Python Assignment -3

1.what is the process for loading a dataset from an external sources?

Ans. You can integrate any database with Exasol as long as the external source supports a JDBC interface. You need to upload the corresponding JDBC driver into EXAoperation (see Manage JDBC Drivers) and then use the generic JDBC interface for `IMPORT` and `EXPORT`. By supporting native interfaces to Exasol and Oracle databases, we achieve even better performance.

You can load data using the `IMPORT` and `EXPORT` statements and combine that with further post-processing through subselects. You can even specify a statement instead of a table name for the external data source which is executed on that system, to for example, load a certain part of a table into Exasol.

This section describes how you can connect to sources with Exasol for importing or exporting of data.

2.How can we use pandas to read JSON files ?

Ans. Reading JSON Files using pandas

To read the files, we use read_json() function and through it, we pass the path to the JSON file we want to read. Once we do that, it returns a "DataFrame"(A table of rows and columns) that stores data. If we want to read a file that is located on remote servers then we pass the link to its location instead of a local path.

Example 1: Reading JSON file

- Python3

```
import pandas as pd
```

```
df = pd.read_json("FILE_JSON.json")
df.head()
```

Output:

```
   One  Two
0   60  110
1   60  117
2   60  103
3   45  109
4   45  117
5   60  102
```

Example 2: Creating JSON data and reading in dataframe

Here we will create JSON data and then create a dataframe through it using **pd.Dataframe() methods.**

- Python3

```
import pandas as pd


data = {
    "One": {
        "0": 60,
        "1": 60,
        "2": 60,
        "3": 45,
        "4": 45,
        "5": 60
```

```
    },
    "Two": {
       "0": 110,
       "1": 117,
       "2": 103,
       "3": 109,
       "4": 117,
       "5": 102
    }
 }


 df = pd.DataFrame(data)


 print(df)
```

Output:

```
   One  Two
0   60  110
1   60  117
2   60  103
3   45  109
4   45  117
5   60  102
```

3.Describe the significance of DASK?

Ans. The **DASK** was the first computer in Denmark. It was commissioned in 1955, designed and constructed by Regnecentralen, and began operation in September 1957. **DASK** is

an acronym for **D**ansk **A**ritmetisk **S**ekvens **K**alkulator or *Danish Arithmetic Sequence Calculator*. Regnecentralen almost didn't allow the name, as the word *dask* means "slap" in Danish. In the end however, it was named so as it fit the pattern of the name BESK, the Swedish computer which provided the initial architecture for DASK.

- DASK traces its origins to 1947 and a goal set by *Akademiet for de Tekniske Videnskaber* (*Academy for the Technical Sciences* or *Academy of Applied Sciences*), which was to follow the development of the modern computing devices. Initial funding was obtained through the Ministry of Defence (Denmark) as the Danish Military had been given a grant through the Marshall Plan for cipher machines for which the military saw no immediate need.
- Originally conceived to be a copy of BESK, the rapid advancement in the field allowed improvements to be made during the development such that in the end, it was not a copy of BESK. The DASK was a one-off design that took place in a villa. The machine became so big that the floor had to be reinforced to support its mass of 3.5 metric tons.
- DASK is notable for being the subject of one of the earliest ALGOL implementations, referred to as **DASK ALGOL**,[1] which counted Jørn Jensen and Peter Naur among its contributors.

Analysts often use tools like Pandas, Scikit-Learn, Numpy, and the rest of the Python ecosystem to analyze data on their personal computer. They like these tools because they are efficient, intuitive, and widely trusted. However, when they choose to apply their analyses to larger datasets, they find that these tools were not designed to scale beyond a single machine. And so, the analyst rewrites their computation using a more scalable tool, often in another language altogether. This rewrite process slows down discovery and causes frustration.Dask provides ways to scale Pandas, Scikit-Learn, and Numpy workflows more natively, with minimal rewriting. It integrates well with these tools so

that it copies most of their API and uses their data structures internally. Moreover, Dask is co-developed with these libraries to ensure that they evolve consistently, minimizing friction when transitioning from a local laptop, to a multi-core workstation, and then to a distributed cluster. Analysts familiar with Pandas/Scikit-Learn/Numpy will be immediately familiar with their Dask equivalents, and have much of their intuition carry over to a scalable context.

4.Describe the functions of dask?

Ans. Dask is composed of two parts:

1. **Dynamic task scheduling** optimized for computation. This is similar to *Airflow, Luigi, Celery, or Make*, but optimized for interactive computational workloads.
2. **"Big Data" collections** like parallel arrays, dataframes, and lists that extend common interfaces like *NumPy, Pandas, or Python iterators* to larger-than-memory or distributed environments. These parallel collections run on top of dynamic task schedulers.

Dask emphasizes the following virtues:

- **Familiar**: Provides parallelized NumPy array and Pandas DataFrame objects
- **Flexible**: Provides a task scheduling interface for more custom workloads and integration with other projects.
- **Native**: Enables distributed computing in pure Python with access to the PyData stack.
- **Fast**: Operates with low overhead, low latency, and minimal serialization necessary for fast numerical algorithms
- **Scales up**: Runs resiliently on clusters with 1000s of cores
- **Scales down**: Trivial to set up and run on a laptop in a single process

- **Responsive**: Designed with interactive computing in mind, it provides rapid feedback and diagnostics to aid humans

5.Describe cassandra's features?

Ans. Apache Cassandra is an open source, user-available, distributed, NoSQL DBMS which is designed to handle large amounts of data across many servers. It provides zero point of failure. Cassandra offers massive support for clusters spanning multiple datacentres.

There are some massive features of Cassandra. Here are some of the features described below:

1. **Distributed:**
   Each node in the cluster has has same role. There's no question of failure & the data set is distributed across the cluster but one issue is there that is the master isn't present in each node to support request for service.
2. **Supports replication & Multi data center replication:**
   Replication factor comes with best configurations in cassandra. Cassandra is designed to have a distributed system, for the deployment of large number of nodes for across multiple data centers and other key features too.
3. **Scalability:**
   It is designed to r/w throughput, Increase gradually as new machines are added without interrupting other applications.
4. **Fault-tolerance:**
   Data is automatically stored & replicated for fault-tolerance. If a node Fails, then it is replaced within no time.
5. **MapReduce Support:**
   It supports Hadoop integration with MapReduce support.Apache Hive & Apache Pig is also supported.
6. **Query Language:**
   Cassandra has introduced the CQL (Cassandra Query Language). Its a simple interface for accessing the Cassandra.

Cassandra Query Language (CQL) :

CQL has simple interface for accessing the Cassandra, also an

alternative for the traditional SQL. CQL adds an abstraction layer to hide the implementation of structure & also provides the native syntax for collections

- Before the updates of versions of Cassandra, upto Cassandra 1.0, Cassandra wasn't row level consistent, which means inserting & updating the table. It may affect the same row that are processed at approximately the same time may affect the non-key columns in a inconsistent manner.
  Cassandra 1.1 solved this using row level isolation.
- Deletion of markers called the Tombstones (source Internet) are also known to causes performance degradation upto severe consequence levels.
- Cassandra, essentially a hybrid between a key-value & a organised tabular DBMS.Tables can be created, dropped and altered at run time without blocking updates & queries.
- A column family called table represents a RDBMS. Each row is specifically identified by a row & key, name, value, timestamp etc. A table in Cassandra is a disturbed multi dimensional map monitored by a key. Further more applications are specified by a super column family.