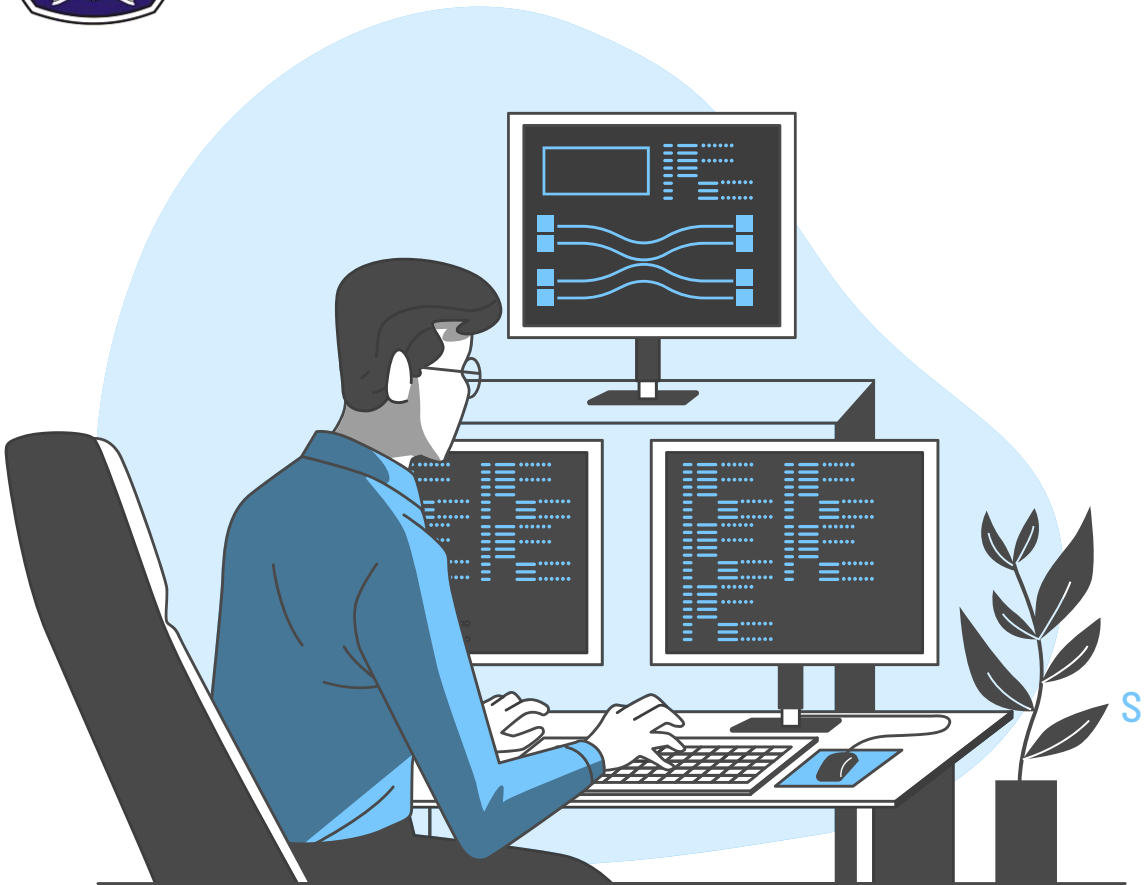
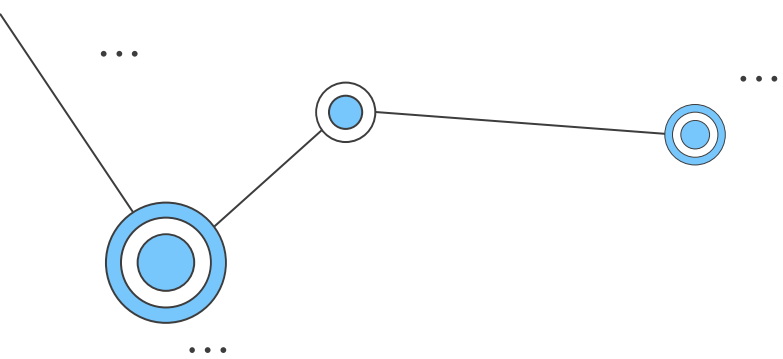


ADVANCED DATABASE

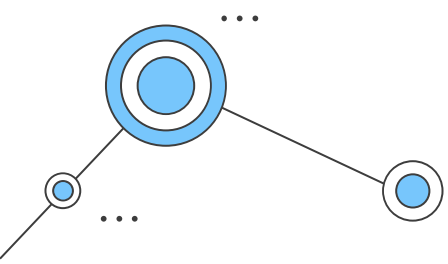
Week 2

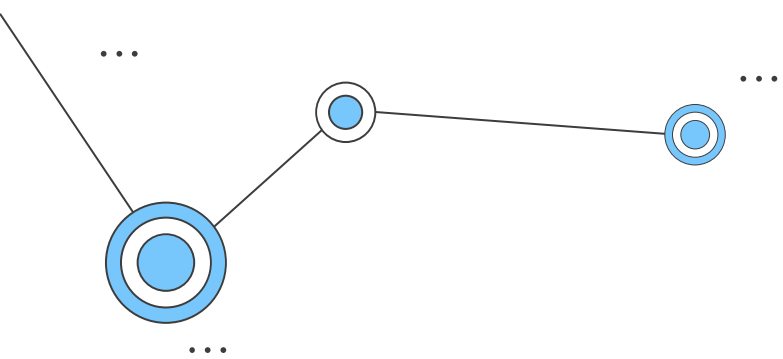
SELECT, JOIN, SORTING, FILTERING



OUTLINE

- T-SQL
- Comment
- SELECT, DISTINCT
- Use of ALIAS
- Draft CASE expression
- JOIN
- Sorting
- Filtering
- Top and OFFSET-FETCH





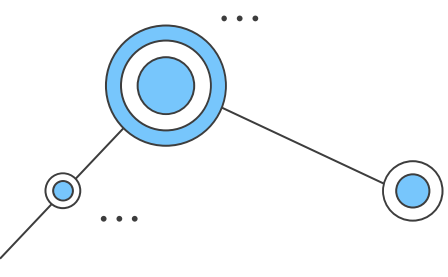
WHAT IS T-SQL?

- **SQL**

Structured Query Language; language used For access and perform manipulation a database.

- **Transact-SQL (abbreviated as T-SQL)**

SQL database language issued by Microsoft and Sybase companies . Microsoft SQL Server only recognize This T-SQL type SQL language .



T-SQL TYPES

- DDL (Data Definition Language)

Used For define database structure , database and tables .

order basic : **CREATE, ALTER, DROP**

- DML (Data Manipulation Language)

Used For do manipulation or data processing in table .

Order basics : **SELECT, INSERT, UPDATE, DELETE**

T-SQL TYPES (1)

- DCL (Data Control Language)

Used For arrangement right good user access to servers, databases and table .

Order basic : **GRANT, REVOKE**

- TCL (Transaction Control Language)

Used For regulate and control T-SQL transactions . Guarantee transaction succeed done and not done violate database

Order basic : **BEGIN TRAN, COMMIT TRAN, ROLLBACK**

COMMENTS ON T-SQL

- Comments in T-SQL use the “--” symbol for single-line comments.
- /* … */ sign for comments on more than one line
- For example:

```
-- Komentar satu baris

/* tanda awal komentar multi baris
    komentar - komentar
    tanda akhir komentar multi baris */
```

T-SQL

SELECT

SELECT

- Used For display data from table a database
- For displaying required data :
  - ✓ What just the column you want displayed
  - ✓ The name of the table to be displayed
  - ✓ Condition For display data

USE OF SELECT COMMAND

Projection

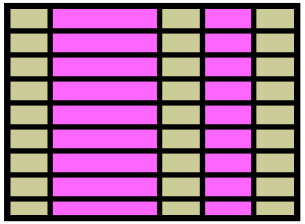


Table 1

Selection

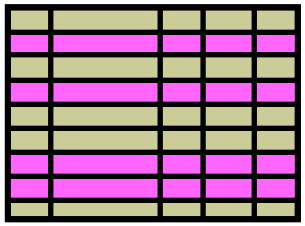


Table 1

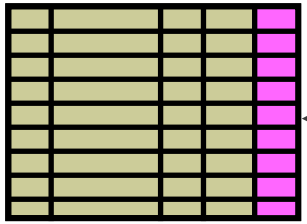


Table 1

Join

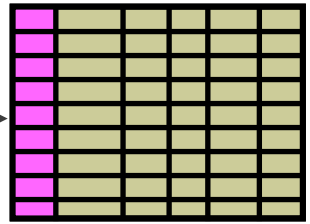


Table 2

SELECT ALL DATA IN TABLE

- Syntax:
  SELECT * FROM [table_name];

```
SELECT *
FROM departments;
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

8 rows selected.

SELECT SPECIFIC DATA IN TABLE

- Syntax:

  SELECT [column_name], [column_name]

  FROM [table_name]

```
SELECT department_id, location_id
FROM departments;
```

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

8 rows selected.

SELECT BASED ON CERTAIN CONDITIONS

- The WHERE command can be used to limit the rows to be displayed.
- Syntax:

  SELECT [column_name], [column_name]
  FROM [table_name]
  WHERE [condition];

```
SELECT last_name, salary
FROM employees
WHERE salary <= 3000 ;
```

LAST_NAME	SALARY
Matos	2600
Vargas	2500

USING ARITHMETIC OPERATORS

```
SELECT last_name , salary, salary + 300
FROM employees;
```

LAST_NAME	SALARY	SALARY+300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Hunold	9000	9300
Ernst	6000	6300

...

20 rows selected.

DISTINCT

- The SELECT query displays entire row, including duplicate
- DISTINCT command is used For eliminate duplicate
- Syntax:

  SELECT DISTINCT [column_name]

  FROM [table_name];

```
SELECT department_id
FROM employees;
```

1

DEPARTMENT_ID
90
90
90

...

20 rows selected.

```
SELECT DISTINCT
department_id
FROM employees;
```

2

DEPARTMENT_ID
10
20
50

...

8 rows selected.

ALIAS

- Used to rename a table or column
- To make it shorter and simpler to write table or column names
- Syntax:

  SELECT [column_attribute] AS [alias]

  FROM [table_name];

```
SELECT last_name AS name,
commission_pct comm
FROM employees;
```

NAME	COMM
King	
Kochhar	
De Haan	

...

20 rows selected.

T-SQL

CASE

CASE EXPRESSION

- Its uses similar like IF-THEN-ELSE
- Syntax:

```
SELECT [ column_name ], [ column_name ],
CASE [expr]
WHEN [comparison_exp1] THEN [return_expr1]
WHEN [comparison_exp2] THEN [return_expr2]
WHEN [comparison_exp3] THEN [return_expr3]
ELSE [ else_expr ]
END AS [ alias_name ]
FROM [ table_name ];
```

CASE EXPRESSION (1)

```
SELECT last_name, job_id, salary,
CASE job_id WHEN 'IT_PROG' THEN 1.10*salary
WHEN 'ST_CLERK' THEN 1.15*salary
WHEN 'SA_REP' THEN 1.20*salary
ELSE salary
END AS REVISED_SALARY
FROM employees;
```

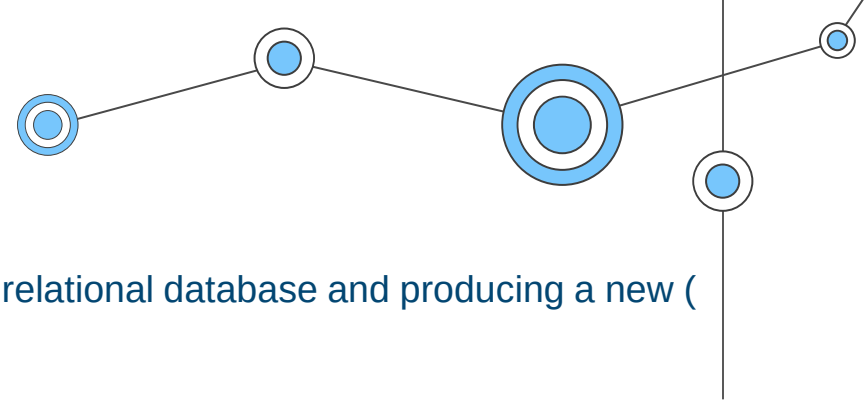
LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...			
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Rajs	ST_CLERK	3500	4025
...			
Gietz	AC_ACCOUNT	8300	8300

20 rows selected.

T-SQL

Join

JOIN



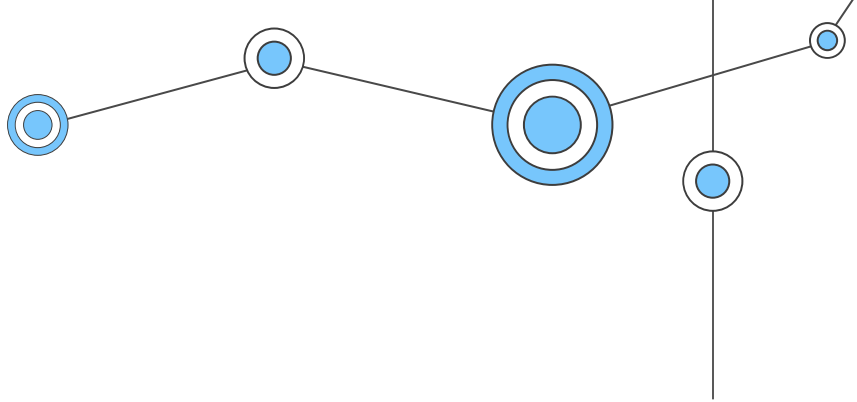
- Combination of records from two or more tables in a relational database and producing a new (*temporary*) table called a *joined table.*

- Table merging is done through certain columns/keys that have related values to get one set of data.

Benefits of JOIN

- allows us to retrieve data from multiple tables through a single *query.*
- link one table to another table

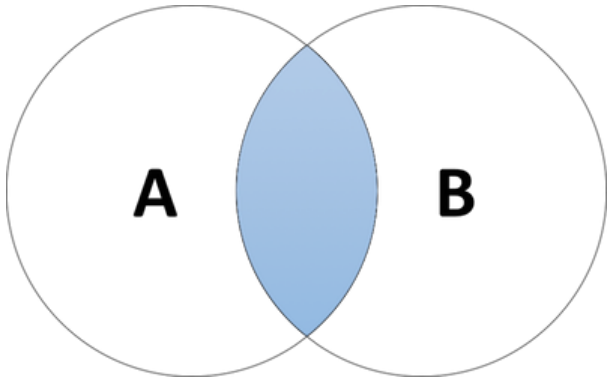
JOIN TYPE



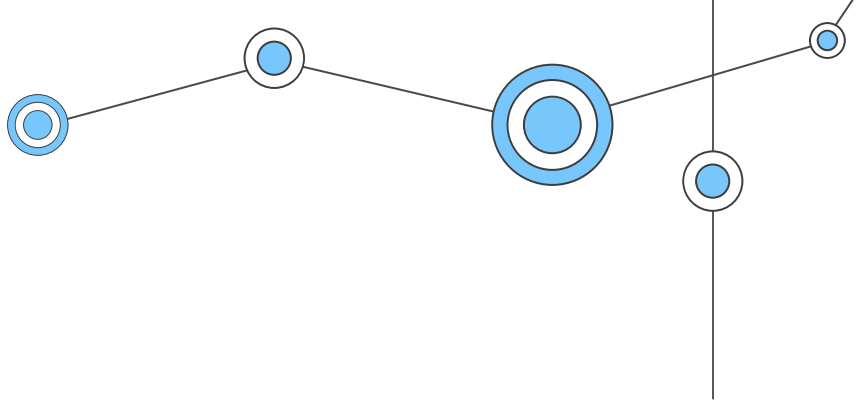
- Inner Join
- Outer Join
- Self Join
- Cross Join

INNER JOIN

- The type of JOIN used to obtain related data from two or more tables.
- Inner Join will not display unrelated data



INNER JOIN



- Syntax explicit

```
SELECT columns
FROM TableA as A
INNER JOIN TableB as B
ON A.columnName = B.columnName ;
```

- Implicit inner join

```
SELECT columns
FROM TableA as A,TableB as B
WHERE A. columnName = B . columnName ;
```

INNER JOIN

- Join multiple tables

```
SELECT columns
FROM TableA as A
INNER JOIN TableB as B
ON A.columnName = B.columnName
INNER JOIN Table C as C
ON C . columnName = B. columnName
```

Example table

- Customer Table

	id_pelanggan	nama	email
1	1	Alfa	alfa@yahoo.com
2	2	Beta	beta@yahoo.com
3	3	Charlie	charlie@gmail.com
4	4	Delta	delta@gmail.com

- Sales Table

	id_transaksi	id_pelanggan	tgl_transaksi	total_transaksi
1	1	1	2017-02-22	230000
2	2	3	2017-02-22	195000
3	3	2	2017-01-01	1710000
4	4	1	2017-02-04	310000
5	5	NULL	2017-02-10	80000

INNER JOIN

- Example

```
SELECT *
FROM customers
JOIN sales
ON customer . customer_id = sales . customer_id ;
```

id_pelanggan	nama	email	id_transaksi	id_pelanggan	tgl_transaksi	total_transaksi
1	Alfa	alfa@yahoo.com	1	1	2017-02-22	230000
3	Charlie	charlie@gmail.com	2	3	2017-02-22	195000
2	Beta	beta@yahoo.com	3	2	2017-01-01	1710000
1	Alfa	alfa@yahoo.com	4	1	2017-02-04	310000

Customer table

Sales table

INNER JOIN

- Example

```
SELECT
pl . customer_id , pl . name ,
pn . transaction_date , pn . total_transactions
FROM customer pl
INNER JOIN sales pn
ON pl . customer_id = pn . customer_id ;
```

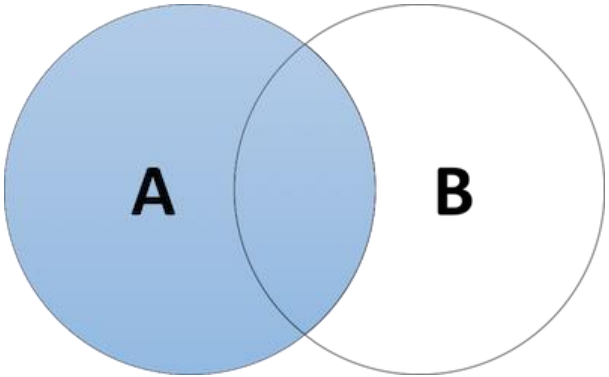
id_pelanggan	nama	tgl_transaksi	total_transaksi
1	Alfa	2017-02-22	230000
3	Charlie	2017-02-22	195000
2	Beta	2017-01-01	1710000
1	Alfa	2017-02-04	310000

OUTER JOIN

- Outer Join is divided into three types, namely:
  - Left Outer Join
  - Right Outer Join
  - Full Outer Join

LEFT OUTER JOIN

- *Left outer join* or *left join* displays all data from the left table, plus matching values from the right table or **NULL** if there are no matching values.



Left outer JOIn

- Syntax

```
SELECT columns
FROM TableA as A
LEFT OUTER JOIN TableB as B
ON A.columnName = B.columnName ;
```

Or

```
SELECT columns
FROM TableA as A
LEFT JOIN TableB as B
ON A.columnName = B.columnName ;
```

Example table

- Customer Table

	id_pelanggan	nama	email
1	1	Alfa	alfa@yahoo.com
2	2	Beta	beta@yahoo.com
3	3	Charlie	charlie@gmail.com
4	4	Delta	delta@gmail.com

- Sales Table

	id_transaksi	id_pelanggan	tgl_transaksi	total_transaksi
1	1	1	2017-02-22	230000
2	2	3	2017-02-22	195000
3	3	2	2017-01-01	1710000
4	4	1	2017-02-04	310000
5	5	NULL	2017-02-10	80000

LEFT OUTER JOIN

- Example

```
SELECT *
FROM customer pl
LEFT JOIN pn sales
ON pl . customer_id = pn . customer_id ;
```

- Tabel Pelanggan

	id_pelanggan	nama	email
1	1	Alfa	alfa@yahoo.com
2	2	Beta	beta@yahoo.com
3	3	Charlie	charlie@gmail.com
4	4	Delta	delta@gmail.com

- Tabel Penjualan

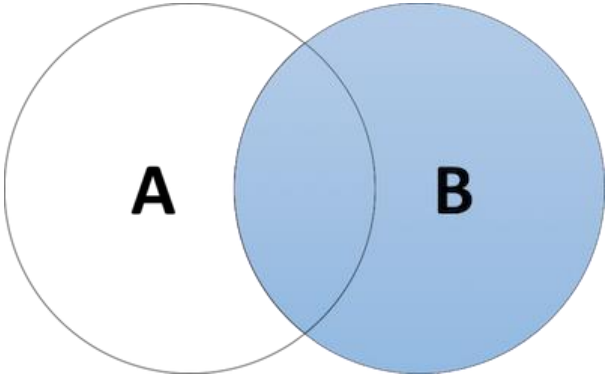
	id_transaksi	id_pelanggan	tgl_transaksi	total_transaksi
1	1	1	2017-02-22	230000
2	2	3	2017-02-22	195000
3	3	2	2017-01-01	1710000
4	4	1	2017-02-04	310000
5	5	NULL	2017-02-10	80000

	id_pelanggan	nama	email	id_transaksi	id_pelanggan	tgl_transaksi	total_transaksi
1	1	Alfa	alfa@yahoo.com	1	1	2017-02-22	230000
2	1	Alfa	alfa@yahoo.com	4	1	2017-02-04	310000
3	2	Beta	beta@yahoo.com	3	2	2017-01-01	1710000
4	3	Charlie	charlie@gmail.com	2	3	2017-02-22	195000
5	4	Delta	delta@gmail.com	NULL	NULL	NULL	NULL

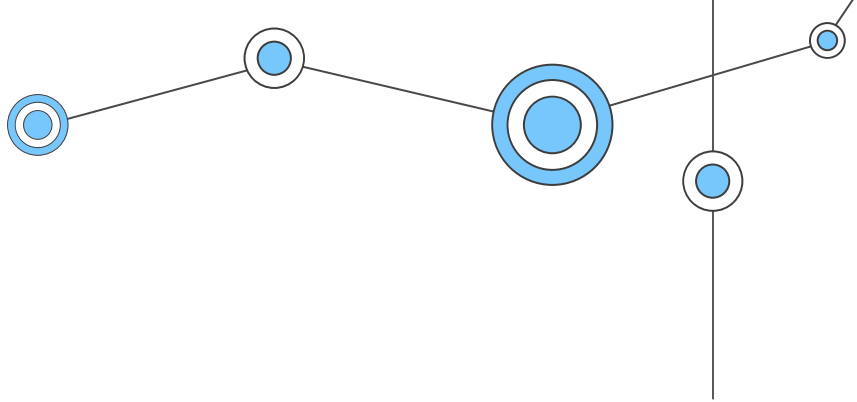
There is no sales data available related to customer data

RIGHT OUTER JOIN

- *A right outer join* or *right join* displays all data from the right table, plus matching values from the left table or **NULL** if there are no matching values.
- The opposite of Left Outer Join



RIGHT OUTER JOIN



- Syntax

```
SELECT columns
FROM TableA as A
RIGHT OUTER JOIN TableB as B
ON A.columnName = B.columnName ;
```

Or

```
SELECT columns
FROM TableA as A
RIGHT JOIN TableB as B
ON A.columnName = B.columnName ;
```

TABLE EXAMPLE

- Customer Table

	id_pelanggan	nama	email
1	1	Alfa	alfa@yahoo.com
2	2	Beta	beta@yahoo.com
3	3	Charlie	charlie@gmail.com
4	4	Delta	delta@gmail.com

- Sales Table

	id_transaksi	id_pelanggan	tgl_transaksi	total_transaksi
1	1	1	2017-02-22	230000
2	2	3	2017-02-22	195000
3	3	2	2017-01-01	1710000
4	4	1	2017-02-04	310000
5	5	NULL	2017-02-10	80000

- Tabel Pelanggan

	id_pelanggan	nama	email
1	1	Alfa	alfa@yahoo.com
2	2	Beta	beta@yahoo.com
3	3	Charlie	charlie@gmail.com
4	4	Delta	delta@gmail.com

- Tabel Penjualan

	id_transaksi	id_pelanggan	tgl_transaksi	total_transaksi
1	1	1	2017-02-22	230000
2	2	3	2017-02-22	195000
3	3	2	2017-01-01	1710000
4	4	1	2017-02-04	310000
5	5	NULL	2017-02-10	80000

RIGHT OUTER JOIN

- Tabel Pelanggan

	id_pelanggan	nama	email
1	1	Alfa	alfa@yahoo.com
2	2	Beta	beta@yahoo.com
3	3	Charlie	charlie@gmail.com
4	4	Delta	delta@gmail.com

- Tabel Penjualan

	id_transaksi	id_pelanggan	tgl_transaksi	total_transaksi
1	1	1	2017-02-22	230000
2	2	3	2017-02-22	195000
3	3	2	2017-01-01	1710000
4	4	1	2017-02-04	310000
5	5	NULL	2017-02-10	80000

- Example

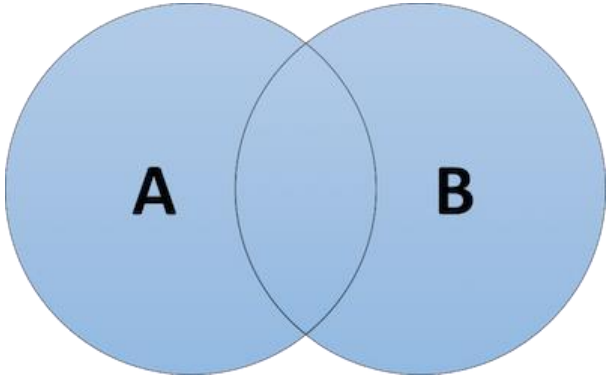
```
SELECT *
FROM customer pl
RIGHT JOIN sales pn
ON pl . customer_id = pn . customer_id ;
```

id_pelanggan	nama	email	id_transaksi	id_pelanggan	tgl_transaksi	total_transaksi
1	Alfa	alfa@yahoo.com	1	1	2017-02-22	230000
3	Charlie	charlie@gmail.com	2	3	2017-02-22	195000
2	Beta	beta@yahoo.com	3	2	2017-01-01	1710000
1	Alfa	alfa@yahoo.com	4	1	2017-02-04	310000
NULL	NULL	NULL	5	NULL	2017-02-10	80000

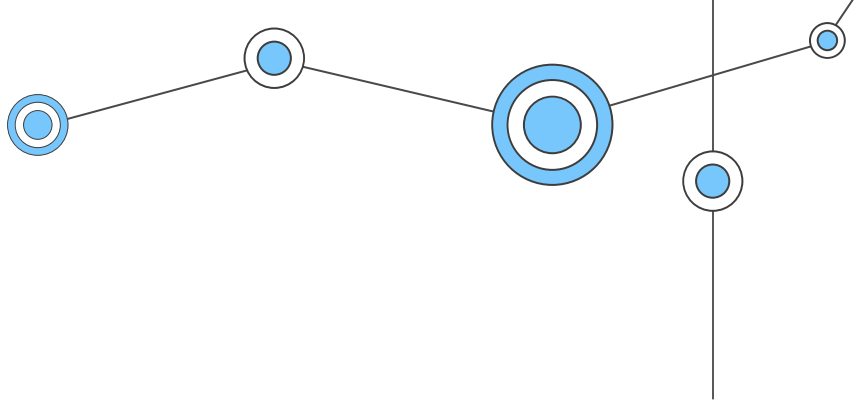
There is no customer data related to sales data

FULL OUTER JOIN

- *Full outer join* or *full join* is essentially a combination of left and right join.
- A will return **all rows from both tables** including data with NULL values.



FULL OUTER JOIN



- Syntax

```
SELECT columns
FROM TableA as A
FULL OUTER JOIN TableB as B
ON A.columnName = B.columnName ;
```

Or

```
SELECT columns
FROM TableA as A
FULL JOIN TableB as B
ON A.columnName = B.columnName ;
```

Example table

- Customer Table

	id_pelanggan	nama	email
1	1	Alfa	alfa@yahoo.com
2	2	Beta	beta@yahoo.com
3	3	Charlie	charlie@gmail.com
4	4	Delta	delta@gmail.com

- Sales Table

	id_transaksi	id_pelanggan	tgl_transaksi	total_transaksi
1	1	1	2017-02-22	230000
2	2	3	2017-02-22	195000
3	3	2	2017-01-01	1710000
4	4	1	2017-02-04	310000
5	5	NULL	2017-02-10	80000

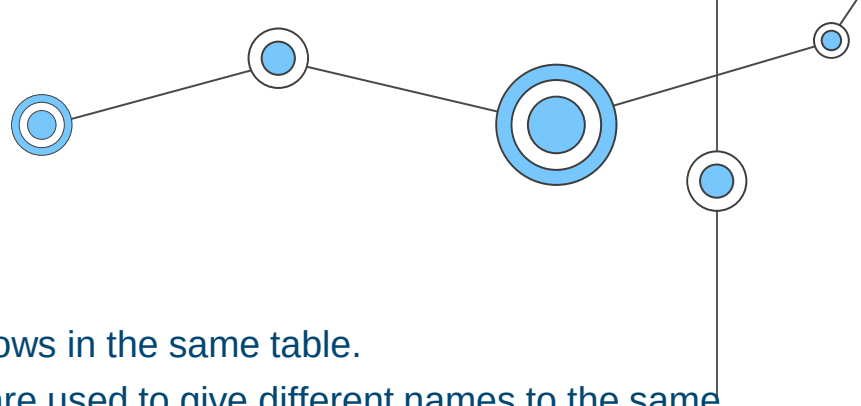
FULL OUTER JOIN

- Example

```
SELECT *
FROM customer pl
FULL JOIN pn sales
ON pl . customer_id = pn . customer_id ;
```

id_pelanggan	nama	email	id_transaksi	id_pelanggan	tgl_transaksi	total_transaksi
1	Alfa	alfa@yahoo.com	1	1	2017-02-22	230000
1	Alfa	alfa@yahoo.com	4	1	2017-02-04	310000
2	Beta	beta@yahoo.com	3	2	2017-01-01	1710000
3	Charlie	charlie@gmail.com	2	3	2017-02-22	195000
4	Delta	delta@gmail.com	NULL	NULL	NULL	NULL
NULL	NULL	NULL	5	NULL	2017-02-10	80000

SELF JOIN



- *Self Join* allows us to join a table with itself.
- Useful for displaying data hierarchies or comparing rows in the same table.
- Self Join uses the INNER JOIN clause, and aliases are used to give different names to the same table.
- Syntax

```
SELECT a.column_name , b.column_name ...
FROM table1 a, table1 b
WHERE a.common_field = b.common_field ;
```

SELF JOIN

- Example

staff_id	first_name	last_name	email	phone	active	store_id	manager_id
1	Fabiola	Jackson	fabiola.jackson@bikes.shop	(831) 555-5554	1	1	NULL
2	Mireya	Copeland	mireya.copeland@bikes.shop	(831) 555-5555	1	1	1
3	Genna	Serrano	genna.serrano@bikes.shop	(831) 555-5556	1	1	2
4	Virgie	Wiggins	virgie.wiggins@bikes.shop	(831) 555-5557	1	1	2
5	Jannette	David	jannette.david@bikes.shop	(516) 379-4444	1	2	1
6	Marcelene	Boyer	marcelene.boyer@bikes.shop	(516) 379-4445	1	2	5
7	Venita	Daniel	venita.daniel@bikes.shop	(516) 379-4446	1	2	5
8	Kali	Vargas	kali.vargas@bikes.shop	(972) 530-5555	1	3	1
9	Layla	Terrell	layla.terrell@bikes.shop	(972) 530-5556	1	3	7
10	Bernardine	Houston	bernardine.houston@bikes.shop	(972) 530-5557	1	3	7

- In the sales.staffs table, there is a **manager_id column** that shows the manager of each staff.
- For example, Mireya will give her work report to Fabiola.

SELF JOIN

- To get each staff member who will report their work to whom, the following query is used:

```
SELECT
e.first_name + ' ' + e.last_name employee ,
m.first_name + ' ' + m.last_name manager
FROM
    sales.staffs e
INNER JOIN sales.staffs m
ON m.staff_id = e.manager_id;
```

employee	manager
Mireya Copeland	Fabiola Jackson
Jannette David	Fabiola Jackson
Kali Vargas	Fabiola Jackson
Marcelene Boyer	Jannette David
Venita Daniel	Jannette David
Genna Serrano	Mireya Copeland
Virgie Wiggins	Mireya Copeland
Layla Terrell	Venita Daniel
Bernardine Houston	Venita Daniel

CROSS JOIN

- *Cross Join* is used to get **combined data** from two or more tables.
- For example, n = number of rows of data in the table on the left, and m = number of rows of data in the table on the right. Then the result of the number of rows from CROSS JOIN is **n X m rows of data** .
- Syntax:

```
SELECT columns
FROM TableA
CROSS JOIN TableB ;
```

CROSS JOIN

For example, viewing all customer and sales combinations.

```
SELECT *
FROM customers
CROSS JOIN sales ;
```

id_pelanggan	nama	email	id_transaksi	id_pelanggan	tgl_transaksi	total_transaksi
1	Alfa	alfa@yahoo.com	1	1	2017-02-22	230000
1	Alfa	alfa@yahoo.com	2	3	2017-02-22	195000
1	Alfa	alfa@yahoo.com	3	2	2017-01-01	1710000
1	Alfa	alfa@yahoo.com	4	1	2017-02-04	310000
1	Alfa	alfa@yahoo.com	5	NULL	2017-02-10	80000
2	Beta	beta@yahoo.com	1	1	2017-02-22	230000
2	Beta	beta@yahoo.com	2	3	2017-02-22	195000
2	Beta	beta@yahoo.com	3	2	2017-01-01	1710000
2	Beta	beta@yahoo.com	4	1	2017-02-04	310000
2	Beta	beta@yahoo.com	5	NULL	2017-02-10	80000
3	Charlie	charlie@gmail.com	1	1	2017-02-22	230000
3	Charlie	charlie@gmail.com	2	3	2017-02-22	195000
3	Charlie	charlie@gmail.com	3	2	2017-01-01	1710000
3	Charlie	charlie@gmail.com	4	1	2017-02-04	310000
3	Charlie	charlie@gmail.com	5	NULL	2017-02-10	80000
4	Delta	delta@gmail.com	1	1	2017-02-22	230000
4	Delta	delta@gmail.com	2	3	2017-02-22	195000
4	Delta	delta@gmail.com	3	2	2017-01-01	1710000
4	Delta	delta@gmail.com	4	1	2017-02-04	310000
4	Delta	delta@gmail.com	5	NULL	2017-02-10	80000

T-SQL

SORTING

sorting

- A process of rearranging data with a certain pattern, so that it is arranged regularly according to certain rules.
- In SQL, the ORDER BY function is used to display data in order.

SORTING

- There are 2 types of sorting:
  - **ASCENDING** (ascending order), sorting data from smaller values to larger values. (A to Z or 1 to 99)
  - **DESCENDING** (descending order), sorting data from larger values to smaller values. (Z to A or 99 to 1)

- Syntax

```
SELECT column1 , column2 , ...
FROM table_name
ORDER BY column1, column2, ... ASC | DESC ;
```

If you only use ORDER BY, without ASC / DESC, then **by default, the data will be sorted ascending.**

SORTING

- Example:

```
SELECT contact name , address , city , phone
FROM Sales . Customers
ORDER BY contactname desc ;
```

contactname	address	city	phone
Young, Robin	0123 Grizzly Peak Rd.	Butte	(406) 555-0121
Wojciechowska, Agnieszka	P.O. Box 1234	Lander	(307) 555-0114
Wickham, Jim	Luisenstr. 0123	Münster	0251-456789
Welcker, Brian	4567 Wadhurst Rd.	London	(171) 901-2345
Watters, Jason M.	Gran Vía, 4567	Madrid	(91) 567 8901
Voss, Florian	Strada Provinciale 7890	Reggio Emilia	0522-012345
Veronesi, Giorgio	Taucherstraße 1234	Cunewalde	0372-12345
Veninga, Tjeerd	1234 DaVinci Blvd.	Kirkland	(206) 555-0124
Uppal, Sunil	Estrada da saúde n. 6789	Lisboa	(1) 789-0123
Tuntisangaroon, Sittichai	6789, rue du Commerce	Lyon	78.90.12.34
Tollevsen, Bjørn	5678, boulevard Charonne	Paris	(1) 89.01.23.45

T-SQL

FILTERING

TOP N

- Used to display records that contain a certain number which is the top or bottom order of a set of records.

- Syntax

```
SELECT TOP number | column_name ( number )
FROM table_name
WHERE conditions ;
```

TOP N

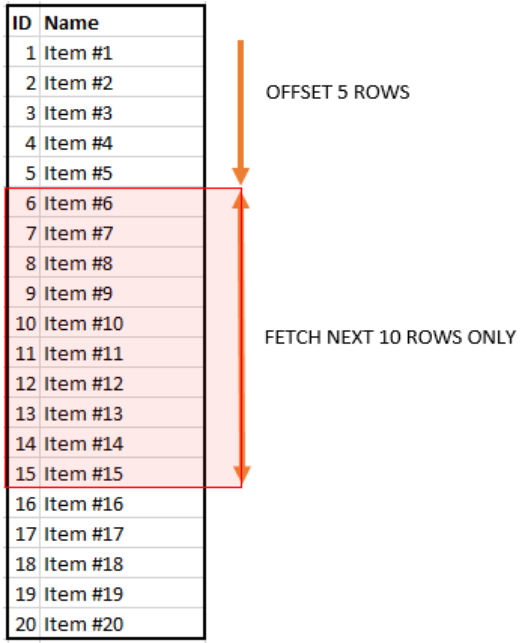
- Example

```
SELECT TOP 5
 contact name , address , city , phone
FROM Sales . Customers ;
```

	contactname	address	city	phone
1	Allen, Michael	Obere Str. 0123	Berlin	030-3456789
2	Hassall, Mark	Avda. de la Constitución 5678	México D.F.	(5) 789-0123
3	Peoples, John	Mataderos 7890	México D.F.	(5) 123-4567
4	Arndt, Torsten	7890 Hanover Sq.	London	(171) 456-7890
5	Higginbotham, Tom	Berguvsvägen 5678	Luleå	0921-67 89 01

OFFSET FETCH

- *OFFSET-FETCH* is used together with the SELECT and ORDER BY clauses to retrieve a series of records (range).



OFFSET

- *OFFSET* is used to specify the number of rows to skip before starting to return rows from the query.
- Offset can only be used with the ORDER BY clause.
- The OFFSET value must be greater than or equal to zero.
- Syntax

```
SELECT column_name (s)
FROM table_name
WHERE condition
ORDER BY column_name
OFFSET rows_to_skip ROWS;
```

OFFSET

- Example:

```
SELECT product_name , list_price
FROM production . products
ORDER BY list_price , product_name
OFFSET 10 ROWS ;
```

product_name	list_price
Strider Classic 12 Balance Bike - 2018	89.99
Sun Bicycles Lil Kitt'n - 2017	109.99
Trek Boy's Kickster - 2015/2017	149.99
Trek Girl's Kickster - 2017	149.99
Trek Kickster - 2018	159.99
Trek Precaliber 12 Boys - 2017	189.99
Trek Precaliber 12 Girls - 2017	189.99
Trek Precaliber 12 Boy's - 2018	199.99
Trek Precaliber 12 Girl's - 2018	199.99
Haro Shredder 20 - 2017	209.99
Haro Shredder 20 Girls - 2017	209.99
Trek Precaliber 16 Boy's - 2018	209.99
Trek Precaliber 16 Boys - 2017	209.99
Trek Precaliber 16 Girl's - 2018	209.99
Trek Precaliber 16 Girls - 2017	209.99
Trek Precaliber 20 Boy's - 2018	229.99
Trek Precaliber 20 Girl's - 2018	229.99
Haro Shredder Pro 20 - 2017	249.99

product_name	list_price
Haro Shredder 20 Girls - 2017	209.99
Trek Precaliber 16 Boy's - 2018	209.99
Trek Precaliber 16 Boys - 2017	209.99
Trek Precaliber 16 Girl's - 2018	209.99
Trek Precaliber 16 Girls - 2017	209.99
Trek Precaliber 20 Boy's - 2018	229.99
Trek Precaliber 20 Girl's - 2018	229.99
Haro Shredder Pro 20 - 2017	249.99
Strider Sport 16 - 2018	249.99
Trek MT 201 - 2018	249.99
Sun Bicycles Revolutions 24 - 2017	250.99
Sun Bicycles Revolutions 24 - Girl's - 2017	250.99
Electra Cruiser 1 (24-Inch) - 2016	269.99
Electra Cruiser 1 (24-Inch) - 2016	269.99
Electra Cruiser 1 - 2016/2017/2018	269.99

FETCH

- *FETCH* is used to specify the number of rows to be returned after the OFFSET clause is processed.
- The OFFSET clause is mandatory while the FETCH clause is optional.
- Syntax

```
SELECT column_name (s)
FROM table_name
WHERE condition
ORDER BY column_name
OFFSET rows_to_skip ROWS
FETCH NEXT number_of_rows ROWS ONLY;
```

FETCH

- For example, skip the first 10 products and show the next 10 products:

```
SELECT product_name , list_price
FROM production . products
ORDER BY list_price , product_name
OFFSET 10 ROWS
FETCH NEXT 10 ROWS ONLY ;
```

Thanks!

Do you have any questions?

Advanced
JTI POLINEMA