

ЛИСИЧКИН ПАВЕЛ ВЛАДИМИРОВИЧ

# Реализация хранилища данных на основе Greenplum

Школа  
Data Engineer

Февраль 2022

ОБО МНЕ

Лисичкин  
Павел Владимирович

ТЕЛЕФОННЫЙ НОМЕР	—	+7 (915) 426-47-70
ЭЛЕКТРОННАЯ ПОЧТА	—	pavel.lisichkin@gmail.com
ГОРОД ПРОЖИВАНИЯ	—	Московская область, г. Монино

Образование высшее  
МГТУ "МАМИ"  
инженер по специальности  
"Управление и информатика  
в технических системах"  
2010 г.

Текущая должность:  
Главный Специалист группы  
анализа состояния бортовых  
систем космических аппаратов  
АО "Газпром  
Космические системы"

В этом проекте я разработал хранилище данных на основе MPP СУБД Greenplum.

Я разработал ETL-процесс, который включает в себя:

- выгрузку данных из внешнего источника
- обработку данных
- отслеживание актуальности данных посредством реализации механизма Slowly Changing Dimensions type2
- формирование витрины данных.

Также был реализован механизм запуска ETL-процесса по расписанию.

# ИСПОЛЬЗУЕМЫЙ СТЕК

Greenplum 

Psycopg2 

Cron 

Оконные функции

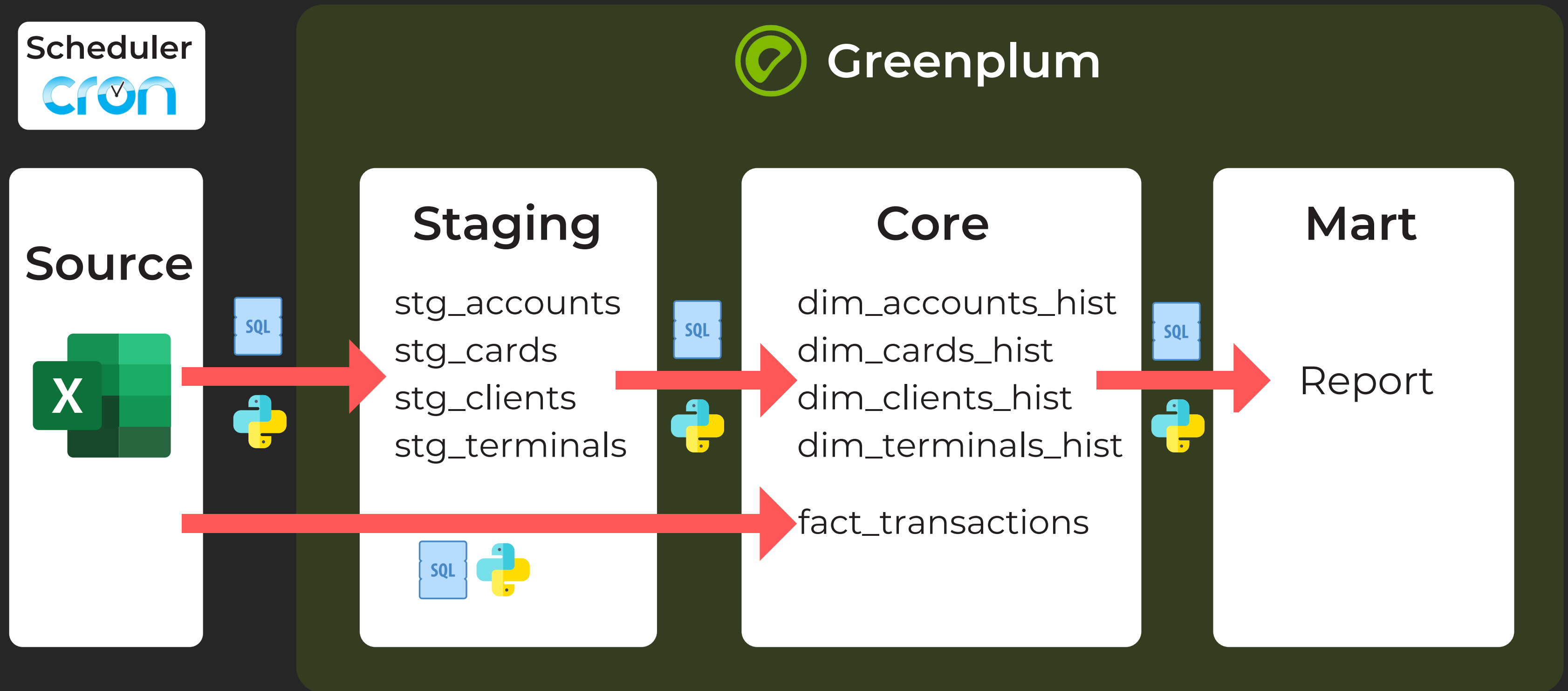
Python 

RegEx

Pandas 

View

# DATA PIPELINE




# ОБРАБОТКА ВХОДЯЩЕГО ФАЙЛА И ЗАПОЛНЕНИЕ STG ТАБЛИЦ

Приходящие .xlsx файлы обрабатываю с помощью 

Pandas отлично справляется с чтением данных из различных источников. Читаю файл и сохраняю данные в датафрейм встроенным методом:

```
df = pd.read_excel(stg_dir + filename)
```

Из датафрейма выбираю записи только за нужную дату. Дату, за которую будут обрабатываться записи, беру из имени полученного файла.

 transactions\_01052020.xlsx

Для заполнения таблиц данными создаю функцию:

```
def prep_load_df(table_name: str, columns_list: list, db_settings: dict)
```

В цикле передаю название таблицы и список колонок на вход функции:

```
stg_tables = {  
    'stg_accounts' : ['account', 'account_valid_to', 'client'],  
    'stg_cards': ['card', 'account'],  
    'stg_clients': ['client', 'last_name', 'first_name', 'patronymic',  
    'date_of_birth', 'passport', 'passport_valid_to', 'phone'],  
    'stg_terminals': ['terminal', 'terminal_type', 'city', 'address'],  
    'fact_transactions': ['trans_id', 'date', 'card', 'oper_type', 'amount',  
    'oper_result', 'terminal']  
}
```

Далее по этим данным формируются датафреймы.

Для стейджинговых таблиц к датафрейму добавляется системное поле *start\_dt* с текущим временем.

Данные из датафреймов копируются в таблицы *greenplum* с помощью библиотеки *psycopg2*.



*Psycopg2* - это python адаптер для подключения к *PostgreSQL*.

Для соединения с БД необходимо указать настройки подключения.

В моём случае настройки будут такие:

```
db_settings = {  
    'host' : '93.123.236.98',  
    'database' : 'example_base',  
    'user' : 'example_user',  
    'password' : 'user'  
}
```



Далее открываю соединение и создаю *курсор*.

*Курсор* - это класс библиотеки *psycopg2*, который позволяет выполнять запросы.

```
conn = psycopg2.connect(**db_settings)
gp_cursor = conn.cursor()
```

После того, как функция *prep\_load\_df* отработала и данные были записаны по таблицам, файл *.xlsx* перемещается в папку *load\_completed*.

Папка создается в той же директории, куда приходят *.xlsx* файлы.

# РЕАЛИЗАЦИЯ ИСТОРИЧНОСТИ ДАННЫХ SCD2

*Медленно меняющиеся измерения* (англ. *Slowly Changing Dimensions, SCD*) — это механизм отслеживания изменений в данных измерения в терминах хранилища данных.

SCD2 использует добавление новой строки и дополнительных столбцов. Такой подход позволяет сохранить историчность.

Дополнительно можно добавить служебные столбцы, которые могут отвечать за версионирование, статус, временной интервал, в течение которого данные строки можно считать актуальными.

ID записи	Табельный номер	ФИО	Должность	Отдел	Дата начала	Дата окончания
1026	ИБ-69420	Иванов Сергей Петрович	Младший специалист	Отдел оптовых закупок	2000-01-01T00:00:00	2008-08-08T00:00:00
1027	ИБ-69420	Иванов Сергей Петрович	<b>Главный специалист</b>	<b>Отдел продаж</b>	2008-08-08T00:00:00	NULL

Вместо *NULL* в значении столбца «Дата окончания» можно использовать значение «9999-12-31T00:00:00» для обозначения того, что данная строка наиболее актуальная.

Для отслеживания изменений данных я создал представления, с помощью которых можно найти:

- строки, которые есть в *stg\_* таблицах, но отсутствуют в соответствующих им *dim\_* таблицах
- строки, которые уже есть в *dim\_* таблицах, но в *stg\_* таблицы для этих строк пришли какие-то изменения по одному или нескольким полям.

Для создания представления *stg\_* таблицу соединяем с *dim\_* таблицей при помощи *left join* по полям, изменение которых мы отслеживаем. Выводим только те строки, которые отсутствуют в таблице *dim*. Как это работает, рассмотрим на примере таблицы *clients*.

```
create or replace view clients_update_view as
select
  stg.*
from
  project.stg_clients stg
left join project.dim_clients_hist dth
on
  dth.client_id = stg.client_id
  and dth.last_name = stg.last_name
  and dth.first_name = stg.first_name
  and dth.patronymic = stg.patronymic
  and dth.date_of_birth = stg.date_of_birth
  and dth.passport_num = stg.passport_num
  and dth.passport_valid_to = stg.passport_valid_to
  and dth.phone = dth.phone
where
  dth.client_id is null;
```

После очередной выгрузки в *stg\_*, *view* для таблицы *clients* показывает нам, что появилась строка для обновления/вставки.

clients\_update\_view 1 ×

select \* from clients\_update\_view | Введите SQL выражение чтобы отфильтровать результаты

лица	client_id	last_name	first_name	patronymic	date_of_birth	passport_num	passport_valid_to	phone	start_dt
1	3-58577	Семенцов	Антон	Александрович	1989-08-27	5080911147	2030-07-16	79869514150	2022-02-10 22:35:08.027

Проверяю таблицу *dim\_clients\_hist* и нахожу строку, которая должна быть обновлена. Вижу, что клиент с одним и тем же *client\_id* изменил фамилию.

dim\_clients\_hist 1 ×

select \* from dim\_clients\_hist where client\_id = | Введите SQL выражение чтобы отфильтровать результаты

лица	clier	last_næ	first	patronymic	date_of	passport_nun	passport_valid_to	phone	start_dt	end_dt
1	3-58577	Омелюшкин	Антон	Александрович	1989-08-27	5080911147	2030-07-16	79869514150	2022-02-10 22:33:58.155	9999-12-31 00:00:00.000



После этого выполняю запрос на *update*, который изменит *end\_dt* таблицы *dim\_clients\_hist* на *start\_dt* в таблице *clients\_update\_view*.

clients_update_view 1 ×										
select * from clients_update_view   Введите SQL выражение чтобы отфильтровать результаты										
Таблица	client_id	last_name	first_name	patronymic	date_of_birth	passport_num	passport_valid_to	phone	start_dt	
1	3-58577	Семенцов	Антон	Александрович	1989-08-27	5080911147	2030-07-16	79869514150	2022-02-10 22:35:08.027	

dim_clients_hist 1 ×										
select * from dim_clients_hist where client_id =   Введите SQL выражение чтобы отфильтровать результаты										
Таблица	client_id	last_name	first_name	patronymic	date_of_birth	passport_num	passport_valid_to	phone	start_dt	end_dt
1	3-58577	Омелюшкин	Антон	Александрович	1989-08-27	5080911147	2030-07-16	79869514150	2022-02-10 22:33:58.155	9999-12-31 00:00:00.000

Затем выполняю запрос на *insert* записи из *clients\_update\_view* в *dim\_clients\_hist* с добавлением в поле *end\_dt* значения '31-12-9999'. Эта дата является признаком актуальной записи.

В результате получаю в таблице *dim\_clients\_hist* историю изменения записей.

dim\_clients\_hist 1 ×

select \* from dim\_clients\_hist where client\_id =

Введите SQL выражение чтобы отфильтровать результаты

Таблица		ABC clier	ABC last_na	ABC first	ABC patronymic	date_of	ABC passport_nun	passport_valid_to	ABC phone	start_dt	end_dt
	1	3-58577	Омелюшкин	Антон	Александрович	1989-08-27	5080911147	2030-07-16	79869514150	2022-02-10 22:33:58.155	2022-02-10 22:35:08.027
	2	3-58577	Семенов	Антон	Александрович	1989-08-27	5080911147	2030-07-16	79869514150	2022-02-10 22:35:08.027	9999-12-31 00:00:00.000

Для *update/insert* пишу функцию:

```
def scd_update_insert(table_name : str, db_settings : dict):
```

На вход функции в цикле подается следующий список:

```
    tablenames = ['terminals', 'cards', 'accounts', 'clients']  
    for tablename in tablenames:  
        scd_update_insert(tablename, db_settings)
```

Это позволит обратиться к нужным *view*, в которых отображаются изменения в данных, и к соответствующим таблицам измерений.

```
    dim_{table_name}_hist  
    {table_name}_update_view
```

После этого выполняю *update* и *insert* в соответствующие таблицы.



# ФОРМИРОВАНИЕ ОТЧЁТА

Для составления отчёта пишу функцию, которая будет вставлять записи в таблицу *report* по запросу на каждый тип фрода.

```
def generate_report(report_query : str, db_settings : dict):
    # Открываю соединение. Создаю курсор
    conn = psycopg2.connect(**db_settings)
    cursor = conn.cursor()
    conn.autocommit = True

    # Формирую запрос на insert строк в отчет
    insert_query = f"""
        INSERT INTO project.report(fraud_dt, passport, fio, phone, fraud_type, report_dt)
        {report_query}"""

    cursor.execute(insert_query)
    conn.close()
```

Отчёты формируются после каждой загрузки данных.  
В данном случае дописываются в таблицу за каждый день.

Отчёт за 01-05-2020 записывается в таблицу *report*.

🕒 fraud_dt 📶	ABC passport 📶	ABC fio 📶	ABC phone 📶	ABC fraud_type 📶	🕒 report_dt 📶
2020-05-01 02:07:27.000	2589442754	Манвелян Игорь Петрович	79661011475	3	2022-02-14 17:05:30.152
2020-05-01 02:45:05.000	5263317668	Петропуло Евгений Павлович	79714131751	2	2022-02-14 17:05:30.047
2020-05-01 04:54:37.000	8108438217	Антощук Игорь Владимирович	79451696230	2	2022-02-14 17:05:30.047
2020-05-01 05:10:50.000	9600352239	Гуляков Михаил Александрович	79529669124	2	2022-02-14 17:05:30.047
2020-05-01 11:48:27.000	4783507057	Гуляков Кирилл Петрович	79698158422	3	2022-02-14 17:05:30.152
2020-05-01 12:37:08.000	5670092603	Поджарый Роман Константинович	79410212254	2	2022-02-14 17:05:30.047
2020-05-01 14:18:08.000	3904347595	Фель Евгений Никитович	79692321575	2	2022-02-14 17:05:30.047
2020-05-01 14:40:41.000	6403074379	Аченгов Антон Дмитриевич	79458324907	2	2022-02-14 17:05:30.047

Затем в таблицу *report* дописывается отчёт за 02-05-2020.

🕒 fraud_dt 📶	ABC passport 📶	ABC fio 📶	ABC phone 📶	ABC fraud_type 📶	🕒 report_dt 📶
2020-05-01 14:40:41.000	6403074379	Аченгов Антон Дмитриевич	79458324907	2	2022-02-14 17:05:30.047
2020-05-02 00:40:46.000	5670092603	Поджарый Роман Константинович	79410212254	2	2022-02-14 17:06:39.106
2020-05-02 01:03:32.000	8778457371	Белохов Михаил Романович	79954610552	3	2022-02-14 17:06:39.212
2020-05-02 01:36:40.000	2023060665	Наранов Константин Иванович	79729384491	2	2022-02-14 17:06:39.106
2020-05-02 01:41:22.000	2023060665	Наранов Константин Иванович	79729384491	2	2022-02-14 17:06:39.106
2020-05-02 02:46:19.000	2603146357	Манвелян Петр Кириллович	79441042802	2	2022-02-14 17:06:39.106
2020-05-02 03:03:44.000	2873672645	Сноркин Иван Дмитриевич	79668999229	2	2022-02-14 17:06:39.106
2020-05-02 04:23:39.000	8108438217	Антощук Игорь Владимирович	79451696230	2	2022-02-14 17:06:39.106
2020-05-02 07:38:40.000	8307742530	Городиловский Михаил Антонович	79211450416	2	2022-02-14 17:06:39.106
2020-05-02 08:15:12.000	6708391435	Барецкий Валерий Валериевич	79596741150	2	2022-02-14 17:06:39.106
2020-05-02 13:27:53.000	9599815430	Гуляков Роман Олегович	79362135846	2	2022-02-14 17:06:39.106
2020-05-02 17:27:46.000	1414936096	Штыкашов Дмитрий Никитович	79919933977	3	2022-02-14 17:06:39.212
2020-05-02 18:17:20.000	3407932346	Гладишкин Никита Валериевич	79556552427	2	2022-02-14 17:06:39.106
2020-05-02 18:59:53.000	8108438217	Антощук Игорь Владимирович	79451696230	2	2022-02-14 17:06:39.106
2020-05-02 22:09:16.000	7076445954	Мисик Сергей Николаевич	79497481039	2	2022-02-14 17:06:39.106
2020-05-02 22:50:18.000	3904347595	Фель Евгений Никитович	79692321575	2	2022-02-14 17:06:39.106
2020-05-02 23:21:26.000	5670092603	Поджарый Роман Константинович	79410212254	2	2022-02-14 17:06:39.106
2020-05-02 23:31:27.000	8108438217	Антощук Игорь Владимирович	79451696230	3	2022-02-14 17:06:39.212

начало отчёта  
за 02-05-2020

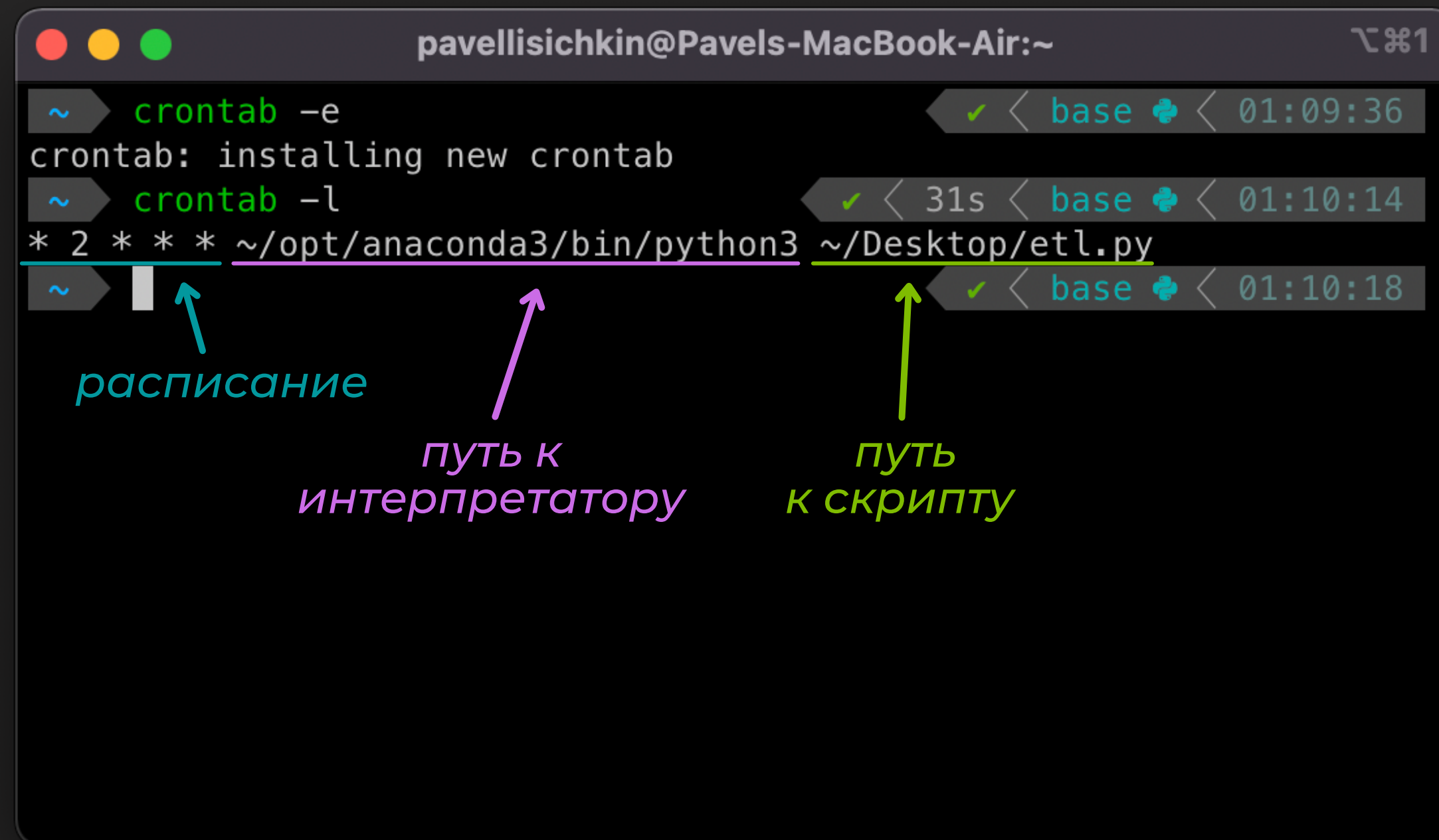


Затем в таблицу *report* дописывается отчёт за 03-05-2020.

🕒 fraud_dt 📶	ABC passport 📶	ABC fio 📶	ABC phone 📶	ABC fraud_type 📶	🕒 report_dt 📶
2020-05-02 23:31:27.000	8108438217	Антощук Игорь Владимирович	79451696230	3	2022-02-14 17:06:39.212
2020-05-03 00:49:39.000	8108438217	Антощук Игорь Владимирович	79451696230	1	2022-02-14 17:07:45.338
2020-05-03 02:50:10.000	8108438217	Антощук Игорь Владимирович	79451696230	1	2022-02-14 17:07:45.338
2020-05-03 02:57:51.000	8108438217	Антощук Игорь Владимирович	79451696230	1	2022-02-14 17:07:45.338
2020-05-03 03:54:33.000	8108438217	Антощук Игорь Владимирович	79451696230	1	2022-02-14 17:07:45.338
2020-05-03 05:22:31.000	8108438217	Антощук Игорь Владимирович	79451696230	1	2022-02-14 17:07:45.338
2020-05-03 05:50:49.000	9861450589	Белохов Александр Васильевич	79323907036	2	2022-02-14 17:07:45.450
2020-05-03 09:42:24.000	8108438217	Антощук Игорь Владимирович	79451696230	1	2022-02-14 17:07:45.338
2020-05-03 10:30:26.000	9188863087	Рубахов Иван Никитович	79817798654	3	2022-02-14 17:07:45.557
2020-05-03 11:23:23.000	8108438217	Антощук Игорь Владимирович	79451696230	1	2022-02-14 17:07:45.338
2020-05-03 13:56:52.000	7173036990	Поджарый Роман Александрович	79343500689	3	2022-02-14 17:07:45.557
2020-05-03 15:04:34.000	7909208960	Городиловский Владимир Сергеевич	79593567861	3	2022-02-14 17:07:45.557
2020-05-03 15:12:18.000	7076445954	Мисик Сергей Николаевич	79497481039	3	2022-02-14 17:07:45.557
2020-05-03 15:16:29.000	1952408369	Юнцов Кирилл Кириллович	79207942259	2	2022-02-14 17:07:45.450
2020-05-03 16:30:08.000	4750296809	Барецкий Андрей Владимирович	79733608401	2	2022-02-14 17:07:45.450
2020-05-03 17:15:49.000	8108438217	Антощук Игорь Владимирович	79451696230	1	2022-02-14 17:07:45.338
2020-05-03 18:34:57.000	8108438217	Антощук Игорь Владимирович	79451696230	1	2022-02-14 17:07:45.338
2020-05-03 18:34:57.000	8108438217	Антощук Игорь Владимирович	79451696230	2	2022-02-14 17:07:45.450
2020-05-03 19:02:54.000	8550980625	Сноркин Павел Олегович	79618054901	2	2022-02-14 17:07:45.450
2020-05-03 20:17:17.000	4750296809	Барецкий Андрей Владимирович	79733608401	2	2022-02-14 17:07:45.450
2020-05-03 20:34:01.000	5670092603	Поджарый Роман Константинович	79410212254	2	2022-02-14 17:07:45.450
2020-05-03 21:08:27.000	1414936096	Штыкашов Дмитрий Никитович	79919933977	2	2022-02-14 17:07:45.450
2020-05-03 23:13:23.000	4958811076	Фель Петр Кириллович	79989865703	2	2022-02-14 17:07:45.450
2020-05-03 23:46:40.000	9861450589	Белохов Александр Васильевич	79323907036	4	2022-02-14 17:07:45.673

начало отчёта  
за 03-05-2020

Для периодического выполнения скрипта оберну его в *cron* с нужным интервалом. Лучше всего делать выгрузку и формировать отчёт ночью, например, каждые сутки в 2 часа утра.



```
pavellisichkin@Pavels-MacBook-Air:~  
~ ➤ crontab -e  
crontab: installing new crontab  
~ ➤ crontab -l  
* 2 * * * ~/opt/anaconda3/bin/python3 ~/Desktop/etl.py  
~ ➤
```

Annotations:

- расписание (points to the cron schedule)
- путь к интерпретатору (points to the python3 path)
- путь к скрипту (points to the etl.py script path)

Terminal status bar shows: ✓ < base < 01:09:36, ✓ < 31s < base < 01:10:14, ✓ < base < 01:10:18

БЛАГОДАРЮ ЗА ВНИМАНИЕ!

# ЛИСТИНГ ФУНКЦИЙ

```
def scd_update_insert(table_name : str, db_settings : dict):
    # Открываю соединение. Создаю курсор
    conn = psycopg2.connect(**db_settings)
    cursor = conn.cursor()
    conn.autocommit = True

    # Получаю список колонок для таблицы
    cursor.execute(f"select * from project.dim_{table_name}_hist dth")
    colnames = [desc[0] for desc in cursor.description]

    # Формирую запрос на update строк в таблице измерений
    update_query = f"""update project.dim_{table_name}_hist dth
                        set end_dt = {table_name}_update_view.start_dt
                        from project.{table_name}_update_view
                        where {table_name}_update_view.{colnames[0]} = dth.{colnames[0]}
                        and end_dt = '9999-12-31';"""

    # Формирую запрос на insert строк в таблицу измерений из стейджинговой таблицы
    insert_query = f"""insert into project.dim_{table_name}_hist
                        ({','.join([str(elem) for elem in colnames])})
                        select {','.join([str(elem) for elem in colnames[:-1]])}, '9999-12-31'::date
                        from project.{table_name}_update_view ;"""

    cursor.execute(update_query)
    cursor.execute(insert_query)

    conn.close()
```

# ЛИСТИНГ ФУНКЦИЙ

```
def prep_load_df(table_name: str, columns_list: list, db_settings: dict):

    # Формирую датафрейм из списка колонок. Оставляю только уникальные
    строки
    stg_df = df[columns_list].drop_duplicates()

    # Проверяю имя таблицы. Если название начинается с 'stg_' то к
    датафрейму добавляю
    # колонку с датой выгрузки. Для таблицы фактов поле start_dt не
    добавляю
    if table_name[:4] == 'stg_':
        stg_df['start_dt'] = pd.to_datetime('now')

    # Записываю датафрейм в csv
    csv_io = io.StringIO()
    stg_df.to_csv(csv_io, sep='\t', header=False, index=False)
    csv_io.seek(0)

    # Соединяюсь с базой. Заливаю csv в нужную таблицу

    conn = psycopg2.connect(**db_settings)
    gp_cursor = conn.cursor()
    gp_cursor.copy_from(csv_io, f'project.{table_name}')
    conn.commit()

    # Закрываю соединение
    conn.close()
```