-- DROP Statements

-------

-------

-- drop all tables in reverse order

DROP TABLE IF EXISTS OrderDetail

GO

DROP TABLE IF EXISTS ProdServ

GO

DROP TABLE IF EXISTS InvoiceOrderBridge

GO

DROP TABLE IF EXISTS JobOrder

GO

DROP TABLE IF EXISTS Customer

GO

DROP TABLE IF EXISTS EmployeePayment

GO

DROP TABLE IF EXISTS Employee

GO

DROP TABLE IF EXISTS JobRole

GO

DROP TABLE IF EXISTS Invoice

GO

-------

--DROP FUNCTIONS

DROP FUNCTION IF EXISTS dbo.CustomerIDLookup

GO

DROP FUNCTION IF EXISTS dbo.EmployeeIDLookup

GO

-------

--DROP Procedures/Views

DROP PROCEDURE IF EXISTS DeleteEmployee

GO

DROP PROCEDURE IF EXISTS AddProduct

GO

DROP PROCEDURE IF EXISTS DeleteProduct

GO

DROP PROCEDURE IF EXISTS UpdateProduct

GO

DROP PROCEDURE IF EXISTS UpdatePriority

GO

DROP PROCEDURE IF EXISTS UpdateInvoice

GO

DROP PROCEDURE IF EXISTS DeleteCustomer

GO

DROP PROCEDURE IF EXISTS UpdateJobOrder

GO

DROP VIEW IF EXISTS MonthlyPayroll

GO

DROP VIEW IF EXISTS TotalRevenue

GO

DROP VIEW IF EXISTS CurrentOpenInvoices

GO

DROP VIEW IF EXISTS CurrentClosedInvoices

GO

DROP VIEW IF EXISTS JobOrderTimesCustomer

GO

DROP VIEW IF EXISTS AmountSpentPerCustomer

GO

DROP VIEW IF EXISTS AvgPerOrder

GO

DROP VIEW IF EXISTS TotalPerCustomer

GO

DROP VIEW IF EXISTS MonthlyRevenue

GO

-------

--CREATE Statements in reverse order

--CREATE TABLE Invoice

--CREATE TABLE JobRole

--CREATE TABLE Employee

--CREATE TABLE EmployeePayment

--CREATE TABLE Customer

--CREATE TABLE JobOrder

--CREATE TABLE InvoiceOrderBridge

--CREATE TABLE ProdServ

--CREATE TABLE OrderDetail

--Begin Creating the Invoice Table

GO

CREATE TABLE Invoice(

--Columns for the Invoice table

InvoiceID int identity,

Amount int not null,

Date datetime not null default GetDate(),

Status char(15) not null,

--Constraints on the Invoice Table

CONSTRAINT PK\_Invoice PRIMARY KEY (InvoiceID)

)

GO

-- End Creating Invoice Table

-------

--Begin Creating the JobRole Table

GO

CREATE TABLE JobRole(

--Columns for the JobRole table

JobRoleID int identity,

Title char(30) not null,

Salary int not null,

DutyName char(100) not null,

--Constraints on the JobRole Table

CONSTRAINT PK\_JobRole PRIMARY KEY (JobRoleID)

)

GO

-- End Creating JobRole Table

-------

--Begin Creating the Employee Table

GO

CREATE TABLE Employee(

--Columns for the Employee table

EmployeeID int identity,

FirstName char(30) not null,

LastName char(30) not null,

Address char(50) not null,

City char(50) not null,

State char(2) not null,

Zipcode char(15) not null,

EmailAddress char(30),

PhoneNum char(15) not null,

StartDate datetime not null,

SkillsName char(130) not null,

JobRoleID int not null,

--Constraints on the Employee Table

CONSTRAINT PK\_Employee PRIMARY KEY (EmployeeID),

CONSTRAINT FK1\_Employee FOREIGN KEY (JobRoleID) REFERENCES JobRole(JobRoleID),

CONSTRAINT U1\_Employee UNIQUE (EmailAddress),

CONSTRAINT U2\_Employee UNIQUE (PhoneNum)

)

GO

-- End Creating Employee Table

--------

--Begin Creating the EmployeePayment Table

GO

CREATE TABLE EmployeePayment(

--Columns for the EmployeePayment table

EmployeePaymentID int identity,

EmployeeID int not null,

Amount int not null,

Date datetime not null default GetDate(),

--Constraints on the EmployeePayment Table

CONSTRAINT PK\_EmployeePayment PRIMARY KEY (EmployeePaymentID),

CONSTRAINT FK1\_EmployeePayment FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)

)

GO

-- End Creating EmployeePayment Table

-------

--Begin Creating the Customer Table

GO

CREATE TABLE Customer(

--Columns for the Customer table

CustomerID int identity,

BusinessName char(50) not null,

StartDate datetime not null,

POCFirstName char(30) not null,

POCLastName char(30) not null,

BusinessAddress char(50) not null,

BusinessCity char(50) not null,

BusinessState char(2) not null,

BusinessZipcode char(15) not null,

EmailAddress char(30) not null,

PhoneNum char(15) not null,

--Constraints on the Customer Table

CONSTRAINT PK\_Customer PRIMARY KEY (CustomerID),

CONSTRAINT U1\_Customer UNIQUE (BusinessName),

CONSTRAINT U2\_Customer UNIQUE (EmailAddress),

CONSTRAINT U3\_Customer UNIQUE (PhoneNum)

)

GO

-- End Creating Customer Table

-------

--Begin Creating the JobOrder Table

GO

CREATE TABLE JobOrder(

--Columns for the JobOrder table

OrderID int identity,

CustomerID int not null,

OrderType char(10) not null,

Date datetime not null default GetDate(),

Status char(10) not null,

EmployeeID int not null,

--Constraints on the JobOrder Table

CONSTRAINT PK\_JobOrder PRIMARY KEY (OrderID),

CONSTRAINT FK1\_JobOrder FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),

CONSTRAINT FK2\_JobOrder FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)

)

GO

-- End Creating JobOrder Table

-------

--Begin Creating the InvoiceOrderBridge Table

GO

CREATE TABLE InvoiceOrderBridge(

--Columns for the InvoiceOrderBridge table

InvoiceOrderID int identity,

InvoiceID int not null,

OrderID int not null,

--Constraints on the InvoiceOrderBridge Table

CONSTRAINT PK\_InvoiceOrderBridge PRIMARY KEY (InvoiceOrderID),

CONSTRAINT FK1\_InvoiceOrderBridge FOREIGN KEY (InvoiceID) REFERENCES Invoice(InvoiceID),

CONSTRAINT FK2\_InvoiceOrderBridge FOREIGN KEY (OrderID) REFERENCES JobOrder(OrderID)

)

GO

-- End Creating InvoiceOrderBridge Table

-------

--Begin Creating the ProdServ Table

GO

CREATE TABLE ProdServ(

--Columns for the ProdServ table

ProdServID int identity,

ProductType char(30) not null,

ProductPricesqft int,

ProductPricehour int,

ProductPriceother int,

--Constraints on the ProdServ Table

CONSTRAINT PK\_ProdServ PRIMARY KEY (ProdServID)

)

GO

-- End Creating ProdServ Table

-------

--Begin Creating the OrderDetail Table

GO

CREATE TABLE OrderDetail(

--Columns for the OrderDetail table

OrderDetID int identity,

OrderID int not null,

ProdServID int not null,

DescriptionofRequest char(100) not null,

Deadline datetime not null,

Priority char(10) not null,

--Constraints on the OrderDetail Table

CONSTRAINT PK\_OrderDet PRIMARY KEY (OrderDetID),

CONSTRAINT FK1\_OrderDet FOREIGN KEY (OrderID) REFERENCES JobOrder(OrderID),

CONSTRAINT FK2\_OrderDet FOREIGN KEY (ProdServID) REFERENCES ProdServ(ProdServID)

)

GO

-- End Creating OrderDetail Table

--------

-------

--Adding Data to Tables

-------

-------

--Adding Data to Table JobRole

INSERT INTO JobRole(Title, Salary, DutyName)

VALUES

('Production','31200','Runs printing production'),

('Accountant', '41600', 'Invoices clients, pays vendors, runs payroll'),

('Owner', '150000', 'Owns company'),

('Salesperson', '100000','Ownership of client accounts, client acquisition, responsible for all deliverables'),

('Designer','84000','Designs all projects, sets up files for print, asset management')

SELECT \* FROM JobRole --check results

-------

--Adding Data to Table Employee

--SELECT \* FROM JobRole debug, trying to get subselect stmt to run without error, investigate further...(SELECT JobRoleID WHERE Name = ‘Role 5’))

INSERT INTO Employee(FirstName, LastName, Address, City, State, Zipcode, EmailAddress, PhoneNum, StartDate, SkillsName, JobRoleID)

VALUES

('Willie','Makit','3579 Stonecoal Road','Sacramento','CA','95831','CallofDutyismyLife@xbox.com','916-596-3817','04/12/2020','Large-Format Printing, Product Finishing, Supply/Inventory','1'), --(SELECT JobRoleID WHERE Title = 'Production')),

('Paige','Turner','1181 Deer Ridge Drive','Sacramento','CA','95822','MoneyMama@gmail.com','916-404-4380','05/02/1989','Bookkeeping, Accounts Receivable, Accounts Payable, Payroll', '2'),--(SELECT JobRoleID WHERE Title = 'Accountant')),

('Anita','Bath','1037 Stuart Street','Rancho Murieta','CA','95683','ABath@hotmail.com','916-579-0359','01/23/2010','Owner', '3'),--(SELECT JobRoleID WHERE Title = 'Owner')),

('Augusta','Wind','2756 Green Hill Road','Carmichael','CA','95608','WindyCity999@yahoo.com','916-354-3694','07/17/2015','Business Development, Full-Cycle Account Management', '4'),--(SELECT JobRoleID WHERE Title = 'Salesperson')),

('Saul','Goodmate','1497 Lunetta Street','Elk Grove','CA','95624','DesignedthisWay@Adobe.com','916-515-5070','12/31/1999','Adobe Creative Suite, Graphic Design, Asset Management, Brand Strategy', '5')--(SELECT JobRoleID WHERE Title = 'Designer'))

SELECT \* FROM Employee --check results

-------

--Adding Data to Table Customer

INSERT INTO Customer(BusinessName, StartDate, POCFirstName, POCLastName, BusinessAddress, BusinessCity, BusinessState, BusinessZipcode, EmailAddress, PhoneNum)

VALUES

('Acme Corporation', '07/04/1978', 'Rhodes', 'Runnert', '3655 N Cliff St', 'San Jose', 'CA', '95134', 'RRunnert@Acme.com', '215-834-8729'),

('Globex Corporation', '06/06/2006', 'Hank', 'Scorpio', '123 Globex Blvd', 'Cypress Creek', 'CA', '95132', 'HScorpio@GlobexCO.com', '507-926-7292'),

('Soylent Corp', '08/08/2015', 'Bill', 'Simonson' , '777 S Peoples St', 'Los Angeles', 'CA', '90021', 'BS@Soylent.com', '903-380-2339'),

('Initech', '09/23/2019', 'Milton', 'Waddams', '156 Swingline Rd', 'Mountain View', 'CA', '95030', 'MiltonW@Initech.com', '412-601-8970'),

('Umbrella Corporation', '4/13/2018', 'Alicia', 'Marcus', '7000 Resident Blvd N', 'Las Vegas', 'NV', '89115', 'AMarcus@Umbrella.com', '412-292-1659'),

('Hooli','11/11/2020','Anthony','Dude', '2500 Wideway Dr', 'Santa Monica', 'CA', '90404', 'ADude@Hooli.com', '724-388-7389'),

('Pet Tricks', '03/28/2011','Chelsea','Hill', '100 Remington Cir', 'Los Gatos', 'CA', '95032', 'CHill@PetTricks.com', '412-861-2211')

--------

--Adding Data to Table ProdServ

INSERT INTO ProdServ(ProductType, ProductPricesqft)

VALUES

('Wallpaper', '25'),

('Glass Graphic', '25'),

('Privacy Glass', '25'),

('Banner', '10'),

('Mural','25')

INSERT INTO ProdServ(ProductType, ProductPricehour)

VALUES

('Design', '100')

INSERT INTO ProdServ(ProductType, ProductPriceother)

VALUES

('Annual Report', '200'),

('Interior Sign', '150'),

('Wayfinding sign', '150')

SELECT \* FROM Customer --check results

-------

--Drop Function at beginning

--Function to retrieve CustomerID with BusinessName

GO

CREATE FUNCTION dbo.CustomerIDLookup(@BusinessName char(50))

RETURNS int AS -- CustomerID is int

BEGIN

DECLARE @returnValue int -- assigns CustomerID that matches business name temporarily to variable

SELECT @returnValue = CustomerID FROM Customer

WHERE BusinessName = @BusinessName

RETURN @returnValue --send back CustomerID

END

GO

SELECT dbo.CustomerIDLookup('Acme Corporation')

SELECT dbo.CustomerIDLookup('Globex Corporation')

SELECT dbo.CustomerIDLookup('Soylent Corp')

SELECT dbo.CustomerIDLookup('Initech')

SELECT dbo.CustomerIDLookup('Umbrella Corporation')

SELECT dbo.CustomerIDLookup('Hooli')

SELECT dbo.CustomerIDLookup('Pet Tricks')

SELECT \* FROM Employee --check results

-------

INSERT INTO JobOrder(CustomerID, OrderType, Date, Status, EmployeeID)

VALUES

('1', 'design', '01/01/2020', 'Closed','4'),

('1', 'design', '04/01/2020', 'Closed','4'),

('1', 'design', '07/01/2020', 'Closed','4'),

('1', 'print', '02/15/2020', 'Closed','4'),

('2', 'design', '04/04/2020', 'Closed','4'),

('2', 'design', '07/07/2020', 'Closed','4'),

('2', 'print', '02/20/2020', 'Closed','4'),

('2', 'design', '05/05/2020', 'Closed','4'),

('3', 'design', '08/01/2020', 'Closed','4'),

('3', 'design', '01/15/2020', 'Closed','4'),

('4', 'design', '05/01/2020', 'Closed','4'),

('4', 'design', '08/23/2020', 'Closed','4'),

('4', 'print', '02/15/2020', 'Closed','4'),

('5', 'design', '05/05/2020', 'Closed','4'),

('5', 'design', '01/15/2020', 'Closed','4'),

('6', 'print', '02/18/2020', 'Closed','4'),

('6', 'design', '05/15/2020', 'Closed','4'),

('6', 'design', '09/01/2020', 'Closed','4'),

('6', 'design', '03/14/2020', 'Closed','4'),

('7', 'print', '06/01/2020', 'Closed','4'),

('7', 'design', '09/09/2020', 'Closed','4')

-------

SELECT \* FROM JobOrder

--Adding Data to Table Invoice

INSERT INTO Invoice(Amount, Date, Status)

VALUES

('150', '03/01/2020', 'PAID'),--Acme

('2000', '06/01/2020', 'PAID'),--Acme

('1150', '09/01/2020', 'PAID'),--Acme

('200', '04/15/2020', 'PAID'),--Acme

('150', '06/04/2020', 'PAID'),--Globex

('1150', '09/07/2020', 'PAID'),--Globex

('200', '04/20/2020', 'PAID'),--Globex

('12000', '06/20/2020', 'PAID'),--Globex

('11400','09/01/2020', 'PAID'),--Soylent

('11500','02/15/2020', 'PAID'),--Soylent

('1150', '07/01/2020', 'PAID'),--Initech

('1500', '10/08/2020', 'PAID'),--Initech

('200', '04/15/2020', 'PAID'),--Initech

('17000', '06/05/2020', 'PAID'),--Umbrella

('2200', '02/15/2020', 'PAID'),--Umbrella

('200', '04/18/2020', 'PAID'),--Hooli

('1150', '07/15/2020', 'PAID'),--Hooli

('300', '11/01/2020', 'PAID'),--Hooli

('4000', '05/14/2020', 'PAID'),--Hooli

('23000', '07/01/2020', 'PAID'),--Pet Tricks

('61200', '10/09/2020', 'PAID')--Pet Tricks

SELECT \* FROM Invoice --check results

-------

--Drop Function at beginning

--Function to retrieve EmployeeID with Employee Name

GO

CREATE FUNCTION dbo.EmployeeIDLookup(@FirstName char(30))

RETURNS int AS --EmployeeID is int

BEGIN

DECLARE @returnValue int -- assigns EmployeeID that matches first name temporarily to variable

SELECT @returnValue = EmployeeID FROM Employee

WHERE FirstName = @FirstName

RETURN @returnValue --send back EmployeeID

END

GO

SELECT dbo.EmployeeIDLookup('Willie')

SELECT dbo.EmployeeIDLookup('Paige')

SELECT dbo.EmployeeIDLookup('Anita')

SELECT dbo.EmployeeIDLookup('Augusta')

SELECT dbo.EmployeeIDLookup('Saul')

SELECT \* FROM Employee--check results

-------

--Adding Data to Table EmployeePayment

INSERT INTO EmployeePayment (EmployeeID, Amount, Date)

VALUES

('1','2600', '02/01/2020'),

('1','2600', '5/01/2020'),

('1','2600', '9/01/2020'),

('2','3467', '02/01/2020'),

('2','3467', '5/01/2020'),

('2','3467', '9/01/2020'),

('3','12500', '02/01/2020'),

('3','12500', '5/01/2020'),

('3','12500', '9/01/2020'),

('4','8333', '02/01/2020'),

('4','8333', '5/01/2020'),

('4','8333', '9/01/2020'),

('5','7000','02/01/2020'),

('5','7000', '5/01/2020'),

('5','7000', '9/01/2020')

SELECT \* FROM EmployeePayment

SELECT \* FROM Invoice

SELECT \* FROM JobOrder

-------

--Adding Data to Table InvoiceOrderBridge

INSERT INTO InvoiceOrderBridge (InvoiceID, OrderID)

VALUES

('1','1'),

('2','2'),

('3','3'),

('4','4'),

('5','5'),

('6','6'),

('7','7'),

('8','8'),

('9','9'),

('10','10'),

('11','11'),

('12','12'),

('13','13'),

('14','14'),

('15','15'),

('16','16'),

('17','17'),

('18','18'),

('19','19'),

('20','20'),

('21','21')

SELECT \* FROM JobOrder

SELECT \* FROM ProdServ

-------

--Adding Data to Table OrderDetail

INSERT INTO OrderDetail (OrderID, ProdServID, DescriptionofRequest, Deadline, Priority)

VALUES

('1','9', 'Interior wayfinding sign, branded- armory, 1x2', '03/01/2020', 'LOW'),

('2','1', 'single wall wallpaper graphic, 8x10', '06/01/2020', 'LOW'),

('3','1', 'branded HR poster, 2x3', '09/01/2020', 'LOW'),

('4','7', 'annual report layout, no design', '04/15/2020', 'LOW'),

('5','9', 'Interior wayfinding sign, branded- cafeteria, 1x2', '06/04/2020', 'LOW'),

('6','1', 'branded HR poster, 2x3', '09/07/2020', 'LOW'),

('7','7', 'annual report layout, no design', '04/20/2020', 'LOW'),

('8','1', '10x Custom wallpaper pattern, trendy vintage, 8x10 ea', '06/20/2020', 'MED'),

('9','3', 'Privacy glass with custom science graphic, 8x32','09/01/2020','RED FLAG'),

('10','9', '70x Interior wayfinding signs, custom design- restroom, 1x2','02/15/2020','RED FLAG'),

('11','1', 'branded HR poster, 2x3', '07/01/2020', 'LOW'),

('12','9', '10x Interior wayfinding signs, branded- restroom, 1x2', '10/08/2020', 'MED'),

('13','7', 'annual report layout, no design', '04/15/2020', 'LOW'),

('14','1', '5x wall wallpaper graphics, original art mural, 8x12 ea', '06/05/2020', 'RED FLAG'),

('15','7', 'annual report layout, custom', '02/15/2020', 'RED FLAG'),

('16','7', 'annual report layout, no design', '04/18/2020', 'LOW'),

('17','1', 'branded poster, 2x3', '07/15/2020', 'LOW'),

('18','9', '2x Interior wayfinding signs, branded- cafeteria, 1x2', '11/01/2020', 'LOW'),

('19','1', 'single wall wallpaper graphic, key art, 8x10', '05/14/2020', 'MED'),

('20','1', '20x custom branded posters, new-releases key art, 2x3', '07/01/2020', 'RED FLAG'),

('21','1', '18x wallpaper graphics, award-winners key art, 8x12 ea', '10/09/2020', 'RED FLAG')

--

--Invoke Function to retrieve EmployeeID with Employee Name

SELECT dbo.EmployeeIDLookup('Willie')

--Update an Employee Address

UPDATE Employee SET Address = '1100 Fruit Bat Way' WHERE EmployeeID = 1

SELECT \* FROM Employee WHERE FirstName = 'Willie'

------

--Invoke Function to retrieve CustomerID with Business Name

SELECT dbo.CustomerIDLookup('Acme Corporation')

--Update a Customer POC

UPDATE Customer SET POCFirstName ='Wile.E' WHERE CustomerID = 1

UPDATE Customer SET POCLastName ='Coyote' WHERE CustomerID = 1

UPDATE Customer SET EmailAddress ='WECoyote@Acme.com' WHERE CustomerID = 1

UPDATE Customer SET StartDate ='12/31/1999' WHERE CustomerID = 1

--Check Records

SELECT \* FROM Customer WHERE BusinessName = 'Acme Corporation'

------

--Invoke Function to retrieve EmployeeID with Employee Name

SELECT dbo.EmployeeIDLookup('Willie')

--Update an Employee Phone Number

UPDATE Employee SET PhoneNum = '916-444-9860' WHERE EmployeeID = 1

SELECT \* FROM Employee WHERE FirstName = 'Willie'

--------

--------

--STORED PROCEDURES

--------

--------

--Add a Product(stored procedure)

--DROP PROCEDURE IF EXISTS AddProduct

--GO

--Drop procedure at beginning

GO

CREATE PROCEDURE AddProduct(@ProductType char(30), @ProductPriceother int)

AS

BEGIN

INSERT INTO ProdServ(ProductType, ProductPriceother)

VALUES (@ProductType, @ProductPriceother)

END

GO

EXEC AddProduct 'Custom Annual Report', '1200'

EXEC AddProduct 'Custom Annual Report', ''

EXEC AddProduct 'Custom Poster', '1150'

EXEC AddProduct 'Custom Wayfinding sign', '1150'

--------

--Delete a product(stored procedure)

--DROP PROCEDURE IF EXISTS DeleteProduct

--GO

--Drop procedure at beginning

GO

CREATE PROCEDURE DeleteProduct(@ProductType char(30), @ProductPriceother int)

AS

BEGIN

DELETE ProdServ

WHERE ProductPriceother = @ProductPriceother

END

GO

EXEC DeleteProduct 'Custom Annual Report', '1200'

--------

--Update a product(stored procedure)

--DROP PROCEDURE IF EXISTS UpdateProduct

--GO

--Drop procedure at beginning

GO

CREATE PROCEDURE UpdateProduct(@ProductType char(30), @ProductPriceother int)

AS

BEGIN

UPDATE ProdServ SET ProductPriceother = @ProductPriceother

WHERE ProductType = @ProductType

END

GO

EXEC UpdateProduct 'Custom Annual Report','1200'

SELECT \* FROM ProdServ --check results

-------

SELECT \* FROM Customer WHERE BusinessName = 'Initech'

SELECT \* FROM JobOrder WHERE CustomerID = 4

SELECT \* FROM OrderDetail WHERE OrderID = 11

SELECT \* FROM OrderDetail WHERE OrderID = 12

SELECT \* FROM OrderDetail WHERE OrderID = 13

-------

--Update OrderDetail Priority(stored procedure)

--DROP IF EXISTS PROCEDURE UpdatePriority

--GO

--Drop procedure at beginning

GO

CREATE PROCEDURE UpdatePriority(@Priority char(10), @OrderID int)

AS

BEGIN

UPDATE OrderDetail SET Priority = @Priority

WHERE OrderID = @OrderID

END

GO

EXEC UpdatePriority 'MED', '11'

SELECT \* FROM OrderDetail WHERE OrderID = '11' --check results

--------

--Update Invoice when status changes- SENT, PAID, UNPAID(stored procedure)

--DROP PROCEDURE IF EXISTS UpdateInvoice

--GO

--Drop procedure at beginning

GO

CREATE PROCEDURE UpdateInvoice(@Status char(15),@InvoiceID int)

AS

BEGIN

UPDATE Invoice SET Status = @Status

WHERE InvoiceID = @InvoiceID

END

GO

EXEC UpdateInvoice 'UNPAID','1'

SELECT \* FROM Invoice --check results

-------

SELECT \* FROM Customer WHERE BusinessName = 'Acme Corporation'

--Update JobOrder when Invoice OPEN(UNPAID) or CLOSED(PAID)(stored procedure)

--DROP PROCEDURE IF EXISTS UpdateJobOrder

--GO

--Drop procedure at beginning

GO

CREATE PROCEDURE UpdateJobOrder(@Status char(10),@OrderID int)

AS

BEGIN

UPDATE JobOrder SET Status = @Status

WHERE OrderID = @OrderID

END

GO

EXEC UpdateJobOrder 'Open','1'

SELECT \* FROM JobOrder --check results

-------

--Invoke Function to retrieve EmployeeID with Employee Name

SELECT dbo.EmployeeIDLookup('Paige')

-------

--DROP PROCEDURE IF EXISTS DeleteEmployee

--GO

--Drop procedure at beginning

GO

CREATE PROCEDURE DeleteEmployee(@EmployeeID int)

AS

BEGIN

DELETE EmployeePayment

WHERE EmployeeID = @EmployeeID;

DELETE Employee

WHERE EmployeeID = @EmployeeID;

END

GO

EXEC DeleteEmployee '2'

SELECT \* FROM EMPLOYEE --check employee records

-------

--Select statement for monthly payroll after firing employee

SELECT

Employee.FirstName,

Employee.LastName,

SUM(Amount) as Payroll

FROM EmployeePayment

JOIN Employee ON Employee.EmployeeID = EmployeePayment.EmployeeID

WHERE EmployeePayment.Date = '2020-09-01'

GROUP BY

Employee.FirstName,

Employee.LastName

ORDER BY Employee.FirstName

-------

-------

--Views

-------

-------

--Create a View for monthly payroll

--DROP VIEW IF EXISTS MonthlyPayroll

--GO

--Drop view at beginning

GO

CREATE VIEW MonthlyPayroll AS

SELECT

Employee.FirstName,

Employee.LastName,

SUM(Amount) as Payroll,

EmployeePayment.Date

FROM EmployeePayment

JOIN Employee ON Employee.EmployeeID = EmployeePayment.EmployeeID

WHERE EmployeePayment.Date = '2020-09-01'

GROUP BY

Employee.FirstName,

Employee.LastName,

EmployeePayment.Date

GO

SELECT \* FROM MonthlyPayroll

-------

--Create a View for current open invoices

--DROP VIEW IF EXISTS CurrentOpenInvoices

--GO

--Drop at beginning

GO

CREATE VIEW CurrentOpenInvoices AS

SELECT

JobOrder.OrderID,

JobOrder.CustomerID,

Invoice.Amount,

Invoice.Date,

Customer.BusinessName

FROM

JobOrder

JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID

JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID

JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID

WHERE Invoice.Status = 'UNPAID'

GROUP BY

JobOrder.OrderID,

JobOrder.CustomerID,

Invoice.Amount,

Invoice.Date,

Customer.BusinessName

GO

SELECT \* FROM CurrentOpenInvoices

-------

--Create a View for current Closed invoices

--DROP VIEW IF EXISTS CurrentClosedInvoices

--GO

--Drop at beginning

GO

CREATE VIEW CurrentClosedInvoices AS

SELECT

JobOrder.OrderID,

JobOrder.CustomerID,

Invoice.Amount,

Invoice.Date,

Customer.BusinessName

FROM

JobOrder

JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID

JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID

JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID

WHERE Invoice.Status = 'PAID'

GROUP BY

JobOrder.OrderID,

JobOrder.CustomerID,

Invoice.Amount,

Invoice.Date,

Customer.BusinessName

GO

SELECT \* FROM CurrentClosedInvoices

-------

-------

--Additional select statements for analysis

-------

-------

--SELECT customers with CLOSED orders

SELECT

JobOrder.OrderID,

JobOrder.CustomerID,

Customer.BusinessName

FROM

JobOrder

JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID

JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID

JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID

WHERE JobOrder.Status = 'CLOSED'

GROUP BY

JobOrder.OrderID,

JobOrder.CustomerID,

Customer.BusinessName

ORDER BY Customer.BusinessName

--------

--SELECT CLOSED Orders with time per order in days

SELECT

JobOrder.OrderID,

Invoice.InvoiceID,

Customer.CustomerID,

Customer.BusinessName,

((AVG(DateDiff(n, JobOrder.Date, Invoice.Date))/60)/24) as AvgOrderTimeinDays

FROM

JobOrder

JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID

JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID

JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID

WHERE JobOrder.Status = 'CLOSED'

GROUP BY

JobOrder.OrderID,

Invoice.InvoiceID,

Customer.CustomerID,

Customer.BusinessName

ORDER BY Customer.BusinessName

-------

--SELECT customers with OPEN orders

SELECT

JobOrder.OrderID,

JobOrder.CustomerID,

Customer.BusinessName

FROM

JobOrder

JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID

JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID

JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID

WHERE JobOrder.Status = 'OPEN'

GROUP BY

JobOrder.OrderID,

JobOrder.CustomerID,

Customer.BusinessName

ORDER BY Customer.BusinessName

-------

--SELECT OPEN Orders with time per order in days

SELECT

JobOrder.OrderID,

Invoice.InvoiceID,

Customer.CustomerID,

Customer.BusinessName,

((AVG(DateDiff(n, JobOrder.Date, Invoice.Date))/60)/24) as AvgOrderTimeinDays

FROM

JobOrder

JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID

JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID

JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID

WHERE JobOrder.Status = 'OPEN'

GROUP BY

JobOrder.OrderID,

Invoice.InvoiceID,

Customer.CustomerID,

Customer.BusinessName

ORDER BY Customer.BusinessName

------

--Customer order information summary

SELECT

Customer.BusinessName,

SUM(Amount) as SumGrossRevenue,

MIN(Amount) as MinGrossRevenue,

AVG(Amount) as AvgGrossRevenue,

MAX(Amount) as MaxGrossRevenue

FROM

JobOrder

JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID

JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID

JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID

WHERE Invoice.Status = 'PAID'

GROUP BY

Customer.BusinessName

ORDER BY Customer.BusinessName

-------

--Amount spent per order by customer

SELECT

JobOrder.OrderID,

Customer.BusinessName,

SUM(Amount) as AmountSpentPerOrder

FROM

JobOrder

JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID

JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID

JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID

WHERE Invoice.Status = 'PAID'

GROUP BY

JobOrder.OrderID,

Customer.BusinessName

ORDER BY Customer.BusinessName

-------

-------

-------

-------

-------

-------

-------DATA QUESTIONS

-------

-------

-------

----

----

----

--Data Question #1 What is the total amount spent per Customer?

--DROP VIEW IF EXISTS TotalPerCustomer

--GO

--Drop at beginnning

--Create view for total spent per customer

GO

CREATE VIEW TotalPerCustomer AS

SELECT

Customer.CustomerID,

Customer.BusinessName,

SUM(Amount) as TotalOrderPerCustomer

FROM

JobOrder

JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID

JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID

JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID

WHERE Invoice.Status = 'PAID'

GROUP BY

Customer.CustomerID,

Customer.BusinessName

GO

SELECT \* FROM TotalPerCustomer

----

--Data Question #2 What is the average amount spent per Customer Order?

--DROP VIEW IF EXISTS AvgPerOrder

--GO

--Drop at beginnning

--Create view for average spent per order

GO

CREATE VIEW AvgPerOrder AS

SELECT

Customer.CustomerID,

Customer.BusinessName,

AVG(Amount) as AvgOrderPerCustomer

FROM

JobOrder

JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID

JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID

JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID

WHERE Invoice.Status = 'PAID'

GROUP BY

Customer.CustomerID,

Customer.BusinessName

GO

SELECT \* FROM AvgPerOrder

----

--Data Question #3 Who are the Top Performing Customers?

--Identify the top customers by invoice revenue

SELECT

TOP 3 BusinessName,

Customer.CustomerID,

Customer.BusinessName,

SUM(Amount) as TotalOrderPerCustomer

FROM

JobOrder

JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID

JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID

JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID

WHERE Invoice.Status = 'PAID'

GROUP BY

Customer.CustomerID,

Customer.BusinessName

ORDER BY SUM(Amount) DESC

----

--Data Question #4 Who are the Lowest Performing Customers?

--Identify the lowest performing customers by invoice revenue

SELECT

TOP 3 BusinessName,

Customer.CustomerID,

Customer.BusinessName,

SUM(Amount) as TotalOrderPerCustomer

FROM

JobOrder

JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID

JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID

--JOIN InvoiceOrderBridge ON Invoice.InvoiceID = InvoiceOrderBridge.InvoiceID

--JOIN JobOrder ON InvoiceOrderBridge.OrderID = JobOrder.OrderID

JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID

WHERE Invoice.Status = 'PAID'

GROUP BY

Customer.CustomerID,

Customer.BusinessName

ORDER BY SUM(Amount) ASC

----

--Data Question #5 What is the monthly revenue for all Customers?

--DROP VIEW IF EXISTS MonthlyRevenue

--GO

--Drop at beginnning

--Create view for monthly revenue, borrowed snippets from sample

GO

CREATE VIEW MonthlyRevenue AS

SELECT

DATEADD(month, DATEDIFF(month, 0, Date),0) 'MonthBegins',

SUM(Amount) as MonthlyProfit

FROM

Invoice

GROUP BY

DATEADD(month, DATEDIFF(month, 0, Date),0)

GO

SELECT \* FROM MonthlyRevenue

----

--Data Question #6 What is the total revenue for all Customers?

--DROP VIEW IF EXISTS TotalRevenue

--GO

--Drop at beginnning

--Create view for total revenue, borrowed snippets from sample

GO

CREATE VIEW TotalRevenue AS

SELECT

SUM(Amount) as TotalProfit,

GETDATE() as RangeDate

FROM

Invoice

WHERE Invoice.Status = 'PAID'

GO

SELECT \* FROM TotalRevenue

----

--Data Question #7 What is the Order Count by Customer?

SELECT

Customer.BusinessName,

Customer.StartDate,

Customer.POCFirstName,

Customer.POCLastName,

Customer.BusinessAddress,

Customer.BusinessCity,

Customer.BusinessState,

Customer.BusinessZipcode,

Customer.EmailAddress,

Customer.PhoneNum,

Customer.CustomerID,

COUNT(JobOrder.OrderID) as TotalOrders

FROM

JobOrder

JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID

JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID

JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID

WHERE Invoice.Status = 'PAID'

GROUP BY

Customer.BusinessName,

Customer.StartDate,

Customer.POCFirstName,

Customer.POCLastName,

Customer.BusinessAddress,

Customer.BusinessCity,

Customer.BusinessState,

Customer.BusinessZipcode,

Customer.EmailAddress,

Customer.PhoneNum,

Customer.CustomerID

ORDER BY TotalOrders DESC, Customer.BusinessName

------

--Data Question # 8 What is the relationship between order price and time spent on order?

--SELECT CLOSED Orders with time per order in days

SELECT

JobOrder.OrderID,

Invoice.InvoiceID,

Invoice.Amount,

Customer.BusinessName,

((AVG(DateDiff(n, JobOrder.Date, Invoice.Date))/60)/24) as OrderTimeinDays

FROM

JobOrder

JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID

JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID

JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID

WHERE JobOrder.Status = 'CLOSED'

GROUP BY

JobOrder.OrderID,

Invoice.InvoiceID,

Invoice.Amount,

Customer.BusinessName

ORDER BY Invoice.Amount DESC

-------

--Data Question # 9 What is the relationship between order price, time per order, and a High-Priority Customer??

--SELECT Highest CLOSED Orders with time per order in days

SELECT

TOP 10 BusinessName,

Invoice.Amount,

((AVG(DateDiff(n, JobOrder.Date, Invoice.Date))/60)/24) as AvgOrderTimeinDays

FROM

JobOrder

JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID

JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID

JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID

WHERE JobOrder.Status = 'CLOSED'

GROUP BY

Invoice.Amount,

Customer.BusinessName

ORDER BY Invoice.Amount DESC

-------

--Data Question # 10 What is the relationship between order price, time per order, and a Low-Priority Customer??

--SELECT Highest CLOSED Orders with time per order in days

SELECT

TOP 10 BusinessName,

Invoice.Amount,

((AVG(DateDiff(n, JobOrder.Date, Invoice.Date))/60)/24) as AvgOrderTimeinDays

FROM

JobOrder

JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID

JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID

JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID

WHERE JobOrder.Status = 'CLOSED'

GROUP BY

Invoice.Amount,

Customer.BusinessName

ORDER BY Invoice.Amount ASC

-------