Wolfgang Gatterbauer
70-455
Cheuk Yin Nicholas Cheung, Zuojun Gong, Yutong Li, Peter Weon

Database Design Written Report

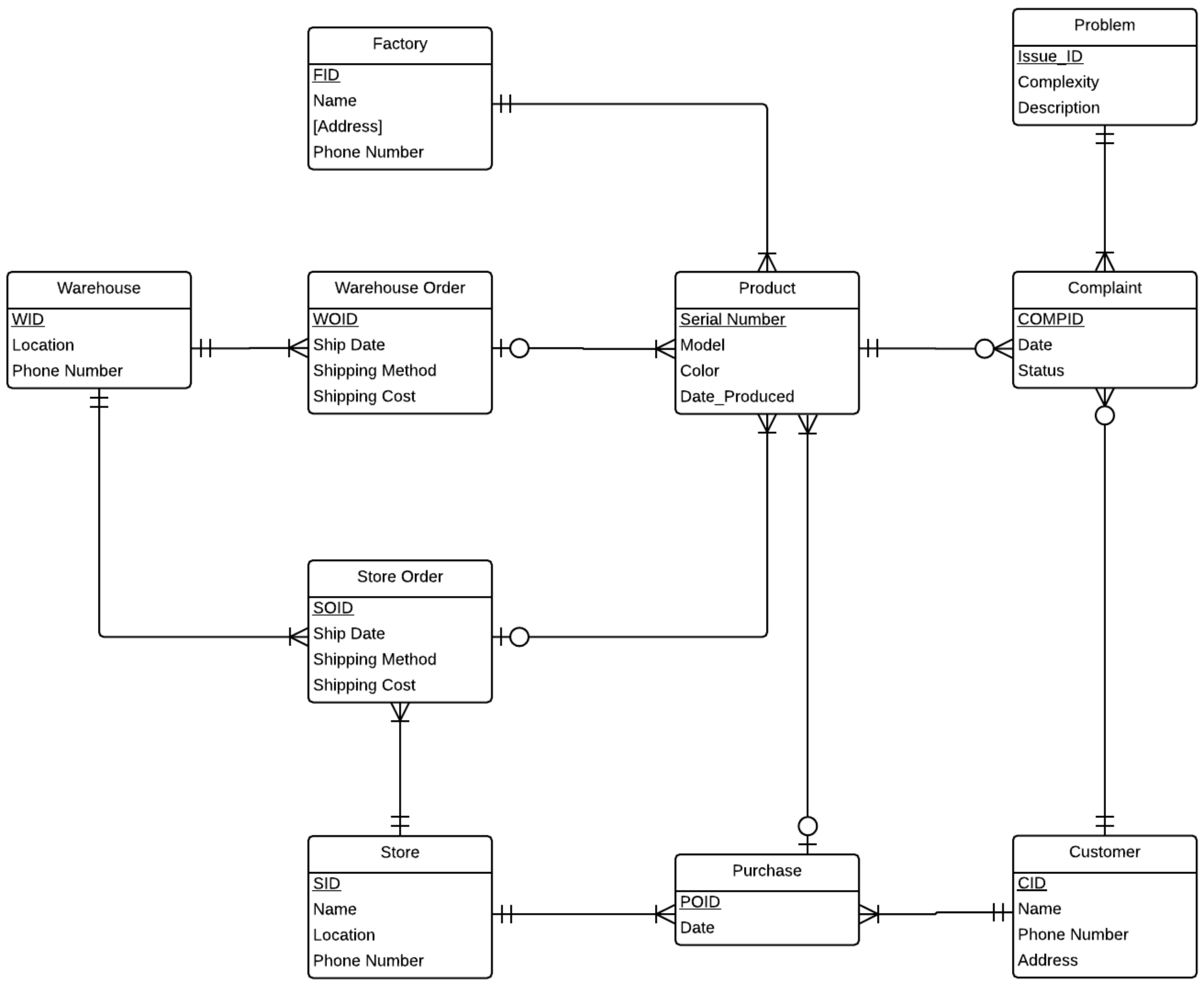
Executive Summary

We constructed a useful database for Apple, a popular consumer product company, to follow the entire lifecycle of a particular product to keep track of important information and identify patterns in customer complaints. There were three aspects that we focused on: Manufacturing, Retail/Distribution, and Customer Service/Quality Control. Apple produces various lines of products such as the iPhone, iPad, iPod, MacBook, etc., and the three aspects that we focused roughly completes the life cycle for an Apple product. Our objective was to pinpoint where such problems occur in the process in order to prevent further costs for the company.

Problem Context

The problem we are addressing is that in a situation of a high customer complaint rate, the inability to identify the root cause will lead to a huge loss in the company's profits. Our database connects departments along a production chain and allow users to keep track of each and every product. For our database, the end users will be Apple employees who are in regular interaction with the company's products, namely quality control specialists, sales managers, customer care representatives, and manufacturers. Users can monitor the information in the respective departments while enabling other users to extract and aggregate information in order to perform high level analysis on the potential factors of product defect rates. The information can benefit departments from Sales Revenue and Quality Control report to manufacturing analysis and supply chain management. Identifying specific manufacturing issues early on can therefore reduce or prevent further costs in the future.

ER Diagram



When designing the ER diagram for our database, we first came up with several business rules for Apple in order to simplify the complex real-life scenario to one that we are able to recreate.

1. *Every complaint filed should only contain a specific problem for one particular product.*
2. *Products will only be purchased once by one customer at one store.*
3. *The company only uses one shipping carrier.*

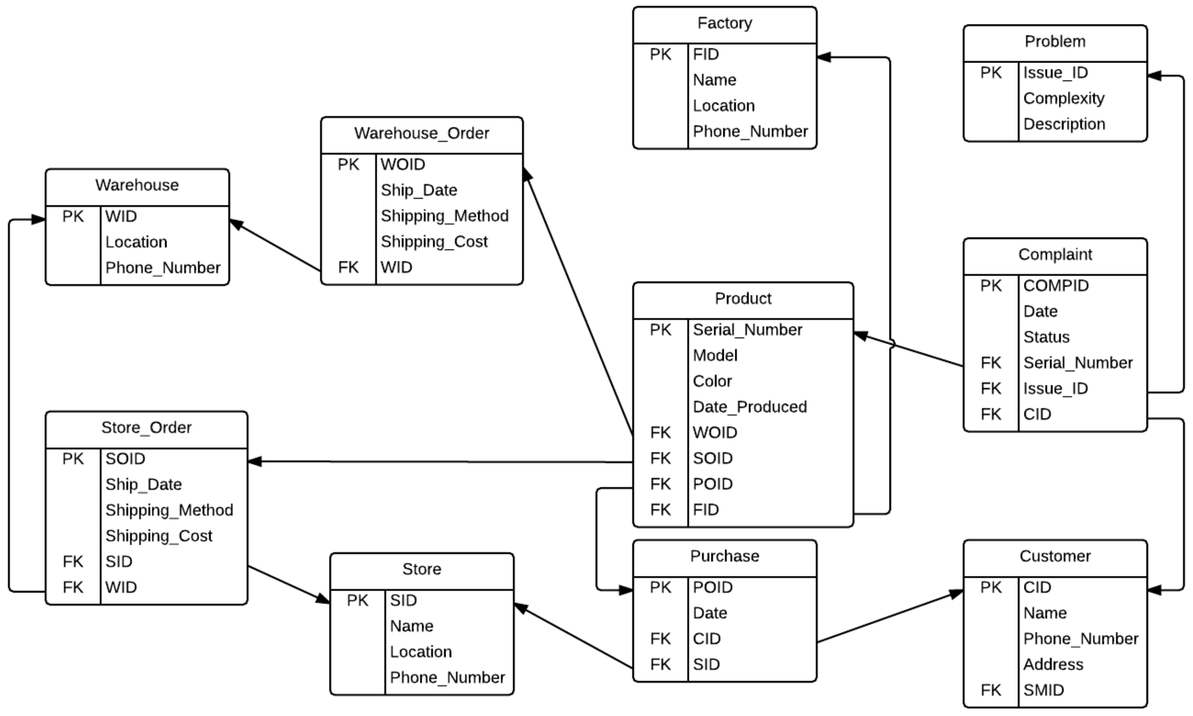
In the ER diagram, we identified 10 entities in our database to complete the entire lifecycle of a product. The life cycle begins when a factory produces a product. To illustrate this, we created entities “Factory” and “Product”. As a factory can produce many products, we created a one-to-many relationship to connect the two. Then, a batch of products is shipped to a warehouse. To show this batch connecting “Product” to “Warehouse”, we created “Warehouse Order”, which tracks its own useful information such as “Ship Date”, “Shipping Method”, “Shipping Cost”. The same applies to the shipping process from the warehouse to the store, but with an entity “Store Order” instead. After the product reaches the store, it is sold to the customer through a purchase. Therefore, we created “Purchase” as an associative entity between “Store” and “Customer”.

It might be quite bizarre to see the entity “Product” being linked with so many different entities, but this is because ever since a product is produced, we would like to track where the product is whenever it is delivered to a warehouse, store or the customers in case a complaint is filed for it. To do so, we made the entity “Product” share one-to-many relationships with “Warehouse Order”, “Store Order” and “Purchase”. As there are some cases where products may not be delivered yet, these relationships are made optional.

Then, we proceed to the most important part of the database: customer complaints. When a customer files a complaint, they file a complaint about a specific problem regarding a specific product. Therefore we created an entity “Complaint”, which is an associative entity for a ternary relationship between “Product”, “Customer” and “Problem”. From the entity “Complaint”, we can track back to the root cause through the entity “Product”, which, as mentioned above, could trace back to all its production stages.

After modeling the life cycle with an ER diagram, it is interesting to see that there are multiple entities in our ER diagram that possess multiple identities. For instance, the entity “Store_Order” serves as an associative entity when it connects “Store” and “Warehouse” together with two one-to-many relationships pointing towards “Store Order”. At the same time, “Store_Order” is also a normal entity if we look at its one-to-many relationship with “Product”, which points at the outwards (one supply schedule has many products). This hybrid type of entity can also be seen in entity “Purchase”.

Database Schema



For our database schema, we converted every entity to a table and every unique identifier within it into a primary key. We also added the attributes linked to the entities originally to their respective tables. In addition to the existing attributes, we inserted foreign keys into the tables that were on the “many” end of one-to-many relationships in the ER diagram. These foreign keys are linked to the primary key of the tables on the “one” end of the one-to-many relationships so that end users will be able to trace information of a specific entry between two tables. For example, the foreign key “SOID” in “Product” links to the primary key “SOID” in “Store_Order” and allows users to identify which store order a specific product belongs to.

Possible Queries

For this section, we will introduce some queries that end users can possibly make use of to facilitate their decision-making process regarding product management. Please refer to the queries listed in the appendix or the .txt file attached.

A. Products bought by a specific customer:
Using query A, customer service managers are able to look up which products a certain customer, Nicholas in this case, bought. This might be useful when a customer service manager needs to refer to a product when he or she receives a customer complaint.

B. Model with the most complaints:
For query B, we used the function "count" to determine the model that has the largest number of complaints. Quality analysts can use this query to find out which model is found as the most defective and potentially requires a recall.

C. Complete information about a specific product:
Query C uses left outer join to show complete information for all products, even those with NULL values for certain foreign keys. This query is very beneficial to managers who work with product distribution as they can obtain the big picture of the life cycle of a product.

D. For every factory, number of defective products produced:
Query D counts the distinct product serial numbers for those that received the complaint in order to determine the number of defective products produced. Quality analysts can therefore assess the performance of every factory that the company has and determine whether defects occur more often when the products is produced by a specific factory.

E. For every factory, number of all products produced
Query E is a basic query that counts all products every factory produced. This is a step necessary to look at the defect rate for every factory, which can show the efficiency of every factory.

F. Defect rate for every factory
Query F is the most complicated yet interesting query. It uses "with" statements to combine the previous two queries described with a new query and calculate the defect rate for every factory. By comparing the defect rates, the manufacturing manager can identify the least or most efficient factory and follow up with further action, such as monitoring the poorly performing factories more closely or increasing production for the most efficient factories.

Insertion and Modification Statements

In addition to issuing queries, end users can use the following insertion and modification statements listed in the appendix or the .txt file attached to update the database if needed.

A. Create new warehouse order
Whenever a warehouse makes a new order, end users can insert a new entry in the "Warehouse_Order" table by using this statement.

B. Update product after shipped to warehouse and store
After a new warehouse order is made, the database user has to update the products ordered by assigning them a foreign key "WOID", which was previously a null value.

C. Cancel store order
Whenever the store made a mistake and wants to cancel an order, this statement will be useful as it changes the "SOID" of products which were initially in the to-be-cancelled store order back to NULL and then removes the unwanted store order.

Discussion
Our group found this project extremely useful and interesting as we were able to put our database modeling techniques to test and apply them to solve a real-life issue that we are interested in. In class, we were only able to apply certain concepts of database modeling to illustrate specific real-life scenarios, but this project really challenged us to tackle a problem of a larger scale and model a complete database after it.

When we first came up with the problem that we would like to address, we thought that our database would consist of straightforward relationships we often encountered in class. However, when we started drawing the ER diagram, we learnt that the modeling process was not that clear-cut, as there were certain entities that end up sharing a lot of complicated relationships with other entities. Simply applying the binary and ternary relationship we learned in class was not sufficient to capture the complicated relationship among different entities in our database. Instead, an abnormal ternary relationship has to be applied among Product, Customer and Store because we modelled product individually rather than as a type, which was what we saw in class examples most of the time.

Additionally, we learnt that the way relationships are modeled greatly affects the meaning of the situation. For example, when two entities (e.g. factory and product) are linked together through another entity (e.g. warehouse order), their relationship can be shown only if the entity they are mutually tied to exists, which may not be desirable at times. Once we acknowledged this fact, we started looking into our database diagrams with increased scrutiny.

Appendix A: SQL Code for Database Creation

```
-------------------------
-- Drop tables if they already exist
-------------------------

if exists (
      select *
      from sys.tables
      where name = 'Factory'
      and type = 'U')
BEGIN
drop table Factory
END

if exists (
      select *
      from sys.tables
      where name = 'Warehouse_Order'
      and type = 'U')
BEGIN
drop table Warehouse_Order
END


if exists (
      select *
      from sys.tables
      where name = 'Warehouse'
      and type = 'U')
BEGIN
drop table Warehouse
END

if exists (
      select *
      from sys.tables
      where name = 'Product'
      and type = 'U')
BEGIN
drop table Product
END

if exists (
     select *
     from sys.tables
     where name = 'Problem'
     and type = 'U')
BEGIN
drop table Problem
END

if exists (
     select *
     from sys.tables
     where name = 'Complaint'
     and type = 'U')
BEGIN
drop table Complaint
END

if exists (
     select *
     from sys.tables
     where name = 'Customer'
     and type = 'U')
BEGIN
drop table Customer
END

if exists (
     select *
     from sys.tables
     where name = 'Purchase'
     and type = 'U')
BEGIN
drop table Purchase
END

if exists (
     select *
     from sys.tables
     where name = 'Store'
     and type = 'U')
BEGIN
drop table Store
END
```

```
if exists (
      select *
      from sys.tables
      where name = 'Store_Order'
      and type = 'U')
BEGIN
drop table Store_Order
END

---------------------------
-- Create the tables
---------------------------

create table Problem (
      Issue_ID int PRIMARY KEY,
      Complexity varchar(20),
       Description varchar(100));

create table Warehouse (
      WID int PRIMARY KEY,
      Location   varchar(20),
      Phone_Number varchar(20));

create table Store (
      SID int PRIMARY KEY,
      Name  varchar(20),
      Location varchar(20),
      Phone_Number varchar(20));

create table Factory (
      FID int PRIMARY KEY,
      Name  varchar(20),
      Location varchar(20),
      Phone_Number varchar(20));


create table Customer (
      CID int PRIMARY KEY,
      Name  varchar(20),
      Phone_Number    varchar(20),
Address varchar(20));

create table Warehouse_Order (
      WOID int PRIMARY KEY,
      Ship_Date  DATE,
      Shipping_Method varchar(20),
      Shipping_Cost decimal(9, 2),
      WID int,
FOREIGN KEY (WID) REFERENCES Warehouse(WID));

create table Store_Order (
      SOID int PRIMARY KEY,
      Ship_Date  DATE,
      Shipping_Method varchar(20),
      Shipping_Cost decimal(9, 2),
      SID int,
      WID int,
FOREIGN KEY (SID) REFERENCES Store(SID),
FOREIGN KEY (WID) REFERENCES Warehouse(WID));


create table Purchase (
      POID int PRIMARY KEY,
      Date DATE,
      CID  int,
      SID int,
FOREIGN KEY (CID) REFERENCES Customer (CID),
FOREIGN KEY (SID) REFERENCES Store (SID));


create table Product (
      Serial_Number varchar(20) PRIMARY KEY,
      Model varchar(20),
      Color varchar(20),
      Date_Produced DATE,
      WOID int NULL,
      SOID int NULL,
      POID int NULL,
      FID int
FOREIGN KEY(WOID) REFERENCES Warehouse_Order(WOID),
FOREIGN KEY(FID) REFERENCES Factory(FID),
FOREIGN KEY(SOID) REFERENCES Store_Order(SOID),
FOREIGN KEY(POID) REFERENCES Purchase(POID));

create table Complaint (
      COMPID int PRIMARY KEY,
      Date  DATE,
      Status varchar(20),
      Serial_Number varchar(20),
     Issue_ID int,
     CID int,
FOREIGN KEY (Serial_Number) REFERENCES Product(Serial_Number),
FOREIGN KEY(Issue_ID) REFERENCES Problem(Issue_ID),
FOREIGN KEY(CID) REFERENCES Customer(CID));



---------------------------
-- Populate the tables
---------------------------

insert into Factory values (1, 'FoxConn', 'Shanghai','8610738028374');
insert into Factory values (2, 'FoxConn', 'Fujian','8674839393021');
insert into Factory values (3, 'MacFac', 'Chongxing','8601947284930');
insert into Factory values (4, 'PhoneZone',
'Beijing','8672930193837');

insert into Warehouse values (1, 'Chicago','3127445000');
insert into Warehouse values (2, 'New York City','2126399675');
insert into Warehouse values (3, 'San Francisco','4155774400');
insert into Warehouse values (4, 'Denver','7208651111');

insert into Store values (1,
'Shadyside','Pittsburgh,PA','4123162460');
insert into Store values (2, 'Fashion Center','Washington
D.C.','7032361550');
insert into Store values (3, 'Beverly Center','Los Angeles,LA',
'4242040010');
insert into Store values (4, 'Knox St','Dallas,TX', '2145208532');


insert into Store_Order values (1,'05/06/2015','Ground',25.00,1,1);
insert into Store_Order values (2,'05/06/2015','Ground',30.00,2,2);
insert into Store_Order values (3,'05/06/2015','Ground',20.00,3,3);
insert into Store_Order values (4,'05/06/2015','Air',40.00,4,4);
insert into Store_Order values (5,'04/07/2015','Air',10.00,1,1);
insert into Store_Order values (6,'04/07/2015','Ground',20.00,2,2);
insert into Store_Order values (7,'04/07/2015','Air',25.00,3,3);
insert into Store_Order values (8,'04/07/2015','Ground',38.00,4,4);


insert into Problem values (1, 'Low', 'Mac Screen failure');
insert into Problem values (2, 'Medium', 'Mac Screen failure');
insert into Problem values (3, 'High', 'Mac Screen failure');
insert into Problem values (4, 'Low', 'iPhone Screen failure');
insert into Problem values (5, 'Medium', 'iPhone Screen failure');
insert into Problem values (6, 'High', 'iPhone Screen failure');
insert into Problem values (7, 'Low', 'Mac charger failure');
insert into Problem values (8, 'Medium', 'Mac charger failure');
insert into Problem values (9, 'High', 'Mac charger failure');
insert into Problem values (10, 'Low', 'iPhone charger failure');
insert into Problem values (11, 'Medium', 'iPhone charger failure');
insert into Problem values (12, 'High', 'iPhone charger failure');
insert into Problem values (13, 'Low', 'Mac OS failure');
insert into Problem values (14, 'Medium', 'Mac OS failure');
insert into Problem values (15, 'High', 'Mac OS failure');
insert into Problem values (16, 'Low', 'iPhone OS failure');
insert into Problem values (17, 'Medium', 'iPhone OS failure');
insert into Problem values (18, 'High', 'iPhone OS failure');


insert into Customer values (1,'Lydia','8008541110', '5032 Forbes
Ave,PA');
insert into Customer values (2,'Mandy','6468769282', '23 Ranch
Ave,TX');
insert into Customer values (3,'Nancy','4126946555', '220 Vegas Blvd,
NV');
insert into Customer values (4,'Rachel', '4128778383', '432 5th
Ave,NY');
insert into Customer values (5,'Sheldon', '4124804044', '3804 Madison
Ave,NY');
insert into Customer values (6,'Peter', '4125769896', '201 Plano
Rd,TX');
insert into Customer values (7,'Nicholas', '4129854281', '3804 Mission
St,LA');
insert into Customer values (8,'Alvin', '4189657723', '7625 K
Ave,DC');
insert into Customer values (9,'Kevin', '4129870302', '8823 Bayard St,
PA');
insert into Customer values (10,'Sherry', '4125462314', '201 Wisconsin
Ave,DC');

insert into Warehouse_Order values (1, '04/02/2015', 'Air', 19, 1);
insert into Warehouse_Order values (2, '05/02/2015', 'Cargo', 20, 2)
insert into Warehouse_Order values (3, '07/03/2015', 'Air', 16, 3)
insert into Warehouse_Order values (4, '08/03/2015', 'Air', 22, 4)
insert into Warehouse_Order values (5, '09/04/2015', 'Air', 23, 1)
insert into Warehouse_Order values (6, '10/04/2015', 'Cargo', 18, 2)
insert into Warehouse_Order values (7, '11/05/2015', 'Air', 22, 3)
insert into Warehouse_Order values (8, '12/05/2015', 'Air', 21, 4)

insert into Purchase values (1, '01/01/2015', 1, 1);
insert into Purchase values (2, '02/01/2015', 2, 2);
insert into Purchase values (3, '03/01/2015', 3, 4);
insert into Purchase values (4, '04/01/2015', 4, 3);
insert into Purchase values (5, '04/01/2015', 5, 3);
insert into Purchase values (6, '05/01/2015', 6, 1);
insert into Purchase values (7, '07/01/2015', 7, 3);
insert into Purchase values (8, '07/01/2015', 8, 3);

insert into Product values ('00100001','MacBook
Air','Silver','04/02/2015',1,1,1,1);
insert into Product values ('00100002','MacBook
Air','Silver','04/02/2015',2,2,2,2);
insert into Product values ('00100003','MacBook
Air','Silver','04/02/2015',3,3,3,3);
insert into Product values ('00100004','MacBook
Air','Silver','05/02/2015',4,4,4,4);
insert into Product values ('00100005','MacBook
Air','Silver','05/02/2015',5,5,5,1);
insert into Product values ('00100006','MacBook
Air','Silver','05/02/2015',6,6,6,2);
insert into Product values ('00100007','MacBook
Air','Silver','06/02/2015',7,7,7,3);
insert into Product values ('00100008','MacBook
Air','Silver','06/02/2015',8,8,8,4);
insert into Product values ('00100009','MacBook
Air','Silver','06/02/2015',1,1,1,1);
insert into Product values
('00200001','iPhone6s','Gold','04/02/2015',2,2,2,2);
insert into Product values
('00200002','iPhone6s','Gold','04/02/2015',3,3,3,3);
insert into Product values
('00200003','iPhone6s','Black','04/02/2015',4,4,4,4);
insert into Product values
('00200004','iPhone6s','Black','04/02/2015',5,5,5,1);
insert into Product values
('00200005','iPhone6s','Black','04/02/2015',NULL,NULL,NULL,1);


insert into Complaint values (1, '06/02/2015', 'Not yet started',
'00100001', 2, 1);
insert into Complaint values (2, '07/02/2015', 'Working', '00100002',
1, 2);
```

```
insert into Complaint values (3, '07/03/2015', 'Finished', '00100003',
3, 3);
insert into Complaint values (4, '09/03/2015', 'Finished', '00100004',
9, 4);
insert into Complaint values (5, '04/04/2015', 'Working', '00100005',
8, 5);
insert into Complaint values (6, '07/04/2015', 'Not yet started',
'00100001', 2, 1);
insert into Complaint values (7, '08/05/2015', 'Working', '00100002',
1, 2);
insert into Complaint values (8, '09/05/2015', 'Not yet started',
'00100003', 3, 3);
insert into Complaint values (9, '04/06/2015', 'Finished', '00100004',
9, 4);
insert into Complaint values (10, '06/06/2015', 'Not yet started',
'00100005', 8, 5);
```

Appendix B: Useful Queries

```
-------------------------
A. Products bought by a specific customer
-------------------------

select serial_number,model,color,date_produced
from customer join purchase
on customer.cid=purchase.cid
join product
on product.poid=purchase.poid
where customer.name='Nicholas'

-------------------------
B. Model with the most complaints
-------------------------

Select Model,count(CID) as Occurrence
From Complaint C, Product P
where C.Serial_Number = P.Serial_Number
Group By Model

-------------------------
C. Complete information about a specific product
-------------------------

Select *
From Product P
Left Outer Join Factory F
on P.FID = F.FID
Left Outer Join Warehouse_Order WO
on WO.WOID = P.WOID
Left Outer Join Warehouse W
on W.WID = WO.WID
Left Outer Join Store_Order SO
on P.SOID = SO.SOID
Left Outer Join Store S
on SO.SID = S.SID
Left Outer Join Purchase
on Purchase.POID = P.POID
Left Outer Join Customer C
on Purchase.CID = C.CID
where p.serial_number='00200004'
```

```
-------------------------
D. For every factory, number of defective products produced
-------------------------

select Factory.FID,Factory.name,count(distinct Product.Serial_Number)
from Product join Complaint
on Product.Serial_Number=Complaint.Serial_Number
join Factory
on Product.FID=Factory.FID
group by Factory.FID,Factory.name

-------------------------
E. For every factory, number of all products produced
-------------------------

select Factory.FID,Factory.name, count(Serial_Number)
from Product join Factory
on Product.FID=Factory.FID
group by Factory.FID,Factory.name

-------------------------
F. Defect rate for every factory
-------------------------

with alias as (Select F.FID, count(distinct P.Serial_Number) as
Defects
From Complaint, Product P, Factory F
where Complaint.Serial_Number = P.Serial_Number
and P.FID = F.FID
Group By F.FID),
voila as (Select F.FID, count(*) as Total
From  Product P, Factory F
where P.FID = F.FID
Group by F.FID)
Select alias.FID, cast(round(100.0*Defects / Total,2) as
numeric(10,2)) as Defective_Rate
From alias, voila
where alias.FID = voila.FID
```

Appendix C: Insert and Modification queries

```
-------------------------
```

```
A. Create new warehouse order
-------------------------

insert into Warehouse_Order values (9, '15/05/2015', 'Cargo', 20, 2)

-------------------------
B. Update product after shipped to warehouse and store
-------------------------

UPDATE Product
SET WOID=8, SOID=8
WHERE Serial_Number='00200005';

-------------------------
C. Cancel store order
-------------------------

UPDATE Product
SET SOID=NULL
WHERE Product.SOID=1
DELETE FROM Store_Order
WHERE SOID=1;
```