



Basis Data Lanjut

Join, Sorting, Filtering Data

JTI-POLINEMA

OUTLINE



1. JOIN

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

2. Sorting

3. Filtering

- Top dan OFFSET-FETCH

JOIN

JOIN



- Kombinasi record dari dua atau lebih tabel di dalam basis data relasional dan menghasilkan sebuah tabel (*temporary*) baru yang disebut sebagai *joined tabel*
- Penggabungan tabel yang dilakukan melalui kolom / key tertentu yang memiliki nilai terkait untuk mendapatkan satu set data.

MANFAAT JOIN



- memperbolehkan kita untuk mengambil data dari beberapa tabel melalui satu *query*
- menghubungkan satu tabel dengan tabel yang lain

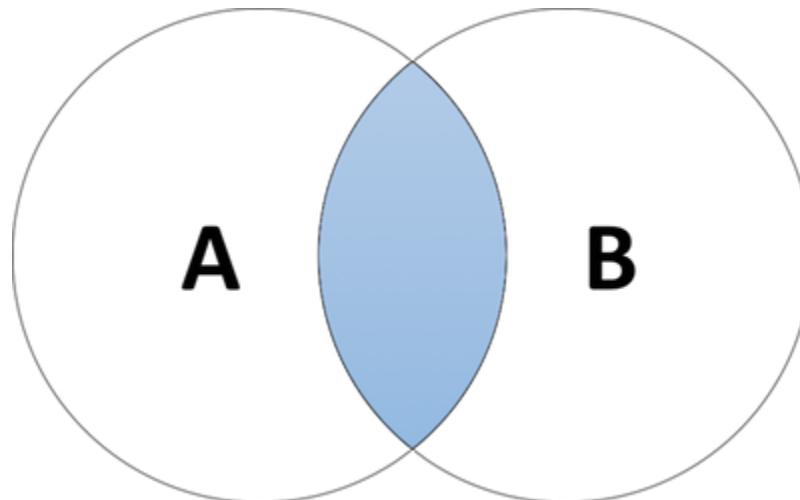


TIPE JOIN

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

INNER JOIN

- Jenis JOIN yang digunakan untuk mendapatkan data yang saling berelasi dari dua tabel atau lebih
- Inner Join tidak akan menampilkan data yang tidak berelasi





INNER JOIN

- Syntax

```
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 table

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



CONTOH TABEL

- 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 |

INNER JOIN



- Contoh

SELECT *

FROM pelanggan

JOIN penjualan

ON pelanggan.id_pelanggan = penjualan.id_pelanggan;

| 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 |

Tabel pelanggan

Tabel penjualan

INNER JOIN



- Contoh

SELECT

```
pl.id_pelanggan, pl.nama,  
pn.tgl_transaksi, pn.total_transaksi  
FROM pelanggan pl  
JOIN penjualan pn  
ON pl.id_pelanggan = pn.id_pelanggan;
```

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

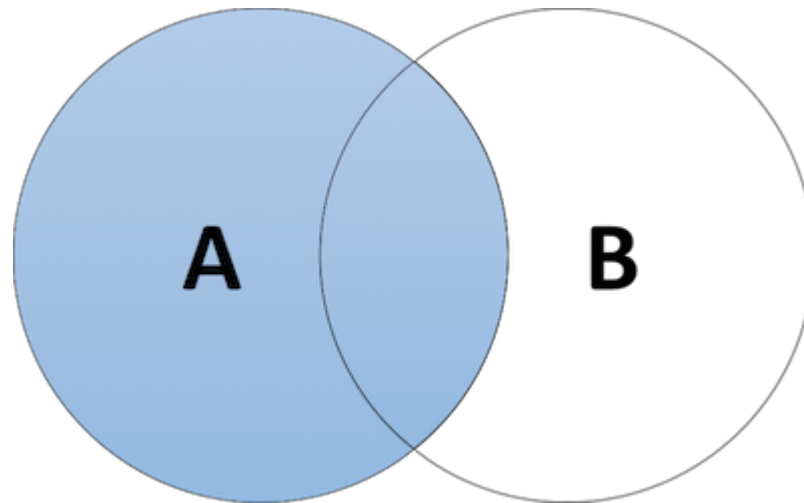


OUTER JOIN

- Outer Join terbagi menjadi tiga jenis, yaitu:
 - Left Outer Join
 - Right Outer Join
 - Full Outer Join

LEFT OUTER JOIN

- *Left outer join* atau *left join* menampilkan semua data dari tabel sebelah kiri, ditambah dengan nilai dari tabel sebelah kanan yang sesuai atau **NULL** jika tidak ada nilai yang sesuai.





LEFT OUTER JOIN

- Syntax

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

Atau

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



CONTOH TABEL

- 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 |

LEFT OUTER JOIN



- Contoh

SELECT *

FROM pelanggan pl

LEFT JOIN penjualan pn

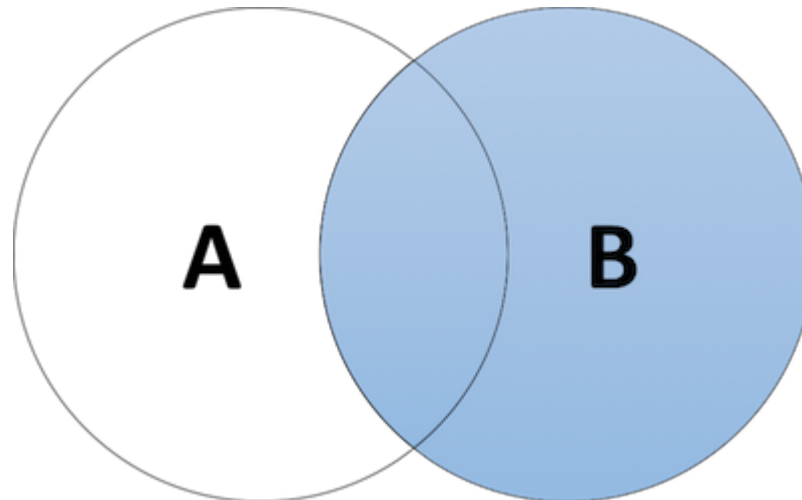
ON pl.id_pelanggan = pn.id_pelanggan;

| | 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 |

Tidak ada data penjualan yang berelasi dengan data pelanggan

RIGHT OUTER JOIN

- *Right outer join* atau *right join* menampilkan semua data dari tabel sebelah kanan, ditambah dengan nilai dari tabel sebelah kiri yang sesuai atau **NULL** jika tidak ada nilai yang sesuai.
- Kebalikan dari Left Outer Join





RIGHT OUTER JOIN

- Syntax

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

Atau

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



CONTOH TABEL

- 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



- Contoh

SELECT *

FROM pelanggan p1

RIGHT JOIN penjualan pn

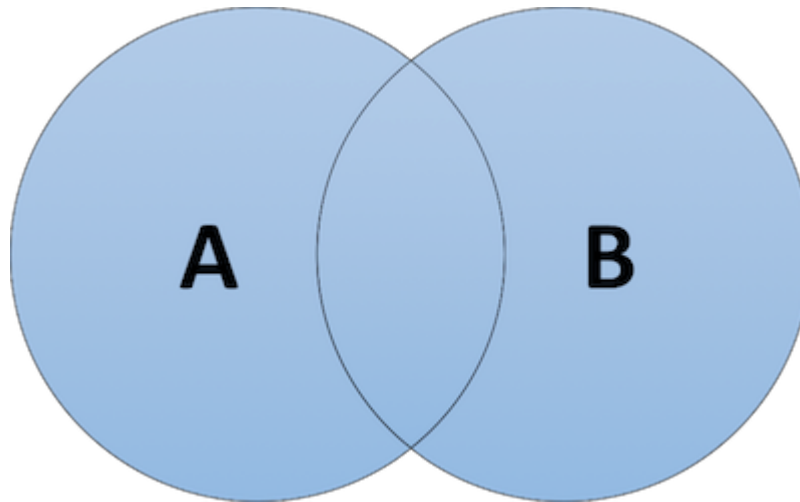
ON p1.id_pelanggan = pn.id_pelanggan;

| 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 |

Tidak ada data pelanggan yang berelasi dengan data penjualan

FULL OUTER JOIN

- *Full outer join* atau *full join* pada hakikatnya merupakan kombinasi dari left dan right join
- Akan mengembalikan **seluruh baris dari kedua tabel** termasuk data-data yang bernilai NULL





FULL OUTER JOIN

- Syntax

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

Atau

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



CONTOH TABEL

- 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 |

FULL OUTER JOIN



- Contoh

SELECT *

FROM pelanggan p1

FULL JOIN penjualan pn

ON p1.id_pelanggan = pn.id_pelanggan;

| id_pelanggan | nama | email | id_transaksi | id_pelanggan | tgl_transaksi | total_transaksi |
|--------------|---------|-------------------|--------------|--------------|---------------|-----------------|
| 1 | Afa | alfa@yahoo.com | 1 | 1 | 2017-02-22 | 230000 |
| 1 | Afa | 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* memungkinkan kita untuk menggabungkan sebuah tabel dengan tabel itu sendiri.
- Berguna untuk menampilkan hirarki data atau membandingkan baris pada tabel yang sama.
- Self Join menggunakan klausa INNER JOIN, dan alias digunakan untuk memberikan nama yang berbeda untuk tabel yang sama.



SELF JOIN

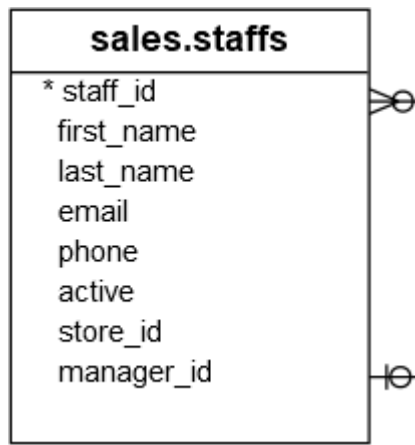
- Syntax

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



SELF JOIN

- Contoh



| 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 |

- Pada tabel sales.staffs, terdapat kolom **manager_id** yang menunjukkan manajer dari setiap staff.
- Misal, Mireya akan memberikan laporan pekerjaannya pada Fabiola



SELF JOIN

- Untuk mendapatkan setiap staff akan melaporkan pekerjaannya pada siapa saja, digunakan query sebagai berikut:

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* digunakan untuk mendapatkan **data kombinasi** dari dua tabel atau lebih.
- Misal, n = jumlah baris data pada tabel di sebelah kiri, dan m = jumlah baris data pada tabel di sebelah kanan. Maka hasil jumlah baris dari CROSS JOIN adalah **$n \times m$ baris data**.
- Syntax:

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



CROSS JOIN

Contoh, melihat semua kombinasi pelanggan dan penjualan.

SELECT *

FROM pelanggan

CROSS JOIN penjualan;

| 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 |

SORTING

SORTING



- Suatu proses menyusun kembali data dengan suatu pola tertentu, sehingga tersusun secara teratur menurut aturan tertentu.
- Pada SQL, digunakan fungsi ORDER BY untuk menampilkan data secara terurut.



SORTING

- Terdapat 2 jenis pengurutan:
 - **ASCENDING** (urut naik), pengurutan data dari nilai yang lebih kecil menuju ke nilai yang lebih besar. (A ke Z atau 1 ke 99)
 - **DESCENDING** (urut turun), pengurutan data dari nilai yang lebih besa menuju ke nilai yang lebih kecil. (Z ke A atau 99 ke 1)
- Syntax

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

Jika hanya menggunakan ORDER BY, tanpa ASC / DESC, maka **secara default, data akan terurut naik**



SORTING

- Contoh:

```
SELECT contactname, 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 |
| Tollefsen, Bjørn | 5678, boulevard Charonne | Paris | (1) 89.01.23.45 |

FILTERING



TOP N

- Digunakan untuk menampilkan record yang berisi sejumlah tertentu yang merupakan urutan teratas atau urutan terbawah dari sekumpulan record.
- Syntax

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



TOP N

- Contoh

SELECT TOP 5

contactname, 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 | Amdt, Torsten | 7890 Hanover Sq. | London | (171) 456-7890 |
| 5 | Higginbotham, Tom | Berguvsvägen 5678 | Luleå | 0921-67 89 01 |



OFFSET-FETCH

- *OFFSET-FETCH* digunakan bersama dengan klausa `SELECT` dan `ORDER BY` untuk mengambil serangkaian record (range).

| ID | Name |
|----|----------|
| 1 | Item #1 |
| 2 | Item #2 |
| 3 | Item #3 |
| 4 | Item #4 |
| 5 | Item #5 |
| 6 | Item #6 |
| 7 | Item #7 |
| 8 | Item #8 |
| 9 | Item #9 |
| 10 | Item #10 |
| 11 | Item #11 |
| 12 | Item #12 |
| 13 | Item #13 |
| 14 | Item #14 |
| 15 | Item #15 |
| 16 | Item #16 |
| 17 | Item #17 |
| 18 | Item #18 |
| 19 | Item #19 |
| 20 | Item #20 |

OFFSET 5 ROWS

FETCH NEXT 10 ROWS ONLY



OFFSET

- *OFFSET* digunakan untuk menentukan jumlah baris untuk dilewati sebelum mulai mengembalikan baris dari query.
- Offset hanya dapat digunakan dengan klausa **ORDER BY**
- Nilai **OFFSET** harus lebih besar dari atau sama dengan nol
- Syntax

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


OFFSET



- Contoh:

```
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* digunakan untuk menentukan jumlah baris yang akan dikembalikan setelah klausa *OFFSET* diproses.
- Klausa *OFFSET* bersifat mandatory sedangkan klausa *FETCH* bersifat opsional
- 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

- Contoh, melewati 10 produk pertama dan menampilkan 10 produk berikutnya:

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



TERIMA KASIH