

TECHNICAL REPORT
PEMROGRAMAN DESKTOP
MODUL 5



Disusun Oleh :

TGL. PRAKTIKUM	: Jum'at, 11 Desember 2020
NAMA	: Achmad Farid Alfa Waid
NRP	: 190411100073
KELOMPOK	: 2
DOSEN	: Moh. Kautsar Sophan, S.Kom., M.MT.

TELAH DISETUJUI TANGGAL :

.....
ASISTEN PRAKTIKUM

Nadia Asri
(180411100063)

LABORATORIUM MULTIMEDIA COMPUTING
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA

BAB I

TUJUAN DAN DASAR TEORI

A. TUJUAN

Pada praktikum kali ini bertujuan untuk memahami tentang cara membuat database, table, menambahkan data, menampilkan data, dan mencari data pada database.

B. DASAR TEORI

Database berfungsi untuk menyimpan dan mengelompokkan suatu data berdasarkan identifikasi data yang sudah dibuat, meminimalisir terjadinya duplikasi data pada saat penyimpanan data, memudahkan pengguna dalam proses (akses, edit, tambah, dan delete)

BAB II

PEMBAHASAN

A. SOAL

1. Buat aplikasi yang terkoneksi ke database
2. Siapkan fungsi untuk
 - a) Membuat database, membuat tabel
 - b) Memasukkan data ke dalam tabel dengan perintah SQL
 - c) Menampilkan isi tabel
 - d) Mencari data dalam tabel

B. JAWABAN

1. modul 5.py

```
import sys
from PyQt5 import *
from PyQt5.QtWidgets import *
from PyQt5.Qt import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from PyQt5 import QSql
import sqlite3

class Film(QWidget):
    #Membuat fungsi init untuk inisialisasi class Mahasiswa
    def __init__(self):
        #untuk mengembalikan semua atribut dan method yang ada
        super().__init__()
        #membuka database
        self.OpenDatabase()
        self.createTabel()
        #memanggil fungsi Layout yang sudah dibuat agar ditampilkan hasilnya
        self.Layout()

    def OpenDatabase(self):
```

```

#Mendeklarasikan database
db = QSql.QSqlDatabase.addDatabase('QSQLITE')
#Membuat nama database
db.setDatabaseName('test.db')
#Mengecek Database Apakah sudah terkoneksi atau belum
if db.open():
    print('Berhasil membuka Database')
else:
    print('Gagal membuka Database!')

def createTabel(self):
    query = QSql.QSqlQuery()
    query.exec_("create table listfilm(" "id integer primary key
AUTOINCREMENT, "
    "judul varchar(20), tahun_rilis int(10), genre varchar(20),
review varchar(100), rating varchar(20))")
    print ("True")

def Layout(self):
    #Membuat Grid Layout
    grid = QGridLayout()

    #Membuat label, Line Edit, dan Button yang akan dimasukkan
ke dalam layout Grid
    open_database = QLabel("Buka Database")
    grid.addWidget(open_database,0,0)

    open_button = QPushButton("Open Database")
    grid.addWidget(open_button,0,1,1,2)

    add_data = QLabel("Tambah Data:")
    grid.addWidget(add_data,1,0,1,0)

    add_id = QLabel("ID:")
    grid.addWidget(add_id,2,0)

    self.Id = QLineEdit(self)
    grid.addWidget(self.Id,2,1,1,2)

    add_judul = QLabel("Judul:")

```

```
grid.addWidget(add_judul,3,0)

self.Judul = QLineEdit(self)
grid.addWidget(self.Judul,3,1,1,2)

tahun_rilis = QLabel("Tahun Rilis Film:")
grid.addWidget(tahun_rilis,4,0)

self.Tahun = QLineEdit(self)
grid.addWidget(self.Tahun,4,1,1,2)

add_genre = QLabel("Genre:")
grid.addWidget(add_genre,5,0)

self.Genre = QLineEdit(self)
grid.addWidget(self.Genre,5,1,1,2)

add_review = QLabel("Review:")
grid.addWidget(add_review,6,0)

self.Review = QLineEdit(self)
grid.addWidget(self.Review,6,1,1,2)

add_rating = QLabel("Rating:")
grid.addWidget(add_rating,7,0)

self.Rating = QLineEdit(self)
grid.addWidget(self.Rating,7,1,1,2)

add_button = QPushButton("Tambah Data")
grid.addWidget(add_button,8,0,1,0)

search = QLabel("Cari Data:")
grid.addWidget(search,9,0)

self.Cari = QLineEdit(self)
grid.addWidget(self.Cari,9,1)

search_button = QPushButton("Cari")
grid.addWidget(search_button,9,2)
```

```

        #Membuat widget table view yang diberi nama "Data" dan akan
        dimasukkan ke dalam layout Grid
        self.tableview = QTableView(self)
        self.tableview.setObjectName("Data")
        grid.addWidget(self.tableview,10,0,1,0)

        #Ketika button di klik akan memanggil fungsi masing - masing
        open_button.clicked.connect(self.tampilData)
        add_button.clicked.connect(self.tambahData)
        search_button.clicked.connect(self.filterData)

        #Layout grid di jadikan layout utama
        self.setLayout(grid)

#Fungsi untuk menampilkan data di dalam table
def tampilData(self):
    #Membuat Model
    model = QSqlQueryModel()
    #Mendefinisikan sql
    sql = "SELECT * FROM listfilm"
    #Mengeksekusi Model Query
    model.setQuery(sql)
    #Mengeset Data Model Ke table view
    self.tableview.setModel(model)
    #self.tableview.setWindowTitle(title)
    return self.tableview

#Fungsi tambah Data ke dalam tabel listfilm
def tambahData(self):
    #Mengambil Text inputan
    Id = str(self.Id.text())
    Judul = str(self.Judul.text())
    Tahun = str(self.Tahun.text())
    Genre = str(self.Genre.text())
    Review = str(self.Review.text())
    Rating = str(self.Rating.text())
    #Mendefinisikan Query
    query = QSqlQuery()
    #Menjalankan Perintah Sql

```

```

        query.prepare("INSERT INTO listfilm VALUES ('" + Id + "', '"
+ Judul + "', '" + Tahun + "', '" + Genre + "', '" + Review + "',
'" + Rating + "')")

        #Mengecek apakah query berjalan dengan baik
        if query.exec_():
            self.Id.setText("")
            self.Judul.setText("")
            self.Tahun.setText("")
            self.Genre.setText("")
            self.Review.setText("")
            self.Rating.setText("")
            #Menampilkan Data
            self.tampilData()
        else:
            #Apabila error akan menampilkan errornya ke dalam
            terminal

            print("Insert Error: ", query.lastError().text())

#Fungsi Filter Data
def filterData(self):
    #Membuat Model
    model = QSqlQueryModel()
    #Mengambil Inputan filter
    filter_search = str(self.Cari.text())
    sql = "SELECT * FROM listfilm WHERE id LIKE
'" + str(filter_search) + "%' OR judul LIKE '" + str(filter_search) + "%'
OR tahun_rilis LIKE '" + str(filter_search) + "%' OR genre LIKE
'" + str(filter_search) + "%'"
    self.Cari.setText("")
    #Mengeksekusi Model Query
    model.setQuery(sql)
    #Mengeset Data Model Ke table view
    self.tableview.setModel(model)
    #mengembalikan nilai table view
    return self.tableview

if __name__ == '__main__':
    #Inisialisasi pyqt
    app = QApplication(sys.argv)
    #mengatur style di window menjadi style fusion

```

```

app.setStyle("fusion")
#membuat variabel ex yang berisi class Film
ex = Film()

#Menentukan ukuran window dan title untuk menampilkan
ex.setGeometry(100,100,800,600)
#membuat judul window
ex.setWindowTitle("Database Sqlite in Pyqt5")
#menampilkan isi dari variabel ex
ex.show()
#membuat system exit
sys.exit(app.exec_())

```

2. a) Fungsi membuat database dan tabel

```

def OpenDatabase(self):
    #Mendeklarasikan database
    db = QSql.QSqlDatabase.addDatabase('QSQLITE')
    #Membuat nama database
    db.setDatabaseName('test.db')
    #Mengecek Database Apakah sudah terkoneksi atau belum
    if db.open():
        print('Berhasil membuka Database')
    else:
        print('Gagal membuka Database!')

def createTabel(self):
    query = QSql.QSqlQuery()
    query.exec_("create table listfilm(" "id integer primary key
AUTOINCREMENT, "
        "judul varchar(20), tahun_rilis int(10), genre varchar(20),
review varchar(100), rating varchar(20))")
    print ("True")

```

b) Fungsi memasukkan data ke dalam tabel

```

#Fungsi tambah Data ke dalam tabel listfilm
def tambahData(self):
    #Mengambil Text inputan
    Id = str(self.Id.text())
    Judul = str(self.Judul.text())
    Tahun = str(self.Tahun.text())
    Genre = str(self.Genre.text())
    Review = str(self.Review.text())

```



```

        Rating = str(self.Rating.text())
        #Mendefinisikan Query
        query = QSqlQuery()
        #Menjalankan Perintah Sql
        query.prepare("INSERT INTO listfilm VALUES ('" + Id + "', '"
+ Judul + "', '" + Tahun + "', '" + Genre + "', '" + Review + "',
'" + Rating + "')")
        #Mengecek apakah query berjalan dengan baik
        if query.exec_():
            self.Id.setText("")
            self.Judul.setText("")
            self.Tahun.setText("")
            self.Genre.setText("")
            self.Review.setText("")
            self.Rating.setText("")
            #Menampilkan Data
            self.tampilData()
        else:
            #Apabila error akan menampilkan errornya ke dalam
            terminal
            print("Insert Error: ", query.lastError().text())

```

c) Fungsi menampilkan isi tabel

```

#Fungsi untuk menampilkan data di dalam table
def tampilData(self):
    #Membuat Model
    model = QSqlQueryModel()
    #Mendefinisikan sql
    sql = "SELECT * FROM listfilm"
    #Mengeksekusi Model Query
    model.setQuery(sql)
    #Mengeset Data Model Ke table view
    self.tableview.setModel(model)
    #self.tableview.setWindowTitle(title)
    return self.tableview

```

d) Fungsi mencari data dalam tabel

```

#Fungsi Filter Data
def filterData(self):
    #Membuat Model
    model = QSqlQueryModel()
    #Mengambil Inputan filter

```

```

        filter_search = str(self.Cari.text())
        sql = "SELECT * FROM listfilm WHERE id LIKE
'%"+str(filter_search)+"%' OR judul LIKE
'%"+str(filter_search)+"%' OR tahun_rilis LIKE
'%"+str(filter_search)+"%' OR genre LIKE
'%"+str(filter_search)+"%'"
        self.Cari.setText("")
        #Mengeksekusi Model Query
        model.setQuery(sql)
        #Mengeset Data Model Ke table view
        self.tableview.setModel(model)
        #mengembalikan nilai table view
        return self.tableview

```

3. Penjelasan Kode Program

- *class Film(QWidget):*

#Membuat fungsi init untuk inisialisasi class Mahasiswa

def __init__(self):

#untuk mengembalikan semua atribut dan method yang ada

super().__init__()

#membuka database

self.OpenDatabase()

self.createTabel()

#memanggil fungsi Layout yang sudah dibuat agar ditampilkan hasilnya

self.Layout()

Membuat sebuah class Film dan fungsi init untuk menampilkan semua fungsi yang telah dibuat seperti fungsi OpenDatabase(), createTabel(), dan Layout()

- *def OpenDatabase(self):*

#Mendeklarasikan database

db = QSql.QSqlDatabase.addDatabase('QSQLITE')

#Membuat nama database

```

db.setDatabaseName('test.db')
#Mengecek Database Apakah sudah terkoneksi atau belum
if db.open():
    print('Berhasil membuka Database')
else:
    print('Gagal membuka Database!')

```

Membuat fungsi OpenDatabase yang bersi variable db untuk menambahkan database ke dalam aplikasi QSQLITE, dan memberi nama database dengan syntax setDatabaseName. Dan membuat sebuah kondisi dimana jika database berhasil dibuka . jika gagal.

- *def createTabel(self):*
query = QSql.QSqlQuery()
query.exec_("create table listfilm(" "id integer primary key
AUTOINCREMENT, "
"judul varchar(20), tahun_rilis int(10), genre varchar(20), review
varchar(100), rating varchar(20))")
print ("True")

Membuat fungsi createTabel yang bersi variable query untuk mendeklarasikan syntax QSqlQuery, dan menjalankan proses pembuatan table dengan syntax contoh di atas. Jika berhasil di terminal akan tampil tulisan True.

- *def Layout(self):*
#Membuat Grid Layout
grid = QGridLayout()

#Membuat label, Line Edit, dan Button yang akan dimasukkan ke
dalam layout Grid
open_database = QLabel("Buka Database")
grid.addWidget(open_database,0,0)

```
open_button = QPushButton("Open Database")
grid.addWidget(open_button,0,1,1,2)
```

```
add_data = QLabel("Tambah Data:")
grid.addWidget(add_data,1,0,1,0)
```

```
add_id = QLabel("ID:")
grid.addWidget(add_id,2,0)
```

```
self.Id = QLineEdit(self)
grid.addWidget(self.Id,2,1,1,2)
```

```
add_judul = QLabel("Judul:")
grid.addWidget(add_judul,3,0)
```

```
self.Judul = QLineEdit(self)
grid.addWidget(self.Judul,3,1,1,2)
```

```
tahun_rilis = QLabel("Tahun Rilis Film:")
grid.addWidget(tahun_rilis,4,0)
```

```
self.Tahun = QLineEdit(self)
grid.addWidget(self.Tahun,4,1,1,2)
```

```
add_genre = QLabel("Genre:")
grid.addWidget(add_genre,5,0)
```

```
self.Genre = QLineEdit(self)
grid.addWidget(self.Genre,5,1,1,2)
```

```
add_review = QLabel("Review:")  
grid.addWidget(add_review,6,0)
```

```
self.Review = QLineEdit(self)  
grid.addWidget(self.Review,6,1,1,2)
```

```
add_rating = QLabel("Rating:")  
grid.addWidget(add_rating,7,0)
```

```
self.Rating = QLineEdit(self)  
grid.addWidget(self.Rating,7,1,1,2)
```

```
add_button = QPushButton("Tambah Data")  
grid.addWidget(add_button,8,0,1,0)
```

```
search = QLabel("Cari Data:")  
grid.addWidget(search,9,0)
```

```
self.Cari = QLineEdit(self)  
grid.addWidget(self.Cari,9,1)
```

```
search_button = QPushButton("Cari")  
grid.addWidget(search_button,9,2)
```

#Membuat widget table view yang diberi nama "Data" dan akan dimasukkan ke dalam layout Grid

```
self.tableview = QTableView(self)  
self.tableview.setObjectName("Data")  
grid.addWidget(self.tableview,10,0,1,0)
```

Membuat fungsi layout yang di dalamnya berisi beberapa widget. Dan membuat sebuah layout grid yang dimasukkan ke dalam variable grid. Membuat label, Line Edit, dan Button dengan identifikasi variable yang berbeda dan akan dimasukkan ke dalam layout Grid dengan syntax addWidget, dan mengatur posisi masing – masing widget tersebut agar rapi dan tidak bertumpukan. Dan yang terakhir Membuat widget table view yang diberi nama "Data" dan akan dimasukkan ke dalam layout Grid.

- *#Ketika button di klik akan memanggil fungsi masing - masing*
open_button.clicked.connect(self.tampilData)
add_button.clicked.connect(self.tambahData)
search_button.clicked.connect(self.filterData)

#Layout grid di jadikan layout utama
self.setLayout(grid)

Membuat 3 signal dimana ketika masing – masing button di klik, akan memanggil fungsi yang sudah di set. Serta menjadikan layout grid menjadi layout utama pada window.

- *#Fungsi untuk menampilkan data di dalam table*
def tampilData(self):
#Membuat Model
model = QSqlQueryModel()
#Mendefinisikan sql
*sql = "SELECT * FROM listfilm"*
#Mengeksekusi Model Query
model.setQuery(sql)
#Mengeset Data Model Ke table view
self.tableview.setModel(model)
#self.tableview.setWindowTitle(title)
return self.tableview

Membuat fungsitampilData untuk menampilkan data yang ada di dalam database. Membuat variable model yang didalamnya berisi syntax untuk menggunakan query, membuat variable sql dan berisi syntax untuk menampilkan data dala table list film. Setelah itu mengeksekusi model query dan mengembalikan nilai tampilan pada tableview.

- *#Fungsi tambah Data ke dalam tabel listfilm*

```
def tambahData(self):
    #Mengambil Text inputan
    Id = str(self.Id.text())
    Judul = str(self.Judul.text())
    Tahun = str(self.Tahun.text())
    Genre = str(self.Genre.text())
    Review = str(self.Review.text())
    Rating = str(self.Rating.text())
    #Mendefinisikan Query
    query = QSql.QSqlQuery()
    #Menjalankan Perintah Sql
    query.prepare("INSERT INTO listfilm VALUES ('" + Id + "', '" +
Judul + "', '" + Tahun + "', '" + Genre + "', '" + Review + "', '" + Rating
+ "'")
    #Mengecek apakah query berjalan dengan baik
    if query.exec_():
        self.Id.setText("")
        self.Judul.setText("")
        self.Tahun.setText("")
        self.Genre.setText("")
        self.Review.setText("")
        self.Rating.setText("")
    #Menampilkan Data
    self.tampilData()
```

else:

#Apabila error akan menampilkan errornya ke dalam terminal

print("Insert Error: ", query.lastError().text())

Membuat fungsi `tambahData` untuk menambahkan data hasil inputan dari line edit ke dalam database. Dan membuat suatu kondisi dimana jika query berhasil dieksekusi value dalam line edit akan dikosongkan. Kemudian menampilkan data record terbaru ke dalam tableview dengan fungsi `tampilData`. Jika gagal maka akan menampilkan letak error ke dalam terminal.

- *#Fungsi Filter Data*

def filterData(self):

#Membuat Model

model = QSqlQueryModel()

#Mengambil Inputan filter

filter_search = str(self.Cari.text())

```
sql = "SELECT * FROM listfilm WHERE id LIKE
'" + str(filter_search) + "%' OR judul LIKE
'" + str(filter_search) + "%' OR tahun_rilis LIKE
'" + str(filter_search) + "%' OR genre LIKE
'" + str(filter_search) + "%'"
```

self.Cari.setText("")

#Mengeksekusi Model Query

model.setQuery(sql)

#Mengeset Data Model Ke table view

self.tableview.setModel(model)

#mengembalikan nilai table view

return self.tableview

Membuat fungsi `filterData` untuk mencari data record dengan cari menginputkan value ke dalam line edit, dimana value di dalam line edit akan diterima dan dimasukkan ke dalam variable `filter_search`.

Membuat variable sql yang berisi syntax query untuk mencari data yang diinputkan berdasarkan value dari filter_search. Dan kemudian akan ditampilkan ke dalam tableview.

- *if __name__ == '__main__':*
#Inisialisai pyqt
app = QApplication(sys.argv)
#mengatur style di window menjadi style fusion
app.setStyle("fusion")
#membuat variabel ex yang berisi class Film
ex = Film()

#Menentukan ukuran window dan title untuk menampilkan
ex.setGeometry(100,100,800,600)
#membuat judul window
ex.setWindowTitle("Database Sqlite in Pyqt5")
#menampilkan isi dari variabel ex
ex.show()
#membuat system exit
sys.exit(app.exec_())

Mendeklarasikan QApplication di dalam variable app, memasukkan value yang ada di class Film ke dalam variable ex, dan memberikan style fusion ke dalam variable app. Lalu mengset ukuran geometry window dan memberikan title pada window dengan syntax setTitle. Kemudian menampilkan variable tersebut dengan fungsi show(), dan membuat system exit.

4. Hasil Running Program

Tampilan awal

Database Sqlite in Pyqt5

Buka Database

Tambah Data:

ID:

Judul:

Tahun Rilis Film:

Genre:

Review:

Rating:

Cari Data:

	id	judul	tahun_rilis	genre	review	rating
--	----	-------	-------------	-------	--------	--------

Untuk menampilkan data record pada database, yaitu dengan cari menekan button Open Database, maka pada widget tableview akan menampilkan data yang ada seperti gambar berikut.

Database Sqlite in Pyqt5

Buka Database

Tambah Data:

ID:

Judul:

Tahun Rilis Film:

Genre:

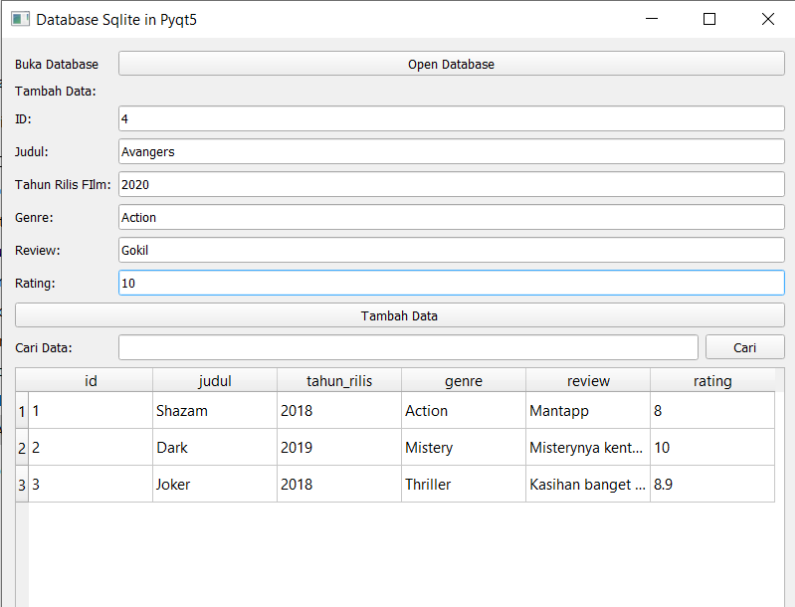
Review:

Rating:

Cari Data:

	id	judul	tahun_rilis	genre	review	rating
1	1	Shazam	2018	Action	Mantapp	8
2	2	Dark	2019	Mistery	Misterinya kent...	10
3	3	Joker	2018	Thriller	Kasihannya banget ...	8.9

Untuk menambahkan data ke dalam database harus mengisi pada beberapa line edit yang ada seperti berikut.



Buka Database

Tambah Data:

ID:

Judul:

Tahun Rilis Film:

Genre:

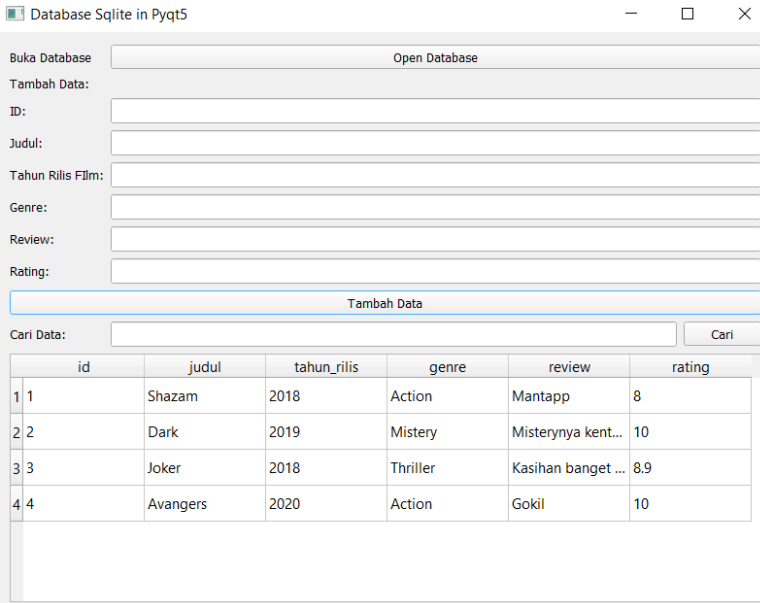
Review:

Rating:

Cari Data:

	id	judul	tahun_rilis	genre	review	rating
1	1	Shazam	2018	Action	Mantapp	8
2	2	Dark	2019	Mistery	Misterynya kent...	10
3	3	Joker	2018	Thriller	Kasihannya banget ...	8.9

Kemudian menekan button Tambah Data untuk menambahkan data yang diinputkan di line edit ke dalam database. Dan akan menampilkan data baru seperti gambar berikut.



Buka Database

Tambah Data:

ID:

Judul:

Tahun Rilis Film:

Genre:

Review:

Rating:

Cari Data:

	id	judul	tahun_rilis	genre	review	rating
1	1	Shazam	2018	Action	Mantapp	8
2	2	Dark	2019	Mistery	Misterynya kent...	10
3	3	Joker	2018	Thriller	Kasihannya banget ...	8.9
4	4	Avangers	2020	Action	Gokil	10

Untuk mencari data, kita haru menginputkan value yang akan dicari ke dalam line edit yang ada terlebih dahulu seperti gambar berikut.

Database Sqlite in Pyqt5

Buka Database

Tambah Data:

ID:

Judul:

Tahun Rilis Film:

Genre:

Review:

Rating:

Cari Data:

	id	judul	tahun_rilis	genre	review	rating
1	1	Shazam	2018	Action	Mantapp	8
2	2	Dark	2019	Mistery	Misterynya kent...	10
3	3	Joker	2018	Thriller	Kasihan banget ...	8.9
4	4	Avangers	2020	Action	Gokil	10

Kemudian kita harus menekan button Cari agar value yang yang kita inputkan di line edit dapat diproses, dan akan menampilkan data seperti gambar berikut.

Database Sqlite in Pyqt5

Buka Database

Tambah Data:

ID:

Judul:

Tahun Rilis Film:

Genre:

Review:

Rating:

Cari Data:

	id	judul	tahun_rilis	genre	review	rating
1	2	Dark	2019	Mistery	Misterynya kent...	10

BAB II

PENUTUP

A. Kesimpulan

1. Kita bisa mendapatkan value dari inputan user menggunakan line edit.
2. Dengan adanya signal dan slot dapat membuat aplikasi yang di buat berfungsi lebih baik.
3. Database sangat membantu user untuk menyimpan beberapa data, dan data tersbut bisa diakses kapan saja

B. Saran

Banyak mencoba dan mengeksplorasi widget yang lain agar lebih paham