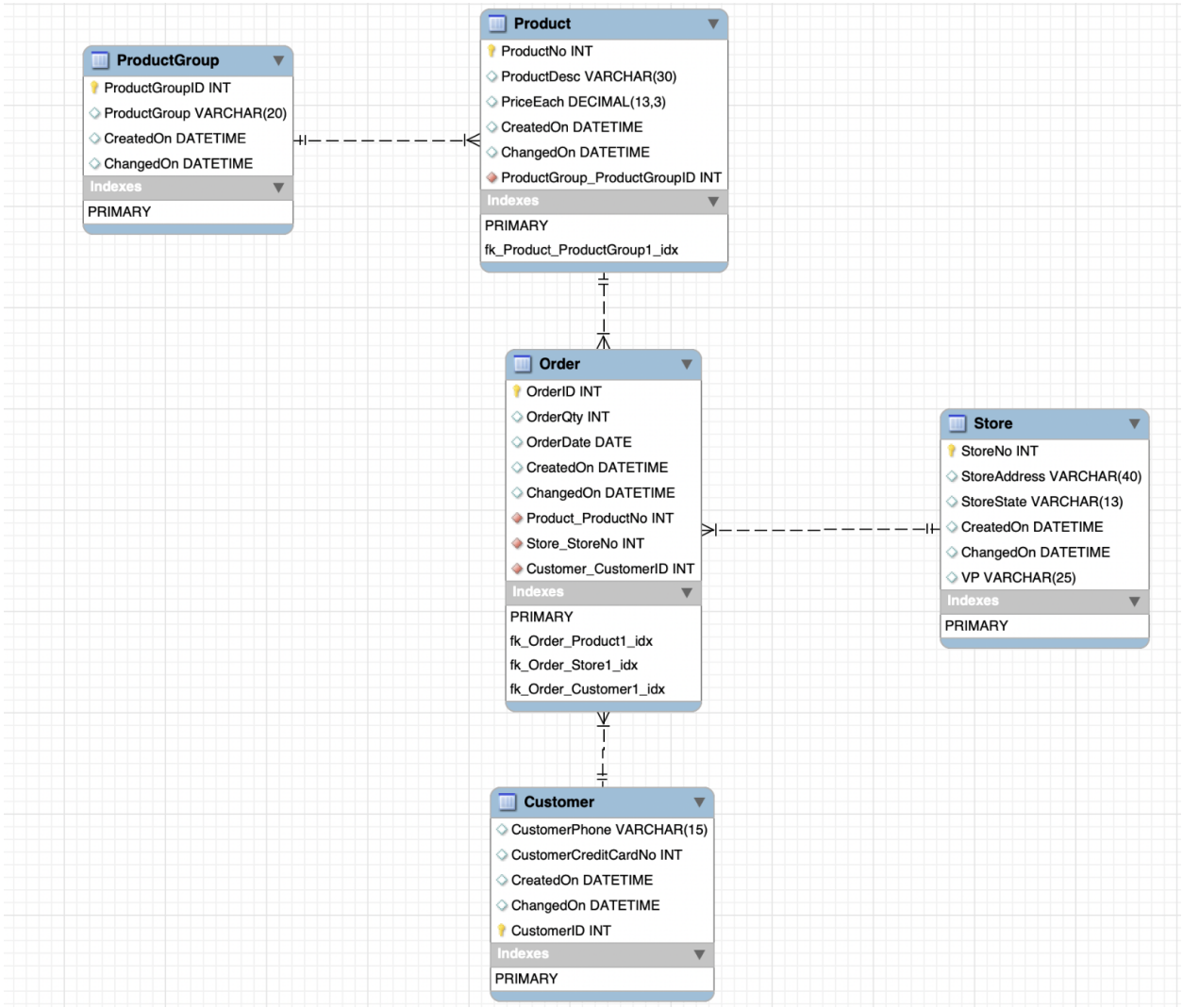
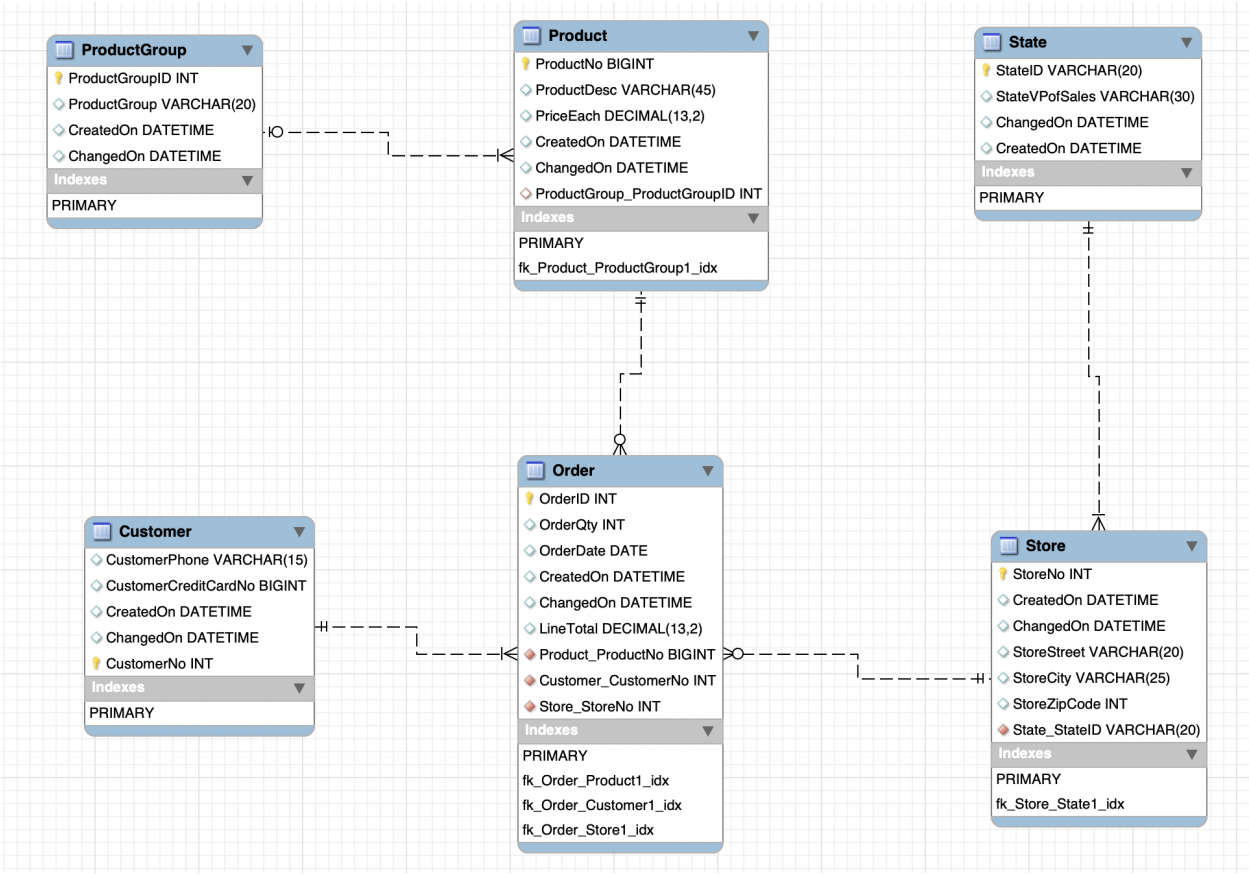
Group 7
Semester Project Part 2
Sales Area 2: Massachusetts and Maine

1.

Original EER model:



Updated EER model:



In terms of changes that were made, a State table was created in order to address a transitive property. The StateVPofSales should not have been an attribute in the Store table, but rather an attribute in a new State table. The State table primary key should have been a foreign key in the Store table because the VP depended on the State itself, not the Store. When it comes to the Order table, LineTotal was left out as an attribute in the original model, so it needed to be added to the updated model. Some of the cardinalities were incorrect in the original EER model as well. A product does not necessarily have to belong to a product group, so an optional cardinality for Product Group was included, as well as two more between the Order and Store relationship and the Order and Product relationship considering a Product does not necessarily have to be attached to an Order and a Store does not necessarily have to place an Order. Additionally, the Store table originally had the attribute of StoreAddress, but this was actually a multi-valued attribute that needed to be separated into Street, City, and Zip Code. State was already accounted for in the State table. Finally, some of the metadata needed to be updated, for both ProductNo and CustomerCreditCardNo required BIGINT rather than just INT.

2.

Describing the Product Group table:

```
12  DESCRIBE group7_fortis.ProductGroup;
13
```

100% 37:12

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
ProductGroupID	int	NO	PRI	NULL	auto_increment
ProductGroupDesc	varchar(25)	YES		NULL	
CreatedOn	datetime	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED
ChangedOn	datetime	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

Describing the Customer table:

```
31  DESCRIBE group7_fortis.Customer;
32
```

100% 29:31

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
CustomerNo	int	NO	PRI	NULL	auto_increment
CustomerCreditCardNo	bigint	YES		NULL	
CustomerPhone	varchar(15)	YES		NULL	
CreatedOn	datetime	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED
ChangedOn	datetime	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

Describing the State table:

```
49  DESCRIBE group7_fortis.State;
50
```

100% 23:49

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
StateID	varchar(20)	NO	PRI	NULL	
StateVPofSales	varchar(45)	YES		NULL	
CreatedOn	datetime	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED
ChangedOn	datetime	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

Describing the Store table:

```
71  DESCRIBE group7_fortis.Store;
72
```

100% 21:71

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
StoreNo	int	NO	PRI	NULL	
StateID	varchar(20)	YES	MUL	NULL	
StoreStreet	varchar(35)	YES		NULL	
StoreCity	varchar(20)	YES		NULL	
StoreZip	int	YES		NULL	
CreatedOn	datetime	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED
ChangedOn	datetime	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

Describing the Product table:

```
92  describe Group7_Fortis.Product;
93
```

100% 27:92

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
ProductNo	bigint	NO	PRI	NULL	
ProductGroupID	int	YES	MUL	NULL	
ProductDesc	varchar(45)	YES		NULL	
PriceEach	decimal(13,2)	YES		NULL	
CreatedOn	datetime	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED
ChangedOn	datetime	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

Describing the Order table:

```
118  DESCRIBE Group7_Fortis.Order;
119
```

100% 25:118

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
OrderID	int	NO	PRI	NULL	
ProductNo	bigint	NO	MUL	NULL	
CustomerNo	int	NO	MUL	NULL	
StoreNo	int	NO	MUL	NULL	
OrderDate	date	YES		NULL	
OrderQty	int	YES		NULL	
LineTotal	decimal(13,2)	YES		NULL	
CreatedOn	datetime	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED
ChangedOn	datetime	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

3.

Displaying the data in the Product Group table:

```
18 •  SELECT * FROM group7_fortis.ProductGroup;
19
```

100% 21:18

Result Grid | Filter Rows: Search | Edit:

ProductGroupID	ProductGroupDe...	CreatedOn	ChangedOn
1	Display Unit	2023-04-14 22:24:36	2023-04-14 22:24:36
2	Headphone	2023-04-14 22:24:36	2023-04-14 22:24:36
3	Cellphone	2023-04-14 22:24:36	2023-04-14 22:24:36
4	Utensils	2023-04-14 22:24:36	2023-04-14 22:24:36
5	Accessory	2023-04-14 22:24:36	2023-04-14 22:24:36
6	Laptop	2023-04-14 22:24:36	2023-04-14 22:24:36
NULL	NULL	NULL	NULL

Displaying the data in the Customer table:

```
37 •  SELECT * FROM group7_fortis.Customer;
38
```

100% 25:37

Result Grid | Filter Rows: Search | Edit: | Export/Import

CustomerNo	CustomerCreditCard...	CustomerPhone	CreatedOn	ChangedOn
8789	9953683385183360	(840) 142-1925	2023-04-14 22:26:17	2023-04-14 22:26:17
8790	9954729404787400	(855) 300-7261	2023-04-14 22:26:17	2023-04-14 22:26:17
8791	9954855285720770	(985) 302-5900	2023-04-14 22:26:17	2023-04-14 22:26:17
8792	9956938102011040	(505) 574-0417	2023-04-14 22:26:17	2023-04-14 22:26:17
8793	9958168052499420	(968) 585-1636	2023-04-14 22:26:17	2023-04-14 22:26:17
8794	9958502831289710	(929) 305-2506	2023-04-14 22:26:17	2023-04-14 22:26:17
8795	9959677951988370	(779) 665-4831	2023-04-14 22:26:17	2023-04-14 22:26:17
8796	9961219442002340	(620) 261-0556	2023-04-14 22:26:17	2023-04-14 22:26:17
8797	9962622668374800	(999) 710-2121	2023-04-14 22:26:17	2023-04-14 22:26:17
8798	9968821506857320	(932) 409-7989	2023-04-14 22:26:17	2023-04-14 22:26:17
8799	9970202658597420	(517) 177-7155	2023-04-14 22:26:17	2023-04-14 22:26:17
8800	9973805436059330	(849) 321-7271	2023-04-14 22:26:17	2023-04-14 22:26:17
8801	9973858888899800	(744) 965-7726	2023-04-14 22:26:17	2023-04-14 22:26:17
8802	9980746454330680	(668) 546-0362	2023-04-14 22:26:17	2023-04-14 22:26:17
8803	9982465348024020	(953) 838-9163	2023-04-14 22:26:17	2023-04-14 22:26:17
8804	9982969287066500	(809) 843-4410	2023-04-14 22:26:17	2023-04-14 22:26:17
8805	9985781052468310	(920) 879-2181	2023-04-14 22:26:17	2023-04-14 22:26:17
8806	9986287450750360	(635) 519-9375	2023-04-14 22:26:17	2023-04-14 22:26:17
8807	9997313591837240	(676) 510-1371	2023-04-14 22:26:17	2023-04-14 22:26:17
8808	9999544765884230	(766) 878-4730	2023-04-14 22:26:17	2023-04-14 22:26:17
NULL	NULL	NULL	NULL	NULL

Displaying the data in the State table:

```
55  SELECT * FROM group7_fortis.State;
56
```

100% 22:55

Result Grid | Filter Rows: Search | Edit:

StateID	StateVPofSales	CreatedOn	ChangedOn
Maine	Astrid Copeland	2023-04-14 22:29:39	2023-04-14 22:29:39
Massachusetts	Bessie Maldonado	2023-04-14 22:29:39	2023-04-14 22:29:39
NULL	NULL	NULL	NULL

Displaying the data in the Store table:

```
77  SELECT * FROM group7_fortis.Store;
78
```

100% 24:77

Result Grid | Filter Rows: Search | Edit: | Export/Import:

StoreNo	StateID	StoreStreet	StoreCity	StoreZip	CreatedOn	ChangedOn
93526	Massachusetts	996 Sunset St	Boston	2215	2023-04-14 22:32:01	2023-04-14 22:32:01
93544	Massachusetts	997 Adams St	Boston	2215	2023-04-14 22:31:57	2023-04-14 22:31:57
93553	Massachusetts	997 Elm St	Boston	2215	2023-04-14 22:32:09	2023-04-14 22:32:09
93554	Massachusetts	997 Forest St	Boston	2215	2023-04-14 22:32:05	2023-04-14 22:32:05
93555	Maine	997 Forest St	Portland	4101	2023-04-14 22:31:47	2023-04-14 22:31:47
93564	Massachusetts	997 River St	Boston	2215	2023-04-14 22:31:52	2023-04-14 22:31:52
93565	Massachusetts	997 South St	Boston	2215	2023-04-14 22:32:10	2023-04-14 22:32:10
93569	Massachusetts	998 11th St	Boston	2215	2023-04-14 22:32:02	2023-04-14 22:32:02
93574	Maine	998 4th St	Portland	4101	2023-04-14 22:31:46	2023-04-14 22:31:46
93579	Massachusetts	998 9th St	Boston	2215	2023-04-14 22:32:02	2023-04-14 22:32:02
93587	Massachusetts	998 Hill St	Boston	2215	2023-04-14 22:32:01	2023-04-14 22:32:01
93589	Massachusetts	998 Jefferson St	Boston	2215	2023-04-14 22:31:57	2023-04-14 22:31:57
93593	Massachusetts	998 Madison St	Boston	2215	2023-04-14 22:32:06	2023-04-14 22:32:06
93597	Massachusetts	998 Pine St	Boston	2215	2023-04-14 22:31:52	2023-04-14 22:31:52
93609	Massachusetts	999 10th St	Boston	2215	2023-04-14 22:32:01	2023-04-14 22:32:01
93619	Massachusetts	999 8th St	Boston	2215	2023-04-14 22:32:03	2023-04-14 22:32:03
93620	Massachusetts	999 9th St	Boston	2215	2023-04-14 22:32:04	2023-04-14 22:32:04
93626	Massachusetts	999 Dogwood St	Boston	2215	2023-04-14 22:31:55	2023-04-14 22:31:55
93628	Massachusetts	999 Lakeview St	Boston	2215	2023-04-14 22:31:50	2023-04-14 22:31:50
93634	Massachusetts	999 South St	Boston	2215	2023-04-14 22:31:47	2023-04-14 22:31:47
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Displaying the data in the Product table:

```
98 •  SELECT * FROM group7_fortis.Product;
99
```

100% 24:98

Result Grid | Filter Rows: Search | Edit: | Export/Import:

ProductNo	ProductGroupID	ProductDesc	PriceEach	CreatedOn	ChangedOn
794567898946	1	34in Ultrawide Monitor	379.99	2023-04-14 22:56:27	2023-04-14 22:56:27
794567899254	NULL	Vareebadd Phone	400.00	2023-04-14 22:56:27	2023-04-14 22:56:27
794567899750	1	20in Monitor	109.99	2023-04-14 22:56:27	2023-04-14 22:56:27
794567899779	2	Apple Airpods Headphones	150.00	2023-04-14 22:56:27	2023-04-14 22:56:27
794567901549	NULL	AAA Batteries (4-pack)	2.99	2023-04-14 22:56:27	2023-04-14 22:56:27
794567902286	5	Lightning Charging Cable	14.95	2023-04-14 22:56:27	2023-04-14 22:56:27
794567903176	6	Macbook Pro Laptop	1700.00	2023-04-14 22:56:27	2023-04-14 22:56:27
794567903342	4	LG Washing Machine	600.00	2023-04-14 22:56:27	2023-04-14 22:56:27
794567904233	3	Google Phone	600.00	2023-04-14 22:56:27	2023-04-14 22:56:27
794567905173	1	Flatscreen TV	300.00	2023-04-14 22:56:27	2023-04-14 22:56:27
794567905206	6	ThinkPad Laptop	999.99	2023-04-14 22:56:27	2023-04-14 22:56:27
794567905239	1	27in 4K Gaming Monitor	389.99	2023-04-14 22:56:27	2023-04-14 22:56:27
794567906339	2	Bose SoundSport Headph…	99.99	2023-04-14 22:56:27	2023-04-14 22:56:27
794567906345	3	iPhone	700.00	2023-04-14 22:56:27	2023-04-14 22:56:27
794567906613	4	LG Dryer	600.00	2023-04-14 22:56:27	2023-04-14 22:56:27
794567907519	2	Wired Headphones	11.99	2023-04-14 22:56:27	2023-04-14 22:56:27
794567908249	5	USB-C Charging Cable	11.95	2023-04-14 22:56:27	2023-04-14 22:56:27
794567908346	1	27in FHD Monitor	149.99	2023-04-14 22:56:27	2023-04-14 22:56:27
NULL	NULL	NULL	NULL	NULL	NULL

Displaying the data in the Order table:

```
124 •  SELECT * FROM group7_fortis.Order;
125
```

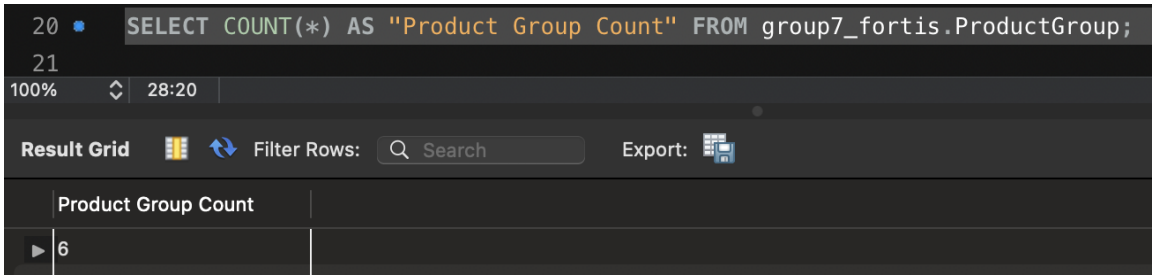
100% 24:124

Result Grid | Filter Rows: Search | Edit: | Export/Import:

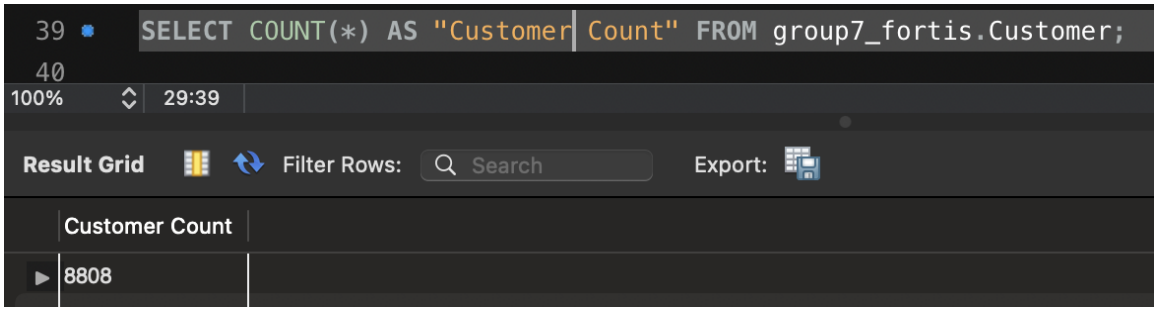
OrderID	ProductNo	CustomerNo	StoreNo	OrderDate	OrderQty	LineTotal	CreatedOn	ChangedOn
176507	794567903176	8806	69489	2019-03-06	1	1700.00	2023-04-14 23:00:34	2023-04-14 23:00:34
176508	794567901549	4532	93051	2019-03-14	1	3.84	2023-04-14 23:00:00	2023-04-14 23:00:00
176526	794567902286	4533	61428	2019-03-25	1	14.95	2023-04-14 23:00:00	2023-04-14 23:00:00
176530	794567901549	4534	61373	2019-03-30	1	2.99	2023-04-14 23:00:00	2023-04-14 23:00:00
176547	794567901549	8807	65546	2019-03-01	5	14.95	2023-04-14 23:00:34	2023-04-14 23:00:34
176556	794567902286	8808	65942	2019-03-14	1	14.95	2023-04-14 23:00:34	2023-04-14 23:00:34
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

4.

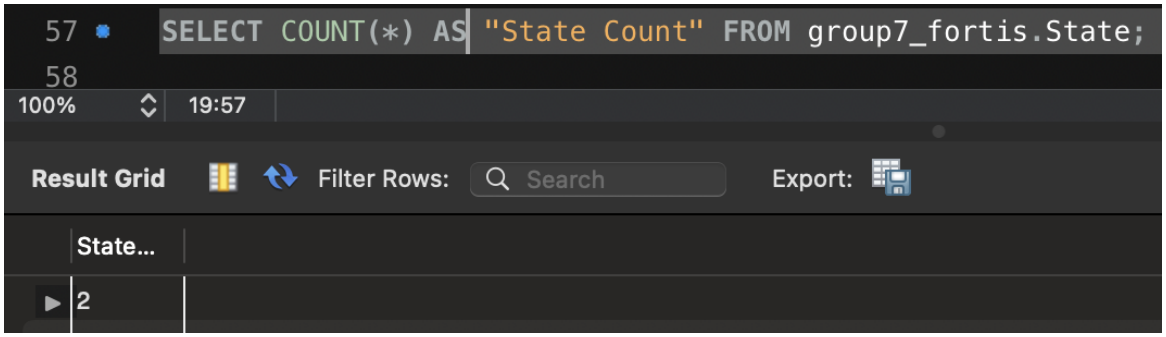
Displaying the count in the Product Group Table:



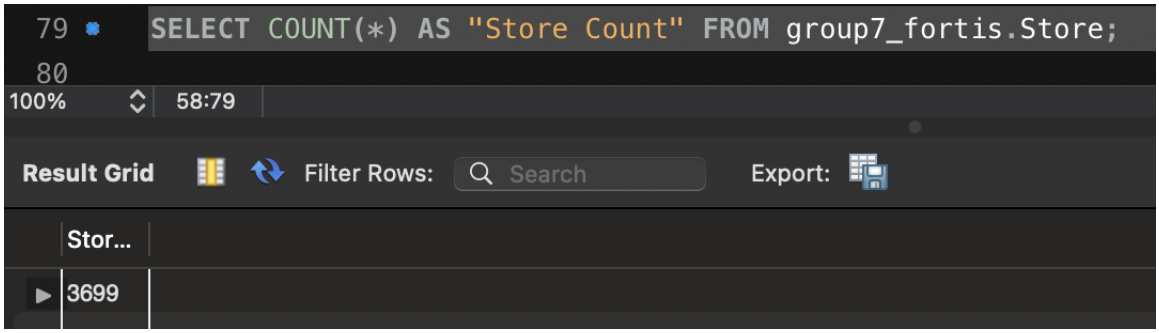
Displaying the count in the Customer table:



Displaying the count in the State table:



Displaying the count in the Store table:



Displaying the count in the Product table:

```
100 •   SELECT COUNT(*) AS "Product Count" FROM group7_fortis.Product;
101
```

100% 21:100

Result Grid | Filter Rows: Search | Export:

Product Count
18

Displaying the count in the Order table:

```
126 •   SELECT COUNT(*) AS "Order Count" FROM group7_fortis.Order;
127
```

100% 5:126

Result Grid | Filter Rows: Search | Export:

Ord...
8808

5.

Query 1:

```
130     # Query 1: Top 3 products to keep in stock
131 •   SELECT t1.ProductDesc AS "Product", SUM(t2.LineTotal) AS "Total Revenue"
132     FROM group7_fortis.Product t1
133     INNER JOIN group7_fortis.Order t2 ON t1.ProductNo = t2.ProductNo
134     GROUP BY t1.ProductDesc
135     ORDER BY SUM(t2.LineTotal) DESC
136     LIMIT 3;
137
```

100% 8:136

Result Grid | Filter Rows: Search | Export: | Fetch rows:

Product	Total Revenue
Macbook Pro Laptop	329800.00
iPhone	233100.00
ThinkPad Laptop	195998.04

Initially, our group thought that it would make the most sense to keep in stock the products that are bought most frequently; however, after running queries, it occurred to us that, just because a product is ordered more often than another, it does not necessarily translate to higher revenue. We considered what made most business sense and decided that recommending the products

that bring in the most revenue are more valuable to the company than recommending the products that are bought more often. From the query above, it is apparent that the top three revenue generators from highest to lowest are the Macbook Pro Laptop, the iPhone, and the ThinkPad Laptop.

Query 2:

```
# Query 2: Top 3 products to not keep in stock
SELECT t1.ProductDesc AS "Product", SUM(t2.LineTotal) AS "Total Revenue"
FROM group7_fortis.Product t1
INNER JOIN group7_fortis.Order t2 ON t1.ProductNo = t2.ProductNo
GROUP BY t1.ProductDesc
ORDER BY SUM(t2.LineTotal) ASC
LIMIT 3;
```

100% 8:144

Result Grid | Filter Rows: Search | Export: | Fetch rows:

Product	Total Revenue
AAA Batteries (4-pack)	9498.90
Wired Headphones	11582.34
USB-C Charging Cable	13133.05

Opposite to the first query, Sales Area 2 should consider not carrying the products that generate the least amount of revenue. The three products that were found to generate the least revenue from lowest to highest were AAA Batteries (4-pack), Wired Headphones, and USB-C Charging Cable. We think it is important to note that, had we gone with our initial recommendation of keeping in stock the products with the highest quantity ordered, we would have been recommending the top product and the third top product that generate the least revenue, thus possibly risking vital revenue for Fortis Electronics.

Query 3:

```
# Query 3: Out of the products with a product group, what are the top 3 most ordered in January?
SELECT ProductDesc AS "Product Name", ProductGroupDesc AS "Type of Product", SUM(OrderQty) AS "Quantity Ordered"
FROM group7_fortis.ProductGroup t1
INNER JOIN group7_fortis.Product t2 ON t1.ProductGroupID = t2.ProductGroupID
INNER JOIN group7_fortis.Order t3 ON t2.ProductNo = t3.ProductNo
WHERE OrderDate <= "2019-01-31" AND ProductGroupDesc <> "NULL"
GROUP BY ProductGroupDesc, ProductDesc
ORDER BY SUM(OrderQty) DESC
LIMIT 3;
```

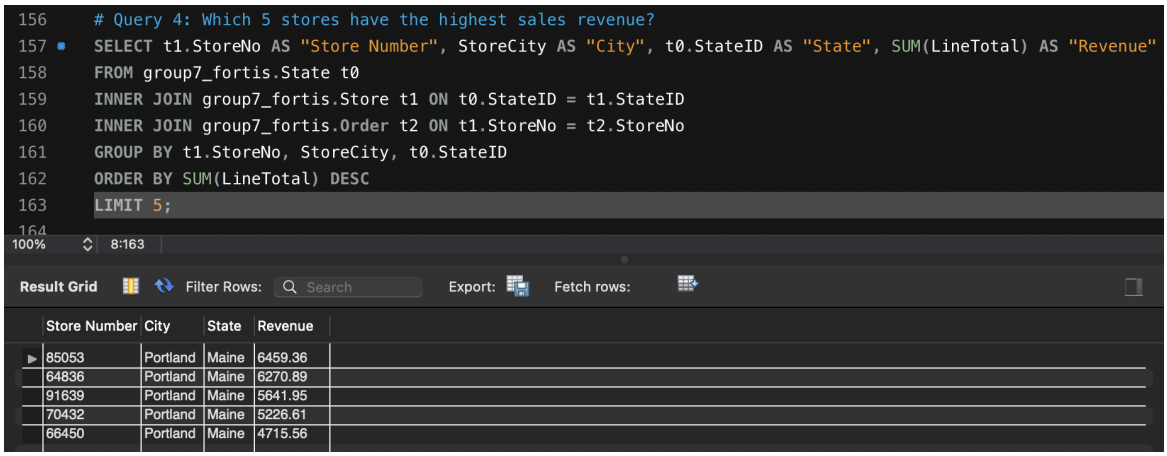
100% 9:154

Result Grid | Filter Rows: Search | Export: | Fetch rows:

Product Name	Type of Produ...	Quantity Ordered
USB-C Charging Cable	Accessory	298
Lightning Charging Cable	Accessory	255
Wired Headphones	Headphone	249

To find insight as to how products with a product group perform during a certain time of the year, we ran a query that found, out of all of the products that have a product group, which three were ordered the most often in the month of January. From highest frequency to lowest frequency, these products were the USB-C Charging Cable, the Lightning Charging Cable, and the Wired Headphones. The former two products are both grouped as Accessory, thus leading us to recommend to the company that they stock more products grouped as Accessory in the month of January in order to satisfy customer demand.

Query 4:



```
156  # Query 4: Which 5 stores have the highest sales revenue?
157  SELECT t1.StoreNo AS "Store Number", StoreCity AS "City", t0.StateID AS "State", SUM(LineTotal) AS "Revenue"
158  FROM group7_fortis.State t0
159  INNER JOIN group7_fortis.Store t1 ON t0.StateID = t1.StateID
160  INNER JOIN group7_fortis.Order t2 ON t1.StoreNo = t2.StoreNo
161  GROUP BY t1.StoreNo, StoreCity, t0.StateID
162  ORDER BY SUM(LineTotal) DESC
163  LIMIT 5;
164
```

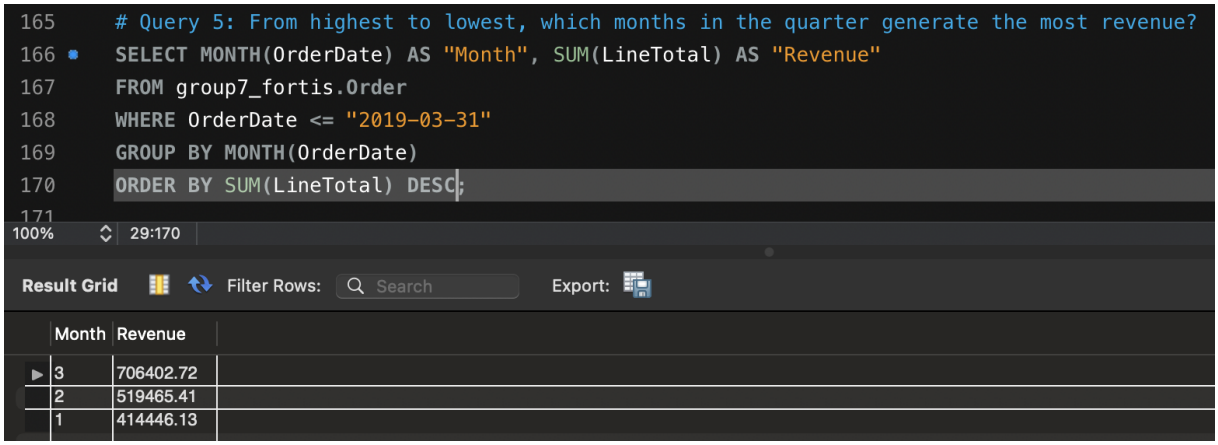
100% 8:163

Result Grid Filter Rows: Search Export: Fetch rows:

Store Number	City	State	Revenue
85053	Portland	Maine	6459.36
64836	Portland	Maine	6270.89
91639	Portland	Maine	5641.95
70432	Portland	Maine	5226.61
66450	Portland	Maine	4715.56

Next, we wanted to focus on the top revenue performers in Sales Area 2. This query found the top 5 stores in our specific sales area based on the revenue they generated. For more context, we also found their corresponding city and state. We recommend Fortis Electronics to determine what makes these stores successful relative to all stores in the region and possibly use that information to increase the revenue of the other stores in Portland, Maine and all of those in Boston, Massachusetts.

Query 5:



```
165  # Query 5: From highest to lowest, which months in the quarter generate the most revenue?
166  SELECT MONTH(OrderDate) AS "Month", SUM(LineTotal) AS "Revenue"
167  FROM group7_fortis.Order
168  WHERE OrderDate <= "2019-03-31"
169  GROUP BY MONTH(OrderDate)
170  ORDER BY SUM(LineTotal) DESC;
171
```

100% 29:170

Result Grid Filter Rows: Search Export:

Month	Revenue
3	706402.72
2	519465.41
1	414446.13

Considering the data covered only the first quarter of 2019, our group decided to see and rank how much revenue was generated in January, February, and March. We found that March

generated the most revenue, February generated the second most revenue, and January generated the least revenue. We recommend that Fortis use this finding to discover why January generates such less revenue in comparison to March considering the relatively short time frame between the two months. This could allow Fortis to capitalize on potential untapped revenue and perform better on a quarterly basis than in years past.

6.

When it came to a timeline for completing this project, our group decided to work in chunks in order to avoid cramming and leave room for asking questions, especially prior to class on Wednesday, April 12th. The first step was to tackle exporting the data dump into .csv files for each data load required to import the data into our tables. We decided to have our data cleaned and exported no later than Wednesday, April 5th so that we had a full week to load the data into MySQL and handle any potential hiccups.

After the data was properly cleaned and exported, our group decided not to load all of the data into the tables at once. Rather, we decided to start with the tables that did not require foreign keys, such as the Product Group table, the Customer table, and the State table. After this was completed in one session, we decided to move on with the load the next day and upload the data into the Product table, the Store table, and the Order table, having all of our data uploaded prior to what would have been class on Monday, April 10th.

Finally, we decided to have our queries completed (or at least attempted) prior to the beginning of class on Wednesday, April 12th in order to be able to ask any final questions regarding the data analysis. In having all of our data uploaded and queries completed before this date, we were left with three entire days to work on the corresponding report. In the end, we ended up having to fix a few errors that were easily remedied due to our now intimate knowledge of the data dump and our proposed database.

Indications that the data in Excel was dirty included the multi-valued attribute for Store Address, the format of the Price Each, Line Total, and Order Date columns, and the data for Product Group spread across multiple sheets. It was important to separate the Store Address column into Store Street, Store City, and Store Zip Code. The newly created Store State column from the Store Address was deleted considering there was already a column indicating which state the store was located in. The separation of the Store Address column was achieved using the Text to Columns ability in Excel.

In terms of the formatting of Price Each and Line Total, the dollar signs needed to be removed to obtain a general number format. The Order Date column also needed to be changed to the Year-Month-Day format in order to be imported into MySQL. Finally, the Product Group issue was addressed by using the VLOOKUP function in order to return the proper Product Group for an item based on its ProductNo. In order to avoid blanks/errors with products that did not have a designated product group, we used IFERROR to return “NULL” as the product group for the products in question.

The order in which the data was loaded was described above, but prior to this and the exporting of the data into the .csv files, the data was sorted in alphabetical order by state. By doing this, we were able to copy the data pertaining to only stores in Maine and Massachusetts, or Sales Area 2, and paste it as values in a new sheet, maintaining all of the original data while not having thousands of extra pieces of data loaded into the database. The data was then further copied and pasted corresponding to the columns required for each table into new sheets. Once this was accomplished, we utilized the Remove Duplicates function in Excel in order to properly load the data pertaining to the ProductNo, StateID, and StoreNo considering they all acted as primary keys of their respective tables.

One of the biggest takeaways our team had regarding an industrial strength database design process is that you are most likely not going to get the data load 100% correct on the first try. Each of our tables was truncated numerous times after failed data import attempts. We also had to export the data from Excel with tweaks after realizing things about our data as we went. Because of this takeaway, we were aware of how important it was to not leave a task such as this one to the last minute. This takeaway was proven true even further after you made us aware of our data redundancy issue, which we would like to again thank you for allowing us an extension on this assignment in order to properly fix it.

The other takeaway our group had was the value of outside help. Not only are you not going to get the design process correct on the first try, but you are also going to run into obstacles that you do not necessarily know how to overcome on your own. In particular, the value of forums cannot be overstated. Rather than immediately send an email to you for help or sit tight and wait to ask in person before class, it was vital to the process that we be able to sift through a myriad of websites and already answered questions in order to at least attempt to solve our problems, thus not stalling the process while allowing us to further expand our database design knowledge.