TECHNICAL REPORT

ON MUSIC STORE DATABASE

Submitted To:

 Dr. Angelina Ziesemer

Submitted By:

Rajpreet Singh

Urvish

Shabnam

Scenario for a music store database

A website of music store that offer its albums to customers and allow them to place and order online, needs database that can hold its information.

A music store offer the customers to listen or preview the songs before they buy. one customer can preview songs as many be want before placing an order music store on website. Music store wants to hold information how many customer order using online feature, what they order, when they order which employee served the customer in handling the queries.

customer's order is furthermore associated with the order\_item table and then album table.

Albums table is connected to the songs table and genre table and artist table. Amusic store want a returns table that have complete information about which ablum is returned by which customer.

So There a music store want a database that will record its whole information about employee, customer, albums, orders, and returns.

Technical Report

We created a database for a website of a mobile store. While creating this database we tried to make tables on all the entities (such as employees, customers, albums, etc.…) and further we have breakdown them into other tables to make them normalize.

While creating this database we tried to capture the information on the basis of ‘Who’,’What’,’How’.

A music store wants to know ‘Who’ is its customer, so to record that data we made ‘customers’ table and to make it normalize, we made another table called addresses to store the address information of a customer.

Then we made table to answer ‘What’ a customer can buy and his preference. So, for that we made a table named albums the actual product that company sale to its customers and give them option to order it online from website and this will be delivered to customers shipping address. Then to make it normalize we divided the object album into songs table, artist table and genre table. And the relationship between these was based on the idea to give a complete view of a product(album). For example information about album, which songs are related to this, who is the artist, how many artists sung this particular song and the album genre.

The problem that we solved in this is to get knowledge about albums easily, like how many customers buy this album to get idea about top sold and least sold product record. Information about people preview of songs so that we can offer albums to buy according songs listened by them. There is an order table that is made to record the information about order like who order that product, when the product is ordered, and how much product the customer ordered, to make it normalize we breaked it down to a order\_items table and a relationship is made between order\_items table and a albums table.

To know which customer take the help from our employee via online chat or on a website. And which employee handle these queries of a customer, to record this information we made a employee table and relationship between these tables made a third table called ‘employee\_has\_customer’ to know who help to which customer.

To record information about who returns the album, there is a returns table that record information about customer, order\_id, album\_id that customer return.

Requirement document

In this database for a website of music store that delivers albums to customers we have made 12 tables to capture the information in a meaningful way. Out of these 12 tables, some of the tables we have created and some are product of many-to-many relationship between them.

* In the **customer table**, customer\_id is a primary key and on email\_address a unique\_index is created to make email address unique for each customer. On password, first\_name, last\_name and on rest all columns of this tables are NOT NULL constraint is used.
* In **employee table**, employee\_id is the primary key constraint and on email\_address column unique key index is created to make the search easy and other columns first\_name and last\_name use NOT NULL constraint.
* In the **employee has customer table**, this table is auto generated due to many-to-many relationship between employee and customer. This table have two columns employee\_id and customer\_id, both are composite primary key and have NOT NULL constraint.
* In **order table**, order\_id is used as a primary key and customer\_id as foreign key from customer table. All columns except then first column have NOT NULL constraint.
* In **order items table**, order\_item\_id is the primary key and order\_id and album\_id is a foreign key constraint that help in identifying which order\_no ordered from order table. Here quantity have NOT NULL constraint and by default value is 1.
* In **genre table**, defines the category(pop, hip hop) of albums and have one primary key genre\_id. It has genre\_name columns as NOT NULL constraint.
* In the **artist table**, artist\_id is a primary key and other column like artist first\_name and last\_name and image as NOT NULL constraint and have no foreign key.
* In the **albums table,** album\_id is a primary key in this table and artist\_id and genre\_id is a foreign key. Foreign key constraint taken from album and artist to specify their relationship with these tables.
* In the **song table**, song\_id is used as its primary key. It has album\_id as foreign key to represent, song is related to which album. song\_name, size, image column uses NOT NULL constraint.
* In the **preview table**, preview is created because of many-to-many relationship between customers and song table, so it has two primary key customer\_id and song\_id with NOT NULL constraint.
* In the **addresses table**, it has one-to-many relationship with customer table. It has address\_id as a primary key and customer\_id as a foreign key to represent their relationship.
* In the **returns table**, it is made to store the information about the items returned by the customers. Return\_id is a primary key in this table. Customer\_id, order\_id, album\_id are foreign keys those show one-to-many relationship.

Problems solved

This database is made to designed to gain complete knowledge about customer, like what they are buying, when they are buying, customer music preferences. Which employee helped him and which album a particular customer return. To capture all this information through website, we design this database.

* Using this database, a music store can easily know how many albums have same price. They can group these albums to offer albums with the similar price range.
* Music store can easily find in which genre they do not have albums. They do not need to check manually all the albums to find out which genre we do not have. Using this database, it becomes very easy because it uses 3rd normalization form and keeps genre and albums table separate. So, to get the information from the huge data, it becomes easy to get it.
* As a company promotes product with actual price and discount price, it is very difficult to calculate the price that customer should pay. This database helps in solving this too. There is also a one view that helps in finding the grand total of all albums price.
* Database provide the solution in finding the albums not only by using name but also by their name’s first letter. If one wants to know only first name not the complete name then this database gives the solution.
* To check whether the albums order is shipped or not, we can easily use this database. So, it helps the music store in finding which product is shipped or not.
* To keep track on purchase of products in music store, we can use this database, using this system one can find out how many albums customer has purchased till now, how much discount the customer gets on that albums and how many actually pay for their albums after being discounted.
* To know when the particular customer made his first order, we can use this database, in return we can get customer details, when product was ordered with it’s date, type, category.