

**TECHNICAL REPORT**  
**PEMROGRAMAN DESKTOP**  
**MODUL 6**



**Disusun Oleh :**

TGL. PRAKTIKUM	: Jum'at, 18 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 menghubungkan ke database Mysql online, menampilkan data, mencari data, menambahkan data, merubah data, dan menghapus data pada table yang ada di 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, searching, dan delete)

## **BAB II**

### **PEMBAHASAN**

#### **A. SOAL**

1. Buat koneksi database dengan rincian informasi berikut:
  - a) Host: kprikaryasehat.site
  - b) Username: kprikary\_kuliah
  - c) Password: unijoyo2020
  - d) Databasename: kprikary\_resto
  - e) Fokus di tabel : menu1
2. Buat form untuk:
  - a) Menampilkan isi menu1, dan mencari data
  - b) Menambahkan data di tabel tersebut
  - c) Merubah data
  - d) Menghapus data

#### **B. JAWABAN**

##### **1. modul 6.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
import MySQLdb as mdb
import pymysql.cursors

class Mysql(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()
        #memanggil fungsi Layout yang sudah dibuat agar ditampilkan
        hasilnya
        self.Layout()

    def OpenDatabase(self):
        try:
            self.db = mdb.connect('kprikaryasehat.site',
            'kprikary_kuliah', 'unijoyo2020', 'kprikary_resto')
            QMessageBox.about(self, 'Connection', 'Database
            Connected Successfully')

        except mdb.Error as e:
            QMessageBox.about(self, 'Connection', 'Failed To
            Connect Database')

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

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

        menu = QLabel("Nama Menu:")
        grid.addWidget(menu,2,0)

        self.Menu = QLineEdit(self)
        grid.addWidget(self.Menu,2,1,1,2)
        self.Menu.setStyleSheet("background-color: #f6f6f6;")

        keterangan = QLabel("Keterangan:")
        grid.addWidget(keterangan,3,0)

        self.Keterangan = QLineEdit(self)
        grid.addWidget(self.Keterangan,3,1,1,2)
        self.Keterangan.setStyleSheet("background-color:
        #f6f6f6;")

```

```
harga = QLabel("Harga:")
grid.addWidget(harga,4,0)

self.Harga = QLineEdit(self)
grid.addWidget(self.Harga,4,1,1,2)
self.Harga.setStyleSheet("background-color: #f6f6f6;")

satunya = QLabel("Satuan:")
grid.addWidget(satua,5,0)

self.Satua = QLineEdit(self)
grid.addWidget(self.Satua,5,1,1,2)
self.Satua.setStyleSheet("background-color: #f6f6f6;")

add_button = QPushButton("Tambah Data")
grid.addWidget(add_button,6,0,1,0)
add_button.setStyleSheet("background-color: #ffc7c7;")

edit_button = QPushButton("Edit Data")
grid.addWidget(edit_button,7,0,1,1)
edit_button.setStyleSheet("background-color: #ffc7c7;")

update_button = QPushButton("Update Data")
grid.addWidget(update_button,7,1,1,2)
update_button.setStyleSheet("background-color: #ffc7c7;")

search = QLabel("Cari Data:")
grid.addWidget(search,8,0)

self.Cari = QLineEdit(self)
grid.addWidget(self.Cari,8,1)
self.Cari.setStyleSheet("background-color: #f6f6f6;")

search_button = QPushButton("Cari")
grid.addWidget(search_button,8,2)
search_button.setStyleSheet("background-color: #ffc7c7;")

delete_button = QPushButton("Delete Data")
grid.addWidget(delete_button,9,0,1,0)
delete_button.setStyleSheet("background-color: #ffc7c7;")
```

```

        #Membuat widget table widget yang diberi nama "Data" dan
akan dimasukkan ke dalam layout Grid
        self.tablewidget = QTableWidgetItem(self)
        self.tablewidget.setObjectName("Data")
        self.tablewidget.setStyleSheet("background-color:
#f6f6f6;")
        grid.addWidget(self.tampilData(),11,0,5,0)

        #Ketika button di klik akan memanggil fungsi masing - masing
        add_button.clicked.connect(self.tambahData)
        search_button.clicked.connect(self.filterData)
        edit_button.clicked.connect(self.editData)
        update_button.clicked.connect(self.updateData)
        delete_button.clicked.connect(self.deleteData)

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

#Fungsi menampilkan Data menu1 ke dalam tabelwidget
def tampilData(self):
    # Membuat Cursor
    cur = self.db.cursor()
    # Mengeksekusi perintah sql
    cur.execute("SELECT * FROM menu1")
    # Mengambil Semua Data
    data = cur.fetchall()
    # Menjadikan Data bentuk list
    record = list(data)
    # Membuat Baris Yang Akan Ditampilkan DI table
    self.tablewidget.setRowCount(len(record)+1)
    # Membuat Kolom yang akan ditampilkan di table
    self.tablewidget.setColumnCount(8)
    # Membuat Header
    self.tablewidget.setItem(0, 0, QTableWidgetItem("ID Menu"))
    self.tablewidget.setItem(0, 1, QTableWidgetItem("ID Menu
Kat"))
    self.tablewidget.setItem(0, 2, QTableWidgetItem("ID
Resto"))

```

```

        self.tablewidget.setItem(0, 3, QTableWidgetItem("Nama
Menu"))
        self.tablewidget.setItem(0, 4,
QTableWidgetItem("Keterangan"))
        self.tablewidget.setItem(0, 5, QTableWidgetItem("Gambar"))
        self.tablewidget.setItem(0, 6, QTableWidgetItem("Harga"))
        self.tablewidget.setItem(0, 7, QTableWidgetItem("Satuan"))
        # Menampilkan Data Yang Diambil Dari SQL
        for i in range(len(record)):
            baris = i + 1
            self.tablewidget.setItem(baris, 0,
QTableWidgetItem(str(record[i][0])))
            self.tablewidget.setItem(baris, 1,
QTableWidgetItem(str(record[i][1])))
            self.tablewidget.setItem(baris, 2,
QTableWidgetItem(str(record[i][2])))
            self.tablewidget.setItem(baris, 3,
QTableWidgetItem(str(record[i][3])))
            self.tablewidget.setItem(baris, 4,
QTableWidgetItem(str(record[i][4])))
            self.tablewidget.setItem(baris, 5,
QTableWidgetItem(str(record[i][5])))
            self.tablewidget.setItem(baris, 6,
QTableWidgetItem(str(record[i][6])))
            self.tablewidget.setItem(baris, 7,
QTableWidgetItem(str(record[i][7])))
        # Membuat Agar Table Stretch

self.tablewidget.horizontalHeader().setStretchLastSection(True)

self.tablewidget.horizontalHeader().setSectionResizeMode(QHeaderVi
ew.Stretch)

        cur.close()
        return self.tablewidget

#Fungsi tambah Data ke dalam tabel menu1
def tambahData(self):
    # Mengambil Text inputan
    menu = str(self.Menu.text())
    keterangan = str(self.Keterangan.text())

```

```

        harga = int(self.Harga.text())
        satuan = str(self.Satuan.text())
        # Mendefinisikan Cursor
        cur = self.db.cursor()
        id = self.db.cursor()
        # Menjalankan Perintah SQL
        id.execute("SELECT idmenu FROM menu1 ORDER BY idmenu DESC
LIMIT 1")
        # Mengambil Satu Data
        idMenu = id.fetchone()
        # Membuat Id Auto Increment
        idMenu = idMenu[0] + 1
        # Menuliskan Perintah SQL
        sql = "INSERT INTO
menu1(idmenu,idmenukat,idresto,namamenu,keterangan,filegambar,harg
a,satuan) VALUES ('%d','%d','%d','%s','%s','%s','%d','%s') " %
(idMenu,1,5,menu,keterangan,'https://img-
global.cpcdn.com/recipes/84fae0149dbe9168/751x532cq70/telor-
balado-foto-resep-utama.jpg',harga,satuan)
        # Menjalankan Perintah Try
        try:
            # Eksekusi Perintah SQL
            cur.execute(sql)
            # Mengcommit agar perubahan Tersimpan
            self.db.commit()
            # Membuat Notifikasi Berhasil Tambah Data
            QMessageBox.about(self, 'Berhasil', 'Berhasil Menambah
Data')
            # Mengubah Inputan Agar Menjadi Kosong
            self.Menu.setText("")
            self.Keterangan.setText("")
            self.Harga.setText("")
            self.Satuan.setText("")
            # Menampilkan Data
            self.tampilData()
            id.close()
            cur.close()
        # Jika Terjadi Error
        except:
            # Merollback data

```



```

        self.db.rollback()
        print("Gagal")
#Fungsi edit Data tabel menu1
def editData(self):
    # Mengambil Data Yang Dipilih
    index = self.tablewidget.selectedIndexes()[0]
    id = self.tablewidget.model().data(index)
    # Membuat Cursor
    cur = self.db.cursor()
    # Perintah Mencari Data Berdasarkan ID
    sql = "SELECT * FROM menu1 WHERE idmenu = '%s' " % (id)
    # Eksekusi Perintah SQL
    cur.execute(sql)
    # Mengambil Satu Data
    data = cur.fetchone()
    # Menyimpan Data Id Di variabel idEdit
    self.idEdit = data[0]
    # Menampilkan Data Ke Textbox
    self.Menu.setText(data[3])
    self.Keterangan.setText(data[4])
    self.Harga.setText(str(data[6]))
    self.Satuan.setText(data[7])

# Fungsi Update Data menu1
def updateData(self):
    # Mengambil Data Dari Inputan
    menu = str(self.Menu.text())
    keterangan = str(self.Keterangan.text())
    harga = int(self.Harga.text())
    satuan = str(self.Satuan.text())
    # Membuat Cursor
    cur = self.db.cursor()
    # Perintah SQL Update
    sql = "UPDATE menu1 SET namamenu = '%s', keterangan = '%s',
harga = '%d', satuan = '%s' WHERE idmenu = '%d'" %
(menu,keterangan,harga,satuan,self.idEdit)
    try:
        # Eksekusi Perintah SQL
        cur.execute(sql)
        # Commit DB Agar Terjadi Perubahan DI database

```

```

        self.db.commit()
        # Membuat Message Box
        QMessageBox.about(self, 'Berhasil', 'Berhasil Update
Data')

        self.Menu.setText("")
        self.Keterangan.setText("")
        self.Harga.setText("")
        self.Satuan.setText("")
        self.tampilData()
        print("Berhasil")
    except:
        self.db.rollback()
        print("Gagal")

# Fungsi Hapus Data pada table menu1
def deleteData(self):
    # Mengambil Data Yang Di pilih
    index = self.tablewidget.selectedIndexes()[0]
    id = self.tablewidget.model().data(index)
    # Membuat Cursor
    cur = self.db.cursor()
    # Perintah SQL Untuk Menghapus Data
    sql = "DELETE FROM menu1 WHERE idmenu = '%s' " % (id)
    try:
        # Eksekusi Perintah SQL
        cur.execute(sql)
        # Commit Ke Database Agar Perubahan Tersimpan
        self.db.commit()
        # Menampilkan Data
        self.tampilData()
        print("Berhasil")
    except:
        # Merollback data
        self.db.rollback()
        print("Gagal")

#Fungsi serching Data yang ada di dalam tabel menu1
def filterData(self):
    # Membuat Cursor
    cur = self.db.cursor()

```

```

        filter_search = str(self.Cari.text())
        # Mengeksekusi perintah sql
        cur.execute("SELECT * FROM menu1 WHERE namamenu LIKE
'%" + str(filter_search) + "%' OR keterangan LIKE
'%" + str(filter_search) + "%' OR harga LIKE '%" + str(filter_search) + "%'
OR satuan LIKE '%" + str(filter_search) + "%'")
        # Mengambil Semua Data
        data = cur.fetchall()
        # Menjadikan Data bentuk list
        record = list(data)
        # Membuat Baris Yang Akan Ditampilkan DI table
        self.tablewidget.setRowCount(len(record)+1)
        # Membuat Kolom yang akan ditampilkan di table
        self.tablewidget.setColumnCount(8)
        # Membuat Header
        self.tablewidget.setItem(0, 0, QTableWidgetItem("ID Menu"))
        self.tablewidget.setItem(0, 1, QTableWidgetItem("ID Menu
Kat"))
        self.tablewidget.setItem(0, 2, QTableWidgetItem("ID
Resto"))
        self.tablewidget.setItem(0, 3, QTableWidgetItem("Nama
Menu"))
        self.tablewidget.setItem(0, 4,
QTableWidgetItem("Keterangan"))
        self.tablewidget.setItem(0, 5, QTableWidgetItem("Gambar"))
        self.tablewidget.setItem(0, 6, QTableWidgetItem("Harga"))
        self.tablewidget.setItem(0, 7, QTableWidgetItem("Satuan"))
        # Menampilkan Data Yang Diambil Dari SQL
        for i in range(len(record)):
            baris = i + 1
            self.tablewidget.setItem(baris, 0,
QTableWidgetItem(str(record[i][0])))
            self.tablewidget.setItem(baris, 1,
QTableWidgetItem(str(record[i][1])))
            self.tablewidget.setItem(baris, 2,
QTableWidgetItem(str(record[i][2])))
            self.tablewidget.setItem(baris, 3,
QTableWidgetItem(str(record[i][3])))
            self.tablewidget.setItem(baris, 4,
QTableWidgetItem(str(record[i][4])))

```

```

        self.tablewidget.setItem(baris,
                                QTableWidgetItem(str(record[i][5])))
        self.tablewidget.setItem(baris,
                                QTableWidgetItem(str(record[i][6])))
        self.tablewidget.setItem(baris,
                                QTableWidgetItem(str(record[i][7])))
        # Membuat Agar Table Stretch

self.tablewidget.horizontalHeader().setStretchLastSection(True)

self.tablewidget.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)

        cur.close()
        return self.tablewidget

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 FormulaMath
    ex = Mysql()
    ex.setStyleSheet("background-color: #fcf1f1;")
    ye = QPushButton()

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

```

## 2. Penjelasan Kode Program

- *class Mysql(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()
        #memanggil fungsi Layout yang sudah dibuat agar ditampilkan
        hasilnya
        self.Layout()

```

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

- *def OpenDatabase(self):*  
     *try:*  
         *self.db = mdb.connect('kprikaryasehat.site', 'kprikary\_kuliah',*  
         *'unijoyo2020', 'kprikary\_resto')*  
         *QMessageBox.about(self, 'Connection', 'Database Connected*  
         *Successfully')*

```

        except mdb.Error as e:
            QMessageBox.about(self, 'Connection', 'Failed To Connect
            Database')

```

Membuat fungsi OpenDatabase yang bersi variable db untuk menghubungkan database, jika berhasil masuk maka akan menampilkan Qessagebox berhasil, namun jika gagal akan menampilkan QMessageBox gagal.

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

```

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

```

```

        menu = QLabel("Nama Menu:")

```

```
grid.addWidget(menu,2,0)
```

```
self.Menu = QLineEdit(self)
```

```
grid.addWidget(self.Menu,2,1,1,2)
```

```
self.Menu.setStyleSheet("background-color: #f6f6f6;")
```

```
keterangan = QLabel("Keterangan:")
```

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

```
self.Keterangan = QLineEdit(self)
```

```
grid.addWidget(self.Keterangan,3,1,1,2)
```

```
self.Keterangan.setStyleSheet("background-color: #f6f6f6;")
```

```
harga = QLabel("Harga:")
```

```
grid.addWidget(harga,4,0)
```

```
self.Harga = QLineEdit(self)
```

```
grid.addWidget(self.Harga,4,1,1,2)
```

```
self.Harga.setStyleSheet("background-color: #f6f6f6;")
```

```
satuan = QLabel("Satuan:")
```

```
grid.addWidget(satuan,5,0)
```

```
self.Satuan = QLineEdit(self)
```

```
grid.addWidget(self.Satuan,5,1,1,2)
```

```
self.Satuan.setStyleSheet("background-color: #f6f6f6;")
```

```
add_button = QPushButton("Tambah Data")
```

```
grid.addWidget(add_button,6,0,1,0)
```

```
add_button.setStyleSheet("background-color: #ffc7c7;")
```

```
edit_button = QPushButton("Edit Data")
grid.addWidget(edit_button,7,0,1,1)
edit_button.setStyleSheet("background-color: #ffc7c7;")
```

```
update_button = QPushButton("Update Data")
grid.addWidget(update_button,7,1,1,2)
update_button.setStyleSheet("background-color: #ffc7c7;")
```

```
search = QLabel("Cari Data:")
grid.addWidget(search,8,0)
```

```
self.Cari = QLineEdit(self)
grid.addWidget(self.Cari,8,1)
self.Cari.setStyleSheet("background-color: #f6f6f6;")
```

```
search_button = QPushButton("Cari")
grid.addWidget(search_button,8,2)
search_button.setStyleSheet("background-color: #ffc7c7;")
```

```
delete_button = QPushButton("Delete Data")
grid.addWidget(delete_button,9,0,1,0)
delete_button.setStyleSheet("background-color: #ffc7c7;")
```

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

```
self.tablewidget = QTableWidget(self)
self.tablewidget.setObjectName("Data")
self.tablewidget.setStyleSheet("background-color: #f6f6f6;")
grid.addWidget(self.tampilData(),11,0,5,0)
```

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 tablewidget yang diberi nama "Data" dan akan dimasukkan ke dalam layout Grid.

- *#Ketika button di klik akan memanggil fungsi masing - masing*

```
add_button.clicked.connect(self.tambahData)
search_button.clicked.connect(self.filterData)
edit_button.clicked.connect(self.editData)
update_button.clicked.connect(self.updateData)
delete_button.clicked.connect(self.deleteData)
```

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

Membuat 5 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 menampilkan Data menu1 ke dalam tabelwidget*

```
def tampilData(self):
    # Membuat Cursor
    cur = self.db.cursor()
    # Mengeksekusi perintah sql
    cur.execute("SELECT * FROM menu1")
    # Mengambil Semua Data
    data = cur.fetchall()
    # Menjadikan Data bentuk list
    record = list(data)
    # Membuat Baris Yang Akan Ditampilkan DI table
```



```

self.tablewidget.setRowCount(len(record)+1)
# Membuat Kolom yang akan ditampilkan di table
self.tablewidget.setColumnCount(8)
# Membuat Header
self.tablewidget.setItem(0, 0, QTableWidgetItem("ID Menu"))
self.tablewidget.setItem(0, 1, QTableWidgetItem("ID Menu
Kat"))
self.tablewidget.setItem(0, 2, QTableWidgetItem("ID Resto"))
self.tablewidget.setItem(0, 3, QTableWidgetItem("Nama Menu"))
self.tablewidget.setItem(0, 4, QTableWidgetItem("Keterangan"))
self.tablewidget.setItem(0, 5, QTableWidgetItem("Gambar"))
self.tablewidget.setItem(0, 6, QTableWidgetItem("Harga"))
self.tablewidget.setItem(0, 7, QTableWidgetItem("Satuan"))
# Menampilkan Data Yang Diambil Dari SQL
for i in range(len(record)):
    baris = i + 1
    self.tablewidget.setItem(baris, 0,
QTableWidgetItem(str(record[i][0])))
    self.tablewidget.setItem(baris, 1,
QTableWidgetItem(str(record[i][1])))
    self.tablewidget.setItem(baris, 2,
QTableWidgetItem(str(record[i][2])))
    self.tablewidget.setItem(baris, 3,
QTableWidgetItem(str(record[i][3])))
    self.tablewidget.setItem(baris, 4,
QTableWidgetItem(str(record[i][4])))
    self.tablewidget.setItem(baris, 5,
QTableWidgetItem(str(record[i][5])))
    self.tablewidget.setItem(baris, 6,
QTableWidgetItem(str(record[i][6])))

```

```

        self.tablewidget.setItem(baris,
7,
QTableWidgetItem(str(record[i][7])))
        # Membuat Agar Table Stretch
        self.tablewidget.horizontalHeader().setStretchLastSection(True)

self.tablewidget.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)
        cur.close()
        return self.tablewidget

```

Membuat fungsitampilData untuk menampilkan data yang ada di dalam database. Membuat variable cur yang didalamnya berisi syntax untuk db.cursor, dan kemudia mengeksekusi perintah sql dan dimasukkan ke dalam variable data, dan nantinya akan dilakukan perulangan untuk menampilkan data – data yang ada di dalam table, dan dimasukkan ke dalam tablewidget.

- #Fungsi tambah Data ke dalam tabel menu1
 

```

def tambahData(self):
    # Mengambil Text inputan
    menu = str(self.Menu.text())
    keterangan = str(self.Keterangan.text())
    harga = int(self.Harga.text())
    satuan = str(self.Satuan.text())
    # Mendefinisikan Cursor
    cur = self.db.cursor()
    id = self.db.cursor()
    # Menjalankan Perintah SQL
    id.execute("SELECT idmenu FROM menu1 ORDER BY idmenu
DESC LIMIT 1")
    # Mengambil Satu Data
    idMenu = id.fetchone()

```

```

# Membuat Id Auto Increment
idMenu = idMenu[0] + 1
# Menuliskan Perintah SQL
sql = "INSERT INTO
menu1(idmenu,idmenukat,idresto,namamenu,keterangan,filegambar,h
arga,satuan) VALUES ('%d','%d','%d','%s','%s','%s','%d','%s')" %
(idMenu, 1, 5,menu,keterangan,'https://img-
global.cpcdn.com/recipes/84fae0149dbe9168/751x532cq70/telor-
balado-foto-resep-utama.jpg',harga,satuan)
# Menjalankan Perintah Try
try:
    # Eksekusi Perintah SQL
    cur.execute(sql)
    # Mengcommit agar perubahan Tersimpan
    self.db.commit()
    # Membuat Notifikasi Berhasil Tambah Data
    QMessageBox.about(self, 'Berhasil', 'Berhasil Menambah
Data')
    # Mengubah Inputan Agar Menjadi Kosong
    self.Menu.setText("")
    self.Keterangan.setText("")
    self.Harga.setText("")
    self.Satuan.setText("")
    # Menampilkan Data
    self.tampilData()
    id.close()
    cur.close()
# Jika Terjadi Error
except:
    # Merollback data

```

```
self.db.rollback()
print("Gagal")
```

Membuat fungsi `tambahData` untuk menambahkan data hasil inputan dari line edit kedalam database. Dan membuat suatu kondisi dimana jika query berhasil dieksekusi value dalam line edit akan dikosongkan. Kembali kemudian menampilkan data record terbaru kedalam `tablewidget` dengan fungsi `tampilData`. Jika gagal maka akan mengeluarkan output `Gagal`.

- *#Fungsi edit Data tabel menu1*

```
def editData(self):
    # Mengambil Data Yang Dipilih
    index = self.tablewidget.selectedIndexes()[0]
    id = self.tablewidget.model().data(index)
    # Membuat Cursor
    cur = self.db.cursor()
    # Perintah Mencari Data Berdasarkan ID
    sql = "SELECT * FROM menu1 WHERE idmenu = '%s' " % (id)
    # Eksekusi Perintah SQL
    cur.execute(sql)
    # Mengambil Satu Data
    data = cur.fetchone()
    # Menyimpan Data Id Di variabel idEdit
    self.idEdit = data[0]
    # Menampilkan Data Ke Textbox
    self.Menu.setText(data[3])
    self.Keterangan.setText(data[4])
    self.Harga.setText(str(data[6]))
    self.Satuan.setText(data[7])

# Fungsi Update Data menu1
```

```

def updateData(self):
    # Mengambil Data Dari Inputan
    menu = str(self.Menu.text())
    keterangan = str(self.Keterangan.text())
    harga = int(self.Harga.text())
    satuan = str(self.Satuan.text())
    # Membuat Cursor
    cur = self.db.cursor()
    # Perintah SQL Update
    sql = "UPDATE menu1 SET namamenu = '%s', keterangan = '%s',
harga = '%d', satuan = '%s' WHERE idmenu = '%d'" %
(menu,keterangan,harga,satuan,self.idEdit)
    try:
        # Eksekusi Perintah SQL
        cur.execute(sql)
        # Commit DB Agar Terjadi Perubahan DI database
        self.db.commit()
        # Membuat Message Box
        QMessageBox.about(self, 'Berhasil', 'Berhasil Update Data')
        self.Menu.setText("")
        self.Keterangan.setText("")
        self.Harga.setText("")
        self.Satuan.setText("")
        self.tampilData()
        print("Berhasil")
    except:
        self.db.rollback()
        print("Gagal")

```

Membuat fungsi editdata dan update data untuk merubah data yang ada pada table menu 1, dengan cara memilih data record mana yang mau di

update kemudian setelah dipilih data record tersebut akan ditampilkan di line edit, selanjutnya jika sudah diedit maka akan masuk ke fungsi updatedata, dengan membuat variable cur yang berisi db.cursor kemudian mengeksekusi perintah sql. Jika berhasil akan menampilkan QMessageBox berhasil dan mengosongkan lineedit Kembali, namun jika gagal akan megoutputkan gagal pada terminal.

- # Fungsi Hapus Data pada table menu1

```
def deleteData(self):  
    # Mengambil Data Yang Di pilih  
    index = self.tablewidget.selectedIndexes()[0]  
    id = self.tablewidget.model().data(index)  
    # Membuat Cursor  
    cur = self.db.cursor()  
    # Perintah SQL Untuk Menghapus Data  
    sql = "DELETE FROM menu1 WHERE idmenu = '%s' " % (id)  
    try:  
        # Eksekusi Perintah SQL  
        cur.execute(sql)  
        # Commit Ke Database Agar Perubahan Tersimpan  
        self.db.commit()  
        # Menampilkan Data  
        self.tampilData()  
        print("Berhasil")  
    except:  
        # Merollback data  
        self.db.rollback()  
        print("Gagal")
```

Pertama memilih data record mana yang mau dihapus, kemudian mengeksekusi perintah sql pada data record yang dipilih, jika berhasil

akan menampilkan Berhasil pada terminal, namun jika gagal akan menampilkan Gagal pada terminal.

- *#Fungsi serching Data yang ada di dalam tabel menu1*

```
def filterData(self):
```

```
    # Membuat Cursor
```

```
    cur = self.db.cursor()
```

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

```
    # Mengeksekusi perintah sql
```

```
    cur.execute("SELECT * FROM menu1 WHERE namamenu LIKE
```

```
    '%"+str(filter_search)+"%'          OR      keterangan      LIKE
```

```
    '%"+str(filter_search)+"%'          OR      harga          LIKE
```

```
    '%"+str(filter_search)+"%'          OR      satuan          LIKE
```

```
    '%"+str(filter_search)+"%")
```

```
    # Mengambil Semua Data
```

```
    data = cur.fetchall()
```

```
    # Menjadikan Data bentuk list
```

```
    record = list(data)
```

```
    # Membuat Baris Yang Akan Ditampilkan DI table
```

```
    self.tablewidget.setRowCount(len(record)+1)
```

```
    # Membuat Kolom yang akan ditampilkan di table
```

```
    self.tablewidget.setColumnCount(8)
```

```
    # Membuat Header
```

```
    self.tablewidget.setItem(0, 0, QTableWidgetItem("ID Menu"))
```

```
    self.tablewidget.setItem(0, 1, QTableWidgetItem("ID Menu Kat"))
```

```
    self.tablewidget.setItem(0, 2, QTableWidgetItem("ID Resto"))
```

```
    self.tablewidget.setItem(0, 3, QTableWidgetItem("Nama Menu"))
```

```
    self.tablewidget.setItem(0, 4, QTableWidgetItem("Keterangan"))
```

```
    self.tablewidget.setItem(0, 5, QTableWidgetItem("Gambar"))
```

```
    self.tablewidget.setItem(0, 6, QTableWidgetItem("Harga"))
```

```
    self.tablewidget.setItem(0, 7, QTableWidgetItem("Satuan"))
```

```

# Menampilkan Data Yang Diambil Dari SQL
for i in range(len(record)):
    baris = i + 1
    self.tablewidget.setItem(baris, 0,
QTableWidgetItem(str(record[i][0])))
    self.tablewidget.setItem(baris, 1,
QTableWidgetItem(str(record[i][1])))
    self.tablewidget.setItem(baris, 2,
QTableWidgetItem(str(record[i][2])))
    self.tablewidget.setItem(baris, 3,
QTableWidgetItem(str(record[i][3])))
    self.tablewidget.setItem(baris, 4,
QTableWidgetItem(str(record[i][4])))
    self.tablewidget.setItem(baris, 5,
QTableWidgetItem(str(record[i][5])))
    self.tablewidget.setItem(baris, 6,
QTableWidgetItem(str(record[i][6])))
    self.tablewidget.setItem(baris, 7,
QTableWidgetItem(str(record[i][7])))
# Membuat Agar Table Stretch
self.tablewidget.horizontalHeader().setStretchLastSection(True)

self.tablewidget.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)

cur.close()

return self.tablewidget

```

Membuat fungsi filterData untuk mencari data record dengan cara menginputkan value ke dalam line edit, dimana value di dalam line edit akan diterima dan dimasukkan ke dalam variable filter\_search. Kemudian mengeksekusi perintah sql dan akan menampilkan data



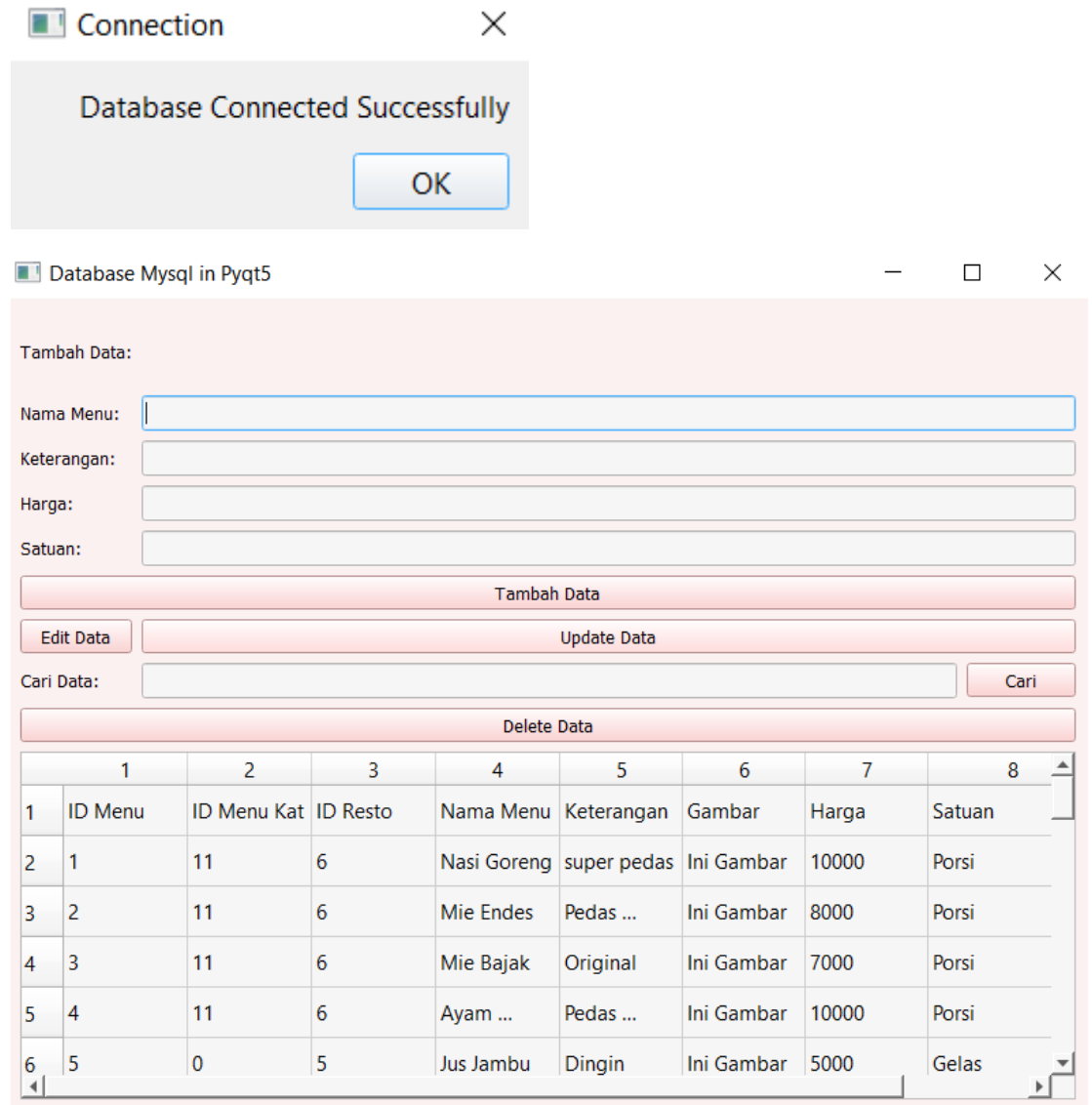
pada tablewidget sesuai value variable filter\_search. Dan akan ditampilkan dengan syntax yang sama seperti fungsi tampildata.

- *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 FormulaMath*  
*ex = Mysql()*  
*ex.setStyleSheet("background-color: #fcf1f1;")*  
*ye = QPushButton()*  
  
*#Menentukan ukuran window dan title untuk menampilkan*  
*ex.setGeometry(100,100,800,600)*  
*#membuat judul window*  
*ex.setWindowTitle("Database Mysql 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.

### 3. Hasil Running Program

Tampilan awal



Connection

Database Connected Successfully

OK

Database Mysql in Pyqt5

Tambah Data:

Nama Menu:

Keterangan:

Harga:

Satuan:

Tambah Data

Edit Data Update Data

Cari Data:  Cari

Delete Data

	1	2	3	4	5	6	7	8
1	ID Menu	ID Menu Kat	ID Resto	Nama Menu	Keterangan	Gambar	Harga	Satuan
2	1	11	6	Nasi Goreng	super pedas	Ini Gambar	10000	Porsi
3	2	11	6	Mie Endes	Pedas ...	Ini Gambar	8000	Porsi
4	3	11	6	Mie Bajak	Original	Ini Gambar	7000	Porsi
5	4	11	6	Ayam ...	Pedas ...	Ini Gambar	10000	Porsi
6	5	0	5	Jus Jambu	Dingin	Ini Gambar	5000	Gelas

Untuk menambah data pada table database, yaitu dengan cara mengisi line edit dan menekan tombol tambah data.

Database Mysql in Pyqt5

Tambah Data:

Nama Menu: kripik tempe

Keterangan: camilan khas malang

Harga: 5000

Satuan: porsi

Edit Data

Cari Data:

Cari

Tambah Data

Berhasil

Berhasil Menambah Data

OK

	1	2	3	4	5	6	7	8
	ID Menu	ID Menu Kat	ID Resto	Nama Menu	Keterangan	Gambar	Harga	Satuan
1	1	11	6	Nasi Goreng	super pedas	Ini Gambar	10000	Porsi
2	2	11	6	Mie Endes	Pedas ...	Ini Gambar	8000	Porsi
3	3	11	6	Mie Bajak	Original	Ini Gambar	7000	Porsi
4	4	11	6	Ayam ...	Pedas ...	Ini Gambar	10000	Porsi
5	5	0	5	Jus Jambu	Dingin	Ini Gambar	5000	Gelas

Untuk mengaupdate data, yaitu dengan cara memilih salah satu data records, dan kemudian merubah data yang diperlukan di line edit dan menekan tombol update data.

Database Mysql in Pyqt5

Tambah Data:

Nama Menu: kripik tempe

Keterangan: camilan khas malang

Harga: 8000

Satuan: porsi

Edit Data

Cari Data:

Cari

Tambah Data

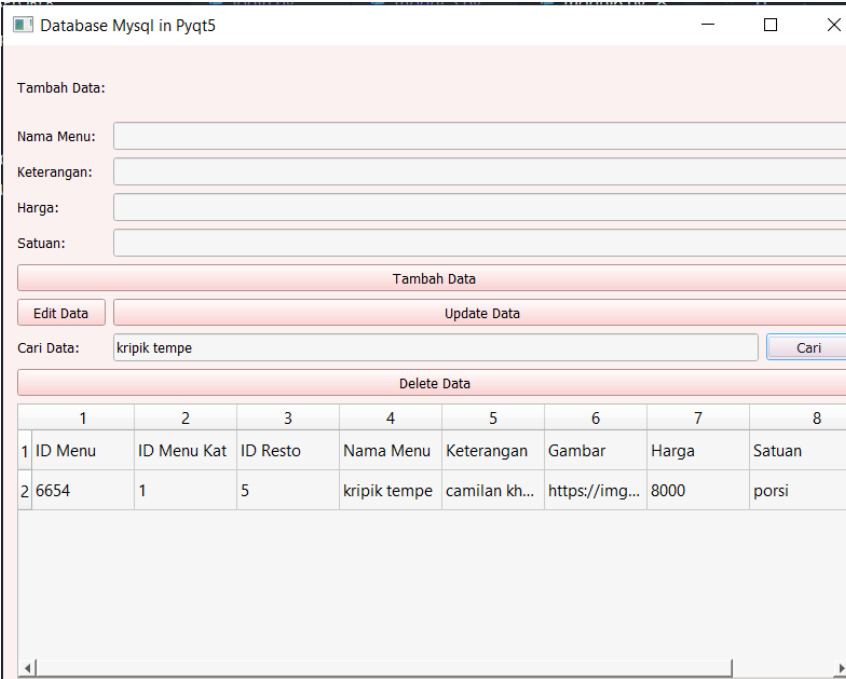
Berhasil

Berhasil Update Data

OK

	1	2	3	4	5	6	7	8
	ID Menu	ID Menu Kat	ID Resto	Nama Menu	Keterangan	Gambar	Harga	Satuan
168	6644	5	6	Jus Nanas	Jus	https://...	50000	gelas
169	6645	5	6	Jus salak	jus	https://...	50000	gelas
170	6646	5	6	Jus Timun	jus	https://...	50000	gelas
171	6650	5	6	Jus Naga	Jus Naga	jus naga.jpg	19500	Gelas
172	6653	1	3	Pecel Jamur	Pecel ...	976-...	7000	Porsi
173	6654	1	5	kripik tempe	camilan ...	https://im...	5000	porsi

Selanjutnya untuk mencari data, kita masukkan keyword data yang dicari dan menekan tombol cari, lalu pada table widget akan menampilkan data sesuai keyword yang akan dicari.



Tambah Data:

Nama Menu:

Keterangan:

Harga:

Satuan:

Tambah Data

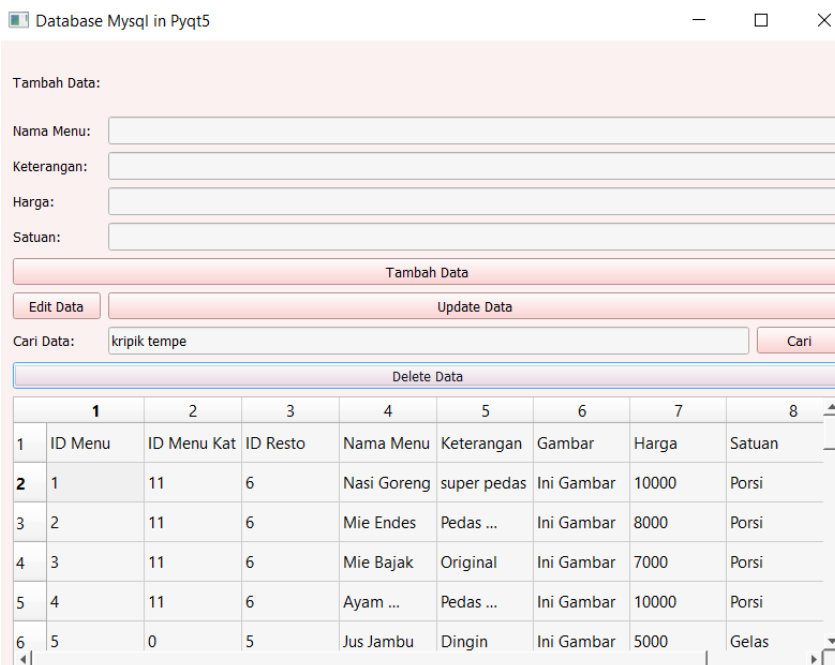
Edit Data Update Data

Cari Data:  Cari

Delete Data

	1	2	3	4	5	6	7	8
1	ID Menu	ID Menu Kat	ID Resto	Nama Menu	Keterangan	Gambar	Harga	Satuan
2	6654	1	5	kripik tempe	camilan kh...	https://img...	8000	porsi

Untuk menghapus data yaitu memilih data record yang akan dihapus pada table widget dan menekan tombol delete data.



Tambah Data:

Nama Menu:

Keterangan:

Harga:

Satuan:

Tambah Data

Edit Data Update Data

Cari Data:  Cari

Delete Data

	1	2	3	4	5	6	7	8
1	ID Menu	ID Menu Kat	ID Resto	Nama Menu	Keterangan	Gambar	Harga	Satuan
2	1	11	6	Nasi Goreng	super pedas	Ini Gambar	10000	Porsi
3	2	11	6	Mie Endes	Pedas ...	Ini Gambar	8000	Porsi
4	3	11	6	Mie Bajak	Original	Ini Gambar	7000	Porsi
5	4	11	6	Ayam ...	Pedas ...	Ini Gambar	10000	Porsi
6	5	0	5	Jus Jambu	Dingin	Ini Gambar	5000	Gelas

## **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