Костылев МТ-302

11. БД "Гостиница".

Задача – информационная поддержка деятельности гостиницы.

БД должна осуществлять:

* ведение списка постояльцев;
* учёт забронированных мест;
* ведение архива выбывших постояльцев за последний год.

Необходимо предусмотреть:

* Получение списка свободных номеров (по количеству мест и классу);
* Получение списка номеров, освобождающихся сегодня и завтра;
* Выдачу информации по конкретному номеру;
* Автоматизацию выдачи счета на оплату номера;
* Получение списка заселенных номеров;
* Получение списка броней клиента.
* Получение списка сотрудников, которые работали в нужный день и время.

Сущности и их свойства:

* **Бронь** (id брони, id постояльца, дата въезда, дата выезда, номер забронированного номера, размер предоплаты)
* **Номер** (номер номера, кол-во мест в номере, класс номера, стоимость номера)
* **Состояние** (id состояния, код состояния, время смены состояния, дата смены состояния, id работающего сотрудника, номер номера)
* **Постоялец** (id, фио, телефон, паспортные данные, id организации)
* **Организация** (id, тип организации, название организации)
* **Сотрудник** (id, фио, должность, номер телефона, паспортные данные)
* **Сотрудник на смене** (id смены сотрудника, id сотрудника, дата и время начала смены, время смены)
* **Класс номера** (класс номера, базовая стоимость, добавочная стоимость за место)
* **Должность** (должность, ставка)
* **Код состояния** (код состояния, название состояния)

Отношения:

* **Организация – Постоялец:**

Связь: Один - Много

У одной организации может быть много сотрудников (потенциальных постояльцев), а у постояльца может быть только одна организация.

* **Постоялец – Бронь:**

Связь: Один – Много

У одного постояльца может быть много броней, а за каждой бронью прикреплен только один постоялец.

* **Бронь – Номер:**

Связь: Много – Один

У одного номера может быть много броней, а за каждой бронью прикреплен только один номер.

* **Номер – Состояние:**

Связь: Один – Много

У каждого номера может быть несколько состояний, а за каждым состоянием закреплен один номер.

* **Номер – Класс номера:**

Связь: Много – Один

Номеров с одинаковыми классами много, но у номера только один класс.

* **Состояние – Сотрудник на смене:**

Связь: Много – Один

Сотрудник на смене, за смену может сменить несколько состояний, а за каждым состоянием закреплен один сотрудник.

* **Состояние – Код состояния:**

Связь: Много – Один

Состояний с одинаковыми кодами состояний много, а вот у состояния может быть только один код.

* **Сотрудник на смене – Сотрудник:**

Связь: Много – Один

У сотрудника может быть много смен. Но сотрудник на смене, это только один сотрудник.

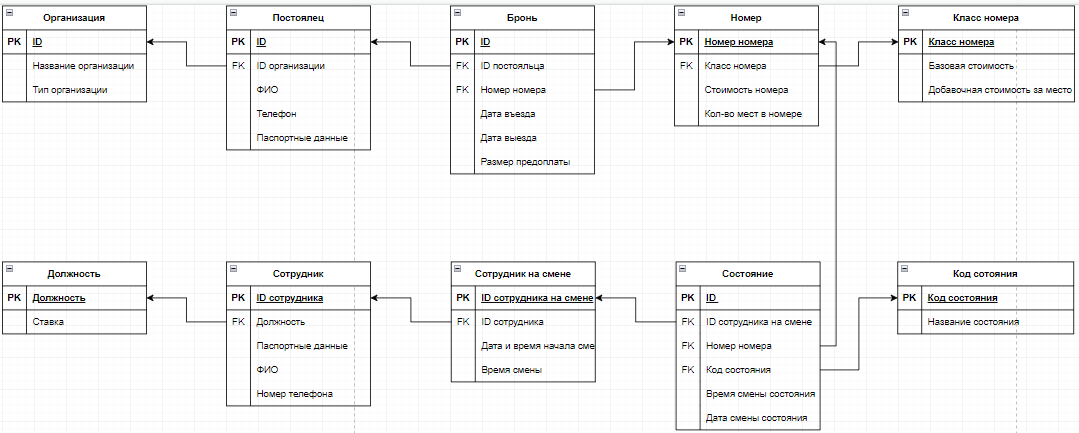
* **Сотрудник – Должность:**

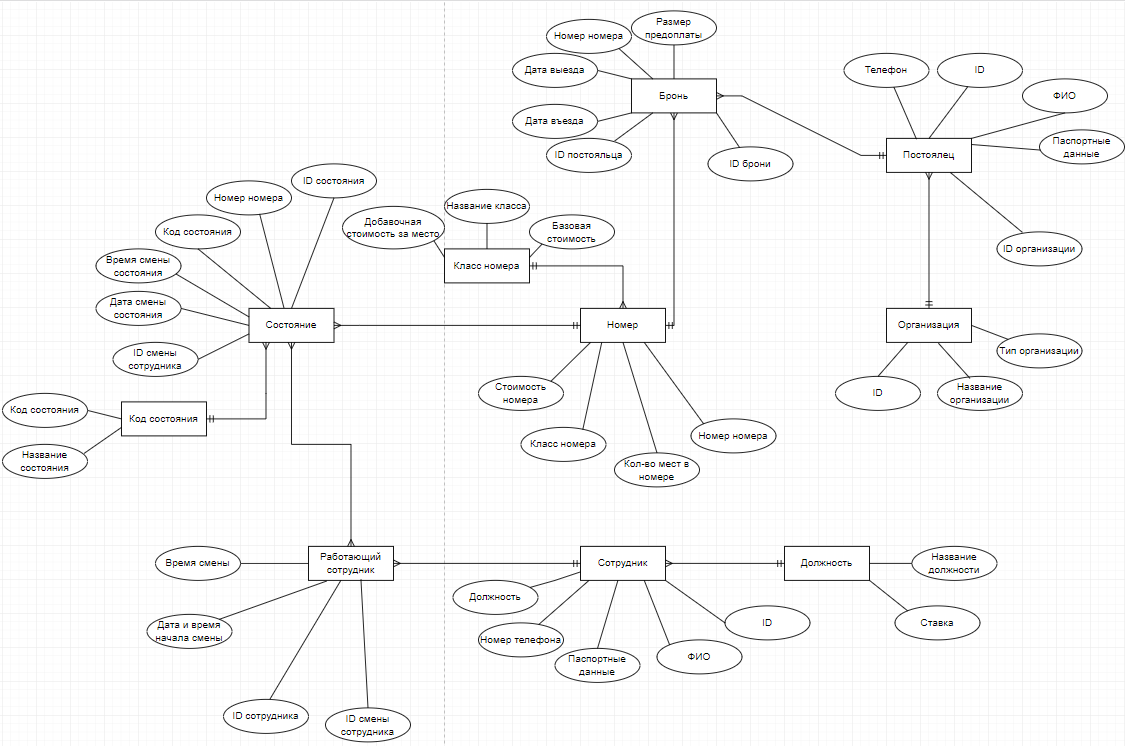
Связь: Много – Один

Сотрудников с одинаковыми должностями много, а вот у сотрудника может быть только одна должность.

РАЗРАБОТКА

Схема отношений



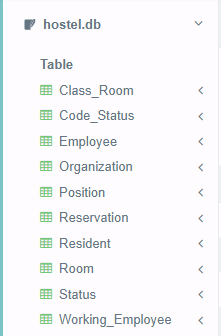


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблица | Свойство | Тип | Ограничения | Описание |
| Бронь | id | INT auto\_inc | Primary key | Ид. |
| id\_resident | INT | Not null | Ид. постояльца |
| check\_out\_date | DATE (TEXT) | Not null | Дата выселения |
| check\_in\_date | DATE (TEXT) | Not null | Дата заселения |
| room | INT | Not null | Номер комнаты |
| prepay | INT | Not null | Сумма предоплаты |
| Номер | number | INT auto\_inc | Primary key | Номер номера |
| class | TEXT | Not null | Класс номера |
| cost | INT | Default null | Стоимость номера |
| number\_of\_bed | INT | Not null | Количество спальных мест |
| Класс номера | class | TEXT | Primary key | Класс номера |
| base\_cost | INT | Not null | Базовая стоимость номера |
| added\_value | INT | Noy null | Добавочная стоимость за место |
| Состояние | status\_code | INT | Not null | Код состояния |
| time\_change | TIME (TEXT) | Not null | Время смены состояния |
| date\_change | DATE (TEXT) | Not null | Дата смены состояния |
| id\_working\_employee | INT | Not null | Ид. Смены работника сменившего состояние. |
| room | INT | Not null | Номер комнаты |
| id | INT auto\_inc | Primary key | Ид. |
| Код состояния | id\_code | INT | Primary key | Ид. кода состояния |
| name\_code | TEXT | Not null | Название состояния |
| Постоялец | name | TEXT | Not null | ФИО |
| id\_organization | INT | Default null | Ид. Организации |
| phone\_number | INT | Default null | Номер телефона |
| passport\_details | INT | Not null | Паспортные данные (серия + номер) |
| id | INT auto\_inc | Primary key | Ид. |
| Организация | name | TEXT | Not null | Название организации |
| type | TEXT | Not null | Тип организации |
| id | INT auto\_inc | Primary key | Ид. |
| Сотрудник | name | TEXT | Not null | ФИО |
| position | TEXT | Not null | Должность |
| phone\_number | INT | Default null | Номер телефона |
| passport\_details | INT | Not null | Паспортные данные (серия + номер) |
| id | INT auto\_inc | Primary key | Ид. |
| Сотрудник на смене | id\_employee | INT | Not null | Ид. сотрудника |
| shift\_start\_date\_time | DATETIME (TEXT) | Not null | Дата и время начала смены |
| shift\_time | TIME (TEXT) | Default null | Время смены |
| id | INT | Primary key | Ид. сотрудника на смене |
| Должность | position | TEXT | Primary key | Должность |
| wage-rate | INT | Not null | Ставка |

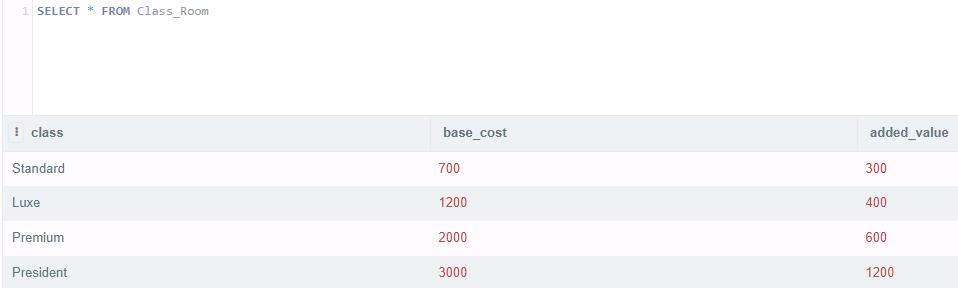
Примечание: Ид. = Идентификатор

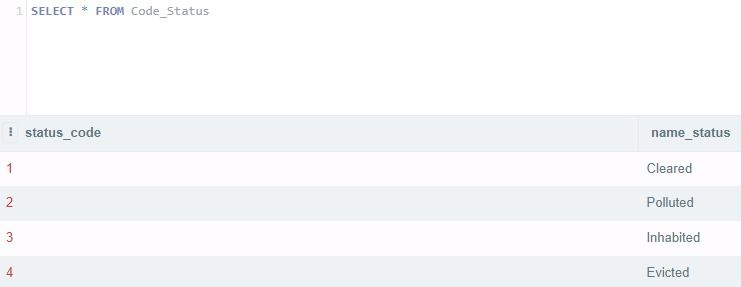
Для проверки запросов и просмотра таблиц использовался сайт <https://sqliteonline.com/>

Созданные таблицы:



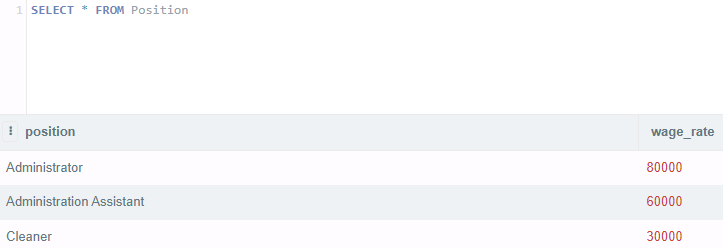
Наполненность:

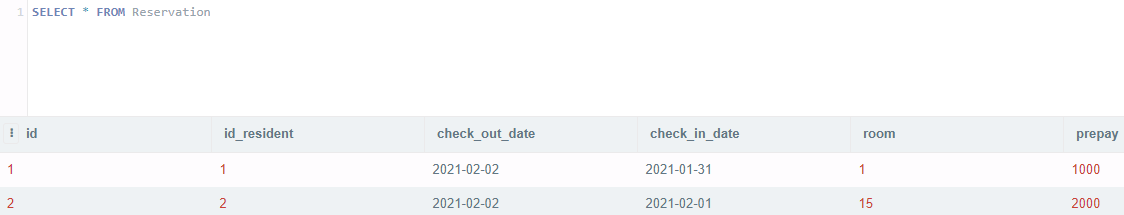




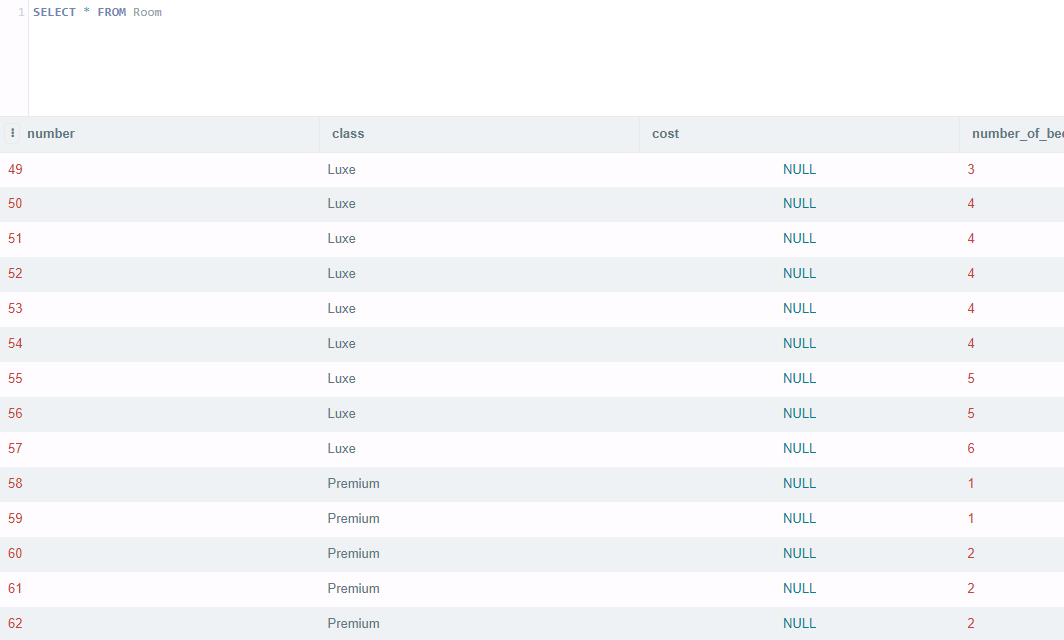


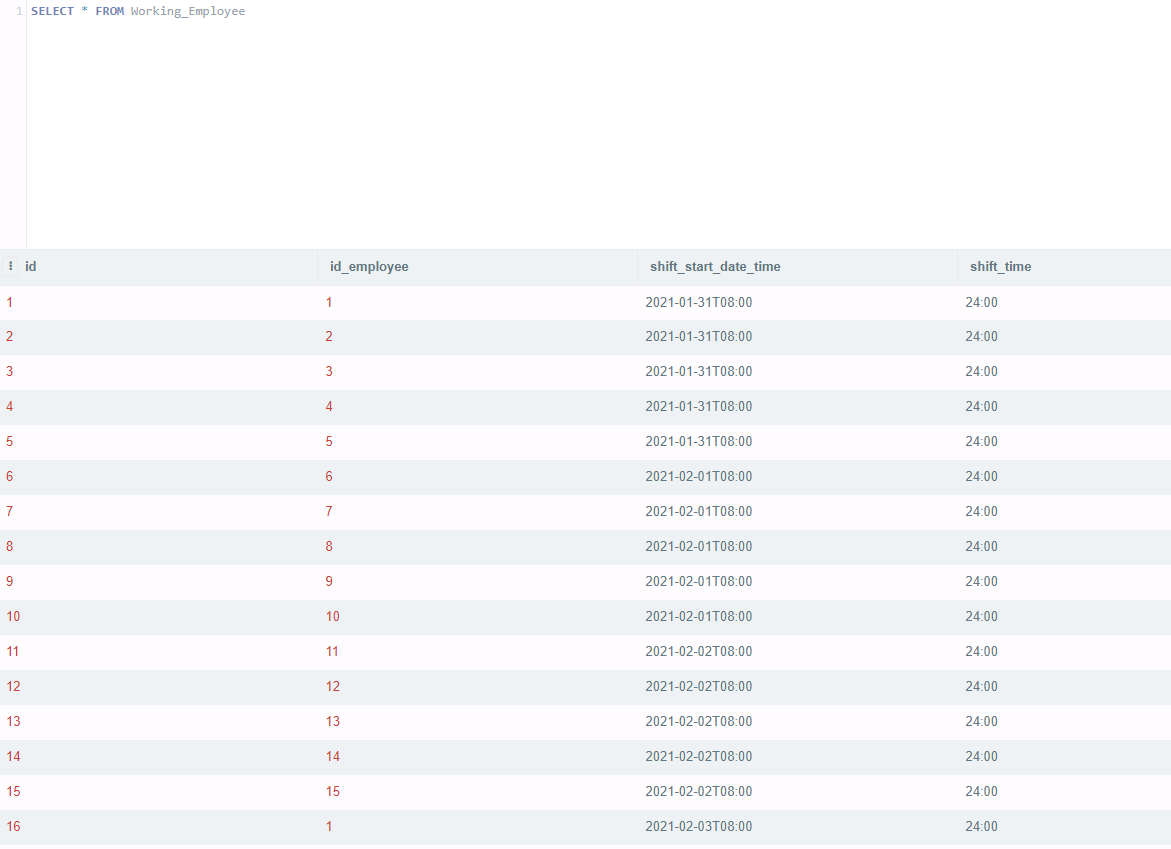


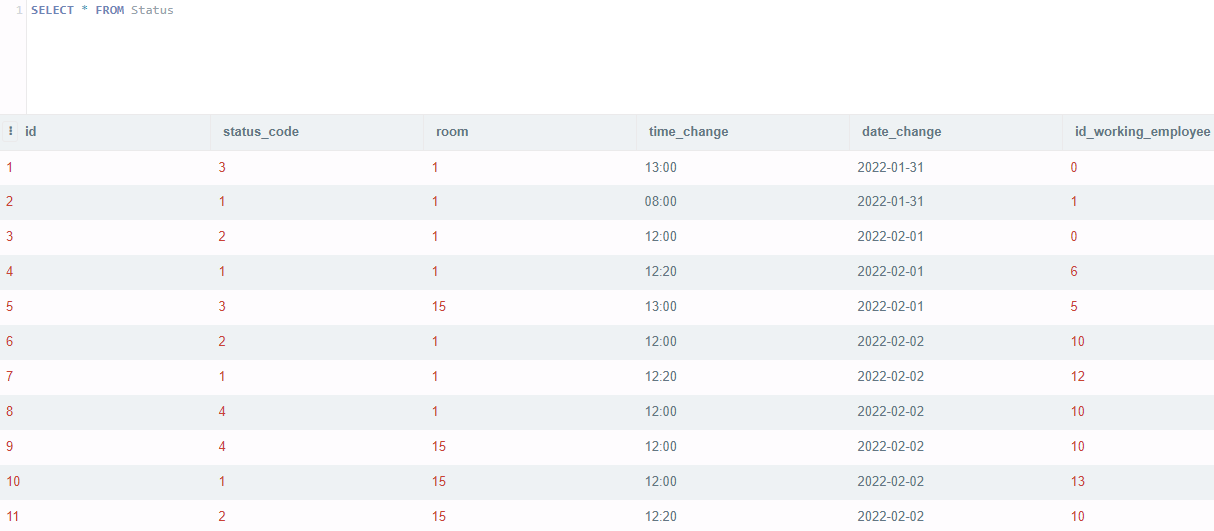












**Создание и заполнение БД на sqlite3 и python**

import sqlite3  
  
connection = sqlite3.connect('hostel.db')  
  
cursor = connection.cursor()  
  
  
# Создание таблиц.  
def create\_tables():  
 # Создаем таблицу комнат.  
 cursor.execute('''CREATE TABLE IF NOT EXISTS Room  
 (number INTEGER NOT NULL PRIMARY KEY,   
 class TEXT NOT NULL,  
 cost INTEGER DEFAULT NULL,  
 number\_of\_bed INT NOT NULL,  
 FOREIGN KEY (class) REFERENCES Class\_Room (class)  
 )''')  
  
 # Создаем таблицу постояльцев.  
 cursor.execute('''CREATE TABLE IF NOT EXISTS Resident  
 (id INTEGER NOT NULL PRIMARY KEY,   
 name TEXT NOT NULL,  
 id\_organization INTEGER DEFAULT NULL,  
 phone\_number INTEGER DEFAULT NULL,  
 passport\_details INTEGER NOT NULL,  
 FOREIGN KEY (id\_organization) REFERENCES Organization (id)  
 )''')  
  
 # Создаем таблицу организаций.  
 cursor.execute('''CREATE TABLE IF NOT EXISTS Organization  
 (id INTEGER NOT NULL PRIMARY KEY,   
 name TEXT NOT NULL,   
 type TEXT NOT NULL  
 )''')  
  
 # Создаем таблицу состояний.  
 cursor.execute('''CREATE TABLE IF NOT EXISTS Status  
 (id INTEGER NOT NULL PRIMARY KEY,   
 status\_code INTEGER NOT NULL,   
 room INTEGER NOT NULL,   
 time\_change TEXT NOT NULL,  
 date\_change TEXT NOT NULL,  
 id\_working\_employee INTEGER NOT NULL,  
 FOREIGN KEY (id\_working\_employee) REFERENCES Working\_Employee (id)  
 )''')  
  
 # Создаем таблицу броней.  
 cursor.execute('''CREATE TABLE IF NOT EXISTS Reservation  
 (id INTEGER NOT NULL PRIMARY KEY,   
 id\_resident INTEGER NOT NULL,   
 check\_out\_date TEXT NOT NULL,  
 check\_in\_date TEXT NOT NULL,  
 room INTEGER NOT NULL,  
 prepay INTEGER NOT NULL,  
 FOREIGN KEY (id\_resident) REFERENCES Resident (id),  
 FOREIGN KEY (room) REFERENCES Room (number)  
 )''')  
  
 # Создаем таблицу сотрудников.  
 cursor.execute('''CREATE TABLE IF NOT EXISTS Employee  
 (id INTEGER NOT NULL PRIMARY KEY,   
 name TEXT NOT NULL,   
 position TEXT NOT NULL,  
 phone\_number INTEGER DEFAULT NULL,  
 passport\_details INTEGER NOT NULL,  
 FOREIGN KEY (position) REFERENCES Employee (id)  
 )''')  
  
 # Создаем таблицу работающих сотрудников.  
 cursor.execute('''CREATE TABLE IF NOT EXISTS Working\_Employee  
 (id INTEGER NOT NULL PRIMARY KEY,   
 id\_employee INTEGER NOT NULL,   
 shift\_start\_date\_time TEXT NOT NULL,   
 shift\_time TEXT DEFAULT NULL,  
 FOREIGN KEY (id\_employee) REFERENCES Employee (id)  
 )''')  
  
 # Создаем таблицу Должностей  
 cursor.execute('''CREATE TABLE IF NOT EXISTS Position  
 (position TEXT NOT NULL PRIMARY KEY,   
 wage\_rate INT NOT NULL  
 )''')  
  
 # Создаем таблицу Классов номеров.  
 cursor.execute('''CREATE TABLE IF NOT EXISTS Class\_Room  
 (class TEXT NOT NULL PRIMARY KEY,   
 base\_cost INT NOT NULL,  
 added\_value INT NOT NULL  
 )''')  
  
 # Создаем таблицу Кодов состояний.  
 cursor.execute('''CREATE TABLE IF NOT EXISTS Code\_Status  
 (status\_code INTEGER NOT NULL PRIMARY KEY,   
 name\_status TEXT NOT NULL  
 )''')  
  
  
# Заполнение таблиц.  
def fill\_tables():  
 # Заполнение данными таблицу комнат  
 rooms = open('data/room.txt')  
 for line in rooms:  
 separate\_line = line.split('/')  
 for i in range(int(separate\_line[0])):  
 data = separate\_line[1] + ',' + separate\_line[2]  
 cursor.execute(f'''INSERT INTO Room  
 (class, number\_of\_bed)  
 VALUES({data})  
 ''')  
  
 # Заполнение данными таблицу классов номеров  
 class\_room = open('data/class\_room.txt')  
 for line in class\_room:  
 separate\_line = line.split('/')  
 data = separate\_line[0] + ',' + separate\_line[1] + ',' + separate\_line[2]  
 cursor.execute(f'''INSERT INTO Class\_Room  
 (class, base\_cost, added\_value)  
 VALUES({data})  
 ''')  
  
 # Заполнение данными таблицу работников  
 employee = open('data/employee.txt')  
 for line in employee:  
 print(line)  
 separate\_line = line.split('/')  
 data = separate\_line[0] + ',' + separate\_line[1] + ',' + separate\_line[2] + ',' + separate\_line[3]  
 cursor.execute(f'''INSERT INTO Employee  
 (name, position, phone\_number, passport\_details)  
 VALUES({data})  
 ''')  
  
 # Заполнение данными таблицу должностей  
 position = open('data/position.txt')  
 for line in position:  
 separate\_line = line.split('/')  
 data = separate\_line[0] + ',' + separate\_line[1]  
 cursor.execute(f'''INSERT INTO Position  
 (position, wage\_rate)  
 VALUES({data})  
 ''')  
  
 # Заполнение данными таблицу работников на смене  
 working\_employee = open('data/working\_employee.txt')  
 for line in working\_employee:  
 separate\_line = line.split('/')  
 data = separate\_line[0] + ',' + separate\_line[1] + ',' + separate\_line[2]  
 cursor.execute(f'''INSERT INTO Working\_Employee  
 (id\_employee, shift\_start\_date\_time, shift\_time)  
 VALUES({data})  
 ''')  
  
 # Заполнение данными таблицу постояльцев  
 resident = open('data/resident.txt')  
 for line in resident:  
 separate\_line = line.split('/')  
 data = separate\_line[0] + ',' + separate\_line[1] + ',' + separate\_line[2] + ',' + separate\_line[3]  
 cursor.execute(f'''INSERT INTO Resident  
 (name, id\_organization, phone\_number, passport\_details)  
 VALUES({data})  
 ''')  
  
 # Заполнение данными таблицу организаций  
 organization = open('data/organization.txt')  
 for line in organization:  
 separate\_line = line.split('/')  
 data = separate\_line[0] + ',' + separate\_line[1]  
 cursor.execute(f'''INSERT INTO Organization  
 (name, type)  
 VALUES({data})  
 ''')  
  
 # Заполнение данными таблицу кодов состояний  
 status\_code = open('data/status\_code.txt')  
 for line in status\_code:  
 print(line)  
 cursor.execute(f'''INSERT INTO Code\_Status  
 (name\_status)  
 VALUES({line})  
 ''')  
  
 # Заполнение данными таблицу состояний  
 status = open('data/status.txt')  
 for line in status:  
 separate\_line = line.split('/')  
 data = separate\_line[0] + ',' + separate\_line[1] + ',' + separate\_line[2] + ',' + separate\_line[3] + \  
 ',' + separate\_line[4]  
 cursor.execute(f'''INSERT INTO Status  
 (status\_code, room, time\_change, date\_change, id\_working\_employee)  
 VALUES({data})  
 ''')  
  
 # Заполнение данными таблицу броней  
 reservation = open('data/reservation.txt')  
 for line in reservation:  
 separate\_line = line.split('/')  
 data = separate\_line[0] + ',' + separate\_line[1] + ',' + separate\_line[2] + ',' + separate\_line[3] +\  
 ',' + separate\_line[4]  
 cursor.execute(f'''INSERT INTO Reservation  
 (id\_resident, check\_in\_date, check\_out\_date, room, prepay)  
 VALUES({data})  
 ''')  
  
  
create\_tables()  
fill\_tables()  
  
connection.commit()  
connection.close()

**Запросы**

**Получение стоимости номера по его классу и количеству спальных мест:**

UPDATE Room

SET cost =

(SELECT base\_cost FROM Class\_Room WHERE Class\_Room.class == Room.class) +

(SELECT added\_value FROM Class\_Room WHERE Class\_Room.class == Room.class) \* Room.number\_of\_bed

**Получение списка свободных номеров (по количеству мест и классу):**

SELECT \* FROM Room

WHERE (

NOT Room.number IN (

SELECT room FROM Reservation

WHERE(

check\_out\_date >= (SELECT(DATE('now')))

AND check\_in\_date < (SELECT(DATE('now')))

And (SELECT(TIME('12:00'))) < (SELECT(TIME('13:00')))

) OR (

check\_out\_date > (SELECT(DATE('now')))

AND check\_in\_date <= (SELECT(DATE('now')))

And (SELECT(TIME('12:00'))) >= (SELECT(TIME('13:00')))

))

) AND (Room.number\_of\_bed == 1) AND (Room.class == 'Standard')

**Получение списка номеров, освобождающихся сегодня и завтра:**

SELECT \* FROM Room WHERE Room.number IN

(SELECT room FROM Reservation

WHERE check\_out\_date == (SELECT(DATE('now'))) OR check\_out\_date == (SELECT(DATE('now','+1 day'))))

**Выдачу информации по конкретному номеру:**

SELECT \* FROM Room

Where number == 70

**Автоматизацию выдачи счета на оплату номера:**

SELECT

(SELECT julianday(check\_out\_date) - julianday(check\_in\_date)

FROM Reservation

WHERE check\_out\_date == (SELECT(DATE(‘now’))) AND id\_resident IN

(SELECT id FROM Resident WHERE

Resident.name == "Monica Geller")) \*

(SELECT cost FROM Room

WHERE number in (SELECT room FROM Reservation

Where Reservation.check\_out\_date == (SELECT(DATE(‘now))) AND Reservation.id\_resident IN

(SELECT id FROM Resident WHERE

Resident.name == "Monica Geller"))) as total

**Получение списка заселенных номеров:**

SELECT \* FROM Room

WHERE Room.number IN (

SELECT room FROM Reservation

WHERE(

check\_out\_date >= (SELECT(DATE('now')))

AND check\_in\_date < (SELECT(DATE('now')))

And (SELECT(TIME('12:00'))) < (SELECT(TIME('13:00')))

) OR (

check\_out\_date > (SELECT(DATE('now')))

AND check\_in\_date <= (SELECT(DATE('now')))

And (SELECT(TIME('12:00'))) >= (SELECT(TIME('13:00')))

))

**Получение списка броней клиента:**

SELECT \* FROM Reservation

WHERE id\_resident IN

(SELECT id FROM Resident WHERE

Resident.name == "Monica Geller")

**Получение списка сотрудников, которые работали в нужный день и время:**

SELECT \* FROM Working\_Employee WHERE

strftime('%s', Working\_Employee.shift\_start\_date\_time) < strftime('%s', '2021-02-01 09:00:00')

AND

strftime('%s', Working\_Employee.shift\_time) - strftime('%s', '2000-01-01 00:00:00') > strftime('%s', '2021-02-01 09:00:00') - strftime('%s', Working\_Employee.shift\_start\_date\_time)