ANALYTIXLABS

MS SQL Server

Module 3

Data Manipulation Language

Disclaimer: This material is protected under copyright act AnalytixLabs ©, 2011-2020. Unauthorized use and/ or duplication of this material or any part of this material including data, in any form without explicit and written permission from AnalytixLabs is strictly prohibited. Any violation of this copyright will attract legal actions

Module 3 : Data Manipulation Language

Lesson 1

Select Statement and its clauses

Lesson 2

Filtering operations

Relational operators

Pattern Identifiers

Lesson 3

Nulls in SQL

Lesson 4

Aggregations in SQL

Lesson 5

Filtering on aggregated data

Lesson 6

Sorting operations

Case statements

Lesson 7

Top clause

Lesson 8

Scalar functions in SQL

Module 3 : Data Manipulation Language

Lesson 9

Introduction to Joins

Lesson 10

Joins explained in Detail

Lesson 11

Unions, Except, Intersect

Lesson 12

Apply the concepts learnt in this Module

Lesson 13

Best practice and key take aways

Module 3 : Data Manipulation Language

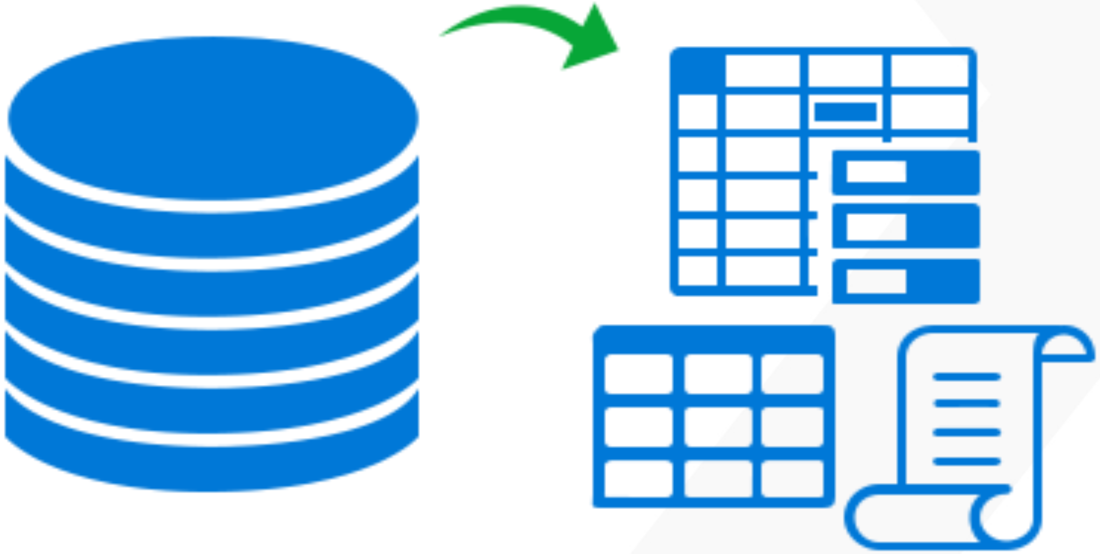
Customer Supplier Database:

- This database houses 5 tables. Customer, Order, OrderItem, Product, Supplier
- The database has information about customers who are situated across the globe and the orders they have placed.
- The order details are related to the products they have ordered.
- The database also has information about suppliers responsible to supply these products either directly to the consumer or through Wholesalers and Retailers.

- Please Note:
  - Entity Relationship Diagram and brief description of each tables can be found in the spreadsheet attached with this module.
  - You need to already have the database and tables set up before you perform analysis as prescribed in each lesson. Please watch recitation videos of this module that explains this set up before you commence the lesson.

Lesson 1 SELECT statement

A SELECT statement retrieves zero or more rows from one or more database tables or database views. In most applications, SELECT is the most commonly used DML statement.



Syntax

```
SELECT
[ ALL | DISTINCT ]
[TOP ( expression ) [PERCENT] [ WITH TIES ] ]
select_list [ INTO new_table ]
[ FROM table_source ]
[ WHERE search_condition ]
[ GROUP BY group_by_expression ]
[ HAVING search_condition ]
[ ORDER BY order_expression [ ASC | DESC ] ]
```

SELECT statement is executed in following order:

1. FROM
2. JOIN
3. WHERE
4. GROUP BY
5. HAVING
6. SELECT
7. ORDER BY

SELECT * vs Select c1,c2…cn

- Use Select * from <table_name> only when you have to display all the columns in a table.
- Using Select * … all the time may lead to unnecessary overhead.
- Avoid Select *.. When your table consists of confidential columns. Instead retrieve data from a specific column.

```
SELECT * from Customer;
Select ID, FistName, LastName from Customer;
```

ALIAS

You can give a table or a column another name by using an alias. This can be a good thing to do if you have very long or complex table names or column names.

An alias name could be anything, but usually it is short.

Syntax for Alias on table names:

SELECT column_name(s) FROM table_name AS alias_name

Select * from Customer as Customertable;

Syntax for Alias on column names:

SELECT column_name AS alias_name FROM table_name

Select ID, UnitPrice*Quantity as TotalAmount from OrderItem;

DISTINCT Keyword

The DISTINCT keyword can be used to return only distinct (different) values.

In a table, some of the columns may contain duplicate values. This is not a problem, however, sometimes you will want to list only the different (distinct) values in a table.

Syntax:

SELECT DISTINCT <column_name/s> FROM <table_name>

Example:

Select **DISTINCT** FirstName FROM Customer;

SELECT **DISTINCT** FirstName, LastName FROM CUSTOMER;

Lesson 2: WHERE Clause

The WHERE clause is used to extract only those records that fulfill a specified criterion.

Syntax:

```
SELECT <column_name/s> FROM <table_name>
WHERE
<column_name> operator value;
```

Example:

```
SELECT * FROM CUSTOMER WHERE FirstName = ‘Maria’;
```

SQL uses single quotes around text values, as shown in the example above!

Example

```
SELECT * FROM CUSTOMER WHERE LastName = ‘Maria’ AND FirstName = ‘Anders’;

SELECT * FROM CUSTOMER WHERE FirstName = ‘Maria’ OR FirstName = ‘John’;

SELECT * FROM CUSTOMER

WHERE

FirstName = ‘Maria’ AND (LastName = ‘Anders’ OR LastName = ‘Larsson’);
```

Boolean Operators

Conditions in the WHERE clause can be combined with AND, OR, and NOT operators.

- AND/OR operators are used to filter records based on more than one condition
  - AND operator displays a record if **all the conditions** separated by AND is TRUE.
  - OR operator displays a record if **any of the conditions** separated by OR is TRUE.
- NOT operator displays a record if the condition(s) is NOT TRUE.

Syntax:

WHERE *condition1* AND *condition2* AND NOT *condition3* ...;

WHERE *condition1* OR *condition2* OR *condition3* ...;

IN Operator

The IN operator allows you to specify multiple values in a WHERE clause.

Syntax:

SELECT column_name(s) FROM table_name WHERE column_name IN (value1,value2,...)

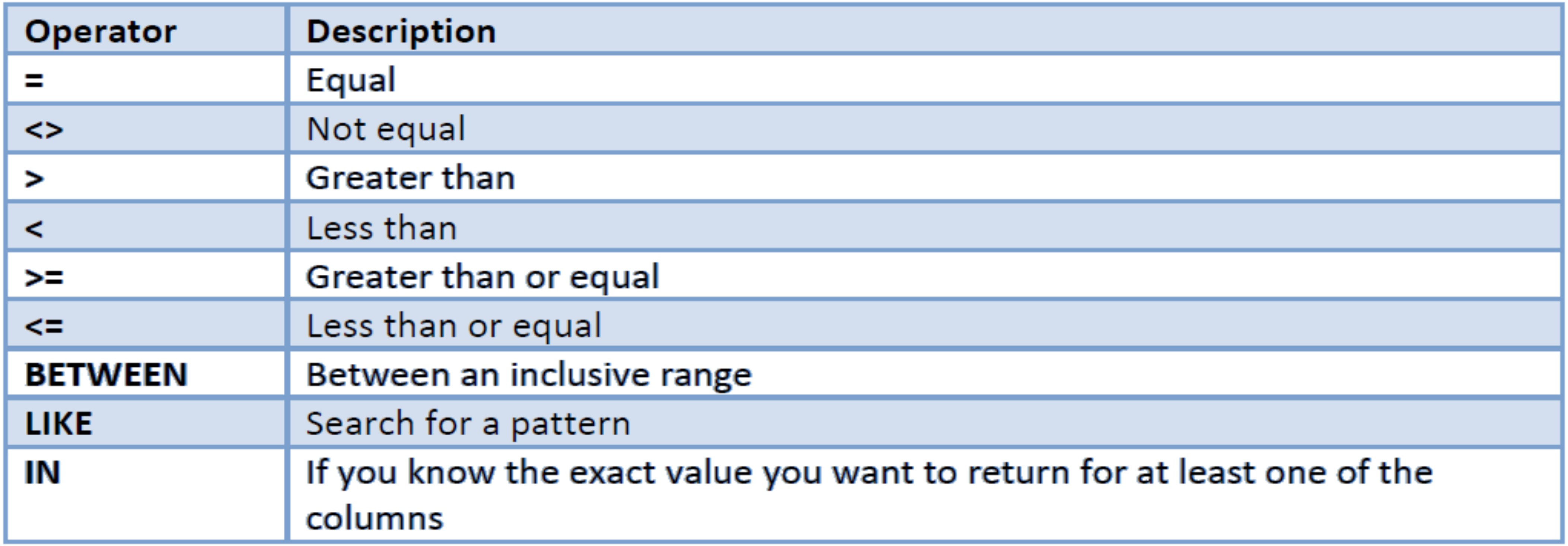
Example:

Select * from Customer Where City IN ('Berlin', 'London');

SELECT * FROM CUSTOMER WHERE ID IN (12, 14);

Comparison Operators

With the WHERE clause, the following operators can be used.



Operator	Description
=	Equal
<>	Not equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
BETWEEN	Between an inclusive range
LIKE	Search for a pattern
IN	If you know the exact value you want to return for at least one of the columns

BETWEEN Operator

The BETWEEN operator selects a range of data between two values. The values can be numbers or dates.

Syntax:

SELECT column_name(s) FROM table_name

WHERE

column_name BETWEEN value1 AND value2

Example:

Select * From OrderItem where UnitPrice **Between** 10 and 20;

LIKE Operator

The LIKE operator is used to search for a specified pattern in a column. The "%" sign can be used to define wildcards (missing letters in the pattern) both before and after the pattern.

Syntax:

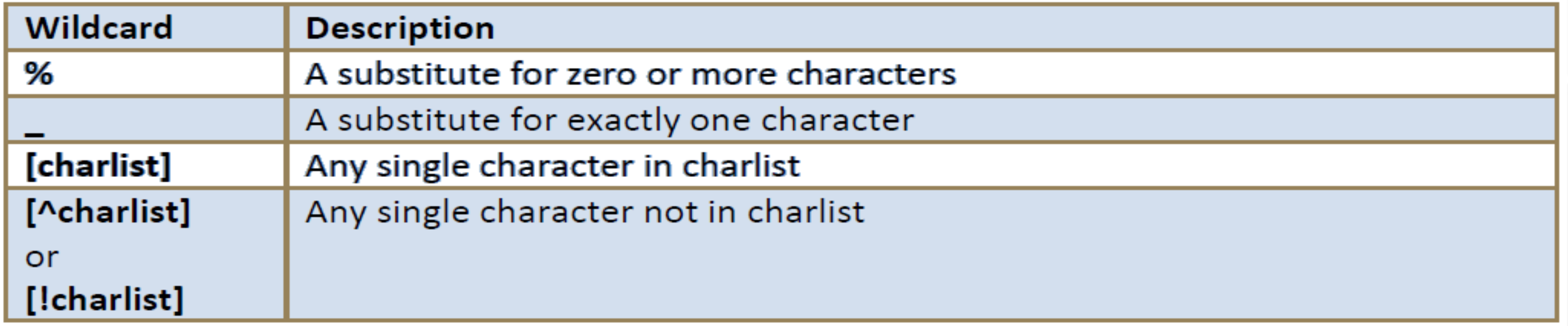
```
SELECT column_name(s) FROM table_name WHERE column_name LIKE pattern
```

Example:

```
SELECT * FROM CUSTOMER WHERE LastName LIKE ‘M%’;
SELECT * FROM CUSTOMER WHERE LastName NOT LIKE ‘%ON%’;
SELECT * FROM CUSTOMER WHERE LastName LIKE ‘%ON%A%’;
```

WILDCARDS

SQL wildcards can substitute for one or more characters when searching for data in a database. SQL wildcards must be used with the SQL LIKE operator.



Wildcard	Description
%	A substitute for zero or more characters
_	A substitute for exactly one character
[charlist]	Any single character in charlist
[^charlist] or [!charlist]	Any single character not in charlist

Example

```
SELECT * FROM CUSTOMER WHERE FirstName LIKE ‘_N_’;

SELECT * FROM CUSTOMER WHERE ID LIKE '[10]%’;

SELECT * FROM CUSTOMER WHERE ID LIKE '[536]%’;
```

DELETE

The DELETE statement is used to delete the records in a table.

Syntax:

```
DELETE FROM table_name
WHERE
some_column = some_value;
```

Notice the WHERE clause in the DELETE syntax. The WHERE clause specifies which record or records should be deleted.

If you omit the WHERE clause, all records will be DELETED from a table!

Continued…

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact.

Syntax:

```
DELETE FROM table_name;
```

Make sure to do this only when you really mean it! You cannot UNDO this statement!

Example

```
DELETE FROM Product_Copy
WHERE
Id = 10;
```

Always include the WHERE clause when using the DELETE command!

Lesson 3: SQL NULL Values

A field with a NULL value is a field with no value.

- If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.
- NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation.
- Null is unknown. So anything cannot be equated to a null while performing filtering operations. Hence we use ISNULL

Test for NULL Values

It is not possible to test for NULL values with comparison operators, such as =, <, or <>. We will have to use the IS NULL and IS NOT NULL operators instead.

Syntax:

```
SELECT column_name(s) FROM table_name WHERE column_name IS [NOT] NULL;
```

Example:

```
SELECT * FROM Supplier WHERE Fax IS NULL;
SELECT * FROM Supplier WHERE Fax IS NOT NULL;
```

SQL ISNULL, NULLIF, Coalesce

- **ISNULL**(expression, value) : returns value if expression is Null. If expression is not null, expression is returned. Expression can be a column names as well.
  - Select isnull('Welcome to AnalytixLabs', 'SQL') as remarks; -- returns *Welcome to AnalytixLabs*
  - Select isnull(Null, 100) as remarks; --returns *100*
  - Select isnull(Null, ID) as remarks from Customer; --returns ID column from Customer Table
- **NULLIF**(exp1, exp2) : returns NULL if two expressions are equal, otherwise it returns the first expression. Expression can be a column names as well.
  - Select Nullif('Available', 'Available') as remarks; -- returns *Null*
  - Select Nullif(25, 100) as remarks; --returns *25*
- **Coalesce**(val1, val2,…..,val_n) : returns first non null value in a list of values considered. Values can be a column names as well.
  - Select Coalesce('Null', Null', 'Welcome to AnalytixLabs', 'SQL') – returns *Welcome to AnalytixLabs*

Lesson 4: GROUP BY Clause

The GROUP BY statement is often used with aggregate functions (COUNT, MAX, MIN, SUM, AVG) to group the result-set by one or more columns. Columns which are not a part of aggregation in a select statement should be a part of "group by" statement

Syntax:

SELECT

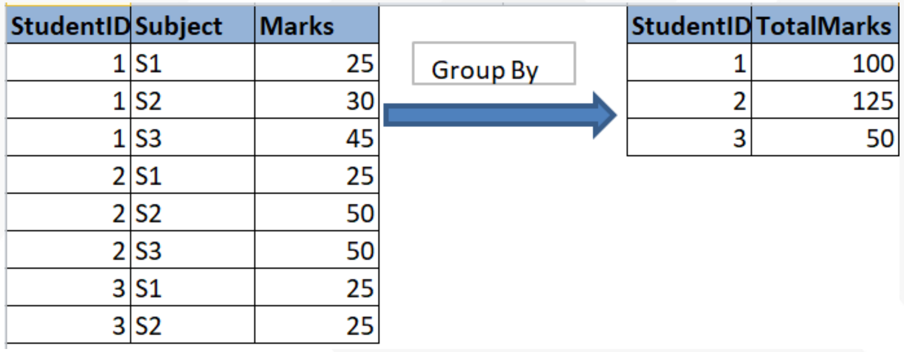
column_name(s), aggregate_function(column_name) AS AggCol

FROM

table_name

GROUP BY

column_name(s);



StudentID	Subject	Marks
1	S1	25
1	S2	30
1	S3	45
2	S1	25
2	S2	50
2	S3	50
3	S1	25
3	S2	25

Group By

StudentID	TotalMarks
1	100
2	125
3	50

Lesson 5: HAVING Clause

HAVING clause is used to extract only those records that fulfill a specified criterion when condition should be applied on aggregate/derived fields.

Syntax:

```
SELECT
column_name(s), AGG(column_name) AS AggCol
FROM table_name
GROUP BY column_name(s)
HAVING
AGG(column_name) condition/s
```

Aggregations cannot be used in Where clause. Hence Having clause is used to perform filtering on aggregations

Lesson 6 : ORDER BY Clause

Used when you want the data to appear in a specific order.

- If you use the “order by” keyword, the default order is ascending (“asc”). If you want the order to be opposite, i.e., descending, then you need to use the “desc” keyword.

- ORDER BY clause should be the last statement in SQL SELECT query.

- Three variants of Sorting
  - Sorting with One field
  - Hierarchical Sorting
  - Custom Sorting

Example

```
SELECT * FROM CUSTOMER ORDER BY LastName; -- Sorting with one field


--Sort in descending order:
SELECT * FROM CUSTOMER ORDER BY LastName desc;

--You may sort by several columns:
SELECT * FROM CUSTOMER ORDER BY FirstName, LastName; -- Hierarchical sorting


--You may also sort by columns positions (will sort data by FirstName and LastName):
SELECT * FROM CUSTOMER ORDER BY 2;
```

CASE statement

CASE statement is used to check for conditions. Using CASE in SELECT can help to get conditional output or to generate data in multiple columns based on the conditions.

Syntax:

```
CASE case_value
   WHEN when_value THEN statement_list
   [WHEN when_value THEN statement_list] ...
   [ELSE statement_list]
END
```

Example of Custom Sorting

```
Select Id, FirstName, LastName,Country from
Customer
Order by case Country
when 'Germany' then 1
when 'UK' Then 2
when 'Mexico' then 3
else 4
end
```

Lesson 7: TOP Clause

The TOP clause is used to specify the number of records to return.

The TOP clause can be very useful on large tables with thousands of records. Returning a large number of records can impact on performance.

Syntax:

SELECT TOP number | percent column_name(s) FROM table_name

Example:

SELECT TOP 1 * FROM CUSTOMER;

SELECT TOP 60 PERCENT * FROM CUSTOMER;

Offset fetch next

- *Offset..Fetch Next* command returns a specific range of rows by offsetting a certain number of rows. This should always be preceded by an Order by Clause.
- This is helpful when you do not want to fetch bottom or top N rows but intermediate rows of data

Syntax:

```
Select Column_name(s) from Table_name
Order by Column_name(s)
OffSet N rows
 Fetch Next M rows only;
Fetch Next M rows only;
```

Example:

```
Select ID, ProductName, Unitprice
from Product
 Order by ID
 Offset 10 rows
Fetch Next 10 rows only;
```

Lesson 8: Scalar Functions | Number Functions

The following scalar functions perform an operation on a numeric input value and return a numeric value

ABS	COUNT
CEILING	MIN
FLOOR	MAX
RAND	SUM
ROUND	AVG
SQUARE	ISNUMERIC
SQRT	

Text/String Functions

The following functions perform an operation on a string input value and return a string or numeric value

ASCII	CONCAT	REPLICATE
CHAR	LTRIM	SPACE
CHARINDEX	RTRIM	REVERSE
REPLACE	LEFT	
STUFF	RIGHT	
LOWER	SUBSTRING	
UPPER	LEN	

Date Functions

The following functions perform an operation on a date, integer or string input value and return a date, string or integer value

GETDATE	CURRENT_TIMESTAMP
YEAR	DATENAME
DAY	ISDATE
MONTH	
DATEPART	
DATEDIFF	
DATEADD	

Conversion Functions

The following functions converts the fields from one datatype to another.

CAST(*expression* AS *data_type(length)*)

CONVERT(*data_type(length), expression, style*)

Follow the link to get all style types:

https://docs.microsoft.com/en-us/sql/t-sql/functions/cast-and-convert-transact-sql?view=sql-server-2017

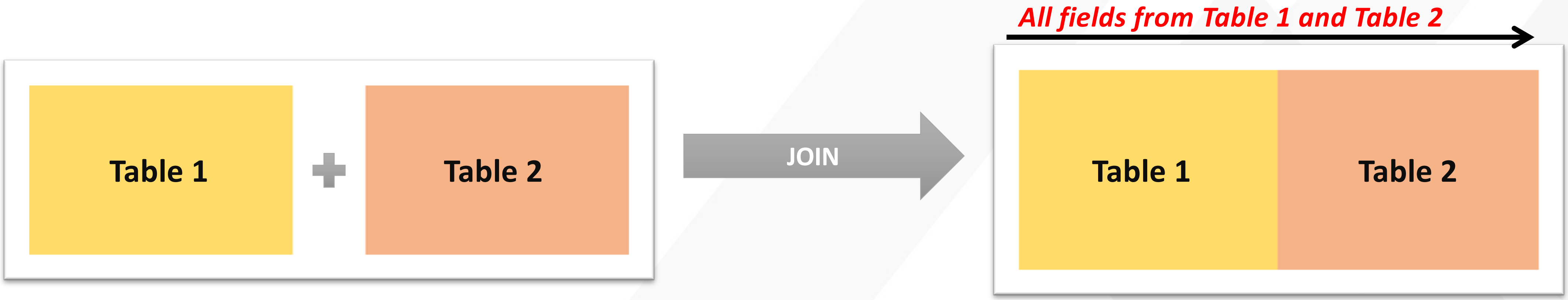
Cast and Convert Example

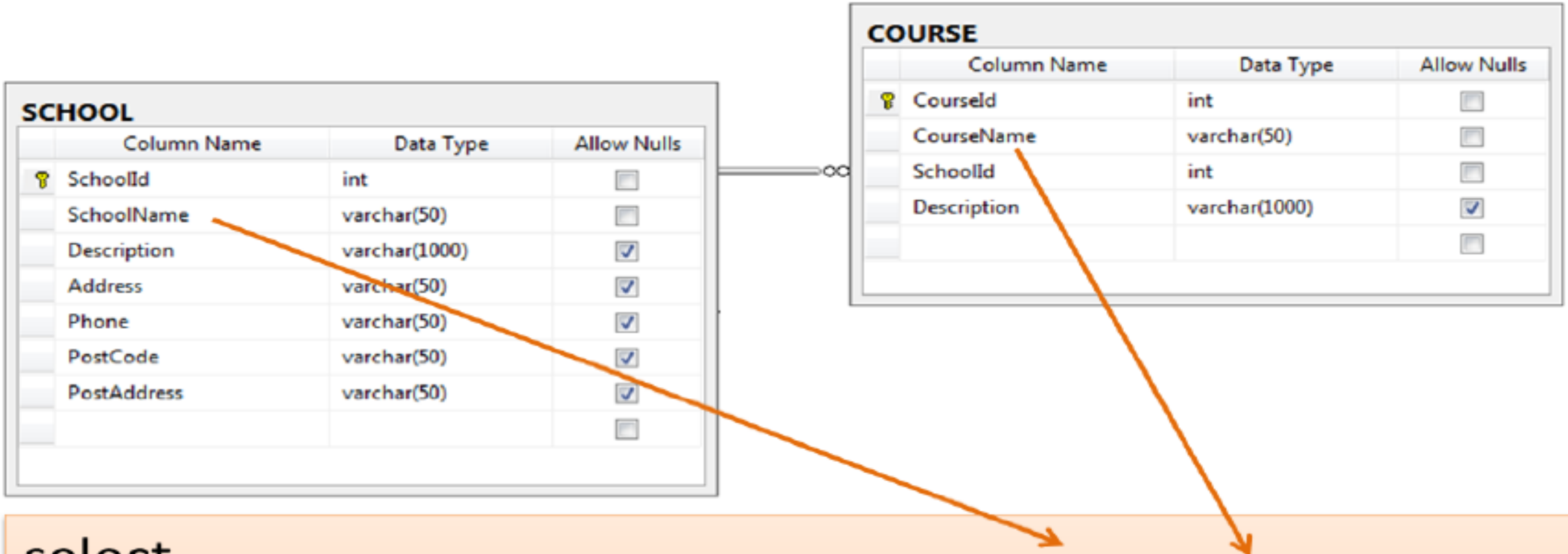
```
Select OrderNumber, CustomerId, TotalAmount,
‘The Amount of OrderNumber '+CAST(OrderNumber as Varchar(100))+ ' is ' +
CAST(TotalAmount as varchar(100)) as description from [Order];

Select OrderDate, CONVERT(varchar, OrderDate, 107) as Format107
from [Order]

Select OrderDate, CONVERT(varchar, OrderDate, 100) as Format100
from [Order]
```

Lesson 9: JOINS

SQL joins are used to query data from two or more tables, based on a relationship between certain columns in these tables. In Joins we combine rows from two or more tables, based on a related column between them.

- Resultant output after the join will have all the columns from participating tables.
- Number of records in output will depend on join type.





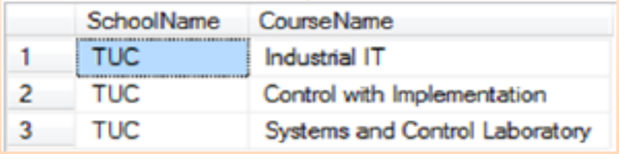
SCHOOL

	Column Name	Data Type	Allow Nulls
🔑	SchoolId	int	☐
	SchoolName	varchar(50)	☐
	Description	varchar(1000)	☑
	Address	varchar(50)	☑
	Phone	varchar(50)	☑
	PostCode	varchar(50)	☑
	PostAddress	varchar(50)	☑
			☐

COURSE

	Column Name	Data Type	Allow Nulls
🔑	CourseId	int	☐
	CourseName	varchar(50)	☐
	SchoolId	int	☐
	Description	varchar(1000)	☑
			☐

```
select
SchoolName,
CourseName
from
SCHOOL
inner join COURSE on SCHOOL.SchoolId = COURSE.SchoolId
```



	SchoolName	CourseName
1	TUC	Industrial IT
2	TUC	Control with Implementation
3	TUC	Systems and Control Laboratory

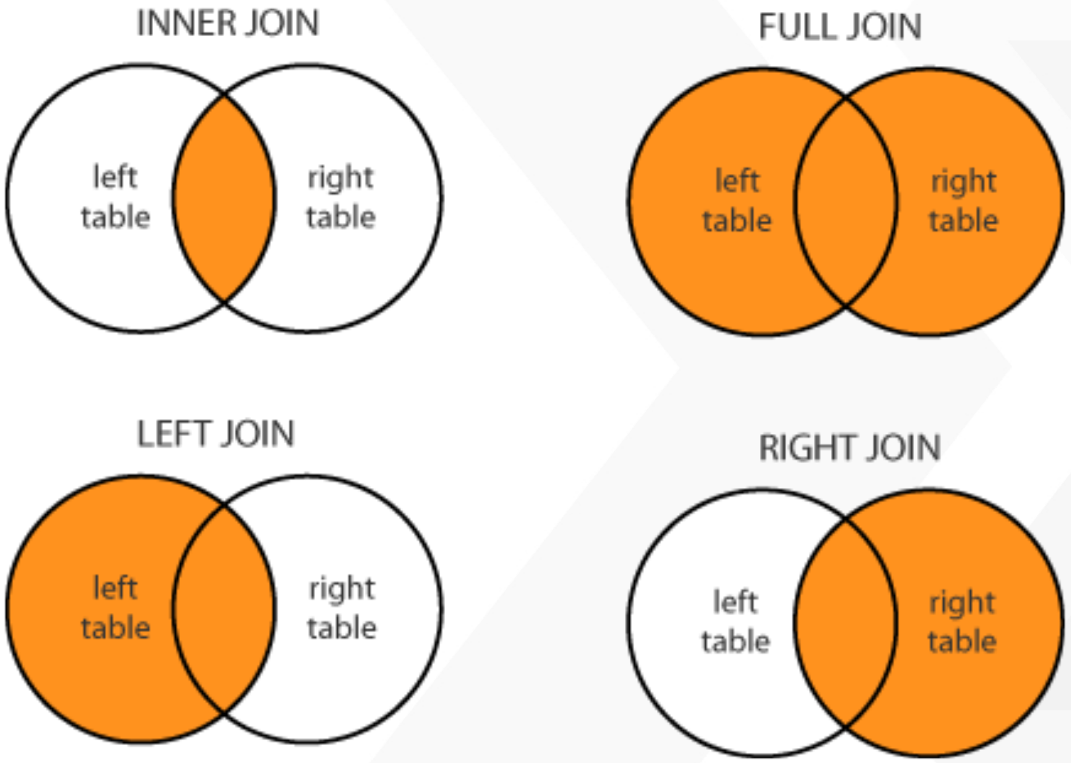
You link Primary Keys and Foreign Keys together

Join Types

We can join two or more tables using following join types:

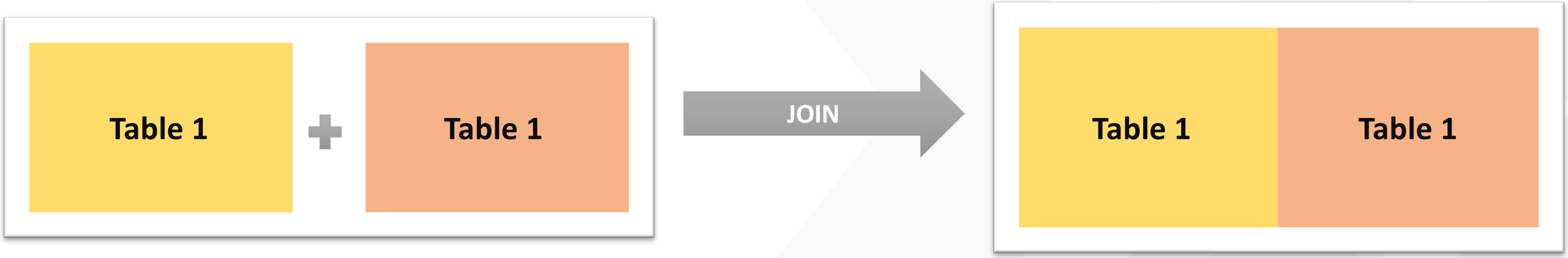
- Cross Join : Returns Cartesian product of number of rows of both tables. If table A has “m” rows and Table B has “n” rows, resultant table will have “m*n” rows
- INNER JOIN: Return rows when there is at least one match in both tables.
- LEFT JOIN: Return all rows from the left table, even if there are no matches in the right table.
- RIGHT JOIN: Return all rows from the right table, even if there are no matches in the left table.
- FULL JOIN: Return rows when there is a match in one of the tables.

Continued..


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

SELF JOIN

In SELF JOIN a table is joined to itself. That is, each row of the table is joined with itself and all other rows depending on some conditions. In other words we can say that it is a join between two copies of the same table.



Same table is joined with itself.

Example:

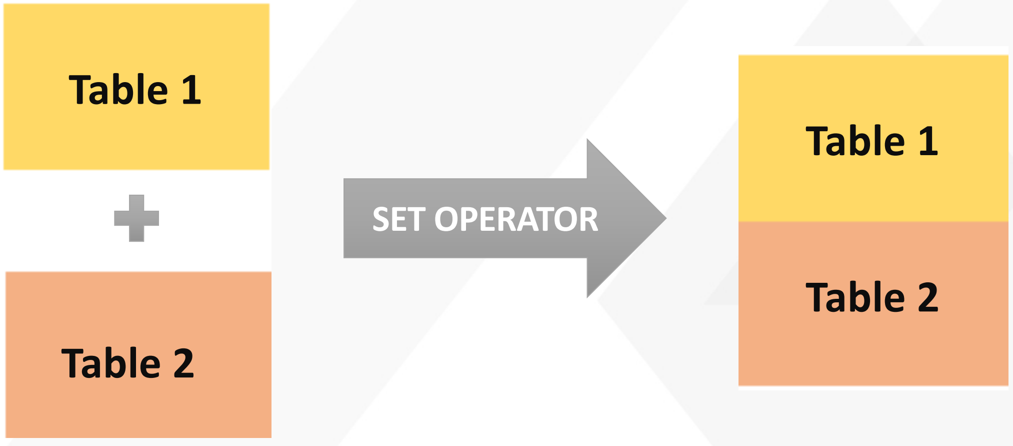
Get manager names of all the employees in the organization.

Lesson 11: SET OPERATORS

Set operators allow to combine the results of multiple separate SELECT queries into a single result set. Useful when data combining has to be done vertically.

Each SELECT statement within **SET Operators** must have the same number of fields with similar data types. SQL supports following SET operations.

- UNION ALL
- UNION
- INTERSECT
- EXCEPT (MINUS)



UNION ALL

UNION ALL is used to combine output of two SELECT statements. Output dataset will retain all records from both tables.

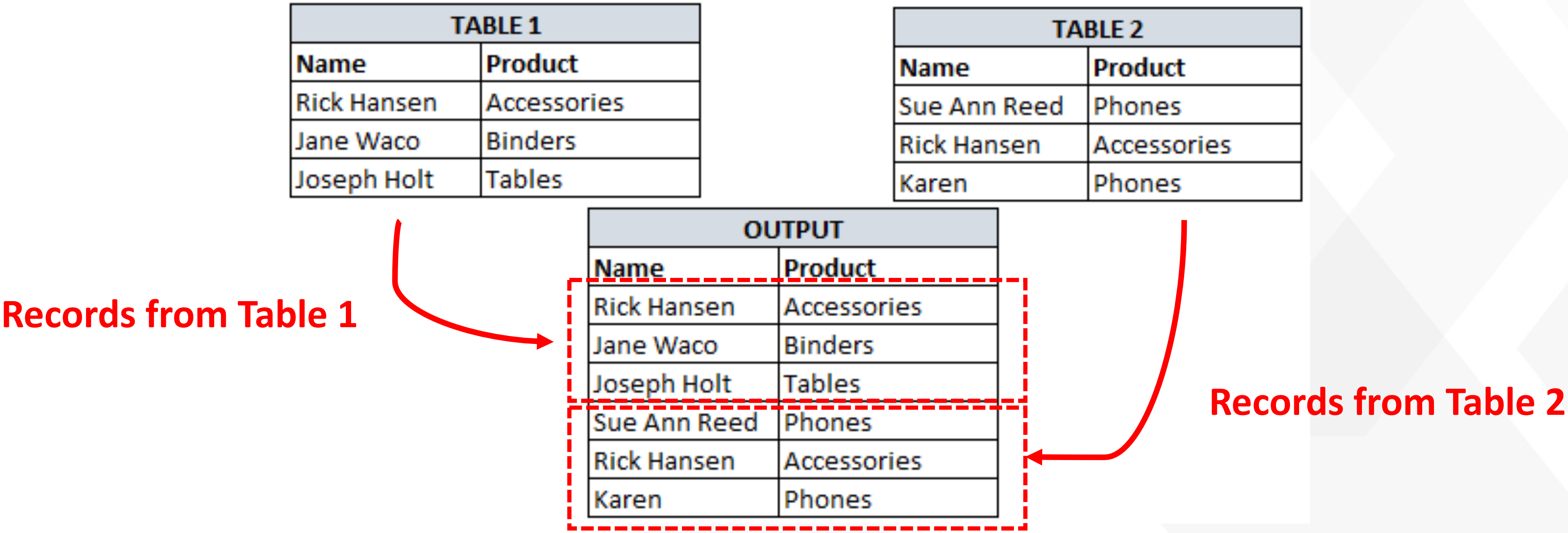


TABLE 1	
Name	**Product**
Rick Hansen	Accessories
Jane Waco	Binders
Joseph Holt	Tables

TABLE 2	
Name	**Product**
Sue Ann Reed	Phones
Rick Hansen	Accessories
Karen	Phones

Records from Table 1

OUTPUT	
Name	**Product**
Rick Hansen	Accessories
Jane Waco	Binders
Joseph Holt	Tables
Sue Ann Reed	Phones
Rick Hansen	Accessories
Karen	Phones

Records from Table 2

UNION

UNION is used to combine output of two SELECT statements. Output dataset will retain all records from both tables; however it will remove the duplicate/common records from the output dataset and will sort the data in ascending order.

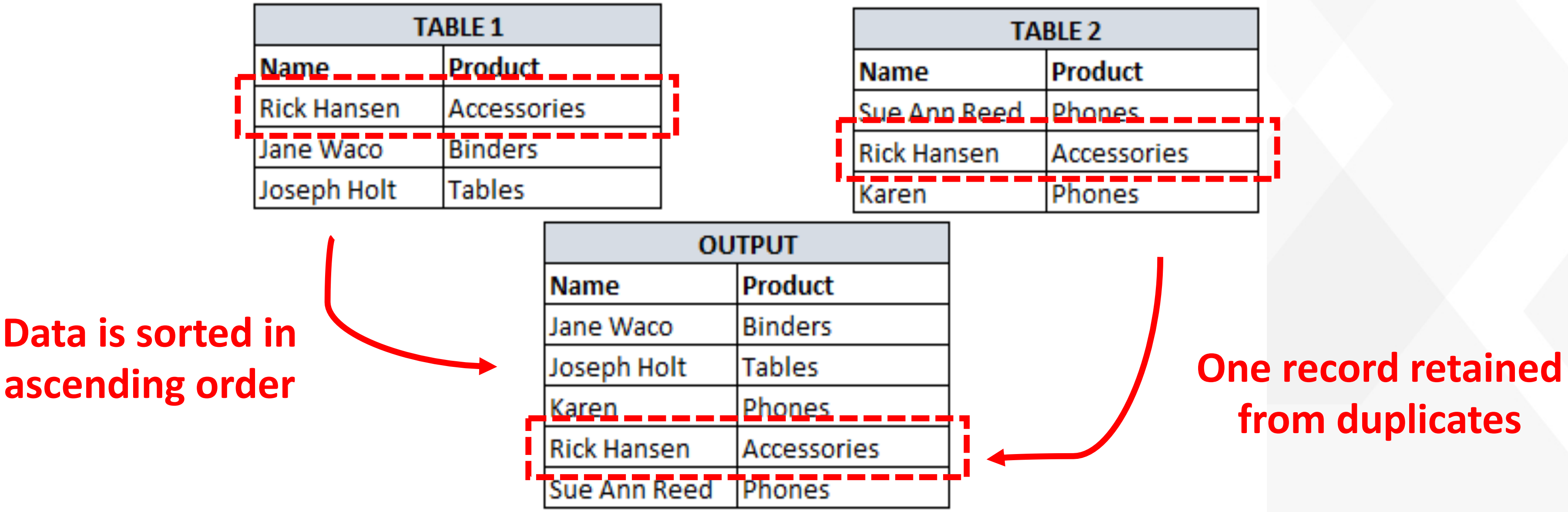


TABLE 1

Name	Product
Rick Hansen	Accessories
Jane Waco	Binders
Joseph Holt	Tables

TABLE 2

Name	Product
Sue Ann Reed	Phones
Rick Hansen	Accessories
Karen	Phones

OUTPUT

Name	Product
Jane Waco	Binders
Joseph Holt	Tables
Karen	Phones
Rick Hansen	Accessories
Sue Ann Reed	Phones

Data is sorted in ascending order

One record retained from duplicates

INTERSECT

Intersect operation is used to combine output of two SELECT statements; only common records are returned from both SELECT statements in the output.

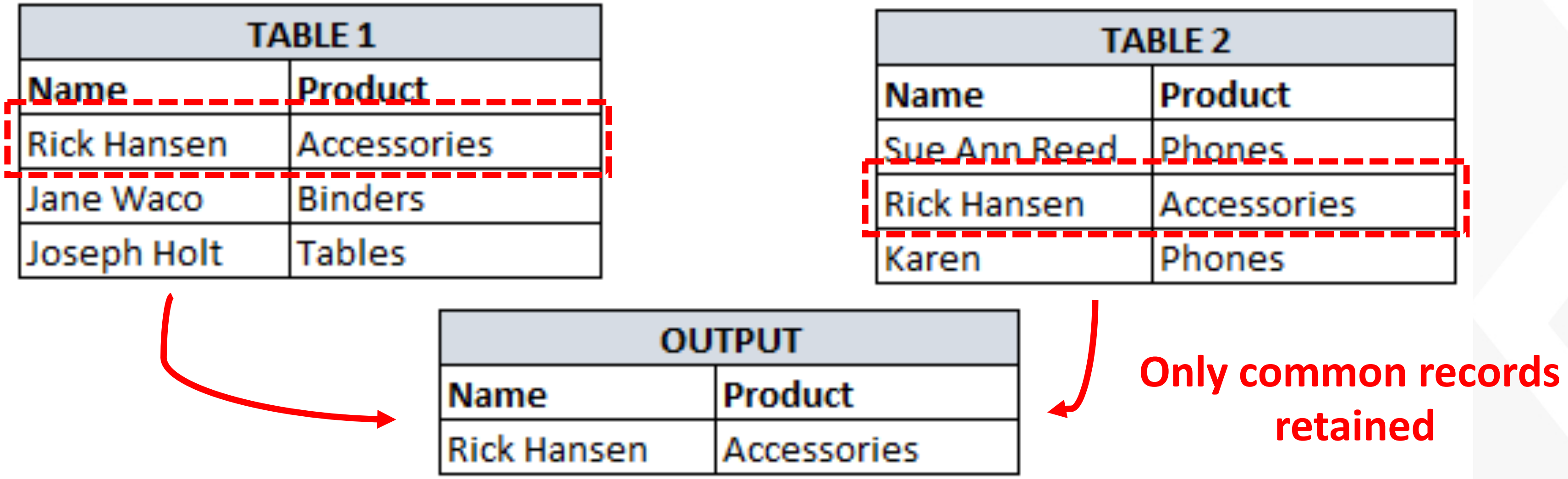


TABLE 1	
Name	Product
Rick Hansen	Accessories
Jane Waco	Binders
Joseph Holt	Tables

TABLE 2	
Name	Product
Sue Ann Reed	Phones
Rick Hansen	Accessories
Karen	Phones

OUTPUT	
Name	Product
Rick Hansen	Accessories

Only common records retained

EXCEPT

Except operation is used to combine output of two SELECT statements; only those records from first SELECT statements are retained which don’t exist in second dataset.

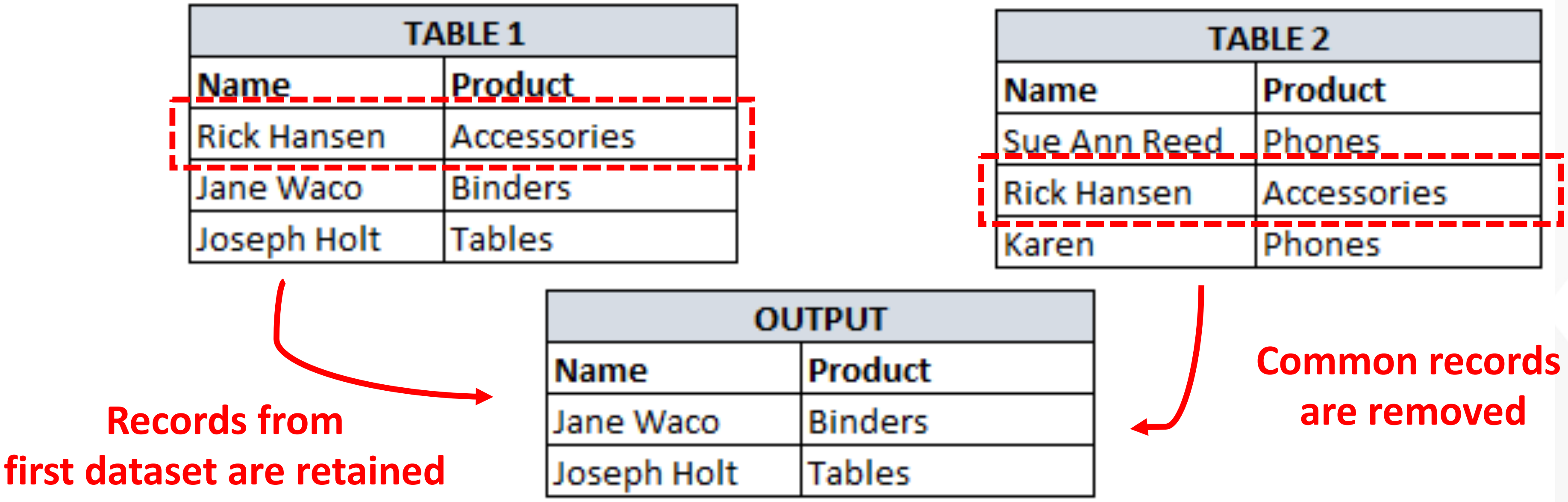


TABLE 1

Name	Product
Rick Hansen	Accessories
Jane Waco	Binders
Joseph Holt	Tables

TABLE 2

Name	Product
Sue Ann Reed	Phones
Rick Hansen	Accessories
Karen	Phones

OUTPUT

Name	Product
Jane Waco	Binders
Joseph Holt	Tables

Records from
first dataset are retained

Common records
are removed

Join vs Set Based operators

- Joins combines two tables based on a linking field and gives a resultant table. Set based operators combines the output of two select statements.

- Two or more similar Select statement which has joins can be combined using Set base operators.

- In Set based operators, the resultant table will have the same column names as that of the first select statement.

Lesson 12: More Examples solved..

1. You have been told to identify customers whose average order amount is between $1000 and $1200.

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

More Examples solved..

2. Show all orders, sorted by total amount, the largest first, within each year, oldest first.

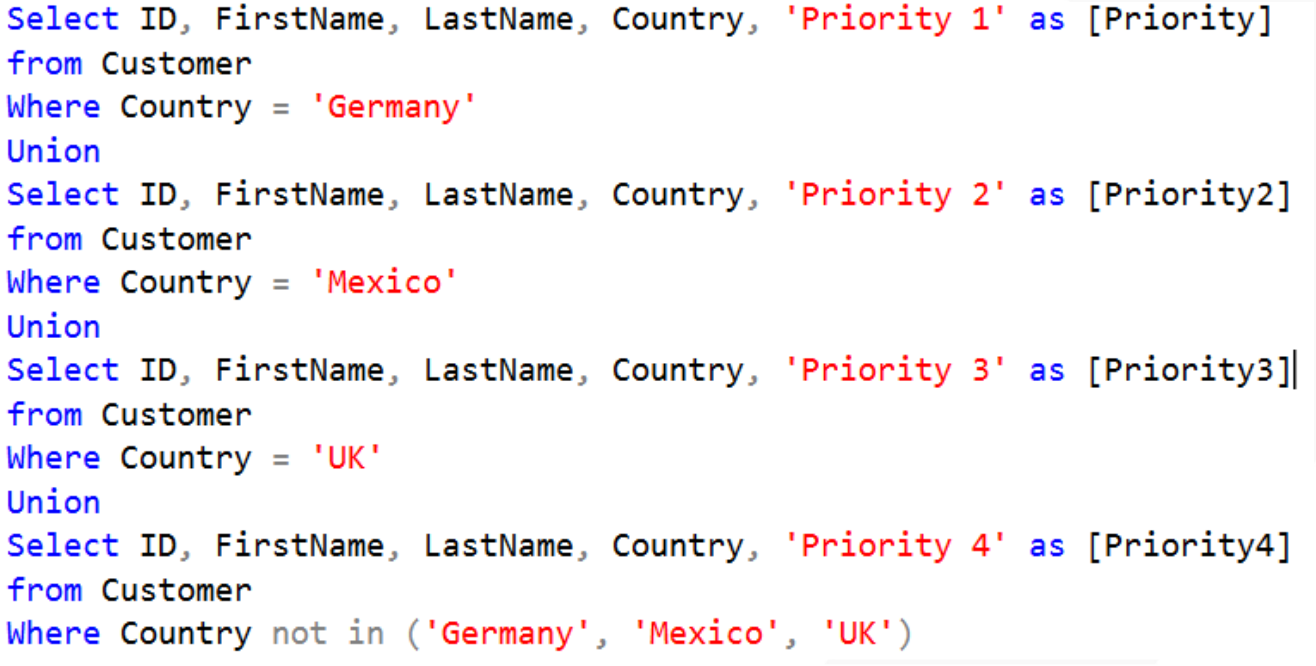
```
SELECT Id, OrderDate, CustomerId, TotalAmount
FROM [Order]
ORDER BY YEAR(OrderDate) ASC, TotalAmount DESC;
```

3. List all suppliers that do have a fax number.

```
SELECT Id, CompanyName, Phone, Fax
FROM Supplier
WHERE Fax IS NULL;
```

More Examples solved..

4. Customers from Germany, Mexico and UK are Priority customers 1, 2 and 3 respectively. Customers from other countries fall in Priority 4 bracket. Device a set based query to give this information.



```
Select ID, FirstName, LastName, Country, 'Priority 1' as [Priority]
from Customer
Where Country = 'Germany'
Union
Select ID, FirstName, LastName, Country, 'Priority 2' as [Priority2]
from Customer
Where Country = 'Mexico'
Union
Select ID, FirstName, LastName, Country, 'Priority 3' as [Priority3]
from Customer
Where Country = 'UK'
Union
Select ID, FirstName, LastName, Country, 'Priority 4' as [Priority4]
from Customer
Where Country not in ('Germany', 'Mexico', 'UK')
```

Let’s tie up everything!

5. Get top 50 customers’ details who have ordered more than 50 quantities in 2013.

```
Select Top 50 C.ID, C.FirstName, O.OrderDate, Sum(Quantity) as TotalQuantity
From Customer C
Inner Join [Order] O on
C.id = O.CustomerId
Inner Join OrderItem OI on
O.id = OI.OrderId
where YEAR(OrderDate) = 2013
Group by C.Id, C.FirstName, O.OrderDate
Having Sum(Quantity) > 50
Order by Sum(Quantity) desc
```

Lesson 13: Best Practices – for faster query execution

- Avoid unnecessary columns in SELECT clause [SELECT * not recommended]
- DISTINCT and UNION should be used only if it is necessary. Use UNION ALL when ever possible.
- Sort (ORDER BY) is one of the costliest operation; use only if required.
- Instead of outer joins; use inner join where ever possible.
- Avoid using functions in left side of the comparison operators.
- Prefer using Indexed fields in JOIN, WHERE, ORDER BY and GROUP BY clauses.
- Avoid using wildcard (%) when ever possible. Ensure you are NOT using them at least in the beginning of the comparison values [e.g. field LIKE ‘%value’].
- Do Cross Joins only on required segments of data; Remember doing cross join of two tables with million rows yields a table of trillion rows! May slow down execution time.

Key Take aways

- Select statement displays the information in a table.
- Where clause performs Filtering operation. It can be used with variety of operators including nulls to get meaningful insights into the data.
- Data aggregation is performed using Group by clause.
- Having clause performs filtering on aggregations
- Sorting can be Normal, hierarchical or custom sorting.
- Top Clause/Offset..Fetch Next keywords are used to return desired number of rows.
- Scalar functions return a single valued output. Can be Numeric/String/Date functions.
- Joins combine two tables with or without using a linking field.
- Set Operators combined two or more select statements.

God is Unknown and so are Nulls in SQL !! Thank you!

Visit Us

Visit us : https://www.analytixlabs.co.in/

Email us: training@analytixlabs.co.in

Call us: (+91) 9555219007

Gurgaon Address:

GF 382, Sector 29,
Adjoining IFFCO Chowk Metro
Station (Gate 2),
Next to Vasan Eye Care Hospital,
Gurgaon, Haryana 122001,
India

Bengaluru Address:

Bldg 41, First floor,
14th Main Road, Near BDA
complex,
Sector 7, HSR Layout
Bengaluru - 560102
Landmark: Max store