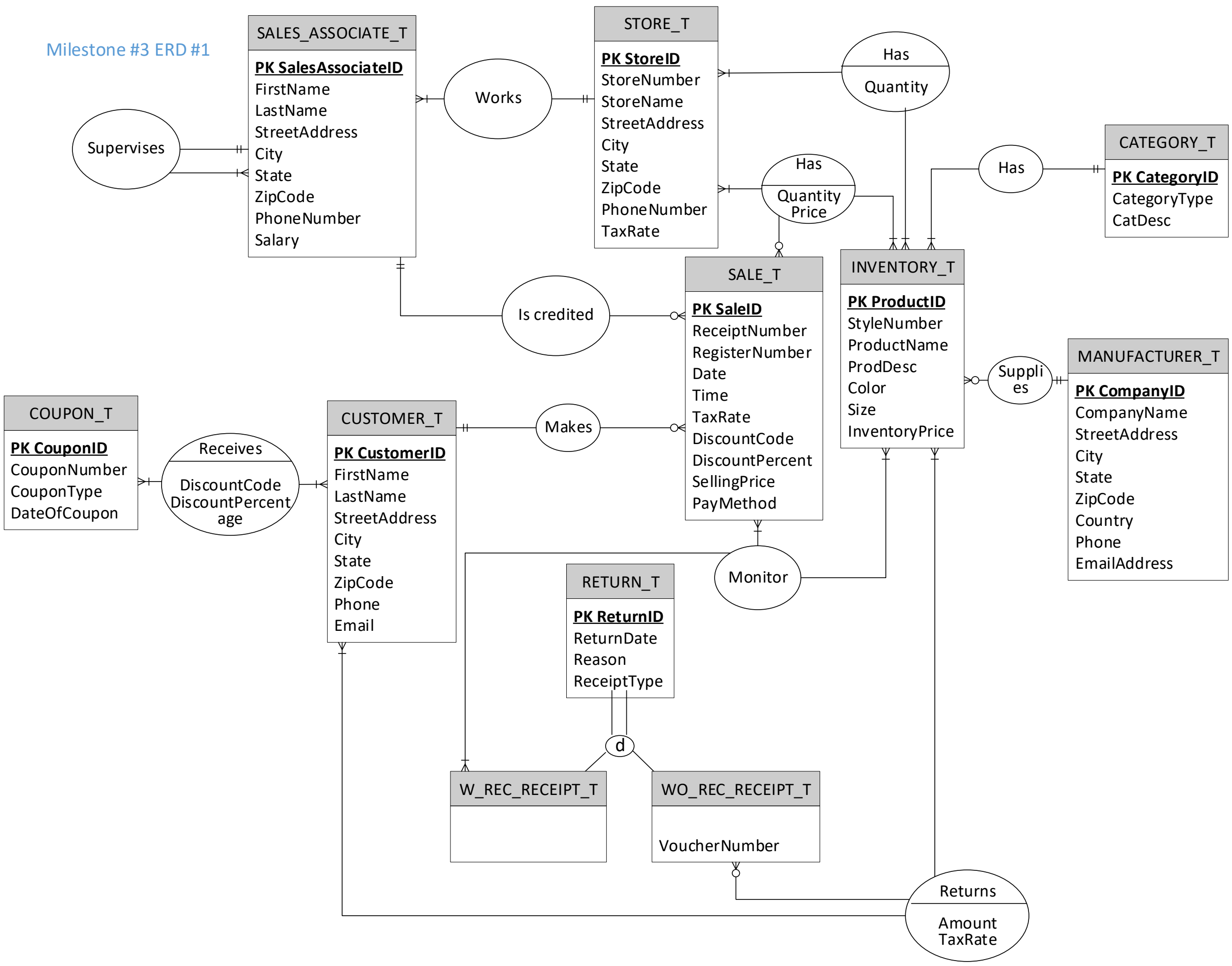
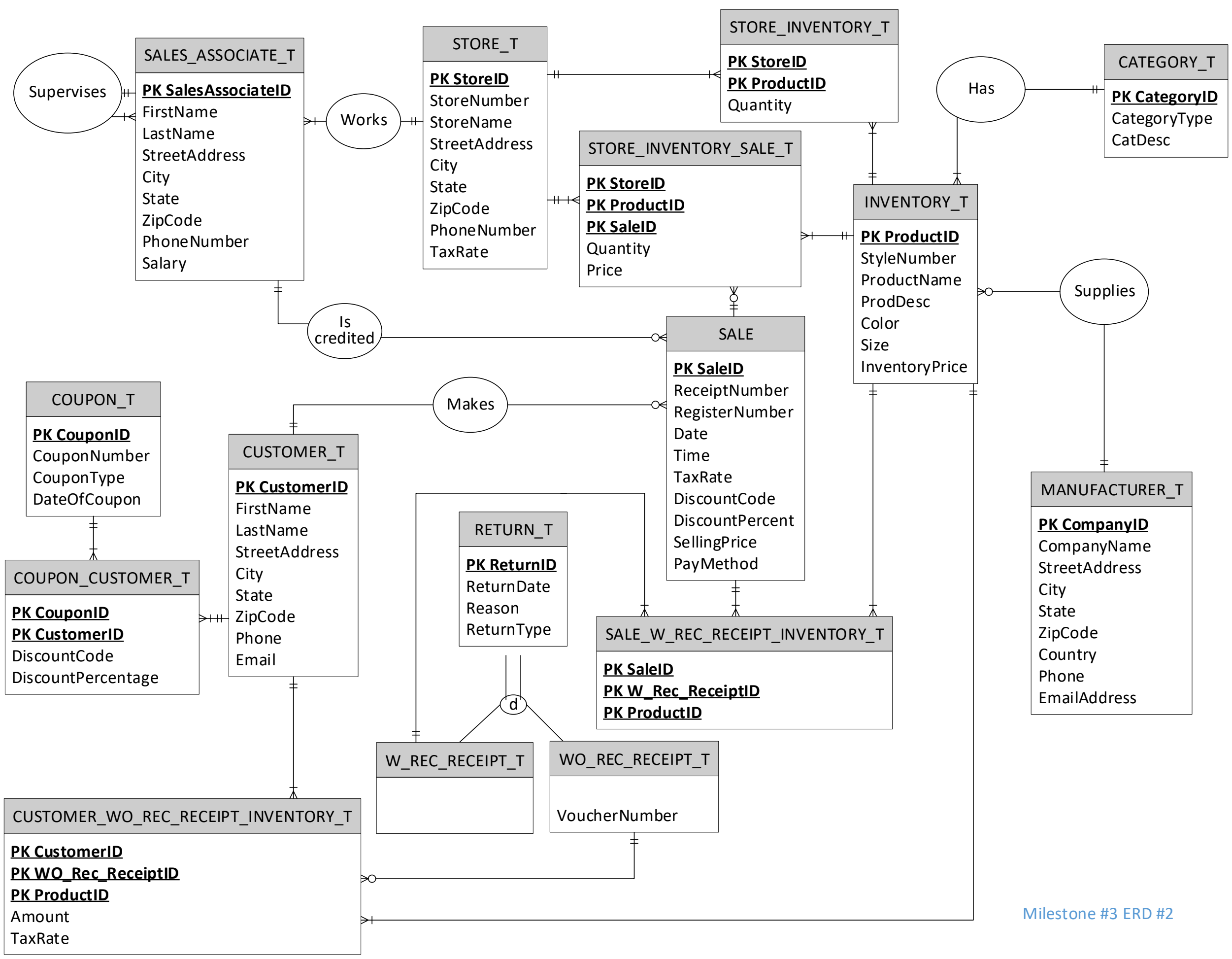
HBI Database Report

Patel, Romil

EERD #1

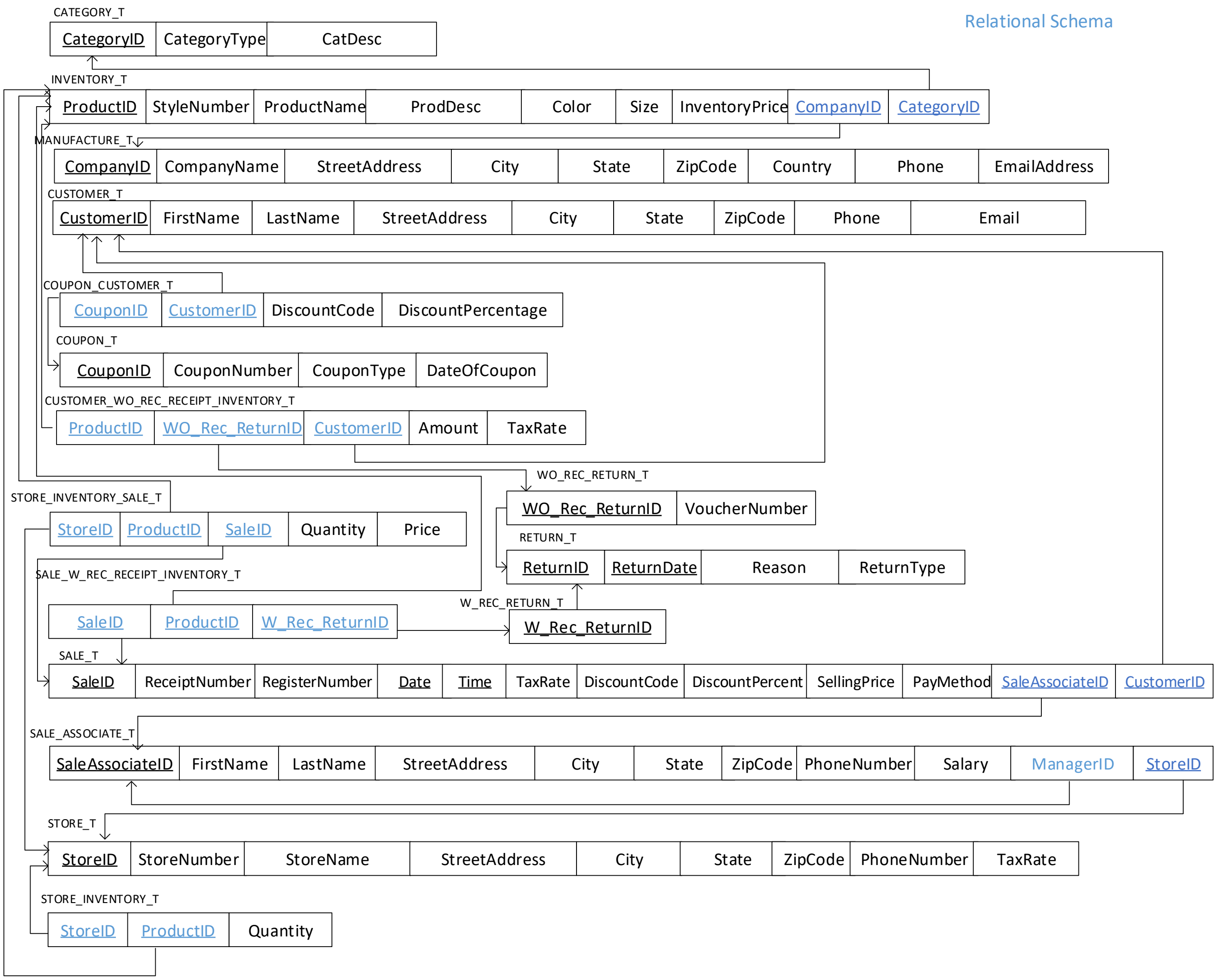

Milestone #3 ERD #1
SALES_ASSOCIATE_T
PK SalesAssociateID
FirstName
LastName
StreetAddress
City
State
ZipCode
PhoneNumber
Salary
Supervises
Works
STORE_T
PK StoreID
StoreNumber
StoreName
StreetAddress
City
State
ZipCode
PhoneNumber
TaxRate
Has
Quantity
Has
Quantity
Price
Has
CATEGORY_T
PK CategoryID
CategoryType
CatDesc
SALE_T
PK SaleID
ReceiptNumber
RegisterNumber
Date
Time
TaxRate
DiscountCode
DiscountPercent
SellingPrice
PayMethod
INVENTORY_T
PK ProductID
StyleNumber
ProductName
ProdDesc
Color
Size
InventoryPrice
Suppli
es
MANUFACTURER_T
PK CompanyID
CompanyName
StreetAddress
City
State
ZipCode
Country
Phone
EmailAddress
Is credited
COUPON_T
PK CouponID
CouponNumber
CouponType
DateOfCoupon
Receives
DiscountCode
DiscountPercent
age
CUSTOMER_T
PK CustomerID
FirstName
LastName
StreetAddress
City
State
ZipCode
Phone
Email
Makes
Monitor
RETURN_T
PK ReturnID
ReturnDate
Reason
ReceiptType
d
W_REC_RECEIPT_T
WO_REC_RECEIPT_T
VoucherNumber
Returns
Amount
TaxRate

EERD #2

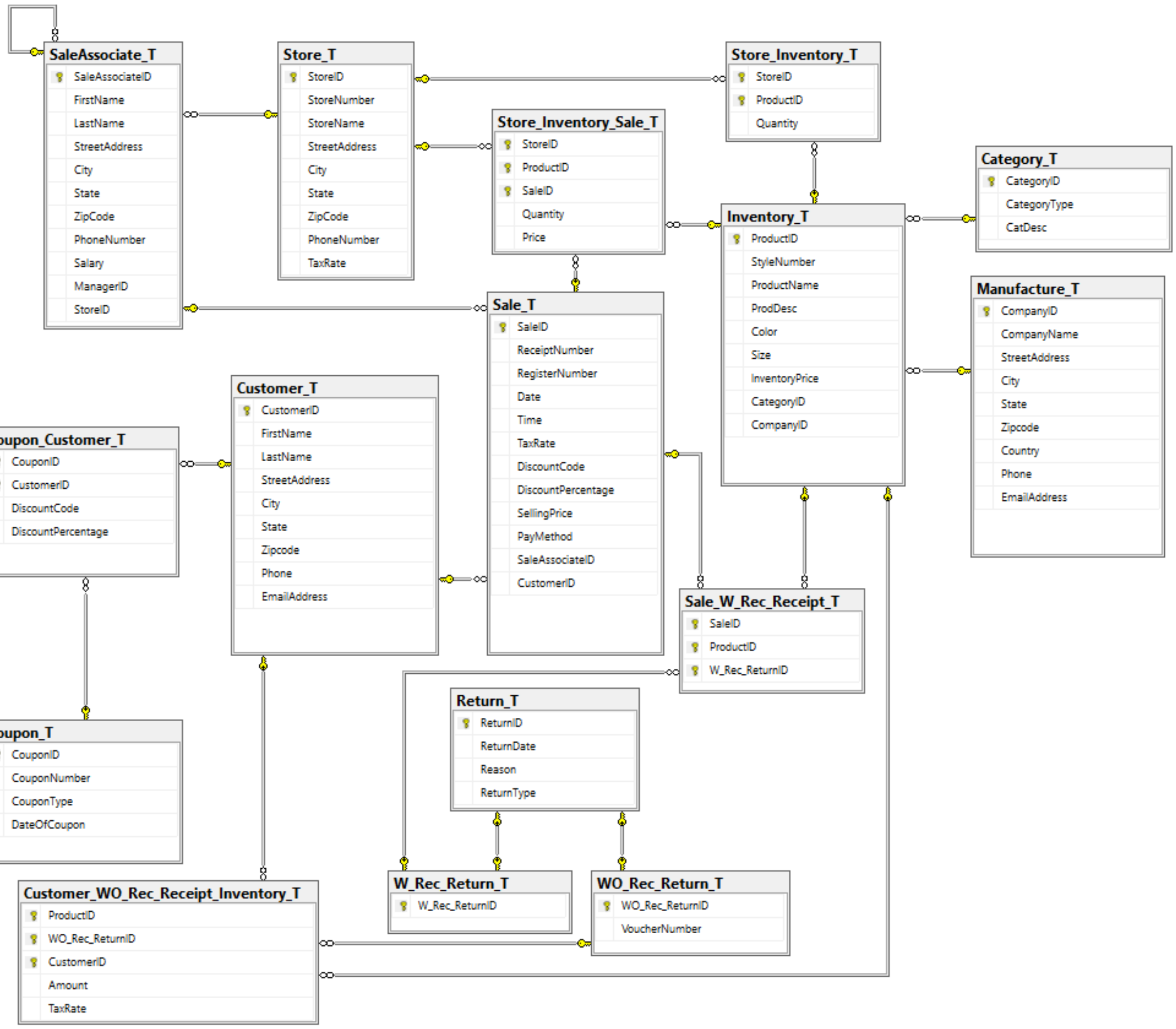


SALES_ASSOCIATE_T
PK SalesAssociateID
FirstName
LastName
StreetAddress
City
State
ZipCode
PhoneNumber
Salary
Supervises
Works
STORE_T
PK StoreID
StoreNumber
StoreName
StreetAddress
City
State
ZipCode
PhoneNumber
TaxRate
STORE_INVENTORY_T
PK StoreID
PK ProductID
Quantity
STORE_INVENTORY_SALE_T
PK StoreID
PK ProductID
PK SaleID
Quantity
Price
Has
CATEGORY_T
PK CategoryID
CategoryType
CatDesc
INVENTORY_T
PK ProductID
StyleNumber
ProductName
ProdDesc
Color
Size
InventoryPrice
Supplies
Is credited
SALE
PK SaleID
ReceiptNumber
RegisterNumber
Date
Time
TaxRate
DiscountCode
DiscountPercent
SellingPrice
PayMethod
COUPON_T
PK CouponID
CouponNumber
CouponType
DateOfCoupon
Makes
CUSTOMER_T
PK CustomerID
FirstName
LastName
StreetAddress
City
State
ZipCode
Phone
Email
COUPON_CUSTOMER_T
PK CouponID
PK CustomerID
DiscountCode
DiscountPercentage
RETURN_T
PK ReturnID
ReturnDate
Reason
ReturnType
d
SALE_W_REC_RECEIPT_INVENTORY_T
PK SaleID
PK W_Rec_ReceiptID
PK ProductID
MANUFACTURER_T
PK CompanyID
CompanyName
StreetAddress
City
State
ZipCode
Country
Phone
EmailAddress
W_REC_RECEIPT_T
WO_REC_RECEIPT_T
VoucherNumber
CUSTOMER_WO_REC_RECEIPT_INVENTORY_T
PK CustomerID
PK WO_Rec_ReceiptID
PK ProductID
Amount
TaxRate
Milestone #3 ERD #2

Relational Schema



Database Diagram



Assumptions

- Store may have at least one or more than one product inventory.
- Product Inventory may at least one store or more than one store.
- Manufacturer may not supply any inventory.
- Customer may not make any sales.
- Sale may be made by one and only one customer.
- Customer may use at least one payment method.
- Store may have at least one inventory.
- Store may not have any sales.
- Sale may be made at least one store.
- Store may have at least one sales associate.
- Sales associates may not help any customer, if customer don't need help.
- Sales associates may not credit for any sales.
- Manager may supervise at least one sales associate.
- At least one customer may get at least one coupon.
- For return with receipt there may be at least one sale or at least one item to be purchased.
- For return without receipt at least one or many items may purchase by at least one customer.

Data Dictionary

TABLE: CATEGORY_T

COLUMN NAME	TYPE/LENGTH	CONSTRAINTS
CategoryID	**INT**	**Primary key**
CategoryType	**CHAR**	
CatDesc	**NTEXT**	

TABLE: INVENTORY_T

COLUMN NAME	TYPE/LENGTH	CONSTRAINTS
ProductID	**INT**	**Primary key**
StyleNumber	**VARCHAR (30)**	
ProductName	**CHAR (20)**	
ProdDesc	**NTEXT**	
Color	**CHAR (20)**	
Size	**VARCHAR (8)**	

InventoryPrice	**FLOAT (10)**	
CompanyID	**INT**	**Foreign key**
CategoryID	**INT**	**Foreign key**

TABLE: MANUFACTURE_T

COLUMN NAME	TYPE/LENGTH	CONSTRAINTS
CompanyID	**INT**	**Primary key**
CompanyName	**CHAR (30)**	
StreetAddress	**VARCHAR (30)**	
City	**VARCHAR (20)**	
State	**VARCHAR (20)**	
ZipCode	**CHAR (10)**	
Country	**VARCHAR (20)**	
Phone	**NVARCHAR (13)**	
EmailAddress	**VARCHAR (100)**	

TABLE: CUSTOMER_T

COLUMN NAME	TYPE/LENGTH	CONSTRAINTS
CustomerID	**INT**	**Primary key**
FirstName	**VARCHAR (30)**	
LastName	**VARCHAR (30)**	
StreetAddress	**VARCHAR (30)**	
City	**VARCHAR (20)**	
State	**VARCHAR (20)**	
ZipCode	**CHAR (10)**	
Phone	**NVARCHAR (13)**	
EmailAddress	**VARCHAR (100)**	

TABLE: COUPON_CUSTOMER_T

COLUMN NAME	TYPE/LENGTH	CONSTRAINTS
CouponID	**INT**	**Primary key: Foreign key**
CustomerID	**INT**	**Primary key: Foreign key**
DiscountCode	CHAR (30)	
DiscountPercentage	FLOAT (4)	

TABLE: COUPON_T

COLUMN NAME	TYPE/LENGTH	CONSTRAINTS
CouponID	**INT**	**Primary key**
CouponNumber	**VARCHAR (30)**	
CouponType	**CHAR (2)**	
DateOfCoupon	**DATE**	

TABLE: CUSTOMER_WO_REC_RECEIPT_INVENTORY_T

COLUMN NAME	TYPE/LENGTH	CONSTRAINTS
ProductID	**INT**	**Primary key: Foreign key**
WO_Rec_ReturnID	**INT**	**Primary key: Foreign key**
CustomerID	**INT**	**Primary key: Foreign key**
Amount	**FLOAT (9)**	
TaxRate	**FLOAT (3)**	

TABLE: STORE_INVENTORY_SALE_T

COLUMN NAME	TYPE/LENGTH	CONSTRAINTS
StoreID	**INT**	**Primary key: Foreign key**
ProductID	**INT**	**Primary key: Foreign key**
SaleID	**INT**	**Primary key: Foreign key**
Quantity	**INT**	
Price	**FLOAT (10)**	

TABLE: SALE_W_REC_RECEIPT_INVENTORY_T

COLUMN NAME	TYPE/LENGTH	CONSTRAINTS
SaleID	**INT**	**Primary key: Foreign key**
ProductID	**INT**	**Primary key: Foreign key**
W_Rec_ReturnID	**INT**	**Primary key: Foreign key**

TABLE: SALE_T

COLUMN NAME	TYPE/LENGTH	CONSTRAINTS
SaleID	**INT**	**Primary key**
ReceiptNumber	**INT**	
RegisterNumber	**INT**	
Date	**DATE**	**Primary key**
Time	**TIME**	**Primary key**
TaxRate	**FLOAT (3)**	
DiscountCode	**CHAR (30)**	
DiscountPercentage	**FLOAT (4)**	
SellingPrice	**FLOAT (10)**	
PayMethod	**CHAR**	
SaleAssociateID	**INT**	**Foreign key**
CustomerID	**INT**	**Foreign key**

TABLE: SALE_ASSOCIATE_T

COLUMN NAME	TYPE/LENGTH	CONSTRAINTS
SaleAssociateID	**INT**	**Primary key**
FirstName	**VARCHAR (30)**	
LastName	**VARCHAR (30)**	
StreetAddress	**VARCHAR (30)**	
City	**VARCHAR (20)**	
State	**VARCHAR (20)**	
ZipCode	**CHAR (10)**	
PhoneNumber	**NVARCHAR (13)**	
Salary	**FLOAT (10)**	
ManagerID	**INT**	**Foreign key**
StoreID	**INT**	**Foreign key**

TABLE: STORE_T

COLUMN NAME	TYPE/LENGTH	CONSTRAINTS
StoreID	**INT**	**Primary key**
StoreNumber	**NCHAR (30)**	
StoreName	**VARCHAR (30)**	
StreetAddress	**VARCHAR (30)**	
City	**VARCHAR (20)**	
State	**VARCHAR (20)**	
ZipCode	**CHAR (10)**	
PhoneNumber	**NVARCHAR (13)**	
TaxRate	**CHAR (30)**	

TABLE: STORE_INVENTORY_T

COLUMN NAME	TYPE/LENGTH	CONSTRAINTS
SaleID	**INT**	**Primary key: Foreign key**
ProductID	**INT**	**Primary key: Foreign key**
Quantity	**INT**	

TABLE: WO_REC_RETURN_T

COLUMN NAME	TYPE/LENGTH	CONSTRAINTS
WO_Rec_ReturnID	**INT**	**Primary key**
VoucherNumber	**VARCHAR (9)**	

TABLE: RETURN_T

COLUMN NAME	TYPE/LENGTH	CONSTRAINTS
ReturnID	**INT**	**Primary key**
ReturnDate	**DATE**	**Primary key**
Reason	**TEXT**	
ReturnType	**VARCHAR (9)**	

TABLE: W_REC_RETURN_T

COLUMN NAME	TYPE/LENGTH	CONSTRAINTS
W_Rec_ReturnID	**INT**	**Primary key**

Discussion

I have totally sixteen tables in the logical design. I discussed all these tables to see if they are in normal forms. As below there are the description of the different normalization forms for our tables:

First Normal Form (1NF):

- No Multivalued Attribute
- None of the entities has any multivalued attributes, so relational schema is already in First Normal Form.
- **For Example:** Inventory table has several attributes like ProductID, StyleNumber, ProductName, Description, Color, Size, Inventory, Price. None of these attributes has any multivalued attributes, but all of these are independent by its own.
- By all three anomalies, First Normal Form is satisfied because we can do Insert, Update and Delete for the tables.
- By Insert, Delete and Update, data of other tables does not affect.

Second Normal Form(2NF):

- 1NF (No Multivalued), No Partial Dependency
- It is a normalization form which has attributes of the entities which represent primary key of that entities.
- **For Example:** Inventory table has values different attributes which represent to ProductID, Store table has different attributes which represent to StoreID, Sale table has different attributes which represent to SaleID. These three tables are independent to each other. There is an associate table of these three tables named Store_Inventory_Sale which has attributes of these three tables primary keys, Quantity and Price as other attributes which represent to that table. Foreign keys are representing to the primary key of each individual table.
- By all three anomalies, Second Normal Form is satisfied because we can do Insert, Update and Delete for the tables also with the associate tables.
- By Insert, Delete and Update, data of other tables does not affect.
- This representation shows that relational schema is in Second Normal Form(2NF).

Third Normal Form(3NF):

- 1NF (No Multivalued), No Partial Dependencies and No Transitive Dependencies
- It is a normalization form in which there should have non-key attribute, and that attribute create primary key for new relation.
- In a relation is a functional dependency between primary key and one or more non-key attribute which are dependent on primary key via another non-key attribute.
- 3NF form could be resolve in 1:M relations.

- **For Example:** Sale_Associate and Store has 1:M relationship in which Sale_Associate has attributes like FirstName, LastName, StreetAddress, City, State, ZipCode, PhoneNumber, Salary and Store table has attributes like StoreNumber, StoreName, StreetAddress, City, State, ZipCode, PhoneNumber, TaxRate. Sale_Associate has non-key attribute of Store_ID which represent the Store. One Sale Associate works for only one store only.
- By all three anomalies, Third Normal Form is satisfied because we can do Insert, Update and Delete for the tables.
- By Insert, Delete and Update, data of other tables does not affect.
- This representation shows that relational schema is in Third Normal Form (3NF).

I finally find that our logical design is normalized till Third Normal Form.

There is one disadvantage of normalization that if I have multiple relations between tables I must make joints to find the data with different table names. It will increase execution time for finding data and also cost of query is higher too.

I need denormalization in the future. When I maintain history of store_inventory_sale table, I want to see directly from one table, denormalization is a good choice because I do not need to make a change to the value. The disadvantage of anomalies will not happen in that case. Additionally, generating report will speed up by denormalization. What's more, if I need to improve performance (e.g. I want to see sales of store), I could use denormalization of frequently used tables which are sale table and store table. Denormalization is a good technical method in some situations. However, it has disadvantages. I should consider both benefits and costs when deciding whether to use it or not.

Data Requirement and Queries

/*1*/ /* Information Selection Query */

/* List the ProductID, ProductName, Inventory Price for all the Products with a InventoryPrice in range of between 50 and 250 Put the list in order of CategoryID, then descending order of Inventory Price */

Select CategoryID, InventoryPrice, ProductID, ProductName

FROM Inventory_T

where InventoryPrice BETWEEN 50 AND 250

ORDER BY CategoryID, InventoryPrice DESC;

	CategoryID	InventoryPrice	ProductID	ProductName
1	1	80	3	Shoes
2	2	250	7	Jackets
3	2	150	8	Watch
4	2	110	4	Dress
5	2	50	1	Hangbag
6	3	100	5	Toys
7	3	70	6	Blanket

/*2*/ /*Aggregate Function Query */

/*What is the total of inventory price for all the products in CategoryID 2*/

SELECT SUM(InventoryPrice) AS 'Total'

FROM Inventory_T

Where CategoryID IN (1,2,3);

	Total
1	1395

/*3*/ /*2 table Query */

/*For sales with the ID's 2705,2708,2709, show the SaleID, Date of Sale, Time of Sale, CustomerID,, and the Phone number of the Customer who purchases the product. Put the list in descending order of SaleID */

SELECT S.SaleID, S.Date, C.CustomerID, C.FirstName ,C.Phone

FROM Sale_T AS S, Customer_T AS C

WHERE S.CustomerID = C.CustomerID

AND S.SaleID BETWEEN 2700 AND 2707

ORDER BY S.SaleID DESC;

	SaleID	Date	CustomerID	FirstName	Phone
1	2707	2019-04-05	82	Ada	8051235896
2	2706	2019-05-05	85	Salliea	8058946230
3	2705	2019-05-01	82	Ada	8051235896
4	2704	2019-03-08	82	Ada	8051235896
5	2703	2019-01-15	88	Carole	8058996540
6	2702	2019-04-18	87	Mary	8050009630
7	2701	2019-05-02	85	Salliea	8058946230
8	2700	2019-05-15	89	Magda	8058963332

/*4*/ /*3 Table Query */

/*List the CategoryID, CategoryType, ProductID, ProductName, InventoryPrice, CompanyName for all the products that belong in the categories 1 and 2, also which has CompanyName ZARA, Michael Kors. Put List in order of CategoryID, then Descending order of Inventory Price*/

SELECT CT.CategoryID, CT.CategoryType, CT.CatDesc, I.ProductID, I.ProductName, I.InventoryPrice, M.CompanyName

FROM Category_T AS CT, Inventory_T AS I, Manufacture_T AS M

WHERE M.CompanyID=I.CompanyID

AND I.CategoryID=Ct.CategoryID

AND CT.CategoryID IN (1,2)

AND M.CompanyName IN ('ZARA', 'Michael Kors', 'ADIDAS','NIKE');

	CategoryID	CategoryType	CatDesc	ProductID	ProductName	InventoryPrice	CompanyName
1	2	W	Women	1	Hangbag	50	MICHAEL KORS
2	2	W	Women	4	Dress	110	ZARA
3	1	M	Men	7	Jackets	250	NIKE
4	1	M	Men	8	Watch	150	ADIDAS
5	2	W	Women	10	Jwellery	500	ZARA

/*5*/ /*4 table Query */

/* For all sale from customers with ID's 1,2,3, List the CustomerName, SaleID, ProductName, Quantity, Price, and individual total of (Quantity * Price). USe the column alias 'Total' for (Quantity * Price). Put your list in order of CustomerName, then Descending order of SaleDate*/

SELECT CONCAT(C.FirstName,C.LastName) AS 'Customer Name', S.SaleID, S.Date, I.ProductName, SIS.Quantity, SIS.Price, (SIS.Quantity * SIS.Price) AS 'Total'

FROM Customer_T AS C, Sale_T AS S, Store_Inventory_Sale_T AS SIS, Inventory_T AS I

WHERE C.CustomerID=S.CustomerID

AND S.SaleID=SIS.SaleID

AND SIS.ProductID=I.ProductID

AND C.CustomerID BETWEEN 80 AND 89

ORDER BY CONCAT(C.FirstName,C.LastName), S.Date DESC;

	Customer Name	SaleID	Date	ProductName	Quantity	Price	Total
1	AdaYule	2705	2019-05-01	Blanket	5	400	2000
2	AdaYule	2707	2019-04-05	Watch	3	338	1014
3	AdaYule	2704	2019-03-08	Toys	1	399	399
4	AdaYule	2709	2019-01-05	Jwellery	4	550	2200
5	CaroleCyril	2703	2019-01-15	Dress	4	120	480
6	MagdaWater	2700	2019-05-15	Hangbag	2	525	1050
7	MaryWhite	2708	2019-05-20	Shirts	2	360	720
8	MaryWhite	2702	2019-04-18	Shoes	1	250	250
9	SallieaAsh	2706	2019-05-05	Jackets	1	425	425
10	SallieaAsh	2701	2019-05-02	Jeans	1	699	699

/*6*/ /*5 table Query*/

/*For each Store from the manager with the ID's 4400,4600, or 4700, list the ManagerID, StoreID, StoreName, StoreCity, ProductName, SaleID, individual total (Price*Quantity). Use the column alias 'Total', CompanyID, CompanyName. Put the list in descending order of StoreID*/

SELECT SAT.ManagerID, St.StoreID, St.StoreName, ST.City, I.ProductName, SI.SaleID, SI.Quantity, SI.Price, (SI.Price*SI.Quantity) AS 'Total', M.CompanyID, M.CompanyName

FROM SaleAssociate_T AS SAT, Store_T AS St, Inventory_T AS I, Store_Inventory_Sale_T AS SI, Manufacture_T M

WHERE SAT.StoreID = St.StoreID

AND St.StoreID = SI.StoreID

AND SI.ProductID = I.ProductID

AND I.CompanyID = M.CompanyID

AND SAT.ManagerID IN (4400,4600,4700)

ORDER BY StoreID DESC;

	ManagerID	StoreID	StoreName	City	ProductName	SaleID	Quantity	Price	Total	CompanyID	CompanyName
1	4600	1400	High Fashion	Clayton	Blanket	2705	5	400	2000	4	VANS
2	4600	1400	High Fashion	Clayton	Blanket	2705	5	400	2000	4	VANS
3	4600	1400	High Fashion	Clayton	Blanket	2705	5	400	2000	4	VANS
4	4700	1300	High Fashion	Richmond	Toys	2704	1	399	399	10	MAVI JEANS
5	4700	1300	High Fashion	Richmond	Toys	2704	1	399	399	10	MAVI JEANS
6	4400	900	High Fashion	Buffalo	Hangbag	2700	2	525	1050	8	MICHAEL KORS
7	4400	900	High Fashion	Buffalo	Hangbag	2700	2	525	1050	8	MICHAEL KORS
8	4400	900	High Fashion	Buffalo	Hangbag	2700	2	525	1050	8	MICHAEL KORS

Sub Queries: Subqueries are the queries which has a query within a query.

It is also referred as a Nested Queries. One example of subqueries with the join table.

/*List CompanyID, CompanyName, InventoryPrice, ProductID, ProductName, ProdDesc for all the products that have an InventoryPrice less than Sum of all the InventoryPrice*/

SELECT I.CompanyID, M.CompanyName,I.InventoryPrice, I.ProductID, I.ProductName, I.ProdDesc, C.CategoryID, C.CatDesc

FROM Inventory_T AS I, Category_T AS C,Manufacture_T AS M

WHERE InventoryPrice <

(SELECT SUM(InventoryPrice) AS 'Total'

FROM Inventory_T

Where CategoryID IN (1,2,3))

AND I.CategoryID=C.CategoryID

AND I.CompanyID=M.CompanyID

ORDER BY I.CompanyID, I.InventoryPrice DESC;

	CompanyID	CompanyName	InventoryPrice	ProductID	ProductName	ProdDesc	CategoryID	CatDesc
1	1	ZARA	500	10	Jwellery	Accesories	2	Women
2	1	ZARA	110	4	Dress	Party Wear	2	Women
3	2	ADIDAS	150	8	Watch	Accessories	1	Men
4	3	NIKE	250	7	Jackets	Winter Wear	1	Men
5	4	VANS	70	6	Blanket	Home Items	3	Children
6	5	CALVIN KLEIN	40	2	Jeans	Cloths	1	Men
7	7	OLD NAVY	45	9	Shirts	Cloth	1	Men
8	8	MICHAEL KORS	50	1	Hangbag	Accesories	2	Women
9	9	SKECHERS	80	3	Shoes	Footwear	1	Men
10	10	MAVI JEANS	100	5	Toys	Play	3	Children

Appendix

Assumptions:

- Store should have one or more inventory.
- Inventory should have one or more inventory.
- Manufacturer supply at least one product.
- Customers may not purchase any products.
- The product may not have any customer purchase.
- Each manager should supervise at least one sales associate.
- Each costumer may receive none coupon; each may be received by no costumer.
- Store may not receive any returns.
- Each return only at one store.
- Every store has its own inventory. At the same time, different stocks correspond to different storefronts.
- There are different products in the inventory, and each product is stored in a different warehouse.
- Manufacturers can produce multiple products, but one product can only be produced by one manufacturer.
- A sales assistant can only serve one store, but a store can have multiple assistants. At the same time, one manager can supervise multiple assistants, one assistant is only supervised by one manager.
- A sales assistant can help multiple customers, and one customer can only be served by one assistant.
- Customers can buy a lot of products, and products can also be purchased by many customers.
- Customers receive a lot of discounts every month, and the company also publishes many discounts to customers every month.
- The customer can return multiple items, and one item returned at that time corresponds to only one customer. Whether it is a receipt or not.
- Store may have more than one product inventory.
- Product Inventory may at least one store or more than one store.
- Manufacturer may not supply any inventory.
- Customer may not make any sales.
- Sale may be made by one and only one customer.
- Sale may be made at least one store.
- Store may have at least one sales associate.
- Sales associates may not help any customer, if customer don't need help.
- Manager may supervise at least one sales associate.
- Customer may get at least one coupon.
- For return with receipt there may be at least one sale or at least one item to be purchased.
- For return without receipt at least one or many items may purchase by one or many customer.
- Each store has at least one inventory. One inventory only supports one store.

- Manufacturers can supply multiple products while a certain product can only be supplied by one manufacturer.
- Customers can buy different products and one type of products can be bought by different customers.
- Customers can have many discounts during shopping and many discounts can be open to different customers.
- Store can return different items, but one item can be only accepted by one store.