Batch Processing ETL Pipeline for Candy Store

Project Introduction

In this project, I've implemented batch processing logic to process raw order transactions at the end of each day. This process includes validating transaction details and verifying inventory levels to ensure successful order shipment. The daily sales and profit numbers are to be put into a time series forecasting model to predict future sales and profits.

▼ Dataset Description

The dataset contains information about a candy store's customer data and product information. The data spans from February 1st to 10th, 2024.

1. **Customers** Dataset
   Contains information about the store's customers.
   - customer_id (Integer): Unique identifier for each customer.
   - first_name (String): First name of the customer
   - last_name (String): Last name of the customer
   - email (String): Email address of the customer
   - address (String): Physical address of the customer
   - phone (String): Phone number of the customer

customer_id	first_name	last_name	email	address	phone
1	Mitchell	Clements	reginabrown(	3154 Marvin Place Apt. 069, Newtonmouth, ND 60366	3163213149
2	Alexis	Clark	nporter@exa	USCGC Wallace, FPO AA 24551	4974637956
3	Chelsea	Gamble	taylordavid@	88666 Heather Cape, South Calvinbury, AZ 93725	+1-394-379-1810x55116
4	Michael	Mccarty	vsnyder@exa	6803 Wilson Fort, Leeport, NC 40170	(984)878-0389
5	Jennifer	Warren	corycarter@	12343 Green Way Suite 898, North Joanna, NH 34378	(377)853-2017
6	Tina	Casey	sullivanrache	PSC 5222, Box 1030, APO AP 46040	+1-527-662-0548x26528
7	John	Willis	hschroeder@	7083 Heather Isle, Yolandaburgh, SC 36166	676-446-5549x5169
8	Mary	Adams	brownbrittan	02164 Angela Squares, Castillofort, PA 95504	412.491.8585x05553
9	Amy	Collins	denisemoore	91677 Cindy Loop, Johnhaven, MA 43969	689-868-5497x26583
10	Michelle	Riley	tonyroth@ex	1396 Carlson Lock, Andreashire, TX 03335	699.778.1655x352

2. **Products** Dataset
   Contains information about the candy.
   - product_id (Integer): Unique identifier for each product.
   - product_name (String): Name of the product
   - product_category (String): Type of candy (e.g., Chocolate, Gummy, Hard Candy)
   - product_subcategory (String): Subcategory of the candy
   - product_<property_1>: <description of this property>
   - sales_price (Decimal): Retail price of the product. Range: $0.50 to $10.00
   - cost_to_make (Decimal): Production cost of the product. Range: 40% to 70% of the sales price
   - stock (Integer): Current inventory count. Range: 2000 to 5000

product_id	product_name	product_category	product_subcategory	product_shape	sales_price	cost_to_make	stock
1	Salted Bites Ropes	Licorice	Bites	Ropes	9.26	5.01	4236
2	Black/Red/White Bites Ropes	Licorice	Bites	Ropes	0.59	0.4	4961
3	Salted Bites Cubes	Licorice	Bites	Cubes	2.16	1.14	4589
4	Black/Red/White Bites Cubes	Licorice	Bites	Cubes	4.36	2.84	2715
5	Salted Australian Ropes	Licorice	Australian	Ropes	7.91	5.44	3476
6	Black/Red/White Australian Ropes	Licorice	Australian	Ropes	3.97	1.83	2414
7	Salted Australian Cubes	Licorice	Australian	Cubes	2.35	1.27	2597
8	Black/Red/White Australian Cubes	Licorice	Australian	Cubes	6.16	4.08	3052
9	Salted Laces Ropes	Licorice	Laces	Ropes	7.28	4.75	4561

3. **Raw Order Transactions (from 2024/02/01 to 2024/02/10, 10 files)**:


```
{
 "transaction_id": 73434473,
 "customer_id": 29,
 "timestamp": "2024-02-02T12:00:40.808092",
 "items": [
  {
   "product_id": 17,
```

```
      "product_name": "Sea Salt Crackle Enrobed Bites",
      "qty": 5
    },
    {
      "product_id": 18,
      "product_name": "Almond Shards Enrobed Bites",
      "qty": null
    },
    {
      "product_id": 3,
      "product_name": "Powdered Sugar Sticks Rectangles",
      "qty": 2
    }
  ]
},
```


▼ Requirements

Note: 1. Pandas wasn't used for processing data

2. Print processing progress and intermediate results to terminal

1. Data Loading

- Successfully load initial data from CSV to MySQL
  - customers.csv
  - products.csv
- Successfully load initial data from JSON to MongoDB
  - transactions_20240201.json - transactions_20240210.json
- Load data from MySQL dataset to Spark session and display data preview with dimensions
- Load data from MongoDB to Spark session and display data preview with dimensions

---

2. Batch Processing ETL

Note:
1. Batch processing means process each transaction files (import data,

process/cancel orders) at end of each day, then combine results at the end of 10 days.
2. After demo, no need to keep loading initial data into MySQL and MongoDB.

- Implement daily batch processing of order transactions
- Transform transaction data into the `orders` table and `order_line_items` table
- In `orders` table, sort data by **order_id**

order_id	order_datetime	customer_id	total_amount	num_items
3934	2024-02-02T17:15:04.912637	14	26.3	2
6833	2024-02-07T04:11:37.499569	25	73.4	5
7079	2024-02-04T10:08:29.441517	7	83.04	4
9026	2024-02-02T15:44:46.219967	6	80.37	4
26273	2024-02-04T10:19:59.278406	18	24.68	4
28062	2024-02-05T02:47:16.969907	15	7.11	2
62049	2024-02-06T12:56:34.326163	20	37.15	3

- In `order_line_items` table, sort order items by **order_id** and **product_id**

order_id	product_id	quantity	unit_price	line_total
5413	7	3	1.18	3.54
5413	26	2	2.03	4.06
5413	34	5	0.68	3.4
19857	9	1	5.62	5.62
19857	29	1	7.26	7.26
23176	11	4	1.4	5.6
23176	20	3	1.37	4.11
25950	5	5	3.55	17.75
25950	15	3	1	3

- As you process each order, remove order items that have **"qty": null**
- You also need to check the inventory level for each item. If the inventory is insufficient to fulfill the order:
  - Cancel the item (not the entire order) and print a message to the terminal
  - In the `order_line_items` table, set both **quantity** and **line_total** to 0

1	order_id	product_id	quantity	unit_price	line_total
1140	43883977	15	4	3.93	15.72
1141	6131095	15	1	3.93	3.93
1142	71402574	15	1	3.93	3.93
1143	60740432	15	2	3.93	7.86
1144	90112	15	5	3.93	19.65
1145	25486887	15	0	3.93	0
1146	21468208	15	0	3.93	0
1147	85828162	15	2	3.93	7.86

    - Save both the processed `orders` table and `order_line_items` table as CSV files, ensuring that file names, column names, and column ordering match the expected format exactly

3. Data Analytics
    - After processing all 10 batches, create a `daily_summary` table that summarizes the daily order statistics

date	num_orders	total_sales	total_profit
2/1/24	185	7800.59	3427.15
2/2/24	1121	47177.79	20882.56
2/3/24	485	22620.82	10034.27
2/4/24	1499	62000.09	27430.27
2/5/24	908	38347.65	16947.58
2/6/24	1387	59034.42	26217.82
2/7/24	687	29462.84	13090.24
2/8/24	1485	63622.61	28014.4
2/9/24	1106	45560.6	20260.71
2/10/24	994	37466.74	16833.79

    - Dynamically update the inventory levels without restocking any products, and save the final product inventory in the `products_updated` table. Sort the table by **product_id**. Note that some products' inventory may be 0

1	product_id	product_name	current_stock
10	9	Drizzle Swirl Animals	2468
11	10	Glitter Swirl Animals	268
12	11	Drizzle Swirl Rings	0
13	12	Glitter Swirl Rings	51
14	13	Powdered Sugar Bubble Balls	264
15	14	Sugar-Free Coat Bubble Balls	0

- Forecasting sales:
  - Send the daily sales and profit values to the time series forecasting model
  - Do not modify the time series forecasting model
  - Print the MAE and MSE values of the forecast. If `daily_summary` passes CI check, the forecasting metrics should match the expected values.
  - Save forecasting results into `sales_profit_forecast` table, it only has one row.

date	forecasted_sales	forecasted_profit
2/11/24	28878.64	12369.52

4. Apache Airflow DAG
   - Successfully set up and run Example_DAG
   - Create a new Python file to implement a DAG that processes orders (following the same workflow as `main.py`)
     - Load configuration files
     - Import data from MySQL and MongoDB
     - Process orders in batches and generate the required tables
     - Forecast sales and profits, then display forecasting metrics