UNTERNEH

Supply Chain Management Analytics

This report analyzes important metrics in the supply chain management of **Unterneh** – a semiconductor company

Presented by **Ngoc Anh Nguyen**

Contents

Presented by ***Ngoc Anh Nguyen***

Project Overview

This project focuses on analyzing and extracting valuable insights from the provided database, which consists of two fact tables and six dimension tables.

The analysis centers on three key areas of **Unterneh**'s supply chain operations: sales, purchasing orders, and shipments.

The primary objectives of these analyses are to enhance the flow of goods, improve the efficiency of inventory and goods management, and optimize profitability.

*Presented by **Ngoc Anh Nguyen***

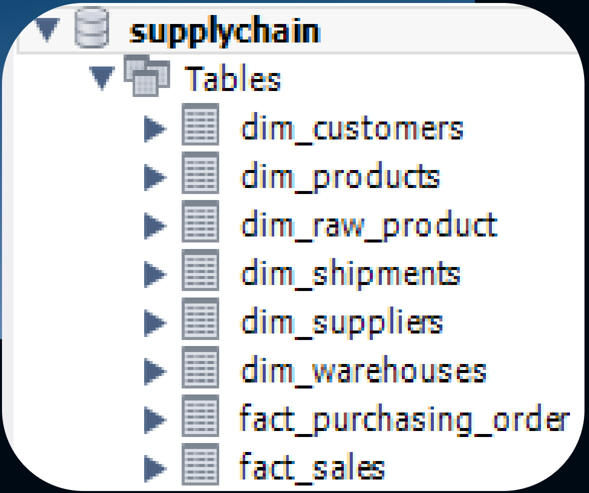
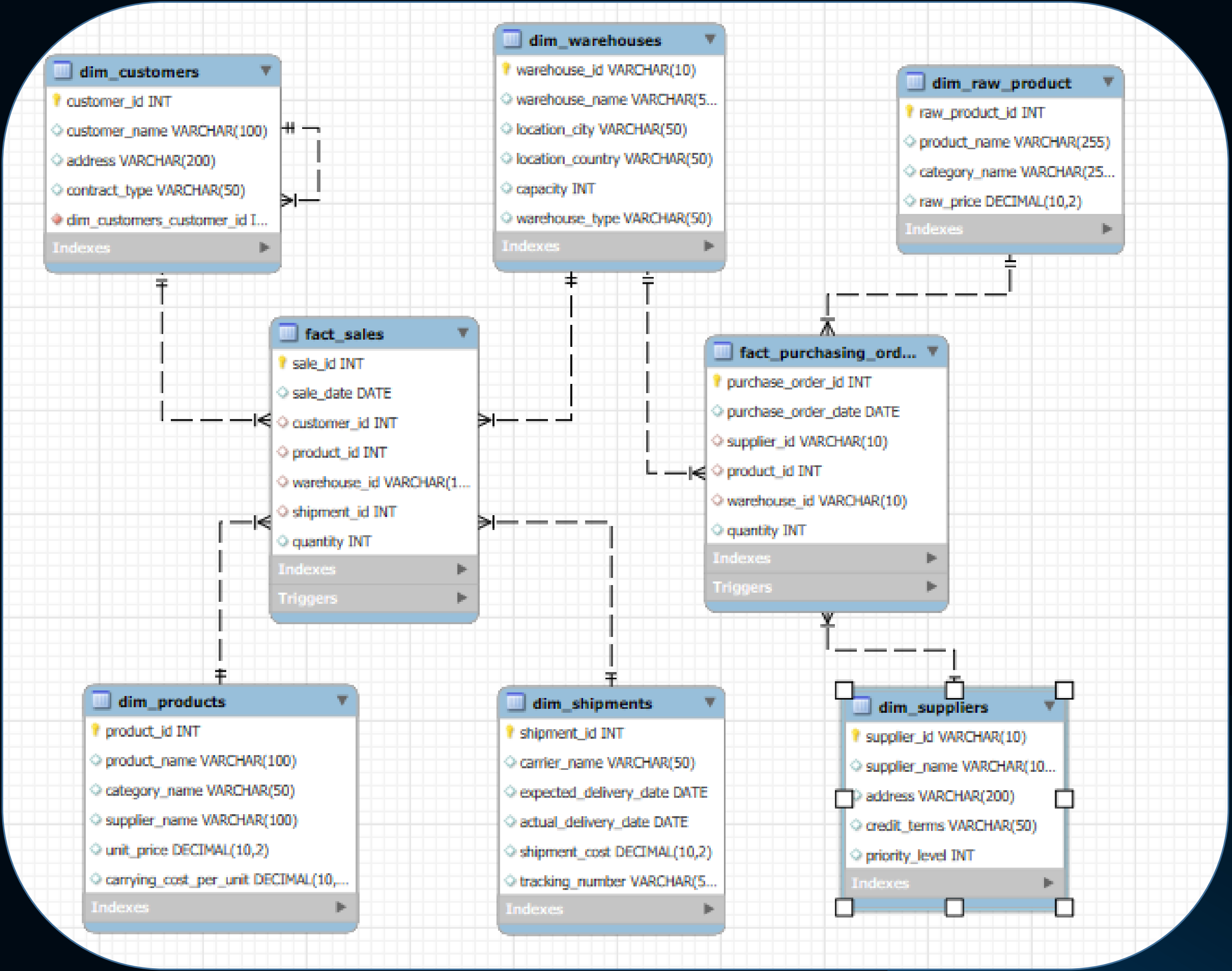

supplychain
Tables
- dim_customers
- dim_products
- dim_raw_product
- dim_shipments
- dim_suppliers
- dim_warehouses
- fact_purchasing_order
- fact_sales

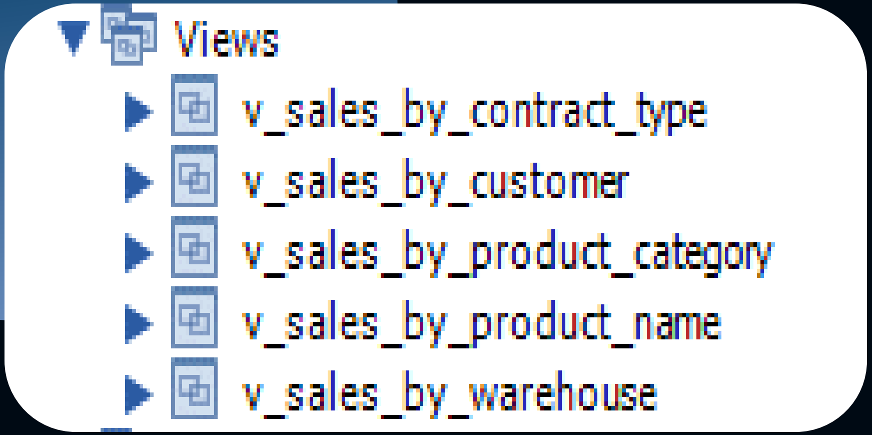
Table Structure

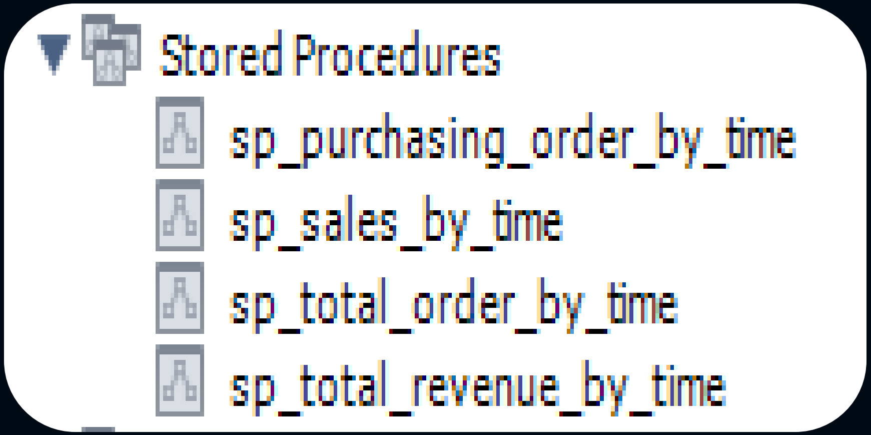
Data schema Overview

*Presented by **Ngoc Anh Nguyen***



Entity Relationship Diagram


Views
v_sales_by_contract_type
v_sales_by_customer
v_sales_by_product_category
v_sales_by_product_name
v_sales_by_warehouse

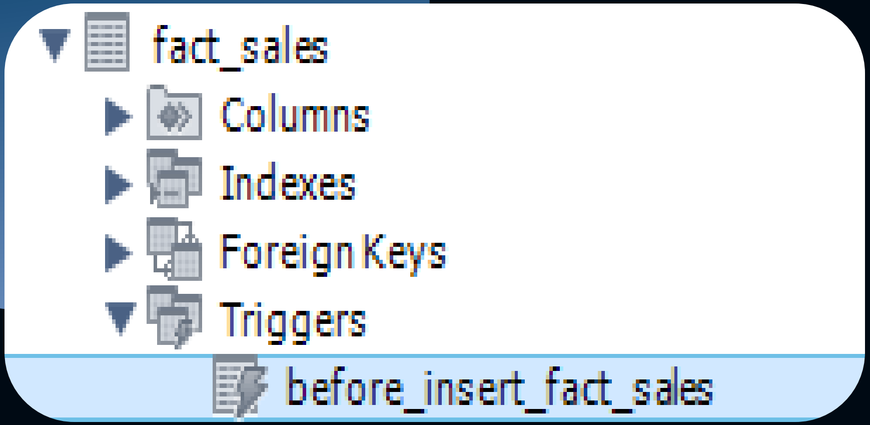

Stored Procedures
sp_purchasing_order_by_time
sp_sales_by_time
sp_total_order_by_time
sp_total_revenue_by_time

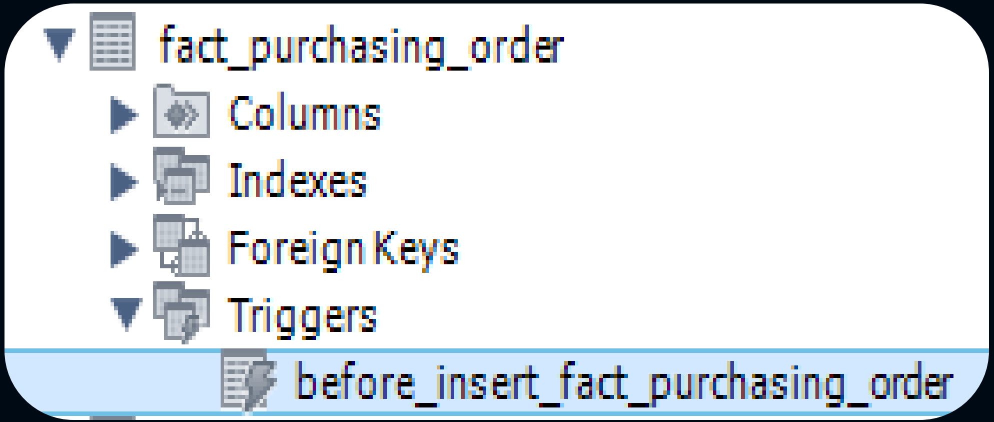
VIEWS

Views are virtual tables based on the result of a SELECT query. They provide a way to represent specific data subsets or join operations without altering the underlying database structure.

STORED PROCEDURES

Stored Procedures are precompiled sets of one or more SQL statements that are stored in the database. They are useful for encapsulating complex logic or performing repetitive tasks.





TRIGGER FOR SALES

- Before inserting
- Conditions
  - Quantity > 0
  - Sale_date is not in the future

TRIGGER FOR PURCHASING ORDER

- Before inserting
- Conditions
  - Quantity > 0
  - Sale_date is not in the future
  - Warehouse's type is "raw_material"

Analysis Structure

Revenue and profit analysis

- By customer
- By product
- By time
- By warehouse

Shipment

- Shipping cost
- Delay time (shipment, carrier, warehouse)

Purchasing order

- By supplier
- By product
- By time

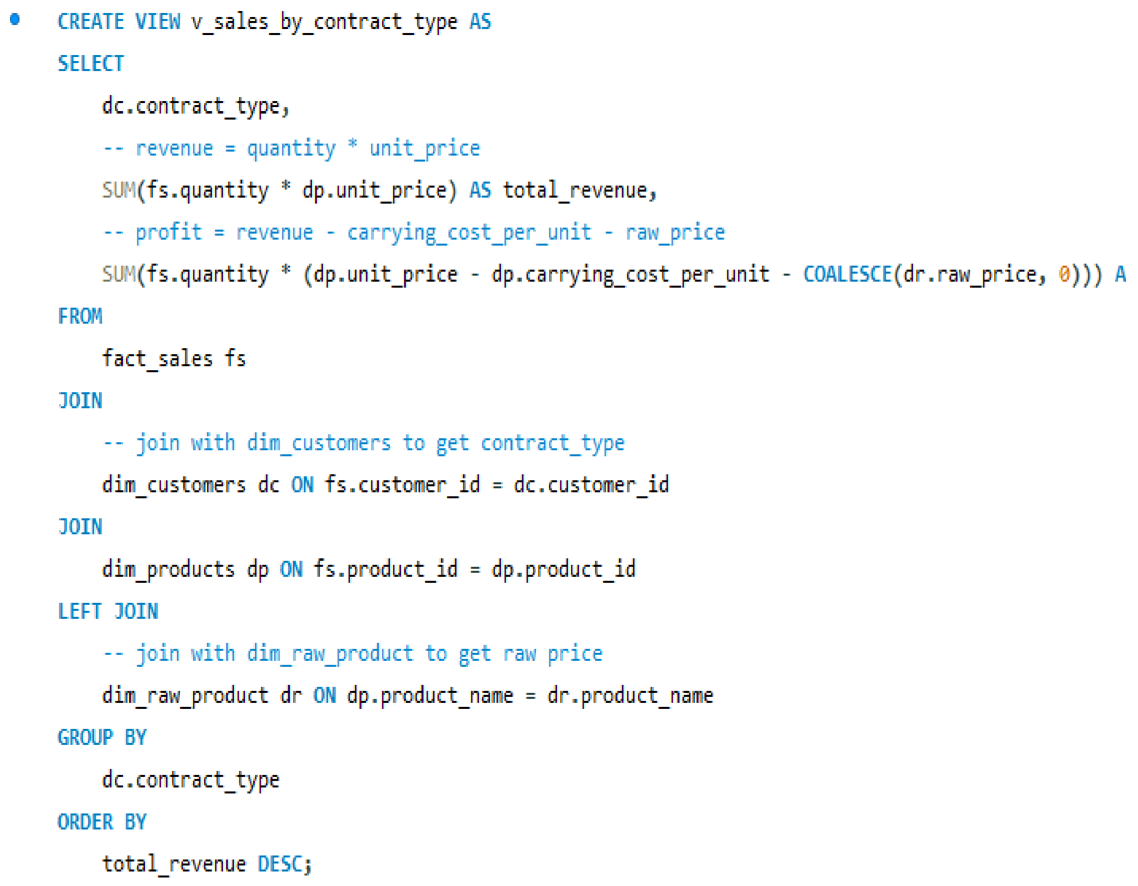
Others

- Stock-out and Overstock
- Predict product demand

Presented by ***Ngoc Anh Nguyen***

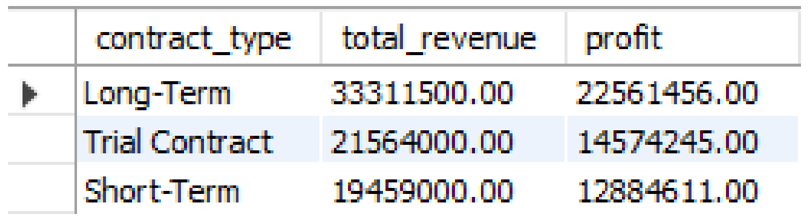
Total Revenue and Profit by contract type

MySQL QUERY



```
CREATE VIEW v_sales_by_contract_type AS
SELECT
    dc.contract_type,
    -- revenue = quantity * unit_price
    SUM(fs.quantity * dp.unit_price) AS total_revenue,
    -- profit = revenue - carrying_cost_per_unit - raw_price
    SUM(fs.quantity * (dp.unit_price - dp.carrying_cost_per_unit - COALESCE(dr.raw_price, 0))) A
FROM
    fact_sales fs
JOIN
    -- join with dim_customers to get contract_type
    dim_customers dc ON fs.customer_id = dc.customer_id
JOIN
    dim_products dp ON fs.product_id = dp.product_id
LEFT JOIN
    -- join with dim_raw_product to get raw price
    dim_raw_product dr ON dp.product_name = dr.product_name
GROUP BY
    dc.contract_type
ORDER BY
    total_revenue DESC;
```

OUTPUT

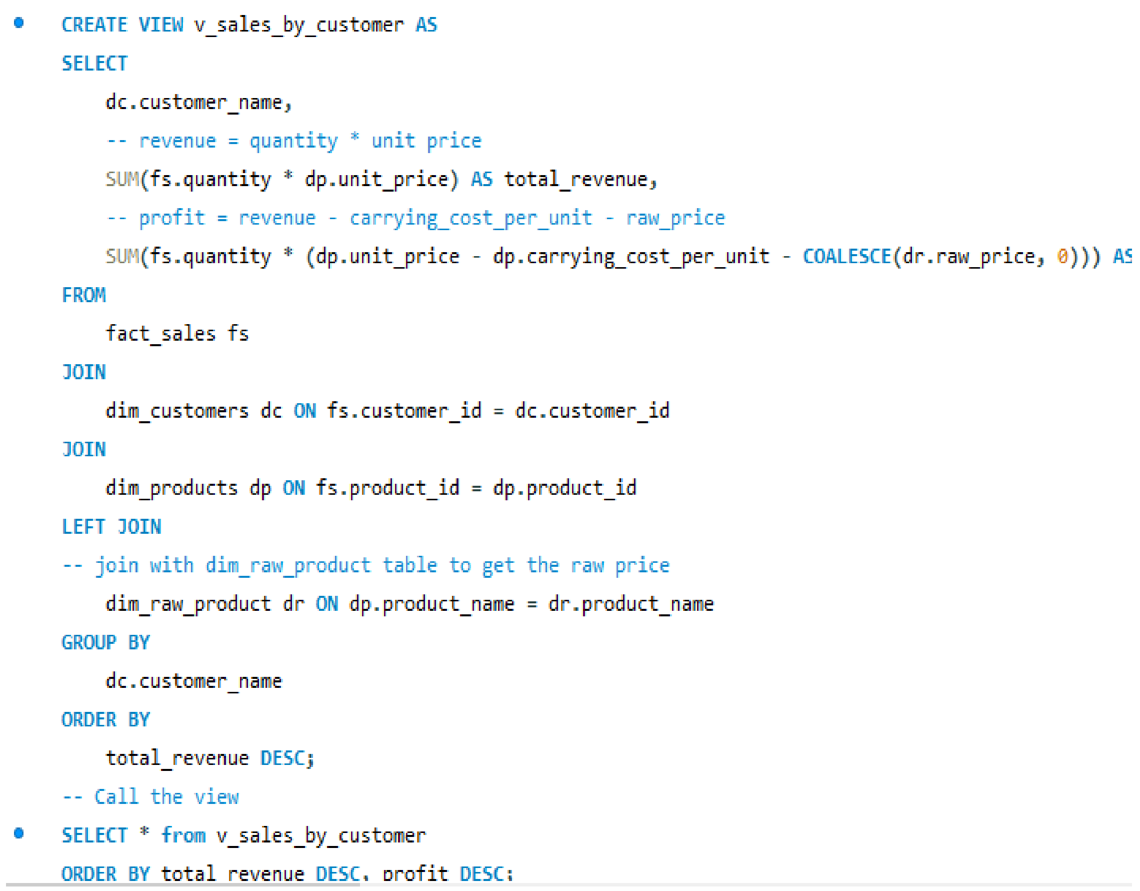


contract_type	total_revenue	profit
Long-Term	33311500.00	22561456.00
Trial Contract	21564000.00	14574245.00
Short-Term	19459000.00	12884611.00

Presented by ***Ngoc Anh Nguyen***

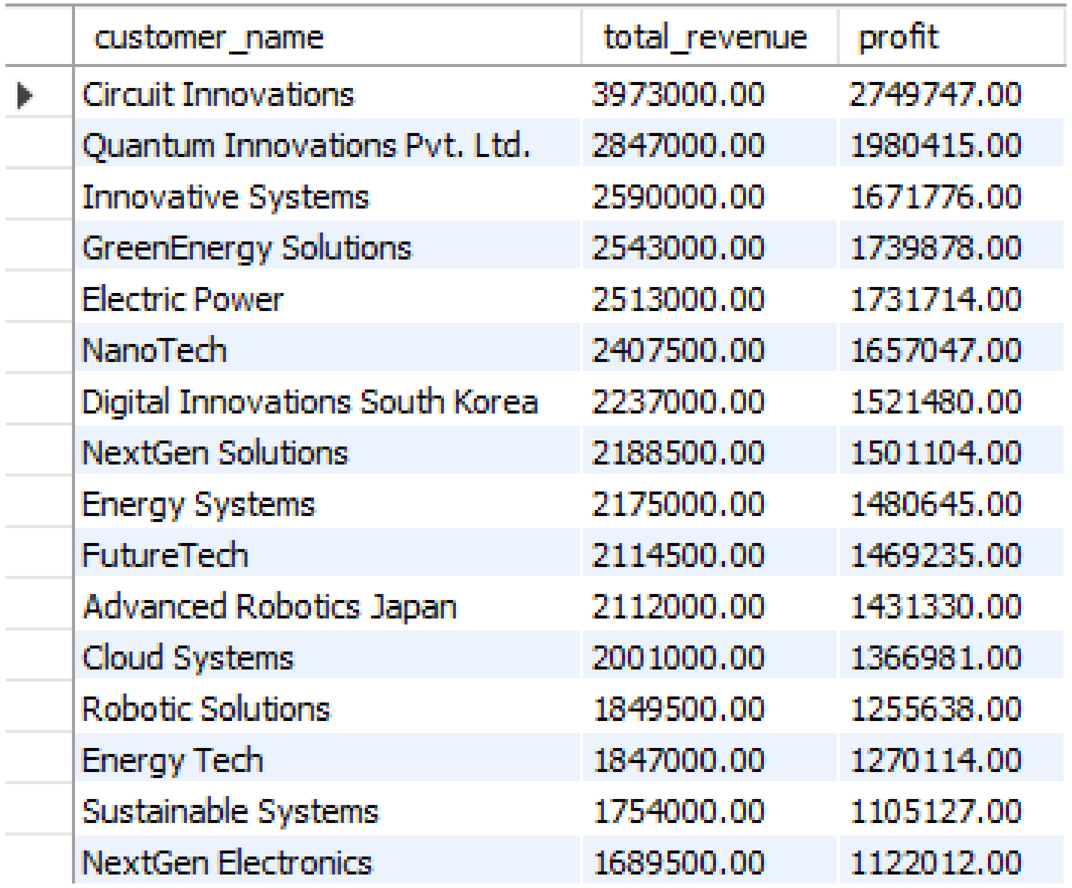
Total Revenue and Profit by each customer

MySQL QUERY



```
CREATE VIEW v_sales_by_customer AS
SELECT
    dc.customer_name,
    -- revenue = quantity * unit price
    SUM(fs.quantity * dp.unit_price) AS total_revenue,
    -- profit = revenue - carrying_cost_per_unit - raw_price
    SUM(fs.quantity * (dp.unit_price - dp.carrying_cost_per_unit - COALESCE(dr.raw_price, 0))) AS
FROM
    fact_sales fs
JOIN
    dim_customers dc ON fs.customer_id = dc.customer_id
JOIN
    dim_products dp ON fs.product_id = dp.product_id
LEFT JOIN
-- join with dim_raw_product table to get the raw price
    dim_raw_product dr ON dp.product_name = dr.product_name
GROUP BY
    dc.customer_name
ORDER BY
    total_revenue DESC;
-- Call the view
SELECT * from v_sales_by_customer
ORDER BY total_revenue DESC, profit DESC;
```

OUTPUT

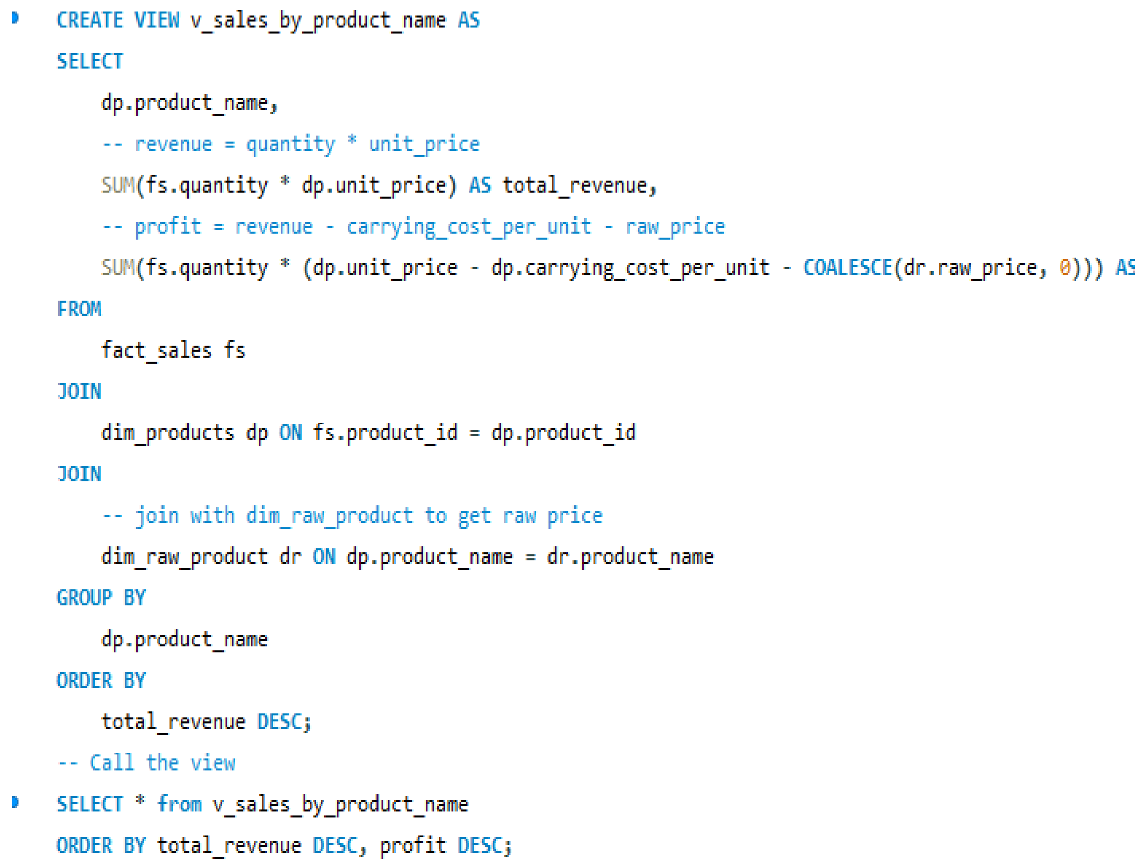


customer_name	total_revenue	profit
Circuit Innovations	3973000.00	2749747.00
Quantum Innovations Pvt. Ltd.	2847000.00	1980415.00
Innovative Systems	2590000.00	1671776.00
GreenEnergy Solutions	2543000.00	1739878.00
Electric Power	2513000.00	1731714.00
NanoTech	2407500.00	1657047.00
Digital Innovations South Korea	2237000.00	1521480.00
NextGen Solutions	2188500.00	1501104.00
Energy Systems	2175000.00	1480645.00
FutureTech	2114500.00	1469235.00
Advanced Robotics Japan	2112000.00	1431330.00
Cloud Systems	2001000.00	1366981.00
Robotic Solutions	1849500.00	1255638.00
Energy Tech	1847000.00	1270114.00
Sustainable Systems	1754000.00	1105127.00
NextGen Electronics	1689500.00	1122012.00

Presented by ***Ngoc Anh Nguyen***

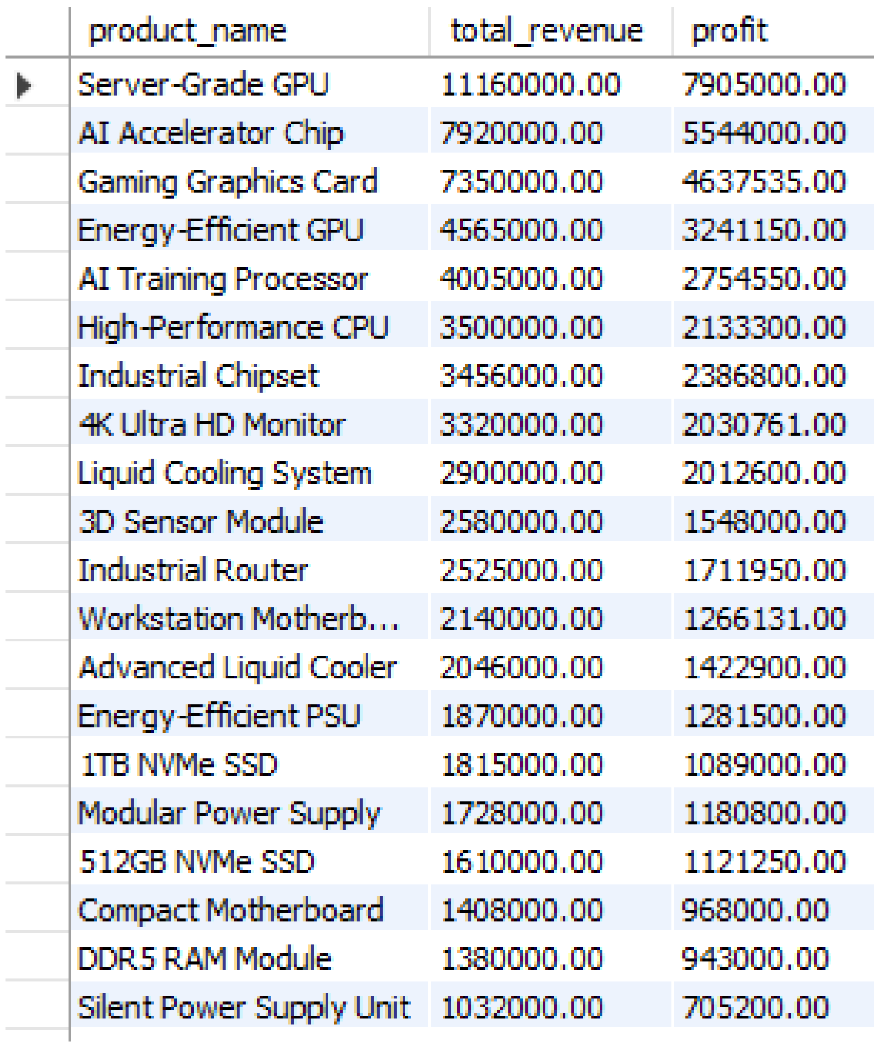
Total Revenue and Profit by product name

MySQL QUERY



```
CREATE VIEW v_sales_by_product_name AS
SELECT
    dp.product_name,
    -- revenue = quantity * unit_price
    SUM(fs.quantity * dp.unit_price) AS total_revenue,
    -- profit = revenue - carrying_cost_per_unit - raw_price
    SUM(fs.quantity * (dp.unit_price - dp.carrying_cost_per_unit - COALESCE(dr.raw_price, 0))) AS
FROM
    fact_sales fs
JOIN
    dim_products dp ON fs.product_id = dp.product_id
JOIN
    -- join with dim_raw_product to get raw price
    dim_raw_product dr ON dp.product_name = dr.product_name
GROUP BY
    dp.product_name
ORDER BY
    total_revenue DESC;
-- Call the view
SELECT * from v_sales_by_product_name
ORDER BY total_revenue DESC, profit DESC;
```

OUTPUT



product_name	total_revenue	profit
Server-Grade GPU	11160000.00	7905000.00
AI Accelerator Chip	7920000.00	5544000.00
Gaming Graphics Card	7350000.00	4637535.00
Energy-Efficient GPU	4565000.00	3241150.00
AI Training Processor	4005000.00	2754550.00
High-Performance CPU	3500000.00	2133300.00
Industrial Chipset	3456000.00	2386800.00
4K Ultra HD Monitor	3320000.00	2030761.00
Liquid Cooling System	2900000.00	2012600.00
3D Sensor Module	2580000.00	1548000.00
Industrial Router	2525000.00	1711950.00
Workstation Motherb...	2140000.00	1266131.00
Advanced Liquid Cooler	2046000.00	1422900.00
Energy-Efficient PSU	1870000.00	1281500.00
1TB NVMe SSD	1815000.00	1089000.00
Modular Power Supply	1728000.00	1180800.00
512GB NVMe SSD	1610000.00	1121250.00
Compact Motherboard	1408000.00	968000.00
DDR5 RAM Module	1380000.00	943000.00
Silent Power Supply Unit	1032000.00	705200.00

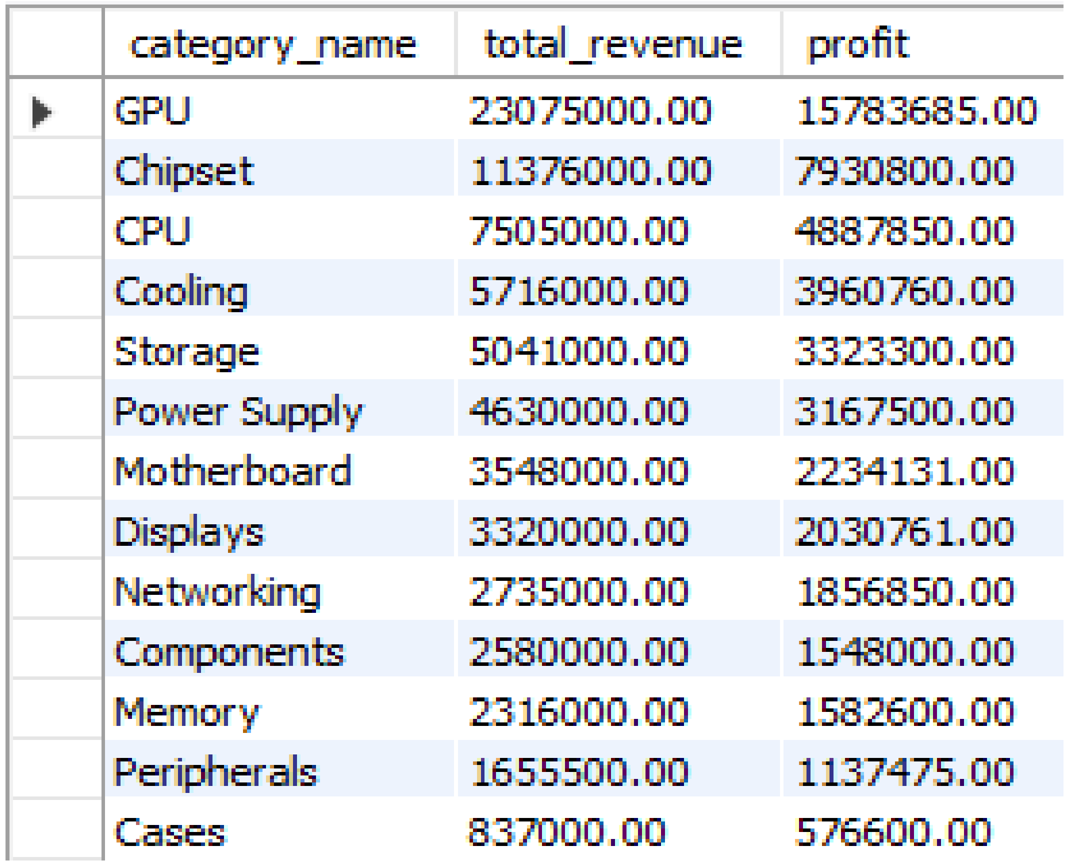
*Presented by **Ngoc Anh Nguyen***

Total Revenue and Profit by category

MySQL QUERY

```
CREATE VIEW v_sales_by_product_category AS
SELECT
    dp.category_name,
    -- revenue = quantity * unit_price
    SUM(fs.quantity * dp.unit_price) AS total_revenue,
    -- profit = revenue - carrying_cost_per_unit - raw_price
    SUM(fs.quantity * (dp.unit_price - dp.carrying_cost_per_unit - COALESCE(dr.raw_price, 0))) AS
FROM
    fact_sales fs
JOIN
    dim_products dp ON fs.product_id = dp.product_id
JOIN
    -- join with dim_raw_product to get raw price
    dim_raw_product dr ON dp.product_name = dr.product_name
GROUP BY
    dp.category_name
ORDER BY
    total_revenue DESC;
-- Call the view
SELECT * from v_sales_by_product_name
ORDER BY total_revenue DESC, profit DESC;
```

OUTPUT

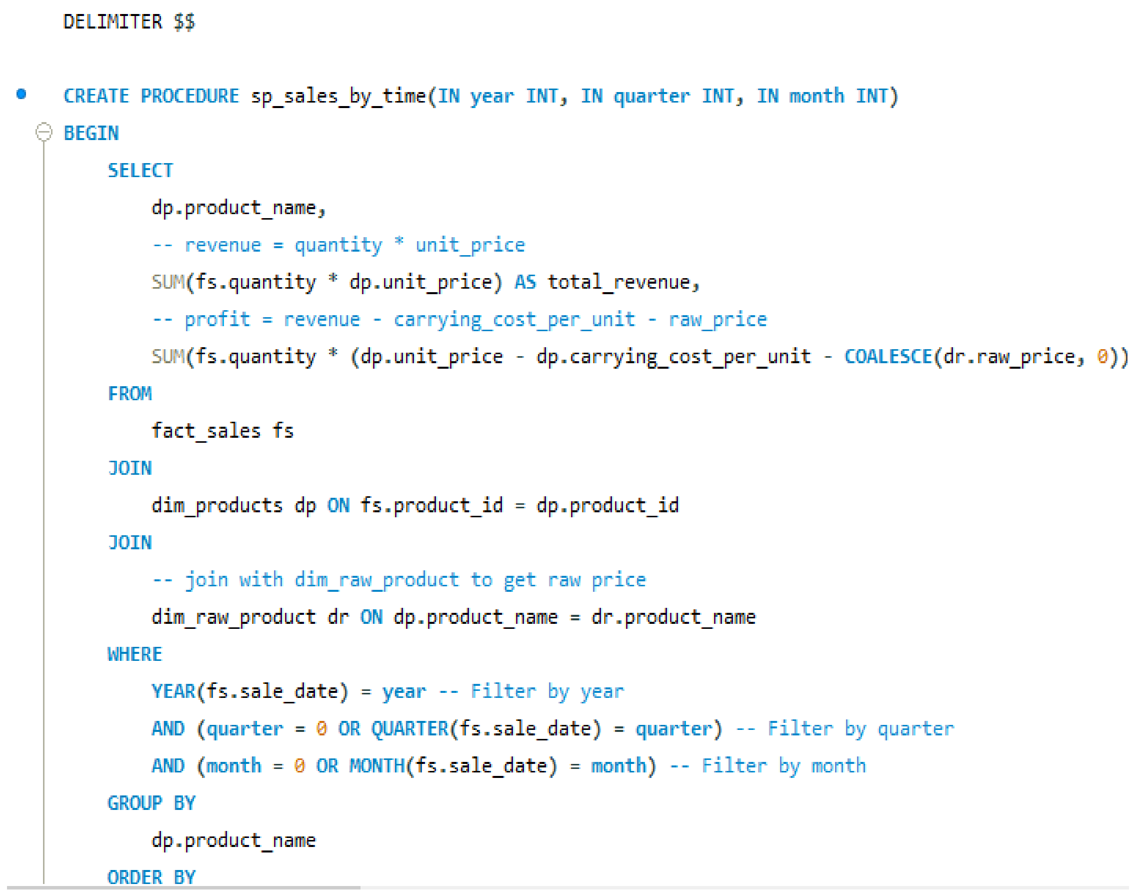


category_name	total_revenue	profit
GPU	23075000.00	15783685.00
Chipset	11376000.00	7930800.00
CPU	7505000.00	4887850.00
Cooling	5716000.00	3960760.00
Storage	5041000.00	3323300.00
Power Supply	4630000.00	3167500.00
Motherboard	3548000.00	2234131.00
Displays	3320000.00	2030761.00
Networking	2735000.00	1856850.00
Components	2580000.00	1548000.00
Memory	2316000.00	1582600.00
Peripherals	1655500.00	1137475.00
Cases	837000.00	576600.00

*Presented by **Ngoc Anh Nguyen***

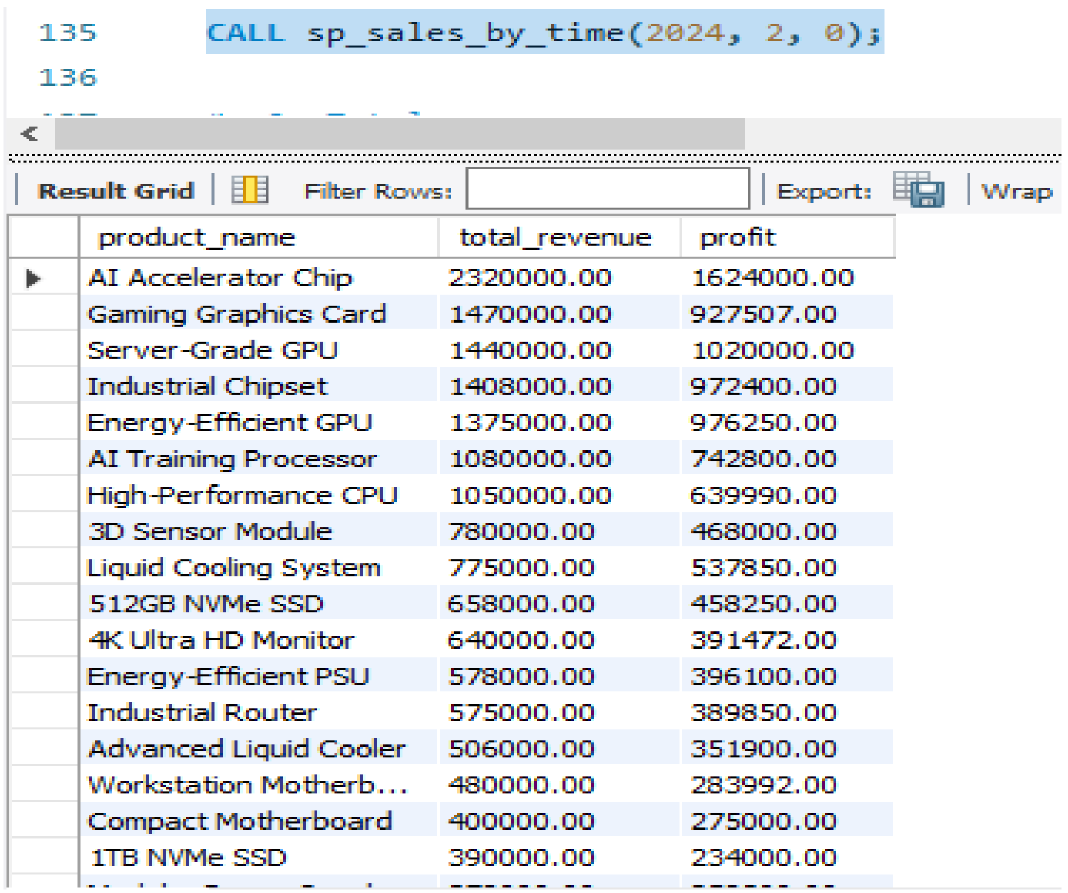
Total Revenue and Profit by time & product

MySQL QUERY



```
DELIMITER $$

CREATE PROCEDURE sp_sales_by_time(IN year INT, IN quarter INT, IN month INT)
BEGIN
    SELECT
        dp.product_name,
        -- revenue = quantity * unit_price
        SUM(fs.quantity * dp.unit_price) AS total_revenue,
        -- profit = revenue - carrying_cost_per_unit - raw_price
        SUM(fs.quantity * (dp.unit_price - dp.carrying_cost_per_unit - COALESCE(dr.raw_price, 0))
    FROM
        fact_sales fs
    JOIN
        dim_products dp ON fs.product_id = dp.product_id
    JOIN
        -- join with dim_raw_product to get raw price
        dim_raw_product dr ON dp.product_name = dr.product_name
    WHERE
        YEAR(fs.sale_date) = year -- Filter by year
        AND (quarter = 0 OR QUARTER(fs.sale_date) = quarter) -- Filter by quarter
        AND (month = 0 OR MONTH(fs.sale_date) = month) -- Filter by month
    GROUP BY
        dp.product_name
    ORDER BY
```

OUTPUT



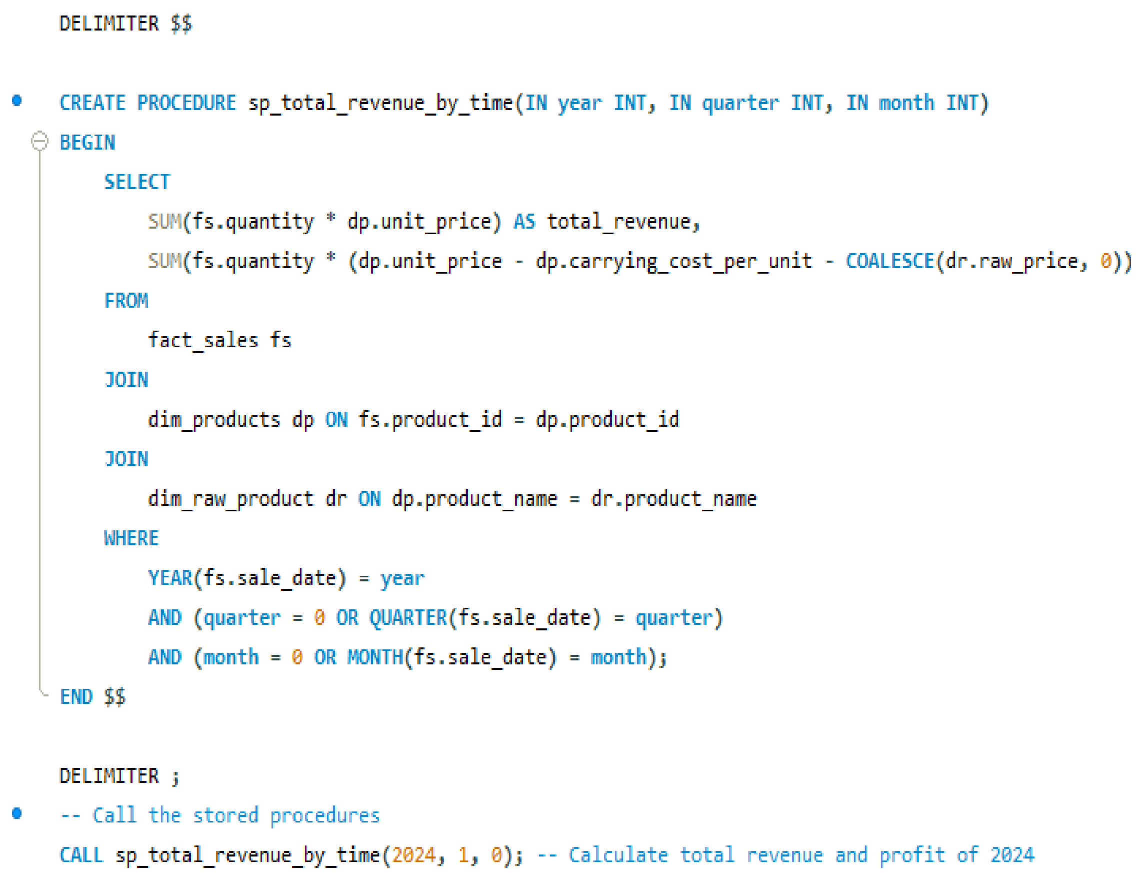
```
CALL sp_sales_by_time(2024, 2, 0);
```

product_name	total_revenue	profit
AI Accelerator Chip	2320000.00	1624000.00
Gaming Graphics Card	1470000.00	927507.00
Server-Grade GPU	1440000.00	1020000.00
Industrial Chipset	1408000.00	972400.00
Energy-Efficient GPU	1375000.00	976250.00
AI Training Processor	1080000.00	742800.00
High-Performance CPU	1050000.00	639990.00
3D Sensor Module	780000.00	468000.00
Liquid Cooling System	775000.00	537850.00
512GB NVMe SSD	658000.00	458250.00
4K Ultra HD Monitor	640000.00	391472.00
Energy-Efficient PSU	578000.00	396100.00
Industrial Router	575000.00	389850.00
Advanced Liquid Cooler	506000.00	351900.00
Workstation Motherb...	480000.00	283992.00
Compact Motherboard	400000.00	275000.00
1TB NVMe SSD	390000.00	234000.00

Presented by ***Ngoc Anh Nguyen***

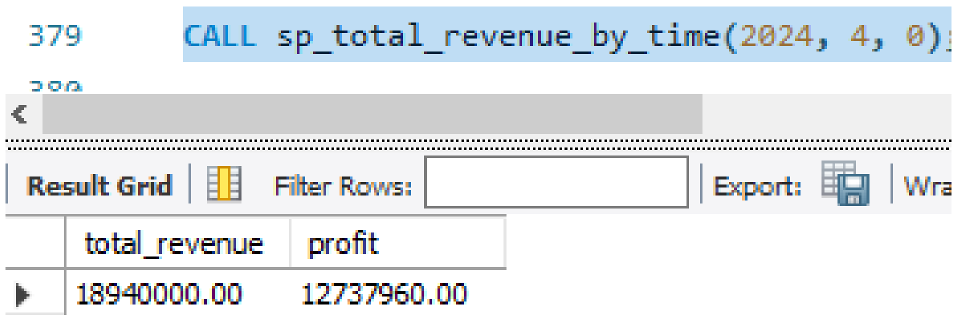
Total Revenue and Profit by time

MySQL QUERY



```sql
DELIMITER $$

CREATE PROCEDURE sp_total_revenue_by_time(IN year INT, IN quarter INT, IN month INT)
BEGIN
    SELECT
        SUM(fs.quantity * dp.unit_price) AS total_revenue,
        SUM(fs.quantity * (dp.unit_price - dp.carrying_cost_per_unit - COALESCE(dr.raw_price, 0))
    FROM
        fact_sales fs
    JOIN
        dim_products dp ON fs.product_id = dp.product_id
    JOIN
        dim_raw_product dr ON dp.product_name = dr.product_name
    WHERE
        YEAR(fs.sale_date) = year
        AND (quarter = 0 OR QUARTER(fs.sale_date) = quarter)
        AND (month = 0 OR MONTH(fs.sale_date) = month);
END $$

DELIMITER ;
-- Call the stored procedures
CALL sp_total_revenue_by_time(2024, 1, 0); -- Calculate total revenue and profit of 2024
```

OUTPUT



```
CALL sp_total_revenue_by_time(2024, 4, 0);
```

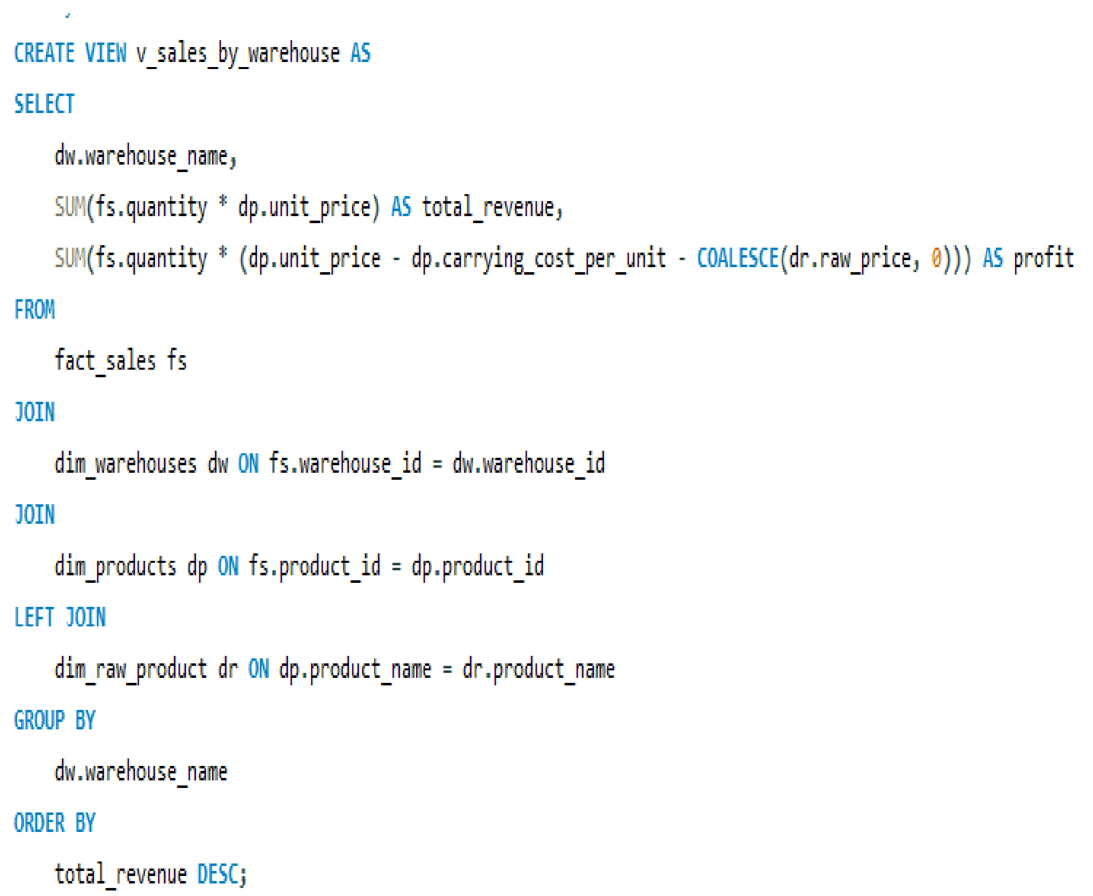
Result Grid | Filter Rows: | Export: | Wra

```
  total_revenue   profit
▶ 18940000.00     12737960.00
```

Presented by ***Ngoc Anh Nguyen***

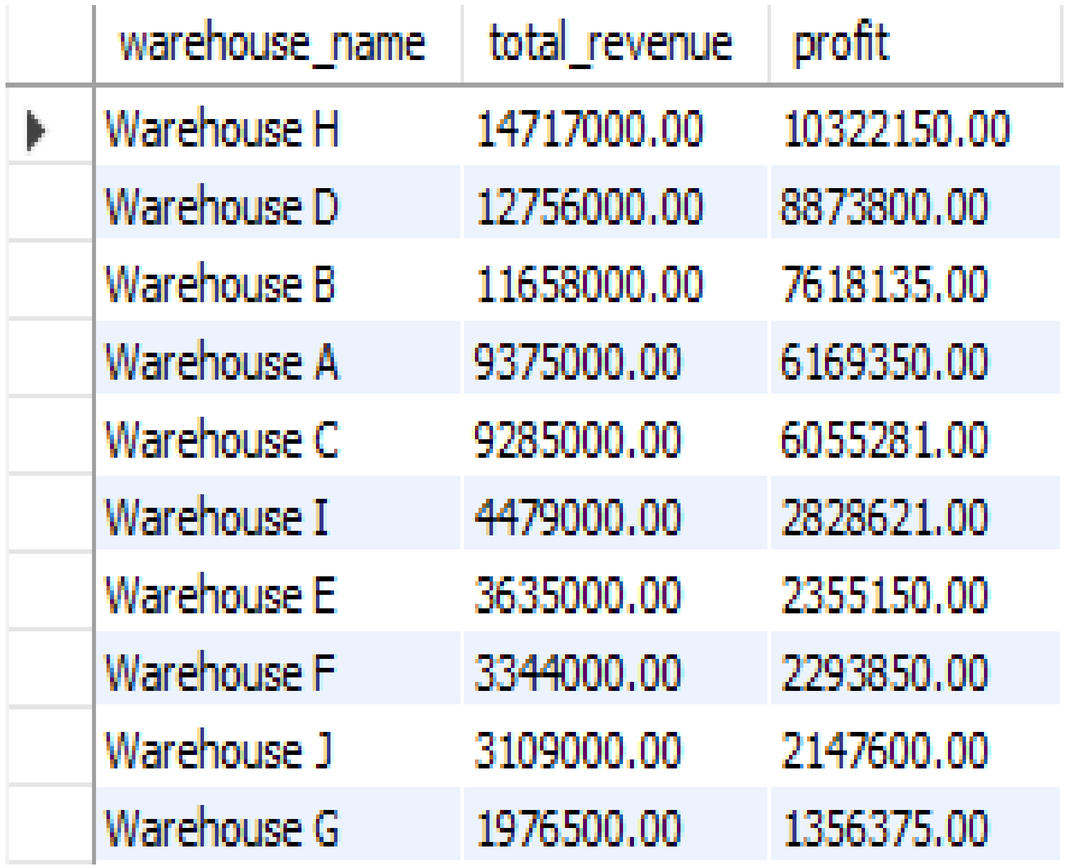
Total Revenue and Profit by warehouse

MySQL QUERY



```
CREATE VIEW v_sales_by_warehouse AS
SELECT
    dw.warehouse_name,
    SUM(fs.quantity * dp.unit_price) AS total_revenue,
    SUM(fs.quantity * (dp.unit_price - dp.carrying_cost_per_unit - COALESCE(dr.raw_price, 0))) AS profit
FROM
    fact_sales fs
JOIN
    dim_warehouses dw ON fs.warehouse_id = dw.warehouse_id
JOIN
    dim_products dp ON fs.product_id = dp.product_id
LEFT JOIN
    dim_raw_product dr ON dp.product_name = dr.product_name
GROUP BY
    dw.warehouse_name
ORDER BY
    total_revenue DESC;
```

OUTPUT

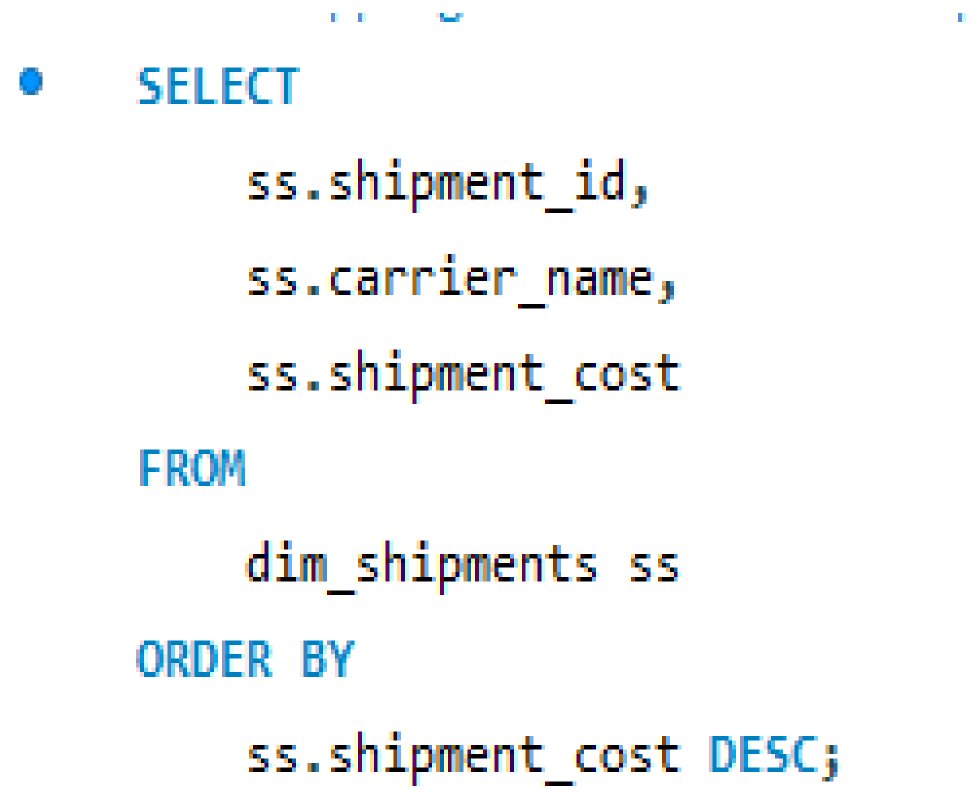


warehouse_name	total_revenue	profit
Warehouse H	14717000.00	10322150.00
Warehouse D	12756000.00	8873800.00
Warehouse B	11658000.00	7618135.00
Warehouse A	9375000.00	6169350.00
Warehouse C	9285000.00	6055281.00
Warehouse I	4479000.00	2828621.00
Warehouse E	3635000.00	2355150.00
Warehouse F	3344000.00	2293850.00
Warehouse J	3109000.00	2147600.00
Warehouse G	1976500.00	1356375.00

Presented by ***Ngoc Anh Nguyen***

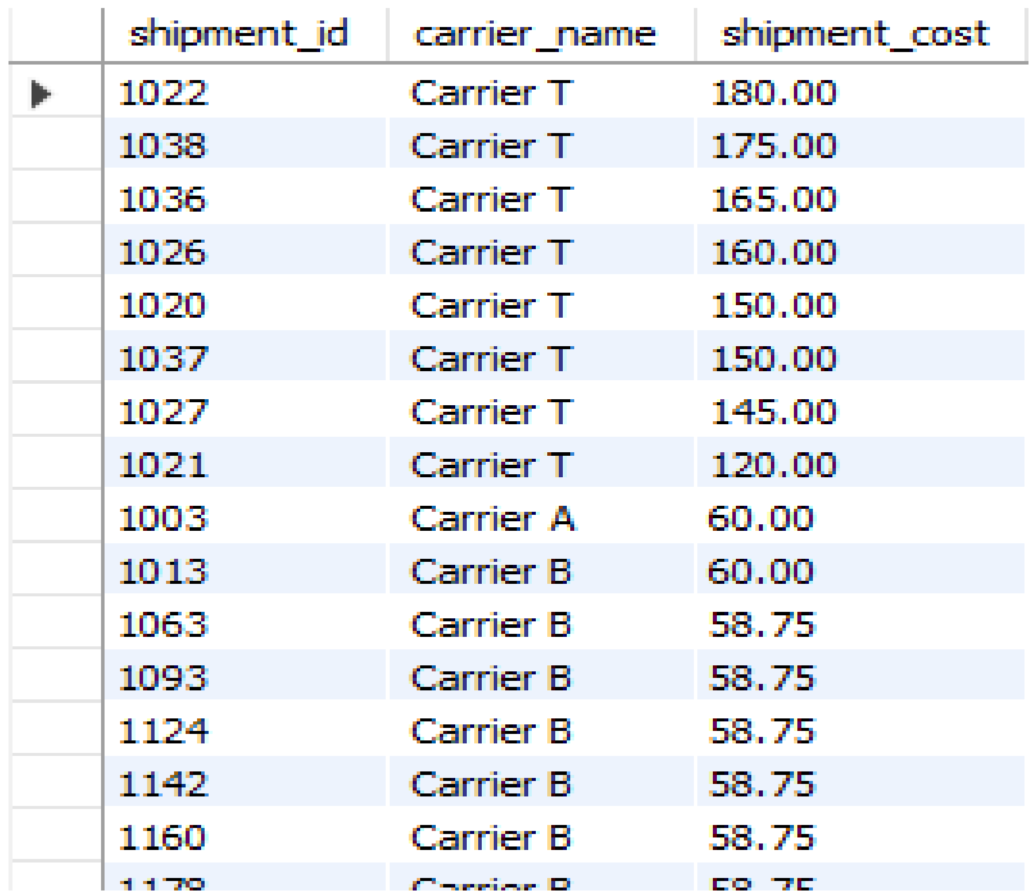
Shipping cost by each shipment

MySQL QUERY



```
SELECT
    ss.shipment_id,
    ss.carrier_name,
    ss.shipment_cost
FROM
    dim_shipments ss
ORDER BY
    ss.shipment_cost DESC;
```

OUTPUT

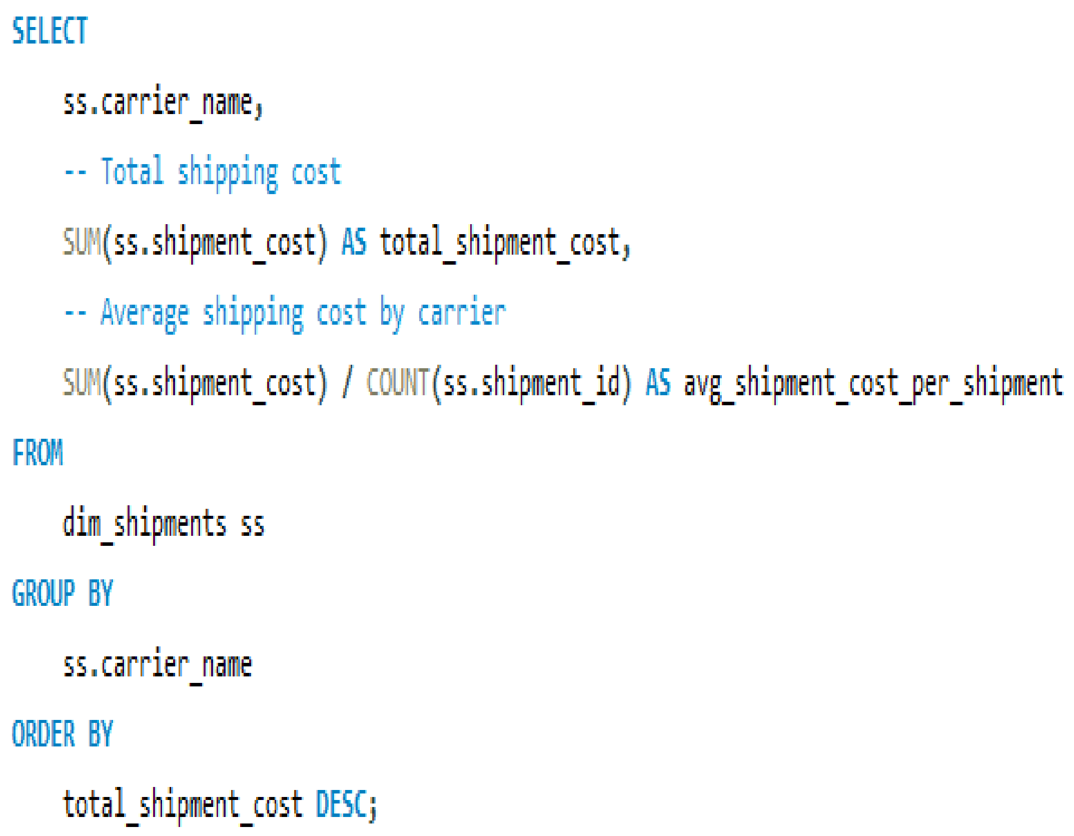


shipment_id	carrier_name	shipment_cost
1022	Carrier T	180.00
1038	Carrier T	175.00
1036	Carrier T	165.00
1026	Carrier T	160.00
1020	Carrier T	150.00
1037	Carrier T	150.00
1027	Carrier T	145.00
1021	Carrier T	120.00
1003	Carrier A	60.00
1013	Carrier B	60.00
1063	Carrier B	58.75
1093	Carrier B	58.75
1124	Carrier B	58.75
1142	Carrier B	58.75
1160	Carrier B	58.75

Presented by ***Ngoc Anh Nguyen***

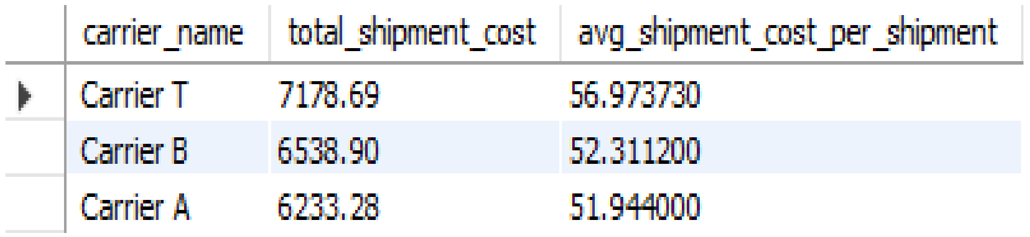
Shipping cost by carrier

MySQL QUERY



```sql
SELECT
    ss.carrier_name,
    -- Total shipping cost
    SUM(ss.shipment_cost) AS total_shipment_cost,
    -- Average shipping cost by carrier
    SUM(ss.shipment_cost) / COUNT(ss.shipment_id) AS avg_shipment_cost_per_shipment
FROM
    dim_shipments ss
GROUP BY
    ss.carrier_name
ORDER BY
    total_shipment_cost DESC;
```

OUTPUT



carrier_name	total_shipment_cost	avg_shipment_cost_per_shipment
Carrier T	7178.69	56.973730
Carrier B	6538.90	52.311200
Carrier A	6233.28	51.944000

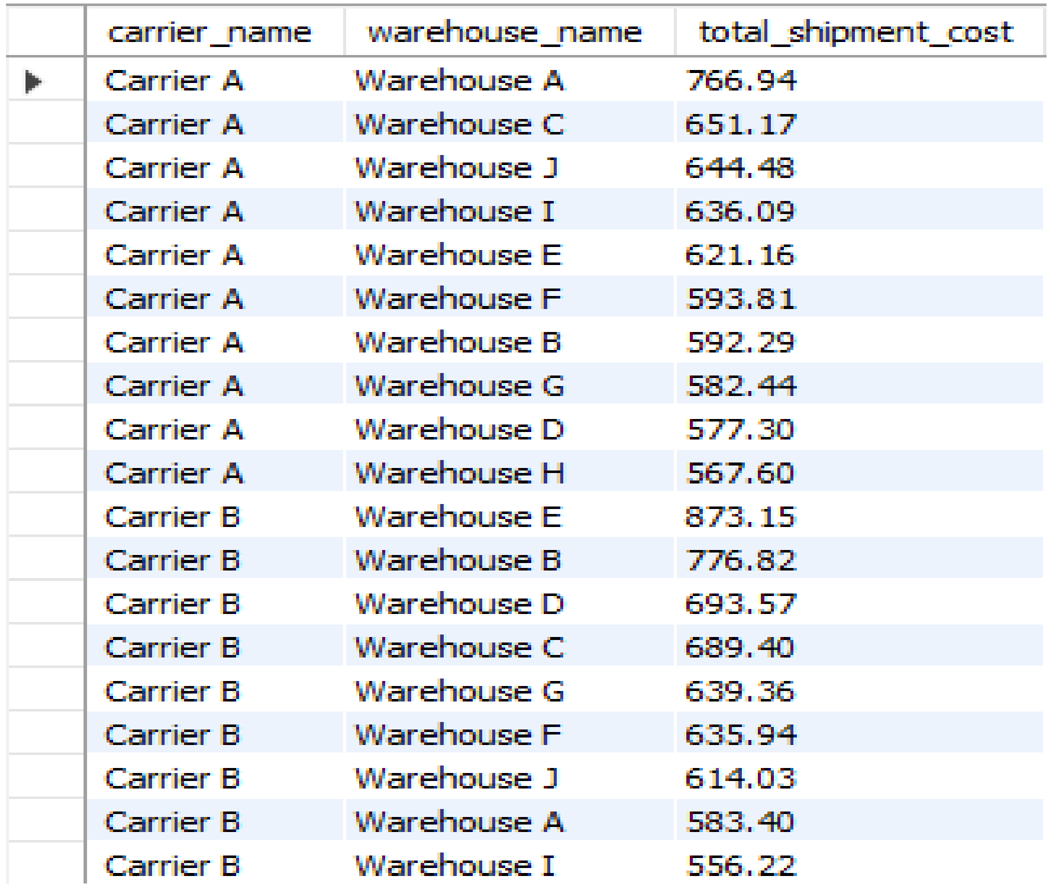
Presented by ***Ngoc Anh Nguyen***

Shipping cost by carrier and warehouse

MySQL QUERY

```sql
SELECT
    ss.carrier_name,
    dw.warehouse_name,
    SUM(ss.shipment_cost) AS total_shipment_cost
FROM
    dim_shipments ss
JOIN
    fact_sales fs ON ss.shipment_id = fs.shipment_id
JOIN
    dim_warehouses dw ON fs.warehouse_id = dw.warehouse_id
GROUP BY
    dw.warehouse_name, ss.carrier_name
ORDER BY
    ss.carrier_name, total_shipment_cost DESC;
```

OUTPUT

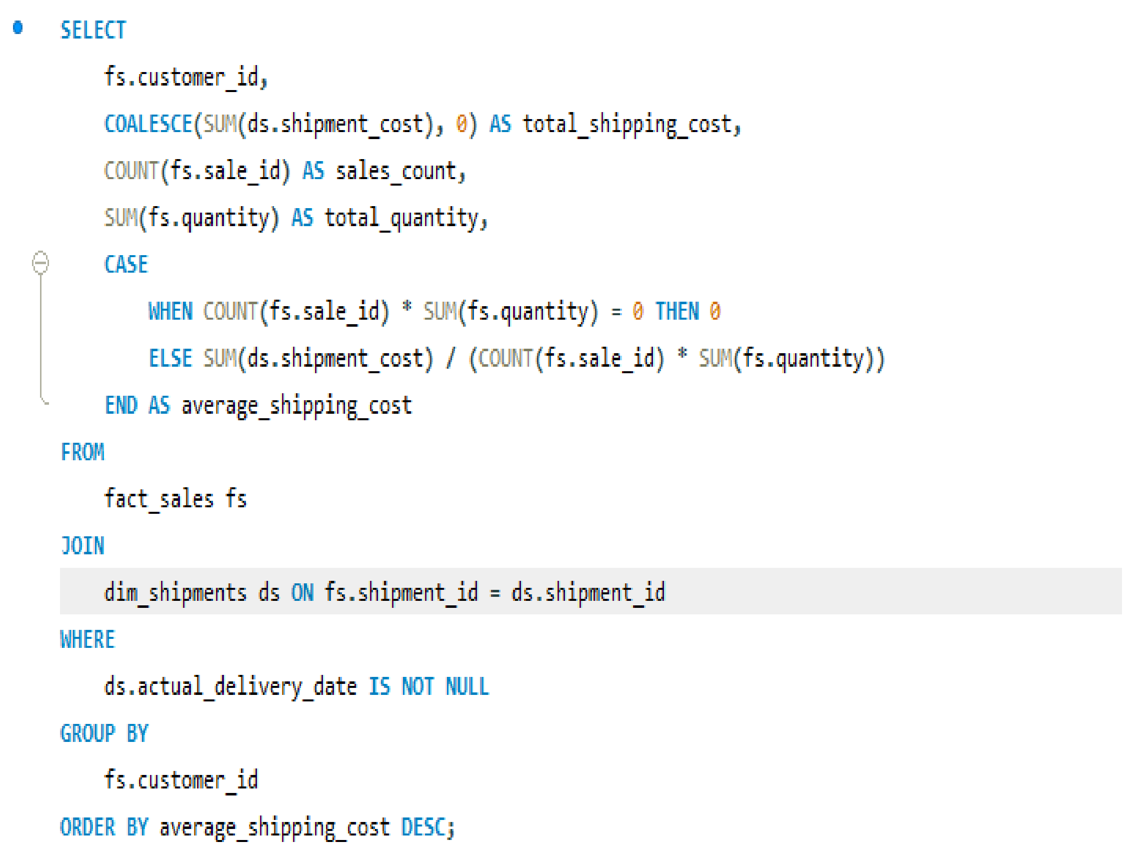


carrier_name	warehouse_name	total_shipment_cost
Carrier A	Warehouse A	766.94
Carrier A	Warehouse C	651.17
Carrier A	Warehouse J	644.48
Carrier A	Warehouse I	636.09
Carrier A	Warehouse E	621.16
Carrier A	Warehouse F	593.81
Carrier A	Warehouse B	592.29
Carrier A	Warehouse G	582.44
Carrier A	Warehouse D	577.30
Carrier A	Warehouse H	567.60
Carrier B	Warehouse E	873.15
Carrier B	Warehouse B	776.82
Carrier B	Warehouse D	693.57
Carrier B	Warehouse C	689.40
Carrier B	Warehouse G	639.36
Carrier B	Warehouse F	635.94
Carrier B	Warehouse J	614.03
Carrier B	Warehouse A	583.40
Carrier B	Warehouse I	556.22

Presented by ***Ngoc Anh Nguyen***

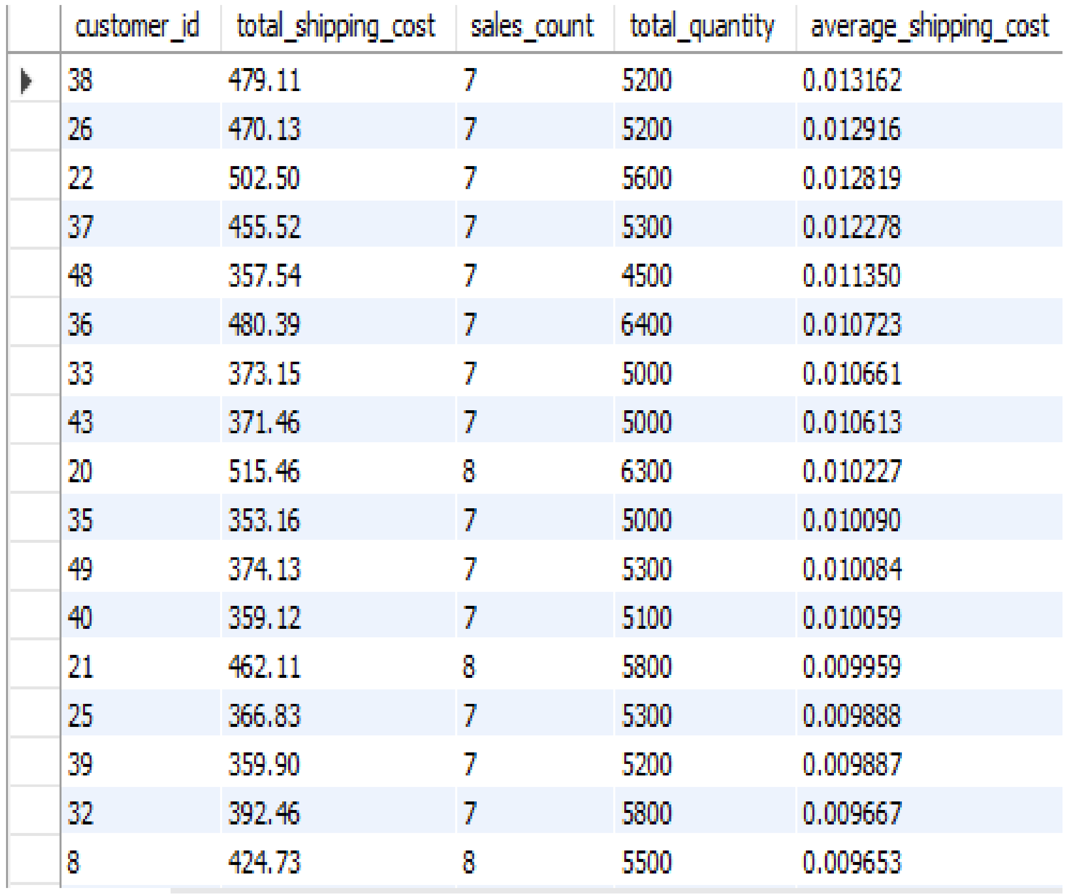
Shipping cost by customer

MySQL QUERY



```
SELECT
    fs.customer_id,
    COALESCE(SUM(ds.shipment_cost), 0) AS total_shipping_cost,
    COUNT(fs.sale_id) AS sales_count,
    SUM(fs.quantity) AS total_quantity,
    CASE
        WHEN COUNT(fs.sale_id) * SUM(fs.quantity) = 0 THEN 0
        ELSE SUM(ds.shipment_cost) / (COUNT(fs.sale_id) * SUM(fs.quantity))
    END AS average_shipping_cost
FROM
    fact_sales fs
JOIN
    dim_shipments ds ON fs.shipment_id = ds.shipment_id
WHERE
    ds.actual_delivery_date IS NOT NULL
GROUP BY
    fs.customer_id
ORDER BY average_shipping_cost DESC;
```

OUTPUT



customer_id	total_shipping_cost	sales_count	total_quantity	average_shipping_cost
38	479.11	7	5200	0.013162
26	470.13	7	5200	0.012916
22	502.50	7	5600	0.012819
37	455.52	7	5300	0.012278
48	357.54	7	4500	0.011350
36	480.39	7	6400	0.010723
33	373.15	7	5000	0.010661
43	371.46	7	5000	0.010613
20	515.46	8	6300	0.010227
35	353.16	7	5000	0.010090
49	374.13	7	5300	0.010084
40	359.12	7	5100	0.010059
21	462.11	8	5800	0.009959
25	366.83	7	5300	0.009888
39	359.90	7	5200	0.009887
32	392.46	7	5800	0.009667
8	424.73	8	5500	0.009653

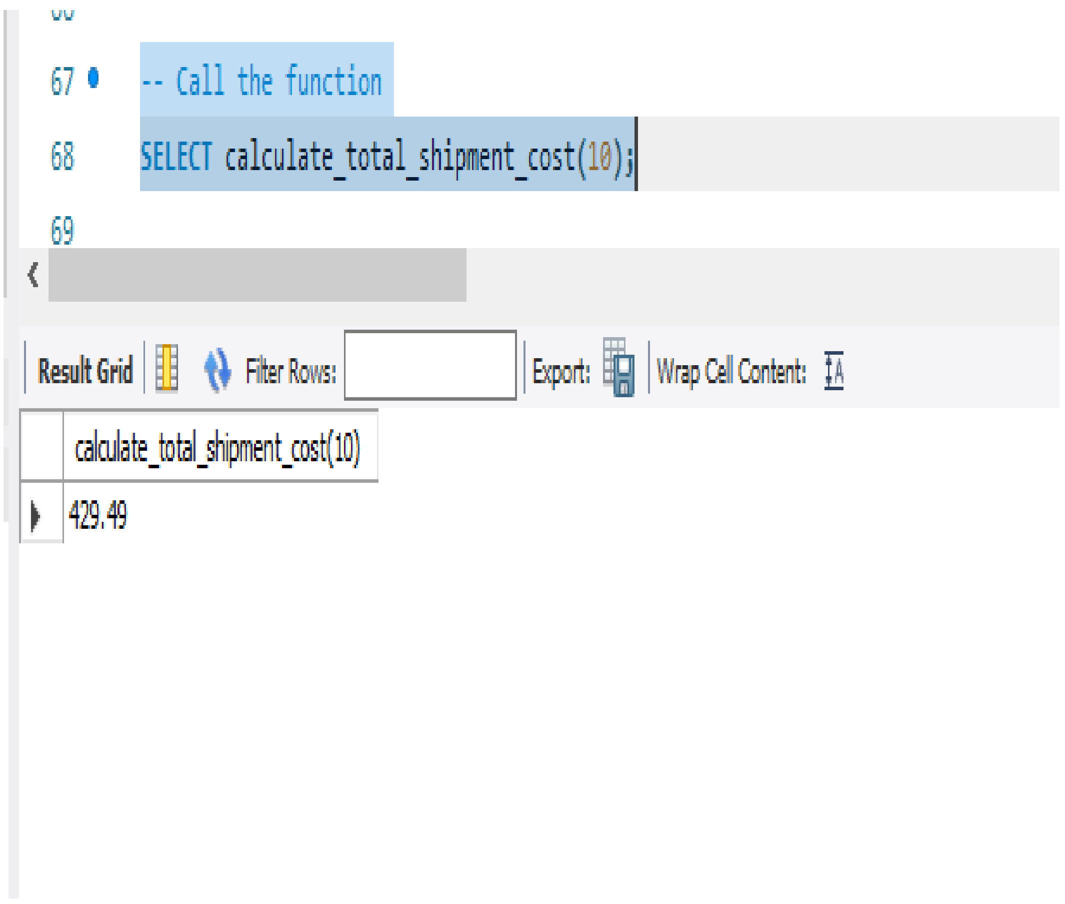
Presented by ***Ngoc Anh Nguyen***

FUNCTION: Shipping cost by customer

MySQL QUERY

```
DELIMITER //

CREATE FUNCTION calculate_total_shipment_cost(customer_id INT)
RETURNS DECIMAL(10, 2)
DETERMINISTIC
BEGIN
    DECLARE total_cost DECIMAL(10, 2);

    -- Total cost
    SELECT COALESCE(SUM(ds.shipment_cost), 0) INTO total_cost
    FROM fact_sales fs
    JOIN dim_shipments ds ON fs.shipment_id = ds.shipment_id
    WHERE ds.actual_delivery_date IS NOT NULL
    AND (customer_id IS NULL OR fs.customer_id = customer_id);

    -- Output
    RETURN total_cost;
END //

DELIMITER ;
```

OUTPUT



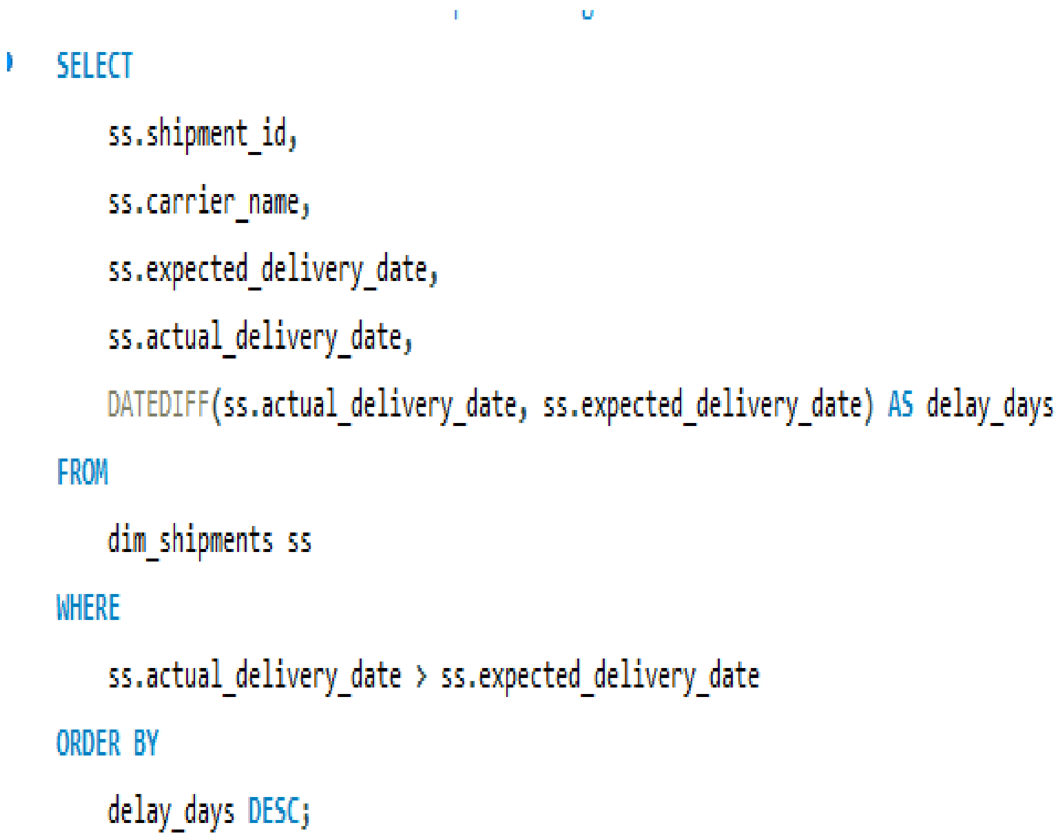
```
-- Call the function
SELECT calculate_total_shipment_cost(10);
```

```
calculate_total_shipment_cost(10)
429.49
```

Presented by ***Ngoc Anh Nguyen***

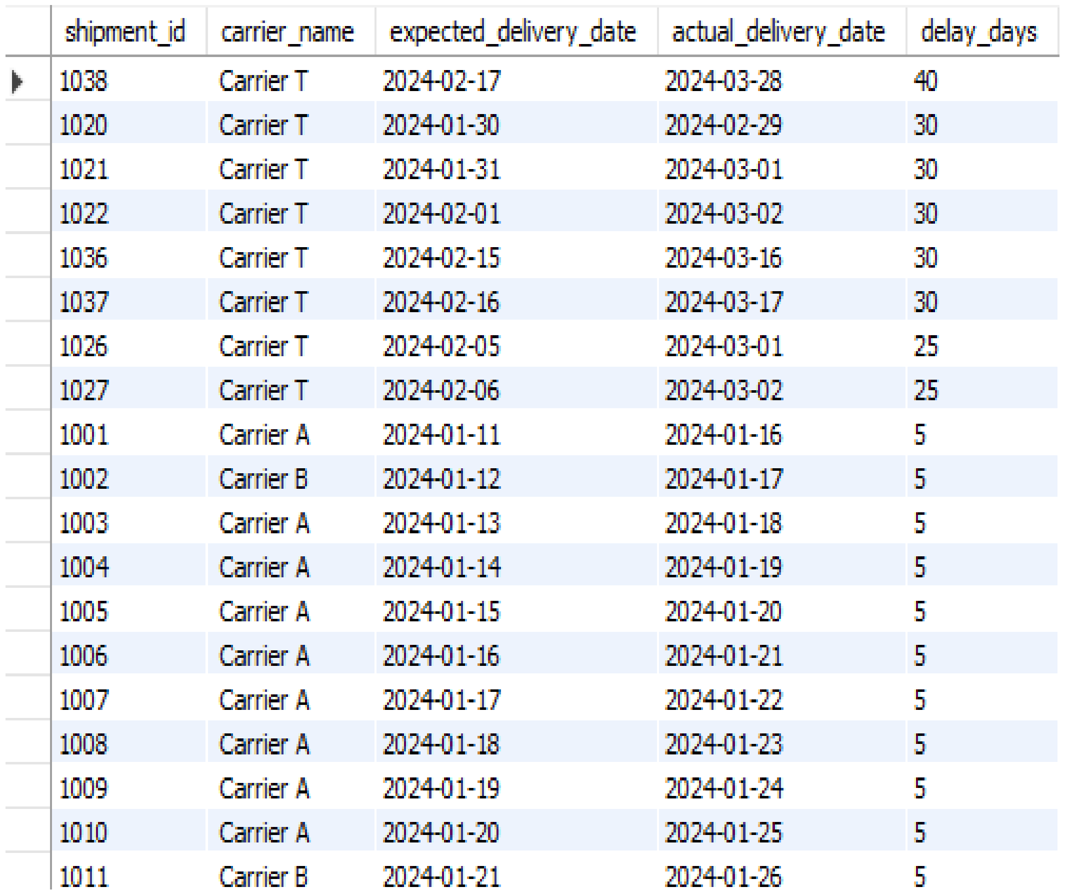
Delay time by shipment

MySQL QUERY



```
SELECT
    ss.shipment_id,
    ss.carrier_name,
    ss.expected_delivery_date,
    ss.actual_delivery_date,
    DATEDIFF(ss.actual_delivery_date, ss.expected_delivery_date) AS delay_days
FROM
    dim_shipments ss
WHERE
    ss.actual_delivery_date > ss.expected_delivery_date
ORDER BY
    delay_days DESC;
```

OUTPUT

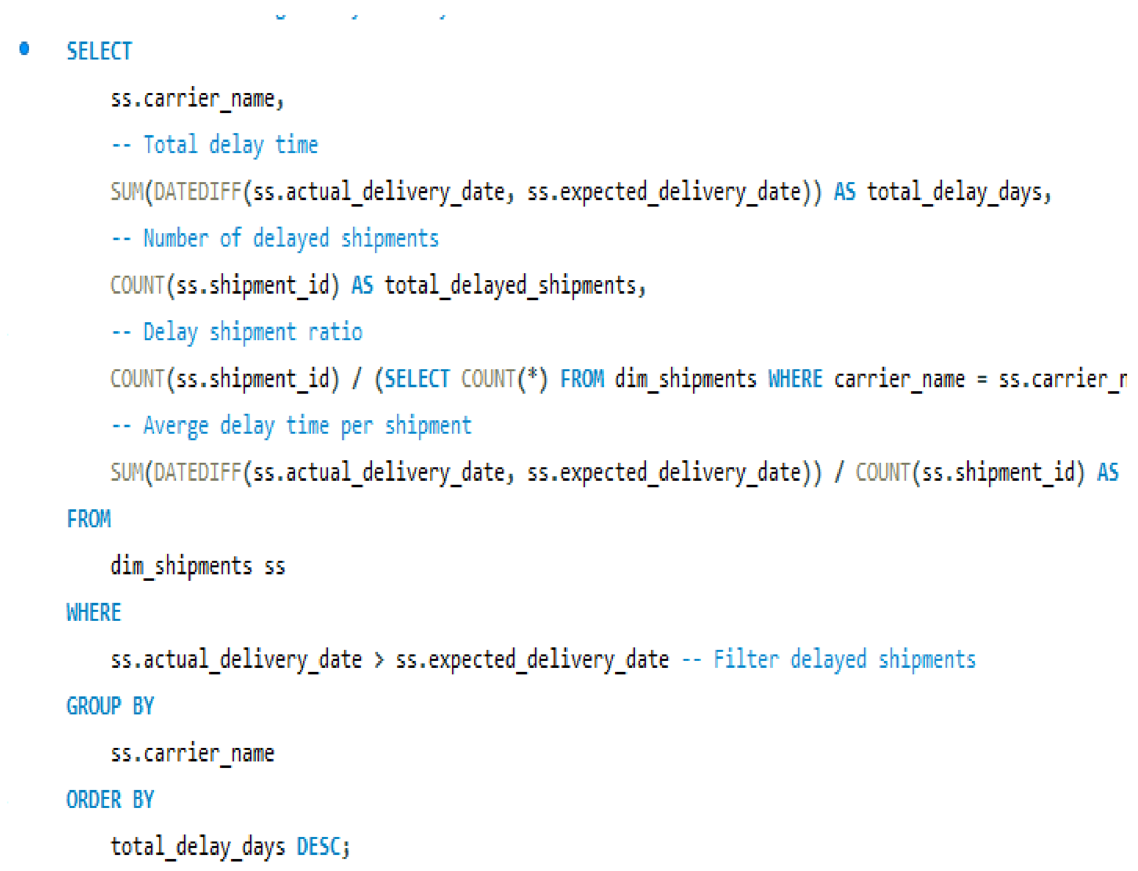


shipment_id	carrier_name	expected_delivery_date	actual_delivery_date	delay_days
1038	Carrier T	2024-02-17	2024-03-28	40
1020	Carrier T	2024-01-30	2024-02-29	30
1021	Carrier T	2024-01-31	2024-03-01	30
1022	Carrier T	2024-02-01	2024-03-02	30
1036	Carrier T	2024-02-15	2024-03-16	30
1037	Carrier T	2024-02-16	2024-03-17	30
1026	Carrier T	2024-02-05	2024-03-01	25
1027	Carrier T	2024-02-06	2024-03-02	25
1001	Carrier A	2024-01-11	2024-01-16	5
1002	Carrier B	2024-01-12	2024-01-17	5
1003	Carrier A	2024-01-13	2024-01-18	5
1004	Carrier A	2024-01-14	2024-01-19	5
1005	Carrier A	2024-01-15	2024-01-20	5
1006	Carrier A	2024-01-16	2024-01-21	5
1007	Carrier A	2024-01-17	2024-01-22	5
1008	Carrier A	2024-01-18	2024-01-23	5
1009	Carrier A	2024-01-19	2024-01-24	5
1010	Carrier A	2024-01-20	2024-01-25	5
1011	Carrier B	2024-01-21	2024-01-26	5

Presented by ***Ngoc Anh Nguyen***

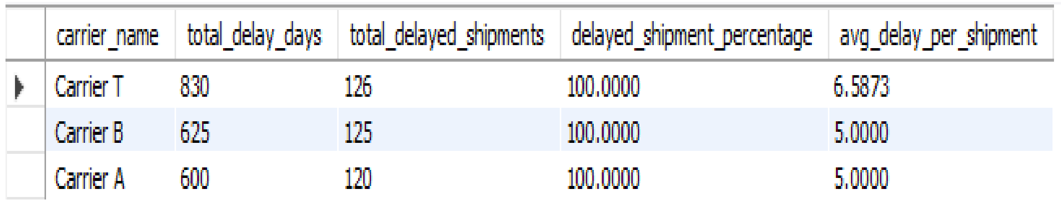
Delay time by carrier

MySQL QUERY



```
SELECT
    ss.carrier_name,
    -- Total delay time
    SUM(DATEDIFF(ss.actual_delivery_date, ss.expected_delivery_date)) AS total_delay_days,
    -- Number of delayed shipments
    COUNT(ss.shipment_id) AS total_delayed_shipments,
    -- Delay shipment ratio
    COUNT(ss.shipment_id) / (SELECT COUNT(*) FROM dim_shipments WHERE carrier_name = ss.carrier_r
    -- Averge delay time per shipment
    SUM(DATEDIFF(ss.actual_delivery_date, ss.expected_delivery_date)) / COUNT(ss.shipment_id) AS
FROM
    dim_shipments ss
WHERE
    ss.actual_delivery_date > ss.expected_delivery_date -- Filter delayed shipments
GROUP BY
    ss.carrier_name
ORDER BY
    total_delay_days DESC;
```

OUTPUT

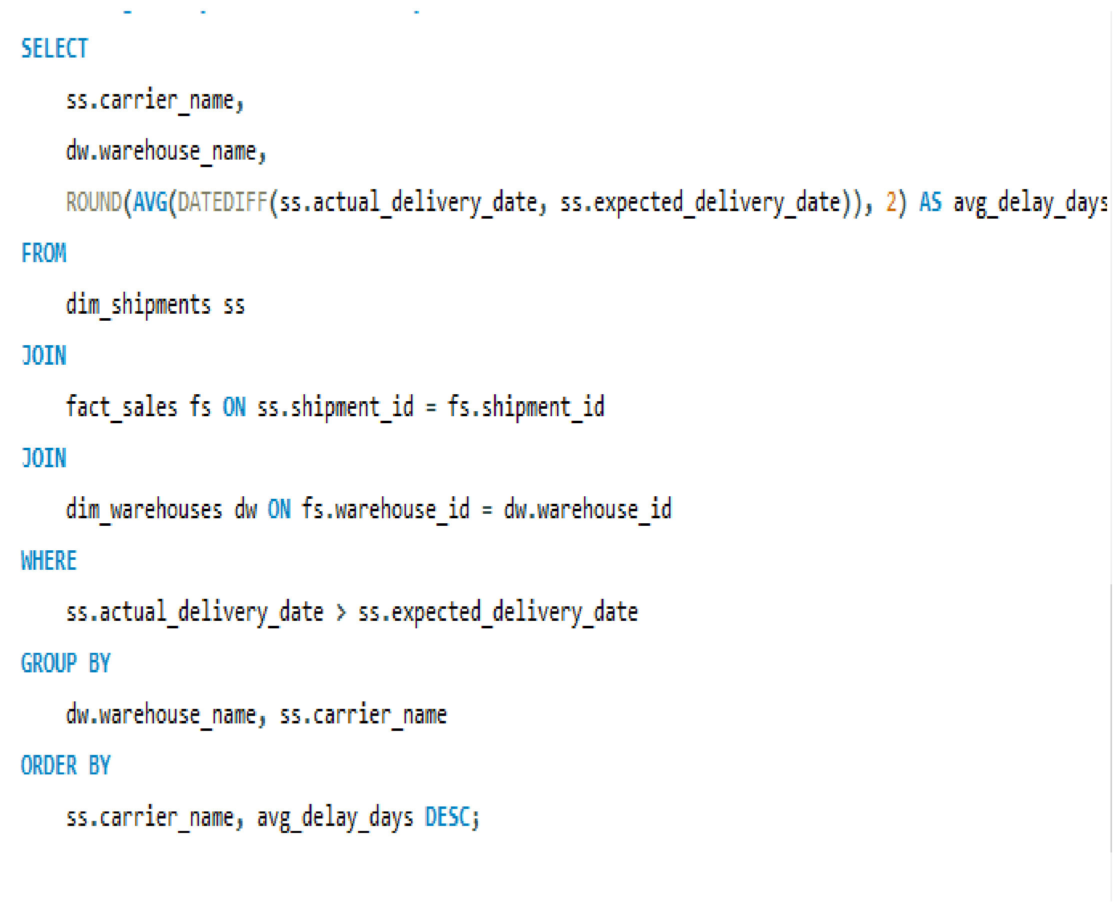


carrier_name	total_delay_days	total_delayed_shipments	delayed_shipment_percentage	avg_delay_per_shipment
Carrier T	830	126	100.0000	6.5873
Carrier B	625	125	100.0000	5.0000
Carrier A	600	120	100.0000	5.0000

Presented by ***Ngoc Anh Nguyen***

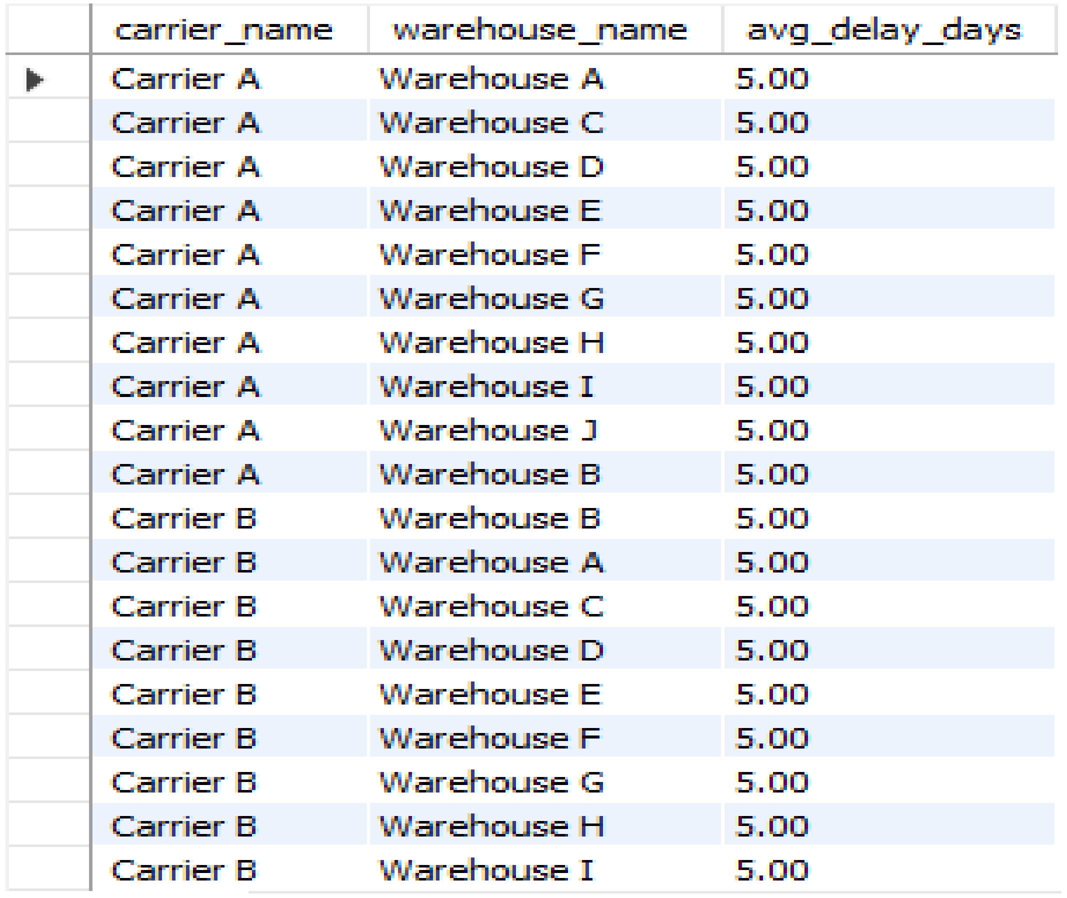
Delay time by warehouse

MySQL QUERY



```
SELECT
    ss.carrier_name,
    dw.warehouse_name,
    ROUND(AVG(DATEDIFF(ss.actual_delivery_date, ss.expected_delivery_date)), 2) AS avg_delay_days
FROM
    dim_shipments ss
JOIN
    fact_sales fs ON ss.shipment_id = fs.shipment_id
JOIN
    dim_warehouses dw ON fs.warehouse_id = dw.warehouse_id
WHERE
    ss.actual_delivery_date > ss.expected_delivery_date
GROUP BY
    dw.warehouse_name, ss.carrier_name
ORDER BY
    ss.carrier_name, avg_delay_days DESC;
```

OUTPUT



carrier_name	warehouse_name	avg_delay_days
Carrier A	Warehouse A	5.00
Carrier A	Warehouse C	5.00
Carrier A	Warehouse D	5.00
Carrier A	Warehouse E	5.00
Carrier A	Warehouse F	5.00
Carrier A	Warehouse G	5.00
Carrier A	Warehouse H	5.00
Carrier A	Warehouse I	5.00
Carrier A	Warehouse J	5.00
Carrier A	Warehouse B	5.00
Carrier B	Warehouse B	5.00
Carrier B	Warehouse A	5.00
Carrier B	Warehouse C	5.00
Carrier B	Warehouse D	5.00
Carrier B	Warehouse E	5.00
Carrier B	Warehouse F	5.00
Carrier B	Warehouse G	5.00
Carrier B	Warehouse H	5.00
Carrier B	Warehouse I	5.00

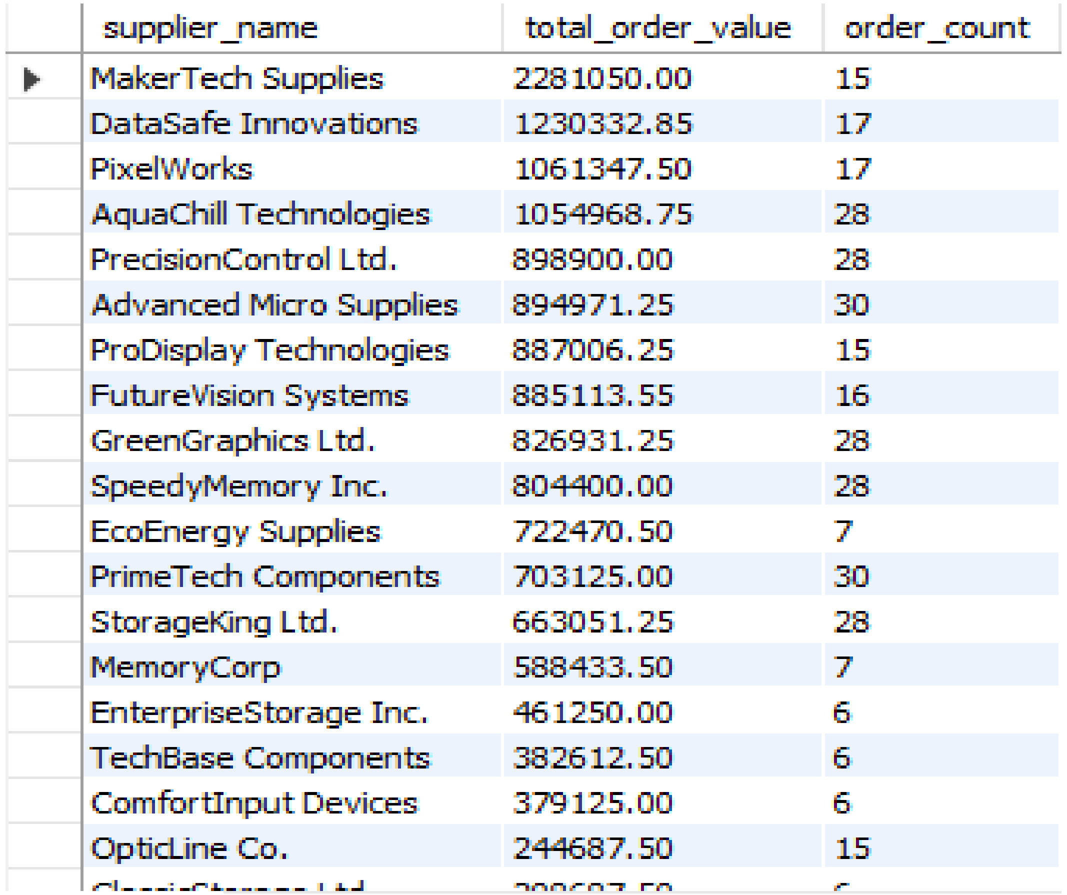
Presented by ***Ngoc Anh Nguyen***

Number & value of order by supplier

MySQL QUERY

```sql
SELECT
    ds.supplier_name,
    SUM(po.quantity * dr.raw_price) AS total_order_value,
    COUNT(po.purchase_order_id) AS order_count
FROM
    fact_purchasing_order po
JOIN
    dim_suppliers ds ON po.supplier_id = ds.supplier_id
JOIN
    dim_raw_product dr ON po.product_id = dr.raw_product_id
GROUP BY
    ds.supplier_name
ORDER BY
    total_order_value DESC;
```

OUTPUT

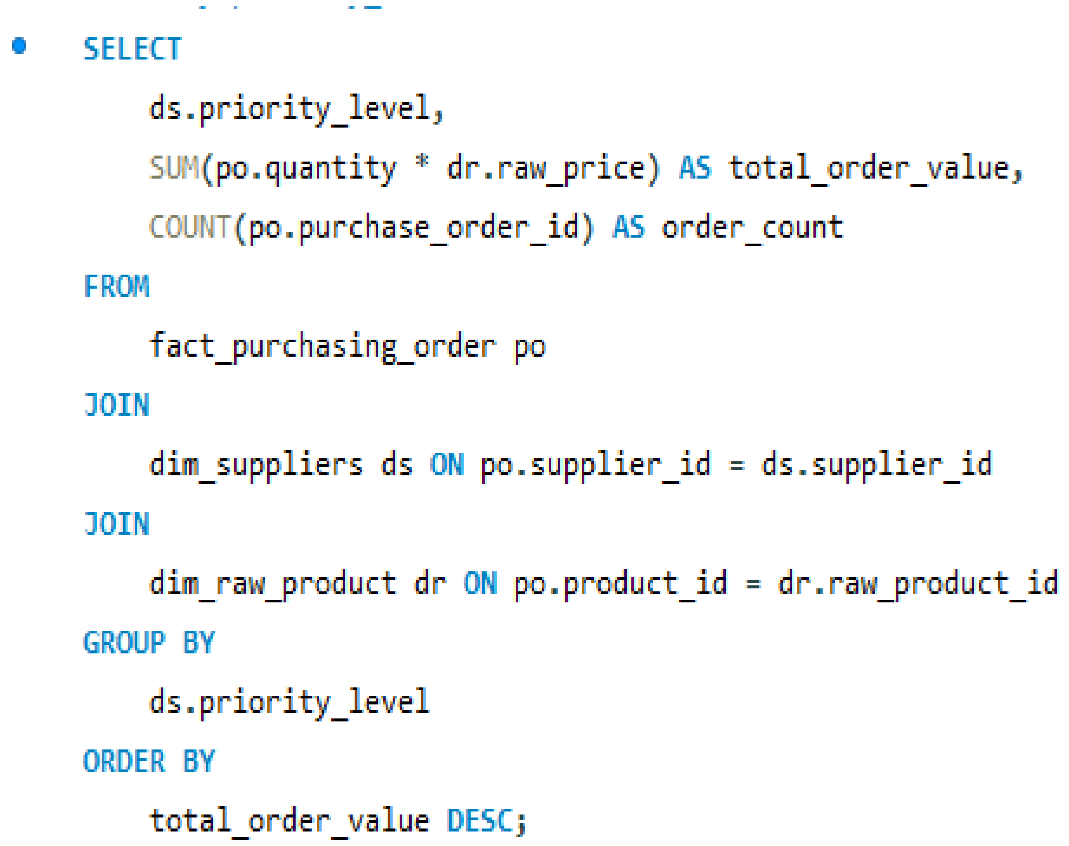


supplier_name	total_order_value	order_count
MakerTech Supplies	2281050.00	15
DataSafe Innovations	1230332.85	17
PixelWorks	1061347.50	17
AquaChill Technologies	1054968.75	28
PrecisionControl Ltd.	898900.00	28
Advanced Micro Supplies	894971.25	30
ProDisplay Technologies	887006.25	15
FutureVision Systems	885113.55	16
GreenGraphics Ltd.	826931.25	28
SpeedyMemory Inc.	804400.00	28
EcoEnergy Supplies	722470.50	7
PrimeTech Components	703125.00	30
StorageKing Ltd.	663051.25	28
MemoryCorp	588433.50	7
EnterpriseStorage Inc.	461250.00	6
TechBase Components	382612.50	6
ComfortInput Devices	379125.00	6
OpticLine Co.	244687.50	15
[illegible]	[illegible]	[illegible]

Presented by ***Ngoc Anh Nguyen***

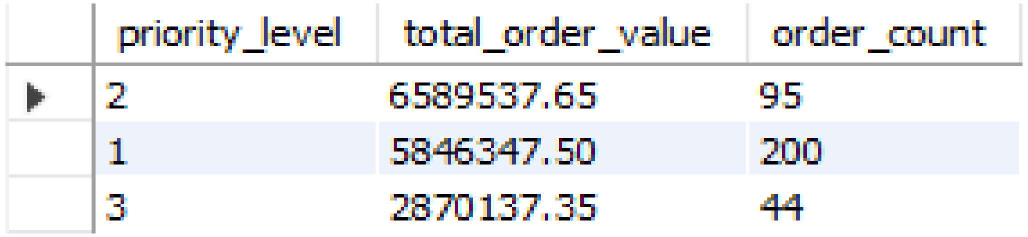
Number & value of order by priority level

MySQL QUERY



```
SELECT
    ds.priority_level,
    SUM(po.quantity * dr.raw_price) AS total_order_value,
    COUNT(po.purchase_order_id) AS order_count
FROM
    fact_purchasing_order po
JOIN
    dim_suppliers ds ON po.supplier_id = ds.supplier_id
JOIN
    dim_raw_product dr ON po.product_id = dr.raw_product_id
GROUP BY
    ds.priority_level
ORDER BY
    total_order_value DESC;
```

OUTPUT

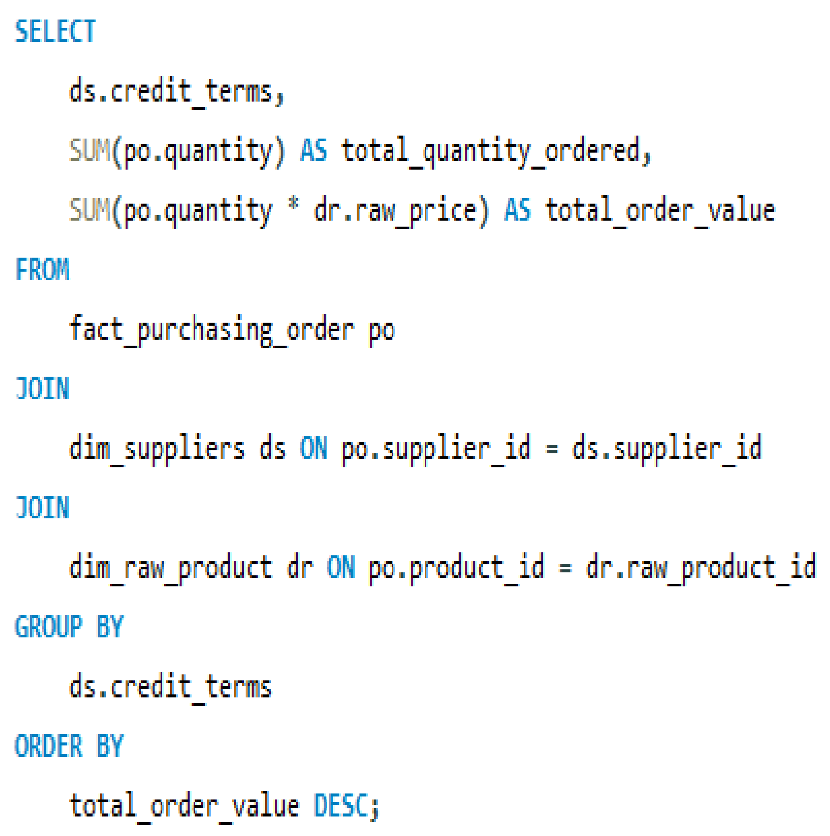


priority_level	total_order_value	order_count
2	6589537.65	95
1	5846347.50	200
3	2870137.35	44

Presented by ***Ngoc Anh Nguyen***

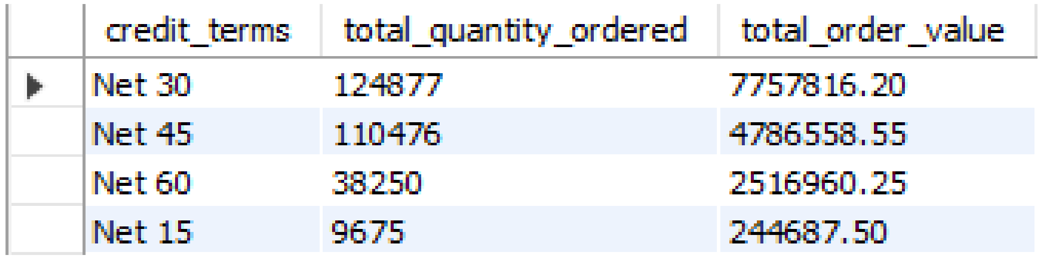
Number & value of order by credit term

MySQL QUERY



```
SELECT
    ds.credit_terms,
    SUM(po.quantity) AS total_quantity_ordered,
    SUM(po.quantity * dr.raw_price) AS total_order_value
FROM
    fact_purchasing_order po
JOIN
    dim_suppliers ds ON po.supplier_id = ds.supplier_id
JOIN
    dim_raw_product dr ON po.product_id = dr.raw_product_id
GROUP BY
    ds.credit_terms
ORDER BY
    total_order_value DESC;
```

OUTPUT

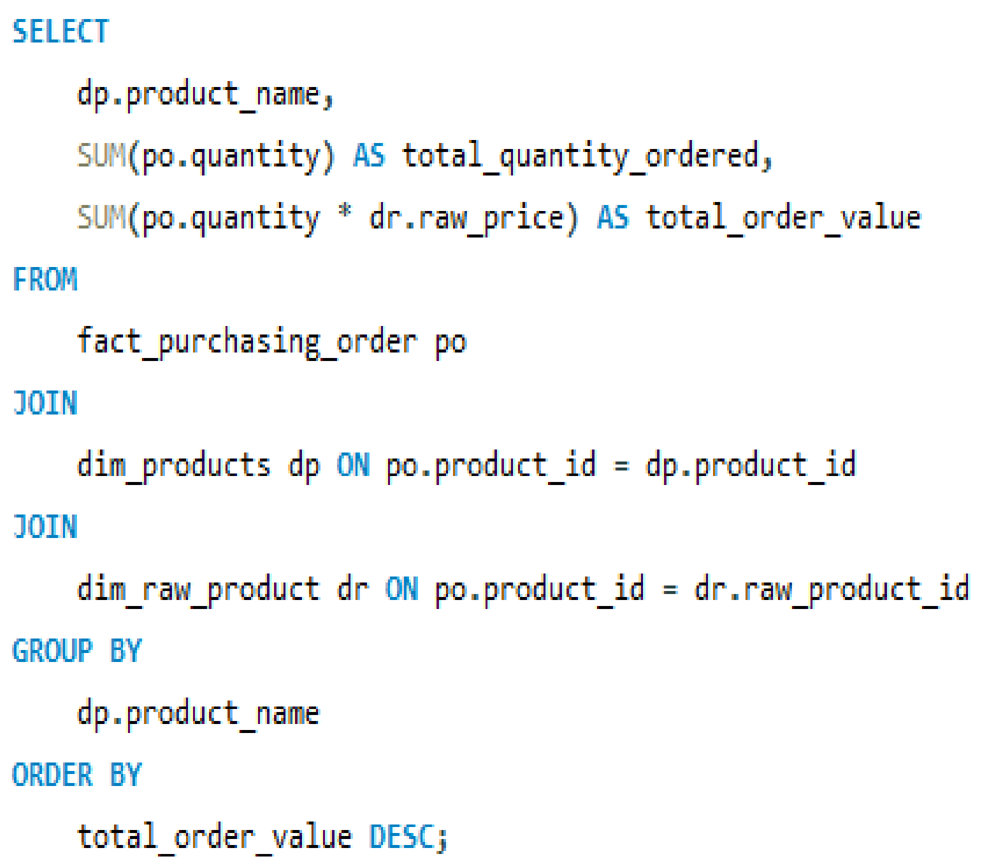


credit_terms	total_quantity_ordered	total_order_value
Net 30	124877	7757816.20
Net 45	110476	4786558.55
Net 60	38250	2516960.25
Net 15	9675	244687.50

Presented by ***Ngoc Anh Nguyen***

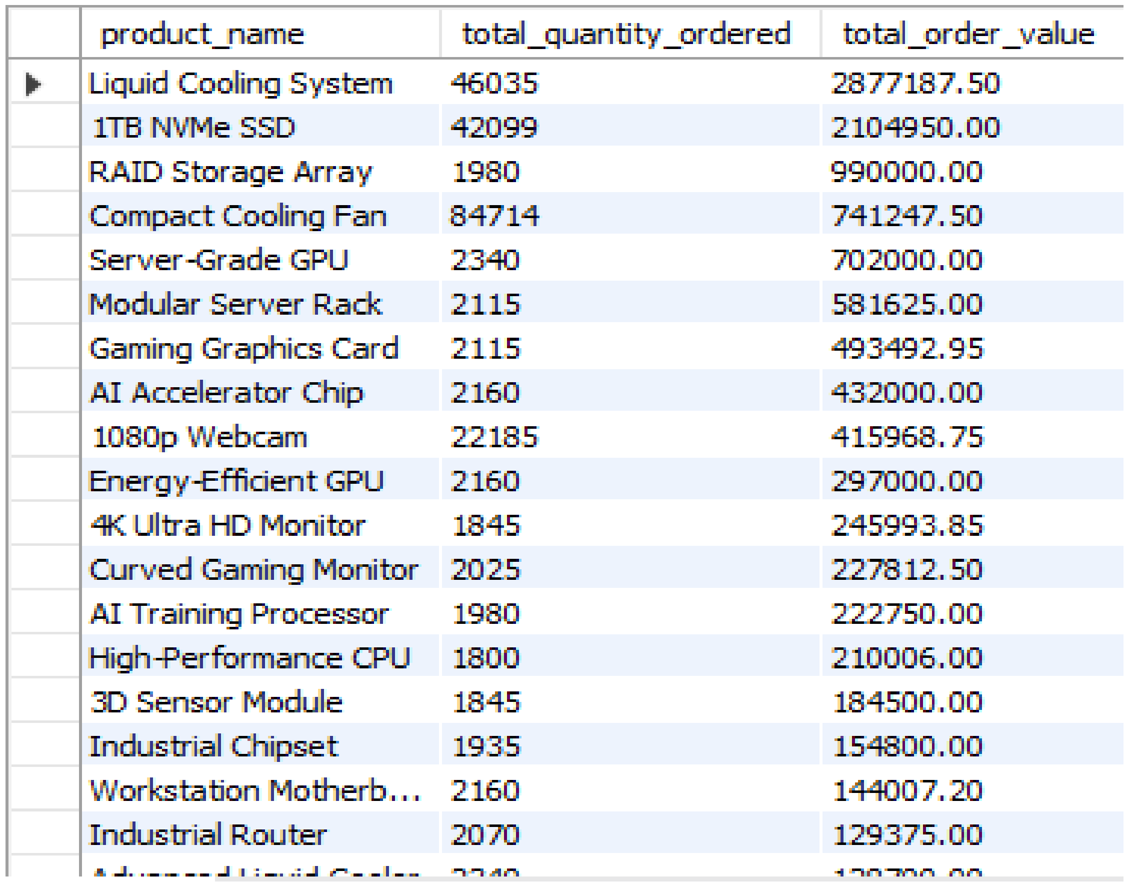
Number & value of order by product

MySQL QUERY



```
SELECT
    dp.product_name,
    SUM(po.quantity) AS total_quantity_ordered,
    SUM(po.quantity * dr.raw_price) AS total_order_value
FROM
    fact_purchasing_order po
JOIN
    dim_products dp ON po.product_id = dp.product_id
JOIN
    dim_raw_product dr ON po.product_id = dr.raw_product_id
GROUP BY
    dp.product_name
ORDER BY
    total_order_value DESC;
```

OUTPUT

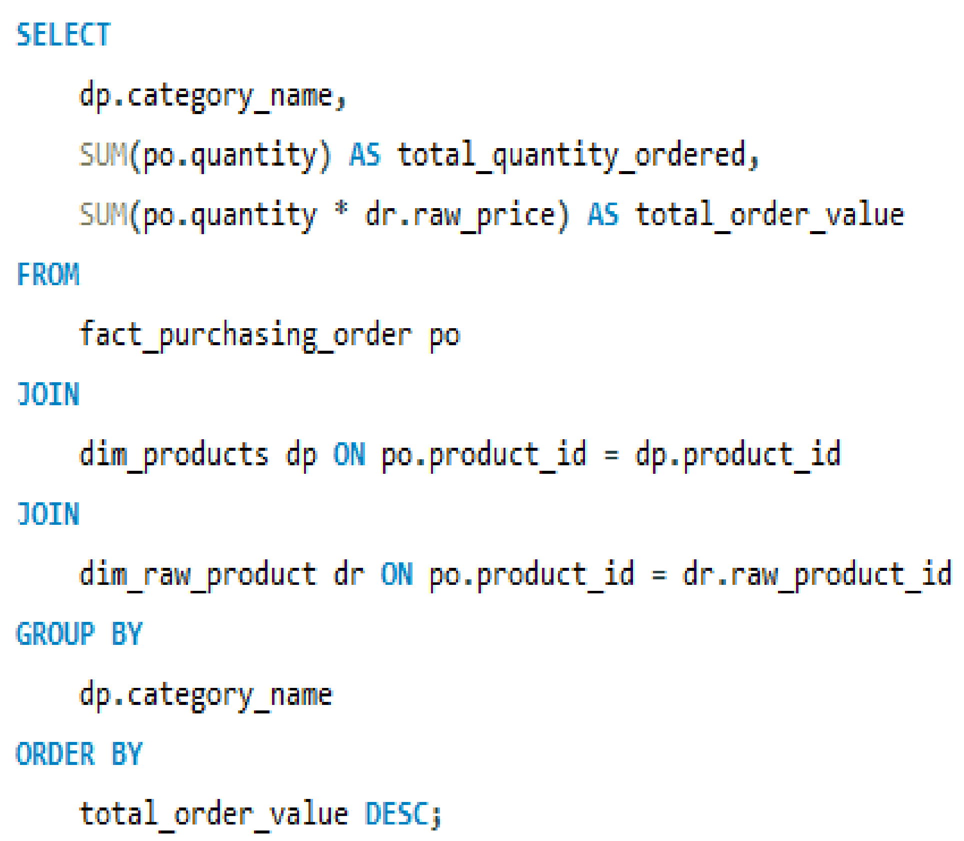


product_name	total_quantity_ordered	total_order_value
Liquid Cooling System	46035	2877187.50
1TB NVMe SSD	42099	2104950.00
RAID Storage Array	1980	990000.00
Compact Cooling Fan	84714	741247.50
Server-Grade GPU	2340	702000.00
Modular Server Rack	2115	581625.00
Gaming Graphics Card	2115	493492.95
AI Accelerator Chip	2160	432000.00
1080p Webcam	22185	415968.75
Energy-Efficient GPU	2160	297000.00
4K Ultra HD Monitor	1845	245993.85
Curved Gaming Monitor	2025	227812.50
AI Training Processor	1980	222750.00
High-Performance CPU	1800	210006.00
3D Sensor Module	1845	184500.00
Industrial Chipset	1935	154800.00
Workstation Motherb...	2160	144007.20
Industrial Router	2070	129375.00
Advanced Liquid Cooler	2340	[illegible]

Presented by ***Ngoc Anh Nguyen***

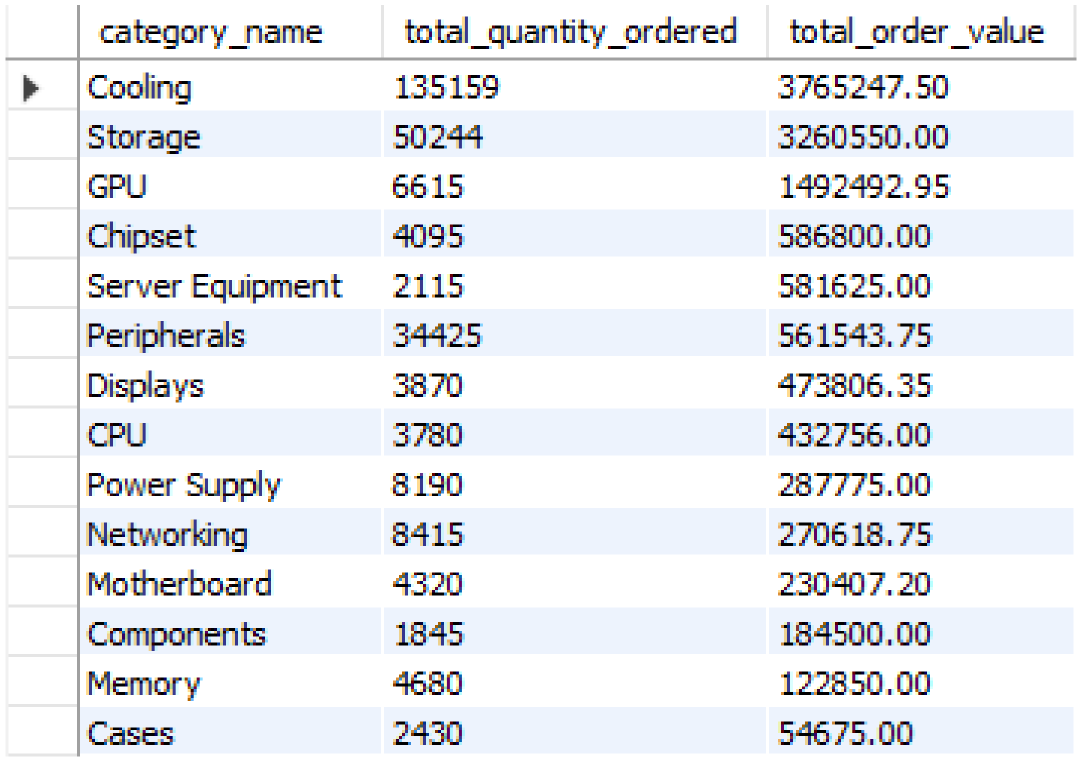
Number & value of order by category

MySQL QUERY



```
SELECT
    dp.category_name,
    SUM(po.quantity) AS total_quantity_ordered,
    SUM(po.quantity * dr.raw_price) AS total_order_value
FROM
    fact_purchasing_order po
JOIN
    dim_products dp ON po.product_id = dp.product_id
JOIN
    dim_raw_product dr ON po.product_id = dr.raw_product_id
GROUP BY
    dp.category_name
ORDER BY
    total_order_value DESC;
```

OUTPUT

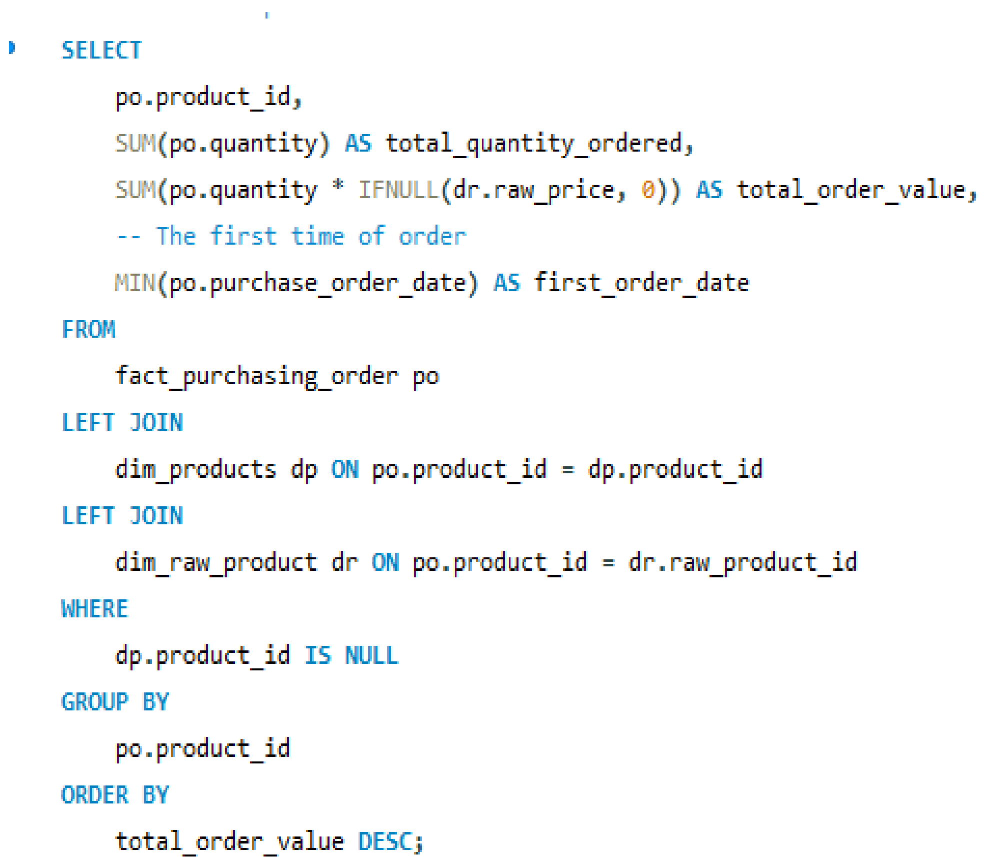


category_name	total_quantity_ordered	total_order_value
Cooling	135159	3765247.50
Storage	50244	3260550.00
GPU	6615	1492492.95
Chipset	4095	586800.00
Server Equipment	2115	581625.00
Peripherals	34425	561543.75
Displays	3870	473806.35
CPU	3780	432756.00
Power Supply	8190	287775.00
Networking	8415	270618.75
Motherboard	4320	230407.20
Components	1845	184500.00
Memory	4680	122850.00
Cases	2430	54675.00

*Presented by **Ngoc Anh Nguyen***

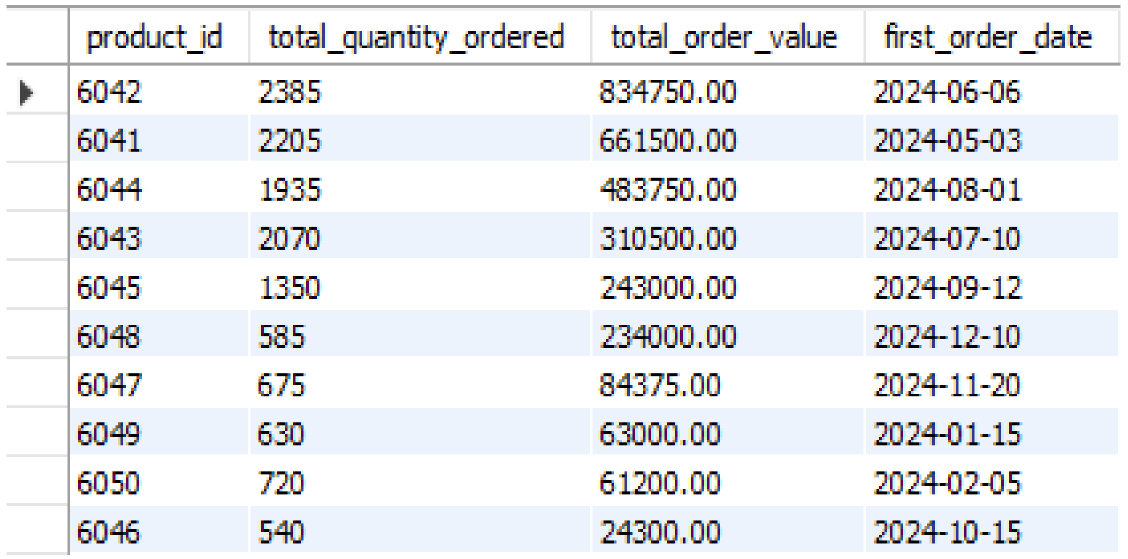
Ordered products but still not sold

MySQL QUERY



```
SELECT
    po.product_id,
    SUM(po.quantity) AS total_quantity_ordered,
    SUM(po.quantity * IFNULL(dr.raw_price, 0)) AS total_order_value,
    -- The first time of order
    MIN(po.purchase_order_date) AS first_order_date
FROM
    fact_purchasing_order po
LEFT JOIN
    dim_products dp ON po.product_id = dp.product_id
LEFT JOIN
    dim_raw_product dr ON po.product_id = dr.raw_product_id
WHERE
    dp.product_id IS NULL
GROUP BY
    po.product_id
ORDER BY
    total_order_value DESC;
```

OUTPUT

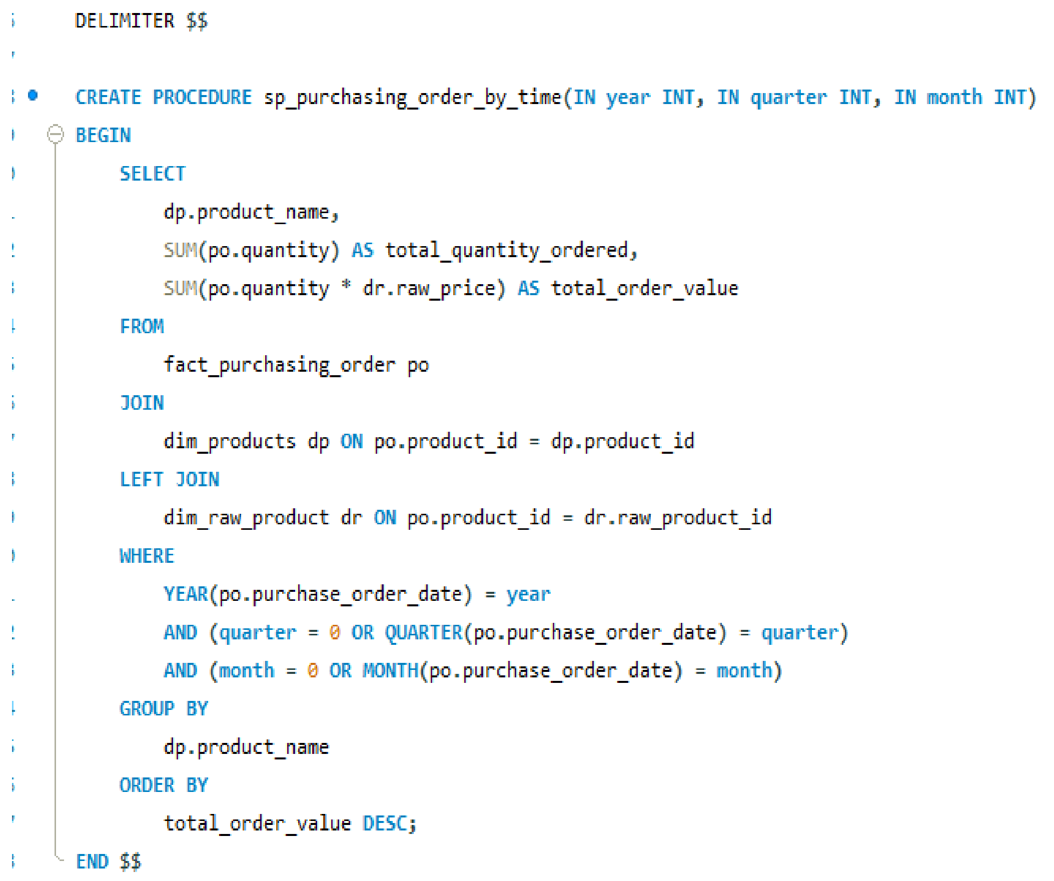


product_id	total_quantity_ordered	total_order_value	first_order_date
6042	2385	834750.00	2024-06-06
6041	2205	661500.00	2024-05-03
6044	1935	483750.00	2024-08-01
6043	2070	310500.00	2024-07-10
6045	1350	243000.00	2024-09-12
6048	585	234000.00	2024-12-10
6047	675	84375.00	2024-11-20
6049	630	63000.00	2024-01-15
6050	720	61200.00	2024-02-05
6046	540	24300.00	2024-10-15

Presented by ***Ngoc Anh Nguyen***

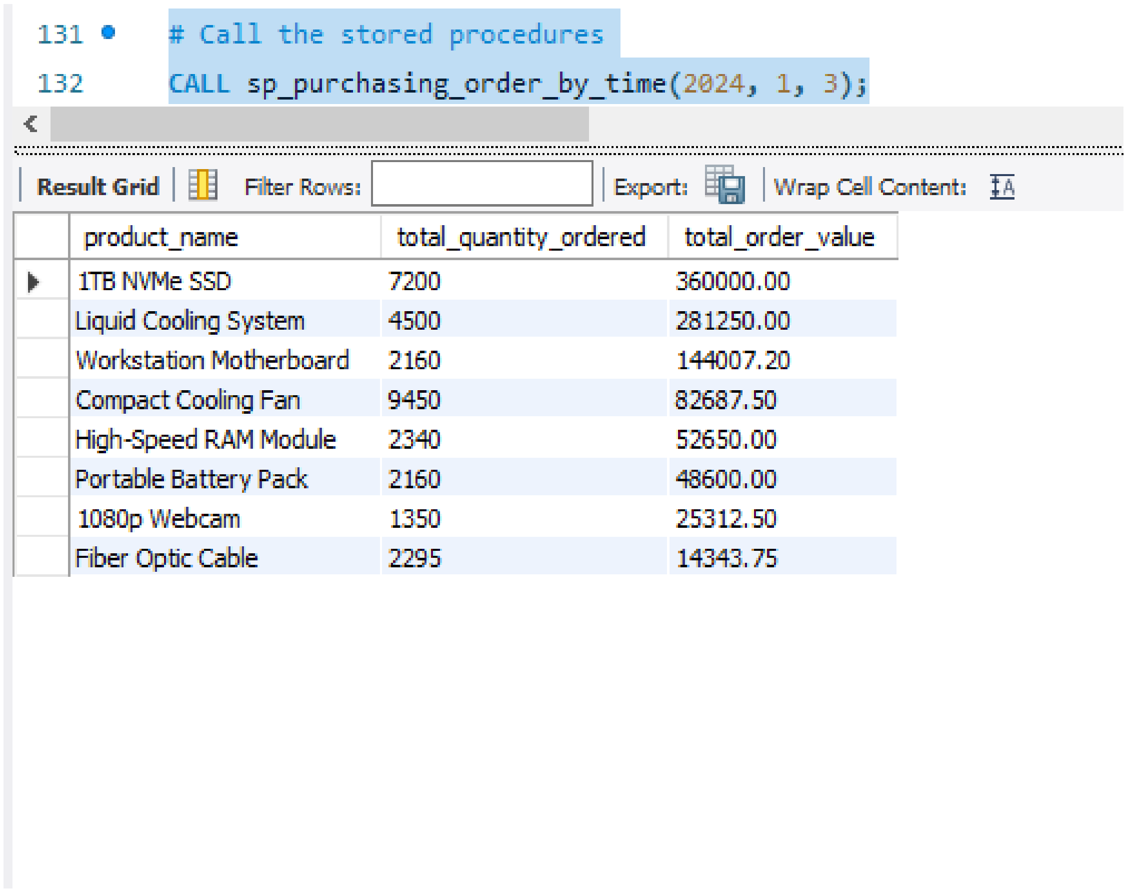
Number & value of order by time and product

MySQL QUERY



```sql
DELIMITER $$

CREATE PROCEDURE sp_purchasing_order_by_time(IN year INT, IN quarter INT, IN month INT)
BEGIN
    SELECT
        dp.product_name,
        SUM(po.quantity) AS total_quantity_ordered,
        SUM(po.quantity * dr.raw_price) AS total_order_value
    FROM
        fact_purchasing_order po
    JOIN
        dim_products dp ON po.product_id = dp.product_id
    LEFT JOIN
        dim_raw_product dr ON po.product_id = dr.raw_product_id
    WHERE
        YEAR(po.purchase_order_date) = year
        AND (quarter = 0 OR QUARTER(po.purchase_order_date) = quarter)
        AND (month = 0 OR MONTH(po.purchase_order_date) = month)
    GROUP BY
        dp.product_name
    ORDER BY
        total_order_value DESC;
END $$
```

OUTPUT



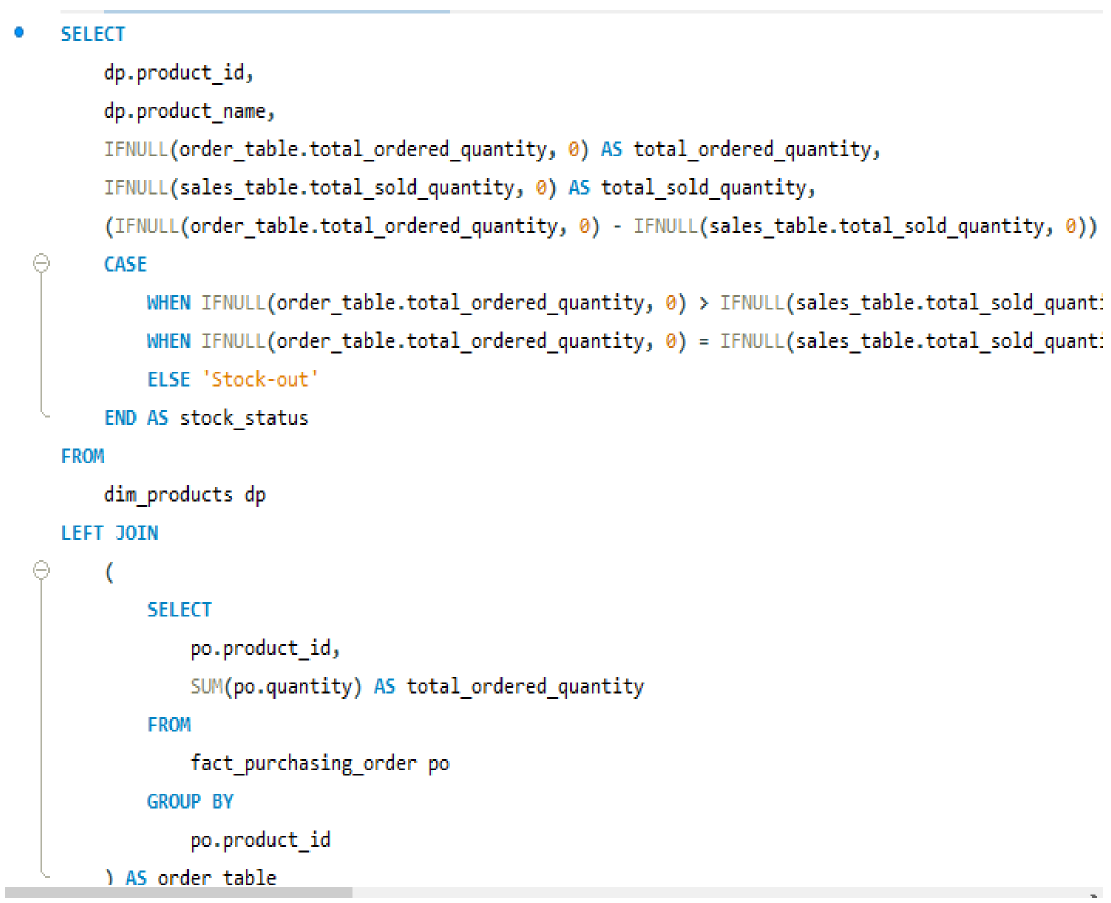
```sql
# Call the stored procedures
CALL sp_purchasing_order_by_time(2024, 1, 3);
```

product_name	total_quantity_ordered	total_order_value
1TB NVMe SSD	7200	360000.00
Liquid Cooling System	4500	281250.00
Workstation Motherboard	2160	144007.20
Compact Cooling Fan	9450	82687.50
High-Speed RAM Module	2340	52650.00
Portable Battery Pack	2160	48600.00
1080p Webcam	1350	25312.50
Fiber Optic Cable	2295	14343.75

Presented by ***Ngoc Anh Nguyen***

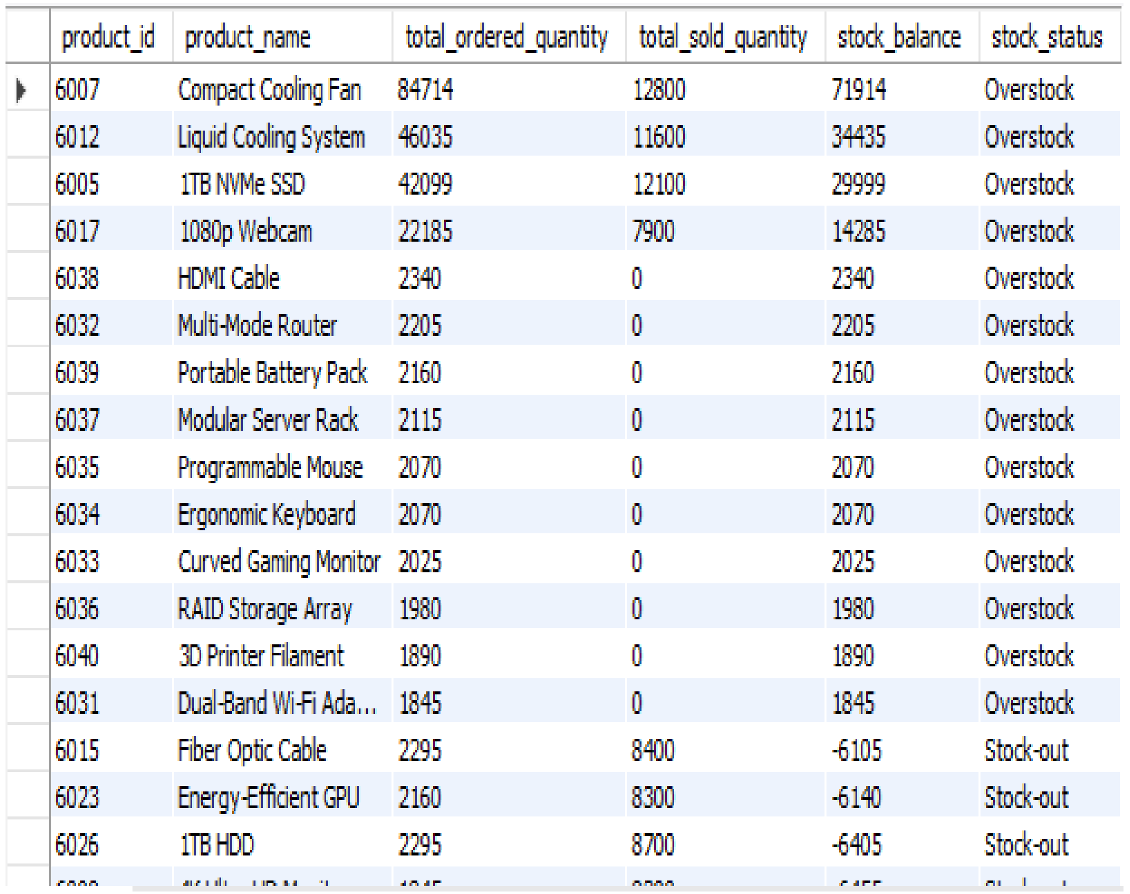
Inventory Analysis

MySQL QUERY



```
SELECT
    dp.product_id,
    dp.product_name,
    IFNULL(order_table.total_ordered_quantity, 0) AS total_ordered_quantity,
    IFNULL(sales_table.total_sold_quantity, 0) AS total_sold_quantity,
    (IFNULL(order_table.total_ordered_quantity, 0) - IFNULL(sales_table.total_sold_quantity, 0))
    CASE
        WHEN IFNULL(order_table.total_ordered_quantity, 0) > IFNULL(sales_table.total_sold_quant:
        WHEN IFNULL(order_table.total_ordered_quantity, 0) = IFNULL(sales_table.total_sold_quant:
        ELSE 'Stock-out'
    END AS stock_status
FROM
    dim_products dp
LEFT JOIN
    (
        SELECT
            po.product_id,
            SUM(po.quantity) AS total_ordered_quantity
        FROM
            fact_purchasing_order po
        GROUP BY
            po.product_id
    ) AS order_table
```

OUTPUT

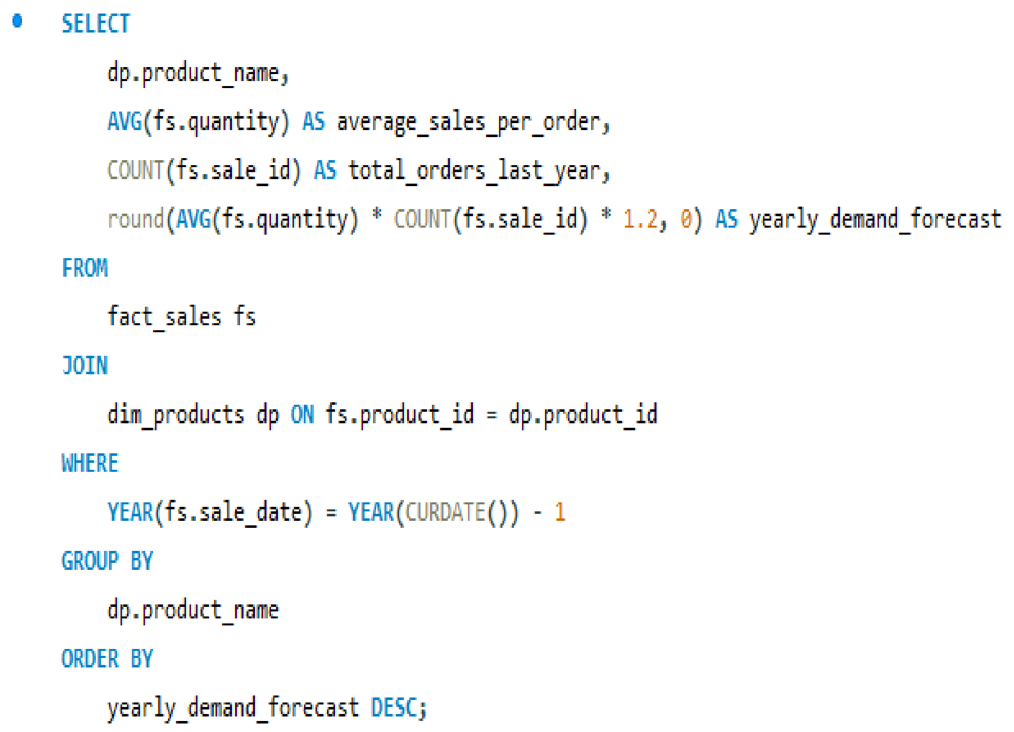


product_id	product_name	total_ordered_quantity	total_sold_quantity	stock_balance	stock_status
6007	Compact Cooling Fan	84714	12800	71914	Overstock
6012	Liquid Cooling System	46035	11600	34435	Overstock
6005	1TB NVMe SSD	42099	12100	29999	Overstock
6017	1080p Webcam	22185	7900	14285	Overstock
6038	HDMI Cable	2340	0	2340	Overstock
6032	Multi-Mode Router	2205	0	2205	Overstock
6039	Portable Battery Pack	2160	0	2160	Overstock
6037	Modular Server Rack	2115	0	2115	Overstock
6035	Programmable Mouse	2070	0	2070	Overstock
6034	Ergonomic Keyboard	2070	0	2070	Overstock
6033	Curved Gaming Monitor	2025	0	2025	Overstock
6036	RAID Storage Array	1980	0	1980	Overstock
6040	3D Printer Filament	1890	0	1890	Overstock
6031	Dual-Band Wi-Fi Ada...	1845	0	1845	Overstock
6015	Fiber Optic Cable	2295	8400	-6105	Stock-out
6023	Energy-Efficient GPU	2160	8300	-6140	Stock-out
6026	1TB HDD	2295	8700	-6405	Stock-out

*Presented by **Ngoc Anh Nguyen***

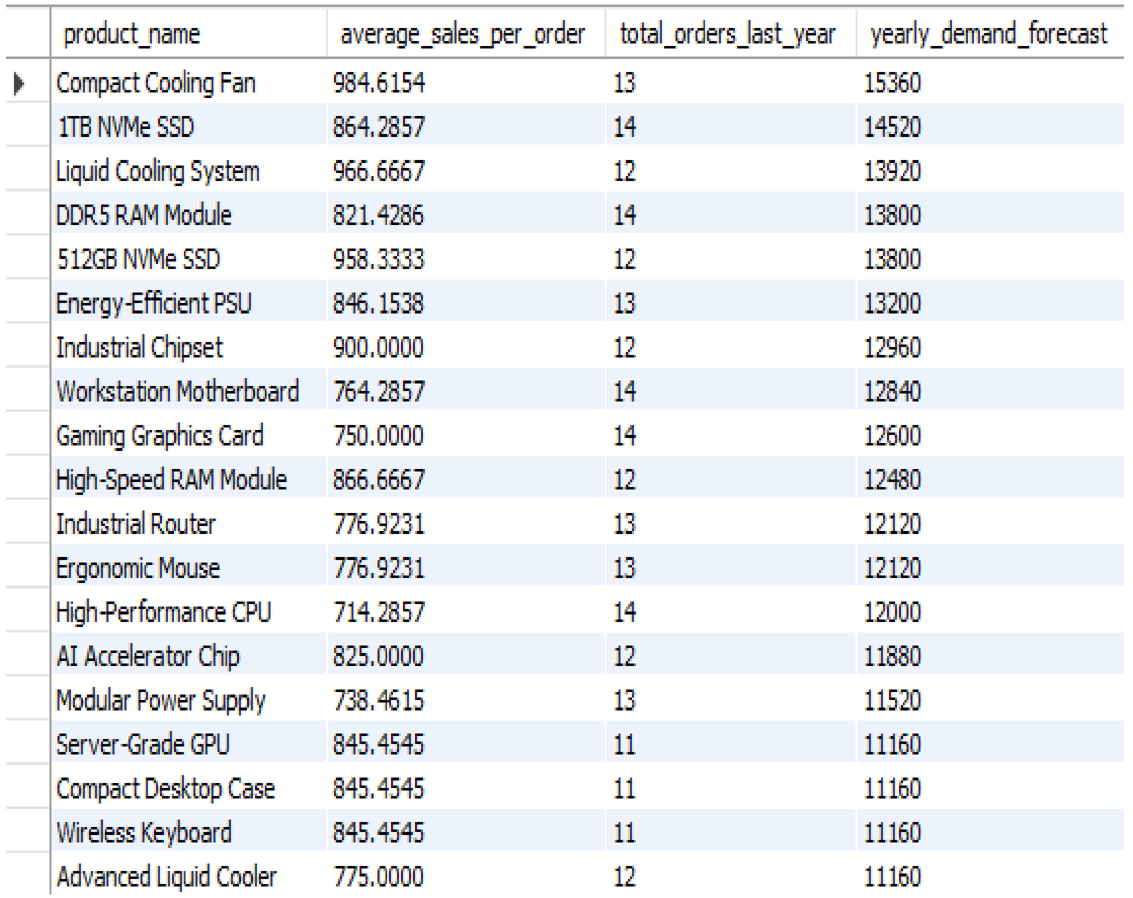
Demand Forecast

MySQL QUERY



```
SELECT
    dp.product_name,
    AVG(fs.quantity) AS average_sales_per_order,
    COUNT(fs.sale_id) AS total_orders_last_year,
    round(AVG(fs.quantity) * COUNT(fs.sale_id) * 1.2, 0) AS yearly_demand_forecast
FROM
    fact_sales fs
JOIN
    dim_products dp ON fs.product_id = dp.product_id
WHERE
    YEAR(fs.sale_date) = YEAR(CURDATE()) - 1
GROUP BY
    dp.product_name
ORDER BY
    yearly_demand_forecast DESC;
```

OUTPUT



product_name	average_sales_per_order	total_orders_last_year	yearly_demand_forecast
Compact Cooling Fan	984.6154	13	15360
1TB NVMe SSD	864.2857	14	14520
Liquid Cooling System	966.6667	12	13920
DDR5 RAM Module	821.4286	14	13800
512GB NVMe SSD	958.3333	12	13800
Energy-Efficient PSU	846.1538	13	13200
Industrial Chipset	900.0000	12	12960
Workstation Motherboard	764.2857	14	12840
Gaming Graphics Card	750.0000	14	12600
High-Speed RAM Module	866.6667	12	12480
Industrial Router	776.9231	13	12120
Ergonomic Mouse	776.9231	13	12120
High-Performance CPU	714.2857	14	12000
AI Accelerator Chip	825.0000	12	11880
Modular Power Supply	738.4615	13	11520
Server-Grade GPU	845.4545	11	11160
Compact Desktop Case	845.4545	11	11160
Wireless Keyboard	845.4545	11	11160
Advanced Liquid Cooler	775.0000	12	11160

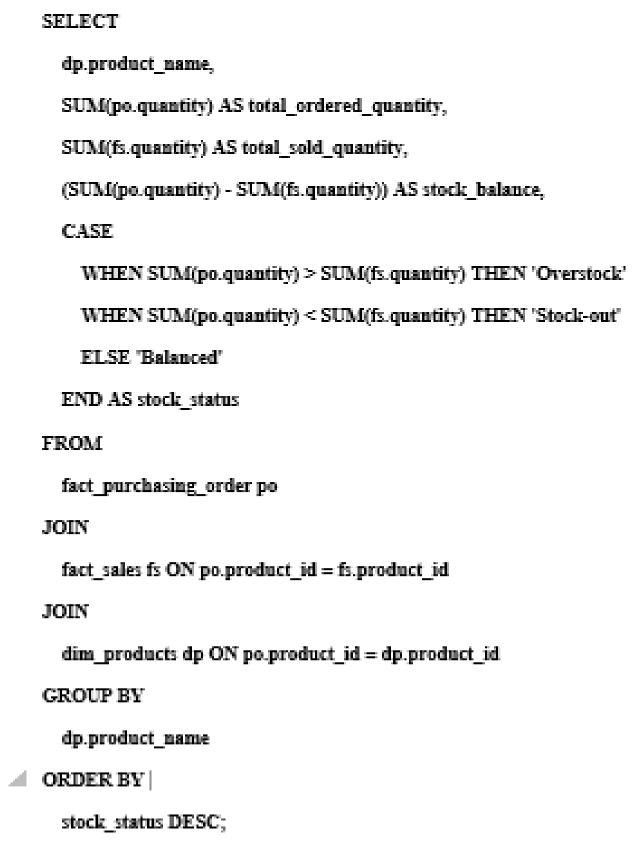
Presented by ***Ngoc Anh Nguyen***

Problems

- Altering or deleting data type of a column in a table which is also the foreign key of another table requires adjustments in that table as well.

- Import data from Excel file to MySQL: pay attention to formats (especially date format in Excel is somehow different to format in MySQL), blank spaces in each cell, etc.

- Stock-out and Overstock Analysis: cannot use join for 3 tables (right pic) but have to use subquery.



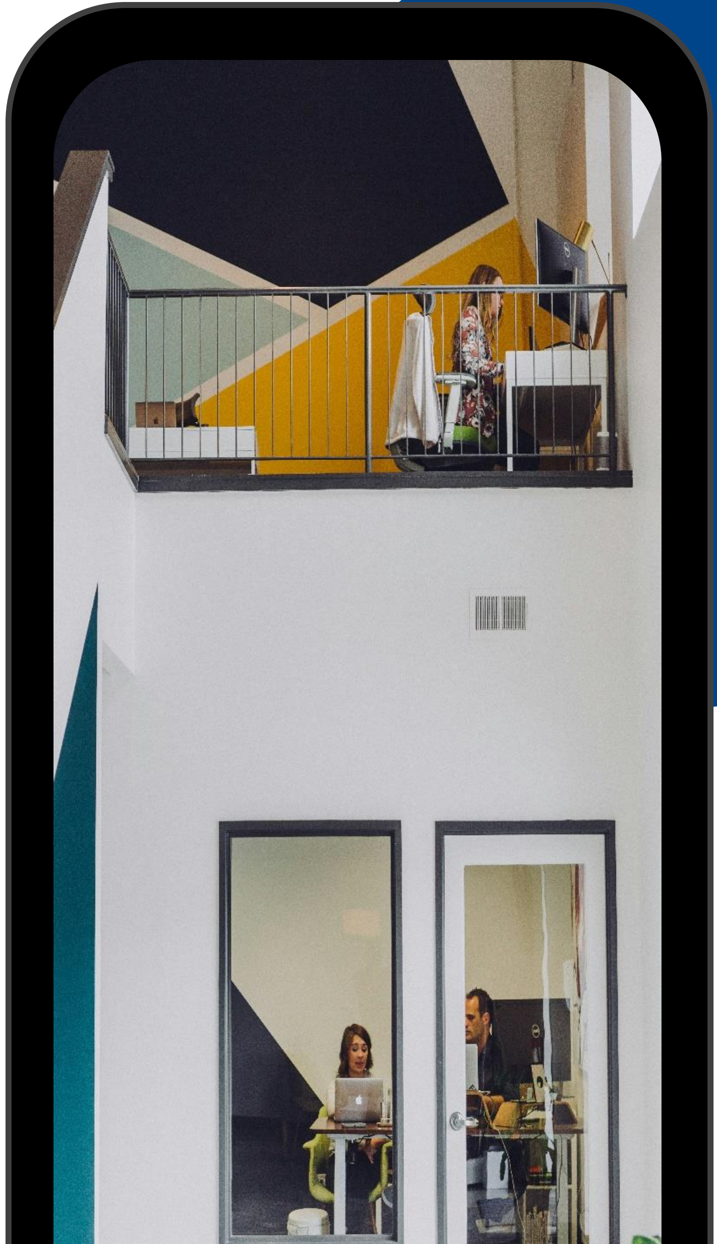
```
SELECT
  dp.product_name,
  SUM(po.quantity) AS total_ordered_quantity,
  SUM(fs.quantity) AS total_sold_quantity,
  (SUM(po.quantity) - SUM(fs.quantity)) AS stock_balance,
  CASE
    WHEN SUM(po.quantity) > SUM(fs.quantity) THEN 'Overstock'
    WHEN SUM(po.quantity) < SUM(fs.quantity) THEN 'Stock-out'
    ELSE 'Balanced'
  END AS stock_status
FROM
  fact_purchasing_order po
JOIN
  fact_sales fs ON po.product_id = fs.product_id
JOIN
  dim_products dp ON po.product_id = dp.product_id
GROUP BY
  dp.product_name
ORDER BY
  stock_status DESC;
```

*Presented by **Ngoc Anh Nguyen***

Conclusion

- In 2024, Circuit Innovations was the largest customer of Unterneh with sales revenue of 3.973 million USD and profit 2.749 million USD. Customers with long-term contract still took account of the largest proportion of Unterneh's sales revenue and profit.
- Unterneh's top-selling products in 2024 were server-grade GPU, AI Accelerator Chip, and Gaming Graphics Card. These products also belong to the GPU and Chipset categories, which ranked first and second in the list of selling categories, respectively.
- Total sales revenue was 74.33 and profit was 50.02 million USD. Quarters I and IV had higher sales revenues compared to the other two quarters, mainly because of holiday seasonality and promotion campaigns like Black Friday or Cyber Monday.

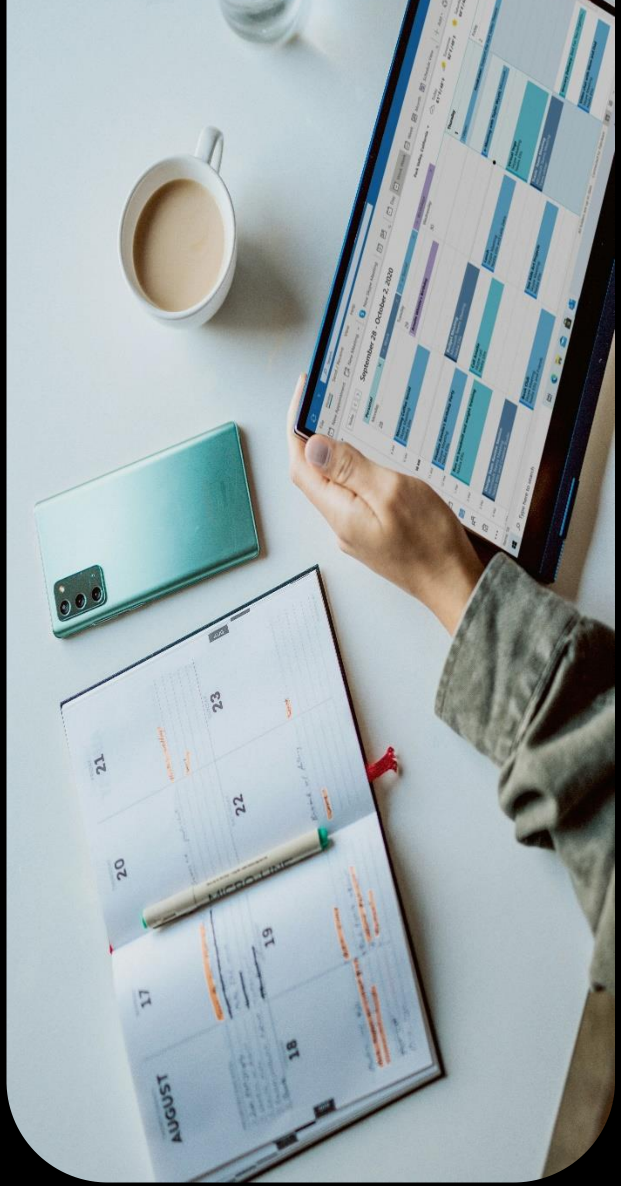
Presented by ***Ngoc Anh Nguyen***



Conclusion

- Average shipping cost through Carrier T was the highest, followed by Carrier B and Carrier A. Carrier T also has the highest number of delay days, the highest number of delayed shipments, and average delay days per shipment. Unterneh could rethink about choosing carrier T as one of the main carriers in their supply chain flow.
- Meanwhile, goods usually came from Warehouse A, C, and J through Carrier A; from Warehouse E, B, and D through Carrier B and from H, G, F to Carrier T. The company could consider the distances between warehouses and carriers to rearrange the shipping schedule or to select more appropriate carriers for each warehouse.

Presented by ***Ngoc Anh Nguyen***



Conclusion

- Unterneh usually ordered goods from suppliers like Advanced Micro Supplies, PrimeTech Components, and AquaChill Technologies. These are also suppliers with priority level 1. Besides, Unterneh's orders also had the common credit terms of Net 30 and Net 45.
- Products in categories like Cooling, Storage, GPU, and Chipset had the highest order volumes. However, Cooling and Storage items did not appear on the top-selling list. Therefore, Unterneh may want to reevaluate and adjust its purchasing strategy.
- Quarters I and II had the highest order volumes, which suggests that preparations were being made for sales throughout the entire year.

*Presented by **Ngoc Anh Nguyen***