USE CASE STUDY PROJECT

Course : IE6700 Data Management For Analytics

Group No. : 11

Student Names : Prabhat Chanda & Ruchirkanth Gandikota

(chanda.p@northeastern.edu, gandikota.r@northeastern.edu)

Executive Summary :

In the technologically advancing and fast-paced world, people are choosing the online shopping option over traditional window shopping due to the comfort it provides and the time it saves. E-commerce businesses like Amazon, Walmart etc. are growing exponentially due to the variety of goods and services and the ease they provide to people for choosing them over the traditional in-person stores. Even though some parts of the population are reluctant to shop online for their basic needs due to their orthodox thinking, the advantages overweigh the disadvantages which is why a majority of the people have transitioned their ways to shop following the latest trends.

We aim at developing a database for an e-commerce website to store the data of customers, goods, payments, orders etc. The company will benefit in various ways, like restocking their inventory when particular goods run out, keeping a check of the payments, analysing the purchasing trends of the customers across the globe etc. Our project intends to aid the convenience of data storage for the e-commerce company with an efficiently architectured database model.

Introduction :

An e-commerce company will have numerous customers ordering eclectic products online through their websites. The company will have to store data in a segregated manner and we will have to maintain dedicated databases for specific types of data so that the company can use this streamlined data to obtain valuable insights and take clear profitable business decisions.

Our project will have information about the customers. Customers will be uniquely identified by their customer id and they will also have to provide their name, phone number and address. A customer can earn benefit points depending on the amount and the frequency of their shopping, which can be redeemed later.

Membership would be another entity, which will contain the membership type of each customer. A customer can be in the 'gold', 'silver' or 'platinum' type, and new customers will not have any membership type associated with them, their type would be seen in the table as 'NULL'.

The data regarding products offered will be a dedicated entity type which will have different attributes like product_id(which will be uniquely identifiable), name of the product, quantity in stock and unit price.

The database will also have an entity type that records data about the orders that are placed by the customers. The orders will be unique by an order ID and will be linked to the customer ID attribute from the customer's entity to clearly distinguish the order placed by a respected customer. The order date, expected delivery date and the shipper ID will also be part of the orders entity. Orders will also have a status_id associated with them, whether the order is 'processed', 'shipped' or 'delivered', and this information will be linked to another table, named order_status.

Shippers will be a separate entity type that will contain the data about the contracted suppliers of the goods to the company. They will have a unique Shipper ID and the name of the shipper as the attributes and this data can be inherited by the other entities wherever the reference of the shippers is required.

Another entity type would be payments. They will contain information about the date, amount and mode of payment. Payment ID will be used to uniquely identify them, and a separate invoice will be generated for each payment. Each payment will then be linked to the specific customer.

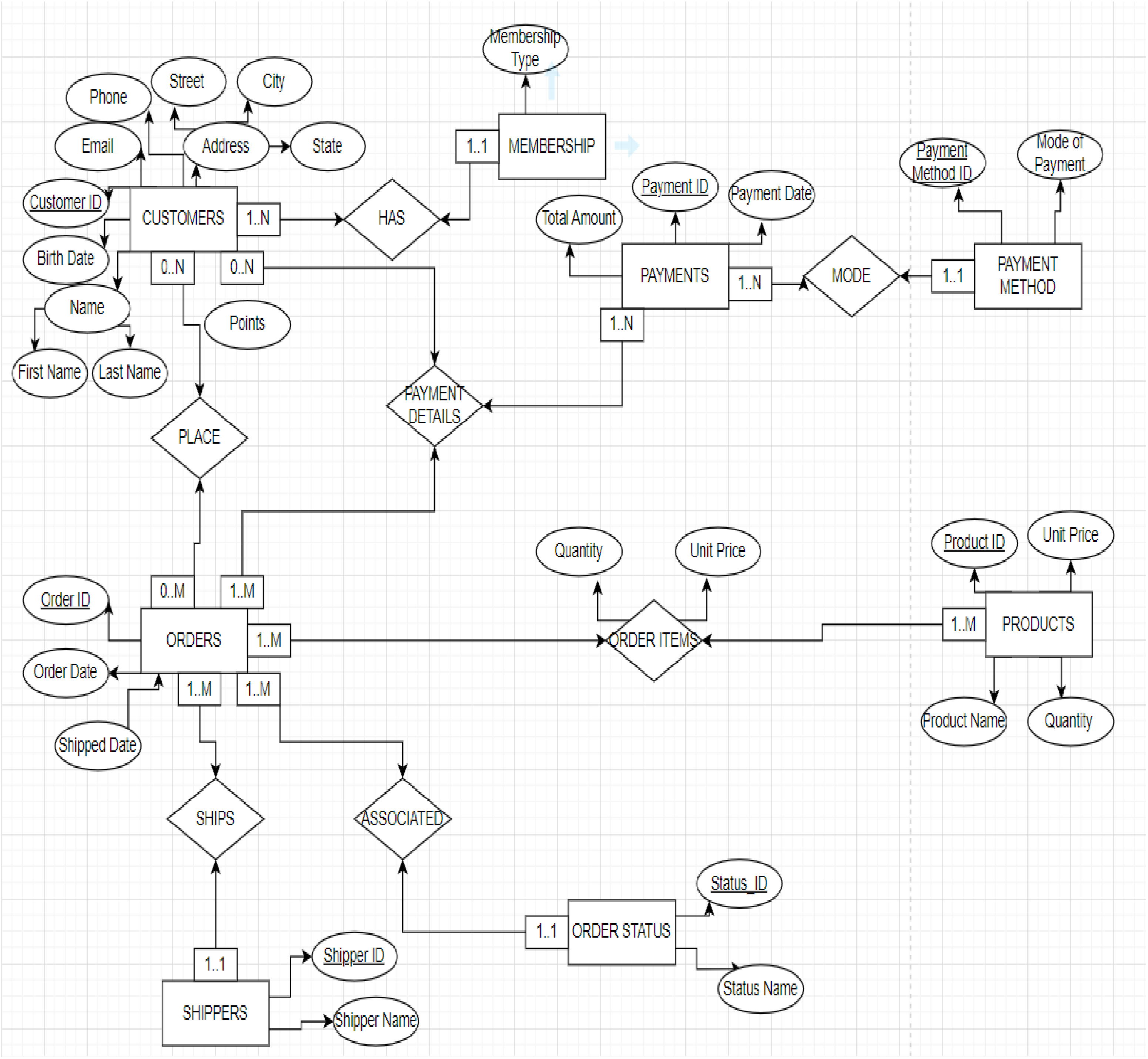
Mode of payment will be a table and would contain the different payment modes.

Order_payments_details will represent a ternary relationship between customers, orders and payments, and will contain the primary keys of the above mentioned 3 tables.

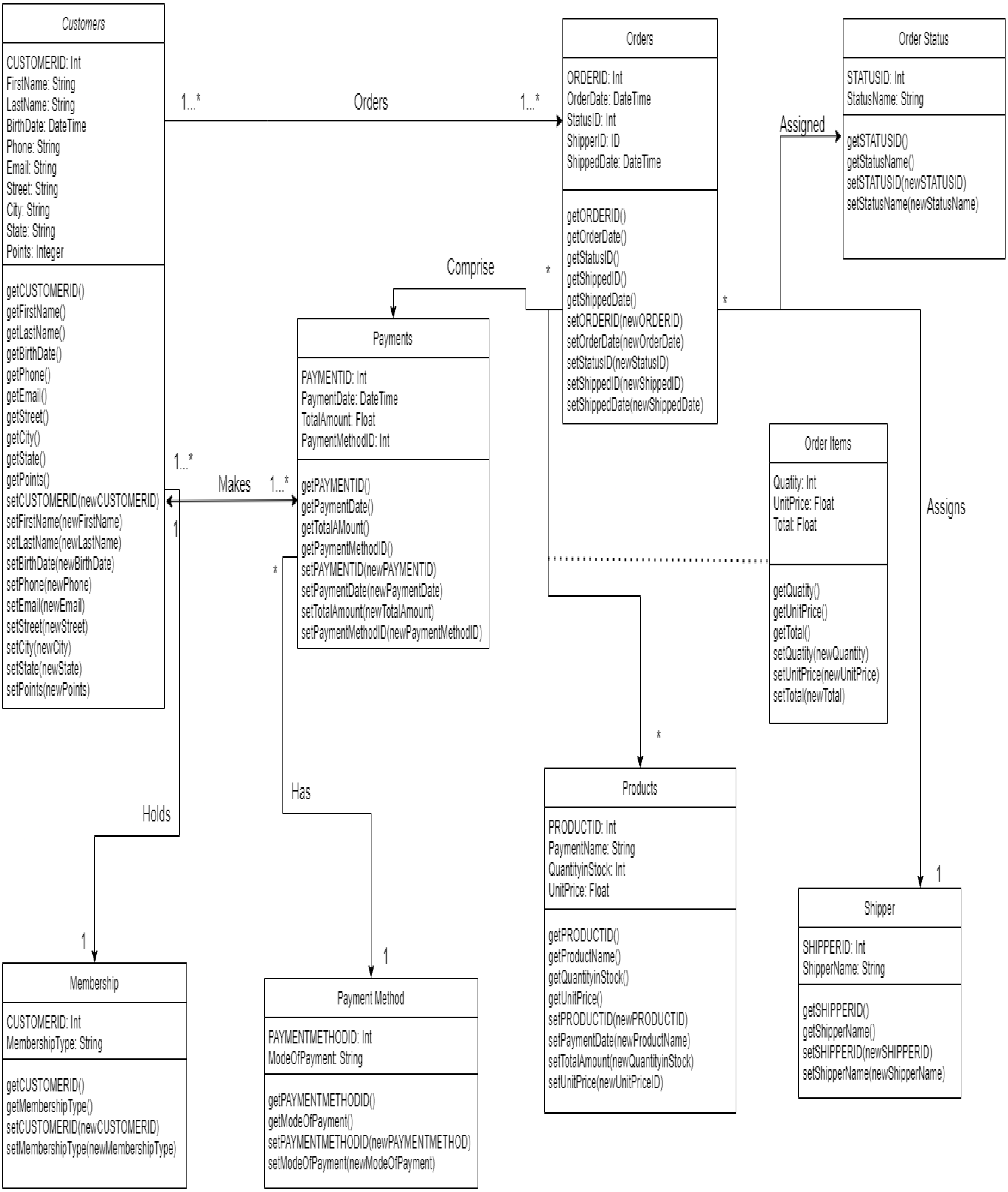
Order_items relationship between the tables orders and products, will also have attribute types quantity and unit price associated with it.

Conceptual Data Modelling :

1. EER Diagram :



2. UML Diagram :



Mapping Conceptual Model to Relational Model :

(Primary Keys are UNDERLINED and Foreign Keys are in *ITALICS*)

Customers (Customer_ID, First Name, Last Name, Birth Date, Phone, Street Address, City, State, Points, Email)

Membership (*Customer_ID*, Membership Type)
Customer_ID Is the primary key and the foreign key and it refers to the primary key in the relation Customers; NULL not allowed.

Orders (Order_ID, Order Date, *Status ID*, Shipped Date, *Shipper_ID*)
Shipper_ID Is the foreign key and it refers to the primary key in the relation Shippers; NULL not allowed.
Status ID Is the foreign key and it refers to the primary key in the relation order_status; NULL not allowed.

Place (*Customer_ID*, *Order_ID*)
Customer_ID Is the foreign key and it refers to the primary key in the relation Customers; NULL not allowed.
Order_ID Is the foreign key and it refers to the primary key in the relation Orders; NULL not allowed.
Primary Key is the combination of both the above foreign keys.

Shippers (Shipper_ID, Shipper Name)

References (*Order ID*, *Order Items ID*)
Order Items ID Is the foreign key and it refers to the primary key in the relation Order Items; NULL not allowed.
Order ID Is the foreign key and it refers to the primary key in the relation Orders; NULL not allowed.
Primary Key is the combination of both the above foreign keys.

Belongs (*Order Item ID*, *Product_ID*)
Order Items ID Is the foreign key and it refers to the primary key in the relation Order Items; NULL not allowed.
Product_ID Is the foreign key and it refers to the primary key in the relation Products; NULL not allowed.
Primary Key is the combination of both the above foreign keys.

Order Items (*Order ID*, *Product_ID*, Quantity, Unit Price, Total)
Order_ID Is the foreign key and it refers to the primary key in the relation Orders; NULL not allowed.
Product_ID Is the foreign key and it refers to the primary key in the relation Products; NULL not allowed.
Primary Key is the combination of both the above foreign keys.

Products (Product_ID, Product Name, Quantity in Stock, Unit Price)

Payment Method (Payment Method ID, Mode of Payment)

Payments (Payment_ID, Payment Date, Total Amount, *Payment Method ID*)
Payment Method ID Is the foreign key and it refers to the primary key in the relation Payment Method; NULL not allowed.

Order_Payments_Details (*Customer_ID, Order_ID, Payment_ID)*
Customer_ID Is the foreign key and it refers to the primary key in the relation Customers; NULL not allowed.
Order_ID Is the foreign key and it refers to a key in the relation Orders; NULL not allowed.
Payment_ID Is the foreign key and it refers to the primary key in the relation Payments; NULL not allowed

Order_Status(Status_ID, Status Name)

Implementation :

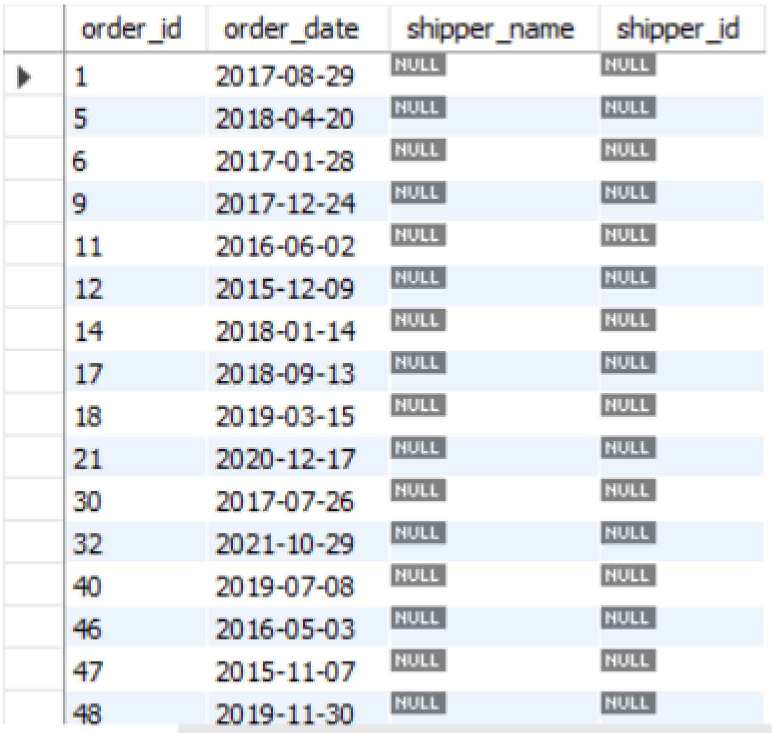
MySQL :

The database was created in MySQL and the following queries were performed :

Query 1 :

Retrieving data of orders that do not have any shippers.

```
SELECT o.order_id, o.order_date, s.shipper_name, s.shipper_id
FROM orders o
LEFT JOIN shippers s
USING(shipper_id)
WHERE s.shipper_id IS NULL
```

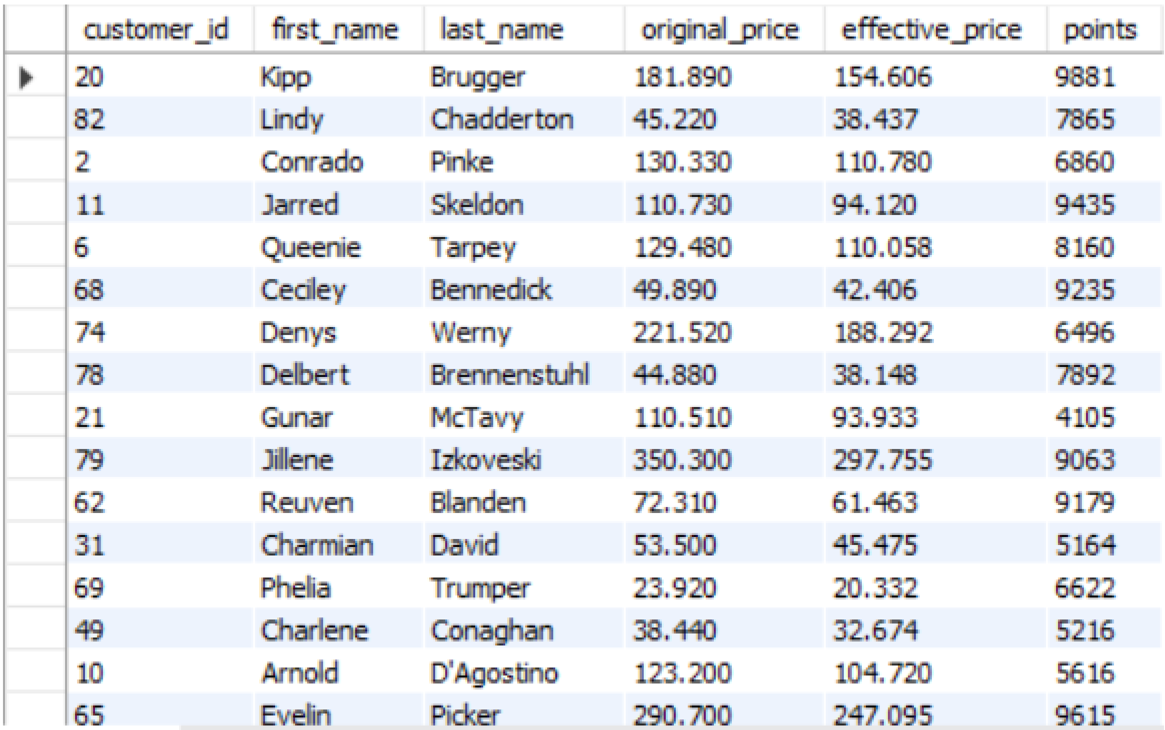


order_id	order_date	shipper_name	shipper_id
1	2017-08-29	NULL	NULL
5	2018-04-20	NULL	NULL
6	2017-01-28	NULL	NULL
9	2017-12-24	NULL	NULL
11	2016-06-02	NULL	NULL
12	2015-12-09	NULL	NULL
14	2018-01-14	NULL	NULL
17	2018-09-13	NULL	NULL
18	2019-03-15	NULL	NULL
21	2020-12-17	NULL	NULL
30	2017-07-26	NULL	NULL
32	2021-10-29	NULL	NULL
40	2019-07-08	NULL	NULL
46	2016-05-03	NULL	NULL
47	2015-11-07	NULL	NULL
48	2019-11-30	NULL	NULL

Query 2 :

Retrieving data of customers having points > 4000, and they are provided a discount of 15%, with their final amount as well.

```
SELECT c.customer_id, c.first_name, c.last_name, sum(oi.total) as original_price, sum((0.85*oi.total)) as effective_price, c.points
FROM customers c
JOIN order_payments_details opd
USING(customer_id)
JOIN order_items oi
USING(order_id)
GROUP BY c.customer_id
HAVING c.points > 4000
```

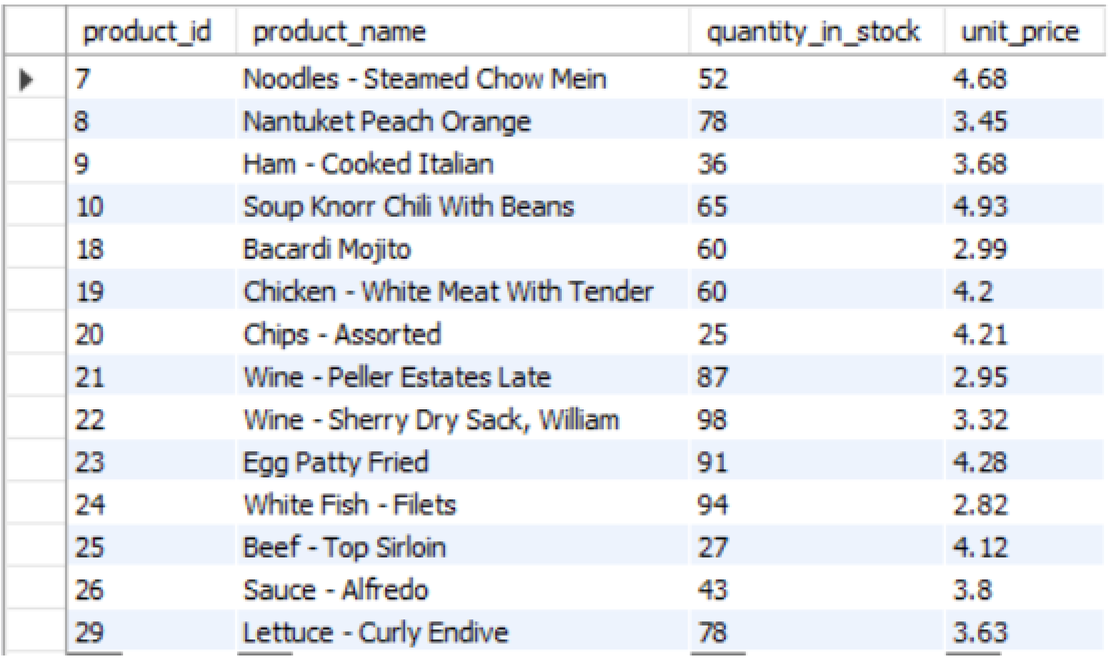


customer_id	first_name	last_name	original_price	effective_price	points
20	Kipp	Brugger	181.890	154.606	9881
82	Lindy	Chadderton	45.220	38.437	7865
2	Conrado	Pinke	130.330	110.780	6860
11	Jarred	Skeldon	110.730	94.120	9435
6	Queenie	Tarpey	129.480	110.058	8160
68	Cecley	Bennedick	49.890	42.406	9235
74	Denys	Werny	221.520	188.292	6496
78	Delbert	Brennenstuhl	44.880	38.148	7892
21	Gunar	McTavy	110.510	93.933	4105
79	Jillene	Izkoveski	350.300	297.755	9063
62	Reuven	Blanden	72.310	61.463	9179
31	Charmian	David	53.500	45.475	5164
69	Phelia	Trumper	23.920	20.332	6622
49	Charlene	Conaghan	38.440	32.674	5216
10	Arnold	D'Agostino	123.200	104.720	5616
65	Evelin	Picker	290.700	247.095	9615

Query 3 :

Retrieving data of the products that are more expensive than milk

```
SELECT *
FROM products
WHERE unit_price > (
        SELECT unit_price
    FROM products
    WHERE product_name="Milk - 2%"
    )
```

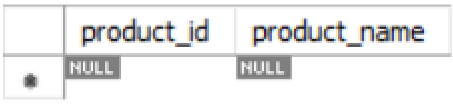


	product_id	product_name	quantity_in_stock	unit_price
▶	7	Noodles - Steamed Chow Mein	52	4.68
	8	Nantuket Peach Orange	78	3.45
	9	Ham - Cooked Italian	36	3.68
	10	Soup Knorr Chili With Beans	65	4.93
	18	Bacardi Mojito	60	2.99
	19	Chicken - White Meat With Tender	60	4.2
	20	Chips - Assorted	25	4.21
	21	Wine - Peller Estates Late	87	2.95
	22	Wine - Sherry Dry Sack, William	98	3.32
	23	Egg Patty Fried	91	4.28
	24	White Fish - Filets	94	2.82
	25	Beef - Top Sirloin	27	4.12
	26	Sauce - Alfredo	43	3.8
	29	Lettuce - Curly Endive	78	3.63

Query 4 :

Retrieving product_id and product_name of the products that were never ordered

```
SELECT DISTINCT product_id, product_name
FROM products
WHERE product_id NOT IN (
        SELECT DISTINCT product_id
    FROM order_items )
```

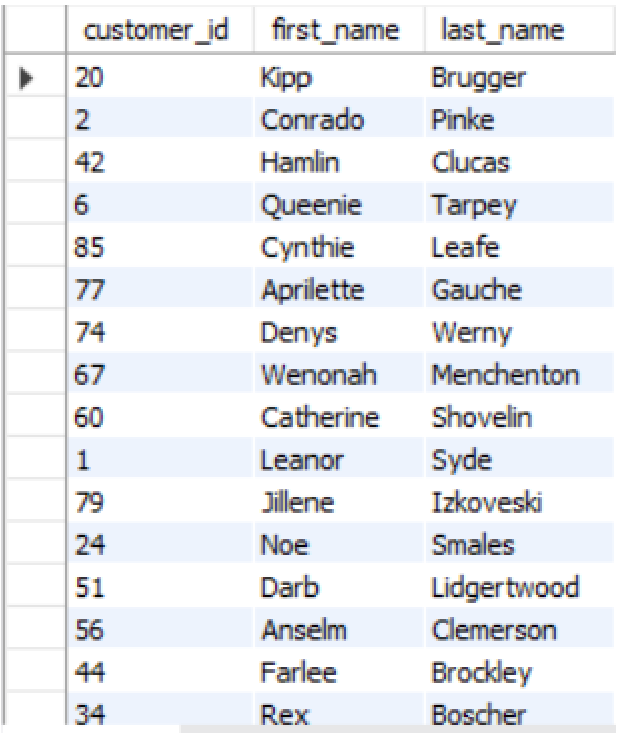


	product_id	product_name
✱	NULL	NULL

Query 5 :

Retrieving data of the customers who spent more than average amount spent by all the customers.

```
SELECT customer_id, first_name, last_name
FROM customers
JOIN order_payments_details
USING(customer_id)
JOIN order_items
USING(order_id)
GROUP BY customer_id
HAVING SUM(total)>
(SELECT AVG(sum)
FROM (
        SELECT customer_id, SUM(total) AS sum
        FROM customers
        JOIN order_payments_details
        USING(customer_id)
        JOIN order_items
        USING(order_id)
        GROUP BY customer_id) AS m)
```

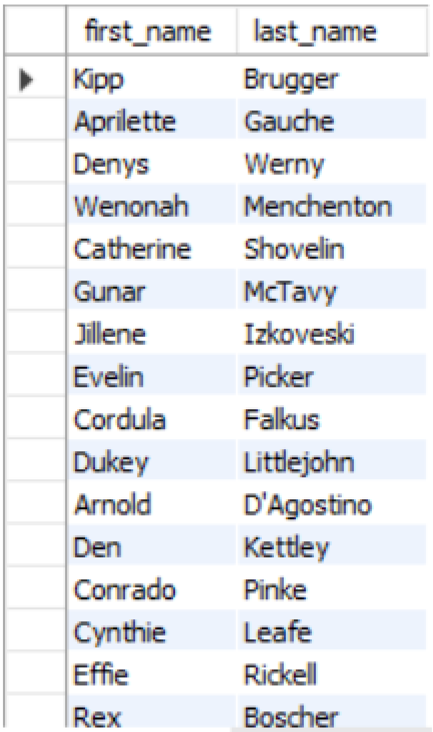


customer_id	first_name	last_name
20	Kipp	Brugger
2	Conrado	Pinke
42	Hamlin	Clucas
6	Queenie	Tarpey
85	Cynthie	Leafe
77	Aprilette	Gauche
74	Denys	Werny
67	Wenonah	Menchenton
60	Catherine	Shovelin
1	Leanor	Syde
79	Jillene	Izkoveski
24	Noe	Smales
51	Darb	Lidgertwood
56	Anselm	Clemerson
44	Farlee	Brockley
34	Rex	Boscher

Query 6 :

Retrieving names of all the customers who ordered cabbage and tea

```
SELECT first_name, last_name
FROM customers
WHERE customer_id IN(
        SELECT customer_id
        FROM order_payments_details
        WHERE order_id IN(
                SELECT order_id
                FROM order_items
                WHERE product_id IN(
                        SELECT product_id
                        FROM products
                        WHERE product_name IN ('Cabbage - Nappa', 'Tea - Green'))))
```

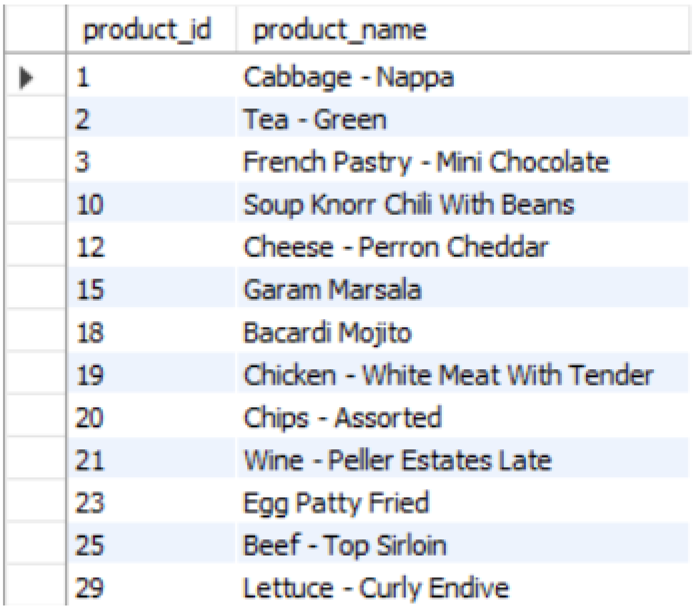


first_name	last_name
Kipp	Brugger
Aprilette	Gauche
Denys	Werny
Wenonah	Menchenton
Catherine	Shovelin
Gunar	McTavy
Jillene	Izkoveski
Evelin	Picker
Cordula	Falkus
Dukey	Littlejohn
Arnold	D'Agostino
Den	Kettley
Conrado	Pinke
Cynthie	Leafe
Effie	Rickell
Rex	Boscher

Query 7 :

The product_id and product_name of those products which are present in more than 10 orders.

```
SELECT P.product_id, P.product_name
FROM products AS P
WHERE 10 <
(SELECT COUNT(*)
 FROM order_items AS OI
 WHERE P.product_id = OI.product_id)
```

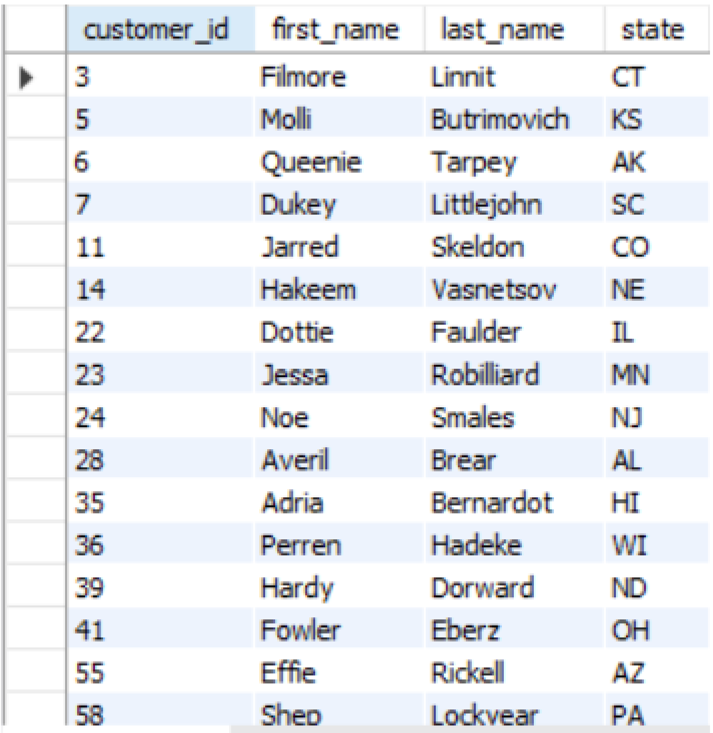


	product_id	product_name
▶	1	Cabbage - Nappa
	2	Tea - Green
	3	French Pastry - Mini Chocolate
	10	Soup Knorr Chili With Beans
	12	Cheese - Perron Cheddar
	15	Garam Marsala
	18	Bacardi Mojito
	19	Chicken - White Meat With Tender
	20	Chips - Assorted
	21	Wine - Peller Estates Late
	23	Egg Patty Fried
	25	Beef - Top Sirloin
	29	Lettuce - Curly Endive

Query 8 :

The customer_id, first_name, last_name and state of each customer who has the highest number of points of all the customers in the same state.

```
 SELECT C1.customer_id, C1.first_name, C1.last_name, C1.state
 FROM customers AS C1
 WHERE C1.points >= ALL
 (SELECT C2.points
  FROM customers AS C2
  WHERE C1.state = C2.state)
```

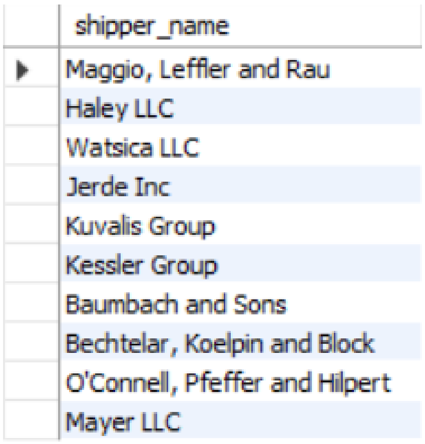


customer_id	first_name	last_name	state
3	Filmore	Linnit	CT
5	Molli	Butrimovich	KS
6	Queenie	Tarpey	AK
7	Dukey	Littlejohn	SC
11	Jarred	Skeldon	CO
14	Hakeem	Vasnetsov	NE
22	Dottie	Faulder	IL
23	Jessa	Robilliard	MN
24	Noe	Smales	NJ
28	Averil	Brear	AL
35	Adria	Bernardot	HI
36	Perren	Hadeke	WI
39	Hardy	Dorward	ND
41	Fowler	Eberz	OH
55	Effie	Rickell	AZ
58	Shep	Lockyear	PA

Query 9 :

shipper_name of all the shippers who have delivered their products

```
SELECT shipper_name
FROM shippers AS S
WHERE EXISTS(
        SELECT *
    FROM orders AS O
    WHERE S.shipper_id = O.shipper_id
    AND o.status_id = 3)
```

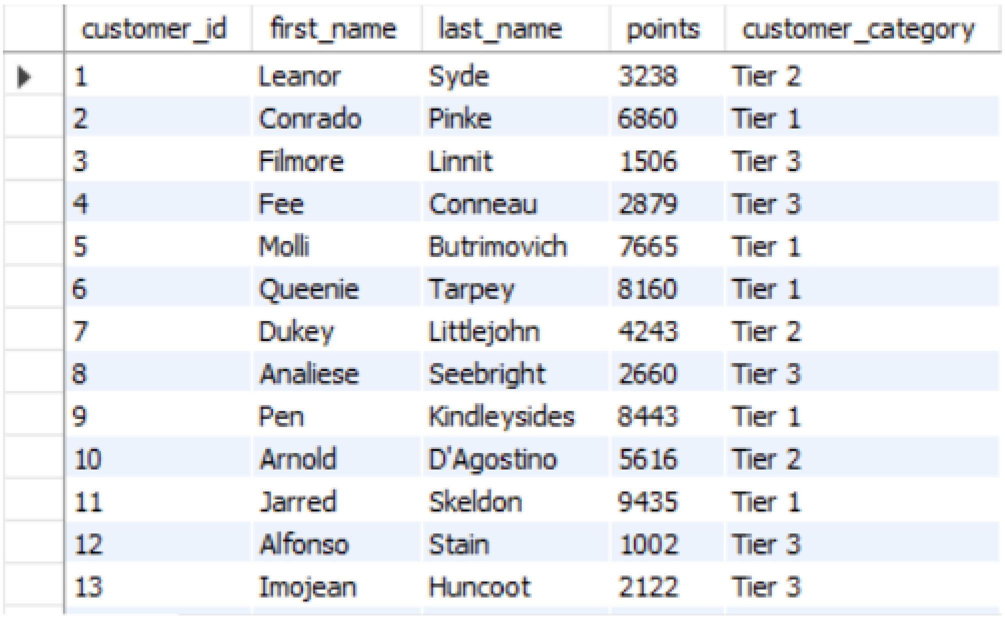


shipper_name
Maggio, Leffler and Rau
Haley LLC
Watsica LLC
Jerde Inc
Kuvalis Group
Kessler Group
Baumbach and Sons
Bechtelar, Koelpin and Block
O'Connell, Pfeffer and Hilpert
Mayer LLC

Query 10:

Specify the category of each customer according to their points earned

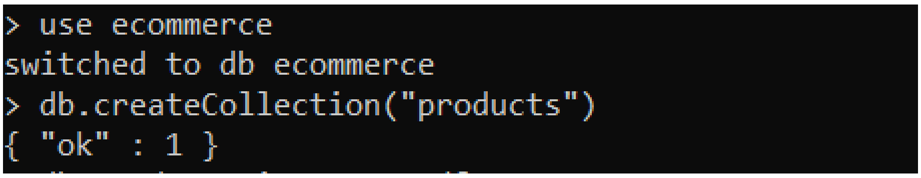
```
SELECT customer_id, first_name, last_name, points,
CASE
        WHEN (points BETWEEN 1 AND 3000) THEN 'Tier 3'
        WHEN (points BETWEEN 3000 AND 6000) THEN 'Tier 2'
        WHEN (points > 6000) THEN 'Tier 1'
END AS customer_category
FROM customers
```



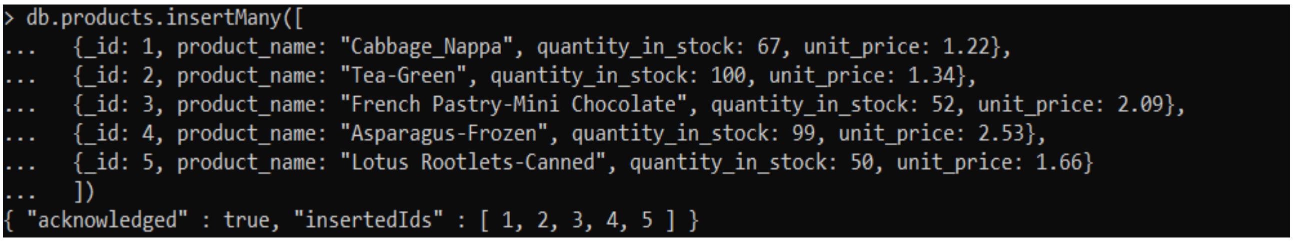
customer_id	first_name	last_name	points	customer_category
1	Leanor	Syde	3238	Tier 2
2	Conrado	Pinke	6860	Tier 1
3	Filmore	Linnit	1506	Tier 3
4	Fee	Conneau	2879	Tier 3
5	Molli	Butrimovich	7665	Tier 1
6	Queenie	Tarpey	8160	Tier 1
7	Dukey	Littlejohn	4243	Tier 2
8	Analiese	Seebright	2660	Tier 3
9	Pen	Kindleysides	8443	Tier 1
10	Arnold	D'Agostino	5616	Tier 2
11	Jarred	Skeldon	9435	Tier 1
12	Alfonso	Stain	1002	Tier 3
13	Imojean	Huncoot	2122	Tier 3

NoSQL :

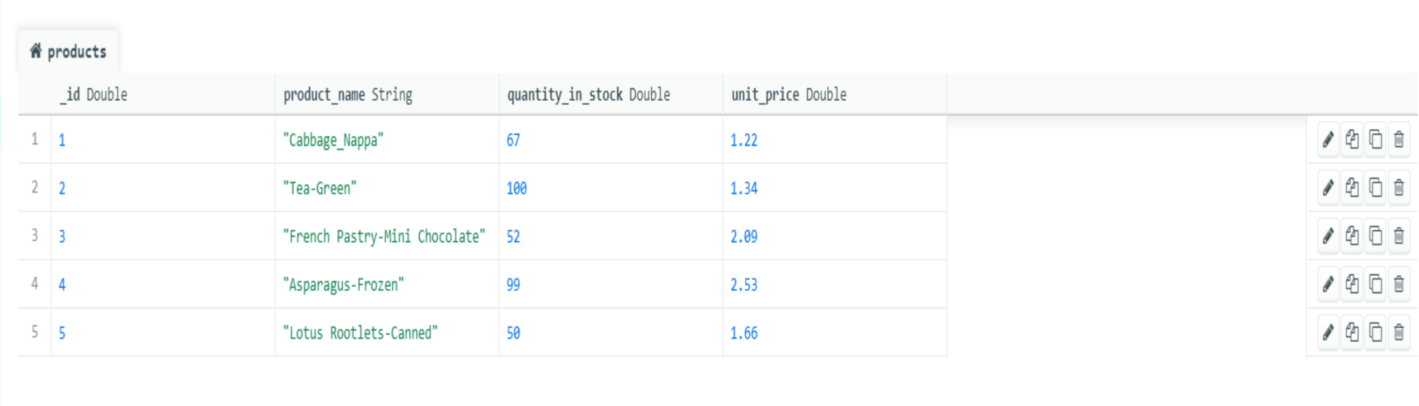
3 Tables (Products, Customers and Customers_Orders) were created in MongoDB and few queries were executed.



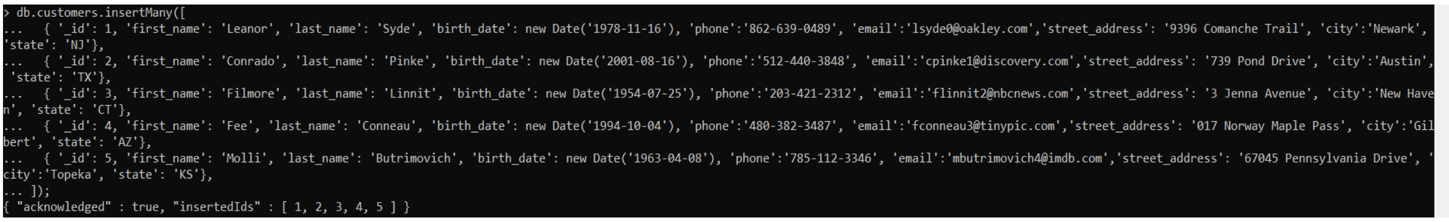
```
> use ecommerce
switched to db ecommerce
> db.createCollection("products")
{ "ok" : 1 }
```



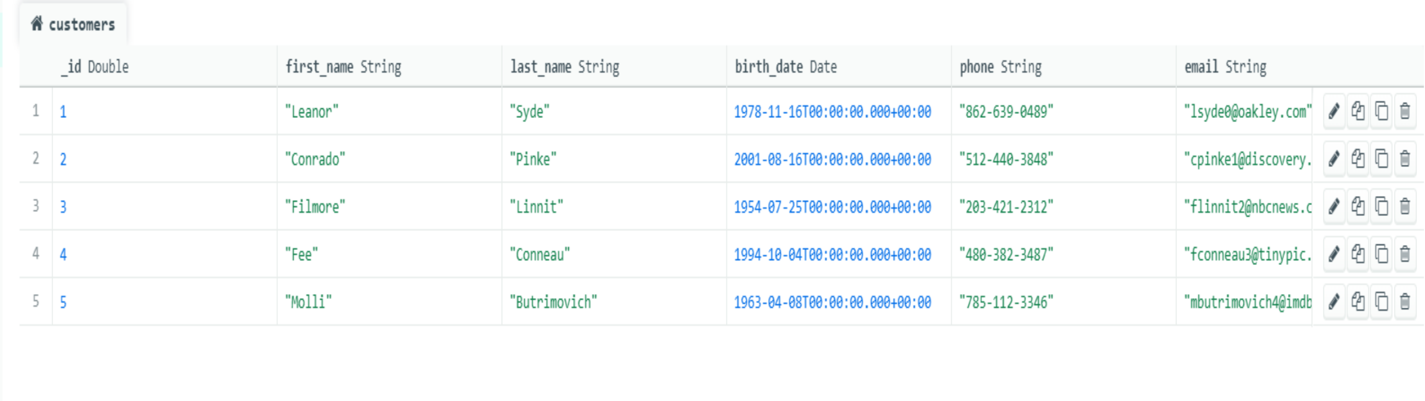
```
> db.products.insertMany([
...    {_id: 1, product_name: "Cabbage_Nappa", quantity_in_stock: 67, unit_price: 1.22},
...    {_id: 2, product_name: "Tea-Green", quantity_in_stock: 100, unit_price: 1.34},
...    {_id: 3, product_name: "French Pastry-Mini Chocolate", quantity_in_stock: 52, unit_price: 2.09},
...    {_id: 4, product_name: "Asparagus-Frozen", quantity_in_stock: 99, unit_price: 2.53},
...    {_id: 5, product_name: "Lotus Rootlets-Canned", quantity_in_stock: 50, unit_price: 1.66}
...    ])
{ "acknowledged" : true, "insertedIds" : [ 1, 2, 3, 4, 5 ] }
```



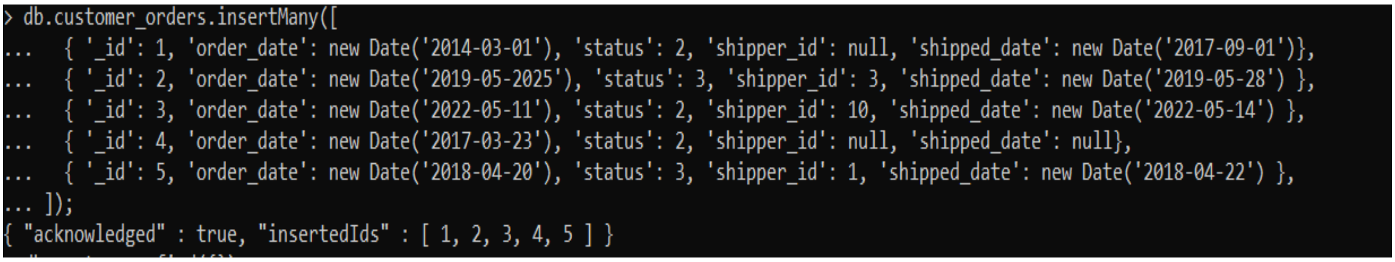
	_id Double	product_name String	quantity_in_stock Double	unit_price Double
1	1	"Cabbage_Nappa"	67	1.22
2	2	"Tea-Green"	100	1.34
3	3	"French Pastry-Mini Chocolate"	52	2.09
4	4	"Asparagus-Frozen"	99	2.53
5	5	"Lotus Rootlets-Canned"	50	1.66



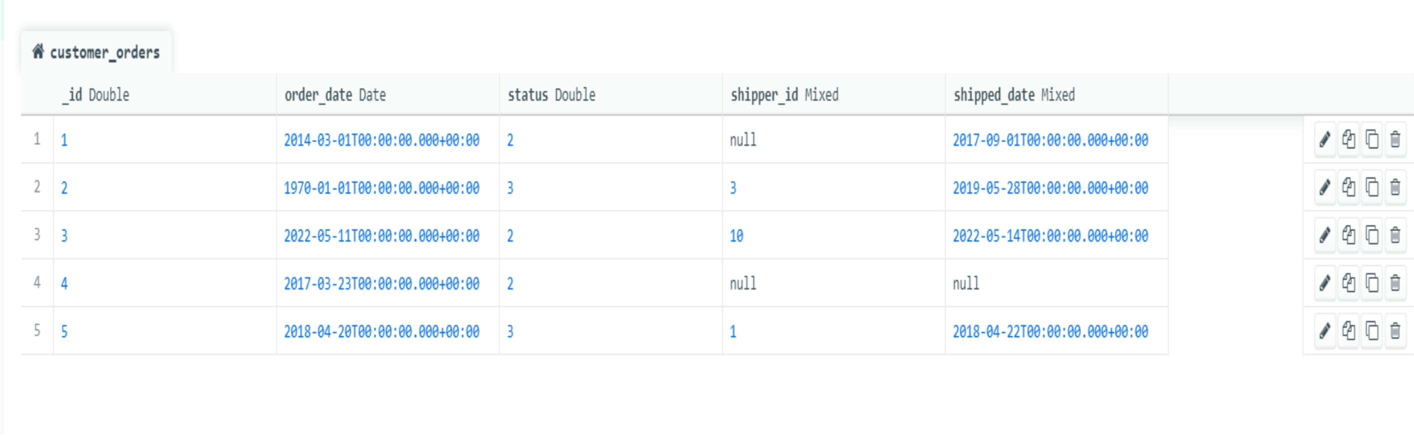
```
> db.customers.insertMany([
...    { '_id': 1, 'first_name': 'Leanor', 'last_name': 'Syde', 'birth_date': new Date('1978-11-16'), 'phone':'862-639-0489', 'email':'lsyde0@oakley.com','street_address': '9396 Comanche Trail', 'city':'Newark',
'state': 'NJ'},
...    { '_id': 2, 'first_name': 'Conrado', 'last_name': 'Pinke', 'birth_date': new Date('2001-08-16'), 'phone':'512-440-3848', 'email':'cpinke1@discovery.com','street_address': '739 Pond Drive', 'city':'Austin',
 'state': 'TX'},
...    { '_id': 3, 'first_name': 'Filmore', 'last_name': 'Linnit', 'birth_date': new Date('1954-07-25'), 'phone':'203-421-2312', 'email':'flinnit2@nbcnews.com','street_address': '3 Jenna Avenue', 'city':'New Have
n', 'state': 'CT'},
...    { '_id': 4, 'first_name': 'Fee', 'last_name': 'Conneau', 'birth_date': new Date('1994-10-04'), 'phone':'480-382-3487', 'email':'fconneau3@tinypic.com','street_address': '017 Norway Maple Pass', 'city':'Gil
bert', 'state': 'AZ'},
...    { '_id': 5, 'first_name': 'Molli', 'last_name': 'Butrimovich', 'birth_date': new Date('1963-04-08'), 'phone':'785-112-3346', 'email':'mbutrimovich4@imdb.com','street_address': '67045 Pennsylvania Drive', '
city':'Topeka', 'state': 'KS'},
... ]);
{ "acknowledged" : true, "insertedIds" : [ 1, 2, 3, 4, 5 ] }
```



	_id Double	first_name String	last_name String	birth_date Date	phone String	email String
1	1	"Leanor"	"Syde"	1978-11-16T00:00:00.000+00:00	"862-639-0489"	"lsyde0@oakley.com"
2	2	"Conrado"	"Pinke"	2001-08-16T00:00:00.000+00:00	"512-440-3848"	"cpinke1@discovery.
3	3	"Filmore"	"Linnit"	1954-07-25T00:00:00.000+00:00	"203-421-2312"	"flinnit2@nbcnews.c
4	4	"Fee"	"Conneau"	1994-10-04T00:00:00.000+00:00	"480-382-3487"	"fconneau3@tinypic.
5	5	"Molli"	"Butrimovich"	1963-04-08T00:00:00.000+00:00	"785-112-3346"	"mbutrimovich4@imdb



```
> db.customer_orders.insertMany([
...     { '_id': 1, 'order_date': new Date('2014-03-01'), 'status': 2, 'shipper_id': null, 'shipped_date': new Date('2017-09-01')},
...     { '_id': 2, 'order_date': new Date('2019-05-2025'), 'status': 3, 'shipper_id': 3, 'shipped_date': new Date('2019-05-28') },
...     { '_id': 3, 'order_date': new Date('2022-05-11'), 'status': 2, 'shipper_id': 10, 'shipped_date': new Date('2022-05-14') },
...     { '_id': 4, 'order_date': new Date('2017-03-23'), 'status': 2, 'shipper_id': null, 'shipped_date': null},
...     { '_id': 5, 'order_date': new Date('2018-04-20'), 'status': 3, 'shipper_id': 1, 'shipped_date': new Date('2018-04-22') },
... ]);
{ "acknowledged" : true, "insertedIds" : [ 1, 2, 3, 4, 5 ] }
```

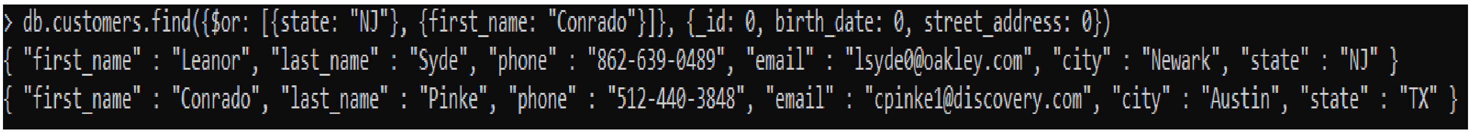


customer_orders

	_id Double	order_date Date	status Double	shipper_id Mixed	shipped_date Mixed
1	1	2014-03-01T00:00:00.000+00:00	2	null	2017-09-01T00:00:00.000+00:00
2	2	1970-01-01T00:00:00.000+00:00	3	3	2019-05-28T00:00:00.000+00:00
3	3	2022-05-11T00:00:00.000+00:00	2	10	2022-05-14T00:00:00.000+00:00
4	4	2017-03-23T00:00:00.000+00:00	2	null	null
5	5	2018-04-20T00:00:00.000+00:00	3	1	2018-04-22T00:00:00.000+00:00

Query 1 :

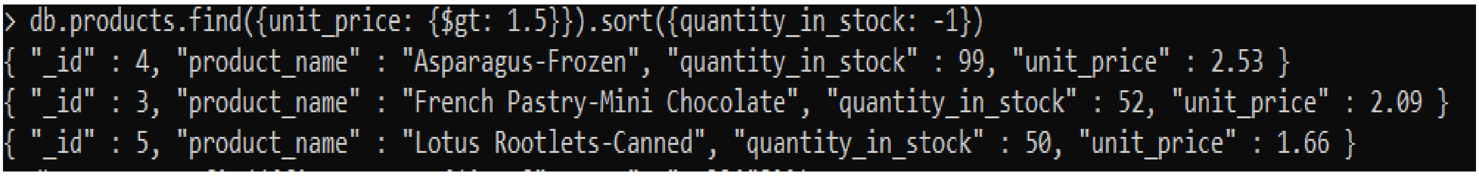
Get everything, except id, birth_date and street_address, from ‘customers’ collection, where either state is ‘NJ’ or first_name is ‘Conrado’.



```
> db.customers.find({$or: [{state: "NJ"}, {first_name: "Conrado"}]}, {_id: 0, birth_date: 0, street_address: 0})
{ "first_name" : "Leanor", "last_name" : "Syde", "phone" : "862-639-0489", "email" : "lsyde0@oakley.com", "city" : "Newark", "state" : "NJ" }
{ "first_name" : "Conrado", "last_name" : "Pinke", "phone" : "512-440-3848", "email" : "cpinke1@discovery.com", "city" : "Austin", "state" : "TX" }
```

Query 2 :

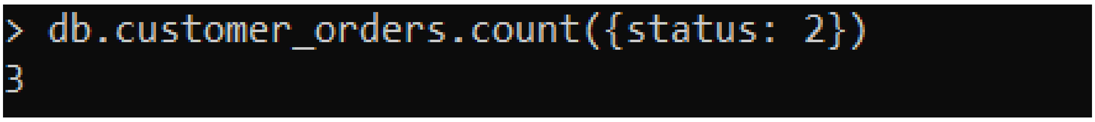
Get everything from ‘products’ collection, where unit_price > 1.5 and sort the results by quantity_in_stock in descending order.



```
> db.products.find({unit_price: {$gt: 1.5}}).sort({quantity_in_stock: -1})
{ "_id" : 4, "product_name" : "Asparagus-Frozen", "quantity_in_stock" : 99, "unit_price" : 2.53 }
{ "_id" : 3, "product_name" : "French Pastry-Mini Chocolate", "quantity_in_stock" : 52, "unit_price" : 2.09 }
{ "_id" : 5, "product_name" : "Lotus Rootlets-Canned", "quantity_in_stock" : 50, "unit_price" : 1.66 }
```

Query 3 :

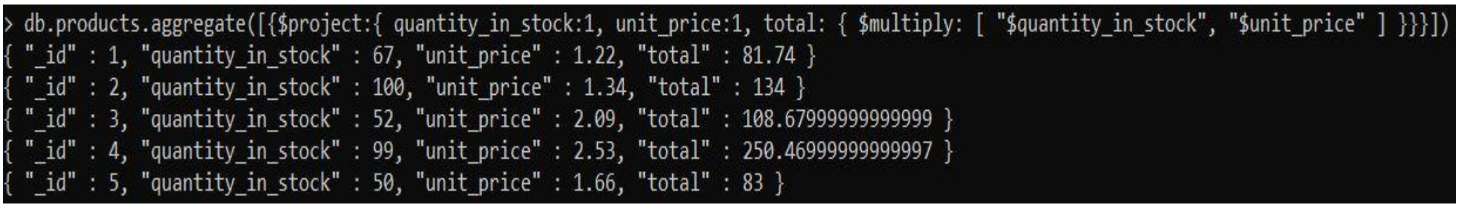
Count the number of orders from the ‘customer_orders’ collection, where the status = 2.



```
> db.customer_orders.count({status: 2})
3
```

Query 4 :

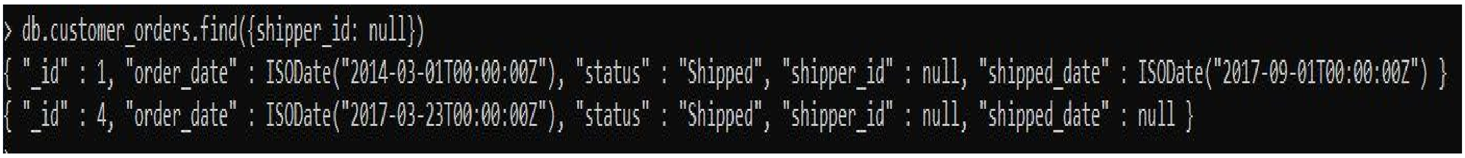
From ‘products’ collection, get the total amount related to the products present, for all the products.



```
> db.products.aggregate([{$project:{ quantity_in_stock:1, unit_price:1, total: { $multiply: [ "$quantity_in_stock", "$unit_price" ] }}}])
{ "_id" : 1, "quantity_in_stock" : 67, "unit_price" : 1.22, "total" : 81.74 }
{ "_id" : 2, "quantity_in_stock" : 100, "unit_price" : 1.34, "total" : 134 }
{ "_id" : 3, "quantity_in_stock" : 52, "unit_price" : 2.09, "total" : 108.67999999999999 }
{ "_id" : 4, "quantity_in_stock" : 99, "unit_price" : 2.53, "total" : 250.46999999999997 }
{ "_id" : 5, "quantity_in_stock" : 50, "unit_price" : 1.66, "total" : 83 }
```

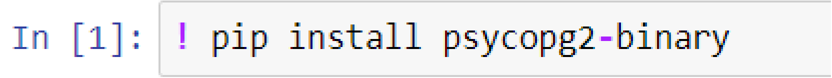
Query 5 :

From ‘customer_orders’ collection, find all the orders who do not have any shippers associated with them i.e. where shipper_id is NULL.

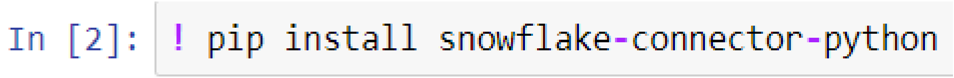


```
> db.customer_orders.find({shipper_id: null})
{ "_id" : 1, "order_date" : ISODate("2014-03-01T00:00:00Z"), "status" : "Shipped", "shipper_id" : null, "shipped_date" : ISODate("2017-09-01T00:00:00Z") }
{ "_id" : 4, "order_date" : ISODate("2017-03-23T00:00:00Z"), "status" : "Shipped", "shipper_id" : null, "shipped_date" : null }
```

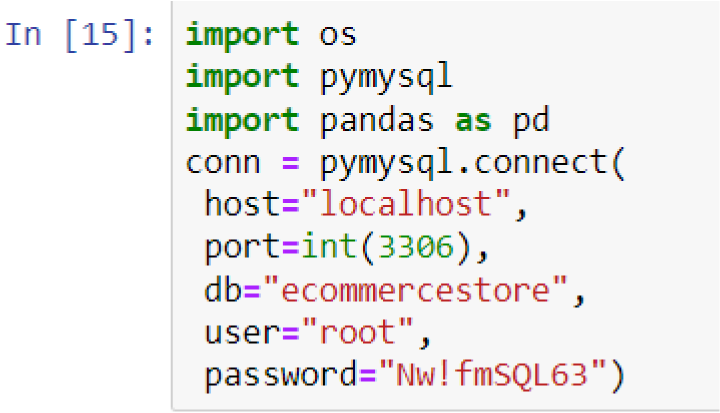
Database access via Python :



```
In [1]: ! pip install psycopg2-binary
```



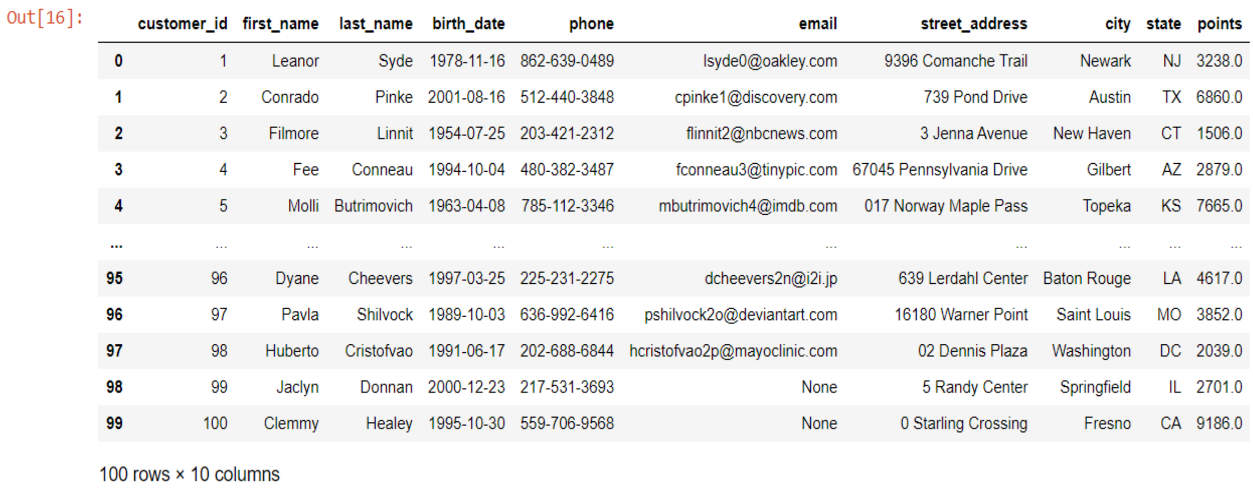
```
In [2]: ! pip install snowflake-connector-python
```



```
In [15]: import os
         import pymysql
         import pandas as pd
         conn = pymysql.connect(
          host="localhost",
          port=int(3306),
          db="ecommercestore",
          user="root",
          password="Nw!fmSQL63")
```

Query 1 :

df = pd.read_sql_query("SELECT * FROM customers", conn)
df



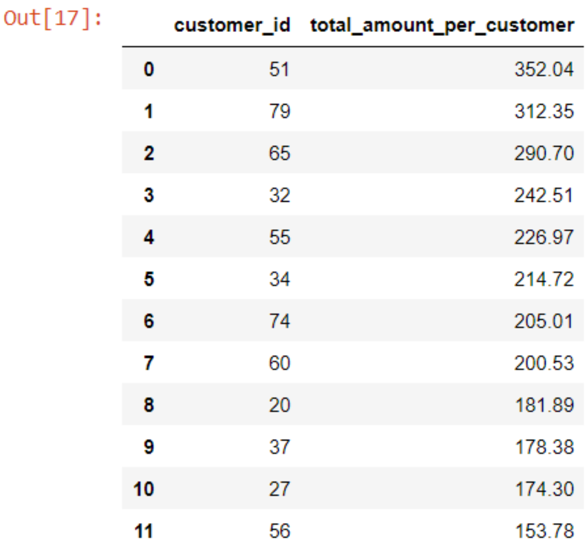
Out[16]:

	customer_id	first_name	last_name	birth_date	phone	email	street_address	city	state	points
0	1	Leanor	Syde	1978-11-16	862-639-0489	lsyde0@oakley.com	9396 Comanche Trail	Newark	NJ	3238.0
1	2	Conrado	Pinke	2001-08-16	512-440-3848	cpinke1@discovery.com	739 Pond Drive	Austin	TX	6860.0
2	3	Filmore	Linnit	1954-07-25	203-421-2312	flinnit2@nbcnews.com	3 Jenna Avenue	New Haven	CT	1506.0
3	4	Fee	Conneau	1994-10-04	480-382-3487	fconneau3@tinypic.com	67045 Pennsylvania Drive	Gilbert	AZ	2879.0
4	5	Molli	Butrimovich	1963-04-08	785-112-3346	mbutrimovich4@imdb.com	017 Norway Maple Pass	Topeka	KS	7665.0
...	...	...	...	...	...	...	...	...	...	...
95	96	Dyane	Cheevers	1997-03-25	225-231-2275	dcheevers2n@i2i.jp	639 Lerdahl Center	Baton Rouge	LA	4617.0
96	97	Pavla	Shilvock	1989-10-03	636-992-6416	pshilvock2o@deviantart.com	16180 Warner Point	Saint Louis	MO	3852.0
97	98	Huberto	Cristofvao	1991-06-17	202-688-6844	hcristofvao2p@mayoclinic.com	02 Dennis Plaza	Washington	DC	2039.0
98	99	Jaclyn	Donnan	2000-12-23	217-531-3693	None	5 Randy Center	Springfield	IL	2701.0
99	100	Clemmy	Healey	1995-10-30	559-706-9568	None	0 Starling Crossing	Fresno	CA	9186.0

100 rows × 10 columns

Query 2 :

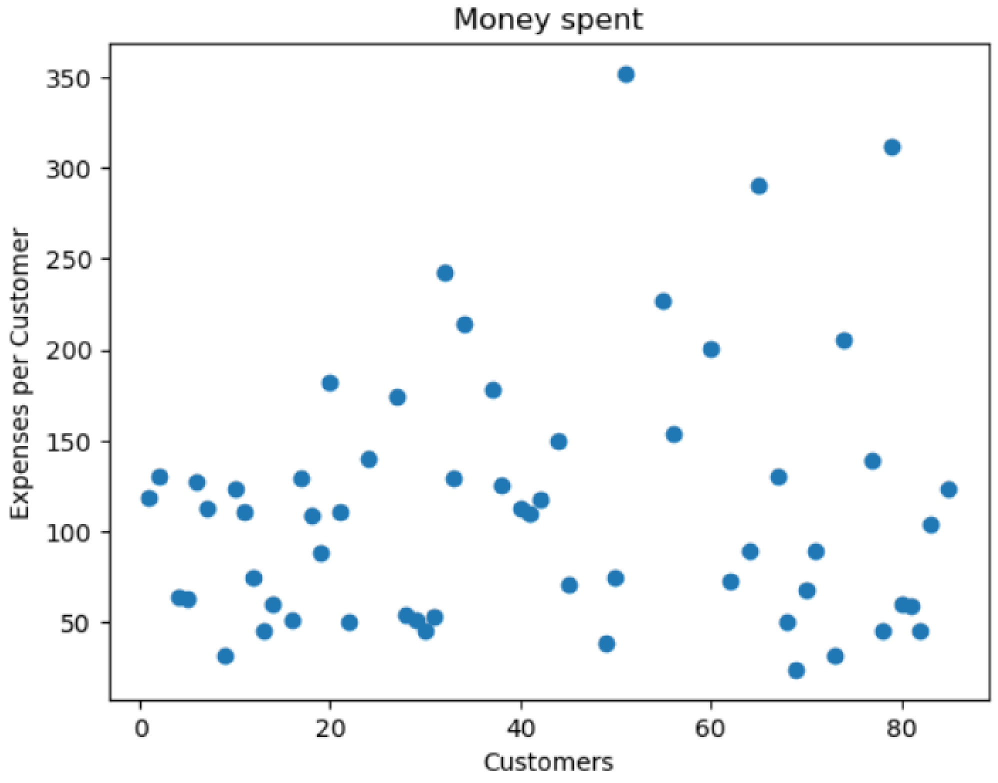
df = pd.read_sql_query("select customer_id, sum(total) as total_amount_per_customer from customers join order_payments_details using(customer_id) join order_items using(order_id) group by customer_id order by total_amount_per_customer desc", conn) df



Out[17]:

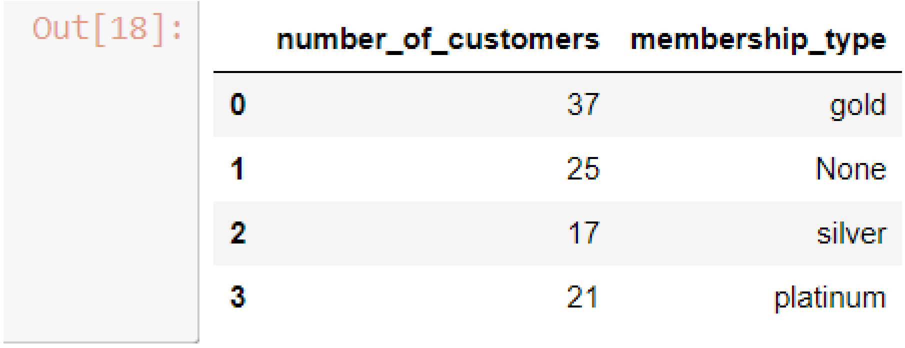
	customer_id	total_amount_per_customer
0	51	352.04
1	79	312.35
2	65	290.70
3	32	242.51
4	55	226.97
5	34	214.72
6	74	205.01
7	60	200.53
8	20	181.89
9	37	178.38
10	27	174.30
11	56	153.78

```
import matplotlib.pyplot as plt
plt.scatter(df["customer_id"], df["total_amount_per_customer"])
plt.title("Money spent")
plt.xlabel("Customers")
plt.ylabel("Expenses per Customer")
plt.show()
```



Query 2 :

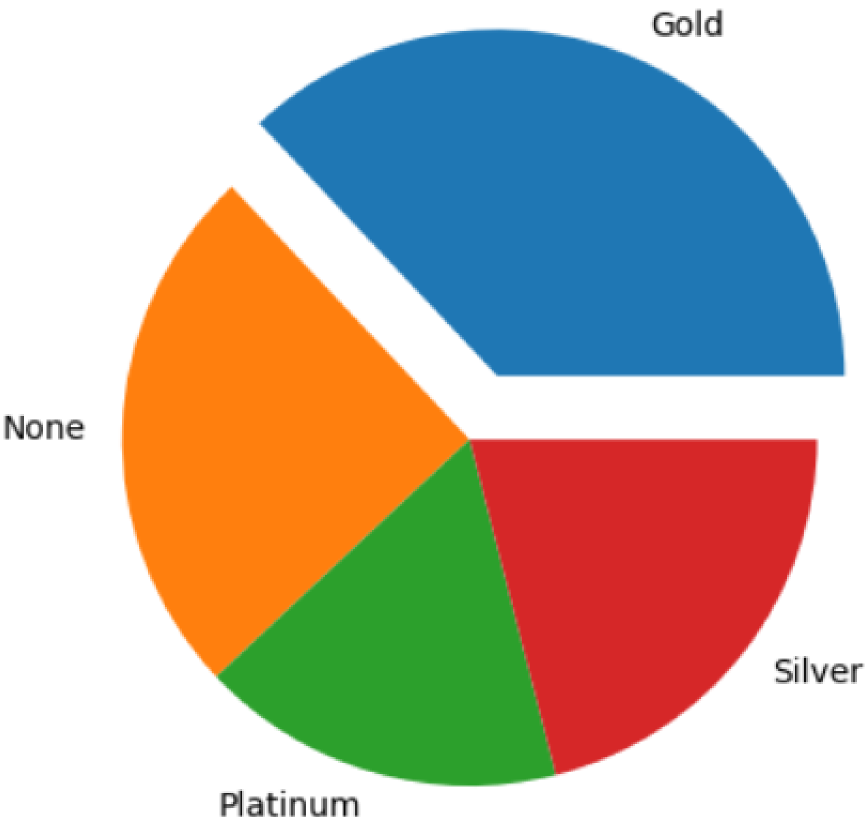
```
df=pd.read_sql_query("SELECT COUNT(C.customer_id) AS
number_of_customers, M.membership_type FROM customers AS C INNER
JOIN membership AS M ON C.customer_id = M.customer_id GROUP BY
M.membership_type", conn)
df
```



Out[18]:

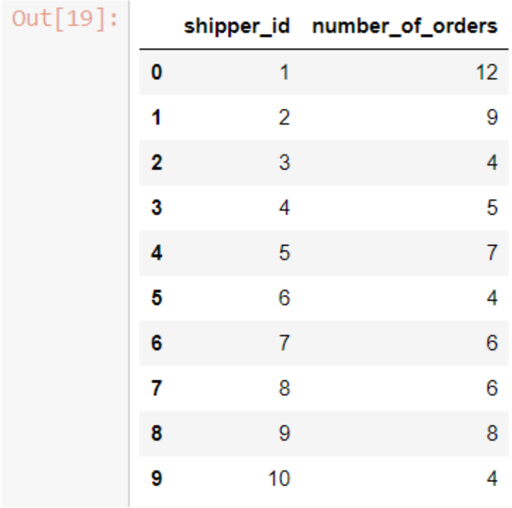
	number_of_customers	membership_type
0	37	gold
1	25	None
2	17	silver
3	21	platinum

```
mylabels = ["Gold", "None", "Platinum", "Silver"]
myexplode = [0.2, 0, 0, 0]
plt.pie(df["number_of_customers"], labels=mylabels, explode=myexplode)
```



Query 3 :

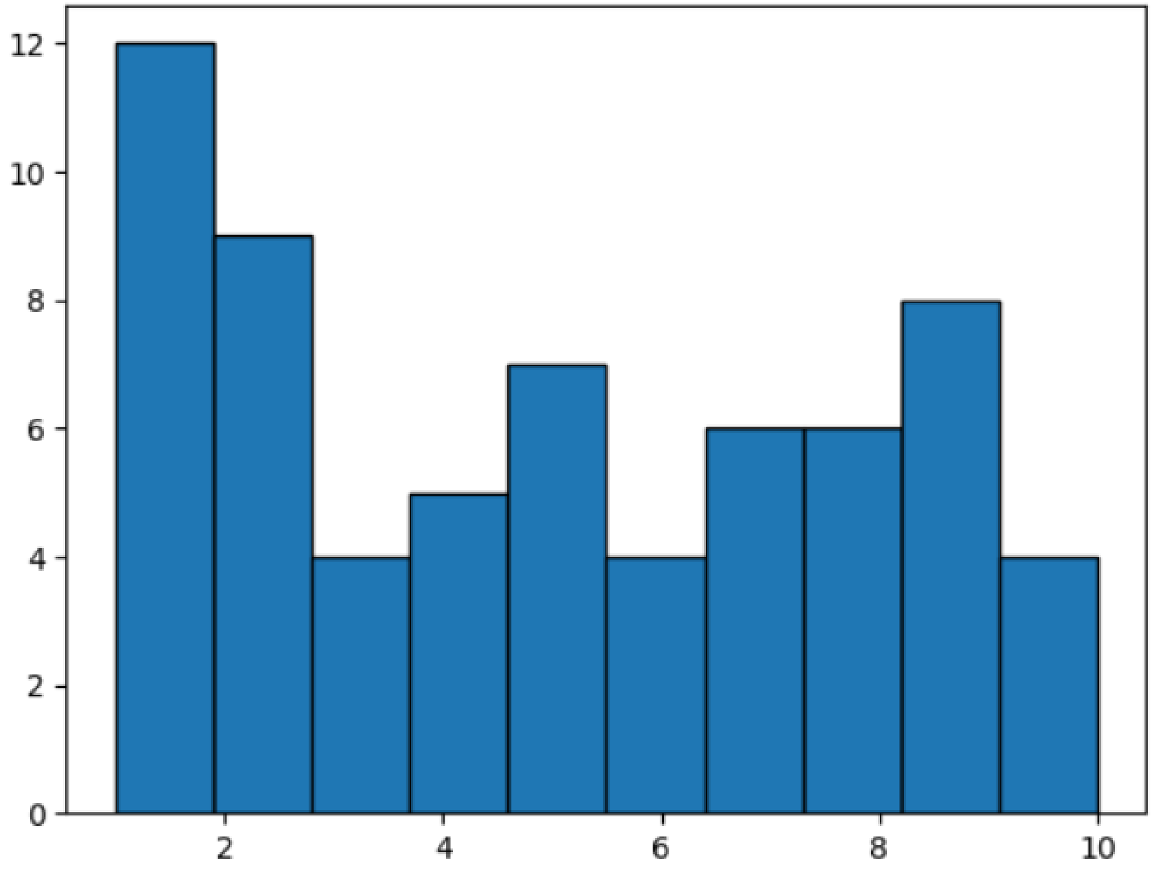
```
df=pd.read_sql_query("SELECT s.shipper_id, COUNT(*) AS
number_of_orders FROM shippers s  JOIN orders o USING(shipper_id)
GROUP BY s.shipper_id", conn)
df
```



Out[19]:

	shipper_id	number_of_orders
0	1	12
1	2	9
2	3	4
3	4	5
4	5	7
5	6	4
6	7	6
7	8	6
8	9	8
9	10	4

```
plt.hist(df['shipper_id'], weights=df["number_of_orders"], edgecolor = "black")
plt.show
```



Summary and Recommendation :

This project concentrates on the design and development of the E-commerce database which stores data in the most uncomplicated way. Using this kind of database schema would help in reducing costs related to storing the customer data. The database is developed in the 3rd normalisation form, which means the tables are well segregated and straightforward to understand at a glance.

The procured data can also be used for data analysis which is demonstrated in this report. Python is connected with MySQL to read the database and data analysis techniques were used to get insightful visualisations.

The NoSQL implementation of a few tables is also demonstrated in this report by running a few simple and aggregate queries. The E-commerce database schema is best when implemented as a relational schema which proves better than the NoSQL implementation due to the nature of the need. More research has to be conducted to efficiently implement a relational data schema in a NoSQL manner but would be very complicated and less efficient for this project point of view.