МОСКОВСКИЙ
ГОРОДСКОЙ
УНИВЕРСИТЕТ
МГПУ

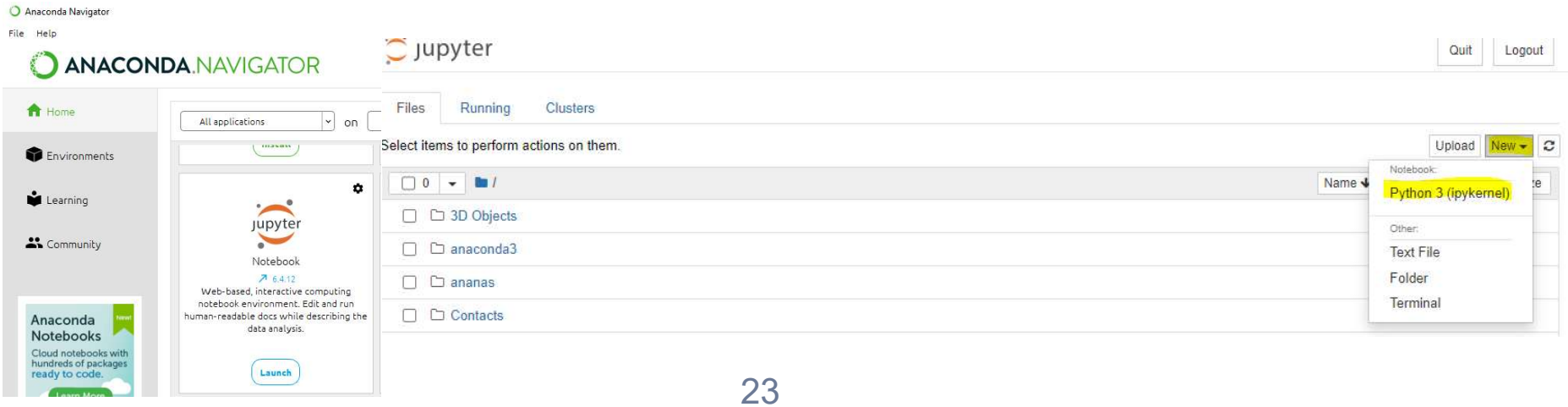
Использование Python с базой данных

Упражнение 2. Экспорт данных из базы данных в Python

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

1. Загрузите и установите Anaconda: https://www.anaconda.com/distribution/
2. Code in the Cloud https://www.anaconda.com/products/distribution/start-coding-immediately
3. После установки откройте **Terminal** для Mac или **cmd** для Windows. Введите «**python**» в командной строке/

```
C:\windows\system32>python
Python 3.10.2 (tags/v3.10.2:a58ebcc, Jan 17 2022, 14:12:15) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
```



Упражнение 2. Экспорт данных из базы данных в Python

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

4. Далее нужно установить клиент базы данных **PostgreSQL** для Python, **psycopg2**.
5. Загрузить и установить этот пакет с помощью менеджера пакетов **Anaconda**, **conda**

Ввести следующую команду в командной строке, чтобы установить клиент базы данных **Postgres**:

conda install psycopg2

Упражнение 2. Экспорт данных из базы данных в Python

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

4. Введите **python** в командной строке, чтобы открыть интерпретатор Python.

5. Пишем скрипт Python для загрузки данных:

```
import psycopg2
with psycopg2.connect(host="ip", user="login", password="pass", dbname="db_name", port=5432) as conn:
    with conn.cursor() as cur:
        cur.execute("SELECT * FROM customers LIMIT 5")
        records = cur.fetchall()
records
```

```
[(1,
  None,
  'Arlena',
  'Riveles',
  None,
  'ariveles0@stumbleupon.com',
  'F',
  '98.36.172.246',
  None,
  None,
  None,
  None,
  None,
  None,
  None,
  datetime.datetime(2017, 4, 23, 0, 0)),
```

Упражнение 2. Экспорт данных из базы данных в Python

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

6. Теперь, когда у нас есть соединение, нам нужно создать объект курсора, который позволит читать из базы данных.

conn.cursor() создает объект курсора базы данных, что позволяет выполнять SQL в соединении с базой данных, а оператор **with** позволяет нам автоматически удалять курсор, когда он нам больше не нужен.

7. **records = cur.fetchall()** извлекает все оставшиеся строки в результате запроса и присваивает эти строки переменной records.
8. Теперь, когда отправили запрос в базу данных и получили записи, можем сбросить уровень отступа, можем просмотреть результат, введя выражение (в данном случае только запись имени переменной) и нажав Enter. Эти выходные данные представляют собой пять записей о клиентах, которые извлекли.

Улучшение доступа к Postgres в Python с помощью SQLAlchemy и Pandas

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

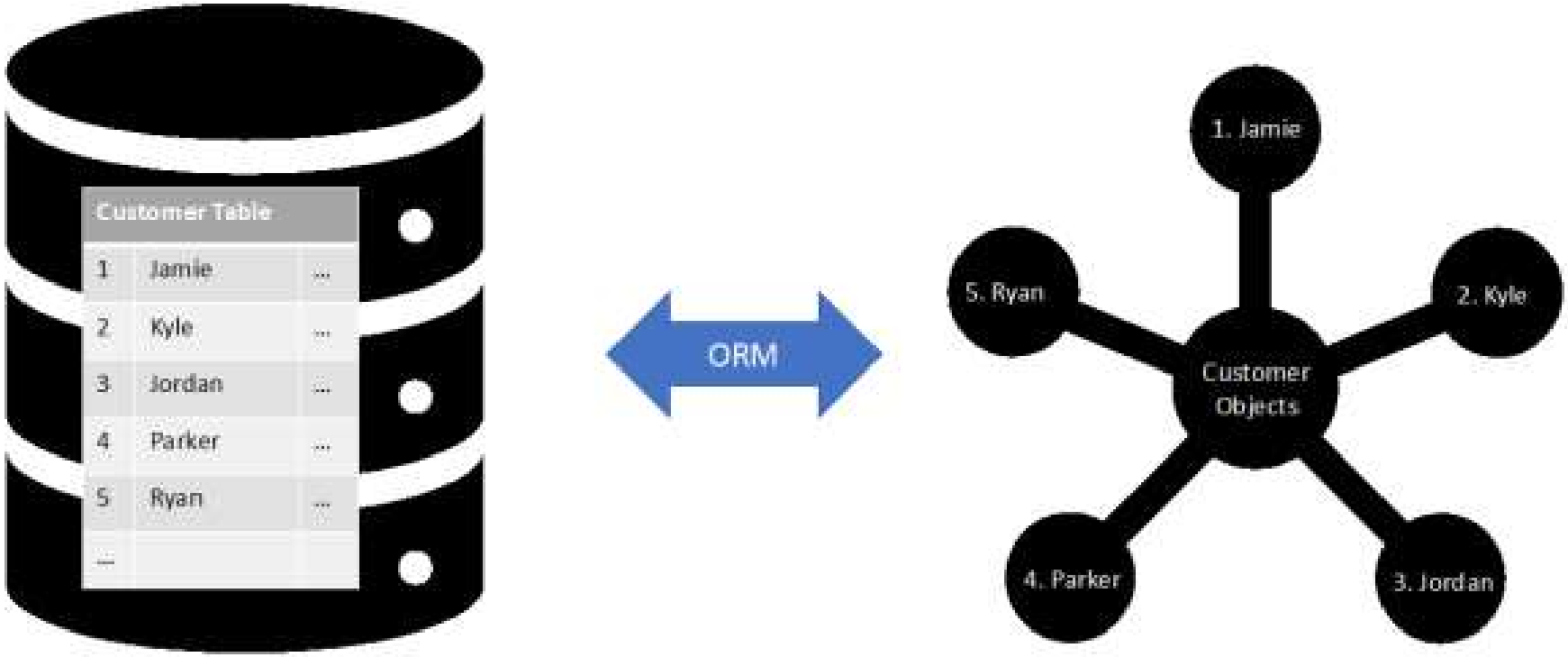
Хотя **psycopg2** — это мощный клиент базы данных для доступа к Postgres из Python, мы можем упростить код, используя несколько других пакетов, а именно Pandas и **SQLAlchemy**.

У **Pandas** также есть высокоуровневые API, которые позволят считывать данные из базы данных всего за несколько строк кода.

Улучшение доступа к Postgres в Python с помощью SQLAlchemy и Pandas

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ
МГПУ

Объектно-реляционный преобразователь сопоставляет строки в базе данных с объектами в



Улучшение доступа к Postgres в Python с помощью SQLAlchemy и Pandas

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

SQLAlchemy — это набор инструментов SQL. Хотя он предлагает отличные функциональные возможности, ключевым преимуществом, на котором мы сосредоточимся здесь, является объект **SQLAlchemy Engine**.

Объект **SQLAlchemy Engine** содержит информацию о типе базы данных (в нашем случае PostgreSQL) и пуле соединений. Пул соединений допускает несколько одновременных подключений к базе данных.

Улучшение доступа к Postgres в Python с помощью SQLAlchemy и Pandas

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Пул соединений также удобен тем, что он не создает соединение до тех пор, пока запрос не будет отправлен для выполнения. Поскольку эти соединения не формируются до тех пор, пока запрос не будет выполнен, говорят, что объект **Engine** демонстрирует **ленивую инициализацию**. Термин «**ленивый**» используется для обозначения того, что ничего не происходит (соединение не устанавливается) до тех пор, пока не будет сделан запрос. Это удобно, поскольку минимизирует время соединения и снижает нагрузку на базу данных.

Еще одним преимуществом **SQLAlchemy Engine** является то, что он автоматически фиксирует (автоматически фиксирует) изменения в базе данных из-за **CREATE TABLE**, **UPDATE**, **INSERT** или других операторов, которые изменяют базу данных.

Использование Python с ноутбуками Jupyter

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

```
from sqlalchemy import create_engine
import pandas as pd
```

настроим ноутбук для отображения графиков и визуализаций в строке. Мы можем сделать это с помощью следующей команды:

```
%matplotlib inline
```

В следующей ячейке определим строку подключения:

```
cnxn_string = ("postgresql+psycopg2://{username}:{pswd}"
               "@{host}:{port}/{database}")
print(cnxn_string)
```

Использование Python с ноутбуками Jupyter

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

заполнить параметры и создать базу данных

```
engine = create_engine(cnxn_string.format(
    username="your_username",
    pswd="your_password",
    host="your_host",
    port=5432,
    database="your_database_name"))
```

Поскольку **SQLAlchemy** ленив, не узнаем, было ли соединение с базой данных успешным, пока не попытаемся отправить команду, можем проверить, работает ли этот механизм базы данных, выполнив следующую команду

```
engine.execute("SELECT * FROM customers LIMIT 2;").fetchall()
```

```
[(1, None, 'Arlena', 'Riveles', None, 'ariveles0@stumbleupon.com', 'F', '98.36.172.246', None, None, None, None, None, None, No
ne, datetime.datetime(2017, 4, 23, 0, 0)),
 (2, 'Dr', 'Ode', 'Stovin', None, 'ostovin1@npr.org', 'M', '16.97.59.186', '314-534-4361', '2573 Fordem Parkway', 'Saint Loui
s', 'MO', '63116', 38.5814, -90.2625, datetime.datetime(2014, 10, 2, 0, 0))]
```

Чтение и запись в базу данных с помощью Pandas

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Python поставляется с отличными структурами данных, включая списки, словари и кортежи. Хотя это полезно, наши данные часто могут быть представлены в форме таблицы со строками и столбцами, аналогично тому, как мы храним данные в нашей базе данных. Для этих целей особенно полезен объект **DataFrame** в **Pandas**.

Pandas предлагает:

- Функциональность для чтения данных непосредственно из базы данных
- Визуализация данных
- Инструменты анализа данных

Чтение и запись в базу данных с помощью Pandas

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

Можем использовать объект **SQLAlchemy Engine** для чтения данных в **Pandas DataFrame**:

```
customers_data = pd.read_sql_table('customers', engine)
```

Упражнение 2. Выполнение визуализации данных с помощью Pandas

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

В этом упражнении будем считывать данные из базы данных и визуализировать результаты с помощью Python, блокнотов Jupyter, SQLAlchemy и Pandas.

Будем анализировать демографическую информацию о клиентах по городам, чтобы лучше понять нашу целевую аудиторию.

1. Откройте блокнот Jupyter из предыдущего раздела и щелкните последнюю пустую ячейку.
2. Введите следующий запрос, заключенный в тройные кавычки (тройные кавычки позволяют использовать строки, занимающие несколько строк в Python):

Упражнение 2. Выполнение визуализации данных с помощью Pandas

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

1. Откройте блокнот Jupyter из предыдущего раздела и щелкните последнюю пустую ячейку.
2. Введите следующий запрос, заключенный в тройные кавычки (тройные кавычки позволяют использовать строки, занимающие несколько строк в Python):

```
query = """
    SELECT city,
           COUNT(1) AS number_of_customers,
           COUNT(NULLIF(gender, 'M')) AS female,
           COUNT(NULLIF(gender, 'F')) AS male
    FROM customers
    WHERE city IS NOT NULL
    GROUP BY 1
    ORDER BY 2 DESC
    LIMIT 10
"""
```

```
top_cities_data = pd.read_sql_query(query, engine)
```

```
top_cities_data
```

	city	number_of_customers	female	male
0	Washington	1447	734	713
1	Houston	904	446	458
2	New York City	731	369	362
3	El Paso	713	369	344
4	Dallas	607	309	298
5	Atlanta	571	292	279
6	Sacramento	506	244	262
7	Los Angeles	466	241	225
8	San Antonio	426	207	219
9	Miami	426	195	231

Упражнение 2. Выполнение визуализации данных с помощью Pandas

МОСКОВСКИЙ ГОРОДСКОЙ УНИВЕРСИТЕТ МГПУ

1. Теперь получим график количество мужчин и женщин в каждом из 10 лучших городов. Поскольку мы хотим просмотреть статистику для каждого города отдельно, мы можем использовать простую гистограмму для просмотра данных:

```
ax = top_cities_data.plot.bar('city', y=['female', 'male'], title='Number of Customers by Gender and City')
```