**NAME**: James Hundley

By entering my name above I certify that I completed this project without any assistance from any other person. Any honor code violation on this project will result in failure of the course.   
  
**Connect to the Virtual Computing Lab and then connect to the database assigned to you.   
DE SECTION: “zzDE”[first 4 letters of last name]   
FTF SECTION: “zz1”[first 4 letters of last name]**

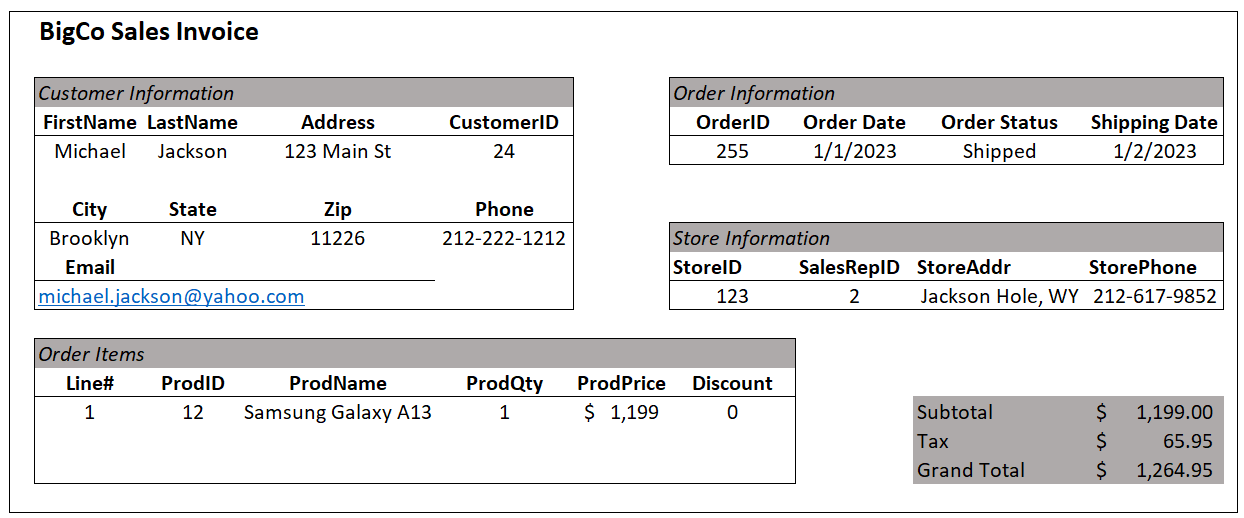
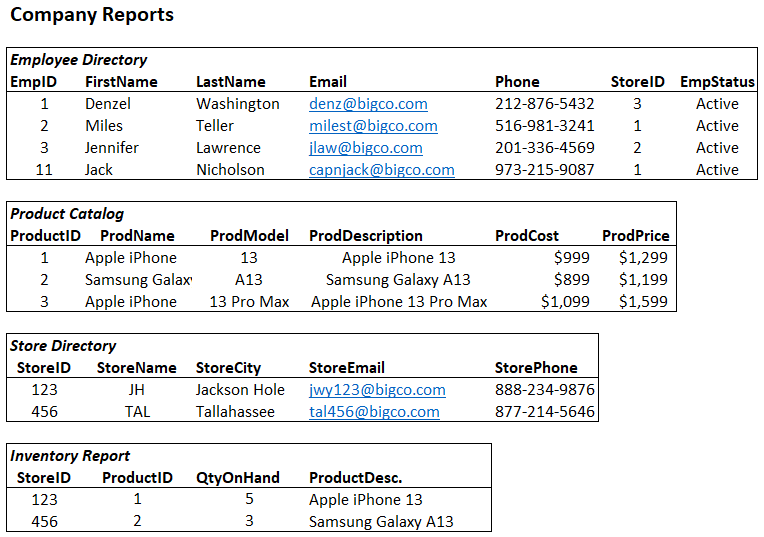
All code must be properly indented as seen in the textbook, solutions to homework and assignments.

The project is worth a significant percentage of your grade (check syllabus for details). Therefore, please do not leave any questions blank. Partial credit may be available if you know some of the code or some of the key concepts for each question. If your code does not run properly, you can send it to me for review. I will do my best to review it and give feedback/direction if there is adequate time before the deadline.   
 **Overview**

In this project, you are required to create a database diagram and subsequently a database, based on an invoice and some reports from BigCo Inc. You will then have to make changes to that database including changing data values, dropping and adding columns and changing a table name. There are 5 parts to this project:

Part 1: Normalization (10%)  
Part 2: Databse Design (25%)  
Part 3: Database Development using SQL (40%)  
Part 4: Database Data Modification using SQL (5%)  
Part 5: Database Redesign (20%)

**Part 1: Normalization (10%)**Consider the following sales invoice and company reports:

****   
Using the invoice and report information:

1. Create a set of normalized entities that are in 3rd Normal Form or higher that will capture the information found in the BigCo invoice and reports. Any areas that are in gray do not need to be added to the database. For example, the term “customer information” can be skipped, but the fields found in the “Customer Information” section need to be included in your tables. Sales tax and total amount of the invoice can be skipped.   
   **HINT:** You should have 7 entities: CUSTOMER, STORE, PRODUCT, INVENTORY, EMPLOYEE, CUST\_ORDER, ORDER\_ITEM.

**RECOMMENDATION:** Strive to reduce redundancy in this stage. ProdDesc might appear in multiple reports and the invoice. However, normalization is meant to reduce redundancy and update/delete anomalies that result from redundancies. I recommend carefully typing each attribute in a spreadsheet or on paper by hand to ensure you capture all the fields on the report/invoice. Then carefully group the attributes into entities. Then eliminate duplicate attributes and add your foreign keys. This is the foundation of your database, so give it the time and attention it deserves. Time invested here will save time later.   
  
Using relational notation, list your normalized entities and attribtes. Be sure to underline identifiers and italicise foreign keys. For example, a VEHICLE entity with VIN as the identifier and MfgName as a foreign key would be listed as: VEHICLE(VIN, *MfgName*, Model, Version, Year, Transmission, etc..) (20 pts)

**List your normalized entities in relational notation here:**

Customer (CustomerID, CustFName, CustLName, Address, City, State, Zip, Phone)

STORE (StoreID, StoreName, StoreAddr, StorePhone, StoreCity)

PRODUCT (ProdID, ProdName, ProdModel, ProdDesc, ProdCost, ProdPrice)

EMPLOYEE (EmpID, EmpFName, EmpLName, EmpEmail, EmpPhone, *StoreID,* EmpStatus)

INVENTORY (*StoreID*, *ProdID,* QtyOnHand)

CUST\_ORDER ( OrderID, OrderDate, OrderStatus, ShippingDate, *SalesRepID*, *CustomerID*)

ORDER\_ITEM (Line#,OrderID,ProdID, ProdQty, Discount)

**Part 2: Databse Design (25%)**

Create a logical, database diagram, using Visio or LucidChart, from the tables you listed above. (You do not need to include a conceptual diagram.) When creating relationships between the entities/tables, use the business rules below to determine the minimum and maximum cardinality:

1. A customer can place many orders, and an order can have only and only one customer.
2. An order must have at least one line item, and can have many lines. Each line must relate to one and only one order.
3. An order can be assigned to only one store, and a store can have many orders.
4. An order line can include only one product per line (though quantiy can be more than one), and a product can be included in many different orders.
5. An employee is assigned to only one store. A store must have at least one employee and can have many employees assigned to it.
6. A product can be held in inventory at many stores and a store can hold many products in inventory.

Be sure to identify primary keys and foreign keys. Be sure to place your foreign keys to create relationships that satisfy the above business rules.

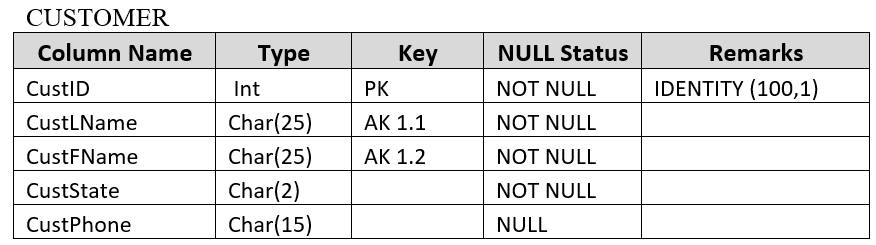
**Paste your daatabase diagram here:**

Diagram

Description automatically generated

**Part 3: Database Development using SQL (40%)**

Write SQL statements to create each of the tables you listed. Be sure to write your code using good form as shown in the text. **Use an auto-incremented surrogate key in at least two tables.** NOTE: You should not use surrogate keys in any associative or intersection tables.

RECOMMENDATION: I recommend you create a spreadsheet or other document to keep track of all the entities, attributes, datatypes and null statuses. Something like this will be helpful (just an example):

BIG HINT: The order in which you create your tables matters a lot! Foreign key constraints are not able to be created if the table in which that foreign key is a primary key has not yet been created. Create the tables that have no foreign keys first. Then only create tables that have foreign keys from already created tables. If you are stuck, you can add the foreign key constraints after all tables have been created.

**Paste your CREATE TABLE Statements here:**

**CREATE TABLE CUSTOMER (**

**CustomerID INT IDENTITY (1,1) NOT NULL,**

**CustFName VARCHAR(50) NOT NULL,**

**CustLName VARCHAR(50) NOT NULL,**

**CustAddr VARCHAR(255) NOT NULL,**

**City VARCHAR(50) NOT NULL,**

**CustState VARCHAR(50) NOT NULL,**

**Zip INT NOT NULL,**

**Phone VARCHAR(20) NOT NULL**

**CONSTRAINT Customer CustomerID\_PK PRIMARY KEY (CustomerID)**

**);**

Text

Description automatically generated

CREATE TABLE STORE (

StoreID INT NOT NULL,

StoreName VARCHAR(50) NOT NULL,

StoreAddr VARCHAR(255) NOT NULL,

StorePhone VARCHAR(50) NOT NULL,

StoreCity VARCHAR(50) NOT NULL

CONSTRAINT StoreID\_PK PRIMARY KEY (StoreID)

);

Text

Description automatically generated

CREATE TABLE PRODUCT (

ProdID INT NOT NULL,

ProdName VARCHAR(50) NOT NULL,

ProdModel VARCHAR(50) NULL,

ProdDesc VARCHAR(200) NULL,

ProdCost MONEY NOT NULL,

ProdPrice MONEY NULL

CONSTRAINT ProdID\_PK PRIMARY KEY (ProdID)

);

Text

Description automatically generated

CREATE TABLE EMPLOYEE (

EmpID INT IDENTITY (1,1) NOT NULL,

EmpFName VARCHAR(50) NOT NULL,

EmpLName VARCHAR(50) NOT NULL,

EmpEmail VARCHAR(100) NULL,

EmpPhone VARCHAR(20) NOT NULL,

StoreID INT NOT NULL,

EmpStatus VARCHAR(10) NULL

CONSTRAINT EmpID\_PK PRIMARY KEY (EmpID),

CONSTRAINT StoreID\_FK FOREIGN KEY (StoreID) REFERENCES STORE (StoreID)

);

Text

Description automatically generated

CREATE TABLE INVENTORY (

StoreID INT NOT NULL,

ProdID INT NOT NULL,

QtyOnHand INT NULL

CONSTRAINT StoreID\_ProdID\_PK PRIMARY KEY (StoreID,ProdID),

CONSTRAINT StoreID\_INV\_FK FOREIGN KEY (StoreID) REFERENCES STORE (STOREID),

CONSTRAINT ProdID\_FK FOREIGN KEY (ProdID) REFERENCES PRODUCT (ProdID)

);

Text, chat or text message

Description automatically generated

CREATE TABLE CUST\_ORDER (

OrderID INT NOT NULL,

OrderDate DATE NOT NULL,

OrderStatus VARCHAR(20) NULL,

ShippingDate DATE NULL,

SalesRepID INT NOT NULL,

CustomerID INT NOT NULL

StoreID INT NOT NULL

CONSTRAINT OrderID\_PK PRIMARY KEY (OrderID),

CONSTRAINT SalesRepID\_FK FOREIGN KEY (SalesRepID) REFERENCES EMPLOYEE (EmpID),

CONSTRAINT CustomerID\_FK FOREIGN KEY (CustomerID) REFERENCES CUSTOMER (CustomerID),

CONSTRAINT FK\_CUST\_ORDER\_StoreID FOREIGN KEY (StoreID) REFERENCES STORE (StoreID)

);

Text, chat or text message

Description automatically generated

CREATE TABLE ORDER\_ITEM (

Line# INT NOT NULL,

OrderID INT NOT NULL,

ProdID INT NOT NULL,

ProdQty INT NULL,

ProdDiscount DECIMAL NULL

CONSTRAINT Line#\_OrderID\_PK PRIMARY KEY (Line#,OrderID),

CONSTRAINT OrderID\_Order\_FK FOREIGN KEY (OrderID) REFERENCES CUST\_ORDER (OrderID),

CONSTRAINT ProdID\_Order\_FK FOREIGN KEY (ProdID) REFERENCES PRODUCT (ProdID)

);

Text

Description automatically generated

1. INSERT Data: Add at least two rows to each table. Similar to creating tables, the order in which you insert data matters and for the same reasons.

**Paste your insert statements here:**

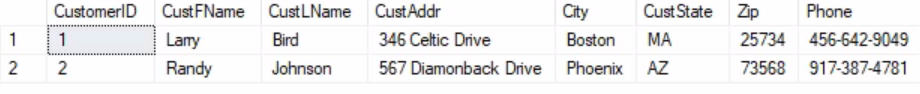
**INSERT INTO CUSTOMER**

**(CustFName,CustLName,CustAddr,City,CustState,Zip,Phone)**

**VALUES**

**('Larry','Bird','346 Celtic Drive','Boston','MA','25734','456-642-9049'),**

**('Randy','Johnson','567 Diamonback Drive','Phoenix','AZ','73568','917-387-4781');**



INSERT INTO STORE

(StoreID,StoreName,StoreAddr,StorePhone,StoreCity)

VALUES

('5','Phone Hub','123 Clear Blvd','302-141-0912','Charlotte'),

('10','Phone Shack','789 Shack Drive','709-482-1793','Myrtle Beach');

Graphical user interface, text, application

Description automatically generated

INSERT INTO PRODUCT

(ProdID,ProdName,ProdModel,ProdDesc,ProdCost,ProdPrice)

VALUES

('50','Samsung Galaxy','S23',NULL,$1000,NULL),

('60','IPhone','23',NULL,$1200,NULL);

Text

Description automatically generated with medium confidence

INSERT INTO EMPLOYEE

(EmpFName,EmpLName,EmpEmail,EmpPhone,StoreID,EmpStatus)

VALUES

('Jerry','Garcia','GarciaJ@gmail.com','123-456-7890','5','Active'),

('David','Gilmoure','GilmoureD@gmail.com','234-567-8901','10','Active');

Graphical user interface, text, application

Description automatically generated

INSERT INTO INVENTORY

(StoreID,ProdID,QtyOnHand)

VALUES

('5','50','27'),

('10','60','54');

Table

Description automatically generated

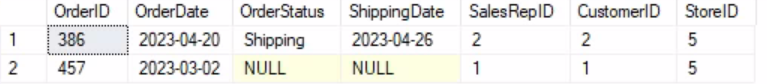
INSERT INTO CUST\_ORDER

(OrderID,OrderDate,OrderStatus,ShippingDate,SalesRepID,CustomerID,StoreID)

VALUES

('457','03-02-2023',NULL,NULL,'1','1',’5’)

('386','04-20-2023','Shipping','04-26-2023','2','2',’5’)



INSERT INTO ORDER\_ITEM

(Line#,OrderID,ProdID,ProdQty,ProdDiscount)

VALUES

('1','457','50','1',NULL),

('1','386','60','1',NULL);

Table

Description automatically generated with medium confidence

1. Create a VIEW:

Write a SQL statement to create a view that recreates the information found on the Invoice, as follows:

*CustID, FirstName, LastName, OrderDate, StoreName, TotalOrderPrice*

**Paste your create view statement here:**

**CREATE VIEW Customer\_View AS**

**SELECT c.CustomerID,c.CustFName AS FirstName,c.CustLName AS LastName,o.OrderDate,s.StoreName,SUM (p.ProdCost) AS TotalOrderPrice**

**FROM CUSTOMER c**

**JOIN CUST\_ORDER o**

**ON c.CustomerID=o.CustomerID**

**JOIN STORE s**

**ON o.StoreID=s.StoreID**

**JOIN ORDER\_ITEM oi**

**ON o.OrderID=oi.OrderID**

**JOIN PRODUCT p**

**ON oi.ProdID=p.ProdID**

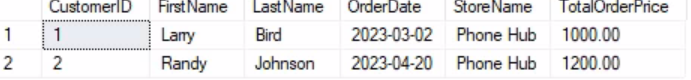
**GROUP BY c.CustomerID,c.CustFName,c.CustLName,o.OrderDate,s.StoreName;**

1. Use a SELECT statement to show the results of your view.

**Code:**

**Results: SELECT CustomerID,FirstName, LastName, OrderDate, StoreName, TotalOrderPrice**

**FROM Customer\_View**



**Part 4: Database Data Modification using SQL (5%)**

Write SQL statements to complete the following steps.

1. UPDATE the value of at least one record using an update statement. NOTE: This will require THREE statements and results.
   1. First, write a SELECT statement showing the “original” values you will be updating and paste results in “Results before UPDATE”.
   2. Write your UPDATE statement and copy the code into the space labeled UPDATE statement.
   3. Then write a SELECT statement to show the new value(s).

**Results before UPDATE:**

**SELECT ProdModel**

**FROM PRODUCT**

A picture containing calendar

Description automatically generated

**UPDATE Code:**

**UPDATE PRODUCT**

**SET ProdModel='S24'**

**WHERE ProdID=50;**

**Results after UPDATE:**

Calendar

Description automatically generated with low confidence

**Part 5: Database Redesign (20%)**

Write SQL statements to complete the following steps.

1. Add a DEFAULT constraint to EMPLOYEE.EmpStatus (or whatever similar name you gave the attribute listed in the Employee Directory). The default value should be “Active”.   
   **Code:**

ALTER TABLE EMPLOYEE

ADD CONSTRAINT EmpStatus\_DEFAULT DEFAULT 'Active' FOR EmpStatus;

1. In this step you will be **replacing** the INVENTORY table with a new table that contains the same records as the INVENTORY table.
   1. Use a CREATE TABLE statement to create a NEW inventory table named PROD\_INV. Before running the CREATE TABLE statement, add another attribute called “QtyOnOrder”. It should be an INT datatype, allowing NULL values with a default constraint making the default value equal to 0.

**Code:**

CREATE TABLE PROD\_INV (

StoreID INT NOT NULL,

ProdID INT NOT NULL,

QtyOnHand INT NULL,

QtyOnOrder INT DEFAULT '0' NULL

CONSTRAINT PK\_ProdID\_StoreID PRIMARY KEY (StoreID,ProdID),

CONSTRAINT FK\_ProdID FOREIGN KEY (ProdID) REFERENCES PRODUCT (ProdID),

CONSTRAINT FK\_StoreID FOREIGN KEY (StoreID) REFERENCES STORE (StoreID),

);

1. Write a SQL statement to copy the existing rows from INVENTORY to PROD\_INV. You’ll need to accommodate for the new attribute you created in the previous step.

**Code:**  
INSERT INTO PROD\_INV

(StoreID,ProdID,QtyOnHand)

VALUES

('5','50','27'),

('10','60','54')

;

Table

Description automatically generated

1. Drop the INVENTORY table so there will be only be a PROD\_INV table and no INVENTORY table.

**Code:**

**DROP TABLE INVENTORY;**

**CONGRATULATIONS! You are DONE!**