

**Дипломная работа  
по профессии  
"Инженер данных"**

**Выполнил:**

**Тульев Александр Сергеевич**

**Группа: DEG-4**

**февраль 2022г.**

# Содержание

	<b>№ стр</b>
1. Вступление	3
2. Сбор (получение) данных из базы источника (MS SQL)	5
3. Анализ данных (Python. Jupyter Notebook)	6
4. Структура Хранилища Данных	8
5. Скрипт создания БД, таблиц (PostgreSql)	11
6. ETL (Pentaho)	14
7. Построение дашбордов (MetaBase)	22
8. Выводы	34
9. Использованное ПО	38

## 1. Вступление.

Вот и пролетел год учебы по профессии "Дата-инженер" (Data Engineer).

А кто же такой дата инженер и чем он занимается?

Дата-инженер обычно отвечает за управление рабочими процессами, конвейерами обработки и ETL-процессами.

Из названия следует, что инженерия данных связана с данными, а именно с их доставкой, хранением и обработкой. Соответственно, основная задача инженеров — обеспечить надежную инфраструктуру для данных.

То есть инженер данных отвечает за следующие этапы:

- ) **сбор данных;**
- ) **перемещение и хранение данных;**
- ) **подготовка данных.**

Дата-инженер (Data Engineer) занимается **ETL**-процессами, то есть обрабатывает данные: достает (**extract**) их из сырых источников, трансформирует (**transform**) и загружает (**load**).

После предварительной обработки, очистки от повторов, ошибок, ненужных уточнений, он автоматизирует выполнение скриптов и, если нужно, настраивает мониторинги, алерты (сигналы о том, что в моделях что-то пошло не так), задает расписание, по которому сервис или программа будут работать с данными (шедуллит).

Инженер данных нужен для того, чтобы аналитики имели возможность использовать данные для решения бизнес-задач — например, для оптимизации запросов, оценки прибыльности и рентабельности продуктов, отчетности и так далее. Он создает pipeline данных (последовательные стадии работы с данными), интеграцию различных систем и источников, предоставляет пользователям инструменты работы с данными.

Как я пришел на данный курс?

Руководство нашей организации очень заинтересовано в анализе данных. До недавнего времени весь анализ велся только в Excel.

Чуть позже был установлен и запущен в работу MetaBase. Меня назначили ответственным. Из знаний у меня было только знание SQL.

Знаний стало не хватать.

Было принято решение начинать учиться.

Курс выбирали недолго, подбирали так, чтобы он охватывал много различных направлений (на тот момент не понимали до конца, что нам в итоге нужно).

В процессе обучения я приобрел необходимые знания и теперь уже использую их на практике в своей работе.

## 2. Сбор (получение) данных из базы источника (MS SQL).

Я решил не использовать предложенный набор данных, а получить свой.

Итак, у нас на предприятии изначально имеются две MSSQL базы (две программы. Разработчики этих систем не мы).

1. Система управления гостиничным бизнесом;
2. Система управления клиникой (медицинские услуги).

Что я сделал: подготовил sql скрипт, с помощью которого вытаскиваю первоначальный набор данных и сохраняю его в csv файл.

Для данной работы использовал только одну базу данных: Система управления гостиничным бизнесом.

Период для выгрузки использовал: 01.01.2019 – 30.06.2019

В итоге получил 2531 запись.

Отдельно надо отметить, что я обезличил персональные данные и данные по суммам взял не из базы, а сгенерировал случайным образом (функция RAND() в MSSQL).

В данном документе SQL скрипт не привожу. С ним можно ознакомиться в Github в папке:

*\Скрипты\Скрипт выгрузки брони MSSQL.sql*

Полученный набор данных смотрим:

*\InputData\Bookings.csv*

### 3. Анализ данных (Python. Jupyter Notebook).

После получения данных я приступил к анализу данных.

База данных "Bookings" содержит данные за период с 01.01.2019 – 30.06.2019.

Описание полей БД.

№	Поле	Описание
1	BookingNumber	Номер брони
2	Room	Номер комнаты по брони
3	TypeRoom	Тип комнаты, занимаемой гостем
4	Sex	Пол гостя
5	NumberVisits	Кол-во визитов гостя
6	N_Arrival	Порядковый номер заезда гостя
7	DateArrival	Дата заезда гостя
8	DateDeparture	Дата выезда гостя
9	ReservationStatus	Статус брони, может принимать значения (IN, OUT, CANCEL, RES)
10	DaysInClinic	Проведено дней в клинике
11	Rate_Passed	Мед. программа, которую прошел гость
12	BirthDay	День рождения гостя
13	AgeOnArrival	Возраст на момент заезда
14	City	Город гостя
15	Geo	Геолокация гостя
16	Amount_Bookings	Сумма брони
17	Amount_Additionally	Сумма дополнительных услуг

Для основного анализа использовал **Python. Jupyter Notebook**.

Краткие выводы по анализу:

Всего в наборе 2531 запись и 17 полей.

Целочисленных: 6 полей, полей объектов: 10

В наборе имеются пропуски (Null) по полям: Room (5 записей), BirthDay (4 записи), City (125 записей) и указанным возрастом гостей = 200 лет.

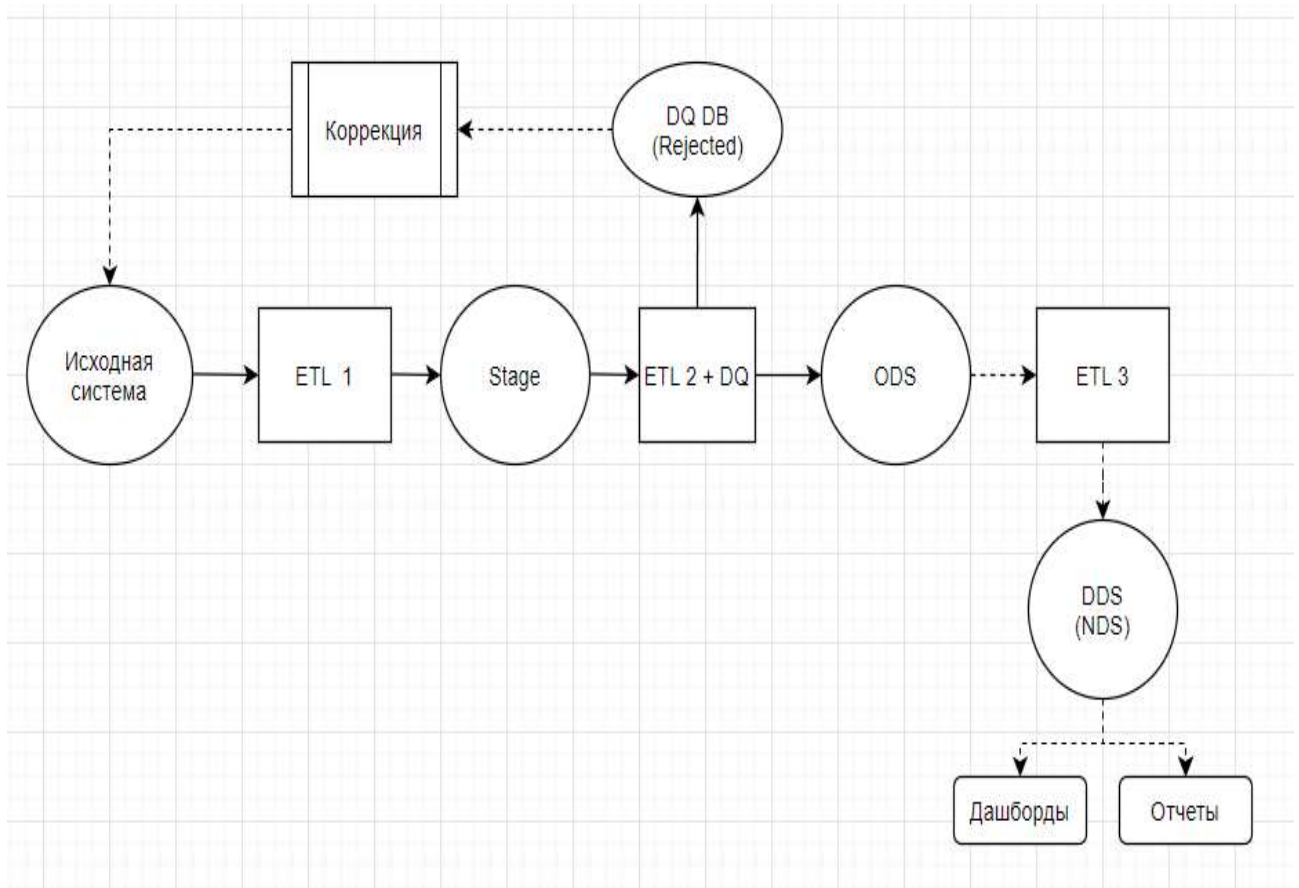
В целом с набором данных не так плохо. Надо будет выкидывать строки с Null значениями и с возрастом = 200.

**С подробным анализом можно ознакомиться:**

*\\_Jupyter Notebook\Bookings\_Analysis.ipynb*

#### 4. Структура Хранилища данных.

Для выполнения данной работы решил использовать хранилище БД со следующей структурой:



Оригинал скрина:

\\Скрины\\1. Состав ХД схема.png

**Stage (Staging)** Загрузка данных из исходной системы. Служит для уменьшения нагрузки на исходную систему, путем загрузки всех данных в отдельную БД. Нет внешних ключей;

**ODS (Operational Data Store)** - отдельная структура между Staging и DWH. Данные из Stage проверяются на корректность и очищаются;



**NDS (Normalized Data Storage)** - внутреннее хранилище данных в виде нормализованной БД;

**DDS (Dimensional Data Storage)** - хранилище многомерных витрин данных. Данные нормализуются, приводятся к единому формату таблицы фактов и измерений;

**ETL (Extract Transform Load)** - скрипты загрузки и трансформации;

**DQ (Data Quality)** - процесс проверки данных в ХД на корректность и полноту.

**DQ DB или Rejected (отклоненные записи)**. Сюда собираю записи с некачественными строками.

Отдельно слой **NDS (Normalized Data Storage)** не стал делать.

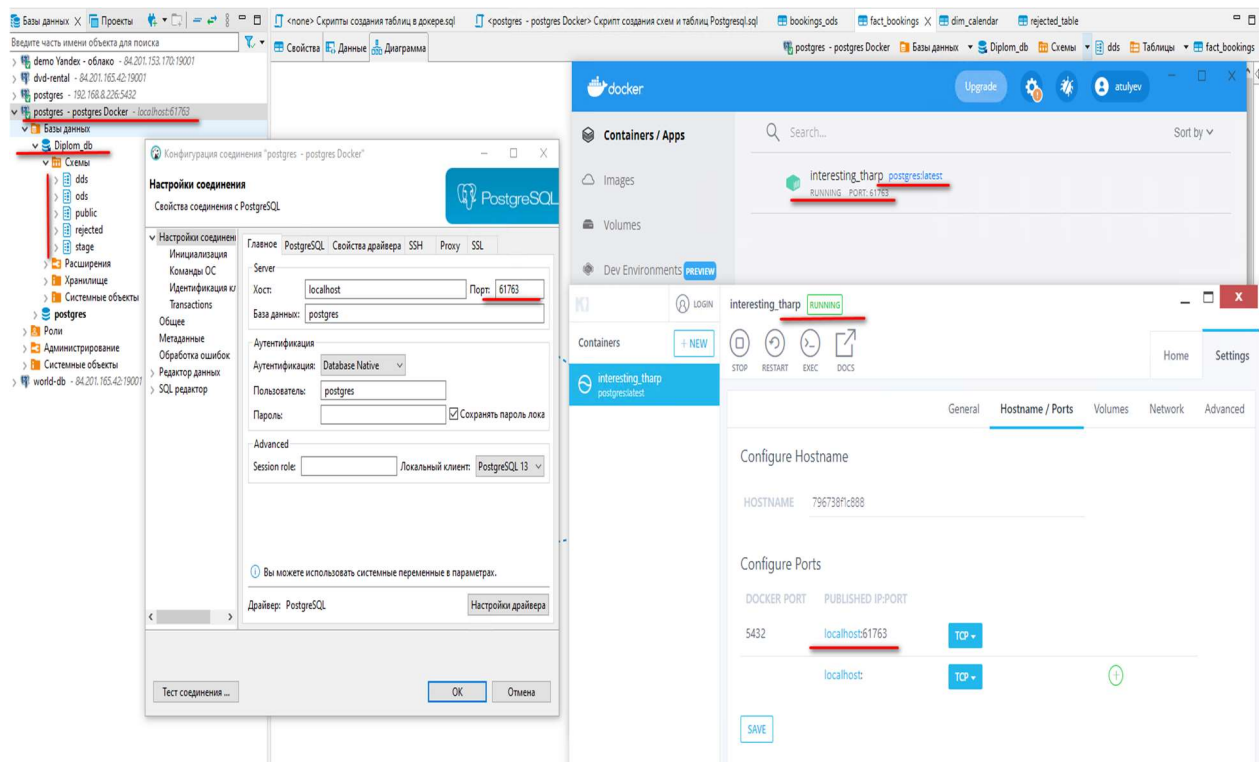
Слой **DDS (Dimensional Data Storage)** также является нормализованным (отвечает условиям нормальных форм), пусть и не таким высоким.

Про исходную систему я уже писал: это MSSQL плюс выгрузка в csv файл.

Слои Stage, ODS, NDS, DDS, DQ (Rejected) я создаю в Postgresql.

Хотя у меня и имеется локальная установка Postgresql, я использовал установку Postgresql в контейнере Docker.

Настройки Docker, Kitematic и соединения Postgresql:



Оригинал скрина:

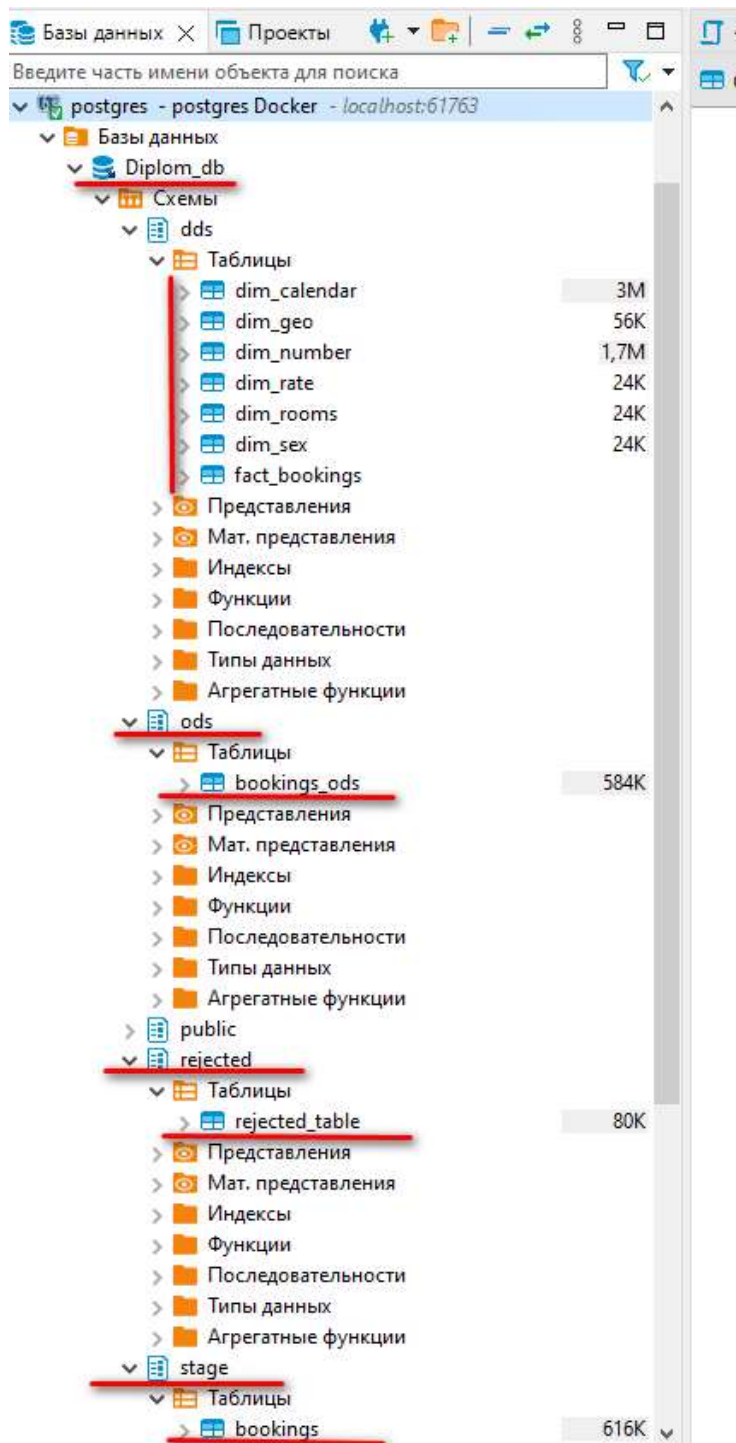
└─ Скрины\2. Docker.png

## 5. Скрипт создания БД, таблиц (PostgreSql).

Далее я создал SQL скрипт с помощью которого создаю базу данных, схемы и таблицы, а также заполняю слой stage.

Я создал базу данных: `Diplom_db` и в ней несколько схем: `stage`, `ods`, `dds` и `rejected` (по одной схеме на каждый слой). Названия схем соответствуют слоям.

Прикладываю скрин Базы данных, схем и таблиц в Postgresql:



Прикладываю ER-диаграммы созданных схем Stage, Ods и Rejected:

<none> Скрипты создания таблиц в докере.sql

<postgres - postgres Docker> Скрипт создания схем и таблиц Postgresql.sql

bookings\_ods

Свойства

Диаграмма

stage.bookings

123 bookingnumber	int4
ABC room	bpchar(10)
ABC typeroom	bpchar(10)
ABC sex	bpchar(1)
123 numbervisits	int2
123 n_arrival	int2
datearrival	date
datedeparture	date
ABC reservationstatus	bpchar(3)
123 daysinclinic	int2
ABC rate_passed	bpchar(10)
birthday	date
123 ageonarrival	int2
ABC city	bpchar(50)
ABC geo	bpchar(50)
123 amount_booking	numeric(10, 2)
123 amount_additionaly	numeric(10, 2)

ods ×

Свойства

Диаграмма

ods.bookings\_ods

123 bookingnumber	int4
ABC room	bpchar(10)
ABC typeroom	bpchar(10)
ABC sex	bpchar(1)
123 numbervisits	int2
123 n_arrival	int2
datearrival	date
datedeparture	date
ABC reservationstatus	bpchar(3)
123 daysinclinic	int2
ABC rate_passed	bpchar(10)
birthday	date
123 ageonarrival	int2
ABC city	bpchar(50)
ABC geo	bpchar(50)
123 amount_bookings	numeric(10, 2)
123 amount_additionaly	numeric(10, 2)

rejected ×

Свойства

Диаграмма

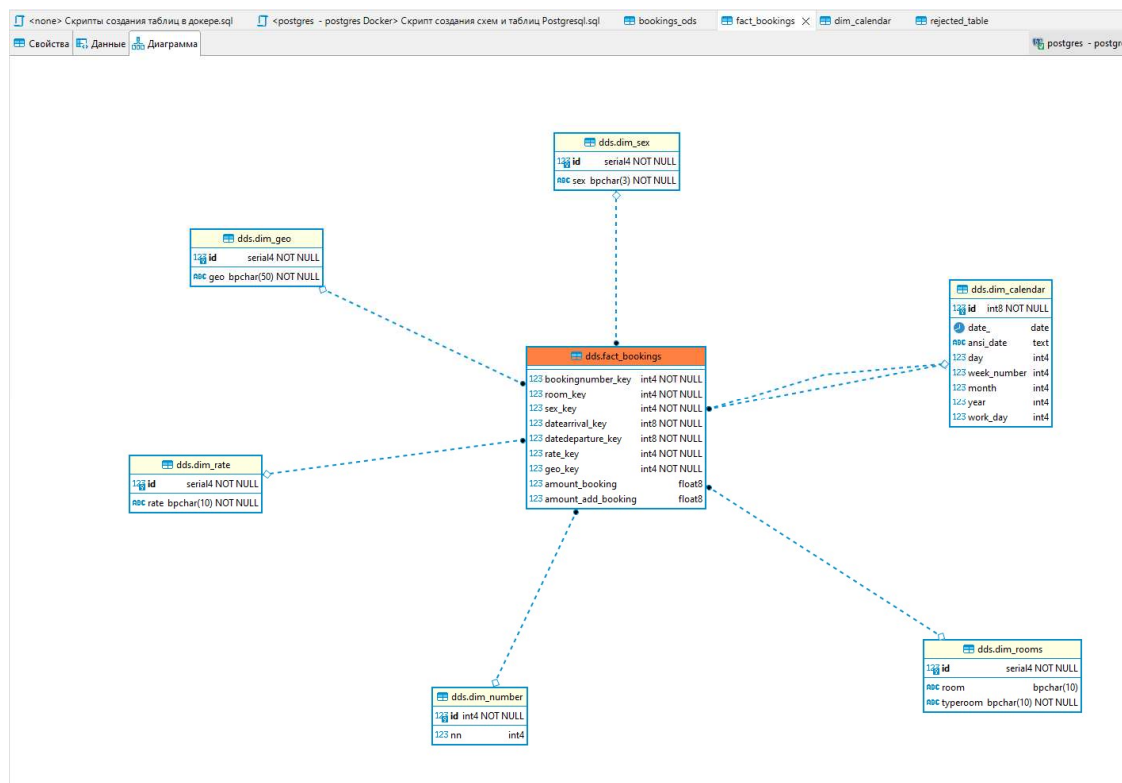
rejected.rejected\_table

actualdate	timestamp
ABC error	text
123 bookingnumber	int4
ABC room	bpchar(10)
ABC typeroom	bpchar(10)
ABC sex	bpchar(1)
123 numbervisits	int2
123 n_arrival	int2
datearrival	date
datedeparture	date
ABC reservationstatus	bpchar(3)
123 daysinclinic	int2
ABC rate_passed	bpchar(10)
birthday	date
123 ageonarrival	int2
ABC city	bpchar(50)
ABC geo	bpchar(50)
123 amount_booking	numeric(10, 2)
123 amount_additionaly	numeric(10, 2)

Оригинал скрина:

\\Скрины\\3. ER диаграммы Stage, ODS, Rejected.png

И, наконец, прикладываю ER диаграмму схемы DDS:



Оригинал скрина:

[\Скринь\4. ER-диаграмма ХД \(DDS\).png](#)

В данном документе не привожу скрипт по созданию этих таблиц.

Со скриптом можно ознакомиться:

[\Скрипты\Скрипт создания схем и таблиц Postgresql.sql](#)

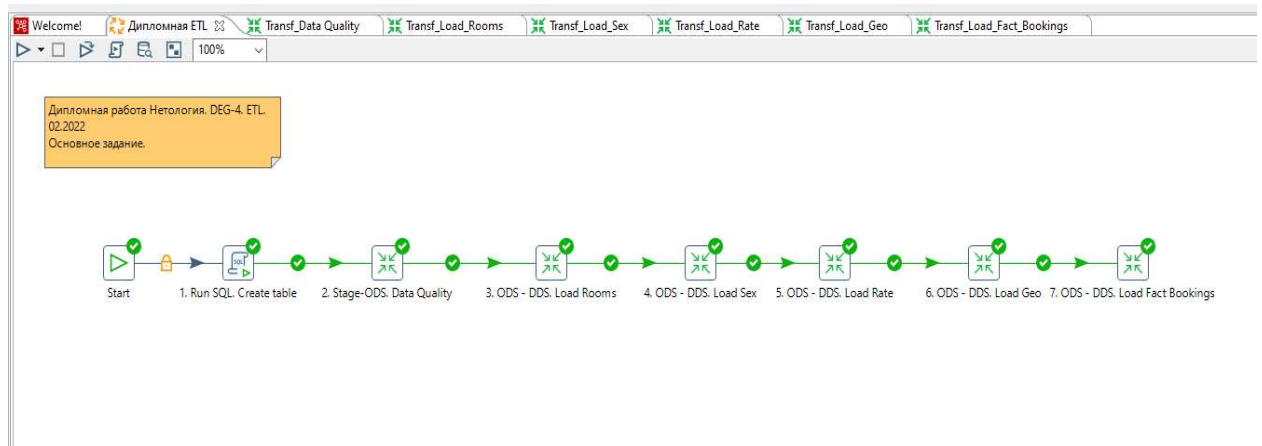
В дальнейшем этот скрипт использую в трансформации Pentaho (об этом подробнее позже).

Этим же скриптом произвожу заполнение слоя stage, для этого скопировал файл csv в контейнер.

## 6. ETL (Pentaho).

Наполнение базы данных при помощи ETL построил по следующему варианту:

В основу выбрал задание (Job).



Оригинал скрина:

Скриншоты\5.Основной Job.png

Оригиналы всех трансформаций и задания находятся:

Pentaho\

**Последовательно** выполняю шаги:

1. Шаг 1. SQL-скрипт создания таблиц из внешнего файла.

То есть каждый раз при запуске задания удаляю и по-новому создаю все таблицы и отношения.

Считаю, что каждый раз загружаю полный объем информации (нужный период) и работаю только с ним. Времени это много не занимает.

Таблица фактов имеет внешние ключи на таблицы измерений.

Таблицы измерений имеют первичный ключ тип serial, то есть автоинкремент.

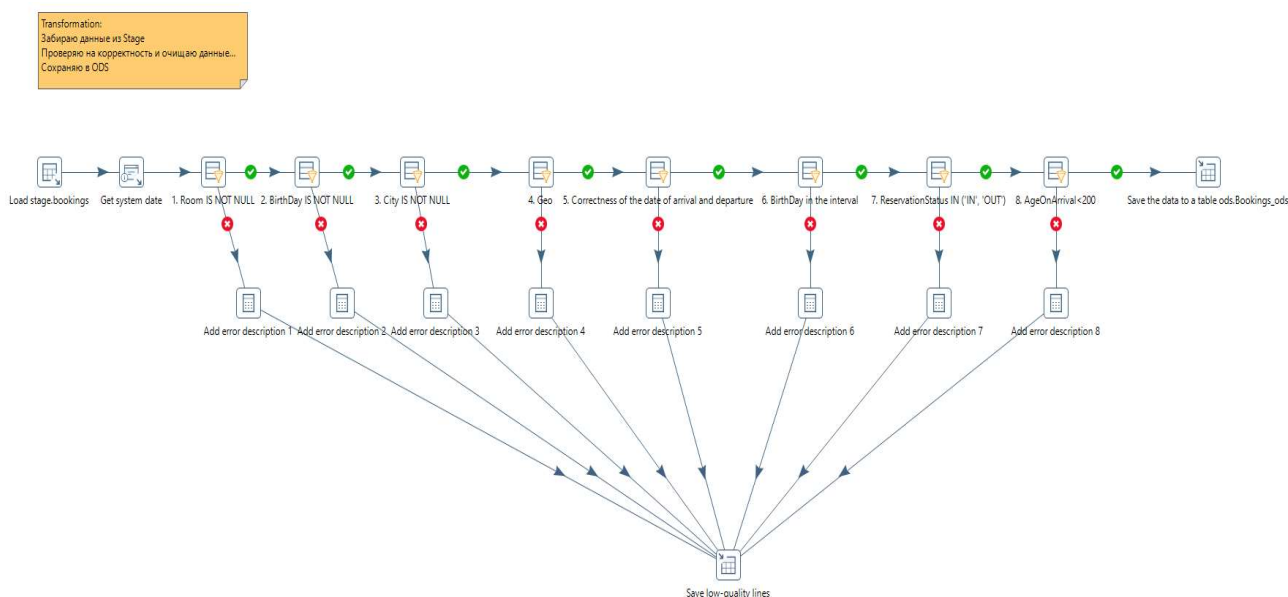
Отдельно нужно остановиться на создании и заполнении таблиц измерений `dds.dim_Calendar` и `dds.dim_Number`. Создаю эту таблицу SQL-скриптом и **заполняю** также SQL-скриптом.

Скрипт создания таблиц:

Скрипты\Скрипт создания схем и таблиц Postgresql.sql

2. Шаг 2. Загрузка данных из Stage в ODS. На данном шаге данные проверяю на корректность и очищаю. Отклоненные записи сохраняю в таблице rejected.rejected\_table.
3. Шаги с 3 по 6. Загрузка данных в таблицы измерения. Прекрасно понимаю, что эти шаги можно было запустить параллельно, но сделал так.
4. Шаг 7. Используя уже загруженные таблицы измерения, загружаю данные в таблицу фактов и подтягиваю к этим данным ключи из таблиц измерений.

Теперь посмотрим на трансформации (шаги 2 - 7 из задания).



Оригинал скрина:

Скриншоты \ 6. 2 шаг. Stage-ODS. Data Quality.png

В этой трансформации я читаю данные из схемы Stage таблицы Bookings.

Добавляю колонку с текущей системной датой и временем во все строки с одинаковым значением. Эта колонка будет также записываться в таблицу bookings\_ods схемы ods. Эта колонка обозначает дату и время последнего переноса данных.

После этого проверяю данные на Null и корректность.

Строки не проходящие проверку уходят на шаги "Add error description N" где добавляю описание ошибки и записываю в таблицу rejected.rejected\_table на шаге "Save low-quality lines".

Проводимые проверки:

- На Null проверяю поля: Room, BirthDay, City;
- Гео проверяю на значение: "Уточнить при заезде";
- Проверяю корректность даты выезда. Дата выезда должна быть позже даты заезда;
- Проверяю чтоб дата рождения находилась в интервале: 01.01.1931 – 01.07.2019;
- Проверяю статус брони, чтобы он был в списке: ('IN','OUT');
- Проверяю возраст на момент заезда не равнялся 200 (200 выставляется в момент выгрузки из базы источника строкам без указания даты рождения).

На шаге "Save the data to a table ods.Bookings\_ods" сохраняю нормальные данные, поступившие после фильтров, в таблицу ods.Bookings\_ods.

Truncate table не использую так как таблицы при запуске задания каждый раз пересоздаются.

Теперь посмотрим на шаги 3-6: Создание таблиц измерений DDS.

Создание таблицы измерений dds.dim\_rooms (шаг 3 задания).

Transformation:  
Забираю данные из ODS.  
Уникальные записи  
Подготовка таблицы измерений Rooms



Оригинал скрина:

\\Скринь\7. 3 шаг. ODS - DDS. Load Rooms.png



В этой трансформации всего два шага. На первом шаге SQL запросом выбираю нужные данные и на втором шаге записываю эти данные в ранее созданную таблицу измерений.

SQL запрос:

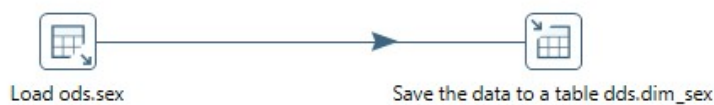
```
SELECT
    distinct room
    , typeroom
FROM
    ods.bookings_ods
```

Аналогично работают следующие трансформации (шаги 4,5,6 задания).

Посмотрим и на них.

Создание таблицы измерений dds.dim\_sex (шаг 4 задания).

Transformation:  
Забираю данные из ODS  
Уникальные записи  
Подготовка таблицы измерений Sex



Оригинал скрина:

\\Скрины\\8. 4 шаг. ODS - DDS. Load Sex.png

SQL запрос:

```
SELECT
    distinct
    CASE
        WHEN sex='M' THEN 'Муж'
        WHEN sex='F' THEN 'Жен'
        ELSE sex
    END AS sex
FROM
    ods.bookings_ods
```

Здесь дополнительно перевожу пол в более понятный вариант.

Создание таблицы измерений dds.dim\_rate (шаг 5 задания).

Transformation:  
Забираю данные из ODS  
Уникальные записи  
Подготовка таблицы измерений Rate



Оригинал скрина:

*\\_Скринь\9. 5 шаг. ODS - DDS. Load Rate.png*

SQL запрос:

```
SELECT
    distinct rate_passed as rate
FROM
    ods.bookings_ods
```

## Создание таблицы измерений dds.dim\_geo (шаг 6 задания).

Transformation:  
Забираю данные из ODS  
Уникальные записи  
Подготовка таблицы измерений Geo



Оригинал скрина:

*\\_Скрины\10. 6 шаг. ODS - DDS. Load Geo.png*

SQL запрос:

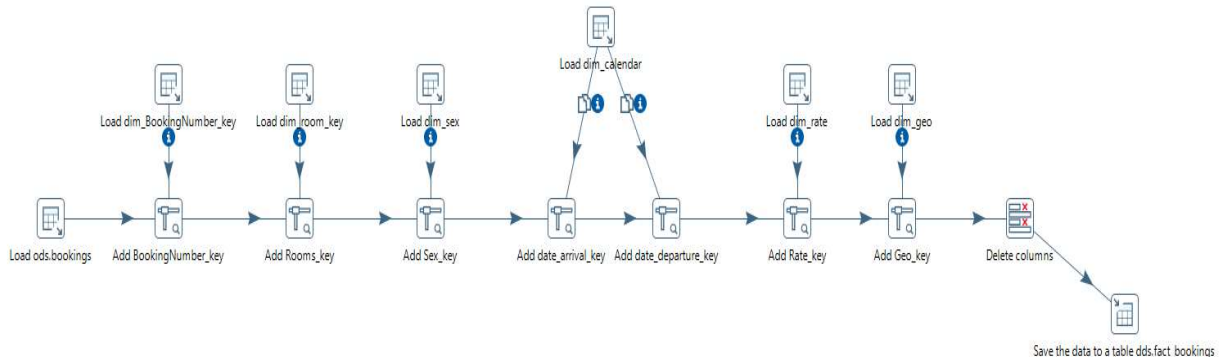
```
SELECT
    distinct Geo
FROM
    ods.bookings_ods
```

Все эти трансформации у меня получились простые (мало шагов в трансформации). Все проверки по качеству провел раньше. Дополнительных полей в рамках данной работы решил не создавать.

После этой трансформации у меня готовы и заполнены данными все таблицы измерения.

В результате, я подошел к основной трансформации: наполнение таблицы фактов dds.fact\_bookings.

Transformation:  
Забирать данные из ODS  
Подготовка таблицы фактов Bookings



Оригинал скрина:

*\\_Скринь\11. 7 шаг. ODS - DDS. Load Fact Bookings.png*

Оригинал данной трансформации находится:

*\\_Pentaho\Transf\_Load\_Fact\_Bookings.ktr*

В этой трансформации можно было использовать для добавления ключей из таблиц измерений следующие элементы Pentaho "DataBase lookup" или "Stream lookup" или "Join". Я решил остановиться на "Stream lookup". По производительности особо разницы не заметил. Да и набор данных не такой большой.

**И так логика работы основной трансформации (Шаг 7 задания):**

1. Шаг 1. SQL запросом получаю необходимые данные с таблицы ods.bookings\_ods.

SQL запрос:

```
SELECT
    bookingnumber,
    room,
    typeroom,
    CASE
        WHEN sex='M' THEN 'Муж'
        WHEN sex='F' THEN 'Жен'
        ELSE sex
    END AS sex,
    datearrival,
```

```

datedeparture,
rate_passed,
geo,
amount_booking,
amount_additionally as amount_add_booking

FROM
ods.bookings_ods

```

2. Шаги 2-3. Читаю таблицу измерений dds.dim\_Number и подтягиваю к данным ключ из этой таблицы. То есть добавляю поле BookingNumber\_key.
3. Шаги 4-5. Читаю таблицу измерений dds.dim\_Rooms и подтягиваю к данным ключ из этой таблицы. Добавляю поле Room\_key.
4. Шаги 6-7. Читаю таблицу измерений dds.dim\_Sex и подтягиваю к данным ключ из этой таблицы. Добавляю поле sex\_key.
5. Шаги 8-10. Читаю таблицу измерений dds.dim\_Calendar и подтягиваю к данным ключ из этой таблицы. Добавляю поля datearrival\_key и datedeparture\_key.
6. Шаги 11-12. Читаю таблицу измерений dds.dim\_Rate и подтягиваю к данным ключ из этой таблицы. Добавляю поля rate\_key.
7. Шаги 13-14. Читаю таблицу измерений dds.dim\_Geo и подтягиваю к данным ключ из этой таблицы. Добавляю поля geo\_key.
8. Шаг 15. Удаляю 8 уже не нужных столбцов из набора данных, которые были необходимы для связи таблиц.
9. Шаг 16. Записываю данные в таблицу фактов dds.fact\_bookings. Commit size установил в значение 50000.

Ну вот и все, наполнил хранилище данными.

Понимаю, что можно было уменьшить количество шагов, но решил для себя все сделать более подробно.

В итоге во время последней трансформации я создаю 6 таблиц измерений, 1 таблицу фактов (схема звезда).

Набор данных небольшой. Общее время отработки всего задания буквально несколько секунд на моем ПК. Считаю это отличным временем выполнения.

## 7. Построение дашбордов (MetaBase).

Теперь, когда данные готовы, можно приступить к визуализации и созданию дашбордов.

Я решил воспользоваться инструментом **MetaBase - open-source инструмент для бизнес-аналитики**, в котором можно писать запросы к данным нескольких видов и визуализировать результаты на дашбордах. А также можно отправлять данные, построенные графики и таблицы на почту и в Slack.

MetaBase предоставляет официальный образ Docker для установки.

Запускаю MetaBase и устанавливаю настройки для подключения к базе данных PostgreSQL. В которой уже хранятся мои загруженные данные.

The screenshot shows the MetaBase administration interface in a web browser. The address bar indicates the URL is `metabase.kivach.local:3000/admin/databases/12`. The browser's address bar also shows several search engines and services like Яндекс, Конвертация, Релизы, Управление аптеч..., Форум 1С програм..., Волшебный форум, GISMETEO: погода..., Gmail, Восстановление б..., and Зарегистри. The interface has a dark blue header with navigation links: Управление Metabase, Настройки, Люди, Модель данных, Базы данных, Привилегии, and Разрешение проблем. The main content area is titled 'БАЗЫ ДАННЫХ > DIPL\_1'. It contains a form for configuring a database connection. The 'Тип базы данных' (Database type) is set to 'PostgreSQL'. The 'Имя' (Name) is 'Dipl\_1'. The 'Хост' (Host) is '192.168.8.226'. The 'Порт' (Port) is '61763'. The 'Имя базы данных' (Database name) is 'Diplom\_db'. The 'Имя пользователя' (Username) is 'postgres'. The 'Пароль' (Password) is masked with dots. There is a checkbox for 'Использовать безопасное подключение (SSL)?' (Use secure connection (SSL)?). On the right side, there are two buttons: 'Произвести синхронизацию схемы' (Sync schema) and 'Пересканировать значения полей' (Rescan field values). Below these, there is a red box labeled 'Опасная зона' (Danger zone) containing two red buttons: 'Сбросить сохраненные значения полей' (Reset saved field values) and 'Удалить эту базу данных' (Delete this database).

Оригинал скрина:

Скриншоты\12. Настройка подключения MetaBase к БД.png

Все, можно создавать графики, таблицы, отчеты и строить дашборды.

Я создал два дашборда:

- 1) Данные по количеству гостей в разных разрезах;
- 2) Данные по суммам в разных разрезах.

На первом дашборде для получения данных я воспользовался пользовательскими запросами, то есть без написания sql кода (примеры чуть позже).

Плюсы первого варианта:



- 1) Не надо руками писать SQL запрос;
- 2) На графиках можно проваливаться в цифры для более подробного изучения данных;
- 3) Можно создавать сводные (pivot) таблицы.

Минус первого варианта:

- 1) Нельзя добавлять пользовательские фильтры.

На втором дашборде я получал данные непосредственно написанием sql кода. И здесь же, для примера, добавлена возможность фильтрации данных по периоду.

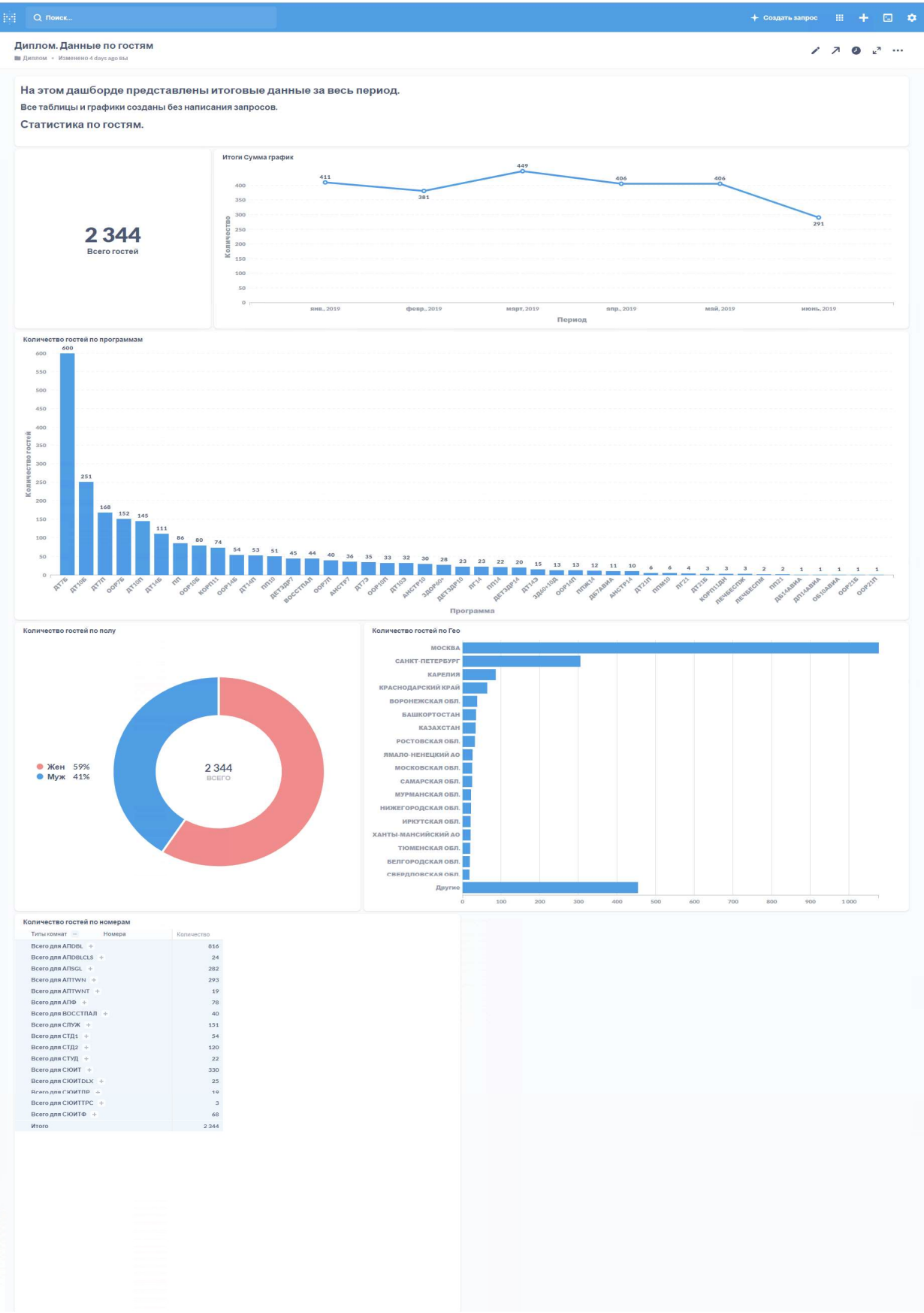
Итак у меня получились следующие дашборды:

Диплом			
Тип	Имя ^	Изменено	Изменено
	Диплом. Данные по гостям	Александр Тульев	February 21, 2022 ...
	Диплом. Данные по суммам (SQL)	Александр Тульев	February 21, 2022 ...

Еще раз, все оригиналы скринов находятся:

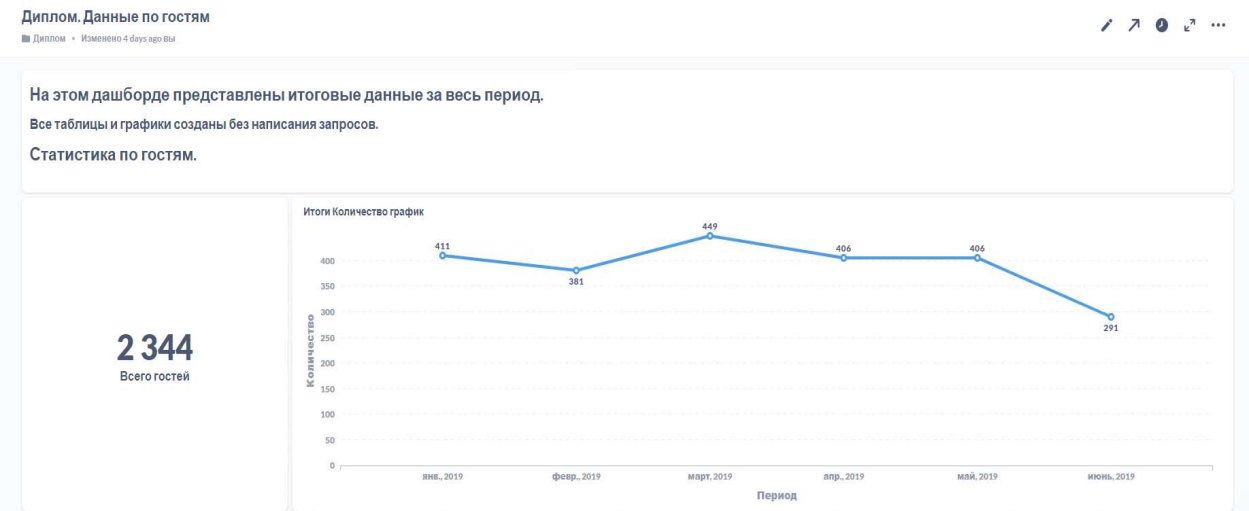
\\_Скрины\

Посмотрим на первый дашборд:

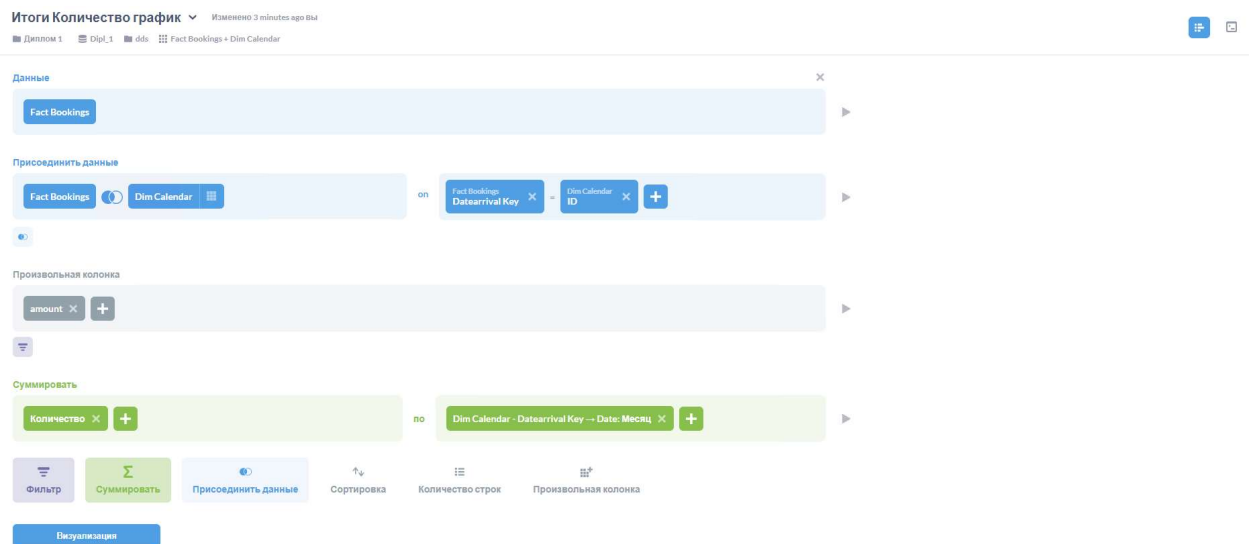




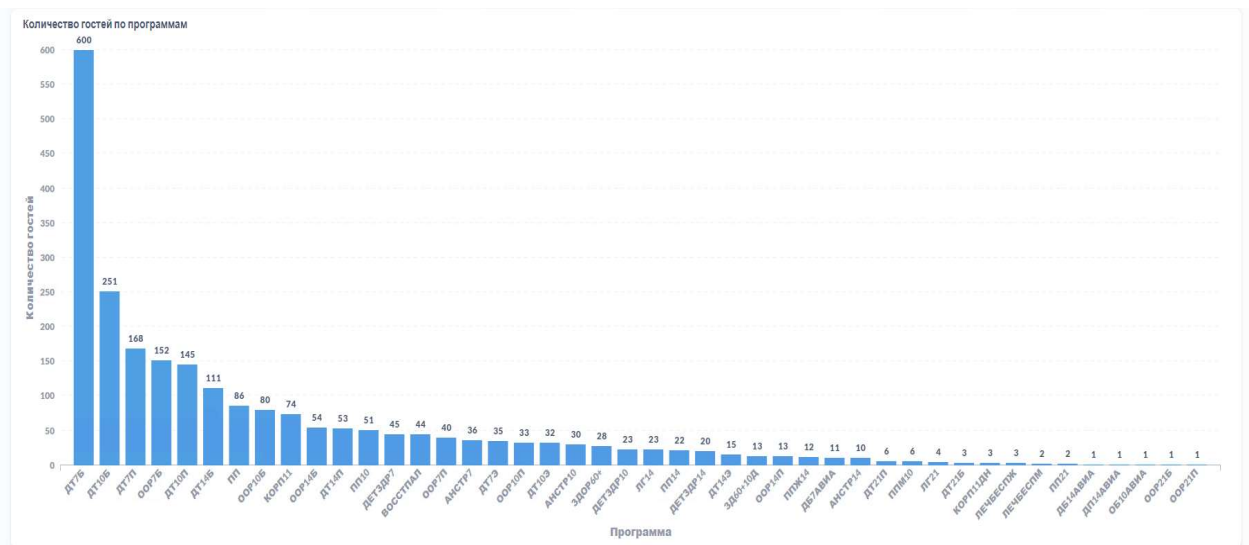
Приведу теперь скрины по отдельности для лучшей наглядности:



А вот так выглядит создание (подготовка):



Еще пример:



## Количество гостей по программам

Изменено 5 days ago Вы

Диплом 1 Dipl\_1 dds Fact Bookings + Dim Rate

Данные

Fact Bookings

Присоединить данные

Fact Bookings

on

Dim Rate

Fact Bookings

Rate Key

-

Dim Rate

ID

+

Суммировать

Количество

+

по

Dim Rate - Rate Key → Rate

+

Сортировка

↓ Количество

+

Фильтр

Суммировать

Присоединить данные

Количество строк

Произвольная колонка

Визуализация

А вот так выглядит настройка визуализации:

## Количество гостей по программам

Изменено 5 days ago Вы

Диплом 1 Dipl\_1 dds Fact Bookings + Dim Rate

Фильтр

Суммировать

Исходные данные

Гистограмма значений

Вид

Оси

Метки

Данные

Объединение

Не объединять

Объединить

Объединить - 100%

Линия достижения

Показывать значения на точках данных

Значения, чтобы показать

Столько, сколько может поместиться

Все

Форматирование значения

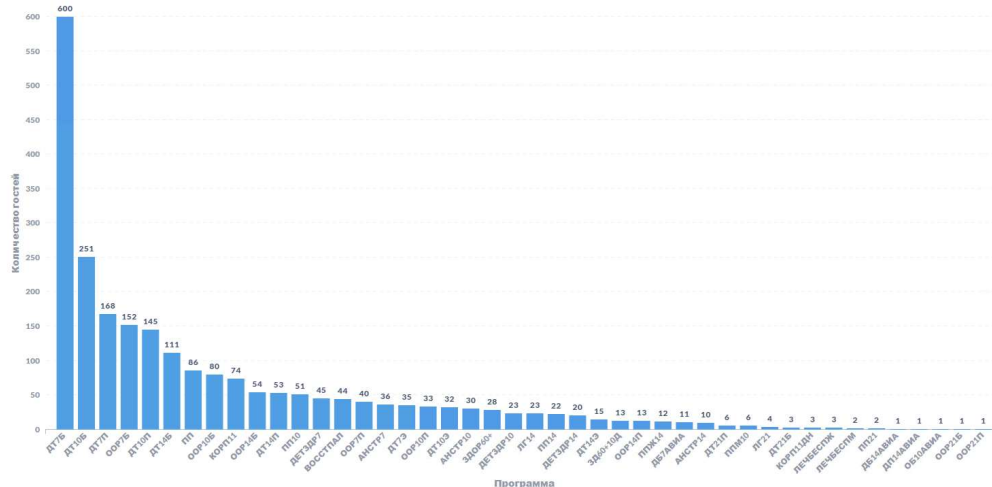
Авто

Компактно

Полностью

Какая ось?

Количество



Количество гостей по номерам		
Типы комнат	Номера	Количество
Всего для АПДВЛ +		816
Всего для АПДВЛСЛС +		24
Всего для АПСGL +		282
Всего для АПТWN +		293
Всего для АПТWNТ +		19
Всего для АПФ +		78
Всего для ВОССТПАЛ +		40
Всего для СЛУЖ +		151
Всего для СТД1 +		54
Всего для СТД2 +		120
Всего для СТУД +		22
Всего для СЮИТ +		330
Всего для СЮИТDLX +		25
Всего для СЮИТПР +		19
Всего для СЮИТПРС +		3
Всего для СЮИТФ +		68
Итого		2 344

Эту Pivot таблицу можно раскрывать:

Количество гостей по номерам		
Типы комнат	Номера	Количество
	351	22
	353	25
	604	25
	617	27
	654	26
	667	25
	814	30
Всего для АПТWN		293
Всего для АПТWNТ +		19
Всего для АПФ +		78
Всего для ВОССТПАЛ +		40
Всего для СЛУЖ +		151
Всего для СТД1 +		54
Всего для СТД2 +		120
Всего для СТУД +		22
СЮИТ	251	20
	350	37
	611	30
	661	30
	668	26
	701	25
	703	24
	704	25
	751	26
	753	22
	754	36
	859	29
Всего для СЮИТ		330
Всего для СЮИТDLX +		25
Всего для СЮИТПР +		19
Всего для СЮИТПРС +		3
Всего для СЮИТФ +		68
Итого		2 344

И любую из приведенных выше диаграмм и таблиц можно раскрывать для изучения данных:

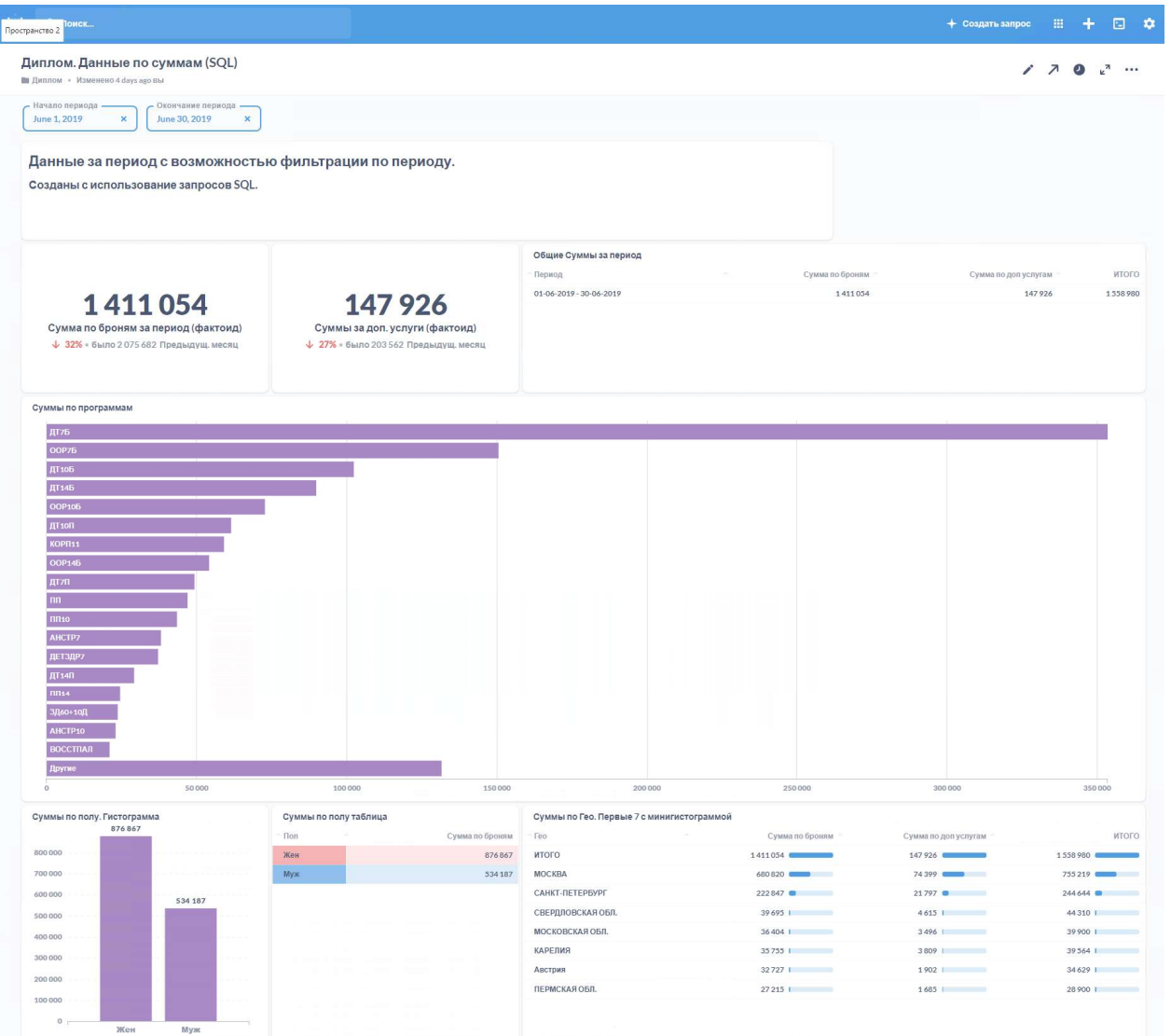
Dipl\_1 \* dds \* Fact Bookings + Dim Rate Начать с: Количество гостей по программам

Rate - АНСТР14 X

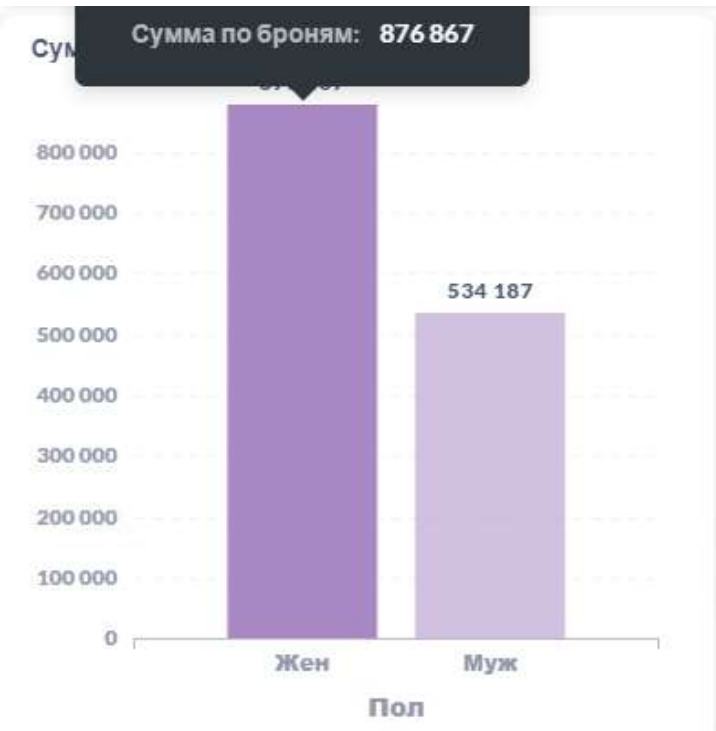
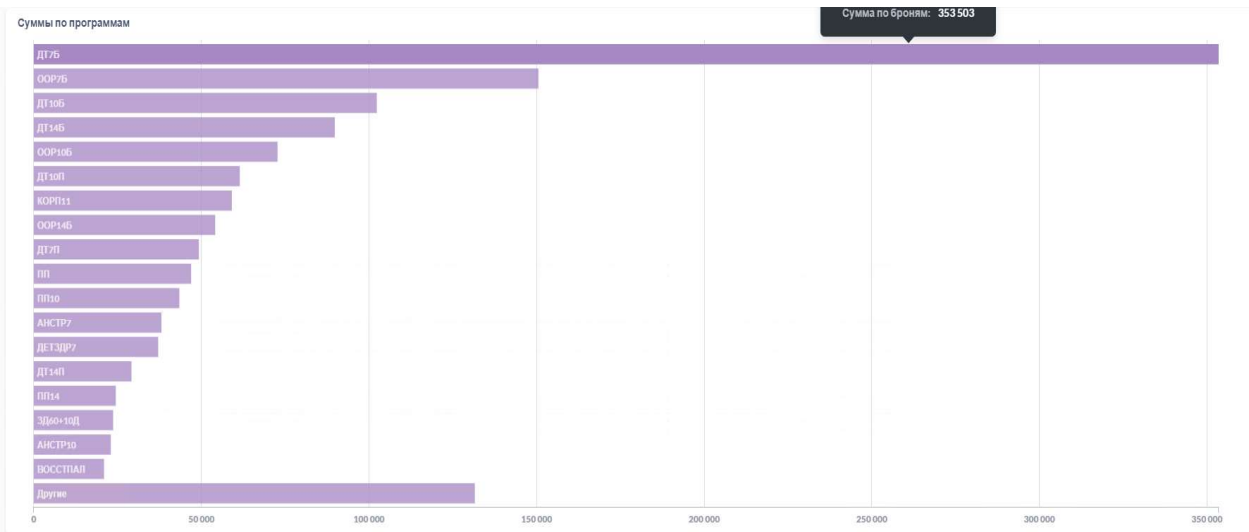
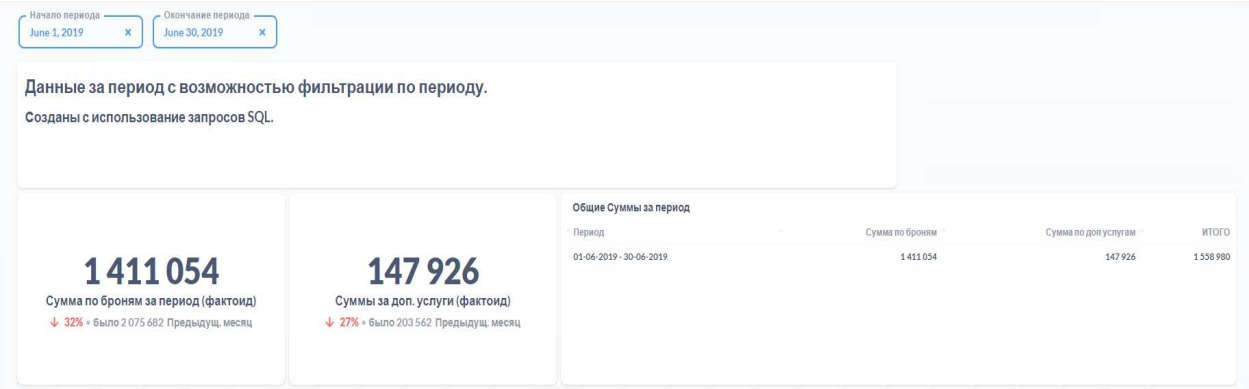
Сохранить Фильтр + Суммировать

Bookingnumber Key	Room Key	Sex Key	Datearrival Key	Datedeparture Key	Rate Key	Geo Key	Amount Booking	Amount Add Booking	Dim Rate - Rate Key - ID	Dim Rate - Rate Key - Rate
3190	29	2	20190407	20190420	12	6	4 742	620	12	АНСТР14
3046	40	2	20190324	20190406	12	13	3 763	698	12	АНСТР14
3335	2	1	20190414	20190427	12	50	6 963	731	12	АНСТР14
2063	56	2	20190317	20190329	12	47	8 207	150	12	АНСТР14
2220	25	2	20190303	20190316	12	33	7 974	733	12	АНСТР14
2815	85	1	20190414	20190427	12	38	9 631	629	12	АНСТР14
1963	100	1	20190120	20190202	12	83	8 031	323	12	АНСТР14
2503	46	1	20190317	20190330	12	69	7 709	586	12	АНСТР14
3577	77	2	20190526	20190608	12	83	9 529	167	12	АНСТР14
3868	95	2	20190616	20190629	12	50	2 538	453	12	АНСТР14

Теперь привиду скриншоты со второго дашборда:



Ну и чуть более крупно каждый:



**Суммы по полу таблица**

Пол	Сумма по броням
Жен	876 867
Муж	534 187

**Суммы по Гео. Первые 7 с минигистограммой**

Гео	Сумма по броням	Сумма по доп услугам	ИТОГО
ИТОГО	1 411 054	147 926	1 558 980
МОСКВА	680 820	74 399	755 219
САНКТ-ПЕТЕРБУРГ	222 847	21 797	244 644
СВЕРДЛОВСКАЯ ОБЛ.	39 695	4 615	44 310
МОСКОВСКАЯ ОБЛ.	36 404	3 496	39 900
КАРЕЛИЯ	35 755	3 809	39 564
Австрия	32 727	1 902	34 629
ПЕРМСКАЯ ОБЛ.	27 215	1 685	28 900

Все эти диаграммы и таблицы получены SQL кодом. Пример:

## Суммы по Гео. Первые 7 с минигистограммой ▾

Изменено 4 days ago Вы

■ Диплом 2    📄 Dipl\_1

Dipl\_1 ▾    Date1 June 1, 2019 ×    Date2 June 30, 2019 ×

```
SELECT
  -- geo as "Гео",
  CASE WHEN geo IS NULL THEN 'ИТОГО' ELSE geo END AS "Гео",
  sum(amount_booking) AS "Сумма по броням",
  sum(amount_add_booking) AS "Сумма по доп услугам",
  sum(amount_booking+amount_add_booking) AS "ИТОГО"
FROM
  dds.fact_bookings
LEFT JOIN dds.dim_calendar cc ON dds.fact_bookings.datearrival_key=cc.id
LEFT JOIN dds.dim_geo geo ON dds.fact_bookings.geo_key=geo.id
WHERE
  date_ >= {{date1}} AND date_ <= {{date2}}
GROUP BY
  ROLLUP(geo)
ORDER BY
  sum(amount_booking+amount_add_booking) DESC
LIMIT 8
```

Гео ▾	Сумма по броням	Сумма по доп услугам	ИТОГО
ИТОГО	1411054	147926	1558980
МОСКВА	680820	74399	755219
САНКТ-ПЕТЕРБУРГ	222847	21797	244644
СВЕРДЛОВСКАЯ ОБЛ.	39695	4615	44310
МОСКОВСКАЯ ОБЛ.	36404	3496	39900
КАРЕЛИЯ	35755	3809	39564
Австрия	32727	1902	34629
ПЕРМСКАЯ ОБЛ.	27215	1685	28900

📊 Визуализация

⚙️ Настройки

И последнее, что я сделал, это настроил отправку дашбордов на электронную почту по расписанию:

☒ Отправить эту панель по электронной почте

Примечание: диаграммы в вашей подписке не будут выглядеть так же, как на дашборде [Узнать больше](#).

Кому:

Александр Тульев ✕

Отправлено Ежедневный ▾

в 10:00 ▾ AM PM

Электронная почта будет отправлена в 10:00 AM GMT, часовой пояс вашего Metabase.

Отправить Email прямо сейчас

Значения фильтра ⓘ

Если для фильтра дашборда задано значение по умолчанию, оно будет применено при отправке подписки.

Не отправлять, если нет результатов ☐

Прикрепите результаты ⓘ



Формат файла

.csv

.xlsx

☒ Вопросы для прикрепления

☒ Всего гостей

☒ Итоги Количество график

☒ Количество гостей по программам

☒ Количество гостей по полу

☒ Количество гостей по Гео

☒ Количество гостей по номерам

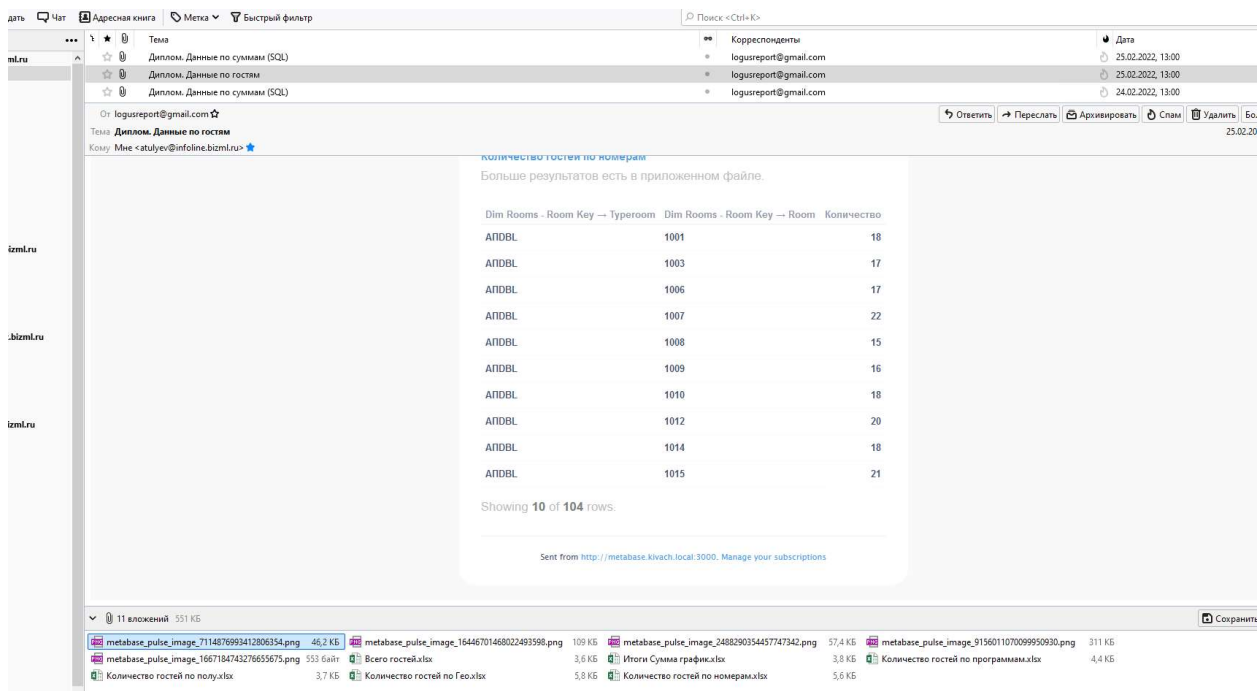
Удалить эту подписку

Отмена

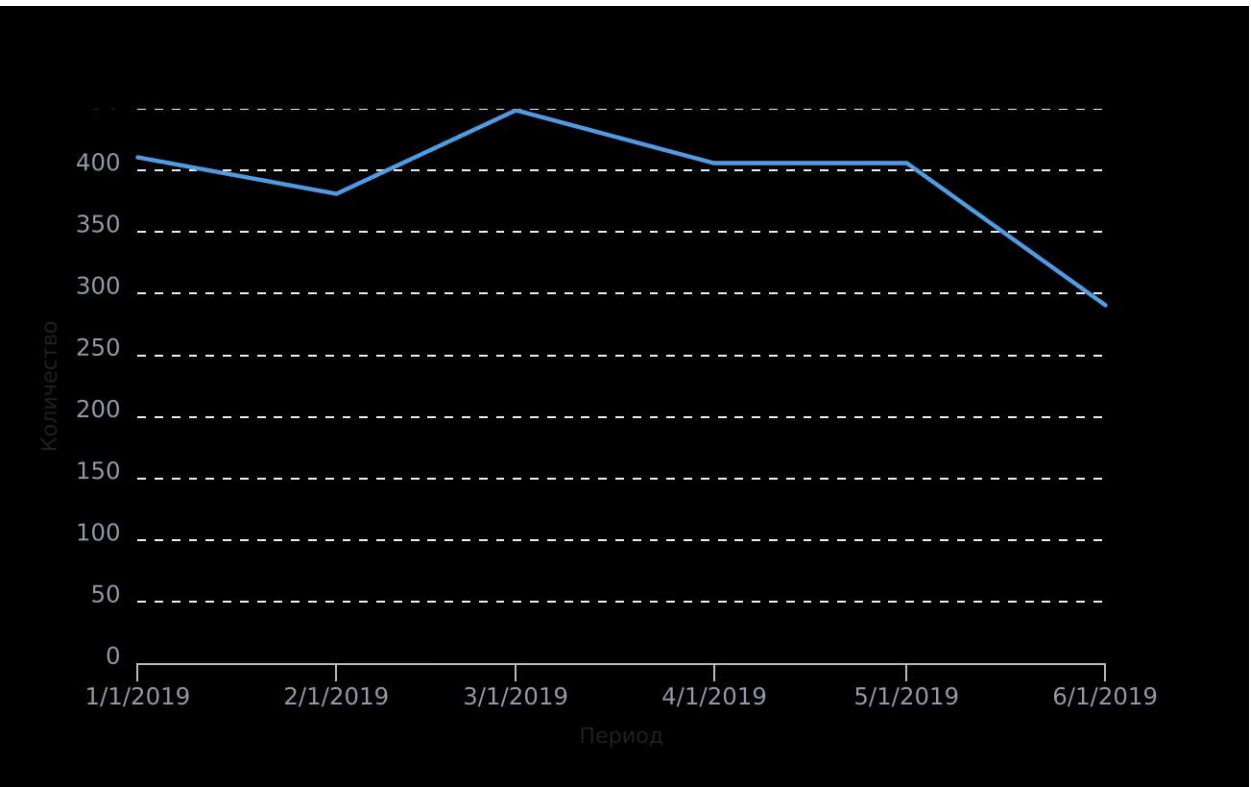
Готово



Пример полученного письма с данными и вложениями:



Пример диаграммы из письма:



Данные получены и визуализированны.

Более подробно о MetaBase в рамках данной работы не останавливаюсь.

## **8. Выводы.**

**Бизнес-задачи, которые можно решить, используя полученное хранилище данных и визуализации на дашбордах.**

Накопленные в БД массивы данных позволят бизнесу получать на основе истории следующую структурированную информацию и аналитику и в дальнейшем их применять:

Данные о степени заполнения клиники и получаемой выручке в различных разрезах (продаваемые программы, сезонность, пол, гео, номера...).

Анализ этих данных позволит принимать решения о выгодности продажи программ, номеров гостям в зависимости от разных показателей (например, сезонность).

**Естественно в этом проекте (работе) есть что дорабатывать:**

1. В базе источнике есть много параметров, которые я не переносил или не сделал визуализацию на основе перенесенных данных:

- ) транзакции (деление сумм по услугам);
- ) оказываемые медицинские услуги;
- ) занятость медицинских кабинетов и медицинского персонала;
- ) оказываемые услуги и продаваемые товары;
- ) занятость обслуживающего персонала;
- ) информация о первичниках, повторниках;
- ) информация о вернувшихся и потерянных клиентах;
- ) информация по номеру заезда гостей и по переходу с программы на программу;
- ) информация о возрасте гостей;
- ) деление гостей на взрослых и детей;
- ) информация о сроке пребывания в клинике;
- ) информация о корпусах пребывания гостей;
- ) информация о среднем чеке и медианах;

- ) занятость номерного фонда по дням;
- ) информация о программе питания для гостей;
- ) информация о скидках и бонусах;
- ) ...

Этот список можно продолжать и продолжать...

После доработки хранилища, ETL процессов и визуализации анализ новых данных позволит :

- ) построить рекомендательную систему, которая будет "распознавать" клиента, предлагать ему путевку, услуги, товары, мед. услуги и опции "похожие" на те, что он уже приобретал ранее;

- ) "нарисовать" портрет типового и "выгодного" клиентов (определить несколько параметров, описывающих их), а далее проводить определенные маркетинговые акции именно тем клиентам, которые их с той или иной степенью вероятности приобретут (в конечном итоге, решить задачу повышения персонализации коммуникаций, то есть сформировать рекомендации в соответствии с уникальными предпочтениями, потребностями и желаниями клиента в целях увеличения продаж и прибыли).

- ) предсказывать сезонный спрос (может быть, применяя алгоритмы машинного обучения) и далее планировать загрузку, плановые ремонты, ТО оборудования, отпуска персонала...;

2. Естественно необходимо улучшить (увеличить количество) проверку качества данных (Data Quality);

3. Также еще необходимо улучшить проверку качества данных в следующем направлении: сейчас у меня проверка реализована таким образом, если в одной строке будет несколько ошибок, то в таблицу rejected. rejected\_table попадет описание только первой ошибки, а остальные проигнорируются. Это желательно переделать.

4. Я не использовал Airflow (платформа для создания, мониторинга и оркестрации pipelines);

5. Также можно сделать отдельный слой метаданных в хранилище, а также дашборды на основании данных из этого слоя, где будет отображаться количество прогрузок и их статусы.

Но в этой работе данные из первоисточника и во все слои грузились только один раз (то есть не подгружал данные и не грузил дельту), и поэтому я посчитал, что данный слой из одной записи будет совсем не показателен и создавать и загружать этот слой не стал.

6. Также можно написать скрипт запуска основного задания из консоли (kitchen.bat) и запускать это задание уже из консоли (или удаленно на Carte сервере);

7. Я не воспользовался в задании и трансформациях переменными. Лучше пользоваться переменными, чтоб улучшить и облегчить работу и дальнейшее сопровождение.

### **Теперь немного информации о моем рабочем проекте.**

Визуализацией и анализом данных я занимаюсь уже примерно 1,5 года.

Использую MetaBase. Почему именно MetaBase?

1. Потому что эту систему предложили программисты наших двух основных БД и программ, которыми мы пользуемся. Они работают именно с MetaBase. И первое время помогали мне разобраться с этой системой;
2. Это open-source проект, который очень активно развивается и не плохо поддерживается. Также существует и коммерческий проект.

### **Основные сложности с которыми я сталкиваюсь:**

1. Нет описания структур источников БД;
2. Много вопросов о подзагрузке в БД дельты (на курсе об этом совсем не было информации);
3. Мало настоящих специалистов-аналитиков на фирме;
4. Часто задания от пользователей мне по разработке визуализации превращают систему чисто в отчетную, а не аналитическую. Новых отчетов от программистов баз источников практически не дожждаться. Вот и приходится делать их в MetaBase. Но с другой стороны очень трудно разделить отчетную и аналитическую системы.

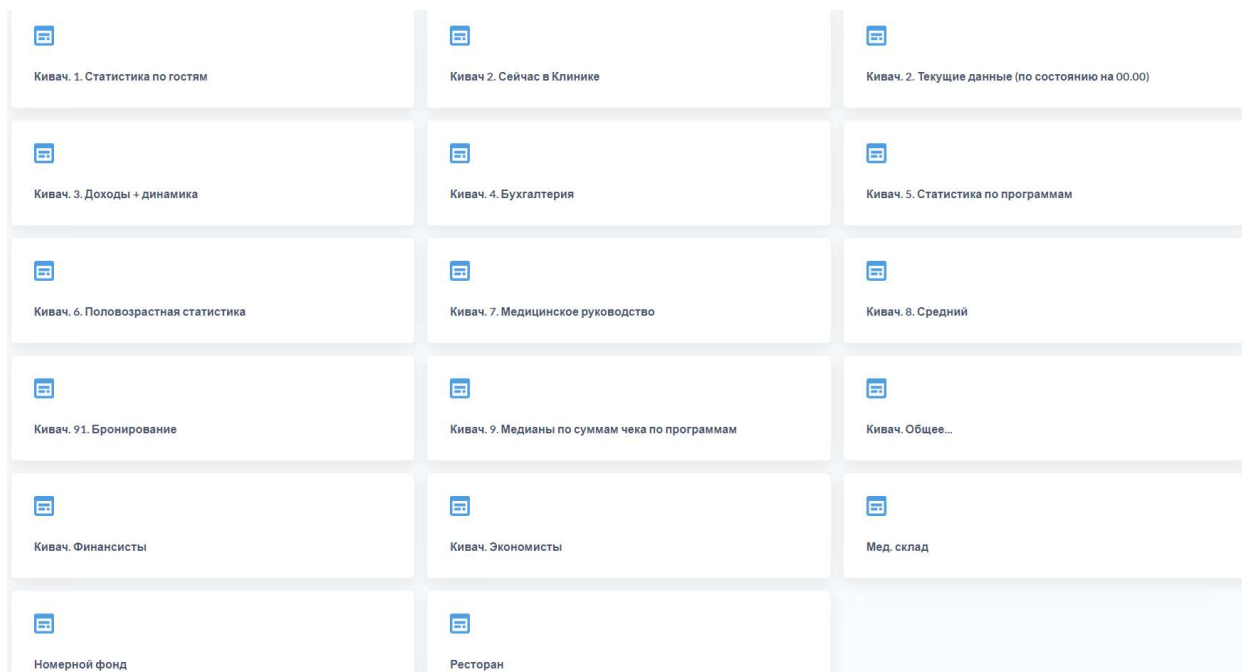
### **В данное время у нас два хранилища (оба на базе MS SQL):**

1. Хранилище, созданное и заполненное разработчиками двух основных программ и БД, которыми мы пользуемся, - это покупное ПО, разработанное не нашими программистами.

Это хранилище не позволяет решить все возникающие вопросы, которые нам бы хотелось. Описание этого ХД есть, а вот описания ETL процессов нет совсем. Насколько мне удалось выяснить, они пользуются программой Datarumper;

2. Хранилище, которое разрабатываю я. На данном этапе я пока использую только скрипты MS SQL.

На данный момент уже реализовано порядка 18 дашбордов и около 150 запросов.



Подробные данные не привожу.

Также настроены разрешения (ограничения) по доступу к данным и визуализациям.

Проект активно развивается и пользуется спросом.

## **9. Используемое ПО**

- а) Microsoft SQL Server Management Studio 17;
- б) SQL Server 14.0.2037.2;
- в) Dbeaver ver 21.3.3;
- г) Pentaho Data Integration ver 9.2;
- д) Jupyter Notebook (Anaconda3) ver 6.3.0;
- е) Python ver 3.8.8;
- ж) Drawio ver 13.9.9;
- з) Docker Desktop ver 4.4.4;
- и) Kitematic 0.17.13;
- к) MetaBase ver 0.41.5;
- л) PostgreSQL ver 14.1;
- м) GitHub.