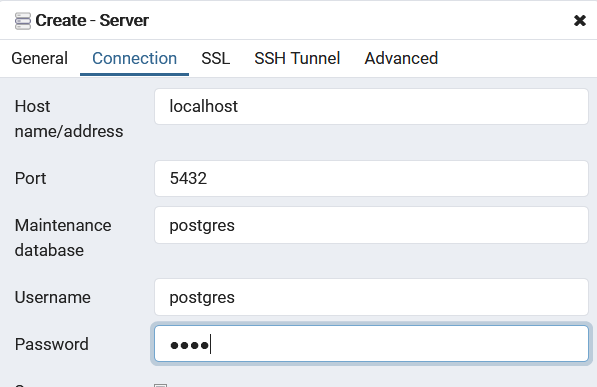
# **Урок № 2 – Загрузка данных в таблицы**

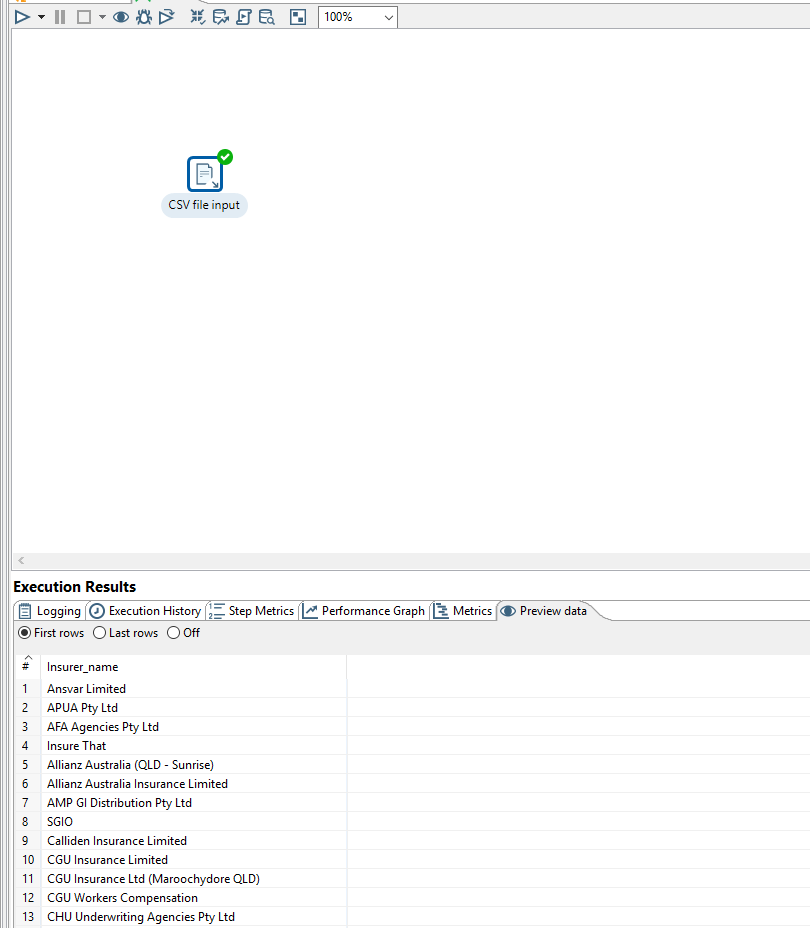
1. Скачать PostgreSQL 11 версии по ссылке: <https://www.postgresql.org/download/>
2. Установить PostgreSQL на локальный компьютер. Во время установки попросить создать пароль для суперпользователя postgres, очень важно запомнить этот пароль. Также, важно не снимать галочку с компонента pgAdmin на этапе, когда будет предложен выбор компонентов для установки.
3. Подключиться к только что созданному серверу используя пользователя «postgres» и пароль введённый при установке:



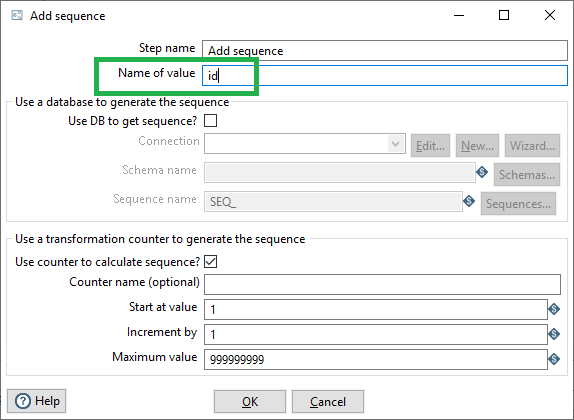
1. Создать новую базу данных назвав её «insurance». Отступление: обычно для каждой базы данных создаются пользователи с разными уровнями доступа к объектам базы данных, но в данном примере мы это опустим для упрощения и всегда будем использовать суперпользователя postgres.
2. Создать трансформации загружающие данные из файлов в таблицы в базе данных «insurance». Всего потребуется 4 трансформации: три трансформации будут наполнять справочники (insurers, producttype и states) и ещё одна трансформация будет наполнять таблицу фактов (fact.csv). В итоге из этих таблиц в базе данных должна получиться схема-звезда (подробнее прочитать про эту схему можно тут: <https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B7%D0%B2%D0%B5%D0%B7%D0%B4%D1%8B>).

Справочники будут заполняться по одному принципу: название компании (типа продукта, штата) будет браться из файла, каждому названию будет присваиваться уникальный номер и результат будет сохраняться в таблицу, содержащую два столбца: порядковый номер и наименование.

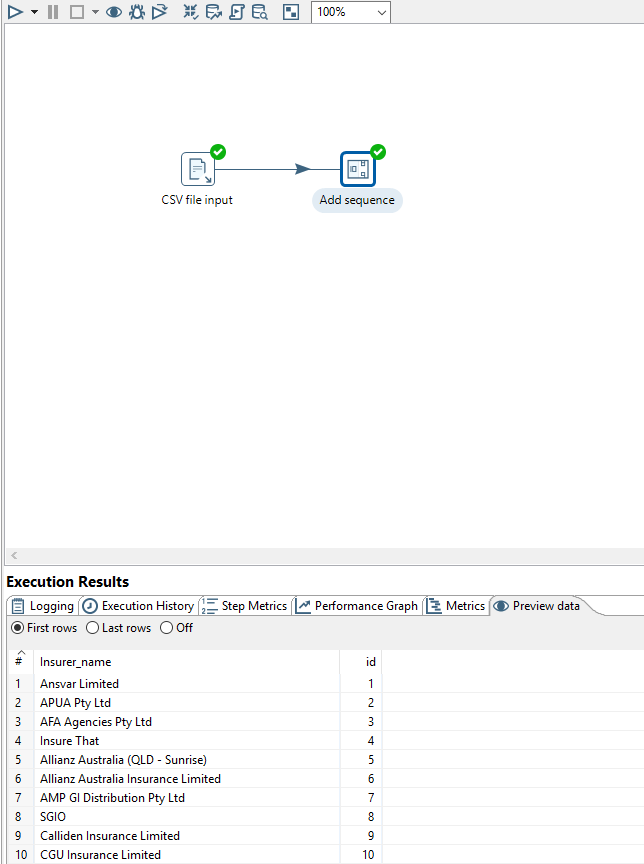
По аналогии с шагом 3 из урока 2 создадим новую трансформацию и добавим на неё компонент «CSV file input». Укажем что исходный файл будет «insurers.csv» сформированный во время выполнения второго урока. Наличия этого компонента уже достаточно чтобы запустить трансформацию и посмотреть результат на вкладке «Preview data»:



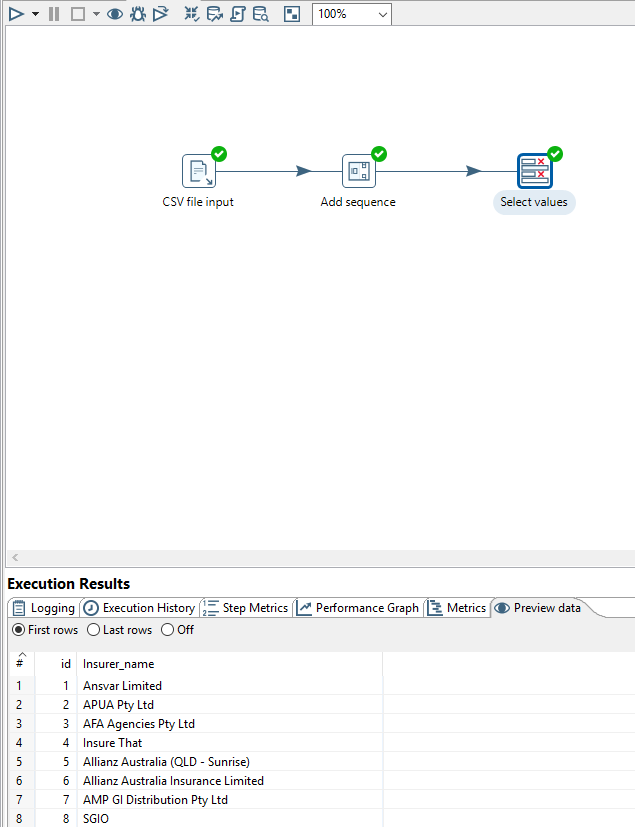
Пока у нас есть только один столбец «insurer\_name» содержащий название компаний. Нужно добавить ещё компонент «Add sequence» изменив параметр «Name of value» на «id»



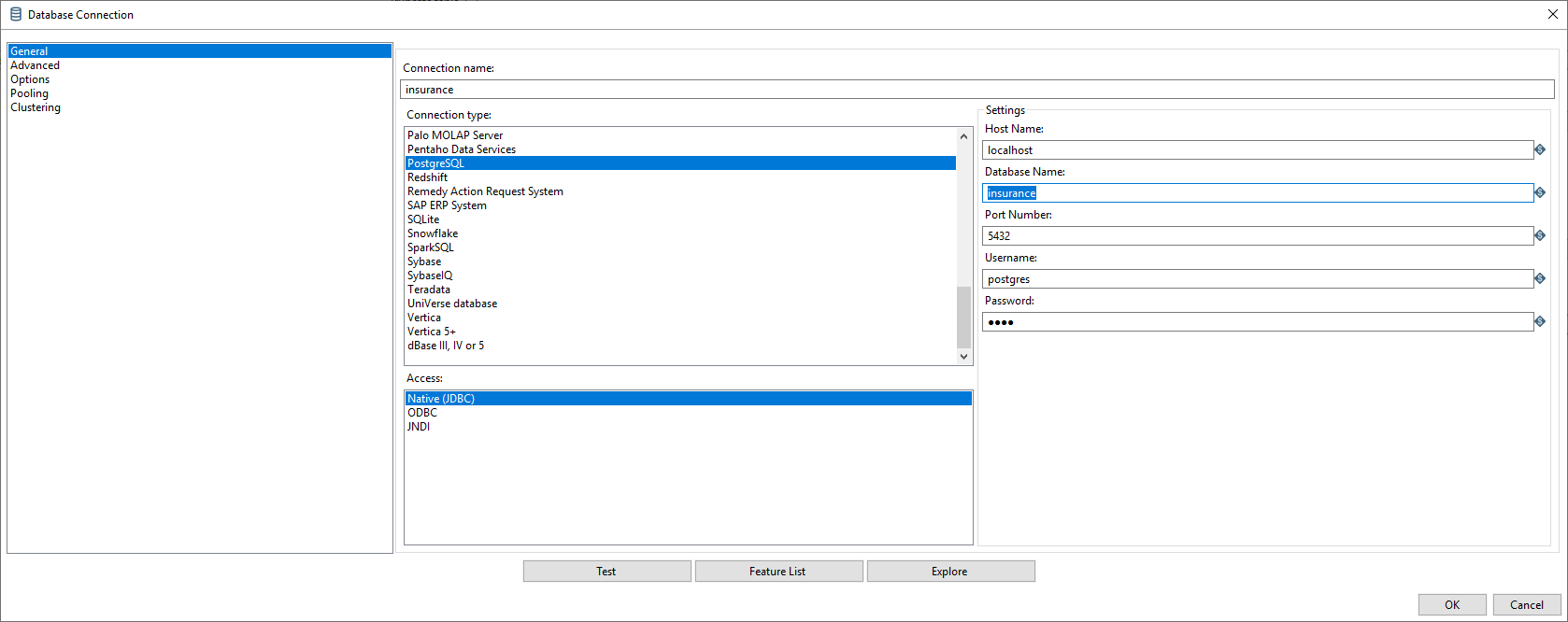
Выполнив трансформацию ещё раз, видим, что добавился ещё один столбец с названием «id» содержащий уникальный порядковый номер строки.



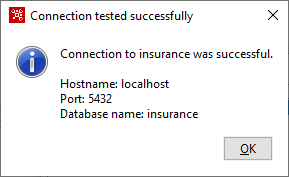
Чтобы смотрелось более читаемо, поменяем столбцы местами. Для этого добавим компонент «Select values» и в его свойствах на вкладке «Select & Alter» укажем что мы хотим видеть первым столбец «id» и вторым столбец «Insurer\_name». Запустим трансформацию и проверим что порядок столбцов тот что нам нужен:



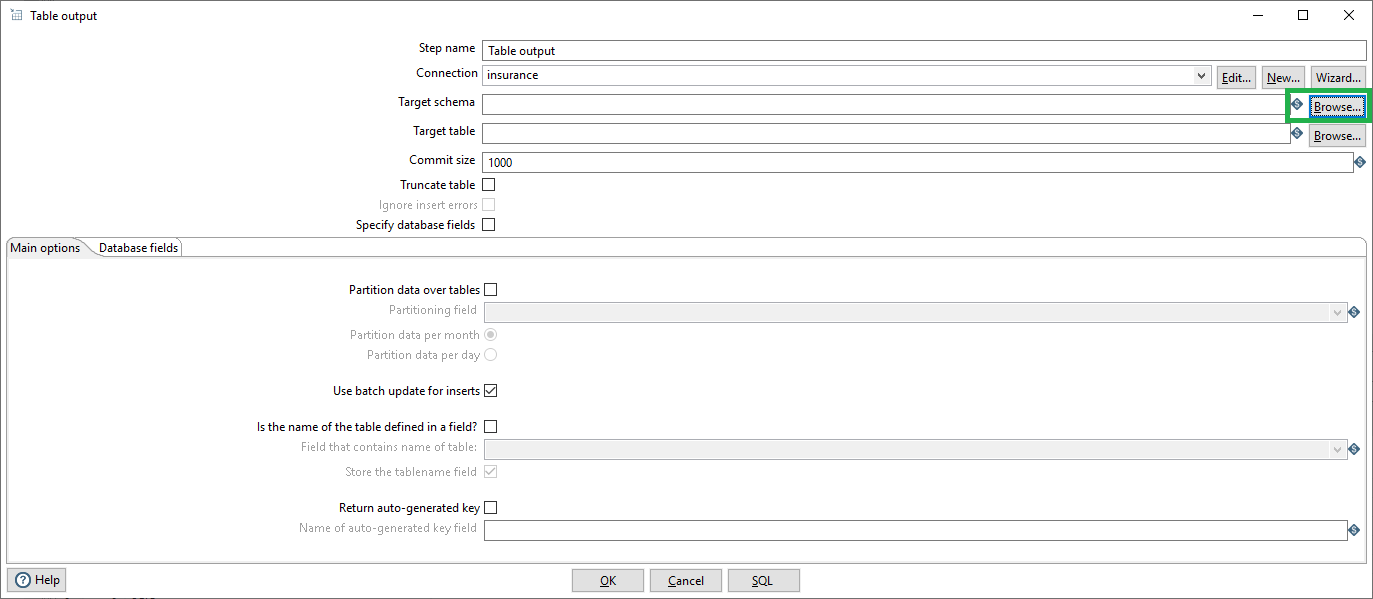
Всё готово чтобы записать эти данные в таблицу в PostgreSQL. Для этого добавим компонент «Table output» в трансформацию. Чтобы работать с таблицей в БД, надо создать подключение к таблице, для этого жмём «New» напротив «Connection» и указываем что имя подключения будет называться «insurance», Host Name – «localhost», Database Name – «insurance», порт 5432 по умолчанию (либо тот что был указан при установке PostgreSQL), Username – «postgres», password – тот что был указан при установке:



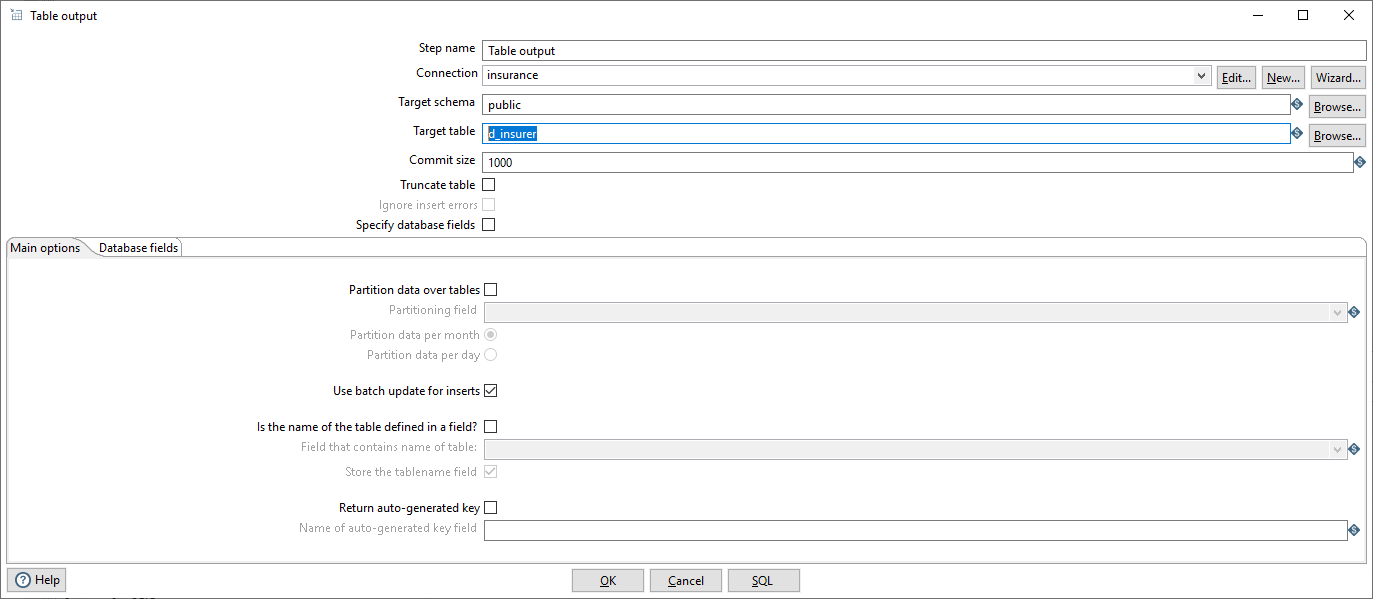
Можно нажать «Test» и проверить что все параметры соединения указаны верно и соединение работает:



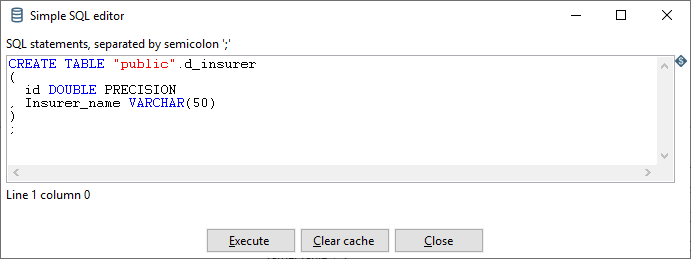
После того, как удалось установить соединение с базой, требуется выбрать схему в которой будут располагаться таблицы с данными. (Подробнее про схемы в базах данных можно прочитать тут: <https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85#%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%BA%D0%B0%D0%BA_%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85>) Коротко, схема (schema) – это каталог в котором хранятся таблицы. Обычно их много и служат они для разделения таблиц по смыслу или по каким-то свойствам. В текущем примере мы воспользуемся схемой по умолчанию – «public». Нажимаем на «Browse» и выбираем её:



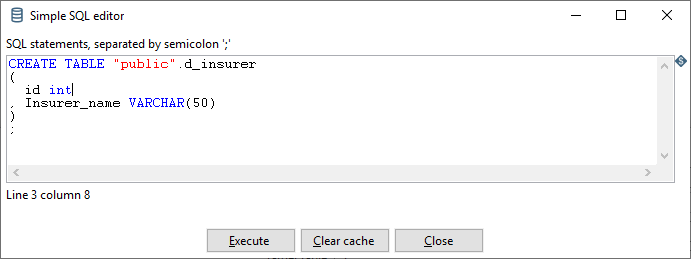
Если нажать «Browse» напротив «Target table», то мы ничего не сможем выбрать, потому что наша база данных сейчас пустая – в схеме «public» нет ни одной таблицы. Соответственно, эту таблицу нам надо сейчас создать. Сделаем это с помощью Pentaho: укажем что имя нашей целевой таблицы будет d\_insurer (префикс «d\_» это чтобы всем пользователям было понятно, что данная таблица является справочником; «d» – «коротко от dimension»):



Далее нажмём SQL, Pentaho генерирует SQL скрипт создания таблицы с нудным нам именем и столбцами:



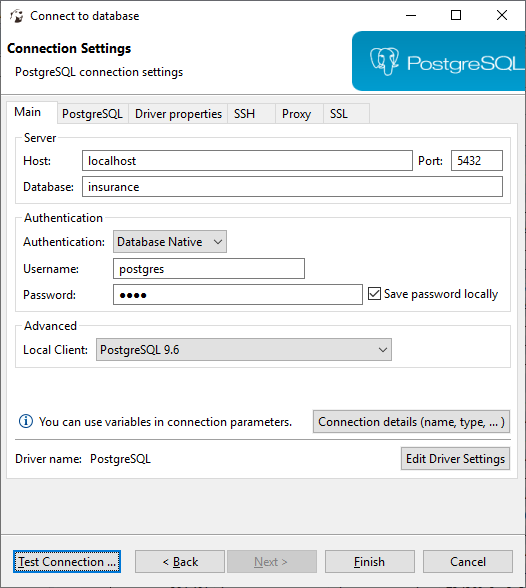
Единственное, тип «Double precision» не очень подходит для хранения порядковых номеров, лучше руками исправить на «int» и выполнить скрипт нажав «Execute»:



Далее, нажать close, ok и сохранить трансформацию, но пока не запускать её.

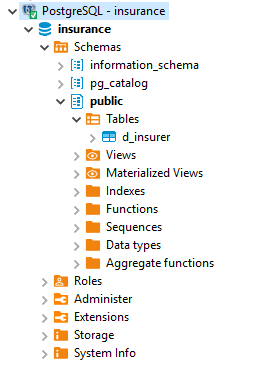
Требуется проверить действительно ли мы создали её в базе данных. Если по какой-то причине результат создания таблицы вам не нравится, можно выполнить скрипт DROP TABLE ИМЯ\_ТАБЛИЦЫ и таблица будет удалена. Далее её можно пересоздать с корректными параметрами.

Скачаем бесплатный клиент баз данных «dbeaver» по ссылке <https://dbeaver.io/download/> . Если ссылка перестала работать, найти в интернете актуальную ссылку по ключевым словам: «dbeaver Community Edition». Установить dbeaver, запустить его и нажать «New database connection», выбрать PostgreSQL и ввести те же параметры, которые вводились во время настройки подключения в базе PostgreSQL в Pentaho:

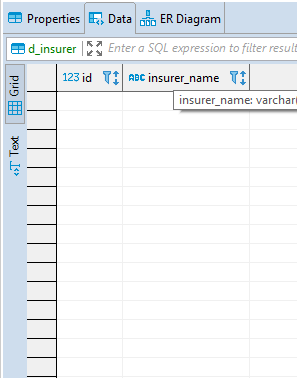


Тут так же присутствует кнопка «Test Connection» для проверки корректности параметров. Если соединение удалось установить, жмём «Finish».

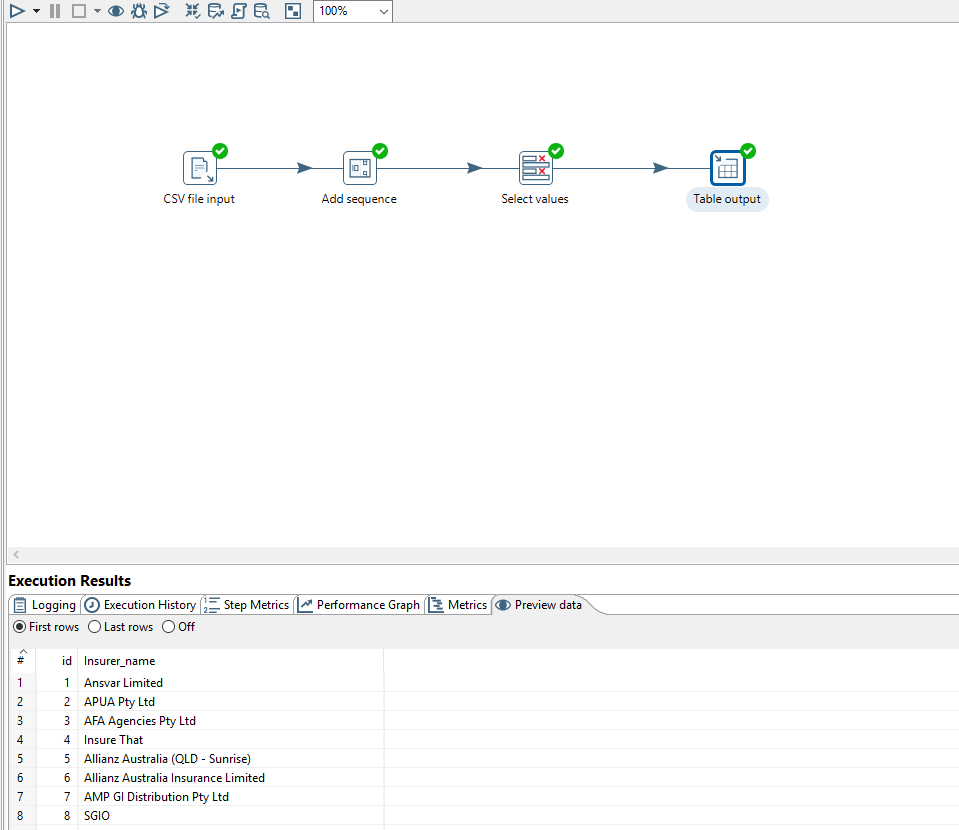
Если всё сделано корректно, в дереве появится наша база и созданной таблицей d\_insurer в схеме public:



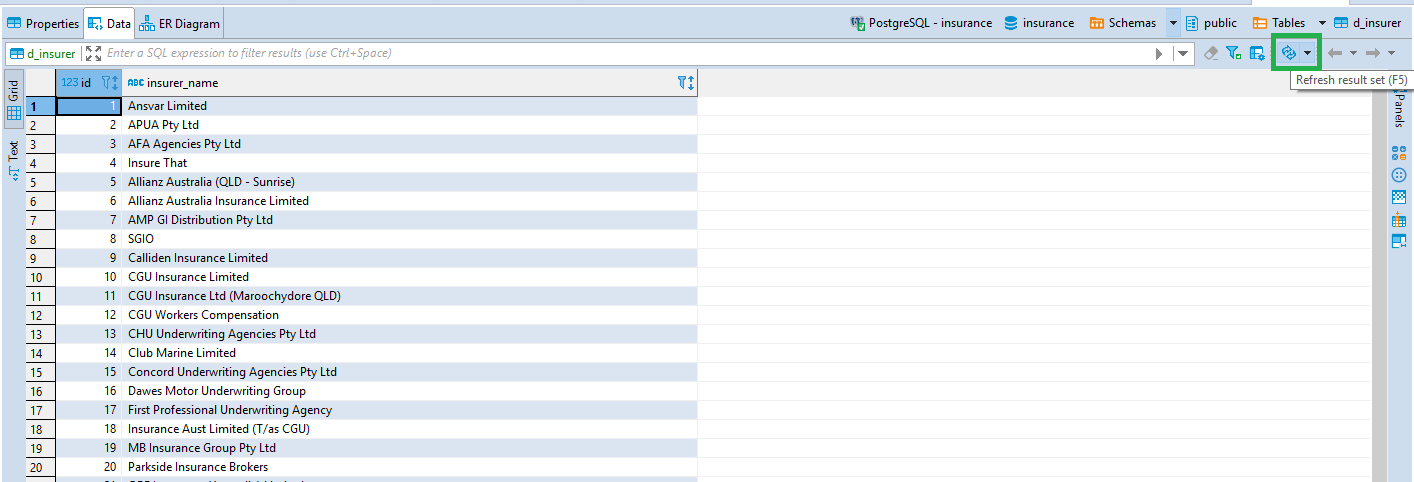
Прямо сейчас можно открыть таблицу двойным щелчком и убедиться что в ней два столбца, но пока нет данных:



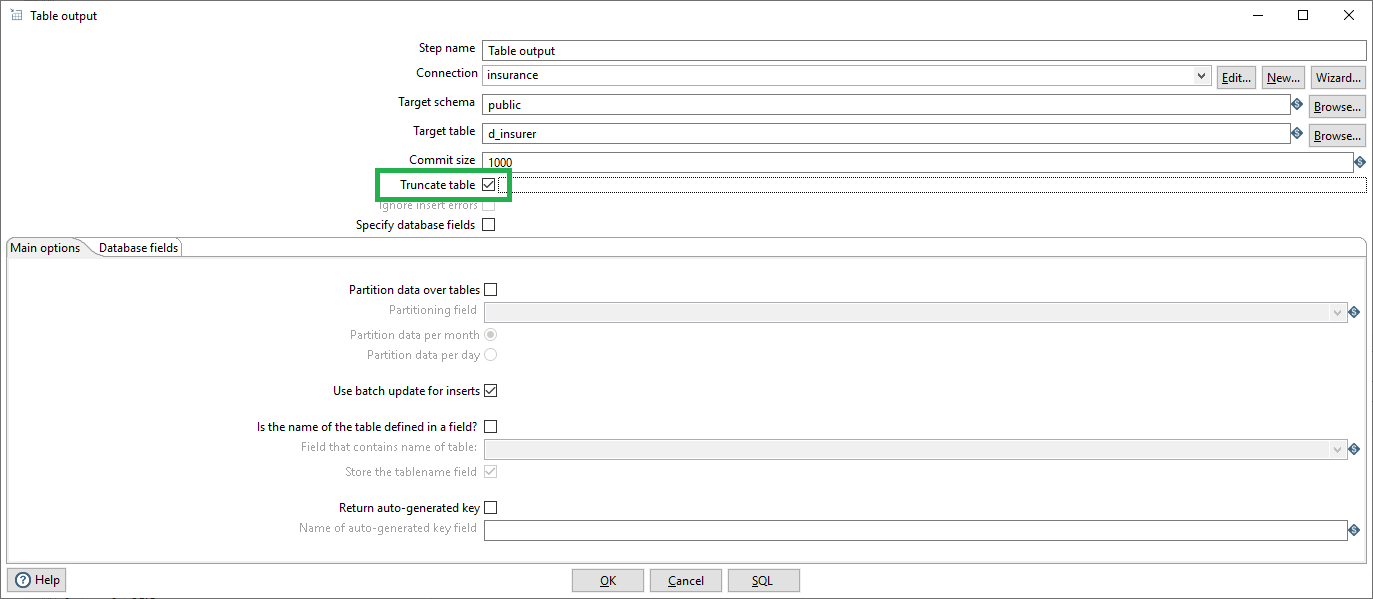
Теперь самое время запустить трансформацию в Pentaho. Трансформация отработала и на preview видно, что данные успешно ушли в таблицу:



Нажав «Refresh» в dbeaver, видимо что данные в таблице действительно появились.



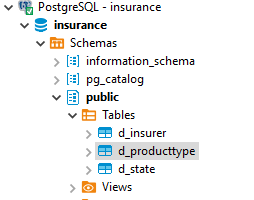
Единственное что осталось сделать – поставить галку «Truncate table» в свойствах таблицы в Pentaho. Это позволит каждый раз очищать таблицу перед вставкой в неё данных.



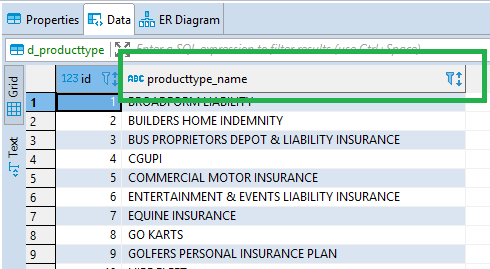
Рекомендация: запустить трансформацию несколько раз с галкой и без и понаблюдать за результатом через dbeaver.

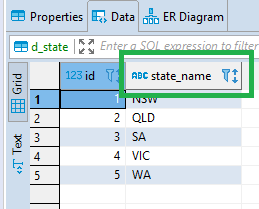
Далее, повторить все шаги для фалов producttype и states.

В результате выполнения должны быть созданы три таблицы наполненные данными:



Особое внимание стоит уделить названием столбцов:





Чтобы названия полей были корректными, надо в шаге «CSV file input» при изменении исходного файла нажимать «Get fields» чтобы перечитать содержимое файла.

1. Загрузить таблицу фактов. Из описания <https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B7%D0%B2%D0%B5%D0%B7%D0%B4%D1%8B> видно что таблица фактов содержит несколько целочисленных колонок-ключей для доступа к таблицам измерений. В исходном файле (fact.csv) у нас нет колонок-ключей. Вместо них присутствуют текстовые названия, например, вторая строка файла содержит (в первой находится заголовок):

|  |  |  |  |
| --- | --- | --- | --- |
| insurer\_name | state\_name | type\_name | revenue |
| Insurance Aust Limited (T/as CGU) | SA | PERSONAL ACCIDENT TOP UP | 267 |

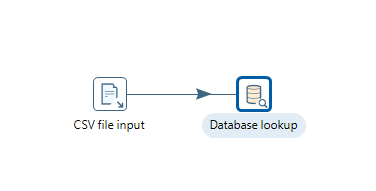
То есть компания Insurance Aust Limited (T/as CGU) из штата SA продала страховку PERSONAL ACCIDENT TOP UP за 267 долларов. Для человека этот вид хранения приемлем и хорошо читаем, но в базах данных обычно вместо текстового обозначения хранят порядковый номер строки в конкретном справочнике. При данном подходе экономится место, потому что целое число занимает меньше байт чем текстовое поле, и поиск нужной информации выполняется быстрее, потому что операция сравнения чисел работает быстрее чем сравнение одного текстового значения с другим. Поэтому эта же строка в таблице фактов в базе будет выглядеть следующим образом:

|  |  |  |  |
| --- | --- | --- | --- |
| insurer\_name | state\_name | type\_name | revenue |
| 18 | 3 | 23 | 267 |

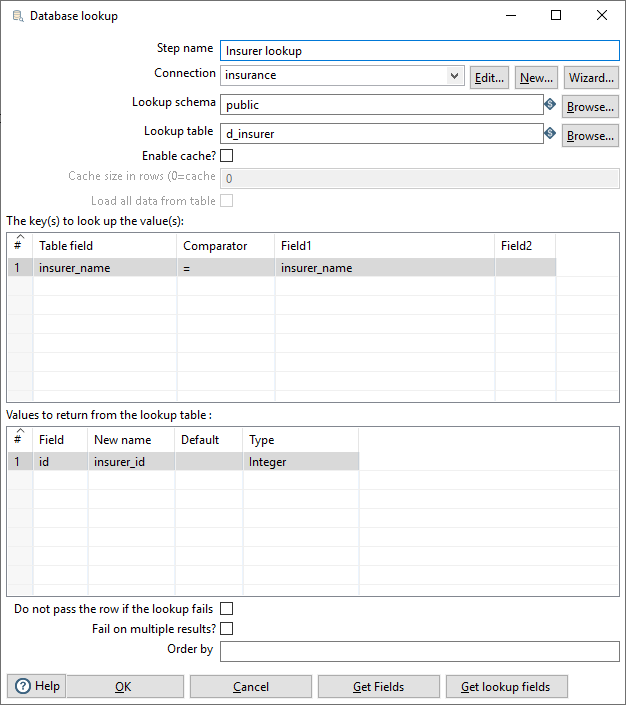
Где 18 – порядковый номер у строки, содержащей «Insurance Aust Limited (T/as CGU)» в таблице public.d\_insurer, 3 – порядковый номер для «SA» в таблице штатов и 23 - по аналогии.

Для того чтобы подменить текстовое значение на порядковый номер в PDI используется компонент «Database lookup». Он позволяет взять текстовое поле, найти его порядковый номер в таблице-справочнике и использовать его для передачи в таблицу фактов.

Создадим новую трансформацию, добавим «CSV file input» берущий данные из «fact.csv». Следующим шагом ставим «Database lookup» и соединяем их:

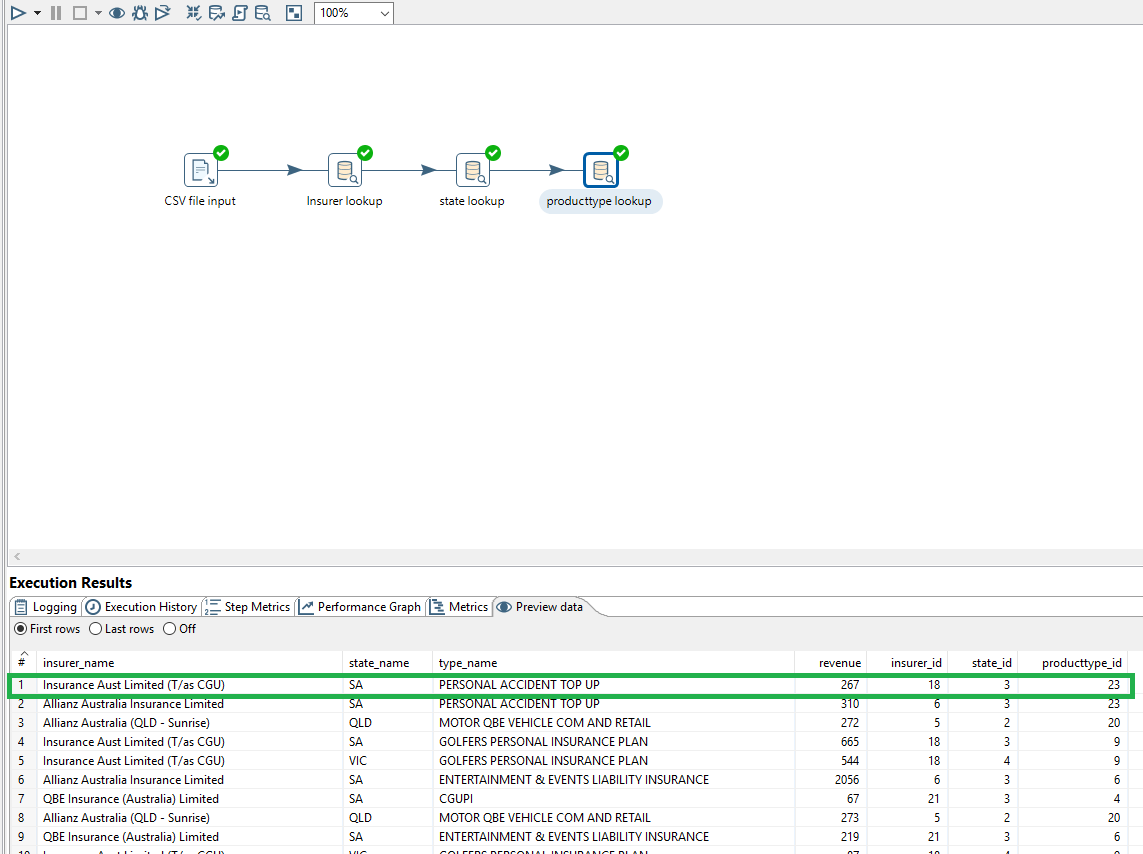


Открываем свойства «Database lookup», меняем Step name на «Insurer lookup», указываем что мы будем работать с таблицей public.d\_insurer и будем брать поле id из этой таблицы при условии что значение поля insurer\_name из таблицы равно полю insurer\_name из файла «fact.csv»



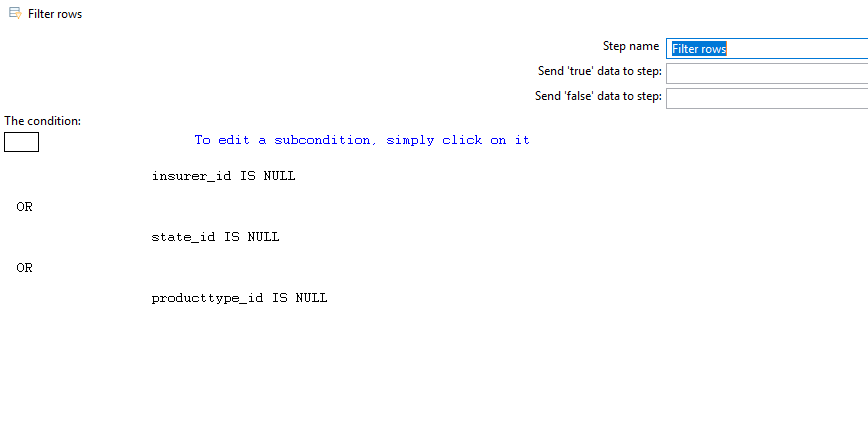
Сохраняем трансформацию, запускаем её и видим, что из файла приходит 4 столбца, а на шаге «Insurer lookup» добавляется ещё один столбец «insurer\_id» содержащий порядковый номер для текстового значения «insurer\_name» из таблицы public.d\_insurer.

По аналогии делаем для двух других таблиц (d\_state и d\_producttype), запускаем и видим что все текстовые значения успешно подменились на порядковые номера:

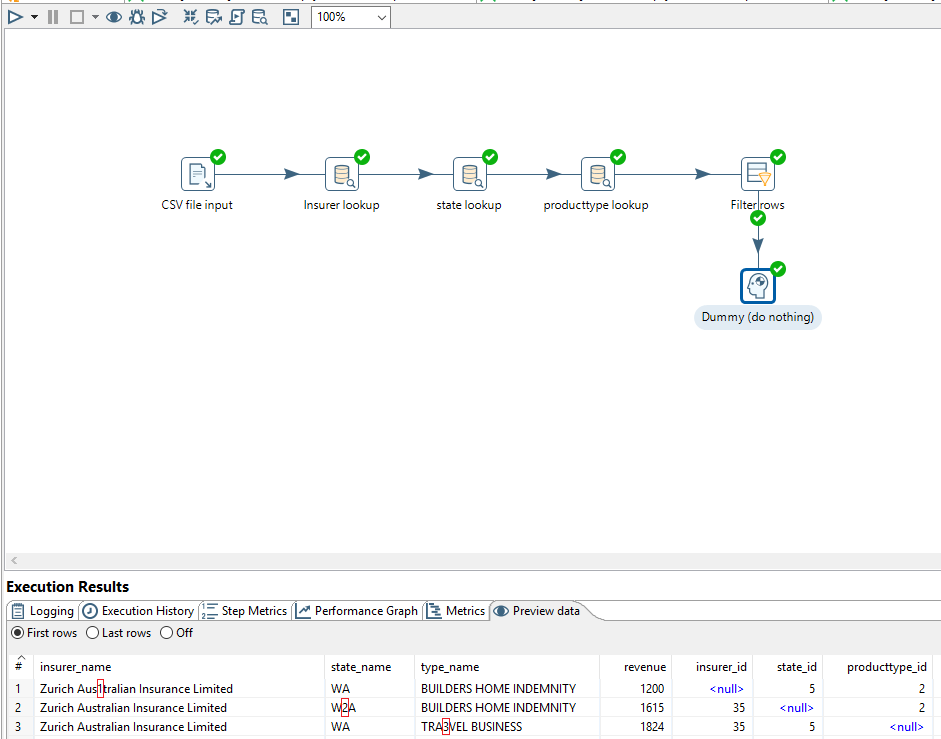


Стоит отметить что id значения совпадают с примером данным выше.

Если внимательно посмотреть на результаты, то можно увидеть, что не для всех текстовых значений подставился порядковый номер. Это значит что либо файл «fact.csv» содержит значения, которые отсутствуют в справочниках, либо про заполнении справочников произошли ошибки. Попытаемся отфильтровать эти ошибки. Добавим компонент «Filter rows» для отбора строк у которых один из столбцов «…\_id» пустой:

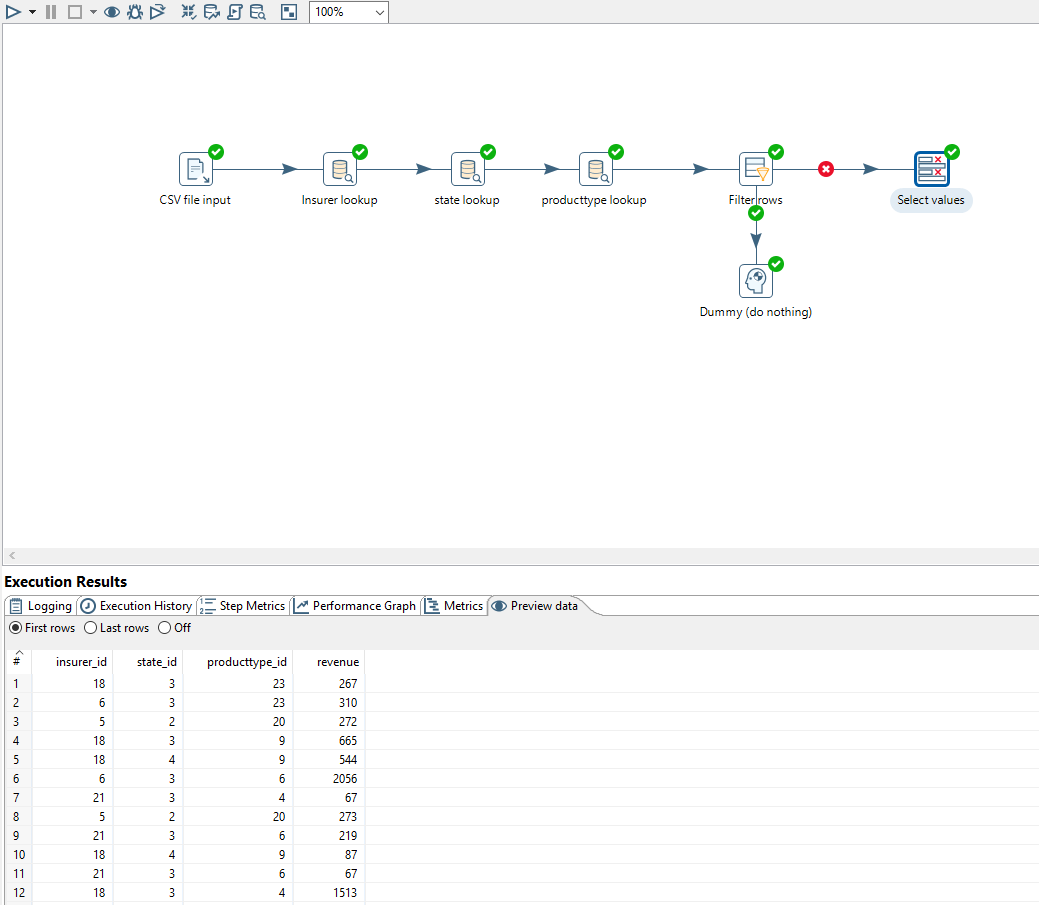


После фильтра поставим компонент «Dummy (do nothing)». Этот компонент не делает ничего с данными – он просто позволяет их просматривать. Компонент «Filter rows» позволяет разбить данные на две части: одна часть данных будет удовлетворять условию фильтра – вторая нет.

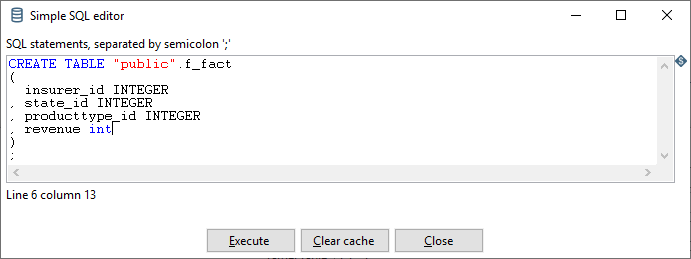


Теперь видно почему для данных значений не нашлись порядковые номера – в данных в файле содержатся опечатки. В реально жизни надо было бы сохранить эти записи отдельно и отправить их заказчику с вопросом что с ними делать. В данном примере мы их просто отбросим.

Теперь можно отсечь ненужные текстовые поля у тех записей для которых нашлись все «…\_id» поля с помощью компонента «Select values»:



Осталось эти значения запихнуть в таблицу f\_fact по аналогии с предыдущими шагами. Для всех полей необходимо установить тип «int»



Проверяем данные в таблице через dbeaver:

