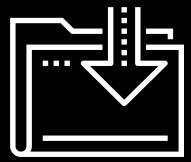
Project 2:
Extract, Transform, and Load

Data Boot Camp

Lesson 13.1




WELCOME

The Week Ahead

Day 1 (Today)	Introduction to ETL
	Introduction to the ETL project (goals and requirements)
	Develop feasible project idea (with the help of instructors and TAs)
	Submit project proposal
Day 2	Work on projects with the assistance instructors and TAs.
Day 3	Projects due!

Instructor Demonstration

Introduction to the Case Study Project

Case Study Project Requirements

Data sources

You must have at least two sources:	Recommended sources include: • Kaggle • Data.world • Google Dataset Search (https://datasetsearch.research.google.com/) • APIs may be used as an alternative source
Once your datasets are identified:	Perform the ETL process and create your documentation. Your documentation must include: • Datasets used and their sources • Types of data wrangling performed (data cleaning, joining, filtering, aggregating) • The schemata used in the final production database

Instructor Demonstration

Introduction to ETL

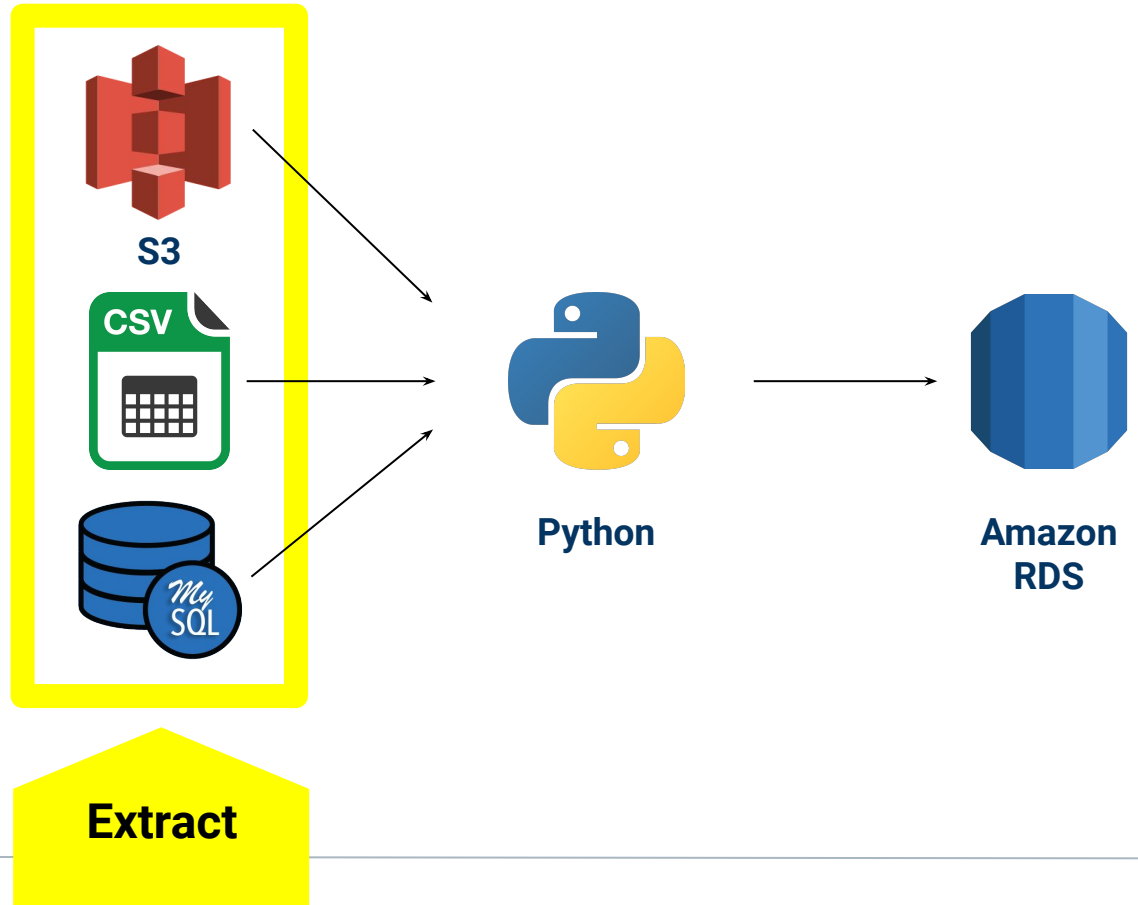
Introduction to ETL

ETL: Extract, Transform, and Load

Introduction to ETL: **Extract**

Data may come from disparate sources, such as:

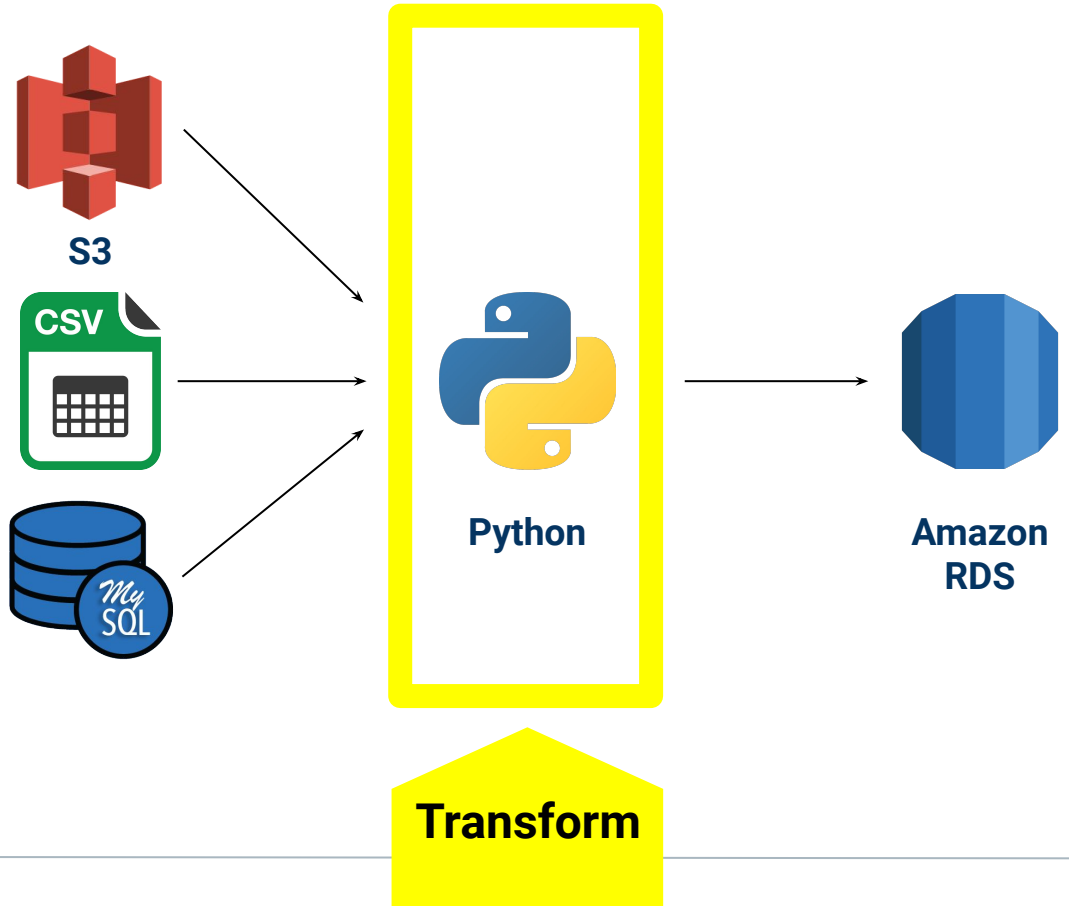
- CSV files
- JSON files
- HTML tables
- SQL databases
- Spreadsheets



Introduction to ETL: **Transform**

Transform the data to suit business needs, including:

- Data cleaning
- Summarization
- Selection
- Joining
- Filtering
- Aggregating

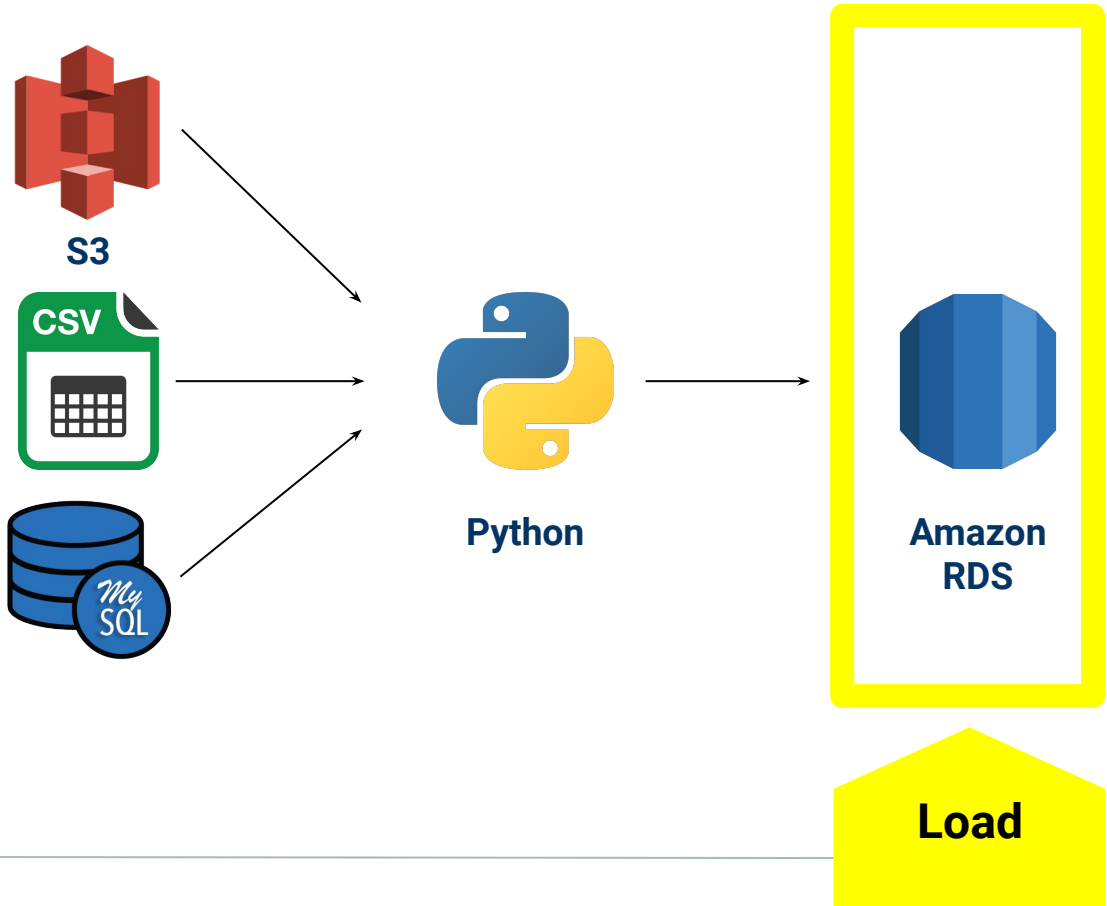


Note: We will use Python and Pandas for transformation, which can also be done with SQL or a specialized ETL tool.

Introduction to ETL: Load

Load the data into a final database that can be used for future analysis or business applications:

- Can be a relational or non-relational database
- Can be local or in the cloud
- Can be a data lake or data warehouse



Questions?

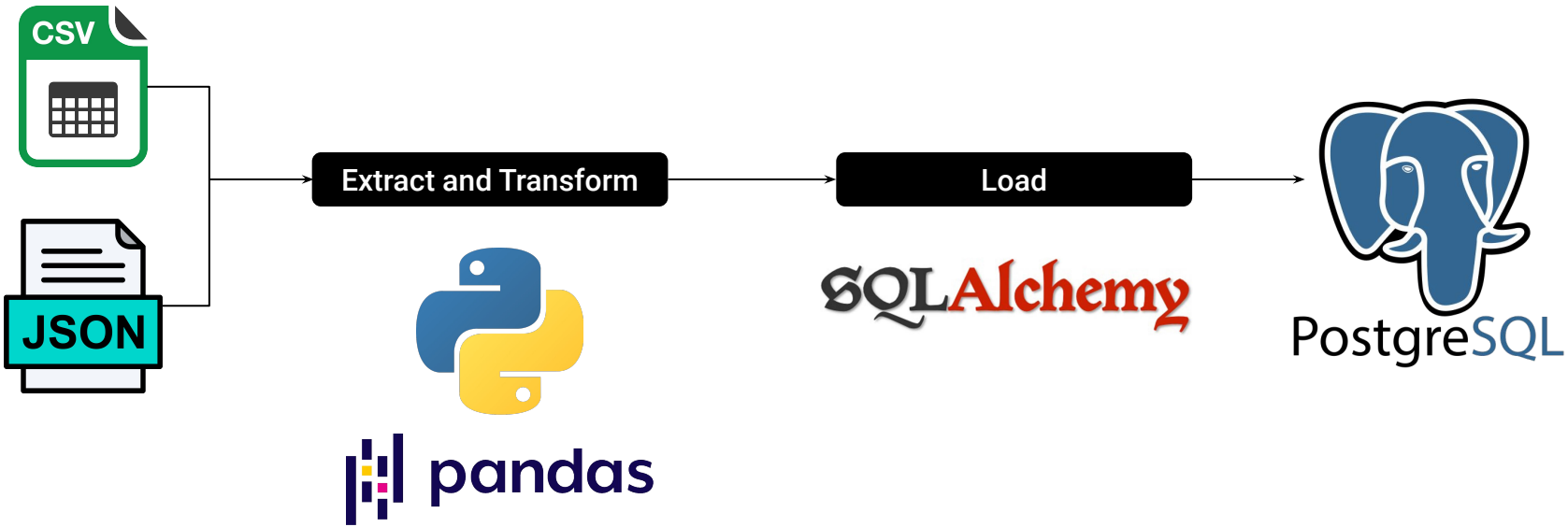
Instructor Demonstration

ETL with Pandas

ETL Pipeline Using Python, Pandas, and SQLAlchemy

The ETL process is performed in a variety of ways.

For this demonstration, we will use the following ETL pipeline.



We have a few things to prepare before we can proceed.
Let’s find out what they are.

ETL Setup

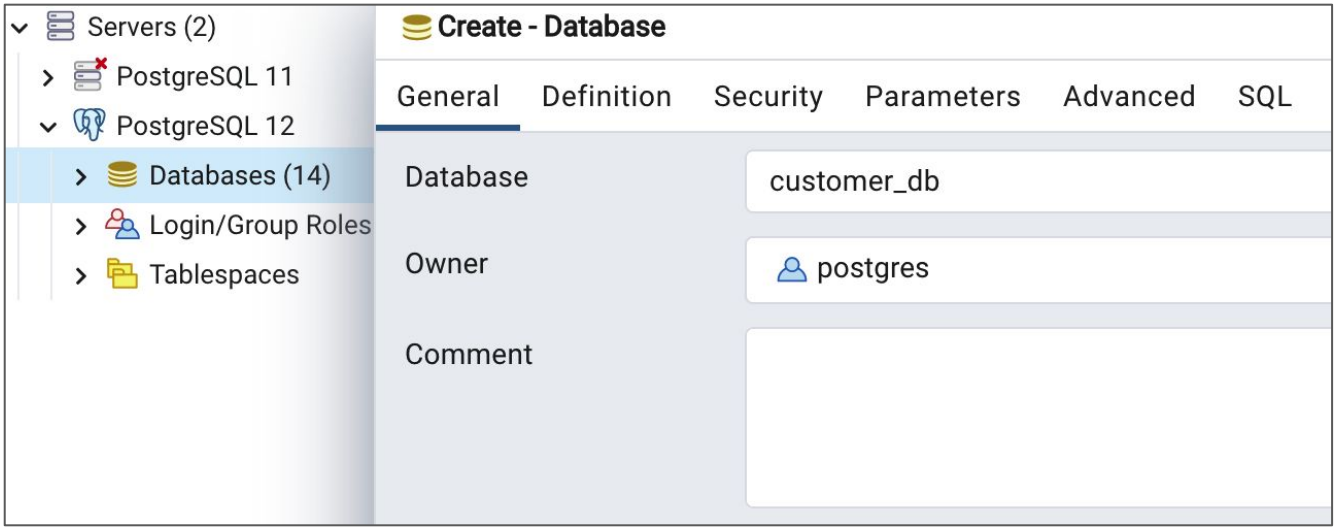
pip install psycopg2

```
pip install psycopg2
```

Psycopg is a package—
an adapter for Python that works as a wrapper for **libpq**,
which is the official PostgreSQL client library.

ETL: Create a Database

pgAdmin PostgreSQL
Next, we need to open pgAdmin and connect to a local server and create a new **database**, `customer_db`.



ETL: Create Table Schema

Then, we create our **tables**.

pgAdmin PostgreSQL

Query Editor | Explain | Query History

```
1  -- Create tables for raw data to be loaded into
2  CREATE TABLE customer_name (
3  id INT PRIMARY KEY,
4  first_name TEXT,
5  last_name TEXT
6  );
7
8  CREATE TABLE customer_location (
9  id INT PRIMARY KEY,
10 address TEXT,
11 us_state TEXT
12 );
```

ETL: Extract

During the ETL process we use Pandas to Extract the data, Transform the data, and then Load the DataFrames into the postgreSQL tables.

First, we extract the data.

Store CSV into DataFrame

```
csv_file = "../Resources/customer_data.csv"
customer_data_df = pd.read_csv(csv_file)
customer_data_df.head()
```

	id	first_name	last_name	email	gender	car
0	0	Jeff	Gregory	johnsonmichael@example.net	MALE	Buick
1	1	Robin	Johnson	ernest54@example.net	FEMALE	Lexus
2	2	Shannon	Thompson	jmathis@example.com	FEMALE	Honda
3	3	Sandy	Collier	thomaskaren@example.com	FEMALE	Toyota
4	4	Morgan	Simpson	allendakota@example.org	FEMALE	Dodge

Store JSON data into a DataFrame

```
json_file = "../Resources/customer_location.json"
customer_location_df = pd.read_json(json_file)
customer_location_df.head()
```

	id	address	longitude	latitude	us_state
0	0	01487 Allen Point Apt. 315	-77.0092	38.8301	DC
1	1	119 Woods Meadows Suite 838	-71.1960	43.0320	NH
2	2	84269 Harold Knoll Apt. 388	-83.1734	32.9053	GA
3	3	772 John Roads Apt. 017	-98.2423	47.4709	ND
4	4	186 Peterson Land Apt. 060	-97.5300	34.6300	OK

ETL: Transform

Next, we transform the data.

Create new data with select columns

```
new_customer_data_df = customer_data_df[['id', 'first_name', 'last_name']].copy()
new_customer_data_df.head()
```

	id	first_name	last_name
0	0	Jeff	Gregory
1	1	Robin	Johnson
2	2	Shannon	Thompson
3	3	Sandy	Collier
4	4	Morgan	Simpson

Clean DataFrame

```
new_customer_location_df = customer_location_df[["id", "address", "us_state"]].copy()
new_customer_location_df.head()
```

	id	address	us_state
0	0	01487 Allen Point Apt. 315	DC
1	1	119 Woods Meadows Suite 838	NH
2	2	84269 Harold Knoll Apt. 388	GA
3	3	772 John Roads Apt. 017	ND
4	4	186 Peterson Land Apt. 060	OK

ETL: Load

Next, we connect to the local database. Once connected, we check for the tables that we created earlier in the process.

Then, we dump the newly created and trimmed DataFrames into the database.

Connect to local database

```
rds_connection_string = "<INSERT USERNAME>:<INSERT PASSWORD>@localhost:5432/customer_db"
engine = create_engine(f'postgresql://{rds_connection_string}')
```

Check for tables

```
engine.table_names()
```

```
['customer_name', 'customer_location']
```

Use pandas to load csv converted DataFrame into database

```
new_customer_data_df.to_sql(name='customer_name', con=engine, if_exists='append', index=False)
```

Use pandas to load json converted DataFrame into database

```
new_customer_location_df.to_sql(name='customer_location', con=engine, if_exists='append', index=False)
```

ETL

At this point, all the data that we extracted and transformed are successfully loaded into our PostgreSQL database.

To double-check, as a best practice, we perform queries for both tables at the database.

Confirm data has been added by querying the customer_name table

- NOTE: can also check using pgAdmin

```
pd.read_sql_query('select * from customer_name', con=engine).head()
```

	id	first_name	last_name
0	0	Jeff	Gregory
1	1	Robin	Johnson
2	2	Shannon	Thompson
3	3	Sandy	Collier
4	4	Morgan	Simpson

Confirm data has been added by querying the customer_location table

```
pd.read_sql_query('select * from customer_location', con=engine).head()
```

	id	address	us_state
0	0	01487 Allen Point Apt. 315	DC
1	1	119 Woods Meadows Suite 838	NH
2	2	84269 Harold Knoll Apt. 388	GA
3	3	772 John Roads Apt. 017	ND
4	4	186 Peterson Land Apt. 060	OK

ETL

Finally, we come back to pgAdmin and check the data in both tables.

```
1  SELECT * FROM customer_name;
2
3
```

Notifications | Messages | Data Output

	id [PK] integer	first_name text	last_name text
1	0	Jeff	Gregory
2	1	Robin	Johnson
3	2	Shannon	Thompson
4	3	Sandy	Collier
5	4	Morgan	Simpson

```
1  SELECT * FROM customer_location;
2
3
```

Notifications | Messages | Data Output

	id [PK] integer	address text	us_state text
1	0	01487 Allen Point Apt. 315	DC
2	1	119 Woods Meadows Suite 838	NH
3	2	84269 Harold Knoll Apt. 388	GA
4	3	772 John Roads Apt. 017	ND

ETL

Then, join the two tables in pgAdmin.

```
1  SELECT customer_name.id,
2      customer_name.first_name,
3      customer_name.last_name,
4      customer_location.address,
5      customer_location.us_state
6  FROM customer_name
7  JOIN customer_location
8  ON customer_name.id = customer_location.id;
9
```

Notifications | Messages | Data Output

	id integer	first_name text	last_name text	address text	us_state text
1	0	Jeff	Gregory	01487 Allen Point Apt. 315	DC
2	1	Robin	Johnson	119 Woods Meadows Suite 838	NH
3	2	Shannon	Thompson	84269 Harold Knoll Apt. 388	GA
4	3	Sandy	Collier	772 John Roads Apt. 017	ND
5	4	Morgan	Simpson	186 Peterson Land Apt. 060	OK

ETL

Or, join the two tables in with Pandas and SQLAlchemy.

```
sql_join = r"""SELECT customer_name.id,
customer_name.first_name, customer_name.last_name,
customer_location.address, customer_location.us_state
FROM customer_name
JOIN customer_location
ON customer_name.id = customer_location.id"""
```

```
pd.read_sql_query(sql_join, con=engine).head()
```

	id	first_name	last_name	address	us_state
0	0	Jeff	Gregory	01487 Allen Point Apt. 315	DC
1	1	Robin	Johnson	119 Woods Meadows Suite 838	NH
2	2	Shannon	Thompson	84269 Harold Knoll Apt. 388	GA
3	3	Sandy	Collier	772 John Roads Apt. 017	ND
4	4	Morgan	Simpson	186 Peterson Land Apt. 060	OK

Activity: Pandas ETL

In this activity, you will perform your very first ETL process!

Suggested Time:

20 minutes

Activity: Pandas ETL

Instructions	Create a `customer_db database` in pgAdmin 4, and then create the following two tables within the database: • A premise table that contains the columns `id`, `premise_name`, and `county_id`. • A county table that contains the columns `id`, `county_name`, `license_count`, and `county_id`. • Be sure to assign a primary key, as Pandas will not be able to do so.
	In Jupyter Notebook, perform all ETL steps.
Extract	Put each CSV into a Pandas DataFrame.
Transform	Copy only the columns needed into a new DataFrame.
	Rename columns to fit the tables created in the database.
	Handle any duplicates. **Hint:** Some locations have the same name, but each license number is unique.
	Set the index to the previously created primary key.

Activity: Pandas ETL

Instructions	Continued
Load	Create a connection to the database.
	Check for a successful connection to the database, and confirm that the tables have been created.
	Append DataFrames to tables. Be sure to use the index set earlier.
FInal Steps	Then, we perform the following final steps: • Confirm a successful **load** by querying the database. • Join the two tables, and select the `id` and `premise_name` from the `premise` table and `county_name` from the `county` table.

Time’s Up! Let’s Review.


Break

Project Overview

Project Week (This Week)!

Day 1	Form groups
	Identify datasets
	Perform ETL on the data
Day 2	Develop database
Day 3	Complete final report

Project Proposals

Team effort

- Due to the brief timeline, teamwork will be crucial to your success!
- Work closely with your team through all phases of the project to ensure that there are no surprises at the end of the week.
- Working in a group enables you to address more difficult problems than you'd be able to manage on your own.
- Take advantage by working smart and dreaming big!

Project proposal

- Before you write any code, remember that you only have one week to complete this project.
- Think of this project like a typical work assignment. Imagine that a bunch of data came in, and you and your team have been tasked with migrating it to a production database.
- Try to take advantage of instructor and TA support during office hours and in-class project time.

Project 2: ETL

Data Cleanup and Analysis Requirements

Teams will be responsible for:

- Citing the data sources
- Extracting the data from those sources
- Transforming the data (cleaning, joining, filtering, aggregating, etc.)
- Loading the data into a database (relational or non-relational)

Report Requirements

You will also prepare a report to address the following points:

Extract

Your original data sources and how the data were formatted (CSV, JSON, pgAdmin 4, etc.)

Transform

What data cleaning or transformation was required

Load

The final database, tables/collections, and why this was chosen.

Project Rubric

Rubric Summary

Grading Categories

Project proposal	(20 points)
Technical report	(20 points)
GitHub repository	(20 points)

Data Suggestions

Data Suggestions

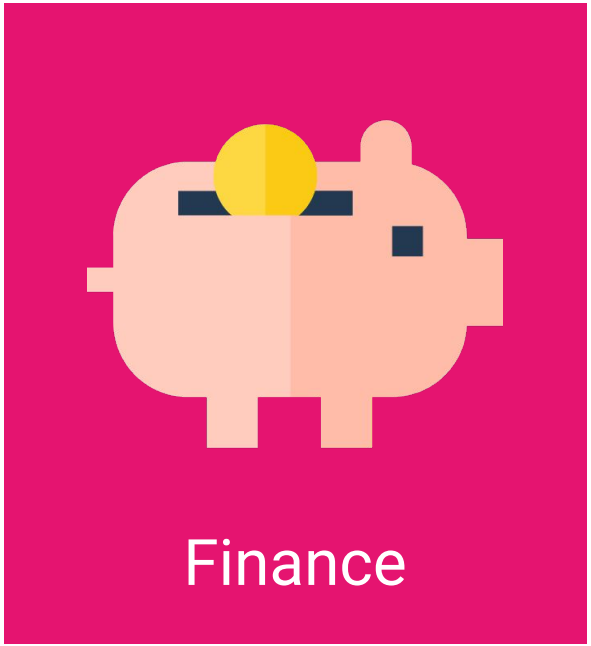
Feel free to ask us for input, but our general advice is to use data sources that:

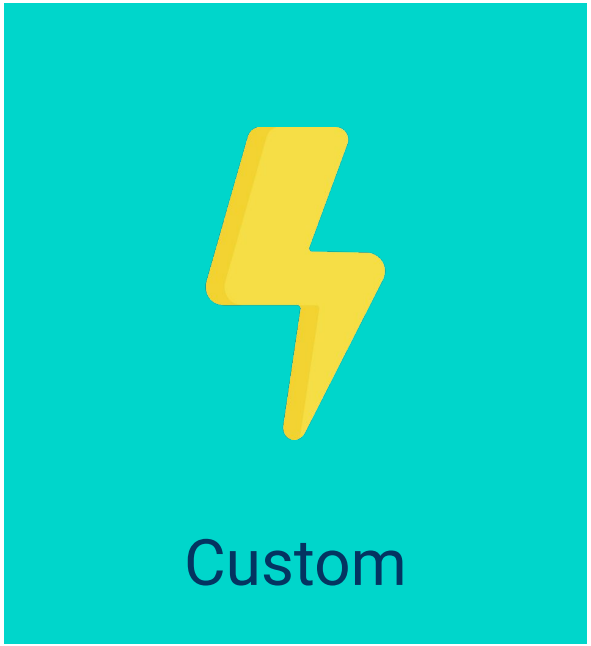
- Are sufficiently large
- Have a consistent format
- Ideally, contain more data than we need
- Are well documented

Choosing a Project Track

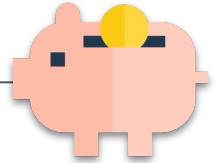
Choosing a Project Track

For this project, you can focus your efforts on a specific industry, including the following specializations:





ETL and Finance



When to use ETL in finance

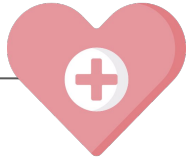
Current treasury benchmarks are at an all-time low, and a financial analyst has decided to study the last 30 years worth of rates.

After pulling historical data, the analyst cleans and explores the data to perform their analysis, with the intent of predicting future benchmark trends.

Once the historical data have been collected, processed, and loaded into a database, the financial analyst turns their attention to present-day data. Using an API, they pull the most up-to-date information so it can be added to their established database.

They've already extracted the new data, but before loading them into the existing database, they need to ensure that they have the correct format. Once the data are transformed, they can load them to the database and continue with the analysis.

ETL and Healthcare



When to use ETL in healthcare

An analyst working at a major hospital is tasked with reviewing policies regarding the upcoming flu season. The analyst is keeping the following questions in mind:

- How many patients does the hospital expect this year?
- How severe will flu season be this year?
- Will there be regional differences? Similarities?

The analyst wants to collect and analyze data from different sources so they can make predictions about the upcoming flu season.

Before combining the hospital's own data with regional data acquired externally, the analyst will need to extract the new data, transform them to match the existing data, and then load them into the database.

ETL in the Wild

Several other industries use the ETL process, as well.
Customize it!

In marketing, analysts may acquire data from competitors to see how their products measure up. Multiple data sources would need to be extracted, transformed, and loaded into a common database prior to analysis.

An analyst working for a large retail chain is in charge of moving a legacy database into a cloud-based data warehouse.

An entrepreneur has a big business idea but wants to get a feel for their product idea. hey use web scraping and APIs to pull data from a variety of social media platforms with the intent of analyzing consumer reactions.

Questions?

Questions?


The End