

## PRAKTIKUM 6

### Materi:

Pengenalan SQL, DDL, dan DML

### Tujuan Praktikum:

Setelah mengikuti praktikum ini diharapkan praktikan memahami konsep mengenai tentang SQL, DDL, dan DML agar dapat dilakukan penerapannya dalam kasus tertentu.

**PENYAJIAN:** Konsep yang dijelaskan oleh asisten (30 menit)

### SQL

SQL (*Structured Query Language*) merupakan sebuah bahasa pemrograman yang khusus digunakan untuk melakukan manajemen data di dalam *database*. SQL (Structured Query Language) adalah salah satu bahasa generasi level ke-4 (4thGL) yang awalnya dikembangkan oleh IBM di San Jose Research Laboratory. Berbeda dengan bahasa pemrograman level ke-3 (3thGL), SQL adalah bahasa yang bersifat *request oriented* dan bersifat non-prosedural sehingga lebih mudah untuk dipelajari karena sintaks yang digunakan hampir menyerupai bahasa yang digunakan oleh manusia untuk berkomunikasi, sehingga SQL lebih fleksibel dalam penggunaannya. Selain itu SQL juga bersifat *non case sensitif*. Pada dasarnya SQL Server merupakan bahasa komputer standar yang ditetapkan oleh ANSI (American National Standard Institute) untuk mengakses dan memanipulasi data. Banyak vendor-vendor pembuat DBMS (Database Management System) saat ini. Pada Praktikum ini, kita menggunakan DBMS PostgreSQL.

Pernyataan SQL dapat dikelompokkan menjadi 3 kelompok, yaitu DDL, DML dan DCL.

1. **DDL (Data Definition Language)**, merupakan kelompok perintah yang berfungsi untuk mendefinisikan atribut-atribut basis data, tabel, atribut (kolom), batasan-batasan terhadap suatu atribut, serta hubungan antar tabel. Yang termasuk dalam kelompok DDL adalah CREATE, ALTER dan DROP.
2. **DML (Data Manipulation Language)**, yaitu bahasa yang berhubungan dengan proses manipulasi data pada tabel, record. Misal INSERT, UPDATE, DELETE.
3. **DCL (Data Control Language)**, yaitu bahasa yang berhubungan dengan pengendalian akses ke database. Misal GRANT, REVOKE

### Data Definition Language (DDL) di PostgreSQL

Data Definition Language, atau singkatan dari DDL yaitu perintah sql yang digunakan untuk membuat, memodifikasi struktur data pada object dalam Database. DDL dibagi menjadi 5 yaitu:

1. **CREATE**, digunakan untuk membuat object pada database seperti table, view, sequence, schema dan lain-lain.
2. **DROP**, digunakan untuk menghapus object pada database seperti table, view, sequence, schema dan lain-lain.
3. **ALTER**, digunakan untuk memodifikasi object pada database seperti menambahkan kolom pada table, menambahkan constraint pada kolom tertentu.
4. **TRUNCATE**, digunakan untuk menghapus semua record dalam suatu table.
5. **RENAME**, digunakan untuk mengubah nama database yang ada.

Contoh Penggunaan

### Create Database

Perintah **CREATE** pada database, digunakan untuk membuat database baru dengan perintah seperti berikut:

```
CREATE DATABASE your_database_name WITH OWNER your_user;
```

## Drop Database

Perintah **DROP** pada database, digunakan untuk menghapus database yang ada dengan perintah seperti berikut:

```
DROP DATABASE your_database_name;
```

## Create Table

Perintah DDL **Create** pada Table di database Postgresql, ininya adalah untuk membuat table dalam Schema di Database. Berikut adalah contoh perintanya:

```
CREATE TABLE [your_schema].your_table_name (  
    first_column <data-type> [constraints...] [default <value>],  
    second_column <data-type> [constraints...] [default <value>]  
);
```

Contoh penggunaanya:

```
create table test_table (  
    id                serial                primary key,  
    first_name        character varying(60) not null,  
    birtdate          date                  not null,  
    is_active          boolean              default false,  
    balance            decimal              not null    default 0,  
    created_datetime  datetime             not null    default now()  
);
```

## Drop Table

Perintah **DROP** table, digunakan untuk menghapus table yang ada di dalam schema di Database tertentu. Berikut adalah perintahnya:

```
DROP TABLE [IF EXISTS] your_table_name [CASCADE | RESTRICT];
```

## Alter Table

Perintah **ALTER** table, digunakan untuk memodifikasi struktur object pada table contohnya

1. Menambahkan kolom  
**ALTER TABLE** <table-name> **ADD COLUMN** <column-name> <data-type>
2. Menghapus kolom  
**ALTER TABLE** <table-name> **DROP COLUMN** <column-name>
3. Menambahkan constraint  
**ALTER TABLE** <table-name> **ADD** <constraint>

4. Menghapus constraint  
`ALTER TABLE <table-name> DROP CONSTRAINT <constraint-name>`
5. Mengubah default value  
`ALTER TABLE <table-name> ALTER COLUMN <column-name> SET  
 DEFAULT <new-default-value>;`
6. Mengubah tipe data  
`ALTER TABLE <table-name> ALTER COLUMN <column-name> TYPE  
 <data-type>;`
7. Mengubah nama kolom  
`ALTER TABLE <table-name> RENAME COLUMN <old-column-name> TO  
 <new-column-name>;`
8. Mengubah nama tabel  
`ALTER TABLE <old-table-name> RENAME TO <new-table-name>;`

### Data Manipulation Language (DML) di PostgreSQL

Data Manipulation Language terdiri dari perintah : SELECT, INSERT, UPDATE dan DELETE. Biasanya perintah DML dilakukan terhadap tabel atau view dalam database MySQL. Adapun penjelasan singkatnya adalah sebagai berikut :

1. **SELECT**, merupakan perintah yang digunakan untuk menampilkan data yang berasal dari tabel atau view.
2. **INSERT**, merupakan perintah yang digunakan untuk memasukkan data atau record ke dalam tabel.
3. **UPDATE**, merupakan perintah yang digunakan untuk memperbarui data atau record pada tabel.
4. **DELETE**, merupakan perintah yang digunakan untuk menghapus data atau record yang ada pada tabel.

### SELECT

Perintah/klausa `select` pada dasarnya adalah perintah yang dilakukan *query* terhadap *database* untuk mengambil atau mendapatkan data berupa baris (*rows*) dan kolom (*columns*) dalam sebuah *table*.

Jadi perintah *sql* `select` memiliki format sebagai berikut:

```

1
2  select
3      * | columns...
4  from
      table_name

```

### INSERT

Ketika tabel telah terbuat, biasanya tidak ada data yang tersedia di dalam tabel tersebut. Untuk mengisi datanya kita harus menjalankan perintah insert atau selain itu kita juga bisa menggunakan fitur backup/restore database, tpi fitur backup/restore tidak akan saya jelaskan di materi bootcamp sekarang. Jadi kita fokus ke materi dml dulu.

Format penulisan insert statement yaitu sebagai berikut:

```

INSERT INTO table_name (column1, column2, ...)
VALUES (<value1>, <value2>, <value3>, ...);

```

Contoh penggunaanya adalah sebagai berikut, contohnya saya menambahkan data baru ke table `regions` dengan data seperti berikut:

```
INSERT INTO regions (region_name) VALUES ('Asia Tenggara');
```

## UPDATE

Perintah update data, biasanya kita melakukan modifikasi data terhadap beberapa column dalam sebuah tabel di Database dengan / tanpa menggunakan where klausa.

Format penulisan update statement yaitu sebagai berikut:

```
UPDATE table_name SET
    column_update1 = <value1>,
    column_update2 = <value2>
WHERE column_key = <valueKey>
```

Contoh penggunaanya adalah sebagai berikut, contohnya saya mau update data pada table `regions` kolom `region_name` yang valuenya `Asia Tenggara` menjadi `Oceania`, maka berikut querynya:

```
UPDATE regions
SET region_name = 'Oceania'
WHERE region_id = 5;
```

## DELETE

Perintah delete data, biasanya kita gunakan untuk menghapus data dari table tertentu dengan / tanpa menggunakan where klausa.

Format penulisan delete statement yaitu seperti berikut:

```
DELETE FROM table_name
WHERE column_key = <valueKey>
```

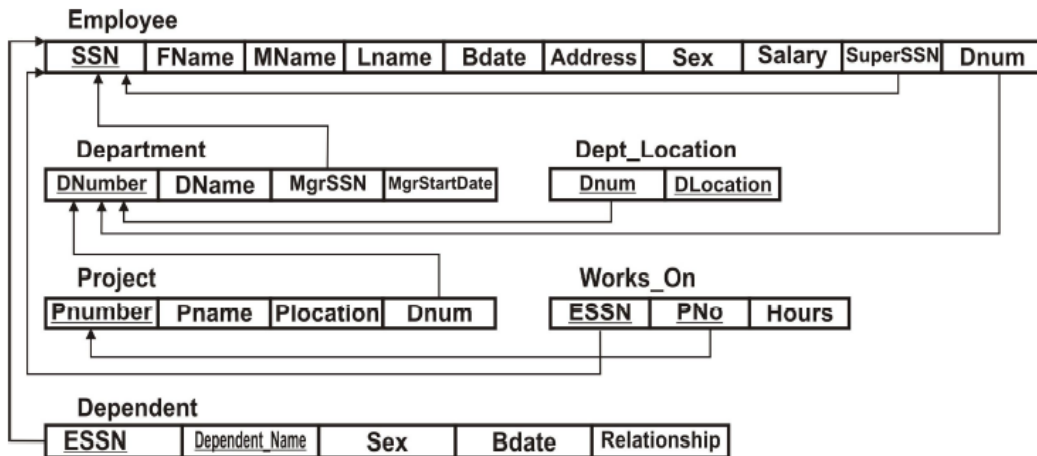
Contoh penggunaanya adalah sebagai berikut, contohnya saya mau menghapus data yang telah saya insert tadi, yaitu dengan `region_id = 5`, maka berikut querynya:

```
delete from regions
where region_id = 5;
```

## LATIHAN

Lakukan latihan dengan membuat database **Company** berikut:

### Database Company



#### ➤ Membuat database company dengan PSQL

- Klik Start > All Programs > PostgreSQL > SQL Shell
- Ketik \l, untuk melihat daftar database yang ada.
- Pastikan database Company belum ada dalam postgres seperti gambar di bawah ini:

```

SQL Shell (psql)
Server [localhost]:
Database [postgres]:
Port [5432]:
Username [postgres]:
Password for user postgres:
psql (9.0.3)
WARNING: Console code page (850) differs from Windows code page (1252)
8-bit characters might not work correctly. See psql reference
page "Notes for Windows users" for details.
Type "help" for help.

postgres=# \l
          List of databases
  Name | Owner | Encoding | Collation | Ctype
-----+-----+-----+-----+-----
 postgres | postgres | UTF8 | Indonesian, Indonesia | Indonesian, Indonesia
  project | postgres | UTF8 | Indonesian, Indonesia | Indonesian, Indonesia
   proyek | postgres | UTF8 | Indonesian, Indonesia | Indonesian, Indonesia
 template0 | postgres | UTF8 | Indonesian, Indonesia | Indonesian, Indonesia
 -c/postgres |  |  |  | 
  postgres=Ctc/postgres |  |  |  | 
 template1 | postgres | UTF8 | Indonesian, Indonesia | Indonesian, Indonesia
 -c/postgres |  |  |  | 
  postgres=Ctc/postgres |  |  |  | 
(5 rows)

postgres=#
  
```

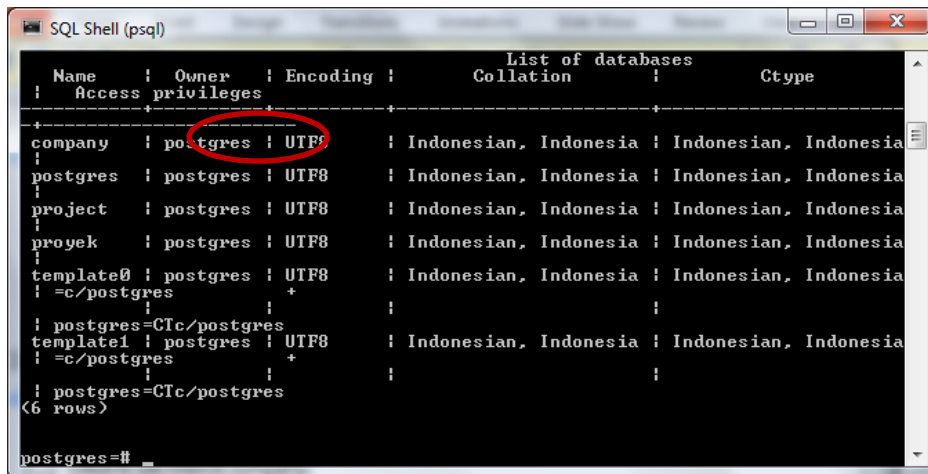
- Membuat database Company  
CREATE DATABASE company;

```

SQL Shell (psql)

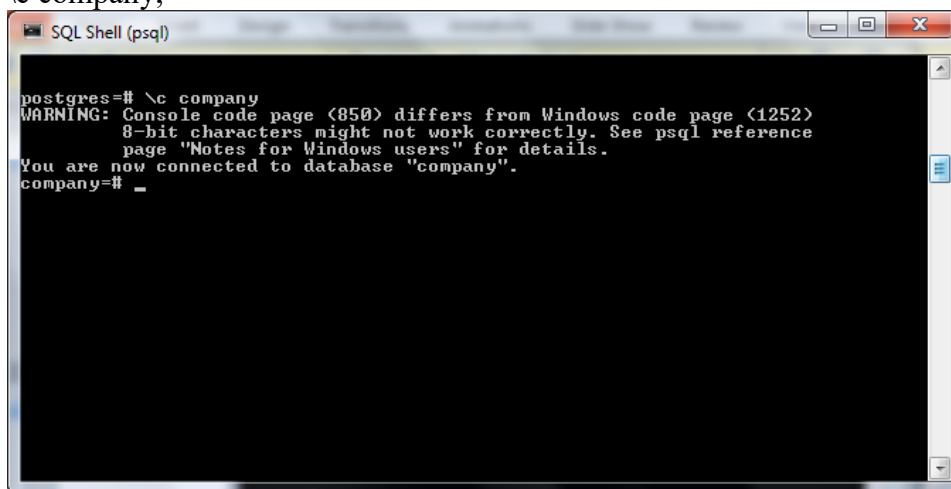
postgres=# CREATE DATABASE company;
CREATE DATABASE
postgres=#
  
```

Cek kembali dengan mengetikkan \l , untuk melihat apakah database Company sudah masuk ke PostgreSQL



- Masuk ke database Company

Sebelum membuat tabel, kita harus masuk dulu ke database COMPANY, dengan cara mengetikan:  
\c company;



- Membuat Tabel Employee
 

```

CREATE TABLE Employee(
SSN CHAR(9) NOT NULL,
FName VARCHAR(15) NOT NULL,
MName CHAR,
LName VARCHAR(15) NOT NULL,
BDate DATE,
Address VARCHAR(30),
Sex CHAR,
Salary DECIMAL(10,2),
SuperSSN CHAR(9),
DNum INT NOT NULL,
CONSTRAINT Employee_SSN_PK PRIMARY KEY(SSN),
CONSTRAINT Employee_DNum_FK FOREIGN KEY(DNum) REFERENCES
Department(DNumber),
CONSTRAINT Employee_SuperSSN_FK FOREIGN KEY(SuperSSN)
REFERENCES Employee(SSN));

```

```

company=# CREATE TABLE Employee(
company=#     SSN CHAR(9) NOT NULL,
company=#     FName VARCHAR(15) NOT NULL,
company=#     MName CHAR,
company=#     LName VARCHAR(15) NOT NULL,
company=#     BDate DATE,
company=#     Address VARCHAR(30),
company=#     Sex CHAR,
company=#     Salary DECIMAL(10,2),
company=#     SuperSSN CHAR(9),
company=#     DNum INT NOT NULL,
company=#     CONSTRAINT Employee_SSN_PK PRIMARY KEY(SSN),
company=#     CONSTRAINT Employee_DNum_FK FOREIGN KEY(DNum) REFERENCES Department
company=#     (DNumber),
company=#     CONSTRAINT Employee_SuperSSN_FK FOREIGN KEY(SuperSSN)
company=#     REFERENCES Employee(SSN));
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "employee_ssn_pk"
for table "employee"
ERROR: relation "department" does not exist
company=#
company=#

```

Error dikarenakan CONSTRAINT Employee\_Dnum\_FK FOREIGN KEY <DNum> REFERENCES Department <DNumber>, constraint ini tidak dapat dibuat karena tabel Department belum ada karena itu constraint tersebut dihilangkan terlebih dahulu.

Cek apakah tabel sudah dibuat atau belum dengan mengetikan \d

```

company=# CREATE TABLE Employee(
company=#     SSN CHAR(9) NOT NULL,
company=#     FName VARCHAR(15) NOT NULL,
company=#     MName CHAR,
company=#     LName VARCHAR(15) NOT NULL,
company=#     BDate DATE,
company=#     Address VARCHAR(30),
company=#     Sex CHAR,
company=#     Salary DECIMAL(10,2),
company=#     SuperSSN CHAR(9),
company=#     DNum INT NOT NULL,
company=#     CONSTRAINT Employee_SSN_PK PRIMARY KEY(SSN),
company=#     CONSTRAINT Employee_SuperSSN_FK FOREIGN KEY(SuperSSN)
company=#     REFERENCES Employee(SSN));
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "employee_ssn_pk"
for table "employee"
company=# \d
List of relations
Schema | Name      | Type  | Owner
-----+-----+-----+-----
public | employee | table | postgres
(1 row)

```

- Membuat Tabel Department

```

CREATE TABLE Department(
DNumber INT NOT NULL,
DName VARCHAR(15) NOT NULL,
MgrSSN CHAR(9) NOT NULL,
MgrStartDate DATE,
CONSTRAINT Dept_DNumber_PK PRIMARY KEY(DNumber),
CONSTRAINT Dept_DName_Unique UNIQUE(DName),
CONSTRAINT Dept_MgrSSN_FK FOREIGN KEY(MgrSSN) REFERENCES Employee(SSN));

```

- Add column

Setelah membuat tabel Departemen berarti kita dapat membuat CONSTRAINT atribut Dnum sebagai foreign key pada tabel Employee

```

ALTER TABLE Employee ADD CONSTRAINT Employee_DNum_FK FOREIGN KEY(DNum)
REFERENCES Department(DNumber);

```

```

SQL Shell (psql)
company=# CREATE TABLE Department(
company=#     DNumber INT NOT NULL,
company=#     DName VARCHAR(15) NOT NULL,
company=#     MgrSSN CHAR(9) NOT NULL,
company=#     MgrStartDate DATE,
company=#     CONSTRAINT Dept_DNumber_PK PRIMARY KEY(DNumber),
company=#     CONSTRAINT Dept_DName_Unique UNIQUE(DName),
company=#     CONSTRAINT Dept_MgrSSN_FK FOREIGN KEY(MgrSSN) REFERENCES Employee(
company=#     ESSN));
NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "dept_dnumber_pk"
for table "department"
NOTICE: CREATE TABLE / UNIQUE will create implicit index "dept_dname_unique" fo
r table "department"
CREATE TABLE
company=# ALTER TABLE Employee ADD CONSTRAINT Employee_DNum_FK FOREIGN KEY(DNum)
REFERENCES Department(DNumber);
ALTER TABLE
company=#

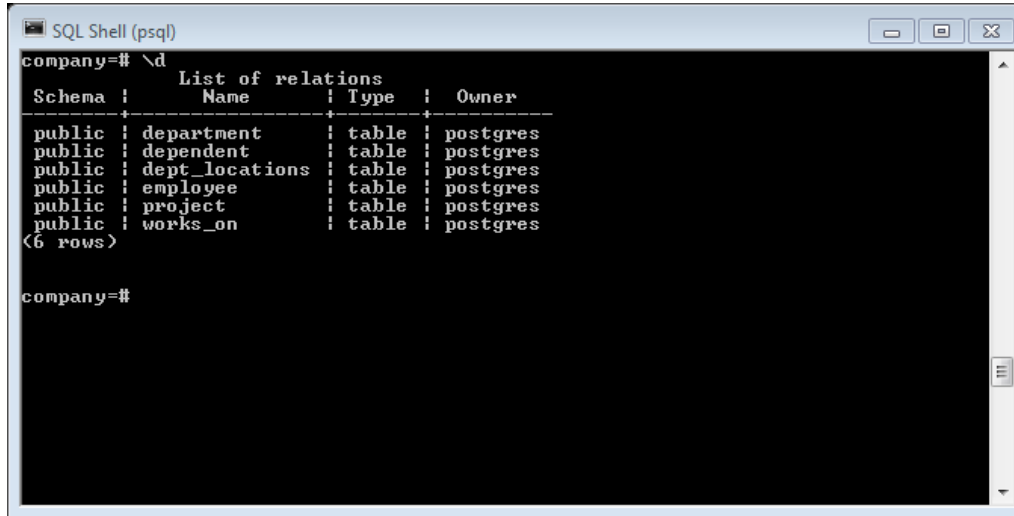
```

- Membuat tabel Dept\_Locations  
CREATE TABLE Dept\_Locations(  
DNum INT NOT NULL,  
DLocation VARCHAR(15) NOT NULL,  
CONSTRAINT DNumber\_DLocation\_PF PRIMARY KEY(DNum,DLocation),  
CONSTRAINT DLoc\_DNum\_FK FOREIGN KEY(DNum) REFERENCES  
Department(DNumber));
- Membuat tabel Project  
CREATE TABLE Project(  
PNumber INT NOT NULL,  
PName VARCHAR(15) NOT NULL,  
PLocation VARCHAR(15),  
DNum INT NOT NULL,  
CONSTRAINT Project\_PNumber\_PK PRIMARY KEY(PNumber),  
CONSTRAINT Project\_PName\_Unique UNIQUE(PName),  
CONSTRAINT Project\_DNum\_FK FOREIGN KEY(DNum)REFERENCES  
Department(DNumber));
- Membuat tabel Works\_on  
CREATE TABLE Works\_On(  
ESSN CHAR(9) NOT NULL,  
PNum INT NOT NULL,  
Hours DECIMAL(3,1) NOT NULL,  
CONSTRAINT Works\_ESSN\_PNum\_PK PRIMARY KEY(ESSN,PNum),  
CONSTRAINT Works\_ESSN\_FK FOREIGN KEY(ESSN) REFERENCES Employee(SSN),  
CONSTRAINT Works\_PNum\_FK FOREIGN KEY(PNum) REFERENCES Project(PNumber));
- Membuat tabel Dependent  
CREATE TABLE Dependent(  
ESSN CHAR(9) NOT NULL,  
Dependent\_Name VARCHAR(15) NOT NULL,  
Sex CHAR,  
BDate DATE,  
Relationship VARCHAR(8),  
CONSTRAINT Dependent\_ESSN\_DepName\_PK PRIMARY KEY(ESSN,Dependent\_Name),



CONSTRAINT Dependent\_ESSN\_FK FOREIGN KEY(ESSN) REFERENCES Employee(SSN));

- Mengecek semua daftar tabel dengan mengetikan \d



- Insert data ke dalam Tabel Employee
 

```

INSERT INTO Employee VALUES('E001', 'Hakim', null, 'Arifin', '12-Jan-1987','BATENG', 'M',
4000000, null, 1);
INSERT INTO Employee VALUES('E002','Yuni',null,'Arti','15-Feb-1987',
'BARA','F',4000000,null,2);
INSERT INTO Employee VALUES('E003','Mutia',null,'Aziza','23-Mar-1987',
'BATENG','F',4000000,null,3);
INSERT INTO Employee VALUES('E004','Hanif',null,'Affandi','21-Jan-1987',
'BARA','M',4000000,null,4);
INSERT INTO Employee VALUES('E005','Vera',null,'Yunita','16-May-1987',
'BALEBAK','F',3500000,'E001',1);
INSERT INTO Employee VALUES('E006','Pritasri',null,'Palupiningsih','09-Dec-1987',
'BADONENG','F',3500000,'E001',1);
INSERT INTO Employee VALUES('E007','Rifki','Y','Haidar','02-Aug-1987',
'BATENG','M',3000000,'E001',1);
INSERT INTO Employee VALUES('E008','Muhammad','A','Rosyidi','22-Jun-1987',
'PERUMDOS','M',2750000,'E001',1);
INSERT INTO Employee VALUES('E009','Ferry',null,'Pratama','11-Jul-1987',
'BARA','M',3000000,'E002',2);
INSERT INTO Employee VALUES('E010','Andi',null,'Sasmita','15-Feb-1987',
'BATENG','M',3000000,'E002',2);
INSERT INTO Employee VALUES('E011','Yuhan','A','Kusuma','16-Mar-1987',
'BARA','M',2500000,'E002',2);
INSERT INTO Employee VALUES('E012','Ferdian',null,'Feisal','23-Mar-1987',
'BATENG','M',2000000,'E002',2);
INSERT INTO Employee VALUES('E013','Albertus','A','M',22-May-1986',
'BARA','M',3000000,'E003',3);
INSERT INTO Employee VALUES('E014','Benedika','F','Hutabarat','21-Jun-1987',
'BADONENG','M',3250000,'E003',3);
INSERT INTO Employee VALUES('E015','Herbet',null,'Sianipar','16-Jul-1987',
'BARA','M',3750000,'E003',3);

```

```
ALTER TABLE Employee ADD CONSTRAINT Employee_DNum_FK FOREIGN KEY(DNum)
REFERENCES Department(DNumber);
```

Kemudian masukkan lagi insert data employee.

- Insert data ke tabel dept\_locations

```
INSERT INTO Dept_Locations VALUES(1,'Darmaga');
INSERT INTO Dept_Locations VALUES(3,'Darmaga');
INSERT INTO Dept_Locations VALUES(2,'Darmaga');
INSERT INTO Dept_Locations VALUES(4,'Baranang Siang');
```

- Insert data ke tabel project

```
INSERT INTO Project VALUES(1,'AAA','Bogor',1);
INSERT INTO Project VALUES(2,'BBB','Jakarta',2);
INSERT INTO Project VALUES(3,'CCC','Tangerang',2);
INSERT INTO Project VALUES(4,'DDD','Bekasi',2);
INSERT INTO Project VALUES(5,'EEE','Depok',3);
INSERT INTO Project VALUES(6,'FFF','Bogor',3);
INSERT INTO Project VALUES(7,'GGG','Tangerang',4);
INSERT INTO Project VALUES(8,'HHH','Jakarta',4);
```

- Insert data works\_on

```
INSERT INTO Works_On VALUES('E001',1,90);
INSERT INTO Works_On VALUES('E001',2,98);
INSERT INTO Works_On VALUES('E002',2,55);
INSERT INTO Works_On VALUES('E002',3,78);
INSERT INTO Works_On VALUES('E003',3,53);
INSERT INTO Works_On VALUES('E003',4,77);
INSERT INTO Works_On VALUES('E004',4,77);
INSERT INTO Works_On VALUES('E004',5,98);
INSERT INTO Works_On VALUES('E004',7,85);
INSERT INTO Works_On VALUES('E004',8,68);
INSERT INTO Works_On VALUES('E005',5,57);
INSERT INTO Works_On VALUES('E005',6,87);
INSERT INTO Works_On VALUES('E006',7,45);
INSERT INTO Works_On VALUES('E006',6,87);
INSERT INTO Works_On VALUES('E007',7,40);
INSERT INTO Works_On VALUES('E007',8,88);
INSERT INTO Works_On VALUES('E008',1,78);
INSERT INTO Works_On VALUES('E008',8,87);
INSERT INTO Works_On VALUES('E009',1,88);
INSERT INTO Works_On VALUES('E009',2,65);
INSERT INTO Works_On VALUES('E010',2,34);
INSERT INTO Works_On VALUES('E010',3,78);
INSERT INTO Works_On VALUES('E011',1,68);
INSERT INTO Works_On VALUES('E011',3,88);
```

- Insert data dependent

```
INSERT INTO Dependent VALUES('E001','Rita','F','18-Sep-2005','DAUGHTER');
INSERT INTO Dependent VALUES('E001','Doni','M','09-Jan-2007','SON');
INSERT INTO Dependent VALUES('E002','Wawan','M','23-Oct-1984','HUSBAND');
INSERT INTO Dependent VALUES('E002','Roy','M','15-Dec-2006','SON');
INSERT INTO Dependent VALUES('E003','Roni','M','23-AUG-1985','HUSBAND');
INSERT INTO Dependent VALUES('E003','Dewi','F','01-Jan-2006','DAUGHTER');
INSERT INTO Dependent VALUES('E004','Susi','F','05-Sep-1987','WIFE');
INSERT INTO Dependent VALUES('E004','Rani','M','10-Feb-2007','DAUGHTER');
```

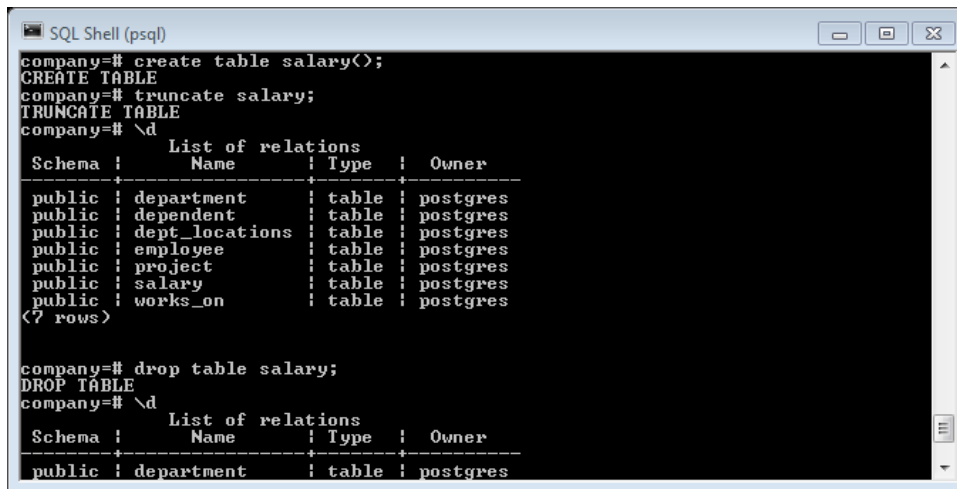
```

INSERT INTO Dependent VALUES('E011','Dina','F','13-Jan-1987','WIFE');
INSERT INTO Dependent VALUES('E011','Riko','M','21-Mar-2006','SON');
INSERT INTO Dependent VALUES('E013','Rini','F','15-Aug-1987','WIFE');
INSERT INTO Dependent VALUES('E013','Tina','F','17-Dec-2005','DAUGHTER');
INSERT INTO Dependent VALUES('E014','Ayu','F','08-Dec-1988','WIFE');
INSERT INTO Dependent VALUES('E014','Didiet','M','05-Dec-2006','SON');
INSERT INTO Dependent VALUES('E020','Nita','F','25-Jan-1987','WIFE');

```

- Truncate table

Perintah truncate merupakan perintah “menghapus data”. Namun kenyataannya perintah truncate tidak menghapus data, tetapi dia akan mengalokasikan semua data page dan menghapus pointer yang menuju ke indexnya.



```

company=# create table salary();
CREATE TABLE
company=# truncate salary;
TRUNCATE TABLE
company=# \d

```

Schema	Name	Type	Owner
public	department	table	postgres
public	dependent	table	postgres
public	dept_locations	table	postgres
public	employee	table	postgres
public	project	table	postgres
public	salary	table	postgres
public	works_on	table	postgres

```

<7 rows>

company=# drop table salary;
DROP TABLE
company=# \d

```

Schema	Name	Type	Owner
public	department	table	postgres

- Update

```
UPDATE Employee SET Salary = 5000000 WHERE Address= 'BARA';
```

Coba lihat lagi :

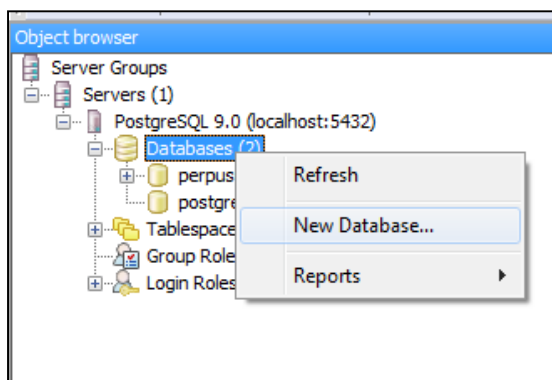
```
SELECT * FROM Employee;
```

- Delete

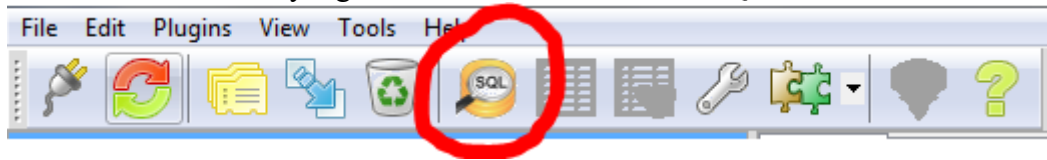
```
Delete FROM Employee WHERE Address='PERUMDOS';
```

➤ **Membuat database menggunakan pgAdmin**

1. Buka pgAdmin
2. Klik kanan di icon database pada object browser, pilih new database lalu beri nama



3. Pilih icon database yang sudah dibuat lalu klik icon SQL.



4. Tuliskan query seperti yang dilakukan pada PSQL. Ingat, pembuatan database dengan nama yang sudah ada tidak diperkenankan. Setelah selesai tekan F5 untuk eksekusi baris query.

### Data Manipulation Language (DML)

Data Manipulation Language (DML) merupakan instruksi SQL untuk mengatur dan memanipulasi data pada database. Beberapa instruksi yang digunakan dalam DML diantaranya:

**LIKE Comparison**, untuk perbandingan dari karakter atau kata tertentu.

**LIKE 'A%'** : dimulai dengan karakter 'A'

**LIKE '%A%'** : terdiri dari karakter 'A'

**LIKE '\_A%'** : ada karakter 'A' di posisi kedua

**LIKE 'A%i%'** : dimulai dengan karakter 'A' dengan mengandung karakter 'i'

**LIKE 'A%i%o'** : dimulai dengan karakter 'A', mengandung karakter 'i', lalu karakter 'o'

**NOT LIKE 'A%'** : dimulai bukan dengan karakter 'A'

Contoh query:

- Menentukan Employee yang memiliki FName diawali huruf 'H':

```
SELECT SSN, FName, LName, BDate FROM Employee
WHERE FName LIKE 'H%';
```

```
company=# SELECT SSN, FName, LName, BDate FROM Employee
company=# WHERE FName LIKE 'H%';
  ssn      |  fname  |  lname  |  bdate
-----+-----+-----+-----
 E001      |  Hakim  |  Arifin | 1987-01-12
 E004      |  Hanif  |  Affandi | 1987-01-21
 E015      |  Herbert |  Sianipar | 1987-07-16
(3 rows)
```

• **CASE**, untuk menentukan conditional dari hasil query. Contoh query:

- Mengelompokkan level Employee berdasarkan Salary (Jika > 3500000, maka Level = 'Top'; Selainnya Level = 'Middle'):

```
SELECT SSN, FName, Salary,
CASE
WHEN Salary > 3500000 THEN 'Top'
ELSE 'Middle'
END AS Level
FROM Employee
ORDER BY SSN;
```

```

company=# SELECT SSN, FName, Salary,
company=#         CASE
company=#         WHEN Salary > 3500000 THEN 'Top'
company=#         ELSE 'Middle'
company=#         END AS Level
company=# FROM Employee
company=# ORDER BY SSN;

```

ssn	fname	salary	level
E001	Hakim	4000000.00	Top
E002	Yuni	4000000.00	Top
E003	Mutia	4000000.00	Top
E004	Hanif	4000000.00	Top
E005	Vera	3500000.00	Middle
E006	Pritasari	3500000.00	Middle
E007	Rifki	3000000.00	Middle
E008	Muhammad	2750000.00	Middle
E009	Ferry	3000000.00	Middle
E010	Andi	3000000.00	Middle
E011	Yuhan	2500000.00	Middle
E012	Perdian	2000000.00	Middle
E013	Albertus	3000000.00	Middle
E014	Benedika	3250000.00	Middle
E015	Herbert	3750000.00	Top

<15 rows>

- **Aggregate Function**, digunakan untuk melakukan query fungsi operasi dasar dari kumpulan data tertentu. Jenis fungsi yang digunakan diantaranya:

**SUM(nama\_kolom)** : total dari nilai kolom

**MAX(nama\_kolom)** : nilai terbesar dari suatu kolom

**MIN(nama\_kolom)** : nilai terkecil dari suatu kolom

**AVG(nama\_kolom)** : nilai rata-rata dari suatu kolom

**COUNT(\*)** : banyaknya nilai dari tabel yang tidak berulang

**COUNT(nama\_kolom)** : banyaknya nilai dari suatu kolom yang tidak berulang

Contoh query :

- Menentukan Salary terbesar, Salary terkecil, dan Salary rata-rata dari Employee :

```
SELECT MAX(Salary), MIN(Salary), AVG(Salary) FROM Employee;
```

```

company=# SELECT MAX(Salary), MIN(Salary), AVG(Salary) FROM Employee;

```

max	min	avg
4000000.00	2000000.00	3283333.3333333333

<1 row>

- Menentukan banyaknya Employee yang beralamat di BARA:

```
SELECT COUNT(SSN) AS Jumlah_Employee_di_BARA FROM Employee WHERE Address = 'BARA';
```

```

company=# SELECT COUNT(SSN) AS Jumlah_Employee_di_BARA FROM Employee
company=#         WHERE Address = 'BARA';

```

jumlah_employee_di_bara
6

<1 row>

- Menentukan jumlah lokasi project (tanpa pengulangan dan dengan pengulangan) :

```
SELECT COUNT(DISTINCT PLocation) AS "distinct lokasi", COUNT(PLocation) AS "total lokasi" FROM Project;
```

```
company=# SELECT COUNT(DISTINCT PLocation) AS "distinct_lokasi",
company=#         COUNT(PLocation) AS "total_lokasi"
company=#         FROM Project;
distinct_lokasi | total_lokasi
-----+-----
5              | 8
(1 row)
```

- **GROUP BY**, untuk mengelompokkan aggregate functions berdasarkan kolom tertentu.
- **ORDER BY**, untuk mengurutkan hasil query berdasarkan kolom tertentu. Ascending (ASC) mengurutkan data dari yang terkecil. Descending (DESC) mengurutkan data dari yang terbesar.

Contoh query:

- Menentukan Salary terkecil, Salary terbesar, dan Salary rata-rata dari tiap Department diurutkan dari Salary terbesar:

```
SELECT Dnum, Min(Salary), MAX(Salary), AVG(Salary) FROM Employee
GROUP BY DNum
ORDER BY DNum DESC;
```

```
company=# SELECT DNum, MIN(Salary), MAX(Salary), AUG(Salary) FROM Employee
company=#         GROUP BY DNum
company=#         ORDER BY DNum DESC;
 dnum |      min      |      max      |      avg
-----+-----+-----+-----
  4   | 4000000.00    | 4000000.00    | 4000000.000000000000
  3   | 3000000.00    | 4000000.00    | 3500000.000000000000
  2   | 2000000.00    | 4000000.00    | 2900000.000000000000
  1   | 2750000.00    | 4000000.00    | 3350000.000000000000
(4 rows)
```

- **HAVING**, untuk melakukan query pada kolom yang memiliki nilai tertentu.

Contoh query:

- Menentukan Address yang memiliki lebih dari empat orang Employee:

```
SELECT Address, COUNT(*) FROM Employee
GROUP BY Address
HAVING COUNT(*) > 4
ORDER BY Address;
```

```
company=# SELECT Address, COUNT(*)
company=# FROM Employee
company=# GROUP BY Address
company=# HAVING COUNT(*) > 4
company=# ORDER BY Address;
 address | count
-----+-----
  BARA   | 6
  BATENG | 5
(2 rows)
```

- **Magic Variables**

**CURRENT\_DATE**, untuk menentukan tgl terbaru

**CURRENT\_TIME**, untuk menentukan waktu terbaru

**CURRENT\_TIMESTAMP**, untuk memberikan penanda waktu terbaru

**CURRENT\_USER**, untuk memberikan user database terbaru

Contoh query :

- Menentukan user dan waktu terakhir :

```
SELECT CURRENT_USER, CURENT_DATE, CURRENT_TIME,
CURRENT_TIMESTAMP;
```

```
company=# SELECT CURRENT_USER, CURRENT_DATE, CURRENT_TIME, CURRENT_TIMESTAMP;
current_user |      date      |      timetz      |      now
-----+-----+-----+-----
postgres    | 2012-03-19    | 15:39:51.234+07 | 2012-03-19 15:39:51.234+07
(1 row)
```

- Menentukan usia Employee (berdasarkan tanggal lahirnya)

```
SELECT SSN, FName, Age(CURRENT_DATE, BDate) AS Age FROM Employee
ORDER BY Age ASC;
```

```
company=# SELECT SSN, FName, Age(CURRENT_DATE, BDate) AS Age FROM Employee
company=# ORDER BY Age ASC;
ssn      |  fname  |      age
-----+-----+-----
E006     | Pritasari | 24 years 3 mons 10 days
E007     | Rifki    | 24 years 7 mons 17 days
E015     | Herbert  | 24 years 8 mons 3 days
E009     | Ferry    | 24 years 8 mons 8 days
E008     | Muhammad | 24 years 8 mons 27 days
E014     | Benedika | 24 years 8 mons 28 days
E013     | Albertus | 24 years 9 mons 28 days
E005     | Vera     | 24 years 10 mons 3 days
E003     | Mutia    | 24 years 11 mons 27 days
E012     | Ferdian  | 24 years 11 mons 27 days
E011     | Yuhan    | 25 years 3 days
E010     | Andi     | 25 years 1 mon 4 days
E002     | Yuni     | 25 years 1 mon 4 days
E004     | Hanif    | 25 years 1 mon 29 days
E001     | Hakim    | 25 years 2 mons 7 days
(15 rows)
```



**LEMBAR KERJA PRAKTIKUM (45 menit)**

Nama:	Tanggal Praktikum:
NRP :	Waktu Praktikum:
Nilai :	Nama Asisten :

**Soal**

1. PT Melodi Indah menyimpan informasi pemusik yang rekaman di perusahaan ini dalam sebuah **Database Company**.

- Setiap musisi mempunyai SSN, nama, alamat dan nomor telepon. Beberapa musisi mempunyai alamat yang sama dan beberapa tidak mencantumkan alamat.
- Setiap instrumen yang digunakan mempunyai nama (gitar, flute, dll) dan kunci musik (C, B#, dll).
- Setiap album yang direkam di perusahaan ini diberi label yang berisi, tanggal copyright, format (CD, MC), id album.
- Setiap lagu yang direkam mempunyai judul dan pengarangnya.
- Setiap musisi mungkin memainkan beberapa instrumen dan instrumen juga dapat dimainkan beberapa musisi.
- Setiap album mempunyai banyak lagu di dalamnya namun tidak ada lagu yang ada di banyak album.
- Setiap lagu ditampilkan oleh satu atau lebih musisi tapi musisi juga bisa menampilkan lebih dari satu lagu.
- Setiap album tentu saja mempunyai musisi sebagai produsernya dan seorang musisi tentu saja dapat memproduksi lebih dari satu album. Buat dalam SQL dan sertakan *screen shot* nya!

3. Dari **Database Company** lakukan query sebagai berikut:

- a) Tampilkan dependent\_name dan relationship dengan employee yang namanya diawali huruf R ?
- b) Banyaknya employee yang mengerjakan project PNum = 1 ?
- c) Banyaknya employee yang memiliki salary lebih dari 3500000 ?
- d) Banyaknya project yang dikerjakan DNum = 2 ?
- e) Hitung total dan rata-rata salary dari setiap departemen ?
- f) Banyaknya employee dari setiap department dan urutkan berdasarkan employee terbanyak ?
- g) Total hours perweek dari semua employee untuk setiap project ?
- h) Employee yang memiliki total hours perweek lebih besar dari 140 hours dan urutkan berdasarkan jumlah jam kerja terbanyak?
- i) Kelompokkan bonus employee berdasarkan jumlah jam kerjanya ? (Jika  $\geq 200$  hours, maka bonus = 50%; Jika  $\geq 150$  hours, maka bonus = 25%, Selainnya bonus = 10%)
- j) Banyaknya project yang dikerjakan tiap employee dan urutkan dari yang terbanyak ?
- k) Employee yang bekerja pada 4 project ?
- l) Employee yang memiliki rata-rata hours perweek = 70 jam dan bekerja pada 2 project ?
- m) Banyaknya Dependent berdasarkan relationship dengan employee ?
- n) Berapa lama Manager tiap Department sudah menjabat ?
- o) Lokasi project yang menjadi tempat lebih dari satu department?

