**Дегтярев Денис**

**Документация итоговой работы по модулю**

**“DWH”**

В рамках итоговой работы будет необходимо:

1. ***Запустить чистую базу данных в докере и создать там таблицы с фактами и размерностями***

Чистая БД была запущена при помощи следующего скрипта в командной сроке:

docker run --name netology\_DWH -e POSTGRES\_PASSWORD=DWH123 -e POSTGRES\_USER=DWH -e POSTGRES\_DB=DWH-db-final -d postgres

1. ***Наполнить базу данными из бд bookings при помощи ETL***

***А) Сперва в новой БД были созданы соответствующие схемы и таблицы***

1. -- создаем схемы
2. **CREATE** **SCHEMA** dim; **CREATE** **SCHEMA** fact;
3. /\* Таблица Fact\_Flights - содержит совершенные перелеты.
4. \* Если в рамках билета был сложный маршрут с пересадками - каждый сегмент учитываем независимо\*/
5. **CREATE** **TABLE** fact.fact\_flights (
6. flight\_no bpchar(6) **NOT** **NULL**, -- в задании не указано поле с номером рейса, но совокупность номера рейса (flight\_no) и даты отправления (scheduled\_departure или actual\_departure)
7. --являются естественным ключом, поэтому добавление данного поля упростит работу конечного пользователя с таблицей фактов
8. passenger\_id text **references** dim.dim\_passengers(passenger\_id), -- Пассажир (bookings.tickets.passenger\_name)
9. actual\_departure timestamptz **NOT** **NULL**, -- Дата и время вылета (факт) (bookings.flights.actual\_departure)
10. actual\_arrival timestamptz **NOT** **NULL**, -- Дата и время прилета (факт) (bookings.flights.actual\_arrival)
11. delay\_time\_departure int4 **NOT** **NULL**, -- Задержка вылета (разница между фактической и запланированной датой в секундах) (actual\_departure - scheduled\_departure)
12. delay\_time\_arrival int4 **NOT** **NULL**, -- Задержка прилета (разница между фактической и запланированной датой в секундах) (actual\_arrival - scheduled\_arrival)
13. aircraft\_code **varchar**(30) **NOT** **NULL** **REFERENCES** dim.dim\_aircrafts(aircraft\_code), -- код самолета (bookings.flights.aircraft\_code)
14. airports\_departure\_code **varchar**(30) **NOT** **NULL** **REFERENCES** dim.dim\_airports(airport\_code), -- Аэропорт вылета (bookings.flights.airports\_departure)
15. airports\_arrival\_code **varchar**(30) **NOT** **NULL** **REFERENCES** dim.dim\_airports(airport\_code), -- Аэропорт прилета (bookings.flights.airports\_arrival)
16. fare\_conditions\_id **int** **REFERENCES** dim.dim\_tariff(fare\_conditions\_id), -- ключ класса обслуживания (dim.dim\_tariff)
17. amount **numeric**(10,2) -- стоимость (bookings.ticket\_flights.amount)
18. );
19. -- создаем справочник самолетов dim\_aircrafts
20. **CREATE** **TABLE** dim.dim\_aircrafts (
21. aircraft\_code bpchar(3) **PRIMARY** **KEY**, -- ключ (bookings.aircrafts.aircraft\_code)
22. model **varchar**(20) **NOT** **NULL**, -- модель (bookings.aircrafts.model)
23. manufacturer **varchar**(20) **NOT** **NULL**, -- производитель - первое слово в названии модели самолета
24. "range" int4 **NOT** **NULL** -- расстояние (bookings.aircrafts."range")
25. );
26. -- создаем справочник аэропортов dim\_airports
27. **CREATE** **TABLE** dim.dim\_airports (
28. airport\_code bpchar(3) **PRIMARY** **KEY**, -- ключ (bookings.airports.airport\_code)
29. airport\_name **varchar**(50) **NOT** **NULL**, -- название аэропорта (bookings.airports.airport\_name)
30. airport\_city **varchar**(50) **NOT** **NULL**, -- город аэропорта (bookings.airports.city)
31. longitude **float**(8) **NOT** **NULL**, -- долгота города (bookings.airports.longitude)
32. latitude **float**(8) **NOT** **NULL** -- широта города (bookings.airports.latitude)
33. );
34. -- создаем справочник пассажиров dim\_passengers
35. **CREATE** **TABLE** dim.dim\_passengers (
36. passenger\_id **varchar**(20) **PRIMARY** **KEY**, -- ключ пасажира (bookings.tickets.passenger\_id)
37. passenger\_name text **NOT** **NULL**, -- ФИО пасажира (bookings.tickets.passenger)
38. phone **varchar**(20), -- телефон пассажира (bookings.tickets.contact\_data ->> 'phone')
39. email **varchar**(150) -- email пассажира (bookings.tickets.contact\_data ->> 'email')
40. );
41. -- создаем справочник тарифов dim\_tariff
42. **CREATE** **TABLE** dim.dim\_tariff (
43. fare\_conditions\_id serial **PRIMARY** **KEY**, -- ключ тарифа - не задан, задаем сами
44. fare\_conditions **varchar**(10) **NOT** **NULL** -- тариф (bookings.seats.fare\_conditions)
45. );
46. -- создаем справочник дат dim\_calendar
47. **CREATE** **TABLE** dim.dim\_calendar (
48. **id** serial **PRIMARY** **KEY**, -- ключ
49. "date" **date** **UNIQUE** **NOT** **NULL** , -- дата
50. **YEAR** int4 **NOT** **NULL**, -- год
51. n\_week int4 **NOT** **NULL**, -- неделя
52. day\_week **varchar**(10) **NOT** **NULL**-- день недели
53. );

***Б) Справочник* dim\_calendar в соответствии с условием был сгенерирован скриптом SQL**

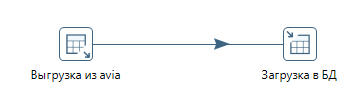
1. -- заполняем справочник дат dim\_calendar скриптом SQL
2. **INSERT** **INTO** dim.dim\_calendar("date",**YEAR**,n\_week, day\_week)
3. **SELECT** gs, date\_part('year', gs), date\_part('week', gs), **to\_char**(gs, 'day')
4. **FROM** generate\_series('2016-09-13', **current\_date**, **interval** '1 day') **as** gs;
5. -- 2016-09-13 - мин дата в таблице полетов. через запрос ее вложить нельзя, так как базы схем между собой не связаны. Макс дата равна текущей на тот случай, если данные будут обновляться.

***B) Для наполнения справочников в БД была создана таблица с метаданными***

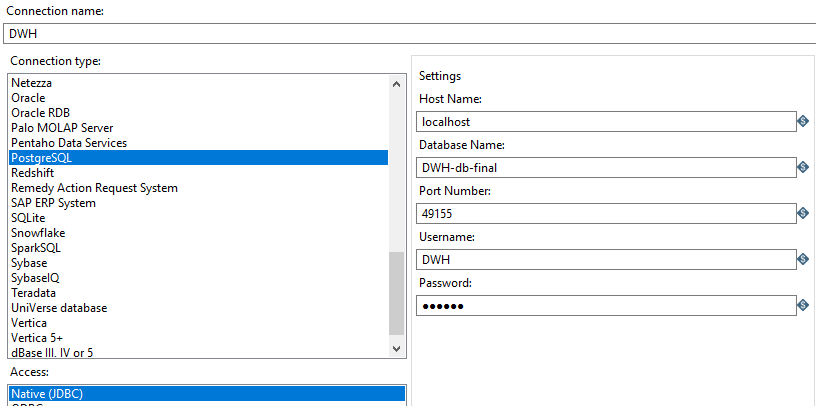
1. -- создаем схему для метаданных
2. **create** **schema** metadata;
3. -- создаем таблицу метаданных для справочников
4. **create** **table** metadata.dim (
5. script **varchar**(256) **not** **null**, -- скрипт выгрузки
6. schema\_name **varchar**(256) **not** **null**, -- название схемы
7. table\_name **varchar**(256) **not** **null**,
8. field **varchar**(256) **not** **null**,
9. ord **smallint**,
10. **primary** **key** (schema\_name, table\_name, field));
11. -- загружаем в таблицу metadata.dim данные для загрузки справочников
12. **insert** **into** metadata.dim (script,schema\_name, table\_name, field, ord)
13. **Values**
14. ('SELECT DISTINCT s.fare\_conditions FROM bookings.seats s ORDER BY s.fare\_conditions', 'dim', 'dim\_tariff', 'fare\_conditions',1),
15. ('SELECT DISTINCT t.passenger\_id, t.passenger\_name, t.contact\_data ->> ''phone'' AS phone, t.contact\_data ->> ''email'' AS email FROM bookings.tickets t',
16. 'dim', 'dim\_passengers', 'passenger\_id',1),
17. ('SELECT DISTINCT t.passenger\_id, t.passenger\_name, t.contact\_data ->> ''phone'' AS phone, t.contact\_data ->> ''email'' AS email FROM bookings.tickets t',
18. 'dim', 'dim\_passengers', 'passenger\_name',2),
19. ('SELECT DISTINCT t.passenger\_id, t.passenger\_name, t.contact\_data ->> ''phone'' AS phone, t.contact\_data ->> ''email'' AS email FROM bookings.tickets t',
20. 'dim', 'dim\_passengers', 'phone',3),
21. ('SELECT DISTINCT t.passenger\_id, t.passenger\_name, t.contact\_data ->> ''phone'' AS phone, t.contact\_data ->> ''email'' AS email FROM bookings.tickets t',
22. 'dim', 'dim\_passengers', 'email',4),
23. ('SELECT DISTINCT a.airport\_code, a.airport\_name, a.city as airport\_city, a.latitude, a.longitude FROM bookings.airports a', 'dim', 'dim\_airports', 'airport\_code',1),
24. ('SELECT DISTINCT a.airport\_code, a.airport\_name, a.city as airport\_city, a.latitude, a.longitude FROM bookings.airports a', 'dim', 'dim\_airports', 'airport\_name',2),
25. ('SELECT DISTINCT a.airport\_code, a.airport\_name, a.city as airport\_city, a.latitude, a.longitude FROM bookings.airports a', 'dim', 'dim\_airports', 'city',3),
26. ('SELECT DISTINCT a.airport\_code, a.airport\_name, a.city as airport\_city, a.latitude, a.longitude FROM bookings.airports a', 'dim', 'dim\_airports', 'latitude',4),
27. ('SELECT DISTINCT a.airport\_code, a.airport\_name, a.city as airport\_city, a.latitude, a.longitude FROM bookings.airports a', 'dim', 'dim\_airports', 'longitude',5),
28. ('SELECT DISTINCT a.aircraft\_code, a.model, LEFT(a.model, POSITION('' '' in model)-1) AS manufacturer, a."range" FROM bookings.aircrafts a','dim', 'dim\_aircrafts', 'aircraft\_code',1),
29. ('SELECT DISTINCT a.aircraft\_code, a.model, LEFT(a.model, POSITION('' '' in model)-1) AS manufacturer, a."range" FROM bookings.aircrafts a','dim', 'dim\_aircrafts', 'model',2),
30. ('SELECT DISTINCT a.aircraft\_code, a.model, LEFT(a.model, POSITION('' '' in model)-1) AS manufacturer, a."range" FROM bookings.aircrafts a','dim', 'dim\_aircrafts', 'manufacturer',3),
31. ('SELECT DISTINCT a.aircraft\_code, a.model, LEFT(a.model, POSITION('' '' in model)-1) AS manufacturer, a."range" FROM bookings.aircrafts a','dim', 'dim\_aircrafts', 'range',4);

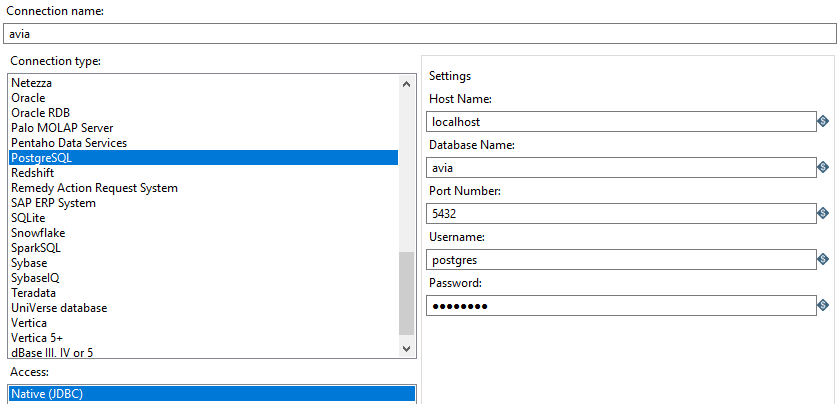
***Г) Описание ETL по наполнению справочников***

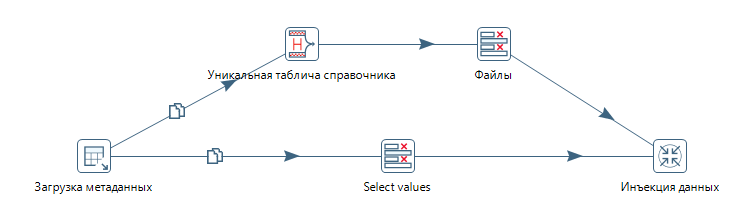
***2 -1) Трансформация загрузки***



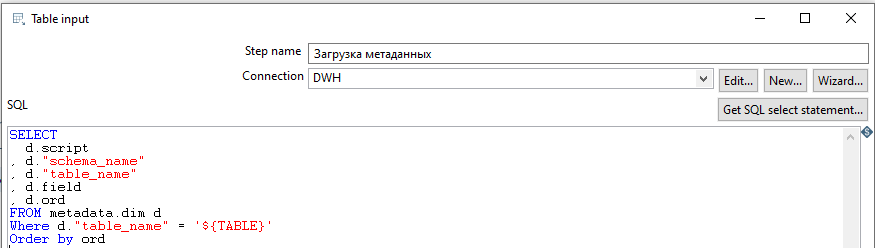
*Подтрансформация с пустыми элементами. Единственное что указано, это БД для выгрузки avia в table input и БД для загрузки DWH в table output. Данные заполняются через инъекцию данных*



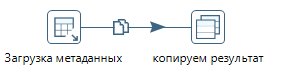


***2-2 Инъекция данных***

Загрузка метаданных осуществляется из таблице с метаданными. На ETL инъекцию в трансформацию загрузки подаем SQL скрипт, название схемы, таблицы и полей. Также в трансформации задаем переменную TABLE, которая нам позволит осуществить последовательную загрузку данных в БД

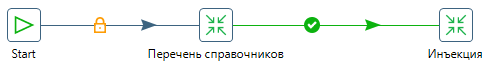


***2-3 Перечень справочников***

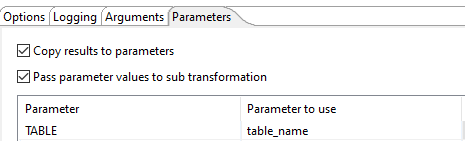


Извлекаем перечень таблиц под заполнения, для последовательной загрузки таблиц в задании.

**2-4 Задание на создание всех справочников**

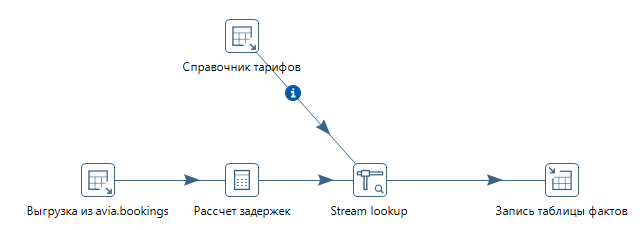


Задание на последовательную инъекцию и загрузку данных в БД. Последовательность задается через присвоение параметру TABLE значения из поля table\_name перечня справочника и активации «execute every input row» .

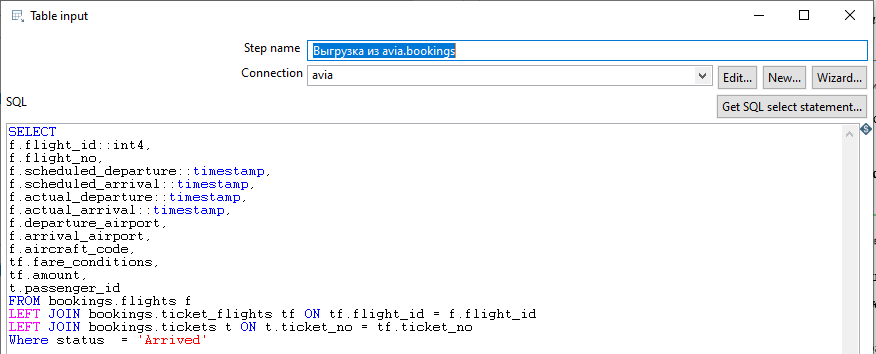


***Д) Описание ETL по заполнению таблицы фактов***

**3-1 Таблица фактов**



Сперва выгружаем всю доступную информацию из БД avia. Далее рассчитываем время задержки вылета и прилета. Так как в БД avia нет естественного ключа тарифов, то к значениям подтягиваем данные из справочника. Далее записываем результат в БД.



1. ***Проверка качества таблиц***

***А) Создаем схему и rejected-таблицы в БД "DWH-db-final"***

Поля данных rejected-таблицы идентичны полям таблиц справочником, за исключением:

1. Поля не содержат PRIMARY KEY и FOREIGN KEY
2. Поля не содержат какие либо ограничения
3. Тип автоматически сформированных полей serial изменен на int4
4. Добавлено поле reason\_for\_rejection, в которое будет помещна причина отсеивания данных в rejected-таблицы.
5. **CREATE** **SCHEMA** rejected;
6. **CREATE** **TABLE** rejected.fact\_flights (
7. flight\_no bpchar(6),
8. passenger\_id text,
9. actual\_departure timestamptz ,
10. actual\_arrival timestamptz,
11. delay\_time\_departure int4,
12. delay\_time\_arrival int4,
13. aircraft\_code **varchar**(30),
14. airports\_departure\_code **varchar**(30),
15. airports\_arrival\_code **varchar**(30),
16. fare\_conditions\_id **int**,
17. amount **numeric**(10,2),
18. reason\_for\_rejection TEXT -- поле с причиной отклонения
19. );
20. **CREATE** **TABLE** rejected.dim\_aircrafts (
21. aircraft\_code bpchar(3),
22. model **varchar**(20),
23. manufacturer **varchar**(20),
24. "range" int4,
25. reason\_for\_rejection TEXT
26. );
27. **CREATE** **TABLE** rejected.dim\_airports (
28. airport\_code bpchar(3),
29. airport\_name **varchar**(50),
30. airport\_city **varchar**(50),
31. longitude **float**(8),
32. latitude **float**(8),
33. reason\_for\_rejection TEXT -- поле с причиной отклонения
34. );
35. **CREATE** **TABLE** rejected.dim\_passengers (
36. passenger\_id **varchar**(20),
37. passenger\_name text,
38. phone **varchar**(20),
39. email **varchar**(150),
40. reason\_for\_rejection TEXT -- поле с причиной отклонения
41. );
42. **CREATE** **TABLE** rejected.dim\_tariff (
43. fare\_conditions\_id int4,
44. fare\_conditions **varchar**(10),
45. reason\_for\_rejection TEXT -- поле с причиной отклонения
46. );
47. **CREATE** **TABLE** rejected.dim\_calendar (
48. **id** int4 ,
49. "date" **date** ,
50. **YEAR** int4 ,
51. n\_week int4 ,
52. day\_week **varchar**(10),
53. reason\_for\_rejection TEXT -- поле с причиной отклонения
54. );

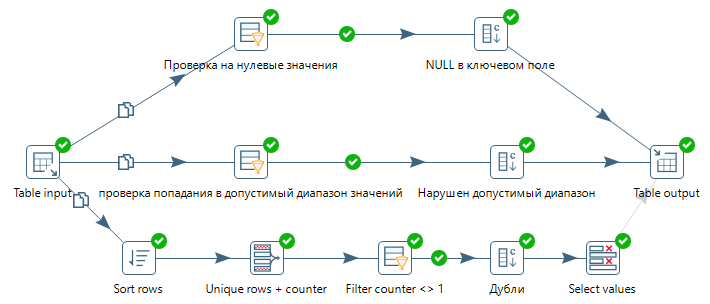
***Проверка dim\_aircraft***



Сперва выгружаем данные и копии выгрузки отправляем на проверки.

* 1. Данный справочник не должен содержать полей с NULL. Фильтруем на наличие ошибочных строк. К результату добавляем поле с причиной отбора «NULL в ключевом поле»
  2. Убедимся, что значения с максимальным расстоянием перелетов самолетов корректно - значения больше 0, через фильтр «range < 0» (отбираем ошибочные). Добавим поле с причиной отбора «Неверное значение в поле range».
  3. В названии модели первое слово является названием производителя. Проверим, что значения поле с названием производителя является первым словом в модели и добавим причину отбора для ошибок.
  4. Проверим на наличие дублей. Сперва отсортируем проверяемое поле, сформируем список уникальных значений с добавлением поля с количеством повторов в исходной таблице. Отфильтруем строки, которые имею повторы и добавим поле с причиной отбора. Уберем поля с кол-вом, которого нет в rejected-таблице.
  5. Результат 1-4 загрузим в соответствующую rejected-таблицу.

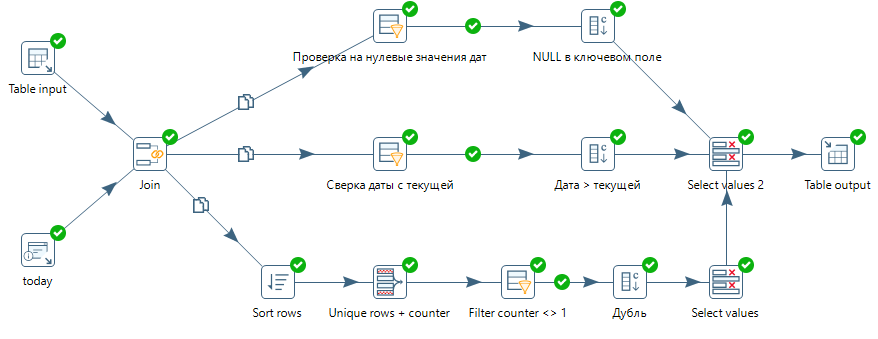
**Проверка dim\_airports**



Сперва выгружаем данные и копии выгрузки отправляем на проверки.

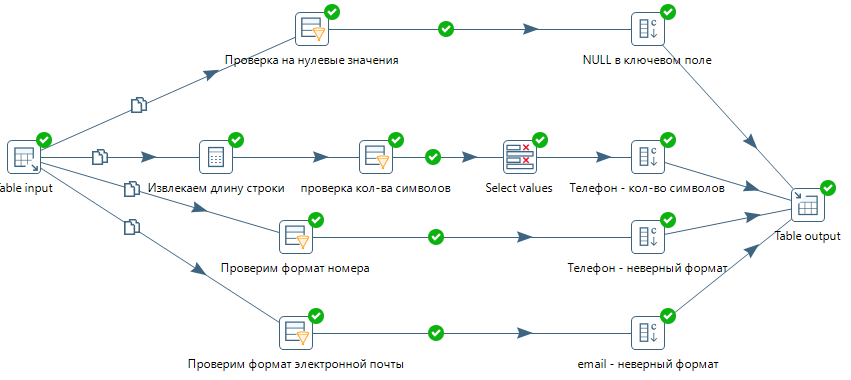
* 1. Данный справочник не должен содержать полей с NULL. Фильтруем на наличие ошибочных строк. К результату добавляем поле с причиной отбора «NULL в ключевом поле»
  2. Значения широты и долготы должны находится в диапазоне от минус 180 до 180. У строкам со значением вне диапазона добавляем причину отбора.
  3. Проверим на наличие дублей. Сперва отсортируем проверяемое поле, сформируем список уникальных значений с добавлением поля с количеством повторов в исходной таблице. Отфильтруем строки, которые имею повторы и добавим поле с причиной отбора. Уберем поля с кол-вом, которого нет в rejected-таблице.
  4. Результат предыдущих шагов загрузим в соответствующую rejected-таблицу.

**Проверка dim\_calendar**



Сперва выгружаем данные, джоиним к каждой сроке текущую дату и копии результата отправляем на проверки.

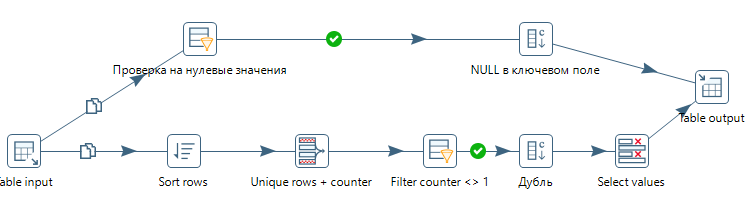
* 1. Поле с датой не должно содержать NULL. Фильтруем на наличие ошибочных строк. К результату добавляем поле с причиной отбора «NULL в ключевом поле». В заключение уберем поле с текущей датой.
  2. Убедимся, что указанные даты меньше или равны текущей. К строкам с ошибками добавляем причину отбора. В заключение уберем поле с текущей датой.
  3. Проверим на наличие дублей. Сперва отсортируем проверяемое поле, сформируем список уникальных значений с добавлением поля с количеством повторов в исходной таблице. Отфильтруем строки, которые имею повторы и добавим поле с причиной отбора. Уберем поля с кол-вом, которого нет в rejected-таблице.
  4. Результат предыдущих шагов загрузим в соответствующую rejected-таблицу.

**Проверка dim\_passengers**

Сперва выгружаем данные и копии выгрузки отправляем на проверки.

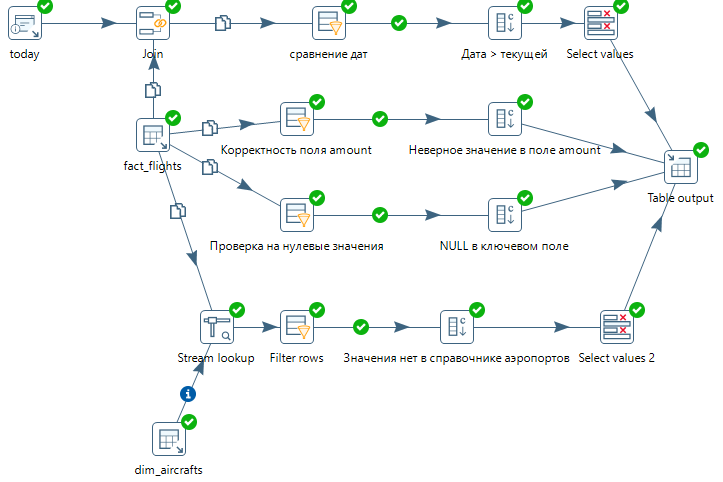
* 1. Ключевые поля не должны содержать NULL. Фильтруем на наличие ошибочных строк. К результату добавляем поле с причиной отбора «NULL в ключевом поле». В заключение уберем поле с текущей датой.
  2. Поле с телефоном должно содержать 12 символов. Извлекаем длину, проверяем ее на равенство 12. Убираем добавленное поле с длинной и добавляем поле с причиной отбора.
  3. Формат поля телефонного новемер – первый знак + далее 11 цифр. Отбираем строки, значения поля которого не является «\+[0-9]{11}». Добавляем причину отбора.
  4. Emal обязательно должен содержать символ @ и точку после данного символа. Осуществляем проверку регулярки на соответствие «%@\w+\.\w+» с пропуском нулевых значений. К строкам с ошибками добавляем причину отбора.
  5. Результат предыдущих шагов загрузим в соответствующую rejected-таблицу.

**Проверка dim\_tariff**



Сперва выгружаем данные и копии выгрузки отправляем на проверки. Делаем стандартную проверку на NULL и дубли, с последующей загрузкой ошибок.

**Проверка таблицы фактов**



Сперва выгружаем данные и копии выгрузки отправляем на проверки.

* 1. Сперва к выгрузке джоиним текущую дату. Затем отбираем строки, у которых дата фактического вылета превышает текущую. Добавляем причину отбора и убираем поле с текущей датой
  2. Проверяем что значение поля amount >= 0. К срокам с отрицательными числами добавляем причину отбора
  3. Выполняем стандартную проверку на NULL в ключевых полях
  4. К выгрузке добавляем значение поля из справочника с самолетами. Отфильтровываем поля, значения которых не было в справочнике. Добавляем поле с причиной отбора. Убираем лишние поля.
  5. Результат предыдущих шагов загрузим в соответствующую rejected-таблицу.

**Ссылка на github:**