**UR HUYE CAMPUS**

**Module: SYSTEMS ENGINEERING**

**Faculty: BIT LEVEL 2**

**Academic year:2021\_2022**

**Date:5/June/2022**

**Reg number:220013685**

**Name: Umurerwa Mireille**

TOPIC: UZURI ONLINE SHOPPING SYSTEM

Part 1: **SYSTEMS ENGINEERING**

TABLE OF CONTENTS

1.0: Introduction

1.1: General objective

1.2: Specific objective

1.3: Problems with the existing system

1.4: Solution to the problems

1.5: Functional requirements

1.6: Nonfunctional requirements

1.7: Data Flow Diagram

1.7.1: DFD Level 0

1.7.2: DFD Level 1

1.7.3: DFD Level 2

1.8: Use case diagram

1.8: Class diagram

1.9: Sequence diagram

TOPIC: UZURI ONLINE SHOPPING SYSTEM

1: Introduction

Uzuri is an African shoe brand and manufacturer established in Rwanda since 2013. The company was founded by two entrepreneurs (Kevine&Ysolde) who met at university while studying Creative Design. UZURI is located in Kigali with blanches at Nyarugenge.KN 4 Ave, Kibagabaga.KG 19 Ave, Kigali international airport. Kanombe.

1.1: General objective

The general objectives of Uzuri online shopping system are the following:

* To pre-define on which the whole system works to achieve them by managing the details of all customers, payments, bills, products, shopping, and so on.
* This system helps in boosting efficient services.

1.2: Specific objective

Specific objective of uzuri online shopping system are the following:

* To shop while in the comfort of your own home, without having to step out of the door
* To be able to easily save money and compare prices from website to website
* Online resellers tent to sell at a lower price due to less overhead expenses
* To provide the community with functional and sophisticated products.
* To also share their skills with other local tailors
* To bring Rwanda fashion industry and make difference all over the Africa

1.3: Problems with the existing system

* Additional charges: it is when users at the end of a checkout process are faced with unexpected high charges.
* Lack of security and privacy leaks: there are few events that are more damaging to online retailers than data leaks. Every year there are thousands of these ranging from small ecommerce sites being hacked by backdoor exploiting bots to large scandals.
* Missing or fake product reviews: besides the fact that fake product reviews are lazy and outright lies, they also tend to be very obvious. The reason for this is that they are written by you or your team, and not the customers, and more often than not they won’t sound genuine.
* The lack of personalization: if you provide a personalized shopping experience, you have much better chance of converting your visitors to customers, simply because you can show them products, they are interested in instead of showcasing your entire product range.
* Bad user experience: the deterioration of user experience is usually related to elements that shouldn’t be noticed by the user in the first place.in design this could mean that your site is not mobile friendly and falls apart on a small screen.

1.4: Solution to the problems

* Always display the full price, include tax and all additional prices in it and indicate the shipping prices throughout the process, not only at the check-out and don’t have any hidden charges.
* Never sell products without offering a guarantee it will seem suspicious and plant uncertainty in the customer’s mind.
* Let your customers review and rate your products and if the reviews are bad, check what is wrong instead of covering it up, because they will quickly erode trust in

your brand and become one of the reasons why customers don’t buy from you.

* Personalize as much you can or even on search result pages.
* Make sure that your hosting provider is up to the job and that you have the appropriate service for the number of visitors your site experiences at peak volume.

1.5: Functional requirements

Functional requirements determine what system should.

Functional requirements are the following:

* Shopping o Create account o Allow customers to log in
* Allow customers to view available shoes o Display photos of available shoes
* Allow customers to give commands of their own design of shoes
* Allow customers to change choices o Allow customers to cancel their commands or orders
* Allow clients to view prices of shoes o Allow clients to log out
* Allow clients to print receipt

1.6: Nonfunctional requirements

Nonfunctional requirements determine how or to what extent something should be done.

Nonfunctional requirements are the following:

* + System should be easy to use it
  + Reliable
  + Security
  + Portable
  + Interactive design

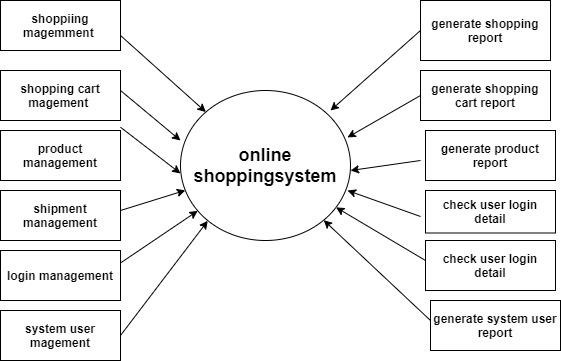
1.7: Data Flow Diagram

Data flow diagram is a way of representing a flow of data through a process or a system. DFD also provides information about outputs and inputs of each entity and the process itself. A data flow diagram has no control flow. There are no rules and no loops.

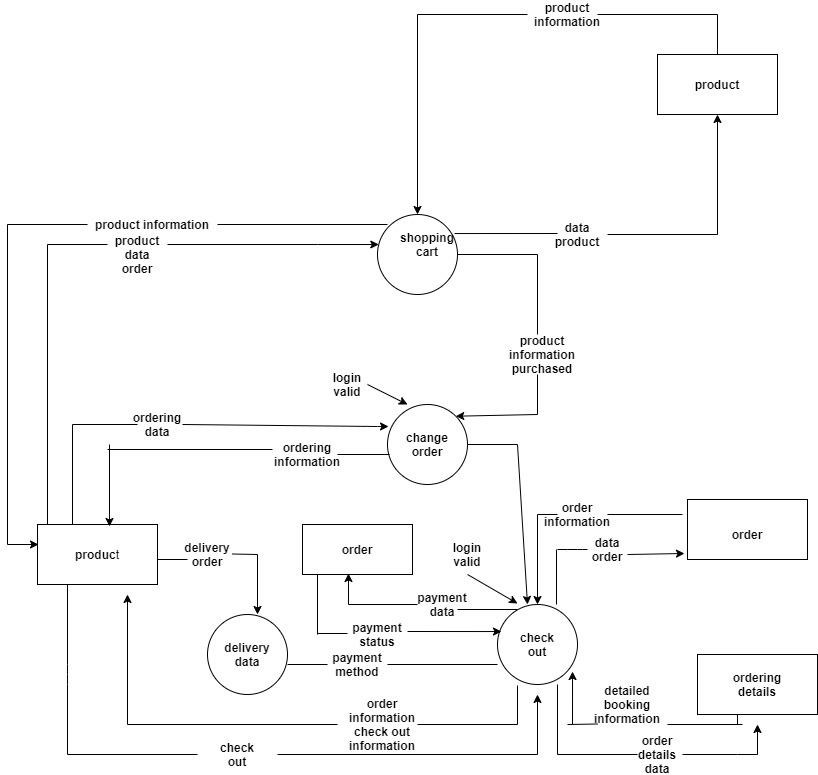
1.7.1: DFD Level 0



1.7.2: DFD Level 1

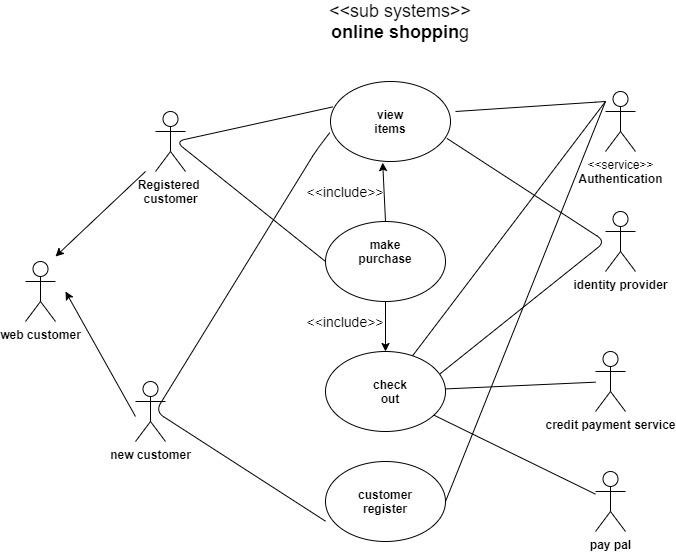


1.7.3: DFD Level 2



1.8: Use case diagram

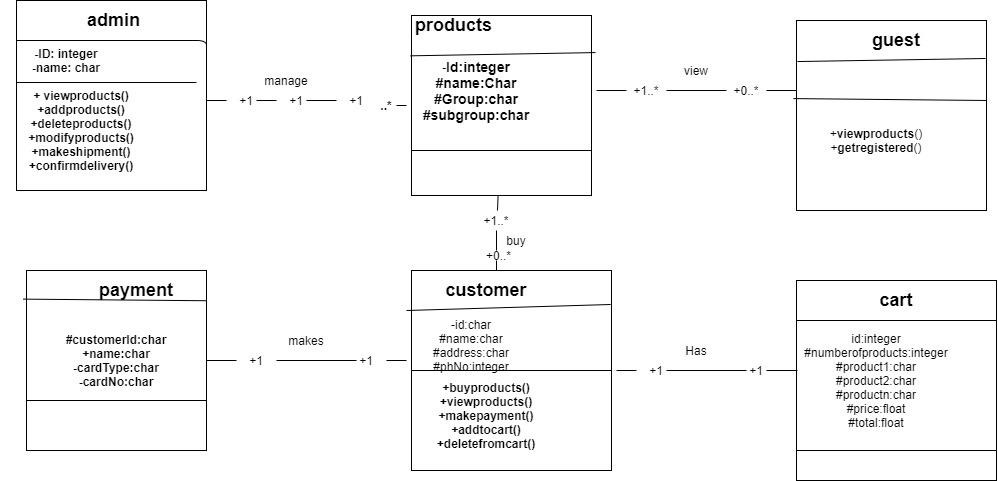
A use case diagram is a graphical depiction of a user’s possible interactions with a system. A use case diagram shows the various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases represented by either circles or ellipses.



1.8: Class diagram

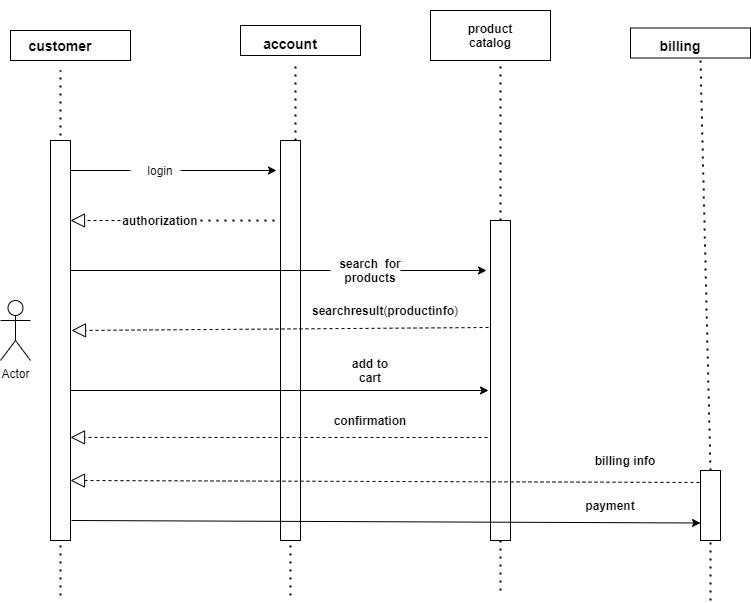
A class diagram in the unified modeling language is a type of static structure diagram that describes the structure of a

system by showing the system’s classes, their attributes, operation and the relationships among objects.



1.9: Sequence diagram

A sequence diagram or system diagram shows process interactions arranged in time sequence in the field of software engineering. it depicts the processes involved and the sequence of messages exchanged between the processes needed to carry out the functionality.



**PART 2: DATABASE MANAGEMENT SYSTEM(DBMS)**

Database management system are software systems used to store, retrieve and run queries on data.

In this Uzuri online shopping system the DBMS will serve as an interface between an end-user and a database, allowing users to create, read, update, and delete data in the database.

this database will contain different entities/tables like customers, admin, products, cart, payment (transactions), order and each entity will be having their correspondence attributes hence make database functionable.

**Section 1: ENTITY, ATTRIBUTES AND RELATIONSHIPS**

**1.1 ENTITY AND THEIR ATTRIBUTES**

1. **ADMIN ENTITY**

This is the DBMS entity which will help Admin to manage all access to the database like view product, add product, delete product, modify product, make shipment and confirming delivery. The following are attributes:

* ID: this attribute will store admin identity and primary key of this table
* Username: this attribute will store admin username in table
* Password: this attribute will help admin to login using password

1. **CUSTOMERS ENTITY**

This is one of DBMS entity which will help in recording different customer’s information using the following attributes:

* Customer\_id: this attributes will store customers identity number and it represent primary key in customer table.
* Customer\_names: this attribute will store customers names in customer table.
* Phone\_number: this attribute will store customers telephone number in customer table.
* location: this attribute will store customers address or location in customer table.
* Email: this attribute will store email address of customer
* Username: this attribute will store username of customer to help him/her to login
* Password: this attribute will store password of customer to help him/her to login

1. **PRODUCTS ENTITY**

The product table stores detail about products we intend to offer our clients. The attributes are:

* Pro\_Id: this attribute will be primary key of this entity
* Pro\_Name: this attribute will store the name of product
* Pro\_Price: this attribute will store the price of product
* Pro\_Category: this attribute will store category of product
* Pro\_Quantity: this attribute will store quantity of product
* Pro\_quality: this attribute will store quality of product

1. **PAYMENT ENTITY**

This table holds the information for payments made. The attributes are:

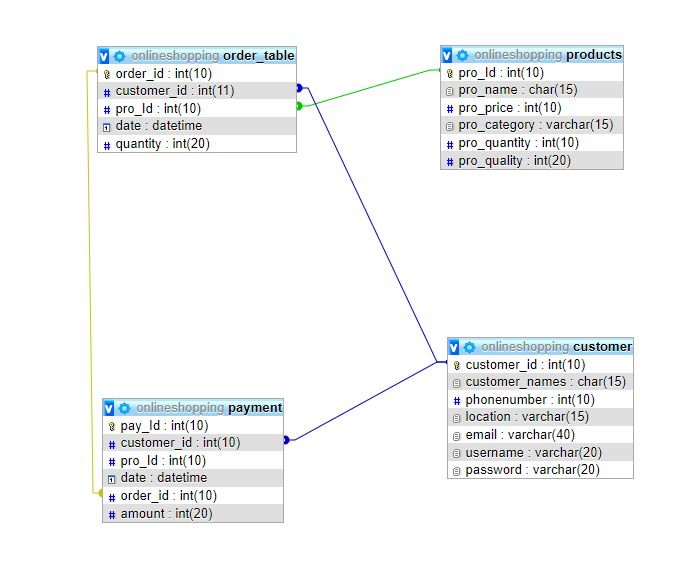
* Pay\_Id: this attribute will be primary key of this entity
* Customer id: this attribute will be foreign key from customer table
* Customer\_id:this attribute will store customer id as foreign key
* Pro\_Id:this attribute will be foreign key from product table
* Date:this attribute will store date and time of payment
* Order\_id:this attribute will store foreign key from order table
* Amount:this attribute will store amount customer

Paid

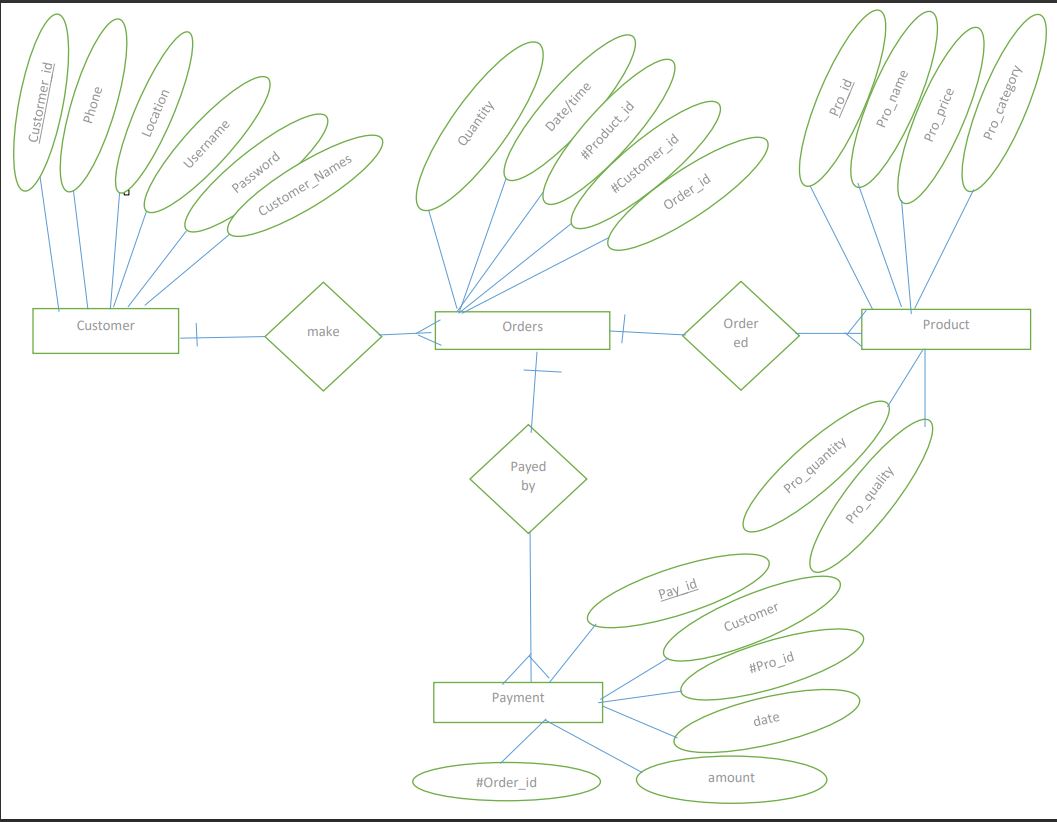
1. **ORDER ENTITY**

This table holds information made by customers.the attributes are:

* Order-id:this attribute will be primary key of this entity
* Date:this attribute will store date and time when orders are made
* Customer\_id: attribute will store customer id as foreign key
* Pro\_Id: this attribute will be foreign key from product table
* Quantity: this attribute will store quantity of product

**Logical data mode**

**Entity relationship diagram**

****

**Querries**

**Enter password:**

**Welcome to the MySQL monitor. Commands end with ; or \g.**

**Your MySQL connection id is 1**

**Server version: 5.5.24-log MySQL Community Server (GPL)**

**Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.**

**Oracle is a registered trademark of Oracle Corporation and/or its**

**affiliates. Other names may be trademarks of their respective**

**owners.**

**Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.**

**mysql> create database uzurionlineshopping;**

**Query OK, 1 row affected (0.00 sec)**

**mysql> show databases;**

**+---------------------+**

**| Database |**

**+---------------------+**

**| information\_schema |**

**| mysql |**

**| performance\_schema |**

**| test |**

**| uzurionlineshopping |**

**+---------------------+**

**6 rows in set (0.00 sec)**

**mysql> use uzurionlineshopping;**

**Database changed**

**mysql> create table admin(ID int(10) primary key not null,username char(15),password varchar(20));**

**Query OK, 0 rows affected (0.09 sec)**

**mysql> create table customer(customer\_id int(10) primary key not null,customer\_names char(15),phonenumber int(10),location varchar(15),email varchar(10),username varchar(20),password varchar(20));**

**Query OK, 0 rows affected (0.08 sec)**

**mysql> create table products(pro\_Id int(10) primary key not null,pro\_name char(15),pro\_price int(10),pro\_category varchar(15),pro\_quantity int(10),pro\_quality int(20));**

**Query OK, 0 rows affected (0.08 sec)**

**mysql> create table payment(pay\_Id int(10) primary key not null,customer\_id char(15),pro\_Id int(10),date datetime,order\_id int(10),amount int(20));**

**Query OK, 0 rows affected (0.17 sec)**

**mysql> create table order\_table(order\_id int(10) primary key not null,customer\_id char(15),pro\_Id int(10),date datetime,quantity int(20));**

**Query OK, 0 rows affected (0.09 sec)**

**Relationship**

**Constraints for table order\_table**

**ALTER TABLE `order\_table`**

**ADD CONSTRAINT `order\_table\_ibfk\_1` FOREIGN KEY (`pro\_Id`) REFERENCES `products` (`pro\_Id`) ,**

**ADD CONSTRAINT `order\_table\_ibfk\_2` FOREIGN KEY (`customer\_id`) REFERENCES `customer` (`customer\_id`);**

**Constraints for table payment**

**ALTER TABLE `payment`**

**ADD CONSTRAINT `payment\_ibfk\_1` FOREIGN KEY (`customer\_id`) REFERENCES `customer` (`customer\_id`),**

**ADD CONSTRAINT `payment\_ibfk\_2` FOREIGN KEY (`order\_id`) REFERENCES `order\_table` (`order\_id`);**

**mysql> INSERT INTO admin(ID,username,password) VALUES ('1', 'mimi','12345');**

**Query OK, 1 row affected (0.08 sec)**

**mysql> INSERT INTO customer(customer\_id, customer\_names,phonenumber,location,email,username,password) VALUES ('1', 'aliane', '0789267133', 'huye','alianeumutesi@gmail.com','umutesialiane','67898');**

**Query OK, 1 row affected (0.06 sec)**

**mysql> INSERT INTO products (pro\_Id, pro\_name, pro\_price,pro\_category,pro\_quantity,pro\_quality) VALUES ('1', 'shoes', '15000', 'open shoes', '2', '1');**

**Query OK, 1 row affected (0.07 sec)**

**mysql> INSERT INTO payment(pay\_Id,customer\_id,pro\_Id, date, order\_id, amount) VALUES ('1', '1', '1', '2022-07-12 00:00:00', '1', '30000');**

**Query OK, 1 row affected (0.09 sec)**

**mysql> INSERT INTO order\_table(order\_id,customer\_id,pro\_Id,date,quantity) VALUES ('1', '1', '1', '2022-07-12 00:00:00', '2');**

**Query OK, 1 row affected (0.08 sec)**

**mysql> select \* from admin;**

**+----+----------+----------+**

**| ID | username | password |**

**+----+----------+----------+**

**| 1 | mimi | 12345 |**

**+----+----------+----------+**

**1 row in set (0.13 sec)**

**mysql> select \* from customer;**

**+-------------+----------------+-------------+----------+-------------------------+---------------+----------+**

**| customer\_id | customer\_names | phonenumber | location | email | username | password |**

**+-------------+----------------+-------------+----------+-------------------------+---------------+----------+**

**| 1 | aliane | 789267133 | huye | alianeumutesi@gmail.com | umutesialiane | 67898 |**

**+-------------+----------------+-------------+----------+-------------------------+---------------+----------+**

**1 row in set (0.05 sec)**

**mysql> select \* from products;**

**+--------+----------+-----------+--------------+--------------+-------------+**

**| pro\_Id | pro\_name | pro\_price | pro\_category | pro\_quantity | pro\_quality |**

**+--------+----------+-----------+--------------+--------------+-------------+**

**| 1 | shoes | 15000 | open shoes | 2 | 1 |**

**+--------+----------+-----------+--------------+--------------+-------------+**

**1 row in set (0.02 sec)**

**mysql> select \* from payment;**

**+--------+-------------+--------+---------------------+----------+--------+**

**| pay\_Id | customer\_id | pro\_Id | date | order\_id | amount |**

**+--------+-------------+--------+---------------------+----------+--------+**

**| 1 | 1 | 1 | 2022-07-12 00:00:00 | 1 | 30000 |**

**+--------+-------------+--------+---------------------+----------+--------+**

**1 row in set (0.02 sec)**

**mysql> select \* from order\_table;**

**+----------+-------------+--------+---------------------+----------+**

**| order\_id | customer\_id | pro\_Id | date | quantity |**

**+----------+-------------+--------+---------------------+----------+**

**| 1 | 1 | 1 | 2022-07-12 00:00:00 | 2 |**

**+----------+-------------+--------+---------------------+----------+**

**1 row in set (0.04 sec)**

**UPDATE admin SET username = 'vanessa' WHERE admin.ID = 1;**

**Query OK, 0 rows affected (0.06 sec)**

**mysql> UPDATE customer SET location = 'gatsibo' WHERE customer.customer\_id = 1;**

**Query OK, 0 rows affected (0.07 sec)**

**mysql> UPDATE products SET pro\_quantity = '3' WHERE products.pro\_Id = 1;**

**Query OK, 1 row affected (0.07 sec)**

**mysql> UPDATE payment SET amount ='45000' WHERE payment.pay\_Id = 1;**

**Query OK, 1 row affected (0.07 sec)**

**mysql> UPDATE order\_table SET quantity ='3' WHERE order\_table.order\_id = 1;**

**Query OK, 1 row affected (0.07 sec)**

**mysql> create view admin\_view as select \* from admin;**

**Query OK, 0 rows affected (0.04 sec)**

**mysql> create view customer\_view as select \* from customer;**

**Query OK, 0 rows affected (0.02 sec)**

**mysql> create view products\_view as select \* from products;**

**Query OK, 0 rows affected (0.14 sec)**

**mysql> create view payment\_view as select \* from payment;**

**Query OK, 0 rows affected (0.06 sec)**

**mysql> create view order\_table\_view as select \* from order\_table;**

**Query OK, 0 rows affected (0.11 sec)**

**mysql> INSERT INTO admin\_view (ID,username,password) VALUES ('2', 'anisia', '2020');**

**Query OK, 1 row affected (0.08 sec)**

**mysql> INSERT INTO customer\_view (customer\_id,customer\_names,phonenumber,location,email,username,password) VALUES ('2', 'anny', '07844356', 'gisenyi', 'anny@gmail.com', 'annymutoni', '30500');**

**Query OK, 1 row affected (0.07 sec)**

**mysql> UPDATE customer\_view SET phonenumber = '0789267133',location = 'nyamasheke' WHERE customer\_view.customer\_id = 2;**

**Query OK, 0 rows affected (0.07 sec)**

**mysql> UPDATE order\_table\_view SET quantity = '5' WHERE order\_table\_view.order\_id = 1;**

**Query OK, 0 rows affected (0.07 sec)**

**mysql> delete from admin\_view where ID='2';**

**Query OK, 1 row affected (0.08 sec)**

**mysql> delete from products\_view where pro\_Id='1';**

**Query OK, 1 row affected (0.07 sec)**

**mysql> create view subquerry as select \* from products where pro\_quantity <=(select AVG(pro\_quantity) from products);**

**Query OK, 0 rows affected (0.04 sec)**

**Section 4**

**CREATE PROCEDURE admin\_insert (IN `ID` INT(10), IN password CHAR(15), IN `username` VARCHAR(20))**

**insert into admin (ID, password, username) values (ID,password, username)**

**create procedure admin\_display ()**

**select \*from admin**

**CREATE PROCEDURE `admin\_update`(INOUT `username` CHAR(20), INOUT `ID` INT(10))**

**update admin set username=username where admin.ID=ID**

**CREATE PROCEDURE `admin\_delete`(INOUT `ID` INT(10))**

**delete from admin where admin.ID=ID**

**section 5**

**create trigger insert\_trigger\_to\_order\_table**

**after**

**insert on customer**

**for each ROW**

**insert into order\_table (new.customer\_id) values (new.customer\_id)**

**CREATE TRIGGER `customer\_update\_trigger`**

**AFTER UPDATE ON `customer`**

**FOR EACH ROW**

**update order\_table set customer\_id=customer\_id where customer.customer\_id=customer\_id**

**CREATE TRIGGER `delete customer trigger` AFTER DELETE ON `customer` FOR EACH ROW delete from customer where customer.customer\_id=customer\_id;**

**Section 6**

**create user 'umurerwa mireille’identified by '220013685'**

**grant all PRIVILEGES on \*.\* to 'umurerwa mireille’**

**CREATE USER 'aline-semi' identified by '220013685'**

**GRANT INSERT,UPDATE,DELETE ON \*.\* TO ' mireille -semi';**

**revoke ALL PRIVILEGES on \*.\* from ' mireille -semi'**

**REVOKE GRANT OPTION on \*.\* from ' mireille -semi'**

**REVOKE INSERT \*.\* from 'mireille-semi'**