CLOUDERA

Transparency and Visibility
CDSW Monitoring Implementation Steps

Implementation

Implementing CDSW monitoring includes these tasks:

1. Create External Hive Table
2. Deploy a bash/shell script
3. Connect the table to a visualization tool
4. Build a Dashboard
5. Implementing end user alerting mechanism

- Create External Hive Table

How it works: Create an external Hive table, where the job details data is stored on the HDFS storage layer.

Steps to perform :

```
CREATE EXTERNAL TABLE edhoperations.cdsw_jobs_details (  project_id STRING, job_name STRING,  job_owner STRING,  job_id STRING,  status STRING, running_at TIMESTAMP,  finished_at TIMESTAMP,  field_id STRING,  project_name STRING ) ROW FORMAT DELIMITED FIELDS TERMINATED BY '|' WITH SERDEPROPERTIES ('field.delim'='|', 'serialization.format'='|') STORED AS TEXTFILE LOCATION 'hdfs://nameservice1/edhoperations/cdsw' ;
```

- Deploy a bash/shell Script

  How it works: The bash/shell script makes the connection to the backend Kubectl DB pod CDSW database, retrieves the job details data from the Kubectl DB pod for CURRENTDAY-1 and loads the data in the HDFS location in CSV format which is accessed by Hive table for querying.

  Bash/Shell script provided at the end of this article.

  Steps to Perform:

  1. Place this script on the same host that hosts the CDSW Master Server.
  2. Schedule a crontab on the same server where the bash/shell script is deployed and set it to run everyday.

- Connect the table to a visualization tool

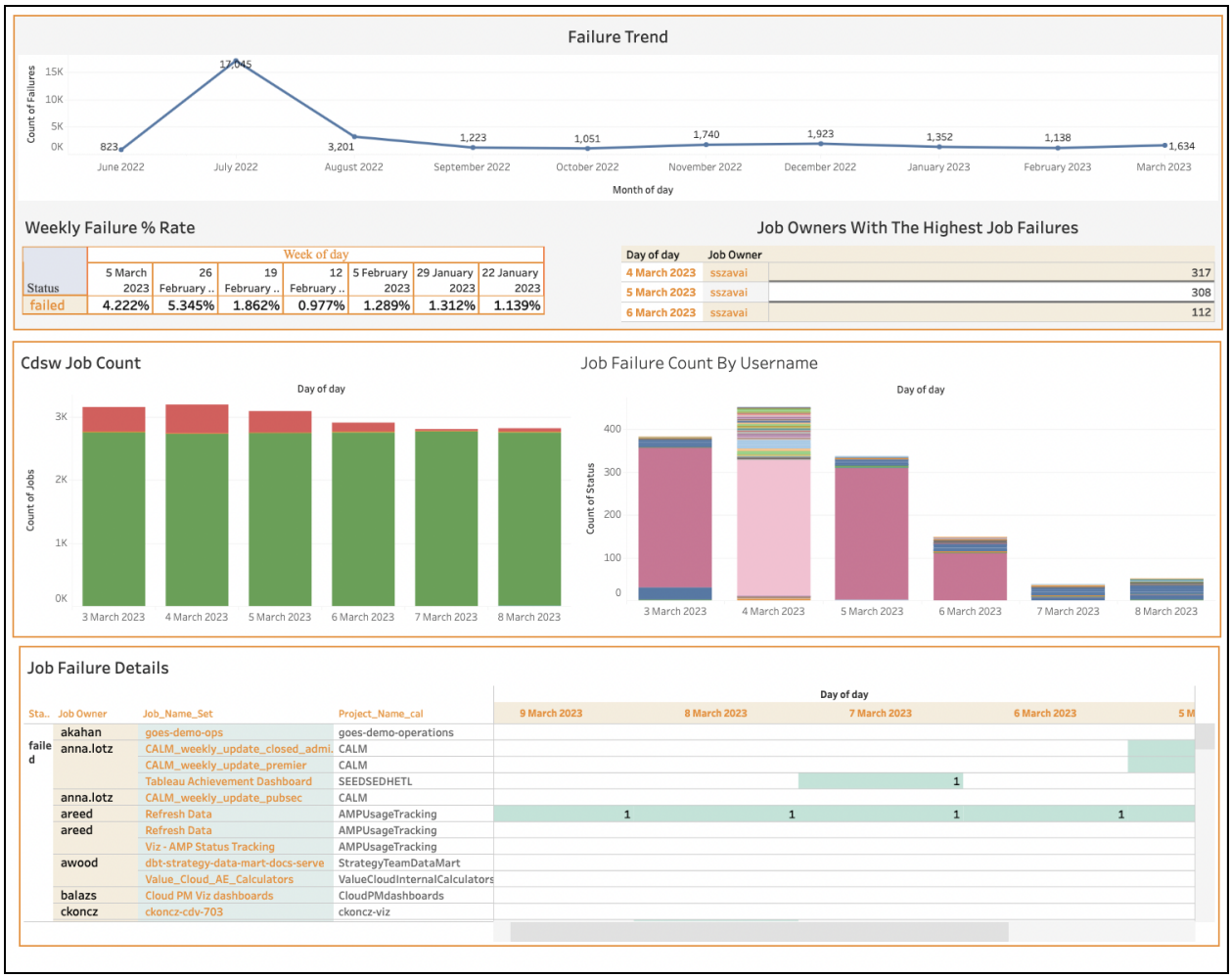
  How it works: To explore and represent the data, you can make use of any visualization tool that has connectivity to Hive or Impala through a JDBC/ODBC connection. We'll demonstrate using Tableau.

  Steps to Perform:

  1. Follow the article link Cloudera Hadoop Tableau Connection to connect Tableau to a Cloudera Data Platform Hive Database.

- Build a Dashboard

  How it works: The basic structure of your visualization should look like below. As seen from the example, different views are brought together to build the dashboard.



Before you begin: Make sure that you have connected to your table data source.

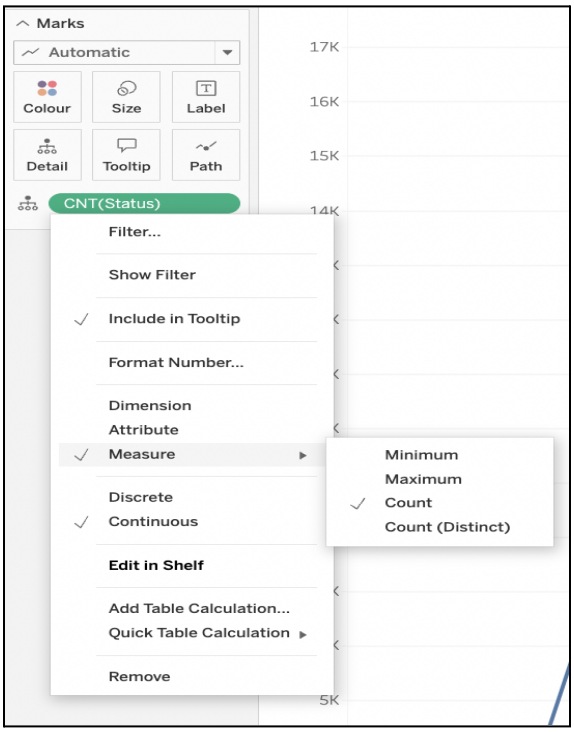
Steps to Perform:

1. Building the Worksheets

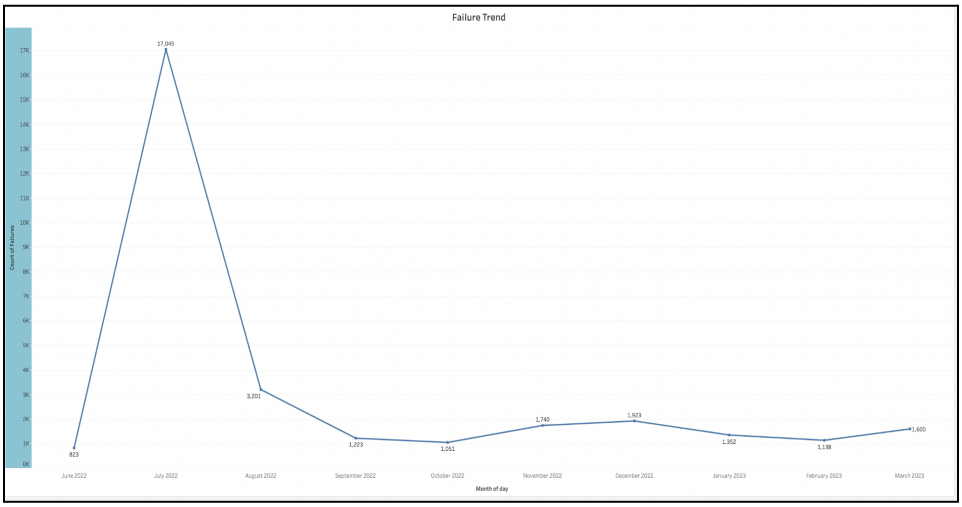
    a. **Failure Trend:** The total count of CDSW job failures is shown in this view with its historical trend.

        1. In your new workbook, Navigate to a New Worksheet and name it "Failure Trend"
        2. From the Data pane, drop **Day** in the **Column** shelf and **Status** in **Row** Shelf
        3. In the **Columns** shelf, right-click **Day** and select **Month** in the format May 2015.
        4. In the **Rows** shelf, right-click **Status** and select **Measure -> Count.**

5. Drop **Status** field in the **Filter** section. Right Click -> **Edit Filter** and select **Failed** from the list of Status.
6. In the **Marks** section, drop **Status** in the **Detail** Icon. Convert it into **Count** Measure. For example :

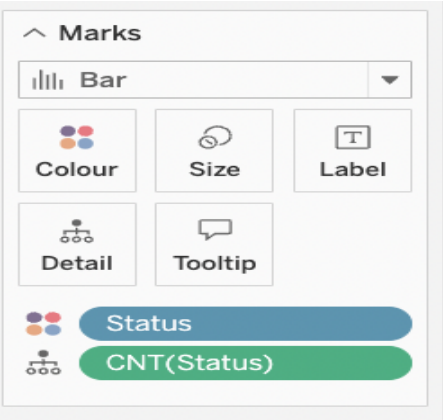


The visualization should look like this:



b. **CDSW Job Count:** This view shows the overall count of jobs executed within the CDSW cluster for CURRENTDAY-1, broken down by failed and successful job color representation.

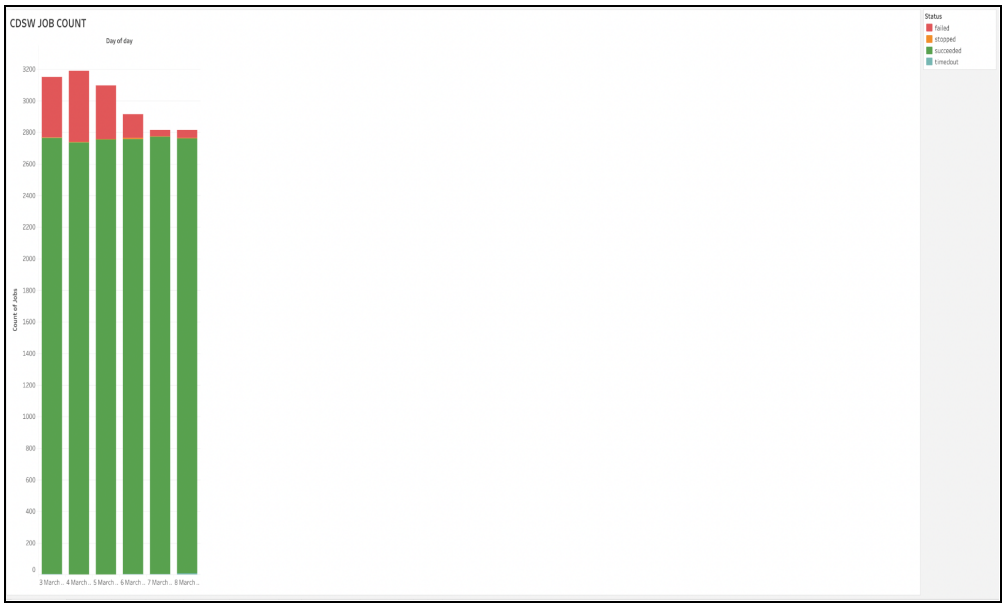
1. Navigate to a New Worksheet and name it "CDSW Job Count"
2. From the Data pane, drop **Day** in the **Columns** shelf and select **Day** in the format **8th May, 2015.**
3. Drop **Status** in the **Rows** shelf and select **Measure -> Count**.
4. Select **Bar Chart** representation from the **Marks** section
5. In the **Marks** Section, drop **Status** in the **Color** and **Detail** icon (Convert the **Status** field into **Count**) . For example:



6. Drop **Day** and **Status** in the **Filter** section.

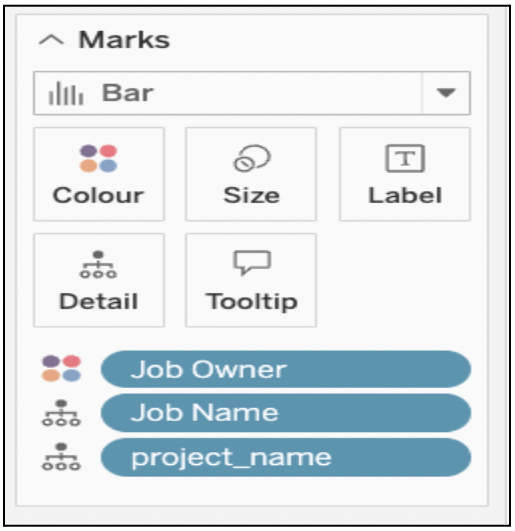
   In this view, we are filtering to just show the last 7 days of job data. Right click on Day, **Edit Filter** -> **Relative dates,** select **Days** from the drop down and enter **7**.
   Do the same for **Status** field, Edit **Filter** -> Select **use all** from the **List**

The visualization should look like this:

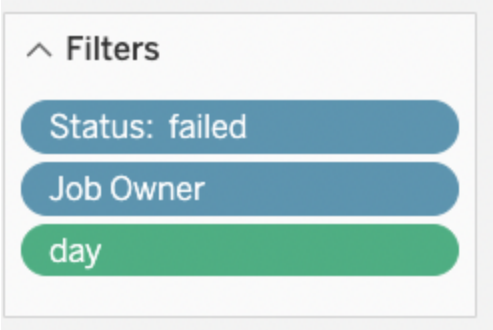


c. **Job Failure Count by User:** This view displays the number of job failures by usernames with job details.

1. Navigate to a New Worksheet and name it "CDSW Job Count"
2. From the Data pane, drop **Day** in the **Columns** shelf and select **Day** in the format **8th May, 2015.**
3. Drop **Status** in the **Rows** shelf and select **Measure -> Count**.
4. Select **Bar Chart** representation from the **Marks** section
5. In the **Marks** section, drop **Job Owner** field in the **Color** icon and **Job Name** and **project_name** field in the **Detail** icon. For example:



6. In the **Filters** section, drop the fields as shown in the below example:



- Status is filtered to only show Failed jobs. Edit **Filter** -> Select **use Failed** from the **List**
- In this view, we are filtering to just show the last 7 days of job data. Right click on Day, **Edit Filter** -> **Relative dates,** select **Days** from the drop down and enter **7**.

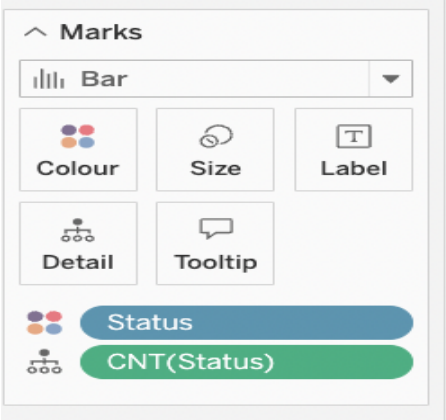
The visualization should look like this:

d. Job Failure Details : This view offers details about the daily failed jobs, such as the job owner, job name, and project name.

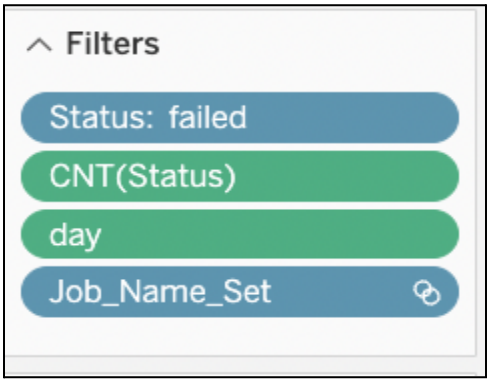
1. Navigate to a New Worksheet and name it "Job Failure Details"
2. From the Data pane, drop **Day** in the **Columns** shelf and select **Day** in the format **8th May, 2015.**

3. Drop the below 4 fields as shown in the example below in the **Rows** shelf:

4. Select **Square** representation from the **Marks** section
5. In the **Marks** Section, drop **Status** in the **Color** and **Detail** icon (Convert the **Status** filed into **Count**) . For example:

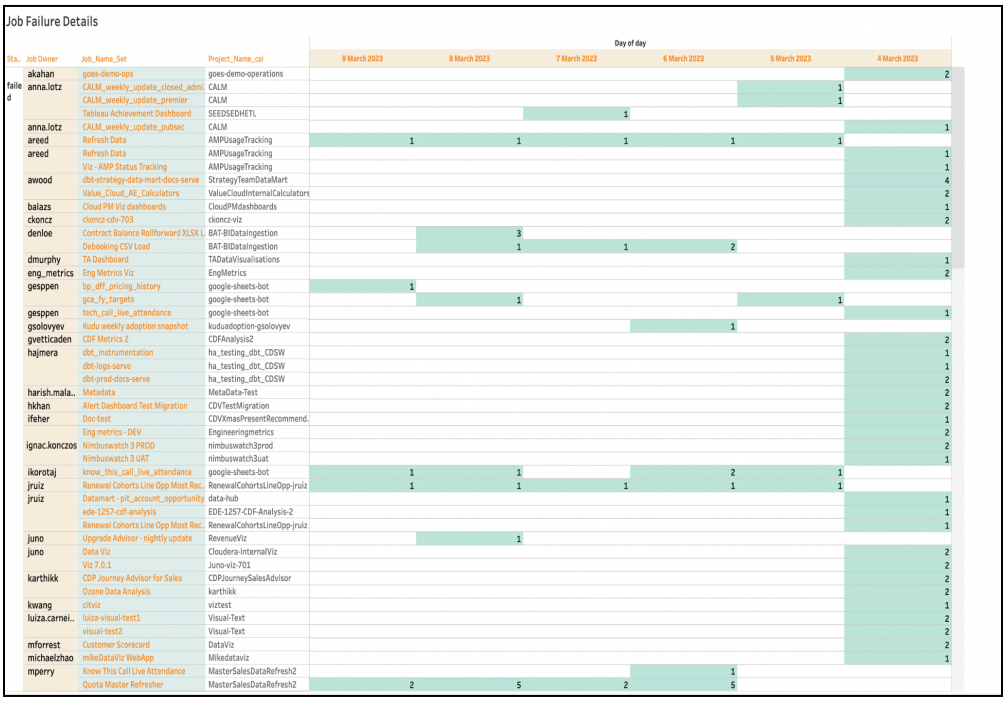


6. In the Filters section, drop the below fields as shown in the example:



- Status is filtered to only show Failed jobs. Edit **Filter** -> Select **use Failed** from the **List**
- In this view, we are filtering to just show the last 7 days of job data. Right click on Day, **Edit Filter** -> **Relative dates,** select **Days** from the drop down and enter **7**.
- Convert Status measure to count. Right Click on **Status** -> **Measure** -> **Count**

The visualization should look like this:



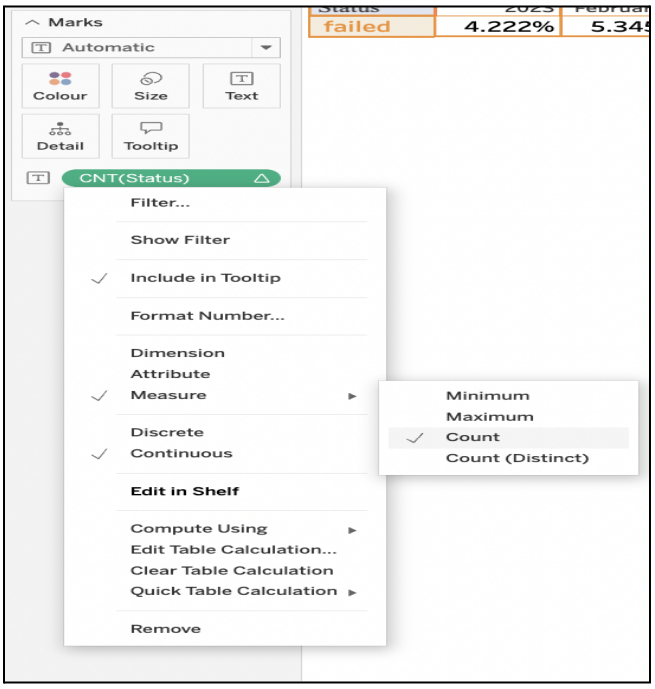
Job Failure Details

Sta..	Job Owner	Job_Name_Set	Project_Name_cal	Day of day: 9 March 2023	8 March 2023	7 March 2023	6 March 2023	5 March 2023	4 March 2023
failed	akahan	goes-demo-ops	goes-demo-operations						2
	anna.lotz	CALM_weekly_update_closed_admi..	CALM					1	
		CALM_weekly_update_premier	CALM					1	
		Tableau Achievement Dashboard	SEEDSEDHETL			1			
	anna.lotz	CALM_weekly_update_pubsec	CALM						1
	areed	Refresh Data	AMPUsageTracking	1	1	1	1	1	
	areed	Refresh Data	AMPUsageTracking						1
		Viz - AMP Status Tracking	AMPUsageTracking						1
	awood	dbt-strategy-data-mart-docs-serve	StrategyTeamDataMart						4
		Value_Cloud_AE_Calculators	ValueCloudInternalCalculators						2
	balazs	Cloud PM Viz dashboards	CloudPMdashboards						1
	ckoncz	ckoncz-cdv-703	ckoncz-viz						2
	denloe	Contract Balance Rollforward XLSX L..	BAT-BIDataIngestion		3				
		Debooking CSV Load	BAT-BIDataIngestion		1	1	2		
	dmurphy	TA Dashboard	TADataVisualisations						1
	eng_metrics	Eng Metrics Viz	EngMetrics						2
	gesppen	bp_dff_pricing_history	google-sheets-bot	1					
		gca_fy_targets	google-sheets-bot		1			1	
	gesppen	tech_call_live_attendance	google-sheets-bot						1
	gsolovyev	Kudu weekly adoption snapshot	kuduadoption-gsolovyev				1		
	gvetticaden	CDF Metrics 2	CDFAnalysis2						2
	hajmera	dbt_instrumentation	ha_testing_dbt_CDSW						1
		dbt-logs-serve	ha_testing_dbt_CDSW						1
		dbt-prod-docs-serve	ha_testing_dbt_CDSW						2
	harish.mala..	Metadata	MetaData-Test						2
	hkhan	Alert Dashboard Test Migration	CDVTestMigration						2
	ifeher	Doc-test	CDVXmasPresentRecommend..						1
		Eng metrics - DEV	Engineeringmetrics						2
	ignac.konczos	Nimbuswatch 3 PROD	nimbuswatch3prod						2
		Nimbuswatch 3 UAT	nimbuswatch3uat						1
	ikorotaj	know_this_call_live_attendance	google-sheets-bot	1	1		2	1	
	jruiz	Renewal Cohorts Line Opp Most Rec..	RenewalCohortsLineOpp-jruiz	1	1	1	1	1	
	jruiz	Datamart - pit_account_opportunity	data-hub						1
		ede-1257-cdf-analysis	EDE-1257-CDF-Analysis-2						1
		Renewal Cohorts Line Opp Most Rec..	RenewalCohortsLineOpp-jruiz						1
	juno	Upgrade Advisor - nightly update	RevenueViz		1				
	juno	Data Viz	Cloudera-InternalViz						2
		Viz 7.0.1	Juno-viz-701						2
	karthikk	CDP Journey Advisor for Sales	CDPJourneySalesAdvisor						2
		Ozone Data Analysis	karthikk						2
	kwang	citviz	viztest						1
	luiza.carnei..	luiza-visual-test1	Visual-Text						2
		visual-test2	Visual-Text						2
	mforrest	Customer Scorecard	DataViz						2
	michaelzhao	mikeDataViz WebApp	Mikedataviz						1
	mperry	Know This Call Live Attendance	MasterSalesDataRefresh2				1		
		Quota Master Refresher	MasterSalesDataRefresh2	2	5	2	5		

e. **Weekly Failure % Rate :** This view offers information on the % job failure rate over the last seven weeks.

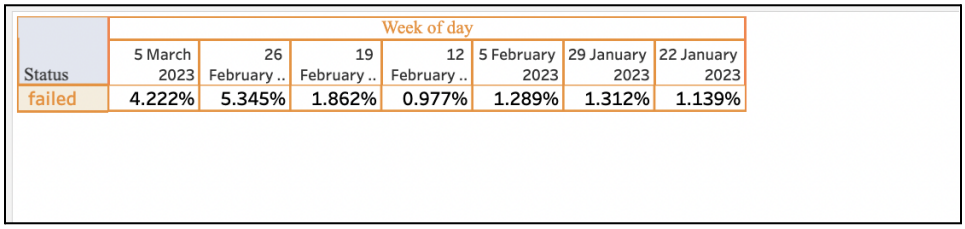
1. Navigate to a New Worksheet and name it “Weekly % failure rate”
2. From the Data pane, drop **day** in the **Columns** shelf and select **Week** in the format **Week 5, 2015.**
3. Drop **Status** in the **Rows** shelf
4. In the **Marks** section, select **Table** as the data visualization preference. Drop **Status** in the **Text** icon and select **Measure -> Count**. For

example:



5. Drop **Day** field in the Filters section. As in this view we are just demonstrating last 7 weeks data. Click on Edit Filter -> Select Relative Dates -> select **Weeks** from the drop down and enter **7** in the Last weeks tab.

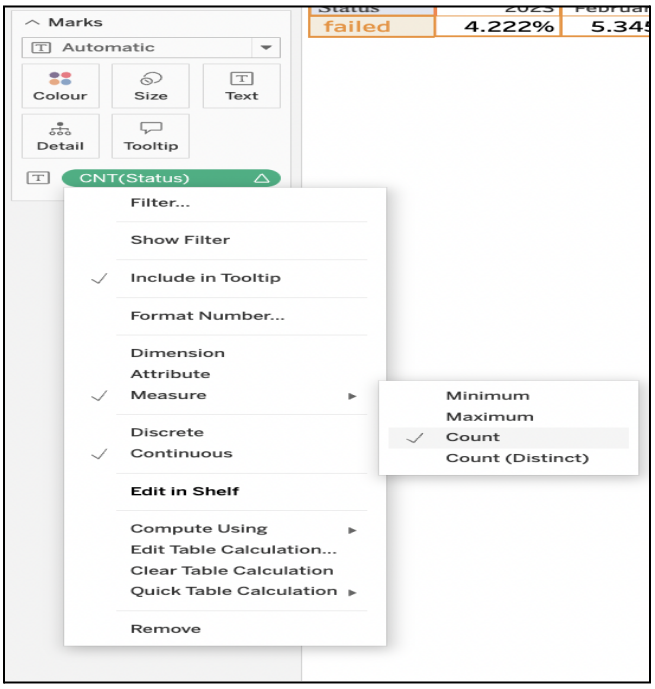
   The visualization should look like this:



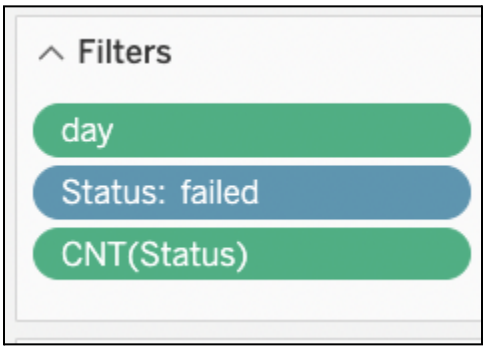
	Week of day						
Status	5 March 2023	26 February ..	19 February ..	12 February ..	5 February 2023	29 January 2023	22 January 2023
failed	4.222%	5.345%	1.862%	0.977%	1.289%	1.312%	1.139%

f. **Job Owners With The Highest Job Failures:** This view displays the names of the job owners and the number of failed jobs they had in the previous seven days Note : Usernames with more than 20 job failures are shown in this view.
   1. Navigate to a New Worksheet and name it "Job Owners With The Highest Job Failures"
   2. From the Data pane, **day** and **Job Owner** in the **Rows** shelf. Select **Day** in the format **8th May, 2015.**
   3. In the **Marks** section, select **Table** as the data visualization preference. Drop **Status** in the **Text** icon and select **Measure -> Count**. For

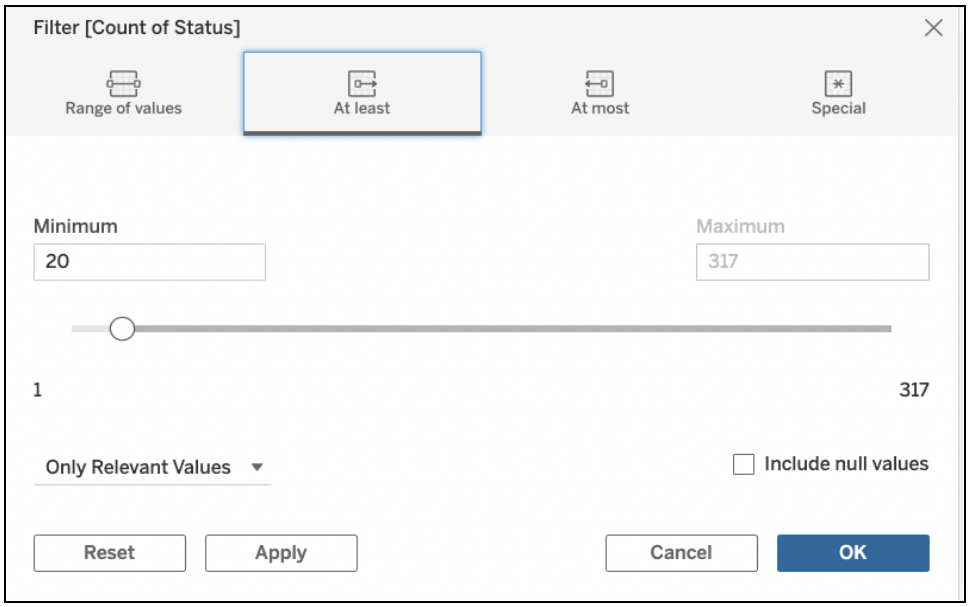
example:



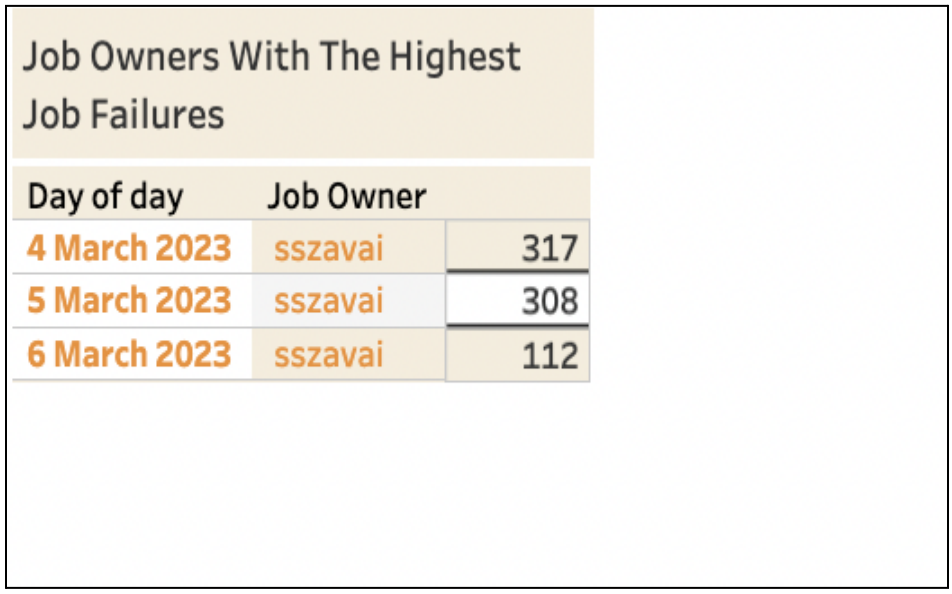
4. Filter Section : Drop the below following fields in the Filters section as shown in the example.



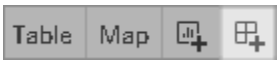
- As in this view we are just demonstrating the last 7 days of data. Right on **day** - > select **Edit Filter** -> Select **Relative Dates** -> select **Days** from the drop down and enter **7** in the Last days tab.
- **Status** is filtered to only show Failed jobs. Edit **Filter** -> Select use **Failed** from the **List**
- In this view we are just displaying the names of users with above 20 failures. To do this, Convert the Status **Measure** into **Count** and Lastly, Select **Edit Filter** -> mention the **Minimum** value as **20**. For example:



The visualization updates to the following



2. Gathering the views to build the dashboard
   a. At the bottom of the workbook, click the New Dashboard icon:



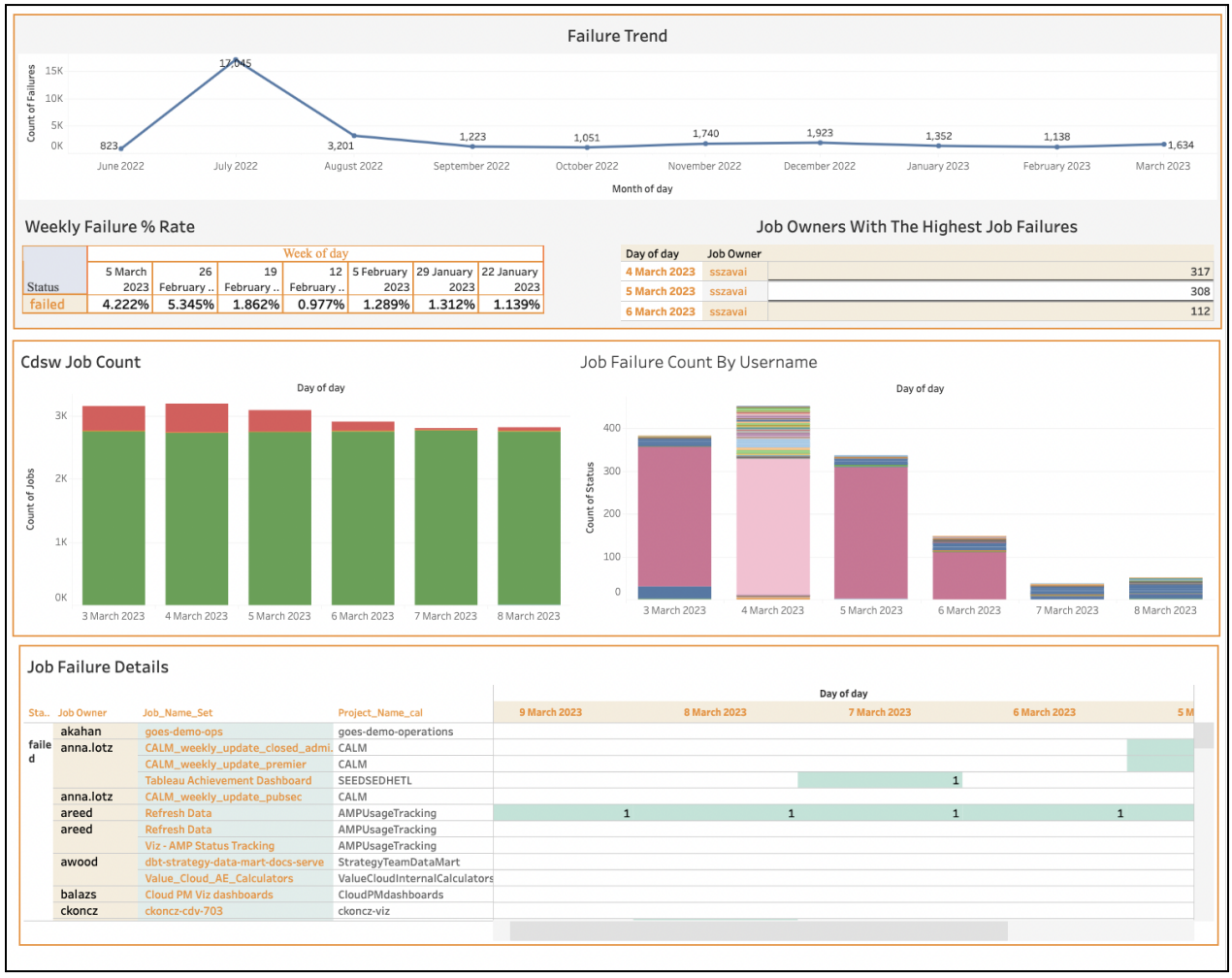
b. You can use horizontal and vertical objects to provide a visual appeal to the dashboard and group different worksheet views together.

c. Make use of filters on your views where necessary.

Note : Design your dashboard as per your visualization preferences.

For more details and best practices on building a dashboard in Tableau refer link : [Create a Dashboard]

Your dashboard visualization updates as below if the views are sequenced properly:



- Implementing End User Alerting Mechanism

How it works: The deployed airflow dag scheduled to run every 23 hours scans across the hive table to collect the day-1 Yarn/Spark job failure data, and then sends an email alert with the failure details to the recipient.

Steps to perform: On the Airflow master host, deploy the below dag code in the dags directory and schedule it to run every 23 hours

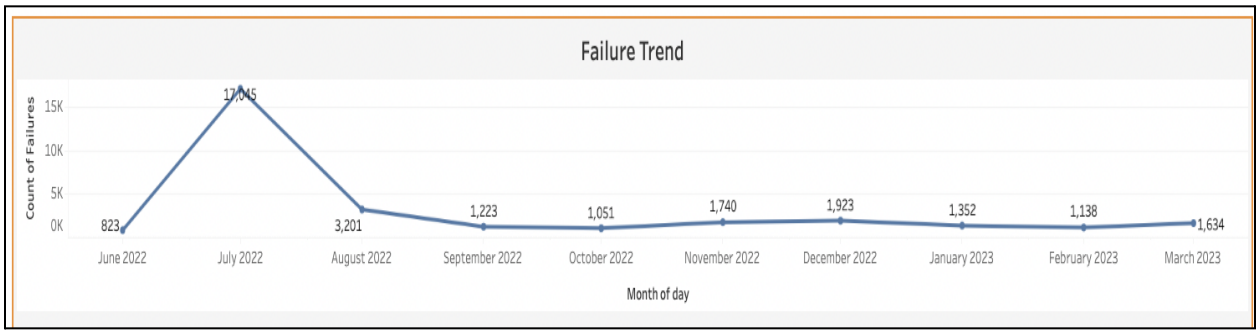
Airflow Dag Code provided at the end of this article

Note: The code provided is merely for reference purposes. Customize the airflow dag per your environment requirements.

Actionable Insights

A good rule of thumb is to keep a view of data showing no less than 6 months to give you a comprehensive view.

1. Tracking progress overtime :



- Pay appropriate attention to any surge in the number of failures using the area chart above, and identify the source of the problem to understand which jobs and users caused the spikes. Use the views in the next section to get job details for troubleshooting.

- Observe and compare the present and past numbers to visualize and understand if there's an overall increase or reduction in the trend of job failures. If you observe a decrease in the failure trend it's a good indication that your CDSW job failures are decreasing; however, if they are increasing, it may be an indication that your operations team should identify and action root causes (see next section).
- Understand and identify the jobs running and the load on your CDSW service within the timespan and pinpoint the root cause if you notice an odd increase in the failure rate happening on a repetitive basis for a specific day or time.

2. Your daily/weekly view for identifying problems, spotting issue trends, and taking action for long term preventive measures.

- "CDSW Job Count" and "Job Failure Count By Username"- Keep a close eye on the total number of daily job failures and identify job owners in the Details view to report the issue to them and ask for action.
    - Reference the total number of jobs getting executed within CDSW to get a basic idea of CDSW Load. Keep an eye out for surges in total and failed jobs count.

    - Assess the incline and decline patterns by comparing the current day count with the previous six days.
- Use the "Job Owners With The Highest Job Failures" to identify the top users causing the highest failure rates, and use the "Job Failure Details" for details of the jobs that should be analyzed.
- Use the "Weekly Failure % Rate" to compare and keep a weekly tab on your failure % rate. Track progress on a weekly basis.
- Use "Job Failure Details" to obtain user, project, and job name information for your daily CDSW job failures. This information will help you identify the projects that are most frequently responsible for job failures, allowing you to further pinpoint the cause using CDSW UI logs.

Recommended Operational Processes

- Schedule daily monitoring calls to analyze CDSW job load and troubleshoot significant failures with the help of the dashboard views.
- KPIs to closely monitor daily :
    1. Increase or sudden spikes in the total number of job failures.
    2. Users resulting in the highest number of job failures.
    3. Should any of these failures prompt either job tuning or service tuning?
- Periodically assess job health progress utilizing the historic data trend view to reduce job failure rate.

Python

Python Script to Retrieve CDSW Job Details Day-1

```
#!/bin/bash
####################
## This script fetches the CDSW job details from Kubectl DB pod
export
PATH="/usr/lib64/qt-3.3/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin
:/usr/bin:/usr/java/latest/bin:/usr/java/latest/jre/bin:/root/bin"
kinit -kt /root/user.keytab user@domian <mention the user keytab and principal
name>
dir="/root/cdsw"
host=`hostname`
hdfs_dir="/edhoperations/cdsw"
today=$(date +%Y-%m-%d)
email_id="user email"


kubectl exec `kubectl get pods --selector=app=db --no-headers | awk '{ print $1
}'` -- psql -d sense -c 'select dashboards.project_id ,  dashboards.name AS
"JOB NAME", users.username AS "JOB OWNER", dashboards.job_id AS "JOB ID",
dashboards.status , dashboards.running_at , dashboards.finished_at ,
dashboards.id AS "FIELD_ID" , projects.name AS "Project_Name" from dashboards
LEFT JOIN users ON dashboards.creator_id = users.id  LEFT JOIN projects ON
dashboards.project_id = projects.id where  DATE(dashboards.finished_at) =
current_date - 1 ' > $dir/cdsw_jobs_$today.csv

COMMAND_STATUS=$?
if [ $COMMAND_STATUS -ne 0 ];then
   echo "$host CDSW fetch jobs detail script failed.Please login $host  and
check script `pwd`/$0 "| mailx  -s "Alert: $host CDSW fetch jobs detail script
failed" "$email_id"
fi;

echo "hostname $host CDSW Jobs details backup at $dir for date `date +%d-%m-%Y`
"


sed -i '1,2d' $dir/cdsw_jobs_$today.csv

sed -i "$(( $(wc -l < $dir/cdsw_jobs_$today.csv )-3+1 )),$ d"
$dir/cdsw_jobs_$today.csv

hdfs dfs -put /root/cdsw/cdsw_jobs_$today.csv $hdfs_dir/cdsw_jobs_$today.csv
```

```
COMMAND_STATUS=$?
if [ $COMMAND_STATUS -ne 0 ];then
  echo "$host $dir cdsw jobs file for date $today Faield to copy HDFS location $hdfs_dir  and check script `pwd`/$0 "| mailx  -s "$hostname $dir cdsw jobs file for date $today Faield to copy HDFS location" "$email_id"
fi;
echo "$host $dir/cdsw_jobs_$today.csv CDSW jobs details file for date $today copied to HDFS location $hdfs_dir "

#Refresh the impala table data
impala-shell  -i  <mention-impala  gateway  hostname  >  -k  --ssl  -q  "use edhoperations; INVALIDATE METADATA cdsw_jobs_details;"

echo "Impala table data refreshed/updated in table"
```

Python

Airflow Dag Code

```
#Importing the Modules
from datetime import datetime, timedelta
import json
from airflow import DAG
from airflow.operators.python import PythonOperator
from airflow.providers.jdbc.operators.jdbc import JdbcOperator
from airflow.models.variable import Variable
from airflow.operators.email_operator import EmailOperator
from airflow.providers.jdbc.hooks.jdbc import JdbcHook
import pandas as pd
from email.mime.text import MIMEText
from email.mime.application import MIMEApplication
from email.mime.multipart import MIMEMultipart
from smtplib import SMTP
import smtplib
import sys

#Defining Default Arguments for the Dag

default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'email': ['job owner email address'],
```

```
    'email_on_failure': True,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=2)
}

doc = '''
## CDSW job monitoring dag to send an email to the Data engineering team
informing about the  job failures that occurred in the last 24 hours.


'''
#Initiating the Dag
JOB_ID = 'EDH_CDSW_failure_jobs_Alert'

dag = DAG(
    dag_id=JOB_ID,
    doc_md=doc,
    default_args=default_args,
    description='EDH_CDSW_failure_jobs_Alert',
    schedule_interval="0 */23 * * *",
    start_date=datetime(2021, 1, 1),
    tags=['EDH', 'Monitoring', 'failure jobs'],
    max_active_runs=1,
    catchup=False
)


#Defining a callable function
This dag example is created with the assumption that the job owner name
corresponds to your email username. You will need to modify the function
definition in accordance with your use case if this is not the case in your
situation.

def func(jdbc_conn_id, sql, **kwargs):
    """Print df from JDBC """
    print(kwargs)
    hook = JdbcHook(jdbc_conn_id=jdbc_conn_id)
    df = hook.get_pandas_df(sql=sql,parameters=None)
    print("printing the jobs details")
    print(df.to_string())
    print(df)
    print(df['job_owner'].astype(str)+'@gmail.com.com')
    df.job_owner = df.job_owner.str.strip()
```

```
    job_own = df['job_owner'].astype(str)+'@gmail.com'
    recipients = ['recipient email address']
    emaillist = [elem.strip().split(',') for elem in recipients]
    emaillist2 = [elem.strip().split(',') for elem in job_own]
    print(emaillist)
    print(emaillist2)
    msg = MIMEMultipart()
    msg['Subject'] = "Alert : CDSW failed jobs for Day -1 "
    msg['From'] = 'sender email address'


    html = """\
    <html>
      <head></head>
      <body>
    <pre>
    Hello Team,

Please fix the below CDSW jobs which failed yesterday ( Day -1 ). For more
detail visit to below dashboard -
             <Mention the link to the dashboard visualization>

    </pre>

        {0}

      <pre>

      Please reach out to the ops team in case of any queries. Thanks

      Regards
      OPS TEAM

      </pre>

      </body>
    </html>
    """.format(df.to_html())

    part1 = MIMEText(html, 'html')
    msg.attach(part1)

    server = smtplib.SMTP('SMTP HOSTNAME', 25)
    server.sendmail(msg['From'], emaillist , msg.as_string())
```

```
#Creating a Task
run_this = PythonOperator(
    task_id='CDSW_Alert_Email',
    python_callable=func,
    op_kwargs={'jdbc_conn_id': 'dcoe_impala', 'sql': 'select
project_id,job_name,job_owner,job_id,status,project_name,trunc(cast(finished_at
as timestamp), "dd") as finished_date from edhoperations.cdsw_jobs_details
where status like "%failed%" and trunc(cast(finished_at as timestamp), "dd") =
date_sub(CURRENT_DATE(), interval 1 days);'},
    dag=dag,
)
```