DOCUMENTATION (DEADLINE- 5)

NAME – SHANTANU PRAKASH (2021285), SHIVESH GULATI (2021286)

GROUP –25 ONLINE RETAIL STORE (FLYING GROCERIES)

Embedded SQL Queries:

1) `SELECT RequestId, User_FirstName, User_MiddleName, User_LastName,PhoneNumber FROM TransporterRequests ORDER BY RequestId`

   Displays all the transporter requests on the admin page which is ordered by the RequestId (Primary key of TransporterRequests Table)

2) `UPDATE TransporterRequests SET RequestStatus = %s WHERE RequestId = %s`
   `DELETE FROM TransporterRequests Where RequestId = %s`

   Based on the accept/reject input by the admin on the frontend, changes are made in the database using the update query in transporter requests table and the entry is then deleted from TransporterRequests table using delete query which is further handled by trigger.

3) `SELECT * FROM Category ORDER By CategoryId`

   Displays all the categories on the shop page of front end ordered by CategoryId.

4) `SELECT * FROM SubCategory WHERE CategoryId = %s Order By SubCategoryId`

   Displays all the SubCategories corresponding to the selected category.

5) `SELECT * FROM PRODUCT WHERE CategoryId = %s and SubCategoryId = %s Order By ProductId`

   Displays all the products corresponding to selected Category and SubCategory.

Triggers Implemented:

1)
```
delimiter //

CREATE TRIGGER del_and_insert_req

BEFORE DELETE

ON TransporterRequests

FOR EACH ROW

BEGIN

if old.RequestStatus =1 THEN

INSERT INTO
Transporter(UserName,User_FirstName,User_MiddleName,User_LastName,PhoneNumber,AadharNumber,Driving
LicenseNumber)

SELECT CONCAT(User_FirstName,RequestId), User_FirstName,
User_MiddleName,User_LastName,PhoneNumber,AadharNumber,DrivingLicenseNumber

FROM TransporterRequests

WHERE RequestId = OLD.RequestId;

end if;

end//

delimiter ;
```

This Trigger runs before deleting an entry from TransporterRequests. Whenever RequestStatus of any TransporterRequests entry is set to 1 on admin page at frontend, it indicates that the request of the transporter has been accepted by the admin and the record of this transporter request is inserted into Transporter Table.

2)
```
delimiter //

CREATE TRIGGER del_and_insert_cart

AFTER DELETE

ON Cart

FOR EACH ROW

BEGIN

SELECT TransporterId INTO @Transporter_Id FROM Transporter WHERE Transporter.ActiveStatus = 1 and
Transporter.currentActiveOrders<10 ORDER BY TransporterId LIMIT 1;

SELECT CurrentActiveOrders INTO @Active_Orders FROM Transporter WHERE Transporter.TransporterId =
@Transporter_Id;

SELECT QUANTITY INTO @Prod_Quantity FROM PRODUCT WHERE Product.ProductId = OLD.ProductId and
Product.CategoryId = OLD.CategoryID and Product.SubCategoryId = OLD.SubCategoryId;

SELECT COUNT(ProductId) INTO @TOT_PROD FROM Cart WHERE CustomerId = OLD.CustomerId Group By
CustomerId
```

```
if @Prod_Quantity>OLD.Quantity THEN

INSERT INTO Delivery
(PaymentId,CustomerId,TransporterId,CategoryId,SubCategoryId,ProductId,Quantity,PaymentType)
VALUES
(2000+Old.CustomerId,Old.CustomerId,@Transporter_Id,OLD.CategoryId,OLD.SubCategoryId,OLD.ProductId
,OLD.Quantity,"PayTM");

UPDATE Transporter

SET CurrentActiveOrders = CurrentActiveOrders +1

WHERE CurrentActiveOrders<10 and Transporter.TransporterId = @Transporter_Id;

    UPDATE Product

SET Quantity = Quantity - OLD.Quantity

WHERE Product.ProductId = OLD.ProductId and Product.CategoryId = OLD.CategoryId and
Product.SubcategoryId = OLD.subCategoryId ;

if @Active_Orders =9 THEN

UPDATE Transporter

SET ActiveStatus = 0

WHERE TransporterId  = @Transporter_Id;

end if;

end if;

end //

delimiter ;
```

This Trigger runs after deleting an entry from cart whose order is to be placed. To Place the order for the deleted entry of cart, we select transporterId and CurrentActiveOrders of a transporter to be assigned to the order into variables. Trigger ensures that customer cannot place an order for a product in a quantity which is greater than the quantity of product present in the product table representing the stock of products. Trigger also ensures changes are made in transporter table corresponding to the transporter who has been allotted this order.

OLAP Queries:

1)
```
SELECT CategoryId, SubCategoryId, ProductId, count(CustomerId) as 'NumberOfCustomers'
FROM Cart
GROUP BY CategoryId, SubCategoryId, ProductId WITH ROLLUP;
```

It Displays all records from the cart table in which total number of customers is shown grouped by (categoryId), (CategoryId,SubCategoryId), (CategoryId,SubCategoryId,ProductId) using ROLLUP.

2)
```
SELECT CheckOutDateAndTime, ProductId, count(OrderId), GROUPING(CheckOutDateAndTime), GROUPING(ProductId)
FROM Delivery
GROUP BY CheckOutDateAndTime, ProductId WITH ROLLUP;
```

It Displays all records from the Delivery Table which display the number of orders placed on the site on different dates and number of orders placed on the site on different dates for different products.

3)
```
SELECT TIMESTAMPDIFF(DAY,CURDATE(),product.ExpiryDate) as 'TimeLeft', Discount, count(productId)
FROM Product
GROUP BY TimeLeft, Discount WITH ROLLUP;
```

It Displays all total number of products from the Product Table grouped by (TimeLeftForUse), (TimeLeftForUse, Discount) indicating discount and TimeLeftForUse for products and can help the admin gain a useful insight about what discount has been set on products which are close to expiry and can help them gain profits by analysing this data.

4)
```
SELECT CustomerId, ProductId, CheckoutDateAndTime, count(Quantity)
FROM Delivery
GROUP BY CustomerId, ProductId, CheckOutDateAndTime WITH ROLLUP HAVING (GROUPING(CustomerId)=0 and
GROUPING(ProductId)=0) or (GROUPING(ProductId)=1 and GROUPING(CustomerId)=0);
```

It Displays all entries from Delivery grouped by (CustomerId, ProductId), (CustomerId). It gives a useful insight of orders of which products are placed and at what times are they placed by the customer at different dates.

5)
```
SELECT AdminId,RequestStatus,Count(RequestId)
FROM Reviews JOIN TransporterRequests on TransporterRequests.RequestId=Reviews.RequestId
GROUP BY Cube (AdminId,RequestStatus);
```

It Displays entries from table (Reviews JOIN TransporterRequests) and displays records grouped by all combinations of AdminId, RequestStatus using Cube providing an insight of how many requests accepted, declined and on hold.

6)
```
SELECT Transporter.TransporterId, Transporter.ActiveStatus, count(OrderId)
FROM GetsAssigned LEFT JOIN Transporter ON GetsAssigned.TransporterId=Transporter.TransporterId
GROUP BY TransporterId, ActiveStatus WITH ROLLUP;
```

It Displays entries from table (GetsAssigned LEFT JOIN Transporter) and displays records grouped by TransporterId, ActiveStatus with ROllup providing an insight of how many orders have been allotted to transporters along with their active status