# ASSIGNMENT

**1.** You are working as a Data Engineer in ABC Company. You are dealing with a large dataset of customer transactions **(/public/trendytech/datasets/cust_transf.csv)** in HDFS, including information such as customer ID, purchase date, product id, and amount.

**A)** Design a caching mechanism using dataframes to enhance the performance of data retrieval for the following use cases:

**A.1)** Your marketing team wants to identify the top-selling products based on revenue for a given time period. The query is expected to be executed frequently, and the results need to be returned quickly. Design a caching strategy that efficiently retrieves the top-selling products by revenue.

Additionally, demonstrate the impact of caching by comparing the retrieval time for Top 10 best-selling products from start_date = "2023-05-01" to end_date = "2023-06-08" before and after implementing the caching strategy. [**Note** : Strategize your caching in such a way that the right Dataframes are cached at the right time for maximal performance gains]

**A.2)** Find the top 10 customers with maximum transaction amount for the same date range of start_date = "2023-05-01" to end_date = "2023-06-08"

**A.3)** Implement all of the above using Spark Table (Create an External Table).

**A.4)** Find the top 10 regular customers (having atleast one purchase in any month) who are eligible for a special offer. Also demonstrate the performance gains achieved by using persist.

**A.5)** Illustrate the difference in the performance results while using cache() Vs persist() with storage level MEMORY_AND_DISK_SER for the above query. Showcase the amount of cached data and the time taken

**A.6)** Demonstrate the changes in the performance of the query A.4 for the following persistent storage levels.

- MEMORY_ONLY
- MEMORY_ONLY_SER
- MEMORY_AND_DISK
- MEMORY_AND_DISK_SER
- DISK_ONLY

**B)** The customer service team frequently needs to access the transaction history of a specific customer to resolve any issues or provide personalized assistance. Design a caching mechanism that allows fast retrieval of a customer's transaction history.(hint: you can use user defined functions to pass customer_id to get the transaction details)

**C)** Empty the cached Dataframe and Spark Table to free up the resources.

**2.** Consider a scenario where you have a large dataset **(/public/trendytech/datasets/hotel_data.csv)** in HDFS. Design a caching mechanism using spark external tables to improve the query performance on this dataset. The dataset contains the following columns: booking_id, guest_name, checkin_date, checkout_date, room_type, and total_price.

**A)** Write a query to fetch the total count of hotel bookings in the hotel_bookings table and compare the duration it took to determine the impact of caching.

**B)** Calculate the average total price of bookings grouped by room_type in the hotel_bookings table without caching. Execute the same query after caching the table and compare the duration.

**C)** After performing the above use-case, un-cache the table to free up the memory.