Student ID Number *(Do not include student name as anonymous marking is implemented)*	TUD17454382		
Programme Title	Data Modelling & SQL Language		
Module Title	Data Modelling & SQL Language		
Module Code *(listed on Moodle and in LTAFP)*	QAC020C155A		
Module Convenor	Sharjeel Aslam		
Coursework Title	UShop		
Academic Declaration: *Students are reminded that the electronic copy of their essay may be checked, at any point during their degree, with Turnitin or other plagiarism detection software for plagiarized material.*			
Word Count		**Date Submitted**	

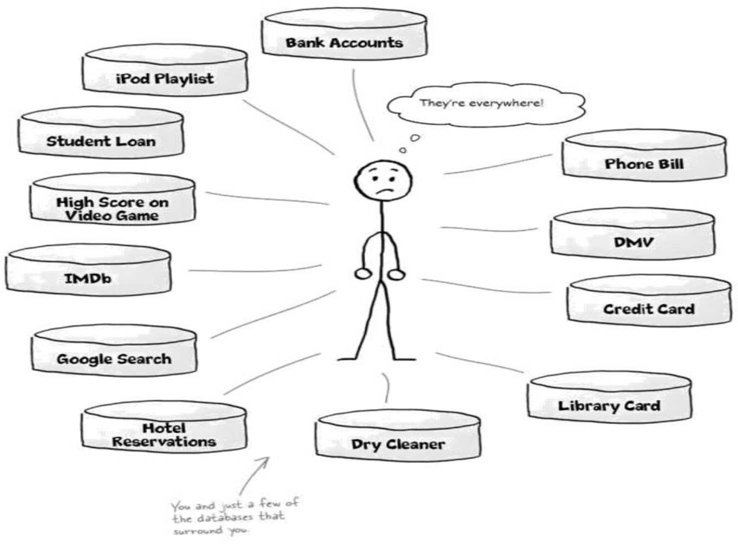
Introduction

Don't you just hate losing things and time for looking after them? Whether it's your car keys, that 25% off coupon for Tesco, or your application data, there nothing worse than not being able to keep up with your need's ….

This is about to finish… Finally, you can enjoy the more free time, you can be now chill and happy by knowing that your stuff is now stacked nicely in a nice "container".

Understanding the Database

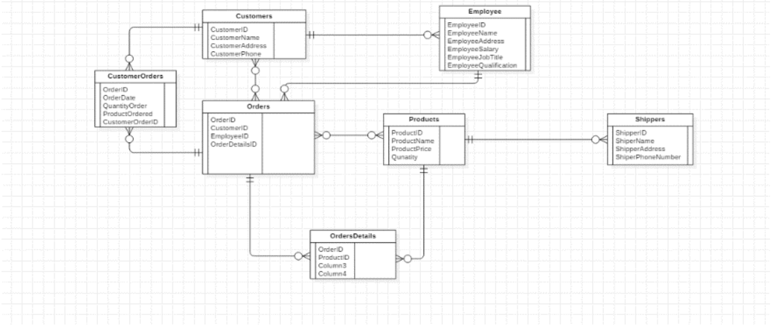
Every time you search online, go shopping, call information, use your TV, make a reservation, get a speeding ticket or buy groceries, a "container" is used to extract particular information from it. This "container" is a so-called DATABASE. A Database is a "container" that holds tables(folders) and other SQL structures related to those tables. Another terminology use "Entity" terminology for table and "Attributes" used for any information introduced in "Entity". Everything around us is a database.



Anatomy of a database.

A database contains a table (Entity's). A table is a structure inside a database that contains data (Attributes) organized in columns and rows. A table row contains all the information about one object inside it. All tables contain a column and a row.

All of the tables in an exceeding information ought to be connected in a way. For example, here are the tables I have to choose them to be used for my online shop named UShop.



A column may be a piece of information hold on by your table. A row may be a single set of columns that describe attributes of one issue. Both, together, make up a table.

Conceptual Modelling

By knowing this minimal information, I dare you to submerge with me in further knowledge in this wonderful part of Computing Technology**.** In what next, I will try to make you love SQL and everything about the database, by creating with you, a database for an online shop named "UShop".

UShop

UShop is an online retail company which is looking to develop an effective Relational Database Management System (RDBMS) to cater to the needs of their growing business. As a junior developer, you have been asked to develop a database for UShop which will satisfy the following information requirements:

Tasks:

1.1

Address (Postcode, Street, Town)

Customers (CustomersID, CustomersFName, CustomerLName, *Postcode**, CustomerPhone, *EmployeeID*)*

Employee (EmployeeID, EmployeeName, EmployeeStreet, EmployeeTown, *EmployeePostcode**, EmployeeSalary, EmployeeJobTitle, EmployeeQualification)

Orders (OrderID, *CustomerID*, EmployeeID**, OderDate, ProductReguest)

OrdersDetails (*OrderID*, ProductID*,* Quantity)

Products (ProductID, ProductName, ProductPrice, Quantity)

ProductsRequest (ProductRequestID, *ProductID**, NumberRequested, NumberDelivered, DateDelivered, DeliveryCost)

Shippers (ShipperID, ShipperName, ShipperAddress, ShipperPhoneNumber, *ProductID*)

Supplier (<u>SupplierID,</u> SupplierName, ContactPerson, Street, City, Postcode)

Logical Modelling

1.1.2

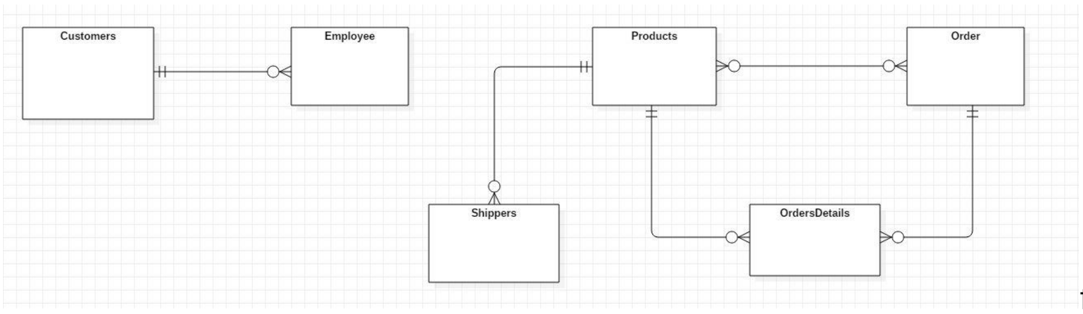


fig.1

In fig.1 I choose to make all the relationship according to my assumptions and necessity of my database. I also choose to create an extra entity for completing a MANY-to-Many relation. An example is a relationship between "Products" and "Order" which is a "**<u>many-to-many</u>**," reason why I choose to make a 3rd entity called "OrderDetails", where primary key from "Products" and "Order" 'travel' to "OrderDetails" and both together become "Composite Key"

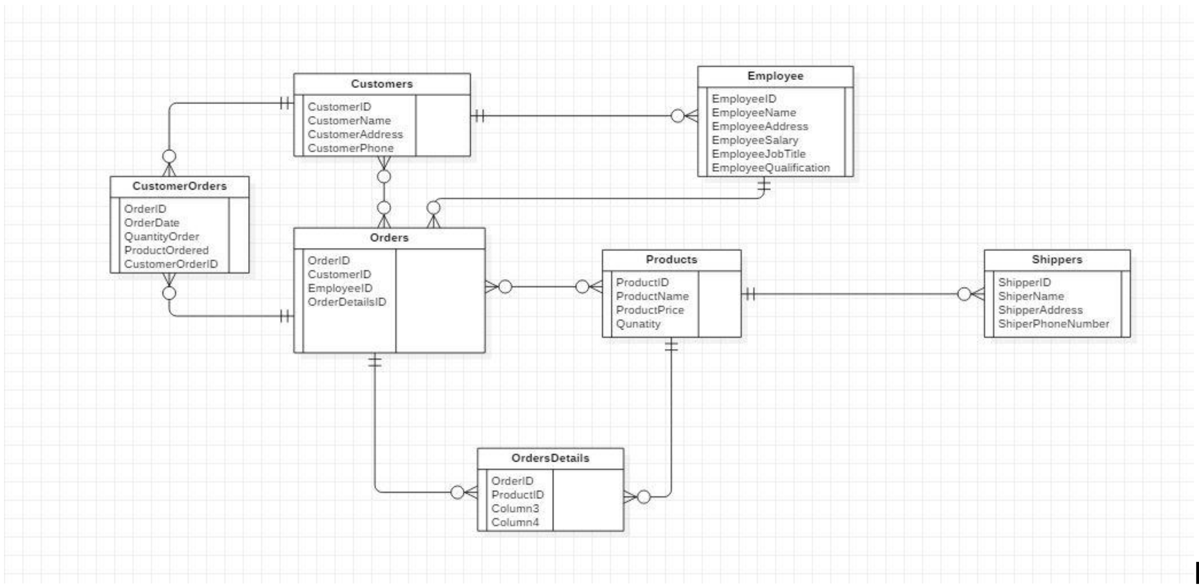


Fig.2

In Fig.2 I start to add to my entities all attributes necessary to fulfil my database requirements. Also, I choose to make 3rd Normalization form. I will speak a bit later about all Normalization steps.

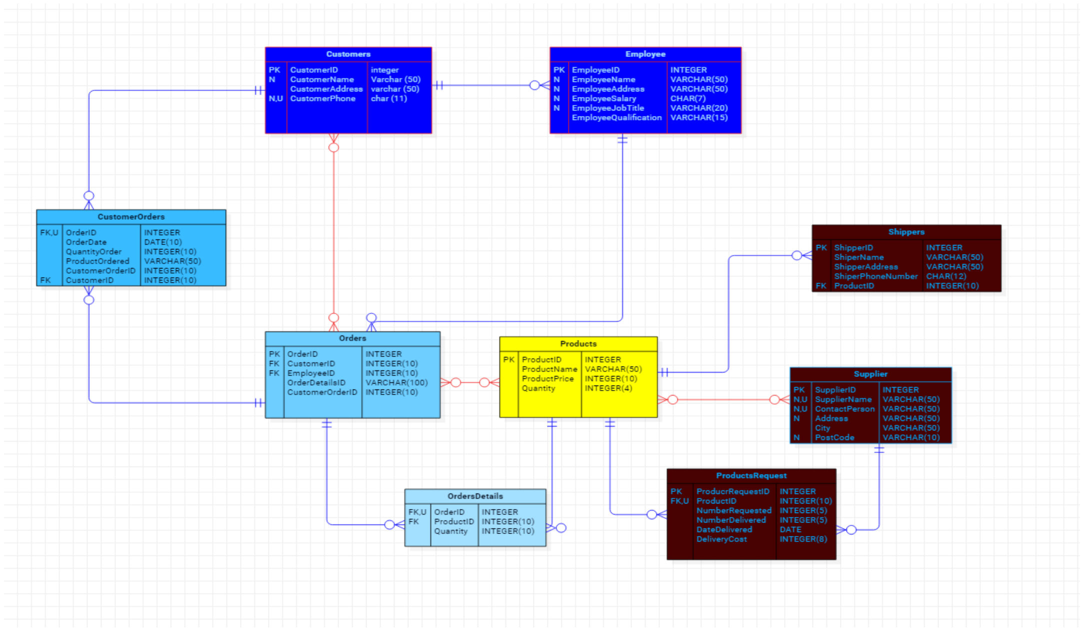

Customers
PK CustomerID integer
N CustomerName Varchar (50)
CustomerAddress varchar (50)
N,U CustomerPhone char (11)
Employee
PK EmployeeID INTEGER
N EmployeeName VARCHAR(50)
N EmployeeAddress VARCHAR(50)
N EmployeeSalary CHAR(7)
N EmployeeJobTitle VARCHAR(20)
EmployeeQualification VARCHAR(15)
CustomerOrders
FK,U OrderID INTEGER
OrderDate DATE(10)
QuantityOrder INTEGER(10)
ProductOrdered VARCHAR(50)
CustomerOrderID INTEGER(10)
FK CustomerID INTEGER(10)
Shippers
PK ShipperID INTEGER
ShiperName VARCHAR(50)
ShipperAddress VARCHAR(50)
ShiperPhoneNumber CHAR(12)
FK ProductID INTEGER(10)
Orders
PK OrderID INTEGER
FK CustomerID INTEGER(10)
FK EmployeeID INTEGER(10)
OrderDetailsID VARCHAR(100)
CustomerOrderID INTEGER(10)
Products
PK ProductID INTEGER
ProductName VARCHAR(50)
ProductPrice INTEGER(10)
Quantity INTEGER(4)
Supplier
PK SupplierID INTEGER
N,U SupplierName VARCHAR(50)
N,U ContactPerson VARCHAR(50)
N Address VARCHAR(50)
City VARCHAR(50)
N PostCode VARCHAR(10)
OrdersDetails
FK,U OrderID INTEGER
FK ProductID INTEGER(10)
Quantity INTEGER(10)
ProductsRequest
PK ProducrRequestID INTEGER
FK,U ProductID INTEGER(10)
NumberRequested INTEGER(5)
NumberDelivered INTEGER(5)
DateDelivered DATE
DeliveryCost INTEGER(8)

fig.3

In fig.3 I give to the attributes all the values they should contain in order to make the database to run smooth and to be easy and friendly for any user. This picture shows clearly where the Primary key and Foreign key are placed, and also can provide proof of a minimal form of constrained by usage of “Not Null” constraint. I use this constraint because I find it necessary for not allowing the user to skip any important /critical information.

SQL and database implementation

Task 2: Logical Model

2.1

“*Normalization of Database*

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy(repetition) and undesirable characteristics like Insertion, Update and Deletion Anomalies. It is a multi-step process that puts data into a tabular form, removing duplicated data from the relation tables.

Normalization is used for mainly two purposes,

- *Eliminating redundant(useless) data.*
- *Ensuring data dependencies make sense i.e. data is logically stored.* *(www.studytonight.com / no dated)* [1*]

“Normalization Rule

Normalization rules are divided into the following normal forms:

1. First Normal Form
2. Second Normal Form
3. Third Normal Form

First Normal Form (1NF)

For a table to be in the First Normal Form, it should follow the following 4 rules:

1. It should only have a single(atomic) valued attributes/columns.
2. Values stored in a column should be of the same domain
3. All the columns in a table should have unique names.
4. And the order in which data is stored does not matter.

In the next tutorial, we will discuss the **First Normal Form** in details.

Second Normal Form (2NF)

For a table to be in the Second Normal Form,

1. It should be in the First Normal form.
2. And, it should not have Partial Dependency.

To understand what is Partial Dependency and how to normalize a table to 2nd normal for, jump to the **Second Normal Form** tutorial.

Third Normal Form (3NF)

A table is said to be in the Third Normal Form when,

1. It is in the Second Normal form.
2. And, it doesn't have Transitive Dependency.

Here is the **Third Normal Form** tutorial. But we suggest you to first study about the second normal form and then head over to the third normal form. “(studytonight.com / no dated) 2*

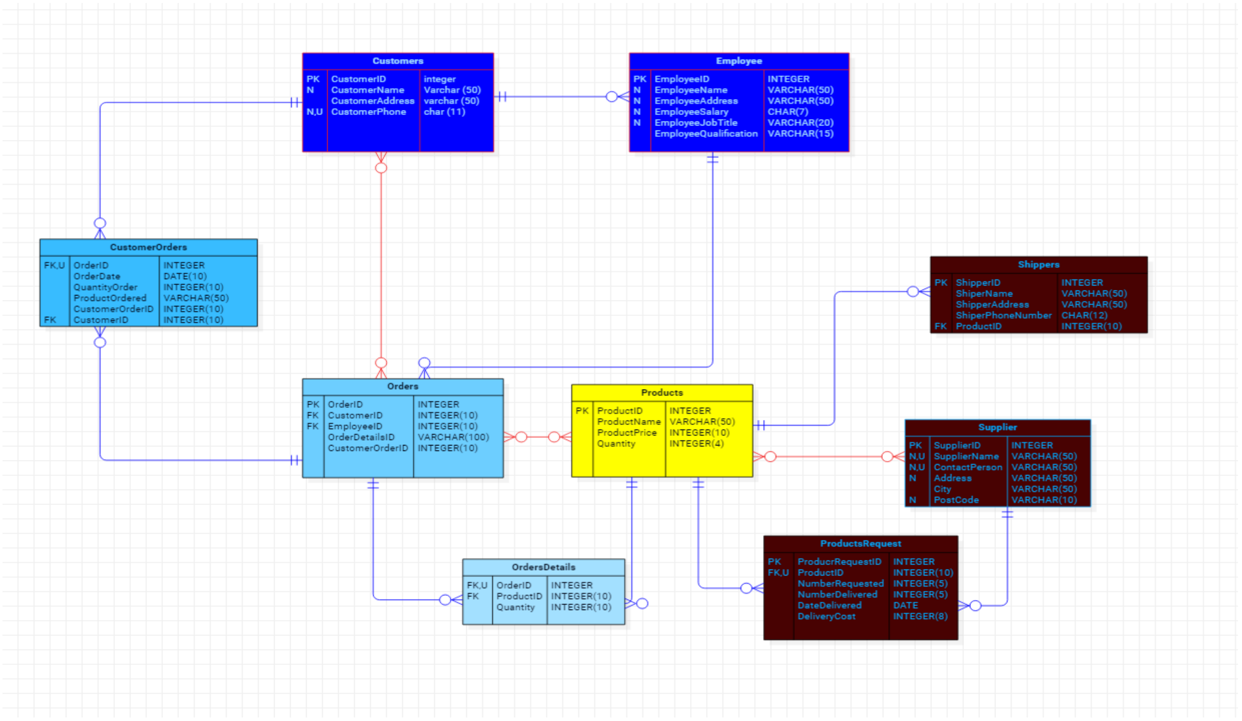
UNF	FNF	SNF	TNF
CustomerName **CustomerAddress**	CustomerFname CustomerLname Postcode CustomersStreet CustomersTown CustomerCounty CustomerPostcode CustomerPhone1 CustomerPhone2	**CustomerID** CustomerFname CustomerLname CustomerPostcode CustomerPhone Email	**CustomerID** CustomerFname CustomerLname CustomerPostcode CustomerPhone Email
		Postcode Street Town City	**Postcode** Street Town City
EmployeeName **EmployeeAddress**	EmployeeF_name EmployeeL_name Hire-date Manager 1 Manager 2 Phone number Salary Street Town County	**EmployeeID** EmployeeF_name EmployeeL_name Hire-date ManagerID Salary _Postcode*_	**EmployeeID** EmployeeF_name EmployeeL_name Hire-date ManagerID Salary _Postcode*_
ManagerName **ManagerAddress**	ManagerFname ManagerLname ManagerPhonenumber Manager	**ManagerID** ManagerFname ManagerLname ManagerPhonenumber	**ManagerID** ManagerFname ManagerLname ManagerPhonenumber _Postcode*_
		ManagerAddress **Postcode** Street Town county	ManagerAddress **Postcode** Street Town county
Order	_CustomerID*_ OrderDate ProductRequest	**OrderID** _CustomerID*_ OrderDate ProductRequest	
Orderdetails		_orderid*_ _Pid*_	_orderid*_ _Pid*_

		Quantity	*Quantity*
Productstitle **ProductPrice** **ProductQuantity**	Pdescription Ptitle Pquantity PoductPrice	**<u>Pid</u>** Pdescription Ptitle Pquantity PoductPrice	**<u>Pid</u>** Pdescription Ptitle Pquantity PoductPrice
SuppliersName **SuppliersAddress**	SuppliersName SuppliersAddress SuppliersPhone	**<u>SuppliersID</u>** SuppliersName SuppliersAddress SuppliersPhone *Employee_id**	**<u>SuppliersID</u>** SuppliersName SuppliersAddress SuppliersPhone *Employee_id**

Task

2.2

Now I wanna speak a little bit about de-normalization. In my opinion de-normalization process, it means that delivery not doing the normalization because is just creating more tables whit fewer data.


Customers
PK CustomerID integer
N CustomerName Varchar (50)
CustomerAddress varchar (50)
N,U CustomerPhone char (11)
Employee
PK EmployeeID INTEGER
N EmployeeName VARCHAR(50)
N EmployeeAddress VARCHAR(50)
N EmployeeSalary CHAR(7)
N EmployeeJobTitle VARCHAR(20)
EmployeeQualification VARCHAR(15)
CustomerOrders
FK,U OrderID INTEGER
OrderDate DATE(10)
QuantityOrder INTEGER(10)
ProductOrdered VARCHAR(50)
CustomerOrderID INTEGER(10)
FK CustomerID INTEGER(10)
Shippers
PK ShipperID INTEGER
ShiperName VARCHAR(50)
ShipperAddress VARCHAR(50)
ShiperPhoneNumber CHAR(12)
FK ProductID INTEGER(10)
Orders
PK OrderID INTEGER
FK CustomerID INTEGER(10)
FK EmployeeID INTEGER(10)
OrderDetailsID VARCHAR(100)
CustomerOrderID INTEGER(10)
Products
PK ProductID INTEGER
ProductName VARCHAR(50)
ProductPrice INTEGER(10)
Quantity INTEGER(4)
Supplier
PK SupplierID INTEGER
N,U SupplierName VARCHAR(50)
N,U ContactPerson VARCHAR(50)
N Address VARCHAR(50)
City VARCHAR(50)
N PostCode VARCHAR(10)
ProductsRequest
PK ProducrRequestID INTEGER
FK,U ProductID INTEGER(10)
NumberRequested INTEGER(5)
NumberDelivered INTEGER(5)
DateDelivered DATE
DeliveryCost INTEGER(8)
OrdersDetails
FK,U OrderID INTEGER
FK ProductID INTEGER(10)
Quantity INTEGER(10)

What I wanna say by that? If you take the example picture above you will notice that in the table “customers” I have “customerphone”. If I were to choose “customerphone” row to normalize I suppose to end up with another table called a Phone number and just one maybe two phone numbers. I think was much faster and easier for a user to access required data. As an example of normalization, I should have in table “customers” a row called “**<u>PhoneID</u>**” as a Primary Key and another table called Phones with “*CustomerID**” as Foreign Key.

Task:

2.3

The first relation I establish is between “Customers” and “Employee”.

This relationship is **one-to-many**. I manage to deduce that by using next 2 sentences:

“One Customer can be served by ONE employee.

One employee can server MANY customers”.

These two tables communicate between them by a primary key and a foreign key. “**CustomerID**” is a primary key in “Customer” entity. Instead “*EmployeeID*”, as a “Foreign key”, travel from “Employee” table (where is a Primary Key) straight to Customers table as a “Foreign key”.*

Next relationship I establish is between “Orders” and “Products” via a relationship

many-to-many. I use next two sentences to define relationship I need it to use it:

“One order can contain **many** products.

One product can be sold in **many** orders.”

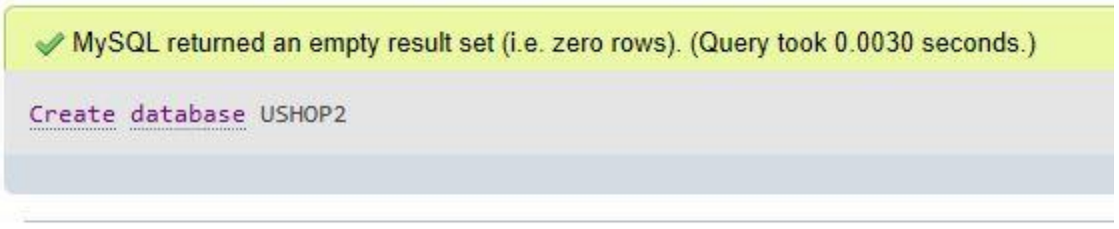
By definition this two tables cannot coexist together by themselves, that why it was necessary to create a 3rd table named “OrderDetails” where both Primary key from Orders and Products travels in OrderDetails and become foreign key.

Task 3: Physical Model

3.1

Creating database UShop

“ Create database USHOP2 “



```
✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0030 seconds.)

Create database USHOP2
```

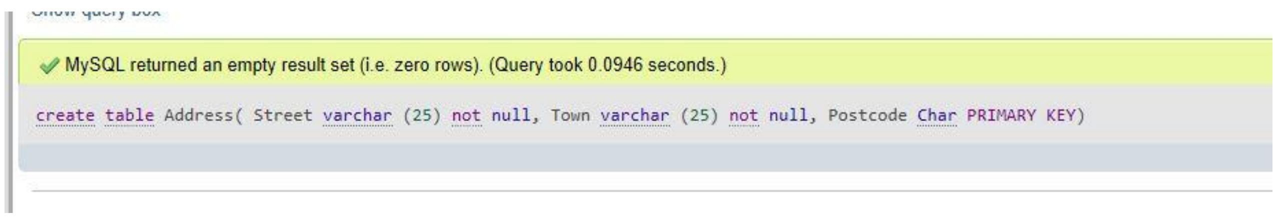
Creating table “Address”:

“Create table Address (

Street varchar (25) not null,

Town varchar (25) not null,

Postcode char (8) primary key) “



✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0946 seconds.)

```
create table Address( Street varchar (25) not null, Town varchar (25) not null, Postcode Char PRIMARY KEY)
```

Creating table “Customers”

“Create table Customers (

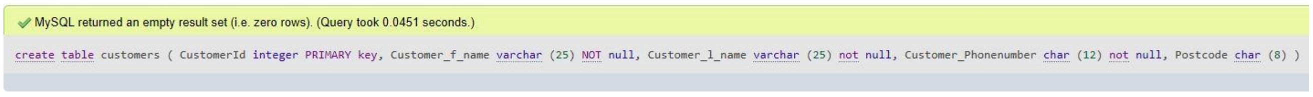
Cid integer Primary key,

Customer_f_name varchar (25) Not null,

Customer_l_name varchar (25) Not null,

Customer_Phonenumber char (12),

Postcode char (8)) “



✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0451 seconds.)

```
create table customers ( CustomerId integer PRIMARY key, Customer_f_name varchar (25) NOT null, Customer_l_name varchar (25) not null, Customer_Phonenumber char (12) not null, Postcode char (8) )
```

Creating table Employee

“Create table Employee (

EmployeeID integer Primary key,

EmployeeFname varchar (25) not null,

EmployeeStreet varchar (25) not null,

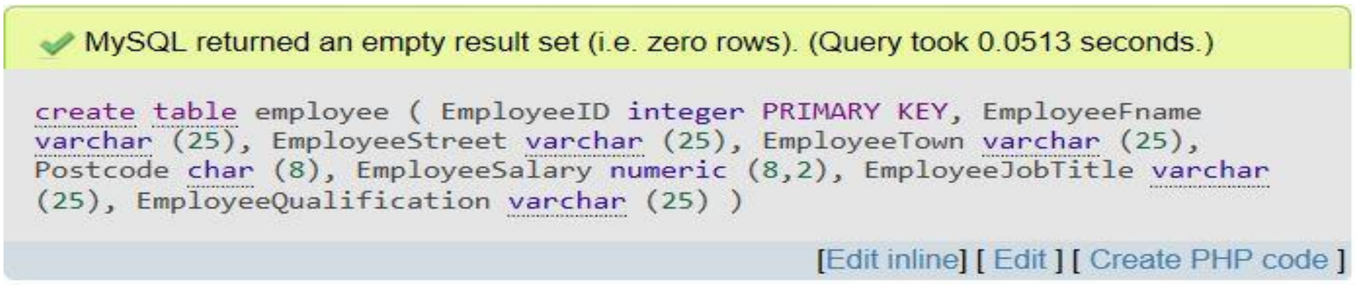
EmployeeTown varchar (25) not null,

Postcode char (8),

EmployeeSalary decimal (8,2),

EmployeeJobTitle varchar (25),

EmployeeQualification varchar (25)) “



```
✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0513 seconds.)

create table employee ( EmployeeID integer PRIMARY KEY, EmployeeFname
varchar (25), EmployeeStreet varchar (25), EmployeeTown varchar (25),
Postcode char (8), EmployeeSalary numeric (8,2), EmployeeJobTitle varchar
(25), EmployeeQualification varchar (25) )

[Edit inline] [ Edit ] [ Create PHP code ]
```

Creating table Manager

“create table Manager (

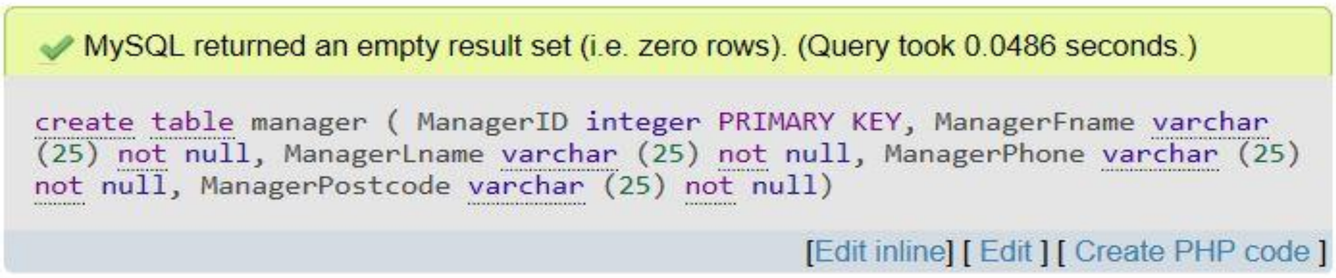
ManagerID integer Primary Key,

ManagerFname varchar (25),

ManagerLname varchar (25),

ManagerPhone char (8) not null,

ManagerPostcode varchar (25))”



```
✔ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0486 seconds.)

create table manager ( ManagerID integer PRIMARY KEY, ManagerFname varchar
(25) not null, ManagerLname varchar (25) not null, ManagerPhone varchar (25)
not null, ManagerPostcode varchar (25) not null)

[Edit inline] [ Edit ] [ Create PHP code ]
```

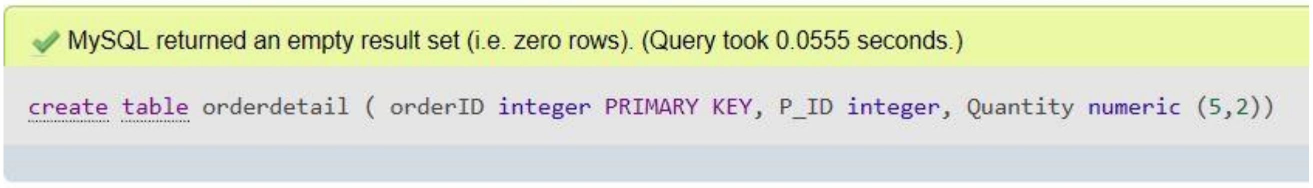
Creating table OrderDetails

“create table orderdetail (

orderID integer primary key,

P_ID integer,

Quantity decimal (5,2)) “


MySQL returned an empty result set (i.e. zero rows). (Query took 0.0555 seconds.)

```
create table orderdetail ( orderID integer PRIMARY KEY, P_ID integer, Quantity numeric (5,2))
```

Creating table Orders

“create table orders

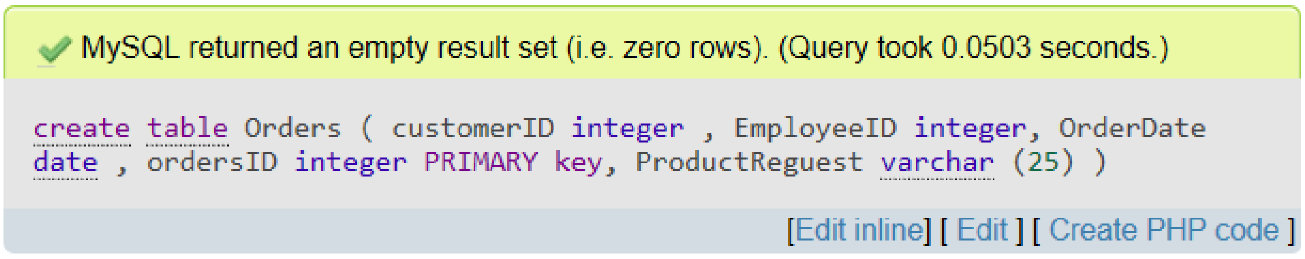
CustomerID integer,

EmployeeID integer,

OrderDate date,

ordersID integer primary key,

ProductReguest varchar (25)) “


MySQL returned an empty result set (i.e. zero rows). (Query took 0.0503 seconds.)

```
create table Orders ( customerID integer , EmployeeID integer, OrderDate
date , ordersID integer PRIMARY key, ProductReguest varchar (25) )
```

[Edit inline] [Edit] [Create PHP code]

Creating table Product

“create table products (

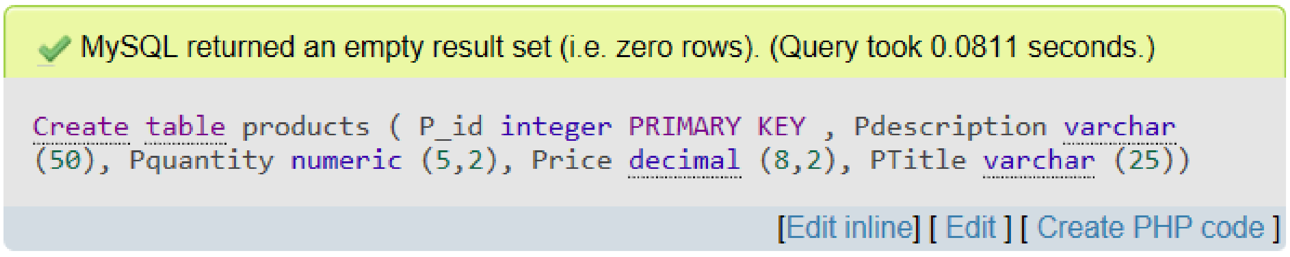
Pid integer primary key,

Pquantity varchar (50),

Price varchar (10),

Ptitle varchar (20),

Pdescription varchar (250)) “


MySQL returned an empty result set (i.e. zero rows). (Query took 0.0811 seconds.)

```
Create table products ( P_id integer PRIMARY KEY , Pdescription varchar
(50), Pquantity numeric (5,2), Price decimal (8,2), PTitle varchar (25))
```

[Edit inline] [Edit] [Create PHP code]

Creating table Suppliers

“create table Suppliers (

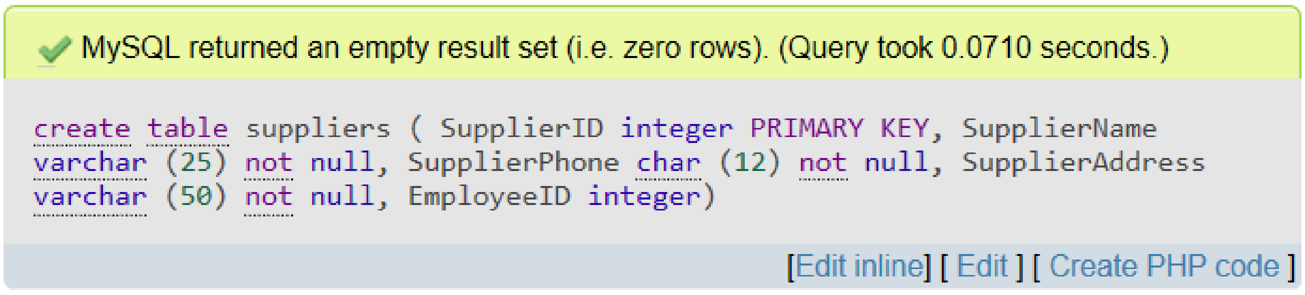
SuppliersID integer primary key,

SupplierName varchar (25),

SuppliersPhone char (12),

SuppliersAddress varchar (50),

EmployeeID integer) “



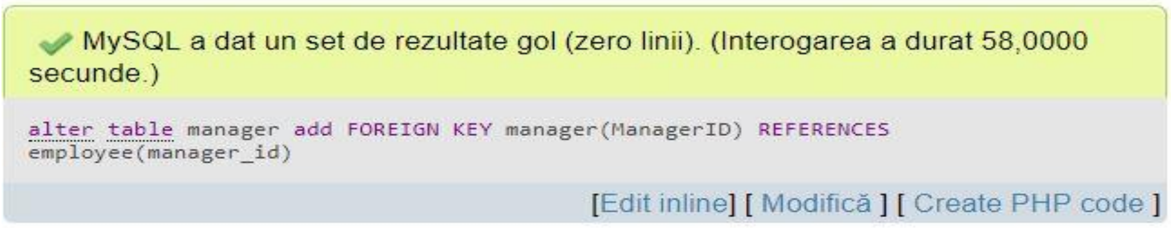
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0710 seconds.)

```
create table suppliers ( SupplierID integer PRIMARY KEY, SupplierName
varchar (25) not null, SupplierPhone char (12) not null, SupplierAddress
varchar (50) not null, EmployeeID integer)
```

Creating Foreign keys

“ Alter table manager

Add foreign key manager (ManagerID) REFERENCES employee(manager_id) “



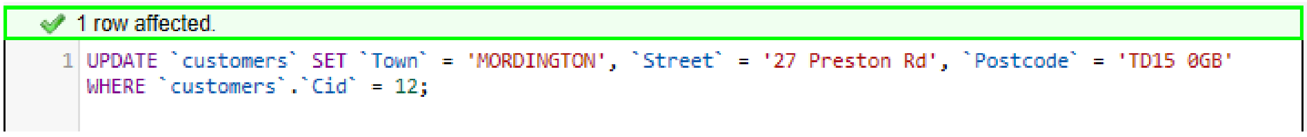
MySQL a dat un set de rezultate gol (zero linii). (Interogarea a durat 58,0000 secunde.)

```
alter table manager add FOREIGN KEY manager(ManagerID) REFERENCES
employee(manager_id)
```

Update table

“Update ‘customers’ set ‘town’ = ‘MORDINGTON’, ‘street’ = ’27 Preston Rd’, ‘Postcode’ = ‘TD15 0GB’

Where ‘customers’.’Cid’ = 12;



1 row affected.

```
UPDATE `customers` SET `Town` = 'MORDINGTON', `Street` = '27 Preston Rd', `Postcode` = 'TD15 0GB'
WHERE `customers`.`Cid` = 12;
```

Task

3.2

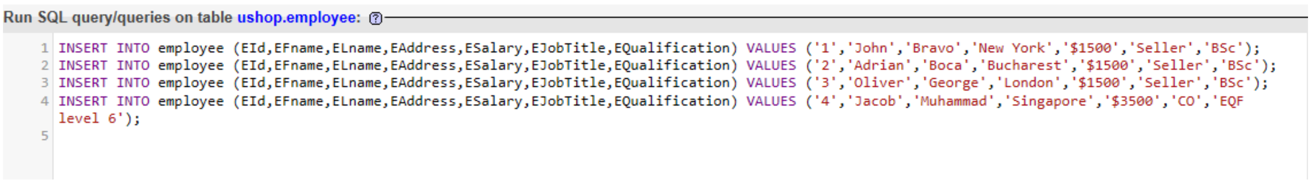
“

Insert into employee (Eid,EFname,Eaddress,ESalary,EJobTitle,EQualification) VALUES (1,’John’,’Bravo’,’New York’,’$1500’,’Seller’,’BSc’);

Insert into employee (Eid,EFname,Eaddress,ESalary,EJobTitle,EQualification) VALUES (2,’Adrian,’Boca’,’Bucuresto’,’$1500’,’Seller’,’BSc’);

Insert into employee (Eid,EFname,Eaddress,ESalary,EJobTitle,EQualification) VALUES (3,’Olliver’,’George’,’London’,’$1500’,’Seller’,’BSc’);

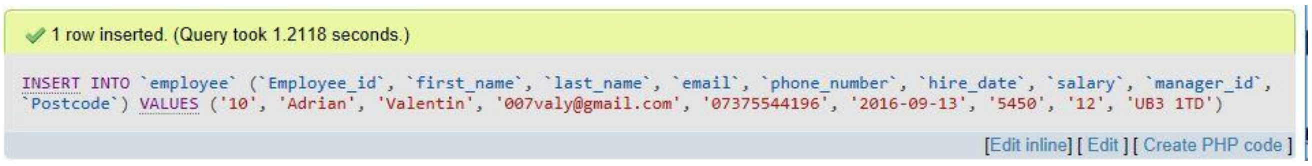
Insert into employee (Eid,EFname,Eaddress,ESalary,EJobTitle,EQualification) VALUES (4,’Jacob’,’Muhammad’,’Singapore’,’$3500’,’CEO’,’EQF’);



Run SQL query/queries on table ushop.employee:

```
INSERT INTO employee (EId,EFname,ELname,EAddress,ESalary,EJobTitle,EQualification) VALUES ('1','John','Bravo','New York','$1500','Seller','BSc');
INSERT INTO employee (EId,EFname,ELname,EAddress,ESalary,EJobTitle,EQualification) VALUES ('2','Adrian','Boca','Bucharest','$1500','Seller','BSc');
INSERT INTO employee (EId,EFname,ELname,EAddress,ESalary,EJobTitle,EQualification) VALUES ('3','Oliver','George','London','$1500','Seller','BSc');
INSERT INTO employee (EId,EFname,ELname,EAddress,ESalary,EJobTitle,EQualification) VALUES ('4','Jacob','Muhammad','Singapore','$3500','CO','EQF
level 6');
```

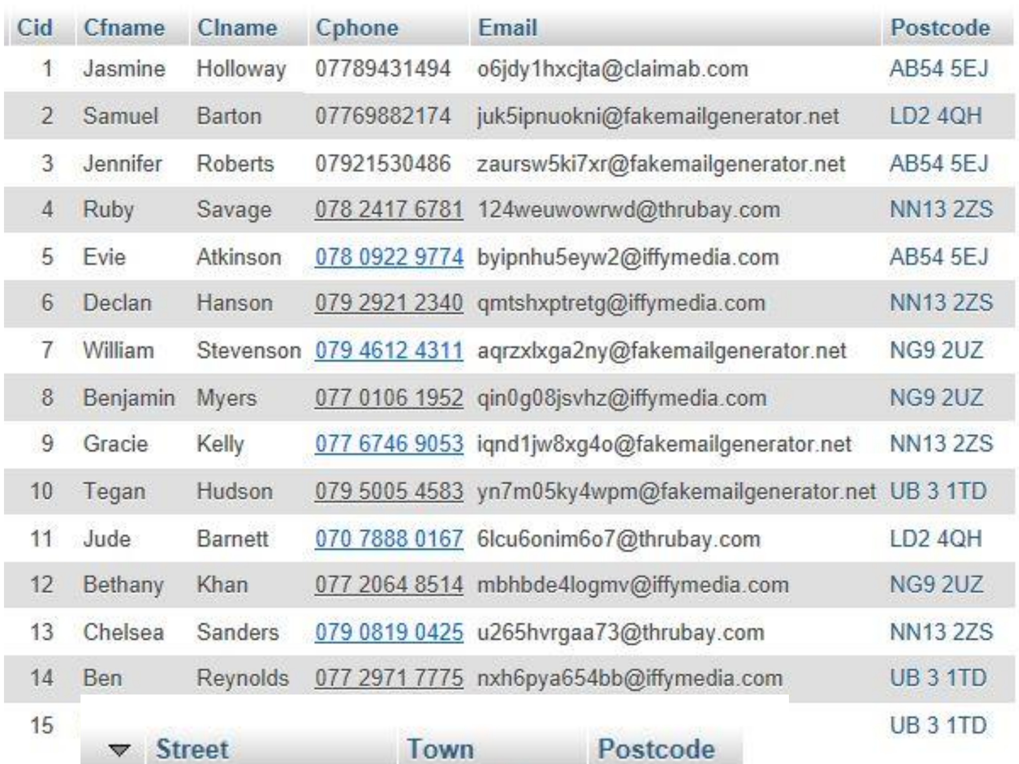
```
INSERT INTO `employee` (`Employee_id`, `first_name`, `last_name`, `email`,
`phone_number`, `hire_date`, `salary`, `manager_id`, `Postcode`) VALUES ('10',
'Adrian', 'Valentin', '007valy@gmail.com', '07375544196', '2016-09-13', '5450',
'12', 'UB3 1TD')
```



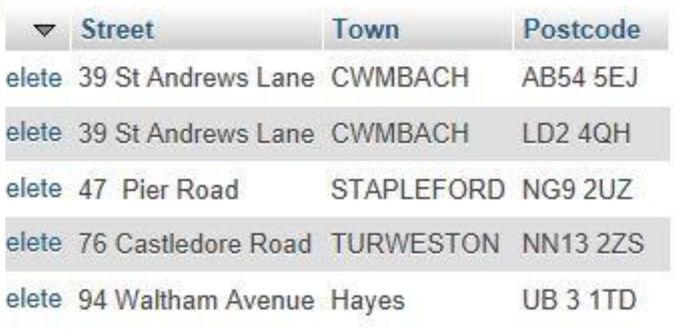
1 row inserted. (Query took 1.2118 seconds.)

```
INSERT INTO `employee` (`Employee_id`, `first_name`, `last_name`, `email`, `phone_number`, `hire_date`, `salary`, `manager_id`,
`Postcode`) VALUES ('10', 'Adrian', 'Valentin', '007valy@gmail.com', '07375544196', '2016-09-13', '5450', '12', 'UB3 1TD')
```

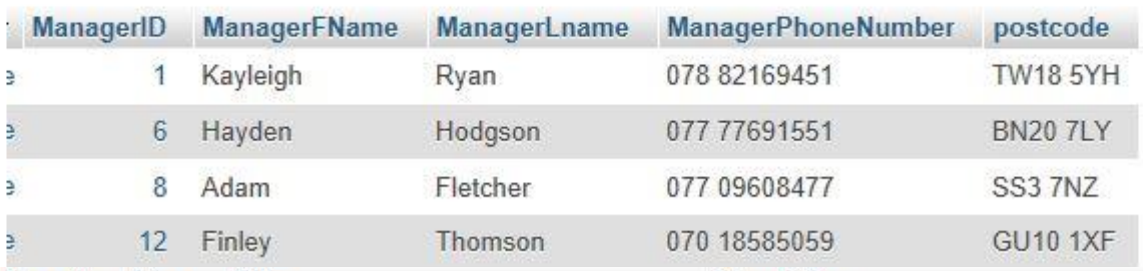
I use some insertion code for all another table. Prove I have in the next picture



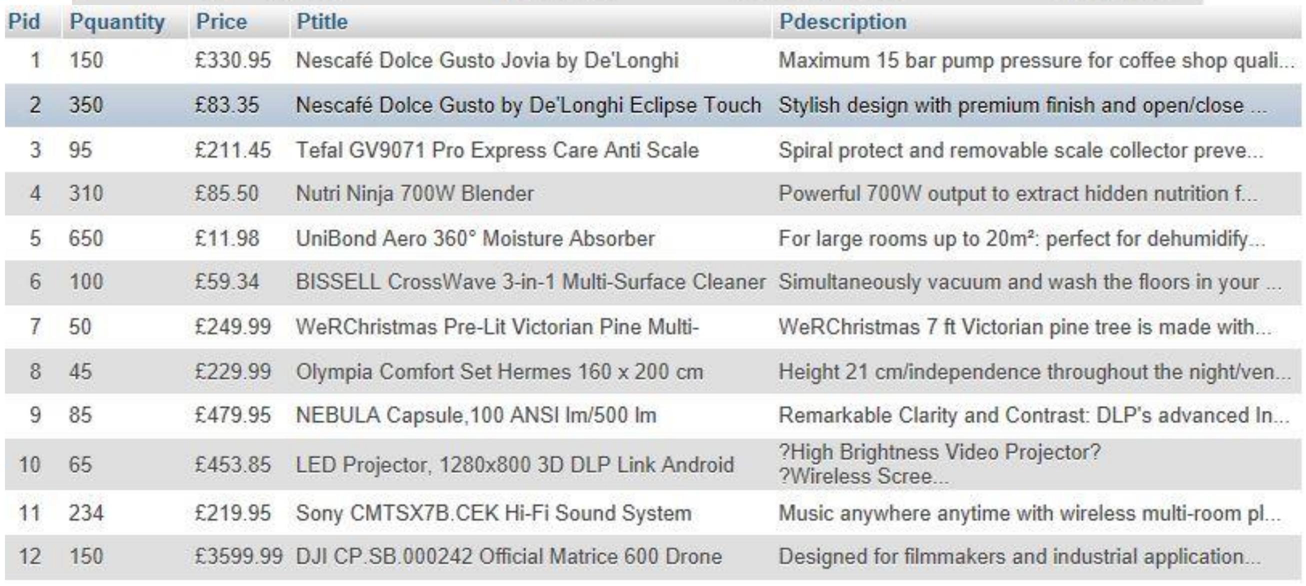
Cid	Cfname	Clname	Cphone	Email	Postcode
1	Jasmine	Holloway	07789431494	o6jdy1hxcjta@claimab.com	AB54 5EJ
2	Samuel	Barton	07769882174	juk5ipnuokni@fakemailgenerator.net	LD2 4QH
3	Jennifer	Roberts	07921530486	zaursw5ki7xr@fakemailgenerator.net	AB54 5EJ
4	Ruby	Savage	078 2417 6781	124weuwowrwd@thrubay.com	NN13 2ZS
5	Evie	Atkinson	078 0922 9774	byipnhu5eyw2@iffymedia.com	AB54 5EJ
6	Declan	Hanson	079 2921 2340	qmtshxptretg@iffymedia.com	NN13 2ZS
7	William	Stevenson	079 4612 4311	aqrzxlxga2ny@fakemailgenerator.net	NG9 2UZ
8	Benjamin	Myers	077 0106 1952	qin0g08jsvhz@iffymedia.com	NG9 2UZ
9	Gracie	Kelly	077 6746 9053	iqnd1jw8xg4o@fakemailgenerator.net	NN13 2ZS
10	Tegan	Hudson	079 5005 4583	yn7m05ky4wpm@fakemailgenerator.net	UB 3 1TD
11	Jude	Barnett	070 7888 0167	6lcu6onim6o7@thrubay.com	LD2 4QH
12	Bethany	Khan	077 2064 8514	mbhbde4logmv@iffymedia.com	NG9 2UZ
13	Chelsea	Sanders	079 0819 0425	u265hvrgaa73@thrubay.com	NN13 2ZS
14	Ben	Reynolds	077 2971 7775	nxh6pya654bb@iffymedia.com	UB 3 1TD
15					UB 3 1TD



	Street	Town	Postcode
elete	39 St Andrews Lane	CWMBACH	AB54 5EJ
elete	39 St Andrews Lane	CWMBACH	LD2 4QH
elete	47 Pier Road	STAPLEFORD	NG9 2UZ
elete	76 Castledore Road	TURWESTON	NN13 2ZS
elete	94 Waltham Avenue	Hayes	UB 3 1TD



	ManagerID	ManagerFName	ManagerLname	ManagerPhoneNumber	postcode
e	1	Kayleigh	Ryan	078 82169451	TW18 5YH
e	6	Hayden	Hodgson	077 77691551	BN20 7LY
e	8	Adam	Fletcher	077 09608477	SS3 7NZ
e	12	Finley	Thomson	070 18585059	GU10 1XF



Pid	Pquantity	Price	Ptitle	Pdescription
1	150	£330.95	Nescafé Dolce Gusto Jovia by De'Longhi	Maximum 15 bar pump pressure for coffee shop quali...
2	350	£83.35	Nescafé Dolce Gusto by De'Longhi Eclipse Touch	Stylish design with premium finish and open/close ...
3	95	£211.45	Tefal GV9071 Pro Express Care Anti Scale	Spiral protect and removable scale collector preve...
4	310	£85.50	Nutri Ninja 700W Blender	Powerful 700W output to extract hidden nutrition f...
5	650	£11.98	UniBond Aero 360° Moisture Absorber	For large rooms up to 20m²: perfect for dehumidify...
6	100	£59.34	BISSELL CrossWave 3-in-1 Multi-Surface Cleaner	Simultaneously vacuum and wash the floors in your ...
7	50	£249.99	WeRChristmas Pre-Lit Victorian Pine Multi-	WeRChristmas 7 ft Victorian pine tree is made with...
8	45	£229.99	Olympia Comfort Set Hermes 160 x 200 cm	Height 21 cm/independence throughout the night/ven...
9	85	£479.95	NEBULA Capsule,100 ANSI lm/500 lm	Remarkable Clarity and Contrast: DLP's advanced In...
10	65	£453.85	LED Projector, 1280x800 3D DLP Link Android	?High Brightness Video Projector? ?Wireless Scree...
11	234	£219.95	Sony CMTSX7B.CEK Hi-Fi Sound System	Music anywhere anytime with wireless multi-room pl...
12	150	£3599.99	DJI CP.SB.000242 Official Matrice 600 Drone	Designed for filmmakers and industrial application...

Referencing:

1*

DBMS&SQL (no dated) *Normalization of Database.* Available at:
https://www.studytonight.com/dbms/database-normalization.php
Accessed (11/11/18)

2*

www.studytonight.com (no dated) *DBMS & SQL.* Available at:
https://www.studytonight.com/dbms/database-normalization.php
Accessed (11/11/18)

APPENDIX

ushop (2).sql

Pictures.rar