Project Report

Analyze data of a model car database with MySQL Workbench

Author: Janit Jindal | Organization/Institution: Coursera

Table of contents: - **Page No.**

- **Project Scenario: -**

This project is to analyze a model car database with MySQL Workbench to see which product storage warehouse can be shut down with least obstacles.

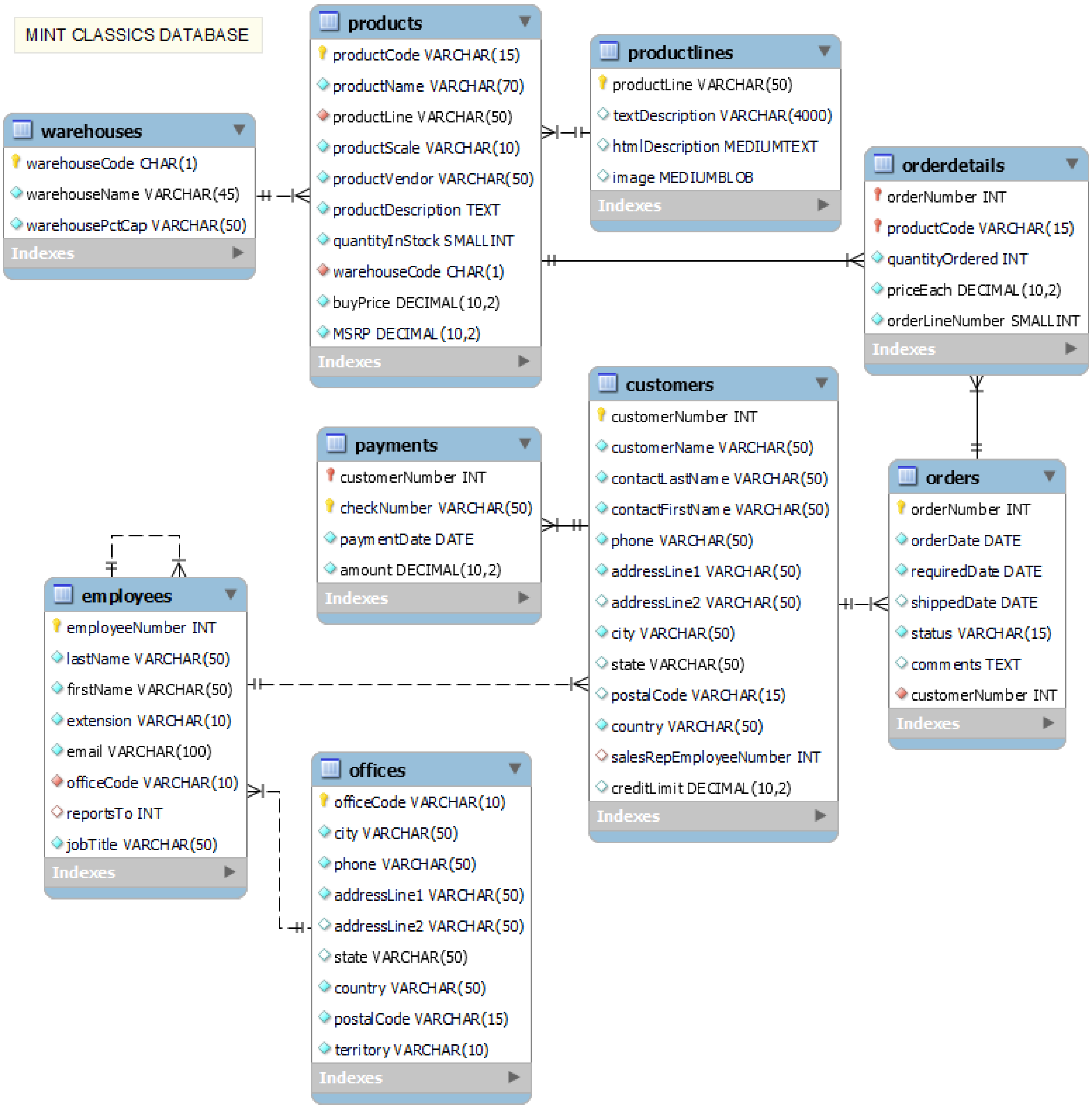
Mint Classics Company, a retailer of classic model cars and other vehicles, is looking at closing one of their storage facilities. To support a data-based business decision, they are looking for suggestions and recommendations for reorganizing or reducing inventory, while still maintaining timely service to their customers.

- **Project Objectives: -**

1. Explore products currently in inventory
2. Determine important factors that may influence inventory reorganization/reduction.
3. Provide analytic insights and data-driven recommendation.

- **About Data: -**

The Database contains nine entities with various features mainly representing the connection between sales and product purchasing by various customers. The EER (Extended Entity-Relationship) diagram that models the structure of the Mint Classics database is provided here:



➢ Methodology/ Method of approach:

To approach this obstacle, it would be convenient to break the objective into several questions and insights that are required to be answered: -

Q1. Which warehouse to eliminate?

Description: As company's basic demand is to shut down a warehouse, we need to think which warehouse is the least and the most beneficial to the company.

Q2. Which warehouse can handle more products of required transferring?

Description: Even if we manage to give insights on shutting down the warehouse, we need to resolve inventory accommodation for all of the products left behind.

Q3. Which employees will be transferred to which location?

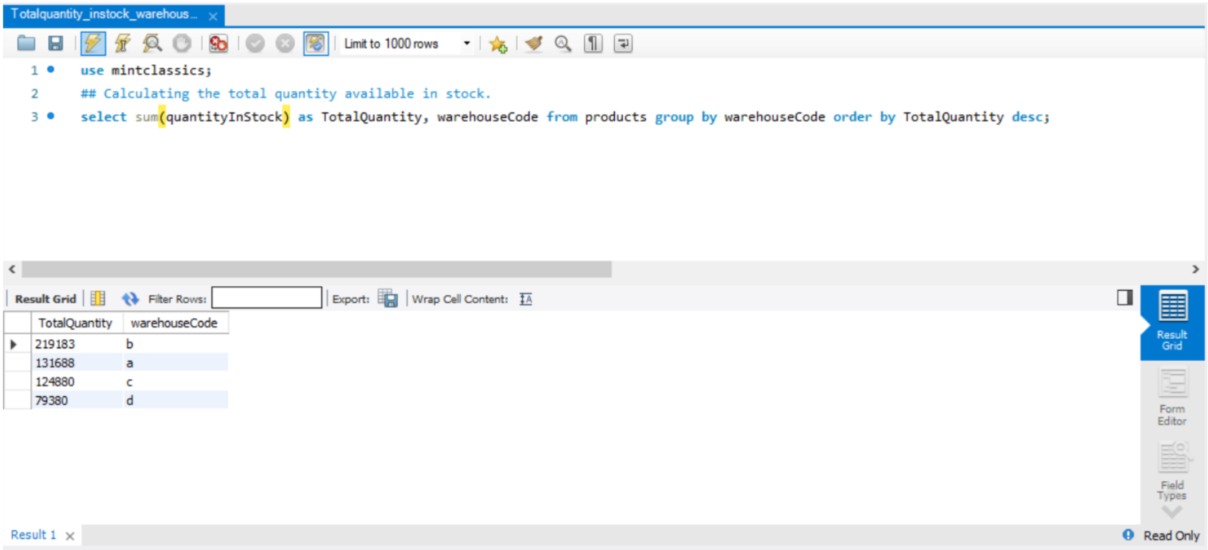
Description: After the transfer of goods, more work power will be required in the remaining warehouses, thus it would be convenient to observe who are the top salesman of the company.

Steps used to gain insights: -

- Step1:

There are four warehouses, warehouse a, b, c and d.

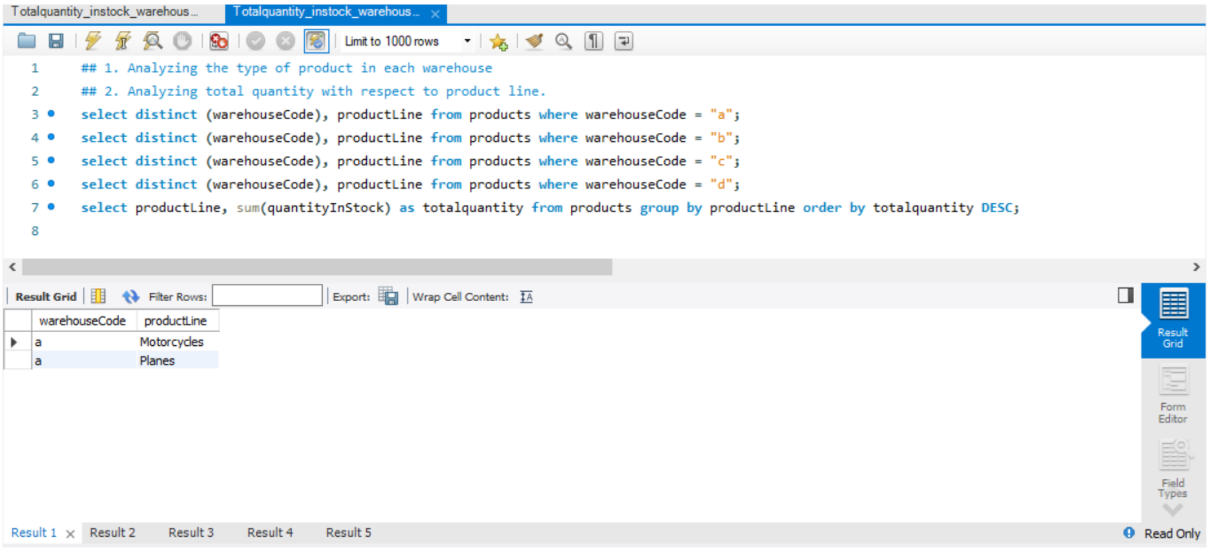
To initiate, I used the "products" table to analyze the total amount of products available in each warehouse and representing it in descending order (as shown in figure-a). Along with this, "productLine" entity was used to review the variety of products in each warehouse (as shown in figure-b.1, b.2, b.3, b.4).



```
use mintclassics;
## Calculating the total quantity available in stock.
select sum(quantityInStock) as TotalQuantity, warehouseCode from products group by warehouseCode order by TotalQuantity desc;
```

TotalQuantity	warehouseCode
219183	b
131688	a
124880	c
79380	d

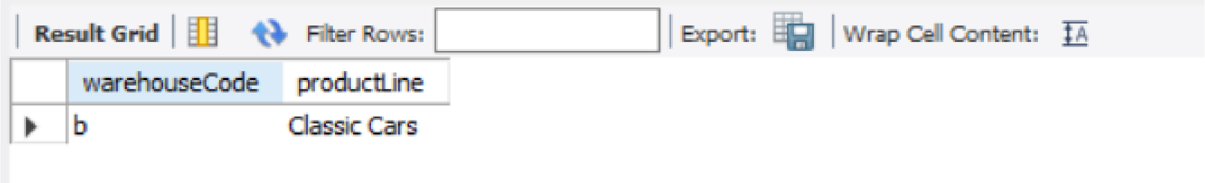
Fig. (a)



```
## 1. Analyzing the type of product in each warehouse
## 2. Analyzing total quantity with respect to product line.
select distinct (warehouseCode), productLine from products where warehouseCode = "a";
select distinct (warehouseCode), productLine from products where warehouseCode = "b";
select distinct (warehouseCode), productLine from products where warehouseCode = "c";
select distinct (warehouseCode), productLine from products where warehouseCode = "d";
select productLine, sum(quantityInStock) as totalquantity from products group by productLine order by totalquantity DESC;
```

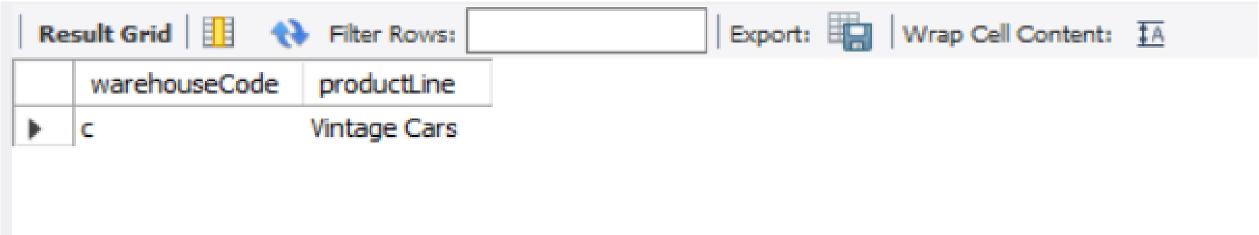
warehouseCode	productLine
a	Motorcycles
a	Planes

Fig. (b.1)



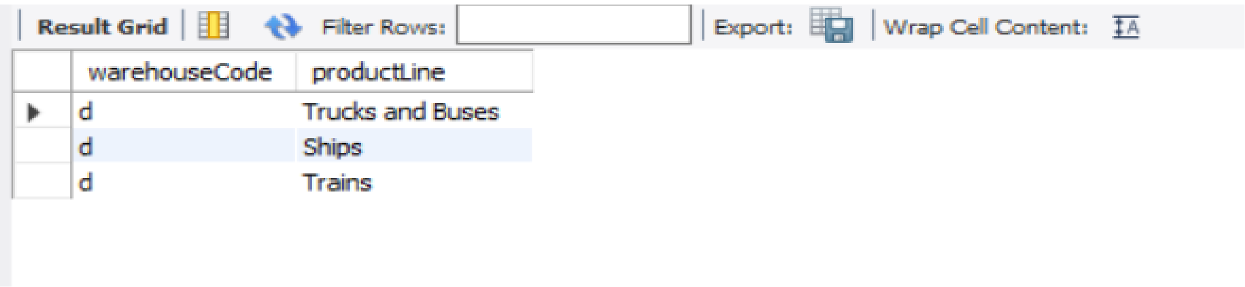
warehouseCode	productLine
b	Classic Cars

Fig. (b.2)



warehouseCode	productLine
c	Vintage Cars

Fig. (b.3)



warehouseCode	productLine
d	Trucks and Buses
d	Ships
d	Trains

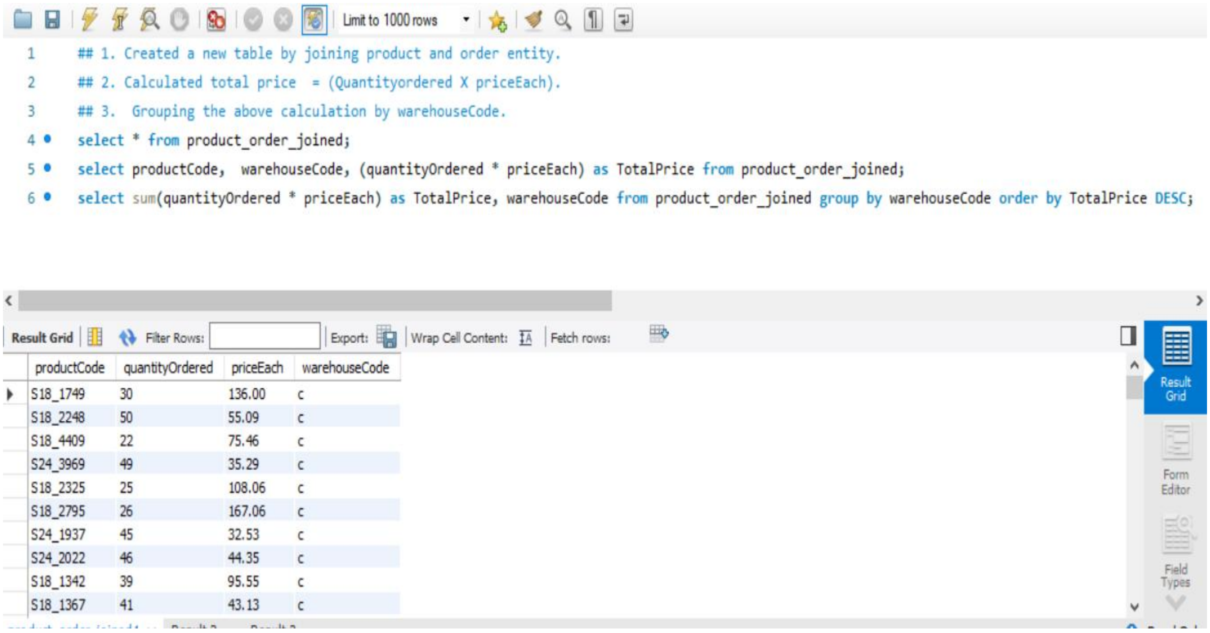
Fig. (b.4)

- Step2:

After reviewing the quantity of product available, I had searched the total prices of products by combining “products” and “orders” entities using left join on “productCode” (show in figure-c.1).

After that, by multiplying quantity ordered with each price given (shown in figure-c.2), a new feature “total price” had been created to see which warehouse had the highest cost products (shown in figure-c.3).

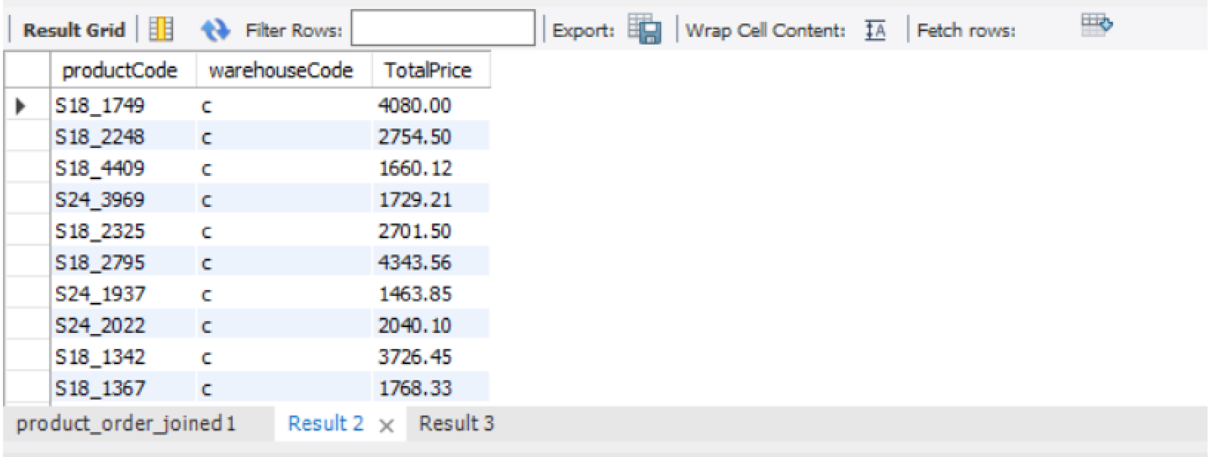
Here, we can consider the price given as total revenue earned as the order entity represents each order/sale taken and processed through the warehouses.



```
## 1. Created a new table by joining product and order entity.
## 2. Calculated total price  = (Quantityordered X priceEach).
## 3.  Grouping the above calculation by warehouseCode.
select * from product_order_joined;
select productCode,  warehouseCode, (quantityOrdered * priceEach) as TotalPrice from product_order_joined;
select sum(quantityOrdered * priceEach) as TotalPrice, warehouseCode from product_order_joined group by warehouseCode order by TotalPrice DESC;
```

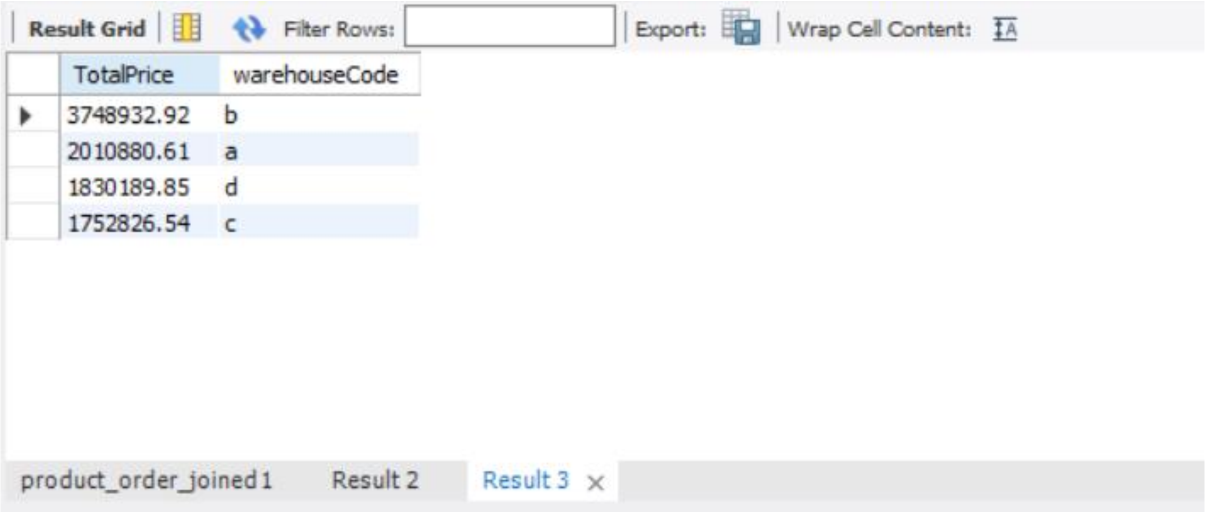
productCode	quantityOrdered	priceEach	warehouseCode
S18_1749	30	136.00	c
S18_2248	50	55.09	c
S18_4409	22	75.46	c
S24_3969	49	35.29	c
S18_2325	25	108.06	c
S18_2795	26	167.06	c
S24_1937	45	32.53	c
S24_2022	46	44.35	c
S18_1342	39	95.55	c
S18_1367	41	43.13	c

Fig. (c.1)



productCode	warehouseCode	TotalPrice
S18_1749	c	4080.00
S18_2248	c	2754.50
S18_4409	c	1660.12
S24_3969	c	1729.21
S18_2325	c	2701.50
S18_2795	c	4343.56
S24_1937	c	1463.85
S24_2022	c	2040.10
S18_1342	c	3726.45
S18_1367	c	1768.33

Fig. (c.2)

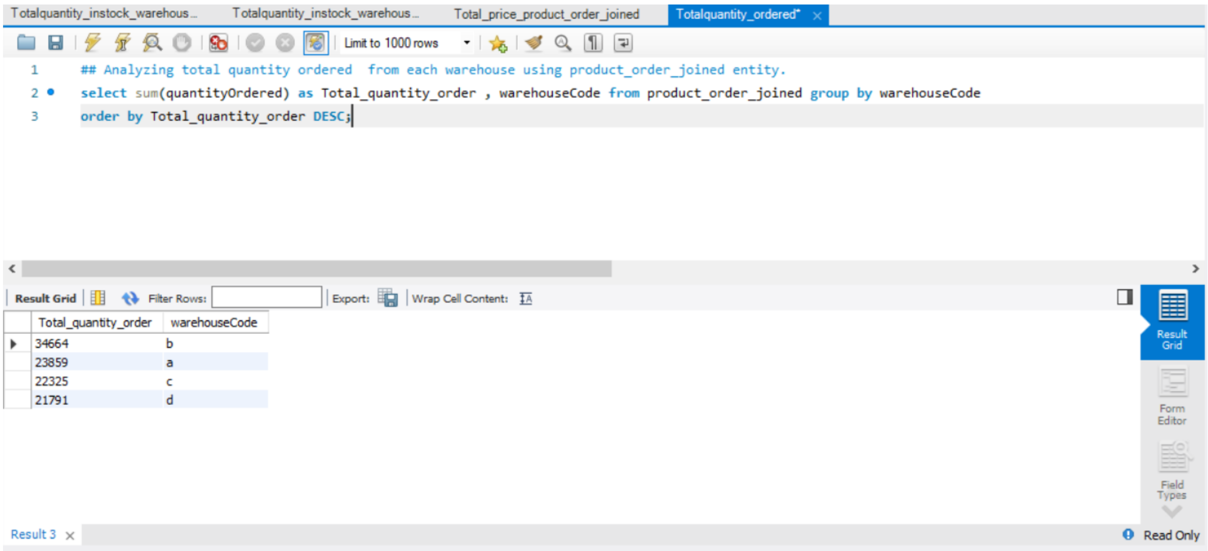


TotalPrice	warehouseCode
3748932.92	b
2010880.61	a
1830189.85	d
1752826.54	c

Fig. (c.3)

- <u>Step3:</u>

  To match the above results, I had summed the quantity ordered and displayed it with each warehouse (shown in figure-d).



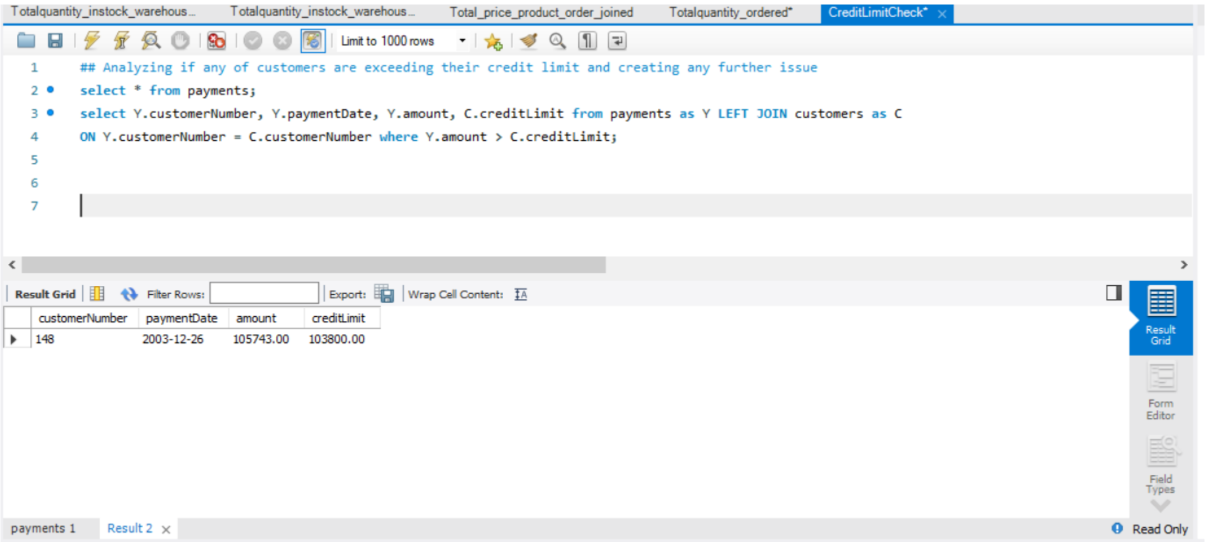
```
## Analyzing total quantity ordered  from each warehouse using product_order_joined entity.
select sum(quantityOrdered) as Total_quantity_order , warehouseCode from product_order_joined group by warehouseCode
order by Total_quantity_order DESC;
```

Total_quantity_order	warehouseCode
34664	b
23859	a
22325	c
21791	d

Fig. (d)

- Step4:

  As a precautionary measure to not go forward with any failure of sale and orders processed, a LEFT JOIN command was implemented between the “payments” and “customers” tables, which further analyzed any customer surpassing their credit limit (as shown in figure-e).



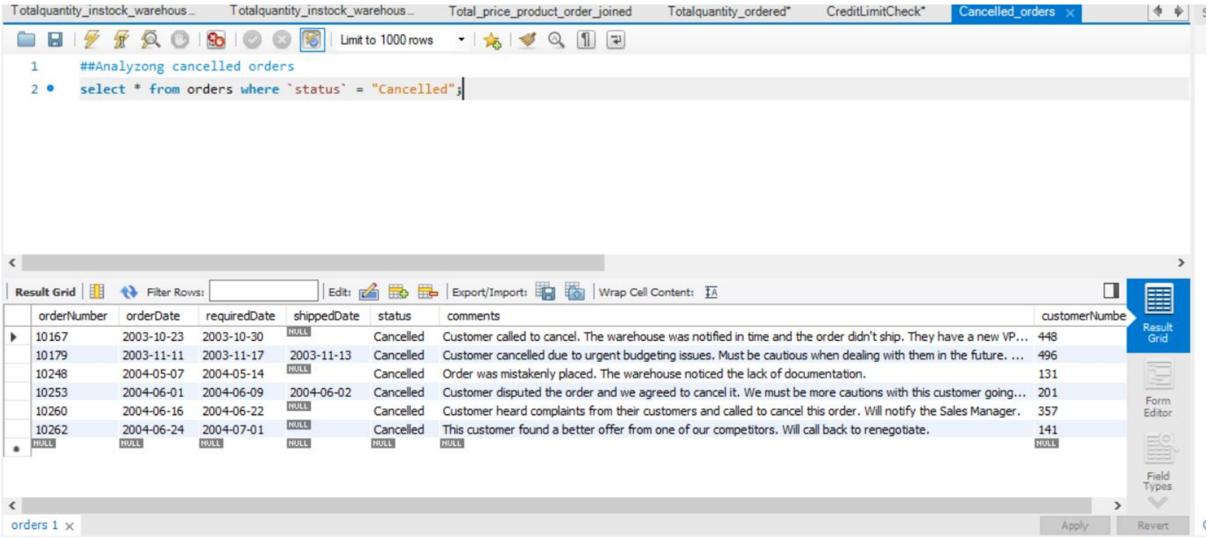
```
## Analyzing if any of customers are exceeding their credit limit and creating any further issue
select * from payments;
select Y.customerNumber, Y.paymentDate, Y.amount, C.creditLimit from payments as Y LEFT JOIN customers as C
ON Y.customerNumber = C.customerNumber where Y.amount > C.creditLimit;
```

customerNumber	paymentDate	amount	creditLimit
148	2003-12-26	105743.00	103800.00

Fig. (e)

- Step5:

  Following the above step, each “cancelled” order was reviewed for any errors occurred in the warehouse production.



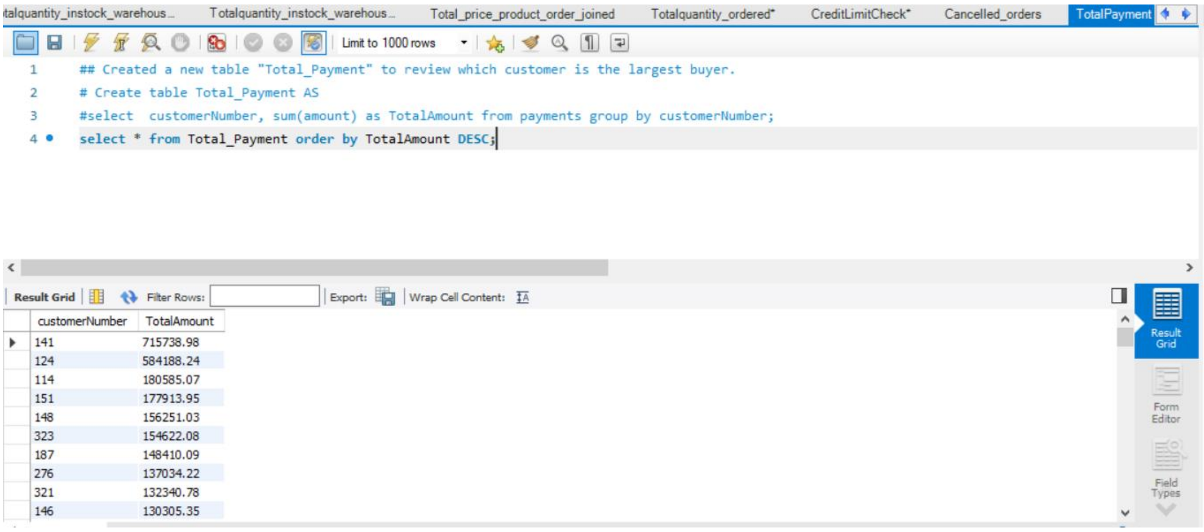
```
##Analyzong cancelled orders
select * from orders where `status` = "Cancelled";
```

orderNumber	orderDate	requiredDate	shippedDate	status	comments	customerNumbe
10167	2003-10-23	2003-10-30	NULL	Cancelled	Customer called to cancel. The warehouse was notified in time and the order didn't ship. They have a new VP...	448
10179	2003-11-11	2003-11-17	2003-11-13	Cancelled	Customer cancelled due to urgent budgeting issues. Must be cautious when dealing with them in the future. ...	496
10248	2004-05-07	2004-05-14	NULL	Cancelled	Order was mistakenly placed. The warehouse noticed the lack of documentation.	131
10253	2004-06-01	2004-06-09	2004-06-02	Cancelled	Customer disputed the order and we agreed to cancel it. We must be more cautions with this customer going...	201
10260	2004-06-16	2004-06-22	NULL	Cancelled	Customer heard complaints from their customers and called to cancel this order. Will notify the Sales Manager.	357
10262	2004-06-24	2004-07-01	NULL	Cancelled	This customer found a better offer from one of our competitors. Will call back to renegotiate.	141
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fig. (f)

- <u>Step6:</u>

  A new table “Total_Payment” was created to see the highest buyer (as shown in figure-g).



```sql
## Created a new table "Total_Payment" to review which customer is the largest buyer.
# Create table Total_Payment AS
#select  customerNumber, sum(amount) as TotalAmount from payments group by customerNumber;
select * from Total_Payment order by TotalAmount DESC;
```

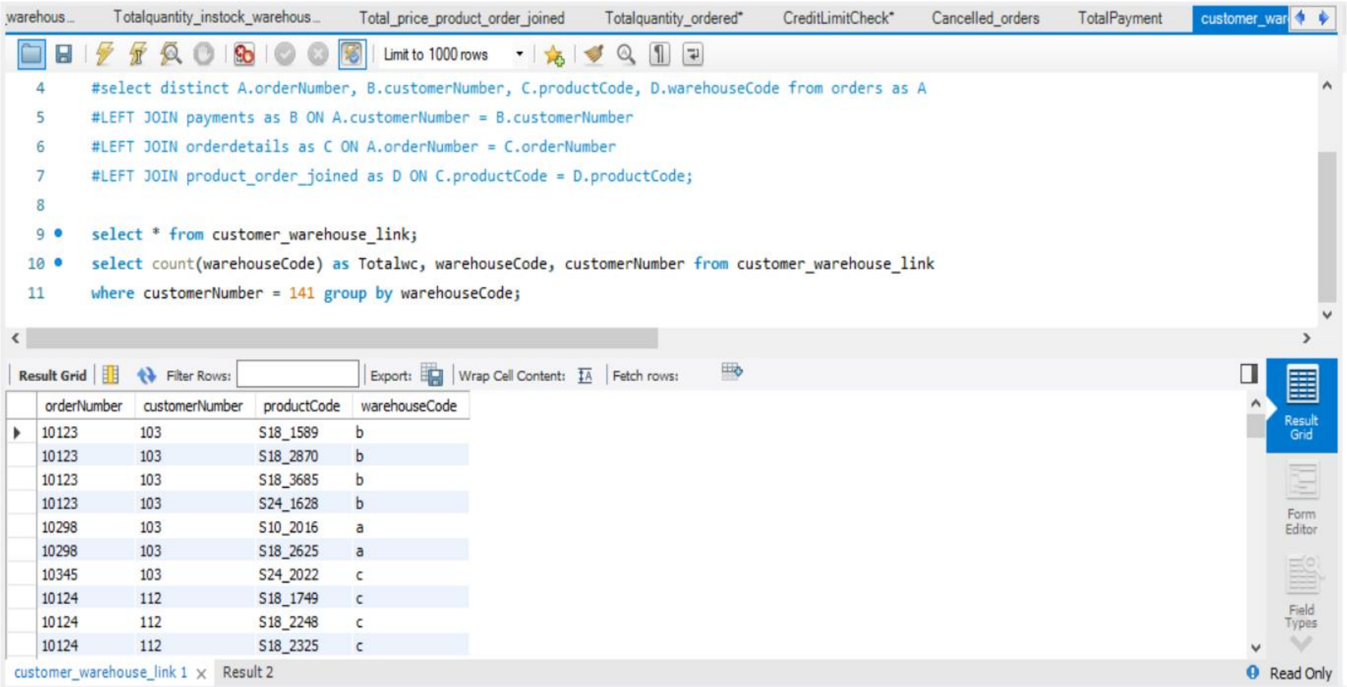
customerNumber	TotalAmount
141	715738.98
124	584188.24
114	180585.07
151	177913.95
148	156251.03
323	154622.08
187	148410.09
276	137034.22
321	132340.78
146	130305.35

Fig. (g)

- <u>Step7:</u>

  Lastly, a match between the highest buyer and warehouse was implemented by combining four entities namely orders, payments, orderdetails and product_order_joined into a new table customer_warehouse_link (as shown in figure-h.1).

  In addition to it, a count of products ordered by highest buyer is analyzed (as shown in fig-h.2)

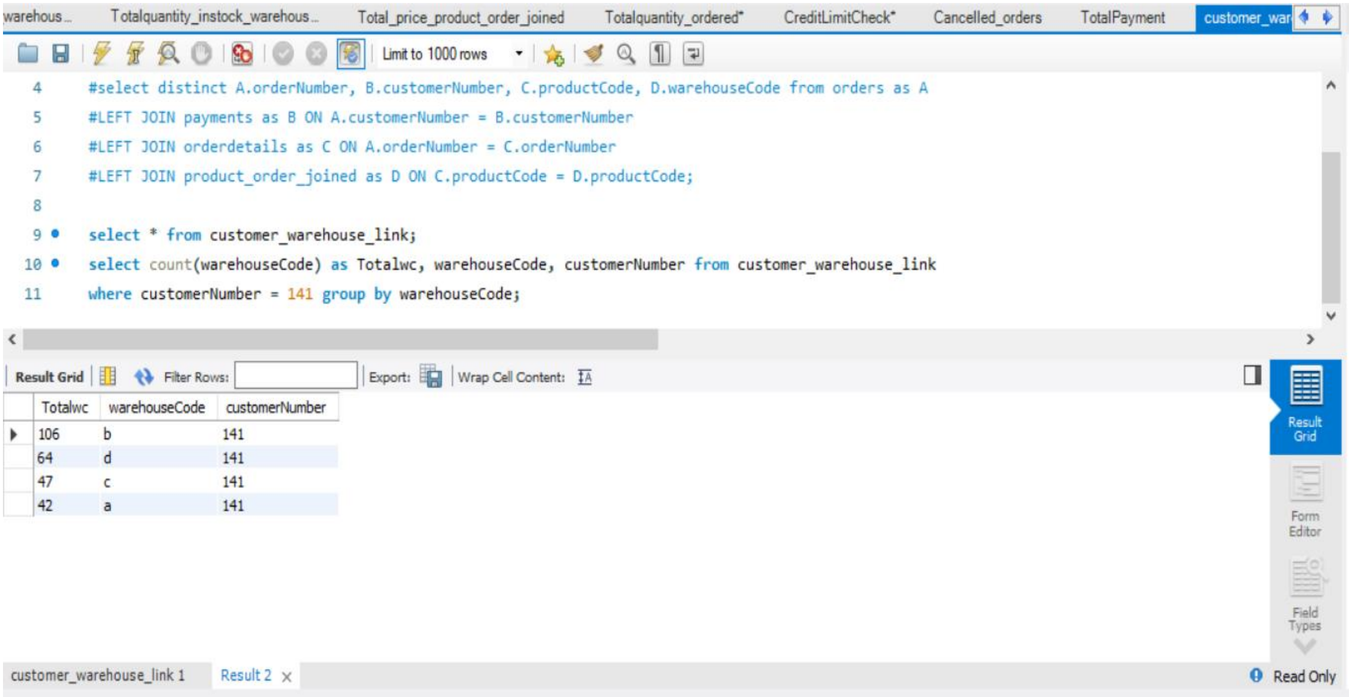


```
#select distinct A.orderNumber, B.customerNumber, C.productCode, D.warehouseCode from orders as A
#LEFT JOIN payments as B ON A.customerNumber = B.customerNumber
#LEFT JOIN orderdetails as C ON A.orderNumber = C.orderNumber
#LEFT JOIN product_order_joined as D ON C.productCode = D.productCode;

select * from customer_warehouse_link;
select count(warehouseCode) as Totalwc, warehouseCode, customerNumber from customer_warehouse_link
where customerNumber = 141 group by warehouseCode;
```

orderNumber	customerNumber	productCode	warehouseCode
10123	103	S18_1589	b
10123	103	S18_2870	b
10123	103	S18_3685	b
10123	103	S24_1628	b
10298	103	S10_2016	a
10298	103	S18_2625	a
10345	103	S24_2022	c
10124	112	S18_1749	c
10124	112	S18_2248	c
10124	112	S18_2325	c

Fig. (h.1)



```
#select distinct A.orderNumber, B.customerNumber, C.productCode, D.warehouseCode from orders as A
#LEFT JOIN payments as B ON A.customerNumber = B.customerNumber
#LEFT JOIN orderdetails as C ON A.orderNumber = C.orderNumber
#LEFT JOIN product_order_joined as D ON C.productCode = D.productCode;

select * from customer_warehouse_link;
select count(warehouseCode) as Totalwc, warehouseCode, customerNumber from customer_warehouse_link
where customerNumber = 141 group by warehouseCode;
```

Totalwc	warehouseCode	customerNumber
106	b	141
64	d	141
47	c	141
42	a	141

Fig. (h.2)

Results and Analysis:

Sno.	Question	Maximum	Minimum	Insight
1.	Product quantity	Warehouse b (219K)	Warehouse d (79K)	Warehouse "d" seems to be most convenient to dispatch all products.
2.	Total sales/orders by customers	Warehouse b (3.7M)	Warehouse c (1.7M) Warehouse d (1.8M)	Warehouse "b" is the most profitable.
3.	Highest buyer w/ warehouse	Customer no. 141 Warehouse b (106)	- Warehouse a (42) Warehouse c (47) Warehouse d (64)	Warehouse "d" still seems to be profitable as well because of the highest buyer.
4.	Product cancellation issues	-	-	No (all issues were resolved)
5.	Employment transferring	-	-	No correlation is seen among employees and other tables.

- Overall, warehouse "d" has the minimum amount of quantity as well as has the second most products sold to the highest purchaser.
- If, it is to be decided to transfer the goods to another warehouse, warehouses "c" and "a" can accommodate the remaining items till full sell out.
- If, it is to be decided to sell all the goods rather than accommodating, a slight decrement in price may not affect the overall profit earned by warehouse "d" and may assist to sell everything till full termination.

Feature of approach:

- Explainability of queries with each approach.
- Filtered data and new creations as per needed.
- Edit, delete, and mark tasks as complete, thus adequate time management.

Conclusion:

Optimum results with insights are were earned solving the objectives of the project.

Challenges and Learnings:

- Lack of correlation among few entities and features.
- Learnt importance of naming conventions.
- Learnt how imperative is to have proper management.

References:

https://www.coursera.org/projects/showcase-analyze-data-model-car-database-mysql-workbench