**Module:** Introduction to Databases

**Academic Year:** 2020/21, Sem I

**Programs:** HDAIML\_SEP, HDCSDEV\_INT, HDSDEV\_SEP, HDAIML\_SEPOL, HDBC\_SEPOL

**Assessment type**: Project

**Student ID** = X20170386

**Last name** = Ryan, First name = Arthur

**Part 1: Conceptual Design**

1. **Describe/introduce the chosen retail business and its objectives (200-400 words).**

**Description:**

My online retail ‘Way-faire’ business specializes in food and grocery areas.

Starting on 1st of January 2020 the company currently serves 1,000 customers. Based in the eurozone, it delivers to a large number of countries around the world. The firms objective is to grow market share as fast possible and once at material share of the market (8-10%) then raise prices to monetise the market share that was built up on the back of the low price leader / low to minimum pricing strategy.

Derived from Order-Details Sales for the first-year total €51,086 euro amount from query covering 3,030 quantity of individual items/products. The entire shopping experience is done online / on the web through the company portal/website. Customers fill out a profile page and securely register a card. Then they are free to start safely shopping, earning discounts based on their individual shopping activity. The app is secure, easy to use and reliable. The range available is currently narrow at a thousand products fulfilled by a hand-picked selection of thousand suppliers across numerous warehouses with adjoining management offices overhead. Delivery is from these warehouses through one of the multiple delivery companies available for each order despatch.

Through measuring activity of the orders we can see the performance of the company in revenue, direct costs, then deriving from revenue and direct costs we get gross profit, leading to overhead costs, which when taken off gross profit lead to operating profit before tax. Further as the company is customer centric its satisfaction is measured with the customer on three bases. One, satisfaction with the product, did the product perform in the way the customer expected it to. Two, speed of order, this being the time take to get the order delivered to the final customer, was it on time, late, or never arrived. Three, web experience i.e. experiences with using the online website to shop and create the order. Related to point two is RTM, which stands for return to manufacturer, we measure this as well to better understand supplier quality and minimise defective products in the company.

Business Assumptions:

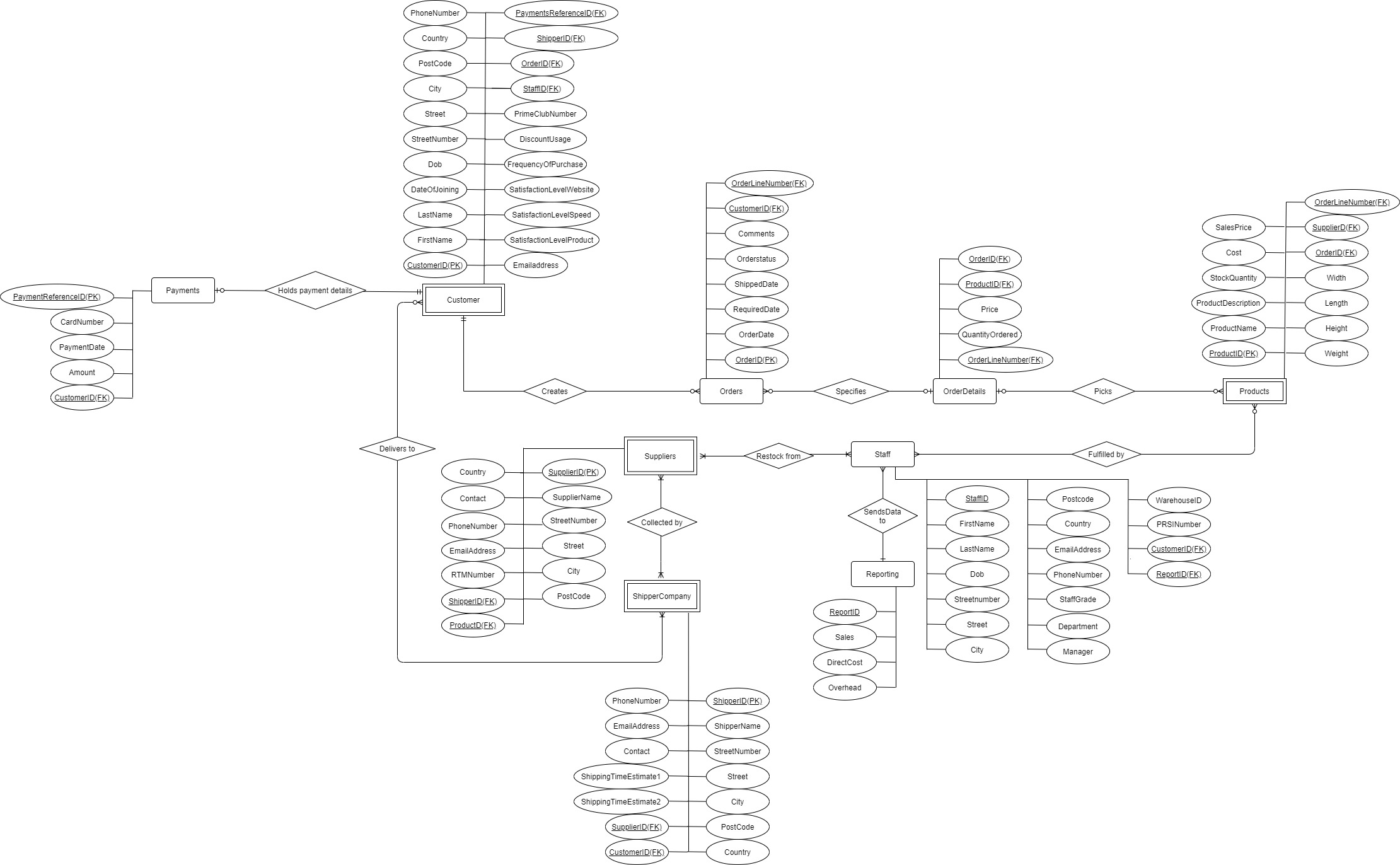
#1 - The use of bridge tables is not implemented to handle the many to many relationship where these arise. The reason for this is that the Mockuroo data generator randomly assigns values for OK in the bridge table from the original tables. This introduces duplicates. So by making the use of bridge tables not possible.

#2 – Every customer purchases at least one item over the year.

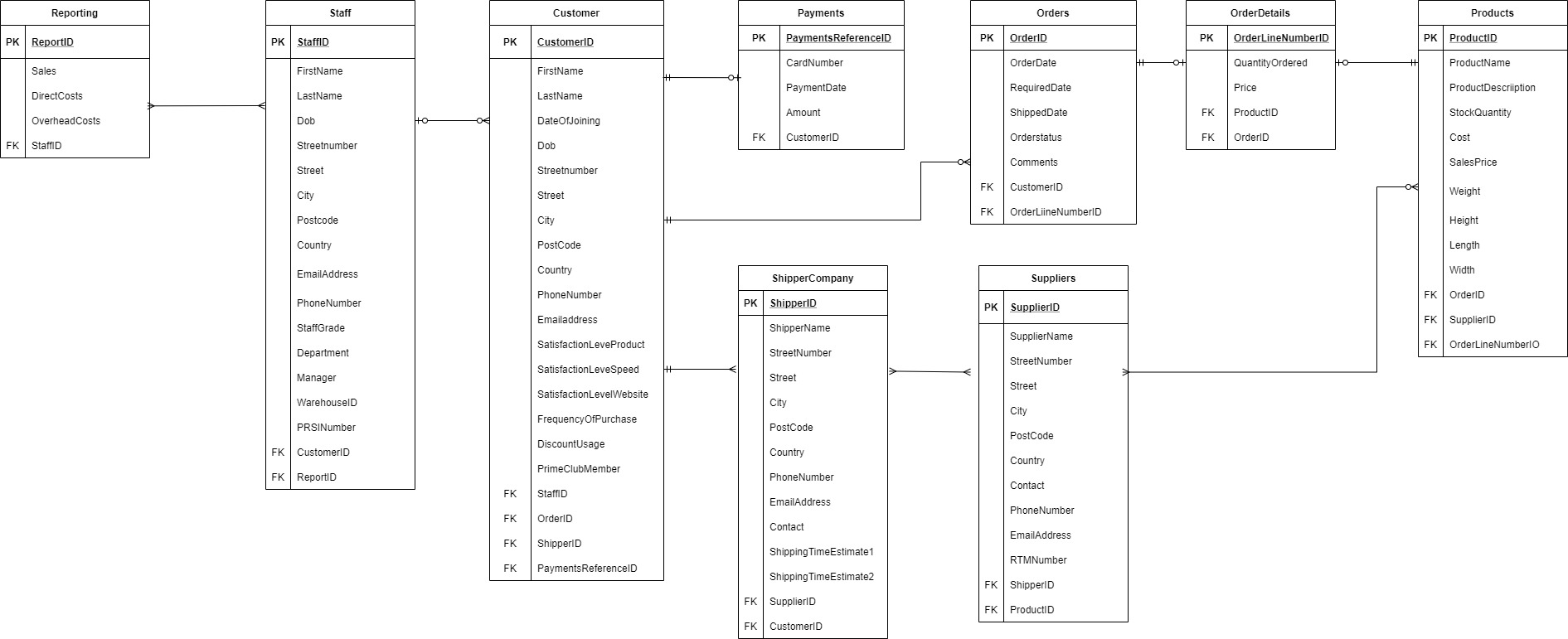
**2 & 3. Identify the relevant entities of the business with their respective attributes, entity types (strong or weak) and primary keys. You should include all the core entities of the business.**

**Relevant entities, attributes, entity types are described in the below ERD and Logical table diagram:**

**ERD diagram**



**Logical Table diagram**



**Normalisation grid**

Find inserted Normalisation diagram (please note that there is no export to JPG available in Excel (as was done for the ERD and Logical tables) so the excel file icon with link to the excel file is attached below



NB -> Below is the Excel file containing the Normalisation grid in more readable format



Adobe copy of Normalisation grid



NB – given that 3NF produced 23 tables, it was judged a better trade off to use 9 of these tables which loses very little in performance and only adds extremely minorly in data duplication.

**Part 2: Logical and physical design**

-- x20170386 Ryan Arthur

-- Project CA Part 2 - item 1 + 2 + 3 + 4

-- DROP DATABASE onlineretail;

CREATE DATABASE onlineretail;

USE onlineretail;

CREATE TABLE reporting (

reportid INT NOT NULL,

sales INT,

directcosts INT,

overheadcosts INT,

staffID INT,

PRIMARY KEY (reportid)

);

CREATE TABLE staff (

staffID INT NOT NULL,

firstname VARCHAR(20),

lastname VARCHAR(20),

dob DATE,

streetnumber INT,

street VARCHAR(35),

city VARCHAR(35),

postcode VARCHAR(14),

country VARCHAR(35),

emailaddress VARCHAR(40),

phonenumber VARCHAR(14),

staffgrade varchar(20),

department VARCHAR(15),

manager VARCHAR(21),

warehouseID INT,

prsinumber INT,

reportID INT,

customerID INT,

PRIMARY KEY (staffID)

);

CREATE TABLE customer (

customerID INT NOT NULL,

firstname VARCHAR(20),

lastname VARCHAR(20),

dob DATE,

dateofjoining DATE,

streetnumber INT,

street VARCHAR(35),

city VARCHAR(35),

postcode VARCHAR(14),

country VARCHAR(35),

phonenumber VARCHAR(14),

emailaddress VARCHAR(40),

satisfactionlevelproduct INT,

satisfactionlevelspeed INT,

satisfactionlevelwebsite INT,

frequencyofpurchase INT,

discountusage VARCHAR(2),

primeclubmember INT,

staffID INT,

orderID INT,

shipperID INT,

paymentsreferenceID INT,

PRIMARY KEY (customerID)

);

CREATE TABLE payments (

paymentsreferenceID INT NOT NULL,

cardnumber BIGINT(20),

paymentdate DATE,

amount INT,

customerID INT,

PRIMARY KEY (paymentsreferenceID)

);

CREATE TABLE orders (

orderID INT NOT NULL,

orderdate DATE,

requireddate DATE,

shippeddate DATE,

orderstatus VARCHAR(10),

comments VARCHAR(750),

customerID INT,

productID INT,

orderlinenumberID INT,

PRIMARY KEY (orderID)

);

CREATE TABLE orderdetails (

orderlinenumberID INT NOT NULL,

quantityordered INT,

price INT,

productID INT,

orderID INT,

PRIMARY KEY (orderlinenumberID)

);

CREATE TABLE products (

productID INT NOT NULL,

productname VARCHAR(50),

productdescription VARCHAR(750),

stockquantity INT,

cost DOUBLE,

salesprice DOUBLE,

weight DOUBLE,

height DOUBLE,

length DOUBLE,

width DOUBLE,

orderID INT,

supplierID INT,

orderlinenumberID INT,

PRIMARY KEY (productID)

);

CREATE TABLE suppliers (

supplierID INT NOT NULL,

suppliername VARCHAR(30),

streetnumber INT,

street VARCHAR(35),

city VARCHAR(50),

postcode VARCHAR(14),

country VARCHAR(50),

contact VARCHAR(35),

phonenumber VARCHAR(14),

emailaddress VARCHAR(40),

rtmnumber INT,

shipperID INT,

productID INT,

PRIMARY KEY (supplierID)

);

CREATE TABLE shippercompany (

shipperID INT NOT NULL,

shippername VARCHAR(50),

streetnumber INT,

street VARCHAR(35),

city VARCHAR(35),

postcode VARCHAR(14),

country VARCHAR(35),

contact VARCHAR(35),

phonenumber VARCHAR(14),

emailaddress VARCHAR(40),

shippingtimeestimate1 INT,

shippingtimeestimate2 INT,

supplierID INT,

customerID INT,

PRIMARY KEY (shipperID)

);

/\* Foreign Key additions \*/

ALTER TABLE reporting

ADD FOREIGN KEY (staffID) REFERENCES staff (staffID);

ALTER TABLE staff

ADD FOREIGN KEY (customerID) REFERENCES customer (customerID),

ADD FOREIGN KEY (reportID) REFERENCES reporting (reportID);

ALTER TABLE customer

ADD FOREIGN KEY (staffID) REFERENCES staff (staffID),

ADD FOREIGN KEY (orderID) REFERENCES orders (orderID),

ADD FOREIGN KEY (shipperID) REFERENCES shippercompany (shipperID),

ADD FOREIGN KEY (paymentsreferenceID) REFERENCES payments (paymentsreferenceID);

/\* ADD FOREIGN KEY (customerID, paymentsreferenceID) REFERENCES payments (customerID, paymentsreferenceID); point of failure \*/

ALTER TABLE payments

ADD FOREIGN KEY (customerID) REFERENCES customer (customerID);

ALTER TABLE orders

ADD FOREIGN KEY (customerID) REFERENCES customer (customerID),

ADD FOREIGN KEY (orderlinenumberID) REFERENCES orderdetails (orderlinenumberID);

ALTER TABLE orderdetails

ADD FOREIGN KEY (orderID) REFERENCES orders (orderID),

ADD FOREIGN KEY (productID) REFERENCES products (productID);

ALTER TABLE products

ADD FOREIGN KEY (orderID) REFERENCES orders (orderID),

ADD FOREIGN KEY (supplierID) REFERENCES suppliers (supplierID);

ALTER TABLE suppliers

ADD FOREIGN KEY (shipperID) REFERENCES shippercompany (shipperID),

ADD FOREIGN KEY (productID) REFERENCES products (productID);

ALTER TABLE shippercompany

ADD FOREIGN KEY (supplierID) REFERENCES suppliers (supplierID),

ADD FOREIGN KEY (customerID) REFERENCES customer (customerID);

/\* End of Foreign Key additions \*/

/\*

SELECT \* FROM reporting;

SELECT \* FROM staff;

SELECT \* FROM customer;

SELECT \* FROM payments;

SELECT \* FROM orders;

\*/

/\*

SELECT \* FROM orderdetails;

SELECT \* FROM products;

SELECT \* FROM suppliers;

SELECT \* FROM shippercompany;

\*/

**Part 3: Write SQL Statements to answer the following queries**

**Question 1 - Show all the details of the products that have a price greater than 100.**

-- x20170386 Ryan Arthur

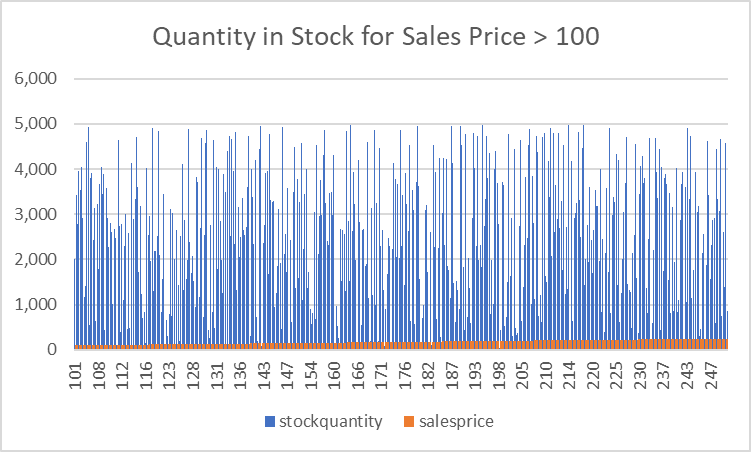
-- Project CA Part 3 - Question 1

-- "Show all the details of the products that have a price greater than 100""

use onlineretail;

Select \* from products

having salesprice > 100;



Description: The chart shows the quantity in stock on the vertical axis for all products (1 to 250) with price greater than €100. The tables in the below Excel file shows the tabular form where individual product descriptions can be viewed.



**Question 2 - Show all the products along with the supplier detail who supplied the products.**

-- x20170386 Ryan Arthur

-- Project CA Part 3 - Question 2

-- "Show all the products along the supplier detail who supplied the products"

use onlineretail;

select \* from suppliers;

select \* from products;

select \* from products left join suppliers on products.productID=suppliers.productID;



**Description:** The output is displayed in the table in the attached Excel file above. It would not fit in the word doc.

**Question 3 - Create a stored procedure that takes the start and end dates of the sales and display all the sales transactions between the start and the end dates.**

-- x20170386 Ryan Arthur

-- Project CA Part 3 - Question 3

-- "Create a stored procedure thast takes the start and end dates of the sales and display all the sales transactions betweeen the start and the end dates"

use onlineretail;

SELECT \*

FROM

payments

ORDER BY paymentdate

LIMIT 50;

USE onlineretail;

DROP procedure uspGetSales;

DELIMITER $$

CREATE PROCEDURE uspGetSales (in startdate date, in enddate date)

BEGIN

SELECT

paymentdate AS 'Date',

customerID AS 'CustomerID',

productID AS 'ProductID',

productname AS 'Product Name',

amount AS 'Sales Value',

paymentsreferenceID AS 'Payment Reference',

cardnumber AS 'Card#'

FROM

payments,

products

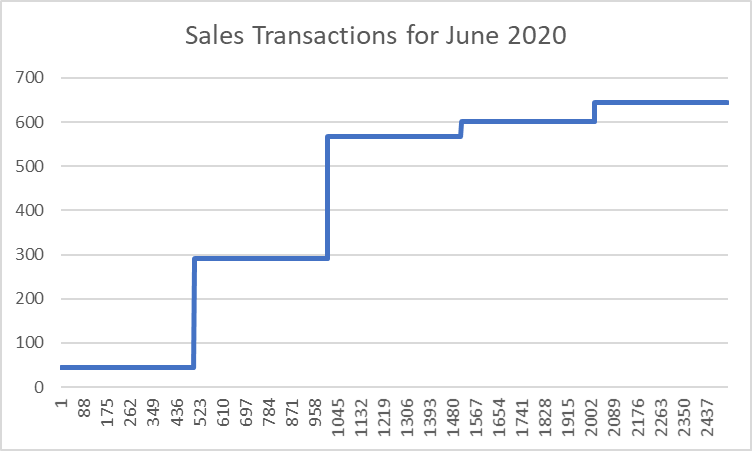
WHERE paymentdate >= startdate AND paymentdate <= enddate

ORDER BY paymentdate;

END$$

DELIMITER ;

CALL uspGetSales('2020/06/01', '2020/06/30');





Description: Find above a chart displaying the sales prices of products on the vertical axis and the number of transaction at that particular sales price for the month of June.

**Question 4 - Create a view that shows the total number of items a customer buys from the business in October 2020 along with the total price (use group by)**

-- x20170386 Ryan Arthur

-- Project CA Part 3 - Question 4

-- "Create a view that shows the total number of items a customer buys from the business in October 2020 along with

-- the total prices (use group by)"

use onlineretail;

SELECT

\*

FROM

orders

INNER JOIN

orderdetails USING (orderID)

LIMIT 5;

CREATE VIEW customerOrders

AS

SELECT

orderdate AS 'Order Date',

customerID AS 'CustomerID',

SUM(quantityordered) AS 'Total Number that Customer Bought in Time Period',

SUM(price) AS 'Total Value in Euro'

FROM

orders

INNER JOIN

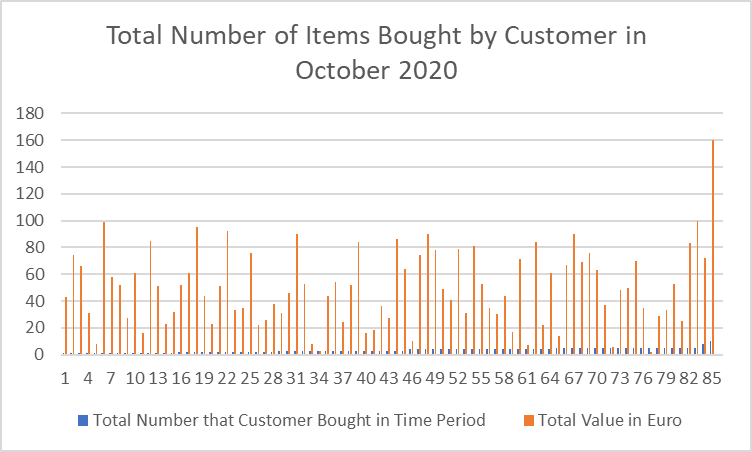
orderdetails USING (orderID)

WHERE orderdate >= '2020/10/01' AND orderdate <= '2020/10/31'

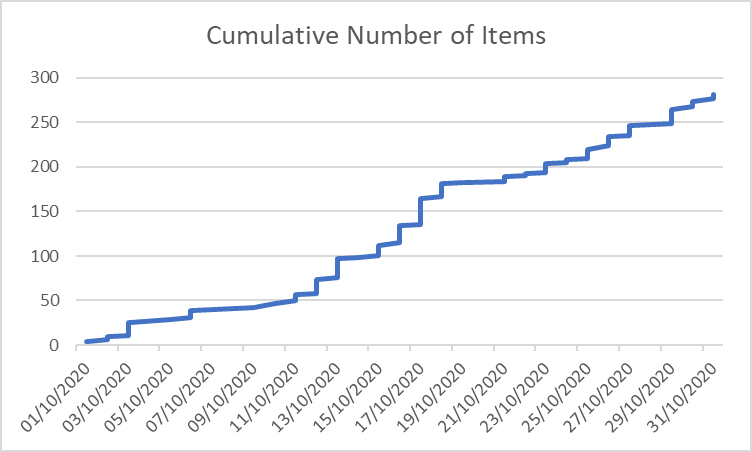
GROUP BY quantityordered, price;

-- ORDER BY orderdate ASC, customerID ASC, quantityordered DESC;

SELECT \* FROM customerOrders;



Description: In the chart we see the total euro value of items bought by customers in October 2020.



Description: In the chart we see the cumulative number of items bought by customers in October 2020.



**Question 5 - Create a trigger that adjusts the stock level every time a product is sold.**

-- x20170386 Ryan Arthur

-- Project CA Part 3 - Question 5

-- "Create a trigger that adjusts the stock level every time a product is sold. "

-- DROP TRIGGER after\_sold\_update ;

use onlineretail;

DELIMITER $$

CREATE TRIGGER after\_sold\_update

AFTER UPDATE ON payments

FOR EACH ROW

INSERT INTO products

SET action = 'update',

products.stockquantity = (products.stockquantity - 1),

products.cost = (products.cost + 1);

$$

DELIMITER ;

UPDATE payments

SET paymentdate = '2020/12/02'

WHERE customerID = '446';

/\*

SELECT \* FROM products;

SELECT \* FROM payments;

DESCRIBE payments;

DESCRIBE products;\*/

Eugene – I could not get this above code to run – it rejects/falls at the terms ‘action’

This also means there is no table or graph

**Question 6 - Create a report of the annual sales (2020) of the business showing the total number of products sold and the total price sold every month (use A group by with roll-up)**

-- x20170386 Ryan Arthur

-- Project CA Part 3 - Question 6

-- "Create a report of the annual sales (2020) of the business showing the total number of products sold and the

-- total prices sold every month (use A group by with roll-up"

use onlineretail;

SELECT

\*

FROM

orders

INNER JOIN

orderdetails USING (orderID)

GROUP BY orderdate

ORDER BY orderdate

LIMIT 5;

CREATE VIEW salestotalunitseurobymonth

AS

SELECT

orderdate AS 'Order Date',

SUM(quantityordered) AS 'Total Quantity',

SUM(price) AS 'Total Sales in Euro'

FROM

orders

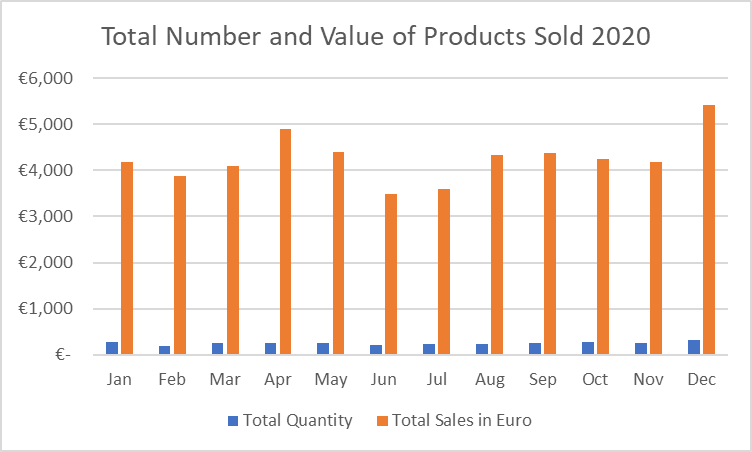
INNER JOIN

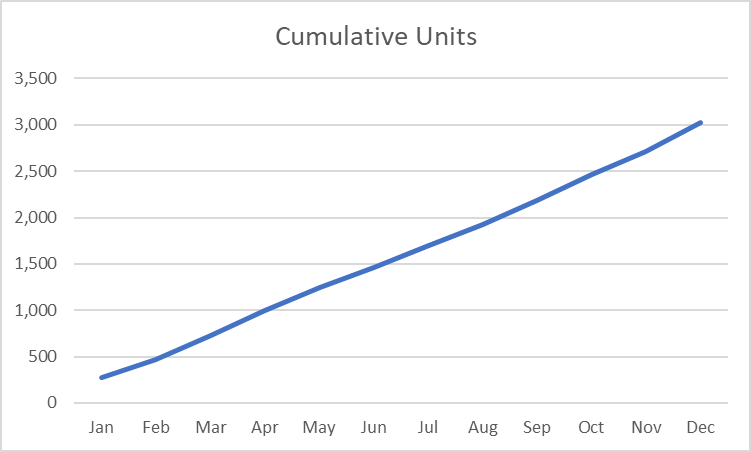
orderdetails USING (orderID)

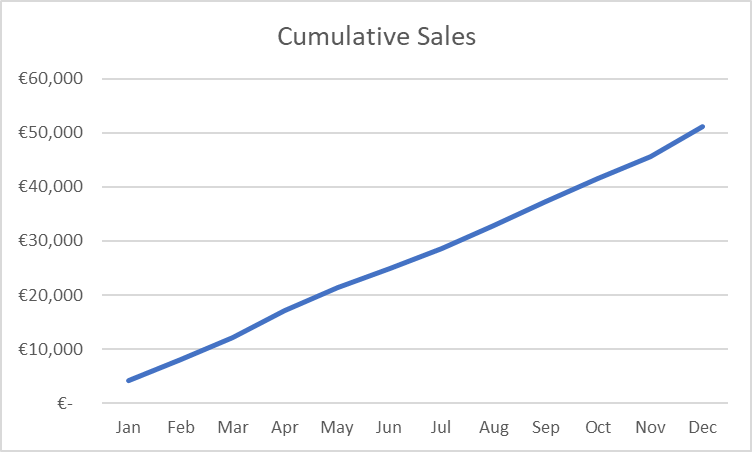
GROUP BY MONTH (orderdate)

ORDER BY orderdate ASC;

SELECT \* FROM salestotalunitseurobymonth;







Description: The chart displays the total number and value of products sold in 2020 showing a reasonably consistent level for both euro value of sales and quantity of products sold.



**Question 7 - Display the growth in sales/services (as a percentage) for your business, from the 1st month of opening until now.**

-- x20170386 Ryan Arthur

-- Project CA Part 3 - Question 7

-- "Display the growth in sales/services (as a percentage) for your business, form te 1st motnh of openning until now"

use onlineretail;

USE onlineretail;

-- DROP FUNCTION percentsalesgrowth;

delimiter $$

CREATE FUNCTION percentsalesgrowth()

RETURNS DOUBLE

DETERMINISTIC

BEGIN

DECLARE firstmonthsales INT;

DECLARE lastmonthsales INT;

DECLARE percentchange DOUBLE;

SELECT SUM(orderdetails.price) INTO firstmonthsales

FROM orderdetails, orders

WHERE orderdate >= '2020/01/01' AND orderdate <= '2020/01/31';

SELECT SUM(orderdetails.price) INTO lastmonthsales

FROM orderdetails, orders

WHERE orderdate >= '2020/01/01' AND orderdate <= '2020/12/31';

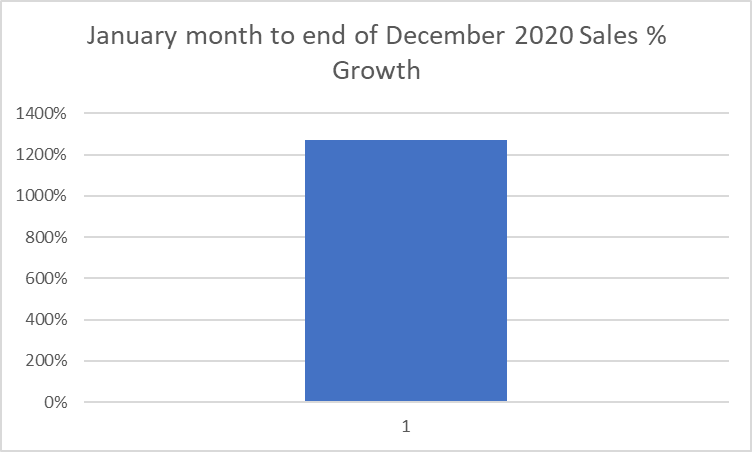
SET percentchange = round (((lastmonthsales / firstmonthsales) - 1) \* 100);

RETURN percentchange;

END $$

delimiter ;

SELECT percentsalesgrowth();



Description: The chart shows that the percentage change in sales at the end of 2020 comapared to the sales at the end of the first month’s operations is 1270%, which is understandable as the denominator is just one month’s sales and the numerator is 12 months sales.



**Question 8 - Delete all customers who never buy a product from the business.**

-- x20170386 Ryan Arthur

-- Project CA Part 3 - Question 8

-- "Delete all customers who never buy a product from the business"

USE onlineretail;

SELECT \* FROM customer

WHERE orderID = null;

DELETE FROM customer

WHERE orderID = null;

NB – All my customers made at least one purchase so I do not have null order values matching to customers.

**Submission checklist:**

1.Done - A pdf/word document containing the answers to each of the above questions. Please add all your answers including descriptions, SQL Scripts and diagrams (ERD, report charts or graphs in this file) for each of the questions where applicable.

2. An SQL script file that contains all the SQL statements used for part 2 and part 3. Each answer should have a clear label indicating the question number.

3. The dump file (self-contained SQL script) of the full database, including triggers and stored procedures.

**Submission Guideline:**

1. Put the Word document file and the SQL script file and the dump (self-contained) file in a folder and add them to a single zip folder. All your file names should use the following file name pattern.

[YourID]\_[YourProgramCode]\_[YourLastName]\_[YourFirstName]\_Description.doc/pdf

[YourID]\_[YourProgramCode]\_[YourLastName]\_[YourFirstName]\_SQL.sql

[YourID]\_[YourProgramCode]\_[YourLastName]\_[YourFirstName]\_Dump.sql

[YourID]\_[YourProgramCode]\_[YourLastName]\_[YourFirstName]\_Project.zip

Example:

X123456\_ HDSWT\_SEP1 \_Abgaz\_Yalemisew\_Description.doc

X123456\_ HDSWT\_SEP1 \_Abgaz\_Yalemisew\_SQL.sql

X123456\_ HDSWT\_SEP1 \_Abgaz\_Yalemisew\_Dump.sql

X123456\_ HDSWT\_SEP1 \_Abgaz\_Yalemisew\_Project.zip

Submission Deadline: Week 9

Marking rubrics: See associated file