RAPIDS cuDF is an open-source Python library for GPU-accelerated DataFrames. cuDF provides a Pandas-like API that allows data engineers, analysts, and data engineers can use perform data manipulation and analysis tasks on large datasets and time series data using the power of NVIDIA GPUs allowing for faster data processing and analysis.

Getting started with cuDF is straightforward, especially if you have experience using Python and libraries like Pandas. While both cuDF and Pandas offer similar APIs for data manipulation, there are specific types of problems in which cuDF can provide significant performance improvements over Pandas, including large scale datasets, data preprocessing and engineering, real-time analytics, and, of course, parallel processing. The bigger the dataset, the greater the performance benefits.

### Installation

- **Anaconda**
  ```bash
  $ conda create -n rapids-23.02 -c rapidsai -c conda-forge -c nvidia rapids=23.02 python=3.10 cudatoolkit=11.8
  
  ```

- **PIP Install**
  ```bash
  ```

### Reading Dataset

The cupy is similar to NumPy API, which runs on GPU. We will import libraries and use cupy to generate random numbers to create cudf data frame.

```python
import cupy
import cudf
df = cudf.DataFrame(
    {  
    "A": cupy.random.randint(5, 35, size=1000),  
    "B": cupy.random.randint(300, 400, size=1000),  
    "C": cupy.random.randint(1, 50, size=1000),  
    }
)
df.head()
```

### Writing Dataframes

- **Saving CSV file**
  ```python
df.to_csv("processed_california_housing.csv")
```

- **Other formats**
  - `df.to_json()
  - df.to_parquet()
  - df.to_hdf()
  - df.to_feather()`

- **Saving CSV file**
  ```python
df.to_csv("processed_california_housing.csv")
```

### Apply Function

- **Concat & Join**
  ```python
def add_five(num):
    return num + 5

df_c = df.copy()
df_final = cudf.concat([df, df_c])
df.merge(df_c, on="B", how="left")
```

- **Grouping by B columns and summimg**
  ```python
df.groupby("B").sum()
```

- **Grouping and applying statistical functions to specific columns**
  ```python
df.groupby("B").agg({'A': 'max', 'C': 'mean'})
```

### Selection

- **Geting Data**
  ```python
df["B"][0:4]
```

- **Selection by Label**
  ```python
df.loc[0:4, ['A', 'B']]
```

- **Selection by Position**
  ```python
df.iloc[0:4, 0:2]
```

- **Boolean Indexing**
  ```python
df[df.A > 5]
```

- **Query**
  ```python
df.query("B == 344")
```

### Converting Data Representation

- **Converting Data Representation**
  ```python
df_pd = df.to_pandas()
df_numpy = df.to_numpy()
df_arrow = to_arrow()
```

### Subscribe to KDnuggets News

- **Subscribe to KDnuggets News**
  ```python
  def add_five(num):
    return num + 5

df_c = df.copy()
df_final = cudf.concat([df, df_c])
df.merge(df_c, on="B", how="left")
```