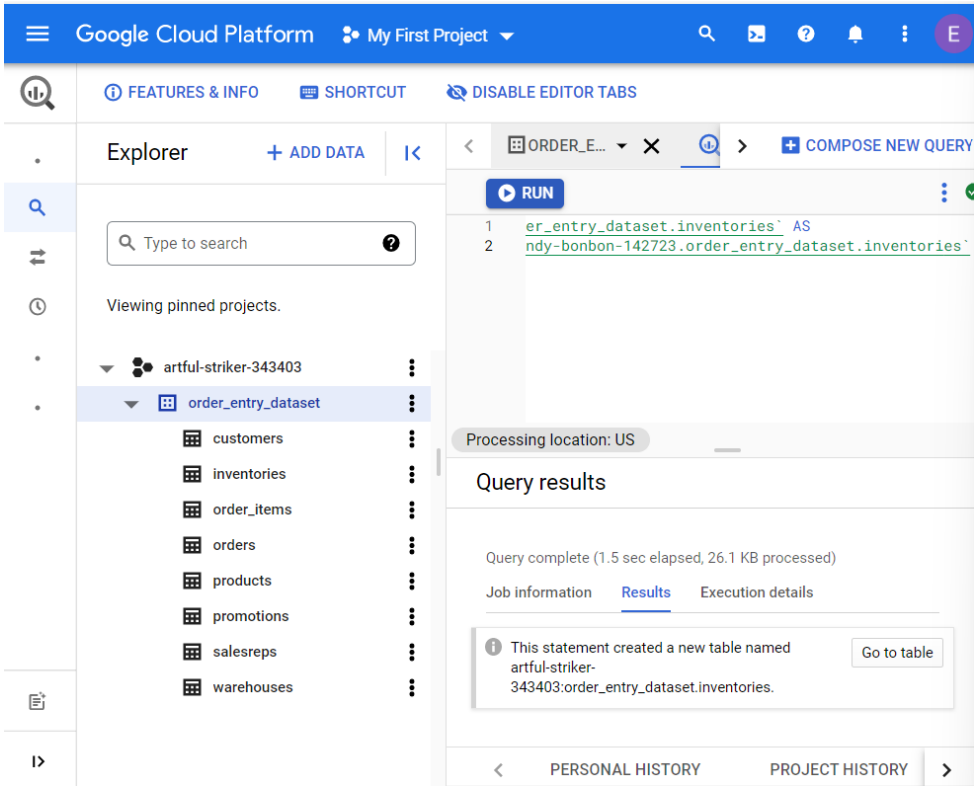
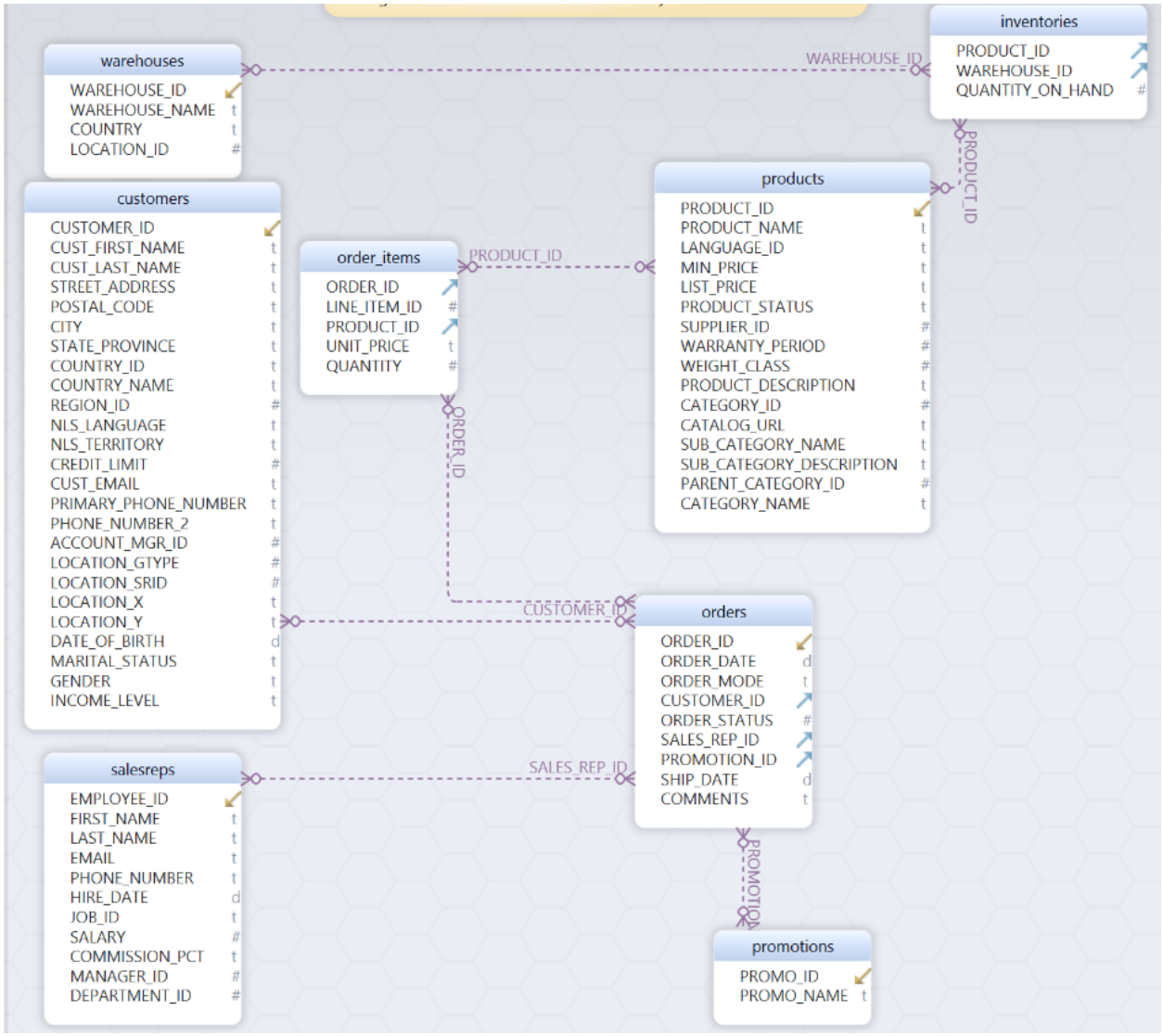
Esme Gonzalez
CIS 4400 Data Warehousing for Analytics
Homework Assignment #2 Analytical SQL

1. 

2. 
warehouses
WAREHOUSE_ID
WAREHOUSE_NAME
COUNTRY
LOCATION_ID
inventories
PRODUCT_ID
WAREHOUSE_ID
QUANTITY_ON_HAND
customers
CUSTOMER_ID
CUST_FIRST_NAME
CUST_LAST_NAME
STREET_ADDRESS
POSTAL_CODE
CITY
STATE_PROVINCE
COUNTRY_ID
COUNTRY_NAME
REGION_ID
NLS_LANGUAGE
NLS_TERRITORY
CREDIT_LIMIT
CUST_EMAIL
PRIMARY_PHONE_NUMBER
PHONE_NUMBER_2
ACCOUNT_MGR_ID
LOCATION_GTYPE
LOCATION_SRID
LOCATION_X
LOCATION_Y
DATE_OF_BIRTH
MARITAL_STATUS
GENDER
INCOME_LEVEL
order_items
ORDER_ID
LINE_ITEM_ID
PRODUCT_ID
UNIT_PRICE
QUANTITY
products
PRODUCT_ID
PRODUCT_NAME
LANGUAGE_ID
MIN_PRICE
LIST_PRICE
PRODUCT_STATUS
SUPPLIER_ID
WARRANTY_PERIOD
WEIGHT_CLASS
PRODUCT_DESCRIPTION
CATEGORY_ID
CATALOG_URL
SUB_CATEGORY_NAME
SUB_CATEGORY_DESCRIPTION
PARENT_CATEGORY_ID
CATEGORY_NAME
orders
ORDER_ID
ORDER_DATE
ORDER_MODE
CUSTOMER_ID
ORDER_STATUS
SALES_REP_ID
PROMOTION_ID
SHIP_DATE
COMMENTS
salesreps
EMPLOYEE_ID
FIRST_NAME
LAST_NAME
EMAIL
PHONE_NUMBER
HIRE_DATE
JOB_ID
SALARY
COMMISSION_PCT
MANAGER_ID
DEPARTMENT_ID
promotions
PROMO_ID
PROMO_NAME
WAREHOUSE_ID
PRODUCT_ID
PRODUCT_ID
ORDER_ID
CUSTOMER_ID
SALES_REP_ID
PROMOTION_ID

3. Write a SQL statement to show the total dollar amount sold to customers summarized by product category name and each month of each Year (YYYY-MM). Only include non-canceled orders that were shipped within 7 days of ordering

SQL Q3:

```
SELECT FORMAT_DATE("%Y-%m", order_date) AS Order_month,
category_name,
 ROUND( SUM(unit_price * quantity), 2) AS Amount_sold
FROM `order_entry_dataset.customers`
 INNER JOIN
 `order_entry_dataset.orders` USING (customer_id)
 INNER JOIN
 `order_entry_dataset.order_items` USING (order_id)
 inner join
 `order_entry_dataset.products` USING (product_id)
Where ORDER_STATUS >= 4 AND DATE_DIFF(ship_date, order_date, DAY) < 7
GROUP BY Order_month,CATEGORY_NAME
ORDER BY Order_month,CATEGORY_NAME;
```

85 Results:

Query complete (1.2 sec elapsed, 80.2 KB processed)

Job information | Results | JSON | Execution details

Row	Order_month	category_name	Amount_sold
1	2019-06	hardware	77781.5
2	2019-06	office equipment	5501.7
3	2019-07	hardware	33114.4
4	2019-07	office equipment	13530.9
5	2019-07	software	18603.2
6	2019-08	hardware	84736.8
7	2019-08	office equipment	12759.3

Rows per page: 100 1 - 85 of 85

4. Write a SQL statement to show the total dollar amount sold summarized by Customer marital status and Year along with RANK. The largest sales by marital status by year should be ranked #1.

Question 4:

```
SELECT FORMAT_DATE("%Y", order_date) AS Year_Order,
       marital_status,
       ROUND( SUM(unit_price * quantity), 2) AS Total_Amount_Sold,
       RANK() OVER (ORDER BY SUM(unit_price * quantity) DESC)AS Ranking
FROM  `order_entry_dataset.customers`
       INNER JOIN
       `order_entry_dataset.orders` USING (customer_id)
       INNER JOIN
       `order_entry_dataset.order_items` USING (order_id)
GROUP BY Year_Order, marital_status
ORDER BY Ranking ASC , Total_Amount_sold;
```

7 Results:

Row	Year_Order	marital_status	Total_Amount_Sold	Ranking
1	2021	married	2998764.0	1
2	2021	single	2422835.5	2
3	2020	married	1878543.1	3
4	2019	married	1611426.8	4
5	2020	single	1120060.1	5
6	2019	single	522361.1	6
7	2022	married	65571.8	7

5. Write a SQL statement to show the total dollar amount sold across product categories for all orderable products. Calculate the percentage contribution of each product category's sales to the overall total sales.

SQL Q5:

```
SELECT category_name, Amount_Sold, SUM(Amount_Sold) OVER () AS TOTAL,
100*Amount_Sold/SUM(Amount_Sold) OVER () AS Percentage
FROM
( SELECT category_name, ROUND( SUM(unit_price * quantity), 2) AS Amount_Sold
FROM
 `order_entry_dataset.orders`
 INNER JOIN
 `order_entry_dataset.order_items` USING (order_id)
 inner join
 `order_entry_dataset.products` USING (product_id)
 WHERE SHIP_DATE IS NOT null
GROUP BY CATEGORY_NAME
ORDER BY CATEGORY_NAME)
```

3 Results:

Row	category_name	Amount_Sold	TOTAL	Percentage
1	office equipment	3179422.0	1.019469E7	31.187039527440266
2	hardware	6281185.8	1.019469E7	61.61232759407103
3	software	734082.2	1.019469E7	7.200632878488704

6. Write a SQL statement to show the most profitable product over all orders. (unit price above Min Price). Only consider products that are available in the US or Canadian warehouses with list price over $50.

SQL Q6:

```
SELECT (Unit_price - Min_price) AS Profitable , Country , List_Price , Product_ID ,
Product_Name
FROM `order_entry_dataset.order_items`
 inner join
 `order_entry_dataset.products` USING (product_id)
 inner join
 `order_entry_dataset.inventories` USING (product_id)
```

```
inner join
`order_entry_dataset.warehouses` USING (warehouse_id)
WHERE country = ('US')
OR country = ('CA')
AND list_price>50
ORDER BY Profitable DESC
```

2416 Results:

Row	Profitable	Country	List_Price	Product_ID	Product_Name
1	193.69999999999982	US	3379.95	3003	Laptop 128/12/56/v90/110
2	193.69999999999982	US	3379.95	3003	Laptop 128/12/56/v90/110
3	193.69999999999982	US	3379.95	3003	Laptop 128/12/56/v90/110
4	193.69999999999982	US	3379.95	3003	Laptop 128/12/56/v90/110
5	193.69999999999982	CA	3379.95	3003	Laptop 128/12/56/v90/110

Rows per page: 100 1 - 100 of 2416 First page

7. Which month (be sure to say from which year) had the largest percentage increase in sales over the prior month? Justify your rationale and show your SQL query (Hint: use the LAG function).

SQL Q7:

```
SELECT Order_Month, MonthlySales, PriorMonthSales, MonthlySales-PriorMonthSales AS
MonthlySalesDifference, 100*ABS((MonthlySales-PriorMonthSales))/ (PriorMonthSales) AS
PercentageChangeInSales
FROM(SELECT Order_Month, MonthlySales,
LAG (MonthlySales,1,0)
OVER (ORDER BY Order_Month) AS PriorMonthSales
FROM
(SELECT FORMAT_DATE("%Y-%m", DATE(order_date)) AS Order_Month, SUM(Unit_price *
Quantity) AS MonthlySales
FROM  `order_entry_dataset.customers`
       INNER JOIN
       `order_entry_dataset.orders` USING (customer_id)
       INNER JOIN
       `order_entry_dataset.order_items` USING (order_id)
```

```
GROUP BY Order_Month))
WHERE PriorMonthSales <> 0
ORDER BY PercentageChangeInSales DESC
```

31 Results:

Row	Order_Month	MonthlySales	PriorMonthSales	MonthlySalesDifference	PercentageChangeInSales
1	2019-09	434015.5	97496.1	336519.4	345.16190904046414
2	2020-12	759097.4999999998	198009.70000000007	561087.7999999997	283.36379480399165
3	2020-05	338531.0999999998	130551.4	207979.6999999998	159.30867076109473
4	2019-07	390539.4000000001	152899.99999999997	237639.4000000001	155.4214519293657
5	2020-10	417839.49999999994	165608.7	252230.79999999993	152.30528347846453
6	2019-12	406763.10000000003	162071.9	244691.20000000004	150.9769429493947
7	2021-06	1038720.5000000000	426148.5000000001	612582.0000000007	143.74850550923066

Rows per page: 100 1 - 31 of 31

8. Who is the “best” Sales Manager? Justify your rationale and back it up with queries and data. You may also wish to graph various data to support your justification. DO NOT just total up sales. Consider multiple factors and build a weighted model with SQL. Look at other tables beyond just orders.

SQL Q8 Total sales:

```
SELECT Manager_ID ,
       ROUND( SUM(unit_price * quantity), 2) AS Total_Amount_Sold,
FROM `order_entry_dataset.salesreps` AS S
Inner Join `order_entry_dataset.orders` AS O
on S.Employee_ID=O.Sales_Rep_ID
INNER JOIN `order_entry_dataset.order_items` USING (order_id)
GROUP BY Manager_ID
ORDER BY Total_Amount_sold DESC
```

5 Results:

Row	Manager_ID	Total_Amount_Sold
1	147	2512812.1
2	148	2412208.9
3	149	2149284.6
4	146	1649670.6
5	145	1647657.6

As you can see from the query results, Manger ID 147 has the highest total amount of sales then the four other managers. To create the highest amount to be row one I did order Total_Amount_sold. This shows who had the second highest amount the the result. Another Example would be to do profitability from another example like six.

SQL Q8 Profitability:

```
SELECT Manager_ID, SUM(Unit_price - Min_price) AS Profitable,
FROM `order_entry_dataset.salesreps` AS S
Inner Join `order_entry_dataset.orders` AS O
on S.Employee_ID=O.Sales_Rep_ID
INNER JOIN `order_entry_dataset.order_items` USING (order_id)
INNER JOIN `order_entry_dataset.products` USING (product_id)
GROUP BY Manager_ID
ORDER BY profitable DESC
```

5 Results:

Row	Manager_ID	Profitable
1	147	5552.550000000007
2	149	3896.2000000000025
3	148	3821.0500000000056
4	146	3376.7250000000013
5	145	3087.7999999999997

Looking at my results on this query, you can see how again we have five results with Manger Id 147 having the highest results . Showing that 147 is the best sales manager but another example is seeing the amount of customers.

SQL Q8 Amount of Customers:

```
SELECT SUM(Customer_ID) AS Amount_of_Customers, Manager_ID
FROM `order_entry_dataset.salesreps` AS S
Inner Join `order_entry_dataset.orders` AS O
on S.Employee_ID=O.Sales_Rep_ID
INNER JOIN `order_entry_dataset.order_items` USING (order_id)
INNER JOIN `order_entry_dataset.customers` USING (customer_id)
GROUP BY  Manager_ID
ORDER BY  Amount_of_Customers DESC
```

5 Results:

Row	Amount_of_Customers	Manager_ID
1	115653	148
2	113919	147
3	89997	149
4	73564	146
5	60439	145

This query shows that Manger Id 148 has the most customers out of everyone but this proves that 147 is the best manager. The reason is because 147 made the most sales with not the highest number of customers. If you remember from the first query 147 was the highest and 148 was the second highest in sales. If you think about it, if 147 and 148 have the same amount of customers then 147 will still have the highest number of sales.

9) Create a query that joins together all of the columns in these tables: PRODUCTS, CUSTOMERS, ORDERS, ORDER_ITEMS, PROMOTIONS and SALES REP. Be aware that not all orders have sales reps. Export this view to a CSV file. Click Save Results button and then select CSV File (Google Drive). Download the CSV file from Google Drive.

SQL Q9:

```
SELECT *
FROM `order_entry_dataset.salesreps` AS SR
Inner Join `order_entry_dataset.orders` AS ORT
on SR.EMPLOYEE_ID=ORT.SALES_REP_ID
INNER JOIN `order_entry_dataset.order_items` USING (order_id)
INNER JOIN `order_entry_dataset.promotions` AS p
on p.PROMO_ID=ORT.PROMOTION_ID
inner join
```

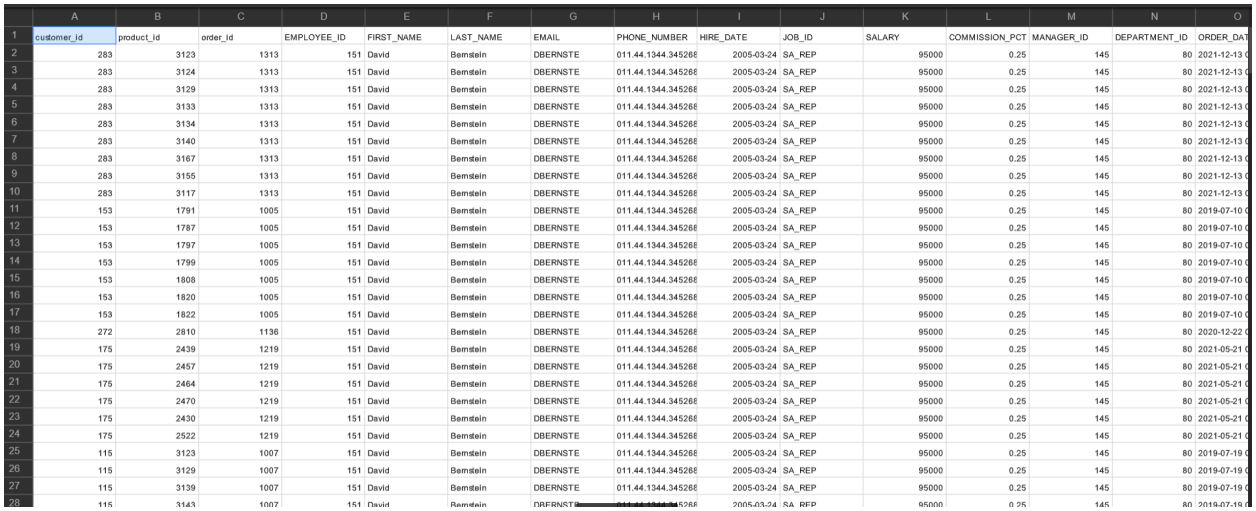
```
        `order_entry_dataset.products` USING (product_id)
INNER JOIN `order_entry_dataset.customers` USING (customer_id)
Where sales_rep_id is NOT null
```

1789 Results:

Row	customer_id	product_id	order_id	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_
1	283	3123	1313	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	
2	283	3124	1313	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	
3	283	3129	1313	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	
4	283	3133	1313	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	
5	283	3134	1313	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	

Rows per page: 100 | 1 - 100 of 1789 | First page | Last page

https://drive.google.com/drive/folders/1p8sxJxqD0l1h5_W654Tqtq-ZPNj8FgAo?usp=sharing

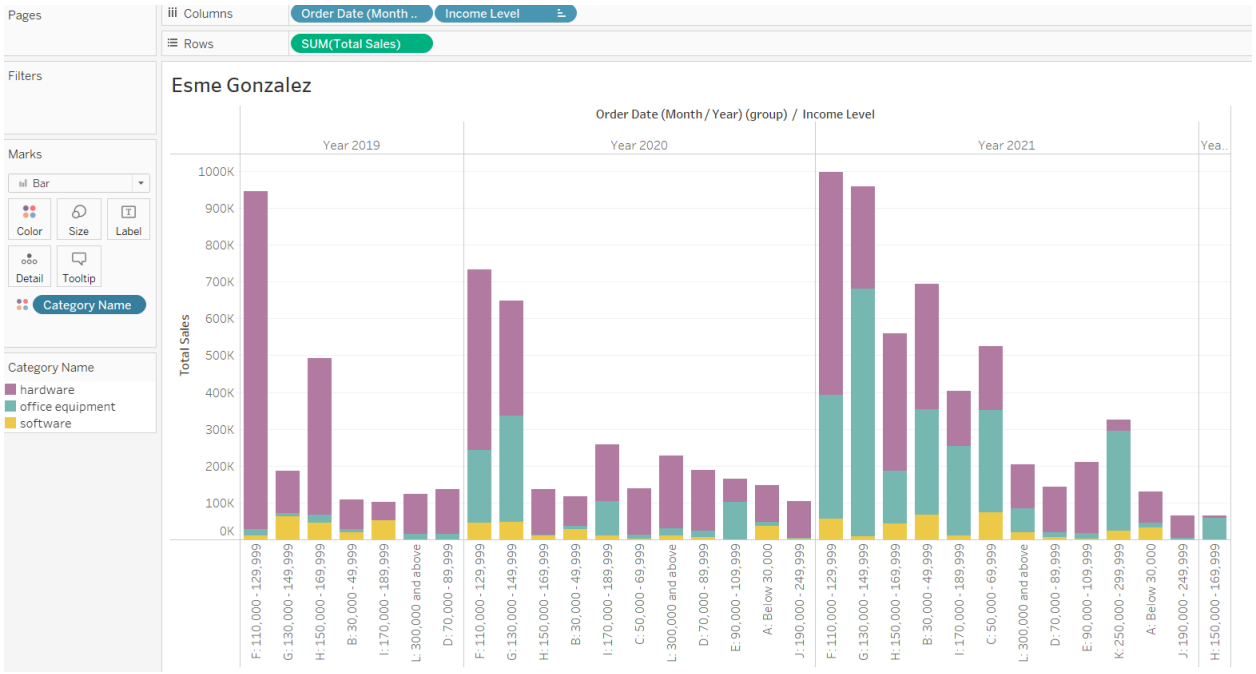


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	customer_id	product_id	order_id	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID	ORDER_DAT
2	283	3123	1313	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2021-12-13 0
3	283	3124	1313	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2021-12-13 0
4	283	3129	1313	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2021-12-13 0
5	283	3133	1313	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2021-12-13 0
6	283	3134	1313	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2021-12-13 0
7	283	3140	1313	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2021-12-13 0
8	283	3167	1313	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2021-12-13 0
9	283	3155	1313	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2021-12-13 0
10	283	3117	1313	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2021-12-13 0
11	153	1791	1005	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2019-07-10 0
12	153	1787	1005	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2019-07-10 0
13	153	1797	1005	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2019-07-10 0
14	153	1799	1005	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2019-07-10 0
15	153	1808	1005	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2019-07-10 0
16	153	1820	1005	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2019-07-10 0
17	153	1822	1005	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2019-07-10 0
18	272	2810	1136	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2020-12-22 0
19	175	2439	1219	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2021-05-21 0
20	175	2457	1219	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2021-05-21 0
21	175	2464	1219	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2021-05-21 0
22	175	2470	1219	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2021-05-21 0
23	175	2430	1219	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2021-05-21 0
24	175	2522	1219	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2021-05-21 0
25	115	3123	1007	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2019-07-19 0
26	115	3129	1007	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2019-07-19 0
27	115	3139	1007	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2019-07-19 0
28	115	3143	1007	151	David	Bernstein	DBERNSTE	011.44.1344.345268	2005-03-24	SA_REP	95000	0.25	145	80	2019-07-19 0

10) Import all of the data from your VIEW into Microsoft Excel. Create a pivot table from the resulting data set and then summarize the data according to total sales by product category and customer country.

Row Labels	Sum of Total Sales
CH	**81027471**
hardware	44116805.6
office equipment	3926980.2
software	108763.5
CN	**9982969.2**
hardware	4440473.7
office equipment	490842.8
software	18036.3
DE	**24870165.6**
hardware	11813304.8
office equipment	1310738.1
software	97991
IN	**24397250.5**
hardware	17153973
office equipment	424074
software	1659
IT	**86170076**
hardware	42715556.4
office equipment	2987759.5
software	695550.1
TH	**10430256**
hardware	7728384
office equipment	29798.5
software	62587.8
US	**13078826797**
hardware	5533609486
office equipment	802904461.2
software	121573919.2
(blank)	**0**
(blank)	0
Grand Total	**22500638172**

11) Import all of the data from your VIEW into Tableau or Google Data Studio. Create an appropriate visualization from the resulting data set that summarizes the data according to total sales by product category and customer income level over time (e.g., months or years).



At the end of the assignment answers, please tell me:

1. How many hours did you spend working on the assignment? I believe I spent 2-3 days on it
2. What was the most difficult part of completing the assignment? Getting started with the sql was the most difficult. The reason is to have a starting point and use all the learning materials provided. But overall, It was a great learning experience I will be able to use!

Sources:

- http://holowczak.com/exploring-analytical-sql-with-google-bigquery/7/
- http://holowczak.com/getting-started-with-google-bigquery-on-google-cloud-platform/6/
- http://holowczak.com/reverse-engineering-a-google-bigquery-schema-with-dbschema/5/
- http://holowczak.com/creating-a-service-account-and-key-file-for-google-bigquery/3/