DBMS Mini Project 2022:

Shipment Tracking System

SUBMITTED BY:

AYUSHA PRIYADARSHANI, 211627005

HARIKA KONDUR, 211627016

Acknowledgement

We are thankful for the help and cooperation of the college authorities, our DBMS teacher, Mr. Manmohana Krishna for the successful completion of our investigatory project, ”Shipment Tracking System‟

We are also thankful to the ICAS, Manipal for letting us utilize their resources and lab to complete our project.

We would also like to express our gratitude to our parents and classmates for their well-meant support and for giving us a helping hand whenever required.

Table of Contents

INTRODUCTION

From factories to storage units to customers’ doorstep, tons of packages are shipped daily. Therefore, integrating a reliable system for tracking every delivery is crucial for e-commerce success. We aim to build a software that enables clients to efficiently register and track their shipments online, increasing all businesses' efficiency and productivity.

OBJECTIVES

Customer:

- Track shipment
- View previous shipment details (if any)

Seller:

- Input customer, package details
- Update shipment details
- Track shipment
- View past customers
- View past packages

Hardware and Software Requirements

Software Requirements:

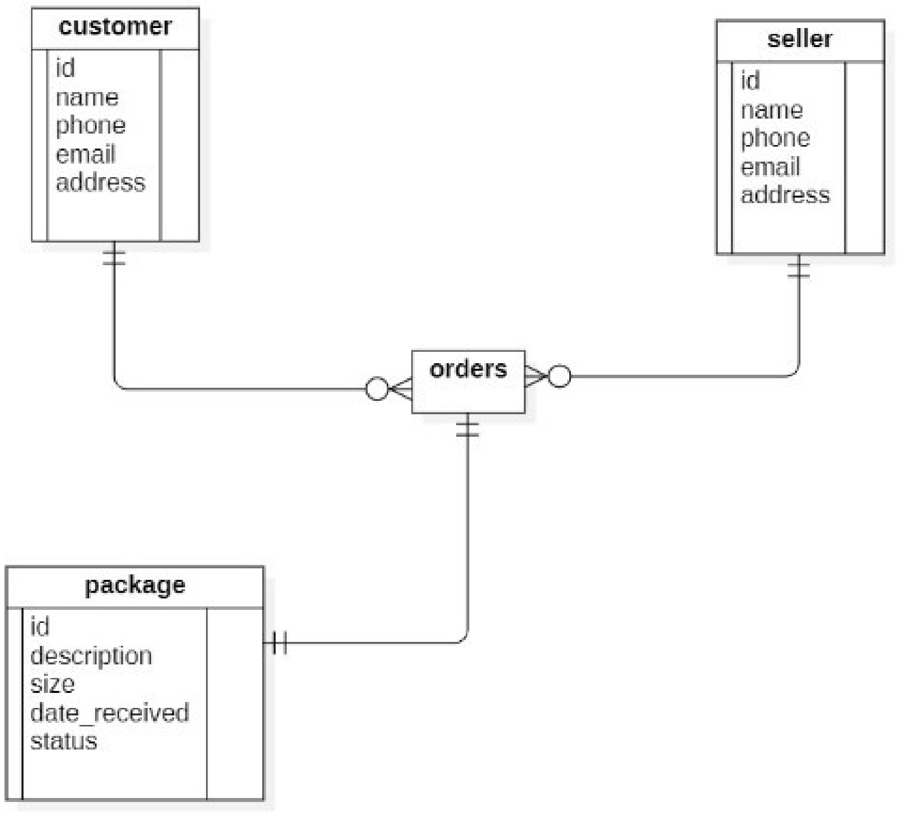
Language: Python
Versions: 3.6.X.
Database: MySQL Server
Frontend Implementation: PyQt

Hardware Requirements:

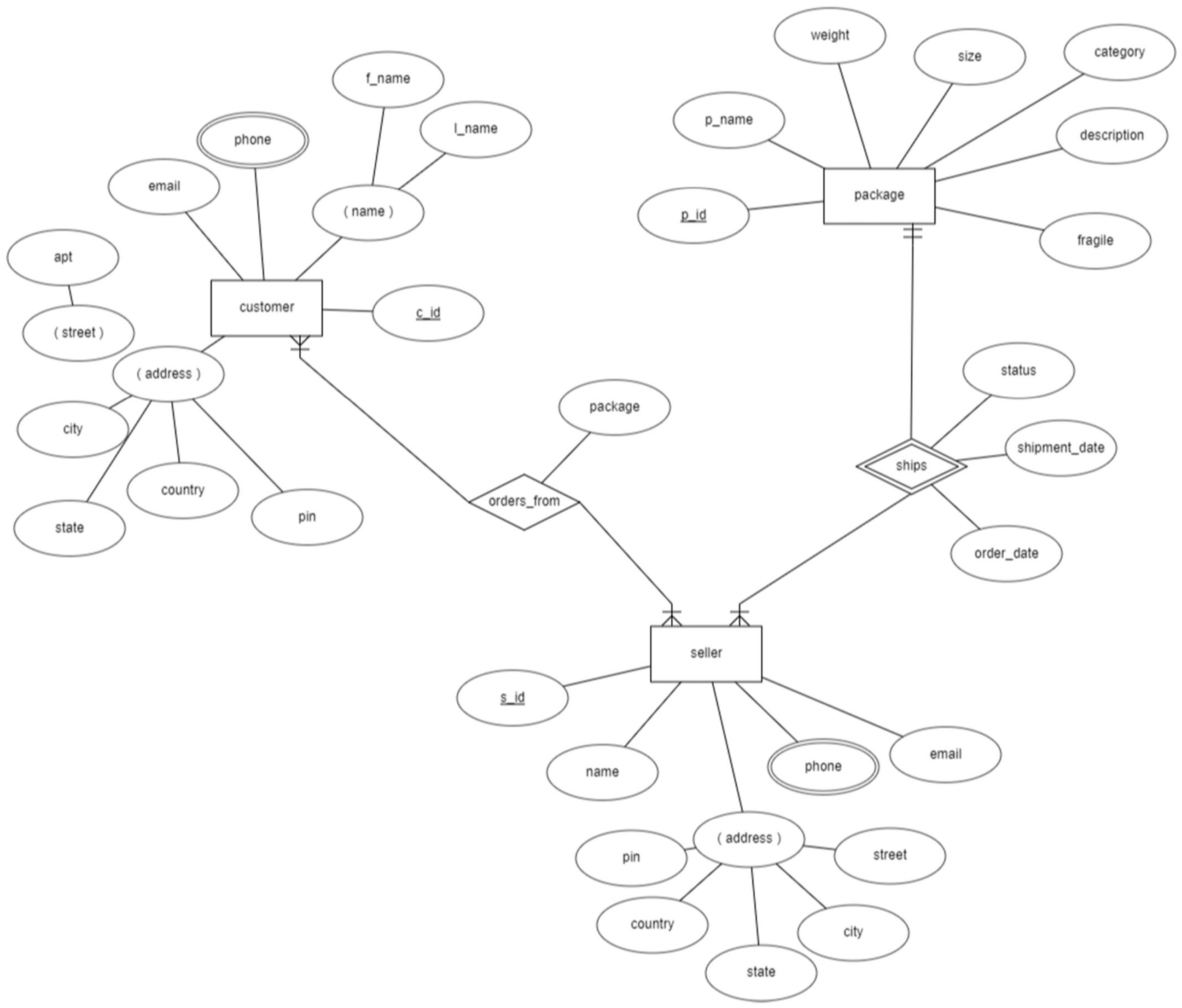
PLATFORM
Processor: Minimum 1 GHz; Recommended 2GHz or more.
RAM: Minimum 1 GB; Recommended 4 GB or above.
Disk space: 1 GB.
Operating systems: Windows* 7 or later, macOS, and Linux.
Display Option: Monitor
Input Devices: Keyboard, Mouse (optional)

DESIGNS

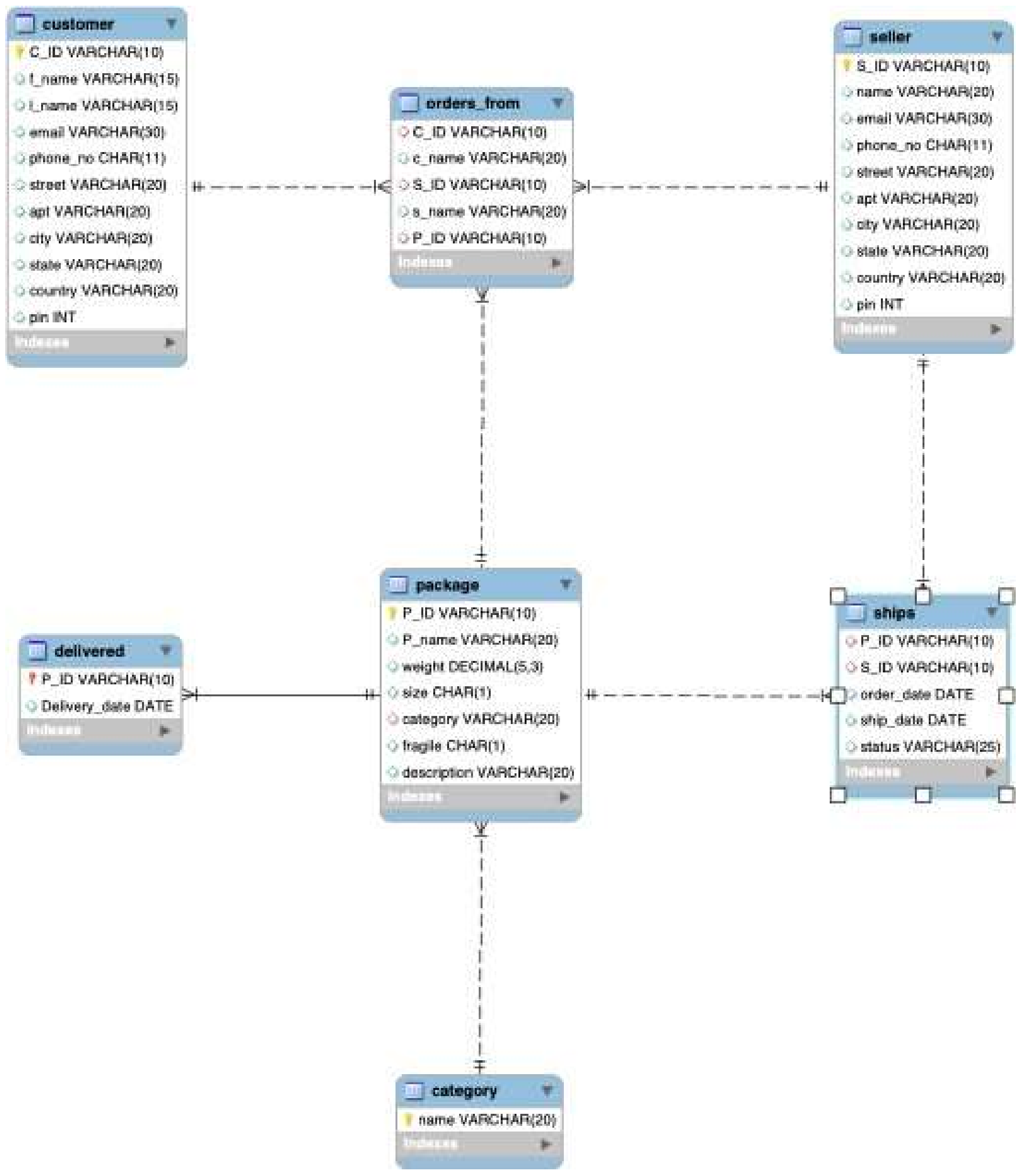
Conceptual Model:



Initial ER Diagram



Initial Database Schema



Normalization Process

Upon analysis of our first database schema, it was observed that the schema's atomicity could be compromised if a customer or seller had more than one phone number. It was also observed that the seller and customer tables were cluttered with address-related attributes.

C_ID	f_name	l_name	email	phone_no	apt	street	city	State	country	pin
2022000010	John	Smith	jsmith@gmail.com	9456772315, 9722975883	101	MG Road	Manipal	Karnataka	India	576104

customer table before normalization

To solve these issues, we first converted the respective tables into first normal form (1NF) by creating a customer phone table to remove the possibility of a multivalued row. In addition, the customer table was further decomposed to form a separate address table to declutter the essential attributes present in the main customer details. The same was applied to the seller table.

customer

C_ID	f_name	l_name	email
2022000010	John	Smith	jsmith@gmail.com

C_phone

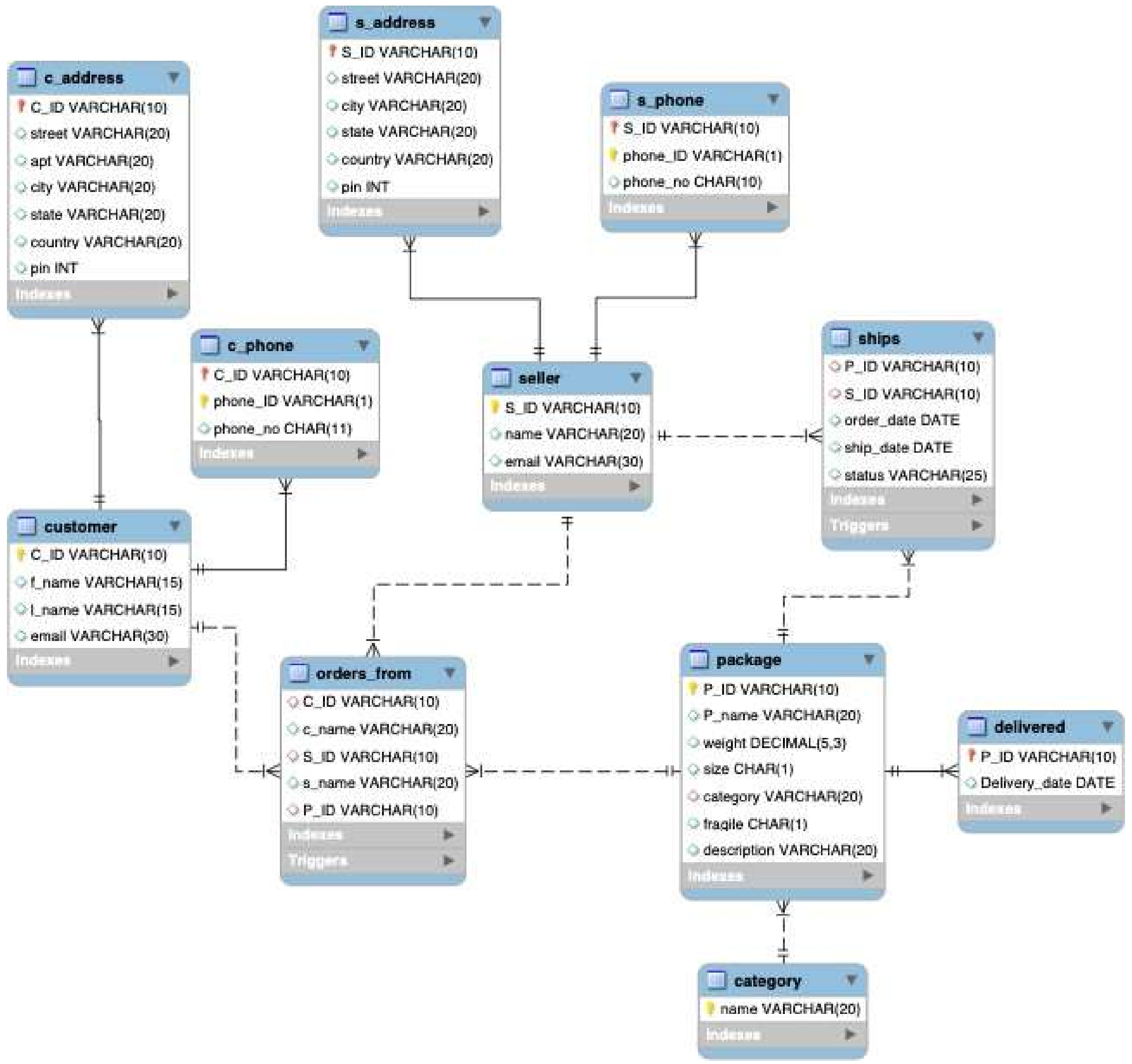
C_ID	phone_id	phone_no
2022000010	1	9456772315
2022000010	2	9722975883

C_address

C_ID	apt	street	city	state	country	pin
2022000010	101	MG Road	Manipal	Karnataka	India	576104

tables after normalization and decomposition

Final ER Diagram:

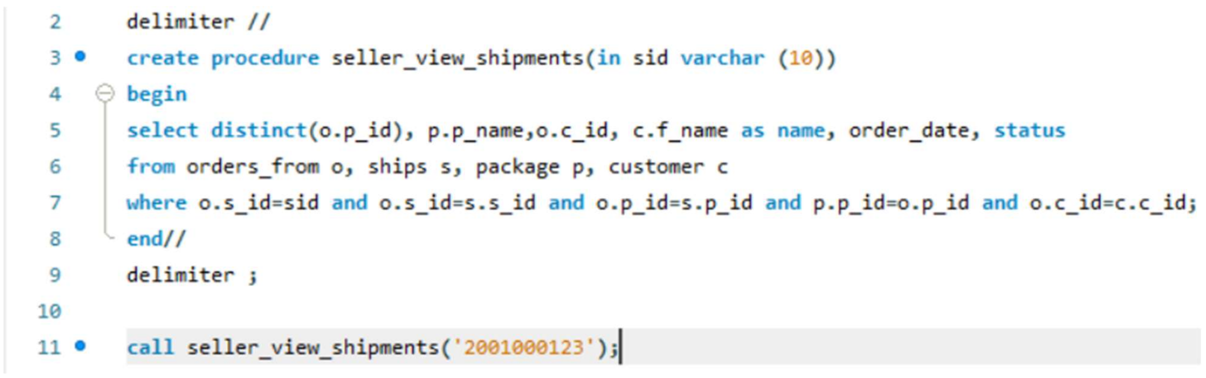


IMPLEMENTATION

Stored Procedures:

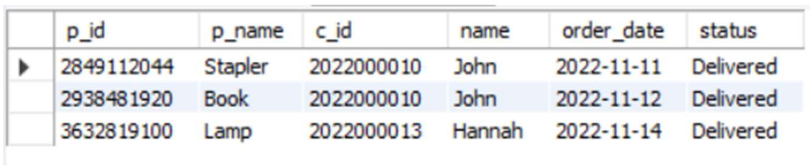
Procedure 1: seller_view_shipments()

- Sellers can view packages shipped by their company



```
delimiter //
create procedure seller_view_shipments(in sid varchar (10))
begin
select distinct(o.p_id), p.p_name,o.c_id, c.f_name as name, order_date, status
from orders_from o, ships s, package p, customer c
where o.s_id=sid and o.s_id=s.s_id and o.p_id=s.p_id and p.p_id=o.p_id and o.c_id=c.c_id;
end//
delimiter ;

call seller_view_shipments('2001000123');
```

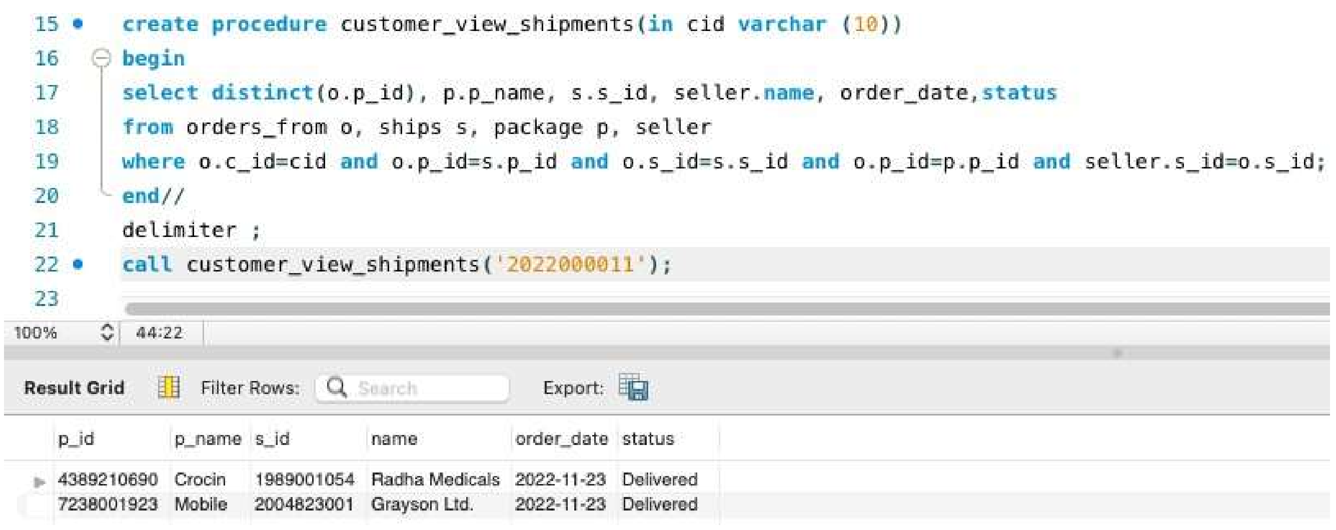
Result:



p_id	p_name	c_id	name	order_date	status
2849112044	Stapler	2022000010	John	2022-11-11	Delivered
2938481920	Book	2022000010	John	2022-11-12	Delivered
3632819100	Lamp	2022000013	Hannah	2022-11-14	Delivered

Procedure 2: customer_view_shipments()

- Customers can view packages they ordered by supplying their customer id

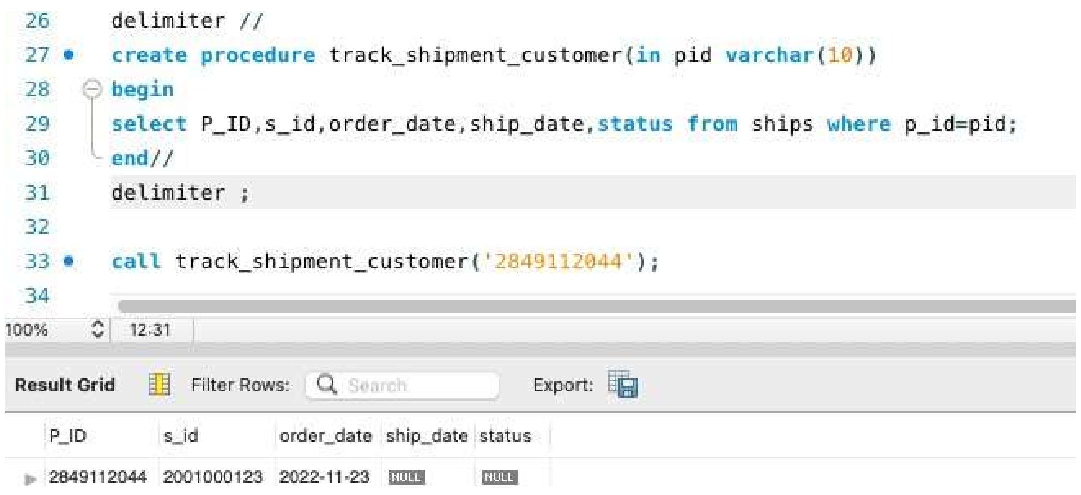


```
create procedure customer_view_shipments(in cid varchar (10))
begin
select distinct(o.p_id), p.p_name, s.s_id, seller.name, order_date,status
from orders_from o, ships s, package p, seller
where o.c_id=cid and o.p_id=s.p_id and o.s_id=s.s_id and o.p_id=p.p_id and seller.s_id=o.s_id;
end//
delimiter ;
call customer_view_shipments('2022000011');
```

p_id	p_name	s_id	name	order_date	status
4389210690	Crocin	1989001054	Radha Medicals	2022-11-23	Delivered
7238001923	Mobile	2004823001	Grayson Ltd.	2022-11-23	Delivered

Procedure 3: track_shipment_customer()

- Customers can track their shipments

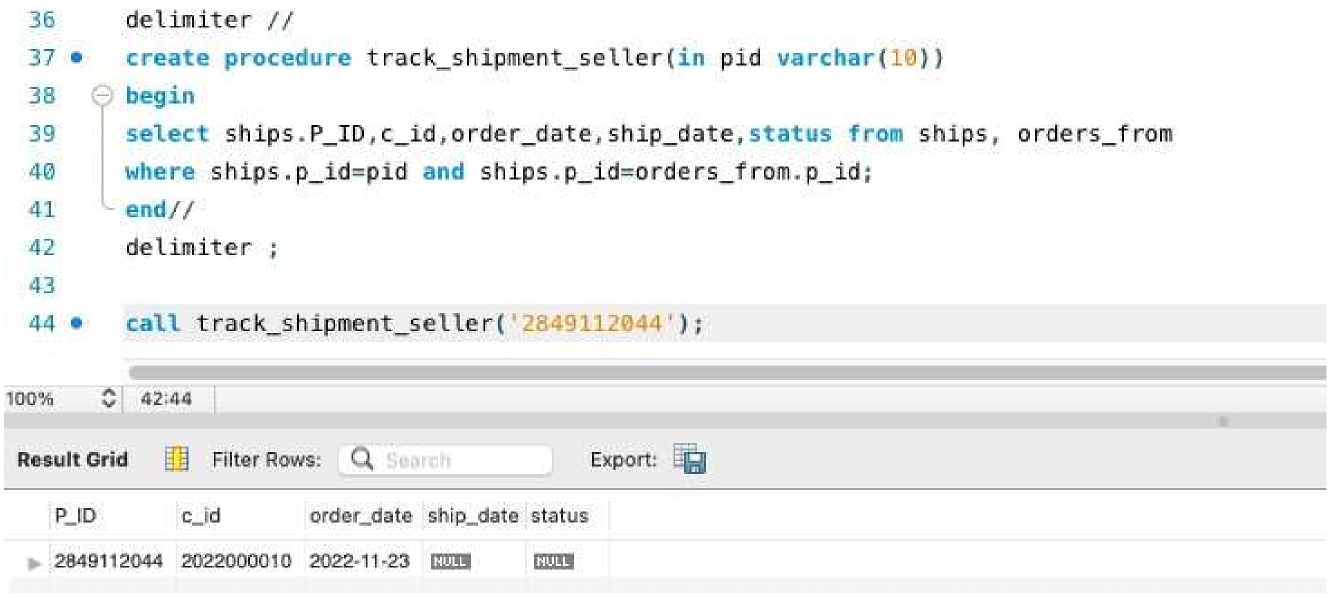


```
delimiter //
create procedure track_shipment_customer(in pid varchar(10))
begin
select P_ID,s_id,order_date,ship_date,status from ships where p_id=pid;
end//
delimiter ;

call track_shipment_customer('2849112044');
```

```
P_ID        s_id        order_date  ship_date  status
2849112044  2001000123  2022-11-23  NULL       NULL
```

Procedure 4: track_shipment_seller()

- Sellers can track their shipments

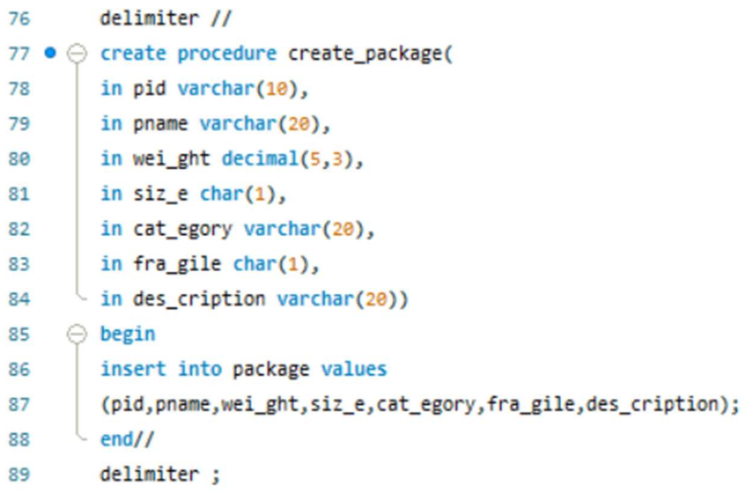


```
delimiter //
create procedure track_shipment_seller(in pid varchar(10))
begin
select ships.P_ID,c_id,order_date,ship_date,status from ships, orders_from
where ships.p_id=pid and ships.p_id=orders_from.p_id;
end//
delimiter ;

call track_shipment_seller('2849112044');
```

```
P_ID        c_id        order_date  ship_date  status
2849112044  2022000010  2022-11-23  NULL       NULL
```

Procedure 5: create_package()

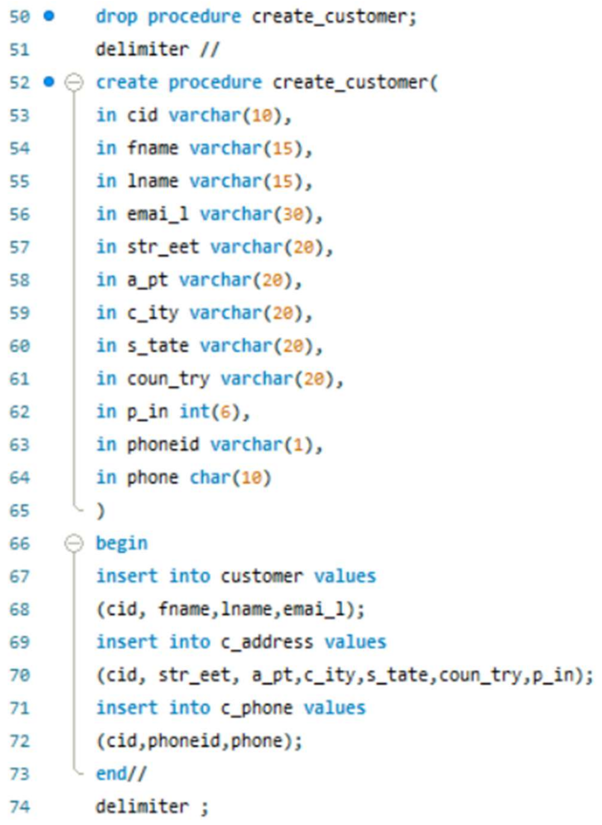
- Creates a package



```sql
delimiter //
create procedure create_package(
in pid varchar(10),
in pname varchar(20),
in wei_ght decimal(5,3),
in siz_e char(1),
in cat_egory varchar(20),
in fra_gile char(1),
in des_cription varchar(20))
begin
insert into package values
(pid,pname,wei_ght,siz_e,cat_egory,fra_gile,des_cription);
end//
delimiter ;
```

Procedure 6: create_customer()

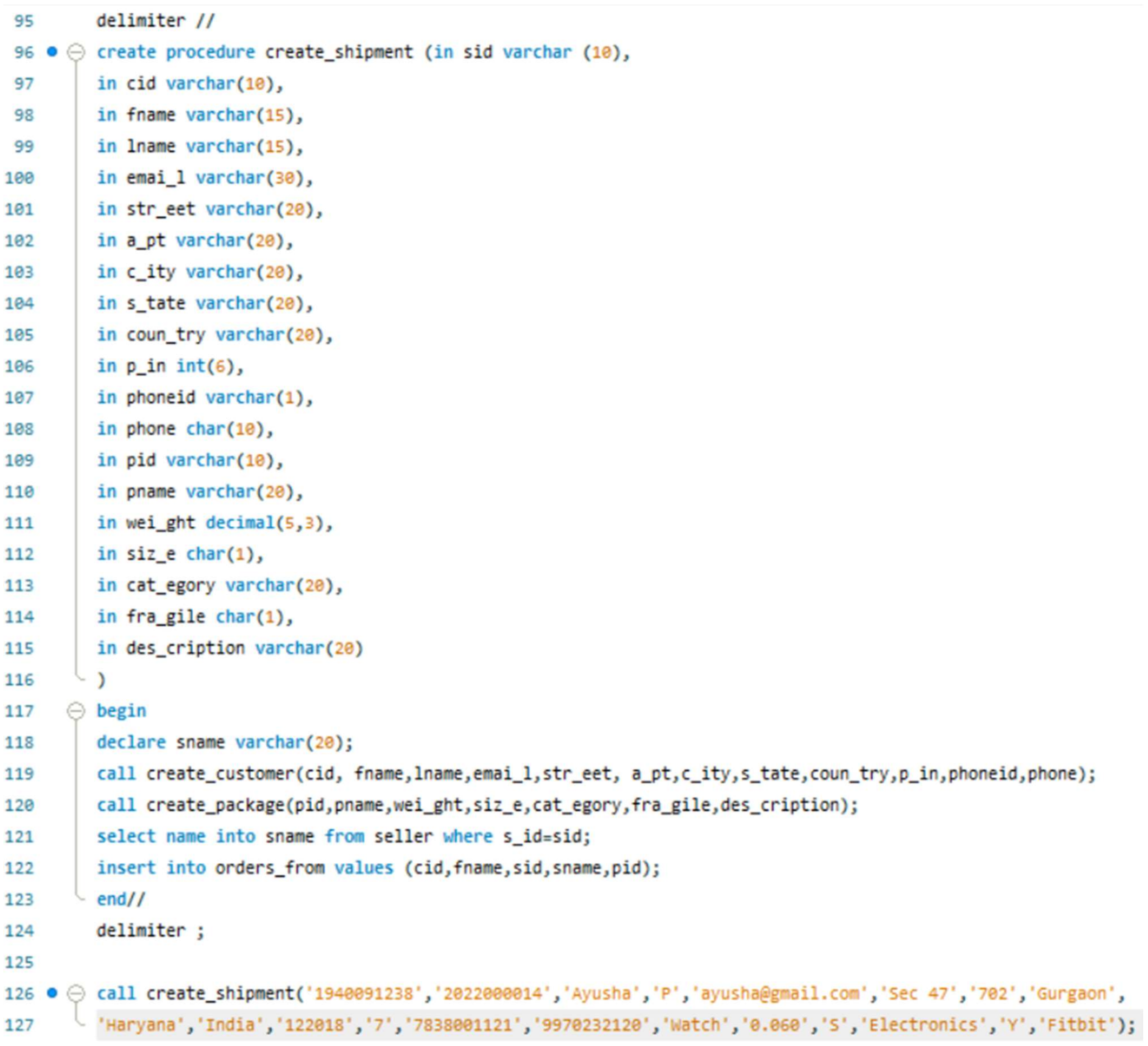
- Creates a customer profile



```sql
drop procedure create_customer;
delimiter //
create procedure create_customer(
in cid varchar(10),
in fname varchar(15),
in lname varchar(15),
in emai_l varchar(30),
in str_eet varchar(20),
in a_pt varchar(20),
in c_ity varchar(20),
in s_tate varchar(20),
in coun_try varchar(20),
in p_in int(6),
in phoneid varchar(1),
in phone char(10)
)
begin
insert into customer values
(cid, fname,lname,emai_l);
insert into c_address values
(cid, str_eet, a_pt,c_ity,s_tate,coun_try,p_in);
insert into c_phone values
(cid,phoneid,phone);
end//
delimiter ;
```

Procedure 7: create_shipment()

- Creates a shipment by calling procedure 4 and 5

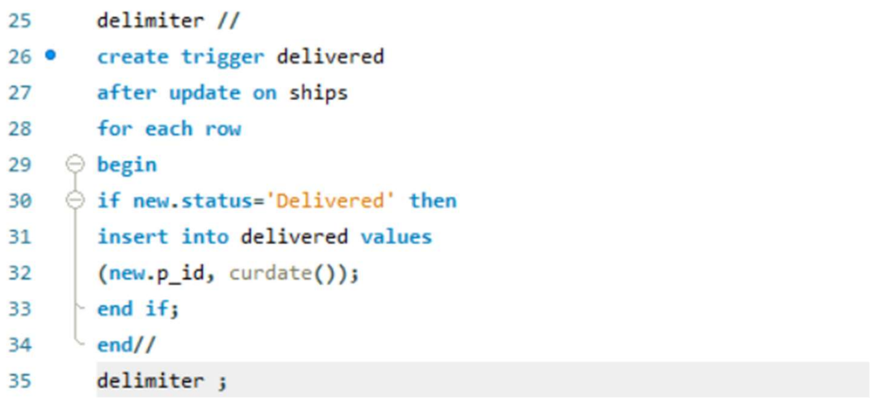


```sql
delimiter //
create procedure create_shipment (in sid varchar (10),
in cid varchar(10),
in fname varchar(15),
in lname varchar(15),
in emai_l varchar(30),
in str_eet varchar(20),
in a_pt varchar(20),
in c_ity varchar(20),
in s_tate varchar(20),
in coun_try varchar(20),
in p_in int(6),
in phoneid varchar(1),
in phone char(10),
in pid varchar(10),
in pname varchar(20),
in wei_ght decimal(5,3),
in siz_e char(1),
in cat_egory varchar(20),
in fra_gile char(1),
in des_cription varchar(20)
)
begin
declare sname varchar(20);
call create_customer(cid, fname,lname,emai_l,str_eet, a_pt,c_ity,s_tate,coun_try,p_in,phoneid,phone);
call create_package(pid,pname,wei_ght,siz_e,cat_egory,fra_gile,des_cription);
select name into sname from seller where s_id=sid;
insert into orders_from values (cid,fname,sid,sname,pid);
end//
delimiter ;

call create_shipment('1940091238','2022000014','Ayusha','P','ayusha@gmail.com','Sec 47','702','Gurgaon',
'Haryana','India','122018','7','7838001121','9970232120','Watch','0.060','S','Electronics','Y','Fitbit');
```

Triggers:

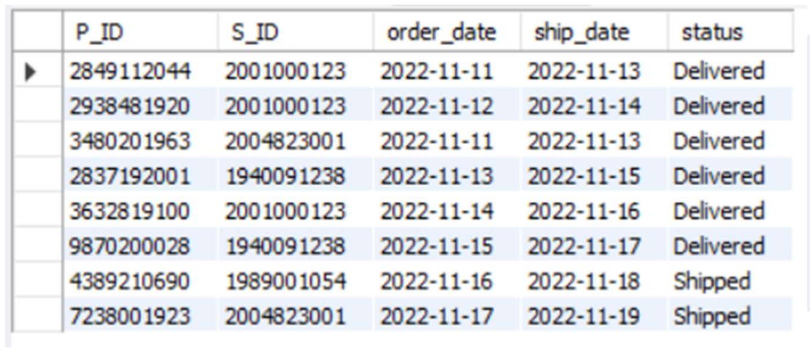
Trigger 1: delivered

- Automatically inserts record into delivered table after package status is updated to "Delivered"

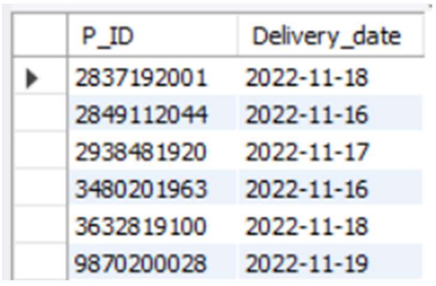


```
delimiter //
create trigger delivered
after update on ships
for each row
begin
if new.status='Delivered' then
insert into delivered values
(new.p_id, curdate());
end if;
end//
delimiter ;
```

Ships table and Delivered packages table before trigger

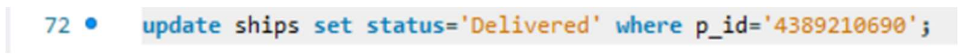


P_ID	S_ID	order_date	ship_date	status
2849112044	2001000123	2022-11-11	2022-11-13	Delivered
2938481920	2001000123	2022-11-12	2022-11-14	Delivered
3480201963	2004823001	2022-11-11	2022-11-13	Delivered
2837192001	1940091238	2022-11-13	2022-11-15	Delivered
3632819100	2001000123	2022-11-14	2022-11-16	Delivered
9870200028	1940091238	2022-11-15	2022-11-17	Delivered
4389210690	1989001054	2022-11-16	2022-11-18	Shipped
7238001923	2004823001	2022-11-17	2022-11-19	Shipped

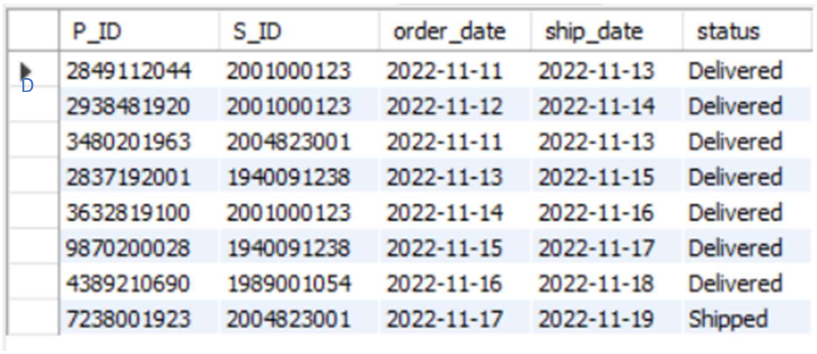


P_ID	Delivery_date
2837192001	2022-11-18
2849112044	2022-11-16
2938481920	2022-11-17
3480201963	2022-11-16
3632819100	2022-11-18
9870200028	2022-11-19

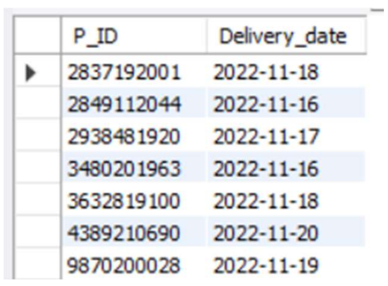
Ships table Delivered packages table after trigger



```
update ships set status='Delivered' where p_id='4389210690';
```



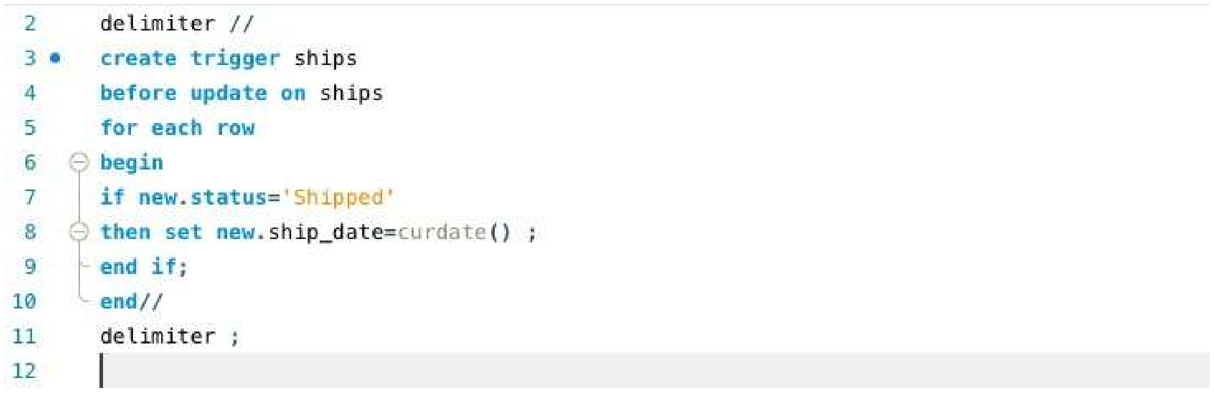
P_ID	S_ID	order_date	ship_date	status
2849112044	2001000123	2022-11-11	2022-11-13	Delivered
2938481920	2001000123	2022-11-12	2022-11-14	Delivered
3480201963	2004823001	2022-11-11	2022-11-13	Delivered
2837192001	1940091238	2022-11-13	2022-11-15	Delivered
3632819100	2001000123	2022-11-14	2022-11-16	Delivered
9870200028	1940091238	2022-11-15	2022-11-17	Delivered
4389210690	1989001054	2022-11-16	2022-11-18	Delivered
7238001923	2004823001	2022-11-17	2022-11-19	Shipped



P_ID	Delivery_date
2837192001	2022-11-18
2849112044	2022-11-16
2938481920	2022-11-17
3480201963	2022-11-16
3632819100	2022-11-18
4389210690	2022-11-20
9870200028	2022-11-19

Trigger 2: ships

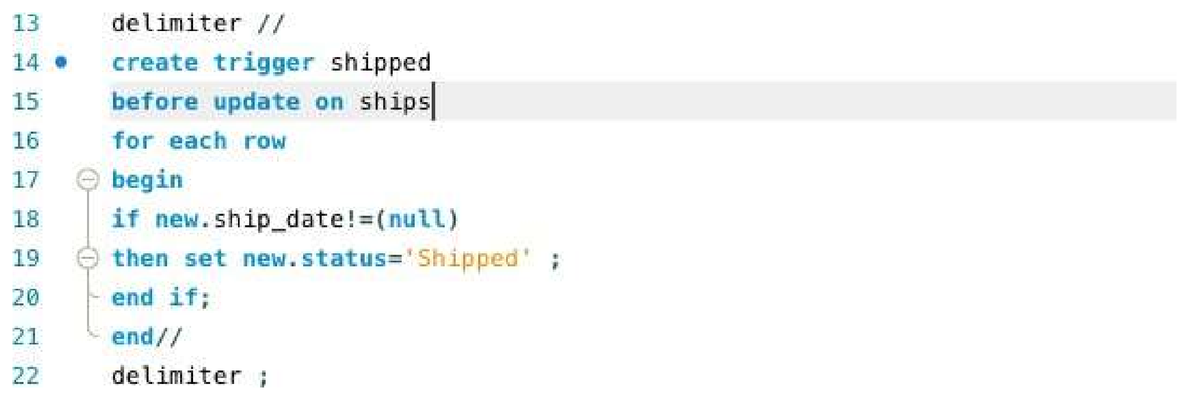
- Automatically adds the ship date of a package to ships table before the status of a package is set to ‘Shipped’



```
delimiter //
create trigger ships
before update on ships
for each row
begin
if new.status='Shipped'
then set new.ship_date=curdate() ;
end if;
end//
delimiter ;
```

Trigger 3: shipped

- Updates the status of the package to ‘Shipped’ once a ship date is entered by the seller



```
delimiter //
create trigger shipped
before update on ships
for each row
begin
if new.ship_date!=(null)
then set new.status='Shipped' ;
end if;
end//
delimiter ;
```

MySQL Connectivity

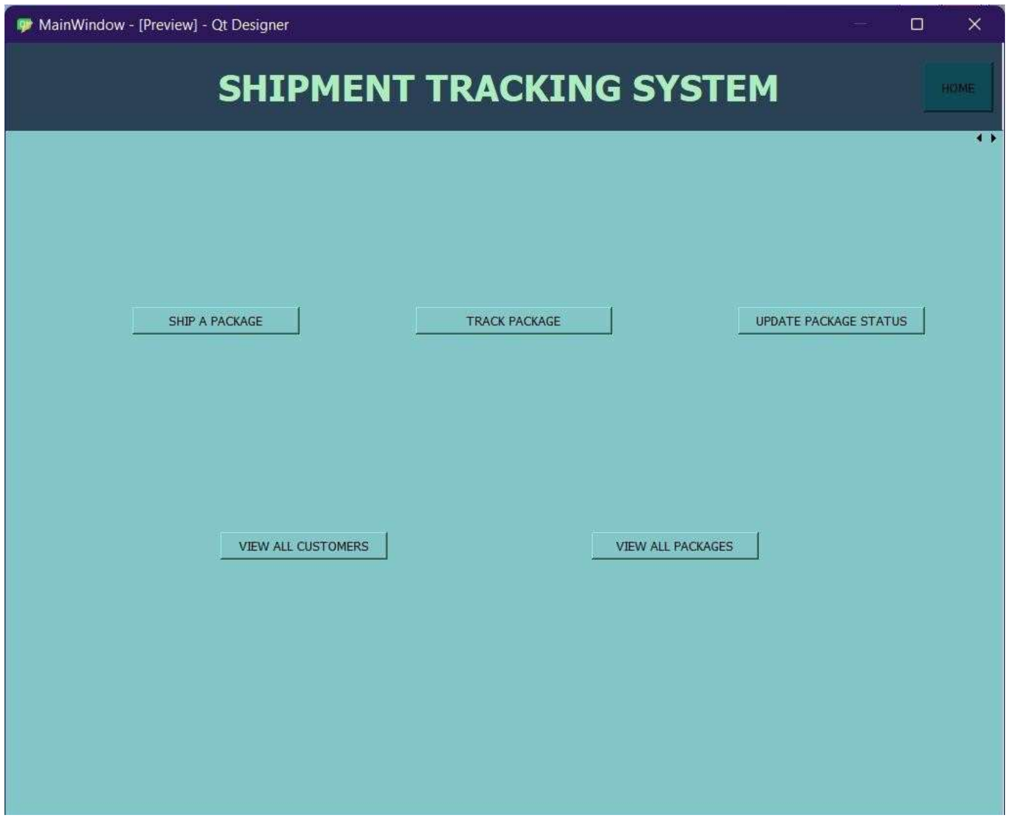
```
import mysql.connector

class DB:
   def connectToDatabase(self):
      try:
         self.db = mysql.connector.connect(
         host='localhost',user='root',password='  ',database='mini'
         )
         self.dbcursor = self.db.cursor()
         self.db.autocommit = True

         print("Connected to Database Successfully")

         return self.dbcursor

      except Exception as e:
         print("Error connecting to database")
         print(e)
         quit(-1)

   def _init_(self):
      self.connectToDatabase()
```

Output Snapshots

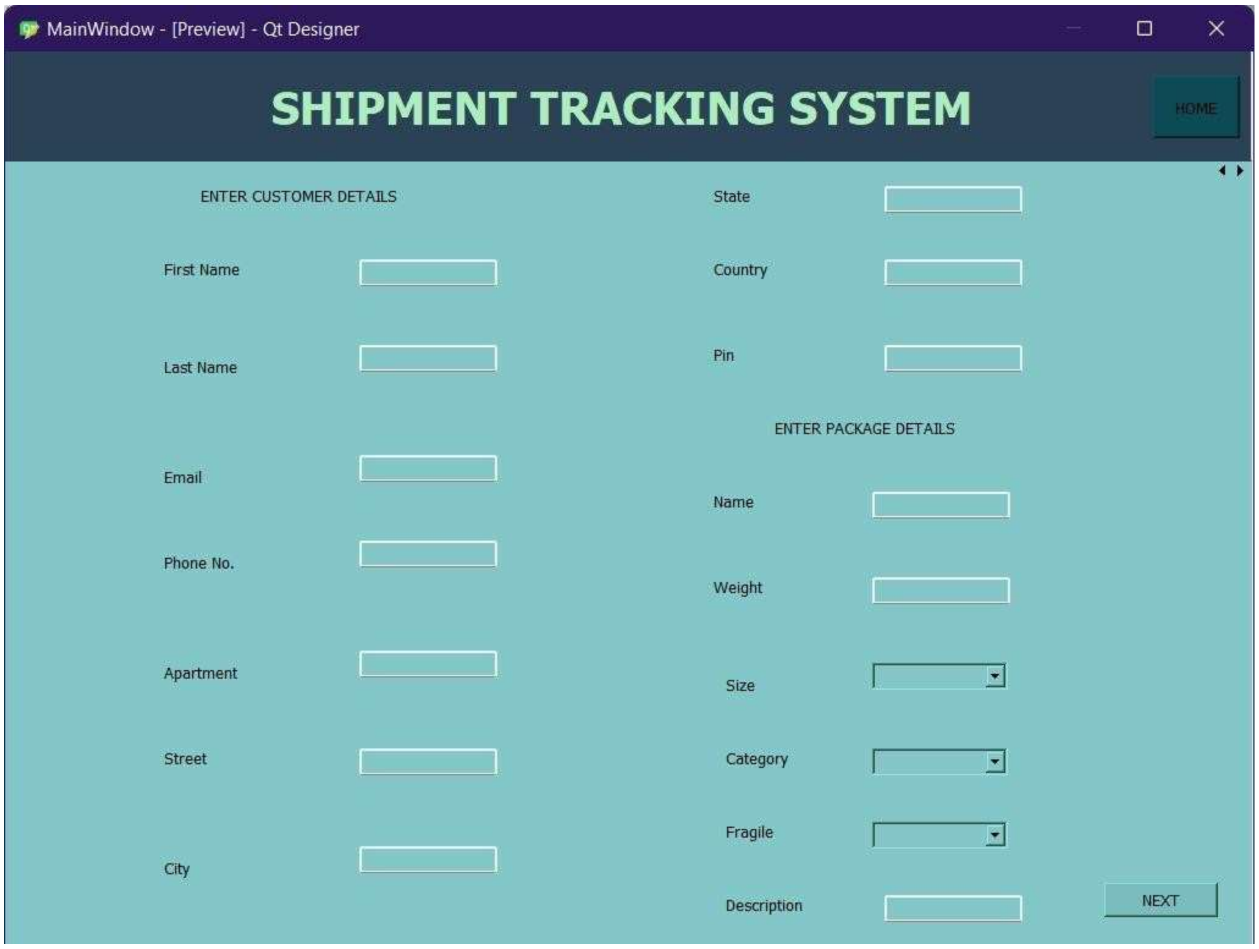
There are two users for the Shipment Tracking System.

- Seller
- Customer

First is the seller, who is responsible for creating shipments,updating their status' and analysing information related to the shipments,etc.



The Seller can ship a package by entering the customer's details and package details.



MainWindow - [Preview] - Qt Designer

SHIPMENT TRACKING SYSTEM

HOME

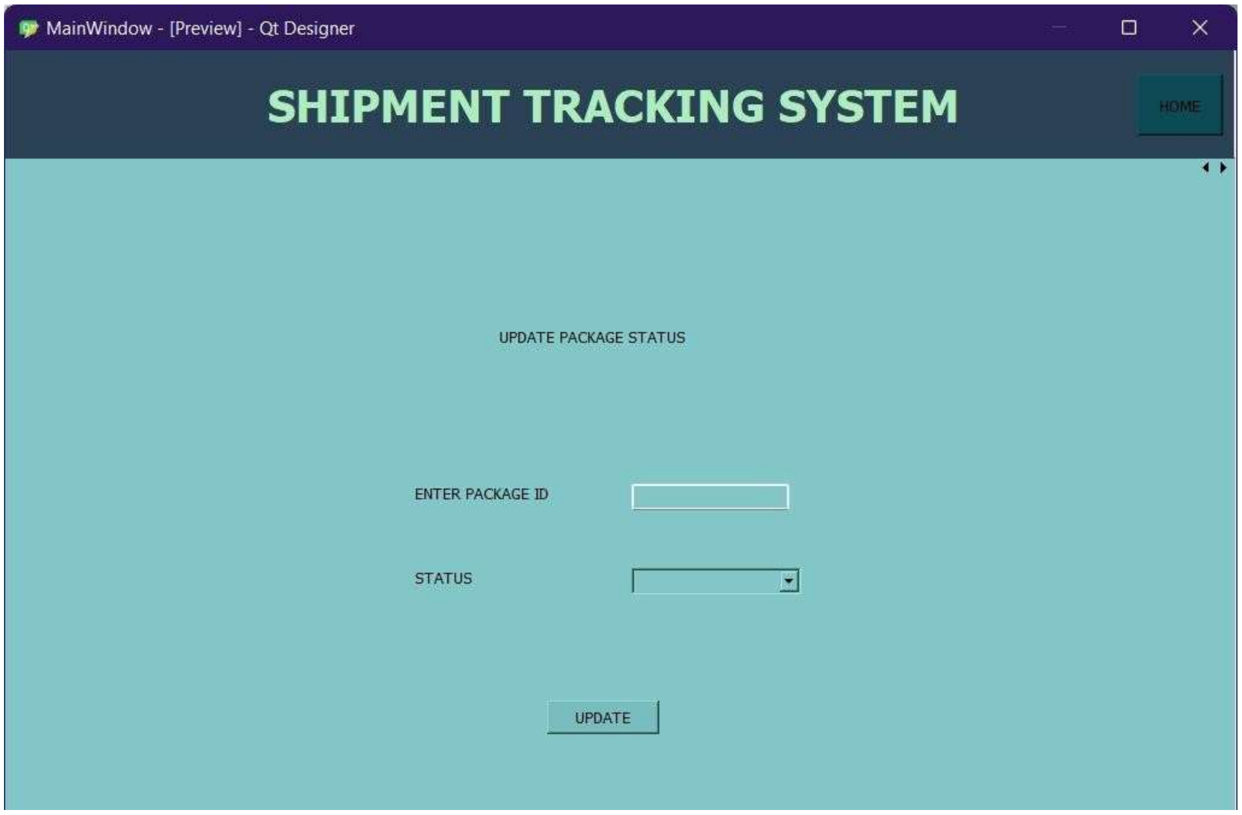
ENTER CUSTOMER DETAILS

First Name	
Last Name	
Email	
Phone No.	
Apartment	
Street	
City	
State	
Country	
Pin	

ENTER PACKAGE DETAILS

Name	
Weight	
Size	
Category	
Fragile	
Description	

NEXT



MainWindow - [Preview] - Qt Designer

SHIPMENT TRACKING SYSTEM

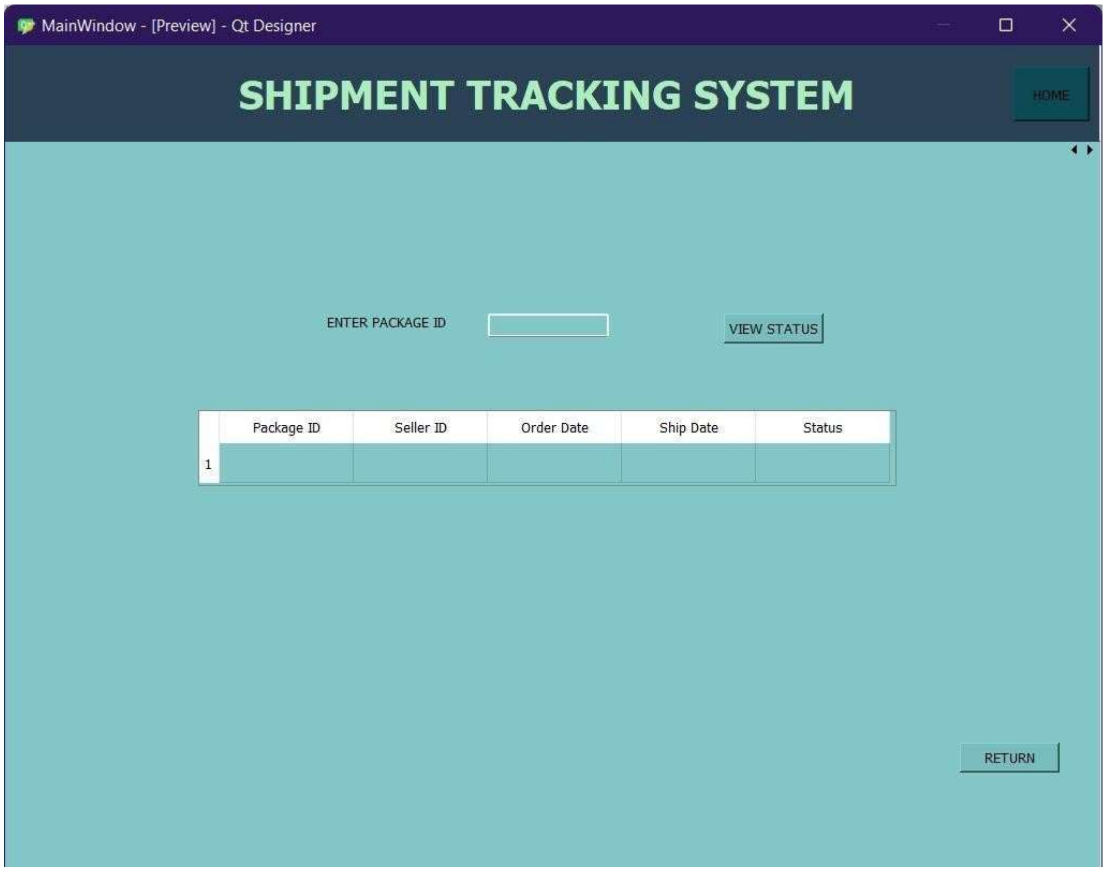
HOME

UPDATE PACKAGE STATUS

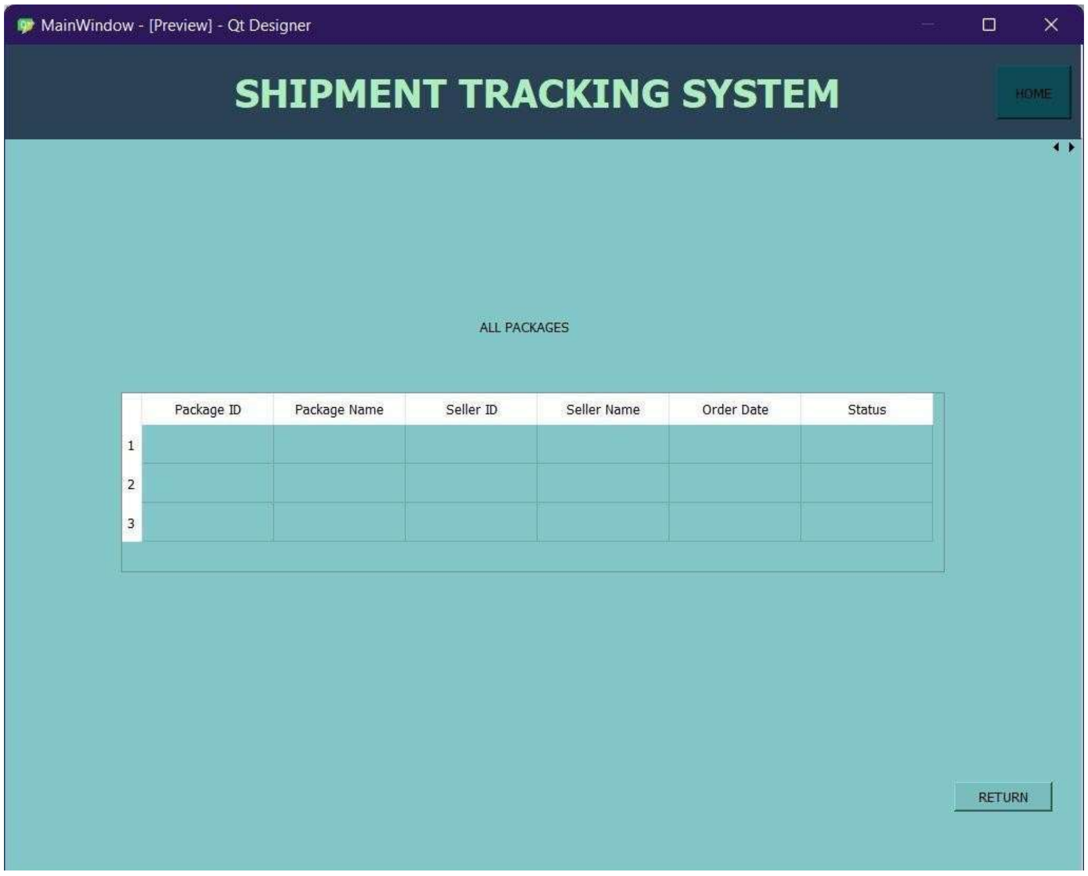
ENTER PACKAGE ID	
STATUS	

UPDATE

Second, is the customer who has the ability to track their package as well as view their previous shipments



	Package ID	Seller ID	Order Date	Ship Date	Status
1					



	Package ID	Package Name	Seller ID	Seller Name	Order Date	Status
1						
2						
3						

CONCLUSION

In this evolutionary age of on-demand delivery and online purchase, there is an immense rise in online purchases which, in turn, results in high demand for an efficient and accurate goods delivery-tracking system. By means of this project, our aim was to build a simple, minimized but thorough and efficient form of one such software.

Working on this project pushed us to think beyond our limits and was an enriching experience. Completing this project filled us with immense satisfaction and thus, we would like to thank our professors for giving us the opportunity to work on this project.