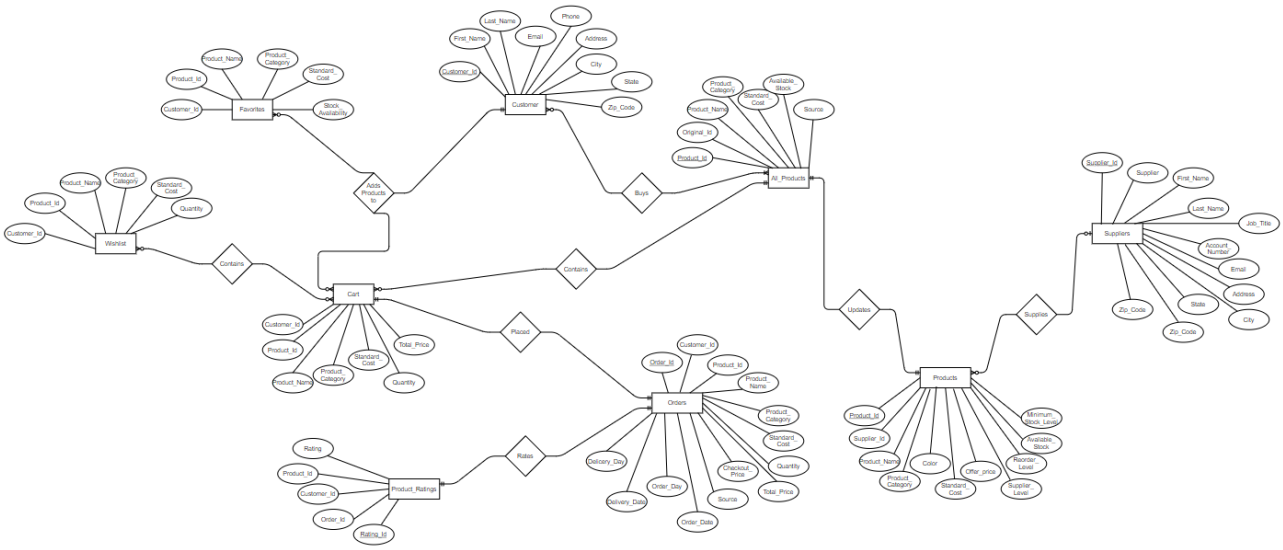
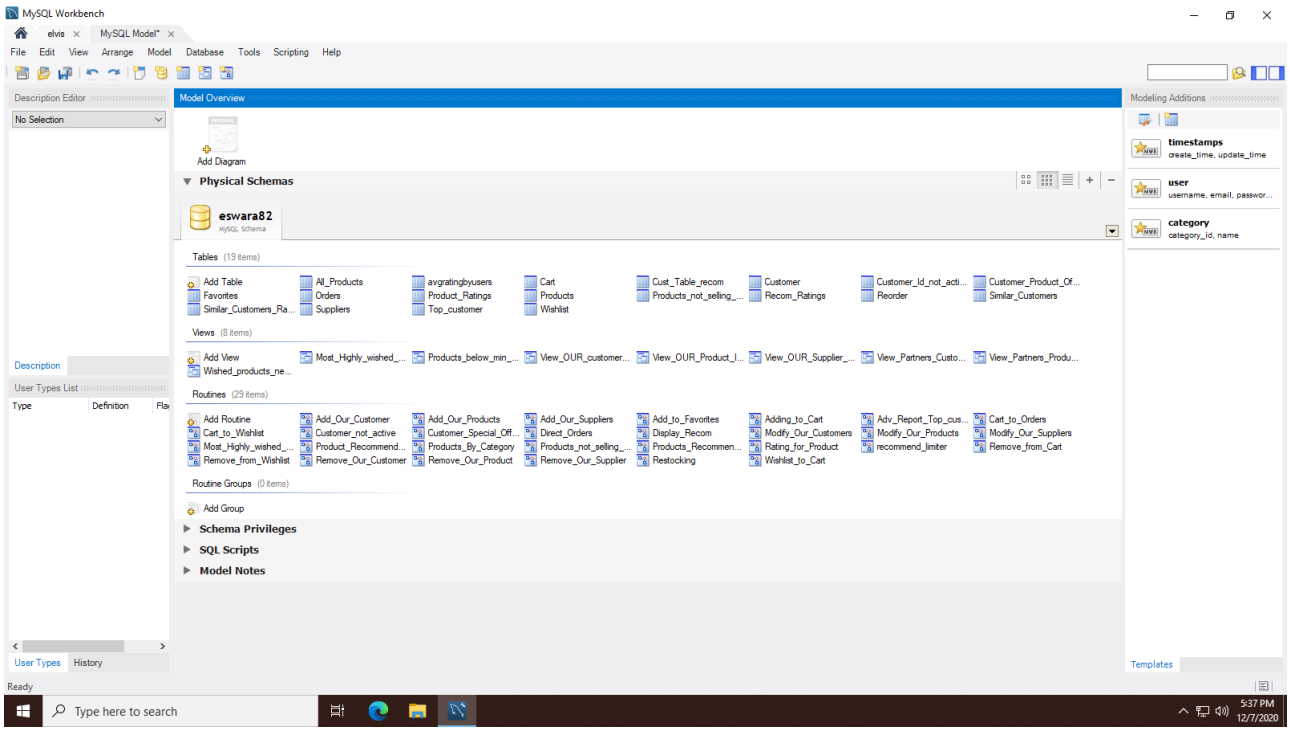
Design Document

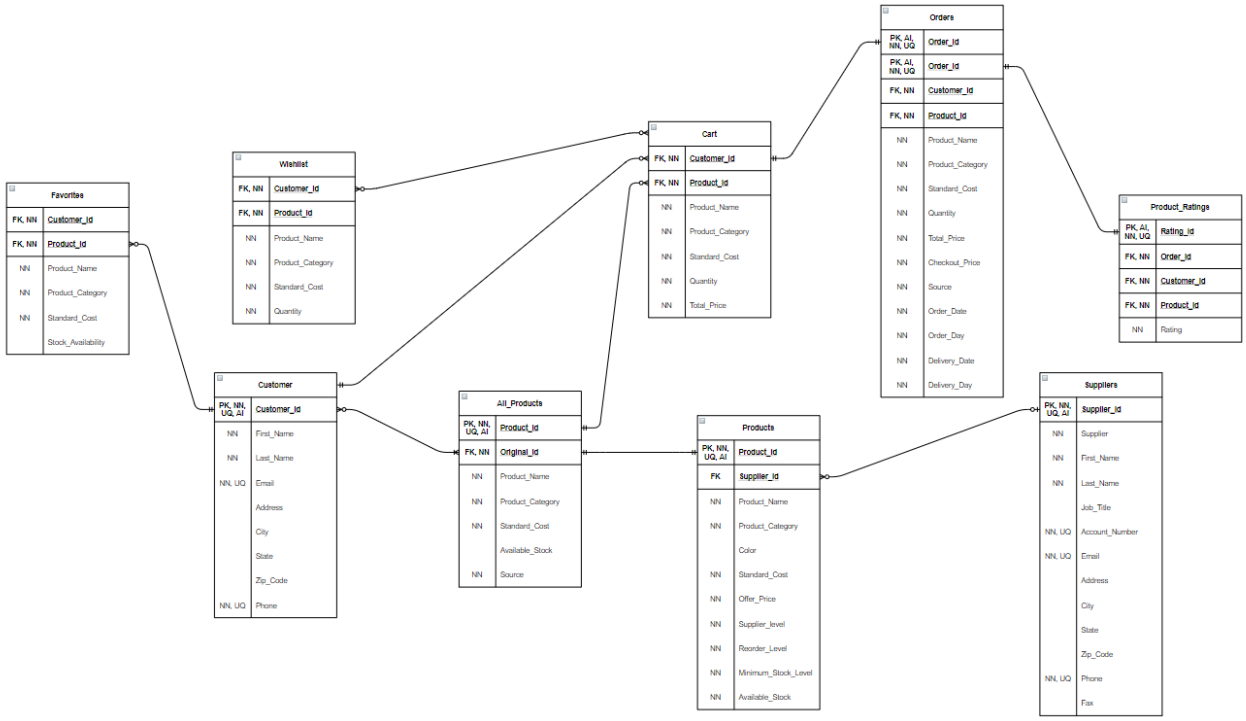
Team:
Eswara Bharat Teja Mamidi
Lakshmi Karthik Kallepalli

ER Diagram:



Physical Schema diagram from MySQL





- **<u>All SQL statements that support the basic and advanced functionality and reports of the system</u>**

1. Ability to view, add, remove, and modify your customer information

View our Customer Information: VIEW `View_OUR_customer_information`

Query: SELECT * FROM eswara82.View_OUR_customer_information;

Add_Our_customer

call eswara82.Add_Our_Customer('Jack', 'Sparrow', 'jack_sparrow@gmail.com', '10th Street', 'Glassboro', 'New Jersey', '08028', '9463157985');

Modify_Our_Customers

call eswara82.Modify_Our_Customers(8, '', '', '', '31st street', 'Detroit', 'Michigan', '645899', '9465871365');

Remove_Our_Customer

call eswara82.Remove_Our_Customer(31);

2. Ability to view your partners' customer information

VIEW `View_Partners_Customer_Info`

Query: SELECT * FROM eswara82.View_Partners_Customer_Info;

3. Ability to view, add, remove, and modify your supplier information

VIEW `View_OUR_Supplier_Info`

Query: SELECT * FROM eswara82.View_OUR_Supplier_Info;

Add_Our_Suppliers

call eswara82.Add_Our_Suppliers('Supplier U', 'Alex', 'Johnson', 'Sales Manager', 123456768, 'alex_john1@gmail.com', '1245th Street', 'Seattle', 'Washington', '136547', '5463124789', '2654785236');

Modify_Our_Supplier

call eswara82.Modify_Our_Suppliers(12, '', '', '', 'Sales Supervisor', '', '', '16th Avenue', 'Newark', 'New Jersey', '021569', '', '');

Remove_Our_Supplier

call eswara82.Remove_Our_Supplier(16);

4. Ability to view, add, remove, and modify your product information

VIEW `View_OUR_Product_Information_Inventory`

Query: SELECT * FROM eswara82.View_OUR_Product_Information_Inventory;

Add_Our_Products

call eswara82.Add_Our_Products(14, 'Havit Gaming Mouse', 'Electronics', 'Black', 36, 6, 15, 10, 5, 10);

Modify_Our_Products

call eswara82.Modify_Our_Products(46, 14, 'Apple iPhone 11 Mini', '', 'Yellow', 699, 59, 10, 10, 5, 15);

Remove_Our_Product

call eswara82.Remove_Our_Product(45);

5. Ability to view, your partners' product information

VIEW `View_Partners_Product_Info`

Query: SELECT * FROM eswara82.View_Partners_Product_Info;

6. Ability to view your product inventory

View `View_OUR_Product_Information_Inventory`

Query: SELECT * FROM eswara82.View_OUR_Product_Information_Inventory;

7. Ability to generate a restocking order (should be saved in a "restocking" table) if the supply of any of your products falls below the minimum stock level

CALL `eswara82`.`Restocking`();

Restocking procedure will **insert data into reorder table** for products if the supply of any of your products falls below the minimum stock level

8. Ability of a customer to place an order, which consists of adding your or your partners' items to a shopping cart and then checking out.

9. Ability to browse the product catalog by category (We know that Northwind sells food items, AdventureWorks sells bikes and accessories, and Sakila sells movies. Your product catalog should include items from your partners' categories as well as items from other categories that your partners do not traffic in.

CALL `eswara82`.`Products_By_Category`('Beverages');

CALL `eswara82`.`Products_By_Category`('Electronics');

CALL `eswara82`.`Products_By_Category`('Baked Goods & Mixes');

CALL `eswara82`.`Products_By_Category`('Condiments');

CALL `eswara82`.`Products_By_Category`('Canned Fruit & Vegetables');

CALL `eswara82`.`Products_By_Category`('Dairy Products');

/* Partners Products Categories*/

CALL `eswara82`.`Products_By_Category`('Accessories');

CALL `eswara82`.`Products_By_Category`('Bikes');

CALL `eswara82`.`Products_By_Category`('Components');

CALL `eswara82`.`Products_By_Category`('Clothing');

CALL `eswara82`.`Products_By_Category`('Action');

10. List of all your products whose inventory has fallen below the minimum stock level

VIEW `Products_below_min_stock`

Query: SELECT * FROM eswara82.Products_below_min_stock;

11. List of customers who have not been “too active”(you define this) and for whom special offers should be made

Definition: Customers never placed an Order, nor placed any product into Cart or never placed any product into Wishlist - so they are defined as INACTIVE customers and special offers can be made to them

PROCEDURE : CALL `eswara82`.`Customer_not_active`();

12. List of products that are not selling “too well”(you define this), which might be offered as specials.

Definition: Products that were not sold at any time and never been into Orders table are considered to be products not selling too well

PROCEDURE : CALL `eswara82`.`Products_not_selling_well`();

13. When the products purchased will ship (Shipping will occur four weekdays from now, e.g., if today is Monday, they will ship on Friday.

14. Ability of a customers to place an item on his/her “wish list.”

Can place an item into Wishlist from CART - Just like Buy Later

PROCEDURE :

/*Add to Cart*/

CALL `eswara82`.`Adding_to_Cart`(20, 105, 1);

/*Then add to Wishlist*/

CALL `eswara82`.`Cart_to_Wishlist`(20, 105);

15. An algorithm (manifested as a query) to suggest additional products that a customer might be interested in based on their order history, their wish list, or anything else you would like to program.

Recommendations:

Recommendation system may either produce top-*k* ranking (list of “best” items) or prediction of ratings. The focus of the result may be generic (everyone receives the same recommendations), demographic (everyone in the same category receives the same recommendations) or personal.

Over here we have concentrated on Personal recommendation where, recommendations are based on the personal behaviour which might be similar to a similar user . The context may rely on the user’s current activity or on her/his long-term interests.

1. Collaborative filtering -

• **User-based collaborative filtering -** Building User to user similarity matrix to find out the similar user by calculating the distance between two.

The correlation between a pair of users is computed by comparing their given Product ratings with Average Ratings.

In a simple way we find the correlation between you and all the other users and calculated the scores(distances between one user to all).

2. Correlation:

There exist several possible measures of correlations - we have used - cosine similarity (Formula used can be seen in the procedure below)

3. Calculate the similarity between the one user to all then store the correlated users whose behaviour is close to yours into a table called "Similar_Customers"

Procedures:

CALL `eswara82`.`Products_Recommendation_Tables_update`(); This will update all the recommendations related table like "avgratingbyusers", "Recom_Ratings","Similar_Customers" CALL `eswara82`.`Product_Recommendation`(<{IN Cust_Id INT}>);

- From this we will take the similar customers based on the TOP RANK which is associated with the score(Stored in Similar customer Ranking table).
- Displays the TOP 10 customers based on the Customer_ID which is Passed through INPUT parameters
- Then it is joined to the Orders to see what products are ordered and in those TOP 10 products were recommended to the Customer which we passed through the procedure.

Queries:

CALL `eswara82`.`Products_Recommendation_Tables_update`();

CALL `eswara82`.`Product_Recommendation`(17);

CALL `eswara82`.`Product_Recommendation`(49);

CALL `eswara82`.`Product_Recommendation`(22);

16. Ability of customers to rate products

Able to rate the product when Order is Placed, and ratings are stored in "Product_Ratings" table which is then used for providing recommendations as stated above - by calculating the similarity distances between two users

PROCEDURE : CALL `eswara82`.`Rating_for_Product`(20,4);

17. Ability to view the ratings of products in two ways

- The average rating based on all rating activity

Procedure: CALL `eswara82`.`Products_Recommendation_Tables_update`();

/*Update avgratingbyusers table to get the updated avg ratings*/

- A more intelligent rating that uses an algorithm to weight some customer's ratings higher than others.

From the above calculating the Avg ratings from the customer for what they have rated, We can predict the ratings of the products for a particular customer (Passed as INPUT through Recommendation algorithm) by calculating the similarity distance(Cosine similarity) and these ratings can be used by the algorithm to provide recommendations of products in finding the nearest customers Procedure: CALL `eswara82`.`Products_Recommendation_Tables_update`();

/* User-based collaborative filtering --------Finding correlation with cosine correlation and dumping similar_customers into Similar_Customers table */

18. A report showing the most highly wished for products in every category

Procedure: `Most_Highly_wished_products_in_every_category

CALL `eswara82`.`Most_Highly_wished_products_in_every_category`('Accessories');

CALL `eswara82`.`Most_Highly_wished_products_in_every_category`('Bikes');

CALL `eswara82`.`Most_Highly_wished_products_in_every_category`('Components');

CALL `eswara82`.`Most_Highly_wished_products_in_every_category`('Clothing');

CALL `eswara82`.`Most_Highly_wished_products_in_every_category`('Horror');

CALL `eswara82`.`Most_Highly_wished_products_in_every_category`('Electronics');

OR

VIEW `Most_Highly_wished_products_in_every_category_Report` which gives most highly wished products in every category

Query: SELECT * FROM eswara82.Most_Highly_wished_products_in_every_category_Report;

19. A report showing wished for products that were never purchased by the customers who wished for them

VIEW `Wished_products_never_purchased`

Query: SELECT * FROM eswara82.Wished_products_never_purchased;

20. EXTRA CREDIT: What other innovative reports can you think of?

Definition: Top 3 customers/orders every month in what product category for an year in a single report

PROCEDURE: CALL `eswara82`.`Adv_Report_Top_customers_per_Month`();

All Create Table Scripts for your MySQL Tables

```
CREATE TABLE `All_Products` (
 `Product_Id` int NOT NULL AUTO_INCREMENT,
 `Original_Id` int NOT NULL,
 `Product_Name` varchar(100) NOT NULL,
 `Product_Category` varchar(45) NOT NULL,
 `Standard_Cost` int NOT NULL,
 `Available_Stock` varchar(45) DEFAULT NULL,
 `Source` varchar(45) NOT NULL,
 PRIMARY KEY (`Product_Id`),
 UNIQUE KEY `Product_Id_UNIQUE` (`Product_Id`),
 KEY `FK_Product_Id_Original_Id_idx` (`Original_Id`)
) ENGINE=InnoDB AUTO_INCREMENT=4096 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `avgratingbyusers` (
 `nbusers` bigint NOT NULL DEFAULT '0',
 `avgrating` decimal(12,1) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `Cart` (
 `Customer_Id` int NOT NULL,
 `Product_Id` int NOT NULL,
 `Product_Name` varchar(45) NOT NULL,
 `Product_Category` varchar(45) NOT NULL,
 `Standard_Cost` int NOT NULL,
 `Quantity` int NOT NULL,
 `Total_Price` int NOT NULL,
 KEY `Cust_Id_FK_idx` (`Customer_Id`),
 CONSTRAINT `Cust_Id_FK` FOREIGN KEY (`Customer_Id`) REFERENCES `Customer` (`Customer_Id`) ) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `Cust_Table_recom` (
 `Customer_Id` int NOT NULL,
 PRIMARY KEY (`Customer_Id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `Customer` (
 `Customer_Id` int NOT NULL AUTO_INCREMENT,
 `First_Name` varchar(45) NOT NULL,
 `Last_Name` varchar(45) NOT NULL,
 `Email` varchar(45) NOT NULL,
 `Address` varchar(45) DEFAULT NULL,
 `City` varchar(45) DEFAULT NULL,
 `State` varchar(45) DEFAULT NULL,
 `Zip_Code` varchar(45) DEFAULT NULL,
 `Phone` varchar(10) NOT NULL,
 PRIMARY KEY (`Customer_Id`),
 UNIQUE KEY `Customer_Id_UNIQUE` (`Customer_Id`),
 UNIQUE KEY `Email_UNIQUE` (`Email`),
 UNIQUE KEY `Phone_UNIQUE` (`Phone`)
) ENGINE=InnoDB AUTO_INCREMENT=64 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `Customer_Id_not_active` (
 `Customer_Id` int NOT NULL,
 PRIMARY KEY (`Customer_Id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `Customer_Product_Offers` (
 `Customer_Id` int NOT NULL,
 `Product_Id` varchar(100) NOT NULL,
 `Product_Name` varchar(100) DEFAULT NULL,
 `Product_Category` varchar(100) DEFAULT NULL,
 `Standard_Cost` int DEFAULT NULL,
 `Offer_Price` int DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `Orders` (
 `Order_Id` int NOT NULL AUTO_INCREMENT,
 `Customer_Id` int NOT NULL,
 `Product_Id` varchar(45) NOT NULL,
 `Product_Name` varchar(45) NOT NULL,
 `Product_Category` varchar(45) NOT NULL,
 `Standard_Cost` int NOT NULL,
`Standard_Cost` int NOT NULL,
 `Quantity` int NOT NULL,
 `Total_Price` int NOT NULL,
 `Checkout_Price` float NOT NULL,
 `Source` varchar(45) NOT NULL,
 `Order_Date` varchar(25) NOT NULL,
 `Order_Day` varchar(15) NOT NULL,
 `Delivery_Date` varchar(25) NOT NULL,
 `Delivery_Day` varchar(15) NOT NULL,
 PRIMARY KEY (`Order_Id`),
 UNIQUE KEY `Order_Id_UNIQUE` (`Order_Id`)
) ENGINE=InnoDB AUTO_INCREMENT=55 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `Product_Ratings` (
 `id` int NOT NULL AUTO_INCREMENT,
 `Order_Id` int DEFAULT NULL,
 `Customer_Id` int DEFAULT NULL,
 `Product_Id` int DEFAULT NULL,
 `Rating` int DEFAULT NULL,
 PRIMARY KEY (`id`),
 KEY `usersdata_index` (`Customer_Id`),
 KEY `productsdata_index` (`Product_Id`)
) ENGINE=InnoDB AUTO_INCREMENT=48 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `Products` (
 `Product_Id` int NOT NULL AUTO_INCREMENT,
 `Supplier_Id` int NOT NULL,
 `Product_Name` varchar(100) NOT NULL,
 `Product_Category` varchar(100) NOT NULL,
 `Color` varchar(50) DEFAULT NULL,
 `Standard_Cost` int NOT NULL,
 `Offer_Price` int NOT NULL,
 `Supplier_level` int NOT NULL,
 `Reorder_Level` int NOT NULL,
 `Minimum_Stock_Level` int NOT NULL,
 `Available_Stock` int NOT NULL,
 PRIMARY KEY (`Product_Id`),
 UNIQUE KEY `Id_UNIQUE` (`Product_Id`),
 KEY `FK_Supplier_id_idx` (`Supplier_Id`),
 CONSTRAINT `FK_Supplier_id` FOREIGN KEY (`Supplier_Id`) REFERENCES `Suppliers` (`Supplier_Id`) ) ENGINE=InnoDB
AUTO_INCREMENT=109 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `Products_not_selling_well` (
 `Product_Id` int NOT NULL DEFAULT '0',
 `Product_Name` varchar(100) NOT NULL,
 `Source` varchar(45) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `Recom_Ratings` (
 `Customer_Id` int NOT NULL,
 `Product_Id` int NOT NULL,
 `Rating` int DEFAULT NULL,
 PRIMARY KEY (`Customer_Id`,`Product_Id`),
 KEY `usersratings_index` (`Customer_Id`),
 KEY `itemsratings_index` (`Product_Id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `Reorder` (
 `Product_Id` varchar(45) DEFAULT NULL,
 `Supplier_level` int DEFAULT NULL,
 `Minimum_Stock_Level` int NOT NULL,
 `Reorder_Level` int DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `Similar_Customers` (
 `Customer_Id` int NOT NULL,
 `score` double DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `Similar_Customers_Ranking` (
 `Rank` int NOT NULL AUTO_INCREMENT,
 `Customer_Id` int DEFAULT NULL,
 PRIMARY KEY (`Rank`)
) ENGINE=InnoDB AUTO_INCREMENT=32 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `Suppliers` (
 `Supplier_Id` int NOT NULL AUTO_INCREMENT,
 `Supplier` varchar(100) NOT NULL,
 `First_Name` varchar(45) NOT NULL,
 `Last_Name` varchar(45) NOT NULL,
 `Job_Title` varchar(45) DEFAULT NULL,
 `Account_Number` varchar(15) NOT NULL,
 `Email` varchar(45) NOT NULL,
 `Address` varchar(45) DEFAULT NULL,
 `City` varchar(45) DEFAULT NULL,
 `State` varchar(45) DEFAULT NULL,
 `Zip_Code` varchar(6) DEFAULT NULL,
 `Phone` varchar(10) NOT NULL,
 `Fax` varchar(45) DEFAULT NULL,
 PRIMARY KEY (`Supplier_Id`),
 UNIQUE KEY `Id_UNIQUE` (`Supplier_Id`),
 UNIQUE KEY `Account_Number_UNIQUE` (`Account_Number`),
 UNIQUE KEY `Email_UNIQUE` (`Email`),
 UNIQUE KEY `Phone_UNIQUE` (`Phone`)
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
) ENGINE=InnoDB AUTO_INCREMENT=21 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `Top_customer` (
 `Order_Id` int NOT NULL DEFAULT '0',
 `Customer_Id` int NOT NULL,
 `Order_Date` varchar(25) NOT NULL,
 `order_year` varchar(10) DEFAULT NULL,
 `order_month` varchar(2) DEFAULT NULL,
 `Order_amount` float NOT NULL,
```

```
 `order_rank` bigint DEFAULT NULL,
 `Current_month` varchar(2) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```
CREATE TABLE `Wishlist` (
 `Customer_Id` int NOT NULL,
 `Product_Id` int NOT NULL,
 `Product_Name` varchar(45) NOT NULL,
 `Product_Category` varchar(45) NOT NULL,
 `Standard_Cost` int DEFAULT NULL,
 `Quantity` int DEFAULT NULL,
 KEY `FK_PID_idx` (`Product_Id`),
 KEY `Cust_Id_FK_idx` (`Customer_Id`),
 KEY `Cust_Id_INDEX` (`Customer_Id`),
 KEY `Custo_Id_FK_idx` (`Customer_Id`),
 CONSTRAINT `FK_PID` FOREIGN KEY (`Product_Id`) REFERENCES `All_Products` (`Product_Id`) ) ENGINE=InnoDB DEFAULT
CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

All Index Creation Scripts for your MySQL Tables

```
create index usersratings_index on Recom_Ratings (Customer_Id);
create index itemsratings_index on Recom_Ratings (Product_Id); and the above PK defined columns
```

All Create View Scripts for your MySQL Tables

```
CREATE
 ALGORITHM = UNDEFINED
 DEFINER = `eswara82`@`%`
 SQL SECURITY DEFINER
VIEW `Most_Highly_wished_products_in_every_category_Report` AS
 SELECT
 `Wishlist`.`Product_Id` AS `Product_Id`,
 `Wishlist`.`Product_Name` AS `Product_Name`,
 `Wishlist`.`Product_Category` AS `Product_Category`,
 `Wishlist`.`Quantity` AS `Quantity`
 FROM
 `Wishlist`
 WHERE
 (`Wishlist`.`Product_Category` = 'Accessories')
 UNION SELECT
 `Wishlist`.`Product_Id` AS `Product_Id`,
 `Wishlist`.`Product_Name` AS `Product_Name`,
 `Wishlist`.`Product_Category` AS `Product_Category`,
 `Wishlist`.`Quantity` AS `Quantity`
 FROM
 `Wishlist`
 WHERE
 (`Wishlist`.`Product_Category` = 'Bikes')
 UNION SELECT
 `Wishlist`.`Product_Id` AS `Product_Id`,
 `Wishlist`.`Product_Name` AS `Product_Name`,
 `Wishlist`.`Product_Category` AS `Product_Category`,
 `Wishlist`.`Quantity` AS `Quantity`
 FROM
 `Wishlist`
 WHERE
 (`Wishlist`.`Product_Category` = 'Components')
 UNION SELECT
 `Wishlist`.`Product_Id` AS `Product_Id`,
 `Wishlist`.`Product_Name` AS `Product_Name`,
 `Wishlist`.`Product_Category` AS `Product_Category`,
 `Wishlist`.`Quantity` AS `Quantity`
```

```
FROM
`Wishlist`
WHERE
(`Wishlist`.`Product_Category` = 'Horror')
UNION SELECT
`Wishlist`.`Product_Id` AS `Product_Id`,
`Wishlist`.`Product_Name` AS `Product_Name`,
`Wishlist`.`Product_Category` AS `Product_Category`,
`Wishlist`.`Quantity` AS `Quantity`
FROM
`Wishlist`
WHERE
(`Wishlist`.`Product_Category` = 'Electronics')
UNION SELECT
`Wishlist`.`Product_Id` AS `Product_Id`,
`Wishlist`.`Product_Name` AS `Product_Name`,
`Wishlist`.`Product_Category` AS `Product_Category`,
`Wishlist`.`Quantity` AS `Quantity`
FROM
`Wishlist`
WHERE
(`Wishlist`.`Product_Category` = 'Personal Care')
(`Wishlist`.`Product_Category` = 'Personal Care')
UNION SELECT
`Wishlist`.`Product_Id` AS `Product_Id`,
`Wishlist`.`Product_Name` AS `Product_Name`,
`Wishlist`.`Product_Category` AS `Product_Category`,
`Wishlist`.`Quantity` AS `Quantity`
FROM
`Wishlist`
WHERE
(`Wishlist`.`Product_Category` = 'Baked Goods & Mixes')
UNION SELECT
`Wishlist`.`Product_Id` AS `Product_Id`,
`Wishlist`.`Product_Name` AS `Product_Name`,
`Wishlist`.`Product_Category` AS `Product_Category`,
`Wishlist`.`Quantity` AS `Quantity`
FROM
`Wishlist`
WHERE
(`Wishlist`.`Product_Category` = 'Clothing')
UNION SELECT
`Wishlist`.`Product_Id` AS `Product_Id`,
`Wishlist`.`Product_Name` AS `Product_Name`,
`Wishlist`.`Product_Category` AS `Product_Category`,
`Wishlist`.`Quantity` AS `Quantity`
FROM
`Wishlist`
WHERE
(`Wishlist`.`Product_Category` = 'Dried Fruit & Nuts')
ORDER BY `Quantity` DESC
```

```
CREATE
ALGORITHM = UNDEFINED
DEFINER = `eswara82`@`%`
SQL SECURITY DEFINER
VIEW `Products_below_min_stock` AS
SELECT
`Products`.`Product_Name` AS `Product_Name`,
`Products`.`Product_Category` AS `Product_Category`,
`Products`.`Standard_Cost` AS `Standard_Cost`,
`Products`.`Minimum_Stock_Level` AS `Minimum_Stock_Level`,
`Products`.`Available_Stock` AS `Available_Stock`
FROM
`Products`
WHERE
```

```
(`Products`.`Available_Stock` < `Products`.`Minimum_Stock_Level`)
--------------------------------------------------------------------------------------------------------------------------------------------------------

CREATE
 ALGORITHM = UNDEFINED
 DEFINER = `eswara82`@`%`
 SQL SECURITY DEFINER
VIEW `View_OUR_customer_information` AS
 SELECT
 `Customer`.`Customer_Id` AS `Customer_Id`,
 `Customer`.`First_Name` AS `First_Name`,
 `Customer`.`Last_Name` AS `Last_Name`,
 `Customer`.`Email` AS `Email`,
 `Customer`.`Address` AS `Address`,
 `Customer`.`City` AS `City`,
 `Customer`.`State` AS `State`,
 `Customer`.`Zip_Code` AS `Zip_Code`,
 `Customer`.`Phone` AS `Phone`
 FROM
 `Customer`
--------------------------------------------------------------------------------------------------------------------------------------------------------

CREATE
 ALGORITHM = UNDEFINED
 DEFINER = `eswara82`@`%`
 SQL SECURITY DEFINER
VIEW `View_OUR_Product_Information_Inventory` AS
 SELECT
 `Products`.`Product_Name` AS `Product_Name`,
 `Products`.`Product_Category` AS `Product_Category`,
 `Products`.`Standard_Cost` AS `Standard_Cost`,
 `Products`.`Minimum_Stock_Level` AS `Minimum_Stock_Level`,
 `Products`.`Available_Stock` AS `Available_Stock`
 FROM
 `Products`
--------------------------------------------------------------------------------------------------------------------------------------------------------

CREATE
 ALGORITHM = UNDEFINED
 DEFINER = `eswara82`@`%`
 SQL SECURITY DEFINER
VIEW `View_OUR_Supplier_Info` AS
 SELECT
 `Suppliers`.`Supplier_Id` AS `Supplier_Id`,
 `Suppliers`.`Supplier` AS `Supplier`,
 `Suppliers`.`First_Name` AS `First_Name`,
 `Suppliers`.`Last_Name` AS `Last_Name`,
 `Suppliers`.`Job_Title` AS `Job_Title`,
 `Suppliers`.`Account_Number` AS `Account_Number`,
 `Suppliers`.`Email` AS `Email`,
 `Suppliers`.`Address` AS `Address`,
 `Suppliers`.`City` AS `City`,
 `Suppliers`.`State` AS `State`,
`Suppliers`.`State` AS `State`,
 `Suppliers`.`Zip_Code` AS `Zip_Code`,
 `Suppliers`.`Phone` AS `Phone`,
 `Suppliers`.`Fax` AS `Fax`
 FROM
 `Suppliers`
--------------------------------------------------------------------------------------------------------------------------------------------------------

CREATE
 ALGORITHM = UNDEFINED
 DEFINER = `eswara82`@`%`
 SQL SECURITY DEFINER
VIEW `View_Partners_Customer_Info` AS
 SELECT
 `northwind`.`customers`.`id` AS `Customer_Id`,
```

```
`northwind`.`customers`.`first_name` AS `First_Name`,
`northwind`.`customers`.`last_name` AS `Last_Name`,
NULL AS `Account_Number`,
'Northwind' AS `Source`
FROM
`northwind`.`customers`
UNION SELECT
`adventureworks`.`customer`.`CustomerID` AS `Customer_Id`,
'null' AS `First_Name`,
'null' AS `Last_Name`,
`adventureworks`.`customer`.`AccountNumber` AS `Account_Number`,
'adventureworks' AS `Source`
FROM
`adventureworks`.`customer`
UNION SELECT
`sakila`.`customer`.`customer_id` AS `Customer_Id`,
`sakila`.`customer`.`first_name` AS `First_Name`,
`sakila`.`customer`.`last_name` AS `Last_Name`,
'No Account Number available' AS `Account_Number`,
'sakila' AS `Source`
FROM
`sakila`.`customer`
ORDER BY `Customer_Id`
```

```
CREATE
ALGORITHM = UNDEFINED
DEFINER = `eswara82`@`%`
SQL SECURITY DEFINER
VIEW `View_Partners_Product_Info` AS
SELECT
`northwind`.`products`.`id` AS `Product_Id`,
`northwind`.`products`.`product_name` AS `product_name`,
`northwind`.`products`.`description` AS `description`,
'Northwind' AS `Source`
FROM
`northwind`.`products`
UNION SELECT
`adventureworks`.`product`.`ProductID` AS `Product_Id`,
`adventureworks`.`product`.`Name` AS `product_name`,
'No description available' AS `description`,
'adventureworks' AS `Source`
FROM
`adventureworks`.`product`
UNION SELECT
`sakila`.`film`.`film_id` AS `Product_Id`,
`sakila`.`film`.`title` AS `product_name`,
`sakila`.`film`.`description` AS `description`,
'sakila' AS `Source`
FROM
`sakila`.`film`
ORDER BY `Product_Id` , `Source`
```

```
CREATE
ALGORITHM = UNDEFINED
DEFINER = `eswara82`@`%`
SQL SECURITY DEFINER
VIEW `Wished_products_never_purchased` AS
SELECT
`Wishlist`.`Customer_Id` AS `Customer_Id`,
`Wishlist`.`Product_Id` AS `Product_Id`,
`Wishlist`.`Product_Name` AS `Product_Name`,
`Wishlist`.`Product_Category` AS `Product_Category`,
`Wishlist`.`Quantity` AS `Quantity`
FROM
`Wishlist`
```

```
WHERE
`Wishlist`.`Customer_Id` IN (SELECT
`Orders`.`Customer_Id`
FROM
`Orders`)
IS FALSE
```

• **All grant scripts for your MySQL tables and views**

• **A description of the algorithms you used for "suggested products" and more accurate product ratings (Note: these should be PROCEDURE based as much as possible).**

Recommendation system may either produce top-k ranking (list of "best" items) or prediction of ratings. The focus of the result may be generic (everyone receives the same recommendations), demographic (everyone in the same category receives the same recommendations) or personal.

Over here we have concentrated on Personal recommendation where, recommendations are based on the personal behaviour which might be similar to a similar user . The context may rely on the user's current activity or on her/his long-term interests.

1. Collaborative filtering -

• **User-Based Collaborative Filtering -** Building User to user similarity matrix to find out the similar user by calculating the distance between two.

▪ The correlation between a pair of users is computed by comparing their given Product ratings - Averageratings that the Customers have given. As we have our product ratings distributed unevenly when

▪ we compare this to the normal curve, we can normalizing the curve by calculating the Avg(Ratings) and substracting that with product ratings as a part of normalization.

▪ In a simple way we find the correlation between you and all the other users and calculated the scores(distances between one user to all).

2. Then Correlation between users:

There exist several possible measures of correlations - we have used - cosine similarity

3. Calculate the similarity between the one user to all then store the correlated users whose behavior is close to yours into a table called "Similar_Customers"

Procedures:

CALL `eswara82`.`Products_Recommendation_Tables_update`(); This will update all the recommendations related table like "avgratingbyusers", "Recom_Ratings","Similar_Customers" CALL `eswara82`.`Product_Recommendation`(<{IN Cust_Id INT}>);

- From this we will take the similar customers based on the TOP RANK which is associated with the score(Stored in Similar customer Ranking table).
- Displays the TOP 10 customers based on the Customer_ID which is Passed through INPUT parameters
- Then it is joined to the Orders to see what are products ordered and in those TOP 10 products were recommended to the Customer which we passed through the procedure.

Queries:

```
CALL `eswara82`.`Products_Recommendation_Tables_update`();
CALL `eswara82`.`Product_Recommendation`(17);
CALL `eswara82`.`Product_Recommendation`(49);
CALL `eswara82`.`Product_Recommendation`(22);
```

Source Code for Any Database Procedures or Triggers

```
CREATE DEFINER=`eswara82`@`%` PROCEDURE `Add_Our_Customer`(IN F_N VARCHAR(45), IN L_N VARCHAR(45), IN Eml
VARCHAR(45), IN Addrs VARCHAR(45), IN Cty VARCHAR(45), IN Sta VARCHAR(45), IN Zp_Code  VARCHAR(45), IN Phn VARCHAR(10))
BEGIN
	INSERT INTO eswara82.Customer(First_Name, Last_Name, Email, Address, City, State, Zip_Code, Phone)
	VALUES (F_N, L_N, Eml, Addrs, Cty, Sta, Zp_Code, Phn);
END
```

```
CREATE DEFINER=`eswara82`@`%` PROCEDURE `Add_Our_Products`(IN Supp_Id INT, IN Prod_Name VARCHAR(45), IN Pro_Categ
VARCHAR(45), IN Clr VARCHAR(45), IN Std_Cost INT, IN Off_Price INT, IN Supp_Lvl INT, IN  Reord_Lvl INT, IN Min_Stk_Lvl INT, IN
Avl_Stk INT)
BEGIN
	IF Supp_Id IN (SELECT Suppliers.Supplier_Id FROM eswara82.Suppliers) THEN
		INSERT INTO eswara82.Products(Supplier_Id, Product_Name, Product_Category, Color, Standard_Cost,
		Offer_Price, Supplier_level,Reorder_Level, Minimum_Stock_Level, Available_Stock) VALUES (Supp_Id,
		Prod_Name, Pro_Categ, Clr, Std_Cost, Off_Price, Supp_Lvl, Reord_Lvl, Min_Stk_Lvl, Avl_Stk);
		 INSERT INTO eswara82.All_Products(Original_Id, Product_Name, Product_Category, Standard_Cost,
		Available_Stock, `Source`)
		VALUES ((SELECT Product_Id FROM eswara82.Products ORDER BY Product_Id DESC LIMIT 1), Prod_Name,
		Pro_Categ, Std_Cost, Avl_Stk, 'OUR_OWN');
	END IF;
END
-----------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Add_Our_Suppliers`(IN Supp VARCHAR(45), IN F_N VARCHAR(20), IN L_N
VARCHAR(20), IN Jb_Titl VARCHAR(45), IN Acc_No DOUBLE, IN Eml VARCHAR(45), IN Addrs  VARCHAR(45), IN Cty VARCHAR(45), IN
Sta VARCHAR(45), IN Zp_Code VARCHAR(10), IN Phn VARCHAR(10), IN Fx VARCHAR(10))
BEGIN
	INSERT INTO eswara82.Suppliers(Supplier, First_Name, Last_Name, Job_title, Account_Number, Email, Address, City,
	State, Zip_Code, Phone, Fax)
	VALUES(Supp, F_N, L_N, Jb_Titl, Acc_No, Eml, Addrs, Cty, Sta, Zp_Code, Phn, Fx);
END

-----------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Add_to_Favorites`(IN Cust_Id INT, IN Prod_Id INT)
BEGIN
	IF (SELECT All_Products.Available_Stock FROM eswara82.All_Products WHERE Product_Id = Prod_Id) > 0 THEN
		SET @result = 'YES';
	ELSE
		BEGIN
			SET @result = 'NO';
		END;
	END IF;
	IF Cust_Id = (SELECT Favorites.Customer_Id FROM eswara82.Favorites WHERE Customer_Id = Cust_Id AND Product_Id =
	Prod_Id) AND Prod_Id = (SELECT Favorites.Product_Id FROM eswara82.Favorites WHERE  Customer_Id = Cust_Id AND
	Product_Id = Prod_Id) THEN
	BEGIN
	END;
	ELSE
	BEGIN
		IF (Cust_Id IN (SELECT Customer.Customer_Id FROM eswara82.Customer) AND Prod_Id IN (SELECT
		All_Products.Product_Id FROM eswara82.All_Products)) THEN
		INSERT INTO eswara82.Favorites (Customer_Id, Product_Id, Product_Name, Product_Category, Standard_Cost,
		Stock_Availability)
		VALUES(Cust_Id, Prod_Id, (SELECT All_Products.Product_Name FROM All_Products WHERE Product_Id =
		Prod_Id),
		(SELECT All_Products.Product_Category FROM All_Products WHERE Product_Id = Prod_Id),
		(SELECT All_Products.Standard_Cost FROM All_Products WHERE Product_Id = Prod_Id), @result);
		END IF;
	END;
	END IF;
END
-----------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Adding_to_Cart`(IN Cust_Id INT, In Prod_Id INt, IN Qty INT)
BEGIN
	IF (Qty > 0 AND Qty < 11) THEN
	IF (Cust_Id IN (SELECT Customer.Customer_Id FROM eswara82.Customer) AND Prod_Id IN (SELECT
	All_Products.Product_Id FROM eswara82.All_Products)) THEN
	IF (SELECT All_Products.`Source` FROM eswara82.All_Products WHERE Product_Id = Prod_Id) = 'OUR_OWN' THEN
```

```
IF ((SELECT All_Products.Available_Stock FROM eswara82.All_Products WHERE Product_Id = Prod_Id) >= Qty) THEN
IF Cust_Id = (SELECT Cart.Customer_Id FROM eswara82.Cart WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id)
AND Prod_Id = (SELECT Cart.Product_Id FROM eswara82.Cart WHERE  Customer_Id = Cust_Id AND Product_Id = Prod_Id)
THEN
UPDATE eswara82.Cart
SET Quantity = Quantity + Qty
WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id;
UPDATE eswara82.Cart
SET Total_Price = Standard_Cost * Quantity
WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id;
WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id;
ELSE
BEGIN
IF (Cust_Id IN (SELECT Customer.Customer_Id FROM eswara82.Customer) AND Prod_Id IN (SELECT
All_Products.Product_Id FROM eswara82.All_Products)) THEN INSERT INTO eswara82.Cart(Customer_Id, Product_Id,
Product_Name, Product_Category, Standard_Cost, Quantity, Total_Price)
VALUES(Cust_Id, Prod_Id, (SELECT All_Products.Product_Name FROM All_Products WHERE Product_Id = Prod_Id),
(SELECT All_Products.Product_Category FROM All_Products WHERE Product_Id = Prod_Id),
(SELECT All_Products.Standard_Cost FROM All_Products WHERE Product_Id = Prod_Id), Qty,
(SELECT All_Products.Standard_Cost * Qty FROM All_Products WHERE Product_Id = Prod_Id));
END IF;
END;
END IF;
UPDATE eswara82.All_Products
SET Available_Stock = Available_Stock - Qty
WHERE Product_Id = Prod_Id;
UPDATE eswara82.Products
SET Available_Stock = Available_Stock - Qty
WHERE Product_Id = (SELECT All_Products.Original_Id FROM All_Products WHERE Product_Id = Prod_Id);
END IF;
ELSE
BEGIN
IF Cust_Id = (SELECT Cart.Customer_Id FROM eswara82.Cart WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id)
AND Prod_Id = (SELECT Cart.Product_Id FROM eswara82.Cart WHERE  Customer_Id = Cust_Id AND Product_Id = Prod_Id)
THEN
UPDATE eswara82.Cart
SET Quantity = Quantity + Qty
WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id;
UPDATE eswara82.Cart
SET Total_Price = Standard_Cost * Quantity
WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id;
ELSE
BEGIN
IF (Cust_Id IN (SELECT Customer.Customer_Id FROM eswara82.Customer) AND Prod_Id IN (SELECT
All_Products.Product_Id FROM eswara82.All_Products)) THEN INSERT INTO eswara82.Cart(Customer_Id, Product_Id,
Product_Name, Product_Category, Standard_Cost, Quantity, Total_Price)
VALUES(Cust_Id, Prod_Id, (SELECT All_Products.Product_Name FROM All_Products WHERE Product_Id = Prod_Id),
(SELECT All_Products.Product_Category FROM All_Products WHERE Product_Id = Prod_Id),
(SELECT All_Products.Standard_Cost FROM All_Products WHERE Product_Id = Prod_Id), Qty,
(SELECT All_Products.Standard_Cost * Qty FROM All_Products WHERE Product_Id = Prod_Id));
END IF;
END;
END IF;
UPDATE eswara82.All_Products
SET Available_Stock = Available_Stock - Qty
WHERE Product_Id = Prod_Id;
UPDATE eswara82.Products
SET Available_Stock = Available_Stock - Qty
WHERE Product_Id = (SELECT All_Products.Original_Id FROM All_Products WHERE Product_Id = Prod_Id);
END;
END IF;
END IF;
```

```
        END IF;
END
-----------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Adv_Report_Top_customers_per_Month`()
BEGIN
        TRUNCATE TABLE Top_customer;
        SET @order_rank = 0;
        INSERT INTO Top_customer (SELECT Order_Id, Customer_Id, Order_Date,
        substring(Order_Date, 7, 10) AS order_year,
        substring(Order_Date, 1, 2) AS order_month,
         Checkout_Price AS Order_amount,
         @order_rank := IF(@current_month = substring(Order_Date, 1, 2),@order_rank + 1, 1) AS order_rank,
         @current_month := substring(Order_Date, 1, 2) AS Current_month
        FROM Orders
        ORDER BY order_year, order_month, Order_amount DESC);
        /*we now pull the top 3 orders out of every group*/
        SELECT DISTINCT Top_customer.Customer_Id , Top_customer.Order_Id, Top_customer.Order_Date,
        Top_customer.Order_amount,Top_customer.order_rank,Orders.Product_Category FROM Top_customer
        JOIN Orders ON Top_customer.Order_Id = Orders.Order_Id
        WHERE order_rank <= 3;
        /*SELECT Customer_Id, Order_Id, Order_Date, Order_amount,order_rank
        FROM Top_customer
        WHERE order_rank <= 3;*/
END
-----------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Cart_to_Orders`(IN Cust_Id INT, IN Prod_Id INT)
BEGIN
        IF Cust_Id IN (SELECT Cart.Customer_Id From eswara82.Cart) AND Prod_Id IN (SELECT Cart.Product_Id From
        eswara82.Cart) THEN
        INSERT INTO eswara82.Orders(Customer_Id, Product_Id, Product_Name, Product_Category, Standard_Cost, Quantity,
        Total_Price, Checkout_Price, `Source`, Order_Date, Order_Day, Delivery_date,  Delivery_day)
        VALUES((SELECT Cart.Customer_Id FROM Cart WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id),
         (SELECT Cart.Product_Id FROM Cart WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id),
         (SELECT Cart.Product_Name FROM Cart WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id),
         (SELECT Cart.Product_Category FROM Cart WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id),
         (SELECT Cart.Standard_Cost FROM Cart WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id),
         (SELECT Cart.Quantity FROM Cart WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id),
         (SELECT Cart.Total_Price FROM Cart WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id),
         (SELECT Cart.Total_Price+(Cart.Total_Price*(1.8/100)) FROM Cart WHERE Customer_Id = Cust_Id AND Product_Id =
        Prod_Id),
         (SELECT All_Products.`Source` FROM All_Products WHERE Product_Id = Prod_Id),
         DATE_FORMAT(current_date(), "%m-%d-%Y"), dayname(current_date()),
         DATE_FORMAT(date_add(current_date(), interval 4 day), "%m-%d-%Y"),
         dayname(date_add(current_date(), interval 4 day)));
        DELETE FROM eswara82.Cart WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id;
        DELETE FROM eswara82.Cart WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id;
         END IF;
END
-----------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Cart_to_Wishlist`(IN Cust_Id INT, IN Prod_Id INT)
BEGIN
        IF Cust_Id IN (SELECT Cart.Customer_Id From eswara82.Cart WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id)
        AND Prod_Id IN (SELECT Cart.Product_Id From eswara82.Cart WHERE Customer_Id = Cust_Id  AND Product_Id = Prod_Id)
        THEN
        IF Cust_Id NOT IN (SELECT Wishlist.Customer_Id From eswara82.Wishlist WHERE Customer_Id = Cust_Id AND Product_Id
        = Prod_Id) AND Prod_Id NOT IN (SELECT Wishlist.Product_Id From eswara82.Wishlist  WHERE Customer_Id = Cust_Id
        AND Product_Id = Prod_Id) THEN
        INSERT INTO eswara82.Wishlist(Customer_Id,Product_Id,Product_Name,Product_Category,Standard_Cost,Quantity)
        SELECT Cart.Customer_Id,Cart.Product_Id,Cart.Product_Name,Cart.Product_Category,Cart.Standard_Cost,Cart.Quantity
        FROM Cart
```

```
        WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id;
        END IF;
         DELETE FROM eswara82.Cart WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id;
        END IF;
END
-----------------------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Customer_not_active`()
BEGIN
        TRUNCATE eswara82.Customer_Id_not_active;
        INSERT INTO Customer_Id_not_active(Customer_Id)
        SELECT Customer.Customer_Id FROM Customer
        WHERE Customer.Customer_Id NOT IN (SELECT DISTINCT Cart.Customer_Id FROM Cart
        LEFT OUTER JOIN Wishlist ON Cart.Customer_Id = Wishlist.Customer_Id
        UNION ALL
        SELECT DISTINCT Wishlist.Customer_Id FROM Wishlist
        LEFT OUTER JOIN Cart ON Wishlist.Customer_Id = Cart.Customer_Id
         UNION ALL
         SELECT DISTINCT Orders.Customer_Id FROM Orders
         LEFT OUTER JOIN Cart ON Orders.Customer_Id = Cart.Customer_Id);
         SELECT * FROM Customer_Id_not_active;
END
-----------------------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Direct_Orders`(IN Cust_Id INT, IN Prod_Id INT, IN Qty INT)
BEGIN
        CALL eswara82.Add_to_Cart(Cust_Id, Prod_Id, Qty);
        CALL eswara82.Cart_to_Orders(Cust_Id, Prod_Id);
END
-----------------------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Display_Recom`(IN start_limit INT)
BEGIN
        INSERT INTO eswara82.Cust_Table_recom (Customer_Id) (SELECT Customer_Id FROM
        eswara82.Similar_Customers_Ranking LIMIT start_limit, 5);
         SELECT Product_Id,Product_Name AS Recommended_Products FROM eswara82.Cust_Table_recom
        RIGHT OUTER JOIN Orders ON Cust_Table_recom.Customer_Id = Orders.Customer_Id
        WHERE Cust_Table_recom.Customer_Id IS NOT NULL LIMIT 10;
END
-----------------------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Modify_Our_Customers`(IN Cust_Id INT, IN F_N VARCHAR(45), IN L_N
VARCHAR(45), IN EmlVARCHAR(45), IN Addrs VARCHAR(45), IN Cty VARCHAR(45), IN Sta VARCHAR(45),  IN Zp_Code VARCHAR(45),
IN Phn VARCHAR(10))
BEGIN
        IF F_N != "" THEN
        UPDATE eswara82.Customer
        SET First_Name = F_N
        WHERE Customer_Id = Cust_Id;
        ELSE
        BEGIN
         END;
        END IF;
        IF L_N != "" THEN
        UPDATE eswara82.Customer
        SET Last_Name = L_N
        WHERE Customer_Id = Cust_Id;
        ELSE
        BEGIN
         END;
        END IF;
        IF Eml != "" THEN
        UPDATE eswara82.Customer
```

```
        SET Email = Eml
        WHERE Customer_Id = Cust_Id;
        ELSE
        BEGIN
         END;
        END IF;
        IF Addrs != "" THEN
        UPDATE eswara82.Customer
        SET Address = Addrs
        WHERE Customer_Id = Cust_Id;
        ELSE
        BEGIN
         END;
        END IF;
        IF Cty != "" THEN
        UPDATE eswara82.Customer
        SET City = Cty
        WHERE Customer_Id = Cust_Id;
        ELSE
        BEGIN
         END;
        END IF;
        IF Sta != "" THEN
        UPDATE eswara82.Customer
        SET State = Sta
        ELSE
         END;
        SET State = Sta
        WHERE Customer_Id = Cust_Id; BEGIN
        END IF;
        IF Zp_Code != "" THEN
        UPDATE eswara82.Customer
        SET Zip_Code = Zp_Code
        WHERE Customer_Id = Cust_Id;
        ELSE
        BEGIN
         END;
        END IF;
        IF Phn != "" THEN
        UPDATE eswara82.Customer
        SET Phone = Phn
        WHERE Customer_Id = Cust_Id;
        ELSE
        BEGIN
         END;
        END IF;
END
-----------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Modify_Our_Products`(IN Prod_Id INT, IN Supp_Id INT, IN Prod_Name
VARCHAR(45), IN Prod_Categ VARCHAR(45), IN Clr VARCHAR(45), IN Std_Cost INT, IN Off_Price INT, IN  Supp_Lvl INT, IN Reord_Lvl
INT, IN Min_Stk_Lvl INT, IN Avl_Stk INT)
BEGIN
        IF Supp_Id != "" THEN
        IF Supp_Id IN (SELECT Suppliers.Supplier_Id FROM eswara82.Suppliers) THEN
        UPDATE eswara82.Products
        SET Supplier_Id = Supp_Id
        WHERE Product_Id = Prod_Id;
        END IF;
        ELSE
        BEGIN
        END;
```

```
END IF;
 IF Prod_Name != "" THEN
UPDATE eswara82.Products
SET Product_Name = Prod_Name
WHERE Product_Id = Prod_Id;
 UPDATE eswara82.All_Products
 SET Product_Name = Prod_Name
 WHERE Original_Id = Prod_Id AND `Source` = 'OUR_OWN';
ELSE
BEGIN
 END;
END IF;
IF Prod_Categ != "" THEN
UPDATE eswara82.Products
SET Product_Category = Prod_Categ
WHERE Product_Id = Prod_Id;
 UPDATE eswara82.All_Products
 SET Product_Category = Prod_Categ
 WHERE Original_Id = Prod_Id AND `Source` = 'OUR_OWN';
ELSE
BEGIN
 END;
END IF;
IF Clr != "" THEN
UPDATE eswara82.Products
SET Color = Clr
WHERE Product_Id = Prod_Id;
ELSE
BEGIN
 END;
END IF;
IF Std_Cost != "" THEN
UPDATE eswara82.Products
SET Standard_Cost = Std_Cost
WHERE Product_Id = Prod_Id;
UPDATE eswara82.All_Products
SET Standard_Cost = Std_Cost
WHERE Original_Id = Prod_Id AND `Source` = 'OUR_OWN';
ELSE
BEGIN
END;
END IF;
IF Off_Price != "" THEN
UPDATE eswara82.Products
SET Offer_Price = Off_Price
WHERE Product_Id = Prod_Id;
ELSE
BEGIN
END;
END IF;
IF Supp_Lvl != "" THEN
UPDATE eswara82.Products
SET Supplier_Level = Supp_Lvl
WHERE Product_Id = Prod_Id;
ELSE
BEGIN
END;
END IF;
IF Reord_Lvl != "" THEN
IF Reord_Lvl != "" THEN
UPDATE eswara82.Products
SET Reorder_Level = Reord_Lvl
```

```
        WHERE Product_Id = Prod_Id;
        ELSE
        BEGIN
        END;
        END IF;
        IF Min_Stk_Lvl != "" THEN
        UPDATE eswara82.Products
        SET Minimum_Stock_Level = Min_Stk_Lvl
        WHERE Product_Id = Prod_Id;
        ELSE
        BEGIN
        END;
        END IF;
        IF Avl_Stk != "" THEN
        UPDATE eswara82.Products
        SET Available_Stock = Avl_Stk
        WHERE Product_Id = Prod_Id;
        UPDATE eswara82.All_Products
        SET Available_Stock = Avl_Stk
        WHERE Original_Id = Prod_Id AND `Source` = 'OUR_OWN';
        ELSE
        BEGIN
        END;
        END IF;
END
-----------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Modify_Our_Suppliers`(IN Supp_Id INT, IN Supp VARCHAR(45), IN F_N
VARCHAR(20), IN L_N VARCHAR(20), IN Jb_Titl VARCHAR(45), IN Acc_No VARCHAR(15), IN Eml  VARCHAR(45), IN Addrs
VARCHAR(45), IN Cty VARCHAR(45), IN Sta VARCHAR(45), IN Zp_Code VARCHAR(10), IN Phn VARCHAR(10), IN Fx VARCHAR(10))
BEGIN
        IF Supp != "" THEN
        UPDATE eswara82.Suppliers
        SET Supplier =Supp
        WHERE Supplier_Id = Supp_Id;
        ELSE
        BEGIN
         END;
        END IF;
        IF F_N != "" THEN
        UPDATE eswara82.Suppliers
        SET First_Name = F_N
        WHERE Supplier_Id = Supp_Id;
        ELSE
        BEGIN
         END;
        END IF;
        IF L_N != "" THEN
        UPDATE eswara82.Suppliers
        SET Last_Name = L_N
        WHERE Supplier_Id = Supp_Id;
        ELSE
        BEGIN
         END;
        END IF;
        IF Jb_Titl != "" THEN
        UPDATE eswara82.Suppliers
        SET Job_Title = Jb_Titl
        WHERE Supplier_Id = Supp_Id;
        ELSE
        BEGIN
         END;
```

```
END IF;
IF Acc_No != "" THEN
UPDATE eswara82.Suppliers
SET Account_Number = Acc_No
WHERE Supplier_Id = Supp_Id;
ELSE
BEGIN
 END;
END IF;
IF Eml != "" THEN
UPDATE eswara82.Suppliers
SET Email = Eml
WHERE Supplier_Id = Supp_Id;
ELSE
BEGIN
 END;
END IF;
IF Addrs != "" THEN
UPDATE eswara82.Suppliers
SET Address = Addrs
WHERE Supplier_Id = Supp_Id;
ELSE
BEGIN
 END;
END IF;
IF Cty != "" THEN
UPDATE eswara82.Suppliers
SET City = Cty
WHERE Supplier_Id = Supp_Id;
ELSE
BEGIN
 END;
END;
END IF;
IF Sta != "" THEN
UPDATE eswara82.Suppliers
SET State = Sta
WHERE Supplier_Id = Supp_Id;
ELSE
BEGIN
 END;
END IF;
IF Zp_Code != "" THEN
UPDATE eswara82.Suppliers
SET Zip_Code = Zp_Code
WHERE Supplier_Id = Supp_Id;
ELSE
BEGIN
 END;
END IF;
IF Phn != "" THEN
UPDATE eswara82.Suppliers
SET Phone = Phn
WHERE Supplier_Id = Supp_Id;
ELSE
BEGIN
 END;
END IF;
IF Fx != "" THEN
UPDATE eswara82.Suppliers
SET Fax = Fx
WHERE Supplier_Id = Supp_Id;
```

```
        ELSE
        BEGIN
         END;
        END IF;
END
-----------------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Most_Highly_wished_products_in_every_category`(IN V_Product_Category
VARCHAR(100))
BEGIN
        SELECT Product_Id,Product_Name,Product_Category,Quantity FROM Wishlist
        WHERE Product_Category = V_Product_Category
        ORDER BY Quantity DESC ;
END
-----------------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Products_Recommendation_Tables_update`()
BEGIN
        /* This will upadte avgratingbyusers,Recom_Ratings,Similar_Customers/*
        /*Update avgratingbyusers table to get the upaded avg ratings*/
        TRUNCATE TABLE avgratingbyusers;
        INSERT INTO avgratingbyusers (
        SELECT count(Customer_Id) AS nbusers, avgrating
        FROM (SELECT round(AVG(Rating),1) AS avgrating, Customer_Id
         FROM Product_Ratings
         GROUP BY Customer_Id
        ) AS avgratingbyusers
        GROUP BY avgrating
        ORDER BY avgrating DESC);
        /* Update Recom_Ratings table- to get the updated values -
         Trying to centralize the curve of normal distribution of the average ratings - this is called "Normalization"*/
        TRUNCATE TABLE Recom_Ratings;
        INSERT INTO Recom_Ratings (Customer_Id,Product_Id,Rating)
        (select distinct Product_Ratings.Customer_Id, Product_Ratings.Product_Id,
         Product_Ratings.Rating - avgratingbyusers.avgrating
         from Product_Ratings,avgratingbyusers,
         (select Customer_Id,AVG(Rating)
         from Product_Ratings
         group by Customer_Id
         ) as avgratingbyusers
         where Product_Ratings.Customer_Id=avgratingbyusers.Customer_Id
         );
        /* User-based collaborative filtering --------Finding correlation with cosine correlation and dumping similar_customers
        into Similar_Customers table */ TRUNCATE TABLE Similar_Customers;
        INSERT INTO Similar_Customers(SELECT distances.Customer_Id AS Customer_Id, dist/(sqrt(my.norm)*sqrt(users.norm))
        AS score
        FROM (SELECT Recom_Ratings.Customer_Id, sum((Recom_Ratings.Rating)*(Product_Ratings.Rating)) AS dist
         FROM Recom_Ratings, Product_Ratings
         WHERE Product_Ratings.Product_Id = Recom_Ratings.Product_Id
         GROUP BY Customer_Id
        ) AS distances,
        (SELECT Recom_Ratings.Customer_Id, sum((Rating)*(Rating)) AS norm
         FROM Recom_Ratings
         GROUP BY Customer_Id
        ) AS users,
        (SELECT sum((rating)*(rating)) AS norm
         FROM Product_Ratings
        ) AS my
        WHERE users.Customer_Id = distances.Customer_Id ORDER BY score DESC LIMIT 30);
END
-----------------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Product_Recommendation`(IN Cust_Id INT)
```

```
BEGIN
BEGIN
	CALL `eswara82`.`Products_Recommendation_Tables_update`();
	TRUNCATE eswara82.Cust_Table_recom;
	IF (SELECT `Rank` FROM eswara82.Similar_Customers_Ranking WHERE Customer_Id = Cust_Id) = 1 THEN
	INSERT INTO eswara82.Cust_Table_recom (Customer_Id) (SELECT Customer_Id FROM eswara82.Similar_Customers
	WHERE Customer_Id = Cust_Id); ELSE IF (SELECT `Rank` FROM eswara82.Similar_Customers_Ranking WHERE
	Customer_Id = Cust_Id) = 2 THEN
	INSERT INTO eswara82.Cust_Table_recom (Customer_Id) (SELECT Customer_Id FROM eswara82.Similar_Customers LIMIT
	1);
	ELSE IF (SELECT `Rank` FROM eswara82.Similar_Customers_Ranking WHERE Customer_Id = Cust_Id) = 3 THEN
	INSERT INTO eswara82.Cust_Table_recom (Customer_Id) (SELECT Customer_Id FROM eswara82.Similar_Customers LIMIT
	2);
	ELSE IF (SELECT `Rank` FROM eswara82.Similar_Customers_Ranking WHERE Customer_Id = Cust_Id) = 4 THEN
	INSERT INTO eswara82.Cust_Table_recom (Customer_Id) (SELECT Customer_Id FROM eswara82.Similar_Customers LIMIT
	3);
	ELSE IF (SELECT `Rank` FROM eswara82.Similar_Customers_Ranking WHERE Customer_Id = Cust_Id) = 5 THEN
	INSERT INTO eswara82.Cust_Table_recom (Customer_Id) (SELECT Customer_Id FROM eswara82.Similar_Customers LIMIT
	4);
	ELSE IF (SELECT `Rank` FROM eswara82.Similar_Customers_Ranking WHERE Customer_Id = Cust_Id) = 6 THEN
	INSERT INTO eswara82.Cust_Table_recom (Customer_Id) (SELECT Customer_Id FROM eswara82.Similar_Customers LIMIT
	5);
	 ELSE
	 BEGIN
	SET @display = (SELECT `Rank` FROM eswara82.Similar_Customers_Ranking WHERE Customer_Id = Cust_Id);
	SET @displ = (SELECT eswara82.recommend_limiter(@display));
	CALL `eswara82`.`Display_Recom`(@displ);
	 END;
	END IF;
	 END IF;
	 END IF;
	 END IF;
	 END IF;
	 END IF;
END
-----------------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Products_By_Category`(IN V_Category VARCHAR(45))
BEGIN
	SELECT Products.Product_Id As `Product_Id`,
	Products.Product_Name AS `Product_Name`,
	Products.Product_Category AS `Category`,
	'OUR_OWN' AS `Source`
	FROM eswara82.Products
	WHERE `Product_Category` = V_Category
	UNION
	SELECT DISTINCT product.ProductID, product.`Name` AS Product_Name, productcategory.`Name` AS Category,
	'adventureworks' AS `Source`
	FROM `adventureworks`.`product`
	JOIN adventureworks.productsubcategory ON `adventureworks`.`product`.ProductSubcategoryID =
	adventureworks.productsubcategory.ProductSubcategoryID JOIN adventureworks.productcategory ON
	product.ProductSubcategoryID = productsubcategory.ProductSubcategoryID
	WHERE productcategory.`Name` = V_Category
	UNION
	SELECT sakila.film.film_id AS Product_Id,
	sakila.film.title AS Product_Name,
	sakila.category.`name` AS Category,
	'sakila' AS `Source`
	FROM sakila.film
	JOIN sakila.film_category ON film.film_id = film_category.film_id
	JOIN sakila.category ON film_category.category_id = category.category_id
	WHERE `name` = V_Category
```

```
	UNION
	SELECT
	northwind.products.id AS ProductID,
	northwind.products.product_name AS Product_Name,
	northwind.products.category AS Category,
	'northwind' AS `Source`
	FROM northwind.products
	WHERE northwind.products.category = V_Category
	/*ORDER BY `Category`*/;
END
-----------------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Products_not_selling_well`()
BEGIN
	TRUNCATE TABLE Products_not_selling_well;
	INSERT INTO Products_not_selling_well(
	SELECT DISTINCT Product_Id,Product_Name,`Source` FROM All_Products WHERE Product_Id NOT IN (SELECT
	Orders.Product_Id FROM Orders)  AND `Source` = 'OUR_OWN');
	 SELECT * FROM Products_not_selling_well;
END
-----------------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Rating_for_Product`(IN Ord_Id INT, IN Rate INT)
BEGIN
	IF Rate > 0 AND Rate < 6 THEN
	IF Ord_Id IN (SELECT Orders.Order_Id FROM eswara82.Orders) THEN
	IF Ord_Id IN (SELECT Product_Ratings.Order_Id FROM eswara82.Product_Ratings) THEN
	UPDATE eswara82.Product_Ratings
	SET Rating = Rate
	WHERE Order_Id = Ord_Id;
	ELSE
	INSERT INTO eswara82.Product_Ratings(Order_Id,Customer_Id,Product_Id,Rating)
	VALUES(Ord_Id,(SELECT Orders.Customer_Id FROM Orders WHERE Order_Id = Ord_Id), (SELECT Orders.Product_Id FROM
	Orders WHERE Order_Id = Ord_Id), Rate); END IF;
	END IF;
	END IF;
END
-----------------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Remove_from_Cart`(IN Cust_Id INT, IN Prod_Id INT)
BEGIN
	UPDATE eswara82.All_Products
	SET Available_Stock = Available_Stock + (SELECT Quantity FROM eswara82.Cart WHERE Customer_Id = Cust_Id AND
	Product_Id = Prod_Id)
	WHERE Product_Id = Prod_Id;
	WHERE Product_Id = Prod_Id;
	 UPDATE eswara82.Products
	SET Available_Stock = Available_Stock + (SELECT Quantity FROM eswara82.Cart WHERE Customer_Id = Cust_Id AND
	Product_Id = Prod_Id)
	WHERE Product_Id = (SELECT All_Products.Original_Id FROM All_Products WHERE Product_Id = Prod_Id);
	 DELETE FROM eswara82.Cart WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id;
END
-----------------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Remove_from_Wishlist`(IN Cust_Id INT, IN Prod_Id INT)
BEGIN
	UPDATE eswara82.All_Products
	SET Available_Stock = Available_Stock + (SELECT Quantity FROM eswara82.Wishlist WHERE Customer_Id = Cust_Id AND
	Product_Id = Prod_Id)
	WHERE Product_Id = Prod_Id;
	 UPDATE eswara82.Products
	SET Available_Stock = Available_Stock + (SELECT Quantity FROM eswara82.Wishlist WHERE Customer_Id = Cust_Id AND
	Product_Id = Prod_Id)
```

```
        WHERE Product_Id = (SELECT All_Products.Original_Id FROM All_Products WHERE Product_Id = Prod_Id);
        DELETE FROM eswara82.Wishlist WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id;
END
-----------------------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Remove_Our_Customer`(IN Cust_Id INT)
BEGIN
        DELETE FROM eswara82.Customer WHERE Customer_Id = Cust_Id;
END
-----------------------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Remove_Our_Product`(IN Prod_Id INT)
BEGIN
        DELETE FROM eswara82.Products WHERE Product_Id = Prod_Id;
        DELETE FROM eswara82.All_Products WHERE Original_Id = Prod_Id AND `Source` = 'OUR_OWN';
END
-----------------------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Remove_Our_Supplier`(IN Supp_Id INT)
BEGIN
        DELETE FROM eswara82.Suppliers WHERE Supplier_Id = Supp_Id;
END
-----------------------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Restocking`()
BEGIN
        TRUNCATE TABLE eswara82.Reorder;
        INSERT INTO eswara82.Reorder(Product_Id,Reorder_Level,Supplier_level,Minimum_Stock_Level)
        SELECT Product_Id,Reorder_Level,Supplier_level,Minimum_Stock_Level FROM eswara82.Products
        WHERE Supplier_level < Minimum_Stock_Level;
        SELECT * FROM eswara82.Reorder;
END
-----------------------------------------------------------------------------------------------------------------------------------------------------------

CREATE DEFINER=`eswara82`@`%` PROCEDURE `Wishlist_to_Cart`(IN Cust_Id INT, IN Prod_Id INT)
BEGIN
        IF Cust_Id NOT IN (SELECT Cart.Customer_Id From eswara82.Cart WHERE Customer_Id = Cust_Id AND Product_Id =
        Prod_Id) AND Prod_Id NOT IN (SELECT Cart.Product_Id From eswara82.Cart WHERE Customer_Id =  Cust_Id AND
        Product_Id = Prod_Id) THEN
        INSERT INTO
        eswara82.Cart(Customer_Id,Product_Id,Product_Name,Product_Category,Standard_Cost,Quantity,Total_Price)
        SELECT
        Wishlist.Customer_Id,Wishlist.Product_Id,Wishlist.Product_Name,Wishlist.Product_Category,Wishlist.Standard_Cost,Wis
        hlist.Quantity,Wishlist.Standard_Cost * Quantity FROM Wishlist
        WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id;
        DELETE FROM eswara82.Wishlist WHERE Customer_Id = Cust_Id AND Product_Id = Prod_Id;
        END IF;
END
```

- **<u>All database statements from your second database that support the basic functionality and reports of the system</u>**

View for Customer Information

```
db.createView("View_Customer_Information", "Customer", [ { $project: { _id:0, "Customer Id": "$Customer_Id","Full_Name": {
$concat: [ "$First_Name", ", ", "$Last_Name" ] }, "Email": "$Email", "Address": {$concat:  ["$Address", ", ", "$City", ", ", "$State", ",
", "$Zip_Code" ] }, "Phone": "$Phone" } } ] )
```

Query: db.View_Customer_Information.find().pretty()

View Partners Customer Information

Query: db.getSiblingDB('sakila').customers.find().pretty()

View for Supplier Information

db.createView("View_Supplier_Information", "Suppliers", [{ $project: { _id:0, "Supplier Id":"$Supplier_Id", "Company":"$Supplier", "Full_Name": { $concat: ["$First_Name", " ", "$Last_Name"] }, "Title":"$Job_Title", "Account Number":"$Account_Number", "Email":"$Email", "Address": {$concat: ["$Address", ", ", "$City", ", ", "$State", ", ", "$Zip_Code"] }, "Phone":"$Phone"}}])
Query: db.View_Supplier_Information.find().pretty()

View for Product Information
db.createView("View_Product_Information", "Products", [{ $project: { _id:0, "Product Id": "$Product_Id", "Name":"$Product_Name", "Category":"$Product_Category", "Color":"$Color", "Price":"$Standard_Cost", "Available Stock":"$Available_Stock" } }])
Query: db.View_Product_Information.find().pretty()

View Partners Product Information
Query: db.getSiblingDB('sakila').films.find().pretty()

View for Product Inventory
db.createView("View_Product_Inventory", "Products", [{ $project: { _id:0, "Product Id": "$Product_Id", "Supplier Id":"$Supplier_Id", "Name":"$Product_Name", "Category":"$Product_Category", "Color":"$Color", "Price":"$Standard_Cost", "Offer Price":"$Offer_Price", "Supplier Level":"$Supplier_level", "Reorder Level":"$Reorder_Level","Minimum Stock Level":"$Minimum_Stock_Level", "Available Stock":"$Available_Stock" } }])
Query: db.View_Product_Inventory.find().pretty()

Browse Products by Category
Query: db.Products.find({"Product_Category" : "Electronics"}).pretty()
Query: db.Products.find({"Product_Category" : "Beverages"}).pretty()

View Products of Minimum Stock Level
Query: db.Products.find({"$where": "this.Available_Stock < this.Minimum_Stock_Level"}).pretty()

- **A comprehensive description of the differences between the MySQL implementation and the other implementation. This should include well articulated pros and cons of each implementation.**

MySQL	MongoDB
Need to define the schema of the table	Do not need to define the schema
MySQL supports JOIN operations	MongoDB does not support JOIN operations
Structured Query Language (SQL)	JavaScript as query language
Table structure format, in this query takes longer time to retrieve data with larger data volume/Multiple joins	Json structure format, this will be easier to work with data more flexibility in Jason format. MongoDB's document model stores related data together, it is often faster to retrieve a single document from MongoDB than to JOIN data across multiple tables in MySQL.
MYSQL is a great choice if we have structured data and need a traditional relational database as per the requirement what we have in this project	Ideal choice if we have unstructured and/or structured data with the potential for rapid growth, but insertion of data and pulling of data is easier in this NoSQL DB
Query will be a bit more lines when compared to Mango DB query for joining multiple tables and retrieving the data	Query can be writing in few lines even though we need join multiple documents for retrieving the data
Only structured type of file can be used or stored in MYSQL DB	MongoDB allows you to store any type of file which can be any size without effecting our stack
SQL procedures are developed using SQL language, multiple lines of code are developed for procedure in order perform the required functionality	Basically, uses JavaScript objects in place of the procedure
Primary key column can be indexed in MYSQL DB	Indexes can be created to improve the performance of searches within MongoDB. Any field in a MongoDB document can be indexed
The schema cannot be changed. The inputs following the given schema are only entered.	NoSQL database. This means that pre-defined structure for the incoming data can be defined and adhered to but also, if required different documents in a collection can have different structures. It has a dynamic schema.
MySQL requires you to define your tables and columns before you can store anything, and every row in a table must have the same columns.	In MongoDB, you do not need to define the schema. Instead, you just drop in documents do not even need to have the same fields.