Case study - Bike Share

Data cleaning and manipulation using KNIME Analytics Platform

Capstone project for the Google Data Analyst Professional Certificate

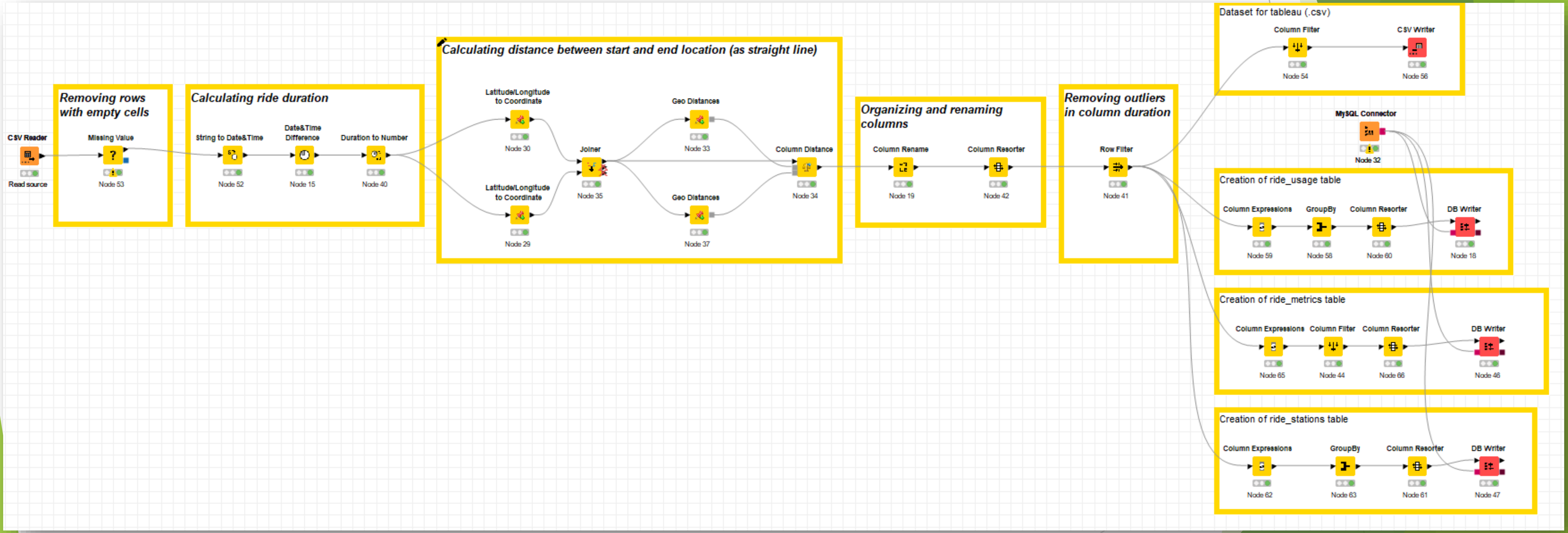
Paulo Argôlo

https://www.linkedin.com/in/pauloargolo

July 2023

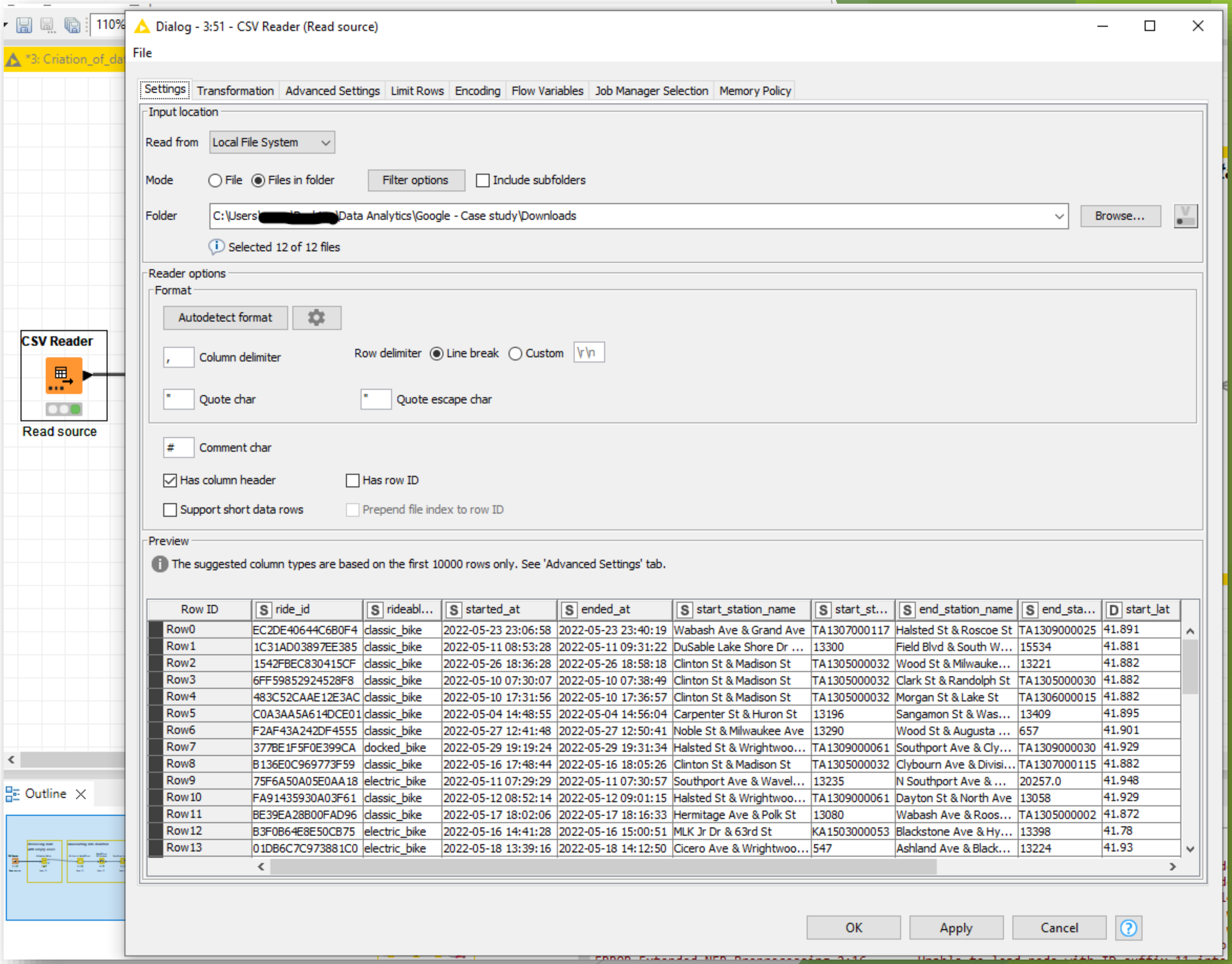
Complete workflow created in KNIME

See next slides for detailed information on the nodes used



CSV Reader

- In the CSV Reader node I was able to set the folder where the 12 csv files are placed as source for the data ingestion.
- In there preview below I could check the data types of each row and plan the necessary steps for data cleaning and manipulation.



Dialog - 3:51 - CSV Reader (Read source)

Input location

Read from: Local File System

Mode: ○ File ● Files in folder □ Include subfolders

Folder: C:\Users\[redacted]Data Analytics\Google - Case study\Downloads

Selected 12 of 12 files

Reader options

Format

Column delimiter: ,

Row delimiter: ● Line break ○ Custom \r\n

Quote char: "

Quote escape char: "

Comment char: #

☑ Has column header □ Has row ID

□ Support short data rows □ Prepend file index to row ID

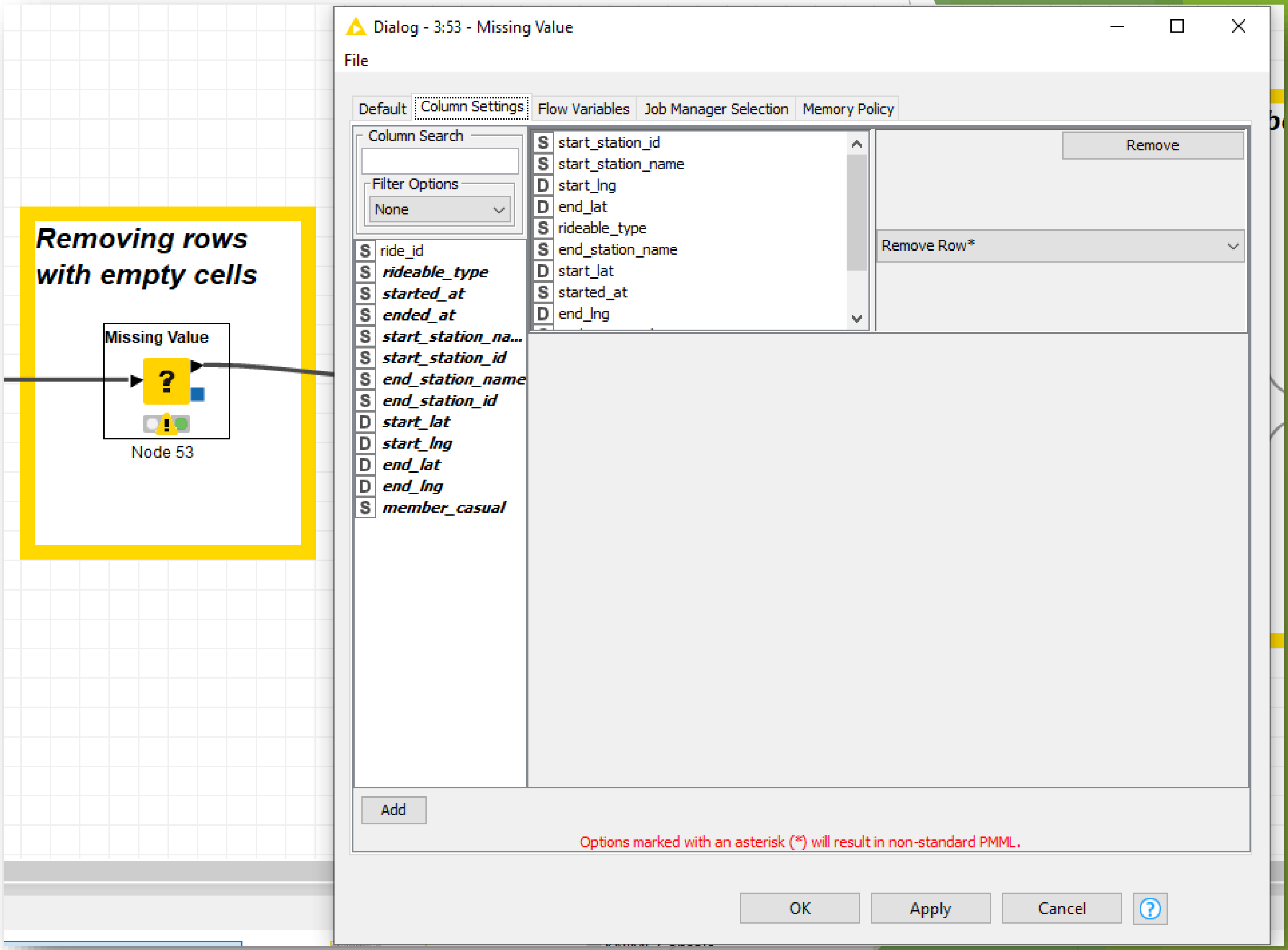
Preview

The suggested column types are based on the first 10000 rows only. See 'Advanced Settings' tab.

Row ID	S ride_id	S rideabl...	S started_at	S ended_at	S start_station_name	S start_st...	S end_station_name	S end_sta...	D start_lat
Row0	EC2DE40644C6B0F4	classic_bike	2022-05-23 23:06:58	2022-05-23 23:40:19	Wabash Ave & Grand Ave	TA1307000117	Halsted St & Roscoe St	TA1309000025	41.891
Row1	1C31AD03897EE385	classic_bike	2022-05-11 08:53:28	2022-05-11 09:31:22	DuSable Lake Shore Dr ...	13300	Field Blvd & South W...	15534	41.881
Row2	1542FBEC830415CF	classic_bike	2022-05-26 18:36:28	2022-05-26 18:58:18	Clinton St & Madison St	TA1305000032	Wood St & Milwauke...	13221	41.882
Row3	6FF59852924528F8	classic_bike	2022-05-10 07:30:07	2022-05-10 07:38:49	Clinton St & Madison St	TA1305000032	Clark St & Randolph St	TA1305000030	41.882
Row4	483C52CAAE12E3AC	classic_bike	2022-05-10 17:31:56	2022-05-10 17:36:57	Clinton St & Madison St	TA1305000032	Morgan St & Lake St	TA1306000015	41.882
Row5	C0A3AA5A614DCE01	classic_bike	2022-05-04 14:48:55	2022-05-04 14:56:04	Carpenter St & Huron St	13196	Sangamon St & Was...	13409	41.895
Row6	F2AF43A242DF4555	classic_bike	2022-05-27 12:41:48	2022-05-27 12:50:41	Noble St & Milwaukee Ave	13290	Wood St & Augusta ...	657	41.901
Row7	377BE1F5F0E399CA	docked_bike	2022-05-29 19:19:24	2022-05-29 19:31:34	Halsted St & Wrightwoo...	TA1309000061	Southport Ave & Cly...	TA1309000030	41.929
Row8	B136E0C969773F59	classic_bike	2022-05-16 17:48:44	2022-05-16 18:05:26	Clinton St & Madison St	TA1305000032	Clybourn Ave & Divisi...	TA1307000115	41.882
Row9	75F6A50A05E0AA18	electric_bike	2022-05-11 07:29:29	2022-05-11 07:30:57	Southport Ave & Wavel...	13235	N Southport Ave & ...	20257.0	41.948
Row10	FA91435930A03F61	classic_bike	2022-05-12 08:52:14	2022-05-12 09:01:15	Halsted St & Wrightwoo...	TA1309000061	Dayton St & North Ave	13058	41.929
Row11	BE39EA28B00FAD96	classic_bike	2022-05-17 18:02:06	2022-05-17 18:16:33	Hermitage Ave & Polk St	13080	Wabash Ave & Roos...	TA1305000002	41.872
Row12	B3F0B64E8E50CB75	electric_bike	2022-05-16 14:41:28	2022-05-16 15:00:51	MLK Jr Dr & 63rd St	KA1503000053	Blackstone Ave & Hy...	13398	41.78
Row13	01DB6C7C973881C0	electric_bike	2022-05-18 13:39:16	2022-05-18 14:12:50	Cicero Ave & Wrightwoo...	547	Ashland Ave & Black...	13224	41.93

Missing Value

- While inspecting the data I could find several row where at least one value was missing.
- I made a decision to remove these rows using the Missing Value node based on the assumption the missing values are coming from some technical error and could steer the analysis in a certain direction.



Removing rows with empty cells

Missing Value

Node 53

Dialog - 3:53 - Missing Value

S ride_id
S rideable_type
S started_at
S ended_at
S start_station_na...
S start_station_id
S end_station_name
S end_station_id
D start_lat
D start_lng
D end_lat
D end_lng
S member_casual

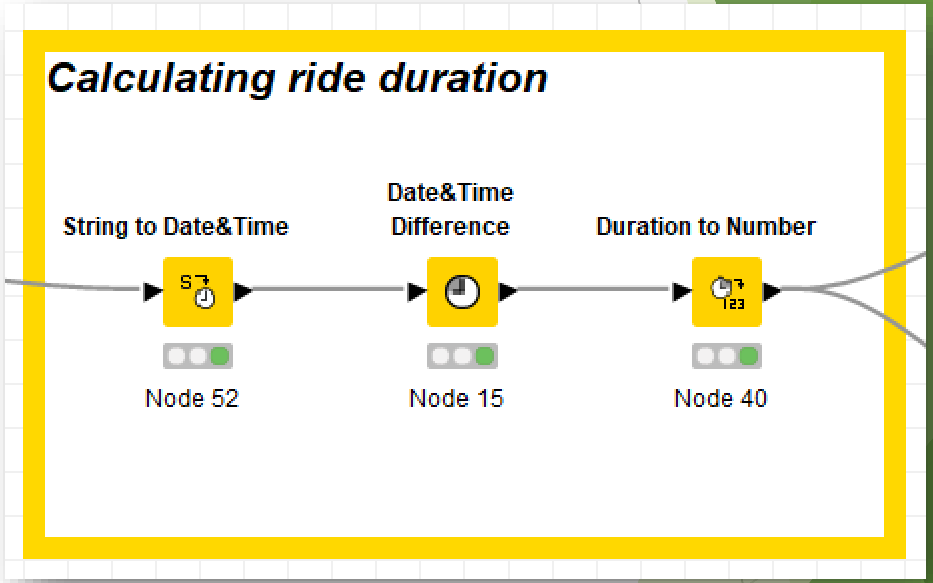
S start_station_id
S start_station_name
D start_lng
D end_lat
S rideable_type
S end_station_name
D start_lat
S started_at
D end_lng

Remove Row*

Options marked with an asterisk (*) will result in non-standard PMML.

Calculation of ride duration

- Here I used three nodes
    1. Converte columns with start and end timestamp into date&time type,
    2. calculate the duration of each ride as a new column based on the difference between end and start,
    3. create a new column labled "seconds" with the duration as INT.
- The second node in this subworkflow could be removed because MySQL could not interprete the data type duration. For the analysis in MySQL I Only used just the duration in seconds provided by the third node.



Calculating ride duration

Calculation of distance between pick-up and drop-off location

- This was one of the biggest advantes of using KNIME in my data manipulation process. Using the Palladian extension I managed to calculate the distance between pick-up and drop-off locations. This information lead to new insights regarding the behaviour of the customers.
- First I translated double values into coordinates of the start stations, then the end station and joined the two.
- Geo Distances and Column Distance calculate the distance between the coordinates as a straight line.
- To validate the data I used the omni calculator found in this link. I was amazed when the numbers were precisely the same! :) https://www.omnicalculator.com/other/latitude-longitude-distance

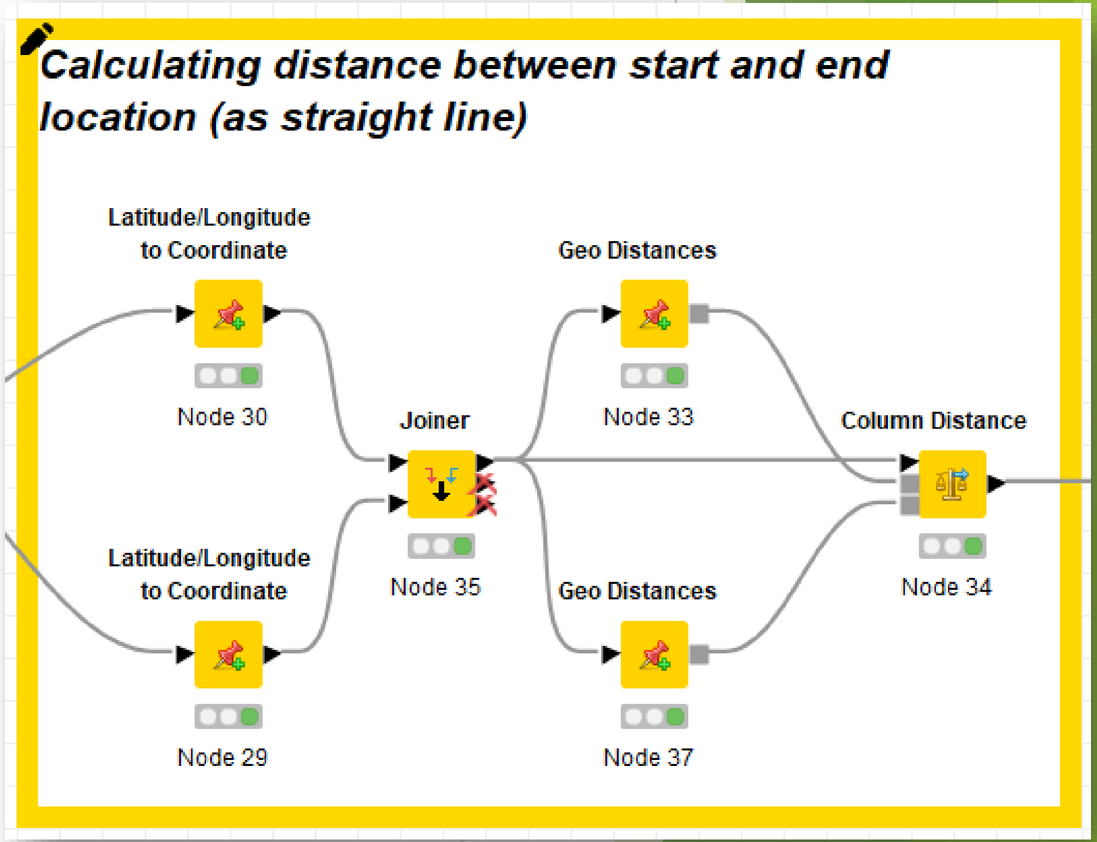
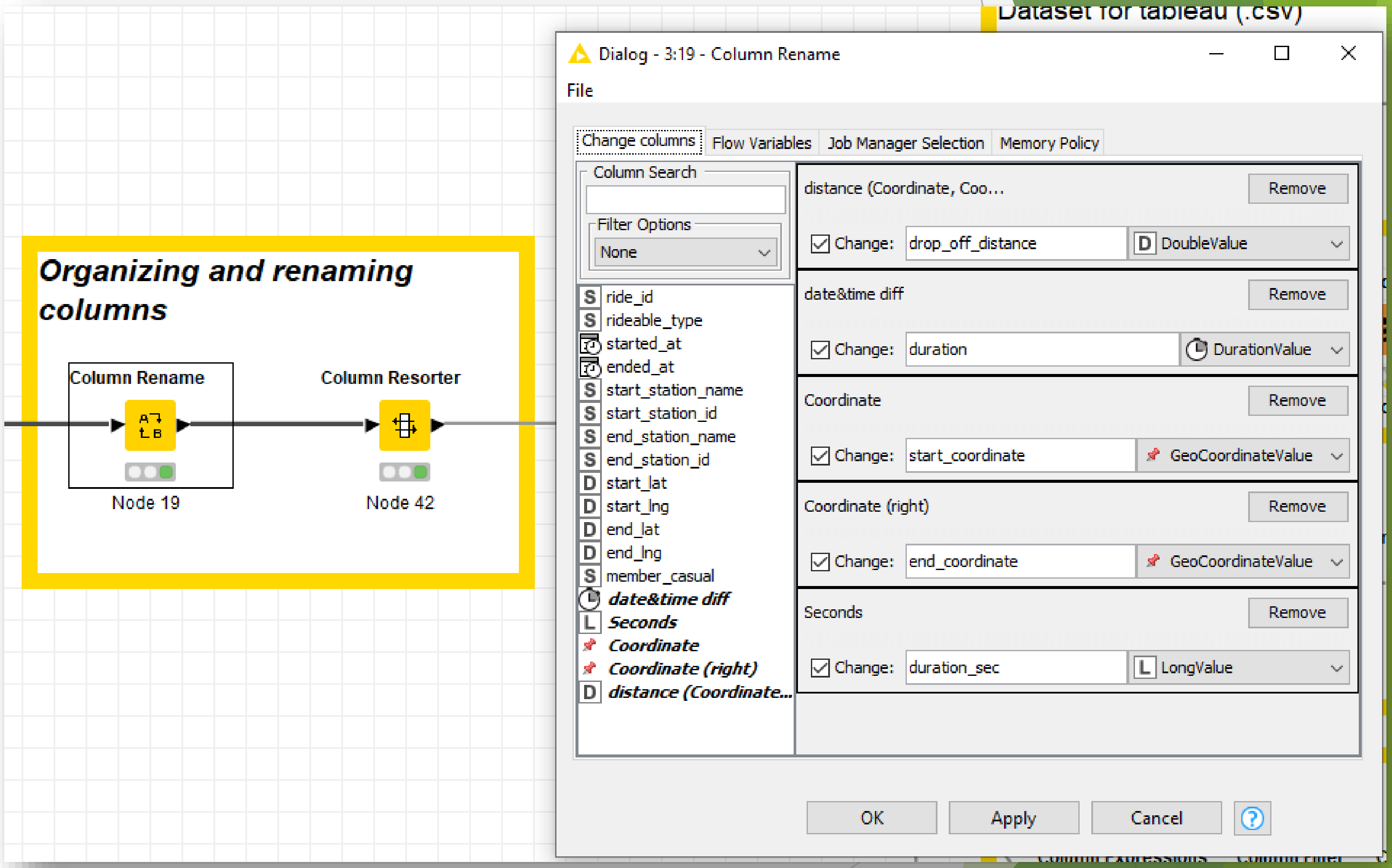

Calculating distance between start and end location (as straight line)
Latitude/Longitude to Coordinate
Node 30
Latitude/Longitude to Coordinate
Node 29
Joiner
Node 35
Geo Distances
Node 33
Geo Distances
Node 37
Column Distance
Node 34

Table structure

- In this step I Only reordered some of the columns and renamed a few so that it would be easier to work with them in MySQL and Tableau



Filter Options: None

S ride_id
S rideable_type
started_at
ended_at
S start_station_name
S start_station_id
S end_station_name
S end_station_id
D start_lat
D start_lng
D end_lat
D end_lng
S member_casual
date&time diff
L ***Seconds***
Coordinate
Coordinate (right)
D ***distance (Coordinate...***

distance (Coordinate, Coo...
☑ Change: drop_off_distance D DoubleValue

date&time diff
☑ Change: duration DurationValue

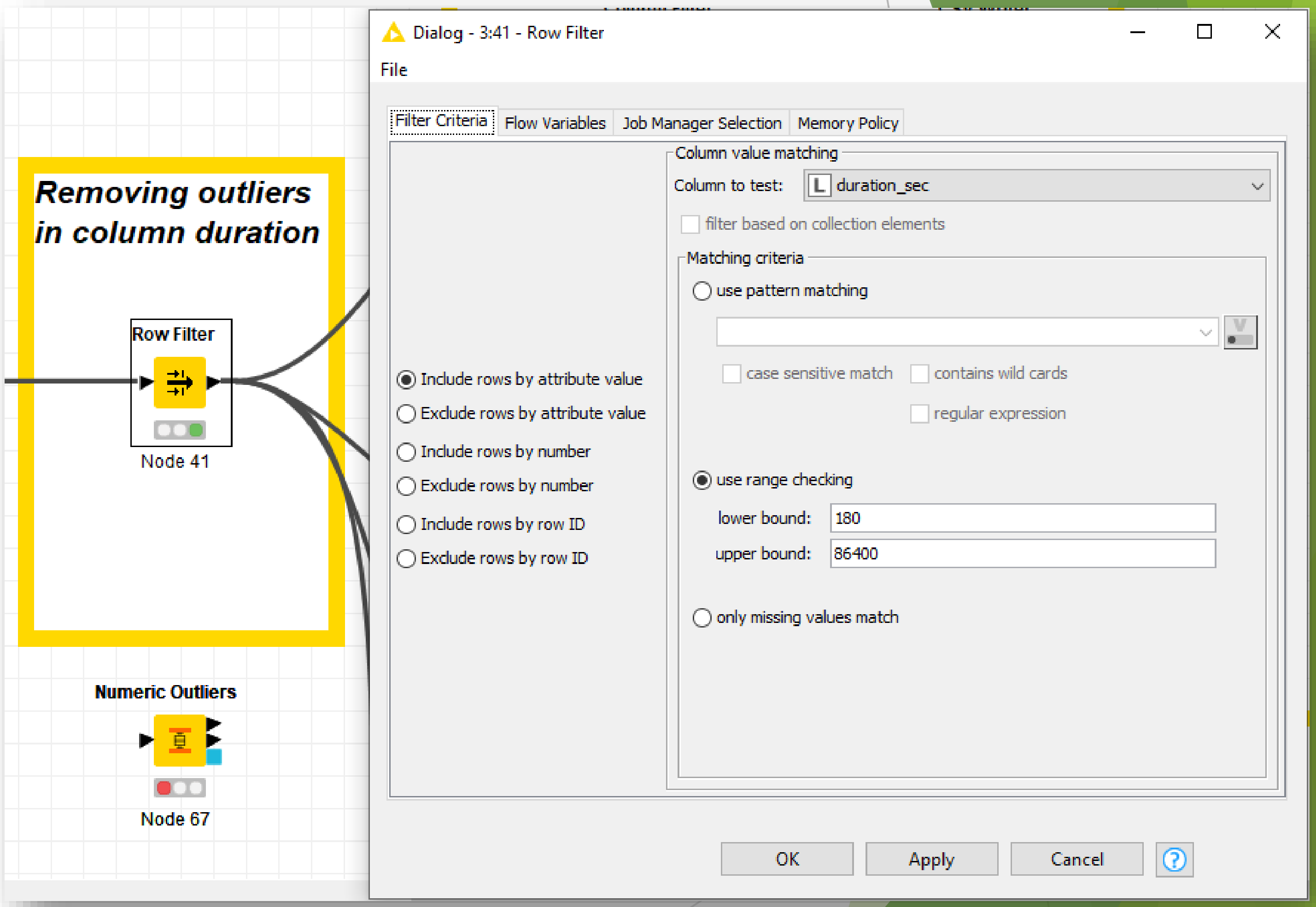
Coordinate
☑ Change: start_coordinate GeoCoordinateValue

Coordinate (right)
☑ Change: end_coordinate GeoCoordinateValue

Seconds
☑ Change: duration_sec L LongValue

Removing outliers in column “duration_sec”

- Initially I tried using the node “numeric outliers” to eliminate the outliers in the column “duration_sec” but I could not figure out the correct settings to work it properly.
- Alternatively I used a row filter based on the values. I kept rides with duration between three minutes and 24 hours. Values outside of this range I considered as system errors.



Removing outliers in column duration

● Include rows by attribute value
○ Exclude rows by attribute value
○ Include rows by number
○ Exclude rows by number
○ Include rows by row ID
○ Exclude rows by row ID

Column value matching

Column to test: duration_sec

☐ filter based on collection elements

Matching criteria

○ use pattern matching

☐ case sensitive match ☐ contains wild cards

☐ regular expression

● use range checking

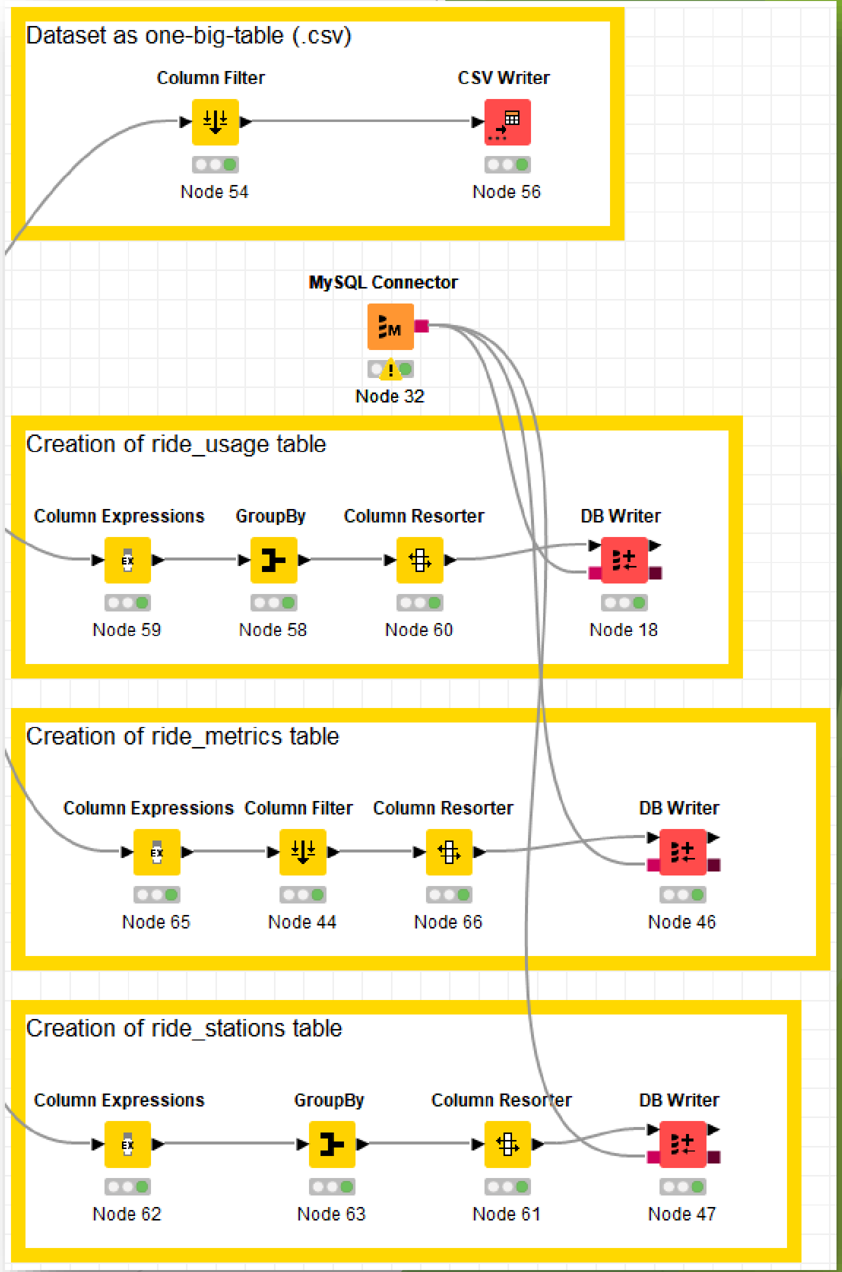
lower bound: 180

upper bound: 86400

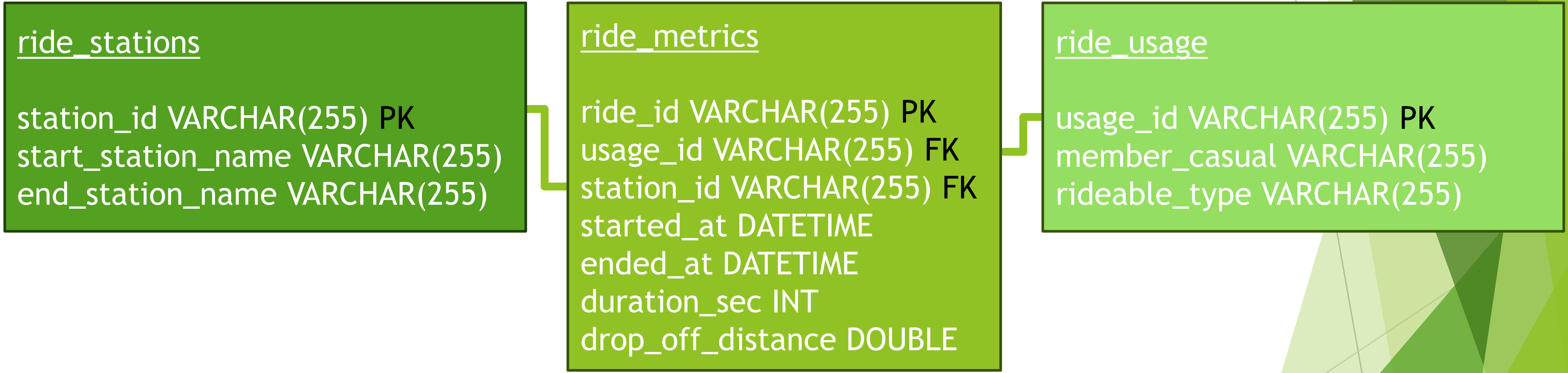
○ only missing values match

Creation of dataset

- In this last step I went with two approaches
  1. Creation of one-big-table as a reference to compare to the my first completion of this case study loading the 12 csv files directly to MySQL and doing the whole data cleaning and manipulation there, and
  2. Creation of three tables, splitting the content in the 12 csv files, and loading them directly to MySQL Server.
- In the second approach I could reduce a lot the number of rows in tables “ride_usage” and “ride_stations” resulting in a much smaller size of the complete data, in comparison to the one-big-table. More details in the next slide.



Data structure



ride_stations

station_id VARCHAR(255) PK
start_station_name VARCHAR(255)
end_station_name VARCHAR(255)

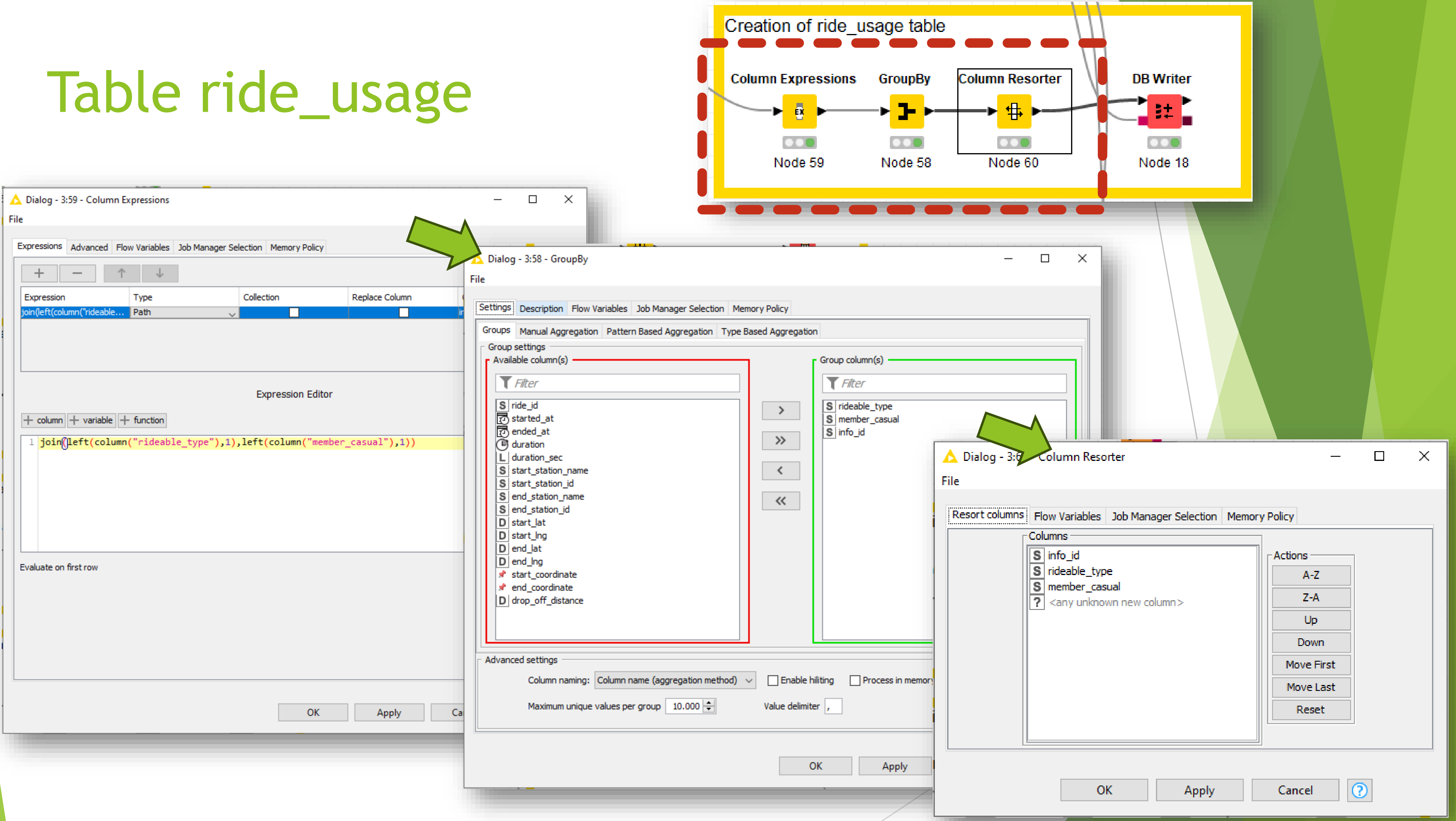
ride_metrics

ride_id VARCHAR(255) PK
usage_id VARCHAR(255) FK
station_id VARCHAR(255) FK
started_at DATETIME
ended_at DATETIME
duration_sec INT
drop_off_distance DOUBLE

ride_usage

usage_id VARCHAR(255) PK
member_casual VARCHAR(255)
rideable_type VARCHAR(255)

Table ride_usage



Dialog - 3:59 - Column Expressions

Expression	Type	Collection	Replace Column
join(left(column("rideable...	Path	□	□

Expression Editor

```
join(left(column("rideable_type"),1),left(column("member_casual"),1))
```

Evaluate on first row

Dialog - 3:58 - GroupBy

Group settings

Available column(s)

- S ride_id
- started_at
- ended_at
- duration
- L duration_sec
- S start_station_name
- S start_station_id
- S end_station_name
- S end_station_id
- D start_lat
- D start_lng
- D end_lat
- D end_lng
- start_coordinate
- end_coordinate
- D drop_off_distance

Group column(s)

- S rideable_type
- S member_casual
- S info_id

Advanced settings

Column naming: Column name (aggregation method)

□ Enable hiliting

□ Process in memory

Maximum unique values per group 10.000

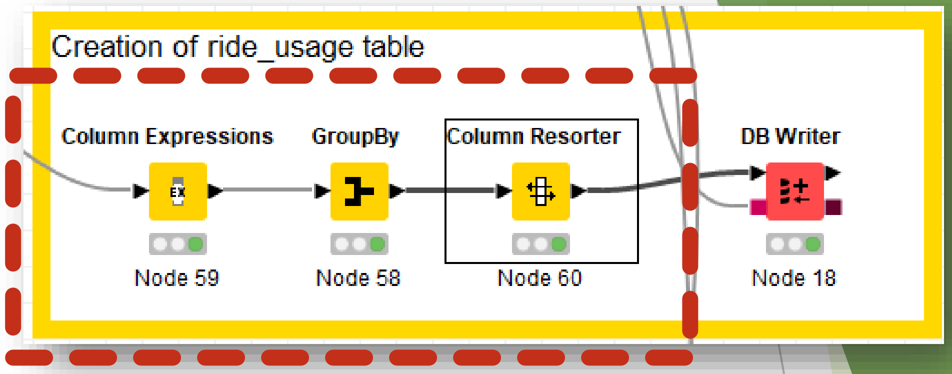
Value delimiter ,

Dialog - 3:60 - Column Resorter

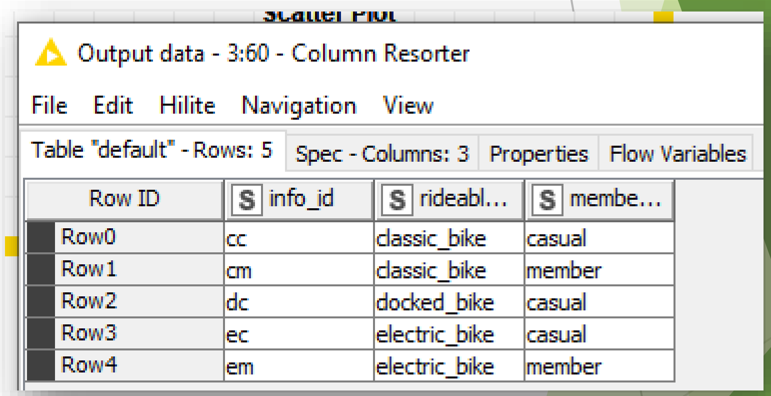
Columns

- S info_id
- S rideable_type
- S member_casual
- ? <any unknown new column>

Table ride_usage



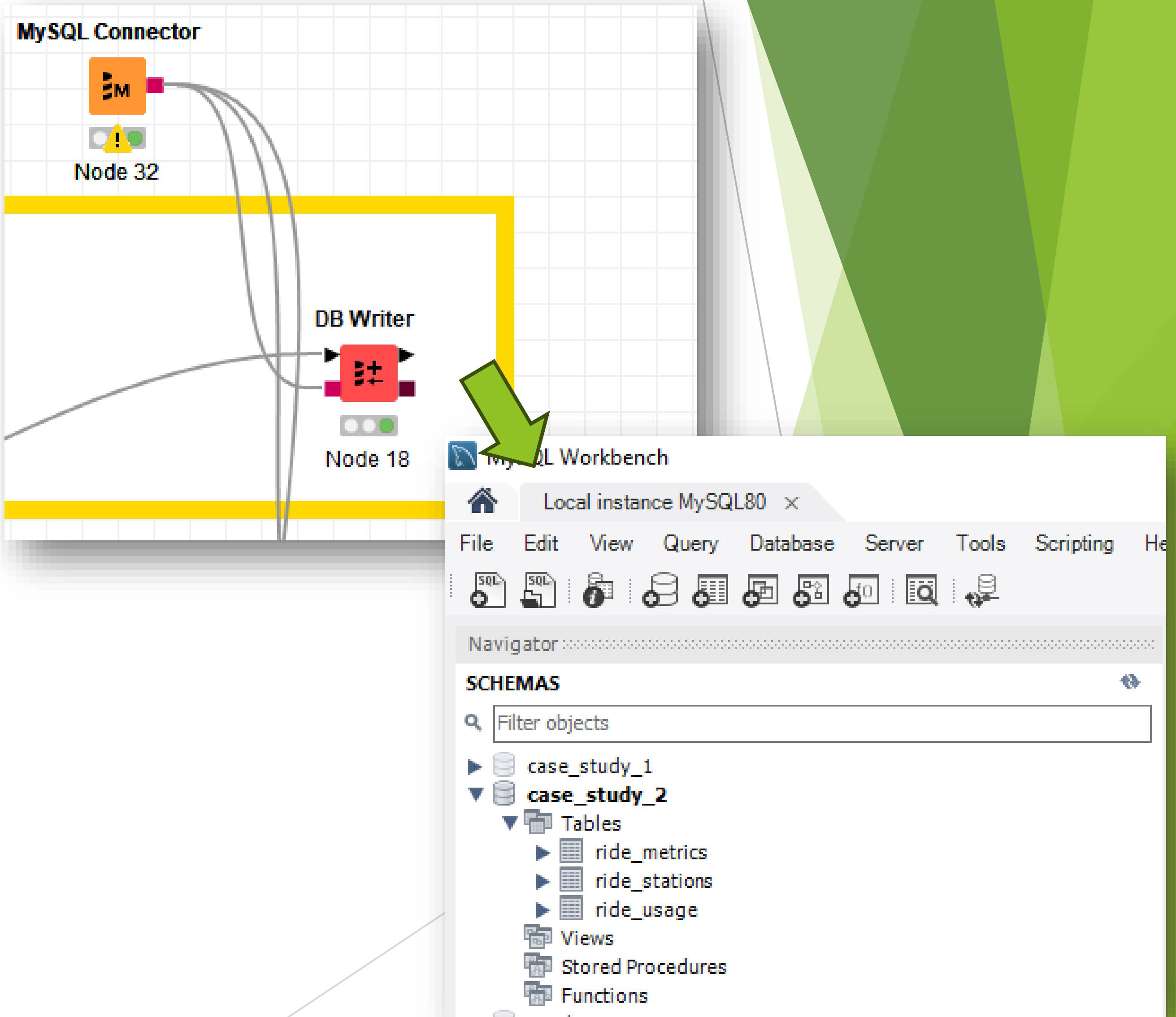
- I used the node Column Expressions to concatenate the first letters of the other two columns and create a unique key to be used as a primary key.
- With the Group By node I reduced from the 4.200.433 rows to the 5 rows that make this table possible to use in MySQL joining tables.
- The node Column Resorter was just to rearrange the columns having the primary key as the first column instead of the last.



Row ID	S info_id	S rideabl...	S membe...
Row0	cc	classic_bike	casual
Row1	cm	classic_bike	member
Row2	dc	docked_bike	casual
Row3	ec	electric_bike	casual
Row4	em	electric_bike	member

Loading the tables into MySQL

- With the MySQL Connetor I managed to enter my authentication for MySQL Server and connect it to the DB Writer node.
- After setting up the target data schema and table I could load the tables into MySQL and start writing queries.



Final thoughts

- I was really happy to use KNIME Analyics Platform and this second version of my case study using the Bike Share data. KNIME made loading the data into MySQL a load easier and the data cleaning and manipulation feel much more structured.
- I’m looking forward to what I will be able to accomplish with the next knowledge I acquire and the future tools I learn how to use.