“Different File Format Comparison: Guide To CSV, Excel, Avro, Parquet and ORC”

PRESENTATION BY:

NIMISHA MALVIYA (989544)

AGENDA

1

INTRODUCTION

2

WHY CHOOSE BETWEEN THIS?

3

COMMA SEPARATED VALUES(CSV) FILE

4

EXCEL WORKBOOK XLXS

5

COMPARISON OF EXCEL & CSV

6

AVRO

7

PARQUET

8

ORC (OPTIMIZED ROW COLUMNAR)

9

RECOMMENDATIONS

10

CONCLUSION

INTRODUCTION

- Handling data efficiently is vital for every organization.

The right file format choice optimizes storage, processing, and analysis.

- **Comparative Analysis**:

Explore popular file formats like CSV, XLSX, Avro, Parquet, and ORC to make informed decisions for our data warehouse operations.

Why choose between different file formats?

DATA STRUCTURE COMPLEXITY

Different data structures require different file formats to efficiently store and represent the data.

PERFORMANCE OPTIMIZATION

Columnar storage formats like Parquet & ORC are designed for analytical queries & Row-based formats like CSV may be more suitable for transactional data.

SCHEMA EVOLUTION AND FLEXIBILITY

This flexibility is essential for evolving data requirements and adapting to changing business needs.

SECURITY AND ACCESS CONTROL

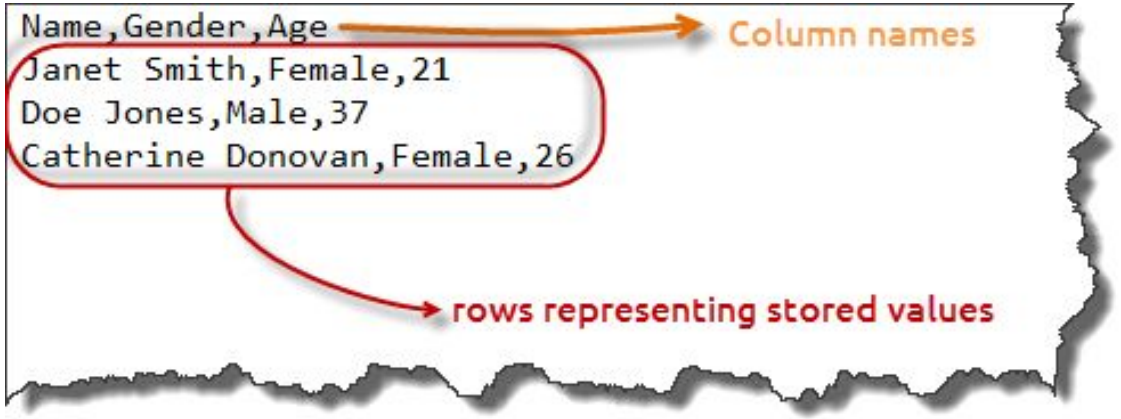
Encryption and access control mechanisms, to protect sensitive data from unauthorized access or tampering.

COMMA SEPARATED VALUES(CSV) FILE

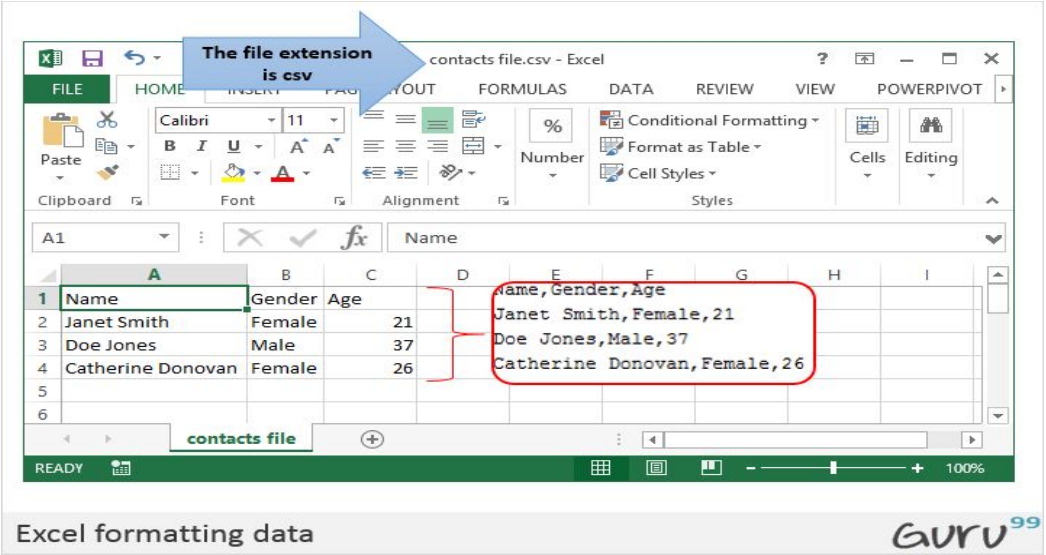
It is used to store tabular data where the column names and row values are separated using commas.

Advantages:-

- Simplicity: Easy to create and understand, suitable for basic data storage.
- Human-readable: Can be opened and edited using a simple text editor.
- Widely supported: Compatible with most data processing tools and programming languages.



```
Name,Gender,Age
Janet Smith,Female,21
Doe Jones,Male,37
Catherine Donovan,Female,26
```



The file extension is csv

contacts file.csv - Excel

	A	B	C
1	Name	Gender	Age
2	Janet Smith	Female	21
3	Doe Jones	Male	37
4	Catherine Donovan	Female	26
5			
6			

```
Name,Gender,Age
Janet Smith,Female,21
Doe Jones,Male,37
Catherine Donovan,Female,26
```

Excel formatting data

Guru99

LIMITATIONS

LACK OF DATA TYPE INFORMATION

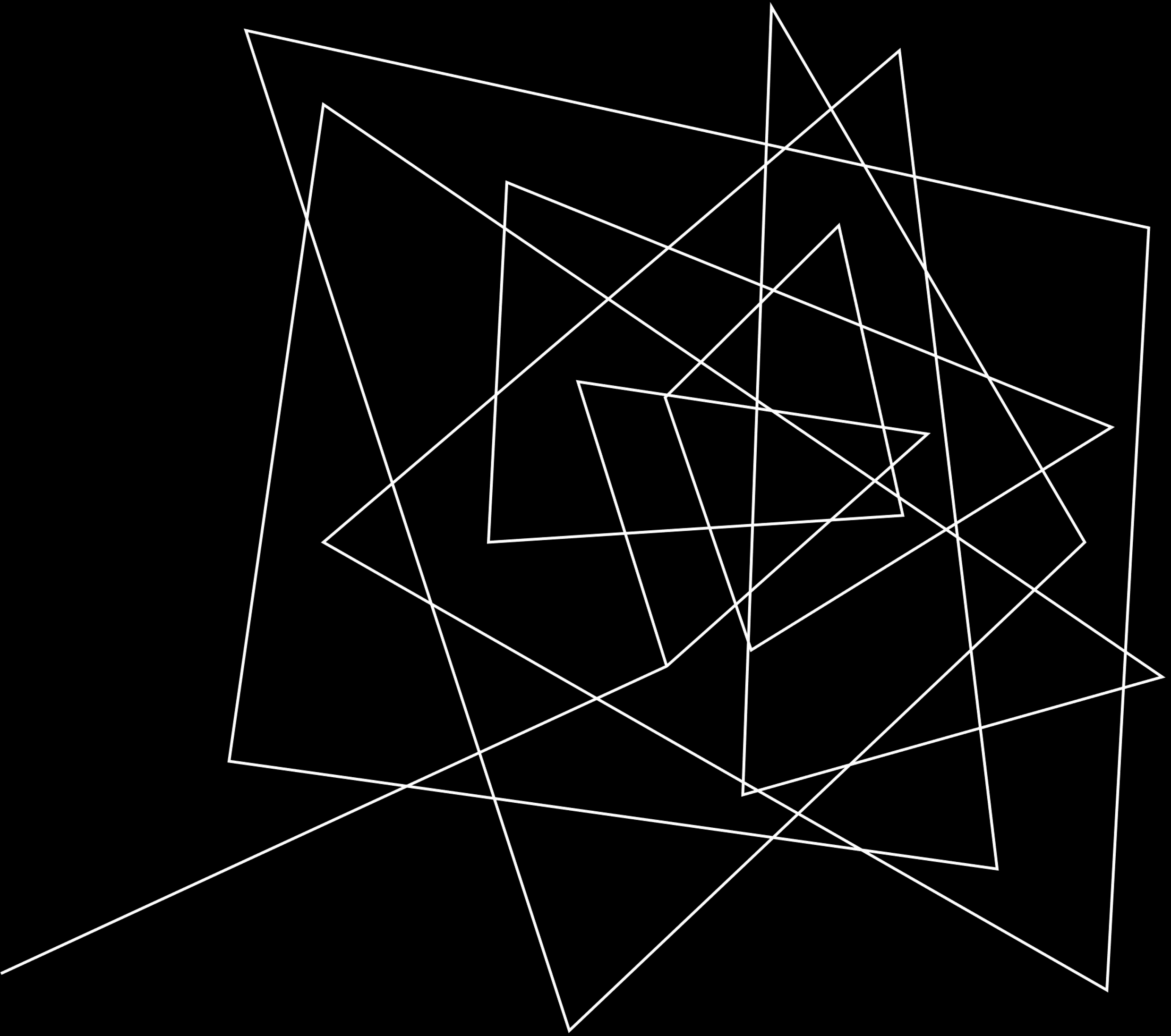
INEFFICIENT FOR LARGE DATASETS

POTENTIAL DATA LOSS

LIMITED METADATA SUPPORT

LIMITED DATA VALIDATION

PERFORMANCE DEGRADATION WITH WIDE TABLES



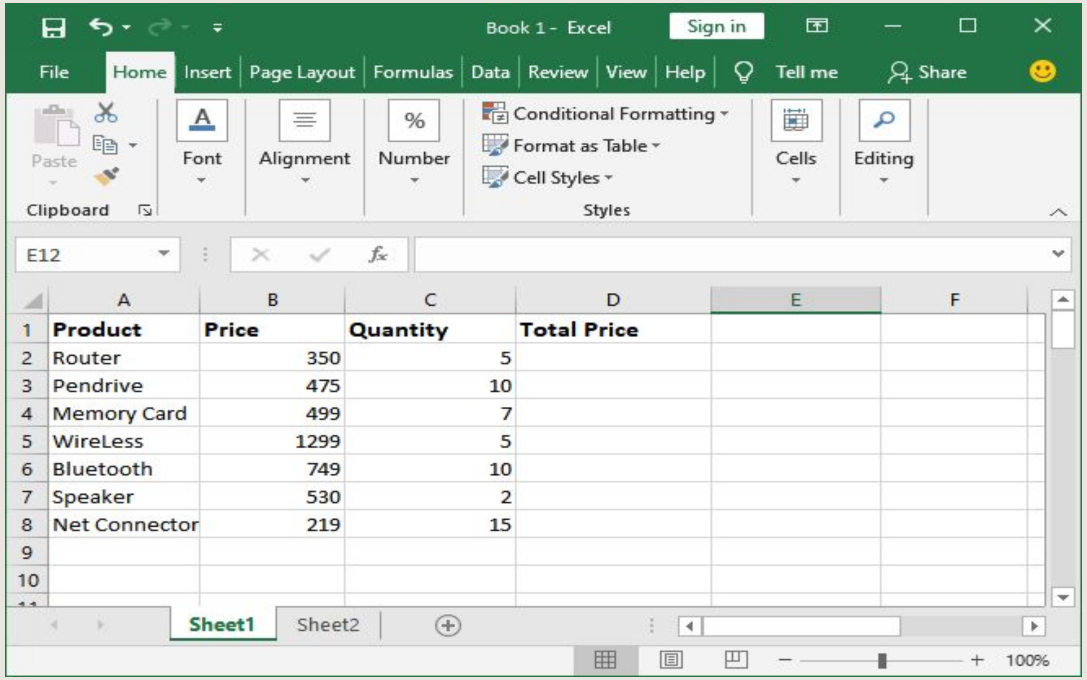
MS EXCEL WORKBOOK

XLSX(MS EXCEL WORKBOOK)

- XLSX is a binary file format used by Microsoft Excel to store spreadsheet data.
- It supports multiple sheets within a single workbook, as well as various formatting options and formulas.

Advantages:-

- Supports multiple sheets
- Formulas: Enables the use of formulas for calculations and data manipulation.
- Formatting: Supports various formatting options for text, numbers, and visuals.

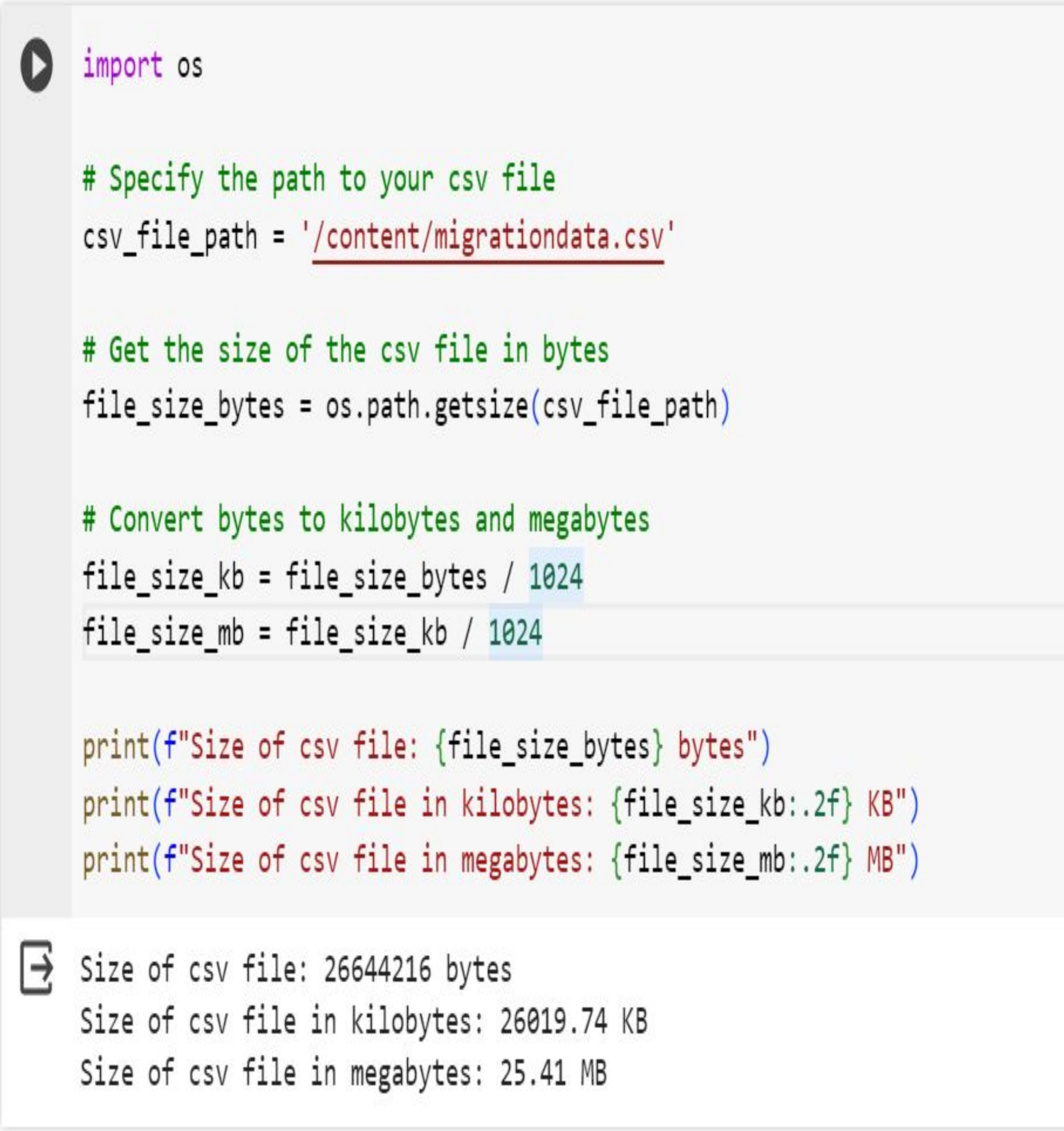


	A	B	C	D	E	F
1	**Product**	**Price**	**Quantity**	**Total Price**		
2	Router	350	5			
3	Pendrive	475	10			
4	Memory Card	499	7			
5	WireLess	1299	5			
6	Bluetooth	749	10			
7	Speaker	530	2			
8	Net Connector	219	15			
9						
10						

Comparison between CSV File and Excel

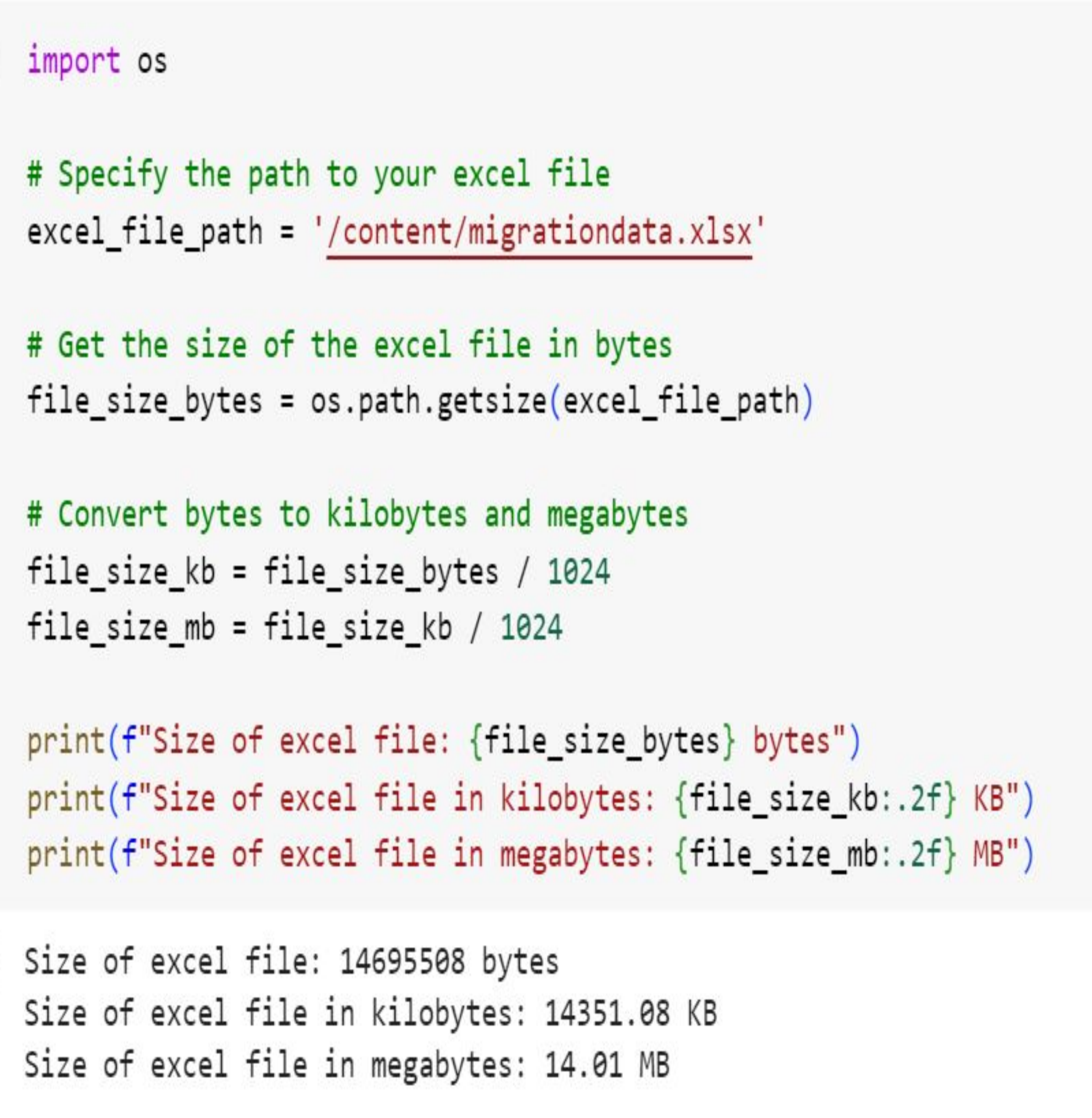
COMPARISON BETWEEN CSV AND EXCEL

CSV	EXCEL
CSV format is a plain text format with a series of values separated by commas.	Excel is a binary file that holds information about all the worksheets in a file, including both content and formatting
Opened with any text editor in Windows like notepad, MS Excel, OpenOffice,etc.	Only be read by applications that have been especially written to read their format, and can only be written in the same way.
Importing CSV files can be much faster, consumes less memory	Consumes more memory while importing data
Apart from text, data can also be stored in form of charts and graphs	CSV can not store charts or graphs
In data-warehouse, CSV follows a fairly flat, simple schema	In data-warehouse, Excel is preferable for detailed standardized schema specification
Can connect to external data sources to fetch data. It allows for Review of Data with detailed tracking and commenting feature	All this functionality is not possible in CSV



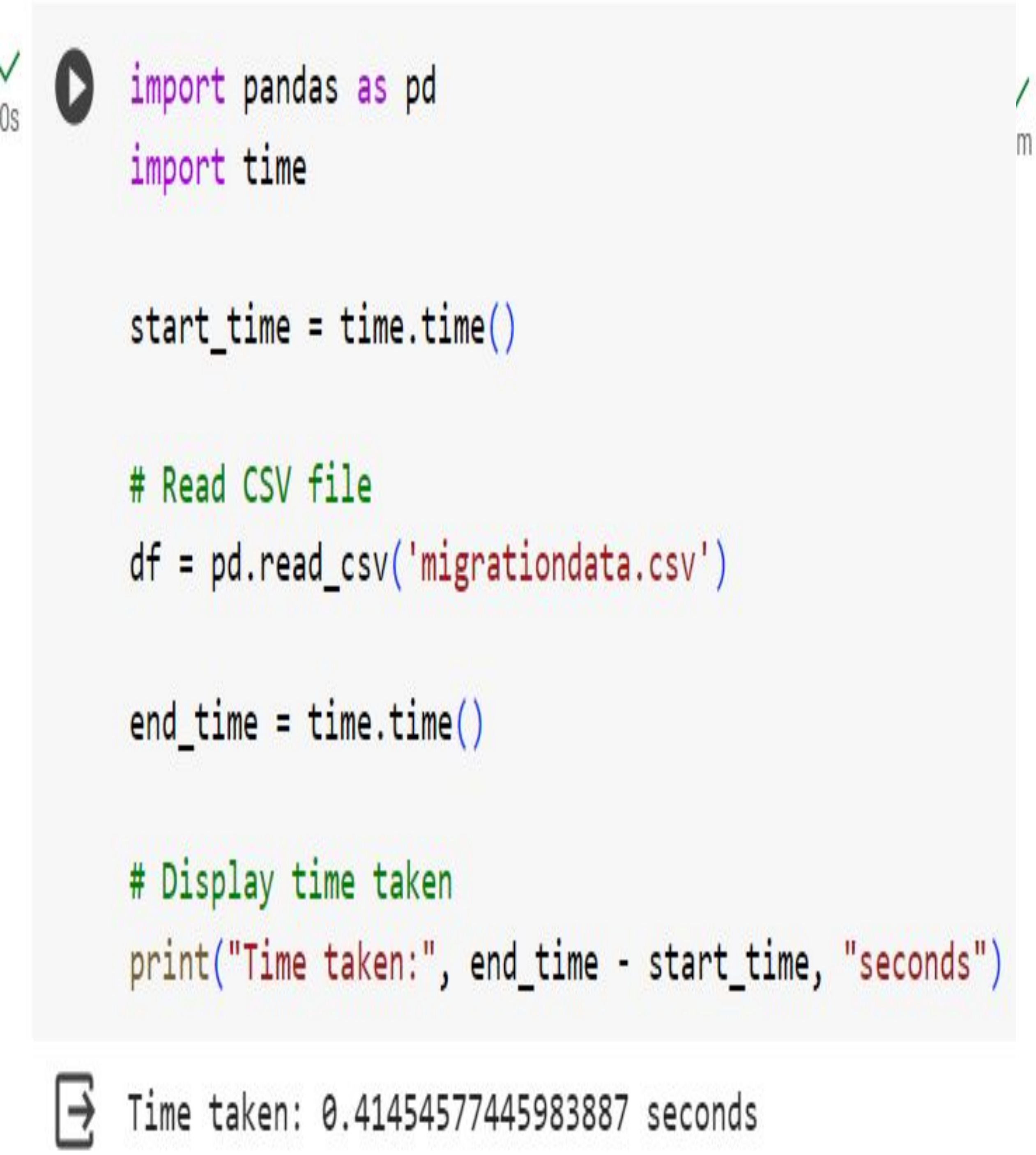
```
import os

# Specify the path to your csv file
csv_file_path = '/content/migrationdata.csv'

# Get the size of the csv file in bytes
file_size_bytes = os.path.getsize(csv_file_path)

# Convert bytes to kilobytes and megabytes
file_size_kb = file_size_bytes / 1024
file_size_mb = file_size_kb / 1024

print(f"Size of csv file: {file_size_bytes} bytes")
print(f"Size of csv file in kilobytes: {file_size_kb:.2f} KB")
print(f"Size of csv file in megabytes: {file_size_mb:.2f} MB")
```

```
Size of csv file: 26644216 bytes
Size of csv file in kilobytes: 26019.74 KB
Size of csv file in megabytes: 25.41 MB
```



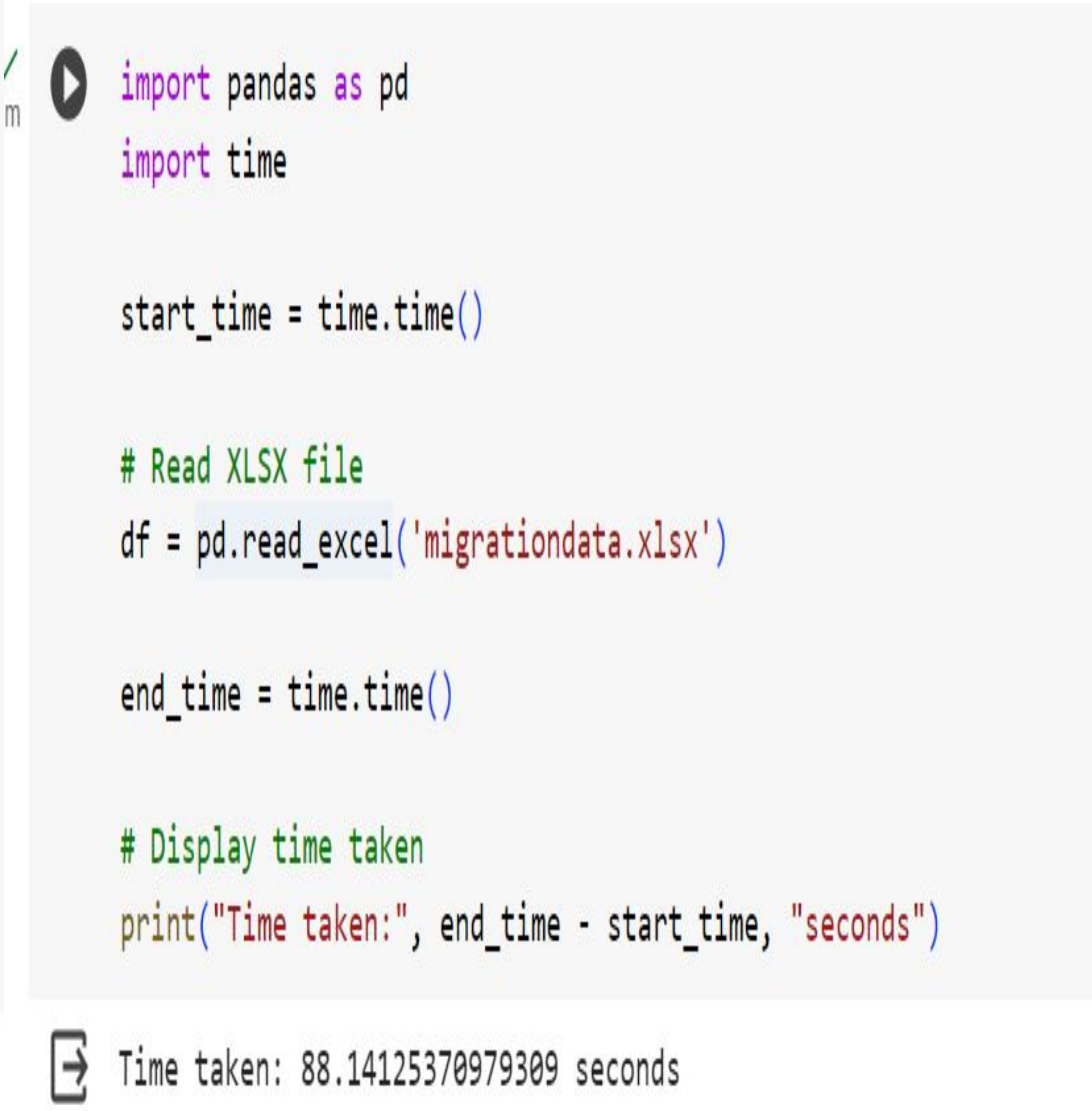
```
import os

# Specify the path to your excel file
excel_file_path = '/content/migrationdata.xlsx'

# Get the size of the excel file in bytes
file_size_bytes = os.path.getsize(excel_file_path)

# Convert bytes to kilobytes and megabytes
file_size_kb = file_size_bytes / 1024
file_size_mb = file_size_kb / 1024

print(f"Size of excel file: {file_size_bytes} bytes")
print(f"Size of excel file in kilobytes: {file_size_kb:.2f} KB")
print(f"Size of excel file in megabytes: {file_size_mb:.2f} MB")
```

```
Size of excel file: 14695508 bytes
Size of excel file in kilobytes: 14351.08 KB
Size of excel file in megabytes: 14.01 MB
```



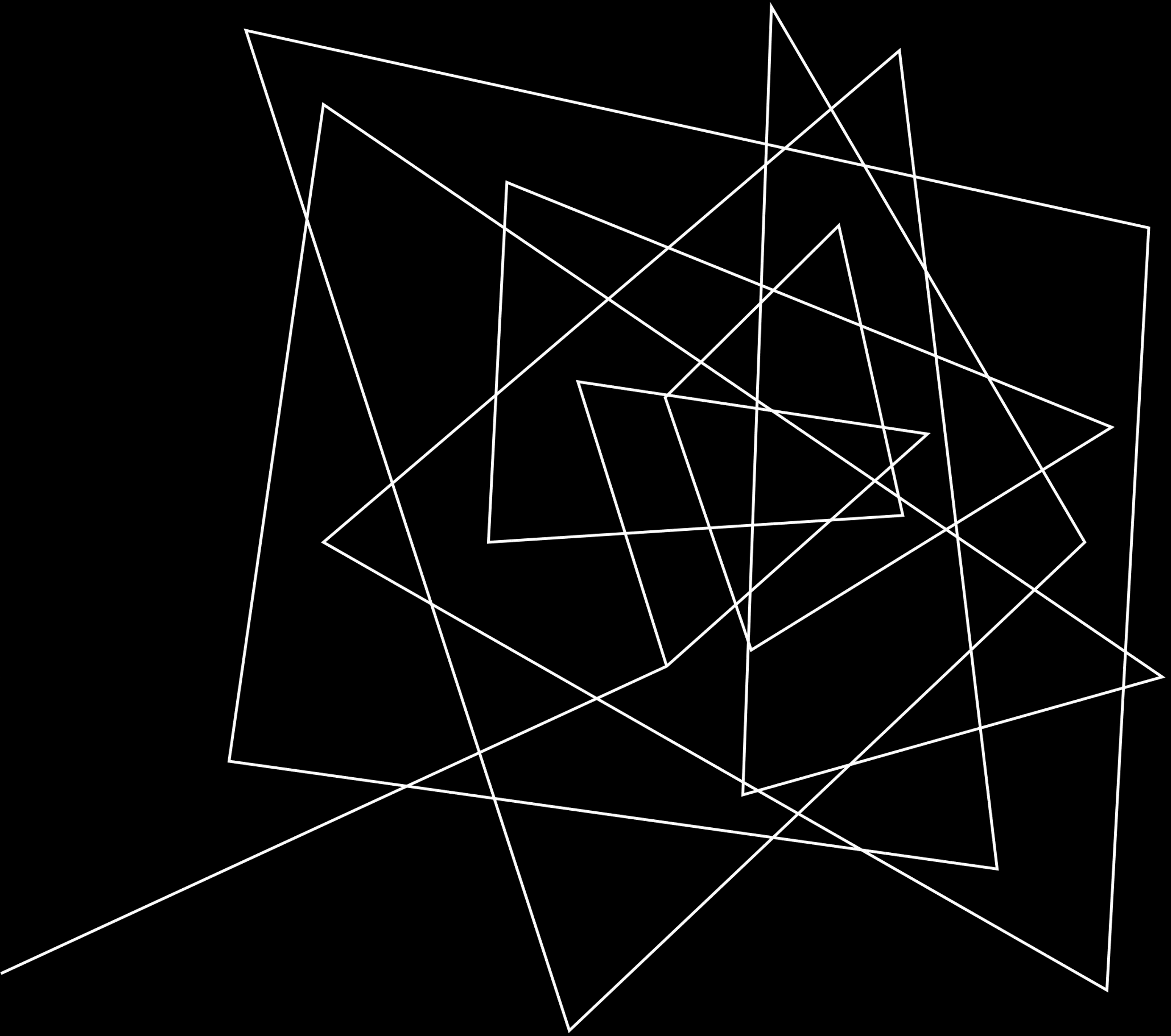
```
import pandas as pd
import time

start_time = time.time()

# Read CSV file
df = pd.read_csv('migrationdata.csv')

end_time = time.time()

# Display time taken
print("Time taken:", end_time - start_time, "seconds")
```

```
Time taken: 0.41454577445983887 seconds
```



```
import pandas as pd
import time

start_time = time.time()

# Read XLSX file
df = pd.read_excel('migrationdata.xlsx')

end_time = time.time()

# Display time taken
print("Time taken:", end_time - start_time, "seconds")
```

```
Time taken: 88.14125370979309 seconds
```



AVRO

AVRO

AVRO FORMAT

- Avro is a binary serialization format developed within the Apache Hadoop project.
- It defines a schema for the data, allowing for schema evolution over time.

ADVANTAGES

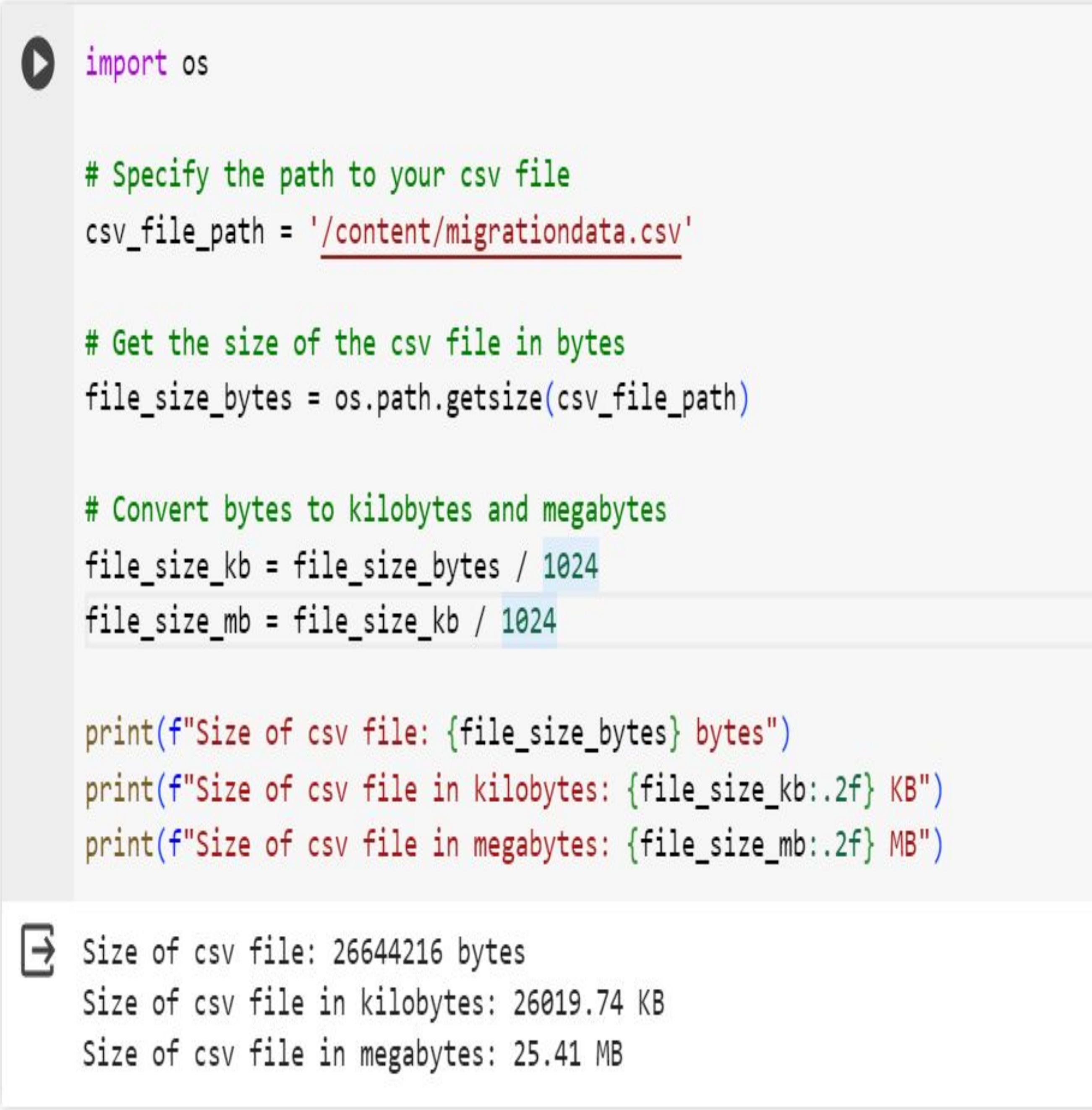
- Schema evolution
- Efficient serialization: Compact binary format leads to efficient serialization and deserialization of data.
- Data compression

LIMITATIONS

- Not human-readable: Avro files are binary and not easily readable without specific tools or libraries.
- Limited support in some ecosystems: While widely used in Hadoop ecosystems, support may be limited in other environments or tools.

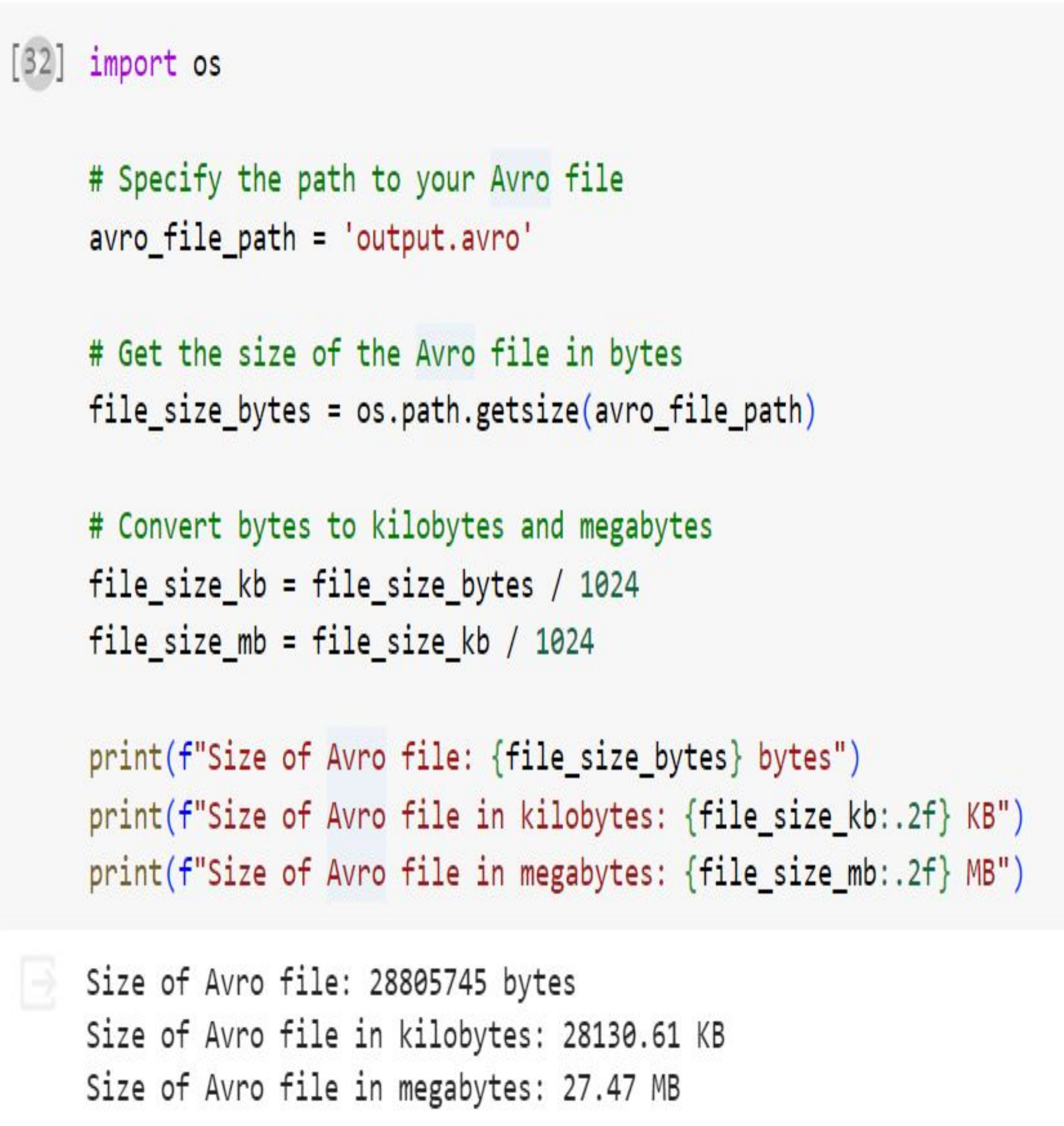
AVRO FORMAT FEATURES
Can be shared by programs using different languages
Self-describing; bundles serialized data with data schema
Supports schema evolution and flexibility
Splittable
Compression options including uncompressed, snappy, deflate, bzip2, and xz

AVRO FORMAT USE CASE
Write-heavy operations (such as ingestion into a data lake) due to serialized row-based storage.
When writing speed with schema evolution (adaptability to change in metadata) is critical.



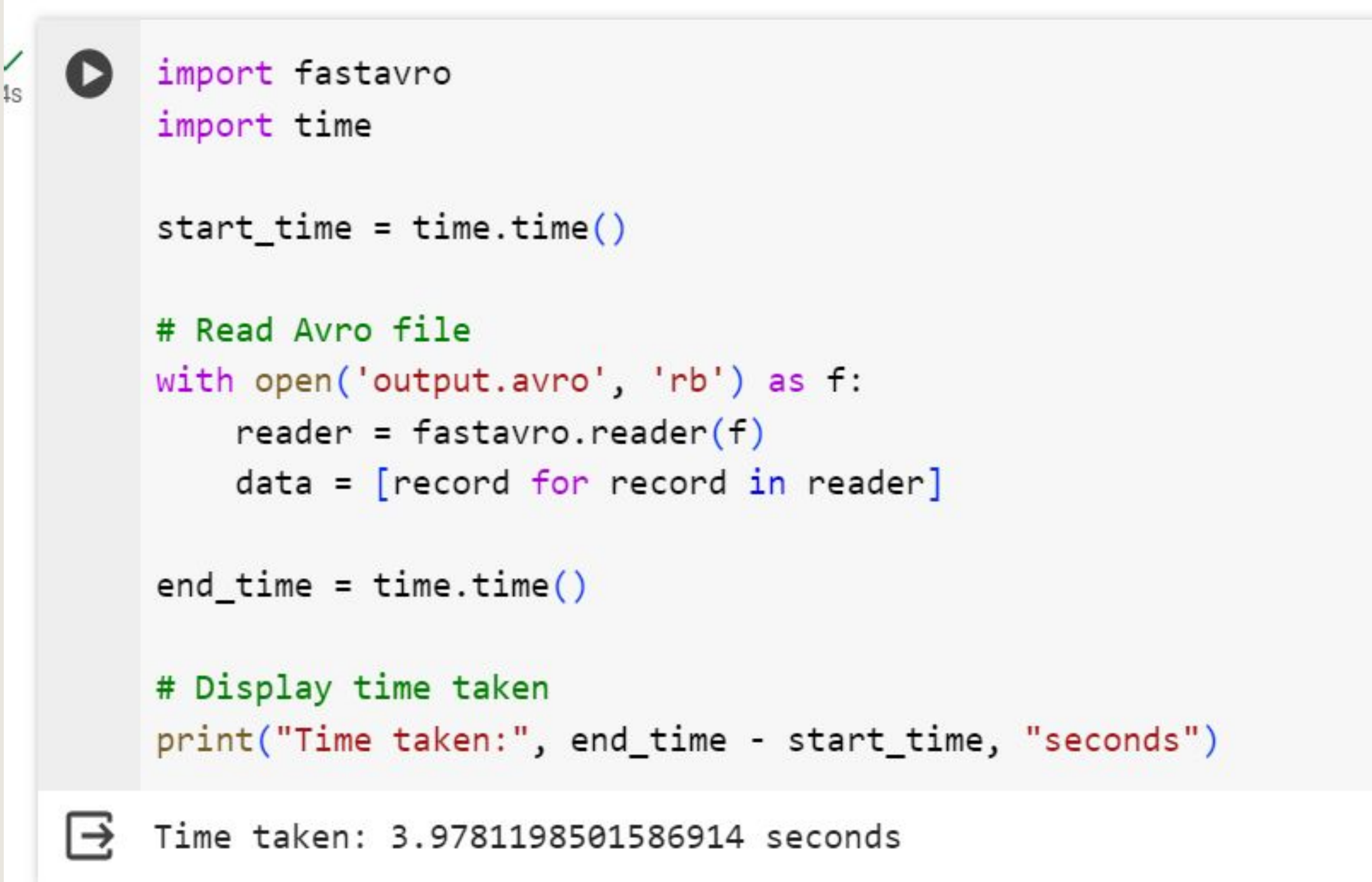
```python
import os

# Specify the path to your csv file
csv_file_path = '/content/migrationdata.csv'

# Get the size of the csv file in bytes
file_size_bytes = os.path.getsize(csv_file_path)

# Convert bytes to kilobytes and megabytes
file_size_kb = file_size_bytes / 1024
file_size_mb = file_size_kb / 1024

print(f"Size of csv file: {file_size_bytes} bytes")
print(f"Size of csv file in kilobytes: {file_size_kb:.2f} KB")
print(f"Size of csv file in megabytes: {file_size_mb:.2f} MB")
```

```
Size of csv file: 26644216 bytes
Size of csv file in kilobytes: 26019.74 KB
Size of csv file in megabytes: 25.41 MB
```



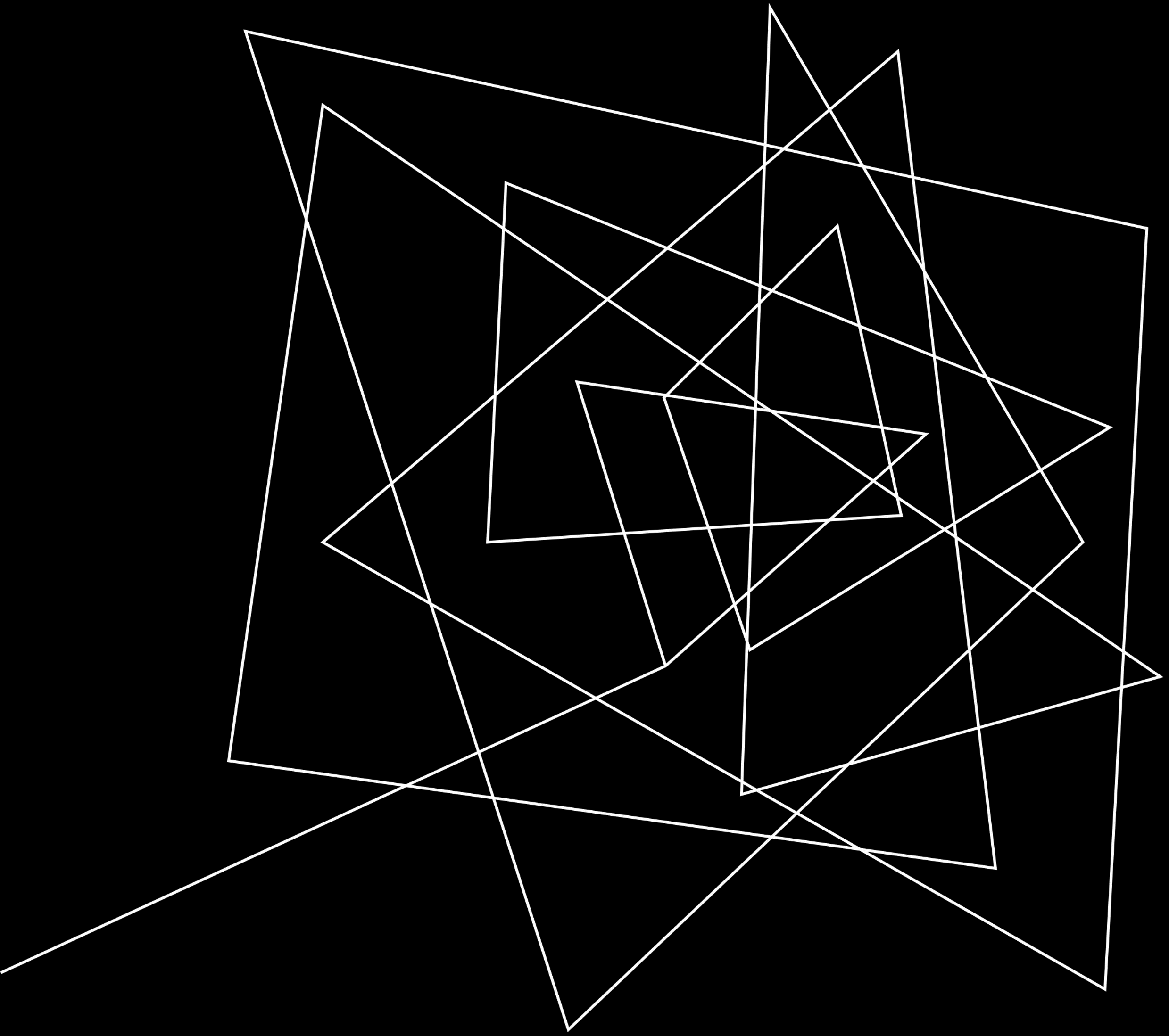
```python
import os

# Specify the path to your Avro file
avro_file_path = 'output.avro'

# Get the size of the Avro file in bytes
file_size_bytes = os.path.getsize(avro_file_path)

# Convert bytes to kilobytes and megabytes
file_size_kb = file_size_bytes / 1024
file_size_mb = file_size_kb / 1024

print(f"Size of Avro file: {file_size_bytes} bytes")
print(f"Size of Avro file in kilobytes: {file_size_kb:.2f} KB")
print(f"Size of Avro file in megabytes: {file_size_mb:.2f} MB")
```

```
Size of Avro file: 28805745 bytes
Size of Avro file in kilobytes: 28130.61 KB
Size of Avro file in megabytes: 27.47 MB
```



```python
import fastavro
import time

start_time = time.time()

# Read Avro file
with open('output.avro', 'rb') as f:
    reader = fastavro.reader(f)
    data = [record for record in reader]

end_time = time.time()

# Display time taken
print("Time taken:", end_time - start_time, "seconds")
```

```
Time taken: 3.9781198501586914 seconds
```



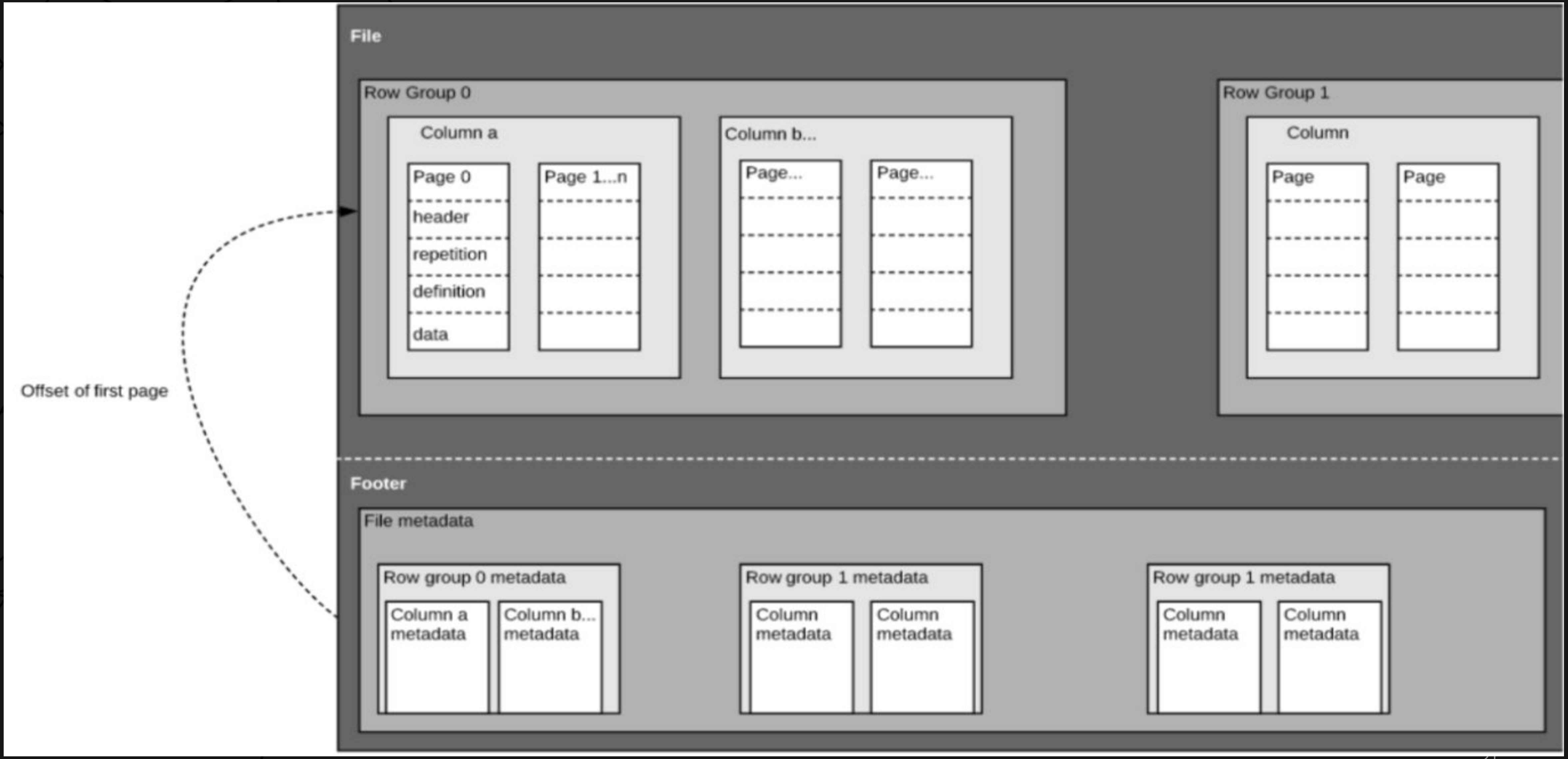
PARQUET

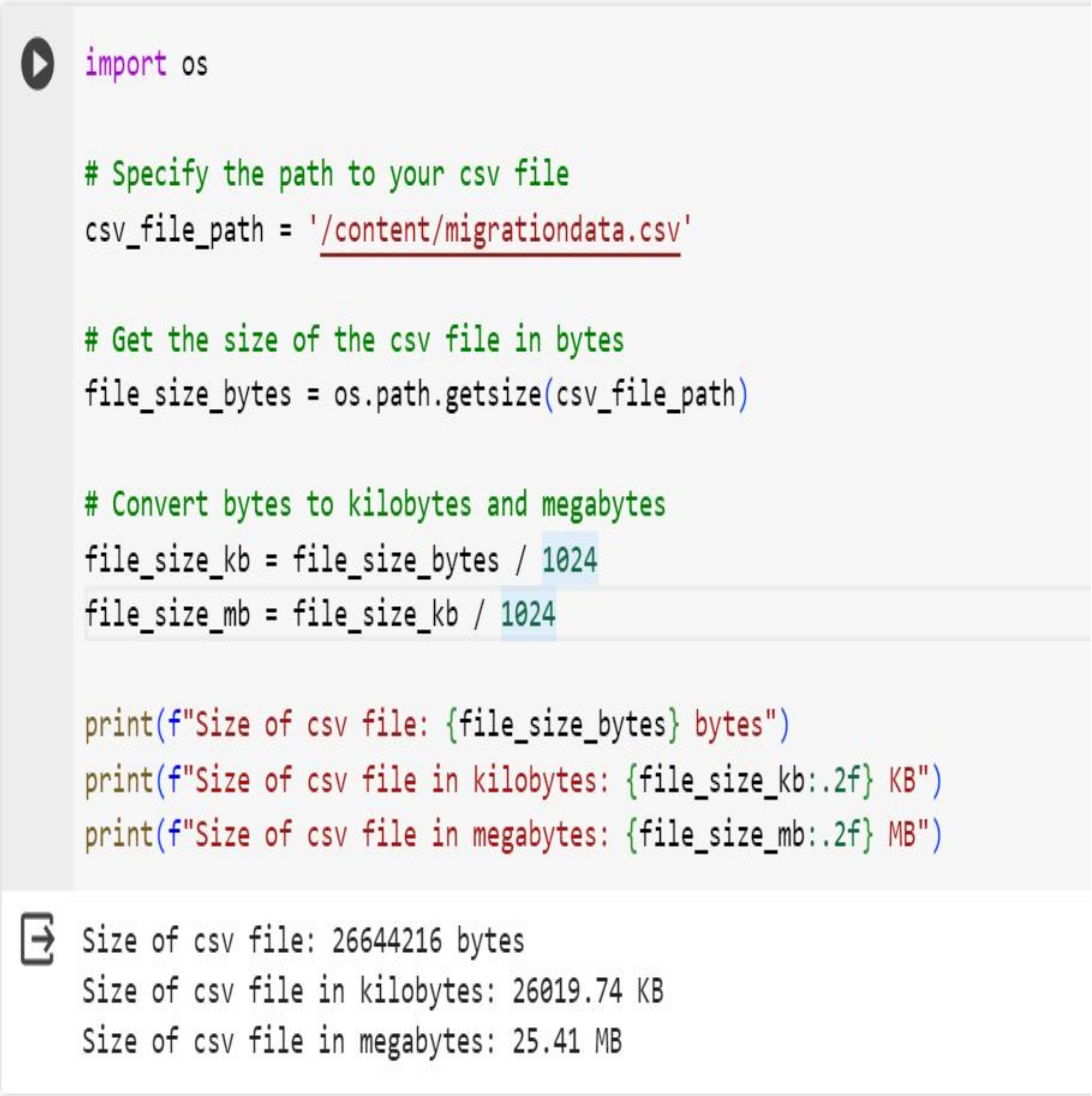
PARQUET

Parquet Format Features
Language agnostic
Supports complex data types
Multiple flexible compression options
Supports schema evolution
Enables data skipping, reduced I/O

Ideal Parquet Use Case
Storing big data of any kind (structured data tables, images, videos, documents).
Ideal for services such as AWS Athena and Amazon Redshift Spectrum, which are serverless, interactive technologies
A good fit for Snowflake as it supports extremely efficient compression and encoding schemes.
When your full dataset has many columns, but you only need to access a subset.
When you want multiple services to consume the same data from object storage.
When you're largely or wholly dependent on Spark.

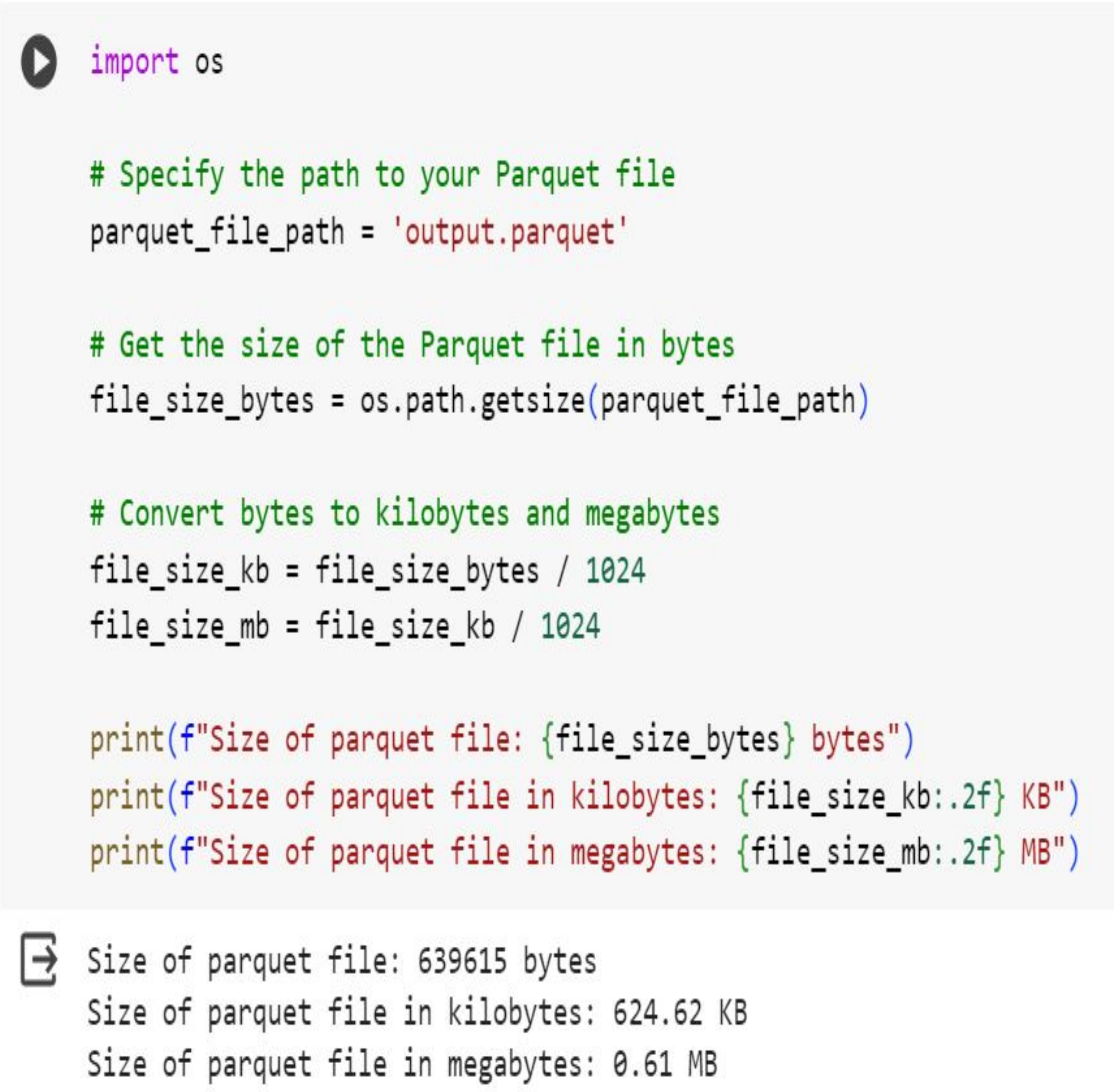
PARQUET


File
Row Group 0
Column a
Page 0
header
repetition
definition
data
Page 1...n
Column b...
Page...
Page...
Row Group 1
Column
Page
Page
Offset of first page
Footer
File metadata
Row group 0 metadata
Column a metadata
Column b... metadata
Row group 1 metadata
Column metadata
Column metadata
Row group 1 metadata
Column metadata
Column metadata



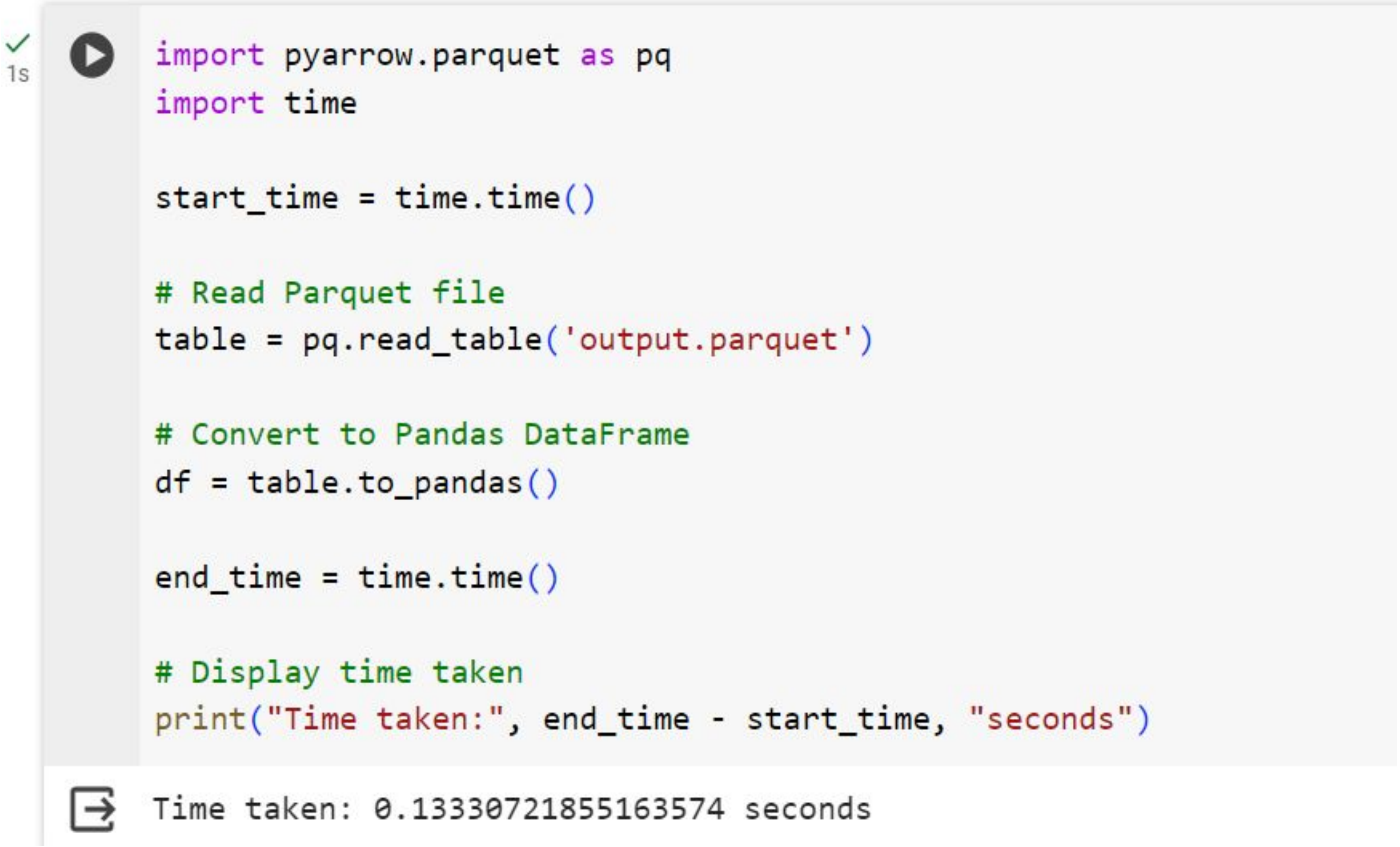
```python
import os

# Specify the path to your csv file
csv_file_path = '/content/migrationdata.csv'

# Get the size of the csv file in bytes
file_size_bytes = os.path.getsize(csv_file_path)

# Convert bytes to kilobytes and megabytes
file_size_kb = file_size_bytes / 1024
file_size_mb = file_size_kb / 1024

print(f"Size of csv file: {file_size_bytes} bytes")
print(f"Size of csv file in kilobytes: {file_size_kb:.2f} KB")
print(f"Size of csv file in megabytes: {file_size_mb:.2f} MB")
```

```
Size of csv file: 26644216 bytes
Size of csv file in kilobytes: 26019.74 KB
Size of csv file in megabytes: 25.41 MB
```



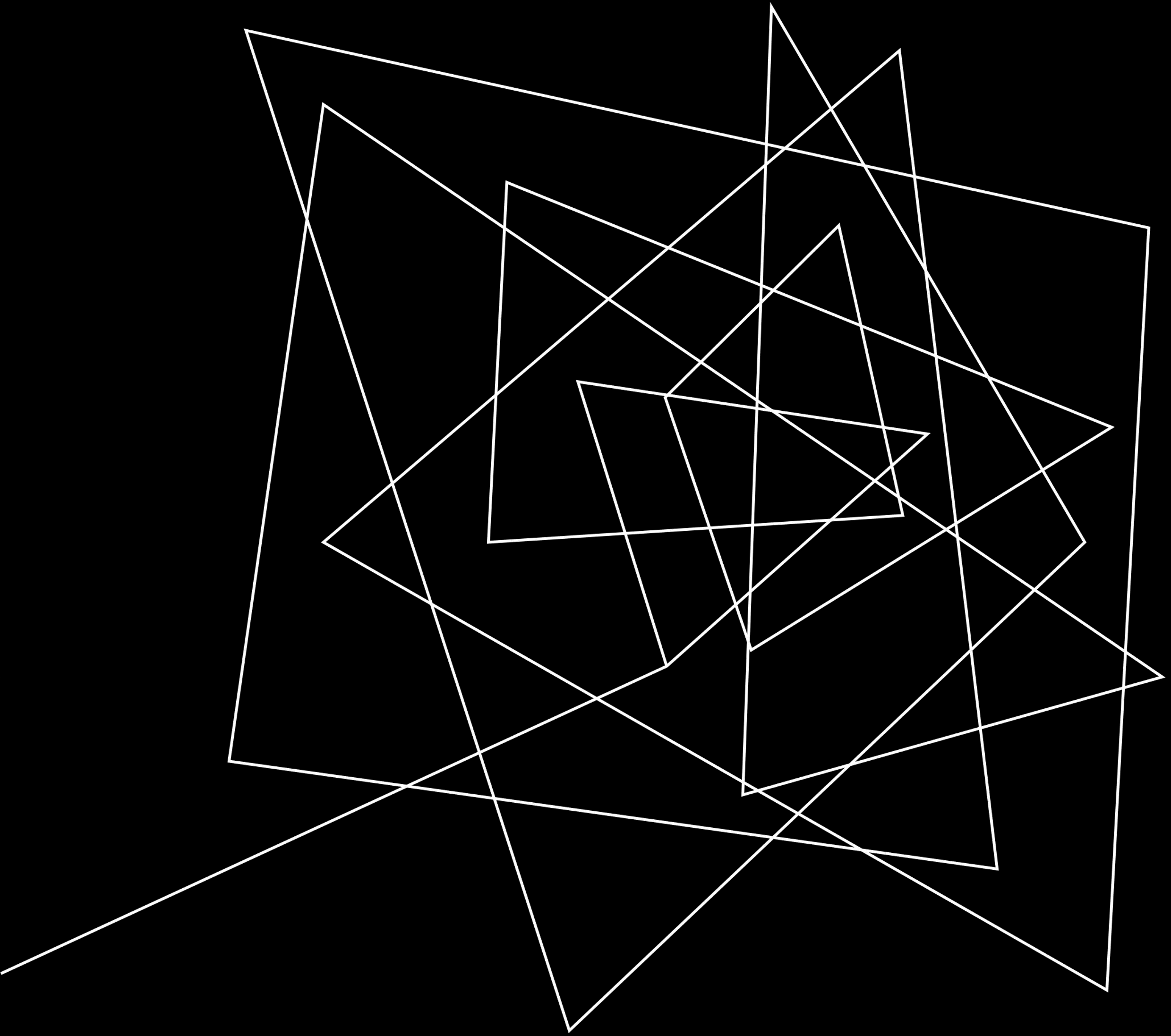
```python
import os

# Specify the path to your Parquet file
parquet_file_path = 'output.parquet'

# Get the size of the Parquet file in bytes
file_size_bytes = os.path.getsize(parquet_file_path)

# Convert bytes to kilobytes and megabytes
file_size_kb = file_size_bytes / 1024
file_size_mb = file_size_kb / 1024

print(f"Size of parquet file: {file_size_bytes} bytes")
print(f"Size of parquet file in kilobytes: {file_size_kb:.2f} KB")
print(f"Size of parquet file in megabytes: {file_size_mb:.2f} MB")
```

```
Size of parquet file: 639615 bytes
Size of parquet file in kilobytes: 624.62 KB
Size of parquet file in megabytes: 0.61 MB
```



```python
import pyarrow.parquet as pq
import time

start_time = time.time()

# Read Parquet file
table = pq.read_table('output.parquet')

# Convert to Pandas DataFrame
df = table.to_pandas()

end_time = time.time()

# Display time taken
print("Time taken:", end_time - start_time, "seconds")
```

```
Time taken: 0.1330721855163574 seconds
```



ORC (OPTIMIZED ROW COLUMNAR)

OPTIMIZED ROW COLUMNAR(ORC)

Columnar Storage

Developed primarily for Apache Hive,a data warehouse infrastructure built on top of Hadoop

Predictive Pushdown

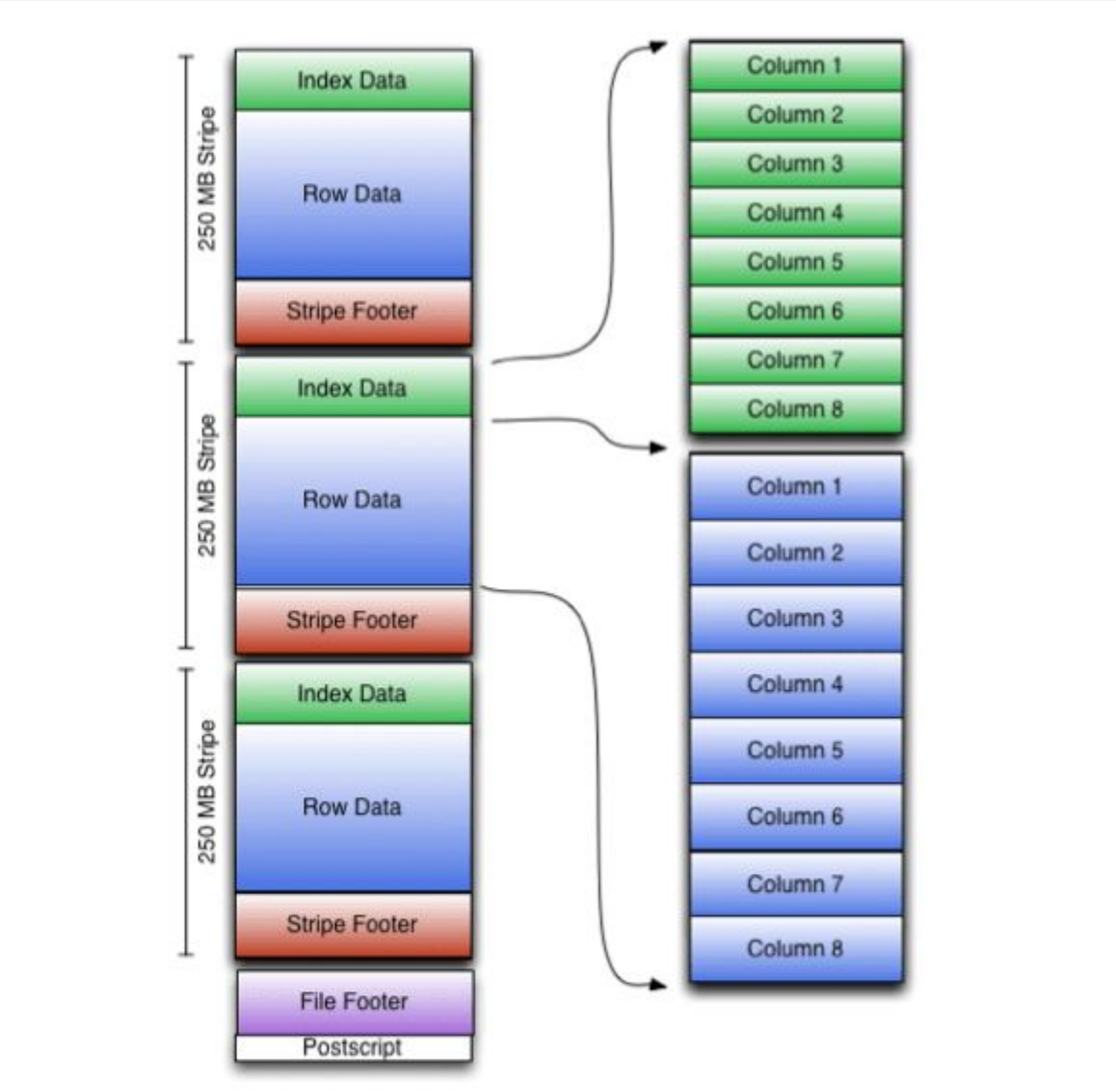
Filters are applied during data reading, reducing the data that needs to be processed.

Efficient Compression

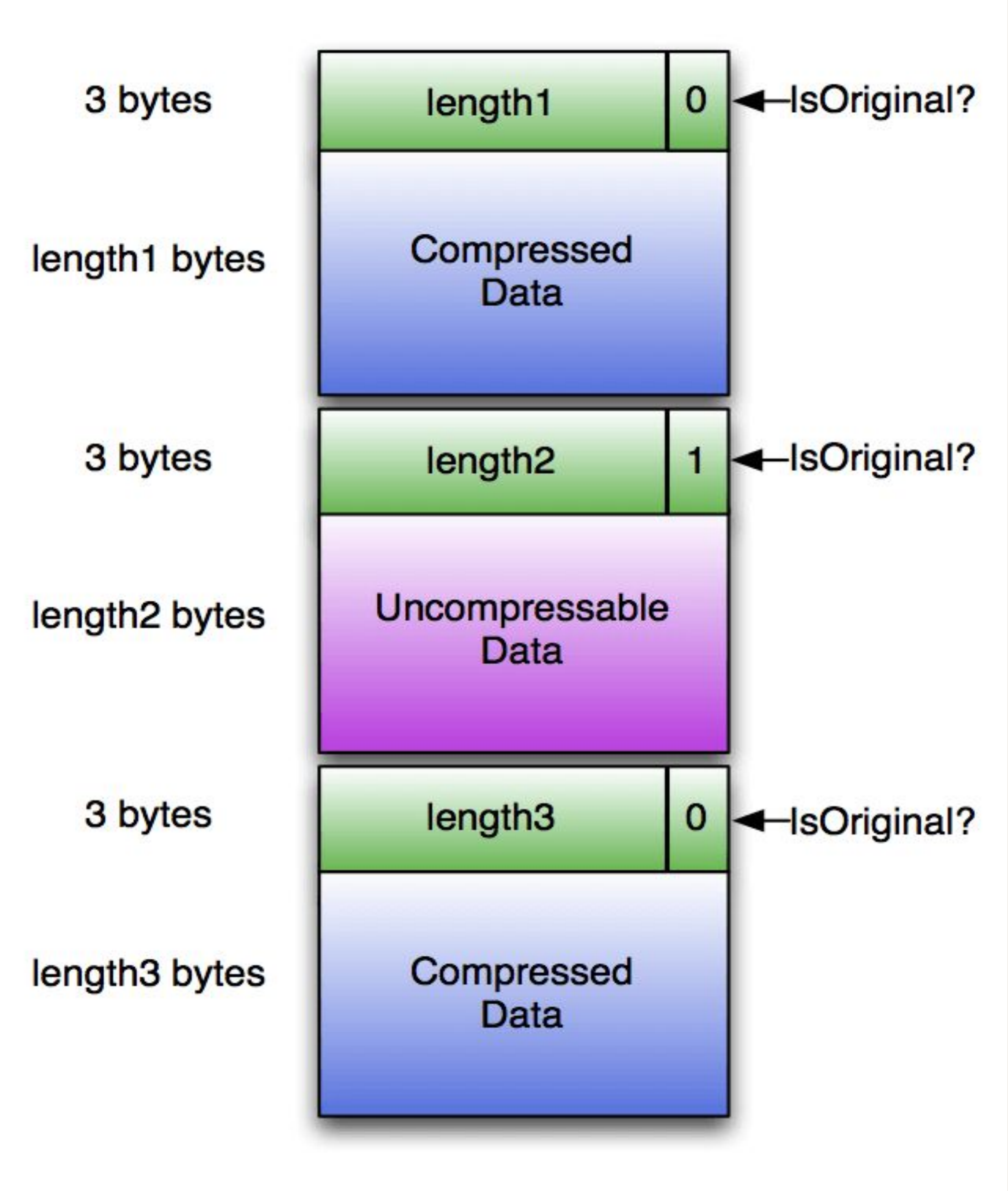
Compression options including Snappy,

Complex Type Support

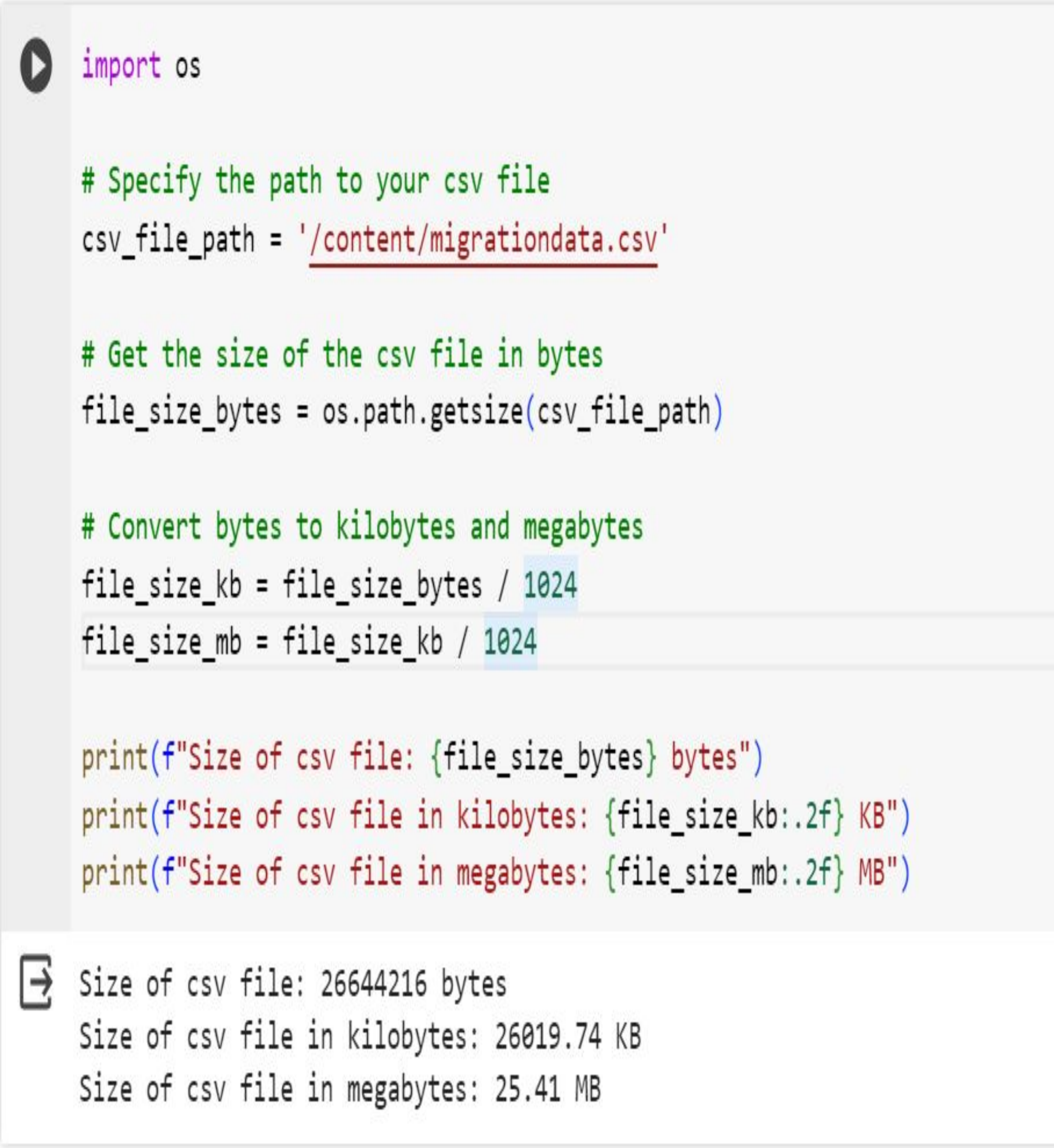
Including Datetime, struct, list, map, and union



ORC file structure

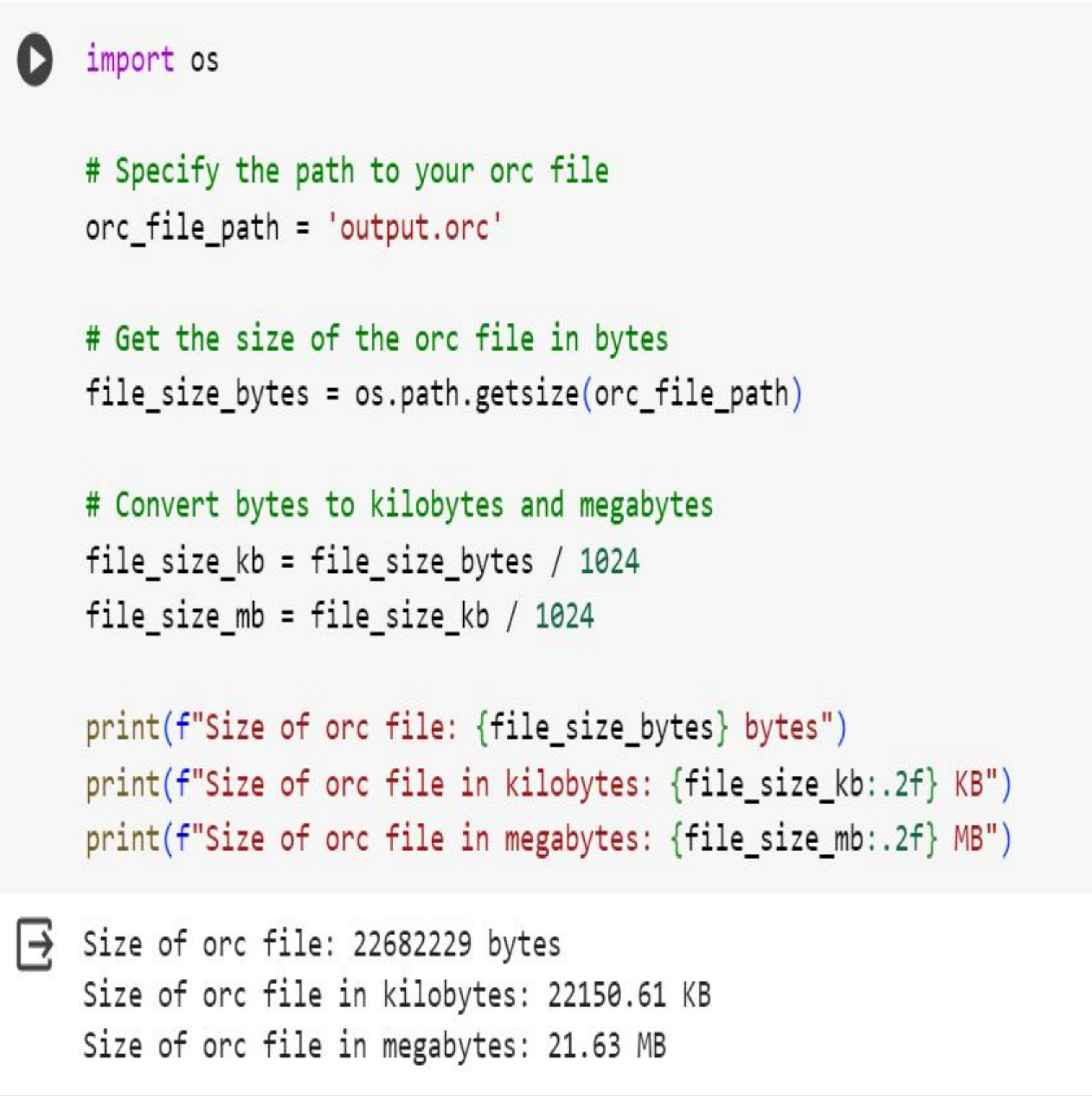


ORC compression chunk



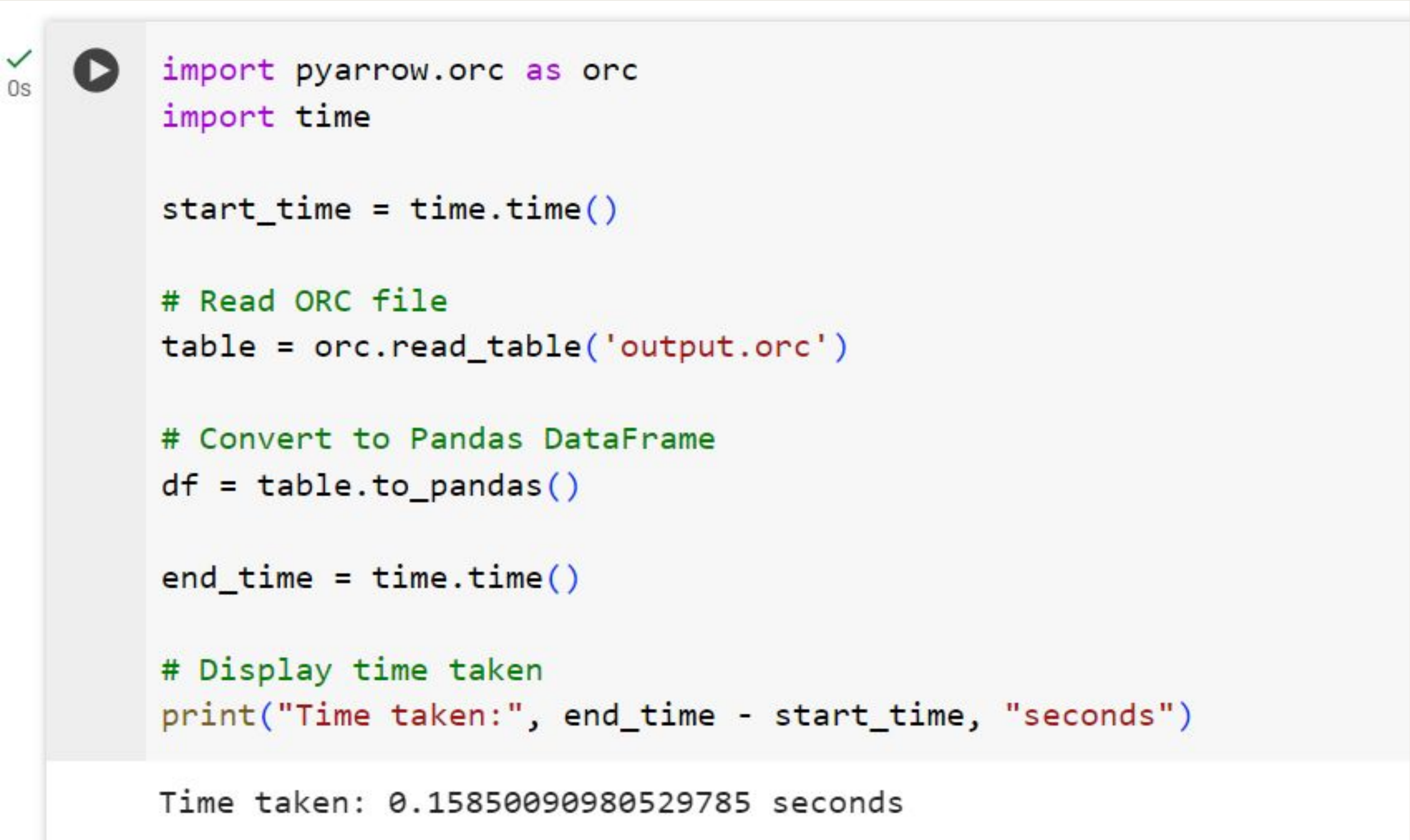
```python
import os

# Specify the path to your csv file
csv_file_path = '/content/migrationdata.csv'

# Get the size of the csv file in bytes
file_size_bytes = os.path.getsize(csv_file_path)

# Convert bytes to kilobytes and megabytes
file_size_kb = file_size_bytes / 1024
file_size_mb = file_size_kb / 1024

print(f"Size of csv file: {file_size_bytes} bytes")
print(f"Size of csv file in kilobytes: {file_size_kb:.2f} KB")
print(f"Size of csv file in megabytes: {file_size_mb:.2f} MB")
```

```
Size of csv file: 26644216 bytes
Size of csv file in kilobytes: 26019.74 KB
Size of csv file in megabytes: 25.41 MB
```



```python
import os

# Specify the path to your orc file
orc_file_path = 'output.orc'

# Get the size of the orc file in bytes
file_size_bytes = os.path.getsize(orc_file_path)

# Convert bytes to kilobytes and megabytes
file_size_kb = file_size_bytes / 1024
file_size_mb = file_size_kb / 1024

print(f"Size of orc file: {file_size_bytes} bytes")
print(f"Size of orc file in kilobytes: {file_size_kb:.2f} KB")
print(f"Size of orc file in megabytes: {file_size_mb:.2f} MB")
```

```
Size of orc file: 22682229 bytes
Size of orc file in kilobytes: 22150.61 KB
Size of orc file in megabytes: 21.63 MB
```



```python
import pyarrow.orc as orc
import time

start_time = time.time()

# Read ORC file
table = orc.read_table('output.orc')

# Convert to Pandas DataFrame
df = table.to_pandas()

end_time = time.time()

# Display time taken
print("Time taken:", end_time - start_time, "seconds")
```

```
Time taken: 0.15850090980529785 seconds
```

RECOMMENDATIONS

DATA CHARACTERISTICS

Choose CSV for simple, portable data, and Parquet/ORC for big data analytics.

PROCESSING NEEDS

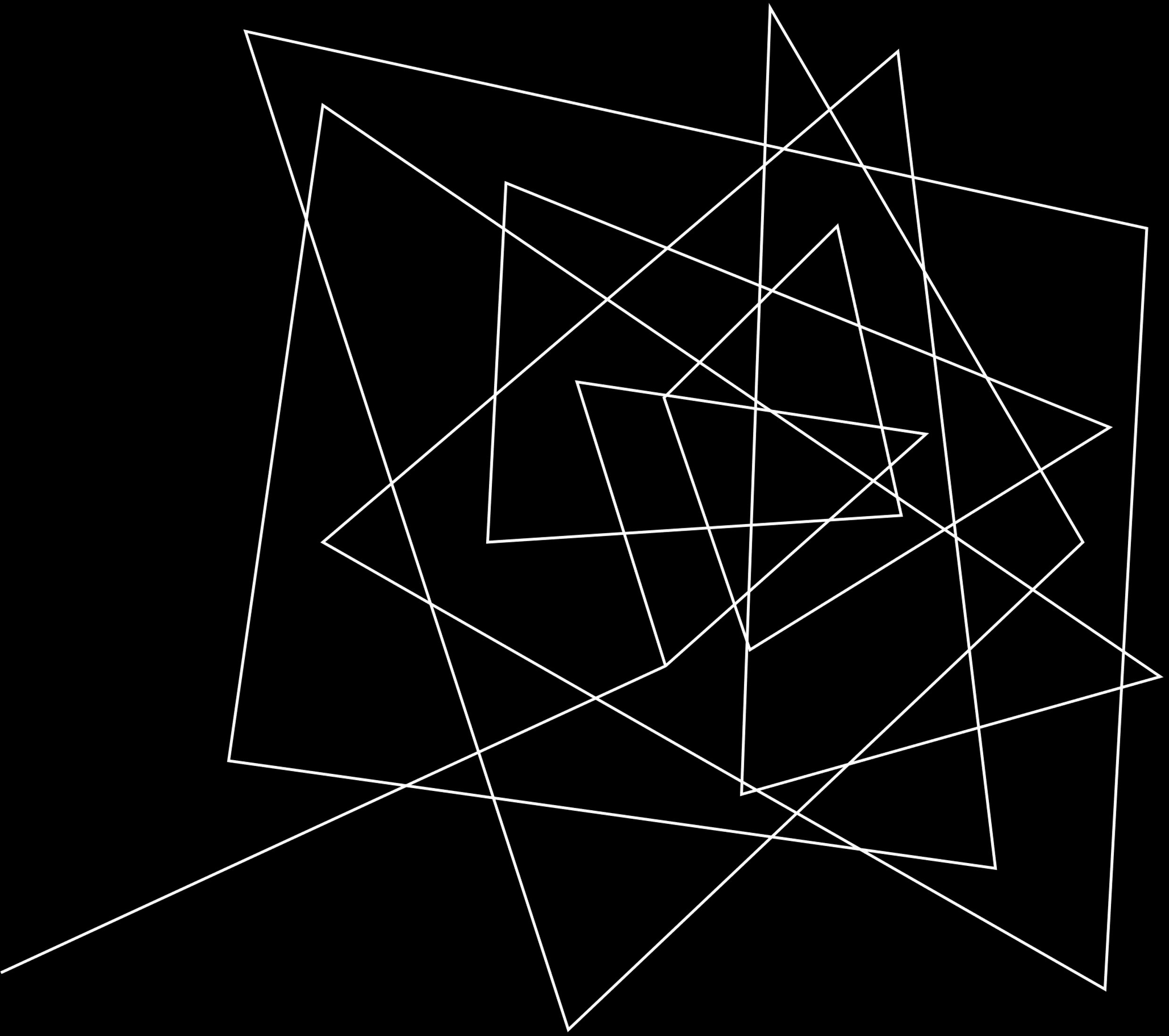
Select XLSX for business data with formulas & formatting Avro for Hadoop-based processing with evolving schemas.

ECOSYSTEM COMPATIBILITY

Consider compatibility with existing systems and tools, favoring formats widely supported in the organization's ecosystem.

FUTURE SCALABILITY

Selecting formats that can scale efficiently with the organization's future needs.

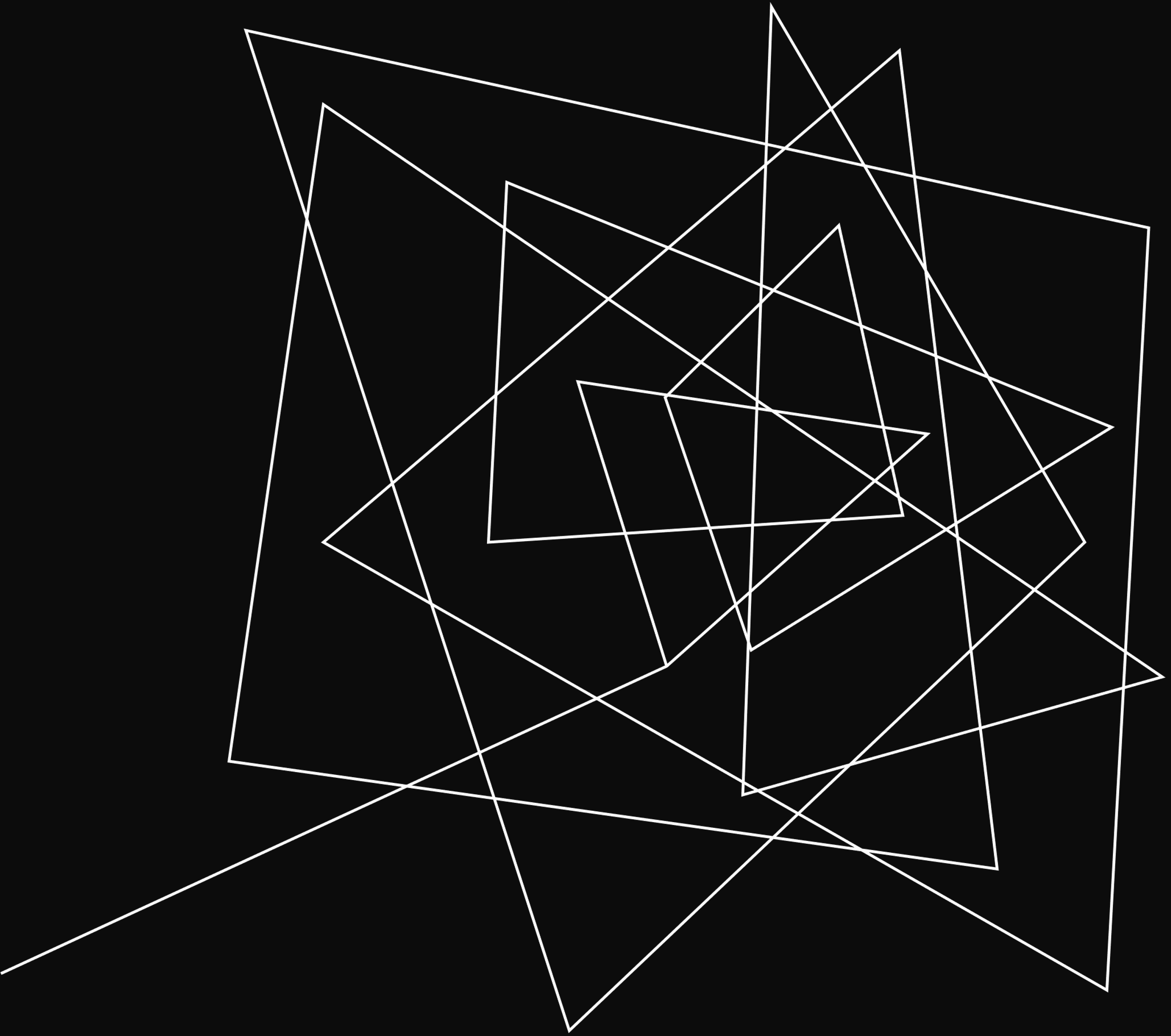


CONCLUSION

It's imperative to recognize the importance of selecting the right file format to optimize our operations.

Prioritizing readability with CSV, harnessing power of columnar storage with Parquet , etc.

Our choices profoundly impact our ability to derive insights and drive informed decisions from our data.



THANK YOU