Gisma University of Applied Sciences

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………[illegible]……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

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.

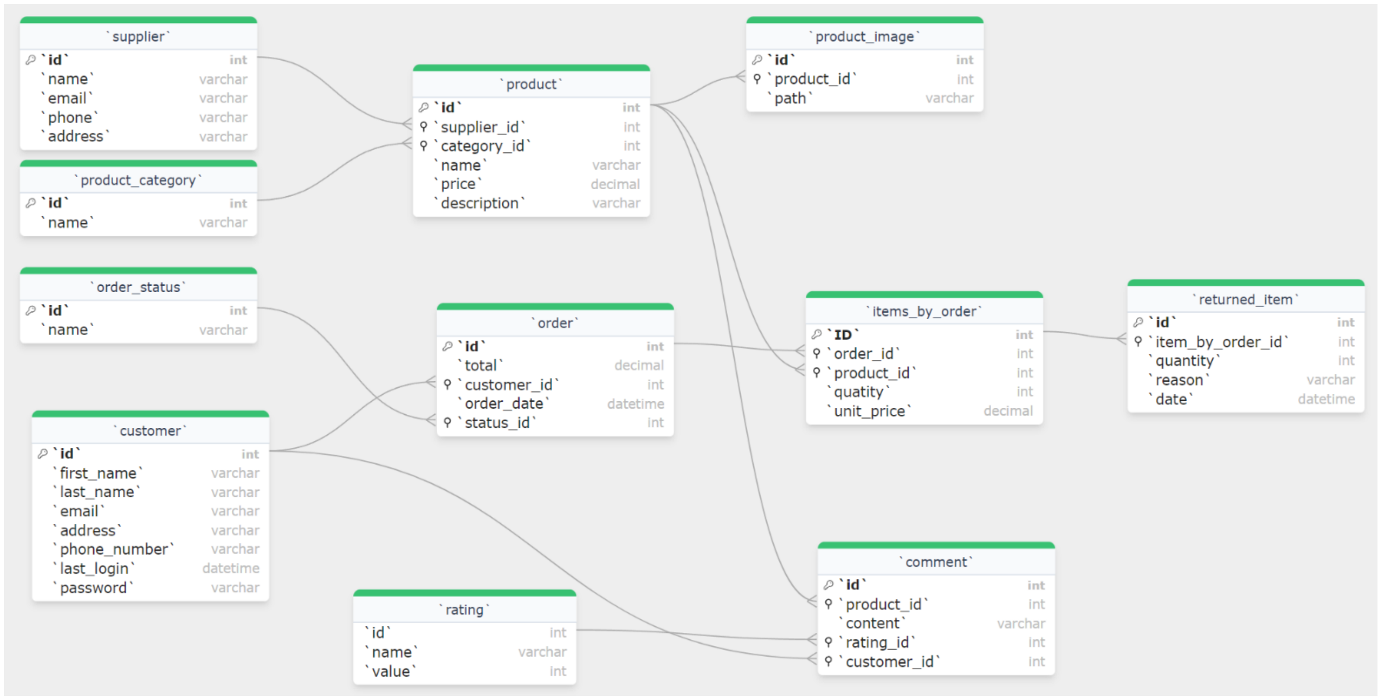
DESCRIPTION

”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:



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

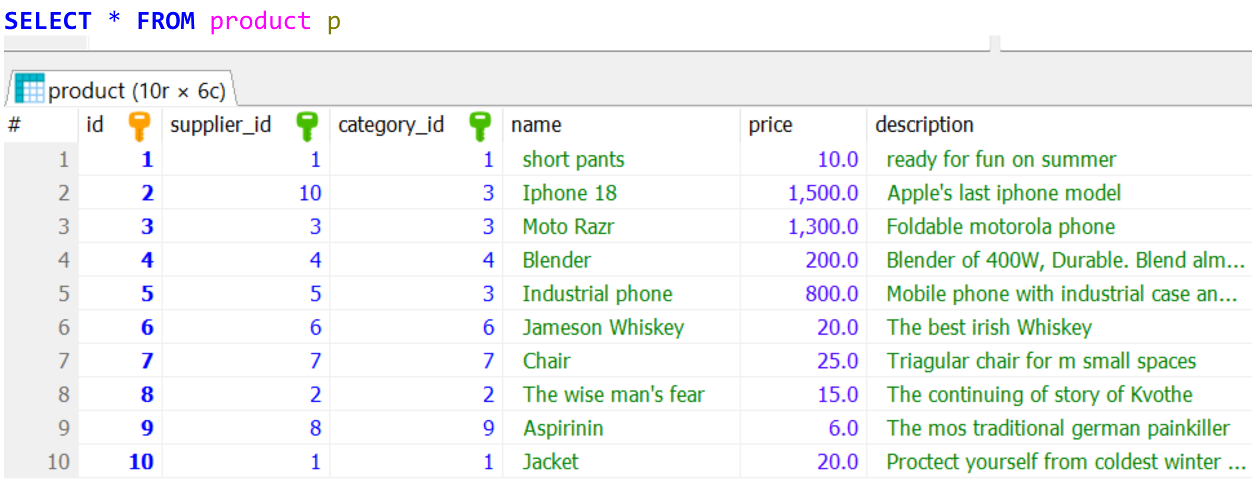
For this model two type of table were considered:

- **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*	

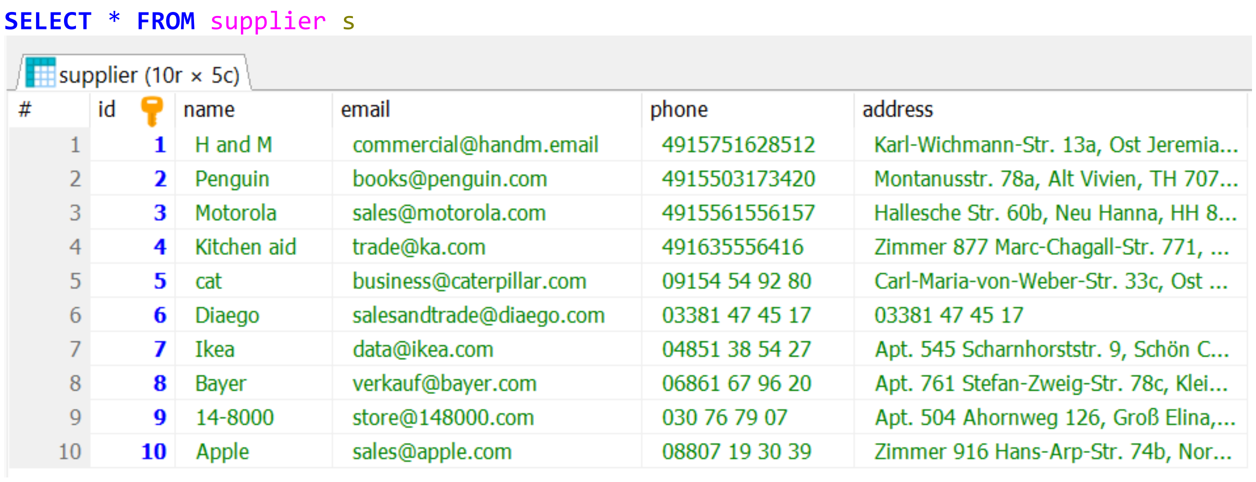
DATABASE DATA



```
SELECT * FROM product p
```

product (10r × 6c)

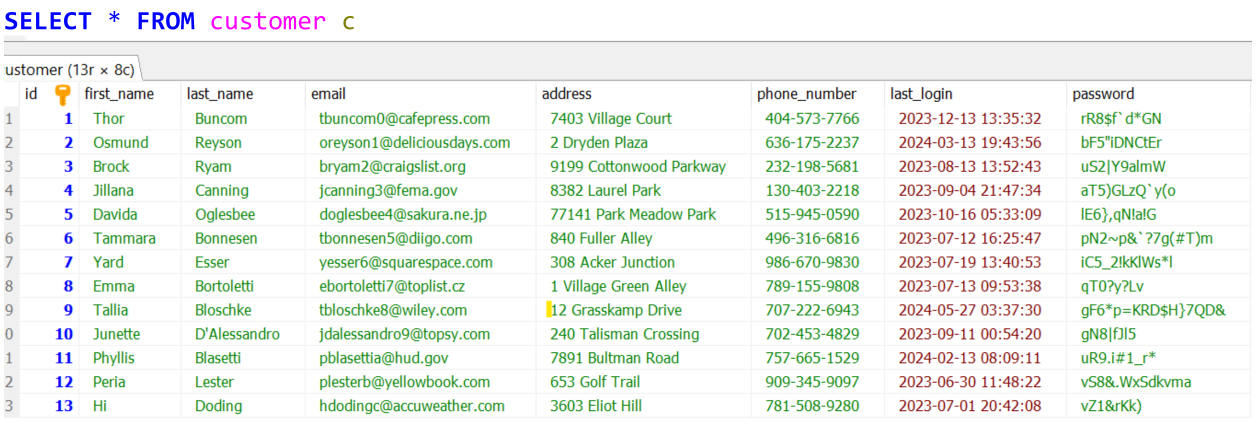
#	id	supplier_id	category_id	name	price	description
1	1	1	1	short pants	10.0	ready for fun on summer
2	2	10	3	Iphone 18	1,500.0	Apple's last iphone model
3	3	3	3	Moto Razr	1,300.0	Foldable motorola phone
4	4	4	4	Blender	200.0	Blender of 400W, Durable. Blend alm...
5	5	5	3	Industrial phone	800.0	Mobile phone with industrial case an...
6	6	6	6	Jameson Whiskey	20.0	The best irish Whiskey
7	7	7	7	Chair	25.0	Triagular chair for m small spaces
8	8	2	2	The wise man's fear	15.0	The continuing of story of Kvothe
9	9	8	9	Aspirinin	6.0	The mos traditional german painkiller
10	10	1	1	Jacket	20.0	Proctect yourself from coldest winter ...



```sql
SELECT * FROM supplier s
```

supplier (10r × 5c)

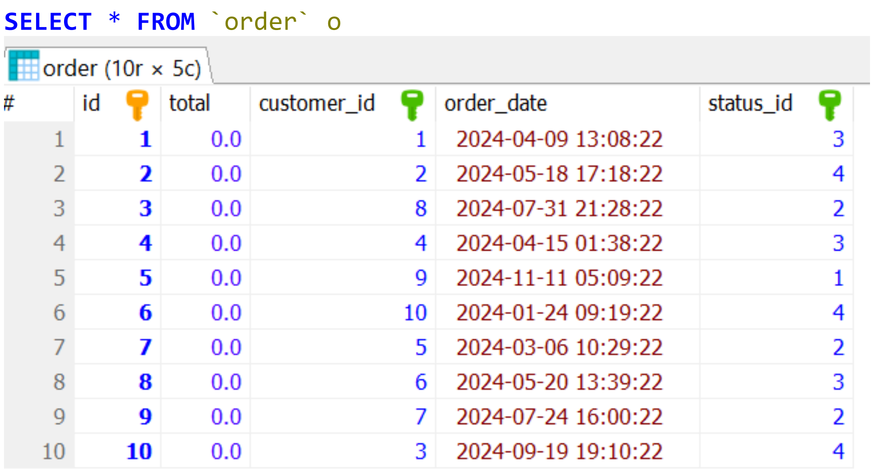
#	id	name	email	phone	address
1	1	H and M	commercial@handm.email	4915751628512	Karl-Wichmann-Str. 13a, Ost Jeremia...
2	2	Penguin	books@penguin.com	4915503173420	Montanusstr. 78a, Alt Vivien, TH 707...
3	3	Motorola	sales@motorola.com	4915561556157	Hallesche Str. 60b, Neu Hanna, HH 8...
4	4	Kitchen aid	trade@ka.com	491635556416	Zimmer 877 Marc-Chagall-Str. 771, ...
5	5	cat	business@caterpillar.com	09154 54 92 80	Carl-Maria-von-Weber-Str. 33c, Ost ...
6	6	Diaego	salesandtrade@diaego.com	03381 47 45 17	03381 47 45 17
7	7	Ikea	data@ikea.com	04851 38 54 27	Apt. 545 Scharnhorststr. 9, Schön C...
8	8	Bayer	verkauf@bayer.com	06861 67 96 20	Apt. 761 Stefan-Zweig-Str. 78c, Klei...
9	9	14-8000	store@148000.com	030 76 79 07	Apt. 504 Ahornweg 126, Groß Elina,...
10	10	Apple	sales@apple.com	08807 19 30 39	Zimmer 916 Hans-Arp-Str. 74b, Nor...



```sql
SELECT * FROM customer c
```

ustomer (13r × 8c)

	id	first_name	last_name	email	address	phone_number	last_login	password
1	1	Thor	Buncom	tbuncom0@cafepress.com	7403 Village Court	404-573-7766	2023-12-13 13:35:32	rR8$f` d*GN
2	2	Osmund	Reyson	oreyson1@deliciousdays.com	2 Dryden Plaza	636-175-2237	2024-03-13 19:43:56	bF5"iDNCtEr
3	3	Brock	Ryam	bryam2@craigslist.org	9199 Cottonwood Parkway	232-198-5681	2023-08-13 13:52:43	uS2|Y9almW
4	4	Jillana	Canning	jcanning3@fema.gov	8382 Laurel Park	130-403-2218	2023-09-04 21:47:34	aT5)GLzQ`y(o
5	5	Davida	Oglesbee	doglesbee4@sakura.ne.jp	77141 Park Meadow Park	515-945-0590	2023-10-16 05:33:09	lE6},qN!a!G
6	6	Tammara	Bonnesen	tbonnesen5@diigo.com	840 Fuller Alley	496-316-6816	2023-07-12 16:25:47	pN2~p&`?7g(#T)m
7	7	Yard	Esser	yesser6@squarespace.com	308 Acker Junction	986-670-9830	2023-07-19 13:40:53	iC5_2!kKlWs*l
8	8	Emma	Bortoletti	ebortoletti7@toplist.cz	1 Village Green Alley	789-155-9808	2023-07-13 09:53:38	qT0?y?Lv
9	9	Tallia	Bloschke	tbloschke8@wiley.com	12 Grasskamp Drive	707-222-6943	2024-05-27 03:37:30	gF6*p=KRD$H}7QD&
0	10	Junette	D'Alessandro	jdalessandro9@topsy.com	240 Talisman Crossing	702-453-4829	2023-09-11 00:54:20	gN8|fJl5
1	11	Phyllis	Blasetti	pblasettia@hud.gov	7891 Bultman Road	757-665-1529	2024-02-13 08:09:11	uR9.i#1_r*
2	12	Peria	Lester	plesterb@yellowbook.com	653 Golf Trail	909-345-9097	2023-06-30 11:48:22	vS8&.WxSdkvma
3	13	Hi	Doding	hdodingc@accuweather.com	3603 Eliot Hill	781-508-9280	2023-07-01 20:42:08	vZ1&rKk)

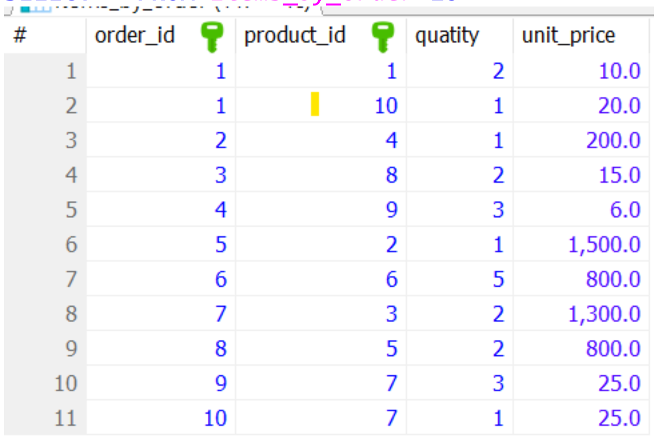


```sql
SELECT * FROM `order` o
```

order (10r × 5c)

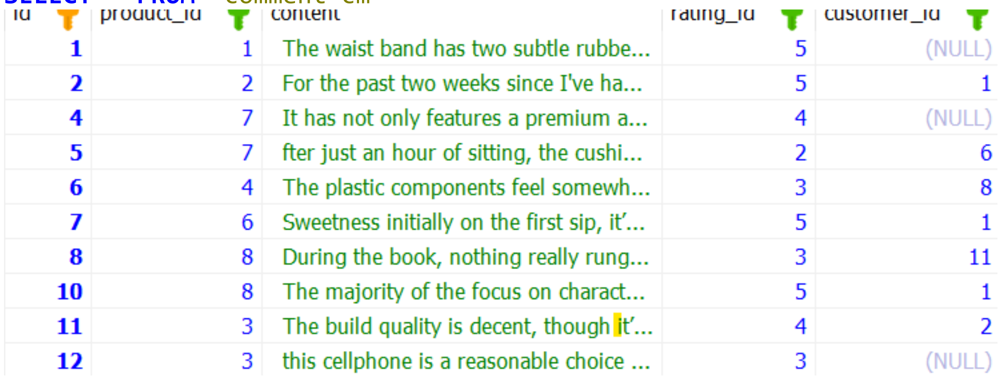
#	id	total	customer_id	order_date	status_id
1	1	0.0	1	2024-04-09 13:08:22	3
2	2	0.0	2	2024-05-18 17:18:22	4
3	3	0.0	8	2024-07-31 21:28:22	2
4	4	0.0	4	2024-04-15 01:38:22	3
5	5	0.0	9	2024-11-11 05:09:22	1
6	6	0.0	10	2024-01-24 09:19:22	4
7	7	0.0	5	2024-03-06 10:29:22	2
8	8	0.0	6	2024-05-20 13:39:22	3
9	9	0.0	7	2024-07-24 16:00:22	2
10	10	0.0	3	2024-09-19 19:10:22	4

```sql
SELECT * FROM items_by_order io
```



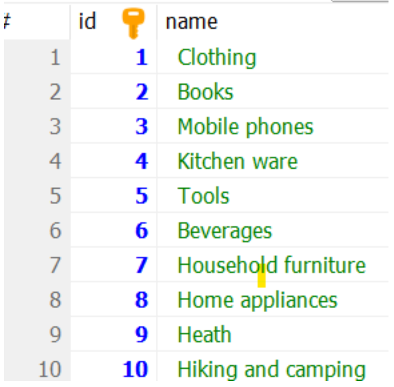
#	order_id	product_id	quatity	unit_price
1	1	1	2	10.0
2	1	10	1	20.0
3	2	4	1	200.0
4	3	8	2	15.0
5	4	9	3	6.0
6	5	2	1	1,500.0
7	6	6	5	800.0
8	7	3	2	1,300.0
9	8	5	2	800.0
10	9	7	3	25.0
11	10	7	1	25.0

```sql
SELECT * FROM `comment` cm
```



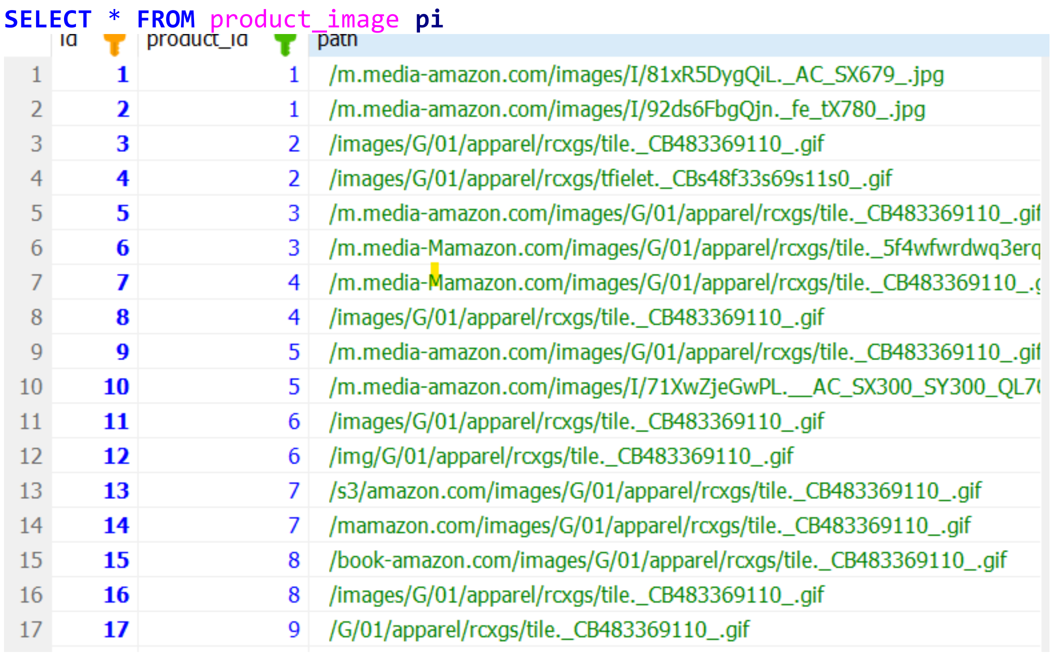
id	product_id	content	rating_id	customer_id
1	1	The waist band has two subtle rubbe...	5	(NULL)
2	2	For the past two weeks since I've ha...	5	1
4	7	It has not only features a premium a...	4	(NULL)
5	7	fter just an hour of sitting, the cushi...	2	6
6	4	The plastic components feel somewh...	3	8
7	6	Sweetness initially on the first sip, it'...	5	1
8	8	During the book, nothing really rung...	3	11
10	8	The majority of the focus on charact...	5	1
11	3	The build quality is decent, though it'...	4	2
12	3	this cellphone is a reasonable choice ...	3	(NULL)

```sql
SELECT * FROM product_category pc
```



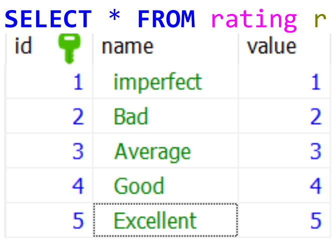
#	id	name
1	1	Clothing
2	2	Books
3	3	Mobile phones
4	4	Kitchen ware
5	5	Tools
6	6	Beverages
7	7	Household furniture
8	8	Home appliances
9	9	Heath
10	10	Hiking and camping

```sql
SELECT * FROM product_image pi
```



	id	product_id	path
1	1	1	/m.media-amazon.com/images/I/81xR5DygQiL._AC_SX679_.jpg
2	2	1	/m.media-amazon.com/images/I/92ds6FbgQjn._fe_tX780_.jpg
3	3	2	/images/G/01/apparel/rcxgs/tile._CB483369110_.gif
4	4	2	/images/G/01/apparel/rcxgs/tfielet._CBs48f33s69s11s0_.gif
5	5	3	/m.media-amazon.com/images/G/01/apparel/rcxgs/tile._CB483369110_.gif
6	6	3	/m.media-Mamazon.com/images/G/01/apparel/rcxgs/tile._5f4wfwrdwq3erq
7	7	4	/m.media-Mamazon.com/images/G/01/apparel/rcxgs/tile._CB483369110_.
8	8	4	/images/G/01/apparel/rcxgs/tile._CB483369110_.gif
9	9	5	/m.media-amazon.com/images/G/01/apparel/rcxgs/tile._CB483369110_.gif
10	10	5	/m.media-amazon.com/images/I/71XwZjeGwPL.__AC_SX300_SY300_QL7
11	11	6	/images/G/01/apparel/rcxgs/tile._CB483369110_.gif
12	12	6	/img/G/01/apparel/rcxgs/tile._CB483369110_.gif
13	13	7	/s3/amazon.com/images/G/01/apparel/rcxgs/tile._CB483369110_.gif
14	14	7	/mamazon.com/images/G/01/apparel/rcxgs/tile._CB483369110_.gif
15	15	8	/book-amazon.com/images/G/01/apparel/rcxgs/tile._CB483369110_.gif
16	16	8	/images/G/01/apparel/rcxgs/tile._CB483369110_.gif
17	17	9	/G/01/apparel/rcxgs/tile._CB483369110_.gif

```sql
SELECT * FROM rating r
```



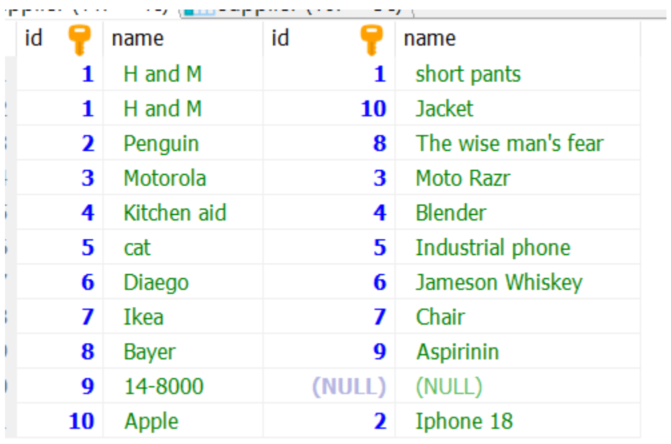
id	name	value
1	imperfect	1
2	Bad	2
3	Average	3
4	Good	4
5	Excellent	5

REPORTING

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

This query shows the product provider by each supplier:

```sql
SELECT s.id, s.name, p.id, p.name
FROM supplier s
LEFT JOIN product p ON s.id = p.supplier_id;
```

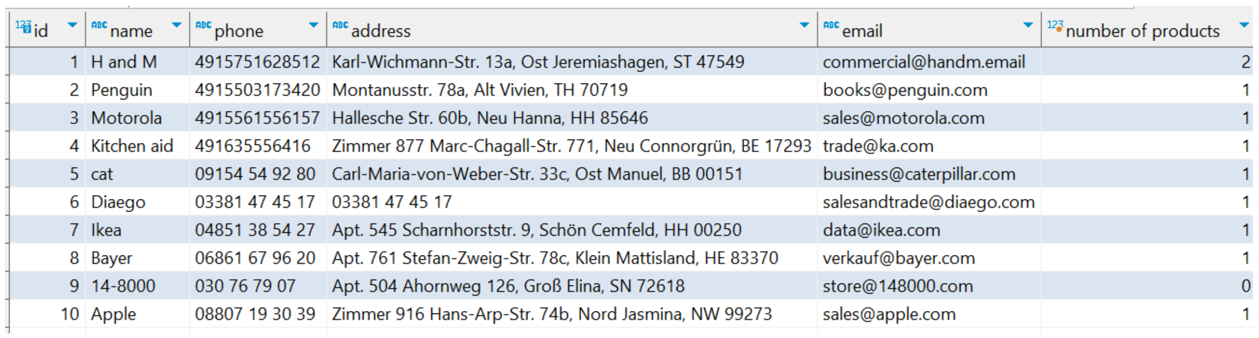


id	name	id	name
1	H and M	1	short pants
1	H and M	10	Jacket
2	Penguin	8	The wise man's fear
3	Motorola	3	Moto Razr
4	Kitchen aid	4	Blender
5	cat	5	Industrial phone
6	Diaego	6	Jameson Whiskey
7	Ikea	7	Chair
8	Bayer	9	Aspirinin
9	14-8000	(NULL)	(NULL)
10	Apple	2	Iphone 18

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:

```sql
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 ;
```



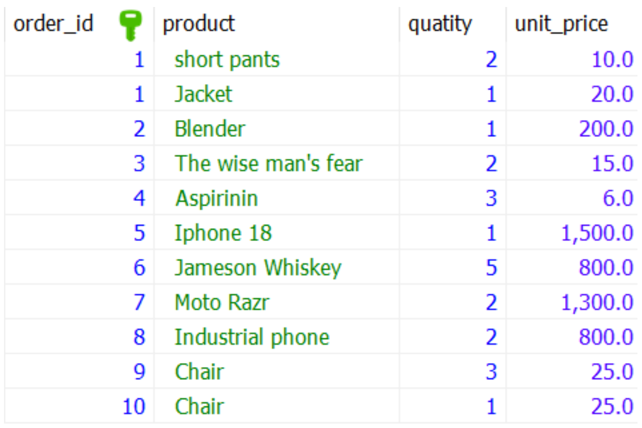
id	name	phone	address	email	number of products
1	H and M	4915751628512	Karl-Wichmann-Str. 13a, Ost Jeremiashagen, ST 47549	commercial@handm.email	2
2	Penguin	4915503173420	Montanusstr. 78a, Alt Vivien, TH 70719	books@penguin.com	1
3	Motorola	4915561556157	Hallesche Str. 60b, Neu Hanna, HH 85646	sales@motorola.com	1
4	Kitchen aid	491635556416	Zimmer 877 Marc-Chagall-Str. 771, Neu Connorgrün, BE 17293	trade@ka.com	1
5	cat	09154 54 92 80	Carl-Maria-von-Weber-Str. 33c, Ost Manuel, BB 00151	business@caterpillar.com	1
6	Diaego	03381 47 45 17	03381 47 45 17	salesandtrade@diaego.com	1
7	Ikea	04851 38 54 27	Apt. 545 Scharnhorststr. 9, Schön Cemfeld, HH 00250	data@ikea.com	1
8	Bayer	06861 67 96 20	Apt. 761 Stefan-Zweig-Str. 78c, Klein Mattisland, HE 83370	verkauf@bayer.com	1
9	14-8000	030 76 79 07	Apt. 504 Ahornweg 126, Groß Elina, SN 72618	store@148000.com	0
10	Apple	08807 19 30 39	Zimmer 916 Hans-Arp-Str. 74b, Nord Jasmina, NW 99273	sales@apple.com	1

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:

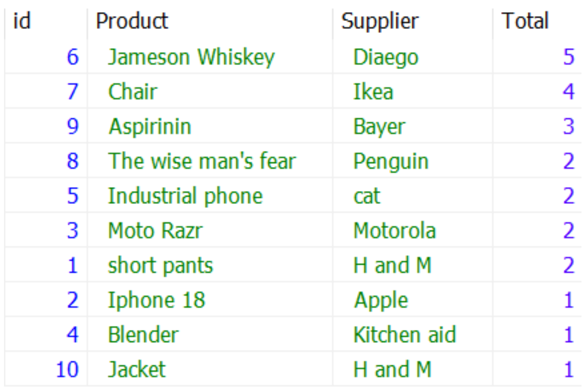
```
SELECT ibo.order_id, p.name AS product, ibo.quatity, ibo.unit_price
FROM items_by_order ibo
INNER JOIN product p ON p.id = ibo.product_id;
```



order_id	product	quatity	unit_price
1	short pants	2	10.0
1	Jacket	1	20.0
2	Blender	1	200.0
3	The wise man's fear	2	15.0
4	Aspirinin	3	6.0
5	Iphone 18	1	1,500.0
6	Jameson Whiskey	5	800.0
7	Moto Razr	2	1,300.0
8	Industrial phone	2	800.0
9	Chair	3	25.0
10	Chair	1	25.0

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;
```

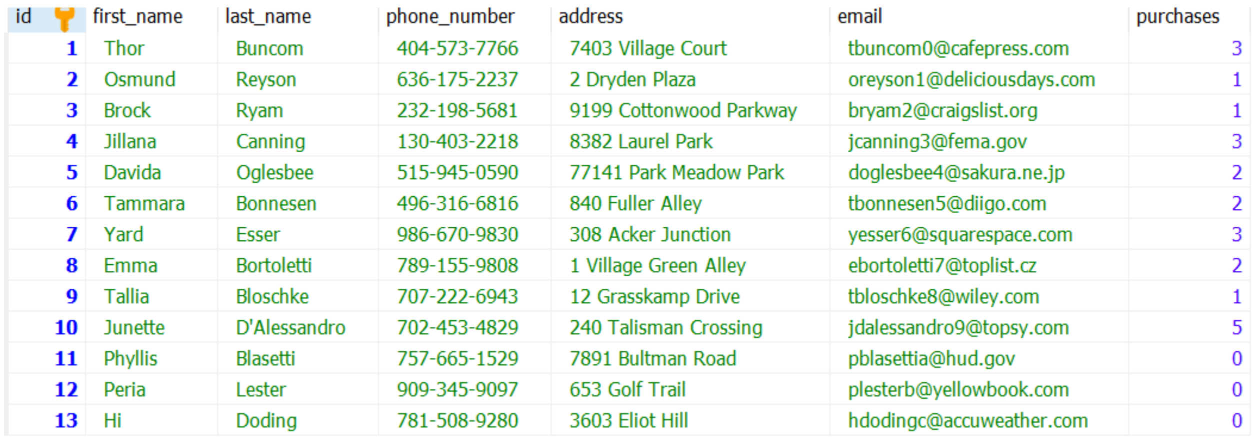


id	Product	Supplier	Total
6	Jameson Whiskey	Diaego	5
7	Chair	Ikea	4
9	Aspirinin	Bayer	3
8	The wise man's fear	Penguin	2
5	Industrial phone	cat	2
3	Moto Razr	Motorola	2
1	short pants	H and M	2
2	Iphone 18	Apple	1
4	Blender	Kitchen aid	1
10	Jacket	H and M	1

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
```

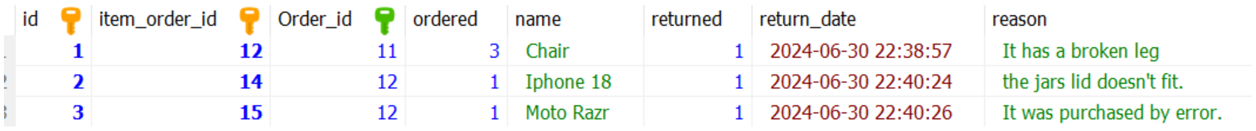


id	first_name	last_name	phone_number	address	email	purchases
1	Thor	Buncom	404-573-7766	7403 Village Court	tbuncom0@cafepress.com	3
2	Osmund	Reyson	636-175-2237	2 Dryden Plaza	oreyson1@deliciousdays.com	1
3	Brock	Ryam	232-198-5681	9199 Cottonwood Parkway	bryam2@craigslist.org	1
4	Jillana	Canning	130-403-2218	8382 Laurel Park	jcanning3@fema.gov	3
5	Davida	Oglesbee	515-945-0590	77141 Park Meadow Park	doglesbee4@sakura.ne.jp	2
6	Tammara	Bonnesen	496-316-6816	840 Fuller Alley	tbonnesen5@diigo.com	2
7	Yard	Esser	986-670-9830	308 Acker Junction	yesser6@squarespace.com	3
8	Emma	Bortoletti	789-155-9808	1 Village Green Alley	ebortoletti7@toplist.cz	2
9	Tallia	Bloschke	707-222-6943	12 Grasskamp Drive	tbloschke8@wiley.com	1
10	Junette	D'Alessandro	702-453-4829	240 Talisman Crossing	jdalessandro9@topsy.com	5
11	Phyllis	Blasetti	757-665-1529	7891 Bultman Road	pblasettia@hud.gov	0
12	Peria	Lester	909-345-9097	653 Golf Trail	plesterb@yellowbook.com	0
13	Hi	Doding	781-508-9280	3603 Eliot Hill	hdodingc@accuweather.com	0

This query displays a list of all customers' information, and their total product purchases. The LEFT JOIN clauses join information from the c*ustomer*, *order*, 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;
```



id	item_order_id	Order_id	ordered	name	returned	return_date	reason
1	12	11	3	Chair	1	2024-06-30 22:38:57	It has a broken leg
2	14	12	1	Iphone 18	1	2024-06-30 22:40:24	the jars lid doesn't fit.
3	15	12	1	Moto Razr	1	2024-06-30 22:40:26	It was purchased by error.

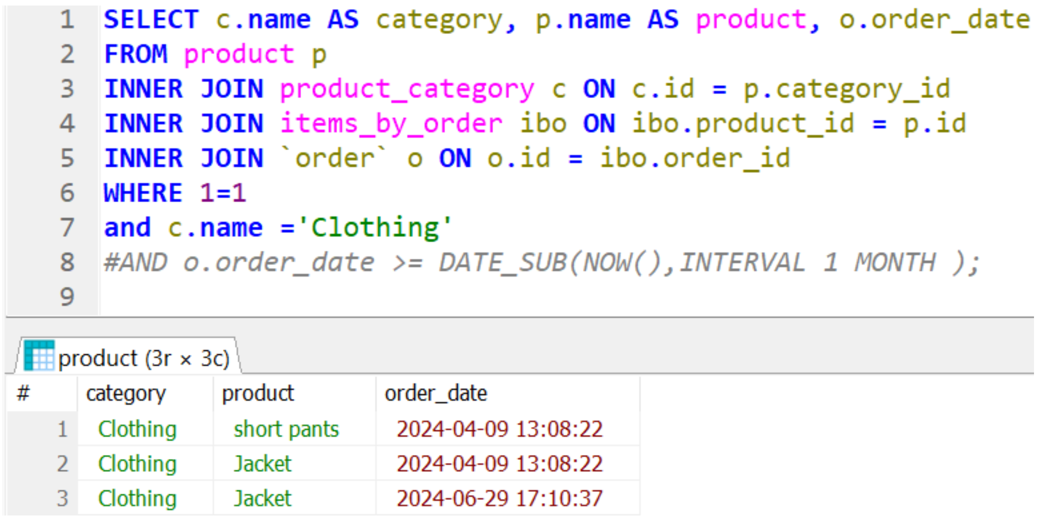
This query retrieves information about returned items, including the returned_item's ID, item_by_order's ID, the product's name, the number of ordered and returned itmes, Return's Date and the reason. It uses JOIN operations on the Returns, items_by_order, and Products tables to gather relevant data 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.

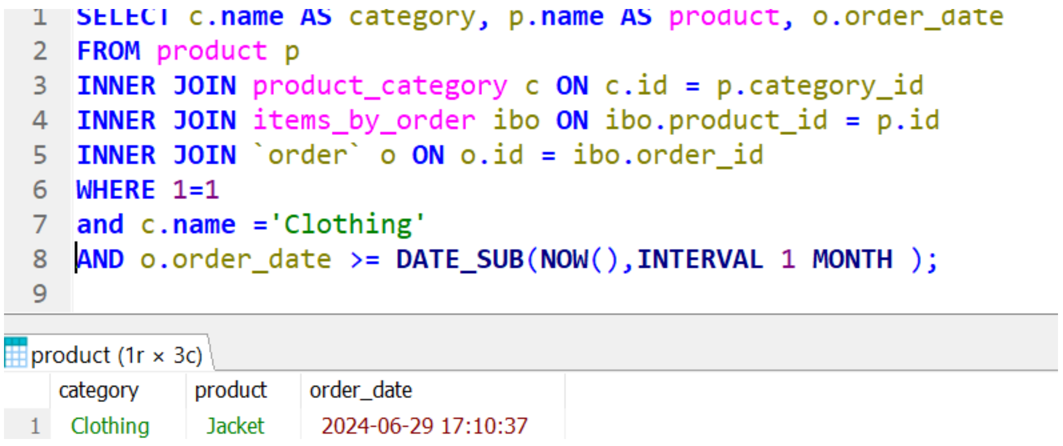


```
SELECT c.name AS category, p.name AS product, o.order_date
FROM product p
INNER JOIN product_category c ON c.id = p.category_id
INNER JOIN items_by_order ibo ON ibo.product_id = p.id
INNER JOIN `order` o ON o.id = ibo.order_id
WHERE 1=1
and c.name ='Clothing'
#AND o.order_date >= DATE_SUB(NOW(),INTERVAL 1 MONTH );
```

product (3r × 3c)

#	category	product	order_date
1	Clothing	short pants	2024-04-09 13:08:22
2	Clothing	Jacket	2024-04-09 13:08:22
3	Clothing	Jacket	2024-06-29 17:10:37

With the order date condition



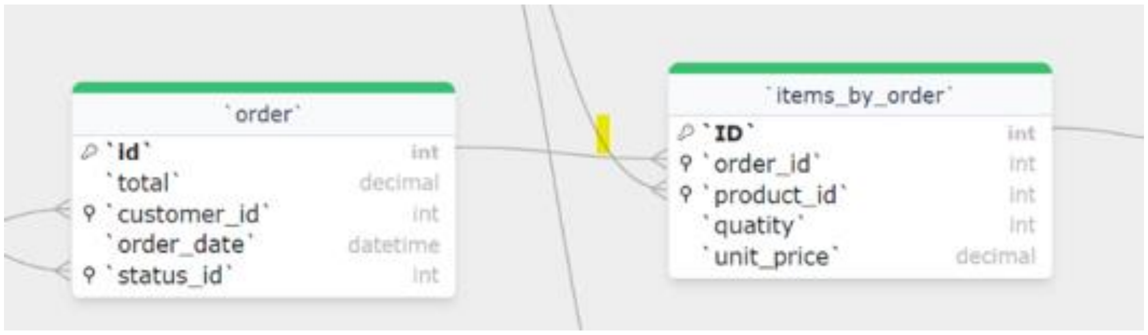
```sql
SELECT c.name AS category, p.name AS product, o.order_date
FROM product p
INNER JOIN product_category c ON c.id = p.category_id
INNER JOIN items_by_order ibo ON ibo.product_id = p.id
INNER JOIN `order` o ON o.id = ibo.order_id
WHERE 1=1
and c.name ='Clothing'
AND o.order_date >= DATE_SUB(NOW(),INTERVAL 1 MONTH );
```

product (1r × 3c)

	category	product	order_date
1	Clothing	Jacket	2024-06-29 17:10:37

STORE PROCEDURE AND TRIGGERS

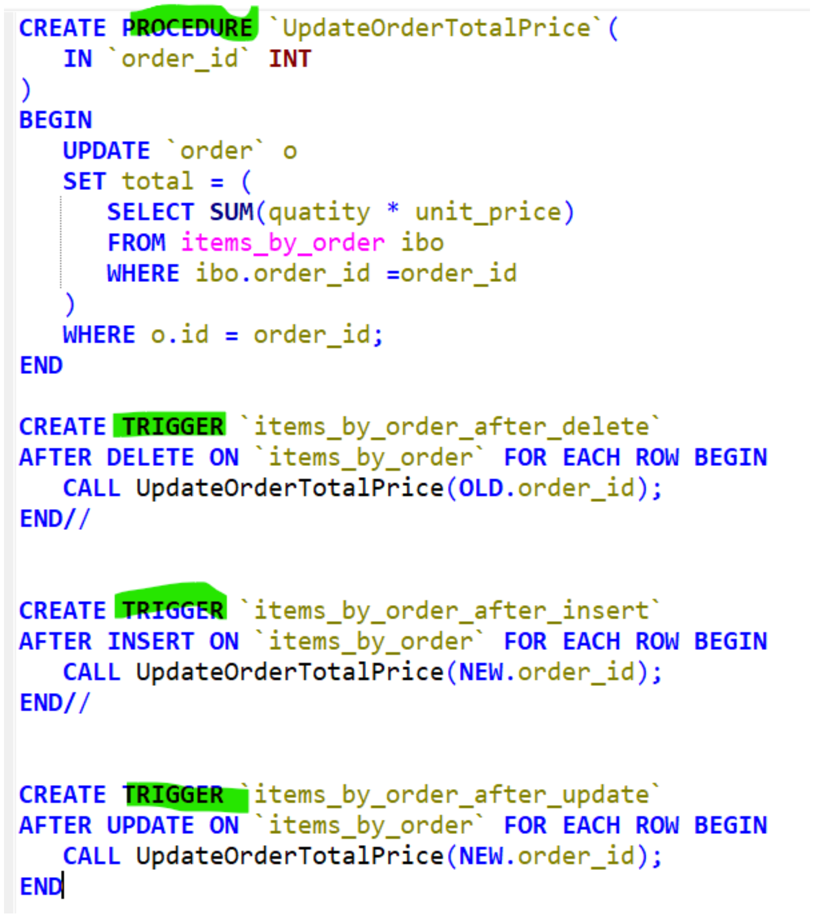
As can be seen in the model there is a relationship between the tables *order* and *items_by_order.* Since the table *order* stores the general information of an order, but the details of the purchased products must be stored in the table *items_by_order*. So, to update the total value of an order, using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



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.

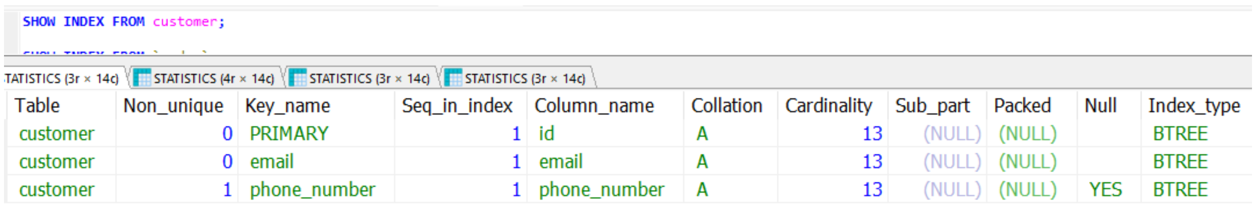
For the proposed model, the following objects were implemented. Consisted in one stored procedure and three triggers. Those objects make sure that whenever an item is deleted, inserted or updated into the table *item_by_order,* the total price for the order to belongs to, will be automatically calculated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



```sql
CREATE PROCEDURE `UpdateOrderTotalPrice`(
   IN `order_id` INT
)
BEGIN
   UPDATE `order` o
   SET total = (
      SELECT SUM(quatity * unit_price)
      FROM items_by_order ibo
      WHERE ibo.order_id =order_id
   )
   WHERE o.id = order_id;
END

CREATE TRIGGER `items_by_order_after_delete`
AFTER DELETE ON `items_by_order` FOR EACH ROW BEGIN
   CALL UpdateOrderTotalPrice(OLD.order_id);
END//


CREATE TRIGGER `items_by_order_after_insert`
AFTER INSERT ON `items_by_order` FOR EACH ROW BEGIN
   CALL UpdateOrderTotalPrice(NEW.order_id);
END//


CREATE TRIGGER `items_by_order_after_update`
AFTER UPDATE ON `items_by_order` FOR EACH ROW BEGIN
   CALL UpdateOrderTotalPrice(NEW.order_id);
END
```

INDEXES

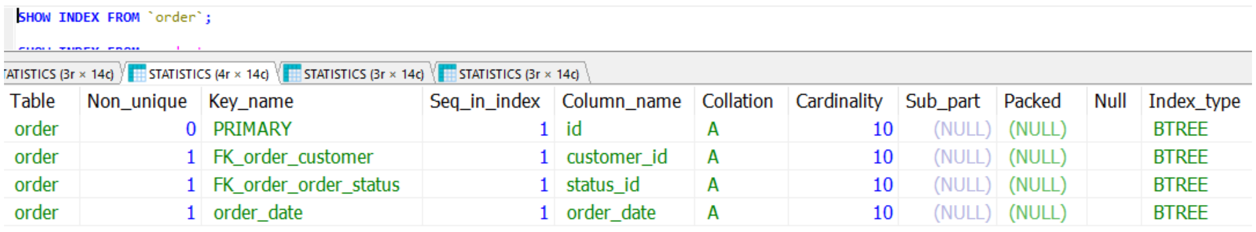
Considering that the business tables will increase their sizes as time will come, a set of indexes will help to improve the performance of queries that search or filter based on specific columns, such as *email* and *phone* in the *Customers* tables, since will be important for *ABC Online Store* locate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



```sql
SHOW INDEX FROM customer;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
customer	0	PRIMARY	1	id	A	13	(NULL)	(NULL)		BTREE
customer	0	email	1	email	A	13	(NULL)	(NULL)		BTREE
customer	1	phone_number	1	phone_number	A	13	(NULL)	(NULL)	YES	BTREE

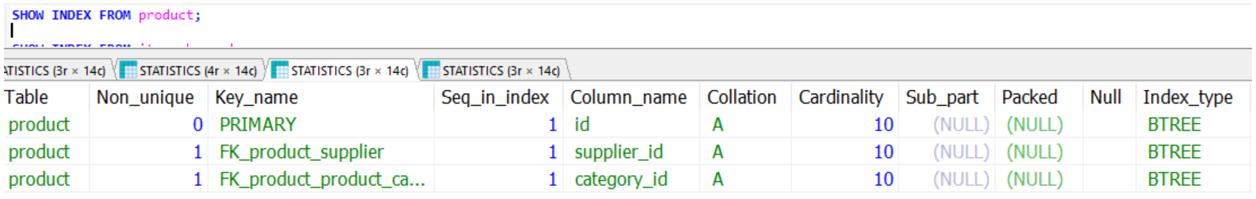
Besides of the default indexes (primary and foreign keys indexes), a index on the *order_date* field of the table *order,* will help to the store locate the orders in a date range because this type of index can be use with the most common comparison operators in this type of field.



```sql
SHOW INDEX FROM `order`;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
order	0	PRIMARY	1	id	A	10	(NULL)	(NULL)		BTREE
order	1	FK_order_customer	1	customer_id	A	10	(NULL)	(NULL)		BTREE
order	1	FK_order_order_status	1	status_id	A	10	(NULL)	(NULL)		BTREE
order	1	order_date	1	order_date	A	10	(NULL)	(NULL)		BTREE

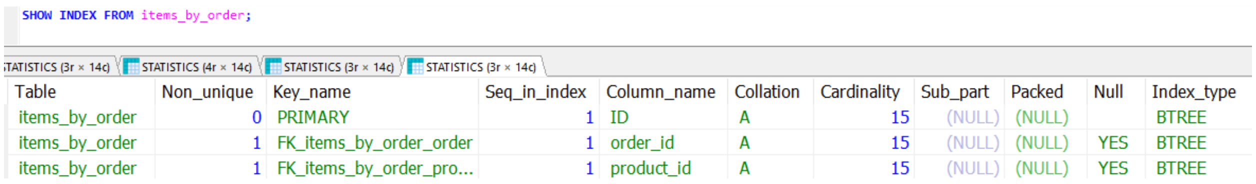
Indexes in the *product* table for the fields *supplier_id* and *category_id* will let the store classify the products either by supplier or category. These indexes were created by default by the RDBMS.



```sql
SHOW INDEX FROM product;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
product	0	PRIMARY	1	id	A	10	(NULL)	(NULL)		BTREE
product	1	FK_product_supplier	1	supplier_id	A	10	(NULL)	(NULL)		BTREE
product	1	FK_product_product_ca...	1	category_id	A	10	(NULL)	(NULL)		BTREE

A similar case applies to the table *items_by_order.*



```sql
SHOW INDEX FROM items_by_order;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type
items_by_order	0	PRIMARY	1	ID	A	15	(NULL)	(NULL)		BTREE
items_by_order	1	FK_items_by_order_order	1	order_id	A	15	(NULL)	(NULL)	YES	BTREE
items_by_order	1	FK_items_by_order_pro...	1	product_id	A	15	(NULL)	(NULL)	YES	BTREE

,

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.