A Case Study of Hancock Graphics

Chileen Duncan

IST 659

Project Deliverable 2

3/29/2021

Summary

Hancock Graphics is a printing business located in Sacramento, CA. They originally opened their doors in the 1950s as a reprographics firm, supplying blueprints to the construction industry. In the early 2000s, they bought a small color printing firm, and began to expand into color printing and digital graphics, turning their focus away from blueprints, a dying industry. Fast forward a few years, and now they are the sole supplier of custom printed wallpapers and window graphics for the office buildings of a leading streaming service, Pet Tricks. The business also provides standard graphic design services and general printing, and has many other lower-level clients.

The flow of this business is very straightforward, but has many participants involved. There is currently a tracking system for orders, and a tracking system for invoices, but there is a disconnect between the two, and employees have reported that this creates two major issues. The first is that jobs get “lost” and not worked on for a long time, because no one person is actually accountable for any one job, only what is currently passing through their department. The second issue is that employees are not feeling as if they are accurately valued for their contributions to jobs with very high sticker prices, and there is no quantitative way to justify demands for a pay increase. In addition, because there are so many participants involved, and higher priority jobs are paid more attention by the sales team, it is often that lower priority jobs slip through the cracks, and are not worked on for months. This is a problem, as employees report that lower priority jobs are much higher in volume, and account for a high percentage of revenue.

We propose to implement a database that will more efficiently match the actual workflow of The Company, rather than simply assigning arbitrary order numbers that serve a limited purpose, and will fill the need caused by not having a complete tracking system in place. Our system will help identify jobs that are flying under the radar of the sales team because they are deemed less worth their time, but that are actually very important to the other stakeholders due to their sheer volume and impact on revenue. It

will also help quantify the work of employees, especially the design department, who is the most responsible for customer retention, and legitimize employee contributions to high-value orders. Finally, our system will help identify customer status as regular or high-priority based on generated revenue, not a high-profile name or perceived importance.

This proposal will generally outline the database concepts and logical model, while defining the players involved and the terminology used.

Privacy film window graphic recently designed for Soylent Corp

The Stakeholders

Hancock Graphics is a small business, with just under 50 employees. The people who work there are like a family, for better or worse, and retention is high, although morale is often very low, as many feel under-appreciated and undervalued. The stakeholders in this case study include: the owner, the employees, the sales team, the customers, and a potential partner, Cracked Cow Studios, who often works as a subcontractor for The Company, but has built a substantial client list of their own. For the purpose of this study, we decided to treat the sales employees as a separate group, due to their high level of influence on operations, although they are not owners.

The implementation of this database will help streamline the order process, which will allow for more orders to be in progress at once, and increase the rate of invoicing. This will lead to higher profits for the owner and the sales team, who work on commission. The ability to better understand an individual employee's contribution to an order will support the employees in pay negotiations, and legitimize their importance to management. This will increase morale and retention. The customers will benefit from the new system by faster turnaround on their orders, which will lead to more orders, and more profit. The potential partner, Cracked Cow Studios, is a small, but successful, graphic design start-up. They are interested in becoming a larger player in the custom interiors space, and already work with many of The Company's customers, to include Pet Tricks, on their own projects. They have reservations about the age of the company, and its ability to change with the times. The implementation of this new system, and a commitment to progress and growth would help instill confidence in this budding partnership.

Business Rules

1. A customer request creates an order
2. A salesperson can have one or more customer
3. A salesperson takes an order
4. A salesperson passes an order to the correct department head
5. The department head passes the complete order to the customer
6. An employee can only have one job
7. An employee is paid once a month for their hours per pay rate
8. An order can pass between a department head and a customer one or more times due to proofing
9. An order contains order details
10. An order detail can contain one or more product/service requests, but they are individually unique.
11. An order status updates the order
12. An order date is included in the order
13. An order type is included in an order detail
14. Priority is included in the order detail.
15. An order can jump priority levels in both directions.
16. A description of the requested products/services is included in the order detail
17. An order deadline is included in the order detail
18. An order deadline can change
19. An order can be cancelled
20. An order ends with the invoice
21. An order can only have one invoice
22. An invoice can pay more than one order
23. A customer can have one or more invoice
24. A customer can have one or more order
25. An invoice status updates the invoice

Glossary

Customer- A person who requests a product or service provided by The Company. Can also be called a Client.

Employee- Any person, to include department heads and the sales team, employed by the company.

High-priority Customer- A customer with a high profile company, or a customer that regularly place orders of a high value. Approximately $10,000 and over per job, or $100,000 or more per year.

Invoice- The bill sent to the customer after the product or service has been delivered or installed.

Order- A detailed request by a customer for a product or service

Order Detail- Requested product or service, detailed description of request, preferred deadline of deliverables, the amount quoted by the salesperson, and the current status of the order.

Order Status- The status for an order is IP(in progress), OFP(out for proof), PRO(in production), NTB(need to bill), INVOICED(bill sent to customer) and CLOSED(invoice paid).

Order Type- This categorizes the product/service into DESIGN ONLY, PRINT ONLY, and DESIGN/PRINT.

Priority- This categorizes a product/service into RED FLAG, HOT, WARM, and COLD.

Products/Services- These are items The Company can provide to a customer for payment. This list is constantly evolving as customers request custom items and new products and innovation with existing products emerge.

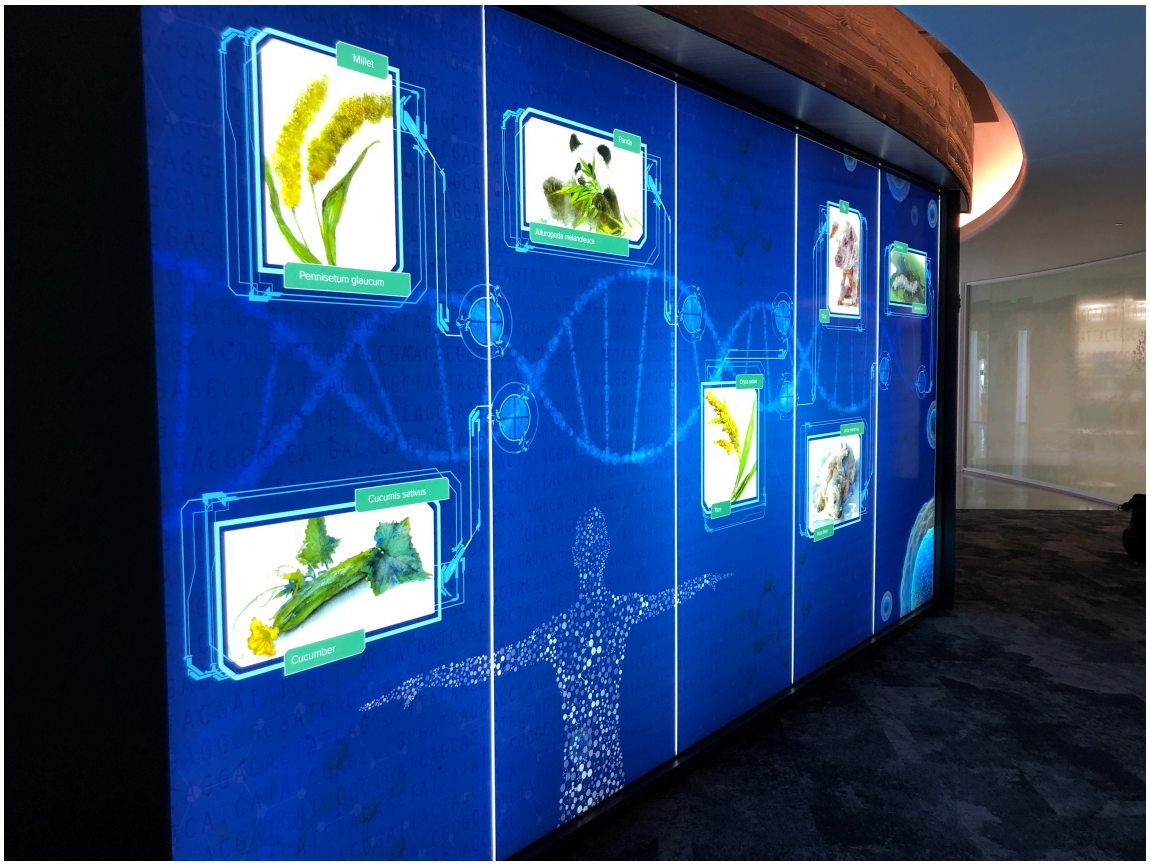
Red Flag- This denotes a rush job, and is placed inside a red folder.

Salesperson- An employee who maintains a direct relationship with customers, takes orders directly from customers, and is responsible for assuring customers are satisfied after orders are delivered.

The Company- This term will be used to refer to Hancock Graphics for the duration of this case study.

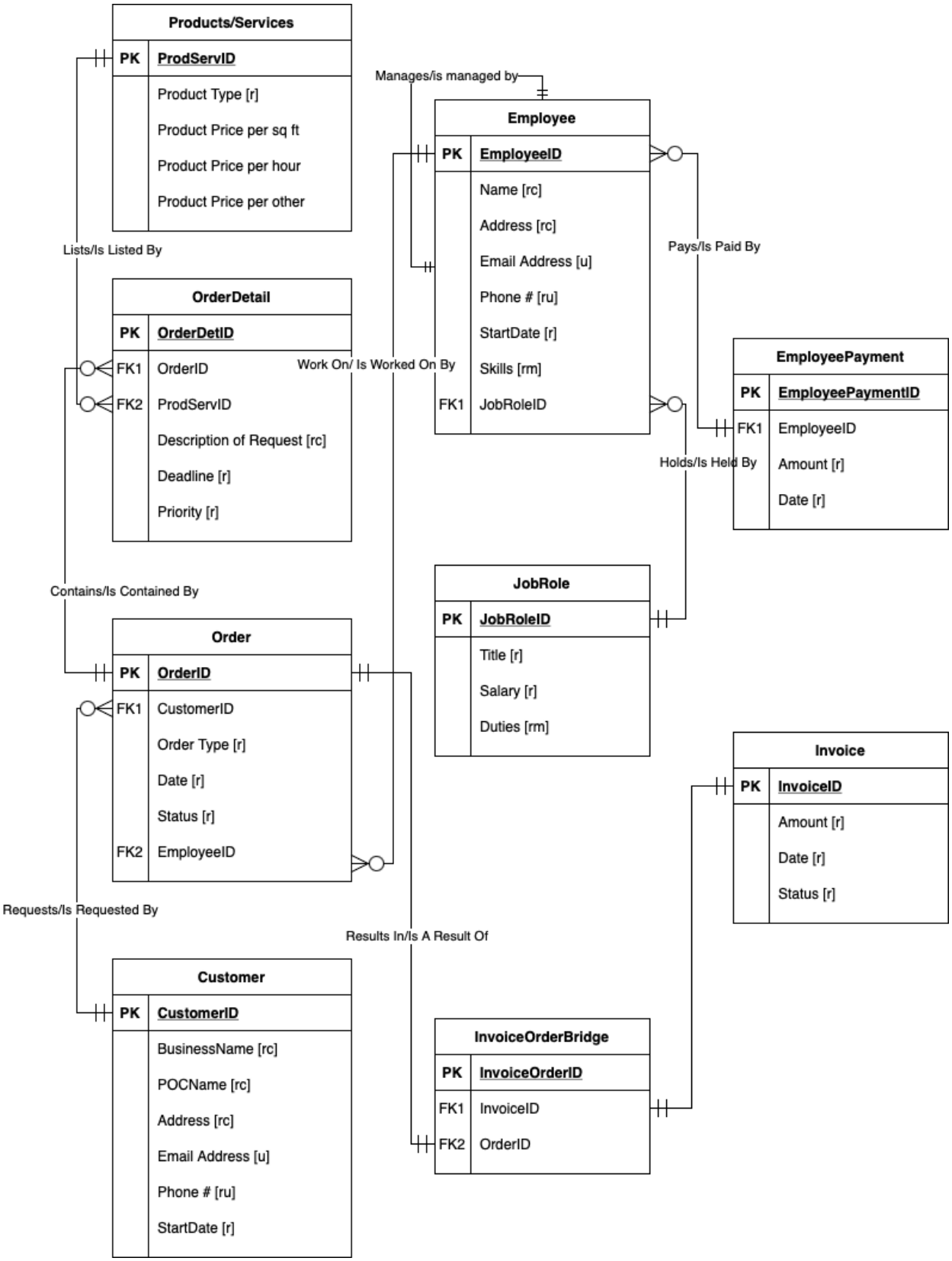
Data Questions

- 1. What is the total amount spent per customer?
- 2. What is the average amount spent per customer order?
- 3. Who are the top performing customers?
- 4. Who are the lowest performing customers
- 5. What is the monthly revenue for all customers?
- 6. What is the total revenue for all customers?
- 7. What is the order count by customer?
- 8. What is the relationship between order price and time spent on order?
- 9. What is the relationship between order price, time per order, and High-priority customers?
- 10. What is the relationship between order price, time per order, and Low-priority customers?

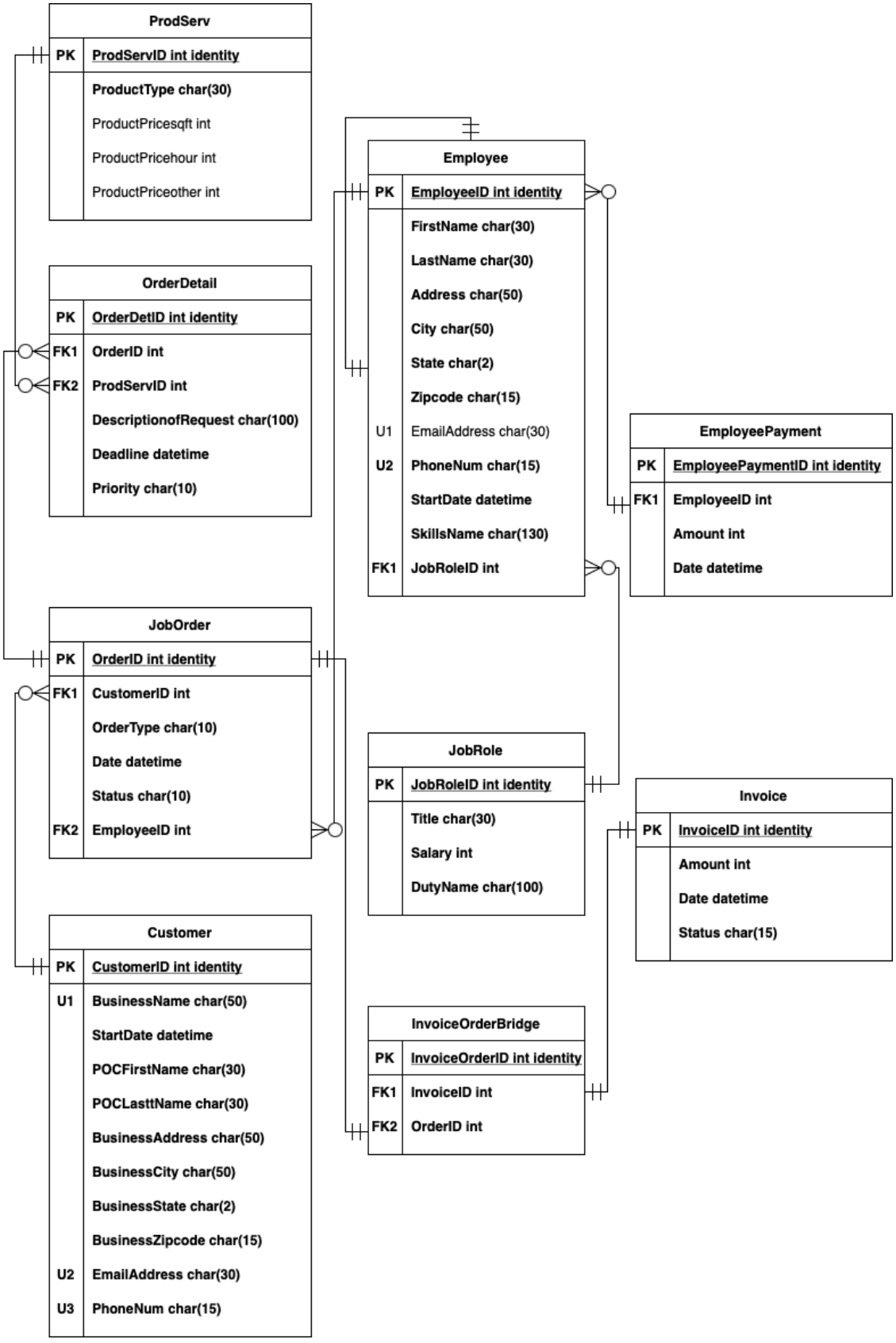


LED backlit window graphic recently designed for Soylent Corp

Conceptual Model



Logical Model



Data Samples

ORDER NUMBER	PAYMENT TYPE
	☐ ACCOUNT

Hancock Graphics

2420 X Street, Sacramento California 95818

P 916 454 CMYK • F 916 454 2616

www.Hancock.com

CUSTOMER NAME .....

ADDRESS .....

CITY ..... STATE ..... ZIP .....

CONTACT ..... PHONE .....

DATE IN ..... TIME .....

PROOF DUE ..... TIME .....

FINAL DUE ..... TIME .....

☐ DELIVERY ☐ CALL WHEN DONE .....

☐ SHIPPING TO ARRIVE BY :

DATE ..... TIME .....

☐ USE CUSTOMER SHIPPING ACCOUNT

.....

☐FedEx ☐UPS ☐DHL ☐GSO

SHIPPING IF DIFFERENT

CUSTOMER NAME .....

ADDRESS .....

CITY ..... STATE ..... ZIP .....

CONTACT ..... PHONE .....

.....

PROJECT NAME / NUMBER .....

TRANS.
- ☐ Mac
- ☐ PC
- ☐ Email
- ☐ FTP
- ☐ CD
- ☐ Orig
- ☐ Other .....

SOFTWARE
- ☐ Photoshop .....
- ☐ Illustrator .....
- ☐ InDesign .....
- ☐ Acrobat .....
- ☐ Word .....
- ☐ Excel .....
- ☐ Other .....

PAGE INFORMATION

☐ 8.5 x 11 ☐ 11 x 17 ☐ 12 x 18
☐ 18 x 24 ☐ 24 x 36 ☐ 30 x 42

☐ Custom Size ..... x .... x .....
☐ Folded Size ..... x ..... ☐ Folds .....

☐ Dbl Sided ☐ Bleeds ☐ Crops

QUANTITY

File Name	Orig	Sets

Note Special Instructions on Back

OUTPUT

Printer / Media

Small Document Printing
- ☐ Docucolor
- ☐ Docutech

☐ 20# T ☐ 24# T ☐ 90# I
☐ 100# T Gloss ☐ 100# C Silk ☐ .....

Large Document Prinitng
- ☐ UV

☐ Wallpaper ☐ Canvas ☐ Privacy
☐ Matte Bond ☐ Gloss Bond ☐ Film

Outdoor Printing
- ☐ Solvent
- ☐ UV

☐ Adh Vinyl ☐ Banner ☐ Paper
☐ Auto Vinyl ☐ Canvas ☐ .....

SPECIAL FINISHING

Binding
☐ 3 Hole ☐ 2 Hole
☐ Staple ☐ Saddle Staple
☐ Coil ☐ Comb
☐ Padding ☐ .....

Folding
☐ Half Fold ☐ Tri Fold
☐ Z Fold ☐ .....

Finishing
☐ Trim Only ☐ Shrink Wrap
☐ Pack for Shipping
☐ Edge Trim ☐ Wht ☐ Blk ☐ .....
☐ Velcro ☐ Back Hanger

MOUNTING AND LAMINATING

Lamination

Encapsulation
☐ Gloss 3mil ☐ Matte 3 mil ☐ Gloss 5mil ☐ Matte 5mil
☐ Gloss 10mil ☐ Display ☐ Floor ☐ Decal
☐ Standard Lip ☐ No Lip

Surface
☐ Gloss ☐ Matte ☐ Luster ☐ Textured
☐ Super Gloss ☐ .....

Mounting

Foam Board	Gator Board	Plastics	Other
☐ White	☐ White	☐ Coroplast	☐ Di Bond
☐ Black	☐ Black	☐ Clear Plexi	☐ MDO
☐ 3/16"	☐ 3/16"	☐ Non Glare	☐ Wood
☐ 1/2"	☐ 1/2"	☐ Milk	☐ Provided

☐ .....

ORDER NUMBER 774395 | PAYMENT TYPE ☒ ACCOUNT | RED

Augusta

CUSTOMER NAME * Soylent *

ADDRESS

C 774395 ZIP

C PHONE

* Find me for info !!

SHIPPING IF DIFFERENT

CUSTOMER NAME Soylent Corp.

ADDRESS 777 S Peoples St

CITY Los Angeles STATE CA ZIP 90021

CONTACT Bill Simonson PHONE (903) 580-2339

BS@Soylent.com

PROJECT NAME / NUMBER

Hancock Graphics

2420 X Street, Sacramento California 95818

P 916 454 CMYK • F 916 454 2616

www.Hancock.com

DATE IN 8/1/2020 TIME 12:30pm

PROOF DUE TIME

FINAL DUE ASAP ! TIME

☒ DELIVERY ☐ CALL WHEN DONE

☒ SHIPPING TO ARRIVE BY :

DATE TIME

☐ USE CUSTOMER SHIPPING ACCOUNT

#

☐ FedEx ☒ UPS ☐ DHL ☐ GSO

TRANS.
- ☐ Mac
- ☐ PC
- ☐ Email
- ☐ FTP
- ☐ CD
- ☐ Orig
- ☐ Other

SOFTWARE
- ☐ Photoshop
- ☐ Illustrator
- ☐ InDesign
- ☐ Acrobat
- ☐ Word
- ☐ Excel
- ☐ Other

PAGE INFORMATION
- ☐ 8.5 x 11 ☐ 11 x 17 ☐ 12 x 18
- ☐ 18 x 24 ☐ 24 x 36 ☐ 30 x 42
- ☐ Custom Size x x
- ☐ Folded Size x ☐ Folds
- ☐ Dbl Sided ☐ Bleeds ☐ Crops

QUANTITY

File Name	Orig	Sets

Note Special Instructions on Back

OUTPUT * See notes ! ♡ Augusta

Printer

Media

Small Document Printing
- ☐ Docucolor
- ☐ Docutech

☐ 20# T ☐ 24# T ☐ 90# T
☐ 100# T Gloss ☐ 100# C Silk ☐

Large Document Prinitng
- ☐ UV

☐ Wallpaper ☐ Canvas ☒ Privacy
☐ Matte Bond ☐ Gloss Bond ☐ Film

Outdoor Printing
- ☐ Solvent
- ☐ UV

☐ Adh Vinyl ☐ Banner ☐ Paper
☐ Auto Vinyl ☐ Canvas ☐

SPECIAL FINISHING

Binding
- ☐ 3 Hole ☐ 2 Hole
- ☐ Staple ☐ Saddle Staple
- ☐ Coil ☐ Comb
- ☐ Padding ☐

Folding
- ☐ Half Fold ☐ Tri Fold
- ☐ Z Fold ☐

Finishing
- ☐ Trim Only ☐ Shrink Wrap
- ☐ Pack for Shipping
- ☐ Edge Trim ☐ Wht ☐ Blk ☐
- ☐ Velcro ☐ Back Hanger

MOUNTING AND LAMINATING

Lamination

Encapsulation
- ☐ Gloss 3mil ☐ Matte 3 mil ☐ Gloss 5mil ☐ Matte 5mil
- ☐ Gloss 10mil ☐ Display ☐ Floor ☐ Decal
- ☐ Standard Lip ☐ No Lip

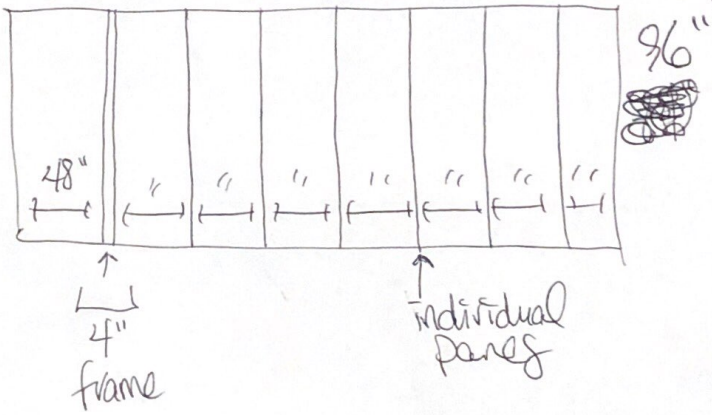
Surface
- ☐ Gloss ☐ Matte ☐ Luster ☐ Textured
- ☐ Super Gloss ☐

Mounting

Foam Board	Gator Board	Plastics	Other
☐ White	☐ White	☐ Coroplast	☐ Di Bond
☐ Black	☐ Black	☐ Clear Plexi	☐ MDO
☐ 3/16"	☐ 3/16"	☐ Non Glare	☐ Wood
☐ 1/2"	☐ 1/2"	☐ Milk	☐ Provided
☐			

* Big Client !!!

- client wants something "science-y" opaque
- privacy film w/ graphic = frosted
- curvy glass wall



Quote $10k

let me know if you have questions

Angela

Database Design

```
-- 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

-------
```

```
160  -------
161
162  --Begin Creating the Customer Table
163  GO
164  CREATE TABLE Customer(
165  --Columns for the Customer table
166  CustomerID int identity,
167  BusinessName char(50) not null,
168  StartDate datetime not null,
169  POCFirstName char(30) not null,
170  POCLastName char(30) not null,
171  BusinessAddress char(50) not null,
172  BusinessCity char(50) not null,
173  BusinessState char(2) not null,
174  BusinessZipcode char(15) not null,
175  EmailAddress char(30) not null,
176  PhoneNum char(15) not null,
177  --Constraints on the Customer Table
178  CONSTRAINT PK_Customer PRIMARY KEY (CustomerID),
179  CONSTRAINT U1_Customer UNIQUE (BusinessName),
180  CONSTRAINT U2_Customer UNIQUE (EmailAddress),
181  CONSTRAINT U3_Customer UNIQUE (PhoneNum)
182  )
183  GO
184  -- End Creating Customer Table
185
186  -------
187
188  --Begin Creating the JobOrder Table
189  GO
190  CREATE TABLE JobOrder(
191  --Columns for the JobOrder table
192  OrderID int identity,
193  CustomerID int not null,
194  OrderType char(10) not null,
195  Date datetime not null default GetDate(),
196  Status char(10) not null,
197  EmployeeID int not null,
198  --Constraints on the JobOrder Table
199  CONSTRAINT PK_JobOrder PRIMARY KEY (OrderID),
200  CONSTRAINT FK1_JobOrder FOREIGN KEY (CustomerID) REFERENCES Customer(CustomerID),
201  CONSTRAINT FK2_JobOrder FOREIGN KEY (EmployeeID) REFERENCES Employee(EmployeeID)
202  )
203  GO
204  -- End Creating JobOrder Table
205
206  -------
```

```
-------

--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
```

INSERT Statements

```
--------
-------
--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, Prod
        ('Paige','Turner','1181 Deer Ridge Drive','Sacramento','CA','95822','MoneyMama@gmail.com','916-404-4380','05/02/1989','Bookkeeping, Accounts Receivable,
        ('Anita','Bath','1037 Stuart Street','Rancho Murieta','CA','95683','ABath@hotmail.com','916-579-0359','01/23/2010','Owner', '3'),--(SELECT JobRoleID WHE
        ('Augusta','Wind','2756 Green Hill Road','Carmichael','CA','95608','WindyCity999@yahoo.com','916-354-3694','07/17/2015','Business Development, Full-Cycl
        ('Saul','Goodmate','1497 Lunetta Street','Elk Grove','CA','95624','DesignedthisWay@Adobe.com','916-515-5070','12/31/1999','Adobe Creative Suite, Graphic


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

-------
```

```
330
331 --Drop Function at beginning
332 --Function to retrieve CustomerID with BusinessName
333 GO
334 CREATE FUNCTION dbo.CustomerIDLookup(@BusinessName char(50))
335 RETURNS int AS -- CustomerID is int
336 BEGIN
337     DECLARE @returnValue int -- assigns CustomerID that matches business name temporarily to variable
338     SELECT @returnValue = CustomerID FROM Customer
339     WHERE BusinessName = @BusinessName
340     RETURN @returnValue --send back CustomerID
341 END
342 GO
343
344 SELECT dbo.CustomerIDLookup('Acme Corporation')
345 SELECT dbo.CustomerIDLookup('Globex Corporation')
346 SELECT dbo.CustomerIDLookup('Soylent Corp')
347 SELECT dbo.CustomerIDLookup('Initech')
348 SELECT dbo.CustomerIDLookup('Umbrella Corporation')
349 SELECT dbo.CustomerIDLookup('Hooli')
350 SELECT dbo.CustomerIDLookup('Pet Tricks')
351
352
353 SELECT * FROM Employee --check results
354
355 -------
356
357 INSERT INTO JobOrder(CustomerID, OrderType, Date, Status, EmployeeID)
358     VALUES
359     ('1', 'design', '01/01/2020', 'Closed','4'),
360     ('1', 'design', '04/01/2020', 'Closed','4'),
361     ('1', 'design', '07/01/2020', 'Closed','4'),
362     ('1', 'print', '02/15/2020', 'Closed','4'),
363     ('2', 'design', '04/04/2020', 'Closed','4'),
364     ('2', 'design', '07/07/2020', 'Closed','4'),
365     ('2', 'print', '02/20/2020', 'Closed','4'),
366     ('2', 'design', '05/05/2020', 'Closed','4'),
367     ('3', 'design', '08/01/2020', 'Closed','4'),
368     ('3', 'design', '01/15/2020', 'Closed','4'),
369     ('4', 'design', '05/01/2020', 'Closed','4'),
370     ('4', 'design', '08/23/2020', 'Closed','4'),
371     ('4', 'print', '02/15/2020', 'Closed','4'),
372     ('5', 'design', '05/05/2020', 'Closed','4'),
373     ('5', 'design', '01/15/2020', 'Closed','4'),
374     ('6', 'print', '02/18/2020', 'Closed','4'),
375     ('6', 'design', '05/15/2020', 'Closed','4'),
376     ('6', 'design', '09/01/2020', 'Closed','4'),
377     ('6', 'design', '03/14/2020', 'Closed','4'),
378     ('7', 'print', '06/01/2020', 'Closed','4'),
379     ('7', 'design', '09/09/2020', 'Closed','4')
380
381 -------
```

```
356
357 INSERT INTO JobOrder(CustomerID, OrderType, Date, Status, EmployeeID)
358     VALUES
359     ('1', 'design', '01/01/2020', 'Closed','4'),
360     ('1', 'design', '04/01/2020', 'Closed','4'),
361     ('1', 'design', '07/01/2020', 'Closed','4'),
362     ('1', 'print', '02/15/2020', 'Closed','4'),
363     ('2', 'design', '04/04/2020', 'Closed','4'),
364     ('2', 'design', '07/07/2020', 'Closed','4'),
365     ('2', 'print', '02/20/2020', 'Closed','4'),
366     ('2', 'design', '05/05/2020', 'Closed','4'),
367     ('3', 'design', '08/01/2020', 'Closed','4'),
368     ('3', 'design', '01/15/2020', 'Closed','4'),
369     ('4', 'design', '05/01/2020', 'Closed','4'),
370     ('4', 'design', '08/23/2020', 'Closed','4'),
371     ('4', 'print', '02/15/2020', 'Closed','4'),
372     ('5', 'design', '05/05/2020', 'Closed','4'),
373     ('5', 'design', '01/15/2020', 'Closed','4'),
374     ('6', 'print', '02/18/2020', 'Closed','4'),
375     ('6', 'design', '05/15/2020', 'Closed','4'),
376     ('6', 'design', '09/01/2020', 'Closed','4'),
377     ('6', 'design', '03/14/2020', 'Closed','4'),
378     ('7', 'print', '06/01/2020', 'Closed','4'),
379     ('7', 'design', '09/09/2020', 'Closed','4')
380
381 -------
382
383 SELECT * FROM JobOrder
384 --Adding Data to Table Invoice
385 INSERT INTO Invoice(Amount, Date, Status)
386     VALUES
387     ('150', '03/01/2020', 'PAID'),--Acme
388     ('2000', '06/01/2020', 'PAID'),--Acme
389     ('1150', '09/01/2020', 'PAID'),--Acme
390     ('200', '04/15/2020', 'PAID'),--Acme
391     ('150', '06/04/2020', 'PAID'),--Globex
392     ('1150', '09/07/2020', 'PAID'),--Globex
393     ('200', '04/20/2020', 'PAID'),--Globex
394     ('12000', '06/20/2020', 'PAID'),--Globex
395     ('11400','09/01/2020', 'PAID'),--Soylent
396     ('11500','02/15/2020', 'PAID'),--Soylent
397     ('1150', '07/01/2020', 'PAID'),--Initech
398     ('1500', '10/08/2020', 'PAID'),--Initech
399     ('200', '04/15/2020', 'PAID'),--Initech
400     ('17000', '06/05/2020', 'PAID'),--Umbrella
401     ('2200', '02/15/2020', 'PAID'),--Umbrella
402     ('200', '04/18/2020', 'PAID'),--Hooli
403     ('1150', '07/15/2020', 'PAID'),--Hooli
404     ('300', '11/01/2020', 'PAID'),--Hooli
405     ('4000', '05/14/2020', 'PAID'),--Hooli
406     ('23000', '07/01/2020', 'PAID'),--Pet Tricks
407     ('61200', '10/09/2020', 'PAID')--Pet Tricks
408
409 SELECT * FROM Invoice --check results
410
411 -------
```

```
-------

--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

-------
```

```
492
493  --Adding Data to Table OrderDetail
494  INSERT INTO OrderDetail (OrderID, ProdServID, DescriptionofRequest, Deadline, Priority)
495          VALUES
496          ('1','9', 'Interior wayfinding sign, branded- armory, 1x2', '03/01/2020', 'LOW'),
497          ('2','1', 'single wall wallpaper graphic, 8x10', '06/01/2020', 'LOW'),
498          ('3','1', 'branded HR poster, 2x3', '09/01/2020', 'LOW'),
499          ('4','7', 'annual report layout, no design', '04/15/2020', 'LOW'),
500          ('5','9', 'Interior wayfinding sign, branded- cafeteria, 1x2', '06/04/2020', 'LOW'),
501          ('6','1', 'branded HR poster, 2x3', '09/07/2020', 'LOW'),
502          ('7','7', 'annual report layout, no design', '04/20/2020', 'LOW'),
503          ('8','1', '10x Custom wallpaper pattern, trendy vintage, 8x10 ea', '06/20/2020', 'MED'),
504          ('9','3', 'Privacy glass with custom science graphic, 8x32','09/01/2020','RED FLAG'),
505          ('10','9', '70x Interior wayfinding signs, custom design- restroom, 1x2','02/15/2020','RED FLAG'),
506          ('11','1', 'branded HR poster, 2x3', '07/01/2020', 'LOW'),
507          ('12','9', '10x Interior wayfinding signs, branded- restroom, 1x2', '10/08/2020', 'MED'),
508          ('13','7', 'annual report layout, no design', '04/15/2020', 'LOW'),
509          ('14','1', '5x wall wallpaper graphics, original art mural, 8x12 ea', '06/05/2020', 'RED FLAG'),
510          ('15','7', 'annual report layout, custom', '02/15/2020', 'RED FLAG'),
511          ('16','7', 'annual report layout, no design', '04/18/2020', 'LOW'),
512          ('17','1', 'branded poster, 2x3', '07/15/2020', 'LOW'),
513          ('18','9', '2x Interior wayfinding signs, branded- cafeteria, 1x2', '11/01/2020', 'LOW'),
514          ('19','1', 'single wall wallpaper graphic, key art, 8x10', '05/14/2020', 'MED'),
515          ('20','1', '20x custom branded posters, new-releases key art, 2x3', '07/01/2020', 'RED FLAG'),
516          ('21','1', '18x wallpaper graphics, award-winners key art, 8x12 ea', '10/09/2020', 'RED FLAG')
517
518  --
519  --Invoke Function to retrieve EmployeeID with Employee Name
520  SELECT dbo.EmployeeIDLookup('Willie')
521
522  --Update an Employee Address
523  UPDATE Employee SET Address = '1100 Fruit Bat Way' WHERE EmployeeID = 1
524
525  SELECT * FROM Employee WHERE FirstName = 'Willie'
526
527  ------
528
529  --Invoke Function to retrieve CustomerID with Business Name
530
531  SELECT dbo.CustomerIDLookup('Acme Corporation')
532
533  --Update a Customer POC
534  UPDATE Customer SET POCFirstName ='Wile.E' WHERE CustomerID = 1
535  UPDATE Customer SET POCLastName ='Coyote' WHERE CustomerID = 1
536  UPDATE Customer SET EmailAddress ='WECoyote@Acme.com' WHERE CustomerID = 1
537  UPDATE Customer SET StartDate ='12/31/1999' WHERE CustomerID = 1
538
539  --Check Records
540  SELECT * FROM Customer WHERE BusinessName = 'Acme Corporation'
541
```

Updates/Deletions

```
519  --Invoke Function to retrieve EmployeeID with Employee Name
520  SELECT dbo.EmployeeIDLookup('Willie')
521
522  --Update an Employee Address
523  UPDATE Employee SET Address = '1100 Fruit Bat Way' WHERE EmployeeID = 1
524
525  SELECT * FROM Employee WHERE FirstName = 'Willie'
526
527  ------
528
529  --Invoke Function to retrieve CustomerID with Business Name
530
531  SELECT dbo.CustomerIDLookup('Acme Corporation')
532
533  --Update a Customer POC
534  UPDATE Customer SET POCFirstName ='Wile.E' WHERE CustomerID = 1
535  UPDATE Customer SET POCLastName ='Coyote' WHERE CustomerID = 1
536  UPDATE Customer SET EmailAddress ='WECoyote@Acme.com' WHERE CustomerID = 1
537  UPDATE Customer SET StartDate ='12/31/1999' WHERE CustomerID = 1
538
539  --Check Records
540  SELECT * FROM Customer WHERE BusinessName = 'Acme Corporation'
541
542  ------
543
544  --Invoke Function to retrieve EmployeeID with Employee Name
545  SELECT dbo.EmployeeIDLookup('Willie')
546
547  --Update an Employee Phone Number
548  UPDATE Employee SET PhoneNum = '916-444-9860' WHERE EmployeeID = 1
549
550  SELECT * FROM Employee WHERE FirstName = 'Willie'
551
552  --------
```

```
--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

-------
```

```
656
657  SELECT * FROM Customer WHERE BusinessName = 'Acme Corporation'
658 --Update JobOrder when Invoice OPEN(UNPAID) or CLOSED(PAID)(stored procedure)
659  --DROP PROCEDURE IF EXISTS UpdateJobOrder
660  --GO
661  --Drop procedure at beginning
662  GO
663 CREATE PROCEDURE UpdateJobOrder(@Status char(10),@OrderID int)
664  AS
665 BEGIN
666
667      UPDATE JobOrder SET Status = @Status
668      WHERE OrderID = @OrderID
669  END
670  GO
671 EXEC UpdateJobOrder 'Open','1'
672
673 SELECT * FROM JobOrder --check results
674
675  -------
676
677  --Invoke Function to retrieve EmployeeID with Employee Name
678  SELECT dbo.EmployeeIDLookup('Paige')
679
680 -------
```

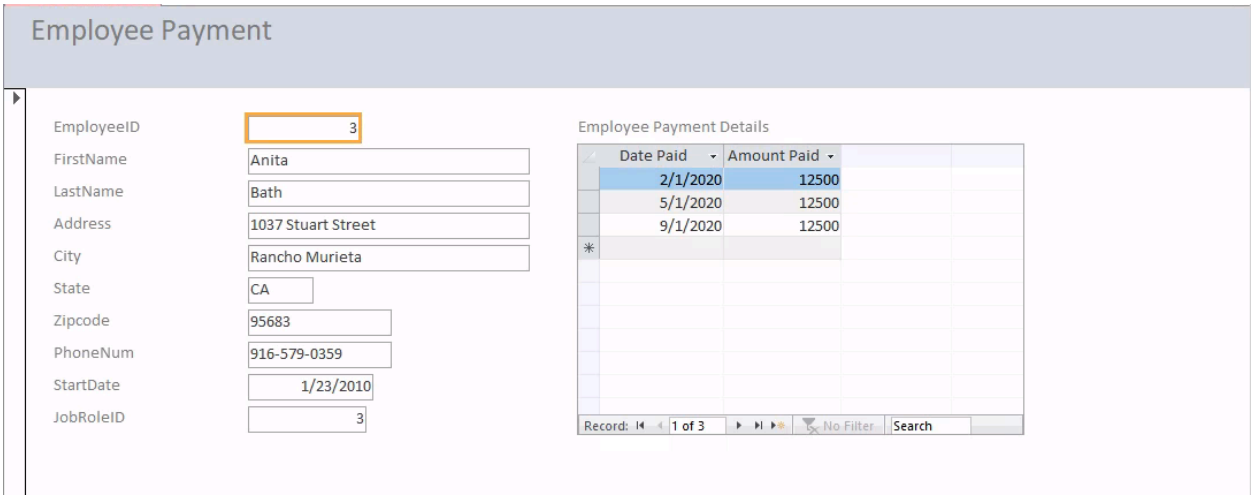
```
680 -------
681
682  --DROP PROCEDURE IF EXISTS DeleteEmployee
683  --GO
684  --Drop procedure at beginning
685  GO
686 CREATE PROCEDURE DeleteEmployee(@EmployeeID int)
687  AS
688 BEGIN
689      DELETE EmployeePayment
690      WHERE EmployeeID = @EmployeeID;
691      DELETE Employee
692      WHERE EmployeeID = @EmployeeID;
693  END
694  GO
695
696 EXEC DeleteEmployee '2'
697
698 SELECT * FROM EMPLOYEE --check employee records
699
700  -------
```

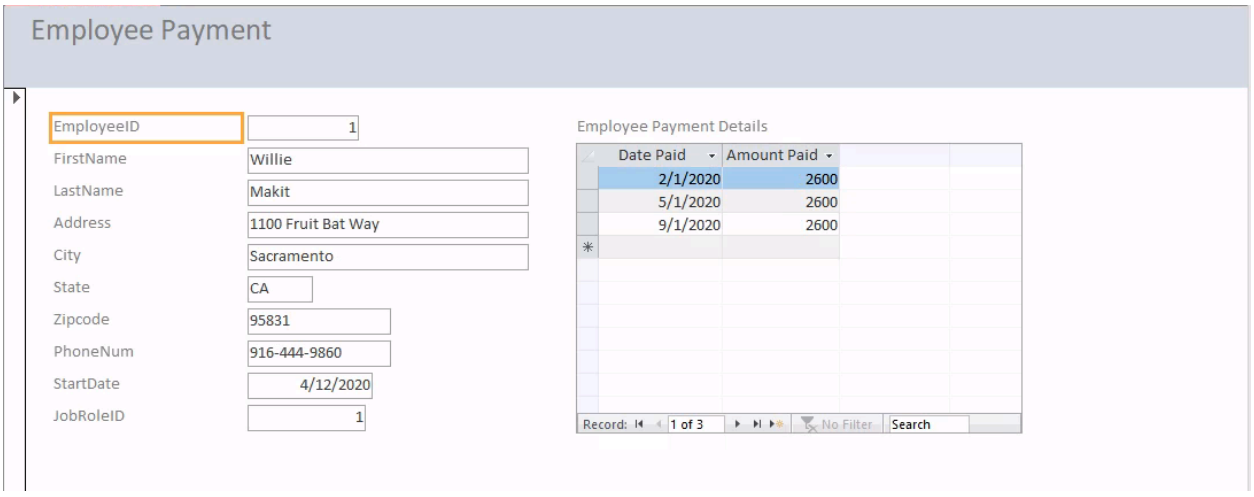
Views

```
720
721 --Create a View for monthly payroll
722 --DROP VIEW IF EXISTS MonthlyPayroll
723 --GO
724 --Drop view at beginning
725 GO
726 CREATE VIEW MonthlyPayroll AS
727     SELECT
728         Employee.FirstName,
729         Employee.LastName,
730         SUM(Amount) as Payroll,
731         EmployeePayment.Date
732     FROM EmployeePayment
733     JOIN Employee ON Employee.EmployeeID = EmployeePayment.EmployeeID
734     WHERE EmployeePayment.Date = '2020-09-01'
735     GROUP BY
736         Employee.FirstName,
737         Employee.LastName,
738         EmployeePayment.Date
739 GO
740
741 SELECT * FROM MonthlyPayroll
742
743 -------
744
745 --Create a View for current open invoices
746 --DROP VIEW IF EXISTS CurrentOpenInvoices
747 --GO
748 --Drop at beginning
749 GO
750 CREATE VIEW CurrentOpenInvoices AS
751     SELECT
752         JobOrder.OrderID,
753         JobOrder.CustomerID,
754         Invoice.Amount,
755         Invoice.Date,
756         Customer.BusinessName
757     FROM
758         JobOrder
759     JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID
760     JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID
761     JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID
762     WHERE Invoice.Status = 'UNPAID'
763     GROUP BY
764         JobOrder.OrderID,
765         JobOrder.CustomerID,
766         Invoice.Amount,
767         Invoice.Date,
768         Customer.BusinessName
769 GO
770
771 SELECT * FROM CurrentOpenInvoices
772
```

```sql
--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

-------
```

Maintenance Forms





Reports

Current Customer Contact Information

BusinessName	POCFirstName	POCLastName	BusinessAddress	BusinessCity	Busine	Business2	EmailAddress
Acme Corporation	Wile.E	Coyote	3655 N Cliff St	San Jose	CA	95134	WECoyote@Acme.com
Globex Corporatio	Hank	Scorpio	123 Globex Blvd	Cypress Creek	CA	95132	HScorpio@GlobexCO.c
Soylent Corp	Bill	Simonson	777 S Peoples St	Los Angeles	CA	90021	BS@Soylent.com
Initech	Milton	Waddams	156 Swingline Rd	Mountain View	CA	95030	MiltonW@Initech.com
Umbrella Corporat	Alicia	Marcus	7000 Resident Blvc	Las Vegas	NV	89115	AMarcus@Umbrella.co
Hooli	Anthony	Dude	2500 Wideway Dr	Santa Monica	CA	90404	ADude@Hooli.com
Pet Tricks	Chelsea	Hill	100 Remington Cir	Los Gatos	CA	95032	CHill@PetTricks.com

Employee Payment Report

EmployeeID	FirstName	LastName	JobRoleID	Amount	Date
1	Willie	Makit	1		
				2600	9/1/2020
				2600	5/1/2020
				2600	2/1/2020
3	Anita	Bath	3		
				12500	9/1/2020
				12500	5/1/2020
				12500	2/1/2020
4	Augusta	Wind	4		
				8333	9/1/2020
				8333	5/1/2020
				8333	2/1/2020
5	Saul	Goodmate	5		
				7000	9/1/2020
				7000	5/1/2020
				7000	2/1/2020

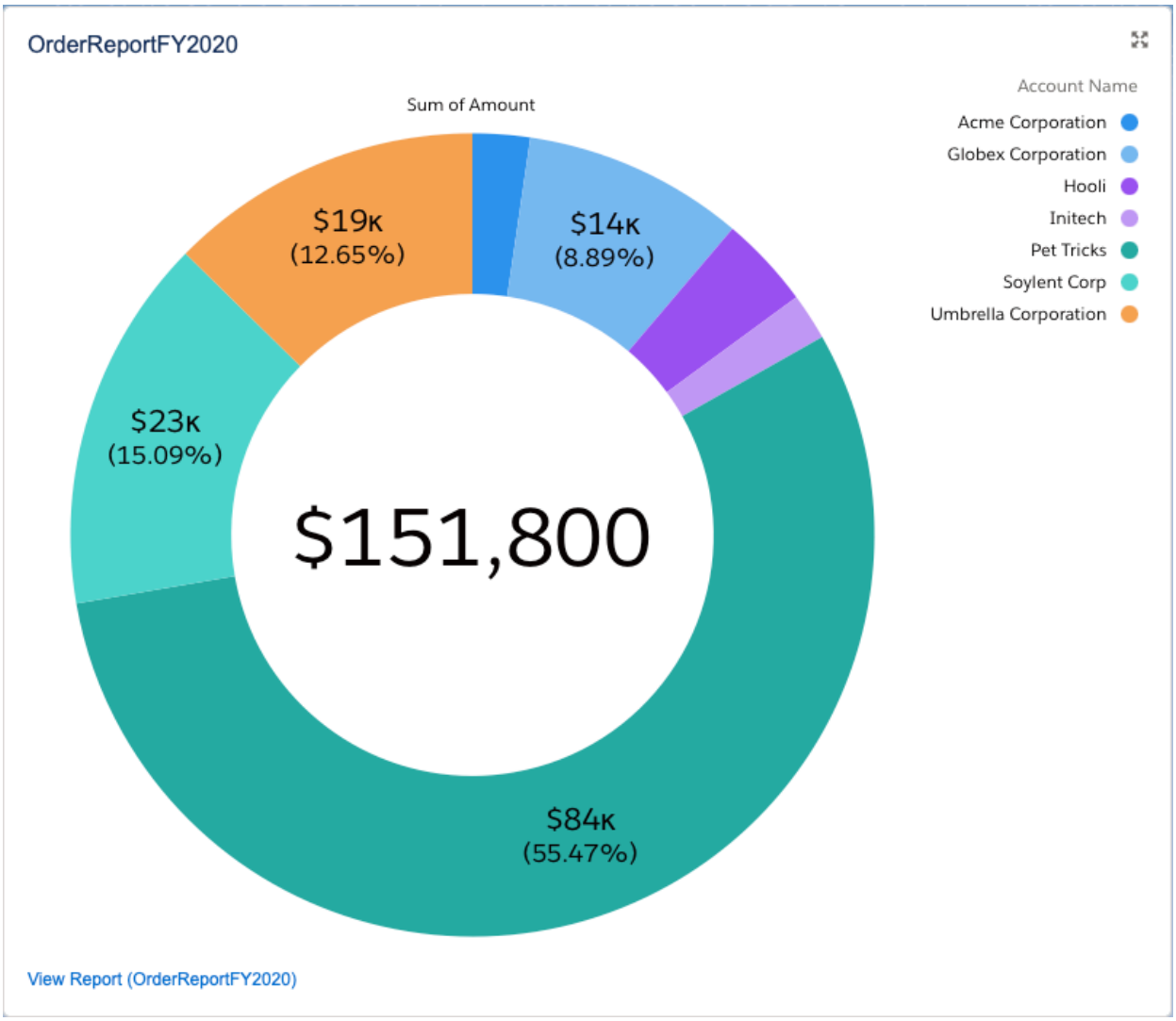
Data Questions

```
1097  ----
1098  --Data Question #1 What is the total amount spent per Customer?
1099
1100  --DROP VIEW IF EXISTS TotalPerCustomer
1101  --GO
1102  --Drop at beginnning
1103  --Create view for total spent per customer
1104  GO
1105  CREATE VIEW TotalPerCustomer AS
1106  SELECT
1107      Customer.CustomerID,
1108      Customer.BusinessName,
1109      SUM(Amount) as TotalOrderPerCustomer
1110  FROM
1111      JobOrder
1112  JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID
1113  JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID
1114  JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID
1115  WHERE Invoice.Status = 'PAID'
1116  GROUP BY
1117      Customer.CustomerID,
1118      Customer.BusinessName
1119  GO
1120  SELECT * FROM TotalPerCustomer
```

100 %

Results | Messages

	CustomerID	BusinessName	TotalOrderPerCustomer
1	1	Acme Corporation	3350
2	2	Globex Corporation	13500
3	3	Soylent Corp	22900
4	4	Initech	2850
5	5	Umbrella Corporation	19200
6	6	Hooli	5650
7	7	Pet Tricks	84200


OrderReportFY2020
Sum of Amount
$19K
(12.65%)
$14K
(8.89%)
$23K
(15.09%)
$151,800
$84K
(55.47%)
Account Name
Acme Corporation
Globex Corporation
Hooli
Initech
Pet Tricks
Soylent Corp
Umbrella Corporation
View Report (OrderReportFY2020)

```sql
--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
```

100 %

Results | Messages

	CustomerID	BusinessName	AvgOrderPerCustomer
1	1	Acme Corporation	1116
2	2	Globex Corporation	3375
3	3	Soylent Corp	11450
4	4	Initech	950
5	5	Umbrella Corporation	9600
6	6	Hooli	1412
7	7	Pet Tricks	42100

```sql
--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
-------
```

100 %

Results | Messages

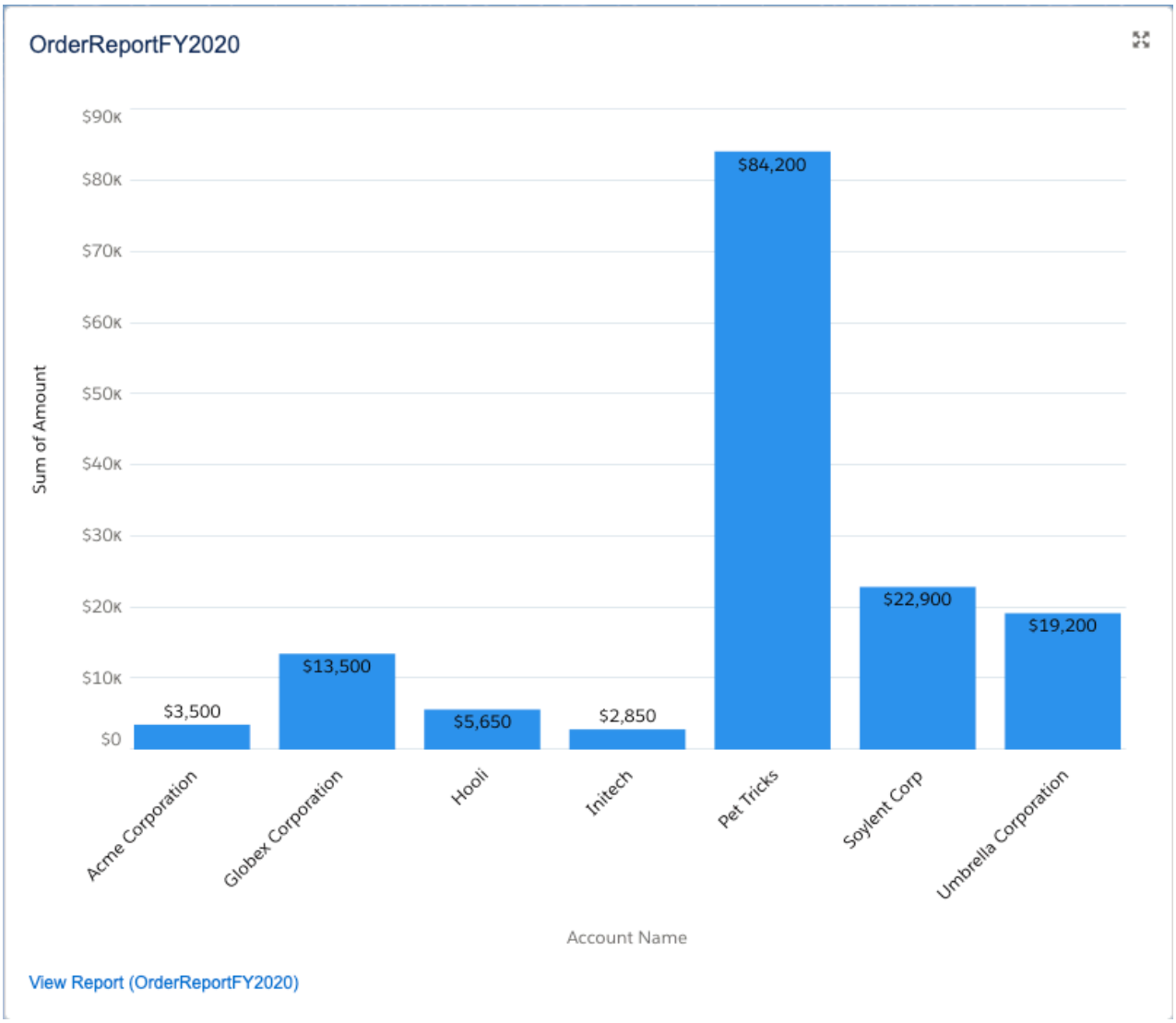
	OrderID	BusinessName	AmountSpentPerOrder
1	2	Acme Corporation	2000
2	3	Acme Corporation	1150
3	4	Acme Corporation	200
4	5	Globex Corporation	150
5	6	Globex Corporation	1150
6	7	Globex Corporation	200
7	8	Globex Corporation	12000
8	16	Hooli	200
9	17	Hooli	1150
10	18	Hooli	300
11	19	Hooli	4000
12	11	Initech	1150
13	12	Initech	1500
14	13	Initech	200
15	20	Pet Tricks	23000
16	21	Pet Tricks	61200
17	9	Soylent Corp	11400
18	10	Soylent Corp	11500
19	14	Umbrella Corporation	17000
20	15	Umbrella Corporation	2200

```sql
--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
```

100 %

Results | Messages

	BusinessName	CustomerID	BusinessName	TotalOrderPerCustomer
1	Pet Tricks	7	Pet Tricks	84200
2	Soylent Corp	3	Soylent Corp	22900
3	Umbrella Corporation	5	Umbrella Corporation	19200


OrderReportFY2020
$90K
$80K
$70K
$60K
$50K
$40K
$30K
$20K
$10K
$0
Sum of Amount
$3,500
$13,500
$5,650
$2,850
$84,200
$22,900
$19,200
Acme Corporation
Globex Corporation
Hooli
Initech
Pet Tricks
Soylent Corp
Umbrella Corporation
Account Name
View Report (OrderReportFY2020)

```sql

--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

----
```

100 %

Results | Messages

	BusinessName	CustomerID	BusinessName	TotalOrderPerCustomer
1	Initech	4	Initech	2850
2	Acme Corporation	1	Acme Corporation	3350
3	Hooli	6	Hooli	5650

```sql
--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
```

100 %

Results | Messages

	MonthBegins	MonthlyProfit
1	2020-02-01 00:00:00.000	13700
2	2020-03-01 00:00:00.000	150
3	2020-04-01 00:00:00.000	800
4	2020-05-01 00:00:00.000	4000
5	2020-06-01 00:00:00.000	31150
6	2020-07-01 00:00:00.000	25300
7	2020-09-01 00:00:00.000	13700
8	2020-10-01 00:00:00.000	62700
9	2020-11-01 00:00:00.000	300

```
770  --Create a View for current Closed invoices
771  --DROP VIEW IF EXISTS CurrentClosedInvoices
772  --GO
773  --Drop at beginning
774  GO
775  CREATE VIEW CurrentClosedInvoices AS
776      SELECT
777          JobOrder.OrderID,
778          JobOrder.CustomerID,
779          Invoice.Amount,
780          Invoice.Date,
781          Customer.BusinessName
782      FROM
783          JobOrder
784      JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID
785      JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID
786      JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID
787      WHERE Invoice.Status = 'PAID'
788      GROUP BY
789          JobOrder.OrderID,
790          JobOrder.CustomerID,
791          Invoice.Amount,
792          Invoice.Date,
793          Customer.BusinessName
794      GO
795
796  SELECT * FROM CurrentClosedInvoices
```

100 %

Results | Messages

	OrderID	CustomerID	Amount	Date	BusinessName
1	2	1	2000	2020-06-01 00:00:00.000	Acme Corporation
2	3	1	1150	2020-09-01 00:00:00.000	Acme Corporation
3	4	1	200	2020-04-15 00:00:00.000	Acme Corporation
4	5	2	150	2020-06-04 00:00:00.000	Globex Corporation
5	6	2	1150	2020-09-07 00:00:00.000	Globex Corporation
6	7	2	200	2020-04-20 00:00:00.000	Globex Corporation
7	8	2	12000	2020-06-20 00:00:00.000	Globex Corporation
8	9	3	11400	2020-09-01 00:00:00.000	Soylent Corp
9	10	3	11500	2020-02-15 00:00:00.000	Soylent Corp
10	11	4	1150	2020-07-01 00:00:00.000	Initech
11	12	4	1500	2020-10-08 00:00:00.000	Initech
12	13	4	200	2020-04-15 00:00:00.000	Initech
13	14	5	17000	2020-06-05 00:00:00.000	Umbrella Corporation
14	15	5	2200	2020-02-15 00:00:00.000	Umbrella Corporation
15	16	6	200	2020-04-18 00:00:00.000	Hooli
16	17	6	1150	2020-07-15 00:00:00.000	Hooli
17	18	6	300	2020-11-01 00:00:00.000	Hooli
18	19	6	4000	2020-05-14 00:00:00.000	Hooli
19	20	7	23000	2020-07-01 00:00:00.000	Pet Tricks
20	21	7	61200	2020-10-09 00:00:00.000	Pet Tricks

```
739  ------------------------------------------------
740  --Create a View for current open invoices
741  --DROP VIEW IF EXISTS CurrentOpenInvoices
742  --GO
743  --Drop at beginning
744  GO
745  CREATE VIEW CurrentOpenInvoices AS
746      SELECT
747          JobOrder.OrderID,
748          JobOrder.CustomerID,
749          Invoice.Amount,
750          Invoice.Date,
751          Customer.BusinessName
752      FROM
753          JobOrder
754      JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID
755      JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID
756      JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID
757      WHERE Invoice.Status = 'UNPAID'
758      GROUP BY
759          JobOrder.OrderID,
760          JobOrder.CustomerID,
761          Invoice.Amount,
762          Invoice.Date,
763          Customer.BusinessName
764  GO
765
766  SELECT * FROM CurrentOpenInvoices
```

100 %

Results | Messages

	OrderID	CustomerID	Amount	Date	BusinessName
1	1	1	150	2020-03-01 00:00:00.000	Acme Corporation

OrderReportFY2020

Account Name ↑	Opportunity Name	Fiscal Period	Close Date	Amount	Stage
Acme Corporation	branded HR poster, 2x3	Q3-2020	9/1/2020	$1,150.00	Closed Won
Acme Corporation	single wall wallpaper graphic, 8x10	Q2-2020	6/1/2020	$2,000.00	Closed Won
Acme Corporation	annual report layout, no design	Q2-2020	4/15/2020	$200.00	Closed Won
Acme Corporation	interior wayfinding sign, branded- armory, 1x2	Q1-2020	3/1/2020	$150.00	Closed Won
Globex Corporation	branded HR poster, 2x3	Q3-2020	9/7/2020	$1,150.00	Closed Won
Globex Corporation	10x Custom wallpaper pattern, trendy vintage, 8x...	Q2-2020	6/20/2020	$12,000.00	Closed Won
Globex Corporation	Interior wayfinding sign, branded- cafeteria, 1x2	Q2-2020	6/4/2020	$150.00	Closed Won
Globex Corporation	annual report layout, no design	Q2-2020	4/20/2020	$200.00	Closed Won
Hooli	2x Interior wayfinding signs, branded- cafeteria, 1x2	Q4-2020	11/1/2020	$300.00	Closed Won
Hooli	branded poster, 2x3	Q3-2020	7/15/2020	$1,150.00	Closed Won
Hooli	single wall wallpaper graphic, key art, 8x10	Q2-2020	5/14/2020	$4,000.00	Closed Won
Hooli	annual report layout, no design	Q2-2020	4/18/2020	$200.00	Closed Won
Initech	10x Interior wayfinding signs, branded- restroom, ...	Q4-2020	10/8/2020	$1,500.00	Closed Won
Initech	branded HR poster, 2x3	Q3-2020	7/1/2020	$1,150.00	Closed Won
Initech	annual report layout, no design	Q2-2020	4/15/2020	$200.00	Closed Won
Pet Tricks	18x wallpaper graphics, award-winners key art, 8x...	Q4-2020	10/9/2020	$61,200.00	Closed Won
Pet Tricks	20x custom branded posters, new-releases key art...	Q3-2020	7/1/2020	$23,000.00	Closed Won
Soylent Corp	Privacy glass with custom science graphic, 8x32	Q3-2020	9/1/2020	$11,400.00	Closed Won
Soylent Corp	70x Interior wayfinding signs, custom design- restr...	Q1-2020	2/15/2020	$11,500.00	Closed Won
Umbrella Corporation	5x wall wallpaper graphics, original art mural, 8x1...	Q2-2020	6/5/2020	$17,000.00	Closed Won
Umbrella Corporation	annual report layout, custom	Q1-2020	2/15/2020	$2,200.00	Closed Won

View Report (OrderReportFY2020)

```
1214  --Data Question #6 What is the total revenue for all Customers?
1215
1216  --DROP VIEW IF EXISTS TotalRevenue
1217  --GO
1218  --Drop at beginnning
1219  --Create view for total revenue, borrowed snippets from sample
1220  GO
1221  CREATE VIEW TotalRevenue AS
1222  SELECT
1223      SUM(Amount) as TotalProfit,
1224      GETDATE() as RangeDate
1225  FROM
1226      Invoice
1227  WHERE Invoice.Status = 'PAID'
1228  GO
1229  SELECT * FROM TotalRevenue
1230  ----
1231  --
```

100 %

Results | Messages

	TotalProfit	RangeDate
1	151650	2021-03-29 15:59:13.323

```sql
--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
```

100 %

Results | Messages

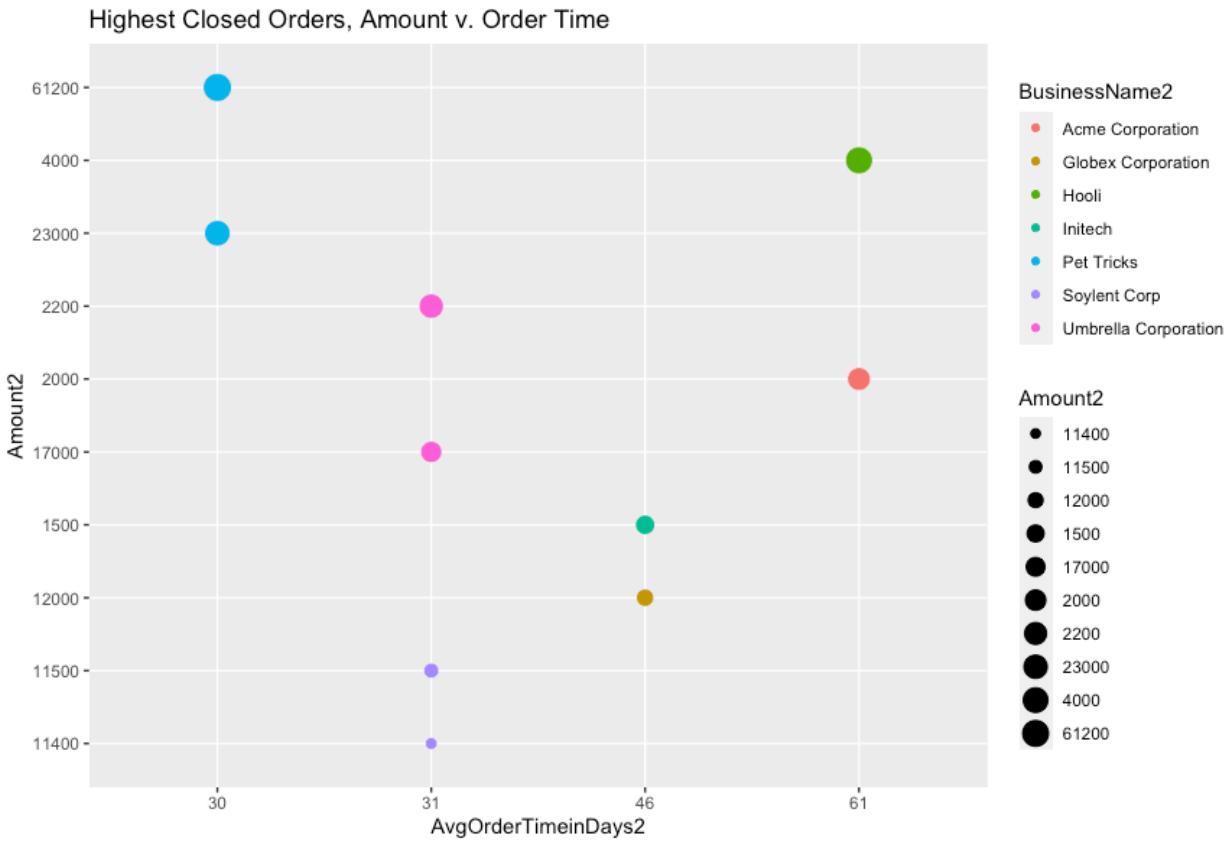
	BusinessName	StartDate	POCFirstName	POCLastName	BusinessAddress	BusinessCity	BusinessState	BusinessZipcode	EmailAddress	PhoneNum	Cust
1	Globex Corporation	2006-06-06 00:00:00.000	Hank	Scorpio	123 Globex Blvd	Cypress Creek	CA	95132	HScorpio@GlobexCO.com	507-926-7292	2
2	Hooli	2020-11-11 00:00:00.000	Anthony	Dude	2500 Wideway Dr	Santa Monica	CA	90404	ADude@Hooli.com	724-388-7389	6
3	Acme Corporation	1999-12-31 00:00:00.000	Wile.E	Coyote	3655 N Cliff St	San Jose	CA	95134	WECoyote@Acme.com	215-834-8729	1
4	Initech	2019-09-23 00:00:00.000	Milton	Waddams	156 Swingline Rd	Mountain View	CA	95030	MiltonW@Initech.com	412-601-8970	4
5	Pet Tricks	2011-03-28 00:00:00.000	Chelsea	Hill	100 Remington Cir	Los Gatos	CA	95032	CHill@PetTricks.com	412-861-2211	7
6	Soylent Corp	2015-08-08 00:00:00.000	Bill	Simonson	777 S Peoples St	Los Angeles	CA	90021	BS@Soylent.com	903-380-2339	3
7	Umbrella Corporation	2018-04-13 00:00:00.000	Alicia	Marcus	7000 Resident Blvd N	Las Vegas	NV	89115	AMarcus@Umbrella.com	412-292-1659	5

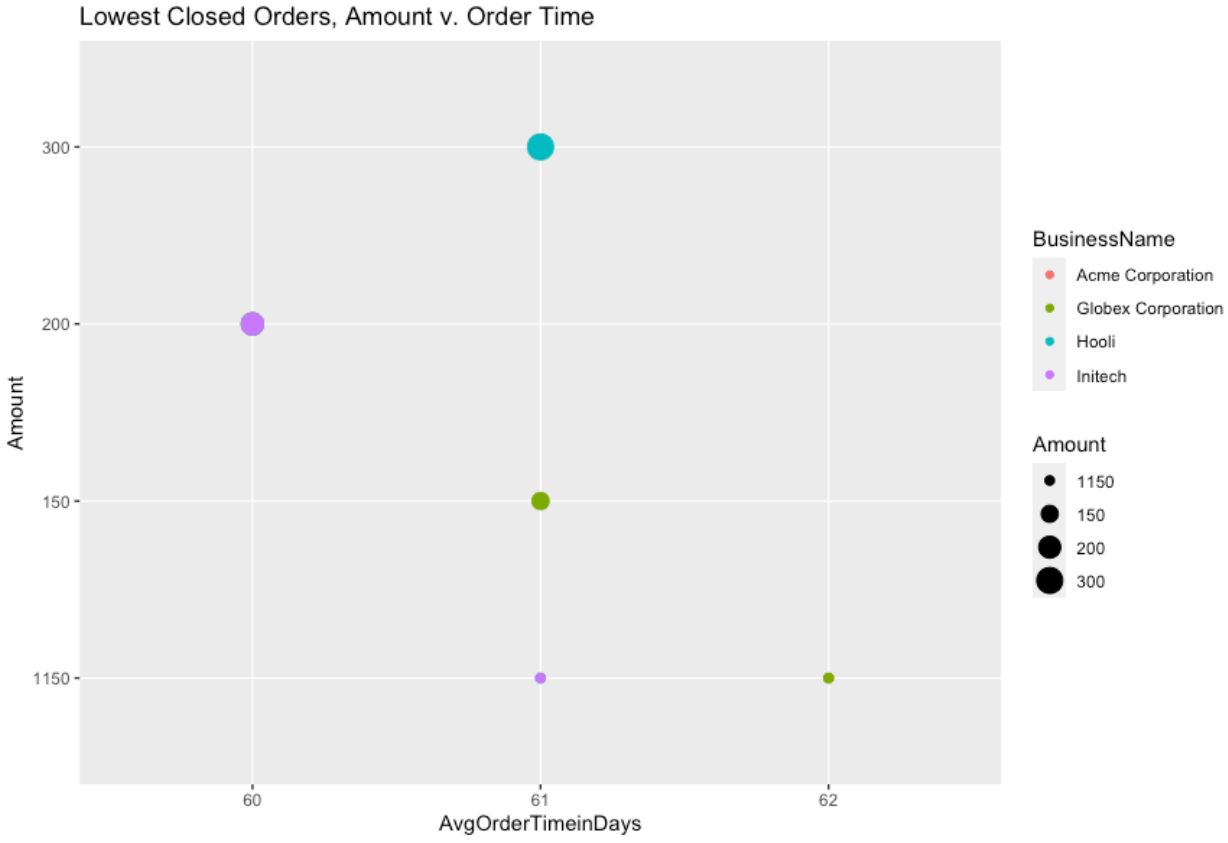
```
--Data Question # 8 What is the relationship between order price and time spent on order?

--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


-------
```

100 %

Results | Messages

	OrderID	InvoiceID	Amount	BusinessName	OrderTimeinDays
1	21	21	61200	Pet Tricks	30
2	20	20	23000	Pet Tricks	30
3	14	14	17000	Umbrella Corporation	31
4	8	8	12000	Globex Corporation	46
5	10	10	11500	Soylent Corp	31
6	9	9	11400	Soylent Corp	31
7	19	19	4000	Hooli	61
8	15	15	2200	Umbrella Corporation	31
9	2	2	2000	Acme Corporation	61
10	12	12	1500	Initech	46
11	11	11	1150	Initech	61
12	17	17	1150	Hooli	61
13	3	3	1150	Acme Corporation	62
14	6	6	1150	Globex Corporation	62
15	18	18	300	Hooli	61
16	16	16	200	Hooli	60
17	13	13	200	Initech	60
18	7	7	200	Globex Corporation	60
19	4	4	200	Acme Corporation	60
20	5	5	150	Globex Corporation	61


Highest Closed Orders, Amount v. Order Time
Amount2
61200
4000
23000
2200
2000
17000
1500
12000
11500
11400
30
31
46
61
AvgOrderTimeinDays2
BusinessName2
Acme Corporation
Globex Corporation
Hooli
Initech
Pet Tricks
Soylent Corp
Umbrella Corporation
Amount2
11400
11500
12000
1500
17000
2000
2200
23000
4000
61200


Lowest Closed Orders, Amount v. Order Time
Amount
300
200
150
1150
60
61
62
AvgOrderTimeinDays
BusinessName
Acme Corporation
Globex Corporation
Hooli
Initech
Amount
1150
150
200
300

```sql
--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
```

100 %

Results | Messages

	OrderID	InvoiceID	CustomerID	BusinessName	AvgOrderTimeinDays
1	2	2	1	Acme Corporation	61
2	3	3	1	Acme Corporation	62
3	4	4	1	Acme Corporation	60
4	5	5	2	Globex Corporation	61
5	6	6	2	Globex Corporation	62
6	7	7	2	Globex Corporation	60
7	8	8	2	Globex Corporation	46
8	16	16	6	Hooli	60
9	17	17	6	Hooli	61
10	18	18	6	Hooli	61
11	19	19	6	Hooli	61
12	11	11	4	Initech	61
13	12	12	4	Initech	46
14	13	13	4	Initech	60
15	20	20	7	Pet Tricks	30
16	21	21	7	Pet Tricks	30
17	9	9	3	Soylent Corp	31
18	10	10	3	Soylent Corp	31
19	14	14	5	Umbrella Corporation	31
20	15	15	5	Umbrella Corporation	31

```
853  -------
854
855  --SELECT OPEN Orders with time per order in days
856  SELECT
857      JobOrder.OrderID,
858      Invoice.InvoiceID,
859      Customer.CustomerID,
860      Customer.BusinessName,
861      ((AVG(DateDiff(n, JobOrder.Date, Invoice.Date))/60)/24) as AvgOrderTimeinDays
862  FROM
863      JobOrder
864  JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID
865  JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID
866  JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID
867  WHERE JobOrder.Status = 'OPEN'
868  GROUP BY
869      JobOrder.OrderID,
870      Invoice.InvoiceID,
871      Customer.CustomerID,
872      Customer.BusinessName
873  ORDER BY Customer.BusinessName
```

100 %

Results | Messages

	OrderID	InvoiceID	CustomerID	BusinessName	AvgOrderTimeinDays
1	1	1	1	Acme Corporation	60

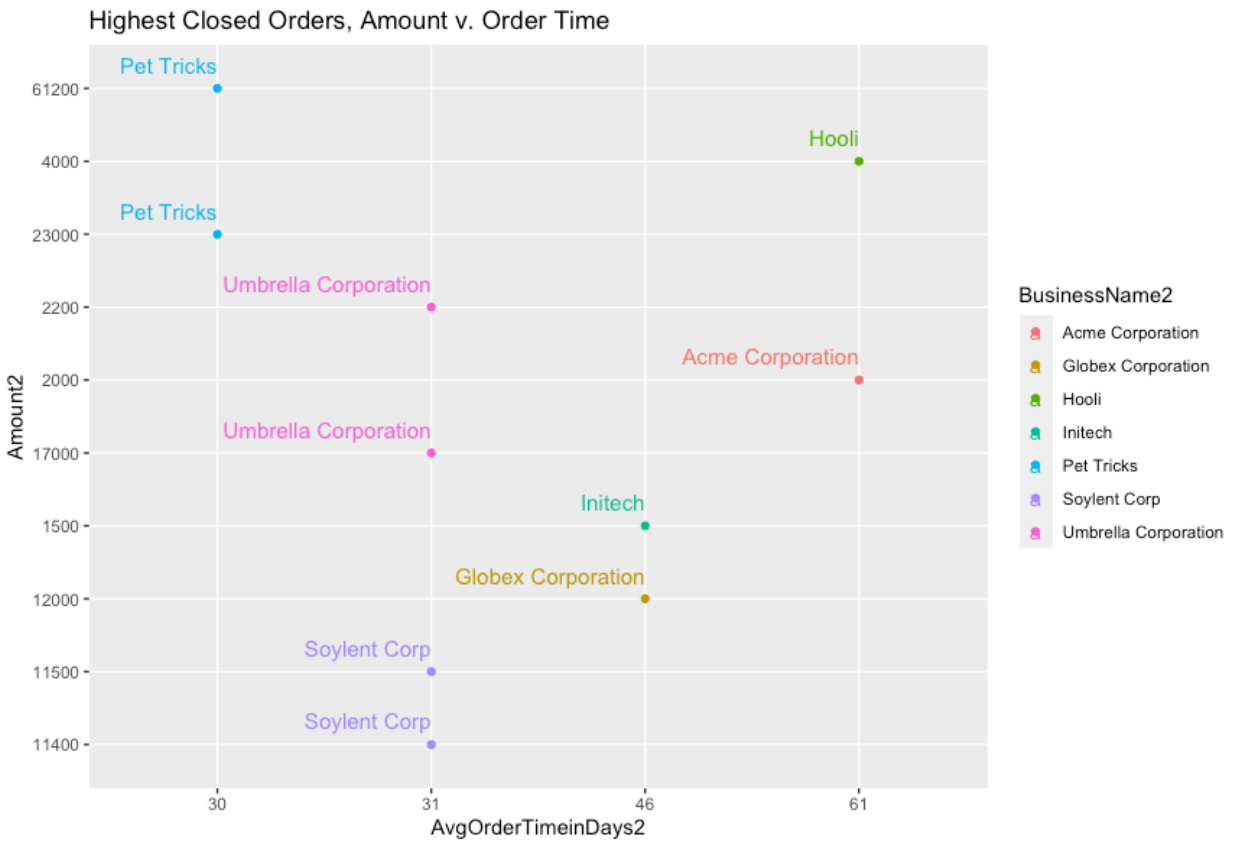
	OrderID	InvoiceID	Amount	BusinessName	OrderTimeinDays
1	21	21	61200	Pet Tricks	30
2	20	20	23000	Pet Tricks	30
3	14	14	17000	Umbrella Corporation	31
4	8	8	12000	Globex Corporation	46
5	10	10	11500	Soylent Corp	31
6	9	9	11400	Soylent Corp	31
7	19	19	4000	Hooli	61
8	15	15	2200	Umbrella Corporation	31
9	2	2	2000	Acme Corporation	61
10	12	12	1500	Initech	46
11	11	11	1150	Initech	61
12	17	17	1150	Hooli	61
13	3	3	1150	Acme Corporation	62
14	6	6	1150	Globex Corporation	62
15	18	18	300	Hooli	61
16	16	16	200	Hooli	60
17	13	13	200	Initech	60
18	7	7	200	Globex Corporation	60
19	4	4	200	Acme Corporation	60
20	5	5	150	Globex Corporation	61

```
1135
1136  --Data Question # 9 What is the relationship between order price, time per order, and a High-Priority Customer??
1137
1138  --SELECT Highest CLOSED Orders with time per order in days
1139  SELECT
1140      TOP 10 BusinessName,
1141      Invoice.Amount,
1142      ((AVG(DateDiff(n, JobOrder.Date, Invoice.Date))/60)/24) as AvgOrderTimeinDays
1143  FROM
1144      JobOrder
1145  JOIN InvoiceOrderBridge ON JobOrder.OrderID = InvoiceOrderBridge.OrderID
1146  JOIN Invoice ON InvoiceOrderBridge.InvoiceID = Invoice.InvoiceID
1147  JOIN Customer ON JobOrder.CustomerID = Customer.CustomerID
1148  WHERE JobOrder.Status = 'CLOSED'
1149  GROUP BY
1150      Invoice.Amount,
1151      Customer.BusinessName
1152  ORDER BY Invoice.Amount DESC
1153
1154  -------
1155
```

100 %

Results | Messages

	BusinessName	Amount	AvgOrderTimeinDays
1	Pet Tricks	61200	30
2	Pet Tricks	23000	30
3	Umbrella Corporation	17000	31
4	Globex Corporation	12000	46
5	Soylent Corp	11500	31
6	Soylent Corp	11400	31
7	Hooli	4000	61
8	Umbrella Corporation	2200	31
9	Acme Corporation	2000	61
10	Initech	1500	46

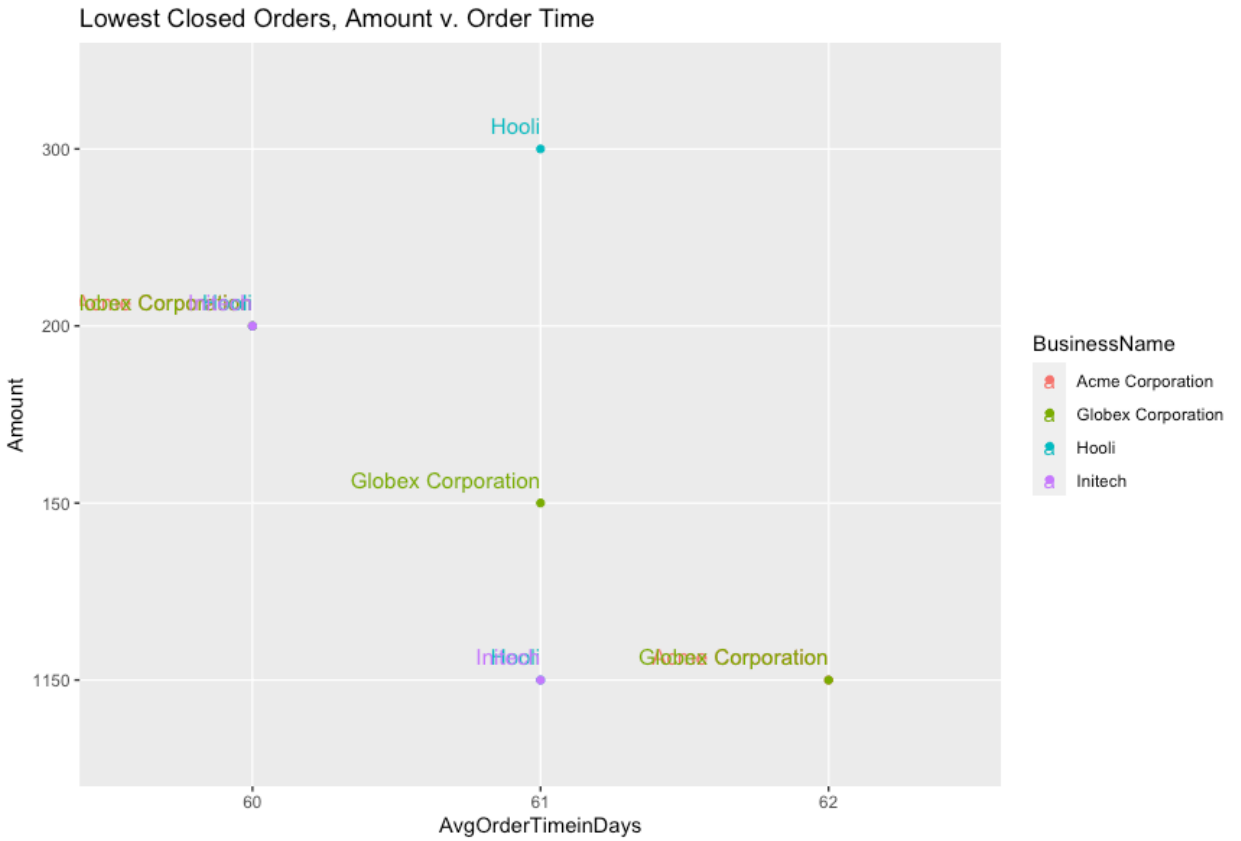


```sql
--Data Question # 10 What is the relationship between order price, time per order, and a Low-Priority Customer??

--SELECT Lowest 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

-------
```

100 %

Results | Messages

	BusinessName	Amount	AvgOrderTimeinDays
1	Globex Corporation	150	61
2	Acme Corporation	200	60
3	Globex Corporation	200	60
4	Hooli	200	60
5	Initech	200	60
6	Hooli	300	61
7	Acme Corporation	1150	62
8	Globex Corporation	1150	62
9	Hooli	1150	61
10	Initech	1150	61



Reflection

This project could easily go on another two months, and I really enjoyed working on it. I realized toward the end that I did not have some of the right kind of data to answer a few questions I had, so I had to adapt a question or two to fit the kind of information I would have been given in the scenario. I was really crunched for time at the end, even though I stayed on track reasonably well, because it was very easy to lose a few days trying to deal with one error, and I was reluctant to ask for help. I learned a lot trying to work through the many problems this project posed, and I can tell it only scratched the surface of capabilities with SQL. I genuinely enjoyed building this database, and will probably return to it for future projects.

My biggest challenge in this project was using the remote lab, because it made everything lag and that was really difficult. Even scrolling was a chore, and my connection would get dropped several tiomes a day. My biggest accomplishment with this project was being able to figure out errors, sometimes a weird missing comma somewhere, and perservering when it felt like the code would never work. It really did help to work on a topic I chose, because my interest in seeing it work properly was sometimes the last bit of motivation to get through a difficult section. I was also grateful for the time spent on the logical modeling, because that was my guide through every step of building and fleshing pout the database.

Overall, it was a great project, and I am looking forward to coming back to Hancock Graphics the next time they need me to refresh their database!

Summary

The data received was from the company database, and did not contain all of the information needed to answer the original data questions. One original path that had to be rerouted was to understand the amount of time an order stayed in a department, and further, hours an employee would work on a single order. This data is not available in this dataset. There was some data regarding the design department, but that was the only department that had hours identified, and those were masked by product prices, and could not be used. If additional analysis is to be done to explore employee hours, more data needs to be gathered.

The original hypothesis based on employee reports that the amount of time spent on customer orders was different for customers of different status levels proved true while examining the hours spent per order. That said, those customers consistently spent more, and the investment appears warranted. The data illustrated a marked difference between the top performing customers and the lowest performing customers, with total revenue from the top three customers well outweighing the rest of the customers, bringing in about 82% of the revenue. Investing more energy in those customers is the most logical thing to do, however, it would be wise to take care of the “little guys” in case one or more top customers finds another company to work with.

The data questions were answered by grouping the customers based on invoice amounts, order counts, and time spent per order. The interface used was Microsoft Access for ease of creating prototype forms, however, most visualizations were done in R and Salesforce. R was used to best display scatter plots, while Salesforce was used for the bar chart and a very relevant and trendy pie chart.

Appendix

```
-- 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','04/1
2/2020','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','07/17/201
5','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

-------
```