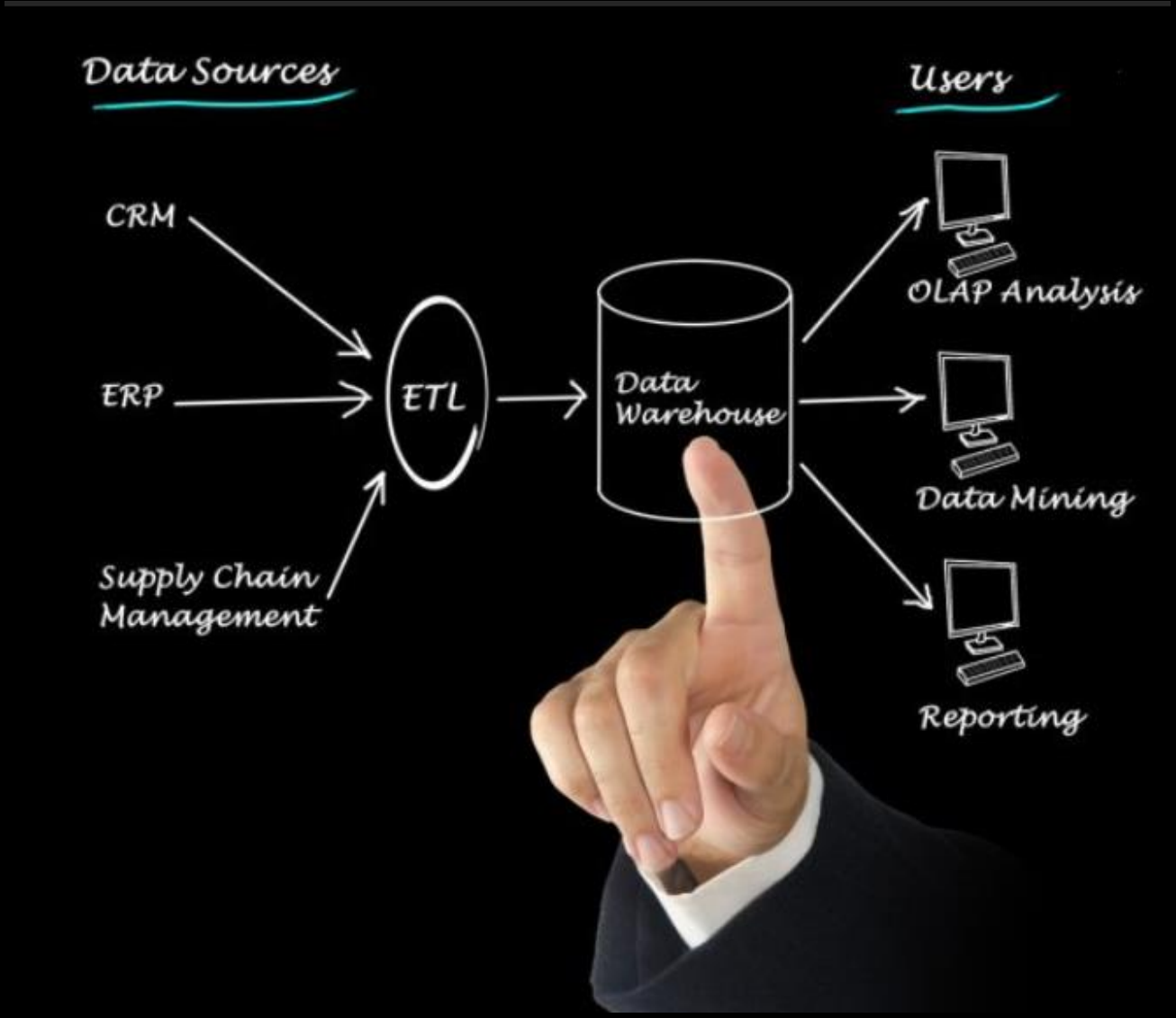
Data Warehouse and Business Intelligence

ETL Workflow Scheduler: **JobHoncho**



March 2020

Objective

JobHoncho is an automated job control system for scheduling & monitoring ETL jobs for BI solutions. This can create a dependency between a job with full flexibility to add/remove any job(s) from ETL flow.

Focus Areas	Key Outcomes
1 **Creation of Batch for daily/weekly/Monthly DW load**	▪ JobHoncho can create batch on specific day of a week (or daily) and schedule ETL jobs to load data in DW.
2 **Dependency between jobs.**	▪ Creating dependency between jobs is just a matter of few row insertion in control tables. ▪ Adding/Removing any job from flow is also easy.
3 **Skipping job in ongoing DW load**	▪ JobHoncho provides basic scheduler options like skipping a job, hung the flow etc.

Problems JobHoncho can solve:

Job dependency

- DS scheduler doesn't allow to set dependency between jobs. With JobHoncho, it's easy to create dependency.

Add or remove job

- To add/remove job in workflow, DS wrapper sequence needs to be changed which is time consuming and involves risk.

Skipping a job on failure

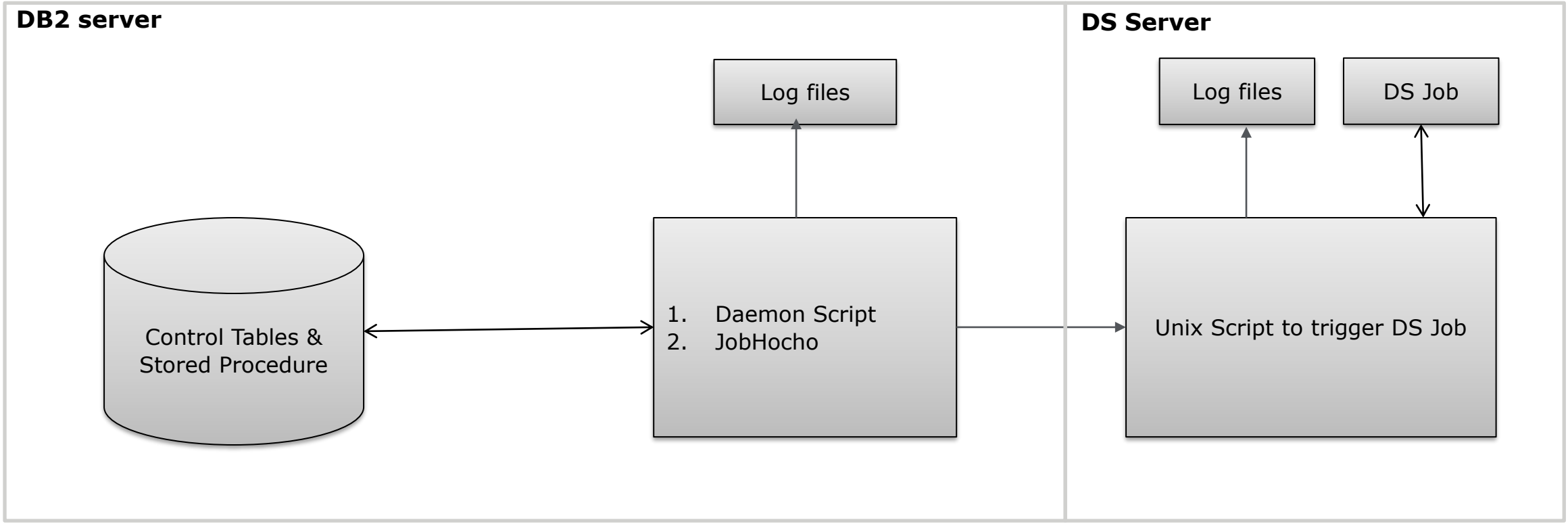
- If a job aborts, there is no way to skip the job in DS scheduler. JobHoncho provide the flexibility to skip a job just by updating the status to 'Complete' in CTRL_BATCH_JOBS table.

Hung the process

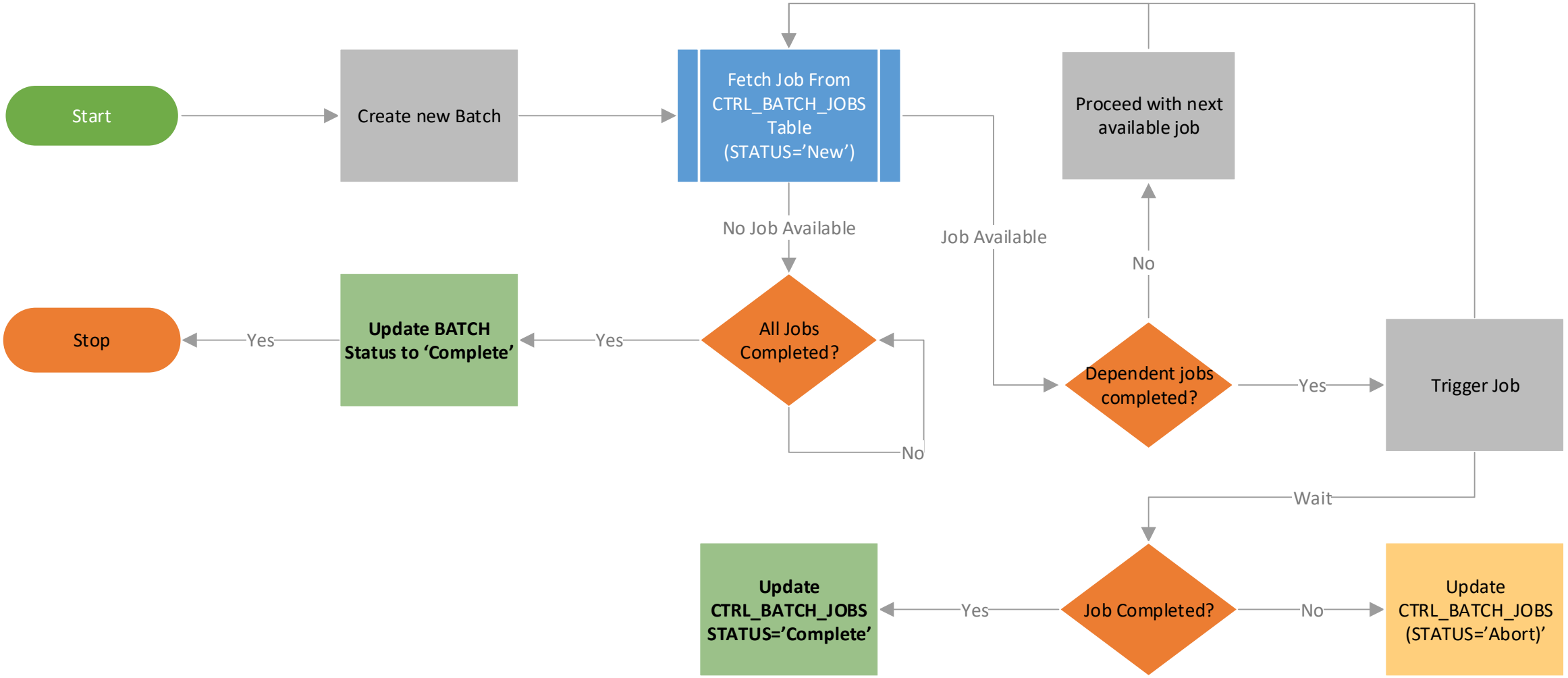
- DS scheduler doesn't provide options to wait for files or another process to complete beyond a pre-defined time. JobHoncho wait for a specific job until it's complete or abort.

JobHoncho Architecture:

- **Control tables:** The Batch Control Tables contain all the metadata about jobs that have been and are to be run.
- **Stored Procedure:** This SP identify the run ability status of a job by checking dependent job status.
- **Unix Scripts:** The Unix Scheduler script is continually running, checking for new jobs to run and reporting back the status of jobs that have been run.
- **DS Jobs:** ETL code to load DW.



JobHoncho Process Flow:



Why JobHoncho?

1 **GPS Data security**

JobHoncho is inhouse code residing in the internal server. Hence give full control & security of data.

2 **Easy to configure & use**

JobHoncho is exclusively built for internal scheduling purposes. That gives the ability to configure a job with a minimum understanding of the system.

JobHoncho
ETL Scheduler

JobHoncho Control tables:

Below are the details about control tables. To configure JobHoncho for new ETL jobs, entries need to be created in the tables.

1. **CTRL_ETL_JOBS:** This table holds details of all ETL jobs that need to be triggered.
2. **CTRL_JOB_DEPENDENCIES:** Dependency between ETL jobs are maintained here.
3. **CTRL_PARAMETERS:** Details about job parameters are stored here.
4. **CTRL_BATCH_JOBS:** After new batch creation, all jobs that need to be triggered are inserted into this table.

CTRL_ETL_JOBS :

Rows 5; SELECT * FROM ATADAS.CTRL_ETL_JOBS — Rows: 1, Cols: 1

Results | MetaData | Info | Overview / Charts | Rotated table | Results as text

INSTANC...	JOB_NAME	JOB_DESCRIPTION	JOB_TYPE	RUN_COMMAND	INCLUDE_IN_BATCH	NOTIFY_ON_FAILURE	RECIPIENT_MAIL	RUN_DAILY_IND	RUN_WEEKLY_IND	RUN_WEEK_DAY	RUN_MO
WYDW	J_TEST_1	<null>	D	<null>	Y	1	atadas@deloitte.com	1	0	1	0
WYDW	J_TEST_2	<null>	D	<null>	Y	1	atadas@deloitte.com	0	1	5	0
WYDW	J_TEST_3	<null>	D	<null>	Y	1	atadas@deloitte.com	0	0	1	1
WYDW	J_TEST_4	<null>	D	<null>	Y	1	atadas@deloitte.com	0	1	1	0
WYDW	PrintDate.sh	<null>	U	'echo date'	Y	1	atadas@deloitte.com	1	0	0	0

CTRL_PARAMETERS:

Rows 2; SELECT * FROM ATADAS.CTRL_PARAMETERS

Results | MetaData | Info | Overview / Charts | Rotated table | Results as text

INSTANCE_ID	JOB_NAME	PARAM_NAME	PARAM_VALUE
WYDW	J_TEST_1	FilePath	/home/developer1/Atanu/Batchman
WYDW	J_TEST_2	FilePath	/home/developer1/Atanu/Batchman

CTRL_JOB_DEPENDENCIES :

Rows 4; SELECT * FROM ATADAS.CTRL_JOB_DEPENDENCIES

Results | MetaData | Info | Overview / Charts | Rotated table | Results as text

INSTANCE_ID	JOB_NAME	DEPEND_ON_JOB_NAME	DESCRIPTION	HARD_DEPEND_IND
WYDW	J_TEST_3	J_TEST_1	<null>	1
WYDW	J_TEST_3	J_TEST_2	<null>	1
WYDW	PrintDate.sh	J_TEST_1	<null>	0
WYDW	PrintDate.sh	J_TEST_3	<null>	0

Process Output:

Creating new batch: Every week/day, new batch will be created on a specific time. Also, add hoc batch can be created through JobHonchoDaemon.

```
[atadas@USSLTC7496v JobHoncho]$ sh JobHonchoDaemon.sh CreateBatch

New batch 181 created.....
Start the daemon.....
[atadas@USSLTC7496v JobHoncho]$ nohup sh JobHonchoDaemon.sh Start &
[1] 24603
```

CTRL_BATCH_NAME:

Rows 1; SELECT * FROM ATADAS.CTRL_BATCH_NAME WHERE BATCH_ID = 181

Results | MetaData | Info | Overview / Charts | Rotated table | Results as text

INSTANCE_ID	BATCH_ID	BATCH_START_TS	BATCH_END_TS	BATCH_STATUS
WYDW	181	2020-05-28 11:04:02.0	<null>	Started

CTRL_BATCH_JOBS:

Rows 4; SELECT * FROM ATADAS.CTRL_BATCH_JOBS WHERE BATCH_ID = 181 — Rows: 1, Cols: 0

Results | MetaData | Info | Overview / Charts | Rotated table | Results as text

BATCH_ID	INSTA...	JOB_PROCE...	JOB_NAME	JOB_TYPE	RUN_COMMAND	RECIPIENT_MAIL	NOTIFY_O...	JOB_STATUS	JOB_START_TS	JOB_END...	EXTRACT_DATE...	EXTRACT_DATE_T
181	WYDW	1	J_TEST_1	D	<null>	atadas@deloitte.com	1	Running	2020-05-28 11:13:32.709553	<null>	2020-05-26 15:42...	2020-05-28 11:04:02
181	WYDW	2	J_TEST_2	D	<null>	atadas@deloitte.com	1	Running	2020-05-28 11:13:33.243317	<null>	2020-05-26 15:42...	2020-05-28 11:04:02
181	WYDW	3	J_TEST_3	D	<null>	atadas@deloitte.com	1	New	<null>	<null>	2020-05-26 15:42...	2020-05-28 11:04:02
181	WYDW	4	PrintDate.sh	U	'echo date'	atadas@deloitte.com	1	New	<null>	<null>	2020-05-26 15:42...	2020-05-28 11:04:02

Process Output: Continue……..

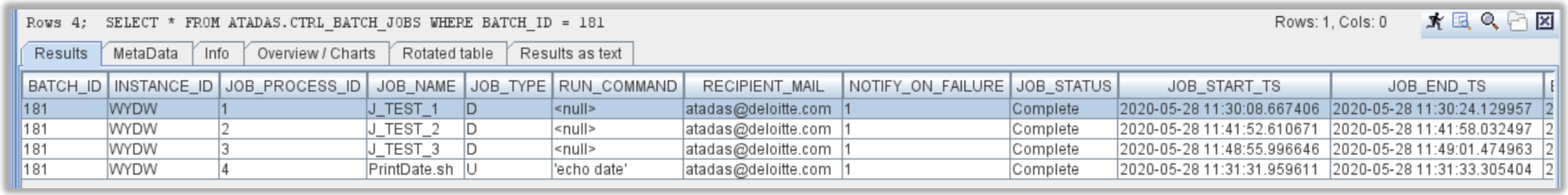
Stop/Start JobHoncho:

[atadas@USSLTC7496v JobHoncho]$ sh JobHonchoDaemon.sh Stop
Script has been stopped...

[atadas@USSLTC7496v JobHoncho]$ nohup sh JobHonchoDaemon.sh Start &
[1] 15881

Completion of Batch:
Once all the ETL job completed successfully, JobHoncho will complete the batch & stop running.

CTRL_BATCH_JOBS:

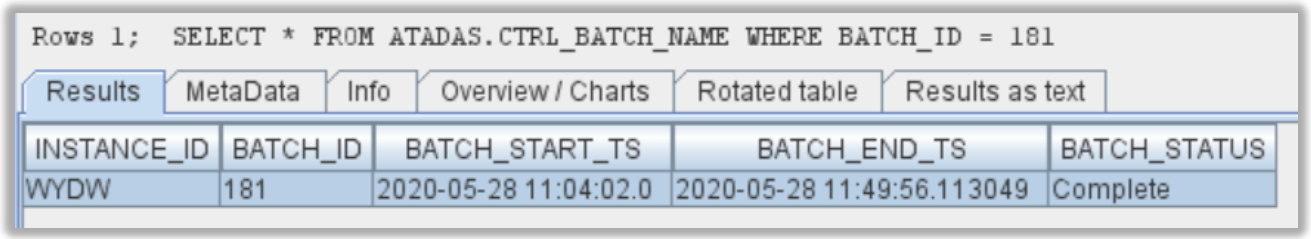


Rows 4; SELECT * FROM ATADAS.CTRL_BATCH_JOBS WHERE BATCH_ID = 181 — Rows: 1, Cols: 0

Results | MetaData | Info | Overview / Charts | Rotated table | Results as text

BATCH_ID	INSTANCE_ID	JOB_PROCESS_ID	JOB_NAME	JOB_TYPE	RUN_COMMAND	RECIPIENT_MAIL	NOTIFY_ON_FAILURE	JOB_STATUS	JOB_START_TS	JOB_END_TS	
181	WYDW	1	J_TEST_1	D	<null>	atadas@deloitte.com	1	Complete	2020-05-28 11:30:08.667406	2020-05-28 11:30:24.129957	2
181	WYDW	2	J_TEST_2	D	<null>	atadas@deloitte.com	1	Complete	2020-05-28 11:41:52.610671	2020-05-28 11:41:58.032497	2
181	WYDW	3	J_TEST_3	D	<null>	atadas@deloitte.com	1	Complete	2020-05-28 11:48:55.996646	2020-05-28 11:49:01.474963	2
181	WYDW	4	PrintDate.sh	U	'echo date'	atadas@deloitte.com	1	Complete	2020-05-28 11:31:31.959611	2020-05-28 11:31:33.305404	2

CTRL_BATCH_NAME:



Rows 1; SELECT * FROM ATADAS.CTRL_BATCH_NAME WHERE BATCH_ID = 181

Results | MetaData | Info | Overview / Charts | Rotated table | Results as text

INSTANCE_ID	BATCH_ID	BATCH_START_TS	BATCH_END_TS	BATCH_STATUS
WYDW	181	2020-05-28 11:04:02.0	2020-05-28 11:49:56.113049	Complete

JobHoncho Language:

1. Access last job run datetime() :

   $LastRunDate

2. Access current job run datetime() :

   $RunDate

3. Insert shell command with in single quote (‘’):

   e.g. 'echo date','sh <path>/<script.sh> <param1> <param2> <param3>'

Thank You