Data Pipeline Lab: ETL for Sales Transactions

Student Name: Faith Chakwanira
Student ID: 670435
Submission Date: June 10, 2025

🎯 Project Goal

This project showcases fundamental Extract-Transform-Load (ETL) techniques, specifically focusing on **Full Data Extraction** and **Incremental Updates**. Using Python and Jupyter Notebooks, it simulates a real-world data pipeline by processing synthetic sales transaction data. The primary goal is to demonstrate efficient methods for data ingestion, whether it's an initial complete dataset load or capturing only new/modified records since the last processing run.

🔑 Key Concepts & Demonstrations

1. Full Extraction

This module illustrates how to perform a complete load of a dataset from its source. Essential for initial setup or periodic full refreshes, this process ensures all available data is captured. The accompanying notebook clearly displays statistical summaries and a data sample to confirm successful execution.

2. Incremental Extraction

A more efficient approach for dynamic datasets, incremental extraction focuses on identifying and retrieving only data that has been newly added or updated since the last ETL run. This significantly optimizes performance and resource usage for frequently changing data sources. The simulation within this project demonstrates how new sales records are seamlessly integrated via this method.

3. Data Simulation & State Management

To provide a realistic scenario, the project includes:

- **Synthetic Data Generation**: A `custom_data.csv` file is created with over 50 realistic sales transactions, incorporating `transaction_date` and `last_updated` timestamps crucial for incremental logic.
- **Persistent Timestamp Tracking**: A `last_extraction.txt` file is used to store the timestamp of the most recent successful incremental extraction. This mechanism simulates a real-world persistent state, allowing the ETL process to know where to resume.

⚙️ Technologies Utilized

- **Python 3**: The core language for scripting the ETL logic.
- **Pandas**: Employed for robust data manipulation, analysis, and efficient handling of DataFrames.
- **Jupyter Notebook**: Provides an interactive environment for developing, documenting, and executing the ETL workflow.
- **Git/GitHub**: For version control, project tracking, and collaborative development.

📂 Project Repository Overview

```
ETL_Extract_Faithchakwanira_670435/
├── etl_extract.ipynb         # Main Jupyter notebook containing all ETL operations
├── custom_data.csv           # Generated synthetic sales dataset
├── last_extraction.txt       # Stores the timestamp for incremental data pulls
├── .gitignore                # Specifies files to exclude from version control
├── README.md                 # This project documentation
├── image.png                 # Screenshot: Data Generation Output
├── image-1.png               # Screenshot: Full Extraction Output
├── image-2.png               # Screenshot: Incremental Extraction Output
└── image-3.png               # Screenshot: Simulate New Data for Incremental Extraction
```

🚀 Quick Start Guide

Follow these steps to set up and run the ETL pipeline on your local system:

Prerequisites

Ensure you have Python 3 installed. Then, install the necessary libraries using pip:

```
pip install pandas jupyter
```

Data Handling

The `custom_data.csv` file is **automatically generated** when you execute the first cell of the `etl_extract.ipynb` notebook. No manual download is required.

Getting the Code

Clone this repository to your local machine:

```
git clone [YOUR_REPOSITORY_URL_HERE] # Replace with your actual repository URL
cd ETL_Extract_Faithchakwanira_670435
```

Running the Notebook

1. Launch Jupyter Notebook from your project directory:

   ```
   jupyter notebook
   ```

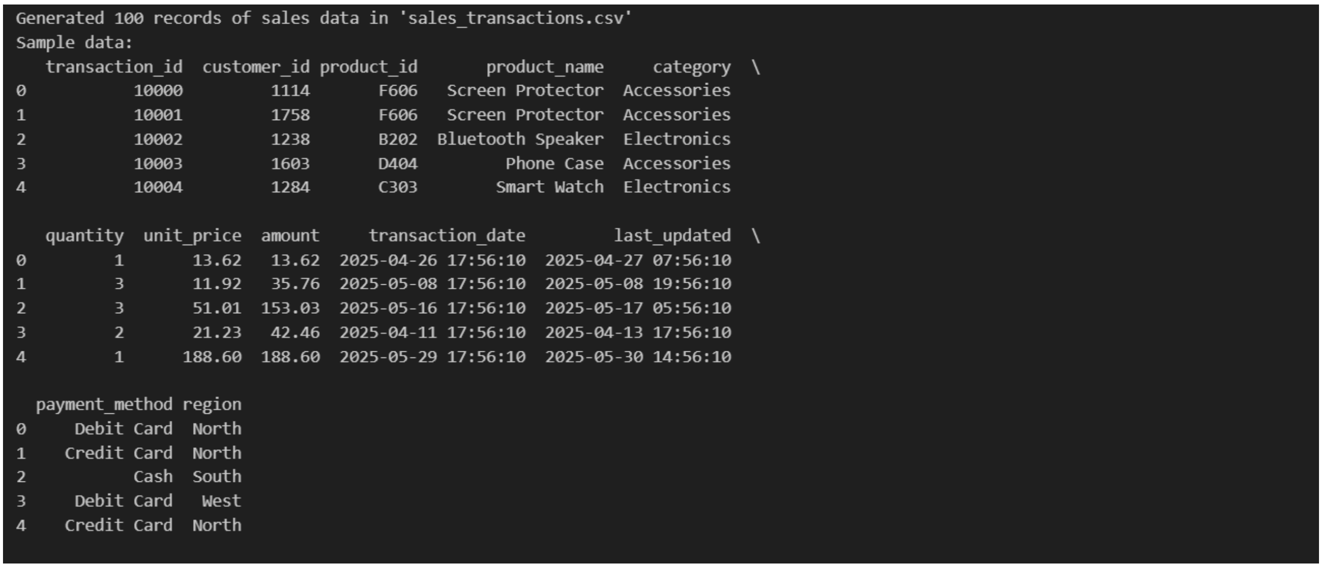
2. In the Jupyter interface, open `etl_extract.ipynb`.
3. Execute all cells in sequence (via `Cell` -> `Run All`) to observe the full ETL process.

📊 Visualizing the Process

The `etl_extract.ipynb` notebook includes clear outputs at each stage. Below are illustrative screenshots:

Data Generation and Initial Snapshot

This image confirms the successful creation of 100 synthetic sales records in `custom_data.csv`, displaying the initial dataset structure including transaction and timestamp fields.



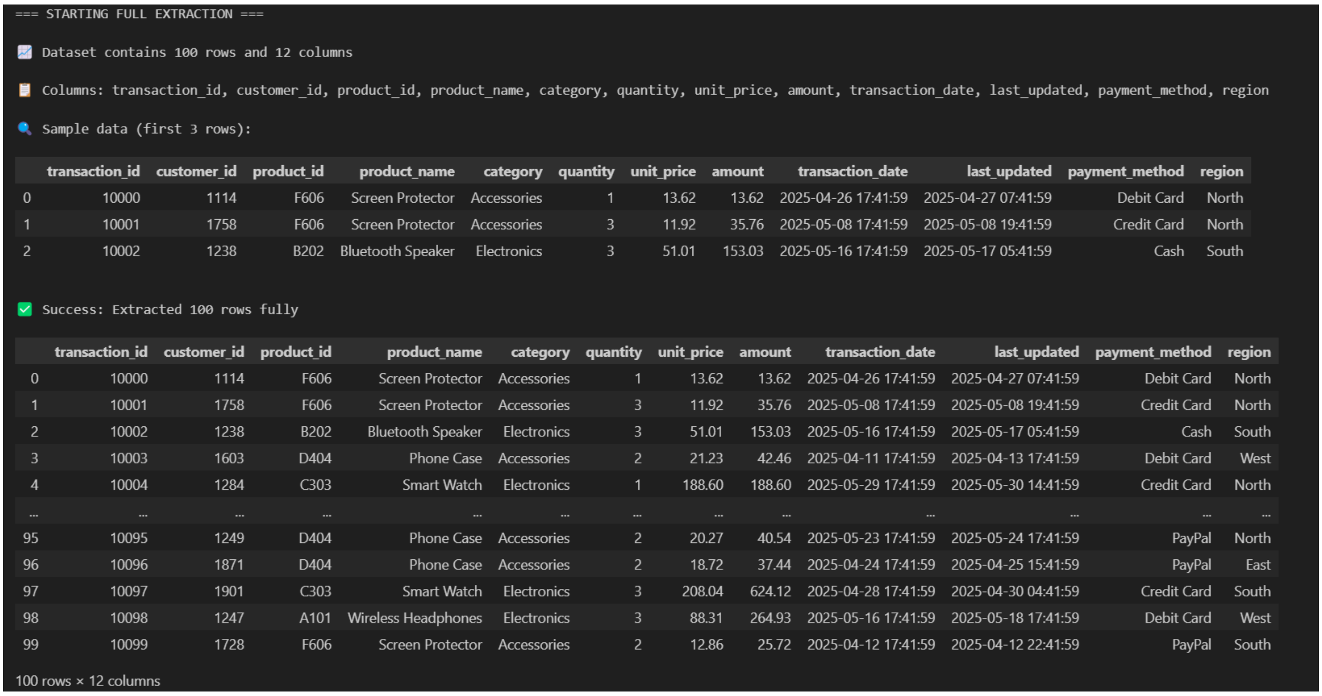
```
Generated 100 records of sales data in 'sales_transactions.csv'
Sample data:
   transaction_id  customer_id product_id       product_name     category  \
0           10000         1114       F606    Screen Protector  Accessories
1           10001         1758       F606    Screen Protector  Accessories
2           10002         1238       B202   Bluetooth Speaker  Electronics
3           10003         1603       D404          Phone Case  Accessories
4           10004         1284       C303         Smart Watch  Electronics

   quantity  unit_price  amount     transaction_date         last_updated  \
0         1       13.62   13.62  2025-04-26 17:56:10  2025-04-27 07:56:10
1         3       11.92   35.76  2025-05-08 17:56:10  2025-05-08 19:56:10
2         3       51.01  153.03  2025-05-16 17:56:10  2025-05-17 05:56:10
3         2       21.23   42.46  2025-04-11 17:56:10  2025-04-13 17:56:10
4         1      188.60  188.60  2025-05-29 17:56:10  2025-05-30 14:56:10

  payment_method region
0     Debit Card  North
1    Credit Card  North
2           Cash  South
3     Debit Card   West
4    Credit Card  North
```

Comprehensive Data Extraction

This screenshot shows the result of the "Full Extraction," confirming that the entire `custom_data.csv` dataset has been loaded, along with row/column counts and a data sample.



```
=== STARTING FULL EXTRACTION ===

📈 Dataset contains 100 rows and 12 columns

📋 Columns: transaction_id, customer_id, product_id, product_name, category, quantity, unit_price, amount, transaction_date, last_updated, payment_method, region

🔍 Sample data (first 3 rows):
```

	transaction_id	customer_id	product_id	product_name	category	quantity	unit_price	amount	transaction_date	last_updated	payment_method	region
0	10000	1114	F606	Screen Protector	Accessories	1	13.62	13.62	2025-04-26 17:41:59	2025-04-27 07:41:59	Debit Card	North
1	10001	1758	F606	Screen Protector	Accessories	3	11.92	35.76	2025-05-08 17:41:59	2025-05-08 19:41:59	Credit Card	North
2	10002	1238	B202	Bluetooth Speaker	Electronics	3	51.01	153.03	2025-05-16 17:41:59	2025-05-17 05:41:59	Cash	South

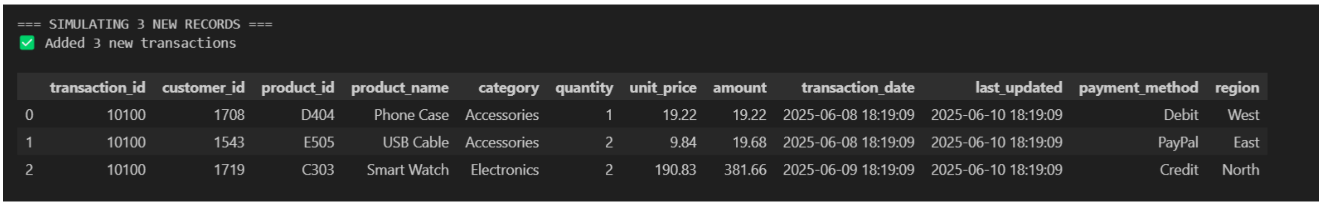
```
✅ Success: Extracted 100 rows fully
```

	transaction_id	customer_id	product_id	product_name	category	quantity	unit_price	amount	transaction_date	last_updated	payment_method	region
0	10000	1114	F606	Screen Protector	Accessories	1	13.62	13.62	2025-04-26 17:41:59	2025-04-27 07:41:59	Debit Card	North
1	10001	1758	F606	Screen Protector	Accessories	3	11.92	35.76	2025-05-08 17:41:59	2025-05-08 19:41:59	Credit Card	North
2	10002	1238	B202	Bluetooth Speaker	Electronics	3	51.01	153.03	2025-05-16 17:41:59	2025-05-17 05:41:59	Cash	South
3	10003	1603	D404	Phone Case	Accessories	2	21.23	42.46	2025-04-11 17:41:59	2025-04-13 17:41:59	Debit Card	West
4	10004	1284	C303	Smart Watch	Electronics	1	188.60	188.60	2025-05-29 17:41:59	2025-05-30 14:41:59	Credit Card	North
...	...	...	...	...	...	...	...	...	...	...	...	...
95	10095	1249	D404	Phone Case	Accessories	2	20.27	40.54	2025-05-23 17:41:59	2025-05-24 17:41:59	PayPal	North
96	10096	1871	D404	Phone Case	Accessories	2	18.72	37.44	2025-04-24 17:41:59	2025-04-25 15:41:59	PayPal	East
97	10097	1901	C303	Smart Watch	Electronics	3	208.04	624.12	2025-04-28 17:41:59	2025-04-30 04:41:59	Credit Card	South
98	10098	1247	A101	Wireless Headphones	Electronics	3	88.31	264.93	2025-05-16 17:41:59	2025-05-18 17:41:59	Debit Card	West
99	10099	1728	F606	Screen Protector	Accessories	2	12.86	25.72	2025-04-12 17:41:59	2025-04-12 22:41:59	PayPal	South

100 rows × 12 columns

Simulating New Data for Updates

Before the incremental run, this image demonstrates the addition of new records to `custom_data.csv`, mimicking live system updates.

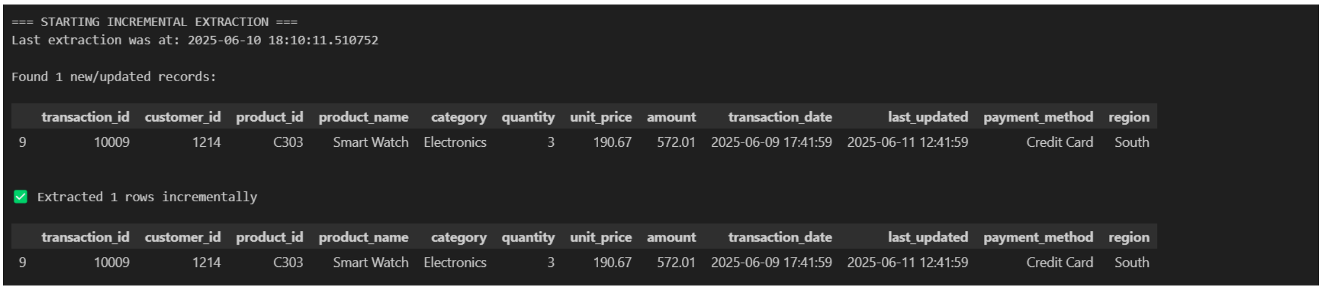


```
=== SIMULATING 3 NEW RECORDS ===
✅ Added 3 new transactions
```

	transaction_id	customer_id	product_id	product_name	category	quantity	unit_price	amount	transaction_date	last_updated	payment_method	region
0	10100	1708	D404	Phone Case	Accessories	1	19.22	19.22	2025-06-08 18:19:09	2025-06-10 18:19:09	Debit	West
1	10100	1543	E505	USB Cable	Accessories	2	9.84	19.68	2025-06-08 18:19:09	2025-06-10 18:19:09	PayPal	East
2	10100	1719	C303	Smart Watch	Electronics	2	190.83	381.66	2025-06-09 18:19:09	2025-06-10 18:19:09	Credit	North

Incremental Data Capture

This final image highlights the outcome of the "Incremental Extraction," showcasing the `last_extraction_timestamp` and specifically the new or updated records (e.g., 4 rows) captured since the last check.



```
=== STARTING INCREMENTAL EXTRACTION ===
Last extraction was at: 2025-06-10 18:10:11.510752

Found 1 new/updated records:
```

	transaction_id	customer_id	product_id	product_name	category	quantity	unit_price	amount	transaction_date	last_updated	payment_method	region
9	10009	1214	C303	Smart Watch	Electronics	3	190.67	572.01	2025-06-09 17:41:59	2025-06-11 12:41:59	Credit Card	South

```
✅ Extracted 1 rows incrementally
```

	transaction_id	customer_id	product_id	product_name	category	quantity	unit_price	amount	transaction_date	last_updated	payment_method	region
9	10009	1214	C303	Smart Watch	Electronics	3	190.67	572.01	2025-06-09 17:41:59	2025-06-11 12:41:59	Credit Card	South

✨ Final Thoughts

This project effectively demonstrates both comprehensive and incremental data extraction methodologies, vital for building efficient and scalable data pipelines. The detailed explanations and practical demonstrations within the notebook and this README aim to provide a thorough understanding of these concepts.