Indian Institute of Information Technology
Vadodara

DBMS Project Report

Course Instructor

Dr Antriksh Goswami

Project Title

Online City Shop

Submitted By

Tushar 201951163

Table of Contents

*These are active links, just click on the title and you will be redirected to the page where topic is present.

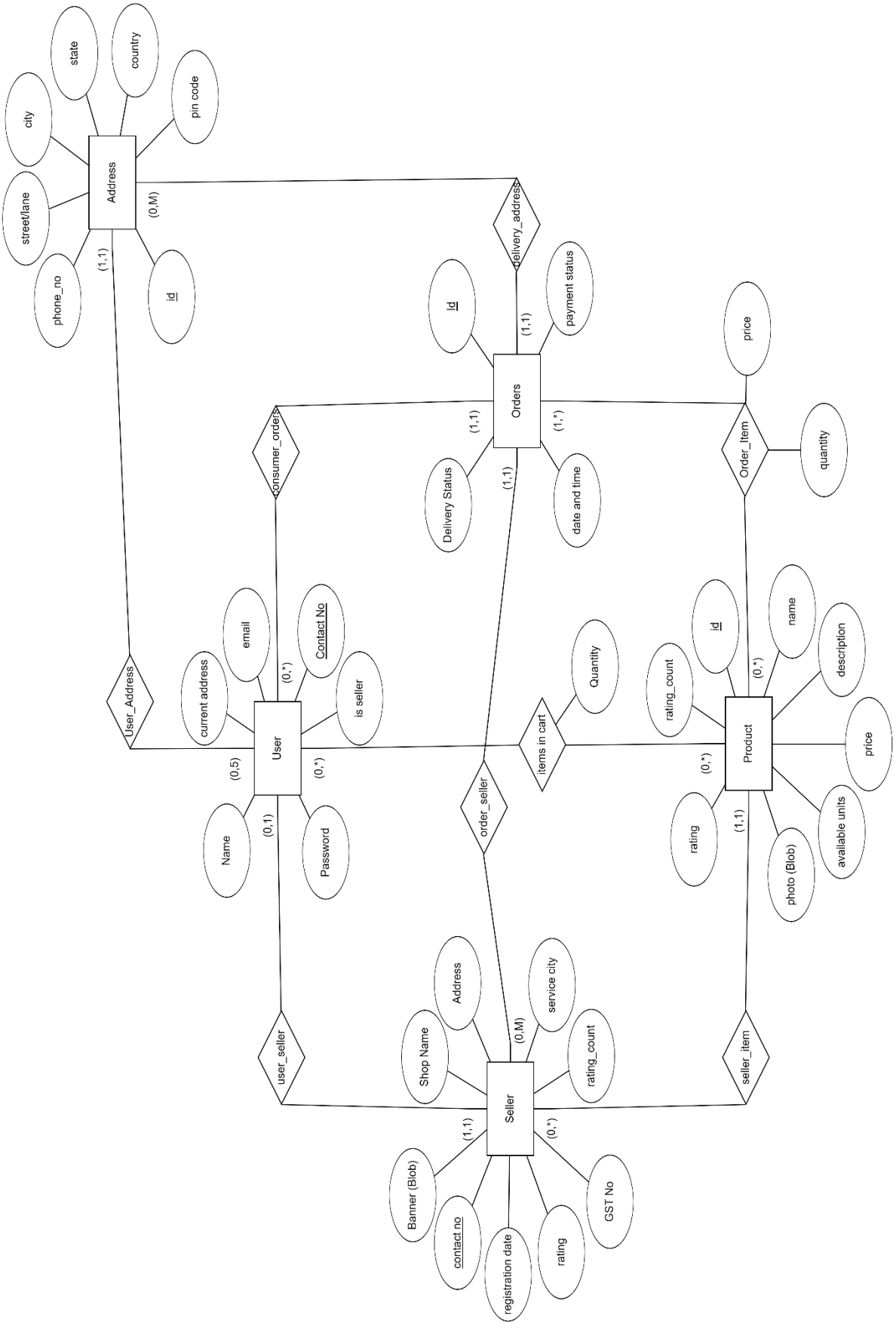
Abstract

Online City Shop is an ecommerce app which implemented in Android. This is app like Swiggy and Zomato and other ecommerce companies, which is working on delivery of products in a city only. In the same way Online City Shop is designed to fulfil requirements of the same city delivery of products. In this app a user can buy a product from a seller which in the same city only. This app has login and signup system, searching of shops by their name and description keywords, user details management system where user can update his information, mechanism of adding and choosing current address, listing of orders placed by the user, management of items in shopping cart, add items to shopping cart, register for seller account and account management, adding products by seller, seller can his view old and new orders, user can place order, in a single order user can buy products from same seller only. All the above-mentioned features are provided in Online City Shop.

Programming Language & Framework

- Development Platform - Android Studio
- Programming Language – Java
- Database – MySql Data base

Entity Relationship Diagram



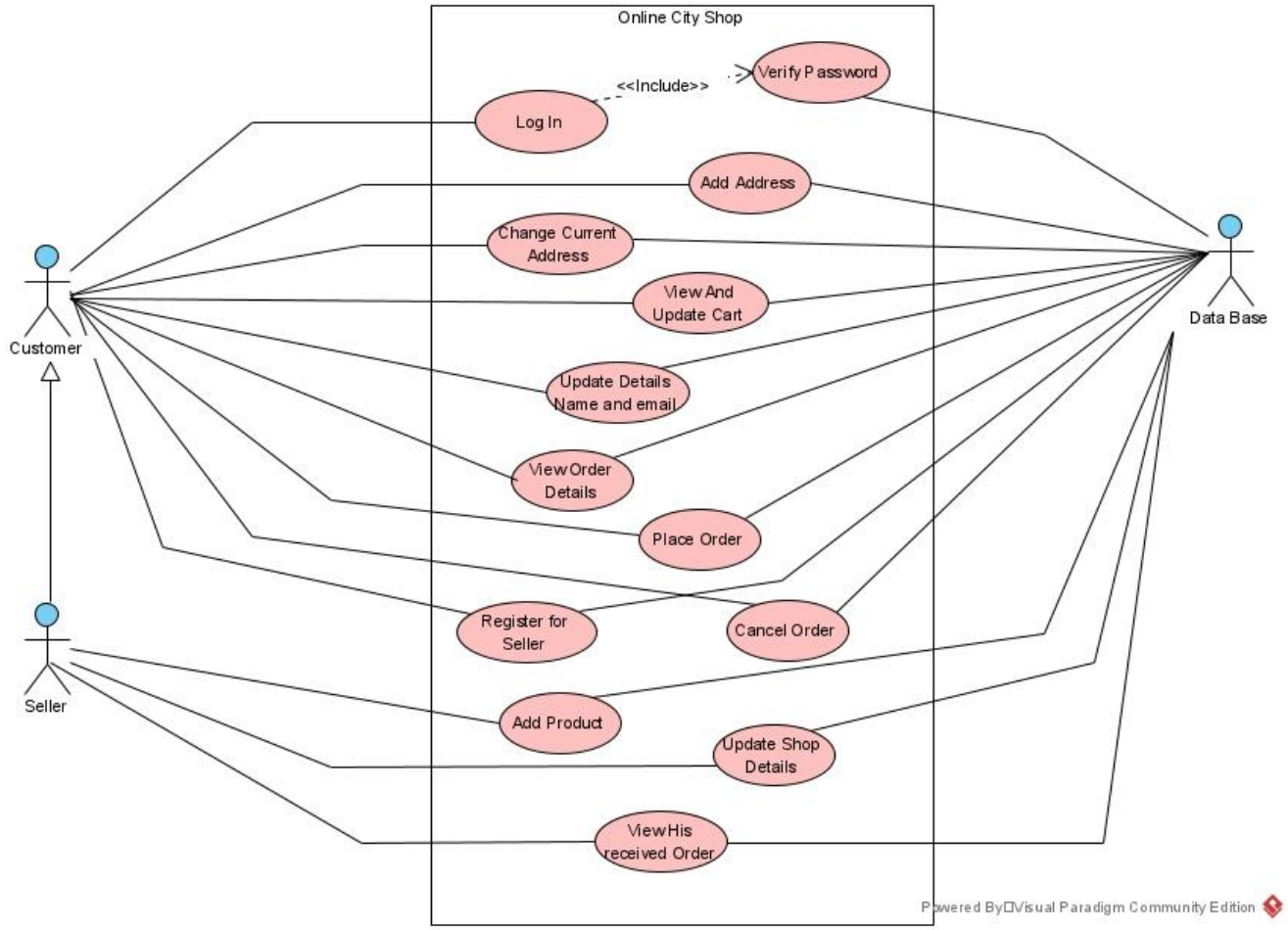
SQL Script

```
create database ocs;
-- Product
create table product(
    seller_id char(10) not null,
    photo mediumblob not null,
    id int auto_increment,
    available_units int not null,
    name varchar(100) not null,
    description varchar(250) not null,
    rating numeric(3,2) default 0,
    rating_count int default 0,
    price numeric(8,2) not null,
    primary key (id),
    foreign key(seller_id) references user(contact_no) on delete cascade,
    fulltext(name),
    fulltext(description)
);

-- Seller
create table seller(
    seller_id char(10) not null,
    contact_no char(10) not null,
    banner mediumblob not null,
    shop_name varchar(50) not null,
    description varchar(250) not null,
    address varchar(250) not null,
    service_city varchar(20) not null,
    rating numeric(3,2) default 0,
    rating_count int default 0,
    registration_date datetime default now(),
    gst_no varchar(20) not null,
    check (contact_no like '6%' or contact_no like '7%' or contact_no like '8%
' or contact_no like '9%'),
    foreign key(seller_id) references user(contact_no) on delete cascade,
    fulltext(shop_name),
    fulltext(description)
);

-- Shopping cart
create table items_in_cart
(
    quantity int not null,
    user_id char(10) not null,
    product_id int not null,
    unique(user_id, product_id),
```

```
    foreign key (user_id) references user(contact_no) on delete cascade,
    foreign key (product_id) references product(id) on delete cascade
);

-- User
create table user (
  name VARCHAR(20) NOT NULL,
  email VARCHAR(45) NOT NULL,
  password VARCHAR(45) NOT NULL,
  contact_no CHAR(10),
  is_seller bit default false,
  curr_address int,
  check (contact_no like '6%' or contact_no like '7%' or contact_no like '8%'
or contact_no like '9%'),
  PRIMARY KEY(contact_no)
);

alter table user add foreign key(curr_address) references address(id);

-- Address
create table address (
    id INT NOT NULL AUTO_INCREMENT,
    pin_code INT NOT NULL,
    country VARCHAR(45) NOT NULL,
    state VARCHAR(45) NOT NULL,
    city VARCHAR(45) NOT NULL,
    street VARCHAR(45) NOT NULL,
    phone_number CHAR(10) NOT NULL,
    user_id CHAR(10) NOT NULL,
    check (phone_number like '6%' or phone_number like '7%' or phone_number li
ke '8%' or phone_number like '9%'),
    PRIMARY KEY (id),
    FOREIGN KEY (user_id) REFERENCES user(contact_no) ON DELETE CASCADE
  );

-- Order
create table orders(
    user_id CHAR(10) NOT NULL,
    id int NOT NULL auto_increment,
    payment_status tinyint NOT NULL,
    address_id INT NOT NULL,
    date_time datetime default now(),
    delivery_status tinyint default 0,
    seller_id char(10) not null,
    PRIMARY KEY (id),
    FOREIGN KEY (seller_id) references seller(seller_id),
    FOREIGN KEY (user_id) REFERENCES user(contact_no) ON DELETE CASCADE,
    FOREIGN KEY (address_id) REFERENCES address(id),
```

```
    CHECK (payment_status>=0 and payment_status<=1 and (delivery_status>=0 and
 delivery_status<=3))
);

-- Order Item
create table order_item (
  price NUMERIC(8,2) NOT NULL,
  quantity INT DEFAULT 1,
  product_id INT NOT NULL,
  order_id INT NOT NULL,
  unique(product_id, order_id),
  FOREIGN KEY (order_id) REFERENCES orders(id) ON DELETE CASCADE
);

-- View
create view order_total
as (
    select SP.order_id, SP.total_price, O.user_id
    from
            (select order_id, sum(total_price) as total_price
            from (
                select order_id, price*quantity as total_price
                from order_item
            ) as temp
            group by order_id) as SP,
            orders as O
    where O.id=SP.order_id
);

create table last_order_id(
    id int,
    data_value int
);

insert into last_order_id values(1, 0);
```

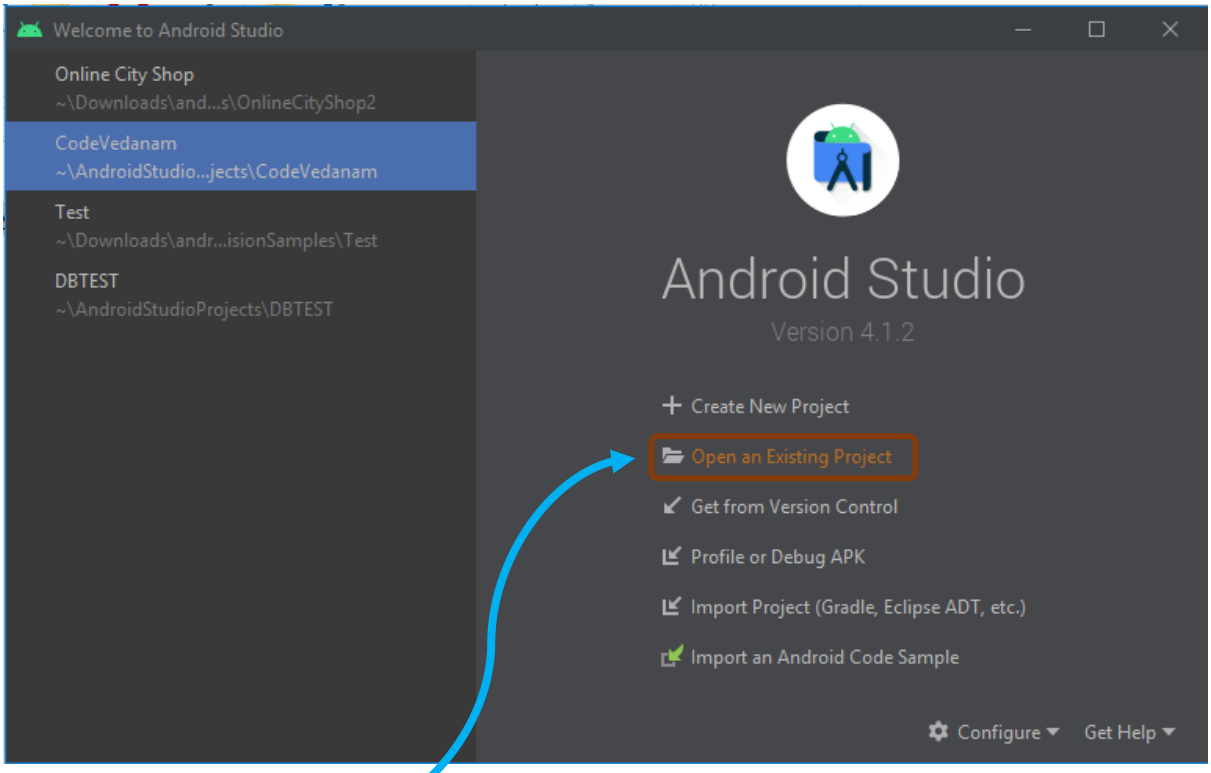
UML – Use Case Diagram



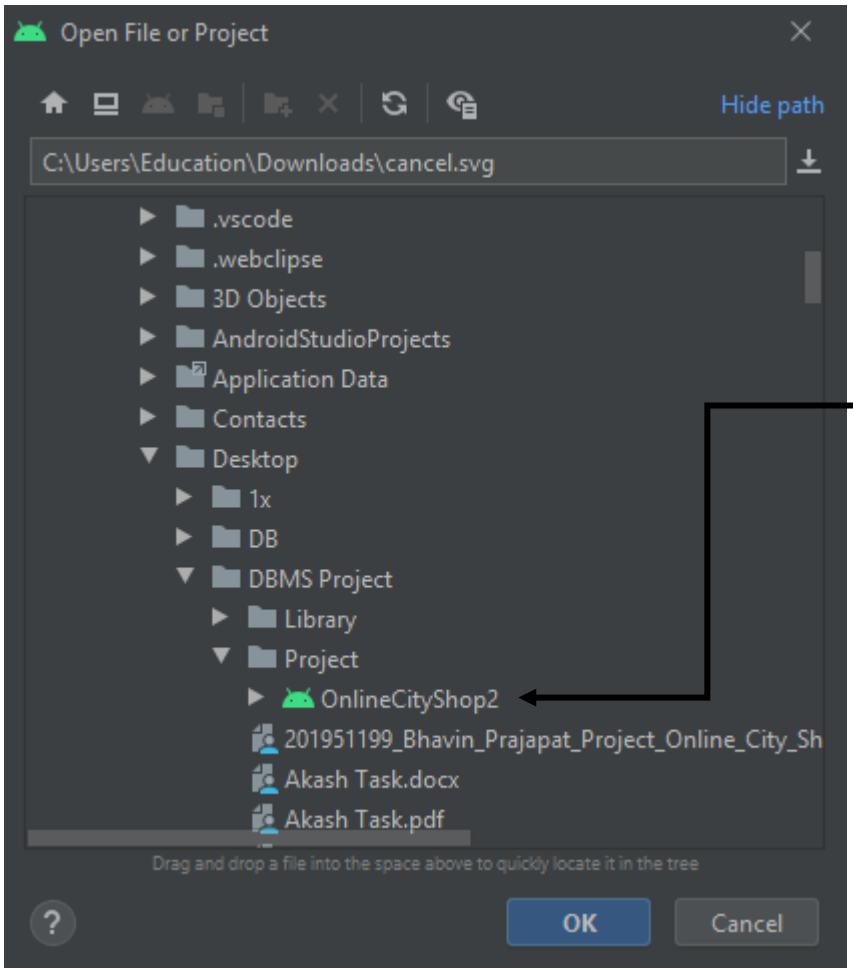
Steps to Run Project

1. Install Android Studio IDE
   https://developer.android.com/studio

2. Open Project in IDE



Click to Open

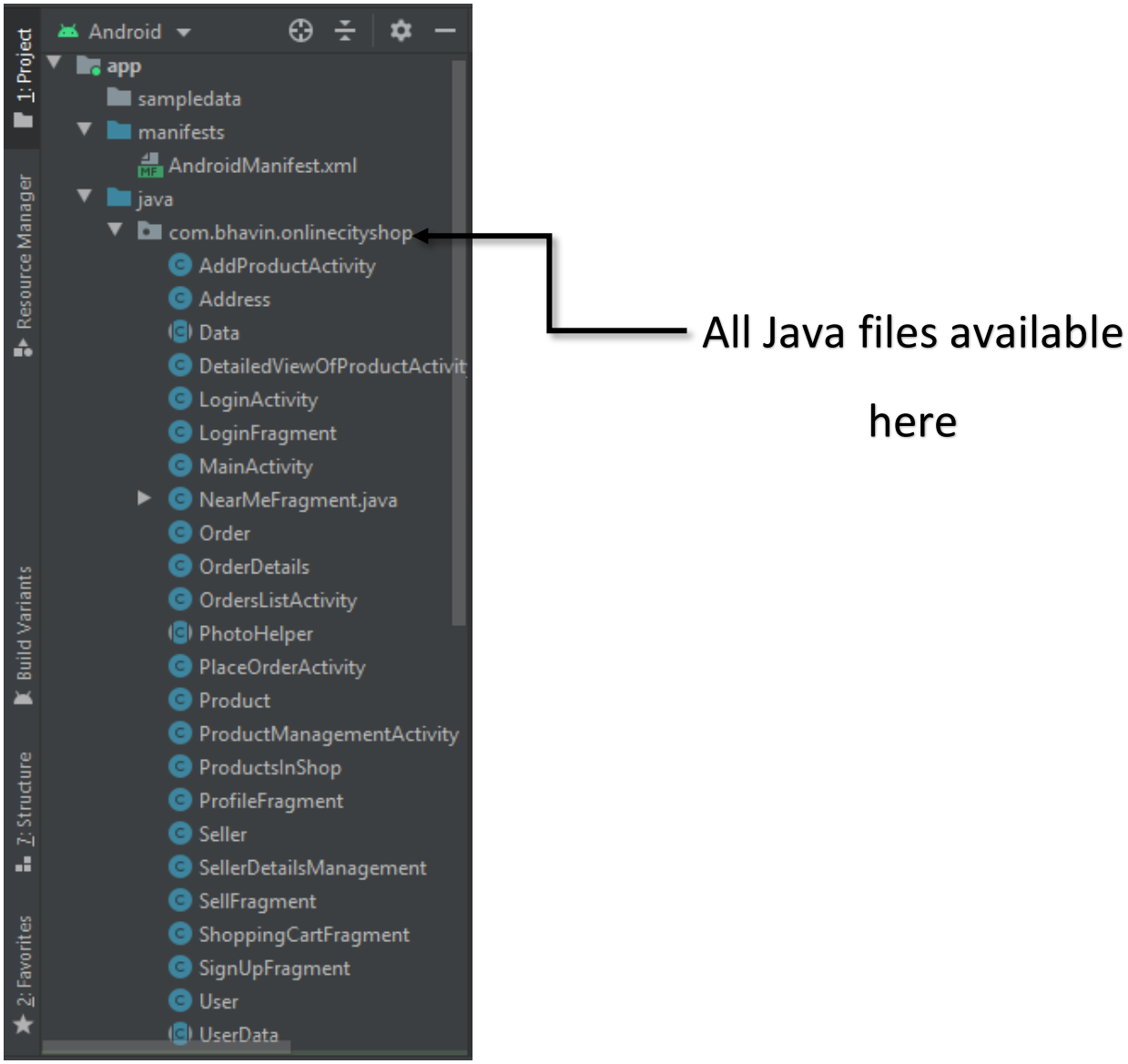


Pick the project

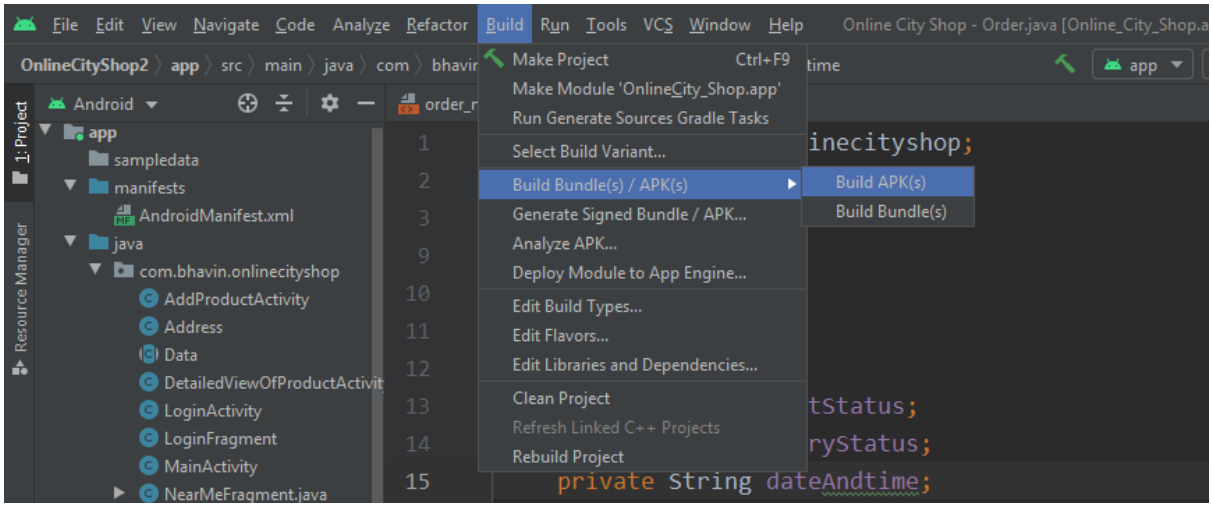
It will show an

Android icon

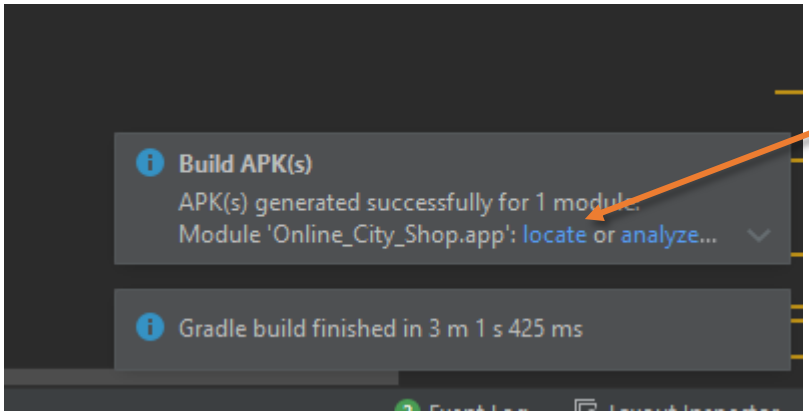
3. After Gradle Build finished



4. Build APK file



5. When build finished



6. Install APK in your device and RUN

There is no need to take care of data base, because it is hosted on a live server at https://heliohost.org . This is a free online data-base server, so it may go offline sometimes and may take too long to connect app with data base. Normally it will connect app in a little amount of time.

Just run the app in your device.

Java Classes

User

```
package com.tushar.onlinecityshop;

import java.util.*;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class User {

    public static final int NO_ADDRESS = -1;

    private String name;
    private String contactNo;
    private String emailId;
    private String password;
    private boolean isSeller;
    private int addressId;

    public User() {
        name = "NA";
        contactNo = "NA";
        emailId = "NA";
        password = "";
        isSeller = false;
        addressId = -1;
    }

    public User(String contactNo, String password, String
emailId, String name, boolean isSeller, int addressId) {
        this.contactNo = contactNo;
        this.emailId = emailId;
        this.password=password;
        this.name=name;
        this.isSeller=isSeller;
        this.addressId=addressId;
    }

    public boolean hasAnyAddress() {
        if(addressId == NO_ADDRESS){
```

```
            return false;
        }else {
            return true;
        }
    }

    public void setContact(String contactNo) {
        this.contactNo=contactNo;
    }

    public String getContact() {
        return contactNo;
    }

    public void setEmailId(String emailId) {
        this.emailId=emailId;
    }

    public String getEmailId() {
        return emailId;
    }

    public void setPassword(String password) {
        this.password=password;
    }

    public String getPassword() {
        return password;
    }

    public void setName(String name) {
        this.name=name;
    }

    public String getName() {
        return name;
    }

    public void setIsSeller(boolean isSeller) {
        this.isSeller=isSeller;
    }

    public boolean getIsSeller() {
        return isSeller;
```

```
    }

    public void setAddressId(int addressId) {
        this.addressId=addressId;
    }

    public int getAddressId() {
        return addressId;
    }

    public static User fetchUser(String userId, Connection
con) throws SQLException {

        User user = new User();

        String query = "select name, email, is_seller,
curr_address from user where contact_no=?";

        try(
                PreparedStatement ps =
con.prepareStatement(query)
        ) {

            ps.setString(1, userId);

            ResultSet rs = ps.executeQuery();
            rs.next();

            user.setName(rs.getString(1));
            user.setEmailId(rs.getString(2));
            user.setContact(userId);
            user.setIsSeller(rs.getBoolean(3));
            user.setAddressId(rs.getInt(4));
            rs.close();

        } catch (SQLException throwables) {
            throw throwables;
        }

        return user;
    }

    public static void addNewUser(User user, Connection con)
throws SQLException {
```

```
        String query = "insert into user(name, email,
password, contact_no) values(?,?,?,?)";

        try(
                PreparedStatement statement =
con.prepareStatement(query))
        {

            String contact = user.getContact();

            if((contact.length() != 10) ||
(contact.charAt(0) <= '5')){
                throw new SQLException("Incorrect Contact");
            }

            statement.setString(4,user.getContact());
            statement.setString(3, user.getPassword());
            statement.setString(2,user.getEmailId());
            statement.setString(1, user.getName());

            statement.executeUpdate();

        } catch (SQLException throwables) {
            if(throwables.getErrorCode() == 1062){
                /**
                 * This error code is for duplicate entry of
contact no
                 */
                SQLException exception = new
SQLException("Phone no already exist");
                throw exception;
            }

            else if(throwables.getErrorCode() == 1406){
                /**
                 * This error code is when input parameters
are not in the range.
                 */
                SQLException exception = new
SQLException("Contact number not valid !!");
                throw exception;
            }
            throw throwables;
```

```
        }

    }

    public static void deleteUser(String contactNo,
Connection con) throws SQLException {

        String query = "delete from user where
contact_no=?";

        try(
                PreparedStatement statement =
con.prepareStatement(query))
        {

            statement.setString(1,contactNo);
            statement.executeUpdate();

        } catch (SQLException throwables) {
            throw throwables;
        }

    }

    public static void updateUser(User user, Connection con)
throws SQLException {

        String query = "update user set name=?, email=? " +
                "where contact_no=?";

        try(
                PreparedStatement statement =
con.prepareStatement(query))
        {

            statement.setString(1,user.getName());
            statement.setString(2,user.getEmailId());
            statement.setString(3, user.getContact());

            statement.executeUpdate();

        } catch (SQLException throwables) {
            throw throwables;
        }
```

```
    }

    public static boolean checkUser(User user, Connection
con) throws SQLException {

        boolean success = false;
        String query = "select password from user where
contact_no=?";

        try(
                PreparedStatement ps =
con.prepareStatement(query)
        )
        {
            String contact = user.getContact();

            if((contact.length() != 10) ||
(contact.charAt(0) <= '5')){
                throw new SQLException("Incorrect Contact");
            }

            ps.setString(1, user.getContact());

            ResultSet rs = ps.executeQuery();

            String password = null;

            while(rs.next()) {
                password = rs.getString(1);
            }

            if(user.getPassword().equals(password)){
                success = true;
            }else{
                if(password == null){
                    throw new SQLException("No Account with
this contact.\nPlease register");
                }
                throw new SQLException("Incorrect
Password");
            }

        } catch (SQLException throwables) {
```

```
            throw throwables;
        }

        return success;
    }

    public static void changeCurrentAddress(String userId,
int addressId, Connection con) throws SQLException {

        String query="update user "+
                "set curr_address=? "+
                "where contact_no=?";
        try(
                PreparedStatement statement =
con.prepareStatement(query);
        ){

            statement.setInt(1, addressId);
            statement.setString(2, userId);
            statement.executeUpdate();

        } catch (SQLException throwables) {
            throw throwables;
        }
    }

    public static ArrayList<Product> fetchItemsInCart(String
contactNo, Connection con) throws SQLException {

        String query =
                "select p.name, p.description, p.price,
T.quantity, p.seller_id, p.photo, p.id " +
                "from (select product_id, quantity " +
                "      from items_in_cart " +
                "      where user_id=?) as T, product as p "
+
                "where p.id=T.product_id ";

        ArrayList<Product> arrList=new ArrayList<>();

        try(
                PreparedStatement statement =
con.prepareStatement(query)
        ){
```

```
            statement.setString(1, contactNo);

            ResultSet rs = statement.executeQuery();

            while (rs.next()) {
                Product product = new Product();

                product.setName(rs.getString(1));
                product.setDescription(rs.getString(2));
                product.setPrice(rs.getFloat(3));
                product.setQuantity(rs.getInt(4));
                product.setSellerId(rs.getString(5));

product.setPhoto(PhotoHelper.convertBlobToBitmap(rs.getBlob(
6)));
                product.setId(rs.getInt(7));

                arrList.add(product);
            }

        } catch (SQLException throwables) {
            throw throwables;
        }

        return arrList;
    }

    public static void addToShoppingCart(String userId, int
productId, Connection con) throws SQLException {

        String query="insert into items_in_cart(quantity,
user_id, product_id) " +
                "values(?,?,?)";

        try (
                PreparedStatement
statement=con.prepareStatement(query)
        ){

            statement.setInt(1, 1);
            statement.setString(2, userId);
            statement.setInt(3, productId);

            statement.executeUpdate();
```

```
        }catch(SQLException throwables) {
            throw throwables;
        }

    }

    public static void removeItemFromCart(String userId, int
productId, Connection con) throws SQLException {


        String query="delete from items_in_cart " +
                "where user_id=? and product_id=?";
        try(
                PreparedStatement
statement=con.prepareStatement(query)
        ){

            statement.setString(1, userId);
            statement.setInt(2, productId);

            statement.executeUpdate();

        }catch(SQLException throwables) {
            throw throwables;
        }

    }

    public static void updateItemQuantityInCart(String
userId, int productId, int quantity, Connection con) throws
SQLException {

        String query="update items_in_cart set quantity=? "
+
                "where user_id=? and product_id=?";

        try (
                PreparedStatement
statement=con.prepareStatement(query)
        ){
            System.out.println("quantity : "+quantity+" user
id "+userId+" pID "+productId);
            statement.setInt(1, quantity);
```

```
            statement.setString(2, userId);
            statement.setInt(3, productId);

            statement.executeUpdate();

        }catch(SQLException throwables) {
            throw throwables;
        }

    }

}
```

Address

```
package com.tushar.onlinecityshop;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

public class Address {

    private int id;
    private String phoneNo;
    private String street;
    private String city;
    private String state;
    private String country;
    private int pinCode;

    public Address() {

    }

    public Address(int id, String phoneNo, String street,
String city, String state, String country, int pinCode) {
        this.id = id;
        this.phoneNo = phoneNo;
        this.street = street;
        this.city = city;
        this.state = state;
```

```
        this.country = country;
        this.pinCode = pinCode;
    }

    public int getId() {
        return id;
    }

    public String getAddress() {
        return getStreet()+", "+getCity()+", "+getState()+",
"+getCountry();
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getPhoneNo() {
        return phoneNo;
    }

    public void setPhoneNo(String phoneNo) {
        this.phoneNo = phoneNo;
    }

    public String getStreet() {
        return street;
    }

    public void setStreet(String street) {
        this.street = street;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

    public String getState() {
        return state;
    }
```

```
    public void setState(String state) {
        this.state = state;
    }

    public String getCountry() {
        return country;
    }

    public void setCountry(String country) {
        this.country = country;
    }

    public int getPinCode() {
        return pinCode;
    }

    public void setPinCode(int pinCode) {
        this.pinCode = pinCode;
    }

    public static Address fetchAddressHavingID(int
id,Connection con) throws SQLException {

        String query="select id, pin_code, country, state,
city, street, phone_number " +
                     "from address where id=?";
        Address a= new Address();

        try(
                PreparedStatement preparedStatement =
con.prepareStatement(query)
        ){
            preparedStatement.setInt(1,id);
            ResultSet rs = preparedStatement.executeQuery();

            while (rs.next()){
                a.setId(rs.getInt(1));
                a.setPinCode(rs.getInt(2));
                a.setCountry(rs.getString(3));
                a.setState(rs.getString(4));
                a.setCity(rs.getString(5));
                a.setStreet(rs.getString(6));
                a.setPhoneNo(rs.getString(7));
```

```
            }

        }
        catch (SQLException throwables) {
            throw throwables;
        }

        return a;
    }

    public static boolean deleteAddressHavingID(int
id,Connection con) throws SQLException {

        String query="delete from address where id=?";
        int i=0;

        try(
                PreparedStatement preparedStatement =
con.prepareStatement(query)
        ){
            preparedStatement.setInt(1,id);
            i=preparedStatement.executeUpdate();
        }
        catch (SQLException throwables) {
            throw throwables;
        }

        if(i>0){
            return true;
        }
        else{
            return false;
        }
    }

    public static boolean updateAddressHavingID(Address
a,Connection con) throws SQLException {

        String query="update address set  street=?,
phone_number=?, city=?, state=?, country=?, pin_code=? where
id=?";
        int i=0;
        try(
                PreparedStatement preparedStatement =
```

```
con.prepareStatement(query)
        ){

            preparedStatement.setString(1,a.getStreet());
            preparedStatement.setString(2,a.getPhoneNo());
            preparedStatement.setString(3,a.getCity());
            preparedStatement.setString(4,a.getState());
            preparedStatement.setString(5,a.getCountry());
            preparedStatement.setInt(6,a.getPinCode());
            preparedStatement.setInt(7, a.getId());

            i=preparedStatement.executeUpdate();
        }
        catch (SQLException throwables) {
            throw throwables;
        }

        if(i>0){
            return true;
        }
        else{
            return false;
        }
    }

    public static boolean addNewAddress(Address f, String
userId, Connection con) throws SQLException{

        String query="insert into address(pin_code, country,
state, city, street, phone_number, user_id) " +
                "values(?, ?, ?, ?, ?, ?, ?)";
        int i=0;

        try(
                PreparedStatement preparedStatement =
con.prepareStatement(query)
        ){
            preparedStatement.setString(6, f.getPhoneNo());
            preparedStatement.setString(5,f.getStreet());
            preparedStatement.setString(4,f.getCity());
            preparedStatement.setString(3,f.getState());
            preparedStatement.setString(2,f.getCountry());
            preparedStatement.setInt(1,f.getPinCode());
            preparedStatement.setString(7,userId);
```

```
            i= preparedStatement.executeUpdate();
        }
        catch (SQLException throwables) {
            throw throwables;
        }

        if(i>0){
            return true;
        }
        else{
            return false;
        }

    }

    public static ArrayList<Address>
fetchAddressesOfUser(String userId, Connection con) throws
SQLException {

        String query="select id, pin_code, country, state,
city, street, phone_number " +
                "from address where user_id=?";
        ArrayList<Address> getdata = new ArrayList<>();

        try(
                PreparedStatement preparedStatement =
con.prepareStatement(query)
        )
        {

            preparedStatement.setString(1,userId);
            ResultSet rs = preparedStatement.executeQuery();

            while (rs.next()) {
                Address a= new Address();
                a.setId(rs.getInt(1));
                a.setPinCode(rs.getInt(2));
                a.setCountry(rs.getString(3));
                a.setState(rs.getString(4));
                a.setCity(rs.getString(5));
                a.setStreet(rs.getString(6));
                a.setPhoneNo(rs.getString(7));
                getdata.add(a);
                System.out.println(a.getId());
```

```
            }

        }
        catch (SQLException throwables) {
            throw throwables;
        }

        return getdata;
    }

}
```

Order

```
package com.tushar.onlinecityshop;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

public class Order {

    private int id;
    private int paymentStatus;
    private int deliveryStatus;
    private String dateAndtime;
    private float totalAmount;
    private int itemCount;
    private int deliveryAddressId;
    private String SellerId;
    private String SellerName;
    private String ConsumerName;

    public static final int NOT_PAID = 0;
    public static final int PAID = 1;
    public static final int ORDER_BOOKED = 0;
    public static final int ON_THE_WAY = 1;
    public static final int DELIVERED = 2;
    public static final int CANCELLED = 3;

    public String getDecodedDeliveryStatus(){
```

```
        switch (deliveryStatus){
            case ORDER_BOOKED:
                return "Order Booked";
            case ON_THE_WAY:
                return "On The Way";
            case DELIVERED:
                return "Delivered";
            case CANCELLED:
                return "Cancelled";
            default:
                return "Invalid Data";
        }
    }

    public String getDecodedPaymentStatus(){
        switch (paymentStatus){
            case PAID:
                return "Paid";
            case NOT_PAID:
                return "Not Paid";
            default:
                return "Invalid Data";
        }
    }

    public String getConsumerName() {
        return ConsumerName;
    }

    public void setConsumerName(String consumerName) {
        ConsumerName = consumerName;
    }

    public String getSellerName() {
        return SellerName;
    }

    public void setSellerName(String sellerName) {
        SellerName = sellerName;
    }

    public String getSellerId() {
        return SellerId;
    }
```

```
public void setSellerId(String sellerId) {
    SellerId = sellerId;
}

public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public int getPaymentStatus() {
    return paymentStatus;
}

public void setPaymentStatus(int paymentStatus) {
    this.paymentStatus = paymentStatus;
}

public int getDeliveryStatus() {
    return deliveryStatus;
}

public void setDeliveryStatus(int deliveryStatus) {
    if(deliveryStatus>=0 && deliveryStatus<=3)
        this.deliveryStatus = deliveryStatus;
    else
        System.out.println("Invalid Address Status");
}

public String getDateAndtime() {
    return dateAndtime;
}

public void setDateAndtime(String dateAndtime) {
    this.dateAndtime = dateAndtime;
}

public float getTotalAmount() {
    return totalAmount;
}
```

```
    public void setTotalAmount(float totalAmount) {
        this.totalAmount = totalAmount;
    }

    public int getItemCount() {
        return itemCount;
    }

    public void setItemCount(int itemCount) {
        this.itemCount = itemCount;
    }

    public int getDeliveryAddressId() {
        return deliveryAddressId;
    }

    public void setDeliveryAddressId(int deliveryAddressId)
{
        this.deliveryAddressId = deliveryAddressId;
    }

    public static ArrayList<Order>
fetchOrdersForUserID(String userId, Connection con) throws
SQLException {

        String query="select O.date_time, O.delivery_status,
S.shop_name, R.total_price, O.id " +
                "from orders as O, order_total as R, seller
as S " +
                "where R.user_id=? and O.user_id=? and
S.seller_id=O.seller_id and R.order_id=O.id";
        ArrayList<Order> getdata = new ArrayList<>();

        try(
                PreparedStatement preparedStatement =
con.prepareStatement(query)
        ) {

            preparedStatement.setString(1,userId);
            preparedStatement.setString(2,userId);
            ResultSet rs = preparedStatement.executeQuery();
            while (rs.next()) {
                Order a= new Order();
                a.setDateAndtime(rs.getString(1));
```

```
                a.setSellerName(rs.getString(3));
                a.setTotalAmount(rs.getInt(4));
                a.setDeliveryStatus(rs.getInt(2));
                a.setId(rs.getInt(5));
                getdata.add(a);
            }
        }
        catch (SQLException throwables) {
            throw throwables;
        }
        return getdata;
    }

    public static ArrayList<Order>
fetchNewOrderReceivedToSeller(String sellerId,

Connection con) throws SQLException {

        String query = "select O.date_time,
O.delivery_status, U.name, R.total_price, O.id " +
                "from orders as O, order_total as R, user as
U " +
                "where R.user_id=O.user_id and
U.contact_no=O.user_id and R.order_id=O.id and
O.delivery_status=? " +
                "and O.seller_id=?";

        ArrayList<Order> getdata = new ArrayList<>();
        try(
                PreparedStatement preparedStatement =
con.prepareStatement(query))
        {
            preparedStatement.setInt(1, ORDER_BOOKED);
            preparedStatement.setString(2, sellerId);
            ResultSet rs = preparedStatement.executeQuery();

            while (rs.next()) {
                Order a= new Order();
                a.setDateAndtime(rs.getString(1));
                a.setConsumerName(rs.getString(3));
                a.setTotalAmount(rs.getInt(4));
                a.setDeliveryStatus(rs.getInt(2));
                a.setId(rs.getInt(5));
                getdata.add(a);
```

```
            }
        }
        catch (SQLException throwables) {
            throw throwables;
        }
        return getdata;
    }

    public static ArrayList<Order>
fetchOldOrderOfSeller(String sellerId, Connection con)
throws SQLException {

        String query = "select O.date_time,
O.delivery_status, U.name, R.total_price, O.id " +
                        "from orders as O, order_total as R,
user as U " +
                        "where R.user_id=O.user_id and
U.contact_no=O.user_id " +
                                "and R.order_id=O.id and
O.delivery_status!=? " +
                                "and O.seller_id=?";

        ArrayList<Order> getdata = new ArrayList<>();
        try(
                PreparedStatement preparedStatement =
con.prepareStatement(query))
        {
            preparedStatement.setInt(1, ORDER_BOOKED);
            preparedStatement.setString(2,sellerId);
            ResultSet rs = preparedStatement.executeQuery();
            while (rs.next()) {
                Order a= new Order();
                a.setDateAndtime(rs.getString(1));
                a.setConsumerName(rs.getString(3));
                a.setTotalAmount(rs.getInt(4));
                a.setDeliveryStatus(rs.getInt(2));
                a.setId(rs.getInt(5));
                getdata.add(a);
            }
            con.close();
        }
        catch (SQLException throwables) {
            throw throwables;
        }
```

```
        return getdata;
    }

    public static void addNewOrder(String user_id, Order
order,
                                   ArrayList<Product>
products, Connection con) throws SQLException {

        int orderId=-1;
        String oid= "select data_value from last_order_id
where id=1";
        String query1 = "insert into orders(id, user_id,
payment_status, address_id, seller_id) " +
                "values(?, ?, ?, ?, ?)";
        String query2 = "insert into order_item(price,
quantity, product_id,order_id) " +
                "values(?,?,?,?)";
        String updateOid ="UPDATE last_order_id SET
data_value = data_value + 1 where id=1";
        try(
                PreparedStatement preparedStatement =
con.prepareStatement(query1);
                PreparedStatement preparedStatement1 =
con.prepareStatement(query2);
                PreparedStatement preparedStatement2 =
con.prepareStatement(oid);
                Statement statement = con.createStatement()
        )
        {
            /**
             * Fetching previos orderId generated and
generate new order id
             */
            ResultSet rs =
preparedStatement2.executeQuery();
            rs.next();
            orderId= rs.getInt(1)+1;
            rs.close();
            /**
             * Insert order to data base
             */
            preparedStatement.setInt(1, orderId);
            preparedStatement.setString(2,user_id);
```

```
preparedStatement.setInt(3,order.getPaymentStatus());

preparedStatement.setInt(4,order.getDeliveryAddressId());

preparedStatement.setString(5,products.get(0).getSellerId())
;
            preparedStatement.executeUpdate();

            /**
             * Update previous data to +1 in data_val
             */
            statement.executeUpdate(updateOid);

            for(int i=0;i<products.size();i++)
            {

preparedStatement1.setFloat(1,products.get(i).getPrice());

preparedStatement1.setInt(2,products.get(i).getQuantity());

preparedStatement1.setInt(3,products.get(i).getId());
                preparedStatement1.setInt(4,orderId);
                preparedStatement1.executeUpdate();
            }
        }
        catch (SQLException throwables)
        {
            throw throwables;
        }
    }

    public static Order fetchOrderDetails(int orderId,
Connection con) throws SQLException {

        String Query = "SELECT u.name, o.payment_status,
o.date_time, o.delivery_status, o.seller_id " +
                       "FROM orders o,user u " +
                       "where o.user_id=u.contact_no and
o.id=?";
        Order a = null;
        try(
                PreparedStatement preparedStatement =
con.prepareStatement(Query)
        ){
```

```
            preparedStatement.setInt(1,orderId);
            ResultSet rs = preparedStatement.executeQuery();
            while(rs.next()){
                a = new Order();
                a.setConsumerName(rs.getString(1));
                a.setId(orderId);
                a.setPaymentStatus(rs.getInt(2));
                a.setDateAndtime(rs.getString(3));
                a.setDeliveryStatus(rs.getInt(4));
                a.setSellerId(rs.getString(5));
            }
        }
        catch (SQLException throwables) {
            throw throwables;
        }
        return a;
    }

    public static Seller fetchSellerOfOrder(int orderId,
Connection con) throws SQLException {

        String query = "select S.shop_name, S.address,
S.banner, S.seller_id, S.rating, S.rating_count " +
                    "from seller as S, orders as O " +
                    "where S.seller_id=O.seller_id and
O.id=?";
        Seller a = null;

        try (
                PreparedStatement preparedStatement =
con.prepareStatement(query)
        ){
            preparedStatement.setInt(1,orderId);
            System.out.println(orderId);
            ResultSet rs = preparedStatement.executeQuery();

            while (rs.next()) {
                a= new Seller();

                a.setShopName(rs.getString(1));
                a.setAddress(rs.getString(2));

a.setBanner(PhotoHelper.convertBlobToBitmap(rs.getBlob(3)));
                a.setSellerId(rs.getString(4));
```

```
                a.setRating(rs.getFloat(5));
                a.setRatingCount(rs.getInt(6));
            }
        }
        catch (SQLException throwables) {
            throw throwables;
        }
        return a;
    }

    public static Address fetchDeliveryAddress(int
OrderId,Connection con) throws SQLException {

        String query="select a.street,a.city,a.phone_number
" +
                    "from address a,orders o " +
                    "where o.address_id=a.id and o.id=?";
        Address a= null;
        try(PreparedStatement preparedStatement =
con.prepareStatement(query))
        {
            preparedStatement.setInt(1,OrderId);
            ResultSet rs = preparedStatement.executeQuery();
            while(rs.next()){
                a = new Address();
                a.setStreet(rs.getString(1));
                a.setCity(rs.getString(2));
                a.setPhoneNo(rs.getString(3));
            }
        }
        catch (SQLException throwables) {
            throw throwables;
        }
        return a;
    }

    public static ArrayList<Product> fetchItemsInOrder(int
orderId, Connection con) throws SQLException {

        String query="select
p.name,oi.price,oi.quantity,oi.product_id,p.photo, p.rating,
p.rating_count " +
                "from product p, order_item oi " +
                "where p.id=oi.product_id and
```

```
oi.order_id=?";
        ArrayList<Product> getdata = new ArrayList<>();

        try(
                PreparedStatement preparedStatement =
con.prepareStatement(query)
        ){
            preparedStatement.setInt(1,orderId);
            ResultSet rs = preparedStatement.executeQuery();
            while (rs.next()) {
                Product a= new Product();

                a.setName(rs.getString(1));
                a.setPrice(rs.getFloat(2));
                a.setQuantity(rs.getInt(3));
                a.setId(rs.getInt(4));

a.setPhoto(PhotoHelper.convertBlobToBitmap(rs.getBlob(5)));
                a.setRating(rs.getFloat(6));
                a.setRatingCount(rs.getInt(7));

                getdata.add(a);
            }
        }
        catch (SQLException throwables) {
            throw throwables;
        }
        return getdata;
    }

    public static void changeDeliveryStatus(int orderId,int
deliveryStatus, Connection con) throws SQLException {

        String query = "UPDATE orders SET delivery_status =
? WHERE orders.id = ?";

        try (
                PreparedStatement preparedStatement =
con.prepareStatement(query)
        ){
            preparedStatement.setInt(2,orderId);
            preparedStatement.setInt(1, deliveryStatus);

            preparedStatement.executeUpdate();
```

```
        }
        catch (SQLException throwables) {
            throw throwables;
        }
    }

    public static void changePaymentStatus(int orderId,int
paymentStatus, Connection con) throws SQLException {

        String query = "UPDATE orders SET payment_status = ?
WHERE orders.id = ?";

        try (
                PreparedStatement preparedStatement =
con.prepareStatement(query)
        ){
            preparedStatement.setInt(2,orderId);
            preparedStatement.setInt(1, paymentStatus);

            preparedStatement.executeUpdate();
        }
        catch (SQLException throwables) {
            throw throwables;
        }
    }
}
```

Product

```
package com.tushar.onlinecityshop;

import android.graphics.Bitmap;

import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;

public class Product {

    private int id;
    private float price;
```

```
    private String name;
    private String description;
    private int availableUnits;
    private float rating;
    private int ratingCount;
    private Bitmap photo;

    private int quantity;
    private String sellerId;

    public Product(){
        this.id = 0;
        this.price = 0;
        this.name = "";
        this.description = "";
        this.availableUnits = 0;
        this.rating=0;
        this.ratingCount=0;
    }

    public Product(int id, float price, String name, String
description,
                   int availableUnits, float rating, int
ratingCount, int quantity) {
        this.id = id;
        this.price = price;
        this.name = name;
        this.description = description;
        this.availableUnits = availableUnits;
        this.rating = rating;
        this.ratingCount = ratingCount;
        this.quantity = quantity;
    }

    public Product(Product product){
        this.id = product.getId();
        this.price = product.getPrice();
        this.name = product.getName();
        this.description = product.getDescription();
        this.availableUnits = product.getAvailableUnits();
        this.rating = product.getRating();
        this.quantity = product.getQuantity();
        this.ratingCount = product.getRatingCount();
        this.photo = product.getPhoto();
```

```
        this.sellerId = product.getSellerId();
    }

    public String getSellerId() {
        return sellerId;
    }

    public void setSellerId(String sellerId) {
        this.sellerId = sellerId;
    }

    public Bitmap getPhoto() {
        return photo;
    }

    public void setPhoto(Bitmap photo) {
        this.photo = photo;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public int getRatingCount() {
        return ratingCount;
    }

    public void setRatingCount(int ratingCount) {
        this.ratingCount = ratingCount;
    }

    public float getRating() {
        return rating;
    }

    public void setRating(float rating) {
        this.rating = rating;
    }

    public float getPrice() {
```

```
        return price;
    }

    public void setPrice(float price) {
        this.price = price;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public int getAvailableUnits() {
        return availableUnits;
    }

    public void setAvailableUnits(int availableUnits) {
        this.availableUnits = availableUnits;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public static Product fetchProduct(int productId,
Connection con) throws SQLException {

        String query = "select photo, name, description,
rating, rating_count, price, available_units " +
```

```
                "from product " +
                "where id=?";
        Product product = null;

        try(
                PreparedStatement statement =
con.prepareStatement(query))
        {

            statement.setInt(1,productId);
            ResultSet resultSet = statement.executeQuery();

            while (resultSet.next()){
                product = new Product();
                //set attributes
                product.setId(productId);

product.setPhoto(PhotoHelper.convertBlobToBitmap(resultSet.g
etBlob(1)));
                product.setPrice(resultSet.getFloat(6));
                product.setName(resultSet.getString(2));

product.setDescription(resultSet.getString(3));

product.setAvailableUnits(resultSet.getInt(7));
                product.setRating(resultSet.getFloat(4));
                product.setRatingCount(resultSet.getInt(5));
            }

        } catch (SQLException throwables) {
            throw throwables;
        }

        return product;
    }

    public static boolean insertProduct(Product product,
String sellerId, Connection con) throws IOException,
SQLException {

        boolean success = false;
        String query = "insert into product(seller_id,
photo, available_units, name, description, price) " +
                "values(?, ?, ?, ?, ?, ?)";
```

```
        try(
                PreparedStatement statement =
con.prepareStatement(query)
        )
        {

            statement.setString(1, sellerId);
            statement.setBinaryStream(2,
PhotoHelper.compressAndConvertToIS(product.getPhoto()));
            statement.setInt(3,
product.getAvailableUnits());
            statement.setString(4, product.getName());
            statement.setString(5,
product.getDescription());
            statement.setFloat(6, product.getPrice());

            int rowAffected = statement.executeUpdate();

            if(rowAffected>0){
                success = true;
            }

        } catch (SQLException | IOException throwables) {
            throw throwables;
        }

        return success;
    }

    public static boolean deleteProduct(int productId,
Connection con) throws SQLException {

        boolean success = false;
        String query = "delete from product where id=?";

        try(
                PreparedStatement statement =
con.prepareStatement(query))
        {

            statement.setInt(1,productId);

            int rowAffected = statement.executeUpdate();
```

```
            if(rowAffected>0){
                success = true;
            }

        } catch (SQLException throwables) {
            throw throwables;
        }

        return success;
    }

    public static boolean updateProduct(Product product,
Connection con) throws IOException, SQLException {

        boolean success = false;
        String query =  "update product " +
                        "set name=?, description=?, photo=?,
price=? " +
                        "where id=?";

        try(
                PreparedStatement statement =
con.prepareStatement(query)
        )
        {

            statement.setString(1,product.getName());
            statement.setString(2,
product.getDescription());
            statement.setBinaryStream(3,
PhotoHelper.compressAndConvertToIS(product.getPhoto()));
            statement.setFloat(4,product.getPrice());

            int rowAffected = statement.executeUpdate();

            if(rowAffected>0){
                success = true;
            }

        } catch (SQLException | IOException throwables) {
            throw throwables;
        }
```

```
        return success;
    }

    public static ArrayList<Product> searchItems(String
search, Connection con) throws SQLException {

        ArrayList<Product> result = new ArrayList<>();

        String query =  "select id, photo, name,
description, rating, rating_count, price, available_units
\n" +
                        "from product \n" +
                        "where match(name) against(?) or
match(description) against(?)";

        try(
                PreparedStatement statement =
con.prepareStatement(query)
        )
        {

            statement.setString(1, search);
            statement.setString(2, search);

            ResultSet resultSet = statement.executeQuery();

            while (resultSet.next()){
                Product product = new Product();
                product.setId(resultSet.getInt(1));

product.setPhoto(PhotoHelper.convertBlobToBitmap(resultSet.g
etBlob(2)));
                product.setName(resultSet.getString(3));

product.setDescription(resultSet.getString(4));
                product.setRating(resultSet.getFloat(5));
                product.setRatingCount(resultSet.getInt(6));
                product.setPrice(resultSet.getFloat(7));

product.setAvailableUnits(resultSet.getInt(8));
                result.add(product);
            }

        } catch (SQLException throwables) {
```

```
            throw throwables;
        }

        return result;
    }

    public boolean addRating(float rating, Connection con)
throws SQLException {
        boolean success = false;
        String query =  "update product set rating=?,
rating_count=? " +
                        "where id=?";

        try(
                PreparedStatement statement =
con.prepareStatement(query)
        )
        {

            float total = (this.rating * ratingCount) +
rating;
            ratingCount++;
            this.rating = total/ratingCount;
            statement.setFloat(1, this.rating);
            statement.setInt(2,ratingCount);
            statement.setInt(3,id);

            int rowAffected = statement.executeUpdate();

            if(rowAffected>0){
                success = true;
            }

        } catch (SQLException throwables) {
            throw throwables;
        }

        return success;
    }

}
```

Seller

```
package com.tushar.onlinecityshop;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.sql.Blob;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

public class Seller {

    private String sellerId;
    private String registrationDate;
    private String serviceCity;
    private String contactNumber;
    private String address;
    private String shopName;
    private String description;
    private Bitmap banner;
    private float rating;
    private int ratingCount;
    private String gstNo;

    public Seller() {

    }

    public void setSellerId(String sellerId) {
        this.sellerId = sellerId;
    }

    public String getSellerId() {
        return sellerId;
    }

    public Bitmap getBanner() {
```

```
        return banner;
    }

    public void setBanner(Bitmap banner) {
        this.banner = banner;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getRegistrationDate() {
        return registrationDate;
    }

    public void setRegistrationDate(String registrationDate)
{
        this.registrationDate = registrationDate;
    }

    public String getServiceCity() {
        return serviceCity;
    }

    public void setServiceCity(String serviceCity) {
        this.serviceCity = serviceCity;
    }

    public String getContactNumber() {
        return contactNumber;
    }

    public void setContactNumber(String contactNumber) {
        this.contactNumber = contactNumber;
    }

    public String getAddress() {
        return address;
    }
```

```
    public void setAddress(String address) {
        this.address = address;
    }

    public String getShopName() {
        return shopName;
    }

    public void setShopName(String shopName) {
        this.shopName = shopName;
    }

    public float getRating() {
        return rating;
    }

    public void setRating(float rating) {
        this.rating = rating;
    }

    public int getRatingCount() {
        return ratingCount;
    }

    public void setRatingCount(int ratingCount) {
        this.ratingCount = ratingCount;
    }

    public String getGstNo() {
        return gstNo;
    }

    public void setGstNo(String gstNo) {
        this.gstNo = gstNo;
    }

    public static boolean addNewSeller(Seller seller,
Connection con) throws SQLException, IOException {

        boolean success = false;
        String query = "insert into seller(seller_id,
contact_no, banner, shop_name, description, address,
service_city, gst_no)\n" +
                " values (?,?, ?, ?, ?, ?, ?, ?)";
```

```
        String query1 = "update user set is_seller=? where
contact_no=?";

        try (
                PreparedStatement statement =
con.prepareStatement(query);
                PreparedStatement statement1 =
con.prepareStatement(query1)
        ){

            statement.setString(1,seller.getSellerId());

statement.setString(2,seller.getContactNumber());
            statement.setBinaryStream(3,

PhotoHelper.compressAndConvertToIS(seller.getBanner()));
            statement.setString(4, seller.getShopName());
            statement.setString(5, seller.getDescription());
            statement.setString(6, seller.getAddress());
            statement.setString(7, seller.getServiceCity());
            statement.setString(8,seller.getGstNo());

            int rowAffected = statement.executeUpdate();

            if(rowAffected>0){
                success = true;
            }

            statement1.setBoolean(1, true);
            statement1.setString(2, seller.getSellerId());
            statement1.executeUpdate();

        } catch (SQLException throwables) {
            throw throwables;
        } catch (IOException e) {
            throw e;
        }

        return success;
    }

    public static boolean deleteSeller(Seller seller,
Connection con) throws SQLException {
```

```
        boolean success = false;
        String query = "delete from seller where
contact_no=?";
        String query1 = "update user set is_seller=? where
contact_no=?";

        try (
                PreparedStatement statement =
con.prepareStatement(query);
                PreparedStatement statement1 =
con.prepareStatement(query1)
        ){


statement.setString(1,seller.getContactNumber());

            int rowAffected = statement.executeUpdate();

            if(rowAffected>0){
                success = true;
            }

            statement1.setBoolean(1, false);
            statement1.setString(2, seller.getSellerId());
            statement1.executeUpdate();

        } catch (SQLException throwables) {
            throw throwables;
        }

        return success;
    }

    public static ArrayList<Product>
fetchProductsOfSeller(String sellerId, Connection con)
throws SQLException {

        String query = "select p.name, p.photo, p.id,
p.rating, p.rating_count, p.seller_id, p.price,
p.description " +
                "from product as p " +
                "where p.seller_id=?";
        ArrayList<Product> arrList = new ArrayList<>();
```

```
        try (
                PreparedStatement statement =
con.prepareStatement(query)
        ){

            statement.setString(1,sellerId);

            ResultSet rs=statement.executeQuery();

            while(rs.next())
            {
                Product product=new Product();

                product.setName(rs.getString(1));

product.setPhoto(PhotoHelper.convertBlobToBitmap(rs.getBlob(
2)));
                product.setId(rs.getInt(3));
                product.setRating(rs.getFloat(4));
                product.setRatingCount(rs.getInt(5));
                product.setSellerId(rs.getString(6));
                product.setPrice(rs.getFloat(7));
                product.setDescription(rs.getString(8));

                arrList.add(product);
            }

        } catch (SQLException throwables) {
            throw throwables;
        }

        return arrList;
    }

    public static boolean updateSeller(Seller seller,
Connection con) throws SQLException, IOException {

        boolean success = false;
        String query = "update seller " +
                "set shop_name=?, description=?, address=?,
service_city=?, gst_no=?, banner=?, contact_no=? " +
                "where seller_id=?";

        try (
```

```
                PreparedStatement statement =
con.prepareStatement(query)
        ){

            statement.setString(1,seller.getShopName());
            statement.setString(2, seller.getDescription());
            statement.setString(3, seller.getAddress());
            statement.setString(4, seller.getServiceCity());
            statement.setString(5,seller.getGstNo());

statement.setBinaryStream(6,PhotoHelper.compressAndConvertTo
IS(seller.getBanner()));
            statement.setString(7,
seller.getContactNumber());
            statement.setString(8, seller.getSellerId());

            int rowAffected = statement.executeUpdate();

            if(rowAffected>0){
                success = true;
            }

        } catch (SQLException throwables) {
            throw throwables;
        } catch (IOException e) {
            throw e;
        }

        return success;
    }

    public static boolean addRating(Seller seller, float
rating, Connection con) throws SQLException {

        boolean success = false;
        String query = "update seller set rating=?,
rating_count=? " +
                "where seller_id=?";

        try (
                PreparedStatement statement =
con.prepareStatement(query)
        ){
```

```
            float total = (seller.rating *
seller.ratingCount) + rating;
            System.out.println(seller.getSellerId());
            seller.ratingCount++;
            seller.rating = total/seller.ratingCount;
            statement.setFloat(1, seller.getRating());
            statement.setInt(2, seller.getRatingCount());
            statement.setString(3,seller.getSellerId());
            System.out.println(seller.getRating());

            int rowAffected = statement.executeUpdate();

            if(rowAffected>0){
                success = true;
            }

        } catch (SQLException throwables) {
            throw throwables;
        }

        return success;
    }

    public static ArrayList<Seller> searchSeller(String
search, String city, Connection con) throws SQLException {

        ArrayList<Seller> result = new ArrayList<>();

        String query = "select seller_id, shop_name,
description, rating, rating_count, banner, address " +
                "from seller " +
                "where (match(shop_name) against(?) or
match(description) against(?))" +
                "      and service_city=?";

        try(
                PreparedStatement statement =
con.prepareStatement(query)
        ){

            statement.setString(1, search);
            statement.setString(2, search);
            statement.setString(3, city);
```

```
            ResultSet resultSet = statement.executeQuery();

            while (resultSet.next()){

                Seller seller = new Seller();

                seller.setSellerId(resultSet.getString(1));
                seller.setShopName(resultSet.getString(2));

seller.setDescription(resultSet.getString(3));
                seller.setRating(resultSet.getFloat(4));
                seller.setRatingCount(resultSet.getInt(5));

seller.setBanner(PhotoHelper.convertBlobToBitmap(resultSet.g
etBlob(6)));
                seller.setAddress(resultSet.getString(7));

                result.add(seller);
            }

        } catch (SQLException throwables) {
            throw throwables;
        }

        return result;
    }

    public static Seller fetchSellerWithUserId(String
userId, Connection con) throws SQLException {
        String query = "select shop_name, description,
rating, rating_count, banner, contact_no, address,
service_city, gst_no, seller_id " +
                "from seller " +
                "where seller_id=?";
        Seller seller = null;

        try(
                PreparedStatement statement =
con.prepareStatement(query)
        ){

            statement.setString(1, userId);

            ResultSet resultSet = statement.executeQuery();
```

```
            while (resultSet.next()){
                seller = new Seller();

                seller.setSellerId(userId);
                seller.setShopName(resultSet.getString(1));

seller.setDescription(resultSet.getString(2));
                seller.setRating(resultSet.getFloat(3));
                seller.setRatingCount(resultSet.getInt(4));

seller.setBanner(PhotoHelper.convertBlobToBitmap(resultSet.g
etBlob(5)));

seller.setContactNumber(resultSet.getString(6));
                seller.setAddress(resultSet.getString(7));

seller.setServiceCity(resultSet.getString(8));
                seller.setGstNo(resultSet.getString(9));
                seller.setSellerId(resultSet.getString(10));
            }

        } catch (SQLException throwables) {
            throw throwables;
        }

        return seller;

    }

    public static ArrayList<Seller> fetchSellerInCity(String
city, Connection con) throws SQLException {

        ArrayList<Seller> arrList = new ArrayList<>();
        String query = "select seller_id, shop_name,
description, rating, rating_count, banner, address " +
                       "from seller " +
                       "where service_city=?";

        try (
                PreparedStatement
statement=con.prepareStatement(query)
        ){
```

```
            statement.setString(1,city);
            ResultSet resultSet = statement.executeQuery();

            while (resultSet.next()){
                Seller seller = new Seller();

                seller.setSellerId(resultSet.getString(1));
                seller.setShopName(resultSet.getString(2));

seller.setDescription(resultSet.getString(3));
                seller.setRating(resultSet.getFloat(4));
                seller.setRatingCount(resultSet.getInt(5));

seller.setBanner(PhotoHelper.convertBlobToBitmap(resultSet.g
etBlob(6)));
                seller.setAddress(resultSet.getString(7));

                arrList.add(seller);
            }

        }catch (SQLException throwables) {
            throw throwables;
        }

        return arrList;
    }

}
```

PhotoHelper

```
package com.tushar.onlinecityshop;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.sql.Blob;
import java.sql.SQLException;

public abstract class PhotoHelper {
```

```
    /**
     * This method helps us to convert a blob object to a
bitmap
     * Takes blob as input and returns a bitmap
     */
    public static Bitmap convertBlobToBitmap(Blob blob)
throws SQLException {
        byte array[] = blob.getBytes(1, (int)blob.length());
        Bitmap bitmap = BitmapFactory.decodeByteArray(array,
0, array.length);
        return bitmap;
    }

    /**
     * This method will take bitmap as input and compress
and
     * convert it to input stream so that we can upload the
photo
     * to our data base
     */

    public static ByteArrayInputStream
compressAndConvertToIS(Bitmap bitmap) throws IOException {

        ByteArrayOutputStream stream = new
ByteArrayOutputStream();
        ByteArrayOutputStream size = new
ByteArrayOutputStream();
        /**
         * This compress method compute bitmap to size
stream
         */
        bitmap.compress(Bitmap.CompressFormat.PNG, 100,
size);
        int quality = 40000000/(size.toByteArray().length);

        if(quality<10){
            quality=10;
        }else if(quality > 100){
            quality = 100;
        }
        size.close();

        /**
```

```
         * This time compress method will compress the
bitmap
         */
        bitmap.compress(Bitmap.CompressFormat.JPEG, quality,
stream);
        byte array[] = stream.toByteArray();

        ByteArrayInputStream inputStream = new
ByteArrayInputStream(array);

        return inputStream;
    }
}
```