Ryan Hoff

PART A: Coffee database

Question 1 - Identify various functional dependencies. Using functional dependencies, evaluate the normal form of the table (Excel spreadsheet)? Justify.

From the table, the primary key is a composite of, Product ID, Area Code and Date
Therefore, the functional relationships are as follows
1.ProductID links to the Product which in turn links to Product Line, Type and Product Type.
2.Area Code links to the State, Market and Market Size
3.The combination primary key (Product ID, Area Code, Date) links to all other database information, Profit, Margin, Sales, COGS, Total Expenses, Marketing, Inventory, Budget Profit, Budget Margin, Budget Sales, and Budget COGS.
These dependencies prove that the table as given is in 1NF form.

Question 2 - Normalize the database into 3rd normal form using functional dependencies identified above. Identify the various tables including primary keys and foreign keys. (Feel free to create unique keys when you have composite primary keys)

Entities

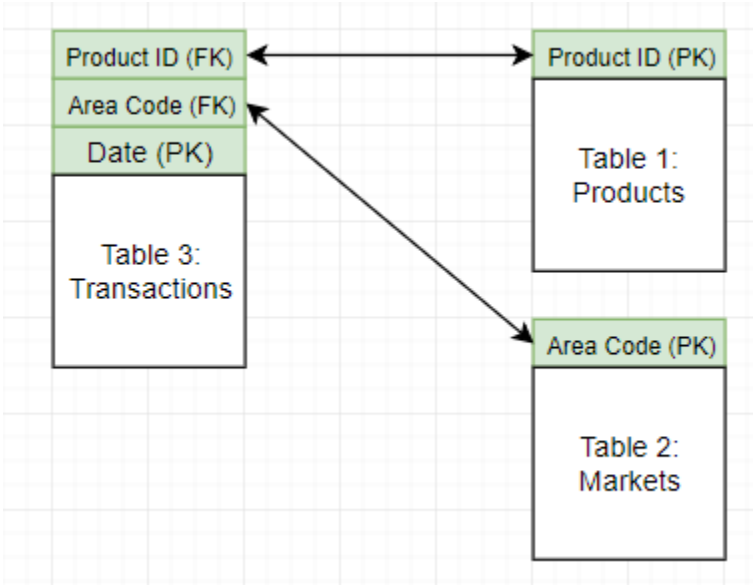
Table 1: Products				
Product ID (PK)	Product	Product Line	Type	Product Type

Table 2: Markets			
Area Code (PK)	Market	State	Market Size

Table 3: Transactions	Product ID (FK)
	Area Code (FK)
	Date (PK)
	Sales(Monthly)
	Profit (Monthly)
	Margin (Monthly)
	Inventory
	Total Expenses (Other than Product Cost)
	Marketing Expenses
	COGS
	Budget Profit
	Budget Margin
	Budget Sales
	Budget COGS

Ryan Hoff

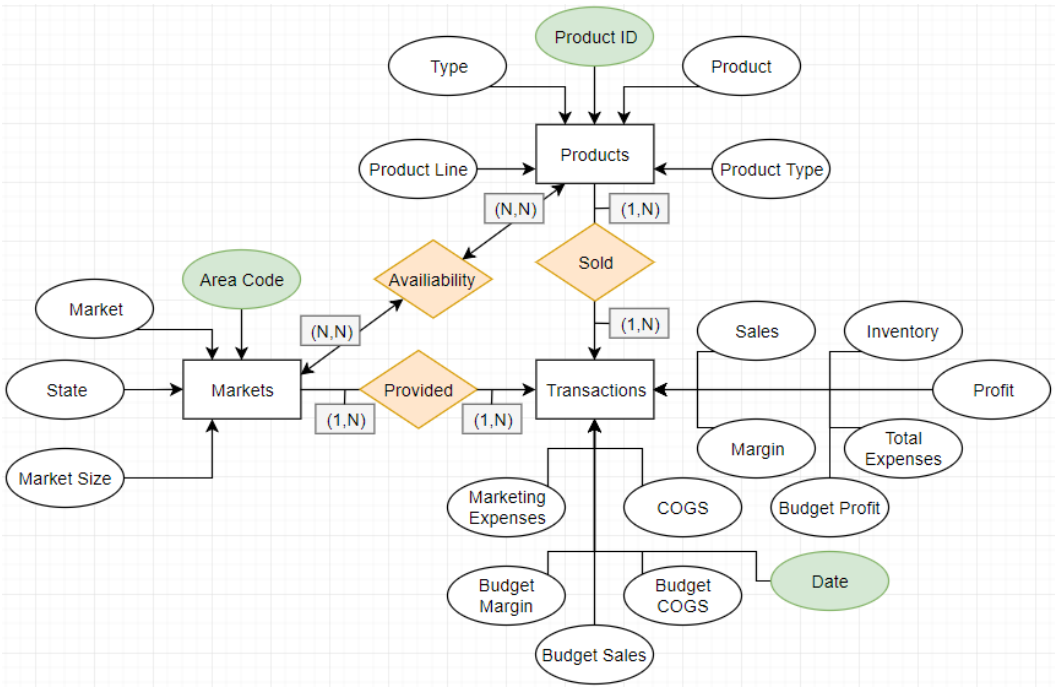
Entities and Keys Chart



Question 3 - Now, identify various entities and relationship types (1-1; 1-M; M-N), draw the E-R diagram. Convert the logical schema into actual table structure. Clearly identify PKs and FKs in the table structure.

1. Each product may be sold in multiple markets, but for any sale we can only have one market and product. So, the relationship of Markets to Sales is one to many (1,N) and Products to Sales is one to many (1,N)
2. Each market may sell multiple products, so the relationship of Products to Markets is Many to Many (N,N)

E-R Diagram



Ryan Hoff

SQL Code

```
CREATE TABLE Products(
     ProductId integer not null,
     "Product" CHAR(25) not null,
     "Product Line" CHAR(25) not null,
     "Type" CHAR(25) not null,
     "Product Type" CHAR(25) not null,
     PRIMARY KEY(ProductId),
     CONSTRAINT Check_Prod_Type CHECK("Product Type" IN
('Coffee', 'Espresso', 'Herbal  Tea', 'Tea')),
     CONSTRAINT Check_Line CHECK("Product Line" IN
('Beans', 'Leaves')),
     CONSTRAINT Check_Type CHECK("Type" IN
('Regular', 'Decaf')));

CREATE TABLE Markets(
    "Area Code" INTEGER not null,
    "State" CHAR(25) not null,
    "Market" CHAR(25) not null,
    "Market Size" CHAR(25) not null,
    PRIMARY KEY("Area Code"),
    CONSTRAINT Check_Market_Size CHECK('Market Size' IN
('Major Market', 'Small Market')),
    CONSTRAINT Check_Market CHECK('Market' IN
('West', 'East', 'South', 'Central'))
);

CREATE TABLE Sales(
    ProductId INTEGER not null,
    "Area Code" INTEGER not null,
    "Date" DATE not null,
    Profit REAL not null,
    Sales REAL not null,
    COGS REAL not null,
    "Total Expenses" REAL,
    Marketing REAL,
    Inventory REAL,
    "Budget Profit" CHAR(10) REAL,
    "Budget Margin" CHAR(10) REAL,
    "Budget Sales" CHAR(10) REAL,
    "Budget COGS" CHAR(10) REAL,
    PRIMARY KEY(ProductID, "Area Code", "Date"),
    FOREIGN KEY(ProductID) REFERENCES Products(ProductID),
    FOREIGN KEY("Area Code") REFERENCES Markets("Area Code")
);
```

Ryan Hoff

PART B: Tommy and Tom (TT)

Question 1 - Identify various entities and the relationships and draw the E-R diagram. Convert the E-R diagram into tables (relational model).

TT have 100s of products each with more than 10,000 components. For each of their products they have a bill of materials(components) and each of their components should be linked to supplier information. The supplier information which is pertinent is price, quality and reliability for each of the components they supply. TT also have a system which maintains order information which includes the date the order was place, expected delivery date, the actual delivery date (if delivered) and the shipment certification.

Entities

Table 1: Product Master		
Product ID (PK)	Product Name	ReOrder Level

Table 2: Bill of Materials		
Product ID (FK)	Component ID (PK)	Quantity

Table 3: Inventory	
Component ID (PK)	Inventory

Table 4: Supplier Master		
Supplier ID (PK)	Supplier Name	Supplier Address

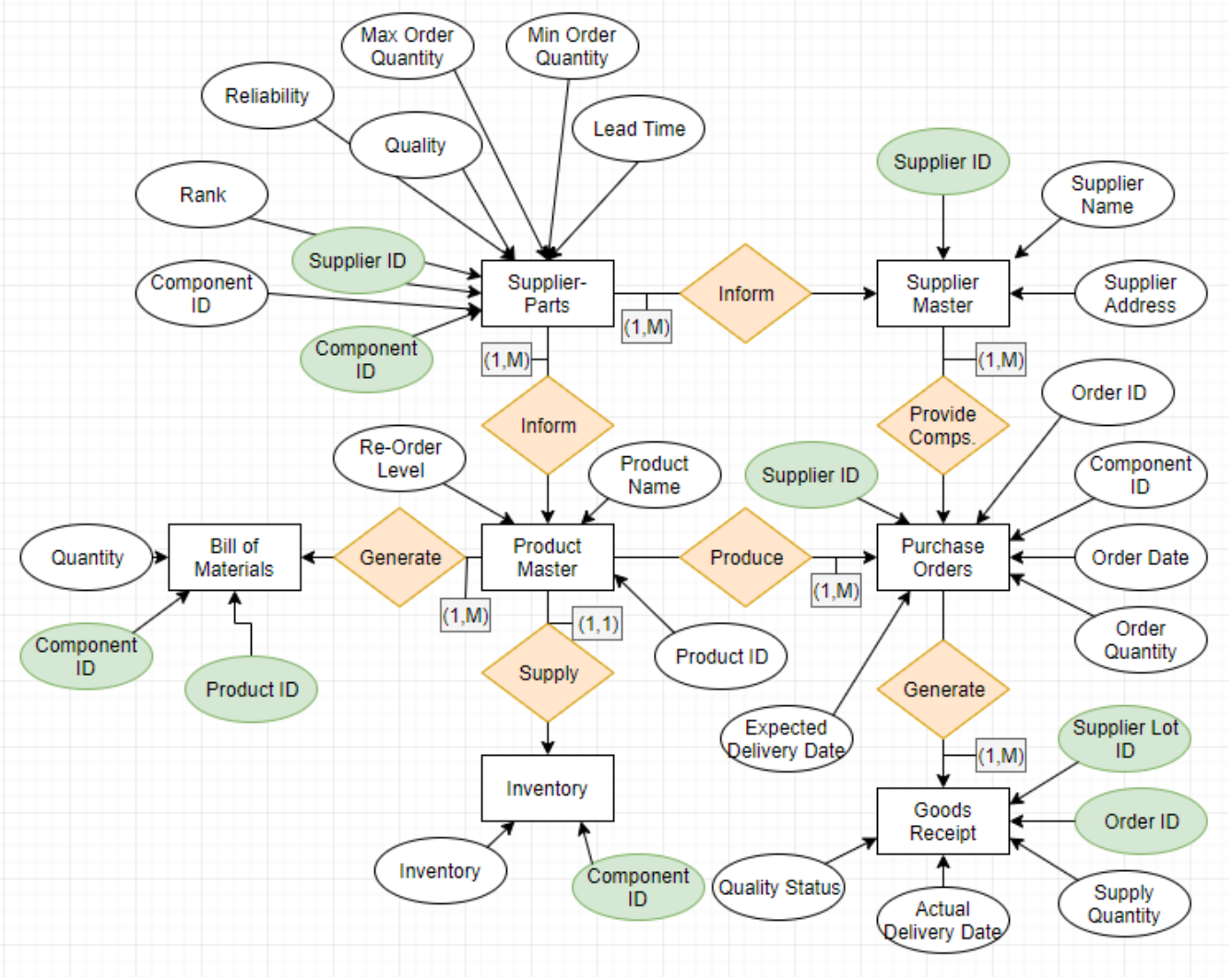
Table 5: Supplier-Parts	
	Supplier ID (FK)
	Component ID (FK)
	Unit Price
	Quality
	Reliability
	Rank
	Lead Time
	Min. Order Quantity
	Max. Order Quantity

Table 6: Purchase Orders	
	Order ID (Pk)
	Component ID
	Supplier ID
	Order Date
	Order Quantity
	Expected Delivery Date

Table 7: Goods Receipt	
	Order ID (PK)
	Supply Lot ID (PK)
	Supply Quantity
	Actual Delivery Date
	Quality Status

Ryan Hoff

E-R Diagram



SQL

```
CREATE TABLE Products(
    ProductId NUMBER not null,
    "Product Name" char(25) not null,
    "ReOrder Level" NUMBER not null,
    PRIMARY KEY(ProductId)
);

CREATE TABLE BOM(
    ProductId NUMBER not null,
    ComponentId NUMBER not null,
    Quantity NUMBER not null,
    PRIMARY KEY(ProductId, ComponentId),
    FOREIGN KEY(ProductId) REFERENCES Products(ProductId),
    FOREIGN KEY(ComponentId) REFERENCES Products(ProductId)
);
```

Ryan Hoff

```
CREATE TABLE Inventory(
    ComponentId NUMBER not null,
    Inventory NUMBER not null,
    PRIMARY KEY(ComponentId),
    FOREIGN KEY(ComponentId) REFERENCES Products(ProductId)
);

CREATE TABLE SupplierMaster(
    SupplierId NUMBER not null,
    SupplierName char(25) not null,
    SupplierAddress char(50) not null,
    PRIMARY KEY(SupplierId)
);

CREATE TABLE SupplierParts(
    SupplierId NUMBER not null,
    ComponentId NUMBER not null,
    "Unit Price" REAL not null,
    Quality REAL not null,
    Reliability REAL not null,
    Rank REAL not null,
    "Lead Time" REAL not null,
    "Min Order Quantity" REAL not null,
    "Max Order Quantity" REAL not null,
    PRIMARY KEY(SupplierId, ComponentId),
    FOREIGN KEY(SupplierId) REFERENCES SupplierMaster(SupplierId),
    FOREIGN KEY(ComponentId) REFERENCES Products(ProductId)
);

CREATE TABLE OrderInformation(
    OrderId NUMBER not null,
    SupplierId NUMBER not null,
    ComponentId NUMBER not null,
    "Order Date" DATE not null,
    "Order Quantity" NUMBER not null,
    "Expected Delivery Date" DATE not null,
    PRIMARY KEY(OrderId),
    FOREIGN KEY(SupplierId) REFERENCES SupplierMaster(SupplierId),
    FOREIGN KEY(ComponentId) REFERENCES Products(ProductId)
);
CREATE TABLE GoodsReceipt(
    OrderId NUMBER not null,
    "Supply Lot Id" NUMBER not null,
    "Supply Quantity" NUMBER not null,
    "Actual Delivery Date" DATE not null,
    "Quality Status" char(25) not null,
    PRIMARY KEY(OrderId, "Supply Lot Id"),
    FOREIGN KEY(OrderId) REFERENCES OrderInformation(OrderId),
    CONSTRAINT Check_Quality CHECK ("Quality Status" IN
    'SATISFACTORY', 'UNSATISFACTORY'))
);
```

Ryan Hoff

Question 2 - What if an order can be supplied in small quantities at various times (that is, a large order can be split into many small orders). How does your E-R diagram and table structure change?

The overall structure will not change. Each individual 'Part' or smaller portion of the order can just be logged under a different Supplier Lot ID. The table entries may still be used in the same fashion, but supply quantity will represent the smaller order portions. The orders can be placed until the full order is fulfilled and quality status can be used to determine whether or not the order is completed. An additional determinant variable may be added to the Goods Receipt table called "Order Status" if need be, but the "Quality Status" should work.