**BAB I**

**PENDAHULUAN**

* 1. **Latar Belakang**

Pada zaman ini, pengelolaan data menjadi aspek penting di berbagai bidang, mulai dari bisnis, pendidikan, hingga pemerintahan. Data yang dikelola sangat beragam, termasuk data pelanggan, transaksi, akademik, hingga data kependudukan. Untuk mengelola data yang besar dan kompleks tersebut, diperlukan sebuah sistem yang mampu melakukannya secara efisien dan efektif.

Salah satu sistem yang digunakan untuk pengelolaan data adalah sistem basis data relasional. Sistem ini memungkinkan pengguna untuk mendefinisikan, membuat, memelihara, dan mengatur akses ke basis data. Bahasa yang digunakan pada sistem ini adalah Structured Query Language (*SQL*).

*SQL* menawarkan berbagai fungsi yang mempermudah pengguna dalam mengelola data, mulai dari membuat, memperbarui, dan menghapus tabel dalam basis data; menambahkan, memperbarui, serta menghapus data di dalam tabel; memilih data tertentu dari satu atau beberapa tabel; mengelompokkan data dan melakukan agregasi; hingga menggabungkan data dari beberapa tabel.

Memahami fungsi-fungsi *SQL* sangatlah penting bagi siapa saja yang bekerja dengan sistem basis data relasional. Dengan pemahaman tersebut, pengguna dapat mengelola data secara lebih efisien dan efektif. Oleh karena itu, makalah ini akan membahas lebih jauh mengenai fungsi-fungsi *SQL* tersebut.

* 1. **Tujuan**

a. Mahasiswa dapat memahami fungsi-fungsi *SQL*.

b. Mahasiswa dapat memahami *single row functions* dalam *SQL*.

c. Mahasiswa dapat memahami *character functions* dalam *SQL*.

**BAB II**

**DASAR TEORI**

**2.1 Pengertian fungsi-fungsi *SQL***

Fungsi *SQL* atau *Structured Query Language* adalah bahasa pemrograman yang digunakan untuk mengakses, mengelola, dan mengambil data dari *database*. Namun secara umum, *SQL* difungsikan sebagai bahasa pengelolaan *database* yang dapat mengelola data-data pada database yang dipakai oleh aplikasi. *SQL* juga membantumu mengakses database dan melakukan *query* yang dibutuhkan.

Fungsi-fungsi *SQL* memungkinkan pembuatan blok *query* dasar menjadi lebih cepat dan mempermudah manipulasi nilai-nilai data. Fungsi-fungsi ini terbagi menjadi beberapa jenis, salah satunya adalah fungsi *single-row* yang bekerja pada karakter, angka, dan tanggal. Pada bagian pertama dari dua pelajaran yang membahas fungsi *SQL* ini, fokusnya adalah fungsi-fungsi *single-row* tersebut. Fungsi ini mempermudah pengguna dalam mengubah tipe data dari satu bentuk ke bentuk lain. Contohnya, data yang awalnya berbentuk karakter dapat dikonversi menjadi data numerik untuk keperluan analisis tertentu. Selain itu, fungsi-fungsi ini memungkinkan perubahan format data tanggal, seperti menampilkan hanya bulan atau tahun dari suatu tanggal.

Dengan adanya fungsi-fungsi ini, pengguna dapat melakukan manipulasi data dengan lebih fleksibel dan efisien, membantu mereka menyesuaikan data sesuai kebutuhan *query* yang lebih spesifik. Penggunaan fungsi-fungsi ini juga memberikan fleksibilitas yang lebih tinggi dalam menghasilkan hasil yang akurat dan relevan tanpa harus memodifikasi data asli. Di samping itu, fungsi-fungsi *single-row* ini memungkinkan penanganan data yang kompleks menjadi lebih sederhana, sehingga pengguna dapat mengekstrak atau menyesuaikan informasi dengan lebih mudah dan efektif. Hal ini sangat bermanfaat untuk meningkatkan kinerja *query* dalam basis data dan mempercepat proses pengolahan data, baik dalam skala kecil maupun besar.

**2.2 Macam-Macam Fungsi *SQL***

Berikut merupakan Macam-macam dari fungsi *SQL* diantarnya yakni sebagai berikut ini:

1. *Single-row function*

*Single-row function* digunakan untuk memanipulasi item-item data. *Single-row* *functions* menerima satu atau lebih argumen-argumen dan mengembalikan satu nilai untuk setiap baris yang dihasilkan oleh suatu *query*.

|  |
| --- |
| SELECT nama, UPPER(nama) AS nama\_uppercase  FROM pegawai; |

**Tabel 4.1** *Contoh sederhana single row function*

Dalam contoh ini, kolom nama diubah menjadi huruf kapital pada kolom baru yang dinamakan nama\_*uppercase*. Jadi, jika ada nama seperti "Danur Wenda," hasilnya akan menjadi "DANUR WENDA".

Fungsi *UPPER* dalam *SQL* digunakan untuk mengubah teks dalam suatu kolom menjadi huruf kapital. Fungsi ini termasuk kategori *single row function*, yang artinya fungsi ini bekerja pada satu baris data sekaligus dan menghasilkan satu *output* untuk setiap *input*. Misalnya, jika kita memiliki tabel bernama pegawai dengan kolom nama, kita bisa menerapkan *UPPER* untuk memastikan semua nama ditampilkan dalam huruf kapital. Dengan menjalankan query seperti SELECT nama, *UPPER*(nama) *AS* nama\_*uppercase* *FROM* pegawai;, *SQL* akan menampilkan dua kolom: kolom nama yang menampilkan nama asli, dan kolom nama\_*uppercase* yang menampilkan hasil perubahan menjadi huruf kapital. Fungsi ini sangat berguna ketika kita membutuhkan tampilan teks yang seragam atau dalam kasus pencarian data yang tidak peka terhadap huruf besar-kecil. Misalnya, kita bisa mencari nama tertentu tanpa peduli apakah hurufnya besar atau kecil, cukup dengan menuliskan kondisi *WHERE* *UPPER*(nama) = 'DANUR WENDA'. Dengan cara ini, *SQL* akan mengabaikan perbedaan huruf kapital dan memberikan hasil yang lebih akurat dan seragam.

1. *Aggregate Functions*

*Aggregate Functions* merupakan salah satu perintahyang paling sering digunakan. Fungsi agregat merupakan fungsi yang melakukan perhitungan pada sekumpulan data yang menghasilkan sebuah nilai. Hasil dari fungsi agregat tersebut umumnya digunakan untuk memberikan kesimpulan dari data yang ada dan kemudian digunakan untuk analisis lanjutan.

1. *Analytical Functions*

Dalam *SQL* *Analytical Functions* adalah alat fungsi analitik untuk menghitung nilai agregat berdasarkan sekelompok baris. Fungsi ini dapat mengembalikan beberapa baris untuk setiap kelompok. Contohnya, untuk menghitung rata-rata bergerak, total berjalan, persentase, atau hasil N teratas dalam suatu kelompok. Fungsi analitik SQL, yang juga dikenal sebagai fungsi jendela, adalah fitur canggih yang melakukan kalkulasi di seluruh baris dalam satu set hasil, yang memungkinkan analisis dan agregasi data tingkat lanjut dalam kueri. Hal ini memberi ilmuwan dan analis data cara untuk mengimplementasikan logika tanpa bahasa pemrograman sisi klien yang beroperasi secara individual pada set hasil (misalnya, *Python*).

*Analytical functions* dalam *SQL* adalah fungsi yang memungkinkan kita melakukan perhitungan kompleks terhadap sekumpulan baris data tanpa harus menggabungkan atau mengelompokkan data seperti pada *GROUP BY*. Fungsi-fungsi ini sering digunakan dalam analisis data karena mereka dapat menghasilkan *output* yang kaya dan informatif dalam setiap baris hasil *query*.

Contoh umum dari *analytical functions* termasuk *RANK*(), *ROW\_NUMBER*(), *DENSE\_RANK*(), dan *SUM*() dengan *OVER*(). *Analytical functions* berbeda dengan agregat standar karena mereka memungkinkan kita untuk mempertahankan semua baris dan memberikan hasil yang lebih rinci, seperti peringkat atau akumulasi nilai dalam data yang diurutkan. Misalnya, jika kita ingin memberi peringkat pada pegawai berdasarkan gaji dalam departemennya, kita bisa menggunakan *RANK*() atau *DENSE\_RANK*() yang ditulis dengan klausa *OVER* (*PARTITION BY* departemen *ORDER BY* gaji *DESC*).

d. *Set Functions*

Operator *set* adalah perintah atau simbol khusus yang digunakan untuk melakukan operasi pada set hasil dari beberapa kueri *SELECT*. Operator ini memungkinkan kita untuk melakukan tugas-tugas seperti menemukan gabungan (semua baris), irisan (baris yang sama), dan selisih (baris unik) antara set data yang berbeda. Operator *set* merupakan alat penting untuk manipulasi dan analisis data dalam *SQL*. Operator *set* menyediakan cara yang ampuh untuk meminta dan memproses data dari basis data relasional.

e. *Scalar Functions*

Fungsi skalar *SQL* adalah fungsi yang ditentukan pengguna yang ditulis dalam *SQL* dan mengembalikan satu nilai setiap kali dipanggil. Fungsi skalar *SQL* berisi kode sumber untuk fungsi yang ditentukan pengguna dalam definisi fungsi yang ditentukan pengguna. Fungsi skalar *SQL* adalah fungsi yang ditentukan pengguna yang ditulis dalam *SQL* dan mengembalikan satu nilai setiap kali dipanggil. Fungsi skalar *SQL* berisi kode sumber untuk fungsi yang ditentukan pengguna dalam definisi fungsi yang ditentukan pengguna. Ada dua jenis fungsi skalar *SQL*, yaitu yang disematkan dan yang dikompilasi.

Semua fungsi skalar *SQL* yang dibuat sebelum DB210 adalah fungsi skalar *SQL* yang disematkan. Dimulai dengan DB210 Fungsi skalar *SQL* dapat dibuat sebagai fungsi sebaris atau dikompilasi. Db2 menentukan apakah fungsi skalar SQL disematkan atau dikompilasi berdasarkan apakah pernyataan *CREATE FUNCTION* yang mendefinisikan fungsi tersebut menggunakan fitur-fitur yang disempurnakan atau tidak.

Misalkan kita memiliki tabel bernama pegawai yang menyimpan informasi tentang setiap karyawan, termasuk nama, departemen, dan gaji mereka. Dengan data ini, kita ingin menentukan peringkat gaji dalam tiap departemen, misalnya siapa yang memiliki gaji tertinggi, kedua tertinggi, dan seterusnya, namun tetap mempertahankan seluruh baris data tanpa menggabungkannya. Untuk melakukan ini, kita dapat menggunakan fungsi *RANK*() dengan klausa *OVER*().

Fungsi *RANK*() bekerja dengan memberi peringkat pada baris data dalam suatu kelompok atau partition. Dalam contoh ini, kita membagi data berdasarkan kolom departemen dan mengurutkan nilai gaji secara menurun menggunakan klausa *OVER*(*PARTITION BY* departemen *ORDER BY* gaji *DESC*). Klausa *PARTITION BY* membagi data berdasarkan departemen, sehingga peringkat hanya diterapkan dalam kelompok yang sama, sementara *ORDER BY* gaji *DESC* memastikan bahwa peringkat diberikan dari nilai gaji tertinggi ke terendah.

Hasil dari *query* ini menunjukkan kolom tambahan bernama peringkat\_gaji, yang menampilkan urutan peringkat pegawai di setiap departemen. Misalnya, jika Danur dan Rina memiliki gaji tertinggi di departemen IT, keduanya mendapat peringkat 1. Karena fungsi *RANK*() tidak memberikan peringkat berurutan pada nilai yang sama, jika ada dua orang di peringkat 1, maka peringkat berikutnya akan dilanjutkan dari 3, melewati peringkat 2. Ini disebut *ranking gap*, di mana fungsi *RANK*() melompati peringkat setelah nilai yang sama.

*Analytical functions* seperti *RANK*() sangat berguna dalam analisis data untuk menghasilkan informasi detail tanpa harus mengelompokkan atau menghapus data asli. Fungsinya memungkinkan kita untuk melakukan analisis tingkat lanjut pada sekumpulan data yang dibagi ke dalam kelompok tertentu, memberikan fleksibilitas dan kontrol lebih dalam membuat laporan, peringkat, dan agregasi data.

**BAB III**

**PERMASALAHAN**

**3.1 Permasalahan**

a. Buatlah perintah *SQL* yang memanfaatkan *single row function* (Masing-masing 1)!

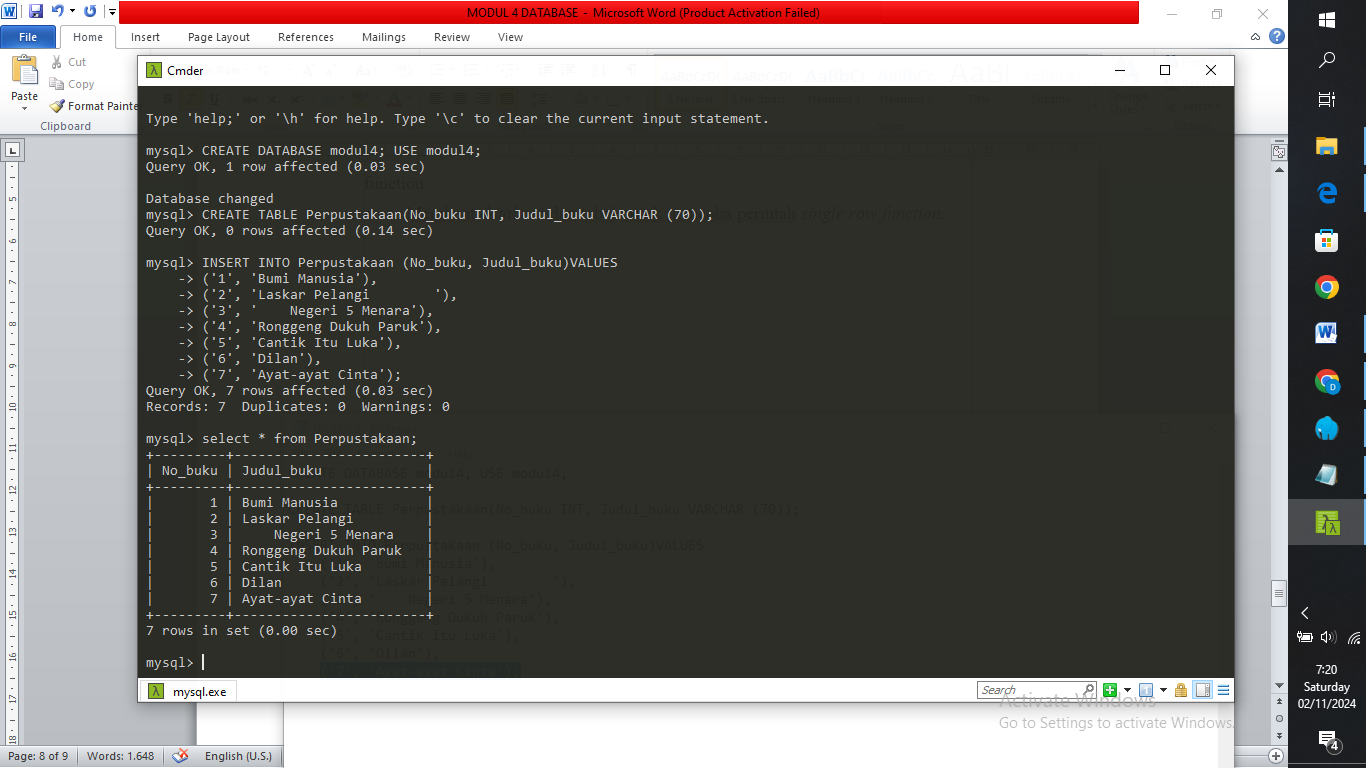
**BAB IV**

**IMPLEMENTASI**

**4.1 Implementasi**

Berikut dibawah ini adalah perintah SQL yang memanfaatkan single row function:

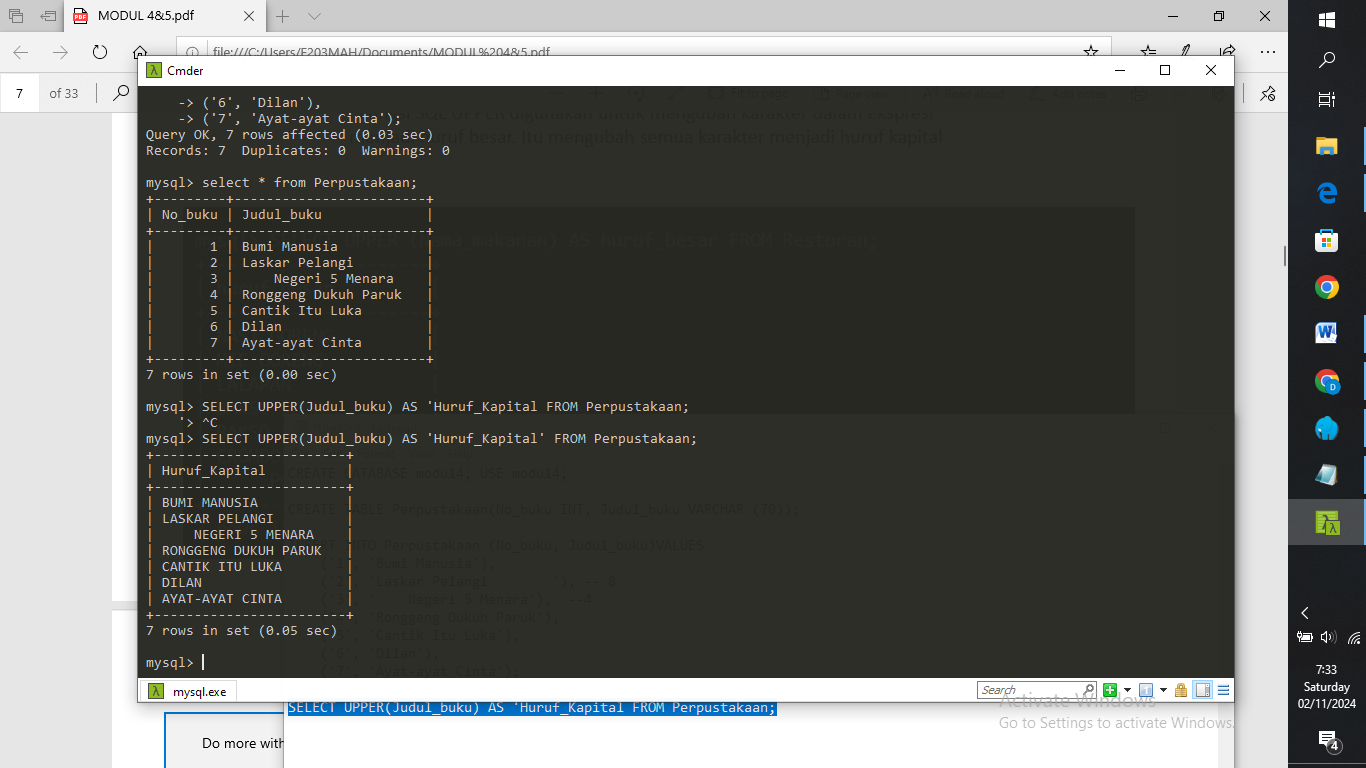
a. Membuat *database* dan tabel untuk mencoba perintah *single row function.*

**

**Gambar 4.1** *Membuat database beserta tabel perpustakaan.*

Kode *SQL* di atas melakukan beberapa hal secara bertahap. Pertama, membuat *database* baru bernama modul4 dan memilihnya sebagai *database* yang aktif dengan perintah USE modul4;. Kemudian, dibuat tabel bernama Perpustakaan yang berisi dua kolom, yaitu No\_buku untuk nomor buku (dengan tipe data angka) dan Judul\_buku untuk judul buku (dengan panjang maksimal 70 karakter).Selanjutnya, data dimasukkan ke dalam tabel Perpustakaan melalui perintah *INSERT INTO*. Data yang dimasukkan terdiri dari tujuh judul buku populer, seperti "Bumi Manusia" dan "Laskar Pelangi", dengan masing-masing buku memiliki nomor urut sebagai identifikasi. Ada beberapa judul buku yang memiliki spasi tambahan di awal atau akhir teks.

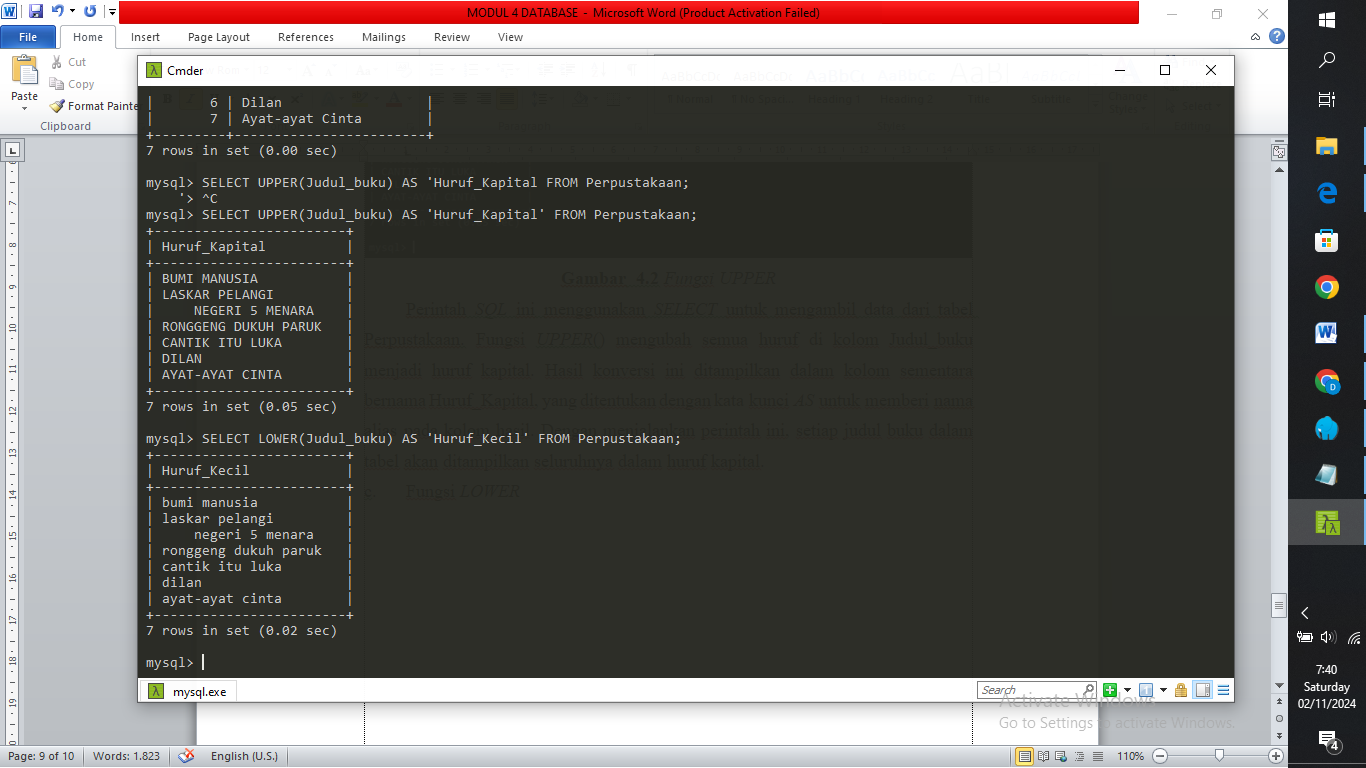
b. Fungsi *UPPER*



**Gambar 4.2** *Fungsi UPPER*

Perintah *SQL* ini menggunakan *SELECT* untuk mengambil data dari tabel Perpustakaan. Fungsi *UPPER*() mengubah semua huruf di kolom Judul\_buku menjadi huruf kapital. Hasil konversi ini ditampilkan dalam kolom sementara bernama huruf kapital, yang ditentukan dengan kata kunci *AS* untuk memberi nama alias pada kolom hasil. Dengan menjalankan perintah ini, setiap judul buku dalam tabel akan ditampilkan seluruhnya dalam huruf kapital.

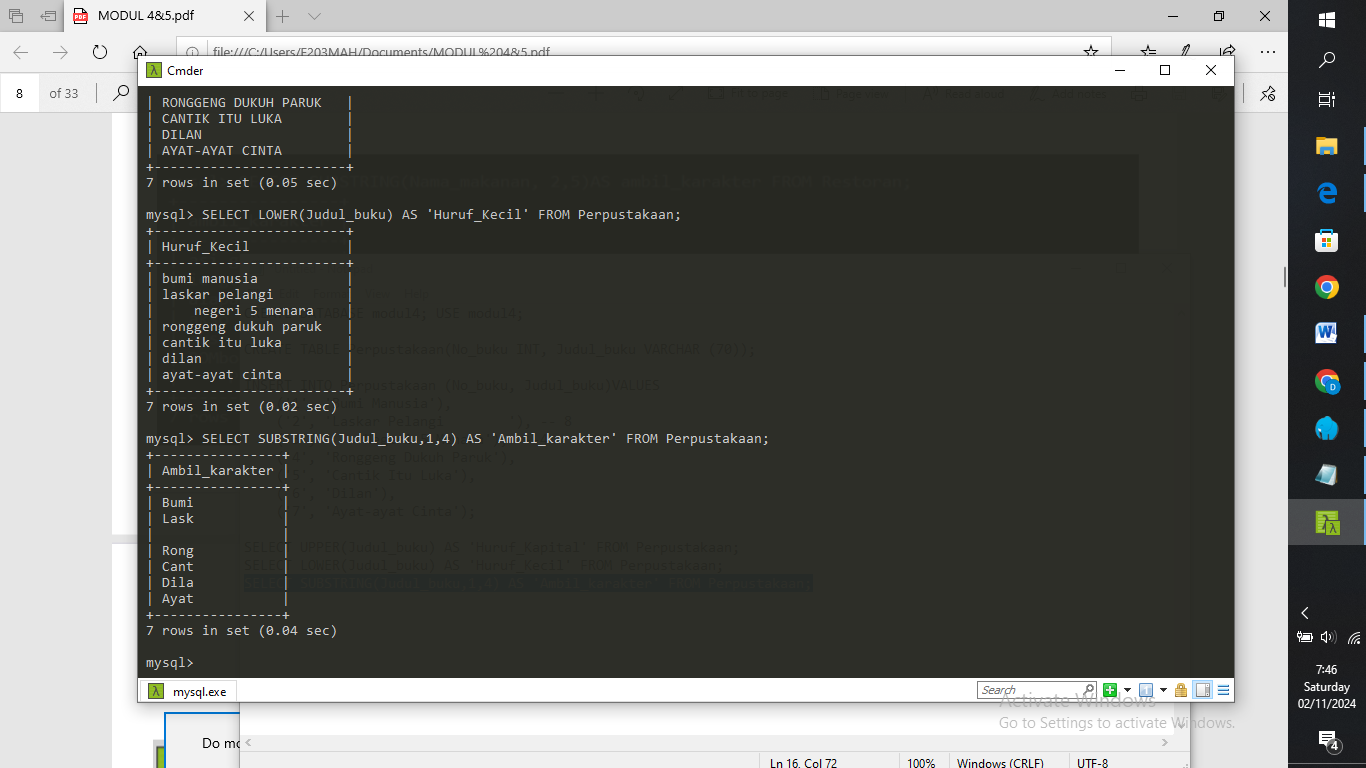
c. Fungsi *LOWER*



**Gambar 4.3** *Fungsi LOWER*

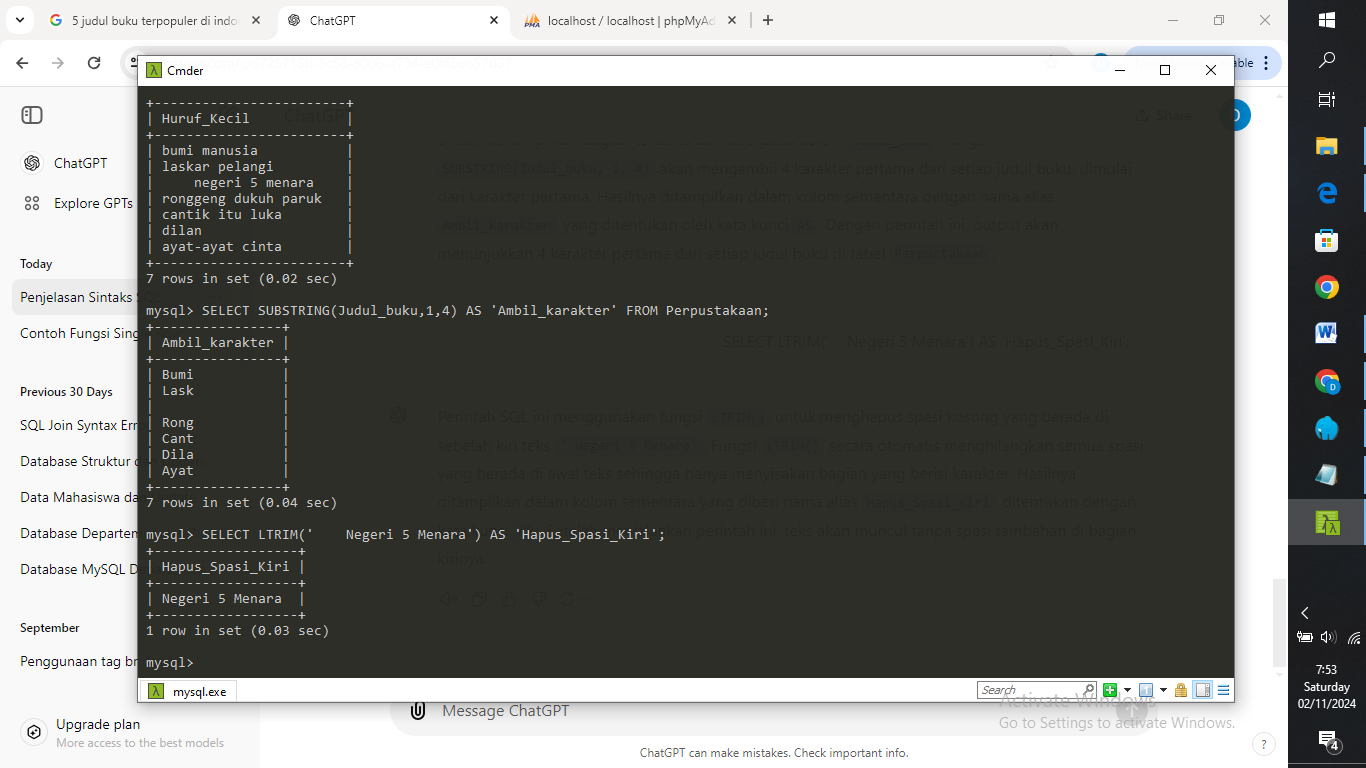
Perintah *SQL* ini menggunakan *SELECT* untuk mengambil data dari tabel Perpustakaan dan mengonversi semua huruf di kolom Judul\_buku menjadi huruf kecil menggunakan fungsi *LOWER*(). Hasil konversi ini ditampilkan dalam kolom sementara yang diberi nama alias huruf kecil dengan menggunakan kata kunci *AS*.

d. Fungsi SUBSTRING

**Gambar 4.4** *Fungsi SUBSTRING*

Perintah *SQL* ini mengambil data dari tabel Perpustakaan dan menggunakan fungsi *SUBSTRING*() untuk menampilkan bagian tertentu dari teks pada kolom Judul\_buku. Fungsi *SUBSTRING*(Judul\_buku, 1, 4) akan mengambil 4 karakter pertama dari setiap judul buku, dimulai dari karakter pertama. Hasilnya ditampilkan dalam kolom sementara dengan nama alias ‘Ambil\_karakter’, yang ditentukan oleh kata kunci *AS*. Dengan perintah ini, *output* akan menunjukkan 4 karakter pertama dari setiap judul buku di tabel Perpustakaan.

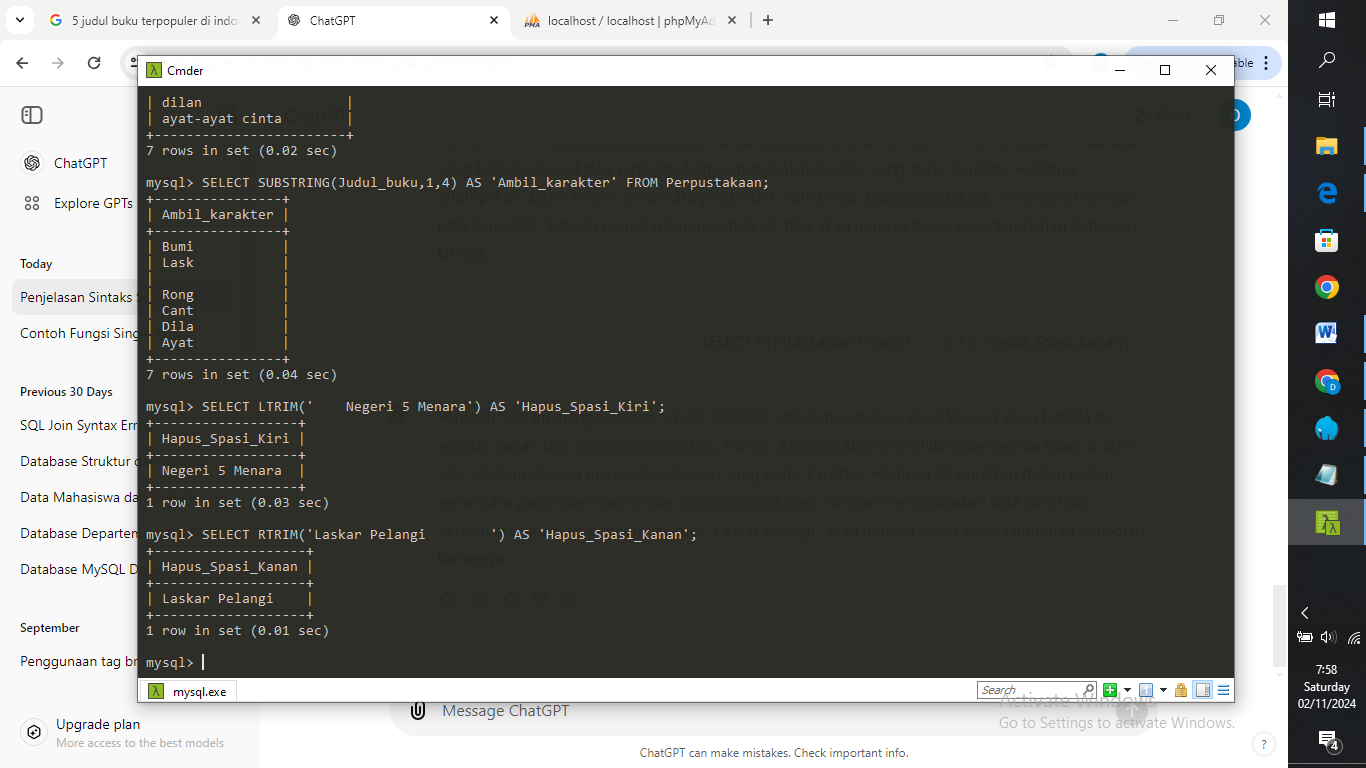
e. Fungsi *LTRIM*



**Gambar 4.5** *Fungsi LTRIM*

Perintah *SQL* ini menggunakan fungsi *LTRIM*() untuk menghapus spasi kosong yang berada di sebelah kiri teks ' Negeri 5 Menara'. Fungsi *LTRIM*() secara otomatis menghilangkan semua spasi yang berada di awal teks sehingga hanya menyisakan bagian yang berisi karakter. Hasilnya ditampilkan dalam kolom sementara yang diberi nama alias ‘Hapus\_Spasi\_Kiri’, ditentukan dengan kata kunci *AS*. Setelah menjalankan perintah ini, teks akan muncul tanpa spasi tambahan di bagian kirinya.

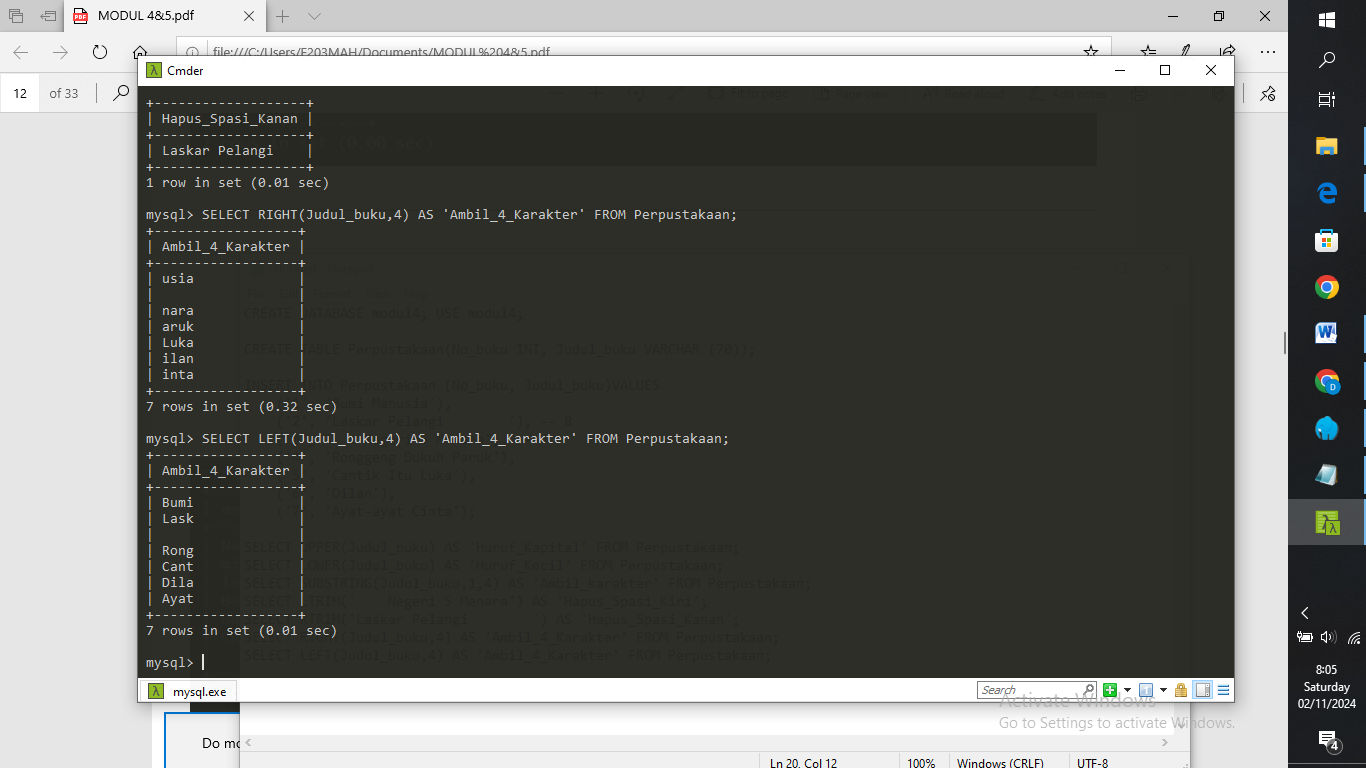
f. Fungsi *RTRIM*



**Gambar 4.5** *Fungsi RTRIM*

Perintah *SQL* ini menggunakan fungsi *RTRIM*() untuk menghapus spasi kosong yang berada di sebelah kanan teks 'Laskar Pelangi '. Fungsi *RTRIM*() akan menghilangkan semua spasi di akhir teks sehingga hanya menyisakan bagian yang berisi karakter. Hasilnya ditampilkan dalam kolom sementara yang diberi nama alias ‘Hapus\_Spasi\_Kanan’, dengan menggunakan kata kunci *AS*. Setelah menjalankan perintah ini, teks "Laskar Pelangi" akan muncul tanpa spasi tambahan di bagian kanannya.

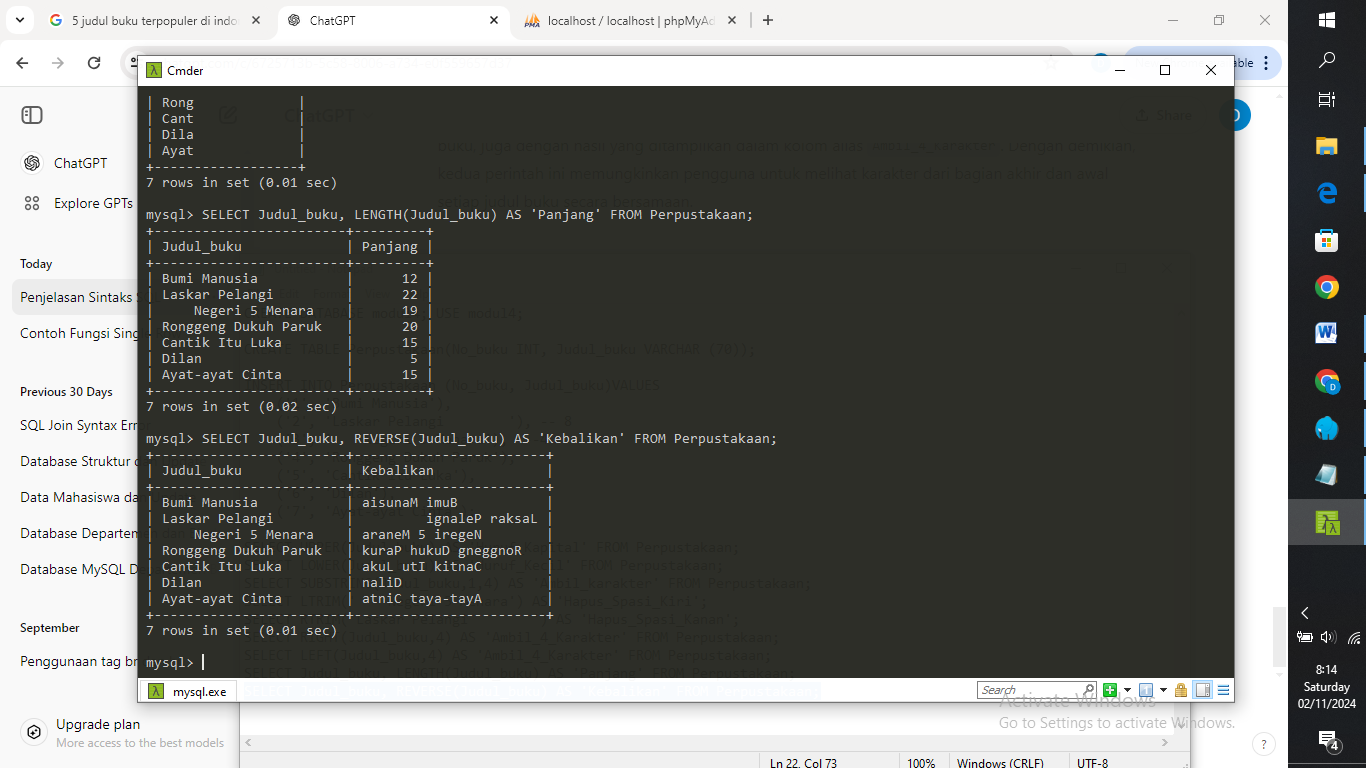
g. Fungsi *RIGHT* dan *LEFT*



**Gambar 4.6** *Fungsi RIGHT dan LEFT*

Dua perintah *SQL* ini menggunakan fungsi *RIGHT*() dan *LEFT*() untuk mengambil sejumlah karakter tertentu dari kolom Judul\_buku dalam tabel Perpustakaan. Perintah pertama, *SELECT RIGHT*(Judul\_buku, 4) *AS* 'Ambil\_4\_Karakter' *FROM* Perpustakaan;, menggunakan fungsi *RIGHT*() untuk mengambil 4 karakter terakhir dari setiap judul buku. Hasilnya ditampilkan dalam kolom sementara yang diberi alias Ambil\_4\_Karakter, yang menunjukkan bagian akhir dari setiap judul. Sementara itu, perintah kedua, *SELECT LEFT*(Judul\_buku, 4) AS 'Ambil\_4\_Karakter' *FROM* Perpustakaan;, menggunakan fungsi *LEFT*() untuk mengambil 4 karakter pertama dari setiap judul buku, juga dengan hasil yang ditampilkan dalam kolom alias ‘Ambil\_4\_Karakter’.

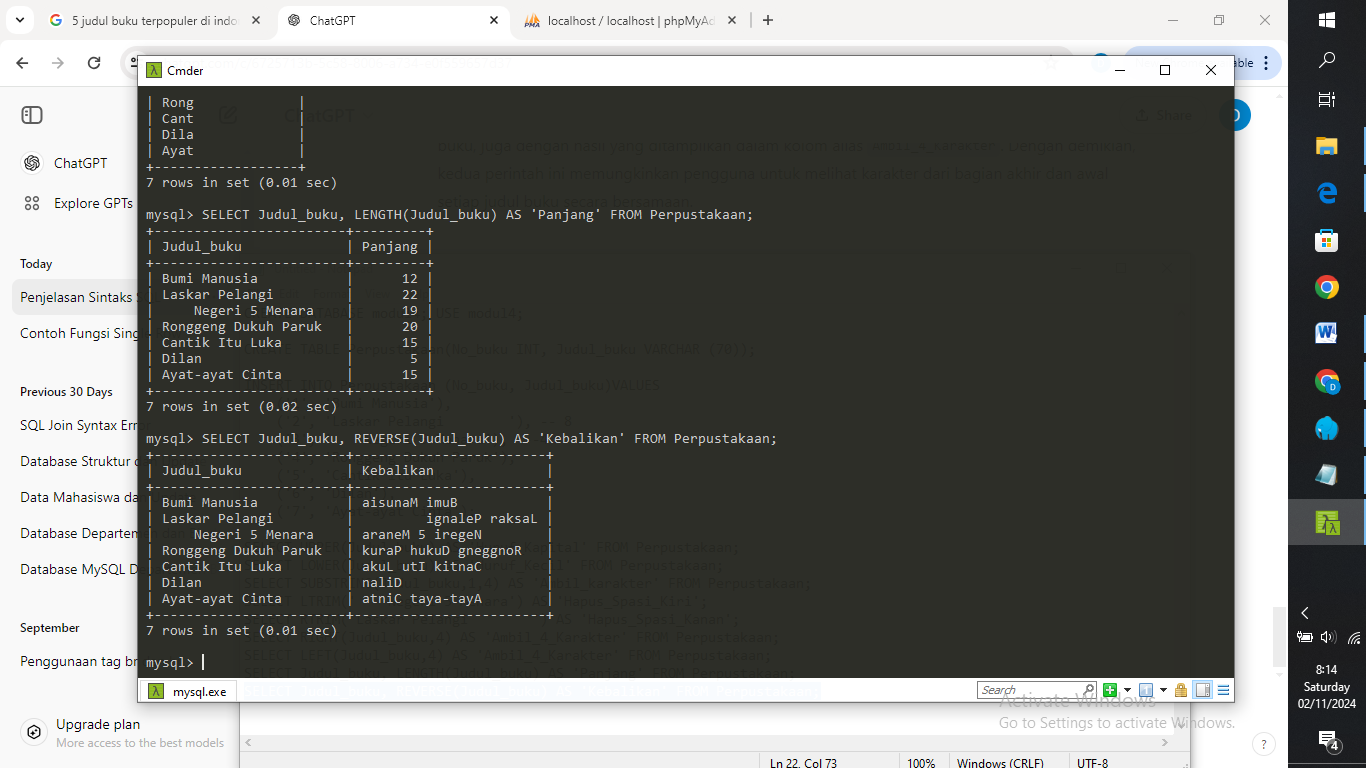
h. Fungsi *LENGTH*



**Gambar 4.7** *Fungsi LENGTH*

Perintah *SQL* ini mengambil data dari tabel Perpustakaan dengan memilih kolom Judul\_buku dan menghitung panjang setiap judul buku menggunakan fungsi *LENGTH*(). Fungsi ini menghitung jumlah karakter dalam setiap judul dan menampilkan hasilnya dalam kolom bernama alias Panjang. Setelah menjalankan perintah ini, pengguna akan melihat daftar judul buku beserta jumlah karakter yang ada pada masing-masing judul.

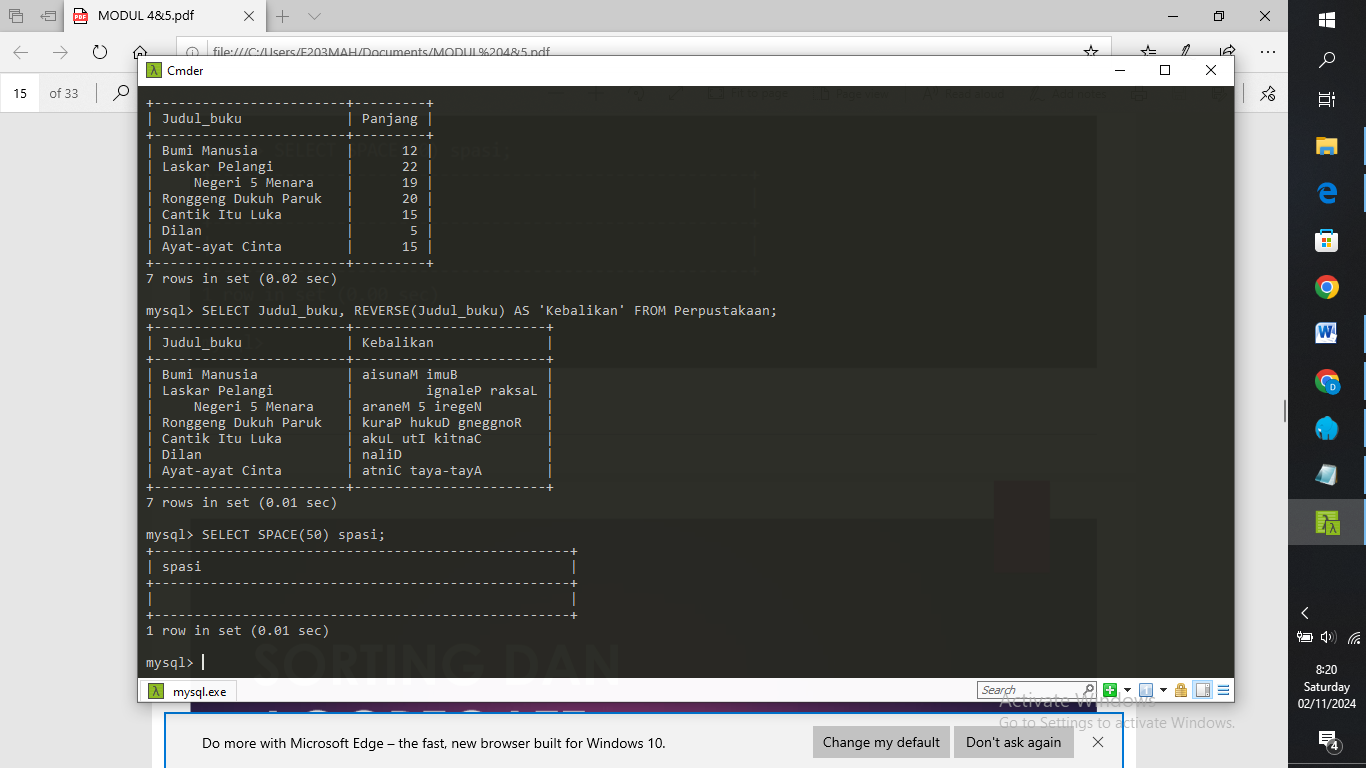
i. Fungsi *REVERSE*



**Gambar 4.8** *Fungsi REVERSE*

Perintah *SQL* ini mengambil data dari tabel Perpustakaan, menampilkan kolom Judul\_buku dan membalikkan urutan karakter dalam setiap judul menggunakan fungsi *REVERSE*(). Hasilnya ditampilkan dalam kolom yang diberi alias Kebalikan.

j. Fungsi *SPACE*



**Gambar 4.8** *Fungsi SPACE*

Perintah *SQL* ini menggunakan fungsi *SPACE*(50) untuk menghasilkan *string* yang terdiri dari 50 spasi kosong. Hasil dari fungsi ini ditampilkan dalam kolom yang diberi alias spasi. Setelah menjalankan perintah ini, pengguna akan melihat satu kolom yang berisi 50 karakter spasi, yang dapat digunakan untuk tujuan *formatting* atau pemisahan dalam *output*.

**BAB V**

**PENUTUP**

**5.1 Kesimpulan**

Fungsi dalam *SQL* adalah blok kode yang dirancang untuk melakukan tugas spesifik pada data. Mereka seperti alat-alat kecil yang dapat kita gunakan untuk memanipulasi, menganalisis, dan mengubah data dalam *database*.

Fungsi-fungsi ini sangat berguna untuk mempermudah penulisan *query* *SQL*, meningkatkan efisiensi, dan membuat kode *SQL* menjadi lebih terstruktur. Dengan menggunakan fungsi-fungsi ini, kita dapat melakukan berbagai operasi pada data, seperti mencari nilai maksimum atau minimum, mengubah format tanggal, atau menggabungkan beberapa kolom menjadi satu.

**5.2 Saran**

Fungsi *SQL* adalah alat yang sangat penting dalam dunia *database*. Dengan memahami dan menguasai fungsi-fungsi *SQL*, Anda akan dapat melakukan analisis data yang lebih kompleks dan efisien. Teruslah belajar dan berlatih untuk meningkatkan kemampuan Anda dalam menggunakan *SQL*.

Untuk meningkatkan efisiensi dan kejelasan dalam penulisan *query SQL*, ada beberapa tips yang bisa Anda ikuti. Pertama, gunakan alias untuk memberikan nama yang lebih deskriptif pada kolom hasil. Kedua, manfaatkan *subquery* untuk melakukan perhitungan yang lebih kompleks. Ketiga, optimalkan kinerja *query* dengan menghindari penggunaan fungsi yang tidak perlu atau terlalu kompleks. Terakhir, jangan lupa untuk menambahkan komentar pada kode *SQL* Anda agar lebih mudah dipahami, baik oleh Anda sendiri maupun oleh orang lain.