In [21]:
```
import numpy as np
import pandas as pd
from sqlalchemy import create_engine
df = pd.read_csv("application_data.csv")
df.columns = df.columns.str.lower()
```

In [31]:
```
df.head(10)
```

Out[31]:

	sk_id_curr	target	name_contract_type	code_gender	flag_own_car	flag_own_realty	cnt_
0	100002	1	Cash loans	M	N	Y	
1	100003	0	Cash loans	F	N	N	
2	100004	0	Revolving loans	M	Y	Y	
3	100006	0	Cash loans	F	N	Y	
4	100007	0	Cash loans	M	N	Y	
5	100008	0	Cash loans	M	N	Y	
6	100009	0	Cash loans	F	Y	Y	
7	100010	0	Cash loans	M	Y	Y	
8	100011	0	Cash loans	F	N	Y	
9	100012	0	Revolving loans	M	N	Y	

10 rows × 122 columns

In [2]:
```
# Define the connection URL:
conn_url = 'postgresql://postgres:123@localhost/project'
# Create an engine that connects to PostgreSQL:
engine = create_engine(conn_url)
# Establish a connection:
connection = engine.connect()
```

In [6]:
```
stmt = """
CREATE TABLE region (
    region_id int not null primary key,
    region_type varchar(50) not null,
    region_rating int not null
    );

CREATE TABLE external_source (
    external_source_id int not null primary key,
    external_source_type varchar(50) not null,
    external_source_value numeric(8,2)
    );

CREATE TABLE AMT_REQ_Credit_Bureau (
    AMT_REQ_Credit_Bureau_id int not null primary key,
    AMT_REQ_Credit_Bureau_hour int,
    AMT_REQ_Credit_Bureau_day int,
    AMT_REQ_Credit_Bureau_week int,
    AMT_REQ_Credit_Bureau_mon int,
    AMT_REQ_Credit_Bureau_qrt int,
    AMT_REQ_Credit_Bureau_year int
    );

CREATE TABLE gender (
    gender_id int not null primary key,
```

```
    gender_type varchar(50)
    );

CREATE TABLE income (
    Income_ID int,
    Annual_Income_Total int NOT NULL,
    Income_Type varchar(50) NOT NULL,
    PRIMARY KEY (income_iD)
    );

CREATE TABLE building_info (
    Building_Info_ID int,
    Building_Info_Type varchar(50) NOT NULL,
    Building_Info_Value numeric(8,2),
    PRIMARY KEY (Building_Info_ID)
    );

CREATE TABLE observation(
    Observation_ID    integer,
    Observation_Type  varchar(50) NOT NULL,
    Observation_Value numeric(8,2),
    PRIMARY KEY (Observation_ID)
    );

CREATE TABLE Loan_Applicant (
    sk_id_curr int not null primary key,
    own_car_age int,
    days_registration int not null,
    days_id_publish int not null,
    days_birth int not null,
    gender_id int not null references gender (gender_id),
    income_id int not null references income (income_id),
    AMT_REQ_Credit_Bureau_id int not null references AMT_REQ_Credit_Bureau (A
    );

CREATE TABLE apl_bld_conn (
    sk_id_curr int not null references loan_applicant (sk_id_curr),
    building_info_id int not null references building_info (building_info_id)
    primary key (sk_id_curr,building_info_id)
    );

CREATE TABLE apl_reg_conn (
    sk_id_curr int not null references loan_applicant (sk_id_curr),
    region_id int not null references region (region_id),
    primary key (sk_id_curr,region_id)
    );

CREATE TABLE apl_etn_conn (
    sk_id_curr int not null references loan_applicant (sk_id_curr),
    external_source_id int not null references external_source (external_sour
    primary key (sk_id_curr,external_source_id)
    );

CREATE TABLE apl_obs_conn (
    sk_id_curr int not null references loan_applicant (sk_id_curr),
    observation_id int not null references observation (observation_id),
    primary key (sk_id_curr,observation_id)
    );

CREATE TABLE property (
    property_id int not null,
    sk_id_curr int not null references loan_applicant (sk_id_curr),
    property_name varchar(50),
    if_owned boolean,
```

```
    primary key (property_id,sk_id_curr)
    );


CREATE TABLE family (
    family_id int not null,
    sk_id_curr int not null references loan_applicant (sk_id_curr),
    cnt_family_members int,
    family_status varchar(50),
    cnt_children int,
    primary key (family_id,sk_id_curr)
    );

CREATE TABLE organization (
    organization_id int not null,
    sk_id_curr int not null references loan_applicant (sk_id_curr),
    organization_type varchar(50),
    primary key (organization_id, sk_id_curr)
    );

CREATE TABLE address_match (
    address_match_id int not null,
    sk_id_curr int not null references loan_applicant (sk_id_curr),
    address_match_type varchar(50),
    address_matched boolean,
    primary key (address_match_id,sk_id_curr)
    );

CREATE TABLE document (
    document_id int not null,
    document_code varchar(30) not null,
    document_provided int,
    primary key (document_id)
    );

CREATE TABLE apl_doc_conn (
    sk_id_curr int not null references loan_applicant (sk_id_curr),
    document_id int not null references document (document_id),
    primary key (sk_id_curr,document_id)
    );

CREATE TABLE occupation (
    occupation_id int not null,
    sk_id_curr int not null,
    occupation_type varchar(50),
    days_employed int,
    primary key (occupation_id, sk_id_curr),
    FOREIGN KEY(SK_ID_CURR) REFERENCES Loan_Applicant(SK_ID_CURR)
    );

CREATE TABLE contact (
    contact_id int not null,
    sk_id_curr int not null,
    contact_type varchar(50),
    contact_provided int,
    primary key (contact_id, sk_id_curr),
    FOREIGN KEY(SK_ID_CURR) REFERENCES Loan_Applicant(SK_ID_CURR)
    );

CREATE TABLE target (
    Target_ID int NOT NULL,
    SK_ID_CURR int NOT NULL,
    Target_Name varchar(50),
    PRIMARY KEY (Target_ID, SK_ID_CURR),
```

```
    FOREIGN KEY (sk_id_curr) REFERENCES loan_applicant(sk_id_curr)
    );

CREATE TABLE suite (
    Suite_ID int NOT NULL,
    Suite_Name varchar(50),
    PRIMARY KEY (Suite_ID)
    );

CREATE TABLE contract (
    Contract_ID int NOT NULL,
    Suite_ID int NOT NULL,
    Target_ID int NOT NULL,
    SK_ID_CURR int NOT NULL,
    Hour_Appr int,
    Weekday_Appr varchar(25),
    Amt_Credit numeric(10,2),
    Amt_Annuity numeric(10,2),
    Contract_Name varchar(50),
    PRIMARY KEY (contract_id, suite_id, target_id, sk_id_curr),
    FOREIGN KEY (Suite_ID) REFERENCES suite(Suite_ID),
    FOREIGN KEY (SK_ID_CURR) REFERENCES Loan_Applicant(SK_ID_CURR)
    );
"""
connection.execute(stmt)
```

Out[6]: <sqlalchemy.engine.result.ResultProxy at 0x7fb5d76a4f40>

In [7]:
```
#region
Region = df[['region_rating_client','region_rating_client_w_city']]
Region=Region.melt()
Region=Region.drop_duplicates()
Region= Region.reset_index(drop=True).reset_index()
Region['index'] = Region['index'] + 1
Region.rename(columns={'index':'region_id','variable':'region_type','value':'
Region.to_sql('region',con=connection,if_exists='append',index=False)
```

In [8]:
```
#external_source
External = df[['ext_source_1','ext_source_2','ext_source_3']]
External=External.melt()
External=External.drop_duplicates()
External= External.reset_index(drop=True).reset_index()
External['index'] = External['index'] + 1
External.rename(columns={'index':'external_source_id','variable':'external_so
External.to_sql('external_source',con=connection,if_exists='append',index=Fal
```

In [9]:
```
#AMT_REQ
AMT = df[['amt_req_credit_bureau_hour','amt_req_credit_bureau_day','amt_req_c
AMT=AMT.drop_duplicates()
AMT= AMT.reset_index(drop=True).reset_index()
AMT['index'] = AMT['index'] + 1
AMT.rename(columns={'index':'amt_req_credit_bureau_id'}, inplace = True)
AMT.to_sql('amt_req_credit_bureau',con=connection,if_exists='append',index=Fa
```

In [10]:
```
amt_1 = df[['sk_id_curr','amt_req_credit_bureau_hour','amt_req_credit_bureau_
con_amt = pd.merge(amt_1,AMT)[['sk_id_curr','amt_req_credit_bureau_id']]
```

In [11]:
```
#gender
gender_df = pd.DataFrame(df.code_gender.unique(), columns=['gender_type'])
gender_df.insert(0, 'gender_id', range(1, 1 + len(gender_df)))
gender_df.to_sql(name='gender', con=engine, if_exists='append', index=False)
```

In [12]:
```
gender_1 = df[['sk_id_curr','code_gender']]
con_gender = pd.merge(gender_1,gender_df,left_on='code_gender',right_on='gend
```

In [13]:
```
#income
income_df=df[['amt_income_total','name_income_type']]
income_df.columns = ['annual_income_total', 'income_type']
income_df=income_df.drop_duplicates()
income_df.insert(0,'income_id', range(1, 1 + len(income_df)))
income_df.to_sql(name='income', con=engine, if_exists='append', index=False)
```

In [14]:
```
income_1 = df[['sk_id_curr','amt_income_total','name_income_type']]
income_1.columns = ['sk_id_curr','annual_income_total', 'income_type']
con_income = pd.merge(income_1,income_df)[['sk_id_curr','income_id']]
```

In [15]:
```
#building_info
cols = list(range(44,85))
building_df=df[df.columns[cols]]####create a temporary dataframe.
building_df=building_df.melt()##Set the id_vars parameter to select column  s
building_df = building_df.rename(columns={'variable': 'building_info_type', '
building_df=building_df.drop_duplicates()## drop the duplicate rows
building_df.insert(0,'building_info_id', range(1, 1 + len(building_df)))
building_df.to_sql(name='building_info', con=engine, if_exists='append', inde
```

In [16]:
```
#observation
observation_df=df.loc[:,('obs_30_cnt_social_circle','def_30_cnt_social_circle
observation_df=observation_df.melt()##Set the id_vars parameter to select col
observation_df = observation_df.rename(columns={'variable': 'observation_type
observation_df=observation_df.drop_duplicates()
observation_df.insert(0,'observation_id', range(1, 1 + len(observation_df)))#
observation_df.to_sql(name='observation', con=engine, if_exists='append', ind
```

In [17]:
```
#Loan_Applicant
Loan_Applicant_df=df[['sk_id_curr', 'own_car_age','days_registration','days_i
Loan_Applicant_df = pd.merge(Loan_Applicant_df,con_amt)
Loan_Applicant_df = pd.merge(Loan_Applicant_df,con_gender)
Loan_Applicant_df = pd.merge(Loan_Applicant_df,con_income)
```

In [18]:
```
Loan_Applicant_df.to_sql(name='loan_applicant', con=engine, if_exists='append
```

In [17]:
```
#apl_bld_conn
cols = [0]+list(range(44,85))
building_1 = df[df.columns[cols]]
building_1=pd.melt(building_1,id_vars=['sk_id_curr'],var_name='building_info_
con_building = pd.merge(building_1,building_df)[['sk_id_curr','building_info_

con_building.to_sql(name='apl_bld_conn', con=engine, if_exists='append', inde
```

In [19]:
```
#apl_reg_conn
Region_1 = df[['sk_id_curr','region_rating_client','region_rating_client']]
Region_1=pd.melt(Region_1,id_vars=['sk_id_curr'],value_vars=['region_rating_c
con_region = pd.merge(Region_1,Region)[['sk_id_curr','region_id']]
con_region.to_sql(name='apl_reg_conn', con=engine, if_exists='append', index=
```

In [33]:
```
#apl_etn_conn
External_1 = df[['sk_id_curr','ext_source_1','ext_source_2','ext_source_3']]
External_1=pd.melt(External_1,id_vars=['sk_id_curr'],value_vars=['ext_source_
con_external = pd.merge(External_1,External)[['sk_id_curr','external_source_i
con_external.to_sql(name='apl_etn_conn', con=engine, if_exists='append', inde
```

In [39]:
```
#apl_obs_conn
```

```
observation_1=df.loc[:,('sk_id_curr','obs_30_cnt_social_circle','def_30_cnt_s
observation_1=pd.melt(observation_1,id_vars=['sk_id_curr'],var_name='observat
con_observation = pd.merge(observation_1,observation_df)[['sk_id_curr','obser
con_observation.to_sql(name='apl_obs_conn', con=engine, if_exists='append', i
```

In [44]:
```
#property
property_df = df[['sk_id_curr','flag_own_car','flag_own_realty']]
property_df = pd.melt(property_df, id_vars=["sk_id_curr"], var_name="property
property_df = property_df.sort_values(["sk_id_curr","property_name"])
property_df.insert(0,'property_id', range(1, 1 + len(property_df)))
property_df.to_sql(name='property', con=engine, if_exists='append', index=Fal
```

In [53]:
```
#family
family_df = df[['sk_id_curr','cnt_children','name_family_status','cnt_fam_meml
family_df.rename(columns = {'name_family_status':'family_status','cnt_fam_meml
family_df = family_df.drop_duplicates()
family_df.insert(0,'family_id', range(1, 1 + len(family_df)))
family_df.to_sql(name='family', con=engine, if_exists='append', index=False)
```

In [57]:
```
#organization
organization_df = df[['sk_id_curr','organization_type']]
organization_df = organization_df.drop_duplicates()
organization_df.insert(0,'organization_id', range(1, 1 + len(organization_df)
organization_df.to_sql(name='organization', con=engine, if_exists='append', i
```

In [60]:
```
#address_match
address_df=df[['sk_id_curr','reg_region_not_live_region',
       'reg_region_not_work_region', 'live_region_not_work_region',
       'reg_city_not_live_city', 'reg_city_not_work_city',
       'live_city_not_work_city']]
address_df.drop_duplicates()
address_df = pd.melt(address_df, id_vars=["sk_id_curr"], var_name="address_ma
address_df = address_df.sort_values(["sk_id_curr","address_match_type"])
address_df.insert(0,'address_match_id', range(1, 1 + len(address_df)))
address_df.columns = [ column.lower() for column in list(address_df.columns)]
address_df['address_matched']=address_df['address_matched'].map({0:'N', 1:'Y'
address_df.to_sql(name='address_match', con=engine, if_exists='append', index
```

In [73]:
```
#document
document_df = df.loc[:, ('flag_document_2','flag_document_3','flag_document_4
                        ,'flag_document_7','flag_document_8','flag_document_9
                        ,'flag_document_13','flag_document_14','flag_document
                        ,'flag_document_19','flag_document_20','flag_document
document_df= document_df.melt()
document_df = document_df.rename(columns = {'variable': 'document_code', 'val
document_df = document_df.drop_duplicates()
document_df.insert(0, 'document_id', range(1, 1 + len(document_df)))
document_df.to_sql(name='document', con=engine, if_exists='append', index=Fal
```

In [74]:
```
#apl_doc_conn
document1 = df.loc[:, ('sk_id_curr','flag_document_2','flag_document_3','flag
                        ,'flag_document_7','flag_document_8','flag_document_9
                        ,'flag_document_13','flag_document_14','flag_document
                        ,'flag_document_19','flag_document_20','flag_document
document1=pd.melt(document1,id_vars=['sk_id_curr'],var_name='document_code',v
con_document = pd.merge(document1,document_df)[['sk_id_curr','document_id']]
con_document.to_sql(name='apl_doc_conn', con=engine, if_exists='append', inde
```

In [76]:
```
#occupation
occupation_df = df.loc[:, ('sk_id_curr','occupation_type','days_employed')]
occupation_df = occupation_df.drop_duplicates()
```

```
occupation_df.insert(0, 'occupation_id', range(1, 1 + len(occupation_df)))
occupation_df.to_sql(name='occupation', con=engine, if_exists='append', index=
```

In [78]:
```
#contact
contact_df = df.loc[:, ('sk_id_curr','flag_mobil','flag_emp_phone','flag_work
                        ,'flag_email')]
contact_df= contact_df.melt(id_vars = ['sk_id_curr'])
contact_df = contact_df.rename(columns = {'variable': 'contact_type', 'value'
contact_df = contact_df.drop_duplicates()
contact_df.insert(0, 'contact_id', range(1, 1 + len(contact_df)))
contact_df.to_sql(name='contact', con=engine, if_exists='append', index=False
```

In [25]:
```
#target
target_df = df[['target']]
con_target = df [['sk_id_curr','target']]
target_df.rename(columns={'target':'target_name'},inplace = True)
con_target.rename(columns={'target':'target_name'},inplace = True)
target_df = target_df.drop_duplicates()
target_df.insert(0, 'target_id', range(1, 1 + len(target_df)))
target_df0 = pd.merge(target_df,con_target)[['sk_id_curr','target_id','target_
target_df0.to_sql(name='target', con=engine, if_exists='append', index=False)
```

In [22]:
```
#suite
suite_df = df[['name_type_suite']]
suite_df.rename(columns={'name_type_suite':'suite_name'},inplace = True)
suite_df = suite_df.drop_duplicates()
suite_df.insert(0, 'suite_id', range(1, 1 + len(suite_df)))
suite_df.to_sql(name='suite', con=engine, if_exists='append', index=False)
```

```
/Users/weiyini/opt/anaconda3/lib/python3.8/site-packages/pandas/core/frame.py:
5039: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().rename(
```

In [24]:
```
#contract
contract_df = df[['sk_id_curr','name_contract_type','amt_credit','amt_annuity
contract_df.rename(columns={'name_contract_type':'contract_name','weekday_app
                            'name_type_suite':'suite_name','target':'target_na
contract_df.drop_duplicates()
contract_df.insert(0, 'contract_id', range(1, 1 + len(contract_df)))
contract_df1 = pd.merge(contract_df,target_df)
contract_df2 = pd.merge(contract_df1,suite_df)
contract_df0 = contract_df2[['sk_id_curr','contract_id','contract_name','amt_
contract_df0.to_sql(name='contract', con=engine, if_exists='append', index=Fa
```

```
/Users/weiyini/opt/anaconda3/lib/python3.8/site-packages/pandas/core/frame.py:
5039: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().rename(
```

In [3]:
```
stmt = """
drop table address_match, amt_req_credit_bureau, building_info, contact, cont
family, gender, income, apl_obs_conn, apl_etn_conn, apl_reg_conn, loan_applic
region, suite, target;
"""
connection.execute(stmt)
```

Out[3]:
```
<sqlalchemy.engine.result.ResultProxy at 0x7fb5d5f63cd0>
```

In [ ]: