# Title: Avoiding large number of Lead score Sidekiq Jobs to be scheduled during Imports / Migrations.

|  |  |
| --- | --- |
| Status | NOT STARTED |
| Stakeholders |  |
| Outcome |  |
| Due date |  |
| Owner | [Bhagirath Goud](https://confluence.freshworks.com/display/~bhagirath.goud) |

We faced Redis Memory outage in Production, during large data imports because of the scheduled jobs of Lead score in Sidekiq.

Scheduling the job incase of normal CRUD within the app is fine. But, when it comes to Migrations/Imports, scheduling "N" jobs for the "N" record's import seems like risky. This is what caused the issue that i mentioned above.

**What can be done ? - Below is my proposed implementation strategy.  (OPEN TO SUGGESTIONS)**

1. We can stop triggering the Lead score jobs when we are running the Migrations.

2. Once the Migration is done, push the affected **record IDs**in a **Redis Set.** And,schedule a **Single** Lead score job with a delay of **6 hrs** passing the redis key as argument to the job**.**

* Since, Redis Set eliminates the duplicates. Even if the Migration is ran N times within 6 hrs with M number of records each, we will have only M entries in the set.
* And, we can even push all the M ids in a single redis write operation. O(1)

3. Once the job is picked up after 6 hrs, we can find the matching records with a **batch size of 500** from Mysql based on the jobs' argument and **calculate the Lead score for them and overwrite the existing score.**

* Calculating the Lead score for each record can be done by taking the **Intersection of Explicit and Implicit rules against record's properties and entries in Timeline feeds table.**This needs to be done for all the matching records.
* The Logic for finding the Intersection mentioned above is already implemented as part of **Lead score Insights feature** that we launched as shown in below pic.
* Since, Migrated / Imported records will be new records, the data for those records will be minimal in Timeline feeds table. So, time for the calculation of score will also be minimal.

![](data:text/html;base64,)

**Pros of the strategy:**

* We can eliminate scheduling of M \* N jobs when an user imports N times with M records each within 6 hrs. This will solve the problem of Redis Memory outage issues that we faced in Production.

**Cons of the strategy:**

* Since, we will be running the job with a delay of 6 hrs. Lead score **will not be Realtime for the Migrated / Imported records**.
* We will be **querying** the Leads and Timeline Feeds tables for calculating the score after 6 hrs. In the earlier approach, we used to pass the data change set as argumnet to each job. So, we never ran queries on Mysql.