Data Exploration with MySQL Workbench.

Date: March 27, 2024

Project completed by: Anton Starshev
http://linkedin.com/in/starshev

Project Source

Coursera Project Network · Analyze Data in a Model Car Database with MySQL Workbench

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

Brief Description

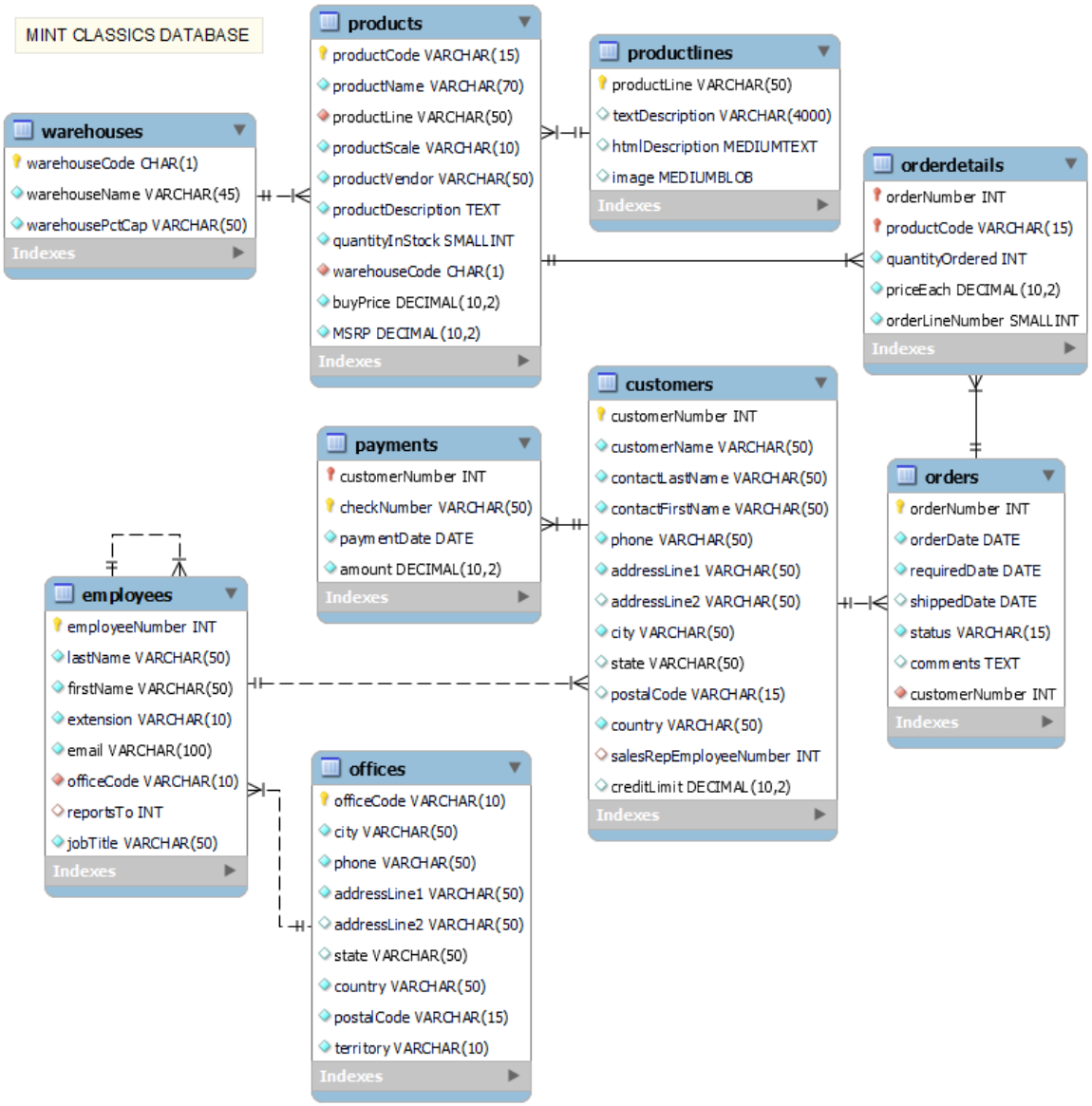
Within the scope of this project, which takes approximately 8-10 hours to complete, an analysis of beginner-level data is proposed in the fictional company *Mint Classics*. The aim is to support inventory-related business decisions that will lead to the closure of one of the warehouse locations.

Project Scenario.

Mint Classics, a retailer specializing in classic automobile and other transportation vehicle models, is considering the possibility of closing one of its warehouse locations. To aid decision-making based on data, the company is seeking proposals for inventory reorganization or reduction. It is proposed to utilize MySQL Workbench for data analytics to familiarize with the company's business by examining existing data.

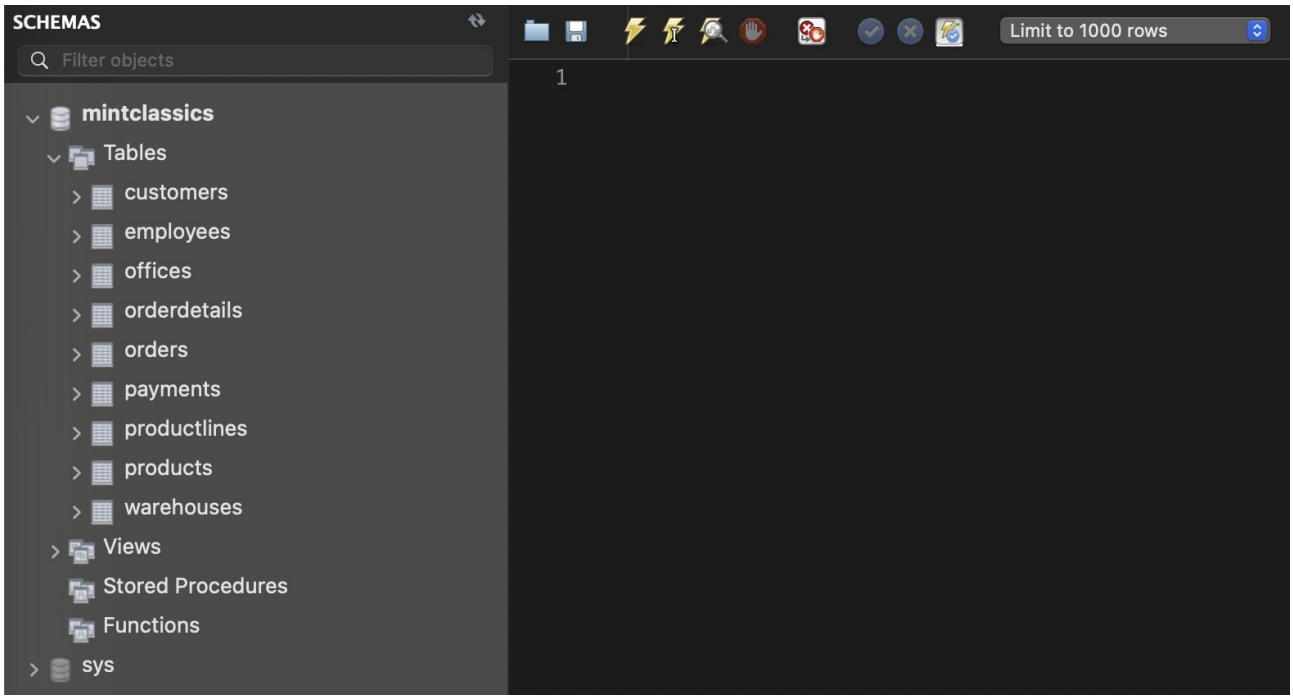
The goal of the project is to conduct exploratory data analysis to identify any patterns and observations that may influence inventory reduction or reorganization at *Mint Classics* warehouses. Additionally, the project aims to provide analytical insights and recommendations based on the data.

A database import script is provided along with the following architectural model:



Database Import.

Upon starting the project, I imported the provided database script as a self-contained file (containing table structure and data) using the Data Import wizard in MySQL Workbench platform.



Familiarization with the Database Model.

After reviewing the database architecture, I identified the tables to be used in my analysis, along with their fields and relationships:

- **orders**: with the primary key *orderNumber* and foreign key *customerNumber* for linking to the *customers* table
- **orderdetails**: with a composite primary key consisting of *orderNumber* and *productCode*, where *productCode* also serves as a foreign key for linking to the *products* table
- **products**: with the primary key *productCode* and a foreign key *warehouseCode* for linking to the *warehouses* table
- **warehouses**: with the primary key *warehouseCode*
- **customers**: with the primary key *customerNumber*

Exploratory Data Analysis.

Commencing the actual research, I formulated several practical metrics (questions, factors, hypotheses) based on the available data.

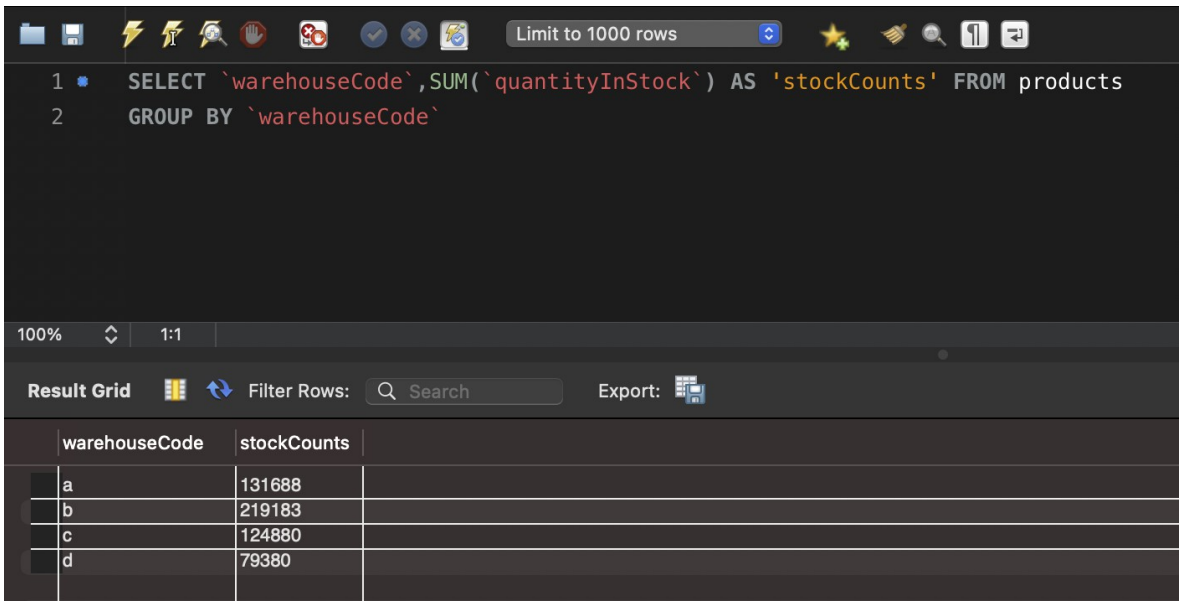
This resulted in a list of 8 target areas (aspects) in the warehousing and product management activities of *Mint Classics*, the analysis of which could shed light on the current situation, serve as a basis for inventory reduction (reorganization) recommendations, and assist in deliberating the idea of closing one of the warehouses, thus achieving the stated project goals:

1. Inventory Levels
2. Hypothesis on Order Portfolio
3. Efficiency
4. Customer Orientation
5. Promptness
6. Growth Patterns
7. Inventory Turnover
8. Geography

Next, I propose the grounding of each of the identified target areas and provide a detailed description of the conducted research, supported by illustrations of executed SQL queries, descriptions of applied techniques, and intermediate analytical observations.

1 · Inventory Levels

Comparing the "ranking" of warehouses based on the volume of inventory could help identify the primary candidate for closure if one warehouse has significantly less inventory than others. I executed a query to determine the quantity of remaining goods (in terms of units) for each warehouse.



Observation: warehouse "d" leads in terms of the minimum number of remaining items with 79,380 units, which is nearly half of the next position in the ranking. The highest number of remaining items is on Warehouse "b" with 219,183 units.

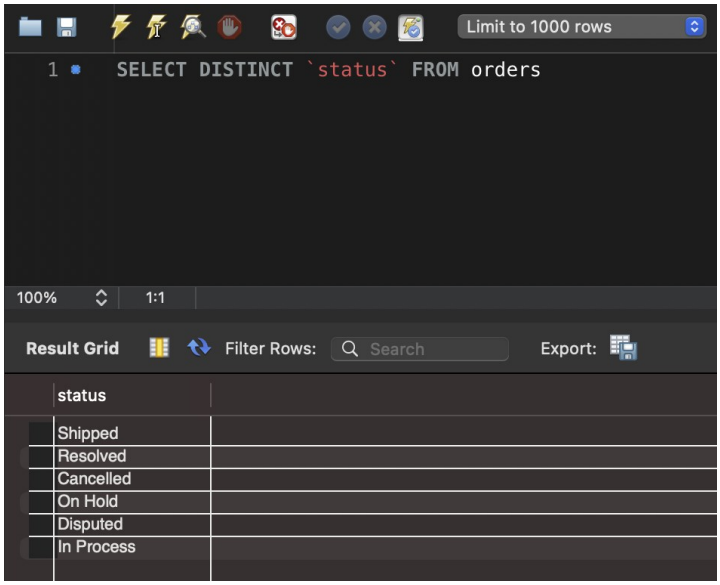
2 · Hypothesis on Order Portfolio

I formulated a hypothesis (assumption):

"The portfolio of open orders is such that if we calculate the volume of remaining inventory for each warehouse (in terms of units) at the time of fulfilling all current orders, it may be found that one specific warehouse has so few remaining items that closing it would be the easiest and most cost-effective option".

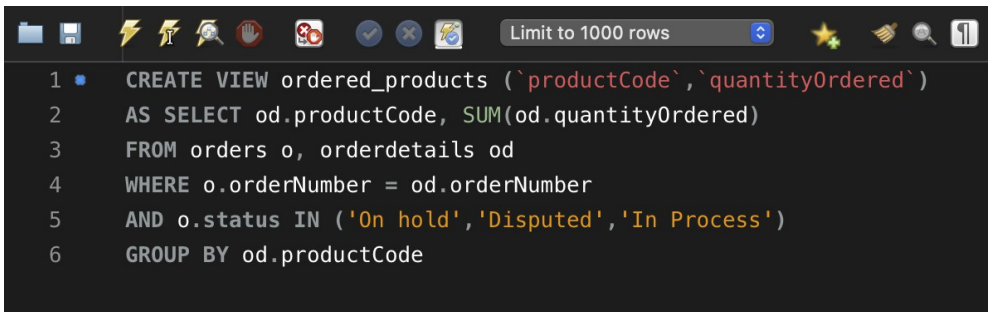
To test the hypothesis, I took the following steps:

a. Identified the different statuses of open orders in the system.



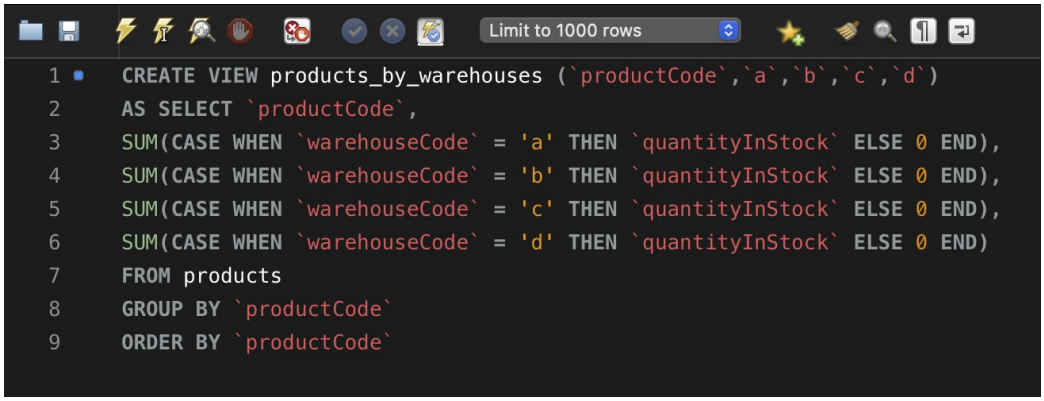
Observation: there are three such statuses: "On hold," "Disputed," and "In process."

b. Created a view containing a list of items and the quantities of each item ordered from all currently open orders.



productCode	quantityOrdered
S10_4962	64
S18_2319	84
S18_2432	53
S18_3232	48
S18_4600	87
S24_2300	91
S18_2581	42
S24_1785	38
S24_3949	64
S24_4278	52
S32_1374	49
S32_4289	62
S50_1341	56
S700_1691	11
S700_2466	85
S700_2834	21
S700_3167	77
S700_4002	40
S18_1129	61
S18_1984	48
S18_3685	65
S18_1589	59
S18_1749	113
S18_2248	78
S18_2870	41
S18_4409	72
S18_4933	102
S24_1046	86
S24_1628	101
S24_2766	115
S24_2887	114
S24_3191	48

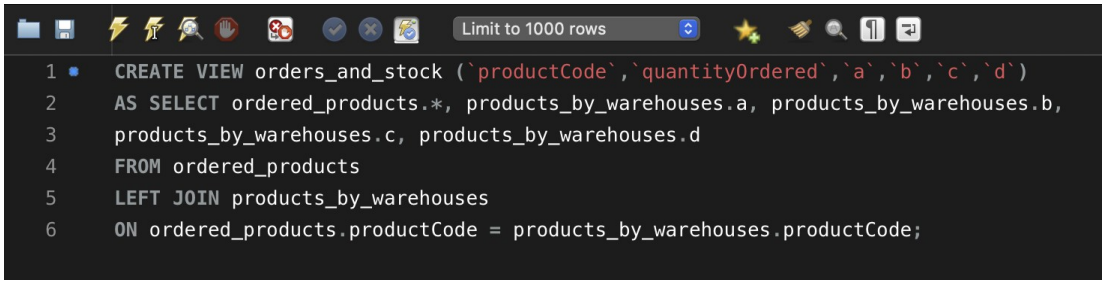
c. Created a view with the distribution of current inventory levels across all warehouses in a pivot table format.



productCode	a	b	c	d
S10_1678	7933	0	0	0
S10_1949	0	7305	0	0
S10_2016	6625	0	0	0
S10_4698	5582	0	0	0
S10_4757	0	3252	0	0
S10_4962	0	6791	0	0
S12_1099	0	68	0	0
S12_1108	0	3619	0	0
S12_1666	0	0	0	1579
S12_2823	9997	0	0	0
S12_3148	0	6906	0	0
S12_3380	0	9123	0	0
S12_3891	0	1049	0	0
S12_3990	0	5663	0	0
S12_4473	0	0	0	6125
S12_4675	0	7323	0	0
S18_1097	0	0	0	2613
S18_1129	0	3975	0	0
S18_1342	0	0	8693	0
S18_1367	0	0	8635	0
S18_1589	0	9042	0	0
S18_1662	5330	0	0	0
S18_1749	0	0	2724	0
S18_1889	0	8826	0	0
S18_1984	0	9772	0	0
S18_2238	0	4724	0	0
S18_2248	0	0	540	0
S18_2319	0	0	0	8258
S18_2325	0	0	9354	0
S18_2432	0	0	0	2018
S18_2581	992	0	0	0
S18_2625	4357	0	0	0
S18_2795	0	0	548	0
S18_2870	0	8164	0	0
S18_2949	0	0	4189	0

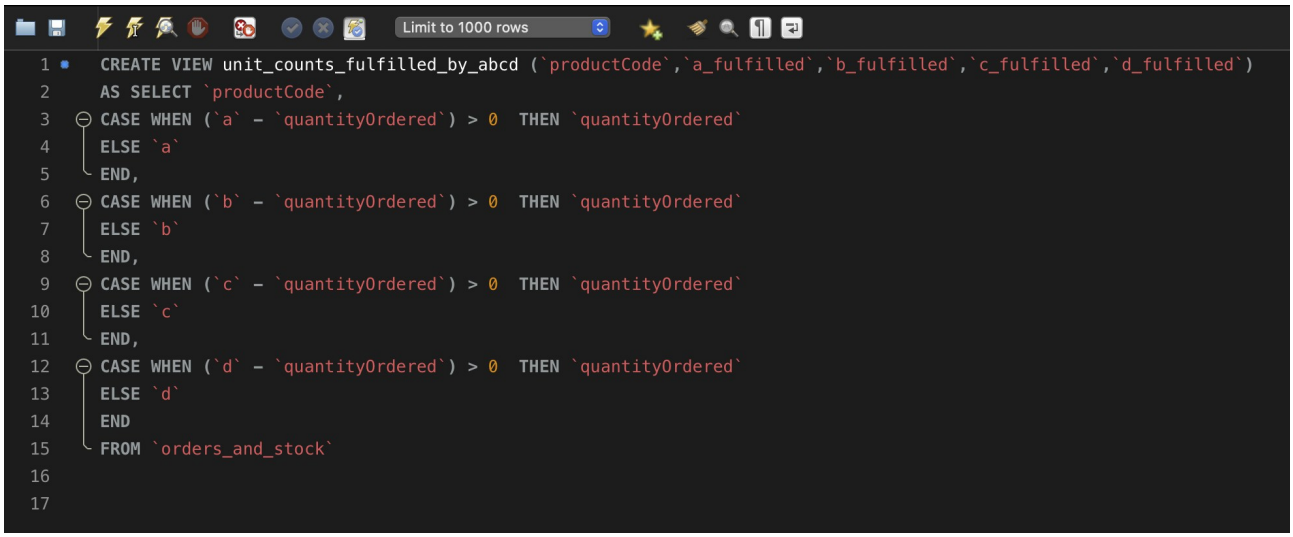
Observation: at this stage, an interesting fact emerged that each individual item is exclusively stored in one of the warehouses.

d. Merged two (previously created) views for convenient viewing of inventory distribution across warehouses and all ordered items within a single table. Additionally, the inventory now only displays those items that are present in open orders.



productCode	quantityOrdered	a	b	c	d
S10_4962	64	0	6791	0	0
S18_2319	84	0	0	0	8258
S18_2432	53	0	0	0	2018
S18_3232	48	0	8347	0	0
S18_4600	87	0	0	0	3128
S24_2300	91	0	0	0	2327
S18_2581	42	992	0	0	0
S24_1785	38	3627	0	0	0
S24_3949	64	6812	0	0	0
S24_4278	52	2756	0	0	0
S32_1374	49	178	0	0	0
S32_4289	62	0	0	136	0
S50_1341	56	0	0	7062	0
S700_1691	11	5841	0	0	0
S700_2466	85	9653	0	0	0
S700_2834	21	7106	0	0	0
S700_3167	77	551	0	0	0
S700_4002	40	8820	0	0	0
S18_1129	61	0	3975	0	0
S18_1984	48	0	9772	0	0
S18_3685	65	0	8990	0	0
S18_1589	59	0	9042	0	0
S18_1749	113	0	0	2724	0
S18_2248	78	0	0	540	0
S18_2870	41	0	8164	0	0
S18_4409	72	0	0	6553	0
S18_4933	102	0	3209	0	0
S24_1046	86	0	1005	0	0
S24_1628	101	0	8197	0	0
S24_2766	115	0	2350	0	0
S24_2887	114	0	1452	0	0
S24_3191	48	0	4695	0	0
S24_3432	69	0	9446	0	0
S10_4757	49	0	3252	0	0
S18_3029	44	0	0	0	4259

e. Created another view to illustrate how many and what types of product items from all open orders are located at each warehouse.



productCode	a_fulfilled	b_fulfilled	c_fulfilled	d_fulfilled
S10_4962	0	64	0	0
S18_2319	0	0	0	84
S18_2432	0	0	0	53
S18_3232	0	48	0	0
S18_4600	0	0	0	87
S24_2300	0	0	0	91
S18_2581	42	0	0	0
S24_1785	38	0	0	0
S24_3949	64	0	0	0
S24_4278	52	0	0	0
S32_1374	49	0	0	0
S32_4289	0	0	62	0
S50_1341	0	0	56	0
S700_1691	11	0	0	0
S700_2466	85	0	0	0
S700_2834	21	0	0	0
S700_3167	77	0	0	0
S700_4002	40	0	0	0
S18_1129	0	61	0	0
S18_1984	0	48	0	0
S18_3685	0	65	0	0
S18_1589	0	59	0	0
S18_1749	0	0	113	0
S18_2248	0	0	78	0
S18_2870	0	41	0	0
S18_4409	0	0	72	0
S18_4933	0	102	0	0
S24_1046	0	86	0	0
S24_1628	0	101	0	0
S24_2766	0	115	0	0
S24_2887	0	114	0	0
S24_3191	0	48	0	0
S24_3432	0	69	0	0
S10_4757	0	49	0	0
S18_3029	0	0	0	44

f. Calculated the volume of remaining inventory for each warehouse (in terms of units) at the time of fulfilling all currently open orders.

```sql
SELECT
t2.a - t1.a AS a_final_stock,
t2.b - t1.b AS b_final_stock,
t2.c - t1.c AS c_final_stock,
t2.d - t1.d AS d_final_stock
FROM
(SELECT
SUM(`a_fulfilled`) AS `a`,
SUM(`b_fulfilled`) AS `b`,
SUM(`c_fulfilled`) AS `c`,
SUM(`d_fulfilled`) AS `d`
FROM `unit_counts_fulfilled_by_abcd`) t1,
(SELECT
SUM(CASE WHEN `warehouseCode` = 'a' THEN `quantityInStock` ELSE 0 END) AS `a`,
SUM(CASE WHEN `warehouseCode` = 'b' THEN `quantityInStock` ELSE 0 END) AS `b`,
SUM(CASE WHEN `warehouseCode` = 'c' THEN `quantityInStock` ELSE 0 END) AS `c`,
SUM(CASE WHEN `warehouseCode` = 'd' THEN `quantityInStock` ELSE 0 END) AS `d`
FROM products) t2
```

100% 1:1

Result Grid Filter Rows: Search Export:

a_final_stock	b_final_stock	c_final_stock	d_final_stock
130887	217903	123829	78431

Observation: overall, the hypothesis was not confirmed, as there will still be a significant amount of inventory at each of the warehouses.

At the same time, it was found that at the completion of all open orders, warehouse "d" will still lead in terms of the minimum amount of remaining inventory with 78,431 units, while warehouse "b" will lead in terms of the maximum with 217,903 units.

3 · Efficiency

In the context of exploring the possibility of closing one of the warehouse locations, it is essential to compare the performance of each warehouse to identify any clear "underperformer".

I formulated 6 different efficiency metrics based on the entire history of successfully completed orders:

- Number of successfully completed orders
- Number of units sold
- Total revenue
- Total profit
- Average product margin
- Average profit per unit of product

Note: since one order may include products stored in different warehouses, for analysis purposes (here and below), each individual line with a product item within each order identifier is considered as one *order*.

a. Initially filtering only completed orders based on their status and placing them into a CTE expression, I calculated the above-mentioned metrics for each warehouse.

```sql
WITH real_orders AS (
SELECT `orderNumber` FROM orders
WHERE status IN ('Shipped', 'Resolved')
)

SELECT p.warehouseCode,
COUNT(od.productCode) AS order_counts,
FORMAT(SUM(od.quantityOrdered), 0) AS units_sold,
FORMAT(SUM(od.quantityOrdered * od.priceEach), 2) AS total_revenue,
FORMAT(SUM(od.quantityOrdered * (od.priceEach - p.buyPrice)), 2) AS total_profit,
CONCAT(FORMAT(SUM(od.quantityOrdered * (od.priceEach - p.buyPrice)) /
SUM(od.quantityOrdered * od.priceEach) * 100, 2), '%') AS avg_profitability,
FORMAT(SUM(od.quantityOrdered * (od.priceEach - p.buyPrice)) /
SUM(od.quantityOrdered), 2) AS avg_profit_per_unit
FROM orderdetails od
INNER JOIN products p ON od.productCode = p.productCode
INNER JOIN real_orders ro ON ro.orderNumber = od.orderNumber
GROUP BY `warehouseCode`
ORDER BY `warehouseCode`
```

100% 25:18

Result Grid Filter Rows: Search Export:

warehouseCode	order_counts	units_sold	total_revenue	total_profit	avg_profitability	avg_profit_per_unit
a	658	23,204	1,951,643.26	788,104.36	40.38%	33.96
b	960	33,643	3,648,921.72	1,446,650.06	39.65%	43.00
c	614	21,332	1,669,114.08	684,751.91	41.02%	32.10
d	586	20,622	1,729,651.46	671,675.80	38.83%	32.57

b. Additionally, based on the CTE expression with the table from the previous query, I computed the percentage deviation from the mean value for each metric for each warehouse to conduct a more detailed statistical analysis of the obtained results.

```
SELECT warehouseCode,
CONCAT(FORMAT((order_counts - (SELECT AVG(order_counts) FROM metriks)) / (SELECT AVG(order_counts) FROM metriks) * 100, 2),
"%") AS dev_mean_order_counts,
CONCAT(FORMAT((units_sold - (SELECT AVG(units_sold) FROM metriks)) / (SELECT AVG(units_sold) FROM metriks) * 100, 2), "%")
AS dev_mean_units_sold,
CONCAT(FORMAT((total_revenue - (SELECT AVG(total_revenue) FROM metriks)) / (SELECT AVG(total_revenue) FROM metriks) * 100,
2), "%") AS dev_mean_total_revenue,
CONCAT(FORMAT((total_profit - (SELECT AVG(total_profit) FROM metriks)) / (SELECT AVG(total_profit) FROM metriks) * 100, 2),
"%") AS dev_mean_total_profit,
CONCAT(FORMAT((avg_profitability - (SELECT AVG(avg_profitability) FROM metriks)) / (SELECT AVG(avg_profitability)
FROM metriks) * 100, 2), "%") AS dev_mean_avg_profitability,
CONCAT(FORMAT((avg_profit_per_unit - (SELECT AVG(avg_profit_per_unit) FROM metriks)) / (SELECT AVG(avg_profit_per_unit)
FROM metriks) * 100, 2), "%") AS dev_mean_avg_profit_per_unit
FROM metriks ORDER BY warehouseCode
```

100% 62:33

Result Grid Filter Rows: Search Export:

warehouseCode	dev_mean_order_counts	dev_mean_units_sold	dev_mean_total_revenue	dev_mean_total_profit	dev_mean_avg_profitability	dev_mean_avg_profit_per_unit
a	-6.60%	-6.06%	-13.25%	-12.22%	1.03%	-4.08%
b	36.27%	36.21%	62.19%	61.13%	-0.81%	21.44%
c	-12.85%	-13.64%	-25.81%	-23.73%	2.64%	-9.35%
d	-16.82%	-16.51%	-23.12%	-25.19%	-2.85%	-8.01%

Observation: it was not possible to identify a trend of any warehouse significantly lagging behind others in terms of efficiency (based on one or several metrics).

On the contrary, it was found that one of the warehouses, "b", significantly outperforms the others in five out of six metrics. Even in the metric where it lags behind warehouses "a" and "c" (average product margin), the deviation from the mean value does not exceed 0.81%.

Meanwhile, for the other five metrics, warehouse "b" shows deviations from 21.44% to 62.19% on the positive side. Specifically, it confidently leads in terms of revenue size and total profit, as well as surpasses all other warehouses in profit per unit of sold item.

Aside from the clearly successful warehouse "b", warehouse "a" also demonstrates significantly higher income compared to others. However, in terms of product margin, it exceeds the average value by 1.03%.

It is also interesting to note that warehouse "c" is the leader in terms of product margin at 41.02%, while demonstrating the lowest revenue.

In absolute terms, the "anti-leaders" for each studied metric are as follows for the warehouse locations:

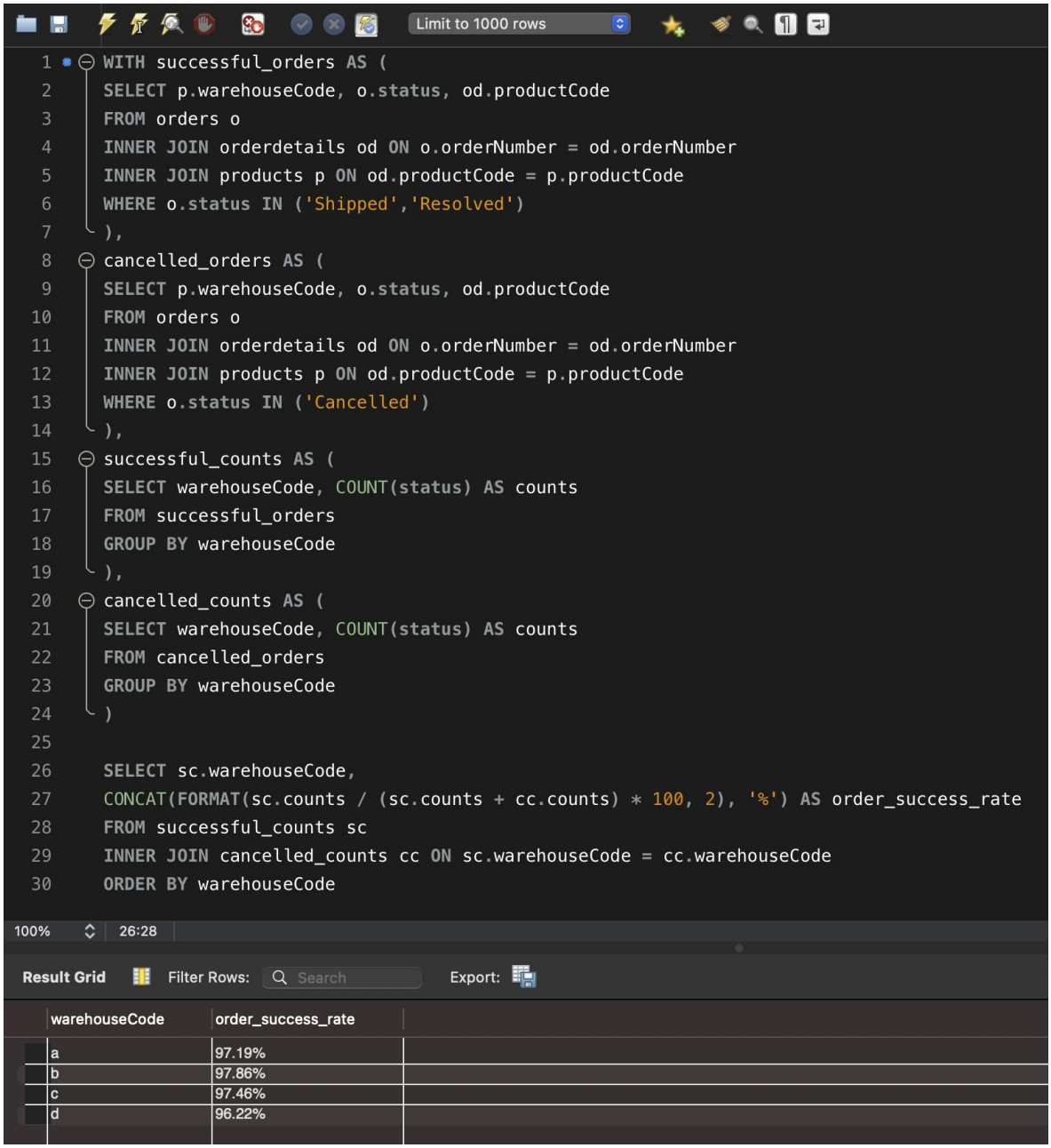
- Lowest number of completed orders: warehouse "d" - 586
- Lowest number of units sold: warehouse "d" - 20,622
- Lowest total revenue: warehouse "c" - 1,669,114.08
- Lowest total profit: warehouse "d" - 671,675.80
- Lowest average product margin: warehouse "d" - 38.83%
- Lowest average profit per unit of product: warehouse "c" - 32.10

4 · Customer Orientation

As it is known, placing an order is good, but what's even better is a successful sale, which in most cases implies a satisfied customer.

In this regard, I decided to calculate the *Order Success Rate* for each warehouse, which is the ratio of the number of successfully completed orders to the total number of orders.

By grouping all orders in the history into completed and canceled ones, then counting their quantity in each group, I calculated the proportion of successful orders in the portfolio of each warehouse.



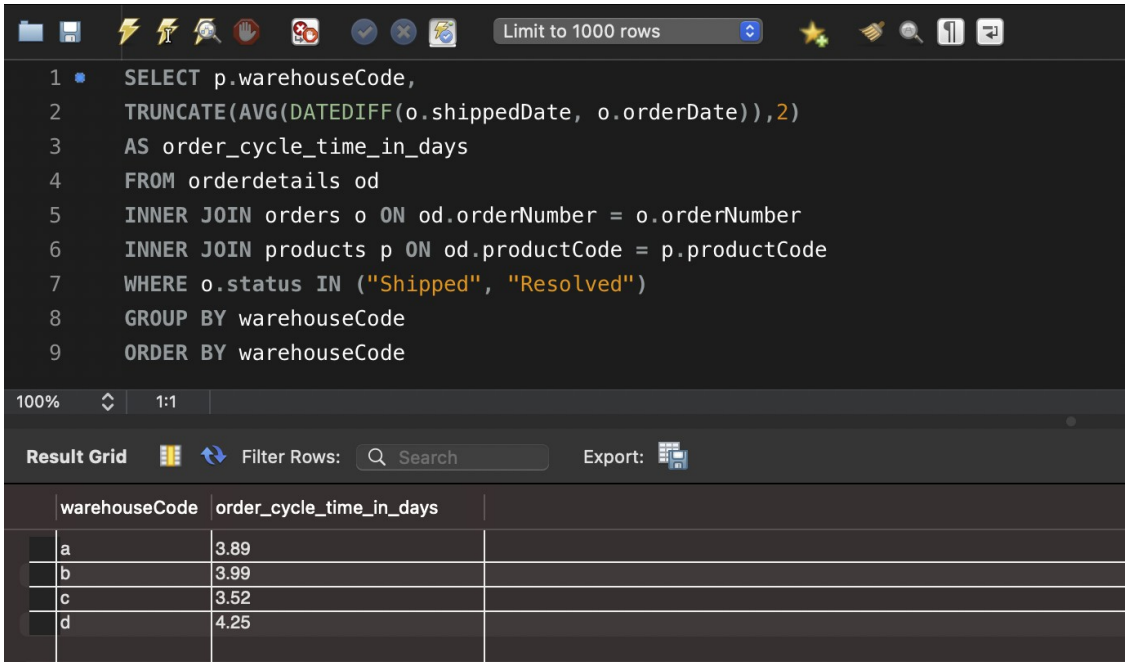
Observation: there is some difference in the metrics across warehouses, but the range boundaries do not appear to be extreme, ranging from 96.22% at warehouse "d" to 97.86% at warehouse "b".

5 · Promptness

The operational efficiency of a warehouse is an integral component of service quality, i.e. customer orientation, and overall company effectiveness.

In my research, I placed this aspect in a separate section and decided to calculate the metric *Order Cycle Time* for warehouses across the entire history of completed orders.

This metric represents the average order fulfillment time (in days) from placement by the customer to the final shipment of the goods.



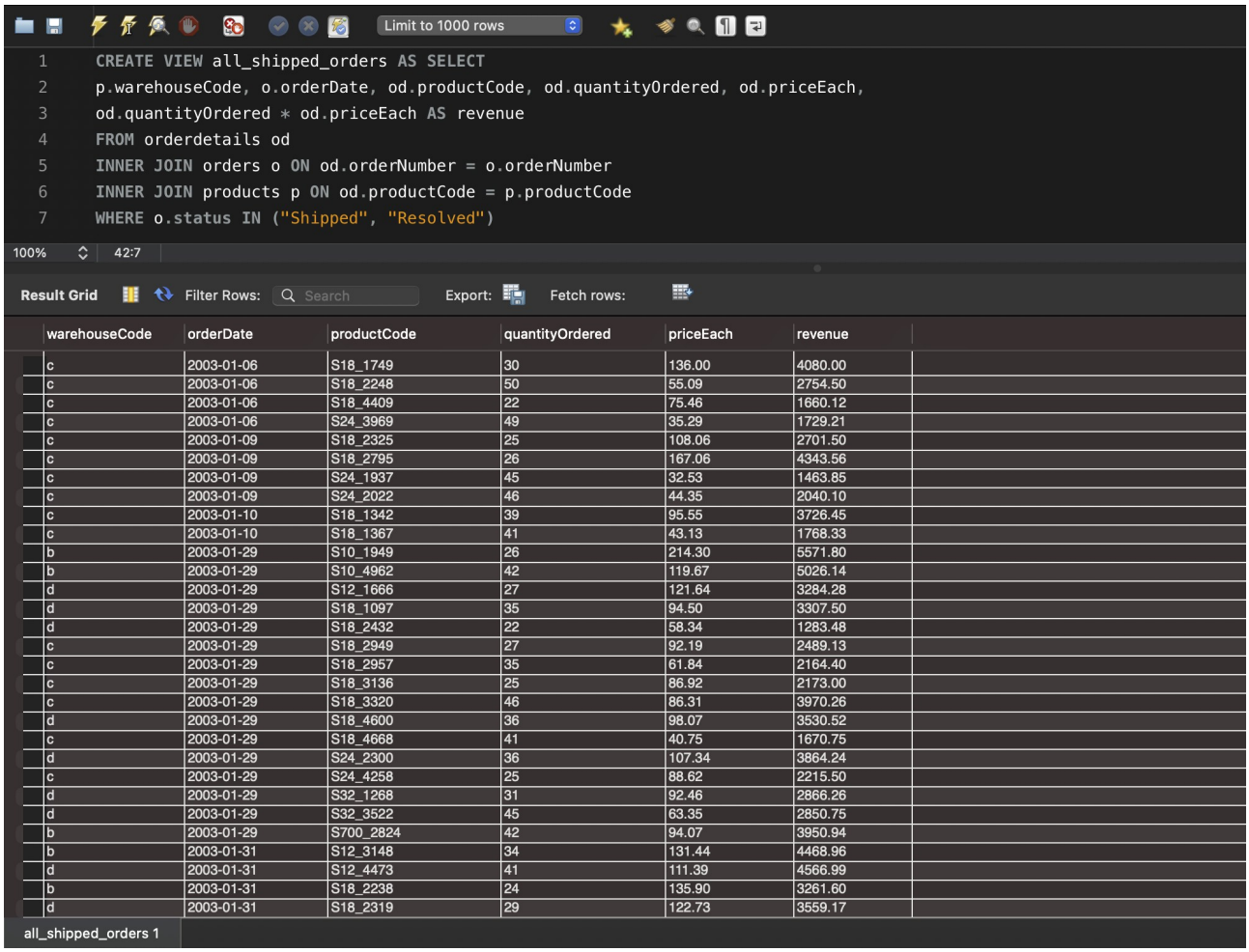
Observation: regarding this metric, warehouse "c" emerges as the champion, with an average order shipment time of three and a half days.

The longest average order processing time is at warehouse "d" - 4.25 days, and on no other warehouse does the metric exceed the level of 4 days.

6 · Growth Patterns

In my project work, I deemed it important to check for the presence of positive trends at any of the warehouses, which may be concealed behind the current snapshot of statistical indicators.

a. In the initial stage, I created a view containing the history of all successfully completed orders, including the order date, warehouse code, product basket and revenue calculation.



```
CREATE VIEW all_shipped_orders AS SELECT
p.warehouseCode, o.orderDate, od.productCode, od.quantityOrdered, od.priceEach,
od.quantityOrdered * od.priceEach AS revenue
FROM orderdetails od
INNER JOIN orders o ON od.orderNumber = o.orderNumber
INNER JOIN products p ON od.productCode = p.productCode
WHERE o.status IN ("Shipped", "Resolved")
```

Limit to 1000 rows · 100% · 42:7

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

warehouseCode	orderDate	productCode	quantityOrdered	priceEach	revenue
c	2003-01-06	S18_1749	30	136.00	4080.00
c	2003-01-06	S18_2248	50	55.09	2754.50
c	2003-01-06	S18_4409	22	75.46	1660.12
c	2003-01-06	S24_3969	49	35.29	1729.21
c	2003-01-09	S18_2325	25	108.06	2701.50
c	2003-01-09	S18_2795	26	167.06	4343.56
c	2003-01-09	S24_1937	45	32.53	1463.85
c	2003-01-09	S24_2022	46	44.35	2040.10
c	2003-01-10	S18_1342	39	95.55	3726.45
c	2003-01-10	S18_1367	41	43.13	1768.33
b	2003-01-29	S10_1949	26	214.30	5571.80
b	2003-01-29	S10_4962	42	119.67	5026.14
d	2003-01-29	S12_1666	27	121.64	3284.28
d	2003-01-29	S18_1097	35	94.50	3307.50
d	2003-01-29	S18_2432	22	58.34	1283.48
c	2003-01-29	S18_2949	27	92.19	2489.13
c	2003-01-29	S18_2957	35	61.84	2164.40
c	2003-01-29	S18_3136	25	86.92	2173.00
c	2003-01-29	S18_3320	46	86.31	3970.26
d	2003-01-29	S18_4600	36	98.07	3530.52
c	2003-01-29	S18_4668	41	40.75	1670.75
d	2003-01-29	S24_2300	36	107.34	3864.24
c	2003-01-29	S24_4258	25	88.62	2215.50
d	2003-01-29	S32_1268	31	92.46	2866.26
d	2003-01-29	S32_3522	45	63.35	2850.75
b	2003-01-29	S700_2824	42	94.07	3950.94
b	2003-01-31	S12_3148	34	131.44	4468.96
d	2003-01-31	S12_4473	41	111.39	4566.99
b	2003-01-31	S18_2238	24	135.90	3261.60
d	2003-01-31	S18_2319	29	122.73	3559.17

all_shipped_orders 1

b. Using a CTE expression, I grouped the table by warehouses and months (year over year in chronological order), calculated the number of orders for each warehouse in each month of the history, and presented the data in a pivot table format.

```sql
WITH orders_grouped AS (
SELECT warehouseCode, DATE_FORMAT(orderDate, '%Y-%m') AS mo_year, COUNT(*) AS order_counts
FROM all_shipped_orders
GROUP BY warehouseCode, mo_year
)

SELECT mo_year,
SUM(CASE WHEN warehouseCode = "a" THEN order_counts ELSE 0 END) AS `a`,
SUM(CASE WHEN warehouseCode = "b" THEN order_counts ELSE 0 END) AS `b`,
SUM(CASE WHEN warehouseCode = "c" THEN order_counts ELSE 0 END) AS `c`,
SUM(CASE WHEN warehouseCode = "d" THEN order_counts ELSE 0 END) AS `d`
FROM orders_grouped
GROUP BY mo_year
ORDER BY mo_year
```

100% 71:11

Result Grid Filter Rows: Search Export:

mo_year	a	b	c	d
2003-01	0	10	16	13
2003-02	20	3	8	10
2003-03	5	26	16	3
2003-04	18	12	8	20
2003-05	7	28	16	7
2003-06	13	10	8	16
2003-07	12	28	16	7
2003-08	24	10	8	16
2003-09	1	37	18	20
2003-10	41	61	28	18
2003-11	55	114	68	69
2003-12	19	34	18	12
2004-01	25	30	24	12
2004-02	25	33	8	22
2004-03	3	17	22	23
2004-04	22	25	17	0
2004-05	25	16	5	14
2004-06	16	17	18	17
2004-07	19	36	24	23
2004-08	31	55	24	23
2004-09	25	23	24	23
2004-10	25	50	41	43
2004-11	75	103	65	52
2004-12	39	35	28	39
2005-01	16	38	22	23
2005-02	20	38	22	17
2005-03	37	27	24	29
2005-04	23	14	15	2

Result 15

Observation: from the obtained table, it is evident that the number of orders varies significantly from month to month (within the year), indicating that this data could serve as the basis for additional research into sales seasonality if such a task is assigned.

c. Since the entire order history covers the period from January 2003 to May 2005, there is an opportunity to display monthly trends from year to year, starting from January 2004, which is the first repeating month in the data.

By placing the obtained pivot table from the previous step into a CTE expression, I queried it using a window function to display the trend of changes in the number of orders each month from year to year.

```sql
WITH orders_grouped AS (
SELECT warehouseCode, DATE_FORMAT(orderDate, '%Y-%m') AS mo_year, COUNT(*) AS order_counts
FROM all_shipped_orders
GROUP BY warehouseCode, mo_year
),
orders_pivot AS (
SELECT mo_year,
SUM(CASE WHEN warehouseCode = "a" THEN order_counts ELSE 0 END) AS `a`,
SUM(CASE WHEN warehouseCode = "b" THEN order_counts ELSE 0 END) AS `b`,
SUM(CASE WHEN warehouseCode = "c" THEN order_counts ELSE 0 END) AS `c`,
SUM(CASE WHEN warehouseCode = "d" THEN order_counts ELSE 0 END) AS `d`
FROM orders_grouped
GROUP BY mo_year
)
SELECT mo_year,
CONCAT(TRUNCATE((a - LAG(a,12) OVER (ORDER BY mo_year)) / LAG(a,12,0) OVER (ORDER BY mo_year) * 100, 2), "%") AS a_trend,
CONCAT(TRUNCATE((b - LAG(a,12) OVER (ORDER BY mo_year)) / LAG(b,12,0) OVER (ORDER BY mo_year) * 100, 2), "%") AS b_trend,
CONCAT(TRUNCATE((c - LAG(a,12) OVER (ORDER BY mo_year)) / LAG(c,12,0) OVER (ORDER BY mo_year) * 100, 2), "%") AS c_trend,
CONCAT(TRUNCATE((d - LAG(a,12) OVER (ORDER BY mo_year)) / LAG(d,12,0) OVER (ORDER BY mo_year) * 100, 2), "%") AS d_trend
FROM orders_pivot LIMIT 10000 OFFSET 12
```

100% 1:21

Result Grid Filter Rows: Search Export:

mo_year	a_trend	b_trend	c_trend	d_trend
2004-01	NULL	300.00%	150.00%	92.30%
2004-02	25.00%	433.33%	-150.00%	20.00%
2004-03	-40.00%	46.15%	106.25%	600.00%
2004-04	22.22%	58.33%	-12.50%	-90.00%
2004-05	257.14%	32.14%	-12.50%	100.00%
2004-06	23.07%	40.00%	62.50%	25.00%
2004-07	58.33%	85.71%	75.00%	157.14%
2004-08	29.16%	310.00%	0.00%	-6.25%
2004-09	2400.00%	59.45%	127.77%	110.00%
2004-10	-39.02%	14.75%	0.00%	11.11%
2004-11	36.36%	42.10%	14.70%	-4.34%
2004-12	105.26%	47.05%	50.00%	166.66%
2005-01	-36.00%	43.33%	-12.50%	-16.66%
2005-02	-20.00%	39.39%	-37.50%	-36.36%
2005-03	1133.33%	141.17%	95.45%	113.04%
2005-04	4.54%	-32.00%	-41.17%	NULL
2005-05	-32.00%	31.25%	-440.00%	-85.71%

d. Following the same algorithm as in the previous two steps, I similarly calculated the statistics of revenue changes each month from year to year.

```sql
orders_pivot AS (
SELECT mo_year,
SUM(CASE WHEN warehouseCode = "a" THEN revenue ELSE 0 END) AS `a`,
SUM(CASE WHEN warehouseCode = "b" THEN revenue ELSE 0 END) AS `b`,
SUM(CASE WHEN warehouseCode = "c" THEN revenue ELSE 0 END) AS `c`,
SUM(CASE WHEN warehouseCode = "d" THEN revenue ELSE 0 END) AS `d`
FROM orders_grouped
GROUP BY mo_year
)
SELECT mo_year,
CONCAT(TRUNCATE((a - LAG(a,12) OVER (ORDER BY mo_year)) / LAG(a,12,0) OVER (ORDER BY mo_year) * 100, 2), "%") AS a_trend,
CONCAT(TRUNCATE((b - LAG(a,12) OVER (ORDER BY mo_year)) / LAG(b,12,0) OVER (ORDER BY mo_year) * 100, 2), "%") AS b_trend,
CONCAT(TRUNCATE((c - LAG(a,12) OVER (ORDER BY mo_year)) / LAG(c,12,0) OVER (ORDER BY mo_year) * 100, 2), "%") AS c_trend,
CONCAT(TRUNCATE((d - LAG(a,12) OVER (ORDER BY mo_year)) / LAG(d,12,0) OVER (ORDER BY mo_year) * 100, 2), "%") AS d_trend
FROM orders_pivot LIMIT 10000 OFFSET 12
```

100% 1:21

Result Grid Filter Rows: Search Export:

mo_year	a_trend	b_trend	c_trend	d_trend
2004-01	NULL	319.44%	164.01%	95.20%
2004-02	34.10%	343.99%	-157.26%	4.90%
2004-03	12.48%	72.99%	102.17%	649.36%
2004-04	-2.72%	50.06%	-34.45%	-101.54%
2004-05	322.76%	54.76%	-17.04%	87.50%
2004-06	56.76%	39.46%	54.46%	50.01%
2004-07	53.60%	96.91%	88.26%	132.07%
2004-08	30.18%	300.08%	-18.38%	-13.91%
2004-09	2286.71%	67.85%	123.83%	103.18%
2004-10	-43.20%	40.57%	-14.53%	6.27%
2004-11	36.33%	55.66%	9.96%	-8.75%
2004-12	95.94%	55.65%	42.59%	126.54%
2005-01	-40.40%	59.59%	-12.84%	-24.27%
2005-02	-23.01%	55.38%	-83.44%	-52.39%
2005-03	827.30%	111.93%	88.47%	103.08%
2005-04	59.76%	37.00%	-8.26%	NULL
2005-05	-44.12%	49.02%	-638.81%	-94.47%

Observation: upon reviewing the obtained data, it can be noted that except for rare negative spikes, all warehouses show a confident growth in activity.

The largest decline in the number of orders is shown by warehouse "c" in May 2004 - by 440%, while the maximum positive spike is observed for warehouse "a" in September 2004 - by 2400%.

The peaks of revenue fluctuations coincide with the extremes in the number of orders: a decrease in sales at warehouse "c" in May 2004 by 638% and an increase at warehouse "a" in September 2004 by 2286%.

It is noteworthy that warehouse "b" is the only one that does not show negative revenue growth for any month in the considered period.

e. Having in the history only one, yet complete, repeating calendar year (2004), I calculated the annual trend of changes in the number of orders, i.e. how the number of orders (on each warehouse) changed in 2004 compared to 2003.

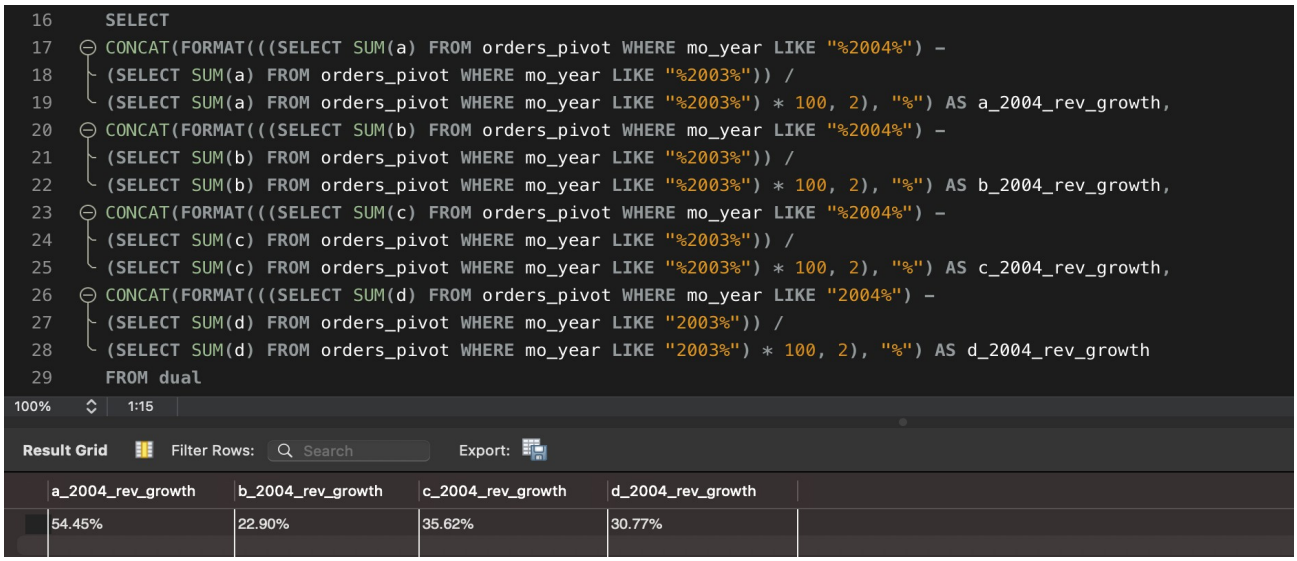
```
16    SELECT
17  CONCAT(FORMAT(((SELECT SUM(a) FROM orders_pivot WHERE mo_year LIKE "%2004%") -
18    (SELECT SUM(a) FROM orders_pivot WHERE mo_year LIKE "%2003%")) /
19    (SELECT SUM(a) FROM orders_pivot WHERE mo_year LIKE "%2003%") * 100, 2), "%") AS a_2004_growth,
20  CONCAT(FORMAT(((SELECT SUM(b) FROM orders_pivot WHERE mo_year LIKE "%2004%") -
21    (SELECT SUM(b) FROM orders_pivot WHERE mo_year LIKE "%2003%")) /
22    (SELECT SUM(b) FROM orders_pivot WHERE mo_year LIKE "%2003%") * 100, 2), "%") AS b_2004_growth,
23  CONCAT(FORMAT(((SELECT SUM(c) FROM orders_pivot WHERE mo_year LIKE "%2004%") -
24    (SELECT SUM(c) FROM orders_pivot WHERE mo_year LIKE "%2003%")) /
25    (SELECT SUM(c) FROM orders_pivot WHERE mo_year LIKE "%2003%") * 100, 2), "%") AS c_2004_growth,
26  CONCAT(FORMAT(((SELECT SUM(d) FROM orders_pivot WHERE mo_year LIKE "2004%") -
27    (SELECT SUM(d) FROM orders_pivot WHERE mo_year LIKE "2003%")) /
28    (SELECT SUM(d) FROM orders_pivot WHERE mo_year LIKE "2003%") * 100, 2), "%") AS d_2004_growth
29    FROM dual
```

100% 33:28

Result Grid Filter Rows: Search Export:

a_2004_growth	b_2004_growth	c_2004_growth	d_2004_growth
53.49%	17.96%	31.58%	37.91%

f. Similarly, I calculated the annual trend of changes in revenue for each warehouse in 2004 compared to 2003.



```
16    SELECT
17  CONCAT(FORMAT(((SELECT SUM(a) FROM orders_pivot WHERE mo_year LIKE "%2004%") -
18    (SELECT SUM(a) FROM orders_pivot WHERE mo_year LIKE "%2003%")) /
19    (SELECT SUM(a) FROM orders_pivot WHERE mo_year LIKE "%2003%") * 100, 2), "%") AS a_2004_rev_growth,
20  CONCAT(FORMAT(((SELECT SUM(b) FROM orders_pivot WHERE mo_year LIKE "%2004%") -
21    (SELECT SUM(b) FROM orders_pivot WHERE mo_year LIKE "%2003%")) /
22    (SELECT SUM(b) FROM orders_pivot WHERE mo_year LIKE "%2003%") * 100, 2), "%") AS b_2004_rev_growth,
23  CONCAT(FORMAT(((SELECT SUM(c) FROM orders_pivot WHERE mo_year LIKE "%2004%") -
24    (SELECT SUM(c) FROM orders_pivot WHERE mo_year LIKE "%2003%")) /
25    (SELECT SUM(c) FROM orders_pivot WHERE mo_year LIKE "%2003%") * 100, 2), "%") AS c_2004_rev_growth,
26  CONCAT(FORMAT(((SELECT SUM(d) FROM orders_pivot WHERE mo_year LIKE "2004%") -
27    (SELECT SUM(d) FROM orders_pivot WHERE mo_year LIKE "2003%")) /
28    (SELECT SUM(d) FROM orders_pivot WHERE mo_year LIKE "2003%") * 100, 2), "%") AS d_2004_rev_growth
29    FROM dual
```

100% 1:15

Result Grid Filter Rows: Search Export:

a_2004_rev_growth	b_2004_rev_growth	c_2004_rev_growth	d_2004_rev_growth
54.45%	22.90%	35.62%	30.77%

Observation: the statistics for the annual period confirm a strong upward trend for both indicators across all warehouse facilities.

Warehouse "a" emerged as the leader in both revenue growth and order quantity growth, with increases of 54.45% and 53.49% respectively. Even the "anti-leader" in each ranking, warehouse "b", still processed 17.96% more orders, with a total value 22.9% higher than the previous year.

It's also noteworthy that there is a clear correlation for each warehouse between the increase in order quantity and revenue growth.

g. The statistics for the most recent year in the order history (2005) cover 5 months from January to May. Data for the same period is also available for 2003 and 2004, thus allowing for an analysis of changes in revenue and order quantity over a 5-month period across multiple reporting periods.

Using the table from previous queries, I filtered the data to include only the months from January to May for each year. Then, I calculated how the number of orders processed by each warehouse changed for the first 5 months of 2004 compared to 2003, and for the first 5 months of 2005 compared to 2004.

```sql
WITH orders_grouped AS (
SELECT warehouseCode, DATE_FORMAT(orderDate, '%Y-%m') AS mo_year, COUNT(*) AS order_counts
FROM all_shipped_orders
GROUP BY warehouseCode, mo_year
),
orders_pivot AS (
SELECT mo_year,
SUM(CASE WHEN warehouseCode = "a" THEN order_counts ELSE 0 END) AS `a`,
SUM(CASE WHEN warehouseCode = "b" THEN order_counts ELSE 0 END) AS `b`,
SUM(CASE WHEN warehouseCode = "c" THEN order_counts ELSE 0 END) AS `c`,
SUM(CASE WHEN warehouseCode = "d" THEN order_counts ELSE 0 END) AS `d`
FROM orders_grouped
GROUP BY mo_year
),
orders_pivot_filtered AS (
SELECT (CASE WHEN mo_year LIKE "%2003%" THEN 2003 WHEN mo_year LIKE "%2004%" THEN 2004
WHEN mo_year LIKE "%2005%" THEN 2005 END) AS period,
a,b,c,d
FROM orders_pivot
WHERE mo_year LIKE "%01" OR mo_year LIKE "%02" OR mo_year LIKE "%03"
OR mo_year LIKE "%04" OR mo_year LIKE "%05"
),
order_counts_per_year AS (SELECT period, SUM(a) AS a, SUM(b) AS b, SUM(c) AS c, SUM(d) AS d
FROM orders_pivot_filtered GROUP BY period)

SELECT CONCAT(FORMAT((LEAD(a) OVER (ORDER BY period) - a) / a * 100, 2), "%") AS a_trend,
CONCAT(FORMAT((LEAD(b) OVER (ORDER BY period) - b) / b * 100, 2), "%") AS b_trend,
CONCAT(FORMAT((LEAD(c) OVER (ORDER BY period) - c) / c * 100, 2), "%") AS c_trend,
CONCAT(FORMAT((LEAD(d) OVER (ORDER BY period) - d) / d * 100, 2), "%") AS d_trend
FROM order_counts_per_year
```

100% 1:1

Result Grid Filter Rows: Search Export:

a_trend	b_trend	c_trend	d_trend
100.00%	53.16%	18.75%	33.96%
13.00%	21.49%	13.16%	18.31%

h. Similarly, I measured the trend of revenue changes from January to May year-over-year.

```sql
orders_pivot_filtered AS (
SELECT (CASE WHEN mo_year LIKE "%2003%" THEN 2003 WHEN mo_year LIKE "%2004%" THEN 2004
WHEN mo_year LIKE "%2005%" THEN 2005 END) AS period,
a,b,c,d
FROM orders_pivot
WHERE mo_year LIKE "%01" OR mo_year LIKE "%02" OR mo_year LIKE "%03"
OR mo_year LIKE "%04" OR mo_year LIKE "%05"
),
rev_per_year AS (SELECT period, SUM(a) AS a, SUM(b) AS b, SUM(c) AS c, SUM(d) AS d
FROM orders_pivot_filtered GROUP BY period)

SELECT CONCAT(FORMAT((LEAD(a) OVER (ORDER BY period) - a) / a * 100, 2), "%") AS a_trend,
CONCAT(FORMAT((LEAD(b) OVER (ORDER BY period) - b) / b * 100, 2), "%") AS b_trend,
CONCAT(FORMAT((LEAD(c) OVER (ORDER BY period) - c) / c * 100, 2), "%") AS c_trend,
CONCAT(FORMAT((LEAD(d) OVER (ORDER BY period) - d) / d * 100, 2), "%") AS d_trend
```

100% 61:29

Result Grid Filter Rows: Search Export:

a_trend	b_trend	c_trend	d_trend
106.65%	57.74%	23.71%	34.47%
16.11%	24.80%	13.85%	18.40%

Observation: while 2004 was clearly a breakthrough year, 2005 brings more modest growth. However, the overall trend appears to work similarly for all warehouses and warrants a separate analysis of its underlying factors.

When comparing the situation at each individual warehouse within the overall trend, the leader in revenue growth in 2004 - warehouse "a" - more than doubled this metric by 106.65%, while the minimum growth was shown by warehouse "c" - by 23.71%.

In 2005, the difference in revenue growth metrics between warehouses is not as significant - ranging from 24.8% for warehouse "b" to 13.85% for warehouse "c", which, worth noting, leads in both of the considered reporting periods' minimums.

As with the annual growth figures, the analysis of 5-month intervals year-over-year shows a strong positive correlation between the growth in the number of orders and the growth in revenue.

7 · Inventory Turnover

In this section of the study, I focused on the properties of the products handled by each warehouse. I calculated the turnover of inventory items within the entire history of fulfilled orders and the *Stock-to-Depletion Ratio* for each product item.

a. Grouped the current inventory and total quantity sold for each product item in a CTE expression, then calculated the average monthly inventory turnover based on the number of months in the history (29) and the *Stock-to-Depletion Ratio* - that is, how many months the current inventory of each item will last if the average sales level remains constant.

```sql
CREATE VIEW product_ratios AS

WITH stock_vs_turnover AS (
SELECT p.productCode,
(SELECT warehouseCode FROM products WHERE productCode = p.productCode) AS warehouse,
(SELECT quantityInStock FROM products WHERE productCode = p.productCode) AS stock,
SUM(pd.quantityOrdered) AS ordered_units_total
FROM products p
INNER JOIN orderdetails pd ON p.productCode = pd.productCode
GROUP BY productCode
),

stock_vs_turnover_updated AS (
SELECT *,
TRUNCATE(ordered_units_total / 29, 1) AS avg_monthly_turnover
FROM stock_vs_turnover
)

SELECT *,
TRUNCATE(stock / avg_monthly_turnover, 1) AS inventory_ratio_in_months
FROM stock_vs_turnover_updated
```

100% 31:21

Result Grid Filter Rows: Search Export:

productCode	warehouse	stock	ordered_units_total	avg_monthly_turnover	inventory_ratio_in_months
S10_1678	a	7933	1057	36.4	217.9
S10_1949	b	7305	961	33.1	220.6
S10_2016	a	6625	999	34.4	192.5
S10_4698	a	5582	985	33.9	164.6
S10_4757	b	3252	1030	35.5	91.6
S10_4962	b	6791	932	32.1	211.5
S12_1099	b	68	933	32.1	2.1
S12_1108	b	3619	1019	35.1	103.1
S12_1666	d	1579	972	33.5	47.1
S12_2823	a	9997	1028	35.4	282.4
S12_3148	b	6906	963	33.2	208.0
S12_3380	b	9123	925	31.8	286.8
S12_3891	b	1049	965	33.2	31.5
S12_3990	b	5663	900	31.0	182.6
S12_4473	d	6125	1056	36.4	168.2
S12_4675	b	7323	992	34.2	214.1
S18_1097	d	2613	999	34.4	75.9
S18_1129	b	3975	947	32.6	121.9
S18_1342	c	8693	1111	38.3	226.9

product_ratios 1

Observation: there is a fairly consistent pattern in the average monthly sales quantity, while a significant variation in the *Stock-to-Depletion Ratios*.

b. Calculated the minimum, maximum, and average values of the average monthly turnover for all product items.

```sql
SELECT MIN(avg_monthly_turnover),
AVG(avg_monthly_turnover),
MAX(avg_monthly_turnover)
FROM product_ratios
```

Result Grid

MIN(avg_monthly_turnover)	AVG(avg_monthly_turnover)	MAX(avg_monthly_turnover)
26.4	33.33761	62.3

Observation: the sales statistics across the assortment appear to be fairly smooth. Despite the nearly threefold difference between the minimum and maximum values, this range seems normal for such a specific category of products as vehicles, where there are various price ranges.

There are no items that haven't sold at all or have sold only a few units, indicating a well-curated assortment.

c. Calculated the number of product items with average monthly turnover less than or equal to the mean value and displayed their distribution across warehouses, grouped by product categories.

```sql
WITH ratios_less_avg AS (
SELECT pr.warehouse, p.productLine, COUNT(pr.productCode) AS product_counts
FROM product_ratios pr
INNER JOIN products p ON pr.productCode = p.productCode
WHERE pr.avg_monthly_turnover <= (SELECT AVG(avg_monthly_turnover) FROM product_ratios)
GROUP BY warehouse, productLine
)

SELECT productLine,
CASE WHEN warehouse = "a" THEN product_counts ELSE 0 END AS `a`,
CASE WHEN warehouse = "b" THEN product_counts ELSE 0 END AS `b`,
CASE WHEN warehouse = "c" THEN product_counts ELSE 0 END AS `c`,
CASE WHEN warehouse = "d" THEN product_counts ELSE 0 END AS `d`
FROM ratios_less_avg
ORDER BY productLine
```

Result Grid

productLine	a	b	c	d
Classic Cars	0	24	0	0
Motorcycles	5	0	0	0
Planes	4	0	0	0
Ships	0	0	0	7
Trains	0	0	0	3
Trucks and Buses	0	0	0	2
Vintage Cars	0	0	13	0

d. Counted the number of product items with monthly turnover higher than the mean value and displayed their distribution across warehouses, grouped by product categories.

```sql
WITH ratios_higher_avg AS (
SELECT pr.warehouse, p.productLine, COUNT(pr.productCode) AS product_counts
FROM product_ratios pr
INNER JOIN products p ON pr.productCode = p.productCode
WHERE pr.avg_monthly_turnover > (SELECT AVG(avg_monthly_turnover) FROM product_ratios)
GROUP BY warehouse, productLine
)

SELECT productLine,
CASE WHEN warehouse = "a" THEN product_counts ELSE 0 END AS `a`,
CASE WHEN warehouse = "b" THEN product_counts ELSE 0 END AS `b`,
CASE WHEN warehouse = "c" THEN product_counts ELSE 0 END AS `c`,
CASE WHEN warehouse = "d" THEN product_counts ELSE 0 END AS `d`
FROM ratios_higher_avg
ORDER BY productLine
```

100% 44:12

Result Grid Filter Rows: Search Export:

productLine	a	b	c	d
Classic Cars	0	13	0	0
Motorcycles	8	0	0	0
Planes	8	0	0	0
Ships	0	0	0	2
Trucks and Buses	0	0	0	9
Vintage Cars	0	0	11	0

e. Assuming a standard inventory turnover period of 6 months, calculated the number of product items with inventory levels within the standard range.

```sql
SELECT * FROM product_ratios
WHERE inventory_ratio_in_months <= 6
```

100% 37:2

Result Grid Filter Rows: Search Export:

productCode	warehouse	stock	ordered_units_total	avg_monthly_turnover	inventory_ratio_in_months
S12_1099	b	68	933	32.1	2.1
S24_2000	a	15	1015	35.0	0.4
S32_1374	a	178	1014	34.9	5.1
S32_4289	c	136	972	33.5	4.0

Observation: it was found that there are only 4 such items.

f. Calculated the number of product items with inventory levels above the standard value.

Limit to 1000 rows

```sql
SELECT COUNT(*) FROM product_ratios
WHERE inventory_ratio_in_months > 6
```

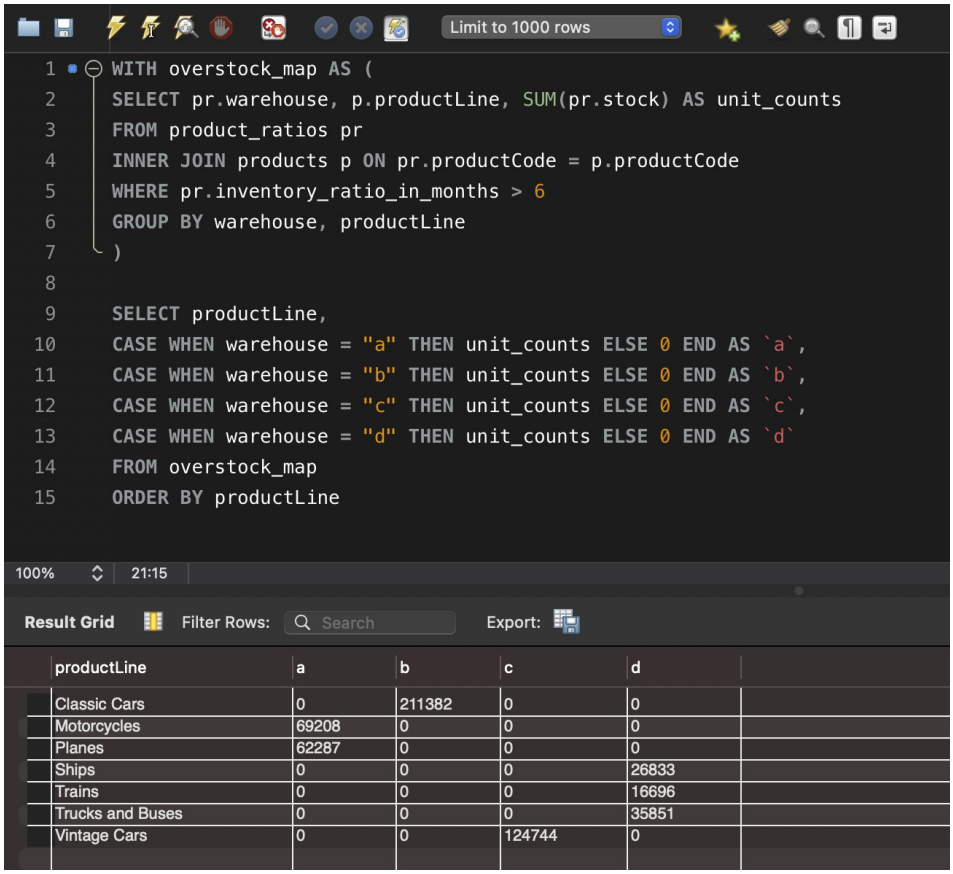
100% 16:1

Result Grid Filter Rows: Search Export:

COUNT(*)
105

Observation: it was found that there are 105 such items.

g. Displayed the distribution of product items with elevated inventory levels (in quantity of items) across all warehouses in a pivot table format with grouping by product categories.

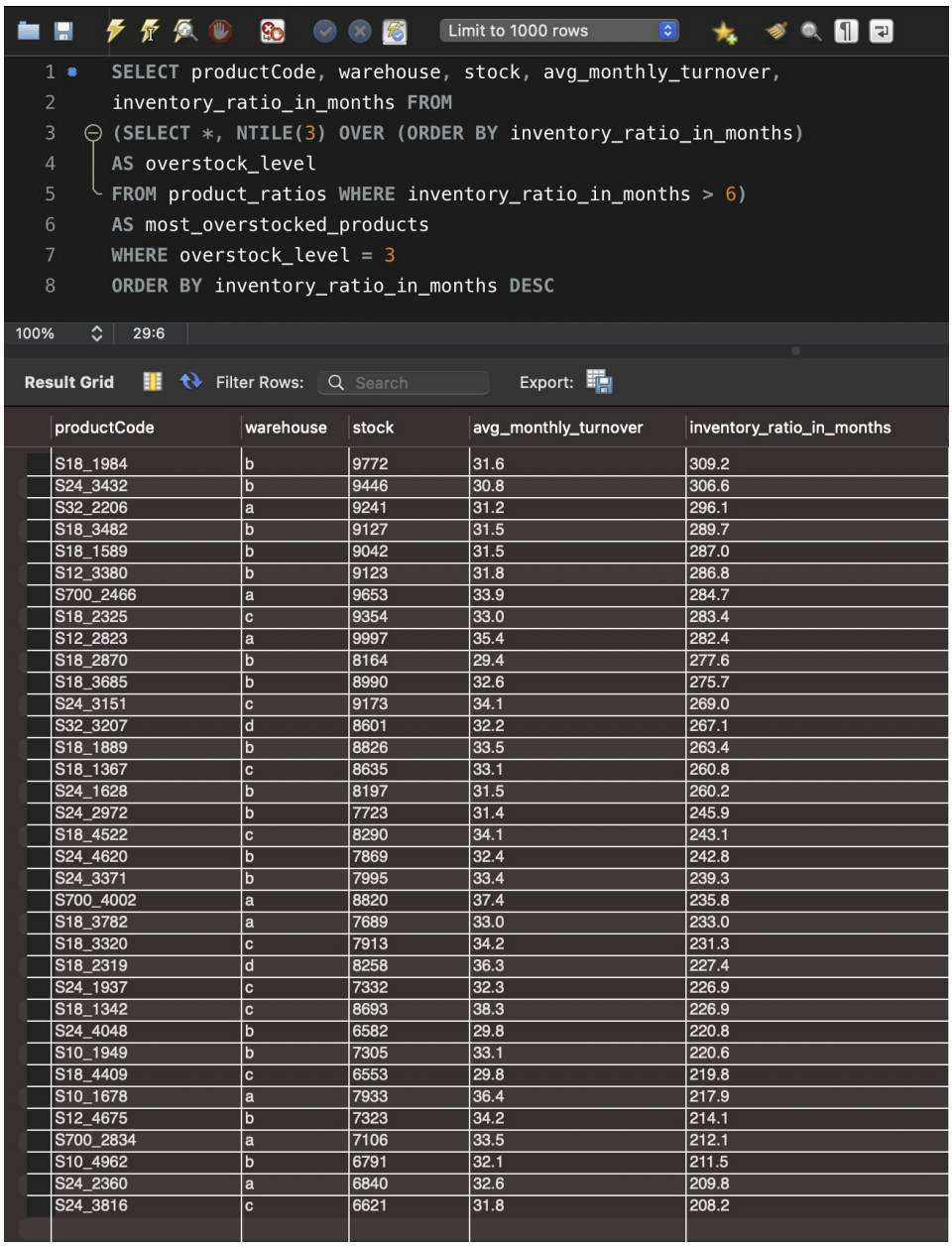


Observation: only warehouses "b" and "c" have stocked positions within a single product category, and in both cases, it's automobiles.

Warehouses "a" and "d" have vast quantities of stocked items across multiple, and in both cases, very bulky product categories, such as airplanes, trains, and ships.

In purely physical terms (dimensions, weight, labor intensity of relocating goods), it's evident that warehouse "c" is the least burdened.

h. Ranked all product items with elevated inventory levels into three levels of stockiness using window functions and filtered the list of product items with the highest *Stock-to-Depletion Ratio*.



```
Limit to 1000 rows

1  SELECT productCode, warehouse, stock, avg_monthly_turnover,
2  inventory_ratio_in_months FROM
3  (SELECT *, NTILE(3) OVER (ORDER BY inventory_ratio_in_months)
4  AS overstock_level
5  FROM product_ratios WHERE inventory_ratio_in_months > 6)
6  AS most_overstocked_products
7  WHERE overstock_level = 3
8  ORDER BY inventory_ratio_in_months DESC
```

100% 29:6

Result Grid Filter Rows: Search Export:

productCode	warehouse	stock	avg_monthly_turnover	inventory_ratio_in_months
S18_1984	b	9772	31.6	309.2
S24_3432	b	9446	30.8	306.6
S32_2206	a	9241	31.2	296.1
S18_3482	b	9127	31.5	289.7
S18_1589	b	9042	31.5	287.0
S12_3380	b	9123	31.8	286.8
S700_2466	a	9653	33.9	284.7
S18_2325	c	9354	33.0	283.4
S12_2823	a	9997	35.4	282.4
S18_2870	b	8164	29.4	277.6
S18_3685	b	8990	32.6	275.7
S24_3151	c	9173	34.1	269.0
S32_3207	d	8601	32.2	267.1
S18_1889	b	8826	33.5	263.4
S18_1367	c	8635	33.1	260.8
S24_1628	b	8197	31.5	260.2
S24_2972	b	7723	31.4	245.9
S18_4522	c	8290	34.1	243.1
S24_4620	b	7869	32.4	242.8
S24_3371	b	7995	33.4	239.3
S700_4002	a	8820	37.4	235.8
S18_3782	a	7689	33.0	233.0
S18_3320	c	7913	34.2	231.3
S18_2319	d	8258	36.3	227.4
S24_1937	c	7332	32.3	226.9
S18_1342	c	8693	38.3	226.9
S24_4048	b	6582	29.8	220.8
S10_1949	b	7305	33.1	220.6
S18_4409	c	6553	29.8	219.8
S10_1678	a	7933	36.4	217.9
S12_4675	b	7323	34.2	214.1
S700_2834	a	7106	33.5	212.1
S10_4962	b	6791	32.1	211.5
S24_2360	a	6840	32.6	209.8
S24_3816	c	6621	31.8	208.2

8 · Geography

In the final section of the study, I considered it important to examine the geographical distribution of orders among warehouses and their alignment with the warehouses' locations.

By grouping all completed orders by servicing warehouses and destination countries in a CTE expression, I presented a summary table with the number of orders from each country for each warehouse.

```sql
WITH geography AS (
SELECT c.country, CONCAT(p.warehouseCode, " - ", w.warehouseName) AS warehouse,
COUNT(od.orderNumber) AS order_counts FROM orderdetails od
INNER JOIN products p ON od.productCode = p.productCode
INNER JOIN warehouses w ON p.warehouseCode = w.warehouseCode
INNER JOIN orders o ON od.orderNumber = o.orderNumber
INNER JOIN customers c ON o.customerNumber = c.customerNumber
WHERE o.status IN ("Shipped", "Resolved")
GROUP BY country, warehouse
)

SELECT country,
SUM(CASE WHEN warehouse = "a - North" THEN order_counts ELSE 0 END) AS "a - North",
SUM(CASE WHEN warehouse = "b - East" THEN order_counts ELSE 0 END) AS "b - East",
SUM(CASE WHEN warehouse = "c - West" THEN order_counts ELSE 0 END) AS "c - West",
SUM(CASE WHEN warehouse = "d - South" THEN order_counts ELSE 0 END) AS "d - South"
FROM geography GROUP BY country ORDER BY country
```

100% 49:17

Result Grid Filter Rows: Search Export:

country	a - North	b - East	c - West	d - South
Australia	49	46	50	22
Austria	11	25	10	9
Belgium	1	4	9	14
Canada	11	14	15	30
Denmark	2	31	7	20
Finland	25	38	7	22
France	100	93	58	50
Germany	11	36	9	6
Hong Kong	13	0	3	0
Ireland	6	6	1	3
Italy	38	29	42	12
Japan	25	8	9	10
New Zealand	33	37	43	17
Norway	25	35	14	11
Philippines	12	13	1	0
Singapore	1	32	14	32
Spain	34	118	68	94
Sweden	3	14	8	10
Switzerland	0	31	0	0
UK	24	32	39	35
USA	234	318	207	189

Observation: the obtained summary table can be further utilized for a more detailed examination of geography, such as grouping countries by regions, if such a task arises.

A cursory overview immediately reveals some inconsistencies, for instance, the northern warehouse "a" serves less than a quarter of orders from Canada, while the southern warehouse "d" handles less than 15% of orders from New Zealand.

Conclusions and Recommendations.

- Despite warehouse "d" having the minimum inventory levels (which might suggest it's easier to close), it's also the least efficient and lags behind in most metrics. However, its inventory mainly consists of ships, airplanes, and other bulky vehicles, making relocation costly. Hence, it's not the best candidate for closure.

  Warehouse "b" demonstrates the highest profitability per unit, has the largest inventory, generates the highest revenue (and profit), and shows the most significant growth over the past five months. It's unlikely that closing the most efficient warehouse would be advisable.

  Warehouse "a" exhibited promising growth in 2004, and its growth over five-month periods was higher than that of warehouse "c" in both reporting periods. In addition, the product categories at warehouse "a" include motorcycles and airplanes, whereas warehouse "c" only stocks automobiles.

  Therefore, the most economically viable solution would be to consider closing warehouse "c" and relocating its goods, which are the most profitable within the company's portfolio (averaging a 41.02% profit margin), to the remaining three warehouses, thus enhancing their profitability.

- If a decision is made to close warehouse "c", it's recommended to analyze the workload levels of warehouses "a", "b" and "d", as well as conduct a geographical analysis of the customers of all products from warehouse "c" to relocate the goods to warehouses that are closer to the main demand region.

- It's advisable to cease purchasing any overstocked items (as identified in section 7 of this study) and monitor sales growth. Quarterly adjustments should be made to the reorder point calculation for each item, resuming purchases only when the *Stock-to-Depletion Ratio* reaches a normal value.

- It's recommended to organize a clearance sale or any other active marketing promotion for all items with the highest overstock levels, as determined in section 7.h. of this study.

Skills.

During the work on this project, I demonstrated the following professional competencies:

- Importing an existing database using MySQL Workbench
- Familiarization with the business and its data through the study of the relational model and exploration of data tables in MySQL Workbench
- Analysis of inventory data using SQL queries in MySQL Workbench, extracting data from a multi-table relational database using SQL commands such as SELECT, WHERE, GROUP BY, HAVING, and ORDER BY
- Creating views and CTE expressions
- Utilizing data aggregation functions and SQL window functions
- Using special SQL language operators such as IN, LIKE, and others
- Creating pivot tables using the CASE operator
- Joining different tables in SQL queries using INNER JOIN and OUTER JOIN methods
- Developing recommendations and proposals to address business needs based on data analysis
- Supporting formulated recommendations and proposals for inventory reduction in the form of scripts (queries) and their results

Acknowledgment.

I would like to express gratitude to the Coursera project for supporting the educational process and providing the opportunity to refine skills acquired through online courses by completing educational projects, including free ones, such as this.