



VRIJE  
UNIVERSITEIT  
BRUSSEL



# E-COMMERCE PROJECT

Open Information Systems

Ardavan Khalij, Amadou Sarjo Jallow, Brenda Ordoñez Lujan,  
Florent Nicolas J Grimaud, and Milan Pavle Ilić

26 December 2021

Academic year: 2021 - 2022

Student numbers: Ardavan 05857061, Amadou 0532989, Brenda 0571129,  
Florent 0591865, Milan 0545802

Email-addresses: ardavan.khalij@vub.be, amadou.sarjo.jallow@vub.be,  
brenda.ordonez.lujan@vub.be, florent.nicolas.j.grimaud@vub.be,  
milan.pavle.ili@vub.be

**1st master computer science**

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Modules . . . . .	2
1.2	Features and data . . . . .	2
1.2.1	Products . . . . .	2
1.2.2	Warehouses . . . . .	3
1.2.3	Costumers . . . . .	4
1.2.4	Suppliers . . . . .	4
1.2.5	Delivery services . . . . .	5
1.3	Work division . . . . .	5
<b>2</b>	<b>Ontology</b>	<b>6</b>
2.1	Products . . . . .	6
2.1.1	Naming differences . . . . .	6
2.1.2	Design and structure differences . . . . .	6
2.2	Warehouses . . . . .	6
2.2.1	Naming differences . . . . .	6
2.2.2	Design and structure differences . . . . .	7
2.3	Costumers . . . . .	7
2.3.1	Naming differences . . . . .	7
2.3.2	Design and structure differences . . . . .	7
2.4	Suppliers . . . . .	7
2.4.1	Naming differences . . . . .	8
2.4.2	Design and structure differences . . . . .	8
2.5	Delivery services . . . . .	8
2.5.1	Naming Difference . . . . .	8
2.5.2	Design and structure differences . . . . .	9
<b>3</b>	<b>Mapping</b>	<b>9</b>
3.1	Design decisions . . . . .	9
3.2	Complications . . . . .	9
<b>4</b>	<b>Queries</b>	<b>10</b>
4.1	Products: Product information . . . . .	10
4.2	Warehouses: Stock quantity sorted with extra information . . . . .	11
4.3	Costumers: Users with at least one complete address . . . . .	12
4.4	Costumers: Customers with orders and the total amount of their orders . . . . .	13
4.5	Suppliers: Products with higher profit margin . . . . .	14
<b>5</b>	<b>Discussion</b>	<b>14</b>

# 1 Introduction

The goal of this project is to design and implement an e-commerce platform. The backbone of this platform is an information system. This system consists out of 5 modules: Products, Warehouses, Costumers, Suppliers and Delivery Services.

Each person was assigned to one module. Firstly we started thinking about the information needs of the database users by creating features and an ER diagram<sup>1</sup> for a module. Once this was achieved, each person created a database with queries, based on the created ER diagram, in standard SQL<sup>2</sup>. We then created one information system by combining the SQL databases. After that, everyone created an ontology, using Protoge, for their module. Then we combined the ontologies to create one information system. Then everyone created a mapping using the R2RML<sup>3</sup> mapping language. Again, we combined our mappings to generate one mapping that represented the whole system. Finally everyone created one SPARQL<sup>4</sup> query, based on one of their features. This SPARQL query was tested using the Ontop system. Ontop translates SPARQL queries into SQL queries, and relies on R2RML mappings [1].

## 1.1 Modules

As metioned before, this project consists of 5 modules:

1. The module Product is about the physical product itself and the properties it has.
2. The module Warehouses is primarily about the management of an inventory.
3. The module Costumers is concerned with reviews, payments, and other things regarding an online costumer.
4. The module Suppliers is primarily about a supplier that supplies to a warehouse.
5. The module Delivery services is about delivering the product from a warehouse to a costumer.

## 1.2 Features and data

### 1.2.1 Products

1. Orders placed: An admin might want to know how many orders have been made on the platform up till date, they may want to know those that were successfully Fulfilled, the number of orders that were cancelled and by which customers etc. A costumer might want to have a history of their orders including the amount spent on each order, the total orders they have placed, basically their order statistics. A supplier might want to know their highest selling products, the amount they have made so far on the platform, their average orders per month and other similar metrics etc. Data used: product/products name or information an order is placed on, the time the order was placed, the order status, the order amount, the delivery service, information about the customer who placed the order and the order state i.e., if the order has been cancelled or not etc.

---

<sup>1</sup>Entity-Relation diagram

<sup>2</sup>Structured Query Language

<sup>3</sup>RDB to RDF Mapping Language

<sup>4</sup>SPARQL Protocol and RDF Query Language

2. Customer cart: An admin will want to send reminders to customers about the products in their shopping cart so that they can purchase them before they are no longer in stock, they may also want to keep a record of the products customers are mostly interested in or at least adding to their shopping carts etc. A customer adds products they want to buy in their shopping cart for later purchase, after browsing through different products or when they have enough money. They may also want to view the items in their shopping cart. A supplier might want to know their highest selling products, the amount they have made so far on the platform, their average orders per month etc. Data used: product or products added to a cart, the time the products were added, the cost of each product etc.
3. Profitable locations: An admin might want to know which locations are the highest selling so that more products will be moved to the warehouses or fulfillment centers in those locations and minimize moving products to least selling locations. Data used: products or product names, product locations, the number of orders that were placed on those products.
4. Rated products: A customer wants to know which products are highly rated to help them in their purchase decisions and avoid those products from suppliers with low ratings or bad reviews. An admin wants to know the highest rated products to recommend to other customers and to find the suppliers with lowly rated products and bad reviews to help filter out suppliers abusing the platform and as such, help retain and attract more customers on the platform. Data used: product name or information, Supplier information, rating values to help find the average rating.
5. Stock quantity: A supplier might want to know the number of items they have in stock for a particular product or if they no longer have a product in stock to increase its quantity. They may also be interested in knowing how fast a particular product is selling. An admin similarly, may also be interested in knowing the number of items they have in stock for a particular product or if they no longer have a product in stock to increase its quantity. They may also be interested in knowing how fast a particular product is selling. Data used: product name or information, stock quantity value, date added.

### 1.2.2 Warehouses

1. Product quantity: An employee of the inventory management wants to find out how much of a certain product there is left in stock. An online customer wants to know if the product is available for purchase. Data used: amount of stock of a certain product.
2. Warehouse incoming and outgoing products: An employee wants to know how many products will arrive/departure on a certain day. An accountant wants to keep track of all the arriving and departing products. Data used: all the incoming products from the supplier and all the outgoing products of the delivery service.
3. Delivery service delivery time: An employee from customer service is asked if the product has already left the warehouse. A customer wants to know how long it will take for his package to arrive. Data used: details about the delivery service.
4. Supplier product delivery time: An employee of a warehouse wants to know when a product from the supplier arrives. A customer wants to know how long it will take for his package to arrive. Data used: details about the supplier.
5. Company information: An employee of a warehouse wants to contact a delivery service or supplier. A delivery service or supplier needs to deliver/pick up a package from the warehouse. Data used: information about a warehouse, delivery service and supplier.

### 1.2.3 Costumers

1. Total price of each receipt of the user plus cost of delivery: I should mention that the field of price has been deleted from the Order description by the team, so the product also needs to be in the data. Users can see all of their orders the total price of each purchase. Data used: information about customers, orders, order descriptions, delivery services, and products.
2. Delivery date of all receipts for a user: I should mention that the original feature was the delivery date of all paid receipts for a user, but we deleted the payment status of the orders as a team. Each user can see the date of each of their purchases. Data used: information about customers, orders, and delivery services.
3. See the not delivered orders: Each user can see their not delivered orders. Data used: information about customers and orders.
4. Edit customer information: I should mention that customers' payment cards have been added to the tables by the group, so the payment table is also in this feature. Customers can edit their information at any time. Data used: information about customers, customer's addresses, and customer's payment cards.
5. Showing all the users' personal information: I should mention that customers' payment cards have been added to the tables by the group, so the payment table is also in this feature. Customers can have access to their information at any time. Data used: information about customers, customer's addresses, and customer's payment cards.

### 1.2.4 Suppliers

1. Response time per supplier: An employee of the logistics area needs the average response time of an order to know when he needs to fill his stock if required. A manager of the logistics area needs it to know the performance of the supplier and to be able to follow up on it. Data used: Expressed in days and hours, it is calculated with the following operation. Date the order was placed - Date the order was delivered.
2. The supplier that gives more facilities: A manager of the logistics area needs it to make the decision to reduce costs if it's possible or choose between suppliers in the future. Data used: For this feature, the count number and the average of the discount for prompt payment per supplier is needed as a field expressed in percentages, the discount is entered when the order is being placed, if the supplier offers it.
3. Maximum payment flexibility: An employee of the logistics area needs it to make budget plans. A manager of the logistics area needs it to make decisions such as changing the payday if necessary or choose between suppliers in the future. Data used: Expressed in days. It is the maximum of the limit days to pay that gives the supplier.
4. Minimum cost of the most product requested: An employee of the logistics area needs it to calculate the necessary budget and predict the days that needs to place orders. A manager of the logistics area needs it to compare with other prices on the market and make decisions between new suppliers. Data used: Expressed in euros, it is minimum cost of the wholesale price of the product that is being ordered the most.

5. Higher and lower delivery costs per supplier: An employee in the logistics area can reduce transportation costs from that value. Analyze whether it is convenient to have a transportation service and present it to the manager as an alternative in the future. It is also an important factor in choosing between providers. Data used: Expressed in euros, it is maximum cost of the delivery cost per supplier.

### 1.2.5 Delivery services

1. Real-time package Tracking: A costumer wants to track their order in real time. Data used: The API key linked to the transport company handling the package the user wants to track.
2. Delivery Time Estimation by packet: A delivery service wants to give the costumer an idea when the package will arrive. Data used: The address registered in the packet's order, the packet's weight and size, the amount of trucks available in the company able to deliver a packet to the given address with the given weight and size and the location of the warehouse where the packet is gathered.
3. Delivery Price Estimation: Delivery service wants to request an extra fee to the costumer. Data used: The address registered in the packet's order, the packet's weight and size, the price per km of the company able to deliver the package (As explained in the previous feature).
4. Average amount of delivery trip made by each delivery companies in all the warehouses: A delivery service wants to track with which companies it is worth having delivery contracts, based on the amount of delivery trips they make for the company. Data used: he amount of delivery trip that occurred in every warehouse in the last X months sorted by the vehicle's company.
5. Average Packet loss during delivery trips sorted by company: A delivery service wants to increase its trust. Data used: The average status of the packets in the deliveries of company X.

## 1.3 Work division

We always started by letting everyone implement their own module and after that a person combined the implementations to make one information system. This was also true for the report, where everyone wrote the parts which had to do with their module.

Ardavan Khalij: Costumer database and queries, Costumer ontology, Costumer mapping, 2 costumer SPARQL queries, ...

Amadou Sarjo Jallow: Product database and queries, Product ontology, Product mapping, SPARQL query, ...

Brenda Ordoñez Lujan: Supplier database and queries, Supplier ontology, Supplier mapping, SPARQL query, ...

Florent Nicolas J Grimaud: Delivery service database and queries, Delivery service ontology, Delivery service mapping, SPARQL query, ...

Milan Pavle Ilic: Warehouses database and queries, Warehouse ontology, Warehouse mapping, 1 warehouse SPARQL query, Report introduction and making the L<sup>A</sup>T<sub>E</sub>X document.

## 2 Ontology

In this project, we used Protege for making our ontology. The main reason was that it was easy to work with.

### 2.1 Products

This module describes the products that are advertised on the platform and all the relevant information about the product including its category, brand etc. It is the central aspect of the platform and as such, it connects to other modules. It is connected with the customers module through the cart, reviews and orders modules. It is connected with the warehouse module through the outbound inventory. A customer carts a product through an object property carts. the object relation Belongs\_To has domain product and range category. The Admin\_advertises and Supplier\_advertises object properties connect a product with a seller, which can be the platform itself, in this case call admin or a supplier who just wants to advertise on our platform.

#### 2.1.1 Naming differences

There were little naming difference made to the product and related modules. The primary key of product was changed to sku for better representation and meaning. Apart from that, all other connected modules remain the same with few field deletions which made little or no difference at all. orders and order\_description tables are same as the receipt\_information and goods\_in\_receipt relations.

#### 2.1.2 Design and structure differences

The products module and its connected modules remain fairly the same. An address field is added to the orders module to reference the delivery address of the customer. Also the primary key of the admin relation was changed from name field to email field. Also orders, cart and reviews had foreign keys pointing to the email of the customer rather than a unique uuid to reflect what is used in the customers module.

### 2.2 Warehouses

This module describes a warehouse with attributes such as: companyname, warehousecode, email, etc. A warehouse has an object property warehouse\_has. This means that a warehouse has multiple inventories. An inventory contains a product, a quantity and the warehousecode corresponding to the warehouse it belongs to. To achieve this, an inventory has an object property inventory\_has.

#### 2.2.1 Naming differences

Amount in stock was renamed to quantity. Product id was renamed to productsku. Telephone number was renamed to simply telephone because it was clear enough. Name was changed to companyname to make it more clear that it is the name of a company.

### 2.2.2 Design and structure differences

The original tables of the warehouse module were a little complicated. Therefore, we reduced the initial warehouses' idea to be more minimalist because it was easier to be compatible with the other modules. The ISA business relation was removed because it would overcomplicate the database. Employee was removed because it was not needed. Delivery service and shipment were removed because these were implemented in a better way by the delivery services module. Product was also removed because implemented in a more accurate way in the product module. Inventory was added to keep track of how many products a warehouse has in store. Finally, a couple of attributes are added to the warehouse to make it more complete.

## 2.3 Costumers

The customer module has information about customers, such as payment cards, addresses, and other information. It also has a part for saving the orders of the customers. Each order has an order description that lists the products in an order. There are also a delivery kind table and delivery status in this part that has the information about the delivery types and delivery status. Each customer has addresses and payment cards, which are connected with Has.Address and Has.Payment.Card object properties in the ontology. Also, each customer has some orders, and each order has an order description available in ontology by places\_order and Has.Order.Description object properties. Orders also have a delivery type available in ontology with the Has.The.Delivery.Type object property. There were some different opinions and differences in the databases' design and choosing the names for designing the ontology.

### 2.3.1 Naming differences

The main difference was about the name of the classes. We changed the name of the class receipt to understand ontology better. Also, goods\_in\_receipt changed to order description.

### 2.3.2 Design and structure differences

The first difference was the receipt in the customer database and order in the product database. The concept of order and cart was merged in the customer database, but there were differences in the product database, so we decided to use the separate concept instead. Moreover, as a result, the payment status column has been deleted from the receipt information table. The second issue was the customer's payment information. So we added the payment information of customers as well. Also, we deleted the column price from the order description because it already existed in the product.

## 2.4 Suppliers

This module allows storing information of the suppliers that use the e-commerce platform as a service to advertise their products. The elements that were considered most important are the company's registration number (bce), name, url, telephone, postal code, mail and address. Each supplier sends its products to a warehouse to be registered within the platform. This property is identified as "supplies" and is\_stored\_in within the ontology. supplies\_has was developed so that a supplier can ship more than one product. These features allow us to take advantage of the track of businesses between a supplier and its products.



### 2.4.1 Naming differences

At the beginning, the purchase orders to suppliers was considered as a company that develop the product and put it on sale. But for the ontology this property was not considered since it was coordinated that the platform will only be an intermediary service between the supplier and the client; so the table of purchase orders developed became a table of supplies. The difference is that order dates, invoices, payments, among other aspects, are no longer considered. Only the date on which the product arrives at the warehouse, the quantity and the price of the product are considered.

Also the supplier contact table was not considered, it was summarized in a phone field within the supplier table. This agreement was reached because we were only interested in contacting the supplier and not in knowing how the supplier was contacted.

### 2.4.2 Design and structure differences

To adapt it to the ontology, a product code field was added to the product table, since initially there was a natural key with two attributes: category and name. To adapt it to the product code, both attributes were combined without considering spaces up to 30 characters.

The major change was to adapt the order table to be a supplies table. To do this, two views were created considering only the necessary data (date of entry, quantity, price and product), there were no problems to reconstruct the information.

## 2.5 Delivery services

The delivery Services modules holds information about the delivery services such as the transport vehicles, transport companies, and delivery trips.

The transport companies represent the available companies for transportation. They have informations such as their name, prices, the zones they are able to deliver, and an API key, that could be used to retrieve information such as the position of each vehicle, and data about the price for a hypothetical delivery.

The transport vehicles represents vehicles owned by a certain company, and their position. This last field is to be seen as a cache, to avoid calling the API of the company numerous amount of times (As the API may not be free). It is linked to delivery trips, and thus, allow us to potentially compute the total load of the vehicle.

The delivery trip represent the trip of multiple orders. It holds information about the starting time of the trip and the end time, to allow the computation of the total time of the trip, and help estimate the future duration of trips. It is linked to the orders it transports, and the warehouse it starts from. This last link could be used to find a route, and thus a duration, for the truck.

### 2.5.1 Naming Difference

The naming differences are mainly found in the attributes of each class, for a better understanding of the ontology. The final database use the names that are in the initial ERD.

### 2.5.2 Design and structure differences

In the new design of the database, we removed the Package class, as it was redundant with the order in itself. The idea at the start was to use the Package class to retain information about what could be just a part of the total order, that could be shipped independently from the rest of the order. It was decided to simplify this part, and we chose that an order should be sent in a single delivery. The status attribute of the Package class was moved to the order class.

## 3 Mapping

The tools used to convert data from different databases to a single RDF document are:

1. R2RML: the language used for mapping our datasets to RDF datasets.
2. Dremio: the platform used to unify the tables in a single space called ecommerce.
3. Ontop: the environment used for test SPARQL features.

### 3.1 Design decisions

The namespace prefix that links to our ontology is: @prefix ecommerce: <http://data.ecommerce#>

Each member generated their triple maps for R2RML mapping. All the tables were applicable. The same structure was used for the template, to make it easier to unify it into one.

The template structure used for the classes is the following:  
"http://data.ecommerce/[entityName]#dataProperty".

The template structure used for the objects properties is the following:  
"http://data.ecommerce[entityName]#dataProperty\_dataProperty".

To overcome the process of having data linking to each other in the different databases, it was decided to map all the tables that correspond to the ontology created. Products being the entity that had to be mapped from each database as it is one of the most important for this platform.

### 3.2 Complications

1. Because the "ID" data was repeated in different databases, there were issues with the "order" class. In this example, we only mapped from one database and made entries in that database for the other orders to ensure they had data that linked.
2. Not all object properties could be mapped. Ontop endpoint did not give errors but at the moment of materializing the triples, an error was generated by the SQL. All queries were always tested in pgadmin to reduce the error; but there was not much detail about the error. The affected object properties were: "Supplier\_advertises", "Admin\_advertises" and "places\_order" that correspond to the ontology.

## 4 Queries

We came up with 5 SPARQL queries that query data across our modules.

### 4.1 Products: Product information

A query to retrieve the products and relevant information and ordering them according to their price in ascending order. This is a useful query for customers when shopping to help filter through cheap products and also for admins when recommending cheap products to customers.

```
PREFIX ecom: <http://data.ecommerce#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?product ?price ?quantity ?color ?model ?product_location ?
      date_added ?description WHERE {
?product a ecom:Product .
?product ecom:Product_price ?price .
?product ecom:Product_date_added ?date_added .
  OPTIONAL {?product ecom:model ?model}.
  OPTIONAL {?product ecom:location ?product_location}.
  OPTIONAL {?product ecom:color ?color}.
  OPTIONAL {?product ecom:stockQuantity ?quantity}.
?product ecom:Product_description ?description
} ORDER BY ?price
```

product	price	quantity	color	model	product_location	date_added	description
1 <http://data.econmer...	"300"	"100"	Vintage...	Redmi Note ...	Ghent	"2021-06-14T06:41:19Z"	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam id dictum purus. Vestibulum ni...
2 <http://data.econmer...	"300"	"100"	Vintage...	Redmi Note ...	Brussels	"2021-06-14T06:41:19Z"	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam id dictum purus. Vestibulum ni...
3 <http://data.econmer...	"400"	"85"	White	MWP22HN/A	California	"2021-10-09T05:08:07Z"	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam id dictum purus. Vestibulum ni...
4 <http://data.econmer...	"450"	"85"	Black	MBP22HN/A	California	"2021-10-11T04:51:12Z"	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam id dictum purus. Vestibulum ni...
5 <http://data.econmer...	"450"	"85"	White	IPhone 11	Brussels	"2021-09-19T04:51:12Z"	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam id dictum purus. Vestibulum ni...
6 <http://data.econmer...	"700"	"154"	Transpa...	VivoBook 1...	Brussels	"2021-10-06T06:41:19Z"	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam id dictum purus. Vestibulum ni...
7 <http://data.econmer...	"700"	"104"	Eclipse ...	ROG Strix ...	Brussels	"2021-07-16T06:41:19Z"	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam id dictum purus. Vestibulum ni...
8 <http://data.econmer...	"700"	"60"	White	IPhone 10	California	"2021-10-09T04:53:01Z"	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam id dictum purus. Vestibulum ni...
9 <http://data.econmer...	"950"	"84"	Black	MBP22HN/A	Brussels	"2021-10-11T04:51:12Z"	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam id dictum purus. Vestibulum ni...

Figure 1: Query output

## 4.2 Warehouses: Stock quantity sorted with extra information

A warehouse has multiple products, all with a quantity. One might want to see the products in the warehouse ordered by ascending order, to see which products are low in stock, or even out of stock. To make it more readable for a human, we also display the name of the product. Suppose that we need information about the 'DHLCONSOLIC' warehouse in this SPARQL query.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ecommerce: <http://data.ecommerce#>

SELECT ?quantity ?productname ?productsku ?warehousename WHERE{
  ?warehouse a ecommerce:Warehouse .
  ?warehouse ecommerce:warehouse_has ?inventory .
  ?warehouse ecommerce:Warehouse_companyName ?warehousename .
  ?inventory a ecommerce:inventory .
  ?inventory ecommerce:InventoryQuantity ?quantity .
  ?inventory ecommerce:productsku_fk ?productsku .
  ?inventory ecommerce:inventory_has ?product .
  ?product a ecommerce:Product .
  ?product ecommerce:Product_description ?productname .
  FILTER langMatches(?warehousename, "DHLCONSOLIC") .
} ORDER BY ?quantity
```

Table		Response	3 results in 0.015 seconds	
	quantity	productname	productsku	warehousename
1	"3"^^<http://www.w3.org/2001/XMLSchema#decimal>	Shrek3	74849305	DHLCONSOLIC
2	"93"^^<http://www.w3.org/2001/XMLSchema#decimal>	ColonistsOfCatan	432890572	DHLCONSOLIC
3	"402"^^<http://www.w3.org/2001/XMLSchema#decimal>	Shrek2	492354322345	DHLCONSOLIC

Showing 1 to 3 of 3 entries

Figure 2: Query output

### 4.3 Costumers: Users with at least one complete address

In a platform such as e-commerce, the admin should check and control customers' data. For handling this information, a query is available that gives the admin all the customers with at least one complete address. This can prevent errors such as a customer placing an order but doesn't have an address. This can also help the admin to manage the customers. This SPARQL query uses customers' information and their addresses. And it is also ordered by the first name of the customers. So the result contains the first name, last name, city address, city, and country of the customer.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ecommerce: http://data.ecommerce#
SELECT ?first_name ?last_name ?cityaddress ?city ?country WHERE{
    ?customer ecommerce:first_name ?first_name;
                ecommerce:last_name ?last_name;
                ecommerce:Customer_email ?email.
    ?address ecommerce:Customer_email ?email;
                ecommerce:Address_address ?cityaddress;
                ecommerce:Address_country ?city;
                ecommerce:Address_city ?country.
} Order by ?first_name
```



The screenshot shows a web interface for displaying query results. At the top, there are tabs for 'Table', 'Response', 'Pivot Table', 'Google Chart', and 'Geo'. The 'Table' tab is selected. Below the tabs, it says 'Showing 1 to 8 of 8 entries (in 3.174 seconds)'. On the right, there is a search bar and a 'Show 50 entries' dropdown. The table has five columns: 'first\_name', 'last\_name', 'cityaddress', 'city', and 'country'. There are 8 rows of data, numbered 1 to 8. The data includes customers like Ardavan, Harry, Monica, Rick, Ross, Ted, and Tony, with their respective last names, addresses, cities, and countries.

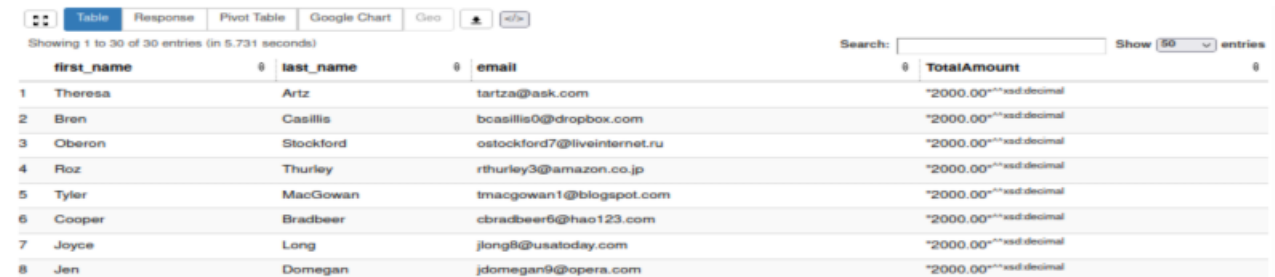
	first_name	last_name	cityaddress	city	country
1	Ardavan	Khalij	Dibaji 10	Iran	Tehran
2	Ardavan	Khalij	Kroonlaan 238	Belgium	Brussels
3	Harry	Potter	Privet Drive 4	UK	London
4	Monica	Geller	Manhattan, 223, 3	USA	New York
5	Rick	Sanchez	2, 10	USA	Seattle
6	Ross	Geller	Manhattan, 233, 23	USA	New York
7	Ted	Mosby	Manhattan, 180, 44	USA	New York
8	Tony	Stark	Manhattan, Stark Tower	USA	New York

Figure 3: Query output

#### 4.4 Costumers: Customers with orders and the total amount of their orders

In a platform such as e-commerce, the admin should be able to access all the order prices. This can help the admin to understand the income of the company. Customers also can access their orders to see the history of their purchases. This can always allow the customers to see what they have bought and re-order the products they want. This SPARQL uses the customers' information and the order information. And the result is also ordered by the total amount of each order. This can help the admin see which customers or customers with which home countries have the most considerable purchase amount per order. The result contains first name, last name, email, and the total amount of an order of the customer.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ecommerce: <http://data.ecommerce#>
SELECT ?first_name ?last_name ?email ?TotalAmount WHERE{
    ?customer ecommerce:places_order ?order.
    ?customer ecommerce:first_name ?first_name;
                                ecommerce:last_name ?last_name;
                                ecommerce:Customer_email ?email.
    ?order ecommerce:orderAmount ?TotalAmount.
} ORDER BY ?TotalAmount
```



The screenshot shows a web interface for displaying SPARQL query results. At the top, there are tabs for 'Table', 'Response', 'Pivot Table', 'Google Chart', and 'Geo'. Below the tabs, it says 'Showing 1 to 30 of 30 entries (in 5.731 seconds)'. On the right, there is a search bar and a 'Show 50 entries' dropdown. The main content is a table with four columns: 'first\_name', 'last\_name', 'email', and 'TotalAmount'. The table contains 8 rows of data, each representing a customer's order.

	first_name	last_name	email	TotalAmount
1	Theresa	Artz	tartza@ask.com	"2000.00""xsd:decimal
2	Bren	Casillis	bcasillis0@dropbox.com	"2000.00""xsd:decimal
3	Oberon	Stockford	ostockford7@liveinternet.ru	"2000.00""xsd:decimal
4	Roz	Thurley	rthurley3@amazon.co.jp	"2000.00""xsd:decimal
5	Tyler	MacGowan	tmacgowan1@blogspot.com	"2000.00""xsd:decimal
6	Cooper	Bradbeer	cbradbeer6@hao123.com	"2000.00""xsd:decimal
7	Joyce	Long	jlong8@usatoday.com	"2000.00""xsd:decimal
8	Jen	Domegan	jdomegan9@opera.com	"2000.00""xsd:decimal

Figure 4: Query output

## 4.5 Suppliers: Products with higher profit margin

A platform such as e-commerce charges a percentage to a supplier for providing its service, in this case as an intermediary between its product and the customer. But the profit per product will depend on the price that the supplier sets. That is why it is necessary to know which products provide more profit to the platform to be able to make disguised biased advertising. As the price per product is not fixed, due to seasonal discounts that may exist, the average profit per product is disclosed, in this case the top 10 products with the highest profit margin. This decision may help the platform to have higher incomes.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX ecommerce: <http://data.ecommerce#>
SELECT ?bce_company ?company ?product_code ?product_description (AVG
  (?price2) - AVG(?price1) AS ?profit)
WHERE {
  ?supplier ecommerce:supplies ?supplies .
  ?supplier ecommerce:Supplierbce ?bce_company ;
    ecommerce:SuppliercompanyName ?company .
  ?supplies ecommerce:supplies_has ?supply_stock .
  ?supply_stock ecommerce:has_contains ?product .
  ?supply_stock ecommerce:suppliesOrderPrice ?price1 .
  ?product ecommerce:sku ?product_code ;
    ecommerce:Product_description ?product_description ;
    ecommerce:price ?price2 .
} GROUP BY ?bce_company ?company ?product_code ?product_description
ORDER BY DESC(?profit) LIMIT 10
```

	bce_company	company	product_code	product_description	profit
1	5079289663	Brown-Medhurst	GROpork-Tenderloin,Frozen	Pork - Tenderloin, Frozen	*1.21**xsd:decimal
2	5618681673	Hyatt, Tromp and Anderson	GROCapers-Pickled	Capers - Pickled	*1.2033333333333331**xsd:decimal
3	5137904499	Gleason Group	GROGuineaFowl	Guinea Fowl	*1.1999999999999993**xsd:decimal
4	2803167613	Haley-Casper	GROBuffalo-Tenderloin	Buffalo - Tenderloin	*1.1599999999999993**xsd:decimal
5	2711492855	Gerhold, Rolfson and Schmidt	GROsausage-Liver	Sausage - Liver	*1.1433333333333344**xsd:decimal
6	3472355586	Franecki-Durgan	GROSwissChard-Red	Swiss Chard - Red	*1.1150000000000002**xsd:decimal
7	8444388162	Brown, Monahan and Walker	HOMGranitev2	Granite v2	*1.1100000000000003**xsd:decimal
8	4266613467	Baumbach, Bernier and Jones	GROMonkfish-Fresh	Monkfish - Fresh	*1.1099999999999994**xsd:decimal
9	4801688228	Trantow-Legros	GROSoup-Campbells-Chicken	Soup - Campbells - Chicken Noodle	*1.0999999999999996**xsd:decimal
10	1160131104	Kuhn-Hills	GROWine-Magnotta-Belpaese	Wine - Magnotta - Belpaese	*1.084999999999999**xsd:decimal

Figure 5: Query output

## 5 Discussion

### References

- [1] Guohui Xiao, Davide Lanti, Roman Kontchakov, Sarah Komla-Ebri, Elem Güzel-Kalayci, Linfang Ding, Julien Corman, Benjamin Cogrel, Diego Calvanese, and Elena Botoeva. (2020) *The Virtual Knowledge Graph System Ontop*, International Semantic Web Conference.
- [2] Diego Calvanese, Benjamin Cogrel, Sarah Komla-Ebri, Roman Kontchakov, Davide Lanti, Martin Rezk, Mariano Rodriguez-Muro, and Guohui Xiao. (2017) *Ontop: Answering SPARQL Queries over Relational Databases*, Semantic Web Journal 8.3, pp. 471–487.