<u>Embedded Queries</u>

1. A logged in admin will have the option to view the price and quantity available of all the products in his/her area

```
def viewAvailableProducts(pincode):
    mycursor.execute(f"select product.product_name, product.price, available.quantity from product,available where product.product_id = available.product_id and available.pincode = {pincode};")
    listprods = mycursor.fetchall()
    print("Product_Name\tPrice\tQuantity")
    for i in listprods:
        print(f"{i[0]}\t{i[1]}\t{i[2]}")
    return 0
```

2. A customer can add products to their cart if the product is available in their area

```
def addToCart(customer_id,pincode):
    product_id = int(input("Enter product id:"))
    quantity = int(input("Enter quantity"))
    mycursor.execute(f"insert into cart (select {customer_id}, {product_id},{quantity} from available where available.pincode = {pincode} and available.product_id ={product_id} and available.quantity>{quantity});")
    print("Done")
    return 0
```

<u>OLAP Queries</u>

1. Viewing all orders on different order dates and delivery dates, and then all orders on a single order date, and then all orders on a single delivery date(cube)

```
select if(grouping(order_date)=1,'all order dates',order_date) as order_date,if(grouping(delivery_date)=1,'all delivery dates',delivery_date) as delivered_on,count(total_price) as no_of_orders

from orders

group by order_date,delivery_date with rollup

union
```

```
select if(grouping(order_date)=1,'all order dates',order_date) as
order_date,if(grouping(delivery_date)=1,'all delivery dates',delivery_date) as
delivered_on,count(total_price) as no_of_orders

from orders group by delivery_date,order_date with rollup;
```

2. Viewing no of products in cart of different users with different product ids , and then all ordrs in the cart of a single user(rollup)

```
select if(grouping(user_ID)=1,'products in cart of all users',user_ID) as
User_ID,if(grouping(product_ID)=1,'all products in the cart of user',product_ID) as Product_ID,
sum(quantity) as no_of_products

from cart

group by user_ID,product_ID with rollup

order by grouping(user_ID) desc;
```

3. Veiwing price or product grouping with only product id (pivot table)

```
select product_ID,product_name,sum(price)

from product group by product_ID,product_name with rollup

having grouping(product_name) = 1;
```

4. Viewing coupons the store has offered upto a date to a given user, and then all the coupons offered upto that date(drill down)

```
select valid_until_date,user_id, max(discount_offered) as max_discount,count(coupon_id) as
total_no_of_coupons_given

from coupon

group by valid_until_date,user_id with rollup

order by( grouping(user_id)+grouping(valid_until_date)) asc;
```

<u>Triggers</u>

1. Check whether price is not null

```
DROP TRIGGER IF EXISTS `flipmart`.`orders_BEFORE_INSERT_1`;


DELIMITER $$
USE `flipmart`$$
CREATE DEFINER = `root`@`localhost` TRIGGER `flipmart`.`orders_BEFORE_INSERT_1` BEFORE
INSERT ON `orders` FOR EACH ROW PRECEDES `orders_BEFORE_INSERT`
if
new.total_price is null
then
set new.total_aprice = 0;
end if;
END
$$
DELIMITER ;
```

2.Insert cart items into products in order relation on placing an order

```
DROP TRIGGER IF EXISTS `flipmart`.`orders_BEFORE_INSERT`;


DELIMITER $$
CREATE DEFINER=`root`@`localhost` TRIGGER `orders_BEFORE_INSERT` BEFORE INSERT ON `orders`
FOR EACH ROW IF
(select order_id from products_in_order where order_id = new.order_id) is null
then
insert into products_in_order


        (select new.order_id,cart.user_id,cart.product_id,cart.quantity
        from cart
```

```
            where cart.user_ID = new.user_id);
end if
END
$$
DELIMITER ;
```

3. Update quantity in available when deleting items from cart

```
DROP TRIGGER IF EXISTS `flipmart`.`cart_BEFORE_DELETE`;


DELIMITER $$
USE `flipmart`$$
CREATE DEFINER=`root`@`localhost` TRIGGER `cart_BEFORE_DELETE` BEFORE DELETE ON `cart` FOR
EACH ROW BEGIN
if
quantity>old.quantity then
update available
set quantity =quantity - (select quantity from cart
                                                        where user_id = old.user_id
                  and product_id = old.product_id)
where available.pincode = (select pincode
                                                                from orders
                   where user_id =old.user_id
                   and order_id = (select max(order_id) from orders
                                                                                                    where
user_id = old.user_id))
and available.product_id = old.product_id;
end if;
END$$
DELIMITER ;
```