Introduction to Databases

CA1 - 20/11/22

Part 1: Conceptual Design:

Exercise 1

I have decided to model this CA around a Supermarket. A supermarket uses a database to record daily transactions along with many other pieces of data. Daily transactions are stored in the OrderDetails table.

Customer information is stored to track spending and purchasing habits. Customers can sign up for a loyalty card which is connected to their customerID number. Special offers can be offered to these customers to encourage return shoppers. The supermarket can also get in touch with the customer to update them with any upcoming products that might interest them.

Employee data is also maintained in the database. Storing each employees information allows the supermarket to get in touch with the employee if there is any need to do so, for example if the employee has not made it to work that day. Employee positions are also important to be stored in the database as any changing of positions needs to be updated for payroll.

Supplier data is stored to ensure that suppliers are paid the correct agreed upon amount and to ensure relations between supermarket and suppliers are maintained. What products come from which suppliers and product amounts are extremely important to keep a record of to ensure waste is kept to a minimum.

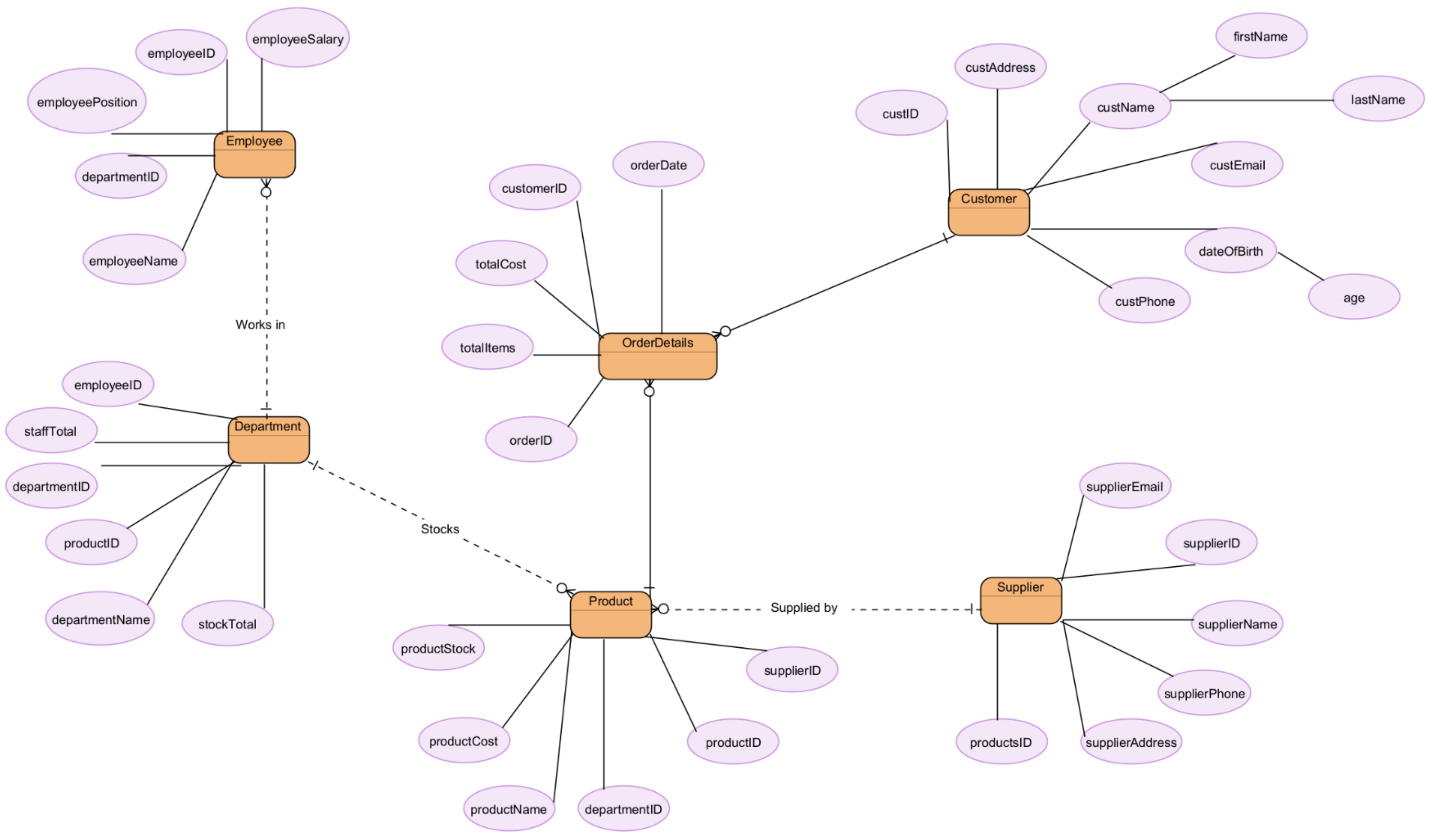
Product data is stored to ensure prices are tracked and sales numbers can be tracked. Maintaining a record of each individual product is incredibly important for stock reasons and balancing budgets and is very helpful to identify which products are selling well and making profits for the supermarket.

Department info is tracked to ensure that the supermarket can identify which departments are performing well and whether a department is staffed correctly. Without this information the supermarket could be losing profits without realising due to short staffing, under stocking or incorrectly stocking departments.

Order information is incredibly important to store for the supermarket. By storing each order information, the supermarket can see exactly how many transactions were completed on a daily basis. Customers can return products by returning a receipt of the order.

Exercise 2

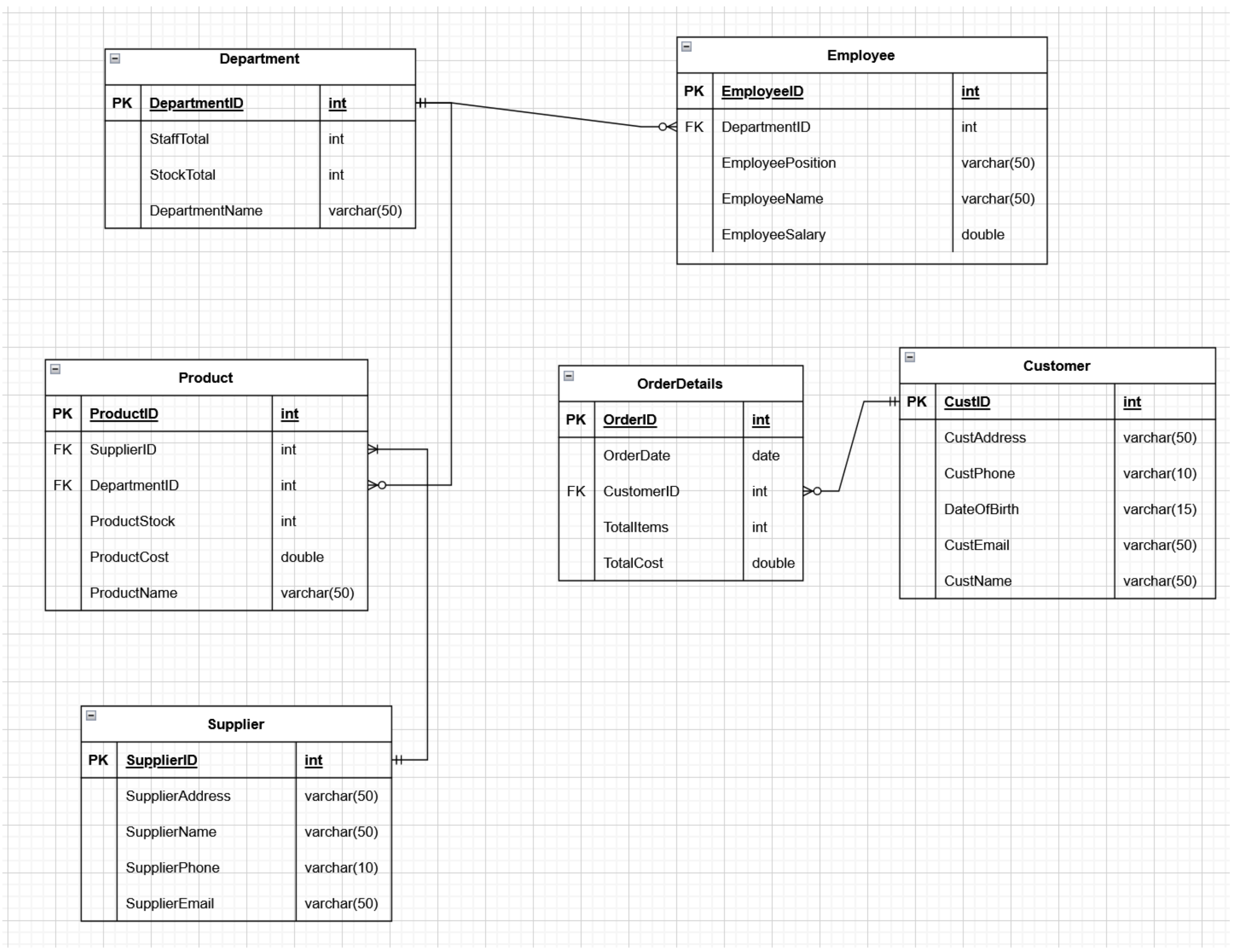
Conceptual ERD



Derived attributes:

- firstName and lastName attributes of the Customer entity are derived from the name attribute
- age attribute of the Customer entity is derived from the dateOfBirth attribute

Relational Model

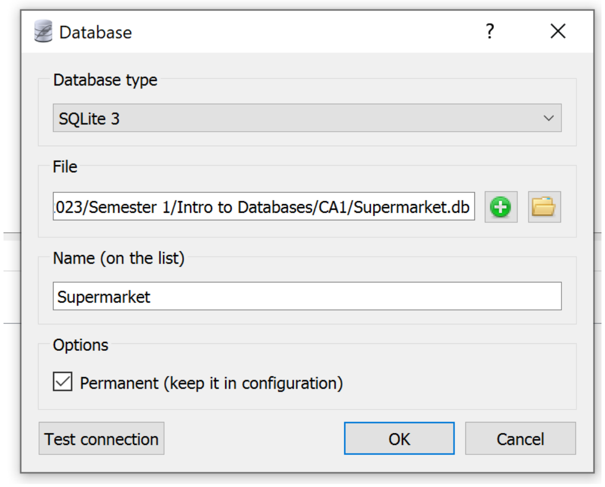


Part 2: Physical Design

(All SQL queries executed using SQLiteStudio)

Exercise 1

Creating the database:



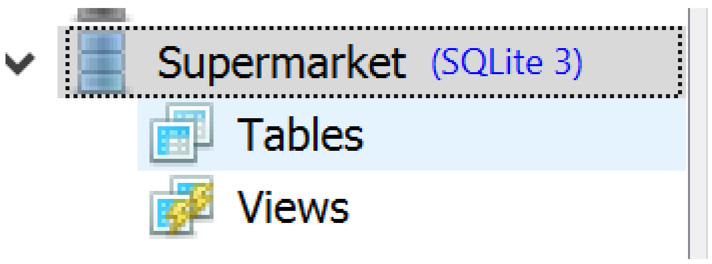
Database

Database type: SQLite 3

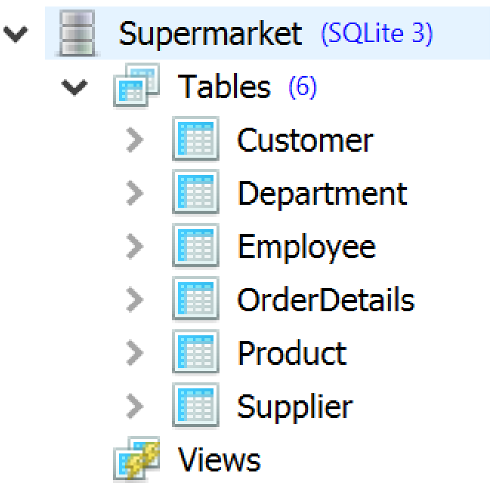
File: 023/Semester 1/Intro to Databases/CA1/Supermarket.db

Name (on the list): Supermarket

Options: ☑ Permanent (keep it in configuration)



Exercise 2



Exercise 3

```
INSERT  INTO Product
VALUES (1234567, 234, 7, 55, 4.95, "Apple tart");

INSERT INTO Employee
VALUES (4043, 10, "Department Lead", "Denis Murray", 24000)

INSERT INTO Department
VALUES (4, 243, 15000, "Hardware")
```

Exercise 4

I used Mockaroo to generate the data for the database. Please see the attached Denis_Murray_SQL.sql file for the sql statements generated for this exercise or follow the links for schemas used:

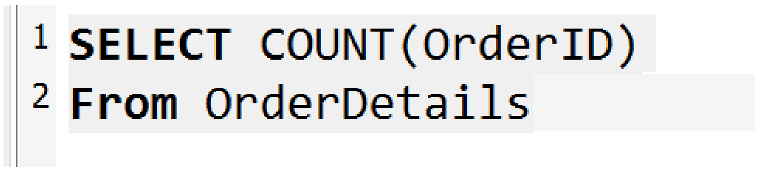
- https://www.mockaroo.com/cb711360
- https://www.mockaroo.com/66fdf1c0
- https://www.mockaroo.com/06e3bb30
- https://www.mockaroo.com/437c9350
- https://www.mockaroo.com/a21601f0
- https://www.mockaroo.com/2f5a1570

Part 3

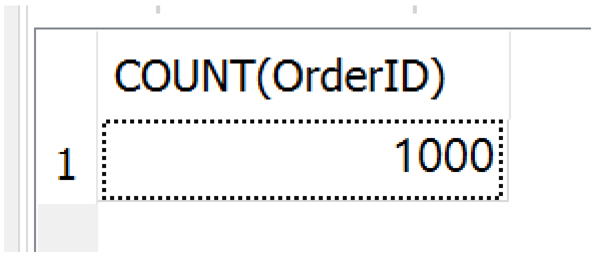
Exercise 1

All transactions:

= 1000 transactions for the year



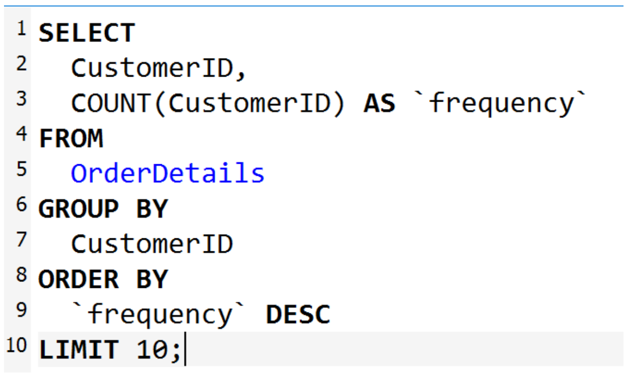
```
SELECT COUNT(OrderID)
From OrderDetails
```



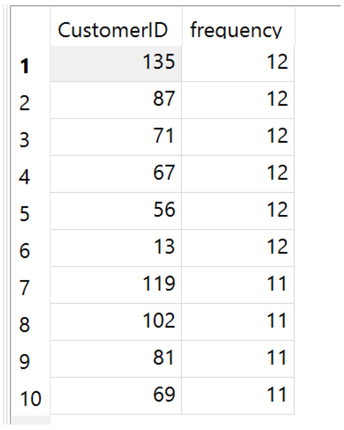
	COUNT(OrderID)
1	1000

Customer with highest number of purchases:

= 6 CustomerID's with 12 purchases each.



```
SELECT
  CustomerID,
  COUNT(CustomerID) AS `frequency`
FROM
  OrderDetails
GROUP BY
  CustomerID
ORDER BY
  `frequency` DESC
LIMIT 10;
```

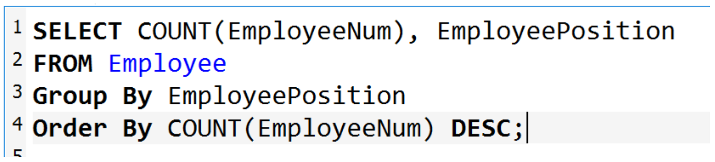


	CustomerID	frequency
1	135	12
2	87	12
3	71	12
4	67	12
5	56	12
6	13	12
7	119	11
8	102	11
9	81	11
10	69	11

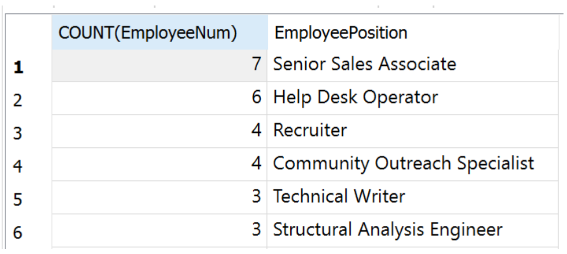
Exercise 2

Includes “Order By” and “Grouped By”

Counting most staffed position in the company, ordered from most staffed position to least staffed position:



```sql
SELECT COUNT(EmployeeNum), EmployeePosition
FROM Employee
Group By EmployeePosition
Order By COUNT(EmployeeNum) DESC;
```

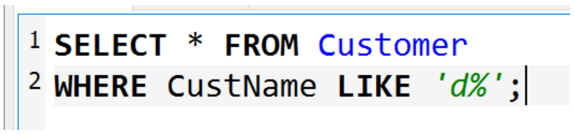


	COUNT(EmployeeNum)	EmployeePosition
1	7	Senior Sales Associate
2	6	Help Desk Operator
3	4	Recruiter
4	4	Community Outreach Specialist
5	3	Technical Writer
6	3	Structural Analysis Engineer

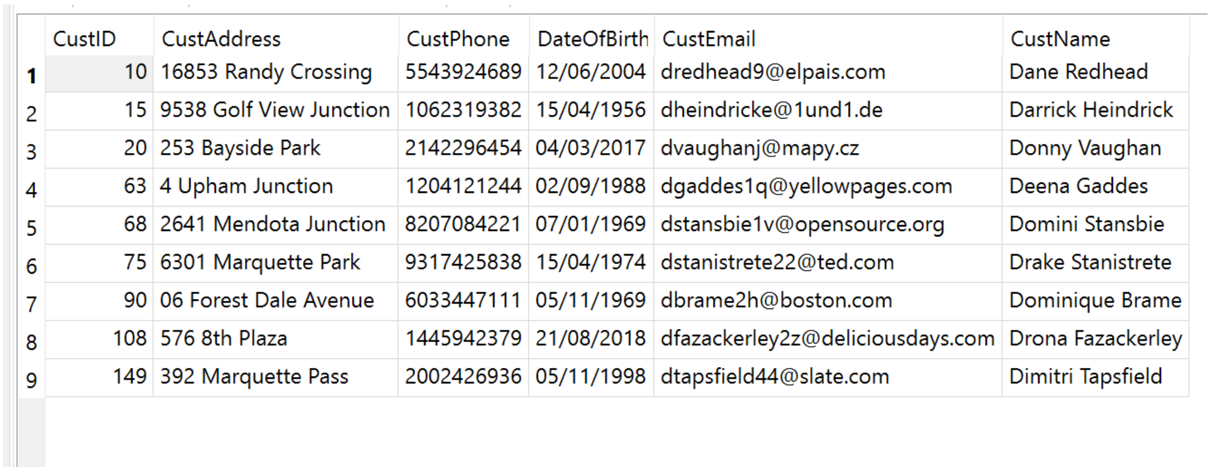
Exercise 3

Pattern matching

Query to find all Customers in Customer table that start with the letter “d”:



```sql
SELECT * FROM Customer
WHERE CustName LIKE 'd%';
```

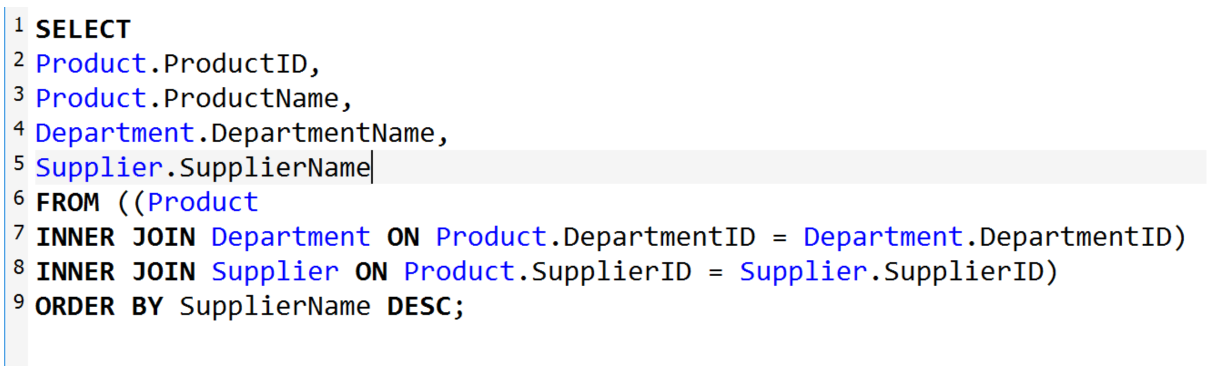


	CustID	CustAddress	CustPhone	DateOfBirth	CustEmail	CustName
1	10	16853 Randy Crossing	5543924689	12/06/2004	dredhead9@elpais.com	Dane Redhead
2	15	9538 Golf View Junction	1062319382	15/04/1956	dheindricke@1und1.de	Darrick Heindrick
3	20	253 Bayside Park	2142296454	04/03/2017	dvaughanj@mapy.cz	Donny Vaughan
4	63	4 Upham Junction	1204121244	02/09/1988	dgaddes1q@yellowpages.com	Deena Gaddes
5	68	2641 Mendota Junction	8207084221	07/01/1969	dstansbie1v@opensource.org	Domini Stansbie
6	75	6301 Marquette Park	9317425838	15/04/1974	dstanistrete22@ted.com	Drake Stanistrete
7	90	06 Forest Dale Avenue	6033447111	05/11/1969	dbrame2h@boston.com	Dominique Brame
8	108	576 8th Plaza	1445942379	21/08/2018	dfazackerley2z@deliciousdays.com	Drona Fazackerley
9	149	392 Marquette Pass	2002426936	05/11/1998	dtapsfield44@slate.com	Dimitri Tapsfield

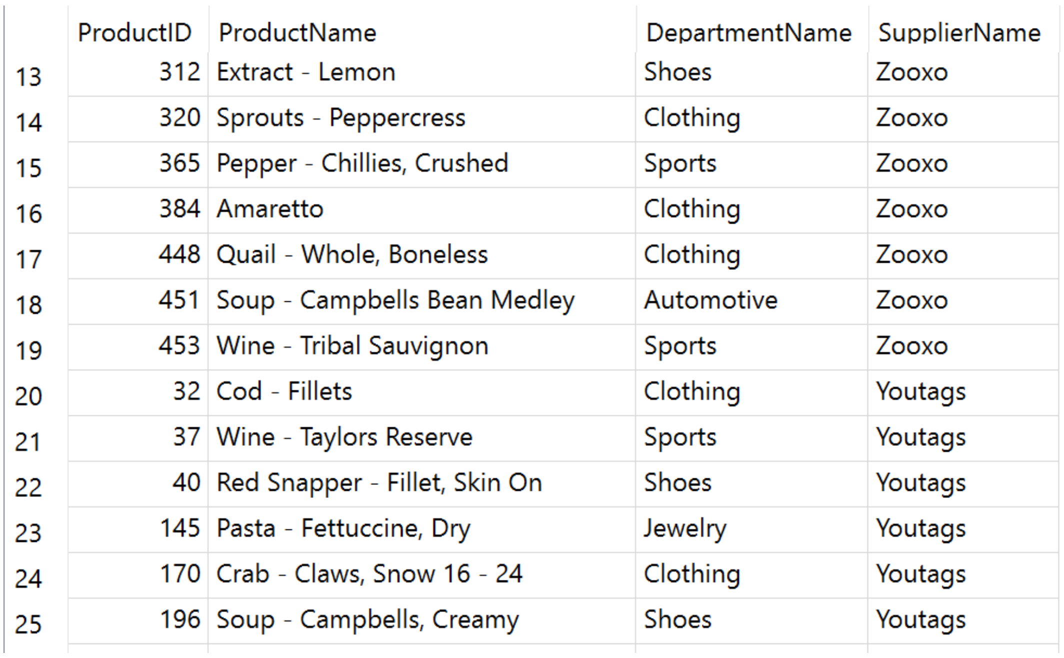
Exercise 4

Show information from three tables

Showing SupplierName from Supplier table, DepartmentName from Department table, and ProductID and ProductName from the Product table. (Ordered by supplier). This shows the products offered by which suppliers and in which department they are found:



```sql
SELECT
Product.ProductID,
Product.ProductName,
Department.DepartmentName,
Supplier.SupplierName
FROM ((Product
INNER JOIN Department ON Product.DepartmentID = Department.DepartmentID)
INNER JOIN Supplier ON Product.SupplierID = Supplier.SupplierID)
ORDER BY SupplierName DESC;
```

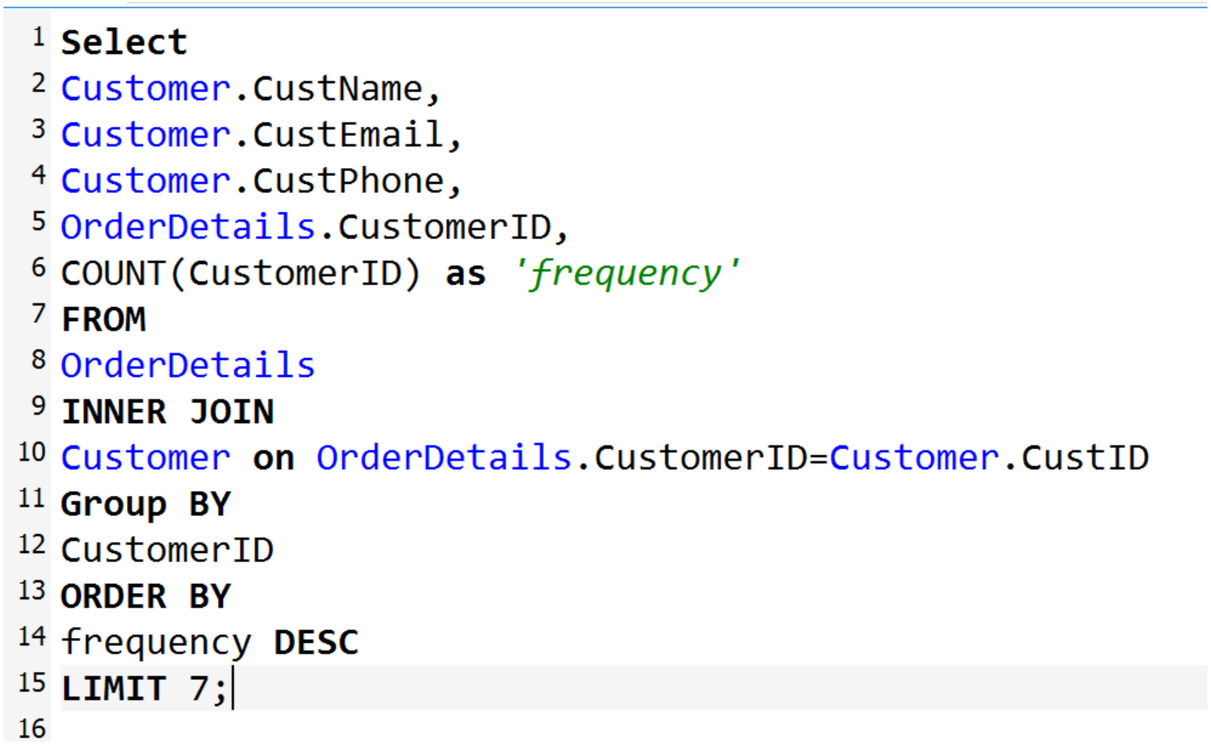


	ProductID	ProductName	DepartmentName	SupplierName
13	312	Extract - Lemon	Shoes	Zooxo
14	320	Sprouts - Peppercress	Clothing	Zooxo
15	365	Pepper - Chillies, Crushed	Sports	Zooxo
16	384	Amaretto	Clothing	Zooxo
17	448	Quail - Whole, Boneless	Clothing	Zooxo
18	451	Soup - Campbells Bean Medley	Automotive	Zooxo
19	453	Wine - Tribal Sauvignon	Sports	Zooxo
20	32	Cod - Fillets	Clothing	Youtags
21	37	Wine - Taylors Reserve	Sports	Youtags
22	40	Red Snapper - Fillet, Skin On	Shoes	Youtags
23	145	Pasta - Fettuccine, Dry	Jewelry	Youtags
24	170	Crab - Claws, Snow 16 - 24	Clothing	Youtags
25	196	Soup - Campbells, Creamy	Shoes	Youtags

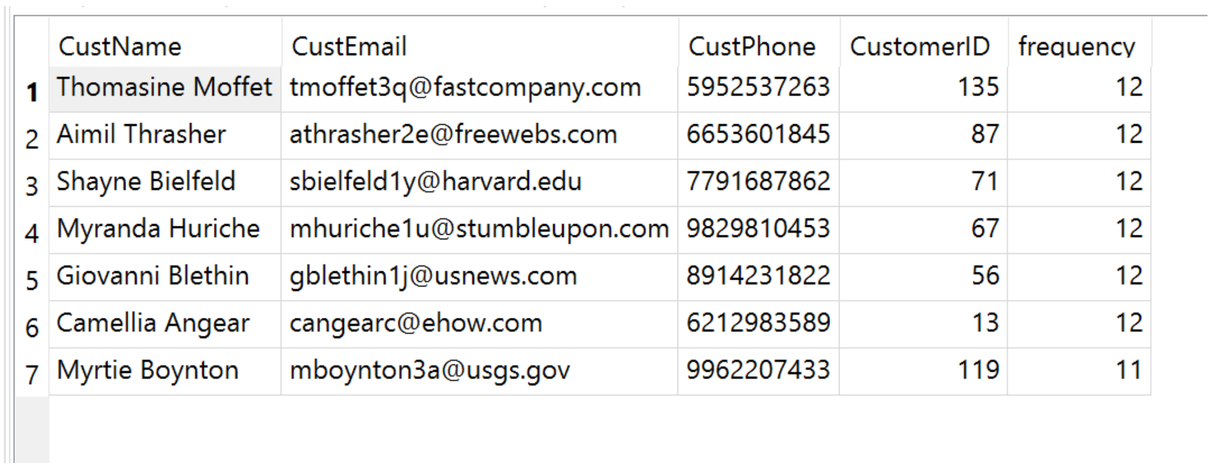
Exercise 5

Information from the most frequent transactions (Customer names)

View contains the names, emails, ID's and phone numbers of the top 7 customers with the most transactions over the year:



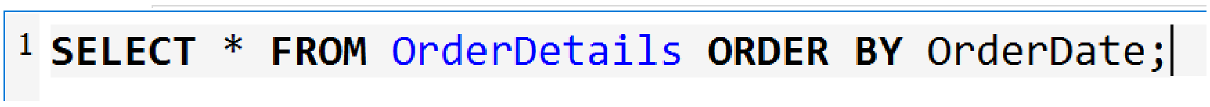
```sql
Select
Customer.CustName,
Customer.CustEmail,
Customer.CustPhone,
OrderDetails.CustomerID,
COUNT(CustomerID) as 'frequency'
FROM
OrderDetails
INNER JOIN
Customer on OrderDetails.CustomerID=Customer.CustID
Group BY
CustomerID
ORDER BY
frequency DESC
LIMIT 7;
```



	CustName	CustEmail	CustPhone	CustomerID	frequency
1	Thomasine Moffet	tmoffet3q@fastcompany.com	5952537263	135	12
2	Aimil Thrasher	athrasher2e@freewebs.com	6653601845	87	12
3	Shayne Bielfeld	sbielfeld1y@harvard.edu	7791687862	71	12
4	Myranda Huriche	mhuriche1u@stumbleupon.com	9829810453	67	12
5	Giovanni Blethin	gblethin1j@usnews.com	8914231822	56	12
6	Camellia Angear	cangearc@ehow.com	6212983589	13	12
7	Myrtie Boynton	mboynton3a@usgs.gov	9962207433	119	11

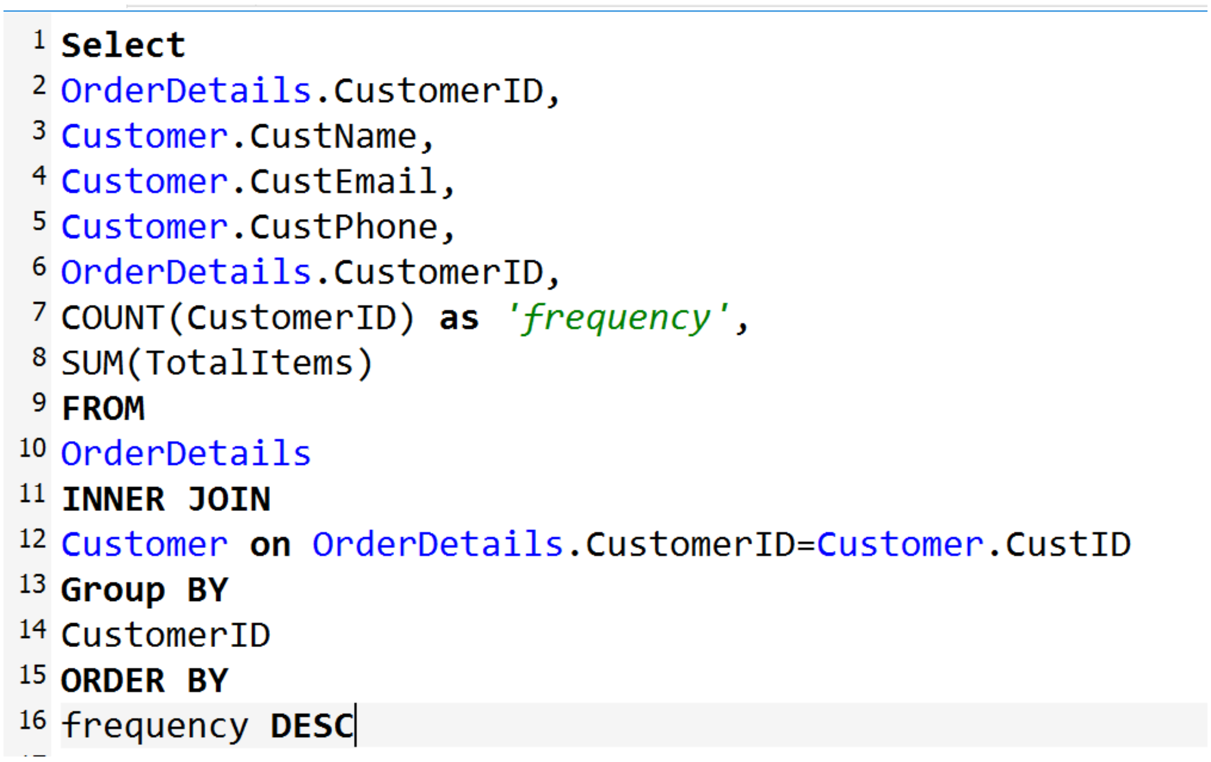
Exercise 6

Transactions sorted from start of the year:



```sql
SELECT * FROM OrderDetails ORDER BY OrderDate;
```

Customers sorted by most frequent shoppers, diplaying customer info and TotalItems purchased for the year:



```sql
Select
OrderDetails.CustomerID,
Customer.CustName,
Customer.CustEmail,
Customer.CustPhone,
OrderDetails.CustomerID,
COUNT(CustomerID) as 'frequency',
SUM(TotalItems)
FROM
OrderDetails
INNER JOIN
Customer on OrderDetails.CustomerID=Customer.CustID
Group BY
CustomerID
ORDER BY
frequency DESC
```