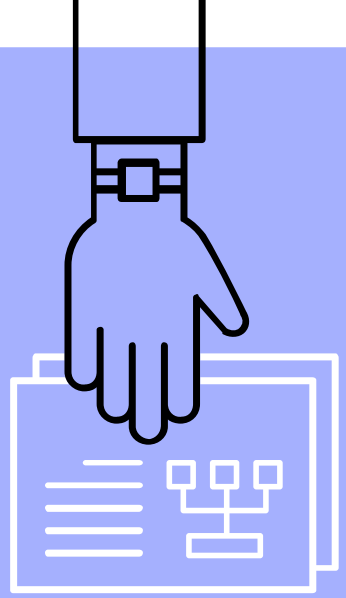
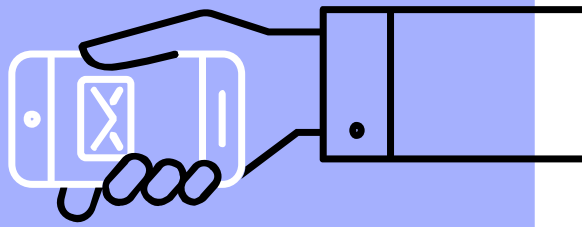
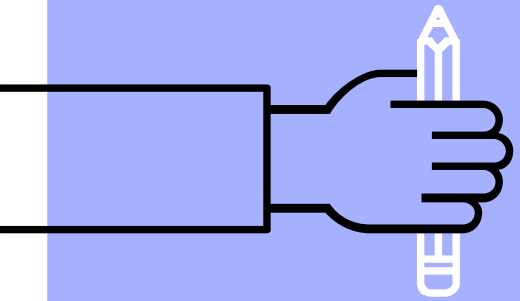
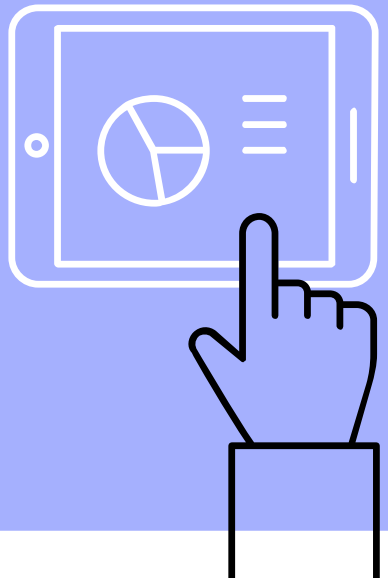




In-Database Analytics

DS with Python
Lecture 7





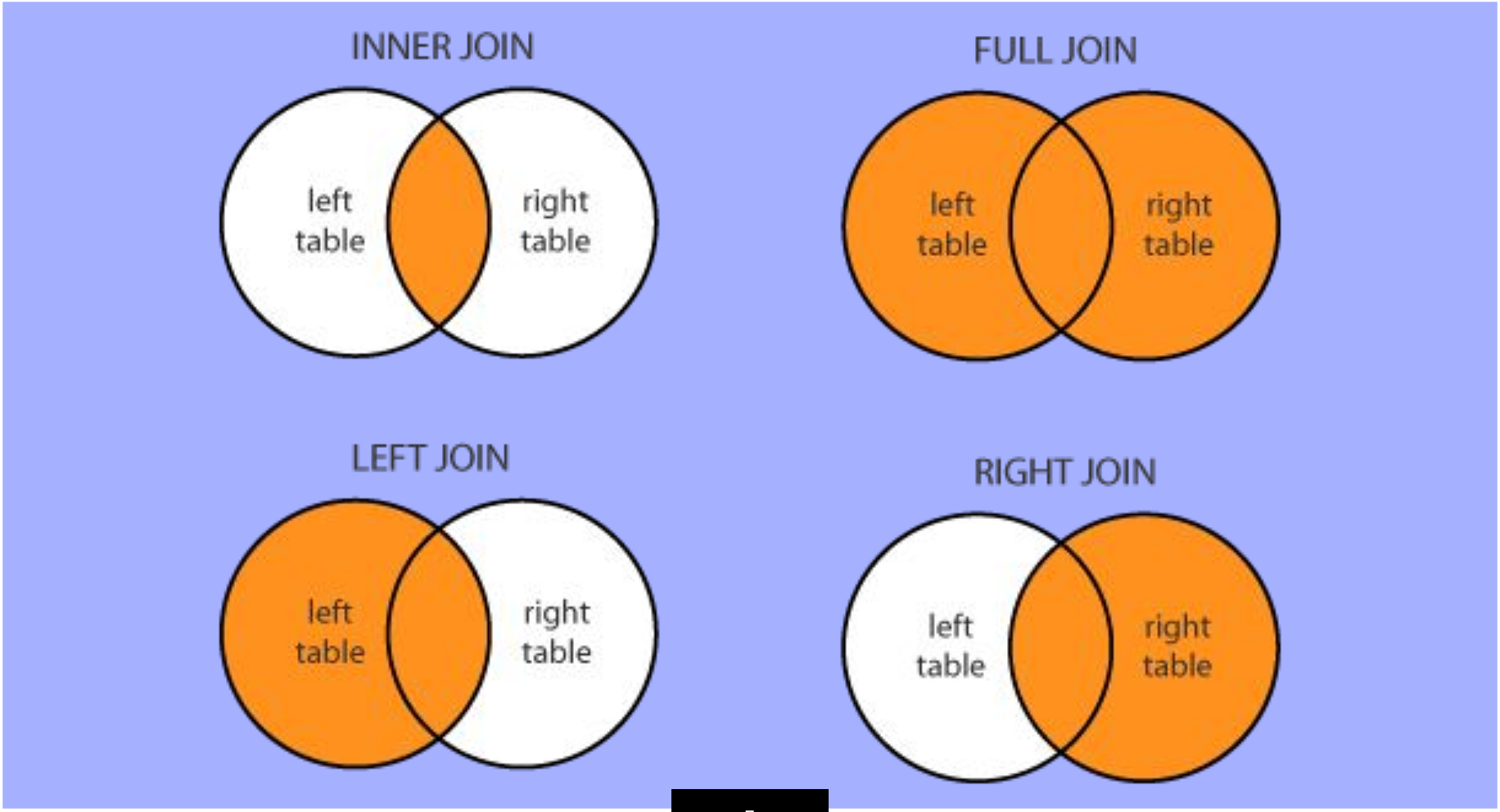
SQL JOIN

SQL ESSENTIALS

- A SQL JOIN combines records from two tables.
- A JOIN locates related column values in the two tables.
- A query can contain zero, one, or multiple JOIN operations.
- INNER JOIN is the same as JOIN; the keyword INNER is optional.

Different Types of JOINS

SQL ESSENTIALS


INNER JOIN
left table
right table
FULL JOIN
left table
right table
LEFT JOIN
left table
right table
RIGHT JOIN
left table
right table

SQL JOIN SYNTAX

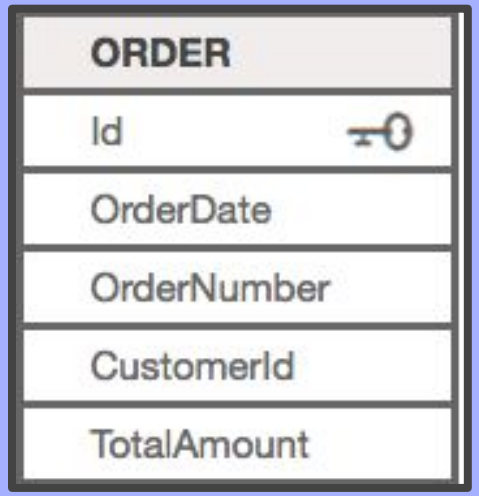
```
1.  SELECT column-names
2.    FROM table-name1 JOIN table-name2
3.      ON column-name1 = column-name2
4.   WHERE condition
```

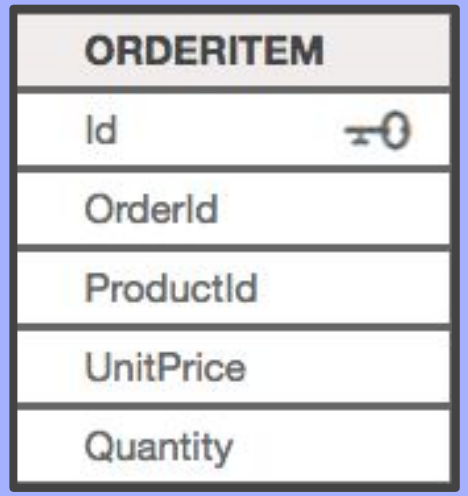
```
1.  SELECT column-names
2.    FROM table-name1 INNER JOIN table-name2
3.      ON column-name1 = column-name2
4.   WHERE condition
```

Note: The INNER keyword is optional.

SQL JOIN EXAMPLE

Problem: List all orders with product names, quantities, and prices





SQL JOIN EXAMPLE

Problem: List all orders with product names, quantities, and prices

```sql
SELECT O.OrderNumber, CONVERT(date,O.OrderDate) AS Date,
       P.ProductName, I.Quantity, I.UnitPrice
  FROM [Order] O
  JOIN OrderItem I ON O.Id = I.OrderId
  JOIN Product P ON P.Id = I.ProductId
ORDER BY O.OrderNumber
```

SQL JOIN EXAMPLE

Problem: List all orders with product names, quantities, and prices

OrderNumber	Date	ProductName	Quantity	UnitPrice
542378	7/4/2012 12:00:00 AM	Queso Cabrales	12	14.00
542378	7/4/2012 12:00:00 AM	Singaporean Hokkien Fried Mee	10	9.80
542378	7/4/2012 12:00:00 AM	Mozzarella di Giovanni	5	34.80
542379	7/5/2012 12:00:00 AM	Tofu	9	18.60
542379	7/5/2012 12:00:00 AM	Manjimup Dried Apples	40	42.40
542380	7/8/2012 12:00:00 AM	Jack's New England Clam Chowder	10	7.70
542380	7/8/2012 12:00:00 AM	Manjimup Dried Apples	35	42.40
542380	7/8/2012 12:00:00 AM	Louisiana Fiery Hot Pepper Sauce	15	16.80
542381	7/8/2012 12:00:00 AM	Gustaf's Knäckebröd	6	16.80
542381	7/8/2012 12:00:00 AM	Ravioli Angelo	15	15.60

SQL LEFT JOIN SYNTAX

```
1.  SELECT column-names
2.    FROM table-name1 LEFT JOIN table-name2
3.      ON column-name1 = column-name2
4.   WHERE condition
```

```
1.  SELECT OrderNumber, TotalAmount, FirstName, LastName, City, Country
2.    FROM Customer C LEFT JOIN [Order] O
3.      ON O.CustomerId = C.Id
4.  ORDER BY TotalAmount
```

- **This will list all customers, whether they placed any order or not.**

SQL LEFT JOIN SYNTAX

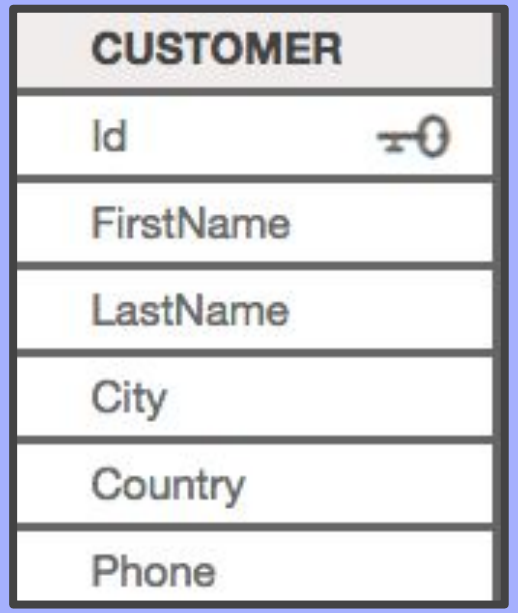
```
1.  SELECT column-names
2.    FROM table-name1 RIGHT OUTER JOIN table-name2
3.      ON column-name1 = column-name2
4.   WHERE condition
```

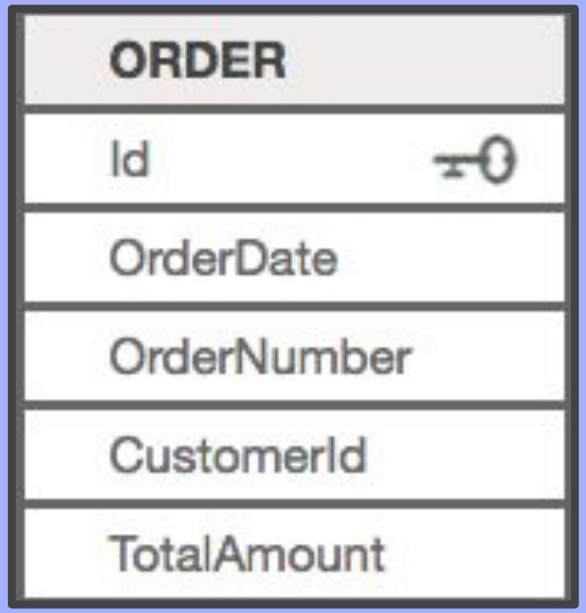
```
1.  SELECT column-names
2.    FROM table-name1 RIGHT JOIN table-name2
3.      ON column-name1 = column-name2
4.   WHERE condition
```

- **RIGHT JOIN and RIGHT OUTER JOIN are the same.**

SQL LEFT JOIN EXAMPLE

Problem: List customers that have not placed orders





SQL LEFT JOIN EXAMPLE

Problem: List customers that have not placed orders

```
1.  SELECT TotalAmount, FirstName, LastName, City, Country
2.    FROM [Order] O RIGHT JOIN Customer C
3.      ON O.CustomerId = C.Id
4.  WHERE TotalAmount IS NULL
```

This returns customers that, when joined, have no matching order.

SQL LEFT JOIN EXAMPLE

Problem: List customers that have not placed orders

Results: 2 records

TotalAmount	FirstName	LastName	City	Country
NULL	Diego	Roel	Madrid	Spain
NULL	Marie	Bertrand	Paris	France

This returns customers that, when joined, have no matching order.

SQL LEFT JOIN SYNTAX

```
1.      SELECT column-names
2.    FROM table-name1 FULL OUTER JOIN table-name2
3.      ON column-name1 = column-name2
4.   WHERE condition
```

```
1.  SELECT column-names
2.    FROM table-name1 FULL JOIN table-name2
3.      ON column-name1 = column-name2
4.   WHERE condition
```

SQL LEFT JOIN EXAMPLE

Problem: Match all customers and suppliers by country

SUPPLIER
Id
CompanyName
ContactName
City
Country
Phone
Fax

CUSTOMER
Id
FirstName
LastName
City
Country
Phone

SQL LEFT JOIN EXAMPLE

Problem: Match all customers and suppliers by country

```
1.  SELECT C.FirstName, C.LastName, C.Country AS
    CustomerCountry, S.Country AS SupplierCountry ,
    S.CompanyName
2.   FROM Customer C FULL JOIN Supplier S
3.     ON C.Country = S.Country
4.  ORDER BY C.Country, S.Country
```

SQL LEFT JOIN EXAMPLE

Problem: Match all customers and suppliers by country

This returns suppliers that have no customers in their country,
and customers that have no suppliers in their country,
and customers and suppliers that are from the same country……..

SQL LEFT JOIN EXAMPLE

Problem: Match all customers and suppliers by country

FirstName	LastName	CustomerCountry	SupplierCountry	CompanyName
NULL	NULL	NULL	Australia	Pavlova, Ltd.
NULL	NULL	NULL	Australia	G'day, Mate
NULL	NULL	NULL	Japan	Tokyo Traders
NULL	NULL	NULL	Japan	Mayumi's
NULL	NULL	NULL	Netherlands	Zaanse Snoepfabriek
NULL	NULL	NULL	Singapore	Leka Trading
Patricio	Simpson	Argentina	NULL	NULL
Yvonne	Moncada	Argentina	NULL	NULL
Sergio	Gutiérrez	Argentina	NULL	NULL
Georg	Pipps	Austria	NULL	NULL
Roland	Mendel	Austria	NULL	NULL
Pascale	Cartrain	Belgium	NULL	NULL
Catherine	Dewey	Belgium	NULL	NULL
Bernardo	Batista	Brazil	Brazil	Refrescos Americanas LTDA
Lúcia	Carvalho	Brazil	Brazil	Refrescos Americanas LTDA
Janete	Limeira	Brazil	Brazil	Refrescos Americanas LTDA

SQL SELF JOIN

- A self JOIN occurs when a table takes a 'selfie'.
- A self JOIN is a regular join but the table is joined with itself.
- This can be useful when modeling hierarchies.
- They are also useful for comparisons within a table.

SQL SELF JOIN SYNTAX

```
1.  SELECT column-names
2.    FROM table-name T1 JOIN table-name T2
3.    WHERE condition
```

T1 and T2 are different table aliases for the same table

SQL SELF JOIN EXAMPLE

Problem: Match customers that are from the same city and country

CUSTOMER	CUSTOMER
Id	Id
FirstName	FirstName
LastName	LastName
City	City
Country	Country
Phone	Phone

SQL SELF JOIN EXAMPLE

Problem: Match customers that are from the same city and country

```sql
SELECT B.FirstName AS FirstName1, B.LastName AS LastName1,
       A.FirstName AS FirstName2, A.LastName AS LastName2,
       B.City, B.Country
  FROM Customer A, Customer B
 WHERE A.Id <> B.Id
   AND A.City = B.City
   AND A.Country = B.Country
 ORDER BY A.Country
```

SQL SELF JOIN EXAMPLE

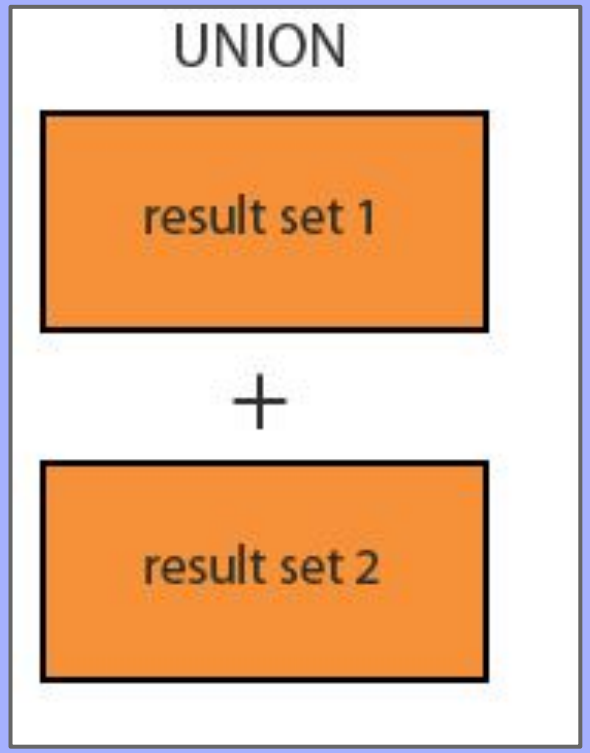
Problem: Match customers that are from the same city and country

FirstName1	LastName1	FirstName2	LastName2	City	Country
Patricio	Simpson	Yvonne	Moncada	Buenos Aires	Argentina
Patricio	Simpson	Sergio	Gutiérrez	Buenos Aires	Argentina
Yvonne	Moncada	Patricio	Simpson	Buenos Aires	Argentina
Yvonne	Moncada	Sergio	Gutiérrez	Buenos Aires	Argentina
Sergio	Gutiérrez	Patricio	Simpson	Buenos Aires	Argentina
Sergio	Gutiérrez	Yvonne	Moncada	Buenos Aires	Argentina
Anabela	Domingues	Lúcia	Carvalho	Sao Paulo	Brazil
Anabela	Domingues	Aria	Cruz	Sao Paulo	Brazil
Anabela	Domingues	Pedro	Afonso	Sao Paulo	Brazil
Bernardo	Batista	Janete	Limeira	Rio de Janeiro	Brazil
Bernardo	Batista	Mario	Pontes	Rio de Janeiro	Brazil
Lúcia	Carvalho	Anabela	Domingues	Sao Paulo	Brazil

SQL UNION (Set Operations)

- UNION combines the result sets of two queries.
- Column data types in the two queries must match.
- UNION combines by column position rather than column name.

SQL UNION (Set Operations)



SQL UNION SYNTAX

```
1.  SELECT column-names
2.    FROM table-name
3.   UNION
4.  SELECT column-names
5.    FROM table-name
```

SQL UNION EXAMPLE

Problem: List all contacts, i.e., suppliers and customers.

SQL UNION EXAMPLE

Problem: List all contacts, i.e., suppliers and customers.

This is a simple example in which the table alias would be useful

```sql
SELECT 'Customer' As Type,
       FirstName + ' ' + LastName AS ContactName,
       City, Country, Phone
  FROM Customer
UNION
SELECT 'Supplier',
       ContactName, City, Country, Phone
  FROM Supplier
```

SQL UNION EXAMPLE RESULTS

Results:

Type	ContactName	City	Country	Phone
Customer	Alejandra Camino	Madrid	Spain	(91) 745 6200
Customer	Alexander Feuer	Leipzig	Germany	0342-023176
Customer	Ana Trujillo	México D.F.	Mexico	(5) 555-4729
Customer	Anabela Domingues	Sao Paulo	Brazil	(11) 555-2167
. . .				
Supplier	Anne Heikkonen	Lappeenranta	Finland	(953) 10956
Supplier	Antonio del Valle Saavedra	Oviedo	Spain	(98) 598 76 54
Supplier	Beate Vileid	Sandvika	Norway	(0)2-953010
Supplier	Carlos Diaz	Sao Paulo	Brazil	(11) 555 4640
Supplier	Chandra Leka	Singapore	Singapore	555-8787
Supplier	Chantal Goulet	Ste-Hyacinthe	Canada	(514) 555-2955
Supplier	Charlotte Cooper	London	UK	(171) 555-2222
. . .				

SQL SUBQUERIES

- A subquery is a SQL query within a query.
- Subqueries are nested queries that provide data to the enclosing query.
- Subqueries can return individual values or a list of records
- Subqueries must be enclosed with parentheses

SQL SUBQUERIES

- There is no general syntax; subqueries are regular queries placed inside parenthesis.
- Subqueries can be used in different ways and at different locations inside a query:
- Here is an subquery with the IN operator

SQL SUBQUERY SYNTAX

```
1.  SELECT column-names
2.    FROM table-name1
3.   WHERE value IN (SELECT column-name
4.                     FROM table-name2
5.                    WHERE condition)
```

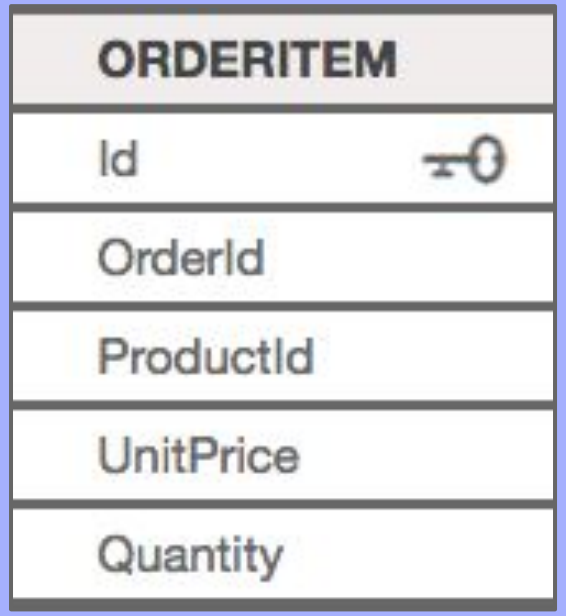
SQL SUBQUERY SYNTAX

Subqueries can also assign column values for each record:

```
SELECT column1 = (SELECT column-name FROM table-name WHERE condition),
       column-names
  FROM table-name
 WEHRE condition
```

SQL SUBQUERY EXAMPLE

Problem: List products with order quantities greater than 100.



SQL SUBQUERY EXAMPLE

Problem: List products with order quantities greater than 100.

```
1.  SELECT ProductName
2.    FROM Product
3.   WHERE Id IN (SELECT ProductId
4.                 FROM OrderItem
5.                WHERE Quantity > 100)
```

SQL SUBQUERY EXAMPLE

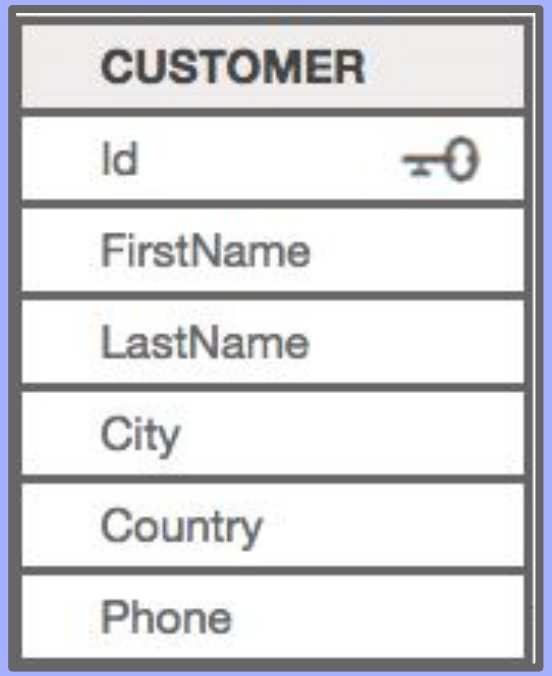
Problem: List products with order quantities greater than 100.

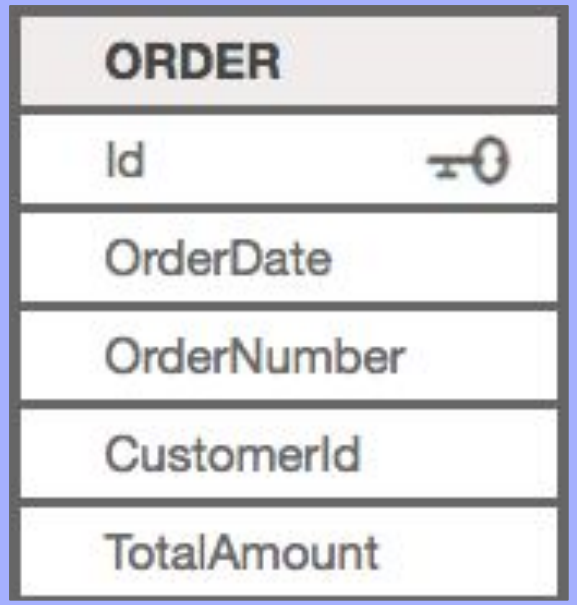
Results: 12 records

PoductName
Guaraná Fantástica
Schoggi Schokolade
Chartreuse verte
Jack's New England Clam Chowder
Rogede sild
Manjimup Dried Apples
Perth Pasties
. . .

SQL SUBQUERY EXAMPLE 2

Problem: List all customers with their total number of orders





SQL SUBQUERY EXAMPLE 2

Problem: List all customers with their total number of orders

This is a **correlated subquery** because the subquery references the enclosing query (i.e. the C.Id in the WHERE clause).

```sql
SELECT FirstName, LastName,
       OrderCount = (SELECT COUNT(O.Id) FROM [Order] O WHERE O.CustomerId = C.Id)
  FROM Customer C
```

SQL SUBQUERY EXAMPLE 2

Problem: List all customers with their total number of orders

Results: 91 records

FirstName	LastName	OrderCount
Maria	Anders	6
Ana	Trujillo	4
Antonio	Moreno	7
Thomas	Hardy	13
Christina	Berglund	18
Hanna	Moos	7
Frédérique	Citeaux	11
Martín	Sommer	3
. . .		

SQL GROUP BY CLAUSE

- The GROUP BY clause groups records into summary rows.
- GROUP BY returns one records for each group.
- GROUP BY typically also involves aggregates: COUNT, MAX, SUM, AVG, etc.
- GROUP BY can group by one or more columns.

SQL GROUP BY SYNTAX

General Syntax

```
1.  SELECT column-names
2.    FROM table-name
3.   WHERE condition
4.   GROUP BY column-names
```

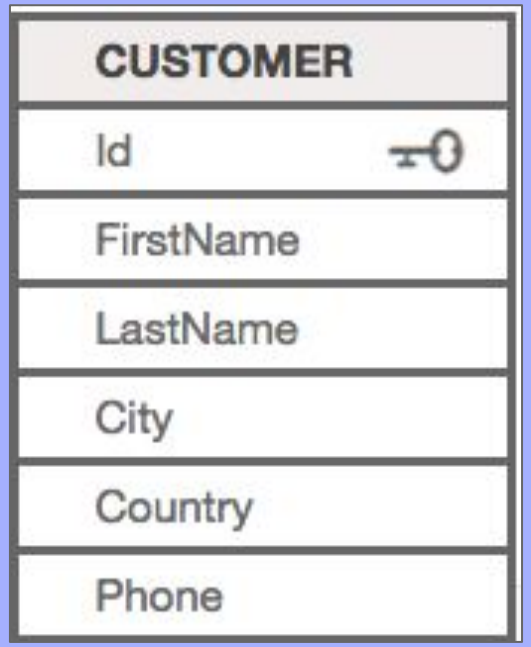
SQL GROUP BY SYNTAX

GENERAL SYNTAX WITH ORDER BY

```
1.   SELECT column-names
2.     FROM table-name
3.    WHERE condition
4.    GROUP BY column-names
5.    ORDER BY column-names
```

SQL GROUP BY EXAMPLE

PROBLEM : List the number of customers in each Country



SQL GROUP BY EXAMPLE

PROBLEM : List the number of customers in each Country

```
1.  SELECT COUNT(Id), Country
2.    FROM Customer
3.   GROUP BY Country
```

SQL GROUP BY EXAMPLE

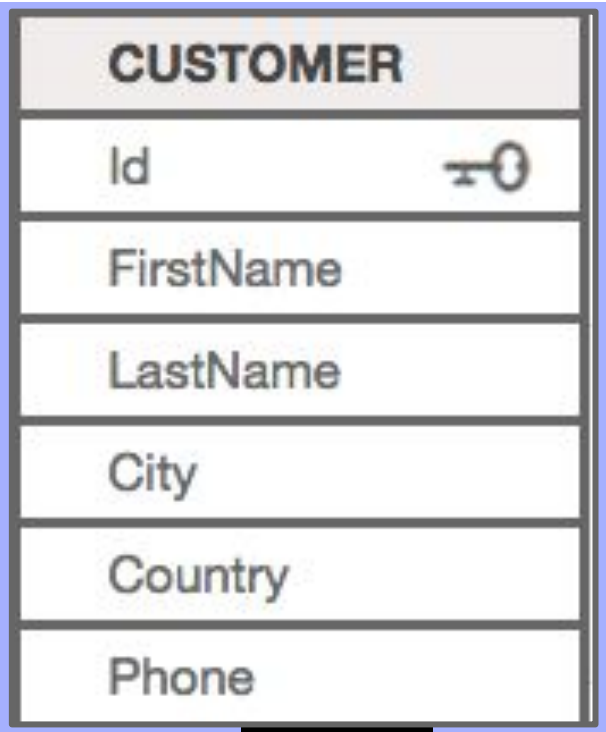
PROBLEM : List the number of customers in each Country

Results: 21 records.

Count	Country
3	Argentina
2	Austria
2	Belgium
9	Brazil
3	Canada
...	

SQL GROUP BY EXAMPLE - 2

PROBLEM : List the number of customers in each Country sorted high to Low



SQL GROUP BY EXAMPLE - 2

PROBLEM : List the number of customers in each Country sorted high to Low

```
1.  SELECT COUNT(Id), Country
2.    FROM Customer
3.   GROUP BY Country
4.   ORDER BY COUNT(Id) DESC
```

SQL GROUP BY EXAMPLE - 2

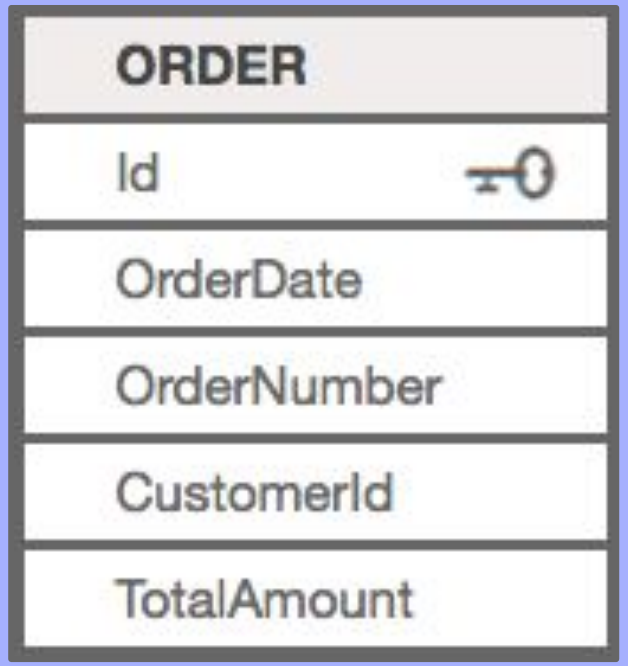
PROBLEM : List the number of customers in each Country sorted high to Low

Results: 21 records.

Count	Country
13	USA
11	France
11	Germany
9	Brazil
7	UK
...	

SQL GROUP BY EXAMPLE - 3

PROBLEM : List the total amount ordered for each customer



SQL GROUP BY EXAMPLE - 3

PROBLEM : List the total amount ordered for each customer .

This query uses a JOIN with Customer to obtain customer names

```
1.  SELECT SUM(O.TotalPrice), C.FirstName
2.    FROM [Order] O JOIN Customer C
3.      ON O.CustomerId = C.Id
4.   GROUP BY C.FirstName, C.LastName
5.   ORDER BY SUM(O.TotalPrice) DESC
```

SQL GROUP BY EXAMPLE - 3

PROBLEM : List the total amount ordered for each customer

Results: 89 records.

Sum	FirstName	LastName
117483.39	Horst	Kloss
115673.39	Jose	Pavarotti
113236.68	Roland	Mendel
57317.39	Patricia	McKenna
52245.90	Paula	Wilson
34101.15	Mario	Pontes
32555.55	Maria	Larsson
. . .		

SQL HAVING CLAUSE

- HAVING filters records that work on summarized GROUP BY results.
- HAVING applies to summarized group records, whereas WHERE applies to individual records.
- Only the groups that meet the HAVING criteria will be returned.
- HAVING requires that a GROUP BY clause is present.
- WHERE and HAVING can be in the same query.

SQL HAVING SYNTAX

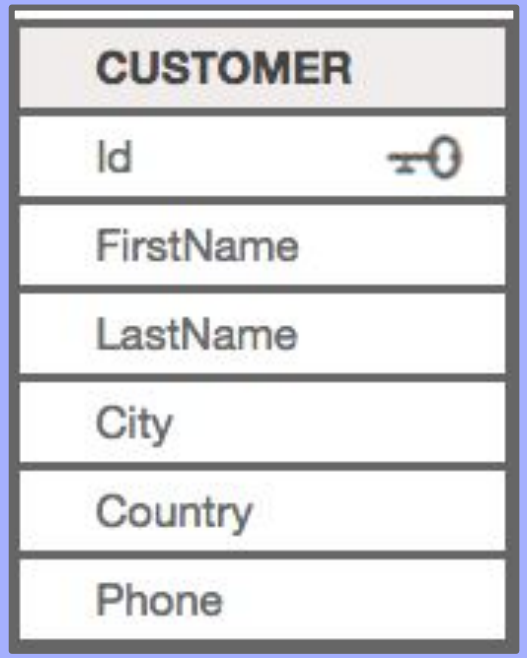
```
1.  SELECT column-names
2.    FROM table-name
3.   WHERE condition
4.   GROUP BY column-names
5.  HAVING condition
```

SQL HAVING SYNTAX

```
1.  SELECT column-names
2.    FROM table-name
3.   WHERE condition
4.   GROUP BY column-names
5.  HAVING condition
6.   ORDER BY column-names
```

SQL HAVING EXAMPLE

Problem: List the number of customers in each country. Only include countries with more than 10 customers.



SQL HAVING EXAMPLE

Problem: List the number of customers in each country. Only include countries with more than 10 customers.

```
1.   SELECT COUNT(Id), Country
2.     FROM Customer
3.    GROUP BY Country
4.   HAVING COUNT(Id) > 10
```

SQL HAVING EXAMPLE

Problem: List the number of customers in each country. Only include countries with more than 10 customers.

Results: 3 records

Count	Country
11	France
11	Germany
13	USA

SQL HAVING EXAMPLE - 2

Problem: List the number of customers in each country, except the USA, sorted high to low. Only include countries with 9 or more customers.

CUSTOMER
Id
FirstName
LastName
City
Country
Phone

```sql
SELECT COUNT(Id), Country
  FROM Customer
 WHERE Country <> 'USA'
 GROUP BY Country
HAVING COUNT(Id) >= 9
 ORDER BY COUNT(Id) DESC
```

SQL HAVING EXAMPLE - 2

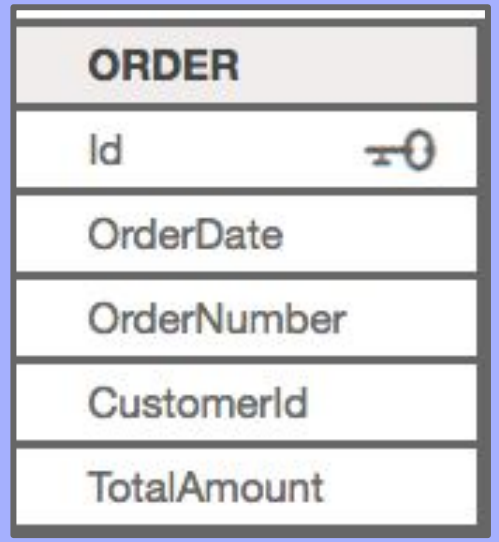
Problem: List the number of customers in each country, except the USA, sorted high to low. Only include countries with 9 or more customers.

Results: 3 records

Count	Country
11	France
11	Germany
9	Brazil

SQL HAVING EXAMPLE - 3

Problem: List all customer with average orders between $1000 and $1200.



SQL HAVING EXAMPLE - 3

Problem: List all customer with average orders between $1000 and $1200.

```
1.  SELECT AVG(TotalAmount), FirstName, LastName
2.    FROM [Order] O JOIN Customer C ON O.CustomerId = C.Id
3.   GROUP BY FirstName, LastName
4.  HAVING AVG(TotalAmount) BETWEEN 1000 AND 1200
```

SQL HAVING EXAMPLE - 3

Problem: List all customer with average orders between $1000 and $1200.

Results: 10 records

Average	FirstName	LastName
1081.215000	Miguel	Angel Paolino
1063.420000	Isabel	de Castro
1008.440000	Alexander	Feuer
1062.038461	Thomas	Hardy
1107.806666	Pirkko	Koskitalo
1174.945454	Janete	Limeira
1073.621428	Antonio	Moreno
1065.385000	Rita	Müller
1183.010000	José	Pedro Freyre
1057.386666	Carine	Schmitt

SQL AGGREGATES

- SELECT MIN returns the minimum value for a column.
- SELECT MAX returns the maximum value for a column.

SQL AGGREGATES

ORDER
Id
OrderDate
OrderNumber
CustomerId
TotalAmount

Problem: Find the last order date in 2013

```
1.  SELECT MAX(OrderDate)
2.    FROM [Order]
3.   WHERE YEAR(OrderDate) = 2013
```

MIN and MAX can also be used with numeric and date types.

Results:

OrderDate
2013-12-31 00:00:00.000

SQL AGGREGATES

- SELECT <u>COUNT</u> returns a count of the number of data values.
- SELECT <u>SUM</u> returns the sum of the data values.
- SELECT <u>AVG</u> returns the average of the data values.

SQL AGGREGATES

Problem: Find the number of customers

```
1.  SELECT COUNT(Id)
2.    FROM Customer
```

Results:

Count
91

CUSTOMER
Id
FirstName
LastName
City
Country
Phone

SQL AGGREGATES

Problem: Compute the total amount sold in 2013

ORDER
Id
OrderDate
OrderNumber
CustomerId
TotalAmount

```
1.  SELECT SUM(TotalAmount)
2.    FROM [Order]
3.   WHERE YEAR(OrderDate) = 2013
```

Results:

Sum
658388.75

SQL AGGREGATES

Problem: Compute the average size of all orders

ORDER
Id
OrderDate
OrderNumber
CustomerId
TotalAmount

```
SELECT AVG(TotalAmount)
  FROM [Order]
```

Results:

Average
1631.877819

TYPES OF FUNCTIONS IN SQL

System Functions: All the built-in functions supported by the SQL are called as System function.

For example, Mathematical Functions, Ranking Functions, String Functions, and so on.

TYPES OF FUNCTIONS IN SQL

SQL Server allows us to create our own functions called as user defined functions in Sql Server. Whenever we need the calculation, we can call it.

For example, if we want to perform some complex calculations then we can place them in separate function, and store it in the database.

TYPES OF FUNCTIONS IN SQL

- **Scalar Function:** It is a function that return single value.
- We can use any SQL data type as the return type except text, image, ntext, cursor, and timestamp.
- The Scalar User defined functions in SQL Server are very useful when you want to return a single value as the resultant.
- For example, total sales, or total investments, total loss, or total expenditure etc.

TYPES OF FUNCTIONS IN SQL

Table Valued Functions: It is a function that return a table.

- **Inline Table valued Functions:** This function returns a table data type based on a single SELECT Statement

- **Multi Statement Table valued Functions:** This function also returns the tabular result set but, unlike the inline table valued function we can use multiple select statements inside the function body.

TYPES OF FUNCTIONS IN SQL

```
-- SQL User Defined Functions - SQL Scalar Function example
USE [SQL Tutorial]
GO
CREATE FUNCTION NoParameters ()
  RETURNS INT
  AS
    BEGIN
       RETURN (SELECT SUM([YearlyIncome]) FROM [MyEmployees Table])
    END
```

TYPES OF FUNCTIONS IN SQL

```
-- SQL User Defined Functions - Inline Functions example
USE [SQL Tutorial]
GO
CREATE FUNCTION CustomerbyDepartment (@profession VARCHAR(50))
  RETURNS TABLE
  AS
     RETURN (
                SELECT  [FirstName]
                ,[LastName]
                ,[Occupation]
                ,[Education]
                ,dept.DepartmentName AS Department
                ,[YearlyIncome] AS Income
                ,[Sales]
                FROM [MyEmployees Table]
                INNER JOIN
                Department AS dept ON
                  Dept.[id] = [MyEmployees Table].DeptID
                WHERE [Occupation] = @profession
                )
```

ORDERED ANALYTICAL FUNCTIONS (WINDOW FUNCTIONs)

```
SELECT item, smonth, sales,
RANK() OVER (PARTITION BY item ORDER BY sales DESC),
AVG(sales) OVER (PARTITION BY item
                 ORDER BY smonth
                 ROWS 3 PRECEDING)
FROM sales_tbl
ORDER BY item, smonth;
```

USING RANK AND AVG

ORDERED ANALYTICAL FUNCTIONS

Item	SMonth	Sales	Rank(Sales)	Moving Avg(Sales)
A	1996-01	110	13	110
A	1996-02	130	10	120
A	1996-03	170	6	137
A	1996-04	210	3	155
A	1996-05	270	1	195
A	1996-06	250	2	225
A	1996-07	190	4	230
A	1996-08	180	5	222
A	1996-09	160	7	195
A	1996-10	140	9	168
A	1996-11	150	8	158
A	1996-12	120	11	142
A	1997-01	120	11	132
B	1996-02	30	5	30
...	...	...	...	...

ORDERED ANALYTICAL FUNCTIONS (WINDOW FUNCTIONs)

```
SELECT store, prodID, sales,
RANK() OVER (PARTITION BY store ORDER BY sales DESC)
FROM sales_tbl
QUALIFY RANK() OVER (PARTITION BY store ORDER BY sales DESC) <=3;
```

USING QUALIFY AND RANK

ORDERED ANALYTICAL FUNCTIONS (WINDOW FUNCTIONs)

Store	ProdID	Sales	Rank(Sales)
1001	D	35000.00	3
1002	A	40000.00	1
1002	C	35000.00	2
1002	D	25000.00	3
1003	D	50000.00	1
1003	A	30000.00	2
1003	C	20000.00	3

Note that every row in the table is returned with the computed value for RANK except those that do not meet the QUALIFY clause (sales rank is less than third within the store).

ORDERED ANALYTICAL FUNCTIONS (WINDOW FUNCTIONs)

```sql
select marks, stuName,
    ROW_NUMBER() over(order by marks desc) as [RowNum],
    RANK() over(order by marks desc) as [Rank],
    DENSE_RANK() over(order by marks desc) as [DenseRank],
    NTILE(3) over(order by marks desc) as [nTile]
from #tempTable
```

ORDERED ANALYTICAL FUNCTIONS (WINDOW FUNCTIONs)

marks	stuName	RowNum	Rank	DenseRank	nTile
90	pooja	1	1	1	1
90	saurabh	2	1	1	1
90	paras	3	1	1	1
80	dinesh	4	4	2	1
80	kanchan	5	4	2	2
80	manoj	6	4	2	2
70	harish	7	7	3	2
70	hema	8	7	3	2
60	nitin	9	9	4	3
50	anita	10	10	5	3
50	kamar	11	10	5	3
50	lalit	12	10	5	3

ORDERED ANALYTICAL FUNCTIONS

```
01. SELECT *, ROW_NUMBER() OVER(ORDER BY EmpName) AS Row_Number
02. FROM Employee
```

Output

SQLQuery2.sql - MC...ER.master (sa (55))* | SQLQuery1.sql - MC...ER.master (sa (51))*

```
SELECT *, ROW_NUMBER() OVER(ORDER BY EmpName) AS Row_Number
FrOM Employee
```

100 %

Results | Messages

	EmpID	EmpName	EmpSalary	Row_Number
1	3	Meths	2000	1
2	6	Meths	9000	2
3	7	Rahul	20000	3
4	4	Rahul	12000	4
5	1	Rohatash	16000	5
6	2	Smith	5000	6
7	5	Smith	13000	7

ORDERED ANALYTICAL FUNCTIONS

(RANK WITH PARTITION BY CLAUSE)

SQLQuery2.sql - MC...ER.master (sa (55))* | SQLQuery1.sql - MC...ER.master (sa (51))*

```
SELECT *, ROW_NUMBER() OVER(Partition by EmpName ORDER BY EmpName) AS Row_Number
FrOM Employee
```

100 %

Results | Messages

	EmpID	EmpName	EmpSalary	Row_Number
1	3	Meths	2000	1
2	6	Meths	9000	2
3	7	Rahul	20000	1
4	4	Rahul	12000	2
5	1	Rohatash	16000	1
6	2	Smith	5000	1
7	5	Smith	13000	2

THANKS!

Any questions?

You can find me at:

ankita.sinha8118@gmail.com