BAUER COLLEGE OF BUSINESS UNIVERSITY of HOUSTON

Review Session

Just in general: be aware of back tics and column names

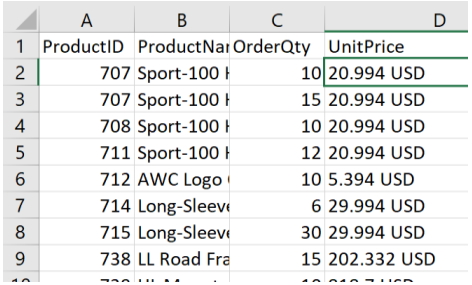
```
LondonSales <- rename(LondonSales, Product Name = ProductName)

# what's wrong here?

LondonSales <- rename(LondonSales, "Product Name" = ProductName)

# now what does the column look like?

LondonSales$`Product Name`

# what are those back tics? Why?

# put it back

LondonSales <- rename(LondonSales, ProductName = "Product Name")

# now what does the column look like?
```

back tics have more "power" than quotes - they can deal with special characters, but quotes will usually work
Tibbles store column names with spaces and special characters using back tics by default

strings

LondonSales$UnitPrice <- as.numeric(str_sub(LondonSales$UnitPrice, -str_length(LondonSales$UnitPrice), -5))

From r4ds 14.4.2 and stringr Cheatsheet: negative numbers count backwards from end example: str_sub(x, -3, -1)

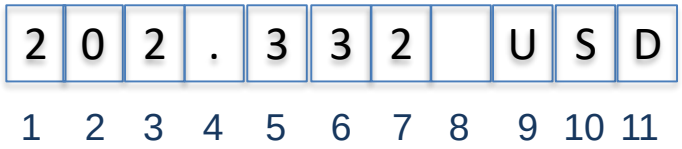


	A	B	C	D
1	ProductID	ProductNar	OrderQty	UnitPrice
2	707	Sport-100 H	10	20.994 USD
3	707	Sport-100 H	15	20.994 USD
4	708	Sport-100 H	10	20.994 USD
5	711	Sport-100 H	12	20.994 USD
6	712	AWC Logo (	10	5.394 USD
7	714	Long-Sleeve	6	29.994 USD
8	715	Long-Sleeve	30	29.994 USD
9	738	LL Road Fra	15	202.332 USD

If you look at this, you might think str_sub(UnitPrice, 1, 6) and that would work:

```
> str_sub(tstStr, 1, 6)
[1] "20.994"
```

But when you get to;




```
> str_sub(tstStr2, 1, 6)
[1] "202.33"
```

BAUER COLLEGE OF BUSINESS
UNIVERSITY of HOUSTON

2	0	.	9	9	9		U	S	D
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

So maybe you go with working from the end:

```
> str_sub(tstStr, -10, -5)
[1] "20.994"
```

But when you get to;

2	0	2	.	3	3	2		U	S	D
-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

```
> str_sub(tstStr2, -10, -5)
[1] "02.332"
```

So we need a reference point that moves with the number of characters in the string

```
> str_sub(tstStr, -str_length(tstStr2), -5)
[1] "20.994"
> str_sub(tstStr2, -str_length(tstStr2), -5)
[1] "202.332"
```

Another way is str_replace *(just be aware of trailing characters)*:

```
> str_replace(tstStr, "USD", "")
[1] "20.994 "
> str_replace(tstStr2, "USD", "")
[1] "202.332 "
```

Then, you just need to convert to a number:

```
> as.numeric(str_replace(tstStr, "USD", ""))
[1] 20.994
```

I said numerous times this semester: “make up your own spreadsheet and transform the data”, “spend time in R – hr / day”, If you did this, these kinds of problems are easy.

LondonSales$UnitPrice <- as.numeric(str_sub(LondonSales$UnitPrice, -str_length(LondonSales$UnitPrice), -5))

BAUER COLLEGE OF BUSINESS UNIVERSITY of HOUSTON

Strings and Numbers => Dates

If you get wacky Excel dates, you'll often need to pull a character string that has the month, the year, or month/day/year. You'll need to convert these to date datatypes.

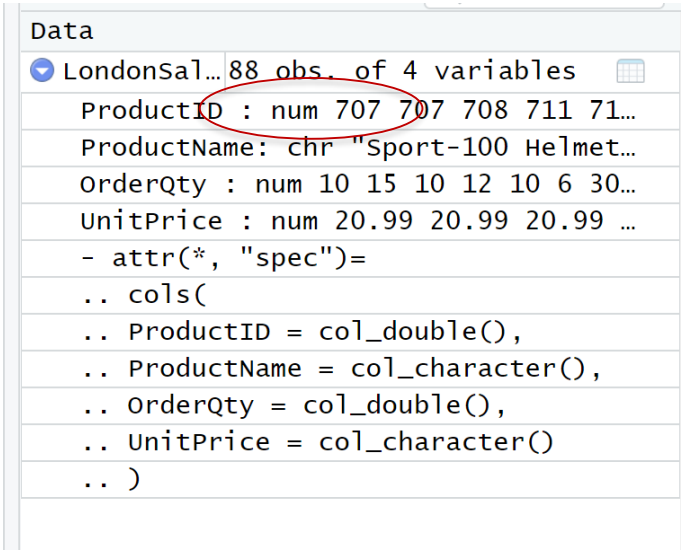
Strings => dates

```
> mdy("10/5/2019")
[1] "2019-10-05"
>
> datestring <- "Oct 19"
>
> mdy(datestring) # ??? not enough info
[1] NA
Warning message:
All formats failed to parse. No formats found.
>
> datestring2 <- "Oct 2019"
>
> mdy(datestring2) # the hueristic is guessing here - probably not what you want
[1] "2019-10-20"
>
> mdy(str_c(str_sub(datestring2, 1, 3), "01",  str_sub(datestring2, 5, 8)))
[1] "2019-10-01"
```

Numbers => dates

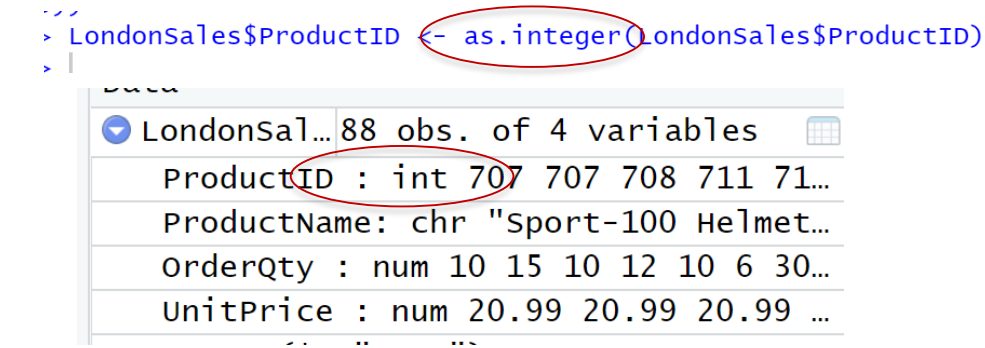
```
[1] "2019-10-01"
> make_date(2019, 10, 1)
[1] "2019-10-01"
```

Dplyr Transformations



Know your data – don't let it eat you up. Pay attention to data types

Is this right? Are Product ID's really continuous?



Then:

```
TopSales <- LondonSales %>% group_by(ProductID)%>%
  summarise(TotalSales = sum(OrderQty * UnitPrice )) %>%
  top_n(TotalSales, n = 10) %>%
  arrange(desc(TotalSales))
```

dplyr is **the** basic skill you need. You will use it constantly, and students that don't know it are students that haven't been working in R.

I see a lot of students confuse summarize with mutate. Mutate just adds a new column – you can compute and apply some functions, but you can't summarise all the records. That's summarise *(and BTW, I use generally use summarise instead summarize to avoid package conflicts)*

Getting the Data

```
ProductDescriptions <- dbGetQuery(con1,"
SELECT  [SalesLT].[Product].[ProductID]
                ,[SalesLT].[ProductModelProductDescription].[Culture]
                ,[SalesLT].[ProductDescription].[Description]
  FROM [SalesLT].[Product]

  INNER JOIN [SalesLT].[ProductModelProductDescription]
  ON [SalesLT].[Product].[ProductModelID] =
  [SalesLT].[ProductModelProductDescription].[ProductModelID]

  INNER JOIN [SalesLT].[ProductDescription]
  ON [SalesLT].[ProductModelProductDescription].[ProductDescriptionID] =
  [SalesLT].[ProductDescription].[ProductDescriptionID]
                ")
```

This is just a basic set of joins. Nothing difficult. But this whitespace should get your attention

```
ProductDe… 1764 obs. of 3 variables
  ProductID : int 994 995 996 984 98…
  Culture : chr "en " "en " "en " "e…
  Description: chr "Chromoly steel."…
TopSales   10 obs. of 2 variables
```

Because of this

```
ProductDescriptions <- filter(ProductDescriptions, Culture == "fr")
```

Returns 0 observations:

```
ProductDe… 0 obs. of 3 variables
```

Checking the data, we see that they're all 6 characters long

```
str_length(ProductDescriptions$Culture)
  [1] 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 [50] 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
 [99] 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
[148] 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
```

```
filter(ProductDescriptions, Culture == "fr    ")
```

Would work, but you're betting that ALL of them are exactly 6 characters – forever. Not a good bet

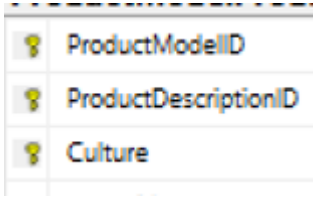
```
> ProductDescriptions$Culture <-  str_trim(ProductDescriptions$Culture, side = c("both"))
> ProductDescriptions <- filter(ProductDescriptions, Culture == "fr")
```

Is a much safer way to go

```
ProductDescriptions <- dbGetQuery(con1,"
SELECT  [SalesLT].[Product].[ProductID]
    ,[SalesLT].[Product].[ProductNumber]
                    ,[SalesLT].[ProductModelProductDescription].[Culture]
                    ,[SalesLT].[ProductDescription].[Description]
  FROM [SalesLT].[Product]
  INNER JOIN [SalesLT].[ProductModelProductDescription]
  ON [SalesLT].[Product].[ProductModelID] =
  [SalesLT].[ProductModelProductDescription].[ProductModelID]
  INNER JOIN [SalesLT].[ProductDescription]
  ON [SalesLT].[ProductModelProductDescription].[ProductDescriptionID] =
  [SalesLT].[ProductDescription].[ProductDescriptionID]
  WHERE TRIM([SalesLT].[ProductModelProductDescription].[Culture]) = 'fr'
                    ")
```

Another point: you can be confident that the lengths match because:

SQL Server does allow variable characters as keys, but in general, it's not good practice (some exceptions).



```
SalesAnalysis <- TopSales %>% inner_join(ProductDescriptions, by =
"ProductID")

SalesAnalysis$TotalSales <- round(SalesAnalysis$TotalSales * .9,0)
```

Using an inner_join here is a little dangerous, but I already know the products I'm after (top 10) exist in both places

Read the instructions:

"He wants you to pull the top 10 product sales in London in **July"**

"Your Fellow Financial analyst, Felicity Flatulation, has a spreadsheet handy with London sales in **July"**

Some students pulled the sales data from the database. Did you need that? What month is that data?

Other things to “bone up” on

Joins

```
CitySales <- dbGetQuery(con1,"

SELECT

  [SalesLT].[Address].[City]
 ,sum([SalesLT].[SalesOrderHeader].[SubTotal]) AS Total

FROM [SalesLT].[Address]

  INNER JOIN [SalesLT].[CustomerAddress]
  ON [SalesLT].[Address].[AddressID] =
  [SalesLT].[CustomerAddress].[AddressID]

  INNER JOIN [SalesLT].[SalesOrderHeader]
  ON [SalesLT].[CustomerAddress].[CustomerID] =
  [SalesLT].[SalesOrderHeader].[CustomerID]

GROUP BY

  [SalesLT].[Address].[City]

")
```

```
CitySales  29 obs. of 2 variables
  City : chr "Abingdon" "Alhambra" "…
  Total: num 37.8 96.1 713.8 1732.9 …
```

	City	Total
1	Abingdon	37.7580
2	Alhambra	96.1088
3	Auburn	713.7960
4	Camarillo	1732.8900
5	Cambridge	1884.3948
6	Cerritos	34118.5356
7	Culver City	221.2560
8	Daly City	2527.1280
9	El Segundo	1856.2068
10	Englewood	10585.0500
11	Fullerton	59894.2092
12	Gloucestershire	53248.6920

```
> sum(CitySales$Total)
[1] 708690.2
```

```
CitySales2 <- dbGetQuery(con1,"

SELECT

  [SalesLT].[Address].[City]
 ,sum([SalesLT].[SalesOrderHeader].[SubTotal]) AS Total

FROM [SalesLT].[Address]

  INNER JOIN [SalesLT].[CustomerAddress]
  ON [SalesLT].[Address].[AddressID] =
  [SalesLT].[CustomerAddress].[AddressID]

  LEFT JOIN [SalesLT].[SalesOrderHeader]
  ON [SalesLT].[CustomerAddress].[CustomerID] =
  [SalesLT].[SalesOrderHeader].[CustomerID]

GROUP BY

  [SalesLT].[Address].[City]

")
```

```
CitySale… 256 obs. of 2 variables
  City : chr "Abingdon" "Albany" "Al…
  Total: num 37.8 NA 96.1 NA 713.8 .…
```

	City	Total
1	Abingdon	37.7580
2	Albany	*NA*
3	Alhambra	96.1088
4	Arlington	*NA*
5	Auburn	713.7960
6	Aurora	*NA*
7	Austin	*NA*
8	Baldwin Park	*NA*
9	Barrie	*NA*
10	Barstow	*NA*
11	Basingstoke Hants	*NA*
12	Baytown	*NA*

```
> sum(CitySales2$Total, na.rm = T)
[1] 708690.2
```

```
CitySales3 <- dbGetQuery(con1,"

SELECT

  [SalesLT].[Address].[City]
 ,sum([SalesLT].[SalesOrderHeader].[SubTotal]) AS Total

FROM [SalesLT].[Address]

  LEFT JOIN [SalesLT].[CustomerAddress]
  ON [SalesLT].[Address].[AddressID] =
  [SalesLT].[CustomerAddress].[AddressID]

  LEFT JOIN [SalesLT].[SalesOrderHeader]
  ON [SalesLT].[CustomerAddress].[CustomerID] =
  [SalesLT].[SalesOrderHeader].[CustomerID]

GROUP BY

  [SalesLT].[Address].[City]

")
```

```
CitySale… 269 obs. of 2 variables
  City : chr "Abingdon" "Albany" "Al…
  Total: num 37.8 NA 96.1 NA NA ...
```

	City	Total
1	Abingdon	37.7580
2	Albany	*NA*
3	Alhambra	96.1088
4	Alpine	*NA*
5	Arlington	*NA*
6	Auburn	713.7960
7	Aurora	*NA*
8	Austin	*NA*
9	Baldwin Park	*NA*
10	Barrie	*NA*
11	Barstow	*NA*
12	Basingstoke Hants	*NA*

```
> sum(CitySales3$Total, na.rm = T)
[1] 708690.2
```

```
SELECT [AddressID]
      ,[AddressLine1]
      ,[AddressLine2]
      ,[City]
  FROM [SalesLT].[Address]
  WHERE [SalesLT].[Address].[City] = 'Alpine'

SELECT [CustomerID]
      ,[AddressID]
      ,[AddressType]
  FROM [SalesLT].[CustomerAddress]
  WHERE [AddressID] = 1063
```

	AddressID	AddressLine1	AddressLine2	City
1	1063	Viejas Outlet Center	NULL	Alpine

CustomerID	AddressID	AddressType

So, just looking here, there is an entry for Alpine in Address, but none in CustomerAddress

Think through it – use the Venn diagram in the book.

BAUER COLLEGE OF BUSINESS UNIVERSITY of HOUSTON

Other stuff

Tibble$NewColumn <- Operation(other columns)

```
TopSales$NewSales <- TopSales$TotalSales * 2
```

What would happen if it was a column in another tibble? Does the length matter?

Dealing with NAs and filtering logic *(play around all this until you understand)*

```
filter(TopSales, TotalSales > 0 )
# this will exclude NAs but what happens if you need neg numbers....
filter(TopSales, !TotalSales > 0 )
filter(TopSales, is.na(TotalSales))
filter(TopSales, !is.na(TotalSales) | !is.na(NewSales))
filter(TopSales, !is.na(TotalSales) & !is.na(NewSales))

filter(TopSales, NewSales > 0 )
filter(TopSales, TotalSales > 0  & NewSales > 0 )
filter(TopSales, TotalSales > 0  | NewSales > 0 )
filter(TopSales, is.na(TotalSales) | is.na(NewSales))
filter(TopSales, is.na(TotalSales) & is.na(NewSales))
sum(TopSales$TotalSales)
sum(TopSales$TotalSales, na.rm = T)
```

NA's are common, esp when you're bringing back data from outerjoins and both finals have outerjoins

Pivots: Pivot Wider

CustomerSales = CustomerSales %>% arrange(CompanyName)
SalesByDate = CustomerSales %>% pivot_wider(names_from = CompanyName, values_from = SubTotal)

	OrderDate	SubTotal	CompanyName
1	2008-06-09	89869.2763	Action Bicycle Specialists
2	2008-06-01	1732.8900	Aerobic Exercise Company
3	2008-06-01	74160.2280	Bulk Discount Store
4	2008-06-01	31.5840	Central Bicycle Specialists
5	2008-06-02	524.6640	Channel Outlet
6	2008-06-05	28950.6781	Closest Bicycle Store
7	2008-06-02	1856.2068	Coalition Bike Company
8	2008-06-03	2527.1280	Discount Tours
9	2008-06-01	65683.3680	Eastside Department Store
10	2008-06-05	2847.4080	Engineered Bike Systems
11	2008-06-01	37.7580	Essential Bike Works
12	2008-06-02	47848.0260	Extreme Riding Supplies

	OrderDate	Action Bicycle Specialists	Aerobic Exercise Company	Bulk Discount Store	Central Bicycle Specialists	Channel Outlet	Clos Bicy Stor
1	2008-06-09	89869.28	NA	NA	NA	NA	NA
2	2008-06-01	NA	1732.89	74160.23	31.584	NA	NA
3	2008-06-02	NA	NA	NA	NA	524.664	NA
4	2008-06-05	NA	NA	NA	NA	NA	2895
5	2008-06-03	NA	NA	NA	NA	NA	NA
6	2008-06-04	NA	NA	NA	NA	NA	NA
7	2008-06-07	NA	NA	NA	NA	NA	NA
8	2008-06-06	NA	NA	NA	NA	NA	NA
9	2008-06-08	NA	NA	NA	NA	NA	NA

	OrderDate	SubTotal	CompanyName
2	2008-06-05T00:00:00Z	33319.986	Professional Sales and Service
3	2008-06-02T00:00:00Z	5533.8689	Remarkable Bike Store
4	2008-06-01T00:00:00Z	74160.228	Bulk Discount Store
5	2008-06-02T00:00:00Z	1856.2068	Coalition Bike Company
6	2008-06-03T00:00:00Z	221.256	Futuristic Bikes
7	2008-06-02T00:00:00Z	524.664	Channel Outlet
8	2008-06-01T00:00:00Z	1732.89	Aerobic Exercise Company
9	2008-06-01T00:00:00Z	858.9	Vigorous Sports Store
10	2008-06-07T00:00:00Z	11528.844	Thrilling Bike Tours
11	2008-06-02T00:00:00Z	47848.026	Extreme Riding Supplies
12	2008-06-09T00:00:00Z	89869.276	Action Bicycle Specialists
13	2008-06-01T00:00:00Z	31.584	Central Bicycle Specialists
14	2008-06-04T00:00:00Z	96.1088	The Bicycle Accessories Compan

Sum of SubTotal	Column Labels					
Row Labels	Action Bicycle Specialists	Aerobic Exercise Company	Bulk Discount Store	Central Bicycle Specialists	Channel Outlet	Closest Bicycle Sto
2008-06-01T00:00:00Z		1732.89	74160.228	31.584		
2008-06-02T00:00:00Z					524.664	
2008-06-03T00:00:00Z						
2008-06-04T00:00:00Z						
2008-06-05T00:00:00Z						28950.67
2008-06-06T00:00:00Z						
2008-06-07T00:00:00Z						
2008-06-08T00:00:00Z						
2008-06-09T00:00:00Z	89869.2763					
Grand Total	89869.2763	1732.89	74160.228	31.584	524.664	28950.67

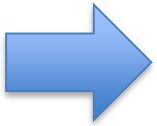
Pivot Longer

```
CustomerSales2 = SalesByDate %>% pivot_longer(
  2:ncol(SalesByDate),
  names_to = "CompanyName",
  values_to = "SubTotal",
  values_drop_na = T )
```

	OrderDate	Action Bicycle Specialists	Aerobic Exercise Company	Bulk Discount Store	Central Bicycle Specialists	Channel Outlet	Clos Bicy Stor
1	2008-06-09	89869.28	NA	NA	NA	NA	NA
2	2008-06-01	NA	1732.89	74160.23	31.584	NA	NA
3	2008-06-02	NA	NA	NA	NA	524.664	NA
4	2008-06-05	NA	NA	NA	NA	NA	289
5	2008-06-03	NA	NA	NA	NA	NA	NA
6	2008-06-04	NA	NA	NA	NA	NA	NA
7	2008-06-07	NA	NA	NA	NA	NA	NA
8	2008-06-06	NA	NA	NA	NA	NA	NA
9	2008-06-08	NA	NA	NA	NA	NA	NA

	OrderDate	CompanyName	SubTotal
1	2008-06-09	Action Bicycle Specialists	89869.2763
2	2008-06-09	Thrifty Parts and Sales	926.9160
3	2008-06-01	Aerobic Exercise Company	1732.8900
4	2008-06-01	Bulk Discount Store	74160.2280
5	2008-06-01	Central Bicycle Specialists	31.5840
6	2008-06-01	Eastside Department Store	65683.3680
7	2008-06-01	Essential Bike Works	37.7580
8	2008-06-01	Good Toys	713.7960
9	2008-06-01	Many Bikes Store	59894.2092
10	2008-06-01	Sports Products Store	2777.1431
11	2008-06-01	Trailblazing Sports	34118.5356

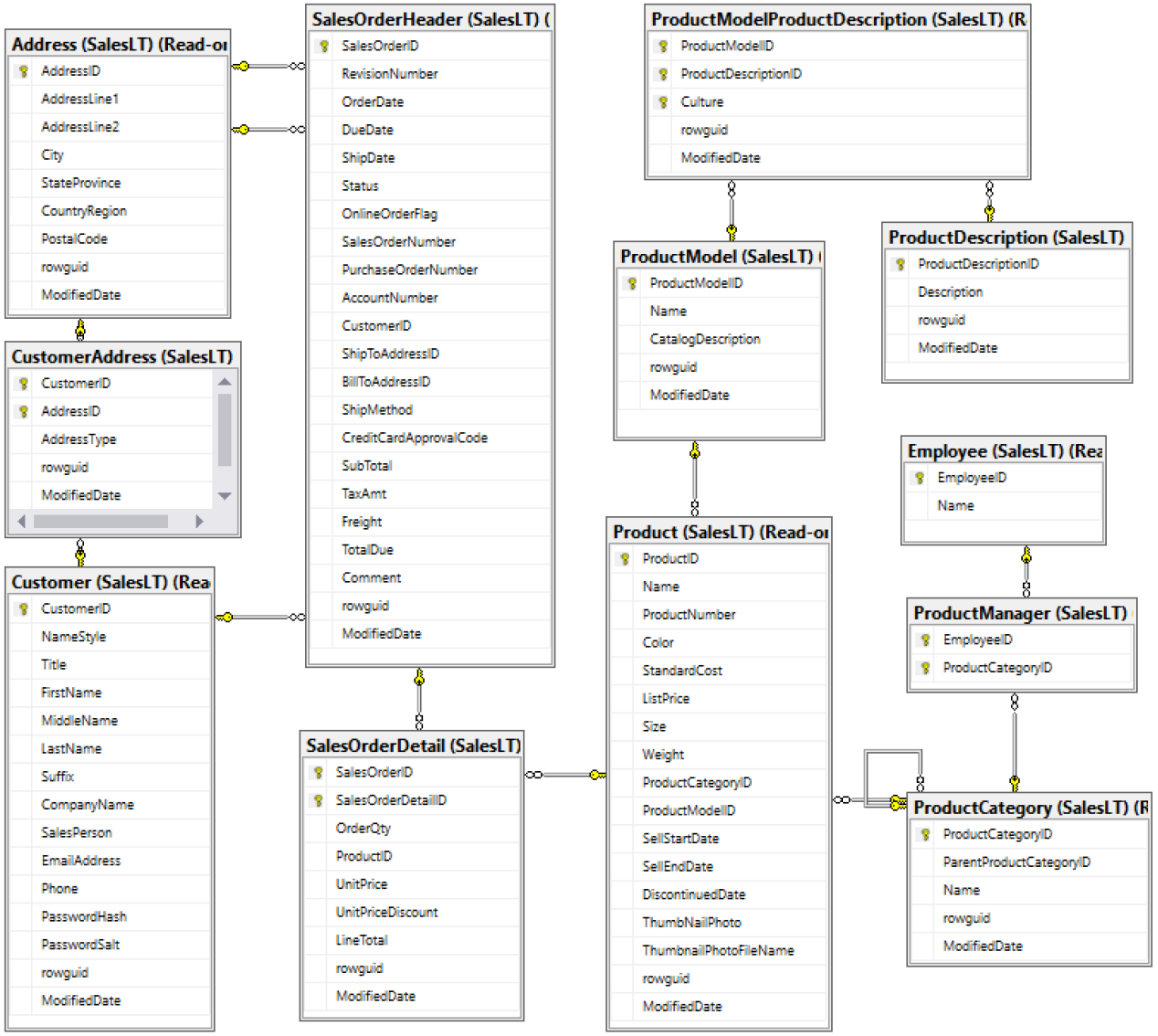
OrderDate	Action Bicy	Aerobic Exe	Bulk Discou	Central Bic	Channel Ou	Closest Bicy	Coaliti
2008-06-09	89869.28	NA	NA	NA	NA	NA	NA
2008-06-01	NA	1732.89	74160.23	31.584	NA	NA	NA
2008-06-02	NA	NA	NA	NA	524.664	NA	1856
2008-06-05	NA	NA	NA	NA	NA	28950.68	NA
2008-06-03	NA	NA	NA	NA	NA	NA	NA
2008-06-04	NA	NA	NA	NA	NA	NA	NA
2008-06-07	NA	NA	NA	NA	NA	NA	NA
2008-06-06	NA	NA	NA	NA	NA	NA	NA
2008-06-08	NA	NA	NA	NA	NA	NA	NA



A bunch of work in PowerPivot

BAUER COLLEGE OF BUSINESS
UNIVERSITY of HOUSTON

Adventure Works Data Model



Associative Entity