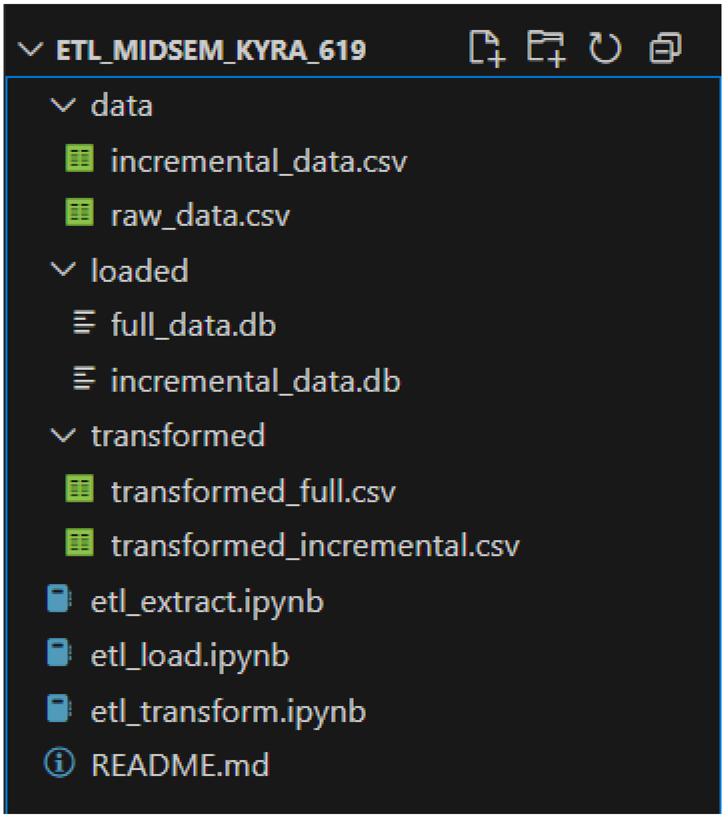
Screenshots

1) File structure



2) ent_extract.ipynb

etl_extract.ipynb > #Display a .head() and .info() of each

```python
import pandas as pd
```

[15] ✓ 0.0s

```python
# Load raw and incremental datasets
raw_df = pd.read_csv("data/raw_data.csv")
inc_df = pd.read_csv("data/incremental_data.csv")
```

[16] ✓ 0.0s

```python
# Preview the first few rows of both datasets
print("Preview of raw_data.csv:")
display(raw_df.head())

print("Preview of incremental_data.csv:")
display(inc_df.head())
```

✓ 0.0s

```
Preview of raw_data.csv:
```

	order_id	customer_name	product	quantity	unit_price	order_date	region
0	1	Diana	Tablet	NaN	500.0	2024-01-20	South
1	2	Eve	Laptop	NaN	NaN	2024-04-29	North
2	3	Charlie	Laptop	2.0	250.0	2024-01-08	NaN
3	4	Eve	Laptop	2.0	750.0	2024-01-07	West
4	5	Eve	Tablet	3.0	NaN	2024-03-07	South

```
Preview of incremental_data.csv:
```

	order_id	customer_name	product	quantity	unit_price	order_date	region
0	101	Alice	Laptop	NaN	900.0	2024-05-09	Central
1	102	NaN	Laptop	1.0	300.0	2024-05-07	Central
2	103	NaN	Laptop	1.0	600.0	2024-05-04	Central
3	104	NaN	Tablet	NaN	300.0	2024-05-26	Central
4	105	Heidi	Tablet	2.0	600.0	2024-05-21	North

```
raw_df.to_csv("data/raw_data.csv", index=False)
inc_df.to_csv("data/incremental_data.csv", index=False)
print("Files saved to /data/")
```

✓ 0.0s

```
Files saved to /data/
```

```
'''
Observations
1) raw_data.csv
    (i) Not Missing Values:
            order_id and product have no missing values.


    (ii) Missing Values:
            customer_name: 1
            quantity: 26
            unit_price: 35
            order_date: 1
            region: 25

        Conclusion:
            35 missing unit_price values may affect any calculations using price.
            26 missing quantity values could interfere with totals or filtering.


    (iii) Duplicate Rows
        1 duplicate row was detected and should be removed during transformation.


(2) incremental_data.csv
    (i) Not Missing Values:
            order_id, product, and unit_price have no missing values.

    (ii) Missing Values:
            customer_name: 6
            quantity: 4
            region: 2

    (iii) Duplicate Rows:
        1 duplicate row was detected and should be removed during transformation.


Conclusion:
    The structure of incremental_data.csv is the same as raw_data.csv, so it should be easy to combine or add to the main dataset during the transformation step.

'''
```

✓ 0.0s

```
#Display a .head() and .info() of each
print("iInfo for raw_data.csv:")
raw_df.info()

print("\ninfo for incremental_data.csv:")
inc_df.info()
```

✓ 0.0s

```
iInfo for raw_data.csv:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   order_id       100 non-null    int64
 1   customer_name  99 non-null     object
 2   product        100 non-null    object
 3   quantity       74 non-null     float64
 4   unit_price     65 non-null     float64
 5   order_date     99 non-null     object
 6   region         75 non-null     object
dtypes: float64(2), int64(1), object(4)
memory usage: 5.6+ KB

info for incremental_data.csv:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   order_id       10 non-null     int64
 1   customer_name  4 non-null      object
 2   product        10 non-null     object
...
 5   order_date     10 non-null     object
 6   region         8 non-null      object
dtypes: float64(2), int64(1), object(4)
memory usage: 692.0+ bytes
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```python
# Missing values
print("Missing values in raw_data.csv:")
print(raw_df.isnull().sum())

print("\nMissing values in incremental_data.csv:")
print(inc_df.isnull().sum())

# Duplicates
print("\nDuplicate rows in raw_data.csv:")
print(raw_df.duplicated().sum())

print("\nDuplicate rows in incremental_data.csv:")
print(inc_df.duplicated().sum())
```

9] ✓ 0.0s

```
Missing values in raw_data.csv:
order_id          0
customer_name     1
product           0
quantity         26
unit_price       35
order_date        1
region           25
dtype: int64

Missing values in incremental_data.csv:
order_id         0
customer_name    6
product          0
quantity         4
unit_price       0
order_date       0
region           2
dtype: int64

Duplicate rows in raw_data.csv:
1

Duplicate rows in incremental_data.csv:
0
```

3) ent_transform.ipynb

Generate + Code + Markdown

```python
import pandas as pd
```

✓ 0.0s

```python
# Load the raw datasets
raw_df = pd.read_csv("data/raw_data.csv")
inc_df = pd.read_csv("data/incremental_data.csv")
```

✓ 0.0s

```python
# (1) Removing duplicate rows (Cleaning)
    # Having duplicate rows can mess up your totals and give misleading results. Removing them just keeps the data clean and makes sure everything adds up correctly.

# Before
print("Raw duplicates:", raw_df.duplicated().sum())
print("Incremental duplicates:", inc_df.duplicated().sum())

raw_df = raw_df.drop_duplicates()
inc_df = inc_df.drop_duplicates()

# After
print("After cleaning → Raw:", raw_df.duplicated().sum(), " | Incremental:", inc_df.duplicated().sum())
```

✓ 0.0s

```
Raw duplicates: 1
Incremental duplicates: 0
After cleaning → Raw: 0  | Incremental: 0
```

```python
# (2) Filling missing customer_name with "N/A" (Cleaning)
    #Customer names are useful for tracking orders, but missing names don't affect calculations. Filling them avoids null values while preserving the rest of the row.
raw_df['customer_name'] = raw_df['customer_name'].fillna("N/A")
inc_df['customer_name'] = inc_df['customer_name'].fillna("N/A")
```

✓ 0.0s

```python
# (3) Drop rows where quantity or unit_price are missing (Cleaning)
    #These columns are needed to calculate the total price, so if they're missing, the row isn't really useful, it's better to just drop it.
raw_df.dropna(subset=['quantity', 'unit_price'], inplace=True)
inc_df.dropna(subset=['quantity', 'unit_price'], inplace=True)
```

✓ 0.0s

```python
# (4) Create total_price Column (Enrichment)
    #This helps show how much each order is actually worth, which is useful for things like tracking sales, comparing customers, or spotting big spenders.
raw_df['total_price'] = raw_df['quantity'] * raw_df['unit_price']
inc_df['total_price'] = inc_df['quantity'] * inc_df['unit_price']

raw_df[['quantity', 'unit_price', 'total_price']].head()
```

✓ 0.0s

	quantity	unit_price	total_price
2	2.0	250.0	500.0
3	2.0	750.0	1500.0
6	2.0	750.0	1500.0
9	1.0	500.0	500.0
10	3.0	750.0	2250.0

ETL_Midsem_Kyra_619

```python
# (5) Convert to order_date to datetime (Structural)
    #Changing the order dates to real date format makes it easier to sort, filter, or do things like track trends over time, like seeing what months had the most sales.
raw_df['order_date'] = pd.to_datetime(raw_df['order_date'], errors='coerce')
inc_df['order_date'] = pd.to_datetime(inc_df['order_date'], errors='coerce')
```

✓ 0.0s

```python
# (6) Group total_price into tiers (Categorization)
    #Grouping the total prices into levels like low, medium, or high makes it easier to spot different types of customers and compare spending patterns across orders.
price_bins = [0, 100, 500, 1000, float('inf')]
labels = ['Low', 'Medium', 'High', 'Very High']

raw_df['price_tier'] = pd.cut(raw_df['total_price'], bins=price_bins, labels=labels)
inc_df['price_tier'] = pd.cut(inc_df['total_price'], bins=price_bins, labels=labels)

raw_df[['total_price', 'price_tier']].head()
```

✓ 0.0s

	total_price	price_tier
2	500.0	Medium
3	1500.0	Very High
6	1500.0	Very High
9	500.0	Medium
10	2250.0	Very High

```python
raw_df.to_csv("transformed/transformed_full.csv", index=False)
inc_df.to_csv("transformed/transformed_incremental.csv", index=False)

print("Transformed files saved to /transformed")
```

✓ 0.0s

(i) transform_full.cvs

transformed > transformed_full.csv > data

```
order_id,customer_name,product,quantity,unit_price,order_date,region,total_price,price_tier
3,Charlie,Laptop,2.0,250.0,2024-01-08,,500.0,Medium
4,Eve,Laptop,2.0,750.0,2024-01-07,West,1500.0,Very High
7,Charlie,Monitor,2.0,750.0,2024-02-02,West,1500.0,Very High
10,Eve,Monitor,1.0,500.0,2024-02-28,North,500.0,Medium
11,N/A,Monitor,3.0,750.0,2024-04-24,West,2250.0,Very High
12,Charlie,Tablet,2.0,750.0,2024-03-26,East,1500.0,Very High
13,Frank,Tablet,1.0,750.0,2024-04-28,West,750.0,High
16,Diana,Monitor,3.0,750.0,,East,2250.0,Very High
18,Frank,Tablet,1.0,250.0,2024-04-11,South,250.0,Medium
22,Alice,Phone,3.0,750.0,2024-01-08,North,2250.0,Very High
23,Diana,Tablet,3.0,500.0,2024-01-27,West,1500.0,Very High
26,Eve,Monitor,3.0,750.0,2024-01-21,West,2250.0,Very High
28,Alice,Laptop,2.0,250.0,2024-04-06,North,500.0,Medium
29,Alice,Monitor,1.0,250.0,2024-01-28,,250.0,Medium
30,Charlie,Monitor,1.0,500.0,2024-04-20,West,500.0,Medium
31,Charlie,Phone,3.0,750.0,2024-03-04,West,2250.0,Very High
32,Bob,Laptop,2.0,750.0,2024-04-06,West,1500.0,Very High
35,Frank,Tablet,3.0,750.0,2024-02-17,,2250.0,Very High
36,Frank,Phone,3.0,750.0,2024-01-19,West,2250.0,Very High
41,Alice,Monitor,1.0,250.0,2024-01-17,West,250.0,Medium
44,Charlie,Tablet,2.0,250.0,2024-01-30,North,500.0,Medium
45,Eve,Laptop,1.0,500.0,2024-04-02,East,500.0,Medium
46,Alice,Tablet,2.0,250.0,2024-02-15,West,500.0,Medium
50,Diana,Tablet,3.0,250.0,2024-04-14,South,750.0,High
51,Frank,Laptop,2.0,250.0,2024-02-06,,500.0,Medium
53,Bob,Phone,1.0,750.0,2024-04-02,West,750.0,Hi
54,Alice,Tablet,1.0,750.0,2024-04-22,,750.0,Hig  Col 9: price_tier
55,Bob,Tablet,1.0,250.0,2024-02-15,North,250.0,Medium
57,Bob,Tablet,1.0,750.0,2024-04-04,South,750.0,High
58,Diana,Tablet,3.0,750.0,2024-04-08,North,2250.0,Very High
61,Diana,Tablet,2.0,750.0,2024-04-25,,1500.0,Very High
64,Frank,Monitor,3.0,500.0,2024-02-01,,1500.0,Very High
69,Bob,Tablet,3.0,750.0,2024-01-25,South,2250.0,Very High
70,Frank,Laptop,2.0,250.0,2024-04-15,East,500.0,Medium
76,Eve,Laptop,2.0,500.0,2024-02-15,North,1000.0,High
80,Bob,Tablet,3.0,250.0,2024-02-16,South,750.0,High
81,Bob,Tablet,3.0,750.0,2024-03-26,South,2250.0,Very High
83,Diana,Monitor,3.0,250.0,2024-03-06,West,750.0,High
84,Frank,Laptop,1.0,500.0,2024-01-27,North,500.0,Medium
86,Diana,Tablet,2.0,500.0,2024-01-02,South,1000.0,High
88,Frank,Tablet,1.0,500.0,2024-01-17,,500.0,Medium
90,Eve,Monitor,2.0,750.0,2024-02-02,South,1500.0,Very High
92,Eve,Phone,2.0,250.0,2024-02-12,South,500.0,Medium
93,Bob,Laptop,3.0,250.0,2024-04-27,South,750.0,High
97,Diana,Phone,1.0,250.0,2024-02-11,,250.0,Medium
98,Eve,Monitor,1.0,500.0,2024-04-28,,500.0,Medium
```

(ii) transform_incremental.cvs

transformed > transformed_incremental.csv > data

```
order_id,customer_name,product,quantity,unit_price,order_date,region,total_price,price_tier
102,N/A,Laptop,1.0,300.0,2024-05-07,Central,300.0,Medium
103,N/A,Laptop,1.0,600.0,2024-05-04,Central,600.0,High
105,Heidi,Tablet,2.0,600.0,2024-05-21,North,1200.0,Very High
106,N/A,Laptop,2.0,600.0,2024-05-18,Central,1200.0,Very High
107,N/A,Tablet,1.0,600.0,2024-05-13,Central,600.0,High
109,Grace,Laptop,2.0,600.0,2024-05-29,Central,1200.0,Very High

```

4) ent_load.ipynb

```python
import pandas as pd
import sqlite3

# Load transformed CSVs
full_df = pd.read_csv("transformed/transformed_full.csv")
inc_df = pd.read_csv("transformed/transformed_incremental.csv")
```

✓ 0.0s

```python
# Connect to SQLite and save full data
conn_full = sqlite3.connect("loaded/full_data.db")
full_df.to_sql("full_data", conn_full, if_exists="replace", index=False)
conn_full.close()

print("full_data.db saved in /loaded")
```

✓ 0.0s

```
full_data.db saved in /loaded
```

```python
# Connect and save incremental data
conn_inc = sqlite3.connect("loaded/incremental_data.db")
inc_df.to_sql("incremental_data", conn_inc, if_exists="replace", index=False)
conn_inc.close()

print("incremental_data.db saved in /loaded")
```

✓ 0.0s

```
incremental_data.db saved in /loaded
```

```python
# Reconnect and run sample query
conn = sqlite3.connect("loaded/full_data.db")
preview = pd.read_sql("SELECT * FROM full_data LIMIT 5;", conn)
conn.close()

# Display preview
preview
```

✓ 0.0s

	order_id	customer_name	product	quantity	unit_price	order_date	region	total_price	price_tier
0	3	Charlie	Laptop	2.0	250.0	2024-01-08	None	500.0	Medium
1	4	Eve	Laptop	2.0	750.0	2024-01-07	West	1500.0	Very High
2	7	Charlie	Monitor	2.0	750.0	2024-02-02	West	1500.0	Very High
3	10	Eve	Monitor	1.0	500.0	2024-02-28	North	500.0	Medium
4	11	None	Monitor	3.0	750.0	2024-04-24	West	2250.0	Very High

(i) full_data.db

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: full_data

Filter in any column

	order_id	customer_name	product	quantity	unit_price	order_date	region	total_price	price_tier
1	3	Charlie	Laptop	2.0	250.0	2024-01-08	NULL	500.0	Medium
2	4	Eve	Laptop	2.0	750.0	2024-01-07	West	1500.0	Very High
3	7	Charlie	Monitor	2.0	750.0	2024-02-02	West	1500.0	Very High
4	10	Eve	Monitor	1.0	500.0	2024-02-28	North	500.0	Medium
5	11	NULL	Monitor	3.0	750.0	2024-04-24	West	2250.0	Very High
6	12	Charlie	Tablet	2.0	750.0	2024-03-26	East	1500.0	Very High
7	13	Frank	Tablet	1.0	750.0	2024-04-28	West	750.0	High
8	16	Diana	Monitor	3.0	750.0	NULL	East	2250.0	Very High
9	18	Frank	Tablet	1.0	250.0	2024-04-11	South	250.0	Medium
10	22	Alice	Phone	3.0	750.0	2024-01-08	North	2250.0	Very High
11	23	Diana	Tablet	3.0	500.0	2024-01-27	West	1500.0	Very High
12	26	Eve	Monitor	3.0	750.0	2024-01-21	West	2250.0	Very High
13	28	Alice	Laptop	2.0	250.0	2024-04-06	North	500.0	Medium
14	29	Alice	Monitor	1.0	250.0	2024-01-28	NULL	250.0	Medium
15	30	Charlie	Monitor	1.0	500.0	2024-04-20	West	500.0	Medium
16	31	Charlie	Phone	3.0	750.0	2024-03-04	West	2250.0	Very High
17	32	Bob	Laptop	2.0	750.0	2024-04-06	West	1500.0	Very High
18	35	Frank	Tablet	3.0	750.0	2024-02-17	NULL	2250.0	Very High
19	36	Frank	Phone	3.0	750.0	2024-01-19	West	2250.0	Very High
20	41	Alice	Monitor	1.0	250.0	2024-01-17	West	250.0	Medium
21	44	Charlie	Tablet	2.0	250.0	2024-01-30	North	500.0	Medium
22	45	Eve	Laptop	1.0	500.0	2024-04-02	East	500.0	Medium
23	46	Alice	Tablet	2.0	250.0	2024-02-15	West	500.0	Medium

24	50	Diana	Tablet	3.0	250.0	2024-04-14	South	750.0	High
25	51	Frank	Laptop	2.0	250.0	2024-02-06	NULL	500.0	Medium
26	53	Bob	Phone	1.0	750.0	2024-04-02	West	750.0	High
27	54	Alice	Tablet	1.0	750.0	2024-04-22	NULL	750.0	High
28	55	Bob	Tablet	1.0	250.0	2024-02-15	North	250.0	Medium
29	57	Bob	Tablet	1.0	750.0	2024-04-04	South	750.0	High
30	58	Diana	Tablet	3.0	750.0	2024-04-08	North	2250.0	Very High
31	61	Diana	Tablet	2.0	750.0	2024-04-25	NULL	1500.0	Very High
32	64	Frank	Monitor	3.0	500.0	2024-02-01	NULL	1500.0	Very High
33	69	Bob	Tablet	3.0	750.0	2024-01-25	South	2250.0	Very High
34	70	Frank	Laptop	2.0	250.0	2024-04-15	East	500.0	Medium
35	76	Eve	Laptop	2.0	500.0	2024-02-15	North	1000.0	High
36	80	Bob	Tablet	3.0	250.0	2024-02-16	South	750.0	High
37	81	Bob	Tablet	3.0	750.0	2024-03-26	South	2250.0	Very High
38	83	Diana	Monitor	3.0	250.0	2024-03-06	West	750.0	High
39	84	Frank	Laptop	1.0	500.0	2024-01-27	North	500.0	Medium
40	86	Diana	Tablet	2.0	500.0	2024-01-02	South	1000.0	High
41	88	Frank	Tablet	1.0	500.0	2024-01-17	NULL	500.0	Medium
42	90	Eve	Monitor	2.0	750.0	2024-02-02	South	1500.0	Very High
43	92	Eve	Phone	2.0	250.0	2024-02-12	South	500.0	Medium
44	93	Bob	Laptop	3.0	250.0	2024-04-27	South	750.0	High
45	97	Diana	Phone	1.0	250.0	2024-02-11	NULL	250.0	Medium
46	98	Eve	Monitor	1.0	500.0	2024-04-28	NULL	500.0	Medium

(ii) incremental_data.db

	order_id	customer_name	product	quantity	unit_price	order_date	region	total_price	price_tier
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	102	NULL	Laptop	1.0	300.0	2024-05-07	Central	300.0	Medium
2	103	NULL	Laptop	1.0	600.0	2024-05-04	Central	600.0	High
3	105	Heidi	Tablet	2.0	600.0	2024-05-21	North	1200.0	Very High
4	106	NULL	Laptop	2.0	600.0	2024-05-18	Central	1200.0	Very High
5	107	NULL	Tablet	1.0	600.0	2024-05-13	Central	600.0	High
6	109	Grace	Laptop	2.0	600.0	2024-05-29	Central	1200.0	Very High