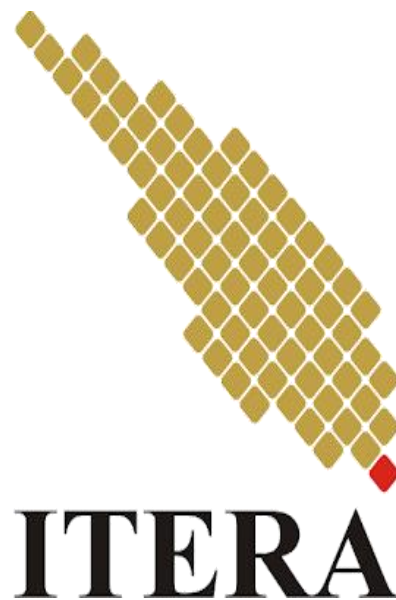


# **Analysis Data Penjualan di Supermarket Dengan SQL**

*Tugas Besar Basis Data – SD2206*



## **Kelompok 5 (RB)**

Ketua Kelompok	120450006	Arfyani Deiastuti
Anggota 1	120450062	Queena Aurora B.
Anggota 2	120450108	Ribka Gabriela S.
Anggota 3	120450036	Syifa Alwan Aulia
Anggota 4	120450044	Ahmad Zidan W.
Anggota 5	120450008	Sophia Yolanda RI

## DATA DESCRIPTION

Pada tugas ini, data yang dipakai diperoleh dari <https://www.kaggle.com> tahun 2019. Data tentang pertumbuhan supermarket di kota-kota terpadat yang semakin meningkat dan persaingan pasar juga tinggi. Dataset tersebut merupakan salah satu history penjualan perusahaan supermarket yang tercatat di 3 cabang berbeda selama 3 bulan yaitu bulan Januari sampai Maret 2019. Data terdiri dari 1000 baris dan 17 kolom. Namun data yang akan di akan digunakan hanyalah 32 baris dan 9 kolom, berikut rinciannya dari atribut yang digunakan :

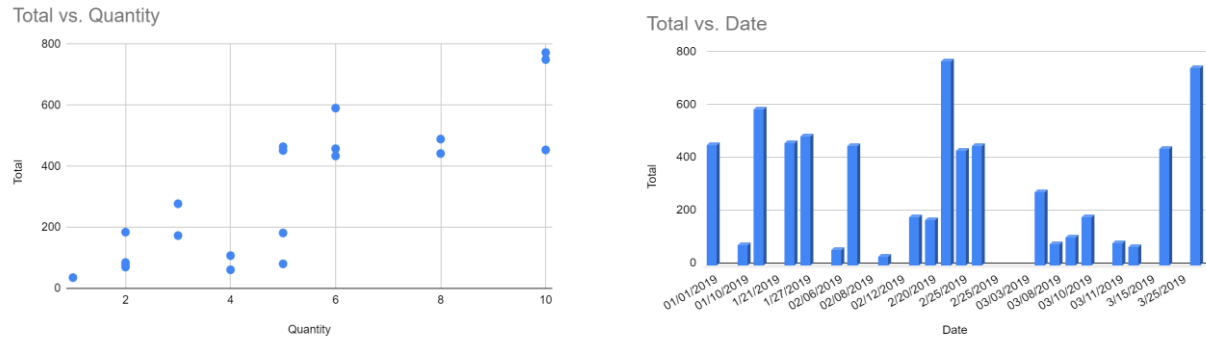
- **Invoice ID** : Nomor identifikasi faktur penjualan yang dihasilkan komputer
- **Date** : Tanggal pembelian (Catatan tersedia dari Januari 2019 hingga Maret 2019)
- **Branch** : Cabang supercenter (3 cabang tersedia diidentifikasi oleh A, B dan C).
- **City** : Lokasi supercenters
- **Product ID** : Nomor produk berdasarkan kategori-kategorinya.
- **Product Line** : Grup kategorisasi barang umum - Aksesori elektronik, Aksesori mode, Makanan dan minuman, Kesehatan dan kecantikan, Rumah dan gaya hidup, Olahraga dan perjalanan
- **Unit Price** : Harga setiap produk dalam \$
- **Quantity** : Jumlah produk yang dibeli oleh pelanggan
- **Total** : Total harga produk yang terjual

**Tabel 1.** Data Penjualan di Supermarket dari Januari hingga Maret 2019

Invoice ID	Branch	City	Product ID	Date	Product line	Unit price	Quantity	Total
750-67-8428	A	Yangon	P1	01/05/2019	Health and beauty	74.69	7	5.489.715
226-31-3081	C	Naypyitaw	P2	03/08/2019	Electronic accessories	15.28	5	80.22
631-41-3108	A	Yangon	P3	03/03/2019	Home and lifestyle	46.33	7	3.405.255
123-19-1176	A	Yangon	P1	1/27/2019	Health and beauty	58.22	8	489.048
373-73-7910	A	Yangon	P4	02/08/2019	Sports and travel	86.31	7	6.343.785
699-14-3026	C	Naypyitaw	P2	3/25/2019	Electronic accessories	85.39	7	6.276.165
355-53-5943	A	Yangon	P2	2/25/2019	Electronic accessories	68.84	6	433.692
315-22-5665	C	Naypyitaw	P3	2/24/2019	Home and lifestyle	73.56	10	772.38
665-32-9167	A	Yangon	P1	01/10/2019	Health and beauty	36.26	2	76.146
692-92-5582	B	Mandalay	P5	2/20/2019	Food and beverages	54.84	3	172.746

<b>351-62-0822</b>	B	Mandalay	P6	02/06/2019	Fashion accessories	14.48	4	60.816
<b>529-56-3974</b>	B	Mandalay	P2	03/09/2019	Electronic accessories	25.51	4	107.142
<b>365-64-0515</b>	A	Yangon	P2	02/12/2019	Electronic accessories	46.95	5	2.464.875
<b>252-56-2699</b>	A	Yangon	P5	02/07/2019	Food and beverages	43.19	10	453.495
<b>829-34-3910</b>	A	Yangon	P1	3/29/2019	Health and beauty	71.38	10	749.49
<b>299-46-1805</b>	B	Mandalay	P4	1/15/2019	Sports and travel	93.72	6	590.436
<b>656-95-9349</b>	A	Yangon	P1	03/11/2019	Health and beauty	68.93	7	5.066.355
<b>765-26-6951</b>	A	Yangon	P4	01/01/2019	Sports and travel	72.61	6	457.443
<b>329-62-1586</b>	A	Yangon	P5	1/21/2019	Food and beverages	54.67	3	1.722.105
<b>319-50-3348</b>	B	Mandalay	P3	03/11/2019	Home and lifestyle	40.3	2	84.63
<b>300-71-4605</b>	C	Naypyitaw	P2	2/25/2019	Electronic accessories	86.04	5	451.71
<b>371-85-5789</b>	B	Mandalay	P1	03/05/2019	Health and beauty	87.98	3	277.137
<b>273-16-6619</b>	B	Mandalay	P3	3/15/2019	Home and lifestyle	33.2	2	69.72
<b>636-48-8204</b>	A	Yangon	P2	2/17/2019	Electronic accessories	34.56	5	181.44
<b>549-59-1358</b>	A	Yangon	P4	03/02/2019	Sports and travel	88.63	3	2.791.845
<b>227-03-5010</b>	A	Yangon	P3	3/22/2019	Home and lifestyle	52.59	8	441.756
<b>649-29-6775</b>	B	Mandalay	P6	02/08/2019	Fashion accessories	33.52	1	35.196
<b>189-17-4241</b>	A	Yangon	P6	03/10/2019	Fashion accessories	87.67	2	184.107
<b>145-94-9061</b>	B	Mandalay	P5	1/25/2019	Food and beverages	88.36	5	463.89
<b>848-62-7243</b>	A	Yangon	P1	3/15/2019	Health and beauty	24.89	9	2.352.105

## Visualisasi Data



Gambar 1. Visualisasi Dataset yang digunakan

## NORMALISASI DATA

Normalisasi adalah teknik atau pendekatan yang digunakan dalam membangun *desain logic database relasional* melalui organisasi himpunan data dengan tingkat ketergantungan fungsional dan keterkaitan yang tinggi sedemikian sehingga menghasilkan struktur tabel yang normal. Tujuan normalisasi adalah meminimalisasi redundansi (pengulangan data), memudahkan identifikasi entitas, dan mencegah terjadinya anomali. Normalisasi ada 3 tahap yaitu :

1. (1NF) bentuk normal pertama apabila memenuhi jika dan hanya jika setiap atribut dari relasi tersebut hanya memiliki nilai tunggal dan tidak ada pengulangan grup atribut dalam baris.
2. Suatu relasi disebut memenuhi bentuk normal kedua (2NF) jika dan hanya jika memenuhi 1NF dan setiap atribut yang bukan kunci utama tergantung secara fungsional terhadap semua atribut kunci dan bukan hanya sebagian atribut kunci (fully functionally dependent).
3. Suatu relasi disebut memenuhi bentuk normal ketiga (3NF) jika dan hanya jika memenuhi 2NF dan setiap atribut yang bukan kunci tidak tergantung secara fungsional terhadap atribut bukan kunci yang lain dalam relasi tsb (tidak terdapat ketergantungan transitif pada atribut bukan kunci).

Data pada tabel 1 adalah data yang sudah ternormalisasi 1NF karena tidak ada atribut yang berulang, oleh sebab itu langkah selanjutnya lakukan normalisasi 2NF dengan mengeluarkan atribut product line menjadi tabel baru yang dianggap sebagai cabang dari atribut product id. Berikut merupakan hasil normalisasi 2NF.

Tabel 2. Tabel Supermarkets

Invoice ID	Branch	City	Product ID	Date	Unit price	Quantity	Total
750-67-8428	A	Yangon	P1	01/05/2019	74.69	7	5.489.715
226-31-3081	C	Naypyitaw	P2	03/08/2019	15.28	5	80.22
631-41-3108	A	Yangon	P3	03/03/2019	46.33	7	3.405.255

123-19-1176	A	Yangon	P1	1/27/2019	58.22	8	489.048
373-73-7910	A	Yangon	P4	02/08/2019	86.31	7	6.343.785
699-14-3026	C	Naypyitaw	P2	3/25/2019	85.39	7	6.276.165
355-53-5943	A	Yangon	P2	2/25/2019	68.84	6	433.692
315-22-5665	C	Naypyitaw	P3	2/24/2019	73.56	10	772.38
665-32-9167	A	Yangon	P1	01/10/2019	36.26	2	76.146
692-92-5582	B	Mandalay	P5	2/20/2019	54.84	3	172.746
351-62-0822	B	Mandalay	P6	02/06/2019	14.48	4	60.816
529-56-3974	B	Mandalay	P2	03/09/2019	25.51	4	107.142
365-64-0515	A	Yangon	P2	02/12/2019	46.95	5	2.464.875
252-56-2699	A	Yangon	P5	02/07/2019	43.19	10	453.495
829-34-3910	A	Yangon	P1	3/29/2019	71.38	10	749.49
299-46-1805	B	Mandalay	P4	1/15/2019	93.72	6	590.436
656-95-9349	A	Yangon	P1	03/11/2019	68.93	7	5.066.355
765-26-6951	A	Yangon	P4	01/01/2019	72.61	6	457.443
329-62-1586	A	Yangon	P5	1/21/2019	54.67	3	1.722.105
319-50-3348	B	Mandalay	P3	03/11/2019	40.3	2	84.63
300-71-4605	C	Naypyitaw	P2	2/25/2019	86.04	5	451.71
371-85-5789	B	Mandalay	P1	03/05/2019	87.98	3	277.137
273-16-6619	B	Mandalay	P3	3/15/2019	33.2	2	69.72
636-48-8204	A	Yangon	P2	2/17/2019	34.56	5	181.44
549-59-1358	A	Yangon	P4	03/02/2019	88.63	3	2.791.845
227-03-5010	A	Yangon	P3	3/22/2019	52.59	8	441.756
649-29-6775	B	Mandalay	P6	02/08/2019	33.52	1	35.196
189-17-4241	A	Yangon	P6	03/10/2019	87.67	2	184.107
145-94-9061	B	Mandalay	P5	1/25/2019	88.36	5	463.89
848-62-7243	A	Yangon	P1	3/15/2019	24.89	9	2.352.105

**Tabel 3.** Tabel Product

Product ID	Product line
P1	Health and beauty
P2	Electronic accessories
P3	Home and lifestyle
P4	Sports and travel
P5	Food and beverages
P6	Fashion accessories

Dari hasil normalisasi 2 NF ternyata ada atribut yang bukan primary key dan tidak memiliki ketergantungan pada primary key yaitu tabel product id. Sehingga bisa dilakukan normalisasi 3NF dengan memisahkan tabel supermarkets (invoice id, date, branch, dan city) dengan tabel item (product id, unit price, quantity, total). Setelah itu pisahkan juga tabel city karena dianggap cabang dari tabel branch. Berikut normalisasi Data dengan 3NF.

**Tabel 4.** Tabel Supermarkets

Invoice ID	Branch
750-67-8428	A
226-31-3081	C
631-41-3108	A
123-19-1176	A
373-73-7910	A
699-14-3026	C
355-53-5943	A
315-22-5665	C
665-32-9167	A
692-92-5582	B
351-62-0822	B
529-56-3974	B
365-64-0515	A
252-56-2699	A
829-34-3910	A
299-46-1805	B
656-95-9349	A
765-26-6951	A
329-62-1586	A
319-50-3348	B
300-71-4605	C
371-85-5789	B
273-16-6619	B
636-48-8204	A
549-59-1358	A
227-03-5010	A
649-29-6775	B
189-17-4241	A
145-94-9061	B
848-62-7243	A

**Tabel 6.** Tabel Location

Branch	City
A	Yangon
B	Mandalay
C	Naypyitaw

**Tabel 5.** Tabel Items

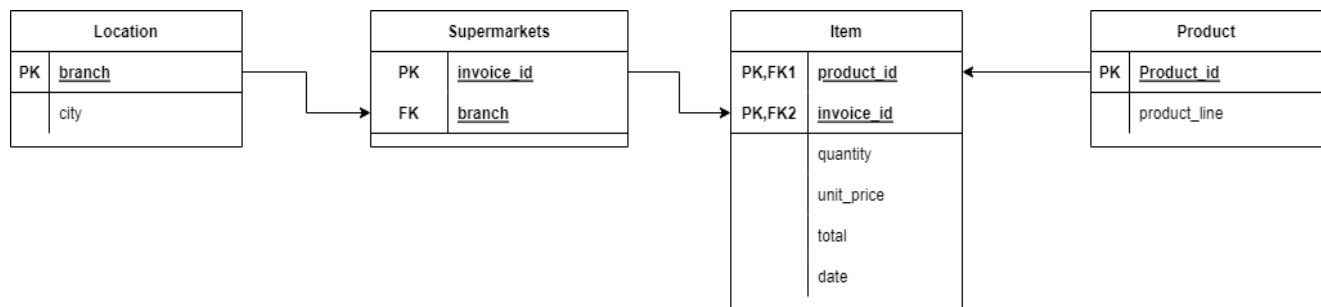
Product ID	Date	Unit price	Quantity	Total
P1	01/05/2019	74.69	7	5.489.715
P2	03/08/2019	15.28	5	80.22
P3	03/03/2019	46.33	7	3.405.255
P1	1/27/2019	58.22	8	489.048
P4	02/08/2019	86.31	7	6.343.785
P2	3/25/2019	85.39	7	6.276.165
P2	2/25/2019	68.84	6	433.692
P3	2/24/2019	73.56	10	772.38
P1	01/10/2019	36.26	2	76.146
P5	2/20/2019	54.84	3	172.746
P6	02/06/2019	14.48	4	60.816
P2	03/09/2019	25.51	4	107.142
P2	02/12/2019	46.95	5	2.464.875
P5	02/07/2019	43.19	10	453.495
P1	3/29/2019	71.38	10	749.49
P4	1/15/2019	93.72	6	590.436
P1	03/11/2019	68.93	7	5.066.355
P4	01/01/2019	72.61	6	457.443
P5	1/21/2019	54.67	3	1.722.105
P3	03/11/2019	40.3	2	84.63
P2	2/25/2019	86.04	5	451.71
P1	03/05/2019	87.98	3	277.137
P3	3/15/2019	33.2	2	69.72
P2	2/17/2019	34.56	5	181.44
P4	03/02/2019	88.63	3	2.791.845
P3	3/22/2019	52.59	8	441.756
P6	02/08/2019	33.52	1	35.196
P6	03/10/2019	87.67	2	184.107
P5	1/25/2019	88.36	5	463.89
P1	3/15/2019	24.89	9	2.352.105

**Tabel 7.** Tabel Product

Product ID	Product line
P1	Health and beauty
P2	Electronic accessories
P3	Home and lifestyle
P4	Sports and travel
P5	Food and beverages
P6	Fashion accessories

## RELATIONAL SCHEMA

Skema relasi merupakan sebuah cara untuk merepresentasikan hubungan antara satu tabel dengan tabel lainnya melalui sebuah kolom kunci. Pada skema relasi sebuah *primary key* suatu tabel merupakan *foreign key* pada tabel lainnya. Kunci tersebut selanjutnya dapat digunakan untuk membantu menggabungkan informasi dari tabel-tabel yang terpisah. Keuntungan menggunakan skema relasi adalah menyimpan data terpisah pada beberapa tabel yang saling terkait, memiliki keuntungan konsistensi, efisiensi, dan komprehensibilitas. Berikut adalah gambar skema relasi dari data kami mengenai penjualan di Supermarket.



**Gambar 2.** Relation Schema Dataset

Berdasarkan skema relasi di atas berikut adalah hubungan antar tabel:

- Location dengan Supermarket adalah one to many
- Supermarket dengan Item adalah one to many
- Item dengan Product adalah many to many

Dengan adanya skema relasi ini dapat membantu kami dalam menginformasikan dalam desain query dan query apa saja yang kami perlukan dalam pembuatan database kami. Selain itu dapat merepresentasikan database yang akan kami buat.

## DDL DAN DML

Data Definition Language (DDL) memungkinkan kita untuk mendefinisikan atribut-atribut database. sintaks query yang dapat digunakan untuk DDL adalah sebagai berikut.

- CREATE**, create perintah yang digunakan untuk membuat, termasuk membuat database tabel baru.  
mysql> CREATE TABLE nama\_tabel ( isi nama\_field 1 tipe\_data(size\_data)constraint (nama\_field\_1)  
);
- DESC**, desc perintah untuk melihat detail tabel.  
mysql> DESCRIBE nama\_tabel;

## Pengaplikasian CREATE dan DESC

```
5 #DDL
6 #Tabel supermarkets
7 CREATE TABLE Supermarkets(
8     invoice_id int(12),
9     branch varchar(5),
10     constraint pkSupermarket primary key(invoice_id)
11 );
12 desc Supermarkets;
```

Result Grid

Field	Type	Null	Key	Default	Extra
invoice_id	int	NO	PRI	NULL	
branch	varchar(5)	YES		NULL	

```
16 #Tabel Location
17 create table Location(
18     branch varchar(5),
19     city varchar(20),
20     constraint pkBranch primary key(branch)
21 );
22 desc Location;
```

Result Grid

Field	Type	Null	Key	Default	Extra
branch	varchar(5)	NO	PRI	NULL	
city	varchar(20)	YES		NULL	

```
25 #Tabel Product
26 create table Product(
27     product_id varchar(5),
28     product_line varchar(25),
29     constraint pkProducts primary key(product_id)
30 );
31 desc Product;
```

Result Grid

Field	Type	Null	Key	Default	Extra
product_id	varchar(5)	NO	PRI	NULL	
product_line	varchar(25)	YES		NULL	

```
33 #Tabel Items
34 create table Items(
35     invoice_id int(12),
36     product_id varchar(5),
37     date_time date,
38     unit_price int(10),
39     quantity int(10),
40     total int(15),
41     constraint fk1 foreign key (invoice_id) REFERENCES Supermarkets (invoice_id),
42     constraint fk2 foreign key (product_id) REFERENCES Products (product_id)
43 );
44 desc Items;
```

Result Grid

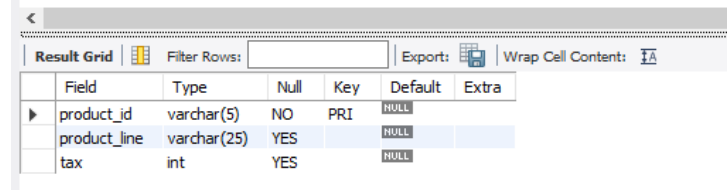
Field	Type	Null	Key	Default	Extra
invoice_id	int	YES	MUL	NULL	
product_id	varchar(5)	YES	MUL	NULL	
date_time	date	YES		NULL	
unit_price	int	YES		NULL	
quantity	int	YES		NULL	
total	int	YES		NULL	



- c. **ALTER**, alter perintah yang digunakan untuk menambah field/ constraint pada tabel.  
mysql> ALTER TABLE nama\_tabel ADD new field name( tipe\_data);

#### Pengaplikasian ALTER and ADD

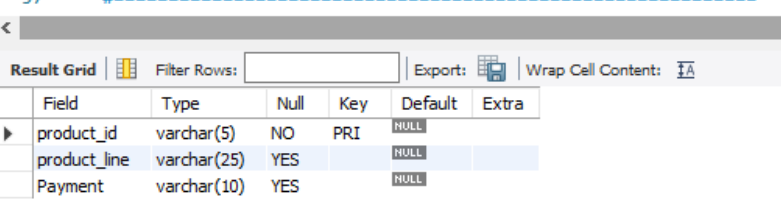
```
47 • alter table Product add tax int(10);
48 • desc Product;
49 • #=====
```



Field	Type	Null	Key	Default	Extra
product_id	varchar(5)	NO	PRI	NULL	
product_line	varchar(25)	YES		NULL	
tax	int	YES		NULL	

- d. **ALTER MODIFY**, alter modify perintah ini digunakan untuk mengubah definisi field/constraint pada suatu tabel.  
mysql> ALTER TABLE nama\_tabel modify edited\_field\_name (tipe\_data);

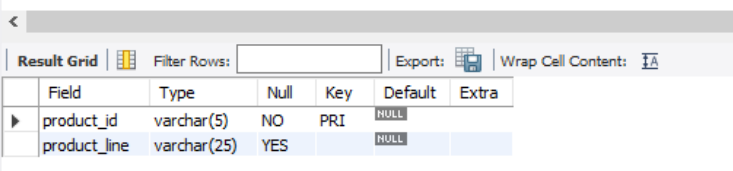
```
55 • alter table Product modify Payment varchar(10);
56 • desc Product;
57 • #=====
```



Field	Type	Null	Key	Default	Extra
product_id	varchar(5)	NO	PRI	NULL	
product_line	varchar(25)	YES		NULL	
Payment	varchar(10)	YES		NULL	

- e. **ALTER DROP**, alter drop perintah ini digunakan untuk mengubah field pada suatu tabel.  
mysql> ALTER TABLE nama\_tabel modify edited\_field\_name (tipe\_data);

```
59 • alter table Product drop Payment;
60 • desc Product;
61 • #=====
```



Field	Type	Null	Key	Default	Extra
product_id	varchar(5)	NO	PRI	NULL	
product_line	varchar(25)	YES		NULL	

Data Manipulation Language (DML) memungkinkan kita untuk memanipulasi data dalam tabel database. Sintaks query yang dapat digunakan untuk DML adalah sebagai berikut.

1. **INSERT**, insert adalah sintaks yang berfungsi untuk menambahkan baris data baru ke tabel.  
mysql> INSERT INTO nama\_tabel VALUES ( isi\_data\_kolom\_1, ..... isi\_data\_kolom\_n );

## Pengaplikasian INSERT

- Tabel Supermarkets

```
3      #Table Supermarkets
4 •    INSERT INTO Supermarkets (invoice_id, branch) values ("750678428", "A");
5 •    INSERT INTO Supermarkets (invoice_id, branch) values ("226313081", "C");
6 •    INSERT INTO Supermarkets (invoice_id, branch) values ("631413108", "A");
7 •    INSERT INTO Supermarkets (invoice_id, branch) values ("123191176", "A");
8 •    INSERT INTO Supermarkets (invoice_id, branch) values ("373737910", "A");
9 •    INSERT INTO Supermarkets (invoice_id, branch) values ("699143026", "C");
10 •   INSERT INTO Supermarkets (invoice_id, branch) values ("355535943", "A");
11 •   INSERT INTO Supermarkets (invoice_id, branch) values ("315225665", "C");
12 •   INSERT INTO Supermarkets (invoice_id, branch) values ("665329167", "A");
13 •   INSERT INTO Supermarkets (invoice_id, branch) values ("692925582", "B");
14 •   INSERT INTO Supermarkets (invoice_id, branch) values ("351620822", "B");
15 •   INSERT INTO Supermarkets (invoice_id, branch) values ("529563974", "B");
16 •   INSERT INTO Supermarkets (invoice_id, branch) values ("365640515", "A");
17 •   INSERT INTO Supermarkets (invoice_id, branch) values ("252562699", "A");
18 •   INSERT INTO Supermarkets (invoice_id, branch) values ("829343910", "A");
19 •   INSERT INTO Supermarkets (invoice_id, branch) values ("299461805", "B");
20 •   INSERT INTO Supermarkets (invoice_id, branch) values ("656959349", "A");
21 •   INSERT INTO Supermarkets (invoice_id, branch) values ("765266951", "A");
22 •   INSERT INTO Supermarkets (invoice_id, branch) values ("329621586", "A");
23 •   INSERT INTO Supermarkets (invoice_id, branch) values ("319503348", "B");
24 •   INSERT INTO Supermarkets (invoice_id, branch) values ("300714605", "B");
25 •   INSERT INTO Supermarkets (invoice_id, branch) values ("371855789", "B");
26 •   INSERT INTO Supermarkets (invoice_id, branch) values ("273166619", "A");
27 •   INSERT INTO Supermarkets (invoice_id, branch) values ("636488204", "A");
28 •   INSERT INTO Supermarkets (invoice_id, branch) values ("549591358", "A");
29 •   INSERT INTO Supermarkets (invoice_id, branch) values ("227035010", "B");
30 •   INSERT INTO Supermarkets (invoice_id, branch) values ("649296775", "A");
31 •   INSERT INTO Supermarkets (invoice_id, branch) values ("189174241", "A");
32 •   INSERT INTO Supermarkets (invoice_id, branch) values ("145949061", "A");
33 •   INSERT INTO Supermarkets (invoice_id, branch) values ("848627243", "B");
34 •   INSERT INTO Supermarkets (invoice_id, branch) values ("871798483", "B");
35 •   INSERT INTO Supermarkets (invoice_id, branch) values ("149716266", "B");
```

Result Grid			Filter Rows:
	invoice_id	branch	
▶	123191176	A	
	145949061	A	
	149716266	B	
	189174241	A	
	226313081	C	
	227035010	B	
	252562699	A	
	273166619	A	
	299461805	B	
	300714605	B	
	315225665	C	
	319503348	B	
	329621586	A	
	351620822	B	
	355535943	A	
	365640515	A	
	371855789	B	
	373737910	A	
	529563974	B	
	631413108	A	
	636488204	A	
	649296775	A	

Result Grid			Filter Rows:
	invoice_id	branch	
	300714605	B	
	315225665	C	
	319503348	B	
	329621586	A	
	351620822	B	
	355535943	A	
	365640515	A	
	371855789	B	
	373737910	A	
	529563974	B	
	631413108	A	
	636488204	A	
	649296775	A	
	656959349	A	
	665329167	A	
	692925582	B	
	699143026	C	
	750678428	A	
	765266951	A	
	829343910	A	
	871798483	B	
*	NULL	NULL	

- Tabel Location

Result Grid			Filter Rows:
	branch	city	
▶	A	Yangon	
	B	Mandalay	
	C	Naypyitaw	
	D	Indonesia	
*	NULL	NULL	

- Tabel Product

```
#Table Product
▶ INSERT INTO Product (product_id, product_line) values ("P1","Health and beauty");
▶ INSERT INTO Product (product_id, product_line) values ("P2","Electronic accessories");
▶ INSERT INTO Product (product_id, product_line) values ("P3","Home and lifestyle");
▶ INSERT INTO Product (product_id, product_line) values ("P4","Sports and travel");
▶ INSERT INTO Product (product_id, product_line) values ("P5","Food and beverages");
▶ INSERT INTO Product (product_id, product_line) values ("P6","Fashion accessories");
▶ INSERT INTO Product (product_id, product_line) values ("P7","Healty and travel");
```






Result Grid			Filter Rows:
	product_id	product_line	
▶	P1	Health and beauty	
	P2	Electronic accessories	
	P3	Home and lifestyle	
	P4	Sports and travel	
	P5	Food and beverages	
	P6	Fashion accessories	
	P7	Healty and travel	
*	NULL	NULL	

- Tabel Items

```

64 #Table Items
65 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("750678428", "P1", "2019-1-05", "74.69", "7", "5489.715");
66 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("226313081", "P2", "2019-3-08", "15.28", "5", "80.22");
67 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("631413108", "P3", "2019-3-03", "46.33", "7", "3405.255");
68 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("123191176", "P1", "2019-1-27", "58.22", "8", "489.048");
69 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("373737910", "P4", "2019-1-27", "86.31", "7", "6343.785");
70 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("699143026", "P2", "2019-3-25", "85.39", "7", "6276.165");
71 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("355535943", "P2", "2019-2-25", "68.84", "6", "433.692");
72 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("315225665", "P3", "2019-2-24", "73.56", "10", "772.38");
73 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("665329167", "P1", "2019-1-10", "36.26", "2", "76.146");
74 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("692925582", "P5", "2019-2-20", "54.84", "3", "172.746");
75 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("351620822", "P6", "2019-2-06", "14.48", "4", "60.816");
76 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("529563974", "P2", "2019-3-09", "25.51", "4", "107.142");
77 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("365640515", "P2", "2019-2-12", "46.95", "5", "2464.875");
78 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("252562699", "P5", "2019-2-07", "43.19", "10", "453.495");
79 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("829343910", "P1", "2019-3-29", "71.38", "10", "749.49");
80 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("299461805", "P4", "2019-1-15", "93.72", "6", "590.436");
81 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("656959349", "P1", "2019-3-11", "68.93", "7", "5066.355");
82 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("765266951", "P4", "2019-1-01", "72.61", "6", "457.443");
83 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("329621586", "P5", "2019-1-21", "54.67", "3", "1722.105");
84 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("319503348", "P3", "2019-3-11", "40.3", "2", "84.63");
85 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("300714605", "P2", "2019-2-25", "86.04", "5", "451.71");
86 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("371855789", "P1", "2019-3-05", "87.98", "3", "277.137");
87 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("273166619", "P3", "2019-3-15", "33.2", "2", "69.72");
88 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("636488204", "P2", "2019-2-17", "34.56", "5", "181.44");
89 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("549591358", "P4", "2019-3-02", "88.63", "3", "2791.845");
90 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("227035010", "P3", "2019-3-22", "52.59", "8", "441.756");
91 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("649296775", "P6", "2019-2-08", "33.52", "1", "35.196");
92 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("189174241", "P6", "2019-3-10", "83.67", "2", "184.107");
93 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("145949061", "P5", "2019-1-25", "88.36", "5", "463.89");
94 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("848627243", "P1", "2019-3-15", "24.89", "9", "2352.105");
95 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("871798483", "P6", "2019-2-25", "94.13", "5", "4941.825");
96 • INSERT INTO Items (invoice_id, product_id, date_time, unit_price, quantity, total) values ("149716266", "P4", "2019-1-28", "78.07", "9", "7377.615");

```

Result Grid	 Filter Rows:	<input type="text"/>	Edit:				Ex
	invoice_id	product_id	date_time	unit_price	quantity	total	
	123191176	P1	2019-01-27	58	8	489	
	145949061	P5	2019-01-25	88	5	464	
	149716266	P4	2019-01-28	78	9	7378	
	189174241	P6	2019-03-10	84	2	184	
	226313081	P2	2019-03-08	15	5	80	
	227035010	P3	2019-03-22	53	8	442	
	252562699	P5	2019-02-07	43	10	453	
	273166619	P3	2019-03-15	33	2	70	
	299461805	P4	2019-01-15	94	6	590	
	300714605	P2	2019-02-25	86	5	452	
	315225665	P3	2019-02-24	74	10	772	
	319503348	P3	2019-03-11	40	2	85	
	329621586	P5	2019-01-21	55	3	1722	
	351620822	P6	2019-02-06	14	4	61	
	355535943	P2	2019-02-25	69	6	434	
	365640515	P2	2019-02-12	47	5	2465	
	371855789	P1	2019-03-05	88	3	277	
	373737910	P4	2019-01-27	86	7	6344	
	529563974	P2	2019-03-09	26	4	107	

Result Grid

Filter Rows:

Edit:

Ex

	invoice_id	product_id	date_time	unit_price	quantity	total
	355535943	P2	2019-02-25	69	6	434
	365640515	P2	2019-02-12	47	5	2465
	371855789	P1	2019-03-05	88	3	277
	373737910	P4	2019-01-27	86	7	6344
	529563974	P2	2019-03-09	26	4	107
	549591358	P4	2019-03-02	89	3	2792
	631413108	P3	2019-03-03	46	7	3405
	636488204	P2	2019-02-17	35	5	181
	649296775	P6	2019-02-08	34	1	35
	656959349	P1	2019-03-11	69	7	5066
	665329167	P1	2019-01-10	36	2	76
	692925582	P5	2019-02-20	55	3	173
	699143026	P2	2019-03-25	85	7	6276
	750678428	P1	2019-01-05	75	7	5490
	765266951	P4	2019-01-01	73	6	457
	829343910	P1	2019-03-29	71	10	749
	848627243	P1	2019-03-15	25	9	2352
	871798483	P6	2019-02-25	94	5	4942
	NULL	NULL	NULL	NULL	NULL	NULL

## 2. UPDATE, update adalah sintaks yang memungkinkan untuk memperbarui data tabel.

mysql> UPDATE nama\_tabel SET nama\_kolom\_n = isi\_data\_n ;

atau

mysql> UPDATE nama\_tabel SET nama\_kolom\_n = isi\_data\_n WHERE nama\_kolom = kondisi\_nilai\_dicari;

## Pengaplikasian UPDATE

- Tabel Supermarkets

```
38 • UPDATE Supermarkets SET branch = "C" WHERE invoice_id = 149716266;
```

invoice_id	branch
123191176	A
145949061	A
149716266	C
189174241	A
226313081	C

- Tabel Location

```
48 • UPDATE Location SET city = "Jakarta" WHERE branch = "D";
```

branch	city
A	Yangon
B	Mandalay
C	Naypyitaw
D	Jakarta
NULL	NULL

- Tabel Product

```
61 • UPDATE Product SET product_line = "Home and Food" WHERE product_id = "P7";
```

product_id	product_line
P1	Health and beauty
P2	Electronic accessories
P3	Home and lifestyle
P4	Sports and travel
P5	Food and beverages
P6	Fashion accessories
P7	Home and Food
NULL	NULL

- Tabel Items

```
99 • UPDATE Items SET product_id = "P7" where invoice_id = "149716266";
```

invoice_id	product_id	date_time	unit_price	quantity	total
123191176	P1	2019-01-27	58	8	489
145949061	P5	2019-01-25	88	5	464
149716266	P7	2019-01-28	78	9	7378
189174241	P6	2019-03-10	84	2	184
226313081	P2	2019-03-08	15	5	80

3. **DELETE**, Delete adalah sintaks yang dapat menghapus baris data. Sintaks delete fungsinya mirip dengan sintaks truncate. Perbedaan sintaks delete adalah dapat menentukan dengan

tepat apa yang ingin dihapus, sedangkan truncate memungkinkan untuk menghapus semua record yang terdapat dalam sebuah tabel.

```
mysql> TRUNCATE TABLE nama_tabel ;
```

atau

```
mysql> DELETE FROM nama_tabel ;
```

atau

```
mysql> DELETE FROM nama_tabel WHERE nama_kolom = kondisi_nilai_dicari ;
```

4. **SELECT**, Select adalah sintaks yang digunakan untuk mengambil data dari objek database seperti tabel.

```
mysql> SELECT * FROM nama_tabel ;
```

atau

```
mysql> SELECT nama_kolom FROM nama_tabel ;
```

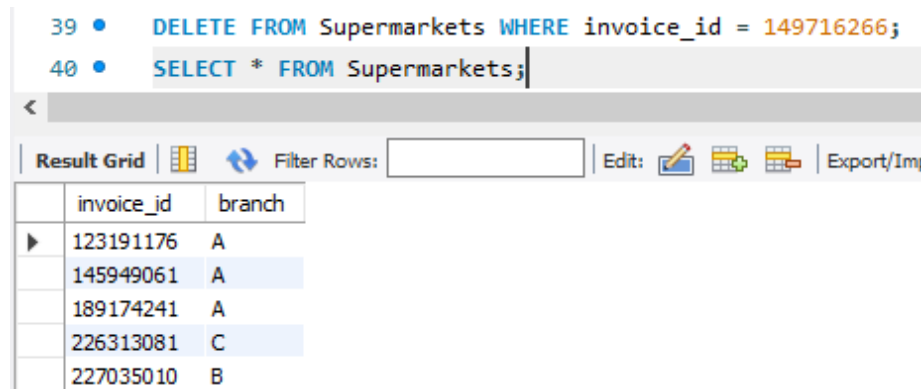
atau

```
mysql> SELECT nama_kolom FROM nama_tabel WHERE nama_kolom = kondisi_nilai_dicari
```

### Pengaplikasian DELETE dan SELECT

- Tabel Supermarkets

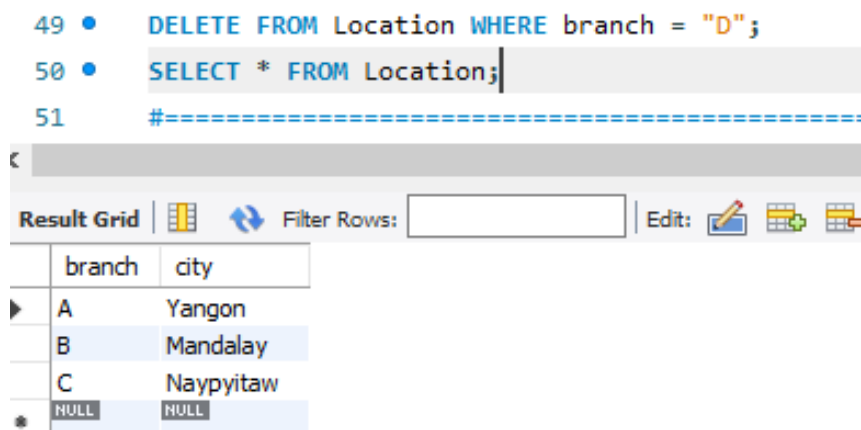
```
39 • DELETE FROM Supermarkets WHERE invoice_id = 149716266;
40 • SELECT * FROM Supermarkets;
```



	invoice_id	branch
▶	123191176	A
	145949061	A
	189174241	A
	226313081	C
	227035010	B

- Tabel Location

```
49 • DELETE FROM Location WHERE branch = "D";
50 • SELECT * FROM Location;
51 #=====
```



	branch	city
▶	A	Yangon
	B	Mandalay
	C	Naypyitaw
*	NULL	NULL

- Tabel Product

```

62 • DELETE FROM Product WHERE product_id = "P7";
63 • SELECT * FROM Product;

```

product_id	product_line
P1	Health and beauty
P2	Electronic accessories
P3	Home and lifestyle
P4	Sports and travel
P5	Food and beverages
P6	Fashion accessories
NULL	NULL

- Tabel Items

```

100 • DELETE FROM Items WHERE invoice_id = "149716266";
101 • SELECT * FROM Items;

```

invoice_id	product_id	date_time	unit_price	quantity	total
123191176	P1	2019-01-27	58	8	489
145949061	P5	2019-01-25	88	5	464
189174241	P6	2019-03-10	84	2	184
226313081	P2	2019-03-08	15	5	80

## ADVANCE QUERY

### 1. Clause Function

Clauses adalah adalah fungsi bawaan yang tersedia bagi kita di SQL. Dengan bantuan clauses, kita dapat menangani data yang disimpan dalam tabel dengan mudah. Clauses membantu kami memfilter dan menganalisis data dengan cepat. Ketika kami memiliki sejumlah besar data yang disimpan dalam database, kami menggunakan clause untuk melakukan query dan mendapatkan data yang dibutuhkan oleh pengguna. Beberapa contoh clauses adalah where, and, or, like, top, dll.

#### Pengaplikasian Clause Functions

```

4 # Advance Query
5 # Clause Functions
6 # Mengetahui kuantitas barang pemberian dan total pembelian disetiap invoice id
7 • SELECT invoice_id, quantity, total
8 FROM Items;

```

invoice_id	quantity	total
123191176	8	489
145949061	5	464
189174241	2	184
226313081	5	80
227035010	8	442
252562699	10	453
273166619	2	70
299461805	6	590
300714605	5	452



18

# Bulan Januari 2019

19

• SELECT \* FROM Items

20

WHERE date\_time BETWEEN '2019-01-01' AND '2019-01-31';

Result Grid

Filter Rows:

Edit:

Export/Imp

	invoice_id	product_id	date_time	unit_price	quantity	total
▶	123191176	P1	2019-01-27	58	8	489
	145949061	P5	2019-01-25	88	5	464
	299461805	P4	2019-01-15	94	6	590
	329621586	P5	2019-01-21	55	3	1722
	373737910	P4	2019-01-27	86	7	6344
	665329167	P1	2019-01-10	36	2	76
	750678428	P1	2019-01-05	75	7	5490
	765266951	P4	2019-01-01	73	6	457
*	NULL	NULL	NULL	NULL	NULL	NULL

24

# Bulan Februari 2019

25

• SELECT \* FROM Items

26

WHERE date\_time BETWEEN '2019-02-01' AND '2019-02-28';

Result Grid

Filter Rows:

Edit:

Export/Imp

	invoice_id	product_id	date_time	unit_price	quantity	total
▶	252562699	P5	2019-02-07	43	10	453
	300714605	P2	2019-02-25	86	5	452
	315225665	P3	2019-02-24	74	10	772
	351620822	P6	2019-02-06	14	4	61
	355535943	P2	2019-02-25	69	6	434
	365640515	P2	2019-02-12	47	5	2465
	636488204	P2	2019-02-17	35	5	181
	649296775	P6	2019-02-08	34	1	35
	692925582	P5	2019-02-20	55	3	173
	871798483	P6	2019-02-25	94	5	4942
*	NULL	NULL	NULL	NULL	NULL	NULL

30

# Bulan Maret 2019

31

• SELECT \* FROM Items

32

WHERE date\_time BETWEEN '2019-03-01' AND '2019-03-31';

Result Grid

Filter Rows:

Edit:

Export/Imp

	invoice_id	product_id	date_time	unit_price	quantity	total
▶	189174241	P6	2019-03-10	84	2	184
	226313081	P2	2019-03-08	15	5	80
	227035010	P3	2019-03-22	53	8	442
	273166619	P3	2019-03-15	33	2	70
	319503348	P3	2019-03-11	40	2	85
	371855789	P1	2019-03-05	88	3	277
	529563974	P2	2019-03-09	26	4	107
	549591358	P4	2019-03-02	89	3	2792
	631413108	P3	2019-03-03	46	7	3405
	656959349	P1	2019-03-11	69	7	5066
	699143026	P2	2019-03-25	85	7	6276
	829343910	P1	2019-03-29	71	10	749
	848627243	P1	2019-03-15	25	9	2352
*	NULL	NULL	NULL	NULL	NULL	NULL

## 2. Aggregation Function

Aggregate functions adalah fungsi yang dilakukan di atas satu atau lebih nilai dan mengembalikan satu nilai. Ada lima jenis utama aggregate functions yang penting untuk



pengkodean SQL:

- a) COUNT(), mengembalikan jumlah baris dalam kolom, tidak termasuk nilai NULL. Jika Anda hanya ingin menghitung jumlah nilai yang berbeda dalam kolom, Anda dapat menggunakan COUNT(DISTINCT).
- b) COUNTIF(), adalah perpanjangan dari COUNT di mana ia mengembalikan jumlah baris yang memenuhi kondisi.
- c) SUM(), mengembalikan jumlah nilai non-null — dengan kata lain, ia menambahkan semua nilai dalam kolom. Jangan bingung ini dengan COUNT. COUNT mengembalikan jumlah baris dalam kolom, sementara SUM menambahkan semua nilai dalam kolom.
- d) AVG(), hanya mengembalikan rata-rata kolom dengan nilai non-null. Secara matematis, AVG menjumlahkan nilai kolom tertentu dan kemudian membaginya dengan jumlah baris yang sesuai.
- e) MIN()/ MAX(), cukup mengembalikan nilai minimum dan nilai maksimum masing-masing untuk kolom.

### Pengaplikasian Aggregation Functions

```
10 # Aggregation Functions
11 • SELECT AVG(unit_price), MAX(unit_price), MIN(unit_price), SUM(unit_price), COUNT(unit_price) FROM
12 items;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	AVG(unit_price)	MAX(unit_price)	MIN(unit_price)	SUM(unit_price)	COUNT(unit_price)
▶	59.3548	94	14	1840	31

```
13 • SELECT AVG(quantity), MAX(quantity), MIN(quantity), SUM(quantity), COUNT(quantity) FROM
14 items;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	AVG(quantity)	MAX(quantity)	MIN(quantity)	SUM(quantity)	COUNT(quantity)
▶	5.3871	10	1	167	31

```
15 • SELECT AVG(total), MAX(total), MIN(total), SUM(total), COUNT(total) FROM
16 items;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	AVG(total)	MAX(total)	MIN(total)	SUM(total)	COUNT(total)
▶	1531.7742	6344	35	47485	31

```
21 • SELECT SUM(total), COUNT(total) FROM Items
22 WHERE date_time BETWEEN '2019-01-01' AND '2019-01-31';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	SUM(total)	COUNT(total)
▶	15632	8

```

27 • SELECT SUM(total), COUNT(total) FROM Items
28 WHERE date_time BETWEEN '2019-02-01' AND '2019-02-28';

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	SUM(total)	COUNT(total)		
▶	9968	10		

```

33 • SELECT SUM(total), COUNT(total) FROM Items
34 WHERE date_time BETWEEN '2019-03-01' AND '2019-03-31';

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	SUM(total)	COUNT(total)		
▶	21885	13		

```

45 #Tabel Pendapatan
46 • INSERT INTO Pendapatan (bulan, jumlah_pendapatan) values ("Januari", "15632");
47 • INSERT INTO Pendapatan (bulan, jumlah_pendapatan) values ("Februari", "9968");
48 • INSERT INTO Pendapatan (bulan, jumlah_pendapatan) values ("Maret", "21885");
49 • SELECT * FROM Pendapatan;

```

Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
no_	bulan	jumlah_pendapatan			
▶ 1	Januari	15632			
2	Februari	9968			
3	Maret	21885			
*	NULL	NULL			

### 3. Group By

Fungsi untuk mengelompokkan data dalam sebuah kolom yang ditunjuk. Fungsi ini akan menghasilkan kelompok data dengan menghilangkan data yang sama dalam satu tabel. Maka apabila dalam satu kolom terdapat beberapa data yang sama maka data yang akan ditampilkan hanya salah satu.

#### Pengaplikasian Group By

```

60 # GROUP BY
61 • SELECT product_id, count(*)
62 FROM Items
63 GROUP BY product_id;

```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	product_id	count(*)		
▶	P1	7		
	P5	4		
	P6	4		
	P2	7		
	P3	5		
	P4	4		

```

65 • SELECT branch, count(*)
66 FROM Supermarkets
67 GROUP BY branch;

```

branch	count(*)
A	17
C	3
B	9

### Pengaplikasian Having

```

64 # Having
65 • SELECT invoice_id, product_id, unit_price, quantity, total FROM Items
66 Having quantity >= 10;

```

invoice_id	product_id	unit_price	quantity	total
252562699	P5	43	10	453
315225665	P3	74	10	772
829343910	P1	71	10	749
NULL	NULL	NULL	NULL	NULL

```

68 • SELECT invoice_id, product_id, unit_price, quantity, total FROM Items
69 Having unit_price >= 90;

```

invoice_id	product_id	unit_price	quantity	total
299461805	P4	94	6	590
871798483	P6	94	5	4942
NULL	NULL	NULL	NULL	NULL

```

71 • SELECT invoice_id, product_id, unit_price, quantity, total FROM Items
72 Having total >= 5000;

```

invoice_id	product_id	unit_price	quantity	total
373737910	P4	86	7	6344
656959349	P1	69	7	5066
699143026	P2	85	7	6276
750678428	P1	75	7	5490
NULL	NULL	NULL	NULL	NULL

4. **Join** adalah penggabungan table yang dilakukan melalui kolom / key tertentu yang memiliki nilai terkait untuk mendapatkan satu set data dengan informasi lengkap. Join terdiri dari 3 jenis yaitu :

- **INNER JOIN** adalah tipe join yang akan kita bahas pertama. Tipe join ini akan mengambil semua row dari table asal dan table tujuan dengan kondisi nilai key yang terkait saja, dan jika tidak maka row tersebut tidak akan muncul.

- **LEFT JOIN** atau biasa juga dikenal dengan LEFT OUTER JOIN merupakan perintah join untuk menampilkan semua data sebelah kiri dari table yang di joinkan dan menampilkan data sebelah kanan yang cocok dengan kondisi join. Jika tidak ditemukan kecocokan, maka akan di set NULL secara otomatis.
- **RIGHT JOIN** Kebalikan dari LEFT JOIN adalah RIGHT JOIN, atau biasa juga dikenal dengan RIGHT OUTER JOIN. RIGHT JOIN akan menampilkan semua data yang ada di table sebelah kanan dan mencari kecocokan key pada table sebelah kiri.

## Pengaplikasian Join

```

73 # Join
74 • SELECT t1.invoice_id, t2.branch, t1.product_id, t1.unit_price, t1.quantity, t1.total
75 FROM Items t1
76 JOIN Supermarkets t2 ON t1.invoice_id = t2.invoice_id;

```

invoice_id	branch	product_id	unit_price	quantity	total
123191176	A	P1	58	8	489
145949061	A	P5	88	5	464
189174241	A	P6	84	2	184
226313081	C	P2	15	5	80
227035010	B	P3	53	8	442
252562699	A	P5	43	10	453
273166619	A	P3	33	2	70
299461805	B	P4	94	6	590
300714605	B	P2	86	5	452

```

78 • SELECT t1.invoice_id, t1.branch, t2.city
79 FROM Supermarkets t1
80 JOIN Location t2 ON t1.branch = t2.branch;

```

invoice_id	branch	city
123191176	A	Yangon
145949061	A	Yangon
189174241	A	Yangon
226313081	C	Naypyitaw
227035010	B	Mandalay
252562699	A	Yangon
273166619	A	Yangon
299461805	B	Mandalay

```

82 • SELECT t1.invoice_id, t1.branch, t2.city, t3.product_id, t3.unit_price, t3.quantity, t3.total
83 FROM Supermarkets t1
84 JOIN Location t2 ON t1.branch = t2.branch
85 JOIN Items t3 ON t1.invoice_id = t3.invoice_id;

```

invoice_id	branch	city	product_id	unit_price	quantity	total
123191176	A	Yangon	P1	58	8	489
145949061	A	Yangon	P5	88	5	464
189174241	A	Yangon	P6	84	2	184
226313081	C	Naypyitaw	P2	15	5	80
227035010	B	Mandalay	P3	53	8	442
252562699	A	Yangon	P5	43	10	453
273166619	A	Yangon	P3	33	2	70
299461805	B	Mandalay	P4	94	6	590
300714605	B	Mandalay	P2	86	5	452
315225665	C	Naypyitaw	P3	74	10	772
319503348	B	Mandalay	P3	40	2	85

5. **Subquery** atau Inner query atau Nested query adalah query dalam query SQL lain dan tertanam dalam klausa WHERE. Sebuah subquery digunakan untuk mengembalikan data yang akan digunakan dalam query utama sebagai syarat untuk lebih membatasi data yang akan diambil.

### Pengaplikasian Subquery

```
74 # Subquery
75 • SELECT *
76 FROM Items
77 where quantity=(
78     SELECT max(quantity)
79     FROM Items
80 );
```

invoice_id	product_id	date_time	unit_price	quantity	total
252562699	P5	2019-02-07	43	10	453
315225665	P3	2019-02-24	74	10	772
829343910	P1	2019-03-29	71	10	749
NULL	NULL	NULL	NULL	NULL	NULL

```
82 • SELECT *
83 FROM Items
84 where unit_price=(
85     SELECT max(unit_price)
86     FROM Items
87 );
```

invoice_id	product_id	date_time	unit_price	quantity	total
299461805	P4	2019-01-15	94	6	590
871798483	P6	2019-02-25	94	5	4942
NULL	NULL	NULL	NULL	NULL	NULL

```
89 • SELECT *
90 FROM Items
91 where total=(
92     SELECT max(total)
93     FROM Items
94 );
```

invoice_id	product_id	date_time	unit_price	quantity	total
373737910	P4	2019-01-27	86	7	6344
NULL	NULL	NULL	NULL	NULL	NULL

## INSIGHT

Dari hasil analysis yang dilakukan pada data yang ada dengan menggunakan berbagai syntax yang ada pada SQL, berikut insight yang didapatkan :

1. Pada tabel items di kolom unit price didapatkan bahwa rata-rata harga setiap produk dalam \$ adalah 59.3548, harga maksimumnya adalah 94, harga minimumnya adalah 14, total harga adalah 1840 dengan jumlah 31 unit.
2. Pada tabel items di kolom quantity didapatkan bahwa rata-rata jumlah produk yang dibeli oleh pelanggan adalah 5.3871, jumlah maksimumnya adalah 10, jumlah minimumnya adalah 1, total produk yang dibeli oleh pelanggan adalah 167 dengan jumlah 31 unit.
3. Pada tabel items di kolom total didapatkan bahwa rata-rata total harga produk yang terjual adalah 59.3548, harga maksimumnya adalah 94, harga minimumnya adalah 14, total harga produk yang terjual adalah 1840 dengan jumlah 31 unit.
4. Pada bulan Januari ada 8 jenis invoice id yang tercatat melakukan penjualan dengan total harga product yang terjual adalah 15632. Pada bulan Februari ada 10 jenis invoice id yang tercatat melakukan penjualan dengan total harga product yang terjual adalah 9968.
5. Pada bulan Maret ada 13 jenis invoice id yang tercatat melakukan penjualan dengan total harga product yang terjual adalah 21885. Dapat disimpulkan bahwa penjualan terbanyak terjadi pada bulan Maret, disusul oleh bulan Januari dan Februari.
6. Jenis product yang paling banyak terjual yaitu dengan total 7 invoice id adalah P1(Health and beauty) dan P2(Electronic Accesories). Sedangkan jenis product dengan penjualan paling sedikit yaitu dengan total 4 invoice id adalah P4(Sport and Travel), P5(Food and Beverages), dan P6(Fashion Accesories).
7. Cabang supermarket yang paling banyak menjual product dengan total 17 invoice id adalah cabang A atau kota Yangon. Sedangkan yang paling sedikit menjual product dengan total 3 invoice id adalah cabang B atau kota Mandalay.
8. Jumlah invoice id yang membeli product dengan jumlah lebih dari atau sama dengan 10 ada 3. Jumlah invoice id yang menjual dengan harga product lebih dari sama dengan 90 ada 2. Jumlah invoice id yang menjual product dengan total lebih dari sama dengan 5000 ada 4.
9. Invoice id dengan no (373737910) adalah supermarkets dengan total harga product yang terjual paling banyak yaitu 6344. Invoice id dengan no (299461805 dan 871798483) adalah supermarkets dengan harga product yang paling mahal yaitu 94 per unit. Invoice id dengan no (252562699, 315225665 dan 829343910) adalah supermarkets dengan jumlah product yang paling banyak terjual yaitu 10 product.

## APPENDIX

1. Link referensi data : <https://www.kaggle.com/datasets/aungpyaeap/supermarket-sales>