

Penulis:

Ardhin Primadewi, S. Si, M.TI

Maimunah, S. Si, M.Cs

Editor:

Tuessi Ari Purnomo, S.T., M. Tech

Integrasi Data pada Sistem Informasi:

Kebutuhan Mengintegrasikan Data pada Sebuah Sistem Informasi dengan Mengoptimalkan SQL

**Integrasi Data pada Sistem
Informasi : Kebutuhan
mengintegrasikan data pada sebuah
Sistem Informasi dengan
mengoptimalkan SQL**

Penulis:

Ardhin Primadewi, S. Si, M. TI

Maimunah, S. Si, M. Cs

Editor:

Tuessi Ari Purnomo, S.T., M.Tech.

UU No 28 tahun 2014 tentang Hak Cipta

Fungsi dan sifat hak cipta Pasal 4

Hak Cipta sebagaimana dimaksud dalam Pasal 3 huruf a merupakan hak eksklusif yang terdiri atas hak moral dan hak ekonomi.

Pembatasan Pelindungan Pasal 26

Ketentuan sebagaimana dimaksud dalam Pasal 23, Pasal 24, dan Pasal 25 tidak berlaku terhadap:

- i. Penggunaan kutipan singkat Ciptaan dan/atau produk Hak Terkait untuk pelaporan peristiwa aktual yang ditujukan hanya untuk keperluan penyediaan informasi aktual;
- ii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk kepentingan penelitian ilmu pengetahuan;
- iii. Penggandaan Ciptaan dan/atau produk Hak Terkait hanya untuk keperluan pengajaran, kecuali pertunjukan dan Fonogram yang telah dilakukan pengumuman sebagai bahan ajar; dan
- iv. Penggunaan untuk kepentingan pendidikan dan pengembangan ilmu pengetahuan yang memungkinkan suatu Ciptaan dan/atau produk Hak Terkait dapat digunakan tanpa izin Pelaku Pertunjukan, Produser Fonogram, atau Lembaga Penyiaran.

Sanksi Pelanggaran Pasal 113

1. Setiap Orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf i untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan/atau pidana denda paling banyak Rp100.000.000 (seratus juta rupiah).
2. Setiap Orang yang dengan tanpa hak dan/atau tanpa izin Pencipta atau pemegang Hak Cipta melakukan pelanggaran hak ekonomi Pencipta sebagaimana dimaksud dalam Pasal 9 ayat (1) huruf c, huruf d, huruf f, dan/atau huruf h untuk Penggunaan Secara Komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan/atau pidana denda paling banyak Rp. 500.000.000,00 (lima ratus juta rupiah).

Integrasi Data pada Sistem Informasi : Kebutuhan mengintegrasikan data pada sebuah Sistem Informasi dengan mengoptimalkan SQL

ISBN: 978-623-7261-67-4

Hak Cipta 2022 pada Penulis

Hak penerbitan pada UNIMMA PRESS. Bagi mereka yang ingin memperbanyak sebagian isi buku ini dalam bentuk atau cara apapun harus mendapatkan izin tertulis dari penulis dan penerbit UNIMMA PRESS.

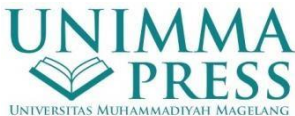
Penulis:

Ardhin Primadewi, S. Si, M. TI. 

Maimunah, S. Si, M. Cs 

Editor:

Tuessi Ari Purnomo, S.T., M.Tech. 



Penerbit:

UNIMMA PRESS

Gedung Rektorat Lt. 3 Kampus 2 Universitas Muhammadiyah
Magelang Jalan Mayjend Bambang Soegeng km.05, Mertoyudan,
Magelang 56172 Telp. (0293) 326945
E-Mail: unimmapress@ummgl.ac.id

Hak Cipta dilindungi
Undang-undang All Right
Reserved
Cetakan I, Januari 2022

DAFTAR ISI

Contents

BAB I Pengantar Basis Data	6
1.1. Data dan informasi	7
Studi Kasus 1.1.1	9
BAB II Pengantar Sistem.....	14
2.1. Siklus Hidup Pengembangan Sistem	14
2.2. Aplikasi Sistem Teknologi Informasi	14
2.3. Mode Bisnis (Business Model)	15
BAB III Pengantar Database dan SQL	20
3.1. Basis Data / Database	20
3.2. Komponen Sistem Basis Data	20
3.3. <i>Structured Query Language</i> (SQL).....	21
BAB IV SQL LEVEL	23
4.1. <i>SQL Beginners</i>	23
1. <i>Menampilkan Data Pada Tabel</i>	23
2. <i>Menampilkan Data menggunakan operator</i> ..	24
4.2. <i>SQL Advance</i>	26
1. Melakukan Query dalam Data Integration	26
a. Filtering data menggunakan Limit	26
b. Filtering data menggunakan is null	28

c.	Menggabungkan data menggunakan JOIN	30
d.	Menggabungkan data menggunakan GROUPING DATA.....	38
e.	Subqueries	45
f.	Derived Table	48
g.	Exists	52
h.	Pengembangan SQL Lainnya	54
2.	Konsep SQL JOIN	55
a.	INNER JOIN	58
b.	LEFT [OUTER] JOIN	58
c.	LEFT [OUTER] JOIN without Intersection...	59
d.	RIGHT [OUTER] JOIN	60
e.	RIGHT [OUTER] JOIN without Intersection	60
f.	FULL [OUTER] JOIN	61
g.	FULL [OUTER] JOIN without Intersection 61	
3.	SUBQUERY	62
4.	Konsep SQL UNION	63
a.	UNION OPERATOR.....	65
b.	INTERSECT OPERATOR.....	66
c.	EXCEPT OPERATOR.....	67
d.	PIPELINE	67
e.	ANALYSIS & REPORTING	68
5.	SQL Challenge	74

BAB I

Pengantar Basis Data ¹

“Data ibarat kompas yang memandu kita dalam pelaksanaan pembangunan. Tanpa data kita tidak dapat membuat perencanaan dan implementasi kegiatan dengan tepat sasaran.”

“Data yang disajikan ke publik banyak terjadi ketidakseragaman antara satu instansi dengan instansi lainnya sehingga diragukan keakuratan dan kevalidannya.”

(Pemprov KEPRI, 2018)

Data merupakan jiwa dalam sebuah instansi ataupun lembaga. Seperti cuplikan berita di atas, data dalam sebuah kantor pemerintahan menjadi dasar dalam perencanaan dan pelaksanaan pembangunan. Bagi pebisnis, data dapat berupa grafik, info-grafis, file dan

¹ “Sistem Teknologi Informasi : Pendekatan Terintegrasi” oleh Prof. Dr. Jogiyanto HM, MBA, Akt. Pada tahun 2005 (Yogyakarta : Andi Offset)

kumpulan catatan. Dalam dunia bisnis, pengumpulan data dan pencatatan data merupakan faktor yang paling penting agar bisnis mampu berkembang dengan baik. Bagi akademisi, data dapat berupa rekam jejak penelitian, data primer dan sekunder dalam penelitian dan data terkait sebuah topik penelitian. Dalam dunia akademik, data pada sebuah topik tertentu menjadi sangat berharga walau data tersebut sudah berusia 10 tahun. Karena data merepresentasikan pola tertentu yang dapat diteliti oleh para akademisi.

1.1. Data dan informasi

Bisnis di dunia semakin berkembang setiap tahunnya seiring dengan berkembangnya teknologi yang membantu perusahaan untuk mencapai tujuan tertentu secara mudah dan cepat. Para *start up* pun mulai berlomba-lomba menunjukkan bahwa teknologi mereka lah yang lebih unggul dibandingkan dengan kompetitornya sehingga banyak *start up* yang dapat bertahan bahkan berkembang hingga saat ini. Kira-kira hal apakah yang dapat membuat para *start up* dapat bertahan ? Jawabannya adalah “data”.

Kata “data” berasal dari bahasa Yunani “*datum*” yang berarti fakta, dan di dalam kamus bahasa Inggris ditulis dengan “*data*”. Data menggambarkan sebuah representasi fakta yang tersusun secara terstruktur. Data juga merepresentasikan deskripsi dari suatu objek atau kejadian (*event*). Data merupakan salah satu hal utama yang dikaji dalam masalah Teknologi Informasi dan Komunikasi (yang selanjutnya disebut TIK). Sedangkan TIK erat kaitannya dengan bidang kajian Informatika dan Ilmu Komputer.

Informatika dan Ilmu Komputer saat ini berkembang sangat pesat khususnya pada bidang kajian yang erat kaitannya dengan “data”. Beberapa kajian yang sudah banyak dibahas yaitu *Data System*, *Database Management System*, *Statistical Method*, *Data Mining*, dan *Artificial Intelligence*. Di saat yang sama, dibutuhkan kajian-kajian khusus terkait “data” yang dapat menjawab perkembangan teknologi pada Industri 4.0 yaitu *Big Data*. Beberapa kajian di atas yang dapat merubah “data” menjadi sebuah “informasi”.

Informasi merupakan sesuatu yang dihasilkan dari pengolahan data. Data yang sudah ada dikemas dan diolah

sedemikian rupa sehingga menjadi sebuah informasi yang berguna. Informasi menggambarkan kejadian yang nyata (fakta). Informasi didapatkan beberapa dari proses analisa data sederhana dan sisanya dari proses analisa lanjut (seperti *Statistical Method*, *Data Mining*, dan *Artificial Intelligence*).

Studi Kasus 1.1.1

Data dianggap sebagai ‘bahan bakar’ baru dalam menentukan keputusan bisnis. Dalam beberapa tahun terakhir, jumlah dan kecepatan produksi data telah meningkat drastis di banyak industri. Kemajuan produksi data pun membuat profesi **data scientist** semakin dicari.

Data menjadi napas bagi pengembangan bisnis start up, contohnya OLX. Untuk membangun produk yang *user-centric*, pengambilan, pengolahan, serta analisis data menjadi sangat penting. Satu kesalahan fatal yang terjadi

pada proses tersebut, akan membuat performa produk menjadi kurang baik.

Kamalesh Bathala, Director of Customers and Analytics OLX Indonesia, menyatakan bahwa sebuah produk yang baik dibangun melalui analisis data yang tepat.

“Di OLX, data diolah dan digunakan dengan sangat serius. Semua keputusan bisnis yang dilakukan berbasis data, misalnya pengembangan *user experience* atau keputusan strategi harga,” ujar Kamalesh, Kamis (7/2/2019) di Jakarta.

Data dianggap menjadi refleksi dari behavior pengguna sebuah platform tentang bagaimana mereka memanfaatkan platform dalam proses jual-beli. Mengolah dan membaca data dengan jeli dapat membantu OLX mengetahui kebiasaan pengguna

dalam memanfaatkan OLX dalam kegiatan jual-beli.

Dengan mengetahui kebiasaan pengguna, OLX dapat mengetahui kebutuhan mereka saat ini, serta memprediksi kebutuhan mereka yang akan datang. Dengan demikian, keputusan yang diambil akan tepat sasaran, contohnya pengembangan *product feature* yang relevan bagi para pengguna.

Data science sendiri merupakan bidang yang terus berevolusi seiring dengan perkembangan teknologi baru, termasuk alat yang digunakan dalam mengolah data. Namun demikian, sumber daya manusia pun menjadi faktor utama dalam proses tersebut. Untuk itu, profesi data scientist mulai banyak dicari oleh berbagai macam perusahaan, termasuk OLX. (WE Online Jakarta, 2019)

Pada studi kasus 1.1.1 tersebut dapat diketahui bahwa data menjadi dasar didapatkannya informasi bagi perusahaan OLX. Data yang dimaksud dalam berita tersebut antara lain : *data trend*, kecenderungan market, data pertumbuhan neraca, data global, dan data stabilitas negara. Seorang pebisnis *online* maupun *offline* harus mengambil langkah jitu dalam bisnisnya *based on data* (berdasarkan pada data). Maksudnya setiap langkah harus berdasarkan informasi dari sebuah analisis data, bukan hanya mengikuti tren saja. Dengan mempelajari data dan mendapatkan informasi, akan mudah mengatur, mengelola dan membuat strategi bisnis, instansi pemerintahan ataupun perusahaan besar. Pada gambar 1 di bawah ini salah satu data dan informasi yang sudah disajikan dalam bentuk yang mudah dipahami pembaca (*user friendly*) terkait bisnis *online* dan *start up* di Indonesia (Buii & Ecomeye, 2017).



Gambar 1 Data dan Informasi Laporan E-Commerce Indonesia yang bersumber dari Google Internal Data

BAB II

Pengantar Sistem

2.1. Siklus Hidup Pengembangan Sistem

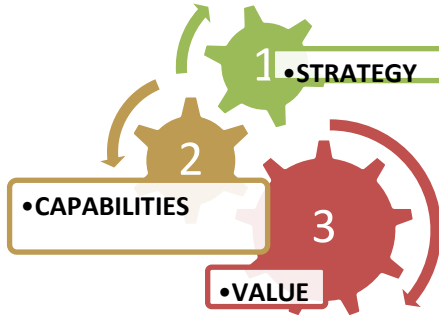
Siklus hidup pengembangan system dapat diartikan menjadi alur dalam pembuatan dan pengubahan system untuk mengembangkan serta menerapkan system informasi. Alur yang dilakukan pada siklus hidup pengembangan system diawal dengan perencanaan system, analisis system dan kebutuhan baik fungsional dan non fungsional.

Hasil dari analisis system dapat dilanjutkan pada design atau perancangan system secara terperinci sekaligus pengimplementasiannya terhadap system. System yang telah dibuat akan dilakukan uji coba serta dilakukan pemeliharaan system. Siklus ini memiliki tahapan alur yang jelas sehingga mudah untuk diterapkan.

2.2. Aplikasi Sistem Teknologi Informasi

Sistem Teknologi Informasi dapat diterapkan di internal atau eksternal organisasi.

2.3.Mode Bisnis (Business Model)



Gambar 2 Poin Utama dalam Business Model (Applegate, Austin, & Soule, 2009)

Business Model (Model Bisnis) mendefinisikan bagaimana organisasi berinteraksi dengan lingkungannya untuk mendefinisikan strategi khusus, membangun kapabilitas yang dibutuhkan untuk mengeksekusi strategi dan membuat Value dari seluruh stakeholder.

Business Model sangat berguna dalam memandu menganalisa strategi dan pembuatan keputusan. Business Model juga sangat berguna untuk memprediksi perkembangan bisnis ke depannya baik sesuai atau tidak sesuai dengan perencanaan bisnis (Applegate et al., 2009).

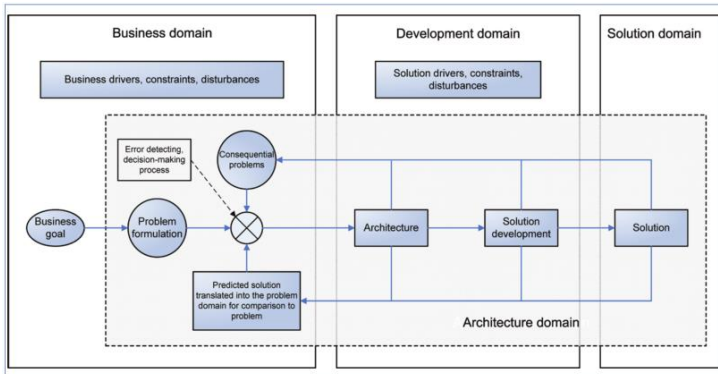
Strategi kompetitif menurut Applegate et al. (2009) yaitu dengan menjadi berbeda. Ada beberapa pilihan untuk mendefinisikan strategi kompetitif berdasarkan 4 poin utama, yaitu :

1. *Market positioning* : menentukan pilihan dari pelanggan yang dilayani, kebutuhan dan harapan yang akan ditemui, sumber yang akan digunakan oleh pelanggan.
2. *Product positioning* : menentukan pilihan produk dan jasa sesuai fitur yang ditawarkan dan harga yang akan dibebankan
3. *Business network positioning* : menentukan peran yang diambil oleh organisasi dan aktivitas yang melingkupinya terkait dengan jaringan supplier, produsen, distributor dan mitra
4. *Boundary positioning* : menentukan pasaran, produk dan bisnis yang TIDAK AKAN DILEWATI.

Dari penjelasan di atas, dapat disimpulkan bahwa *business process* menjadi poin utama dalam keberhasilan sebuah bisnis. Bisnis disini tidak hanya perusahaan atau lembaga berorientasi profit, namun juga seluruh lembaga

dan organisasi yang memiliki kepentingan dan anggaran dalam menjalankan ide bisnisnya. Secara visual, gambar 3 di bawah ini menjelaskan bahwa wilayah bisnis harus fokus pada hal yang mendasari tujuan bisnis, menjadi penghalang bisnis dan batasan-batasan dalam bisnis. Dari 3 aspek tersebut, akan diformulasikan berdasarkan pola-pola yang ditemukan selama bisnis itu berjalan. Formulasi diperkuat dengan adanya data-data dari *error*/kesalahan yang pernah dilakukan saat proses pembuatan keputusan.

Hasil dari analisa data tersebut, bisnis akan dikembangkan agar solusi yang diberikan tidak bias dari tujuan bisnis, menjadi penghalang bisnis dan batasan-batasan dalam bisnis. Secara visual, seluruh solusi akan digambarkan arsitekturnya. Secara proses berkelanjutan maka solusi akan dibandingkan dengan permasalahan yang ada.



Gambar 3 Arsitektur sistem berorientasi problem (Robertson-Dunn, 2012)

Pada gambar 4 di bawah ini menggambarkan *business process* dalam bisnis hubungannya dengan *IT architecture*.

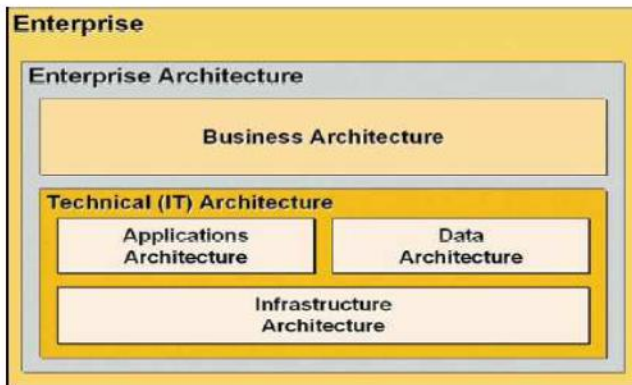


Figure 2. The classic enterprise architecture

Gambar 4 “Enterprise Architecture” secara klasik (Kozina, 2006)

BAB III

Pengantar Database dan SQL

3.1.Basis Data / Database

Merupakan himpunan kelompok data (arsip) yang saling berhubungan, yang diorganisasi sedemikian rupa, sehingga kelak dapat dimanfaatkan kembali dengan cepat.

3.2.Komponen Sistem Basis Data

Komponen yang ada dalam sebuah basis data / *database* yaitu (1) Perangkat Keras (Hardware) seperti komputer, memori, storage (*Harddisk*), *peripheral*, dll, (2) Sistem Operasi (*Operating System*) yang merupakan program yang menjalankan sistem komputer, mengendalikan *resource computer* dan melakukan berbagai operasi dasar sistem komputer, (3) Basis Data (*Database*) yang merupakan tempat menyimpan berbagai objek *database* (struktur tabel, indeks,dll), (4) DBMS (*Database Management System*) yang merupakan Perangkat lunak yang memaintain data dalam jumlah besar, (5) Pemakai (*User*) : Para pemakai *database* dan

sistem, (6) Aplikasi (perangkat lunak) lain dan (7) Program lain dalam DBMS.

3.3.Structured Query Language (SQL)

SQL adalah singkatan dari Structured Query Language. Sedangkan pengertian SQL adalah suatu bahasa (language) yang digunakan untuk mengakses data di dalam sebuah database relasional. SQL adalah sebuah bahasa permintaan database yang terstruktur. Bahasa SQL ini dibuat sebagai bahasa yang dapat merelasikan beberapa tabel dalam database maupun merelasikan antar database.

SQL sering juga disebut dengan istilah query, dan bahasa SQL secara praktiknya digunakan sebagai bahasa standar untuk manajemen database relasional. Hingga saat ini hampir seluruh server database atau software database mengenal dan mengerti bahasa SQL.

Dalam penggunaan SQL terdapat beberapa perintah yang berguna untuk mengakses dan memanajemen data yang terdapat dalam database. Jenis perintah SQL secara umum dibagi kepada tiga sub perintah, yaitu DDL (Data Definition Language), DML (Data Manipulation

Language), dan DCL (Data Control Language). Ketiga sub perintah tersebut sangat perlu untuk dipahami bagi anda yang ingin menguasai bahasa sql dan mahir dalam pembuatan database.

BAB IV

SQL LEVEL

4.1. SQL Beginners

1. Menampilkan Data Pada Tabel

Fungsi Select digunakan untuk menampilkan data pada tabel. Untuk menampilkan sebuah tabel ada beberapa kondisi yang bisa digunakan, antara lain :

1. SELECT ALL

Kondisi ini digunakan untuk menampilkan semua record dan semua field nya dalam sebuah tabel.

Query : `SELECT * FROM nama_tabel;`

2. SELECT FIELD

Kondisi ini digunakan untuk menampilkan semua baris / record yang ada, tetapi hanya kolom / field tertentu saja (kolom / field sesuai permintaan kondisi saja).

Query : `SELECT field1,field2,... FROM nama_tabel;`

3. SELECT RECORD (WHERE)

Kondisi ini digunakan untuk menampilkan isi dari baris / record tertentu saja. WHERE bisa digunakan untuk SELECT ALL maupun SELECT FIELD.

Query : SELECT (ALL/FIELD) FROM nama_tabel
WHERE nama_field = 'value';

2. Menampilkan Data menggunakan operator

Seperti halnya pemrograman java, C++ dll, SQL juga memiliki operator dasar. Operator dalam SQL adalah simbol yang digunakan untuk menginstruksi program untuk melakukan sesuatu. Akan berbeda definisi ketika kita membicarakan operator dalam kehidupan sehari-hari. Banyak sekali operator yang bisa digunakan ketika kita ingin menuliskan query. Disini kita akan membahas satu persatu operator dasar yang bisa kita gunakan dalam menuliskan query. Beberapa operator yang sering digunakan dalam proses query yaitu:

1. AND, &&
2. ||, OR
3. AS
4. ASC
5. BETWEEN ... AND ...

6. NOT BETWEEN ... AND ...
7. DESC
8. DISTINCT
9. GROUP BY
10. IN / NOT IN
11. LIKE
12. NOT LIKE
13. ORDER BY

4.2. SQL Advance

1. Melakukan Query dalam Data Integration

a. Filtering data menggunakan Limit

Limit digunakan untuk membatasi jumlah baris yang ditampilkan dari query. Berikut format penggunaan limit :

```
SELECT select_list FROM table_name LIMIT  
[offset,] row_count;
```

Offset digunakan untuk menentukan urutan dimulai baris berapa output ditampilkan. Row count digunakan untuk menentukan jumlah baris yang akan ditampilkan. Offset secara default akan menampilkan baris pertama, Offset baris pertama adalah 0, bukan 1.

Contoh (1) Menampilkan customerNumber, customerName, creditLimit dari tabel customers berdasarkan 5 creditLimit terbesar

```
SELECT  
    customerNumber,  
    customerName,  
    creditLimit
```

```

FROM
    customers
ORDER BY creditLimit DESC
LIMIT 5;

```

customerNumber	customerName	creditLimit
141	Euro+ Shopping Channel	227600.00
124	Mini Gifts Distributors Ltd.	210500.00
298	Vida Sport, Ltd	141300.00
151	Muscle Machine Inc	138500.00
187	AV Stores, Co.	136800.00

Gambar 5 Hasil query contoh (1) menggunakan limit

Contoh (2) Menampilkan orderNumber dan quantityOrdered dari tabel orderdetails dari urutan 5-10 berdasarkan quantityOrdered terendah

```

SELECT
    orderNumber,
    quantityOrdered
FROM orderdetails
ORDER By quantityOrdered ASC
LIMIT 5,5;

```

orderNumber	quantityOrdered
10401	11
10425	11
10419	12
10407	13
10419	15

Gambar 6 Hasil query contoh (2) menggunakan limit

b. Filtering data menggunakan is null

Is null merupakan sebuah operator untuk menguji apakah suatu nilai NULL atau tidak. Is null akan mengembalikan nilai true dan false. Karena IS NULL adalah operator perbandingan, Anda dapat menggunakannya di mana saja yang dapat digunakan oleh operator misalnya, dalam klausa SELECT atau WHERE. IS NULL dapat dinegasi menggunakan NOT menjadi IS NOT NULL.

Contoh (3) Menampilkan customerName, country, salesrepemployeenumber dari tabel customers dengan kondisi tidak memiliki salesrepemployeenumber, diurutkan berdasarkan customerName.

```
SELECT
    customerName,
    country,
    salesrepemployeenumber
FROM
    customers
WHERE
    salesrepemployeenumber IS NULL
```

ORDER BY

customerName;

customerName	country	salesrepemployeenumber
ANG Resellers	Spain	NULL
Anton Designs, Ltd.	Spain	NULL
Asian Shopping Network, Co	Singapore	NULL
Asian Treasures, Inc.	Ireland	NULL
BG&E Collectables	Switzerland	NULL
Cramer Spezialitäten, Ltd	Germany	NULL
Der Hund Imports	Germany	NULL
Feuer Online Stores, Inc	Germany	NULL
Franken Gifts, Co	Germany	NULL
Havel & Zbyszek Co	Poland	NULL
Kommission Auto	Germany	NULL
Kremlin Collectables, Co.	Russia	NULL
Lisboa Souvenirs, Inc	Portugal	NULL
Messner Shopping Network	Germany	NULL
Mit Vergnügen & Co.	Germany	NULL
Natürlich Autos	Germany	NULL
Porto Imports Co.	Portugal	NULL
Raanan Stores, Inc	Israel	NULL
SAR Distributors, Co	South Africa	NULL
Schuyler Imports	Netherlands	NULL
Stuttgart Collectable Exchange	Germany	NULL
Warburg Exchange	Germany	NULL

Gambar 7 Hasil query contoh (3) menggunakan is Null

Contoh (4) Menampilkan customerNumber, addressLine2 dari tabel customers yang tidak memiliki addressLine2 dan State-nya adalah MA.

SELECT

customerNumber,

addressLine2

FROM

customers

WHERE

addressLine2 IS NULL AND state LIKE

"%MA%";

customerNumber	addressLine2
173	NULL
198	NULL
204	NULL
286	NULL
320	NULL
362	NULL
379	NULL
462	NULL
495	NULL

Gambar 8 Hasil query contoh (4) menggunakan is Null

c. Menggabungkan data menggunakan JOIN

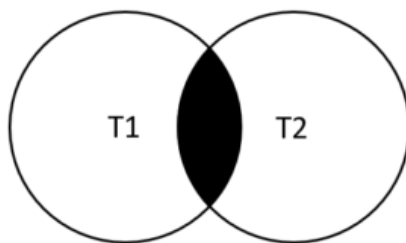
Join adalah metode menghubungkan data antara satu (self-join) atau lebih tabel berdasarkan nilai-nilai kolom di antara tabel. Untuk Join dengan tabel, Anda dapat menggunakan cross join, inner join, left join, atau right join berdasarkan jenis gabungan yang sesuai dengan kebutuhan. Klausa Join digunakan dalam pernyataan SELECT, muncul setelah klausa FROM.

Klausa **Inner Join** menggabungkan dua tabel berdasarkan kondisi yang dikenal sebagai predikat gabungan. Klausa join dalam membandingkan setiap baris dari tabel pertama dengan setiap baris dari tabel kedua. Jika nilai di kedua baris menyebabkan kondisi

gabungan bernilai true, inner join hanya menampilkan baris yang nilainya cocok. Berikut format penggunaan inner join

```
SELECT
    select_list
FROM t1
INNER JOIN t2 ON join_condition1
INNER JOIN t3 ON join_condition2
...;
```

Pertama, tentukan tabel utama yang muncul dalam klausa FROM (t1). Kedua, tentukan tabel yang akan bergabung dengan tabel utama, yang muncul di klausa INNER JOIN (t2, t3, ...). Ketiga, tentukan kondisi Join setelah kata kunci ON pada klausa INNER JOIN.



Gambar 9 Kondisi tabel saat inner join pada yang diwarnai hitam

Kondisi Join menentukan aturan untuk baris yang cocok antara tabel utama dan tabel muncul di klausa INNER JOIN.

Contoh (5) Menampilkan productCode, productName, textDescription dari tabel products yang productline nya berada di tabel productlines

```
SELECT
    productCode,
    productName,
    textDescription
FROM
    products t1
    INNER JOIN productlines t2
        ON t1.productline = t2.productline;
```

Contoh (6) Menampilkan firstname dari tabel employees yang melayani pelanggan dan bekerja di kota Paris.

```
SELECT firstname
FROM offices
    INNER JOIN
        employees using (officeCode)
    INNER JOIN
```

```
customers on  
employees.employeeNumber=customers.sale  
sRepEmployeeNumber  
where offices.city like "%Paris%";
```

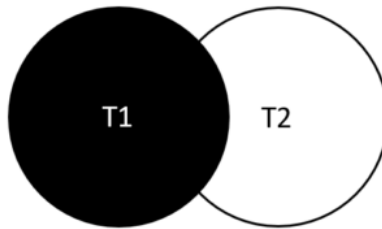
firstname
Loui
Gerard
Pamela
Martin

Gambar 10 Hasil query contoh (6) menggunakan inner join

Left join, ketika Anda menggunakan klausa Left Join, konsep tabel kiri dan tabel kanan diperkenalkan. Dalam sintaks tabel pertama setelah from yaitu t1 dianggap sebagai tabel kiri dan t2 adalah tabel kanan. Klausa LEFT JOIN memilih data mulai dari tabel kiri (t1), mencocokkan setiap baris dari tabel kiri (t1) dengan setiap baris dari tabel kanan (t2) berdasarkan pada *join_condition*.

Jika baris dari kedua tabel menyebabkan kondisi gabungan bernilai TRUE, LEFT JOIN menggabungkan kolom baris dari kedua tabel beserta isinya. Jika baris dari tabel kiri (t1) tidak cocok dengan baris mana pun dari tabel kanan (t2), LEFT JOIN masih menggabungkan

kolom baris dari kedua tabel. Namun isi tabel kanan akan bertuliskan NULL. Dengan kata lain, LEFT JOIN mengembalikan semua baris dari tabel kiri terlepas dari apakah suatu baris dari tabel kiri memiliki baris yang cocok dari tabel kanan atau tidak. Jika tidak ada kecocokan, kolom baris dari tabel kanan akan berisi NULL.



Gambar 11 Kondisi tabel saat left join pada yang diwarnai hitam

Contoh (7) Menampilkan customerNumber, customerName dari tabel customers dan status, orderNumber dari tabel orders dengan kondisi customerNumber berada di tabel orders ataupun tidak.

```
SELECT
    customers.customerNumber,
    customerName,
    orderNumber,
    status
```

FROM

customers

LEFT JOIN orders ON

orders.customerNumber =

customers.customerNumber;

Contoh (8) Menampilkan productName, productCode dari tabel Products dan quantityOrdered dari tabel orderdetails dengan kondisi productCode berada di tabel orderdetails ataupun tidak dan juga productName nya mengandung kata sup.

SELECT p.productName ,p.productCode,

o.quantityOrdered FROM Products P

LEFT JOIN orderdetails o ON

p.productCode=o.productCode

WHERE p.productName like "%sup%";

Right join mulai memilih data dari tabel kanan (t2).

Ini cocok dengan setiap baris dari tabel kanan dengan setiap baris dari tabel kiri. Jika kedua baris menyebabkan kondisi gabungan atau bernilai TRUE, maka akan menggabungkan kolom baris dari kedua tabel beserta isinya. Jika sebuah baris dari tabel kanan tidak memiliki baris yang cocok dari tabel kiri, maka isi tabel kiri akan

tampil NULL. Dengan kata lain, RIGHT JOIN mengembalikan semua baris dari tabel kanan terlepas dari apakah memiliki baris yang cocok dari tabel kiri atau tidak.

Penting untuk menekankan bahwa klausa RIGHT JOIN dan LEFT JOIN secara fungsional setara dan mereka dapat saling menggantikan selama urutan tabel dibalik. Contoh (9) Menampilkan employeeNumber, customerNumber yang employeeNumber ditabel employees tidak ada di tabel customers.

```
SELECT
    employeeNumber,
    customerNumber
FROM
    customers
RIGHT JOIN employees ON
    salesRepEmployeeNumber =
    employeeNumber
WHERE customerNumber is NULL
ORDER BY employeeNumber;
```

Contoh (10) Menampilkan productName, productCode dari tabel Products dan quantityOrdered dari

tabel orderdetails dengan kondisi productCode berada di tabel orderdetails ataupun tidak dan juga productName nya mengandung kata sup.

```
SELECT p.productName ,p.productCode,  
o.quantityOrdered  
FROM orderdetails o  
RIGHT JOIN Products P ON  
p.productCode=o.productCode  
WHERE  
p.productName LIKE "%sup%";
```

Self join sering digunakan untuk kueri data hierarkis atau untuk membandingkan baris dengan baris lain dalam tabel yang sama. Untuk melakukan self join, Anda harus menggunakan alias tabel untuk tidak mengulangi nama tabel yang sama dua kali dalam satu permintaan. Harus diperhatikan bahwa merujuk tabel dua kali atau lebih dalam kueri tanpa menggunakan alias tabel akan menyebabkan kesalahan.

Contoh (11) Menampilkan lastName dan first name dari employes berdasarkan reportsTo Di tabel employees.

```
SELECT
```

```

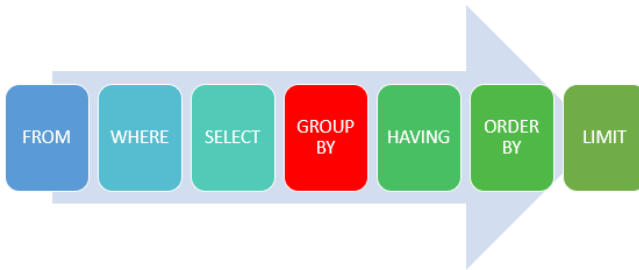
        CONCAT(m.lastName, ' ', m.firstName)
    AS Manager,
        CONCAT(e.lastName, ' ', e.firstName) AS
'Direct report'
FROM
    employees e
INNER JOIN employees m ON
    m.employeeNumber = e.reportsTo
ORDER BY
    Manager;

```

d. Menggabungkan data menggunakan GROUPING DATA

Klausa **GROUP BY** mengelompokkan seperangkat baris menjadi seperangkat baris ringkasan dengan nilai kolom atau ekspresi. Klausa GROUP BY mengembalikan satu baris untuk setiap grup. Dengan kata lain, ini mengurangi jumlah baris yang ditampilkan. Klausa GROUP BY sering digunakan bersama dengan fungsi agregat seperti SUM, AVG, MAX, MIN, dan COUNT. Fungsi agregat yang muncul dalam klausa SELECT memberikan informasi tentang masing-masing kelompok.

Klausu GROUP BY harus muncul setelah klausa FROM dan WHERE. MySQL mengeksekusi klausa GROUP BY setelah klausa FROM, WHERE dan SELECT, dan sebelum klausa HAVING, ORDER BY, dan LIMIT:



Gambar 12 Urutan query dalam klausa Group By

Contoh (12) Menampilkan status dan jumlah baris dari setiap status di tabel orders.

```
SELECT
    status, COUNT(*)
FROM
    orders
GROUP BY status;
```

status	COUNT(*)
Cancelled	6
Disputed	3
In Process	6
On Hold	4
Resolved	4
Shipped	303

Gambar 13 Hasil query contoh (12) menggunakan GROUP BY

Contoh (13) Menampilkan customerNumber dan hasil penjumlahan quantity ordered untuk setiap customer number.

```
SELECT
customerNumber,sum(quantityOrdered)
FROM orderdetails
INNER JOIN orders Using(orderNumber) group
by customerNumber;
```

Klausula **HAVING** digunakan dalam pernyataan SELECT untuk menentukan kondisi filter untuk sekelompok baris atau agregat. Klausula HAVING sering digunakan dengan klausa GROUP BY untuk memfilter grup berdasarkan kondisi tertentu. Jika klausa GROUP BY dihilangkan, klausa HAVING berlaku seperti klausa WHERE. Sintaks klausa HAVING:

```
SELECT
```

```

        select_list
FROM
        table_name
WHERE
        search_condition
GROUP BY
        group_by_expression
HAVING
        group_condition;

```

Dalam sintaks ini, Anda menentukan kondisi di klausa HAVING. Jika sebuah baris, yang dihasilkan oleh GROUP BY, menyebabkan group_condition bernilai true, maka hasilnya akan ditampilkan.

Perhatikan bahwa klausa HAVING menerapkan kondisi filter untuk setiap grup baris, sedangkan klausa WHERE menerapkan kondisi filter untuk setiap baris individu. Klausa HAVING dieksekusi setelah klausa FROM, WHERE, SELECT, dan GROUP BY dan sebelum klausa ORDER BY, dan LIMIT.

Contoh (14) Menampilkan
 ordernumber, SUM(quantityOrdered),
 SUM(priceeach*quantityOrdered) dari tabel orderdetails

yang dikelompokkan berdasarkan ordernumber dengan ketentuan hasil SUM(priceeach*quantityOrdered) setiap kelompok lebih dari 1000.

```
SELECT
    ordernumber,
    SUM(quantityOrdered) AS itemCount,
    SUM(priceeach*quantityOrdered) AS total
FROM
    orderdetails
GROUP BY
    ordernumber
HAVING
    total > 1000;
```

Contoh (15) Menampilkan customerNumber dan hasil penjumlahan quantity ordered untuk setiap customer number dengan ketentuan hasil penjumlahan quantity ordered melebihi 1500.

```
SELECT
    customerNumber,sum(quantityOrdered) as
    jumlah
FROM orderdetails
```

INNER JOIN orders Using(orderNumber) group
by customerNumber
HAVING jumlah>1500;

customerNumber	jumlah
114	1926
119	1832
124	6366
131	1631
141	9327
148	1524
151	1775
187	1778
278	1650
282	1601
323	1691
450	1656
496	1647

Gambar 14 Hasil query contoh (15) menggunakan HAVING

Roll up, Menyiapkan tabel sampel

```
CREATE TABLE sales
SELECT
    productLine,
    YEAR(orderDate) orderYear,
    SUM(quantityOrdered * priceEach)
orderValue
FROM
    orderDetails
    INNER JOIN
```

```
orders USING (orderNumber)
INNER JOIN
products USING (productCode)
GROUP BY
productLine ,
YEAR(orderDate);
```

Klausa **ROLLUP** adalah tambahan dari klausa GROUP BY dengan sintaks berikut:

```
SELECT
select_list
FROM
table_name
GROUP BY
c1, c2, c3 WITH ROLLUP;
```

ROLLUP menghasilkan beberapa kumpulan pengelompokan berdasarkan kolom atau ekspresi yang ditentukan dalam klausa GROUP BY. Contoh (16) Menampilkan productLine, SUM(orderValue) dari tabel sales berdasarkan setiap productline serta menampilkan jumlah semua hasil SUM(orderValue) tersebut.

```
SELECT
```

```

        productLine,
        SUM(orderValue) totalOrderValue
FROM
    sales
GROUP BY
    productline WITH ROLLUP;

```

Contoh (17) menampilkan negara(country), dan jumlah kantor berdasarkan negara masing-masing serta tampilkan jumlah keseluruhan kantor yang dimiliki.

```

SELECT country,count(*)
FROM
    offices group by country WITH ROLLUP;

```

country	count(*)
Australia	1
France	1
Japan	1
UK	1
USA	3
NULL	7

Gambar 15 Hasil query contoh (17) menggunakan Roll Up

e. Subqueries

Subquery MySQL adalah kueri yang bersarang di dalam kueri lain seperti SELECT, INSERT, UPDATE

atau DELETE. Selain itu, subquery dapat bersarang di dalam subquery lain. Subquery MySQL disebut query dalam sedangkan query yang berisi subquery disebut query luar. Subquery dapat digunakan di mana saja ekspresi itu digunakan dan harus ditutup dalam tanda kurung.

Contoh (18) Menampilkan lastName, firstName dari tabel employees yang officeCodanya berada di country USA.

```
SELECT
    lastName, firstName
FROM
    employees
WHERE
    officeCode IN (SELECT
        officeCode
    FROM
        offices
    WHERE
        country = 'USA');
```

lastName	firstName
Murphy	Diane
Patterson	Mary
Firrelli	Jeff
Bow	Anthony
Jennings	Leslie
Thompson	Leslie
Firrelli	Julie
Patterson	Steve
Tseng	Foon Yue
Vanauf	George

Gambar 16 Hasil query contoh (18) menggunakan Subqueries

Subquery mengembalikan semua kode kantor dari kantor yang berlokasi di USA. Query luar memilih nama belakang dan nama depan karyawan yang bekerja di kantor yang kode kantornya dalam hasil yang ditetapkan dikembalikan oleh subquery.

Contoh (19) Menampilkan customerName dari tabel customers yang melakukan pembayaran (paymentDate) pada tahun 2005.

```
SELECT customerName
FROM customers
WHERE customerNumber IN(SELECT
    customernumber
FROM payments
WHERE
    year(paymentDate) = 2005)
```

f. Derived Table

Derived Table adalah tabel virtual yang dikembalikan dari pernyataan SELECT. Derived Table mirip dengan temporary table, tetapi menggunakan Derived Table dalam pernyataan SELECT jauh lebih sederhana daripada temporary table karena tidak memerlukan langkah-langkah membuat temporary table.

Istilah tabel turunan dan subquery sering digunakan secara bergantian. Ketika subquery yang berdiri sendiri digunakan dalam klausa FROM dari pernyataan SELECT, biasa disebut sebagai Derived Table. Berikut ini menggambarkan query yang menggunakan Derived Table:

```
SELECT column_list
FROM (
    SELECT column_list
    FROM table_1
) derived_table_name
WHERE derived_table_name.c1 > 0;
```

The diagram highlights two key components of the query: the subquery itself, which is enclosed in a dashed orange box and labeled 'Derived table', and the alias 'derived_table_name' assigned to the subquery, which is pointed to by a green box labeled 'Must have an alias'.

Gambar 17 Contoh query menggunakan Derived table

Perhatikan bahwa subquery yang berdiri sendiri adalah subquery yang dapat mengeksekusi secara independen dari pernyataan yang memuatnya. Tidak

seperti subquery, tabel turunan harus memiliki alias sehingga Anda bisa merujuk namanya nanti dalam kueri. Jika tabel turunan tidak memiliki alias, MySQL akan mengeluarkan pesan kesalahan

Contoh (20) Menampilkan 5 produk dan pendapatan teratas berdasarkan pendapatan penjualan dari tabel orders dan orderdetails yang disimpan dalam basis data sepanjang tahun 2003. Petunjuk : pendapatan = (kuantitas x harga) setiap kode barang (menggunakan group by) untuk pembulatan angka gunakan fungsi round ()mencari irisan produk dari 2 tabel dapat menggunakan INNER JOIN (jika Primary Key nya sama, dapat menggunakan USING) Mencari beberapa record dari Produk teratas dengan menggunakan LIMIT dan harus diurutkan terlebih dahulu. Setiap derived table harus memiliki nama aliasnya sendiri.

```
SELECT
    productCode,
    ROUND(SUM(quantityOrdered *
    priceEach)) AS sales
FROM
    orderdetails
```

```

INNER JOIN
orders USING (orderNumber)
WHERE
YEAR(shippedDate) = 2003
GROUP BY productCode
ORDER BY sales DESC
LIMIT 5;

```

Kemudian query diatas dijadikan derived table. Menampilkan 5 nama produk dan pendapatan teratas berdasarkan pendapatan penjualan dari tabel orders dan orderdetails yang disimpan dalam basis data sepanjang tahun 2003. (perbedaan dengan nomor 1 adalah pada nomor 2 juga ditampilkan nama produknya) petunjuk : script nomor 1 di inner join kan dengan products (ini termasuk SUBQUERY) Using bisa digunakan jika primary key yang direlasikan sama. Setiap derived table harus memiliki nama aliasnya sendiri

```

SELECT
    productName, sales
FROM
    (SELECT
        productCode,

```

```

ROUND(SUM(quantityOrdered *
priceEach)) AS sales
FROM
    orderdetails
INNER JOIN orders USING
(orderNumber)
WHERE
    YEAR(shippedDate) = 2003
GROUP BY productCode
ORDER BY sales DESC
LIMIT 5) top5products2003
INNER JOIN
    products USING (productCode);

```

Contoh (21) tampilkan firstname employees dan jumlah pelayanan dengan syarat employees melakukan pelayanan lebih dari 1 kali kepada customer dengan nama mengandung kata ltd.

```

SELECT firstname, c.pelayanan FROM
    (select salesRepEmployeeNumber , count(*)
    as pelayanan
FROM customers
WHERE customerName like "%ltd%"

```

```
group by salesRepEmployeeNumber) as c
INNER JOIN employees ON employeeNumber =
salesRepEmployeeNumber
WHERE c.pelayanan > 1
```

g. Exists

Operator **EXISTS** adalah operator Boolean yang mengembalikan benar atau salah. Operator EXISTS sering digunakan untuk menguji keberadaan baris yang dikembalikan oleh subquery. Jika subquery mengembalikan setidaknya satu baris, operator EXISTS mengembalikan true, jika tidak, itu mengembalikan false. Selain itu, operator EXISTS menghentikan pemrosesan lebih lanjut setelah menemukan baris yang cocok, yang dapat membantu meningkatkan kinerja kueri.

Operator NOT meniadakan operator EXISTS. Dengan kata lain, NOT EXISTS mengembalikan true jika subquery tidak mengembalikan baris, jika tidak maka mengembalikan false. Perhatikan bahwa Anda dapat menggunakan SELECT *, kolom SELECT, SELECT a_constant, atau apa pun dalam subquery. Hasilnya sama

karena MySQL mengabaikan daftar pilih yang muncul di klausa SELECT.

Contoh (22) Tampilkan customer yang setidaknya melakukan sekali proses order.

```
SELECT
    customerNumber,
    customerName
FROM
    customers
WHERE
    EXISTS(
        SELECT
            1
        FROM
            orders
        WHERE
            orders.customernumber
            = customers.customernumber);
```

Contoh (23) Tampilkan nama customer yang melakukan setidaknya 1 kali order dan tidak memberikan komentar.

```
SELECT
```

```

distinct customerName
FROM
    customers
WHERE
    EXISTS(
        SELECT
            2
        FROM
            orders
        WHERE
            orders.customernumber
            = customers.customernumber and
            orders.comments is null);

```

h. Pengembangan SQL Lainnya

Beberapa hal yang dapat dipelajari selanjutnya yaitu

:

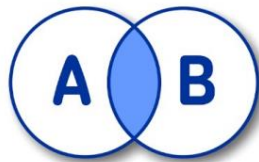
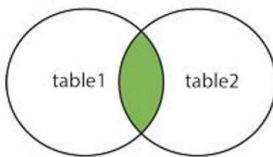
1. Set operator
 - Union / union all
 - Intersect
 - minus
2. Modifying data in MySQL

- Insert multiple rows
 - Insert into select
 - Insert ignore
 - Update join
 - On delete cascade
 - Delete join
3. Managing MySQL databases and tables
- Mysql storage engines
 - Mysql sequence
 - Removing column from table
 - Temporary table
 - Truncate table
4. MySQL data types
- Timestamp
 - JSON
 - ENUM

2. Konsep SQL JOIN

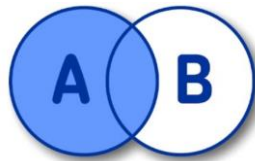
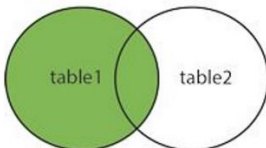
Join adalah cara untuk menghubungkan data yang diambil dari tabel-tabel melalui sebuah kolom yang menghubungkan mereka. Misal, pembaca mungkin ingin menghubungkan tabel alamat dengan tabel nomor telepon

berdasarkan nama seseorang (contoh: "Berikan saya alamat dan nomor telepon seseorang yang bernama John Smith."). Join memperbolehkan kita untuk mengambil data dari beberapa tabel melalui satu query. Hanya menggunakan sebuah tabel artinya kita hanya dapat menyimpan/memperoleh data yang terbatas atau justru menyimpan/memperoleh data yang terlalu banyak sehingga tabelnya menjadi kurang baik. Join menghubungkan satu tabel dengan tabel yang lain (inilah yang dimaksud dengan relational dari istilah relational database).



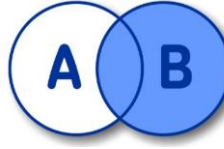
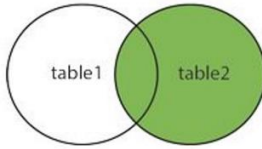
`SELECT * FROM A INNER JOIN B ON A.KEY = B.KEY`

Gambar 18 Contoh SQL INNER JOIN



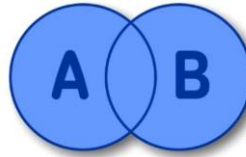
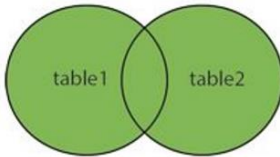
`SELECT * FROM A LEFT JOIN B ON A.KEY = B.KEY`

Gambar 19 Contoh SQL LEFT JOIN



SELECT * FROM A RIGHT JOIN B ON A.KEY = B.KEY

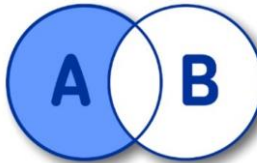
Gambar 20 Contoh SQL RIGHT JOIN



SELECT * FROM A FULL OUTER JOIN B ON

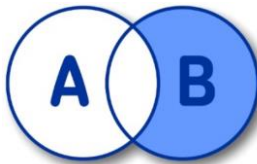
A.KEY = B.KEY

Gambar 21 Contoh SQL FULL OUTER JOIN



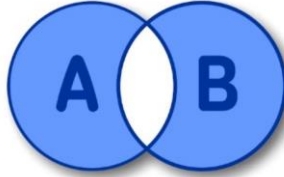
SELECT * FROM A LEFT JOIN B ON A.KEY = B.KEY WHERE B.KEY IS NULL

Gambar 22 Contoh SQL LEFT JOIN WHEN B.KEY IS NULL



SELECT * FROM A RIGHT JOIN B ON A.KEY = B.KEY WHERE A.KEY IS NULL

Gambar 23 Contoh SQL RIGHT JOIN WHEN A.KEY IS NULL



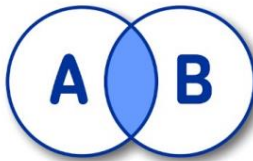
`SELECT * FROM A FULL OUTER JOIN B ON A.KEY = B.KEY WHERE A.KEY IS NULL OR B.KEY IS NULL`

Gambar 24 Contoh SQL FULL OUTER JOIN WHEN B.KEY IS NULL

a. INNER JOIN

Inner join mungkin tipe join yang paling banyak dipakai. Inner join mengembalikan baris-baris dari dua tabel atau lebih yang memenuhi syarat.

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

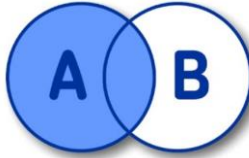


Gambar 25 Contoh SQL INNER JOIN

b. LEFT [OUTER] JOIN

Left outer join (sering disingkat left join) akan mengembalikan seluruh baris dari tabel disebelah kiri yang dikenai kondisi ON dan hanya baris dari tabel disebelah kanan yang memenuhi kondisi join.

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

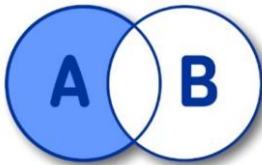


Gambar 26 Contoh SQL LEFT JOIN

c. LEFT [OUTER] JOIN without Intersection

Join ini merupakan variasi dari left outer join. Pada join ini kita hanya akan mengambil data dari tabel sebelah kiri yang dikenai kondisi ON yang juga memenuhi kondisi join tanpa data dari tabel sebelah kanan yang memenuhi kondisi join.

SELECT columns FROM TableA LEFT OUTER
JOIN TableB ON A.columnName = B.columnName
WHERE B.columnName IS NULL

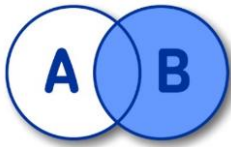


Gambar 27 Contoh SQL LEFT JOIN WHEN B.KEY IS NULL

d. RIGHT [OUTER] JOIN

Right outer join (sering disingkat right join) akan mengembalikan semua baris dari tabel sebelah kanan yang dikenai kondisi ON dengan data dari tabel sebelah kiri yang memenuhi kondisi join. Teknik ini merupakan kebalikan dari left outer join.

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

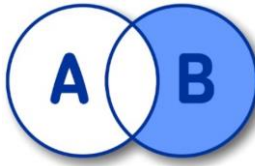


Gambar 28 Contoh SQL RIGHT JOIN

e. RIGHT [OUTER] JOIN without Intersection

Teknik ini merupakan variasi dari right outer join. Pada join ini kita hanya akan mengambil data dari tabel sebelah kanan yang dikenai kondisi ON yang juga memenuhi kondisi join tanpa data dari tabel sebelah kanan yang memenuhi kondisi join.

```
SELECT columns FROM TableA RIGHT OUTER  
JOIN TableB ON A.columnName = B.columnName  
WHERE A.columnName IS NULL
```

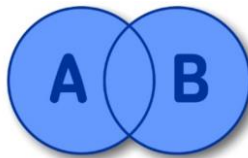


Gambar 29 Contoh SQL RIGHT JOIN WHEN A.KEY IS NULL

f. FULL [OUTER] JOIN

Full [Outer] Join Full outer join (sering disingkat full join) akan mengembalikan seluruh baris dari kedua tabel yang dikenai ON termasuk data-data yang bernilai NULL.

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

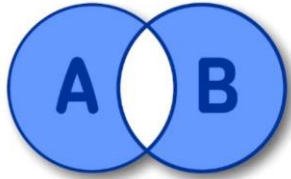


Gambar 30 Contoh SQL FULL OUTER JOIN

g. FULL [OUTER] JOIN without Intersection

Full [Outer] Join without Intersection Variasi lain dari full outer join yang akan mengembalikan seluruh data dari kedua tabel yang dikenai ON tanpa data yang memiliki nilai NULL.

SELECT columns FROM TableA FULL JOIN
TableB ON A.columnName = B.columnName WHERE
A.columnName IS NULL OR B.columnName IS NULL



**Gambar 31 Contoh SQL FULL OUTER JOIN WHEN A.KEY IS
NULL B.KEY IS NULL**

3. SUBQUERY

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. Subqueries dapat digunakan dengan SELECT, INSERT, UPDATE, dan DELETE statements bersama dengan operator seperti =, >, <, <=, IN, BETWEEN dll.

Ada beberapa aturan yang subqueries harus mengikuti:

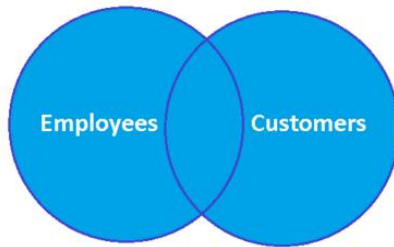
- Subqueries harus tertutup dalam tanda kurung.

- Sebuah subquery hanya dapat memiliki satu kolom pada klausa SELECT, kecuali beberapa kolom yang di query utama untuk subquery untuk membandingkan kolom yang dipilih.
- ORDER BY tidak dapat digunakan dalam subquery, meskipun permintaan utama dapat menggunakan ORDER BY. GROUP BY dapat digunakan untuk melakukan fungsi yang sama seperti ORDER BY dalam subquery.
- Subqueries yang kembali lebih dari satu baris hanya dapat digunakan dengan beberapa value operator, seperti operator IN.
- Operator BETWEEN tidak dapat digunakan dengan subquery; Namun, BETWEEN dapat digunakan dalam subquery

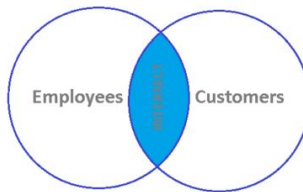
4. Konsep SQL UNION

Klausa UNION, INTERSECT, dan EXCEPT digunakan untuk menggabungkan atau mengecualikan baris serupa dari dua tabel atau lebih. Union berguna saat Anda perlu menggabungkan hasil dari query terpisah menjadi satu hasil tunggal. Union berbeda dari Join di

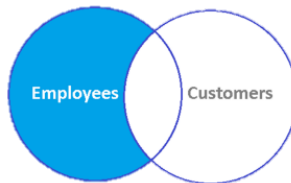
seluruh baris yang cocok dan, sebagai hasilnya, disertakan atau dikecualikan dari hasil gabungan.



Gambar 32 Contoh SQL UNION



Gambar 33 Contoh SQL INTERSECT



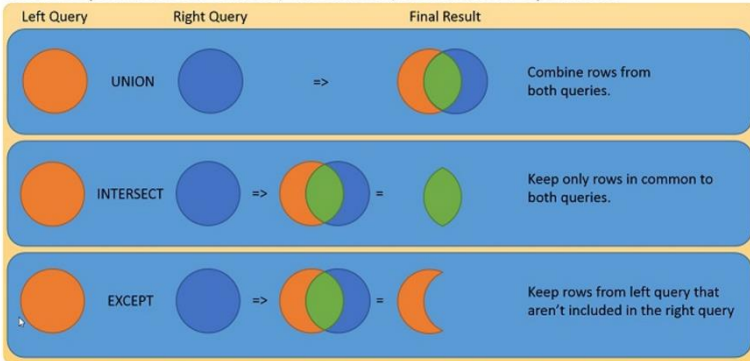
Gambar 34 Contoh SQL EXCEPT

Operator ini dapat digunakan pada query apapun; namun, beberapa kondisi sederhana harus dipenuhi:

1. Kolom nomor dan urutan harus sama di kedua query

2. Tipe data harus sama atau kompatibel.

Visual Explanation of UNION, INTERSECT, and EXCEPT operators



Gambar 35 Penjelasan Visualisasi UNION, INTERSECT & EXCEPT

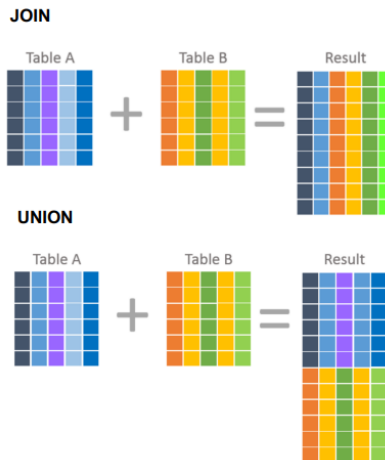
a. UNION OPERATOR

Operator Union mengembalikan baris dari kedua tabel. Jika digunakan sendiri, UNION mengembalikan daftar baris yang berbeda. Menggunakan UNION ALL mengembalikan semua baris dari kedua tabel. UNION berguna saat Anda ingin mengurutkan hasil dari dua query terpisah sebagai satu hasil gabungan. Misalnya, jika Anda memiliki dua tabel, Vendor, dan Pelanggan, dan Anda menginginkan daftar nama gabungan, Anda dapat melakukannya dengan mudah menggunakan:

```
SELECT `Vendor`, V.Name FROM Vendor V
UNION
```

```
SELECT `Customer`, C.Name FROM Customer C  
ORDER BY Name
```

Perhatikan klausa ORDER BY berlaku untuk hasil gabungan.



Gambar 36 Visualisasi Perbedaan Join dan Union

b. INTERSECT OPERATOR

Gunakan operator Intersect untuk mengembalikan baris yang sama di antara dua tabel ; itu mengembalikan baris unik dari query kiri dan kanan . query ini berguna saat Anda ingin menemukan hasil yang sama di antara dua query . Melanjutkan Vendor, dan Pelanggan, misalkan Anda ingin mencari vendor yang juga merupakan

pelanggan . Anda dapat melakukannya dengan mudah menggunakan :

```
SELECT V.Name FROM Vendor V INTERSECT  
SELECT C.Name FROM Customer C ORDER BY  
Name
```

c. EXCEPT OPERATOR

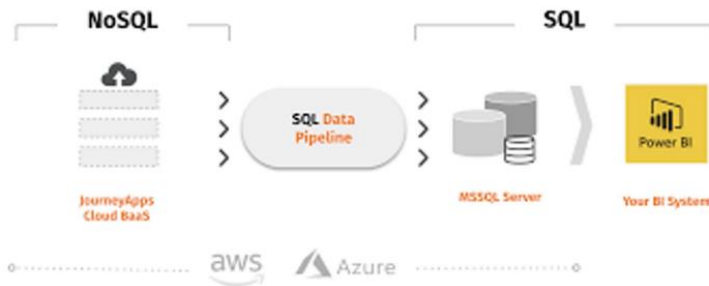
Gunakan Operator KECUALI untuk mengembalikan hanya baris yang ditemukan di query kiri . Ini mengembalikan baris unik dari query kiri yang tidak ada di hasil Query yang benar . query ini berguna saat Anda ingin menemukan baris yang ada di satu kumpulan tetapi tidak di kumpulan lainnya . Misalnya, untuk membuat daftar semua vendor yang bukan pelanggan Anda bisa menulis :

```
SELECT V.Name FROM Vendor V EXCEPT  
SELECT C.Name FROM Customer C ORDER BY  
Name
```

d. PIPELINE

Pipeline SQL adalah proses yang menggabungkan beberapa resep berurutan (masing-masing menggunakan mesin SQL yang sama) dalam alur kerja DSS. Resep

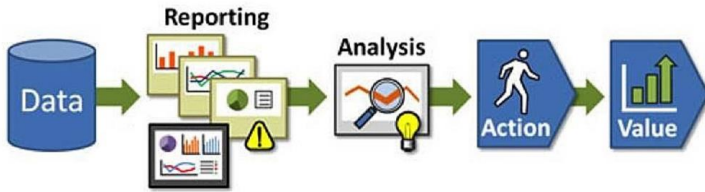
gabungan ini, yang bisa berupa resep visual dan "kueri SQL", kemudian dapat dijalankan sebagai aktivitas pekerjaan tunggal. Penggunaan pipeline SQL sangat meningkatkan performa dengan menghindari penulisan dan pembacaan yang tidak perlu dari intermediate dataset



Gambar 37 Visualisasi Perbedaan Join dan Union

e. ANALYSIS & REPORTING

Memahami perbedaan antara pelaporan dan analisis data akan membantu Anda menghindari kesalahan data serupa pada bisnis Anda. Pelaporan menggunakan data untuk melacak kinerja bisnis Anda, sedangkan analisis menggunakan data untuk menjawab pertanyaan strategis tentang bisnis Anda. Meskipun mereka berbeda, pelaporan dan analisis saling mengandalkan. Pelaporan menjelaskan pertanyaan apa yang harus diajukan, dan sebuah analisis mencoba menjawab pertanyaan tersebut.



Gambar 38 Visualisasi Tahapan Bisnis Kerja

Reporting

Pelaporan data adalah proses mengatur data ke dalam bagan dan tabel untuk melacak kinerja bisnis Anda. Data mentah ini membuat Anda tetap sadar akan apa yang terjadi dengan bisnis Anda . Ketika bisnis Anda tidak mencapai salah satu tujuannya, bagan pelaporan Anda akan mengingatkan Anda tentang masalah tersebut, yang mendorong Anda untuk menanggapi. Meskipun laporan adalah garis pertahanan pertama untuk kesehatan bisnis Anda, seringkali tidak mungkin untuk mengekstrak wawasan yang dapat membantu Anda memperbaiki masalah atau memanfaatkan peluang. Misalnya, laporan Kemenangan / Kerugian Penjualan ini memberi Anda indikasi kinerja penjualan individu, tetapi data tidak menjelaskan mengapa setiap perwakilan memiliki hasil yang berbeda

Analysis Data

Analisis data adalah proses memeriksa data dengan tujuan menjawab pertanyaan bisnis yang mendukung pengambilan keputusan. Analisis dapat mengungkapkan wawasan yang kuat jika Anda dapat mengungkapkan mengapa sesuatu terjadi dan apa yang dapat Anda lakukan. Berikut tiga langkah utama untuk membangun analisis yang membantu mengungkapkan wawasan:

1. Mulailah Dengan Pertanyaan Khusus.

Sebelum Anda menggali data Anda, tulis pertanyaan apa yang perlu dijawab untuk mencapai tujuan Anda. Semakin banyak pertanyaan yang diajukan, semakin berharga dan dapat ditindaklanjuti jawabannya.

2. Identifikasi Sumber Data.

Saat Anda memulai dengan pertanyaan terperinci, Anda dapat menunjukkan dengan tepat data yang diperlukan untuk merumuskan jawaban dari pertanyaan itu.

3. Menafsirkan Hasil.

Analisis data masih mengharuskan Anda membuat kesimpulan tentang temuan Anda. Ketika Anda menemukan fakta atau pola yang menarik dan menempatkannya dalam konteks pertanyaan bisnis Anda,

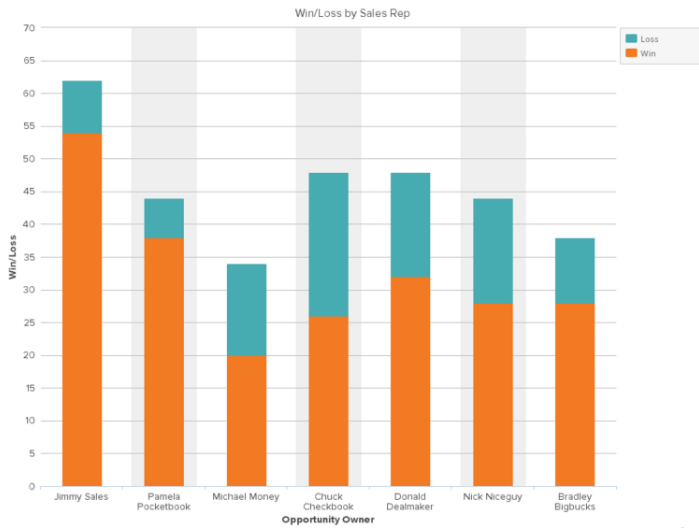
Anda akan ingin menguji kesimpulan Anda dengan bertanya pada diri sendiri.

Contoh : Laporan Rasio Menang & Rugi

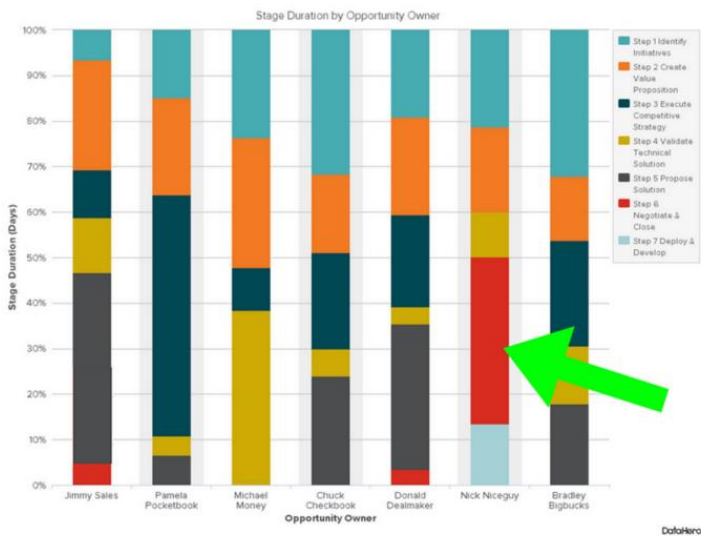
Atribut

- Sales Owner
- Menang / Kalah

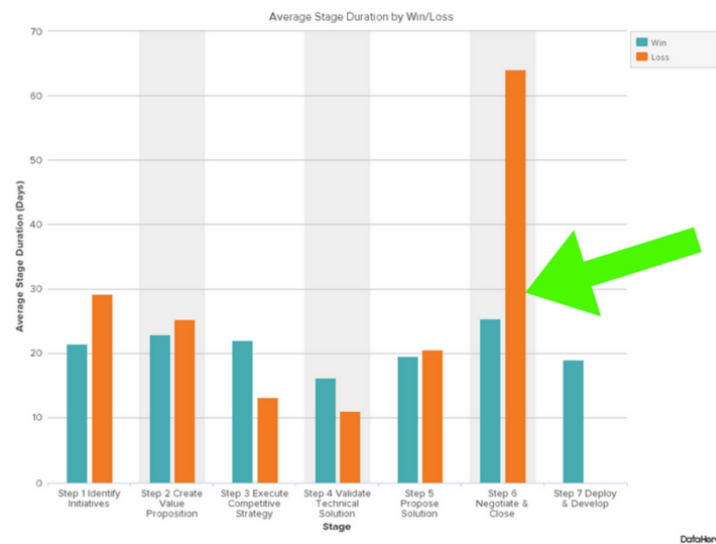
Laporan ini menyoroti berapa banyak kesepakatan yang telah dimenangkan atau dikalahkan untuk setiap perwakilan penjualan. Sebagai kepala penjualan, Anda memerlukan data kinerja ini untuk mengevaluasi tim Anda, tetapi Anda juga ingin mempelajari bagaimana Anda dapat membantu meningkatkan kinerja yang lebih rendah. Mungkin pertanyaan pertama Anda berdasarkan laporan ini adalah, "Bagaimana perwakilan berperforma tinggi mengalokasikan waktu mereka dalam proses dibandingkan dengan yang berkinerja lebih rendah?"



Gambar 39 Visualisasi Permasalahan Laporan Menang dan Rugi

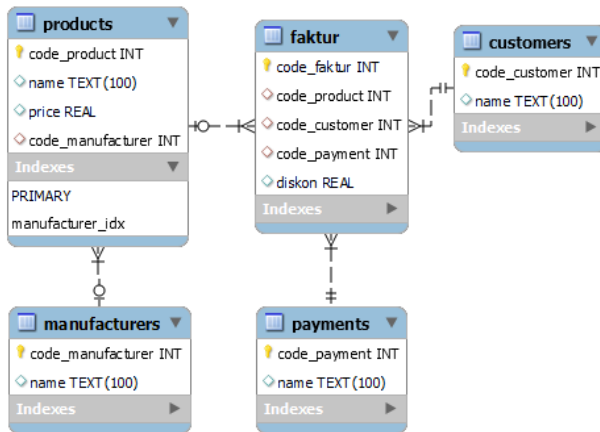


Gambar 40 Visualisasi Permasalahan Performa kinerja



Gambar 41 Visualisasi Insight yang diperoleh

5. SQL Challenge



Gambar 42 Contoh tabel yang digunakan

1. Pilih nama semua produk di toko.
2. Pilih nama dan harga semua produk di toko.
3. Pilih nama produk dengan harga kurang dari atau sama dengan \$ 200.
4. Pilih semua produk dengan harga antara \$ 60 dan \$ 120.
5. Pilih nama dan harga dalam sen (yaitu, harga harus dikalikan dengan 100).
6. Hitung harga rata-rata semua produk.
7. Hitung harga rata-rata semua produk dengan kode pabrik sama dengan 2.
8. Hitung jumlah produk dengan harga lebih besar dari atau sama dengan \$ 180.
9. Pilih nama dan harga semua produk dengan harga lebih besar dari atau sama dengan \$ 180, dan urutkan terlebih dahulu menurut harga (dalam

urutan menurun), lalu menurut nama (dalam urutan naik).

10. Pilih semua data dari produk, termasuk semua data untuk setiap produsen produk.
11. Pilih nama produk, harga, dan nama pabrikan dari semua produk.
12. Pilih harga rata-rata dari setiap produk produsen, dengan hanya menampilkan kode produsen.
13. Pilih harga rata-rata dari setiap produk pabrikan, dengan menunjukkan nama pabrikan.
14. Pilih nama pabrikan yang produknya memiliki harga lebih besar dari atau sama dengan \$ 150.
15. Pilih nama dan harga produk termurah.
16. Pilih nama masing-masing pabrikan beserta nama dan harga produk termahalnya.
17. Pilih nama masing-masing pabrikan yang memiliki harga rata-rata di atas \$ 145 dan berisi setidaknya 2 produk berbeda.
18. Tambahkan produk baru: keyboard, \$ 200, pabrikan 2.
19. Perbarui nama produk keyboard menjadi "Laser Printer".
20. Terapkan diskon 10% untuk semua produk yang ada di faktur
21. Terapkan diskon 10% untuk semua produk dengan harga lebih besar dari atau sama dengan \$ 120
22. Terapkan harga terendah dari transaksi yang dibayar secara cash
23. Terapkan pelanggan dengan pembelian > 1
24. Terapkan produk yang dibeli paling sering
25. Terapkan pelanggan yang tidak pernah membeli barang.

LAMPIRAN

1. Pilih nama semua produk di toko.

`select * from products;`

```
MariaDB [integrasi]> select*from products;
```

code_product	name_product	price	code_manufacturer
1	Loudspeaker	150	2
2	Printer	300	2
3	Papper A4	50	1
4	Monitor	350	2
5	Books	10	1
6	Mouse	75	2
7	Pencil	5	1
8	Flashdisk	60	2
9	Microwave	300	5
10	Blender	250	5
11	tshirt	75	3
12	bag	90	3
13	milk	15	4
14	shoe	10	3
15	sugar	10	4

15 rows in set (0.016 sec)

2. Pilih nama dan harga semua produk di toko.

`select name_product, price from products;`

```
MariaDB [integrasi]> select name_product, price from products;
```

name_product	price
Loudspeaker	150
Printer	300
Papper A4	50
Monitor	350
Books	10
Mouse	75
Pencil	5
Flashdisk	60
Microwave	300
Blender	250
tshirt	75
bag	90
milk	15
shoe	10
sugar	10

15 rows in set (0.001 sec)

3. Pilih nama produk dengan harga kurang dari atau sama dengan \$ 200.

select name_product from products where price <= 200;

```
MariaDB [integrasi]> select name_product from products where price <=200;

+-----+
| name_product |
+-----+
| Loudspeaker  |
| Papper A4    |
| Books        |
| Mouse        |
| Pencil       |
| Flashdisk    |
| tshirt       |
| bag          |
| milk         |
| shoe         |
| sugar        |
+-----+
11 rows in set (0.013 sec)
```

4. Pilih semua produk dengan harga antara \$ 60 dan \$ 120.

select * from products where price > 60 and price <120;

```
MariaDB [integrasi]> select*from products where price > 60 and price < 120;

+-----+-----+-----+-----+
| code_product | name_product | price | code_manufacturer |
+-----+-----+-----+-----+
| 6           | Mouse       | 75    | 2                 |
| 11          | tshirt      | 75    | 3                 |
| 12          | bag         | 90    | 3                 |
+-----+-----+-----+-----+
3 rows in set (0.005 sec)
```

5. Pilih nama dan harga dalam sen (yaitu, harga harus dikalikan dengan 100).

select name_product, price*100 as "price in sen" from products;

```
MariaDB [integrasi]> select name_product, price*100 as "price in sen" from products;
```

name_product	price in sen
Loudspeaker	15000
Printer	30000
Papper A4	5000
Monitor	35000
Books	1000
Mouse	7500
Pencil	500
Flashdisk	6000
Microwave	30000
Blender	25000
tshirt	7500
bag	9000
milk	1500
shoe	1000
sugar	1000

```
15 rows in set (0.007 sec)
```

6. Hitung harga rata-rata semua produk.

`select avg(price) from products;`

```
MariaDB [integrasi]> SELECT AVG(price) as "Mean" FROM products;
```

Mean
116.66666666666667

```
1 row in set (0.013 sec)
```

7. Hitung harga rata-rata semua produk dengan kode pabrikan sama dengan 2.

`select avg(price) as "mean" from products where code_manufacturer = 2;`

```
MariaDB [integrasi]> SELECT AVG(price) as "Mean" FROM products where code_manufacturer = 2;
```

Mean
187

```
1 row in set (0.001 sec)
```

8. Hitung jumlah produk dengan harga lebih besar dari atau sama dengan \$ 180.

`select count(name_product) as "jumlah produk" from products where price >= 180;`

```
MariaDB [integrasi]> select Count(name_product) as "jumlah produk"
from products where price >= 180;
+-----+
| jumlah produk |
+-----+
| 4 |
+-----+
1 row in set (0.001 sec)
```

9. Pilih nama dan harga semua produk dengan harga lebih besar dari atau sama dengan \$ 180, dan urutkan terlebih dahulu menurut harga (dalam urutan menurun), lalu menurut nama (dalam urutan naik).
 select name_product, price from products where price >= 180 order by price desc, name_product asc;

```
MariaDB [integrasi]> select name_product, price from products where
price >= 180 order by price desc, name_product asc;
+-----+-----+
| name_product | price |
+-----+-----+
| Monitor      | 350   |
| Microwave    | 300   |
| Printer      | 300   |
| Blender      | 250   |
+-----+-----+
4 rows in set (0.001 sec)
```

10. Pilih semua data dari produk, termasuk semua data untuk setiap produsen produk.
 select*from products join manufacturers on products.code_manufacturer = manufacturers.code_manufacturer;

```
MariaDB [integrasi]> select*from products join manufacturers on products.code_manufacturer = manufacturers.code_manufacturer;
+-----+-----+-----+-----+-----+
| code_product | name_product | price | code_manufacturer | code_manufacturer | name_manufacturer |
+-----+-----+-----+-----+-----+
| 1 | Loudspeaker | 150 | 2 | 2 | Elektronik Komputer |
| 2 | Peliter | 300 | 2 | 2 | Elektronik Komputer |
| 3 | Papper Ad | 50 | 1 | 1 | Alat Tulis |
| 4 | Monitor | 350 | 2 | 2 | Elektronik Komputer |
| 5 | Books | 10 | 1 | 1 | Alat Tulis |
| 6 | House | 75 | 2 | 2 | Elektronik Komputer |
| 7 | Pencil | 5 | 1 | 1 | Alat Tulis |
| 8 | Flashdisk | 60 | 2 | 2 | Elektronik Komputer |
| 9 | Microwave | 300 | 5 | 5 | Elektronik Dapur |
| 10 | Blender | 250 | 5 | 5 | Elektronik Dapur |
| 11 | tshirt | 75 | 3 | 3 | Pakaian |
| 12 | bag | 90 | 3 | 3 | Pakaian |
| 13 | milk | 15 | 4 | 4 | Pangan |
| 14 | shoe | 10 | 3 | 3 | Pakaian |
| 15 | sugar | 10 | 4 | 4 | Pangan |
+-----+-----+-----+-----+-----+
15 rows in set (0.001 sec)
```

11. Pilih nama produk, harga, dan nama pabrikan dari semua produk.

```
select products.name_product, products.price,
manufacturers.name_manufacturer from products
join manufacturers on products.code_manufacturer =
manufacturers.code_manufacturer;
```

```
MariaDB [integrasi]> select products.name_product, products.price,
manufacturers.name_manufacturer from products join manufacturers on
products.code_manufacturer = manufacturers.code_manufacturer;
```

name_product	price	name_manufacturer
Loudspeaker	150	Elektronik Komputer
Printer	300	Elektronik Komputer
Papper A4	50	Alat Tulis
Monitor	350	Elektronik Komputer
Books	10	Alat Tulis
Mouse	75	Elektronik Komputer
Pencil	5	Alat Tulis
Flashdisk	60	Elektronik Komputer
Microwave	300	Elektronik Dapur
Blender	250	Elektronik Dapur
tshirt	75	Pakaian
bag	90	Pakaian
milk	15	Pangan
shoe	10	Pakaian
sugar	10	Pangan

```
15 rows in set (0.004 sec)
```

12. Pilih harga rata-rata dari setiap produk produsen, dengan hanya menampilkan kode produsen.

```
select manufacturers.code_manufacturer from manufacturers
left join products on
manufacturers.code_manufacturer =
products.code_product group
by manufacturers.code_manufacturer order
by avg(products.price);
```

```
MariaDB [integrasi]> select manufacturers.code_manufacturer from manufacturers left
join products on manufacturers.code_manufacturer = products.code_product group by
manufacturers.code_manufacturer order by avg(products.price);
```

code_manufacturer
5
3
1
2
4

```
5 rows in set (0.001 sec)
```

13. Pilih harga rata-rata dari setiap produk pabrikan, dengan menunjukkan nama pabrikan.

```
select manufacturers.name_manufacturer from
manufacturers left join products on
manufacturers.code_manufacturer =
products.code_product group by
manufacturers.code_manufacturer order by
avg(products.price);
```

```
MariaDB [integrasi]> select manufacturers.name_manufacturer from manufacturers left
join products on manufacturers.code_manufacturer = products.code_product group by
manufacturers.code_manufacturer order by avg(products.price);
+-----+
| name_manufacturer |
+-----+
| Elektronik Dapur  |
| Pakaian           |
| Alat Tulis        |
| Elektronik Komputer |
| Pangan            |
+-----+
5 rows in set (0.034 sec)
```

14. Pilih nama pabrikan yang produknya memiliki harga lebih besar dari atau sama dengan \$ 150.

```
select manufacturers.name_manufacturer from
manufacturers left join products on
manufacturers.code_manufacturer =
products.code_product where products.price >= 150;
```

```
MariaDB [integrasi]> select manufacturers.name_manufacturer from manufacturers
left join products on manufacturers.code_manufacturer = products.code_product
where products.price >= 150;
+-----+
| name_manufacturer |
+-----+
| Alat Tulis        |
| Elektronik Komputer |
| Pangan            |
+-----+
3 rows in set (0.001 sec)
```

15. Pilih nama dan harga produk termurah.

```
select name_product, price from products order by
price asc;
```

```
MariaDB [integrasi]> select name_product, price from products order by price asc;
```

name_product	price
Pencil	5
Books	10
shoe	10
sugar	10
milk	15
Papper A4	50
Flashdisk	60
Mouse	75
tshirt	75
bag	90
Loudspeaker	150
Blender	250
Printer	300
Microwave	300
Monitor	350

```
15 rows in set (0.001 sec)
```

16. Pilih nama masing-masing pabrikan beserta nama dan harga produk termahalnya.

```
select manufacturers.name_manufacturer,
products.name_product, products.price from
products left join manufacturers on
products.code_manufacturer =
manufacturers.code_manufacturer order by
products.price desc;
```

```
MariaDB [integrasi]> select manufacturers.name_manufacturer,products.name_product,
products.price from products left join manufacturers on products.code_manufacturer
= manufacturers.code_manufacturer order by products.price desc;
```

name_manufacturer	name_product	price
Elektronik Komputer	Monitor	350
Elektronik Komputer	Printer	300
Elektronik Dapur	Microwave	300
Elektronik Dapur	Blender	250
Elektronik Komputer	Loudspeaker	150
Pakaian	bag	90
Elektronik Komputer	Mouse	75
Pakaian	tshirt	75
Elektronik Komputer	Flashdisk	60
Alat Tulis	Papper A4	50
Pangan	milk	15
Alat Tulis	Books	10
Pakaian	shoe	10
Pangan	sugar	10
Alat Tulis	Pencil	5

```
15 rows in set (0.002 sec)
```

17. Pilih nama masing-masing pabrikan yang memiliki harga rata-rata di atas \$ 145 dan berisi setidaknya 2 produk berbeda.

```
select manufacturers.name_manufacturer from
manufacturers left join products on
manufacturers.code_manufacturer =
products.code_manufacturer order by
avg(products.price)>145 and
count(products.code_manufacturer)>1;
```

```
MariaDB [integrasi]> select manufacturers.name_manufacturer from manufacturers left
join products on manufacturers.code_manufacturer = products.code_manufacturer order
by avg(products.price)>145 and count(products.code_manufacturer)>1;

+-----+
| name_manufacturer |
+-----+
| Elektronik Komputer |
+-----+
1 row in set (0.001 sec)
```

18. Tambahkan produk baru: keyboard, \$ 200, pabrikan 2.

```
insert into products values (16,'keyboard',200,2);
```

```
MariaDB [integrasi]> INSERT INTO products VALUES (16,'Keyboard',200,2);
Query OK, 1 row affected (0.051 sec)
```

```
MariaDB [integrasi]> select*from products;
```

```
+-----+-----+-----+-----+
| code_product | name_product | price | code_manufacturer |
+-----+-----+-----+-----+
| 1 | Loudspeaker | 150 | 2 |
| 2 | Printer | 300 | 2 |
| 3 | Papper A4 | 50 | 1 |
| 4 | Monitor | 350 | 2 |
| 5 | Books | 10 | 1 |
| 6 | Mouse | 75 | 2 |
| 7 | Pencil | 5 | 1 |
| 8 | Flashdisk | 60 | 2 |
| 9 | Microwave | 300 | 5 |
| 10 | Blender | 250 | 5 |
| 11 | tshirt | 75 | 3 |
| 12 | bag | 90 | 3 |
| 13 | milk | 15 | 4 |
| 14 | shoe | 10 | 3 |
| 15 | sugar | 10 | 4 |
| 16 | Keyboard | 200 | 2 |
+-----+-----+-----+-----+
16 rows in set (0.002 sec)
```

19. Perbarui nama produk keyboard menjadi "Laser Printer".

update products set name_product="Laser Printer"
where code_product=16;

```
MariaDB [integrasi]> update products set name_product="Laser Printer" where
code_product=16;
Query OK, 1 row affected (0.016 sec)
Rows matched: 1 Changed: 1 Warnings: 0

MariaDB [integrasi]> select*from products;
```

code_product	name_product	price	code_manufacturer
1	Loudspeaker	150	2
2	Printer	300	2
3	Papper A4	50	1
4	Monitor	350	2
5	Books	10	1
6	Mouse	75	2
7	Pencil	5	1
8	Flashdisk	60	2
9	Microwave	300	5
10	Blender	250	5
11	tshirt	75	3
12	bag	90	3
13	milk	15	4
14	shoe	10	3
15	sugar	10	4
16	Laser Printer	200	2

```
16 rows in set (0.001 sec)
```

20. Terapkan diskon 10% untuk semua produk yang ada di faktur

select products.name_product , products.price*0.9
from products right join faktur on
products.code_product = faktur.code_product;

```
MariaDB [integrasi]> select products.name_product , products.price*0.9 from products
right join faktur on products.code_product = faktur.code_product;
```

name_product	products.price*0.9
Monitor	315
Printer	270
Pencil	4.5
Printer	270
Blender	225
Microwave	270
Papper A4	45

```
7 rows in set (0.004 sec)
```

21. Terapkan diskon 10% untuk semua produk dengan harga lebih besar dari atau sama dengan \$ 120
select name_product, price*0.9 from products where
price>=120;

```
MariaDB [integrasi]> select name_product, price*0.9 from products where price>=120;
```

name_product	price*0.9
Loudspeaker	135
Printer	270
Monitor	315
Microwave	270
Blender	225

```
5 rows in set (0.001 sec)
```

22. Terapkan harga terendah dari transaksi yang dibayar secara cash

```
select products.price from products right join faktur
on products.code_product = faktur.code_product
where faktur.code_payment=2 order by
products.price asc;
```

```
MariaDB [integrasi]> select products.price from products right join faktur on
products.code_product = faktur.code_product where faktur.code_payment=2 order
by products.price asc;
```

price
250
300
300
350

```
4 rows in set (0.008 sec)
```

23. Terapkan pelanggan dengan pembelian lebih dari 1
- ```
select customers.name_customer from customers
right join faktur on customers.code_customer =
faktur.code_customer group by
customers.code_customer order by
faktur.code_customer >= 2;
```

```
MariaDB [integrasi]> select customers.name_customer from customers right join faktur
on customers.code_customer = faktur.code_customer group by customers.code_customer
order by faktur.code_customer >= 2;
```

| name_customer |
|---------------|
| Abi           |
| Bima          |
| Citra         |
| Ena           |

```
4 rows in set (0.007 sec)
```

24. Terapkan produk yang dibeli paling sering
- ```
select products.name_product from products right
join faktur on products.code_product =
```

faktur.code_product group by faktur.code_product
order by count(faktur.code_product);

```
MariaDB [integrasi]> select products.name_product from products right join faktur  
on products.code_product = faktur.code_product group by faktur.code_product order  
by count(faktur.code_product);
```

name_product
Monitor
Pencil
Blender
Microwave
Papper A4
Printer

```
6 rows in set (0.007 sec)
```

25. Terapkan pelanggan yang tidak pernah membeli barang.

select customers.name_customer from customers left
join faktur on customers.code_customer =
faktur.code_customer where faktur.code_customer is
null;

```
MariaDB [integrasi]> select customers.name_customer from customers left join faktur on  
customers.code_customer = faktur.code_customer where faktur.code_customer is null;
```

name_customer
Dea

```
1 row in set (0.001 sec)
```

