

## Установка PostgreSQL для пользователей Windows/Linux

Для пользователей *Windows* удобнее всего работать с *PostgreSQL* из-под *WSL*. Инструкция по установке *WSL* (*подсистема Linux для Windows*) есть в инструкции по установке *Docker* в материалах модуля 32.1 курса «Введение в *Data Science*».

Инструкция по установке *PostgreSQL* через командную строку *Linux Ubuntu* (*WSL*):

1. Установите необходимые пакеты программного обеспечения, которые будут использоваться для загрузки и установки сертификатов ПО для безопасного соединения через *SSL*:

**sudo apt install wget ca-certificates**

2. Получите сертификат, добавьте его в утилиту управления ключами *apt* и создайте новый файл конфигурации с официальным адресом репозитория *PostgreSQL* внутри:

**wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -**

**sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ \$(lsb\_release -cs)-pgdg main" >> /etc/apt/sources.list.d/pgdg.list'**

3. Перед фактической установкой рекомендуем всегда загружать информацию обо всех доступных для установки пакетах из настроенных вами источников:

**sudo apt update**

4. Установите последнюю версию *PostgreSQL* при фактической установке:

**sudo apt install postgresql postgresql-contrib**

5. Запустите службу:

**sudo service postgresql start**

6. Проверьте статус:

**sudo service postgresql status**

Если вы видите статус *online*, то всё сделано верно:

```
> sudo service postgresql status  
15/main (port 5432): online
```

## Установка PostgreSQL для пользователей macOS

1. Установите менеджер пакетов Homebrew следующей командой:

```
/bin/bash -c "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

2. Установите PostgreSQL с помощью Homebrew:

```
brew install postgres
```

3. Запустите PostgreSQL:

```
brew services start postgresql
```

Если команда не работает, запустите сервис вручную:

```
pg_ctl -D /usr/local/var/postgres start
```

4. Проверьте версию установленного сервиса:

```
brew info postgresql
```

## Восстановление дампа базы данных в PostgreSQL для пользователей Windows/Linux

Чтобы восстановить дамп базы данных, установите пароль для админского логина в PostgreSQL:

1. Зайдите в клиентский интерфейс PostgreSQL от пользователя postgres:

```
sudo -u postgres psql
```

2. Измените пароль командой:

```
\password postgres
```

После ввода команды будет предложено ввести новый пароль.


- Итак, когда PostgreSQL установлен, а также настроен пароль пользователя postgres, вам нужно отредактировать конфиг (например, через редактор nano):

**sudo nano /etc/postgresql/15/main/pg\_hba.conf**

В зависимости от версии, пути к конфигу могут различаться.

- Внутри конфига найдите строку, как на скриншоте, и измените значение в строке с peer на md5:

```
# Database administrative login by Unix domain socket
local all postgres md5
```



- Перезагрузите сервер:

**sudo service postgresql restart**

- Создайте базу, в которую будет заливаться дамп:

**createdb -U postgres module\_6**

- Находясь в директории с дампом, залейте базу командой:

**pg\_restore -U postgres -d module\_6 database.dump**

- Проверьте залитую базу. Для этого войдите в командный интерфейс от пользователя postgres:

**sudo -u postgres psql**

Командой \l (или \list) просмотрите базы и найдите залитую базу:

List of databases							
Name	Owner	Encoding	Collate	Ctype	ICU Locale	Locale Provider	Access privileges
module_6	postgres	UTF8	C.UTF-8	C.UTF-8		libc	
postgres	postgres	UTF8	C.UTF-8	C.UTF-8		libc	

Для подключения введите команду \c <имя базы>:

**\c module\_6**

Командой \dt проверьте таблицы в базе:

```
module_6=# \dt
```

List of relations

Schema	Name	Type	Owner
public	d_agreement	table	postgres
public	d_clients	table	postgres
public	d_close_loan	table	postgres
public	d_job	table	postgres
public	d_last_credit	table	postgres
public	d_loan	table	postgres
public	d_pens	table	postgres
public	d_salary	table	postgres
public	d_work	table	postgres
(9 rows)			

Можно также сделать выборку из базы:

```

module_6=# select * from d_agreement limit 10;
 agreement_rk | id_client | target
-----+-----+-----
      59910150 | 106804370 |      0
      59910230 | 106804371 |      0
      59910525 | 106804372 |      0
      59910803 | 106804373 |      0
      59911781 | 106804374 |      0
      59911784 | 106804375 |      0
      59911832 | 106804376 |      0
      59912034 | 106804377 |      0
      59912560 | 106804378 |      0
      59912659 | 106804379 |      0
(10 rows)

```

## Восстановление дампа базы данных в PostgreSQL для пользователей MacOS

1. Подключение на macOS немного отличается от Linux. Проверьте пользователей:

**psql -l**

2. Для создания базы укажите значение из столбца Owner таблицы, которая выведена командой из пункта 1:

**createdb -U <username> module\_6**

3. Залейте дамп в базу, находясь в директории с файлом database.dump:

**pg\_restore -U <username> -d module\_6 database.dump**

4. Попасть в базу для проверки можно так:

**sudo psql -U <username> -d module\_6**

## Считывание данных из базы в Python

1. Для подключения к базе через Python используется специальный адаптер — psycopg2. Его можно установить через менеджер пакетов pip:

### pip install psycopg2-binary

2. Импортируйте адаптер и вызовите функцию connect, которая создаст подключение:

```
import psycopg2 as pg

conn = pg.connect(
    dbname='module_6', user='postgres', password='postgres', host='localhost'
)
cursor = conn.cursor()
```

3. Обращаясь к методам курсора execute и fetchall, можно делать SQL-запросы. Выполните два запроса: первым считайте имена таблиц из базы, а вторым — первые десять записей из таблицы d\_agreement:

```
cursor.execute('''
    SELECT table_name
    FROM information_schema.tables
    WHERE table_schema='public'
    AND table_type='BASE TABLE';
''')
cursor.fetchall()
```

```
[('d_work',),
 ('d_clients',),
 ('d_agreement',),
 ('d_last_credit',),
 ('d_job',),
 ('d_loan',),
 ('d_close_loan',),
 ('d_salary',),
 ('d_pens',)]
```

```

cursor.execute('''
    SELECT *
    FROM d_agreement
    LIMIT 10;
''')
cursor.fetchall()

```

```

[(Decimal('59910150'), Decimal('106804370'), Decimal('0')),
 (Decimal('59910230'), Decimal('106804371'), Decimal('0')),
 (Decimal('59910525'), Decimal('106804372'), Decimal('0')),
 (Decimal('59910803'), Decimal('106804373'), Decimal('0')),
 (Decimal('59911781'), Decimal('106804374'), Decimal('0')),
 (Decimal('59911784'), Decimal('106804375'), Decimal('0')),
 (Decimal('59911832'), Decimal('106804376'), Decimal('0')),
 (Decimal('59912034'), Decimal('106804377'), Decimal('0')),
 (Decimal('59912560'), Decimal('106804378'), Decimal('0')),
 (Decimal('59912659'), Decimal('106804379'), Decimal('0'))]

```

Для удобства можете сразу считывать данные в датафрейм Pandas функцией `read_sql`, которая первым аргументом будет принимать запрос, а вторым — подключение к базе:

```

import pandas as pd
import psycopg2 as pg

conn = pg.connect(
    dbname='module_6', user='postgres', password='postgres', host='localhost'
)
query = '''
    SELECT *
    FROM d_agreement
    LIMIT 10;
'''

d_agreement_df = pd.read_sql(query, conn)
d_agreement_df.head(15)

```

	agreement_rk	id_client	target
0	59910150.0	106804370.0	0.0
1	59910230.0	106804371.0	0.0
2	59910525.0	106804372.0	0.0
3	59910803.0	106804373.0	0.0
4	59911781.0	106804374.0	0.0
5	59911784.0	106804375.0	0.0
6	59911832.0	106804376.0	0.0
7	59912034.0	106804377.0	0.0
8	59912560.0	106804378.0	0.0
9	59912659.0	106804379.0	0.0