

Исследовательский анализ данных с MySQL Workbench.

Дата: 27 марта 2024

Проект выполнил: Антон Старшев

<http://linkedin.com/in/starshev>

Источник проекта

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>

Краткое описание

В рамках этого проекта, который занимает около 8-10 часов, предложено провести анализ данных начинающего уровня в вымышленной компании *Mint Classics* с целью поддержки бизнес-решений, связанных с инвентаризацией, которые приведут к закрытию одного из складских помещений.

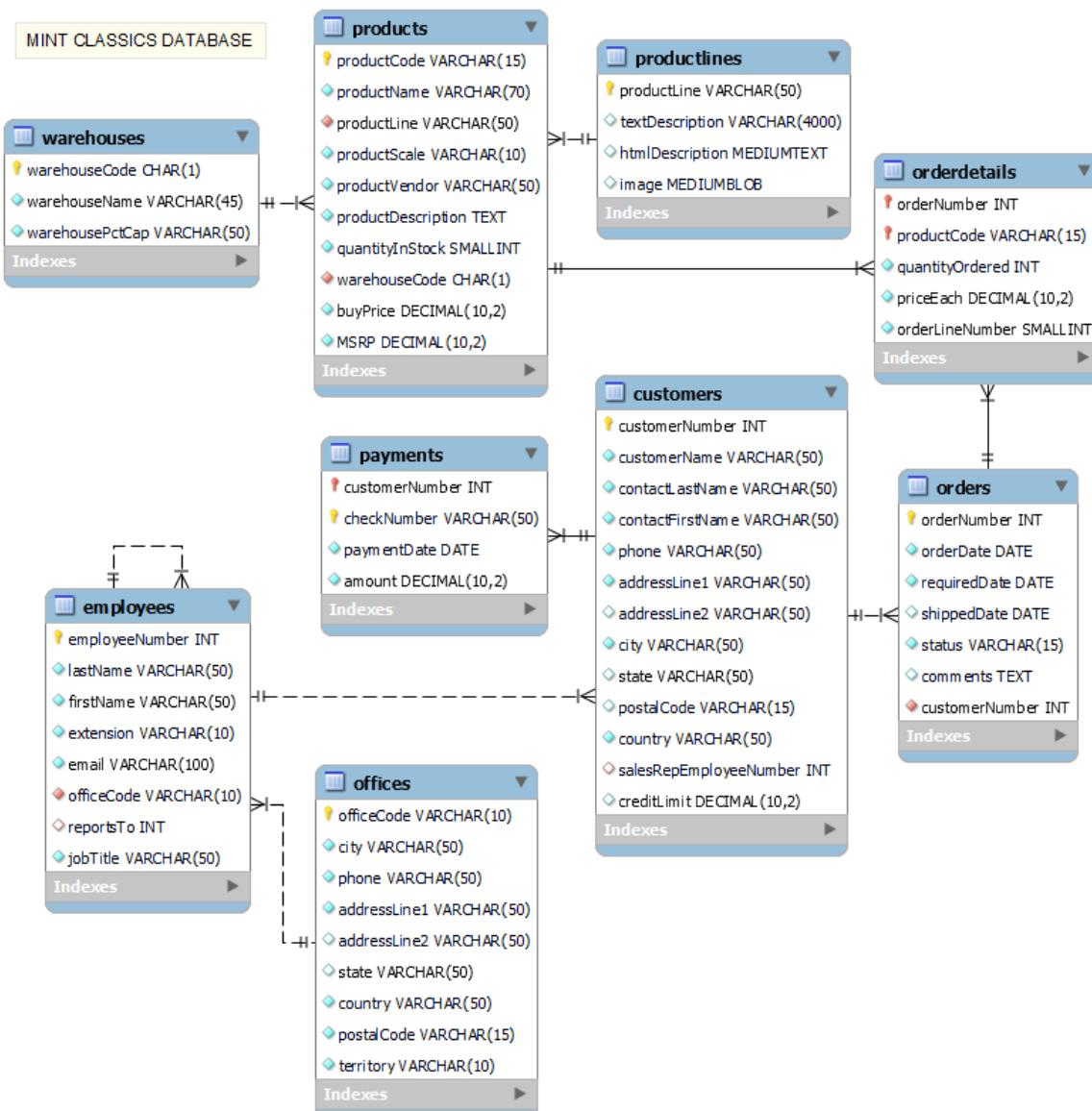
Сценарий проекта.

Компания *Mint Classics*, ритейлер классических моделей автомобилей и других транспортных средств, рассматривает возможность закрытия одного из своих складских помещений.

Для поддержки принятия решений на основе данных, компания ищет предложения по реорганизации или сокращению запасов. Предлагается в качестве аналитика данных использовать MySQL Workbench для ознакомления с бизнесом компании путем изучения текущих данных.

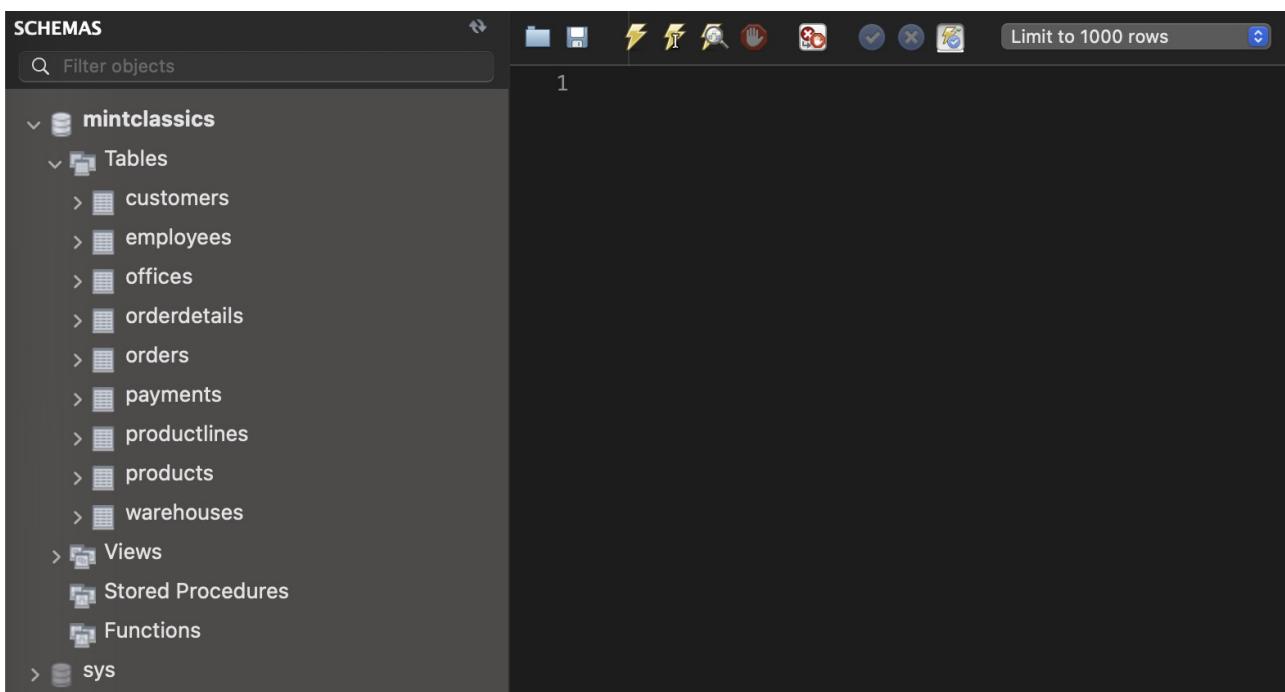
Целью проекта является проведение исследовательского анализа данных для выявления каких-либо закономерностей и наблюдений, которые могут повлиять на сокращение или реорганизацию запасов на складах *Mint Classics*, а также предоставление аналитических исследований и рекомендаций на основе данных.

Предоставляется скрипт для импортирования базы данных и следующая архитектурная модель:



Импорт базы данных.

Приступив к выполнению проекта, я импортировал предоставленный скрипт базы данных как self-contained файл (содержащий структуру таблиц и данные) с помощью мастера Data Import платформы MySQL Workbench.



Ознакомление с моделью базы данных.

Изучив архитектуру базы данных, я выявил таблицы, которые будут использованы в моем исследовании, а также их поля и связи:

- *orders* (заказы) с первичным ключом *orderNumber* (номер заказа) и внешним ключом *customerNumber* (номер клиента) для связи с таблицей *customers* (клиенты)
- *orderdetails* (детали заказов) с составным первичным ключом *orderNumber* (номер заказа) и *productCode* (код продукта), который также является внешним ключом для связи с таблицей *products* (товары)
- *products* (товары) с первичным ключом *productCode* (код продукта) и внешним ключом *warehouseCode* (код склада) для связи с таблицей *warehouses* (склады)
- *warehouses* (склады) с первичным ключом *warehouseCode* (код склада)
- *customers* (клиенты) с первичным ключом *customerNumber* (номер клиента)

Исследовательский анализ данных.

Начиная собственно исследование, я сформулировал несколько практических метрик (вопросов, факторов, гипотез) на основе данных, которые имеются в распоряжении.

На их базе получился список из 8 таргетных областей (аспектов) в складской и товароведческой деятельности компании *Mint Classics*, анализ которых мог бы пролить свет на текущее положение дел, стать основой для рекомендаций по сокращению (реорганизации) запасов, а также помочь в проработке идеи закрытия одного из складов, то есть реализовать заявленные цели проекта:

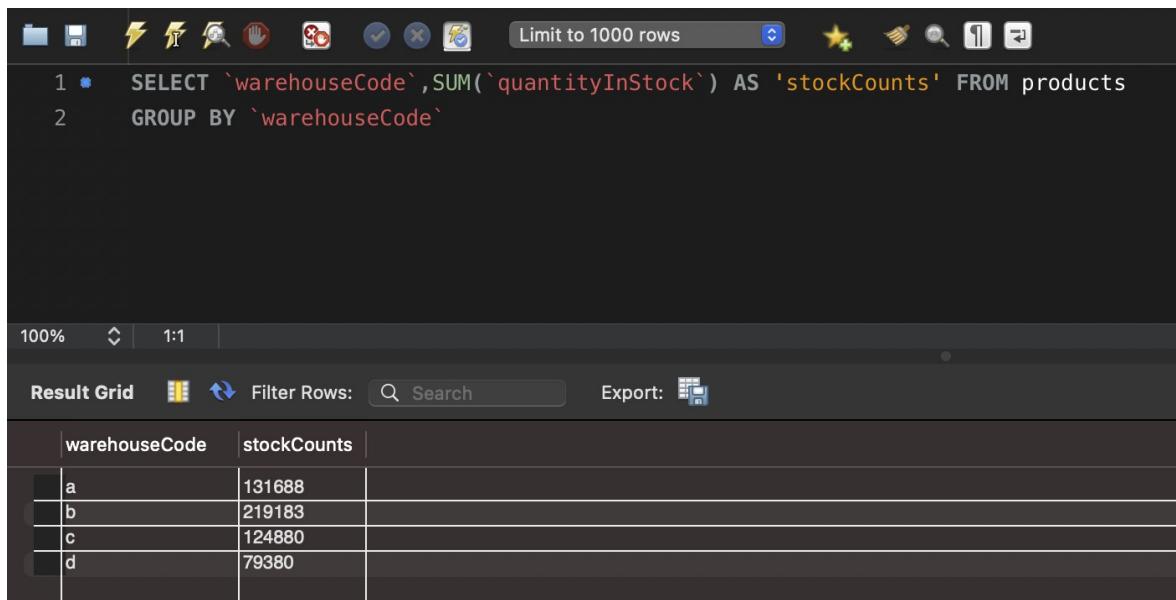
1. Остатки
2. Гипотеза о портфеле заказов
3. Эффективность
4. Клиентоориентированность
5. Оперативность
6. Паттерны роста
7. Оборачиваемость
8. География

Далее я предлагаю обоснование каждой из обозначенных таргетных областей и детализацию проведенного исследования с подкреплением иллюстрациями выполненных SQL-запросов, описанием применяемых техник и промежуточными аналитическими наблюдениями.

1 · Остатки

Сравнение «рейтинга» складов по объему остатков могло бы помочь определить главного претендента на закрытие в случае, если на каком-то из складов их значительно меньше, чем на других.

Выполнил запрос для определения количества товарных остатков (в количестве товарных единиц) по каждому складу.



The screenshot shows a MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is a query editor window containing the following SQL code:

```
1 *  SELECT `warehouseCode`,SUM(`quantityInStock`) AS 'stockCounts' FROM products
2   GROUP BY `warehouseCode`
```

Below the query editor is a results grid titled "Result Grid". The results are displayed in a table with two columns: "warehouseCode" and "stockCounts". The data is as follows:

warehouseCode	stockCounts
a	131688
b	219183
c	124880
d	79380

Наблюдение: по минимальному числу остатков лидирует склад «d» - 79380 единиц, что почти вдвое меньше следующего места в «рейтинге». Больше всего остатков – на складе «b» - 219183 единицы.

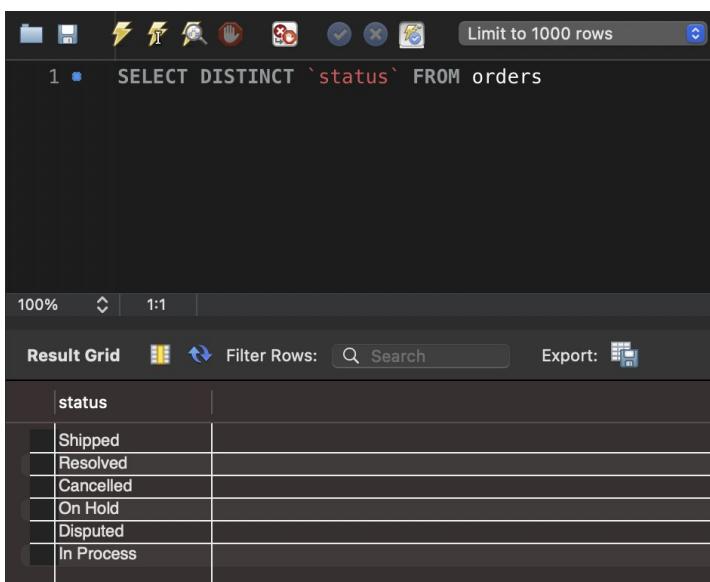
2 · Гипотеза о портфеле заказов

Сформулировал гипотезу (предположение):

“портфель открытых заказов таков, что если просчитать объем остатков по каждому складу (в количестве товарных единиц) на момент исполнения всех текущих заказов, то (возможно) окажется, что на каком-то конкретном складе будет так мало остатков, что именно его будет закрыть проще и экономичнее всего”.

Для проверки гипотезы выполнил следующие шаги:

- a.** Выявил, какие бывают статусы открытых заказов в системе.



The screenshot shows a MySQL Workbench interface with a query editor and a results grid. The query is:

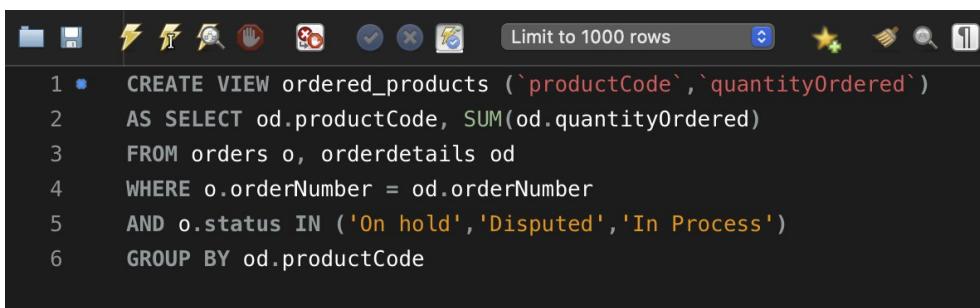
```
1 •  SELECT DISTINCT `status` FROM orders
```

The results grid displays the following data:

status
Shipped
Resolved
Cancelled
On Hold
Disputed
In Process

Наблюдение: таких статусов получилось три: «On hold», «Disputed» и «In process».

- b.** Создал представление со списком артикулов и количествами заказанных единиц каждого артикула из всех открытых на текущий момент заказов.



The screenshot shows a MySQL Workbench interface with a query editor and a results grid. The query is:

```
1 •  CREATE VIEW ordered_products (`productCode`, `quantityOrdered`)
2     AS SELECT od.productCode, SUM(od.quantityOrdered)
3     FROM orders o, orderdetails od
4     WHERE o.orderNumber = od.orderNumber
5     AND o.status IN ('On hold', 'Disputed', 'In Process')
6     GROUP BY od.productCode
```

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_3101	48

с. Создал представление с распределением текущих остатков товаров по всем складам в формате сводной таблицы.

```

CREATE VIEW products_by_warehouses (`productCode`, `a`, `b`, `c`, `d`)
AS SELECT `productCode`,
SUM(CASE WHEN `warehouseCode` = 'a' THEN `quantityInStock` ELSE 0 END),
SUM(CASE WHEN `warehouseCode` = 'b' THEN `quantityInStock` ELSE 0 END),
SUM(CASE WHEN `warehouseCode` = 'c' THEN `quantityInStock` ELSE 0 END),
SUM(CASE WHEN `warehouseCode` = 'd' THEN `quantityInStock` ELSE 0 END)
FROM products
GROUP BY `productCode`
ORDER BY `productCode`
```

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	

Наблюдение: на данном этапе выяснился интересный факт о том, что каждый отдельно взятый артикул хранится исключительно на каком-то одном из складов.

d. Объединил два (созданных ранее) представления для удобного просмотра распределения остатков по складам и всех заказанных артикулов в рамках одной таблицы. При этом в остатках теперь отображаются только те артикулы, которые присутствуют в открытых заказах.

```

1 • CREATE VIEW orders_and_stock (`productCode`, `quantityOrdered`, `a`, `b`, `c`, `d`)
2   AS SELECT ordered_products.*,
3             products_by_warehouses.a,
3             products_by_warehouses.b,
3             products_by_warehouses.c,
3             products_by_warehouses.d
4   FROM ordered_products
5   LEFT JOIN products_by_warehouses
6     ON ordered_products.productCode = products_by_warehouses.productCode;

```

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. Создал еще одно представление, в котором показал, сколько и каких товарных артикулов из всех открытых заказов находится на каждом из складов.

```

CREATE VIEW unit_counts_fulfilled_by_abcd (`productCode`, `a_fulfilled`, `b_fulfilled`, `c_fulfilled`, `d_fulfilled`)
AS SELECT `productCode`,
CASE WHEN (`a` - `quantityOrdered`) > 0 THEN `quantityOrdered`
ELSE `a`
END,
CASE WHEN (`b` - `quantityOrdered`) > 0 THEN `quantityOrdered`
ELSE `b`
END,
CASE WHEN (`c` - `quantityOrdered`) > 0 THEN `quantityOrdered`
ELSE `c`
END,
CASE WHEN (`d` - `quantityOrdered`) > 0 THEN `quantityOrdered`
ELSE `d`
END
FROM `orders_and_stock`
```

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. Рассчитал объем остатков по каждому складу (в количестве товарных единиц) на момент исполнения всех открытых на текущий момент заказов.

```

1 •   SELECT
2     t2.a - t1.a AS a_final_stock,
3     t2.b - t1.b AS b_final_stock,
4     t2.c - t1.c AS c_final_stock,
5     t2.d - t1.d AS d_final_stock
6   FROM
7   (SELECT
8     SUM(`a_fulfilled`) AS `a`,
9     SUM(`b_fulfilled`) AS `b`,
10    SUM(`c_fulfilled`) AS `c`,
11    SUM(`d_fulfilled`) AS `d`
12   FROM `unit_counts_fulfilled_by_abcd` ) t1,
13   (SELECT
14     SUM(CASE WHEN `warehouseCode` = 'a' THEN `quantityInStock` ELSE 0 END) AS `a` ,
15     SUM(CASE WHEN `warehouseCode` = 'b' THEN `quantityInStock` ELSE 0 END) AS `b` ,
16     SUM(CASE WHEN `warehouseCode` = 'c' THEN `quantityInStock` ELSE 0 END) AS `c` ,
17     SUM(CASE WHEN `warehouseCode` = 'd' THEN `quantityInStock` ELSE 0 END) AS `d`
18   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

Наблюдение: в целом, гипотеза не подтвердилась, поскольку остатков по-прежнему будет очень много на каждом из складов.

В то же время выяснилось, что на момент завершения всех открытых заказов по минимальному количеству остатков по-прежнему будет лидировать склад «d» - 78431 единица, а по максимальному - склад «b» - 217903 единицы.

3 · Эффективность

В контексте проработки возможности закрытия одного из складских помещений, немаловажным является сравнение производительности каждого склада на предмет выявления очевидного «аутсайдера».

Сформулировал 6 различных метрик эффективности на базе всей истории успешно выполненных заказов:

- Количество успешно завершенных заказов
- Количество реализованных товарных единиц
- Суммарная выручка
- Суммарная прибыль
- Средняя маржинальность товаров
- Средняя прибыль на единицу товара

Примечание: поскольку один заказ может включать в себя товары, находящиеся на разных складах, то в целях анализа (здесь и далее) за один заказ принимается каждая отдельная строка с товарным артикулом внутри каждого идентификатора заказа.

a. Предварительно отфильтровав только завершенные заказы на основании статуса и уместив их в CTE-выражение, рассчитал вышеуказанные метрики для каждого склада.

```
1 WITH real_orders AS (
2     SELECT `orderNumber` FROM orders
3     WHERE status IN ('Shipped', 'Resolved')
4 )
5
6     SELECT p.warehouseCode,
7         COUNT(od.productCode) AS order_counts,
8         FORMAT(SUM(od.quantityOrdered), 0) AS units_sold,
9         FORMAT(SUM(od.quantityOrdered * od.priceEach), 2) AS total_revenue,
10        FORMAT(SUM(od.quantityOrdered * (od.priceEach - p.buyPrice)), 2) AS total_profit,
11        CONCAT(FORMAT(SUM(od.quantityOrdered * (od.priceEach - p.buyPrice)) /
12                     SUM(od.quantityOrdered * od.priceEach) * 100, 2), '%') AS avg_profitability,
13        FORMAT(SUM(od.quantityOrdered * (od.priceEach - p.buyPrice)) /
14                     SUM(od.quantityOrdered), 2) AS avg_profit_per_unit
15     FROM orderdetails od
16     INNER JOIN products p ON od.productCode = p.productCode
17     INNER JOIN real_orders ro ON ro.orderNumber = od.orderNumber
18     GROUP BY `warehouseCode`
19     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

б. В дополнение, на основе СТЕ-выражения с таблицей из предыдущего запроса, для каждого склада я вычислил процентное отклонение от среднего значения по каждой метрике для более детального статистического анализа полученных результатов.

```

21   SELECT warehouseCode,
22   CONCAT(FORMAT((order_counts - (SELECT AVG(order_counts) FROM metriks)) / (SELECT AVG(order_counts) FROM metriks) * 100, 2),
23   "%") AS dev_mean_order_counts,
24   CONCAT(FORMAT((units_sold - (SELECT AVG(units_sold) FROM metriks)) / (SELECT AVG(units_sold) FROM metriks) * 100, 2), "%")
25   AS dev_mean_units_sold,
26   CONCAT(FORMAT((total_revenue - (SELECT AVG(total_revenue) FROM metriks)) / (SELECT AVG(total_revenue) FROM metriks) * 100,
27   2), "%") AS dev_mean_total_revenue,
28   CONCAT(FORMAT((total_profit - (SELECT AVG(total_profit) FROM metriks)) / (SELECT AVG(total_profit) FROM metriks) * 100, 2),
29   "%") AS dev_mean_total_profit,
30   CONCAT(FORMAT((avg_profitability - (SELECT AVG(avg_profitability) FROM metriks)) / (SELECT AVG(avg_profitability)
31   FROM metriks) * 100, 2), "%") AS dev_mean_avg_profitability,
32   CONCAT(FORMAT((avg_profit_per_unit - (SELECT AVG(avg_profit_per_unit) FROM metriks)) / (SELECT AVG(avg_profit_per_unit)
33   FROM metriks) * 100, 2), "%") AS dev_mean_avg_profit_per_unit
34   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%

```

Наблюдение: обнаружить тенденцию какого-либо из складов сильно уступать другим в эффективности (по одной или нескольким метрикам) не удалось.

Напротив, выяснилось, что один из складов - «*б*» - существенно превосходит остальные по пяти метрикам из шести. И даже по той метрике, по которой он уступает складам «*а*» и «*с*» (средняя маржинальность товаров) отклонение от среднего значения не превышает 0.81%.

В то время как по остальным пятью метрикам склад «*б*» показывает отклонение от 21.44% до 62.19% в положительную сторону. В частности, он уверенно лидирует по размеру выручки и суммарной прибыли, а также опережает все другие склады по прибыли на единицу реализованного товара.

Вынося за скобки явно преуспевающий на общем фоне склад «*б*», доход ощутимо выше среди остальных показывает склад «*а*», вместе с тем по уровню маржинальности товаров он превосходит среднее значение на 1.03%.

Интересно также отметить, что лидером по маржинальности товаров является склад «*с*» - 41.02%, демонстрируя при этом наименьший показатель выручки.

В абсолютных величинах «анти-лидерами» по каждой изученной метрике являются следующие складские помещения:

- Наименьшее количество завершенных заказов - у склада «*д*» - 586
- Наименьшее количество единиц реализованных товаров - у склада «*д*» - 20622
- Наименьшая суммарная выручка - у склада «*с*» - 1,669,114.08
- Наименьшая суммарная прибыль - у склада «*д*» - 671,675.80
- Наименьшая средняя маржинальность - у склада «*д*» - 38.83%
- Наименьшая средняя прибыль на единицу товара - у склада «*с*» - 32.10

4 · Клиентоориентированность

Как известно, размещенный заказ – это хорошо, а что еще лучше – это успешная продажа, которая в большинстве случаев подразумевает довольного покупателя.

В этом смысле я решил посчитать уровень *Order Success Rate* для каждого склада, то есть отношение количества успешно выполненных заказов к их общему количеству.

Сгруппировав все в истории заказы на выполненные и отмененные, затем посчитав их количество в каждой группе, вычислил долю успешных заказов в портфеле каждого склада.

The screenshot shows a database query editor interface with a dark theme. At the top, there are various icons for file operations, search, and export. A toolbar includes a 'Limit to 1000 rows' button. Below the toolbar is the SQL query itself, numbered from 1 to 30. The query uses common table expressions (CTEs) to calculate the success rate for four warehouses (a, b, c, d). The results are displayed in a grid at the bottom.

```
1 • WITH successful_orders AS (
2     SELECT p.warehouseCode, o.status, od.productCode
3     FROM orders o
4     INNER JOIN orderdetails od ON o.orderNumber = od.orderNumber
5     INNER JOIN products p ON od.productCode = p.productCode
6     WHERE o.status IN ('Shipped', 'Resolved')
7 ),
8     cancelled_orders AS (
9         SELECT p.warehouseCode, o.status, od.productCode
10        FROM orders o
11       INNER JOIN orderdetails od ON o.orderNumber = od.orderNumber
12       INNER JOIN products p ON od.productCode = p.productCode
13      WHERE o.status IN ('Cancelled')
14 ),
15     successful_counts AS (
16         SELECT warehouseCode, COUNT(status) AS counts
17         FROM successful_orders
18        GROUP BY warehouseCode
19 ),
20     cancelled_counts AS (
21         SELECT warehouseCode, COUNT(status) AS counts
22         FROM cancelled_orders
23        GROUP BY warehouseCode
24 )
25
26     SELECT sc.warehouseCode,
27     CONCAT(FORMAT(sc.counts / (sc.counts + cc.counts) * 100, 2), '%') AS order_success_rate
28     FROM successful_counts sc
29     INNER JOIN cancelled_counts cc ON sc.warehouseCode = cc.warehouseCode
30     ORDER BY warehouseCode
```

Result Grid Filter Rows: Export:

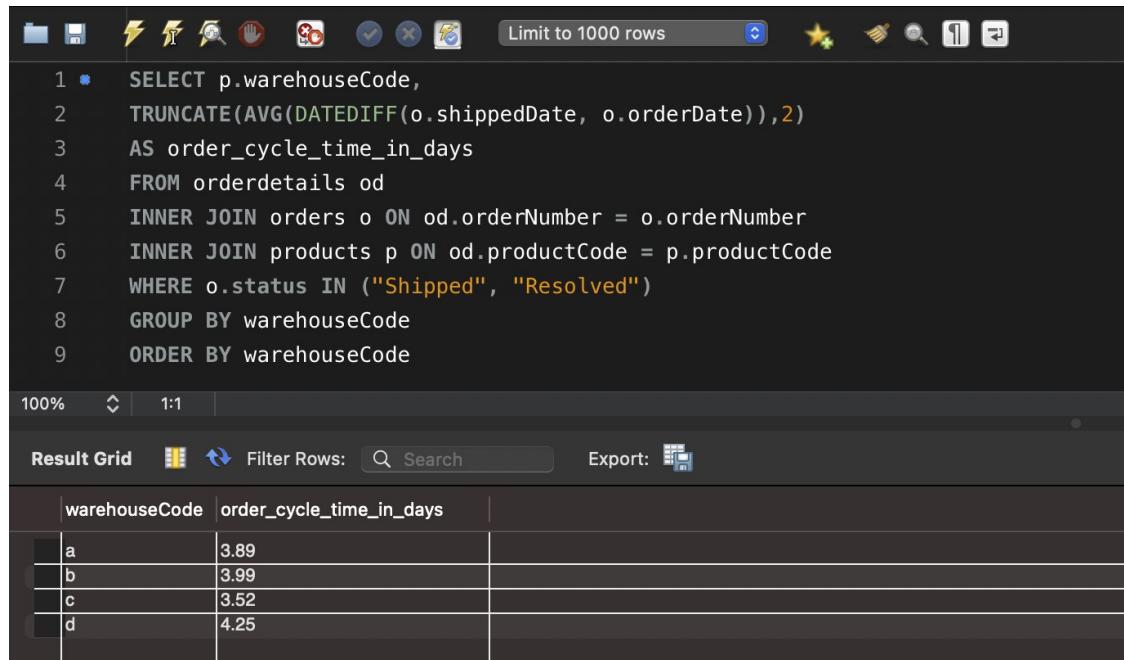
warehouseCode	order_success_rate
a	97.19%
b	97.86%
c	97.46%
d	96.22%

Наблюдение: есть некоторая разница в показателях метрики по складам, но границы диапазона не выглядят экстремальными – от 96.22% у склада «d» до 97.86% у склада «b».

5 · Оперативность

Оперативность работы склада является неотъемлемой составляющей и качества обслуживания, то есть клиентоориентированности, и общей эффективности компании.

В своем исследовании я поместил этот аспект в отдельный пункт и решил посчитать метрику *Order Cycle Time* по складам в рамках всей истории выполненных заказов, то есть среднее время выполнения заказа (в количестве дней) от размещения его покупателем до финальной отгрузки товара.



The screenshot shows a MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is a query editor window containing the following SQL code:

```
1 •  SELECT p.warehouseCode,
2      TRUNCATE(AVG(DATEDIFF(o.shippedDate, o.orderDate)),2)
3      AS order_cycle_time_in_days
4  FROM orderdetails od
5  INNER JOIN orders o ON od.orderNumber = o.orderNumber
6  INNER JOIN products p ON od.productCode = p.productCode
7  WHERE o.status IN ("Shipped", "Resolved")
8  GROUP BY warehouseCode
9  ORDER BY warehouseCode
```

Below the code, the results are displayed in a grid:

warehouseCode	order_cycle_time_in_days
a	3.89
b	3.99
c	3.52
d	4.25

Наблюдение: по данной метрике чемпионом является склад «с», который в среднем отгружает заказ через три с половиной дня.

Самое долгое среднее время обработки заказов – у склада «д» - 4.25 дня и, вместе с тем, ни на одном другом складе значение метрики не превышает уровень в 4 дня.

6 · Паттерны роста

В своей работе над проектом я посчитал важным проверить наличие позитивных тенденций на каком-либо из складов, которые могут быть скрыты за сиюминутной картиной статистических показателей.

- a.** На первом этапе я создал представление с историей всех успешно выполненных заказов, включив в таблицу дату заказа, код склада, товарную корзину и подсчет выручки.

The screenshot shows a database interface with a SQL editor at the top containing the following code:

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

Below the code, the interface displays a result grid with the following columns: warehouseCode, orderDate, productCode, quantityOrdered, priceEach, and revenue. The data consists of approximately 40 rows of order details. At the bottom of the grid, it says "all_shipped_orders 1".

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

- b.** Используя СТЕ-выражение, сгруппировал таблицу по складам и месяцам (год за годом в хронологическом порядке), вычислил количество заказов по каждому складу в каждом месяце истории и представил данные в формате сводной таблицы.

```

1 • WITH orders_grouped AS (
2     SELECT warehouseCode, DATE_FORMAT(orderDate, '%Y-%m') AS mo_year, COUNT(*) AS order_counts
3     FROM all_shipped_orders
4     GROUP BY warehouseCode, mo_year
5 )
6
7     SELECT mo_year,
8     SUM(CASE WHEN warehouseCode = "a" THEN order_counts ELSE 0 END) AS `a`,
9     SUM(CASE WHEN warehouseCode = "b" THEN order_counts ELSE 0 END) AS `b`,
10    SUM(CASE WHEN warehouseCode = "c" THEN order_counts ELSE 0 END) AS `c`,
11    SUM(CASE WHEN warehouseCode = "d" THEN order_counts ELSE 0 END) AS `d`
12    FROM orders_grouped
13    GROUP BY mo_year
14    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

Наблюдение: из полученной таблицы видно, что количество заказов сильно разнится от месяца к месяцу (внутри года), так что полученные данные могли бы лежать в основу дополнительного исследования сезонности продаж, если будет поставлена такая задача.

с. Поскольку вся история заказов покрывает период от января 2003 до мая 2005, то есть возможность отобразить ежемесячные тренды от года к году, начиная с января 2004, то есть с первого повторяющегося в данных месяца.

Поместив полученную в предыдущем пункте сводную таблицу в СТЕ-выражение, я сделал запрос к ней с помощью оконной функции, чтобы отобразить тренд изменения количества заказов в каждом месяце от года к году.

```

1 • Ⓛ WITH orders_grouped AS (
2     SELECT warehouseCode, DATE_FORMAT(orderDate, '%Y-%m') AS mo_year, COUNT(*) AS order_counts
3     FROM all_shipped_orders
4     GROUP BY warehouseCode, mo_year
5 ),
6 Ⓛ orders_pivot AS (
7     SELECT mo_year,
8         SUM(CASE WHEN warehouseCode = "a" THEN order_counts ELSE 0 END) AS `a`,
9         SUM(CASE WHEN warehouseCode = "b" THEN order_counts ELSE 0 END) AS `b`,
10        SUM(CASE WHEN warehouseCode = "c" THEN order_counts ELSE 0 END) AS `c`,
11        SUM(CASE WHEN warehouseCode = "d" THEN order_counts ELSE 0 END) AS `d`
12     FROM orders_grouped
13     GROUP BY mo_year
14 )
15     SELECT mo_year,
16     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,
17     CONCAT(TRUNCATE((b - LAG(b,12) OVER (ORDER BY mo_year)) / LAG(b,12,0) OVER (ORDER BY mo_year) * 100, 2), "%") AS b_trend,
18     CONCAT(TRUNCATE((c - LAG(c,12) OVER (ORDER BY mo_year)) / LAG(c,12,0) OVER (ORDER BY mo_year) * 100, 2), "%") AS c_trend,
19     CONCAT(TRUNCATE((d - LAG(d,12) OVER (ORDER BY mo_year)) / LAG(d,12,0) OVER (ORDER BY mo_year) * 100, 2), "%") AS d_trend
20    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	HULL	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%	HULL
2005-05	-32.00%	31.25%	-440.00%	-85.71%

д. Повторив алгоритм действий из предыдущих двух пунктов, аналогичным образом посчитал статистику изменения выручки в каждом месяце от года к году.

```

6 Ⓛ orders_pivot AS (
7     SELECT mo_year,
8         SUM(CASE WHEN warehouseCode = "a" THEN revenue ELSE 0 END) AS `a`,
9         SUM(CASE WHEN warehouseCode = "b" THEN revenue ELSE 0 END) AS `b`,
10        SUM(CASE WHEN warehouseCode = "c" THEN revenue ELSE 0 END) AS `c`,
11        SUM(CASE WHEN warehouseCode = "d" THEN revenue ELSE 0 END) AS `d`
12     FROM orders_grouped
13     GROUP BY mo_year
14 )
15     SELECT mo_year,
16     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,
17     CONCAT(TRUNCATE((b - LAG(b,12) OVER (ORDER BY mo_year)) / LAG(b,12,0) OVER (ORDER BY mo_year) * 100, 2), "%") AS b_trend,
18     CONCAT(TRUNCATE((c - LAG(c,12) OVER (ORDER BY mo_year)) / LAG(c,12,0) OVER (ORDER BY mo_year) * 100, 2), "%") AS c_trend,
19     CONCAT(TRUNCATE((d - LAG(d,12) OVER (ORDER BY mo_year)) / LAG(d,12,0) OVER (ORDER BY mo_year) * 100, 2), "%") AS d_trend
20    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	HULL	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%	HULL
2005-05	-44.12%	49.02%	-638.81%	-94.47%

Наблюдение: просматривая полученные данные, можно отметить, что за исключением редких отрицательных всплесков, все склады показывают уверенный рост активности.

Наибольшее проседание по количеству заказов показывает склад «с» в мае 2004 - на 440%, максимальный положительный всплеск – у склада «а» в сентябре 2004 - на 2400%.

Пики колебания выручки совпадают с экстремумами по количеству заказов: падение продаж на складе «с» в мае 2004 на 638% и рост на складе «а» в сентябре 2004 на 2286%.

Отдельно можно отметить склад «б» как единственный, который не показывает отрицательного роста выручки ни по одному месяцу в рассматриваемом периоде.

e. Располагая в истории всего одним, но все же целым повторяющимся календарным годом (2004), я вычислил годовой тренд изменения количества заказов, то есть как изменилось количество заказов (на каждом складе) в 2004 году по отношению к 2003 году.

```

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. Аналогичным образом вычислил годовой тренд изменения выручки по каждому складу.

```

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%

Наблюдение: статистика за годовой период подтверждает сильный восходящий тренд по обоим показателям и по всем складским помещениям.

Лидером и по росту выручки, и по росту количества заказов оказался склад «а» - 54.45% и 53.49% соответственно, и даже «анти-лидер» в каждом рейтинге – склад «б» - все равно обработал на 17.96% больше заявок, общей стоимостью на 22.9% выше, чем годом ранее.

Можно также отметить, что для каждого склада наблюдается четкая корреляция между увеличением количества заказов и ростом выручки.

g. Статистика за самый «свежий» год в истории заказов (2005) охватывает 5 месяцев с января по май, а данные этого же периода имеются также за 2003 и 2004 годы, таким образом есть возможность провести анализ изменения выручки и количества заказов за 5-месячный отрезок года сразу по нескольким отчетным периодам.

Пользуясь таблицей из предыдущих запросов, я произвел фильтрацию данных, чтобы оставить только месяцы с января по май в рамках каждого года, и затем посчитал, как изменилось количество обработанных каждым складом заказов за первые 5 месяцев 2004 года по отношению к 2003 году, и за первые 5 месяцев 2005 года относительно 2004 года.

```
1 • ⊖ WITH orders_grouped AS (
2     SELECT warehouseCode, DATE_FORMAT(orderDate, '%Y-%m') AS mo_year, COUNT(*) AS order_counts
3     FROM all_shipped_orders
4     GROUP BY warehouseCode, mo_year
5 ),
6 ⊖ orders_pivot AS (
7     SELECT mo_year,
8         SUM(CASE WHEN warehouseCode = "a" THEN order_counts ELSE 0 END) AS `a`,
9         SUM(CASE WHEN warehouseCode = "b" THEN order_counts ELSE 0 END) AS `b`,
10        SUM(CASE WHEN warehouseCode = "c" THEN order_counts ELSE 0 END) AS `c`,
11        SUM(CASE WHEN warehouseCode = "d" THEN order_counts ELSE 0 END) AS `d`
12     FROM orders_grouped
13     GROUP BY mo_year
14 ),
15 ⊖ orders_pivot_filtered AS (
16     SELECT (CASE WHEN mo_year LIKE "%2003%" THEN 2003 WHEN mo_year LIKE "%2004%" THEN 2004
17     WHEN mo_year LIKE "%2005%" THEN 2005 END) AS period,
18     a,b,c,d
19     FROM orders_pivot
20     WHERE mo_year LIKE "%01" OR mo_year LIKE "%02" OR mo_year LIKE "%03"
21     OR mo_year LIKE "%04" OR mo_year LIKE "%05"
22 ),
23 ⊖ order_counts_per_year AS (SELECT period, SUM(a) AS a, SUM(b) AS b, SUM(c) AS c, SUM(d) AS d
24     FROM orders_pivot_filtered GROUP BY period)
25
26     SELECT CONCAT(FORMAT((LEAD(a) OVER (ORDER BY period) - a) / a * 100, 2), "%") AS a_trend,
27     CONCAT(FORMAT((LEAD(b) OVER (ORDER BY period) - b) / b * 100, 2), "%") AS b_trend,
28     CONCAT(FORMAT((LEAD(c) OVER (ORDER BY period) - c) / c * 100, 2), "%") AS c_trend,
29     CONCAT(FORMAT((LEAD(d) OVER (ORDER BY period) - d) / d * 100, 2), "%") AS d_trend
30     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. Аналогичным образом измерил тренд изменения выручки в январе – мае от года к году.

```

15   ⊖ orders_pivot_filtered AS (
16     ⊖ SELECT (CASE WHEN mo_year LIKE "%2003%" THEN 2003 WHEN mo_year LIKE "%2004%" THEN 2004
17       WHEN mo_year LIKE "%2005%" THEN 2005 END) AS period,
18       a,b,c,d
19     FROM orders_pivot
20     WHERE mo_year LIKE "%01" OR mo_year LIKE "%02" OR mo_year LIKE "%03"
21     OR mo_year LIKE "%04" OR mo_year LIKE "%05"
22   ),
23   ⊖ rev_per_year AS (SELECT period, SUM(a) AS a, SUM(b) AS b, SUM(c) AS c, SUM(d) AS d
24     FROM orders_pivot_filtered GROUP BY period)
25
26   SELECT CONCAT(FORMAT((LEAD(a) OVER (ORDER BY period) - a) / a * 100, 2), "%") AS a_trend,
27   CONCAT(FORMAT((LEAD(b) OVER (ORDER BY period) - b) / b * 100, 2), "%") AS b_trend,
28   CONCAT(FORMAT((LEAD(c) OVER (ORDER BY period) - c) / c * 100, 2), "%") AS c_trend,
29   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%

Наблюдение: в то время как 2004 год был явно прорывным, 2005 год приносит более скромный рост, однако общая тенденция, очевидно, работает одинаково для всех складов и нуждается в отдельном анализе ее предпосылок.

Если же мы сравниваем непосредственно ситуацию на каждом отдельном складе внутри общей тенденции, то лидер по росту выручки в 2004 году – склад «а» - увеличил этот показатель более чем вдвое – на 106.65%, а минимальный рост показал склад «с» - на 23.71%.

В 2005 году разница в показателях роста выручки по складам уже не такая значительная – от 24.8% у склада «б» до 13.85% у склада «с», который, надо отметить, лидирует по минимумам в обоих рассмотренных отчетных периодах.

Как и в случае с годовыми показателями роста, анализ 5-месячных отрезков год к году показывает сильную положительную корреляцию между ростом количества заказов и ростом выручки.

7 · Оборачиваемость

В данном пункте исследования я сфокусировал внимание на свойствах товаров, которыми занимается каждый из складов. Посчитал оборачиваемость товарных позиций в рамках всей истории выполненных заказов и коэффициент запасов до исчерпания по каждому товарному артикулу.

a. Сгруппировал в CTE-выражении текущие остатки и общее количество проданных единиц по каждому товарному артикулу, далее рассчитал среднемесячную оборачиваемость на основании количества месяцев в истории (29) и коэффициент запасов до исчерпания - то есть, на сколько месяцев хватит текущих запасов каждого артикула, если средний уровень продаж останется константой.

```
1  CREATE VIEW product_ratios AS
2
3  ⊕ WITH stock_vs_turnover AS (
4    SELECT p.productCode,
5      (SELECT warehouseCode FROM products WHERE productCode = p.productCode) AS warehouse,
6      (SELECT quantityInStock FROM products WHERE productCode = p.productCode) AS stock,
7      SUM(pd.quantityOrdered) AS ordered_units_total
8    FROM products p
9    INNER JOIN orderdetails pd ON p.productCode = pd.productCode
10   GROUP BY productCode
11  ),
12
13  ⊕ stock_vs_turnover_updated AS (
14    SELECT *,
15      TRUNCATE(ordered_units_total / 29, 1) AS avg_monthly_turnover
16    FROM stock_vs_turnover
17  )
18
19  SELECT *,
20  TRUNCATE(stock / avg_monthly_turnover, 1) AS inventory_ratio_in_months
21  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			

Наблюдение: налицо весьма однородная картина по среднему количеству ежемесячных продаж и одновременно огромный разброс по коэффициенту запасов до исчерпания.

b. Посчитал минимальное, максимальное и среднее значение среднемесячной оборачиваемости по всем товарным позициям.

```

1 •  SELECT MIN(avg_monthly_turnover),
2     AVG(avg_monthly_turnover),
3     MAX(avg_monthly_turnover)
4   FROM product_ratios

```

Result Grid | Filter Rows: | Search | Export:

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

Наблюдение: подтвердилась достаточно гладкая статистика продаж по ассортименту.

Несмотря на разницу между минимальным и максимальным значением почти в три раза, этот диапазон представляется нормальным для такой специфической категории товаров, как транспортные средства, где существует палитра разных ценовых категорий.

Отсутствуют позиции, которые не продавались бы вовсе или в количестве всего нескольких штук, что говорит о грамотной подборке ассортимента.

с. Посчитал количество товарных артикулов со среднемесячной обрачиваемостью меньшей или равной среднему значению и отобразил их распределение по складам, сгруппировав по товарным категориям.

```

1 •  WITH ratios_less_avg AS (
2     SELECT pr.warehouse, p.productLine, COUNT(pr.productCode) AS product_counts
3     FROM product_ratios pr
4     INNER JOIN products p ON pr.productCode = p.productCode
5     WHERE pr.avg_monthly_turnover <= (SELECT AVG(avg_monthly_turnover) FROM product_ratios)
6     GROUP BY warehouse, productLine
7   )
8
9   SELECT productLine,
10    CASE WHEN warehouse = "a" THEN product_counts ELSE 0 END AS `a`,
11    CASE WHEN warehouse = "b" THEN product_counts ELSE 0 END AS `b`,
12    CASE WHEN warehouse = "c" THEN product_counts ELSE 0 END AS `c`,
13    CASE WHEN warehouse = "d" THEN product_counts ELSE 0 END AS `d`
14  FROM ratios_less_avg
15  ORDER BY productLine

```

Result Grid | Filter Rows: | Search | Export:

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

д. Посчитал количество товарных артикулов с ежемесячной обрачиваемостью выше среднего значения и отобразил их распределение по складам, сгруппировав по товарным категориям.

```

1 • Ⓛ WITH ratios_higher_avg AS (
2     SELECT pr.warehouse, p.productLine, COUNT(pr.productCode) AS product_counts
3     FROM product_ratios pr
4     INNER JOIN products p ON pr.productCode = p.productCode
5     WHERE pr.avg_monthly_turnover > (SELECT AVG(avg_monthly_turnover) FROM product_ratios)
6     GROUP BY warehouse, productLine
7 )
8
9     SELECT productLine,
10    CASE WHEN warehouse = "a" THEN product_counts ELSE 0 END AS `a`,
11    CASE WHEN warehouse = "b" THEN product_counts ELSE 0 END AS `b`,
12    CASE WHEN warehouse = "c" THEN product_counts ELSE 0 END AS `c`,
13    CASE WHEN warehouse = "d" THEN product_counts ELSE 0 END AS `d`
14   FROM ratios_higher_avg
15   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. Приняв нормальный для отрасли срок в 6 месяцев за стандартное значение коэффициента запасов до исчерпания, вычислил количество товарных артикулов с запасами в пределах стандартного значения.

```

1 •   SELECT * FROM product_ratios
2      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

Наблюдение: выяснилось, что таких артикулов лишь 4.

f. Вычислил количество товарных артикулов с уровнем запасов выше стандартного значения.

```

1 •   SELECT COUNT(*) FROM product_ratios
2     WHERE inventory_ratio_in_months > 6

```

Result Grid | Filter Rows: | Search | Export:

COUNT(*)
105

Наблюдение: выяснилось, что таких артикулов 105.

g. Отобразил распределение товарных артикулов с повышенным уровнем запасов (в количестве товарных единиц) по всем складам в формате сводной таблицы с группировкой по товарным категориям.

```

1 • ⊖ WITH overstock_map AS (
2     SELECT pr.warehouse, p.productLine, SUM(pr.stock) AS unit_counts
3     FROM product_ratios pr
4     INNER JOIN products p ON pr.productCode = p.productCode
5     WHERE pr.inventory_ratio_in_months > 6
6     GROUP BY warehouse, productLine
7 )
8
9     SELECT productLine,
10    CASE WHEN warehouse = "a" THEN unit_counts ELSE 0 END AS `a`,
11    CASE WHEN warehouse = "b" THEN unit_counts ELSE 0 END AS `b`,
12    CASE WHEN warehouse = "c" THEN unit_counts ELSE 0 END AS `c`,
13    CASE WHEN warehouse = "d" THEN unit_counts ELSE 0 END AS `d`
14    FROM overstock_map
15    ORDER BY productLine

```

Result Grid | Filter Rows: | Search | Export:

productLine	a	b	c	d
Classic Cars	0	211382	0	0
Motorcycles	69208	0	0	0
Planes	62287	0	0	0
Ships	0	0	0	26833
Trains	0	0	0	16696
Trucks and Buses	0	0	0	35851
Vintage Cars	0	0	124744	0

Наблюдение: только склады «b» и «c» имеют затоваренные позиции в рамках одной товарной категории, и в обоих случаях это автомобили.

Склады «а» и «д» имеют огромные количества затоваренных артикулов по нескольким, и в обоих случаях - очень громоздким товарным категориям, таким как самолеты, поезда и корабли.

В чисто физическом измерении (габариты, вес, трудоемкость релокации товаров) очевидно наименее загруженным является склад «с».

h. Произвел ранжирование всех товарных артикулов с повышенными запасами по трем уровням затоваренности с использованием оконной функции и отфильтровал список товарных артикулов с наивысшим коэффициентом запасов до исчерпания.

The screenshot shows a database query editor interface with the following details:

Query:

```
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
```

Result Grid:

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 · География

В заключительном пункте исследования я посчитал важным рассмотреть географическое распределение заказов по складам и его соответствие месторасположению складов.

Сгруппировав в CTE-выражении все выполненные заказы по обслуживающим складам и странам назначения, представил сводную таблицу с количеством заказов из каждой страны по каждому складу.

```
1 • ⓕ WITH geography AS (
2     SELECT c.country, CONCAT(p.warehouseCode, " - ", w.warehouseName) AS warehouse,
3     COUNT(od.orderNumber) AS order_counts FROM orderdetails od
4     INNER JOIN products p ON od.productCode = p.productCode
5     INNER JOIN warehouses w ON p.warehouseCode = w.warehouseCode
6     INNER JOIN orders o ON od.orderNumber = o.orderNumber
7     INNER JOIN customers c ON o.customerNumber = c.customerNumber
8     WHERE o.status IN ("Shipped", "Resolved")
9     GROUP BY country, warehouse
10    )
11
12    SELECT country,
13    SUM(CASE WHEN warehouse = "a - North" THEN order_counts ELSE 0 END) AS "a - North",
14    SUM(CASE WHEN warehouse = "b - East" THEN order_counts ELSE 0 END) AS "b - East",
15    SUM(CASE WHEN warehouse = "c - West" THEN order_counts ELSE 0 END) AS "c - West",
16    SUM(CASE WHEN warehouse = "d - South" THEN order_counts ELSE 0 END) AS "d - South"
17    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

Наблюдение: полученную сводную таблицу можно в дальнейшем использовать для более детального изучения географии, например, с группировкой стран по регионам, если будет поставлена такая задача.

Беглый обзор позволяет сразу заметить некоторые несоответствия, например, северный склад «а» обслуживает менее четверти заказов из Канады, а южный склад «д» обрабатывает менее 15% заказов из Новой Зеландии.

Выводы и рекомендации

- Несмотря на то, что склад «d» лидирует по минимальному числу остатков (и казалось бы, в таком случае его закрыть проще других), является самым нерасторопным и проигрывает прочим складам по большинству метрик, в ходе исследования выяснилось, что основные категории товаров, хранящиеся там – это корабли, самолеты и прочие громоздкие транспортные средства, что невилирует и во многом объясняет его отставание, и одновременно делает этот склад анти-лидером в рейтинге кандидатов на закрытие, поскольку релоцировать эти категории товаров на другие склады было бы в высшей степени затратно.

Склад «b» имеет самую высокую доходность на единицу реализованного товара, имеет больше всех остатков (по количеству единиц), обеспечивает самую высокую выручку (и прибыль), а также является лидером роста в течение последних 5 месяцев. Вряд ли стоило бы закрывать самый эффективный склад.

Склад «a» показал самый многообещающий рост в 2004 году, а по 5-месячным периодам его рост оказался выше, чем у склада «c» в обоих отчетных периодах.

Вместе с тем, товарные категории склада «a» – это мотоциклы и самолеты (в частности, 62,287 единиц летательных аппаратов), в то время как склад «c» хранит только автомобили.

Таким образом, в качестве наиболее экономически целесообразного решения я рекомендую рассмотреть ликвидацию склада «c» и одновременную релокацию его товаров, которые являются самыми высокомаржинальными в рамках всего портфолио компании (в среднем 41.02%) на все три оставшихся склада, повысив уровень их рентабельности.

- В случае принятия положительного решения о закрытии склада «c», в целях определения наиболее эффективного порядка и пропорций, в которых будет произведена релокация товаров, рекомендую провести анализ уровня загруженности складов «a», «b» и «d», а также (в продолжение пункта 8. настоящего исследования) анализ географии покупателей всех товаров склада «c», чтобы переместить товары именно на те складские помещения, которые приближены к основному региону спроса на них.
- Рекомендую прекратить закупки любых товаров, по которым обнаружена затоваренность (пункт 7. настоящего исследования), в дальнейшем - наблюдать за ростом продаж, поквартально корректировать расчет коэффициента до исчерпания по каждому артикулу и возобновлять закупки только при достижении нормального значения этого коэффициента.
- Рекомендую устроить распродажу или любое другое активное маркетинговое продвижение всех товаров с самим высоким показателем затоваренности, определенных в пункте 7.h. настоящего исследования.

Навыки

В ходе работы над данным проектом мной продемонстрированы следующие профессиональные компетенции:

- Импорт существующей базы данных с использованием MySQL Workbench
- Ознакомление с бизнесом и его данными путем изучения реляционной модели и исследования таблиц данных в MySQL Workbench
- Анализ данных инвентаризации с использованием SQL-запросов в MySQL Workbench, извлекающих данные из многотабличной реляционной базы данных с использованием SQL-команд, таких как: SELECT, WHERE, GROUP BY, HAVING и ORDER BY
- Создание представлений (View)
- Использование функций агрегации данных и оконных функций SQL
- Использование специальных операторов языка SQL, таких как IN, LIKE и др.
- Создание сводных таблиц с помощью оператора CASE
- Объединение различных таблиц в ходе выполнения SQL-запросов с помощью методов INNER JOIN и OUTER JOIN
- Разработка рекомендаций и предложений для решения бизнес-потребности (проблемы) на основе анализа данных
- Поддержка сформулированных рекомендаций и предложений по сокращению запасов в форме скриптов (запросов) и их результатов

Благодарность

Хотелось бы выразить признательность проекту Coursera за поддержку образовательного процесса и возможность презентации полученных на онлайн-курсах навыков посредством выполнения учебных проектов, в том числе бесплатных, таких как этот.