Команда № 8 GlowByte Autumn Hack 2022

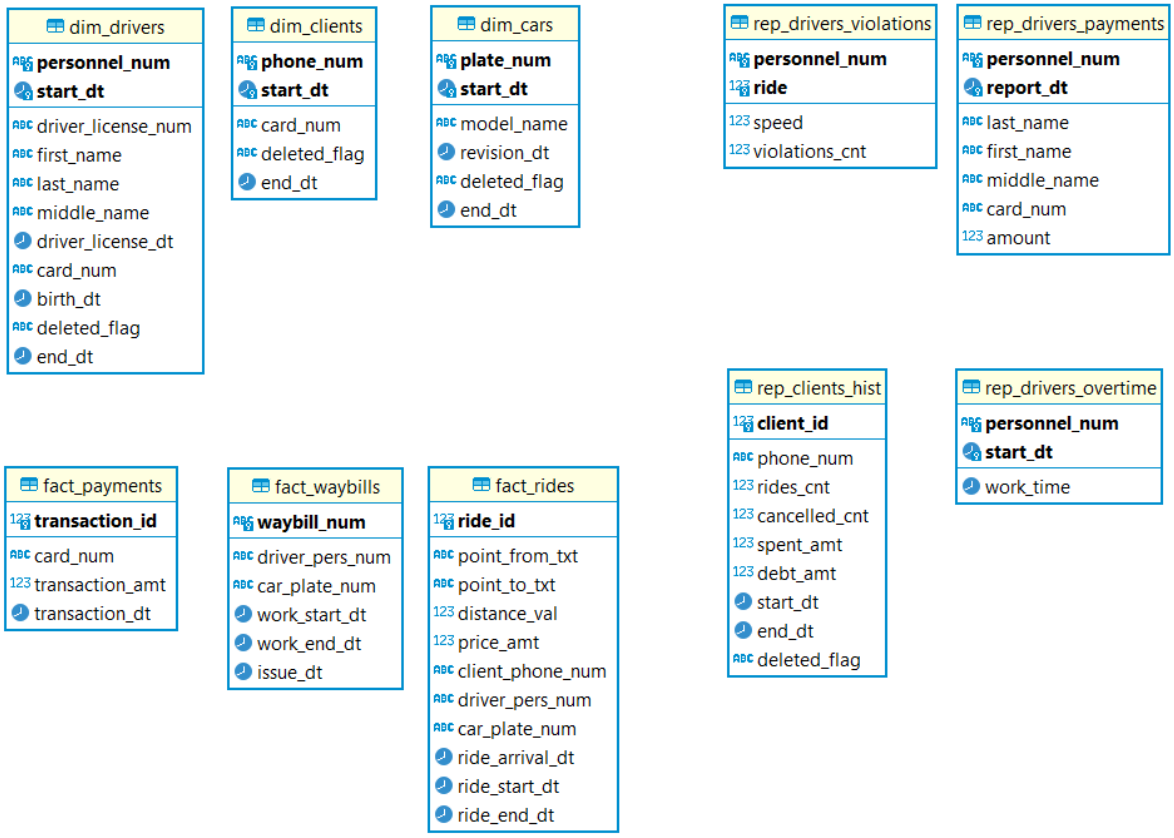
В решении принимали участие:

Морозов Виктор, программирование, python и SQL, координация

Решенин Денис, программирование, python и SQL

Альбина Садыкова – первичная подготовка схемы данных

Схема данных:



Описание сущностей на схеме в приложении: «Описание сущностей.pdf»

1. Запуск обработок осуществляется через *all_report_start.py*
2. Пароли хранятся в переменных окружения, в случае проблем с доступом к переменным окружения предусмотрен ручной ввод паролей.
3. Для всех соединений добавлено {ssl : require}

Заполнение FACT_

Описание реализации заполнения fact_payments:

1. Запрашиваем из fact_payments данные необходимые для определения, какие файлы мы еще не загрузили
2. Загружаем новые файлы на локальный диск
3. Производим их обработку (создаем pd.DataFrame)
4. Удаляем загруженные файлы
5. Загружаем полученный pd.DataFrame в БД

Описание реализации заполнения fact_waybills:

1. Запрашиваем из fact_waybills данные необходимые для определения, какие файлы мы еще не загрузили
2. Загружаем новые файлы на локальный диск
3. Производим их обработку (создаем pd.DataFrame)
4. Удаляем загруженные файлы
5. Загружаем полученный pd.DataFrame в БД

Описание реализации заполнения fact_rides:

1. Из базы fact_rides получаем самую последнюю по значению MAX(ride_end_dt)
2. Используя максимальное значение MAX(ride_end_dt) формируем запрос к таблице (JOIN таблиц rides и movement). Если MAX(ride_end_dt) пустой – тогда запрос без ограничения по времени.
3. Делаем и дополнительные столбцы со смещением LEAD(1) и LEAD(2) относительно столбцов dt и event). Тем самым получаем время прибытия, старта и завершения работы, а также флаги ARRIVEL, BEGIN, END, CANCEL в одной строке.
4. Отбираем только интересующие нас столбцы
5. Обрабатываем полученные данные и загружаем их в целевую таблицу

Заполнение DIM_

Описание реализации заполнения dim_drivers:

1. Из базы dim_drivers получаем последнею дату start_dt
2. Делаем запрос данных к соответствующему источнику. Отбираем все строки update_dt которых больше start_dt. Если start_dt – пуст, отбираем все строки
3. Обрабатываем полученные данные и загружаем их в целевую таблицу
4. Для всех строк с deleted_flag = ‘N’ обновляем значение end_dt, используя оконную функцию LEAD (start_dt)
5. Если end_dt больше не равно ‘2222-12-31 23:59:59.000’, то меняем его значение на ‘Y’

Описание реализации заполнения dim_clients:

1. Из базы dim_clients получаем последнею дату start_dt
2. Делаем запрос данных к rides. Отбираем все строки dt которых больше start_dt. Если start_dt – пуст, отбираем все строки
3. Отбираем только интересующие нас столбцы
4. Обрабатываем полученные данные и загружаем их в целевую таблицу
5. Для всех строк с deleted_flag = ‘N’ обновляем значение end_dt, используя оконную функцию LEAD (start_dt)
6. Если end_dt больше не равно ‘2222-12-31 23:59:59.000’, то меняем его значение на ‘Y’

Описание реализации заполнения dim_cars:

1. Из базы dim_cars получаем последнею дату start_dt
2. Делаем запрос данных к соответствующему источнику. Отбираем все строки update_dt которых больше start_dt. Если start_dt – пуст, отбираем все строки
3. Обрабатываем полученные данные и загружаем их в целевую таблицу
4. Для всех строк обновляем значение end_dt, используя оконную функцию LEAD (start_dt)

Заполнение REP_

Описание реализации заполнения rep_driver_payments:

1. Из rep_driver_payments запрашиваем последнюю дату отчета
2. Запрашиваем агрегированные данные из fact_rides с разбивкой по дням и актуальный список водителей из dim_drivers. Объединяем таблицы по учетному номеру водителя
3. По правилам формируем новый столбец с выплатой водителю за текущий день. Данные за неполный день исключаем из загрузки.
4. Формируем запрос на добавление данных в rep_driver_payments и обновляем rep_driver_payments.

Описание реализации заполнения rep_driver_overtime:

1. Запрашиваем из rep_driver_overtime дату последнего записанного нарушения.
2. Используя эту дату, формируем запрос к fact_waybill. Получаем актуальные необработанные путевые листы.
3. Для каждого путевого листа определяем начало и конец отчетного периода для построения отчета по переработкам.
4. Готовим отчет по нарушителям. Путевые листы для которых сутки еще не закончились исключаем из отчета.
5. Записываем отчет в rep_driver_overtime.

Описание реализации заполнения rep_driver_violations:

1. Из rep_driver_violations запрашиваем максимальный номер поездки
2. Формируем отчет по водителям используя расстояние и разницу между началом и концом поездки для вычисления средней скорости. Отмененные и поезди с номером менее максимального из предыдущего запроса исключаем из отчета.
3. Записываем данные в rep_driver_violations
4. Обновляем количество нарушений для каждой незаполненной строки в столбце violatons_cnt

Описание реализации заполнения rep_clients_hist:

Отчет формируется каждый раз заново с помощью SQL запросов.
Описание запроса:

1. Чистим данные.

2. Изначально объединяем dim_clients и fact_payments (inner join) при объединении учитываем время свершения транзакции и номер карты
3. Получаем 3 таблицы:
    a. Все успешно оплаченные поездки. fact_rides объединяем с предыдущей таблицей (inner join) по полям fr.price_amt = c.transaction_amt , fr.client_phone_num = c.phone_num, fr.ride_arrival_dt < c.transaction_dt
    b. Все отменённые поездки
    c. Все не оплаченные поездки. fact_rides объединяем с предыдущей таблицей (right join) по полям fr.price_amt = c.transaction_amt , fr.client_phone_num = c.phone_num, fr.ride_arrival_dt < c.transaction_dt при условии, что where transaction_dt is null

Через union объединяем их

4. Готовим технические столбцы необходимые нам для интересующих нас подсчетов и столбцы dt_interval и id_row, которые помогут нам выявить дубли (id_row больше 1 только при одинаковых номерах транзакций)
5. Формируем итоговую таблицу обрабатываем дубликаты. Добавляем столбец clien_id (нумеруем все данные отсортировав по дате, при повторных заполнениях нумерация останется постоянной)
6. Записываем данные в БД