LinkShops

데이터 추출 파이프라인 구축

데이터 흐름 정의서

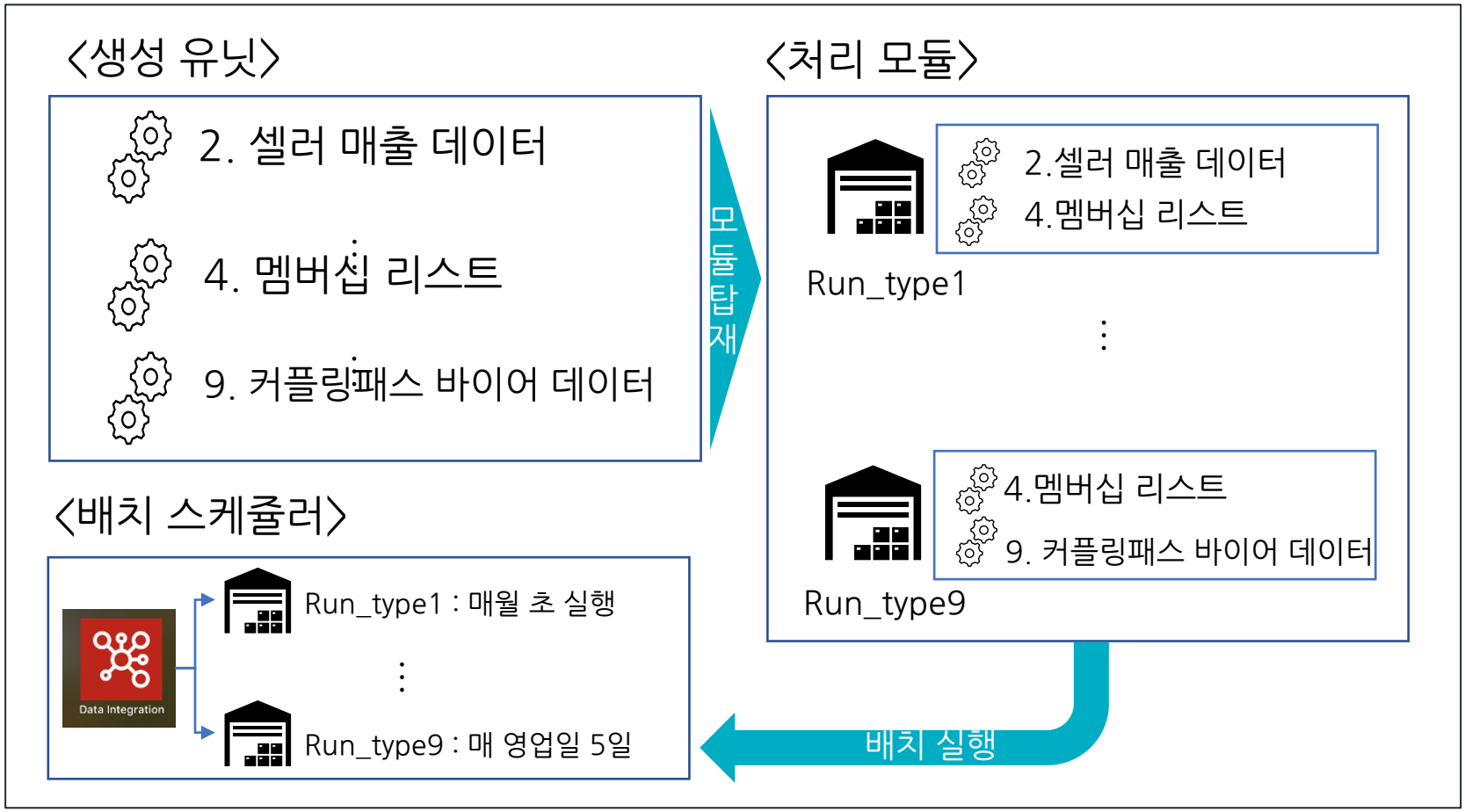
개정이력

번호	버전	개정내용	개정자	일자
1	1.0	최초 생산	Vida	2022-12-31

0. 기본 아키텍쳐

0. 기본 아키텍쳐

1. 파이프라인은 작은 단위부터 **생성유닛**, **처리모듈**, **배치 스케쥴러** 세 단계의 층위로 구성
2. 모듈화 설계를 통해 현업 부서의 잦고 빠른 추출 패턴 변경 요청에 대응 가능



1. BuyerDC

1. BuyerDC

- 각 바이어별 한달간 주문건별 매출액 / 배송비 /검수검품비 / 사입수수료 할인 현황 전달

1. dm_orderseries 데이터마트

1	**orderId**
2	buyerId
3	createdAt
4	initTotalKRW
5	initHandlingFee
6	initQualityInspectionFee
7	initKRW
8	shippingCost
9	dcShippingCost
10	dcHandlingFee
11	dcQualityInspectionFee
12	productid
13	initSubtotal
14	initQuantity
15	brandId
16	Building
17	countryCode
18	Status_order

Marketing_dm_kr(1차)

1		**orderId**	**Integer**
2		사업자ID	Integer
3		구매날짜	Integer
4		총 결제금액	Integer
5		사입수수료	Integer
6		검수검품비	Integer
7		상품 금액	Integer
8		배송비	Integer
9		사입수수료할인	Integer
10		검수검품비할인	Integer
11		상품번호	String
12		상품 금액	Integer
13		수량	Integer
14		브랜드ID	Integer
15		상가	String
16		국가코드	string

- 데이터 전처리 Process

1. 데이터 필터링
   -. CountryCode 'KR'인 데이터만 추출
   -. Status_order가 100, 200, 201, 202, 300, 400인 주문건만 추출

2. 대량 주문건 처리
   -. productId가 '_(언더대시)'로 구분되어 있는 경우 => 대량 주문건에 해당하는 주문내역
   -. 해당 주문건들은 현업 요청에 따라 제외
   -. 구체적으로, **'_'를 기준**으로 **productId**를 **분할**했을 때 **두 개 이상**인 주문 내역은 **제외**

3. 열 삭제
   -. dcShippingCost, status_order 열 삭제

4. 한글명 변경
   -. 영어로 되어있는 컬럼명을 한국어로 변경

1. BuyerDC

- 각 바이어별 한달간 주문건별 매출액 / 배송비 /검수검품비 / 사입수수료 할인 현황 전달

Marketing_dm_kr(1차)

1		**orderId**	**Integer**
2		사업자ID	Integer
3		구매날짜	Integer
4		총 결제금액	Integer
5		사입수수료	Integer
6		검수검품비	Integer
7		상품 금액	Integer
8		배송비	Integer
9		사입수수료할인	Integer
10		검수검품비할인	Integer
11		상품번호	String
12		상품 금액	Integer
13		수량	Integer
14		브랜드ID	Integer
15		상가	String
16		국가코드	string

Marketing_dm_kr(2차)

1		**orderId**	**Integer**
2		사업자ID	Integer
3		구매날짜	Integer
4		총 결제금액**(중복)**	Integer
5		사입수수료**(중복)**	Integer
6		검수검품비	Integer
7		상품 금액**(중복)**	Integer
8		배송비	Integer
9		사입수수료할인**(중복)**	Integer
10		검수검품비할인	Integer
11		상품번호	String
12		상품 금액	Integer
13		수량	Integer
14		브랜드ID	Integer
15		상가	String
16		국가코드	string

- 데이터 전처리 Process

1. 상품 주문번호를 주문번호 단위로 GroupBy

   -. 현재 데이터는 상품 주문번호 단위로 데이터를 추출한 상태
   -. 현업에서 원하는 것은 orderId 단위이므로, 주문번호 단위로 GroupBy하여 총합
   -. **initTotalKRW, initSubtotal, initHandlingFee, dcHandlingFee**를 **orderId**로 **GroupBy**하여 **총합**

2. 상품 번호 단위를 상품 주문번호 단위로

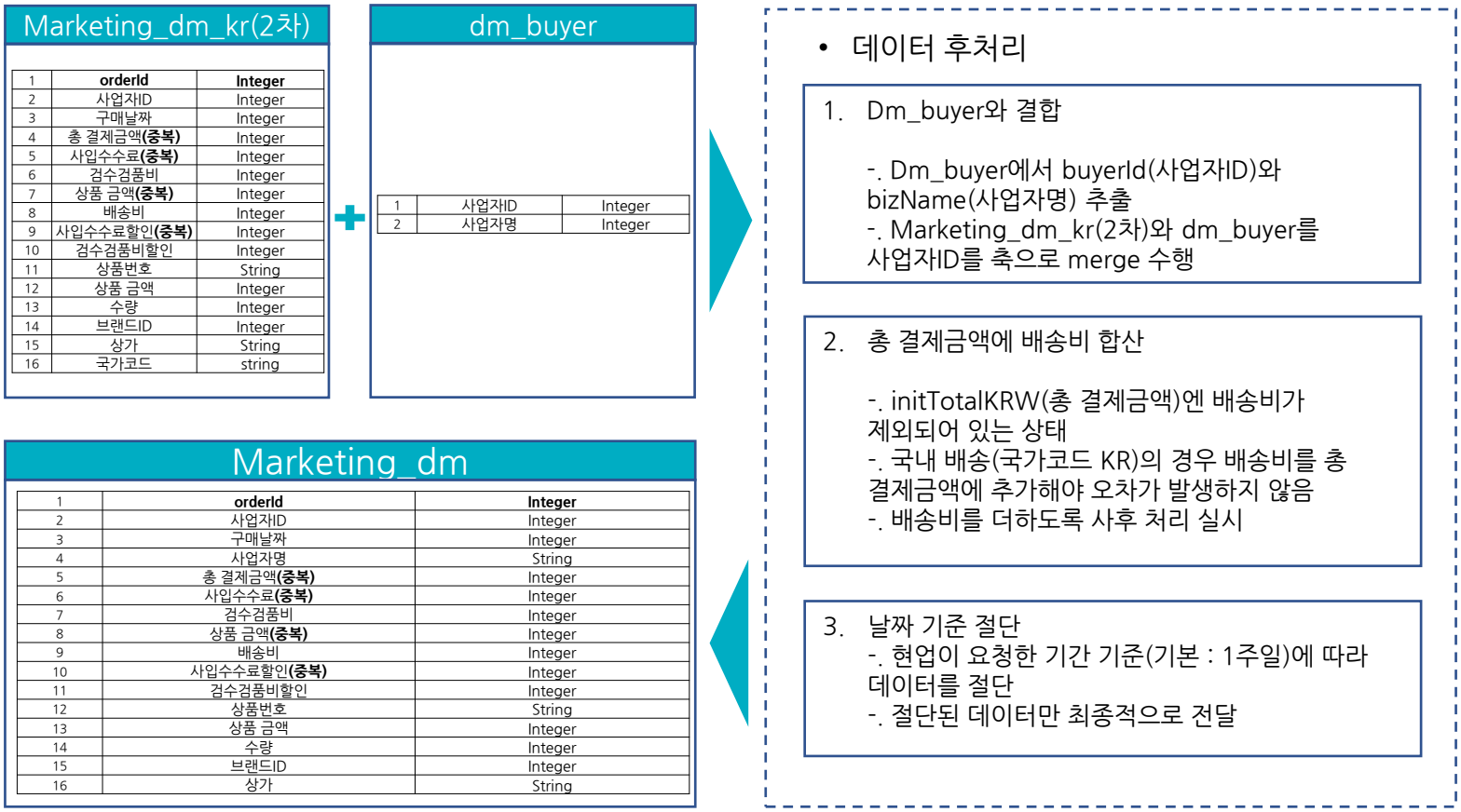
   -. 위에서 **orderId** 단위로 **GroupBy Sum을** 진행하였으나, 현업에서 원하는 것은 **orderId** 단위와 **상품 주문번호** 단위 사이의 절충점
   -. 따라서, 동일한 orderId를 공유하는 주문내역은 initTotalKRW, initSubtotal, initHandlingFee, dcHandlingFee가 같은 값이 반복되도록 처리
   -. 구체적으로, **(1)원본 dm_orderseries**에 **(2)Groupby Sum한 데이터**를 **orderId**에 대하여 **Merge**를 **수행**하여 **Outer Join** 수행

3. 날짜 변경
   -. 시분초 까지 있는 형태를 연-월-일 형태로 변경

#LinkShops

1. BuyerDC

- 각 바이어별 한달간 주문건별 매출액 / 배송비 /검수검품비 / 사입수수료 할인 현황 전달



#LinkShops

2. Seller

2. Seller

- 셀러 데이터 기반으로 주문 정보를 확인
- 주문 상태 기준 / 대량 주문건 처리로 데이터 사전 준비

1. dm_orderseries 데이터마트

1	**orderId**
2	buyerId
3	createdAt
4	initTotalKRW
5	initHandlingFee
6	initQualityInspectionFee
7	initKRW
8	shippingCost
9	productid
10	initSubtotal
11	initQuantity
12	brandId
13	countryCode
14	Status_order

Marketing_dm_seller(1차)

1	orderId	**Integer**
2	buyerId	Integer
3	createdAt	Integer
4	initTotalKRW	Integer
5	initHandlingFee	Integer
6	initQualityInspectionFee	Integer
7	initKRW	Integer
8	shippingCost	Integer
9	productid	Integer
10	initSubtotal	Integer
11	initQuantity	String
12	brandId	Integer
13	countryCode	Integer
14	Status_order	Integer

- 데이터 전처리 Process

1. 데이터 필터링
   -. Status_order가 100, 200, 201, 202, 300, 400인 주문건만 추출

2. 대량 주문건 처리
   -. productId가 '_(언더대시)'로 구분되어 있는 경우 => 대량 주문건에 해당하는 주문내역
   -. 해당 주문건들은 현업 요청에 따라 제외
   -. 구체적으로, **'_'를 기준**으로 **productId**를 **분할**했을 때 **두 개 이상**인 주문 내역만 따로 선별

#LinkShops

2. Seller

- 셀러 데이터 기반으로 주문 정보를 확인
- 대량 주문건만 따로 추출하여 사후 보정 수행

Marketing_dm_seller(1차)

1	orderId	**Integer**
2	buyerId	Integer
3	createdAt	Integer
4	initTotalKRW	Integer
5	initHandlingFee	Integer
6	initQualityInspectionFee	Integer
7	initKRW	Integer
8	shippingCost	Integer
9	productid	Integer
10	initSubtotal	Integer
11	initQuantity	String
12	brandId	Integer
13	countryCode	Integer
14	Status_order	Integer

Dm_seller

1	brandId	**Integer**
2	Membership	Integer
3	membershipGrade	Integer

- 데이터 전처리 Process

1. 대량 주문건 보정
   -. 대량 주문건엔 다음의 사항이 누락되어 있기 때문에 보정 수행
   -. **initSubtotal(총금액)** : initTotalKRW 값만 존재하기 때문에 이 값으로 initSubtotal을 대치 (대량주문건은 order_products에 내역이 존재하지 않기 때문에 여기서 파생되는 initSubtotal이 없음)
   -. initTotalKRW(총 결제금액) : 대치한 initSubtotal에 initTax, initHandlingFee를 더해 initTotalKRW를 다시 최신화

2. 셀러 정보 Join
   -. Dm_seller에서 membership, membershipGrade를 가져와서 marketing_dm_selle와 Join 수행

Marketing_dm_seller(2차)

1	**주문번호**	**Integer**
2	사업자ID	Integer
3	구매일자	Integer
4	총 결제금액	Integer
5	사입수수료	Integer
6	검수검품비	Integer
7	상품금액	Integer
8	배송비	Integer
9	상품번호	Integer
10	총금액	Integer
11	수량	String
12	브랜드ID	Integer
13	국가코드	Integer
14	글로벌셀러/베이직셀러	String
15	멤버십등급	String
16	빅바이어/일반바이어/탈퇴	String

2. Seller

- 셀러 데이터 기반으로 주문 정보를 확인
- 셀러 멤버십 관련 내용 처리

Marketing_dm_seller(2차)

1	**주문번호**	**Integer**
2	사업자ID	Integer
3	구매일자	Integer
4	총 결제금액	Integer
5	사입수수료	Integer
6	검수검품비	Integer
7	상품금액	Integer
8	배송비	Integer
9	상품번호	Integer
10	총금액	Integer
11	수량	String
12	브랜드ID	Integer
13	국가코드	Integer
14	글로벌셀러/베이직셀러	String
15	멤버십등급	String
16	빅바이어/일반바이어/탈퇴	String

Seller

1	**주문번호**	**Integer**
2	사업자ID	Integer
3	구매일자	Integer
4	총 결제금액	Integer
5	사입수수료	Integer
6	검수검품비	Integer
7	상품금액	Integer
8	배송비	Integer
9	상품번호	Integer
10	총금액	Integer
11	수량	String
12	브랜드ID	Integer
13	국가코드	Integer
14	글로벌셀러/베이직셀러	String
15	멤버십등급	String
16	빅바이어/일반바이어/탈퇴	String

- 데이터 후처리

1. 셀러 멤버십 처리
   -. 글로벌셀러/베이직셀러 값이 1인 경우 글로벌셀러, 0인 경우 베이직셀러로 일괄 하드코딩
   -. 멤버십등급 값이 비어있는 경우(즉, len(x) == 0) 해당 등급을 '일반'으로 대체

2. 총 결제금액에 배송비 합산

   -. initTotalKRW(총 결제금액)엔 배송비가 제외되어 있는 상태
   -. 국내 배송(국가코드 KR)의 경우 배송비를 총 결제금액에 추가해야 오차가 발생하지 않음
   -. 배송비를 더하도록 사후 처리 실시

3. 날짜 기준 절단
   -. 현업이 요청한 기간 기준(기본 : 1주일)에 따라 데이터를 절단
   -. 절단된 데이터만 최종적으로 전달

#LinkShops

3. GlobalSKU

3. GlobalSKU

- Paid Seller만 선정 후 해당 셀러가 업로드한 상품들의 SKU정보 등록 현황을 정오표(O/X)로 제공
- 브랜드 정보 추출 후 유료 셀러 추출 수행

dm_merchandise

1	**productId**
2	variantId
3	date
4	totalKRW
5	finalTotalKRW
6	finalQuantity
7	quantity
8	SKUData
9	VariantData
10	brand
11	createdAt_prod
12	updatedAt_prod
13	createdAt_variant
14	updatedAt_variant
15	isBulkOrder

dm_seller

1	**brandId**
2	Membership
3	membershipStatus
4	membershipGrade

- 데이터 전처리 Process

1. 브랜드 정보 추출
   -. 브랜드의 정보중 BrandId(브랜드아이디), brandName(브랜드명), Building(상가정보) 추출

2. Paid Seller 추출
   -. Membership이 1이면서, membershipStatus가 200인 셀러만 추출(Membership = 1은 글로벌 셀러, membershipStatus 200은 승인 완료를 의미)
   -. 추출된 글로벌 셀러 중 membershipGrade가 'black'인 Seller만 최종 추출

Marketing_dm_merchandise

1	**productId**	**Integer**
2	variantId	
3	date	Integer
4	totalKRW	Integer
5	finalTotalKRW	Integer
6	finalQuantity	Integer
7	quantity	Integer
8	SKUData	Json
9	VariantData	Json
10	brand	Integer
11	createdAt_prod	Integer
12	updatedAt_prod	String
13	createdAt_variant	Integer
14	updatedAt_variant	Integer
15	isBulkOrder	Integer
16	brandId	Integer
17	brandName	String
18	building	string

3. GlobalSKU

- Paid Seller만 선정 후 해당 셀러가 업로드한 상품들의 SKU정보 등록 현황을 정오표(O/X)로 제공
- 소재 정보, 사이즈 정보를 람다 함수를 통해 Json에서 추출

Marketing_dm_merchandise

1	**productId**	**Integer**
2	variantId	
3	date	Integer
4	totalKRW	Integer
5	finalTotalKRW	Integer
6	finalQuantity	Integer
7	quantity	Integer
8	SKUData	Json
9	VariantData	Json
10	brand	Integer
11	createdAt_prod	Integer
12	updatedAt_prod	String
13	createdAt_variant	Integer
14	updatedAt_variant	Integer
15	isBulkOrder	Integer
16	brandId	Integer
17	brandName	String
18	building	string

1. Result_material_ko

1	**productId**	**Integer**
2	**VariantId**	**Integer**
3	Materical_ko	Json

2. Result_size_ko

1	**productId**	**Integer**
2	**VariantId**	**Integer**
3	size_ko	Json

- 데이터 전처리 Process

1. 소재 정보 추출(Json 까기)
   -. 혼방률(ex. 면 80%, 레이온 20% 등) 추출 목적
   -. SKUdata에 존재하는 Json 내에서 data -> specifications 항목의 값을 추출
   -. **(람다함수)** Specifications 내에 존재하는 한글명(name -> 'ko')와 해당 한글명 소재의 비율('rate')를 추출
   -. Productid, variantId, 위에서 추출한 혼방비율 세 가지 정보를 통합한 DataFrame을 반환

2. 사이즈 정보 추출(Json 까기)
   -. 사이즈 정보(ex. 총장, 어깨너비, 가슴길이, 팔길이)를 추출 목적
   -. SKUdata에 존재하는 Json 내에서 data -> 'measurement'의 값을 추출
   -. **(람다함수)** measurement 내에 존재하는 한글명(name -> 'ko')에서 해당 한글명 항목(ex. 총장 / 어깨너비 등등) 과 그에 수반되는 사이즈('size)를 추출
   -. prodId, varId, 사이즈 정보 통합 DF 반환

3. GlobalSKU

- Paid Seller만 선정 후 해당 셀러가 업로드한 상품들의 SKU정보 등록 현황을 정오표(O/X)로 제공
- 세탁 정보, 피팅 정보를 람다함수를 통해 Json에서 추출

Marketing_dm_merchandise

1	**productId**	**Integer**
2	variantId	
3	date	Integer
4	totalKRW	Integer
5	finalTotalKRW	Integer
6	finalQuantity	Integer
7	quantity	Integer
8	SKUData	Json
9	VariantData	Json
10	brand	Integer
11	createdAt_prod	Integer
12	updatedAt_prod	String
13	createdAt_variant	Integer
14	updatedAt_variant	Integer
15	isBulkOrder	Integer
16	brandId	Integer
17	brandName	String
18	building	string

3. Result_washing_ko

1	**productId**	**Integer**
2	**VariantId**	**Integer**
3	washinfo_ko	Json

4. Result_fit_ko

1	**productId**	**Integer**
2	**VariantId**	**Integer**
3	fitinfo_ko	Json

- 데이터 전처리 Process

3. 세탁 정보 추출(Json 까기)
   -. 세탁 정보(ex. 비틀기 금지, 드라이 클리닝 등)을 추출 목적
   -. SKUdata에 존재하는 Json 내에서 data -> 'Specifications' 추출
   -. **(람다함수)** Specifications 내에 존재하는 세탁방법론 중 등록자(셀러)가 체크한 것(=True)인 세탁방법론만 추출
   -. Productid, variantId, 위에서 추출한 세탕방법 세 가지 정보를 통합한 DataFrame을 반환

4. 피팅정보 추출
   -. 피팅 정보(ex. 비침, 두께감 등)을 추출 목적
   -. SKUdata에 존재하는 Json 내에서 data -> 'Specifications' 추출
   -. **(람다함수)** 핏 정보는 각각의 핏정보 **항목**(비침, 두께감) 과 각각의 **항목의 정도**(ex. 비침 - (없음/약간/비침), 두께감 - (두꺼움/보통/얇음) 로 구성된 행렬임.
   -. 각 **항목**별로 Check(=True)인 **정도**만 가져와 추출 (ex. 비침 - 약간, 두께감 - 보통 등)

3. GlobalSKU

- Paid Seller만 선정 후 해당 셀러가 업로드한 상품들의 SKU정보 등록 현황을 정오표(O/X)로 제공
- 정오표(O / X) 변환 수행

1. Result_material_ko

1	**productId**	**Integer**
2	**VariantId**	**Integer**
3	Materical_ko	Json

2. Result_size_ko

1	**productId**	**Integer**
2	**VariantId**	**Integer**
3	sizeinfo_ko	Json

3. Result_washing_ko

1	**productId**	**Integer**
2	**VariantId**	**Integer**
3	washinfo_ko	Json

4. Result_fit_ko

1	**productId**	**Integer**
2	**VariantId**	**Integer**
3	fitinfo_ko	Json

Marketing_dm_merchandise(2차)

1	**productId**	**Integer**
2	variantId	
3	date	Integer
4	totalKRW	Integer
…	…	
15	brandId	Integer
16	brandName	String
17	Building	string
18	Material_ko	String
19	Washinfo_ko	String
20	Sizeinfo_ko	String
21	Fitinfo_ko	string
22	SKUData	Json

- 데이터 전처리 Process

1. 인덱스 정보 통일 및 Concat
   -. 1 ~ 4 테이블의 인덱스를 productid + variantId 결합 (prodId)_(varId)로 모두 변경
   -. 동일한 인덱스(즉 동일한 prodId와 varId)를 가진 행끼리 Column을 붙여 가로로 긴 테이블로 결합

2. 1,3,4번 테이블 정오표(O / X) 변환
   -. 1 ~ 3 테이블을 결합하는 과정에서 값이 없는 경우(즉 셀러가 해당 항목을 입력하지 않은 경우)는 NaN값으로 처리
   -. NaN값인 경우 입력하지 않은 것으로 보고 'X'를, NaN값이 아닌 경우엔 셀러가 입력한 것으로 보고 'O'로 대체

3. 2번 테이블(사이즈 정보) 정오표(O / X) 변환
   -. 2번 테이블(사이즈 정보)의 경우 어떤 값은 셀러가 성실하게 입력했으나 어떤 값은 입력하지 않은(NaN) 경우 존재
   -. 모든 값이 NaN이 아닌 경우에만 사이즈 정보를 입력한 것으로 보고 'O'로, 그 외의 경우 'X' 처리

#LinkShops

3. GlobalSKU

- Paid Seller만 선정 후 해당 셀러가 업로드한 상품들의 SKU정보 등록 현황을 정오표(O/X)로 제공
- 기타 데이터를 Json에서 추출하여 최종 정리

Marketing_dm_merchandise(2차)

1	**productId**	**Integer**
2	variantId	
3	date	Integer
4	totalKRW	Integer
…	…	
15	brandId	Integer
16	brandName	String
17	Building	string
18	Material_ko	String
19	Washinfo_ko	String
20	Sizeinfo_ko	String
21	Fitinfo_ko	string
22	SKUData	Json

Marketing_dm

1	**productId**	**Integer**
2	상품명	String
3	등록날짜	Datetime
4	업데이트날짜	Datetime
5	브랜드 아이디	Integer
6	브랜드명	String
7	상가	String
8	카테고리	String
9	사이즈	String
10	세탁정보	String
11	피팅정보	string

- 데이터 전처리 Process

1. 각종 데이터 추출(Json 까기)
   -. SKUData에서 updatedAt을 추출하여 updatedAt 열 추가(업데이트일)
   - SKUData에서 createdAt을 추출하여 createdAt열 추가(생성일)
   -. SKUData에서 name -> 'ko'를 추출하여 names열 추가(상품명)
   -. SKUdata에서 data->'categoryPath'->'ko'를 추출하여 categoryPath열 추가(카테고리)

2. 한글명 변환
   -. 일부 컬럼만 추출하여 한글명칭으로 변경

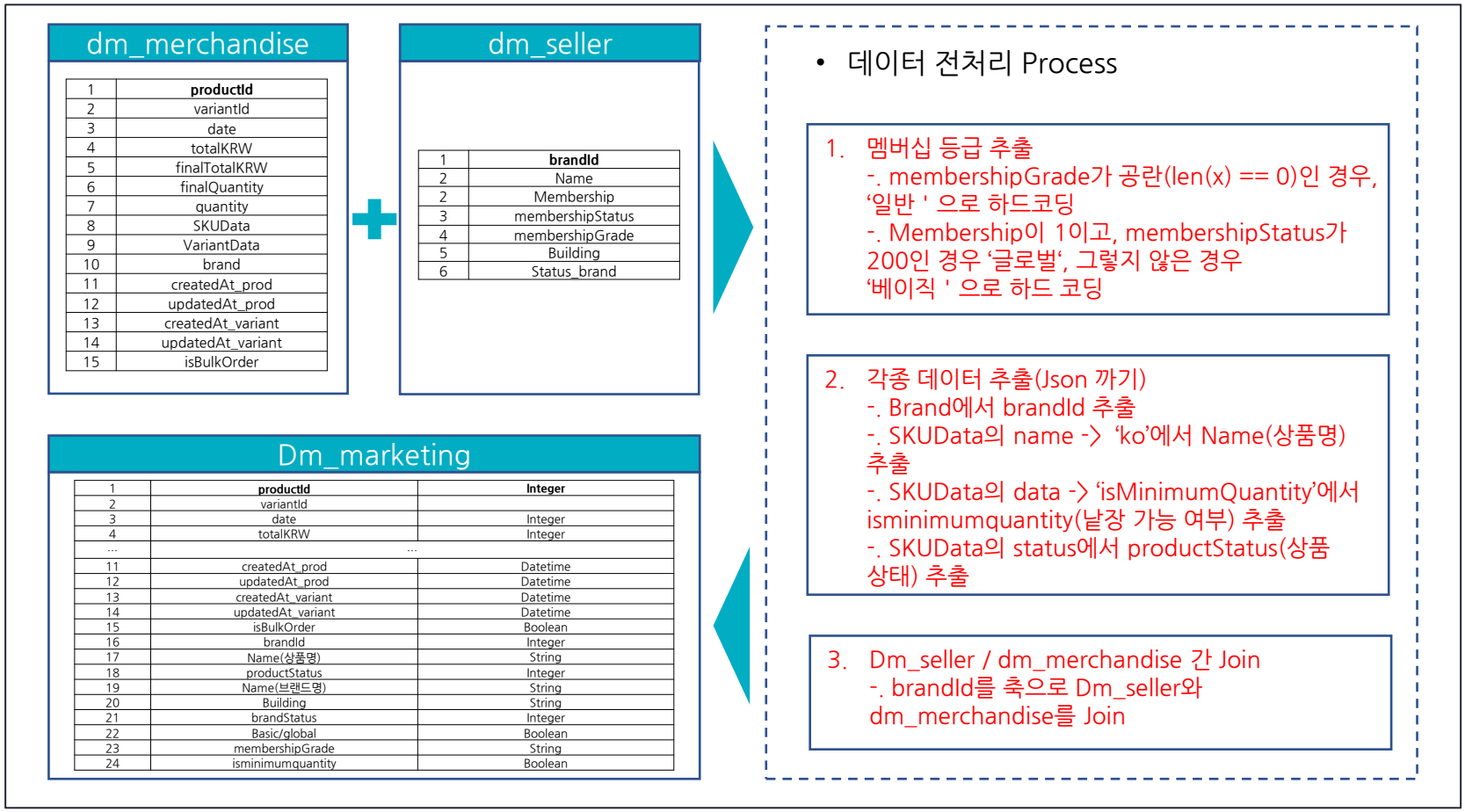
3. TimeZone 표준화
   -. UTC 기준인 TimeZone을 Asia/Seoul로 변경

#LinkShops

4. WeeklyGlobalSeller

4. WeeklyGlobalSeller

- 매 10일마다 글로벌 셀러의 판매 현황을 리포트 형식으로 전달



#LinkShops

5. MinimumAndNewer

5. MinimumAndNewer

- 신상 / 낱장 기획전용 데이터를 추출
- 셀러가 최근 등록 / 업데이트한 상품들의 낱장 가능 여부를 판별

dm_merchandise

1	**productId**
2	variantId
3	date
4	totalKRW
5	finalTotalKRW
6	finalQuantity
7	quantity
8	SKUData
9	VariantData
10	brand
11	createdAt_prod
12	updatedAt_prod
13	createdAt_variant
14	updatedAt_variant
15	isBulkOrder

dm_seller

1	**brandId**
2	Name
2	Membership
3	membershipStatus
4	membershipGrade
5	Building
6	Status_brand

Dm_marketing

1	**productId**	**Integer**
2	variantId	
3	date	Integer
4	totalKRW	Integer
…	…	
11	createdAt_prod	Datetime
12	updatedAt_prod	Datetime
13	createdAt_variant	Datetime
14	updatedAt_variant	Datetime
15	isBulkOrder	Boolean
16	brandId	Integer
17	Name(상품명)	String
18	productStatus	Integer
19	Name(브랜드명)	String
20	Building	String
21	brandStatus	Integer
22	Basic/global	Boolean
23	membershipGrade	String
24	isminimumquantity	Boolean

- 데이터 전처리 Process

1. 멤버십 등급 추출
   -. membershipGrade가 공란(len(x) == 0)인 경우, ‘일반 ' 으로 하드코딩
   -. Membership이 1이고, membershipStatus가 200인 경우 ‘글로벌‘, 그렇지 않은 경우 ‘베이직 ' 으로 하드 코딩

2. 각종 데이터 추출(Json 까기)
   -. Brand에서 brandId 추출
   -. SKUData의 name -> ‘ko’에서 Name(상품명) 추출
   -. SKUData의 data -> ‘isMinimumQuantity’에서 isminimumquantity(낱장 가능 여부) 추출
   -. SKUData의 status에서 productStatus(상품 상태) 추출

3. Dm_seller / dm_merchandise 간 Join
   -. brandId를 축으로 Dm_seller와 dm_merchandise를 Join

5. MinimumAndNewer

- 신상 / 낱장 기획전용 데이터를 추출
- 셀러가 최근 등록 / 업데이트한 상품들의 낱장 가능 여부를 판별

Dm_marketing

1	**productId**	**Integer**
2	variantId	
3	date	Integer
4	totalKRW	Integer
…	…	
11	createdAt_prod	Datetime
12	updatedAt_prod	Datetime
13	createdAt_variant	Datetime
14	updatedAt_variant	Datetime
15	isBulkOrder	Boolean
16	brandId	Integer
17	Name(상품명)	String
18	productStatus	Integer
19	Name(브랜드명)	String
20	Building	String
21	brandStatus	Integer
22	Basic/global	Boolean
23	membershipGrade	String
24	isminimumquantity	Boolean

Dm_marketing_minimum_newseven

1	**productid**	**Integer**
2	상품명	String
3	등록날짜	Integer
4	업데이트날짜	Integer
5	브랜드id	Integer
6	브랜드명	String
7	글로벌셀러/베이직셀러	String
8	멤버십그레이드	String
9	상가	String
10	낱장 가능	String

- 데이터 전처리 Process

1. 활성 상태 제품 추출
   -. 브랜드 상태가 0(= 폐업)이 아니면서 상품 상태가 100(=품절)이 아닌 상품만 추출

2. 낱장 가능 상품 추출
   -. isMinimumQuantity가 False(즉 낱장 가능) 상품만 추출해 dm_marketing_minimum 데이터 프레임으로 선언

3. 신상 제품 추출
   -. 업데이트 날짜가 기준 날짜(unit_date, 기본 7일) 이후인 상품을 신상 제품으로 추출해 dm_marketing_newseven으로 선언

4. 낱장 가능 상품 / 신상 제품 concat
   -. Dm_marketing_minimum과 dm_marketing_newseven을 통합

#LinkShops

6. BuyingReport

6. BuyingReport

- 각 월별 10일, 20일, 말일마다
주문건별 매출액과 배송비/검수검품비/사입수수료 할인 현황을 전달
- 국가 코드, 주문 상태, 대량 주문 여부를 기준으로 데이터 필터링 후 준비

dm_orderseries

1	**orderId**
2	buyerId
3	createdAt
4	initTotalKRW
5	initHandlingFee
6	initQualityInspectionFee
7	shippingCost
8	productid
9	initSubtotal
10	dcHandlingFee
11	dcQualityInspectionFee
12	dcShippingCost
13	countryCode
14	Status_order

Marketing_dm_kr

1	**오더아이디**	**Integer**
2	바이어아이디	
3	결제일	Integer
4	총 결제금액	Integer
5	상품금액	Datetime
6	사입비	Datetime
7	검수비	Datetime
8	배송비	Datetime
9	사입비할인	Boolean
10	검수비할인	Integer
11	배송비할인	String
12	국가코드	Integer

- 데이터 전처리 Process

1. 데이터 필터링
   -. 배송지기준 국가코드가 'KR'인 이력만 추출
   -. 주문 상태가 100, 200, 201, 202, 300, 400인 이력만 추출(삭제됨, 생성됨 제외)
   -. 대량 주문건 제외(productId가 'bp_' 꼴인 이력 제외)

2. 컬럼명 한글 변환
   -. 컬럼명을 한글로 변환
   -. initTotalKRW -> 총 결제금액
   -. initSubtotal -> 상품금액
   -. initHandlingFee -> 사입비
   -. initQualityInspectionFee -> 검수비
   -. shippingCost -> 배송비
   -. dcHandlingFee -> 사입비할인
   -. dcQualityInspectionFee -> 검수비할인
   -. dcShippingCost -> 배송비할인

6. BuyingReport

- 각 월별 10일, 20일, 말일마다
주문건별 매출액과 배송비/검수검품비/사입수수료 할인 현황을 전달
- orderProductId 기준인 데이터를 orderId 기준으로 변환 수행

Marketing_dm_kr

1	**오더아이디**	**Integer**
2	바이어아이디	
3	결제일	Integer
4	총 결제금액	Integer
5	상품금액	Datetime
6	사입비	Datetime
7	검수비	Datetime
8	배송비	Datetime
9	사입비할인	Boolean
10	검수비할인	Integer
11	배송비할인	String
12	국가코드	Integer

- 데이터 전처리 Process

1. 상품 주문번호를 주문번호 단위로 GroupBy

   -. 현재 데이터는 상품 주문번호 단위로 데이터를 추출한 상태
   -. 현업에서 원하는 것은 orderId 단위이므로, 주문번호 단위로 GroupBy하여 총합
   -. **총 결제금액, 상품금액, 사입비, 검수비, 사입비 할인, 검수비 할인**을 **orderId**로 **GroupBy**하여 **총합**

Marketing_dm_kr(중복제거)

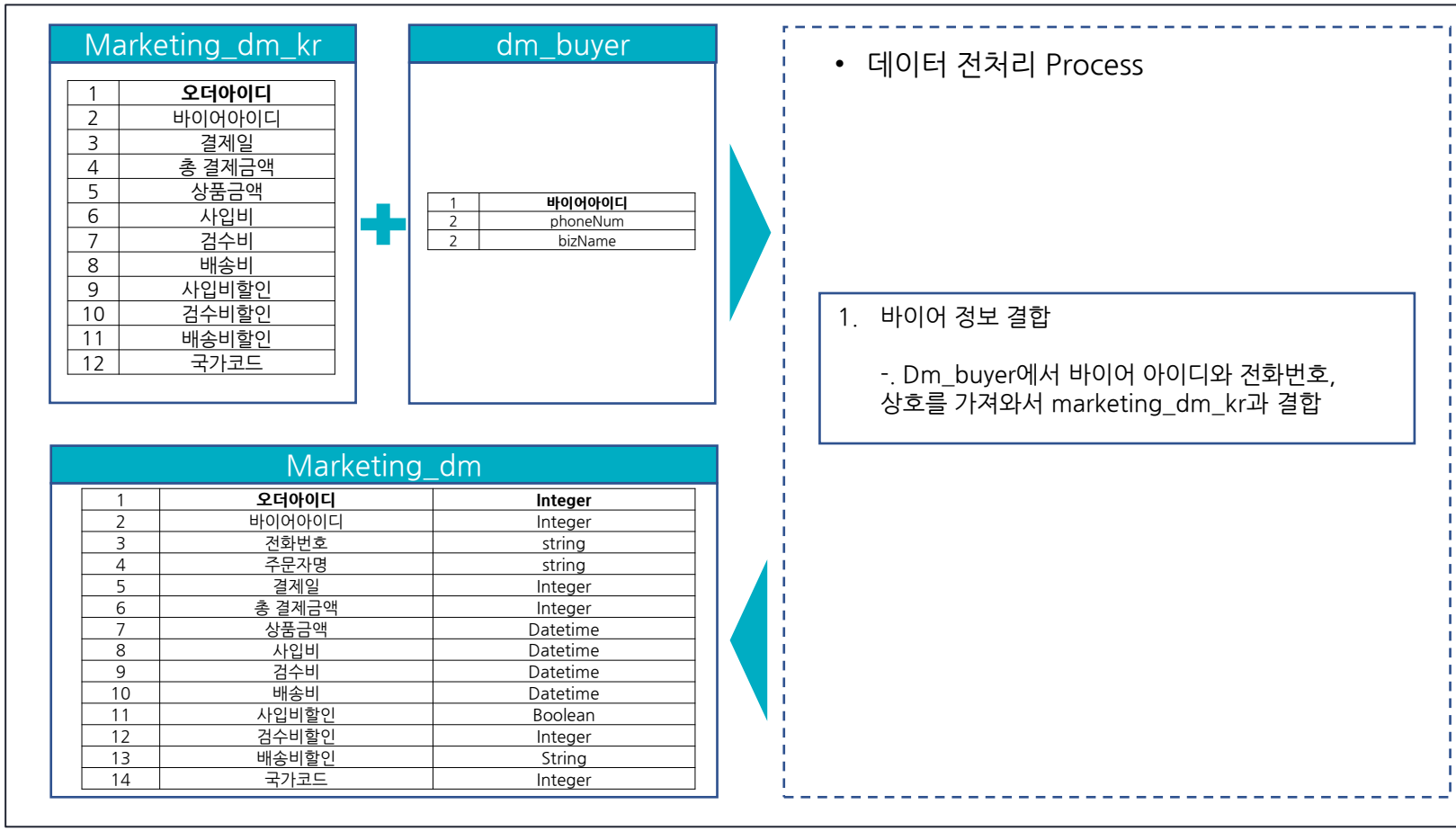
1	**오더아이디**	**Integer**
2	바이어아이디	
3	결제일	Integer
4	총 결제금액	Integer
5	상품금액	Datetime
6	사입비	Datetime
7	검수비	Datetime
8	배송비	Datetime
9	사입비할인	Boolean
10	검수비할인	Integer
11	배송비할인	String
12	국가코드	Integer

2. 상품 번호 단위를 상품 주문번호 단위로

   -. **(1)원본 marketing_dm_kr**에 **(2)Groupby Sum한 데이터**를 **orderId**에 대하여 **Merge**를 **수행**하여 **Outer Join** 수행
   -. 배송비 및 배송비 할인은 이미 orderId별로 중복되는 값이 들어가 있으므로 GroupBy할 필요 X
   -. Merge를 수행한 데이터는 오더 아이디별로 동일한 값이 반복되는 duplicated 데이터이므로, 중복값을 제거(drop_duplicates())

6. BuyingReport

- 각 월별 10일, 20일, 말일마다
주문건별 매출액과 배송비/검수검품비/사입수수료 할인 현황을 전달



#LinkShops

7. GlobalMembership

7. GlobalMembership

- 각 월별 10일, 20일, 말일마다
주문건별 매출액과 배송비/검수검품비/사입수수료 할인 현황을 전달

dm_orderseries

1	**orderId**
2	buyerId
3	createdAt
4	initTotalKRW
5	initHandlingFee
6	initQualityInspectionFee
7	shippingCost
8	productid
9	initSubtotal
10	dcHandlingFee
11	dcQualityInspectionFee
12	dcShippingCost
13	countryCode
14	Status_order

Marketing_dm_kr

1	**오더아이디**	**Integer**
2	바이어아이디	
3	결제일	Integer
4	총 결제금액	Integer
5	상품금액	Datetime
6	사입비	Datetime
7	검수비	Datetime
8	배송비	Datetime
9	사입비할인	Boolean
10	검수비할인	Integer
11	배송비할인	String
12	국가코드	Integer

- 데이터 전처리 Process

1. 데이터 필터링
   -. 배송지기준 국가코드가 'KR'인 이력만 추출
   -. 주문 상태가 100, 200, 201, 202, 300, 400인 이력만 추출(삭제됨, 생성됨 제외)
   -. 대량 주문건 제외(productId가 'bp_' 꼴인 이력 제외)

2. 컬럼명 한글 변환
   -. 컬럼명을 한글로 변환
   -. initTotalKRW -> 총 결제금액
   -. initSubtotal -> 상품금액
   -. initHandlingFee -> 사입비
   -. initQualityInspectionFee -> 검수비
   -. shippingCost -> 배송비
   -. dcHandlingFee -> 사입비할인
   -. dcQualityInspectionFee -> 검수비할인
   -. dcShippingCost -> 배송비할인

8. CouplingPassBuyer

8. CouplingPassBuyer

- 커플링패스 가입 바이어의 손익 추정을 위해 배송비 / 검수검품비 / 사입수수료 내역을 추출
- 시작 / 종료일자가 모두 다른 커플링패스를 단위기간 기준으로 모두 추출

Tb_discount_promotion

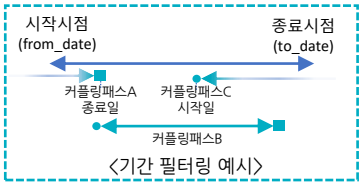
1	id
2	discount_promotion_coverage_type
3	title
4	display_title
5	description
6	start_at
7	end_at
8	max_use_count
9	cm_used_count
10	created_at
11	creator_id
12	updated_at
13	updater_id
14	use_yn

Dataframe_coupling

1	Discount_promotion_Id

- 데이터 전처리 Process

1. 데이터 필터링
   -. 제목(title)에 '커플링 ' 이 들어가는 할인 프로모션 아이디(discount_promotion_id)를 추출
   -. 이 때, 해당 프로모션의 종료일은 시작 시점(from_date)보단 뒤어야 하며, 시작일은 종료시점(to_date)보단 앞이어야 함



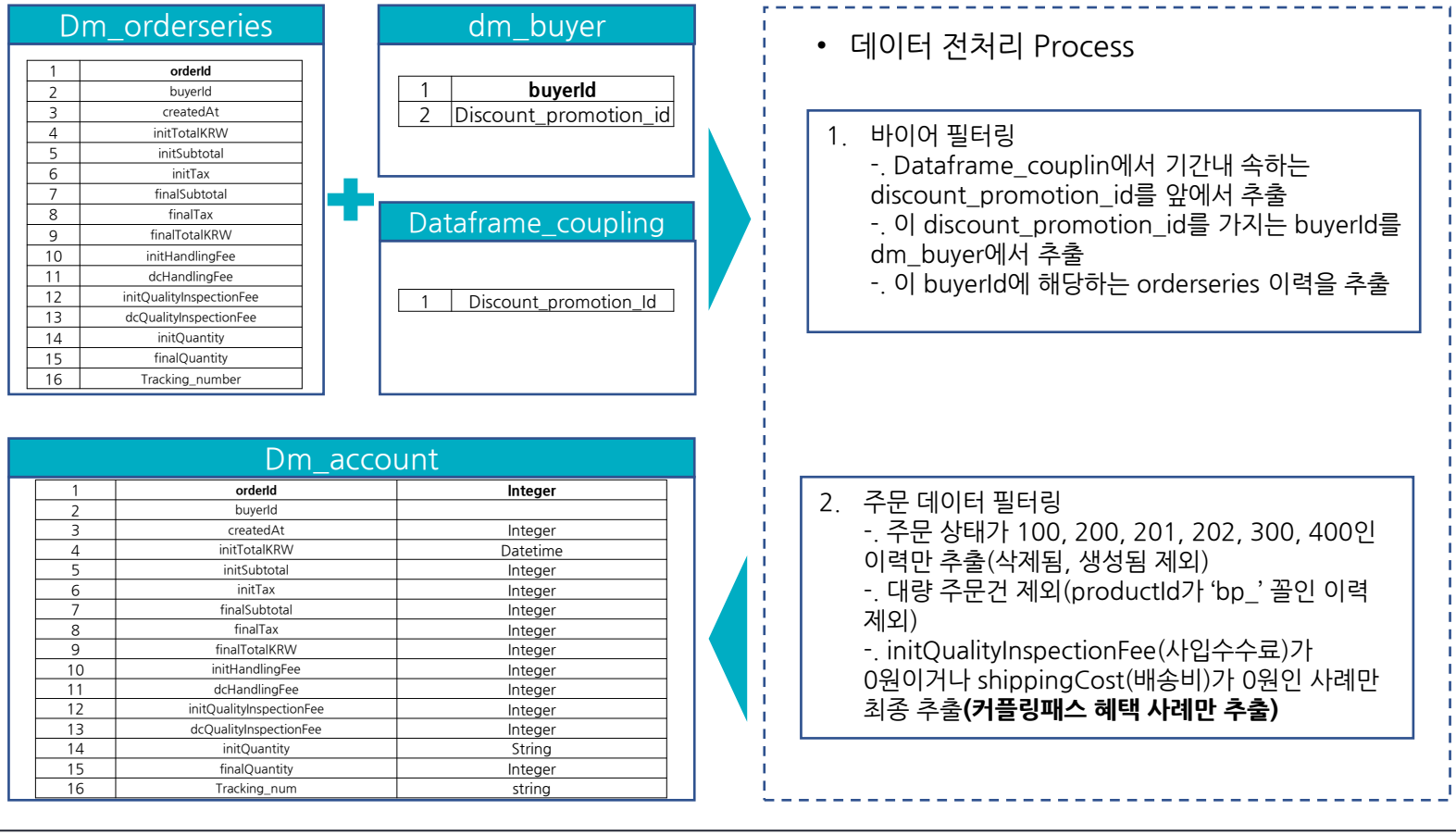
〈기간 필터링 예시〉

From_date ~ to_date
해당 기간동안
커플링패스 A,B,C가
모두 **포함**되어야함

#LinkShops

8. CouplingPassBuyer

- 커플링패스 가입 바이어의 손익 추정을 위해 배송비 / 검수검품비 / 사입수수료 내역을 추출
- 커플링패스 대상 바이어 / 주문 상태 기준 필터링 수행



#LinkShops

8. CouplingPassBuyer

- 커플링패스 가입 바이어의 손익 추정을 위해 배송비 / 검수검품비 / 사입수수료 내역을 추출
- orderProductId 기준 데이터를 orderId 단위로 변경 수행

Dm_account

1	**orderId**	**Integer**
2	buyerId	
3	createdAt	Integer
4	initTotalKRW	Datetime
5	initSubtotal	Integer
6	initTax	Integer
7	finalSubtotal	Integer
8	finalTax	Integer
9	finalTotalKRW	Integer
10	initHandlingFee	Integer
11	dcHandlingFee	Integer
12	initQualityInspectionFee	Integer
13	dcQualityInspectionFee	Integer
14	initQuantity	String
15	finalQuantity	Integer
16	Tracking_num	string

- 데이터 전처리 Process

1. 상품 주문번호를 주문번호 단위로 GroupBy

   -. 현재 데이터는 상품 주문번호 단위로 데이터를 추출한 상태
   -. 현업에서 원하는 것은 orderId 단위이므로, 주문번호 단위로 GroupBy하여 총합
   -. **최초 결제금액, 상품금액, 최초 VAT, 사입비, 검수비, 사입비 할인, 검수비 할인, 최종 결제금액, 최종 상품금액, 최종 VAT, 주문수량, 최종입고 수량**을 **orderId**로 **GroupBy**하여 **총합**

Dm_account

1	**orderId**	**Integer**
2	buyerId	Integer
3	createdAt	Datetime
4	initTotalKRW	Integer
5	initHandlingFee	Integer
6	initQualityInspectionFee	Integer
7	initKRW	Integer
8	shippingCost	Integer
9	productid	Integer
10	initSubtotal	Integer
11	initQuantity	Integer
12	brandId	Integer
13	countryCode	String
14	Status_order	Integer

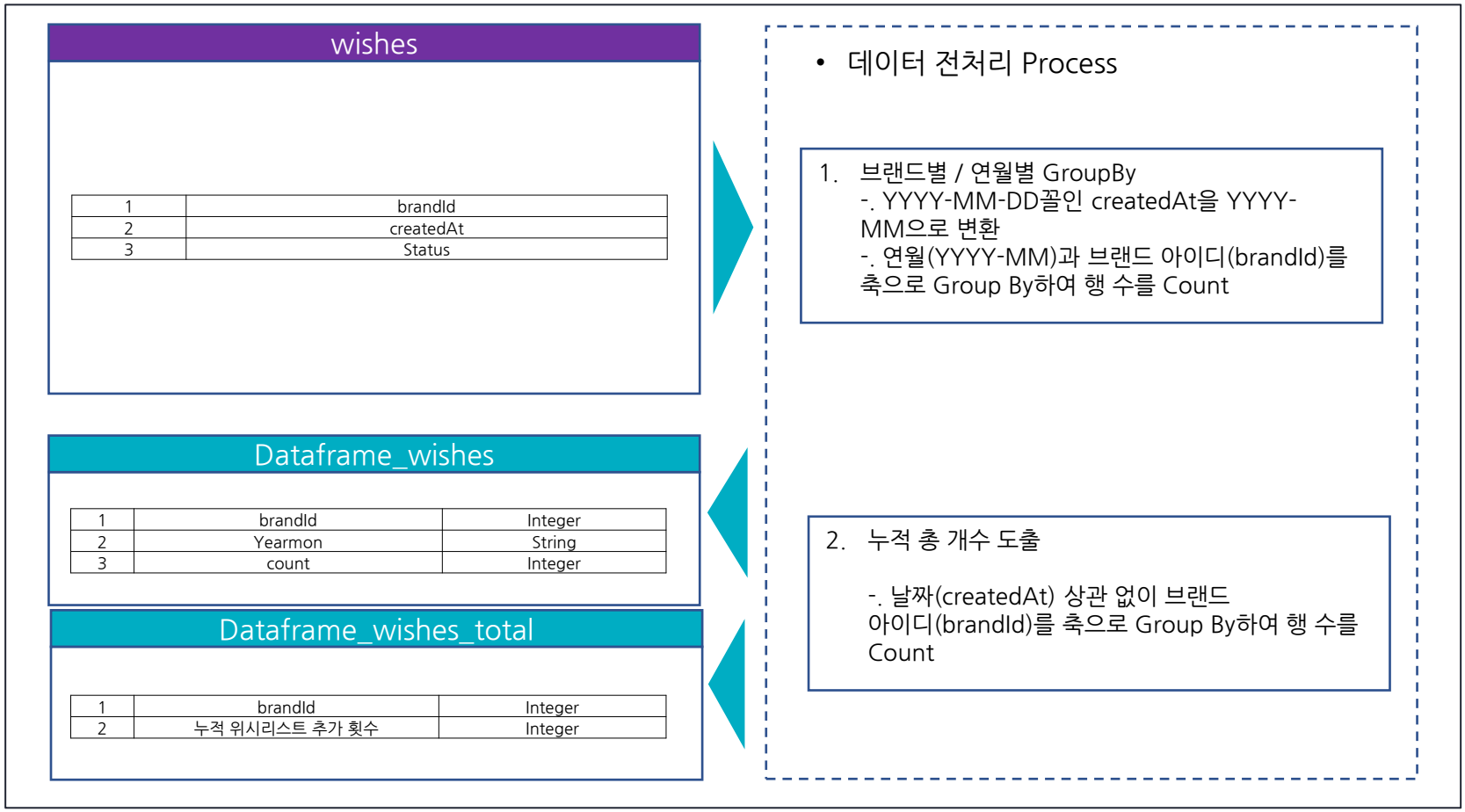
2. 배송 관련 데이터 정리
   -. 상단 GroupBy에서 제외된 배송비는 orderId별로 중복된 값을 제거하여 Unique한 값만 뽑아 압축
   -. 배송번호(tracking_num)을 총 개수와 중복값 제거 unique한 개수만 세서 각각 shipment_num_total과 shipment_num_unique로 선언

#LinkShops

9. BlackLabelReport

9. BlackLabelReport

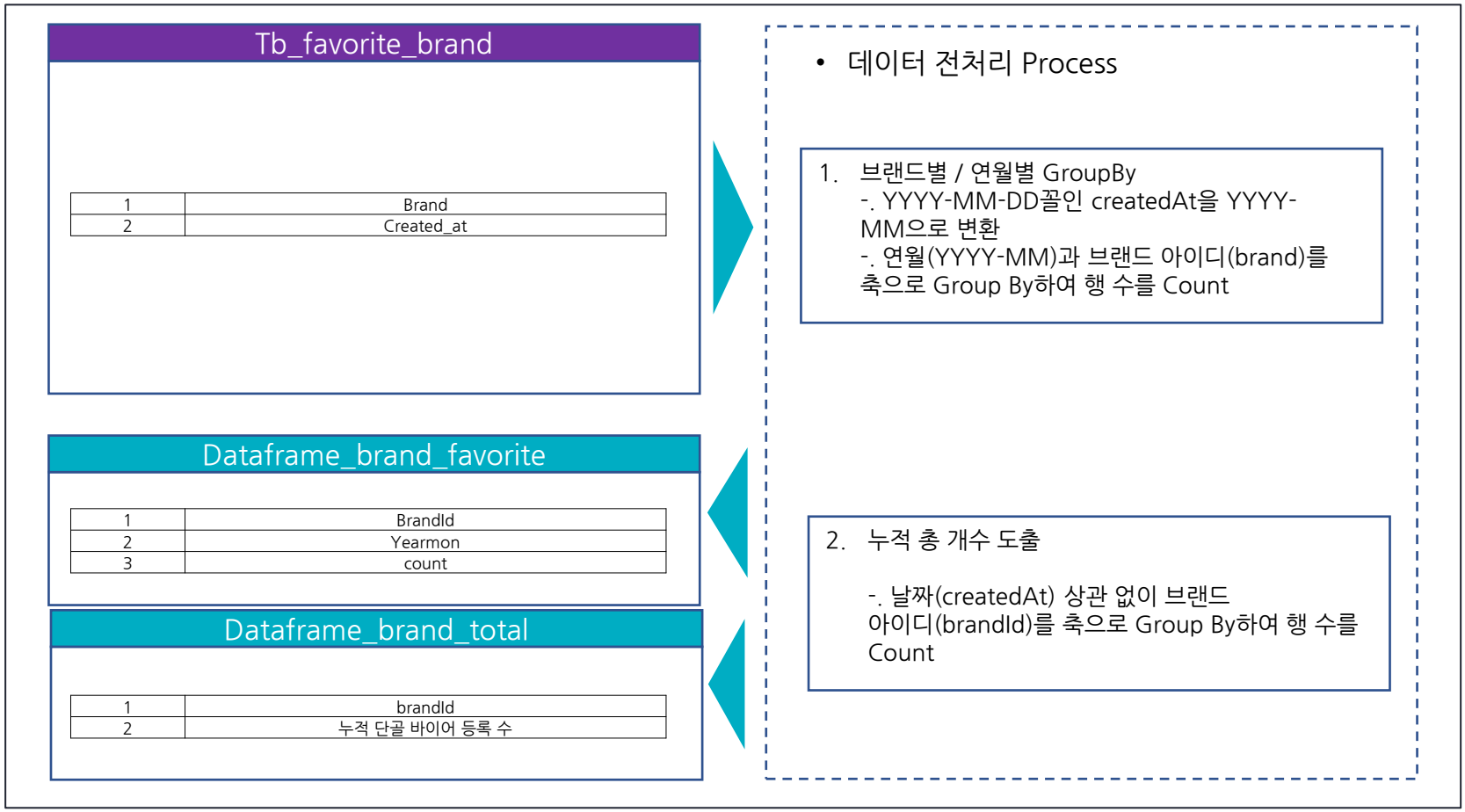
- 블랙라벨 브랜드 한정 서비스로 판매 현황을 셀러에게 제공
- Wishes, 단골 바이어 등록 현황은 RDB에서 직접 추출하여 활용



#LinkShops

9. BlackLabelReport

- 블랙라벨 브랜드 한정 서비스로 판매 현황을 셀러에게 제공
- Wishes, 단골 바이어 등록 현황은 RDB에서 직접 추출하여 활용



#LinkShops

9. BlackLabelReport

- 블랙라벨 브랜드 한정 서비스로 판매 현황을 셀러에게 제공
- 필터링을 통해 기본 데이터를 준비하고, 결측값을 처리

Dm_orderseries

1	**orderId**
2	buyerId
3	productId
3	brandId
4	createdAt
5	initTotalKRW
6	initSubtotal
7	finalSubtotal
8	finalTotalKRW
9	initQuantity
10	finalQuantity
11	Status_order
12	countryCode

Dm_merchandise

1	**ProductId**
2	productName

Dm_Seller

1	**brandId**
2	sellerId

Dm_account

1	**orderId**	**Integer**
2	buyerId	Integer
3	productId	Integer
4	productName	String
5	brandId	Integer
6	sellerId	Integer
7	createdAt	Datetime
8	initTotalKRW	Integer
9	initSubtotal	Integer
10	finalSubtotal	Integer
11	finalTotalKRW	Integer
12	initQuantity	Integer
13	finalQuantity	Integer
14	countryCode	string

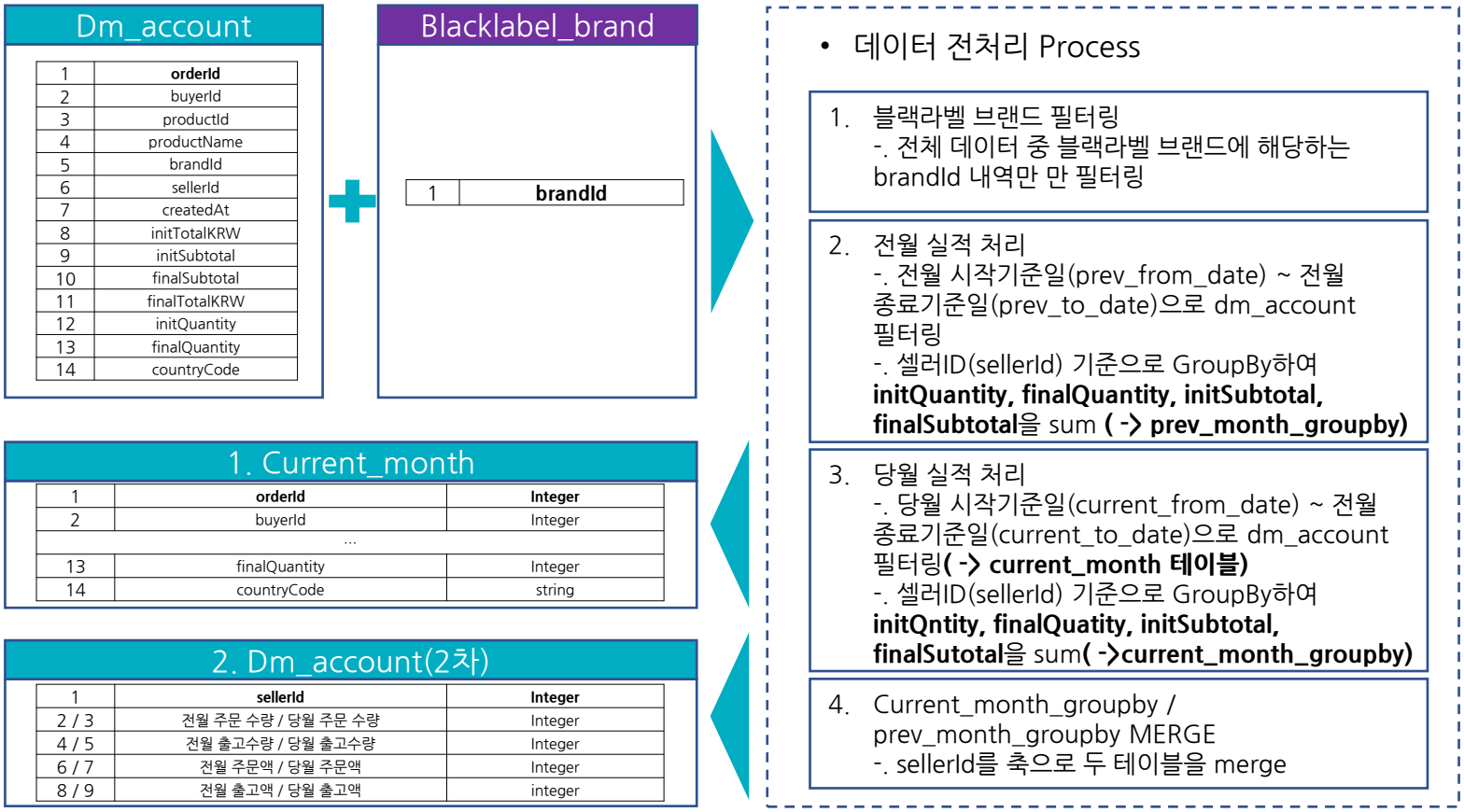
- 데이터 전처리 Process

1. 기본 데이터 준비
   -. Dm_orderseries의 brandId와 dm_seller의 brandId를 축으로 **sellerId**를 가져옴
   -. Dm_orderseries의 productId와 dm_merchandise의 productId를 축으로 **productName**을 가져옴
   -. 주문 상태(status)가 각각 '삭제됨'(-1)과 '생성됨(0)인 주문 이력은 dm_orderseries에서 제외
   -. 상품 가격이 0보다 큰 사례만 추출 (0인 경우는 대개 교/반/샘에 해당하는 사례임)

2. 결측값 처리
   -. initSubtotal이 결측(NaN)인 경우 initTotalKRW의 값으로 대체
   -. finalSubtotal이 결측(NaN)인 경우 finalTotalKRW의 값으로 대체
   -. initSubtotal이나 finalSubtotal이 결측인 사례는 대량주문건 -> 따라서 totalKRW 값으로 대체함

9. BlackLabelReport

- 블랙라벨 브랜드 한정 서비스로 판매 현황을 셀러에게 제공
- 전월 / 당월의 매출 관련 실적을 처리



#LinkShops

9. BlackLabelReport

- 블랙라벨 브랜드 한정 서비스로 판매 현황을 셀러에게 제공
- Top N 국가 / 상품 관련 데이터를 처리

1. Current_month

1	**orderId**	**Integer**
2	buyerId	Integer
3	productid	Integer
4	brandId	Integer
5	createdAt	Datetime
6	initTotalKRW	Integer
7	initSubtotal	Integer
8	finalSubtotal	Integer
9	finalTotalKRW	Integer
10	initQuantity	Integer
11	finalQuantity	Integer
12	Status_order	Integer
13	countryCode	string

2. Dm_account(2차)

1	**sellerId**	**Integer**
2	전월 주문 수량	Integer
3	전월 출고수량	Integer
4	전월 주문액	Integer
5	전월 출고액	Datetime
6	당월 주문 수량	Integer
7	당월 출고수량	Integer
8	당월 주문액	Integer
9	당월 출고액	Integer
10/11/12	출고액 1 / 2 / 3 위 국가	String
13/14/15	1 / 2 / 3위 국가 비중	Integer
16/17/18	판매량 1 / 2 / 3위 상품명	String
19/20/21	판매량 1 / 2 / 3위 상품 판매수량	Integer

- 데이터 전처리 Process

1. 셀러별 Top N 국가 도출
   -. 각 셀러ID별로 countryCode로 finalSubtotal을 GroupBy하여 SUM
   -. SUM한 결과물을 내림차순으로 정렬하고, Top N 개 행을 추출(N 기본값 : 3)
   -. 각 행의 CountryCode와 해당 CountryCode의 SUM한 finalSubtotal 추출

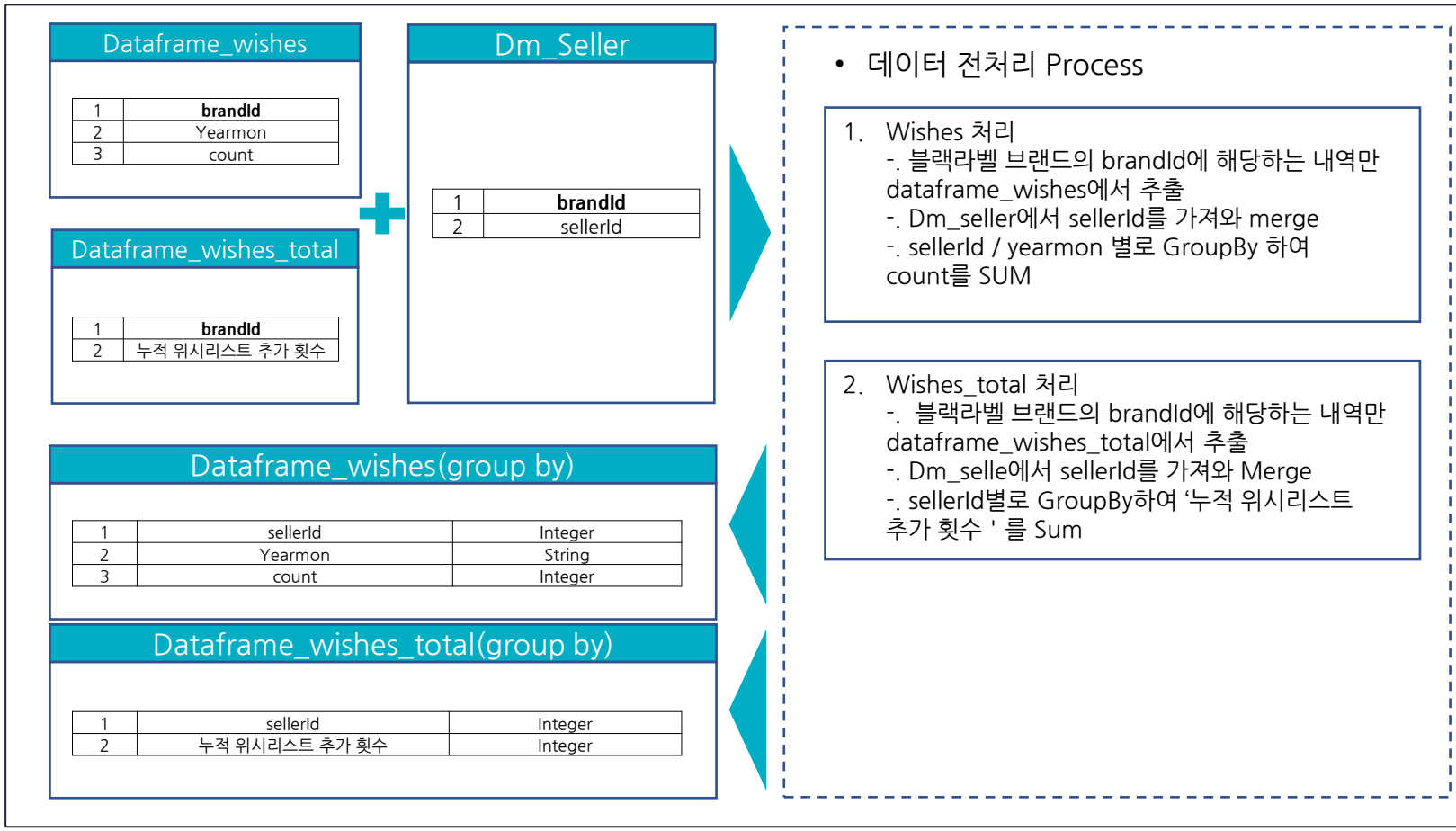
2. 셀러별 Top N 상품 도출
   -. 각 셀러ID별로 ProductName으로 finalSubtotal을 GroupBy하여 SUM (productId가 아닌 productName으로 Groupby하는 이유는, 상품 재등록등의 이유로 동일 상품이 다른 productId를 부여받는 경우를 다루기 위함)
   -. SUM한 결과물을 내림차순으로 정렬하고, Top N 개 행을 추출(N 기본값 : 3)
   -. 각 행의 productname과 해당 CountryCode의 SUM한 finalSubtotal 추출

3. Top N 국가 / 상품 Merge
   -. sellerId를 축으로 두 테이블 Merge

#LinkShops

9. BlackLabelReport

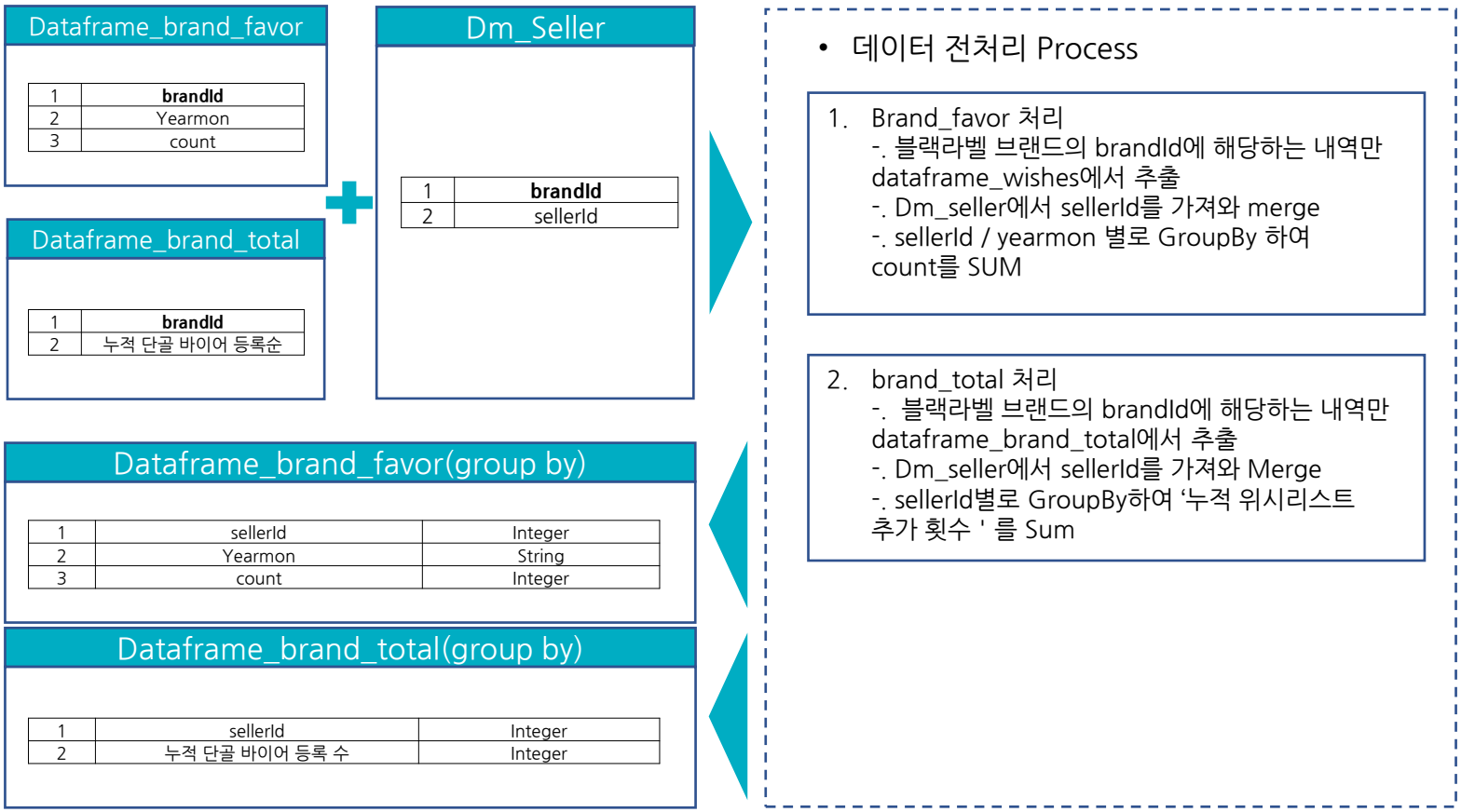
- 블랙라벨 브랜드 한정 서비스로 판매 현황을 셀러에게 제공
- Top N 국가 / 상품 관련 데이터를 처리



#LinkShops

9. BlackLabelReport

- 블랙라벨 브랜드 한정 서비스로 판매 현황을 셀러에게 제공
- Top N 국가 / 상품 관련 데이터를 처리



#LinkShops

9. BlackLabelReport

- 블랙라벨 브랜드 한정 서비스로 판매 현황을 셀러에게 제공
- Top N 국가 / 상품 관련 데이터를 처리

Dataframe_wishes(group by)

1	sellerId	Integer
2	Yearmon	String
3	count	Integer

Dataframe_brand_favor(group by)

1	sellerId	Integer
2	Yearmon	String
3	count	Integer

temp

1	sellerId	Integer
2	전월 위시리스트 추가 횟수	Integer
3	전월 단골 바이어 등록 수	Integer
4	당월 위시리스트 추가 횟수	Integer
5	당월 단골 바이어 등록 수	Integer

- 데이터 전처리 Process

1. Dataframe_wishes / dataframe_brand_favor 피보팅
   -. Yearmon을 열로, Count를 값으로, SellerId를 인덱스로 갖는 테이블로 형태를 재구축(피보팅)

brandId	yearmon	count
12375	2022-11	35
12375	2022-12	41
…		

index	2022-11	2022-12
12375	35	41
…		

<피보팅 예시>

2. Dataframe_wishes / dataframe_brand_favor MERGE
   -. 두 테이블을 sellerId를 축으로 하여 Merge 수행

#LinkShops

9. BlackLabelReport

- 블랙라벨 브랜드 한정 서비스로 판매 현황을 셀러에게 제공
- Top N 국가 / 상품 관련 데이터를 처리

temp

1	**sellerId**
2	전월 위시리스트 추가 횟수
3	전월 단골 바이어 등록 수
4	당월 위시리스트 추가 횟수
5	당월 단골 바이어 등록 수

Dataframe_wishes_total(group by)

1	**sellerId**
2	누적 위시리스트 추가 횟수

Dataframe_brand_total(group by)

1	**sellerId**
2	누적 단골 바이어 등록수

Dm_account

1	셀러명	string
2	**sellerId**	**integer**
3	집계기간	string
4	번역처리값	-
5	당월 총 큐레이션 제공 횟수	-
6	전월 대비 당월 주문 금액	-
	...	
35	판매량 2위 상품명	string
36	판매량 2위 상품 판매 수량	integer
37	판매량 3위 상품명	string
38	판매량 3위 상품 판매 수량	integer

- 데이터 전처리 Process

1. Temp와 dataframe_wishes_total, dataframe_brand_total MERGE
   -. 앞에서 처리한 temp와 wishes_total, brand_total을 SellerID를 축으로 MERGE
   -. 모든 테이블에서 누락되는 내역이 없도록 OUTER JOIN을 수행

2. 한글명 컬럼으로 변경
   -. 전월에 해당하는 yearmon을 '전월 ~'로, 당월에 해당하는 yearmon을 '당월 ~'로 각각 변경

3. 공란 처리 컬럼 추가
   -. 현업 요구에 따라 **'번역처리값','당월 총 큐레이션 제공 횟수','전월 대비 당월 주문 금액','전월 대비 당월 주문액 변화(%)','전월 대비 당월 주문 수량 변화(건), '전월','당월','전월 대비 당월 출고액 변화(%), 전월 대비 당월 출고 수량 변화 '** 를 NaN값으로 채워 추가

#LinkShops

10. CouplingPassList

10. CouplingPassList

- couplingPassBuyer 데이터에 수반되는 보조 데이터로 커플링패스 가입 현황을 공유
- 커플링패스 가입 상태인 바이어의 Id만 추출

Tb_discount_promotion

1	Discount_promotion_id
2	Description
3	start_at
4	end_at

Tb_discount_promotion_target_buyer

1	buyerId
2	Discount_promotion_id
3	Use_yn
4	Created_at

- 데이터 전처리 Process

1. 데이터 필터링
   -. Tb_discount_promotion_target_buyer(이하 tdptb)의 Use_yn이 'Y'인 buyerId만 추출
   -. Tb_discount_promotion의 description에 '커플링' 이 들어가는 discount_promotion_id만 추출
   -. Use_yn = 'Y'만 필터링한 Tdbtp에서 해당 discount_promotion_id만 다시 필터링

Dataframe_coupling

1	buyerId	Integer
2	Description	String
3	Discount_promotion_id	Integer
4	createdAt	Datetime
5	startAt	Datetime
6	endAt	Datetime

10. CouplingPassList

- couplingPassBuyer 데이터에 수반되는 보조 데이터로 커플링패스 가입 현황을 공유
- 단위 기간 기준으로 활성화 상태인 모든 커플링패스 이력을 추출

Dataframe_coupling

1	buyerId
2	Description
3	Discount_promotion_id
4	createdAt
5	startAt
6	endAt

dm_buyer

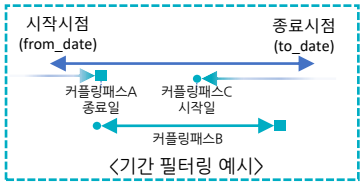
1	buyerId
2	Discount_promotion_id
3	Name
4	bizName
5	roles

Dm_account

1	buyerId	Integer
2	Discount_promotion_id	Integer
3	Description	String
3	Name	String
4	bizName	String
5	roles	string

- 데이터 전처리 Process

1. 시간 기준 데이터 필터링
   -. Dataframe_coupling에서 종료일(endAt)이 시작 시점(from_date)보단 뒤에 있고, 시작일(startAt)은 종료시점(to_date)보단 앞인 buyerId 추출



<기간 필터링 예시>

From_date ~ to_date
해당 기간동안
커플링패스 A,B,C가
모두 **포함**되어야함

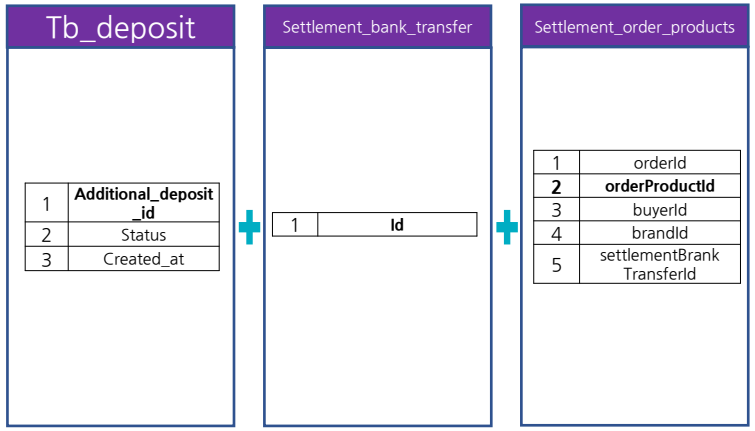
2. Dm_buyer와 Merge
   -. Dm_buyer에서 이름(name), 상호명(bizName), 바이어구분(roles)를 가져와 Merge 수행

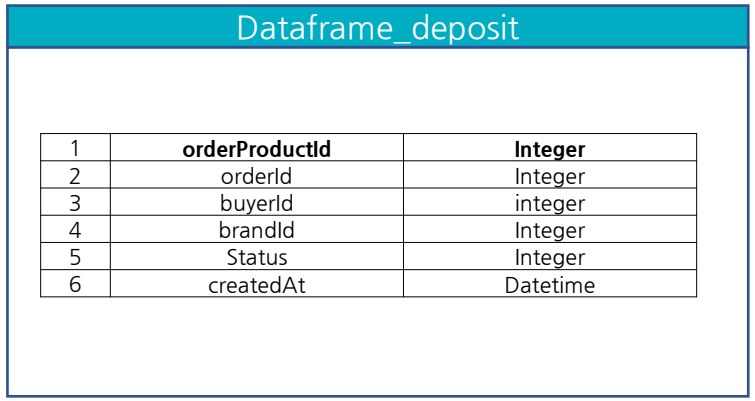
#LinkShops

11. OmittedRefund

11. omittedRefund

- 바이어에게 환불을 이미 해줬으나 셀러에게 환불금액을 받지 못한 '미입금 사례' 중 CO팀에서 이력 생성을 하지 않은 누락 사례 도출
- 입금 사례를 dataframe_deposit으로 도출





- 데이터 전처리 Process

1. Settlement ID기반 데이터 필터링
   -. settlement_bank_transfer에서 ID를 가져와 tb_deposit과 JOIN
   -. Settlement_bank_transfer에 이력이 존재하는 경우는 셀러가 LS에 정상적으로 환불금액을 이체한 경우를 의미(우리의 관심대상 외)
   -. 이 이력 밖에 존재하는 구매 이력은 **'셀러가 LS에 환불금액을 이체하지 않은 사례'** 로 간주 가능

2. 필요 데이터 JOIN
   -. settlement_order_products에서 orderProductId, orderId, buyerId, brandId를 가져옴
   -. Tb_deposit에서 status와 created_at을 가져옴
   -. Created_at 기준으로 시작기준일(from_date) 이후 데이터만 가져오도록 필터링

11. omittedRefund

- 바이어에게 환불을 이미 해줬으나 셀러에게 환불금액을 받지 못한 '미입금 사례' 중 CO팀에서 이력 생성을 하지 않은 누락 사례 도출
- 추출을 위한 데이터 사전처리 수행

dm_orderseries 데이터마트

1	**orderId**
2	buyerId
3	productId
4	createdAt
6	initTotalKRW
7	initHandlingFee
8	initQualityInspectionFee
9	initSubtotal
10	initQuantity
11	initTax
12	initKRW
13	finalTotalKRW
14	finalSubtotal
15	settledPrice
16	settledQuantity
17	Status_order_prod
18	Status_order

orderseries

1	**orderId**	**Integer**
2	buyerId	Integer
3	productId	Integer
4	createdAt	Datetime
6	initTotalKRW	Integer
7	initHandlingFee	Integer
8	initQualityInspectionFee	Integer
9	initSubtotal	Integer
10	initQuantity	Integer
11	initTax	Integer
12	initKRW	Integer
13	finalTotalKRW	Integer
14	finalSubtotal	Integer
15	settledPrice	Integer
16	settledQuantity	Integer
17	Status_order_prod	Integer
18	Status_order	Integer

- 데이터 전처리 Process

1. 시간 기준 데이터 필터링
   -. createdAt 을 기준시작일(from_date) ~ 기준종료일(to_date) 사이로 절단

2. 대량 주문건 처리
   -. productId가 'bp_' 꼴로 들어가있는 대량주문건만 추출하여 Initsubtotal을 initTotalKRW로 대체
   -. 다시 initTotalKRW는 initSubtotal + initTax + initHandlingFee의 합산값으로 대체
   (대량주문건은 데이터마트 처리 시 initSubtotal 개념에 해당하는 값을 initTotalKRW에 저장해서 이런 후처리가 필요)
   -. finalSubtotal은 finalTotalKRW로 대체

3. 데이터 필터링
   -. initTotalKRW가 0 초과인 내역만 추출
   -. initTotalKRW가 0인 내역은 교 / 반 / 샘임

#LinkShops

11. omittedRefund

- 바이어에게 환불을 이미 해줬으나 셀러에게 환불금액을 받지 못한 '미입금 사례' 중 CO팀에서 이력 생성을 하지 않은 누락 사례 도출
- 미송 후 품절 처리 되었고, 셀러한테 돌려받을 금액이 있는 경우만 추출

orderseries

1	**orderId**
2	buyerId
3	productId
4	createdAt
6	initTotalKRW
7	initHandlingFee
8	initQualityInspectionFee
9	initSubtotal
10	initQuantity
11	initTax
12	initKRW
13	finalTotalKRW
14	finalSubtotal
15	settledPrice
16	settledQuantity
17	Status_order_prod
18	Status_order

Orderseries_refund

1	**orderId**	**Integer**
2	buyerId	Integer
3	productId	Integer
4	createdAt	Datetime
6	initTotalKRW	Integer
7	initHandlingFee	Integer
8	initQualityInspectionFee	Integer
9	initSubtotal	Integer
10	initQuantity	Integer
11	initTax	Integer
12	initKRW	Integer
13	finalTotalKRW	Integer
14	finalSubtotal	Integer
15	settledPrice	Integer
16	settledQuantity	Integer
17	Status_order_prod	Integer
18	Status_order	Integer

- 데이터 전처리 Process

1. 미송 후 품절 처리 내역 필터링
   -. **바이어**가 **미송시 대처 방법**을 **'미송 ' 으로 선택**했으나 최종적으로 **품절**로 **마무리된 사례**들 중 **셀러한테 돌려받을 금액**이 있는 경우만 필터링 수행
   -. 다음의 기준으로 내역을 필터링

	값	비교 방법	비교 대상	이유
1	**settledPrice**	보다 큼(>)	0	LS가 셀러한테 준 정산액 존재
2	**Status(order_product)**	같음(=)	400	상품주문 상태는 '픽업 예정 없음(품절)' 로 마무리
3	**settledQuantity**	보다 큼(>)	0	LS가 인정한 정산 수량이 존재
4	**Status(order)**	아님(Not in)	[0, -1]	주문 상태가 취소됨, 생성됨은 제외
5	**waitShipment Type**	같음(=)	100	바이어가 선택한 해당 상품주문의 미송시 대응 방법이 '미송'이어야함 (즉시환불 제외)
6	**InitQuantity**	같지 않음(/=)	finalquantity	주문수량과 출고수량에 차이 존재
7	**finalSubtotal**	같지 않음(/=)	settledPrice	출고액과 정산액간 차이 존재

#LinkShops

11. omittedRefund

- 바이어에게 환불을 이미 해줬으나 셀러에게 환불금액을 받지 못한 '미입금 사례' 중 CO팀에서 이력 생성을 하지 않은 누락 사례 도출
- 셀러한테 돌려받을 금액 중 실제 입금이 완료된 사례는 제외

Orderseries_refund

1	**orderId**
2	buyerId
3	productId
4	createdAt
6	initTotalKRW
7	initHandlingFee
8	initQualityInspectionFee
9	initSubtotal
10	initQuantity
11	initTax
12	initKRW
13	finalTotalKRW
14	finalSubtotal
15	settledPrice
16	settledQuantity
17	Status_order_prod
18	Status_order

Dataframe_deposit

1	**orderProductId**
2	orderId
3	buyerId
4	brandId
5	Status
6	createdAt

Dm_account

1	**buyerId**	**Integer**
2	processedDate	Datetime
3	orderProductId	Integer
3	initQuantity	Integer
4	finalQuantity	Intiger
5	settledQuantity	Integer
6	initKRW	Integer
7	finalSubtotal	Integer
8	settledPrice	Integer
9	finalKRW	Integer
10	initTotalKRW	Integer
11	finalTotalKRW	integer

- 데이터 전처리 Process

1. 입금 완료된 내역 제외
   -. 앞서 추출한 셀러한테 돌려받을 금액 내역 중 셀러가 LS에 송금 완료한 내역은 제외 필요
   -. 맨 처음 도출한 dataframe_deposit을 활용
   -. Dataframe_depoisit에 존재하는 orderProductId는 orderseries_refund에서 제외

12. ZeroRatingEvent

12. zeroRatingEvent

- 사입수수료 0% 이벤트 적용 주문 내역 확인을 위한 데이터
- 필터링을 통해 데이터를 준비하고, 0% 이벤트 대상 내역만 추출

dm_orderseries 데이터마트

1	**orderId**
2	buyerId
3	productId
4	createdAt
6	initTotalKRW
7	initHandlingFee
8	initQualityInspectionFee
9	initSubtotal
10	initQuantity
11	initTax
12	initKRW
13	finalTotalKRW
14	finalSubtotal
15	settledPrice
16	settledQuantity
17	Status_order_prod
18	Status_order

Dm_acount

1	**orderId**	**Integer**
2	buyerId	Integer
3	Month	String
4	productId	Integer
5	createdAt	Datetime
6	initTotalKRW	Integer
7	initHandlingFee	Integer
8	initQualityInspectionFee	Integer
9	initSubtotal	Integer
10	Status_order	Integer

- 데이터 전처리 Process

1. 데이터 필터링
   -. createdAt 을 기준시작일(from_date) ~ 기준종료일(to_date) 사이로 절단
   -. createdAt에서 연 / 월 정보를 뽑아 month(연-월) 컬럼 생성
   -. 주문 상태가 준비중(100), 발송됨(200), 발송완료됨(201), 부분발송됨(202)인 내역만 추출
   -. productId가 'bp_'로 구성된 대량주문건은 제외

2. 0% 이벤트 대상 내역 추출
   -. 사입수수료가 0인 주문(orderId 기준)만 추출

#LinkShops

12. zeroRatingEvent

- 사입수수료 0% 이벤트 적용 주문 내역 확인을 위한 데이터
- orderProductId 기준인 데이터를 orderId 기준으로 변환

dm_account

1	**orderId**	**Integer**
2	buyerId	Integer
3	Month	String
4	productId	Integer
5	createdAt	Datetime
6	initTotalKRW	Integer
7	initHandlingFee	Integer
8	initQualityInspectionFee	Integer
9	initSubtotal	Integer
10	Status_order	Integer

orderseries

1	**buyerId**	**Integer**
2	년-월	String
3	결제금액	Integer
4	상품금액	Integer
5	판매수수료	Integer
6	판매수수료 할인금액	Integer
7	검수검품비	Integer
8	배송비	Integer
9	국가코드	string

- 데이터 전처리 Process

1. 상품주문번호 단위를 주문번호 단위로 통합(**groupby Sum**)
   -. Dm_orderseries는 상품주문번호(**orderProductId**)를 Key로 갖는 데이터마트
   -. 이를 주문번호(**orderId**)단위로 처리하기 위해 변환 필요
   -. **orderId로 Groupby**하여 **initTotalKRW, initSubtotal, initHandlingFee, dcHandlingFee, initQualityInspectionFee**를 SUM

2. 배송비 처리(shippingCost)
   -. shippingCost는 orderProductId에 종속되지 않고 orderId에 종속된 값
   -. shippingCost만 따로 분리하여 orderId를 축으로 중복값을 제거하여 압축(drop_duplicates)
   -. 위에서 처리한 orderId 단위로 변환된 데이터프레임과 Merge 수행

3. buyerId 단위로 통합(groupby Sum)
   -. orderId 단위로 된 Merge 데이터 프레임을 buyerId로 변환하기 위해 groupby Sum 수행

13. FireSales

13. FireSales

- 땡처리 기획전용 데이터를 추출
- 제품명 앞에 '20XXsales' 접두사가 붙은 상품들의 낱장 가능 여부를 판별

products

1	**id**	**Integer**
2	sku	String
3	name	Jsonb
4	brand	Jsonb
5	creator	Jsonb
6	isActive	Bool
7	categories	Jsonb
8	appImages	Json
9	data	Jsonb
10	createdAt	Datetime
11	updatedAt	datetime
...		

- 데이터 전처리 Process

1. 땡처리 대상 상품 추출
   -. 상품명(name)에 '20XXsales' 접두사가 달린 상품ID(productid)만 추출

Dataframe_product_sale

1	**productId**	**Integer**
2	Name	String
3	productStatus	Integer
4	SKUData	Json
5	Brand	Json
6	createdAt_prod	Datetime
7	updatedAt_prod	datetime

13. FireSales

- 땡처리 기획전용 데이터를 추출
- 멤버십 관련 데이터를 처리하고, 기타 데이터를 Json에서 추출

Dataframe_product_sales

1	**productId**
2	Name
3	productStatus
4	SKUData
5	Brand
6	createdAt_prod
7	updatedAt_prod

dm_seller

1	**brandId**
2	Name
2	Membership
3	membershipStatus
4	membershipGrade
5	Building
6	Status_brand

Dm_marketing

1	**productId**	**Integer**
2	Name	String
3	productStatus	Integer
4	SKUData	Json
5	Brand	Integer
6	createdAt	Datetime
7	updatedAt	datetime
8	isBulkOrder	Boolean
9	brandId	Integer
10	Name(상품명)	String
11	productStatus	Integer
12	Name(브랜드명)	String
13	Building	String
14	brandStatus	Integer
15	Basic/global	Boolean
16	membershipGrade	String
17	isminimumquantity	Boolean

- 데이터 전처리 Process

1. 멤버십 등급 추출
   -. membershipGrade가 공란(len(x) == 0)인 경우, '일반 ' 으로 하드코딩
   -. Membership이 1이고, membershipStatus가 200인 경우 '글로벌', 그렇지 않은 경우 '베이직 ' 으로 하드 코딩

2. 각종 데이터 추출(Json 까기)
   -. Brand에서 brandId 추출
   -. SKUData의 name -> 'ko'에서 Name(상품명) 추출
   -. SKUData의 data -> 'isMinimumQuantity'에서 isminimumquantity(낱장 가능 여부) 추출
   -. SKUData의 status에서 productStatus(상품 상태) 추출

3. Dm_seller / dm_merchandise 간 Join
   -. brandId를 축으로 Dm_seller와 dataframe_product_sales를 Join

13. FireSales

- 땡처리 기획전용 데이터를 추출
- 활성(판매중) 상태인 상품만 추출하고, 낱장 가능 여부를 정오표(O / X)로 변환

Dm_marketing

1	**productId**	**Integer**
2	Name	String
3	productStatus	Integer
4	SKUData	Json
5	Brand	Integer
6	createdAt	Datetime
7	updatedAt	datetime
8	isBulkOrder	Boolean
9	brandId	Integer
10	Name(상품명)	String
11	productStatus	Integer
12	Name(브랜드명)	String
13	Building	String
14	brandStatus	Integer
15	Basic/global	Boolean
16	membershipGrade	String
17	isminimumquantity	Boolean

Dm_marketing(최종)

1	**productid**	**Integer**
2	상품명	String
3	등록날짜	Integer
4	업데이트날짜	Integer
5	브랜드id	Integer
6	브랜드명	String
7	멤버십그레이드	String
8	상가	String
9	낱장 가능	String

- 데이터 전처리 Process

1. 활성 상태 제품 추출
   -. 브랜드 상태가 0(= 폐업)이 아니면서 상품 상태가 100(=품절)이 아닌 상품만 추출

2. 낱장 가능 여부 정오표(O / X) 변환
   -. isMinimumQuantity가 False이면 'O'(낱장 가능)으로, 그 외의 경우는 'X'로 변환

#LinkShops