Importing ARCOS Data with Dask

Last week, we used dask to play with a few datasets to get a feel for how dask works. In order to help us develop code that would run quickly, however, we worked with very small, safe datasets.

Today, we will continue to work with dask, but this time using much larger datasets. This means that (a) doing things incorrectly may lead to your computer crashing (So save all your open files before you start!), and (b) many of the commands you are being asked run will take several minutes each.

For familiarity, and so you can see what advantages dask can bring to your workflow, today we'll be working with the DEA ARCOS drug shipment database published by the Washington Post! However, to strike a balance between size and speed, we'll be working with a slightly thinned version that has only the last two years of data, instead of all six.

Exercise 1

Download the thinned ARCOS data from this link. It should be about 2GB zipped, 25 GB unzipped.

In [ ]:
```python
# Importing the libraries
import pandas as pd
import os
import warnings

warnings.filterwarnings("ignore")
```

In [ ]:
```python
# Reading the data with file chunk size of 100,000
chunk_size = 100_000
tsv_path = r"C:\Users\divya\OneDrive\Documents\Duke\MIDS\Semester 1\IDS 720 - PDS\1
chunks = []


for chunk in pd.read_csv(tsv_path, sep="\t", chunksize=chunk_size):
    chunks.append(chunk)

# Concatenating the chunks into a single DataFrame
df = pd.concat(chunks, axis=0)
```

Exercise 2

Our goal today is going to be to find the pharmaceutical company that has shipped the most opioids (`MME_Conversion_Factor * CALC_BASE_WT_IN_GM`) in the US.

When working with large datasets, it is good practice to begin by prototyping your code with a subset of your data. So begin by using `pandas` to read in the first 100,000 lines of the ARCOS data and write pandas code to compute the shipments from each shipper (the group that reported the shipment).

```
In [ ]: # Subsetting for top 100,000 rows
        dt_100k = df.head(100_000)
        # Creating a new column for the total opioid shipments
        dt_100k["Opioid_Shipment"] = (
            dt_100k["MME_Conversion_Factor"] * dt_100k["CALC_BASE_WT_IN_GM"]
        )
```

```
In [ ]: dt_100k.head(5)
```

Out[ ]:

	Unnamed: 0	REPORTER_DEA_NO	REPORTER_BUS_ACT	REPORTER_NAME	REPORTER_ADDI
0	0	PA0006836	DISTRIBUTOR	ACE SURGICAL SUPPLY CO INC	
1	9	PA0021179	DISTRIBUTOR	APOTHECA INC	
2	10	PA0021179	DISTRIBUTOR	APOTHECA INC	
3	16	PA0021179	DISTRIBUTOR	APOTHECA INC	
4	17	PA0021179	DISTRIBUTOR	APOTHECA INC	

5 rows × 46 columns

```
In [ ]: dt_100k.columns
```

```
Out[ ]: Index(['Unnamed: 0', 'REPORTER_DEA_NO', 'REPORTER_BUS_ACT', 'REPORTER_NAME',
               'REPORTER_ADDL_CO_INFO', 'REPORTER_ADDRESS1', 'REPORTER_ADDRESS2',
               'REPORTER_CITY', 'REPORTER_STATE', 'REPORTER_ZIP', 'REPORTER_COUNTY',
               'BUYER_DEA_NO', 'BUYER_BUS_ACT', 'BUYER_NAME', 'BUYER_ADDL_CO_INFO',
               'BUYER_ADDRESS1', 'BUYER_ADDRESS2', 'BUYER_CITY', 'BUYER_STATE',
               'BUYER_ZIP', 'BUYER_COUNTY', 'TRANSACTION_CODE', 'DRUG_CODE', 'NDC_NO',
               'DRUG_NAME', 'QUANTITY', 'UNIT', 'ACTION_INDICATOR', 'ORDER_FORM_NO',
               'CORRECTION_NO', 'STRENGTH', 'TRANSACTION_DATE', 'CALC_BASE_WT_IN_GM',
               'DOSAGE_UNIT', 'TRANSACTION_ID', 'Product_Name', 'Ingredient_Name',
               'Measure', 'MME_Conversion_Factor', 'Combined_Labeler_Name',
               'Revised_Company_Name', 'Reporter_family', 'dos_str', 'date', 'year',
               'Opioid_Shipment'],
              dtype='object')
```

```
In [ ]: # Grouping by the reporter name and summing the opioid shipments
        df_sum = dt_100k.groupby("REPORTER_NAME")["Opioid_Shipment"].sum().reset_index()
```

```
In [ ]: df_sum.head(5)
```

Out[ ]:

	REPORTER_NAME	Opioid_Shipment
0	ACE SURGICAL SUPPLY CO INC	0.605400
1	AMERICAN SALES COMPANY	3432.058005
2	AMERISOURCEBERGEN DRUG CORP	34561.394892
3	APOTHECA INC	23.913300
4	BLOODWORTH WHOLESALE DRUGS	1782.827325

Exercise 3

Now let's turn to dask. Re-write your code for dask, and calculate the total shipments by reporting company. Remember:

- Activate a conda environment with a clean dask installation.
- Start by spinning up a distributed cluster.
- Dask won't read compressed files, so you have to unzip your ARCOS data.
- Start your cluster in a cell all by itself since you don't want to keep re-running the "start a cluster" code.

If you need to review dask basic code, check here.

As you run your code, make sure to click on the Dashboard link below where you created your cluster:

Among other things, the bar across the bottom should give you a sense of how long your task will take:

(For context, my computer (which has 10 cores) only took a couple seconds. My computer is fast, but most computers should be done within a couple minutes, tops).

In [ ]:
```python
from dask.distributed import Client

client = Client()

# Reading the data using Dask and calculating total shipments by reporting company
import dask.dataframe as dd

dask_data = dd.read_csv(
    tsv_path,
    sep="\t",
    assume_missing=True,
    dtype={
        "ORDER_FORM_NO": "object",
        "REPORTER_ADDL_CO_INFO": "object",
        "REPORTER_ADDRESS2": "object",
        "ACTION_INDICATOR": "object",
        "NDC_NO": "object",
        "UNIT": "object",
    },
)
```

In [ ]:
```python
dask_data["Opioid_Shipment"] = (
    dask_data["MME_Conversion_Factor"] * dask_data["CALC_BASE_WT_IN_GM"]
)
```

In [ ]:
```python
total_shipments = dask_data.groupby("REPORTER_NAME")["Opioid_Shipment"].sum().compu
```

In [ ]:
```python
total_shipments.head(5)
```

Out[ ]:
```
REPORTER_NAME
ACE SURGICAL SUPPLY CO INC      2.421600e+00
AMERICAN SALES COMPANY          2.798722e+04
AMERISOURCEBERGEN DRUG CORP     3.999211e+06
APOTHECA INC                    1.935657e+02
BLOODWORTH WHOLESALE DRUGS      1.048958e+04
Name: Opioid_Shipment, dtype: float64
```

Exercise 4

Now let's calculate, *for each state*, what company shipped the most pills?

Note you will quickly find that you can't sort in dask -- sorting in parallel is *really* tricky! So you'll have to work around that. Do what you need to do on the big dataset first, then compute it all so you get it as a regular pandas dataframe, then finish.

Does this seem like a situation where a single company is responsible for the opioid epidemic?

In [ ]:
```
total_shipments_state = (
    dask_data.groupby(["REPORTER_NAME", "BUYER_STATE"])["Opioid_Shipment"]
    .sum()
    .compute()
    .reset_index()
)
```

In [ ]:
```
total_shipments_state.head(10)
```

Out[ ]:

	REPORTER_NAME	BUYER_STATE	Opioid_Shipment
0	ACE SURGICAL SUPPLY CO INC	MA	0.605400
1	AMERICAN SALES COMPANY	CT	3352.357095
2	AMERICAN SALES COMPANY	DC	68.773440
3	AMERICAN SALES COMPANY	DE	248.092920
4	AMERICAN SALES COMPANY	MA	2786.232420
5	AMERICAN SALES COMPANY	MD	6528.058470
6	AMERICAN SALES COMPANY	NJ	966.869205
7	AMERICAN SALES COMPANY	NY	2433.753405
8	AMERICAN SALES COMPANY	PA	5962.887300
9	AMERICAN SALES COMPANY	RI	734.395605

In [ ]:
```
df_test = (
    total_shipments_state.groupby("BUYER_STATE")["Opioid_Shipment"].max().reset_ind
)
```

In [ ]:
```
out = pd.merge(total_shipments_state, df_test, on=["BUYER_STATE", "Opioid_Shipment"
```

In [ ]:
```
out.head(10)
```

Out[ ]:

	REPORTER_NAME	BUYER_STATE	Opioid_Shipment
0	AMERISOURCEBERGEN DRUG CORP	KY	193379.917570
1	AMERISOURCEBERGEN DRUG CORP	TN	302713.057219
2	CARDINAL HEALTH 110, LLC	NY	566836.079217
3	LOUISIANA WHOLESALE DRUG CO	LA	117206.957966
4	MCKESSON CORPORATION	AR	97547.962187
5	MCKESSON CORPORATION	AZ	343528.893987
6	MCKESSON CORPORATION	CA	801977.350276
7	MCKESSON CORPORATION	CO	183853.423334
8	MCKESSON CORPORATION	HI	62476.302371
9	MCKESSON CORPORATION	ID	52476.842788

In [ ]:
```
out["REPORTER_NAME"].value_counts()
```

Out[ ]:
```
REPORTER_NAME
MCKESSON CORPORATION               24
CARDINAL HEALTH                    14
WALGREEN CO                         6
AMERISOURCEBERGEN DRUG CORP         4
CARDINAL HEALTH 110, LLC            1
LOUISIANA WHOLESALE DRUG CO         1
DAKOTA DRUG                         1
DROGUERIA BETANCES, LLC             1
CARDINAL HEALTH P.R. 120, INC.      1
AMERISOURCEBERGEN DRUG              1
MCKESSON DRUG COMPANY               1
Name: count, dtype: int64[pyarrow]
```

> Since Mckensson Corporation is the largest pharmaceutical distributor in 24 out of the 54 listed states in the dataset, with a very high share in the opioid shipments, it is safe to say that a single company is responsible for the opioid epidemic.

Exercise 5

Now go ahead and try and re-do the chunking you did by hand for your project (with this 2 years of data) -- calculate, for each year, the total morphine equivalents sent to each county in the US.

In [ ]:
```
df6 = (
    (
        dask_data.groupby(["BUYER_STATE", "BUYER_COUNTY", "year"])[
            "Opioid_Shipment"
        ].sum()
    )
    .compute()
    .reset_index()
)
```

In [ ]:
```
df6.head(10)
```

Out[ ]:

	BUYER_STATE	BUYER_COUNTY	year	Opioid_Shipment
0	AL	AUTAUGA	2011.0	4181.623992
1	AL	AUTAUGA	2012.0	4346.607187
2	AL	BARBOUR	2011.0	1317.385035
3	AL	BARBOUR	2012.0	1385.788223
4	AL	BUTLER	2011.0	1186.474868
5	AL	BUTLER	2012.0	1306.989960
6	AL	CALHOUN	2011.0	11047.729564
7	AL	CHAMBERS	2012.0	2819.326193
8	AL	CHEROKEE	2012.0	3005.192123
9	AL	COFFEE	2011.0	3023.785815

Exercise 6

Now, re-write your opioid project's initial opioid import using dask. Each person on your team should create a NEW branch to try this. The person who wrote the initial chunking code can help everyone else understand what they did originally and the data, but everyone should write their own code.

WARNING: You will probably run into a lot of type errors (depending on how the ARCOS data has changed since last year). With real world messy data one of the biggest problems with dask is that it struggles if halfway through dataset it discovers that the column it *thought* was floats contains text. That's why, in the dask reading, I specified the column type for so many columns as `objects` explicitly. Then, because occasionally there data cleanliness issues, I had to do some converting data types by hand.

Group 8 - https://github.com/MIDS-at-Duke/opioid-2023-group-8-final-opioid/tree/dask_branching_ds655

Link to Github Branch