Data Exploration & Preprocessing

SQL 2

Hello, everyone!

Sebelum kita memulai kelas, kita awali dengan:

1. Berdoa
2. siapkan diri
3. Pre test

Pertemuan lalu...

Let’s Recall

pertemuan sebelumnya kita sudah membahas menggunakan MySQL dan SQLite. Mulai dari menambah database, membuat table,input data, dll.

Target

1. Aggregate Functions
2. Group By
3. Order by
4. Like condition
5. Date Functions
6. Primary key, auto increment and foreign key
7. Database normalization
8. Join
9. Subquery
10. Export and import data

Tools

https://colab.research.google.com/

https://dbeaver.io/download/

Aggregate Functions

Aggregate Functions (fungsi agregat) merupakan fungsi yang melakukan kalkulasi pada sekumpulan data. Hasil dari fungsi agregat tersebut umumnya digunakan untuk memberikan insight atau wawasan dari data yang ada dan kemudian digunakan untuk analisis lanjutan.
Fungsi agregat dapat dibagi menjadi beberapa jenis yaitu:

- SUM () : untuk menghitung jumlah/total nilai dari sebuah kolom
- AVG () : untuk menghitung nilai rata-rata dari sebuah kolom
- COUNT () : untuk menghitung jumlah baris dalam sebuah tabel / banyaknya nilai pada sebuah kolom.
- MAX () : mencari nilai tertinggi dalam sebuah kolom
- MIN () : mencari nilai terendah dalam sebuah kolom

Group by

Group by digunakan untuk melakukan pengelompokan terhadap data-data yang memiliki kategori yang serupa sebelum sebelum data tersebut di kalkulasi. Penggunaan fungsi agregat biasanya di barengin dengan Group by. Contoh:

```
SELECT kolom_1, count(distinct id) as jumlah
FROM nama_kolom
GROUP BY kolom_1

SELECT kolom_1,kolom_2, count(distinct id) as jumlah
FROM nama_kolom
GROUP BY kolom_1,kolom_2
```

Pengelompokan yang dilakukan tidak terbatas pada satu kategori saja, melainkan juga dapat dikelompokkan berdasarkan beberapa kategori.

Order by

Order by digunakan untuk mengurutkan hasil query pada sebuah kolom berdasarkan dengan nilai terbesar atau terkecilnya. ASCENDING (asc) digunakan untuk mengurutkan dari kecil ke besar dan DESCENDING (desc) digunakan untuk mengurutkan dari besar ke kecil.
Contoh:

```
SELECT kolom_1, kolom_2, kolom_3
FROM nama_kolom
ORDER BY kolom_1 asc

SELECT kolom_1, kolom_2, kolom_3
FROM nama_kolom
ORDER BY kolom_1 desc

SELECT kolom_1, kolom_2, kolom_3
FROM nama_kolom
ORDER BY kolom_1 desc, kolom_2 asc
```

Like condition

Operator LIKE digunakan dalam SQL untuk membandingkan string dengan suatu pola. Hal ini sering digunakan dengan karakter wildcard "%" dan "_". Karakter "%" cocok dengan urutan apa pun yang berisi nol karakter atau lebih, dan karakter "_" cocok dengan karakter tunggal.
Contoh:

```
SELECT kolom_1, kolom_2, kolom_3
FROM nama_kolom
WHERE kolom_1 like ‘%timur%’

SELECT kolom_1, kolom_2, kolom_3
FROM nama_kolom
WHERE kolom_1 like ‘_ _ _’
```

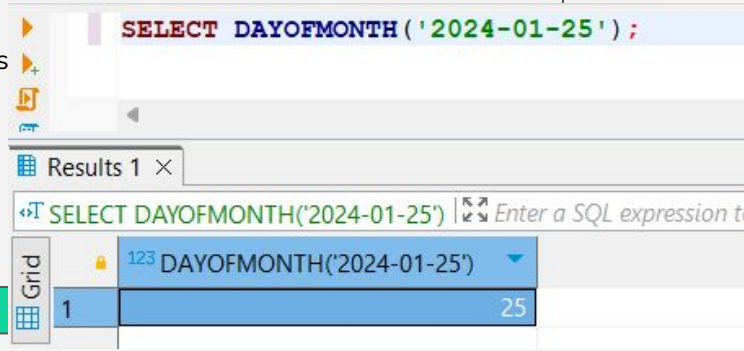
Date Functions

Date (tanggal) merupakan informasi mengenai peristiwa dan membentuk cara kita menganalisis informasi. SQL menyediakan fungsi tanggal dan waktu mendapatkan insight peristiwa di data. Fungsi date ada beberapa jenis yaitu:

1. DAYOFMONTH: Return the day of the month (0-31).
2. DATE_ADD: Add time values (intervals) to a date value
3. DATEDIFF: Subtract two dates
4. Dll.

Dokumentasi date function:

1. MySQL: https://dev.mysql.com/doc/refman/8.0/en/date-and-time-functions
2. Mariadb: https://mariadb.com/kb/en/date-time-functions/
3. Dll



Primary Key

Primary Key adalah kolom yang berisi nilai unik untuk mengidentifikasi dan memastikan tiap baris dalam tabel punya sifat unik.
Sifat unik artinya punya kode-kode khusus dan mempermudah dalam pengambilan data.
Kolom tersebut punya nilai berbeda dengan kolom lainnya, dan nggak boleh **NULL**
AUTO_INCREMENT digunakan untuk mendefinisikan kolom id terisi secara otomatis

```
CREATE TABLE department (id INT NOT NULL
AUTO_INCREMENT, name varchar(255), PRIMARY KEY (id));
```

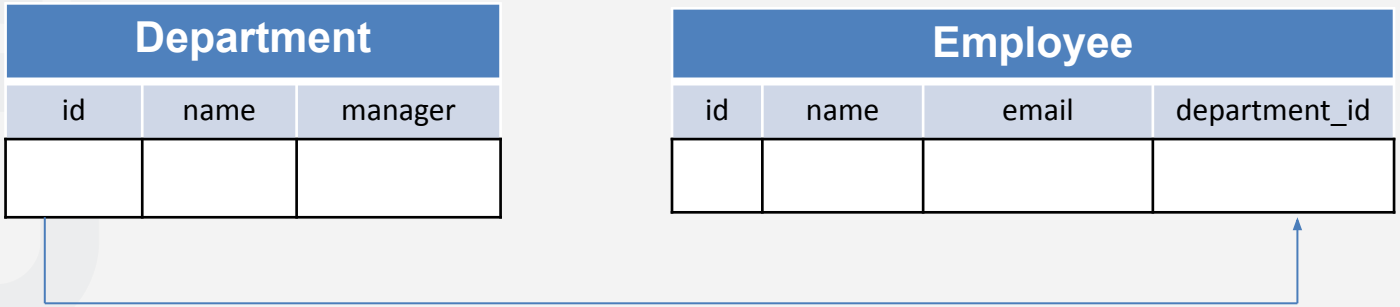
Department

id	name	manager

Foreign Key

Foreign adalah kolom primary key dalam satu tabel yang terhubung dengan primary key dari tabel lain. Tujuannya, untuk berelasi dari tabel satu ke tabel lainnya.

```
CREATE TABLE employee (id INT NOT NULL AUTO_INCREMENT,
name varchar(255), email varchar(255), department_id INT,
PRIMARY KEY (id), FOREIGN KEY (department_id)
REFERENCES(department_id));
```



Contohnya, foreign key di kedua data ini ada pada kolom "employee.department_id" dan department.id. Karena kedua tabel tersebut primary key "employee.department_id" dan "department.id" saling terhubung, sehingga bisa disebut dengan foreign key.

Database normalization

Database normalization adalah teknik desain database yang mengurangi redundansi data dan menghilangkan karakteristik yang tidak diinginkan seperti Anomali Penyisipan, Pembaruan, dan Penghapusan. Aturan normalisasi membagi tabel yang lebih besar menjadi tabel yang lebih kecil dan menghubungkannya menggunakan hubungan. Tujuan Normalisasi pada SQL adalah untuk menghilangkan data yang berlebihan (berulang) dan memastikan data disimpan secara logis.

Teknik Database Normalization

1. First Normal Form
2. Second Normal Form:
3. Third Normal Form
4. BCNF (Boyce and Codd Normal Form)

Ilustrasi Database Normalization

1. First Normal Form: tabel harus mempunyai nilai atomik (tunggal)

Dalam contoh berikut, Husna memiliki dua nomor telepon yang disimpan dalam satu sel yang melanggar aturan 1NF.

Jadi kita perlu memastikan bahwa sel dalam tabel tidak boleh menampung banyak nilai, baru kemudian kita dapat mengatakan bahwa tabel tersebut dalam 1NF.

ID	Name	Area	Phone
101	Nadya	Tangerang Selatan	0801982730
102	Husna	Bogor	0810002901 08001001002

ID	Name	Area	Phone
101	Nadya	Tangerang Selatan	0801982730
102	Husna	Bogor	0810002901
102	Husna	Bogor	08001001002

Ilustrasi Database Normalization

2. Second Normal Form :

Pada contoh tabel diatas, atribut utama tabel adalah Employee ID dan ID Department. Tabel tersebut memiliki ketergantungan parsial, karena Nama Employee dapat ditentukan oleh Employee ID dan Nama Department dapat ditentukan oleh ID Department. Jadi, tabel Employee Department melanggar aturan 2NF.

Untuk menghilangkan sebagian dependensi dari tabel ini dapat menormalkannya menjadi bentuk normal kedua, kita dapat mendekomposisi menjadi tiga tabel seperti disamping

ID Employee	Employee Name	ID Department	Department Name
101	Nadya	D03	Analis
101	Nadya	D01	Marketing
102	Husna	D04	Bisnis
103	Asanul	D02	Sales

ID Employee	Employee Name
101	Nadya
101	Nadya
102	Husna
103	Asanul

ID Employee	ID Department
101	D03
101	D01
102	D04
103	D02

ID Department	Department Name
D03	Analis
D01	Marketing
D04	Bisnis
D02	Sales

Ilustrasi Database Normalization

Third Normal Form:

Pada contoh disamping Tabel di atas tidak dalam 3NF karena memiliki

ID Employee -> Employee City Ketergantungan transitif karena:

ID Employee -> Employee Zipcode

Employee Zipcode -> Employee City

Selain itu, Employee Zipcode bukanlah kunci utama dan Employee City bukanlah atribut utama.

Untuk menghilangkan ketergantungan transitif dari tabel ini lalu menormalkannya menjadi bentuk normal ketiga, kita dapat mendekomposisi tabel <EmployeeDetail> menjadi dua tabel berikut:

Employee Detail

ID Employee	Employee Name	Employee Zipcode	Employee City
101	Nadya	11431	Jakarta Selatan
101	Nadya	11220	Tangerang Selatan
102	Husna	10112	Bogor
103	Asanul	10312	Depok

ID Employee	Employee Name	Employee Zipcode
101	Nadya	11431
101	Nadya	11220
102	Husna	10112
103	Asanul	10312

Employee Zipcode	Employee City
11431	Jakarta Selatan
11220	Tangerang Selatan
10112	Bogor
10312	Depok

Ilustrasi Database Normalization

BCNF (Boyce and Codd Normal Form): Bentuk Normal Boyce-Codd adalah versi lanjutan dari 3NF karena mengandung batasan tambahan dibandingkan dengan 3NF.

Employee Detail

ID Employee	ID Department	Head
101	D03	Lia
101	D01	Lusi
102	D04	Leon
103	D02	Lala

ID Employee	ID Department
101	D03
101	D01
102	D04
103	D02

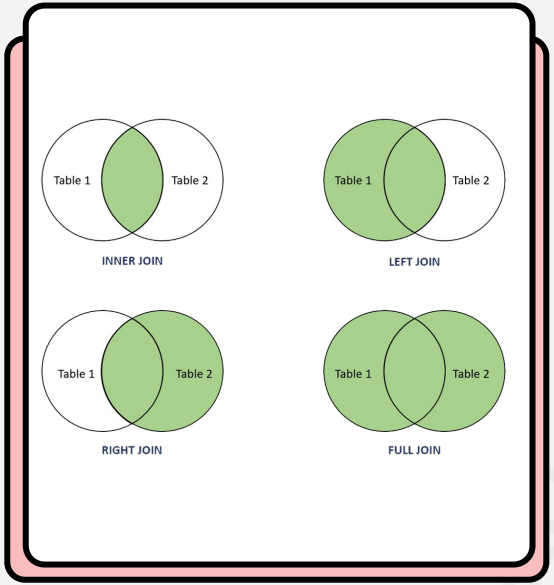
ID Department	Head
D03	Lia
D01	Lusi
D04	Leon
D02	Lala

JOIN

Jenis-jenis JOIN yang sering digunakan:

1. INNER JOIN
2. LEFT JOIN
3. RIGHT JOIN
4. FULL OUTER JOIN.

Tiap JOIN berikut ini mempunyai *output* yang berbeda terhadap data yang ditampilkan.



Contoh Tabel

TABEL 1

Contract_id	Invoice_id
1234	931
1235	120
1122	121

TABEL 2

Nama_customer	Invoice_id
Lia	931
Laili	111
Lala	131

Misal:
Tabel 1 = tabel contract
Tabel 2 = tabel invoice

INNER JOIN

TABEL 1

Contract_id	Invoice_id
1234	931
1235	120
1122	121

TABEL 2

Nama_customer	Invoice_id
Lia	931
Laili	111
Lala	131

(INNER) JOIN : Fungsi yang mengembalikan *records* dengan nilai yang bersesuaian di kedua tabel

```
SELECT
  *
FROM contract
Join invoice on contract.invoice_id = invoice.invoice_id
```

OUTPUT

Contract_id	Invoice_id	Nama_customer	Invoice_id1
1234	931	Lia	931

LEFT JOIN

TABEL 1

Contract_id	Invoice_id
1234	931
1235	120
1122	121

TABEL 2

Nama_customer	Invoice_id
Lia	931
Laili	111
Lala	131

LEFT (OUTER) JOIN : Fungsi yang akan mengembalikan semua records dari tabel kiri, dan hanya records yang sesuai dari tabel kanan

```
SELECT
  contract.contract_id,contract.invoice_id,invoice.nama_custo
mer
FROM contract
LEFT JOIN invoice on contract.invoice_id = invoice.invoice_id
```

OUTPUT

Contract_id	Invoice_id	Nama_customer
1234	931	Lia
1235	120	NULL
1122	121	NULL

RIGHT JOIN

TABEL 1

Contract_id	Invoice_id
1234	931
1235	120
1122	121

TABEL 2

Nama_customer	Invoice_id
Lia	931
Laili	111
Lala	131

RIGHT (OUTER) JOIN : Fungsi yang akan mengembalikan semua record dari tabel kanan, dan hanya record yang cocok dari tabel kiri

```
SELECT
  contract.contract_id,contract.invoice_id,invoice.nama_custo
mer,invoice.invoice_id
FROM contract
RIGHT JOIN invoice on contract.invoice_id =
invoice.invoice_id
```

OUTPUT

Contract_id	Invoice_id	Nama_customer	Invoice_id1
1234	931	Lia	931
NULL	NULL	Laili	111
NULL	NULL	Lala	131

FULL OUTER JOIN

TABEL 1

Contract_id	Invoice_id
1234	931
1235	120
1122	121

TABEL 2

Nama_customer	Invoice_id
Lia	931
Laili	111
Lala	131

FULL (OUTER) JOIN : Fungsi yang akan menggabungkan right dan left join untuk mengembalikan semua records yang sesuai di kedua tabel

```
SELECT
  contract.contract_id,contract.invoice_id,
invoice.nama_customer
FROM contract
OUTER JOIN invoice on contract.invoice_id =
invoice.invoice_id
```

OUTPUT

Contract_id	Invoice_id	Nama_customer	Invoice_id1
1234	931	Lia	931
1235	120	NULL	NULL
1122	121	NULL	NULL
NULL	NULL	Laili	111
NULL	NULL	Lala	131

SUBQUERY dan CTE

1. CTE kepanjangan dari Common Table Expression
2. Digunakan untuk menyederhanakan JOIN pada SQL ke dalam subqueries, yang hasilnya disimpan sementara oleh system
3. Jenis-jenis subquery:
    - Scalar subquery
    - Single row subquery
    - Multiple row subquery
    - Correlated subquery
    - Nested subqueries

Contoh Tabel

Tabel USER					
ID	Name	Domisili	Job	Age	Ukuran Baju
1	Nadya	Solo	Marketing	20	M
2	Husna	Semarang	Data Analis	23	M
3	Ahsan	Jakarta	Sales	21	S
4	Medina	Makasar	Guru	34	L
5	Nici	Surabaya	Dosen	38	M
6	Naza	Makasar	Polisi	40	XL

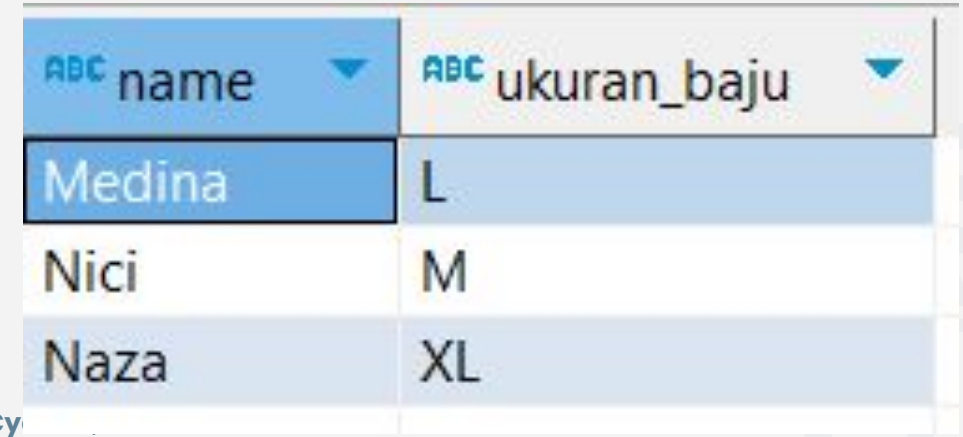
KODE_DAERAH	
Domisili	KODE
Solo	C012
Semarang	C013
Jakarta	C014
Makasar	C015
Surabaya	C016

SCALAR SUBQUERY

Pada *scalar subquery*, Kita akan membuat subquery yang bernilai tunggal untuk membandingkan data yang ada pada *query* utama dengan nilai tunggal yang dihasilkan dari *subquery*.

Sebagai contoh, Kita ingin mencari nama dan ukuran_baju penduduk yang memiliki umur di atas umur rata-rata penduduk. Dengan menggunakan contoh slide disebelumnya, *scalar subquery* dapat dituliskan dengan *logic* sebagai berikut:

```
SELECT name, ukuran_baju
FROM USER
WHERE age > (SELECT avg (age) FROM USER)
```



ABC name	ABC ukuran_baju
Medina	L
Nici	M
Naza	XL

Single row subquery

Pada *single row subquery*, Kita akan membuat subquery yang menghasilkan data dalam bentuk satu baris untuk membandingkan data yang ada pada *query* utama dengan data yang dihasilkan dari *subquery*.

Sebagai contoh, Kita ingin mencari nama dan pekerjaan penduduk yang memiliki kode_daerah domisili adalah C015. Dengan menggunakan contoh slide disebelumnya, *single subquery* dapat dituliskan dengan *logic* sebagai berikut:

```
SELECT name, job
FROM user_
WHERE domisili = (SELECT domisili
FROM kode_daerah WHERE kode =
'C015');
```

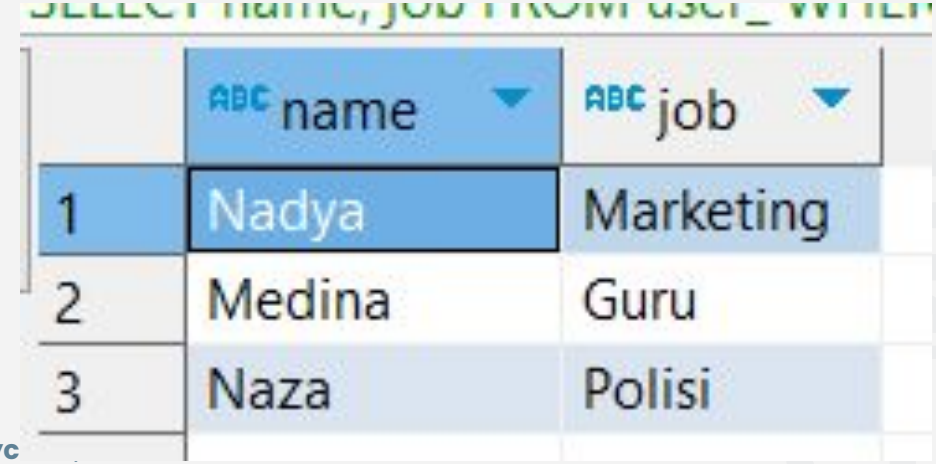
	name	job
1	Medina	Guru
2	Naza	Polisi

Multiple row subquery

Pada *single row subquery*, Kita akan membuat subquery yang menghasilkan data dalam bentuk beberapa baris untuk membandingkan data yang ada pada *query* utama dengan data yang dihasilkan dari *subquery*.

Sebagai contoh, Kita ingin mencari nama dan pekerjaan penduduk yang memiliki kode_daerah domisili adalah C015 atau C012. Dengan menggunakan contoh slide disebelumnya, *multiple subquery* dapat dituliskan dengan *logic* sebagai berikut:

```
SELECT name, job
FROM user_
WHERE domisili in (SELECT domisili FROM
kode_daerah WHERE (kode = 'C015' or kode =
'C012'));
```



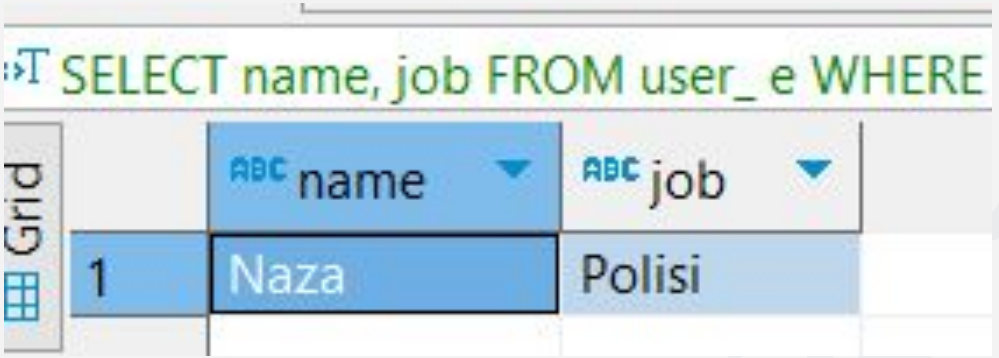
	name	job
1	Nadya	Marketing
2	Medina	Guru
3	Naza	Polisi

Correlated subquery

Pada *correlated subquery*, Kita akan membuat subquery yang hasilnya bergantung dengan query utama

Sebagai contoh, Kita ingin mencari nama dan pekerjaan penduduk yang memiliki domisili yang sama dan juga umur lebih besar dari umur penduduk di domisili yang sama. Dengan menggunakan contoh slide disebelumnya, *correlated subquery* dapat dituliskan dengan *logic* sebagai berikut:

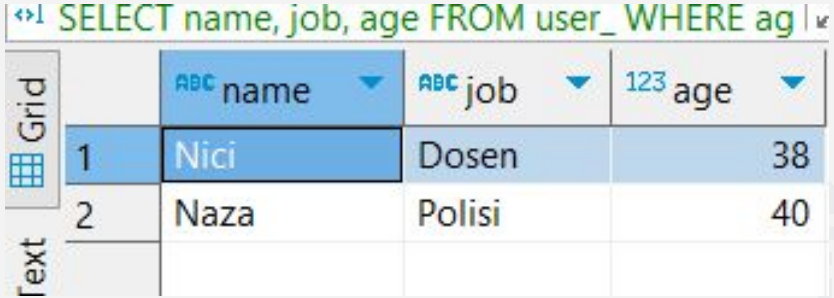
```
SELECT name, job
FROM user_ e
WHERE age >(SELECT AVG(age)
FROM user_ WHERE domisili =
e.domisili)
```



Nested subquery

Pada *nested subquery*, Kita akan membuat subquery dimana didalam subquery akan ada subquery lainnya. contoh, Kita ingin mencari nama, pekerjaan, umur penduduk yang memiliki domisili di Makasar dan juga umur lebih besar dari umur penduduk.. Dengan menggunakan contoh slide disebelumnya, *Nested subquery* dapat dituliskan dengan *logic* sebagai berikut:

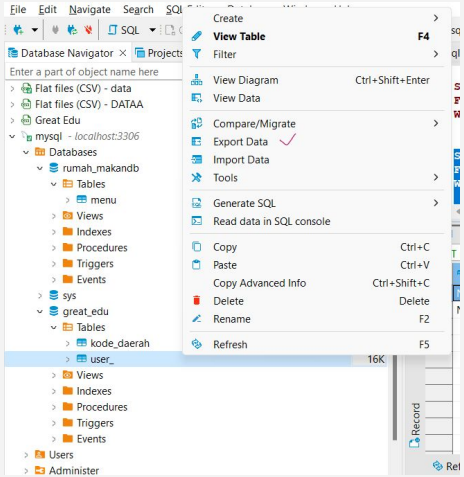
```
SELECT name, job, age
FROM user_
WHERE age>(SELECT AVG(age)
FROM (SELECT * FROM user_ WHERE
domisili = "Makasar") AS s)
```

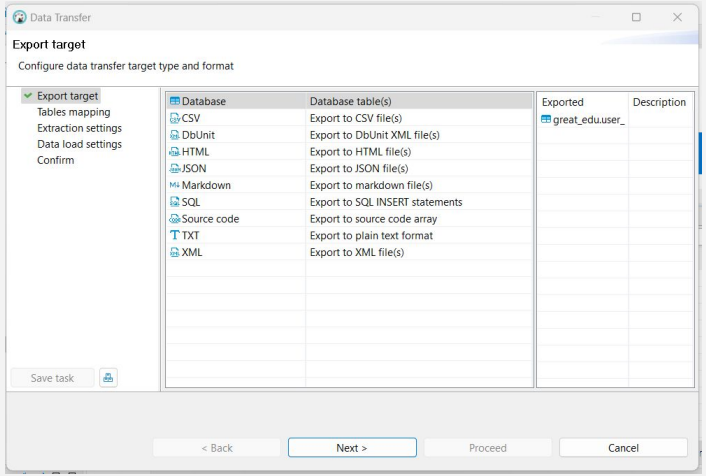


Export dan Import Data

Export Data

Contoh export data di Dbeaver. Kita dapat mentransfer data dari satu database ke database lain atau mengekspornya dalam tipe dan format berbeda seperti CSV, JSON, XML, dll. Caranya kita klik kanan dan pilih export data. Lalu kita pilih tipe dan format uang diinginkan. Lalu kita NEXT dan atur fungsi yang diinginkan. Terakhir klik proceed/start.



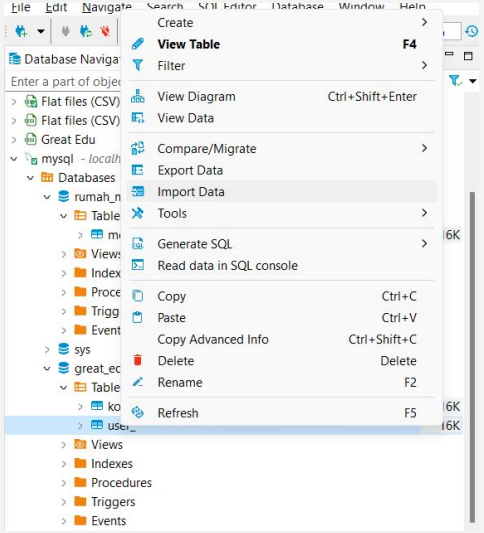


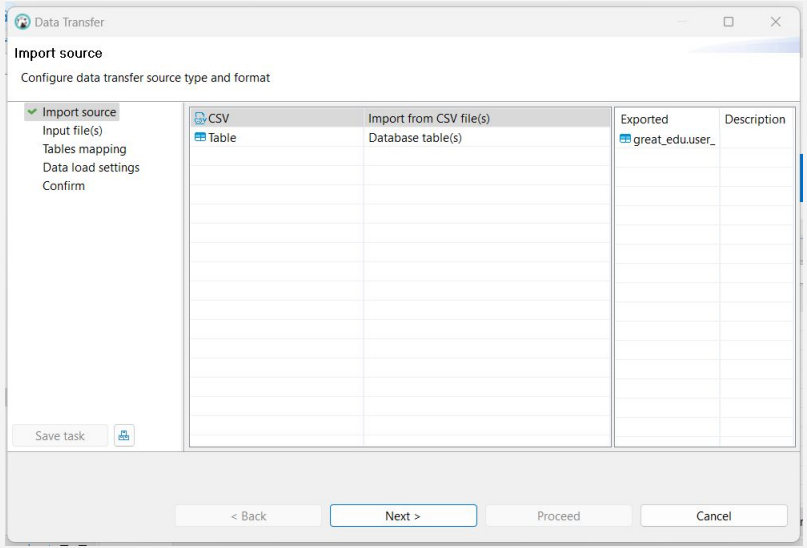
Export dan Import Data

Import Data

Contoh import data di Dbeaver. Kita dapat mengimpor data ke database dari file CSV, XLSX, dan XML.

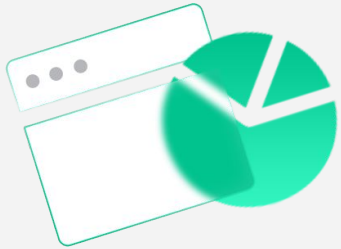
Caranya kita klik kanan dan pilih import data. Lalu kita pilih tipe dan format uang diinginkan. Lalu kita NEXT dan pilih data yang akan kita import setelah itu kita mapping tabel seperti value dalam kolom tersebut. Jika sudah start dan finish data suda terimport





NEXT!!

Git & Version Control System



Terima Kasih