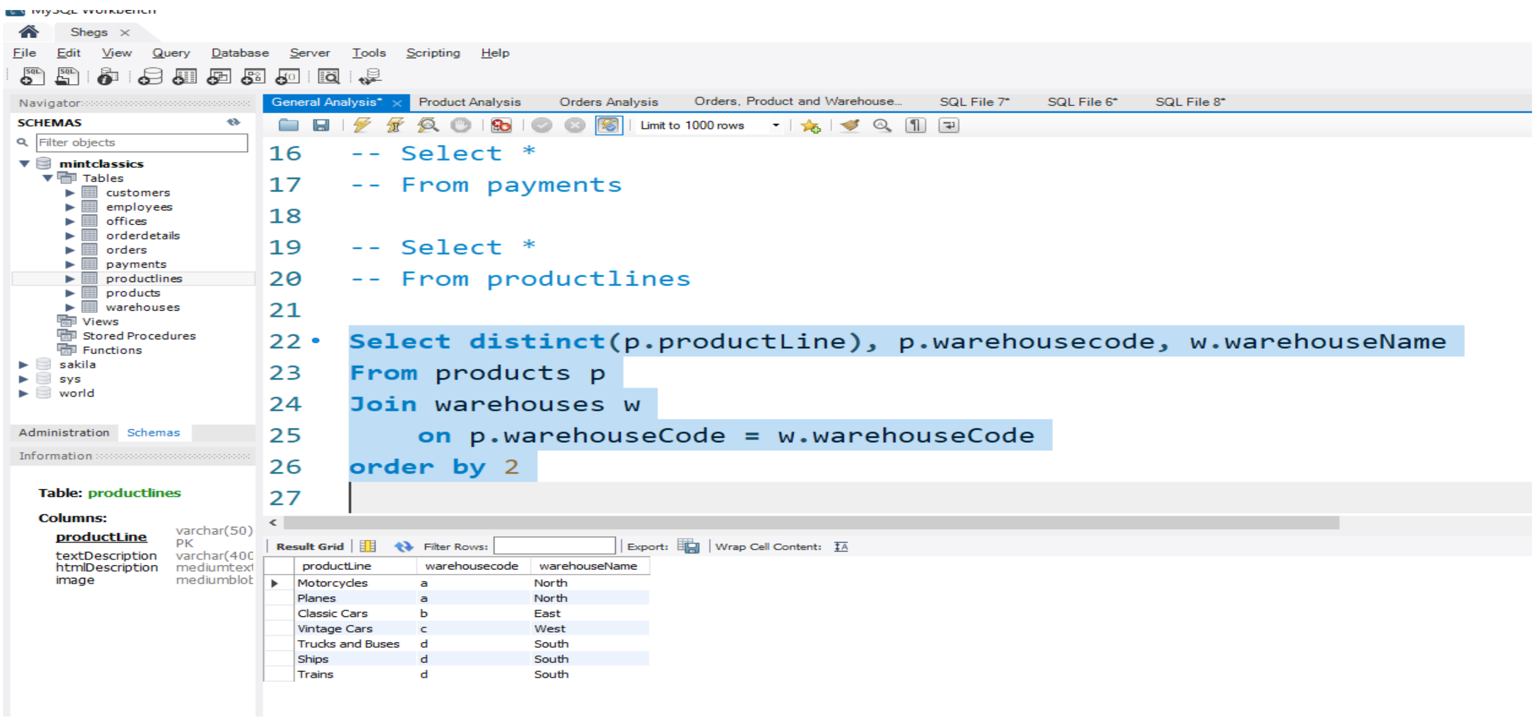
Communicating My Analysis on The Mint Classics Databases

First and foremost, I will like to look at the warehouses on Mint Classics Databases. Now let's look at the below query and result:



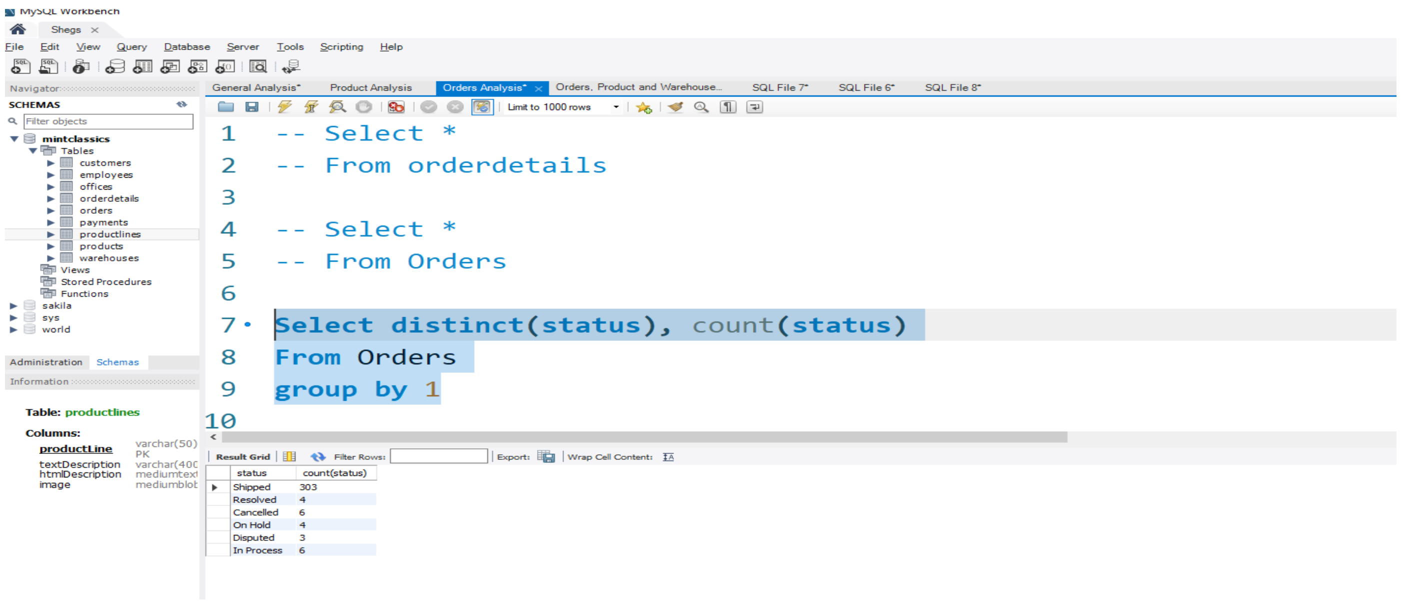
```sql
-- Select *
-- From payments

-- Select *
-- From productlines

Select distinct(p.productLine), p.warehousecode, w.warehouseName
From products p
Join warehouses w
    on p.warehouseCode = w.warehouseCode
order by 2
```

productLine	warehousecode	warehouseName
Motorcycles	a	North
Planes	a	North
Classic Cars	b	East
Vintage Cars	c	West
Trucks and Buses	d	South
Ships	d	South
Trains	d	South

Fig 1:sql query and result showing the different product line, warehouse code and warehouse name

From fig 1, we can see that we have four warehouses with warehouse names, North, East, West and South with code as 'a', 'b', 'c' and 'd' respectively. Also, warehouse with code 'a' have motorcycles and planes as the product line stored in with while that of 'b' have Classic cars, that of 'c', have vintage cars and finally 'd' has Ships, Trains, Trucks and Buses stored in it.

Having, understand the specifics of each warehouse, we now looked at the different quantity of stock (product line) ordered that have been shipped out of these warehouses.

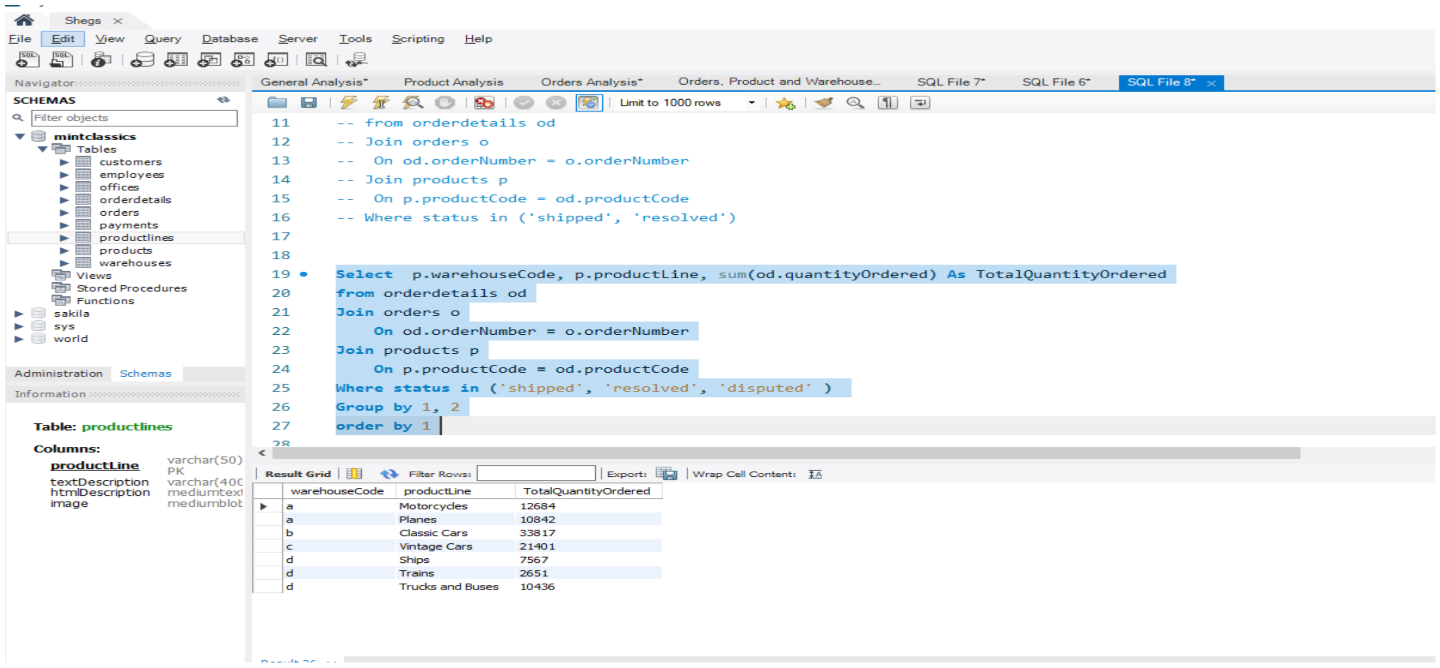


```sql
-- Select *
-- From orderdetails

-- Select *
-- From Orders

Select distinct(status), count(status)
From Orders
group by 1
```

status	count(status)
Shipped	303
Resolved	4
Cancelled	6
On Hold	4
Disputed	3
In Process	6

Fig 2:sql query and result showing the different status on all orders

Looking at fig2; we can see the different status and their counts of orders made on Mint classic. Our focus will be on those that have been shipped out of the warehouses. The ‘shipped’, ‘resolved’ and ‘disputed’ status are what we will be looking at as this status indicate that stocks have been shipped off the warehouse. Though 2 sets of stocks in the cancelled status were shipped but in view of the reasons, they will be brought back to the warehouse.


Shegs
File Edit View Query Database Server Tools Scripting Help
Navigator
SCHEMAS
mintclassics
Tables
customers
employees
offices
orderdetails
orders
payments
productlines
products
warehouses
Views
Stored Procedures
Functions
sakila
sys
world
Administration Schemas
Information
General Analysis* Product Analysis Orders Analysis* Orders, Product and Warehouse... SQL File 7* SQL File 6* SQL File 8*
Limit to 1000 rows
Result Grid Filter Rows: Export: Wrap Cell Content:

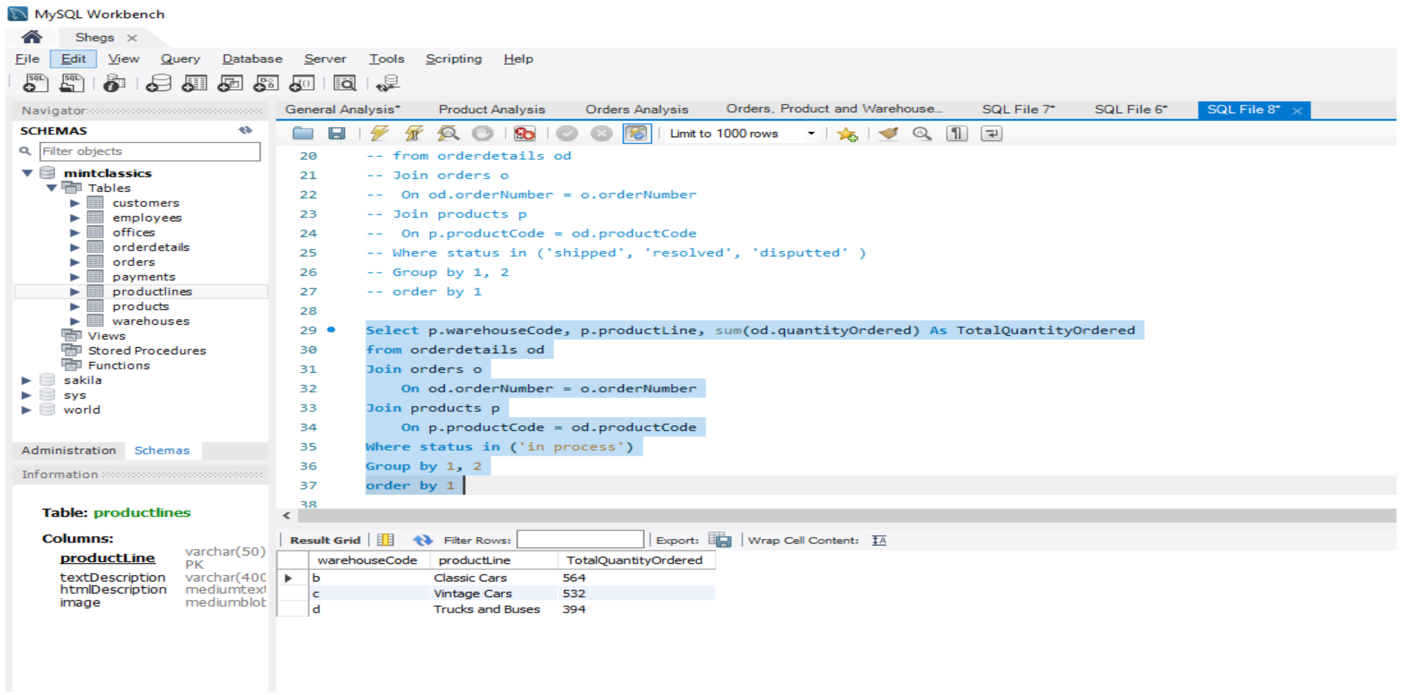
```sql
-- from orderdetails od
-- Join orders o
--  On od.orderNumber = o.orderNumber
-- Join products p
--  On p.productCode = od.productCode
-- Where status in ('shipped', 'resolved')


Select  p.warehouseCode, p.productLine, sum(od.quantityOrdered) As TotalQuantityOrdered
from orderdetails od
Join orders o
    On od.orderNumber = o.orderNumber
Join products p
    On p.productCode = od.productCode
Where status in ('shipped', 'resolved', 'disputed' )
Group by 1, 2
order by 1
```

warehouseCode	productLine	TotalQuantityOrdered
a	Motorcycles	12684
a	Planes	10842
b	Classic Cars	33817
c	Vintage Cars	21401
d	Ships	7567
d	Trains	2651
d	Trucks and Buses	10436

Table: productlines
Columns:
productLine varchar(50) PK
textDescription varchar(400
htmlDescription mediumtext
image mediumblob

Fig 3: sql query and result on ordered product having status of ‘shipped’, ‘resolved’ and ‘disputed’.

From fig 3, we can see the total quantity of stock ordered and shipped with it’s different product lines. Now I would love to include the quantity of product ordered it status report ‘in process’ as this product would be shipped out of their warehouses in no distant time.



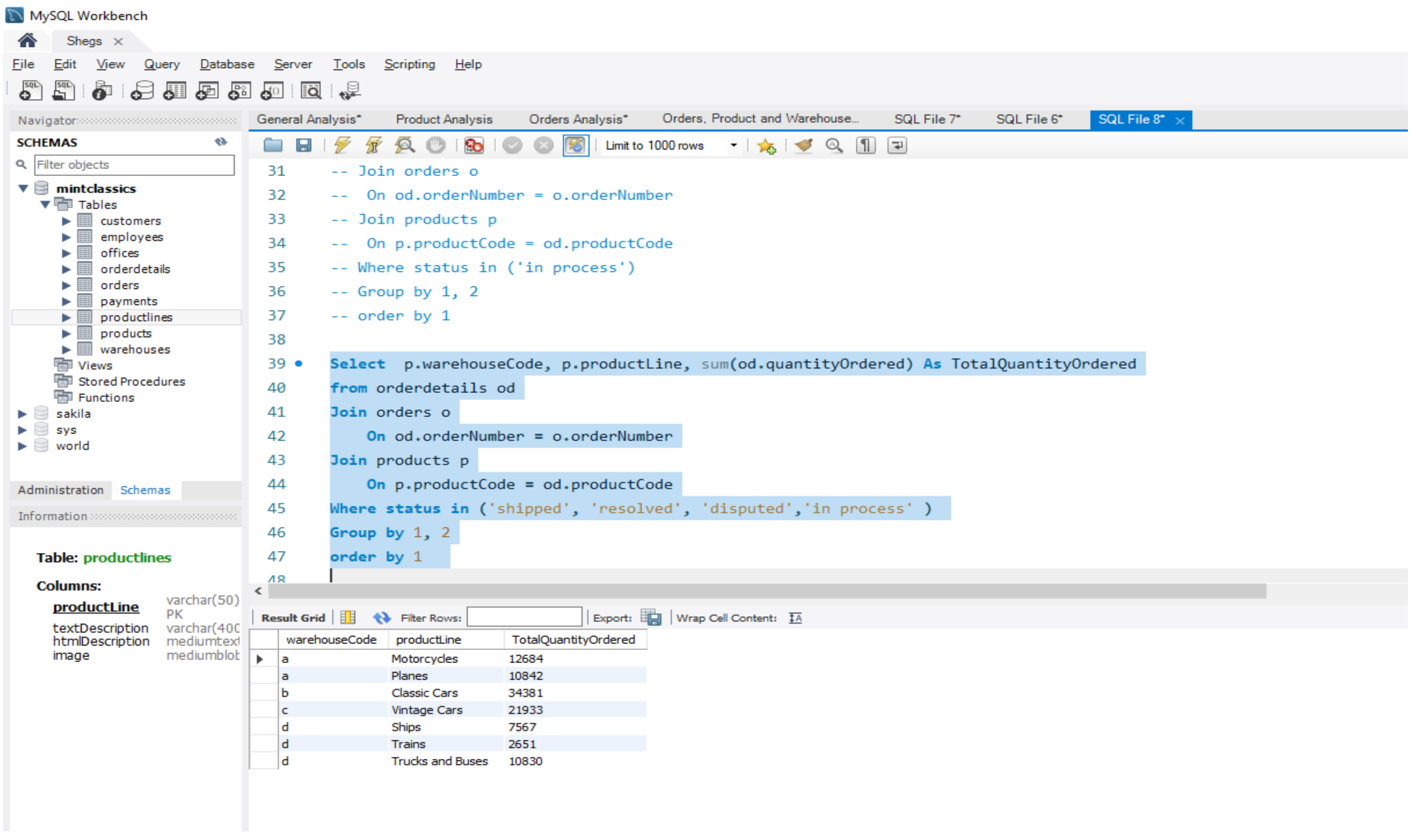
```sql
-- from orderdetails od
-- Join orders o
--  On od.orderNumber = o.orderNumber
-- Join products p
--  On p.productCode = od.productCode
-- Where status in ('shipped', 'resolved', 'disputted' )
-- Group by 1, 2
-- order by 1

Select p.warehouseCode, p.productLine, sum(od.quantityOrdered) As TotalQuantityOrdered
from orderdetails od
Join orders o
    On od.orderNumber = o.orderNumber
Join products p
    On p.productCode = od.productCode
Where status in ('in process')
Group by 1, 2
order by 1
```

warehouseCode	productLine	TotalQuantityOrdered
b	Classic Cars	564
c	Vintage Cars	532
d	Trucks and Buses	394

Table: productlines
Columns:
productLine varchar(50) PK
textDescription varchar(400
htmlDescription mediumtext
image mediumblob

Fig 4: sql query and result of product ordered with status report ‘in process’.

We can see from fig 4 that only the product line classic cars, vintage cars and truck and buses are with the status ‘in process’, for this would soon leave its respectively warehouse. So, I want to include them on total quantity of product that have been shipped.



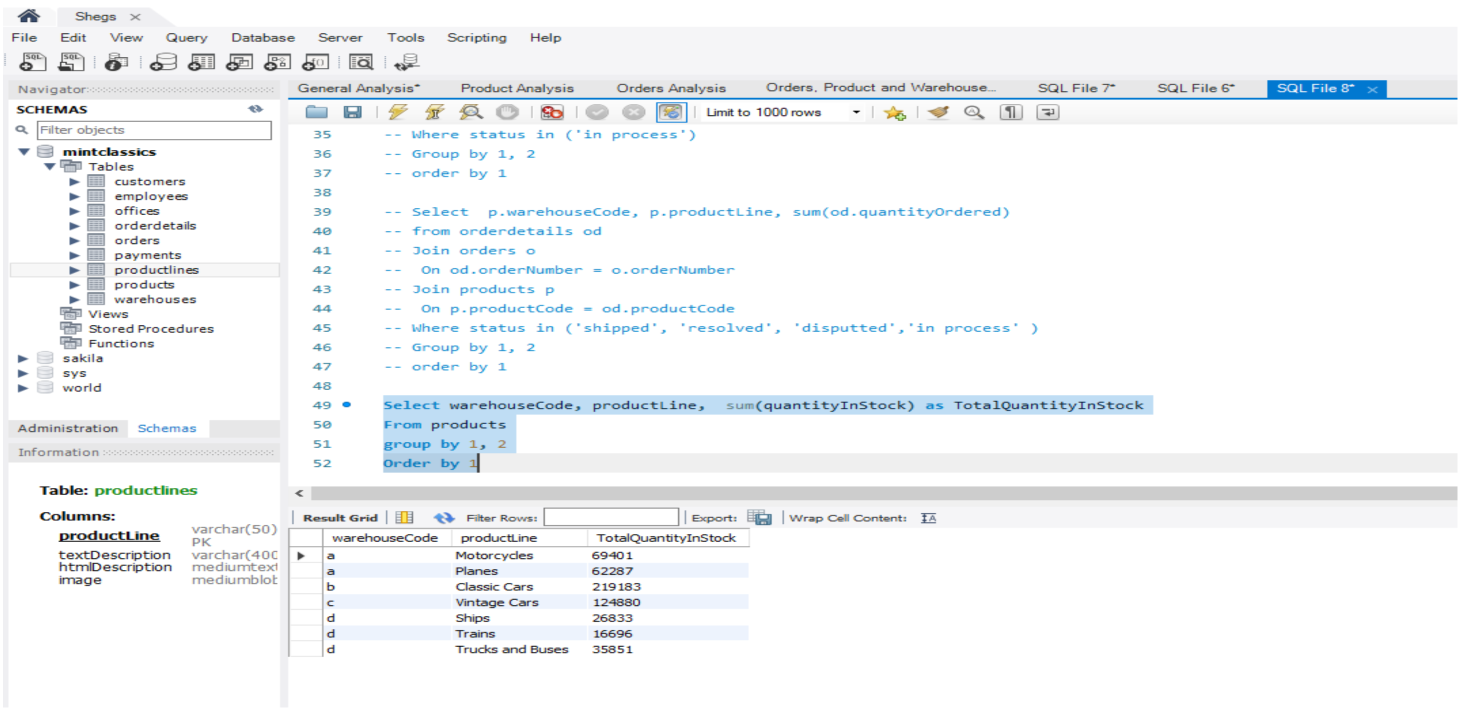
```
-- Join orders o
--  On od.orderNumber = o.orderNumber
-- Join products p
--  On p.productCode = od.productCode
-- Where status in ('in process')
-- Group by 1, 2
-- order by 1

Select  p.warehouseCode, p.productLine, sum(od.quantityOrdered) As TotalQuantityOrdered
from orderdetails od
Join orders o
    On od.orderNumber = o.orderNumber
Join products p
    On p.productCode = od.productCode
Where status in ('shipped', 'resolved', 'disputed','in process' )
Group by 1, 2
order by 1
```

warehouseCode	productLine	TotalQuantityOrdered
a	Motorcycles	12684
a	Planes	10842
b	Classic Cars	34381
c	Vintage Cars	21933
d	Ships	7567
d	Trains	2651
d	Trucks and Buses	10830

Fig 5: sql query and result on all ordered products with status ‘shipped’, ‘resolved’, ‘disputed’ and ‘in process’.

Now we can clearly see the total quantity that have been ordered and shipped and will soon be shipped out of the warehouse as we spoken about earlier.

Let’s pause a bit and see the total quantity in stock initially in the warehouse.

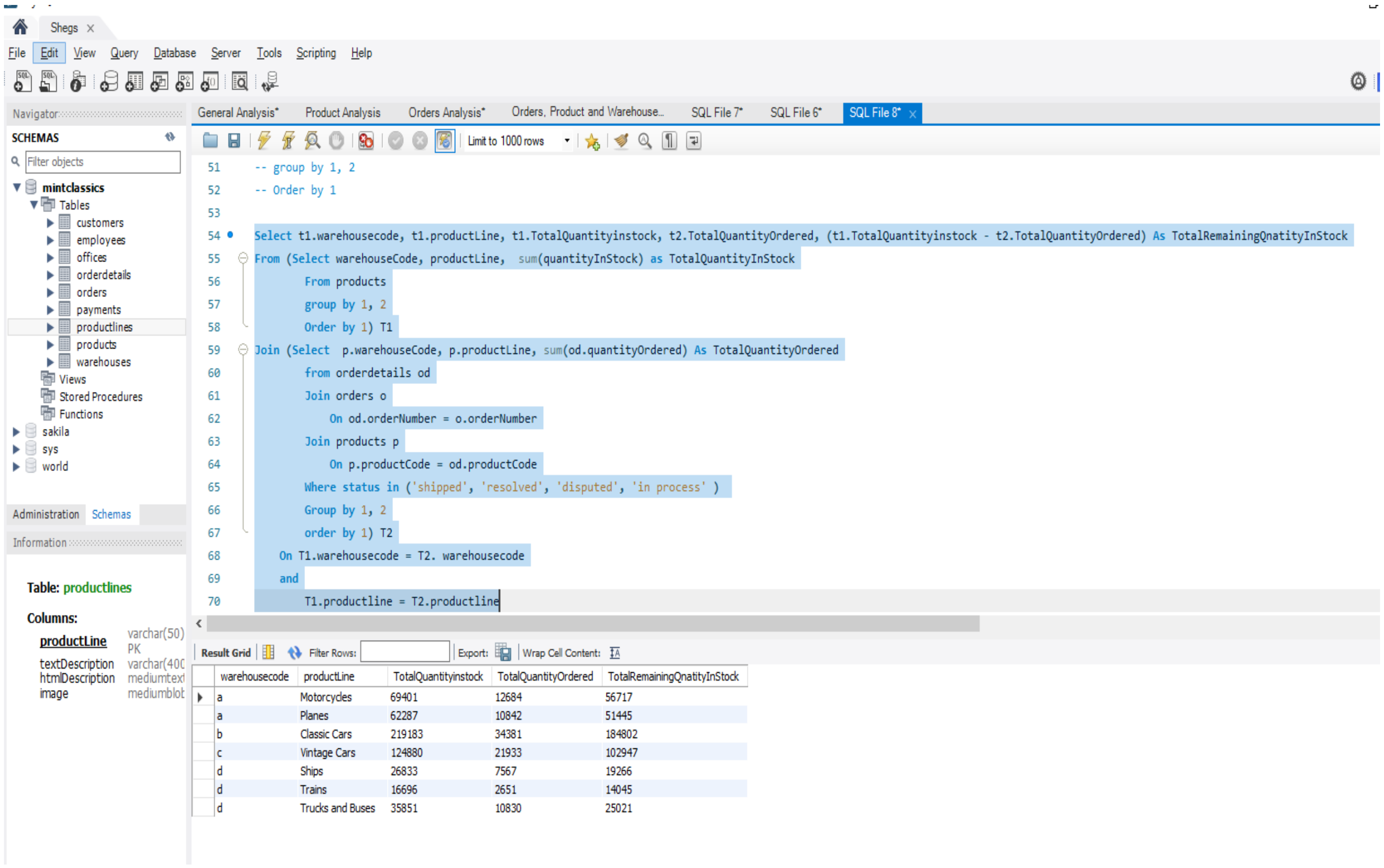


```
-- Where status in ('in process')
-- Group by 1, 2
-- order by 1

-- Select  p.warehouseCode, p.productLine, sum(od.quantityOrdered)
-- from orderdetails od
-- Join orders o
--  On od.orderNumber = o.orderNumber
-- Join products p
--  On p.productCode = od.productCode
-- Where status in ('shipped', 'resolved', 'disputted','in process' )
-- Group by 1, 2
-- order by 1

Select warehouseCode, productLine,  sum(quantityInStock) as TotalQuantityInStock
From products
group by 1, 2
Order by 1
```

warehouseCode	productLine	TotalQuantityInStock
a	Motorcycles	69401
a	Planes	62287
b	Classic Cars	219183
c	Vintage Cars	124880
d	Ships	26833
d	Trains	16696
d	Trucks and Buses	35851

Fig 6: sql query and result of all product line and their total quantity in stock in their respective warehouses.

We can analyze fig 5 and fig 6 so as to get the total number of quantities of each product line remaining in each warehouse.



```sql
-- group by 1, 2
-- Order by 1

Select t1.warehousecode, t1.productLine, t1.TotalQuantityinstock, t2.TotalQuantityOrdered, (t1.TotalQuantityinstock - t2.TotalQuantityOrdered) As TotalRemainingQnatityInStock
From (Select warehouseCode, productLine,  sum(quantityInStock) as TotalQuantityInStock
        From products
        group by 1, 2
        Order by 1) T1
Join (Select  p.warehouseCode, p.productLine, sum(od.quantityOrdered) As TotalQuantityOrdered
        from orderdetails od
        Join orders o
            On od.orderNumber = o.orderNumber
        Join products p
            On p.productCode = od.productCode
        Where status in ('shipped', 'resolved', 'disputed', 'in process' )
        Group by 1, 2
        order by 1) T2
    On T1.warehousecode = T2. warehousecode
    and
        T1.productline = T2.productline
```

```
warehousecode  productLine      TotalQuantityinstock  TotalQuantityOrdered  TotalRemainingQnatityInStock
a              Motorcycles      69401                 12684                 56717
a              Planes           62287                 10842                 51445
b              Classic Cars     219183                34381                 184802
c              Vintage Cars     124880                21933                 102947
d              Ships            26833                 7567                  19266
d              Trains           16696                 2651                  14045
d              Trucks and Buses 35851                 10830                 25021
```

Fig 7:sql query and result showing the remaining quantity of each product line in each warehouse.

Now, let's talk about the fig 7. We can see;

1. Total of **12,684 motorcycles** was shipped out of the warehouse with code 'a' leaving **56,717 motorcycles**
2. A total of **10,842 planes** shipped leaving **51,445 planes** remaining in same warehouse as that of the motorcycles.
3. A grand total of **108,162** products remaining in warehouse '**a'**
4. A total of **34, 381 classic cars** was shipped out of warehouse 'b' and this leaves a total of **184,802 classic cars** remaining in warehouse **'b'.**
5. A total of **21, 933 vintage cars** was shipped out of warehouse 'c' leaving a total of **102,947 vintage cars** remaining in warehouse **'c'.**
6. A total of **7,567 ships** was shipped out of warehouse 'd' leaving a total of **19,288 ships** remaining
7. A total of **2,651 trains** was shipped out of warehouse 'd' leaving a total of **14,045 trains** remaining.
8. A total of **10,830 trucks and buses** was shipped out of the warehouse ''d' leaving a total of **25,021 trucks and buses** remaining.
9. A grand total of **58,354** of products are remaining in warehouse **'d'**

Projection /Solution

1. In view of the about numbers, we can see that the sum of both the quantities of ships and that of train give rise to **33,333** quantities and this quantity is not up to the total quantity that left the warehouse ‘b’ since only **34,381** classic cars was shipped out. So, we can reorganize warehouse ‘b’ to contain the vintage cars, ships and train as this won’t affect its capacity.
2. Also, we just have a difference of **3,088** quantities as regard the number of trucks and buses left in warehouse ‘d’ and that of the total quantities of vintage cars that left warehouse ‘c’. So, we can combine both the vintage car product line and that of the truck and buses product line in warehouse ‘c’
3. Warehouse a still retains it usual product line which is motorcycles and planes product line

In conclusion we see that we have successful eliminated warehouse ‘d’ as its formal products have been shifted to warehouse ‘b’ and ‘c’. Therefore, we have

Product line	**warehouse Code**
• **Motorcycles**	**a**
• **Planes**	**a**
• **Classic cars**	**b**
• **Ships**	**b**
• **Trains**	**b**
• **Vintage Cars**	**c**
• **Trucks and Buses**	**c**

Benefits:

1. Optimized Warehouse Utilization: By consolidating product lines, Mint Classics can maximize the utilization of their warehouses, leading to better space management and cost savings.

2. Streamlined Operations: The reduction in the number of warehouses simplifies logistics and inventory management processes, improving overall operational efficiency.

3. Cost Savings: Operating fewer warehouses reduces overhead costs associated with maintenance, utilities, and staffing.

4. Minimal Disruption: The proposed relocation ensures that the normal sales cycle remains unaffected, maintaining timely service to customers.

Conclusion:

In conclusion, the analysis highlights a viable solution for Mint Classics Company to optimize warehouse utilization by consolidating product lines across warehouses. This strategic relocation enables the company to reduce the number of warehouses from four to three without compromising operational efficiency or increasing costs. By implementing this solution, Mint Classics can achieve greater efficiency in inventory management while maintaining high standards of customer service.