Nama : Firza Ridwan Hafidz

NIM : 2023071014

Tugas SQL

1. PPT Slide 2
   Cara membuat tabel Employee, menggunakan perintah CREATE dengan kolom-kolom sebagai berikut: id_emp dengan tipe data VARCHAR (5), nama_emp dengan tipe data VARCHAR (50), juga age dengan tipe data INT. Setelah tabel dibuat, saya memasukkan data dengan id_emp = '123', nama_emp = 'Budi', dan age = 21 sebanyak dua kali menggunakan perintah INSERT.

2. PPT Slide 3
   Untuk menampilkan data yang telah dimasukkan, saya menggunakan perintah SELECT * FROM Employee, yang artinya saya memilih semua data dari tabel Employee. Karena data tersebut dimasukkan dua kali, mengakibatkan adanya duplikasi dalam tabel. Oleh karena itu, saya menggunakan perintah UPDATE untuk memperbaiki data yang duplikat tersebut.

3. PPT Slide 4
   Selanjutnya, saya ingin menetapkan kolom id_emp pada tabel Employee sebagai Primary Key dengan menggunakan perintah ALTER TABLE. Dengan penetapan ini, setiap nilai id_emp yang dimasukkan harus unik dan tidak boleh sama dengan nilai id_emp yang sudah ada dalam tabel. Sebagai contoh, ketika saya mencoba menambahkan data baru dengan nama Arief dan id_emp = 123, data tersebut tidak dapat dimasukkan karena id_emp = 123 sudah ada dalam tabel dengan entri sebelumnya yaitu Budi, sehingga terjadi error.

4. PPT Slide
   Data berhasil dimasukkan karena setiap nilai id_emp yang diinput adalah unik, sehingga tidak terjadi error. Meskipun terdapat kesamaan pada kolom age, hal tersebut diperbolehkan karena age bukan merupakan Primary Key, sehingga nilai pada kolom ini dapat sama dengan data lainnya. Setelah saya menghapus data dengan id_emp = 123, nama_emp = 'Intan', dan age = 21, data yang tersisa hanya mencakup data milik Arief. Hal ini disebabkan karena data dengan id_emp = 134, nama_emp = 'Arief', dan age = 24 tidak termasuk dalam data yang saya hapus, sehingga hanya tersisa satu entri.

5. PPT Slide 6
   Setelah menghapus data sebelumnya, saya menambahkan kembali data baru ke dalam tabel Employee dengan id_emp = 211, nama_emp = 'Mulya', dan age = 21; id_emp = 212, nama_emp = 'Dewi', dan age = 22; serta id_emp = 213, nama_emp = 'Ayu', dan age = 23. Untuk menampilkan data hasil penambahan, saya menggunakan perintah SELECT * FROM Employee. Selanjutnya, saya menjalankan perintah SQL DELETE FROM Employee untuk menghapus semua data dari tabel. Setelah perintah ini dijalankan, saya memeriksa tabel dengan SELECT * FROM Employee untuk memastikan bahwa tabel sudah kosong. Kemudian, untuk membatalkan penghapusan data, saya menggunakan fitur transaksi ROLLBACK, jika sebelumnya dilakukan dalam sebuah transaksi. Jika tidak menggunakan transaksi, penghapusan data tidak dapat dibatalkan dan data yang telah dihapus tidak dapat dipulihkan.

6. PPT Slide 7
   Setelah menambahkan kembali semua data ke dalam tabel Employee dengan id_emp = 123, nama_emp = 'Budi B', dan age = 21; id_emp = 234, nama_emp = 'Dani', dan age = 23; id_emp = 134, nama_emp = 'Budi A', dan age = 24; id_emp = 144, nama_emp = 'Intan', dan age = 20; serta id_emp = 155, nama_emp = 'Budi W', dan age = 25, saya menampilkan data yang berisi nama Budi dengan usia di atas 21 tahun menggunakan perintah SELECT * FROM Employee WHERE nama_emp LIKE 'Budi%' AND age > 21. Selain itu, untuk menampilkan data dengan usia antara 22 hingga 24 tahun, saya menggunakan perintah SELECT * FROM Employee WHERE age BETWEEN 22 AND 24.

7. PPT slide 8
   menambahkan kolom gol dengan tipe data VARCHAR(5) pada tabel Employee menggunakan perintah ALTER TABLE Employee ADD COLUMN gol VARCHAR(5), saya kemudian mengisi kolom tersebut dengan data yang sesuai. Jika saya mencoba mengubah data pada tabel tanpa menggunakan kondisi WHERE, maka perubahan tersebut akan diterapkan pada seluruh baris dalam tabel. Misalnya, jika saya menjalankan perintah UPDATE Employee SET gol = 'A' tanpa menyertakan kondisi WHERE, maka setiap baris dalam tabel akan mendapatkan nilai gol = 'A' meskipun seharusnya hanya baris tertentu yang perlu diubah. Hal ini dapat menyebabkan data menjadi tidak akurat atau tidak sesuai dengan yang diharapkan.

8. PPT Slide 9
   - **BETWEEN (numerik)**:
     Perintah BETWEEN digunakan untuk memilih data dalam rentang nilai numerik tertentu. Misalnya, SELECT * FROM Employee WHERE age BETWEEN 20 AND 30; akan menampilkan semua data dengan nilai age di antara 20 hingga 30, termasuk kedua nilai batas tersebut.
   - **NOT BETWEEN**:
     NOT BETWEEN adalah kebalikan dari BETWEEN. Perintah ini digunakan untuk memilih data yang berada di luar rentang yang ditentukan. Misalnya, SELECT * FROM Employee WHERE age NOT BETWEEN 20 AND 30; akan menampilkan semua data dengan nilai age yang kurang dari 20 atau lebih dari 30.
   - **BETWEEN (varchar)**:
     BETWEEN juga dapat digunakan dengan data tipe VARCHAR (teks) untuk memilih data dalam rentang alfabetis. Misalnya, SELECT * FROM Employee WHERE nama_emp BETWEEN 'A' AND 'M'; akan menampilkan semua nama yang alfabetisnya berada di antara 'A' dan 'M'. Ini berguna untuk pengurutan berdasarkan huruf.
   - **IN/NOT IN**:
     IN digunakan untuk memeriksa apakah suatu nilai berada dalam sekumpulan nilai yang ditentukan. Misalnya, SELECT * FROM Employee WHERE age IN (21, 22, 23); akan menampilkan semua data dengan age yang bernilai 21, 22, atau 23. Sebaliknya, NOT IN akan memilih data yang nilainya tidak ada dalam kumpulan tersebut.
   - **LIKE**:
     LIKE digunakan untuk mencocokkan pola teks menggunakan wildcard seperti % dan _. Misalnya, SELECT * FROM Employee WHERE nama_emp LIKE 'Budi%'; akan menampilkan semua data yang dimulai dengan kata 'Budi'. Tanda % berarti ada karakter bebas setelah 'Budi'.

- **AND**:
  AND digunakan untuk menggabungkan dua atau lebih kondisi dalam perintah SQL, yang semuanya harus benar agar hasilnya ditampilkan. Misalnya, SELECT * FROM Employee WHERE age > 21 AND nama_emp LIKE 'B%'; akan menampilkan data yang memenuhi kedua kondisi, yaitu usia di atas 21 dan nama yang dimulai dengan 'B'.

9. PPT slide 10
   Saya menggunakan perintah INSERT INTO untuk menambahkan beberapa baris data dari tabel lain ke dalam tabel Employee. Dengan perintah INSERT INTO Employee (id_emp, nama_emp, age) SELECT kolom_a, kolom_b, kolom_c FROM tabel_lain WHERE kondisi;, saya mengambil data dari tabel lain yang memiliki struktur kolom yang sama. Sebelum menjalankan perintah ini, saya membuat tabel lain dengan struktur yang sama seperti tabel Employee dan mengisi data ke dalamnya. Setelah itu, saya mempraktikkan perintah tersebut untuk menginsert data dari tabel lain ke tabel Employee dengan menggunakan kondisi WHERE untuk memastikan hanya data tertentu yang dimasukkan ke dalam tabel Employee.

10. PPT slide 11
    Untuk menambahkan data ke kolom tertentu saja, saya menggunakan perintah INSERT INTO nama_tabel (kolom1, kolom2) VALUES (nilai1, nilai2);, yang hanya mengisi kolom-kolom tertentu tanpa mempengaruhi kolom lainnya dalam tabel. Jika saya ingin membatalkan operasi INSERT INTO setelah data dimasukkan, saya dapat menggunakan transaksi dan perintah ROLLBACK (apabila database mendukung transaksi). Langkahnya adalah memulai transaksi dengan BEGIN TRANSACTION, kemudian menjalankan perintah INSERT INTO, dan jika ingin membatalkan perubahan, saya dapat menggunakan ROLLBACK untuk mengembalikan tabel ke keadaan sebelum data ditambahkan, sehingga operasi INSERT tidak berpengaruh. Jika transaksi sudah sesuai, maka dapat menggunakan COMMIT untuk menyimpan perubahan secara permanen.

11. PPT slide 12
    Saya mulai dengan membuat tabel Golongan menggunakan perintah CREATE TABLE dan mengisi data yang diperlukan ke dalam tabel tersebut. Setelah itu, saya menambahkan Foreign Key (FK) pada tabel Employee untuk menghubungkan kolom gol di tabel Employee dengan kolom gol di tabel Golongan. Langkah ini dilakukan dengan menggunakan perintah ALTER TABLE Employee ADD CONSTRAINT fk_golongan FOREIGN KEY (gol) REFERENCES Golongan(gol);. Dengan demikian, setiap nilai yang dimasukkan ke dalam kolom gol di tabel Employee harus sesuai dengan nilai yang ada di kolom gol dalam tabel Golongan, menjaga konsistensi data antara kedua tabel.

12. PPT slide 13
    Untuk menampilkan nama golongan untuk masing-masing karyawan, saya melakukan query SELECT yang menggabungkan tabel Employee dan tabel Golongan berdasarkan Foreign Key gol. Perintah yang digunakan adalah SELECT Employee.id_emp, Employee.nama_emp, Golongan.nama_golongan FROM Employee JOIN Golongan ON Employee.gol = Golongan.gol;. Dengan menggunakan perintah JOIN untuk merelasikan kedua tabel melalui Foreign Key gol, saya dapat menampilkan nama golongan yang sesuai untuk setiap karyawan yang ada dalam tabel Employee. Hasilnya adalah data yang mencakup identitas karyawan dan nama golongan mereka, menghubungkan informasi dari kedua tabel secara akurat.