select \* from dbo.categories;

select \* from dbo.customers;

select \* from dbo.payments;

select \* from dbo.inventory;

select \* from dbo.order\_items;

select \* from dbo.orders;

select \* from dbo.product\_reviews;

select \* from dbo.products;

select \* from dbo.shippings;

select \* from dbo.suppliers;

-------------------Customer an-------------------

----List the top 10 customers by total spend.

select

top 10

c.customer\_id, sum(try\_cast(p.amount as decimal (18,2))) total\_amount , count(\*) as total\_payments

from dbo.customers c

join dbo.payments p

on c.customer\_id = p.customer\_id

group by c.customer\_id

order by total\_amount desc

---------top 10 customer by rank for more accurate result...........

with ranked\_customers AS(

select c.customer\_id , sum(try\_cast(p.amount as decimal (18,2))) total\_amount , count(\*) as payment\_amount,

RANK()over(order by sum(try\_cast(p.amount as decimal (18,2))) desc)AS rnk\_no

from dbo.customers c

join dbo.payments p

on c.customer\_id = p.customer\_id

group by c.customer\_id

)

select \*

from

ranked\_customers

where rnk\_no < = 10

ORDER BY rnk\_no, total\_amount DESC;

------Identify customers who haven't placed any orders in the last 6 months.

select

c.customer\_id, c.city

from dbo.customers c

left join

dbo.orders o

on c.customer\_id = o.customer\_id

and try\_cast(order\_date as date) > = dateadd(month, -6, getdate())

where o.customer\_id IS NULL;

-------Calculate the average order value per customer.

select

o.order\_id,

sum(try\_cast(oi.quantity as decimal(18,2))) \*sum(try\_cast(oi.unit\_price as decimal(18,2)))/count(try\_cast(o.order\_id as decimal(18,2))) as average\_order\_value

from dbo.orders o

join dbo.order\_items oi

on o.order\_id = oi.order\_id

group by o.order\_id

order by average\_order\_value desc

-------Find customers who only gave 5-star product reviews.

select product\_id, customer\_id

from dbo.product\_reviews

where

rating = 5

------Identify customers who ordered more than 5 times but never reviewed a product.

select \* from dbo.orders;

select \* from dbo.payments;

select \* from dbo.product\_reviews;

select

c.customer\_id, c.first\_name , c.last\_name ,o.order\_id, count(distinct O.order\_id) as total\_orders

from dbo.customers c

join

dbo.orders o

on o.customer\_id = c.customer\_id

left join dbo.product\_reviews pr

on c.customer\_id = pr.customer\_id

group by c.customer\_id, c.first\_name , c.last\_name ,o.order\_id

having count(distinct o.order\_id) >5 and count(pr.rating) =0

-----Rank customers based on number of products bought (dense\_rank).

select c.customer\_id, c.first\_name, c.last\_name ,

dense\_rank()over(order by sum(try\_cast(oi.quantity as decimal (18,2))) desc) as rnk

from dbo.customers c

join dbo.orders o

on c.customer\_id =o.customer\_id

join dbo.order\_items oi

on oi.order\_id = o.order\_id

GROUP BY

c.customer\_id,

c.first\_name,

c.last\_name

ORDER BY

rnk;

-------Perform RFM (Recency, Frequency, Monetary) analysis for each customer.

select c.customer\_id, datediff(day, max(try\_cast(o.order\_date as date), getdate()) as recency\_days,

count(try\_cast(o.order\_id as decimal (18,2))) as frquency,

sum(try\_cast(o.order\_id as decimal (18,2))) as monetary

from dbo.customers c

join dbo.orders o

on c.customer\_id = o.customer\_id

group by

c.customer\_id;

-------Find the top 5 best-selling products by quantity sold.

with ranked\_products as(

select order\_id, product\_id,

sum(try\_cast(quantity as decimal (18,2))) as total\_quantity\_sold,

RANK()over(order by sum(try\_cast(quantity as decimal (18,2)))desc) as rnk

from dbo.order\_items

group by product\_id,order\_id

)

select order\_id, product\_id

from ranked\_products

where rnk <=5

--------------------by group by -----------another way-----

select top 5

product\_id , order\_id , sum(try\_cast(quantity as decimal(18,2))) as total\_quantity\_sold

from dbo.order\_items

group by

product\_id , order\_id

order by

total\_quantity\_sold desc

----------Identify products with inventory less than 10 units and high sales in the last month.

select

i.product\_id, i.stock\_quantity,

sum(try\_cast(oi.quantity as decimal (18,2))) as total\_quantity\_sold

from dbo.inventory i

join dbo.order\_items oi

on i.product\_id = oi.product\_id

join

dbo.orders o

on oi.order\_id = o.order\_id

where i.stock\_quantity<10

AND o.order\_date >= DATEADD(MONTH, DATEDIFF(MONTH, 0, GETDATE()) - 1, 0) -- first day of last month

AND o.order\_date < DATEADD(MONTH, DATEDIFF(MONTH, 0, GETDATE()), 0) -- first day of this month

group by i.product\_id , i.stock\_quantity

having sum(try\_cast(oi.quantity as decimal(18,2))) >50-----high sales condition

------Get average rating per product and filter for those with >4 stars.

select product\_id , avg(try\_cast(rating as decimal (18,2))) as avg\_rating

from dbo.product\_reviews

group by product\_id

having avg(try\_cast(rating as decimal (18,2))) >4

-------Identify products that have never been ordered.

select i.product\_id

FROM dbo.inventory i

where i.product\_id NOT IN (

select distinct product\_id

from dbo.order\_items

where

product\_id IS NOT NULL

);

-----------List all products that have been out of stock recently (e.g., stock\_quantity = 0).

SELECT product\_id, stock\_quantity

FROM dbo.inventory

WHERE stock\_quantity = 0;

----------Join with categories to find the best-selling category.

select p.category\_id ,

sum(try\_cast(oi.quantity as decimal (18,2))\*try\_cast(oi.unit\_price as decimal(18,2)))as best\_selling\_product

from dbo.products p

join dbo.order\_items oi

on oi.product\_id = p.product\_id

group by p.category\_id

order by best\_selling\_product desc

---------Get products that are priced above the average in their category.

select

p.product\_id,

p.product\_name,

p.category\_id,

oi.unit\_price

from dbo.products p

join dbo.order\_items oi

on p.product\_id = oi.product\_id

where oi.unit\_price > (

select avg(try\_cast(oi2.unit\_price as decimal(18,2)))

from dbo.order\_items oi2

join dbo.products p2

on oi2.product\_id = p2.product\_id

where p2.category\_id = p.category\_id

);

------Calculate monthly total sales.

select

year(try\_cast (o.order\_date as date )) as sales\_year,

month(try\_cast (o.order\_date as date)) as sales\_month,

sum(try\_cast( oi.quantity as decimal(18,2))\*try\_cast(oi.unit\_price as decimal(18,2))) as total\_sales

from dbo.orders o

join dbo.order\_items oi

on o.order\_id = oi.order\_id

group by

year(try\_cast (o.order\_date as date )),

month(try\_cast (o.order\_date as date))

order by

sales\_year, sales\_month

-------Find the average number of products per order.

select

avg(product\_count) as avg\_product\_per\_order

from

(

select o.order\_id ,

count(try\_cast(oi.product\_id as decimal (18,2))) as product\_count

from dbo.orders o

join order\_items oi

on

o.order\_id = oi.order\_id

group by o.order\_id

)

as

sub;

----Detect the percentage of orders that were cancelled.

select

(count( case when status = 'Cancelled' THEN 1 END )\*100/count(\*))

AS cancelled\_percentage

from dbo.orders

-----Compare order volume between different shipping methods.

select

shipping\_id , shipping\_method,

count(\*) AS total\_orders

from dbo.shippings

group by

shipping\_id , shipping\_method

order by total\_orders desc;

-----Find orders with unusually high total value (e.g., > ₹25,000).

SELECT

o.order\_id,

o.customer\_id,

o.order\_date,

SUM(TRY\_CAST(oi.quantity AS DECIMAL(18,2)) \* TRY\_CAST(oi.unit\_price AS DECIMAL(18,2))) AS total\_order\_value

FROM dbo.orders o

JOIN dbo.order\_items oi

ON o.order\_id = oi.order\_id

GROUP BY o.order\_id, o.customer\_id, o.order\_date

HAVING SUM(TRY\_CAST(oi.quantity AS DECIMAL(18,2)) \* TRY\_CAST(oi.unit\_price AS DECIMAL(18,2))) > 25000

ORDER BY total\_order\_value DESC;

-----Detect duplicate orders placed by same customer within 1 hour.

----

SELECT

customer\_id,

TRY\_CONVERT(DATE, order\_date, 105) AS order\_date,

COUNT(\*) AS duplicate\_orders

FROM dbo.orders

GROUP BY customer\_id, TRY\_CONVERT(DATE, order\_date, 105)

HAVING COUNT(\*) > 1

ORDER BY duplicate\_orders DESC;

------Calculate average delivery time per shipping method.

select

s.shipping\_method ,

avg(datediff(day, try\_convert(date, o.order\_date , 105), try\_convert(date , s.delivery\_date, 105))) as avg\_delivery\_days

FROM dbo.orders o

JOIN dbo.shippings s

on o.shipping\_id = s.shipping\_id

group by

s.shipping\_method

order by avg\_delivery\_days;

-------Find the shipping method with the fastest average delivery time.

select top 1

s.shipping\_method,

avg(datediff(day, try\_convert(date, o.order\_date, 105), try\_convert(date, s.delivery\_date, 105))) as avg\_delivery\_days

from dbo.orders o

join dbo.shippings s

on o.shipping\_id = s.shipping\_id

group by s.shipping\_method

order by avg\_delivery\_days asc

-----Identify delayed deliveries (where delivery\_date > shipping\_date + 7).

select o.order\_id , o.customer\_id ,

try\_convert(date, o.order\_date , 105) AS order\_date ,

try\_convert(date , s.delivery\_date , 105) as delivery\_date,

datediff(day, try\_convert(date, o.order\_date , 105) , try\_convert(date , s.delivery\_date , 105)) as delivery\_days

from dbo.orders o

join

dbo.shippings s

on o.shipping\_id = s.shipping\_id

where datediff(day, try\_convert(date, o.order\_date , 105) , try\_convert(date , s.delivery\_date , 105)) > 7

order by delivery\_days;

------Count number of deliveries per city.

select c.city, count(\*) as total\_deliveries

from dbo.customers c

join dbo.orders o

on o.customer\_id = c.customer\_id

join

dbo.shippings s

on o.shipping\_id = s.shipping\_id

group by c.city

order by

total\_deliveries;

-------Average shipping cost per category (via joining with products → orders).

SELECT

c.category\_name,

AVG(try\_cast(s.shipping\_cost as decimal(18,2))) AS avg\_shipping\_cost

FROM orders o

JOIN order\_items oi

ON o.order\_id = oi.order\_id

JOIN products p

ON oi.product\_id = p.product\_id

JOIN categories c

ON p.category\_id = c.category\_id

JOIN shippings s

ON o.shipping\_id = s.shipping\_id

GROUP BY c.category\_name

ORDER BY avg\_shipping\_cost DESC;

-------Count failed or zero-value payments (if applicable).

SELECT

COUNT(\*) AS failed\_or\_zero\_payments

FROM payments

WHERE payment\_status = 'Failed'

OR amount = 0;

--Total revenue collected through UPI vs Credit Card.

select

payment\_id, payment\_method,sum(try\_cast(amount as decimal (18,2))) as Total\_revenue\_collected

from dbo.payments

WHERE payment\_method IN ('UPI', 'Credit Card')

group by payment\_id , payment\_method

order by Total\_revenue\_collected desc

--Average payment amount per method.

select payment\_method, avg(try\_cast(amount as decimal (18,2))) as avg\_payment\_amount

from dbo.payments

group by payment\_method

order by avg\_payment\_amount desc

--Detect customers who paid multiple times for the same order ID.

SELECT

o.customer\_id,

o.order\_id,

COUNT(p.payment\_id) AS payment\_count,

SUM(TRY\_CAST(p.amount AS DECIMAL(18,2))) AS total\_amount\_paid

FROM dbo.orders o

JOIN dbo.payments p

ON o.customer\_id = p.customer\_id -- linking via customer

GROUP BY o.customer\_id, o.order\_id

HAVING COUNT(p.payment\_id) > 1

ORDER BY payment\_count DESC, total\_amount\_paid DESC;

--Find products with the most reviews.

SELECT

product\_id,

COUNT(review\_id) AS review\_count

FROM dbo.product\_reviews

GROUP BY product\_id

ORDER BY review\_count DESC;

-------Count number of reviews per rating (1 to 5).

select \* from dbo.product\_reviews;

SELECT

rating,

COUNT(\*) AS total\_reviews

FROM product\_reviews

GROUP BY rating

ORDER BY rating;

----Detect reviews where comment is empty or null.

SELECT review\_id,

product\_id,

customer\_id,

rating,

comment

FROM product\_reviews

WHERE comment IS NULL

OR TRIM(comment) = '';

--Calculate average rating per supplier.

select avg(try\_cast( r.rating as decimal (18,2))) as avg\_rating, s.supplier\_id

from dbo.suppliers s

join dbo.products p

on p.supplier\_id = s.supplier\_id

join dbo.product\_reviews r

on p.product\_id = r.product\_id

group by s.supplier\_id

order by avg\_rating desc;

--Get number of reviews each customer wrote per month.

SELECT

c.customer\_id,

CONCAT(c.first\_name, ' ', c.last\_name) AS customer\_name,

DATEFROMPARTS(

YEAR(CONVERT(DATE, pr.review\_date, 103)), -- 103 = dd/mm/yyyy

MONTH(CONVERT(DATE, pr.review\_date, 103)),

1

) AS review\_month,

COUNT(pr.review\_id) AS total\_reviews

FROM customers c

JOIN product\_reviews pr

ON c.customer\_id = pr.customer\_id

GROUP BY

c.customer\_id, c.first\_name, c.last\_name,

DATEFROMPARTS(

YEAR(CONVERT(DATE, pr.review\_date, 103)),

MONTH(CONVERT(DATE, pr.review\_date, 103)),

1

)

ORDER BY

review\_month,

total\_reviews DESC;

--🔹 Advanced SQL Concepts

---Use ROW\_NUMBER() to find the first order placed by each customer.

with ranked\_orders as (

select o.order\_id,

o.customer\_id,

o.order\_date,

o.status,

row\_number()over ( PARTITION BY o.customer\_id

ORDER BY o.order\_date ASC)

AS rnk

from dbo.orders o

)

select

ro.order\_id ,

ro.customer\_id ,

ro.order\_date,

status

from

ranked\_orders ro

JOIN customers c

ON ro.customer\_id = c.customer\_id

WHERE rnk = 1

ORDER BY ro.customer\_id;

--Use a CTE to get top 3 products in each category by sales.

WITH ProductSales AS (

-- 1st CTE: calculate total sales per product

SELECT

p.product\_id,

p.product\_name,

p.category\_id,

SUM(TRY\_CAST(oi.quantity AS DECIMAL(18,2)) \* TRY\_CAST(oi.unit\_price AS DECIMAL(18,2))) AS total\_sales

FROM products p

JOIN order\_items oi ON p.product\_id = oi.product\_id

GROUP BY p.product\_id, p.product\_name, p.category\_id

),

RankedProducts AS (

-- 2nd CTE: rank products within each category

SELECT

ps.\*,

ROW\_NUMBER() OVER (PARTITION BY ps.category\_id ORDER BY ps.total\_sales DESC) AS rn

FROM ProductSales ps

)

-- Main query: get only top 3 products per category

SELECT \*

FROM RankedProducts

WHERE rn <= 3

ORDER BY category\_id, total\_sales DESC;

--Use a window function to calculate rolling 7-day order count.

SELECT

order\_date,

COUNT(order\_id) OVER (

ORDER BY order\_date

ROWS BETWEEN 6 PRECEDING AND CURRENT ROW

) AS rolling\_7day\_order\_count

FROM orders

ORDER BY order\_date;

----Use a correlated subquery to get products costlier than the average of their category.

SELECT product\_id,

product\_name,

price,

category\_id

FROM products p

WHERE price > (

SELECT AVG(try\_cast(price as decimal(18,2))) as price

FROM products

WHERE category\_id = p.category\_id

);

---Use a pivot (CASE WHEN) to show count of orders by status per month.

SELECT

CONVERT(VARCHAR(7), CONVERT(DATE, order\_date, 105), 120) AS order\_month,

COUNT(CASE WHEN status = 'Processing' THEN 1 END) AS Processing,

COUNT(CASE WHEN status = 'Shipped' THEN 1 END) AS Shipped,

COUNT(CASE WHEN status = 'Delivered' THEN 1 END) AS Delivered,

COUNT(CASE WHEN status = 'Cancelled' THEN 1 END) AS Cancelled

FROM orders

GROUP BY CONVERT(VARCHAR(7), CONVERT(DATE, order\_date, 105), 120)

ORDER BY order\_month;

----Which supplier generates the most revenue?

select \* from dbo.suppliers;

select \* from dbo.order\_items

select \* from dbo.products;

select top 1

s.supplier\_id ,

sum(try\_cast(oi.quantity as decimal (18,2))\*try\_cast(oi.unit\_price as decimal (18,2))) as total\_revenue

from dbo.order\_items oi

join dbo.products p

on p.product\_id = oi.product\_id

join dbo.suppliers s

on s.supplier\_id = p.supplier\_id

group by s.supplier\_id

order by total\_revenue desc ;

----What is the return rate (cancelled orders vs total)?

SELECT

COUNT(\*) AS total\_orders,

SUM(CASE WHEN status = 'Cancelled' THEN 1 ELSE 0 END) AS cancelled\_orders,

CAST(SUM(CASE WHEN status = 'Cancelled' THEN 1 ELSE 0 END) AS FLOAT)

/ COUNT(\*) \* 100 AS return\_rate\_percentage

FROM orders;

----Which city generates the most revenue?

select \* from dbo.suppliers;

select \* from dbo.order\_items

select \* from dbo.products;

select top 1

s.city ,

sum(try\_cast(oi.quantity as decimal (18,2))\*try\_cast(oi.unit\_price as decimal (18,2))) as total\_revenue

from dbo.order\_items oi

join dbo.products p

on p.product\_id = oi.product\_id

join dbo.suppliers s

on s.supplier\_id = p.supplier\_id

group by s.city

order by total\_revenue desc ;

-----Most popular day of week to place orders?

-- Count orders by day of the week

SELECT

DATENAME(WEEKDAY, CONVERT(DATETIME, order\_date, 105)) AS DayOfWeek,

COUNT(\*) AS TotalOrders

FROM orders

GROUP BY DATENAME(WEEKDAY, CONVERT(DATETIME, order\_date, 105))

ORDER BY TotalOrders DESC;

----Average basket size (number of items per order).

SELECT

AVG(try\_cast(total\_items as decimal (18,2))) AS average\_basket\_size

FROM (

SELECT

order\_id,

SUM(try\_cast(quantity as decimal(18,2))) AS total\_items

FROM order\_items

GROUP BY order\_id

) AS order\_totals;

-----Revenue growth month-over-month.

WITH MonthlyRevenue AS (

SELECT

YEAR(TRY\_CONVERT(DATE, o.order\_date, 105)) AS OrderYear,

MONTH(TRY\_CONVERT(DATE, o.order\_date, 105)) AS OrderMonth,

SUM(try\_cast(oi.quantity as decimal(18,2)) \* try\_cast(oi.unit\_price as decimal (18,2))) AS Revenue

FROM dbo.orders o

JOIN dbo.order\_items oi ON o.order\_id = oi.order\_id

GROUP BY YEAR(TRY\_CONVERT(DATE, o.order\_date, 105)),

MONTH(TRY\_CONVERT(DATE, o.order\_date, 105))

)

SELECT

OrderYear,

OrderMonth,

Revenue,

LAG(Revenue) OVER (ORDER BY OrderYear, OrderMonth) AS PreviousMonthRevenue,

CASE

WHEN LAG(Revenue) OVER (ORDER BY OrderYear, OrderMonth) = 0 THEN NULL

ELSE ROUND(

(Revenue - LAG(Revenue) OVER (ORDER BY OrderYear, OrderMonth)) \* 100.0

/ LAG(Revenue) OVER (ORDER BY OrderYear, OrderMonth), 2

)

END AS MoM\_Growth\_Percent

FROM MonthlyRevenue

ORDER BY OrderYear, OrderMonth;

-----Customers with increasing order frequency over the last 3 months.

WITH MonthlyOrders AS (

SELECT

customer\_id,

DATEFROMPARTS(

YEAR(TRY\_CONVERT(DATE, order\_date, 105)),

MONTH(TRY\_CONVERT(DATE, order\_date, 105)),

1

) AS order\_month,

COUNT(order\_id) AS order\_count

FROM orders

WHERE TRY\_CONVERT(DATE, order\_date, 105) IS NOT NULL

GROUP BY customer\_id,

DATEFROMPARTS(

YEAR(TRY\_CONVERT(DATE, order\_date, 105)),

MONTH(TRY\_CONVERT(DATE, order\_date, 105)),

1

)

),

Last3Months AS (

SELECT

mo.customer\_id,

mo.order\_month,

mo.order\_count,

ROW\_NUMBER() OVER (PARTITION BY mo.customer\_id ORDER BY mo.order\_month DESC) AS rn

FROM MonthlyOrders mo

WHERE mo.order\_month >= DATEADD(MONTH, -3, GETDATE())

),

Pivoted AS (

SELECT

customer\_id,

MAX(CASE WHEN rn = 3 THEN order\_count END) AS month1\_orders,

MAX(CASE WHEN rn = 2 THEN order\_count END) AS month2\_orders,

MAX(CASE WHEN rn = 1 THEN order\_count END) AS month3\_orders

FROM Last3Months

GROUP BY customer\_id

)

SELECT

customer\_id,

month1\_orders,

month2\_orders,

month3\_orders

FROM Pivoted

WHERE month1\_orders IS NOT NULL

AND month2\_orders IS NOT NULL

AND month3\_orders IS NOT NULL

AND month1\_orders < month2\_orders

AND month2\_orders < month3\_orders

ORDER BY customer\_id;

----Find top 10 profitable product (total revenue minus shipping cost per order).

select top 10

oi.product\_id , sum(try\_cast(oi.quantity as decimal(18,2)) \* try\_cast(oi.unit\_price as decimal (18,2)) - try\_cast(s.shipping\_cost as decimal (18,2))) as total\_profit

from dbo.order\_items oi

join dbo.orders o

on o.order\_id = oi.order\_id

Join dbo.shippings s

on s.shipping\_id = o.shipping\_id

group by oi.product\_id

order by total\_profit desc;

----Customer Segmentation & Behavior

-----Segment customers based on total amount spent (High/Medium/Low).

SELECT

c.customer\_id,

c.first\_name,

c.last\_name,

SUM(try\_cast(oi.quantity as decimal(18,2)) \* try\_cast(oi.unit\_price as decimal(18,2))) AS total\_spent,

CASE

WHEN SUM(try\_cast(oi.quantity as decimal(18,2)) \* try\_cast(oi.unit\_price as decimal(18,2))) >= 50000 THEN 'High'

WHEN SUM(try\_cast(oi.quantity as decimal(18,2)) \* try\_cast(oi.unit\_price as decimal(18,2))) >= 20000 THEN 'Medium'

ELSE 'Low'

END AS spending\_segment

FROM customers c

JOIN orders o

ON c.customer\_id = o.customer\_id

JOIN order\_items oi

ON o.order\_id = oi.order\_id

GROUP BY c.customer\_id, c.first\_name, c.last\_name

ORDER BY total\_spent DESC;

----Identify repeat customers who placed more than 2 orders every month.

select \* from dbo.orders

SELECT

customer\_id,

MONTH(TRY\_CONVERT(date, order\_date, 105)) AS order\_month,

YEAR(TRY\_CONVERT(date, order\_date, 105)) AS order\_year,

COUNT(order\_id) AS total\_orders

FROM dbo.orders

GROUP BY

customer\_id,

MONTH(TRY\_CONVERT(date, order\_date, 105)),

YEAR(TRY\_CONVERT(date, order\_date, 105))

HAVING COUNT(order\_id) > 2

ORDER BY

customer\_id,

order\_year,

order\_month;

-----Find customers who ordered from at least 3 different categories.

SELECT

c.customer\_id,

c.first\_name,

c.last\_name,

COUNT(DISTINCT p.category\_id) AS category\_count

FROM customers c

JOIN orders o

ON c.customer\_id = o.customer\_id

JOIN order\_items oi

ON o.order\_id = oi.order\_id

JOIN products p

ON oi.product\_id = p.product\_id

JOIN categories cat

ON p.category\_id = cat.category\_id

GROUP BY c.customer\_id, c.first\_name, c.last\_name

HAVING COUNT(DISTINCT p.category\_id) >= 3

ORDER BY category\_count DESC;

---Identify customers with more than one shipping city in order history.

SELECT

o.customer\_id,

COUNT(DISTINCT s.city) AS unique\_shipping\_cities

FROM dbo.orders o

join dbo.order\_items oi

on o.order\_id = oi.order\_id

JOIN dbo.products p

ON oi.product\_id = p.product\_id

JOIN dbo.suppliers s

on s.supplier\_id = p.supplier\_id

GROUP BY o.customer\_id

HAVING COUNT(DISTINCT s.city) > 1

ORDER BY unique\_shipping\_cities DESC;

----Find top customers who ordered using multiple payment methods.

--

SELECT top 10

c.customer\_id,

c.first\_name,

c.last\_name,

COUNT(DISTINCT p.payment\_method) AS distinct\_payment\_methods

FROM dbo.customers c

JOIN dbo.orders o

ON c.customer\_id = o.customer\_id

JOIN dbo.payments p

ON o.payment\_id = p.payment\_id

GROUP BY c.customer\_id, c.first\_name, c.last\_name

HAVING COUNT(DISTINCT p.payment\_method) > 1

----Track customer acquisition trends by month (from created\_at).

select \* from dbo.customers;

SELECT

FORMAT(TRY\_CONVERT(DATE, created\_at), 'yyyy-MM') AS Month\_Year,

COUNT(customer\_id) AS New\_Customers

FROM customers

WHERE TRY\_CONVERT(DATE, created\_at) IS NOT NULL -- Ignore bad dates

GROUP BY FORMAT(TRY\_CONVERT(DATE, created\_at), 'yyyy-MM')

ORDER BY Month\_Year;

----Get average number of days between orders per customer.

WITH OrderGaps AS (

SELECT

customer\_id,

order\_id,

TRY\_CONVERT(DATE, order\_date, 103) AS OrderDate, -- 103 = dd/MM/yyyy format

LAG(TRY\_CONVERT(DATE, order\_date, 103)) OVER (

PARTITION BY customer\_id

ORDER BY TRY\_CONVERT(DATE, order\_date, 103)

) AS Prev\_Order\_Date

FROM orders

WHERE TRY\_CONVERT(DATE, order\_date, 103) IS NOT NULL -- Skip bad date rows

)

SELECT

customer\_id,

AVG(DATEDIFF(DAY, Prev\_Order\_Date, OrderDate)) AS Avg\_Days\_Between\_Orders

FROM OrderGaps

WHERE Prev\_Order\_Date IS NOT NULL

GROUP BY customer\_id

ORDER BY Avg\_Days\_Between\_Orders;

---Identify customers who placed their first order on a weekend.

SELECT

o.customer\_id,

MIN(o.order\_date) AS first\_order\_date,

DATENAME(WEEKDAY, TRY\_CONVERT(DATE, MIN(o.order\_date))) AS first\_order\_day

FROM orders o

GROUP BY o.customer\_id

HAVING DATENAME(WEEKDAY, TRY\_CONVERT(DATE, MIN(o.order\_date))) IN ('Saturday', 'Sunday');

----Detect churned customers (no orders in last 90 days).

SELECT

c.customer\_id,

c.first\_name,

c.last\_name

FROM customers c

LEFT JOIN orders o

ON c.customer\_id = o.customer\_id

WHERE

TRY\_CONVERT(DATE, o.order\_date, 103) IS NULL

OR TRY\_CONVERT(DATE, o.order\_date, 103) < DATEADD(DAY, -90, CAST(GETDATE() AS DATE));

----Which customers have never used COD?

SELECT DISTINCT c.customer\_id

FROM customers c

LEFT JOIN orders o

ON c.customer\_id = o.customer\_id

LEFT JOIN payments p

ON o.payment\_id = p.payment\_id

AND p.payment\_method = 'COD'

WHERE p.payment\_id IS NULL;

----Find frequently bought together product pairs (using self-join on order\_items).

SELECT

oi1.product\_id AS product\_1,

oi2.product\_id AS product\_2,

COUNT(\*) AS times\_bought\_together

FROM order\_items oi1

JOIN order\_items oi2

ON oi1.order\_id = oi2.order\_id

AND oi1.product\_id < oi2.product\_id

GROUP BY oi1.product\_id, oi2.product\_id

ORDER BY times\_bought\_together DESC;