LAPORAN PENUGASAN

"Python dan SQL”

DISUSUN OLEH:

Muhammad Aryasatya Nugroho (22083010085)

DOSEN PENGAMPU:

Kartika Maulida Hindrayani, S.Kom, M.Kom (NIP : 199209092022032009)

Dr. Basuki Rahmat,S.Si.,MT

PROGRAM STUDI SAINS DATA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN” JAWA TIMUR

2023

Dalam tugas kali ini, database yang saya buat adalah database sederhana yang digunakan untuk mengelola pemesanan makanan pada sebuah restoran. Database ini terdiri dari tiga tabel yaitu Pelanggan, Menu, dan Pesanan. Tabel Pelanggan berisi informasi tentang pelanggan seperti id, nama, alamat, dan nomor telepon. Tabel Menu berisi informasi tentang menu yang tersedia seperti id, nama, deskripsi, harga, dan kategori. Tabel Pesanan menghubungkan tabel Pelanggan dan Menu serta berisi informasi tentang pemesanan seperti id pesanan, id pelanggan, id menu, dan jumlah pesanan. Dengan database ini, restoran dapat dengan mudah mengelola pemesanan makanan pelanggan.

1. Membuat Database dengan perintah Python

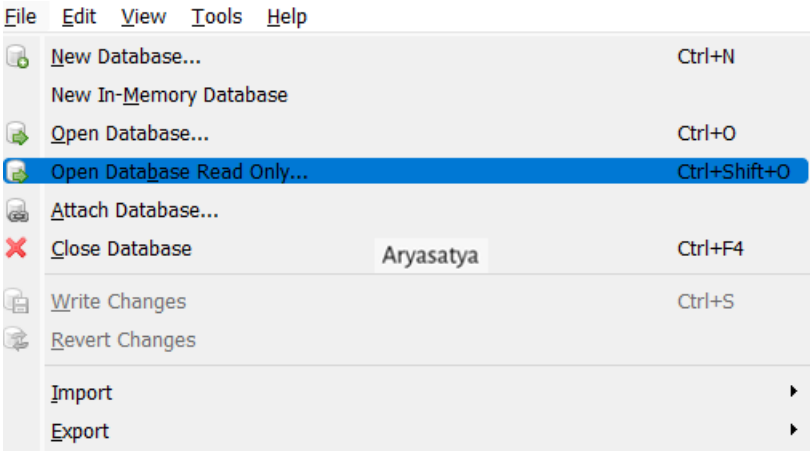
Kode script:

```
1 import sqlite3
2 con = sqlite3.connect("pemesanan_makanan.db", timeout=10)
3 print("Database pemesanan_makanan.db berhasil di buat")
4
5 con.close()
```


Kode tersebut membuat koneksi ke database SQLite dengan nama "pemesanan_makanan.db" menggunakan modul "sqlite3". Metode "connect" digunakan untuk membuat koneksi, dan waktu tunggu sebesar 10 detik diatur untuk koneksi tersebut. Setelah koneksi berhasil dibuat, skrip mencetak pesan yang menunjukkan bahwa database telah berhasil dibuat. Terakhir, metode "close" dipanggil untuk menutup koneksi ke database. Output jika berhasil dibuat:

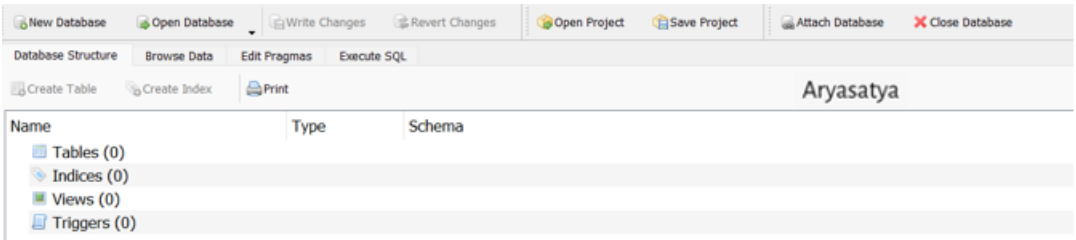
```
Database pemesanan_makanan.db berhasil di buat
```

Jika database berhasil dibuat, kita dapat mengecek langsung melalui aplikasi DB Browser for SQLite dan membuka databasenya melalui menu open file



Setelah kita klik yang bertanda biru diatas, kita akan diarahkan ke penyimpanan komputer dan buka database yang dibuat di dalam file directory tempat kita menyimpan kode script python

Jika file sudah ditemukan, dapat kita buka dan tampilan dalam aplikasi DB seperti berikut:



2. Buat tabel dan tambahkan data dengan perintah python

Membuat database pemesanan_makanan menggunakan bahasa pemrograman Python dan modul SQLite3. Proses pembuatan tabel dimulai dengan membuat koneksi ke database, Pengulangan for digunakan untuk mengeksekusi perintah yang terdapat dalam blok kode secara berulang, kemudian membuat objek cursor dan mengeksekusi perintah CREATE TABLE untuk membuat tabel. Setelah itu, koneksi ditutup dan perubahan pada database di-commit. Contoh pembuatan tabel:

Tabel Menu:

```
import sqlite3

con = sqlite3. connect ("pemesanan_makanan.db", timeout=10)
cur = con.cursor ()

for row in cur.execute("CREATE TABLE Menu (id_menu int, nama_menu varchar(
    print (row)

con.commit ()
con.close ()
```


Tabel Pelanggan:

```
import sqlite3

con = sqlite3.connect("pemesanan_makanan.db", timeout=10)
cur = con.cursor()

for row in cur.execute("CREATE TABLE Pelanggan (id_pelanggan int, nama_pel
    print(row)

con.commit()
con.close()
```


Tabel Pesanan:

```
import sqlite3

con = sqlite3.connect("pemesanan_makanan.db", timeout=10)
cur = con.cursor()

for row in cur.execute("CREATE TABLE Pesanan (id_pesanan int, id_pelanggan
    print(row)

con.commit()
con.close()
```


Setelah ketiga tabel tersebut berhasil ditambahkan, dapat dicek di aplikasi DB seperti berikut:

Name	Type	Schema
Tables (3)		
Menu		CREATE TABLE Menu (id_menu int, nama_menu varchar(50), deskripsi text, harga int(20), kategori varchar(20))
id_menu	int	"id_menu" int
nama_menu	varchar(50)	"nama_menu" varchar(50)
deskripsi	text	"deskripsi" text
harga	int(20)	"harga" int(20)
kategori	varchar(20)	"kategori" varchar(20)
Pelanggan		CREATE TABLE Pelanggan (id_pelanggan int, nama_pelanggan varchar(50), alamat text, no_telepon integer(20))
id_pelanggan	int	"id_pelanggan" int
nama_pelanggan	varchar(50)	"nama_pelanggan" varchar(50)
alamat	text	"alamat" text
no_telepon	integer(20)	"no_telepon" integer(20)
Pesanan		CREATE TABLE Pesanan (id_pesanan int, id_pelanggan int, id_menu int, jumlah_pesanan int)
id_pesanan	int	"id_pesanan" int
id_pelanggan	int	"id_pelanggan" int
id_menu	int	"id_menu" int
jumlah_pesanan	int	"jumlah_pesanan" int
Indices (0)		
Views (0)		
Triggers (0)		

Menambahkan data ke dalam tabel Menu, Pelanggan, Dan Pesanan:

Tabel Menu:

```
import sqlite3                                                    Aryasatya

con = sqlite3. connect("pemesanan_makanan.db", timeout=10)
cur = con. cursor()

for row in cur.execute ("INSERT INTO Menu VALUES (1, 'Nasi Goreng', 'Sebua
    print(row)
for row in cur.execute("INSERT INTO Menu VALUES (2, 'Es Teh Hangat', 'Teh
    print(row)

con.commit ()
con.close ()
```

Script di atas menambahkan dua buah data pada tabel Menu dalam database pemesanan_makanan.db menggunakan bahasa Python dan library sqlite3. Data pertama yang ditambahkan memiliki id_menu bernilai 1, nama_menu bernilai "Nasi Goreng", deskripsi bernilai "Sebuah nasi putih yang digoreng jadilah nasi goreng", harga sebesar 20000, dan kategori bernilai "Makanan". Sedangkan data kedua yang ditambahkan memiliki id_menu bernilai 2, nama_menu bernilai "Es Teh Hangat", deskripsi bernilai "Teh hangat dengan tambahan es batu", harga sebesar 5000, dan kategori bernilai "Minuman". Dengan menggunakan perintah INSERT INTO pada query SQL, data baru berhasil ditambahkan pada tabel Menu dan kemudian dikommit ke database.

Tabel Pelanggan:

```
import sqlite3
                                                                  Aryasatya
con = sqlite3. connect("pemesanan_makanan.db", timeout=10)
cur = con. cursor()

for row in cur.execute ("INSERT INTO Pelanggan VALUES (1, 'julian', 'Zuricl
    print(row)
for row in cur.execute("INSERT INTO Pelanggan VALUES (2, 'xavier', 'Davos
    print(row)

con.commit ()
con.close ()
```

Script di atas menambahkan data pelanggan ke dalam tabel Pelanggan pada database pemesanan_makanan. Pertama, dilakukan koneksi ke database menggunakan modul sqlite3, kemudian dibuat cursor yang akan digunakan untuk menjalankan query SQL. Setelah itu, dilakukan pengulangan menggunakan for untuk mengeksekusi perintah SQL INSERT INTO, yaitu menambahkan data pelanggan ke dalam tabel Pelanggan dengan nilai kolom id_pelanggan, nama_pelanggan, alamat_pelanggan, dan nomor_telepon_pelanggan. Data pelanggan yang ditambahkan adalah Julian dan Xavier. Terakhir, dilakukan commit untuk menyimpan perubahan dan menutup koneksi ke database.

Tabel Pesanan:

```python
import sqlite3

con = sqlite3. connect("pemesanan_makanan.db", timeout=10)
cur = con. cursor()

for row in cur.execute ("INSERT INTO Pesanan VALUES (1, 1, 1, 2)"):
    print(row)
for row in cur.execute("INSERT INTO Pesanan VALUES (2, 2, 2, 1)"):
    print(row)

con.commit ()
con.close ()
```


Kode di atas menambahkan data ke dalam tabel Pesanan pada database pemesanan_makanan.db dengan menggunakan perintah SQL INSERT INTO. Terdapat dua data yang ditambahkan pada tabel, yaitu pesanan pertama dengan id_pesanan = 1, id_pelanggan = 1, id_menu = 1, dan jumlah_pesanan = 2, serta pesanan kedua dengan id_pesanan = 2, id_pelanggan = 2, id_menu = 2, dan jumlah_pesanan = 1. Setelah data ditambahkan, dilakukan perintah con.commit() untuk menyimpan perubahan pada database dan perintah con.close() untuk menutup koneksi ke database.

Cek data inputan di aplikasi DB SQLite:

Setelah menjalankan script untuk menambahkan data ke dalam database, kita dapat membuka aplikasi SQLite dan melihat apakah data telah berhasil ditambahkan ke dalam tabel yang sesuai. Dengan membuka tabel-tabel yang telah dibuat, kita dapat melihat data yang telah berhasil dimasukkan dan memastikan bahwa data tersebut sesuai dengan yang diinginkan.

	id_menu	nama_menu	deskripsi	harga	kategori
	Filter	Filter	Filter	Filter	Filter
1	1	Nasi Goreng	Sebuah nasi putih yang digoreng ...	20000	Makanan
2	2	Es Teh Hangat	Teh hangat dengan tambahan es batu	5000	Minuman

	id_pelanggan	nama_pelanggan	alamat	no_telepon
	Filter	Filter	Filter	Filter
1	1	julian	Zurich Street 1	81234567890
2	2	xavier	Davos Street 1	89876543210

	id_pesanan	id_pelanggan	id_menu	jumlah_pesanan
	Filter	Filter	Filter	Filter
1	1	1	1	2
2	2	2	2	1

3. Pemutakhiran data pada tabel yang telah dibuat dengan python

Kode di atas menunjukkan contoh bagaimana metode UPDATE pada SQL dapat digunakan untuk memperbarui data yang sudah ada. Pada contoh tersebut, data pada kolom 'nama_pelanggan' pada tabel Pelanggan dengan id_pelanggan=1 diperbarui menjadi 'nana'. Setelah proses UPDATE selesai, data yang telah diubah akan tersimpan di dalam database SQLite. Kode Script:

```
import sqlite3

con = sqlite3.connect ("pemesanan_makanan.db", timeout=10)
cur = con. cursor ()

for row in cur.execute("UPDATE Pelanggan SET nama_pelanggan = 'nana' WHERE
    print(row)

con. commit()
con.close ()
```

Aryasatya

Selanjutnya, dilakukan pengecekan data pada tabel Pelanggan dengan menggunakan perintah SELECT pada bahasa SQL. Setiap baris pada tabel Pelanggan akan dicetak pada layar dengan perintah print(). Hasil pencetakan dapat dilihat pada output yang dihasilkan. Perintah commit() dilakukan untuk menyimpan perubahan data yang telah diambil. Terakhir, koneksi ke database ditutup dengan menggunakan perintah close(). Dengan cara ini, kita dapat memeriksa apakah data pada tabel Pelanggan sudah sesuai dengan hasil pemutakhiran yang telah dilakukan. Kode Script:

```
import sqlite3

con = sqlite3. connect ("pemesanan_makanan.db", timeout=10)
cur = con. cursor ()

for row in cur.execute("SELECT * FROM Pelanggan"):
    print(row)

con. commit ()
con.close ()
```

Aryasatya

Setelah pemutakhiran data nama Pelanggan, dapat kita cek bahwa perintah ini berhasil dengan melihat di aplikasi DB SQLite seperti berikut:

	id_pelanggan	nama_pelanggan	alamat	no_telepon
	Filter	Filter	Filter	Filter
1	1	nana	Zurich Street 1	81234567890
2	2	xavier	Davos Street 1	89876543210

Aryasatya

Dapat dilihat bahwa nama pelanggan 2 yang mulanya Julian, berhasil update menjadi nana.

4. Mengelompokkan data pada tabel yang telah ada dengan python.

Sebelum menggunakan fitur Group By dalam database, pertama-tama harus menambahkan data agar dapat dikelompokkan. Untuk melakukan pengelompokan, data harus memiliki kolom yang sama sebagai acuan. Dalam contoh di atas, kita menggunakan data pemesanan makanan yang terdiri dari

pelanggan, menu, dan pesanan, sehingga data ditambahkan ke masing-masing tabel. Setelah data ditambahkan, kita dapat menggunakan fitur Group By untuk mengelompokkan data berdasarkan kolom yang sama dan melakukan operasi yang diinginkan pada setiap kelompok data.

Menambahkan data menu:

```
import sqlite
                                                    Aryasatya

con = sqlite3.connect ("pemesanan_makanan.db", timeout=10)
cur = con. cursor ()

for row in cur.execute("INSERT INTO Menu VALUES (3, 'Ayam Geprek', 'Ayam y
    print (row)
for row in cur.execute("INSERT INTO Menu VALUES (4, 'Lemon Tea', 'Teh deng
    print (row)

con.commit ()
con. close ()
```

Menambahkan data pelanggan:

```
import sqlite3
                                                    Aryasatya

con = sqlite3. connect("pemesanan_makanan.db", timeout=10)
cur = con. cursor()

for row in cur.execute ("INSERT INTO Pelanggan VALUES (1, 'julian', 'Zuric
    print(row)
for row in cur.execute("INSERT INTO Pelanggan VALUES (2, 'xavier', 'Davos
    print(row)

con.commit ()
con.close ()
```

Menambahkan data Pesanan

```
import sqlite3
                                                    Aryasatya

con = sqlite3.connect ("pemesanan_makanan.db", timeout=10)
cur = con. cursor ()

for row in cur.execute("INSERT INTO Pesanan VALUES (3, 3, 3, 2)"):
    print (row)
for row in cur.execute("INSERT INTO Pesanan VALUES (4, 4, 4, 1)"):
    print (row)

con.commit ()
con. close ()
```

Selanjutnya tampilkan data dari tabel Menu di dalam database pemesanan_makanan.db, dengan melakukan pengurutan data secara menurun (descending) berdasarkan harga makanan atau minuman. Pengurutan dilakukan dengan menggunakan perintah ORDER BY pada kolom harga dan disertai dengan parameter DESC (descending) agar data diurutkan mulai dari harga tertinggi. Setelah data diambil, script akan menampilkan seluruh baris atau data yang ada di tabel Menu dengan menggunakan perintah for loop dan perintah print untuk menampilkan output ke layar. Terakhir, perintah commit() dan close() digunakan untuk menyimpan perubahan dan menutup koneksi ke database. Kode Script:

```python
import sqlite3

con = sqlite3. connect ("pemesanan_makanan.db", timeout=10)
cur = con.cursor ()

for row in cur.execute("SELECT * FROM Menu ORDER BY harga DESC"):
    print(row)

con.commit ()
con.close ()
```

Aryasatya

Sehingga output yang dihasilkan:

```
(1, 'Nasi Goreng', 'Sebuah nasi putih yang digoreng jadilah nasi goreng', 2000
0, 'Makanan')
(2, 'Es Teh Hangat', 'Teh hangat dengan tambahan es batu', 5000, 'Minuman')
```

Selanjutnya mengelompokkan table jenis kelamin dan menghitungnya menggunakan COUNT dan GROUP BY:

```python
#menampilkan data
import sqlite3
# Create a SQL connection to our SQLite database
con = sqlite3. connect ("penitipan.db", timeout=10)
cur = con.cursor ()
# The result of a "cursor.execute" can be iterated over by row

for row in cur.execute("SELECT COUNT (*), jenis_kelamin FROM identitas GRO
    print(row)
con.commit ()
# Be sure to close the connection
con.close ()
```

Aryasatya

Output:

```
(1, 'L')
(3, 'P')
```

Output dari kode tersebut adalah hasil pengelompokkan data pada tabel Pelanggan berdasarkan nama_pelanggan dan menghitung jumlah data yang memiliki nama_pelanggan yang sama dengan menggunakan fungsi COUNT(). Output yang dihasilkan berupa pasangan nilai (jumlah data, nama_pelanggan) yang diurutkan secara ascending berdasarkan nama_pelanggan. Dengan demikian, kita dapat melihat jumlah pelanggan yang memiliki nama yang sama dalam database kita.

5. Melakukan join data pada tabel yang telah dibuat dengan python

Dalam database, kita dapat menggunakan fitur join untuk menggabungkan dua tabel atau lebih berdasarkan kolom yang sama. Dalam contoh ini, kita akan melakukan join pada tabel yang telah dibuat sebelumnya dan menampilkan kolom yang spesifik dengan query join pDada bahasa pemrograman Python. Kode script yang dapat digunakan:

```python
import sqlite3

con = sqlite3.connect("pemesanan_makanan.db", timeout=10)
cur = con.cursor()

query = "SELECT Pesanan.id_pesanan, Pelanggan.nama_pelanggan, Pelanggan.no

for row in cur.execute(query):
    print(row)

con.close()
```

Aryasatya

Pada script di atas, dilakukan join antara tiga tabel yaitu Pesanan, Menu, dan Pelanggan menggunakan perintah JOIN. Query ini akan menampilkan kolom id_pesanan, nama_pelanggan, no_telepon, nama_menu, jumlah_pesanan, harga, dan total_harga, dengan menggunakan operasi perkalian untuk menghitung total harga dari setiap pesanan. Hasil join ini akan menampilkan semua data yang telah dihubungkan dari ketiga tabel tersebut. Output:

```
(1, 'nana', 81234567890, 'Nasi Goreng', 2, 20000, 40000)
(2, 'xavier', 89876543210, 'Es Teh Hangat', 1, 5000, 5000)
```

Output yang dihasilkan adalah tabel yang menunjukkan id_pesanan, nama_pelanggan, no_telepon, nama_menu, jumlah_pesanan, harga, dan total_harga untuk setiap pesanan. Lalu, kita dapat Menambah komen pada tampilan output dengan perintah sebagai berikut dengan kode script:

```python
import sqlite3
con = sqlite3.connect("penitipan.db", timeout=10)
cur = con.cursor()

rows = cur.execute("SELECT p.nama_barang, p.warna_barang, p.jumlah_barang,

for row in rows:

    nama_barang = row[0]
    warna_barang = row[1]
    jumlah_barang = row[2]
    tanggal_penitipan = row[3]
    nomor_karcis = row[4]
    nama_pemilik = row[5]
    jenis_kelamin = row[6]

    print("Nama barang:", nama_barang)
    print("Warna barang:", warna_barang)
    print("Jumlah barang:", jumlah_barang)
    print("Tanggal penitipan:", tanggal_penitipan)
    print("Nomor karcis:", nomor_karcis)
    print("Nama pemilik:", nama_pemilik)
    print("Jenis kelamin:", jenis_kelamin)
    print()

con.commit()
con.close()
```

Aryasatya

Script di atas merupakan contoh penggunaan join untuk menggabungkan data dari tabel Pesanan, Menu, dan Pelanggan dalam satu query. Kemudian, untuk menampilkan output yang lebih informatif, kita dapat menambahkan komentar atau keterangan pada setiap baris data menggunakan perintah print() dan menambahkan variabel yang sesuai dengan nilai kolom pada setiap baris data. Dalam contoh di atas, setiap baris data akan ditampilkan dengan informasi seperti ID Pesanan, Nama Menu, Nama Pelanggan, Alamat, Nomor Telepon, Jumlah Pesanan, Harga, dan Total Harga. Dengan

menambahkan komentar pada setiap baris data, kita dapat membuat tampilan output yang lebih informatif dan mudah dipahami oleh pengguna. Output yang dihasilkan:

```
ID Pesanan: 1
Nama Menu: Nasi Goreng
Nama Pelanggan: nana
Alamat: Zurich Street 1
Nomor Telepon: 81234567890
Jumlah Pesanan: 2
Harga: 20000
Total Harga: 40000
```

Aryasatya

```
ID Pesanan: 2
Nama Menu: Es Teh Hangat
Nama Pelanggan: xavier
Alamat: Davos Street 1
Nomor Telepon: 89876543210
Jumlah Pesanan: 1
Harga: 5000
Total Harga: 5000
```

6. Kesimpulan

Dalam tugas ini, kita telah mempelajari tentang database dan penggunaannya dalam bahasa pemrograman Python dengan menggunakan SQLite. Pertama-tama, kita membuat database dan tabel-tabel yang dibutuhkan untuk menyimpan data. Selanjutnya, kita melakukan beberapa operasi pada database seperti menambahkan data, menghapus data, dan memperbarui data. Selain itu, kita juga mempelajari tentang query SQL dan cara penggunaannya dalam bahasa pemrograman Python untuk menampilkan data secara spesifik dengan menggunakan beberapa metode seperti GROUP BY dan JOIN. Terakhir, kita juga belajar bagaimana menambahkan komentar pada output untuk memudahkan pembacaan. Dengan menguasai konsep-konsep ini, kita dapat memanfaatkan database secara lebih efektif dalam pengembangan aplikasi atau sistem yang kita kerjakan.