Описание алгоритма процесса (на минимальный уровень).

Данный алгоритм позволит вам свести воедино все материалы курса и подготовить цельный проект, правда, на минимальный балл. Дальнейшие улучшения вы можете делать самостоятельно, применяя знания, полученные на курсе.

Если вы не отладили процесс написания кода у себя на компьютере и загрузки его на сервер – рекомендуем вам писать код сразу на сервере в nano.

В работе активно используйте заготовки, полученные на курсе – файл с загрузкой/выгрузкой данных из/в Excel (`snippet_pg.py`), файл с кодом инкрементальной загрузки (`SCD1_incremental_load.sql`).

Создайте на сервере нужные каталоги:

- `~/xxxx/project`
- `~/xxxx/project/archive`

и файлы:

- `~/xxxx/project/main.py`
- `~/xxxx/project/main.ddl`
- `~/xxxx/project/main.cron`

Не забудьте файл `main.py` сделать исполняемым. Загрузите в `~/xxxx/project` файлы задания. Заполните файл `main.ddl`. В нем вы создадите таблицы в базе `edu` (внимательно следуйте ТЗ!):

- `xxxx_stg_transactions`
- `xxxx_stg_terminals`
- `xxxx_stg_blacklist`
- `xxxx_stg_cards`
- `xxxx_stg_accounts`
- `xxxx_stg_clients`
- `xxxx_dwh_fact_transactions`
- `xxxx_dwh_fact_passport_blacklist`
- `xxxx_dwh_dim_terminals`
- `xxxx_dwh_dim_cards`
- `xxxx_dwh_dim_accounts`
- `xxxx_dwh_dim_clients`
- `xxxx_rep_fraud`

Алгоритм для файла `main.py`:

- Подключитесь к базе `bank`.
- Подключитесь к базе `edu`.
- Очистите весь стейджинг.

- Загрузите файл `transactions_01032021.txt` в стейджинг (используйте код из `snippet_pg.py`). Для простейшего варианта решения допустимо использовать имя файла «хардкодом», то есть записать в код как есть.
- Загрузите файл `terminals_01032021.xlsx` в стейджинг аналогично предыдущему пункту.
- Загрузите файл `passport_blacklist_01032021.xlsx` в стейджинг аналогично предыдущему пункту.
- Загрузите таблицу `bank.clients` в стейджинг. Используйте следующий подход:
    - Выполните запрос к базе `bank`
    - Сохраните полученный результат в DataFrame
    - Загрузите DataFrame в базу `edu`
- Загрузите таблицу `bank.accounts` в стейджинг. Используйте код из предыдущего пункта.
- Загрузите таблицу `bank.cards` в стейджинг. Используйте код из предыдущего пункта.
- Загрузите данные из стейджинга в целевую таблицу `xxxx_dwh_dim_terminals` (используйте код из `SCD1_incremental_load.sql`). Для простейшего случая можно пропустить шаги инкрементальной загрузки, удаления и управления метаданными.
- Загрузите данные из стейджинга в целевую таблицу `xxxx_dwh_dim_cards`. Используйте код из предыдущего пункта.
- Загрузите данные из стейджинга в целевую таблицу `xxxx_dwh_dim_accounts`. Используйте код из предыдущего пункта.
- Загрузите данные из стейджинга в целевую таблицу `xxxx_dwh_dim_clients`. Используйте код из предыдущего пункта.
- Загрузите данные из стейджинга в целевую таблицу `xxxx_dwh_fact_passport_blacklist`. Напоминаем, что в фактовые таблицы данные перекладываются «простым инсертом», то есть необходимо выполнить один `INSERT INTO … SELECT …`
- Загрузите данные из стейджинга в целевую таблицу `xxxx_dwh_fact_transactions`. Используйте код из предыдущего пункта.
- Напишите скрипт, соединяющий нужные таблицы для поиска операций, совершенных при недействующем договоре (это самый простой случай мошенничества). Отладьте ваш скрипт для одной даты в DBeaver, он должен выдавать результат. В простейшем варианте допустимо использовать «хардкод» для задания дня отчета.
- Результат выполнения скрипта загружайте в таблицу `xxxx_rep_fraud`. Не забывайте сформировать поле `report_dt`.
- Зафиксируйте изменения. Отключитесь от баз.

- Переименуйте обработанные файлы и перенесите их в другой каталог. Используйте для этого следующую заготовку в python:

```
import os
os.rename('/path1/file1.txt', '/path2/file2.txt')
```

Все, на этом `main.py` завершен. Вам нужно отладить его работоспособность на всех трех днях загрузки.

Заполните файл `main.cron` расписанием и командой исполнения вашего скрипта. Расписание установите так, как считаете нужным чтобы ваши данные заполнились корректно.

После этого можно сдавать проект.