TEAM 3

ETL PROJECT

Amanda Pesch, Chloe Lee, Dave Mueller,
John Burke, Jordan Cizmja, Rna Babikar

Our ETL Approach

EXTRACT

Covid Infection Rates (CSV)

Vaccination Rates (JSON)

TRANSFORM

Pandas Profiling

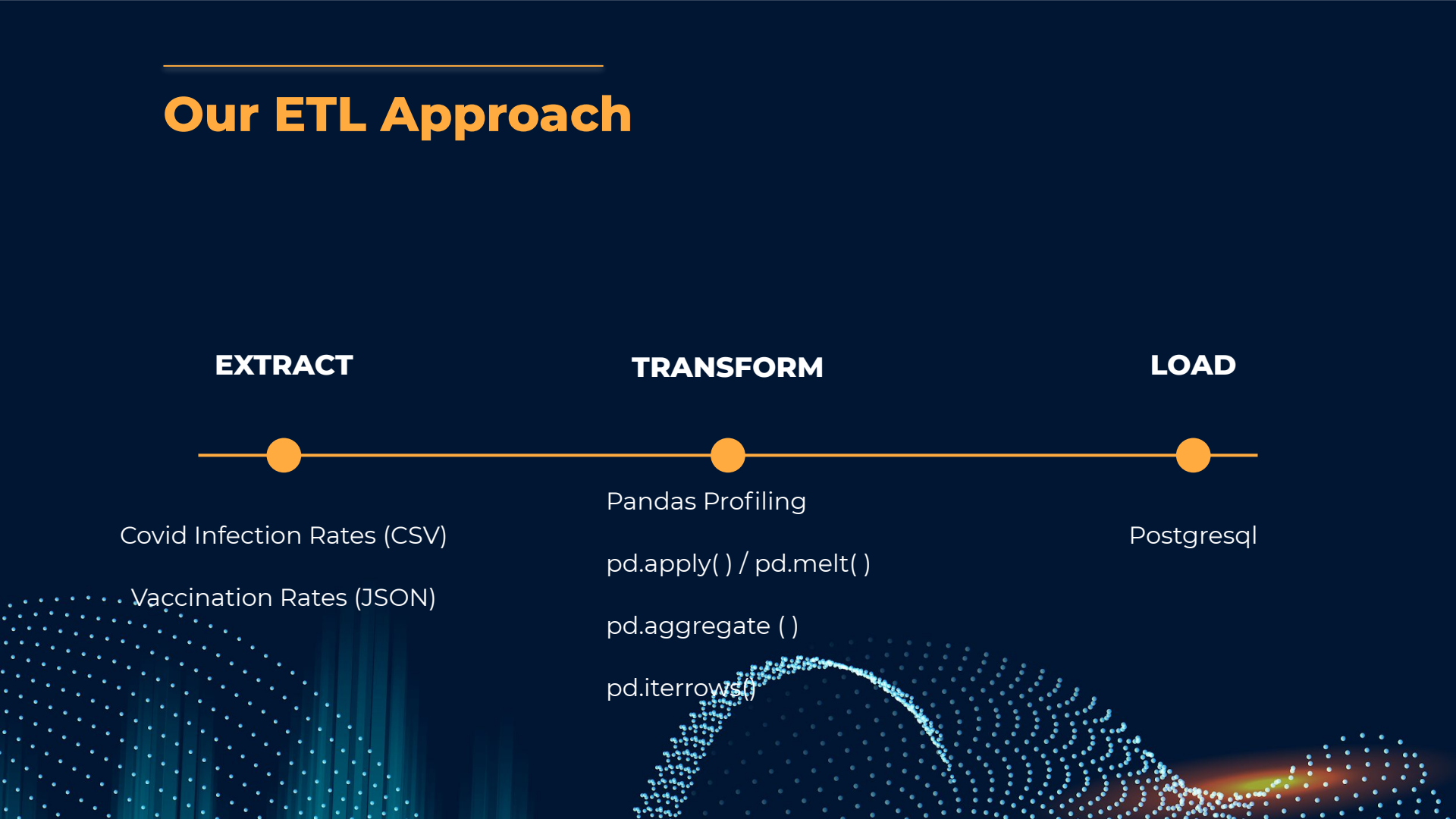
pd.apply( ) / pd.melt( )

pd.aggregate ( )

pd.iterrows()

LOAD

Postgresql



EXTRACT

COVID-19 Case Data (1)

Johns Hopkins Center for Systems Science & Engineering

GitHub page **csv** (time_series_covid19_confirmed_US)

- Confirmed cases in the US only
- Updated once a day around 23:59 (UTC)

EXTRACT

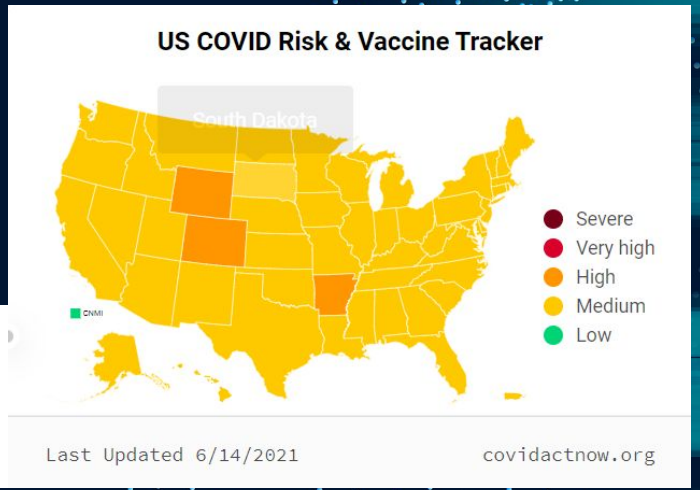
COVID-19 Case Data (2)

COVID ACT NOW

COVID ACT NOW : Realtime US Covid-19 Map & Vaccine Tracker

Using API → **JSON** (State & County Info.)
(https://api.covidactnow.org/v2/counties.json?apiKey=)

- Confirmed cases in the US only
- Data is updated daily around noon (EST)



TRANSFORM

COVID-19 Case Data

Johns Hopkins Center for Systems Science & Engineering

- **pandas.melt function** to unpivot data in a DataFrame
- **Unpivot:** transpose columns into rows, changing from wide to long format
- **When to use:** when there are two data types in one column (column header data type differs from the column value data) or to better organize a DataFrame to save space
- Link to .melt documentation

Before (prepping the data frame use .melt()):

```
# Creates new DF with just county and dates

prep_unpivot_df = mi_covid_df.drop(labels=['FIPS','State'], axis=1)
prep_unpivot_df.head()
```

	County	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20	1/29/20	1/30/20	...	6/1/21	6/2/21
0	Alcona	0	0	0	0	0	0	0	0	0	...	738	738
1	Alger	0	0	0	0	0	0	0	0	0	...	515	515
2	Allegan	0	0	0	0	0	0	0	0	0	...	11092	11099

TRANSFORM

COVID-19 Case Data

Johns Hopkins Center for Systems Science & Engineering

- **.melt() parameters**:
  - id_vars = Unpivoted by county name
  - var_name = column header name for the columns to be unpivoted
  - value_name = column header name for the values

After:

```
df_unpivoted = prep_unpivot_df.melt(id_vars=['County'], var_name='Date', value_name='Cases')
df_unpivoted.tail(3)
```

	County	Date	Cases
44193	Washtenaw	6/12/21	26439
44194	Wayne	6/12/21	165215
44195	Wexford	6/12/21	2890

TRANSFORM

COVID-19 Case Data

Johns Hopkins Center for Systems Science & Engineering

- **pandas.nlargest()** find county with largest case counts overall
- **pandas.loc()** on that county
- **pandas.diff()** find largest daily change in case numbers
- **pandas.pct_change()** % change in those case numbers

```
# Displays county, dates, and cases with largest case counts
top_df = df_unpivoted.nlargest(50, 'Cases', keep='first')
# Confirmed - it was Wayne county
# Now dive deeper - top 10 largest instances of daily change from one day to the next
wayne_df = df_unpivoted.loc[df_unpivoted['County']=='Wayne']
wayne_df = wayne_df.set_index('County')
wayne_df['Cases_diff'] = wayne_df['Cases'].diff()
wayne_top_df = wayne_df.nlargest(10, 'Cases_diff', keep='first')
wayne_top_df['Pct_change'] = wayne_top_df['Cases'].pct_change().astype(float).map("{:.2%}".format)
# TABLE wayne_top_df
wayne_top_df
```

County	Date	Cases	Cases_diff	Pct_change
Wayne	4/5/21	125364	2393.0	nan%
Wayne	11/27/20	62273	2368.0	-50.33%
Wayne	4/13/21	137370	2123.0	120.59%

TRANSFORM

COVID-19 Case Data

Johns Hopkins Center for Systems Science & Engineering

- pandas.iterrows() function
  - Did a str.split() to split the date column into Month, Day, & Year columns
  - Used iterrows to convert “4” to “April”
  - Then merged to get “Month YYYY”

```
df_unpivoted[['Month','Day','Year']] = df_unpivoted['Date'].str.split("/", expand = True)

df_unpivoted = df_unpivoted.loc[df_unpivoted['Month']=='4']
df_unpivoted

month_dict = {
    '4': 'April'
}

for index,row in df_unpivoted.iterrows():
        april_df = df_unpivoted.replace({"Month": month_dict})
april_df
```

	County	Date	Cases	Month	Day	Year
6090	Alcona	4/1/20	0	April	1	20
6091	Alger	4/1/20	0	April	1	20
6092	Allegan	4/1/20	5	April	1	20
6093	Alpena	4/1/20	0	April	1	20

```
april_df["Month Year"] = april_df["Month"] + " 20" + april_df["Year"]
april_df = april_df.sort_values('County')
april_final_df = april_df.drop(labels= ['Date','Month','Day','Year'], axis=1)
april_final_df
```

	County	Cases	Month Year
6090	Alcona	0	April 2020
38715	Alcona	565	April 2021

TRANSFORM

COVID-19 Case Data

COVID ACT NOW

- **Pandas.apply function :** Apply a function along an axis of the DataFrame
- (axis = 0 : apply function to columns, axis = 1 : apply function to each rows)

In [7]:
```
# Using .apply() fuction to get vaccinate completion rate
vaccination_df["Completion"] = vaccination_df.apply(
    lambda x: x['Vaccination Completed']/x['Population'], axis =1)

# Change formattingPo
format_dict = {'Completion': '{:.1%}'}
vaccination_df.head().style.format(format_dict)
```

Out[7]:

	State	Fips	County	Population	Total Current Cases	Vaccination Completed	Vaccination Initiated	Vaccination Administered	Completion
0	MI	26001	Alcona County	10405	739	4900	5157	10057	47.1%
1	MI	26003	Alger County	9108	667	4509	4950	9459	49.5%
2	MI	26005	Allegan County	118081	11154	45894	50431	96325	38.9%
3	MI	26007	Alpena County	28405	2268	12675	13355	26030	44.6%
4	MI	26009	Antrim County	23324	1632	10994	11705	22699	47.1%

TRANSFORM

COVID-19 Case Data

COVID ACT NOW

- **Pandas.aggregate function :** Aggregate using one or more operations over the specified axis (Get Min & Max)

```
In [8]: # Using .aggregate()
vaccination_MI = vaccination_df.groupby('State', as_index=True).agg({'Completion': ['min', 'max']})
vaccination_MI
```

Out[8]:

	Completion	
	min	max
State		
MI	0.239333	0.615091

```
In [9]: vaccination_df.sort_values(by=['Completion'], ascending=False).style.format(format_dict)
```

Out[9]:

	State	Fips	County	Population	Total Current Cases	Vaccination Completed	Vaccination Initiated	Vaccination Administered	Completion
44	MI	26089	Leelanau County	21761	1242	13385	14188	27573	61.5%
27	MI	26055	Grand Traverse County	93088	6472	49911	53591	103502	53.6%
80	MI	26161	Washtenaw County	367601	27083	193089	209895	402984	52.5%
23	MI	26047	Emmet County	33415	2382	17538	19021	36559	52.5%
9	MI	26019	Benzie County	17766	1280	9223	9783	19006	51.9%

TRANSFORM

COVID-19 Case Data

COVID ACT NOW

- **Pandas.profileReport:**

  (from pandas_profiling import ProfileReport)

  Pandas profile report generates multiple statistic results based on given dataset.

  Each variable shows high correlation (Pearson's r)

TRANSFORM

COVID-19 Case Data

COVID ACT NOW

- Two table created from Covid Act Now API using a for loop to obtain data and convert to DataFrame.

```
fips_list_2 = []
test_postive_ratio  =[]
case_density =[]

for x in covid_19_data:
    fips_list_2.append(x['fips'])
    test_postive_ratio.append(x['metrics']['testPositivityRatio'])
    case_density.append(x['metrics']['caseDensity'])

test_case_ratios = pd.DataFrame({

    "Fips" : fips_list_2,
    "Test Positive Ratio": test_postive_ratio,
    "Case Density": case_density

})
```

```
fips_list_1 = []
deaths = []
infection_rate = []

for x in covid_19_data:
    fips_list_1.append(x['fips'])
    deaths.append(x['actuals']['deaths'])
    infection_rate.append(x['metrics']['infectionRate'])

death_infection = pd.DataFrame({

    "Fips" : fips_list_1,
    "Deaths" : deaths,
    "Infection Rate": infection_rate

})
```

LOAD

COVID-19 Case Data

- Postgresql: Create connection to Postgresql and create 4 tables in SQL to import (death_infections, test_case_ratios, april_2020_2021, and vaccination
- Used pandas to load csv and json files into Postgresql tables
- Joined vaccination table and april_2020_2021 table on counties to show total current cases vs cases in April 2021.

```
In [26]: rds_connection_string = "postgres:Wzslyz0@9@localhost:5432/ETL_db"
         engine = create_engine(f'postgresql://{rds_connection_string}')

In [27]: engine.table_names()
Out[27]: ['death_infections', 'test_case_ratios', 'april_2020_2021', 'vaccination']

In [28]: vaccination_df.to_sql(name='vaccination', con=engine, if_exists='replace', index=False)

In [29]: death_infection.to_sql(name='death_infections', con=engine, if_exists='replace', index=False)

In [30]: test_case_ratios.to_sql(name='test_case_ratios', con=engine, if_exists='replace', index=False)

In [31]: new_df.to_sql(name='april_2020_2021', con=engine, if_exists='replace', index=False)

In [32]: pd.read_sql_query('select * from vaccination', con=engine).head()
```

Out[32]:

	State	Fips	County	Population	Total_Current_Cases	Vaccination_Completed	Vaccination_Initiated	Vaccination_Administered	Completion
0	MI	26001	Alcona	10405	739	4900	5157	10057	0.470927
1	MI	26003	Alger	9108	667	4509	4950	9459	0.495059
2	MI	26005	Allegan	118081	11154	45894	50431	96325	0.388665
3	MI	26007	Alpena	28405	2268	12675	13355	26030	0.446224
4	MI	26009	Antrim	23324	1632	10994	11705	22699	0.471360

```
SELECT * FROM vaccination;


SELECT va."County", va."Total_Current_Cases", ap."Cases"
FROM vaccination as va
JOIN april_2020_2021 as ap
ON (ap."County" = va."County");
```

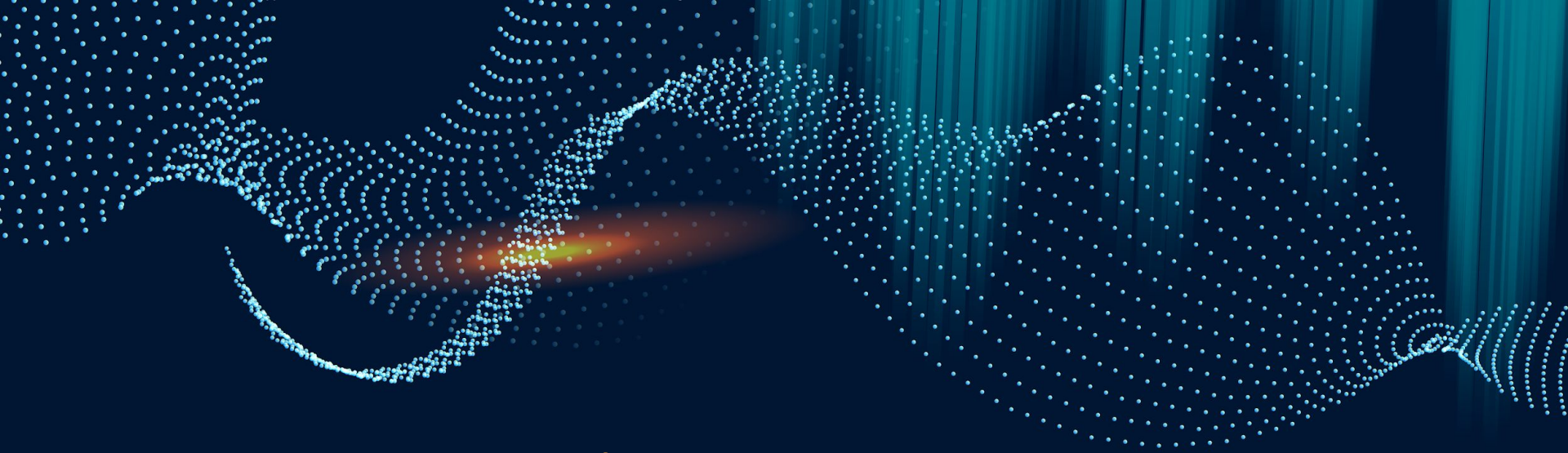
Output | Explain | Messages | Notifications

County text	Total_Current_Cases bigint	Cases bigint
Alcona	739	18216
Alger	667	14300
Allegan	11154	281281
Alpena	2268	55521
Antrim	1632	41832
Arenac	1208	28454

Sources

1. Johns Hopkins (JHU CSSE)
    - This data set is licensed under the Creative Commons Attribution 4.0 International (CC BY 4.0) by the Johns Hopkins University on behalf of its Center for Systems Science in Engineering. Copyright Johns Hopkins University 2020.
    - "COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University" or "JHU CSSE COVID-19 Data" for short,.
    - URL:https://github.com/CSSEGISandData/COVID-19/blob/master/csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_confirmed_US.csv

2. COVID ACT NOW
    - The Covid Act NOW API provides access to comprehensive COVID data (both current and historical.
    - The data is available for all US states, counties and metros and is aggregated from a number of official sources, quality assured and updated daily.
    - URL : https://covidactnow.org/data-api



Questions or Comments?

Thank you!