ONLINE APPLICATION ECOMMERCE PROJECT USING ORACLE

**STEP ONE:**

The project discussed in this paper was developed and tested using the Oracle database. I was able to implement this project after following the steps of setting up the working environment. This was achieved by installing oracle in my personal, and I could work from there. Before working on my project, I could set up the USER ID as indicated in the instructions.

**STEP TWO:**

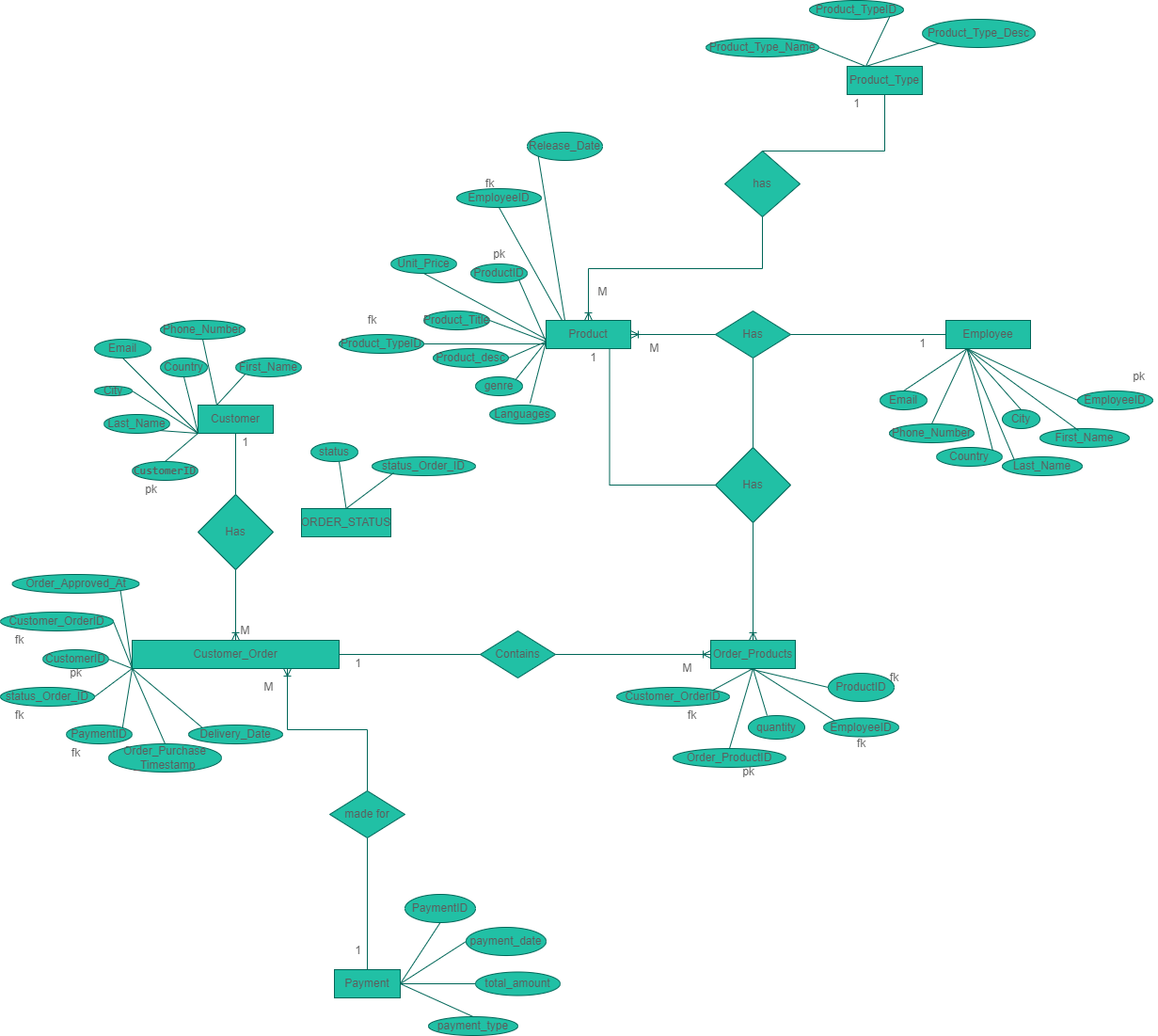
**PROJECT BUSINESS REQUIREMENTS**

With my topic at hand, I could come up with my business requirements. Using these requirements, I could implement the database for my application. The said requirements are as follows;

The e-shop is meant to sell music records, movies, and books to its customers around the globe. The use case specification that I come up with is as follows.

1. A customer submits orders for products.
2. A customer can place one order at a time.
3. An employee controls the products and is responsible for collecting the products for delivery.
4. The products are movies, books, and music records.
5. The product data includes title, description, unit price, release date, language genre, and product type.
6. Customer data shall consist of first name, last name, phone number, email, city, and country.

# ENTITY RELATIONSHIP DIAGRAM



**CLASS DIAGRAM**

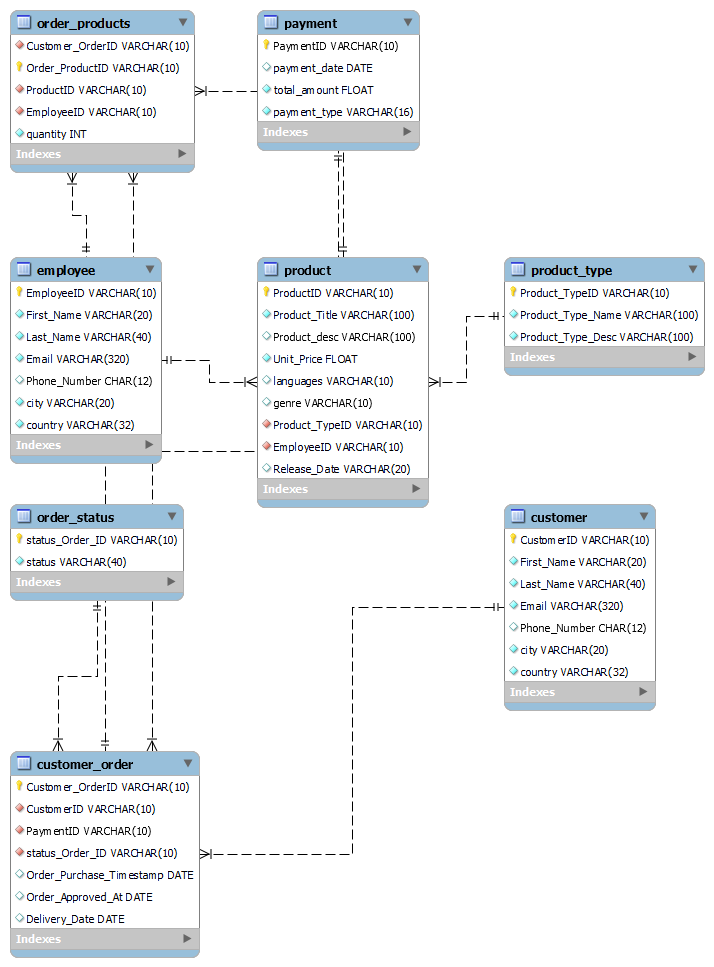
****

Figure 1 ER Diagram with appropriate relationship and cardinality for Use case provided.

**STEP THREE:**

I implemented my application project database by creating an online e-shop database followed by creating the normalized tables. The tables created in our database are Product, Customer, Product Type, Employee, Payment, Order products, Order status, and Customer Order. I did table normalization since it plays an essential role in the management of information in the whole database by preventing the replication of data in various tables at the same time. Through normalization, we employ simplicity in our database and make it concise.

There were relationships between various tables in our online e-shop database. The foreign key functionality achieved this. Each table had at least one primary key for the unique identification of entries in the database. I applied constraints such as NOT NULL and UNIQUE on various attributes of different tables based on the business requirement. I could implement the database schema with the help of data creation commands such as CREATE. I performed data information in each table so that I can be able to execute SQL queries in my database. Below is a screenshot of the same.

**STEP FOUR:**

After my database was done and had data already, I performed various basic SQL queries, such as SELECT, FROM, UPDATE, etc., to access the information I inserted in my database. This is one way of testing my database schema. Below is a screenshot of the same

Furthermore, I later performed advanced SQL queries. These queries involve accessing more than one table at the same time. In this case, I used SQL commands such as INNER JOIN to perform this. The advantage of using advanced queries is that it helps test the referential integrity created between various tables in our database. Below is a screenshot of the same

**Basic Queries**

**-- Extract all the customers from a specific city.**

SELECT \* from Customer as c

WHERE

c.city = "Lloyds";

**-- Search for a product of a specific genre**

SELECT \* from Product as p

WHERE

p.genre = 'genre-b';

**-- Count how many customers are from a specific city.**

SELECT COUNT(CustomerID)

FROM Customer

WHERE

city = "Lloyds";

-- **Calculate the average of the unit price.**

SELECT AVG(Unit\_Price)

FROM Product;

# Advanced Queries

**-- Extract all orders for books that has the keyword “the” in their description**

SELECT \*

FROM Order\_Products as op

INNER JOIN Product ON op.ProductID=Product.ProductID

WHERE

Product.Product\_desc LIKE '%the%';

**-- Extract all payments with credit cards for music records.**

SELECT \*

FROM Order\_Products as op

INNER JOIN Product ON op.ProductID=Product.ProductID

INNER JOIN Product\_Type ON Product.Product\_TypeID=Product\_Type.Product\_TypeID

INNER JOIN Customer\_Order ON op.Customer\_OrderID=Customer\_Order.Customer\_OrderID

INNER JOIN Payment ON Customer\_Order.PaymentID=Payment.PaymentID

WHERE

Product\_Type.Product\_Type\_Name = "music records" AND

Payment.payment\_type = "Credit card";

-- **Count how many employees handle music records.**

SELECT COUNT(Employee.EmployeeID)

FROM Product

INNER JOIN Employee ON Product.EmployeeID=Employee.EmployeeID

INNER JOIN Product\_Type ON Product.Product\_TypeID=Product\_Type.Product\_TypeID

WHERE

Product\_Type.Product\_Type\_Name = "music records";