

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ

**Национальный исследовательский ядерный университет «МИФИ»**



**ЛАПЛАЗ**

**Кафедра кибернетики (№ 22)**

**Отчёт о работе по курсу**

**«Базы данных (теоретические основы баз данных)»**

**Вариант «РЖД»**

Выполнил	Азаров Иван, Ильин Владислав, Умаров Азиз
Группа	Б21-215
Вариант	РЖД
Преподаватель	Павленко А. Д.
Проверяющий	
Оценка	

**Москва 2023**

## Содержание

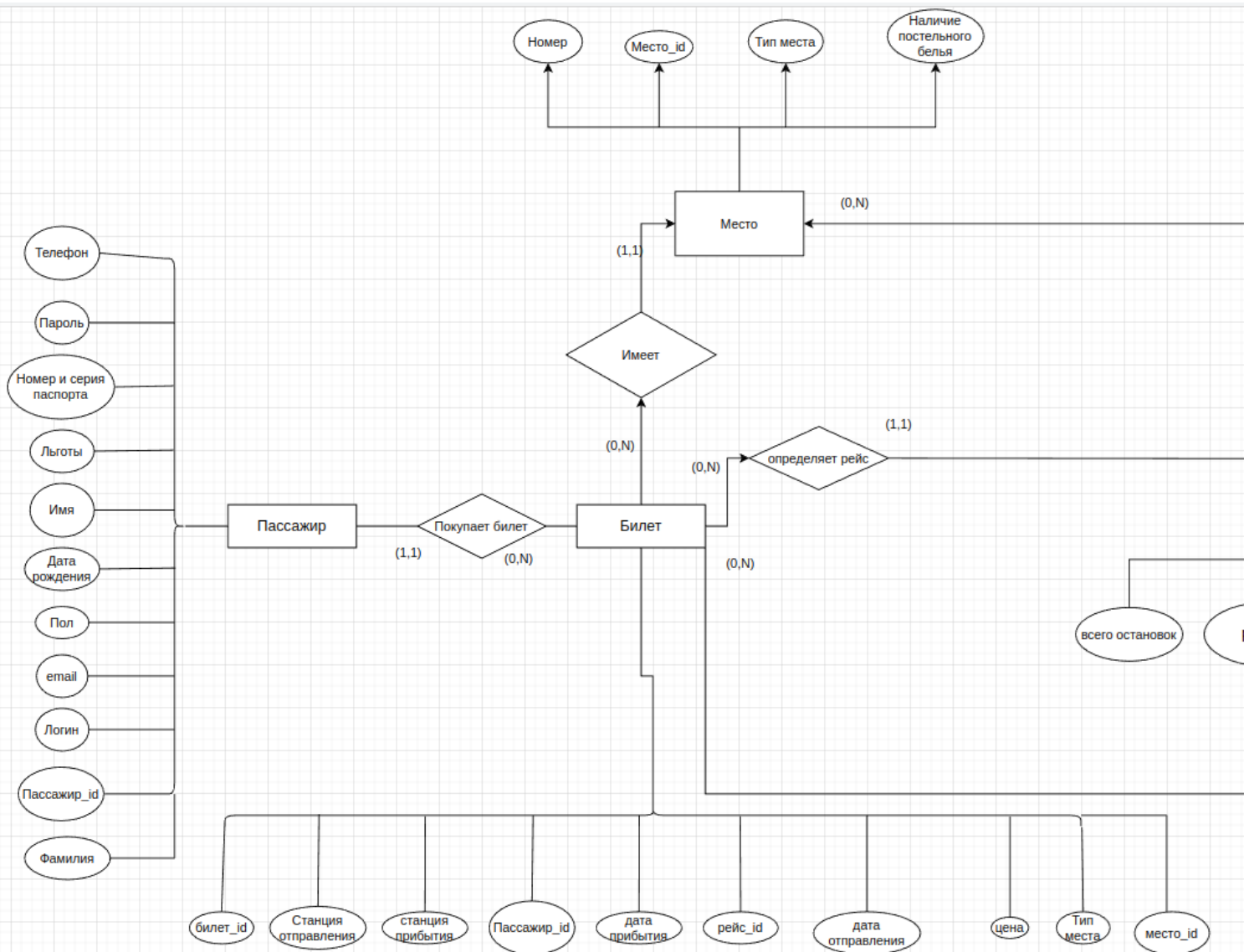
1. Формулировка задания .....	3
2. Концептуальная модель базы данных .....	3
1.1. Конкретизация предметной области .....	4
1.2. Описание предметной области и принятых ограничений .....	4
1.3. Описание атрибутов.....	5
3. Логическое проектирование .....	7
4. Физическое проектирование.....	8
1.4. Создание таблиц     8	
1.5. Заполнение базы данных .....	10
1.5.1. Подготовка данных             11	
1.5.2. Программа заполнения базы данных .....	13
1.5.3. Результаты заполнения         15	
5. Выполнение запросов .....	19

## 1. Формулировка задания

Спроектировать базу данных для сети железных дорог «РЖД», расположенных в разных городах России и осуществляющих перевозку пассажиров на поездах. База данных должна содержать информацию о рейсах, отражать пассажиров, непосредственно сеть станций и информацию о поездах.

## 2. Концептуальная модель базы данных

После проведения анализа предметной области была спроектирована следующая концептуальная модель:



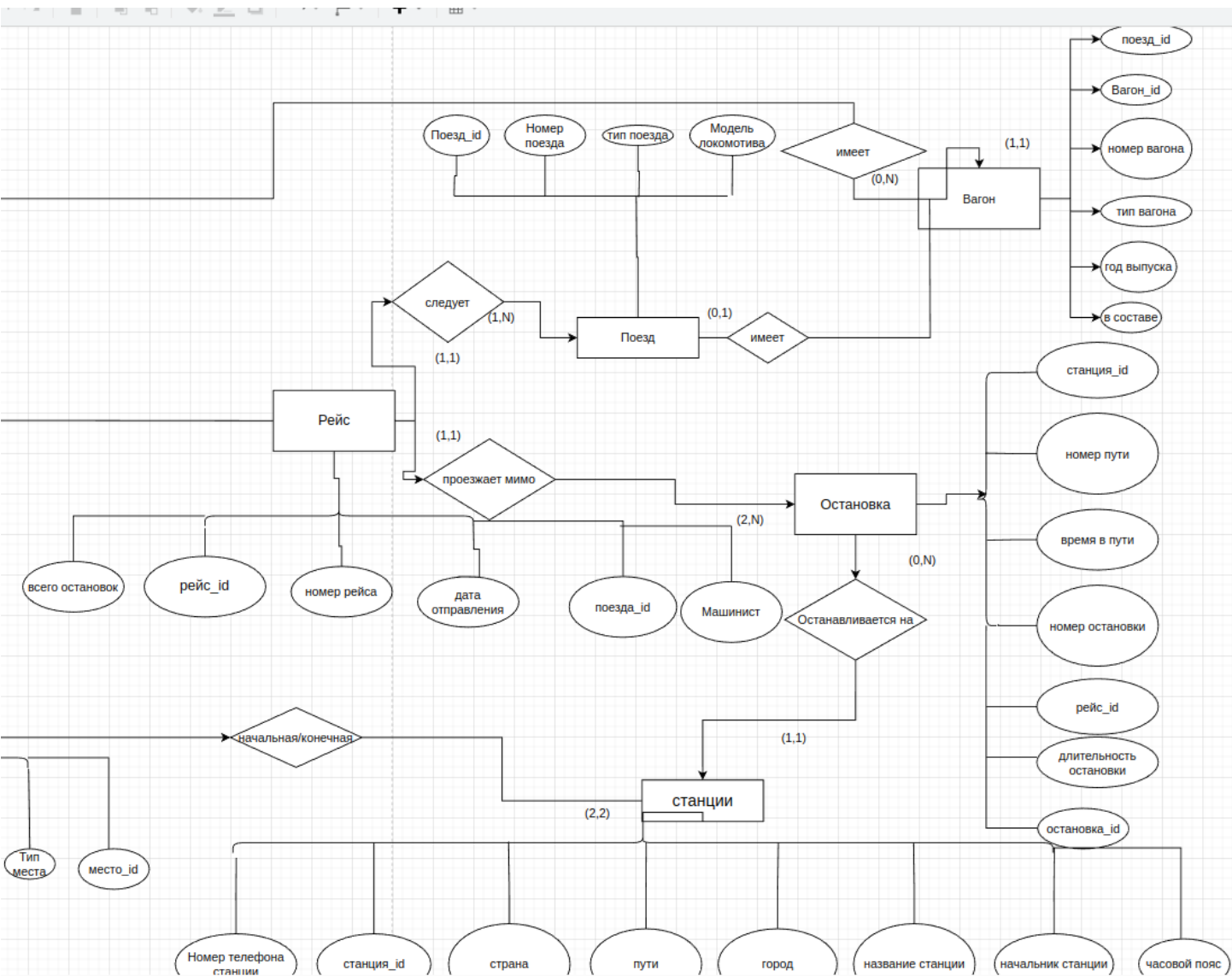


Рисунок 1 – Концептуальная модель базы данных

### **1.1. Конкретизация предметной области**

Необходимо создать систему, отражающую информацию о рейсах поездов по нескольким странам. На рейсы надо рассадить различных пассажиров.

База данных должна отражать по каким рейсам следовал, следует и будет следовать поезд. Сам рейс должен содержать информацию о пассажирах, купивших в него билет.

### **1.2. Описание предметной области**

Любой пользователь может узнать информацию об интересующих его рейсах — номер рейса, путь следования, тип поезда.

Каждый человек может зарегистрироваться в приложении или на сайте компании-РЖД, указав свой номер телефона и имя, став клиентом сети.

Пассажир может приобретать билеты из ассортимента, информация о котором хранится в базе данных и отражается в приложении и на сайте. Также, пассажиру доступна информация о всех билетах, купленных им с момента регистрации. Даже если рейс, содержащийся в билете, больше не активен, т.е. его невозможно приобрести в сети РЖД, клиент увидит его. То же касается и станций отправления и прибытия, к которым привязывается заказ при оформлении.

Сотрудники могут проводить анализ и учёт всех рейсов, совершенных с момента начала функционирования системы. Кроме того, В базе данных отражаются текущие и будущие рейсы, купленных на него билетах, а также информация о поезде.

Определенные лица компании: HR-менеджеры, администраторы сети, директора компании, - также могут вносить изменения в назначениях сотрудников, информации о времени работы пиццерий и вносить данные новых работников и зданий.

В итоге, система реализует следующий функционал:

- учёт рейсов;
- регистрация всех билетов;
- учёт персональной информации пассажиров;

- учёт информации непосредственных поездов;
- Предоставление справочной информации о каждой станции. Таким образом, были выделены следующие сущности:
- Пассажир
- Билет
- Рейс
- Остановка
- Станция
- Поезд
- Вагон
- Место

### 1.3. Описание атрибутов

В процессе анализа были выделены следующие атрибуты, название и описание которых приведены в таблице ниже.

Имя атрибута	Расшифровка
Passenger_id, Ticket_id, Van_id, Place_id, flight_id, Station_id, arrive_id, departure_id	Уникальный идентификатор соответственно пассажира, билета, вагона, места, рейса, станции, станции отправления и прибытия
name	Имя пассажира
login	Логин пассажира
Phone number	Номер телефона пассажира, станции
email	Адрес электронной почты пользователя
birth_date	Дата рождения пользователя
password	Пароль пользователя
passport	Серия и номер паспорта
fees	Льготы
gender	Пол
place	Номер места
van	Номер вагона
price	Цена

place_type	Тип места
dep_date	Дата отправления
arrive_date	Дата прибытия
flnumber	Номер рейса
driver	Имя машиниста
total_stop	Всего остановок
stnumber	Номер остановки
stoptime	Время стоянки
waynumber	Номер пути
dep_date	Дата прибытия на остановку
arrive_date	Дата отправления с остановки
stname	Название станции
city	Город в котором расположена станция
way	Всего путей
Country	Страна в которой находится станция
director	Имя начальника станции
timezone	Часовой пояс станции
train_number	Номер поезда
train_type	Тип поезда
loc_model	Модель локомотива
van_type	Тип вагона
issue_year	Дата выпуска вагона
vannumber	Номер вагона
intrain	Вагон в составе поезда или нет
place_type	Тип места
underwear	Наличие постельного белья
place_number	Номер места

## **2. Логическое проектирование**

Следующим шагом на основе КМПО была разработана логическая модель базы данных, представленная ниже:



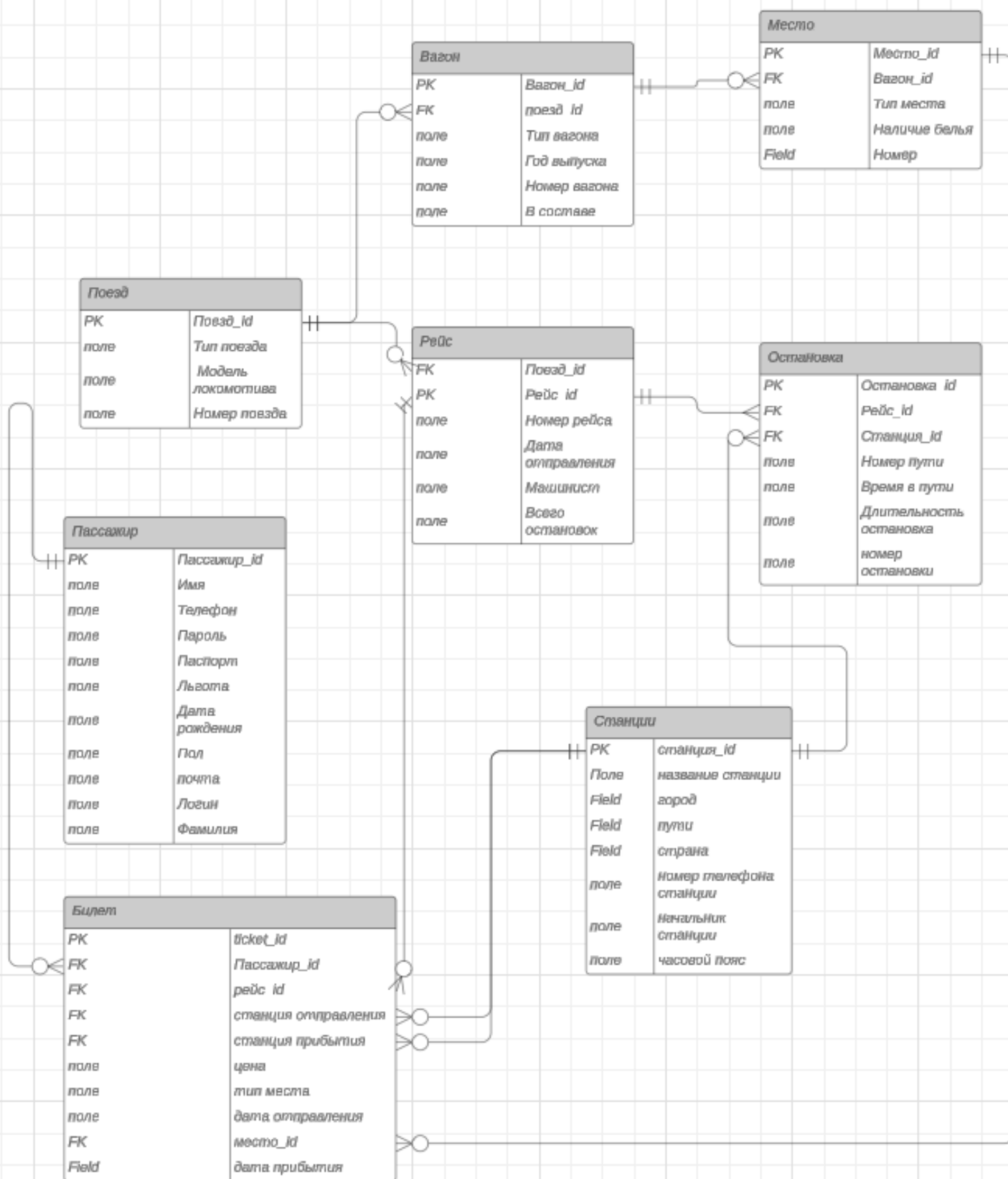


Рисунок 2 – Логическая модель базы данных

### **3. Физическое проектирование**

В качестве СУБД для реализации разработанной базы данных была выбрана PostgreSQL. В связи с проведённым анализом предметной области и была проработана следующая физическая схема базы данных. Она представлена на следующем рисунке:

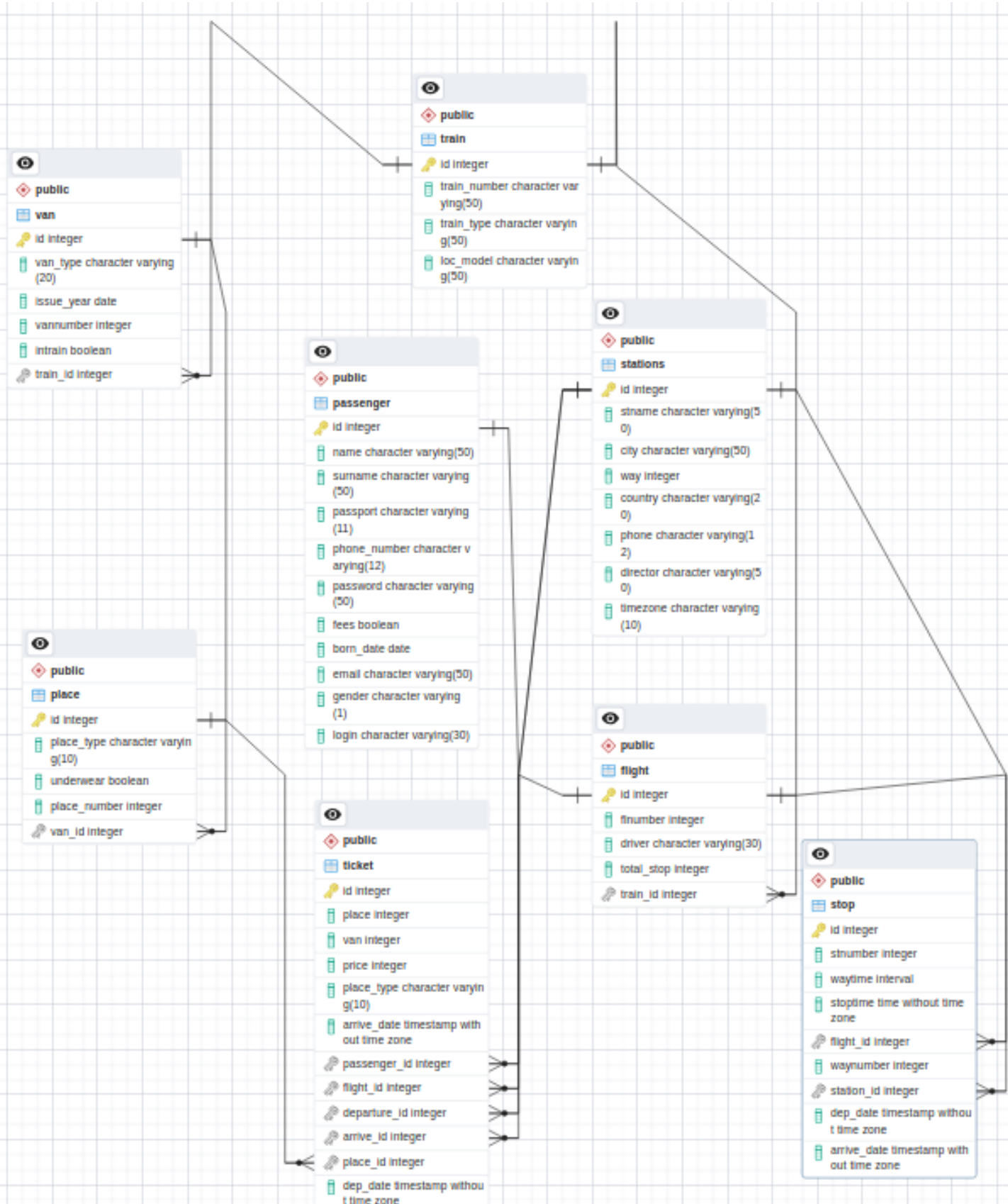


Рисунок 3 – Графическое представление базы данных

### 3.1. Создание таблиц

Ниже приведены код на языке Python со скриптами SQL для создания таблиц для таблиц, описанных выше.

Подключение к серверу:

```
1  import psycopg2
2  from psycopg2 import Error
3
4  try:
5      # Подключиться к существующей базе данных
6      connection = psycopg2.connect(user="postgres",
7                                     # пароль, который указали при установке PostgreSQL
7                                     password="1",
8                                     host="127.0.0.1",
9                                     port="5432",
10                                    database="rzd")
11
12     # Создайте курсор для выполнения операций с базой данных
13     cursor = connection.cursor()
14     # SQL-запрос для создания новой таблицы
15     create_table_query = ''
16
```

Создание таблицы Passenger:

```
17  drop table if exists Passenger cascade;
18  CREATE TABLE Passenger
19  (
20      id SERIAL PRIMARY KEY,
21      name varchar(50) NOT NULL,
22      surname varchar(50) NOT NULL,
23      passport varchar(11) NOT NULL,
24      phone_number varchar(12),
25      password varchar(50) NOT NULL,
26      fees boolean,
27      born_date date NOT NULL,
28      email varchar(50),
29      gender varchar(1) NOT NULL,
30      login varchar(30) NOT NULL
31  );
```

Создание таблицы Train:

```
32 drop table if exists Train cascade;  
33 create table Train  
34 (  
35 id Serial primary key,  
36 train_number varchar(50) NOT NULL,  
37 train_type varchar(50) NOT NULL,  
38 loc_model varchar(50) NOT NULL  
39 );  
40
```

Создание таблицы Flight:

```
41 drop table if exists Flight cascade;  
42 create table Flight  
43 (  
44 id Serial primary key,  
45 flnumber int NOT NULL,  
46 driver varchar(30) NOT NULL,  
47 total_stop int NOT NULL,  
48 train_id int NOT NULL,  
49 foreign key (train_id) references Train (id)  
50 );  
51
```

Создание таблицы Stations:

```
52 drop table if exists Stations cascade;  
53 create table Stations  
54 (  
55 id Serial primary key,  
56 stname varchar(50) NOT NULL,  
57 city varchar(50) NOT NULL,  
58 way int NOT NULL,  
59 country varchar(20) NOT NULL,  
60 phone varchar(12) NOT NULL,  
61 director varchar(50) NOT NULL,  
62 timezone varchar(10) NOT NULL  
63 );  
64
```

Создание таблицы Van:

```
65 drop table if exists Van cascade;  
66 create table Van  
67 (  
68 id Serial primary key,  
69 van_type varchar(20) NOT NULL,  
70 issue_year date NOT NULL,  
71 vannumber int NOT NULL,  
72 intrain boolean NOT NULL,  
73 train_id int NOT NULL,  
74 foreign key (train_id) references Train (id)  
75 );  
76
```

Создание таблицы Place:



```
77      drop table if exists Place cascade;
78      create table Place
79      (
80      id serial primary key,
81      place_type varchar(10) NOT NULL,
82      underwear boolean NOT NULL,
83      place_number int NOT NULL,
84      van_id int NOT NULL,
85      foreign key (van_id) references Van (id)
86      );
87
```

Создание таблицы Ticket:

```
88      drop table if exists Ticket cascade;
89      create table Ticket
90      (
91      id SERIAL PRIMARY KEY,
92      place INT NOT NULL,
93      van INT NOT NULL,
94      price int NOT NULL,
95      place_type varchar(10) NOT NULL,
96      arrive_date timestamp NOT NULL,
97      passenger_id int NOT NULL,
98      flight_id int NOT NULL,
99      departure_id int NOT NULL,
100     arrive_id int NOT NULL,
101     place_id int NOT NULL,
102     dep_date timestamp NOT NULL,
103     foreign key (passenger_id) references Passenger (id),
104     foreign key (flight_id) references Flight (id),
105     foreign key (departure_id) references Stations (id),
106     foreign key (arrive_id) references Stations (id),
107     foreign key (place_id) references Place (id)
108     );
109
```

Создание таблицы Stop:

```

110 drop table if exists Stop cascade;
111 create table Stop
112 (
113     id Serial primary key,
114     stnumber int NOT NULL,
115     waytime interval NOT NULL,
116     stoptime time NOT NULL,
117     flight_id int NOT NULL,
118     waynumber int NOT NULL,
119     station_id int NOT NULL,
120     dep_date timestamp NOT NULL,
121     arrive_date timestamp NOT NULL,
122     foreign key (flight_id) references Flight (id),
123     foreign key (station_id) references Stations (id)
124 )
125 ''

```

Обработка исключений и окончание работы:

```

126     # Выполнение команды: это создает новую таблицу
127     cursor.execute(create_table_query)
128     connection.commit()
129     print("Таблица успешно создана в PostgreSQL")
130
131 except (Exception, Error) as error:
132     print("Ошибка при работе с PostgreSQL", error)
133 finally:
134     if connection:
135         cursor.close()
136         connection.close()
137     print("Соединение с PostgreSQL закрыто")
138

```

Как видно, происходит обыкновенный INSERT запрос, который содержит данные, выбранные случайным образом. Также осуществляется SELECT запрос к БД, чтобы на выходе получить относительно достоверные данные. В данном случае, заказы клиента осуществляются в городах, который он указывал в своих адресах.

### 3.2. Заполнение базы данных

Заполнение базы данных производилось при помощи ЯП Python.



### 3.2.1. Подготовка данных

В сети Интернет были найдены списки имён, фамилий, названий городов и станций, а также часовые пояса городов, различные домены электронных адресов, и были обёрнуты в объект csv. Краткое описание объекта представлено ниже:

```
city.csv
1  Индекс,Тип региона,Регион,Тип района,Район,Тип города,Город,Тип н/п,И
2  385200,Респ,Адыгея,,,г,Адыгейск,,,0100000200000,ccdfd496-8108-4655-a
3  385000,Респ,Адыгея,,,г,Майкоп,,,0100000100000,8cfbe842-e803-49ca-934
4  649000,Респ,Алтай,,,г,Горно-Алтайск,,,0400000100000,0839d751-b940-4d
5  658125,край,Алтайский,,,г,Алейск,,,2200000200000,ae716080-f27b-40b6-a
6  656000,край,Алтайский,,,г,Барнаул,,,2200000100000,d13945a8-7017-46ab
7  659900,край,Алтайский,,,г,Белокуриха,,,2200000300000,e4edca96-9b86-4
8  659300,край,Алтайский,,,г,Бийск,,,2200000400000,52f876f6-cb1d-4f23-a
9  658420,край,Алтайский,р-н,Локтевский,г,Горняк,,,2202700100000,094b36
10 659100,край,Алтайский,,,г,Заринск,,,22000001100000,142e04ef-dec1-44fa
11 658480,край,Алтайский,р-н,Змеиногорский,г,Змеиногорск,,,2201500100000
12 658700,край,Алтайский,р-н,Каменский,г,Камень-на-Оби,,,2201800100000,
13 658041,край,Алтайский,,,г,Новоалтайск,,,2200000800000,aa288d9f-4b2a-
```

```
names.csv
1  name
2  Аарон
3  Аба
4  Аббас
5  Абд аль-Узза
6  Абдуллах
7  Абид
8  Аботур
9  Аввакум
10 Август
11 Авдей
12 Авель
13 Аверкий
```

stations.csv

1	stations
2	ПЕТРОЗАВОДСК
3	ТОМИЦЫ
4	ШУЙСКАЯ
5	СУНА
6	ЗАДЕЛЬЕ
7	КОНДОПОГА
8	КЕДРОЗЕРО
9	НОВЫЙ ПОСЕЛОК
0	НИГОЗЕРО
1	КЯППЕСЕЛЬГА
2	ПЕРГУБА
3	МЕДВЕЖЬЯ ГОРА

surnames.csv

1	surname
2	Абабков
3	Абаимов
4	Абакишин
5	Абакулин
6	Абакулов
7	Абакумкин
8	Абакумов
9	Абакушин
10	Абакшин
11	Абалакин
12	Абалаков
13	Абалдуев

## Программа заполнения базы данных

Выполнение запросов к БД осуществляется посредством транзакций:

```
1 import psycopg2
2 from psycopg2 import Error
3
4 try:
5     # Подключиться к существующей базе данных
6     connection = psycopg2.connect(user="postgres",
7                                     # пароль, который указали при установке PostgreSQL
8                                     password="1",
9                                     host="127.0.0.1",
10                                    port="5432",
11                                    database="rzd")
12
13     # Создайте курсор для выполнения операций с базой данных
14     cursor = connection.cursor()
15     # SQL-запрос для создания новой таблицы
16     create_table_query = '''
17     drop table if exists Passenger cascade;
18     CREATE TABLE Passenger
19     (id SERIAL PRIMARY KEY,
20      name varchar(50) NOT NULL,
21      surname varchar(50) NOT NULL,
22      passport varchar(11) NOT NULL,
23      phone_number varchar(12),
24      password varchar(50) NOT NULL,
25      fees boolean,
26      born_date date NOT NULL,
27      email varchar(50),
28      gender varchar(1) NOT NULL,
29      login varchar(30) NOT NULL
30     );
31
32     drop table if exists Train cascade;
33     create table Train
34     (
35     id Serial primary key,
36     train_number varchar(50) NOT NULL,
37     train_type varchar(50) NOT NULL,
38     loc_model varchar(50) NOT NULL
39     );
40
41     drop table if exists Flight cascade;
42     create table Flight
43     (
44     id Serial primary key,
```

Как видно, происходит обыкновенный INSERT запрос, который содержит данные, выбранные случайным образом. Также осуществляется SELECT запрос к БД, чтобы на выходе получить относительно достоверные данные. В данном случае, заказы клиента осуществляются в городах, который он указывал в своих адресах.

### 3.2.2. Результаты заполнения

Далее представлены результаты работы программы на примере таблиц, соответствующих функциям, приведенным выше.

- Таблица Passenger

	id [PK] integer	name character varying (50)	surname character varying (50)	passport character varying (11)	phone_number character varying (12)	password character varying (50)	fees boolean
1	1	Дарья	Собакаев(а)	5350 189055	89896976740	4212402875050	true
2	2	Амелия	Агашкин(а)	9971 174322	89334781552	5070952030552	true
3	3	Рон	Оскрометов(а)	2450 249142	89767272907	9610225710746	true
4	4	Валерия	Талалыкин(а)	3768 245149	89991455634	5638187060773	true
5	5	Магна	Федюнов(а)	7751 758964	89147477597	7407746698723	false
6	6	Наина	Пакулов(а)	4715 957325	89910633633	330093138007	true
7	7	Федора	Анисимцев(а)	9202 852914	8937582248	3798080601311	true
8	8	Дарья	Собакаев(а)	5350 189055	89896976740	4212402875050	true

- Таблица Ticket

	<div><div>Id</div><div>[PK] integer</div></div>	<div><div>place</div><div>integer</div></div>	<div><div>van</div><div>integer</div></div>	<div><div>price</div><div>integer</div></div>	<div><div>place_type</div><div>character var</div></div>	<div><div>arrive_date</div><div>timestamp without time zone</div></div>	<div><div>passenger_id</div><div>integer</div></div>	<div><div>flight_id</div><div>integer</div></div>	<div><div>departure_id</div><div>integer</div></div>	<div><div>arrive_id</div><div>integer</div></div>
1	1	1	2	1622	сидячий	2023-07-04 02:19:00	163	276	78	14
2	2	3	5	2364	купе	2023-03-30 19:23:00	29	249	24	7
3	3	9	1	3753	купе	2023-11-15 17:23:00	130	255	13	97
4	4	4	2	2151	плацкарт	2023-03-24 22:13:00	280	233	93	31
5	5	9	1	1096	купе	2023-04-14 21:57:00	258	141	44	62
6	6	5	5	4365	сидячий	2023-08-20 08:36:00	285	30	90	86
7	7	9	3	1059	плацкарт	2023-06-14 18:19:00	217	104	46	28
8	8	2	4	1734	сидячий	2023-01-29 07:40:00	38	188	47	1

- Таблица flight

	id [PK] integer	finumber integer	driver character varying (30)	total_stop integer	train_id integer
1	1	937	Аарон	4	131
2	2	104	Аба	7	86
3	3	518	Аббас	3	62
4	4	638	Абд аль-Узза	4	154
5	5	750	Абдуллах	6	29
6	6	768	Абид	6	2
7	7	844	Аботур	3	160
8	8	622	Аввакум	6	154

✓ Successfully run. Total query runtime: 166 msec. 300 rows affected. ✕

- Таблица Stop

	id [PK] integer	stnumber integer	waytime interval	stoptime time without time zone	flight_id integer	waynumber integer	station_id integer	dep_date timestamp without time zone	arrive_date timestamp without time zone
1	1	1	00:00:00	00:00:00	1	5	89	2023-02-10 17:23:00	2023-02-10 17:23:00
2	2	2	13:49:00	00:02:00	1	1	21	2023-02-11 07:14:00	2023-02-11 07:12:00
3	3	3	1 day 07:...	00:05:00	1	1	22	2023-02-12 14:54:00	2023-02-12 14:49:00
4	4	4	14:15:00	00:00:00	1	2	14	2023-02-13 05:09:00	2023-02-13 05:09:00
5	5	1	00:00:00	00:00:00	2	4	63	2023-03-16 10:59:00	2023-03-16 10:59:00
6	6	2	15:51:00	00:02:00	2	5	56	2023-03-17 02:52:00	2023-03-17 02:50:00
7	7	3	20:51:00	00:02:00	2	2	27	2023-03-17 23:45:00	2023-03-17 23:43:00
8	8	4	13:48:00	00:07:00	2	3	13	2023-03-18 13:40:00	2023-03-18 13:33:00

- Таблица Station

	id [PK] integer	stname character varying (50)	city character varying (50)	way integer	country character varying (20)	phone character varying (12)	director character varying (50)
1	1	ПЕТРОЗАВОДСК	Белогорск	4	Малороссия	8958327425	Аарон
2	2	ТОМИЦЫ	Чапаевск	3	Россия	89365602876	Аба
3	3	ШУЙСКАЯ	Андреаполь	2	Белоруссия	89253419312	Аббас
4	4	СУНА	Биробиджан	2	Азербайджан	89452444576	Абд аль-Узаа
5	5	ЗАДЕЛЬЕ	Анива	4	Таджикистан	89969912713	Абдуллах
6	6	КОНДОПОГА	Сураж	2	Малороссия	89321042334	Абид
7	7	КЕДРОЗЕРО	Медвежьегорск	4	Грузия	89435803948	Аботур
8	8	НОВЫЙ ПОСЕЛОК	Западная Двина				

✓ Successfully run. Total query runtime: 140 msec. 100 rows affected. ✕

- Таблица Train

	id [PK] integer	train_number character varying (50)	train_type character varying (50)	loc_model character varying (50)
1	1	440jj	фирменный	пассажирский
2	2	366bi	высокоскоростной	пассажирский
3	3	984ez	скоростной	пассажирский
4	4	744iy	высокоскоростной	пассажирский
5	5	842zy	фирменный	пассажирский
6	6	370hf	скоростной	пассажирский
7	7	312nz	скоростной	пассажирский
8	8	482cj	фирменный	пассажирский

✓ Successfully run. Total query runtime: 122 msec. 200 rows affected. ✕

- Таблица Van

	id [PK] integer	van_type character varying (20)	issue_year date	vannumber integer	intrain boolean	train_id integer
1	1	сидячий	2005-01-12	1	true	1
2	2	купе	1987-08-18	2	true	1
3	3	плацкарт	1995-06-11	3	true	1
4	4	купе	1996-10-14	4	true	1
5	5	купе	1975-06-22	5	true	1
6	6	сидячий	2008-04-03	1	true	2
7	7	купе	2021-06-21	2	true	2
8	8	плацкарт	1970-01-09	3	true	2

✓ Successfully run. Total query runtime: 132 msec. 1111 rows affected. ✕

- Таблица Place

	id [PK] integer	place_type character varying (10)	underwear boolean	place_number integer	van_id integer
1	1	сидячий	false	1	1
2	2	сидячий	false	2	1
3	3	сидячий	false	3	1
4	4	сидячий	false	4	1
5	5	сидячий	false	5	1
6	6	сидячий	false	6	1
7	7	сидячий	false	7	1
8	8	сидячий	false		

✓ Successfully run. Total query runtime: 147 msec. 16648 rows affected. ✕

## 4. Выполнение запросов

В этом разделе приведены различные запросы к реализованной базе данных — их краткие описания, непосредственно запрос на SQL языке и результат выполнения.

- ⑩ Показать процент людей, который отправляются с первой станции рейса и прибывают на последнюю для каждого рейса

```
4 with f1 as (  
5   select * from  
6   (select T.flight_id as flight, S.station_id as first_st,  
7     T.departure_id, T.arrive_id  
8    from Ticket as T  
9   left join (select station_id, flight_id from Stop  
10  where stnumber = 1) as S  
11  on T.flight_id = S.flight_id) as T1  
12  order by T1.flight),  
13 f2 as (  
14   select f1.flight, f1.departure_id, f1.arrive_id,  
15     f1.first_st, S1.station_id as last_st  
16   from f1  
17   join (select station_id, flight_id from Stop  
18   where stnumber = (select total_stop from Flight  
19   where id = flight_id)) as S1  
20   on f1.flight = S1.flight_id),  
21 f3 as (  
22   select flight, count(*) from f1  
23   group by flight  
24   ),  
25 f4 as (  
26   select flight, count(*) from f2  
27   where departure_id = first_st and arrive_id = last_st  
28   group by flight)  
29   select f3.flight, coalesce(round(f4.count * 100 / f3.count, 2), 0)  
30   from f3  
31   left join f4  
32   on f3.flight = f4.flight  
33   order by f3.flight
```

	flight integer	coalesce numeric
1	1	0
2	2	0
3	3	12.00
4	4	15.00
5	5	20.00
6	6	28.00
7	7	14.00
8	8	28.00
9	9	0
10	10	28.00
11	11	0



- 10 Показать какие места предпочитают люди в зависимости от пола

```
3 with t1 as(  
4   select T.place_type, count(P.gender) as female  
5   from Ticket as T  
6   left join Passenger as P  
7   on T.passenger_id = P.id  
8   where P.gender = 'Ж'  
9   group by T.place_type),  
10 t2 as(  
11   select T.place_type, count(P.gender) as male  
12   from Ticket as T  
13   left join Passenger as P  
14   on T.passenger_id = P.id  
15   where P.gender = 'М'  
16   group by T.place_type)  
17 select t1.place_type, t1.female, t2.male  
18 from t1  
19 join t2  
20 on t1.place_type = t2.place_type
```

	place_type character varying (10)	female bigint	male bigint
1	сидячий	626	451
2	купе	529	413
3	плацкарт	555	426

- 10 Найти среднюю цену и ранг каждой цены для определенного рейса

```

3 select flight_id,price,
4 dense_rank() over(partition by flight_id order by price),
5 round(avg(price) over(partition by flight_id),2)
6 from Ticket
7 where flight_id = 1
8

```

Data Output   Сообщения   Notifications					
	flight_id integer	price integer	dense_rank bigint	round numeric	
1	1	2098	1	2519.60	
2	1	2127	2	2519.60	
3	1	2222	3	2519.60	
4	1	2682	4	2519.60	
5	1	3469	5	2519.60	

⑩ Найти количество мест для каждого поезда

```

3  with t1 as (
4      select V.train_id, V.id, count(*)
5      from Van as V
6      join Place as P1
7      on P1.van_id = V.id
8      group by V.id
9      order by V.id)
10 select t1.train_id, sum(t1.count)
11 from t1
12 group by t1.train_id
13 order by train_id

```

	train_id integer	sum numeric
1	1	49
2	2	128
3	3	96
4	4	107
5	5	102
6	6	92
7	7	68
8	8	52
9	9	103
10	10	62

- ⑩ Найти самого старшего пассажира для каждого рейса

```
3 select T.flight_id, max(age(current_date, p.born_date))
4 from Ticket as T
5 left join Passenger as P
6 on T.passenger_id = P.id
7 group by T.flight_id
8 order by flight_id
```

Data Output				Сообщения	Notifications
	flight_id integer		max interval		
1	1		102 years 1 mon 9 days		
2	2		115 years 11 mons 13 days		
3	3		68 years 2 mons 6 days		
4	4		119 years 27 days		
5	5		116 years 18 days		
6	6		117 years 9 mons 7 days		

- Найти имена пассажиров, цена билетов которых ниже чем средняя цена билетов

```

3 with t1 as (
4   select passenger_id as passenger,
5   price, round(avg(price) over(), 2) as avgprice
6   from Ticket
7 )
8 select t1.price, t1.avgprice, P.name
9 from t1
10 join Passenger as P
11 on t1.passenger = P.id
12 where t1.price < t1.avgprice
13 order by P.name

```

	price integer	avgprice numeric	name character varying (50)
1	2510	2774.27	Авдей
2	1997	2774.27	Авдей
3	2764	2774.27	Авдей
4	2624	2774.27	Авксентий
5	1803	2774.27	Авксентий
6	2724	2774.27	Авксентий
7	1307	2774.27	Авксентий
8	1937	2774.27	Авксентий
9	2368	2774.27	Агапа

- Для каждого типа места найти id поезда с наибольшим количеством мест

```

3 with t1 as (
4   select train_id, van_type, count(*) from van
5   left join train
6   on van.train_id = train.id
7   group by train_id, van_type
8   order by train_id),
9 t2 as (
10  select van_type, max(t1.count)
11  from t1
12  group by van_type
13  ),
14 t3 as (
15  select t1.*, t2.max
16  from t1
17  join t2
18  on t1.van_type = t2.van_type
19  where t1.count = t2.max)
20 select * from t3

```

	train_id integer	van_type character varying (20)	count bigint	max bigint
1	2	купе	6	6
2	111	плацкарт	6	6
3	169	сидячий	6	6
4	182	купе	6	6

- Найти поезда у которых один тип места

```

3  with t1 as (
4      select train_id, van_type from van
5      left join train
6      on van.train_id = train.id
7      group by train_id, van_type
8      order by train_id),
9  t2 as(
10     select train_id, count(*)
11     from t1
12     group by train_id)
13     select train_id from t2
14     where t2.count = 1

```

	train_id	
	integer	
1	15	
2	140	
3	149	
4	183	
5	199	

- Найти количество пассажиров для каждого рейса


```
3  select flight_id, count(*)
4  from Ticket
5  group by flight_id
6  order by flight_id
```

	flight_id integer	count bigint
1	1	5
2	2	11
3	3	8
4	4	13
5	5	5
6	6	7
7	7	7
8	8	7
9	9	7

- Найти пассажиров, имена которых начинаются на А



```
3 select P.name from Passenger as P
4 where P.name like 'A%'
```

	name character varying (50) 
1	Анисья
2	Анджела
3	Агриппина
4	Александра
5	Айрат
6	Алипий
7	Антипа
8	Азамат
9	Алешан