Lab 3

1. **Using a subquery, display the product names and product ID numbers from the Production.Product table that have been ordered.**
   1. SELECT ProductID, Name FROM Production.Product WHERE ProductID IN (SELECT ProductID FROM Sales.SalesOrderDetail);
2. **Change the query written in question 1 to display the products that have not been ordered.**
   1. SELECT ProductID, Name FROM Production.Product WHERE ProductID NOT IN (

SELECT ProductID FROM Sales.SalesOrderDetail WHERE ProductID IS NOT NULL);

1. **Write a UNION query that combines the ModifiedDate from Person.Person and the HireDate from HumanResources.Employee.**
   1. SELECT ModifiedDate FROM Person.Person

UNION

SELECT HireDate FROM HumanResources.Employee;

1. **Create a table called dbo.testCustomer. Include a CustomerID that is an identity column primary key. Include FirstName and LastName columns. Include an Age column with a check constraint specifying that the value must be less than 120. Include an Active column that is one character with a default of Y and allows only Y or N. Add some rows to the table.**
   1. CREATE TABLE dbo.testCustomer ( CustomerID INT NOT NULL IDENTITY PRIMARY KEY, FirstName VARCHAR(25), LastName VARCHAR(25), Age INT, Active CHAR(1) DEFAULT 'Y', CONSTRAINT ch\_testCustomer\_Age CHECK (Age < 120), CONSTRAINT ch\_testCustomer\_Active CHECK (Active IN ('Y','N')));

GO

INSERT INTO dbo.testCustomer(FirstName, LastName,Age) VALUES ('Kathy','Morgan',35),('Lady B.','Kellenberger',14), ('Luke','Moore',30);

1. **Create a table called dbo.testOrder. Include a CustomerID column that is a foreign key pointing to dbo.testCustomer. Include an OrderID column that is an identity column primary key. Include an OrderDate column that defaults to the current date and time. Include a ROWVERSION column. Add some rows to the table.**
   1. CREATE TABLE dbo.testOrder (CustomerID INT NOT NULL, OrderID INT NOT NULL IDENTITY PRIMARY KEY, OrderDate DATETIME DEFAULT GETDATE(), RW ROWVERSION,

CONSTRAINT fk\_testOrders FOREIGN KEY (CustomerID) REFERENCES dbo.testCustomer(CustomerID));

GO

INSERT INTO dbo.testOrder (CustomerID) VALUES (1),(2),(3);

1. **Create a table called dbo.testOrderDetail. Include an OrderID column that is a foreign key pointing to dbo.testOrder. Include an integer ItemID column, a Price column, and a Qty column. The primary key should be a composite key composed of OrderID and ItemID. Create a computed column called LineItemTotal that multiplies Price times Qty. Add some rows to the table.**
   1. CREATE TABLE dbo.testOrderDetail( OrderID INT NOT NULL, ItemID INT NOT NULL, Price Money NOT NULL, Qty INT NOT NULL, LineItemTotal AS (Price \* Qty), CONSTRAINT pk\_testOrderDetail PRIMARY KEY (OrderID, ItemID), CONSTRAINT fk\_testOrderDetail FOREIGN KEY (OrderID) REFERENCES dbo.testOrder(OrderID));

GO

INSERT INTO dbo.testOrderDetail(OrderID,ItemID,Price,Qty) VALUES (1,1,10,5), (1,2,5,10);

1. **Create a view called dbo.vw\_Products that displays a list of the products from the Production.Product table joined to the Production.ProductCostHistory table. Include columns that describe the product and show the cost history for each product. Test the view by creating a query that retrieves data from the view.**
   1. CREATE VIEW dbo.vw\_Products AS (

SELECT P.ProductID, P.Name, P.Color, P.Size, P.Style, H.StandardCost, H.EndDate, H.StartDate FROM Production.Product AS P INNER JOIN Production.ProductCostHistory AS H ON P.ProductID = H.ProductID );

GO

SELECT ProductID, Name, Color, Size, Style, StandardCost, EndDate, StartDate FROM dbo.vw\_Products;

1. **Create a view called dbo.vw\_CustomerTotals that displays the total sales from the TotalDue column per year and month for each customer. Test the view by creating a query that retrieves data from the view.**
   1. CREATE VIEW dbo.vw\_CustomerTotals AS (SELECT C.CustomerID, YEAR(OrderDate) AS OrderYear, MONTH(OrderDate) AS OrderMonth, SUM(TotalDue) AS TotalSales FROM Sales.Customer AS C INNER JOIN Sales.SalesOrderHeader AS SOH ON C.CustomerID = SOH.CustomerID GROUP BY C.CustomerID, YEAR(OrderDate), MONTH(OrderDate));

GO

SELECT CustomerID, OrderYear, OrderMonth, TotalSales FROM dbo.vw\_CustomerTotals;

1. **Create a user-defined function called dbo.fn\_AddTwoNumbers that accepts two integer parameters.Return the value that is the sum of the two numbers. Test the function.**
   1. CREATE FUNCTION dbo.fn\_AddTwoNumbers (@NumberOne INT, @NumberTwo INT)

RETURNS INT AS

BEGIN

RETURN @NumberOne + @NumberTwo;

END;

GO

SELECT dbo.fn\_AddTwoNumbers(1,2);

1. **Create a user-defined function called dbo.Trim that takes a VARCHAR(250) parameter. This function should trim off the spaces from both the beginning and the end of a string. Test the function.**

CREATE FUNCTION dbo.Trim (@Expression VARCHAR(250))

RETURNS VARCHAR(250) AS

BEGIN

RETURN LTRIM(RTRIM(@Expression));

END;

GO

SELECT '\*' + dbo.Trim(' test ') + '\*';

1. **Create a function called dbo.fn\_RemoveNumbers that removes any numeric characters from a VARHCHAR(250) string. Test the function. Hint: The ISNUMERIC function checks to see whether a string is numeric. Check Books Online to see how to use it.**
   1. CREATE FUNCTION dbo.fn\_RemoveNumbers (@Expression VARCHAR(250))

RETURNS VARCHAR(250) AS

BEGIN

DECLARE @NewExpression VARCHAR(250) = '';

DECLARE @Count INT = 1;

DECLARE @Char CHAR(1);

WHILE @Count <= LEN(@Expression)

BEGIN

SET @Char = SUBSTRING(@Expression,@Count,1);

IF ISNUMERIC(@Char) = 0

BEGIN

SET @NewExpression += @Char;

END

SET @Count += 1;

END;

RETURN @NewExpression;

END;

GO

SELECT dbo.fn\_RemoveNumbers('abc 123 baby you and me');

1. **Write a function called dbo.fn\_FormatPhone that takes a string of ten numbers. The function will format the string into this phone number format: “(###) ###-####.” Test the function.**
   1. CREATE FUNCTION dbo.fn\_FormatPhone (@Phone VARCHAR(10))

RETURNS VARCHAR(14) AS BEGIN

DECLARE @NewPhone VARCHAR(14);

SET @NewPhone = '(' + SUBSTRING(@Phone,1,3) + ') ';

SET @NewPhone = @NewPhone + SUBSTRING(@Phone,4,3) + '-';

SET @NewPhone = @NewPhone + SUBSTRING(@Phone,7,4)

RETURN @NewPhone;

END;

GO

SELECT dbo.fn\_FormatPhone('5555551234');

1. **Create a stored procedure called dbo.usp\_CustomerTotals that displays the total sales from the TotalDue column per year and month for each customerTest the stored procedure.**
   1. CREATE PROCEDURE dbo.usp\_CustomerTotals AS

SELECT C.CustomerID, YEAR(OrderDate) AS OrderYear,

MONTH(OrderDate) AS OrderMonth, SUM(TotalDue) AS TotalSales

FROM Sales.Customer AS C

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

GROUP BY C.CustomerID, YEAR(OrderDate), MONTH(OrderDate)

GO

EXEC dbo.usp\_CustomerTotals;