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

ДИСЦИПЛИНА:
Проектный практикум по разработке ETL-решений

Лабораторная работа №4

Тема:

«Работа с XML и JSON файлами**»**

Выполнил(а): Морозова Валерия АДЭУ-211

Преподаватель:

Москва

2025

Вариант 10. Parking_Violations_Issued_-_Fiscal_Year_2017.csv.

Практическая работа **4.1. Анализ данных с помощью DASK**

Задание.

4.1.1. Настроить среду и рабочий каталог.

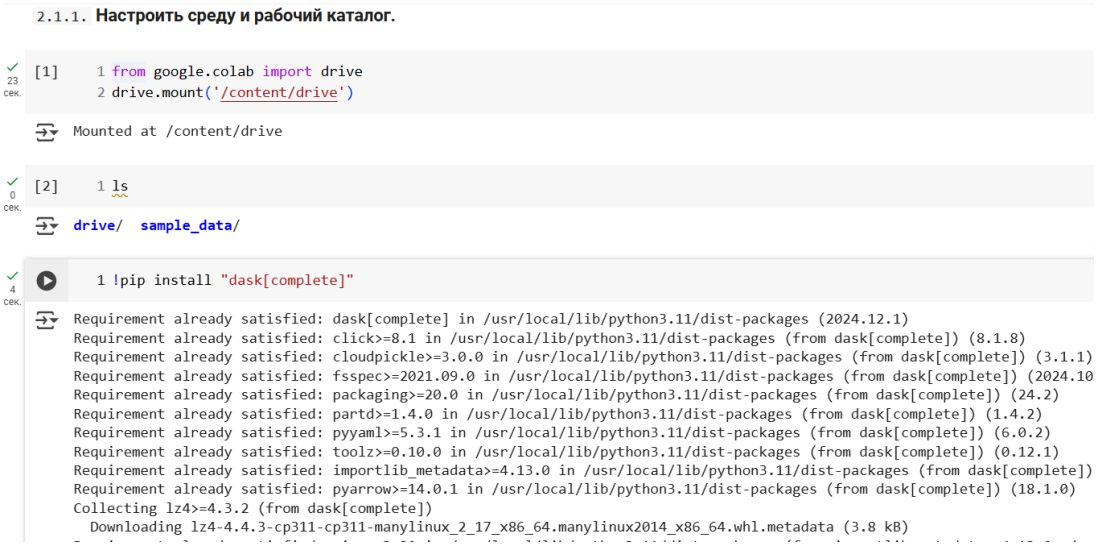


Рисунок 1. Среда и рабочий каталог настроены

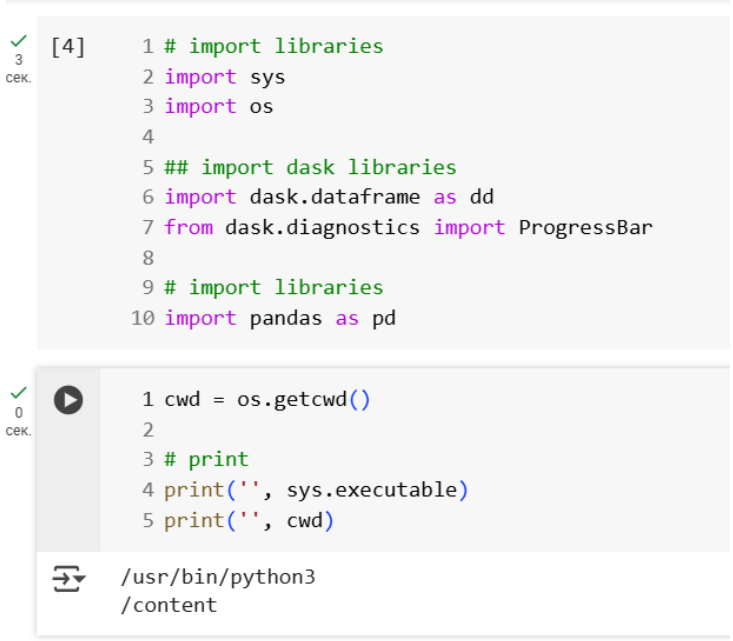


Рисунок 2. Импортированы библиотеки

4.1.2. Загрузить данные.

2.1.2. **Загрузить данные.**

```
[9]  1 df = dd.read_csv('/content/drive/MyDrive/Проектный практикум по ETL/Parking_Violations_Issued_-_Fiscal_Year_2017.csv')
     2 df
```

Dask DataFrame Structure:

	Summons Number	Plate ID	Registration State	Plate Type	Issue Date	Violation Code	Vehicle Body Type	Vehicle Make	Issuing Agency	Street Code1	Street Code2	Street Code3	Vehicle Expiration Date	Violation Location	Violation Precinct	Issuer Precinct
npartitions=32																
	int64	string	string	string	string	int64	string	string	string	int64	int64	int64	int64	float64	int64	int64
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Dask Name: to_string_dtype, 2 expressions

Рисунок 3. Данные загружены

Dask DataFrame отображает только структуру данных, а не сами данные, чтобы избежать загрузки всех данных в память при выводе на экран.

При попытке отобразить первые 10 строк данных вышла ошибка (рисунок 4).

```
ValueError: Mismatched dtypes found in `pd.read_csv`/`pd.read_table`.

+----------------------+--------+----------+
| Column               | Found  | Expected |
+----------------------+--------+----------+
| House Number         | object | float64  |
| Time First Observed | object | float64  |
+----------------------+--------+----------+

The following columns also raised exceptions on conversion:

- House Number
  ValueError("could not convert string to float: '150-34'")
- Time First Observed
  ValueError("could not convert string to float: '1138A'")

Usually this is due to dask's dtype inference failing, and
*may* be fixed by specifying dtypes manually by adding:

dtype={'House Number': 'object',
       'Time First Observed': 'object'}

to the call to `read_csv`/`read_table`.
```

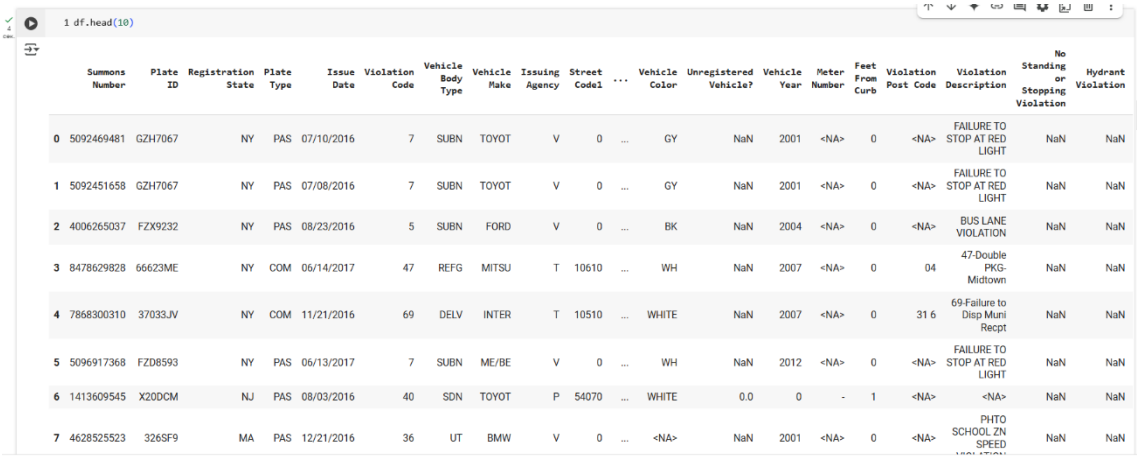
Рисунок 4. Ошибка определения типов данных

```
[9]  1 dtype = {
     2     'House Number': 'object',
     3     'Time First Observed': 'object'
     4 }
```

```
[10] 1 df = dd.read_csv(
     2     '/content/drive/MyDrive/Проектный практикум по ETL/Parking_Violations_Issued_-_Fiscal_Year_2017.csv',
     3     dtype=dtype
     4 )
```

Рисунок 5. Изменение типа данных

Согласно рекомендации колаб по решению проблемы, ошибка была исправлена и данные перезагружены с учетом изменений.



```
1 df.head(10)
```

	Summons Number	Plate ID	Registration State	Plate Type	Issue Date	Violation Code	Vehicle Body Type	Vehicle Make	Issuing Agency	Street Code1	...	Vehicle Color	Unregistered Vehicle?	Vehicle Year	Meter Number	Feet From Curb	Violation Post Code	Violation Description	No Standing or Stopping Violation	Hydrant Violation
0	5092469481	GZH7067	NY	PAS	07/10/2016	7	SUBN	TOYOT	V	0	...	GY	NaN	2001	<NA>	0	<NA>	FAILURE TO STOP AT RED LIGHT	NaN	NaN
1	5092451658	GZH7067	NY	PAS	07/08/2016	7	SUBN	TOYOT	V	0	...	GY	NaN	2001	<NA>	0	<NA>	FAILURE TO STOP AT RED LIGHT	NaN	NaN
2	4006265037	FZX9232	NY	PAS	08/23/2016	5	SUBN	FORD	V	0	...	BK	NaN	2004	<NA>	0	<NA>	BUS LANE VIOLATION	NaN	NaN
3	8478629828	66623ME	NY	COM	06/14/2017	47	REFG	MITSU	T	10610	...	WH	NaN	2007	<NA>	0	04	47-Double PKG-Midtown	NaN	NaN
4	7868300310	37033JV	NY	COM	11/21/2016	69	DELV	INTER	T	10510	...	WHITE	NaN	2007	<NA>	0	31 6	69-Failure to Disp Muni Recpt	NaN	NaN
5	5096917368	FZD8593	NY	PAS	06/13/2017	7	SUBN	ME/BE	V	0	...	WH	NaN	2012	<NA>	0	<NA>	FAILURE TO STOP AT RED LIGHT	NaN	NaN
6	1413609545	X20DCM	NJ	PAS	08/03/2016	40	SDN	TOYOT	P	54070	...	WHITE	0.0	0	-	1	<NA>	<NA>	NaN	NaN
7	4628525523	326SF9	MA	PAS	12/21/2016	36	UT	BMW	V	0	...	<NA>	NaN	2001	<NA>	0	<NA>	PHTO SCHOOL ZN SPEED	NaN	NaN

Рисунок 6. Отображение первых 10 строк

4.1.3. Проверить качество данных (например, отсутствующие значения и выбросы).

2.1.3. Проверить качество данных (например, отсутствующие значения и выбросы).

```
[10]  1 # count missing values
      2 missing_values = df.isnull().sum()
      3 missing_values
```

```
Dask Series Structure:
npartitions=1
Date First Observed    int64
Violation Time           ...
Dask Name: sum, 5 expressions
Expr=(~ NotNull(frame=ArrowStringConversion(frame=FromMapProjectable(4254b71)))).sum()
```

Рисунок 7. Проверка качества данных

4.1.4. Удалить столбцы (множество пропусков в значениях, бесполезные столбцы для анализа).

```
[15]  1 result = df.compute()
      2 print(result)
```

```
/usr/local/lib/python3.11/dist-packages/dask/dataframe/io/csv.py:199: DtypeWarning: Columns (18,38) have mixed types. Specify dtype option on import or set low_memory=False.
  df = reader(bio, **kwargs)
        Summons Number Plate ID Registration State Plate Type  Issue Date  \
0           5092469481  GZH7067                NY        PAS  07/10/2016
1           5092451658  GZH7067                NY        PAS  07/08/2016
2           4006265037  FZX9232                NY        PAS  08/23/2016
3           8478629828  66623ME                NY        COM  06/14/2017
4           7868300310  370333V                NY        COM  11/21/2016
...                ...      ...               ...        ...         ...
352313      1415891400  HGK6453                NJ        PAS  11/02/2068
352314      1384716543  GRA6240                NY        PAS  07/12/2069
352315      1413536554   RC8S28                PA        PAS  08/14/2069
352316      1415514203  HGU9544                NY        PAS  11/15/2069
352317      1415995370  GPP1608                NY        PAS  11/19/2069

        Violation Code Vehicle Body Type Vehicle Make Issuing Agency  \
0                    7              SUBN         TOYOT              V
1                    7              SUBN         TOYOT              V
2                    5              SUBN          FORD              V
3                   47              REFG         MITSU              T
4                   69              DELV         INTER              T
...                ...               ...           ...            ...
352313              21               SDN         HONDA              S
352314              20              SUBN         TOYOT              X
352315              46              SUBN          <NA>              P
352316              40              SUBN          JEEP              P
```

Рисунок 8. Выведение результата

```
[16]  1 # calculate percent missing values
      2 mysize = df.index.size
      3 missing_count = ((missing_values / mysize) * 100)
      4 missing_count

Dask Series Structure:
npartitions=1
Date First Observed    float64
Violation Time             ...
Dask Name: mul, 9 expressions
Expr=(~ NotNull(frame=ArrowStringConversion(frame=FromMapProjectable(4d4d1e8)))).sum() / Index(frame=ArrowStringConversion(frame=FromMapProjectable(4d4d1e8))).size() * 100
```

Рисунок 9. Расчет процента пропущенных значений

```
# запуск вычисления, используя метод подсчета
with ProgressBar():
    missing_count_percent = missing_count.compute()
missing_count_percent
```

Violation Location	19.183510
Violation Precinct	0.000000
Issuer Precinct	0.000000
Issuer Code	0.000000
Issuer Command	19.093212
Issuer Squad	19.101506
Violation Time	0.000583
Time First Observed	92.217488
Violation County	0.366073
Violation In Front Of Or Opposite	20.005826
House Number	21.184968
Street Name	0.037110
Intersecting Street	68.827675

Рисунок 10. Вычисление с помощью метода подсчёта

2.1.4. **Удалить столбцы (пропуски в значениях, бесполезные столбцы для анализа).**

```
# Получаем список столбцов, которые нужно удалить
columns_to_drop = missing_count_percent[missing_count_percent > 60].index

# Проверяем, что columns_to_drop не пуст
if not columns_to_drop.empty:
    print("Столбцы для удаления:", columns_to_drop)

    # Удаляем столбцы
    with ProgressBar():
        df_dropped = df.drop(columns=columns_to_drop).compute()
else:
    print("Нет столбцов для удаления.")
    df_dropped = df.compute()
```

```
Столбцы для удаления: Index(['Time First Observed', 'Intersecting Street', 'Violation Legal Code',
       'Unregistered Vehicle?', 'Meter Number',
       'No Standing or Stopping Violation', 'Hydrant Violation',
       'Double Parking Violation'],
      dtype='object')
[#######                                 ] | 18% Completed | 29.94 s/usr/local/lib/python3.11/dist-packages/dask/data
  df = reader(bio, **kwargs)
[########################################] | 100% Completed | 132.93 s
```

Рисунок 11. Удаление бесполезных столбцов и пропущенных значений

Удаление столбцов осуществляется при условии процента пустых значений больше 60. Чтобы избежать ошибки была добавлена проверка if not columns_to_drop.empty, чтобы убедиться, что columns_to_drop не пуст перед выполнением операции drop. Это предотвращает попытку удаления несуществующих столбцов.

4.2.1. Визуализировать DAG с одним узлом и зависимостями.

Пример

```
0
сек.
    return i + 1

def add(x, y):
    return x + y

# функции переноса внутри отложенного объекта и цепочки
x = delayed(increment)(1)
y = delayed(increment)(2)
z = delayed(add)(x, y)

# visualize the DAG
z.visualize()
```

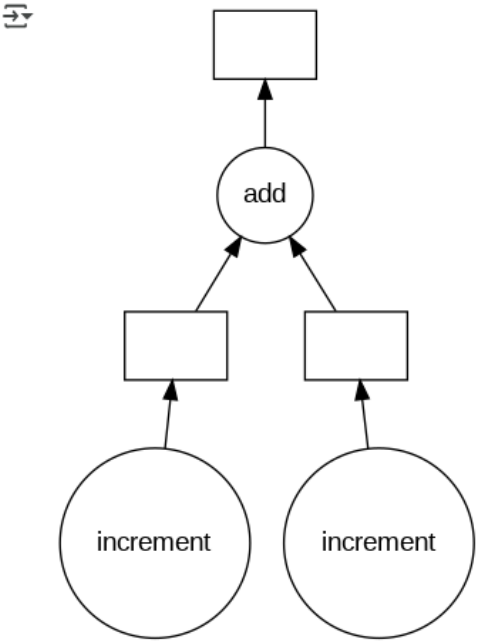


Рисунок 12. Пример визуализации DAG с одним узлом и зависимостями

```
    return f"Column: {column_name}"

# Список столбцов таблицы
columns = [
    "Номер повестки", "Государство регистрации", "Тип номерного знака", "Дата выдачи",
"Код нарушения", "Марка транспортного средства", "Цвет транспортного средства", "Почтовый индекс нарушения", "Описание нарушения"
]

# Создаем отложенные объекты для каждого столбца
delayed_columns = [create_column(columns) for column in columns]

# Создаем отложенный объект для таблицы, которая зависит от всех столбцов
@delayed
def create_table(columns):
    return "Штраф: " + ", ".join(columns)

# Таблица зависит от всех столбцов
delayed_penalty = create_table(delayed_columns)

# Визуализация DAG
delayed_penalty.visualize()
```

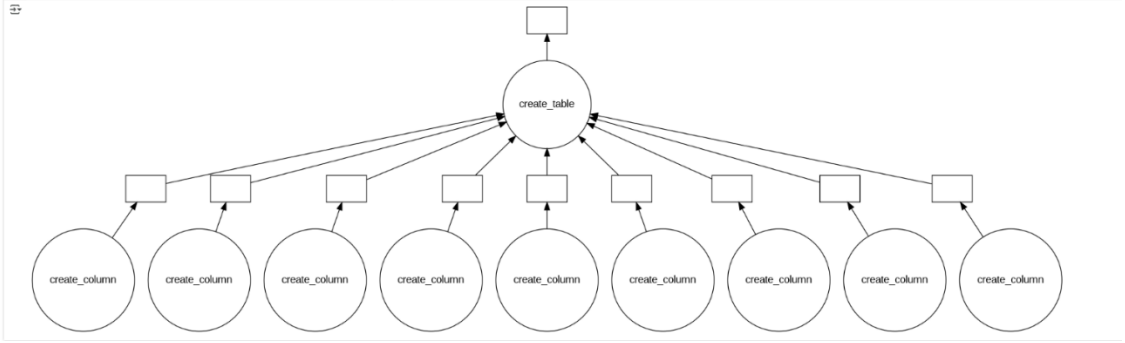


Рисунок 13. Свой вариант визуализации DAG с одним узлом и зависимостями

4.2.1. Визуализировать DAG с более чем одним узлом и зависимостями.

```
def double(x):
    return x * 2

# compile second layer and visualize
layer2 = [delayed(double)(j) for j in layer1]
total2 = delayed(sum)(layer2)#.persist()
total2.visualize()
```

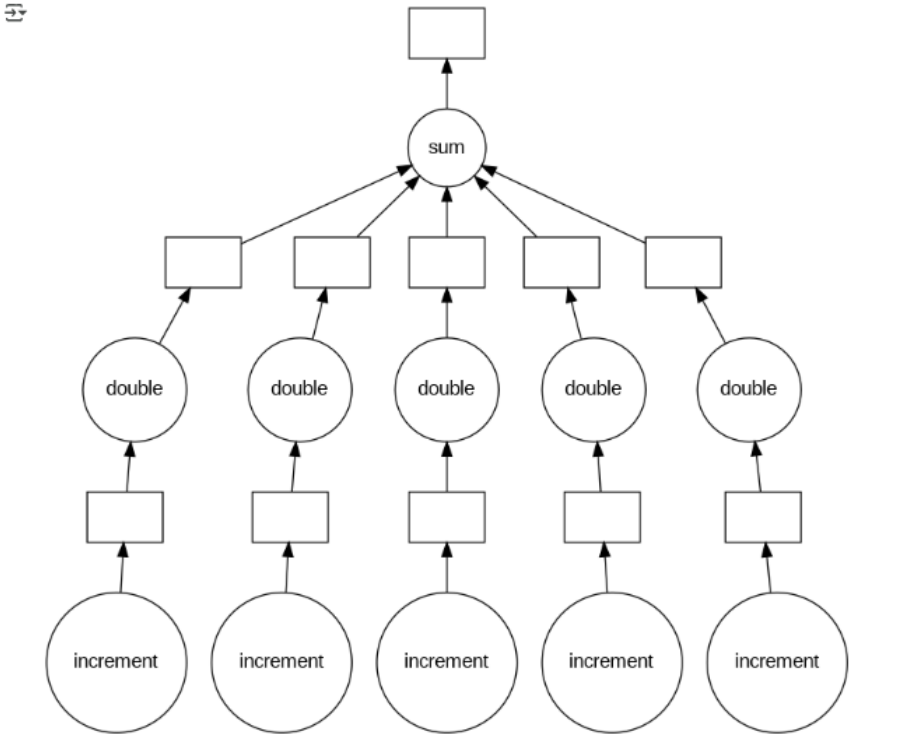


Рисунок 14. Пример визуализации DAG с более чем одним узлом и зависимостями

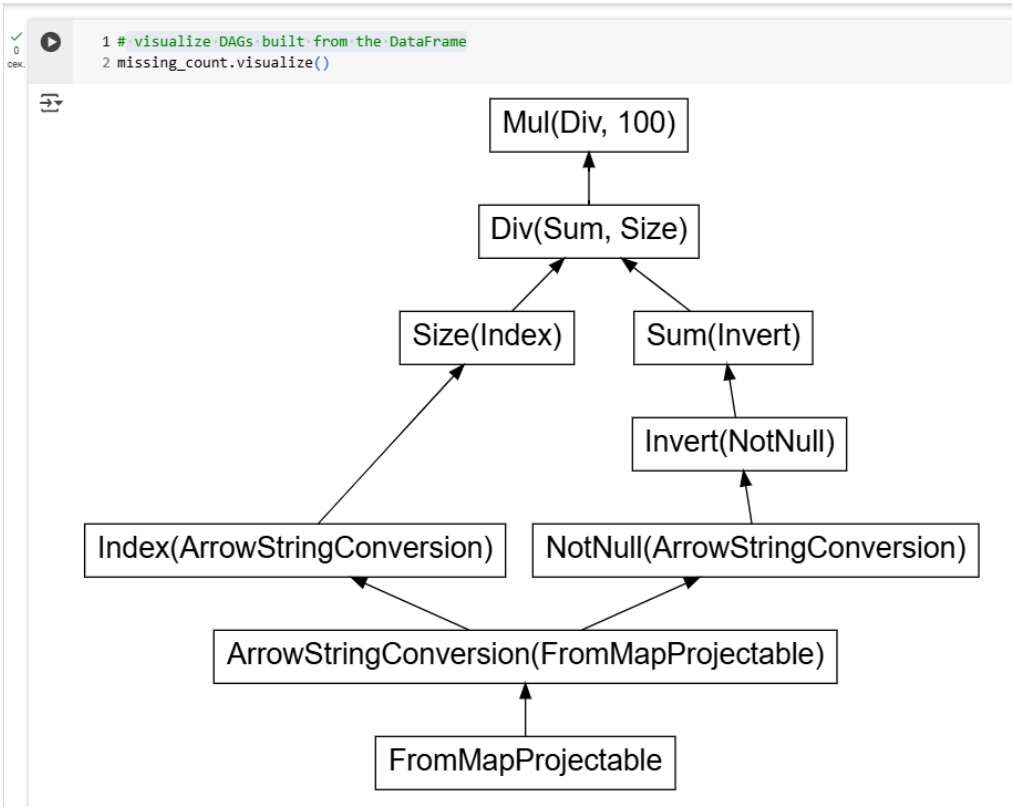


Рисунок 15. Визуализация, построенная на основании data frame