**Spring batch**

**What is batch Processing?**

Batch processing refers to a method of processing data or information in which a large amount of input is collected, processed, and outputted all at once. In this approach, data is collected and grouped together into batches, which are then processed in one go rather than being processed one by one in real-time.

This method is commonly used in applications where a large volume of data needs to be processed in a short period of time. Batch processing is often used in industries such as finance, healthcare, and telecommunications where large amounts of data need to be processed in a timely and efficient manner.

**What is Spring Batch?**

It is a lightweight batch framework based on spring.

It is used for high volume batch processing.

Various I/O options like JSON, XML, SQL databases etc.

Easy to manage i.e start/stop/restart options.

Retry ans skip mechanism.

**What is Job and Step?**

A job is a unit of work that contains a sequence of steps to be executed in a specific order. Each step represents a specific task to be performed as part of the job.

A step in Spring Batch is a self-contained unit of work that performs a specific task within a job.

**What are the different Step Types in spring Batch?**

Tasklet: A tasklet step is a simple step that performs a single task or a set of tasks. The task is defined in a Tasklet implementation, which is executed when the step is executed. This type of step is suitable for performing simple tasks that do not require complex processing or decision-making.

Chuck Oriented: A chunk-oriented step is a step that reads data in chunks, processes each chunk, and writes the results to the output. The chunk size can be configured, and the processing of each chunk is performed by an item processor. This type of step is suitable for processing large amounts of data, such as reading from a database and writing to a file.

Each chuck oriented step may contains

1. ItemReader
2. ItemProcesseor
3. ItemWriter

To read from csv we have to add a job and a step. In step we need to configure the itemreader and itemwriter. The step we are using is a chunck based. Chuck based step ususally take a chunck of items lets say 3, then one by one will be sent to itemreader and itemprocessor but all will be sent to itemwriter in a transaction.

Graphical user interface, website

Description automatically generated

Graphical user interface

Description automatically generated

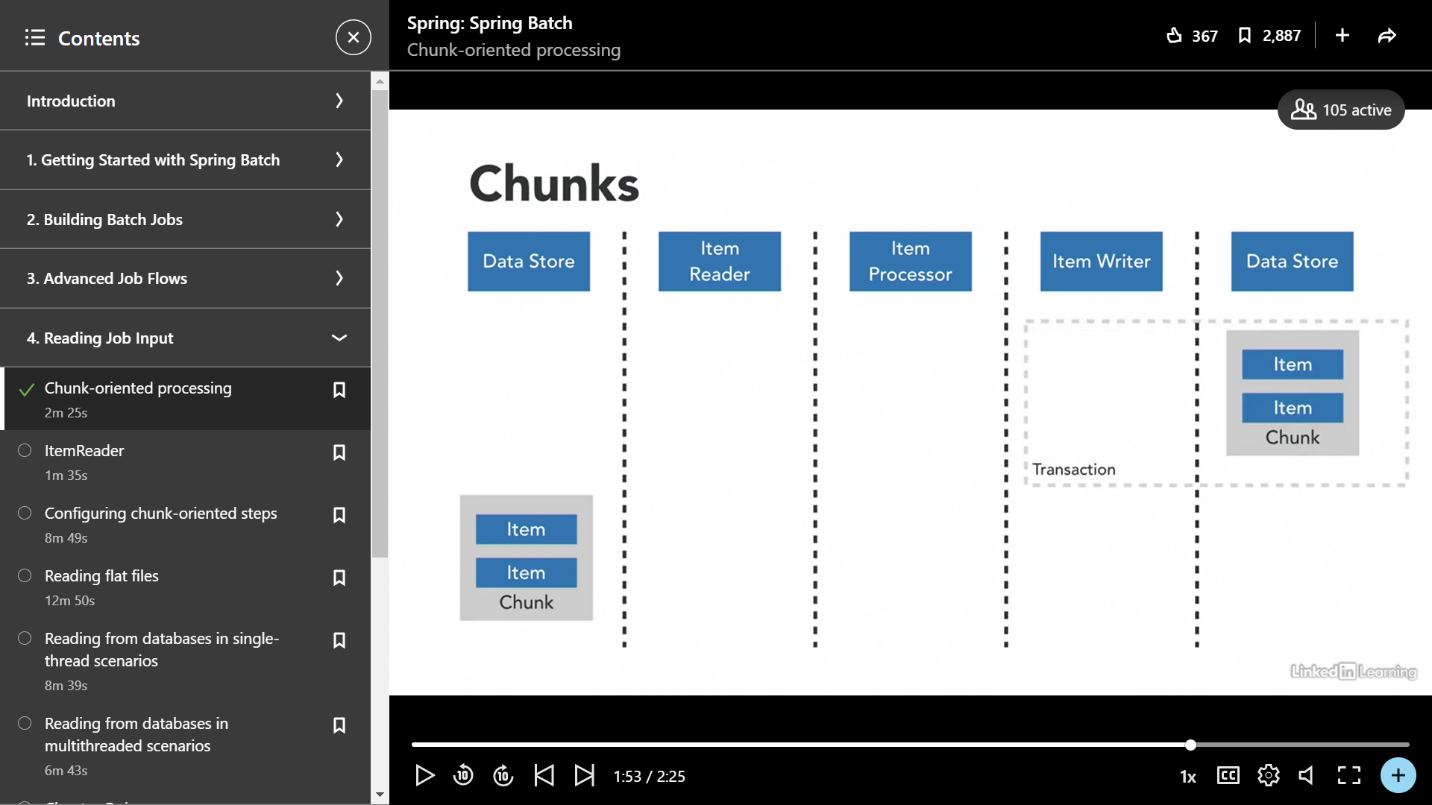
Graphical user interface

Description automatically generatedGraphical user interface

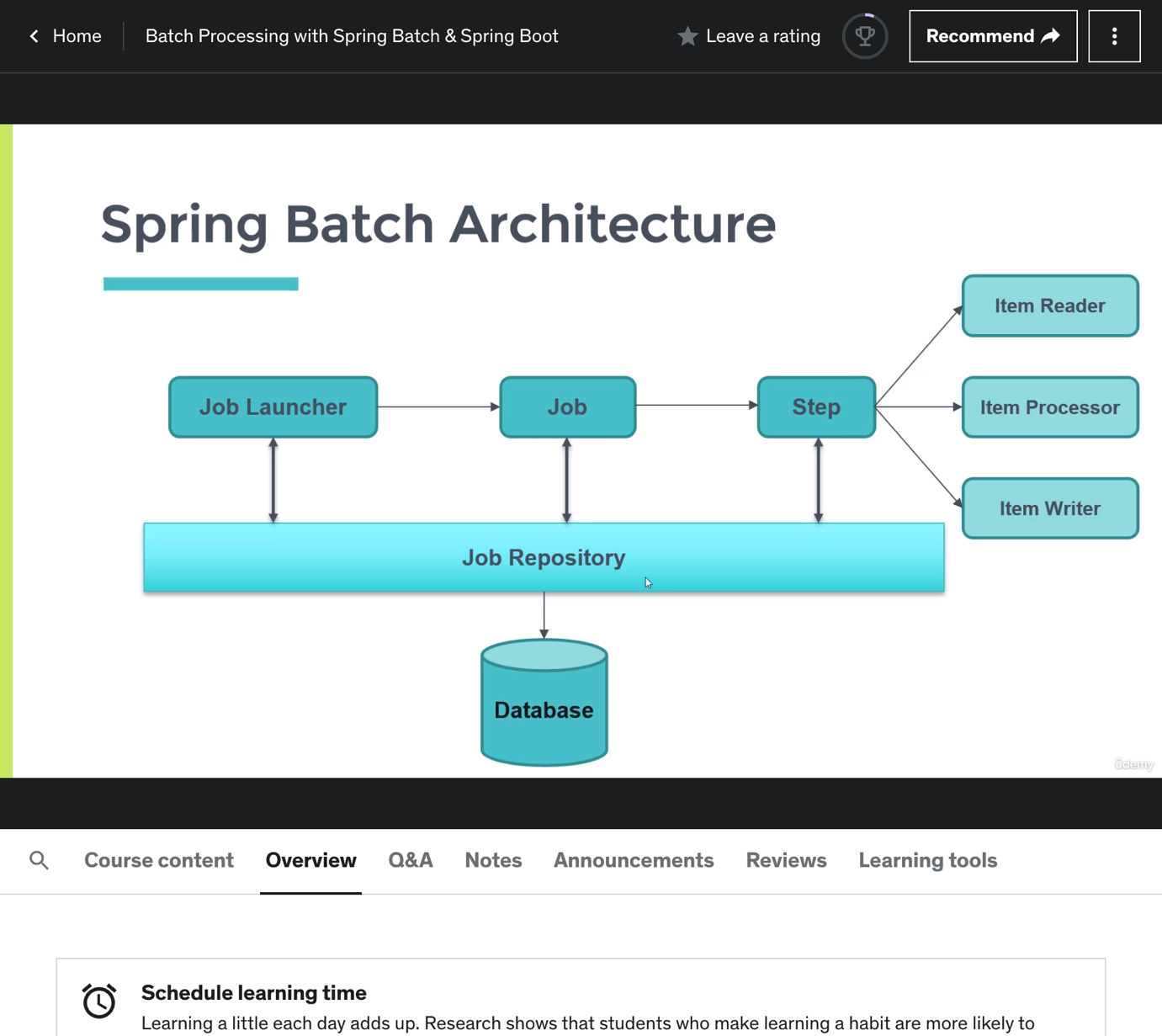
Description automatically generated with medium confidenceGraphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generatedGraphical user interface

Description automatically generated

**Spring Batch Architecture**



**Job Launcher:**

In Spring Batch, a "JobLauncher" is an interface that provides a way to start and run batch jobs. It is responsible for configuring and executing a batch job, including setting up the job parameters and triggering the job execution.

The JobLauncher interface defines a single method, "run", which takes a Job instance and a JobParameters instance as input.

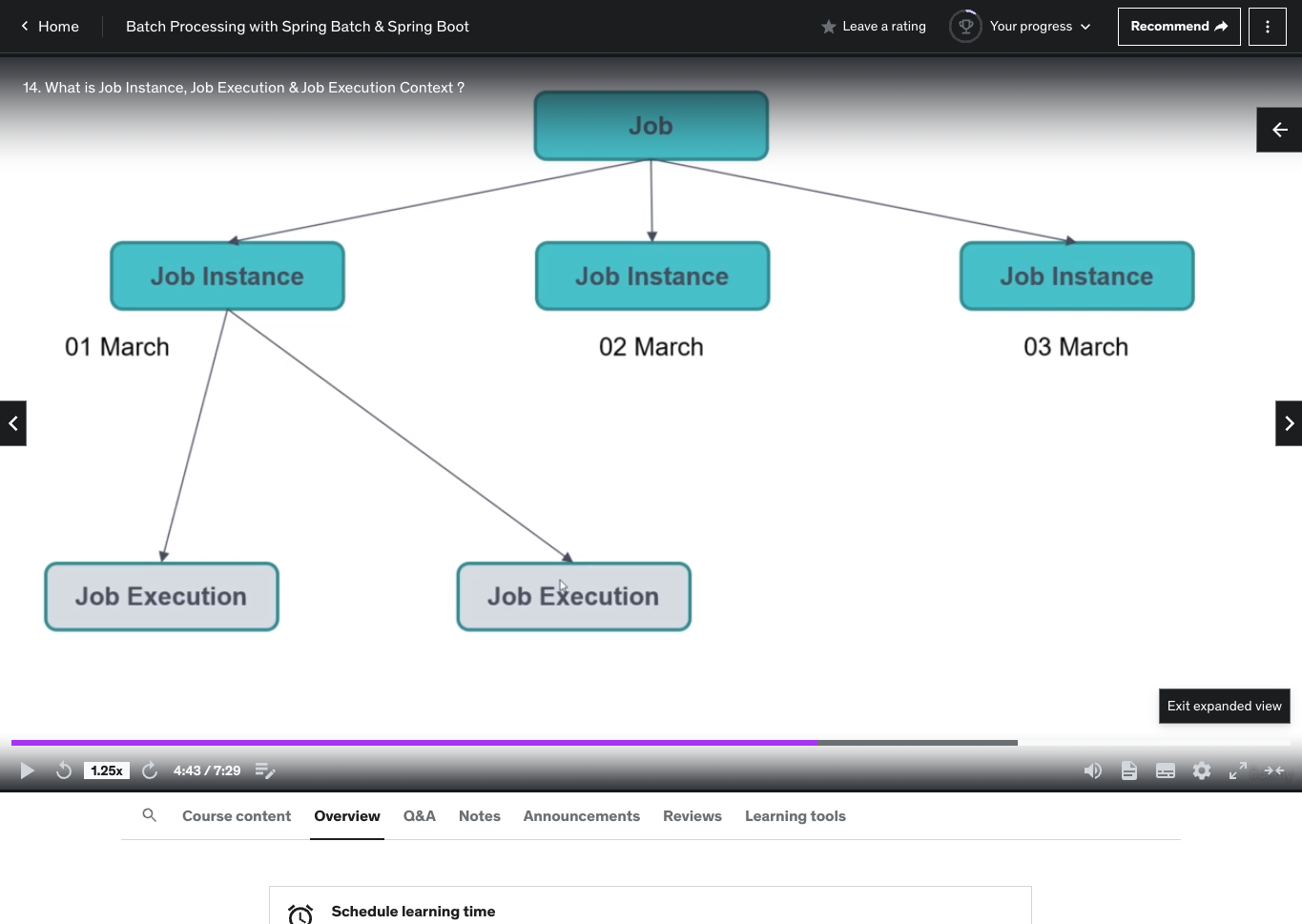
**What is Job Instance and Job Execution?:**

*JobInstance* represents a unique occurrence of a batch job, while JobExecution represents a specific runtime execution of a batch job. JobInstance is used to track the history of job executions, while JobExecution is used to monitor the progress and status of a running batch job.

It is managed by the JobRepository, which stores information about the job instances and job executions, allowing users to track the progress of the batch job and manage any errors or exceptions that occur during the job execution.

If a JobInstance is started at 1st March 2AM in the morning, a jobExecution will be there to track the progress of the job. If the job fails because of any reason a new Job Execution will be created. If at the first time job is success we cant create a new JobExecution for that same JobInstance.

*Job Execution context* is actually a job level context which stores the information of a step in map using key value pairs. This information will be helpful to other steps if needed.



Graphical user interface, diagram, application

Description automatically generated

JobInstances refer to the concept of a logical job run, which is defined by the job and its parameters. Once a job instance has been successfully completed it cannot be restarted. The job can only be re-run with different parameters. Only one JobInstance corresponds to a particular job and its parameters.

Graphical user interface, diagram, application

Description automatically generated

**Whenever a job failed and spring batch re-runs the failed job, same jobinstance will be there but different job execution.**

**Whenever a job instance with a certain parameters are successfully execute it is impossible to re-run the job instane again. Instead we need to create an different job instance with differnt job parameters.**

Spring Batch in Spring Boot

Create a spring boot project version 2.7.7 or 2.7.9 and add dependecies like spring batch , mysql connecter. Remember that if no sql database is not present in the classpath spring batch wont start.

Add @EnableBatchProcessing annotation in the main class. This annotation is used to enable batch processing features and auto-configuration. It automatically configures JobRepository, JobLauncher, JobRegistry, StepScope, TransactionManager.



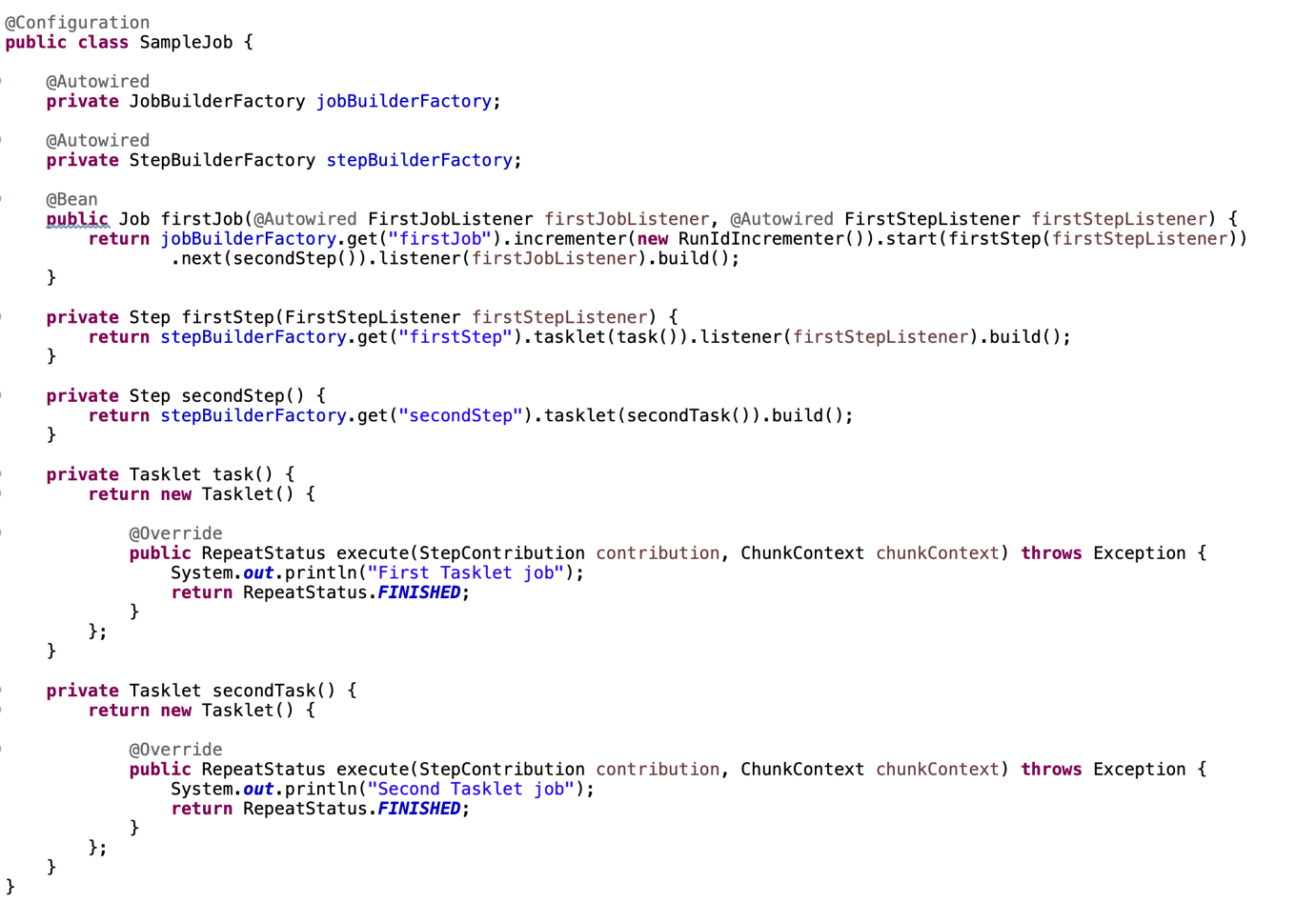
**Tasklet Step code Implementation**

Create a SampleJob config class, annoate the class @Configuration.

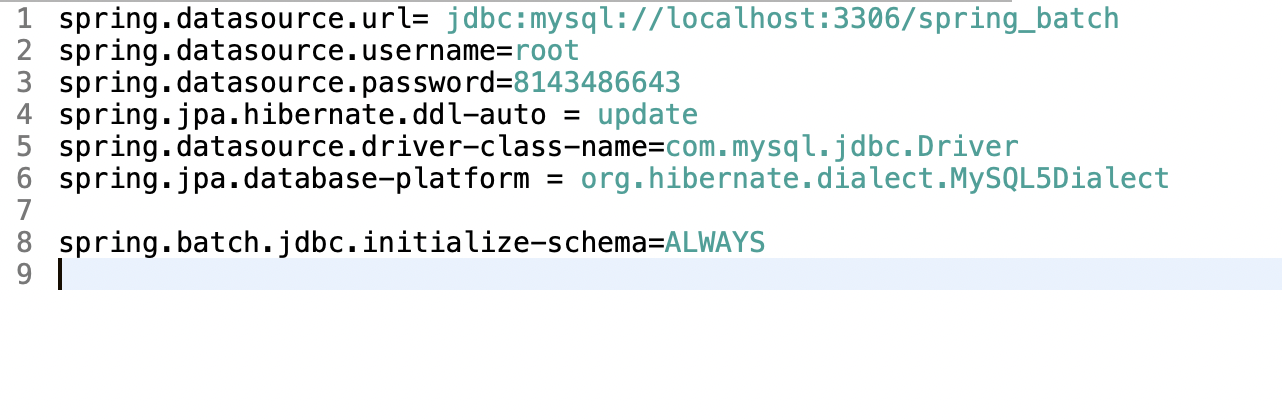
To create a job we need to use jobBuilderFactory “jobBuilderFactory.get(“firstJob”).start(firstStep()).build()” . So we can attached one or more steps to the jobs. I attached a firstStep and called build function. We can attach many steps and also listeners to the job.

To create a tasklet step we need to use stepBuilderFactory “stepBuilderFactory.get(“firstStep”).tasklet(task()).build()”. We can create a task from Tasklet Interface.

Tasklet Interface provides a anonymous class which contains execute method which runs when the step is executed in a job.



After creating a Job and tasklet steps, add the datasource configuration in application.properties



spring.batch.jdbc.initialize-schema=ALWAYS => this configuration will tell spring batch to create the necessary spring batch tables for storing job metadata. If tables are created once wont be created again and again, so don’t need to comment out the configuration after running first time.

**Below tables are created by the spring batch.**

BATCH\_JOB\_EXECUTION\_PARAMS

BATCH\_JOB\_EXECUTION

BATCH\_JOB\_EXECUTION\_CONTEXT

BATCH\_JOB\_EXECUTION\_PARAMS

BATCH\_JOB\_EXECUTION\_SEQ

BATCH\_JOB\_INSTANCE

BATCH\_JOB\_SEQ

BATCH\_STEP\_EXECUTION

BATCH\_STEP\_EXECUTION\_CONTEXT

BATCH\_STEP\_EXECUTION\_SEQ

Important Note:

When you run the spring boot application first time, the job execution will be completed. You can see the sysouts we logged in Tasklet steps in console and also you can see the metadata in the tables.

If you run spring boot application again then application will run and stop. You can see the metadata that no steps are executed and in table BATCH\_JOB\_EXECUTION it says 'All steps already completed or no steps configured for this job.'

If you want to run the job again you need to run the spring boot application with arguments. We can do that with Run as Configuration/Arguments -> name=vishal. If you restart the application we can observe the job runs again and job metadata in table.

If you restart the application again with same arguments application fails with message “job instance is already completed with same parameters”.

If you want to restart the job with same job parameters Spring Batch provides “RunIdIncrementer” which I configured while creating a Job, please refer in above image.

**JobListener and StepListener**

We can consider this listeners as a interceptors which can be use to intercept the job/steps or for logging. I configured the listeners while creating jobs and steps.

