Table of Contents

[Design of solution 1](#_Toc79335281)

[Order data specification 1](#_Toc79335282)

[Step 1 – Ingestion 2](#_Toc79335283)

[Step 2 – transformation and loading 2](#_Toc79335284)

[Step 3 – transformation and loading 2](#_Toc79335285)

Providing moving average of the last N elements added

The “Pizza House” case study covers a lot topic if all mentioned detail is implement. Below solution will focus on order processing, and describe process from ingest raw data to getting the average calculated. At the end, other structures will also be described at the end

# Design of solution

## Order data specification

* Each order consists of :
  + Foreign keys to relate deals, sales representative, product
  + Order lines to describe quantity and amount, and dollar amount
  + Store key that order is put
  + Order timestamp

Staging area

* Csv format
* External hive table

da

Raw Data

Data Mart

* Partitioned
* Bucketing
* Parquet
* Basic Transformation
* Type casting
* Reporting
* Moving AVG

# Step 1 – Ingestion

* Source data is simulated in CSV format
* Uploaded into Databricks DBFS using Azure upload function
* Data type is inferred during table creation
* Batch is simulated in different batch files: store\_order\_txn\_1.csv, store\_order\_txn\_1.csv, …etc.
* Landing table is recreated every time landing a new batch file
* Table: store\_order\_txn
* Notebook: 01\_ingest\_raw

# Step 2 – transformation and loading

* Data type casting
* Insert into partition table while bucketing
* It helps to retrieve data fast by
  + Partition data by store and date
  + Inside each partition, data is clustered by order and sort by capture time
* If case of re-processing historical data, partitions to be re-processed could be dropped and reloaded.
* Table: edw\_store\_order\_txn\_bucket
* Notebook: 02\_load\_edw
* Below table creation is in Notebook: 00\_prepare

|  |
| --- |
| CREATE TABLE edw\_store\_order\_txn\_bucket  ( order\_number string,  order\_type string,  product\_key string,  deal\_key string,  sales\_rep\_key string,  order\_line\_number int,  order\_line\_quantity int,  order\_create\_ts timestamp,  line\_gross\_dollar\_amt decimal,  line\_discount\_dollar\_amt decimal,  line\_net\_dollar\_amt decimal,  capture\_ts timestamp,  store\_key string,  capture\_date date  )  **USING PARQUET**  **PARTITIONED BY** (store\_key, capture\_date)  **clustered by**(order\_number) **sorted by** (capture\_ts) **into 4 buckets**; |

# Step 3 – transformation and loading

* To get moving average of **last N(2) orders’ net dollar amount per store**
  + From table: edw\_store\_order\_txn\_bucket
  + Getting total dollar amount for each order
  + Define window specification for last 2 order: N=2  
    **window** w as (partition by osum.**store\_key** order by osum.**capture\_ts desc** **rows between 1 preceding and current row**)
  + Specify last N elements in above window specification
* Notebook: 03\_moving\_avg

# Other structures

Beside store order data, there are other structures that can describe the case better.

* Store inventory snapshot
* Store inventory transaction
* Store product snapshot
* Supporting dimension tables:
  + Store dimension: detail info about all stores
  + Material dimension: detail material info about all inventories
  + Product dimension: detail product info
  + Deal dimension
  + Sales Representative dimension
* To get required moving average for above scope, the process is similar to what describe for store order data.