



MODUL
DATABASE
(CCD211)

MODUL SESI V

SQL: DATA MANIPULATION

DISUSUN OLEH

NOVIANDI, S.Kom, M.Kom

UNIVERSITAS ESA UNGGUL

2020

BAB V

SQL: Data Manipulation

Tujuan

1. Tujuan dan pentingnya Structured Query Language (SQL).
2. Sejarah dan perkembangan SQL.
3. Bagaimana menulis perintah SQL
4. Cara mengambil data dari database menggunakan pernyataan SELECT
5. Bagaimana membangun pernyataan SQL, yang::
 - a. Menggunakan klausa WHERE untuk mengambil baris yang memenuhi berbagai kondisi.
 - b. Mengurutkan hasil query menggunakan ORDER BY
 - c. Menggunakan fungsi aggregate SQL
 - d. Data kelompok menggunakan GROUP BY
 - e. Menggunakan sub query
 - f. Melakukan set operations (UNION, INTERSECT, EXCEPT).
6. Bagaimana melakukan update database menggunakan INSERT, UPDATE, dan DELETE.

Teori

Pendahuluan

- Bahasa tertentu yang muncul dari pengembangan model relasional adalah Structured Query Language, atau biasa disebut SQL.
- Selama beberapa tahun terakhir, SQL telah menjadi bahasa database relasional standar.
- Pada tahun 1986, standar untuk SQL ditetapkan oleh American National Standards Institute (ANSI) dan kemudian diadopsi pada tahun 1987 sebagai standar internasional oleh Organisasi Internasional untuk Standardisasi (ISO, 1987).
- Lebih dari seratus DBMS sekarang mendukung SQL, berjalan di berbagai platform perangkat keras dari PC hingga mainframe.

Introduction to SQL

Tujuan SQL

Idealnya, bahasa database harus memungkinkan pengguna untuk:

- Membuat database dan struktur relasi
- Melakukan tugas-tugas manajemen data dasar, seperti penyisipan (insertion), modifikasi, dan penghapusan data dari relasi
- Melakukan queri sederhana dan kompleks.

SQL adalah contoh bahasa berorientasi transformasi, atau bahasa yang dirancang untuk menggunakan relasi untuk mengubah *inputs* menjadi *output* yang diperlukan.

Sebagai bahasa, standar ISO SQL memiliki dua komponen utama:

1. Data Definition Language (DDL)
Untuk menentukan struktur database dan mengontrol akses ke data.
2. Data Manipulation Language (DML)
Untuk mengambil dan memperbaharui data.

SQL adalah bahasa yang relatif mudah dipelajari, karena:

1. SQL merupakan bahasa nonprocedural
2. SQL pada dasarnya memiliki format yang bebas, yang berarti bahwa bagian pernyataan tidak harus diketik di lokasi tertentu
3. Struktur perintah terdiri dari kata-kata dalam bahasa inggris standar, seperti: CREATE TABLE, INSERT, SELECT.

Contoh:

- **CREATE TABLE** Staff (staffNo **VARCHAR**(5), IName **VARCHAR**(15), salary **DECIMAL**(7,2));
 - **INSERT INTO** Staff **VALUES** ('SG16', 'Brown', 8300);
 - **SELECT** staffNo, IName, salary
FROM Staff
WHERE salary > 10000;
4. SQL dapat digunakan oleh berbagai pengguna termasuk administrator database (DBA), personel manajemen, pengembang aplikasi, dan banyak jenis end user lainnya.

Sejarah SQL

- 1970-an, system database oracle diproduksi oleh Oracle Corporation dan merupakan implementasi komersial pertama dari DBMS relasional berdasarkan SQL.
- INGRES menyusul dengan bahasa queri yang disebut QUEL yang lebih terstruktur daripada SQL.
- Ketika SQL muncul sebagai bahasa database standar untuk system relasional, INGRES diubah menjadu DBMS berbasis SQL.
- Tahun 1981 dan 1982, IBM memproduksi RDBMS komersial pertama yang disebut dengan SQL/DS untuk lingkungan DOS/VSE dan VM/CMS.
- Tahun 1982, ANSI mulai mengerjakan Relational Database Language (RDL) berdasarkan konsep dari IBM.
- Tahun 1983, IBM memproduksi DB2 untuk lingkungan MVS. Pada tahun yang sama ISO bergabung dalam mendefinisikan standar untuk SQL.
- Tahun 1984, nama RDL dihapus dan draf standar dikembalikan ke bentuk yang lebih mirip dengan implementasi SQL yang ada.
- Awal 1989, ISO menerbitkan sebuah addendum yang mendefinisikan "Fitur Peningkatan Integritas" (ISO, 1989).
- Tahun 1992, revisi besar pertama pada standar ISO terjadi, kadang-kadang disebut sebagai SQL2 atau SQL-92 (ISO, 1992). Meskipun beberapa fitur telah didefinisikan dalam standar untuk pertama kalinya, banyak dari fitur ini telah diimplementasikan sebagian atau dalam bentuk serupa di satu atau lebih dari banyak implementasi SQL.
- Tahun 1999 rilis standar berikutnya, yang biasa disebut sebagai SQL: 1999 (ISO, 1999a), diresmikan. Rilis ini berisi fitur tambahan untuk mendukung manajemen data berorientasi objek.
- Rilis lebih lanjut dari standar pada akhir 2003 (SQL: 2003), pada musim panas 2008 (SQL: 2008), dan pada akhir 2011 (SQL: 2011).

Faktanya, SQL sekarang memiliki serangkaian fitur yang disebut Core SQL yang harus diterapkan oleh vendor untuk mengklaim kesesuaian dengan standar SQL. Banyak dari fitur yang tersisa dibagi menjadi packages; misalnya, ada packages untuk fitur objek dan OLAP (OnLine Analytical Processing).

Pentingnya SQL

- SQL adalah yang pertama dan, sejauh ini, satu-satunya bahasa database standar yang diterima secara luas.
- SQL telah menjadi bagian dari arsitektur aplikasi seperti IBM's Systems Application Architecture (SAA) dan merupakan pilihan strategis dari banyak organisasi besar dan berpengaruh
Misalnya, konsorsium Open Group untuk standar UNIX.
- SQL juga telah menjadi Standar Pemrosesan Informasi Federal (FIPS) yang mengharuskan kesesuaian untuk semua penjualan DBMS kepada pemerintah A.S.
- SQL digunakan dalam standar lain dan bahkan memengaruhi pengembangan standar lain sebagai alat definisi.
Contohnya termasuk standar Sistem Kamus Sumber Daya Informasi (IRDS) ISO dan standar Akses Data Jarak Jauh (RDA).
- Perkembangan bahasa didukung oleh minat akademis yang cukup besar, memberikan dasar teoritis untuk bahasa dan teknik yang dibutuhkan untuk mengimplementasikannya dengan sukses. Hal ini terutama berlaku dalam pengoptimalan query, distribusi data, dan keamanan.
- Saat ini ada implementasi khusus SQL yang diarahkan ke pasar baru, seperti OnLine Analytical Processing (OLAP).

Menulis Perintah SQL

Pernyataan SQL terdiri dari:

1. Kata khusus
 - Bagian tetap dari bahasa SQL dan memiliki arti tetap.
 - Kata khusus harus dieja persis seperti yang diminta dan tidak dapat dipisahkan dan menjadi satu baris.
2. Kata yang ditentukan pengguna
 - Kata yang ditentukan pengguna dibuat oleh pengguna (menurut aturan sintaks tertentu) dan mewakili nama berbagai objek database seperti tabel, kolom, tampilan, indeks, dan sebagainya.
3. Pernyataan SQL dapat diketik dalam huruf besar atau kecil (*case-insensitive*).

Data Manipulation

Statement SQL DML yaitu:

- **SELECT** : Query data dalam database
- **INSERT** : insert data kedalam database
- **UPDATE** : update data yang ada dalam table
- **DELETE** : delete data dari table

Contoh statement SQL menggunakan kejadian pada kasus DreamHome yang terdiri dari table-table sebagai berikut:

Branch	(<u>branchNo</u> , street, city, postcode)
Staff	(<u>staffNo</u> , fName, IName, position, sex, DOB, salary, branchNo)
PropertyForRent	(<u>propertyNo</u> , street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)
Client	(<u>clientNo</u> , fName, IName, telNo, prefType, maxRent, eMail)
PrivateOwner	(<u>ownerNo</u> , fName, IName, address, telNo, eMail, password)
Viewing	(<u>clientNo</u> , propertyNo, viewDate, comment)

Literal

Literal adalah konstanta yang digunakan dalam pernyataan SQL. Ada berbagai bentuk literal untuk setiap tipe data yang didukung oleh SQL

DATA TYPE	DECLARATIONS				
boolean	BOOLEAN				
character	CHAR	VARCHAR			
bit [†]	BIT	BIT VARYING			
exact numeric	NUMERIC	DECIMAL	INTEGER	SMALLINT	BIGINT
approximate numeric	FLOAT	REAL	DOUBLE PRECISION		
datetime	DATE	TIME	TIMESTAMP		
interval	INTERVAL				
large objects	CHARACTER LARGE OBJECT		BINARY LARGE OBJECT		

[†]BIT and BIT VARYING have been removed from the SQL:2003 standard.

Gambar 5.1 ISO SQL Tipe Data

- Semua nilai **data nonnumerik harus diapit tanda kutip tunggal**
- Semua nilai **data numerik tidak boleh diapit tanda kutip tunggal**.

Misalnya, kita bisa menggunakan literal untuk memasukkan data ke dalam tabel:

```
INSERT INTO PropertyForRent(propertyNo, street, city, postcode, type,
                             rooms, rent, ownerNo, staffNo, branchNo)
VALUES ('PA14', '16 Holhead', 'Aberdeen', 'AB7 5SU', 'House', 6,
         650.00, 'CO46', 'SA9', 'B007');
```

Simple Queries

- Tujuan dari pernyataan **SELECT** adalah untuk mengambil dan menampilkan data dari satu atau lebih tabel database.
- **SELECT** adalah perintah yang sangat kuat, mampu melakukan yang setara dengan operasi *Selection*, *Projection*, dan *Join* aljabar relasional dalam satu pernyataan.
- **SELECT** adalah perintah SQL yang paling sering digunakan dan memiliki bentuk umum berikut:

```
SELECT    [DISTINCT | ALL] { * | [columnExpression [AS newName]] [, . . .] }
FROM      TableName [alias] [, . . .]
[WHERE      condition]
[GROUP BY  columnList] [HAVING condition]
[ORDER BY  columnList]
```

Keterangan:

- columnExpression mewakili nama kolom atau ekspresi
- NamaTabel adalah nama tabel atau tampilan database yang ada yang dapat Anda akses,
- alias adalah singkatan opsional untuk NamaTabel.

Urutan pemrosesan dalam pernyataan **SELECT** adalah:

```
FROM      menentukan tabel atau tabel yang akan digunakan
WHERE      memfilter baris yang tunduk pada beberapa kondisi
GROUP BY  membentuk kelompok baris dengan nilai kolom yang sama
```

HAVING memfilter grup yang tunduk pada beberapa kondisi
SELECT menentukan kolom mana yang akan muncul di output
ORDER BY menentukan urutan keluaran

Urutan klausa dalam pernyataan **SELECT** tidak dapat diubah. Dua klausa wajib adalah dua yang pertama: **SELECT** dan **FROM**; sisanya opsional. Operasi **SELECT** ditutup (;).

Contoh memanggil semua kolom dan baris.

Buat daftar detail lengkap dari semua staf.

Cara 1

```
SELECT staffNo, fName, lName, position, sex, DOB, salary, branchNo  
FROM Staff;
```

Cara 2

```
SELECT * FROM Staff;
```

Hasil:

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

Contoh memanggil spesifik kolom dan baris

Buat daftar gaji untuk semua staf, yang hanya menampilkan nomor staf, nama depan dan belakang, dan detail gaji.

```
SELECT staffNo, fName, lName, salary  
FROM Staff;
```


Hasil:

staffNo	fName	lName	salary
SL21	John	White	30000.00
SG37	Ann	Beech	12000.00
SG14	David	Ford	18000.00
SA9	Mary	Howe	9000.00
SG5	Susan	Brand	24000.00
SL41	Julie	Lee	9000.00

Contoh Penggunaan DISTINCT

Buat daftar nomor properti dari semua properti yang telah dilihat.

```
SELECT propertyNo  
FROM Viewing;
```

Hasil:

propertyNo

PA14

PG4

PG4

PA14

PG36

Hasil diatas memiliki beberapa nilai yang duplikasi. Untuk menghilangkan duplikasi tersebut, maka digunakan perintah DISTINCT.

```
SELECT DISTINCT propertyNo  
FROM Viewing;
```

Hasil:

propertyNo

PA14

PG4

PG36

Row Selection (WHERE clause)

Contoh sebelumnya menunjukkan penggunaan pernyataan **SELECT** untuk mengambil semua baris dari tabel. Namun, kami sering kali perlu membatasi baris yang diambil. Ini dapat dicapai dengan klausa **WHERE**, yang terdiri dari kata kunci **WHERE** diikuti dengan kondisi pencarian yang menentukan baris yang akan diambil.

Lima kondisi pencarian dasar (atau predikat, menggunakan terminologi ISO) adalah sebagai berikut:

1. *Comparison* Bandingkan nilai satu ekspresi dengan nilai ekspresi lain.
2. *Range* Menguji apakah nilai ekspresi berada dalam rentang nilai tertentu.
3. *Set membership* Menguji apakah nilai ekspresi sama dengan salah satu set nilai.
4. *Pattern match* Menguji apakah sebuah string cocok dengan pola yang ditentukan.
5. *Null* Uji apakah kolom memiliki nilai null (tidak diketahui).

Contoh Perbandingan kondisi pencarian

Buat daftar semua staf dengan gaji lebih dari £ 10.000

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary > 10000;
```

Hasil:

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG37	Ann	Beech	Assistant	12000.00
SG14	David	Ford	Supervisor	18000.00
SG5	Susan	Brand	Manager	24000.00

Dalam SQL, tersedia operator perbandingan sederhana sebagai berikut:

= *equals*

<> *is not equal to (ISO standard)*

\neq *is not equal to (allowed in some dialects)*
 $<$ *is less than*
 \leq *is less than or equal to*
 $>$ *is greater than*
 \geq *is greater than or equal to*

Predikat yang lebih kompleks dapat dibuat menggunakan operator logika AND, OR, dan NOT, dengan tanda kurung (jika diperlukan atau diinginkan) untuk menunjukkan urutan evaluasi.

Aturan untuk mengevaluasi ekspresi kondisional adalah:

- ekspresi dievaluasi dari kiri ke kanan
- subekspresi dalam tanda kurung dievaluasi terlebih dahulu
- TIDAK dievaluasi sebelum AND dan OR
- AND dievaluasi sebelum OR.

Contoh Compound comparison search condition

Sebutkan alamat semua kantor cabang di London atau Glasgow

```

SELECT *
FROM Branch
WHERE city = 'London' OR city = 'Glasgow';

```

Hasil:

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B003	163 Main St	Glasgow	G11 9QX
B002	56 Clover Dr	London	NW10 6EU

Contoh Range search condition (BETWEEN/NOT BETWEEN)

Buat daftar semua staf dengan gaji antara £ 20.000 dan £ 30.000.

```

SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary BETWEEN 20000 AND 30000;

```

Hasil:

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG5	Susan	Brand	Manager	24000.00

Ada juga versi tes rentang yang dinegasikan (NOT BETWEEN) yang memeriksa nilai-nilai di luar rentang, sebagai berikut:

```
SELECT staffNo, fName, lName, position, salary
FROM Staff
WHERE salary > = 20000 AND salary < = 30000;
```

Sorting Results (ORDER BY Clause)

- Baris table hasil queri dapat diurutkan menggunakan klausa ORDER BY dalam SELECT *statement*.
- Klausa ORDER BY terdiri dari daftar pengidentifikasi kolom yang hasilnya akan diurutkan, dipisahkan dengan koma.
- Klausa ORDER BY memungkinkan baris yang diambil diurutkan dalam urutan menaik/kecil-besar (ASC) atau menurun/besar-kecil (DESC) pada kolom atau kombinasi kolom.

Contoh Single-Column Ordering

Buat daftar gaji untuk semua staf, diatur dalam urutan gaji

```
SELECT staffNo, fName, lName, salary
FROM Staff
ORDER BY salary DESC;
```

Hasil:

staffNo	fName	lName	salary
SL21	John	White	30000.00
SG5	Susan	Brand	24000.00
SG14	David	Ford	18000.00
SG37	Ann	Beech	12000.00
SA9	Mary	Howe	9000.00
SL41	Julie	Lee	9000.00

Contoh multiple column ordering

Menghasilkan daftar singkat properti yang disusun dalam urutan tipe properti.

```
SELECT propertyNo, type, rooms, rent
```

```
FROM PropertyForRent
```

```
ORDER BY type;
```

Hasil:

propertyNo	type	rooms	rent
PL94	Flat	4	400
PG4	Flat	3	350
PG36	Flat	3	375
PG16	Flat	4	450
PA14	House	6	650
PG21	House	5	600

Ada empat flat dalam daftar ini. Karena kami tidak menetapkan kunci pengurutan kecil apa pun, sistem mengatur baris-baris ini dalam urutan apa pun yang dipilihnya. Untuk mengatur properti dalam urutan sewa, kami menetapkan pesanan kecil, sebagai berikut:

```
SELECT propertyNo, type, rooms, rent
```

```
FROM PropertyForRent
```

```
ORDER BY type, rent DESC;
```

Hasil:

propertyNo	type	rooms	rent
PG16	Flat	4	450
PL94	Flat	4	400
PG36	Flat	3	375
PG4	Flat	3	350
PA14	House	6	650
PG21	House	5	600

Menggunakan Fungsi Aggregate SQL

Selain mengambil baris dan kolom dari database, kami sering ingin melakukan beberapa bentuk penjumlahan atau agregasi data, mirip dengan total di bagian bawah laporan. Standar ISO mendefinisikan lima fungsi agregat:

1. COUNT - mengembalikan jumlah nilai dalam kolom tertentu
2. SUM - mengembalikan jumlah nilai dalam kolom tertentu
3. AVG - mengembalikan rata-rata nilai dalam kolom tertentu
4. MIN - mengembalikan nilai terkecil dalam kolom tertentu
5. MAX - mengembalikan nilai terbesar dalam kolom tertentu

Contoh penggunaan COUNT(*)

Berapa banyak properti yang disewakan lebih dari £ 350 per bulan?

```
SELECT COUNT(*) AS myCount  
FROM PropertyForRent  
WHERE rent > 350;
```

Membatasi queri ke properti yang harganya lebih dari £350 per bulan dilakukan dengan menggunakan klausa WHERE. Jumlah total properti yang memenuhi ketentuan ini kemudian dapat ditemukan dengan menerapkan fungsi agregat COUNT.

Hasil:

myCount

5

Contoh menggunakan COUNT(DISTINCT)

Berapa banyak properti berbeda yang dilihat pada Mei 2013?

```
SELECT COUNT(DISTINCT propertyNo) AS myCount  
FROM Viewing  
WHERE viewDate BETWEEN '1-May-13' AND '31-May-13';
```

Hasil:

myCount

2

Contoh menggunakan COUNT dan SUM

Temukan jumlah Manajer dan jumlah gaji mereka.

```
SELECT COUNT(staffNo) AS myCount, SUM(salary) AS mySum  
FROM Staff  
WHERE position = 'Manager';
```

Hasil:

myCount	mySum
2	54000.00

Contoh menggunakan MIN, MAX, AVG

Temukan gaji staf minimum, maksimum, dan rata-rata.

```
SELECT MIN(salary) AS myMin, MAX(salary) AS myMax, AVG(salary) AS  
myAvg  
FROM Staff;
```

Hasil:

myMin	myMax	myAvg
9000.00	30000.00	17000.00

Grouping Results (Group BY Clause)

Klausula GROUP BY dari pernyataan SELECT digunakan untuk mengelompokkan data dari table SELECT dan menghasilkan satu baris ringkasan untuk setiap group. Kolom dalam klausula GROUP BY disebut kolom pengelompokkan.

Klausula SELECT berisikan:

1. Column names
2. Aggregate functions
3. Constans
4. Ekspresi yang melibatkan kombinasi elemen-elemen

Contoh penggunaan GROUP BY

```
SELECT branchNo, COUNT(staffNo) AS myCount, SUM(salary) AS mySum  
FROM Staff
```

GROUP BY branchNo

ORDER BY branchNo;

Hasil:

branchNo	myCount	mySum
B003	3	54000.00
B005	2	39000.00
B007	1	9000.00

Secara konseptual, SQL melakukan query sebagai berikut:

1. SQL membagi staf menjadi beberapa kelompok sesuai dengan nomor cabang masing-masing. Dalam setiap kelompok, semua staf memiliki nomor cabang yang sama. Dalam contoh ini, kami mendapatkan tiga grup:

branchNo	staffNo	salary		COUNT(staffNo)	SUM(salary)
B003	SG37	12000.00	}	3	54000.00
B003	SG14	18000.00			
B003	SG5	24000.00			
B005	SL21	30000.00	}	2	39000.00
B005	SL41	9000.00			
B007	SA9	9000.00	}	1	9000.00

2. Untuk setiap grup, SQL menghitung jumlah anggota staf dan menghitung jumlah nilai di kolom gaji untuk mendapatkan total gaji mereka. SQL menghasilkan satu baris ringkasan dalam hasil kueri untuk setiap grup.
3. Akhirnya, hasilnya diurutkan dalam urutan ascending dari branch number, branchNo.

```
SELECT branchNo, (SELECT COUNT(staffNo) AS myCount
FROM Staff s
WHERE s.branchNo = b.branchNo),
(SELECT SUM(salary) AS mySum
FROM Staff s
WHERE s.branchNo = b.branchNo)
FROM Branch b
ORDER BY branchNo;
```


Membatasi pengelompokan (klausa HAVING)

- Klausa HAVING dirancang untuk digunakan dengan klausa GROUP BY untuk membatasi grup yang muncul di tabel hasil akhir.
- Klausa WHERE memfilter setiap baris yang masuk ke tabel hasil akhir
- Klausa HAVING memfilter mengelompokkan ke dalam tabel hasil akhir.
- Standar ISO mensyaratkan bahwa nama kolom yang digunakan dalam klausa HAVING juga harus muncul dalam daftar GROUP BY atau berada dalam fungsi agregat.

Contoh menggunakan HAVING

Untuk setiap *branch office* yang memiliki lebih dari satu anggota staf, temukan jumlah staf yang bekerja di setiap cabang dan jumlah gaji mereka.

```
SELECT branchNo, COUNT(staffNo) AS myCount, SUM(salary) AS mySum
FROM Staff
GROUP BY branchNo
HAVING COUNT(staffNo) > 1
ORDER BY branchNo;
```

Hasil:

branchNo	myCount	mySum
B003	3	54000.00
B005	2	39000.00

Subqueries

Ada tiga jenis subquery, yaitu:

1. Sebuah subquery skalar mengembalikan satu kolom dan satu baris, yaitu satu nilai. Pada prinsipnya, subquery skalar dapat digunakan setiap kali satu nilai diperlukan.
2. Sebuah subquery baris mengembalikan beberapa kolom, tetapi hanya satu baris. Subquery baris dapat digunakan setiap kali konstruktor nilai baris diperlukan, biasanya dalam predikat.
3. Sebuah subquery tabel mengembalikan satu atau lebih kolom dan beberapa baris. Subquery tabel dapat digunakan kapan pun tabel diperlukan, misalnya, sebagai operan untuk predikat IN.

Contoh menggunakan subquery with equality

Sebutkan staf yang bekerja di cabang di '163 Main St'.

```
SELECT staffNo, fName, IName, position
FROM Staff
WHERE branchNo = (SELECT branchNo
FROM Branch
WHERE street = '163 Main St');
```

SELECT bagian dalam mengembalikan tabel hasil yang berisi nilai tunggal 'B003', sesuai dengan cabang di '163 Main St', dan SELECT luar menjadi:

```
SELECT staffNo, fName, IName, position
FROM Staff
WHERE branchNo = 'B003';
```

Hasil:

staffNo	fName	IName	position
SG37	Ann	Beech	Assistant
SG14	David	Ford	Supervisor
SG5	Susan	Brand	Manager

Menambahkan data kedalam database (INSERT)

Contoh penggunaan notasi INSERT...VALUES

Sisipkan baris baru ke dalam tabel Staff yang menyediakan data untuk semua kolom: staffNo, fName, IName, position, gaji, dan branchNo.

```
INSERT INTO Staff (staffNo, fName, IName, position, salary, branchNo)
VALUES ('SG44', 'Anne', 'Jones', 'Assistant', 8100, 'B003');
```

Contoh penggunaan notasi INSERT....SELECT

Asumsikan bahwa ada tabel StaffPropCount yang berisi nama-nama staf dan jumlah properti yang akan dikelola:

Isi tabel StaffPropCount menggunakan detail dari tabel Staff dan PropertyForRent.

```
INSERT INTO StaffPropCount
(SELECT s.staffNo, fName, IName, COUNT(*)
FROM Staff s, PropertyForRent p
WHERE s.staffNo = p.staffNo
GROUP BY s.staffNo, fName, IName)
UNION
(SELECT staffNo, fName, IName, 0
FROM Staff s
WHERE NOT EXISTS (SELECT * FROM PropertyForRent p
WHERE p.staffNo = s.staffNo));
```

Hasil:

staffNo	fName	IName	propCount
SG14	David	Ford	1
SL21	John	White	0
SG37	Ann	Beech	2
SA9	Mary	Howe	1
SG5	Susan	Brand	0
SL41	Julie	Lee	1

Mengubah data dalam database (UPDATE)

Contoh UPDATE semua baris

Beri semua staf kenaikan gaji 3%.

```
UPDATE Staff
SET salary = salary*1.03;
```

Contoh UPDATE baris tertentu

Beri semua Manajer kenaikan gaji 5%

```
UPDATE Staff
SET salary = salary*1.05
WHERE position = 'Manager';
```

Contoh UPDATE untuk multiple columns

Promosikan David Ford (staffNo = 'SG14') menjadi Manajer dan ubah gajinya menjadi £ 18,000.

UPDATE Staff

SET position = 'Manager', salary = 18000

WHERE staffNo = 'SG14';

Menghapus data dari database (DELETE)

Contoh DELETE baris tertentu

Hapus semua tampilan yang berhubungan dengan properti PG4.

DELETE FROM Viewing

WHERE propertyNo = 'PG4';

Contoh DELETE semua baris

Hapus semua baris dari tabel Viewing.

DELETE FROM Viewing

Latihan

1. Jelaskan secara singkat empat pernyataan SQL DML dasar dan jelaskan penggunaannya.
2. Jelaskan pentingnya dan penerapan klausa WHERE dalam pernyataan UPDATE dan DELETE.
3. Jelaskan fungsi dari masing-masing klausa dalam pernyataan SELECT?
4. Batasan apa yang berlaku untuk penggunaan fungsi agregat dalam pernyataan SELECT? Bagaimana null mempengaruhi fungsi agregat?
5. Bagaimana hasil dari dua query SQL dapat digabungkan? Bedakan cara kerja perintah INTERSECT dan EXCEPT.
6. Membedakan antara tiga jenis subkueri. Mengapa penting untuk memahami sifat hasil subquery sebelum Anda menulis pernyataan SQL?