Ein Bild, das Schrift, Text, Grafiken, Symbol enthält.

Automatisch generierte Beschreibung

**Assessment Submission Form**

|  |  |
| --- | --- |
| **Student Number**  (If this is group work, please include the student numbers of all group participants) | GH1026265 |
| **Assessment Title** | Individual Project Proposal |
| **Module Code** | M605A |
| **Module Title** | Advanced Databases |
| **Module Tutor** | Prof. Alireza Mahmoud |
| **Date Submitted** | 4th July, 2024 |

|  |
| --- |
| **Declaration of Authorship**  I declare that all material in this assessment is my own work except where there is clear acknowledgement and appropriate reference to the work of others.  I fully understand that the unacknowledged inclusion of another person’s writings or ideas or works in this work may be considered plagiarism and that, should a formal investigation process confirms the allegation, I would be subject to the penalties associated with plagiarism, as per GISMA Business School, University of Applied Sciences’ regulations for academic misconduct.  Signed……………Date ……………4th July 2024……………… |

Database Desing and Implementation for ABC online store

GISMA University of Applied Sciences

M605A Advanced Databases

Instructor: Professor Alireza Mahmoud

Final Assessment

July 4, 2024

By:

Eddixon Castillo Ulloa

(GH1026265)

Table of Contents

[INTRODUCTION 4](#_Toc170836370)

[DESCRIPTION 4](#_Toc170836371)

[DATABASE DESIGN 4](#_Toc170836372)

[LIST OF TABLES 6](#_Toc170836373)

[DATABASE DATA 7](#_Toc170836374)

[REPORTING 11](#_Toc170836375)

[Detailed information about suppliers and the number of products that they provide. 11](#_Toc170836376)

[10 best-selling products with the total amount and their supplier. 12](#_Toc170836377)

[List of customers and their total purchases 13](#_Toc170836378)

[List of returned items. 13](#_Toc170836379)

[List of products in a specific category that were sold last month. 14](#_Toc170836380)

[STORE PROCEDURE AND TRIGGERS 15](#_Toc170836381)

[INDEXES 16](#_Toc170836382)

[TECHNICAL SPECIFICATIONS 17](#_Toc170836383)

[CONCLUSIONS 18](#_Toc170836384)

# INTRODUCTION

The objective of this project is to design and implement a database system for "ABC online store". Among the functions that aforementioned database will be able to manage information on customers, orders, available products, suppliers, returns, location of product image files, comments and ratings.

It will begin by describing the problem to be solved. The entity relationship model with its respective cardinalities will be shown below. Then a description of the designed scheme will be made. The stored data samples will then be displayed. Below, a detailed explanation of the SQL queries used to resolve the proposed questions will be given. Finally, some recommendations and general conclusions.

# DESCRIPTION

”ABC Online Store” is a shop company that sells products across the European continent. To have better control over its operations, the company has decided to implement a database that will help to manage its information about clients, products, orders, etc. The model should be able to support the following reports:

* Detailed information about suppliers and the number of products that they provide.
* 10 best-selling products with the total amount and their supplier.
* List of customers and their total purchases.
* List of returned items.
* List of products in the fashion (or any other category) category that were sold last month.

# DATABASE DESIGN

To build the database, the following model has been proposed:

A screenshot of a computer

Description automatically generated

Regarding the model, the following naming convention has been adopted

* Table names should be nouns in singular.
* The primary is always called id, in case the primary key is composed, must be start with a noun followed by the string ‘\_id’.
* The foreign keys must start with the name of the table that belongs to, and then ends with the suffix ‘\_id’.
* All names of tables and columns must be written using snake case naming convention, i.e. using the underscore (‘\_’) as separator. E.g.: returned\_item, last\_login.
* Any discrepancy should be discussed with the database administrator (DBA) or Software Architect (or engineer) on charge.

# LIST OF TABLES

For this model two type of table were considered:

* **Business tables**: These tables are intended to store relevant information that will be used by the company, also their size and structure is varying the most of time. Therefore, on these tables optimization tasks must be performed (e.g.: indexing, adjusting queries, denormalization)
* **Parameters tables**: It is true that certain attributes are inherent and depend on the entities being modelled, e.g.: an order could have several types of status such *in stock, on checkout, shipping, delivered,* etc. To avoid inconsistencies with those status’ names when an order’s status is stored or updated (i.e.: a status attribute can be stored or updated as *delivered* or *DELIvered* ) , is preferred to store those names in **parameters tables (**a.k.a domain tables, configuration tables, dictionary tables**).** These tables help to keep the consistency of the data, normally are consisted of two columns and rarely (or even never) their structure and size change.

|  |  |  |  |
| --- | --- | --- | --- |
| Table | Type of table | Purpose | Foreign Relationships |
| *comment* | Business | Is intended to store the product’s comments. The comments can be anonymous or given by a customer. | *Product:* the product which the comment is written.  *Rating:* The comment should contain a rate the reflects the opinion of the writer’s comment. Can be numerical or descriptive.  *Customer:* the foreign key of the customer who gives the review. If this value is null, the comment is assumed as anonymous. |
| *rating* | Parameters |  |  |
| *customer* | Business | Contains the essential information of the customers of *ABC Online Store*. |  |
| *product* | Business | Keep the relevant information of the products sold by *ABC Online Store* | *Supplier:* Company or person that provide to *ABC Store Online* the products to be sold.  *Category*: classification to which the product belongs |
| *product\_category* | Parameters | Stores the categories in which a product can be classified. |  |
| *product\_image* | Business | Save the paths where the product's photos are allocated. | *Product*: foreign key of the product that the images belong to. |
| *order* | Business | Contains the essential information of order made by customers. | *Order\_status:* the possible statuses that an order can have since it is started until is finished.  *Customer*: who buys items from the store. |
| *order\_status* | Parameters | Keeps the different status’ values than an order can have. |  |
| *items\_by\_order* | Business | Keeps the information of what and how many products were ordered. | Order: Order where the items are included.  Product: ordered product. |
| *returned\_item* | Business | Stores the returned items from an order. | Order: The associated order  Product: the returned products. |
| *supplier* | Business | Person or company that provides the products sold by *ABC Online Store* |  |

# DATABASE DATA

**SELECT** \* **FROM** product p

A screenshot of a computer

Description automatically generated

**SELECT** \* **FROM** supplier s

A screenshot of a computer

Description automatically generated

**SELECT** \* **FROM** customer c

A screenshot of a computer

Description automatically generated

**SELECT** \* **FROM** `order` o

A screenshot of a computer

Description automatically generated

**SELECT** \* **FROM** items\_by\_order **io**

A screenshot of a computer

Description automatically generated

**SELECT** \* **FROM** `comment`cm

A screenshot of a computer

Description automatically generated

**SELECT** \* **FROM** product\_category pc

A screenshot of a computer

Description automatically generated

**SELECT** \* **FROM** product\_image **pi**

A screenshot of a computer

Description automatically generated

**SELECT** \* **FROM** rating r

A screenshot of a computer

Description automatically generated

# REPORTING

## Detailed information about suppliers and the number of products that they provide.

This query shows the product provider by each supplier:

**SELECT** s.id, s.**name**, p.id, p.**name**

**FROM** supplier s

**LEFT** **JOIN** product p **ON** s.id = p.supplier\_id;

A screenshot of a computer

Description automatically generated

As the previous image shows, the supplier *H and M* provides two products, *14-800* provides zero, and the rest provide one for each one of them.

So, the query that will show only the count of products besides the information of each supplier will be:

**SELECT** *s*.id, *s*.name,*s*.phone,*s*.address,*s*.email, **COUNT**(*p*.id) **as** **'number of products'**

**FROM** supplier *s*

**LEFT** **JOIN** product *p* **ON** *s*.id = *p*.supplier\_id

**GROUP** **BY** *s*.id ;

A screenshot of a computer

Description automatically generated

This query fetches specific information about each supplier, such as their ID, Name, contact information, and the total count of products they provide. It employs a LEFT JOIN operation with the *Product* table based on *Supplier\_id* field to count the number of products associated with each supplier.

## 10 best-selling products with the total amount and their supplier.

Regarding the sold products according to the ones stored in the *item\_by\_order* table:

**SELECT** ibo.order\_id, p.**name** **AS** product, ibo.quatity, ibo.unit\_price

**FROM** items\_by\_order ibo

**INNER** **JOIN** product p **ON** p.id = ibo.product\_id;

**A screenshot of a computer

Description automatically generated**

The top 10 of most sold products and their suppliers are:

**SELECT** p.id, p.**name** **AS** 'Product', s.**name** **AS** 'Supplier', **SUM**(ibo.quatity) **as** Total

**FROM** items\_by\_order ibo

**INNER** **JOIN** product p **ON** p.id = ibo.product\_id

**INNER** **JOIN** supplier s **ON** s.id = p.supplier\_id

**GROUP** **BY** p.id, s.id

**ORDER** **BY** Total **DESC**;

A table with a list of products

Description automatically generated

This query determines the top 10 best-selling products by summing up the total quantity sold for each product. It integrates data from the *items\_by\_order*, *Product*, and *Supplier* tables to gather information on products’ and suppliers’ names. The results are grouped by Product’s ID, *product’s* Name, and *supplier’s* Name and sorted in descending order based on total, representing the total number of units sold.

## List of customers and their total purchases

**SELECT** c.id, c.first\_name,c.last\_name, c.phone\_number,c.address, c.email

,**COALESCE** ( **SUM**(ibo.quatity),0) **AS** purchases

**FROM** customer c

**LEFT** **JOIN** `order` o **ON** c.id = o.customer\_id

**LEFT** **JOIN** items\_by\_order ibo **ON** o.id=ibo.order\_id

**GROUP** **BY** c.id

A screenshot of a computer

Description automatically generated

This query displays a list of all customers’ information, and their total product purchases. The LEFT JOIN clauses join information from the c*ustomer*, o*rder*, and *item\_by\_order* tables. The results are grouped by Customer’s Id. The total purchases for each customer are calculated by the aggregate function SUM. If SUM return a NULL value, this is replaced by zero (0) using the function COALESCE.

## List of returned items.

**SELECT** ri.id, ibo.Id **AS** item\_order\_id, ibo.order\_id **AS** Order\_id , ibo.quatity **AS** ordered,p.**name**

, ri.quantity **AS** returned, ri.**date** **AS** return\_date, ri.reason

**FROM** returned\_item ri

**INNER** **JOIN** items\_by\_order ibo **ON** ibo.ID = ri.item\_by\_order\_id

**INNER** **JOIN** product p **ON** p.id = ibo.product\_id;



This query retrieves information about returned items, including the returned\_item’s ID, item\_by\_order’s ID, the product’s name, the number of ordered and returned itmes, Return’s Date and the reason. It uses JOIN operations on the Returns, items\_by\_order, and Products tablesto gather relevant data about the returned items and the products associated with them.

## List of products in a specific category that were sold last month.

**SELECT** c.**name** **AS** category, p.**name** **AS** product, o.order\_date

**FROM** product p

**INNER** **JOIN** product\_category c **ON** c.id = p.category\_id

**INNER** **JOIN** items\_by\_order ibo **ON** ibo.product\_id = p.id

**INNER** **JOIN** `order` o **ON** o.id = ibo.order\_id

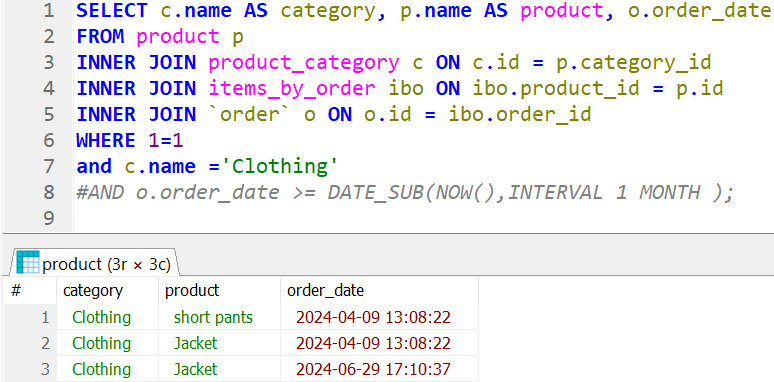
**WHERE** 1=1

**and** c.**name** ='Clothing'

**AND** o.order\_date >= **DATE\_SUB**(**NOW**(),**INTERVAL** 1 **MONTH** );

The statement fetches products categorized as ‘Clothing’ that were ordered within the last month. The join with *category\_product* table helps to get the category name. The join with *items\_by\_order* table brings the items that belongs to the *order.*  The function *DATE\_SUB()* subtracts the given interval unit (i.e.: one month) from the starting date (for this case the current one)

**Without order date condition.**

****

**With the order date condition**

A screenshot of a computer code

Description automatically generated

# STORE PROCEDURE AND TRIGGERS

As can be seen in the model there is a relationship between the tables *order* and *items\_by\_order.* Since the table *order* stores the general information of an order, but the details of the purchased products must be stored in the table *items\_by\_order*. So, to update the total value of an order, using standard SQL instructions can be a very time-consuming task, inclusive errors and miscalculations can be done, if is done manually.

A screenshot of a computer

Description automatically generated

The RDBMS MariaDB provide a built-in language called PL/SQL (Procedural Language for SQL) with which it is possible to automatize these calculations.

With PL/SQL repetitive scripts can be executed by several types of objects: Functions, stored procedures, triggers.

For the proposed model, the following objects were implemented. Consisted in one stored procedure and three triggers. Those objects make sure that whenever an item is deleted, inserted or updated into the table *item\_by\_order,* the total price for the order to belongs to, will be automatically calculated and updated in the *`order`* table, assuring coherence and consistency with the data.

A screenshot of a computer code

Description automatically generated

# INDEXES

Considering that the business tables will increase their sizes as time will come, a set of indexes will help to improve the performance of queries that search or filter based on specific columns, such as *email* and *phone* in the *Customers* tables, since will be important for *ABC Online Store* locate them using these criteria, either to offer discounts and promotions or attending properly their reclamations.

A screenshot of a computer

Description automatically generated

Besides of the default indexes (primary and foreign keys indexes), a index on the *order\_date* field of the table *order,* will help to the store locate the orders in a date range because this type of index can be use with the most common comparison operators in this type of field.

A screenshot of a computer

Description automatically generated

Indexes in the *product* table for the fields *supplier\_id* and *category\_id* will let the store classify the products either by supplier or category. These indexes were created by default by the RDBMS.

A screenshot of a computer

Description automatically generated

A similar case applies to the table *items\_by\_order.*

*A screenshot of a computer

Description automatically generated’*

# TECHNICAL SPECIFICATIONS

The present solution is built using MariaDB RDBMS. The database script will be allocated in the following repository:  
<https://github.com/eddixoncu/M605A_Final>

The contents of the repository consist of:

* Image of the Entity relationship Diagram.
* The present report.
* The full script of the database, including the DDL and DML SQL instructions, i.e.: CREATE tables, triggers, store procedure, also the insertion statements of the data.

# CONCLUSIONS

The ABC Company Database Management System project effectively implements a comprehensive database designed to manage customers, orders, items, purchases, and products. This system is engineered to efficiently handle diverse processes, providing valuable insights through the use of queries, triggers, and performance optimization techniques.

It is expected that the present document will serve to others to help them how to build more robust databases.