

# LEARNING PROGRESS REVIEW

## WEEK 2

Data Science  
Batch 11

By Omicron

# OMICRON

**Anggota Kelompok 3 :**

**Anugrah Yazid Ghani**

**Fajar Achmad**

**Muhammad Fikri Fadila**

**Edo Mohammad Hadad Gibran**

# *Introduction to Data and Database*

# Data

- Data adalah **informasi**, biasanya **fakta** dan **angka**, yang dikumpulkan untuk diuji dan berguna dalam membuat keputusan, atau informasi dalam bentuk elektronik yang terdapat dalam komputer.
- Klasifikasi Data :

Categorical Data		Numerical Data	
<b>Nominal</b> (klasifikasi) berupa <i>gender</i> , kota, nama	<b>Ordinal</b> (klasifikasi berurutan) berupa pendidikan, <i>ranking</i>	<b>Discrete</b> (jumlah) seperti total anak	<b>Continuous</b> (ukuran) seperti interval (suhu), rasio (tinggi, berat)

# Istilah Dalam Teknologi :

## **DBMS (*Database Management System*)**

Perangkat lunak yang dirancang untuk menyimpan, mengambil, mendefinisikan, dan mengelola data dalam *database*

## **SQL (*Structured Query Language*)**

Bahasa khusus domain yang digunakan dalam pemrograman dan dirancang untuk mengelola data yang digunakan dalam sistem manajemen basis data relasional

## **DDL (*Data Definition Language*)**

Perintah SQL untuk mendefinisikan skema

Contoh : **CREATE, DROP, ALTER, TRUNCATE, COMMENT, RENAME**

## **DML (*Data Manipulation Language*)**

Perintah SQL untuk menanyakan informasi, mengisi data, mengubah data, memperbarui data, atau menghapus data dari *database*

Contoh : **SELECT, INSERT, UPDATE, DELETE**

# ***Database***

Merupakan kumpulan data terorganisir yang disimpan dan diakses secara elektronik.

Keunggulan :

1. Mengurangi redundansi data.
2. Integritas data yang lebih besar dan memastikan konsistensi data.
3. Peningkatan akses data ke pengguna melalui penggunaan bahasa *host* dan bahasa *query*.
4. Memberikan keamanan dan privasi data yang lebih besar.
5. Pencadangan dan pemulihan yang kuat.

# Database

## Structured

### Kelebihan :

- › Mudah diimplementasikan pada ML
- › Mudah digunakan oleh pebisnis/pengguna
- › Cocok dengan banyak *tools*

### Kekurangan :

- › Terbatas hanya untuk basis data relasional

*Tools* : MySQL, MS SQL, PostgreSQL, SQLite, dll.

## Unstructured

### Kelebihan :

- › Format tetap original
- › Waktu proses lebih cepat
- › Tempat penyimpanan pada data *lake* lebih hemat
- › Butuh kemampuan teknis khusus

### Kekurangan :

- › Butuh alat khusus

*Tools* : MongoDB, DynamoDB, Hadoop, dll

# Tipe Data

Numeric	Date/time	Character/String	Unicode Character	Binary	Miscellaneous
int	date	char	nchar	binary	clob
bigint	time	varchar	nvarchar	varbinary	blob
smallint	datetime	text	ntext		xml
float	timestamp				json
decimal	year				
real					
bit					





# Model Data



## Relationship Cardinality

- *one to one* (1:1)
- *one to many* (1:M)
- *many to many* (M:M)

## Participant Constraint

- *Mandatory* : Minimal ada satu *entity* yang terasosiasi dengan *entity* yang lain
- *Optional* : Diperbolehkan tidak ada *entity* yang saling berasosiasi

## Tipe *Entity*

- *Strong* : Berdiri sendiri
- *Lemah* : Butuh yang lainnya
- *Asosiatif* : Dibuat oleh *entity* yang lain



# *Basic* SQL

# SQL *Introduction*

**Structured Query Language** adalah bahasa yang digunakan untuk berinteraksi dengan *database* dan merupakan bahasa *programming* yang paling dasar. Bahasa ini memberi akses untuk menangani informasi menggunakan *table* dan hal-hal yang terkait didalamnya. Beberapa *tools* yang digunakan untuk mengakses SQL, yaitu :



# SCHEMA

Merupakan lapisan pertama dalam *database* atau sekumpulan *table*, yaitu seperangkat formula/kalimat sebagai batasan integritas *database*.

*Syntax* untuk pembuatan **SCHEMA** adalah sebagai berikut:

```
create schema [if not exists] schema_name
```

# TABLE

*Table* terletak tepat dibawah **SCHEMA** dan dibuat untuk menyimpan data dalam format tabular. Merupakan objek *database* yang berisi semua data dalam *database*, yang diatur secara logis dalam format baris dan kolom. Setiap baris mewakili catatan *unique* dan setiap kolom mewakili bidang catatan.

Syntax untuk membuat **TABLE** :

```
• create table [if not exists] table_name (
    column1 datatype (Length) column_constraint,
    column2 datatype (Length) column_constraint,
    column3 datatype (Length) column_constraint,
    .....
    columnN datatype (Length) column_constraint,
    table_constraints
    -- PRIMARY KEY (column1)
    -- FOREIGN KEY (column2)
    references table_name(column_name)
)
```

# CREATE TABLE

Membuat **TABLE** baru

bernama **employee\_omicron**

dengan kolom:

- employee\_id
- first\_name
- last\_name
- email
- phone\_number
- hire\_date
- job\_id
- salary
- commision\_pct
- manager\_id
- department\_id



INPUT

```
create table batch_11.employee_omicron
(
  employee_id int4 not null,
  first_name varchar(255) not null,
  last_name varchar(255) not null,
  email varchar(255) not null,
  phone_number varchar(255) unique,
  hire_date varchar(255) not null,
  job_id varchar(255) not null,
  salary float4 not null,
  commission_pct float4 not null,
  manager_id int4 not null,
  department_id int4 not null
)
```

## CREATE TABLE

Output:

The screenshot shows the Oracle SQL Developer interface. The top toolbar includes buttons for Script, Properties, Data, and ER Diagram. The main window displays the 'employee\_omicon' table structure. The columns are: employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, manager\_id, and department\_id. The 'employee\_id' column is highlighted.

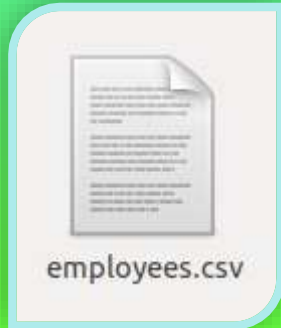
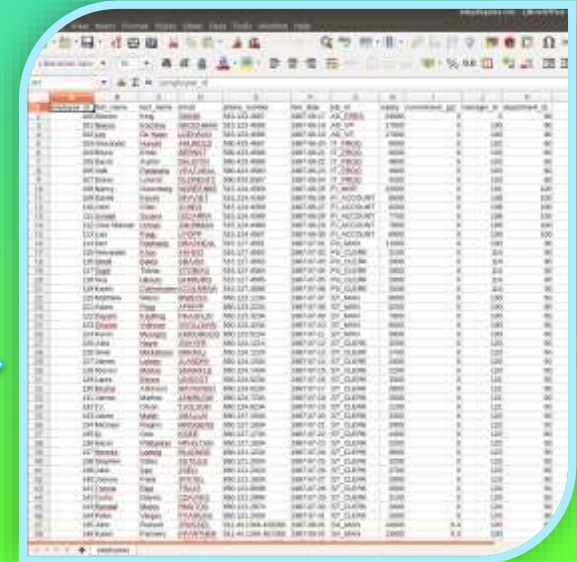
TABLE employee omicron sudah ter-CREATE dengan data masih kosong !



# IMPORT DATA

Fungsi yang digunakan untuk memasukan data dari *directory* lokal kedalam *database*. Dapat menggunakan *TABLE* yang sudah tersedia atau membuat *TABLE* baru. Format yang bisa digunakan adalah *Comma Separated Value* (CSV).

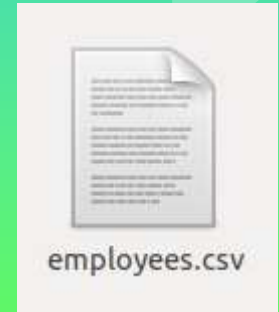
**Contoh:** memasukkan data **employees.csv** ke dalam *TABLE* **employee\_omicon** yang sudah dibuat.

id	nama	email	phone_number	hire_date	job_id	salary	commission_pct	manager_id	department_id
1	John Deere	john.deere@company.com	555 123 4567	1987-08-01	AD_VP	21000	0	9900	90
2	Neena Kocher	neena.kocher@company.com	555 123 4568	1987-08-01	AD_VP	17000	0	9900	90
3	Lex De Haan	lex.dehaan@company.com	555 123 4569	1987-08-01	AD_VP	17000	0	9900	90
4	Alexander Khoo	alex.khoo@company.com	555 123 4570	1987-08-01	AD_VP	17000	0	9900	90
5	Baerenson	baerenson@company.com	555 123 4571	1987-08-01	AD_VP	17000	0	9900	90
6	Kit	kit@company.com	555 123 4572	1987-08-01	AD_VP	17000	0	9900	90
7	Timothy Gietz	timothy.gietz@company.com	555 123 4573	1987-08-01	AD_VP	17000	0	9900	90
8	David	david@company.com	555 123 4574	1987-08-01	AD_VP	17000	0	9900	90
9	John	john@company.com	555 123 4575	1987-08-01	AD_VP	17000	0	9900	90
10	Paul	paul@company.com	555 123 4576	1987-08-01	AD_VP	17000	0	9900	90
11	Christina	christina@company.com	555 123 4577	1987-08-01	AD_VP	17000	0	9900	90
12	Mark	mark@company.com	555 123 4578	1987-08-01	AD_VP	17000	0	9900	90
13	David	david@company.com	555 123 4579	1987-08-01	AD_VP	17000	0	9900	90
14	Neena	neena@company.com	555 123 4580	1987-08-01	AD_VP	17000	0	9900	90
15	Lex	lex@company.com	555 123 4581	1987-08-01	AD_VP	17000	0	9900	90
16	Alexander	alex@company.com	555 123 4582	1987-08-01	AD_VP	17000	0	9900	90
17	Baerenson	baerenson@company.com	555 123 4583	1987-08-01	AD_VP	17000	0	9900	90
18	Kit	kit@company.com	555 123 4584	1987-08-01	AD_VP	17000	0	9900	90
19	Timothy	timothy@company.com	555 123 4585	1987-08-01	AD_VP	17000	0	9900	90
20	David	david@company.com	555 123 4586	1987-08-01	AD_VP	17000	0	9900	90
21	John	john@company.com	555 123 4587	1987-08-01	AD_VP	17000	0	9900	90
22	Paul	paul@company.com	555 123 4588	1987-08-01	AD_VP	17000	0	9900	90
23	Christina	christina@company.com	555 123 4589	1987-08-01	AD_VP	17000	0	9900	90
24	Mark	mark@company.com	555 123 4590	1987-08-01	AD_VP	17000	0	9900	90
25	David	david@company.com	555 123 4591	1987-08-01	AD_VP	17000	0	9900	90
26	Neena	neena@company.com	555 123 4592	1987-08-01	AD_VP	17000	0	9900	90
27	Lex	lex@company.com	555 123 4593	1987-08-01	AD_VP	17000	0	9900	90
28	Alexander	alex@company.com	555 123 4594	1987-08-01	AD_VP	17000	0	9900	90
29	Baerenson	baerenson@company.com	555 123 4595	1987-08-01	AD_VP	17000	0	9900	90
30	Kit	kit@company.com	555 123 4596	1987-08-01	AD_VP	17000	0	9900	90
31	Timothy	timothy@company.com	555 123 4597	1987-08-01	AD_VP	17000	0	9900	90
32	David	david@company.com	555 123 4598	1987-08-01	AD_VP	17000	0	9900	90
33	John	john@company.com	555 123 4599	1987-08-01	AD_VP	17000	0	9900	90
34	Paul	paul@company.com	555 123 4600	1987-08-01	AD_VP	17000	0	9900	90
35	Christina	christina@company.com	555 123 4601	1987-08-01	AD_VP	17000	0	9900	90
36	Mark	mark@company.com	555 123 4602	1987-08-01	AD_VP	17000	0	9900	90
37	David	david@company.com	555 123 4603	1987-08-01	AD_VP	17000	0	9900	90
38	Neena	neena@company.com	555 123 4604	1987-08-01	AD_VP	17000	0	9900	90
39	Lex	lex@company.com	555 123 4605	1987-08-01	AD_VP	17000	0	9900	90
40	Alexander	alex@company.com	555 123 4606	1987-08-01	AD_VP	17000	0	9900	90
41	Baerenson	baerenson@company.com	555 123 4607	1987-08-01	AD_VP	17000	0	9900	90
42	Kit	kit@company.com	555 123 4608	1987-08-01	AD_VP	17000	0	9900	90
43	Timothy	timothy@company.com	555 123 4609	1987-08-01	AD_VP	17000	0	9900	90
44	David	david@company.com	555 123 4610	1987-08-01	AD_VP	17000	0	9900	90
45	John	john@company.com	555 123 4611	1987-08-01	AD_VP	17000	0	9900	90
46	Paul	paul@company.com	555 123 4612	1987-08-01	AD_VP	17000	0	9900	90
47	Christina	christina@company.com	555 123 4613	1987-08-01	AD_VP	17000	0	9900	90
48	Mark	mark@company.com	555 123 4614	1987-08-01	AD_VP	17000	0	9900	90
49	David	david@company.com	555 123 4615	1987-08-01	AD_VP	17000	0	9900	90
50	Neena	neena@company.com	555 123 4616	1987-08-01	AD_VP	17000	0	9900	90
51	Lex	lex@company.com	555 123 4617	1987-08-01	AD_VP	17000	0	9900	90
52	Alexander	alex@company.com	555 123 4618	1987-08-01	AD_VP	17000	0	9900	90
53	Baerenson	baerenson@company.com	555 123 4619	1987-08-01	AD_VP	17000	0	9900	90
54	Kit	kit@company.com	555 123 4620	1987-08-01	AD_VP	17000	0	9900	90
55	Timothy	timothy@company.com	555 123 4621	1987-08-01	AD_VP	17000	0	9900	90
56	David	david@company.com	555 123 4622	1987-08-01	AD_VP	17000	0	9900	90
57	John	john@company.com	555 123 4623	1987-08-01	AD_VP	17000	0	9900	90
58	Paul	paul@company.com	555 123 4624	1987-08-01	AD_VP	17000	0	9900	90
59	Christina	christina@company.com	555 123 4625	1987-08-01	AD_VP	17000	0	9900	90
60	Mark	mark@company.com	555 123 4626	1987-08-01	AD_VP	17000	0	9900	90
61	David	david@company.com	555 123 4627	1987-08-01	AD_VP	17000	0	9900	90
62	Neena	neena@company.com	555 123 4628	1987-08-01	AD_VP	17000	0	9900	90
63	Lex	lex@company.com	555 123 4629	1987-08-01	AD_VP	17000	0	9900	90
64	Alexander	alex@company.com	555 123 4630	1987-08-01	AD_VP	17000	0	9900	90
65	Baerenson	baerenson@company.com	555 123 4631	1987-08-01	AD_VP	17000	0	9900	90
66	Kit	kit@company.com	555 123 4632	1987-08-01	AD_VP	17000	0	9900	90
67	Timothy	timothy@company.com	555 123 4633	1987-08-01	AD_VP	17000	0	9900	90
68	David	david@company.com	555 123 4634	1987-08-01	AD_VP	17000	0	9900	90
69	John	john@company.com	555 123 4635	1987-08-01	AD_VP	17000	0	9900	90
70	Paul	paul@company.com	555 123 4636	1987-08-01	AD_VP	17000	0	9900	90
71	Christina	christina@company.com	555 123 4637	1987-08-01	AD_VP	17000	0	9900	90
72	Mark	mark@company.com	555 123 4638	1987-08-01	AD_VP	17000	0	9900	90
73	David	david@company.com	555 123 4639	1987-08-01	AD_VP	17000	0	9900	90
74	Neena	neena@company.com	555 123 4640	1987-08-01	AD_VP	17000	0	9900	90
75	Lex	lex@company.com	555 123 4641	1987-08-01	AD_VP	17000	0	9900	90
76	Alexander	alex@company.com	555 123 4642	1987-08-01	AD_VP	17000	0	9900	90
77	Baerenson	baerenson@company.com	555 123 4643	1987-08-01	AD_VP	17000	0	9900	90
78	Kit	kit@company.com	555 123 4644	1987-08-01	AD_VP	17000	0	9900	90
79	Timothy	timothy@company.com	555 123 4645	1987-08-01	AD_VP	17000	0	9900	90
80	David	david@company.com	555 123 4646	1987-08-01	AD_VP	17000	0	9900	90
81	John	john@company.com	555 123 4647	1987-08-01	AD_VP	17000	0	9900	90
82	Paul	paul@company.com	555 123 4648	1987-08-01	AD_VP	17000	0	9900	90
83	Christina	christina@company.com	555 123 4649	1987-08-01	AD_VP	17000	0	9900	90
84	Mark	mark@company.com	555 123 4650	1987-08-01	AD_VP	17000	0	9900	90
85	David	david@company.com	555 123 4651	1987-08-01	AD_VP	17000	0	9900	90
86	Neena	neena@company.com	555 123 4652	1987-08-01	AD_VP	17000	0	9900	90
87	Lex	lex@company.com	555 123 4653	1987-08-01	AD_VP	17000	0	9900	90
88	Alexander	alex@company.com	555 123 4654	1987-08-01	AD_VP	17000	0	9900	90
89	Baerenson	baerenson@company.com	555 123 4655	1987-08-01	AD_VP	17000	0	9900	90
90	Kit	kit@company.com	555 123 4656	1987-08-01	AD_VP	17000	0	9900	90
91	Timothy	timothy@company.com	555 123 4657	1987-08-01	AD_VP	17000	0	9900	90
92	David	david@company.com	555 123 4658	1987-08-01	AD_VP	17000	0	9900	90
93	John	john@company.com	555 123 4659	1987-08-01	AD_VP	17000	0	9900	90
94	Paul	paul@company.com	555 123 4660	1987-08-01	AD_VP	17000	0	9900	90
95	Christina	christina@company.com	555 123 4661	1987-08-01	AD_VP	17000	0	9900	90
96	Mark	mark@company.com	555 123 4662	1987-08-01	AD_VP	17000	0	9900	90
97	David	david@company.com	555 123 4663	1987-08-01	AD_VP	17000	0	9900	90
98	Neena	neena@company.com	555 123 4664	1987-08-01	AD_VP	17000	0	9900	90
99	Lex	lex@company.com	555 123 4665	1987-08-01	AD_VP	17000	0	9900	90
100	Alexander	alex@company.com	555 123 4666	1987-08-01	AD_VP	17000	0	9900	90

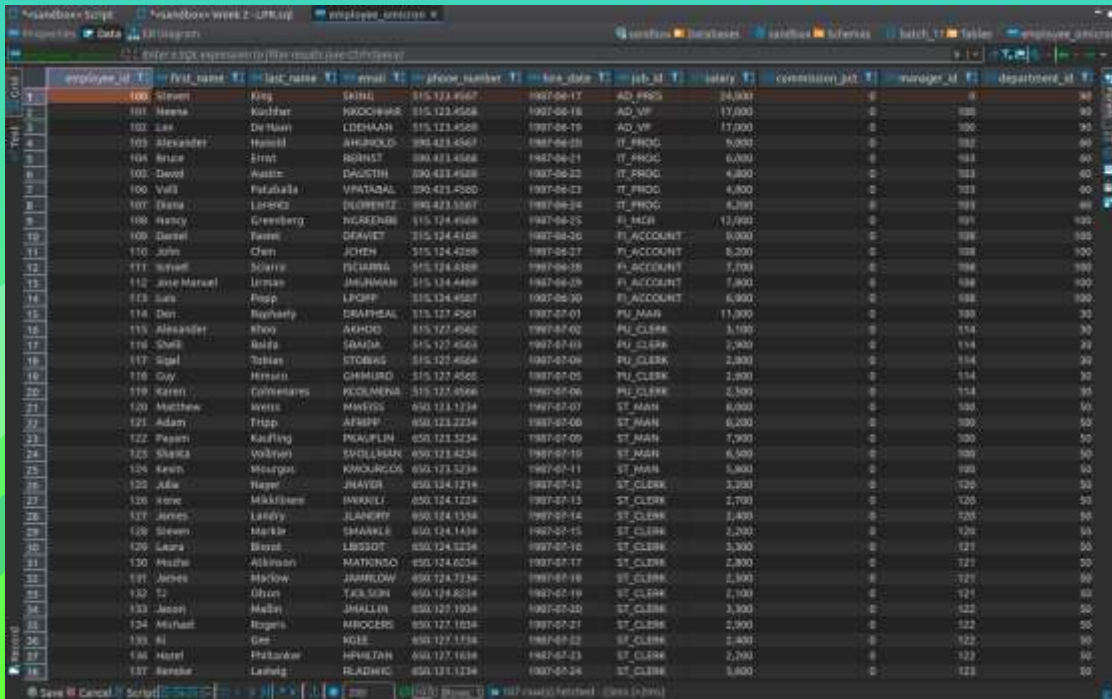
# IMPORT DATA

Klik kanan pada menu **Table** dibawah **Schema** yang ditentukan > **Import Data** > pilih file **employees.csv**



# IMPORT DATA

Output:



employee_id	first_name	last_name	email	phone_number	hire_date	job_id	salary	commission_pct	manager_id	department_id
100	Steven	King	SKING	515.123.4567	1987-06-17	AD_PRES	24,000	0	0	90
101	Neena	Kochhar	NKOCHHAR	515.123.4568	1987-06-18	AD_VP	17,000	0	100	90
102	Lex	DeHaan	LDEHAAN	515.123.4569	1987-06-19	AD_VP	17,000	0	100	90
103	Alexander	Hunold	AHUNOLD	590.423.4567	1987-06-20	IT_PROG	9,000	0	102	90
104	Bruce	Ernst	BERNST	590.423.4568	1987-06-21	IT_PROG	6,000	0	103	90
105	David	Austin	DAUSTIN	590.423.4569	1987-06-22	IT_PROG	4,000	0	103	90
106	Valli	Pataballa	VPATABAL	590.423.4560	1987-06-23	IT_PROG	4,000	0	103	90
107	Diana	Lorentz	DLorentz	590.423.4567	1987-06-24	IT_PROG	4,000	0	103	90
108	Nancy	Greenberg	NGREENBE	515.124.4568	1987-06-25	FI_MGR	12,000	0	100	100
109	Daniel	Favari	DFAVARI	515.124.4569	1987-06-26	FI_ACCOUNT	9,000	0	108	100
110	John	Chen	JCHEN	515.124.4568	1987-06-27	FI_ACCOUNT	8,200	0	108	100
111	Whitney	Schmidt	WSCHMIDT	515.124.4568	1987-06-28	FI_ACCOUNT	7,100	0	108	100
112	Jose Manuel	Urman	JURMAN	515.124.4468	1987-06-29	FI_ACCOUNT	7,000	0	108	100
113	Sam	Patel	SPATEL	515.124.4567	1987-06-30	FI_ACCOUNT	6,900	0	108	100
114	Den	Raphaely	DRAPHAEL	515.127.4567	1987-07-01	PU_MAN	11,000	0	100	30
115	Alexander	Koo	AKOO	515.127.4568	1987-07-02	PU_CLERK	3,100	0	114	30
116	Shelley	Baer	SBAER	515.127.4569	1987-07-03	PU_CLERK	2,900	0	114	30
117	Julia	Abel	JABEL	515.127.4568	1987-07-04	PU_CLERK	2,800	0	114	30
118	Guy	Herrman	GHERMAN	515.127.4569	1987-07-05	PU_CLERK	2,700	0	114	30
119	Kenneth	Coleman	KCOLEMAN	515.127.4568	1987-07-06	PU_CLERK	2,500	0	114	30
120	Matthew	West	MWEST	650.131.1234	1987-07-07	ST_MAN	8,000	0	100	50
121	Adam	Frippe	AFRIPPE	650.132.2234	1987-07-08	ST_MAN	6,200	0	100	50
122	Pavan	Kauffman	PKAUFFMAN	650.132.2234	1987-07-09	ST_MAN	7,500	0	100	50
123	Shanta	Vollman	SVOLLMAN	650.132.2234	1987-07-10	ST_MAN	6,500	0	100	50
124	Kevin	Mourgos	KMOURGOS	650.131.5234	1987-07-11	ST_MAN	5,800	0	100	50
125	Alex	Hayer	AHAYER	650.134.1234	1987-07-12	ST_CLERK	3,200	0	120	50
126	Neena	Malkinson	NMALKINSON	650.124.1234	1987-07-13	ST_CLERK	2,700	0	120	50
127	James	Landry	JLANDRY	650.124.1334	1987-07-14	ST_CLERK	2,400	0	120	50
128	Steven	Martin	SMARTIN	650.124.1434	1987-07-15	ST_CLERK	2,200	0	120	50
129	Laura	Brown	LBROWN	650.124.1534	1987-07-16	ST_CLERK	2,300	0	121	50
130	Matthew	Adams	MADAMS	650.124.1634	1987-07-17	ST_CLERK	2,300	0	121	50
131	James	Blond	JBLOND	650.124.1734	1987-07-18	ST_CLERK	2,300	0	121	50
132	Li	Olsen	LOLSEN	650.124.1834	1987-07-19	ST_CLERK	2,100	0	121	50
133	Joseph	Martin	JMARTIN	650.125.1934	1987-07-20	ST_CLERK	3,300	0	122	50
134	Michael	Rogers	MROGERS	650.127.1034	1987-07-21	ST_CLERK	2,900	0	122	50
135	Ali	Gee	AGEE	650.127.1134	1987-07-22	ST_CLERK	2,400	0	122	50
136	Hazel	Patton	HPATTON	650.127.1234	1987-07-23	ST_CLERK	2,500	0	122	50
137	Renske	Lawler	RLAWLER	650.131.1234	1987-07-24	ST_CLERK	3,600	0	123	50

Data dari **employees.csv** tersimpan dalam **TABLE** **employee\_omicon** yang sudah memiliki **FORMAT YANG SAMA**.

# INSERT INTO

**INSERT INTO** adalah fungsi yang digunakan untuk menambahkan catatan baru didalam **TABLE**, mengisi data **TABLE** secara manual atau dari **TABLE** lain.

> *Syntax* untuk mengisi data dalam **TABLE**:

```
insert into table_name values (data1, data2,...), (data1, data2,...)
insert into table_name (column1, column2,...),
values (data1, data2,...), (data1, data2,...)
```

> *Syntax* untuk mengisi data dari **TABLE** lain

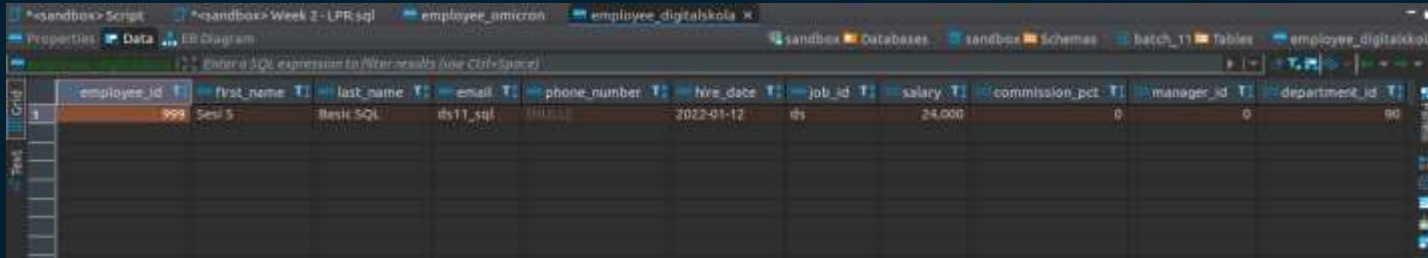
```
• insert into table_name
  select * from table_name

• insert into table_name (column1, column2,...)
  select column1, column2,... from table_name
```

# INSERT INTO

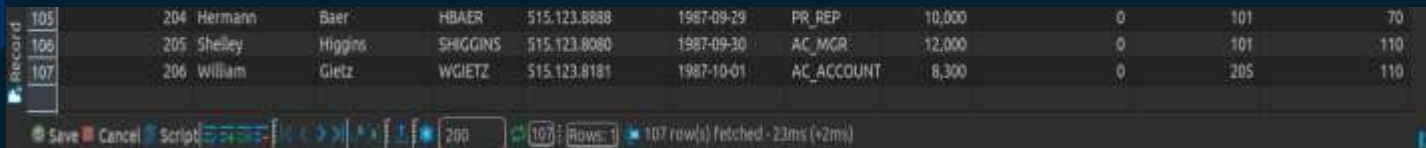


TABEL DATA **employee\_digitalskola**



employee_id	first_name	last_name	email	phone_number	hire_date	job_id	salary	commission_pct	manager_id	department_id
999	Selil	SQL	ds11_sql		2022-01-12	ds	24,000	0	0	90

Data Tabel **employee\_digitalskola** (Gambar diatas) yang akan dimasukkan ke dalam Tabel **employee\_omicron** (Gambar dibawah).



105	204	Hermann	Baer	HBAER	515.123.8888	1987-09-29	PR_REP	10,000	0	101	70
106	205	Shelley	Higgins	SHIGGINS	515.123.8080	1987-09-30	AC_MGR	12,000	0	101	110
107	206	William	Gietz	WGIEZT	515.123.8181	1987-10-01	AC_ACCOUNT	8,300	0	205	110

TABEL DATA **employee\_omicron**

# INSERT INTO



```

● insert into batch_11.employee_omicron
(
  employee_id,
  first_name,
  last_name,
  email,
  phone_number,
  hire_date,
  job_id,
  salary,
  commission_pct,
  manager_id,
  department_id
)
select
  employee_id,
  first_name,
  last_name,
  email,
  phone_number,
  hire_date,
  job_id,
  salary,
  commission_pct,
  manager_id,
  department_id
from batch_11.employee_digitalskola
    
```

## *INSERT* data dari tabel lain

Tabel **employee\_digitalskola** akan di-*INSERT* ke dalam tabel **employee\_omicron** seperti pada gambar di kiri.

```

● insert into batch_11.employee_omicron
  select * from batch_11.employee_digitalskola
    
```

Penggunaan `select * from` akan langsung memasukkan **semua** data dari **semua kolom** yang ada pada tabel **employee\_digitalskola**.



# INSERT INTO

OUTPUT:

Record	105	204	Hermann	Baer	HBAER	515.123.8888	1987-09-29	PR_REP	10,000	0	101	70
	106	205	Shelley	Higgins	SHIGGINS	515.123.8080	1987-09-30	AC_MGR	12,000	0	101	110
	107	206	William	Gietz	WGIEZ	515.123.8181	1987-10-01	AC_ACCOUNT	8,300	0	205	110
	108	999	Sesi 5	Basic SQL	ds11_sql	[NULL]	2022-01-12	ds	24,000	0	0	0

Save Cancel Script 200 108 Rows: 1 108 row(s) fetched - 24ms (+1ms)

Data dari Tabel **employee\_digitalskola** sudah tersimpan di Tabel **employee\_omicron**.



# INSERT INTO

```
insert into batch_11.employee_omicron
values
(
  260,
  'Anugrah',
  'Ghani',
  'YAZID',
  '123.456.7890',
  '2022-01-15',
  'ds',
  26000,
  0,
  0,
  90
)
```

## *INSERT* data manual

memasukkan data secara manual ke dalam tabel **employee\_omicron** sesuai dengan urutan kolom yang ada pada tabel **employee\_omicron** seperti pada gambar di kiri.





# INSERT INTO

OUTPUT:

105	204	Hermann	Baer	HBAER	515.123.8888	1987-09-29	PR_REP	10,000	0	101	70
106	205	Shelley	Higgins	SHIGGINS	515.123.8080	1987-09-30	AC_MGR	12,000	0	101	110
107	206	William	Gietz	WGIEZ	515.123.8181	1987-10-01	AC_ACCOUNT	8,300	0	205	110
108	999	Sasi S	Basic SQL	dist1_sql	[null]	2022-01-12	ds	24,000	0	0	90
109	260	Anugrah	Chani	YAZID	123.456.7890	2022-01-15	ds	26,000	0	0	90

Save Cancel Script [SQL Editor Icons] 200 108 Rows: 1 109 row(s) fetched - 23ms (+1ms)

Data yang diisi manual sudah tersimpan di Tabel employee\_omicron.



# UPDATE

Fungsi yang digunakan untuk mengubah nilai dalam *TABLE* berdasarkan persyaratan yang ditentukan.

*Syntax* UPDATE:

```
• update table_name  
  set column1 = data1, column2 = data2, ....  
  where conditions
```

# UPDATE

```
update batch_11.employee_omicron
set
  first_name = 'Belajar',
  last_name = 'SQL',
  email = 'ds11_sql',
  hire_date = current_timestamp,
  salary = 55123
where employee_id = 100

update batch_11.employee_omicron
set
  first_name = 'Omicron',
  last_name = 'DS',
  email = 'OMICRON',
  hire_date = current_timestamp,
  salary = 26000
where employee_id = 101
```

Melakukan **UPDATE** / mengubah data employee\_id yang mempunyai *value* 100 dan 101 pada Tabel **employee\_omicron**. Data yang diubah adalah data pada:

- > first\_name
- > last\_name
- > email
- > hire\_date
- > salary

SETELAH UPDATE **TIDAK BISA DI UNDO !!!**

# UPDATE

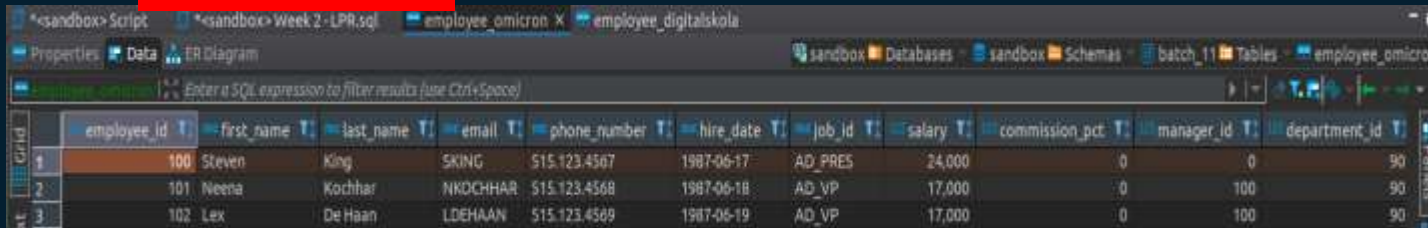
```
update batch_11.employee_omicron
set
  first_name = null,
  last_name = null,
  email = null,
  hire_date = null,
  salary = null
where employee_id = 102
```

Melakukan **UPDATE** / mengubah data employee\_id yang mempunyai *value* **102** pada Tabel **employee\_omicron** menjadi *NULL*. Data yang diubah adalah data pada:

- > first\_name
- > last\_name
- > email
- > hire\_date
- > salary

# UPDATE

## Data **SEBELUM UPDATE**



	employee_id	first_name	last_name	email	phone_number	hire_date	job_id	salary	commission_pct	manager_id	department_id
1	100	Steven	King	SKING	515.123.4567	1987-06-17	AD_PRES	24,000	0	0	90
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568	1987-06-18	AD_VP	17,000	0	100	90
3	102	Lex	De Haan	LDEHAAN	515.123.4569	1987-06-19	AD_VP	17,000	0	100	90

## OUTPUT: Data **SETELAH UPDATE**

*Update data employee\_id = 100 dan 101*



109	100	Belajar	SQL	ds11_sql	515.123.4567	2022-01-17 20:16	AD_PRES	55,123	0	0	90
74	101	Omicron	DS	OMICRON	515.123.4568	2022-01-17 20:16	AD_VP	26,000	0	100	90

*Update data employee\_id = 102*



74	102	[NULL]	[NULL]	[NULL]	515.123.4569	[NULL]	AD_VP	[NULL]	0	100	90
----	-----	--------	--------	--------	--------------	--------	-------	--------	---	-----	----

# SELECT

Fungsi yang digunakan untuk memperlihatkan *TABLE* dalam kondisi yang spesifik.

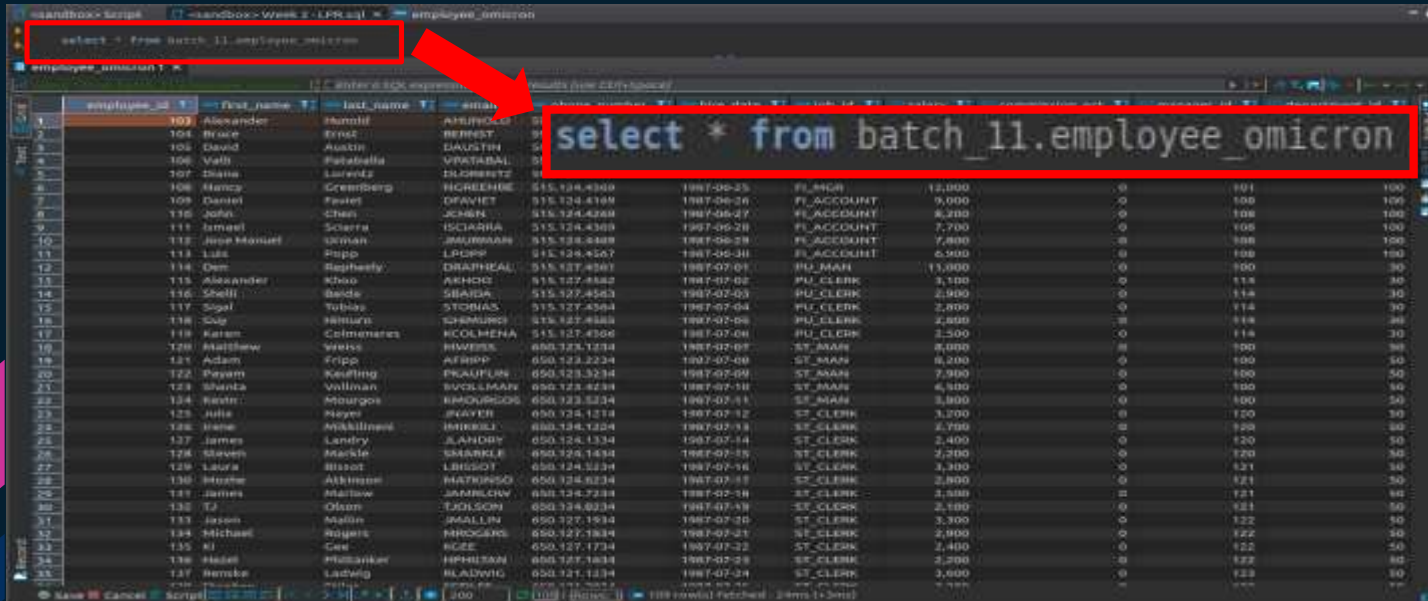
*Syntax* untuk memperlihatkan seluruh isi *TABLE*:

```
select * from table_name
```

*Syntax* untuk memperlihatkan isi *TABLE* dengan syarat yang spesifik:

```
select column1, column2, ... from table_name
```

# SELECT



The screenshot shows a SQL script window with the following content:

```

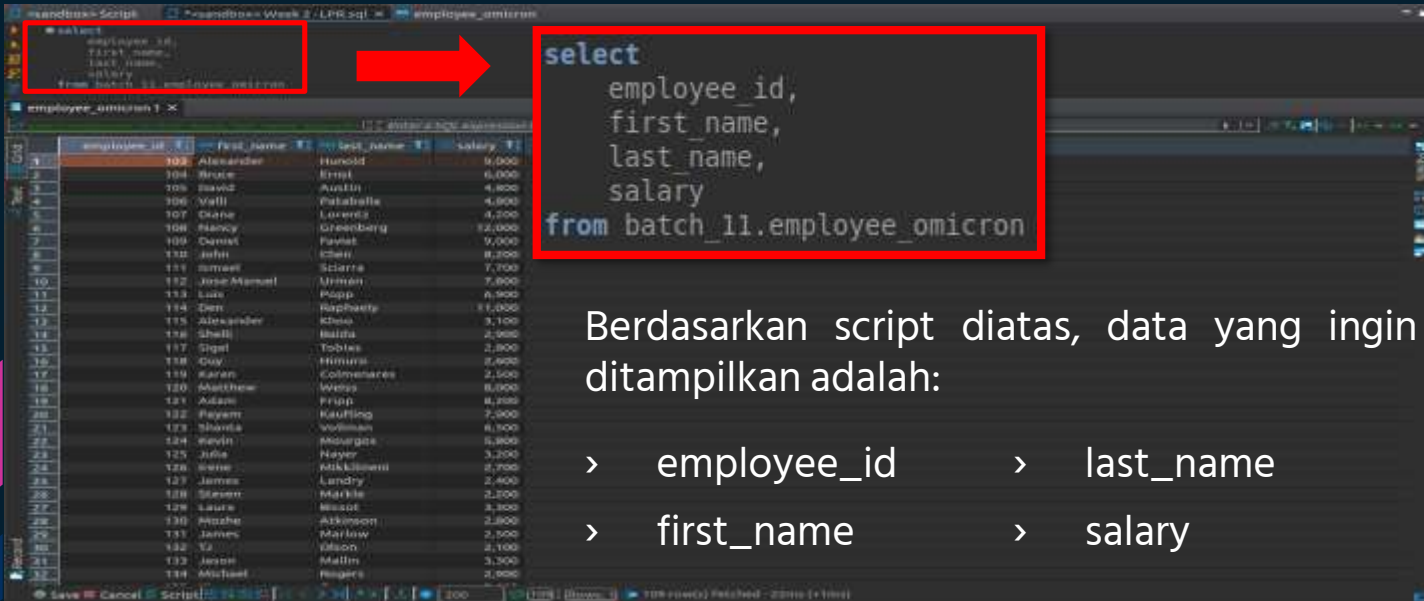
select * from batch_11.employee_omicon

```

The results of the query are displayed in a table with the following columns: employee\_id, first\_name, last\_name, email, phone\_number, hire\_date, job\_id, salary, commission\_pct, department\_id. The table contains 33 rows of data, including employees like Alexander Herold, Bruce Ernst, David Austin, Valli Pataballa, Diana DeSilva, etc.

select \* from akan menampilkan semua data pada tabel employee\_omicon pada window script.

# SELECT



The screenshot shows a SQL script editor with a query and its results. A red box highlights the query, and a red arrow points to a larger red box containing the same query. The results are displayed in a table below the query.

```
select
  employee_id,
  first_name,
  last_name,
  salary
from batch_11.employee_omicron
```

employee_id	first_name	last_name	salary
100	Alexander	Hunold	6,000
101	Bruce	Ernst	6,000
102	David	Austin	4,800
103	Wali	Pataballa	4,800
104	Diana	Lavigne	4,200
105	Nancy	Greenberg	12,000
106	Oliver	Pavlov	9,000
107	John	Chen	8,300
108	Samuel	Stearns	7,000
109	Jose Manuel	Ullrich	7,000
110	Lisa	Popp	6,900
111	Den	Raphaely	11,000
112	Alexander	Ichou	3,100
113	Shelley	Baila	2,900
114	Sigal	Tobias	2,800
115	Guy	Himura	2,600
116	Karen	Colmenares	2,500
117	Matthew	West	8,600
118	Adam	Fripp	8,200
119	Peyton	Kaufing	7,900
120	Shirley	Vollman	8,100
121	Kevin	Mourgos	5,800
122	Jude	Nayer	3,200
123	Wendy	Whalen	2,700
124	James	Landry	2,400
125	Steven	Martin	2,300
126	Laura	Issott	3,300
127	Neena	Adams	2,900
128	Lex	De Haan	2,500
129	Julia	Abel	2,100
130	Joan	Mali	3,300
131	Michael	Rogers	2,900

Berdasarkan script diatas, data yang ingin ditampilkan adalah:

- > employee\_id
- > first\_name
- > last\_name
- > salary

Script diatas akan menampilkan data **yang ingin ditampilkan** di tabel **employee\_omicron** pada *window script*.



# ALTER TABLE

Fungsi yang digunakan untuk melakukan beberapa perubahan didalam **SCHEMA**, seperti menambah dan/atau menghapus kolom.

*Syntax* yang digunakan untuk menambahkan kolom:

```
alter table table_name add column1 datatype ( length )
```

*Syntax* yang digunakan untuk menghapus kolom:

```
alter table table_name drop column column1
```



# ALTER TABLE

```
alter table batch_11.employee_omicron
add hobby varchar(255)
```

**Menambahkan kolom** 'hobby' pada tabel **employee\_omicron** dengan ALTER  
**TABLE table\_name ADD column1 datatype(length)**

Column Name	#	Data type	Identity	Collation	Not Null	Default	Comment
employee_id	1	int4			[ ]		
first_name	2	varchar(255)	default		[ ]		
last_name	3	varchar(255)	default		[ ]		
email	4	varchar(255)	default		[ ]		
phone_number	5	varchar(255)	default		[ ]		
hire_date	6	varchar(255)	default		[ ]		
job_id	7	varchar(255)	default		[ ]		
salary	8	float4			[ ]		
commission_pct	9	float4			[ ]		
manager_id	10	int4			[ ]		
department_id	11	int4			[ ]		

Data kolom **SEBELUM** penggunaan  
**ALTER TABLE - ADD**

Column Name	#	Data type	Identity	Collation	Not Null	Default	Comment
employee_id	1	int4			[ ]		
first_name	2	varchar(255)	default		[ ]		
last_name	3	varchar(255)	default		[ ]		
email	4	varchar(255)	default		[ ]		
phone_number	5	varchar(255)	default		[ ]		
hire_date	6	varchar(255)	default		[ ]		
job_id	7	varchar(255)	default		[ ]		
salary	8	float4			[ ]		
commission_pct	9	float4			[ ]		
manager_id	10	int4			[ ]		
department_id	11	int4			[ ]		
hobby	12	varchar(255)	default		[ ]		

Data kolom **SETELAH** penggunaan  
**ALTER TABLE - ADD**



# ALTER TABLE

Gunakan **UPDATE** untuk **menambahkan data** pada kolom 'hobby' yang sudah di tambahkan dengan **ALTER TABLE** pada tabel **employee\_omicron**.

```
update batch_11.employee_omicron
set
    hobby = 'Coding'
where employee_id = 100

update batch_11.employee_omicron
set
    hobby = 'Main Game'
where employee_id = 101
```

SETELAH ALTER TABLE **TIDAK BISA DI UNDO !!!**



# ALTER TABLE

OUTPUT:

Data **SEBELUM** kolom 'hobby' di UPDATE

	id	first_name	last_name	email	phone_number	hire_date	job_id	salary	commission_pct	manager_id	department_id	hobby
73	101	Omicron	DS	OMICRON	515.123.4568	2022-01-17 20:16	AD_VP	26,000	0	100	90	[NULL]
109	100	Belajar	SQL	ds11_sql	515.123.4567	2022-01-18 09:50	AD_PRES	55,123	0	0	90	[NULL]

Data **SETELAH** kolom 'hobby' di UPDATE

74	101	Omicron	DS	OMICRON	515.123.4568	2022-01-17 20:16	AD_VP	26,000	0	100	90	Main Game
109	100	Belajar	SQL	ds11_sql	515.123.4567	2022-01-17 20:16	AD_PRES	55,123	0	0	90	Coding

Data 'Main Game' dan 'Coding' telah tersimpan pada kolom 'hobby' dimana:

- > 'Main Game' pada row dimana employee\_id = 101
- > 'Coding' pada row dimana employee\_id = 100



# ALTER TABLE

```
alter table batch_11.employee_omicron
drop column
    commission_pct

alter table batch_11.employee_omicron
drop column
    manager_id

alter table batch_11.employee_omicron
drop column
    department_id
```

**Menghapus kolom** 'commission\_pct', 'manager\_id', dan 'department\_id' pada tabel **employee\_omicron** dengan ALTER

**TABLE *table\_name* DROP COLUMN *column1***



# ALTER TABLE

OUTPUT:

Column Name	#	Data type	Identity	Collation	Not Null	Default	Comment
employee_id	1	int4			[ ]		
first_name	2	varchar(255)	<a href="#">default</a>		[ ]		
last_name	3	varchar(255)	<a href="#">default</a>		[ ]		
email	4	varchar(255)	<a href="#">default</a>		[ ]		
phone_number	5	varchar(255)	<a href="#">default</a>		[ ]		
hire_date	6	varchar(255)	<a href="#">default</a>		[ ]		
job_id	7	varchar(255)	<a href="#">default</a>		[ ]		
salary	8	float4			[ ]		
commission_pct	9	float4			[ ]		
manager_id	10	int4			[ ]		
department_id	11	int4			[ ]		
hobby	12	varchar(255)	<a href="#">default</a>		[ ]		

Data kolom **SEBELUM** penggunaan  
ALTER TABLE – DROP COLUMN

Column Name	#	Data type	Identity	Collation	Not Null	Default	Comment
employee_id	1	int4			[ ]		
first_name	2	varchar(255)	<a href="#">default</a>		[ ]		
last_name	3	varchar(255)	<a href="#">default</a>		[ ]		
email	4	varchar(255)	<a href="#">default</a>		[ ]		
phone_number	5	varchar(255)	<a href="#">default</a>		[ ]		
hire_date	6	varchar(255)	<a href="#">default</a>		[ ]		
job_id	7	varchar(255)	<a href="#">default</a>		[ ]		
salary	8	float4			[ ]		
hobby	12	varchar(255)	<a href="#">default</a>		[ ]		

Data kolom **SETELAH** penggunaan  
ALTER TABLE – DROP COLUMN





# DELETE

```
delete from batch_11.employee_omicron
where
    employee_id not in (100, 101)

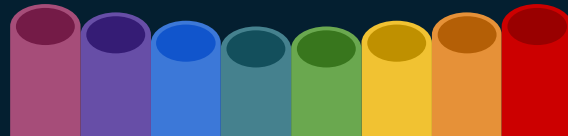
OR

delete from batch_11.employee_omicron
where
    first_name not in ('Belajar', 'Omicron')
```

DELETE digunakan untuk **MENGHAPUS** **ROW** yang dipilih atau bisa menghapus semua row.

**Menghapus semua row** EXCEPT row dengan: (GUNAKAN NOT IN)

- > employee\_id = 100           -> OR first\_name = 'Belajar'
- > first\_name = 'Omicron' -> OR employee\_id = 101





# DELETE

OUTPUT:

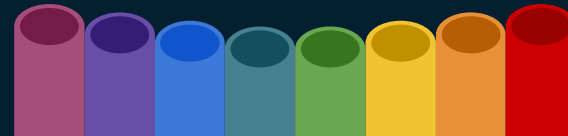
Data **SETELAH DELETE**

	employee_id	first_name	last_name	email	phone_number	hire_date	job_id	salary	hobby
1	101	Omicron	DS	OMICRON	515.123.4568	2022-01-18 09:50:17.335379+07	AD_VP	26,000	Main Game
2	100	Belajar	SQL	ds11_sql	515.123.4567	2022-01-18 09:50:13.121192+07	AD_PRES	55,123	Coding

Row yang **tersisa** hanya tinggal baris dengan :

- > employee\_id = 100 -> OR first\_name = 'Belajar'
- > first\_name = 'Omicron' -> OR employee\_id = 101

SETELAH DELETE **TIDAK BISA DI UNDO !!!**









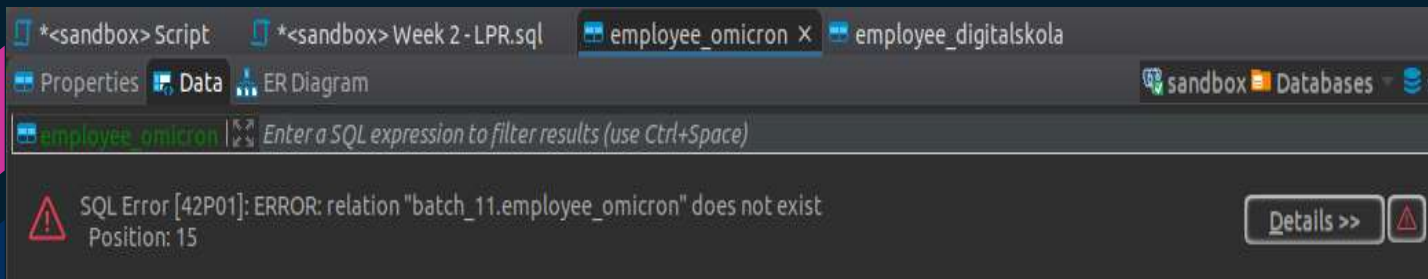
# DROP

```
drop table batch_11.employee_omicron
```

**MENGHAPUS SEMUA DATA** PADA TABEL

SETELAH DROP **TIDAK BISA DI *UNDO* !!!**

OUTPUT:




Output **Setelah** DROP



# *Intermediate* SQL

# General Function



	
	<b>SELECT</b> column1, column2, ... <b>FROM</b> table_name <b>WHERE</b> condition(s) <b>GROUP BY</b> field_name(s) <b>HAVING</b> condition(s) <b>ORDER BY</b> field_name(s) <b>LIMIT</b> number;

\*Fungsi diatas tidak harus diisi tiap barisnya.

(Misal kita hanya perlu menggunakan SELECT statement, Maka kita tidak perlu menuliskan sisa fungsi dibawahnya.

SQL memiliki format penulisan fungsi yang harus diikuti agar fungsi tersebut dapat dijalankan jika tidak maka akan menimbulkan *error*.

Fungsi disamping adalah contoh format fungsi SQL yang terdiri dari *SELECT statement, WHERE, GROUP BY, HAVING, ORDER BY, dan LIMIT*.

# ***SELECT Statement***

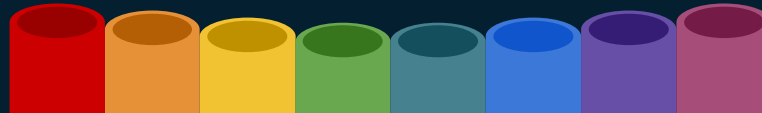


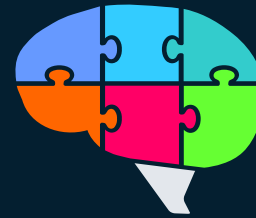
```
SELECT * FROM table_name;
```

```
SELECT column1, column2, ...  
FROM table_name;
```

Tanda *\** dapat digunakan jika kita ingin mengambil seluruh kolom dalam suatu tabel.

> *SELECT statement* adalah fungsi SQL yang digunakan untuk mengambil semua kolom atau kolom tertentu dalam sebuah tabel.






# Prefix dan Alias


## > Prefix :

- Penambahan nama tabel di depan nama kolom pada penulisan *syntax*.
- Format penulisan *prefix* adalah *prefix.column1*
- Dapat digabung dengan *alias*


	<pre>SELECT table_name.column1 FROM table_name;  SELECT table_name.column1 as Kolom FROM table_name as Tabel;</pre>
---	---

## > Alias :

- Memberikan nama ganti dari suatu kolom/tabel untuk mempermudah penulisannya di *syntax*.
- Format penulisan *alias* adalah menambahkan kata **as** setelah penulisan nama kolom/tabel.

	<pre>SELECT column1 as Kolom FROM table_name as Tabel;  SELECT column1 Kolom FROM table_name Tabel;</pre>
---	---

# ***DISTINCT***

	 <pre>SELECT DISTINCT column1 FROM table_name;</pre>

*DISTINCT* adalah fungsi SQL yang digunakan untuk :

1. Menghapus duplikasi data pada suatu kolom.
2. Memperlihatkan satu kolom tanpa menggunakan fungsi **GROUP BY**.
3. Menghitung *unique values*.

Contoh hasil *query* jika menggunakan fungsi **DISTINCT** dan tanpa **DISTINCT**.

Tanpa **DISTINCT**

Domisili
Jakarta
Depok
Tangerang
Jakarta
Tangerang

Pakai **DISTINCT**

Domisili
Jakarta
Depok
Tangerang

# Fungsi *STRING*

➤ Fungsi ***STRING*** digunakan untuk memanipulasi tipe data teks (*string*).

Fungsi	Tipe Data Hasil	Deskripsi	Contoh	Hasil
<code>string    string</code>	text	String concatenation	<code>'Digital'    'Skola'</code>	DigitalSkola
<code>string    non string</code> atau <code>non string    string</code>	text	String concatenation with one non string input	<code>'Umur'    '21'</code>	Umur21
<code>char_length(string)</code> or <code>character_length(string)</code>	int	Number of characters in string	<code>char_length('Digital')</code>	7
<code>lower(string)</code>	text	Convert string to lowercase	<code>lower('SKOLA')</code>	skola
<code>upper(string)</code>	text	Convert string to uppercase	<code>upper('science')</code>	SCIENCE
<code>substring(string [from int] [for int])</code>	text	Extract substring	<code>substring('Science' from 2 for 4)</code>	cien
<code>position(substring in string)</code>	int	Location of specified substring	<code>position('la' in 'skola')</code>	4



# Fungsi Aggregate

- Fungsi **Aggregate** digunakan untuk melakukan operasi pada tiap nilai data di suatu kolom dan menghasilkan satu nilai data.

Function	Argument Type(s)	Return Type	Description
<code>sum(expression)</code>	smallint, int, bigint, real, double precision, numeric, interval, or money	bigint for smallint or int arguments, numeric for bigint arguments, otherwise the same as the argument data type	Sum of expression across all non-null input values
<code>count(*)</code>	any	bigint	Number of input rows
<code>count (expression)</code>	any	bigint	Number of input rows for which the value of expression is not null
<code>avg(expression)</code>	smallint, int, bigint, real, double precision, numeric, or interval	numeric for any integer-type argument, double precision for a floating-point argument, otherwise the same as the argument data type	The average (arithmetic mean) of all non-null input values
<code>max(expression)</code>	any numeric, string, date/time, network, or enum type, or arrays of these types	same as argument type	Maximum value of expression across all non-null input values
<code>min(expression)</code>	any numeric, string, date/time, network, or enum type, or arrays of these types	same as argument type	Minimum value of expression across all non-null input values

# CASE ... WHEN ... Function



```
SELECT column1, column2, ...
CASE
  WHEN condition1 THEN result1
  WHEN condition2 THEN result2
  WHEN condition THEN result
  ELSE result
END as alias
FROM table_name;
```

Catatan : *alias* bersifat opsional.



*CASE... WHEN... Function* digunakan untuk mengevaluasi kondisi yang sudah ditentukan, dimulai dari *condition1* dan akan mengembalikan hasil (*result1*) jika *condition1* terpenuhi (*TRUE*). Jika tidak, maka *condition2* akan dievaluasi dan akan mengembalikan *result2* jika *condition2* terpenuhi, dan seterusnya. Apabila tidak ada kondisi yang terpenuhi maka *result* pada bagian *ELSE* yang akan dikembalikan.

# WHERE

```
SELECT column1, column2, ...  
FROM table_name  
WHERE condition(s);
```

**WHERE** adalah filter di SQL yang digunakan untuk memberikan kondisi logis tertentu terhadap data yang hendak diambil.

Untuk memberikan kondisi logis tertentu lebih dari satu, dapat menggunakan operator **AND** dan **OR**.

Operator **AND** : Membuat pengambilan data dapat dilakukan hanya ketika semua kondisi terpenuhi.

Operator **OR** : Membuat pengambilan data dapat dilakukan ketika salah satu kondisi terpenuhi.

Untuk *extract value* tipe data *string* gunakan **=**, dan untuk tipe data angka gunakan operator matematika (**>**, **<**, **>=**, **<=**).

# GROUP BY



```
SELECT column1, column2, ...
FROM table_name
WHERE condition(s)
GROUP BY column
ORDER BY column;
```

**GROUP BY** digunakan untuk merangkum *value* dan mengelompokkannya berdasarkan kriteria tertentu. Untuk merangkum *value* kita gunakan fungsi **AGGREGATE**.

Hal yang harus diperhatikan :

1. **GROUP BY** digunakan di dalam **SELECT statement**
2. **GROUP BY** ditempatkan setelah **FROM** atau setelah **WHERE**
3. Jika menggunakan **ORDER BY** maka **GROUP BY** ditempatkan sebelum **ORDER BY**.
4. Jika tidak merangkum *value* maka tidak perlu memasukkan fungsi **AGGREGATE**.



# HAVING



```
SELECT column1, column2, ...  
FROM table_name  
GROUP BY column  
HAVING condition(s);
```


**HAVING** adalah filter di SQL yang digunakan untuk memberikan kondisi logis tertentu terhadap data yang hendak diambil apabila menggunakan fungsi **AGGREGATE**.

Hal yang harus diperhatikan :

1. Gunakan beberapa kondisi untuk memperlihatkan lebih banyak baris.
2. Untuk memberikan kondisi logis tertentu lebih dari satu, dapat menggunakan operator AND dan OR.
3. Selalu pisahkan AND dan OR dengan tanda kurung.
4. Jangan lupa untuk selalu masukkan GROUP BY sebelum HAVING!

# ORDER BY



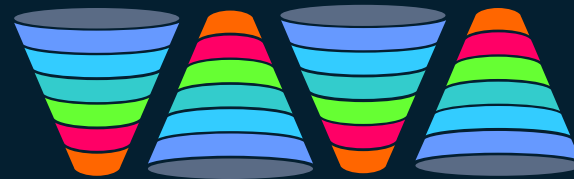
	
	<pre> <b>SELECT</b> column1, column2, ... <b>FROM</b> table_name <b>WHERE</b> condition(s) <b>GROUP BY</b> column <b>ORDER BY</b> column;         </pre>

**ORDER BY** digunakan untuk mengurutkan data berdasarkan suatu kolom tertentu.

Secara *default*, sistem akan mengurutkan data secara ASC (*Ascending*).

Hal yang harus diperhatikan :

1. Dapat menuliskan nama kolom sesuai dengan urutan penulisan nama kolom di *SELECT statement*.
2. Gunakan hanya ketika kita perlu mengurutkan hasil.
3. Gunakan **LIMIT** untuk membuat waktu proses *query* lebih cepat.



# LIMIT



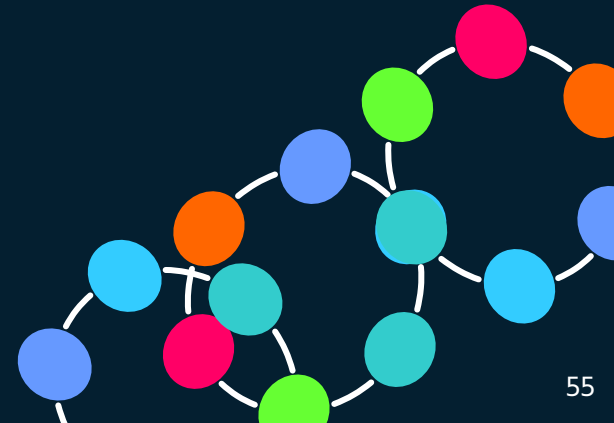
```
SELECT column1, column2, ...  
FROM table_name  
LIMIT number;
```

**LIMIT** digunakan untuk membatasi jumlah data yang akan di *query*.

Manfaat LIMIT tidak akan begitu terasa untuk jumlah data yang sedikit namun akan sangat membantu apabila kita menggunakan *Big Data*.

Hal yang harus diperhatikan :

1. Penggunaan LIMIT dapat membuat waktu proses *query* lebih cepat.
2. Dapat digunakan sebelum JOIN dan/atau bersamaan dengan ORDER BY.



# Query Processing Order

**Bagaimana sistem membaca urutan perintah *query* ?**



1. Memperoleh data (FROM, JOIN)
2. Filter baris (WHERE)
3. Pengelompokkan (GROUP BY)
4. Filter kelompok (HAVING)
5. *Return Expression* (SELECT)
6. *Order & Paging* (ORDER BY & LIMIT / OFFSET)





Terima Kasih !