**Министерство цифрового развития, связи и массовых коммуникаций**

**Российской Федерации**

**Ордена Трудового Красного Знамени федеральное государственное бюджетное**

**Образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

**Лабораторная работа №8**

**«Создание визуального интерфейса для базы данных»**

**По дисциплине**

**«ИТ и Программирование»**

Выполнил: студент группы

БИН2002 Гулякин А.И.

Проверил: Чайка А.Д.

Москва 2022 г.

Оглавление

[**1.** **Цель работы** 3](#_Toc89954253)

[**2.** **Выполнение** 3](#_Toc89954254)

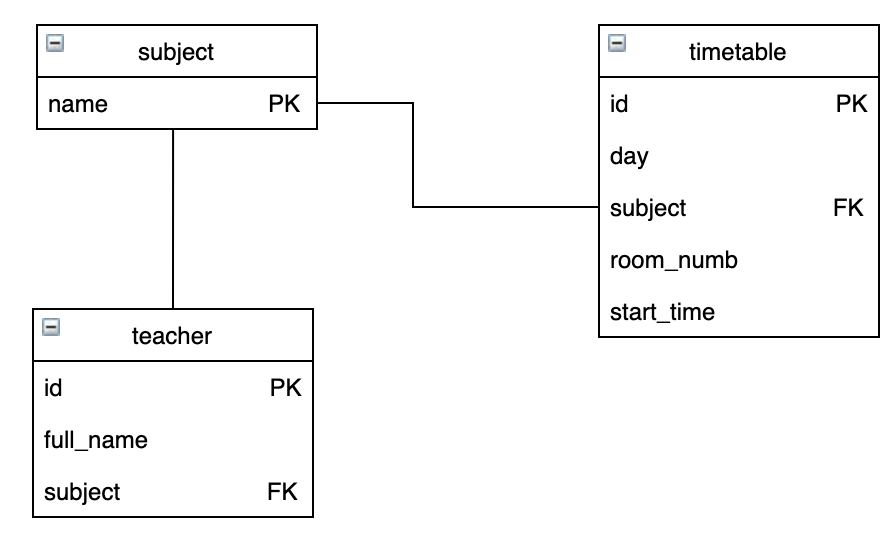
[**3.** **Вывод** 8](#_Toc89954255)

1. **Цель работы**

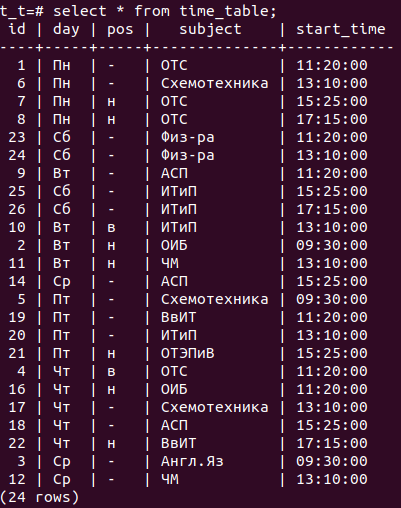
Создать оконное приложение, позволяющее редактировать базу данных с расписанием группы.

1. **Выполнение**

Для выполнения данной лабораторной работы, требуется создание sql базы данных с помощью программы PostgreSQL, включающей в себя три таблиц по схеме:



После создания база дынных выглядит так (рисунок 1)



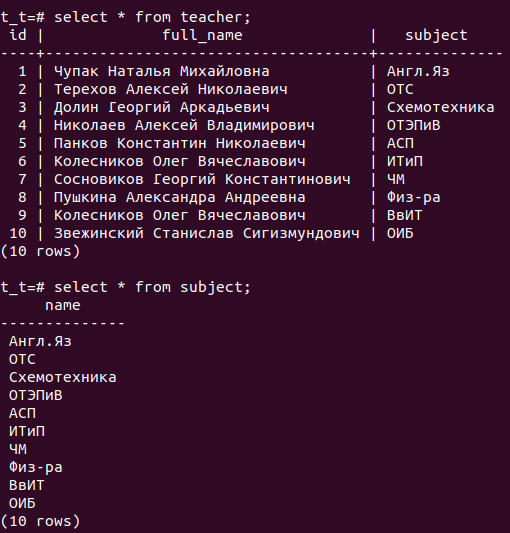
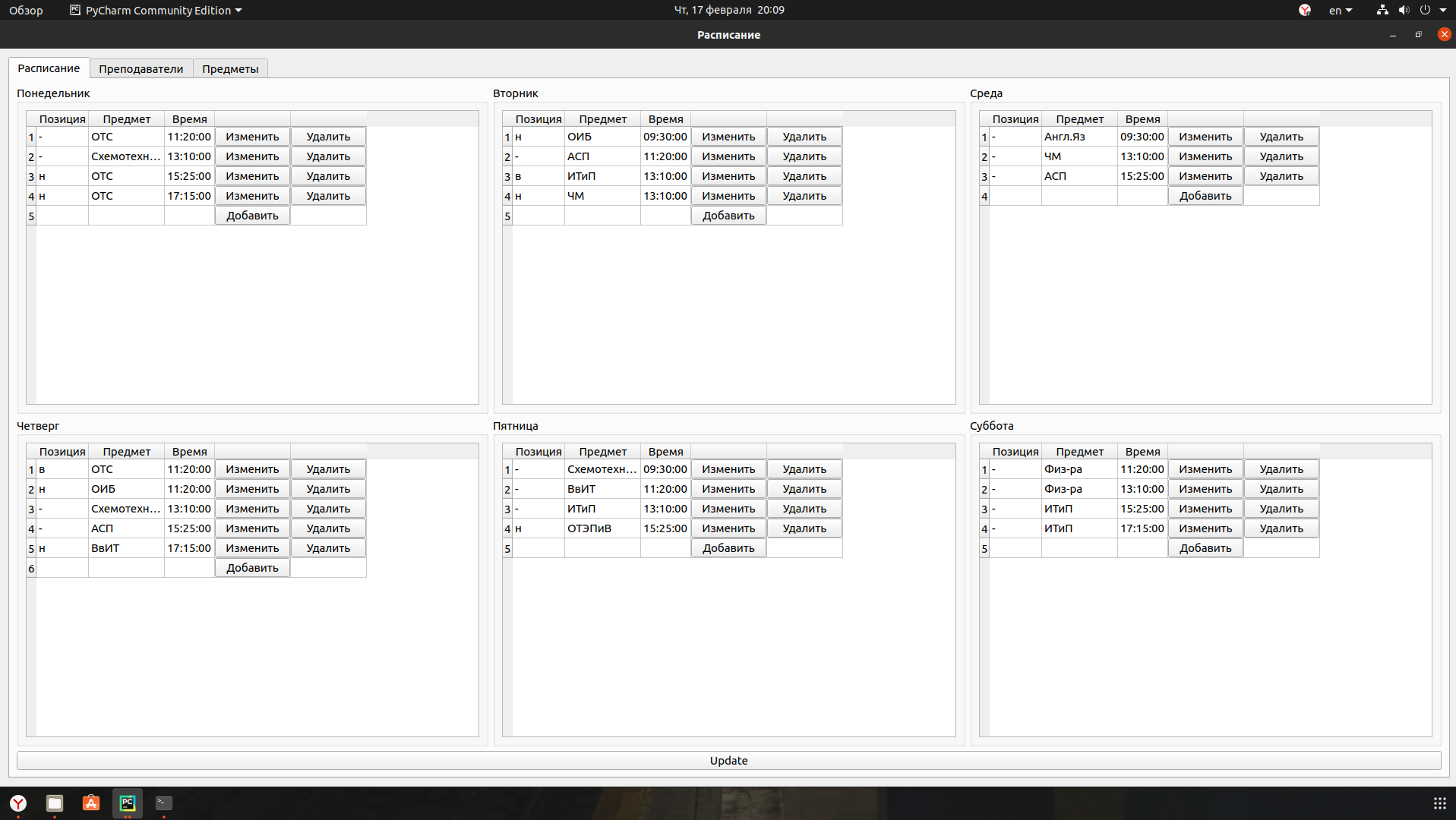
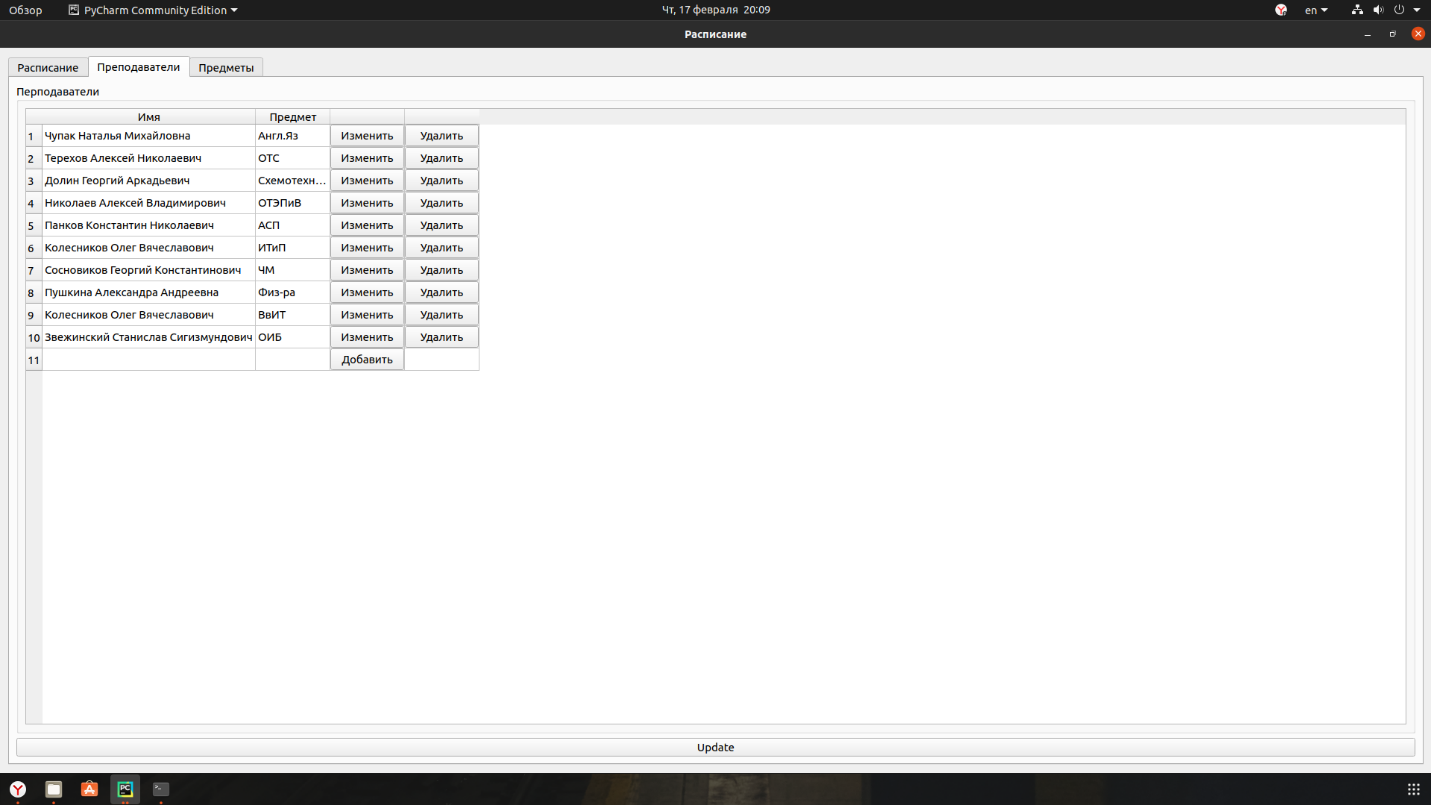


Рисунок – данные из sql таблиц

При запуске программы пользователю выводится приложение с тремя вкладками ‘Расписание’ ‘Преподаватели’ ‘Предметы’ (рисунок 2)





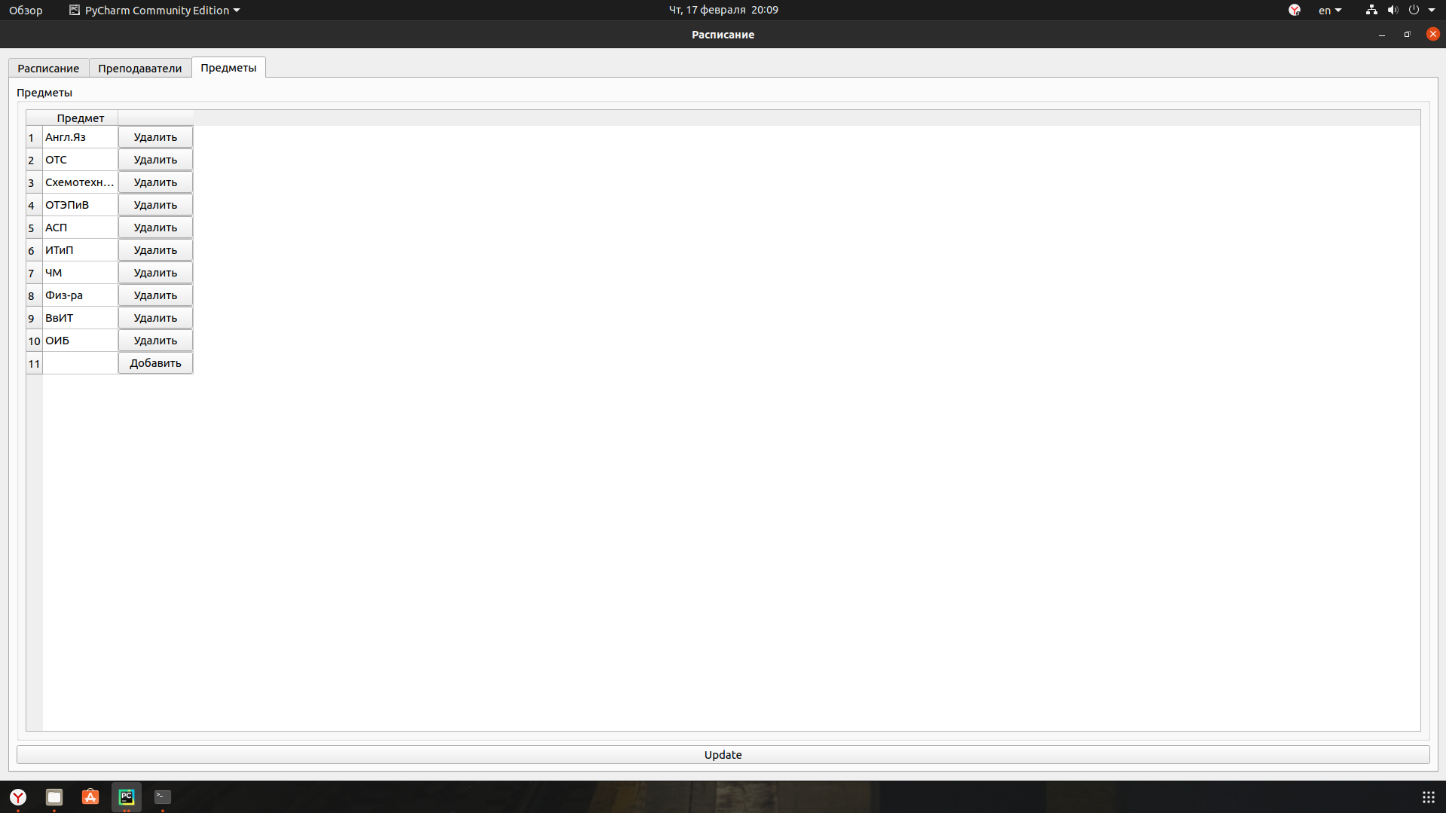
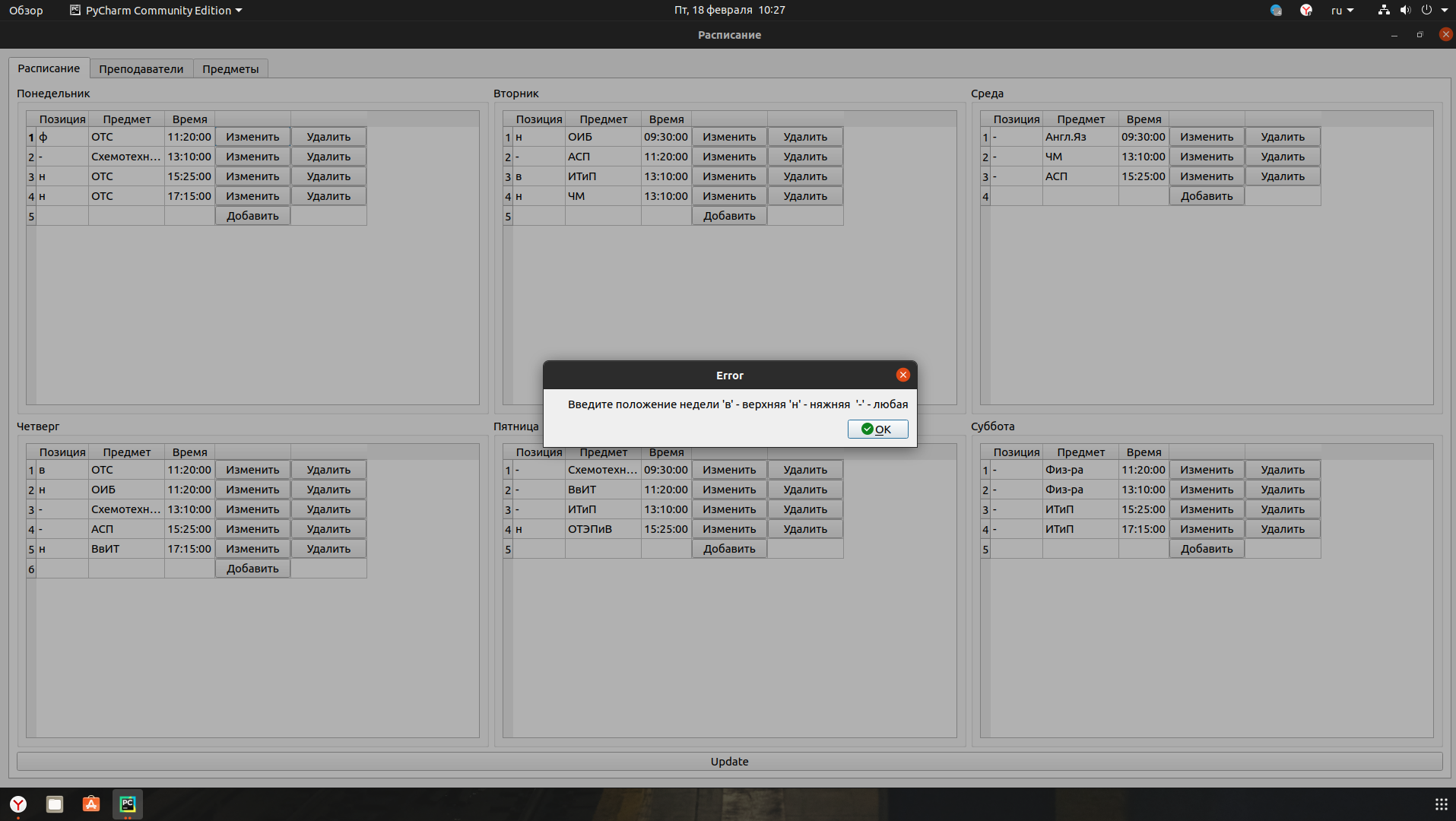


Рисунок - Вкладки таблицы

Все данные в таблице должны иметь строгий формат, поэтому, когда пользователь пытается что-то изменить или добавить, заполнив поле неправильно, ему выводится ошибка (рисунок 3)



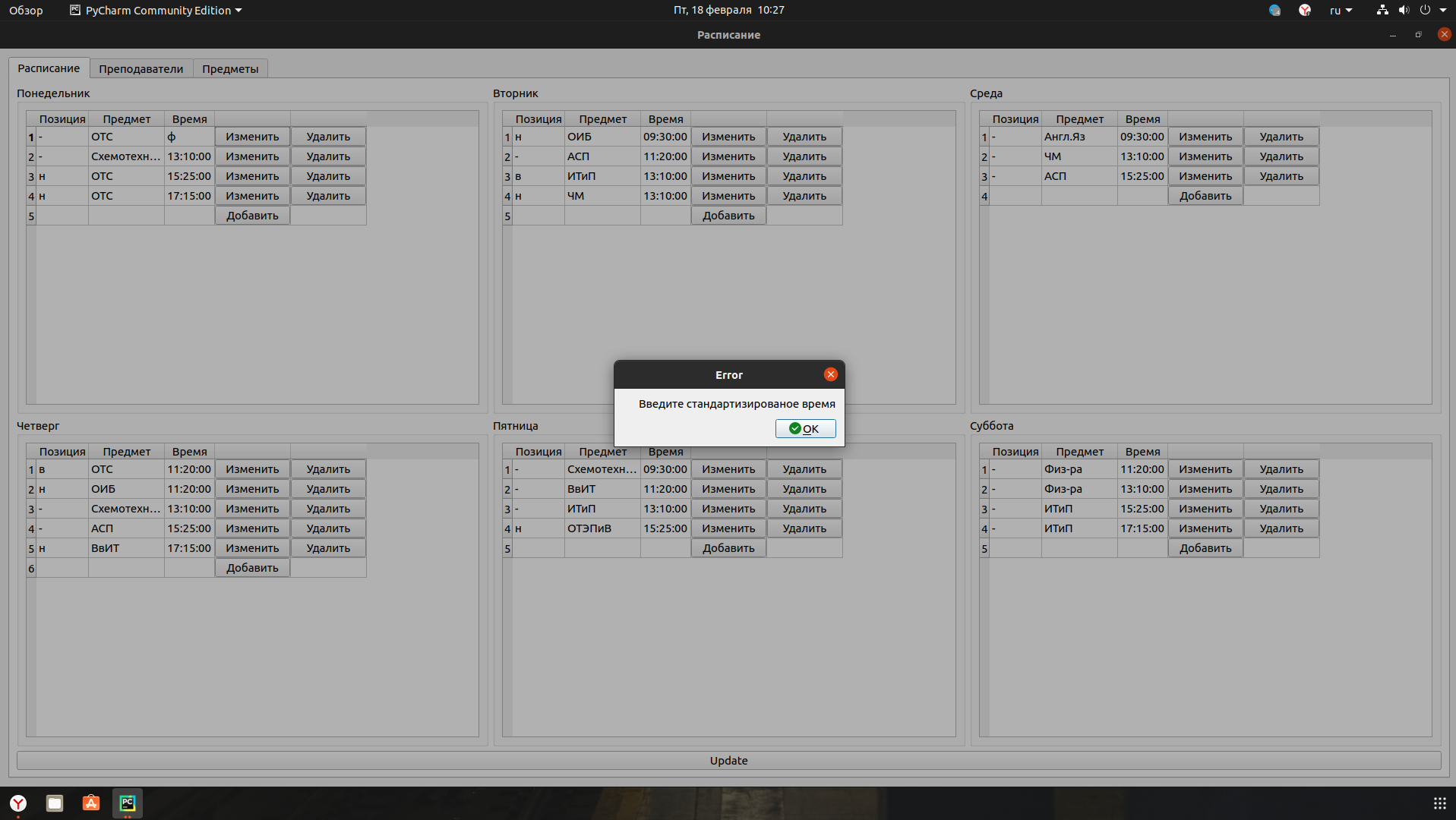
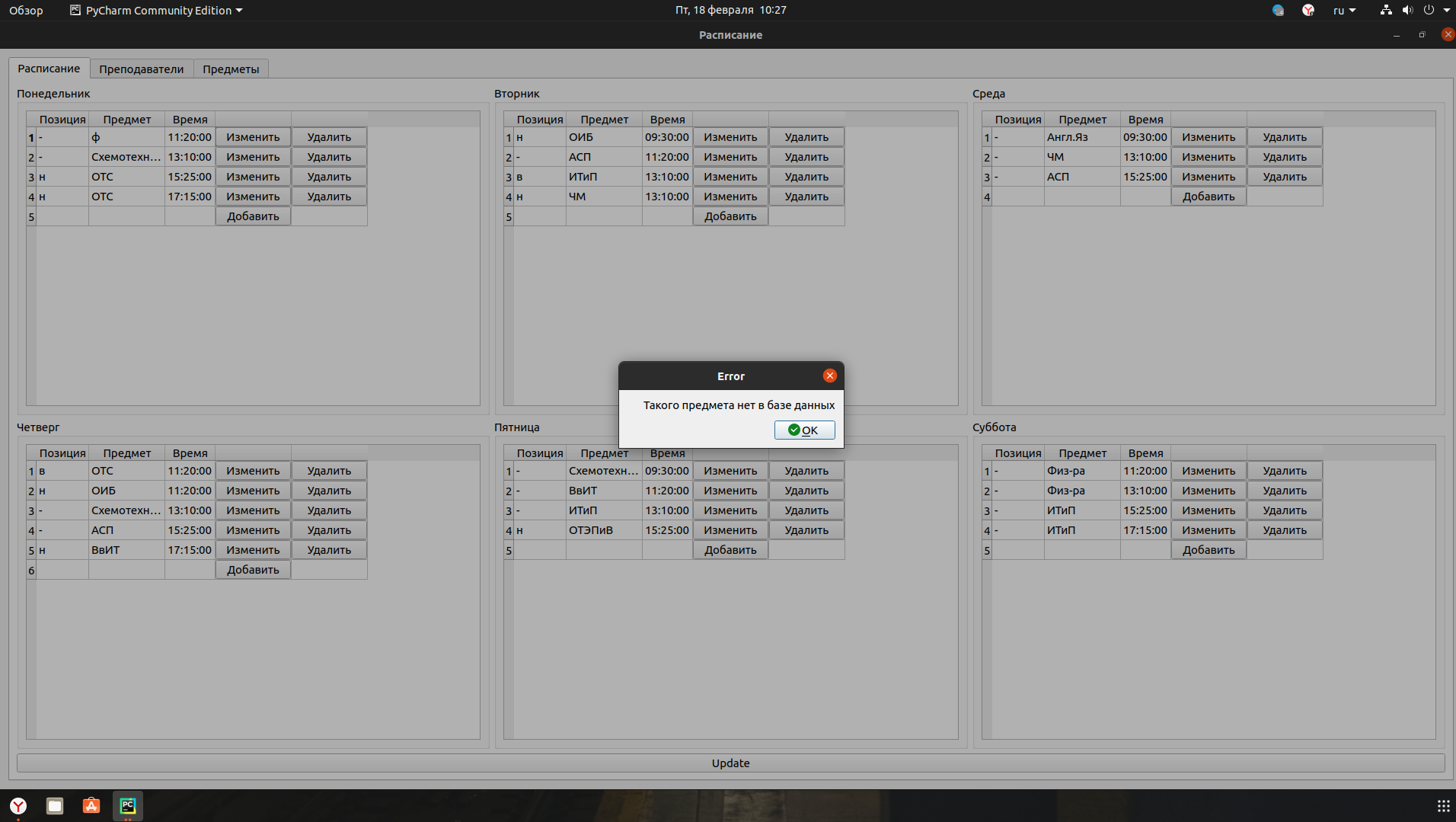


Рисунок - варианты ошибок

Функции изменения и добавления работают только когда пользователь ввел все данные верно

При попытке удаления предмета, который связан с другими таблицами, пользователю выводит ошибку (рисунок 4)

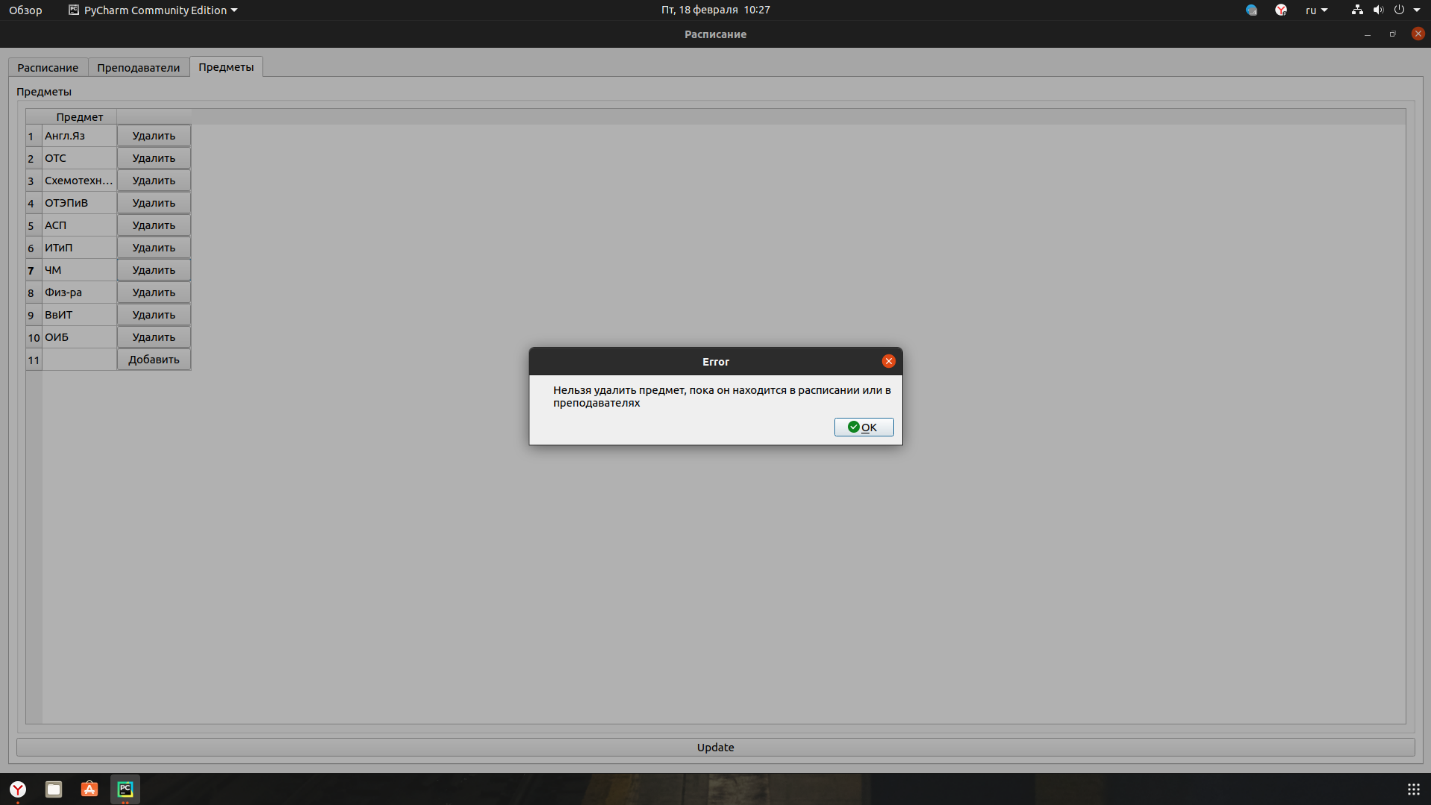


Рисунок – ошибка

1. **Вывод**

В ходе выполнения работы, я научился созданию оконного приложения, для редактирования sql базы данных

**Приложение:**

**Программный код файла maim.py**

import sys  
import psycopg2  
from PyQt5.QtWidgets import (QApplication, QWidget, QTabWidget, QAbstractScrollArea, QVBoxLayout, QHBoxLayout,  
 QTableWidget, QGroupBox, QTableWidgetItem, QPushButton, QMessageBox)  
  
work\_time = ['9:30', '11:20', '13:10', '15:25', '17:15', '09:30', '9:30:00', '11:20:00', '13:10:00', '15:25:00',  
 '17:15:00', '09:30:00']  
  
  
class Window(QWidget):  
 def \_\_init\_\_(self):  
 super(Window, self).\_\_init\_\_()  
 self.setWindowTitle("Расписание")  
  
 self.\_connect\_to\_db()  
  
 self.tabs = QTabWidget(self)  
  
 self.vbox = QVBoxLayout(self)  
 self.vbox.addWidget(self.tabs)  
 self.\_create\_shedule\_tab()  
  
 def \_connect\_to\_db(self):  
 self.conn = psycopg2.connect(database="t\_t",  
 user="postgres",  
 password="ag1212200223",  
 host="localhost",  
 port="5432")  
 self.cursor = self.conn.cursor()  
  
 def \_create\_shedule\_tab(self):  
 self.shedule\_tab = QWidget()  
 self.tabs.addTab(self.shedule\_tab, "Расписание")  
 self.teacher\_tab = QWidget()  
 self.tabs.addTab(self.teacher\_tab, "Преподаватели")  
 self.subject\_tab = QWidget()  
 self.tabs.addTab(self.subject\_tab, "Предметы")  
  
 self.monday\_gbox = QGroupBox("Понедельник")  
 self.tuesday\_gbox = QGroupBox("Вторник")  
 self.wednesday\_gbox = QGroupBox("Среда")  
 self.thursday\_gbox = QGroupBox("Четверг")  
 self.friday\_gbox = QGroupBox("Пятница")  
 self.saturday\_gbox = QGroupBox("Суббота")  
 self.teacher\_gbox = QGroupBox("Перподаватели")  
 self.subject\_gbox = QGroupBox("Предметы")  
  
 self.svbox = QVBoxLayout()  
 self.shbox = QHBoxLayout()  
 self.svboxa = QVBoxLayout()  
 self.svboxb = QVBoxLayout()  
 self.svboxc = QVBoxLayout()  
 self.svbox2 = QVBoxLayout()  
 self.shbox2 = QHBoxLayout()  
 self.svbox3 = QVBoxLayout()  
 self.shbox3 = QHBoxLayout()  
  
 self.svbox.addLayout(self.shbox)  
 self.shbox.addLayout(self.svboxa)  
 self.shbox.addLayout(self.svboxb)  
 self.shbox.addLayout(self.svboxc)  
 self.svbox2.addLayout(self.shbox2)  
 self.svbox3.addLayout(self.shbox3)  
  
 self.svboxa.addWidget(self.monday\_gbox)  
 self.svboxb.addWidget(self.tuesday\_gbox)  
 self.svboxc.addWidget(self.wednesday\_gbox)  
 self.svboxa.addWidget(self.thursday\_gbox)  
 self.svboxb.addWidget(self.friday\_gbox)  
 self.svboxc.addWidget(self.saturday\_gbox)  
 self.shbox2.addWidget(self.teacher\_gbox)  
 self.shbox3.addWidget(self.subject\_gbox)  
  
 self.\_create\_monday\_table()  
 self.\_create\_tuesday\_table()  
 self.\_create\_wednesday\_table()  
 self.\_create\_thursday\_table()  
 self.\_create\_friday\_table()  
 self.\_create\_saturday\_table()  
 self.\_create\_teacher\_table()  
 self.\_create\_subject\_table()  
  
 self.update\_shedule\_btn = QPushButton("Update")  
 self.update\_teacher\_btn = QPushButton("Update")  
 self.update\_subject\_btn = QPushButton("Update")  
 self.svbox.addWidget(self.update\_shedule\_btn)  
 self.svbox2.addWidget(self.update\_teacher\_btn)  
 self.svbox3.addWidget(self.update\_subject\_btn)  
 self.update\_shedule\_btn.clicked.connect(self.\_update)  
 self.update\_teacher\_btn.clicked.connect(self.\_update)  
 self.update\_subject\_btn.clicked.connect(self.\_update)  
  
 self.shedule\_tab.setLayout(self.svbox)  
 self.teacher\_tab.setLayout(self.svbox2)  
 self.subject\_tab.setLayout(self.svbox3)  
  
 def \_create\_day\_table(self, day\_table, day\_gbox, day):  
  
 day\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)  
  
 day\_table.setColumnCount(6)  
 day\_table.setHorizontalHeaderLabels(["Позиция", "Предмет", "Время", "ID", "", ""])  
 day\_table.hideColumn(3)  
  
 self.\_update\_day\_table(day\_table, day)  
  
 self.mvbox = QVBoxLayout()  
 self.mvbox.addWidget(day\_table)  
 day\_gbox.setLayout(self.mvbox)  
  
 def \_create\_monday\_table(self):  
 self.monday\_table = QTableWidget()  
 self.\_create\_day\_table(self.monday\_table, self.monday\_gbox, 'Пн')  
  
 def \_create\_tuesday\_table(self):  
 self.tuesday\_table = QTableWidget()  
 self.\_create\_day\_table(self.tuesday\_table, self.tuesday\_gbox, 'Вт')  
  
 def \_create\_wednesday\_table(self):  
 self.wednesday\_table = QTableWidget()  
 self.\_create\_day\_table(self.wednesday\_table, self.wednesday\_gbox, 'Ср')  
  
 def \_create\_thursday\_table(self):  
 self.thursday\_table = QTableWidget()  
 self.\_create\_day\_table(self.thursday\_table, self.thursday\_gbox, 'Чт')  
  
 def \_create\_friday\_table(self):  
 self.friday\_table = QTableWidget()  
 self.\_create\_day\_table(self.friday\_table, self.friday\_gbox, 'Пт')  
  
 def \_create\_saturday\_table(self):  
 self.saturday\_table = QTableWidget()  
 self.\_create\_day\_table(self.saturday\_table, self.saturday\_gbox, 'Сб')  
  
 def \_create\_teacher\_table(self):  
 self.teacher\_table = QTableWidget()  
  
 self.teacher\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)  
  
 self.teacher\_table.setColumnCount(5)  
 self.teacher\_table.setHorizontalHeaderLabels(["Имя", "Предмет", "ID", "", ""])  
 self.teacher\_table.hideColumn(2)  
  
 self.\_update\_teacher\_table()  
  
 self.mvbox = QVBoxLayout()  
 self.mvbox.addWidget(self.teacher\_table)  
 self.teacher\_gbox.setLayout(self.mvbox)  
  
 def \_create\_subject\_table(self):  
 self.subject\_table = QTableWidget()  
  
 self.subject\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)  
  
 self.subject\_table.setColumnCount(2)  
 self.subject\_table.setHorizontalHeaderLabels(["Предмет", ""])  
  
 self.\_update\_subject\_table()  
  
 self.mvbox = QVBoxLayout()  
 self.mvbox.addWidget(self.subject\_table)  
 self.subject\_gbox.setLayout(self.mvbox)  
  
 def \_update\_day\_table(self, day\_table, day):  
 self.cursor.execute(  
 "select pos, subject, start\_time, id from time\_table where day = '" + day + "' order by start\_time;")  
 records = list(self.cursor.fetchall())  
 day\_table.setRowCount(len(records) + 1)  
  
 for i, r in enumerate(records):  
 r = list(r)  
 day\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))  
 day\_table.setItem(i, 1, QTableWidgetItem(str(r[1])))  
 day\_table.setItem(i, 2, QTableWidgetItem(str(r[2])))  
 day\_table.setItem(i, 3, QTableWidgetItem(str(r[3])))  
 joinbtn = QPushButton("Изменить")  
 day\_table.setCellWidget(i, 4, joinbtn)  
 joinbtn.clicked.connect(lambda checked=None, j=i: self.\_change\_day\_from\_table(j, day\_table, day))  
 delbtn = QPushButton("Удалить")  
 day\_table.setCellWidget(i, 5, delbtn)  
 delbtn.clicked.connect(lambda checked=None, j=i: self.\_del\_from\_time\_table(j, day\_table, day))  
 addbtn = QPushButton("Добавить")  
 day\_table.setItem(i + 1, 0, QTableWidgetItem(''))  
 day\_table.setItem(i + 1, 1, QTableWidgetItem(''))  
 day\_table.setItem(i + 1, 2, QTableWidgetItem(''))  
 day\_table.setItem(i + 1, 3, QTableWidgetItem(''))  
 day\_table.removeCellWidget(i + 1, 4)  
 day\_table.removeCellWidget(i + 1, 5)  
 day\_table.setCellWidget(i + 1, 4, addbtn)  
 addbtn.clicked.connect(lambda checked=None, j=i + 1: self.\_add\_time\_table(j, day\_table, day))  
 day\_table.resizeRowsToContents()  
 day\_table.resizeColumnToContents(0)  
 day\_table.resizeColumnToContents(2)  
  
 def \_update\_teacher\_table(self):  
 self.cursor.execute(  
 "select full\_name, subject, id from teacher;")  
 records = list(self.cursor.fetchall())  
  
 self.teacher\_table.setRowCount(len(records) + 1)  
  
 for i, r in enumerate(records):  
 r = list(r)  
 self.teacher\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))  
 self.teacher\_table.setItem(i, 1, QTableWidgetItem(str(r[1])))  
 self.teacher\_table.setItem(i, 2, QTableWidgetItem(str(r[2])))  
 joinbtn = QPushButton("Изменить")  
 self.teacher\_table.setCellWidget(i, 3, joinbtn)  
 joinbtn.clicked.connect(lambda checked=None, j=i: self.\_change\_teacher\_table(j, self.teacher\_table))  
 delbtn = QPushButton("Удалить")  
 self.teacher\_table.setCellWidget(i, 4, delbtn)  
 delbtn.clicked.connect(lambda checked=None, j=i: self.\_del\_from\_teacher(j, self.teacher\_table))  
 self.teacher\_table.setItem(i + 1, 0, QTableWidgetItem(''))  
 self.teacher\_table.setItem(i + 1, 1, QTableWidgetItem(''))  
 self.teacher\_table.setItem(i + 1, 2, QTableWidgetItem(''))  
 self.teacher\_table.setItem(i + 1, 3, QTableWidgetItem(''))  
 self.teacher\_table.removeCellWidget(i + 1, 4)  
 self.teacher\_table.removeCellWidget(i + 1, 5)  
 addbtn = QPushButton("Добавить")  
 self.teacher\_table.setItem(i + 1, 0, QTableWidgetItem(''))  
 self.teacher\_table.setCellWidget(i + 1, 3, addbtn)  
 addbtn.clicked.connect(lambda checked=None, j=i + 1: self.\_add\_teacher(j, self.teacher\_table))  
 self.teacher\_table.resizeColumnToContents(0)  
  
 def \_update\_subject\_table(self):  
 self.cursor.execute(  
 "select name from subject;")  
 records = list(self.cursor.fetchall())  
  
 self.subject\_table.setRowCount(len(records) + 1)  
  
 for i, r in enumerate(records):  
 r = list(r)  
 self.subject\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))  
 delbtn = QPushButton("Удалить")  
 self.subject\_table.setCellWidget(i, 1, delbtn)  
 delbtn.clicked.connect(lambda checked=None, j=i: self.\_del\_from\_subject(j, self.subject\_table))  
 self.subject\_table.setItem(i + 1, 0, QTableWidgetItem(''))  
 self.subject\_table.setItem(i + 1, 1, QTableWidgetItem(''))  
 self.subject\_table.setItem(i + 1, 2, QTableWidgetItem(''))  
 self.subject\_table.setItem(i + 1, 3, QTableWidgetItem(''))  
 self.subject\_table.removeCellWidget(i + 1, 4)  
 self.subject\_table.removeCellWidget(i + 1, 5)  
 addbtn = QPushButton("Добавить")  
 self.subject\_table.setCellWidget(i + 1, 1, addbtn)  
 addbtn.clicked.connect(lambda checked=None, j=i + 1: self.\_add\_subject(j, self.subject\_table))  
  
 def \_change\_day\_from\_table(self, rown, a, day):  
 row = list()  
 for i in range(a.columnCount()):  
 try:  
 row.append(a.item(rown, i).text())  
 except:  
 row.append(None)  
 if row[0] == '-' or row[0] == 'н' or row[0] == 'в':  
 if row[2] in work\_time:  
 self.cursor.execute("update time\_table set pos = '" + row[0] + "' where id = " + row[3] + ";")  
 self.cursor.execute("update time\_table set subject = '" + row[1] + "' where id = " + row[3] + "")  
 self.cursor.execute(  
 "update time\_table set start\_time = '" + row[2] + "' where id = " + row[3] + ";")  
 self.conn.commit()  
 else:  
 QMessageBox.about(self, "Error", "Введите стандартизированое время")  
 else:  
 QMessageBox.about(self, "Error", "Введите положение недели 'в' - верхняя 'н' - няжняя '-' - любая")  
 self.\_update\_day\_table(a, day)  
  
 def \_change\_teacher\_table(self, rown, a):  
 row = list()  
 for i in range(a.columnCount()):  
 try:  
 row.append(a.item(rown, i).text())  
 except:  
 row.append(None)  
 self.cursor.execute("update teacher set full\_name = '" + row[0] + "' where id = " + row[2] + ";")  
 self.conn.commit()  
  
 def \_del\_from\_time\_table(self, rown, a, day):  
 row = list()  
 for i in range(a.columnCount()):  
 try:  
 row.append(a.item(rown, i).text())  
 except:  
 row.append(None)  
 self.cursor.execute("delete from time\_table where id = " + row[3] + ";")  
 self.conn.commit()  
 self.\_update\_day\_table(a, day)  
  
 def \_del\_from\_teacher(self, rown, a):  
 row = list()  
 for i in range(a.columnCount()):  
 try:  
 row.append(a.item(rown, i).text())  
 except:  
 row.append(None)  
 self.cursor.execute("delete from teacher where id = " + row[2] + ";")  
 self.conn.commit()  
 self.\_update\_teacher\_table()  
  
 def \_del\_from\_subject(self, rown, a):  
 row = list()  
 A = list()  
 for i in range(a.columnCount()):  
 try:  
 row.append(a.item(rown, i).text())  
 except:  
 row.append(None)  
 for i in range(self.teacher\_table.rowCount()):  
 try:  
 A.append(self.teacher\_table.item(i, 1).text())  
 except:  
 row.append(None)  
 if row[0] in A:  
 QMessageBox.about(self, "Error",  
 "Нельзя удалить предмет, пока он находится в расписании или в преподавателях")  
 else:  
 try:  
 self.cursor.execute("delete from subject where name = '" + row[0] + "';")  
 self.conn.commit()  
 except:  
 QMessageBox.about(self, "Error",  
 "Нельзя удалить предмет, пока он находится в расписании или в преподавателях")  
 self.\_update\_subject\_table()  
  
 def \_add\_time\_table(self, rown, a, day):  
 row = list()  
 A = list()  
 for i in range(a.columnCount()):  
 try:  
 row.append(a.item(rown, i).text())  
 except:  
 row.append(None)  
 for i in range(self.subject\_table.rowCount()):  
 try:  
 A.append(self.subject\_table.item(i, 0).text())  
 except:  
 row.append(None)  
 if row[1] in A and row[1] != '':  
 if row[0] == '-' or row[0] == 'н' or row[0] == 'в':  
 if row[2] in work\_time:  
 self.cursor.execute(  
 "insert into time\_table(day, pos, subject, start\_time) values('" + day + "', '" + row[  
 0] + "', '" + row[1] + "', '" + row[2] + "');")  
 self.conn.commit()  
 self.\_update\_day\_table(a, day)  
 else:  
 QMessageBox.about(self, "Error", "Введите стандартизированое время")  
 else:  
 QMessageBox.about(self, "Error", "Введите положение недели 'в' - верхняя 'н' - няжняя '-' - любая")  
 else:  
 QMessageBox.about(self, "Error", "Такого предмета нет в БД")  
  
 def \_add\_teacher(self, rown, a):  
 A = list()  
 row = list()  
 for i in range(self.subject\_table.rowCount()):  
 try:  
 A.append(self.subject\_table.item(i, 0).text())  
 except:  
 row.append(None)  
 for i in range(a.columnCount()):  
 try:  
 row.append(a.item(rown, i).text())  
 except:  
 row.append(None)  
 if row[1] in A:  
 self.cursor.execute(  
 "insert into teacher(full\_name, subject) values('" + row[0] + "', '" + row[1] + "');")  
 self.conn.commit()  
 self.\_update\_teacher\_table()  
 else:  
 QMessageBox.about(self, "Error", "Такого предмета нет в БД")  
  
 def \_add\_subject(self, rown, a):  
 row = list()  
 for i in range(a.columnCount()):  
 try:  
 row.append(a.item(rown, i).text())  
 except:  
 row.append(None)  
 try:  
 self.cursor.execute("insert into subject(name) values('" + row[0] + "');")  
 self.conn.commit()  
 except:  
 QMessageBox(self, "Error", "Enter all fields")  
 self.\_update\_subject\_table()  
  
 def \_update(self):  
 self.\_update\_day\_table(self.monday\_table, 'Пн')  
 self.\_update\_day\_table(self.tuesday\_table, 'Вт')  
 self.\_update\_day\_table(self.wednesday\_table, 'Ср')  
 self.\_update\_day\_table(self.thursday\_table, 'Чт')  
 self.\_update\_day\_table(self.friday\_table, 'Пт')  
 self.\_update\_teacher\_table()  
 self.\_update\_subject\_table()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 win = Window()  
 win.show()  
 sys.exit(app.exec\_())