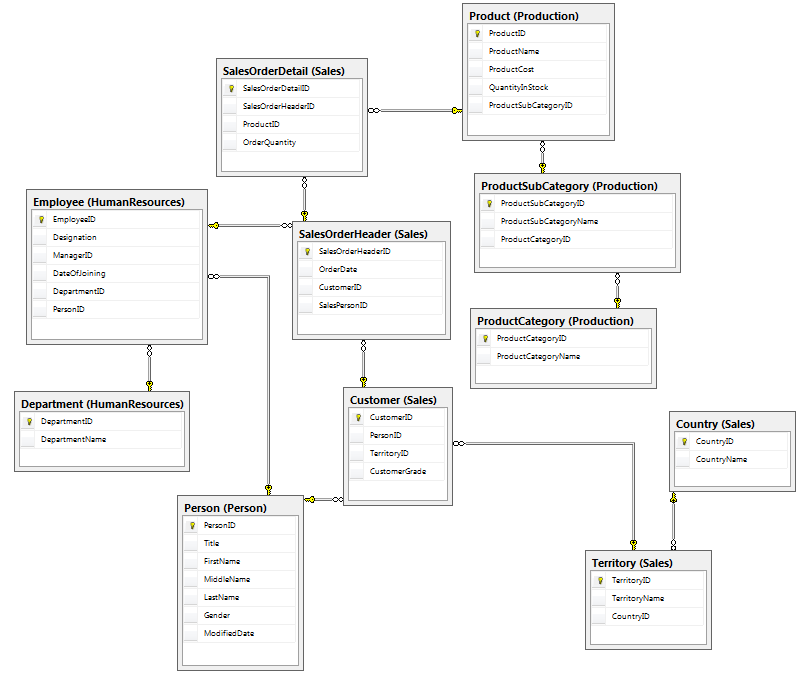
MS SQL SERVER – LABS

1. Create Physical Data Model as per the diagram below. Identify and apply correct datatypes and constraints for the columns of each table

  
Solution:

--Person

CREATE SCHEMA Person;

CREATE TABLE Person.Person (

PersonID INT PRIMARY KEY IDENTITY(1,1),

Title VARCHAR (10) CHECK (Title IN ('Mr.', 'Mrs.', 'Dr.')),

FirstName VARCHAR(20) NOT NULL,

MiddleName VARCHAR(20) NOT NULL,

LastName VARCHAR(20) NOT NULL,

Gender VARCHAR(1) CHECK (Gender IN ('M','F','O','U')),

ModifiedDate DATE DEFAULT GETDATE()

);

------------------------------------------------------------------------------

--Sales

CREATE SCHEMA Sales;

CREATE TABLE Sales.Country(

CountryID INT PRIMARY KEY IDENTITY(1,1),

CountryName VARCHAR(20) NOT NULL

);

--

CREATE TABLE Sales.Territory(

TerritoryID INT PRIMARY KEY IDENTITY(1,1),

TerritoryName VARCHAR(20) NOT NULL,

CountryID INT REFERENCES Sales.Country(CountryID) ON DELETE CASCADE NOT NULL

);

--

CREATE TABLE Sales.Customer(

CustomerID INT PRIMARY KEY IDENTITY(1,1),

PersonID INT REFERENCES Person.Person(PersonID) ON DELETE CASCADE NOT NULL,

TerritoryID INT REFERENCES Sales.Territory(TerritoryID) ON DELETE CASCADE NOT NULL,

CustomerGrade VARCHAR(1) NOT NULL CHECK (CustomerGrade IN ('A','B','C','D','O')),

);

--

CREATE TABLE Sales.SalesOrderHeader (

SalesOrderHeaderID INT PRIMARY KEY IDENTITY(1,1),

OrderDate DATE DEFAULT GETDATE(),

CustomerID INT REFERENCES Sales.Customer(CustomerID),

SalesPersonID INT REFERENCES HumanResources.Employee(EmployeeID)

);

--

CREATE TABLE Sales.SalesOrderDetail (

SalesOrderDetailID INT PRIMARY KEY IDENTITY(100001,1),

SalesOrderHeaderID INT REFERENCES Sales.SalesOrderHeader(SalesOrderHeaderID),

ProductID INT REFERENCES Production.Product(ProductID),

OrderQuantity INT NOT NULL

);

------------------------------------------------------------------------------

--HumanResources

CREATE SCHEMA HumanResources;

CREATE TABLE HumanResources.Department (

DepartmentID INT PRIMARY KEY IDENTITY(1,1),

DepartmentName VARCHAR(20) NOT NULL

);

--

CREATE TABLE HumanResources.Employee (

EmployeeID INT PRIMARY KEY IDENTITY(1,1),

Designation VARCHAR(20) NOT NULL,

ManagerID INT REFERENCES HumanResources.Employee(EmployeeID),

DateOfJoinning DATE NOT NULL,

DepartmentID INT REFERENCES HumanResources.Department(DepartmentID),

PersonID INT REFERENCES Person.Person(PersonID) ON DELETE CASCADE NOT NULL

);

------------------------------------------------------------------------------

--Production

CREATE SCHEMA Production;

CREATE TABLE Production.ProductCategory (

ProductCategoryID INT PRIMARY KEY IDENTITY(10001,1),

ProductCategoryName VARCHAR(30) NOT NULL

);

--

CREATE TABLE Production.ProductSubCategory (

ProductSubCategoryID INT PRIMARY KEY IDENTITY(10001,1),

ProductSubCategory VARCHAR(20) NOT NULL,

ProductCategoryID INT REFERENCES Production.ProductCategory(ProductCategoryID)

);

--

CREATE TABLE Production.Product (

ProductID INT PRIMARY KEY IDENTITY(101,1),

ProductName VARCHAR(20) NOT NULL,

ProductCost MONEY NOT NULL,

ProductSubCategoryID INT REFERENCES Production.ProductSubCategory(ProductSubCategoryID)

);

1. Restore AdventureWorks Database

Step 1: Download: <https://docs.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver16&tabs=ssms> – Download AdventureWorks2019.bak

Step 2: Restore the database from backup file - <https://www.youtube.com/watch?v=mRI8vpn-tyk>

Solution: Done

1. Solve the below (Any 25)
2. Write a query that displays all the rows from the Person.Person table where the rows were modified after December 29, 2000. Display the business entity ID number, the name columns, and the modified date.

**Solution:**

SELECT BusinessEntityID, FirstName, MiddleName, LastName, ModifiedDate

FROM Person.Person WHERE ModifiedDate > '29-Dec-2000';

1. Rewrite the query from question 1 so that it displays the rows that were not modified during December 2000.

**Solution:**

SELECT BusinessEntityID, FirstName, MiddleName, LastName, ModifiedDate FROM Person.Person

WHERE ModifiedDate NOT BETWEEN '1-December-2000' AND '31-December-2000';

1. Write a query that displays the product ID and name for each product from the Production.Product table with the name starting with Chain.

**Solution:**

SELECT ProductID, Name FROM Production.Product

WHERE Name LIKE 'Chain%';

1. Write a query that displays the business entity ID number, first name, middle name, and last name from the Person.Person table for only those rows that have E or B stored in the middle name column.

**Solution:**

SELECT BusinessEntityID, FirstName, MiddleName, LastName FROM Person.Person

WHERE MiddleName IN ('E','B');

1. Write a query displaying the order ID, order date, and total due from the Sales.SalesOrderHeader table. Retrieve only those rows where the order was placed during the month of September 2001 and the total due exceeded $1,000.

**Solution:**

SELECT SalesOrderID, OrderDate, TotalDue FROM Sales.SalesOrderHeader

WHERE OrderDate BETWEEN '1-September-2001' AND '30-September-2001' AND TotalDue = 1000;

1. Write a query displaying the sales orders where the total due exceeds $1,000. Retrieve only those rows where the salesperson ID is 279 or the territory ID is 6.

**Solution:**

SELECT \* FROM Sales.SalesOrderHeader

WHERE TotalDue > 1000 AND SalesPersonID = 279 OR TerritoryID = 6;

1. Write a query displaying the ProductID, Name, and Color columns from rows in the Production.Product table. Display only those rows in which the color is not blue.

**Solution:**

SELECT ProductID, Name, Color FROM Production.Product

WHERE Color <> 'Blue';

1. Write a query that returns the business entity ID and name columns from the Person.Person table. Sort the results by LastName, FirstName, and MiddleName.

**Solution:**

SELECT BusinessEntityID, FirstName, MiddleName, LastName FROM Person.Person

ORDER BY LastName, MiddleName, FirstName;

1. Write a query that displays in the “AddressLine1 (City PostalCode)” format from the Person.Address table.

**Solution:**

SELECT CONCAT(AddressLine1,' (', City, PostalCode,')') FROM Person.Address;

1. Write a query using the Production.Product table displaying the product ID, color, and name columns. If the color column contains a NULL value, replace the color with No Color.

**Solution:**

SELECT ProductID, Name, ISNULL(Color,'No Color') FROM Production.Product

1. Modify the query written in question 2 so that the description of the product is displayed in the “Name: Color” format. Make sure that all rows display a value even if the Color value is missing.

**Solution:**

SELECT ProductID, CONCAT('"', Name, ': ', Color, '"') AS 'Product name with color' FROM Production.Product;

1. Write a query using the Sales.SpecialOffer table. Display the difference between the MinQty and MaxQty columns along with the SpecialOfferID and Description columns.

**Solution:**

SELECT SpecialOfferID, Description, MaxQty - MinQty AS 'Difference' FROM Sales.SpecialOffer;

1. Write a query using the Sales.SpecialOffer table that multiplies the MaxQty column by the DiscountPCT column. If the MaxQty value is null, replace it with the value 10. Include the SpecialOfferID and Description columns in the results.

**Solution:**

SELECT SpecialOfferID, Description, ISNULL(MaxQty,10) \* DiscountPct AS 'Result' FROM Sales.SpecialOffer;

1. Write a query that displays the first 10 characters of the AddressLine1 column in the Person.Address table.

**Solution:**

SELECT SUBSTRING(AddressLine1,1,10) FROM Person.Address;

1. Write a query that calculates the number of days between the date an order was placed and the date that it was shipped using the Sales.SalesOrderHeader table. Include the SalesOrderID, OrderDate, and ShipDate columns.

**Solution:**

SELECT SalesOrderID, OrderDate, ShipDate, DATEDIFF(D, OrderDate,ShipDate) AS [NUMBER OF DAYS] FROM Sales.SalesOrderHeader;

1. Write a query that displays only the date, not the time, for the order date and ship date in the Sales.SalesOrderHeader table. (Use any of the styles that return only date)

**Solution:**

SELECT SalesOrderID, CONVERT(VARCHAR(10), OrderDate,120) AS [Order Date], CONVERT(VARCHAR(10), ShipDate,120) AS [Ship Date] FROM Sales.SalesOrderHeader;

1. Write a query that adds six months to each order date in the Sales.SalesOrderHeader table. Include the SalesOrderID and OrderDate columns.

**Solution:**

SELECT SalesOrderID, OrderDate, DATEADD(M, 6, OrderDate) AS [UPDATED DATE] FROM Sales.SalesOrderHeader;

1. Write a query that displays the year of each order date and the numeric month of each order date in separate columns in the results. Include the SalesOrderID and OrderDate columns.

**Solution:**

SELECT SalesOrderID, OrderDate, MONTH(OrderDate) AS [month], YEAR(OrderDate) AS [Year] FROM Sales.SalesOrderHeader;

1. Write a statement that generates a random number between 1 and 10 each time it is run

**Solution:**

SELECT CAST(RAND() \* 10 AS INT) +1;

1. Write a query using the Sales.SalesOrderHeader table to display the orders placed during 2001 by using a function. Include the SalesOrderID and OrderDate columns in the results.

**Solution:**

SELECT SalesOrderID, OrderDate FROM Sales.SalesOrderHeader

WHERE YEAR(OrderDate) = 2011;

1. Write a query using the Sales.SalesOrderHeader table listing the sales in order of the month the order was placed and then the year the order was placed. Include the SalesOrderID and OrderDate columns in the results.

**Soltuion:**

SELECT SalesOrderID, OrderDate FROM Sales.SalesOrderHeader

ORDER BY MONTH(OrderDate) , YEAR(OrderDate);

1. The HumanResources.Employee table does not contain the employee names. Join that table to the Person.Person table on the BusinessEntityID column. Display the job title, birth date, first name, and last name.

**Solution:**

SELECT EMP.JobTitle, EMP.BirthDate, PER.FirstName, PER.LastName

FROM HumanResources.Employee EMP INNER JOIN Person.Person PER

ON EMP.BusinessEntityID = PER.BusinessEntityID;

1. The customer names also appear in the Person.Person table. Join the Sales.Customer table to the Person.Person table. The BusinessEntityID column in the Person.Person table matches the PersonID column in the Sales.Customer table. Display the CustomerID, StoreID, and TerritoryID columns along with the name columns.

**Solution:**

SELECT CustomerID, StoreID, TerritoryID, FirstName, MiddleName,LastName FROM Person.Person PER INNER JOIN Sales.Customer CUS

ON PER.BusinessEntityID = CUS.PersonID;

1. Write a query that joins the Sales.SalesOrderHeader table to the Sales. SalesPerson table. Join the BusinessEntityID column from the Sales.SalesPerson table to the SalesPersonID column in the Sales.SalesOrderHeader table. Display the SalesOrderID along with the SalesQuota and Bonus.   
   **Solution:**

SELECT SalesOrderID, SalesQuota, Bonus FROM

Sales.SalesOrderHeader SOH INNER JOIN Sales.SalesPerson SP

ON SOH.SalesPersonID = SP.BusinessEntityID

1. The catalog description for each product is stored in the Production.ProductModel table. Display the columns that describe the product from the Production.Product table, such as the color and size along with the catalog description for each product.

**Solution:**

SELECT Color, Size, PRDM.CatalogDescription FROM

Production.Product PRD INNER JOIN Production.ProductModel PRDM

ON PRD.ProductModelID = PRDM.ProductModelID;

1. Write a query that displays the names of the customers along with the product names that they have purchased. Hint: Five tables will be required to write this query!

**Solution:**

SELECT FirstName, MiddleName, LastName, PRD.Name FROM Sales.Customer C

INNER JOIN Person.Person P ON P.BusinessEntityID = C.PersonID

INNER JOIN Sales.SalesOrderHeader SOH ON C.CustomerID = SOH.CustomerID

INNER JOIN Sales.SalesOrderDetail SOD ON SOH.SalesOrderID = SOD.SalesOrderID

INNER JOIN Production.Product PRD ON PRD.ProductID = SOD.ProductID;

1. Write a query that displays all the products along with the SalesOrderID even if an order has never been placed for that product. Join to the Sales.SalesOrderDetail table using the ProductID column.

**Solution:**

SELECT SalesOrderID, P.ProductID, P.Name FROM Sales.SalesOrderDetail SOD RIGHT OUTER JOIN Production.Product P ON SOD.ProductID = P.ProductID;

1. The Sales.SalesOrderHeader table contains foreign keys to the Sales.CurrencyRate and Purchasing.ShipMethod tables. Write a query joining all three tables, making sure it contains all rows from Sales.SalesOrderHeader. Include the CurrencyRateID, AverageRate, SalesOrderID, and ShipBase columns.

**Solution:**

SELECT CR.CurrencyRateID, CR.AverageRate, SOH.SalesOrderID, SM.ShipBase FROM Sales.SalesOrderHeader SOH

LEFT OUTER JOIN Sales.CurrencyRate CR ON SOH.CurrencyRateID = CR.CurrencyRateID

LEFT OUTER JOIN Purchasing.ShipMethod SM ON SOH.ShipMethodID = SM.ShipMethodID;

1. Write a query to determine the number of customers in the Sales.Customer table.

**Solution:**

SELECT COUNT(\*) AS [Number Of Customers] FROM Sales.Customer;

1. Write a query using the Production.Product table that displays the minimum, maximum, and average ListPrice.

**Solution:**

SELECT Min(ListPrice) AS [Minimum Price], MAX(ListPrice) AS [Maximum Price], AVG(ListPrice) AS [Average Price] FROM Production.Product;

1. Write a query that shows the total number of items ordered for each product. Use the Sales.SalesOrderDetail table to write the query.

**Solution:**

SELECT ProductID, SUM(OrderQty) AS [Total Number of items for each product] FROM

Sales.SalesOrderDetail GROUP BY ProductID;

1. Write a query using the Sales.SalesOrderDetail table that displays a count of the detail lines for each SalesOrderID.

**Solution:**

SELECT SalesOrderID, COUNT(\*) AS [Count] FROM Sales.SalesOrderDetail

GROUP BY SalesOrderID;

1. Write a query using the Production.Product table that lists a count of the products in each product line.

**Solution:**

SELECT COUNT(\*) AS [Count of products], ProductLine FROM Production.Product GROUP BY ProductLine;

1. Write a query that displays the count of orders placed by year for each customer using the Sales.SalesOrderHeader table.

**Solution:**

SELECT YEAR(OrderDate) AS [Year], CustomerID, COUNT(\*) AS [Order Placed Per Year By Customer] FROM Sales.SalesOrderHeader GROUP BY YEAR(OrderDate), CustomerID;

1. Write a query that creates a sum of the LineTotal in the Sales.SalesOrderDetail table grouped by the SalesOrderID. Include only those rows where the sum exceeds 1,000.

**Solution:**

SELECT SalesOrderID, SUM(LineTotal) AS [Total Line] FROM Sales.SalesOrderDetail GROUP BY SalesOrderID HAVING SUM(LineTotal) > 1000;

1. Write a query that groups the products by ProductModelID along with a count. Display the rows that have a count that equals 1.

**Solution:**

SELECT ProductModelID, COUNT(\*) AS [Group Of Product Per Model] FROM Production.ProductModel GROUP BY ProductModelID HAVING COUNT(\*) = 1;

1. Write a query using the Sales.SalesOrderHeader, Sales.SalesOrderDetail, and Production.Product tables to display the total sum of products by ProductID and OrderDate.

**Solution:**

SELECT P.ProductID, OrderDate, SUM(SOD.ProductID) AS [Total Product] FROM Sales.SalesOrderHeader SOH

INNER JOIN Sales.SalesOrderDetail SOD ON SOH.SalesOrderID = SOD.SalesOrderID

INNER JOIN Production.Product P ON P.ProductID = SOD.ProductID

GROUP BY P.ProductID, OrderDate;

1. Display the 3rd joined employee.

**Solution:**

SELECT BusinessEntityID, Gender, BirthDate, HireDate FROM (

SELECT BusinessEntityID, Gender, BirthDate, HireDate ,ROW\_NUMBER()OVER(ORDER BY HireDate) AS [ROW NUMBER] FROM HumanResources.Employee) AS [Employee]

WHERE [ROW NUMBER] = 3;

1. Display the customer who has placed 2nd highest orders

**Solution:**

SELECT SalesOrderID, OrderQty, CustomerID,[Rank] FROM (

SELECT SOH.SalesOrderID AS [SalesOrderID], OrderQty, C.CustomerID AS [CustomerID], ROW\_NUMBER()OVER(ORDER BY OrderQty DESC) AS [Rank] FROM

Sales.Customer C INNER JOIN Sales.SalesOrderHeader SOH ON C.CustomerID = SOH.CustomerID

INNER JOIN Sales.SalesOrderDetail SOD ON SOH.SalesOrderID = SOD.SalesOrderID) AS [Cutomer] WHERE [Rank] = 2;

--Lab 4: Based on Stored Procedures  
Q1.    An input string representing passenger data comes in a below format to a procedure.  
 Extract the data from the string and store in a temporary table. String format: “[P9001,John Roy,Male,12-Jan-2009]”

CREATE TABLE #Passenger

(

PassengerID INT PRIMARY KEY IDENTITY(1,1),

PassengerPNR VARCHAR(20),

PassengerName VARCHAR(20),

Gender VARCHAR(10),

DateOFBirth DATE

);

DROP PROCEDURE SP\_GetPassengerDetails;

CREATE PROCEDURE SP\_GetPassengerDetails

(

@pPassengerPNR VARCHAR(20),

@pPassengerName VARCHAR(20),

@pGender VARCHAR(10),

@pDateOFBirth DATE

) AS

BEGIN

IF @pGender IN('Male', 'Female', 'Other') AND @pPassengerName IS NOT NULL

BEGIN

INSERT INTO #Passenger (PassengerPNR, PassengerName, Gender, DateOFBirth)

VALUES (@pPassengerPNR, @pPassengerName, @pGender, @pDateOFBirth);

SELECT CONCAT('"', '[',PassengerPNR, ',', PassengerName, ',', Gender, ',', DateOFBirth, ']', '"') FROM #Passenger;

PRINT 'RECORD SAVED..';

END;

ELSE

BEGIN

PRINT 'INVALID DATA..';

END;

END;

EXECUTE SP\_GetPassengerDetails 'P9001', 'John Roy', 'Male','12-Jan-2009';

Q2.    Modify the Day 4 - Q1 to validate the below  
    A.    No duplicate entry for a passenger must be attempted to insert in table  
    B.    The Age of the passenger must be between 6 to 90

CREATE PROCEDURE SP\_GetPassengerDetailsValidation

(

@pPassengerPNR VARCHAR(20),

@pPassengerName VARCHAR(20),

@pGender VARCHAR(10),

@pDateOFBirth DATE

) AS

BEGIN

DECLARE @pValidatePNR VARCHAR(20);

SELECT @pValidatePNR = PassengerPNR FROM #Passenger WHERE PassengerPNR = @pPassengerPNR

IF (YEAR(GETDATE())-YEAR(@pDateOFBirth) BETWEEN 6 AND 90) AND @pGender IN('Male', 'Female', 'Other')

AND @pPassengerName IS NOT NULL AND @pGender IS NOT NULL AND @pDateOFBirth IS NOT NULL

BEGIN

INSERT INTO #Passenger (PassengerPNR, PassengerName, Gender, DateOFBirth)

VALUES (@pPassengerPNR, @pPassengerName, @pGender, @pDateOFBirth);

SELECT CONCAT('"', '[',PassengerPNR, ',', PassengerName, ',', Gender, ',', DateOFBirth, ']', '"') FROM #Passenger;

PRINT 'RECORD SAVED..';

END;

ELSE

BEGIN

PRINT 'INVALID DATA..';

END;

END;

Lab 5: USER Defined Functions  
Q1.    Create a Function to check and print all the prime numbers between a range defined by user.  
Q2.    Create a Function that takes CategoryName as a parameter and gets the products associated with that category.

Lab 6: Trigger

Q1. Create a trigger to implement banking scenario. Whenever the bank balance is updated for any bank account a transaction is captured in a  
Bank Transaction table. The Bank Transaction table acts as a source to generate bank statements.  
**Hint: Need to create two tables (BankAccounts with some rows and BankTransactions with no records) and a trigger for Update on Balance column of BankAccounts table.**  
Table: BankAccounts  
Columns:  
•    AccountID (Numeric- Autogenerated)  
•    CustomerName  
•    AccountType (Current / Saving)  
•    Balance (>0)  
•    Modified Date  
Table: BankTransactions  
Columns  
•    TransactionID (Numeric- Autogenerated)  
•    AccountID (FK)  
•    TransactionDate  
•    TransactionType (Debit / Credit)  
•    TransactionAmount

Lab 7: Exception Handling  
Q1.    Create a procedure that takes a string parameter. The input string may be a string or a numeric or NULL value.  
Convert the string to Integer. If it cannot be converted write an exception handling section to handle the appropriate error.  
If the string is converted to integer print Hello the input integer number of times  
Q2.    Create a temp table to represent employees. Design a user defined exception to handle the salary input less than 10000.

Lab 8: Transactions and Deadlock  
Q1.Document your understanding and possible solutions to Deadlock concept [Hint: You may explore online]