Snowflake Cost Estimator

Introduction:

This documentation provides guidance on using the Snowflake Cost Estimator, a Python script designed to calculate the estimated costs associated with data storage and computation on the Snowflake platform. The cost estimator includes functions to calculate storage costs, compute costs, and generate an overall cost estimate based on user inputs.

Prerequisites:

Before using the Snowflake Cost Estimator, ensure the following prerequisites are met:

- Python installed (version 3.9 or higher)
- Knowledge of Snowflake edition types and warehouse sizes.
- A clear understanding of the data loading requirements and storage preferences.

Setting Up Snowflake Cost Estimator:

To use the Snowflake Cost Estimator, follow these steps:

- Open your Python IDE (e.g., PyCharm).
- Install the necessary dependencies by running the following command in the terminal:

```
pip install pandas
```

Importing necessary libraries:

```
import pandas as pd
```

Pandas library allows to organize and present the cost estimates and requirements in a structured format, making it easier to understand and analyse the results.

Estimating Storage and Compute Costs:

1. Estimating Storage Cost:

The calculate_storage_cost function calculates the storage cost based on the initial data size, storage type (on-demand or capacity), and the number of months for which the storage is required.

- Convert the initial data size from the user-specified unit to terabytes (TB).
- Determine the storage cost based on the storage type.
- For 'on-demand' storage, calculate the cost as the product of data size, a constant factor (40), and the number of months.
- For 'capacity' storage, calculate the cost as the product of data size, a constant factor (23), and the number of months.
- If the provided storage_type is neither 'on-demand' nor 'capacity', a ValueError is raised to choose a valid storage type.
- Return the calculated storage cost.

```
def calculate_storage_cost(initial_data_size_tb, storage_type, no_of_months):
    # Convert initial data size to TB
    data_size_tb = initial_data_size_tb/5
    if storage_type == 'on-demand':
        storage_cost_tb = data_size_tb * 40 * no_of_months
    elif storage_type == 'capacity':
        storage_cost_tb = data_size_tb * 23 * no_of_months
    else:
        raise ValueError("Invalid storage type. Please choose 'on-demand' or 'capacity'.")
    return storage_cost_tb
```

Variable Description:

initial_data_size_tb: The initial data size in terabytes (TB).

storage_type: The type of storage, either 'on-demand' or 'capacity'.

no_of_months: The number of months for which storage is required.

storage_cost_tb: The estimated total storage cost in terabytes (TB).

2. Estimating Compute Cost:

The calculate_compute_cost function computes the cost associated with data loading and compute requirements. It considers factors such as Snowflake edition type, the number of departments, warehouse size for data loading, and the daily and monthly usage of compute resources.

Data Loading Requirements:

- Determine the credit per hour based on the specified warehouse size.
- Determine the cost per credit based on the edition type.
- Calculate the data loading requirements based on the determined credit per hour, number of days, and hours per day for data loading.

```
def calculate_compute_cost(edition_type,num_departments,warehouse_size_dl,no_of_days_dl,
no_of_hours_per_day_dl,department_data):
    # Determine credit per hour based on warehouse size
    if warehouse_size_dl == 'x-small':
        credit_per_hour_dl = 1.0
    elif warehouse_size_dl == 'small':
        credit_per_hour_dl = 2.0
    elif warehouse_size_dl == 'medium':
        credit_per_hour_dl = 4.0
    elif warehouse_size_dl == 'large':
        credit_per_hour_dl = 8.0
    elif warehouse_size_dl == 'x-large':
        credit_per_hour_dl = 16.0
    elif warehouse_size_dl == '2x-large':
        credit_per_hour_dl = 32.0
    else:
        raise ValueError("Invalid warehouse size. Please choose 'x-small','small','medium','large',
        'x-large', '2x-large'")
```

```
# Determine cost per credit based on edition
if edition_type == "Standard":
    cost_per_credit = 2
elif edition_type == "Enterprise":
    cost_per_credit = 3
elif edition_type == "Business Critical":
    cost_per_credit = 4
else:
    raise ValueError("Invalid Edition type. Please enter the correct edition type")
data_loading_req = credit_per_hour_dl * no_of_days_dl * no_of_hours_per_day_dl
data_loading_req_edt = data_loading_req * cost_per_credit
```

Compute Requirements:

- Iterate through each department, asking the user for information such as department name, warehouse size, hours per day, and days per month.
- Calculate the compute requirements for each department based on the credit per hour, hours per day, and days per month.
- Calculate the total compute requirements by summing up the individual department requirements.
- Calculate the total compute cost by multiplying the sum of data loading and total compute requirements with the cost per credit.
- Return the calculated compute cost, data loading requirements, and data loading requirements with edition cost.

```
total_compute_req = 0
for _ in range(num_departments):
    dept_name = input("Enter the department name: ")
    warehouse_size_cr = input("Enter the warehouse size (x-small/small/medium/large/x-large/2x-large)
    : ")
    no_of_hours_per_day_cr = int(input(f"Enter the number of hours per day for {dept_name}: "))
    no_of_days_per_month_cr = int(input(f"Enter the number of days per month for {dept_name}: "))
    if warehouse_size_cr == 'x-small':
        credit_per_hour_cr = 1.0
    elif warehouse_size_cr == 'small':
        credit_per_hour_cr = 2.0
    elif warehouse_size_cr == 'medium':
        credit_per_hour_cr = 4.0
    elif warehouse_size_cr == 'large':
        credit_per_hour_cr = 8.0
    elif warehouse_size_cr == 'x-large':
        credit_per_hour_cr = 16.0
    elif warehouse_size_cr == '2x-large':
        credit_per_hour_cr = 32.0
    else:
        raise ValueError("Invalid warehouse size. Please choose 'x-small','small','medium','large',
        'x-large','2x-large'")
```

```
dept_compute_req = credit_per_hour_cr * no_of_days_per_month_cr * no_of_hours_per_day_cr
dept_compute_req_edt = dept_compute_req * cost_per_credit
total_compute_req += dept_compute_req
department_data.append({'Department Name': dept_name, 'Credits consumed Individually':
dept_compute_req,'Individual Compute Cost': dept_compute_req_edt})
```

```
sam_compute_cost_tb =  (data_loading_req + total_compute_req)
compute_cost_tb = sam_compute_cost_tb * cost_per_credit
return compute_cost_tb, data_loading_req, data_loading_req_edt
```

Variable Description:

edition_type: The edition type, which can be "Standard," "Enterprise," or "Business Critical."

num_departments: The number of departments in the compute requirements.

warehouse_size_dl: The warehouse size for data loading.

no_of_days_dl: The number of days for data loading requirements.

no_of_hours_per_day_dl: The number of hours per day for data loading requirements.

department_data: A list to store department-specific compute requirements.

compute_cost_tb: The estimated total compute cost.

Estimating Overall Cost:

The generate_cost_estimate function serves as the main entry point for generating a comprehensive cost estimate. It takes user inputs for various parameters like initial data size, edition type, warehouse size for data loading, storage type, number of months for storage, and department-specific compute requirements. It combines the results from the storage and compute cost calculations to provide an overall cost estimate.

- Utilize the calculate_storage_cost function to estimate the storage cost based on the provided input parameters.
- Utilize the calculate_compute_cost function to estimate the compute cost based on the provided input parameters.
- Sum up the storage cost and compute cost to obtain the total annual cost.
- Create pandas DataFrame to organize and present the cost details in a tabular format.
- Concatenate the Data Loading requirements and Compute requirements DataFrames to create a combined summary DataFrame.
- Return the DataFrames containing cost estimates and requirements.

- Get all the inputs required from the user.

```
def generate_cost_estimate():
    print("Please Enter only uncompressed data")
    initial_data_size_tb = float(input("Enter Initial Data Size (TB): "))
    edition_type = input("Enter the edition type(Standard/Enterprise/Business Critical): ")
    warehouse_size_dl = input("Enter Warehouse Size for Data Loading(x-small/small/medium/large/x-large/
    2x-large): ")
    no_of_days_dl = int(input("Enter Number of Days for Data Loading Req: "))
    no_of_hours_per_day_dl = int(input("Enter Number of Hours per Day for Data Loading Req: "))
    storage_type = input("Enter Storage Type (on-demand/capacity): ")
    no_of_months = int(input("Enter Number of Months for Storage: "))
    num_departments = int(input("Enter the number of departments: "))
```

```
department_data = []
storage_cost_tb = calculate_storage_cost(initial_data_size_tb, storage_type, no_of_months)
compute_cost_tb, data_loading_req, data_loading_req_edt = calculate_compute_cost(edition_type,
num_departments,warehouse_size_dl, no_of_days_dl,no_of_hours_per_day_dl,department_data)
total_cost_tb = storage_cost_tb + compute_cost_tb
cost_df = pd.DataFrame({
    'No of months': no_of_months,
    'Storage Cost (TB)': [storage_cost_tb],
    'Compute Cost (TB)': [compute_cost_tb],
    'Total Annual Cost (TB)': [total_cost_tb]
})
compute_req_df = pd.DataFrame(department_data)
data_loading_df = pd.DataFrame({
    'Department Name': ['Data Loading'],
    'Credits consumed Individually': [data_loading_req],
    'Individual Compute Cost': [data_loading_req_edt]
})
combined_df = pd.concat([data_loading_df, compute_req_df], ignore_index=True)
return cost_df, compute_req_df, data_loading_df, combined_df
```

The function outputs two DataFrames: cost_df containing cost details and combined_df summarizing data loading and compute requirements. The results are then displayed in tabular format.

```
cost_estimate_df,compute_req_df,data_loading_df, combined_df = generate_cost_estimate()
print(cost_estimate_df.to_string(index=False))
print("\nRequirements:")
print(combined_df.to_string(index=False))
```

Sample Input:

```
Please Enter only uncompressed data
Enter Initial Data Size (TB): 325
Enter the edition type(Standard/Enterprise/Business Critical): Standard
Enter Warehouse Size for Data Loading(x-small/small/medium/large/x-large/2x-large): small
Enter Number of Days for Data Loading Req: 31
Enter Number of Hours per Day for Data Loading Req: 24
Enter Storage Type (on-demand/capacity): capacity
Enter Number of Months for Storage: 1
Enter the number of departments: 3
Enter the department name: Finance
Enter the warehouse size (x-small/small/medium/large/x-large/2x-large): large
Enter the number of hours per day for Finance: 9
Enter the number of days per month for Finance: 20
Enter the department name: Sales
Enter the warehouse size (x-small/small/medium/large/x-large/2x-large): medium
Enter the number of hours per day for Sales: 16
Enter the number of days per month for Sales: 20
Enter the department name: Complex Query Users
Enter the warehouse size (x-small/small/medium/large/x-large/2x-large): 2x-large
Enter the number of hours per day for Complex Query Users: 2
Enter the number of days per month for Complex Query Users: 4
```

Output:

The output displays a summary of the estimated costs, including storage cost, compute cost, and the total annual cost. Additionally, it provides a detailed breakdown of department-specific data loading and compute requirements.

```
 No of months  Storage Cost (TB)  Compute Cost (TB)  Total Annual Cost (TB)
            1             1495.0             8928.0                 10423.0

Requirements:
     Department Name  Credits consumed Individually  Individual Compute Cost
        Data Loading                         1488.0                   2976.0
             Finance                         1440.0                   2880.0
               Sales                         1280.0                   2560.0
 Complex Query Users                          256.0                    512.0
```

Conclusion:

The Snowflake Cost Estimator provides a valuable tool for users seeking to estimate the financial implications of utilizing Snowflake for data storage and computation. In conclusion, the Snowflake Cost Estimator empowers users to make informed decisions regarding resource allocation and budgeting within the Snowflake platform.