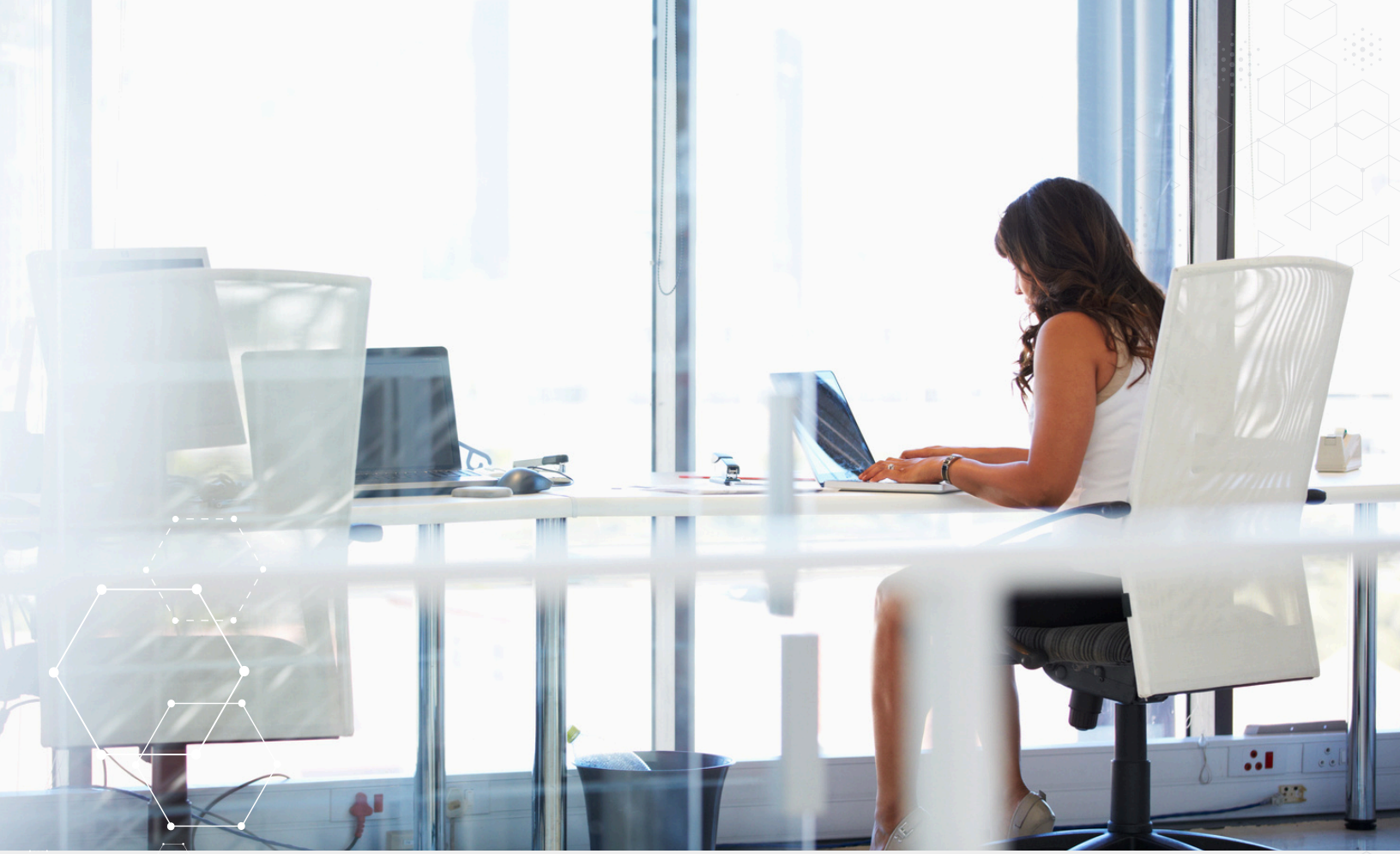
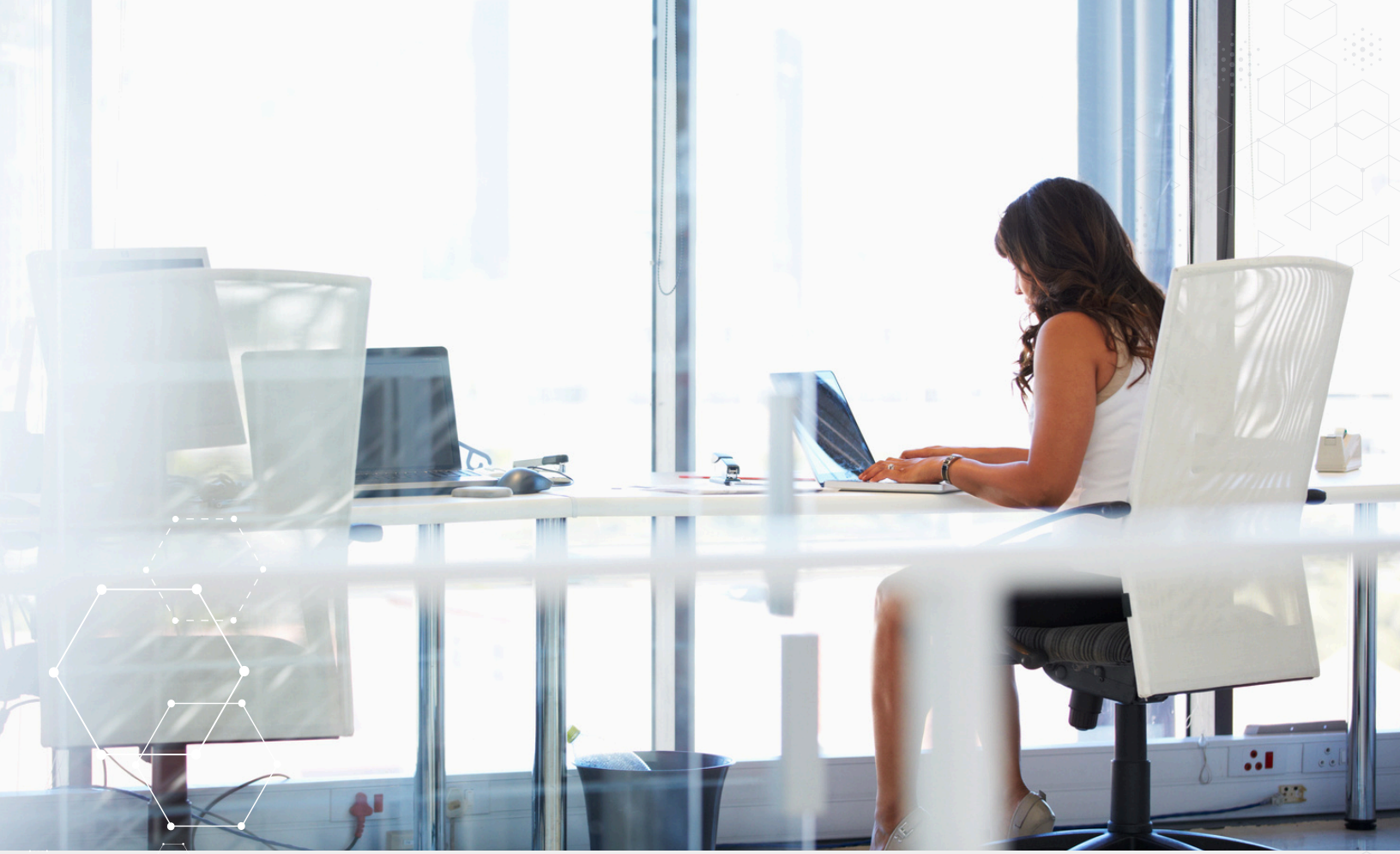


SAP Business One

Advanced Query Writing Guide

Quaint
business solutions



SAP Business One

Aggregate Functions

Quaint
business solutions

AGGREGATE FUNCTIONS

What are aggregate functions?

Aggregate functions perform a calculation on a set of values and return a single result

- Commonly used to summarize large datasets
- Combined with the Group By Clause
  - Group By : Groups rows sharing a property

Common Aggregate Functions:

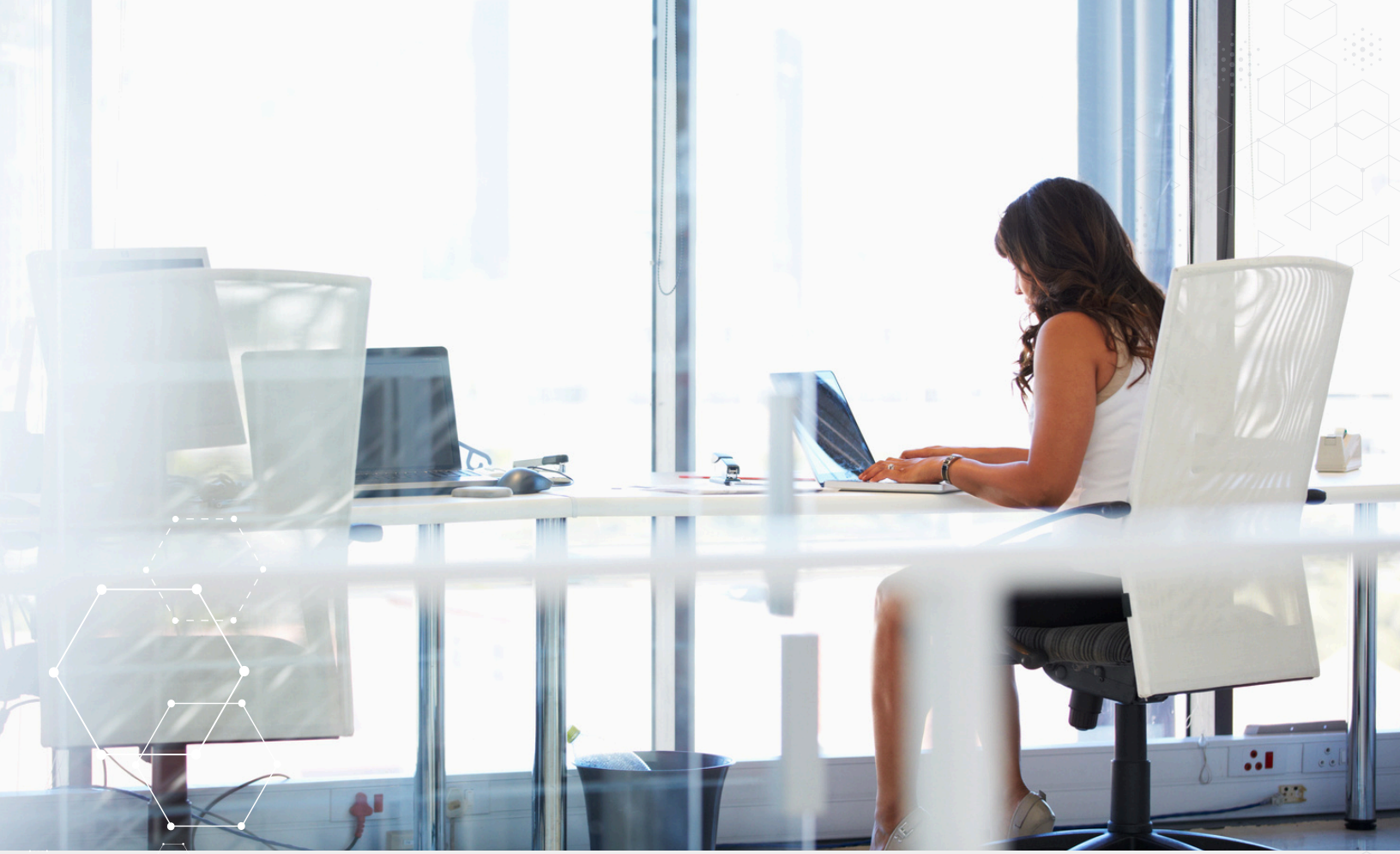
- **COUNT**
  - Returns the number of rows
  - Example: SELECT COUNT(*) FROM OCRD - returns the number of BPs

- **SUM**
- Returns the total sum of a numeric column
  - Example: SELECT SUM(Quantity) FROM INV1 – returns the total quantity sold

- **AVG**
  - Returns the average value of a numeric column
  - SELECT AVG(Price) FROM INV1 – returns average unit price

- **MAX**
  - Returns the maximum value of a column
  - Example: SELECT MAX(DocDate) FROM OINV – returns the most recent invoice date

- **MIN**
  - Returns the minimum value of a column
  - Example: SELECT MIN(Quantity) from INV1 – returns the minimum quantity on an invoice



SAP Business One

Group By, Having vs. Where

Quaint
business solutions

GROUP BY, HAVING VS. WHERE

Query section order review:

SELECT	[Field names separated by commas]
FROM	[Table names separated by joins]
WHERE	[Conditions or Filtering of the Report]
GROUP BY	[Group the Report - fields not grouped must be functions (i.e. sum, count)]
HAVING	[Condition on Grouped Fields]
ORDER BY	[Sort the Report - can include Descending / Ascending]

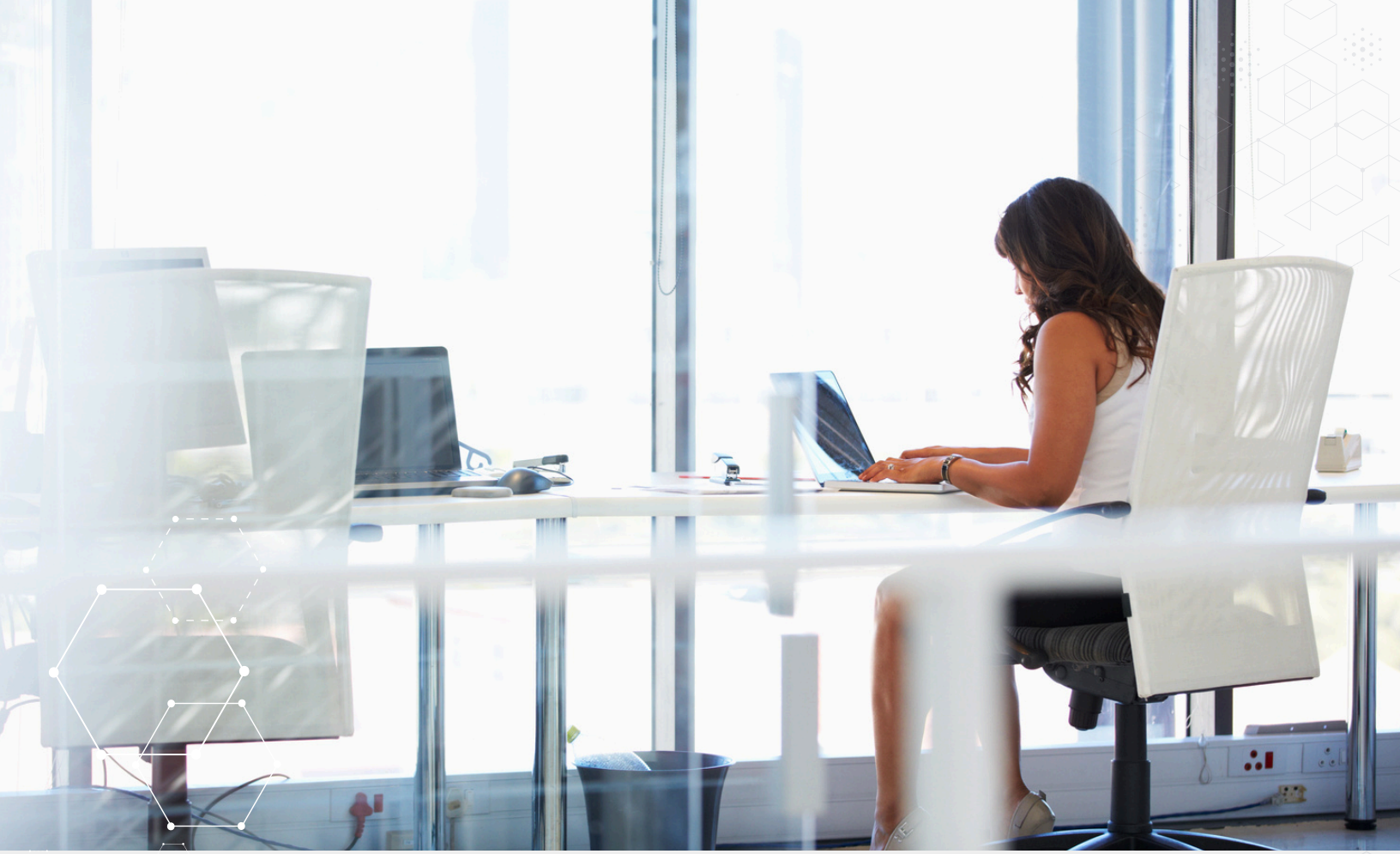
Why use HAVING instead of WHERE?

- **WHERE**
  - Filters rows before grouping
  - Cannot be used with aggregate functions

- **HAVING**
  - Filters groups after the GROUP BY clause is applied
  - Can be used with aggregate functions like COUNT, AVG, SUM, etc.

Examples:

- **WHERE**
  - SELECT T1.[GroupName], COUNT(*) as 'Count'
  - FROM OCRD T0 INNER JOIN OCRG T1 ON T0.GroupCode = T1.GroupCode
  - WHERE T1.GroupName LIKE '%%o%%'
  - GROUP BY T1.[GroupName]
- **HAVING**
  - SELECT T1.[GroupName], COUNT(*) as 'Count'
  - FROM OCRD T0 INNER JOIN OCRG T1 ON T0.GroupCode = T1.GroupCode
  - GROUP BY T1.[GroupName]
  - HAVING COUNT(*) >= 3



SAP Business One

Order By & Top

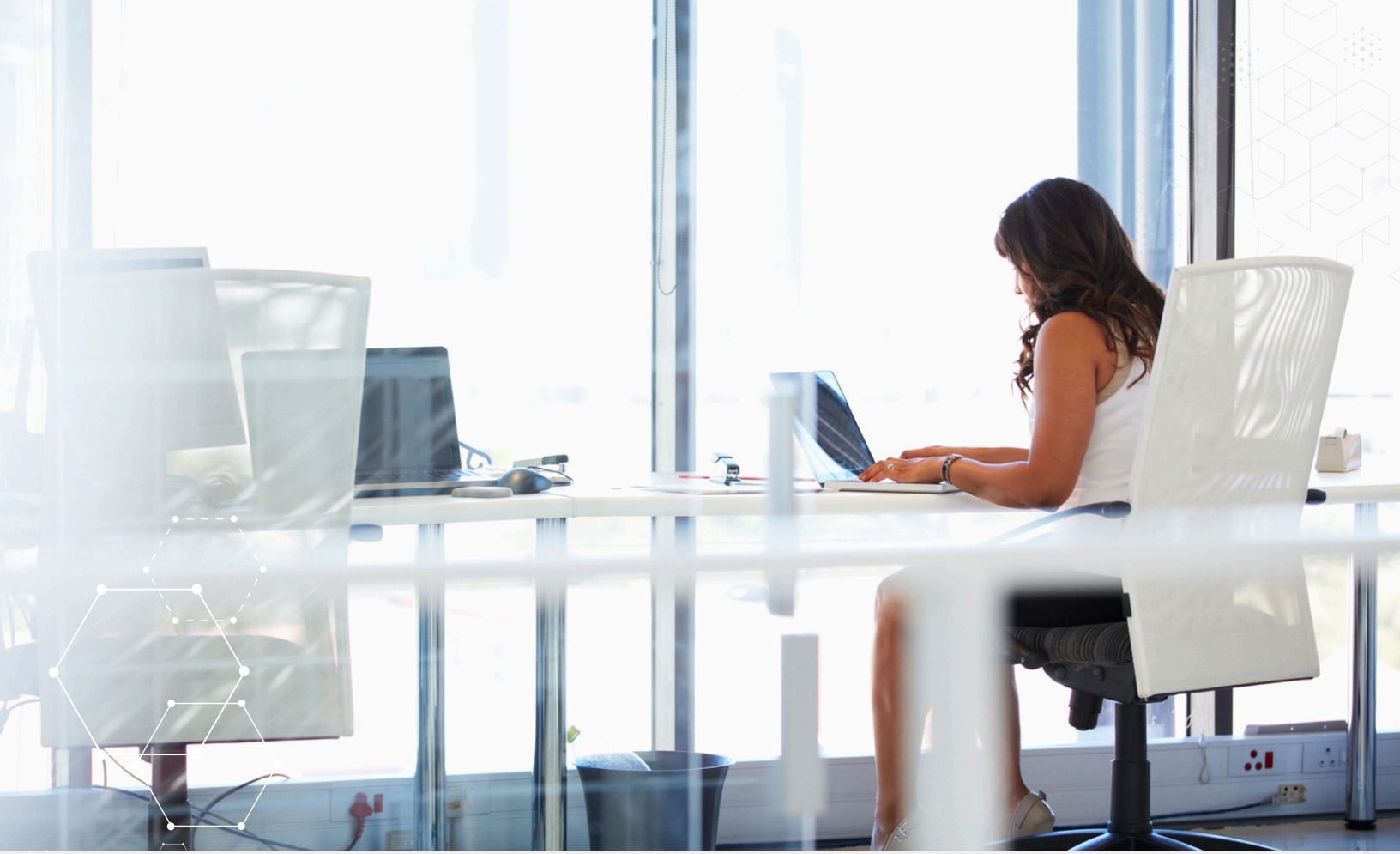
Quaint
business solutions

ORDER BY & TOP

When should I use Order By or Top?

- **ORDER BY**
  - Sorts the result set by one or more columns, either ascending (ASC) or descending (DESC)
  - Can sort by multiple columns
  - Defaults to ascending order if no direction is specified
  - Does not limit the number of rows returned; it only controls order

- **TOP**
  - Limits the number of rows returned based on the specified number or percentage
  - Often used together with Order By to return top “N” rows of a sorted result



SAP Business One

Case Statements

Quaint
business solutions

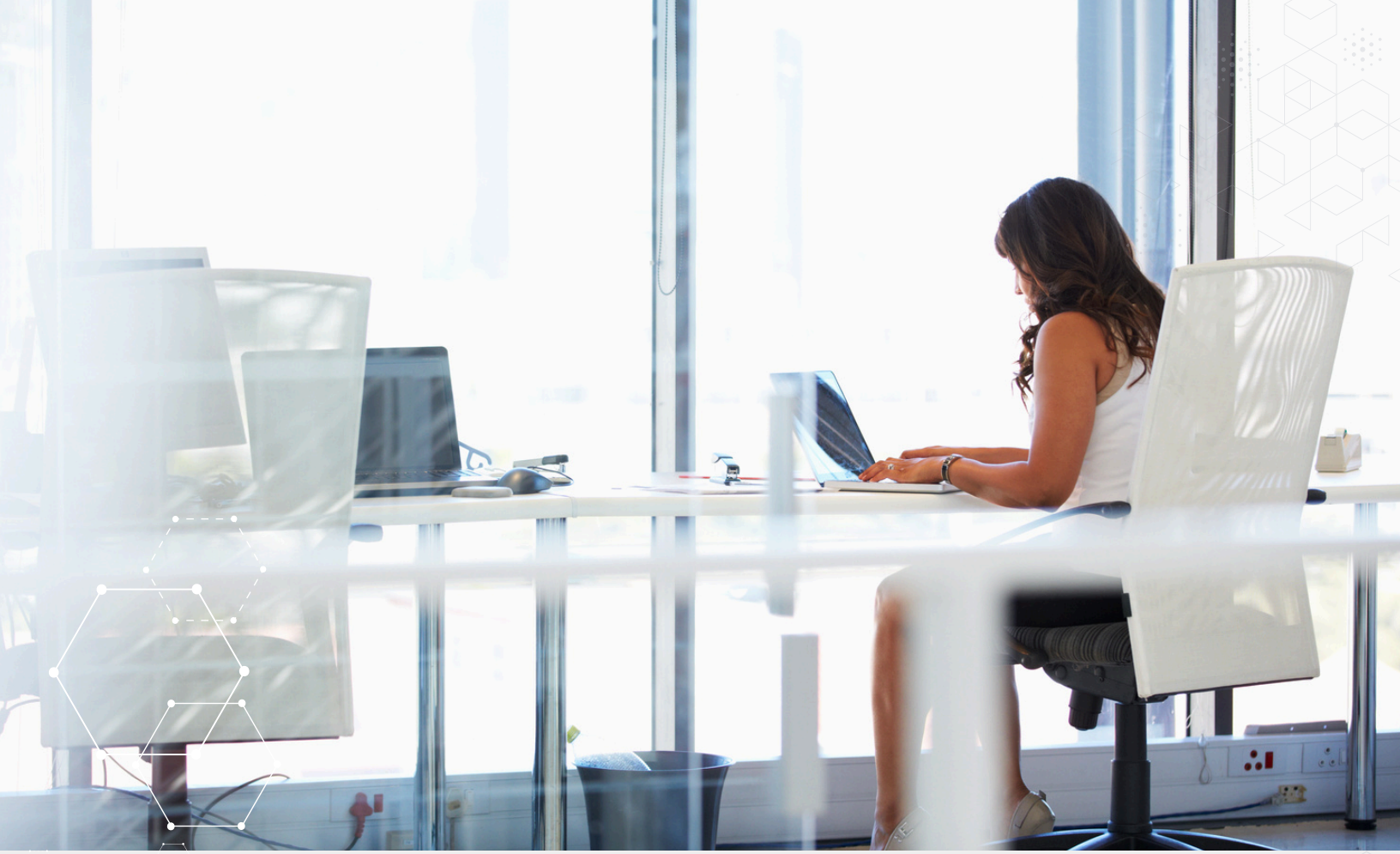
CASE STATEMENTS

What is a CASE STATEMENT?

- A **CASE** statement allows you to apply conditional logic inside your SQL queries
- It works like an **IF – THEN – ELSE** statement, returning different values based on conditions
  - **WHEN** - Specifies the condition
  - **THEN** – Defines the result when the condition is met
  - **ELSE** – (Optional) Provides a default result when no conditions are met
  - **END** – (Required) Marks the end of the **CASE** statement

Key Notes about CASE Statements :

- **ORDER MATTERS** – it works with a top-down approach
- Works with Aggregations
- Can be used in **SELECT**, **WHERE**, **ORDER BY** or even **GROUP BY** clauses
- Can return different data types, but all return values must be the same data type



SAP Business One

Stored Variables & Declarations

Quaint
business solutions

STORED VARIABLES & DECLARATIONS

What are stored variables in SAP SQL?

- Variables in SQL allow you to store data temporarily during the execution of a query
- Stored variables must be declared before they can be used
- SAP allows us to use [%0], [%1], [%2], etc. for user prompted variables
- There are syntax differences in declarations between HANA and SQL
    - Please refer to https://www.quaintbusiness.com/sql-vs-hana

Syntax used:

DECLARE @variablename DataType;
SET @variablename = Value;

Common Data Types:

INT = Integer value; can store whole numbers
CHAR(n) = Fixed length character string; n specifies the length
VARCHAR(n) = Variable length character string; n specifies max length
TEXT = Variable length string; used for large text data

STORED VARIABLES & DECLARATIONS

Common Data Types:

INT = Integer value; can store whole numbers
FLOAT = Floating point number; used in approximate values
DECIMAL(p,s) = Fixed point number; p is the total number of digits, s is the number of digits after the decimal point
CHAR(n) = Fixed length character string; n specifies the length
VARCHAR(n) = Variable length character string; n specifies max length
TEXT = Variable length string; used for large text data
DATE = Stores date values (year, month day)
TIME = Stores time values (hour, minute, second)
DATETIME = Combines date and time into one value
TIMESTAMP = Stores date and time with time zone support

Example:

```
/* SELECT FROM OINV T0 */
DECLARE @MinSales INT;
DECLARE @StartDate DATE;
DECLARE @EndDate DATE;
 SET @MinSales = 10;
 SET @StartDate = /* T0.DocDate */ [%0];
SET @EndDate = /* T0.DocDate */ [%1];

SELECT T0.ItemCode, T0.Dscription, SUM(T0.Quantity) AS 'Total Quantity'
FROM INV1 T0
WHERE T0.DocDate >= @StartDate and T0.DocDate <= @EndDate
GROUP BY T0.ItemCode, T0.Dscription
HAVING SUM(T0.Quantity) >= @MinSales
ORDER BY SUM(T0.Quantity) DESC
```