**Анализ входных данных**

Для того, чтобы понять, какую витрину нам нужно построить и какие цели она должна преследовать, необходимо изучить состав исходных данных, понять, что из этого состава может послужить для «промежуточной обработки».

Мы имеем данные с Kaggle в csv формате [по кадровой аналитике специалистов по данным](https://www.kaggle.com/arashnic/hr-analytics-job-change-of-data-scientists).

3 файла: 2 из них, это некий набор личных данных по специалистам (aug\_test - №1, aug\_train - №2), 1 – это отметки о возможности смены работы (sample\_submission - №3).

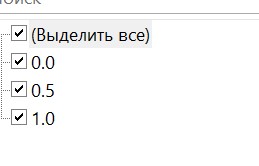
Основная задача – это понять работодателю, кто из прошедших курсы действительно хочет работать у него.

Так как нашей главной целью, как Data Engineer, всё же является не сам анализ, а эффективное построение витрины, на основе которой возможно будет сделать какие-то дополнительные агрегации для выводов, то попробуем обеспечить эту визуализацию с помощью Vertica.

**Метаданные**

Так, как данные располагаются в разных файлах, а нам необходимо видеть общую картину, то объединяем их в один агрегат с помощью Power Query (этот инструмент помогает сконвертировать данные из csv в колоночную форму, а также сделать некоторые соединения и преобразования для удобства). Этот инструмент часто удобен для быстрого первичного анализа данных, а также позволяет увидеть пустоты, несоответствия и др. ошибки.

Сразу видим, что №1 и №2 отличаются наличием поля с целью в №2 и отсутствием его в №1. Но у нас есть №3, и присоединяем его к №1. Все присоединилось. Пустот по этим данным уже не наблюдается. Формат единый.



Теперь есть полный набор метаданных:

|  |  |
| --- | --- |
| enrollee\_id | id сотрудника |
| city | Город |
| city\_development\_index | Индекс развития города (в масштабе) |
| company\_size | Количество сотрудников в компании текущего работодателя |
| company\_type | Тип текущего работодателя |
| education\_level | Уровень образования кандидата |
| enrolled\_university | Тип зачисленных университетских курсов, если таковые имеются. |
| experience | Кандидатский общий стаж в годах |
| gender | Пол |
| last\_new\_job | разница в годах между предыдущей работой и текущей работой |
| major\_discipline | Обучение основной дисциплине кандидата |
| relevent\_experience | Соответствующий опыт кандидата |
| training\_hours | завершенные часы обучения |
| full target | цель: 0 - не ищу смены работы/ 1 - ищу смену работы |

На всякий случай проверяем наличие ошибок и дубликатов по другим колонкам.

Строк с полным отсутствием информации нет, однако все равно имеется много пропусков.

**Вторичные данные**

Нам нужна некая система оценок по критериям. Предположим, что сотрудники бизнес-сектора предоставили нам такую информацию (шкалы).

|  |  |  |  |
| --- | --- | --- | --- |
| Исходные данные | | | Информация  менеджмента и  аналитиков |
| *enrollee\_id* | *id сотрудника* | *Шкала оценки* | |
| city | Город | 7 | |
| city\_development\_index | Индекс развития города (в масштабе) | 7 | |
| company\_size | Количество сотрудников в компании текущего работодателя | 8 | |
| company\_type | Тип текущего работодателя | 6 | |
| education\_level | Уровень образования кандидата | 4 | |
| enrolled\_university | Тип зачисленных университетских курсов, если таковые имеются. | 2 | |
| experience | Кандидатский общий стаж в годах | 21 | |
| gender | Пол | 1 | |
| last\_new\_job | разница в годах между предыдущей работой и текущей работой | 4 | |
| major\_discipline | Обучение основной дисциплине кандидата | 5 | |
| relevent\_experience | Соответствующий опыт кандидата | 1 | |
| training\_hours | завершенные часы обучения | 4 | |
| full target | цель: не ищу смены работы/ищу смену работы | 1 | |

\*в некоторых случаях с null предполагалась нулевая оценка (потом это было исключено в процессе проработки запросов)

Сотрудники, набравшие наибольшее количество баллов, попадают в топ 10.

Система оценок по параметрам представлена в папке «Grades». Здесь хранятся дополнительно созданные таблицы по каждой характеристике.

Наименования столбцов удалены, чтобы упростить работу с форматами данных (особенно если они числовые).

**Запуск Docker и подготовка данных в Vertica**

Подготовим репозиторий для работы:

<https://github.com/PytUs/DWH-Vertica>

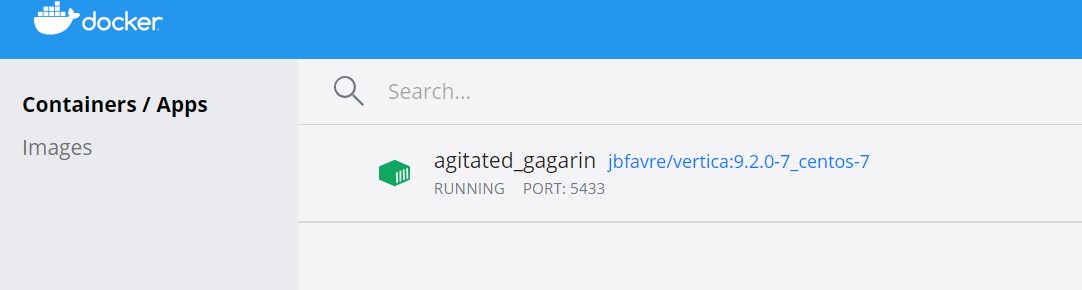
Clone и перейдем в него через cmd Windows 10. Создадим папку data и положим туда исходные и вторичные данные.

Запускаем Docker:

docker pull jbfavre/vertica:9.2.0-7\_centos-7

DWH-Vertica\data>docker run -p 5433:5433 -d -v …\DWH-Vertica\data jbfavre/vertica:9.2.0-7\_centos-7

Docker запущен:

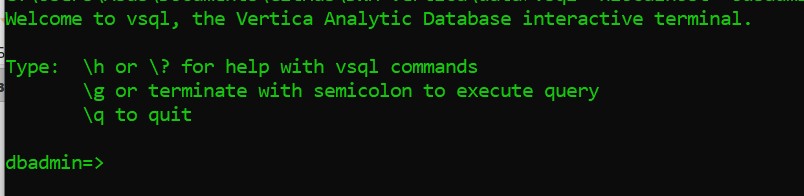


Вводим:

chcp 65001

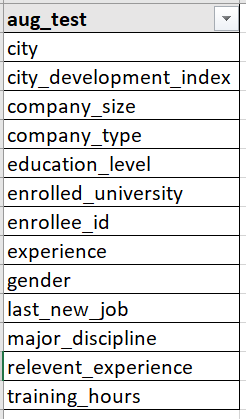
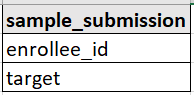
vsql -hlocalhost -Udbadmin

Welcome to vsql Vertica:

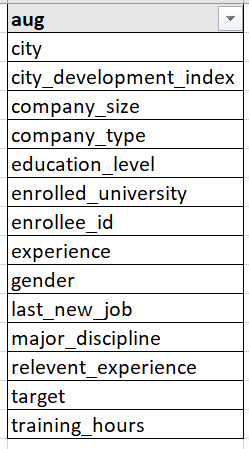


**Схема данных (последовательность создания)**

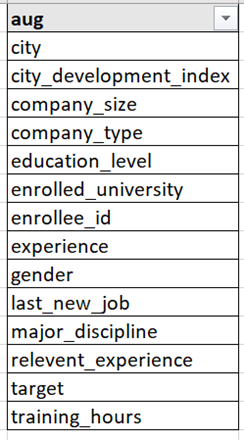
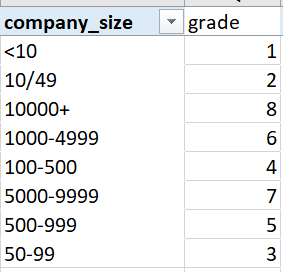
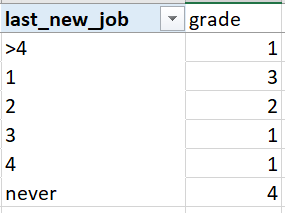
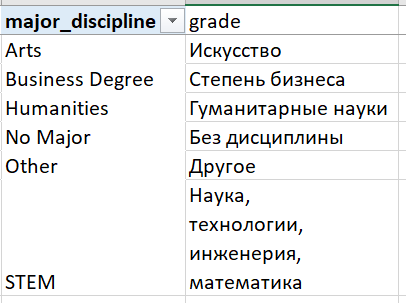
1. Присоединим недостающие строки в таблицу №1 (target) по id:



1. Так как во второй таблице target имеется, то просто соединяем таблицы №1 и №2 в одну aug и получаем следующую схему



1. Нужно исходные данные соединить с вторичными (с оценками по каждому параметру). В качестве примера покажем неполный набор, т.к. по остальным параметрам соединение аналогичное:



**Запросы и результаты**

После первичного анализа входных данных и запуска ресурсов для их обработки, разделим систему DDL и DML sql запросов на части.

*Подготовка данных и создание для них таблиц, вью, схем с более подробным описанием внутри:*

1\_SCHEMA

4\_creating\_full\_data\_table

6\_not\_null

*Импорт данных из папки:*

2\_creating\_original\_tables

7\_copy\_sec\_tables

*Слияния, контроль дублей, проекция, итоги (топ 10):*

8\_joins\_proj\_orders\_totals.

Результат:

