ETL - Feeding the DW

With Tableau Prep

What is Tableau Prep

Tableau Prep is a commercial tool that is part of the Tableau suite

It provides an easy-to-use GUI to build data transformation pipelines

- There exist several ETL tools, some even open-source
  - Talend Open Studio
  - Pentaho Data Integration
- Open-source alternatives are more advanced and require a steeper learning curve
- Tableau Prep is more limited but easier to use
  - Renewable one-year academic license is provided to all students and academic staff

What we are going to do

Guided exercises

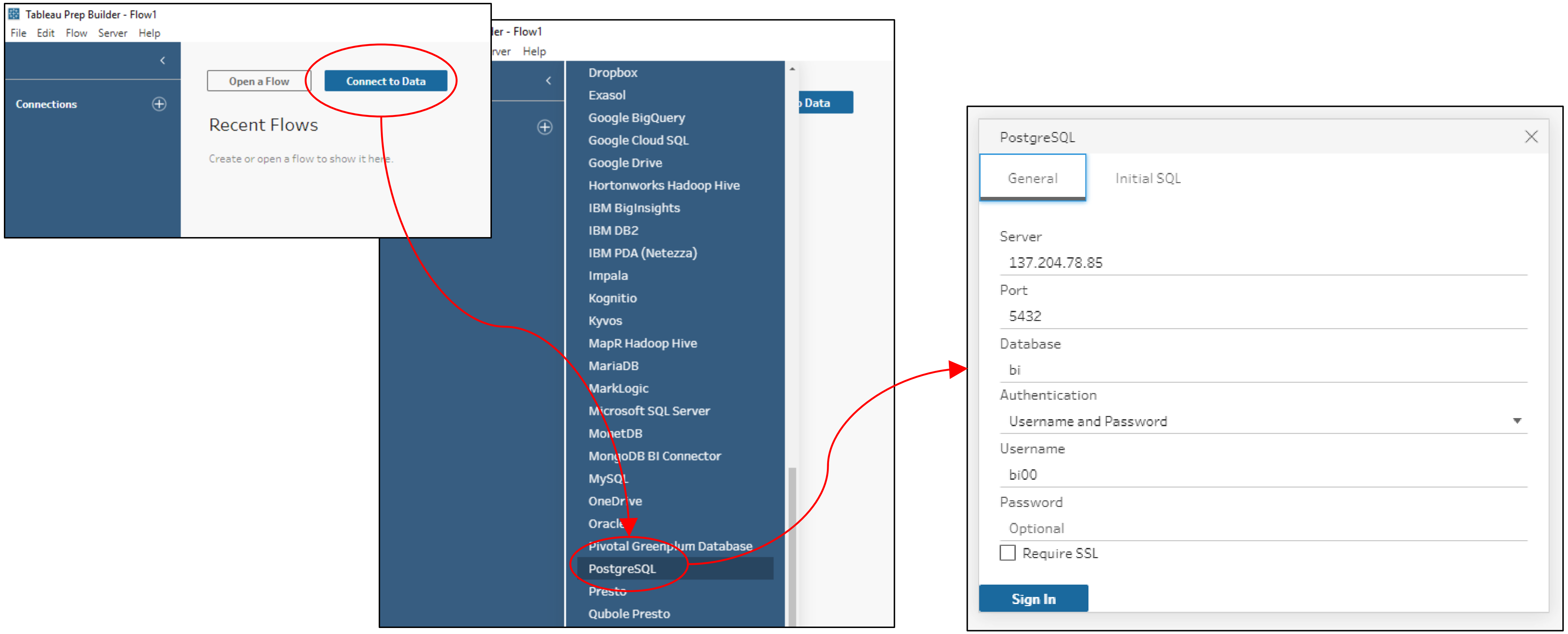
- Tableau Prep Basics
- The first flow (DT_PART)
- Incremental feeding (DT_PART)
- Surrogate keys (DT_PART)

Individual exercises

- ETL flows for the remaining DTs (and FT) in the Sales cube
- ETL flows for the Orders cube

Tableau Prep - Basics

Connect to the DB



PostgreSQL

Server: 137.204.78.85

Port: 5432

Database: bi

Authentication: Username and Password

Username: bi00

Password: Optional

☐ Require SSL

Tableau Prep - Basics

Input: drag & drop the table(s)

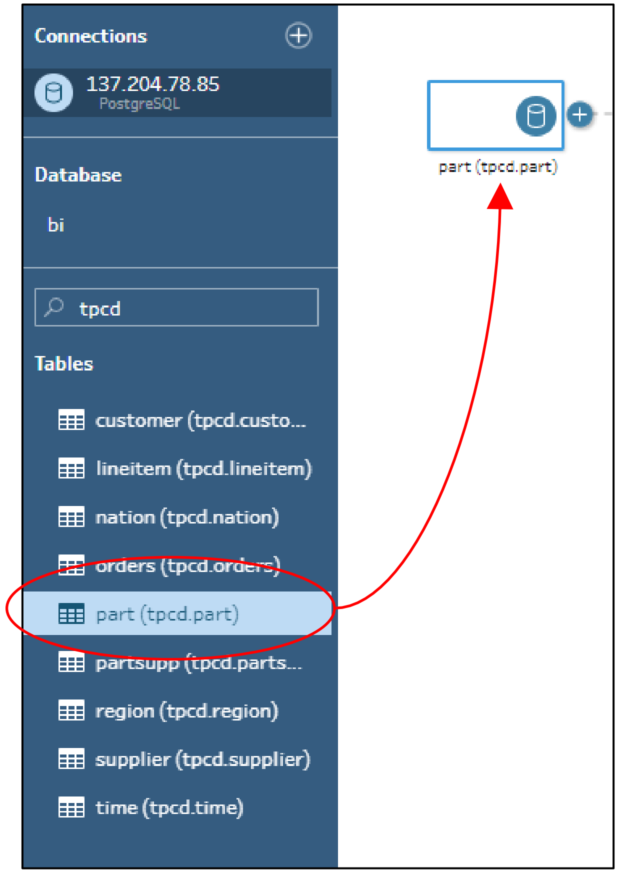


Tableau Prep - Basics

ETL: click on the (+) button to add an ETL step

- Clean: rename fields, create new ones
- Aggregate: carry out a group-by
- Pivot: invert rows with columns (or viceversa)
- Join, Union: intuitive
- Script: run a custom Python or R script
- Prediction: run ML algorithms (requires Salesforce subscription)
- Output: save results to file or table
- Insert flow: send results to a previously saved Tableau Prep flow

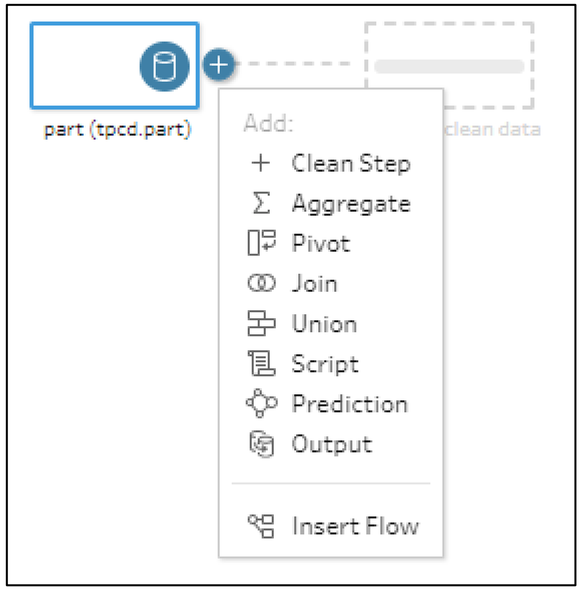
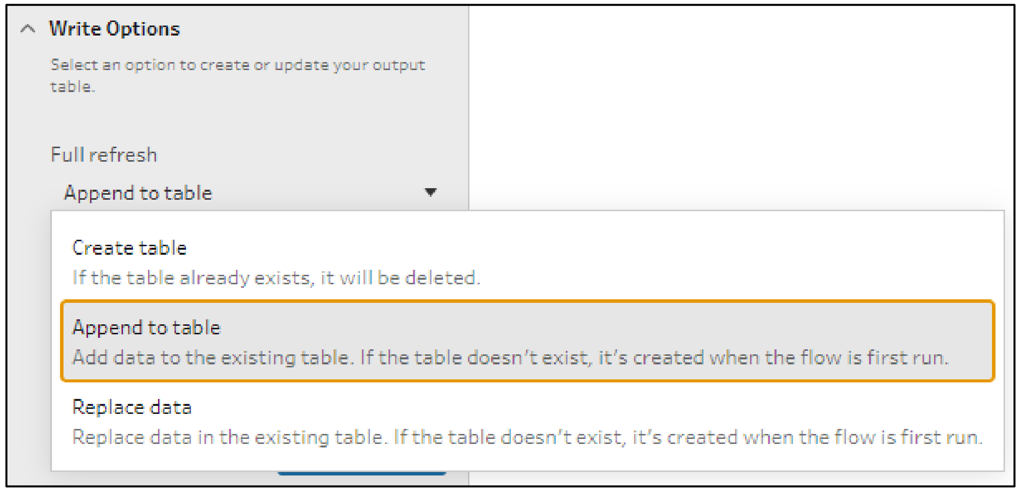


Tableau Prep - Basics

Output options

- Create table
    - Drops the table and re-creates it before adding the data
    - The table schema will be inferred from the data to be added
- Append to table
    - Leaves everything and performs "inserts" of the data
    - Conflicts are not handled
      (i.e., if a new row has the same ID of an old row in the output table, the flow will fail)
- Replace data
    - Truncates the table before adding the data
      (i.e., differently from the Create option, the schema is preserved)


Write Options

Select an option to create or update your output table.

Full refresh

Append to table

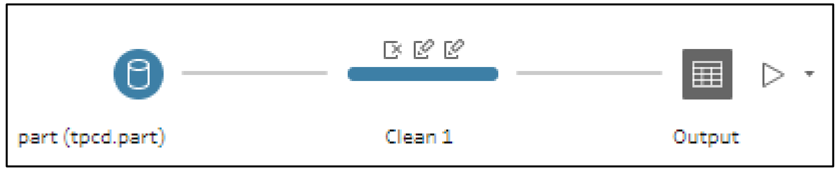
Create table
If the table already exists, it will be deleted.

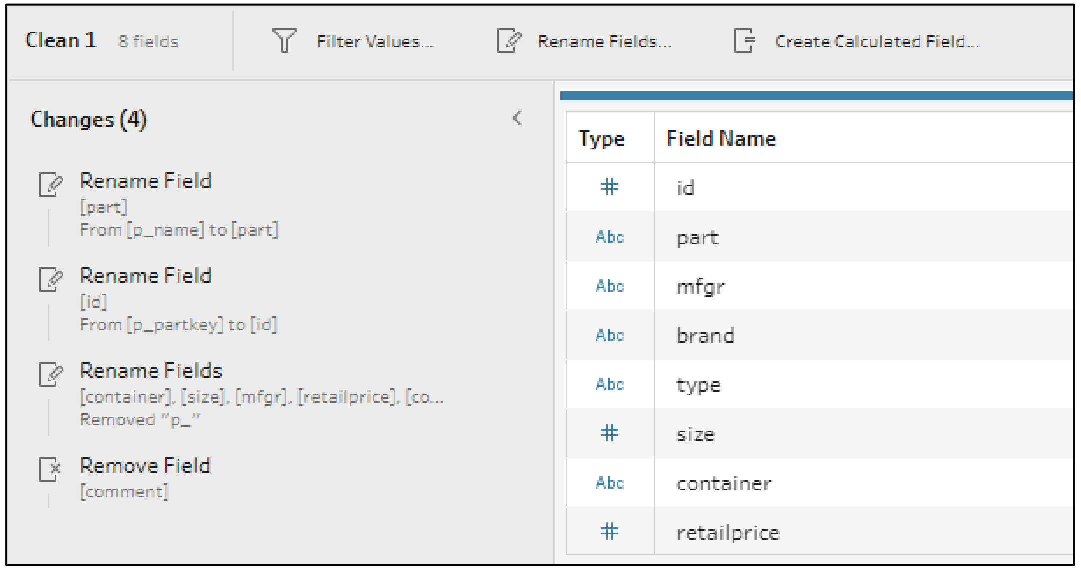
Append to table
Add data to the existing table. If the table doesn't exist, it's created when the flow is first run.

Replace data
Replace data in the existing table. If the table doesn't exist, it's created when the flow is first run.

The first flow

Feeding DT_PART

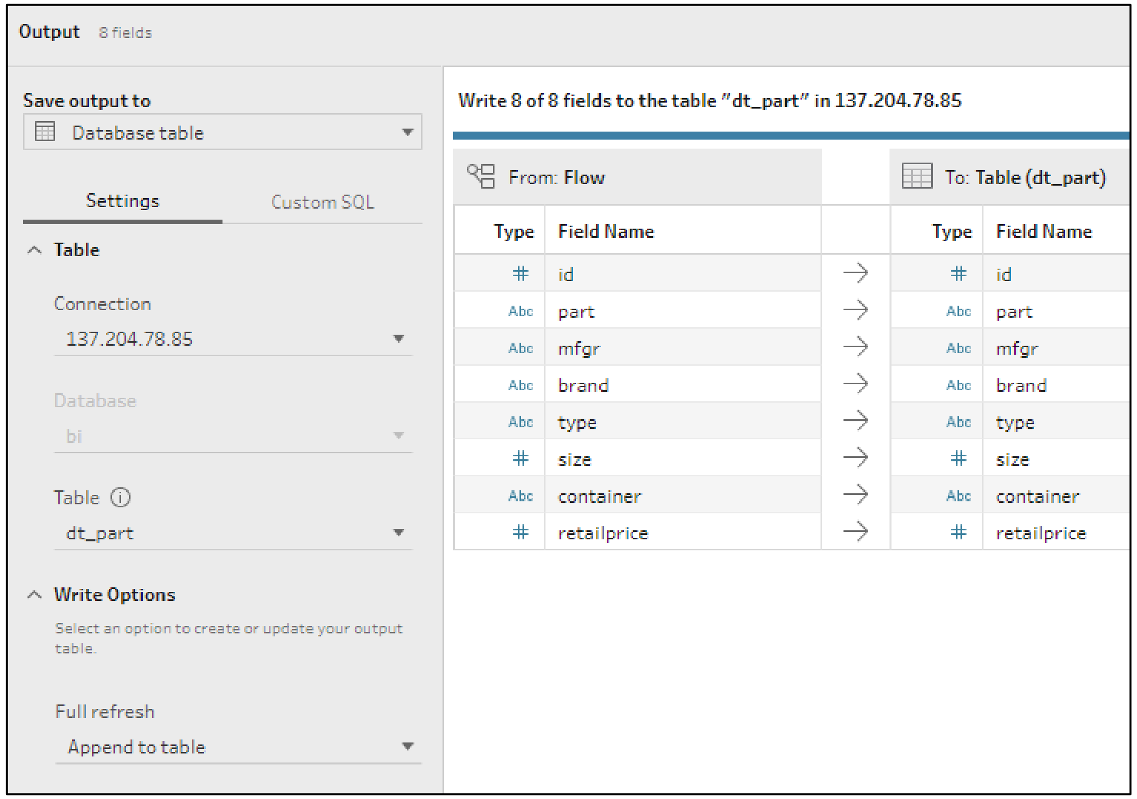



Clean 1 8 fields
Filter Values... Rename Fields... Create Calculated Field...

Changes (4)

- Rename Field
  [part]
  From [p_name] to [part]
- Rename Field
  [id]
  From [p_partkey] to [id]
- Rename Fields
  [container], [size], [mfgr], [retailprice], [co...
  Removed "p_"
- Remove Field
  [comment]

Type	Field Name
#	id
Abc	part
Abc	mfgr
Abc	brand
Abc	type
#	size
Abc	container
#	retailprice


Output 8 fields

Save output to: Database table

Settings | Custom SQL

Table
- Connection: 137.204.78.85
- Database: bi
- Table: dt_part

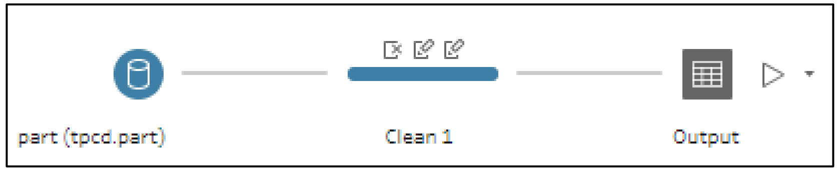
Write Options
Select an option to create or update your output table.
- Full refresh: Append to table

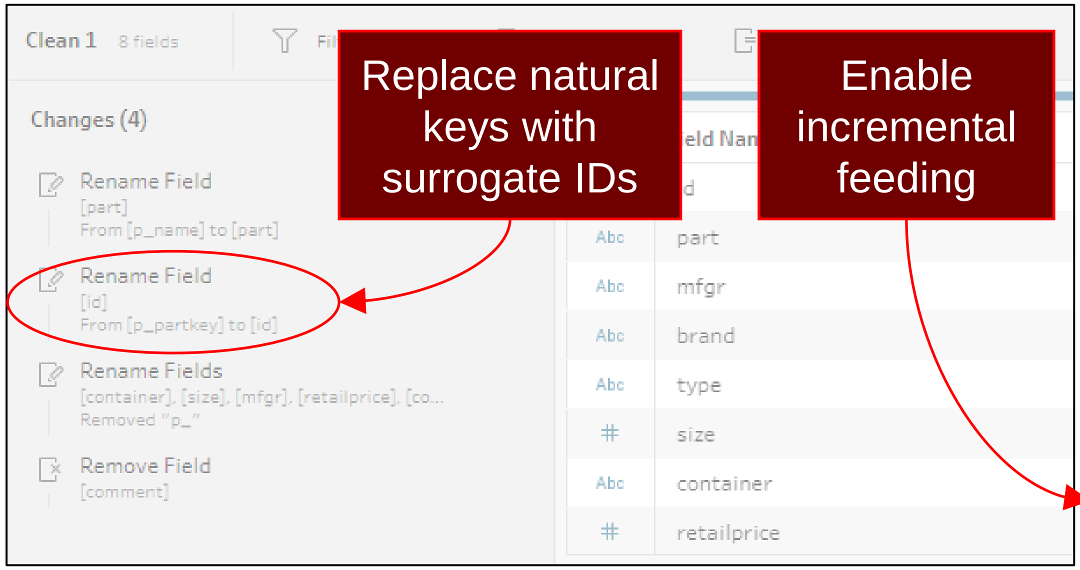
Write 8 of 8 fields to the table "dt_part" in 137.204.78.85

From: Flow			To: Table (dt_part)	
Type	Field Name		Type	Field Name
#	id	→	#	id
Abc	part	→	Abc	part
Abc	mfgr	→	Abc	mfgr
Abc	brand	→	Abc	brand
Abc	type	→	Abc	type
#	size	→	#	size
Abc	container	→	Abc	container
#	retailprice	→	#	retailprice

The first flow

Feeding DT_PART - what's missing?





Clean 1 8 fields

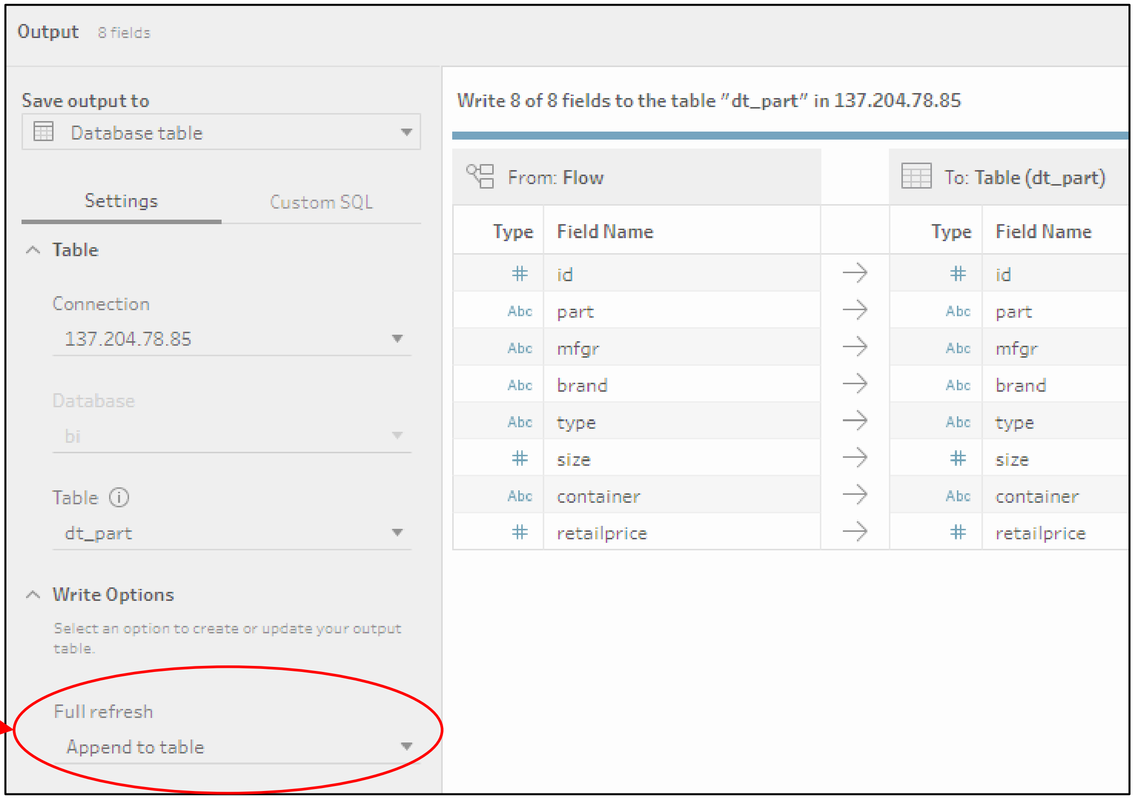
Changes (4)

- Rename Field [part] From [p_name] to [part]
- Rename Field [id] From [p_partkey] to [id]
- Rename Fields [container], [size], [mfgr], [retailprice], [co... Removed "p_"
- Remove Field [comment]

	eld Nam
	d
Abc	part
Abc	mfgr
Abc	brand
Abc	type
#	size
Abc	container
#	retailprice

Replace natural keys with surrogate IDs

Enable incremental feeding



Output 8 fields

Save output to: Database table

Settings | Custom SQL

Table

Connection: 137.204.78.85

Database: bi

Table: dt_part

Write Options

Select an option to create or update your output table.

Full refresh

Append to table

Write 8 of 8 fields to the table "dt_part" in 137.204.78.85

From: Flow Type	Field Name		To: Table (dt_part) Type	Field Name
#	id	→	#	id
Abc	part	→	Abc	part
Abc	mfgr	→	Abc	mfgr
Abc	brand	→	Abc	brand
Abc	type	→	Abc	type
#	size	→	#	size
Abc	container	→	Abc	container
#	retailprice	→	#	retailprice

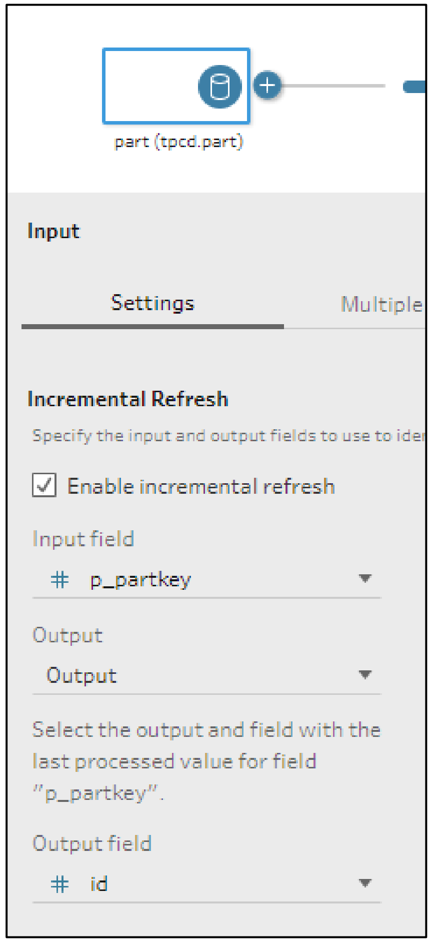
Incremental feeding

Tableau Prep supports some sort of incremental feeding..

- It compares IDs in the input table with IDs in the output
- And it keeps only new IDs

.. but it is not what we need

- What if some data has changed? (e.g., a product name)
- Sadly, the equivalent of a SQL update is not supported
- Thus, we need both new rows and updated rows



Incremental Refresh

Specify the input and output fields to use to ide

☑ Enable incremental refresh

Input field: # p_partkey

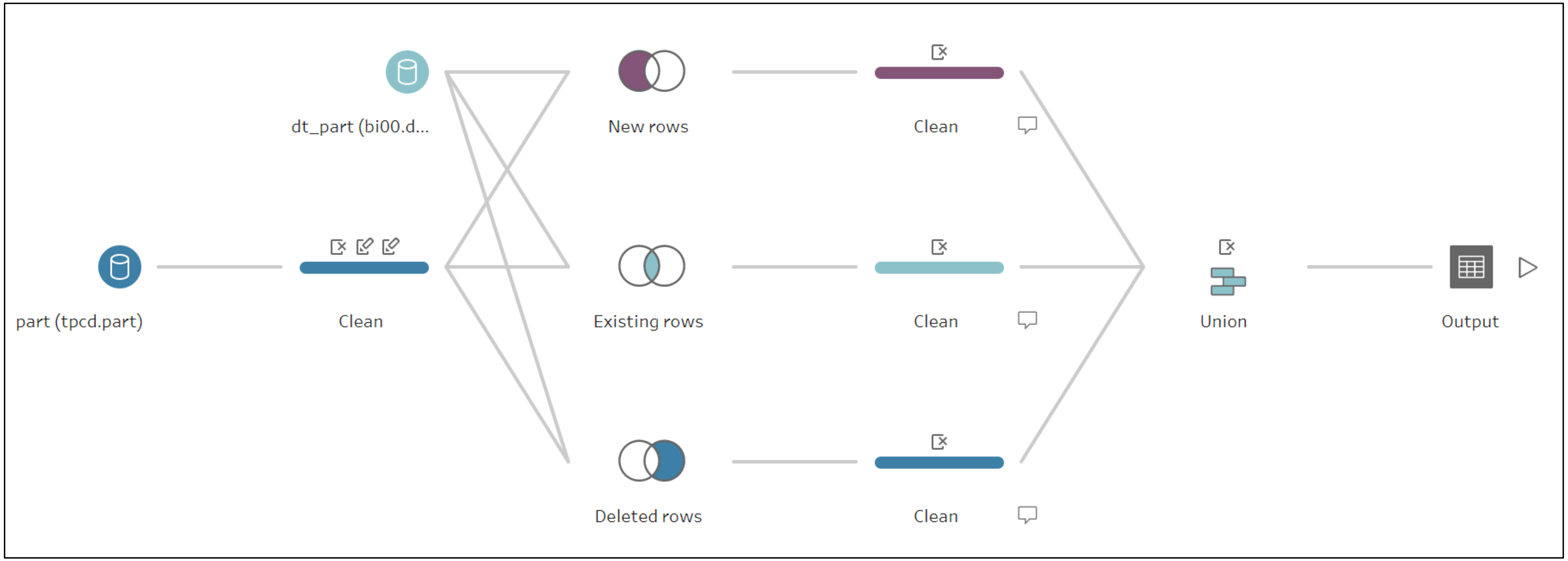
Output: Output

Select the output and field with the last processed value for field "p_partkey".

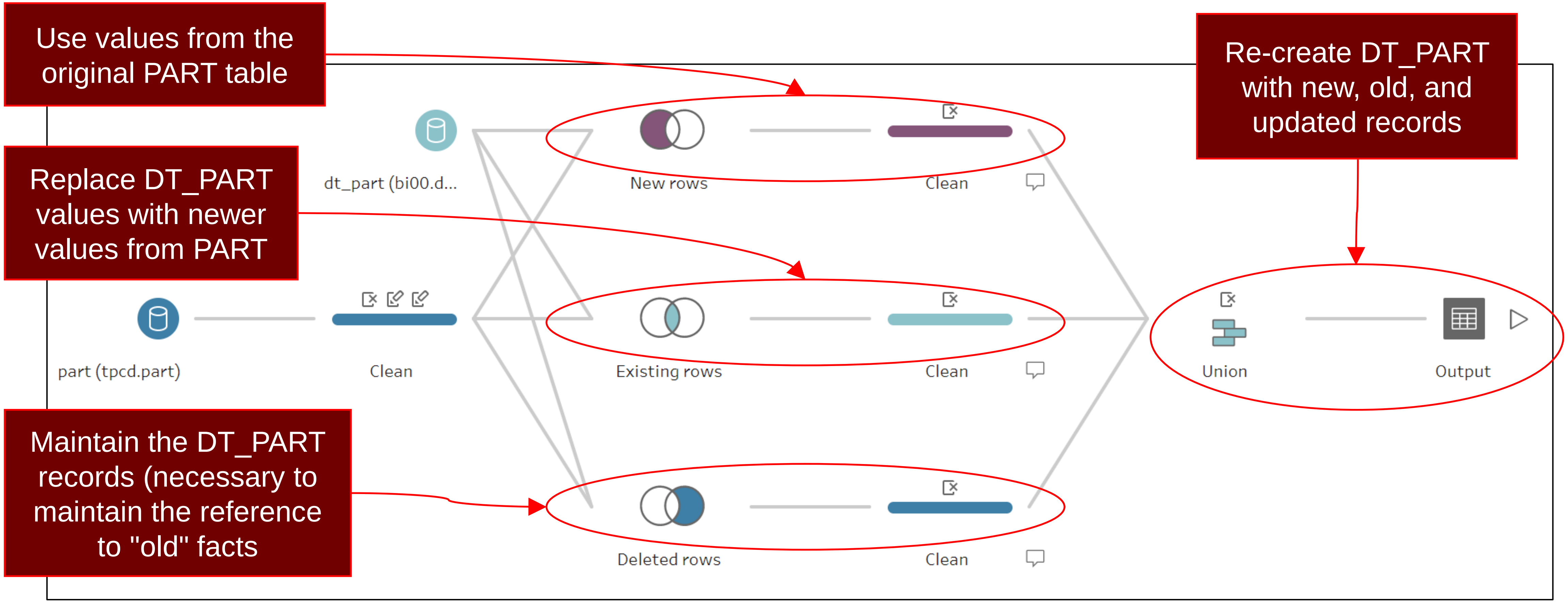
Output field: # id

Incremental feeding

This is what we need



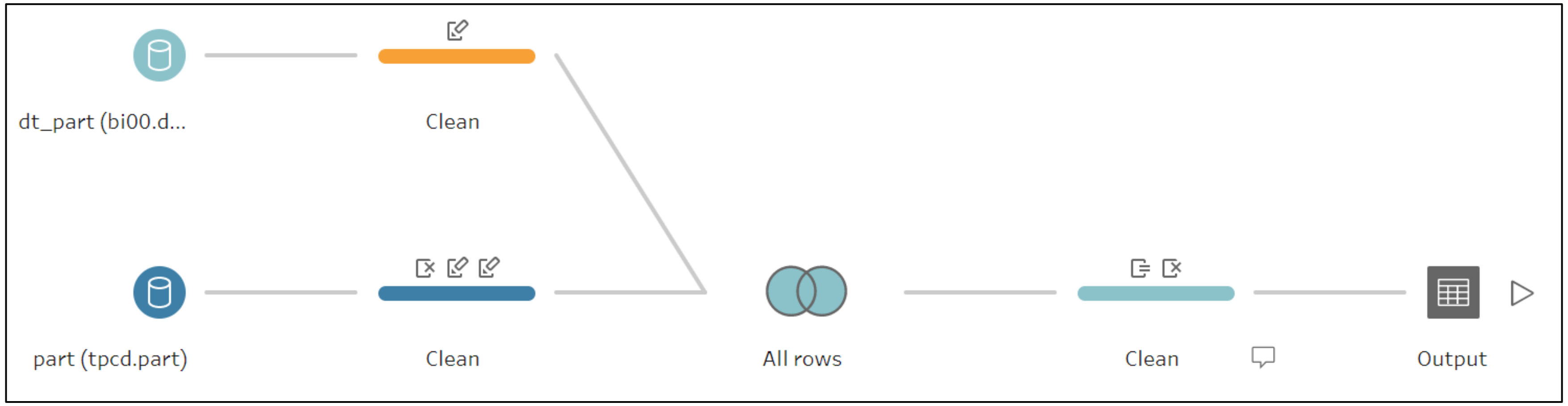
Incremental feeding


Use values from the original PART table
Replace DT_PART values with newer values from PART
Maintain the DT_PART records (necessary to maintain the reference to "old" facts
Re-create DT_PART with new, old, and updated records
dt_part (bi00.d...
part (tpcd.part)
Clean
New rows
Clean
Existing rows
Clean
Deleted rows
Clean
Union
Output

Incremental feeding

Can we make this simpler?

- Yes; we could do a single full-outer join and then use *calculated fields* to decide which values must be kept
  - `IF(ISNULL([size])) THEN [dt_size] ELSE [size] END`



Incremental feeding

Is the complete refresh of the DT the best way?

- No; it is a limit of Tableau Prep
- But DTs are usually limited in size

Would the complete refresh break foreign key constrainst on the fact table?

- Yes: old keys are preserved, but the refresh requires to delete and rewrite
- For simplicity, foreign keys are disabled

Could we implement temporal hierarchies?

- Yes; an extra effort would be required to create new records when an old one has been updated

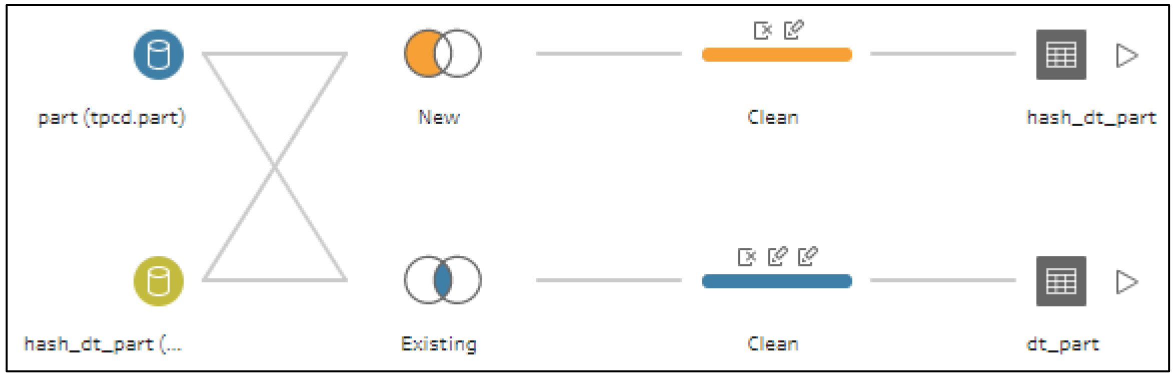
Surrogate keys

We need to match primary keys in the ODS with surrogate keys in the DW

- This requires lookup tables for every DT

-
  ```
  create table hash_dt_part (
   partkey integer primary key,
   id serial
  );
  ```

Surrogate keys

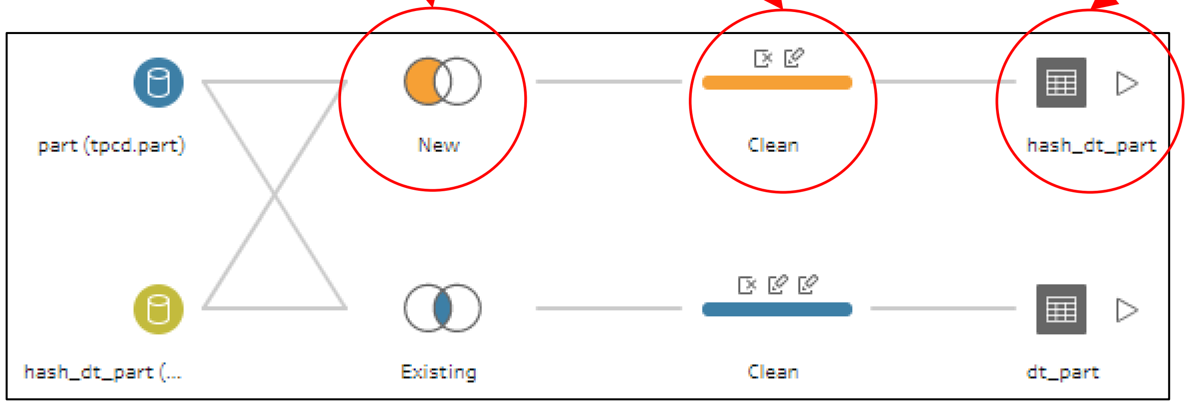


Surrogate keys

Take PART records that are not yet in the DT

Keep only the partkey

Append partkeys; the corresponding IDs are generated automatically

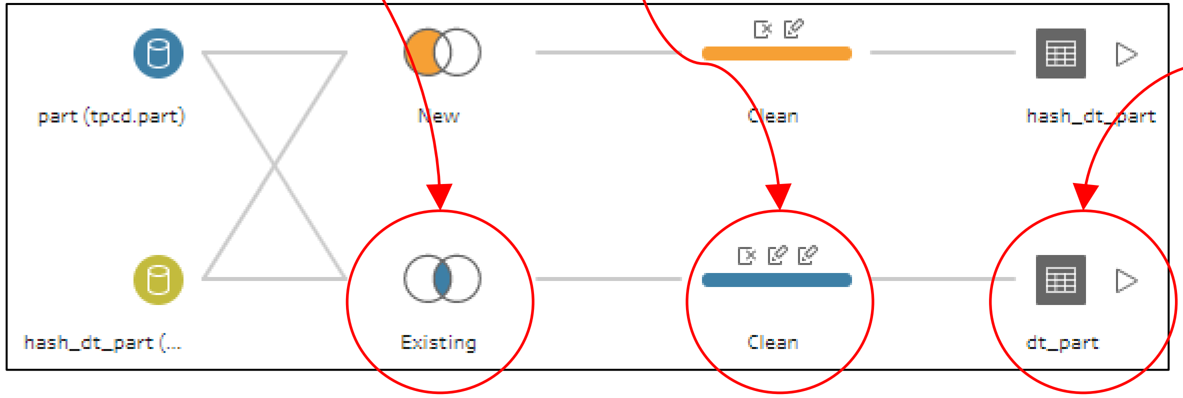


Surrogate keys

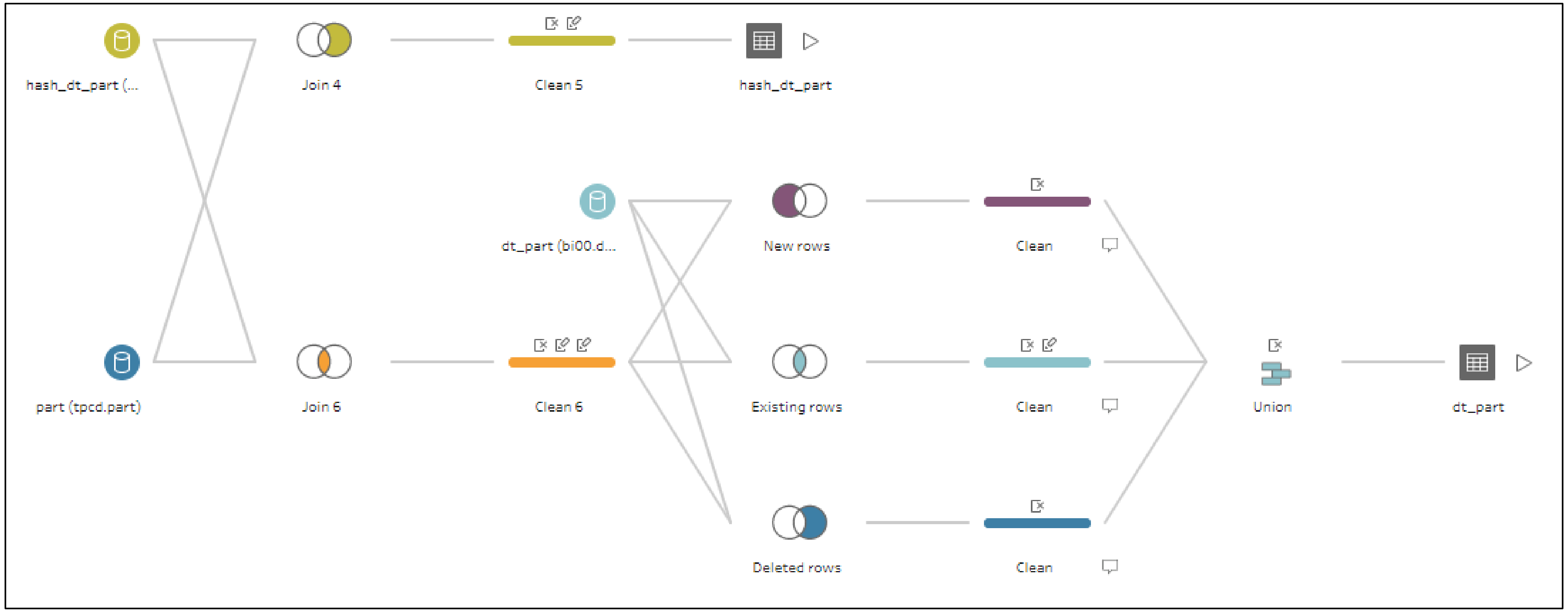
Join PART records with the updated lookup table

Replace partkeys with the corresponding IDs

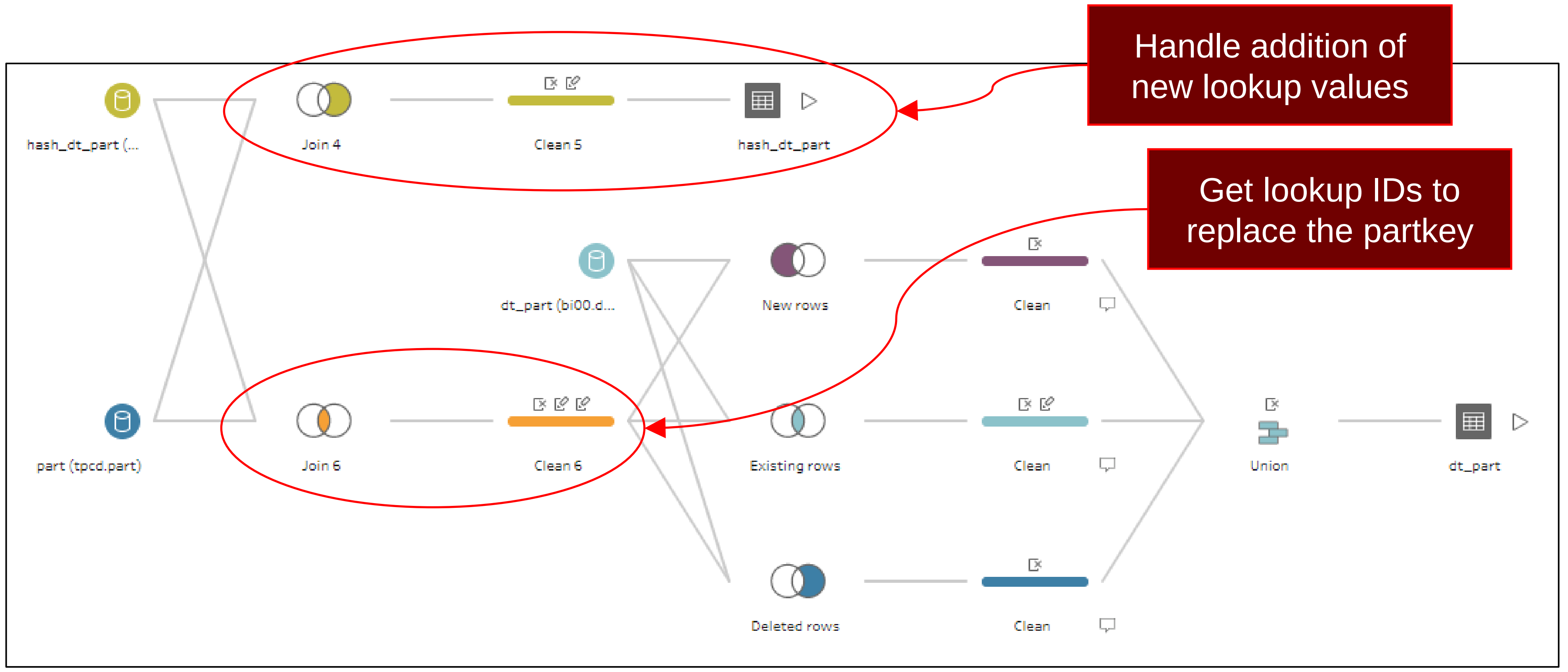
Replace the data in the DT



Incremental feeding with surrogate keys

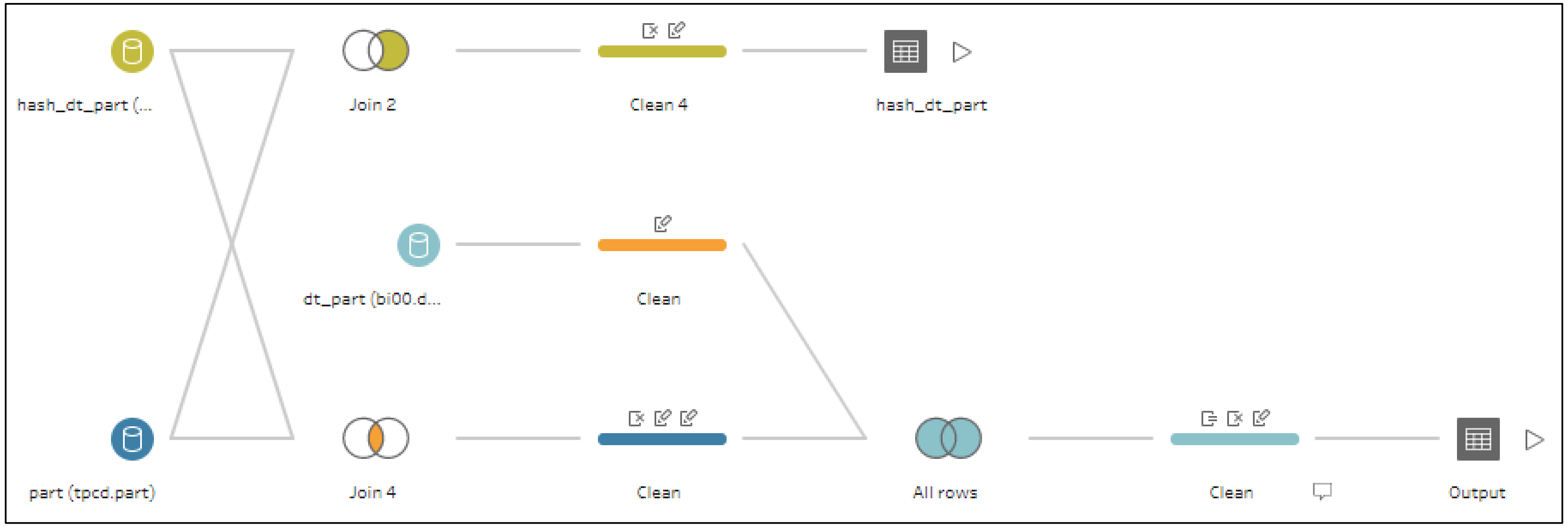


Incremental feeding with surrogate keys



Incremental feeding with surrogate keys

Simpler version

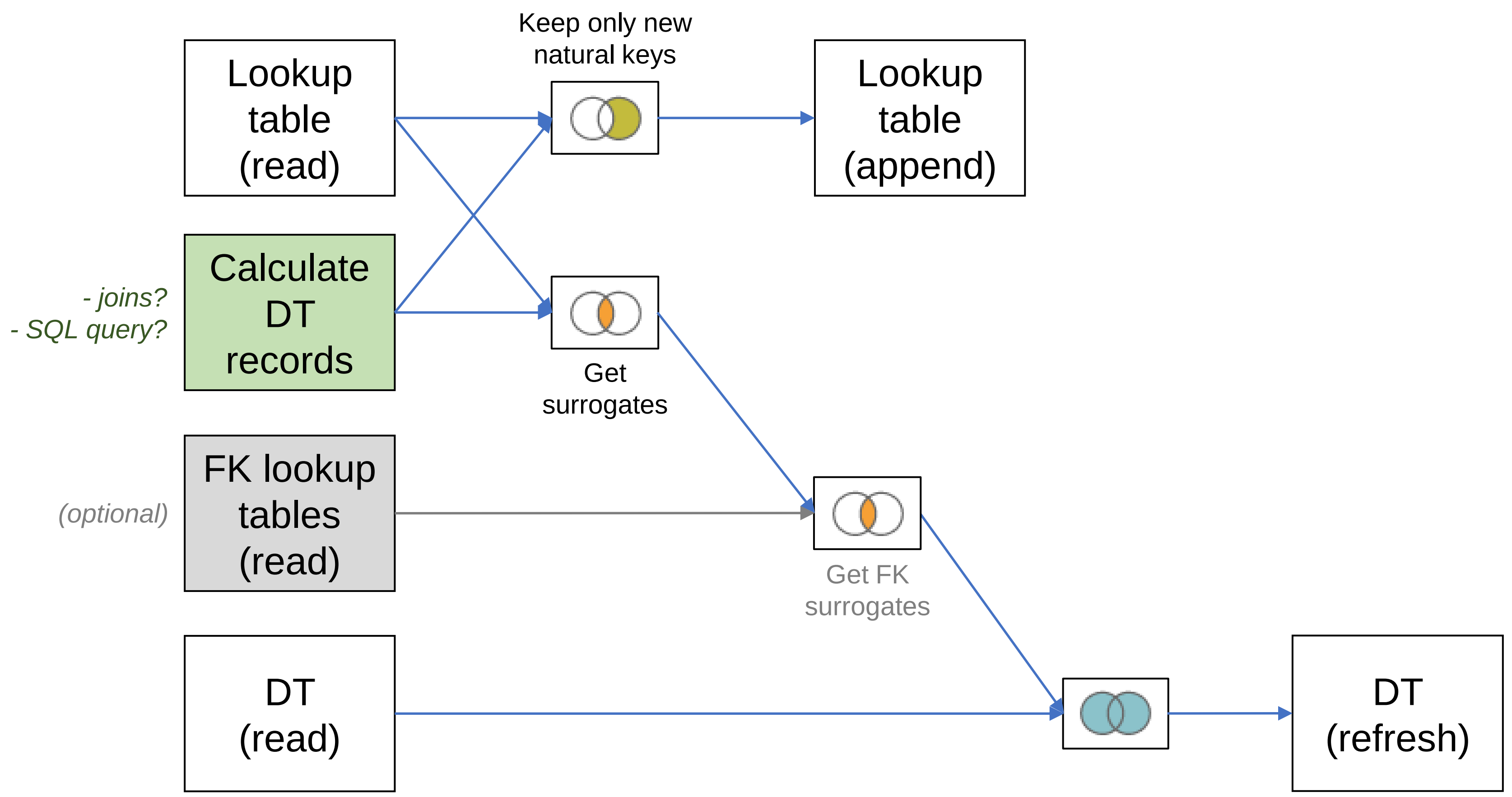


Complexities in feeding

What are other sources of complexity?

- More/Complex source tables
  - DT records hardly come from a single source table
  - Depending on the complexity of the sources and/or the computation that is needed, two options are viable
    - Setup joins in the ETL flow
    - Write a SQL query to obtain DT records
- Foreign keys
  - In case of snowflaking (but also to feed the FT), foreign surrogate keys must be setup
    - Setup joins with lookup tables in the ETL flow

Feeding reference schema



Exercise 1

Setup the same pipeline for the other dimension tables!

Exercise 1 guidelines

DT_SUPPLIER

- Feed the DT with the join of SUPPLIER, NATION, and REGION tables

DT_CUSTOMER

- Feed the DT with the join of CUSTOMER, NATION, and REGION tables

DT_DATE

- This a shared hierarchy
- Dates must be taken from all date fields in LINEITEM and ORDERS tables
    - Either take all distinct fields
    - Or generate all dates between the minimum and the maximum (requires script)

DT_ORDER

- Foreign keys must be collected for dates and customer IDs

DT_SHIPMENT

- Feed the DT with the distinct combination of l_shipmode and l_shipinstruct fields

Feeding the fact table

What's different?

- Foreign keys must be collected from DT lookup tables
- Most importantly, refreshing the whole FT could be expensive

How to handle incrementality?

- Assuming that event data (i) do not change, (ii) are always complete in the ODS
- Load only events from a certain time window
- Store in a lookup table the most recent date of loaded events
- Use the most recent date in the next feeding iteration
  - Using a Custom SQL component is advisable

Exercise 2

Setup the ETL flows for the ORDERS cube!

Exercise 2 guidelines

DT_PART, DT_SUPPLIER, DT_CUSTOMER, DT_DATE

- Same as for Exercise 1, no need to replicate them

JDT_ORDER

- Similar to JDT_SHIPMENT

FT_ORDER

- Feed with data from ORDERS

BRIDGE_PS

- Feed with data from LINEITEM
- Must be done *after* feeding FT_ORDER
- The weight must be computed!
    - *weight = li_extendedprice / o_totalprice*