Analyze Data in a Model Car Database with MySQL Workbench

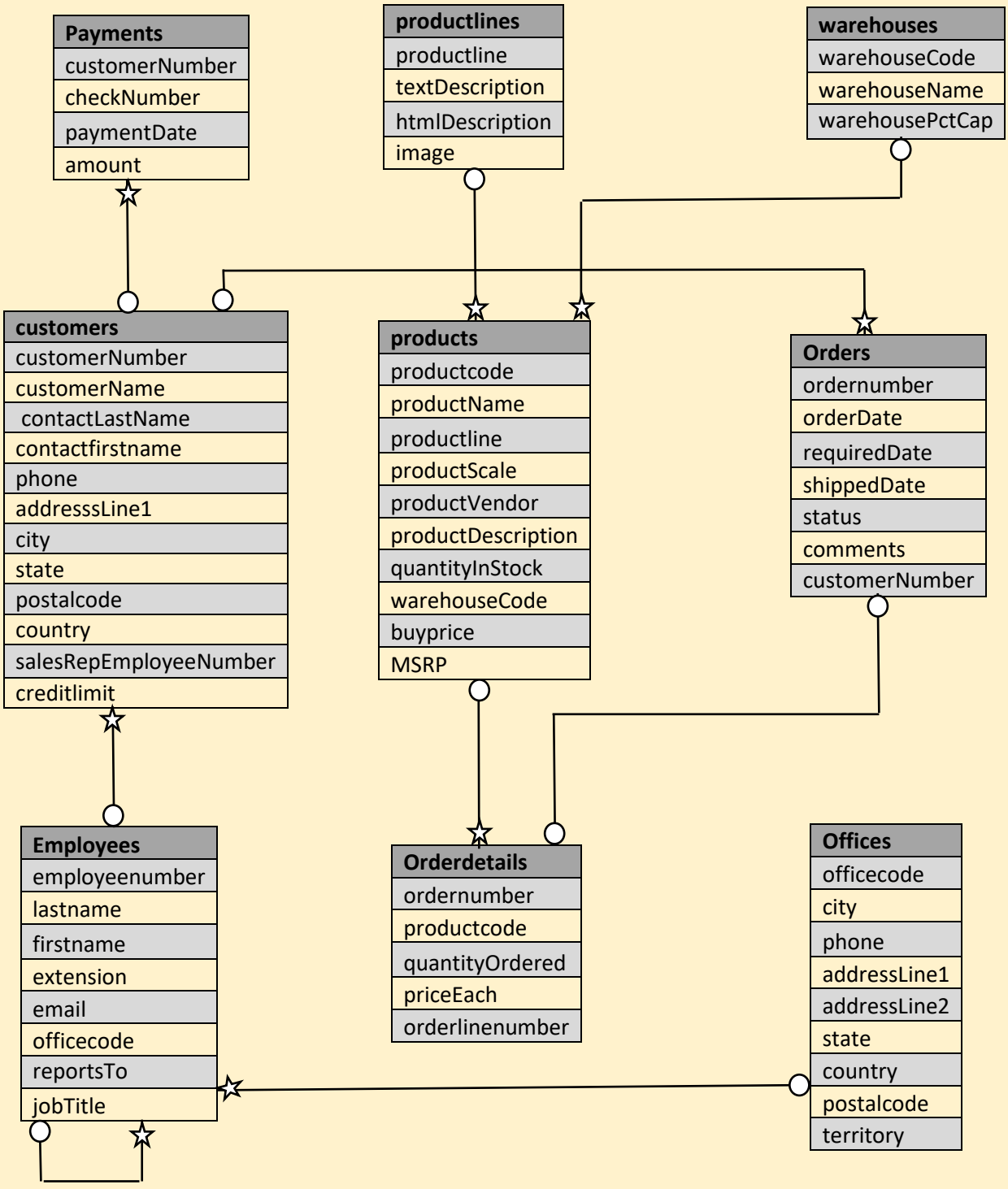
Summary:

In this project, we will conduct a compressive analysis of the data pertaining to classic company, a prominent retailer specializing in model cars. Our objective is to derive valuable insights and make informed decisions based on the available data. The analysis will encompass various aspects of the company's operations and performance, shedding light on its strength, weakness, opportunities and potential threats.

Problem Statement:

The Company is looking to close one of its storage facilities. Out objective is to recommended inventory reduction strategies that won't negatively impact customer service. Using MySQL Workbench, we'll familiarize yourself with the sample database, run SQL Queries to identify factors affecting storage space and propose inventory reduction approaches.

➢ EER(Extended Entity-Relationship) Diagram:-



1. Display total revenue

Query:

SELECT sum(amount) AS ‘total_revenue’ FROM payments;

Output:

total_revenue
8853839.23

2. Display total revenue according to year

Query:

SELECT year(paymentdate) AS 'Year', sum(amount) AS 'total_revenue' FROM payments GROUP BY year(paymentdate);

Output:

Year	total_revenue
2004	4313328.25
2003	3250217.70
2005	1290293.28

3. Display total unique products

Query:

SELECT COUNT(DISTINCT productcode) AS 'total_products' FROM products;

Output:

total_products
110

4. Display total orders according to there delivery status

Query:

select status,count(ordernumber) as'Number_of_orders' from orders group by status;

Output:

status	Number_of_orders
Shipped	303
Resolved	4
Cancelled	6
On Hold	4
Disputed	3
In Process	6

5. Display total number of unique customers

Query:

```
SELECT COUNT(DISTINCT customernumber) AS 'total_customer' from customers;
```

Output:

total_customer
122

6. Display employee wise his position and his head and office code

Query:

```
select e.officeCode,e.firstname as 'Employee_name',m.firstname as 'Manager_name',e.jobTitle from employees e  left join employees m on e.reportsto=m.employeenumber;
```

Output:

officeCode	Employee_name	Manager_name	jobTitle
1	Diane	NULL	President
1	Mary	Diane	VP Sales
1	Jeff	Diane	VP Marketing
1	Anthony	Mary	Sales Manager (NA)
1	Leslie	Anthony	Sales Rep
1	Leslie	Anthony	Sales Rep
2	Julie	Anthony	Sales Rep
2	Steve	Anthony	Sales Rep
3	Foon Yue	Anthony	Sales Rep
3	George	Anthony	Sales Rep
4	Gerard	Mary	Sale Manager (EMEA)
4	Loui	Gerard	Sales Rep

officeCode	Employee_name	Manager_name	jobTitle
4	Gerard	Gerard	Sales Rep
4	Pamela	Gerard	Sales Rep
4	Martin	Gerard	Sales Rep
5	Mami	Mary	Sales Rep
5	Yoshimi	Mami	Sales Rep
6	William	Mary	Sales Manager (APAC)
6	Andy	William	Sales Rep
6	Peter	William	Sales Rep
6	Tom	William	Sales Rep
7	Larry	Gerard	Sales Rep
7	Barry	Gerard	Sales Rep

7. Display total overall profit and profit Percentage

Query:

```
SELECT SUM(pd.buyPrice*od.quantityordered) AS 'Total_buy_price',

SUM(priceeach*quantityordered) as 'total_Selling_price',

(SUM(priceeach*quantityordered)-SUM(pd.buyPrice*od.quantityordered)) AS 'profit',

(SUM(priceeach*quantityordered)-
sum(pd.buyPrice*od.quantityordered))/SUM(pd.buyPrice*od.quantityordered)*100 AS 'profit%'

FROM orderdetails AS od INNER JOIN orders AS ord ON od.ordernumber=ord.ordernumber

INNER JOIN products as pd ON pd.productcode=od.productcode WHERE ord.status='shipped';
```

Output:

Total_buy_price	Total_MSRP	total_Selling_price	profit	profit%
5327566.77	9855391.99	8865094.64	3537527.87	66.400442

8. Display total profit and profit percentage according to year

Query:

```
SELECT YEAR(ord.shippeddate) AS "Year",COUNT(DISTINCT od.ordernumber) AS "total Orders",

SUM(pd.buyPrice*od.quantityordered) AS 'Total_buy_price',

SUM(priceeach*quantityordered) AS 'total_Selling_price',

(SUM(priceeach*quantityordered)-SUM(pd.buyPrice*od.quantityordered)) AS 'profit',

(SUM(priceeach*quantityordered)-
SUM(pd.buyPrice*od.quantityordered))/SUM(pd.buyPrice*od.quantityordered)*100 AS'profit%'

FROM orderdetails AS od INNER JOIN orders AS ord ON od.ordernumber=ord.ordernumber

INNER JOIN products AS pd ON pd.productcode=od.productcode GROUP BY YEAR(ord.shippeddate),
ord.status HAVING ord.status='shipped';
```

Output:

Year	Orders	Total_by_price	Total MSRP	Selling_price	profit	profit%
2003	108	1936919.03	3581586.76	3223095.80	1286176.77	66.403229
2004	145	2576920.58	4772465.23	4300602.99	1723682.41	66.889233
2005	50	813727.16	1501340.00	1341395.85	527668.69	64.845899

9. Display top 10 product according to quantity ordered and sales

Query:

```
SELECT productline,productname,SUM(quantityordered) AS 'total_Quantity_sold',

SUM(priceeach*quantityordered) AS 'total_sales'

FROM orderdetails AS od INNER JOIN products AS pd ON od.productCode=pd.productCode

INNER JOIN orders AS ord ON od.ordernumber=ord.ordernumber

GROUP BY productLine,productname,STATUS HAVING  ord.status='shipped' ORDER BY
SUM(priceeach*quantityordered) DESC LIMIT 10;
```

Output:

productline	productname	total_Quantity_sold	total_sales
Classic Cars	1992 Ferrari 360 Spider red	1720	264132.78
Classic Cars	2001 Ferrari Enzo	973	182439.52
Classic Cars	1952 Alpine Renault 1300	911	179945.96
Motorcycles	2003 Harley-Davidson Eagle Drag Bike	929	161576.48
Classic Cars	1968 Ford Mustang	909	157749.08
Classic Cars	1969 Ford Falcon	921	145082.38
Classic Cars	1998 Chrysler Plymouth Prowler	958	138404.55
Motorcycles	2002 Suzuki XREO	1007	132730.43
Classic Cars	1957 Corvette Convertible	1013	130749.31
Classic Cars	1956 Porsche 356A Coupe	1013	129749.86

10. Display Bottom 10 product according to quantity ordered and sales

Query:

```
SELECT productline,productname,SUM(quantityordered) AS 'total_Quantity_sold',

SUM(priceeach*quantityordered) AS'total_sales'

FROM orderdetails AS od INNER JOIN products AS pd ON od.productCode=pd.productCode

INNER JOIN orders AS ord ON od.ordernumber=ord.ordernumber

GROUP BY productLine,productname,STATUS HAVING  ord.status='shipped' ORDER BY
SUM(priceeach*quantityordered) ASC LIMIT 10;
```

Output:

productline	productname	total_Quantity_sold	total_sales
Vintage Cars	1939 Chevrolet Deluxe Coupe	869	25924.97
Classic Cars	1982 Lamborghini Diablo	836	28278.27
Vintage Cars	1936 Mercedes Benz 500k Roadster	809	29234.04
Classic Cars	1958 Chevy Corvette Limited Edition	952	30641.54
Motorcycles	1982 Ducati 996 R	906	33268.76
Classic Cars	1966 Shelby Cobra 427 S/C	776	35368.24
Vintage Cars	1938 Cadillac V-16 Presidential Limousine	887	35452.81
Planes	Boeing X-32A JSF	807	35910.32
Vintage Cars	1930 Buick Marquette Phaeton	935	36055.46
Ships	Pont Yacht	770	38061.72

11. Display Top 10 city according to quantity ordered and sales

Query:

```
SELECT cs.country,SUM(od.quantityordered) AS 'Total_quantity',

SUM(priceeach*quantityordered) AS'total_sales'

FROM orderdetails AS od INNER JOIN orders AS ord ON ord.ordernumber=od.ordernumber

INNER JOIN customers AS cs ON ord.customernumber=cs.customerNumber GROUP BY cs.country,

 Ord.status having ord.status='shipped' ORDER BY SUM(priceeach*quantityordered) DESC LIMIT 10;
```

Output:

country	Total_quantity	total_sales
USA	32993	3032204.26
France	10663	965750.58
Spain	10646	947470.01
Australia	5550	509385.82
New Zealand	4800	416114.03
UK	4584	391503.90
Italy	4045	360616.81
Finland	3192	295149.35
Norway	2842	270846.30
Singapore	2760	263997.78

12. Display Bottom 10 country according to quantity ordered and sales

Query:

```
SELECT cs.country,SUM(od.quantityordered) AS 'Total_quantity',

SUM(priceeach*quantityordered) AS'total_sales'

FROM orderdetails AS od INNER JOIN orders AS ord ON ord.ordernumber=od.ordernumber

INNER JOIN customers AS cs ON ord.customernumber=cs.customerNumber GROUP BY cs.country,

ord.status having ord,status='shipped' ORDER BY SUM(priceeach*quantityordered) ASC LIMIT 10;
```

Output:

country	Total_quantity	total_sales
Hong Kong	596	45480.79
Ireland	490	49898.27
Philippines	961	87468.30
Belgium	963	91471.03
Switzerland	1078	108777.92
Sweden	1239	120457.09
Austria	1686	161418.16
Japan	1842	167909.95
Denmark	1770	176791.44
Germany	2148	196470.99

13. Display Top 10 customer according to quantity ordered and sales

Query:

```
SELECT cs.customernumber,cs.customername,

SUM(od.quantityOrdered) AS 'total_quantity_order',

SUM((od.quantityOrdered)*(od.priceEach)) AS 'Amount_purchase'

from customers AS cs inner join orders AS ord ON cs.customerNumber=ord.customerNumber

INNER JOIN orderdetails AS od ON od.orderNumber=ord.orderNumber GROUP BY
cs.customernumber,ord.status HAVING ord.status='shipped'

 ORDER BY SUM(od.quantityOrdered) DESC, SUM((od.quantityOrdered)*(od.priceEach)) DESC LIMIT 10;
```

Output:

customernumber	customername	quantity	Amount_purchase
141	Euro+ Shopping Channel	7544	668770.46
124	Mini Gifts Distributors Ltd.	6291	584188.24
114	Australian Collectors, Co.	1926	180585.07
187	AV Stores, Co.	1778	148410.09
151	Muscle Machine Inc	1775	177913.95
323	Down Under Souveniers, Inc	1691	154622.08
278	Rovelli Gifts	1650	127529.69
148	Dragon Souveniers, Ltd.	1524	156251.03
276	Anna's Decorations, Ltd	1469	137034.22
321	Corporate Gift Ideas Co.	1447	132340.78

14. Display Bottom 10 customer according to quantity ordered and sales

Query:

```
SELECT cs.customernumber,cs.customername,

SUM(od.quantityOrdered) AS 'total_quantity_order',

SUM((od.quantityOrdered)*(od.priceEach)) AS 'Amount_purchase'

from customers AS cs inner join orders AS ord ON cs.customerNumber=ord.customerNumber

INNER JOIN orderdetails AS od ON od.orderNumber=ord.orderNumber GROUP BY
cs.customernumber,ord.status HAVING ord.status='shipped'

 ORDER BY SUM(od.quantityOrdered) ASC, SUM((od.quantityOrdered)*(od.priceEach)) ASC LIMIT 10;
```

Output:

customernumber	customername	total_quantity_order	Amount_purchase
219	Boards & Toys Co.	102	7918.60
452	Mini Auto Werke	244	23938.09
103	Atelier graphique	270	22314.36
473	Frau da Collezione	272	25358.32
381	Royale Belge	278	29217.18
198	Auto-Moto Classics Inc.	287	21554.26
489	Double Decker Gift Stores, Ltd	357	29586.15
173	Cambridge Collectables Co.	357	32198.69
456	Microscale Inc.	381	29230.43
328	Tekni Collectables Inc.	391	38281.51

15. Display Employee details along with his total sales

Query:

```
SELECT distinct e.employeeNumber,CONCAT(e.firstName,' ',e.lastname) AS 'Employee_name',

e.jobTitle,o.officeCode,o.country, COUNT( distinct c.customerNumber) AS'totalcustomer',

SUM(p.amount) AS 'Totalsales'

FROM employees e INNER JOIN offices o ON e.officeCode=o.officeCode

INNER JOIN customers c ON c.salesRepEmployeeNumber=e.employeeNumber

INNER JOIN payments p ON p.customerNumber=c.customerNumber

GROUP BY e.employeeNumber ORDER BY sum(p.amount) DESC;
```

Output:

employeeNumber	Employee_name	jobTitle	officeCode	country	totalcustomer	Totalsales
1370	Gerard Hernandez	Sales Rep	4	France	7	1112003.81
1165	Leslie Jennings	Sales Rep	1	USA	6	989906.55
1401	Pamela Castillo	Sales Rep	4	France	10	750201.87
1501	Larry Bott	Sales Rep	7	UK	8	686653.25
1504	Barry Jones	Sales Rep	7	UK	9	637672.65
1323	George Vanauf	Sales Rep	3	USA	8	584406.80
1337	Loui Bondur	Sales Rep	4	France	6	569485.75
1611	Andy Fixter	Sales Rep	6	Australia	5	509385.82
1612	Peter Marsh	Sales Rep	6	Australia	5	497907.16
1286	Foon Yue Tseng	Sales Rep	3	USA	6	488212.67
1621	Mami Nishi	Sales Rep	5	Japan	5	457110.07
1216	Steve Patterson	Sales Rep	2	USA	6	449219.13
1702	Martin Gerard	Sales Rep	4	France	5	387477.47
1188	Julie Firrelli	Sales Rep	2	USA	6	386663.20
1166	Leslie Thompson	Sales Rep	1	USA	6	347533.03

16. Display Stock Availability

Query:

```
SELECT DISTINCT pd.productName, w.warehouseName, pd.quantityInStock,

(SELECT SUM(od.quantityOrdered) FROM orderdetails AS od WHERE od.productCode = pd.productCode)
AS 'totalOrderedQuantity',

(pd.quantityInStock-(SELECT SUM(od.quantityOrdered) FROM orderdetails AS od WHERE
od.productCode = pd.productCode)) AS'remaining'

FROM warehouses AS w INNER JOIN products AS pd ON w.warehouseCode = pd.warehouseCode

INNER JOIN orderdetails AS od ON pd.productCode = od.productCode;
```

Output:

productName	warehouseName	quantityInStock	totalOrderedQuantity	remaining
2002 Suzuki XREO	North	9997	1028	8969
1995 Honda Civic	East	9772	917	8855
America West Airlines B757-200	North	9653	984	8669
2002 Chevy Corvette	East	9446	894	8552
1932 Model A Ford J-Coupe	West	9354	957	8397
1982 Ducati 996 R	North	9241	906	8335
1976 Ford Gran Torino	East	9127	915	8212
1968 Dodge Charger	East	9123	925	8198
1912 Ford Model T Delivery Wagon	West	9173	991	8182
1965 Aston Martin DB5	East	9042	914	8128
1948 Porsche Type 356 Roadster	East	8990	948	8042
1948 Porsche 356-A Roadster	East	8826	972	7854
American Airlines: MD-11S	North	8820	1085	7735
1936 Mercedes-Benz 500K Special Roa...	West	8635	960	7675
1950's Chicago Surface Lines Streetcar	South	8601	934	7667

17. Display Stock which is not available

Query:

```
SELECT DISTINCT pd.productName, w.warehouseName, pd.quantityInStock,

(SELECT SUM(od.quantityOrdered) FROM orderdetails AS od WHERE od.productCode = pd.productCode)
AS 'totalOrderedQuantity',

(pd.quantityInStock-(SELECT SUM(od.quantityOrdered) FROM orderdetails AS od WHERE
od.productCode = pd.productCode)) AS'remaining'

FROM warehouses AS w INNER JOIN products AS pd ON w.warehouseCode = pd.warehouseCode

INNER JOIN orderdetails AS od ON pd.productCode = od.productCode

WHERE (pd.quantityInStock-(SELECT SUM(od.quantityOrdered) FROM orderdetails AS od WHERE
od.productCode = pd.productCode))<0;
```

Output:

productName	warehouseName	quantityInStock	totalOrderedQuantity	remaining
1960 BSA Gold Star DBD34	North	15	1015	-1000
1997 BMW F650 ST	North	178	1014	-836
2002 Yamaha YZR M1	North	600	992	-392
F/A 18 Hornet 1/72	North	551	1047	-496
1968 Ford Mustang	East	68	933	-865
1911 Ford Town Car	West	540	832	-292
1928 Mercedes-Benz SSK	West	548	880	-332
1928 Ford Phaeton Deluxe	West	136	972	-836
1996 Peterbilt 379 Stake Bed with Outrigger	South	814	988	-174
The Mayflower	South	737	898	-161
Pont Yacht	South	414	958	-544

18. Display Average Delivery date difference according to country

Query:

```
SELECT city,AVG(DATEDIFF(requireddate,orderdate)) AS 'Expected_del',

AVG(DATEDIFF(shippeddate,orderdate)) AS 'shiffed_after_placed' ,

AVG(DATEDIFF(requireddate,orderdate))-AVG(DATEDIFF(shippeddate,orderdate)) AS
'aftershipped_delTI_TIME'

FROM orders AS ord INNER JOIN customers AS cs ON ord.customernumber=cs.customerNumber

GROUP BY cs.city;
```

Output:

city	Expected_del	shiffed_after_placed	aftershipped_del_time
Nashua	8.0000	5.0000	3.0000
Frankfurt	8.5000	4.0000	4.5000
NYC	7.7500	3.1333	4.6167
Stavern	8.5000	3.7500	4.7500
Madrid	8.0323	3.7931	4.2392
Kobenhavn	8.8000	2.6000	6.2000
Bergamo	8.6667	4.6667	4.0000
Makati City	9.0000	5.3333	3.6667
Philadelphia	8.2000	4.0000	4.2000
Manchester	7.0000	2.0000	5.0000
San Franci...	8.7143	2.8571	5.8571
Luleå	9.0000	3.3333	5.6667
San Rafael	8.1765	2.8750	5.3015
Paris	7.6667	3.6667	4.0000
Charleroi	8.7500	2.5000	6.2500

19. Display product wise discount

Query:

```
SELECT pd.productname AS 'Product_name',pd.buyprice AS 'buy_price',pd.msrp AS
'MSRP_price',od.priceeach AS 'Selling_price',

(pd.msrp-od.priceeach) AS 'Discount',ROUND(((pd.msrp-od.priceeach)/pd.msrp)*100) AS 'Discount%'

FROM products AS pd INNER JOIN orderdetails AS od ON pd.productcode=od.productcode

ORDER BY ROUND(((pd.msrp-od.priceeach)/pd.msrp)*100) DESC ;
```

Output:

Product_name	buy_price	MSRP_price	Selling_price	Discount	Discount%
1969 Harley Davidson Ultimate Chopper	48.81	95.70	76.56	19.14	20
1932 Alfa Romeo 8C2300 Spider Sport	43.26	92.03	73.62	18.41	20
1969 Harley Davidson Ultimate Chopper	48.81	95.70	76.56	19.14	20
1954 Greyhound Scenicruiser	25.98	54.11	43.29	10.82	20
1969 Harley Davidson Ultimate Chopper	48.81	95.70	76.56	19.14	20
The Mayflower	43.30	86.61	69.29	17.32	20
1952 Alpine Renault 1300	98.58	214.30	171.44	42.86	20
1936 Mercedes Benz 500k Roadster	21.75	41.03	32.82	8.21	20
ATA: B757-300	59.33	118.65	94.92	23.73	20
1936 Mercedes Benz 500k Roadster	21.75	41.03	32.82	8.21	20
1940s Ford truck	84.76	121.08	96.86	24.22	20
1957 Ford Thunderbird	34.21	71.27	57.02	14.25	20
Pont Yacht	33.30	54.60	43.68	10.92	20
1937 Horch 930V Limousine	26.30	65.75	52.60	13.15	20
The Mayflower	43.30	86.61	69.29	17.32	20

20. Display warehouse wise sales

Query:

```
SELECT DISTINCT w.warehousename, p.warehousecode,

sum(DISTINCT od.quantityordered) AS 'total_quantity' ,

SUM((od.quantityOrdered)*(od.priceEach)) AS 'total_sales'

FROM products p INNER JOIN warehouses w ON p.warehouseCode=w.warehouseCode

INNER JOIN orderdetails od on p.productCode=od.productCode

INNER JOIN orders ord ON ord.ordernumber=od.ordernumber GROUP BY w.warehouseCode ,ord.status
HAVING ord.status='shipped' ;
```

Output:

warehousename	warehousecode	total_quantity	total_sales
East	b	1788	3623600.63
North	a	1516	1917657.40
South	d	1390	1680664.12
West	c	1454	1643172.49

21. Display South (d) warehouse products

Query:

```
SELECT DISTINCT p.productcode,w.warehousename, p.warehousecode,p.productname,p.quantityInStock,

SUM(od.quantityordered) AS 'total_quantity',

SUM((od.quantityOrdered)*(od.priceEach)) AS'total_sale'

FROM products p INNER JOIN warehouses w ON p.warehouseCode=w.warehouseCode

INNER JOIN orderdetails od ON p.productCode=od.productCode

INNER JOIN orders ord ON ord.ordernumber=od.ordernumber

group by productCode ,status having status='shipped' and p.warehousecode='d'

order by SUM((od.quantityOrdered)*(od.priceEach)) asc;
```

Output:

productcode	warehousename	warehousecode	productname	quantityInStock	total_quantity	total_sale
S32_3207	South	d	1950's Chicago Surface Lines Streetcar	8601	907	52298.89
S18_2319	South	d	1964 Mercedes Tour Bus	8258	969	108224.50
S700_2610	South	d	The USS Constitution Ship	7083	853	55859.46
S18_3259	South	d	Collectable Wooden Train	6450	799	71789.11
S12_4473	South	d	1957 Chevy Pickup	6125	1023	106778.54
S32_1268	South	d	1980's GM Manhattan Express	5099	846	72436.67
S700_3962	South	d	The Queen Mary	5088	726	64474.56
S18_3029	South	d	1999 Yamaha Speed Boat	4259	761	58625.94
S700_2047	South	d	HMS Bounty	3501	763	62274.05
S18_4600	South	d	1940s Ford truck	3128	929	100304.92
S32_2509	South	d	1954 Greyhound Scenicruiser	2874	944	45965.53
S18_1097	South	d	1940 Ford Pickup Truck	2613	909	95965.88
S24_2300	South	d	1962 Volkswagen Microbus	2327	938	107574.68
S18_2432	South	d	1926 Ford Fire Engine	2018	945	53113.94
S700_3505	South	d	The Titanic	1956	782	70666.45

Conclusion:

According to Warehouse:

In the operations of classic company, which operates four warehouses strategically located at the cardinal points of east, west, south and north, it's noteworthy that the East warehouse stand out worth the highest sales performance compare to the other warehouses. This trend underscore the significance of the East warehouse as a key contributor to the company's revenue.

Additionally, while the East warehouse leads in sales, it's important to highlight that the south warehouse has experienced a notable quantity of orders. This indicated a substantial demand for products stored in the south warehouse.

According to countries:

The sales operations of classic company encompass a wide global reach, spanning across 27 countries. A notable observation emerges from this landscape, with a substantial 80% of these countries falling within the realms of the Europe and Asia continents. This underscores the significance of these two continents as primary contributors to the company's market presence and sales.

Taking a continent focused perspective, classic company sales there products in Europe, Asia, North America, Oceania and Africa. Among the continents where classic company operates, there's a noticeable absence of countries belonging to the southern region, with the exception of Australia.

Suggestion:

As a data analyst, I would like to recommend a strategic decision to classic company. After thorough analysis of the company's sales data and distribution trends, I propose considering the closure of the south(d) warehouse. Because we observe that customer base in southern region is limited. The demand for the products in the south warehouse shows a significant preference in the eastern and northern region.

So, the closure of the south warehouse is expected to have minimal impact on the overall business operations and customer service.