

Департамент образования и науки города Москвы
Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»
Институт цифрового образования
Департамент информатики, управления и технологий

Отчёт по лабораторной работе № 4.1

«Сравнение подходов хранения больших данных»

Вариант №8

Выполнила:

студентка группы АДЭУ-211,
Макарова Екатерина Павловна

Преподаватель:

Босенко Тимур Муртазович,
Доцент, кандидат технических наук

Москва 2024

Цель работы: сравнить производительность и эффективность различных подходов к хранению и обработке больших данных на примере реляционной базы данных PostgreSQL и документно-ориентированной базы данных MongoDB.

Задача: сформировать данные или использовать источники данных не менее 1000 записей для обычных данных и 100000 для больших данных. Оценить эффективность работы с геопространственными данными на примере системы отслеживания корпоративного транспорта.

Оборудование и программное обеспечение:

- Компьютер с операционной системой Ubuntu.
- PostgreSQL.
- MongoDB.
- Python 3.x.
- Библиотеки: psycopg2, pymongo, pandas, matplotlib.

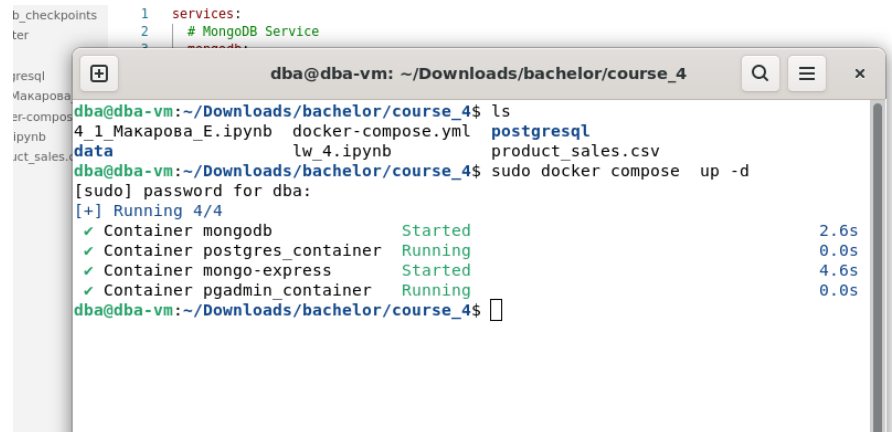
Теоретическая часть

В современном мире объемы данных растут экспоненциально, что приводит к необходимости использования эффективных методов их хранения и обработки. Существует два основных подхода к хранению больших данных:

1. Реляционные базы данных (например, PostgreSQL)
 2. NoSQL базы данных (например, MongoDB)
- Каждый из этих подходов имеет свои преимущества и недостатки, которые мы рассмотрим в ходе выполнения лабораторной работы.

Практическая часть

1. Запуск docker контейнера с MongoDB и PostgreSQL (Рис. 1).



```
services:
  # MongoDB Service
  mongodb:
    image: mongo:4.2
    container_name: mongodb
    restart: always
    ports:
      - 27017:27017
  postgresql:
    image: postgres:12
    container_name: postgresql
    restart: always
    ports:
      - 5432:5432
    environment:
      POSTGRES_DB: postgres
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres

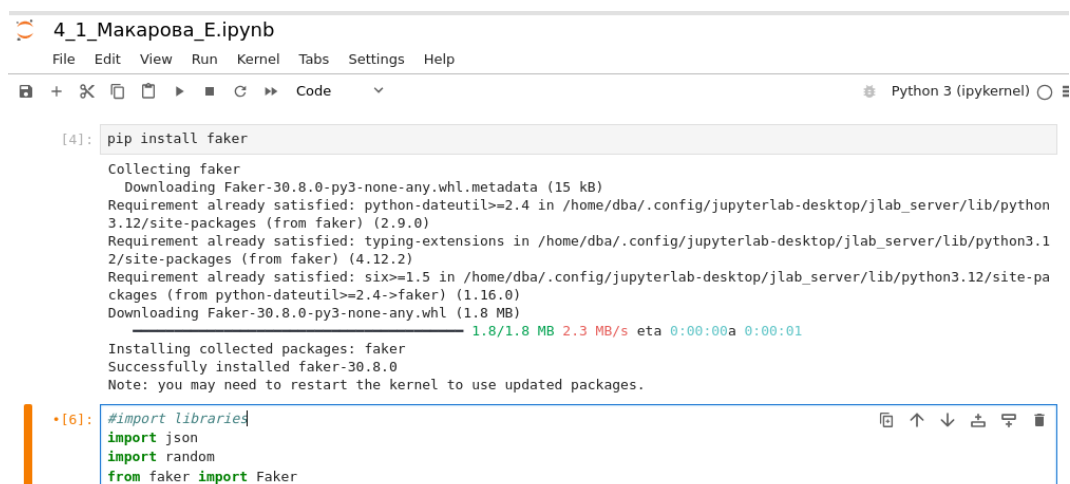
data:
  product_sales.csv
  lw_4.ipynb

dba@dba-vm: ~/Downloads/bachelor/course_4$ ls
4_1_Макарова_E.ipynb  docker-compose.yml  postgresql
data                  lw_4.ipynb          product_sales.csv
dba@dba-vm:~/Downloads/bachelor/course_4$ sudo docker compose up -d
[sudo] password for dba:
[+] Running 4/4
 ✓ Container mongodb                Started      2.6s
 ✓ Container postgres_container     Running      0.0s
 ✓ Container mongo-express          Started      4.6s
 ✓ Container pgadmin_container      Running      0.0s
dba@dba-vm:~/Downloads/bachelor/course_4$
```

Рис. 1

2. Генерация данных

Импорт необходимых библиотек для генерации данных по корпоративному транспорту (Рис. 2).



```
4_1_Макарова_E.ipynb
File Edit View Run Kernel Tabs Settings Help

[4]: pip install faker

Collecting faker
  Downloading Faker-30.8.0-py3-none-any.whl.metadata (15 kB)
Requirement already satisfied: python-dateutil>=2.4 in /home/dba/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (from faker) (2.9.0)
Requirement already satisfied: typing-extensions in /home/dba/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (from faker) (4.12.2)
Requirement already satisfied: six>=1.5 in /home/dba/.config/jupyterlab-desktop/jlab_server/lib/python3.12/site-packages (from python-dateutil>=2.4->faker) (1.16.0)
Downloading Faker-30.8.0-py3-none-any.whl (1.8 MB)
1.8/1.8 MB 2.3 MB/s eta 0:00:00a 0:00:01

Installing collected packages: faker
Successfully installed faker-30.8.0
Note: you may need to restart the kernel to use updated packages.

[6]: #import libraries
import json
import random
from faker import Faker
```

Рис. 2

2.1. Генерация 100 записей (Рис. 3).

```
4_1_Макарова_Е.ipynb
File Edit View Run Kernel Tabs Settings Help

[23]: fake = Faker() # Создание экземпляра Faker

n_records = 100
categories = ["Moving", "Stopped", "Maintenance"]

# Создание списка для хранения данных
vehicles = []

for _ in range(n_records):
    vehicle = {
        "vehicle_id": fake.license_plate(),
        "timestamp": fake.date_time_this_month(),
        "latitude": fake.latitude(),
        "longitude": fake.longitude(),
        "speed": random.uniform(0, 120),
        "fuel_level": random.uniform(0, 100),
        "status": random.choice(categories)
    }
    vehicles.append(vehicle)

# Создание DataFrame из списка словарей
vehicle_df_100 = pd.DataFrame(vehicles)

vehicle_df_100
```

| | vehicle_id | timestamp | latitude | longitude | speed | fuel_level | status |
|---|------------|----------------------------|-------------|-------------|-----------|------------|-------------|
| 0 | 751U3 | 2024-10-03 13:24:18.473794 | 17.864943 | 142.079573 | 30.698638 | 68.800400 | Moving |
| 1 | 1NG 779 | 2024-10-10 17:23:37.582807 | -16.892347 | 119.420863 | 57.073455 | 10.464161 | Maintenance |
| 2 | 44-1115E | 2024-10-10 21:29:17.310157 | 63.9580785 | -126.106468 | 54.662590 | 67.752933 | Stopped |
| 3 | 0-7634M | 2024-10-11 22:24:51.992943 | -30.5158775 | -78.303968 | 87.656107 | 84.808572 | Maintenance |
| 4 | F37-53O | 2024-10-11 19:47:57.264808 | -56.039767 | 99.704443 | 40.514754 | 53.195871 | Maintenance |

Рис. 3

2.2. Генерация 100000 строк данных

```
4_1_Макарова_Е.ipynb
File Edit View Run Kernel Tabs Settings Help

[25]: fake = Faker() # Создание экземпляра Faker

n_records = 100000
categories = ["Moving", "Stopped", "Maintenance"]

# Создание списка для хранения данных
vehicles = []

for _ in range(n_records):
    vehicle = {
        "vehicle_id": fake.license_plate(),
        "timestamp": fake.date_time_this_month(),
        "latitude": fake.latitude(),
        "longitude": fake.longitude(),
        "speed": random.uniform(0, 120),
        "fuel_level": random.uniform(0, 100),
        "status": random.choice(categories)
    }
    vehicles.append(vehicle)

# Создание DataFrame из списка словарей
vehicle_df_100000 = pd.DataFrame(vehicles)

vehicle_df_100000
```

| | vehicle_id | timestamp | latitude | longitude | speed | fuel_level | status |
|-------|------------|----------------------------|-------------|------------|-----------|------------|-------------|
| 0 | EMN B60 | 2024-10-09 11:52:52.527041 | -36.695143 | 28.782672 | 84.256133 | 60.440604 | Moving |
| 1 | 6W 0724G | 2024-10-24 18:25:19.939903 | -13.587268 | -99.664368 | 73.685382 | 47.564543 | Maintenance |
| 2 | WFZ1653 | 2024-10-11 14:24:00.650290 | 69.241028 | 174.731238 | 55.879038 | 69.589794 | Moving |
| 3 | MOJ-660 | 2024-10-03 05:50:34.912282 | 34.813216 | 49.977729 | 78.546407 | 86.332193 | Moving |
| 4 | 873 1WY | 2024-10-04 20:19:02.490102 | 47.0613365 | -82.257132 | 19.568204 | 86.388251 | Maintenance |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 99995 | MUR 076 | 2024-10-06 12:28:27.139102 | -36.6207385 | -52.944329 | 49.106339 | 50.567872 | Maintenance |
| 99996 | JYV-6311 | 2024-10-13 06:49:15.457808 | 34.409509 | -77.970189 | 92.708543 | 15.162408 | Stopped |
| 99997 | CZL-9915 | 2024-10-15 02:32:40.201120 | 89.6447285 | 155.473179 | 59.739902 | 14.813175 | Stopped |
| 99998 | 187 XVO | 2024-10-11 11:14:53.147157 | 85.9899895 | 107.945237 | 61.074852 | 73.258937 | Moving |
| 99999 | 572 AOO | 2024-10-12 09:52:05.183694 | -44.592286 | 160.675507 | 41.309040 | 95.036984 | Moving |

100000 rows x 7 columns

Рис. 4

3. Сохранение данных в CSV файл

```
[28]: #Сохранение в CSV для дальнейшего использования
vehicle_df_100.to_csv('vehicle_df_100.csv', index=False)

print("Данные сгенерированы и сохранены в vehicle_df_100.csv")

Данные сгенерированы и сохранены в vehicle_df_100.csv

[29]: #Сохранение в CSV для дальнейшего использования
vehicle_df_100000.to_csv('vehicle_df_100000.csv', index=False)

print("Данные сгенерированы и сохранены в vehicle_df_100000.csv")

Данные сгенерированы и сохранены в vehicle_df_100000.csv
```

Рис. 5

4. Подключение к MongoDB и загрузка данных

4.1. Загрузка 100 записей в MongoDB коллекцию «vehicle_tracking_100» (Рис. 6).

```
[53]: # Подключение к MongoDB
mongo_client = MongoClient('mongodb://mongouser:mongopass@localhost:27017/')
if check_mongo_connection(mongo_client):
    mongo_db = mongo_client['ecommerce_analytics']
    collection = mongo_db['vehicle_tracking_100']
    print('Ok')
else:
    print('Lost connection')

Успешное подключение к MongoDB
Ok

[70]: records = vehicle_df_100.to_dict('records') # Преобразование DataFrame в список словарей

[71]: # Загрузка данных в MongoDB
collection.insert_many(records)
print("Данные загружены в MongoDB")

Данные загружены в MongoDB
```

Рис. 6

Проверка загруженных данных в MongoExpress (Рис. 7)

Mongo Express Database: ecommerce_analytics Collection: vehicle_tracking_100

Viewing Collection: vehicle_tracking_100

New Document New Index

Simple Advanced

Key Value String Find

Delete all 100 documents retrieved

1 2 3 4 5 > >>











| _id | vehicle_id | timestamp | latitude | longitude | speed | fuel_level | status |
|---|------------|--|-------------|------------|--------------------|--------------------|-------------|
|   671aa7f78babeceaabb15eb1 | 24VQ2 | Sat Oct 05 2024 09:20:01 GMT+0000 (Coordinated Universal Time) | 50.386624 | 35.630166 | 34.3329558378202 | 31.670815107023685 | Moving |
|   671aa7f78babeceaabb15eb2 | 5-Q1087 | Thu Oct 10 2024 03:43:03 GMT+0000 (Coordinated Universal Time) | -44.2505275 | -42.217634 | 40.85774711057108 | 38.74725507963318 | Maintenance |
|   671aa7f78babeceaabb15eb3 | 95T 9540 | Sun Oct 20 2024 06:59:37 GMT+0000 (Coordinated Universal Time) | 30.3260355 | 53.158396 | 104.08592138586928 | 86.3686397049866 | Maintenance |
|   671aa7f78babeceaabb15eb4 | 2K 20956 | Fri Oct 04 2024 12:00:48 GMT+0000 (Coordinated Universal Time) | -12.9830345 | 149.084532 | 27.875098394928088 | 80.6497722941275 | Moving |
|   671aa7f78babeceaabb15eb5 | F 339611 | Sun Oct 13 2024 00:49:22 GMT+0000 (Coordinated Universal Time) | 87.3519 | 99.179476 | 35.26187750437133 | 47.97011891611754 | Maintenance |

Рис. 7

4.2. Загрузка 100000 записей в MongoDB коллекцию «vehicle_tracking_100000» (Рис. 8)

```
6]: mongo_client = MongoClient('mongodb://mongouser:mongopass@localhost:27017/')
if check_mongo_connection(mongo_client):
    mongo_db = mongo_client['ecommerce_analytics']
    collection1 = mongo_db['vehicle_tracking_100000']
    print('ok')
else:
    print('Lost connection')

Успешное подключение к MongoDB
Ok

7]: records1 = vehicle_df_100000.to_dict('records')

8]: # Загрузка данных в MongoDB
collection1.insert_many(records1)
print("Данные загружены в MongoDB")

Данные загружены в MongoDB
```

Рис. 8

Проверка данных в MongoExpress (Рис. 9).

Mongo Express Database: ecommerce_analytics Collection: vehicle_tracking_100000

Viewing Collection: vehicle_tracking_100000

New Document New Index

Simple Advanced

Key

Value

String

Find

Delete all 100000 documents retrieved

1 2 3 4 5 > >>









| _id | vehicle_id | timestamp | latitude | longitude | speed | fuel_level | status |
|--|------------|--|------------|-------------|--------------------|--------------------|-------------|
|   671aabb8babeceaabb15f17 | N 516314 | Sat Oct 12 2024 13:38:36 GMT+0000 (Coordinated Universal Time) | -71.196244 | -106.739031 | 69.37489896015438 | 17.938573276706283 | Stopped |
|   671aabb8babeceaabb15f18 | 219-KBQ | Sat Oct 19 2024 13:08:48 GMT+0000 (Coordinated Universal Time) | -38.201303 | 153.121184 | 104.52368550807378 | 86.23429004107417 | Maintenance |
|   671aabb8babeceaabb15f19 | X 162608 | Thu Oct 10 2024 18:04:12 GMT+0000 (Coordinated Universal Time) | -45.152203 | 97.154357 | 81.27744131303854 | 46.67552521636714 | Stopped |
|   671aabb8babeceaabb15f1a | MJL 510 | Sat Oct 12 2024 10:30:05 GMT+0000 (Coordinated Universal Time) | 49.7923935 | -173.221437 | 59.35738765977808 | 44.01825232843747 | Stopped |

Рис. 9

5. Подключение к PostgreSQL и загрузка данных

5.1. Подключение к PostgreSQL (Рис. 10).

```

try:
    conn = psycopg2.connect(
        dbname="vehicle",
        user="postgres",
        password="changeme",
        host="localhost", # или "postgres", если Jupyter в контейнере
        port="5432"
    )
    print("Подключение успешно")
    conn.close()
except Exception as e:
    print(f"Ошибка подключения: {e}")

```

Подключение успешно

Рис. 10

5.2. Загрузка 100 записей в PostgreSQL (Рис. 11).

```

: # Подключение к PostgreSQL
pg_conn_params = {
    "dbname": "vehicle",
    "user": "postgres",
    "password": "changeme",
    "host": "localhost",
    "port": "5432"
}
pg_conn = check_postgres_connection(pg_conn_params)
if pg_conn:
    try:
        # Создание таблицы
        with pg_conn.cursor() as cur:
            cur.execute("""
                CREATE TABLE IF NOT EXISTS vehicle_100(
                    vehicle_id VARCHAR(10),
                    timestamp DATE,
                    latitude FLOAT,
                    longitude FLOAT,
                    speed FLOAT,
                    fuel_level FLOAT,
                    status VARCHAR(30)
                )
            """)
        # Загрузка данных
        with pg_conn.cursor() as cur:
            for _, row in vehicle_df_100.iterrows():
                cur.execute("""
                    INSERT INTO vehicle_100 (vehicle_id, timestamp, latitude, longitude, speed, fuel_level, status)
                    VALUES (%s, %s, %s, %s, %s, %s, %s)
                """, (row['vehicle_id'], row['timestamp'], row['latitude'], row['longitude'],
                    row['speed'], row['fuel_level'], row['status']))

            pg_conn.commit()
            print("Данные загружены в PostgreSQL")

    except Exception as e:
        print(f"Ошибка при работе с PostgreSQL: {e}")
    finally:
        pg_conn.close()
else:
    print("Пропуск операций с PostgreSQL из-за ошибки подключения")

```

Успешное подключение к PostgreSQL
Данные загружены в PostgreSQL

Рис. 11

5.3. Проверка загрузки данных в pg admin (Рис. 12)

public.vehicle_100/vehicle/postgres@makarova_e

Query Query History

1 SELECT * FROM public.vehicle_100
2 LIMIT 100
3

Data Output Messages Notifications

| | vehicle_id character varying (10) | timestamp date | latitude double precision | longitude double precision | speed double precision | fuel_level double precision | status character varying (30) |
|----|--------------------------------------|-------------------|------------------------------|-------------------------------|---------------------------|--------------------------------|----------------------------------|
| 1 | 24VQ2 | 2024-10-05 | 50.386624 | 35.630166 | 34.3329558378202 | 31.670815107023685 | Moving |
| 2 | 5-Q1087 | 2024-10-10 | -44.2505275 | -42.217634 | 40.85774711057108 | 38.74725507963318 | Maintenance |
| 3 | 95T 9540 | 2024-10-20 | 30.3260355 | 53.158396 | 104.08592138586928 | 86.3686397049866 | Maintenance |
| 4 | 2K 20956 | 2024-10-04 | -12.9830345 | 149.084532 | 27.875098394928088 | 80.6497722941275 | Moving |
| 5 | F 339611 | 2024-10-13 | 87.3519 | 99.179476 | 35.26187750437133 | 47.97011891611754 | Maintenance |
| 6 | 873 SMY | 2024-10-19 | 2.716775 | 17.459836 | 82.31848470432922 | 62.95747862556491 | Stopped |
| 7 | 666 DXS | 2024-10-22 | 79.619553 | -108.691704 | 25.087756061510518 | 28.601619338313235 | Maintenance |
| 8 | 540-335 | 2024-10-20 | 3.517305 | 2.196363 | 67.18321502831328 | 91.44197131380332 | Stopped |
| 9 | PEH 5208 | 2024-10-14 | -42.739649 | 138.753586 | 95.89922518695263 | 7.048074690789052 | Maintenance |
| 10 | 8-61128 | 2024-10-20 | 63.354073 | -89.614824 | 88.80437027786628 | 77.2962791357376 | Maintenance |
| 11 | 5VJ H63 | 2024-10-24 | 40.1474815 | 136.426282 | 101.5492684098903 | 42.885471412934095 | Moving |
| 12 | 19-97635 | 2024-10-09 | -89.2243485 | 167.505332 | 118.88832311778857 | 24.393415155877996 | Moving |
| 13 | MOE 522 | 2024-10-20 | -84.5260325 | 92.100452 | 60.12912084967094 | 21.537678427270592 | Moving |
| 14 | 232-VJP | 2024-10-16 | 71.9753035 | -58.197939 | 41.07138031909747 | 35.238085691488685 | Stopped |
| 15 | 52YP 84 | 2024-10-04 | 87.4077215 | 87.009206 | 9.173530702554938 | 46.582386401100806 | Stopped |
| 16 | 12X BU2 | 2024-10-03 | 47.500484 | 54.704669 | 27.35538031978411 | 24.195231952477236 | Stopped |
| 17 | 7CR 626 | 2024-10-14 | -25.7912955 | -22.956152 | 12.867188760176692 | 0.21736879598361902 | Stopped |
| 18 | L63-96A | 2024-10-18 | -8.006049 | -5.949581 | 103.62610591052675 | 71.15389709313017 | Moving |
| 19 | 3-J9322 | 2024-10-03 | 11.501411 | -43.657201 | 94.05899239614277 | 56.68980405683101 | Moving |
| 20 | Q70 IQY | 2024-10-13 | 17.5476115 | 46.462619 | 90.35082441557364 | 46.717321901710406 | Moving |

Total rows: 100 of 100 Query complete 00:00:00.630 Ln 1, Col 1

Рис. 12

5.4. Загрузка 100000 записей в PostgreSQL (Рис. 13).

```
14]: # Подключение к PostgreSQL
pg_conn_params = {
    "dbname": "vehicle",
    "user": "postgres",
    "password": "changeme",
    "host": "localhost",
    "port": "5432"
}
pg_conn = check_postgres_connection(pg_conn_params)
if pg_conn:
    try:
        # Создание таблицы
        with pg_conn.cursor() as cur:
            cur.execute("""
                CREATE TABLE IF NOT EXISTS vehicle_100000(
                    vehicle_id VARCHAR(10),
                    timestamp DATE,
                    latitude FLOAT,
                    longitude FLOAT,
                    speed FLOAT,
                    fuel_level FLOAT,
                    status VARCHAR(30)
                )
            """)
        # Загрузка данных
        with pg_conn.cursor() as cur:
            for _, row in vehicle_df_100000.iterrows():
                cur.execute("""
                    INSERT INTO vehicle_100000 (vehicle_id, timestamp, latitude, longitude, speed, fuel_level, status)
                    VALUES (%s, %s, %s, %s, %s, %s, %s)
                    """, (row['vehicle_id'], row['timestamp'], row['latitude'], row['longitude'],
                        row['speed'], row['fuel_level'], row['status']))

        pg_conn.commit()
        print("Данные загружены в PostgreSQL")

    except Exception as e:
        print(f"Ошибка при работе с PostgreSQL: {e}")
    finally:
        pg_conn.close()
else:
    print("Пропуск операций с PostgreSQL из-за ошибки подключения")
```

Успешное подключение к PostgreSQL
Данные загружены в PostgreSQL

Рис. 13

5.5. Проверка данных в pgadmin (Рис. 14).

The screenshot shows the pgAdmin interface. At the top, the connection is 'public.vehicle_100000/vehicle/postgres@makarova_e'. Below the toolbar, the 'Query' tab is active, showing a SQL query: `SELECT * FROM public.vehicle_100000`. The 'Data Output' tab is also visible, showing a table with 20 rows and 8 columns. The columns are: vehicle_id, timestamp, latitude, longitude, speed, fuel_level, and status. The status column has values like 'Stopped', 'Maintenance', and 'Moving'. The bottom status bar indicates 'Total rows: 1000 of 100000' and 'Query complete 00:00:01.136'.

| | vehicle_id character varying (10) | timestamp date | latitude double precision | longitude double precision | speed double precision | fuel_level double precision | status character varying (30) |
|----|--------------------------------------|-------------------|------------------------------|-------------------------------|---------------------------|--------------------------------|----------------------------------|
| 1 | N 516314 | 2024-10-12 | -71.196244 | -106.739031 | 69.37489896015438 | 17.938573276706283 | Stopped |
| 2 | 219-KBQ | 2024-10-19 | -38.201303 | 153.121184 | 104.52368550807378 | 86.23429004107417 | Maintenance |
| 3 | X 162608 | 2024-10-10 | -45.152203 | 97.154357 | 81.27744131303854 | 46.67552521636714 | Stopped |
| 4 | MJL 510 | 2024-10-12 | 49.7923935 | -173.221437 | 59.35738765977808 | 44.01825232843747 | Stopped |
| 5 | BGO-285 | 2024-10-09 | 52.500998 | -126.592039 | 0.44049130295893413 | 21.691130577567176 | Maintenance |
| 6 | 63K M85 | 2024-10-03 | 41.0880815 | 85.220542 | 2.6391589778482683 | 51.03759809950691 | Maintenance |
| 7 | 68W-239 | 2024-10-22 | 34.36714 | 98.659091 | 66.16886912565444 | 69.90483541931837 | Stopped |
| 8 | FK-1952 | 2024-10-12 | 74.522524 | -56.621083 | 72.16916211311953 | 6.77775857854196 | Maintenance |
| 9 | 572NEY | 2024-10-06 | -30.054938 | 108.179118 | 76.68435603132404 | 23.776898250354815 | Stopped |
| 10 | 2-32189P | 2024-10-18 | -42.653284 | 17.767893 | 41.36957314822717 | 34.92016830498158 | Moving |
| 11 | SFV-636 | 2024-10-20 | 68.070441 | -3.894063 | 97.97646442604855 | 68.36901981737947 | Stopped |
| 12 | LMO 099 | 2024-10-13 | 2.0162375 | 163.107706 | 74.61893761432206 | 63.30744625794811 | Moving |
| 13 | 0P JX829 | 2024-10-15 | -75.212475 | -57.557728 | 90.16168889932607 | 36.02774895466021 | Moving |
| 14 | HSP 158 | 2024-10-02 | 19.0175615 | -67.815257 | 119.07795816784213 | 21.660484230904597 | Maintenance |
| 15 | F03-20E | 2024-10-09 | 57.855141 | -123.855828 | 35.872618978521885 | 42.32125334142059 | Moving |
| 16 | 36-W324 | 2024-10-19 | -3.109192 | -39.367276 | 64.11236000531875 | 49.41029272050008 | Moving |
| 17 | RHN 222 | 2024-10-10 | 55.6857275 | -172.839881 | 1.2091236543232808 | 90.29952447547073 | Moving |
| 18 | KJV 596 | 2024-10-20 | -84.325235 | 32.877624 | 108.96405136999289 | 52.99358100206032 | Maintenance |
| 19 | 441 LDV | 2024-10-01 | 78.5568345 | 145.5143 | 115.60522750669016 | 15.76023867427031 | Stopped |
| 20 | HKZ 904 | 2024-10-22 | 19.089743 | 19.10152 | 96.38738753362221 | 86.16981413981625 | Maintenance |

Рис. 14

6. Оценка эффективности работы с геопространственными данными

При работе с геопространственными данными необходима чёткая структура данных для отслеживания транспортного средства, поэтому PostgreSQL является более подходящей СУБД для хранения данных.

Структура данных неизменна, при отслеживании транспортного средства входными данными являются – номер транспортного средства, дата и время, долгота, широта, уровень топлива и статус движения.

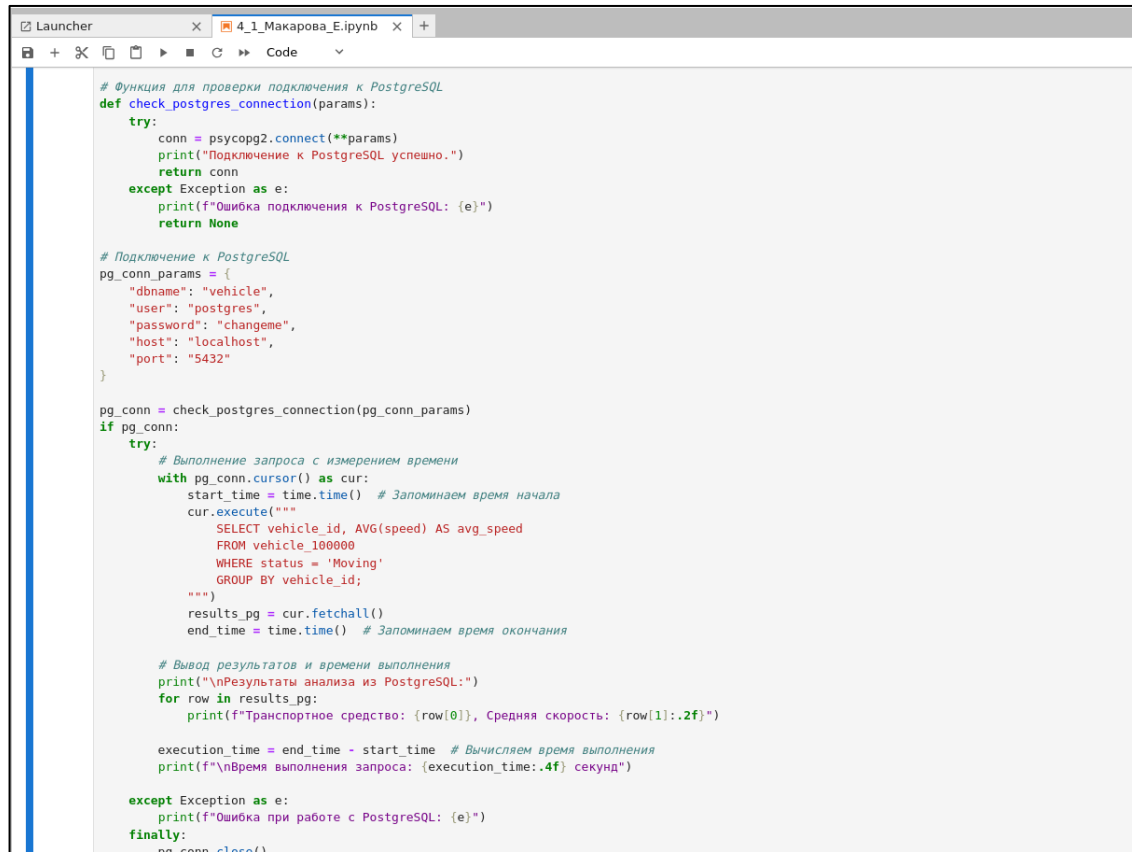
6.1. Выполнение запроса к таблице с 100 строками в PostgreSQL с измерением времени

На рисунке 15 отображён листинг кода с запросом на измерение средней скорости транспортного средства

```
4_1 Макарова Е.А.  
File Edit View Run Kernel Tabs Settings Help  
+ [Icons] Code  
Подключение успешно  
[90]: import psycopg2  
import pandas as pd  
import time # Импортируем модуль time  
  
# Функция для проверки подключения к PostgreSQL  
def check_postgres_connection(params):  
    try:  
        conn = psycopg2.connect(**params)  
        print("Подключение к PostgreSQL успешно.")  
        return conn  
    except Exception as e:  
        print(f"Ошибка подключения к PostgreSQL: {e}")  
        return None  
  
# Подключение к PostgreSQL  
pg_conn_params = {  
    "dbname": "vehicle",  
    "user": "postgres",  
    "password": "changeme",  
    "host": "localhost",  
    "port": "5432"  
}  
  
pg_conn = check_postgres_connection(pg_conn_params)  
if pg_conn:  
    try:  
        # Выполнение запроса с измерением времени  
        with pg_conn.cursor() as cur:  
            start_time = time.time() # Запоминаем время начала  
            cur.execute("""  
                SELECT vehicle_id, AVG(speed) AS avg_speed  
                FROM vehicle_100  
                WHERE status = 'Moving'  
                GROUP BY vehicle_id;  
            """)  
            results_pg = cur.fetchall()  
            end_time = time.time() # Запоминаем время окончания  
  
            # Вывод результатов и времени выполнения  
            print("\nРезультаты анализа из PostgreSQL:")  
            for row in results_pg:  
                print(f"Транспортное средство: {row[0]}, Средняя скорость: {row[1]:.2f}")  
  
            execution_time = end_time - start_time # Вычисляем время выполнения  
            print(f"\nВремя выполнения запроса: {execution_time:.4f} секунд")  
  
    except Exception as e:  
        print(f"Ошибка при работе с PostgreSQL: {e}")  
    finally:  
        pg_conn.close()  
else:  
    print("Пропуск операций с PostgreSQL из-за ошибки подключения")
```

6.2. Выполнение запроса в таблице с 100000 строками в PostgreSQL на измерение средней скорости транспортного средства

Листинг кода выполнения запроса на вычисление средней скорости по транспортному средству (Рис. 17).



```
# Функция для проверки подключения к PostgreSQL
def check_postgres_connection(params):
    try:
        conn = psycopg2.connect(**params)
        print("Подключение к PostgreSQL успешно.")
        return conn
    except Exception as e:
        print(f"Ошибка подключения к PostgreSQL: {e}")
        return None

# Подключение к PostgreSQL
pg_conn_params = {
    "dbname": "vehicle",
    "user": "postgres",
    "password": "changeme",
    "host": "localhost",
    "port": "5432"
}

pg_conn = check_postgres_connection(pg_conn_params)
if pg_conn:
    try:
        # Выполнение запроса с измерением времени
        with pg_conn.cursor() as cur:
            start_time = time.time() # Запоминаем время начала
            cur.execute("""
                SELECT vehicle_id, AVG(speed) AS avg_speed
                FROM vehicle_100000
                WHERE status = 'Moving'
                GROUP BY vehicle_id;
            """)
            results_pg = cur.fetchall()
            end_time = time.time() # Запоминаем время окончания

            # Вывод результатов и времени выполнения
            print("\nРезультаты анализа из PostgreSQL:")
            for row in results_pg:
                print(f"Транспортное средство: {row[0]}, Средняя скорость: {row[1]:.2f}")

            execution_time = end_time - start_time # Вычисляем время выполнения
            print(f"\nВремя выполнения запроса: {execution_time:.4f} секунд")

    except Exception as e:
        print(f"Ошибка при работе с PostgreSQL: {e}")
    finally:
        pg_conn.close()
```

```
her x 4_1_Макарова_E.ipynb x +
Code
# вывод результатов и времени выполнения
print("\nРезультаты анализа из PostgreSQL:")
for row in results_pg:
    print(f"Транспортное средство: {row[0]}, Средняя скорость: {row[1]:.2f}")

execution_time = end_time - start_time # Вычисляем время выполнения
print(f"\nВремя выполнения запроса: {execution_time:.4f} секунд")

except Exception as e:
    print(f"Ошибка при работе с PostgreSQL: {e}")
finally:
    pg_conn.close()
else:
    print("Пропуск операций с PostgreSQL из-за ошибки подключения")

Подключение к PostgreSQL успешно.

Результаты анализа из PostgreSQL:
Транспортное средство: XWI-1194, Средняя скорость: 108.48
Транспортное средство: 4XSH121, Средняя скорость: 19.82
Транспортное средство: MWO 768, Средняя скорость: 0.37
Транспортное средство: MQE 981, Средняя скорость: 98.41
Транспортное средство: 77-41693, Средняя скорость: 110.00
Транспортное средство: 110453, Средняя скорость: 13.63
Транспортное средство: 75 69540, Средняя скорость: 109.16
Транспортное средство: SGN6443, Средняя скорость: 28.07
Транспортное средство: L55-RAF, Средняя скорость: 45.37
Транспортное средство: 2KZ 454, Средняя скорость: 109.83
Транспортное средство: 832 TQD, Средняя скорость: 116.87
Транспортное средство: N83-38A, Средняя скорость: 42.78
Транспортное средство: 459YHA, Средняя скорость: 21.65
Транспортное средство: 731-278, Средняя скорость: 32.27
Транспортное средство: 978 AYT, Средняя скорость: 12.06
Транспортное средство: 677 XEE, Средняя скорость: 92.13
Транспортное средство: 4-1371Z, Средняя скорость: 77.42
Транспортное средство: TOY 030, Средняя скорость: 73.11
Транспортное средство: 957 LST, Средняя скорость: 100.55
Транспортное средство: 8662, Средняя скорость: 3.87
Транспортное средство: 82-4361Z, Средняя скорость: 24.08
Транспортное средство: 740 5179, Средняя скорость: 69.43
Транспортное средство: V 756846, Средняя скорость: 0.87
Транспортное средство: BJN-596, Средняя скорость: 85.89
Транспортное средство: 824-VAD, Средняя скорость: 1.60
Транспортное средство: 484 MBT, Средняя скорость: 84.86
Транспортное средство: 901-GZF, Средняя скорость: 42.33
Транспортное средство: TEL-0194, Средняя скорость: 21.06
Транспортное средство: 526-GGG, Средняя скорость: 72.21
Транспортное средство: 013DFI, Средняя скорость: 66.51
Транспортное средство: 539-YDO, Средняя скорость: 17.23
Транспортное средство: 479-866, Средняя скорость: 88.35
```

Рис. 17

Результат выполнения запроса, а также скорость выполнения указаны на

Рис. 18

```
her x 4_1_Макарова_E.ipynb x +
Code
Транспортное средство: 80-44000, Средняя скорость: 30.27
Транспортное средство: MNM-548, Средняя скорость: 33.79
Транспортное средство: 69-05011, Средняя скорость: 29.44
Транспортное средство: H40 7VK, Средняя скорость: 103.48
Транспортное средство: LJV 611, Средняя скорость: 98.01
Транспортное средство: 0VF 143, Средняя скорость: 64.74
Транспортное средство: 9V0 792, Средняя скорость: 66.60
Транспортное средство: 05A*745, Средняя скорость: 97.84
Транспортное средство: 184-TELM, Средняя скорость: 50.60
Транспортное средство: GAT9997, Средняя скорость: 61.51
Транспортное средство: 461 YVY, Средняя скорость: 101.06
Транспортное средство: 3G QT597, Средняя скорость: 24.99
Транспортное средство: 146 JBI, Средняя скорость: 53.92
Транспортное средство: PJJ-0182, Средняя скорость: 105.35
Транспортное средство: 627XKP, Средняя скорость: 11.79
Транспортное средство: 092QKI, Средняя скорость: 75.10
Транспортное средство: 634 3NV, Средняя скорость: 80.33
Транспортное средство: 919 CKB, Средняя скорость: 37.90
Транспортное средство: 58-I776, Средняя скорость: 3.64

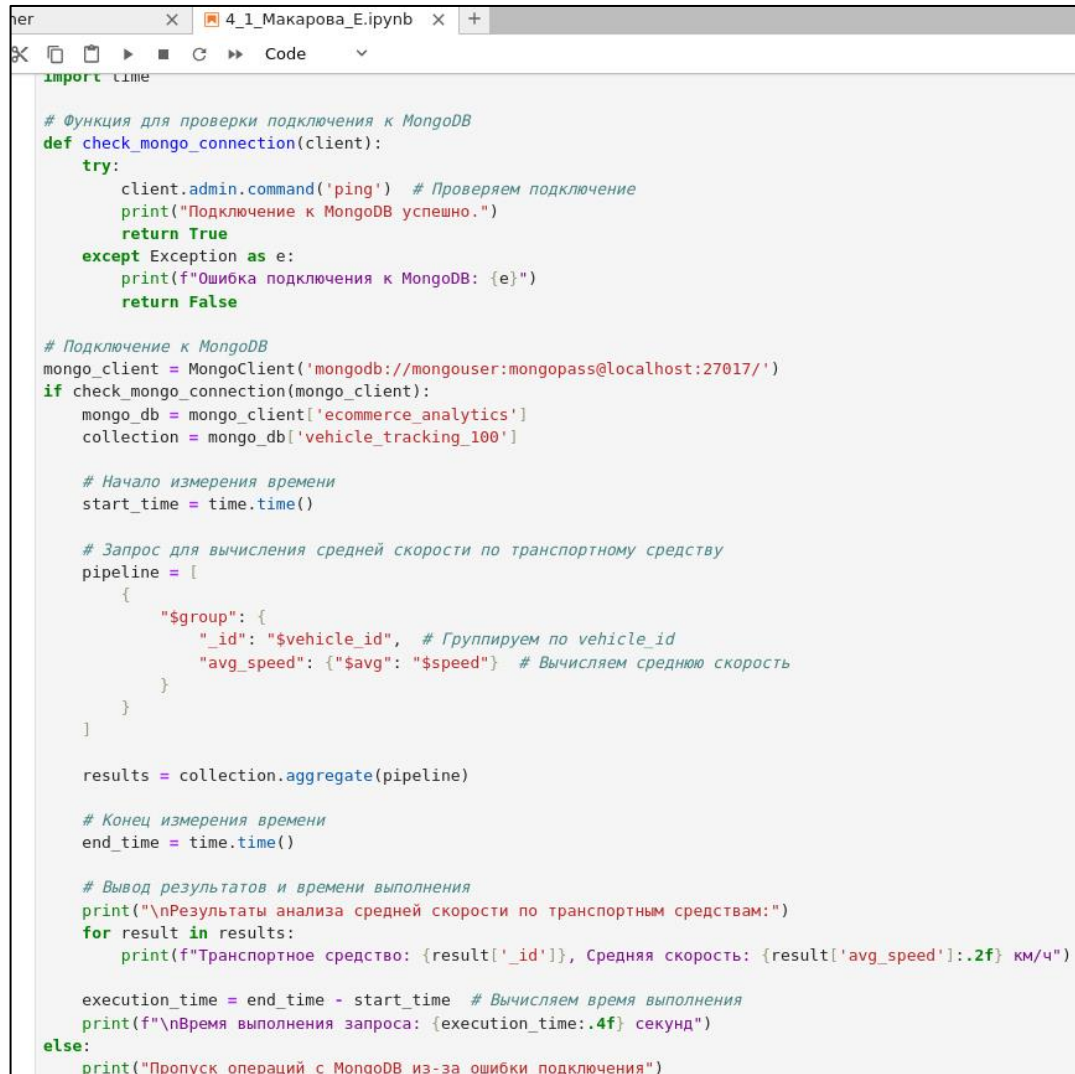
Время выполнения запроса: 0.4153 секунд
```

Рис. 18

Скорость выполнения запроса к 100000 строк в PostgreSQL составила 0.415 секунд.

6.3. Выполнение запроса на вычисление средней скорости по транспортному средству к коллекции с 100 документами в MongoDB

Листинг кода на выполнение запроса с агрегацией среднего значения скорости по транспортному средству представлен на Рис. 19



```
her X 4_1_Макарова_E.ipynb X +
Code
import time

# Функция для проверки подключения к MongoDB
def check_mongo_connection(client):
    try:
        client.admin.command('ping') # Проверяем подключение
        print("Подключение к MongoDB успешно.")
        return True
    except Exception as e:
        print(f"Ошибка подключения к MongoDB: {e}")
        return False

# Подключение к MongoDB
mongo_client = MongoClient('mongodb://mongouser:mongopass@localhost:27017/')
if check_mongo_connection(mongo_client):
    mongo_db = mongo_client['ecommerce_analytics']
    collection = mongo_db['vehicle_tracking_100']

    # Начало измерения времени
    start_time = time.time()

    # Запрос для вычисления средней скорости по транспортному средству
    pipeline = [
        {
            "$group": {
                "_id": "$vehicle_id", # Группируем по vehicle_id
                "avg_speed": {"$avg": "$speed"} # Вычисляем среднюю скорость
            }
        }
    ]

    results = collection.aggregate(pipeline)

    # Конец измерения времени
    end_time = time.time()

    # Вывод результатов и времени выполнения
    print("\nРезультаты анализа средней скорости по транспортным средствам:")
    for result in results:
        print(f"Транспортное средство: {result['_id']}, Средняя скорость: {result['avg_speed']:.2f} км/ч")

    execution_time = end_time - start_time # Вычисляем время выполнения
    print(f"\nВремя выполнения запроса: {execution_time:.4f} секунд")
else:
    print("Пропуск операций с MongoDB из-за ошибки подключения")
```

Рис. 19

Результат выполнения запроса с указанием времени выполнения представлен на Рис. 20

```
Подключение к MongoDB успешно.

Результаты анализа средней скорости по транспортным средствам:
Транспортное средство: LJY 454, Средняя скорость: 49.98 км/ч
Транспортное средство: P96-34U, Средняя скорость: 65.14 км/ч
Транспортное средство: 8-K4566, Средняя скорость: 25.16 км/ч
Транспортное средство: 875 Z80, Средняя скорость: 80.90 км/ч
Транспортное средство: 20-RX99, Средняя скорость: 3.29 км/ч
Транспортное средство: 540-335, Средняя скорость: 67.18 км/ч
Транспортное средство: 3ZJ L02, Средняя скорость: 67.66 км/ч
Транспортное средство: 34-I387, Средняя скорость: 77.73 км/ч
Транспортное средство: 8VA4287, Средняя скорость: 91.42 км/ч
Транспортное средство: 0R 5251F, Средняя скорость: 31.78 км/ч
Транспортное средство: RHS-4179, Средняя скорость: 30.80 км/ч
Транспортное средство: 2SR 591, Средняя скорость: 98.42 км/ч
Транспортное средство: 40V AL8, Средняя скорость: 109.80 км/ч
Транспортное средство: 2P 59587, Средняя скорость: 46.89 км/ч
Транспортное средство: 3IR0653, Средняя скорость: 77.24 км/ч
Транспортное средство: 207-YCGU, Средняя скорость: 71.27 км/ч
Транспортное средство: 691-CYT, Средняя скорость: 106.26 км/ч
Транспортное средство: M0E 522, Средняя скорость: 60.13 км/ч
Транспортное средство: 742149, Средняя скорость: 100.12 км/ч
Транспортное средство: VYM 661, Средняя скорость: 73.71 км/ч
Транспортное средство: OXM 614, Средняя скорость: 17.07 км/ч
Транспортное средство: 9G 15497, Средняя скорость: 116.30 км/ч
Транспортное средство: 861 2254, Средняя скорость: 52.71 км/ч
Транспортное средство: 232HNG, Средняя скорость: 24.12 км/ч
Транспортное средство: 0281 IK, Средняя скорость: 49.44 км/ч
Транспортное средство: F 339611, Средняя скорость: 35.26 км/ч
Транспортное средство: 7CR 626, Средняя скорость: 12.87 км/ч
Транспортное средство: 070 10Y, Средняя скорость: 90.35 км/ч
Транспортное средство: 935 RDV, Средняя скорость: 30.75 км/ч
Транспортное средство: 8Z V3725, Средняя скорость: 71.81 км/ч
Транспортное средство: 8TA 063, Средняя скорость: 4.76 км/ч
Транспортное средство: KYX 120, Средняя скорость: 24.97 км/ч
Транспортное средство: 0977 NN, Средняя скорость: 12.00 км/ч
Транспортное средство: OZ 49135, Средняя скорость: 57.17 км/ч
```

```
her x 4_1_Макарова_E.ipynb x +
X [ ] Code
Транспортное средство: 3GP3027, Средняя скорость: 1.49 км/ч
Транспортное средство: 8-61128, Средняя скорость: 88.80 км/ч
Транспортное средство: 19-97635, Средняя скорость: 118.89 км/ч
Транспортное средство: 588 D0C, Средняя скорость: 109.88 км/ч
Транспортное средство: 5-Q1087, Средняя скорость: 40.86 км/ч
Транспортное средство: 170 1JY, Средняя скорость: 81.39 км/ч
Транспортное средство: 6NU14, Средняя скорость: 105.41 км/ч
Транспортное средство: 0QG-9839, Средняя скорость: 110.75 км/ч
Транспортное средство: 027-19P, Средняя скорость: 34.54 км/ч
Транспортное средство: HBV 125, Средняя скорость: 74.60 км/ч
Транспортное средство: 946-MTQ, Средняя скорость: 25.06 км/ч
Транспортное средство: MNT 468, Средняя скорость: 62.85 км/ч
Транспортное средство: 24-FZ43, Средняя скорость: 54.87 км/ч
Транспортное средство: 826-IMV, Средняя скорость: 28.52 км/ч
Транспортное средство: 12X BU2, Средняя скорость: 27.36 км/ч
Транспортное средство: A72 3FY, Средняя скорость: 69.48 км/ч
Транспортное средство: 7QH04, Средняя скорость: 55.72 км/ч
Транспортное средство: 826 0246, Средняя скорость: 106.37 км/ч
Транспортное средство: 24VQ2, Средняя скорость: 34.33 км/ч
Транспортное средство: 30E 829, Средняя скорость: 86.24 км/ч
Транспортное средство: IH3 6906, Средняя скорость: 4.40 км/ч
Транспортное средство: 396 9428, Средняя скорость: 78.37 км/ч
Транспортное средство: 4FN A92, Средняя скорость: 105.35 км/ч

Время выполнения запроса: 0.0515 секунд
```

Рис. 20

Время запроса составило 0.515 секунд.

6.4. Выполнение запроса на вычисление средней скорости транспортного средства с вычисление времени в MongoDB к коллекции с 100000 документами

Листинг кода на выполнение запроса к коллекции с 100000 документами представлен на Рис. 21

```
[10]: from pymongo import MongoClient
import time

def check_mongo_connection(client):
    try:
        client.admin.command('ping')
        print("Successful connection")
        return True
    except Exception as e:
        print(f"Error connection for MongoDB: {e}")
        return False

mongo_client = MongoClient('mongodb://mongouser:mongopass@localhost:27017/')
if check_mongo_connection(mongo_client):
    mongo_db = mongo_client['ecommerce_analytics']
    collection = mongo_db['vehicle_tracking_100000']

    start_time = time.time()

    |
    pipeline = [
        {
            "$group": {
                "_id": "$vehicle_id",
                "avg_speed": {"$avg": "$speed"}
            }
        }
    ]

    results = collection.aggregate(pipeline)
    end_time = time.time()

    print("\nResult:")
    for result in results:
        print(f"Vehicle: {result['_id']}, AVG_speed: {result['avg_speed']:.2f}")

    execution_time = end_time - start_time
    print(f"\nQuery execution time: {execution_time:.4f} seconds")
```

Рис. 21

Результат выполнения запроса представлен на Рис. 22.

```
Successful connection

Result:
Vehicle: 202 YYP, AVG_speed: 85.80
Vehicle: 923 AAX, AVG_speed: 56.08
Vehicle: 865 6197, AVG_speed: 112.72
Vehicle: 861-IHL, AVG_speed: 64.34
Vehicle: 297 7EH, AVG_speed: 49.11
Vehicle: 725-BTI, AVG_speed: 50.78
Vehicle: HJ1 3036, AVG_speed: 104.28
Vehicle: ZHA 453, AVG_speed: 113.55
Vehicle: RF 7925, AVG_speed: 115.64
Vehicle: GAQ-1456, AVG_speed: 73.85
Vehicle: 7H 5553A, AVG_speed: 80.42
Vehicle: 676G0G, AVG_speed: 103.17
Vehicle: HEO 593, AVG_speed: 25.39
Vehicle: 1-83398T, AVG_speed: 83.14
Vehicle: WPH 288, AVG_speed: 19.23
Vehicle: WYC-0627, AVG_speed: 47.10
Vehicle: 91-X543, AVG_speed: 108.55
Vehicle: 7Z 81877, AVG_speed: 106.52
Vehicle: DKQ5485, AVG_speed: 102.14
Vehicle: ZOI 377, AVG_speed: 116.76
Vehicle: 6QA5499, AVG_speed: 104.47
Vehicle: 628TEU, AVG_speed: 116.23
Vehicle: 5NH3124, AVG_speed: 41.53
Vehicle: 888 D0F, AVG_speed: 6.62
Vehicle: KWV 530, AVG_speed: 31.23
Vehicle: JFZ 9494, AVG_speed: 78.76
Vehicle: 84B 5453, AVG_speed: 15.56
Vehicle: W06-GAQ, AVG_speed: 70.85
Vehicle: 094-UPX, AVG_speed: 23.19
Vehicle: 021 PDM, AVG_speed: 8.59
Vehicle: 7KE D05, AVG_speed: 118.50
Vehicle: 10FK259, AVG_speed: 3.06
Vehicle: 592 3PI, AVG_speed: 64.36
Vehicle: 5P G9502, AVG_speed: 109.79
Vehicle: 2TD L40, AVG_speed: 13.67
Vehicle: ARO-0055, AVG_speed: 115.07
Vehicle: 688MRJ, AVG_speed: 116.89
```

Рис. 22

Время выполнения запроса указан на Рис. 23.

```

Vehicle: AOB 602, AVG_speed: 57.20
Vehicle: VHV 611, AVG_speed: 48.86
Vehicle: OMM-129, AVG_speed: 72.20
Vehicle: 270 BND, AVG_speed: 105.20
Vehicle: 29-V146, AVG_speed: 108.61
Vehicle: 5AL V19, AVG_speed: 51.24
Vehicle: XXC 970, AVG_speed: 100.04

Query execution time: 1.4808 seconds

```

Рис. 23

Время выполнения запроса на расчёт средней скорости по транспортному средства 100000 документов в MongoDB составило 1.4808.

7. Результаты экспериментов указаны в таблице 1.

Таблица 1 – Результаты выполнения экспериментов

| № | Эксперимент | Время выполнения запроса, сек |
|---|--|-------------------------------|
| 1 | Запрос на вычисление средней скорости транспортного средства (100 строк) в PostgreSQL | 0.0586 |
| 2 | Запрос на вычисление средней скорости транспортного средства (100000 строк) в PostgreSQL | 0.415 |
| 3 | Запрос на вычисление средней скорости транспортного средства (100 строк) в MongoDB | 0.515 |
| 4 | Запрос на вычисление средней скорости транспортного средства (100000 строк) в MongoDB | 1.4808 |

8. Вывод

Исходя из результатов экспериментов можно сделать вывод, что выполнение запросов к PostgreSQL на агрегацию значений более быстрое и эффективное.

При работе с геопространственными данными необходима чёткая структура данных для отслеживания транспортного средства, поэтому PostgreSQL является более подходящей СУБД для хранения данных.