Commit 75eb0e5

Benjamin Galecki authored and Benjamin Galecki committed on Nov 4, 2025 · ✕ 1 / 3

use UNLOAD instead of direct query

dev

1 parent f7448ff commit 75eb0e5

6 files changed +194 -17 lines changed

Top

Filter files...

- Prod
  - airflow-MWAA
    - load_research_data_fetch_dag.py
  - loadresearch/sqls
    - fpl_backdate.json
    - fpl_daily.json
    - fpl_reprocess.json
  - css_data_alr_package.tar
  - loadresearch.py

6 files changed +194 -17 lines changed

Search within code

Prod/airflow-MWAA/load_research_data_fetch_dag.py

```
@@ -37,7 +37,7 @@
37  37      }
38  38
39  39      # Initialize the DAG
40      -   dag = DAG('load_research_data_fetch', catchup=False, max_active_runs=1,concurrency=1,
            schedule_interval='00 10 * * *', default_args=default_args)
    40  +   dag = DAG('load_research_data_fetch', catchup=False, max_active_runs=1,concurrency=1,
            schedule_interval='00 08 * * *', default_args=default_args)
```

```
41  41      dag_name = 'load_research_data_fetch'
42  42
43  43      config = Config(connect_timeout=30, read_timeout=30, proxies={'https':
            'http://EVAPzen.fpl.com:10262'})
```

Prod/loadresearch/sqls/fpl_backdate.json

```
        @@ -69,15 +69,23 @@
69  69          "process_order": 9
70  70      },
71  71      {
72      -       "dataframe": {"sql" :"select prem_num as premiseid, metertype, read_strt_time,
            round(mrdg*metermtpl,4)::float as kwh,m.ami_dvc_name as meterid, uiq_chnl_num as channel, uom,
            mrdg_drtn_scnd as spi from utl.meter_duration_read_fact m inner join(select prem.prem_num,
            prem.ami_dvc_name as meterid, prem.kwh_csnt as metermtpl, prem.fpl_mtr_num as metertype,case when
            prem.efct_strt_date <= prem.mtr_set_date then prem.mtr_set_date else prem.efct_strt_date end as
            meterstart, prem.efct_end_date as meterstop from (select prem_num, ami_dvc_name, fpl_mtr_num,
            kwh_csnt, min(mtr_set_date) as mtr_set_date, case when (lag(max(mtr_set_date)) over (partition by
            prem_num, mtr_set_date order by max(efct_end_date) asc)::date) is null then mtr_set_date else
            min(efct_strt_date)::date end as efct_strt_date, case when (lead(max(mtr_set_date)) over (partition
            by  prem_num order by max(efct_end_date) asc)::date) is null then max(efct_end_date)::date else
            (lead(max(mtr_set_date)) over (partition by  prem_num order by max(efct_end_date) asc)::date) end as
            efct_end_date from utl.premise_meter where kwh_csnt is not null and ami_dvc_name != 1 and
            (fpl_mtr_num like '%N' or fpl_mtr_num like '_____N%') group by prem_num, ami_dvc_name, fpl_mtr_num,
            kwh_csnt, mtr_set_date) prem) p on p.meterid = m.ami_dvc_name and p.meterstart <=
            m.read_strt_time::date and p.meterstop > m.read_strt_time::date where read_strt_time >=
            'fecth_from_date'::timestamp with time zone and read_strt_time <'fecth_to_date'::timestamp with time
            zone and uiq_chnl_num in (2,4) and read_chnl_flag is null and mrdg is not null   and
            (all_premise_condition_nm) union all select * from (select s.site_id as premiseid, 'evmeter' as
            metertype, RH.read_strt_dttm as read_strt_time, case when lower(uom) = 'wh' then round(sum(mrdg),
            4)/1000 else round(sum(mrdg), 4) end as kwh, CB.charge_box_id as meterid, 1000 as channel, 'kwh' as
            uom, 15 as spi from billing_fpl_fplnw_consolidated.ev_read_hist RH JOIN
            billing_fpl_fplnw_consolidated.ev_site S on RH.site_pk = S.site_pk AND RH.prod_id = S.prod_id AND
            S.crnt_row_flag = TRUE JOIN billing_fpl_fplnw_consolidated.ev_charge_box CB on RH.charge_box_id =
            CB.charge_box_id and CB.connector_id = RH.connector_id AND CB.site_pk = RH.site_pk AND
            RH.read_strt_dttm BETWEEN CB.efct_strt_dttm AND CB.efct_end_dttm where lower(uom) in ('kwh','wh') and
            rh.read_strt_dttm >= 'fecth_from_date'::timestamp with time zone and rh.read_strt_dttm <
            'fecth_to_date'::timestamp with time zone and (all_ev_condition_nm) and mrdg >= 0 group by S.site_id,
            CB.connector_type,RH.read_strt_dttm,CB.charge_box_id, uom) yy where kwh < 6", "parameters_to_replace"
            :["all_premise_condition_nm","all_ev_condition_nm"], "pool":["fecth_from_date", "fecth_to_date"]
            ,"sql_type": "big"},
    72  +       "dataframe": {"sql": "select prem_num as premiseid, metertype, read_strt_time AT TIME ZONE
            'America/New_York' as read_strt_time, round(mrdg*metermtpl,4)::float as kwh,m.ami_dvc_name as
            meterid, uiq_chnl_num as channel, uom, mrdg_drtn_scnd as spi from utl.meter_duration_read_fact m
            inner join(select prem.prem_num, prem.ami_dvc_name as meterid, prem.kwh_csnt as metermtpl,
```

```
prem.fpl_mtr_num as metertype,case when prem.efct_strt_date <= prem.mtr_set_date then
prem.mtr_set_date else prem.efct_strt_date end as meterstart, prem.efct_end_date as meterstop from
(select prem_num, ami_dvc_name, fpl_mtr_num, kwh_csnt, min(mtr_set_date) as mtr_set_date, case when
(lag(max(mtr_set_date)) over (partition by  prem_num, mtr_set_date order by max(efct_end_date)
asc)::date) is null then mtr_set_date else min(efct_strt_date)::date end as efct_strt_date, case when
(lead(max(mtr_set_date)) over (partition by  prem_num order by max(efct_end_date) asc)::date) is null
then max(efct_end_date)::date else (lead(max(mtr_set_date)) over (partition by  prem_num order by
max(efct_end_date) asc)::date) end as efct_end_date from utl.premise_meter where kwh_csnt is not null
and ami_dvc_name != 1 and (fpl_mtr_num like '%N' or fpl_mtr_num like '_____N%') group by prem_num,
ami_dvc_name, fpl_mtr_num, kwh_csnt, mtr_set_date) prem) p on p.meterid = m.ami_dvc_name and
p.meterstart <= m.read_strt_time::date and p.meterstop > m.read_strt_time::date where read_strt_time
>= 'fecth_from_date'::timestamp with time zone and read_strt_time <'fecth_to_date'::timestamp with
time zone and uiq_chnl_num in (2,4) and read_chnl_flag is null and mrdg is not null and
(all_meter_condition_nm) union all select * from (select s.site_id as premiseid, 'evmeter' as
metertype, RH.read_strt_dttm as read_strt_time, case when lower(uom) = 'wh' then round(sum(mrdg),
4)/1000 else round(sum(mrdg), 4) end as kwh, CB.charge_box_id as meterid, 1000 as channel, 'kwh' as
uom, 15 as spi from billing_fpl_fplnw_consolidated.ev_read_hist RH JOIN
billing_fpl_fplnw_consolidated.ev_site S on RH.site_pk = S.site_pk AND RH.prod_id = S.prod_id AND
S.crnt_row_flag = TRUE JOIN billing_fpl_fplnw_consolidated.ev_charge_box CB on RH.charge_box_id =
CB.charge_box_id and CB.connector_id = RH.connector_id AND CB.site_pk = RH.site_pk AND
RH.read_strt_dttm BETWEEN CB.efct_strt_dttm AND CB.efct_end_dttm where lower(uom) in ('kwh','wh') and
rh.read_strt_dttm >= 'fecth_from_date'::timestamp with time zone and rh.read_strt_dttm <
'fecth_to_date'::timestamp with time zone and (all_ev_condition_nm) and mrdg >= 0 group by S.site_id,
CB.connector_type,RH.read_strt_dttm,CB.charge_box_id, uom) yy where kwh < 6",
+       "column_names": ["premiseid", "metertype", "read_strt_time", "kwh", "meterid", "channel",
"uom", "spi"],
+       "parameters_to_replace": ["all_meter_condition_nm","all_ev_condition_nm","fecth_from_date",
"fecth_to_date"],
+       "sql_type": "redshift_unload"
+       },
          "source": "greenplum",
          "new_columns": [],
-       "temp_dataframe" : "mass_market_nm",
-       "join" : {},
+       "temp_dataframe": "mass_market_nm",
+       "join": {},
          "process_order": 10
       },
       {
-       "dataframe": {"sql" :"select prem_num as premiseid, metertype, read_strt_time,
round(mrdg*metermtpl,4)::float as kwh, m.ami_dvc_name as meterid, uiq_chnl_num as channel, uom,
mrdg_drtn_scnd as spi from utl.meter_duration_read_fact m inner join (select prem.prem_num,
prem.ami_dvc_name as meterid, prem.kwh_csnt as metermtpl, prem.fpl_mtr_num as metertype, case when
prem.efct_strt_date <= prem.mtr_set_date then prem.mtr_set_date else prem.efct_strt_date end as
meterstart, prem.efct_end_date as meterstop from (select prem_num, ami_dvc_name, fpl_mtr_num,
```

kwh_csnt, min(mtr_set_date) as mtr_set_date, case when (lag(max(mtr_set_date)) over (partition by prem_num, mtr_set_date order by max(efct_end_date) asc)::date) is null then mtr_set_date else min(efct_strt_date)::date end as efct_strt_date, case when (lead(max(mtr_set_date)) over (partition by prem_num order by max(efct_end_date) asc)::date) is null then max(efct_end_date)::date else (lead(max(mtr_set_date)) over (partition by prem_num order by max(efct_end_date) asc)::date) end as efct_end_date from utl.premise_meter where kwh_csnt is not null and ami_dvc_name != 1 group by prem_num, ami_dvc_name, fpl_mtr_num, kwh_csnt, mtr_set_date) prem) p on p.meterid = m.ami_dvc_name and p.meterstart <= m.read_strt_time::date and p.meterstop > m.read_strt_time::date where read_strt_time >= 'fecth_from_date'::timestamp with time zone and read_strt_time < 'fecth_to_date'::timestamp with time zone and uiq_chnl_num = 1 and UPPER(uom) in ('MYWH','KWH') and read_chnl_flag is null and mrdg is not null and (all_premise_condition) union all select * from (select s.site_id as premiseid, 'evmeter' as metertype, RH.read_strt_dttm as read_strt_time, case when lower(uom) = 'wh' then round(sum(mrdg), 4)/1000 else round(sum(mrdg), 4) end as kwh, CB.charge_box_id as meterid, 1000 as channel, 'kwh' as uom, 15 as spi from billing_fpl_fplnw_consolidated.ev_read_hist RH JOIN billing_fpl_fplnw_consolidated.ev_site S on RH.site_pk = S.site_pk AND RH.prod_id = S.prod_id AND S.crnt_row_flag = TRUE JOIN billing_fpl_fplnw_consolidated.ev_charge_box CB on RH.charge_box_id = CB.charge_box_id and CB.connector_id = RH.connector_id AND CB.site_pk = RH.site_pk AND RH.read_strt_dttm BETWEEN CB.efct_strt_dttm AND CB.efct_end_dttm where lower(uom) in ('kwh','wh') and rh.read_strt_dttm >= 'fecth_from_date'::timestamp with time zone and rh.read_strt_dttm < 'fecth_to_date'::timestamp with time zone and (all_ev_condition) and mrdg >= 0 group by S.site_id, CB.connector_type,RH.read_strt_dttm,CB.charge_box_id, uom) yy where kwh < 6 union all select s.site_id as premiseid, 'uevmeter' as metertype, RH.read_strt_dttm, sum(case when lower(uom) = 'wh' then round(mrdg, 4)/1000 else round(mrdg, 4) end) as mrdg, 'uevmeterid' as meterid, 2000 as channel, 'kwh' as uom, 15 as spi from billing_fpl_fplnw_consolidated.ev_read_hist RH JOIN billing_fpl_fplnw_consolidated.ev_site S on RH.site_pk = S.site_pk AND RH.prod_id = S.prod_id AND S.crnt_row_flag = TRUE JOIN billing_fpl_fplnw_consolidated.ev_charge_box CB on RH.charge_box_id = CB.charge_box_id and CB.connector_id = RH.connector_id and CB.site_pk = RH.site_pk AND RH.read_strt_dttm BETWEEN CB.efct_strt_dttm AND CB.efct_end_dttm where lower(uom) in ('kwh','wh') and rh.read_strt_dttm >= 'fecth_from_date'::timestamp with time zone and rh.read_strt_dttm < 'fecth_to_date'::timestamp with time zone and (all_uev_condition) and s.prod_id != 3 and mrdg >= 0 group by s.site_id, RH.read_strt_dttm having sum(case when lower(uom) = 'wh' then round(mrdg, 4)/1000 else round(mrdg, 4) end) < 150", "parameters_to_replace" :["all_premise_condition", "all_ev_condition", "all_uev_condition"], "pool":["fecth_from_date", "fecth_to_date"] , "sql_type": "big"},

84 + "dataframe": {"sql" :"select prem_num as premiseid, metertype, read_strt_time AT TIME ZONE 'America/New_York' as read_strt_time, round(mrdg*metermtpl,4)::float as kwh, m.ami_dvc_name as meterid, uiq_chnl_num as channel, uom, mrdg_drtn_scnd as spi from utl.meter_duration_read_fact m inner join (select prem.prem_num, prem.ami_dvc_name as meterid, prem.kwh_csnt as metermtpl, prem.fpl_mtr_num as metertype, case when prem.efct_strt_date <= prem.mtr_set_date then prem.mtr_set_date else prem.efct_strt_date end as meterstart, prem.efct_end_date as meterstop from (select prem_num, ami_dvc_name, fpl_mtr_num, kwh_csnt, min(mtr_set_date) as mtr_set_date, case when (lag(max(mtr_set_date)) over (partition by prem_num, mtr_set_date order by max(efct_end_date) asc)::date) is null then mtr_set_date else min(efct_strt_date)::date end as efct_strt_date, case when (lead(max(mtr_set_date)) over (partition by prem_num order by max(efct_end_date) asc)::date) is null

```
then max(efct_end_date)::date else (lead(max(mtr_set_date)) over (partition by  prem_num order by
max(efct_end_date) asc)::date) end as efct_end_date from utl.premise_meter where kwh_csnt is not null
and ami_dvc_name != 1 group by prem_num, ami_dvc_name, fpl_mtr_num, kwh_csnt, mtr_set_date) prem) p
on p.meterid = m.ami_dvc_name and p.meterstart <= m.read_strt_time::date and p.meterstop >
m.read_strt_time::date where read_strt_time >= 'fecth_from_date'::timestamp with time zone and
read_strt_time < 'fecth_to_date'::timestamp with time zone and uiq_chnl_num = 1 and UPPER(uom) in
('MYWH','KWH') and read_chnl_flag is null and mrdg is not null and (all_meter_condition) union all
select * from (select s.site_id as premiseid, 'evmeter' as metertype, RH.read_strt_dttm as
read_strt_time, case when lower(uom) = 'wh' then round(sum(mrdg), 4)/1000 else round(sum(mrdg), 4)
end as kwh, CB.charge_box_id as meterid, 1000 as channel, 'kwh' as uom, 15 as spi from
billing_fpl_fplnw_consolidated.ev_read_hist RH JOIN billing_fpl_fplnw_consolidated.ev_site S on
RH.site_pk = S.site_pk AND RH.prod_id = S.prod_id AND S.crnt_row_flag = TRUE JOIN
billing_fpl_fplnw_consolidated.ev_charge_box CB on RH.charge_box_id = CB.charge_box_id and
CB.connector_id = RH.connector_id AND CB.site_pk = RH.site_pk AND RH.read_strt_dttm BETWEEN
CB.efct_strt_dttm AND CB.efct_end_dttm where lower(uom) in ('kwh','wh') and rh.read_strt_dttm >=
'fecth_from_date'::timestamp with time zone and rh.read_strt_dttm < 'fecth_to_date'::timestamp with
time zone and (all_ev_condition) and mrdg >= 0 group by S.site_id,
CB.connector_type,RH.read_strt_dttm,CB.charge_box_id, uom) yy where kwh < 6 union all select
s.site_id as premiseid, 'uevmeter' as metertype, RH.read_strt_dttm, sum(case when lower(uom) = 'wh'
then round(mrdg, 4)/1000 else round(mrdg, 4) end) as mrdg, 'uevmeterid' as meterid, 2000 as channel,
'kwh' as uom, 15 as spi from billing_fpl_fplnw_consolidated.ev_read_hist RH JOIN
billing_fpl_fplnw_consolidated.ev_site S on RH.site_pk = S.site_pk AND RH.prod_id = S.prod_id AND
S.crnt_row_flag = TRUE JOIN billing_fpl_fplnw_consolidated.ev_charge_box CB on RH.charge_box_id =
CB.charge_box_id and CB.connector_id = RH.connector_id and CB.site_pk = RH.site_pk AND
RH.read_strt_dttm BETWEEN CB.efct_strt_dttm AND CB.efct_end_dttm where lower(uom) in ('kwh','wh') and
rh.read_strt_dttm >= 'fecth_from_date'::timestamp with time zone and rh.read_strt_dttm <
'fecth_to_date'::timestamp with time zone and (all_uev_condition) and s.prod_id != 3 and mrdg >= 0
group by s.site_id, RH.read_strt_dttm having sum(case when lower(uom) = 'wh' then round(mrdg, 4)/1000
else round(mrdg, 4) end) < 150",
85  +       "column_names": ["premiseid", "metertype", "read_strt_time", "kwh", "meterid", "channel",
        "uom", "spi"],
86  +       "parameters_to_replace": ["all_meter_condition", "all_ev_condition",
        "all_uev_condition","fecth_from_date", "fecth_to_date"],
87  +       "sql_type": "redshift_unload"
88  +       },
81  89          "source": "greenplum",
82  90          "new_columns": [],
83  91          "temp_dataframe" : "mass_market_not_nm",
```

Prod/loadresearch/sqls/fpl_daily.json

```
@@ -85,15 +85,23 @@
85  85          "process_order": 6
86  86      },
87  87      {
```

```
88          -       "dataframe": {"sql" :"select prem_num as premiseid, metertype, read_strt_time,
                    round(mrdg*metermtpl,4)::float as kwh,m.ami_dvc_name as meterid, uiq_chnl_num as channel, uom,
                    mrdg_drtn_scnd as spi from utl.meter_duration_read_fact m inner join(select prem.prem_num,
                    prem.ami_dvc_name as meterid, prem.kwh_csnt as metermtpl, prem.fpl_mtr_num as metertype,case when
                    prem.efct_strt_date <= prem.mtr_set_date then prem.mtr_set_date else prem.efct_strt_date end as
                    meterstart, prem.efct_end_date as meterstop from (select prem_num, ami_dvc_name, fpl_mtr_num,
                    kwh_csnt, min(mtr_set_date) as mtr_set_date, case when (lag(max(mtr_set_date)) over (partition by
                    prem_num, mtr_set_date order by max(efct_end_date) asc)::date) is null then mtr_set_date else
                    min(efct_strt_date)::date end as efct_strt_date, case when (lead(max(mtr_set_date)) over (partition
                    by  prem_num order by max(efct_end_date) asc)::date) is null then max(efct_end_date)::date else
                    (lead(max(mtr_set_date)) over (partition by  prem_num order by max(efct_end_date) asc)::date) end
                    as efct_end_date from utl.premise_meter where kwh_csnt is not null and ami_dvc_name != 1 and
                    (fpl_mtr_num like '%N' or fpl_mtr_num like '_____N%') group by prem_num, ami_dvc_name, fpl_mtr_num,
                    kwh_csnt, mtr_set_date) prem) p on p.meterid = m.ami_dvc_name and p.meterstart <=
                    m.read_strt_time::date and p.meterstop > m.read_strt_time::date where read_strt_time >=
                    'fecth_from_date'::timestamp with time zone and read_strt_time <'fecth_to_date'::timestamp with
                    time zone and uiq_chnl_num in (2,4) and read_chnl_flag is null and mrdg is not null and
                    (all_premise_condition_nm) union all select * from (select s.site_id as premiseid, 'evmeter' as
                    metertype, RH.read_strt_dttm as read_strt_time, case when lower(uom) = 'wh' then round(sum(mrdg),
                    4)/1000 else round(sum(mrdg), 4) end as kwh, CB.charge_box_id as meterid, 1000 as channel, 'kwh' as
                    uom, 15 as spi from billing_fpl_fplnw_consolidated.ev_read_hist RH JOIN
                    billing_fpl_fplnw_consolidated.ev_site S on RH.site_pk = S.site_pk AND RH.prod_id = S.prod_id AND
                    S.crnt_row_flag = TRUE JOIN billing_fpl_fplnw_consolidated.ev_charge_box CB on RH.charge_box_id =
                    CB.charge_box_id and CB.connector_id = RH.connector_id AND CB.site_pk = RH.site_pk AND
                    RH.read_strt_dttm BETWEEN CB.efct_strt_dttm AND CB.efct_end_dttm where lower(uom) in ('kwh','wh')
                    and rh.read_strt_dttm >= 'fecth_from_date'::timestamp with time zone and rh.read_strt_dttm <
                    'fecth_to_date'::timestamp with time zone and (all_ev_condition_nm) and mrdg >= 0 group by
                    S.site_id, CB.connector_type,RH.read_strt_dttm,CB.charge_box_id, uom) yy where kwh < 6",
                    "parameters_to_replace" :["all_premise_condition_nm","all_ev_condition_nm"], "pool":
                    ["fecth_from_date", "fecth_to_date"] ,"sql_type": "big"},
    88      +       "dataframe": {"sql": "select prem_num as premiseid, metertype, read_strt_time AT TIME ZONE
                    'America/New_York' as read_strt_time, round(mrdg*metermtpl,4)::float as kwh,m.ami_dvc_name as
                    meterid, uiq_chnl_num as channel, uom, mrdg_drtn_scnd as spi from utl.meter_duration_read_fact m
                    inner join(select prem.prem_num, prem.ami_dvc_name as meterid, prem.kwh_csnt as metermtpl,
                    prem.fpl_mtr_num as metertype,case when prem.efct_strt_date <= prem.mtr_set_date then
                    prem.mtr_set_date else prem.efct_strt_date end as meterstart, prem.efct_end_date as meterstop from
                    (select prem_num, ami_dvc_name, fpl_mtr_num, kwh_csnt, min(mtr_set_date) as mtr_set_date, case when
                    (lag(max(mtr_set_date)) over (partition by  prem_num, mtr_set_date order by max(efct_end_date)
                    asc)::date) is null then mtr_set_date else min(efct_strt_date)::date end as efct_strt_date, case
                    when (lead(max(mtr_set_date)) over (partition by  prem_num order by max(efct_end_date) asc)::date)
                    is null then max(efct_end_date)::date else (lead(max(mtr_set_date)) over (partition by  prem_num
                    order by max(efct_end_date) asc)::date) end as efct_end_date from utl.premise_meter where kwh_csnt
                    is not null and ami_dvc_name != 1 and (fpl_mtr_num like '%N' or fpl_mtr_num like '_____N%') group
                    by prem_num, ami_dvc_name, fpl_mtr_num, kwh_csnt, mtr_set_date) prem) p on p.meterid =
                    m.ami_dvc_name and p.meterstart <= m.read_strt_time::date and p.meterstop > m.read_strt_time::date
```

```
where read_strt_time >= 'fecth_from_date'::timestamp with time zone and read_strt_time
<'fecth_to_date'::timestamp with time zone and uiq_chnl_num in (2,4) and read_chnl_flag is null and
mrdg is not null and (all_meter_condition_nm) union all select * from (select s.site_id as
premiseid, 'evmeter' as metertype, RH.read_strt_dttm as read_strt_time, case when lower(uom) = 'wh'
then round(sum(mrdg), 4)/1000 else round(sum(mrdg), 4) end as kwh, CB.charge_box_id as meterid,
1000 as channel, 'kwh' as uom, 15 as spi from billing_fpl_fplnw_consolidated.ev_read_hist RH JOIN
billing_fpl_fplnw_consolidated.ev_site S on RH.site_pk = S.site_pk AND RH.prod_id = S.prod_id AND
S.crnt_row_flag = TRUE JOIN billing_fpl_fplnw_consolidated.ev_charge_box CB on RH.charge_box_id =
CB.charge_box_id and CB.connector_id = RH.connector_id AND CB.site_pk = RH.site_pk AND
RH.read_strt_dttm BETWEEN CB.efct_strt_dttm AND CB.efct_end_dttm where lower(uom) in ('kwh','wh')
and rh.read_strt_dttm >= 'fecth_from_date'::timestamp with time zone and rh.read_strt_dttm <
'fecth_to_date'::timestamp with time zone and (all_ev_condition_nm) and mrdg >= 0 group by
S.site_id, CB.connector_type,RH.read_strt_dttm,CB.charge_box_id, uom) yy where kwh < 6",
+       "column_names": ["premiseid", "metertype", "read_strt_time", "kwh", "meterid", "channel",
"uom", "spi"],
+       "parameters_to_replace": ["all_meter_condition_nm","all_ev_condition_nm","fecth_from_date",
"fecth_to_date"],
+       "sql_type": "redshift_unload"
+       },
           "source": "greenplum",
           "new_columns": [],
-       "temp_dataframe" : "mass_market_nm",
-       "join" : {},
+       "temp_dataframe": "mass_market_nm",
+       "join": {},
           "process_order": 12
       },
       {
-       "dataframe": {"sql" :"select prem_num as premiseid, metertype, read_strt_time,
round(mrdg*metermtpl,4)::float as kwh, m.ami_dvc_name as meterid, uiq_chnl_num as channel, uom,
mrdg_drtn_scnd as spi from utl.meter_duration_read_fact m inner join (select prem.prem_num,
prem.ami_dvc_name as meterid, prem.kwh_csnt as metermtpl, prem.fpl_mtr_num as metertype, case when
prem.efct_strt_date <= prem.mtr_set_date then prem.mtr_set_date else prem.efct_strt_date end as
meterstart, prem.efct_end_date as meterstop from (select prem_num, ami_dvc_name, fpl_mtr_num,
kwh_csnt, min(mtr_set_date) as mtr_set_date, case when (lag(max(mtr_set_date)) over (partition by
prem_num, mtr_set_date order by max(efct_end_date) asc)::date) is null then mtr_set_date else
min(efct_strt_date)::date end as efct_strt_date, case when (lead(max(mtr_set_date)) over (partition
by  prem_num order by max(efct_end_date) asc)::date) is null then max(efct_end_date)::date else
(lead(max(mtr_set_date)) over (partition by  prem_num order by max(efct_end_date) asc)::date) end
as efct_end_date from utl.premise_meter where kwh_csnt is not null and ami_dvc_name != 1 group by
prem_num, ami_dvc_name, fpl_mtr_num, kwh_csnt, mtr_set_date) prem) p on p.meterid = m.ami_dvc_name
and p.meterstart <= m.read_strt_time::date and p.meterstop > m.read_strt_time::date where
read_strt_time >= 'fecth_from_date'::timestamp with time zone and read_strt_time <
'fecth_to_date'::timestamp with time zone and uiq_chnl_num = 1 and UPPER(uom) in ('MYWH','KWH') and
read_chnl_flag is null and mrdg is not null and (all_premise_condition) union all select * from
```

```
(select s.site_id as premiseid, 'evmeter' as metertype, RH.read_strt_dttm as read_strt_time, case
when lower(uom) = 'wh' then round(sum(mrdg), 4)/1000 else round(sum(mrdg), 4) end as kwh,
CB.charge_box_id as meterid, 1000 as channel, 'kwh' as uom, 15 as spi from
billing_fpl_fplnw_consolidated.ev_read_hist RH JOIN billing_fpl_fplnw_consolidated.ev_site S on
RH.site_pk = S.site_pk AND RH.prod_id = S.prod_id AND S.crnt_row_flag = TRUE JOIN
billing_fpl_fplnw_consolidated.ev_charge_box CB on RH.charge_box_id = CB.charge_box_id and
CB.connector_id = RH.connector_id AND CB.site_pk = RH.site_pk AND RH.read_strt_dttm BETWEEN
CB.efct_strt_dttm AND CB.efct_end_dttm where lower(uom) in ('kwh','wh') and rh.read_strt_dttm >=
'fecth_from_date'::timestamp with time zone and rh.read_strt_dttm < 'fecth_to_date'::timestamp with
time zone and (all_ev_condition) and mrdg >= 0 group by S.site_id,
CB.connector_type,RH.read_strt_dttm,CB.charge_box_id, uom) yy where kwh < 6 union all select
s.site_id as premiseid, 'uevmeter' as metertype, RH.read_strt_dttm, sum(case when lower(uom) = 'wh'
then round(mrdg, 4)/1000 else round(mrdg, 4) end) as mrdg, 'uevmeterid' as meterid, 2000 as
channel, 'kwh' as uom, 15 as spi from billing_fpl_fplnw_consolidated.ev_read_hist RH JOIN
billing_fpl_fplnw_consolidated.ev_site S on RH.site_pk = S.site_pk AND RH.prod_id = S.prod_id AND
S.crnt_row_flag = TRUE JOIN billing_fpl_fplnw_consolidated.ev_charge_box CB on RH.charge_box_id =
CB.charge_box_id and CB.connector_id = RH.connector_id and CB.site_pk = RH.site_pk AND
RH.read_strt_dttm BETWEEN CB.efct_strt_dttm AND CB.efct_end_dttm where lower(uom) in ('kwh','wh')
and rh.read_strt_dttm >= 'fecth_from_date'::timestamp with time zone and rh.read_strt_dttm <
'fecth_to_date'::timestamp with time zone and (all_uev_condition) and s.prod_id != 3 and mrdg >= 0
group by s.site_id, RH.read_strt_dttm having sum(case when lower(uom) = 'wh' then round(mrdg,
4)/1000 else round(mrdg, 4) end) < 150", "parameters_to_replace" :["all_premise_condition",
"all_ev_condition", "all_uev_condition"], "pool":["fecth_from_date", "fecth_to_date"] , "sql_type":
"big"},
100   +       "dataframe": {"sql" :"select prem_num as premiseid, metertype, read_strt_time AT TIME ZONE
'America/New_York' as read_strt_time, round(mrdg*metermtpl,4)::float as kwh, m.ami_dvc_name as
meterid, uiq_chnl_num as channel, uom, mrdg_drtn_scnd as spi from utl.meter_duration_read_fact m
inner join (select prem.prem_num, prem.ami_dvc_name as meterid, prem.kwh_csnt as metermtpl,
prem.fpl_mtr_num as metertype, case when prem.efct_strt_date <= prem.mtr_set_date then
prem.mtr_set_date else prem.efct_strt_date end as meterstart, prem.efct_end_date as meterstop from
(select prem_num, ami_dvc_name, fpl_mtr_num, kwh_csnt, min(mtr_set_date) as mtr_set_date, case when
(lag(max(mtr_set_date)) over (partition by  prem_num, mtr_set_date order by max(efct_end_date)
asc)::date) is null then mtr_set_date else min(efct_strt_date)::date end as efct_strt_date, case
when (lead(max(mtr_set_date)) over (partition by  prem_num order by max(efct_end_date) asc)::date)
is null then max(efct_end_date)::date else (lead(max(mtr_set_date)) over (partition by  prem_num
order by max(efct_end_date) asc)::date) end as efct_end_date from utl.premise_meter where kwh_csnt
is not null and ami_dvc_name != 1 group by prem_num, ami_dvc_name, fpl_mtr_num, kwh_csnt,
mtr_set_date) prem) p on p.meterid = m.ami_dvc_name and p.meterstart <= m.read_strt_time::date and
p.meterstop > m.read_strt_time::date where read_strt_time >= 'fecth_from_date'::timestamp with time
zone and read_strt_time < 'fecth_to_date'::timestamp with time zone and uiq_chnl_num = 1 and
UPPER(uom) in ('MYWH','KWH') and read_chnl_flag is null and mrdg is not null and
(all_meter_condition) union all select * from (select s.site_id as premiseid, 'evmeter' as
metertype, RH.read_strt_dttm as read_strt_time, case when lower(uom) = 'wh' then round(sum(mrdg),
4)/1000 else round(sum(mrdg), 4) end as kwh, CB.charge_box_id as meterid, 1000 as channel, 'kwh' as
uom, 15 as spi from billing_fpl_fplnw_consolidated.ev_read_hist RH JOIN
```

```
billing_fpl_fplnw_consolidated.ev_site S on RH.site_pk = S.site_pk AND RH.prod_id = S.prod_id AND
S.crnt_row_flag = TRUE JOIN billing_fpl_fplnw_consolidated.ev_charge_box CB on RH.charge_box_id =
CB.charge_box_id and CB.connector_id = RH.connector_id AND CB.site_pk = RH.site_pk AND
RH.read_strt_dttm BETWEEN CB.efct_strt_dttm AND CB.efct_end_dttm where lower(uom) in ('kwh','wh')
and rh.read_strt_dttm >= 'fecth_from_date'::timestamp with time zone and rh.read_strt_dttm <
'fecth_to_date'::timestamp with time zone and (all_ev_condition) and mrdg >= 0 group by S.site_id,
CB.connector_type,RH.read_strt_dttm,CB.charge_box_id, uom) yy where kwh < 6 union all select
s.site_id as premiseid, 'uevmeter' as metertype, RH.read_strt_dttm, sum(case when lower(uom) = 'wh'
then round(mrdg, 4)/1000 else round(mrdg, 4) end) as mrdg, 'uevmeterid' as meterid, 2000 as
channel, 'kwh' as uom, 15 as spi from billing_fpl_fplnw_consolidated.ev_read_hist RH JOIN
billing_fpl_fplnw_consolidated.ev_site S on RH.site_pk = S.site_pk AND RH.prod_id = S.prod_id AND
S.crnt_row_flag = TRUE JOIN billing_fpl_fplnw_consolidated.ev_charge_box CB on RH.charge_box_id =
CB.charge_box_id and CB.connector_id = RH.connector_id and CB.site_pk = RH.site_pk AND
RH.read_strt_dttm BETWEEN CB.efct_strt_dttm AND CB.efct_end_dttm where lower(uom) in ('kwh','wh')
and rh.read_strt_dttm >= 'fecth_from_date'::timestamp with time zone and rh.read_strt_dttm <
'fecth_to_date'::timestamp with time zone and (all_uev_condition) and s.prod_id != 3 and mrdg >= 0
group by s.site_id, RH.read_strt_dttm having sum(case when lower(uom) = 'wh' then round(mrdg,
4)/1000 else round(mrdg, 4) end) < 150",
101 +       "column_names": ["premiseid", "metertype", "read_strt_time", "kwh", "meterid", "channel",
"uom", "spi"],
102 +       "parameters_to_replace": ["all_meter_condition", "all_ev_condition",
"all_uev_condition","fecth_from_date", "fecth_to_date"],
103 +       "sql_type": "redshift_unload"
104 +       },
97  105         "source": "greenplum",
98  106         "new_columns": [],
99  107         "temp_dataframe" : "mass_market_not_nm",
```

Prod/loadresearch/sqls/fpl_reprocess.json

```
@@ -69,15 +69,23 @@
69  69          "process_order": 9
70  70        },
71  71        {
72      -       "dataframe": {"sql" :"select prem_num as premiseid, metertype, read_strt_time,
round(mrdg*metermtpl,4)::float as kwh,m.ami_dvc_name as meterid, uiq_chnl_num as channel, uom,
mrdg_drtn_scnd as spi from utl.meter_duration_read_fact m inner join(select prem.prem_num,
prem.ami_dvc_name as meterid, prem.kwh_csnt as metermtpl, prem.fpl_mtr_num as metertype,case when
prem.efct_strt_date <= prem.mtr_set_date then prem.mtr_set_date else prem.efct_strt_date end as
meterstart, prem.efct_end_date as meterstop from (select prem_num, ami_dvc_name, fpl_mtr_num,
kwh_csnt, min(mtr_set_date) as mtr_set_date, case when (lag(max(mtr_set_date)) over (partition by
prem_num, mtr_set_date order by max(efct_end_date) asc)::date) is null then mtr_set_date else
min(efct_strt_date)::date end as efct_strt_date, case when (lead(max(mtr_set_date)) over (partition
by  prem_num order by max(efct_end_date) asc)::date) is null then max(efct_end_date)::date else
(lead(max(mtr_set_date)) over (partition by  prem_num order by max(efct_end_date) asc)::date) end as
```

efct_end_date from utl.premise_meter where kwh_csnt is not null and ami_dvc_name != 1 and (fpl_mtr_num like '%N' or fpl_mtr_num like '_____N%') group by prem_num, ami_dvc_name, fpl_mtr_num, kwh_csnt, mtr_set_date) prem) p on p.meterid = m.ami_dvc_name and p.meterstart <= m.read_strt_time::date and p.meterstop > m.read_strt_time::date where read_strt_time >= 'fecth_from_date'::timestamp with time zone and read_strt_time <'fecth_to_date'::timestamp with time zone and uiq_chnl_num in (2,4) and read_chnl_flag is null and mrdg is not null and (all_premise_condition_nm) union all select * from (select s.site_id as premiseid, 'evmeter' as metertype, RH.read_strt_dttm as read_strt_time, case when lower(uom) = 'wh' then round(sum(mrdg), 4)/1000 else round(sum(mrdg), 4) end as kwh, CB.charge_box_id as meterid, 1000 as channel, 'kwh' as uom, 15 as spi from billing_fpl_fplnw_consolidated.ev_read_hist RH JOIN billing_fpl_fplnw_consolidated.ev_site S on RH.site_pk = S.site_pk AND RH.prod_id = S.prod_id AND S.crnt_row_flag = TRUE JOIN billing_fpl_fplnw_consolidated.ev_charge_box CB on RH.charge_box_id = CB.charge_box_id and CB.connector_id = RH.connector_id AND CB.site_pk = RH.site_pk AND RH.read_strt_dttm BETWEEN CB.efct_strt_dttm AND CB.efct_end_dttm where lower(uom) in ('kwh','wh') and rh.read_strt_dttm >= 'fecth_from_date'::timestamp with time zone and rh.read_strt_dttm < 'fecth_to_date'::timestamp with time zone and (all_ev_condition_nm) and mrdg >= 0 group by S.site_id, CB.connector_type,RH.read_strt_dttm,CB.charge_box_id, uom) yy where kwh < 6", "parameters_to_replace" :["all_premise_condition_nm","all_ev_condition_nm"], "pool":["fecth_from_date", "fecth_to_date"] ,"sql_type": "big"},

72 + "dataframe": {"sql": "select prem_num as premiseid, metertype, read_strt_time AT TIME ZONE 'America/New_York' as read_strt_time, round(mrdg*metermtpl,4)::float as kwh,m.ami_dvc_name as meterid, uiq_chnl_num as channel, uom, mrdg_drtn_scnd as spi from utl.meter_duration_read_fact m inner join(select prem.prem_num, prem.ami_dvc_name as meterid, prem.kwh_csnt as metermtpl, prem.fpl_mtr_num as metertype,case when prem.efct_strt_date <= prem.mtr_set_date then prem.mtr_set_date else prem.efct_strt_date end as meterstart, prem.efct_end_date as meterstop from (select prem_num, ami_dvc_name, fpl_mtr_num, kwh_csnt, min(mtr_set_date) as mtr_set_date, case when (lag(max(mtr_set_date)) over (partition by prem_num, mtr_set_date order by max(efct_end_date) asc)::date) is null then mtr_set_date else min(efct_strt_date)::date end as efct_strt_date, case when (lead(max(mtr_set_date)) over (partition by prem_num order by max(efct_end_date) asc)::date) is null then max(efct_end_date)::date else (lead(max(mtr_set_date)) over (partition by prem_num order by max(efct_end_date) asc)::date) end as efct_end_date from utl.premise_meter where kwh_csnt is not null and ami_dvc_name != 1 and (fpl_mtr_num like '%N' or fpl_mtr_num like '_____N%') group by prem_num, ami_dvc_name, fpl_mtr_num, kwh_csnt, mtr_set_date) prem) p on p.meterid = m.ami_dvc_name and p.meterstart <= m.read_strt_time::date and p.meterstop > m.read_strt_time::date where read_strt_time >= 'fecth_from_date'::timestamp with time zone and read_strt_time <'fecth_to_date'::timestamp with time zone and uiq_chnl_num in (2,4) and read_chnl_flag is null and mrdg is not null and (all_meter_condition_nm) union all select * from (select s.site_id as premiseid, 'evmeter' as metertype, RH.read_strt_dttm as read_strt_time, case when lower(uom) = 'wh' then round(sum(mrdg), 4)/1000 else round(sum(mrdg), 4) end as kwh, CB.charge_box_id as meterid, 1000 as channel, 'kwh' as uom, 15 as spi from billing_fpl_fplnw_consolidated.ev_read_hist RH JOIN billing_fpl_fplnw_consolidated.ev_site S on RH.site_pk = S.site_pk AND RH.prod_id = S.prod_id AND S.crnt_row_flag = TRUE JOIN billing_fpl_fplnw_consolidated.ev_charge_box CB on RH.charge_box_id = CB.charge_box_id and CB.connector_id = RH.connector_id AND CB.site_pk = RH.site_pk AND RH.read_strt_dttm BETWEEN CB.efct_strt_dttm AND CB.efct_end_dttm where lower(uom) in ('kwh','wh') and rh.read_strt_dttm >= 'fecth_from_date'::timestamp with time zone and rh.read_strt_dttm <

```
'fecth_to_date'::timestamp with time zone and (all_ev_condition_nm) and mrdg >= 0 group by S.site_id,
CB.connector_type,RH.read_strt_dttm,CB.charge_box_id, uom) yy where kwh < 6",
+       "column_names": ["premiseid", "metertype", "read_strt_time", "kwh", "meterid", "channel",
"uom", "spi"],
+       "parameters_to_replace": ["all_meter_condition_nm","all_ev_condition_nm","fecth_from_date",
"fecth_to_date"],
+       "sql_type": "redshift_unload"
+       },
          "source": "greenplum",
          "new_columns": [],
-       "temp_dataframe" : "mass_market_nm",
-       "join" : {},
+       "temp_dataframe": "mass_market_nm",
+       "join": {},
          "process_order": 10
       },
       {
-       "dataframe": {"sql" :"select prem_num as premiseid, metertype, read_strt_time,
round(mrdg*metermtpl,4)::float as kwh, m.ami_dvc_name as meterid, uiq_chnl_num as channel, uom,
mrdg_drtn_scnd as spi from utl.meter_duration_read_fact m inner join (select prem.prem_num,
prem.ami_dvc_name as meterid, prem.kwh_csnt as metermtpl, prem.fpl_mtr_num as metertype, case when
prem.efct_strt_date <= prem.mtr_set_date then prem.mtr_set_date else prem.efct_strt_date end as
meterstart, prem.efct_end_date as meterstop from (select prem_num, ami_dvc_name, fpl_mtr_num,
kwh_csnt, min(mtr_set_date) as mtr_set_date, case when (lag(max(mtr_set_date)) over (partition by
prem_num, mtr_set_date order by max(efct_end_date) asc)::date) is null then mtr_set_date else
min(efct_strt_date)::date end as efct_strt_date, case when (lead(max(mtr_set_date)) over (partition
by  prem_num order by max(efct_end_date) asc)::date) is null then max(efct_end_date)::date else
(lead(max(mtr_set_date)) over (partition by  prem_num order by max(efct_end_date) asc)::date) end as
efct_end_date from utl.premise_meter where kwh_csnt is not null and ami_dvc_name != 1 group by
prem_num, ami_dvc_name, fpl_mtr_num, kwh_csnt, mtr_set_date) prem) p on p.meterid = m.ami_dvc_name
and p.meterstart <= m.read_strt_time::date and p.meterstop > m.read_strt_time::date where
read_strt_time >= 'fecth_from_date'::timestamp with time zone and read_strt_time <
'fecth_to_date'::timestamp with time zone and uiq_chnl_num = 1 and UPPER(uom) in ('MYWH','KWH') and
read_chnl_flag is null and mrdg is not null and (all_premise_condition) union all select * from
(select s.site_id as premiseid, 'evmeter' as metertype, RH.read_strt_dttm as read_strt_time, case
when lower(uom) = 'wh' then round(sum(mrdg), 4)/1000 else round(sum(mrdg), 4) end as kwh,
CB.charge_box_id as meterid, 1000 as channel, 'kwh' as uom, 15 as spi from
billing_fpl_fplnw_consolidated.ev_read_hist RH JOIN billing_fpl_fplnw_consolidated.ev_site S on
RH.site_pk = S.site_pk AND RH.prod_id = S.prod_id AND S.crnt_row_flag = TRUE JOIN
billing_fpl_fplnw_consolidated.ev_charge_box CB on RH.charge_box_id = CB.charge_box_id and
CB.connector_id = RH.connector_id AND CB.site_pk = RH.site_pk AND RH.read_strt_dttm BETWEEN
CB.efct_strt_dttm AND CB.efct_end_dttm where lower(uom) in ('kwh','wh') and rh.read_strt_dttm >=
'fecth_from_date'::timestamp with time zone and rh.read_strt_dttm < 'fecth_to_date'::timestamp with
time zone and (all_ev_condition) and mrdg >= 0 group by S.site_id,
CB.connector_type,RH.read_strt_dttm,CB.charge_box_id, uom) yy where kwh < 6 union all select
```

```
s.site_id as premiseid, 'uevmeter' as metertype, RH.read_strt_dttm, sum(case when lower(uom) = 'wh'
then round(mrdg, 4)/1000 else round(mrdg, 4) end) as mrdg, 'uevmeterid' as meterid, 2000 as channel,
'kwh' as uom, 15 as spi from billing_fpl_fplnw_consolidated.ev_read_hist RH JOIN
billing_fpl_fplnw_consolidated.ev_site S on RH.site_pk = S.site_pk AND RH.prod_id = S.prod_id AND
S.crnt_row_flag = TRUE JOIN billing_fpl_fplnw_consolidated.ev_charge_box CB on RH.charge_box_id =
CB.charge_box_id and CB.connector_id = RH.connector_id and CB.site_pk = RH.site_pk AND
RH.read_strt_dttm BETWEEN CB.efct_strt_dttm AND CB.efct_end_dttm where lower(uom) in ('kwh','wh') and
rh.read_strt_dttm >= 'fecth_from_date'::timestamp with time zone and rh.read_strt_dttm <
'fecth_to_date'::timestamp with time zone and (all_uev_condition) and s.prod_id != 3 and mrdg >= 0
group by s.site_id, RH.read_strt_dttm having sum(case when lower(uom) = 'wh' then round(mrdg, 4)/1000
else round(mrdg, 4) end) < 150", "parameters_to_replace" :["all_premise_condition",
"all_ev_condition", "all_uev_condition"], "pool":["fecth_from_date", "fecth_to_date"] , "sql_type":
"big"},
84  +       "dataframe": {"sql" :"select prem_num as premiseid, metertype, read_strt_time AT TIME ZONE
'America/New_York' as read_strt_time, round(mrdg*metermtpl,4)::float as kwh, m.ami_dvc_name as
meterid, uiq_chnl_num as channel, uom, mrdg_drtn_scnd as spi from utl.meter_duration_read_fact m
inner join (select prem.prem_num, prem.ami_dvc_name as meterid, prem.kwh_csnt as metermtpl,
prem.fpl_mtr_num as metertype, case when prem.efct_strt_date <= prem.mtr_set_date then
prem.mtr_set_date else prem.efct_strt_date end as meterstart, prem.efct_end_date as meterstop from
(select prem_num, ami_dvc_name, fpl_mtr_num, kwh_csnt, min(mtr_set_date) as mtr_set_date, case when
(lag(max(mtr_set_date)) over (partition by  prem_num, mtr_set_date order by max(efct_end_date)
asc)::date) is null then mtr_set_date else min(efct_strt_date)::date end as efct_strt_date, case when
(lead(max(mtr_set_date)) over (partition by  prem_num order by max(efct_end_date) asc)::date) is null
then max(efct_end_date)::date else (lead(max(mtr_set_date)) over (partition by  prem_num order by
max(efct_end_date) asc)::date) end as efct_end_date from utl.premise_meter where kwh_csnt is not null
and ami_dvc_name != 1 group by prem_num, ami_dvc_name, fpl_mtr_num, kwh_csnt, mtr_set_date) prem) p
on p.meterid = m.ami_dvc_name and p.meterstart <= m.read_strt_time::date and p.meterstop >
m.read_strt_time::date where read_strt_time >= 'fecth_from_date'::timestamp with time zone and
read_strt_time < 'fecth_to_date'::timestamp with time zone and uiq_chnl_num = 1 and UPPER(uom) in
('MYWH','KWH') and read_chnl_flag is null and mrdg is not null and (all_meter_condition) union all
select * from (select s.site_id as premiseid, 'evmeter' as metertype, RH.read_strt_dttm as
read_strt_time, case when lower(uom) = 'wh' then round(sum(mrdg), 4)/1000 else round(sum(mrdg), 4)
end as kwh, CB.charge_box_id as meterid, 1000 as channel, 'kwh' as uom, 15 as spi from
billing_fpl_fplnw_consolidated.ev_read_hist RH JOIN billing_fpl_fplnw_consolidated.ev_site S on
RH.site_pk = S.site_pk AND RH.prod_id = S.prod_id AND S.crnt_row_flag = TRUE JOIN
billing_fpl_fplnw_consolidated.ev_charge_box CB on RH.charge_box_id = CB.charge_box_id and
CB.connector_id = RH.connector_id AND CB.site_pk = RH.site_pk AND RH.read_strt_dttm BETWEEN
CB.efct_strt_dttm AND CB.efct_end_dttm where lower(uom) in ('kwh','wh') and rh.read_strt_dttm >=
'fecth_from_date'::timestamp with time zone and rh.read_strt_dttm < 'fecth_to_date'::timestamp with
time zone and (all_ev_condition) and mrdg >= 0 group by S.site_id,
CB.connector_type,RH.read_strt_dttm,CB.charge_box_id, uom) yy where kwh < 6 union all select
s.site_id as premiseid, 'uevmeter' as metertype, RH.read_strt_dttm, sum(case when lower(uom) = 'wh'
then round(mrdg, 4)/1000 else round(mrdg, 4) end) as mrdg, 'uevmeterid' as meterid, 2000 as channel,
'kwh' as uom, 15 as spi from billing_fpl_fplnw_consolidated.ev_read_hist RH JOIN
billing_fpl_fplnw_consolidated.ev_site S on RH.site_pk = S.site_pk AND RH.prod_id = S.prod_id AND
```

```
S.crnt_row_flag = TRUE JOIN billing_fpl_fplnw_consolidated.ev_charge_box CB on RH.charge_box_id =
CB.charge_box_id and CB.connector_id = RH.connector_id and CB.site_pk = RH.site_pk AND
RH.read_strt_dttm BETWEEN CB.efct_strt_dttm AND CB.efct_end_dttm where lower(uom) in ('kwh','wh') and
rh.read_strt_dttm >= 'fecth_from_date'::timestamp with time zone and rh.read_strt_dttm <
'fecth_to_date'::timestamp with time zone and (all_uev_condition) and s.prod_id != 3 and mrdg >= 0
group by s.site_id, RH.read_strt_dttm having sum(case when lower(uom) = 'wh' then round(mrdg, 4)/1000
else round(mrdg, 4) end) < 150",
+       "column_names": ["premiseid", "metertype", "read_strt_time", "kwh", "meterid", "channel",
"uom", "spi"],
+       "parameters_to_replace": ["all_meter_condition", "all_ev_condition",
"all_uev_condition","fecth_from_date", "fecth_to_date"],
+       "sql_type": "redshift_unload"
+       },
           "source": "greenplum",
           "new_columns": [],
           "temp_dataframe" : "mass_market_not_nm",
```

css_data_alr_package.tar 8.5 KB

Binary file not shown.

loadresearch.py

```
@@ -302,6 +302,9 @@
        spark.conf.set("spark.sql.adaptive.enabled", "true")
        spark.conf.set("spark.sql.adaptive.coalescePartitions.enabled", "true")
        spark.conf.set("spark.sql.adaptive.skewJoin.enabled", "true")
+       spark.conf.set("spark.sql.legacy.timeParserPolicy", "LEGACY")
+       spark.conf.set('spark.sql.parquet.datetimeRebaseModeInRead', 'LEGACY')
+       spark.conf.set('spark.sql.legacy.parquet.datetimeRebaseModeInWrite', 'CORRECTED')

        return spark

@@ -513,6 +516,106 @@
                            task_options = {'file_config': file_config}
                            df = pool_big_sql(sql, pool_list, ranges_list, process_to_dataframe,
connection_task, task_options)

+               elif file_config['sql_type'].upper() == 'REDSHIFT_UNLOAD':
+                   # pool_list = None
+                   connection_destination = get_connection_jdbc(sm_client, connection_key,
'alr_s3')
```

```
522 +                     s3_path = 's3://{}/{}/{}'.format(connection_destination['data_bucket_name'],
    backdate + '_unload','filename_')
523 +
524 +                     # Parse S3 path
525 +                     # s3_path = file_config['s3_path']
526 +                     s3_parts = s3_path.replace('s3://', '').split('/', 1)
527 +                     bucket_name = s3_parts[0]
528 +                     prefix = s3_parts[1] if len(s3_parts) > 1 else ''
529 +
530 +                     # Delete existing files
531 +                     s3_client = boto3.client('s3')
532 +                     response = s3_client.list_objects_v2(Bucket=bucket_name, Prefix=prefix)
533 +
534 +                     if 'Contents' in response:
535 +                         delete_objects = [{'Key': obj['Key']} for obj in response['Contents']]
536 +                         s3_client.delete_objects(
537 +                             Bucket=bucket_name,
538 +                             Delete={'Objects': delete_objects}
539 +                         )
540 +                         print(f"Cleared {len(delete_objects)} existing files from {s3_path}")
541 +
542 +                     sql = file_config['sql']
543 +
544 +                     if 'parameters' in file_config:
545 +                         if file_config['parameters'] != None:
546 +                             sql = sql.format(*_p)
547 +                     if 'parameters_to_replace' in file_config:
548 +                         if file_config['parameters_to_replace'] != None:
549 +                             for pr in file_config['parameters_to_replace']:
550 +                                 if eval(pr) == None:
551 +                                     continue_process = False
552 +                                 else:
553 +                                     sql = sql.replace(pr, eval(pr))
554 +
555 +                     connection_iam = get_connection_jdbc(sm_client, connection_key, 'iam_role')
556 +                     iam_role = connection_iam['unload_iam']
557 +                     delimiter = '|'
558 +                     gzip_option = 'GZIP'
559 +                     overwrite_option = 'ALLOWOVERWRITE'
560 +                     maxfilesize = 60
561 +
562 +                     # Build UNLOAD command
563 +                     unload_command = f"""
564 +                     UNLOAD ($${sql}$$)
565 +                     TO '{s3_path}'
```

```
+                    IAM_ROLE '{iam_role}'
+                    DELIMITER '{delimiter}'
+                    {gzip_option}
+                    {overwrite_option}
+                    MAXFILESIZE {maxfilesize}
+                    """.strip()
+
+                    unload_command = to_fStrings(unload_command)
+
+                    print(datetime.strptime(str(datetime.now().strftime('%Y-%m-%d-%H:%M:%S')), '%Y-%m-%d-%H:%M:%S'))
+                    # Execute unload in Redshift
+                    try:
+                        # Execute unload in Redshift
+                        connection_redshift = get_connection_jdbc(sm_client, connection_key, item['source']) if item['source'] != 'sparksql' else None
+                        postgres_connection = PostgresSQLConnection(connection_redshift)
+                        postgres_connection.execute_actions(unload_command)
+
+                        print(f"UNLOAD completed to: {s3_path}")
+
+                        # Read back from S3
+                        df = spark.read \
+                            .option("delimiter", delimiter) \
+                            .option("inferSchema", "true") \
+                            .option("header", "false") \
+                            .csv(s3_path + "*.gz")
+
+                        # CHECK FOR EMPTY DATAFRAME BEFORE PROCESSING
+                        try:
+                            # Evaluation to check if DataFrame has data
+                            row_count = df.count()
+
+                            if row_count == 0:
+                                continue_process = False
+                                print(f"No data returned from UNLOAD for {item['temp_dataframe']} - DataFrame is empty (0 rows)")
+                            else:
+                                print(f"UNLOAD successful: {row_count} rows")
+
+                                # Add column names only if DataFrame has data
+                                if 'column_names' in file_config:
+                                    column_names = file_config['column_names']
+                                    for i, col_name in enumerate(column_names):
+                                        df = df.withColumnRenamed(f"_c{i}", col_name)
```

```
608 +
609 +                         except Exception as e:
610 +                             continue_process = False
611 +                             print(f"Error checking DataFrame for {item['temp_dataframe']}: {e}")
612 +
613 +                     except Exception as e:
614 +                         continue_process = False
615 +                         print(f"UNLOAD/Read operation failed for {item['temp_dataframe']}: {e}")
616 +
617 +                     print(datetime.strptime(str(datetime.now().strftime('%Y-%m-%d-%H:%M:%S')), '%Y-%m-%d-%H:%M:%S'))
618 +
516 619                  else:
517 620                      sql = file_config['sql']
518 621                      if 'parameters' in file_config:
```

```
@@ -1004,6 +1107,8 @@
1004 1107                  if item['temp_dataframe'] == 'mass_market_nm':
1005 1108                      df_nm = df.withColumn('read_date', to_date(F.col('read_strt_time'), 'yyyy-MM-dd'))
1006 1109                      df.unpersist()
     1110 +                    df_nm = df_nm.withColumn('premiseid', F.col("premiseid").cast('string'))
     1111 +                    df_nm = df_nm.withColumn('read_strt_time', F.col("read_strt_time").cast('timestamp'))
1007 1112                      df_nm = df_nm.dropDuplicates()
1008 1113                      df_nm.cache()
1009 1114
```

```
@@ -1127,6 +1232,8 @@
1127 1232
1128 1233                      df_not_nm = df.withColumn('read_date', to_date(F.col('read_strt_time'), 'yyyy-MM-dd'))
1129 1234                      df.unpersist()
     1235 +                    df_not_nm = df_not_nm.withColumn('premiseid', F.col("premiseid").cast('string'))
     1236 +                    df_not_nm = df_not_nm.withColumn('read_strt_time', F.col("read_strt_time").cast('timestamp'))
1130 1237                      df_not_nm = df_not_nm.dropDuplicates()
1131 1238
1132 1239                      df_not_nm.cache()
```

```
@@ -1432,8 +1539,6 @@
1432 1539                      power_billing = power_billing.dropDuplicates()
1433 1540
1434 1541
```

```
1435          -
1436          -
1437    1542                  if item['temp_dataframe'] == 'power_billing_gulf':
1438    1543
1439    1544                      for col in df.columns:
      @@ -1453,7 +1558,13 @@
1453    1558                      g_pb = df.select('premiseid', 'mandt', 'uom', 'profile',  'read_date',
        'divisor' ,F.expr(unpivotExpr))
1454    1559
1455    1560                      g_pb = g_pb.withColumn('read_strt_time_str', (F.concat(F.col('read_date'),
        F.lit('T'), substring('str_time', 4, 2), F.lit(':'),substring('str_time', 6, 2), F.lit(':00Z'))))
1456          -                   g_pb = g_pb.withColumn('read_strt_time',
        F.date_format(F.to_timestamp(F.col('read_strt_time_str'), "yyyy-MM-dd'T'HH:mm:ssXXX"), "yyyy-MM-
        dd HH:mm:ssx").cast('timestamp'))
        1561  +                   # g_pb = g_pb.withColumn('read_strt_time',
        F.date_format(F.to_timestamp(F.col('read_strt_time_str'), "yyyy-MM-dd'T'HH:mm:ssXXX"), "yyyy-MM-
        dd HH:mm:ssx").cast('timestamp'))
        1562  +                   g_pb = g_pb.withColumn('read_strt_time',
        1563  +                                          F.date_format(
        1564  +                                              F.to_timestamp(F.col('read_strt_time_str'), "yyyy-
        MM-dd'T'HH:mm:ssXXX"),
        1565  +                                              "yyyy-MM-dd HH:mm:ss"  # No timezone formatting
        1566  +                                          ).cast('timestamp')
        1567  +                                          )
1457    1568                      #g_pb = g_pb.withColumn('read_strt_time',
        (F.from_utc_timestamp((F.concat(F.col('read_date'), F.lit(' '), substring('str_time', 4,2),
        F.lit(':'),substring('str_time', 6,2), F.lit(':00'))), "America/New_York")).cast('timestamp'))
1458    1569
1459    1570                      g_pb = g_pb.withColumn('spi', F.lit(15)).withColumn('channel',
        F.lit(1)).withColumn('mrdg', F.col('mrdg')/F.col('divisor')).withColumn('read_date',
        F.to_date(F.col('read_strt_time')))
      @@ -1986,6 +2097,49 @@
1986    2097                  df_study = None
1987    2098                  df_study_premise = None
1988    2099
        2100  +           else:  # break_process == True
        2101  +               # Create audit entry for skipped process
        2102  +               audit_datetime_end = datetime.strptime(str(datetime.now().strftime('%Y-%m-%d-
        %H:%M:%S')),
        2103  +                                                      '%Y-%m-%d-%H:%M:%S')
        2104  +
        2105  +               # Create audit row for each target that would have been processed
        2106  +               for target in target_list:
        2107  +                   for db in list(target['target_database']):
```

```
2108 +                          r_audit.append({
2109 +                              'audit_id': job_run_id,
2110 +                              'etl_process_detail_id': etl_process_detail_id,
2111 +                              'etl_process_run_date': process_run_date,
2112 +                              'etl_process_start_dt': audit_datetime_start,
2113 +                              'etl_process_end_dt': audit_datetime_end,
2114 +                              'etl_process_status': 'skipped',
2115 +                              # or 'no_data' - whatever status indicates no processing needed
2116 +                              'etl_process_rows': 0,
2117 +                              'etl_process_id': etl_process_order,
2118 +                              'target_name': db['destination'],
2119 +                              'step': 'target-write'
2120 +                          })
2121 +
2122 +                  # Also create audit entries for post_actions if needed
2123 +                  for post in post_actions_list:
2124 +                      audit_datetime_end = datetime.strptime(str(datetime.now().strftime('%Y-%m-%d-
     %H:%M:%S')),
2125 +                                                             '%Y-%m-%d-%H:%M:%S')
2126 +                      r_audit.append({
2127 +                          'audit_id': job_run_id,
2128 +                          'etl_process_detail_id': etl_process_detail_id,
2129 +                          'etl_process_run_date': process_run_date,
2130 +                          'etl_process_start_dt': audit_datetime_start,
2131 +                          'etl_process_end_dt': audit_datetime_end,
2132 +                          'etl_process_status': 'no data',
2133 +                          'etl_process_rows': 0,
2134 +                          'etl_process_id': etl_process_order,
2135 +                          'target_name': post['connection'],
2136 +                          'step': 'post_action'
2137 +                      })
2138 +
2139 +                  print('Process skipped - no data to process. Audit rows created.')
2140 +                  print('r_audit')
2141 +                  print(r_audit)
2142 +
1989 2143              #spark.sql('DROP TABLE IF EXISTS default.{}'.format(table_name_mm_gulf)).count()
1990 2144
1991 2145
     @@ -2013,7 +2167,6 @@
2013 2167                  write_error(region_name, Topic, cw_log_group, eSubjectName, eerror)
2014 2168
2015 2169
2016      -
```

```
2017  2170          except Exception as e:
2018  2171              print('Error description: {}'.format(e))
2019  2172              eerror = "Hi,\n\nBelow is the Failure details :\n  Failure Summary : \n ----------------
                ------------------------------\n Error: {}\n". \
```

Comments 0

Lock conversation

Comment

Subscribe You're not receiving notifications from this thread.