**Основное приложение это файл App\_library2, база данных создавалась в PostgressSQL 14**

**Создание БД:**

CREATE DATABASE Library2;

**Создание структуры БД:**

CREATE TABLE Books

(

Id SERIAL PRIMARY KEY,

Name VARCHAR(20) NOT NULL,

Author VARCHAR(30) NOT NULL,

genre VARCHAR(30) NOT NULL,

In\_Library BOOL NOT NULL

);

CREATE TABLE Readers

(

Id SERIAL PRIMARY KEY,

Name VARCHAR(30) NOT NULL,

Addres VARCHAR(30) NOT NULL,

Telefon VARCHAR(30) NOT NULL

);

CREATE TABLE Lending\_Books

(

Id\_Book Serial REFERENCES Books (Id) ON DELETE CASCADE,

Id\_Reader Serial REFERENCES Readers (Id) ON DELETE CASCADE,

Data\_Issue DATE NOT NULL,

Data\_Plan DATE NOT NULL,

Data\_Return DATE

);

**Заполнение БД:**

INSERT INTO books

VALUES

(1,'Русские народные сказки', 'Александр Николаевич Афанасьев', true, 'сказки'),

(2,'Сказки 1001 ночи', 'Салье Михаил Александрович', true, 'сказки'),

(3,'Янтарь рассеивает тьму', 'Аквила Люцида', true, 'сказки'),

(4,'Царство Страха', 'Манискалко Керри', false, 'сказки'),

(5,'Давай надеяться на лучшее', 'Сеттерваль Каролина', true, 'приключение'),

(6,'Огонь и пепел', 'Мэйберри Джонатан', false, 'роман'),

(7,'Разрушительная любовь', 'Хуан Ана', true, 'драма');

INSERT INTO readers

VALUES

(1,'Иванов И.И.', 'Пермь-Ленина', '222-11-33'),

(2,'Петров П.П.', 'Пермь-Компрос', '111-22-44'),

(3,'Сидоров С.С.', 'Пермь-Карпинского', '333-44-55'),

(4,'Кукуева К.К.', 'Пермь-Мира', '555-22-11'),

(5,'Шишкина С.И.', 'Кунгур', '99999999'),

(6,'Якушев Я.Я.', 'Слоикамск', '2323232323'),

(7,'Свистунов В.А.', 'Добрянка', '11111111111'),

(8,'Рыбкина М.М.', 'Березники', '333333333'),

(9,'Шушунов М.М.', 'Пермь', '1234567'),

(10,'Кирякова К.К.', 'Оса', '123456');

INSERT INTO lending\_books

VALUES

(1, 1, '12.12.2009', '23.12.2009', '24.12.2009'),

(4, 1, '12.12.2009', '20.12.2009', '24.12.2009'),

(3, 2, '12.12.2009', '23.12.2009', '24.12.2009'),

(2, 5, '12.12.2009', '23.12.2009',NULL),

(5, 5, '11.09.2009', '23.10.2009', '24.12.2009');

**Библиотека для работы с БД change\_db:**

import psycopg2  
from PyQt5.QtWidgets import QTableWidget, QApplication, QMainWindow, QTableWidget  
from PyQt5.QtWidgets import QTableWidgetItem, QWidget, QPushButton, QLineEdit  
  
def add\_author():  
 with psycopg2.connect(user="postgres",  
 password="1",  
 host="localhost",  
 port="5432",  
 database="library2") as conn:  
 cur = conn.cursor()  
 cur.execute(f"insert into readers (name, addres, telefon) values ('Vasiliy', 'Perm', {28})")  
 conn.commit()  
  
*#добавить запись*def ins(table,name1,name2,name3,name4,param1,param2,param3,param4):  
 with psycopg2.connect(user="postgres",  
 password="1",  
 host="localhost",  
 port="5432",  
 database="library2") as conn:  
 cur = conn.cursor()  
 cur.execute(f"insert into {table} ({name1}, {name2}, {name3},{name4}) values ('{param1}', '{param2}', '{param3}', '{param4}')")  
 conn.commit()  
  
def ins\_reader(table,name1,name2,name3,param1,param2,param3):  
 with psycopg2.connect(user="postgres",  
 password="1",  
 host="localhost",  
 port="5432",  
 database="library2") as conn:  
 cur = conn.cursor()  
 cur.execute(f"insert into {table} ({name1}, {name2}, {name3}) values ('{param1}', '{param2}', '{param3}')")  
 conn.commit()  
  
def ins\_lending\_books(table,name1,name2,name3,name4,name5,param1,param2,param3,param4,param5):  
 with psycopg2.connect(user="postgres",  
 password="1",  
 host="localhost",  
 port="5432",  
 database="library2") as conn:  
 cur = conn.cursor()  
 cur.execute(f"insert into {table} ({name1}, {name2}, {name3}, {name4}, {name5}) values ('{param1}', '{param2}', '{param3}','{param4}','{param5}')")  
 conn.commit()  
  
def change\_lending\_books(table,element1,element2, name1, name2, name3, name4, name5, param1, param2, param3, param4, param5):  
 with psycopg2.connect(user="postgres",  
 password="1",  
 host="localhost",  
 port="5432",  
 database="library2") as conn:  
 cur = conn.cursor()  
 ids = int(element1) *# идентификатор строки* ids2=int(element2)  
 cur.execute(f"Update {table} set {name1}='{param1}',{name2}='{param2}',{name3}='{param3}',{name4}='{param4}',{name5}='{param5}' where id\_book = {ids} and id\_reader={ids2}")  
 conn.commit()  
*# удалить из таблицы строку*def dels(element,table):  
 with psycopg2.connect(user="postgres",  
 password="1",  
 host="localhost",  
 port="5432",  
 database="library2") as conn:  
 cur = conn.cursor()  
 ids = int(element) *# идентификатор строки* cur.execute(f"delete from {table} where id={ids}")  
 conn.commit()  
  
*# обновить запись*def change(table,element,name1,name2,name3,name4,param1,param2,param3,param4):  
  
 with psycopg2.connect(user="postgres",  
 password="1",  
 host="localhost",  
 port="5432",  
 database="library2") as conn:  
 cur = conn.cursor()  
 ids = int(element) *# идентификатор строки* cur.execute(f"Update {table} set {name1}='{param1}',{name2}='{param2}',{name3}='{param3}',{name4}='{param4}' where id = {ids}")  
 conn.commit()  
  
def change\_reader(table,element,name1,name2,name3,param1,param2,param3):  
  
 with psycopg2.connect(user="postgres",  
 password="1",  
 host="localhost",  
 port="5432",  
 database="library2") as conn:  
 cur = conn.cursor()  
 ids = int(element) *# идентификатор строки* cur.execute(f"Update {table} set {name1}='{param1}',{name2}='{param2}',{name3}='{param3}' where id = {ids}")  
 conn.commit()  
*#количество книг в библиотеке*def count\_book():  
 with psycopg2.connect(user="postgres",  
 password="1",  
 host="localhost",  
 port="5432",  
 database="library2") as conn:  
 cur = conn.cursor()  
 cur.execute("SELECT COUNT(\*) from books where in\_library=true")  
  
 return cur.fetchall()  
*# количество читателей*def count\_readers():  
 with psycopg2.connect(user="postgres",  
 password="1",  
 host="localhost",  
 port="5432",  
 database="library2") as conn:  
 cur = conn.cursor()  
 cur.execute("SELECT COUNT(\*) from readers")  
 return cur.fetchall()  
*#количество книг, который брал каждый читатель за все время*def count\_books():  
 with psycopg2.connect(user="postgres",  
 password="1",  
 host="localhost",  
 port="5432",  
 database="library2") as conn:  
 cur = conn.cursor()  
 cur.execute("SELECT readers.name,Count(\*) from lending\_books inner join readers on lending\_books.id\_reader=readers.id Group by readers.name")  
 return cur.fetchall()  
*#количество книг у читателя*def count\_reader\_book():  
 with psycopg2.connect(user="postgres",  
 password="1",  
 host="localhost",  
 port="5432",  
 database="library2") as conn:  
 cur = conn.cursor()  
 cur.execute("Select readers.name, COUNT(\*) from lending\_books inner join readers on lending\_books.id\_reader=readers.id where data\_return Is Null Group by readers.name")  
 return cur.fetchall()  
*#дата последнего посещения читетелем библиотеки*def reader\_last(element):  
 with psycopg2.connect(user="postgres",  
 password="1",  
 host="localhost",  
 port="5432",  
 database="library2") as conn:  
 cur = conn.cursor()  
 ids = int(element) *# идентификатор строки* cur.execute(f"Select MAX(data\_issue) from lending\_books where id\_reader={ids}")  
 return cur.fetchall()  
*#самый читаемый автор*def max\_author():  
 with psycopg2.connect(user="postgres",  
 password="1",  
 host="localhost",  
 port="5432",  
 database="library2") as conn:  
 cur = conn.cursor()  
 cur.execute(f"select author from books where id in (Select count(author) from books left join lending\_books on books.id = lending\_books.id\_book group by author)")  
 return cur.fetchall()  
*#самые предпочитаемые жанры*def genre():  
 with psycopg2.connect(user="postgres",  
 password="1",  
 host="localhost",  
 port="5432",  
 database="library2") as conn:  
 cur = conn.cursor()  
 cur.execute(f"select genre from books left join lending\_books on books.id = lending\_books.id\_book order by genre desc")  
 return cur.fetchall()  
*#любимый жанр каждого читателя*def love\_genre():  
 with psycopg2.connect(user="postgres",  
 password="1",  
 host="localhost",  
 port="5432",  
 database="library2") as conn:  
 cur = conn.cursor()  
 cur.execute(f"select readers.name,max(genre) from readers inner join lending\_books on readers.id=lending\_books.id\_reader inner join books on books.id=lending\_books.id\_book group by readers.name")  
 return cur.fetchall()

**Библиотека диалогового окна для вывода отчетов mbox:**

import sys  
  
from PyQt5 import QtGui  
from PyQt5.QtCore import QRect  
from PyQt5.QtWidgets import QTableWidgetItem, QWidget, QPushButton, QLineEdit, QMessageBox, QTableWidget, QSizePolicy, \  
 QApplication  
  
  
class MyMessageBox(QMessageBox):  
 def \_\_init\_\_(self,result1):  
 self.result1=result1  
 QMessageBox.\_\_init\_\_(self)  
  
 self.setSizeGripEnabled (True)  
  
 self.setWindowTitle ('Hello MessageBox')  
  
  
 *#Add TableWidget to QMessageBox* self.addTableWidget (self)  
 currentClick = self.exec\_()  
  
  
 *#Create TableWidget* def addTableWidget (self, parentItem) :  
 self.tableWidget = QTableWidget(parentItem)  
 self.tableWidget.setGeometry (QRect(0, 0, 540, 250))  
 self.tableWidget.setObjectName ('tableWidget')  
  
 self.tableWidget.setColumnCount(2)  
  
  
 self.tableWidget.clear()  
 self.tableWidget.setRowCount(0)  
 self.tableWidget.setHorizontalHeaderLabels(['Результат']) *# заголовки столцов* rows = self.result1  
 i = 0  
 for elem in rows:  
 self.tableWidget.setRowCount(self.tableWidget.rowCount() + 1)  
 j = 0  
 for t in elem: *# заполняем внутри строки* self.tableWidget.setItem(i, j, QTableWidgetItem(str(t).strip()))  
 j += 1  
 i += 1  
  
 self.tableWidget.resizeColumnsToContents()  
 *#Allow resizing of QMessageBox* def event(self, e):  
 result = QMessageBox.event(self, e)  
 self.setMinimumWidth(0)  
 self.setMaximumWidth(16777215)  
 self.setMinimumHeight(0)  
 self.setMaximumHeight(16777215)  
 self.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding)  
 self.resize(550, 300)  
  
 return result  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QApplication(sys.argv)  
 ex = MyMessageBox ()  
 *#ex.show()* sys.exit(app.exec\_())

**Основной код приложения App\_Library2:**

from datetime import datetime  
  
from PyQt5 import QtWidgets  
import psycopg2  
from PyQt5.QtWidgets import QTableWidget, QApplication, QMainWindow, QTableWidget  
from PyQt5.QtWidgets import QTableWidgetItem, QWidget, QPushButton, QLineEdit, QMessageBox  
import sys  
from MainForm import Ui\_MainWindow  
import change\_db as ll  
import mbox as mb  
class mywindow(QtWidgets.QMainWindow):  
 def \_\_init\_\_(self):  
 super(mywindow, self).\_\_init\_\_()  
 self.ui = Ui\_MainWindow()  
 self.ui.setupUi(self)  
  
 self.setWindowTitle('Библиотека')  
 self.ui.tableWidget.setColumnCount(5)  
 self.ui.tableWidget\_2.setColumnCount(4)  
 self.ui.tableWidget\_3.setColumnCount(5)  
 self.ui.tableWidget.verticalHeader().hide()  
 self.ui.tableWidget\_2.verticalHeader().hide()  
 self.ui.tableWidget\_3.verticalHeader().hide()  
 self.updt\_book()  
 self.updt\_reader()  
 self.updt\_lending\_book()  
 self.ui.tableWidget.setEditTriggers(QTableWidget.NoEditTriggers) *# запретить изменять поля* self.ui.tableWidget.cellClicked.connect(self.cellClick\_book)  
 self.ui.tableWidget\_2.cellClicked.connect(self.cellClick\_reader)  
 self.ui.tableWidget\_3.cellClicked.connect(self.cellClick\_lending\_book)  
 self.ui.pushButton.clicked.connect(self.add\_books)  
 self.ui.pushButton\_2.clicked.connect(self.del\_book)  
 self.ui.pushButton\_3.clicked.connect(self.change\_book)  
 self.ui.pushButton\_8.clicked.connect(self.add\_reader)  
 self.ui.pushButton\_7.clicked.connect(self.del\_reader)  
 self.ui.pushButton\_9.clicked.connect(self.change\_reader)  
 self.ui.pushButton\_10.clicked.connect(self.add\_lending\_book)  
 self.ui.pushButton\_11.clicked.connect(self.change\_lending\_book)  
  
 self.ui.action.triggered.connect(self.clickMethod1)  
 self.ui.action\_2.triggered.connect(self.clickMethod2)  
 self.ui.action\_3.triggered.connect(self.clickMethod3)  
 self.ui.action\_4.triggered.connect(self.clickMethod4)  
 self.ui.action\_5.triggered.connect(self.clickMethod5)  
 self.ui.action\_6.triggered.connect(self.clickMethod6)  
 self.ui.action\_7.triggered.connect(self.clickMethod7)  
 self.ui.action\_8.triggered.connect(self.clickMethod8)  
  
 *#заполнитель для книги* def updt\_book(self):  
 self.ui.tableWidget.clear()  
 self.ui.tableWidget.setRowCount(0)  
 self.ui.tableWidget.setHorizontalHeaderLabels(['id', 'Книга', 'Автор', 'В библиотеке','Жанр']) *# заголовки столцов* with psycopg2.connect(user="postgres",  
 password="1",  
 host="localhost",  
 port="5432",  
 database="library2") as conn:  
 self.cur = conn.cursor()  
 self.cur.execute("select \* from books")  
 rows = self.cur.fetchall()  
 i = 0  
 for elem in rows:  
 self.ui.tableWidget.setRowCount(self.ui.tableWidget.rowCount() + 1)  
 j = 0  
 for t in elem: *# заполняем внутри строки* self.ui.tableWidget.setItem(i, j, QTableWidgetItem(str(t).strip()))  
 j += 1  
 i += 1  
  
 self.ui.tableWidget.resizeColumnsToContents()  
 *#заполнитель для читателя* def updt\_reader(self):  
 self.ui.tableWidget\_2.clear()  
 self.ui.tableWidget\_2.setRowCount(0)  
 self.ui.tableWidget\_2.setHorizontalHeaderLabels(['id', 'Читатель', 'Адрес', 'Телефон']) *# заголовки столцов* with psycopg2.connect(user="postgres",  
 password="1",  
 host="localhost",  
 port="5432",  
 database="library2") as conn:  
 self.cur = conn.cursor()  
 self.cur.execute("select \* from readers")  
 rows = self.cur.fetchall()  
 i = 0  
 for elem in rows:  
 self.ui.tableWidget\_2.setRowCount(self.ui.tableWidget\_2.rowCount() + 1)  
 j = 0  
 for t in elem: *# заполняем внутри строки* self.ui.tableWidget\_2.setItem(i, j, QTableWidgetItem(str(t).strip()))  
 j += 1  
 i += 1  
  
 self.ui.tableWidget\_2.resizeColumnsToContents()  
 *#заполнитель для выданных книг* def updt\_lending\_book(self):  
 self.ui.tableWidget\_3.clear()  
 self.ui.tableWidget\_3.setRowCount(0)  
 self.ui.tableWidget\_3.setHorizontalHeaderLabels(['id книги','id читателя','Дата приема', 'Дата планируемого возврата','Дата возврата']) *# заголовки столцов* with psycopg2.connect(user="postgres",  
 password="1",  
 host="localhost",  
 port="5432",  
 database="library2") as conn:  
 self.cur = conn.cursor()  
 self.cur.execute("select \* from lending\_books")  
 rows = self.cur.fetchall()  
 i = 0  
 for elem in rows:  
 self.ui.tableWidget\_3.setRowCount(self.ui.tableWidget\_3.rowCount() + 1)  
 j = 0  
 for t in elem: *# заполняем внутри строки* self.ui.tableWidget\_3.setItem(i, j, QTableWidgetItem(str(t).strip()))  
 j += 1  
 i += 1  
  
 self.ui.tableWidget\_3.resizeColumnsToContents()  
  
 *#обработчик для книги* def cellClick\_book(self, row, col): *# row - номер строки, col - номер столбца* self.id=self.ui.tableWidget.item(row, 0).text().strip()  
 self.ui.lineEdit.setText(self.ui.tableWidget.item(row, 1).text().strip())  
 self.ui.lineEdit\_2.setText(self.ui.tableWidget.item(row, 2).text().strip())  
 self.ui.checkBox.setChecked(eval(self.ui.tableWidget.item(row, 3).text()))  
 self.ui.lineEdit\_4.setText(self.ui.tableWidget.item(row, 4).text().strip())  
  
 *#обработчик для читателя* def cellClick\_reader(self, row, col): *# row - номер строки, col - номер столбца* self.id2=self.ui.tableWidget\_2.item(row, 0).text().strip()  
 self.ui.lineEdit\_3.setText(self.ui.tableWidget\_2.item(row, 1).text().strip())  
 self.ui.lineEdit\_7.setText(self.ui.tableWidget\_2.item(row, 2).text().strip())  
 self.ui.lineEdit\_8.setText(self.ui.tableWidget\_2.item(row, 3).text().strip())  
  
 *# обработчик для читателя* def cellClick\_lending\_book(self, row, col): *# row - номер строки, col - номер столбца* self.id3 = self.ui.tableWidget\_3.item(row, 0).text().strip()  
 self.id4 = self.ui.tableWidget\_3.item(row, 1).text().strip()  
 self.ui.lineEdit\_5.setText(self.ui.tableWidget\_3.item(row, 0).text().strip())  
 self.ui.lineEdit\_10.setText(self.ui.tableWidget\_3.item(row, 1).text().strip())  
 *#print(datetime.strptime(self.ui.tableWidget\_3.item(row,2).text(),"%Y-%m-%d").date())* self.ui.dateEdit.setDate(datetime.strptime(self.ui.tableWidget\_3.item(row,2).text(),"%Y-%m-%d").date())  
 self.ui.dateEdit\_2.setDate(datetime.strptime(self.ui.tableWidget\_3.item(row,3).text(),"%Y-%m-%d").date())  
 self.ui.dateEdit\_3.setDate(datetime.strptime(self.ui.tableWidget\_3.item(row,4).text(),"%Y-%m-%d").date())  
  
 *#обновление полей книги* def upd\_book(self):  
 self.updt\_book()  
 self.ui.lineEdit.setText('')  
 self.ui.lineEdit\_2.setText('')  
 self.ui.lineEdit\_4.setText('')  
 self.ui.checkBox.setChecked(False)  
 *#обновление полей читателя* def upd\_reader(self):  
 self.updt\_reader()  
 self.ui.lineEdit\_3.setText('')  
 self.ui.lineEdit\_7.setText('')  
 self.ui.lineEdit\_8.setText('')  
  
 *# обновление полей выданных книг* def upd\_lending\_book(self):  
 self.updt\_lending\_book()  
 self.ui.lineEdit\_5.setText('')  
 self.ui.lineEdit\_10.setText('')  
 self.ui.dateEdit.clear()  
 self.ui.dateEdit\_2.clear()  
 self.ui.dateEdit\_3.clear()  
  
 *#добавление книги* def add\_books(self):  
 *#print(self.ui.checkBox.isChecked())* name1, author1, in\_library1,genre1 = self.ui.lineEdit.text(), self.ui.lineEdit\_2.text(), self.ui.checkBox.isChecked(),self.ui.lineEdit\_4.text()  
  
 ll.ins('books','name','author','in\_library','genre',name1,author1,in\_library1,genre1)  
 self.upd\_book()  
 *#удаление книги* def del\_book(self):  
 ll.dels(self.id,'books')  
 self.upd\_book()  
 *#изменение книги* def change\_book(self):  
 name1, author1, in\_library1,genre1 = self.ui.lineEdit.text(), self.ui.lineEdit\_2.text(), self.ui.checkBox.isChecked(),self.ui.lineEdit\_4.text()  
 ll.change('books',self.id,'name','author','in\_library','genre',name1,author1,in\_library1,genre1)  
 self.upd\_book()  
 *#добавление читателя* def add\_reader(self):  
 name1, addres1, telefon1 = self.ui.lineEdit\_3.text(), self.ui.lineEdit\_7.text(), self.ui.lineEdit\_8.text()  
 ll.ins\_reader('readers', 'name', 'addres', 'telefon', name1, addres1, telefon1)  
 self.upd\_reader()  
 *#удаление читателя* def del\_reader(self):  
 ll.dels(self.id2,'readers')  
 self.upd\_reader()  
 *#обновление читателя* def change\_reader(self):  
 name1, addres1, telefon1 = self.ui.lineEdit\_3.text(), self.ui.lineEdit\_7.text(), self.ui.lineEdit\_8.text()  
 ll.change\_reader('readers', self.id2, 'name', 'addres', 'telefon', name1, addres1, telefon1)  
 self.upd\_reader()  
  
 *# изменение выданной книги* def change\_lending\_book(self):  
 id1, id2, data1, data2, data3 = self.ui.lineEdit\_5.text(), self.ui.lineEdit\_10.text(), self.ui.dateEdit, self.ui.dateEdit\_2, self.ui.dateEdit\_3  
 data1 = datetime.strptime(data1.text().strip(), "%Y-%m-%d")  
 data2 = datetime.strptime(data2.text().strip(), "%Y-%m-%d")  
 data3 = datetime.strptime(data3.text().strip(), "%Y-%m-%d")  
 ll.change\_lending\_books('lending\_books',self.id3,self.id4, 'id\_book', 'id\_reader', 'data\_issue','data\_plan','data\_return', id1, id2, data1,data2,data3)  
 self.upd\_lending\_book()  
  
 *# добавление выданной книги* def add\_lending\_book(self):  
 id1, id2, data1,data2,data3 = self.ui.lineEdit\_5.text(), self.ui.lineEdit\_10.text(), self.ui.dateEdit,self.ui.dateEdit\_2,self.ui.dateEdit\_3  
 *#print(data1.text())* data1=datetime.strptime(data1.text().strip(),"%Y-%m-%d")  
 data2 = datetime.strptime(data2.text().strip(), "%Y-%m-%d")  
 data3 = datetime.strptime(data3.text().strip(), "%Y-%m-%d")  
 ll.ins\_lending\_books('lending\_books', 'id\_book', 'id\_reader', 'data\_issue','data\_plan','data\_return', id1, id2, data1,data2,data3)  
 self.upd\_lending\_book()  
  
 def clickMethod1(self):  
 info=ll.count\_book()  
 mb.MyMessageBox(result1=info)  
  
 def clickMethod2(self):  
 info=ll.count\_readers()  
 mb.MyMessageBox(result1=info)  
  
 def clickMethod3(self):  
 info=ll.count\_books()  
 mb.MyMessageBox(result1=info)  
  
 def clickMethod4(self):  
 info=ll.count\_reader\_book()  
 mb.MyMessageBox(result1=info)  
  
 def clickMethod5(self):  
 info=ll.reader\_last(self.id2)  
 mb.MyMessageBox(result1=info)  
  
 def clickMethod6(self):  
 info=ll.max\_author()  
 mb.MyMessageBox(result1=info)  
  
 def clickMethod7(self):  
 info=ll.genre()  
 mb.MyMessageBox(result1=info)  
  
 def clickMethod8(self):  
 info=ll.love\_genre()  
 mb.MyMessageBox(result1=info)  
  
app = QtWidgets.QApplication([])  
application = mywindow()  
application.show()  
  
sys.exit(app.exec())