**LAPORAN PRAKTIKUM MAHASISWA**

**PEMEROGRAMAN SQL II**

**“SELECT, SUB QUERY DAN JOIN”**

**Minggu 5, 6, dan 7**

****

**Oleh:**

**NAMA : Haldian**

**NPM : 20753050**

**KELAS : Manajemen Informatika B**

**Dosen : Eko Win Kenali, S.Kom, M.Cs**

**MANAJEMEN INFORMATIKA**

**JURUSAN EKONOMI DAN BISNIS**

**POLITEKNIK NEGERI LAMPUNG**

**2022**

# **KATA PENGANTAR**

Alhamdulillah, puji syukur ke hadirat Allah SWT yang telah memberikan rahmat dan inayah-Nya sehingga kami dapat menyelesaikan Laporan Praktikum Pemrograman SQL Lanjut tentang *Select, Sub Query dan Join*.

Terima kasih saya ucapkan kepada Bapak Eko Win Kenali dan Kak Nabila yang telah membantu kami baik secara moral maupun materi. Terima kasih juga saya ucapkan kepada teman-teman seperjuangan yang telah mendukung kami sehingga kami bisa menyelesaikan tugas ini tepat waktu.

Kami menyadari, bahwa laporan praktikum yang kami buat ini masih jauh dari kata sempurna baik segi penyusunan, bahasa, maupun penulisannya. Oleh karena itu, kami sangat mengharapkan kritik dan saran yang membangun dari semua pembaca guna menjadi acuan agar penulis bisa menjadi lebih baik lagi di masa mendatang. Semoga laporan praktikum ini bisa menambah wawasan para pembaca dan bisa bermanfaat untuk perkembangan dan peningkatan ilmu pengetahuan.

Kalainda, 5 Maret 2022

HALDIAN

# **DAFTAR ISI**

[**DAFTAR ISI** 3](#_Toc100585845)

[**BAB 1. PENDAHULUAN** 4](#_Toc100585846)

[**A. LATAR BELAKANG** 4](#_Toc100585847)

[**B. MASALAH** 4](#_Toc100585848)

[**C. TUJUAN** 4](#_Toc100585849)

[**BAB 2. LANDASAN TEORI** 5](#_Toc100585850)

[**A.** ***TRANSACT-SQL SELECT*** 5](#_Toc100585851)

[B. *TRANSACT-SQL SELECT* DENGAN *JOIN* 10](#_Toc100585852)

[**C. *TRANSACT-SQL SELECT* DENGAN FUNGSI *BUILT-IN*** 13](#_Toc100585853)

[**BAB 3. HASIL DAN PEMBAHASAN** 16](#_Toc100585854)

[**A. TAHAPAN PRAKTIKUM** 16](#_Toc100585855)

[**1. Minggu #5** 16](#_Toc100585856)

[**2. Minggu #6** 30](#_Toc100585857)

[**3. Minggu #7** 35](#_Toc100585858)

[**B.** **TUGAS DAN PERTANYAAN** 41](#_Toc100585859)

[**1. Minggu #5** 41](#_Toc100585860)

[**2. Minggu #6** 46](#_Toc100585861)

[**3. Minggu #7** 54](#_Toc100585862)

[**BAB 4. PENUTUP** 59](#_Toc100585863)

[**A. KESIMPULAN** 59](#_Toc100585864)

[**B. SARAN** 59](#_Toc100585865)

[**PUSTAKA** 60](#_Toc100585866)

# **BAB 1. PENDAHULUAN**

**A. LATAR BELAKANG**

SQL (Structured Query Language) merupakan suatu sintaks atau perintah yang di gunakan untuk mengakses data dalam sebuah *database*, SQL sendiri juga bisa digunakan untuk memanipulasi sebuah data dalam database seperti untuk menambah data (*insert*), mengubah data (*update*), menampilkan data (*select*), dan juga menghapus data (*delete*). Biasanya sintaks ini disebut dengan *Data Manipulation Language* (DML) yang merupakan bagian dari SQL (Structured Query Language).

DML(Data Manipulation Language) merupakan kumpulan suatu perintah-perintah yang juga berfungsi untuk memanipulasi sebuah data dalam *database*, mulai dari menambah data, mengubah data, menampilkan data, dan menghapus data. Yang merupakan dari kumpulan perintah DML yaitu *INSERT, UPDATE, SELECT,* dan *DELETE*. Namun pada laporan untuk kali ini kita hanya akan memperdalam perintah SQL mengenai *SELECT* dengan *Sub-query*, dan *JOIN*.

**B. MASALAH**

1. Mahasiswa belum memahami apa itu *sub-query* dan *JOIN* pada perintah *SELECT*.
2. Mahasiswa belum mengerti bagaimana penerapan *sub-query* dan *JOIN*.
3. Mahasiswa masih belum bisa memaksimalkan penerapan *sub-query* dan *JOIN*.

**C. TUJUAN**

1. Mahasiswa mampu memahami apa itu *sub-query* dan *JOIN* pada perintah *SELECT*.
2. Mahasiswa mampu memahami bagaimana penerapan *sub-query* dan *JOIN*.
3. Mahasiswa mampu memaksimalkan penerapan sub-query dan JOIN.

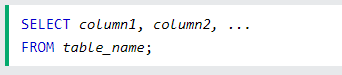
**BAB 2. LANDASAN TEORI**

1. ***TRANSACT-SQL SELECT***

* ***SELECT***

Pernyataan SELECT digunakan untuk memilih data dari database. Data yang dikembalikan disimpan dalam tabel hasil, yang disebut kumpulan hasil.

* **SELECT Syntax**

****

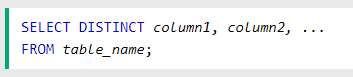
Di sini, kolom1, kolom2, ... adalah nama bidang tabel yang ingin Anda pilih datanya. Jika Anda ingin memilih semua bidang yang tersedia dalam tabel, gunakan sintaks berikut:

****

1. ***DISTINCT***

Pernyataan *SELECT DISTINCT* digunakan untuk mengembalikan nilai yang *distinct* (berbeda). Kata kunci *DISTINCT* menghilangkan *record* yang duplikat dari hasil.

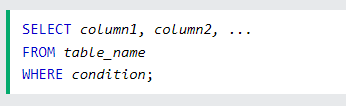
* **SELECT DISTINCT Syntax**

****

1. ***WHERE***

WHERE digunakan untuk memfilter record. Ini digunakan untuk mengekstrak hanya catatan yang memenuhi kondisi tertentu. WHERE tidak hanya digunakan dalam pernyataan SELECT, tetapi juga digunakan dalam UPDATE, DELETE, dll.

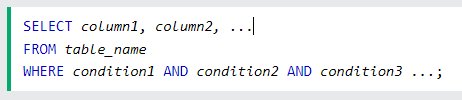
* **WHERE Syntax**

****

1. ***AND, OR, NOT***

WHERE dapat digabungkan dengan operator AND, OR, dan NOT. Operator AND dan OR digunakan untuk memfilter record berdasarkan lebih dari satu kondisi:

* Operator AND menampilkan record jika semua kondisi yang dipisahkan oleh AND adalah TRUE.
* Operator OR menampilkan record jika salah satu kondisi yang dipisahkan oleh OR adalah TRUE.
* Operator NOT menampilkan record jika kondisi NOT TRUE.
* **AND Syntax**

****

* **OR Syntax**

****

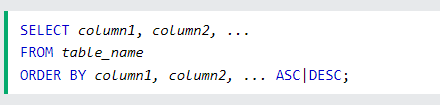
* **NOT Syntax**

****

1. ***ORDER BY***

Kata kunci *ORDER BY* digunakan untuk mengurutkan hasil dalam urutan naik atau turun. Kata kunci *ORDER BY* mengurutkan rekaman dalam urutan naik secara *default*. Untuk mengatur urutan hasil dapat menggunakan kata kunci *ASC/DESC*.

* **ORDER BY Syntax**

****

1. ***GROUP BY***

*GROUP BY* mengelompokkan baris yang memiliki nilai yang sama ke dalam baris ringkasan. Pernyataan *GROUP BY* sering digunakan dengan fungsi agregat (*COUNT(), MAX(), MIN(), SUM(), AVG()*) untuk mengelompokkan hasil yang ditetapkan oleh satu atau lebih kolom.

* **GROUP BY Syntax**

****

1. ***HAVING***

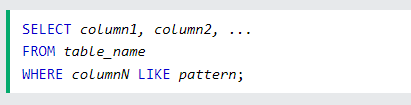
*HAVING* ditambahkan ke SQL karena kata kunci *WHERE* tidak dapat digunakan dengan fungsi agregat.

****

1. ***LIKE***

Operator *LIKE* digunakan dalam klausa *WHERE* untuk mencari pola tertentu dalam kolom. Ada dua karakter pengganti yang sering digunakan terkait dengan operator *LIKE*: Tanda persen (%) mewakili nol, satu, atau beberapa karakter dan Tanda garis bawah (\_) mewakili satu karakter tunggal. *LIKE* juga dapat digabungkan dengan beberapa kondisi menggunakan operator *AND* atau *OR*.

* **Oprator Like**

****

1. ***WILDCARD***

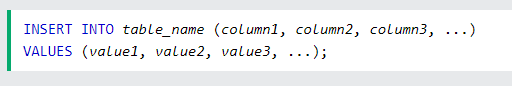
*wildcard* digunakan untuk menggantikan satu atau lebih karakter dalam sebuah *string*. Karakter *wildcard* digunakan dengan operator *LIKE*. Operator *LIKE* digunakan dalam klausa *WHERE* untuk mencari pola tertentu dalam kolom.

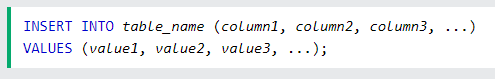
|  |  |  |
| --- | --- | --- |
| **Sbl.** | **Deskripsi** | **Contoh Penggunaan** |
| **%** | Mewakili nol atau lebih karakter. | bl% menemukan bl, black, blue, dan blob. |
| **\_** | Merupakan satu karakter. | h\_t menemukan hot, hat, dan hit. |
| **[]** | Mewakili karakter tunggal apa pun dalam tanda kurung. | h[oa]t menemukan hot dan hat, tetapi tidak hit. |
| **^** | Mewakili karakter apa pun yang tidak ada di dalam tanda kurung. | h[^oa]t menemukan hit, tetapi tidak hot dan hat. |
| **-** | Mewakili berbagai karakter. | c[a-b]t menemukan cat dan cbt. |
|  | | |

1. ***INSERT***

Pernyataan INSERT INTO digunakan untuk menyisipkan record baru ke dalam tabel

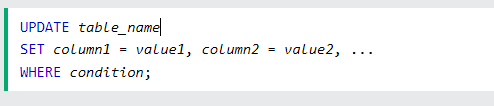
* **INSERT INTO Syntax**

****

****

1. ***UPDATE***

Pernyataan UPDATE digunakan untuk mengubah catatan yang ada dalam tabel.

****

1. ***DELETE***

Pernyataan DELETE digunakan untuk menghapus record yang ada dalam tabel.

****

1. ***DROP***

Pernyataan DROP TABLE digunakan untuk menghapus tabel yang ada dalam database.

****

1. ***TRUNCATE***

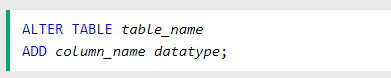
RUNCATE TABLE digunakan untuk menghapus data di dalam tabel, tetapi bukan tabel itu sendiri

****

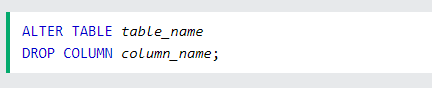
1. ***ALTER***

ALTER TABLE digunakan untuk menambah, menghapus, atau memodifikasi kolom dalam tabel yang sudah ada.

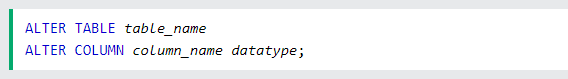
* **ALTER TABLE - ADD Column**

****

* **ALTER TABLE - DROP COLUMN**

****

* **ALTER TABLE - ALTER/MODIFY COLUMN**

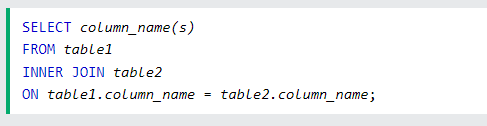
****

## B. *TRANSACT-SQL SELECT* DENGAN *JOIN*

1. ***JOIN***

*JOIN* digunakan untuk menggabungkan baris dari dua atau lebih tabel, berdasarkan kolom terkait di antara mereka. *INNER JOIN* mengembalikan rekaman yang memiliki nilai yang cocok di kedua tabel

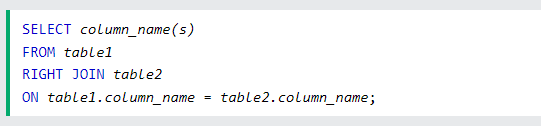
* **INER JOIN Syntax**

****

1. ***RIGHT JOIN***

Kata kunci *RIGHT JOIN* mengembalikan semua *record* dari tabel kanan (table2), dan *record* yang cocok dari tabel kiri (table1). Hasilnya adalah 0 *record* dari sisi kiri, jika tidak ada yang cocok.

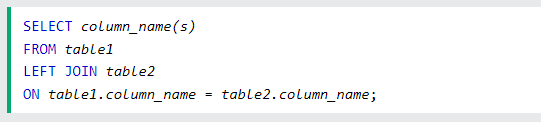
* **RIGHT JOIN Syntax**

****

1. ***LEFT JOIN***

Kata kunci *LEFT JOIN* mengembalikan semua *record* dari tabel kiri (tabel1), dan *record* yang cocok dari tabel kanan (tabel2). Hasilnya adalah 0 *record* dari sisi kanan, jika tidak ada yang cocok.

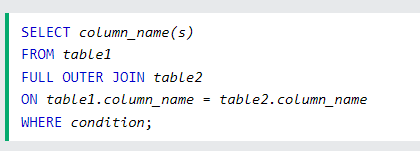
* **LEFT JOIN Syntax**

****

1. ***FULL JOIN***

Kata kunci *FULL OUTER JOIN* mengembalikan semua *record* ketika ada kecocokan dalam *record* tabel kiri (tabel1) atau kanan (tabel2).

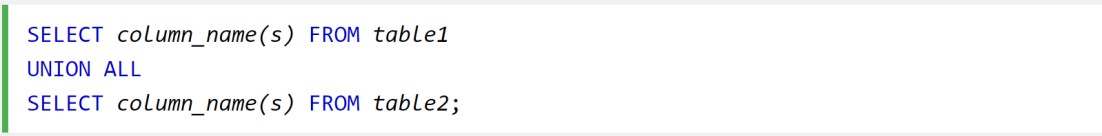
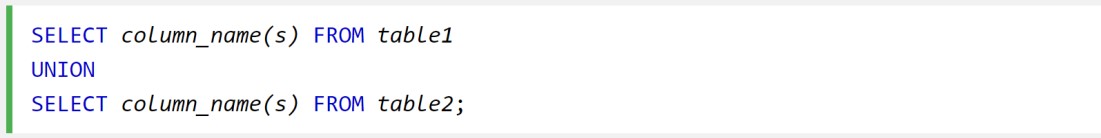
* **FULL JOIN Syntax**

****

1. ***UNION***

Operator *UNION* digunakan untuk menggabungkan kumpulan hasil dari dua atau lebih pernyataan *SELECT*. Setiap pernyataan *SELECT* dalam *UNION* harus memiliki jumlah kolom yang sama, Kolom juga harus memiliki tipe data yang serupa, Kolom di setiap pernyataan SELECT juga harus dalam urutan yang sama.

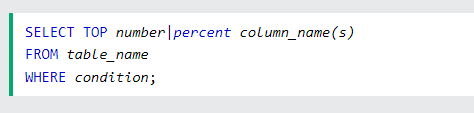
* **UNION Syntax**



1. ***SELECT TOP***

*SELECT TOP* digunakan untuk menentukan jumlah *record* yang akan dikembalikan. *SELECT TOP* berguna pada tabel besar dengan ribuan *record*. Mengembalikan sejumlah besar rekaman dapat memengaruhi kinerja.

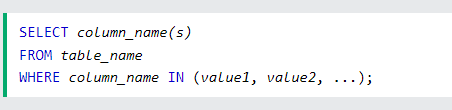
* **SELECT TOP Syntax**

****

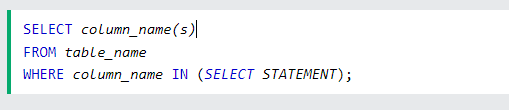
1. ***IN***

Operator *IN* memungkinkan Anda untuk menentukan beberapa nilai dalam klausa *WHERE*. Operator *IN* adalah singkatan dari beberapa kondisi *OR*.

* **IN Syntax**

****

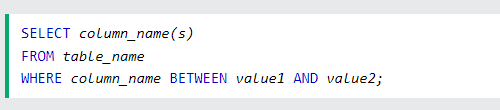
**OR Syntax**

****

1. ***BETWEEN***

Operator *BETWEEN* memilih nilai dalam rentang tertentu. Nilainya bisa berupa angka, teks, atau tanggal. Operator *BETWEEN* bersifat inklusif: nilai awal dan akhir disertakan.

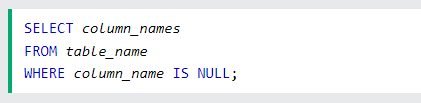
* ***BETWEEN* Syntax**

****

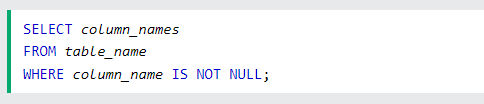
1. ***ISNULL***

Fungsi *ISNULL*() mengembalikan nilai yang ditentukan jika ekspresinya adalah *NULL*. Jika ekspresi *NOT NULL*, fungsi ini mengembalikan ekspresi tersebut.

* **IS NULL Syntax**

****

* **IS NOT NULL Syntax**

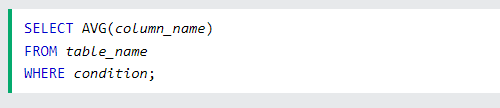
****

**C. *TRANSACT-SQL SELECT* DENGAN FUNGSI *BUILT-IN***

1. ***AVG()***

Fungsi *AVG()* mengembalikan nilai rata-rata kolom numerik.

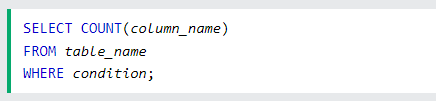
* **AVG() Syntax**

****

1. ***COUNT()***

Fungsi *COUNT()* mengembalikan jumlah rekaman yang dikembalikan oleh *SELECT*.

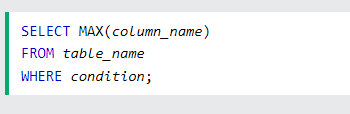
* ***COUNT() Syntax***

****

1. ***MAX()***

Fungsi *MAX()* mengembalikan nilai maksimum dalam satu set nilai.

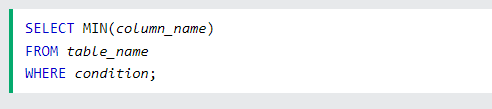
* **MAX() Syntax**

****

1. ***MIN()***

Fungsi *MIN()* mengembalikan nilai minimum dalam satu set nilai.

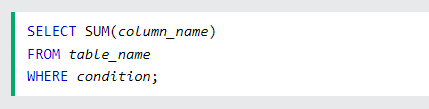
* **MIN() Syntax**

****

1. ***SUM()***

Fungsi *SUM()* menghitung jumlah sekumpulan nilai.

* ***SUM() Syntax***

****

1. ***MID()***

Fungsi MID() mengekstrak substring dari string (dimulai dari posisi apa pun). Catatan: Fungsi MID() dan SUBSTR() sama dengan fungsi SUBSTRING().

**. **

1. ***LEN()***

Fungsi *LEN()* mengembalikan panjang *string*.

****

1. ***ROUND()***

Fungsi *ROUND()* membulatkan angka ke sejumlah tempat desimal tertentu.

****

1. ***NOW()***

NOW() mengembalikan tanggal dan waktu saat ini. Catatan: Tanggal dan waktu ditampilkan sebagai "YYYY-MM-DD HH-MM-SS" (string) atau sebagai YYYYMMDDHHMMSS.uuuuuu (numerik).

****

1. ***GETDATE()***

Fungsi *GETDATE()* mengembalikan tanggal dan waktu sistem *database* saat ini, dalam format 'YYYY-MM-DD hh:mm:ss.mmm'.

****

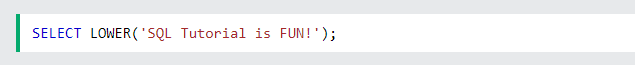
1. ***UPPER()***

Fungsi *UPPER()* mengubah *string* menjadi huruf besar.

****

1. ***LOWER()***

Fungsi *LOWER()* mengubah *string* menjadi huruf kecil.

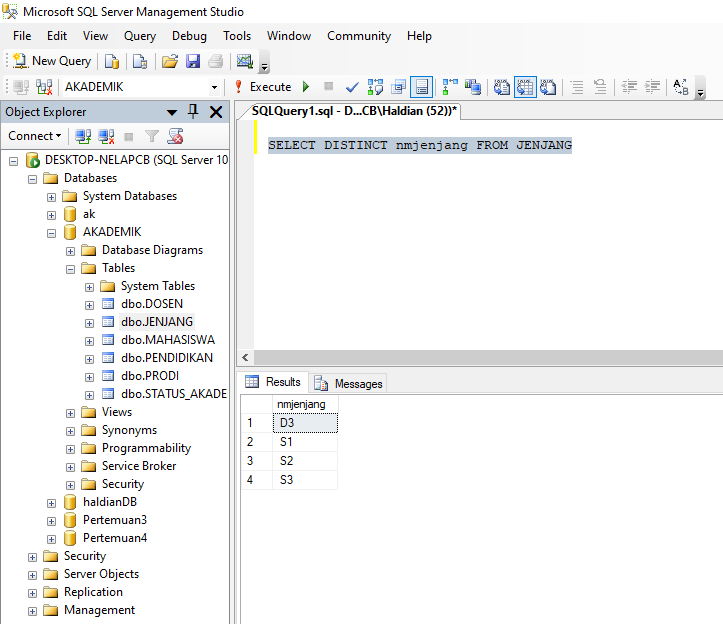
****

# **BAB 3. HASIL DAN PEMBAHASAN**

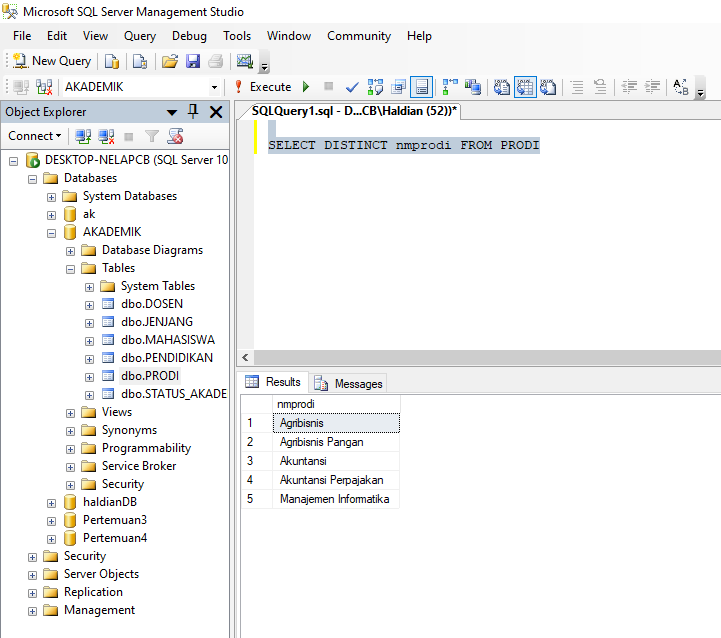
# **A. TAHAPAN PRAKTIKUM**

**1. Minggu #5**

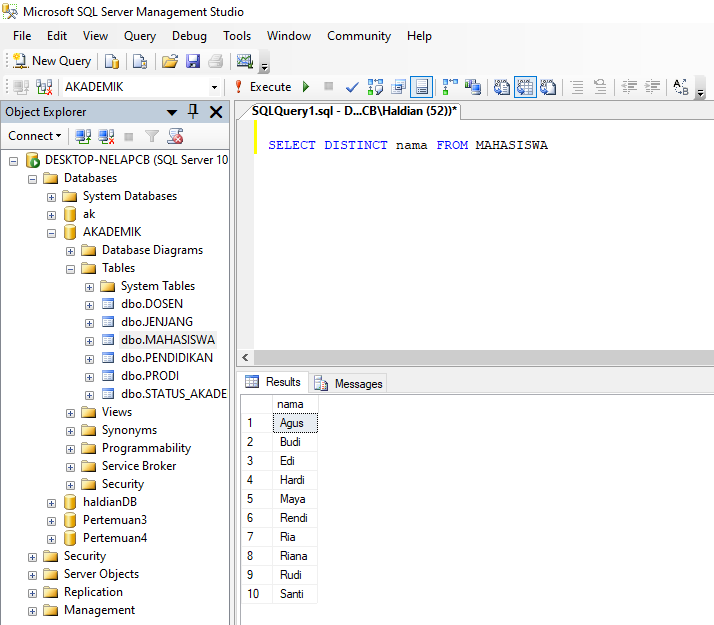
1. Persiapkan alat dan bahan
2. Menjalankan aplikasi: *Microsoft SQL Server Management Studio* (MSQLS-MS)
3. Gunakanlah perintah *transact-SQL Query* (*SELECT*) untuk menyajikan data dari tabel-tabel yang ada pada *database* “AKADEMIK”, dengan memanfaatkan klausa pendukung berikut: *DISTINCT, WHERE, AND, OR, ORDER BY, GROUP BY, HAVING, LIKE,* dan *WILDCARD*. (catatan: masing-masing klausa pendukung 3 contoh perintah, dengan data yang berbeda)
4. **DISTINCT**
5. Menampilkan nama jenjang pada tabel jenjang seperti gambar di bawah ini

****

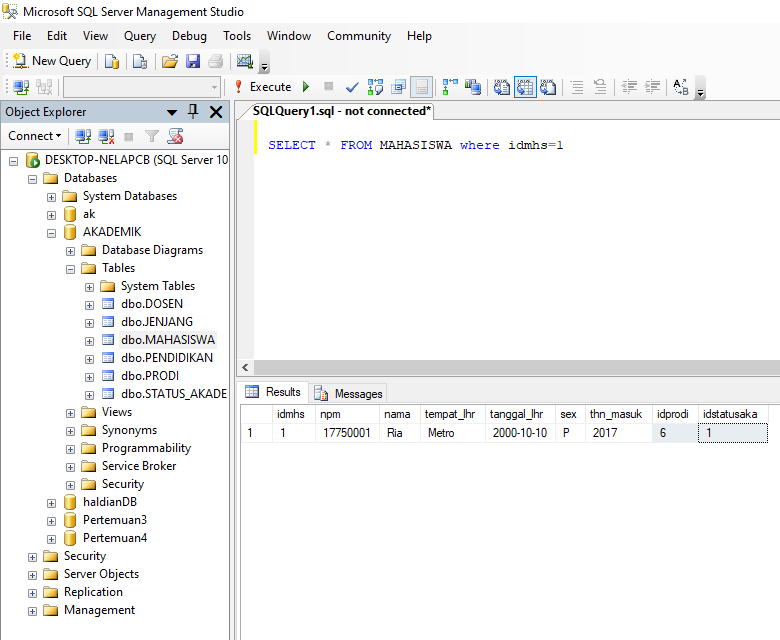
1. Menampilkan namaprodi pada table prodi seperti gambar di bawah ini.

****

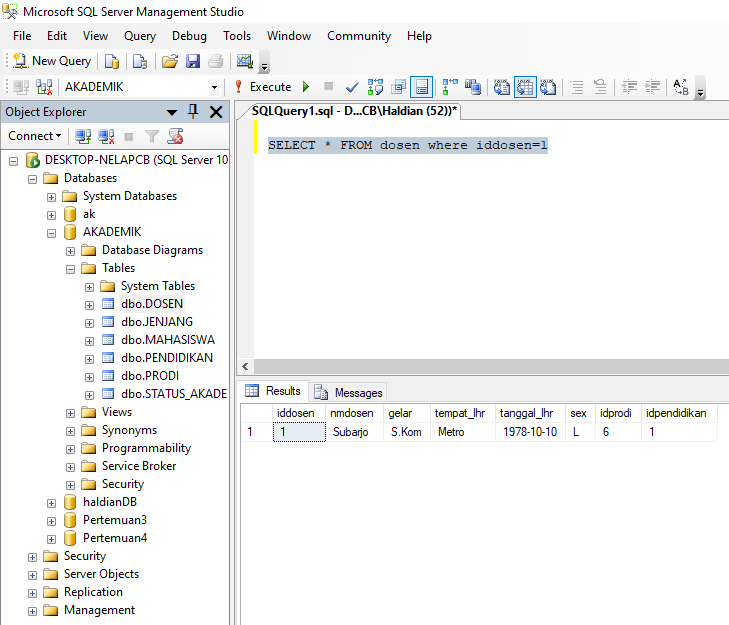
1. Menampilkan nama mahasiswa pada table mahasiswa seperti gambar di bawah ini

****

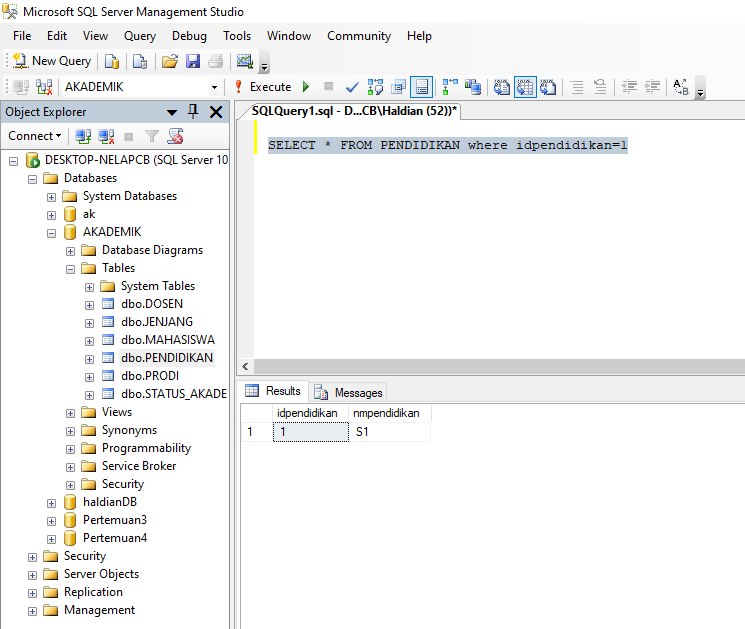
1. **WHERE**
2. Menampilkan mahasiswa dengan idmhs=1 seperti gambar di bawah ini

****

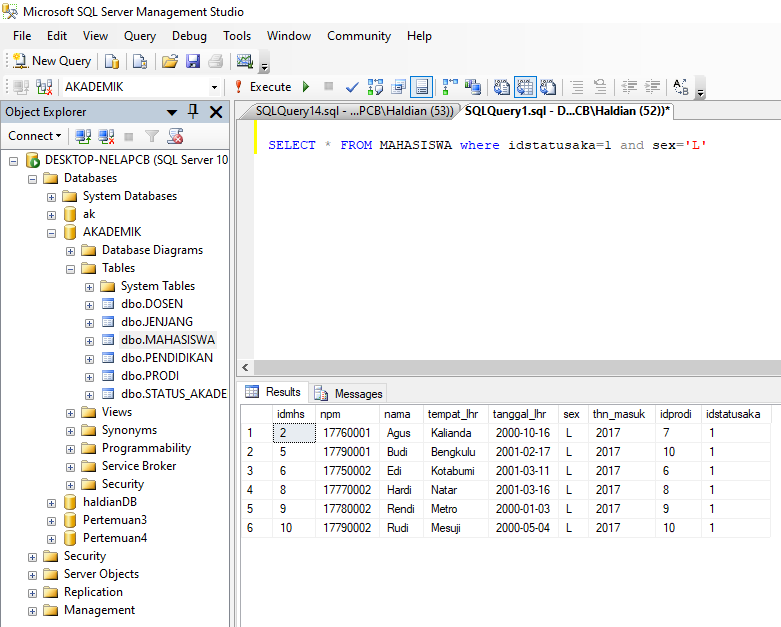
1. Menampilkan dosen dengan iddosen=1 seperti gambar di bawah ini

****

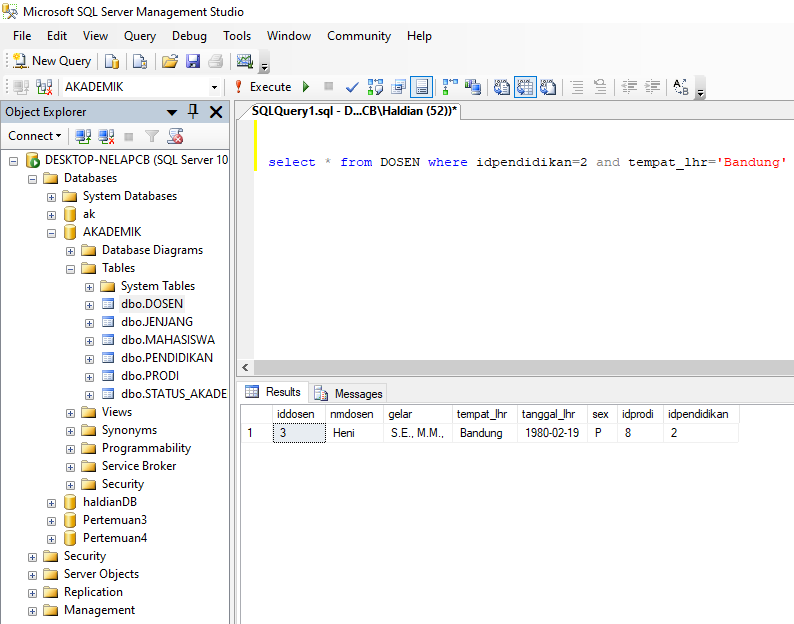
1. Menampilkan pendidikan dengan idpendidikan=1 seperti gambar di bawah ini

****

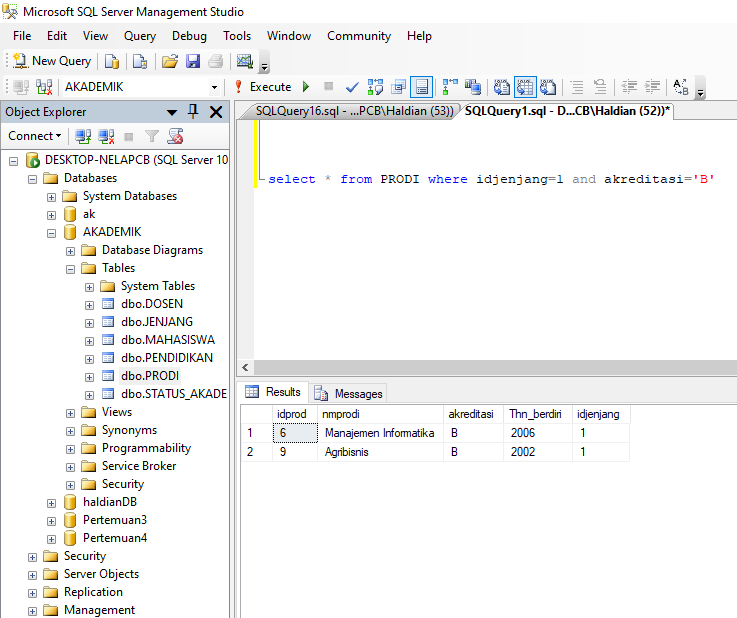
1. **AND**
   1. Menampilkan mahasiswa dari idstatusaka=1 dan menampilkan laki-laki seperti gambar di bawah ini

****

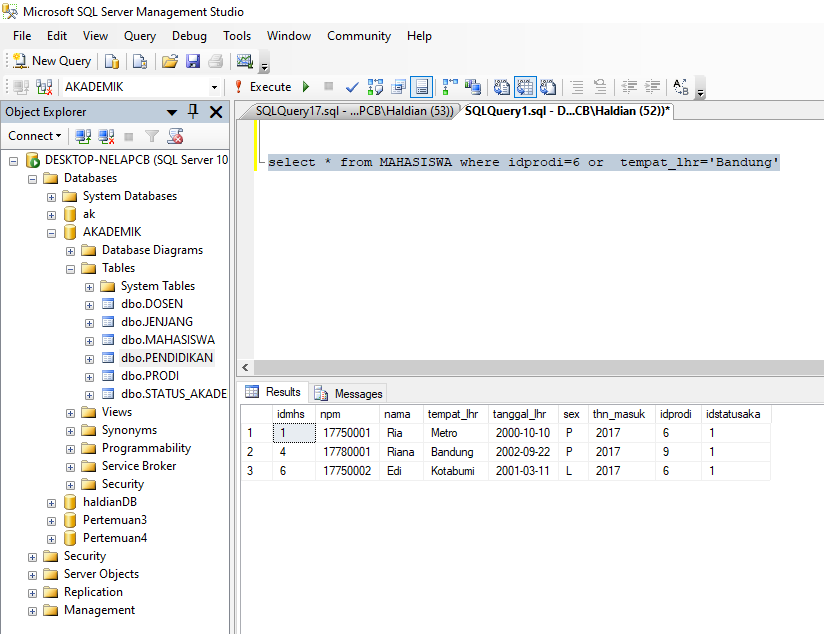
* 1. Menampilkan dosen dari idpendidikan 2 and lahir di bandung seperti gambar di bawah ini

****

* 1. Menampilkan data prodi dari idjenjang=1 dan akreditasi B seperti gambar di bawah ini

****

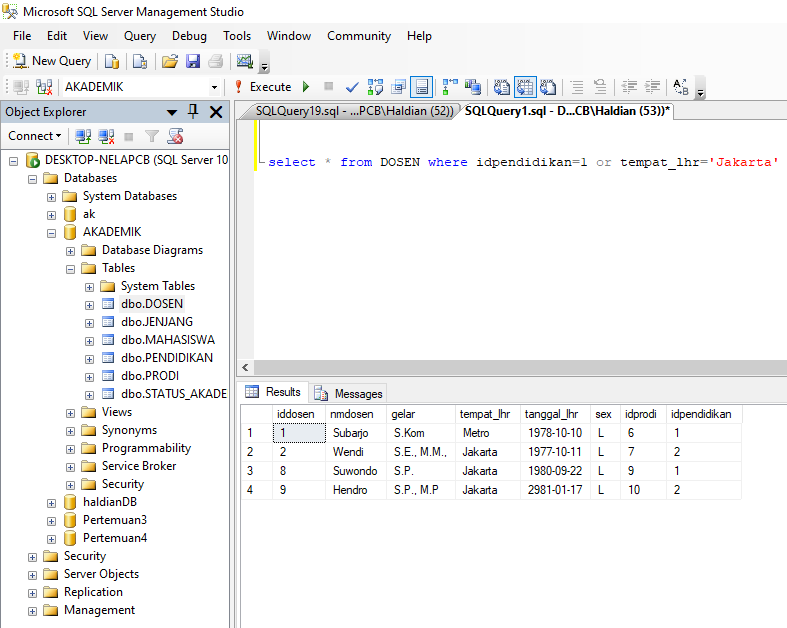
1. ***OR***
2. Menampilkan data mahasiswa idprodi=6 or tempat\_lhr=’Bandung’ seperti gambar di bawah ini

****

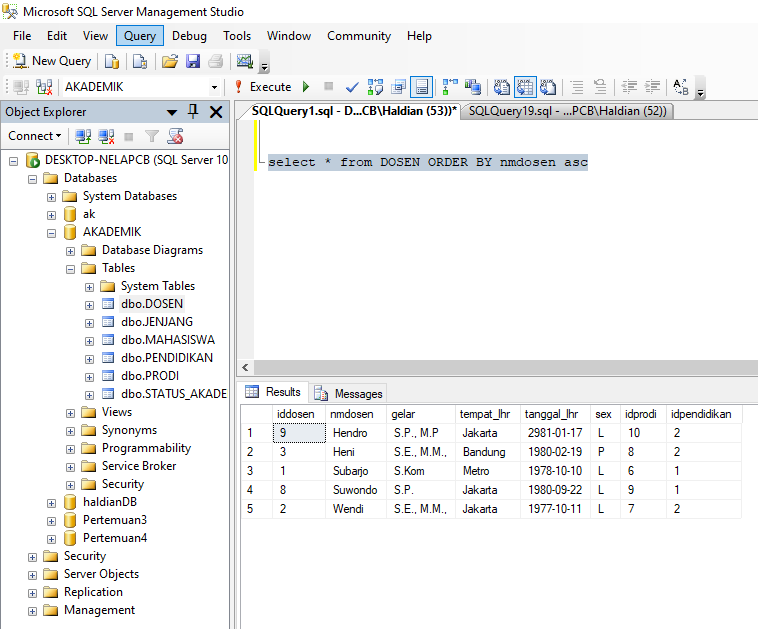
1. Menampilkan prodi idprod=6 or idjenjang=1 seperti gambar di bawah ini

****

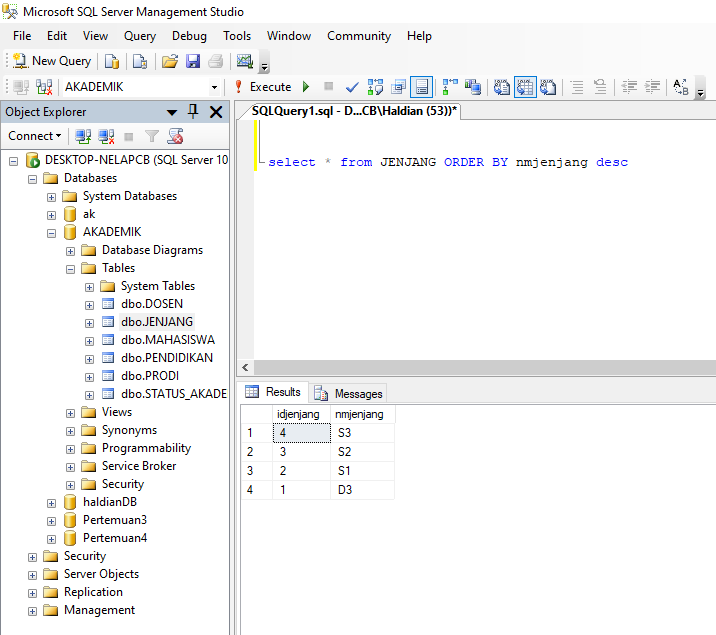
1. Menampilkan data dosen dari idpendidikan=1 or tempat\_lhr=’jakarta’seperti gambar di bawah ini

****

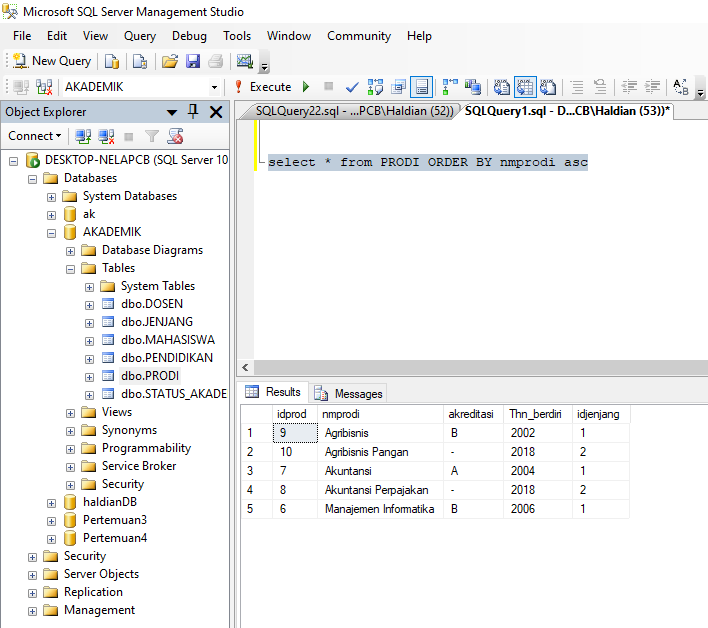
1. **ORDER BY**
2. Menampilkan data dosen mengurutkan nama dosen hasil dalam urutan asc dari atas ke bawah seperti gambar di bawah ini

****

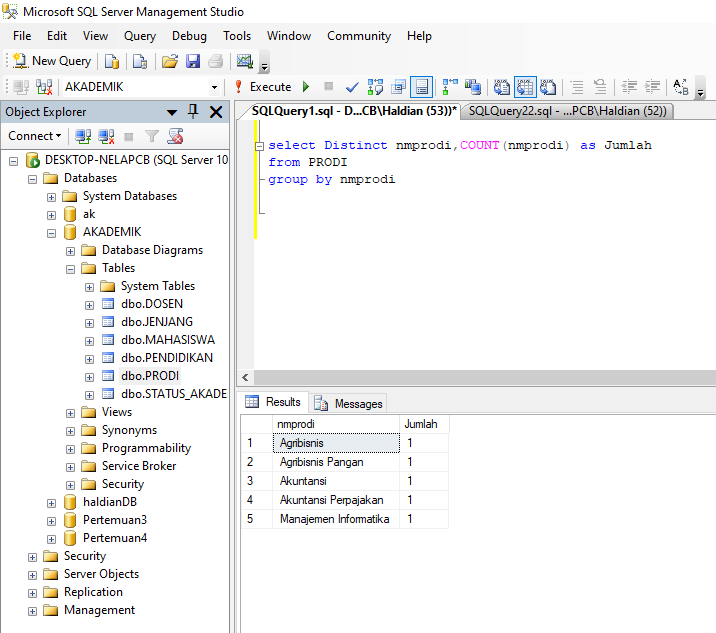
1. Menampilkan nmjenjang dari bawah ke atas dengan desc seperti gambar di bawah ini

****

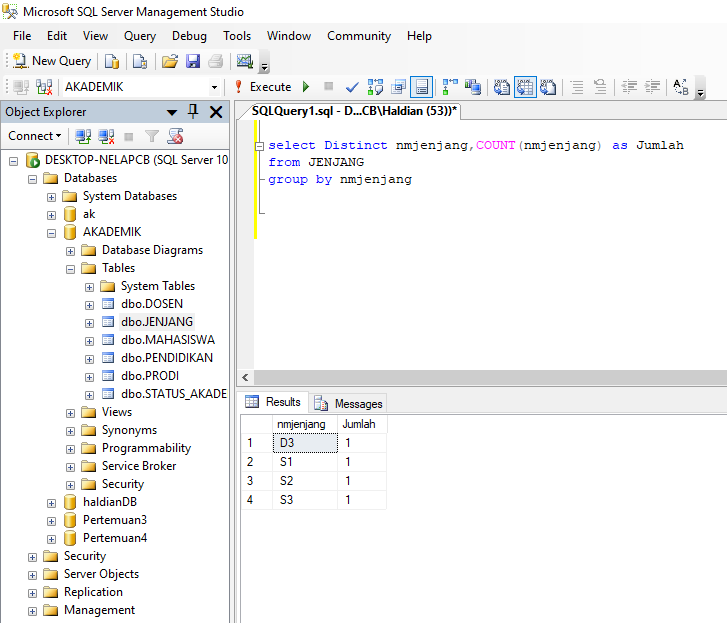
1. Menampilkan prodi dari bawah ke atas dengan asc seperti gambar di bawah ini

****

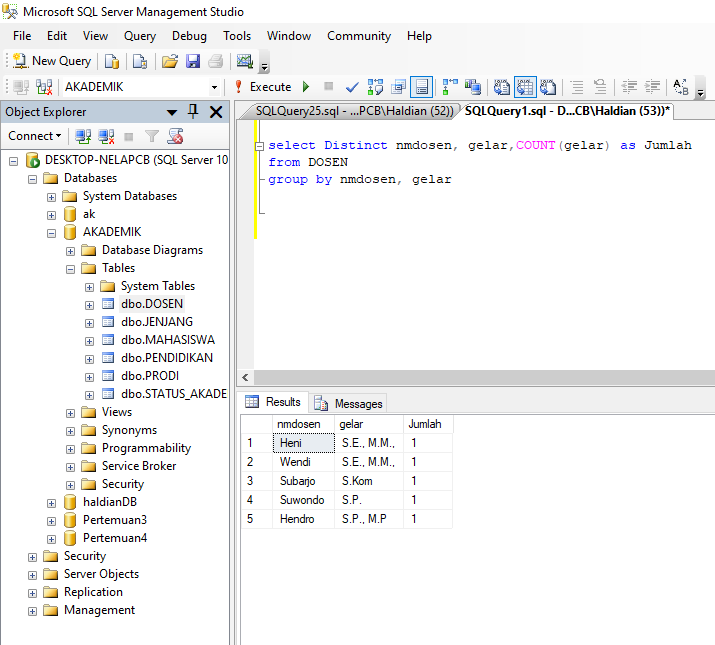
1. ***GROUP BY***
2. Menampilkan nmprodi dengan group by seperti pada gambar di bawa ini

****

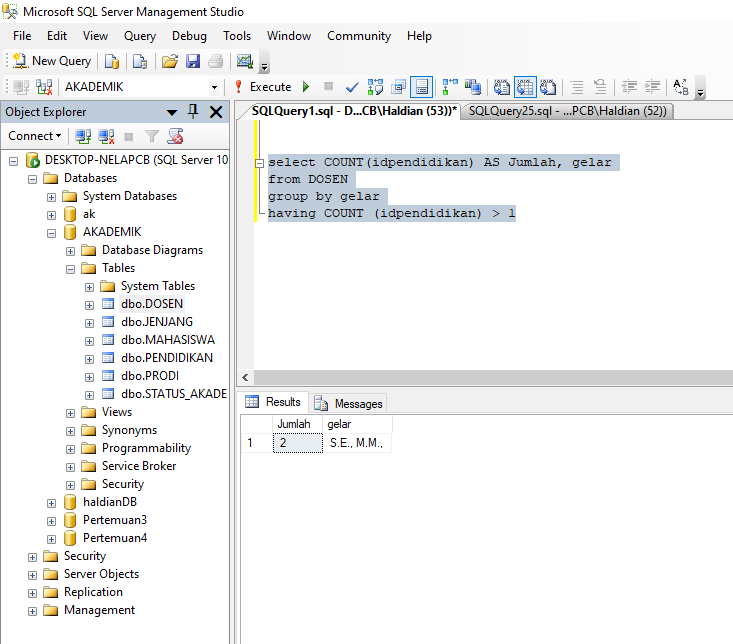
1. Menampilkan nmjenjang pada table jenjang dengan group by seperti gambar di bawah ini

****

1. Menampilkan nama dosen dan gelar dosen pada table dosen dengan group by seperti gambar di bawah ini

****

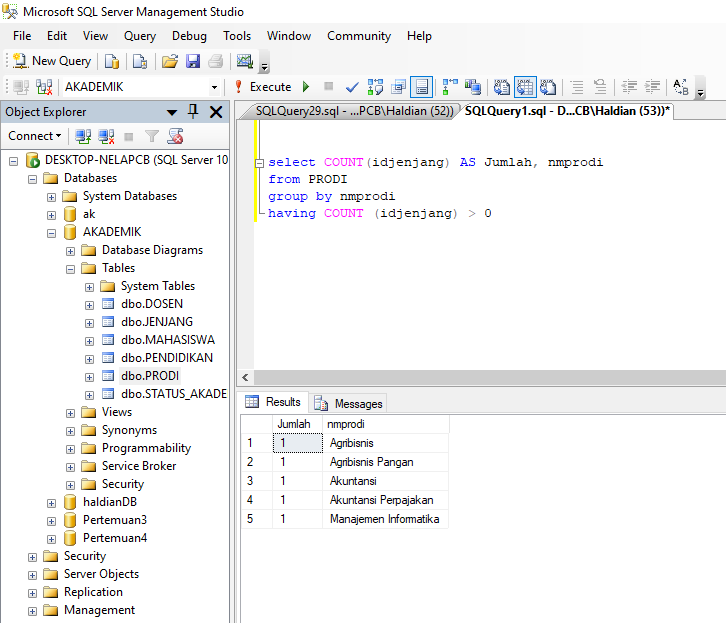
1. ***HAVING***
2. Menampilkan jumlah dosen menggunakan having seperti gambar di bawah ini

****

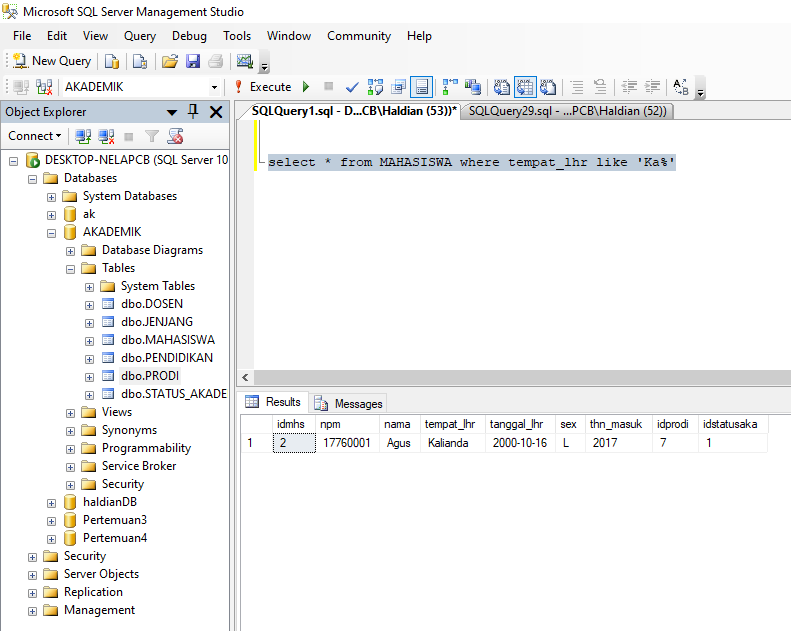
1. Menampilkan table mahasiswa dengan having count idstatusaka > 1 seperti gambar di bawah ini

****

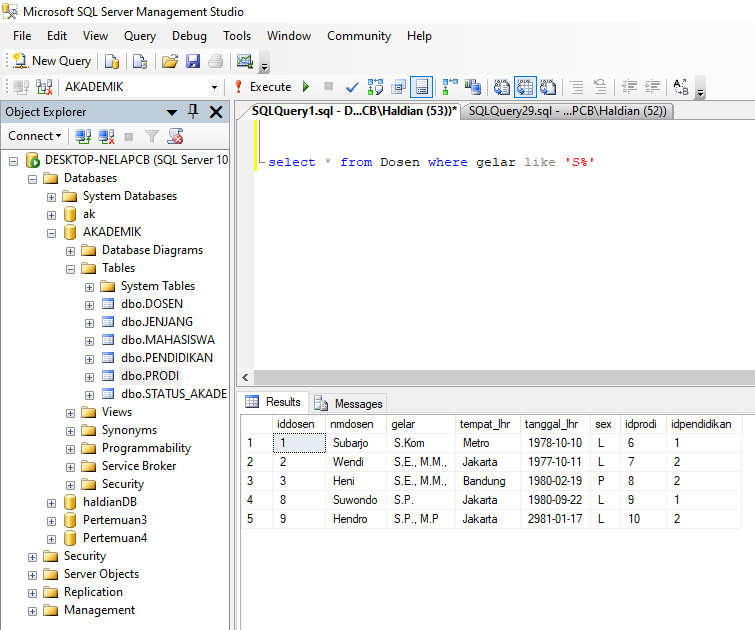
1. Menampilkan nmprodi yang lebih dari 0 dengan having count idjenjang seperti gambar di bawah ini

****

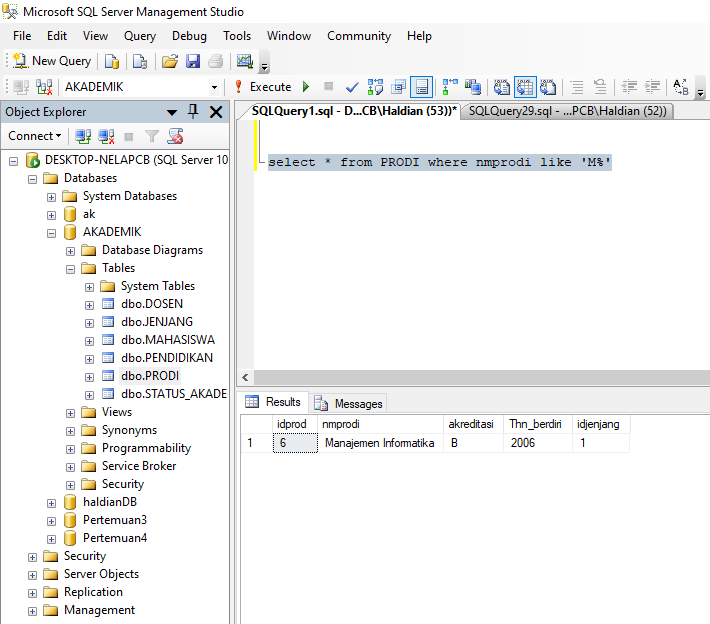
1. **LIKE**
2. Menampilkan data mahasiswa yang di muali dengan Like Ka% seperti gambar di bawah ini

****

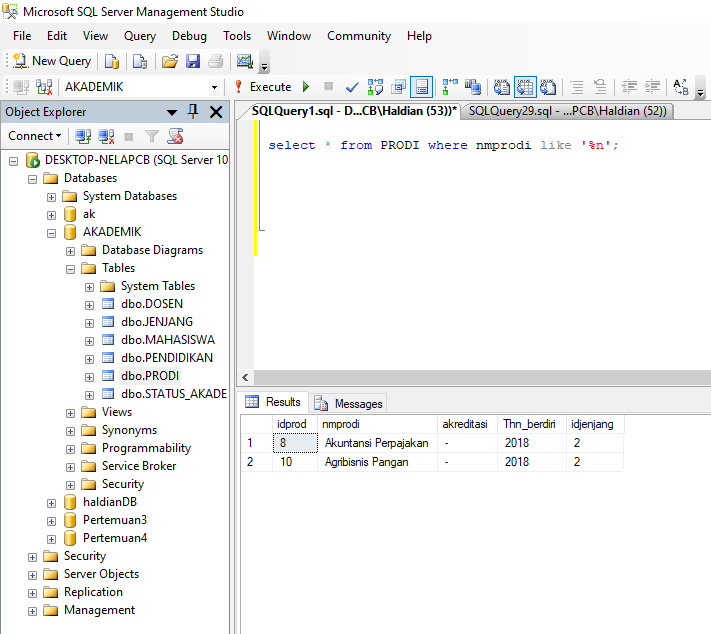
1. Menampilkan data dosen yang di mulai dengan like gelar s% seperti gambar di bawah ini

****

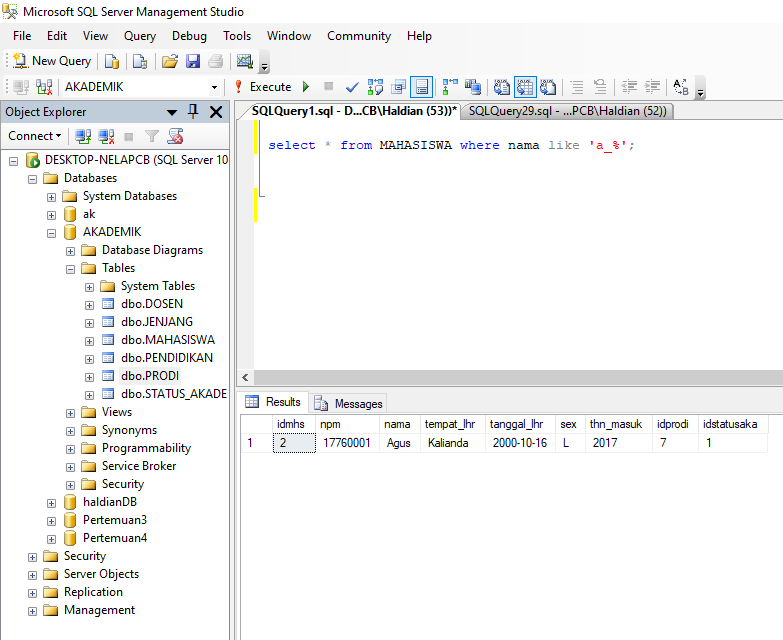
1. Menampilkan data prodi yang di mulai dengan like nmprodi m% seperti gambar di bawah ini

****

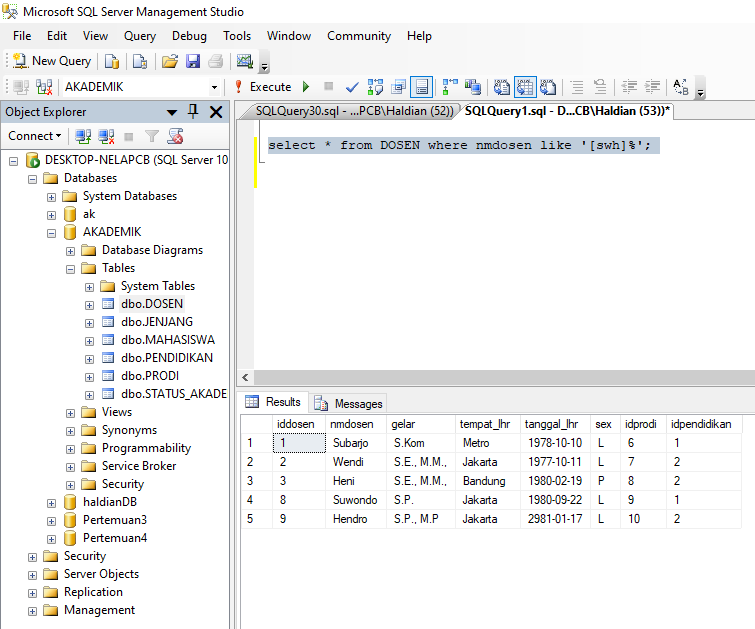
1. ***WILRCARD***
2. Menampilkan data prodi yang di akhiri huruf n seperti gambar di bawah ini

****

1. Menampilkan data mahasiswa yang di awali huruf a seperti gambar di bawah ini

****

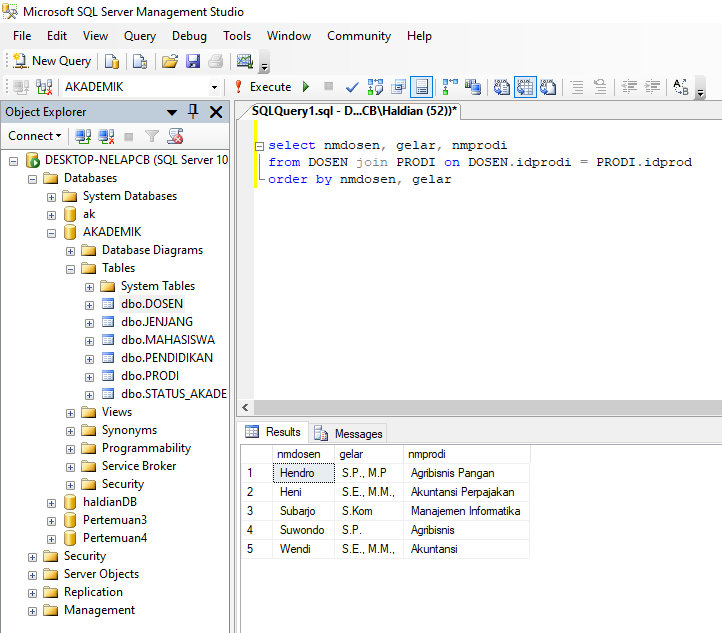
1. Menampilkan data yang memilih semua data ‘s’,’w’,’h’ seperti gambar di bawah ini

****

**2. Minggu #6**

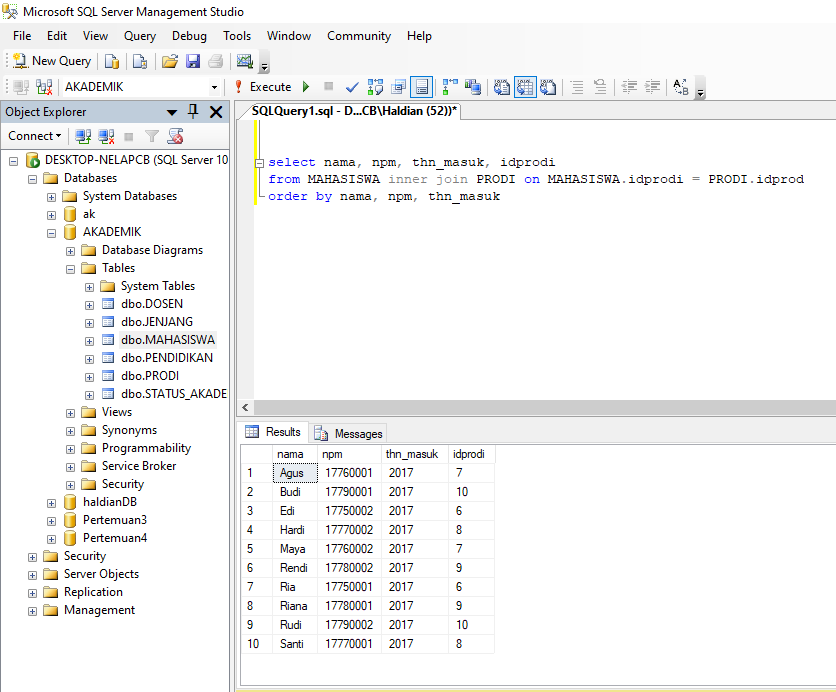
1. Persiapkan alat dan bahan
2. Menjalankan aplikasi: *Microsoft SQL Server Management Studio* (MSQLS-MS)
3. Gunakanlah perintah *transact-SQL Query* (*SELECT*) untuk menyajikan data dari tabel-tabel yang ada pada *database* “AKADEMIK”, dengan memanfaatkan klausa pendukung berikut: *JOIN, INNER JOIN, LEFT JOIN, FULL JOIN, UNION, TOP, IN, BETWEN,* dan *ISNULL*. (catatan: masing-masing klausa, 1 contoh perintah SQL)
4. ***JOIN***

Menampilkan daftar dosen, dan gelarnya beserta nama prodi seperti gambar di bawah ini



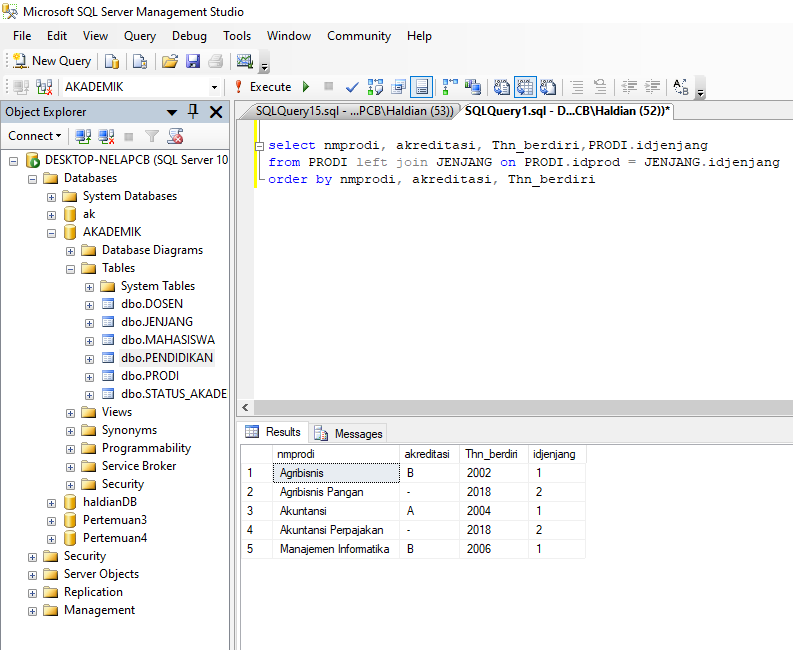
1. ***INNER JOIN***

Menampilkan daftar mahasiswa nama, npm dan tahun masuk serta idprodinya itu sendiri seperti gambar di bawah ini



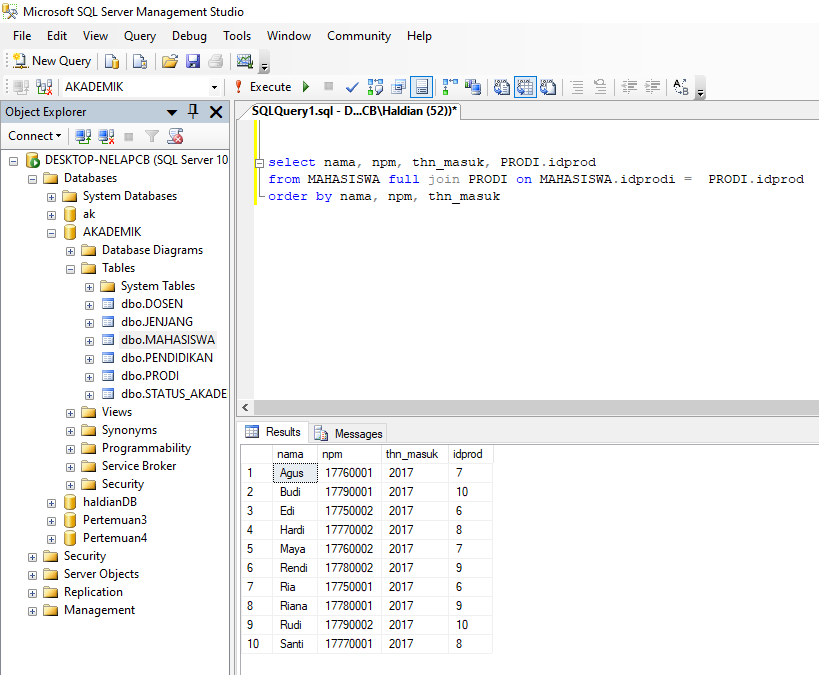
1. ***LEFT JOIN***

Menampilkan nmprodi, akreditasi, thn\_bediri dan idejenjang seperti gambar di bawah ini



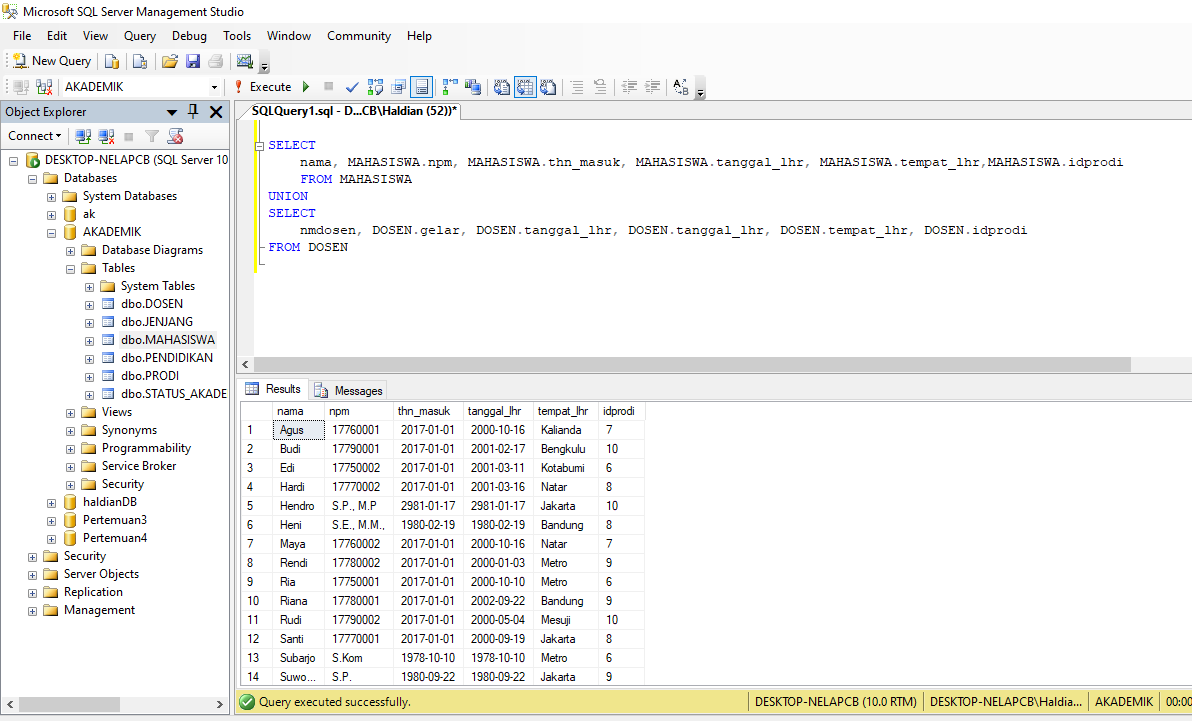
1. ***FULL JOIN***

Menampilkan daftar mahasiswa nama, npm tahum masuk dan idprodi seperi gambar di bawah ini



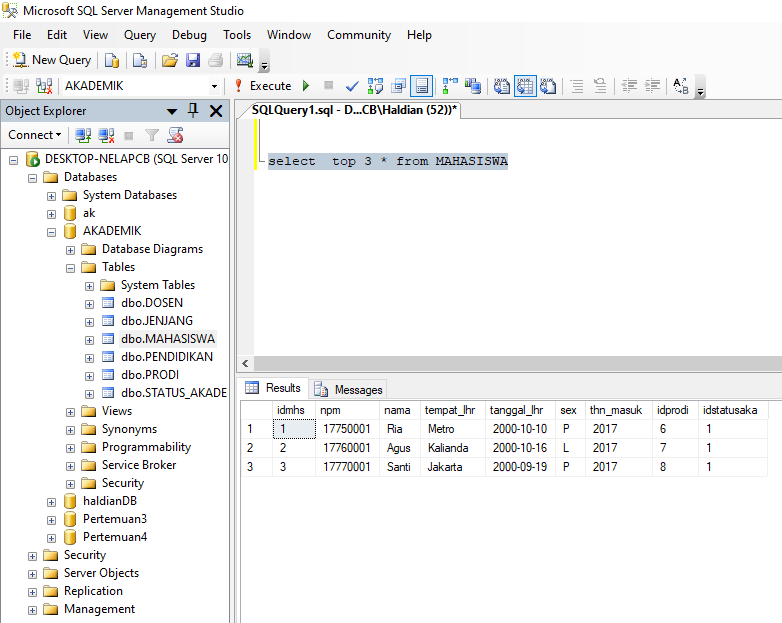
1. ***UNION***

Menampilkan aktivitas akademik dosen dan mahasiswa seperti gambar di bawah ini



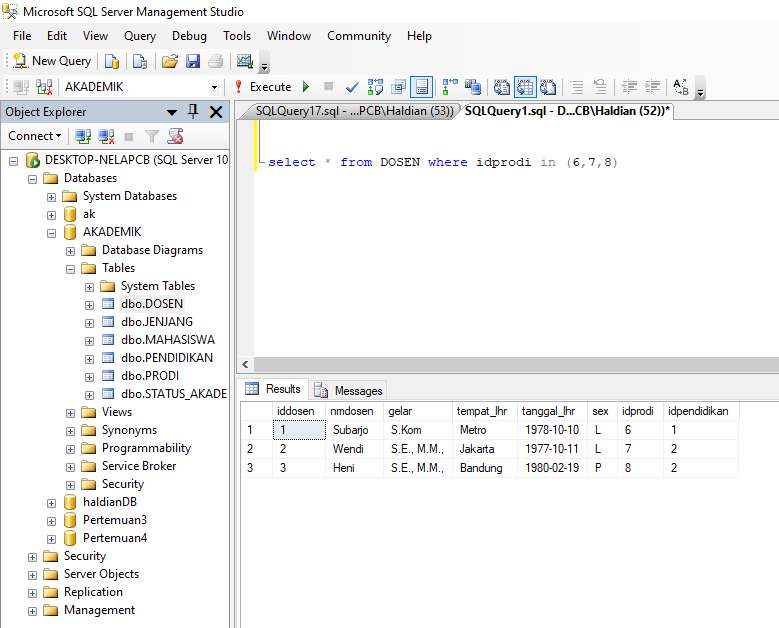
1. ***TOP***

Menampilkan data mahasiswa dari 3 teratas dengan top seperti gambar di bawah ini



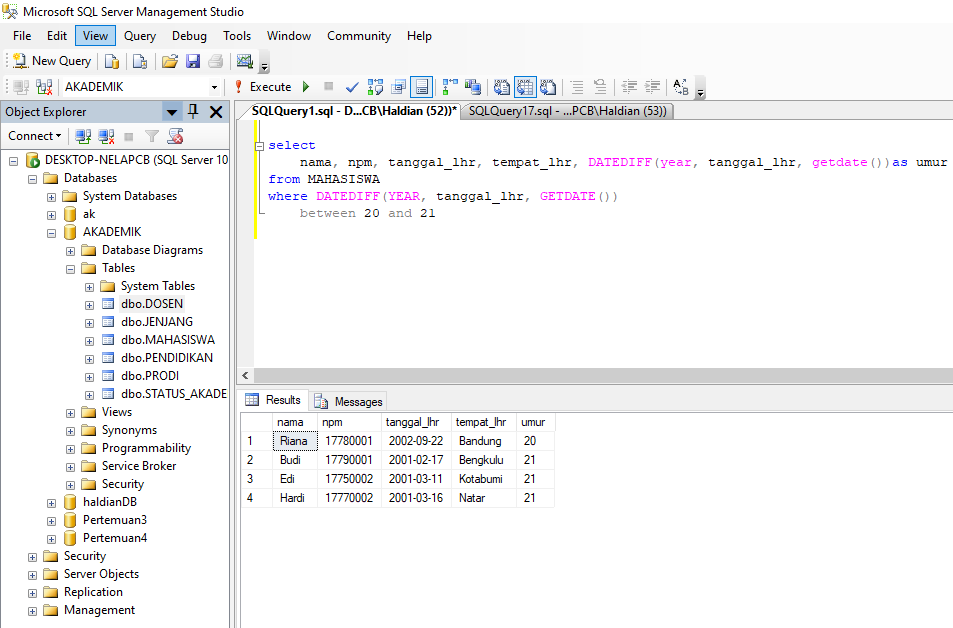
1. ***IN***

Menampilkan data dosen dari idprodi 6.7.8 saja dengan In seperti gambar di bawah ini



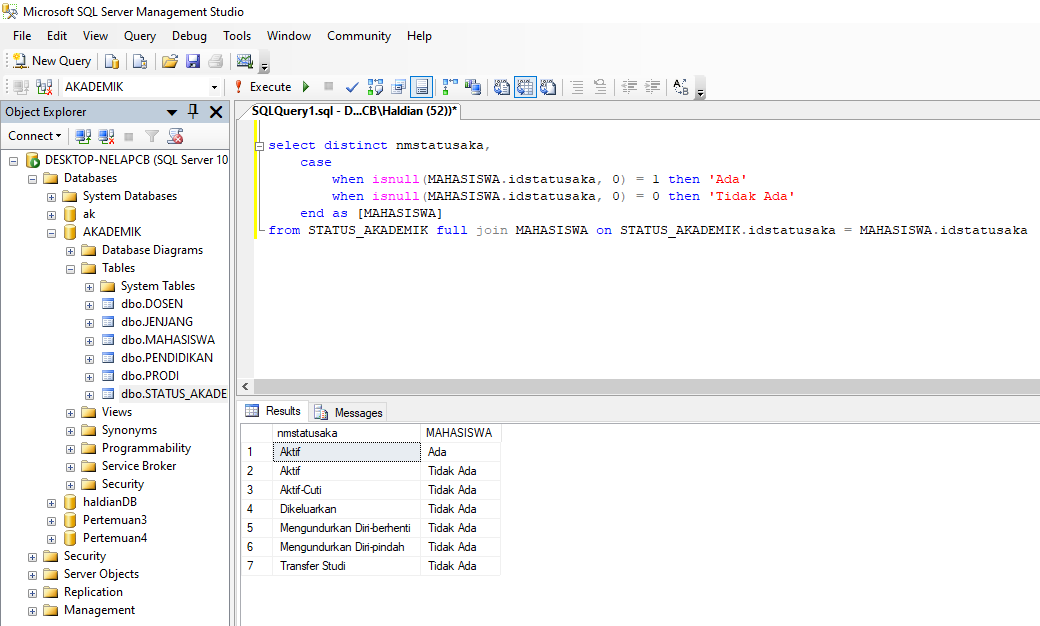
1. ***BETWEEN***

Menampilkan data mahasiwa yang berumur 20 – 21 tahun seperti gambar di bawah ini



1. ***ISNULL***

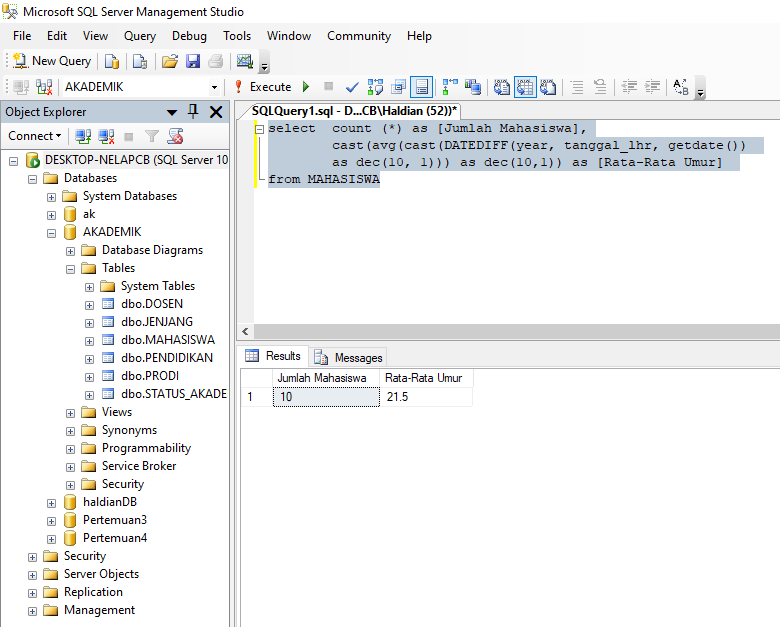
Menampilkan data keterangan status akademik mahasiswa dengan distince case dan full join menghasilkan ‘Ada’ atau ‘Tidak ada’ seperti gambar di bawah ini



**3. Minggu #7**

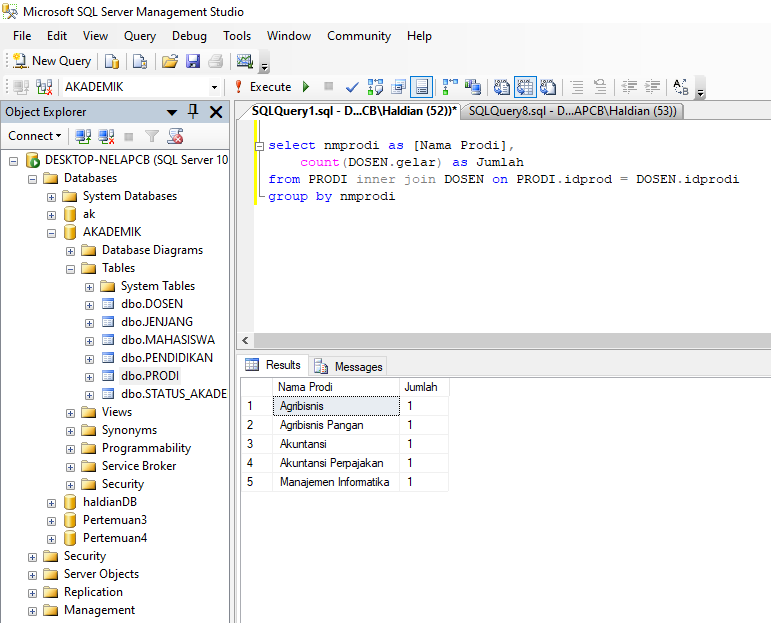
1. Persiapkan alat dan bahan
2. Menjalankan aplikasi: *Microsoft SQL Server Management Studio* (MSQLS-MS)
3. Gunakanlah perintah *transact-SQL Query* (*SELECT*) untuk menyajikan data dari tabel-tabel yang ada pada *database* “AKADEMIK”, dengan memanfaatkan klausa pendukung berikut: *AVG, COUNT, MAX, MIN, SUM, MID, LEN, ROUND, NOW, UCASE, LCASE.* (catatan: masing-masing klausa, 1 contoh perintah SQL)
4. ***AVG()***

Menampilkan jumlah mahasiwa dan rata-rata umr dari mahasiswa seperti gambar di bawah ini



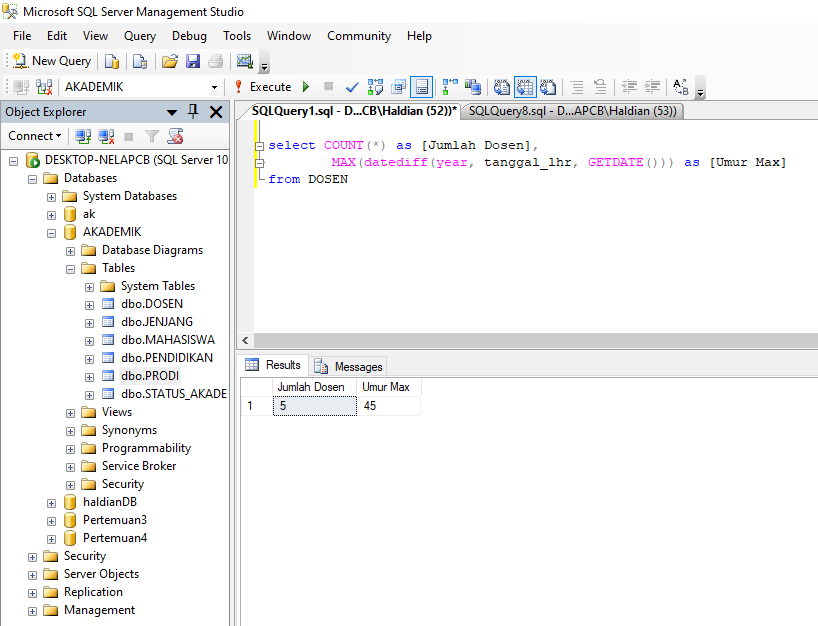
1. ***COUNT()***

Menampilkan jumlah dosen dari masing-masing prodi seperti gambar di bawah ini



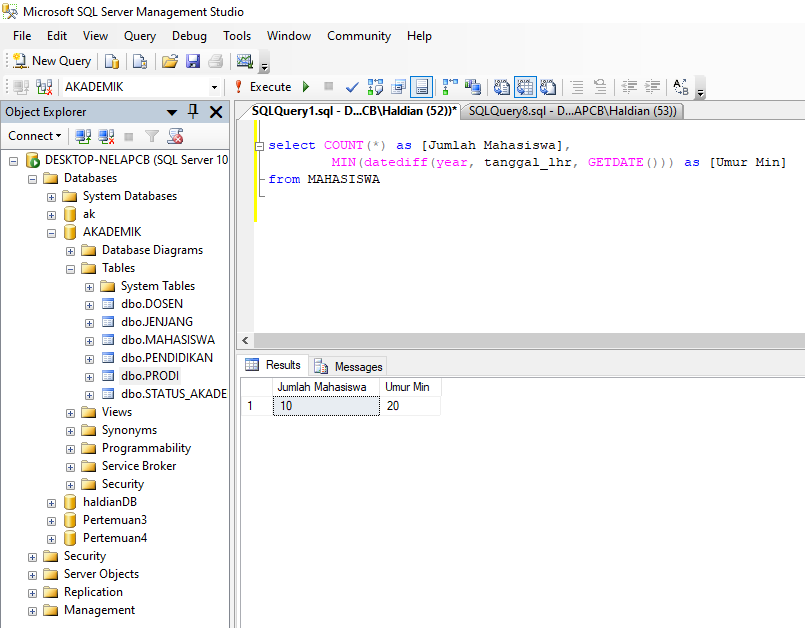
1. ***MAX()***

Menampilkan umur maximal dari dosen seperti gambar di bawah ini



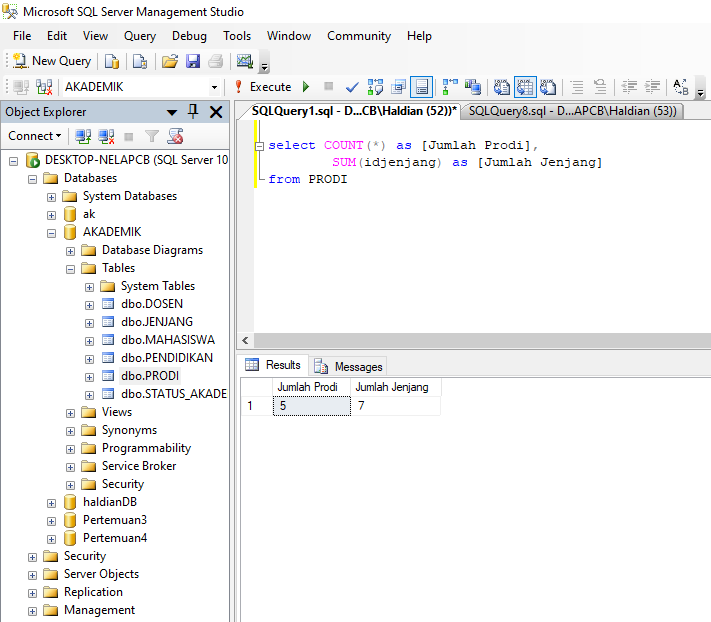
1. ***MIN()***

Menampilkan minimal umur mahasiswa seperti gambar di bawah ini



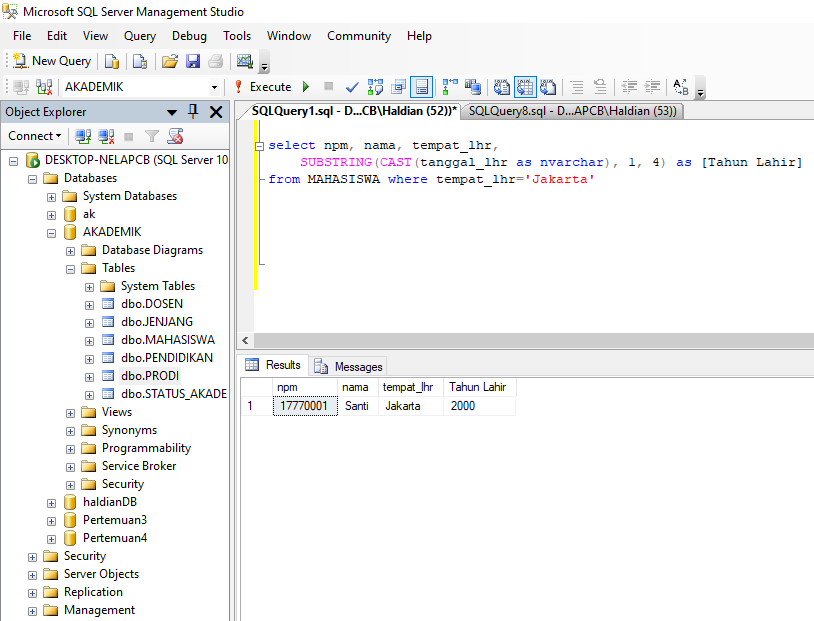
1. ***SUM()***

Menampilkan jumlah prodi dan jumlah jenjang dengan sum seperti gambar di bawah ini



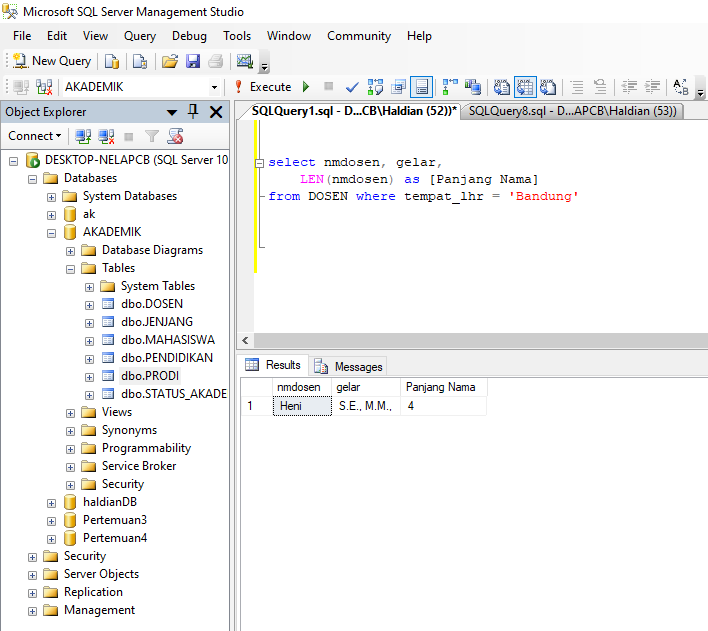
1. ***MID/SUBSTRING***

Menampilkan data mahasiwa yang bertempat lahir di Jakarta dan tahun lahir hanya di ambil tahunnya saja seperti gambar di bawah ini



1. ***LEN***

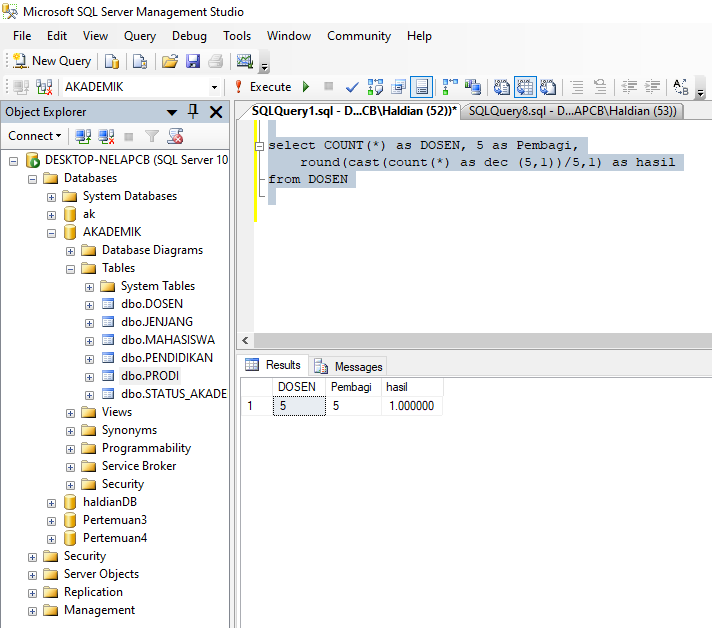
Mengambil data dosen yang lahir di bandung dan mengambil nilai panjang nama dengan LEN seperti gambar di bawah ini



1. ***ROUND()***

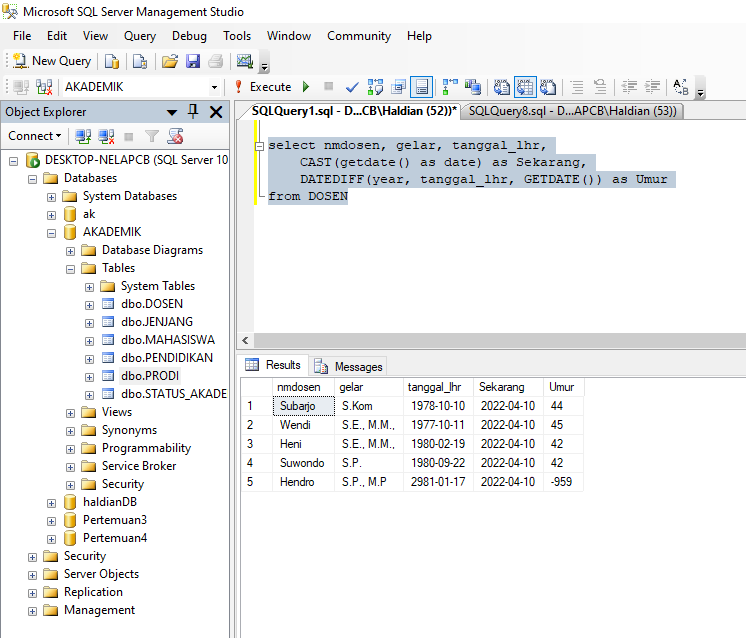
Menampilkan pembulatan dari jumlah DOSEN dengan oprator pembagi

Seperti gambar di bawah ini



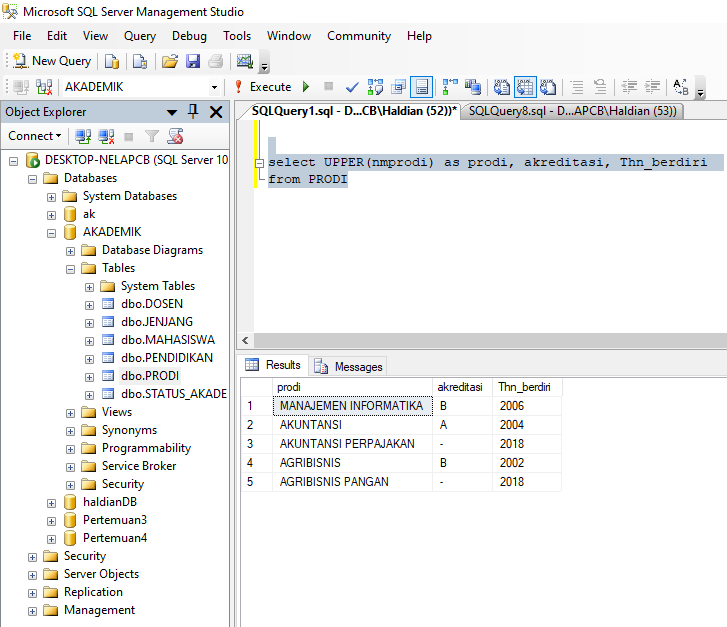
1. ***GETDATE()***

Menampilkan nama dosen, gelar, tanggal\_lhr, sekarang dan umur seperti gambar di bawah ini



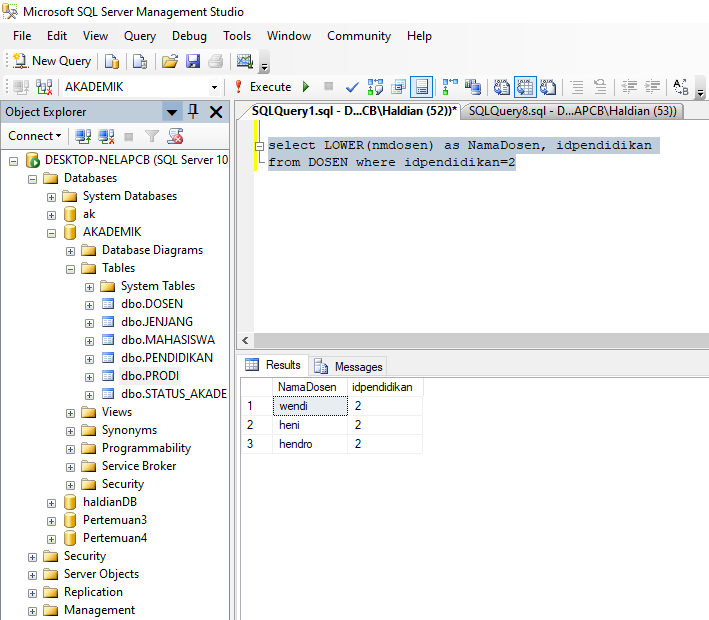
1. ***UPPER()***

Menampilkan nama prodi menjadi besar semua dengan seperti gambar di bawah ini



1. ***LOWER()***

Menampilkan nama dosen menjadi kecil semua seperti gambar di bawah ini

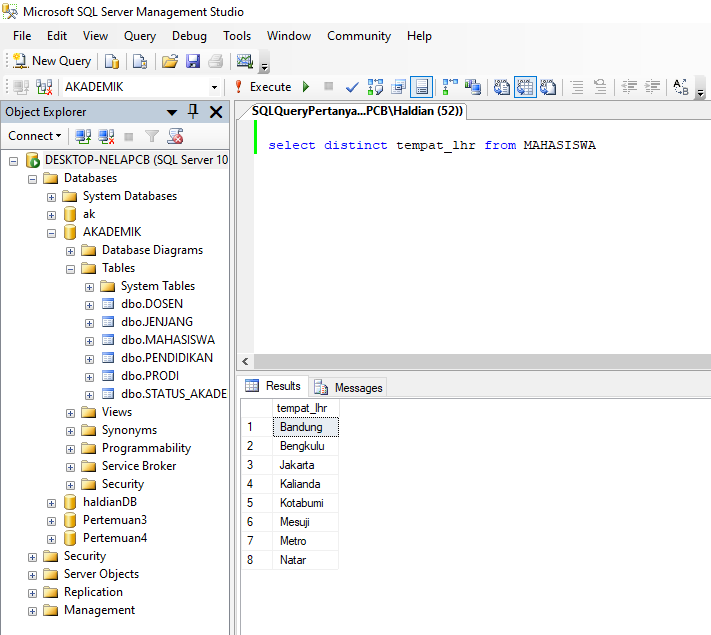


1. **TUGAS DAN PERTANYAAN**

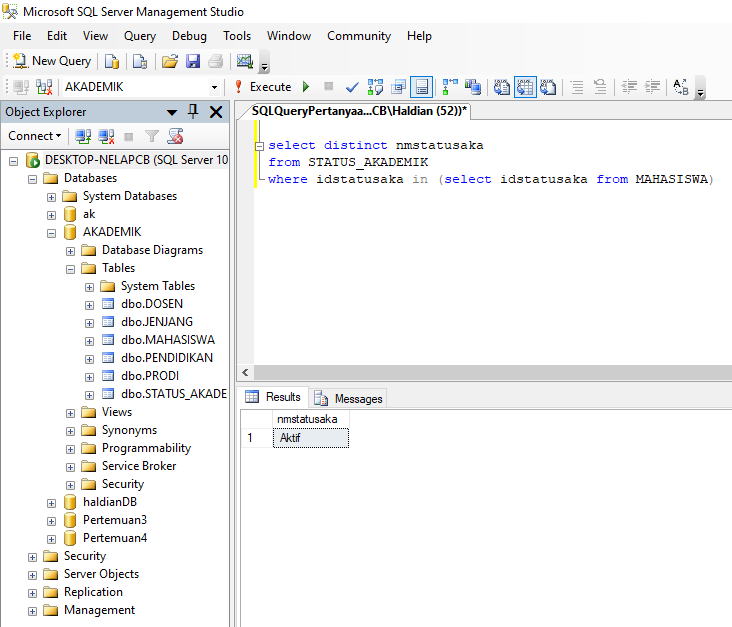
**1. Minggu #5**

1. Buatlah dokumen T-SQL dari praktikum minggu 5 prosedur kerja 3 (Sudah terlampir di prosedur kerja).
2. Menggunakan perintah T-SQL (SELECT), sajikan informasi berikut:

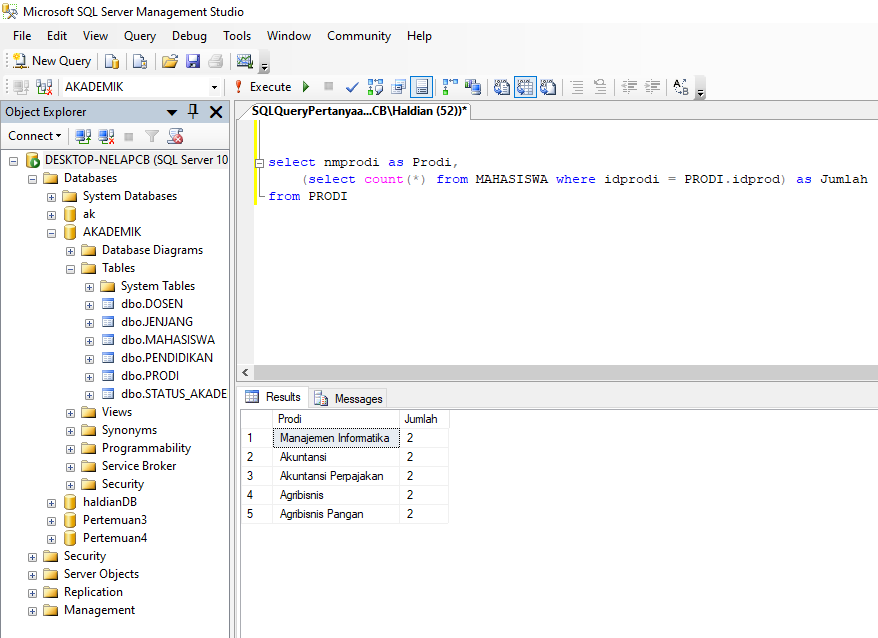
* Data kota tempat lahir mahasiswa



* Data status akademik yang ada pada mahasiswa terdaftar.



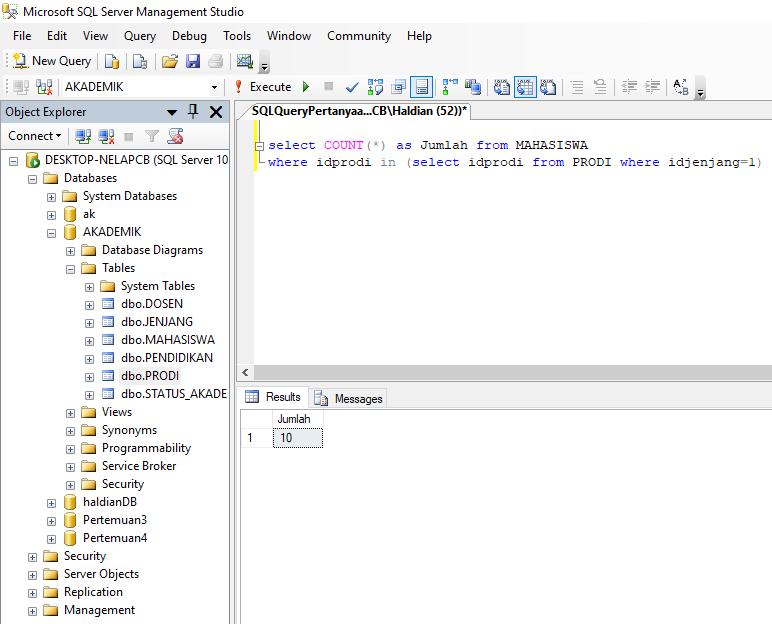
* Data Jumlah mahasiswa berdasarkan program studi



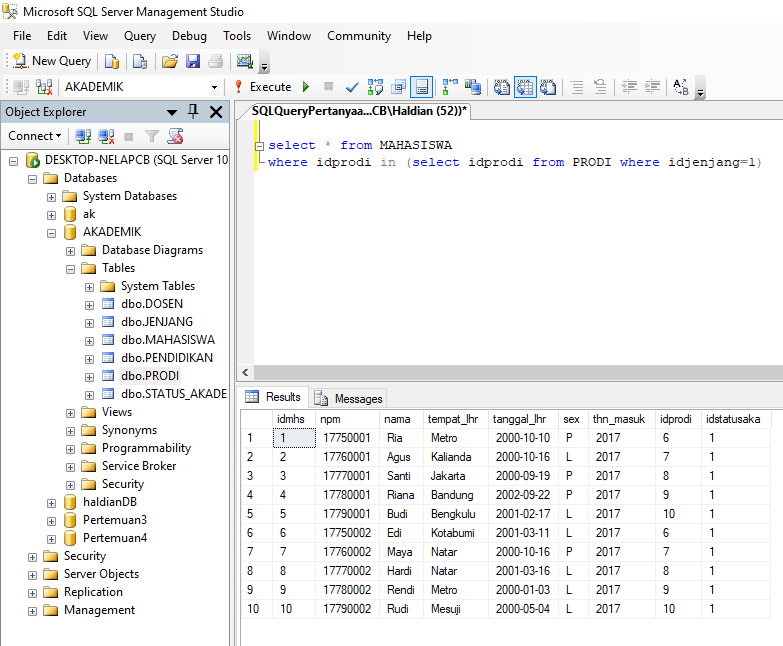
* Data Jumlah dosen berdasarkan program studi



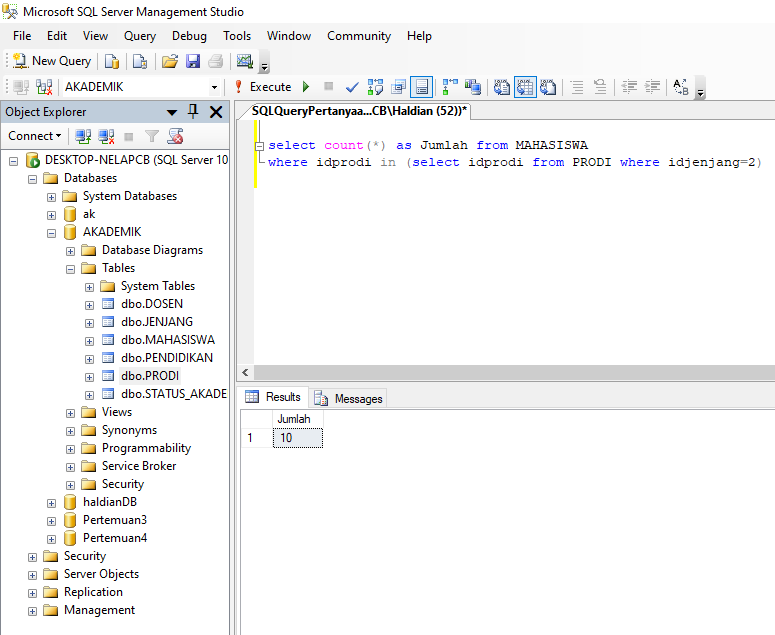
* Data jumlah mahasiswa jenjang diploma



* Data mahasiswa jenjang diploma



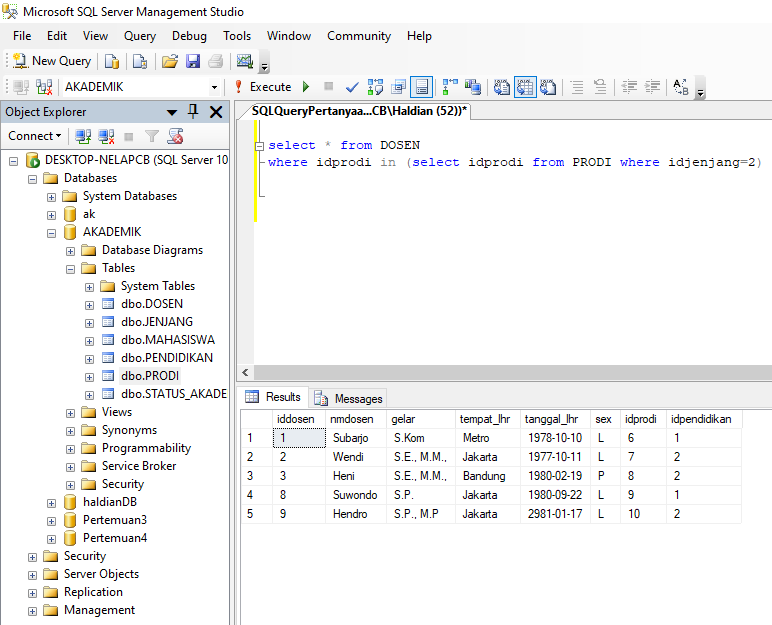
* Data jumlah mahasiswa jenjang sarjana terapan



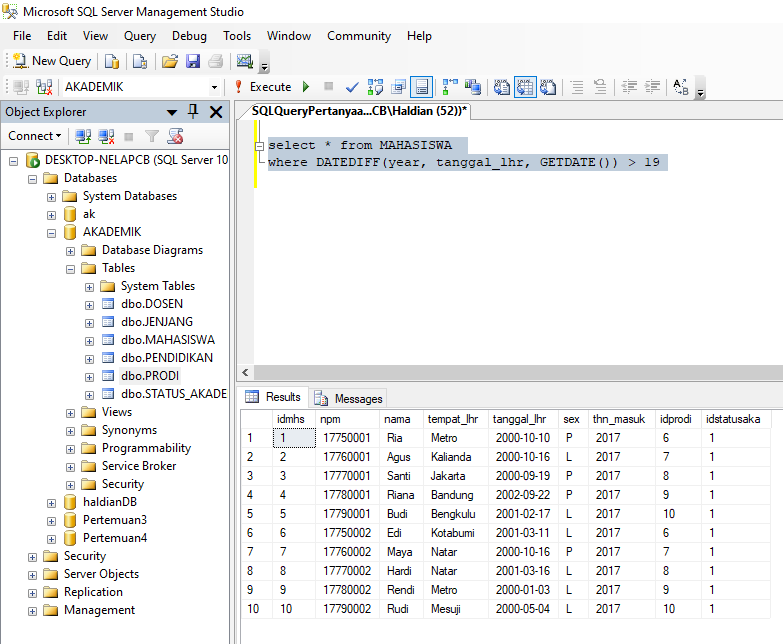
* Data dosen jenjang diploma



* Data dosen jenjang sarjana terapan



* Data mahasiswa yang umurnya saat ini lebih dari 19 tahun



**2. Minggu #6**

1. Buatlah dokumen T-SQL dari praktikum minggu 6 prosedur kerja 3. (Sudah terlampir di prosedur kerja).
2. Menggunakan perintah T-SQL (SELECT), kerjakan tugas berikut:

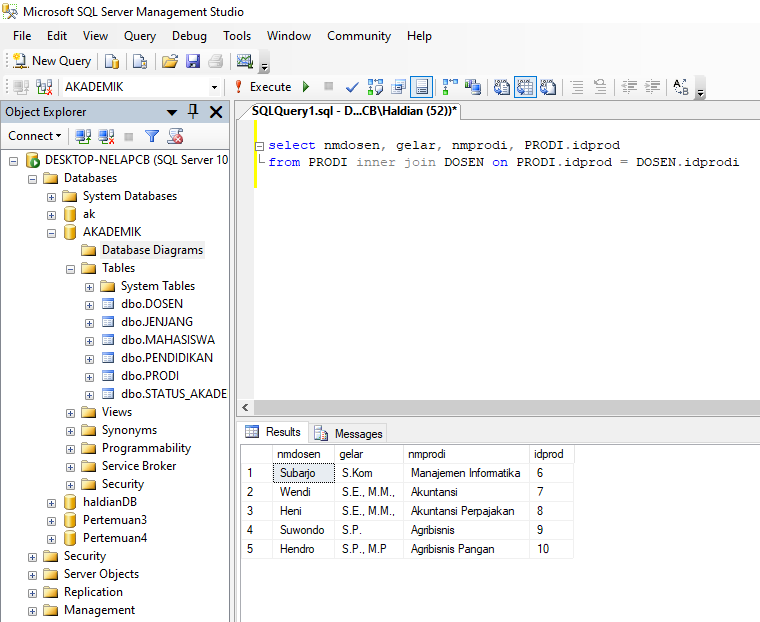
* Jika klausa atau kondisi INNER JOIN diimplementasikan pada tabel-tabel yang ada pada database “AKADEMIK”. Berapa kombinasi relasi 2 tabel yang dapat dibuat, Data apa saja yang dapat disajikan, dan bagaimana perintah SQL yang ditulis untuk penyajian datadata tersebut.
* Kombinasi relasi 2 tabel yang dapat dibuat yaitu: PRODI dengan

JENJANG, PRODI dengan MAHASISWA, PRODI dengan DOSEN,

MAHASISWA dengan STATUS\_AKADEMIK, dan DOSEN dengan PENDIDIKAN.

* Data yang dapat di sajikan yaitu menggabungkan baris dari dua atau lebih tabel, berdasarkan kolom terkait di antara mereka. Serta mengembalikan rekaman yang memiliki nilai yang cocok di kedua tabel
* Contoh penulisan perintah SQL:

Menampilkan daftar DOSEN beserta nama PRODI-nya

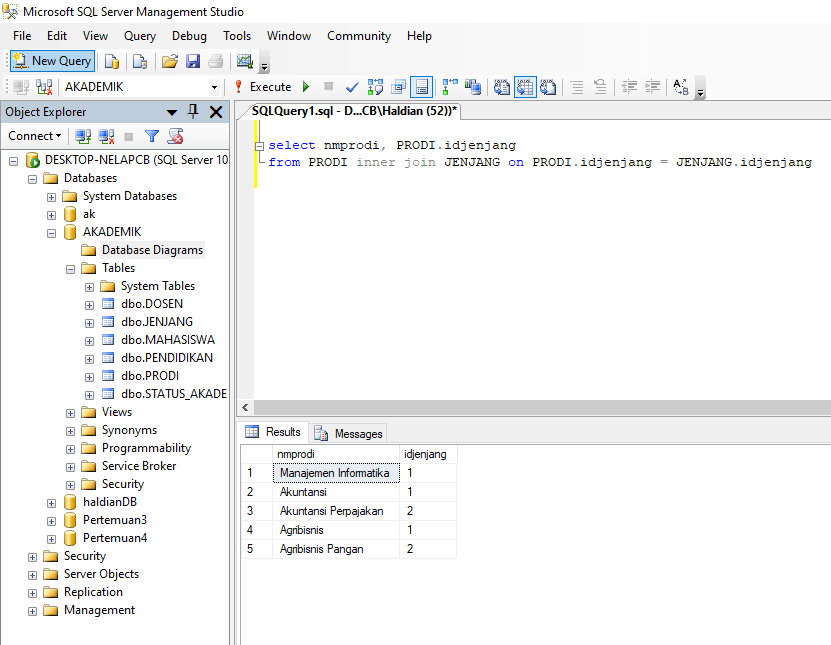


* Implementasikan klausa atau kondisi LEFT JOIN pada 2 tabel yang ada pada database “AKADEMIK”. Tuliskan data apa yang disajikan, dan bagaimana perintah SQL yang ditulis untuk penyajian data tersebut.
* Mengembalikan semua record dari tabel kiri (tabel1), dan record yang cocok dari tabel kanan (tabel2). Hasilnya adalah 0 record dari sisi kanan, jika tidak ada yang cocok.

Menampilkan data. Pada contoh disini saya mengambil data Prodi dan Jenjang

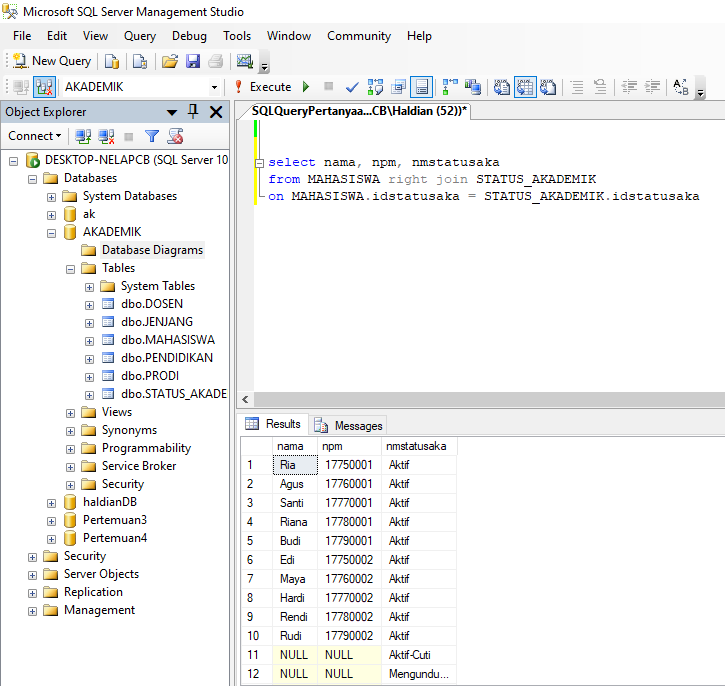
* Contoh penulisan perintah SQL:

Menampilkan daftar PRODI dan JENJANG



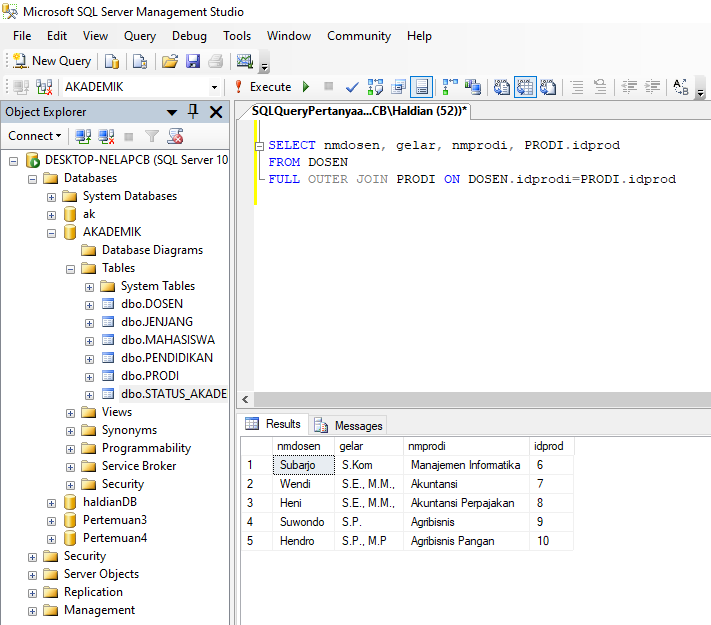
* Implementasikan klausa atau kondisi RIGHT JOIN pada 2 tabel yang ada pada *database* “AKADEMIK”. Tuliskan data apa yang disajikan, dan bagaimana perintah SQL yang ditulis untuk penyajian data tersebut.
* mengembalikan semua record dari tabel kanan (table2), dan record yang cocok dari tabel kiri (table1). Hasilnya adalah 0 record dari sisi kiri, jika tidak ada yang cocok.
* Contoh penulisan perintah SQL:

Menampilkan MAHASISWA dengan STATUS\_AKADEMIK



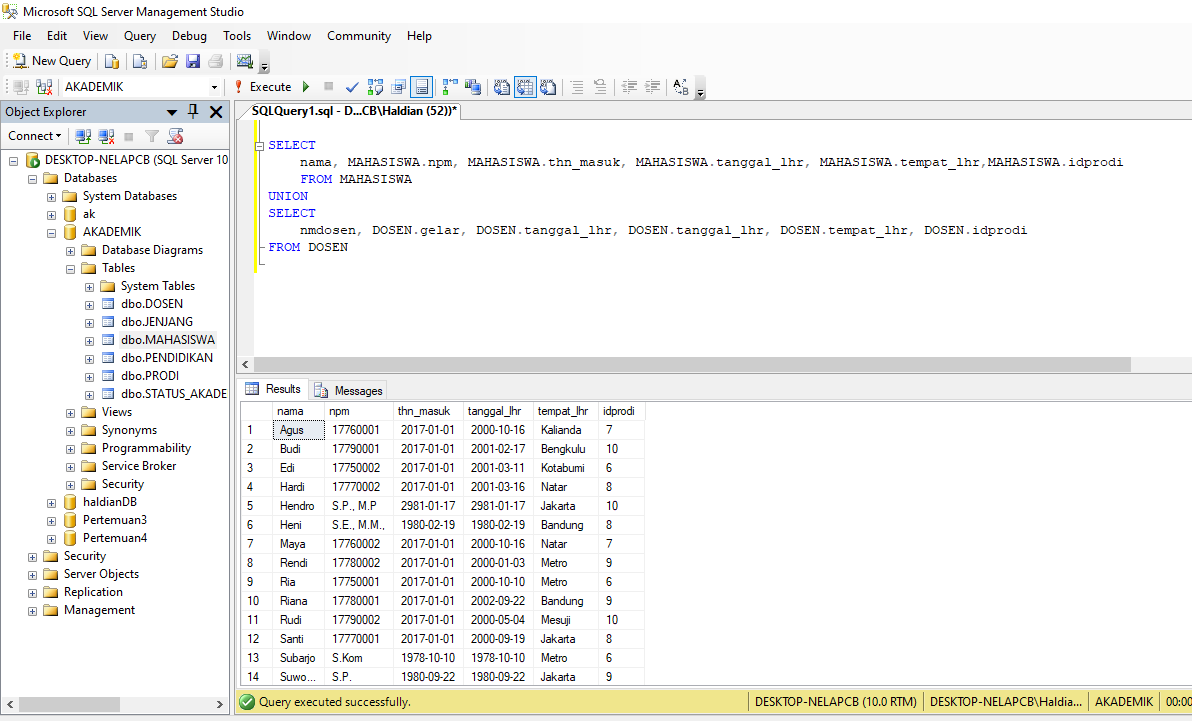
* Implementasikan klausa atau kondisi RIGHT JOIN pada 2 tabel yang ada pada database “AKADEMIK”. Tuliskan data apa yang disajikan, dan bagaimana perintah SQL yang ditulis untuk penyajian data tersebut.
* Mengembalikan semua record ketika ada kecocokan dalam record tabel kiri (tabel1) atau kanan (tabel2).
* Contoh penulisan perintah SQL:

Menampilkan DOSEN dengan PRODI



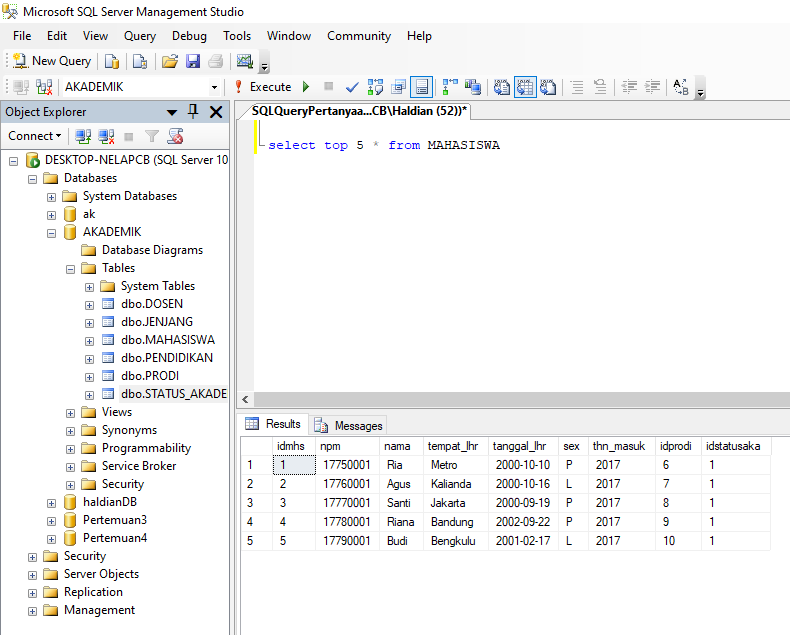
* Implementasikan klausa atau kondisi UNION pada 2 tabel yang ada pada database “AKADEMIK”. tuliskan data apa saja yang disajikan, dan bagaimana perintah SQL yang ditulis untuk penyajian data tersebut
* Menggabungkan kumpulan hasil dari dua atau lebih pernyataan SELECT. Setiap pernyataan SELECT dan harus memiliki jumlah kolom yang sama, Kolom juga harus memiliki tipe data yang serupa, Kolom di setiap pernyataan SELECT juga harus dalam urutan yang sama.
* Contoh penulisan perintah SQL:

Menampilkan seluruh civitas akademika

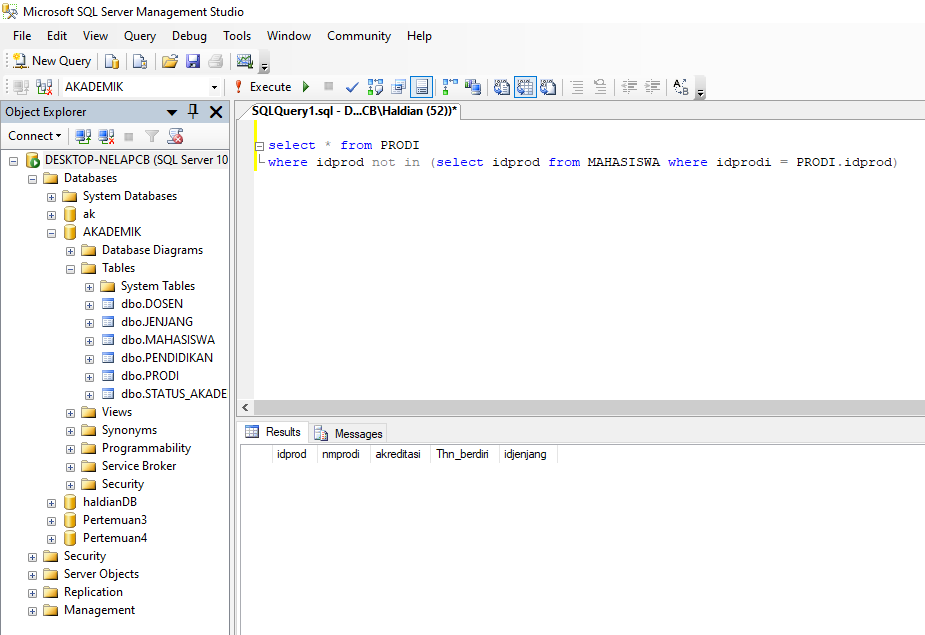


1. Menggunakan perintah T-SQL (SELECT), Sajikan informasi berikut:

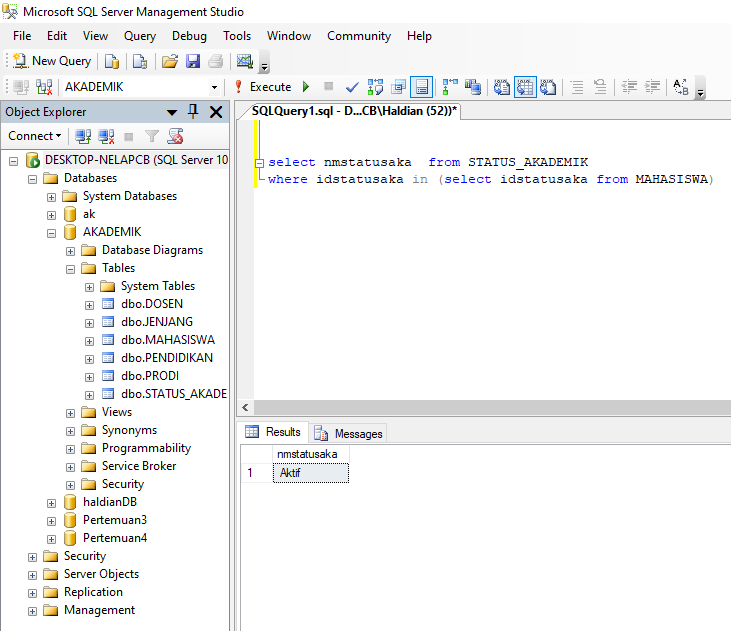
* 5 Data mahasiswa dari seluruh program studi urut berdasarkan nama



* Menampilkan program studi yang tidak memiliki mahasiswa



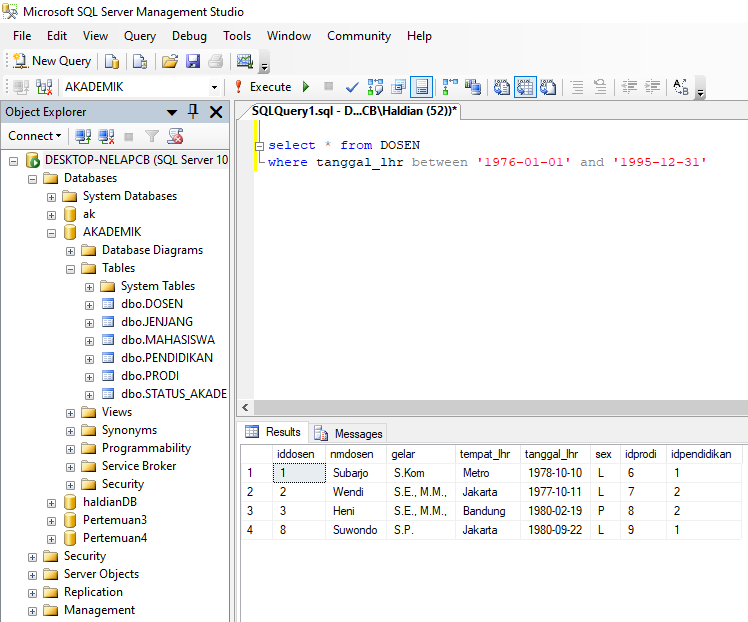
* Menampilkan status akademik yang dimiliki mahasiswa



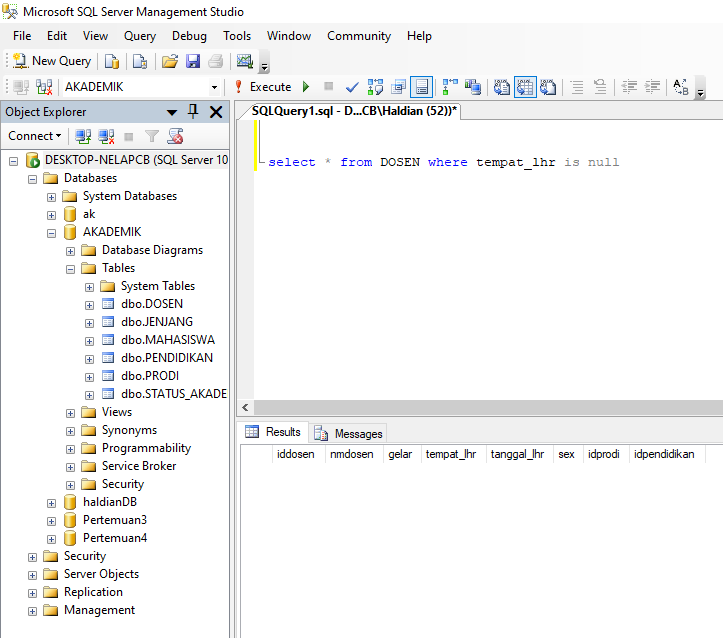
* Menampilkan mahasiswa yang terdaftar pada tahun 2015 s.d. 2017



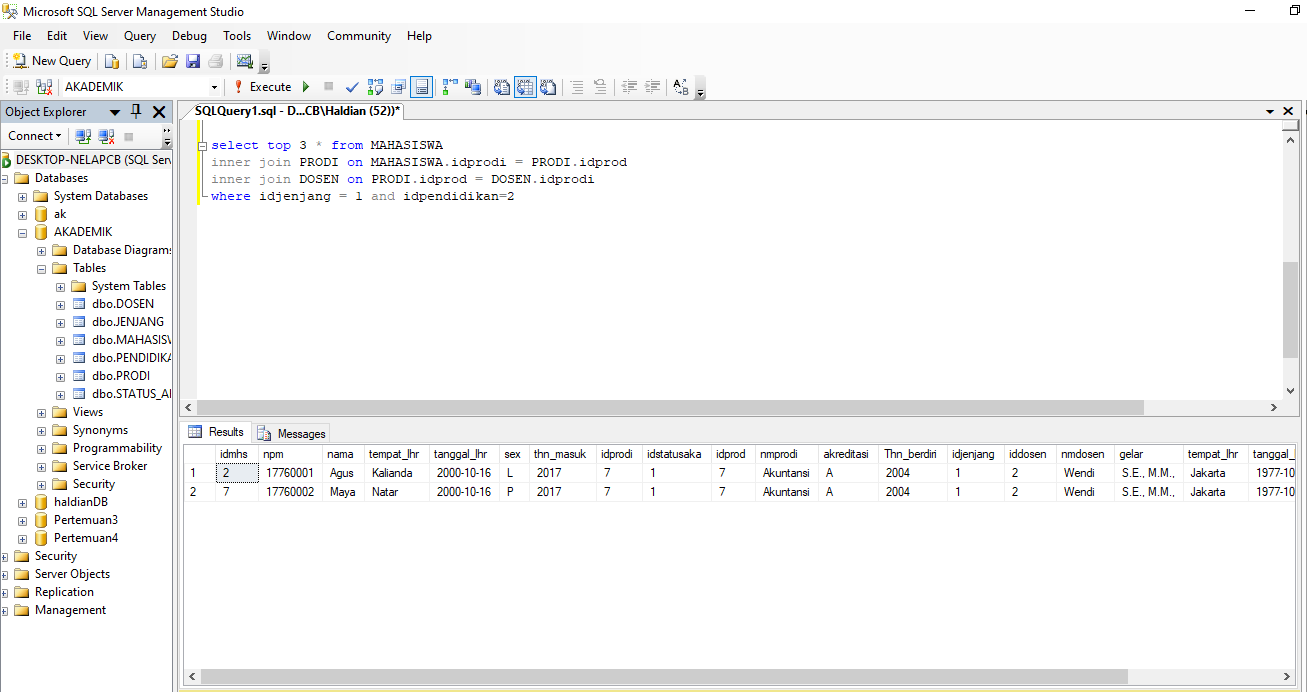
* Menampilkan Dosen yang lahir dari tanggal 01-01-1976 s.d. 31-12-1995



* Menampilkan dosen yang data tempat lahir belum terisi



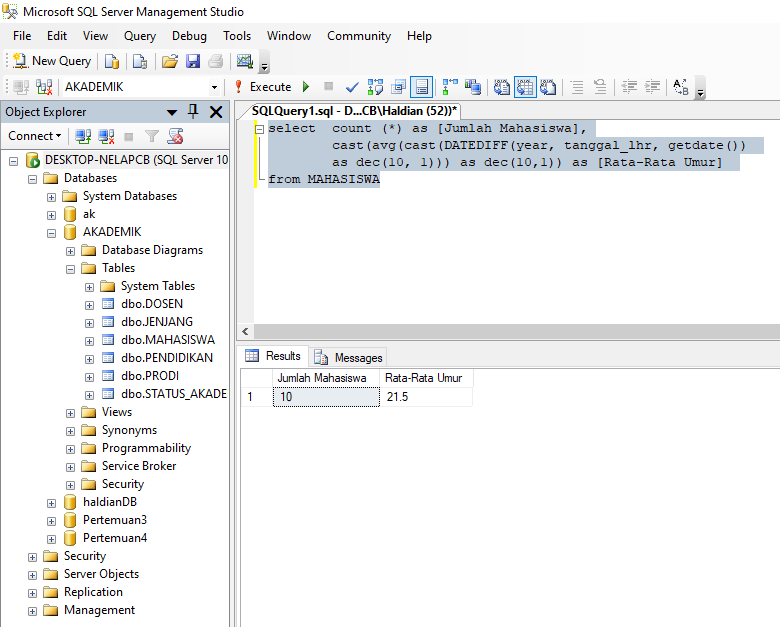
* Menampilkan 3 mahasiswa yang program studinya berjenjang D3 dan dosennya berjenjang S2



**3. Minggu #7**

1. Buatlah dokumen T-SQL dari praktikum minggu 7 prosedur kerja 3. (Sudah terlampir di prosedur kerja).
2. Menggunakan perintah T-SQL (SELECT), Sajikan informasi berikut:

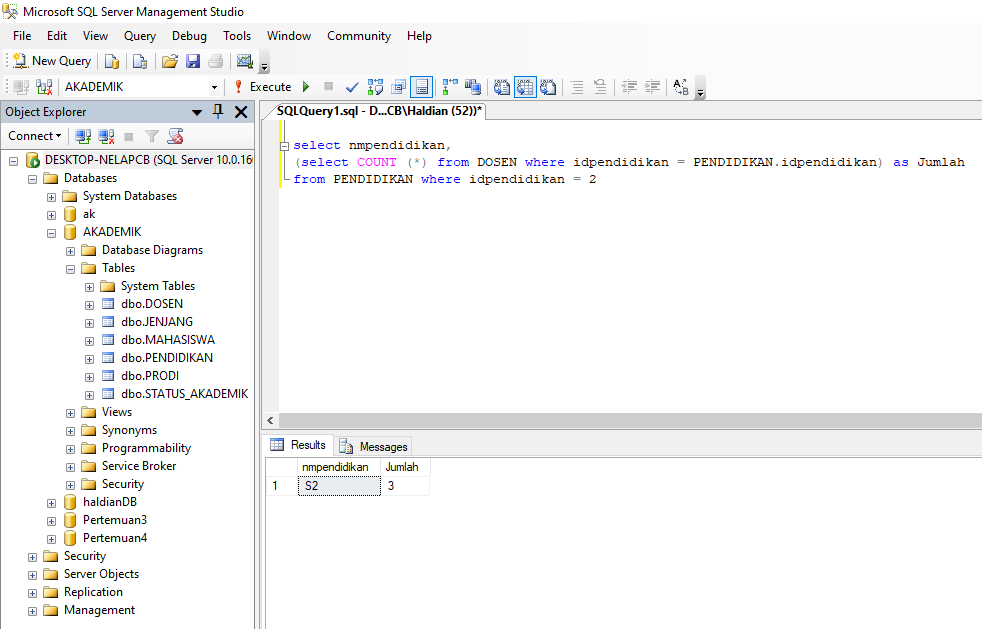
* Rata-rata umur mahasiswa program studi D3 manajemen informatika



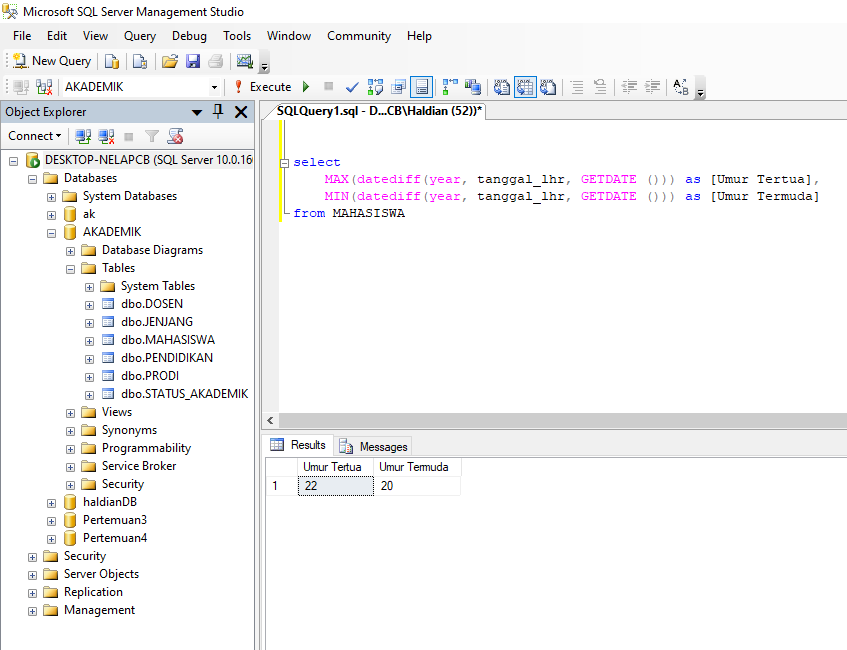
* Jumlah mahasiswa masing-masing program studi



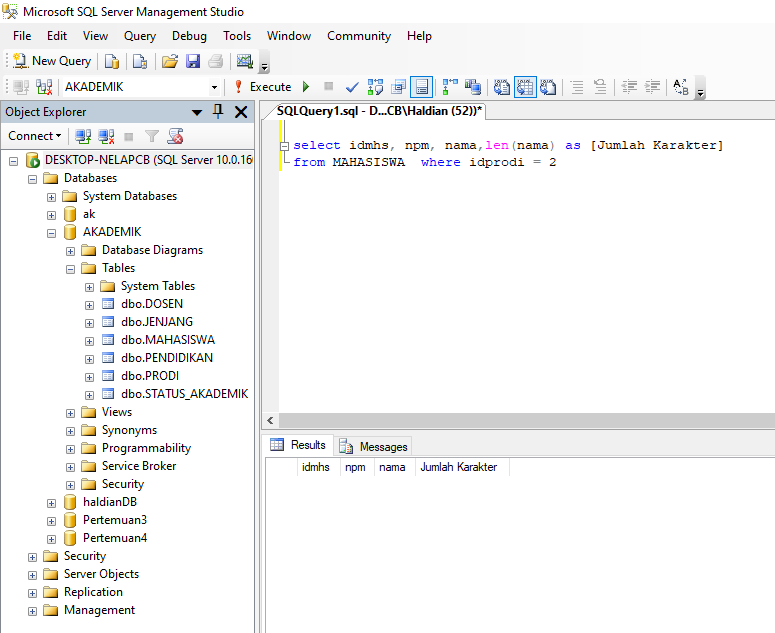
* Jumlah dosen dengan jenjang pendidikan S2



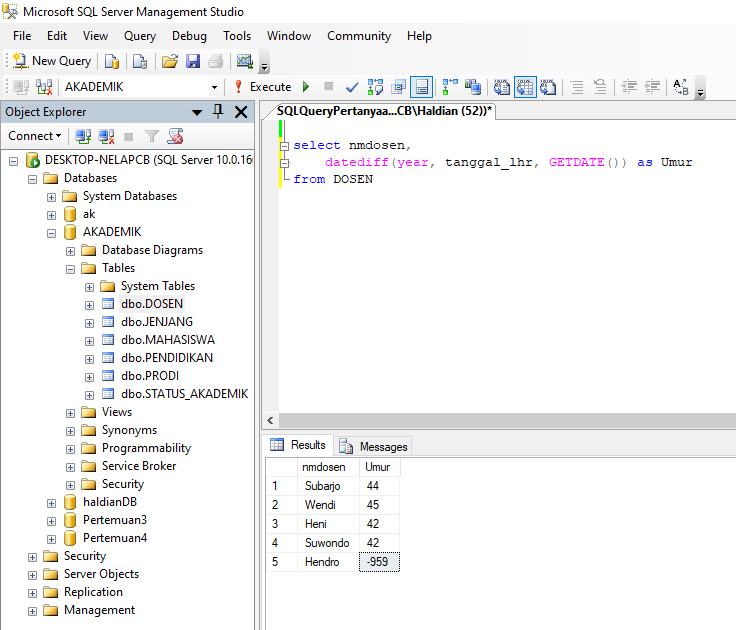
* Umur Mahasiswa Tertua dan Termuda dari seluruh Program Studi



* Jumlah karakter masing-masing nama mahasiswa Program Studi D3 Akuntansi



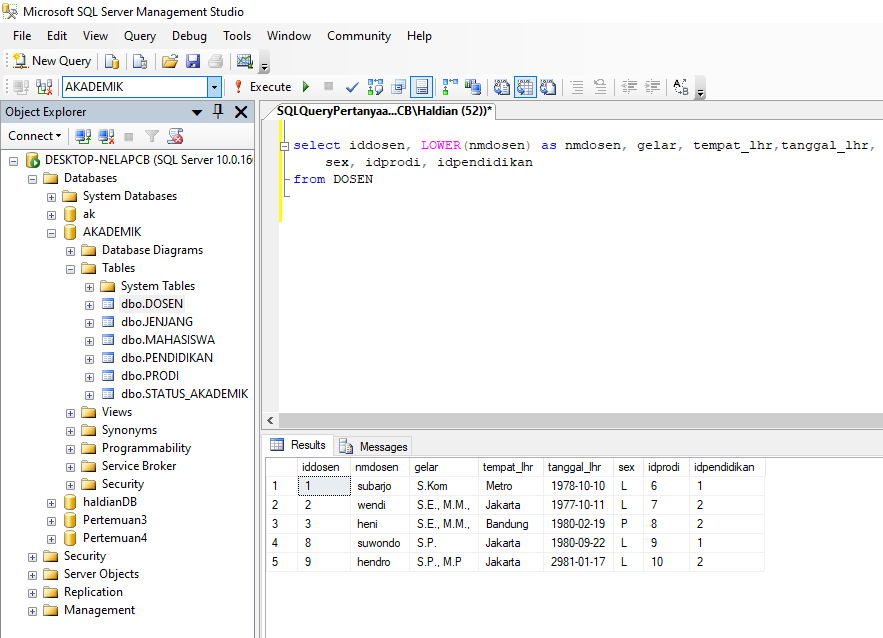
* Daftar Dosen dengan format kolom: ‘nmdosen’ dan ‘umur’



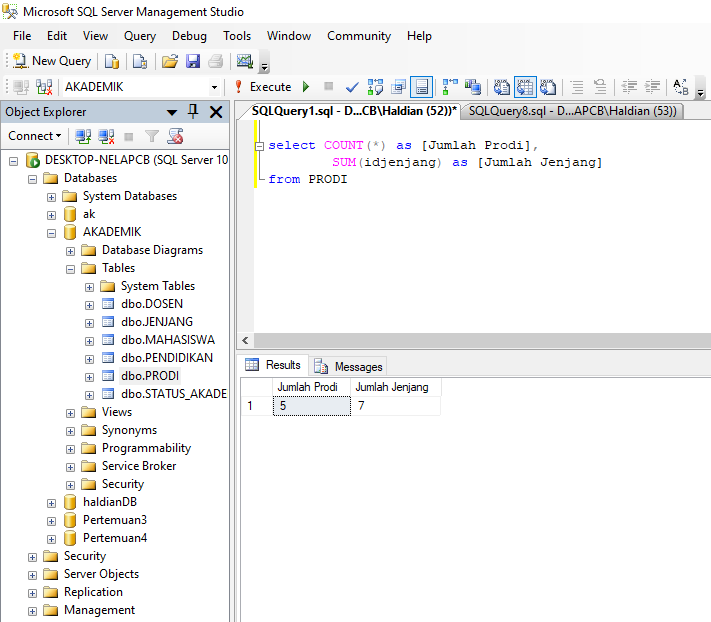
* Daftar Mahasiswa; dimana nama mahasiswa disajikan dengan Huruf Kapital



* Daftar Dosen; dimana nama dosen disajikan dengan huruf kecil

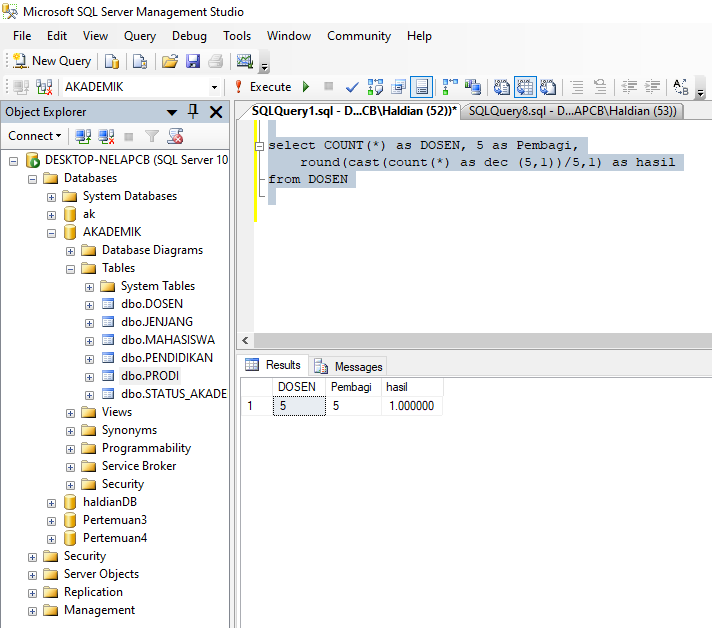


* Contoh penggunaan sintak perintah SUM dan ROUND pada studi kasus Database “AKADEMIK”
* Menampilkan jumlah prodi dan jumlah jenjang dengan sum seperti gambar di bawah ini



* Menampilkan pembulatan dari jumlah DOSEN dengan oprator pembagi

Seperti gambar di bawah ini



# **BAB 4. PENUTUP**

**A. KESIMPULAN**

SQL (Structured Query Language) merupakan suatu sintaks atau perintah yang di gunakan untuk mengakses data dalam sebuah *database*, SQL sendiri juga bisa digunakan untuk memanipulasi sebuah data dalam database seperti untuk menambah data (*insert*), mengubah data (*update*), menampilkan data (*select*), dan juga menghapus data (*delete*). Secara umum, SQL terdiri dari dua bahasa, yaitu *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML). Implementasi DDL dan DML berbeda untuk tiap sistem manajemen basis data (SMBD), namun secara umum implementasi tiap bahasa ini memiliki bentuk standar yang ditetapkan .

**B. SARAN**

Setelah melakukan praktikum *Transact-SQL* tentang *SELECT, SUBQUERY,* dan *JOIN* di harapkan mahasiswa dapat mengimplementasikannya dalam project-project SQL dan mahasiswa dapat melanjutkan materi selanjutnya mengenai SQL Lanjut.

# **PUSTAKA**

Tutorials Point, Microsoft SQL Server, Tutorials Point Ltd, USA, 2003

Ken Henderson, The Guru’s Guide to Transact-SQL, USA, AddisonWesley, 2000

Jan L. Harrington, SQL Clearly Explained, Third Edition, USA, Morgan Kaufmann Publishers, 2010

Eko Win Kenali, Pemrograman SQL menggunakan DBMS Microsoft SQL Server, UP Politeknik Negeri Lampung, 201

SQL Tersedia di: https://www.w3schools.com/sql/ [Diakses 5 April 2022].

Setiadi, M.F. (2017). Data Manipulation Language Pada MySQL. [online] mfikri. Tersedia di: https://mfikri.com/artikel/Data-Manipulation-Language.html [Diakses 5 April 2022].