**Part 1 (40 pts total): - Tim Capehart**

The following diagram illustrates the logical design for a Contacts database. Study the design and then build the database using DDL commands and using appropriate data types that fit the data. Please adhere to the following stipulations:

1. All Id fields should have the IDENTITY property
2. Add constraints for primary and foreign keys where necessary

A screenshot of a video game

Description automatically generated

**After your table is built, please insert the data in the following table.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| First Name | Last Name | Line1 | Line2 | Town | State | Zip | Country | Continent |
| Jiaying | Wong | 8 XUJIAZHAI | XINZHONG XIANG, TIANTAI XIAN | HUAQIAOCUN | ZHEJIANG | 266033 | China | Asia |
| Jack | Zarneth | 373 Church Street | | Colloyn | PRETORIA | 83 | Gauteng | Africa |
| Cecilia | Chapman | 711-2880 Nulla St | | Mankato | Mississippi | 96522 | United States | North America |
| Carlo Francis | Xavier | Escuela Rural 45 | | San Clemente | Buenos Aires Province | X5187XAB | Argentina | South America |
| Walter C | Brown | 49 Featherstone Street | | London | Middlesex | ECY1 8SY | United Kingdom | Europe |

**Deliverable:**

1. **All code needed to create the database and add the data (25 pts)**
2. **SQL statement that will combine all the data back and produce a table result similar to the data table given.(10 pts)**
3. **Screen shot of the output result of #2 (5 pts)**

USE master

CREATE DATABASE Contacts

GO

USE Contacts

CREATE TABLE Continent(

ContinentId int NOT NULL IDENTITY,

Name nvarchar(100) NOT NULL

CONSTRAINT PK\_Continent PRIMARY KEY (ContinentId)

);

CREATE TABLE Country(

CountryId int NOT NULL IDENTITY,

Name nvarchar(100) NOT NULL,

ContinentId int NOT NULL,

CONSTRAINT PK\_Country PRIMARY KEY (CountryId),

CONSTRAINT FK\_Country\_Continent FOREIGN KEY (ContinentId) REFERENCES Continent(ContinentId)

);

CREATE TABLE StateOrProvince(

StateOrProvinceId int NOT NULL IDENTITY,

Name nvarchar(100) NOT NULL,

CountryId int NOT NULL

CONSTRAINT PK\_StateOrProvince PRIMARY KEY (StateOrProvinceId),

CONSTRAINT FK\_StateOrProvince\_Country FOREIGN KEY (CountryID) REFERENCES Country(CountryID)

);

CREATE TABLE Town(

TownId int NOT NULL IDENTITY,

Name nvarchar(100) NOT NULL,

StateOrProvinceId int NOT NULL,

CONSTRAINT PK\_Town PRIMARY KEY (TownId),

CONSTRAINT FK\_Town\_StateOrProvince FOREIGN KEY (StateOrProvinceId) REFERENCES StateOrProvince(StateOrProvinceId)

);

CREATE TABLE Address(

AddressId int NOT NULL IDENTITY,

Line1 nvarchar(255) NOT NULL,

Line2 nvarchar(255) NOT NULL,

TownId int NOT NULL,

ZipCode int NOT NULL

CONSTRAINT PK\_Address PRIMARY KEY (AddressId),

CONSTRAINT FK\_Address\_Town FOREIGN KEY (TownId) REFERENCES Town(TownId)

);

ALTER TABLE Address

ALTER COLUMN Line2 nvarchar(255) NULL

ALTER TABLE Address

ALTER COLUMN ZipCode nvarchar(100)

CREATE TABLE Person(

PersonId int NOT NULL IDENTITY,

FirstName nvarchar(100) NOT NULL,

LastName nvarchar(100) NOT NULL,

AddressId int NOT NULL,

CONSTRAINT PK\_Person PRIMARY KEY (PersonId),

CONSTRAINT FK\_Person\_Address FOREIGN KEY (AddressId) REFERENCES Address(AddressId)

);

INSERT INTO Continent (Name) VALUES

('Asia'),

('Africa'),

('North America'),

('South America'),

('Europe');

SELECT \* from Continent

INSERT INTO Country (Name, ContinentId) VALUES

('China', 1),

('Gauteng', 2),

('United States', 3),

('Argentina', 4),

('United Kingdom', 5);

SELECT \* FROM Country

INSERT INTO StateOrProvince (Name, CountryId) VALUES

('ZHEJIANG', 1),

('PRETORIA', 2),

('Mississippi', 3),

('Buenos Aires Province', 4),

('Middlesex', 5);

SELECT \* FROM StateOrProvince

INSERT INTO Town (Name, StateOrProvinceId) VALUES

('HUAQIAOCUN', 1),

('Colloyn', 2),

('Mankato', 3),

('San Clemente', 4),

('London', 5);

SELECT \* FROM Town

INSERT INTO Address (Line1, Line2, TownId, ZipCode) VALUES

('8 XUJIAZHAI', 'XINZHONG XIANG, TIANTAI XIAN', 1, '266033'),

('373 Church Street', NULL, 2, '83'),

('711-2880 Nulla St', NULL, 3, '96522'),

('Escuela Rural 45', NULL, 4, 'X5187XAB'),

('49 Featherstone Street', NULL, 5, 'ECY18SY');

SELECT \* FROM Address

INSERT INTO Person (FirstName, LastName, AddressId) VALUES

('Jiaying', 'Wong', 1),

('Jack', 'Zarneth', 2),

('Cecilia', 'Chapman', 3),

('Carlo Francis', 'Xavier', 4),

('Walter C', 'Brown', 5);

SELECT p.FirstName, p.LastName, a.Line1, a.Line2, t.Name, s.Name, a.ZipCode, c.Name, cont.Name

FROM Continent cont JOIN Country c

ON cont.ContinentId = c.ContinentId

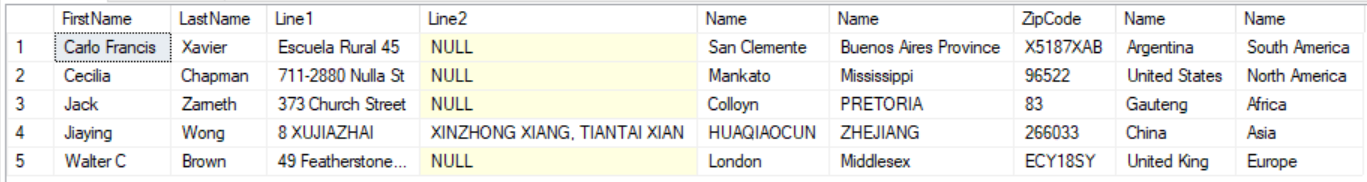
JOIN StateOrProvince s ON s.CountryId = c.CountryId

JOIN Town t ON t.StateOrProvinceId = s.StateOrProvinceId

JOIN Address a ON a.TownId = t.TownId

JOIN Person p ON p.AddressId = a.AddressId

GROUP BY p.FirstName, p.LastName, a.Line1, a.Line2, t.Name, s.Name, a.ZipCode, c.Name, cont.Name



**Part 2 (60 pts total)**

The diagram illustrates the conceptual design for a Sales/Order database for a Bicycle Store. You are being provided with the code to build the database and insert the data. The following pages describe each table in detail. On the last page, you will find reports and other queries that you are required to develop.

**For each question, provide the SQL commands and a screen shot of your output. Unless otherwise, specified, the order of the resuls can be arbitrary.**

A screenshot of a computer

Description automatically generated

As you can see from the diagram, the BikeStores sample database has two schemas sales and production, and these schemas have nine tables.

***Note: To reference a table, you must specify it in the Schema.TableName format. For instance, to reference the Customer table, you would use Sales.Customer.***

## Database Tables

### **Table Sales.Stores**

The  Sales.Stores table includes the store’s information. Each store has a store name, contact information such as phone and email, and an address including street, city, state, and zip code.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | CREATE TABLE sales.stores (  StoreId INT IDENTITY (1, 1) PRIMARY KEY,  StoreName VARCHAR (255) NOT NULL,  Phone VARCHAR (25),  Email VARCHAR (255),  Street VARCHAR (255),  City VARCHAR (255),  State VARCHAR (10),  ZipCode VARCHAR (5)  ); |

### **Table Sales.Staffs**

The  Sales.Staffs table stores the essential information of staffs including first name, last name. It also contains the communication information such as email and phone.

A staff works at a store specified by the value in the StoreId column. A store can have one or more staffs.

A staff reports to a store manager specified by the value in the ManagerId column. If the value in the ManagerId is null, then the staff is the top manager.

If a staff no longer works for any stores, the value in the active column is set to zero.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | CREATE TABLE Sales.Staffs (  StaffId INT IDENTITY (1, 1) PRIMARY KEY,  FirstName VARCHAR (50) NOT NULL,  LastName VARCHAR (50) NOT NULL,  Email VARCHAR (255) NOT NULL UNIQUE,  Phone VARCHAR (25),  Active tinyint NOT NULL,  Store\_id INT NOT NULL,  ManagerId INT,  FOREIGN KEY (StoreId)          REFERENCES Sales.Stores (StoreId)          ON DELETE CASCADE ON UPDATE CASCADE,  FOREIGN KEY (ManagerId)          REFERENCES Sales.Staffs (StaffId)          ON DELETE NO ACTION ON UPDATE NO ACTION  ); |

### **Table Production.Categories**

The Production.Categories table stores the bike’s categories such as children bicycles, comfort bicycles, and electric bikes.

|  |  |
| --- | --- |
| 1  2  3  4 | CREATE TABLE Production.Categories (  CategoryId INT IDENTITY (1, 1) PRIMARY KEY,  CategoryName VARCHAR (255) NOT NULL  ); |

### **Table Production.Brands**

The  Production.Brands table stores the brand’s information of bikes, for example, Electra, Haro, and Heller.

|  |  |
| --- | --- |
| 1  2  3  4 | CREATE TABLE Production.Brands (  BrandId INT IDENTITY (1, 1) PRIMARY KEY,  BrandName VARCHAR (255) NOT NULL  ); |

### **Table Production.Products**

The Production.Products table stores the product’s information such as name, brand, category, model year, and list price.

Each product belongs to a brand specified by the BrandId column. Hence, a brand may have zero or many products.

Each product also belongs a category specified by the CategoryId column. Also, each category may have zero or many products.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | CREATE TABLE Production.Products (  ProductId INT IDENTITY (1, 1) PRIMARY KEY,  ProductName VARCHAR (255) NOT NULL,  BrandId INT NOT NULL,  CategoryId INT NOT NULL,  ModelYear SMALLINT NOT NULL,  ListPrice DECIMAL (10, 2) NOT NULL,  FOREIGN KEY (CategoryId)          REFERENCES Production.Categories (CategoryId)          ON DELETE CASCADE ON UPDATE CASCADE,  FOREIGN KEY (BrandId)          REFERENCES Sales.Brands (BrandId)          ON DELETE CASCADE ON UPDATE CASCADE  ); |

### **Table Sales.Customers**

The  Sales.Customers table stores customer’s information including first name, last name, phone, email, street, city, state and zip code.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | CREATE TABLE Sales.Customers (  CustomerId INT IDENTITY (1, 1) PRIMARY KEY,  FirstName VARCHAR (255) NOT NULL,  LastName VARCHAR (255) NOT NULL,  Phone VARCHAR (25),  Email VARCHAR (255) NOT NULL,  Street VARCHAR (255),  City VARCHAR (50),  State VARCHAR (25),  ZipCode VARCHAR (5)  ); |

### **Table Sales.Orders**

The Sales.Orders table stores the sales order’s header information including customer, order status, order date, required date, shipped date.

It also stores the information on where the sales transaction created (Store) and who created it (Staff).

Each sales order has a row in the SalesOrder table. A sales order has one or many line items stored in the Sales.OrderItems table.

The OrderStatus state can be: 1 = Pending; 2 = Processing; 3 = Rejected; 4 = Completed.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | CREATE TABLE Sales.Orders (  OrderId INT IDENTITY (1, 1) PRIMARY KEY,  CustomerId INT,  OrderStatus tinyint NOT NULL,  OrderDate DATE NOT NULL,  RequiredDate DATE NOT NULL,  ShippedDate DATE,  StoreId INT NOT NULL,  StaffId INT NOT NULL,  FOREIGN KEY (CustomerId)          REFERENCES Sales.Customers (CustomerId)          ON DELETE CASCADE ON UPDATE CASCADE,  FOREIGN KEY (StoreId)          REFERENCES Sales.Stores (StoreId)          ON DELETE CASCADE ON UPDATE CASCADE,  FOREIGN KEY (StaffId)          REFERENCES Sales.Staffs (StaffId)          ON DELETE NO ACTION ON UPDATE NO ACTION  ); |

### **Table Sales.OrderItems**

The Sales.OrderItems table stores the line items of a sales order. Each line item belongs to a sales order specified by the OrderId column.

A sales order line item includes product, order quantity, list price and discount.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | CREATE TABLE Sales.OrderItems(  OrderId INT,  ItemId INT,  ProductId INT NOT NULL,  Quantity INT NOT NULL,  ListPrice DECIMAL (10, 2) NOT NULL,  Discount DECIMAL (4, 2) NOT NULL DEFAULT 0,  PRIMARY KEY (OrderId, ItemId),  FOREIGN KEY (OrderId)          REFERENCES Sales.Orders (OrderId)          ON DELETE CASCADE ON UPDATE CASCADE,  FOREIGN KEY (ProductId)          REFERENCES Production.Products (ProductId)          ON DELETE CASCADE ON UPDATE CASCADE  ); |

### **Table Production.Stocks**

The Production.Stocks table stores the inventory information i.e. the quantity of a product in a specific store.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | CREATE TABLE Production.Stocks (  StoreId INT,  ProductId INT,  Quantity INT,  PRIMARY KEY (StoreId, ProductId),  FOREIGN KEY (StoreId)          REFERENCES Sales.Stores (StoreId)          ON DELETE CASCADE ON UPDATE CASCADE,  FOREIGN KEY (ProductId)          REFERENCES Production.Products (ProductId)          ON DELETE CASCADE ON UPDATE CASCADE  ); |

**Query Reports**

1. (5 pts.) How many unique products does the business have?

SELECT COUNT(\*) FROM Production.Products



1. (5 pts.) Create a report of all staff and their manager. Column listings are:
   1. Staff Full-Name
   2. Manager Full-Name

SELECT s.FirstName + ' ' + s.LastName AS [Staff Full-Name], ss.FirstName + ' ' + ss.LastName AS [Manager Full-Name]

FROM Sales.Staff s JOIN Sales.Staff ss

ON ss.ManagerId = s.StaffId



1. (5 pts.) Create a report that shows all brands and the number of products each has. The two columns listings are:
   1. Count of Products
   2. Brand Name

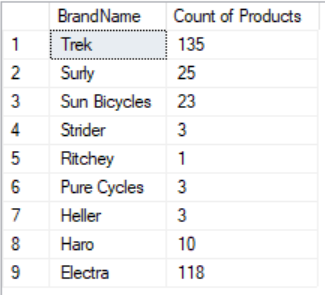
SELECT b.BrandName, COUNT(\*) AS [Count of Products]

FROM Production.Brands b JOIN Production.Products p

ON b.BrandId = p.BrandId

GROUP BY b.BrandName

ORDER BY b.BrandName DESC



Sort the results by brand name in descending order.

1. (5 pts.) Create a report with customers living in Texas or California. The three columns listings are:
   1. Customer's Full-Name
   2. Customer's City and State
   3. E-mail Address

SELECT FirstName + ' ' + LastName AS [Customer's Full-Name], City + ', ' + State AS [Customer's City and State], Email

FROM Sales.Customers

WHERE State = 'CA' OR State = 'TX'

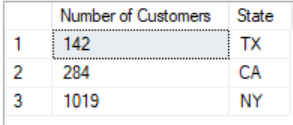


1. (10 pts.) Create a report that shows the number of customers per state. The two columns listing are:
   1. State
   2. Number of Customers

SELECT COUNT(\*) AS [Number of Customers], State

FROM Sales.Customers

GROUP BY State



1. (10 pts.) Calculate the average discount given to customers in California by Brand Name. The two column listings are
   1. Average Discount
   2. Brand Name

SELECT CAST((CAST(AVG(oi.Discount \* 100) as decimal(10, 2))) as varchar) + '%' AS [Average Discount], b.BrandName

FROM Sales.Orders o JOIN Sales.Customers c ON o.CustomerId = c.CustomerId

JOIN Sales.OrderItems oi ON o.OrderId = oi.OrderId

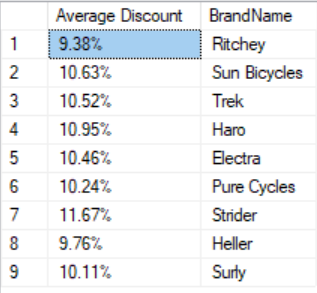
JOIN Production.Products p ON p.ProductId = oi.ProductId

JOIN Production.Brands b ON b.BrandId = p.BrandId

WHERE c.State = 'CA'

GROUP BY BrandName

The format of the discount should be displayed with two decimal places and the percent symbol (e.g. format: 9.83%).



1. (10 pts.) List the total sales by category for the 2017 and 2018. Columns listings are:
   1. Year
   2. Category
   3. Total sales for that category

SELECT YEAR(o.OrderDate) AS [Year] , c.CategoryName, CAST(SUM((oi.ListPrice - oi.ListPrice \* oi.Discount) \* oi.Quantity) as numeric(10,2)) AS [Total Sales]

FROM Production.Categories c JOIN Production.Products p ON c.CategoryId = p.CategoryId

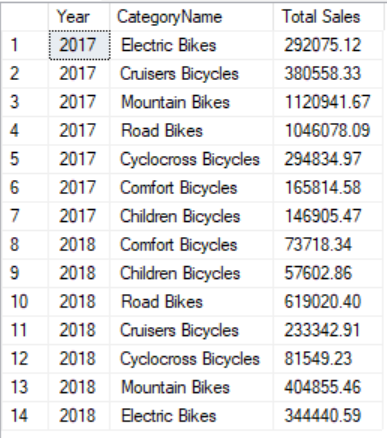
JOIN Sales.OrderItems oi ON oi.ProductId = p.ProductId

JOIN Sales.Orders o ON o.OrderId = oi.OrderId

WHERE YEAR(o.OrderDate) = 2017 OR YEAR(o.OrderDate) = 2018

GROUP BY CategoryName, YEAR(o.OrderDate)

ORDER BY YEAR(o.OrderDate)



1. (10 pts.) The district manager wants to know how many children's bicyles each store sold in 2017. The two columns listings are:
   1. Store Name
   2. # of Children's Bicycles Sold in 2017

SELECT s.StoreName AS [Store Name], COUNT(\*) AS [# of Children's Bicycles Sold in 2017]

FROM Production.Categories c JOIN Production.Products p ON c.CategoryId = p.CategoryId

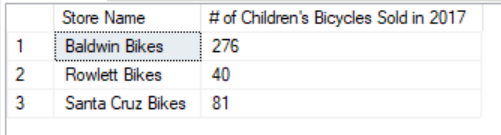
JOIN Sales.OrderItems oi ON oi.ProductId = p.ProductId

JOIN Sales.Orders o ON o.OrderId = oi.OrderId

JOIN Sales.Stores s ON s.StoreId = o.StoreId

WHERE c.CategoryName = 'Children Bicycles' AND YEAR(o.OrderDate) = 2017

GROUP BY StoreName

**Extra Credit (5 pts.)**

**Introduction to Common Table Expressions**

A CTE (Common Table Expression) is temporary result set that you can reference within another SELECT, INSERT, UPDATE, or DELETE statement.

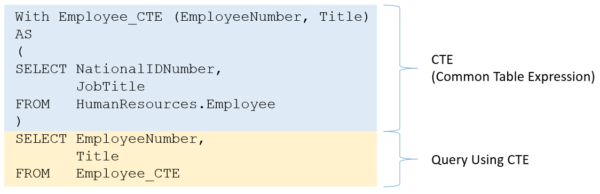
A CTE always returns a result set.  They are used to simplify queries, for example, you could use one to eliminate a derived table from the main query body:

What is a CTE or Common Table Expression in SQL Server?

A CTE (Common Table Expression) defines a temporary result set which you can then use in a SELECT statement.  It becomes a convenient way to manage complicated queries.

Common Table Expressions are defined within the statement using the WITH operator.  You can define one or more common table expression in this fashion.

Here is a really simple example of one CTE:



**Extra-Credit Challenge**

The district manager wants the report re-run in #8 to list the staff member’s name that sold the most children’s bicycles in 2017 so that she can give them a bonus. You may utilize a CTE if it helps you. Provide the code and the screen shot of the output. The three columns listings are:

* 1. Store name
  2. Staff member that sold the most children’s bicycles in 2017
  3. # of Children's Bicycles that this particular staff member sold in 2017