

**Project Title:** MDP Industry-Sponsored Big Data Backbone for Advanced Analytics

**Sponsoring Organization:** Union Pacific Railroad Company

**Sponsor Mentor:** Gena Van Osdel

**Faculty Mentor:** Bill Arthur

**Team Member Names:** Ann Stone, Celina Pan, Neil Kim, Ken Mahattanadul, Conan Wu

**Date:** December 16th, 2022



## **Section 1: Project Introduction**

### **Sponsor's Information and Project Scope**

Our sponsor, Union Pacific (UP), is the largest freight-hauling railroad company in the world, operating 8,300 locomotives over 32,200 miles of rail track in 23 U.S. states. They own or lease approximately 18,000 railcars. UP has several data pipelines that process messages on the activity of their railcars into the Main Equipment Event Table. The Main Equipment Event Table contains all of UP's railcar event data. The Finance Team within UP uses the data in this table to audit revenue and find missing billings. Currently, UP doesn't receive messages on the activity of their railcars after they move onto a different railroad company's rail tracks, called going "offline". This lack of offline visibility hinders the efficiency of revenue auditing because the Finance Team has to manually search Railinc, the provider of rail data to the North American railroad industry, for missing billings and revenue. The goal of the UP MDP Cohort of 2022 is to help solve this problem by providing UP with a more complete picture of their railcars' activity, including when it goes offline. To accomplish our goal, our solution strategy is to bring more data pertaining to offline railcar activity from Railinc into UP's database. To implement this solution strategy, the main objective of our team will be to build a new pipeline capable of automatically extracting, transforming, and loading (ETL) approximately 6-10 million offline messages from Railinc per day. We will be working specifically with SWRPY87 messages, which are a type of Railcar Tracing (RCT) message that detail the offline activity of a railcar, including railcar location, timestamps, on road, to the road, and other railcar-specific data. Another objective of our team is to produce a data analytics report complete with measurable numbers that show the benefits of the SWRPY87 messages and their ability to increase railcar visibility.

### **Project Background**

To create such a pipeline for the massive amount of data UP has, Hadoop, a data framework that was designed specifically to work with Big Data will be used. More specifically, the team worked with two technologies within the Hadoop ecosystem: Hive and Apache Spark. The team used Hive tables to store data to stay in line with common practices done at UP and make full use of our mentor's expertise. Simultaneously, the team used Apache Spark to read and write from our Hive tables as well as complete data processing. Once data is stored and processed, HiveQL and Spark SQL are used to query and analyze our data in a time and memory-efficient manner. While Spark is compatible with many languages, the team used Scala for its conciseness and ability to combine the best of both functional programming and object-oriented programming.

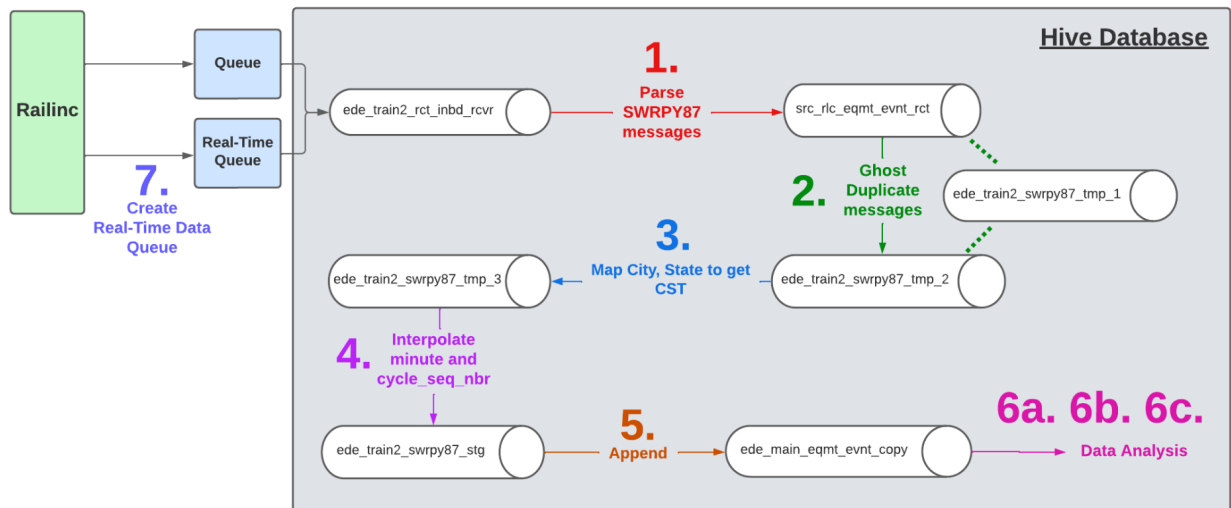
### **Project Organization and Management**

The team is made up of five student members, a faculty mentor, and 4 sponsors. We have decided to work together as a team on this project without subteams. We will be using JIRA to keep track of short and long-term tasks while maintaining a record of the previous and current code on Git. Weekly, our team will hold meetings to discuss individual and overall project progress.

Each week, student members have responsibilities that are rotated amongst each other. The scribe of the week has the responsibility of writing down all the notes we talk about during our meetings to be able to reflect on them afterward. The leader of the week has to take on the responsibilities of creating an organized agenda and leading the meeting while making sure to stay on agenda. Since the scribe took notes on everything said during the meeting, they are most prepared to be the leader of the next meeting. This creates a rotation where the scribe becomes the leader and a new scribe is chosen for the following meeting. We have chosen to meet over Zoom for both the sponsor/faculty and full student team meetings. We have created a student and mentor Google Drive for all weekly agendas, follow-ups, slides, weekly roles, and scribe notes. To share files and communicate with the team and sponsors, we will primarily be using Microsoft Teams.

### Project Deliverable

We coded a data pipeline that obtains the raw SWRPY87 messages from Railinc, extracts the required information, transforms the data, and loads it into the Main Equipment Event Table. Below is a diagram showing the 6 primary steps in our data pipeline. The cylinders represent the data tables used in our project. Please note that step 7 is our stretch goal and will be covered later in this section. The numbering of steps in the below diagram also corresponds to the Requirements in Section 2: Detailed Discussion of Requirements.



The process of taking in messages sent by Railinc and storing them in a receiver table has already been completed by our sponsors. Therefore, this step is considered out of scope. The 7 numbered steps within the diagram, on the other hand, are all within the scope of this project.

The implementation for steps 1-5 has been completed and our code has passed a set of test cases and preliminary audits. This means we have completed our first two deliverables: complete code for the pipeline and documentation on our project. To officially validate these

steps, our code needs to pass a set of audits for 6 months of January 2022 - June 2022 data. Validation progress and details can be found in Appendix B and C. Step 6: Data Analysis corresponds with our third expected deliverable: a Data Analytics report on the benefits of SWRPY87 messages. Out of the tasks required to complete this step, the cohort has learned Tableau, met with the Finance Team to gain a basic understanding of what is expected in the data analytics report, and created a startling report.

As detailed in the above paragraph, we have delivered all three of the expected deliverables for our ETL pipeline. After this, the UP MDP Cohort moved on to our stretch goal: creating an additional queue of railcar data for specific railcars to run on the same pipeline. We can receive data more frequently from Railinc if we specify only a few specific railcars. Due to this, creating an additional queue for specific railcars allows us to provide the Finance Team with more real-time data, which can expedite the revenue auditing process. The two deliverables for the stretch goal are the code for the queue and appropriate documentation for the code. We have completed the portions of our stretch goal, by creating an architecture that connects the Finance's backend system to the queue that would send the specific railcar request message. However, we were not able to fully code up a pipeline that would automate the request of the interested railcars. We did document our progress and findings and handed them off to the engineers at Union Pacific.

## Section 2: Outline of the Project Requirements

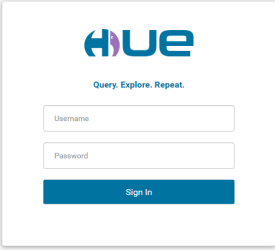

Since this was a year-long project where the team worked with the sponsored company Union Pacific. The team has developed a set of requirements to ensure the understanding of the project's expectations between the sponsors and the student team. In addition, the set of requirements also helps confirm that the team was able to achieve the goals the sponsors had. Following the structure of our data pipeline, we split our requirements into three subsections: Original Data Pipeline, Data Analytics, and Real-Time Data Queue. This is the table that details our requirements:

<b>TABLE 2: Complete List of Requirements</b>			
<b>Requirement Number</b>	<b>Requirement Section</b>	<b>Requirements Target &amp; Units</b>	<b>Origin of Validation Method</b>
1	Data Pipeline	100% raw messages parsed	Sponsor Developed
2		100% of non-ghosted messages have a unique eqmt_init, eqmt_nbr, and main_date_time	Sponsor Developed
3		> 98% messages with the correct city and state names	Student Developed
4		For each of the 6 matching cases, find 2 random examples of successful matching	Sponsor Developed
5		All Audits on each day run successfully	Student Developed
6a	Data Analytics	20% new messages everyday	Student Developed
6b		Find 5 most frequently offline railcar carriers	Sponsor Developed
6c		5% of new messages are from tank cars	Sponsor Developed
7	Real-Time Data Queue	100% of requested messages received	Sponsor Developed

### Section 3: Data Collection, Analysis Methods, and Requirements Validations

Due to a large number of requirements, we will only select the top three most critical requirements and present how we collect and analyze data to validate them. The top three requirements as indicated by the sponsors are requirement 3 (> 98% messages with the correct city and state names), requirement 5 (all message audits on each day run successfully), and requirement 6a (20% new messages every day). The tables below illustrate the team’s approach to each requirement.

#### Requirement 3: Correct City and State Names

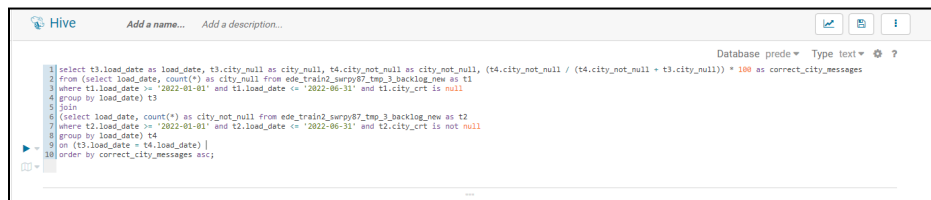
<b>Title:</b>	Data Pipeline Requirement 3 – Correct City and State Names
<b>Specific User Objective/Project Requirement:</b>	At least 98% of daily messages between January 2022 to June 2022 have city and state names that match UP’s location data table (Data Pipeline Subsection; Requirement 3)
<b>Pass/Fail Definition:</b>	<p><b>Pass:</b> For all the days between January 2022 to June 2022, each daily messages have <math>\geq 98\%</math> correct and matched city names</p> <p><b>Fail:</b> For any day between January 2022 to June 2022, each daily messages have <math>&lt; 98\%</math> correct and matched city names</p>
<b>Validation Method:</b>	<ol style="list-style-type: none"> <li>Log in to UP’s virtual machine with the given credentials.</li> <li>Navigate to (<a href="http://go/hue4test">http://go/hue4test</a>).</li> </ol> <div data-bbox="501 1171 1424 1541" style="border: 1px solid black; padding: 10px; margin: 10px 0;">  </div> <ol style="list-style-type: none"> <li>Login with UP’s credentials to access Hue4 for the test server.</li> </ol> <div data-bbox="501 1656 1424 1808" style="border: 1px solid black; padding: 10px; margin: 10px 0;">  </div> <ol style="list-style-type: none"> <li>Enter this SQL script into the Hive console.</li> </ol>

**Title:**

## Data Pipeline Requirement 3 – Correct City and State Names

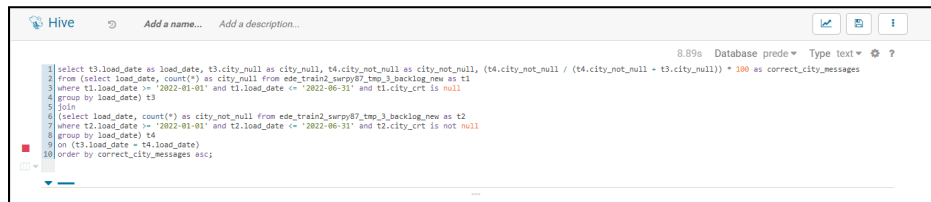
```
select t3.load_date as load_date, t3.city_null as city_null, t4.city_not_null as city_not_null,
(t4.city_not_null / (t4.city_not_null + t3.city_null)) * 100 as correct_city_messages
from (select load_date, count(*) as city_null from
ede_train2_swrpy87_tmp_3_backlog_new as t1
where t1.load_date >= '2022-01-01' and t1.load_date <= '2022-06-30' and t1.city_crt is
null
group by load_date) t3
join
(select load_date, count(*) as city_not_null from
ede_train2_swrpy87_tmp_3_backlog_new as t2
where t2.load_date >= '2022-01-01' and t2.load_date <= '2022-06-30' and t2.city_crt is not
null
group by load_date) t4
on (t3.load_date = t4.load_date)
order by correct_city_messages asc;
```

5. Click the blue run button on the bottom left of the console to start the query.



The screenshot shows the Hive console interface. At the top, there are fields for 'Add a name...' and 'Add a description...'. Below that, the SQL query from the previous block is pasted into the console. The query is highlighted in blue. At the bottom left of the console, there is a blue play button icon, which is the 'run' button mentioned in the instruction.

6. Wait for the query to return results (should return within 10 - 15 minutes depending on the availability of the server).



The screenshot shows the Hive console interface. The SQL query is still visible. At the top right, the execution progress is shown as '8.89s Database prede'. At the bottom left, the blue play button icon is now a red square, indicating that the query has finished running.

7. Once the query finishes running, results should show up under the query console.

**Title:**

### Data Pipeline Requirement 3 – Correct City and State Names

```
group by load_date) t4
on (t3.load_date = t4.load_date)
order by correct_city_messages asc;
```

	load_date	city_null	city_not_null	correct_city_messages
1	2022-02-11	93585	5956798	98.45323841482431
2	2022-02-12	127688	8568229	98.53163271912554
3	2022-02-06	104401	7097105	98.55028934225703
4	2022-02-07	129679	8819365	98.55091784105655
5	2022-02-10	141293	9790370	98.57734802318605
6	2022-02-18	142179	9857597	98.57817815119058
7	2022-02-08	143382	9958918	98.58069944468092
8	2022-02-16	141191	9819035	98.58245184396418
9	2022-02-19	117861	8270932	98.59501837749484
10	2022-02-14	140693	9898052	98.5985001113187
11	2022-02-15	142668	10042877	98.59930911895239
12	2022-02-09	139001	9790830	98.60016751543908
13	2022-02-11	145154	10236398	98.60180828454166
14	2022-02-20	79688	5629005	98.60409379169627

8. Verify that for that date the number of messages with correct city names is greater than 98% by observing the last column (*correct\_city\_messages*). Because the results are sorted with ascending values of *correct\_city\_messages*, if the results of the first few rows return statistics that are greater than or equal to 98%, then the team has successfully passed the requirement. Otherwise, if any of the statistics is less than 98%, then the team has failed to meet the requirement.

**Data Collected:**

There are four columns in the query results: *load\_date*, *city\_null*, *city\_not\_null*, *correct\_city\_messages*.

*Load\_date* is the date on which the messages were loaded into the data pipeline. We often use this field to categorize the input messages.

*City\_null* is the number of messages that the developed algorithm cannot find a match with a correct city name in UP's database.

*City\_not\_null* is the number of messages that the developed algorithm has found that matches the correct city name in UP's database.

*Correct\_city\_messages* is calculated by dividing *city\_not\_null* by the summation of *city\_null* and *city\_not\_null*. This gives the percentage of the number of messages in which the algorithm was able to match with correct city names.

**Data Analysis:**

Since the query sorts the result with ascending values of *correct\_city\_messages*, we would only need to check the first few rows of our result as the number of messages with less *correct\_city\_messages* percentage would show up first. The 98% target value is firm, meaning that if any *load\_date* has *correct\_city\_messages* that are less than 98%, we would need to talk with our sponsors on re-evaluating the design algorithm design for better data accuracy.





**Title:** Data Pipeline Requirement 5 - All Audits run successfully for 6 months of historical data

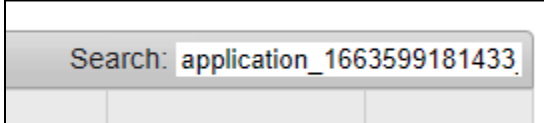
```

2022-10-02 18:38:15 INFO Client:54 -
client token: N/A
diagnostics: N/A
ApplicationMaster host: 45.54.149.4
ApplicationMaster RPC port: 0
queue: default
start time: 1664753870429
final status: UNDEFINED
tracking URL: http://omsdp130f5.uprr.com:8088/proxy/application_1663599181433_2251/
user: dede666
2022-10-02 18:38:16 INFO Client:54 - Application report for application_1663599181433_2251 (state: RUNNING)
2022-10-02 18:38:17 INFO Client:54 - Application report for application_1663599181433_2251 (state: RUNNING)
2022-10-02 18:38:18 INFO Client:54 - Application report for application_1663599181433_2251 (state: RUNNING)
2022-10-02 18:38:19 INFO Client:54 - Application report for application_1663599181433_2251 (state: RUNNING)

```

Here the application ID is *application\_1663599181433\_2251*.

6. Navigate to Hadoop Scheduler (<http://go/hscheduler>) and use the application ID to look up the application.



7. Select the application log and scroll down to **stdout** section.

```

Log type: stdout
Log Upload Time: Sun Oct 02 05:35:34 -0500 2022
Log Length: 4502
Showing 4508 bytes of 8502 total. Click here for the full log
rg to connect to metastore with URI thrift://omsd130e.uprr.com:9083
2022-10-02 05:30:25 INFO metastore:v72 - Connected to metastore.
2022-10-02 05:30:25 INFO metastore:v76 - Metastore configuration hive.metastore.filter.hook changed from org.apache.hadoop.hive.metastore.DefaultMetastoreFilterHookImpl to org.apache.hadoop.hive.security.authorization.plugin.AuthorizationMetastoreFilterHook
2022-10-02 05:30:25 INFO metastore:v76 - Trying to connect to metastore with URI thrift://omsd130e.uprr.com:9083
2022-10-02 05:30:25 INFO metastore:v42 - Failed to connect to the Hadoop server...
2022-10-02 05:30:25 INFO metastore:v76 - Trying to connect to metastore with URI thrift://omsd130e.uprr.com:9083
2022-10-02 05:30:25 INFO metastore:v42 - Connected to metastore.
2022-10-02 05:30:25 INFO metastore:v76 - Trying to connect to metastore with URI thrift://omsd130e.uprr.com:9083
2022-10-02 05:30:25 INFO metastore:v42 - Failed to connect to the Hadoop server...
2022-10-02 05:30:25 INFO metastore:v76 - Trying to connect to metastore with URI thrift://omsd130e.uprr.com:9083
2022-10-02 05:30:25 INFO metastore:v42 - Connected to metastore.
hivefalllover
me3
null

```

8. Check if there is a log output indicating a mismatched number of messages, specifically *"Mismatch source and dest message count. Sending an email to the CORE team"*. If the error log message exists, then the team has failed to achieve this requirement.

9. Repeat the process by running three other command line inputs on the Superputty Test server at **/upapps/hdp/prede/ede-event-rct/bin**. These inputs will include the start load date, end load date, source table, and destination table:

*Command 1: ./runSWRPY87AuditUtility.sh "2022-01-01" "2022-06-30" "prede.src\_rlc\_eqmt\_evt\_rct" "prede.ede\_train2\_swrpy87\_tmp\_2\_backlog"*

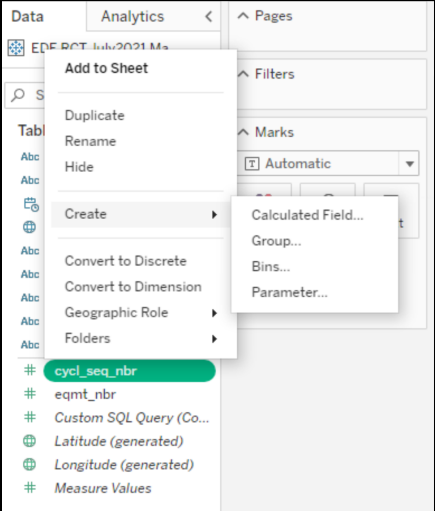
*Command 2: ./runSWRPY87AuditUtility.sh "2022-01-01" "2022-06-30" "prede.ede\_train2\_swrpy87\_tmp\_2\_backlog" "prede.ede\_train2\_swrpy87\_tmp\_3\_backlog"*

*Command 3: ./runSWRPY87AuditUtility.sh "2022-01-01" "2022-06-30"*

<b>Title:</b>	Data Pipeline Requirement 5 - All Audits run successfully for 6 months of historical data
	<p><i>"prede.ede_train2_swrpy87_tmp_3_backlog"</i>  <i>"prede.ede_train2_swrpy87_stg_backlog"</i></p> <p>10. Check the application’s output log after it finishes running for mismatched messages error. If an error message is identified in any of these applications, then the team has failed to meet this requirement.</p>
<b>Data Collected:</b>	The output log of each four audit applications.
<b>Data Analysis:</b>	<p>To validate this requirement, one would need to search through each of the audit application’s output logs for a mismatched message error (<i>"Mismatch source and dest message count. Sending an email to the CORE team"</i>). The target value for this requirement is that the number of messages processed by the pipeline remains constant. Hence, any one mismatched message error would indicate that our pipeline is either losing messages or generating duplicate ones and would indicate a failure to meet the requirement. If no error messages are found, then the requirement passes.</p>

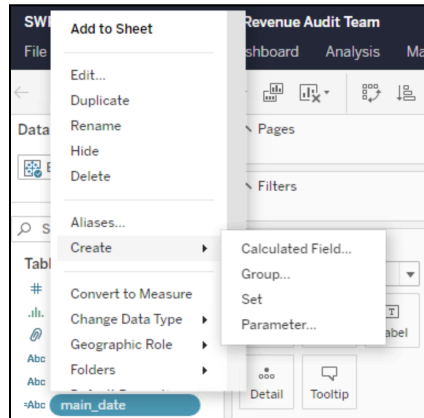
**Requirement 6: 20% new messages every day**

<b>Title:</b>	Data Pipeline Requirement 6 - 20% new messages every day
<b>Specific User Objective/Project Requirement:</b>	20% of the messages processed each day between January 2022 to April 2022 is new
<b>Pass/Fail Definition:</b>	<p><b>Pass:</b> For all the days between January 2022 to April 2022, 20% of the messages in a given day must be new to the Main Equipment Event Table</p> <p><b>Fail:</b> For any day between January 2022 to June 2022, one of the days have less than 20% new messages to be Main Equipment Event Table</p>
<b>Validation Method:</b>	<ol style="list-style-type: none"> <li>1. Log in to UP’s virtual machine with the given credentials.</li> <li>2. Open up Tableau.</li> <li>3. In your desired folder, or “Personal Space”, click “Create Workbook”.</li> <li>4. Find the RCT data tables in the “Connect Data” pop-up. If the data is not yet a source, ask a manager at UP to import it for you. Due to the high volume of data, messages should be imported month by month (so each data source is one month).</li> <li>5. The column <code>cycl_seq_nbr</code> is what we use to determine if a message is</li> </ol>

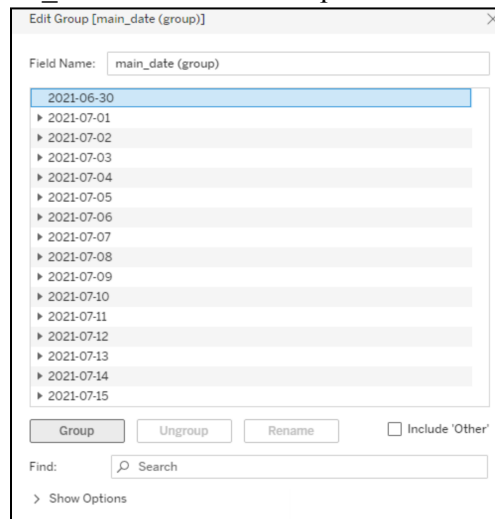
<b>Title:</b>	Data Pipeline Requirement 6 - 20% new messages every day
	<p>new to the Main Equipment Event Table or not. In step 4, we match the Main Equipment Event Table messages to our SWRPY87 messages. Matched messages are given the Main Equipment Event Table cycl_seq_nbr. Messages that could not be matched with a message in UP's main equipment event table, aka a new message, are given the default cycl_seq_nbr 999999999. That means matched messages will have a non-default cycl_seq_nbr.</p> <p>6. To calculate the number of new messages, we are going to create a new field to our data by clicking “cycl_seq_nbr”, then “Create”, then “Calculated Field”.</p>  <p>7. In the popup, name this new field “new_col”. The calculation is:</p> <pre> IF [cycl_seq_nbr]=999999999 THEN 1 ELSE 0 END </pre> <p>8. If the column “main_date” was not imported, you need to create another “Calculated Field”, this time clicking “main_date_time”, then “create”, then “Calculated Field” to create it.</p> <p>9. In the popup, name this new field “main_date”. The calculation is:</p> <pre> SPLIT(STR([main_date_time]), ' ', 1) </pre> <p>10. The messages need to be grouped by the “main_date” to find the overall percentages for each day. To do so, click “main_date” then “Create” then “Group”.</p>

**Title:**

Data Pipeline Requirement 6 - 20% new messages every day

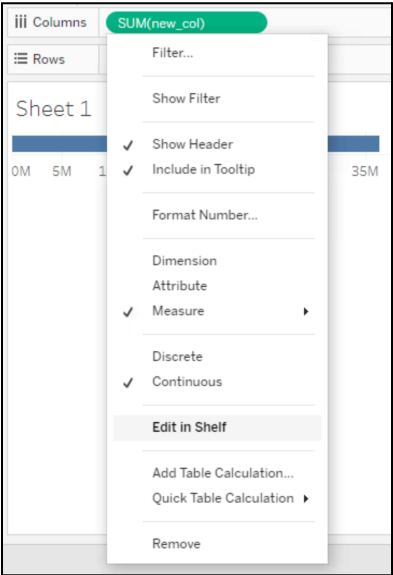


11. Click each main\_date then click “Group”.



10. Now we have all the fields so it is time to create the graph. In the “Columns” section, drag and drop the “new\_col” field. We need to calculate the percentage of new messages here by clicking “Edit in Field”.

**Title:** Data Pipeline Requirement 6 - 20% new messages every day



- 11. Edit the shelf to read:  
$$\text{SUM}([\text{new\_col}]) / (\text{COUNT}([\text{cycl\_seq\_nbr}]))$$
- 12. In the “Rows” section (below “Columns”), drag the grouped main\_date field we created.
- 13. Repeat this same process for every single month.

**Data Collected:** We will be analyzing the dates and cycl\_seq\_nbr of every SWRPY87 message from January 2022 to April 2022. We will also need some messages from 90 days after March 31st, 2022. This is because there is at most a 90-day delay between the date the message occurred and the date the message is received by our pipeline. In order to run our check of 20% new messages on a given day, we must process the 90 days following that date to ensure we are running the check on all messages from that day.

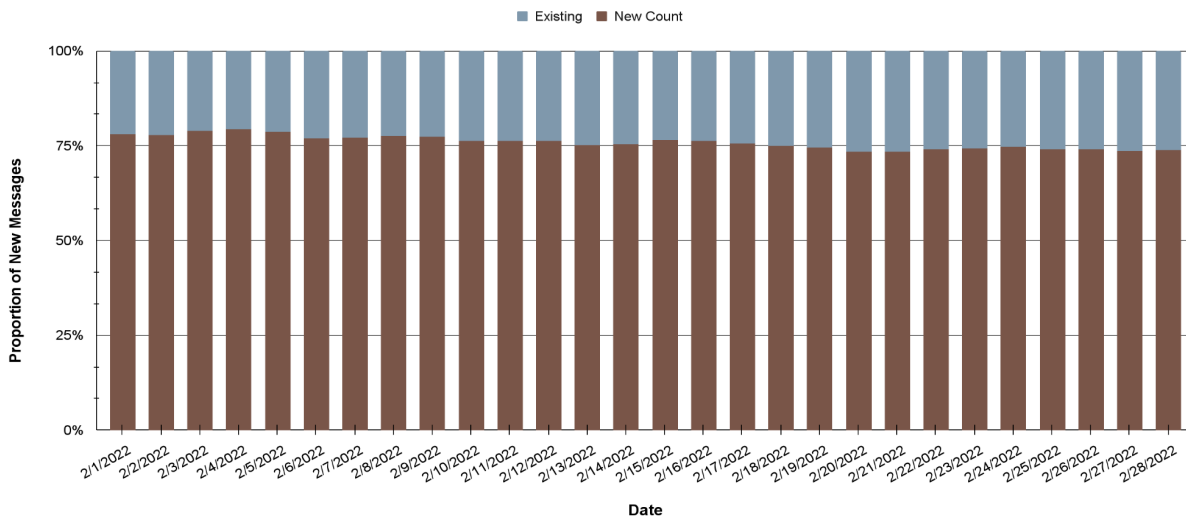
**Data Analysis:** If our process fails to meet this 20% requirement on a given day, we need to review how we determine matching messages and find messages that were incorrectly classified as matching or not matching. We use Railinc as our truth source to determine the correct classification. Since we get 6-10 million messages per day, it would be infeasible to validate the classification of every single message on a given day. Instead, we are going to check the messages that are on the border of the classification line. In our project, we match 8 fields or message values, so perfectly matched messages get a score of 8. We drew the classification line at a matching score of 5, so messages with a score of 5-8 were considered matched and messages with a score of 0-4 were considered not matched. If our project fails to bring in 20% new messages on a given day, we will go

<b>Title:</b>	Data Pipeline Requirement 6 - 20% new messages every day
	<p>through matching indicators 4 and 5 to evaluate if our classification line needs to be shifted.</p> <p>If our process succeeds, there is no further evaluation needed. We need to present the findings to the Finance Team for their feedback.</p>

## Section 4: Project Impact and Concluding Remarks

Union Pacific initiated this project with the goal of giving the Finance team data that is not currently available in their main data table. With this goal, the MDP team uses data engineering to extend UP's current data pipeline for raw data from Railinc Corporation (the provider of rail data for the entire North American rail industry), specifically, the project will be dealing with the rail car tracing (aka RCT) data from Railinc, which is a data set about offline train car events. Offline train car events refer to events that happen to individual UP train cars on a non-UP rail track. UP currently has a data pipeline that continuously streams raw RCT data into a table. The team has converted this current table of unprocessed RCT data into a new table of processed data in the EDE database, which is a UP database of all railcar-related data. For the data analytics part, our team has performed data analysis on the offline event data that we pipeline, putting our findings into an analytical report.

### Outcome: Exceeded Finance Team's Requirement of 20% New Messages Everyday



**Achieved Average:**  
5.5 million new messages/day

This graph shows the proportion of new messages the team pipelined into the EDE system. As you can see, the project really had a significant impact on the data that is available to Union Pacific's Finance team. Initially, the sponsors and the Finance team wanted at least 20% of the messages that we pipelined into the main data table each day to be new messages that are about rail cars that have gone onto another rail company's tracks and are not currently anywhere in the main data table. However, about 75% of the messages we are pipelining in are new messages to the main data table, meaning that each day, the team provides the finance team with about 5.5 million new messages that can help them find missing billings. This makes their work



much faster as they don't need to manually look for the same data on Railinc's website anymore and can instead just query from the main data table.