Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

«База данных для водителей и автотранспортных средств»

Студент гр. 753503
Волков А.В.
Руководитель
Удовин И.А.

Минск 2020

Содержание

1. Описание проекта

1.1 Постановка задачи

Целью проекта было разработать консольное приложение для управления базой данных водителей и автотранспортных средств. В программе должно быть реализовано:

- Возможность добавления новых строк в таблицы
- Возможность редактировать таблицы (удаление/обновление значений)
- Возможность поиска строки по значению одной из колонок ·

Возможность вывода таблиц в удобной форме

Так же должно быть выделено 7-8 сущностей и описаны их связи. Каждую из таблиц следует наполнить 10 записями.

2. Используемые технологии

Приложение “База данных для водителей и автотранспортных средств” реализовано на языке Python в среде программирования Visual Studio Code с использованием библиотек mysql и pandas. MySQL Server(v. 5.7) был установлен на google cloud на ip: *34.125.16.215.*

2.1. Visual Studio Code

Visual Studio Code - это легкий, но мощный редактор исходного кода, который работает на вашем рабочем столе и доступен для Windows, macOS и Linux. Он поставляется со встроенной поддержкой JavaScript, TypeScript и Node.js и имеет богатую экосистему расширений для других языков (таких как C ++, C #, Java, Python, PHP, Go) и сред выполнения (таких как .NET и Unity). Имеет широкие

возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом, но готовые сборки распространяются под проприетарной лицензией.

2.2. Язык программирования Python

Python - высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций. Python поддерживает структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное программирование. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений, высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты. Эталонной реализацией Python является интерпретатор CPython, поддерживающий большинство активно используемых платформ. Он распространяется под свободной лицензией Python Software Foundation License, позволяющей использовать его без ограничений в любых приложениях, включая проприетарные. Есть реализация интерпретатора для JVM с возможностью компиляции, CLR, LLVM, другие независимые реализации. Проект PyPy использует JIT-компиляцию, которая значительно увеличивает скорость выполнения Python-программ.

3. Программная реализация

В данном разделе будет рассмотрена архитектура приложения, а также будут приведены примеры некоторых пользовательских функций и хранимых процедур.

3.1. Схема базы данных

В данном приложении было выделено 9 сущностей. Схема базы данных имеет следующий вид:

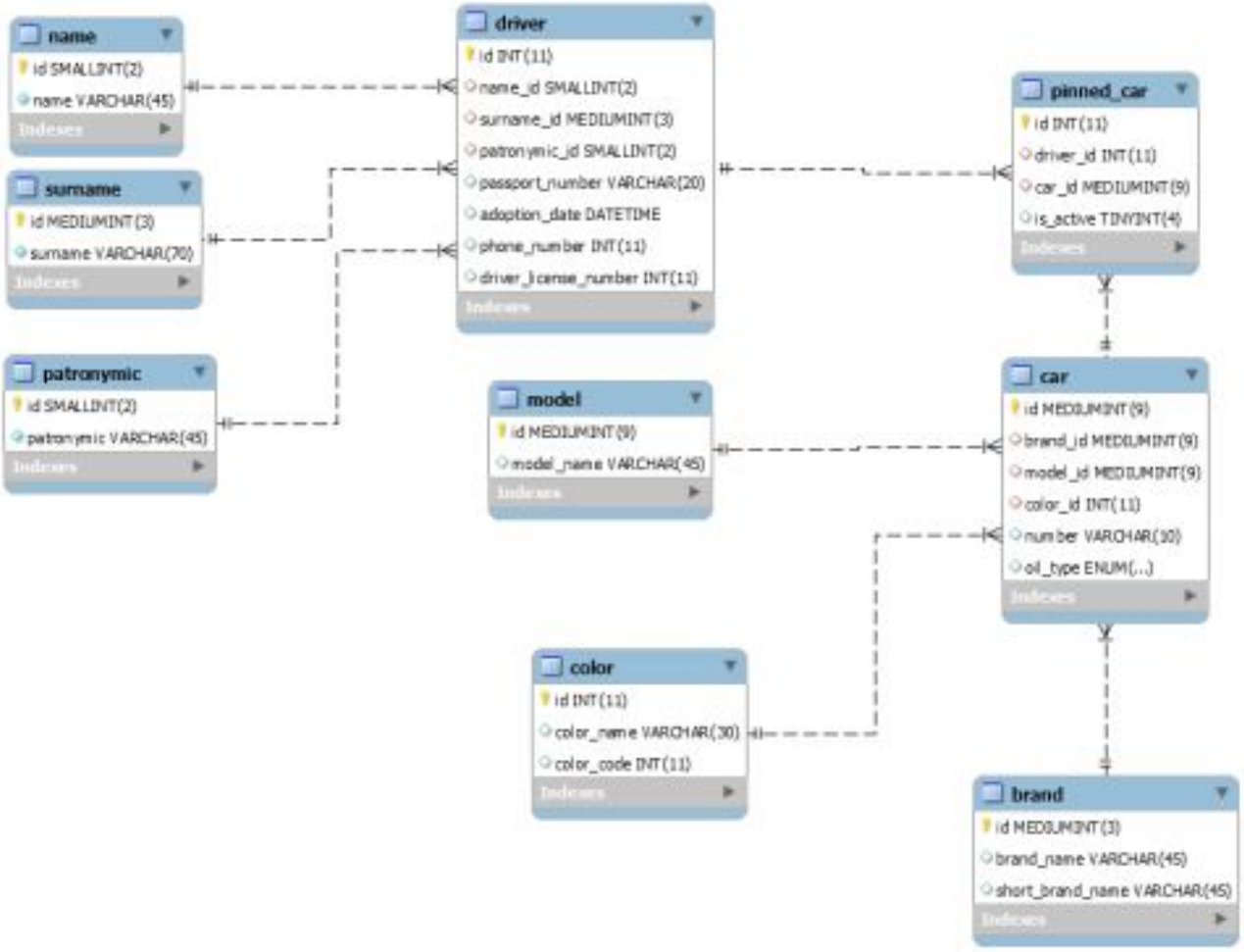


Рисунок 3.3. - Демонстрация схемы базы данных.

Описание таблиц и их предназначения:

· name – таблица содержащая уникальные имена водителей · surname - таблица содержащая уникальные фамилии водителей · patronymic - таблица содержащая уникальные отчества водителей · driver – таблица содержащая id имен, фамилий, отчеств водителей, а так же их номера телефона, номера водительских удостоверений, номера паспортов и даты принятия на работу

· pinned_car – таблица содержащая информацию о закрепленных за водителями машинах: id водителя и id машины, а так же флаг активности данного закрепления.

· color – таблица содержащая информацию о цветах(название и код)

· brand – таблица содержащая информацию о марках автомобилей · model – таблица содержащая информацию о моделях автомобилей · car – таблица содержащая информацию о машинах.

3.2 Описание работы приложения

Приложение запускается из консоли. Возможности приложения разбиты на пункты меню:

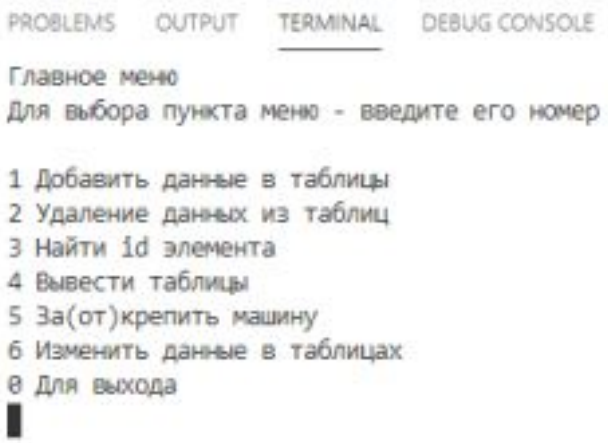


Рисунок 3.4.1 – Главное меню приложения.

Выбор осуществляется с помощью ввода номера интересующего пункта меню.

3.2.1 Добавление новых данных

Первый пункт меню содержит возможность добавления новых строк в таблицы:

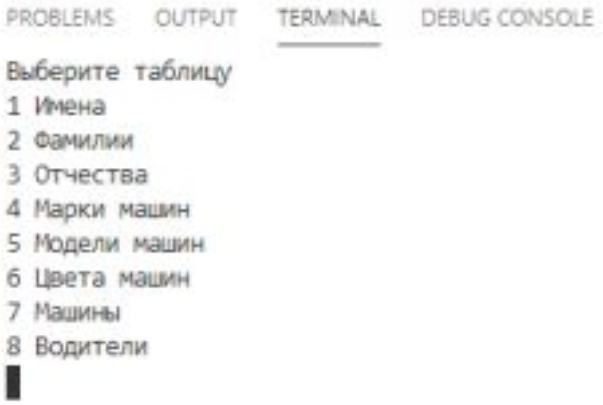
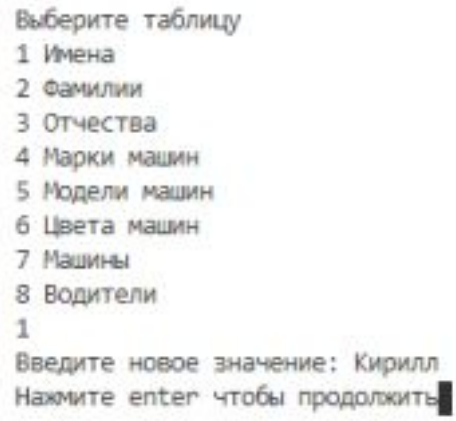


Рисунок 3.4.2 – Первый пункт меню.

Для добавления необходимо следовать инструкциям на



экране.

Рисунок 3.4.3 – Пример добавления нового имени.

При попытке добавления уже существующего имени пользователь получит сообщение об ошибке:

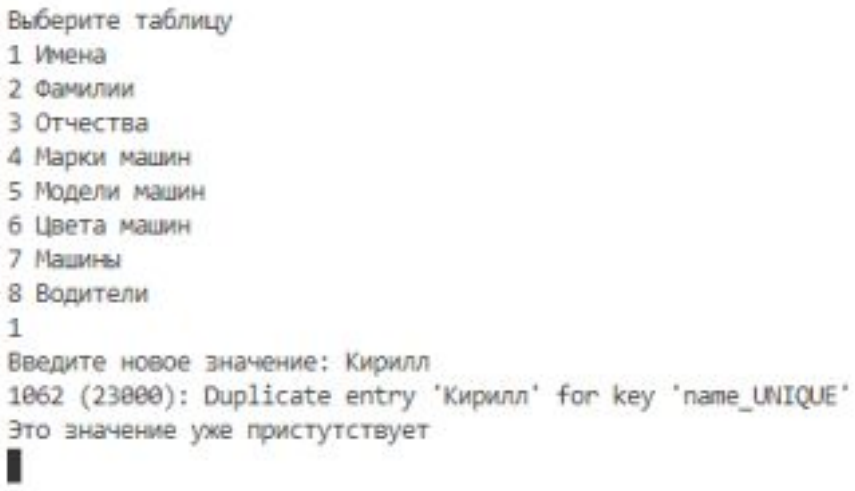


Рисунок 3.4.4 – Пример ввода уже существующего имени.

Вот ещё несколько примеров добавлений новых строк:

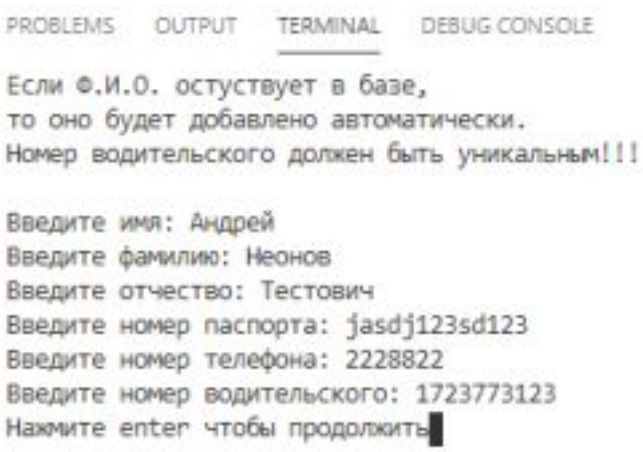


Рисунок 3.4.5 – Пример добавления водителя.

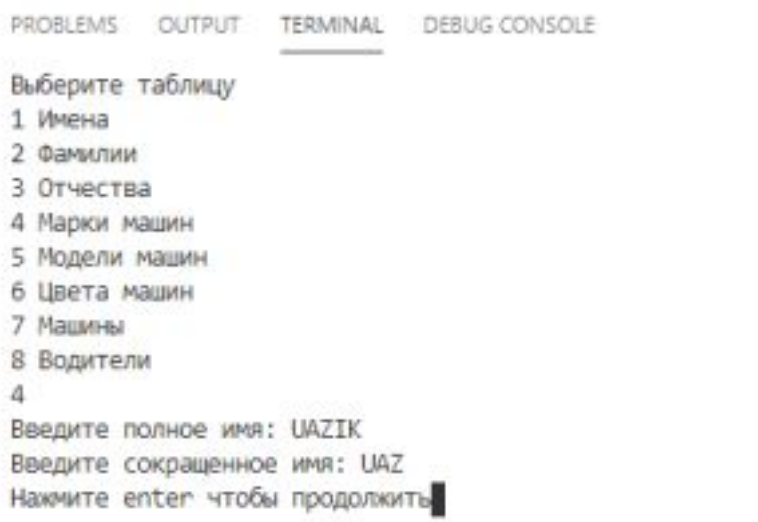


Рисунок 3.4.6 – Пример добавления марки автомобиля.

Стоит заметить, что добавление в таблицу закрепленных машин осуществляется в отдельном пункте меню: 5.

3.2.2 Удаление строк

Удаление осуществляется по id записи. Изначально пользователю опять же предлагается выбрать таблицу:

Рисунок 3.4.7 – Выбор таблицы.

Затем пользователя просят ввести id записи.

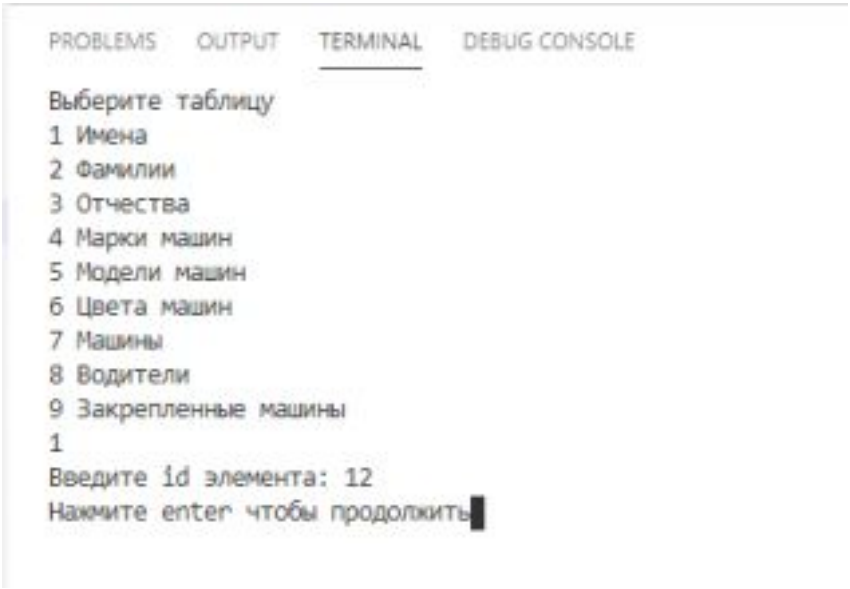


Рисунок 3.4.8 – Пример успешного удаления.

3.2.1 Поиск id записи

Как и в прошлых пунктах, изначально пользователь должен выбрать интересующую его таблицу. Затем следует ввести требующуюся от него информацию. Несколько примеров поиска:

Рисунок 3.4.9 – Пример поиска id модели.

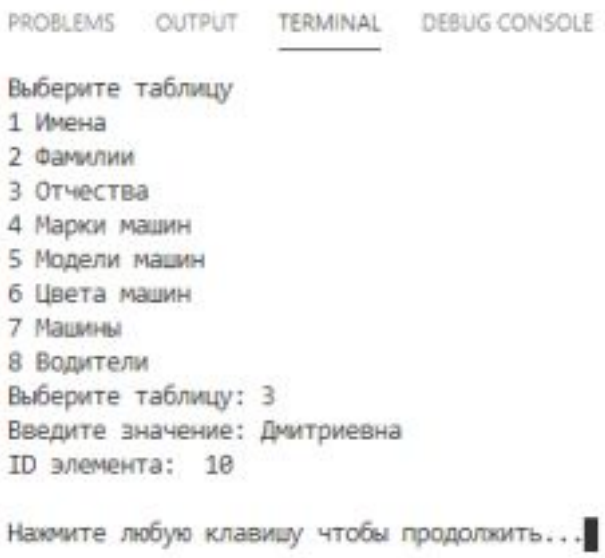


Рисунок 3.4.10 – Пример поиска id отчества.

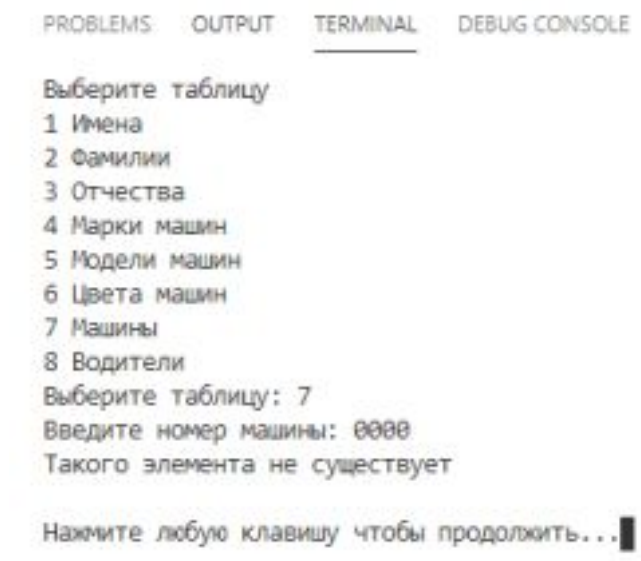


Рисунок 3.4.11 – Пример поиска id машины по несуществующему номеру.

3.2.4 Вывод таблиц

Вывод осуществляется с помощью библиотеки pandas. Как и в

предыдущих пунктах изначально пользователь выбирает таблицу. Вывод осуществляется по 10 записей, так же пользователь в любой момент может его прервать. В таблицу водителей дополнительно реализовано возможность отсортировать записи по определенной колонке. Примеры работы в данном пункте меню:

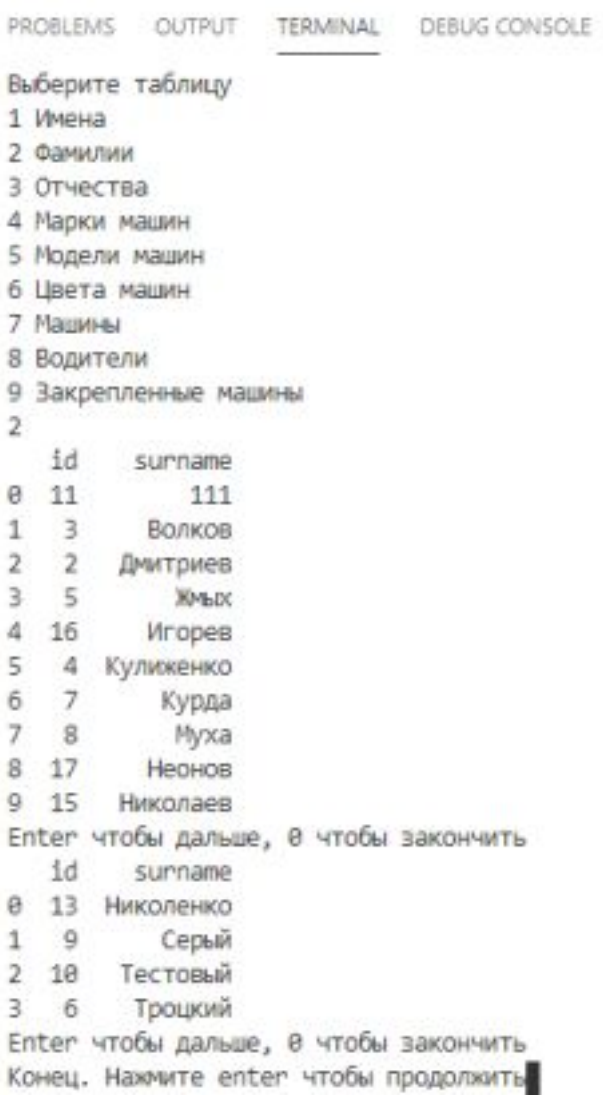


Рисунок 3.4.12 – Пример вывода таблицы фамилий.

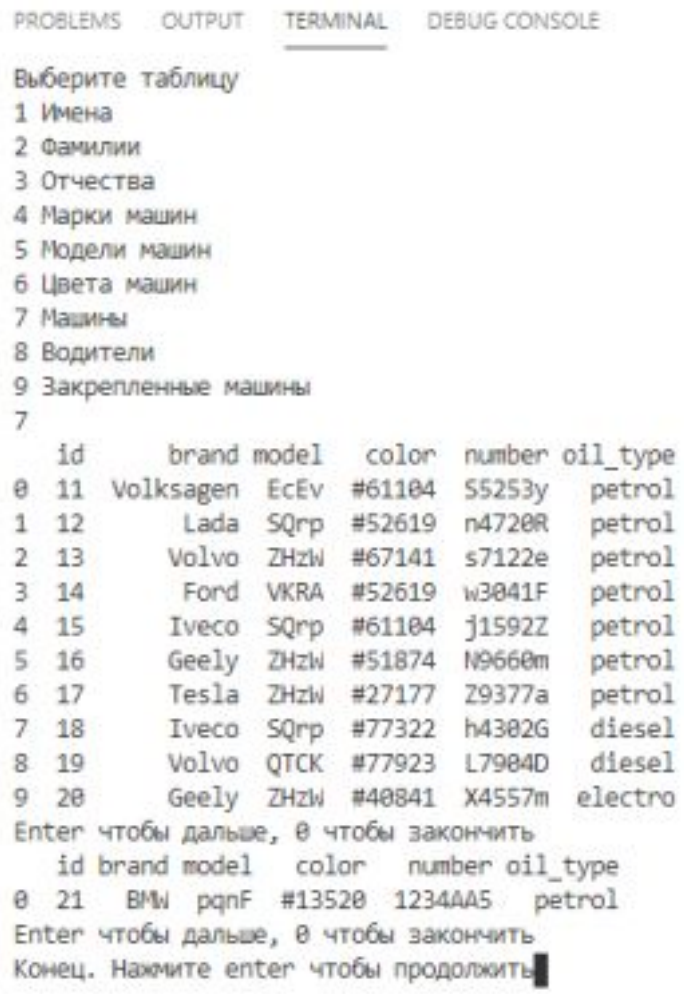


Рисунок 3.4.12 – Пример вывода таблицы автомобилей.

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
Выберите таблицу
1 Имена
2 Фамилии
3 Отчества
4 Марки машин
5 Модели машин
6 Цвета машин
7 Машины
8 Водители
9 Закрепленные машины
9
Вывести:
1 Только активные закрепления
2 Все закрепления
 > 2
   id       name    surname  patronymic      brand model car_number  is_active
0   1  Александр   Тестовый  Данииловна  Volksagen  EcEv     S5253y          1
1   3  Александр   Тестовый  Данииловна        BMW  pqnF    1234AA5          1
2   4      Армен  Кулиженко  Данииловна      Iveco  SQrp     j1592Z          1
3   5      Роман    Троцкий    Егорович      Volvo  QTCK     L7904D          0
4   6  Владислав      Курда    Егорович       Lada  SQrp     n4720R          1
5   7      Роман    Троцкий    Егорович      Iveco  SQrp     h4302G          1
6  11  Владислав      Курда    Егорович      Geely  ZHzW     N9660m          1
7  12      Армен  Кулиженко  Данииловна      Volvo  ZHzW     s7122e          1
8  13    Дмитрий   Тестовый  Дмитриевич      Tesla  ZHzW     Z9377a          1
9  14     Андрей     Неонов    Тестович      Geely  ZHzW     X4557m          1
Enter чтобы дальше, 0 чтобы закончить
```

Рисунок 3.4.12 – Пример вывода закрепленных автотранспортных средств.

3.2.5 Закрепление автомобилей

Данный пункт меню содержит два подпункта: создание новый закреплений и изменение старых:

Рисунок 3.4.13 – Подпункты меню.

Изменение осуществляется по id закрепления:

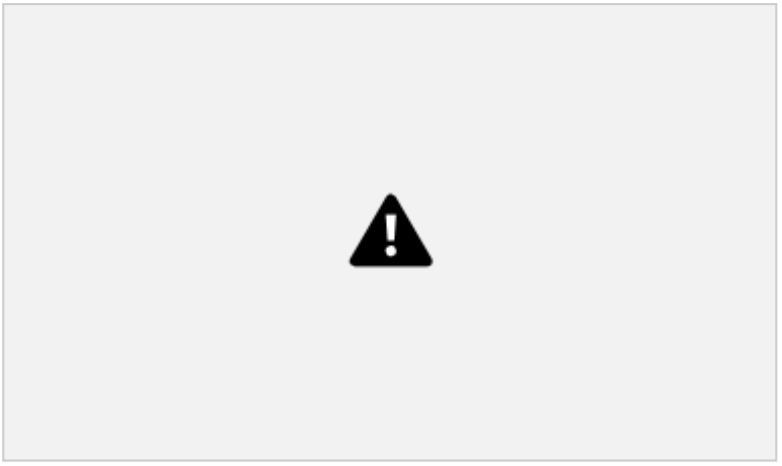


Рисунок 3.4.13 – Пример снятия закрепления автомобиля. При вводе не валидного id пользователь получит сообщение об ошибке:

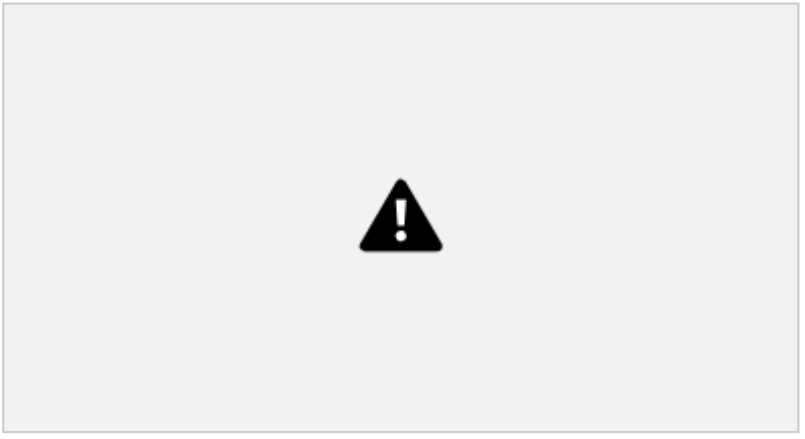


Рисунок 3.4.15 – Пример ошибки.

Добавление осуществляется по id водителя и id машины. Перед добавлением присутствует проверка на наличие таких записей.

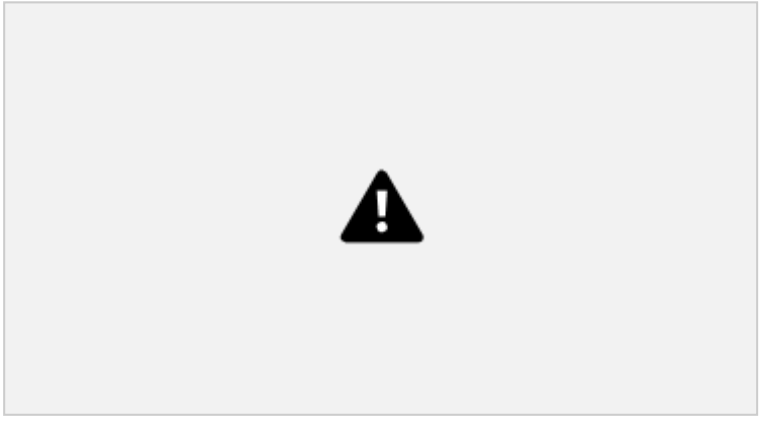


Рисунок 3.4.16 – Пример успешного закрепления автомобиля.

3.2.6 Изменение существующих записей

Чтобы изменить запись, пользователь должен выбрать таблицу, а затем следовать инструкциям. Примеры работы:

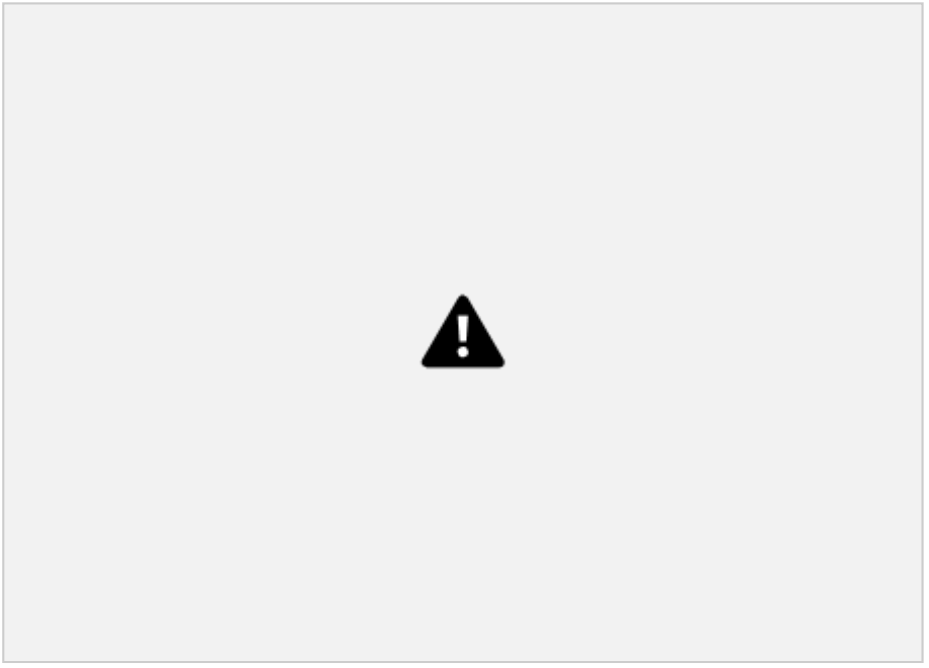


Рисунок 3.4.16 – Пример успешного изменения имени.

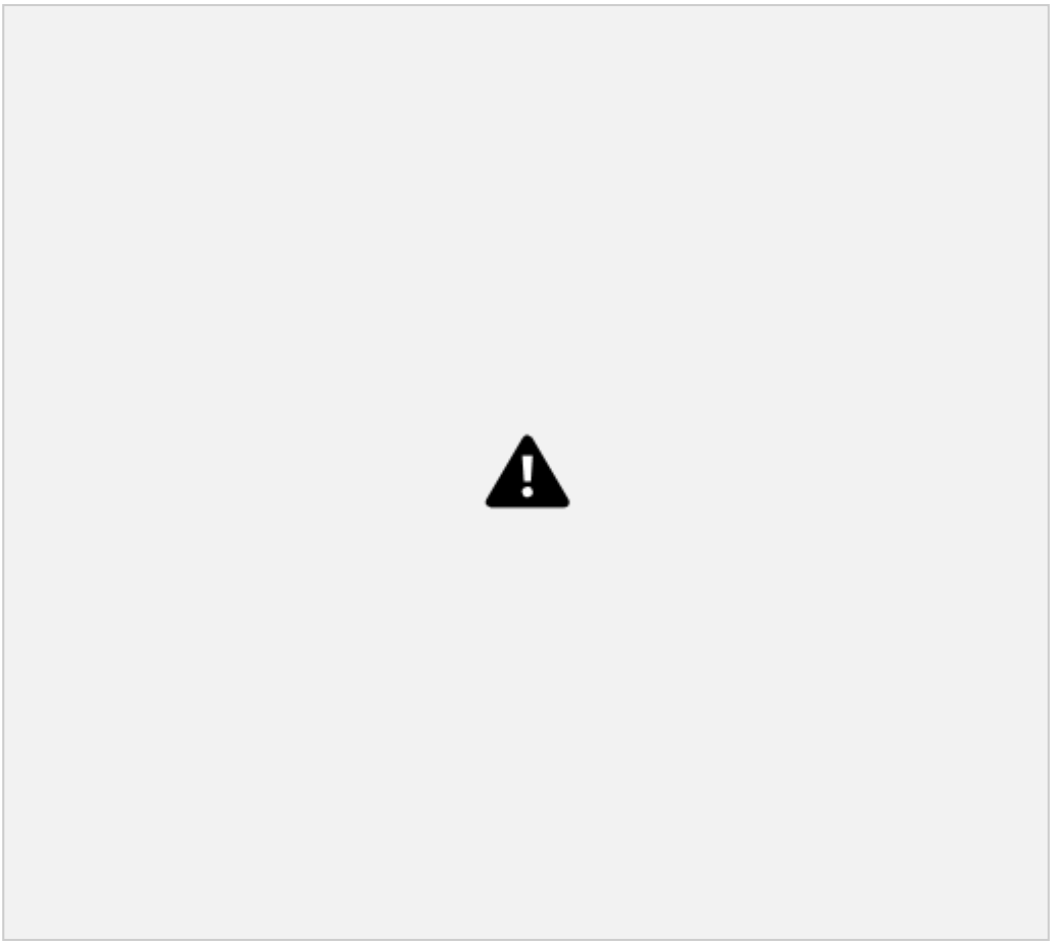


Рисунок 3.4.17 – Пример изменения типа топлива.

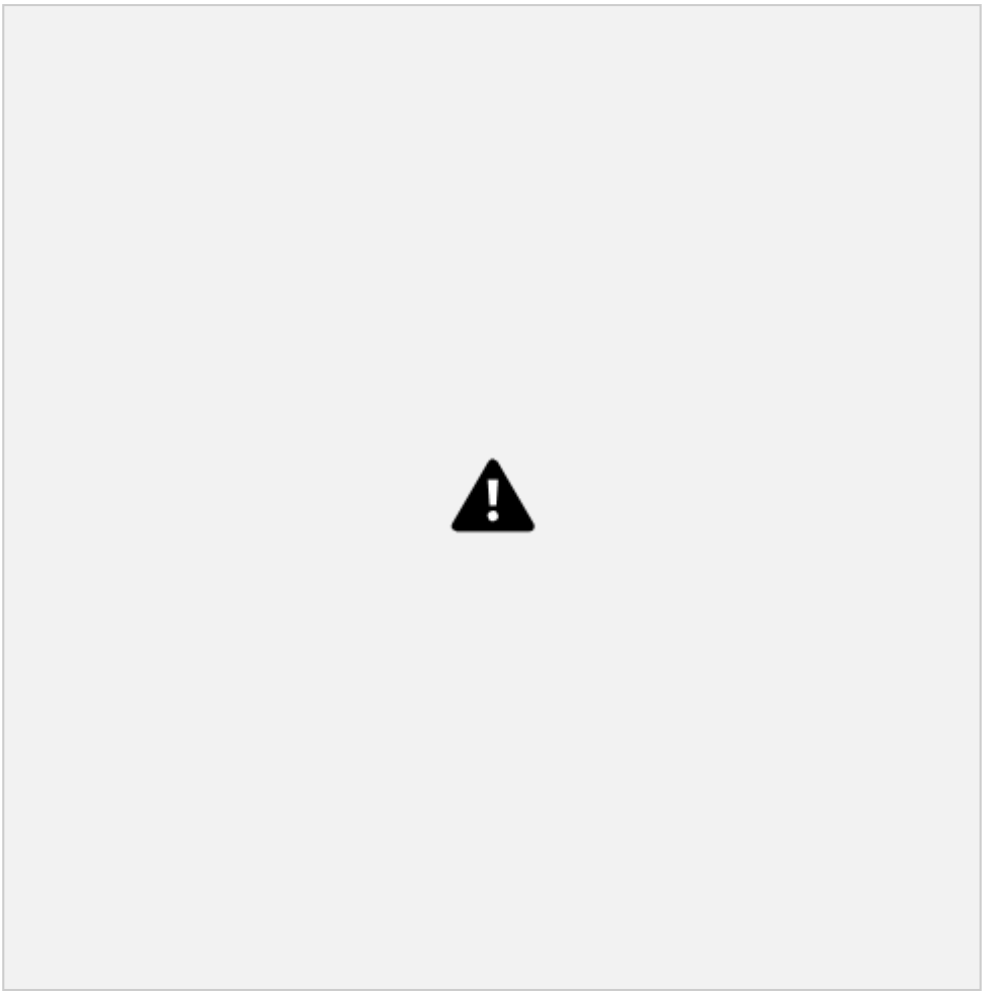


Рисунок 3.4.18 – Пример изменения имени водителя.

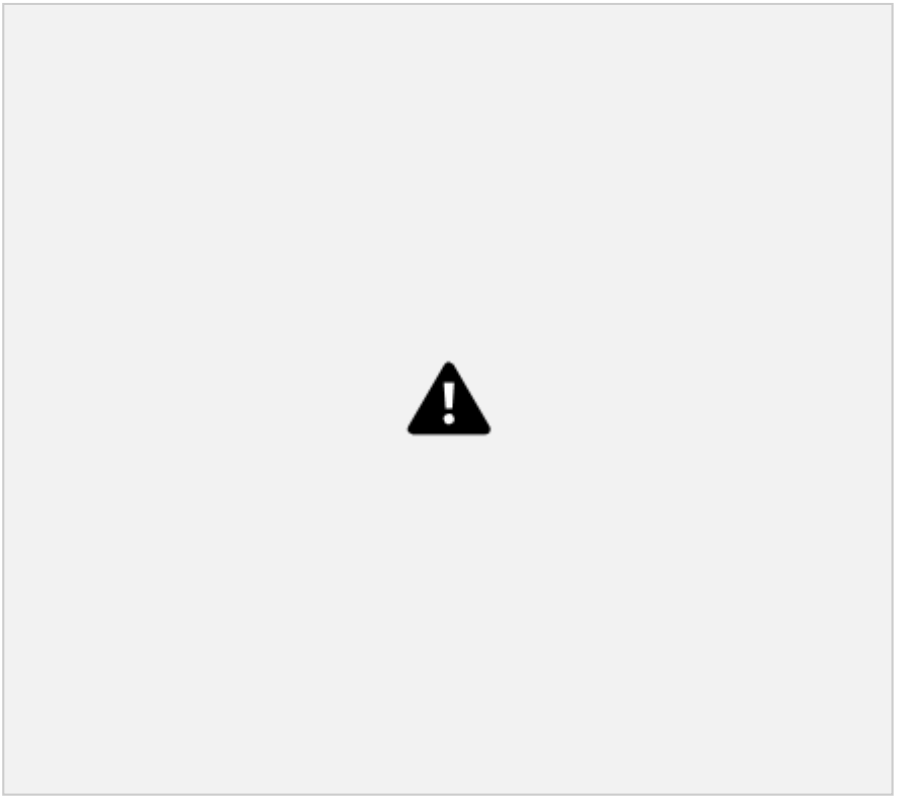


Рисунок 3.4.19 – Пример изменения имени цвета автомобиля.

Заключение

В ходе курсовой работы были закреплены, систематизированы и расширены теоретические знания. Было разработано приложение для работы с базой данных водителей и автотранспортных средств. Разработка осуществлялась с помощью языка Python и СУБД MySQL(v. 5.7). В результате работы программы мы получили протестированное приложение, готовое к использованию пользователями.

Основная цель работы была выполнена, то есть была разработана база данных водителей и автотранспортных средств. Программа выполняется через команды в терминале.

Приложение Текст программы

Файл main.py:

```
import time
from os import system, name
from datetime import datetime

import mysql.connector
from mysql.connector import Error
import pandas as pd
```

```
from test_data import fill_tables
from sql_helper import tables, insert_queries,\
 table_columns
from config import HOST_NAME, DATABASE,\
 USER_NAME, PASSWORD,\
 IS_COLD_START


def clear():
 system('cls') if name == 'nt' else system('clear')


def get_database():
 try:
   database   =   mysql.connector.connect(host=HOST_NAME,   database=DATABASE,
user=USER_NAME, password=PASSWORD)  if database.is_connected():
 return database
 except Error as e:
 print("Error while connecting to MySQL", e)
 exit()

 print("Error while connecting to MySQL")
 exit()


def print_start_menu():
 print('Главное меню\nДля выбора пункта меню -
введите его номер\n')  print('1 Добавить
данные в таблицы')
 print('2 Удаление данных из таблиц')
 print('3 Найти id элемента')
 print('4 Вывести таблицы')
 print('5 За(от)крепить машину')
 print('6 Изменить данные в таблицах')
 print('0 Для выхода')


################################################################################
# ##### INSERT, DELETE, EXIST, GET_ID, SELECT, UPDATE
```



```
################################################################################


def insert(query, values):
 cursor = database.cursor()
 try:
 cursor.execute(query, values)
 database.commit()
 except Exception as e:
```

```
 print(e)
 print('Это значение уже пристутствует')
 database.rollback()
 time.sleep(2)


def exist(column_name, column_value, table_name):
 cursor = database.cursor()
 cursor.execute(f'SELECT count(1) FROM {table_name} WHERE {column_name}="{column
_value}";')
 value = cursor.fetchone()
 if value[0]:
  return True
 else:
 return False


def delete(table_name, idx):
 cursor = database.cursor()
 cursor.execute(f'DELETE FROM {table_name} WHERE id = "{idx}";')
database.commit()
 input('Нажмите enter чтобы продолжить')


def get_last_id():
 cursor = database.cursor()
 cursor.execute('SELECT LAST_INSERT_ID();')
 return cursor.fetchone()[0]


def get_id(column_name, column_value, table_name):
 cursor = database.cursor()
 cursor.execute(f'SELECT id FROM {table_name} WHERE {column_name}="{column_value
}";')
 return cursor.fetchone()[0]


def add_value(table_name):
 if table_name not in {'car', 'driver', 'brand', 'color'}:
 value = input('Введите новое значение: ')
 insert(insert_queries[table_name], (value,))

 elif table_name == 'brand':
```



```
 full_name = input('Введите полное имя: ')
 short_name = input('Введите сокращенное имя: ')
 insert(insert_queries[table_name], (full_name, short_name))

 elif table_name == 'color':
 code = input('Введите код цвета: ')
 color_name = input('Введите имя цвета(можно
```

```
оставить пустым): ')
 if not len(color_name):
 color_name = '#' + str(code)
 insert(insert_queries[table_name], (color_name, code))

 elif table_name == 'driver':
 clear()
 print('Если Ф.И.О. остуствует в базе, \nто оно будет
добавлено автоматическ и.\n'
 'Номер водительского должен быть
уникальным!!!\n')  name = input('Введите имя:
')
 surname = input('Введите фамилию: ')
 patronymic = input('Введите отчество: ')
 passport_number = input('Введите номер паспорта: ')
 adoption_date = datetime.today().strftime('%Y-%m-%d %H:%M:%S')
phone_number = input('Введите номер телефона: ')
 license_number = input('Введите номер водительского: ')

 if not exist('name', name, 'name'):
 insert(insert_queries['name'], (name,))
 name_id = get_last_id()
 else:
 name_id = get_id('name', name, 'name')

 if not exist('surname', surname, 'surname'):
 insert(insert_queries['surname'], (surname,))
 surname_id = get_last_id()
 else:
 surname_id = get_id('surname', surname, 'surname')

 if not exist('patronymic', patronymic, 'patronymic'):
 insert(insert_queries['patronymic'], (patronymic,))
patronymic_id = get_last_id()
 else:
 patronymic_id = get_id('patronymic', patronymic, 'patronymic')

 values = (name_id, surname_id, patronymic_id, passport_number,
adoption_date, phone_number, license_number)

 insert(insert_queries[table_name], values)
 elif table_name == "car":
 clear()
 print('Номер автомобиля должен быть уникальным!!!\n')
```



```
 brand = None
 while True:
 brand = input('Введите марку автомобиля(-1
чтобы выйти): ')  try:
```

```
brand_id = get_id('brand_name', brand, 'brand')  break
except:
if brand == '-1':
return
print('Не существует такой марки.')
input('Enter чтобы продолжить')
return

model = None
while True:
 model = input('Введите модель автомобиля(-1
чтобы выйти): ')  try:
model_id = get_id('model_name', model, 'model')  break
except:
if model == '-1':
return
print('Не существует такой модели.')
input('Enter чтобы продолжить')
return

color_code = None
while True:
color_code = input('Введите код цвета(-1 чтобы
выйти): ')  try:
color_id = get_id('color_code', color_code, 'color')  break
except:
if color_code == '-1':
return
print('Не существует такго цвета.')
input('Enter чтобы продолжить')
return

number = input('Введите номер автомобиля: ')

oil_type = None
while oil_type not in {'б', 'д', 'э'}:
oil_type = input('Введите тип топлива(б-Бензин,
д-Дизель, Э электро): ')

if oil_type == 'б':
oil_type = 'petrol'
elif oil_type == 'д':
oil_type = 'diesel'
else:
```



```
oil_type = 'electro'

values = (brand_id, model_id, color_id, number, oil_type)
insert(insert_queries[table_name], values)
input('Нажмите enter чтобы продолжить')
```

```
def find_value_id(table_name):
 column_name = table_name
 if table_name in {'brand', 'model', 'color'}:
 column_name += '_name'
 value = input('Введите значение: ')
 try:
 idx = get_id(column_name, value, table_name)
 except:
 print('Такого элемента не существует')
 input('\nНажмите любую клавишу чтобы
продолжить...') return
 elif table_name == 'car':
 car_number = input('Введите номер машины: ')
 try:
 idx = get_id('number', car_number, table_name)
 except:
 print('Такого элемента не существует')
 input('\nНажмите любую клавишу чтобы
продолжить...') return
 elif table_name == 'driver':
 passport_number = input('Введите номер паспотра: ')
 try:
 idx = get_id('passport_number', passport_number, table_name) except:
 print('Такого элемента не существует')
 input('\nНажмите любую клавишу чтобы
продолжить...') return
 else:
 value = input('Введите значение: ')
 try:
 idx = get_id(table_name, value, table_name)
 except:
  print('Такого элемента не существует')
 input('\nНажмите любую клавишу чтобы
продолжить...') return
 print('ID элемента: ', idx)
 input('\nНажмите любую клавишу чтобы продолжить...')


def show_one_table(table_name):
 cursor = database.cursor()
 # if table_name == 'driver':
 # val = input('Отсортировать по:\n1 Имя\n2 Фамилия\n3
Отчество\n4 Дате принятия на работу')
```



```
 # quety = 'Order by <SomeColumn> Offset {offset} ROWS LIMIT 10;'  # if
val in set(range(1, 4)):

 if table_name not in {'car', 'driver', 'pinned_car'}:
```

```
 query = f'SELECT * FROM {table_name};'
 elif table_name == 'car':
 query = ('SELECT car.id, brand.brand_name, model.model_name, color.color_n ame,
number, oil_type '
 'FROM car_agency.car '
 'LEFT OUTER JOIN brand ON brand.id = car.brand_id '  'LEFT OUTER
JOIN model ON model.id = car.model_id '  'LEFT OUTER JOIN color ON
color.id = car.color_id;')  elif table_name == 'driver':
 sort_val = None
 while sort_val not in set(['1', '2', '3', '4']):
 sort_val = input('О т с о р т и р о в а т ь  п о :\n1  И м я \n2  Ф а м и л и я \n3
О т ч е с т в о \n4  Д а  т е  п р и н я т и я  н а  р а б о т у \n > ')

  query  =  ('SELECT  driver.id,  name,  surname.surname,  patronymic.patronymic,  '
'passport_number, adoption_date, phone_number, driver_license_numb er '
 'FROM car_agency.driver '
 'LEFT OUTER JOIN name ON name.id = driver.name_id '  'LEFT OUTER JOIN surname ON
surname.id = driver.surname_id '  'LEFT OUTER JOIN patronymic ON patronymic.id =
driver.patronymic_i d')
 if sort_val == '1':
 query += ' ORDER BY name;'
 elif sort_val == '2':
 query += ' ORDER BY surname;'
 elif sort_val == '3':
 query += ' ORDER BY patronymic;'
 else:
 query += ' ORDER BY adoption_date;'
 elif table_name == 'pinned_car':
 filter_val = None
 while filter_val not in set(['1', '2']):
 filter_val = input('В ы в е с т и :\n1  Т о л ь к о  а к т и в н ы е
з а к р е п л е н и я \n2  В с е  з а к р  е п л е н и я \n > ')

 query = ('SELECT pinned_car.id, name, surname, patronymic, brand_name, '  '
model_name, car.number, is_active '
 'FROM car_agency.pinned_car '
 'INNER JOIN car ON car.id = pinned_car.car_id '  ' LEFT OUTER JOIN brand ON
car.brand_id = brand.id '  ' LEFT OUTER JOIN model ON car.model_id = model.id '
'INNER JOIN driver ON driver.id = pinned_car.driver_id '  ' LEFT OUTER JOIN name
ON driver.name_id = name.id '  ' LEFT OUTER JOIN surname ON driver.surname_id =
surname.id '  ' LEFT OUTER JOIN patronymic ON driver.patronymic_id = patronym
ic.id')
```



```
if filter_val == '1':
 query += ' WHERE is_active=1;'
 elif filter_val == '2':
 query += ';'

 cursor.execute(query)
```

```
 data = cursor.fetchall()

 for offset in range(0, 10*10000, 10):
 df = pd.DataFrame(data=data[offset: offset + 10], columns=table_columns[tab
le_name])
 if not len(df):
 input('К о н е ц .  Н а ж м и т е  enter  ч т о б ы  п р о д о л ж и т ь')
 break
 print(df)

 user_input = input('Enter  ч т о б ы  д а л ь ш е ,  0  ч т о б ы
з а к о н ч и т ь')  if user_input == '0':
 break


def update(column_name, column_value, table_name, idx):
 cursor = database.cursor()
 cursor.execute(f'UPDATE {table_name} SET {column_name}="{column_value}" WHERE i
d={idx};')
 database.commit()


def pin_car():
 user_input = None
 while user_input not in ('0', '1'):
 user_input = input('0 С м е н и т ь  з н а ч е н и е\n1 Д о б а в и т ь\n')

 if user_input == '0':
 idx = input('В в е д и т е  id  э л е м е н т а :  ')
 if not exist('id', idx, 'pinned_car'):
 print('Н е т  т а к о г о  э л е м е н т а .')
 input('\nН а ж м и т е  л ю б у ю  к л а в и ш у  ч т о б ы
п р о д о л ж и т ь . . . ')  return

 new_value = None
 while new_value not in {'0', '1'}:
 new_value = input('В в е д и т е  н о в о е  з н а ч е н и е :\n0  А к т и в н о е
з а к р е п л е н и е '  '\n1  Н е а к т и в н о е  з а к р е п л е н и е\n')
 update('is_active', new_value == '0', 'pinned_car', idx)
else:
 driver_id = input('В в е д и т е  id  в о д и т е л я :  ')
 if not exist('id', driver_id, 'driver'):
 print('Н е т  т а к о г о  в о д и т е л я')
 time.sleep(2)
```



```
 return

 car_id = input('В в е д и т е  id  м а ш и н ы :  ')
 if not exist('id', car_id, 'car'):
 print('Н е т  т а к о й  м а ш и н ы')
```

```
 time.sleep(2)
 return

 insert(insert_queries['pinned_car'], (driver_id, car_id, True))
input('\nНажмите любую клавишу чтобы
продолжить...')


def update_table(table_name):
 idx = input('Введите id элемента: ')
 if not exist('id', idx, table_name):
 print('Такого элемента не существует')
 input('Enter чтобы продолжить...')
 return

 if table_name not in {'car', 'driver', 'brand', 'color', 'pinned_car'}:
column_name = table_name
 new_value = input('Введите новое значение: ')
 if table_name == 'model':
 column_name = table_name + '_name'
 update(column_name, new_value, table_name, idx)

 elif table_name == 'brand':
 user_input = None
 while user_input not in {'1', '2'}:
 user_input = input('Выберите:\n1 Полное имя\n2
Сокращенное имя\n')
 new_value = input('Введите новое значение: ')
 column_name = 'brand_name' if user_input == '1' else 'short_brand_name'
update(column_name, new_value, table_name, idx)

 elif table_name == 'color':
 user_input = None
 while user_input not in {'1', '2'}:
 user_input = input('Выберите:\n1 Имя цвета\n2 Код
цвета\n')
 new_value = input('Введите новое значение: ')
 column_name = 'color_name' if user_input == '1' else 'color_code'
update(column_name, new_value, table_name, idx)

 elif table_name == 'driver':
 print('Номер водительского должен быть
уникальным!!!\n')

 user_input = None
 while user_input not in set(map(str, range(1, 7))):
 print('Выберите:\n1 Имя\n2 Фамилия\n3 Отчество')
print('4 Номер паспорта\n5 Номер телефона\n6 Номер
водительского')
```

```
 user_input = input('> ')

 if user_input == '1':
 column_name = 'name'
 elif user_input == '2':
 column_name = 'surname'
 elif user_input == '3':
 column_name = 'patronymic'
 elif user_input == '4':
 column_name = 'passport_number'
 elif user_input == '5':
 column_name = 'phone_number'
 elif user_input == '6':
 column_name = 'driver_license_number'

 new_value = input('Введите новое значение: ')
 if user_input in {'1', '2', '3'}:
 if not exist(column_name, new_value, column_name):  input('Такого
значения не существует.\nEnter чтобы продолжить..')
return
 else:
 new_value_idx = get_id(column_name, new_value, column_name)
update(column_name+'_id', new_value_idx, table_name, idx)  else:
 update(column_name, new_value, table_name, idx)

 elif table_name == "car":
 print('Номер автомобиля должен быть
уникальным!!!\n')  user_input = None
 while user_input not in set(map(str, range(1, 6))):
 print('Выберите:\n1 Марка\n2 Модель\n3 Цвет')
 print('4 Номер\n5 Тип топлива')
 user_input = input('> ')

 if user_input == '1':
 column_name = 'brand'
 elif user_input == '2':
 column_name = 'model'
 elif user_input == '3':
 column_name = 'color'
 elif user_input == '4':
 column_name = 'number'
 elif user_input == '5':
 column_name = 'oil_type'

 new_value = input('Введите новое значение: ')
 if user_input in {'1', '2', '3'}:
 if not exist(column_name + '_name', new_value, column_name):
input('Такого значения не существует.\nEnter чтобы
продолжить..')  return
 else:
```

```
 new_value_idx = get_id(column_name + '_name', new_value, column_nam e)
  update(column_name+'_id', new_value_idx, table_name, idx)  else:
  update(column_name, new_value, table_name, idx)
  input('\nНажмите любую клавишу чтобы продолжить...')



################################################################################
# ##### WRAPPERS
################################################################################


def remove_value(table_name):
  idx = input('Введите id элемента: ')
  delete(table_name, idx)


def show_tables():
  print('Выберите таблицу')
  print('1 Имена\n2 Фамилии \n3 Отчества')
  print('4 Марки машин\n5 Модели машин \n6 Цвета машин')
  print('7 Машины\n8 Водители')


def add_new_data():
  clear()
  show_tables()

  user_input = input()
  if user_input in tables:
  add_value(tables[user_input])
  else:
  add_new_data()


def remove_data():
  clear()
  show_tables()
  print('9 Закрепленные машины')


  user_input = input()
  if user_input in tables:
  remove_value(tables[user_input])
  else:
  remove_data()


def find_id():
  clear()
```

```
 show_tables()
 user_input = input('Выберите
таблицу: ')
 if user_input in tables:
 find_value_id(tables[user_input])
else:
 find_id()


def show_data():
 clear()
 show_tables()
 print('9 Закрепленные
машины')  user_input =
input()

 if user_input in tables:
 show_one_table(tables[user_input])
else:
 show_data()


def update_table_data():
 show_tables()
 clear()
 show_tables()
 user_input = input('Выберите
таблицу: ')  print('\n')
 if user_input in tables:
 update_table(tables[user_input])
else:
 update_table_data()



if __name__ == '__main__':
 database = get_database()

 if IS_COLD_START:
 fill_tables(database)

 functions = {
 '0': exit,
 '1': add_new_data,
 '2': remove_data,
 '3': find_id,
 '4': show_data,
 '5': pin_car,
 '6': update_table_data
 }
```

```
 while True:
```



```
clear()
 print_start_menu()
 choise = input()
 clear()
 if choise in functions:
 functions[choise]()
```

Файл config.py:

```
HOST_NAME = '34.125.16.215'
DATABASE = 'car_agency'

USER_NAME = 'СКРЫТ'
PASSWORD = 'СКРЫТ'

IS_COLD_START = False # if True, programm will add test data into database
```

Файл test_data.py:

```
import string
import random
from datetime import datetime


def fill_tables(database):
 cursor = database.cursor()

 name_query = 'INSERT INTO name (name) VALUES (%s)'
 names = ['Андрей', 'Александр', 'Григорий', 'Владислав',
 'Роман', 'Борис', 'Даниил', 'Михаил',
'Дмитрий', 'Армен']  names = list(map(lambda x: (x, ),
names))

 surname_query = 'INSERT INTO surname (surname) VALUES (%s);'
surnames = ['Иванов', 'Дмитриев', 'Волков',
'Кулиженко',
 'Жмых', 'Троцкий', 'Курда', 'Муха', 'Серый',
'Тестовый']  surnames = list(map(lambda x: (x, ), surnames))

 patronymic_query = 'INSERT INTO patronymic (patronymic) VALUES (%s)'
patronymics = ['Иванович', 'Дмитриевич', 'Олегович',
'Егорович',  'Жмыхич', 'Андреевич', 'Михайлович',
'Владимировна', 'Даниловна', 'Дмитриевна']
 patronymics = list(map(lambda x: (x, ), patronymics))

 car_models_query = 'INSERT INTO model (model_name) VALUES (%s);'  car_models =
[''.join(random.choices(string.ascii_letters, k=4)) for _ in range (10)]
 car_models = list(map(lambda x: (x,), car_models))
```

```
 car_brands_query = 'INSERT INTO brand (brand_name, short_brand_name) VALUES (%s ,
%s);'
 brand_names = ['Ford', 'BMW', 'Volksagen', 'Lada', 'Geely', 'Fiat', 'Iveco', 'U
AZ', 'Volvo', 'Tesla']
```



```
 short_brand_names = ['Ford', 'BMW', 'VW', 'Lada', 'Geely', 'Fiat', 'Iveco', 'UA
Z', 'Volvo', 'Tesla']

 color_query = 'INSERT INTO color (color_name, color_code) VALUES (%s, %s);'
colors = [random.randint(1000, 1e5) for _ in range(10)]
 colors_names = list(map(lambda x: f'#{x}', colors))
 colors_val = list(zip(colors_names, colors))


 cursor.executemany(name_query, names)
 database.commit()

 cursor.executemany(surname_query, surnames)
 database.commit()

 cursor.executemany(patronymic_query, patronymics)
 database.commit()

 cursor.executemany(car_models_query, car_models)
 database.commit()

 cursor.executemany(car_brands_query, list(zip(brand_names, short_brand_names)))
database.commit()

 cursor.executemany(color_query, colors_val)
 database.commit()

 driver_query = 'INSERT INTO driver (name_id, surname_id, patronymic_id,'\
'passport_number, adoption_date, phone_number, driver_license_nu mber)'\
 ' VALUES (%s, %s, %s, %s, %s, %s, %s)'

 drivers_names_id = []
 drivers_surnames_id = []
 drivers_patronymic_id = []

 for _ in range(10):
 cursor.execute("SELECT id from name ORDER BY RAND() LIMIT 1;")
drivers_names_id.append(cursor.fetchone()[0])
 cursor.execute("SELECT id from surname ORDER BY RAND() LIMIT 1;")
drivers_surnames_id.append(cursor.fetchone()[0])
 cursor.execute("SELECT id from patronymic ORDER BY RAND() LIMIT 1;")
drivers_patronymic_id.append(cursor.fetchone()[0])

 driver_dates = [datetime.fromtimestamp(random.randint(1e9, 1e10)) for _ in rang
```

```
e(10)]
 driver_dates = list(map(lambda x: x.strftime('%Y-%m-
%d %H:%M:%S'), driver_dates))
 passport_numbers = [str(random.randint(1e4, 1e7)) for _ in range(10)]
phone_numbers = [random.randint(1e4, 1e7) for _ in range(10)]  driver_numbers
= [str(random.randint(1e7, 1e7 + 1000)) for _ in range(10)]
```



```
 driver_values = list(zip(drivers_names_id, drivers_surnames_id, drivers_patrony
mic_id,
 passport_numbers, driver_dates, phone_numbers, driver_n umbers))

 car_query = 'INSERT INTO car (brand_id, model_id, color_id, number, oil_type) V
ALUES'\
 ' (%s, %s, %s, %s, %s)'
 cars_brand_id = []
 cars_model_id = []
 cars_color_id = []
 letters = string.ascii_letters
 cars_number = [f'{random.choice(letters)}{random.randint(1e3, 1e4)}{random.choi
ce(letters)}'
 for _ in range(10)]
 cars_oils_type = (['petrol']*7) + ['diesel'] * 2 + ['electro']

 for _ in range(10):
 cursor.execute("SELECT id from brand ORDER BY RAND() LIMIT 1;")
cars_brand_id.append(cursor.fetchone()[0])
 cursor.execute("SELECT id from model ORDER BY RAND() LIMIT 1;")
cars_model_id.append(cursor.fetchone()[0])
 cursor.execute("SELECT id from color ORDER BY RAND() LIMIT 1;")
cars_color_id.append(cursor.fetchone()[0])

 cars_values = list(zip(cars_brand_id, cars_model_id, cars_color_id,
cars_number, cars_oils_type))

 pinned_car_query = 'INSERT INTO pinned_car (driver_id, car_id, is_active) VALUE S
(%s, %s, %s)'
 pinned_car_driver_id = []
 pinned_car_car_id = []

 cursor.execute("SELECT id from driver ORDER BY RAND() LIMIT 10;")
pinned_car_driver_id.extend(cursor.fetchall())

 cursor.execute("SELECT id from car ORDER BY RAND() LIMIT 10;")
pinned_car_car_id.extend(cursor.fetchall())
 pinned_car_val = list(zip(pinned_car_driver_id, pinned_car_car_id, [True]*9 + [
False]))

 cursor.executemany(driver_query, driver_values)
 database.commit()
```

```
 cursor.executemany(car_query, cars_values)
 database.commit()

 cursor.executemany(pinned_car_query, pinned_car_val)

 database.commit()
```


Файл config.py:

```
insert_queries = {
 'model': 'INSERT INTO model (model_name) VALUES (%s)',
'name': 'INSERT INTO name (name) VALUES (%s)',
 'surname': 'INSERT INTO surname (surname) VALUES (%s)',
 'patronymic': 'INSERT INTO patronymic (patronymic) VALUES (%s)',  'brand':
'INSERT INTO brand (brand_name, short_brand_name) VALUES (%s, %s) ',
 'color': 'INSERT INTO color (color_name, color_code) VALUES (%s, %s)',  'driver':
'INSERT INTO driver (name_id, surname_id, patronymic_id, '\  'passport_number,
adoption_date, phone_number, driver_license_num ber) '\
 'VALUES (%s, %s, %s, %s, %s, %s, %s)',
 'car': 'INSERT INTO car (brand_id, model_id, color_id, number, oil_type) VALUES'\
 ' (%s, %s, %s, %s, %s)',
 'pinned_car': 'INSERT INTO pinned_car (driver_id, car_id, is_active) VALUES (%s ,
%s, %s)',
}

tables = {
 '1': 'name',
 '2': 'surname',
 '3': 'patronymic',
 '4': 'brand',
 '5': 'model',
 '6': 'color',
 '7': 'car',
 '8': 'driver',
 '9': 'pinned_car'
}

table_columns = {
 'name': ['id', 'name'],
 'surname': ['id', 'surname'],
 'patronymic': ['id', 'patronymic'],
 'brand': ['id', 'brand_name', 'short_brand_name'],
 'model': ['id', 'model_name'],
 'color': ['id', 'color_name', 'color_code'],
 'car': ['id', 'brand', 'model', 'color', 'number', 'oil_type'],   'driver':
['id', 'name', 'surname', 'patronymic', 'passport_number',  'adoption_date',
'phone_number', 'driver_license_number'],  'pinned_car': ['id', 'name',
'surname', 'patronymic', 'brand', 'model',  'car_number', 'is_active']
}
```