Create Database and Load Data

Create Database

```
CREATE DATABASE RealEstateDB ;
GO
USE RealEstateDB ;
GO
```

Create Tables

```
CREATE TABLE Property (
  PropertyID INT PRIMARY KEY,
  PropertyType VARCHAR(50),
  Location VARCHAR(255),
  Size_sqm INT,
  PriceUSD INT
);
CREATE TABLE Clients (
  ClientID INT PRIMARY KEY,
  FirstName VARCHAR(100),
  LastName VARCHAR(100),
  Phone VARCHAR(50),
  Email VARCHAR(100)
);
CREATE TABLE Agents (
  AgentID INT PRIMARY KEY,
  FirstName VARCHAR(100),
  LastName VARCHAR(100),
  Phone VARCHAR(50),
  Email VARCHAR(100)
);
```

```
CREATE TABLE Sales (
  SaleID INT PRIMARY KEY,
  PropertyID INT,
  ClientID INT,
  AgentID INT,
  SaleDate DATE,
  SalePrice INT,
  FOREIGN KEY (PropertyID) REFERENCES Property(PropertyID),
  FOREIGN KEY (ClientID) REFERENCES Clients(ClientID),
  FOREIGN KEY (AgentID) REFERENCES Agents(AgentID)
);
CREATE TABLE Visits (
  VisitID INT PRIMARY KEY,
  PropertyID INT,
  ClientID INT,
  AgentID INT,
  VisitDate DATE,
  FOREIGN KEY (PropertyID) REFERENCES property(PropertyID),
  FOREIGN KEY (ClientID) REFERENCES Clients(ClientID),
  FOREIGN KEY (AgentID) REFERENCES Agents(AgentID)
);
```

Load Data

```
BULK INSERT Property
FROM 'D:\Data Analysis\My Projects\Data\Realstate\property.csv'
WITH (FIRSTROW = 2, FIELDTERMINATOR = ';', ROWTERMINATOR = '\n' );
Select*
From Property;

BULK INSERT Clients
FROM 'D:\Data Analysis\My Projects\Data\Realstate\Clients.csv'
WITH (FIRSTROW = 2, FIELDTERMINATOR = ';', ROWTERMINATOR = '\n' );
```

```
Select*
From Clients;

BULK INSERT Agents
FROM 'D:\Data Analysis\My Projects\Data\Realstate\Agents.csv'
WITH (FIRSTROW = 2, FIELDTERMINATOR = ';', ROWTERMINATOR = '\n' );
Select*
From Agents;

BULK INSERT Sales
FROM 'D:\Data Analysis\My Projects\Data\Realstate\Sales.csv'
WITH (FIRSTROW = 2, FIELDTERMINATOR = ';', ROWTERMINATOR = '\n'  );
Select*
From Sales

BULK INSERT Visits
FROM 'D:\Data Analysis\My Projects\Data\Realstate\Visits.csv'
WITH (FIRSTROW = 2, FIELDTERMINATOR = ';', ROWTERMINATOR = '\n' );
Select*
From Visits;
```

Queries

```
Alter Table Agents
Add FullName AS (FirstName + ' '+ LastName);
Select *
from Agents
```

```
Alter Table Clients
Add FullName AS (FirstName + ' '+ LastName);
Select *
from Clients
```

- Both the **Agents** and **Clients** tables were updated by adding a computed column called **FullName**. This column is generated by combining the first name and last name into a single value, making it easier to display and read the full name directly in queries and reports.

FullName
Susan Adams
Tracy Daniel
Andrea Richardson
Victor Smith

```
ALTER TABLE Clients
ADD TotalVisited INT,
  TotalPurchased INT;
UPDATE c
SET
  c.TotalVisited = ISNULL(v.VisitCount, 0),
  c.TotalPurchased = ISNULL(s.PurchaseCount, 0)
FROM Clients c
LEFT JOIN ( SELECT ClientID, COUNT(*) AS VisitCount
 FROM Visits
 GROUP BY ClientID ) v ON c.ClientID = v.ClientID
```

```
LEFT JOIN ( SELECT ClientID, COUNT(*) AS PurchaseCount
 FROM Sales
 GROUP BY ClientID ) s ON c.ClientID = s.ClientID;
```

- The Clients table was updated by adding two new columns:
- TotalVisited ⟶ shows how many times each client has visited.
- TotalPurchased ⟶ shows how many purchases each client has made.
- These columns were then filled with summarized data from the Visits and Sales tables.
- Using LEFT JOIN ensures that all clients appear, even if they have zero visits or purchases.
- Using ISNULL(..., 0) guarantees that missing values are stored as 0 instead of NULL.

```
ALTER TABLE Clients
ADD IsRepeatBuyer BIT;
UPDATE Clients
SET IsRepeatBuyer = CASE
   WHEN s.PurchaseCount > 1 THEN 1
   ELSE 0
END
FROM Clients c
LEFT JOIN ( SELECT ClientID, COUNT(*) AS PurchaseCount
   FROM Sales
   GROUP BY ClientID ) s ON c.ClientID = s.ClientID;
```

- A new column IsRepeatBuyer was added to the Clients table.
- Data type is BIT, meaning the column will only store 1 (true) or 0 (false).
- The update checks the number of purchases for each client in the Sales table.
- If the client purchased more than once, IsRepeatBuyer is set to 1.
- Otherwise, the value is 0.
- Using LEFT JOIN ensures all clients are included, even those with no purchases.

```
ALTER TABLE Agents
ADD AgentNumber VARCHAR(20);
UPDATE Agents
SET AgentNumber = CAST(AgentID AS VARCHAR);
Select *
from Agents
```

- Added a new column AgentNumber to the Agents table.
- Data type is VARCHAR(20) to allow storing agent numbers as text (up to 20 characters)
- Populated the new column by converting the existing AgentID (numeric) into a text format (VARCHAR).
- This ensures the agent number is stored as a string, which allows more flexibility for formatting or adding prefixes in the future.

```
ALTER TABLE Clients
ADD EmailType VARCHAR(20);
UPDATE Clients
SET EmailType = CASE
  WHEN Email LIKE '%@gmail.%'
    OR Email LIKE '%@yahoo.%'
    OR Email LIKE '%@hotmail.%'
    OR Email LIKE '%@outlook.%'
    OR Email LIKE '%@icloud.%'
    OR Email LIKE '%@live.%'
    THEN 'Personal'
  ELSE 'Business'
END;
```

- Added a new column EmailType to the Clients table.

- Data type is VARCHAR(20) to store the classification of each client's email.
- Each client's email was analyzed to determine whether it is Personal or Business.
- If the email domain belongs to common providers (Gmail, Yahoo, Hotmail, Outlook, iCloud, Live), the EmailType is set to Personal.
- Otherwise, the EmailType is set to Business.

```
SELECT s.SaleID, s.SaleDate, s.SalePrice, s.ClientID, s.AgentID, p.PropertyID, p.PropertyType, p.Location, p.Size_sqm,
p.PriceUSD
INTO SalesProperty
FROM Sales s
JOIN Property p ON s.PropertyID = p.PropertyID;
Select *
from SalesProperty
```

- A new table called SalesProperty was created.
- This table combines information from both Sales and Property tables.
- The join was done using PropertyID (link between sales and properties).
- Columns included:
- From Sales: SaleID, SaleDate, SalePrice, ClientID, AgentID
- From Property: PropertyID, PropertyType, Location, Size (sqm), Price (USD)
- Using SELECT ... INTO both creates the new table and loads it with data in one step.

```
SELECT  p.PropertyType, AVG(s.SalePrice) AS
AvgSalesValue
FROM Sales s
JOIN Property p ON s.PropertyID = p.PropertyID
GROUP BY  p.PropertyType
Order By AvgSalesValue Desc;
```

Results | Messages

	PropertyType	AvgSalesValue
1	Warehouse	799185
2	Office	777720
3	Retail	766495
4	Villa	762813
5	Apartment	750744

- The query calculates the average sale price for each property type.
- It joins the Sales and Property tables using PropertyID.
- GROUP BY PropertyType ensures results are summarized by type (e.g., Apartment, Villa, Studio).
- ORDER BY AvgSalesValue DESC sorts the results from the highest to the lowest average sale value.

```
SELECT
   CAST(s.saleDate AS DATE) AS saleDate,
p.PropertyType, AVG(s.SalePrice) AS AvgSalesValue
FROM Sales s
JOIN Property p ON s.PropertyID = p.PropertyID
WHERE s.saleDate IS NOT NULL
GROUP BY CAST(s.saleDate AS DATE),
p.PropertyType
ORDER BY saleDate, AvgSalesValue DESC;
```

Results Messages

	saleDate	PropertyType	AvgSalesValue
1	2023-05-12	Warehouse	1457403
2	2023-05-12	Apartment	817371
3	2023-05-13	Apartment	373463
4	2023-05-14	Retail	1157984
5	2023-05-15	Retail	1133436
6	2023-05-15	Office	696763
7	2023-05-16	Retail	200664

- This query calculates the average sale price for each property type per day.
- CAST(s.saleDate AS DATE) is used to remove the time part and group only by the calendar date.
- WHERE s.saleDate IS NOT NULL ensures only valid sales dates are included.
- GROUP BY both saleDate and PropertyType gives a daily breakdown by property type.
- ORDER BY saleDate, AvgSalesValue DESC sorts results by date, and within each date, lists property types from highest to lowest average sale value.

```
SELECT p.PropertyType, COUNT(DISTINCT s.SaleID)
* 1.0 / NULLIF(COUNT(DISTINCT v.VisitID), 0)AS
ConversionRate
FROM Property p
LEFT JOIN Sales s ON p.PropertyID = s.PropertyID
LEFT JOIN Visits v ON p.PropertyID = v.PropertyID
GROUP BY p.PropertyType;
```

Results Messages

	PropertyType	ConversionRate
1	Apartment	0.463392029657
2	Villa	0.417560975609
3	Warehouse	0.347422680412
4	Office	0.374868004223
5	Retail	0.387860082304

- This query calculates the conversion rate for each property type.
- Conversion Rate = (Number of Sales ÷ Number of Visits).
- COUNT(DISTINCT s.SaleID) ⟶ counts unique sales per property type.
- COUNT(DISTINCT v.VisitID) ⟶ counts unique visits per property type.
- NULLIF(...,0) prevents division by zero (if there are no visits).
- LEFT JOIN ensures all property types are included, even if some have no sales or visits.

```
SELECT Top 5
  CONCAT(c.FirstName, ' ', c.LastName) AS FullName,
  COUNT(DISTINCT v.PropertyID) AS
PropertiesVisited
FROM Clients c
JOIN Visits v ON c.ClientID = v.ClientID
GROUP BY CONCAT(c.FirstName, ' ', c.LastName)
Order By PropertiesVisited DESC;
```

Results Messages

	FullName	PropertiesVisited
1	William Brown	14
2	Laura Allen	13
3	Michael Anderson	12
4	John Diaz	11
5	Alyssa Mendoza	10

- It combines the client's first and last name into one column called FullName.
- Then it counts how many different properties each client has purchased.

- Finally, it sorts them so the clients with the highest number of purchases come first, and only the top 5 are displayed.

```
SELECT top 5
 CONCAT(c.FirstName, ' ', c.LastName) AS FullName,
  SUM(s.SalePrice) AS TotalPurchasesValue
FROM Clients c
JOIN Sales s ON c.ClientID = s.ClientID
GROUP BY CONCAT(c.FirstName, ' ', c.LastName)
ORDER BY TotalPurchasesValue DESC;
```

Results | Messages

	FullName	TotalPurchasesValue
1	Laura Allen	7031490
2	William Brown	6078011
3	Miranda Gomez	5309528
4	Deborah Gibson	5059593
5	Brenda Cobb	4975928

- This query shows the top 5 clients ranked by the total amount of money they spent on property purchases.
- It puts together each client's first and last name in a single column called FullName.
- It then adds up (SUM) all the sale prices of the properties they bought to get their total purchase value.
- Finally, it sorts the list from the biggest spenders to the lowest and shows only the top 5.

```
SELECT  BuyerType, COUNT(*) AS TotalClients
FROM ( SELECT  CASE  WHEN COUNT(*) = 1 THEN
'First-time Buyer'
    ELSE 'Repeat Buyer'
    END AS BuyerType
  FROM Sales
  GROUP BY ClientID
) AS Buyers
GROUP BY BuyerType;
```

Results | Messages

	BuyerType	TotalClients
1	First-time Buyer	526
2	Repeat Buyer	570

- This query classifies clients into two categories based on how many purchases they made:
- First-time Buyer ⟶ clients who purchased only once.
- Repeat Buyer ⟶ clients who purchased more than once.
- Here's how it works step by step:

- Inside the subquery:
- For each client (GROUP BY ClientID), we count how many sales they made.
- If the count = 1 ⟶ label them as "First-time Buyer".
- Otherwise ⟶ label them as "Repeat Buyer".
- In the outer query:
- We group clients by these buyer types.
- Then count how many clients fall into each group (COUNT(*) AS TotalClients).

```
SELECT
  p.Location,
  COUNT(v.VisitID) AS TotalVisits
FROM Visits v
JOIN Property p ON v.PropertyID = p.PropertyID
GROUP BY p.Location;
```

Results Messages

	Location	TotalVisits
1	Miami	1052
2	Los Angeles	986
3	New York	1006
4	Houston	945
5	Chicago	1004

- We take the Visits table (which records client visits to properties).
- We join it with the Property table so we can know the location of each property.
- For every location (GROUP BY p.Location), we count how many visits happened (COUNT(v.VisitID)).

```
SELECT  p.Location,
  COUNT(s.SaleID) AS SalesCount,
  AVG(s.SalePrice) AS AvgSalePrice
FROM Sales s
JOIN Property p ON s.PropertyID = p.PropertyID
GROUP BY p.Location
ORDER BY SalesCount DESC, AvgSalePrice DESC;
```

Results Messages

	Location	SalesCount	AvgSalePrice
1	New York	428	776928
2	Miami	423	779114
3	Los Angeles	390	762868
4	Chicago	382	738575
5	Houston	374	787422

- Join Sales with Property ⟶ to know the location of each sale.

- COUNT(s.SaleID) AS SalesCount ⟶ counts how many sales happened in each location.
- AVG(s.SalePrice) AS AvgSalePrice ⟶ calculates the average sale price for each location.
- GROUP BY p.Location ⟶ groups the results by location.
- ORDER BY SalesCount DESC, AvgSalePrice DESC ⟶ sorts the results by the highest number of sales first, and if two locations have the same sales count, it sorts them by the highest average price.

```
SELECT  p.Location,  COUNT(DISTINCT s.SaleID) *
1.0 / COUNT(DISTINCT v.VisitID)AS VisitToSaleRatio
FROM Property p
LEFT JOIN Visits v ON p.PropertyID = v.PropertyID
LEFT JOIN Sales s ON p.PropertyID = s.PropertyID
GROUP BY p.Location
Order By VisitToSaleRatio DESC;
```

Results | Messages

	Location	VisitToSaleRatio
1	New York	0.425447316103
2	Miami	0.402091254752
3	Houston	0.395767195767
4	Los Angeles	0.395537525354
5	Chicago	0.380478087649

- This query calculates how effective each property location is at turning visits into actual sales.
- It counts the number of unique sales (SaleID) in each location and divides it by the number of unique visits (VisitID) to get the Visit-to-Sale Ratio.
- Then it groups the results by location so each area has its own ratio.
- Finally, it sorts the list in descending order, so the locations with the highest conversion rate from visits to sales appear first.

```
Select  Location ,PropertyType,COUNT(*) as
Number_of_Property
from Property
Group by  Location ,PropertyType
ORDER BY  Number_of_Property desc;
```

Results | Messages

	Location	PropertyType	Number_of_Property
1	Miami	Apartment	55
2	Miami	Villa	52
3	New York	Office	52
4	Los Angeles	Warehouse	46
5	Chicago	Retail	46
6	Los Angeles	Apartment	45
7	Houston	Warehouse	45
8	New York	Retail	42
9	Chicago	Villa	42

- This query shows how many properties are listed, broken down by both location and property type.

- It groups the data by each combination of location and property type.
- For each group, it counts the total number of properties and labels it as Number_of_Property.
- Finally, it sorts the results so that the property types and locations with the highest number of listings appear first.

```
Select Location as city
,round(AVG(priceUSD/Size_sqm) ,2)as
AvgPricePerSqm
from Property
group by Location
order by AvgPricePerSqm desc;
```

Results Messages

	city	AvgPricePerSqm
1	Chicago	2819
2	Miami	2725
3	Houston	2703
4	New York	2661
5	Los Angeles	2618

- This query calculates the average price per square meter of properties in each city.
- It divides the property price (priceUSD) by its size in square meters (Size_sqm) to get the price per sqm for each property.
- Then it calculates the average of these values for each city, rounding the result to 2 decimal places.
- The results are grouped by Location (shown as city).
- Finally, it sorts the list so that the cities with the highest average price per sqm appear at the top.

```
Select PropertyType ,count(*) AS
NumberOfProperties
from Property
group by PropertyType
order by  NumberOfProperties desc;
```

Results Messages

	PropertyType	NumberOfProperties
1	Apartment	215
2	Villa	208
3	Retail	195
4	Warehouse	191
5	Office	190

- This query shows the number of properties available by property type.
- It groups all the properties by their type (PropertyType).
- Then it counts how many properties fall under each type.
- The column NumberOfProperties shows the total count for each type.

- Finally, it orders the results so that the property type with the highest number of listings comes first.

```
SELECT
 CAST(saleDate AS DATE) AS saleDate
FROM Sales
WHERE saleDate IS NOT NULL
GROUP BY CAST(saleDate AS DATE)
ORDER BY saleDate
```

Results Messages

	saleDate
1	2023-05-12
2	2023-05-13
3	2023-05-14
4	2023-05-15
5	2023-05-16
6	2023-05-17
7	2023-05-18
8	2023-05-19
9	2023-05-20
10	2023-05-21

- This query shows the unique dates on which property sales happened.
- It takes the saleDate column from the Sales table.
- Since saleDate might include both date and time, the query converts it to just the date part (using CAST(... AS DATE)).
- It filters out any rows where saleDate is missing (IS NOT NULL).
- Then it groups the data by these dates so that each date appears only once.
- Finally, it sorts the list in chronological order (from the earliest sale to the latest).

```
Select top(10)PriceUSD,PropertyType, location
,Size_sqm
from Property
order by PriceUSD desc;
```

Results Messages

	PriceUSD	PropertyType	location	Size_sqm
1	998279	Apartment	Miami	488
2	997603	Office	New York	307
3	997351	Apartment	New York	245
4	997349	Villa	New York	229
5	996692	Warehouse	Chicago	392
6	996081	Warehouse	Los Angeles	206
7	994842	Retail	Miami	81
8	994440	Apartment	Miami	283
9	994201	Retail	New York	179
10	989997	Office	Houston	401

- This query lists the top 10 most expensive properties in the dataset.
- It looks at the Property table.
- It selects the property's price (PriceUSD), type (PropertyType), location, and size (Size_sqm).
- It orders all the properties from the highest price to the lowest using ORDER BY PriceUSD DESC.
- Finally, it shows only the first 10 results, which represent the most expensive properties.

```
Select top(10) p.PropertyType,count(v.VisitID)as
totalvisit
from Property p
left join Visits v on p.PropertyID=v.PropertyID
group by p.PropertyType
order by totalvisit desc;
```

Results Messages

	PropertyType	totalvisit
1	Apartment	1079
2	Villa	1025
3	Retail	972
4	Warehouse	970
5	Office	947

- This query shows the top 10 property types ranked by the number of visits.
- It looks at the Property table and joins it with the Visits table.
- For each property type (PropertyType), it counts how many visits (VisitID) the properties of that type received.
- The results are grouped by property type so each type has a single total.
- Then it orders the results from the most visited property types to the least.
- Finally, it shows only the top 10 property types based on total visits.

```
Select p.PropertyType, avg(SalePrice) as avg_sales
from Property p
join Sales s on p.PropertyID=s.PropertyID
group by p.PropertyType
order by avg_sales desc;
```

Results Messages

	PropertyType	avg_sales
1	Warehouse	799185
2	Office	777720
3	Retail	766495
4	Villa	762813
5	Apartment	750744

- This query shows the average sale price for each property type.
- It joins the Property table with the Sales table to connect each property with its sale information.
- For every property type (PropertyType), it calculates the average of the sale prices (SalePrice).
- Then it groups the results by property type so each type has its own average.
- Finally, it orders the list from the highest average sale price to the lowest, so you can easily see which property types sell for more on average.

```
Select a.FullName,a.AgentID,count(s.SaleID)AS
NumberOfSales
from Agents a
right join Sales s on a.AgentID=s.AgentID
group by a.AgentID,a.FullName
order by NumberOfSales desc;
```

Results Messages

	FullName	AgentID	NumberOfSales
1	Francisco Williams	91	29
2	Gordon Wilson	31	28
3	Michelle Davis	87	28
4	Samantha Vargas	93	28
5	Kelli Davis	17	27
6	Taylor Clark	36	26
7	Larry Holloway	6	26
8	Cody Ramsey	46	26
9	Ryan Wolf	9	26
10	Carolyn Gibbs	23	25

- This query shows the top-performing agents ranked by the number of sales they closed.
- It takes each agent's name (FullName) and their unique ID (AgentID).
- Then it counts how many property sales (SaleID) are linked to them.
- The RIGHT JOIN ensures that only sales with a matching agent are included.
- Results are grouped by each agent so we can see their total number of sales.
- Finally, it sorts the list from the highest number of sales to the lowest, so the most successful agents appear at the top.

```
Select a.AgentID,a.FullName ,count(v.VisitID)AS
NumberOfClientVisits
from Agents a
right join Visits v on a.AgentID=v.AgentID
group by a.AgentID,a.FullName
order by NumberOfClientVisits desc;
```

Results Messages

	AgentID	FullName	NumberOfClientVisits
1	23	Carolyn Gibbs	66
2	15	Julie Barrett	66
3	93	Samantha Vargas	64
4	40	Mark Lopez	64
5	92	Carolyn Terry	63
6	35	Ryan Gould	60
7	98	Jeremy Wade	60
8	24	Nicholas Shaffer	60
9	33	Paul Brooks	60
10	80	Kim Jackson	60

- This query shows the number of client visits handled by each agent.
- It takes each agent's ID (AgentID) and name (FullName).
- Then it counts how many client visits (VisitID) are linked to them.
- The RIGHT JOIN makes sure we only count visits that are connected to an agent.
- Results are grouped by agent so we can see the total visits per person.
- Finally, the list is ordered from the agent with the most visits to the least.

```
select a.AgentID, a.FullName
,round(COUNT(DISTINCT v.VisitID) * 1.0 /
NULLIF(COUNT(DISTINCT s.SaleID), 0),2) AS
Conversion_rateperagent
from Agents a
left join Visits v on a.AgentID=v.AgentID
left join Sales s on a.AgentID=s.AgentID
group by a.AgentID,a.FullName
order by Conversion_rateperagent desc;
```

Results Messages

	AgentID	FullName	Conversion_rateperagent
1	15	Julie Barrett	6.000000000000
2	45	Joshua Macias	4.540000000000
3	99	Robert Walker	4.380000000000
4	92	Carolyn Terry	4.200000000000
5	44	Charles Mccall	4.140000000000
6	18	Aimee Cherry	3.920000000000
7	57	Brian Watson	3.690000000000
8	49	Jacob Stevens	3.600000000000
9	59	Mark Rosales	3.570000000000
10	72	Luis Turner	3.470000000000

- This query calculates the conversion rate for each agent — meaning how effectively an agent turns client visits into actual property sales.
- It starts with each agent's ID (AgentID) and name (FullName).
- For each agent, it counts the unique client visits (VisitID) and the unique sales (SaleID).
- The conversion rate is calculated as:
- Number of Visits ÷ Number of Sales
- (with a check using NULLIF to avoid dividing by zero).
- The result is rounded to 2 decimal places.
- Finally, the agents are sorted from the highest conversion rate to the lowest.

```
select a.FullName, a.AgentID , count(s.SaleID)as
NumOfSales,round(avg(s.SalePrice),2) as Avgsale
from Agents a
right join Sales s on a.AgentID=s.AgentID
group by a.AgentID ,a.FullName
order by Avgsale ,numofsales desc;
```

Results Messages

	FullName	AgentID	NumOfSales	Avgsale
1	Kelly Mills	97	17	540365
2	Robert Walker	99	13	559343
3	Adrian Luna	32	14	567369
4	Monica Smith	89	16	574126
5	Charles Mccall	44	14	591499
6	John Johnson	55	21	596633
7	Luis Turner	72	15	610614
8	Tyler Alexander	29	22	618386
9	Debra Smith	7	14	618550
10	Kim Jackson	80	21	626080

- This query shows the average sale value handled by each agent along with the number of sales they closed:

- It selects each agent's name (FullName) and ID (AgentID).
- For every agent, it counts the total number of property sales (NumOfSales) using COUNT(s.SaleID).
- It then calculates the average sale price (Avgsale) handled by that agent using AVG(s.SalePrice), rounded to 2 decimals.
- The results are grouped by agent to ensure calculations are done per individual.
- Finally, the output is ordered first by the average sale value (Avgsale ascending), and then by the number of sales (NumOfSales descending) to break ties.

```
Select c.FullName,c.ClientID ,count(v.PropertyID) as
Numberofproperties
from Clients c
left join Visits v on c.ClientID=v.ClientID
group by c.FullName,c.ClientID
order by Numberofproperties desc
```

Results | Messages

	FullName	ClientID	Numberofproperties
1	Jenny Roberts	1002	10
2	Angela Rios	985	9
3	Courtney Evans	901	9
4	Jimmy Martinez	1118	9
5	Gregory Simpson	1115	9
6	Lauren Robbins	1171	9
7	Claudia Kelly	808	9
8	Karen Hansen	860	9
9	Dalton Fischer	105	9
10	Austin Mills	503	9

- This query shows how many properties each client has visited:
- It selects the client's name (FullName) and ID (ClientID).
- For each client, it counts the total number of properties they visited (NumberOfProperties) using COUNT(v.PropertyID).
- It uses a LEFT JOIN between Clients and Visits to make sure all clients are included, even those who haven't visited any property (they will just show 0).
- The results are grouped by each client (so the count is per client).
- Finally, it orders the output in descending order of the number of properties visited meaning the most active clients appear at the top.

```
Select top( 5) c.FullName,c.ClientID ,
sum(s.SalePrice)as totalsales
from Clients c
left join Sales s on c.ClientID=s.ClientID
group by c.FullName,c.ClientID
order by totalsales desc;
```

Results Messages

	FullName	ClientID	totalsales
1	Miranda Gomez	1034	5309528
2	Deborah Gibson	751	5059593
3	Brenda Cobb	1378	4975928
4	William Brown	315	4788386
5	James Andrade	973	4726575

- This query shows the top 5 clients ranked by their total sales value:
- It selects the client's name (FullName) and ID (ClientID).
- For each client, it calculates the total sales amount (TotalSales) using SUM(SalePrice).
- It uses a LEFT JOIN between Clients and Sales so that every client is included, even if they have no sales (their total will appear as NULL or 0).
- The results are grouped by each client (so the sales are aggregated per client).
- The query returns only the top 5 clients with the highest total sales using TOP(5).
- Finally, the output is ordered in descending order of TotalSales, so the most valuable clients appear at the top.

```
Select c.FullName,c.ClientID , CASE WHEN
COUNT(s.SaleID) = 1 THEN 'First-time Buyer'
ELSE 'Repeat Buyer'
END AS BuyerType,
COUNT(*) AS NumberOfClients
from Clients c
left join Sales s on c.ClientID=s.ClientID
group by c.FullName,c.ClientID
order by NumberOfClients desc;
```

Results Messages

	FullName	ClientID	BuyerType	NumberOfClients
1	Miranda Gomez	1034	Repeat Buyer	7
2	Hunter Moss	894	Repeat Buyer	6
3	Kimberly Bowen	466	Repeat Buyer	6
4	Michelle Martin	486	Repeat Buyer	5
5	Tiffany Garrett	571	Repeat Buyer	5
6	Jason Ramos	310	Repeat Buyer	5
7	William Brown	315	Repeat Buyer	5
8	Nicole Herman	316	Repeat Buyer	5
9	Laura Duke	153	Repeat Buyer	5
10	Misty Williams	177	Repeat Buyer	5

- This query classifies clients as either first-time buyers or repeat buyers:
- It selects each client's name (FullName) and ID (ClientID).

- Using a CASE expression, it checks the number of sales (COUNT(s.SaleID)):
- If the count is exactly 1 ⟶ the client is labeled "First-time Buyer".
- Otherwise (2 or more) ⟶ the client is labeled "Repeat Buyer".
- It also counts the total number of records per client as NumberOfClients.
- A LEFT JOIN ensures all clients appear, even if they haven't made any purchase (they won't be classified as buyers but still show up).
- The results are grouped by client so the calculation is done individually for each one.
- Finally, it orders the output by NumberOfClients in descending order — meaning clients with the most records/sales appear at the top.

```
Select c.FullName ,c.ClientID ,count(v.VisitID)as
totalvisit,p.Location
from Clients c
right join Visits v on c.ClientID=v.ClientID
right join Property p on p.PropertyID=v.PropertyID
group by c.FullName,c.ClientID,p.Location
order by  totalvisit desc;
```

Results | Messages

	FullName	ClientID	totalvisit	Location
1	Jennifer Hall	856	5	New York
2	Jenny Roberts	1002	5	Chicago
3	Julie Hood	408	5	Chicago
4	Karen Hansen	860	4	New York
5	Jimmy Martinez	1118	4	Houston
6	Jon Douglas	573	4	Houston
7	Jeffrey Walker	1233	4	Miami
8	James Romero	602	4	Houston
9	Jamie Tanner	1119	4	Miami
10	Kevin Smith	148	4	Miami

- This query analyzes client interest in properties across different regions (cities):
- It selects each client's name (FullName) and ID (ClientID).
- It counts the total number of visits (COUNT(v.VisitID)) per client as TotalVisit.
- It also pulls in the property's location (p.Location) to know which city/region the visits happened in.
- The query uses RIGHT JOINs:
- Between Clients and Visits ⟶ so only clients who visited at least one property will appear.
- Between Visits and Property ⟶ to link each visit with the property's location.
- The results are grouped by client and location, meaning for each client you'll see how many properties they visited per city.

- Finally, it orders the output in descending order of total visits, so the clients with the most activity (by city) appear first.

```
Select count(v.VisitID)as totalvisit,p.Location
from Clients c
right join Visits v on c.ClientID=v.ClientID
right join Property p on p.PropertyID=v.PropertyID
group by p.Location
order by  totalvisit desc;
```

Results | Messages

	totalvisit	Location
1	1052	Miami
2	1006	New York
3	1004	Chicago
4	986	Los Angeles
5	945	Houston

- This query calculates the total number of property visits per location (city/region):
- It counts the number of visits (COUNT(v.VisitID)) for each location and labels it as TotalVisit.
- It selects the property's location (p.Location) to group visits by city/region.
- It uses RIGHT JOINs:
- Clients ⟶ Visits to include only clients who actually made visits.
- Visits ⟶ Property to link each visit with the property's location.
- The results are grouped by location, meaning you'll see the total number of visits for each city/region.
- Finally, the output is ordered in descending order of visits, so the most visited locations appear first.

```
Select p.PropertyType ,sum(s.SalePrice) as totalsales
from Property p
left join Sales s on p.PropertyID=s.PropertyID
group by p.PropertyType
order by totalsales desc ;
```

Results | Messages

	PropertyType	totalsales
1	Apartment	375372119
2	Villa	326484167
3	Retail	288968872
4	Office	276090710
5	Warehouse	269325634

- This query shows the total sales value generated by each property type:
- It selects the property type (p.PropertyType) from the Property table.
- For each type, it calculates the sum of sales prices (SUM(s.SalePrice)) from the Sales table, and labels it as TotalSales.

- The query uses a LEFT JOIN between Property and Sales, so all property types are included — even if a property type has no sales (its total will just appear as NULL or 0).
- The results are grouped by property type, meaning the sum is calculated per category (e.g., Apartments, Villas, Offices, etc.).
- Finally, it orders the output by total sales in descending order, so the property types bringing in the highest revenue appear first.

```
Select p.Location,count(s.SaleID) as totalsales
,sum(s.SalePrice) as totalprice
from Property p
left join Sales s on p.PropertyID=s.PropertyID
group by p.Location
order by totalsales desc ,totalprice desc;
```

Results | Messages

	Location	totalsales	totalprice
1	New York	428	332525336
2	Miami	423	329565569
3	Los Angeles	390	297518749
4	Chicago	382	282135816
5	Houston	374	294496032

- This query shows sales performance across different property locations:
- It selects the location of each property (p.Location).
- For each location, it calculates:
- The number of sales (COUNT(s.SaleID)) ⟶ labeled as TotalSales.
- The total revenue from sales (SUM(s.SalePrice)) ⟶ labeled as TotalPrice.
- It uses a LEFT JOIN between Property and Sales so that all locations appear, even if some locations have no sales.
- The results are grouped by location, meaning both totals are calculated per city/area.
- Finally, the results are ordered first by number of sales in descending order, then by total revenue so the busiest and highest-grossing locations appear at the top.

```
Select p.PropertyID,p.location ,count(distinct
(s.SaleID ))/nullif(count(distinct(v.VisitID )),0 )as
Conversion_rate
from Property p
left join Sales s on p.PropertyID=s.PropertyID
left join Visits v on p.PropertyID=v.PropertyID
group by p.PropertyID,p.location
order by Conversion_rate desc;
```

Results | Messages

	PropertyID	location	Conversion_rate
1	12	Miami	5
2	77	Houston	5
3	979	Miami	5
4	797	Houston	4
5	707	Chicago	3
6	906	New York	3
7	147	Los Angeles	3
8	129	Los Angeles	3
9	265	Los Angeles	3
10	355	Houston	3

- This query calculates the conversion rate of visits into sales for each property and location:
- p.PropertyID, p.Location ⟶ Identifies each property and where it's located.
- COUNT(DISTINCT s.SaleID) ⟶ Counts the number of unique sales for that property.
- COUNT(DISTINCT v.VisitID) ⟶ Counts the number of unique visits for that property.
- / NULLIF(..., 0) ⟶ Divides sales by visits to get the conversion rate (i.e., how many visits turned into sales). NULLIF avoids division by zero if a property has no visits.
- The query uses LEFT JOINs so even properties with no sales or visits will still appear.
- GROUP BY p.PropertyID, p.Location ⟶ Ensures the conversion rate is calculated per property.
- ORDER BY Conversion_rate DESC ⟶ Sorts results so properties with the highest visit-to-sale conversion come first.