CSE202-DBMS
Project Report
Retail Store Database Management System

Group Number-46 :

Team members-
Rishav (2020569)
Shivanshu Nayak(2020472)
D Likhith (2019038)
Nitin (2020092)

Problem Statement- Small retail stores having several outlets face a lot of difficulties in managing data related to employees, customers, orders, products, etc. In order to solve that problem, we have designed a database management system using SQL so that they can efficiently manage their data and perform various operations on them to get insights about trending market patterns

Scope of Project- We have designed a retail store database management system using MySQL and JavaFX. It contains several stakeholders represented as entities that are connected to each other using several relationships. The aim of the project is to create a simple and user-friendly application for the retailers with several features like the donation to the Charity Foundation which is connected with customers' orders.

Stakeholders and their rolesCustomer-

Customers -

- They are the key stakeholders of the database
- This database will record customer details, they can interact with different products and have their specific cart for adding wishlist products.

- They will also have access to see their current and old orders.

Employees -

- They are the people who work at the store and manage all the work.
- It contains all the required details of the employees, including their ID, password, name, phone number, and salary.

Charity -

- It contains the details of several charities which are related to different sections of the society, like feeding the babies, and the smile foundation for underprivileged kids.
- A certain amount of money is donated to charity on every purchase of the products by the customers.

Store -

- It is also one of the main stakeholders of databases like customers.
- It contains outlets in several parts of the country.
- It maintains the products and orders. And the store receives the products from the supplier.

Supplier -

- It contains information about all the suppliers who supply the products to the store.

Entities: Customer, Store, Products, Employees, Charity, Supplier, Shopping_Cart, Order.

Primary keys : Customer (Customer_ID) , Products(Product_ID), Store(Store_ID), Employees(Employees_ID), Charity(Charity_ID), Supplier(Supplier_ID), Order(Order_ID).

Weak Entity: Shopping_cart because it doesn't exist without Customer_ID. It depends on the customer itself.

Ternary relation: Product_supplied relation is the ternary relation among Products, Store, and Supplier entities. Because the supplier is supplying the

products to the store and this relationship involves the three strong entities together.

Entity-relationship participation and their type:

- Works relationship between employees and store- employees have total participation, and the store also has total participation. It is a many-to-one relationship.

- Maintains relationship between store and products- both products and store have total participation and it is a many-to-many relationship.

- Interacts relationship between products and customer- both products and customer have partial participation and it is a many-to-many relationship.

- Places relationship between customer and order- both customer and order have partial participation and it is a many-to-one relationship.

- Handles relationship between customer and shopping_cart- both customer and shopping cart have total participation and it is a one-to-one relationship.

- Products_Supplied is a ternary relationship between store, supplier, and products and supplier have partial participation whereas both store and products have total participation. It is a many-to-many relationship.

- Manages is a relationship between order and store- Order has total participation whereas store has partial participation. It is a many-to-one relationship.

- Store and Charity have a ‘contributes’ relationship between them- both charity and store have partial participation and it is a many-to-many relationship.

Relationship Schema:

Product(Product_ID, Name, Category, Cost, Average Rating)

Customer(Customer_ID, Name, Password, {Phone}, Address_ID)

Address(Street,City,State,Pincode)

Contributes(Charity_ID,Store_ID)

Order(Order_ID, Customer_ID,Store_ID, Booking Date, Product_ID, Quantity, Prices, Payment_Mode, Total_cost, Delivery_Date, Address_ID)

Products_Supplied(Supplier_ID,Product_ID,Store_ID,EXP,Availability)

Shopping_Cart(Customer_ID, Product_ID, Quantity)

Maintains(Product_ID,Store_ID)

Store(Store_ID,{Phone}, Address_ID)

Supplier(Supplier_ID, Supplier_name, Address_ID, {Phone})

Employees(Employee_ID, Name, Password, {Phone}, Salary, Store_ID, DOB)

Contains(Customer_ID,Product_ID) Charity(Charity_ID, Name, Address_ID, Account_Number)

SQL Queries:

Arranging the orders in descending order of their cost:

```
SELECT *, RANK() OVER (ORDER BY Total_cost DESC) AS Rank_No FROM Orders;
```

Finding the Employee with the second highest salary:

```
SELECT * FROM Employees WHERE Employee_salary = (SELECT MAX(Employee_salary) FROM Employees WHERE Employee_salary != (SELECT MAX(Employee_salary) FROM Employees));
```

Finding the products sold by each category:

SELECT Category, COUNT(*) FROM (SELECT P.Category, P.Product_ID, O.Quantity FROM Products AS P INNER JOIN Orders AS O ON P.Product_ID = O.Product_ID) AS T GROUP BY Category;

Arranging the stores in descending order of total number of orders made from that store:

SELECT *, RANK() OVER (ORDER BY Store_orders DESC) AS Rank_No FROM (SELECT Store_ID, COUNT(*) AS Store_orders FROM Order s GROUP BY Store_ID) AS T;

Finding the quantity of each product sold:

SELECT Product_ID, Product_name, SUM(Quantity) FROM (SELECT P.Category, P.Product_ID, O.Quantity, P.Product_name FROM Products AS P INNER JOIN Orders AS O ON P.Product_ID = O.Product_ID) AS T GROUP BY Product_ID;

Finding the highest paid employees from each store:

SELECT Employee_name, Store_ID, Employee_salary FROM Employees WHERE Employee_salary IN (SELECT MAX(Employee_salary) FROM Employees GROUP BY Store_ID);

Arranging the contribution to charity by each store in descending order:

SELECT *, RANK() OVER (ORDER BY Total_donation DESC) AS Rank_no FROM (SELECT Store_ID, SUM(Charity_donation) AS Total_donation FROM Orders GROUP BY Store_ID) AS T;

Finding the quantity of total products supplied from each supplier:

SELECT Supplier_name, SUM(Availability) AS Quantity_supplied FROM (SELECT S.Supplier_name, S.Supplier_ID, PS.Availability FROM Supplier AS S

INNER JOIN products_supplied AS PS ON S.Supplier_ID = PS.Supplier_ID) AS T GROUP BY Supplier_ID;

Finding the most expensive products from each category:

SELECT Product_ID, Product_name, Category FROM Products WHERE Cost IN (SELECT MAX(Cost) FROM Products GROUP BY Category);

Finding the value of total purchase by each customer:

SELECT Customer_name, SUM(Total_cost) AS Total_purchase_amount FROM (SELECT C.Customer_name, C.Customer_ID, O.Total_cost FROM Customer AS C INNER JOIN Orders AS O ON O.Customer_ID = C.Customer_ID) AS T GROUP BY Customer_ID;

Triggers Added -

Trigger to update total cost in order which is (cost of one item) * (quantity)
CREATE TRIGGER update_cost
BEFORE INSERT
ON Orders
FOR EACH ROW
SET NEW.Total_cost = NEW.Quantity * (SELECT Cost FROM Products WHERE Product_ID = NEW.Product_ID);

Trigger to update donation to charity which is 2% amount of order cost
CREATE TRIGGER charity_donation BEFORE INSERT ON Orders FOR EACH ROW SET NEW.Charity_donation = NEW.Total_cost*0.02;

Indexing added -

CREATE INDEX salary_index ON Employees (Employee_salary);
CREATE INDEX productCost_index ON Products (Cost, Product_name);
CREATE INDEX cost_index ON Orders (Total_cost);

Views added -

View for Customer: CREATE VIEW V1 AS SELECT Product_name, Category, Cost, Average_rating FROM Products;
View for Admin: CREATE VIEW V2 AS SELECT Customer_ID, Customer_name, Phone_number, Address_ID FROM Customer;

Grants added:

drop user Tony@localhost;
drop user jeffrey@localhost;
flush privileges;

CREATE USER 'jeffrey'@'localhost' IDENTIFIED BY 'password';

GRANT ALL ON Orders TO 'jeffrey'@'localhost';

CREATE USER 'Tony'@'localhost' IDENTIFIED BY 'password';

GRANT ALL ON Products TO 'Tony'@'localhost'; GRANT ALL ON * TO 'root'@'localhost';

Contribution of team members:
It was a complete team effort and everyone contributed equally to this project

D Llkhith - Compilation of the project, ER diagram, defining DDL, Populating data in tables, SQL queries, Report documentation, Backend on SQL, Embedded SQL Queries on JAVA, Diverse Queries.

Rishav - ER diagram, defining DDL, Populating data in tables, Relational Schema, Identification of ternary relationships, Frontend on JavaFX, Views and Grants, Triggers, Debugging Java Functions in frontend, Constructing FXML files.

Shivanshu - ER diagram, defining DDL, Populating data in tables, SQL queries, Major Frontend on Java, Merging all Java functions and files on JavaFX and connecting them with back-end, Embedded Queries in Java, Diverse SQL queries.

Nitin - ER diagram, stakeholders and their roles, defining DDL, Populating data in tables, SQL Queries, Report documentation, Frontend on JavaFX, Views and Grants, Triggers, Indexing and Constructing FXML files.