Graduation project from data management track

Big data case study

This project is done by :

Eslam Fayez

Mahmoud hatem

Omar ramadan

Shrorouk abdelraouf

The three stages

Introduction :

This projects involve handling data from a retail company

This is the shape of the received data :

Source system pushes 3 files every hour (branches file, sales agents file, sales transactions file). For the sales transactions file, it contains both branches and online sales. For

branches and sales agents' files, may contain new entries at any point of time.

We applied the "Medallion architecture" in handling our data

It involves 3 stages :

Bronze layer , silver layer and gold layer

In the next pages we will show how we applied each stage

1-Bronze layer :

-Here we move the files in its raw format from the local file path to the hdfs every hour using a crontab job which makes the script running in an automated way every one hour

-We store the files which have the same topic with each other

, so we created a script checks on the name of each moved file to decide where this file will be stored

-There are 3 folders for storing the files (transaction , sales agent , branch)

-It checks if this is already stored in hdfs or not

2-Sliver layer :

Processing the transaction file :

1 - Get_latest_file function :

it takes a directory as an input and checks the modification time of every file in this directory and return the latest file modified in this directory

2 - write_checkpoint function :

it takes 2 arguments : checkpoint path (refers to the path of the checkpoint file) and latest file name

note : checkpoint file : it is a file contains the names of all processed files

this function writes the name of the latest processed file in the checkpoint file

3 - fetch_last_sur function :

it takes 2 arguments : table name and column name

its target is to get the last or the maximum surrogate key from a specific column in a spark data frame using SQL query to use this value in appending new records to this table

4 - write_last_sur function :

it takes 2 arguments : table name , column name and last surrogate key

this function takes a surrogate key value and writes it as a single row to a specified Spark table. It overwrites any existing data in the table

5 – creating a schema for the transaction parquet file

6 - create_audit_dimension function :

It takes 2 arguments (transactions dataframe, audit dimension

It uses fetch last sur to generate the new surrogate key

The target from this function is extracting some metadata from the transaction file to enrich the audit dimension

Examples of the metadata in this case :

what is the source file ?

When is this record created ?

Who is created the record ?

Is this transaction valid ?

Does this transaction contains a valid email ?

This functions records all of this metadata to the audit dimension

7 – Get hash function :

It hashes the values of multiple columns in one value to use it to detect if there is a change in a value of one of these columns or not

8 – column renamer function :

It renames a column by adding a suffix to it to specify if it is historical or current

9 – the processing step :

Here we call the get lastest file() function to search in a specific directory and gets the latest file

Then checks for the name of the latest processed file in the checkpoint file

If the both files were the same then it will print (File already processed before)

And if there is a difference between the files

then the processing process starts :

1 –processing the email column by replacing (\.com.) with (.com)

2 – creating one column called offer to replace the 5 offers columns

by checking the value of these 5 columns

3- calculating a new column (total price) by multiplying (units) by (unit price) and applying the offer discount if it is found

4 – splitting the shipping address to extract city , state and postal code

5 – dropping the useless columns after transformations like the old address and the 5 offers columns

6 – creating audit dimension using create audit dimension() function

7 – calling write checkpoint() function to write the name of the latest processed file in the checkpoint file

8 – calling write_last_sur () function.

It writes the retrieved last_audit_id (maximum audit surrogate key) to (audit_dim) in the column (audit_sur)

It is a way to track the last processed audit ID

10 - Showing the columns types of transaction data frame after processing to validate them

3- Gold layer :

Here we put the data in a data warehouse to make it ready for analytical queries

-Data warehouse design :

After putting all the required business queries in our consideration we decided to create 3 different data marts :

1 – super data mart : which has fact table has all transaction data whether it is an online transaction or it is a branch transaction

And this data mart has also these 5 dimensions :

Customer , product , audit , sales agent and branch

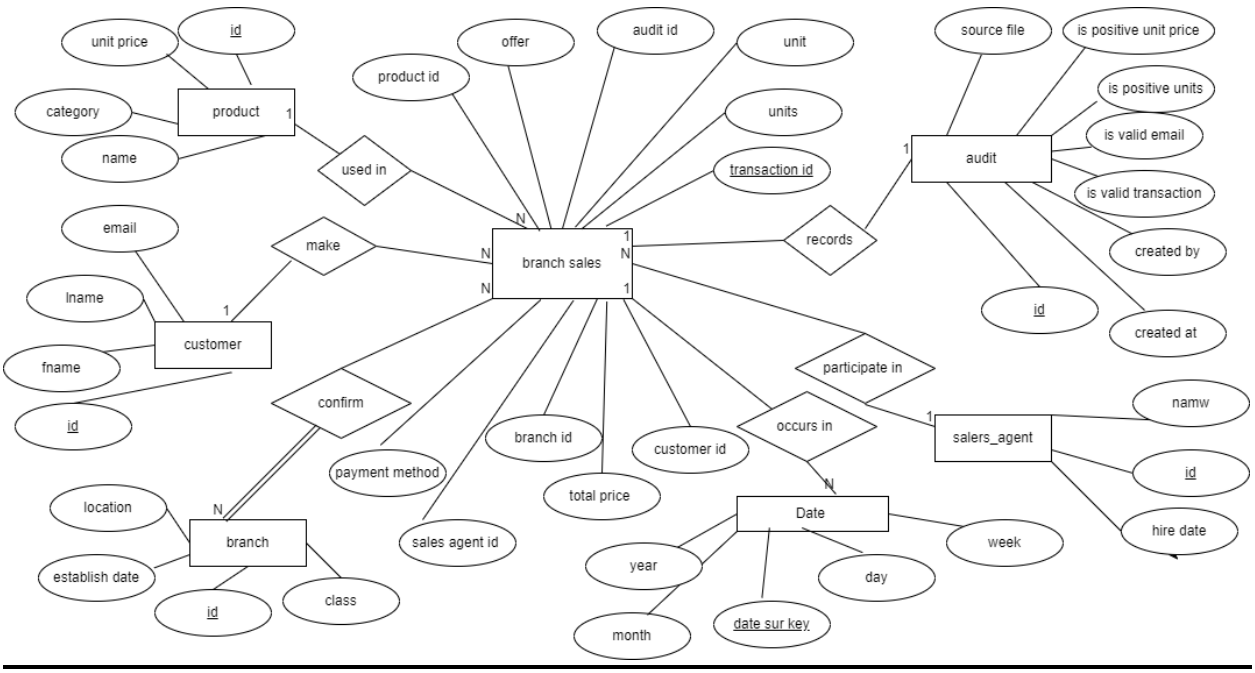
2- branch data mart : it contains the same dimensions as the first data mart but its fact table contains only transactions that were made from a branch

3 online data mart : it contains the same dimensions as the other data marts except the branch dimension

Its fact table has only online transactions

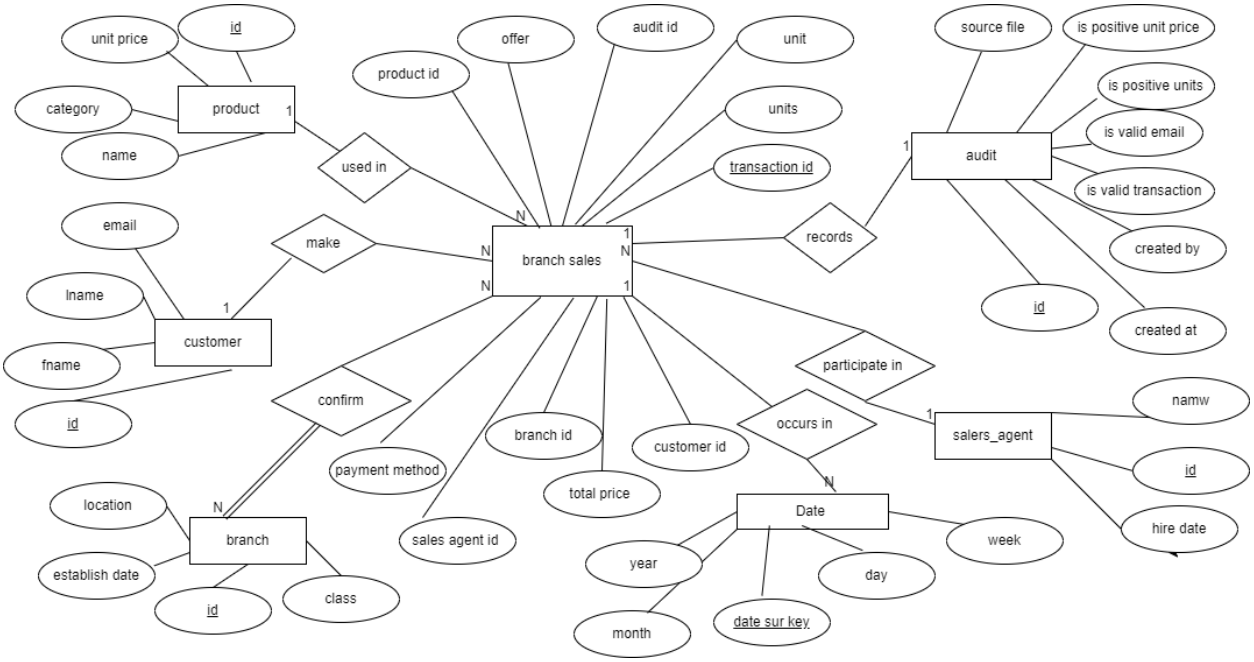
And here we will show the schema design of each data mart from the three

1 – super data mart :

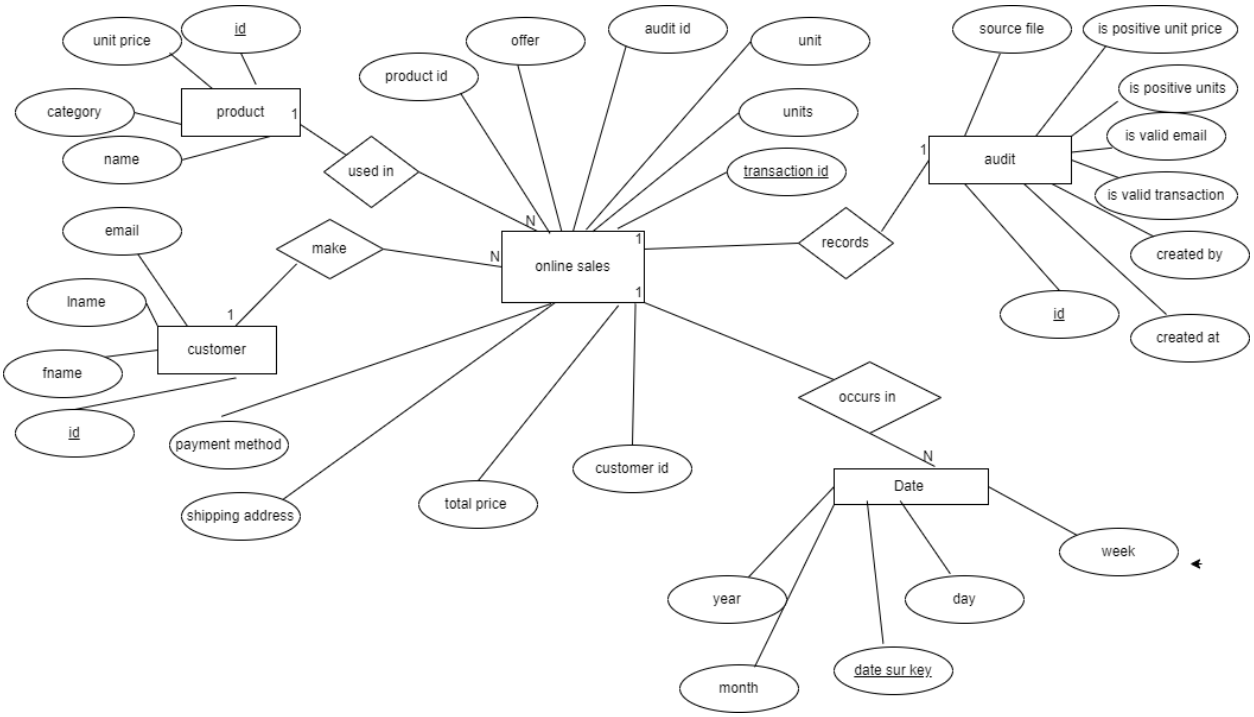


---

2 – branch data mart :



3 – online data mart :



Business requirements:

- You're Representing The Business Team! Read Your Data Carefully And Build Your Own DWH Model With Respect To Remaining Requirements.
  We followed the medallion approach in handling the data.
  Our schema consists of three fact tables (super_fact, online_fact, fact_branches), and the dimension tables are (dim_product, dim_customer, dim_audit, dim_sales_agent, dim_branch).

- Most Query Condition Will Be Used From Teams (Transaction Date)
  So we made dynamic partitions on the column transaction date in the three fact tables (super_fact, online_fact, factbranches) when creating hive tables.

```
# Save the DataFrames as external Hive tables
online_fact.write\
    .mode("overwrite")\
    .partitionBy("transaction_date")\
    .saveAsTable("online_sales_schema.online_fact")
```

- Total Paid Price After Discount Should Be Added As Column In Fact Table

- The Marketing Team Needs To Know Most Selling Products, Most Redeemed Offers From Customers And Most Redeemed Offers Per Product

First : Most Selling Products

Query :

running it on hive for the sales schema (representing the entire sales), online_sales_schema and branches_sales_schema.

```
hive> Select d.product_name , sum(f.units) as product_quantity
    > From sales_schema.dim_product as d
    > join
    > sales_schema.super_fact as f
    > Using ( product_id )
    > Group by d.product_name
    > Order by product_quantity desc;
```

Result:

```
Boots   31662
Sandals 23051
Headphones      23002
Sneakers        20008
Toaster 19024
Blender 18971
Electric Kettle 17050
Blouse  16995
Washing Machine 16576
Microwave       15960
Hair Straightener       15848
Vacuum Cleaner  14999
Printer 14892
Smartphone      14550
Jeans   14500
T-Shirt 13867
Camera  13107
Tablet  12954
Skirt   12768
Coffee Maker    12642
Hair Dryer      12098
Heels   10902
Hoodie  9417
Iron    9266
Monitor 8159
Laptop  7904
Dress   7659
TV      7640
Time taken: 166.994 seconds, Fetched: 28 row(s)
```

Query:

```
hive> Select d.product_name , sum(f.units) as product_quantity
    > From online_sales_schema.dim_product as d
    > join
    > online_sales_schema.online_fact as f
    > Using ( product_id )
    > Group by d.product_name
    > Order by product_quantity desc;
```

Result:

```
Boots   8603
Microwave       7000
Hair Straightener       6720
Blender 6527
Headphones      6386
Blouse  6270
Coffee Maker    6027
Toaster 5974
Sneakers        5795
Sandals 5716
Washing Machine 5656
Vacuum Cleaner  5353
Hair Dryer      4738
Heels   4324
T-Shirt 4263
Smartphone      4200
Iron    4100
Electric Kettle 3960
Jeans   3800
Camera  3774
Tablet  3774
Hoodie  3354
Skirt   3264
Laptop  3230
Printer 3213
Monitor 2747
TV      2120
Dress   1480
Time taken: 409.36 seconds, Fetched: 28 row(s)
```

Query:

```
hive> Select d.product_name , sum(f.units) as product_quantity
    > From branches_sales_schema.dim_product as d
    > join
    > branches_sales_schema.factbranches as f
    > Using ( product_id )
    > Group by d.product_name
    > Order by product_quantity desc;
```

Result:

```
Boots   23059
Sandals 17335
Headphones      16616
Sneakers        14213
Electric Kettle 13090
Toaster 13050
Blender 12444
Printer 11679
Washing Machine 10920
Blouse  10725
Jeans   10700
Smartphone      10350
Vacuum Cleaner  9646
T-Shirt 9604
Skirt   9504
Camera  9333
Tablet  9180
Hair Straightener       9128
Microwave       8960
Hair Dryer      7360
Coffee Maker    6615
Heels   6578
Dress   6179
Hoodie  6063
TV      5520
Monitor 5412
Iron    5166
Laptop  4674
Time taken: 158.845 seconds, Fetched: 28 row(s)
```

Second: Most Redeemed Offers per Customer

Query:

```
hive> Select (c.customer_fname) || (c.customer_lname ) as customer_name , count (f.offers) as num_of_offers
    > From sales_schema.dim_customer as c
    > Join sales_schema.super_fact as f
    > ON c.customer_id = f.customer_id
    > Where offers <> 0
    > GROUP BY c.customer_fname, c.customer_lname
    > Order by num_of_offers desc;
```

Result:

```
JamesJohnson    666
MichaelBrown    570
AlexanderWilliams       451
WilliamTaylor   389
JohnJohnson     382
WilliamDavis    347
AvaSmith        344
MiaDavis        339
AlexanderWilson 334
MichaelJones    326
OliviaBrown     322
JamesJones      312
AvaMoore        299
JohnBrown       289
OliviaDavis     279
AvaMiller       273
AvaWilliams     265
JamesDavis      255
JamesSmith      246
AlexanderJohnson        228
WilliamWilson   199
AvaBrown        198
SophiaMiller    196
MiaJohnson      187
MiaTaylor       180
OliviaJones     180
MichaelMoore    170
MiaWilson       162
AlexanderMoore  162
AvaWilson       153
AvaJones        153
EmmaMiller      153
AlexanderJones  150
MichaelMiller   145
WilliamBrown    140
AlexanderMiller 135
EmmaBrown       120
WilliamMoore    120
MichaelWilson   120
OliviaMoore     119
EmmaJohnson     114
JohnTaylor      112
MiaWilliams     112
MichaelWilliams 112
JohnMoore       96
SophiaWilson    91
JamesMoore      90
```

Query:

```
hive> SELECT offers, COUNT(customer_id) AS offer_count
    > FROM sales_schema.super_fact
    > WHERE offers <> 0
    > GROUP BY offers
    > ORDER BY offer_count DESC;
```

Result:

```
OK
3       168
1       153
4       145
2       144
5       126
Time taken: 146.431 seconds, Fetched: 5 row(s)
```

Third: most redeemed offers per product

Query:

```
hive> Select p.product_name , count (f.offers) as num_of_offers
    > From sales_schema.dim_product as p
    > Join sales_schema.super_fact as f
    > Using ( product_id )
    > Where offers <> 0
    > Group by p.product_name
    > Order by num_of_offers desc;
```

Result:

```
Boots   2708
Blender 2013
Sandals 1943
Hair Straightener       1904
Electric Kettle 1760
Printer 1734
Headphones      1674
Sneakers        1647
Camera  1581
Microwave       1512
Blouse  1485
Washing Machine 1456
Toaster 1450
Vacuum Cleaner  1378
Smartphone      1350
Coffee Maker    1225
T-Shirt 1225
Jeans   1200
Skirt   1104
TV      960
Hair Dryer      920
Heels   920
Tablet  867
Iron    779
Monitor 738
Hoodie  731
Laptop  646
Dress   592
Time taken: 229.521 seconds, Fetched: 28 row(s)
```

- **The marketing team needs to know which lowest cities in online sales to run more campaigns.**

  Query:

```
hive> Select city , count( transaction_id ) as num_of_transactions
    > From online_sales_schema.online_fact
    > Group by city
    > Order by num_of_transactions asc;
```

  Result:

```
OK
Carlsbad        1
Calhoun 1
Burlington      1
Burke   1
Brookline       1
Wheat Ridge     1
Westford        1
West Windsor    1
Weathersfield   1
Waterbury Center        1
Alameda 1
Waltham 1
Underhill       1
Tyngsborough    1
Tyndall Air Force Base  1
Tollhouse       1
Thunderbolt     1
Thetford        1
South Hadley    1
Shelburne Falls 1
Severn  1
Ashland 1
San Leandro     1
San Jose        1
Salisbury       1
Salinas 1
Sacramento      1
Rutland 1
Riverview       1
Richmond        1
Revere  1
Pueblo West     1
Port Charlotte  1
Winchester      1
Pittsfield      1
Peabody 1
Athol   1
Baldwin Park    1
Cambridge       1
North Little Rock       1
Newton  1
Newport 1
Needham 1
Barre   1
Williston       1
Belvidere       1
Middleton       1
```

Also, we have added some other queries:

First: most used payment method

Query:

```
hive>
    > Select payment_method , count (*) as count_of_method
    > From sales_schema.super_fact
    > Group by payment_method
    > Order by count_of_method desc;
```

Result:

```
Credit Card     644
Cash    514
Stripe  176
PayPal  166
```

Second : Highest Branch In Sales

Query:

```
hive> Select branch_id , sum( total_price ) as sales
    > From branches_sales_schema.factbranches
    > Group by branch_id
    > Order by sales desc;
```

Result:

```
4       221458.28000000017
1       213983.77000000002
5       202768.03
2       179407.8900000001
3       151176.51999999993
```

Third: Most Product Sold By Quantity

Query: running it on the three schemas

```
hive> Select p.product_name , sum(f.units) as total_units
    > From sales_schema.super_fact as f
    > Join sales_schema.dim_product p
    > On p.product_id = f.product_id
    > Group by p.product_name
    > Order by total_units desc;
```

Result:

```
Boots   31662
Sandals 23051
Headphones      23002
Sneakers        20008
Toaster 19024
Blender 18971
Electric Kettle 17050
Blouse  16995
Washing Machine 16576
Microwave       15960
Hair Straightener       15848
Vacuum Cleaner  14999
Printer 14892
Smartphone      14550
Jeans   14500
T-Shirt 13867
Camera  13107
Tablet  12954
Skirt   12768
Coffee Maker    12642
Hair Dryer      12098
Heels   10902
Hoodie  9417
Iron    9266
Monitor 8159
Laptop  7904
Dress   7659
TV      7640
```

Query:

```
hive> Select p.product_name , sum(f.units) as total_units
    > From branches_sales_schema.factbranches as f Join sales_schema.dim_product p
    > On p.product_id = f.product_id
    > Group by p.product_name
    > Order by total_units desc
    > ;
```

Result:

```
Boots   23059
Sandals 17335
Headphones      16616
Sneakers        14213
Electric Kettle 13090
Toaster 13050
Blender 12444
Printer 11679
Washing Machine 10920
Blouse  10725
Jeans   10700
Smartphone      10350
Vacuum Cleaner  9646
T-Shirt 9604
Skirt   9504
Camera  9333
Tablet  9180
Hair Straightener       9128
Microwave       8960
Hair Dryer      7360
Coffee Maker    6615
Heels   6578
Dress   6179
Hoodie  6063
TV      5520
Monitor 5412
Iron    5166
Laptop  4674
```

Query:

```
hive> Select p.product_name , sum(f.units) as total_units
    > From online_sales_schema.online_fact as f
    > Join sales_schema.dim_product p
    > On p.product_id = f.product_id
    > Group by p.product_name
    > Order by total_units desc;
```

Result:

```
Boots    8603
Microwave       7000
Hair Straightener       6720
Blender 6527
Headphones      6386
Blouse   6270
Coffee Maker    6027
Toaster 5974
Sneakers        5795
Sandals 5716
Washing Machine 5656
Vacuum Cleaner  5353
Hair Dryer      4738
Heels    4324
T-Shirt 4263
Smartphone      4200
Iron     4100
Electric Kettle 3960
Jeans    3800
Camera   3774
Tablet   3774
Hoodie   3354
Skirt    3264
Laptop   3230
Printer 3213
Monitor 2747
TV       2120
Dress    1480
```

- **B2B team needs a daily dump (csv file) that contains (sales_agent_name ,product_name, total_sold_units) and this file should be sent to local file system.**

```
daily=spark.sql("""
Select s.name, p.product_name, sum(f.units) as count_of_units
From sales_schema.dim_product as p
Join sales_schema.super_fact as f On p.product_id = f.product_id
Join sales_schema.dim_sales_agent as s On s.sales_person_id = f.sales_agent_id
Group by s.name, p.product_name
Order by count_of_units desc
""")
```

name	product_name	count_of_units
Daniel Martinez	Boots	1431222
Olivia Davis	Sandals	1247010
Olivia Davis	Sneakers	1242428
Emma Taylor	Boots	1027270
Sophia Moore	Printer	986742
Christopher Miller	Boots	957162
Emma Taylor	Blender	950508
Jane Smith	Headphones	901747
Jane Smith	Smartphone	874388
Emily Brown	Sandals	841540
Christopher Miller	Toaster	817652
John Doe	Boots	787226
Michael Johnson	Sandals	786410
Jane Smith	Boots	772302
John Doe	T-Shirt	771708
Sophia Moore	Sandals	769690
John Doe	Sandals	755100
Sophia Moore	Headphones	754523