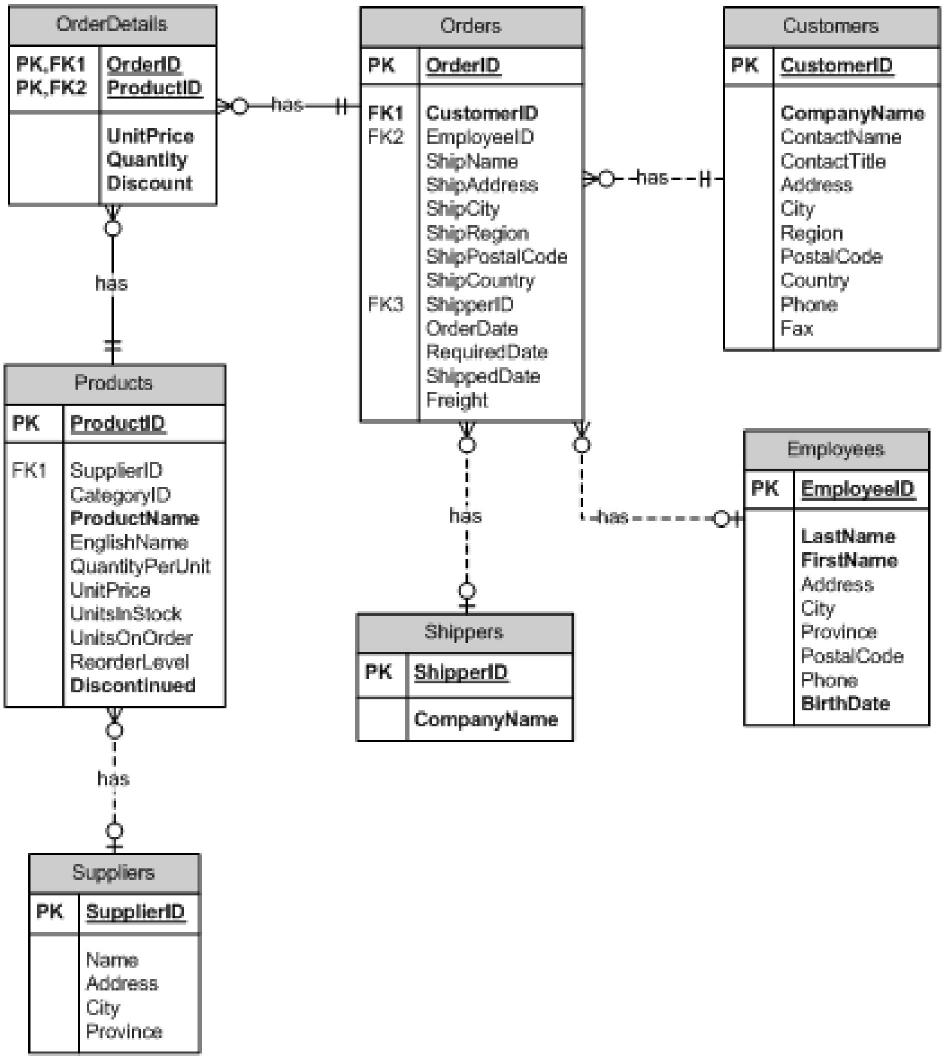
COMP 1630
Relational Database Design and SQL

Term Project

Using Microsoft SQL Server, write the SQL statements for each question necessary to generate the required result set. Clearly identify your answers to the questions. Save your work in the Desire2Learn **Drop box** for **SQL Project**.



Part A - Database and Tables

1. Run the script SQLProjectData.sql to create the Project database, create the tables listed below, and to load the data into the tables.

Customers	91 rows
Employees	9 rows
Shippers	3 rows
Suppliers	15 rows
Products	77 rows
Orders	1078 rows
OrderDetails	2820 rows

There is nothing to hand in for this section!

Part B - SQL Statements

1. List the order detail rows where the quantity is greater than or equal to80 and less than or equal to 90. Display the order id and quantity from the OrderDetails table, the product id from the Products table, and the supplier id from the Suppliers table. Order the result set by the order id. The query should produce the result set listed below.

```
OrderID      Quantity  ProductID  SupplierID
-----------  -----------  -----------  -----------
10027        80        65         2
10052        90        60         8
10052        90        75         12
...
10991        90        76         12
11008        90        34         15
11031        80        13        6

(27 row(s) affected)
```

```
select OrderID, Quantity, p.ProductID, SupplierID
from OrderDetails o JOIN Products p
  on o.ProductID = p.ProductID
where Quantity >= 80 and Quantity <= 90
order by OrderID
```

2. List the product id, product name, and unit price from the Products table where the unit price is less than $10.00. Order the result set by the product name. The query should produce the result set listed below.

```
ProductID    ProductName                        UnitPrice
------------  ----------------------------------  ---------------------
52           Filo Mix                           7.00
33           Geitost                            2.50
24           Guaraná Fantástica                 4.50
…..
54           Tourtière                          7.45
23           Tunnbröd                           9.00
47           Zaanse koeken                      9.50

(11 row(s) affected)
```

```
select ProductID, ProductName, UnitPrice
from Products
where UnitPrice < 10.00
order by ProductName
```

3. List the customer id, company name, contact name, country, and phone from the Customers table where the country is equal to France or Spain. Order the result set by the customer id. The query should produce the result set listed below.

```
CustomerID  CompanyName                       ContactName                Country        Phone
----------  --------------------------------  -------------------------  -------------  --------------------
BLONP       Blondel père et fils              Frédérique Citeaux         France         88.60.15.31
BOLID       Bólido Comidas preparadas         Martín Sommer              Spain          (91) 555 22 82
BONAP       Bon app'                          Laurence Lebihan           France         91.24.45.40
…..
SPECD       Spécialités du monde              Dominique Perrier          France         (1) 47.55.60.10
VICTE       Victuailles en stock              Mary Saveley               France         78.32.54.86
VINET       Vins et alcools Chevalier         Paul Henriot               France         26.47.15.10

(16 row(s) affected)
```

```sql
select CustomerID, CompanyName, ContactName, Country, Phone
from Customers
where Country ='France' or Country = 'Spain'
order by CustomerID
```

4. List the total number of shipping city and their names. Only display the shipping city and the count if there are more than30shipping cities. The query should produce the result set listed below.

```
ShipCity              Count
------------------    -----------
Rio de Janeiro        40
London                47
São Paulo             35
Graz                  33
México D.F.           36
Boise                 42

(6 row(s) affected)
```

```sql
select ShipCity, count(ShipCity) as Count
from Orders
group by ShipCity
 having count(ShipCity) > 30
 order by Count
```

5. List the orders where the shipped date is greater than or equal to July 1, 1992 and less than or equal to June 30, 1993, and calculate the length in years from the shipped date to January 1, 2011. Display the order id, and the shipped date from the Orders table, the company name from the Customers table, and the calculated length in years for each order. Display the shipped date in the format MMM DD YYYY. Order the result set by order id and the calculated years. The query should produce the result set listed below.

```
OrderID      CompanyName                      ShippedDate      ElapsedYear
-----------  -------------------------------  ---------------  ------------
10264        Folk och fä HB                   Jul 17 1992      18
10265        Blondel père et fils             Jul  6 1992      18
10269        White Clover Markets             Jul  3 1992      18
…..
10615        Wilman Kala                      Jun 30 1993      17
10616        Great Lakes Food Market          Jun 29 1993      17
10617        Great Lakes Food Market          Jun 28 1993      17

(346 row(s) affected)
```

```
select OrderID,
       CompanyName,
       format (ShippedDate, 'MM/dd/yyyy') as ShippedDate,
       datediff(year, ShippedDate,'01/01/2011')-1 as ElapsedYear
from Orders o
      JOIN Customers c on c.CustomerID = o.CustomerID
where ShippedDate >= '07/01/1992' and ShippedDate <= '06/30/1993'
order by OrderID, ElapsedYear
```

6. List all the orders where the order date is greater than or equal to January 1, 1992 and less than or equal to March 30, 1992, and the cost of the order is greater than or equal to $1500.00. Display the order id, order date, and a new shipped date calculated by adding 10 days to the shipped date from the Orders table, the product name from the Products table, the company name from the Customer table, and the cost of the order. Format the order date and the shipped date as MON DD YYYY. Use the formula (OrderDetails.Quantity * Products.UnitPrice) to calculate the cost of the order. The query should produce the result set listed below.

```
OrderID    ProductName           CompanyName               OrderDate      NewShippedDate  OrderCost
---------- --------------------- ----------------------    -------------- --------------- -------------
10141      Schoggi Schokolade    Frankenversand            Jan  2 1992    Jan 19 1992     2195.00
10141      Camembert Pierrot     Frankenversand            Jan  2 1992    Jan 19 1992     1700.00
10163      Camembert Pierrot     Frankenversand            Feb  7 1992    Feb 21 1992     1632.00
…..
10195      Côte de Blaye         Que Delícia               Mar 23 1992    Apr  6 1992     5533.50
10196      Côte de Blaye         LILA-Supermercado         Mar 24 1992    May  2 1992     7905.00
10198      Côte de Blaye         Océano Atlántico Ltda.    Mar 26 1992    Apr  9 1992     1581.00
```

```
(11 row(s) affected)
```

```
select o.OrderID,
      ProductName,
      CompanyName,
      format(orderDate,'MMM d yyyy') as OrderDate,
      format(dateadd(day,10,ShippedDate),'MMM d yyyy')
      as NewShippedDate,
      (od.Quantity*p.UnitPrice) as OrderCost

from  Orders o
      JOIN Customers c on o.CustomerID = c.CustomerID
      JOIN OrderDetails od on o.OrderID = od.OrderID
      JOIN Products p on p.ProductID = od.ProductID

where  orderDate >= '01/01/1992'
       and orderDate <= '03/30/1992'
       and (od.Quantity*p.UnitPrice) >= 1500.00
```

7. List all the orders with a shipping city of San Francisco and an order quantity greater than 20. Display the order id from the Orders table, and the unit price and quantity from the OrderDetails table. Order the result set by the order id. The query should produce the result set listed below.

```
OrderID      UnitPrice        Quantity
-----------  ---------------  -----------
10132        22.00            30
10579        7.75             21
10719        7.45             40
10884        10.00            40
10884        38.00            21

(5 row(s) affected)
```

```
select o.OrderID, UnitPrice, Quantity
from Orders o JOIN OrderDetails od
  on o.OrderID = od.OrderID
where ShipCity = 'San Francisco' and Quantity > 20
Order by OrderID
```

8. List the products which contain chai or coffee in their name. Display the product id, product name, quantity per unit and unit price from the Products table. Order the result set by unit price in descending order. The query should produce the result set listed below.

```
ProductID    ProductName                         QuantityPerUnit             UnitPrice
------------ ----------------------------------- --------------------------- ---------------------
43           Ipoh Coffee                         16 - 500 g tins             46.00
1            Chai                                10 boxes x 20 bags          18.00

(2 row(s) affected)
```

```
select ProductID,
       ProductName,
       QuantityPerUnit,
       UnitPrice

from   Products

where  ProductName like '%Coffee%'
OR     ProductName like '%Chai%'

order by UnitPrice DESC
```

Part C - INSERT, UPDATE, DELETE and VIEWS Statements

1. Create a view called vw_supplier_items listing the suppliers and the items they have shipped. Display the supplier id and name from the Suppliers table, and the product id and product name from the Products table. Use the following query to test your view to produce the result set listed below.

```
SELECT *
FROM vw_supplier_items
ORDER BY Name, ProductID
```

```
SupplierID    Name                            ProductID    ProductName
------------- ------------------------------- ------------ ----------------------------------------
5             Supplier                        11           Queso Cabrales
5              Supplier                       12           Queso Manchego La Pastora
1             Supplier A                      1            Chai
...
15            Supplier O                      56           Gnocchi di nonna Alice
15            Supplier O                      69           Gudbrandsdalsost
15            Supplier O                      71           Fløtemysost

(77 row(s) affected)
```

```
create view vw_supplier_items
AS
SELECT s.SupplierID, s.Name, p.ProductID, p.ProductName
FROM Suppliers s JOIN Products p
ON s.SupplierID = p.SupplierID
```

2. Using the UPDATE statement, modify the name of the supplier id 5 to ‘Supplier E’.

```
update Suppliers
set Name = 'Supplier E'
where SupplierID = 5
```

3. Using the INSERT statement, add two rows to the Suppliers table. The first row will have a supplier id of 16, and a supplier name of ‘Supplier P’, and the second row will have a supplier id of 17, and a supplier name of ‘Supplier Q’.

```
INSERT INTO Suppliers(SupplierID, Name)
VALUES ('16', 'Supplier P'),
       ('17', 'Supplier Q');
```

4. Create a view called vw_employee_info to list the employees in the Employee table. Display the employee id, last name, first name, and phone number. Format the name as last name followed by a comma and a space followed by the first name. Display the phone number as opening bracket followed by the first 3 digits of the phone number followed by the closing bracket followed by the next 3 digits of the phone number followed by a dash followed by the last 4 digits of the phone number. Use the following query to test your view to produce the result set listed below.

```
SELECT *
FROM vw_employee_info
WHERE EmployeeID IN ( 1, 3, 5)
```

```
EmployeeID  Name                              PhoneNumber
----------- --------------------------------- -----------------
1           Davolio, Nancy                    (604)555-9857
3           Leverling, Janet                  (604)555-3412
5           Buchanan, Steven                  (604)555-4848

(3 row(s) affected)
```

```
create view vw_employee_info
as
select EmployeeID,
       LastName + ',' + FirstName as Name,
       '(' + substring(Phone, 1, 3) + ')' + substring (Phone, 4, 3)
       + '-' + substring (Phone, 7, 4) as PhoneNumber
from Employees
```

5. Using the UPDATE statement, change the fax value to 000-000-0000 for all rows in the Customers table where the current fax value is NULL. There will be 22 rows updated.

```
update Customers
SET Fax = '000-000-0000'
where Fax IS NULL
```

6. Create a view called vw_order_cost to list the cost of orders. Display the order id and order date from the Orders table, the product id from the Products table, the company name from the Customers table, and the order cost. To calculate the cost of the orders, use the formula (OrderDetails.Quantity * OrderDetails.UnitPrice). Use the following query to test your view to produce the result set listed below.

```
SELECT *
FROM vw_order_cost
```

```
WHERE orderID BETWEEN 10100 AND 10200
ORDER BY ProductID
```

```
OrderID     OrderDate                ProductID     CompanyName                          OrderCost
----------- ------------------------ ------------- ------------------------------------ -------------
10101       1991-10-28 00:00:00      1             Antonio Moreno Taquería               96.00
10156       1992-01-28 00:00:00      1             Richter Supermarkt                    300.00
10159       1992-01-31 00:00:00      1             Maison Dewey                          480.00
...
10170       1992-02-20 00:00:00      77            Reggiani Caseifici                    216.00
10115       1991-11-20 00:00:00      77            Océano Atlántico Ltda.                45.00
10117       1991-11-22 00:00:00      77            Tradição Hipermercados                254.80

(257 row(s) affected)
```

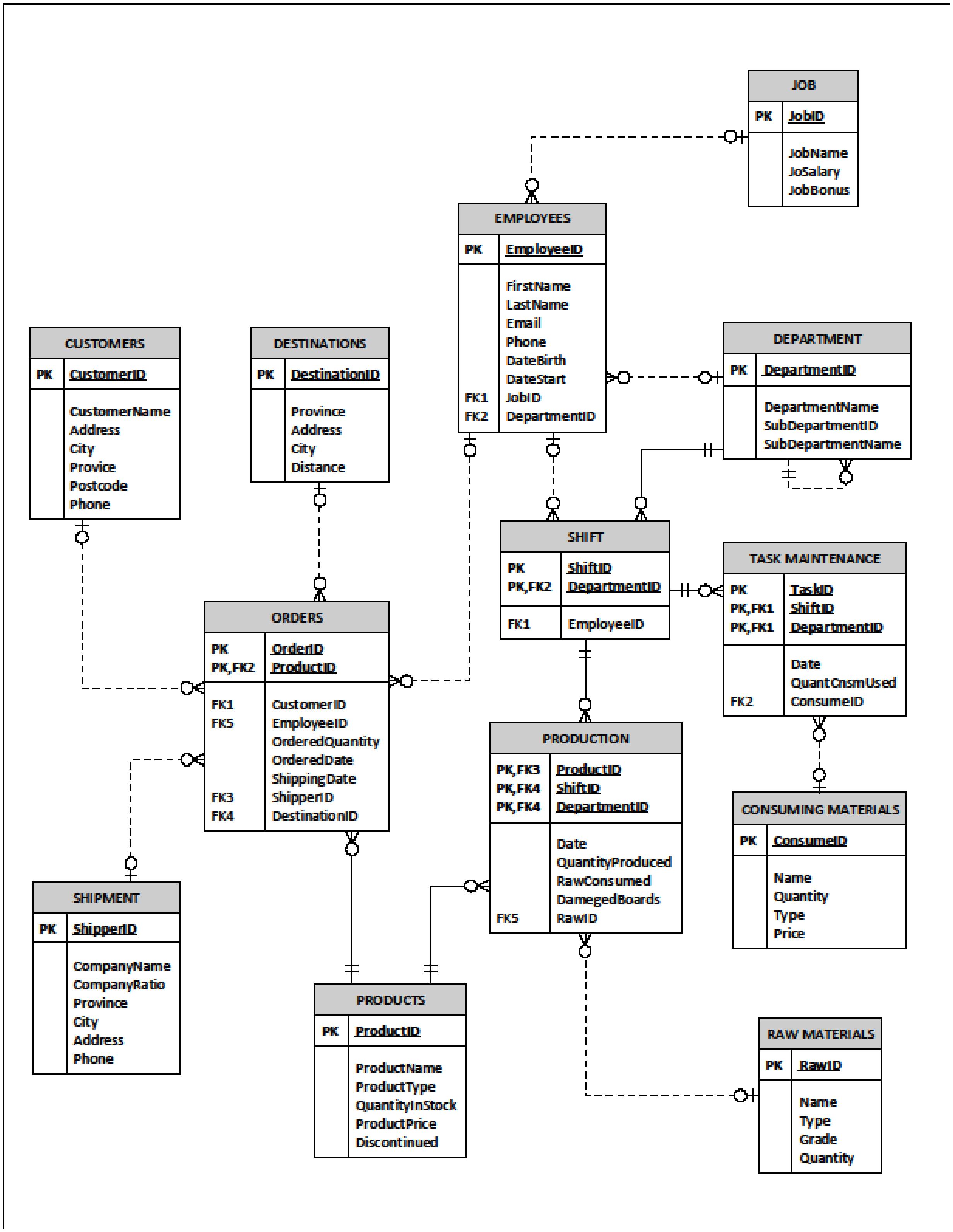
```
create view vw_order_cost
as
Select o.OrderID,
            o.OrderDate,
            p.ProductID,
            c.CompanyName,
            (od.Quantity * od.UnitPrice) as OrderCost
from  Orders o
        JOIN Customers c on o.CustomerID = c.CustomerID
            JOIN OrderDetails od on o.OrderID = od.OrderID
            JOIN Products p on p.ProductID = od.ProductID
```

7. Using the UPDATE statement, increate the unit price in the Products table by 10% for rows with a current unit price less than $5.00. There will be 2 rows updated.

```
update Products
set UnitPrice = UnitPrice*1.1
where UnitPrice < 5.00
```

8. Using the DELETE statement, delete the row with the supplier id of 16, and a supplier name of ‘Supplier P’ from the Suppliers table.

```
delete
from Suppliers
WHERE
supplierID  = 16 and Name = 'Supplier P'
```

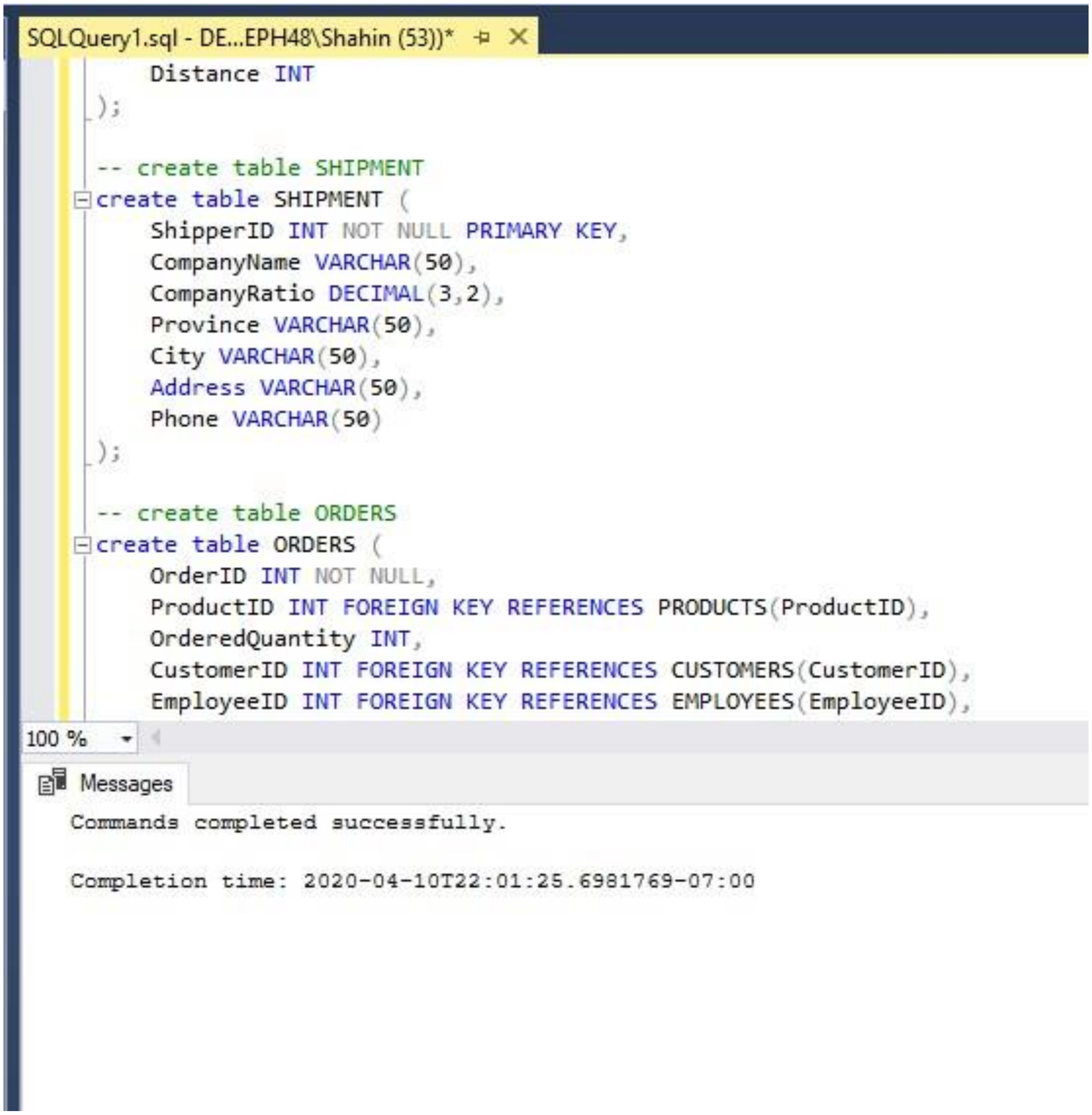

JOB
PK JobID
JobName
JoSalary
JobBonus
EMPLOYEES
PK EmployeeID
FirstName
LastName
Email
Phone
DateBirth
DateStart
FK1 JobID
FK2 DepartmentID
DEPARTMENT
PK DepartmentID
DepartmentName
SubDepartmentID
SubDepartmentName
CUSTOMERS
PK CustomerID
CustomerName
Address
City
Provice
Postcode
Phone
DESTINATIONS
PK DestinationID
Province
Address
City
Distance
SHIFT
PK ShiftID
PK,FK2 DepartmentID
FK1 EmployeeID
TASK MAINTENANCE
PK TaskID
PK,FK1 ShiftID
PK,FK1 DepartmentID
Date
QuantCnsmUsed
FK2 ConsumeID
ORDERS
PK OrderID
PK,FK2 ProductID
FK1 CustomerID
FK5 EmployeeID
OrderedQuantity
OrderedDate
ShippingDate
FK3 ShipperID
FK4 DestinationID
PRODUCTION
PK,FK3 ProductID
PK,FK4 ShiftID
PK,FK4 DepartmentID
Date
QuantityProduced
RawConsumed
DamegedBoards
FK5 RawID
CONSUMING MATERIALS
PK ConsumeID
Name
Quantity
Type
Price
SHIPMENT
PK ShipperID
CompanyName
CompanyRatio
Province
City
Address
Phone
PRODUCTS
PK ProductID
ProductName
ProductType
QuantityInStock
ProductPrice
Discontinued
RAW MATERIALS
PK RawID
Name
Type
Grade
Quantity

Part D – ERD Project SQL Queries

Refer back to your ERD project, fix up any problems with your ERD. Embed your ERD in this project submission and complete the following. You cannot use the same SQL query for multiple questions.

1. Create 5 (or more) tables that correspond to the entities in your ERD project. Choose tables that are in relationship with each other so you can use the tables for the following queries. Show the SQL statements. (Do not forget the primary and foreign keys, and any appropriate constraints.)

A sql file containing the statements has been attached to the project file.

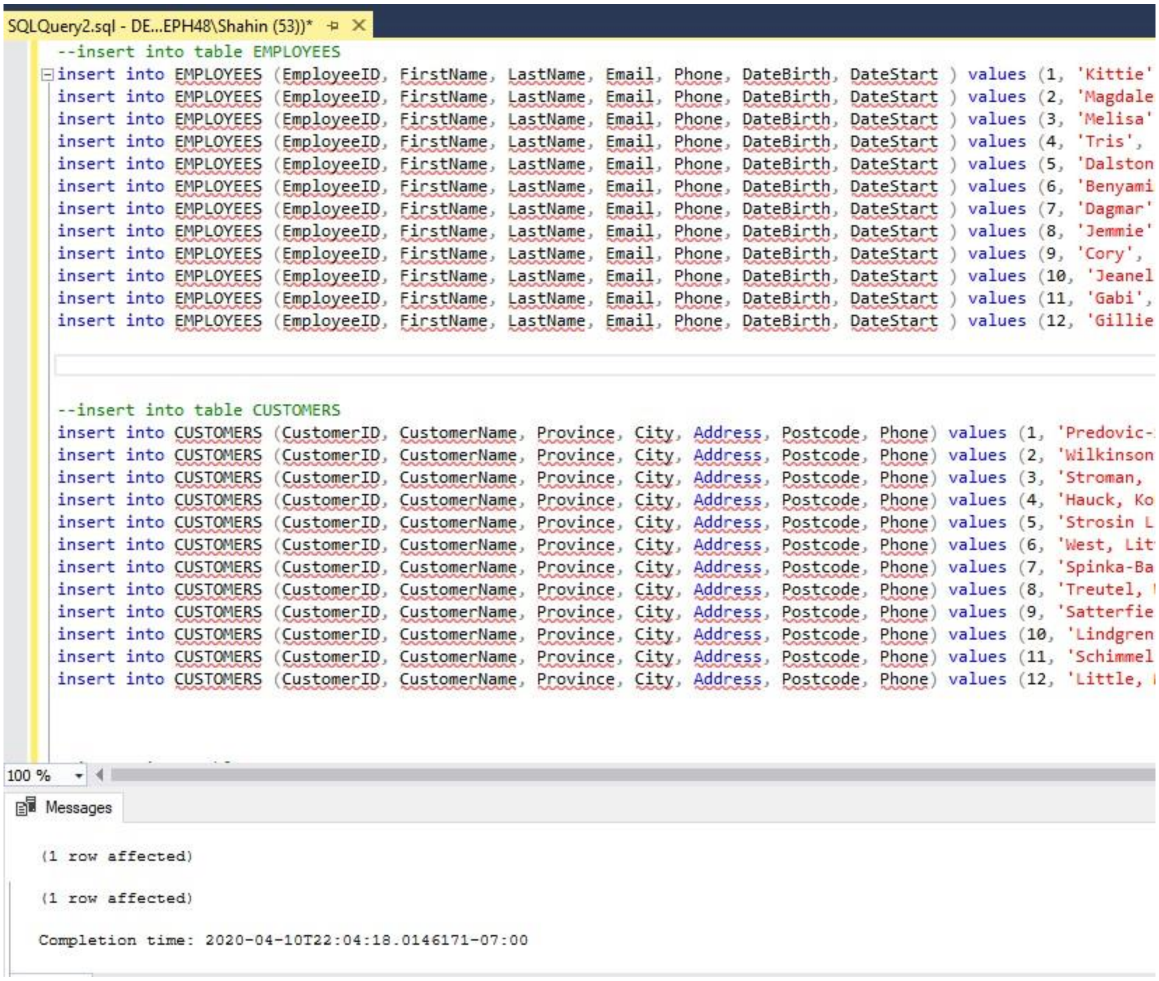


```
    Distance INT
);

-- create table SHIPMENT
create table SHIPMENT (
    ShipperID INT NOT NULL PRIMARY KEY,
    CompanyName VARCHAR(50),
    CompanyRatio DECIMAL(3,2),
    Province VARCHAR(50),
    City VARCHAR(50),
    Address VARCHAR(50),
    Phone VARCHAR(50)
);

-- create table ORDERS
create table ORDERS (
    OrderID INT NOT NULL,
    ProductID INT FOREIGN KEY REFERENCES PRODUCTS(ProductID),
    OrderedQuantity INT,
    CustomerID INT FOREIGN KEY REFERENCES CUSTOMERS(CustomerID),
    EmployeeID INT FOREIGN KEY REFERENCES EMPLOYEES(EmployeeID),
```

```
Commands completed successfully.

Completion time: 2020-04-10T22:01:25.6981769-07:00
```

2. Insert some data into each of the tables. Show the SQL statements, and the content of each table after all the tables are being populated.

A sql file containing the statements has been attached to the project file.

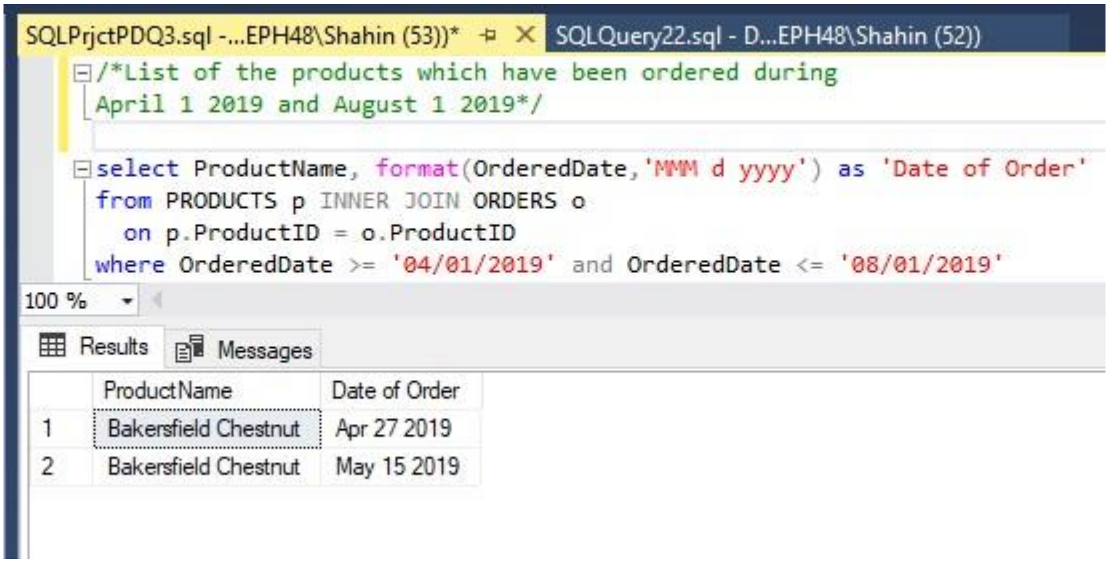


```
--insert into table EMPLOYEES
insert into EMPLOYEES (EmployeeID, FirstName, LastName, Email, Phone, DateBirth, DateStart ) values (1, 'Kittie'
insert into EMPLOYEES (EmployeeID, FirstName, LastName, Email, Phone, DateBirth, DateStart ) values (2, 'Magdale
insert into EMPLOYEES (EmployeeID, FirstName, LastName, Email, Phone, DateBirth, DateStart ) values (3, 'Melisa'
insert into EMPLOYEES (EmployeeID, FirstName, LastName, Email, Phone, DateBirth, DateStart ) values (4, 'Tris',
insert into EMPLOYEES (EmployeeID, FirstName, LastName, Email, Phone, DateBirth, DateStart ) values (5, 'Dalston
insert into EMPLOYEES (EmployeeID, FirstName, LastName, Email, Phone, DateBirth, DateStart ) values (6, 'Benyami
insert into EMPLOYEES (EmployeeID, FirstName, LastName, Email, Phone, DateBirth, DateStart ) values (7, 'Dagmar'
insert into EMPLOYEES (EmployeeID, FirstName, LastName, Email, Phone, DateBirth, DateStart ) values (8, 'Jemmie'
insert into EMPLOYEES (EmployeeID, FirstName, LastName, Email, Phone, DateBirth, DateStart ) values (9, 'Cory',
insert into EMPLOYEES (EmployeeID, FirstName, LastName, Email, Phone, DateBirth, DateStart ) values (10, 'Jeanel
insert into EMPLOYEES (EmployeeID, FirstName, LastName, Email, Phone, DateBirth, DateStart ) values (11, 'Gabi',
insert into EMPLOYEES (EmployeeID, FirstName, LastName, Email, Phone, DateBirth, DateStart ) values (12, 'Gillie


--insert into table CUSTOMERS
insert into CUSTOMERS (CustomerID, CustomerName, Province, City, Address, Postcode, Phone) values (1, 'Predovic-
insert into CUSTOMERS (CustomerID, CustomerName, Province, City, Address, Postcode, Phone) values (2, 'Wilkinson
insert into CUSTOMERS (CustomerID, CustomerName, Province, City, Address, Postcode, Phone) values (3, 'Stroman,
insert into CUSTOMERS (CustomerID, CustomerName, Province, City, Address, Postcode, Phone) values (4, 'Hauck, Ko
insert into CUSTOMERS (CustomerID, CustomerName, Province, City, Address, Postcode, Phone) values (5, 'Strosin L
insert into CUSTOMERS (CustomerID, CustomerName, Province, City, Address, Postcode, Phone) values (6, 'West, Lit
insert into CUSTOMERS (CustomerID, CustomerName, Province, City, Address, Postcode, Phone) values (7, 'Spinka-Ba
insert into CUSTOMERS (CustomerID, CustomerName, Province, City, Address, Postcode, Phone) values (8, 'Treutel,
insert into CUSTOMERS (CustomerID, CustomerName, Province, City, Address, Postcode, Phone) values (9, 'Satterfie
insert into CUSTOMERS (CustomerID, CustomerName, Province, City, Address, Postcode, Phone) values (10, 'Lindgren
insert into CUSTOMERS (CustomerID, CustomerName, Province, City, Address, Postcode, Phone) values (11, 'Schimmel
insert into CUSTOMERS (CustomerID, CustomerName, Province, City, Address, Postcode, Phone) values (12, 'Little,
```

```
(1 row affected)

(1 row affected)

Completion time: 2020-04-10T22:04:18.0146171-07:00
```

3. Come up with one query of your database that requires an inner join of at least two tables. State the query, show the SQL statement, and the output.

```
select ProductName, format(OrderedDate,'MMM d yyyy') as 'Date of
Order'
from PRODUCTS p INNER JOIN ORDERS o
  on p.ProductID = o.ProductID
where OrderedDate >= '04/01/2019' and OrderedDate <= '08/01/2019'
```

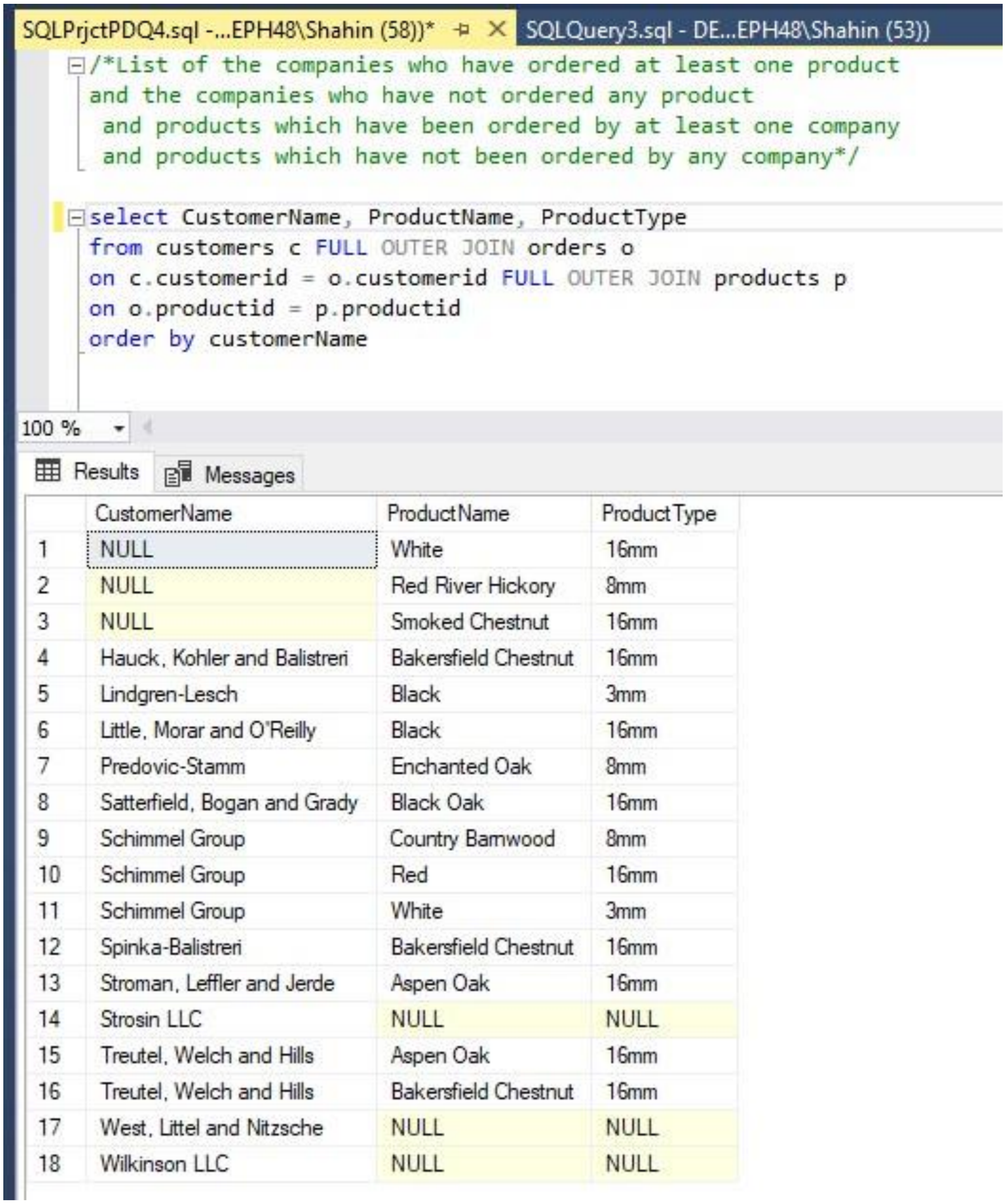

SQLPrjctPDQ3.sql -...EPH48\Shahin (53))*
SQLQuery22.sql - D...EPH48\Shahin (52))
100 %
Results
Messages

```
/*List of the products which have been ordered during
April 1 2019 and August 1 2019*/

select ProductName, format(OrderedDate,'MMM d yyyy') as 'Date of Order'
from PRODUCTS p INNER JOIN ORDERS o
  on p.ProductID = o.ProductID
where OrderedDate >= '04/01/2019' and OrderedDate <= '08/01/2019'
```

	ProductName	Date of Order
1	Bakersfield Chestnut	Apr 27 2019
2	Bakersfield Chestnut	May 15 2019

4. Come up with one query of your database that requires an outer join of at least two tables. State the query, show the SQL statement, and the output.

```
select CustomerName, ProductName, ProductType
from customers c FULL OUTER JOIN orders o
on c.customerid = o.customerid FULL OUTER JOIN products p
on o.productid = p.productid
order by customerName
```



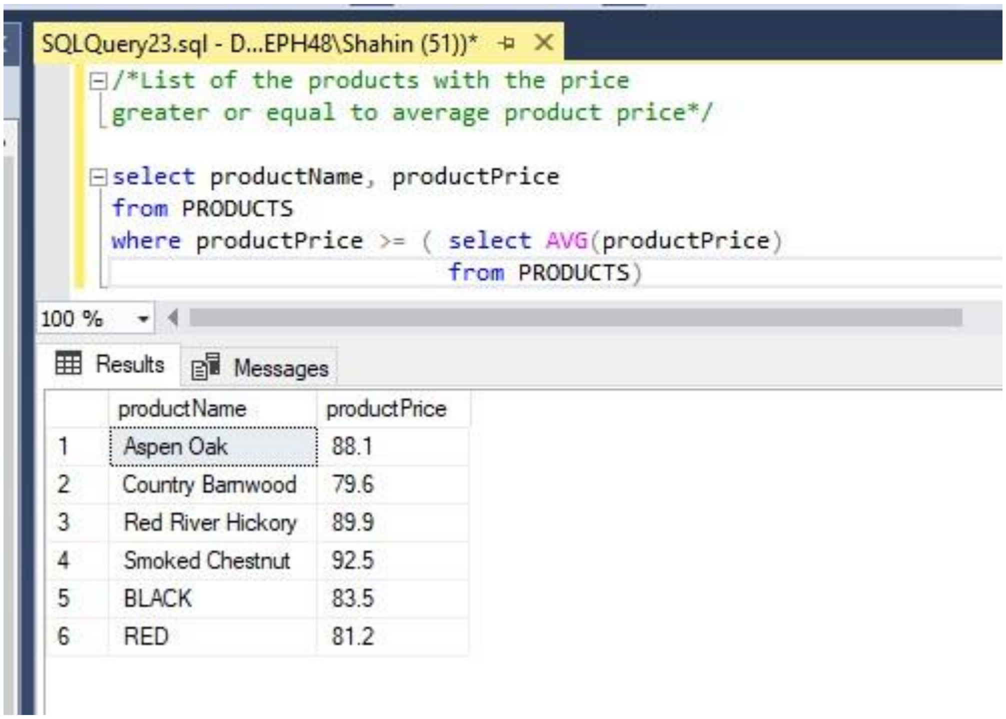
SQLPrjctPDQ4.sql -...EPH48\Shahin (58))*
SQLQuery3.sql - DE...EPH48\Shahin (53))
100 %
Results
Messages

```
/*List of the companies who have ordered at least one product
and the companies who have not ordered any product
 and products which have been ordered by at least one company
 and products which have not been ordered by any company*/

select CustomerName, ProductName, ProductType
from customers c FULL OUTER JOIN orders o
on c.customerid = o.customerid FULL OUTER JOIN products p
on o.productid = p.productid
order by customerName
```

	CustomerName	ProductName	ProductType
1	NULL	White	16mm
2	NULL	Red River Hickory	8mm
3	NULL	Smoked Chestnut	16mm
4	Hauck, Kohler and Balistreri	Bakersfield Chestnut	16mm
5	Lindgren-Lesch	Black	3mm
6	Little, Morar and O'Reilly	Black	16mm
7	Predovic-Stamm	Enchanted Oak	8mm
8	Satterfield, Bogan and Grady	Black Oak	16mm
9	Schimmel Group	Country Barnwood	8mm
10	Schimmel Group	Red	16mm
11	Schimmel Group	White	3mm
12	Spinka-Balistreri	Bakersfield Chestnut	16mm
13	Stroman, Leffler and Jerde	Aspen Oak	16mm
14	Strosin LLC	NULL	NULL
15	Treutel, Welch and Hills	Aspen Oak	16mm
16	Treutel, Welch and Hills	Bakersfield Chestnut	16mm
17	West, Littel and Nitzsche	NULL	NULL
18	Wilkinson LLC	NULL	NULL

5. Come up with one query of your simple database that requires at least one subquery. State the query, show the SQL statement, and the output.

```
select productName, productPrice
from PRODUCTS
where productPrice >= ( select AVG(productPrice)
                        from PRODUCTS)
```

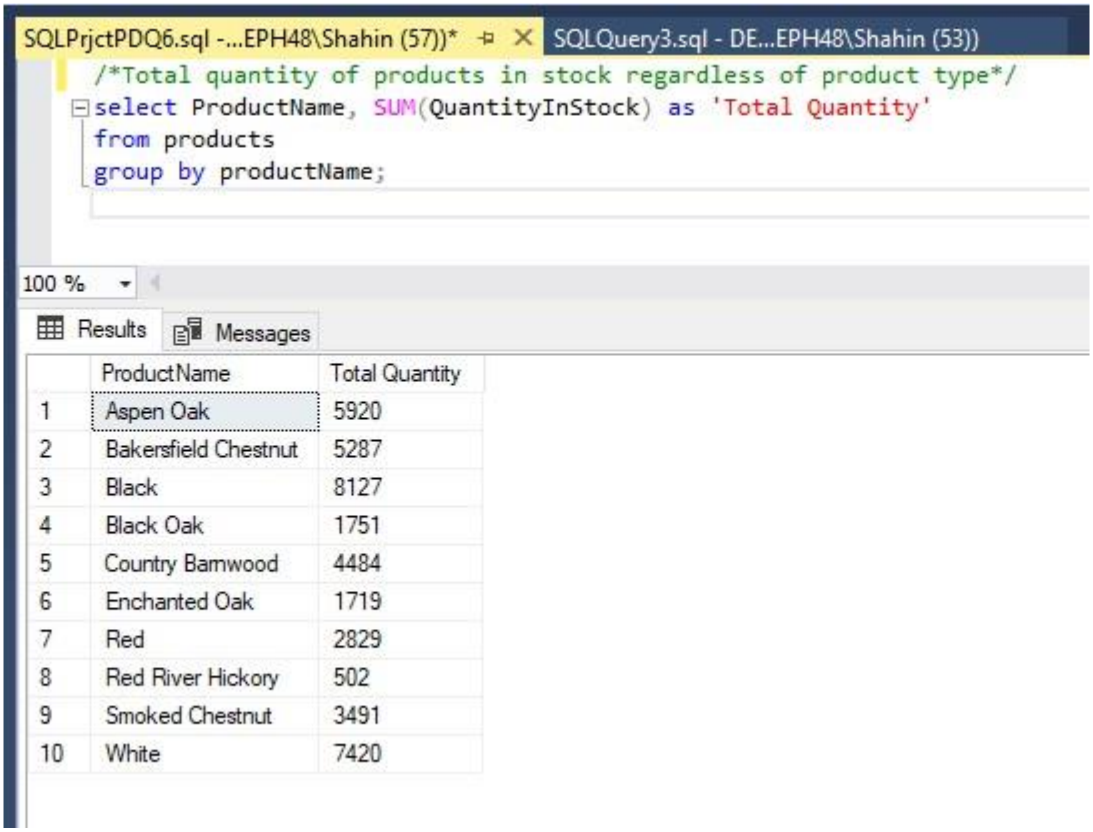


```
/*List of the products with the price
greater or equal to average product price*/

select productName, productPrice
from PRODUCTS
where productPrice >= ( select AVG(productPrice)
                        from PRODUCTS)
```

	productName	productPrice
1	Aspen Oak	88.1
2	Country Barnwood	79.6
3	Red River Hickory	89.9
4	Smoked Chestnut	92.5
5	BLACK	83.5
6	RED	81.2

6. Come up with one query of your simple database that requires an aggregate function. State the query, show the SQL statement, and the output.

```
select ProductName, SUM(QuantityInStock) as 'Total Quantity'
from products
group by productName;
```



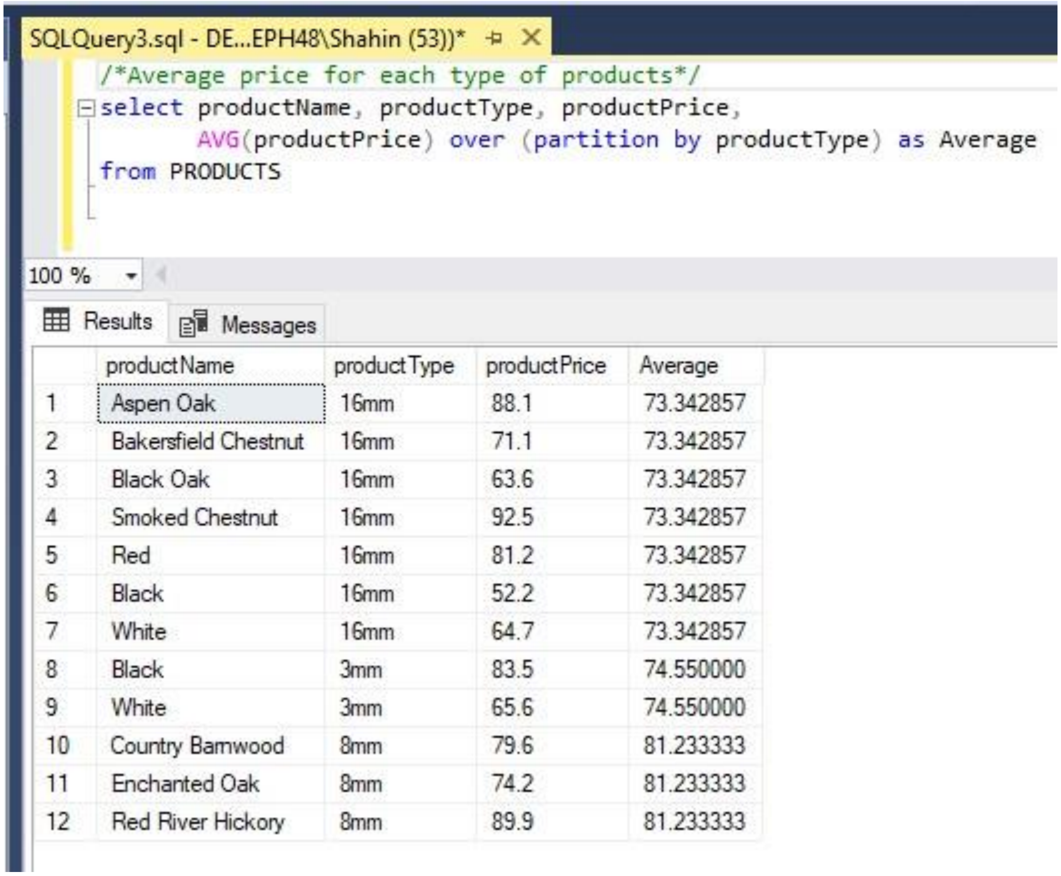
SQLPrjctPDQ6.sql -...EPH48\Shahin (57))*
SQLQuery3.sql - DE...EPH48\Shahin (53))
100 %
Results
Messages

```
/*Total quantity of products in stock regardless of product type*/
select ProductName, SUM(QuantityInStock) as 'Total Quantity'
from products
group by productName;
```

	ProductName	Total Quantity
1	Aspen Oak	5920
2	Bakersfield Chestnut	5287
3	Black	8127
4	Black Oak	1751
5	Country Barnwood	4484
6	Enchanted Oak	1719
7	Red	2829
8	Red River Hickory	502
9	Smoked Chestnut	3491
10	White	7420

7. Come up with one query of your simple database that requires the OVER clause. State the query, show the SQL statement, and the output.

```
select productName, productType, productPrice,
       AVG(productPrice) over (partition by productType) as
       Average
from PRODUCTS
```



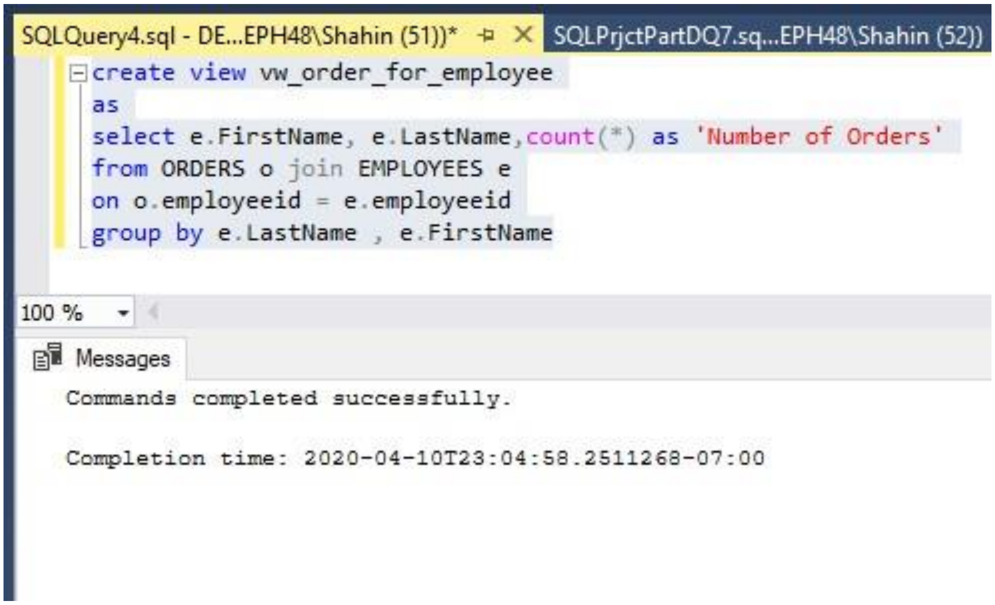
```
/*Average price for each type of products*/
select productName, productType, productPrice,
       AVG(productPrice) over (partition by productType) as Average
from PRODUCTS
```

	productName	productType	productPrice	Average
1	Aspen Oak	16mm	88.1	73.342857
2	Bakersfield Chestnut	16mm	71.1	73.342857
3	Black Oak	16mm	63.6	73.342857
4	Smoked Chestnut	16mm	92.5	73.342857
5	Red	16mm	81.2	73.342857
6	Black	16mm	52.2	73.342857
7	White	16mm	64.7	73.342857
8	Black	3mm	83.5	74.550000
9	White	3mm	65.6	74.550000
10	Country Barnwood	8mm	79.6	81.233333
11	Enchanted Oak	8mm	74.2	81.233333
12	Red River Hickory	8mm	89.9	81.233333

8. Come up with a view of your simple database that limits the user's access to specific rows and columns. State the purpose of the view, show the SQL statement for the view definition, an example of how the view is invoked, and the output.

The purpose for this view is knowing how many orders has been registered by an employee

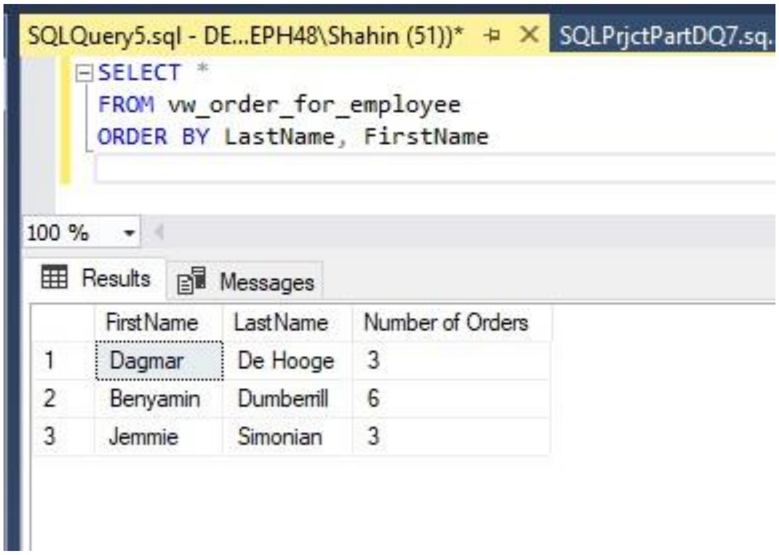
```
create view vw_order_for_employee
as
select e.FirstName, e.LastName,count(*) as 'Number of Orders'
from ORDERS o join EMPLOYEES e
on o.employeeid = e.employeeid
group by e.LastName , e.FirstName
```


SQLQuery4.sql - DE...EPH48\Shahin (51))*
SQLPrjctPartDQ7.sq...EPH48\Shahin (52))
100 %
Messages

```
create view vw_order_for_employee
as
select e.FirstName, e.LastName,count(*) as 'Number of Orders'
from ORDERS o join EMPLOYEES e
on o.employeeid = e.employeeid
group by e.LastName , e.FirstName
```

```
Commands completed successfully.

Completion time: 2020-04-10T23:04:58.2511268-07:00
```

```
SELECT *
FROM vw_order_for_employee
ORDER BY LastName, FirstName
```



```
SELECT *
FROM vw_order_for_employee
ORDER BY LastName, FirstName
```

	FirstName	LastName	Number of Orders
1	Dagmar	De Hooge	3
2	Benyamin	Dumberrill	6
3	Jemmie	Simonian	3