IST 659 Final Project

Zachary L. Chipman

3/29/21

Stakeholders:

The primary stakeholders are the wineries owners, employees, and investors. Through this database, they hope to better track sales and see what factors (grape varietal, vintage, etc.) are affecting profitability and adjust accordingly. Since, the winery works with several other businesses, they have been listed as secondary stakeholders.

Secondary stakeholders include:

The vineyards (who supply the grapes and may or may not be owned by the winery).

The cooperages who supplies the barrels.

The customers who can purchase wines directly through the online store. They are also asked to rate the wines on a scale of 1-5.

Other clients. These include stores and restaurants as well as the distributors who ship the wines to them. Sales numbers are tracked and sent back to the winery.

Business Rules:

1. The vineyards harvest the grapes and send them to the winery.

2. The winery receives barrels from the cooperage and ages the wine in them.

3. The winery bottles and packages the wine. Each varietal and vintage is bottled and packaged separately.

4. The winery sends wine to their clients (customers and businesses).

5. The businesses keep track of sales.

6. The businesses send sales data back to the winery.

7. A client (customer of business) places an order.

8. The winery receives an order form from the client.

9. The order is invoiced by the winery.

10. The invoice is recorded by the winery.

11. The order (with the copy of invoice) is shipped to the client.

Glossary:

A **vineyard** is a plot of land where grapes are grown.

A **wine** made of fermented grapes and is the sole product of the winery.

A **varietal** is the type of grape that is used in the wine. The winery uses six grape varietals to make its wine: Riesling, Chardonnay, Sauvignon Blanc, Merlot, Pinot Noir, and Cabernet Sauvignon.

A **vintage** indicates the year that the grapes were harvested. The quality of a wine can vary from vintage to vintage due to temperature, weather, and the overall climate.

A **barrel** is a container where wine is aged. Barrels are made by a cooperage and can be made of either French or American oak.

A **client** is any independent party that buys the wine. There are two types of clients: customers and businesses. Businesses include wholesalers, retailers, and restaurants.

An **order form** is a document that details the wines and the quantity of wines the customer wants.

An **invoice** is a document that is sent to the customer along with the product. It details the customer's order and a copy is kept be the winery as a record of the transaction.

Data Questions:

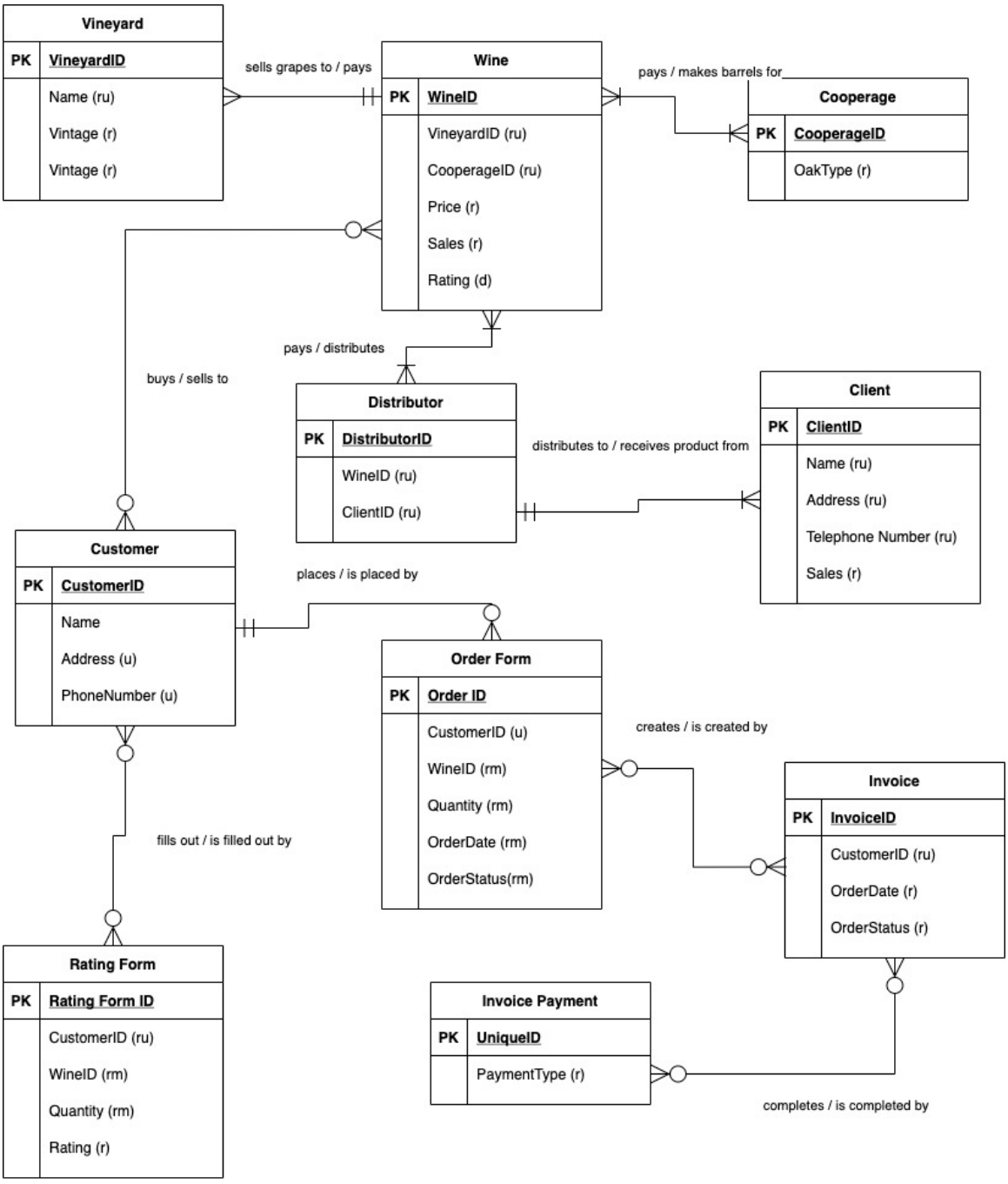
Are poor sales due to the varietal, vintage or oak type?

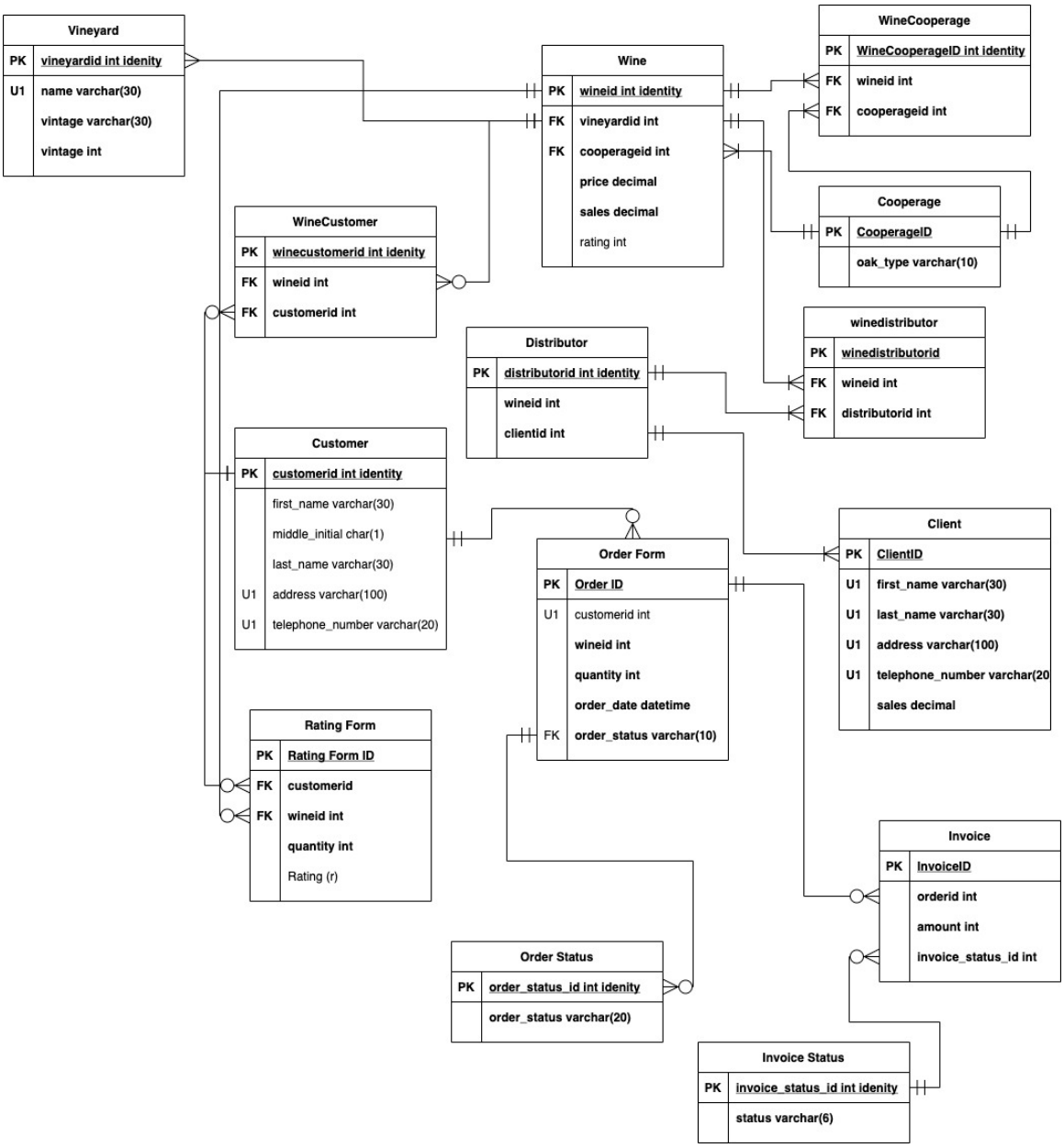
What varietal sells the best?

Do the rating and sales of a wine correlate?

What can be changed to raise a wines ratings/sales?

Which wines sell the best on the online store versus at stores/restaurant?


Vineyard
PK
VineyardID
Name (ru)
Vintage (r)
Vintage (r)
sells grapes to / pays
Wine
PK
WineID
VineyardID (ru)
CooperageID (ru)
Price (r)
Sales (r)
Rating (d)
pays / makes barrels for
Cooperage
PK
CooperageID
OakType (r)
pays / distributes
buys / sells to
Distributor
PK
DistributorID
WineID (ru)
ClientID (ru)
distributes to / receives product from
Client
PK
ClientID
Name (ru)
Address (ru)
Telephone Number (ru)
Sales (r)
Customer
PK
CustomerID
Name
Address (u)
PhoneNumber (u)
places / is placed by
Order Form
PK
Order ID
CustomerID (u)
WineID (rm)
Quantity (rm)
OrderDate (rm)
OrderStatus(rm)
creates / is created by
Invoice
PK
InvoiceID
CustomerID (ru)
OrderDate (r)
OrderStatus (r)
fills out / is filled out by
Rating Form
PK
Rating Form ID
CustomerID (ru)
WineID (rm)
Quantity (rm)
Rating (r)
Invoice Payment
PK
UniqueID
PaymentType (r)
completes / is completed by


Vineyard
PK vineyardid int idenity
U1 name varchar(30)
vintage varchar(30)
vintage int
Wine
PK wineid int identity
FK vineyardid int
FK cooperageid int
price decimal
sales decimal
rating int
WineCooperage
PK WineCooperageID int identity
FK wineid int
FK cooperageid int
Cooperage
PK CooperageID
oak_type varchar(10)
WineCustomer
PK winecustomerid int idenity
FK wineid int
FK customerid int
winedistributor
PK winedistributorid
FK wineid int
FK distributorid int
Distributor
PK distributorid int identity
wineid int
clientid int
Customer
PK customerid int identity
first_name varchar(30)
middle_initial char(1)
last_name varchar(30)
U1 address varchar(100)
U1 telephone_number varchar(20)
Order Form
PK Order ID
U1 customerid int
wineid int
quantity int
order_date datetime
FK order_status varchar(10)
Client
PK ClientID
U1 first_name varchar(30)
U1 last_name varchar(30)
U1 address varchar(100)
U1 telephone_number varchar(20
sales decimal
Rating Form
PK Rating Form ID
FK customerid
FK wineid int
quantity int
Rating (r)
Invoice
PK InvoiceID
orderid int
amount int
invoice_status_id int
Order Status
PK order_status_id int idenity
order_status varchar(20)
Invoice Status
PK invoice_status_id int idenity
status varchar(6)

IST 659 Project Milestone 2

Raw Data Sample:

Previously, the SQL Winery has been using several Excel Spreadsheets to organize their information. The spreadsheet for their clients is shown below.

	A	B	C	D	E
1	**Client Type**	**Client Name**	**Address**	**Telephone Number**	**Sales Quantity**
2	Restaurant	Flagship Steakhouse	6579 Python Street	(679) 564-2458	84,809
3	Retail	Grape Savings Store	3092 Java Drive	(546) 085-0188	70,066
4	Restaurant	Bill Dalton	6831 Tableau Avenue	(533) 422-7465	NULL
5	Customer	The Red Apple	2352 Unix Boulevard	(684) 279-6530	64,779
6	Restaurant	The Vine Dispatch	3648 Oracle Avenue	(257) 208-4486	50,909
7	Customer	Margot Brown	1923 Ruby Road	(304) 343-0332	NULL

Besides being able to better track the sales of their wines, the winery hopes that the database will provide them a system that is able to handle more clients so they can sell to more people and grow their business.

Physical Database Design:

```
/*
The following drops will assist in making a script
that can be repeated without error.
*/

-- Procedure Drops
DROP PROCEDURE IF EXISTS dbo.ChangeRating;
go

DROP PROCEDURE IF EXISTS dbo.ChangeStatus;
go

-- View Drops
DROP VIEW IF EXISTS dbo.ClientTypes;
go

DROP VIEW IF EXISTS dbo.Top3Wines;
go

DROP VIEW IF EXISTS dbo.AllRatings;
go

-- Funnction Drops
DROP FUNCTION IF EXISTS dbo.WineSalesFunction;
go
```

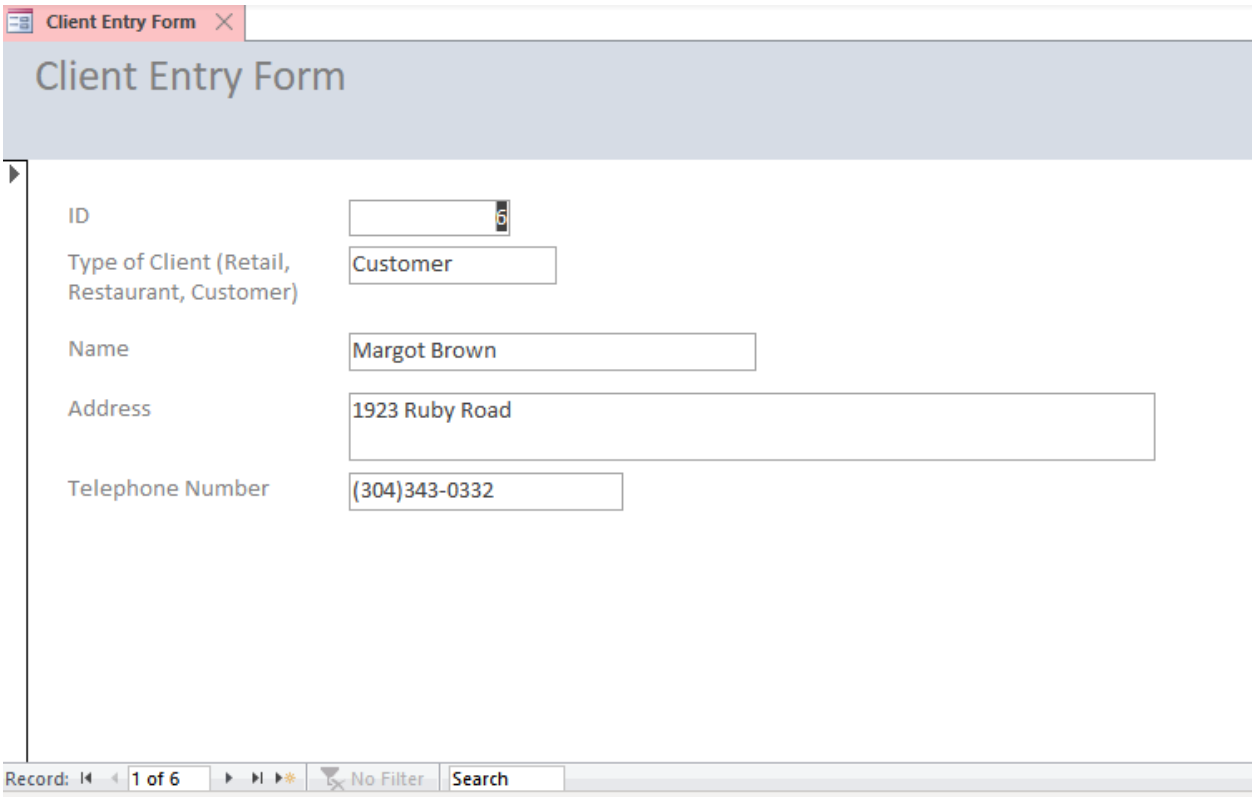
```
-- Table Drops
DROP TABLE IF EXISTS OrderForm;
go

DROP TABLE IF EXISTS WineClient;
go

DROP TABLE IF EXISTS Wine;
go

DROP TABLE IF EXISTS VineyardVintage;
go

DROP TABLE IF EXISTS VineyardVarietal;
go

DROP TABLE IF EXISTS OrderStatus;
go

DROP TABLE IF EXISTS Clients;
go

DROP TABLE IF EXISTS Barrel;
go

DROP TABLE IF EXISTS Vintage;
go

DROP TABLE IF EXISTS Varietal;
go

DROP TABLE IF EXISTS Vineyard;
go

-- create all tables in order of their dependencies

CREATE TABLE Vineyard
(
   -- Add columns
   vineyard_ID int NOT NULL IDENTITY,
   vineyard_name varchar(30) NOT NULL,
   -- Add constraints
   CONSTRAINT PK_Vineyard PRIMARY KEY (vineyard_ID),
   CONSTRAINT U1_Vineyard UNIQUE (vineyard_name)
);
go

CREATE TABLE Varietal
(
   -- Add columns
   varietal_ID int NOT NULL IDENTITY,
   varietal_name varchar(30) NOT NULL,
   -- Add constraints
   varietal_color varchar(10) NOT NULL,
   CONSTRAINT PK_Varietal PRIMARY KEY (varietal_ID),
   CONSTRAINT U1_Varietal UNIQUE (varietal_name)
);
```

```
go

CREATE TABLE Vintage
(
   -- Add columns
   vintage_ID int NOT NULL IDENTITY,
   vintage_year int NOT NULL,
   -- Add constraints
   CONSTRAINT PK_Vintage PRIMARY KEY (vintage_ID)

);
go

CREATE TABLE Barrel
(
   -- Add columns
   barrel_ID int NOT NULL IDENTITY,
   oak_type varchar(10),
   -- Add constraints
   CONSTRAINT PK_Barrel PRIMARY KEY (Barrel_ID)

);
go

CREATE TABLE Clients
(
   -- Add columns
   client_ID int NOT NULL IDENTITY,
   client_type varchar(15) NOT NULL,
   client_name varchar(50) NOT NULL,
   [address] varchar(100) NOT NULL,
   telephone_number varchar(20) NOT NULL,
   sales_quantity int,
   -- Add constraints
   CONSTRAINT PK_Client PRIMARY KEY (client_ID),
   CONSTRAINT U1_Client UNIQUE ([address], telephone_number)
);
go

CREATE TABLE OrderStatus
(
   -- Add columns
   order_status_ID int NOT NULL IDENTITY,
   order_status varchar(20) NOT NULL,
   -- Add constraints
   CONSTRAINT PK_Order_Status PRIMARY KEY (order_status_ID)
);
go

CREATE TABLE VineyardVarietal
(
   -- Add columns
   vineyard_varietal_ID int NOT NULL IDENTITY,
   vineyard_ID int NOT NULL,
   varietal_ID int NOT NULL,
   -- Add constraints
   CONSTRAINT PK_VineyardVarietal PRIMARY KEY (vineyard_varietal_ID),
```

```
   CONSTRAINT FK1_VineyardVarietal FOREIGN KEY (vineyard_ID) REFERENCES Vineyard
(vineyard_ID),
   CONSTRAINT FK2_VineyardVarietal FOREIGN KEY (varietal_ID) REFERENCES Varietal
(varietal_ID)
);
go

CREATE TABLE VineyardVintage
(
   -- Add columns
   vineyard_vintage_ID int NOT NULL IDENTITY,
   vineyard_ID int NOT NULL,
   vintage_ID int NOT NULL,
   -- Add constraints
   CONSTRAINT PK_VineyardVintage PRIMARY KEY (vineyard_vintage_ID),
   CONSTRAINT FK1_VineyardVintage FOREIGN KEY (vineyard_ID) REFERENCES Vineyard
(vineyard_ID),
   CONSTRAINT FK2_VineyardVintage FOREIGN KEY (vintage_ID) REFERENCES Vintage
(vintage_ID)
);
go

CREATE TABLE Wine
(
   -- Add columns
   wine_ID int NOT NULL IDENTITY,
   varietal_ID int NOT NULL,
   vintage_ID int NOT NULL,
   barrel_ID int NOT NULL,
   wine_name varchar(50),
   price decimal(4,2) NOT NULL,
   quantity_sold int NOT NULL,
   average_rating decimal(3,2) NOT NULL,
   -- Add constraints
   CONSTRAINT PK_Wine PRIMARY KEY (wine_ID),
   CONSTRAINT FK1_Wine FOREIGN KEY (varietal_ID) REFERENCES Varietal (varietal_ID),
   CONSTRAINT FK2_Wine FOREIGN KEY (vintage_ID) REFERENCES Vintage (vintage_ID),
   CONSTRAINT FK3_Wine FOREIGN KEY (barrel_ID) REFERENCES Barrel (barrel_ID)
);
go


CREATE TABLE WineClient
(
   -- Add columns
   wine_client_ID int NOT NULL IDENTITY,
   wine_ID int NOT NULL,
   client_ID int NOT NULL,
   -- Add constraints
   CONSTRAINT PK_WineClient PRIMARY KEY (wine_client_ID),
   CONSTRAINT FK1_WineClient FOREIGN KEY (wine_ID) REFERENCES Wine (wine_ID),
   CONSTRAINT FK2_WineClient FOREIGN KEY (client_ID) REFERENCES Clients (client_ID)
);
go


CREATE TABLE OrderForm
(
```

```
   -- Add columns
   order_ID int NOT NULL IDENTITY,
   quantity int NOT NULL,
   -- Is datetime the right variable for this
   order_date datetime NOT NULL,
   wine_ID int NOT NULL,
   client_ID int NOT NULL,
   order_status_ID int NOT NULL,
   -- Add constraints
   CONSTRAINT PK_OrderForm PRIMARY KEY (order_id),
   CONSTRAINT FK1_OrderForm FOREIGN KEY (wine_ID) REFERENCES Wine (wine_ID),
   CONSTRAINT FK2_OrderForm FOREIGN KEY (client_ID) REFERENCES Clients (client_ID),
   CONSTRAINT FK3_OrderForm FOREIGN KEY (order_status_ID) REFERENCES OrderStatus
(order_status_ID)
);
go

/*
We will now insert records into tables.
*/

-- Insert vineyard names into the Vineyard Table.
INSERT Vineyard (vineyard_name)
VALUES  ('Violet Crown'), ('A Squared'), ('East Bay');
go

-- Insert varietal information into the Varietal table.
INSERT Varietal (varietal_name, varietal_color)
VALUES  ('Riesling', 'White'),
       ('Chardonnay', 'White'),
       ('Sauvignon Blanc', 'White'),
       ('Pinot Noir', 'Red'),
       ('Merlot', 'Red'),
       ('Cabernet Sauvignon', 'Red');
go

-- Insert the vintages into the Vintage table.
INSERT Vintage (vintage_year)
VALUES  (2016), (2017), (2018);
go

-- Insert the oak types into the Barrel table.
INSERT Barrel (oak_type)
VALUES  ('American'), ('French');
go

-- Insert the client information into the Clients table.
INSERT Clients (client_type, client_name, [address], telephone_number, sales_quantity)
VALUES  ('Restaurant', 'Flagship Steakhouse', '6579 Python Street', '(679)564-2458',
84809),
       ('Retail', 'Grape Savings Store', '3092 Java Drive', '(546)085-0188', 70066),
       ('Customer', 'Bill Dalton', '6831 Tableau Avenue', '(533)422-7465', NULL),
       ('Restaurant', 'The Red Apple', '2352 Unix Boulevard', '(684)279-6530', 64779),
       ('Retail', 'The Vine Dispatch', '3648 Oracle Avenue', '(257)208-4486', 50909),
       ('Customer', 'Margot Brown', '1923 Ruby Road', '(304)343-0332', NULL);
go

/*
```

```
As the winery plans to expand its operations, it will be
necessary to add new clients to the database. To make it
easier to add customers to the database, we have created
a simple form through Access as shown below.
*/
```




```
-- Insert the order statuses into the OrderStatus table.
INSERT OrderStatus (order_status)
VALUES ('OPEN'), ('CLOSED');
go

-- Insert values into the VineyardVarietal bridge table.
INSERT VineyardVarietal (vineyard_ID, varietal_ID)
VALUES  ((SELECT vineyard_ID FROM Vineyard WHERE vineyard_name = 'Violet Crown'), (SELECT
Varietal_ID FROM Varietal WHERE varietal_name = 'Merlot')),
       ((SELECT vineyard_ID FROM Vineyard WHERE vineyard_name = 'Violet Crown'), (SELECT
Varietal_ID FROM Varietal WHERE varietal_name = 'Cabernet Sauvignon')),
       ((SELECT vineyard_ID FROM Vineyard WHERE vineyard_name = 'A Squared'), (SELECT
Varietal_ID FROM Varietal WHERE varietal_name = 'Chardonnay')),
       ((SELECT vineyard_ID FROM Vineyard WHERE vineyard_name = 'A Squared'), (SELECT
Varietal_ID FROM Varietal WHERE varietal_name = 'Pinot Noir')),
       ((SELECT vineyard_ID FROM Vineyard WHERE vineyard_name = 'East Bay'), (SELECT
Varietal_ID FROM Varietal WHERE varietal_name = 'Riesling')),
       ((SELECT vineyard_ID FROM Vineyard WHERE vineyard_name = 'East Bay'), (SELECT
Varietal_ID FROM Varietal WHERE varietal_name = 'Sauvignon Blanc'));
go

-- Insert values into the VineyardVintage bridge table.
INSERT VineyardVintage (vineyard_ID, vintage_ID)
SELECT Vnrd.vineyard_ID, Vntge.vintage_ID
```

```
FROM (
   VALUES
   ('Violet Crown', 2016),
   ('Violet Crown', 2017),
   ('Violet Crown', 2018),
   ('A Squared', 2016),
   ('A Squared', 2017),
   ('A Squared', 2018),
   ('East Bay', 2016),
   ('East Bay', 2017),
   ('East Bay', 2018)
   ) AS VVintageConstruction(vineyard_name,vintage_year)
LEFT OUTER JOIN Vineyard AS Vnrd ON Vnrd.vineyard_name =
VVintageConstruction.vineyard_name
LEFT OUTER JOIN Vintage AS Vntge ON Vntge.vintage_year =
VVintageConstruction.vintage_year;
go

-- Insert values into the Wine Table
INSERT Wine (wine_name, varietal_ID, vintage_ID, barrel_ID, price, quantity_sold,
average_rating)
SELECT Wine_Name, Var.varietal_ID, Vntge.vintage_ID, Bar.barrel_ID, Price, Quantity_Sold,
Average_Rating
FROM (
   VALUES
   ('Riesling 2016', 'Riesling', 2016, 'French', 12.99, 62326, 1.92),
   ('Riesling 2017', 'Riesling', 2017, 'French', 15.99, 65754, 3.79),
   ('Riesling 2018', 'Riesling', 2018, 'French', 13.99, 69873, 1.21),
   ('Chardonnay 2016', 'Chardonnay', 2016, 'American', 16.99, 53589, 3.09),
   ('Chardonnay 2017', 'Chardonnay', 2017, 'American', 10.99, 52505, 2.17),
   ('Chardonnay 2018', 'Chardonnay', 2018, 'American', 20.99, 77774, 4.32),
   ('Sauvignon Blanc 2016', 'Sauvignon Blanc', 2016, 'American', 11.99, 67018, 3.83),
   ('Sauvignon Blanc 2017', 'Sauvignon Blanc', 2017, 'American', 12.99, 61501, 1.01),
   ('Sauvignon Blanc 2018', 'Sauvignon Blanc', 2018, 'American', 14.99, 81863, 4.70),
   ('Pinot Noir 2016', 'Pinot Noir', 2016, 'French', 18.99, 51426, 4.37),
   ('Pinot Noir 2017', 'Pinot Noir', 2017, 'French', 16.99, 91641, 2.03),
   ('Pinot Noir 2018', 'Pinot Noir', 2018, 'French', 22.99, 51293, 2.37),
   ('Merlot 2016', 'Merlot', 2016, 'French', 12.99, 73491, 3.16),
   ('Merlot 2017', 'Merlot', 2017, 'French', 10.99, 58546, 2.92),
   ('Merlot 2018', 'Merlot', 2018, 'French', 20.99, 51703, 2.69),
   ('Cabernet Sauvignon 2016', 'Cabernet Sauvignon', 2016, 'American', 26.99, 88856,
4.63),
   ('Cabernet Sauvignon 2017', 'Cabernet Sauvignon', 2017, 'American', 30.99, 79397,
3.31),
   ('Cabernet Sauvignon 2018', 'Cabernet Sauvignon', 2018, 'American', 28.99, 56086,
3.84)
   ) AS WineConstruction(wine_name, varietal_name, vintage_year, oak_type, price,
quantity_sold, average_rating)
LEFT OUTER JOIN Varietal AS Var ON Var.varietal_name = WineConstruction.varietal_name
LEFT OUTER JOIN Vintage AS Vntge ON Vntge.vintage_year = WineConstruction.vintage_year
LEFT OUTER JOIN Barrel AS Bar ON Bar.oak_type = WineConstruction.oak_type;
go

-- Insert values into the WineClient Table
INSERT WineClient (client_ID, wine_ID)
SELECT Cli.client_ID, Win.wine_ID
FROM (
   VALUES
```

```
   ('Flagship Steakhouse', 'Riesling 2017'),
   ('The Vine Dispatch', 'Pinot Noir 2017'),
   ('The Red Apple', 'Cabernet Sauvignon 2017'),
   ('Margot Brown', 'Pinot Noir 2018'),
   ('The Vine Dispatch', 'Sauvignon Blanc 2017'),
   ('Bill Dalton', 'Cabernet Sauvignon 2016'),
   ('The Vine Dispatch', 'Riesling 2018'),
   ('The Red Apple', 'Chardonnay 2018'),
   ('Grape Savings Store', 'Sauvignon Blanc 2016'),
   ('Bill Dalton', 'Merlot 2016')
   ) AS WinCliConstruction(client_name, wine_name)
LEFT OUTER JOIN Wine AS Win ON Win.wine_name = WinCliConstruction.wine_name
LEFT OUTER JOIN Clients AS Cli ON Cli.client_name = WinCliConstruction.client_name;
go


INSERT OrderForm (client_ID, wine_ID, quantity, order_date, order_status_ID)
SELECT Cli.client_ID, Win.wine_ID, Quantity, Order_Date, Stat.order_status_ID
FROM (
   VALUES
   ('Flagship Steakhouse', 'Riesling 2017', 240, '12-30-2019', 'OPEN'),
   ('The Vine Dispatch', 'Pinot Noir 2017', 204, '10-20-2019', 'CLOSED'),
   ('The Red Apple', 'Cabernet Sauvignon 2017', 156, '8-25-2019', 'CLOSED'),
   ('Margot Brown', 'Pinot Noir 2018', 7, '5-18-2019', 'CLOSED'),
   ('The Vine Dispatch', 'Sauvignon Blanc 2017', 216, '10-20-2019', 'CLOSED'),
   ('Bill Dalton', 'Cabernet Sauvignon 2016', 8, '9-5-2019', 'CLOSED'),
   ('The Vine Dispatch', 'Riesling 2018', 180, '10-20-2019', 'CLOSED'),
   ('The Red Apple', 'Chardonnay 2018', 204, '8-30-2019', 'CLOSED'),
   ('Grape Savings Store', 'Sauvignon Blanc 2016', 216, '7-20-2019', 'CLOSED'),
   ('Bill Dalton', 'Merlot 2016', 9, '8-12-2019', 'CLOSED')
   ) AS OrderConstruction(client_name, wine_name, quantity, order_date, order_status)
LEFT OUTER JOIN Clients AS Cli ON Cli.client_name = OrderConstruction.client_name
LEFT OUTER JOIN Wine AS Win ON Win.wine_name = OrderConstruction.wine_name
LEFT OUTER JOIN OrderStatus AS Stat On Stat.order_status =
OrderConstruction.order_status;
go
```

Data Manipulation:

```
/*
As we transition the data from the spreadsheets to the
database, we notice that the status of one of the orders
has been left open by accident. To fix this, we have
created a procedure that changes an order's status from
"OPEN" to "CLOSED." This procedure will also be used to
close future orders as well.
*/

/*
First, we will create a view of the OrderForm
table that will be easier to read.
*/
CREATE OR ALTER VIEW dbo.OrderForms
AS
SELECT
       OrderForm.order_ID AS OrderID,
```

```
        Clients.client_name AS ClientName,
        OrderForm.order_date AS [Date],
        Wine.wine_name AS WineName,
        OrderForm.quantity AS Quantity,
        OrderStatus.order_status AS [Status]
FROM OrderForm
JOIN Clients ON Clients.client_ID = OrderForm.client_ID
JOIN Wine ON Wine.wine_Id = OrderForm.wine_ID
JOIN OrderStatus ON OrderStatus.order_status_ID = OrderForm.order_status_ID
go

-- The first order is the one that is still open.
SELECT * FROM dbo.OrderForms
go
```

	OrderID	ClientName	Date	WineName	Quantity	Status
1	1	Flagship Steakhouse	2019-12-30 00:00:00.000	Riesling 2017	240	OPEN
2	2	The Vine Dispatch	2019-10-20 00:00:00.000	Pinot Noir 2017	204	CLOSED
3	3	The Red Apple	2019-08-25 00:00:00.000	Cabernet Sauvignon 2017	156	CLOSED
4	4	Margot Brown	2019-05-18 00:00:00.000	Pinot Noir 2018	7	CLOSED
5	5	The Vine Dispatch	2019-10-20 00:00:00.000	Sauvignon Blanc 2017	216	CLOSED
6	6	Bill Dalton	2019-09-05 00:00:00.000	Cabernet Sauvignon 2016	8	CLOSED
7	7	The Vine Dispatch	2019-10-20 00:00:00.000	Riesling 2018	180	CLOSED
8	8	The Red Apple	2019-08-30 00:00:00.000	Chardonnay 2018	204	CLOSED
9	9	Grape Savings Store	2019-07-20 00:00:00.000	Sauvignon Blanc 2016	216	CLOSED
10	10	Bill Dalton	2019-08-12 00:00:00.000	Merlot 2016	9	CLOSED

```
CREATE PROCEDURE dbo.ChangeStatus (@order_ID int)
AS
BEGIN
        DECLARE @status_ID int
        SELECT @status_ID = OrderForm.order_status_ID FROM OrderForm
        JOIN OrderStatus ON OrderStatus.order_status_ID = OrderForm.order_status_ID
        WHERE OrderStatus.order_status = 'CLOSED'

        UPDATE OrderForm
        SET    order_status_ID = @status_ID
        WHERE order_ID = @order_ID
END
go

DECLARE @OrderID int
SET @OrderID = 1
EXEC dbo.ChangeStatus 1
go

-- Check to see if the procedure was successful.
SELECT * FROM dbo.OrderForms
go
```

	OrderID	ClientName	Date	WineName	Quantity	Status
1	1	Flagship Steakhouse	2019-12-30 00:00:00.000	Riesling 2017	240	CLOSED
2	2	The Vine Dispatch	2019-10-20 00:00:00.000	Pinot Noir 2017	204	CLOSED
3	3	The Red Apple	2019-08-25 00:00:00.000	Cabernet Sauvignon 2017	156	CLOSED
4	4	Margot Brown	2019-05-18 00:00:00.000	Pinot Noir 2018	7	CLOSED
5	5	The Vine Dispatch	2019-10-20 00:00:00.000	Sauvignon Blanc 2017	216	CLOSED
6	6	Bill Dalton	2019-09-05 00:00:00.000	Cabernet Sauvignon 2016	8	CLOSED
7	7	The Vine Dispatch	2019-10-20 00:00:00.000	Riesling 2018	180	CLOSED
8	8	The Red Apple	2019-08-30 00:00:00.000	Chardonnay 2018	204	CLOSED
9	9	Grape Savings Store	2019-07-20 00:00:00.000	Sauvignon Blanc 2016	216	CLOSED
10	10	Bill Dalton	2019-08-12 00:00:00.000	Merlot 2016	9	CLOSED

```
/*
As time passes and more people are drinking the wines,
the rating of the wine will change.
This procedure will provide the means of updating the
average rating of the wines.
*/

CREATE PROCEDURE dbo.ChangeRating(@wine_name varchar(50), @newrating decimal(3,2))
AS
BEGIN
	UPDATE Wine SET average_rating = @newrating
	WHERE wine_name = @wine_name
END
go

/*
In this instance, the 2018 Chardonnay is not as well
precieved as it was when the average rating was last
calculated so we need to change the rating from a
4.32 to a 4.04
*/

-- Now we'll see if the procedure works as intended.
SELECT wine_name AS WineName, average_rating AS AverageRating
FROM Wine
WHERE wine_name = 'Chardonnay 2018'
go
```

Results Messages

	WineName	AverageRating
1	Chardonnay 2018	4.32

```
EXEC dbo.ChangeRating 'Chardonnay 2018', 4.04
go

SELECT wine_name AS WineName, average_rating AS AverageRating
FROM Wine
WHERE wine_name = 'Chardonnay 2018'
go
```

Results | Messages

	WineName	AverageRating
1	Chardonnay 2018	4.04

```
-- We will now begin to answer the data questions.

/*
Data Question 1: Are sales due to vineyard, varietal,
vintage, or oak type?

To answer data question 1, we will find the average rating
based on: vineyard rating, varietal rating, vintage rating,
and oak type rating
*/

-- Vineyard Rating Table
SELECT Vineyard.vineyard_name AS Vineyard, AVG(Wine.average_rating) AS VineyardRating
FROM Wine
JOIN Varietal ON Varietal.varietal_ID = Wine.varietal_ID
JOIN VineyardVarietal ON VineyardVarietal.varietal_ID = Varietal.varietal_ID
JOIN Vineyard ON Vineyard.vineyard_ID = VineyardVarietal.vineyard_ID
GROUP BY vineyard_name
ORDER BY AVG(Wine.average_rating) DESC;
go
```

	Vineyard	VineyardRating
1	Violet Crown	3.425000
2	A Squared	3.011666
3	East Bay	2.743333

```
-- Varietal rating table.
SELECT Varietal.varietal_name AS Varietal, AVG(Wine.average_rating) AS VarietalRating
FROM Wine
JOIN Varietal ON Varietal.varietal_ID = Wine.varietal_ID
GROUP BY varietal_name
ORDER BY AVG(Wine.average_rating) DESC;
go
```

	Varietal	VarietalRating
1	Cabernet Sauvignon	3.926666
2	Sauvignon Blanc	3.180000
3	Chardonnay	3.100000
4	Merlot	2.923333
5	Pinot Noir	2.923333
6	Riesling	2.306666

```
-- Vintage rating table.
SELECT Vintage.vintage_year AS Vintage, AVG(Wine.average_rating) AS VintageRating
FROM Wine
JOIN Vintage ON Vintage.vintage_ID = Wine.vintage_ID
```

```
GROUP BY vintage_year
ORDER BY AVG(Wine.average_rating) DESC;
go
```

	Vintage	VintageRating
1	2016	3.500000
2	2018	3.141666
3	2017	2.538333

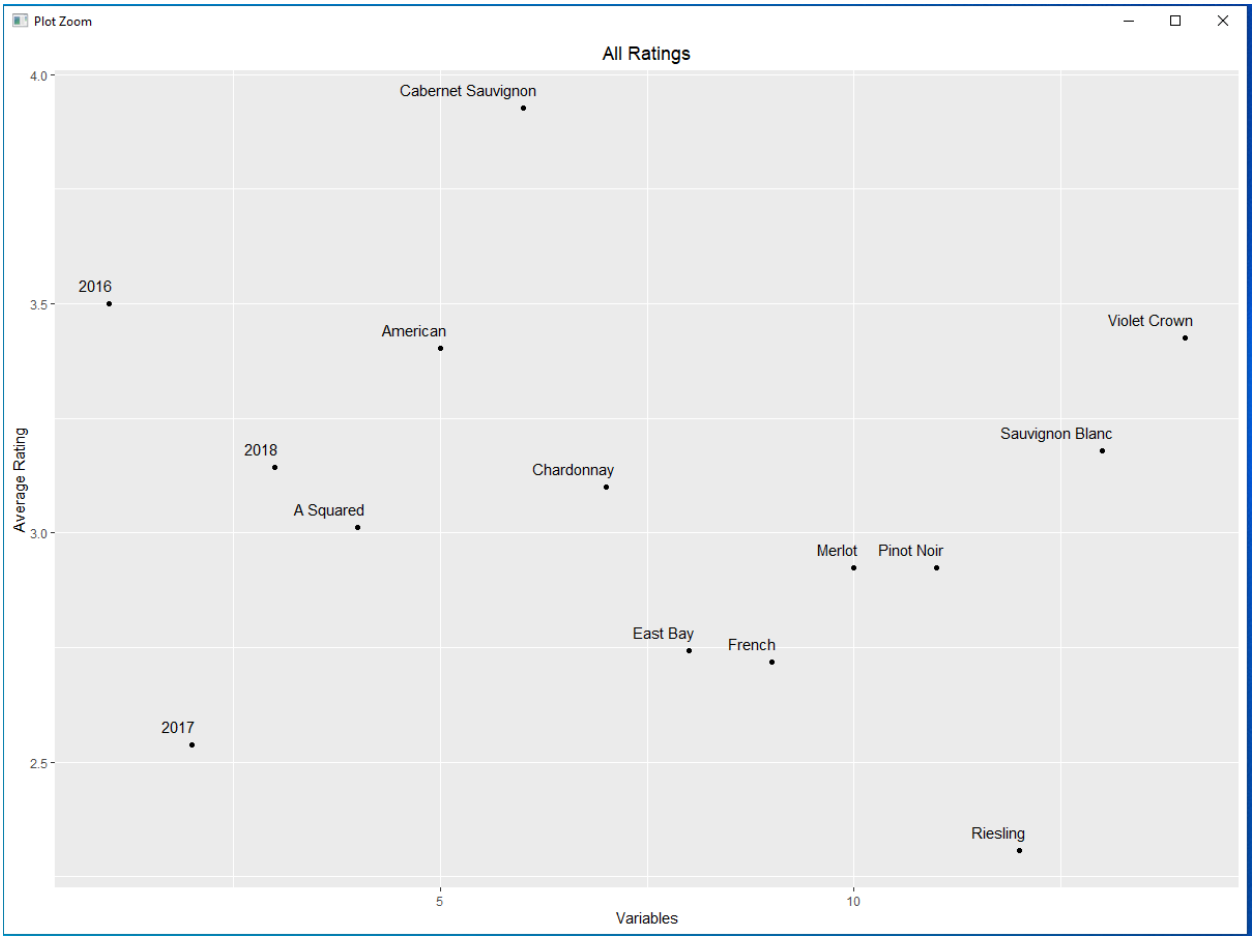
```
-- Oak Type Rating Table
SELECT Barrel.oak_type AS OakType, AVG(Wine.average_rating) AS BarrelRating
FROM Wine
JOIN Barrel ON Barrel.barrel_ID = Wine.barrel_ID
GROUP BY oak_type
ORDER BY AVG(Wine.average_rating) DESC;
go
```

	OakType	BarrelRating
1	American	3.402222
2	French	2.717777

```
-- This will create a view that puts all of the ratings in
-- one table.
CREATE VIEW dbo.AllRatings AS
	SELECT Vineyard.vineyard_name AS Variable, AVG(Wine.average_rating) AS Rating
	FROM Wine
	JOIN Varietal ON Varietal.varietal_ID = Wine.varietal_ID
	JOIN VineyardVarietal ON VineyardVarietal.varietal_ID = Varietal.varietal_ID
	JOIN Vineyard ON Vineyard.vineyard_ID = VineyardVarietal.vineyard_ID
	GROUP BY vineyard_name
	UNION
	-- Varietal rating table.
	SELECT Varietal.varietal_name AS Variable, AVG(Wine.average_rating) AS Rating
	FROM Wine
	JOIN Varietal ON Varietal.varietal_ID = Wine.varietal_ID
	GROUP BY varietal_name
	UNION
	-- Vintage rating table.
	SELECT CAST(Vintage.vintage_year AS char(4)) AS Variable, AVG(Wine.average_rating)
AS Rating
	FROM Wine
	JOIN Vintage ON Vintage.vintage_ID = Wine.vintage_ID
	GROUP BY vintage_year
	UNION
	-- Oak Type Rating Table
	SELECT Barrel.oak_type AS Variable, AVG(Wine.average_rating) AS Rating
	FROM Wine
	JOIN Barrel ON Barrel.barrel_ID = Wine.barrel_ID
	GROUP BY oak_type
go

SELECT * FROM dbo.AllRatings
ORDER BY Rating DESC
```

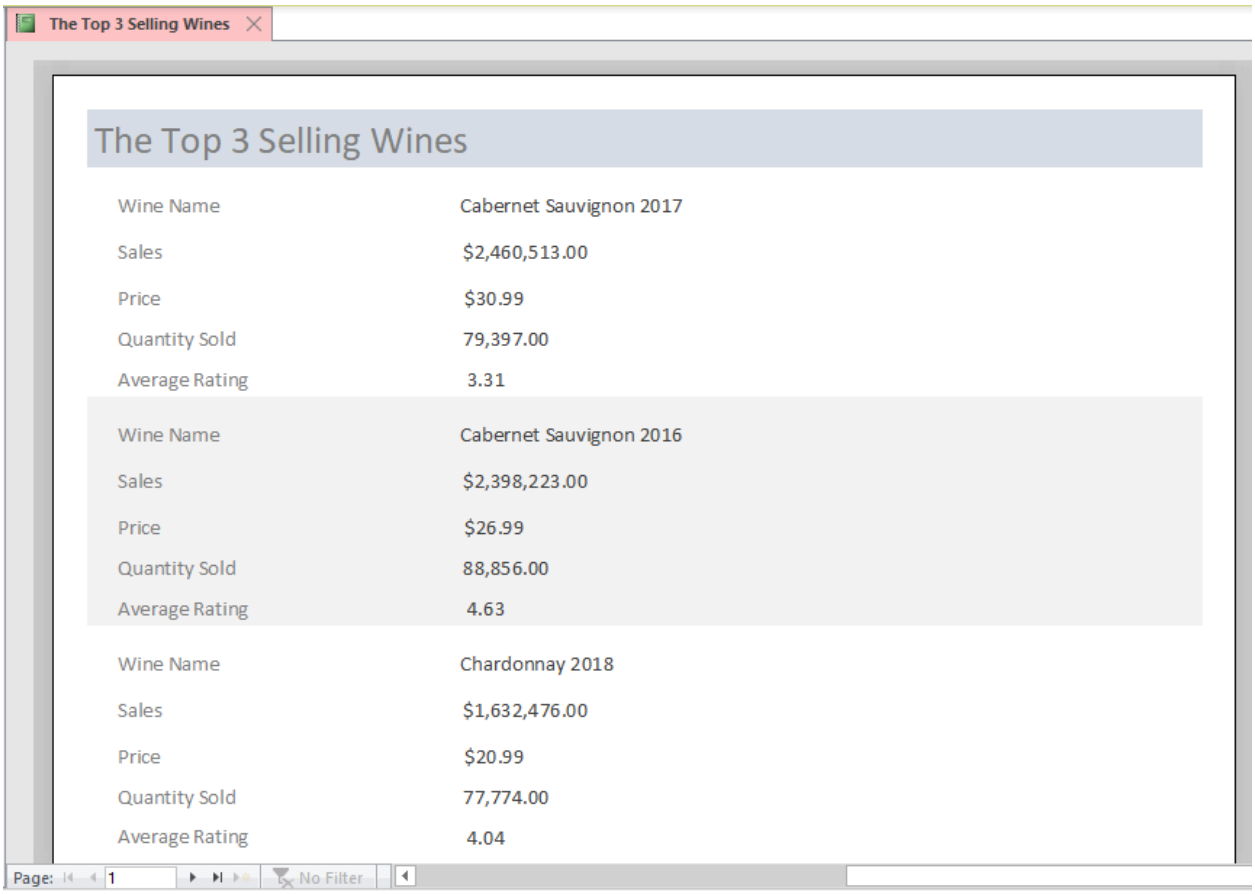
go

	Variable	Rating
1	Cabernet Sauvignon	3.926666
2	2016	3.500000
3	Violet Crown	3.425000
4	American	3.402222
5	Sauvignon Blanc	3.180000
6	2018	3.141666
7	Chardonnay	3.100000
8	A Squared	3.011666
9	Merlot	2.923333
10	Pinot Noir	2.923333
11	East Bay	2.743333
12	French	2.717777
13	2017	2.538333
14	Riesling	2.306666



```
/*
The variables seem to be well distributed with no
clear indication of any one factor determing the
rating of a wine. Cabernet Sauvignon was the
highest rated grape varietal and the highest
rated variable overall with a rating close to 4.
2016 was the highest rated vintage, Violet Crown
was the highest rated vineyard, and American was
the highest rated oak type. The majority of the ratings
fall into the 2.5-3.5 range with two exceptions:
The aforementioned Cabernet Sauvignon and Riesling,
which had a measly rating of around 2.3.
*/


/*
Data Question 2: Which wines sell the best?

To answer this question, we will create a function called
"WineSalesFunction" that multiples the "price" and the
"quantity_sold" objects from the Wine table to get the
total sales of that wine. We will then select the top 3
wines based on their total sales. This select statement
will be created as a view in order to create a report in
access.
*/

Create Function dbo.WineSalesFunction (@wine_id int)
RETURNS decimal AS
BEGIN
	-- Declares a return value
	DECLARE @returnValue decimal

	-- Multiplies the quantity sold and the price to
	-- obtain the total sales
	SELECT @returnValue = quantity_sold * price FROM Wine
	WHERE Wine.wine_ID = @wine_id

	RETURN @returnValue
END
go

CREATE VIEW dbo.Top3Wines AS
	SELECT TOP 3
	wine_name AS WineName,
	price AS Price,
	quantity_sold AS QuantitySold,
	average_rating AS AverageRating,
	dbo.WineSalesFunction(wine_ID) AS Sales
	FROM Wine
	ORDER BY SALES DESC
go

SELECT * FROM dbo.Top3Wines
go
```

	WineName	Price	QuantitySold	AverageRating	Sales
1	Cabernet Sauvignon 2017	30.99	79397	3.31	2460513
2	Cabernet Sauvignon 2016	26.99	88856	4.63	2398223
3	Chardonnay 2018	20.99	77774	4.04	1632476

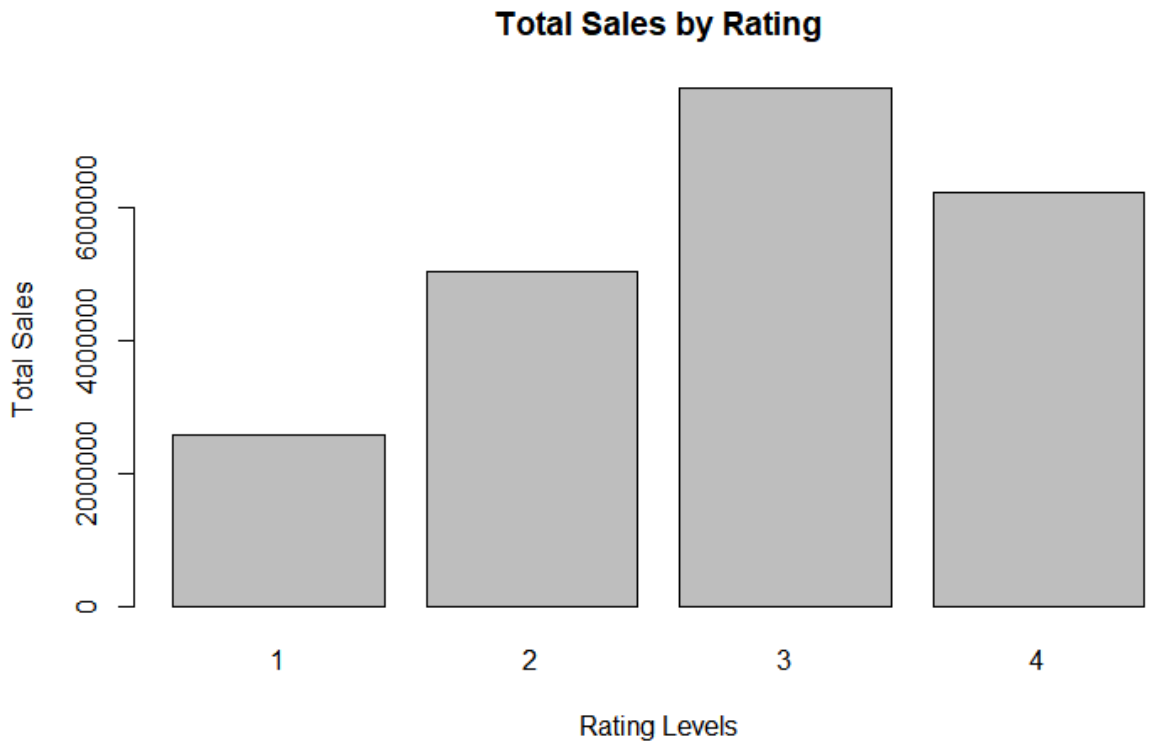



```
/*
Here we see that two of the Cabernets sold the best
followed by the latest vintage of the Chardonnay.
One thing to keep in mind is that the 2016 Cabernet
sold nearly 10,000 more units than the 2017 vintage,
but because the 2017 was four dollars more it sold
better overall.
*/

/*
Data Question 3: Do the ratings and the sales of a wine
correlate?

To answer this question, we will select the average rating
from the Wine table and cast the ratings as integers.

For example:
Wines with a rating of 1.00-1.99 will be floored to 1.
Wines with a rating of 2.00-2.99 will be floored to 2.
and so on...
```

```
This will effectively create the rating levels.

We will also select the sum of the WineSalesFunction
outputs from the Wine table and group by our predefined
levels.
*/

SELECT cast(average_rating as int) AS RatingLevel, SUM(dbo.WineSalesFunction(wine_ID)) AS
TotalSales FROM Wine
GROUP BY cast(average_rating as int)
go
```

Results | Messages

	RatingLevel	TotalSales
1	1	2586036
2	2	5041904
3	3	7806523
4	4	6234405

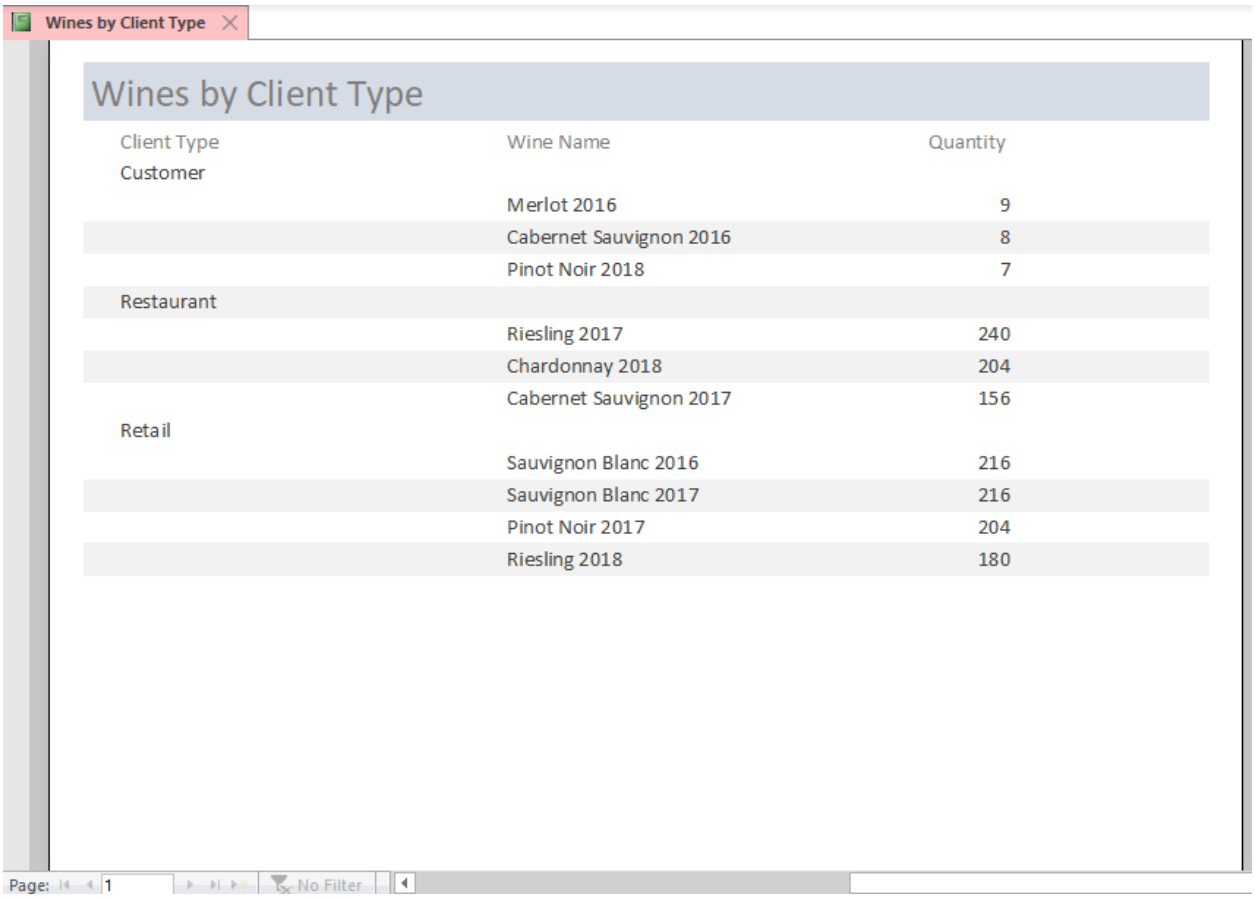



```
/*
There is a general correlation between the rating of a
wine and its sales. Wines rated as a 3 or a 4 sell
significantly better than those rated a 2 or a 1.
*/

/*
Data Question 4: Which wines sell best for each client
```

```
type?

To answer this question, we will create a view called
"ClientTypes" that displays the client_name,
the client_type, the wine_name,
and the quantity from the OrderForm table.
The results were ordered first by client_type
and then by quantity in a SELECT statement.
*/

CREATE VIEW dbo.ClientTypes AS
	SELECT client_name AS ClientName, client_type AS ClientType, wine_name WineName,
quantity AS Quantity
	FROM OrderForm
	JOIN Clients ON Clients.client_ID = OrderForm.client_ID
	JOIN Wine ON Wine.wine_Id = OrderForm.wine_ID
go

SELECT * FROM dbo.ClientTypes
ORDER BY ClientType DESC, Quantity DESC
go
```

Results | Messages

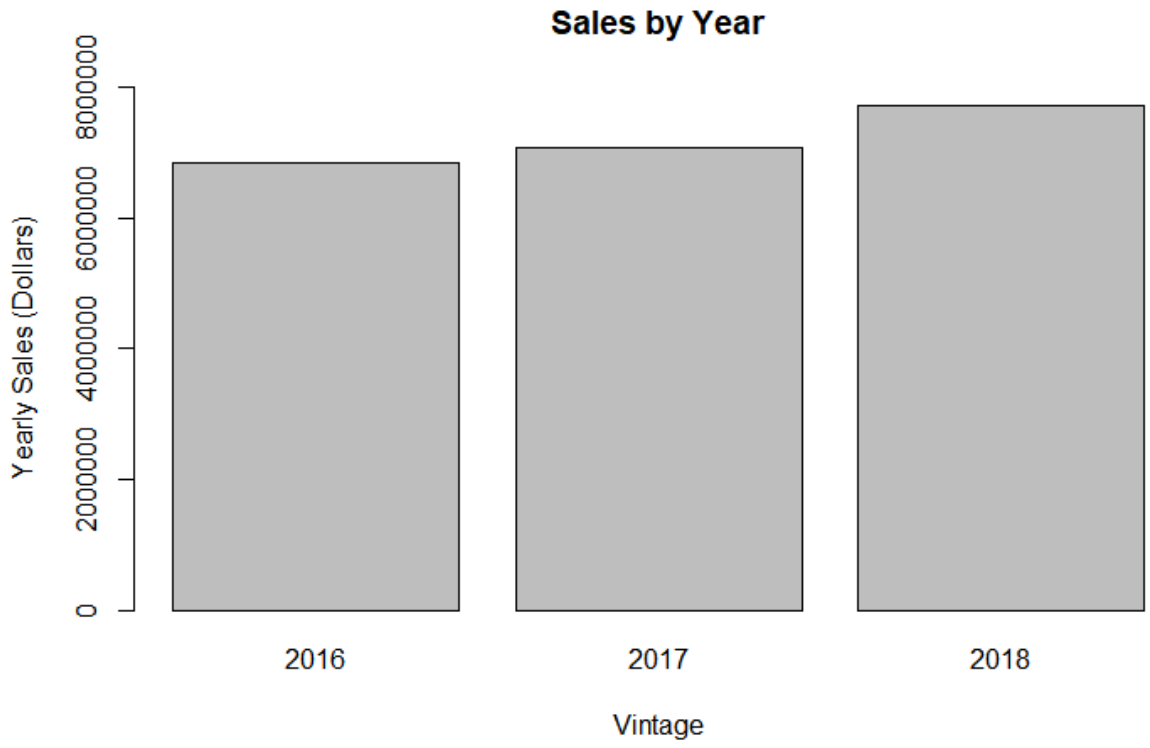
	ClientName	ClientType	WineName	Quantity
1	The Vine Dispatch	Retail	Sauvignon Blanc 2017	216
2	Grape Savings Store	Retail	Sauvignon Blanc 2016	216
3	The Vine Dispatch	Retail	Pinot Noir 2017	204
4	The Vine Dispatch	Retail	Riesling 2018	180
5	Flagship Steakhouse	Restaurant	Riesling 2017	240
6	The Red Apple	Restaurant	Chardonnay 2018	204
7	The Red Apple	Restaurant	Cabernet Sauvignon 2017	156
8	Bill Dalton	Customer	Merlot 2016	9
9	Bill Dalton	Customer	Cabernet Sauvignon 2016	8
10	Margot Brown	Customer	Pinot Noir 2018	7




```
/*
What the report shows is that red wines are more popular
amongst customers while in retail/restaurants, whites
are slightly favored over reds.
*/

/*
Data Question 5: How do the total sales of a wine change
over time?

To answer this question, we will once again use the
"WineSalesFunction" to sum up the total sales of all the
wines for a given vintage.
*/
SELECT Vintage.vintage_year AS Vintage, SUM(dbo.WineSalesFunction(wine_ID)) AS
YearlySales
FROM Wine
LEFT JOIN Vintage ON Vintage.vintage_ID = Wine.vintage_ID
GROUP BY Vintage.vintage_year
go
```

Results | Messages

	Vintage	YearlySales
1	2016	6853089
2	2017	7088249
3	2018	7727530




```
/*
We see a gradual increase in the sales from year to year.
One thing to note is that the increase from 2017 to 2018
was noticeably larger than the increase from 2016 to
2017. This may be a sign of the exponential growth of the
winery.
*/
```

Reflection:

At the start of the project, I wanted to make a very realistic database that accurately accounted for all of the moving parts that make up a winery. It was during the modeling portion of the project that I realized this would not be feasible in the time given. For example, some of the tables I had modeled in Milestone 1 had to be omitted in Milestone 2 because they were not beneficial in answering the data questions. Also, I felt that there are other ways in which I could

manipulate the data such as creating a procedure to allowed users to add new clients/orders but there was not enough time.

During the course of this project, I gained an appreciation of the time and effort that it takes to create a real-world database that is functional. The fact I did not get to implement everything I wanted, though disappointing, was also encouraging because I now see the potential of what Databases can accomplish. As a result, I now have an assortment of skills that I can use the next time I am working on a project in SQL or any other RDBMS.

Summary:

Throughout this project, it became clear what the main goal was, to use SQL to answer a set of business questions. First, the stakeholders and the business rules must be established to create proper questions that will most benefit the business. Conceptual/logical models are then made to show how the tables relate to each other. It is important that these tables are normalized and that foreign keys/unique constraints are well defined as it will make data entry much easier. Once the tables are created and the data inserted, we need to be able to visualize and manipulate the data in order to answer our business questions. This is accomplished through the use of SELECT statements and CRUD operations as well as Functions, Views, and Stored Procedures. Finally, we need to present our answers in ways that are easy and accessible to the business. We are able to utilize applications like Microsoft Access and R Studio to creates forms and charts that aid our stakeholders in making business decisions.

Appendix Code:

```
/*
The following drops will assist in making a script
that can be repeated without error.
*/

-- Procedure Drops
```

```
DROP PROCEDURE IF EXISTS dbo.ChangeRating;

go

DROP PROCEDURE IF EXISTS dbo.ChangeStatus;

go

-- View Drops

DROP VIEW IF EXISTS dbo.ClientTypes;

go

DROP VIEW IF EXISTS dbo.Top3Wines;

go

DROP VIEW IF EXISTS dbo.AllRatings;

go

DROP VIEW IF EXISTS dbo.OrderForms;

go

-- Funnction Drops

DROP FUNCTION IF EXISTS dbo.WineSalesFunction;

go
```

```
-- Table Drops

DROP TABLE IF EXISTS OrderForm;

go


DROP TABLE IF EXISTS WineClient;

go


DROP TABLE IF EXISTS Wine;

go


DROP TABLE IF EXISTS VineyardVintage;

go


DROP TABLE IF EXISTS VineyardVarietal;

go


DROP TABLE IF EXISTS OrderStatus;

go


DROP TABLE IF EXISTS Clients;

go


DROP TABLE IF EXISTS Barrel;
```

```
go

DROP TABLE IF EXISTS Vintage;
go

DROP TABLE IF EXISTS Varietal;
go

DROP TABLE IF EXISTS Vineyard;
go

-- create all tables in order of their dependencies

CREATE TABLE Vineyard
(
  -- Add columns
  vineyard_ID int NOT NULL IDENTITY,
  vineyard_name varchar(30) NOT NULL,
  -- Add constraints
  CONSTRAINT PK_Vineyard PRIMARY KEY (vineyard_ID),
  CONSTRAINT U1_Vineyard UNIQUE (vineyard_name)
);
go
```

```
CREATE TABLE Varietal
(
  -- Add columns
  varietal_ID int NOT NULL IDENTITY,
  varietal_name varchar(30) NOT NULL,
  -- Add constraints
  varietal_color varchar(10) NOT NULL,
  CONSTRAINT PK_Varietal PRIMARY KEY (varietal_ID),
  CONSTRAINT U1_Varietal UNIQUE (varietal_name)
);
go

CREATE TABLE Vintage
(
  -- Add columns
  vintage_ID int NOT NULL IDENTITY,
  vintage_year int NOT NULL,
  -- Add constraints
  CONSTRAINT PK_Vintage PRIMARY KEY (vintage_ID)

);
go
```

```
CREATE TABLE Barrel
(
  -- Add columns
  barrel_ID int NOT NULL IDENTITY,
  oak_type varchar(10),
  -- Add constraints
  CONSTRAINT PK_Barrel PRIMARY KEY (Barrel_ID)

);
go

CREATE TABLE Clients
(
  -- Add columns
  client_ID int NOT NULL IDENTITY,
  client_type varchar(15) NOT NULL,
  client_name varchar(50) NOT NULL,
  [address] varchar(100) NOT NULL,
  telephone_number varchar(20) NOT NULL,
  sales_quantity int,
  -- Add constraints
  CONSTRAINT PK_Client PRIMARY KEY (client_ID),
```

```
  CONSTRAINT U1_Client UNIQUE ([address], telephone_number)
);
go

CREATE TABLE OrderStatus
(
  -- Add columns
  order_status_ID int NOT NULL IDENTITY,
  order_status varchar(20) NOT NULL,
  -- Add constraints
  CONSTRAINT PK_Order_Status PRIMARY KEY (order_status_ID)
);
go

CREATE TABLE VineyardVarietal
(
  -- Add columns
  vineyard_varietal_ID int NOT NULL IDENTITY,
  vineyard_ID int NOT NULL,
  varietal_ID int NOT NULL,
  -- Add constraints
  CONSTRAINT PK_VineyardVarietal PRIMARY KEY (vineyard_varietal_ID),
```

```
  CONSTRAINT FK1_VineyardVarietal FOREIGN KEY (vineyard_ID) REFERENCES Vineyard (vineyard_ID),
  CONSTRAINT FK2_VineyardVarietal FOREIGN KEY (varietal_ID) REFERENCES Varietal (varietal_ID)
);
go

CREATE TABLE VineyardVintage
(
  -- Add columns
  vineyard_vintage_ID int NOT NULL IDENTITY,
  vineyard_ID int NOT NULL,
  vintage_ID int NOT NULL,
  -- Add constraints
  CONSTRAINT PK_VineyardVintage PRIMARY KEY (vineyard_vintage_ID),
  CONSTRAINT FK1_VineyardVintage FOREIGN KEY (vineyard_ID) REFERENCES Vineyard (vineyard_ID),
  CONSTRAINT FK2_VineyardVintage FOREIGN KEY (vintage_ID) REFERENCES Vintage (vintage_ID)
);
go

CREATE TABLE Wine
(
```

```
  -- Add columns
  wine_ID int NOT NULL IDENTITY,
  varietal_ID int NOT NULL,
  vintage_ID int NOT NULL,
  barrel_ID int NOT NULL,
  wine_name varchar(50),
  price decimal(4,2) NOT NULL,
  quantity_sold int NOT NULL,
  average_rating decimal(3,2) NOT NULL,
  -- Add constraints
  CONSTRAINT PK_Wine PRIMARY KEY (wine_ID),
  CONSTRAINT FK1_Wine FOREIGN KEY (varietal_ID) REFERENCES Varietal
(varietal_ID),
  CONSTRAINT FK2_Wine FOREIGN KEY (vintage_ID) REFERENCES Vintage
(vintage_ID),
  CONSTRAINT FK3_Wine FOREIGN KEY (barrel_ID) REFERENCES Barrel (barrel_ID)
);
go



CREATE TABLE WineClient
(
  -- Add columns
```

```
  wine_client_ID int NOT NULL IDENTITY,
  wine_ID int NOT NULL,
  client_ID int NOT NULL,
  -- Add constraints
  CONSTRAINT PK_WineClient PRIMARY KEY (wine_client_ID),
  CONSTRAINT FK1_WineClient FOREIGN KEY (wine_ID) REFERENCES Wine
(wine_ID),
  CONSTRAINT FK2_WineClient FOREIGN KEY (client_ID) REFERENCES Clients
(client_ID)
);
go



CREATE TABLE OrderForm
(
  -- Add columns
  order_ID int NOT NULL IDENTITY,
  quantity int NOT NULL,
  -- Is datetime the right variable for this
  order_date datetime NOT NULL,
  wine_ID int NOT NULL,
  client_ID int NOT NULL,
  order_status_ID int NOT NULL,
```

```
  -- Add constraints
  CONSTRAINT PK_OrderForm PRIMARY KEY (order_id),
  CONSTRAINT FK1_OrderForm FOREIGN KEY (wine_ID) REFERENCES Wine
(wine_ID),
  CONSTRAINT FK2_OrderForm FOREIGN KEY (client_ID) REFERENCES Clients
(client_ID),
  CONSTRAINT FK3_OrderForm FOREIGN KEY (order_status_ID) REFERENCES
OrderStatus (order_status_ID)
);
go

/*
We will now insert records into tables.
*/

-- Insert vineyard names into the Vineyard Table.
INSERT Vineyard (vineyard_name)
VALUES  ('Violet Crown'), ('A Squared'), ('East Bay');
go

-- Insert varietal information into the Varietal table.
INSERT Varietal (varietal_name, varietal_color)
VALUES  ('Riesling', 'White'),
```

```
        ('Chardonnay', 'White'),
        ('Sauvignon Blanc', 'White'),
        ('Pinot Noir', 'Red'),
        ('Merlot', 'Red'),
        ('Cabernet Sauvignon', 'Red');
go

-- Insert the vintages into the Vintage table.
INSERT Vintage (vintage_year)
VALUES  (2016), (2017), (2018);
go

-- Insert the oak types into the Barrel table.
INSERT Barrel (oak_type)
VALUES  ('American'), ('French');
go

-- Insert the client information into the Clients table.
INSERT Clients (client_type, client_name, [address], telephone_number, sales_quantity)
VALUES  ('Restaurant', 'Flagship Steakhouse', '6579 Python Street', '(679)564-2458', 84809),
        ('Retail', 'Grape Savings Store', '3092 Java Drive', '(546)085-0188', 70066),
        ('Customer', 'Bill Dalton', '6831 Tableau Avenue', '(533)422-7465', NULL),
        ('Restaurant', 'The Red Apple', '2352 Unix Boulevard', '(684)279-6530', 64779),
```

```
    ('Retail', 'The Vine Dispatch', '3648 Oracle Avenue', '(257)208-4486', 50909),
    ('Customer', 'Margot Brown', '1923 Ruby Road', '(304)343-0332', NULL);
go

-- Insert the order statuses into the OrderStatus table.
INSERT OrderStatus (order_status)
VALUES ('OPEN'), ('CLOSED');
go

-- Insert values into the VineyardVarietal bridge table.
INSERT VineyardVarietal (vineyard_ID, varietal_ID)
VALUES  ((SELECT vineyard_ID FROM Vineyard WHERE vineyard_name = 'Violet Crown'), (SELECT Varietal_ID FROM Varietal WHERE varietal_name = 'Merlot')),
    ((SELECT vineyard_ID FROM Vineyard WHERE vineyard_name = 'Violet Crown'), (SELECT Varietal_ID FROM Varietal WHERE varietal_name = 'Cabernet Sauvignon')),
    ((SELECT vineyard_ID FROM Vineyard WHERE vineyard_name = 'A Squared'), (SELECT Varietal_ID FROM Varietal WHERE varietal_name = 'Chardonnay')),
    ((SELECT vineyard_ID FROM Vineyard WHERE vineyard_name = 'A Squared'), (SELECT Varietal_ID FROM Varietal WHERE varietal_name = 'Pinot Noir')),
    ((SELECT vineyard_ID FROM Vineyard WHERE vineyard_name = 'East Bay'), (SELECT Varietal_ID FROM Varietal WHERE varietal_name = 'Riesling')),
    ((SELECT vineyard_ID FROM Vineyard WHERE vineyard_name = 'East Bay'), (SELECT Varietal_ID FROM Varietal WHERE varietal_name = 'Sauvignon Blanc'));
go
```

```
-- Insert values into the VineyardVintage bridge table.
INSERT VineyardVintage (vineyard_ID, vintage_ID)
SELECT Vnrd.vineyard_ID, Vntge.vintage_ID
FROM (
  VALUES
  ('Violet Crown', 2016),
  ('Violet Crown', 2017),
  ('Violet Crown', 2018),
  ('A Squared', 2016),
  ('A Squared', 2017),
  ('A Squared', 2018),
  ('East Bay', 2016),
  ('East Bay', 2017),
  ('East Bay', 2018)
  ) AS VVintageConstruction(vineyard_name,vintage_year)
LEFT OUTER JOIN Vineyard AS Vnrd ON Vnrd.vineyard_name =
VVintageConstruction.vineyard_name
LEFT OUTER JOIN Vintage AS Vntge ON Vntge.vintage_year =
VVintageConstruction.vintage_year;
go


-- Insert values into the Wine Table
```

```
INSERT Wine (wine_name, varietal_ID, vintage_ID, barrel_ID, price, quantity_sold,
average_rating)

SELECT Wine_Name, Vari.varietal_ID, Vntge.vintage_ID, Bar.barrel_ID, Price, Quantity_Sold,
Average_Rating

FROM (

  VALUES

  ('Riesling 2016', 'Riesling', 2016, 'French', 12.99, 62326, 1.92),

  ('Riesling 2017', 'Riesling', 2017, 'French', 15.99, 65754, 3.79),

  ('Riesling 2018', 'Riesling', 2018, 'French', 13.99, 69873, 1.21),

  ('Chardonnay 2016', 'Chardonnay', 2016, 'American', 16.99, 53589, 3.09),

  ('Chardonnay 2017', 'Chardonnay', 2017, 'American', 10.99, 52505, 2.17),

  ('Chardonnay 2018', 'Chardonnay', 2018, 'American', 20.99, 77774, 4.32),

  ('Sauvignon Blanc 2016', 'Sauvignon Blanc', 2016, 'American', 11.99, 67018, 3.83),

  ('Sauvignon Blanc 2017', 'Sauvignon Blanc', 2017, 'American', 12.99, 61501, 1.01),

  ('Sauvignon Blanc 2018', 'Sauvignon Blanc', 2018, 'American', 14.99, 81863, 4.70),

  ('Pinot Noir 2016', 'Pinot Noir', 2016, 'French', 18.99, 51426, 4.37),

  ('Pinot Noir 2017', 'Pinot Noir', 2017, 'French', 16.99, 91641, 2.03),

  ('Pinot Noir 2018', 'Pinot Noir', 2018, 'French', 22.99, 51293, 2.37),

  ('Merlot 2016', 'Merlot', 2016, 'French', 12.99, 73491, 3.16),

  ('Merlot 2017', 'Merlot', 2017, 'French', 10.99, 58546, 2.92),

  ('Merlot 2018', 'Merlot', 2018, 'French', 20.99, 51703, 2.69),

  ('Cabernet Sauvignon 2016', 'Cabernet Sauvignon', 2016, 'American', 26.99, 88856, 4.63),

  ('Cabernet Sauvignon 2017', 'Cabernet Sauvignon', 2017, 'American', 30.99, 79397, 3.31),
```

```
  ('Cabernet Sauvignon 2018', 'Cabernet Sauvignon', 2018, 'American', 28.99, 56086, 3.84)
  ) AS WineConstruction(wine_name, varietal_name, vintage_year, oak_type, price,
quantity_sold, average_rating)
LEFT OUTER JOIN Varietal AS Vari ON Vari.varietal_name =
WineConstruction.varietal_name
LEFT OUTER JOIN Vintage AS Vntge ON Vntge.vintage_year =
WineConstruction.vintage_year
LEFT OUTER JOIN Barrel AS Bar ON Bar.oak_type = WineConstruction.oak_type;
go


-- Insert values into the WineClient Table
INSERT WineClient (client_ID, wine_ID)
SELECT Cli.client_ID, Win.wine_ID
FROM (
  VALUES
  ('Flagship Steakhouse', 'Riesling 2017'),
  ('The Vine Dispatch', 'Pinot Noir 2017'),
  ('The Red Apple', 'Cabernet Sauvignon 2017'),
  ('Margot Brown', 'Pinot Noir 2018'),
  ('The Vine Dispatch', 'Sauvignon Blanc 2017'),
  ('Bill Dalton', 'Cabernet Sauvignon 2016'),
  ('The Vine Dispatch', 'Riesling 2018'),
  ('The Red Apple', 'Chardonnay 2018'),
```

```
  ('Grape Savings Store', 'Sauvignon Blanc 2016'),
  ('Bill Dalton', 'Merlot 2016')
  ) AS WinCliConstruction(client_name, wine_name)
LEFT OUTER JOIN Wine AS Win ON Win.wine_name = WinCliConstruction.wine_name
LEFT OUTER JOIN Clients AS Cli ON Cli.client_name = WinCliConstruction.client_name;
go



INSERT OrderForm (client_ID, wine_ID, quantity, order_date, order_status_ID)
SELECT Cli.client_ID, Win.wine_ID, Quantity, Order_Date, Stat.order_status_ID
FROM (
  VALUES
  ('Flagship Steakhouse', 'Riesling 2017', 240, '12-30-2019', 'OPEN'),
  ('The Vine Dispatch', 'Pinot Noir 2017', 204, '10-20-2019', 'CLOSED'),
  ('The Red Apple', 'Cabernet Sauvignon 2017', 156, '8-25-2019', 'CLOSED'),
  ('Margot Brown', 'Pinot Noir 2018', 7, '5-18-2019', 'CLOSED'),
  ('The Vine Dispatch', 'Sauvignon Blanc 2017', 216, '10-20-2019', 'CLOSED'),
  ('Bill Dalton', 'Cabernet Sauvignon 2016', 8, '9-5-2019', 'CLOSED'),
  ('The Vine Dispatch', 'Riesling 2018', 180, '10-20-2019', 'CLOSED'),
  ('The Red Apple', 'Chardonnay 2018', 204, '8-30-2019', 'CLOSED'),
  ('Grape Savings Store', 'Sauvignon Blanc 2016', 216, '7-20-2019', 'CLOSED'),
  ('Bill Dalton', 'Merlot 2016', 9, '8-12-2019', 'CLOSED')
  ) AS OrderConstruction(client_name, wine_name, quantity, order_date, order_status)
```

```
LEFT OUTER JOIN Clients AS Cli ON Cli.client_name = OrderConstruction.client_name
LEFT OUTER JOIN Wine AS Win ON Win.wine_name = OrderConstruction.wine_name
LEFT OUTER JOIN OrderStatus AS Stat On Stat.order_status =
OrderConstruction.order_status;
go

/*
As we transition the data from the spreadsheets to the
database, we notice that the status of one of the orders
has been left open by accident. To fix this, we have
created a procedure that changes an order's status from
"OPEN" to "CLOSED." This procedure will also be used to
close future orders as well.
*/

/*
First, we will create a view of the OrderForm
table that will be easier to read.
*/
CREATE OR ALTER VIEW dbo.OrderForms
AS
SELECT
        OrderForm.order_ID AS OrderID,
```

```
        Clients.client_name AS ClientName,
        OrderForm.order_date AS [Date],
        Wine.wine_name AS WineName,
        OrderForm.quantity AS Quantity,
        OrderStatus.order_status AS [Status]
FROM OrderForm
JOIN Clients ON Clients.client_ID = OrderForm.client_ID
JOIN Wine ON Wine.wine_Id = OrderForm.wine_ID
JOIN OrderStatus ON OrderStatus.order_status_ID = OrderForm.order_status_ID
go

-- The first order is the one that is still open.
SELECT * FROM dbo.OrderForms
go

CREATE PROCEDURE dbo.ChangeStatus (@order_ID int)
AS
BEGIN
    DECLARE @status_ID int
    SELECT @status_ID = OrderForm.order_status_ID FROM OrderForm
    JOIN OrderStatus ON OrderStatus.order_status_ID = OrderForm.order_status_ID
    WHERE OrderStatus.order_status = 'CLOSED'
```

```
    UPDATE OrderForm
    SET    order_status_ID = @status_ID
    WHERE order_ID = @order_ID
END
go

DECLARE @OrderID int
SET @OrderID = 1
EXEC dbo.ChangeStatus 1
go

-- Check to see if the procedure was successful.
SELECT * FROM dbo.OrderForms
go

/*
As time passes and more people are drinking the wines,
the rating of the winery will change.
This procedure will provide the means of updating the
average rating of the wines.
*/
```

```
CREATE PROCEDURE dbo.ChangeRating(@wine_name varchar(50), @newrating
decimal(3,2))
AS
BEGIN
    UPDATE Wine SET average_rating = @newrating
    WHERE wine_name = @wine_name
END
go

/*
In this instance, the 2018 Chardonnay is not as well
precieved as it was when the average rating was last
calculated so we need to change the rating from a
4.32 to a 4.04
*/

-- Now we'll see if the procedure works as intended.
SELECT wine_name AS WineName, average_rating AS AverageRating
FROM Wine
WHERE wine_name = 'Chardonnay 2018'
go

EXEC dbo.ChangeRating 'Chardonnay 2018', 4.04
```

```
go

SELECT wine_name AS WineName, average_rating AS AverageRating
FROM Wine
WHERE wine_name = 'Chardonnay 2018'
go

-- We will now begin to answer the data questions.

/*
Data Question 1: Are sales due to vineyard, varietal,
vintage, or oak type?

To answer data question 1, we will find the average rating
based on: vineyard rating, varietal rating, vintage rating,
and oak type rating
*/

-- Vineyard Rating Table
SELECT Vineyard.vineyard_name AS Vineyard, AVG(Wine.average_rating) AS
VineyardRating
FROM Wine
JOIN Varietal ON Varietal.varietal_ID = Wine.varietal_ID
```

```
JOIN VineyardVarietal ON VineyardVarietal.varietal_ID = Varietal.varietal_ID

JOIN Vineyard ON Vineyard.vineyard_ID = VineyardVarietal.vineyard_ID

GROUP BY vineyard_name

ORDER BY AVG(Wine.average_rating) DESC;

go


-- Varietal rating table.

SELECT Varietal.varietal_name AS Varietal, AVG(Wine.average_rating) AS VarietalRating

FROM Wine

JOIN Varietal ON Varietal.varietal_ID = Wine.varietal_ID

GROUP BY varietal_name

ORDER BY AVG(Wine.average_rating) DESC;

go


-- Vintage rating table.

SELECT Vintage.vintage_year AS Vintage, AVG(Wine.average_rating) AS VintageRating

FROM Wine

JOIN Vintage ON Vintage.vintage_ID = Wine.vintage_ID

GROUP BY vintage_year

ORDER BY AVG(Wine.average_rating) DESC;

go


-- Oak Type Rating Table
```

```
SELECT Barrel.oak_type AS OakType, AVG(Wine.average_rating) AS BarrelRating
FROM Wine
JOIN Barrel ON Barrel.barrel_ID = Wine.barrel_ID
GROUP BY oak_type
ORDER BY AVG(Wine.average_rating) DESC;
go

-- This will create a view that puts all of the ratings in
-- one table.
CREATE VIEW dbo.AllRatings AS
	SELECT Vineyard.vineyard_name AS Variable, AVG(Wine.average_rating) AS Rating
	FROM Wine
	JOIN Varietal ON Varietal.varietal_ID = Wine.varietal_ID
	JOIN VineyardVarietal ON VineyardVarietal.varietal_ID = Varietal.varietal_ID
	JOIN Vineyard ON Vineyard.vineyard_ID = VineyardVarietal.vineyard_ID
	GROUP BY vineyard_name
	UNION
	-- Varietal rating table.
	SELECT Varietal.varietal_name AS Variable, AVG(Wine.average_rating) AS Rating
	FROM Wine
	JOIN Varietal ON Varietal.varietal_ID = Wine.varietal_ID
	GROUP BY varietal_name
	UNION
```

```
	-- Vintage rating table.
	SELECT CAST(Vintage.vintage_year AS char(4)) AS Variable,
AVG(Wine.average_rating) AS Rating
	FROM Wine
	JOIN Vintage ON Vintage.vintage_ID = Wine.vintage_ID
	GROUP BY vintage_year
	UNION
	-- Oak Type Rating Table
	SELECT Barrel.oak_type AS Variable, AVG(Wine.average_rating) AS Rating
	FROM Wine
	JOIN Barrel ON Barrel.barrel_ID = Wine.barrel_ID
	GROUP BY oak_type
go

SELECT * FROM dbo.AllRatings
ORDER BY Rating DESC
go

/*
The variables seem to be well distributed with no
clear indication of any one factor determing the
rating of a wine. Cabernet Sauvignon was the
highest rated grape varietal and the highest
```

rated variable overall with a rating close to 4.
2016 was the highest rated vintage, Violet Crown
was the highest rated vineyard, and American was
the highest rated oak type. The majority of the ratings
fall into the 2.5-3.5 range with two exceptions:
The aforementioned Cabernet Sauvignon and Riesling,
which had a measly rating of around 2.3.
*/

/*
Data Question 2: Which wines sell the best?

To answer this question, we will create a function called
"WineSalesFunction" that multiples the "price" and the
"quantity_sold" objects from the Wine table to get the
total sales of that wine. We will then select the top 3
wines based on their total sales. This select statement
will be created as a view in order to create a report in
access.
*/

```
Create Function dbo.WineSalesFunction (@wine_id int)
RETURNS decimal AS
```

```
BEGIN
    -- Declares a return value
    DECLARE @returnValue decimal

    -- Multiplies the quantity sold and the price to
    -- obtain the total sales
    SELECT @returnValue = quantity_sold * price FROM Wine
    WHERE Wine.wine_ID = @wine_id

    RETURN @returnValue
END
go

CREATE VIEW dbo.Top3Wines AS
        SELECT TOP 3
        wine_name AS WineName,
        price AS Price,
        quantity_sold AS QuantitySold,
        average_rating AS AverageRating,
        dbo.WineSalesFunction(wine_ID) AS Sales
        FROM Wine
        ORDER BY SALES DESC
go
```

```
SELECT * FROM dbo.Top3Wines
go

/*
Here we see that two of the Cabernets sold the best
followed by the latest vintage of the Chardonnay.
One thing to keep in mind is that the 2016 Cabernet
sold nearly 10,000 more units than the 2017 vintage,
but because the 2017 was four dollars more it sold
better overall.
*/

/*
Data Question 3: Do the ratings and the sales of a wine
correlate?

To answer this question, we will select the average rating
from the Wine table and cast the ratings as integers.

For example:
Wines with a rating of 1.00-1.99 will be floored to 1.
Wines with a rating of 2.00-2.99 will be floored to 2.
```

and so on...

This will effectively create the rating levels.

We will also select the sum of the WineSalesFunction

outputs from the Wine table and group by our predefined

levels.

*/

```
SELECT cast(average_rating as int) AS RatingLevel, SUM(dbo.WineSalesFunction(wine_ID))
AS TotalSales FROM Wine

GROUP BY cast(average_rating as int)

go
```

/*

There is a general correlation between the rating of a

wine and its sales. Wines rated as a 3 or a 4 sell

significantly better than those rated a 2 or a 1.

*/

/*

Data Question 4: Which wines sell best for each client

type?

To answer this question, we will create a view called

"ClientTypes" that displays the client_name,

the client_type, the wine_name,

and the quantity from the OrderForm table.

The results were ordered first by client_type

and then by quantity in a SELECT statement.

*/

```
CREATE VIEW dbo.ClientTypes AS
	SELECT client_name AS ClientName, client_type AS ClientType, wine_name WineName, quantity AS Quantity
	FROM OrderForm
	JOIN Clients ON Clients.client_ID = OrderForm.client_ID
	JOIN Wine ON Wine.wine_Id = OrderForm.wine_ID
go

SELECT * FROM dbo.ClientTypes
ORDER BY ClientType DESC, Quantity DESC
go
```

/*

What the report shows is that red wines are more popular

amongst customers while in retail/restaurants, whites

are slightly favores over reds.

*/

/*

Data Question 5: How do the total sales of a wine change

over time?

To answer this question, we will once again use the

"WineSalesFunction" to sum up the total sales of all the

wines for a given vintage.

*/

```
SELECT Vintage.vintage_year AS Vintage, SUM(dbo.WineSalesFunction(wine_ID)) AS YearlySales
FROM Wine
LEFT JOIN Vintage ON Vintage.vintage_ID = Wine.vintage_ID
GROUP BY Vintage.vintage_year
go
```

/*

We see a gradual increase in the sales from year to year.

One thing to note is that the increase from 2017 to 2018

was noticeably larger than the increase from 2016 to

2017. This may be a sign of the exponential growth of the

winery.

*/