Jannells Car Sales

By: The Office

Outline

- Business Objective
- Data Sources
- Data Ingestion
- Database Design
- SQLAlchemy
- SQL Transformation
- Data Validation
- Final Product

Business Objective

- This .csv contains all clients who have updated their preference for either phone or email contact
  - This a snapshot up to June 10th of this year
  - This data is easy to update and redistribute
  - Able to provide other ad hoc reports such as
    - Clients who have not updated their outreach preference
    - Clients who only declined/allowed email or calls

Client_Contact_Preference_06102025.csv

	client_name text	company_name character varying (128)	sales_rep_name text	client_phone_permission boolean	client_phone character (30)	client_email_permission boolean	client_email character varying (128)
1	Angela Henry	Jones-Davis	Melanie Bell	false	974-577-4293x...	false	xwalters@example.com
2	Anna Walker	Schultz Inc	Jonathon Cox	false	939-801-6350x...	false	david92@example.org
3	Andrea Wyatt	Johnson-Farley	Michelle Ferguson	true	667-606-356 ...	true	brenda52@example.net
4	Anthony Johnson	Snyder PLC	Joseph Fox	true	+1-962-586-84...	true	joseph64@example.net
5	Angel Gonzalez	Perry-Sellers	Lauren Morales	false	+1-992-952-51...	false	jjohnson@example.com
6	Amber Scott	Thompson-Collins	Stephen Shah	true	+1-595-639-87...	true	vserrano@example.org
7	Adam Baker	Shelton and Sons	Michele Davis	true	+1-953-708-16...	true	raymondrodriguez@example.org
8	Amy Summers	Stevens-Fox	Mrs. Danielle Ortiz	true	(903)308-0626...	[null]	michelle25@example.net
9	Andrew Ford	Shaw-Johnson	Kenneth Lucas	true	971823170 ...	true	traci24@example.org

Data Ingestion

- API authentication and token management
- Data retrieval by pages
- Prepare for insertion : making tabular
- Loading to PostgreSQL database

```python
def load_header(sample_data, table_name, engine):
    data_df= pd.DataFrame(sample_data)
    data_df.head(0).to_sql(name=table_name, con=engine, if_exists='replace',
    index=False)  # create table with headers
```

Some implementations:

- Token management with "***dotenv***"
- Automatic token refreshes if expired
- Automatically creates database tables based on API response
- Large dataset are loaded in chunks

```python
def load_all(table_name:str, engine):
    is_data = None
    sample_data = get_data(0,5,table_name)
    if sample_data:
        is_data = True
        load_header(sample_data, table_name, engine)
    offset = 0
    limit = 10
    while is_data:
        data_queried = get_data(offset, limit, table_name)
        if not data_queried:
            is_data = False
        else:
            json_to_postgres(data_queried,table_name,engine)
            offset += limit


    return data
```

Data Ingestion

Load csv with pandas.read_csv()

- Read the csv as given
- Read in chunks
- Load the header to the database
- Append the rest
- Do this within loop for large csv files

- Loading as an iterator object allows chunk loading a csv.
- Better if the csv grows in size

```
def load_csv(filepath:str):
    df_iter = pd.read_csv(filepath, iterator=True, chunksize=100)
    df = next(df_iter)  # first chunk
    table_name = 'client_contact_status'
    df.head(n=0).to_sql(name=table_name, con=engine, if_exists='replace')  #
    header first
    df.to_sql(name=table_name, con=engine, if_exists='append')
```

Data Sources

- API with endpoints /people and /clients returns data in JSON format

Practice API:

Base URL: (*https://developyr-api.azurewebsites.netapi*)

- Endpoints :
  - /auth : for authentication:
    - Requires POST method
    - Requires username and password
    - Generates JWT token for authentication
  - /people : returns a list of people
    - Requires GET
    - Requires parameters (OFFSET (0) and LIMIT(10)
  - /clients : returns a list of clients
    - Requires GET
    - Same query parameters

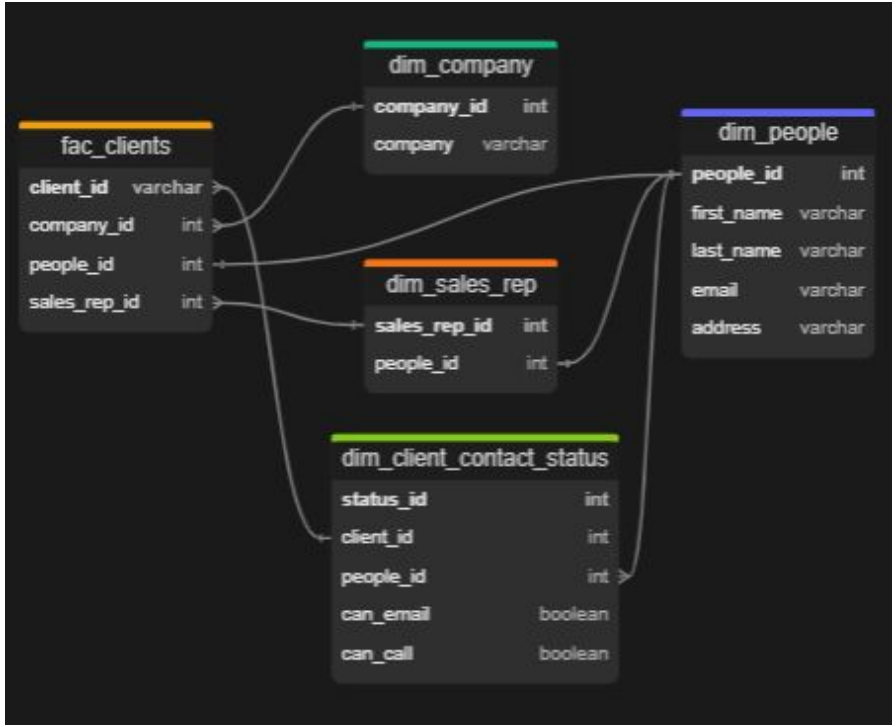
CSV: obtained from (a) CRM software.

- Has null values

```
→ data
[{'first_name': 'Tammy', 'last_name': 'Alexander', 'email': 'davissherri@
example.net', 'address': '53000 Wong Orchard\nHernandezburgh, VT 15497'},
> {'first_name': 'Warren', 'last_name': 'Anderson', 'email': 'denise09@exam
ple.com', 'address': '6653 Smith Island Suite 189\nPort Jefferyshire, OR
48919'}]
```

Database Design

1. Star Schema
2. Sales_rep table
3. Company table



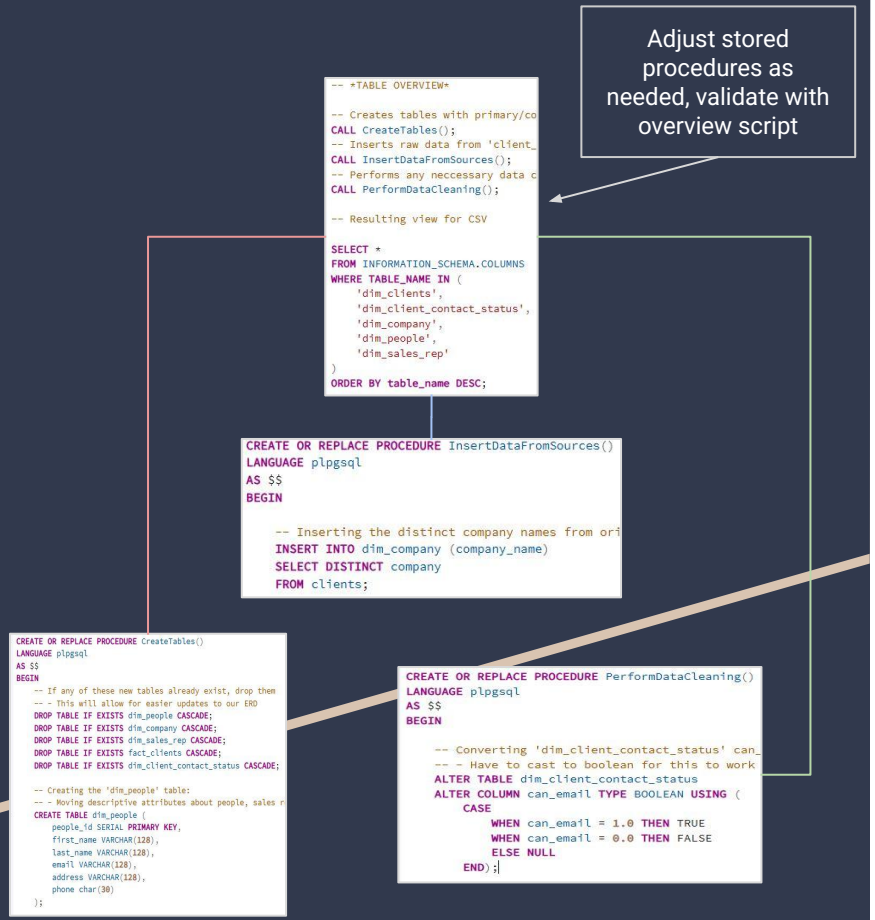
SQLAlchemy

- SQLAlchemy is a library in Python. In this case the purpose of this code is to connect the engine to a postgres database. SQLAlchemy is the library used to manage this connection to the database.

```
if port is None:
    port = '5432'  # default port

# Using the new psycopg driver name in SQLAlchemy URL
engine = create_engine(f'postgresql+psycopg://{user}:{password}@{host}:{port}/{db}')

#connect and run a test query
with engine.connect() as conn:
    result = conn.execute(text("SELECT version()"))
version = result.scalar()
print(version)
print("PostgreSQL version:", version[0])

#load_all('people', engine)
#load_all('clients', engine)
```

SQL Transformation

Adjust stored procedures as needed, validate with overview script



1. Table Creation
    a. CreateTables() Stored Procedure
        i. Creates required tables and adds PK/FK constraints to fulfill ERD
2. Data Insertion
    a. InsertDataFromSources() Stored Procedure
        i. Inserts data from original data sources extracted via Python
3. Data Cleaning
    a. PeformDataCleaning() Stored Procedure
        i. Cleans/transforms data to fulfill 3NF and prepare for creation of view

- Organizing our queries into 3 main stored procedures made our tables easy to maintain
    - The separation of logic allowed for us to test and validate at each stage of transformation

Data Validation

Database Validation

- We made sure that the data was cleaned, stored and consistent.
- Verifying that the data types were correct in order to avoid errors when merging the datasets.

Post Processing validation

- We made sure that the data was cleaned, stored and consistent.
- Verifying that the data types were correct in order to avoid errors when merging the datasets.

```
offset = 0
limit = 2
chunks = pd.read_sql_table('clients', con=engine, chunksize=limit)
for chunk in chunks:
    data_pulled_db = chunk.to_dict(orient='records')
    print(data_pulled_db)
    data_api = get_data(offset,limit,'clients')
    print(data_api)
    if data_pulled_db == data_api:
        print("Data matches between database and API.")
    else:
        print("Data mismatch between database and API.")
    offset += limit
```

To Do:

- Incorporate this to the queried table from the finalized database
- Extend to csv data

Final Product :

client_contact_status_view:

	client_name text	company_name character varying (128)	sales_rep_name text	client_phone_permission boolean	client_phone character (30)	client_email_permission boolean	client_email character varying (128)
1	Adam Baker	Shelton and Sons	Michele Davis	true	+1-953-708-1654x651	true	raymondrodriguez@example.org
2	Amber Scott	Thompson-Collins	Stephen Shah	true	+1-595-639-879	true	vserrano@example.org
3	Amy Summers	Stevens-Fox	Mrs. Danielle Ortiz	true	(903)308-0626x176	[null]	michelle25@example.net
4	Andrea Wyatt	Johnson-Farley	Michelle Ferguson	true	667-606-356	true	brenda52@example.net
5	Andrew Ford	Shaw-Johnson	Kenneth Lucas	true	971823170	true	traci24@example.org
6	Angel Gonzalez	Perry-Sellers	Lauren Morales	false	+1-992-952-5180x43	false	jjohnson@example.com
7	Angela Henry	Jones-Davis	Melanie Bell	false	974-577-4293x1278	false	xwalters@example.com
8	Anna Walker	Schultz Inc	Jonathon Cox	false	939-801-6350x0292	false	david92@example.org
9	Anthony Johnson	Snyder PLC	Joseph Fox	true	+1-962-586-8498x7986	true	joseph64@example.net
10	Ashley Mccarty	Sanchez-Woodward	Loretta Clark	[null]	(448)746-781	[null]	barreravirginia@example.net
11	Brandy Jenkins	Rogers-Jones	Jose Mercer	[null]	761-851-0125x42	[null]	sandrawebb@example.org

clients_can_call_06092025.csv

	client_name text	company_name character varying (128)	sales_rep_name text	client_phone_permission boolean	client_phone character (30)
1	Andrea Wyatt	Johnson-Farley	Michelle Ferguson	true	667-606-356 …
2	Anthony Johnson	Snyder PLC	Joseph Fox	true	+1-962-586-84…
3	Amber Scott	Thompson-Collins	Stephen Shah	true	+1-595-639-87…
4	Adam Baker	Shelton and Sons	Michele Davis	true	+1-953-708-16…
5	Amy Summers	Stevens-Fox	Mrs. Danielle Ortiz	true	(903)308-0626…
6	Andrew Ford	Shaw-Johnson	Kenneth Lucas	true	971823170 …

clients_do_not_email.csv

	client_name text	company_name character varying (128)	sales_rep_name text	client_email_permission boolean	client_email character varying (128)
1	Angela Henry	Jones-Davis	Melanie Bell	false	xwalters@example.com
2	Anna Walker	Schultz Inc	Jonathon Cox	false	david92@example.org
3	Angel Gonzalez	Perry-Sellers	Lauren Morales	false	jjohnson@example.com

- This view contains all columns relevant to business objectives
- From here, you can query specifically for clients who:
  - Granted permission to be called and/or emailed
  - Refused permission for either
  - Have yet to grant or refuse permission
- For this project, we have provided “clients_contact_preference_06102025.csv”
  - This .csv contains all clients who have updated their preference for either calls or emails

Questions


DWIGHT SCHRUTE
DETERMINED
WORKER
INTENSE
GOOD WORKER
HARD WORKER
TERRIFIC