Rakamin Academy

Bank Muamalat

Project Based Internship

# Google BigQuery: Basics Query

## Daftar Isi

## A. Basic SQL clauses: SELECT, FROM, and WHERE

*SQL statement* biasanya berawal dari:

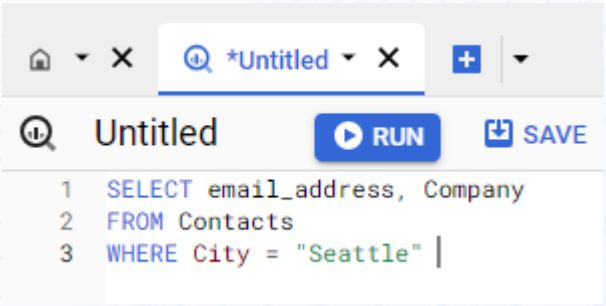
```
SELECT field_1
FROM table_1
WHERE criterion_1 ;
```

**NOTE :**

- Access mengabaikan *line breaks* dalam *SQL statement*. Namun, pertimbangkan untuk menggunakan satu baris untuk setiap klausa untuk membantu meningkatkan keterbacaan *SQL statement* untuk diri sendiri dan orang lain.
- Setiap SELECT *statement* diakhiri dengan tanda titik koma (;). Tanda titik koma dapat muncul di akhir klausa terakhir atau pada baris tersendiri di akhir *SQL statement*.

**Contoh pada *Google Big Query***

Berikut ini mengilustrasikan seperti apa *SQL statement* untuk *query* pilih sederhana di *Google Big Query*:




1. SELECT *clause*
2. FROM *clause*
3. WHERE *clause*

Contoh *SQL statement* ini berbunyi "Pilih data yang disimpan di *field* bernama *E-mail Address* dan *Company* dari tabel bernama *Contacts*, khususnya catatan yang nilai *field*-nya adalah Kota Seattle."

Mari kita lihat contoh tersebut dari satu per satu klausa untuk melihat bagaimana sintaks SQL bekerja.

1. **SELECT *clause***

   **SELECT [E-mail Address], Company**

   Klausa ini terdiri dari operator (SELECT) yang diikuti dengan dua pengidentifikasi ([E-mail Address] dan Company).

   Jika identifier berisi spasi atau karakter khusus (seperti "E-mail Address"), maka harus diapit dengan tanda kurung siku. Klausa SELECT tidak harus menyebutkan tabel mana yang berisi *field*, dan tidak dapat menentukan kondisi apa pun yang harus dipenuhi oleh data yang akan disertakan.

2. **FROM *clause***

   **FROM Contacts**

   Klausa ini terdiri dari operator (FROM) yang diikuti dengan pengenal (Contacts). Klausa FROM tidak mencantumkan *field* yang akan dipilih.

3. **WHERE *clause***

   **WHERE City="Seattle"**

   Klausa ini terdiri dari operator (WHERE) yang diikuti dengan ekspresi (City="Seattle").

   **Note**: Tidak seperti klausa SELECT dan FROM, klausa WHERE bukan merupakan elemen yang wajib ada dalam SELECT *statement*.

   Ada banyak tindakan yang dapat dilakukan oleh SQL dengan menggunakan klausa SELECT, FROM, dan WHERE. Informasi lebih lanjut

tentang cara menggunakan klausa-klausa ini disajikan dalam artikel-artikel tambahan ini:

- Access SQL: SELECT clause
- Access SQL: FROM clause
- Access SQL: WHERE clause

## B. Menyortir Hasil: ORDER BY

Seperti Microsoft Excel, Google Big Query memungkinkan bisa mengurutkan hasil *query* dalam *datasheet* dan juga dapat menentukan urutan hasil ketika *query* dijalankan dengan menggunakan klausa ORDER BY. Klausa ORDER BY adalah klausa terakhir dalam SQL *statement*. Klausa ORDER BY berisi daftar *field* yang ingin digunakan untuk pengurutan, dengan urutan yang sama dengan urutan yang ingin diterapkan pada operasi pengurutan.

Sebagai contoh, misalkan Anda ingin hasil Anda diurutkan terlebih dahulu berdasarkan nilai *field Company* dalam urutan menurun, dan - jika ada catatan dengan nilai yang sama untuk *Company* - diurutkan berikutnya berdasarkan nilai dalam *field E-mail Address* dalam urutan menaik. Klausa ORDER BY Anda akan terlihat seperti berikut ini:

```
ORDER BY Company DESC, [E-mail Address]
```

**Note:** Secara default, Access mengurutkan nilai dalam urutan menaik (A-Z, terkecil ke terbesar). Gunakan kata kunci DESC untuk mengurutkan nilai dalam urutan menurun.

## C. Meringkas Data: GROUP BY dan HAVING

Untuk merangkum data seperti total penjualan dalam sebulan, atau barang termahal dalam inventaris, kita bisa menerapkan fungsi agregat ke suatu bidang dalam klausa SELECT. Misalnya, jika ingin *query* menampilkan jumlah alamat email yang terdaftar untuk setiap perusahaan, klausa SELECT akan seperti berikut ini:

```
SELECT COUNT([E-mail Address]), Company
```

Fungsi agregat yang dapat digunakan bergantung pada jenis data yang ada di *field* atau *expression* yang ingin digunakan.

### 1. GROUP BY clause

Ketika menggunakan fungsi agregat, Anda biasanya juga harus membuat klausa GROUP BY. Klausa GROUP BY mencantumkan semua bidang yang tidak diterapkan fungsi agregat. Jika menerapkan fungsi agregat ke semua bidang dalam *query* , Anda tidak perlu membuat klausa GROUP BY.

Klausa GROUP BY langsung mengikuti klausa WHERE, atau klausa FROM jika tidak ada klausa WHERE. Klausa GROUP BY mencantumkan *field-field* yang muncul di dalam klausa.

**Klausa SELECT**

Sebagai contoh, melanjutkan contoh sebelumnya, jika klausa SELECT menerapkan fungsi agregat ke [*E-mail Address*] tetapi tidak ke *Company*, klausa GROUP BY akan terlihat seperti berikut:

```
GROUP BY Company
```

## 2. HAVING clause

Jika ingin menggunakan kriteria untuk membatasi hasil, tetapi *field* yang ingin diterapkan kriteria digunakan dalam fungsi agregat, Kita tidak bisa menggunakan klausa WHERE. Sebagai gantinya, Kita menggunakan klausa HAVING. Klausa HAVING berfungsi seperti klausa WHERE, tetapi digunakan untuk data agregat.

Sebagai contoh, misalkan menggunakan fungsi AVG (yang menghitung nilai rata-rata) dengan *field* pertama dalam klausa SELECT:

```
SELECT COUNT([E-mail Address]), Company
```

Jika ingin *query* membatasi hasil berdasarkan nilai fungsi COUNT, Kita tidak dapat menggunakan kriteria untuk *field* tersebut dalam klausa WHERE. Sebagai gantinya, meletakkan kriteria dalam klausa HAVING. Misalnya, jika hanya ingin *query* mengembalikan baris jika ada lebih dari satu alamat email yang terkait dengan perusahaan, klausa HAVING akan seperti berikut ini:

```
HAVING COUNT([E-mail Address])>1
```

**Note:** *Query* dapat memiliki klausa WHERE dan klausa HAVING - kriteria untuk bidang yang tidak digunakan dalam fungsi agregat dimasukkan ke dalam klausa WHERE, dan kriteria untuk *field* yang digunakan dengan fungsi agregat dimasukkan ke dalam klausa HAVING.

Untuk informasi lebih lanjut tentang klausa HAVING, lihat topik Klausa HAVING Clause.

## *Case Study*

Anda sebagai seorang *Business Intelligence Analyst* di Bank Muamalat diminta untuk membuat sebuah laporan yang memberikan gambaran tentang performa cabang-cabang di seluruh negeri. Laporan ini harus mencakup informasi tentang total nasabah, total volume tabungan, dan rata-rata volume tabungan per nasabah. Selain itu, hanya cabang-cabang yang memiliki rata-rata volume tabungan per nasabah di atas Rp. 5.000.000 yang akan dimasukkan dalam laporan.

Struktur table: nama table Dana

| ID_Cabang | Nama_Cabang | Kota | Region | ID_Nasabah | Volume_Tabungan | Tanggal_Transaksi |
|---|---|---|---|---|---|---|
| 1 | Cabang A | Jakarta | Jabar | 101 | 5000000 | 1/1/2023 |
| 1 | Cabang A | Jakarta | Jabar | 102 | 7000000 | 1/1/2023 |
| 1 | Cabang A | Jakarta | Jabar | 103 | 4500000 | 1/2/2023 |
| 2 | Cabang B | Surabaya | Jatim | 104 | 6000000 | 1/1/2023 |
| 2 | Cabang B | Surabaya | Jatim | 105 | 8000000 | 1/2/2023 |
| 3 | Cabang C | Bandung | Jabar | 106 | 5500000 | 1/1/2023 |
| 3 | Cabang C | Bandung | Jabar | 107 | 7500000 | 1/2/2023 |
| 3 | Cabang C | Bandung | Jabar | 108 | 5000000 | 1/2/2023 |
| 4 | Cabang D | Medan | Sumut | 109 | 9000000 | 1/1/2023 |
| 4 | Cabang D | Medan | Sumut | 110 | 6000000 | 1/1/2023 |

Hasil yang diharapkan dalam report adalah: ID_Cabang, Nama_Cabang, Kota, Region, Total_Nasabah, Volume_Tabungan, dan Rata_Rata_Volume_Tabungan_Per_Nasabah.

**Hasil query:**

```
SELECT
    ID_Cabang,
    Nama_Cabang,
    Kota,
    Region,
    COUNT(DISTINCT ID_Nasabah) AS Total_Nasabah,
    SUM(Volume_Tabungan) AS Total_Volume_Tabungan,
    AVG(Volume_Tabungan * 1.0) AS
   Rata_Rata_Volume_Tabungan_Per_Nasabah
```

```
FROM
    Dana
GROUP BY
    ID_Cabang, Nama_Cabang, Kota, Region
HAVING
    AVG(Volume_Tabungan * 1.0) > 5000000;
```

**Hasil tabel:**

| ID_Cabang | Region | Total_Nasabah | Total_Volume_Tabungan | AVG_Volume |
|---|---|---|---|---|
| 101 | Jakarta | 5 | 2800000 | 560000.0 |
| 102 | Surabaya | 3 | 1200000 | 400000.0 |
| 103 | Medan | 2 | 1000000 | 500000.0 |

## References

*Access SQL: basic concepts, vocabulary, and syntax*. (n.d.). Microsoft Support. Retrieved October 26, 2023, from https://support.microsoft.com/en-us/office/access-sql-basic-concepts-vocabulary-and-syntax-444d0303-cde1-424e-9a74-e8dc3e460671