INF 214

Practical 01 – Overview

Creating Databases & Tables

Important Notes:

1. Bring the following to the practical session:
    a. A flash disk with all the work that you did in the previous practical session. The flash disk should not have any work for the previous session
    b. The Practical Textbook and/or the lecture notes for the exercise class in which the exercise was explained

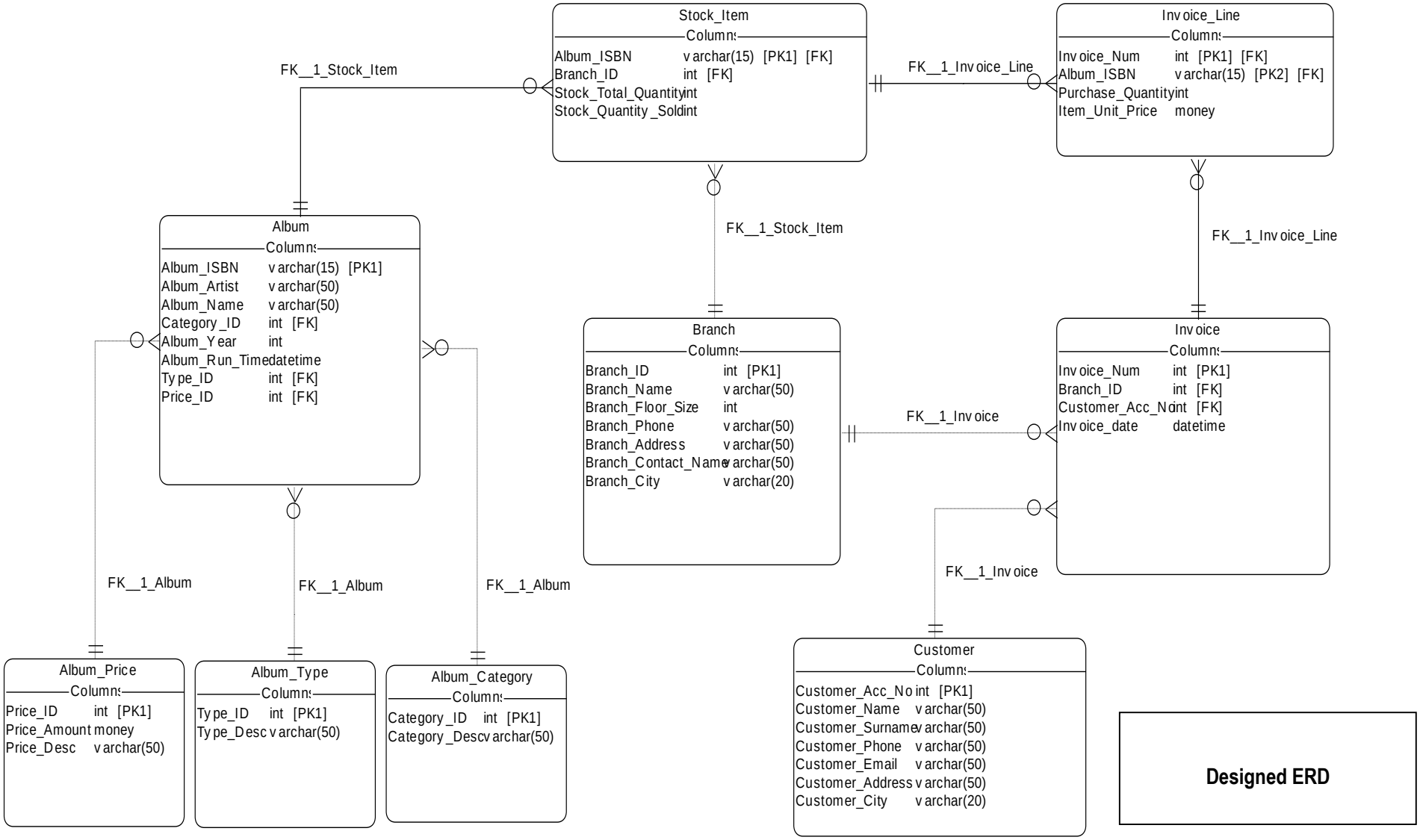
2. SQL scripts are what is marked
    a. Scripts which run and perform what they are supposed to do get full marks
    b. Scripts which do not run will be marked on their own merit

General Instructions:

Make use of class notes and examples to complete practical 1. The following few pages will provide you with an overview of how to create a database and tables.

After the general overview, please refer to the general instructions doe practical assignment 1, as found on 8.

Entity Relationship Diagram - Table Create Example



No instance of a table	DROP Database MusicologyWarehouse GO CREATE Database MusicologyWarehouse GO USE MusicologyWarehouse GO	The creation scrip creates a space in memory by means of the Database Engine that would allow for a database to be loaded into memory. You first Drop a Database before you Create it and Load the data. The reason is to ensure that, if the database is already loaded and active, then it clears and prepares the memory namespace for the loading of the database. This is usually done only once. In an active database, the current instance is saved before it is reloaded otherwise the active and "new data" would be lost.
Branch Columns Branch_ID int [PK1] Branch_Name varchar(50) Branch_Floor_Size int Branch_Phone varchar(50) Branch_Address varchar(50) Branch_Contact_Name varchar(50) Branch_City varchar(20)	CREATE TABLE Branch (Branch_ID int primary key, Branch_Name varchar(50), Branch_Floor_Size int, Branch_Phone varchar(50), Branch_Address varchar(50), Branch_Contact_Name varchar(50), Branch_City varchar(20))	The Create Table Script is direct and to the point. It is sequenced based on how you would like to load data based on the data type associated with the Attribute. For Example Branch_ID is set as an int (short for integer) and it is then set as the primary key. In the Stock Table, this would then be referenced as a Foreign Key by means of "Branch_ID int references Branch(Branch_ID)" – this links the two table together by referencing the Branch_ID as a Primary Key in one table and then a Foreign Key, or reference in another table. If you for example state "Branch_Name varchar(50)," it means that the attribute "Branch_Name" is set as a variable character, and the attribute is only allowed to contain sets of data that is no longer than 50 characters. The "(50)" stipulates the maximum length that can be loaded into a particular attribute.

INSERT into Branch VALUES (1, 'Brooklyn', 250, '012-4603151', '79 Fesen Str', 'Jonathan', 'Johannesburg') INSERT into Branch VALUES (2, 'Menlyn', 250, '012-3465678', '100 Lois Ave', 'Sam', 'Pretoria') INSERT into Branch VALUES (3, 'Lynnwood', 75, '012-3616589', '2 Kings Highway', 'Phil Van DeVenter', 'Pretoria') INSERT into Branch VALUES (11, 'Northridge', 125, '011-2547896', '58 Hans Strijdom drive', 'Kirstin Krauss', 'Polokwane')	The loading script is very specific. It is directly linked to the sequence that the attributes of the entity / table is created. For example, the first attribute that is created is the Primary Key, Branch_ID. As such, the first attribute that need to be loaded in an INSERT Into Statement, would be the Primary key, In this case "1".The following attribute that would then be loaded with data would then be based on the sequence of the "Create Table Branch" • Branch_Name varchar(50), ➔ Brooklyn • Branch_Floor_Size int, ➔ 250 • Branch_Phone varchar(50), ➔ 012-4603151 • Branch_Address varchar(50), ➔ 79 Fesen Str • Branch_Contact_Name varchar(50), ➔ Jonathan • Branch_City varchar(20) ➔ Johannesburg Insert Into Sequence should be exactly the same as the Table Create Sequence. That may not differ. If they do, data will be loaded into inappropriate field.

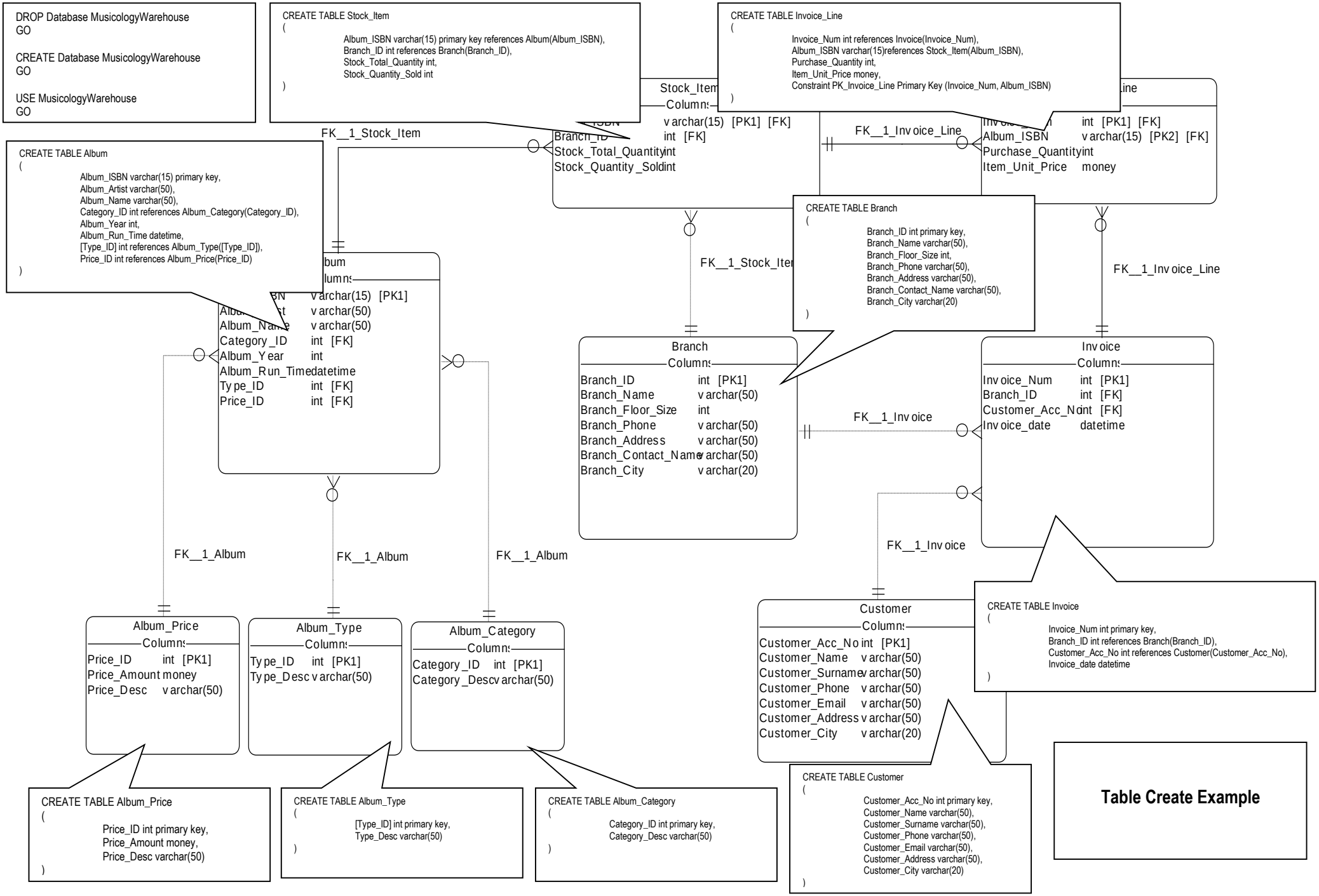


Table Create Example

```
DROP Database MusicologyWarehouse
GO

CREATE Database MusicologyWarehouse
GO

USE MusicologyWarehouse
GO
```

Create Database

&

Database Namespace

```
CREATE TABLE Branch
(
        Branch_ID int primary key,
        Branch_Name varchar(50),
        Branch_Floor_Size int,
        Branch_Phone varchar(50),
        Branch_Address varchar(50),
        Branch_Contact_Name varchar(50),
        Branch_City varchar(20)
)
```

Create Table

7 Attributes

```
CREATE TABLE Customer
(
        Customer_Acc_No int primary key,
        Customer_Name varchar(50),
        Customer_Surname varchar(50),
        Customer_Phone varchar(50),
        Customer_Email varchar(50),
        Customer_Address varchar(50),
        Customer_City varchar(20)
)
```

Create Table

7 Attributes

```
CREATE TABLE Invoice
(
        Invoice_Num int primary key,
        Branch_ID int references Branch(Branch_ID),
        Customer_Acc_No int references Customer(Customer_Acc_No),
        Invoice_date datetime
)
```

Create Table

4 Attributes

```
CREATE TABLE Album_Type
(
        [Type_ID] int primary key,
        Type_Desc varchar(50)
)
```

Create Table

2 Attributes

```
CREATE TABLE Album_Price
(
        Price_ID int primary key,
        Price_Amount money,
        Price_Desc varchar(50)
)
```

Create Table

3 Attributes

```
CREATE TABLE Album_Category
(
        Category_ID int primary key,
        Category_Desc varchar(50)
)
```

Create Table

2 Attributes

```
CREATE TABLE Album
(
        Album_ISBN varchar(15) primary key,
        Album_Artist varchar(50),
        Album_Name varchar(50),
        Category_ID int references Album_Category(Category_ID),
        Album_Year int,
        Album_Run_Time datetime,
        [Type_ID] int references Album_Type([Type_ID]),
```

Create Table

8 Attributes

```
        Price_ID int references Album_Price(Price_ID)
)

CREATE TABLE Stock_Item
(
        Album_ISBN varchar(15) primary key references Album(Album_ISBN),
        Branch_ID int references Branch(Branch_ID),
        Stock_Total_Quantity int,
        Stock_Quantity_Sold int
)

CREATE TABLE Invoice_Line
(
        Invoice_Num int references Invoice(Invoice_Num),
        Album_ISBN varchar(15)references Stock_Item(Album_ISBN),
        Purchase_Quantity int,
        Item_Unit_Price money,
        Constraint PK_Invoice_Line Primary Key (Invoice_Num, Album_ISBN)
)
```

Create Table

4 Attributes

Create Table

5 Attributes

INFORMATIKA • DEPARTMENT
DEPARTEMENT • INFORMATICS
Universiteit van Pretoria
University of Pretoria

Practical 01 – Exercise 01

Creating Databases & Tables (10 marks)

Important Notes:

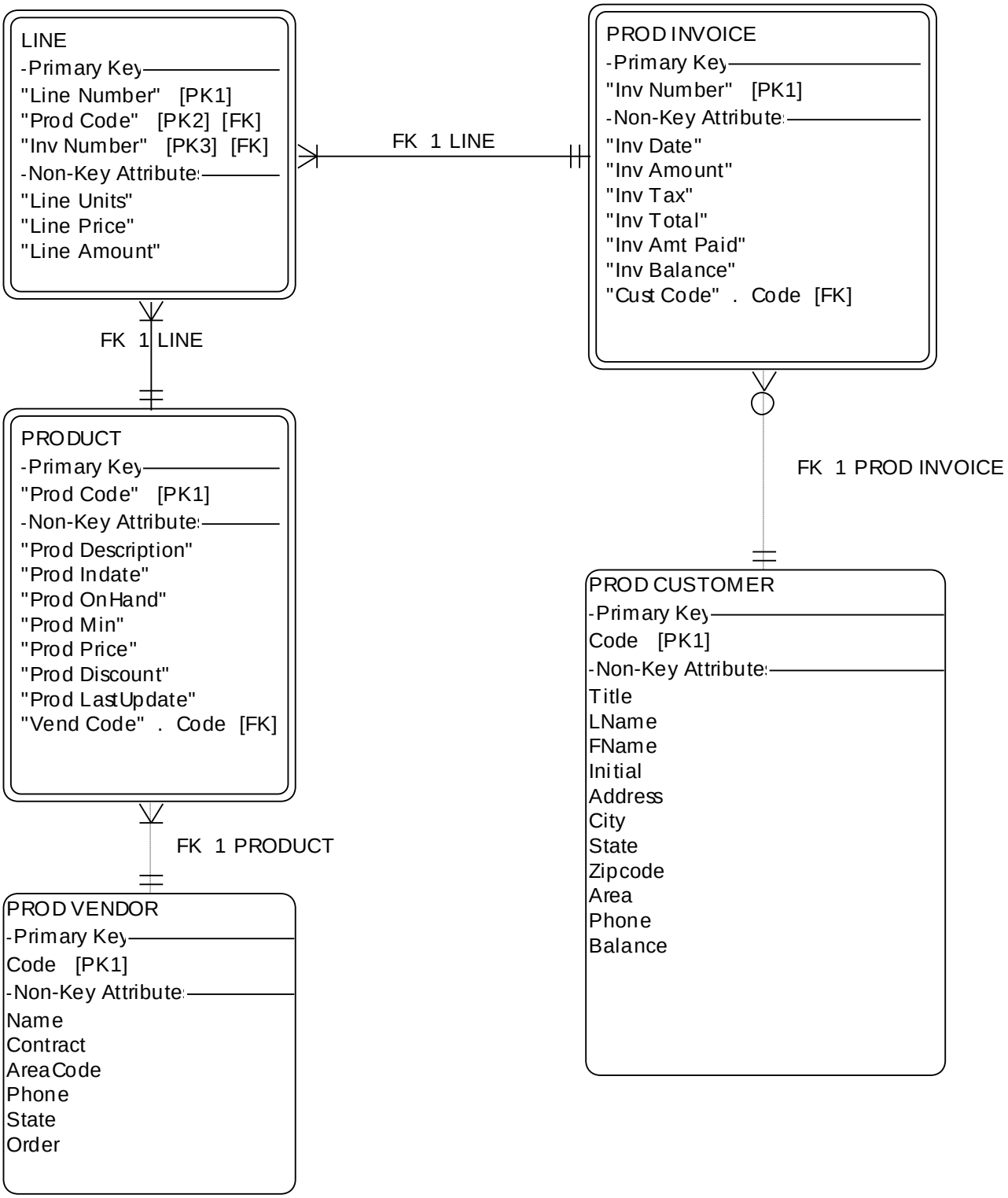
3. Bring the following to the practical session:
    a. A flash disk with all the work that you did in the previous practical session. The flash disk should not have any work for the previous session
    b. The Practical Textbook and/or the lecture notes for the exercise class in which the exercise was explained

4. SQL scripts are what is marked
    a. Scripts which run and perform what they are supposed to do get full marks
    b. Scripts which do not run will be marked on their own merit

Questions:

In this exercise, you will create Product Sales database based on the ERD provided:

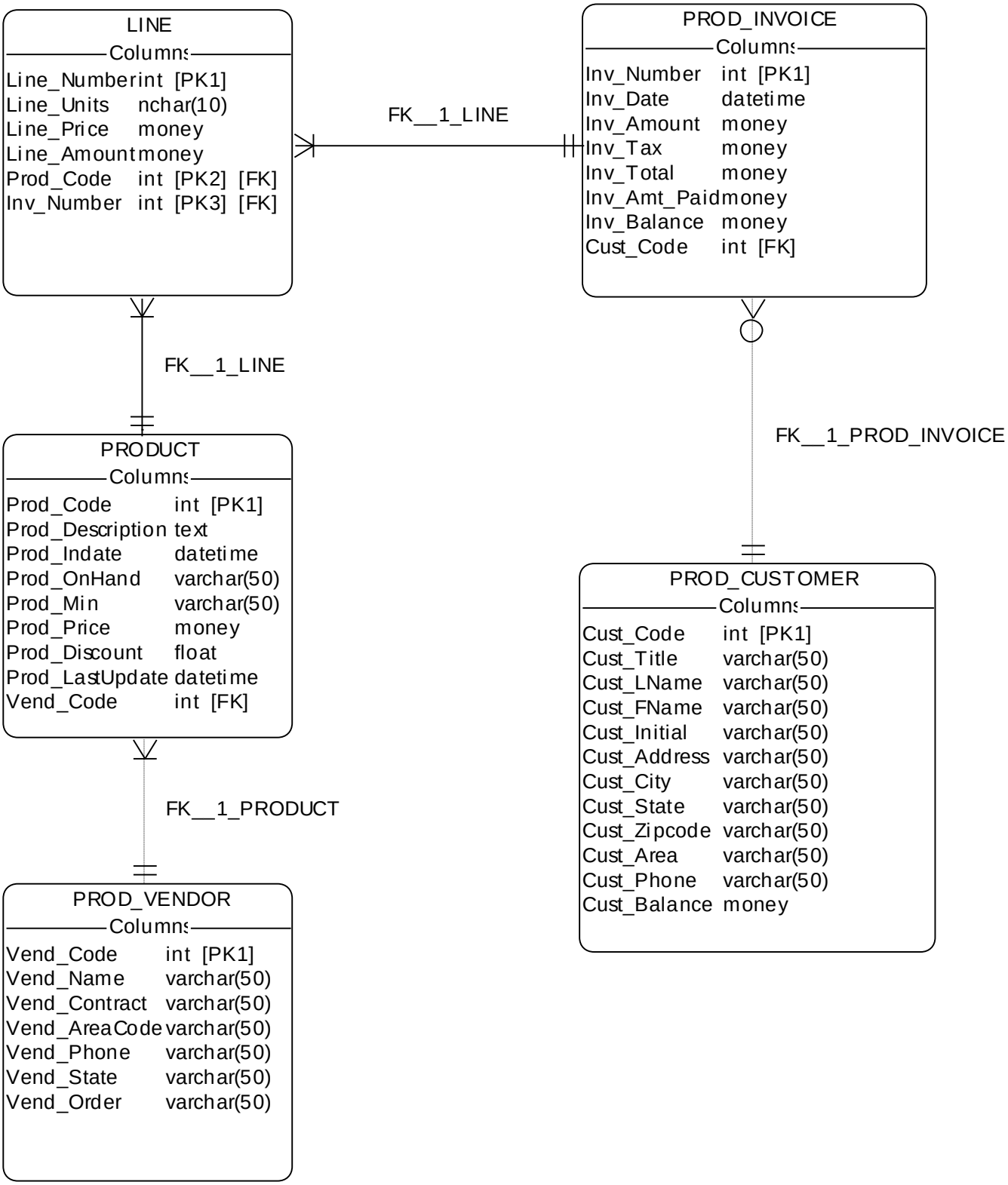
Task	Mark
1. Create the Product Sales database	2
2. Create the corresponding 5 Tables as shown in the ERD with the appropriate constraints and attributes **a)** Prod_Customer **b)** Prod_Invoice **c)** Line **d)** Product **e)** Prod_Vendor	5
3. Create the corresponding Database Diagram – should look similar to the ERD provided (Diagram 3 – SQL Diagram View)	3



)

Logical Design View – The model provides you with an indication of the Entities, Attributes, Primary and Foreign Keys as well as the Entity Relationships.

This is a pure logical design view.



Physical Design View – The model provides you with an indication of the Entities, Attributes, Attribute Data Types, Data Lengths, Primary and Foreign Keys as well as the Entity Relationships.

This view combines details from the Conceptual Model and the internal model to create a physical model.

PRODUCT

Column Name	Data Type	Allow Nulls
Prod_Code	int	☐
Prod_Description	text	☑
Prod_Indate	datetime	☑
Prod_OnHand	varchar(50)	☑
Prod_Min	varchar(50)	☑
Prod_Price	money	☐
Prod_Discount	float	☑
Prod_LastUpdate	datetime	☑
Vend_Code	int	☑
		☐

LINE

Column Name	Data Type	Allow Nulls
Line_Number	int	☐
Line_Units	nchar(10)	☐
Line_Price	money	☐
Line_Amount	money	☐
Prod_Code	int	☑
Inv_Number	int	☑
		☐

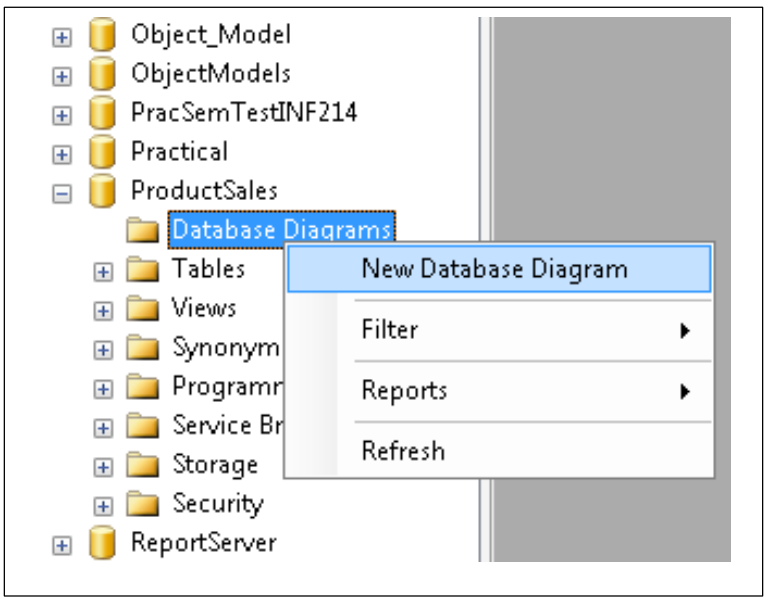
PROD_VENDOR

Column Name	Data Type	Allow Nulls
Vend_Code	int	☐
Vend_Name	varchar(50)	☐
Vend_Contract	varchar(50)	☐
Vend_AreaCode	varchar(50)	☐
Vend_Phone	varchar(50)	☐
Vend_State	varchar(50)	☐
Vend_Order	varchar(50)	☐
		☐

PROD_INVOICE

Column Name	Data Type	Allow Nulls
Inv_Number	int	☐
Inv_Date	datetime	☐
Inv_Amount	money	☐
Inv_Tax	money	☐
Inv_Total	money	☐
Inv_Amt_Paid	money	☐
Inv_Balance	money	☐
Cust_Code	int	☑
		☐

SQL Database Diagram – The SQL database diagram as found within the SQL Management Studio



PROD_CUSTOMER

Column Name	Data Type	Allow Nulls
Cust_Code	int	☐
Cust_Title	varchar(50)	☐
Cust_LName	varchar(50)	☐
Cust_FName	varchar(50)	☐
Cust_Initial	varchar(50)	☐
Cust_Address	varchar(50)	☐
Cust_City	varchar(50)	☐
Cust_State	varchar(50)	☐
Cust_Zipcode	varchar(50)	☐
Cust_Area	varchar(50)	☐
Cust_Phone	varchar(50)	☐
Cust_Balance	money	☐
		☐