile_retail-# FROM customer
mobile_retail-# WHERE address IS NULL;
 customer_name | contact_no
---------------+------------
 Yash Purohit  | 7656453478
(1 row)
```

Figure 2: Question 1

2) List the item id, total number of units sold and profit made for all the items that are sold.

```
SELECT item_id, total_units, (total - total_units*cost_price)
AS profit
FROM  (SELECT SUM(quantity) AS total_units,
              SUM(cost) as total,
              item_id
       FROM orders
       GROUP BY item_id )
as rel1 NATURAL JOIN item;
```

```
mobile_retail=# SELECT item_id, total_units, (total - total_units * cost_price)
mobile_retail-# AS profit
mobile_retail-# FROM (SELECT SUM(quantity) AS total_units,
mobile_retail(#              SUM(cost) AS total,
mobile_retail(#              item_id
mobile_retail(#       FROM orders
mobile_retail(#       GROUP BY item_id)
mobile_retail-# AS rel1 NATURAL JOIN item;
 item_id | total_units | profit
---------+-------------+--------
 PT101   |           1 |   7000
 PT108   |           1 |   9000
 PT140   |           1 |    250
 PT145   |           1 |    350
 PT156   |           4 |   2000
 PT164   |           1 |    100
 PT167   |           1 |    100
(7 rows)
```

Figure 3: Question 2

3) List item id, item name, brand and category name of item which are sold whose category id begins with PH and stored in depot number 103.

```
SELECT item_id, item_name, brand, category_name
FROM item
NATURAL  JOIN (SELECT category_id,category_name
               FROM category
               WHERE category_id LIKE 'PH%' AND depot_no = 103)
AS rel1;
```

```
mobile_retail=# SELECT item_id, item_name, brand, category_name
mobile_retail-# FROM item
mobile_retail-# NATURAL JOIN (SELECT category_id,category_name
mobile_retail(#               FROM category
mobile_retail(#               WHERE category_id LIKE 'PH%' AND depot_no = 103)
mobile_retail-# AS rel1;
 item_id |        item_name         |   brand    |   category_name
---------+--------------------------+------------+--------------------
 PT105   | iphone 11                | Apple      | Full Screen Phones
 PT106   | iphone 11 pro            | Apple      | Full Screen Phones
 PT107   | one plus 8               | Oneplus    | Full Screen Phones
 PT108   | samsung s20              | Samsung    | Full Screen Phones
 PT109   | huawei p30               | Huawei     | Full Screen Phones
 PT110   | nokia 7630               | Nokia      | Keypad Phones
 PT111   | nokia keypad pro         | Nokia      | Keypad Phones
 PT112   | samsung s2101 keypad     | Samsung    | Keypad Phones
 PT113   | blackberry ultra keypad | Blackberry | Keypad Phones
(9 rows)
```

Figure 4: Question 3

4) Items whose department stock count is less than 10 have to be informed to transfer stock from the depot. List the item id, its details and the respective department incharge id whose department stock count is less than 10.

```
SELECT incharge_id, item_id, item_name, brand
FROM department
NATURAL JOIN (SELECT department_id, item_id, item_name, brand
              FROM  (SELECT item_id,category_id, item_name, brand
                     FROM item
                     WHERE department_count < 10)
                     AS rel1
                     NATURAL JOIN category)
AS rel2;
```

```
mobile_retail=# SELECT incharge_id, item_id, item_name, brand
mobile_retail-# FROM department
mobile_retail-# NATURAL JOIN(SELECT department_id, item_id, item_name, brand
mobile_retail(#              FROM (SELECT item_id,category_id, item_name, brand
mobile_retail(#                    FROM item
mobile_retail(#                    WHERE department_count < 10)
mobile_retail(#              AS rel1
mobile_retail(#              NATURAL JOIN category)
mobile_retail-# AS rel2;
 incharge_id | item_id |      item_name      |  brand
-------------+---------+---------------------+---------
        1018 | PT139   | Airpods 2           | Apple
        1018 | PT140   | sony mdr extra bass | Sony
        1018 | PT141   | oneplus shots       | Oneplus
        1018 | PT144   | Apple fast C-type   | Apple
        1018 | PT146   | 14w travel charger  | Onplus
(5 rows)
```

Figure 5: Question 4

5) List the customers who shopped for at least a total of 3 quantities of any item (maybe same item too) and paid an amount greater than ₹ 80000.

```
SELECT customer_id, customer_name
FROM customer
NATURAL JOIN (SELECT customer_id
              FROM (SELECT payment_id
                    FROM orders
                    GROUP BY payment_id
                    HAVING SUM(quantity) > 3 OR SUM(quantity) = 3)
              AS rel1
              NATURAL JOIN payment
              WHERE amount > 80000)
AS rel2;
```

```
mobile_retail=# SELECT customer_id, customer_name
mobile_retail-# FROM customer
mobile_retail-# NATURAL JOIN (SELECT customer_id
mobile_retail(#               FROM (SELECT payment_id
mobile_retail(#                     FROM orders
mobile_retail(#                     GROUP BY payment_id
mobile_retail(#                     HAVING SUM(quantity) > 3 OR SUM(quantity) = 3)
mobile_retail(#               AS rel1
mobile_retail(#               NATURAL JOIN payment
mobile_retail(#               WHERE amount > 80000)
mobile_retail-# AS rel2;
 customer_id | customer_name
-------------+---------------
        2003 | Chethan R
(1 row)
```

Figure 6: Question 5

Conclusion

- A mobile_retail database can store information about its workman, its customers, depot, different departments, many categories in a department, items and payments. For the effective maintenance of the department system provides a way to assign a best workman. A customer's details can be saved without a mandatory address requirement which helps in easy checkout of the 1-time customers. This system provides a functionality which updates a department stock count when a customer buys an item.

- Limitations of the system includes, there is no provision to assign a workman to maintain the specific category of the department. The system doesn't provide the facility for customers who wish to buy items online. Hence, the system for now supports only offline transactions.

- Future enhancement could be implemented to overcome the above-mentioned limitations, This includes a provision for customers to buy products online and check the status of the order. Only the customers order is placed online, the backend should work similar to the now presented system.