**1. Инструкция**

Начало работы:

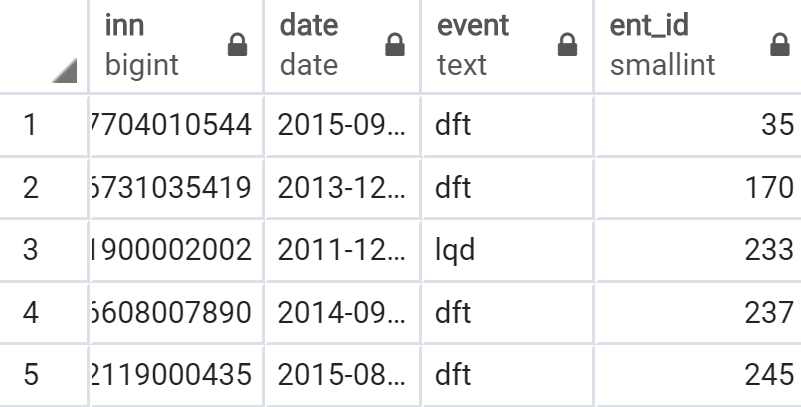
На первом этапе скачиваем и устанавливаем пакет программ Anaconda, включающий необходимые библиотеки для дальнейшего анализа и обработки данных в Python.

После установки заходим в приложение Jupyter Notebook и создаем новый блокнот Python 3, где в ячейках пишем запросы и исполняем с помощью клавиши Run на верхней панели.

Также для работы понадобится СУБД (система управления базами данных), где будут храниться используемые таблицы. В данной работе использована PostgreSQL, где уже были созданы, заполнены и связаны между собой таблицы в первом задании. Именно они будут использованы в дальшейшем при работе с Jupyter.

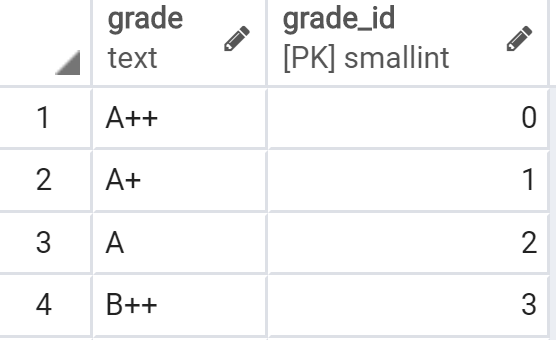
Краткое описание используемых таблиц:

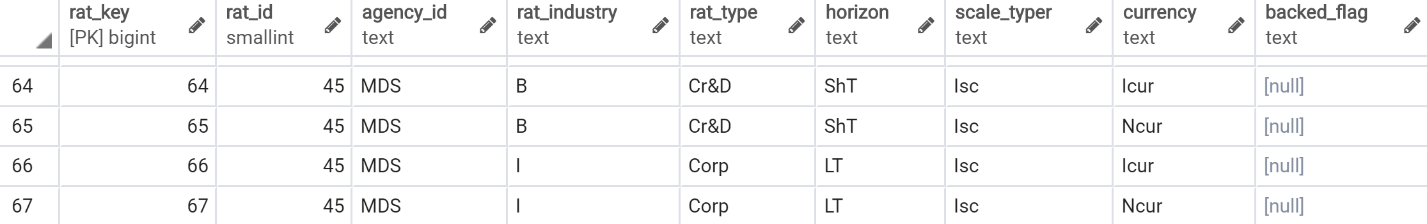
credit\_events\_task – таблица, содержащая информацию о кредитном событии, связанного с компанией и уникальный id для каждой компании



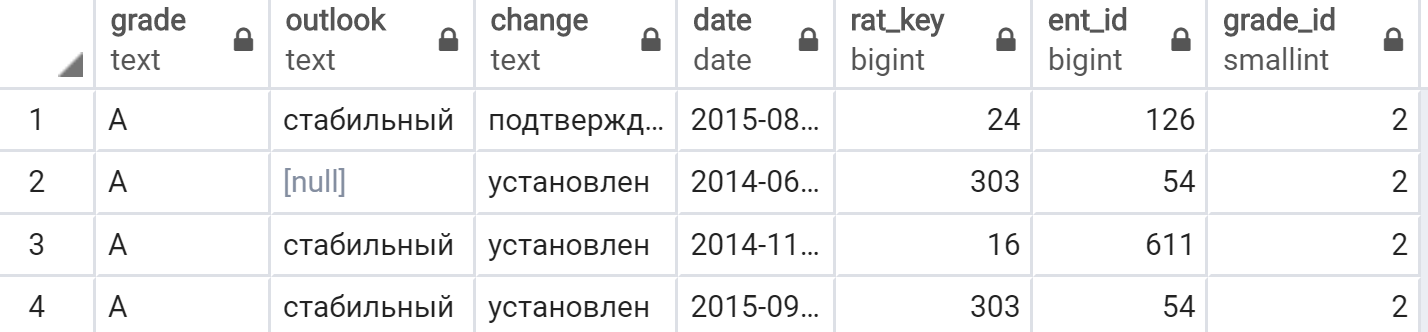
ent\_info - таблица, содержащая информацию о рейтингуемой компании и ее уникальном id

scale\_exp\_task – таблица, содержащая информацию о рейтинговой категории и уникальном id



ratings\_info - таблица, содержащая описание присвоенных рейтингов

ratings\_task - таблица, содержащая присвоенный рейтинг, дату присвоения, прогноз и изменение рейтинга для каждой компании



Описание запросов:

1. На первом этапе устанавливаем библиотеку для работы в Python с PostgreSQL. Для этого заходим в командную строку Anaconda Prompt и выполняем следующую команду:

- для Windows

**pip install psycopg2**

- для MacOS

**pip install psycopg2-binary**

2. Далее создается соединение Python с базой данных. Необходимо проверить называния параметров для соединения перед запуском.

**import psycopg2**

**import getpass** # для скрытия пароля (необязательная команда)

**conn=psycopg2.connect(user='postgres',**

**database='postgres',** # проверяем, чтобы название совпадало с названием в вашей БЗ

**host='localhost',**

**port='5432',**

**password=getpass.getpass('Insert password: ')**

**)**

**print(conn.get\_dsn\_parameters())**

3. Запрашиваем информацию из Python необходимую для следующего пункта с помощью pandas. Под необходимой информацией подразумеваются 2 таблицы. Первая включает данные об id компании, ее рейтинг на дату присвоения и саму дату. В выборку не включаются компании, рейтинг которых был снят или приостановлен. Таблица 2 содержит информацию об id компании, в которой произошел дефолт и дату дефолта, т. е. из исходной таблицы credit\_events\_task были убраны ликвидированные компании.

При построении запроса необходимо обратить внимание на разделение текстовых переменных символом \ , для того чтобы в Python они воспринимались как текст.

**import pandas as pd**

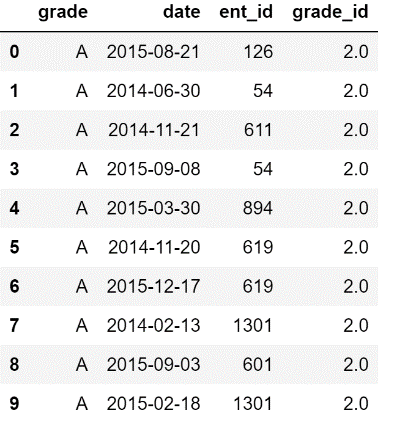
**query1="SELECT \"grade\", \"date\", \"ent\_id\", \"grade\_id\" FROM public.ratings\_task WHERE \"change\" <> 'снят' AND \"change\" <> 'приостановлен';"**

**quer\_dft="SELECT \"date\", \"ent\_id\" FROM public.****credit\_events\_task WHERE \"event\"='dft' AND \"ent\_id\" IS NOT NULL;"**

**data1=pd.read\_sql\_query(query1,conn)**

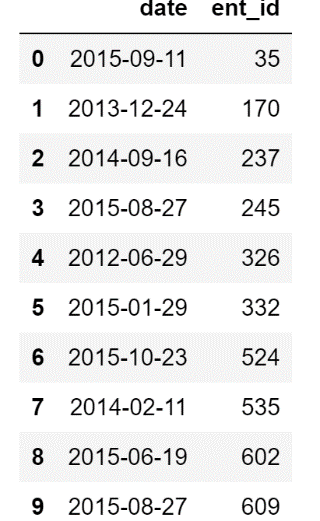
**data\_dft=pd.read\_sql\_query(quer\_dft,conn)**

**data1.head(10) #запрашиваем первые 10 элементов таблицы**

Результат 1:

**data\_dft.head(10) #запрашиваем первые 10 элементов таблицы**

Результат 2:



4. Написание функции, которая строит матричные характеристики рейтинговых переходов.

Справка по вычислению матриц:

1. Когортный метод (матрица P)

Каждый элемент в матрице – вероятность перехода из рейтинговой категории i в j или в состояние дефолта за период времени (Pij).

2. Метод дюраций (матрица A)

Элементы матрицы:

Написание функции:

Задаем аргументы функции: вид рейтинга (grade), границы периода наблюдения (начало периода - time\_1, конец периода time\_2) и тип матрицы (m\_type). Импортируем библиотеку pandas и создаем соединение с СУБД, где посредством SQL запроса создаем новую таблицу, с которой будем далее работать.

**def matrix\_fun(grade,time\_1, time\_2, m\_type, db\_login):**

**import pandas as pd**

**conn=psycopg2.connect(user=db\_login['user'],**

**database=db\_login['database'],**

**host=db\_login['host'],**

**port=db\_login['port'],**

**password=getpass.getpass('Insert password: '))**

**query="SELECT \* FROM (SELECT \"grade\", \"date\", \public.ratings\_task.\"ent\_id\", \"grade\_id\", \"event\", tbl1.\"date\_dft\" FROM public.ratings\_task LEFT JOIN (SELECT \"inn\", \"event\", \"ent\_id\", \"date\" as date\_dft FROM public.credit\_events\_task WHERE \"event\"='dft' AND \"ent\_id\" IS NOT NULL) as tbl1 ON tbl1.ent\_id=public.ratings\_task.ent\_id WHERE \"change\" <> 'снят' AND \"change\" <> 'приостановлен') WHERE \"TIME\">="+str(time\_1)+" AND \"TIME\"<="+str(time\_2)+" AND \"grade\_id\"="+m\_type**

**matrix=pd.read\_sql\_query(query,conn)**

Далее необходимо написать код, возвращающий матрицы, используя информацию из таблицы.