TREND ANALYSIS

RECOMMENDATION OF POPULAR TRENDS BY FURNITURE CATEGORY

Group 4

TABLE OF CONTENTS

ACQUISITION & INGESTION

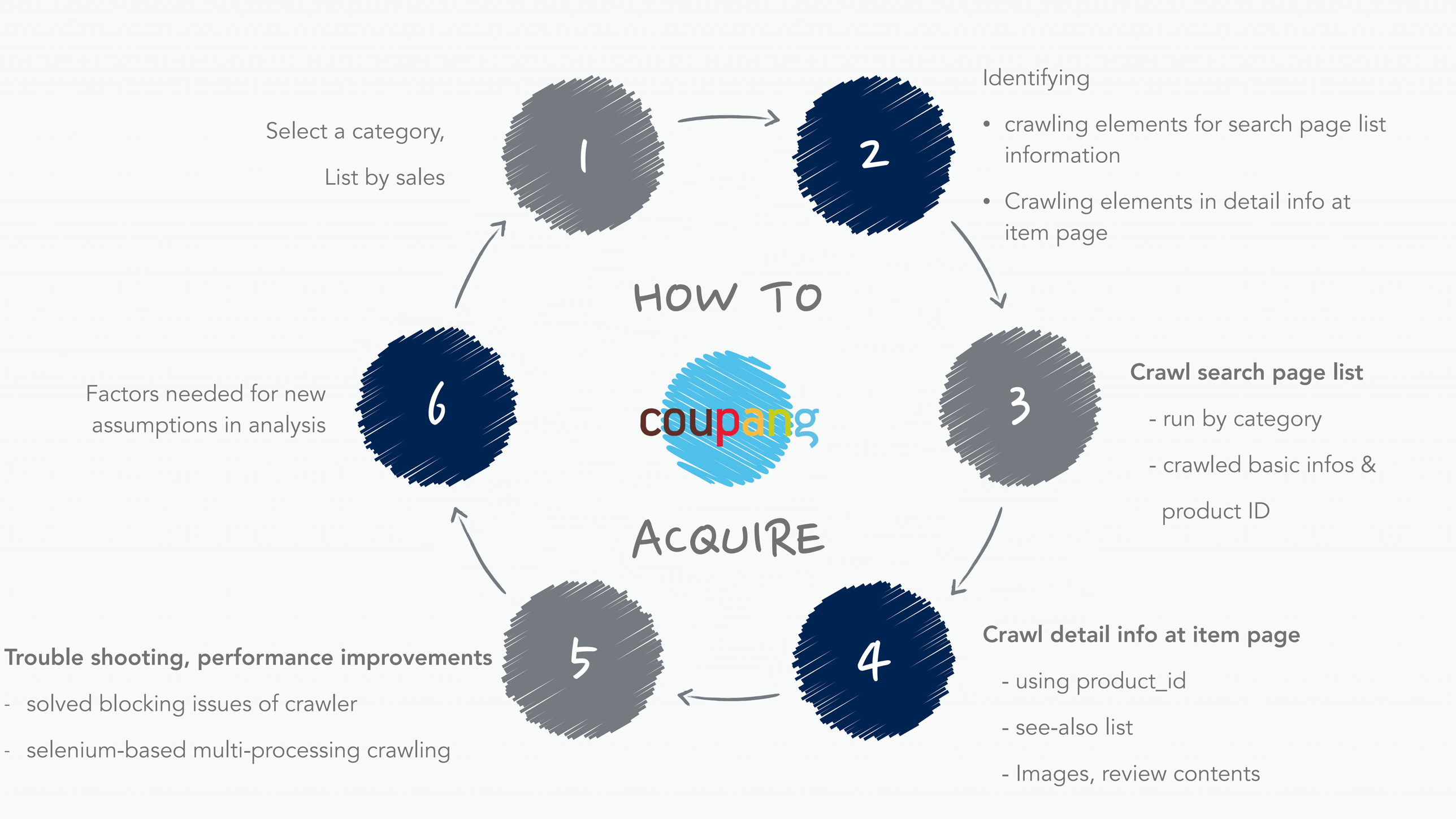
DATA SOURCE SELECTION

NAVER SHOPPING

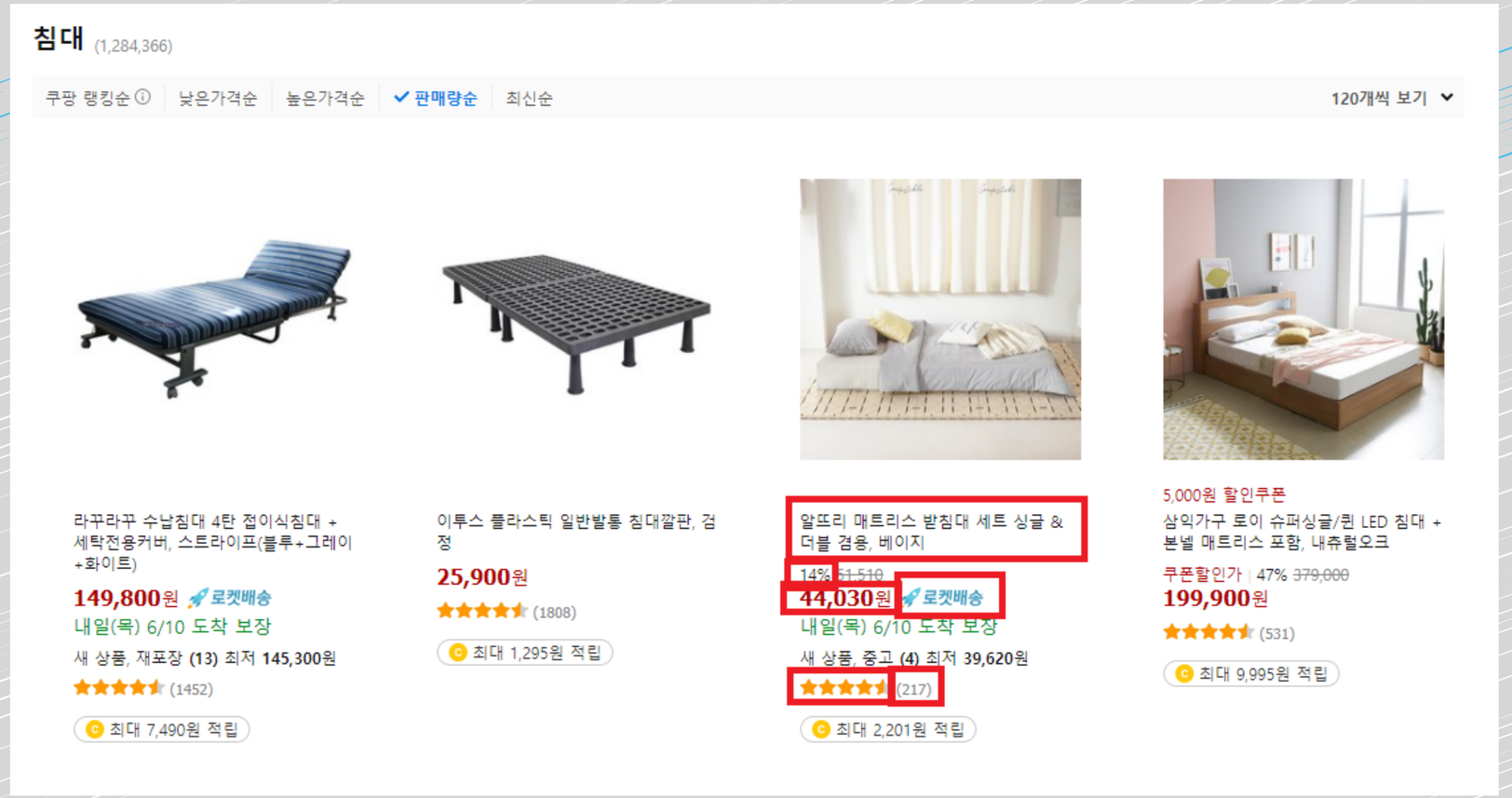
- **Target** : the number of item sold
- **Feature** : product name, price, # of reviews, # of steams, shipping fee, seller rating
- **Cons**
  - Difficult to obtain raw data over 500MB
    - a few items that have the number of item sold
  - limited # of feature
    - Feature selection is limited because the detail pages are not unified

COUPANG

- **Target** : list sorted by the # of item sold
- **Feature** : A total of 14 features including features of Naver shopping, discount rate, whether it is out of stock, internal characteristics list, also-viewed item
  - remedy Naver's shortcomings


Select a category,
List by sales
1
Identifying
• crawling elements for search page list information
• Crawling elements in detail info at item page
2
HOW TO
coupang
ACQUIRE
3
Crawl search page list
- run by category
- crawled basic infos & product ID
Crawl detail info at item page
- using product_id
- see-also list
- Images, review contents
4
Trouble shooting, performance improvements
- solved blocking issues of crawler
- selenium-based multi-processing crawling
5
Factors needed for new assumptions in analysis
6

WHAT WE ACQUIRED



FROM ITEM LIST

1

장바구니 담기

바로구매 >

- 사이즈: 싱글
- 접이 가능여부: 접이식가능
- 각도조절 여부: 각도조절 가능
- 바퀴 유무: 바퀴있음
- 구성품: 본체 + 바퀴 + 사용설명서 + 세탁커버
- 쿠팡상품번호: 10755024 - 46602774

새 상품 (12) / 재포장 (1) 최저 145,300원 >

FROM DETAIL_INFO LIST

라꾸라꾸

라꾸라꾸 수납침대 4탄 접이식침대 + 세탁전용 커버

★★★★½ 1,452개 상품평

149,800원 로켓배송

최대 7,490원 적립

무료배송 다른 판매자 보기(6)

내일(금) 6/11 도착 보장 (23시간 53분 내 주문 시 / 서울·경기 기준)

라꾸라꾸 수납침대 4탄 접이식침대 + 세탁전용커버, 스트라이프(블루+그레이+화이트), 싱글(84 x 197 x 32 cm)
신고하기

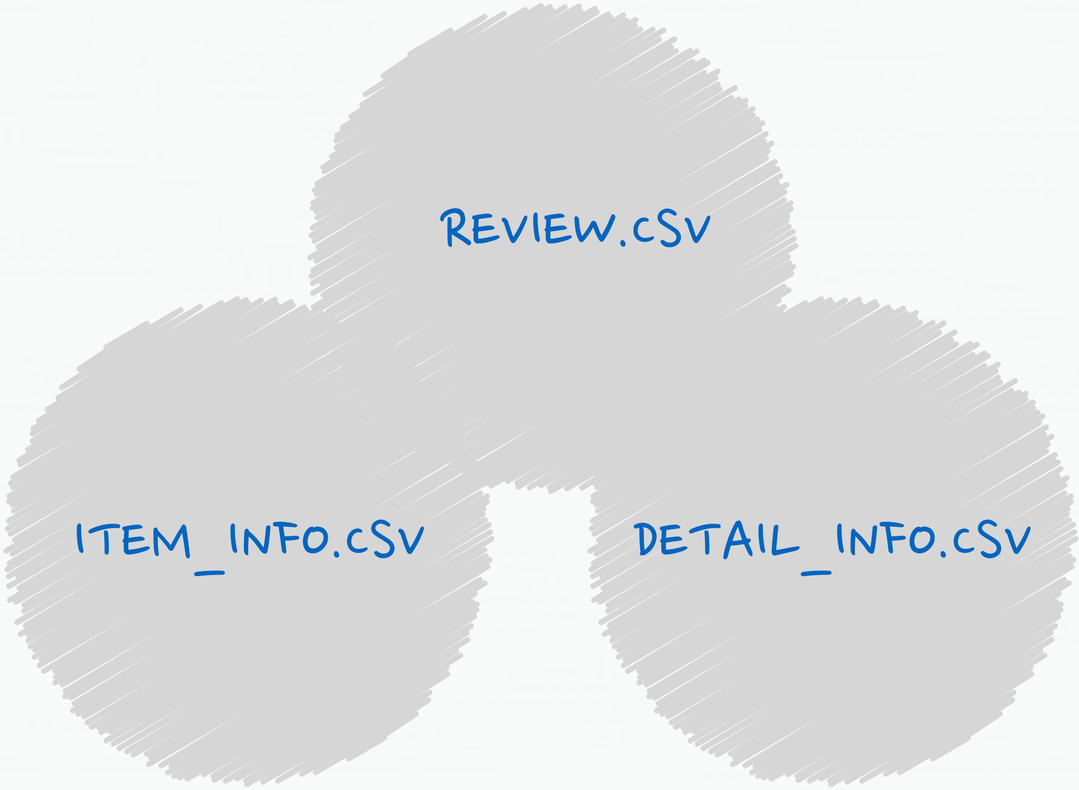
이*도
2021.06.05
라꾸라꾸 수납침대 4탄 접이식침대 + 세탁전용커버, 스트라이프(블루+그레이+화이트), 싱글(84 x 197 x 32 cm)
가성비 부분에서 점수를 많이 드리고 결론적으로 만족합니다.
가격대비 만족할만한 기능성과 견고함입니다.
등받이 기능이 있다는 것도 매우 만족스럽습니다.
설치도 매우 간단하고, 바퀴도 잘 이동이 됩니다.
쉽게 접어 공간 활용에도 용이합니다.

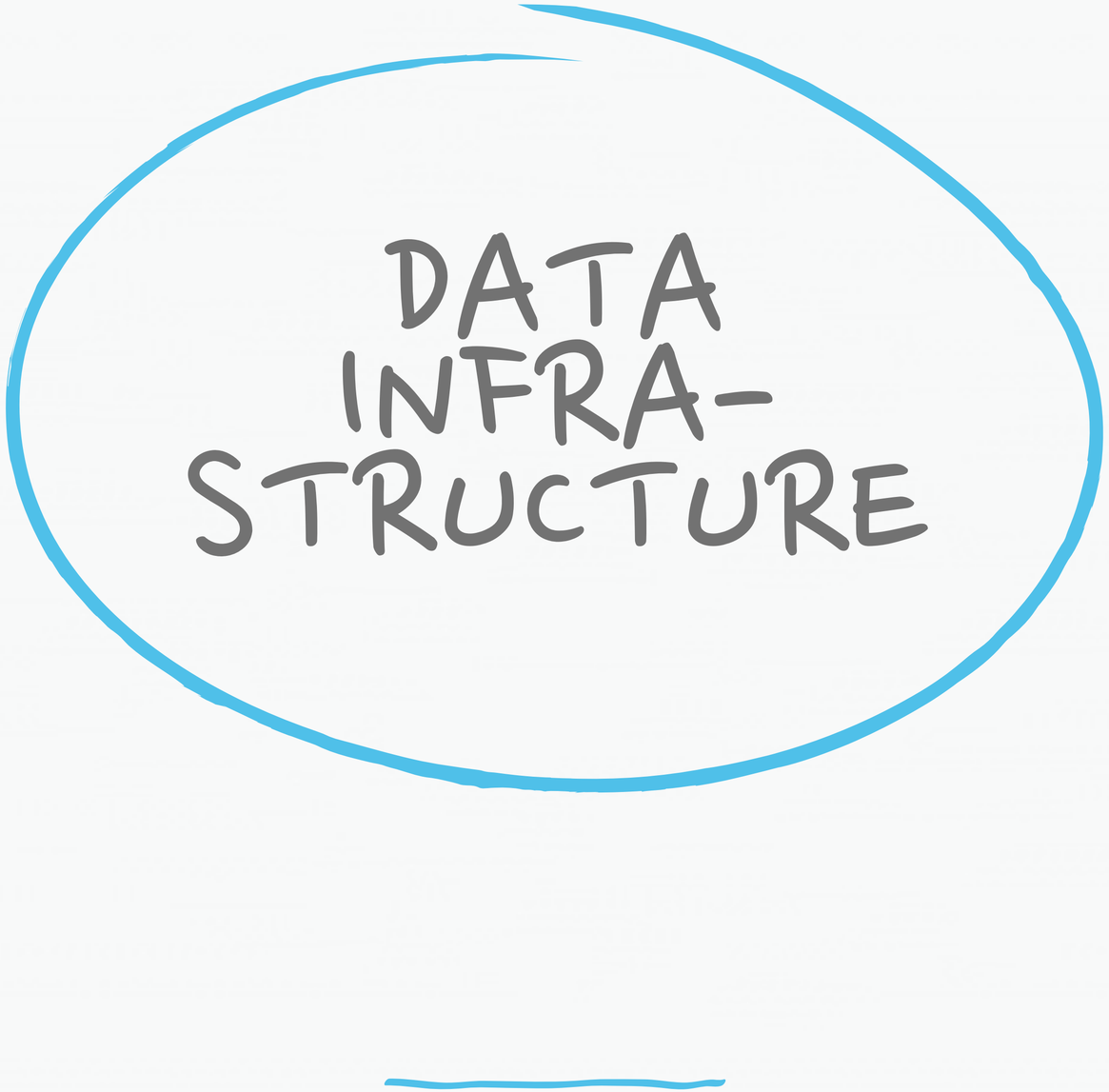
하지만 아무래도 가성비 제품이다 보니 75kg 이하 체중이신 분들이 사용하는것이 무리없이 가장 좋을 것 같습니다.
기본으로 동봉된 매트에만 의존해서 장기간 사용하면 허리에 불편함을 느낄 수 있으므로 적당히 보완이 될만한 이불이나 매트를 2중으로 위에 올려 놓고 사용하면 편안함에도 문제가 없을 것 같습니다.
1명에게 도움 됨
이 상품평이 도움 되었나요?
도움이 돼요 도움 안 돼요 신고하기

FROM REVIEWS

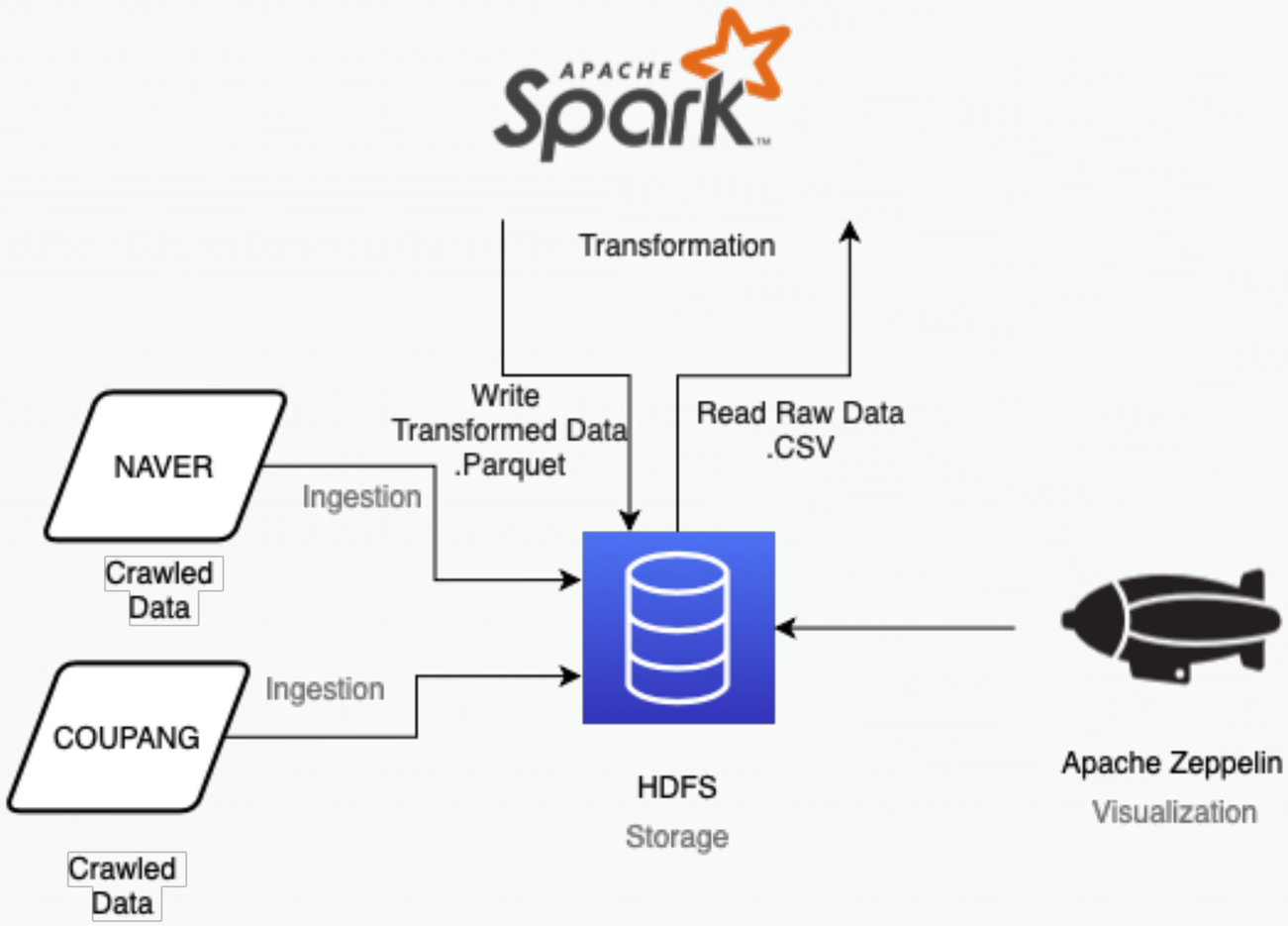
Acquisition Result

Raw data acquisition completed



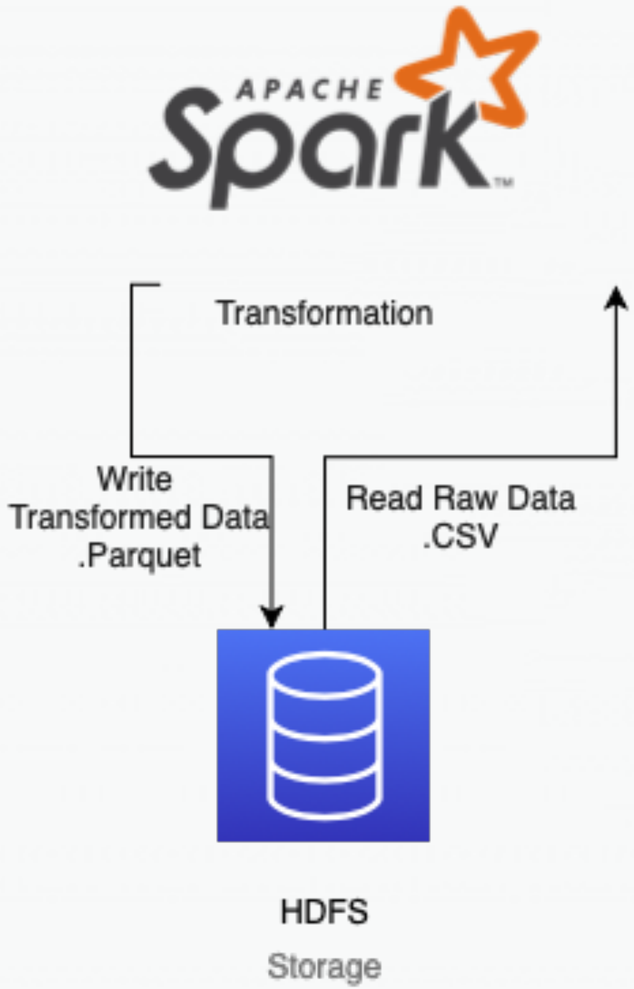

DATA
INFRA-
STRUCTURE

DATA INFRASTRUCTURE SYSTEM



- Data Acquisiton & Ingestion
  - store crawled data from Coupang and Naver on HDFS
- Transformation by **Apache Spark**
  - transform Raw Data(CSV) using **Pyspark**
  - store as **Apache Parquet data type**
- Visualization **with Apache Zeppelin**
  - data analysis using Transformed data

DATA TRANSFORMATION



1. READ RAW DATA FROM HDFS
   - APACHE SPARK

2. DATA PREPROCESSING
   - USING PYSPARK & PANDAS

3. WRITE DATA AS PARQUET DATA TYPE

DATA PREPROCESSING - 1

FUNCTION INITIALIZATION

```
def open_pd_csv(path):
    file_path = 'hdfs://192.168.0.9:9000/' + path
    data = spark.read.csv(file_path, header=True)
    result = data.toPandas()

    return result
```

```
def save_pd_csv(df ,path):
    file_path = 'hdfs://192.168.0.9:9000/' + path
    data = spark.createDataFrame(df)
    data.coalesce(1).write.mode("overwrite").option("header","true").csv(file_path)
```

READ/WRITE RAW DATA
<CSV>

```
def save_pd_parquet(df ,path):

    # ex) user/hadoop/trasformed.parquet
    file_path = 'hdfs://192.168.0.9:9000/' + path
    data = spark.createDataFrame(df)
    data.write \
        .mode("overwrite") \
        .format("parquet") \
        .save("transformed_data.parquet")
```

WRITE TRANSFORMED DATA
<PARQUET>

DATA PREPROCESSING - 2

FUNCTION INITIALIZATION

```python
# spark datafrmae schema 설정
def ad_schema(df):
    intCols = ['rank','category','price',
               'discount_percentage', 'rating_total_count',
               'reviews_for_last1year', 'sales']
    doubleCols = ['rating']
    boolCols = ['rocket_delivery', 'is_out_of_stock']

    for c in df.columns:
        type_col = StringType()
        if c in intCols:
            type_col = IntegerType()
        elif c in doubleCols:
            type_col = DoubleType()
        elif c in boolCols:
            type_col = BooleanType()

        df = df.withColumn(c,df[c].cast(type_col))
    df = df.fillna(0)
    return df
```

```python
def innerTocol(df, attr):
    """ df : detail이라는 column을 가진 데이터프레임 (type : dataframe)
        attr : detail에서 찾아올 내적 특성, column명이 됨 (type : str)
        반환값은 없으며 함수 내부에서 df에 새로운 column이 추가됨 """

    # 추출한 내적특성의 값을 담을 list
    inner = []

    # 내적 특성 값 추출
    for s in df.attribute_list:
        if type(s) != str:
            inner.append('NULL')
        elif s.find(attr) != -1:
            #print(s[s.find(':',s.find(attr))+2:s.find('"',s.find(attr))])
            inner.append(s[s.find(":",s.find(attr))+2:s.find('"',s.find(attr))])
        else:
            inner.append('NULL')

    # 데이터프레임에 새로운 열 추가
    df[attr] = inner
```

DATA PREPROCESSING - 3

MERGE PRODUCT&REVIEW DATA

```
for d in dir_list:
    base = path_dir + d + '/'
    file_list = get_dir(base)
    #파일목록에서 py파일 삭제
    for f in file_list[:15]:
        if '.py' in f:
            file_list.remove(f)

    if 'getReviewData.py' in file_list:
        file_list.remove('getReviewData.py')


    # 상품정보 취합
    data = open_pd_csv(base + file_list[0])

    for file in file_list[1:9]:
        temp = open_pd_csv(base + file)
        data = pd.concat([data,temp])

        # 인덱스 리셋(1~120까지 반복되는 현상 때문에 작성)
        data = data.reset_index()
        data = data.drop(['index'],axis=1)

    # result.csv 목록에서 삭제
    del file_list[len(file_list)-1]
```

```
# 리뷰정보 취합
data2 = open_pd_csv(base + file_list[9])

k = 10

for file in file_list[10:]:
    if (file == 'result.csv') or (file == 'result_review.csv'):
        continue
    print(base, file, k)

    #temp = pd.read_csv(base + file, error_bad_lines=False)
    temp = open_pd_csv(base + file)
    data2 = pd.concat([data2,temp])

    temp = None

    # 인덱스 리셋(1~120까지 반복되는 현상 때문에 작성)
    data2 = data2.reset_index()
    data2 = data2.drop(['index'],axis=1)
    k+=1
```

```
#result 데이터프레임에 상품별 리뷰정보 추가
product_id = file[14:file.find('csv')-1]
df_temp = data2[data2['reg_date'] > '2020.06.01'] # 2020년 6월 1일이후 리뷰만 필터
df_temp = df_temp.groupby('product_id').agg(['count'])
df_temp = df_temp.ratings
data = pd.merge(left = data, right = df_temp, how = 'left', on = 'product_id')

# 중복 제거
data = data.drop_duplicates(['product_id'])

# 상품별 상품정보 취합된 csv파일 저장
save_pd_parquet(data ,base + 'transformed.parquet')

# 상품별 상품정보 취합된 csv파일 저장
save_pd_csv(data ,base + 'result.csv')
```

DATA PREPROCESSING - 4

SPLIT INNER ATTRIBUTE

```
# 상품별 내적특성 설정
attr_list = [['인테리어', '재질', '사이즈', '설치', '색상', '매트리스포함', '침대 프레임'],
             ['사이즈', '색상', '형태', '인테리어', '재질', '접이 가능여부'],
             ['사이즈', '색상', '소재', '재질', '팔걸이', '등받이', '좋류', '인테리어'],
             ['사이즈', '색상', '재질', '인테리어'],
             ['사이즈', '색상', '재질', '인테리어'],
             ['사이즈', '색상', '재질', '인테리어', '스툴', '거울'],
             ['사이즈', '색상', '재질', '인테리어', '행거 종류', '행거 타입', '거울'],
             ['색상', '재질', '인테리어', '논슬립', '묶음 수량', '종류', '회전'],
             ['개당 용량', '개폐방식', '재질', '투명 여부', '접이 가능여부', '바퀴 유무', '손잡이'],
             ['사이즈', '색상', '재질', '형태', '바퀴'],
             ['사이즈', '색상', '발통', '매트/토퍼', '두께', '겉감재질', '충전재'],
             ['색상', '재질', '접이', '바퀴', '사용인원'],
             ['색상', '재질', '인테리어'],
             ['사이즈', '색상', '재질', '소재', '인테리어', '방식', '각도조절'],
             ['사이즈', '잠금방식', '중량', '경보기', '지문인식'],
             ['단', '재질', '형태'],
             ['사이즈', '색상', '재질', '인테리어'],
             ['사이즈', '색상', '재질', '소재', '스툴', '사용인원', '설치', '인테리어'],
             ['사이즈', '색상', '재질', '인테리어', '바퀴', '단'],
             ['사이즈', '색상', '재질', '인테리어', '가로길이', '형태']]

item_list = ['bed', 'bedtray', 'chair', 'closet', 'drawers', 'dressingtable,console',
            'hanger', 'hanger,doorhook', 'livingbox', 'livingroom table', 'mattress',
            'outdoor furniture', 'partition', 'recliner', 'safe', 'shoes shelf', 'smalltable',
            'sofa', 'storage', 'table']
```

```
for d in dir_list:
    base = path_dir + d + '/'
    file_name = 'result.csv'
    item = d[:d.find('(')-1]
    data = open_pd_csv(base + file_name)
    attrs = inner_attr[item]
    for a in attrs:
        innerTocol(data, a)

    # 필요없는 열 삭제
    data = data.drop('attribute_list',axis=1)
    data = data.drop('baby_product_link',axis=1)
    data = data.drop('recommends_list',axis=1)

    # 색상 열에서 '계열'문자열 모두 삭제
    if '색상' in data.columns:
        data["색상"] = data["색상"].str.replace("계열","")
    data = data.rename(columns = {"count":"reviews_for_last1year"})

    save_pd_csv(data ,'user/hadoop/data_analysis/merged_data/'+ item + '_result.csv')
```

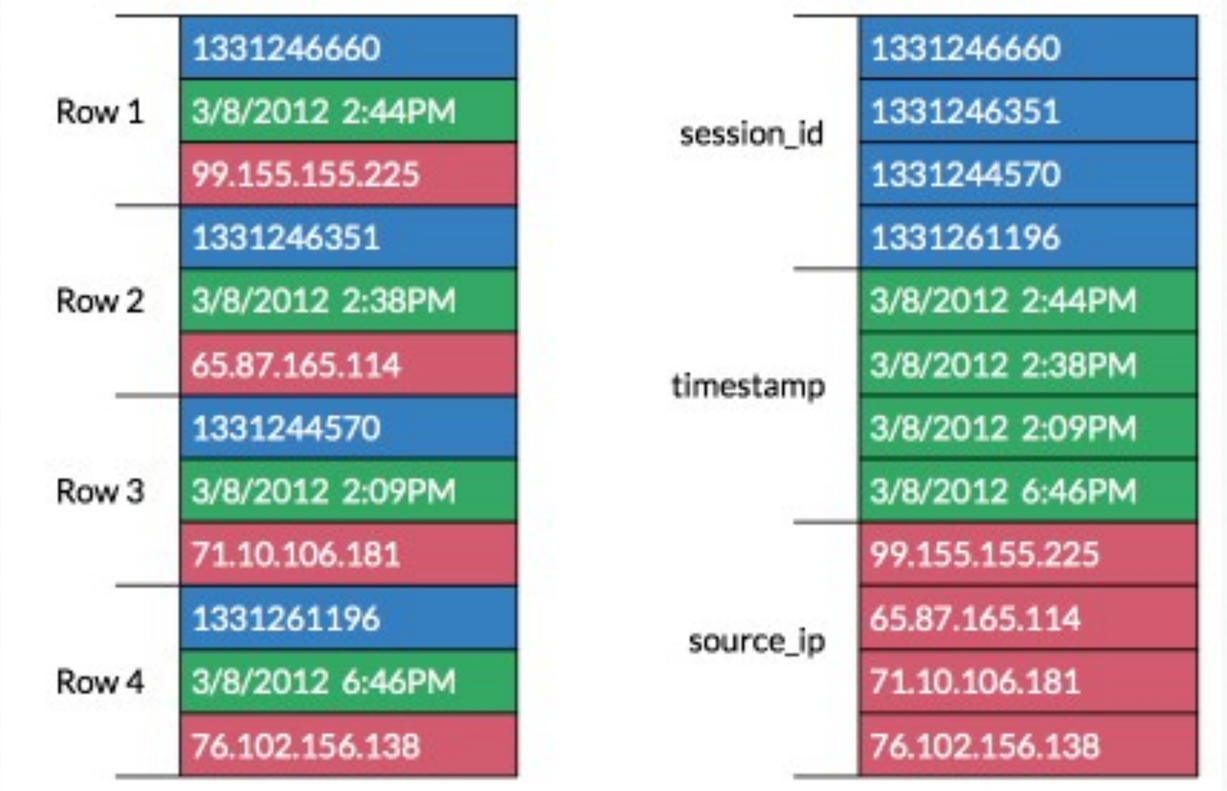
APACHE PARQUET

Unlike row-based formats such as CSV, Apache Parquet is **column-oriented** – meaning the values of each table column are **stored next to each other**, rather than those of each record

metadata including schema and structure is **embedded within each file**, making it a self-describing file format.

compression is performed column by column and it is built to support flexible compression options and extendable encoding schemas per data type

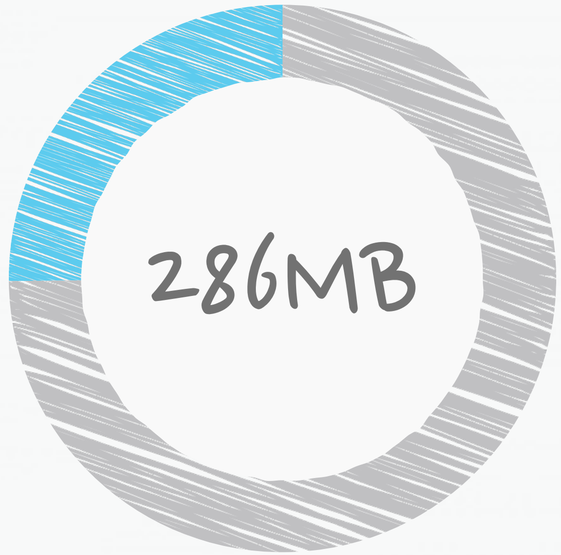
APACHE PARQUET



EACH TABLE COLUMN ARE STORED NEXT TO EACH OTHER

COMPRESSION COMPARISON

STORED IN CSV TYPE



STORED IN PARQUET TYPE

"63.4% LOWER THAN CSV!"

ANALYSIS & INSIGHTS

DATA ANALYSIS

EXTRACT POPULAR FACTORS BY CATEGORY

- EXTRACT COMMON INNER CHARACTERISTICS OF ITEMS IN THE SUBTREE
- FIGURE OUT POPULAR FACTORS BY ITEM

Step 1

Step 2

ASSOCIATION RULES BETWEEN SEE-ALSO ITEMS

- ANALYSIS OF ASSOCIATED TREES USING THE NETWORX MODEL
- EXTRACT MEANINGFUL SUBTREE

Association Rules of Also-viewed Item

Association Rule

"How products are related"

PageRank

"Which product is important"

ASSOCIATION RULES OF ALSO-VIEWED ITEM

ASSOCIATION RULES OF ALSO-VIEWED ITEM

FP GROWTH ALGORITHM

```
# 데이터 sort
for id, list in sorted_list.items():
    try:
        list = sorted(list, key=lambda idx: frequency_table.get(idx, 0), reverse=True)
    except:
        continue
    try: sorted_list[id] = list[:20]
    except: sorted_list[id] = list[:10]
    for listed_id in sorted_list[id]:
        if frequency_table.get(listed_id, 0) == 0:
            sorted_list[id].remove(listed_id)

category_frequency_table[c] = frequency_table
category_sorted_list[c] = sorted_list
category_sales_list[c] = sales_list
```

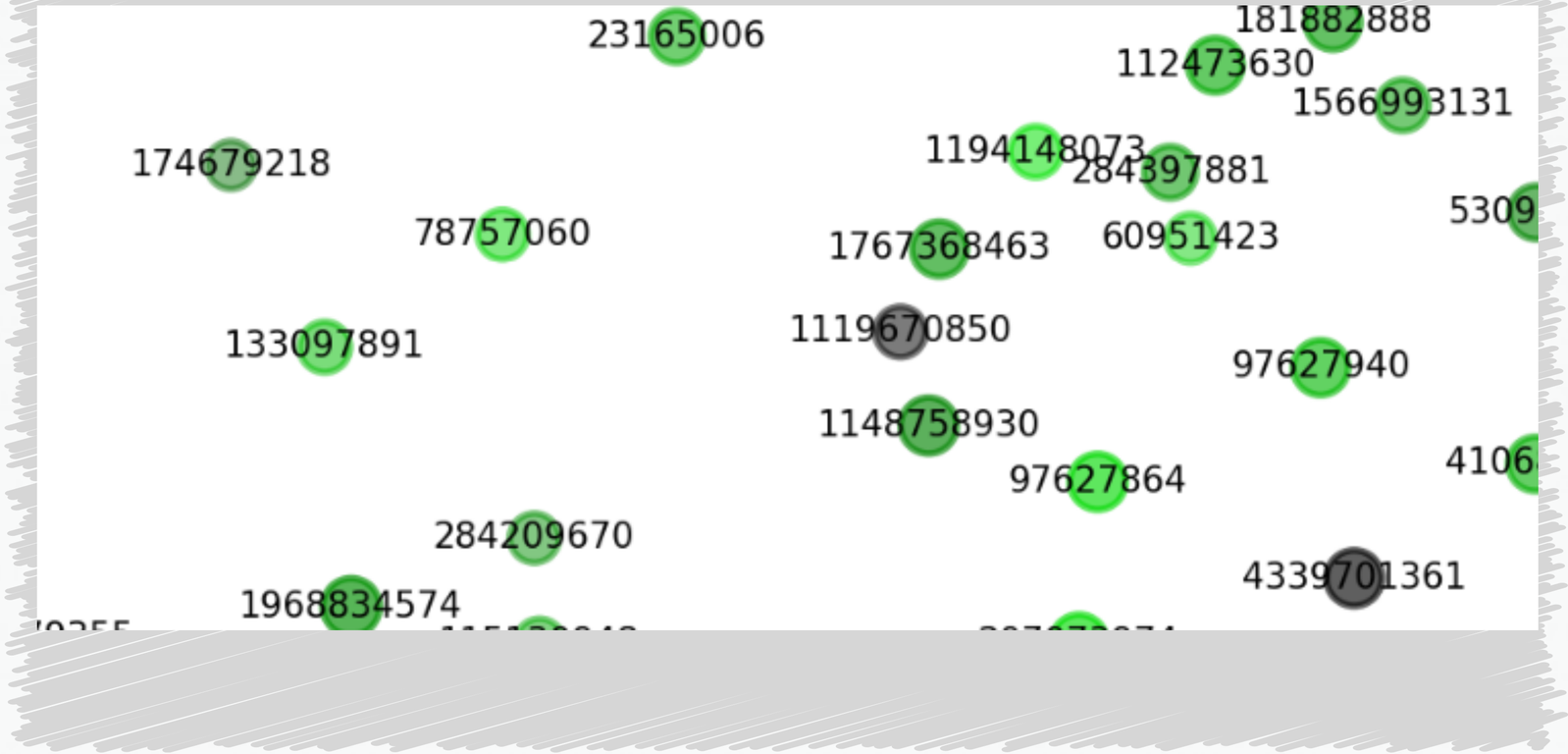
ASSOCIATION RULES OF ALSO-VIEWED ITEM

FP GROWTH TREE

```
#값 normalize
frequency_norm = [frequency_table.get(n, 0.0)/frequency_table.max() for n in G.nodes()]
sold_norm = [sales_list.get(n, 0.0)/max(sales_list.values()) for n in G.nodes()]
node_colors = [(0, (math.log10(sold_norm[i]*9.0 + 1)), 0, (math.log10(frequency_norm[i]*9.0 + 1))) \
        for i in range(len(G.nodes()))]


pos = graphviz_layout(G, prog="neato")

plt.figure(i, figsize=(100, 100))


nx.draw_networkx(G, pos=pos, node_color=node_colors,\
                 edge_color=(0, 0, 0, 0), with_labels=True, font_size=5,\
                 node_size=[math.log10(frequency_table.get(n, 0.0)/frequency_table.max() * 9.0 + 1) * 100 \
                         for n in G.nodes()])
```

ASSOCIATION RULES OF ALSO-VIEWED ITEM

VISUALIZATION



EXTRACT COMMON FACTORS OF SUBTREE

EXTRACT INTERNAL CHARACTERISTIC VALUES BY ITEMS

```
# 해당하는 카테고리의 데이터프레임 가져오기
file_name = 'user/hadoop/data_analysis/result2/' + category + '.csv'
df = spark.read.csv('hdfs://192.168.0.9:9000/' + file_name, header=True)
df = ad_schema(df)

# id_list에 있는 상품들로 dataframe만들기
df_target = spark.createDataFrame(id_list, StringType())
df_target = df_target.selectExpr("value as product_id")
df_target = df.join(df_target, 'product_id', how='inner')


# 공통 요소
commonFactors = []

# 최다공통 요소 출력
for col in df_target.columns[13:len(df_target.columns)-1]:
#     print('특성 :', col)
    result = df.groupby(col).agg(f.count(col).alias("개수"),
                                 f.sum('sales').alias('판매량')).sort(desc('개수'))
#     result.show()
    result = result.filter((f.col(col) != 'NULL'))

    if result.count() != 0:
        key = str(result.select(col).take(1)[0])
        key = key[key.find("=")+2:key.find(")")-1]
        count = str(result.select("개수").take(1)[0])
        count = count[count.find("(")+1:key.find(")")]
        commonFactors.append(col+ ' : ' +key+'('+count+')')

#     result.show()
print("공통 요소 :", commonFactors)
print('-----------------------------------------------------------------------------------------------------------')
```

EXTRACT POPULAR FACTORS BY CATEGORY

```
schema = StructType(devColumns)

# 스파크 dataframe 생성
df = spark.read.csv('hdfs://192.168.0.9:9000/' + file_name, header=True)

# 카테고리 출력
item = file[:file.find('_')]
print('카테고리 : ',item)

# 판매량 산정
temp = df_pd['reviews_for_last1year'].tolist()
temp = temp[::-1]
temp = [x for x in temp if math.isnan(x) == False]
std = int(np.mean(temp[:10]))

df = df.withColumn("reviews_for_last1year",df["reviews_for_last1year"].cast(Int
df = df.withColumn('sales', (1080 - df.rank)*std + (df.reviews_for_last1year)*3

# 상품별 인기 요소
keywords = []

# 특성별 판매량 순위
for col in df_pd.columns[13:]:
    #print('특성 :', col)
    result = df.groupby(col).agg(f.count(col).alias("개수"),
                                 f.sum('sales').alias('판매량')).sort(desc('판매량'
    result = result.withColumn('상품당 판매량',(result.판매량/result.개수)).sort(desc(
    result = result.filter((f.col('개수') > 9) & (f.col(col) != 'NULL'))

    if result.count() != 0:
        key = str(result.select(col).take(1)[0])
        key = key[key.find("=")+2:key.find(")")-1]
        keywords.append(col+ ' : ' +key)

    #result.show()
print("인기 요소 :", keywords)
print('-------------------------------------------------------------------------
```

VISUALIZATION & INSIGHTS

Zeppelin Notebook Job

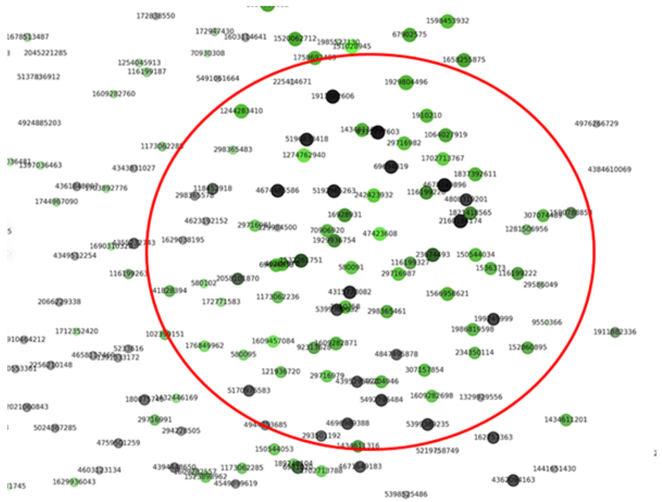
result Head

```
def get_dir(path):
    hdfs = PyWebHdfsClient(host='localhost',port='50070', user_name='master')  # your Namenode IP & username here
    my_dir = path

    result = []

    for i in range(len(hdfs.list_dir(my_dir)['FileStatuses']['FileStatus'])):
        result.append(hdfs.list_dir(my_dir)['FileStatuses']['FileStatus'][i]['pathSuffix'])

    return result

# spark datafrmae schema 설정
def ad_schema(df):
    intCols = ['rank','category','price', 'discount_percentage', 'rating_total_count',
               'reviews_for_lastlyear', 'sales']
    doubleCols = ['rating']
    boolCols = ['rocket_delivery', 'is_out_of_stock']

    for c in df.columns:
        type_col = StringType()
        if c in intCols:
            type_col = IntegerType()
        elif c in doubleCols:
            type_col = DoubleType()
        elif c in boolCols:
            type_col = BooleanType()

        df = df.withColumn(c,df[c].cast(type_col))
    df = df.fillna(0)
    return df

# 여기서 카테고리 별 id_list 작성
category = 'partition'
id_list = [1929804496,1910210,1244283410,1064027919,1702713767,1837392611,16928931,242423932,29
k = 0
for i in id_list:
    id_list[k] = str(i)
    k = k + 1

# 해당하는 카테고리의 데이터프레임 가져오기
file_name = 'user/hadoop/data_analysis/result2/' + category + '.csv'
df = spark.read.csv('hdfs://192.168.0.9:9000/' + file_name, header=True)
df = ad_schema(df)

# id_list에 있는 상품들로 dataframe만들기
df_target = spark.createDataFrame(id_list, StringType())
df_target = df_target.selectExpr("value as product_id")
df_target = df.join(df_target, 'product_id', how='inner')

# 공통 요소
commonFactors = []

# 최다공통 요소 출현
for col in df_target.columns[13:len(df_target.columns)-1]:
#     print('컬럼 :', col)
    result = df.groupby(col).agg(f.count(col).alias("개수"),
                                 f.sum('sales').alias('판매량')).sort(desc('개수'))
#     result.show()
    result = result.filter((f.col(col) != 'NULL'))

    if result.count() != 0:
        key = str(result.select(col).take(1)[0])
        key = key[key.find("=")+2:key.find(")")-1]
        count = str(result.select("개수").take(1)[0])
        count = count[count.find("(")+1:key.find(")")]
        commonFactors.append(col+ ' : ' +key+'('+count+')')

#     result.show()
print("공통 요소 :", commonFactors)
print('-----------------------------------------------------------------------------------------------')
```

```
공통 요소 : ['쓰상 : 화이트(개수=21)', '재질 : 원목(개수=34)', '인테리어 : 모던/심플(개수=19)']
```

사진 - 1231.png

모든 사진 보기 + 다음에 추가...



LET'S MOVE ON TO ZEPPELIN >>