```sql
--[1]-Calculate the total sales revenue from all orders?

SELECT ROUND(SUM(total_amount),0) AS TOTAL_SALES FROM [dbo].[orders]
```

TOTAL_SALES
44943966.00

Insight: A high-level snapshot of how the business is performing financially.

Recommendation: Compare across time (monthly/yearly) to track growth.
Monitor trends to adjust marketing or pricing strategies accordingly..

```sql
--List the top 5 best-selling products by quantity sold.

SELECT TOP 5 (p.id) as product_ID ,P.NAME, SUM( OD.quantity ) AS QUANTITY_SOLD
FROM [dbo].[products] P
JOIN [dbo].[order_details] OD ON P.id =OD.product_id
GROUP BY (p.id), P.NAME
ORDER BY QUANTITY_SOLD DESC
```

	product_ID	NAME	QUANTITY_SOLD
1	591	Vision-oriented scalable archive	253
2	499	Versatile holistic help-desk	252
3	216	Up-sized interactive time-frame	245
4	195	Vision-oriented 3rdgeneration data-warehouse	242
5	348	Team-oriented discrete hierarchy	241

Insight: These products contribute heavily to sales; possibly customer favorites or top performers.

Recommendation: -Prioritize inventory restocking and marketing efforts on these products.
-Consider bundling them with slower-moving products

```sql
--[2]- Identify customers with the highest number of orders.

SELECT (C.ID) as customer_id ,
      CONCAT( C.first_name,' ',
       C.last_name ) as full_name,
       COUNT(C.id) AS ORDER_COUNTS
FROM orders O
JOIN customers C ON O.customer_id =C.id
GROUP BY CONCAT( C.first_name,' ',
       C.last_name ) , C.id, C.first_name,C.last_name
ORDER BY ORDER_COUNTS DESC
```

customer_id	full_name	ORDER_COUNTS
212	Ryan Chang	73
90	Suzanne Bennett	68
134	Robyn Reed	68
116	Zachary Aguirre	66
189	Mark Spence	65
190	Leslie Alvarado	64
188	Tiffany Boone	64
91	Dana Farrell	64
55	John Perez	64
61	Emily Torres	63
60	Emily Walker	62

Insight: These customers are engaged and valuable.

Recommendation: -Target them for loyalty programs, up-selling, or premium services.
-Gather feedback from them to improve the customer experience.

```sql
--[3]- Generate an alert for products with stock quantities below 20 units.

SELECT id,
       name,
       stock_quantity
FROM [dbo].[products]
WHERE stock_quantity < 20
ORDER BY stock_quantity
```

id	name	stock_quantity
84	Upgradable fresh-thinking model	0
12	Advanced modular capacity	3
15	Focused radical protocol	6
3	Function-based well-modulated intranet	6
20	Reverse-engineered fresh-thinking success	6
25	Exclusive actuating open system	6
33	Distributed empowering leverage	6
100	Reactive tertiary moratorium	6
71	Distributed fault-tolerant process improvement	7
72	Cross-platform fault-tolerant secured line	7
34	Configurable optimizing extranet	7

Insight: Products that are running low and may impact sales if not restocked..

Recommendation: -Automate restocking alerts.
-Set reorder thresholds and work with suppliers proactively.

```sql
--[4]- Determine the percentage of orders that used a discount.

WITH discounted_orders AS (
    SELECT DISTINCT o.id AS order_id
    FROM order_details od
    JOIN products ps ON od.product_id = ps.id
    JOIN orders o ON od.order_id = o.id
    LEFT JOIN discounts dp
        ON dp.product_id = od.product_id
        AND dp.is_active = 1
        AND o.order_date BETWEEN dp.start_date AND dp.end_date
    LEFT JOIN discounts dc
        ON dp.id IS NULL
        AND dc.product_id IS NULL
        AND dc.is_active = 1
        AND ps.category_id = dc.category_id
        AND o.order_date BETWEEN dc.start_date AND dc.end_date
            WHERE dp.id IS NOT NULL OR dc.id IS NOT NULL

),
total_orders AS (
    SELECT COUNT(*) AS total_count FROM orders
),
discounted_order_count AS (
    SELECT COUNT(*) AS discount_count FROM discounted_orders
)

SELECT
    t.total_count AS count_total_orders,
    d.discount_count AS count_discountorders,
    CAST(d.discount_count AS FLOAT) / NULLIF(t.total_count, 0) AS percentage_orders_with_discounted_products
FROM total_orders t
CROSS JOIN discounted_order_count d;
```

count_total_orders	count_discountorders	percentage_orders_with_discounted_products
10200	858	0.0841176470588235

Insight: -High percentage might indicate over-reliance on discounts to drive sales.
-Low percentage may suggest under utilization of promotional strategies.

Recommendation: -Balance promotional frequency to protect margins..
-Analyze if discounted orders are contributing to higher average order value or customer acquisition..

```sql
--[5]- Calculate the average rating for each product.

SELECT
    p.id,
    P.name,
    ROUND(AVG(R.rating), 2) AS avg_rating
FROM Reviews R
JOIN Products P ON R.product_id = P.id
GROUP BY P.name, p.id ;
```

id	name	avg_rating
1	Enterprise-wide stable synergy	2
2	Synergized uniform contingency	1
3	Function-based well-modulated intranet	3
4	Monitored bifurcated database	2
5	Versatile modular info-mediaries	3
6	Multi-channeled foreground adapter	3
7	Fundamental grid-enabled definition	3
8	Business-focused zero tolerance functionalities	1
9	Networked bandwidth-monitored secured line	3
10	Secured reciprocal support	3
11	Visionary regional budgetary management	4
12	Advanced modular capacity	2

Insight: -Low-rated products may need improvements or better descriptions.
-High-rated products are likely reliable and meet expectations.

Recommendation: -Promote top-rated products in marketing.
-Investigate and fix issues in poorly rated products.

```sql
--[6] Compute the 30-day customer retention rate after their first purchase

with first_order as (
select customer_id , min(order_date) as first_order_date
from orders
group by customer_id ),

FollowUp_Orders AS (
    SELECT  o.customer_id,COUNT(*) AS Retained
    FROM Orders o
    JOIN first_order f ON o.customer_id = f.Customer_ID
    WHERE o.order_date > f.first_order_date AND o.order_date <= DATEADD(day, 30, f.first_order_date)
    GROUP BY o.customer_id
)SELECT
    COUNT(f.Customer_ID) AS TotalCustomers,
    COUNT(fo.customer_id) AS RetainedCustomers,
    CAST(COUNT(fo.customer_id) AS FLOAT) / COUNT(f.customer_id) * 100 AS RetentionRate_30_Days
FROM first_order f
LEFT JOIN FollowUp_Orders fo ON f.customer_id = fo.customer_id
```

TotalCustomers	RetainedCustomers	RetentionRate_30_Days
249	231	92.7710843373494

Insight: -Measures short-term customer loyalty post-purchase.

Recommendation: -If retention is low, introduce email campaigns, loyalty incentives, or follow-up offers within 30 days of the first purchase.

```sql
--[7] Recommend products frequently bought together with items in customer wishlists

WITH wishlist_products AS (
    SELECT customer_id, product_id AS wishlist_product
    FROM Wishlists
),

orders_by_wishlist_users AS (
    SELECT o.id AS order_id, o.customer_id, wp.wishlist_product
    FROM Orders o
    JOIN wishlist_products wp
        ON o.customer_id = wp.customer_id
),

products_in_orders AS (
    SELECT o.order_id, o.customer_id, o.wishlist_product, od.product_id AS purchased_product
    FROM orders_by_wishlist_users o
    JOIN order_details od ON o.order_id = od.order_id
)

SELECT
    wp.name AS wishlist_product_name,
    pp.name AS co_bought_product_name,
    COUNT(*) AS frequency
FROM products_in_orders poi
JOIN Products wp ON poi.wishlist_product = wp.id
JOIN Products pp ON poi.purchased_product = pp.id
WHERE poi.wishlist_product <> poi.purchased_product
GROUP BY wp.name, pp.name
ORDER BY frequency DESC;
```

wishlist_product_name	co_bought_product_name	frequency
Streamlined tangible ability	Multi-channeled bottom-line orchestration	9
Switchable zero-defect productivity	Organized motivating projection	8
Versatile neutral help-desk	Pre-emptive tertiary implementation	8
Streamlined tangible ability	Down-sized regional collaboration	8
Triple-buffered homogeneous tool...	Enhanced discrete implementation	8
Streamlined tangible ability	Synergized uniform protocol	8
Customizable homogeneous insta...	Up-sized multi-tasking productivity	8
Optional impactful orchestration	Devolved grid-enabled application	7
Streamlined tangible ability	Devolved value-added help-desk	7
Switchable zero-defect productivity	Exclusive 3rdgeneration emulation	7
Switchable zero-defect productivity	Front-line directional middleware	7

Insight: -Shows common purchase patterns and complements wishlist behavior.

Recommendation: -Use in cross-selling and recommendation section
Offer bundle discounts on frequently co-purchased items..

```sql
--[3] Track inventory turnover trends using a 30-day moving average.

SELECT
    od.product_id,
    o.order_date,
    (
        SELECT SUM(od2.quantity)
        FROM order_details od2
        JOIN orders  o2 ON od2.order_id = o2.id
        WHERE od2.product_id = od.product_id
          AND o2.order_date BETWEEN DATEADD(DAY, -29, o.order_date) AND o.order_date
    ) AS moving_30_day_qty_sold
FROM order_details od
JOIN orders o ON od.order_id = o.id
ORDER BY od.product_id, o.order_date;
```

product_id	order_date	moving_30_day_qty_sold
1	2025-02-11 21:44:00.000	1
1	2025-02-20 20:37:26.000	4
1	2025-03-16 21:32:52.000	4
1	2025-04-11 21:05:31.000	4
1	2025-04-15 06:54:00.000	6
1	2025-04-24 17:42:54.000	9
2	2025-01-03 14:50:19.000	5
2	2025-01-30 03:03:22.000	7
2	2025-03-05 17:04:40.000	5
2	2025-03-18 06:16:54.000	8
2	2025-03-28 13:07:53.000	9
2	2025-04-11 21:05:31.000	5
3	2025-03-13 04:21:15.000	3
3	2025-04-15 02:16:41.000	3
4	2025-01-19 04:50:05.000	5

Insight: -Products with slow turnover may tie up capital.
-Fast movers require dynamic restocking.

Recommendation: -Optimize inventory ordering based on actual movement trends.
- Reallocate marketing budgets to slow movers.

```sql
--[9] Identify customers who have purchased every product in a specific category.

WITH category_counts AS (
    SELECT category_id, COUNT(DISTINCT id) AS total_products
    FROM Products
    GROUP BY category_id
),

customer_category_purchases AS (
    SELECT o.customer_id, p.category_id, COUNT(DISTINCT od.product_id) AS purchased
    FROM Orders o
    JOIN [dbo].[order_details] od ON o.id = od.order_id
    JOIN Products p ON od.product_id = p.id
    GROUP BY o.customer_id, p.category_id
)

SELECT ccp.customer_id, ccp.category_id
FROM customer_category_purchases ccp
JOIN category_counts cc ON ccp.category_id = cc.category_id
WHERE ccp.purchased = cc.total_products;
```

customer_id	category_id

Insight: -These customers are ideal for early product testing or exclusive offers..

Recommendation: -Target them with VIP programs, sneak peeks, and personalized offers.
- Ask for reviews or feedback..

```sql
--[10] Find pairs of products commonly bought together in the same order.

SELECT
a.product_id AS product_1_id,
p1.name AS product_1_name,
b.product_id AS product_2_id,
p2.name AS product_2_name,
COUNT(*) AS pair_count
FROM order_details a
JOIN order_details b
ON a.order_id = b.order_id
AND a.product_id < b.product_id
JOIN Products p1 ON a.product_id = p1.id
JOIN Products p2 ON b.product_id = p2.id
GROUP BY a.product_id, p1.name, b.product_id, p2.name
ORDER BY pair_count DESC;
```

product_1_id	product_1_name	product_2_id	product_2_name	pair_count
282	Seamless client-driven analyzer	341	Re-engineered motivating info-mediaries	5
290	Exclusive analyzing open architecture	348	Team-oriented discrete hierarchy	4
209	Focused needs-based matrices	349	Profit-focused bandwidth-monitored toolset	4
293	Extended next generation matrices	354	Fundamental responsive project	4
132	Inverse encompassing circuit	365	Inverse coherent knowledgebase	4
189	Fully-configurable zero tolerance monitoring	290	Exclusive analyzing open architecture	4
285	Multi-tiered foreground support	299	Triple-buffered optimizing contingency	4
131	Optional impactful orchestration	311	User-centric scalable hub	4

Insight: -Supports product bundling and cross-selling strategies.

Recommendation: -Create bundles, discounts, or display these pairs together on product pages.
-Train sales staff or set up e-commerce AI rules.

```sql
--[11] Calculate the time taken to deliver orders in days.

SELECT
o.id AS order_id,
o.customer_id,
o.order_date,
s.shipping_date,
DATEDIFF(DAY, o.order_date, s.shipping_date) AS delivery_time_days
FROM Orders o
JOIN Shipping s ON o.id = s.order_id
WHERE s.status = 'delivered'
ORDER BY delivery_time_days DESC;
```

order_id	customer_id	order_date	shipping_date	delivery_time_days
8	8	2025-03-12 04:09:19.000	2025-03-13 04:09:19.000	1
9	45	2025-01-03 14:50:19.000	2025-01-04 14:50:19.000	1
17	46	2025-01-24 00:10:42.000	2025-01-25 00:10:42.000	1
33	5	2025-03-20 16:32:46.000	2025-03-21 16:32:46.000	1
34	28	2025-05-03 01:18:51.000	2025-05-04 01:18:51.000	1
36	4	2025-03-15 02:35:21.000	2025-03-16 02:35:21.000	1
38	43	2025-01-02 21:19:27.000	2025-01-03 21:19:27.000	1
39	45	2025-03-26 07:52:44.000	2025-03-27 07:52:44.000	1

Insight: -Long delivery times may hurt repeat purchases.

Recommendation: -Work with logistics partners to reduce delays.
-Display estimated delivery times on the site transparently.

```sql
-- sales by delivered
SELECT status , ROUND(SUM(total_amount),0) AS TOTAL_SALES FROM [dbo].[orders]
group by status
having status = 'delivered';
```

status	TOTAL_SALES
delivered	8873678.00

```sql
-- by total status
SELECT status , ROUND(SUM(total_amount),0) AS TOTAL_SALES FROM [dbo].[orders]
group by status
order by status desc;
```

status	TOTAL_SALES
shipped	8918413.00
processing	8976127.00
pending	9308026.00
delivered	8873678.00
cancelled	8867723.00

Insight: -Revenue is only realized upon delivery, not just order placement.

Recommendation: -Investigate undelivered or delayed orders affecting cash flow..