Re-Vision Data Engineering Task

An ETL pipeline in the data engineering Re-Vision team.

1.ETL Architecture

2.Category Tree

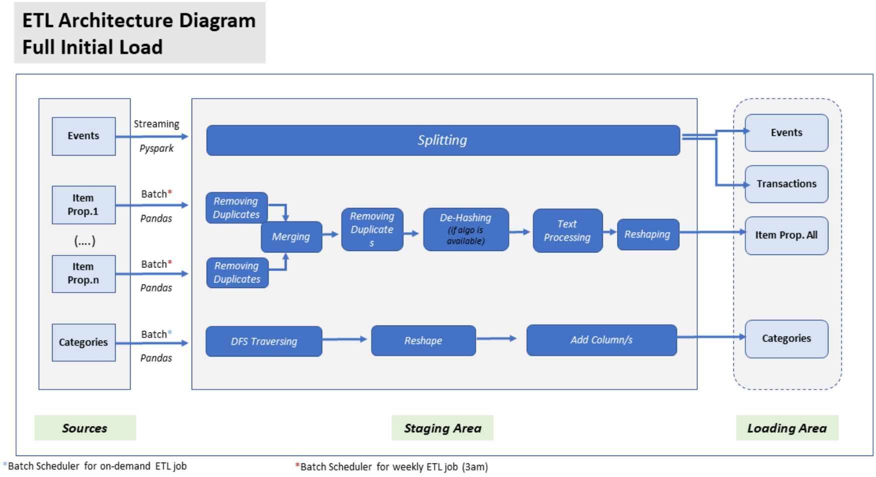
3.Item_properties

4.Events

5.Data Model

6.How to run the code

ETL Architecture Diagram



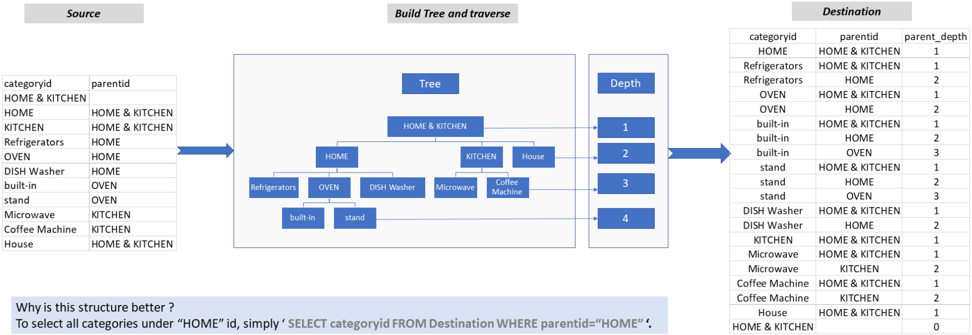
The image above lays out the processing structure. We are going to go through an overview of why we try to adhere to this structure and will try to use the problem set to explain how we'll apply this structure to the problem set.

The first thing to note is our tech stack. We use CSVs for data storage, our transformation logic to move data from A to B is not restricted to but normally carried out in Python or Spark. We are using batch processing in the **"Full Initial Load"** however we can use Spark streaming at **Incremental Load"**.Also the current pipeline export data to CSV we can export it to any On-demand database like MYSQL or Cloud-based like AWS redshift.

Second, The attached scripts is exist in two pictures, **.ipynb** for explanation and **.py** for running.

Third, although all scripts contain the names of files as a variable, we can pass them as Command Line Arguments.

Category Tree



categoryid	parentid
HOME & KITCHEN	
HOME	HOME & KITCHEN
KITCHEN	HOME & KITCHEN
Refrigerators	HOME
OVEN	HOME
DISH Washer	HOME
built-in	OVEN
stand	OVEN
Microwave	KITCHEN
Coffee Machine	KITCHEN
House	HOME & KITCHEN

categoryid	parentid	parent_depth
HOME	HOME & KITCHEN	1
Refrigerators	HOME & KITCHEN	1
Refrigerators	HOME	2
OVEN	HOME & KITCHEN	1
OVEN	HOME	2
built-in	HOME & KITCHEN	1
built-in	HOME	2
built-in	OVEN	3
stand	HOME & KITCHEN	1
stand	HOME	2
stand	OVEN	3
DISH Washer	HOME & KITCHEN	1
DISH Washer	HOME	2
KITCHEN	HOME & KITCHEN	1
Microwave	HOME & KITCHEN	1
Microwave	KITCHEN	2
Coffee Machine	HOME & KITCHEN	1
Coffee Machine	KITCHEN	2
House	HOME & KITCHEN	1
HOME & KITCHEN		0

Why is this structure better ?
To select all categories under “HOME” id, simply ‘ **SELECT categoryid FROM Destination WHERE parentid=“HOME”** ‘.

The Structure of `Catergory_Tree` is memory efficient, however, it makes it hard to retrieve any of the pieces of information under some ParentId. Thus we use **DFS Traverse** to traverse the tree and make vertex between each node, which helps in finding the whole path from the root to leaves

Item_properties

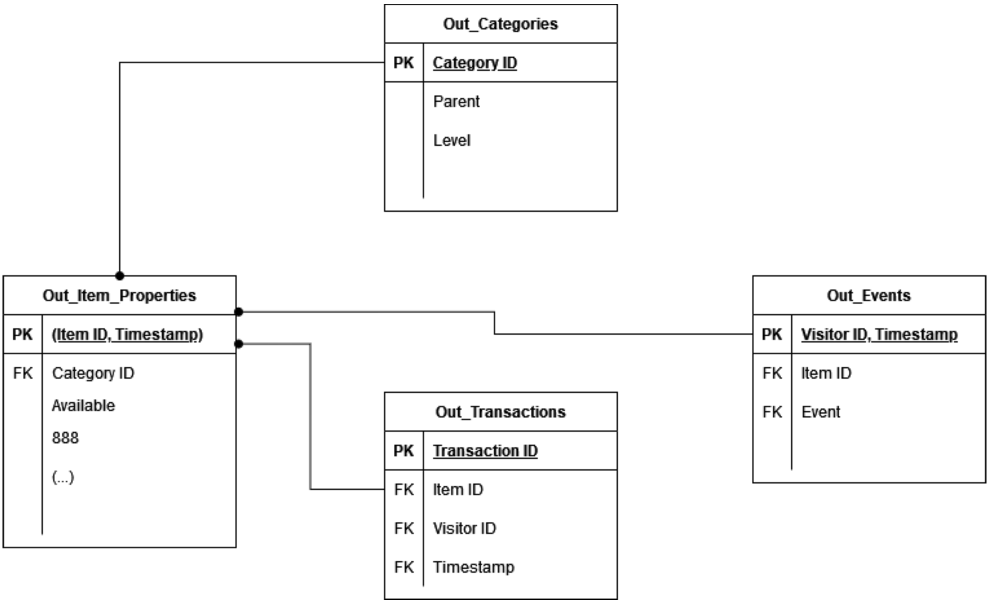
Read parts iteratively and remove duplicates and keep only a single snapshot, and keep those processed parts at the staging area as **CSVs** then after processing all the parts, back to those staging files and apply the same processes and remove staging files and keep only `Out_Item_properties.csv`

Note: 1-we can reshape this file to keep all properties as columns (wide view). 1-if we convert it to wide view the primary key for this table will be `itemId` and `timestamp` together.

Events

Just split it into two tables `Out_Transactions` and `Out_events` to reduce the `NULL` occurrence in the "transactionId" column and drop it from `Out_events` .

Data Model



How to run the code

Colne the repo and add the scripts to the same files folder.

First, To run batch processing

Install `python-3.10.5`

The `requirements.txt` file should list all Python libraries scripts depend on, and they will be installed using:

```
pip install -r requirements.txt
```

To run The item praparties script

```
python ETL_item_properties.py
```

To run The events script

```
python ETL_events.py
```

To run The category script

```
python ETL_category.py
```

second, To run streaming process

Follow this link