Class_3_Notes

Data Prep

Load the following libraries and use read_csv to get the SalesOrderHeader.csv and Customer.csv data:

```
library(tidyverse)
library(stringr)
library(lubridate)

SalesOrderHeader = read_csv(str_c(locationString,"SalesOrderHeader.csv"))
Customer = read_csv(str_c(locationString,"Customer.csv"))
```

Now, let's clean up the data and eliminate the columns we won't use.

```
SalesOrderHeader = SalesOrderHeader %>% select(SalesOrderID, OrderDate, SalesOrderNumber, CustomerID, T

Customer = Customer %>% select(CustomerID, LastName, FirstName, CompanyName)
```

OK, we have a couple of variables that are going to cause problems: OrderDate and ShipSchedule. We'll need to convert these to Date datatypes.

The first one is easy. It's a string, and the format is complete enough to use string string=>date functions, like mdy. Remember the safe way to do this is to assign the function result to a temp column name, check the result, and then assign it back and drop the temp name.

```
SalesOrderHeader$OrderDate2 = mdy(SalesOrderHeader$OrderDate)

# Check results, then:

SalesOrderHeader = SalesOrderHeader %>%
  mutate(OrderDate = OrderDate2) %>%
  select(-OrderDate2)
```

The second one is a little more challenging *(and more typical of what you'll encounter in business - esp accounting)*.

First, there's no year. Since all the orders are in the same year, that's going to be easy - we'll just insert a string with the year.

Second, There's the weekday name, which we don't need *(this is not date data - it's a date format)* So, we'll drop that part.

The rest of the string looks good, So, we can:

1. Get the month name and day number *(pull that out using str_sub)*. str_sub takes 3 arguments: the string, where to begin and where to end. We don't really know where to begin because we have all those weekday names, but they're followed by a comma, which we can locate usibg str_locate, str_locate returns two values in a matrix *(the beginning and end of the string for each row)*. We just want the first column *(start of the string)* and then we'll add 1 to take out the comma.
2. Combine it with a year string *(use str_c to combine)*

Then we'll have complete date information in a string, so we can convert it using mdy.

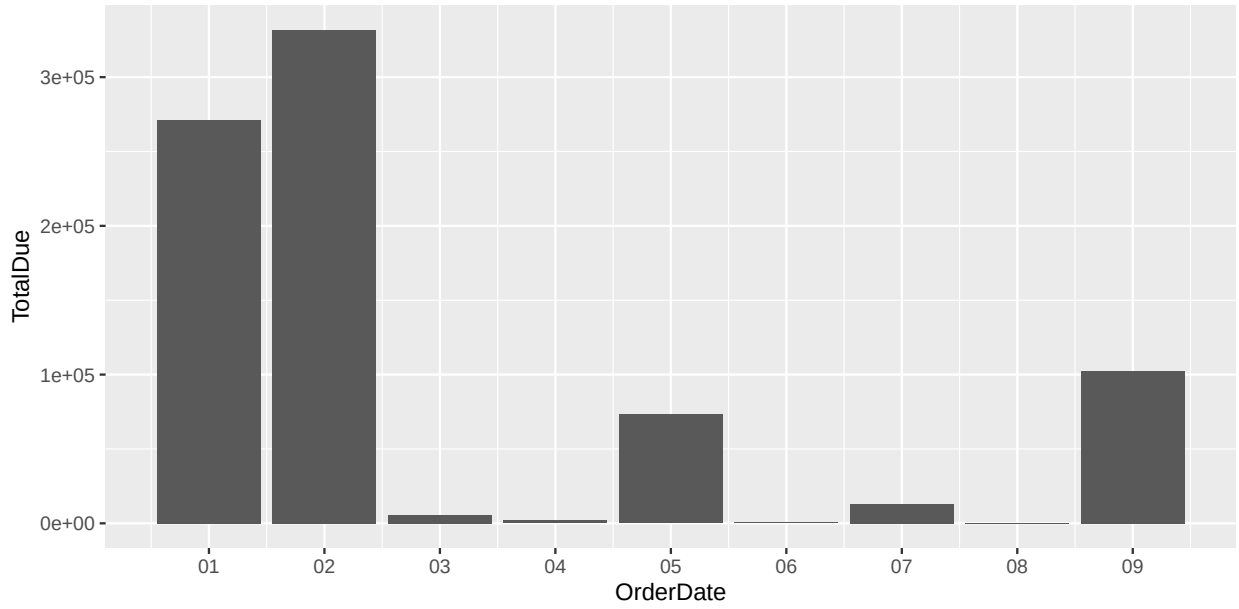
Then replace the column the safe way:

```
SalesOrderHeader = SalesOrderHeader%>%
  mutate(ShipSchedule2 =  mdy(
    str_c(str_sub(ShipSchedule, str_locate(ShipSchedule, ",")[,1]+1,
                              str_length(ShipSchedule)), ". 2008")))

SalesOrderHeader = SalesOrderHeader %>%
  mutate(ShipSchedule = ShipSchedule2) %>%
  select(-ShipSchedule2)
```

EDA (Exploratory Data Analysis)

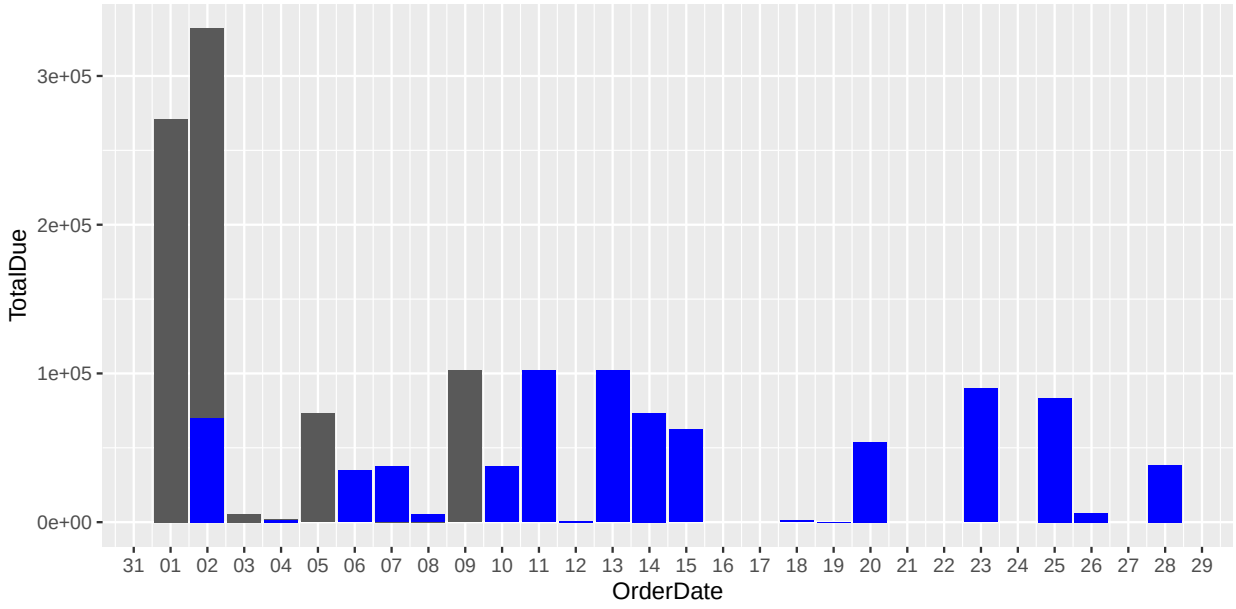
Now, let's take a quick look at the data and see if we can make sense of it:

```
p = ggplot(SalesOrderHeader, aes(OrderDate, TotalDue)) +
  geom_bar(stat = "identity") +
  scale_x_date(breaks = "1 day", date_labels = "%d")

p
```



OK, it appears that orders come in during the early part of the month. Good to know. One question we might ask is: "how long does it take to ship orders out?" Let's take a look *(adding another bar layer)*

```
p = p +
  geom_bar(aes(ShipSchedule, TotalDue), stat = "identity", fill = "blue")
p
```



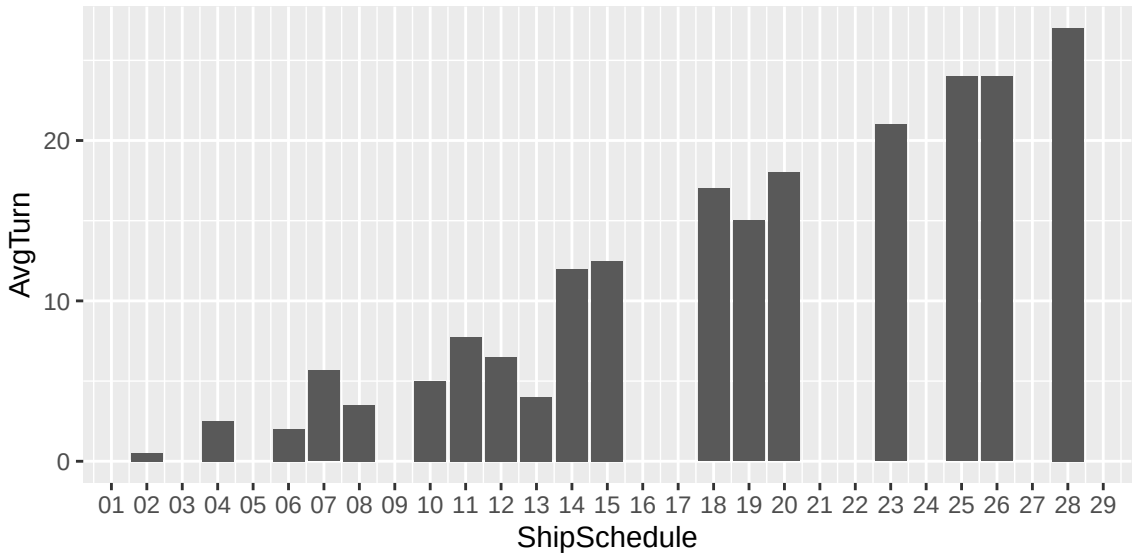
See how easy that is? The layer architecture of ggplot is so good!

This brings up the question: "How long did each order take to ship?" We can find that out with a little date arithmetic:

```
SalesOrderHeader = SalesOrderHeader %>%
  mutate(TurnOver = difftime(ShipSchedule, OrderDate, units = "days"))
```

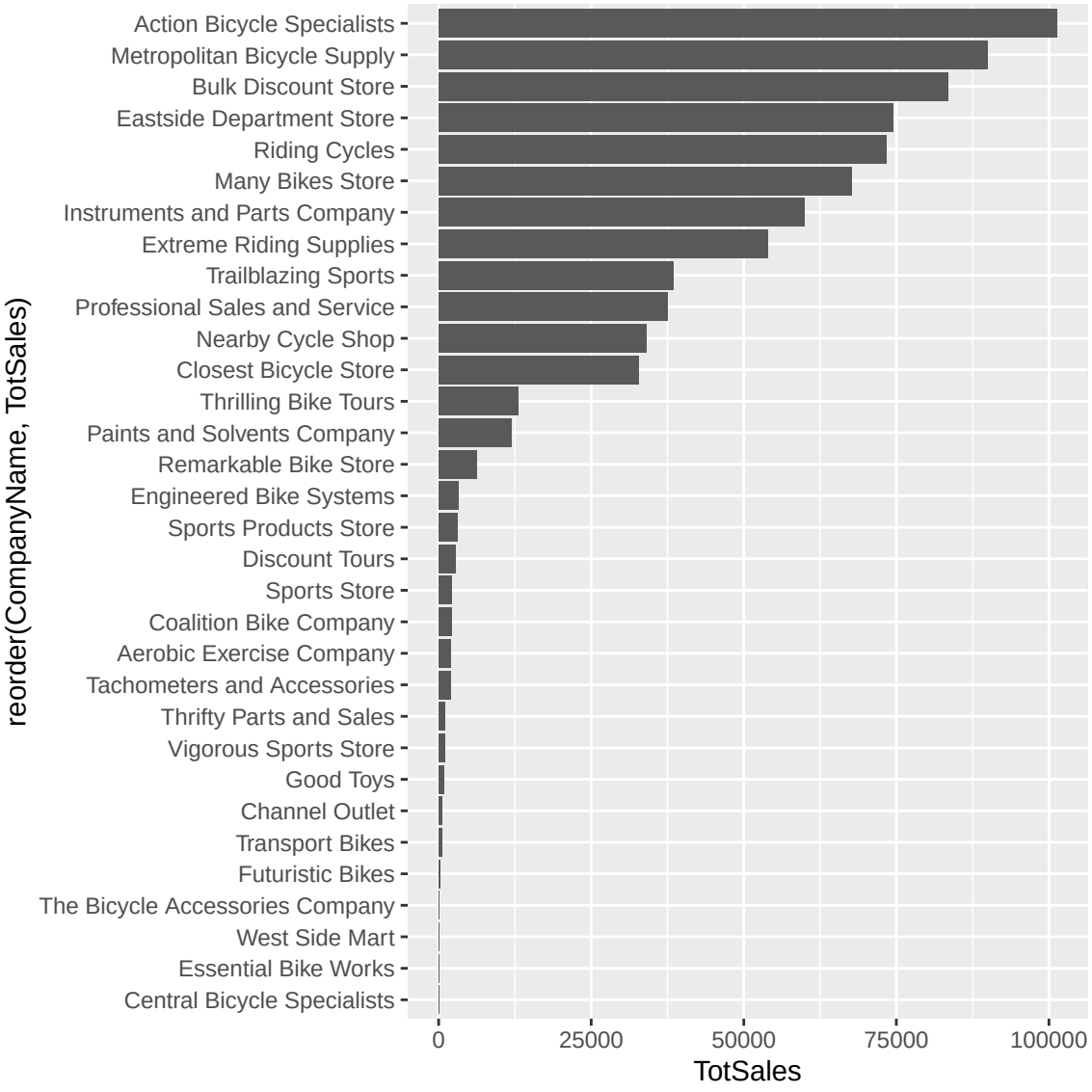
And this gives us the turnover for each order date. Let's take a look, but first, let's summarize by order date and get the average *(mean)* turnover.

```
TurnOver = SalesOrderHeader %>%
  group_by(ShipSchedule) %>%
  summarise(AvgTurn = mean(TurnOver))

ggplot(TurnOver, aes(ShipSchedule, AvgTurn)) +
  geom_bar(stat = "identity") +
  scale_x_date(breaks = "1 day", date_labels = "%d")
```



Finally, let's look at sales by Customer *(Company Name, not contact)*. We need to **Join** the Customer data to our SalesByCustomer data *(we'll spend a lot of time on joins starting next week - this is your first exposure)*. Here, we use a inner_join by CustomerID *(the primary key in the Customer dataset)*. Then, we'll group by Customer and summarize total sales *(reordering by sales so we can see the top customers)*:

```
SalesByCustomer = SalesOrderHeader %>%
  inner_join(Customer, by = "CustomerID") %>%
  group_by(CompanyName) %>%
  summarise(TotSales = sum(TotalDue))

ggplot(SalesByCustomer, aes(reorder(CompanyName, TotSales), TotSales)) +
  geom_bar(stat = "identity") +
  coord_flip()
```


Action Bicycle Specialists
Metropolitan Bicycle Supply
Bulk Discount Store
Eastside Department Store
Riding Cycles
Many Bikes Store
Instruments and Parts Company
Extreme Riding Supplies
Trailblazing Sports
Professional Sales and Service
Nearby Cycle Shop
Closest Bicycle Store
Thrilling Bike Tours
Paints and Solvents Company
Remarkable Bike Store
Engineered Bike Systems
Sports Products Store
Discount Tours
Sports Store
Coalition Bike Company
Aerobic Exercise Company
Tachometers and Accessories
Thrifty Parts and Sales
Vigorous Sports Store
Good Toys
Channel Outlet
Transport Bikes
Futuristic Bikes
The Bicycle Accessories Company
West Side Mart
Essential Bike Works
Central Bicycle Specialists
reorder(CompanyName, TotSales)
0
25000
50000
75000
100000
TotSales