МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕОБРАЗОВАТЕЛЬНОЕУЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

**(ФГБОУ ВО «ВГУ»)**

Прикладной математики, информатики и механики

Кафедра Системного анализа и управления

Отчет по **учебной практике, ознакомительной**

# < Изучение открытого программного обеспечения – Apache airflow.>

Направление 01.03.02 Прикладная математика и информатика

Профиль подготовки «Динамические системы и управление»

Зав.кафедрой _____д.ф.-м.н., проф. Задорожний В.Г.__.__.20__
*Подпись, расшифровка, ученая степень, звание*

Обучающийся студент 1 курса 2 группы Марных Никита Владимирович_.__.20__
*Подпись, расшифровка подписи*

Руководитель практики от предприятия Surf _____ Батищева Е.Е._.__.20__

*Подпись, расшифровка подписи*

Руководитель практики от ВГУ_____к.ф.-м.н., доцент Коструб И.Д._.__.20__

*Подпись, расшифровка подписи, ученая степень, звание*

Воронеж 2023

# Содержание

# Введение

Цель прохождения учебной практики состоит в изучении открытого программного обеспечения для создания, выполнения, мониторинга и оркестровки потоков операций по обработке данных – Apache Airflow.

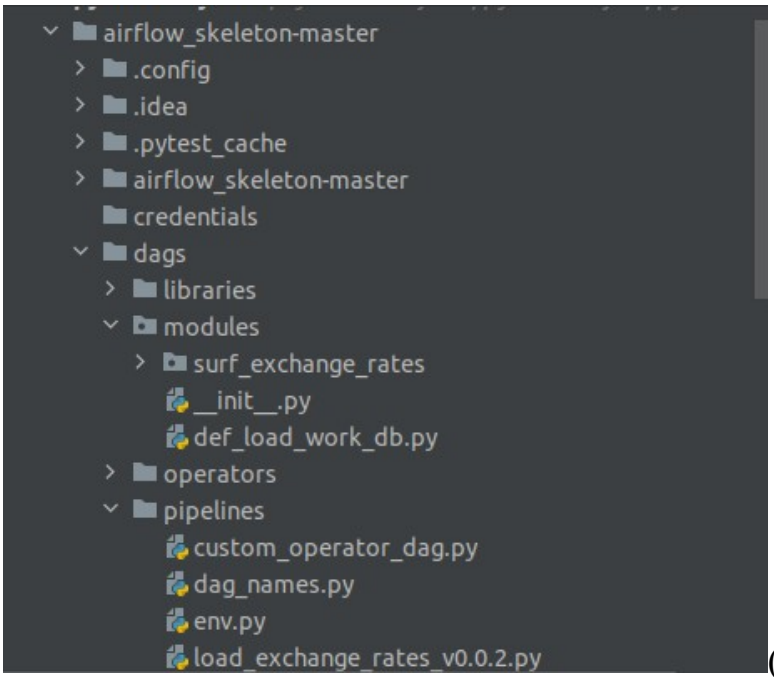
Для достижения данной цели потребовалось решить следующие задачи:

1. Изучение теории для работы с Python-библиотеками: airflow, requests, pandas, sqlalchemy, datetime, Docker
2. Выполнить следующие практические задания:
    2.1. Установить Docker
    2.2. Установить Pycharm и библиотеки airflow, requests, pandas, sqlalchemy, datetime
    2.3. Запустить Airflow
    2.4. Необходимо разработать и реализовать DAG (направленный ациклический граф), который будет, один раз в день, загружать в базу данных таблицу с информацией о рабочих часах в рабочие и праздничные дни за текущий год

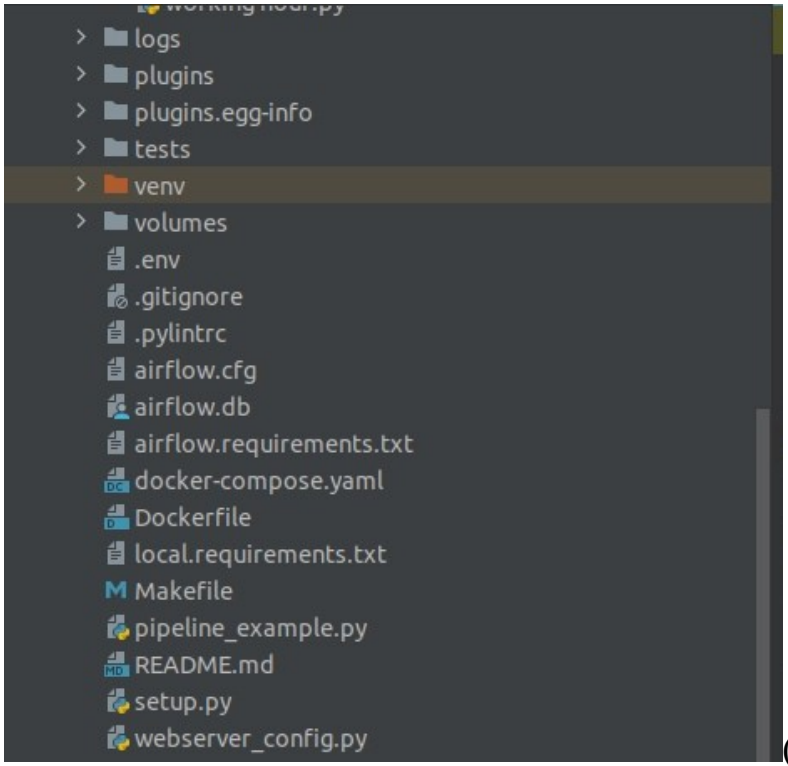
# 1. Описание теоретических и практических аспектов выполненной работы

## *1.1.Примечания*

Вид директории PythonProject в Pycharm



(рис.1)



(рис.2)

**sudo apt update –** обновляет информацию о пакетах.

**sudo apt install –** устанавливает программу на выбор.

**Curl** - это инструмент командной строки для передачи данных на сервер или с сервера.

**sudo apt install docker-ce docker-ce-cli containerd.io –** установка пакета Docker.

**pip install apache-airflow** установка библиотеки Apache Airflow.

**pip install requests** установка библиотеки requests.

**pip install pandas** установка библиотеки pandas.

**pip install sqlalchemy** установка библиотеки sqlalchemy.

**pip install datetime** установка библиотеки datetime.

**def_load_work_db –** содержит функцию load_work_db.

**working_hours –** содержит DAG.

**sudo docker ps** – просмотр запущенных контейнеров.

**datetime.now().year –** получение текущего года.

**datetime.now() –** получение текущего времени.

**DataFrame –** это табличная структура данных.

**to_sql –** записать объект в базу данных.

# 2. Установка Docker и библиотек. Запуск Airflow. Разработка и реализация программы.

## *2.1. Установка Docker*

Для начала работы необходимо открыть терминал для Linux Ubuntu 22.04. Убедимся, что все пакеты системы находятся в актуальном состоянии, выполнив команду: **sudo apt update**. Узнаем, установлен ли пакет “apt-transport-https”, необходимый для подключения репозитория Docker, с помощью команды: **sudo apt install apt-transport-https**. Далее добавим GPG-ключ Docker в систему с помощью команды: **curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg**. Добавим официальный репозиторий Docker в список источников пакетов “aptc” помощью команды: **echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu focal stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null**. Обновим список пакетов, чтобы включить данные из добавленного репозитория Docker с помощью команды: **sudo apt update**. Далее установим пакет Docker с помощью следующей команды: **sudo apt install docker-ce docker-ce-cli containerd.io**.

## *2.2. Установка PyCharm и необходимых библиотек*

Необходимо загрузить установочный пакет PyCharm с официального сайта JetBrains и установить его. Теперь открываем PyCharm и создаём проект. В папке с проектом открываем терминал и устанавливаем библиотеки с помощью команд:

**pip install apache-airflow** используется для установки библиотеки Apache Airflow.

**pip install requests** используется для установки библиотеки requests.

**pip install pandas** используется для установки библиотеки pandas.

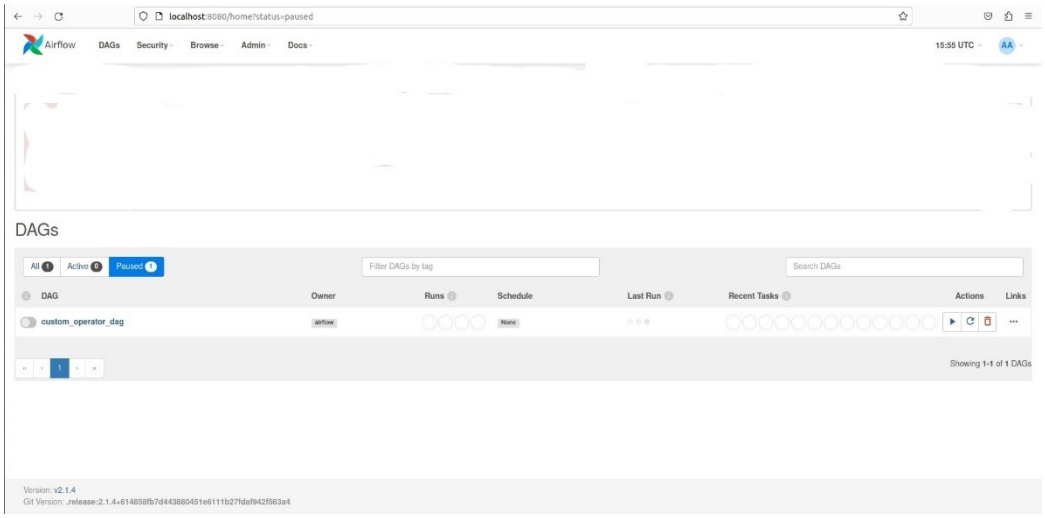
**pip install sqlalchemy** используется для установки библиотеки sqlalchemy.

**pip install datetime** используется для установки библиотеки datetime.

## *2.3. Запуск Airflow*

Необходимо скачать с сайта компании репозиторий по ссылке https://github.com/kenigteh/airflow_skeleton/tree/master.

Далее нужно распаковать zip-архив “airflow_skeleton-master.zip” и перенести в папку с проектом. Теперь открываем Pycharm и в терминале пишем команду: make start-airflow, после запуска заходим в браузер и переходим по ссылке http://localhost:8080.



(рис.3)

**sudo docker ps** – используется для просмотра запущенные контейнеров.

## *2.4. Разработка и реализация DAG*

Для масштабируемости кода разделим его на два файла: **def_load_work_db** и **working_hours**.

**def_load_work_db** – содержит функцию load_work_db, которая создает подключение к базе данных PostgreSQL с использованием заданных переменных DB_USER (имя пользователя), DB_PASSWORD (пароль), DB_HOST (хост), DB_PORT (порт) и DB_NAME (имя базы данных):

```
engine = create_engine(f"postgresql://{DB_USER}:{DB_PASSWORD}@{DB_HOST}:{DB_PORT}/{DB_NAME}")
```
(рис.4)

Получает данные о рабочих и праздничных днях с помощью запроса к API https://isdayoff.ru/api/getdata?year={datetime.now().year}&pre=1. Запрос формируется с использованием текущего года, полученного с помощью функции **datetime.now().year.** Результат запроса сохраняется в переменную working_status в виде списка строк:

```
['1', '1', '1', '1', '1', '1', '1', '1', '0', '0', '0', '0', '0', '1', '1', '0', '0', '0', '0', '0',
```
(рис.5)

**0** - рабочий день

**1** – нерабочий день

**2** – сокращённый рабочий день

Цикл for в функции заменяет значения в списке working_status на нужные значения для работы с базой данных. Если значение равно "0", то заменяется на значение 8 (количество рабочих часов), если значение равно "1", то заменяется на значение 0 (количество рабочих часов в праздничные дни), а если значение отличается от "0" и "1", то заменяется на значение 7 (количество часов в сокращённые дни):

```
for index in range(len(working_status)):
        if working_status[index] == "0":
            working_status[index] = 8
        elif working_status[index] == "1":
            working_status[index] = 0
        else:
            working_status[index] = 7
```

(рис.3)

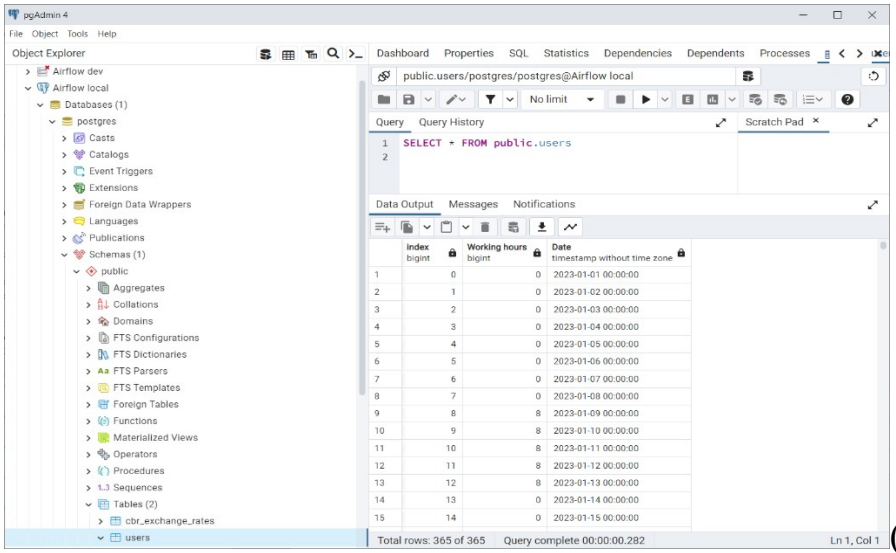
Создает объект **DataFrame** - Data, который состоит из двух столбцов: "Working hours" (количество рабочих часов) и "Date" (дата). Для формирования дат используется функция **date_range** с указанием начальной и конечной даты текущего года:

```
     Working hours       Date
0                0 2023-01-01
1                0 2023-01-02
2                0 2023-01-03
3                0 2023-01-04
4                0 2023-01-05
..             ...        ...
360              8 2023-12-27
361              8 2023-12-28
362              8 2023-12-29
363              0 2023-12-30
364              0 2023-12-31

[365 rows x 2 columns]
```

(рис.4)

Записывает данные из Data в таблицу "working hours" базы данных с использованием метода **to_sql**:



(рис.6)

Файл **working_hours** содержит кодб который создает DAG (Directed Acyclic Graph) для выполнения задачи "working_hours". DAG представляет собой последовательность задач, которые должны быть выполнены в определенном порядке. DAG "working_hours" имеет следующие характеристики:

Идентификатор DAG установлен как "working_hours".

Дата начала установлена на текущую дату и время с помощью **datetime.now().**

Установлен флаг catchup=False, что означает, что DAG не будет отслеживать и запускать пропущенные задачи.

Расписание выполнения задач установлено на ежедневное выполнение с помощью schedule_interval="@daily".

Затем, в DAG определен оператор **PythonOperator**. Этот оператор будет вызывать функцию **load_work_db** из модуля "def_load_work_db.py". **PythonOperator** предоставляет возможность выполнения программного кода Python внутри DAG. Он имеет следующие параметры:

task_id устанавливает идентификатор задачи.

python_callable указывает на вызываемую функцию.

dag устанавливает родительский DAG, в котором будет выполнен оператор.

Таким образом, данный код создает DAG "working_hours" и определяет оператор , который будет вызывать функцию **load_work_db** из модуля

"def_load_work_db.py". Задача "load_work_db" будет выполняться ежедневно в соответствии с расписанием, начиная с указанной даты.

## Заключение

В результате прохождения практики цель была достигнута, все задачи решены в полном объеме, теория и навыки работы с Apache Airflow приобретены.

# Список литературы

1. https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/dags.html
2. https://docs.docker.com/engine/install/ubuntu/
3. Documentation | Apache Airflow

```
from airflow import DAG
from airflow.operators.python import PythonOperator
from dags.modules.def_load_work_db import *



first_dag = DAG(
    dag_id="working_hours",
    start_date=datetime.now(),
    catchup=False,
    schedule_interval="@daily"
        )

load_db = PythonOperator(
    task_id="load_work_db",
    python_callable=load_work_db,
    dag=first_dag
    )
```

```
from requests import get
from pandas import DataFrame, date_range
from sqlalchemy import create_engine
from pipelines.env import *
from datetime import datetime


def load_work_db():
  engine=create_engine(f"postgresql://{DB_USER}:
{DB_PASSWORD}@{DB_HOST}:{DB_PORT}/{DB_NAME}")
   # Получаем данные о рабочих и праздничных днях
   working_status=list(str(get(f"https://isdayoff.ru/api/getdata?
year={datetime.now().year}&pre=1").json()))

   # Заменяем на нужные нам значения
   for index in range(len(working_status)):
      if working_status[index] == "0":
         working_status[index] = 8
      elif working_status[index] == "1":
         working_status[index] = 0
      else:
         working_status[index] = 7

   Data = DataFrame(list(zip(working_status,
date_range(start=f'1/1/{datetime.now().year}',
end=f'31/12/{datetime.now().year}'))), columns=["Working hours", "Date"])
   Data.to_sql('working hours', con=engine, if_exists='append')
```

| | |
|---|---|
| 1 | 1,0,2023-01-02 00:00:00 |
| 2 | 2,0,2023-01-03 00:00:00 |
| 3 | 3,0,2023-01-04 00:00:00 |
| 4 | 4,0,2023-01-05 00:00:00 |
| 5 | 5,0,2023-01-06 00:00:00 |
| 6 | 6,0,2023-01-07 00:00:00 |
| 7 | 7,0,2023-01-08 00:00:00 |
| 8 | 8,8,2023-01-09 00:00:00 |
| 9 | 9,8,2023-01-10 00:00:00 |
| 10 | 10,8,2023-01-11 00:00:00 |
| 11 | 11,8,2023-01-12 00:00:00 |
| 12 | 12,8,2023-01-13 00:00:00 |
| 13 | 13,0,2023-01-14 00:00:00 |
| 14 | 14,0,2023-01-15 00:00:00 |
| 15 | 15,8,2023-01-16 00:00:00 |
| 16 | 16,8,2023-01-17 00:00:00 |
| 17 | 17,8,2023-01-18 00:00:00 |
| 18 | 18,8,2023-01-19 00:00:00 |
| 19 | 19,8,2023-01-20 00:00:00 |
| 20 | 20,0,2023-01-21 00:00:00 |
| 21 | 21,0,2023-01-22 00:00:00 |
| 22 | 22,8,2023-01-23 00:00:00 |
| 23 | 23,8,2023-01-24 00:00:00 |
| 24 | 24,8,2023-01-25 00:00:00 |
| 25 | 25,8,2023-01-26 00:00:00 |
| 26 | 26,8,2023-01-27 00:00:00 |
| 27 | 27,0,2023-01-28 00:00:00 |
| 28 | 28,0,2023-01-29 00:00:00 |
| 29 | 29,8,2023-01-30 00:00:00 |
| 30 | 30,8,2023-01-31 00:00:00 |

| | |
|---|---|
| 31 | 31,8,2023-02-01 00:00:00 |
| 32 | 32,8,2023-02-02 00:00:00 |
| 33 | 33,8,2023-02-03 00:00:00 |
| 34 | 34,0,2023-02-04 00:00:00 |
| 35 | 35,0,2023-02-05 00:00:00 |
| 36 | 36,8,2023-02-06 00:00:00 |
| 37 | 37,8,2023-02-07 00:00:00 |
| 38 | 38,8,2023-02-08 00:00:00 |
| 39 | 39,8,2023-02-09 00:00:00 |
| 40 | 40,8,2023-02-10 00:00:00 |
| 41 | 41,0,2023-02-11 00:00:00 |
| 42 | 42,0,2023-02-12 00:00:00 |
| 43 | 43,8,2023-02-13 00:00:00 |
| 44 | 44,8,2023-02-14 00:00:00 |
| 45 | 45,8,2023-02-15 00:00:00 |
| 46 | 46,8,2023-02-16 00:00:00 |
| 47 | 47,8,2023-02-17 00:00:00 |
| 48 | 48,0,2023-02-18 00:00:00 |
| 49 | 49,0,2023-02-19 00:00:00 |
| 50 | 50,8,2023-02-20 00:00:00 |
| 51 | 51,8,2023-02-21 00:00:00 |
| 52 | 52,7,2023-02-22 00:00:00 |
| 53 | 53,0,2023-02-23 00:00:00 |
| 54 | 54,0,2023-02-24 00:00:00 |
| 55 | 55,0,2023-02-25 00:00:00 |
| 56 | 56,0,2023-02-26 00:00:00 |
| 57 | 57,8,2023-02-27 00:00:00 |
| 58 | 58,8,2023-02-28 00:00:00 |
| 59 | 59,8,2023-03-01 00:00:00 |
| 60 | 60,8,2023-03-02 00:00:00 |

| | |
|---|---|
| 31 | 31,8,2023-02-01 00:00:00 |
| 32 | 32,8,2023-02-02 00:00:00 |
| 33 | 33,8,2023-02-03 00:00:00 |
| 34 | 34,0,2023-02-04 00:00:00 |
| 35 | 35,0,2023-02-05 00:00:00 |
| 36 | 36,8,2023-02-06 00:00:00 |
| 37 | 37,8,2023-02-07 00:00:00 |
| 38 | 38,8,2023-02-08 00:00:00 |
| 39 | 39,8,2023-02-09 00:00:00 |
| 40 | 40,8,2023-02-10 00:00:00 |
| 41 | 41,0,2023-02-11 00:00:00 |
| 42 | 42,0,2023-02-12 00:00:00 |
| 43 | 43,8,2023-02-13 00:00:00 |
| 44 | 44,8,2023-02-14 00:00:00 |
| 45 | 45,8,2023-02-15 00:00:00 |
| 46 | 46,8,2023-02-16 00:00:00 |
| 47 | 47,8,2023-02-17 00:00:00 |
| 48 | 48,0,2023-02-18 00:00:00 |
| 49 | 49,0,2023-02-19 00:00:00 |
| 50 | 50,8,2023-02-20 00:00:00 |
| 51 | 51,8,2023-02-21 00:00:00 |
| 52 | 52,7,2023-02-22 00:00:00 |
| 53 | 53,0,2023-02-23 00:00:00 |
| 54 | 54,0,2023-02-24 00:00:00 |
| 55 | 55,0,2023-02-25 00:00:00 |
| 56 | 56,0,2023-02-26 00:00:00 |
| 57 | 57,8,2023-02-27 00:00:00 |
| 58 | 58,8,2023-02-28 00:00:00 |
| 59 | 59,8,2023-03-01 00:00:00 |
| 60 | 60,8,2023-03-02 00:00:00 |

| | |
|---|---|
| 91 | 91,0,2023-04-02 00:00:00 |
| 92 | 92,8,2023-04-03 00:00:00 |
| 93 | 93,8,2023-04-04 00:00:00 |
| 94 | 94,8,2023-04-05 00:00:00 |
| 95 | 95,8,2023-04-06 00:00:00 |
| 96 | 96,8,2023-04-07 00:00:00 |
| 97 | 97,0,2023-04-08 00:00:00 |
| 98 | 98,0,2023-04-09 00:00:00 |
| 99 | 99,8,2023-04-10 00:00:00 |
| 100 | 100,8,2023-04-11 00:00:00 |
| 101 | 101,8,2023-04-12 00:00:00 |
| 102 | 102,8,2023-04-13 00:00:00 |
| 103 | 103,8,2023-04-14 00:00:00 |
| 104 | 104,0,2023-04-15 00:00:00 |
| 105 | 105,0,2023-04-16 00:00:00 |
| 106 | 106,8,2023-04-17 00:00:00 |
| 107 | 107,8,2023-04-18 00:00:00 |
| 108 | 108,8,2023-04-19 00:00:00 |
| 109 | 109,8,2023-04-20 00:00:00 |
| 110 | 110,8,2023-04-21 00:00:00 |
| 111 | 111,0,2023-04-22 00:00:00 |
| 112 | 112,0,2023-04-23 00:00:00 |
| 113 | 113,8,2023-04-24 00:00:00 |
| 114 | 114,8,2023-04-25 00:00:00 |
| 115 | 115,8,2023-04-26 00:00:00 |
| 116 | 116,8,2023-04-27 00:00:00 |
| 117 | 117,8,2023-04-28 00:00:00 |
| 118 | 118,0,2023-04-29 00:00:00 |
| 119 | 119,0,2023-04-30 00:00:00 |

| | |
|---|---|
| 120 | 120,0,2023-05-01 00:00:00 |
| 121 | 121,8,2023-05-02 00:00:00 |
| 122 | 122,8,2023-05-03 00:00:00 |
| 123 | 123,8,2023-05-04 00:00:00 |
| 124 | 124,8,2023-05-05 00:00:00 |
| 125 | 125,0,2023-05-06 00:00:00 |
| 126 | 126,0,2023-05-07 00:00:00 |
| 127 | 127,0,2023-05-08 00:00:00 |
| 128 | 128,0,2023-05-09 00:00:00 |
| 129 | 129,8,2023-05-10 00:00:00 |
| 130 | 130,8,2023-05-11 00:00:00 |
| 131 | 131,8,2023-05-12 00:00:00 |
| 132 | 132,0,2023-05-13 00:00:00 |
| 133 | 133,0,2023-05-14 00:00:00 |
| 134 | 134,8,2023-05-15 00:00:00 |
| 135 | 135,8,2023-05-16 00:00:00 |
| 136 | 136,8,2023-05-17 00:00:00 |
| 137 | 137,8,2023-05-18 00:00:00 |
| 138 | 138,8,2023-05-19 00:00:00 |
| 139 | 139,0,2023-05-20 00:00:00 |
| 140 | 140,0,2023-05-21 00:00:00 |
| 141 | 141,8,2023-05-22 00:00:00 |
| 142 | 142,8,2023-05-23 00:00:00 |
| 143 | 143,8,2023-05-24 00:00:00 |
| 144 | 144,8,2023-05-25 00:00:00 |
| 145 | 145,8,2023-05-26 00:00:00 |
| 146 | 146,0,2023-05-27 00:00:00 |
| 147 | 147,0,2023-05-28 00:00:00 |
| 148 | 148,8,2023-05-29 00:00:00 |
| 149 | 149,8,2023-05-30 00:00:00 |

| | |
|---|---|
| 150 | 150,8,2023-05-31 00:00:00 |
| 151 | 151,8,2023-06-01 00:00:00 |
| 152 | 152,8,2023-06-02 00:00:00 |
| 153 | 153,0,2023-06-03 00:00:00 |
| 154 | 154,0,2023-06-04 00:00:00 |
| 155 | 155,8,2023-06-05 00:00:00 |
| 156 | 156,8,2023-06-06 00:00:00 |
| 157 | 157,8,2023-06-07 00:00:00 |
| 158 | 158,8,2023-06-08 00:00:00 |
| 159 | 159,8,2023-06-09 00:00:00 |
| 160 | 160,0,2023-06-10 00:00:00 |
| 161 | 161,0,2023-06-11 00:00:00 |
| 162 | 162,0,2023-06-12 00:00:00 |
| 163 | 163,8,2023-06-13 00:00:00 |
| 164 | 164,8,2023-06-14 00:00:00 |
| 165 | 165,8,2023-06-15 00:00:00 |
| 166 | 166,8,2023-06-16 00:00:00 |
| 167 | 167,0,2023-06-17 00:00:00 |
| 168 | 168,0,2023-06-18 00:00:00 |
| 169 | 169,8,2023-06-19 00:00:00 |
| 170 | 170,8,2023-06-20 00:00:00 |
| 171 | 171,8,2023-06-21 00:00:00 |
| 172 | 172,8,2023-06-22 00:00:00 |
| 173 | 173,8,2023-06-23 00:00:00 |
| 174 | 174,0,2023-06-24 00:00:00 |
| 175 | 175,0,2023-06-25 00:00:00 |
| 176 | 176,8,2023-06-26 00:00:00 |
| 177 | 177,8,2023-06-27 00:00:00 |
| 178 | 178,8,2023-06-28 00:00:00 |