[ Anything Deal ] DB설계 프로젝트

3조 최준현, 이기순, 정수현

1 주제

공동구매 기반 오픈마켓

2 개발 배경

1. SNS, 홈쇼핑, 라이브 커머스 등 일부 채널에서만 이루어지는 대량판매 할인 서비스는 광고 비용부터 입점 비용, 쇼 호스트 섭외까지 상품 판매를 하기 위해 부담해야 하는 비용이 상당하여 일반 업체 또는 자영업자들이 이용하기에는 어려움이 많음
2. 일반 판매자들도 폭 넓게 이용할 수 있을 만큼 대량판매 시스템이 잘 갖추어진 매체가 현저하게 없음.

3 설계 목적

1. 영세한 자영업자를 비롯하여 전자상거래를 하는 업체들에게 큰 비용 부담 없이 누구나 대량판매를 기반으로 규모의 경제를 실현시킬 수 있는 편리한 판매 시스템을 만들고자 함
2. 인원 별, 할인율, 마감일 기준에 따라 상품 입고 컨디션과 판매전략을 달리하여 판매 리스크를 줄이고자 함
3. 판매되었던 데이터들을 베이스로 판매자에게 꼭 필요한 매출 관련 리포트를 제공하여 양질의 효과를 보고자 함

4 기대효과

1. 인풋 비용을 제거하여 좀 더 쉽고 편리하게 많은 판매자들의 입점이 가능
2. 대량 판매를 통해 총 판매이윤이 올라가면 양질의 다양한 상품들이 거래가 될 것으로 예상
3. 판매자는 재고 리스크 예측이 가능하며 판매 기간에 따라 입고 및 판매를 조절하여 손실을 최소화할 수 있음
4. 소비자는 시장 거래 금액 보다 더욱 합리적인 가격으로 다양한 상품을 구매할 수 있어 수요 상승
5. 개인이 아닌 불특정 다수의 구매자들이 모여 거래가 이루어짐으로써 소비 부담을 줄여 거래량 증가

5 주요기능

1. 구매인원 대비 할인율 자동 변동

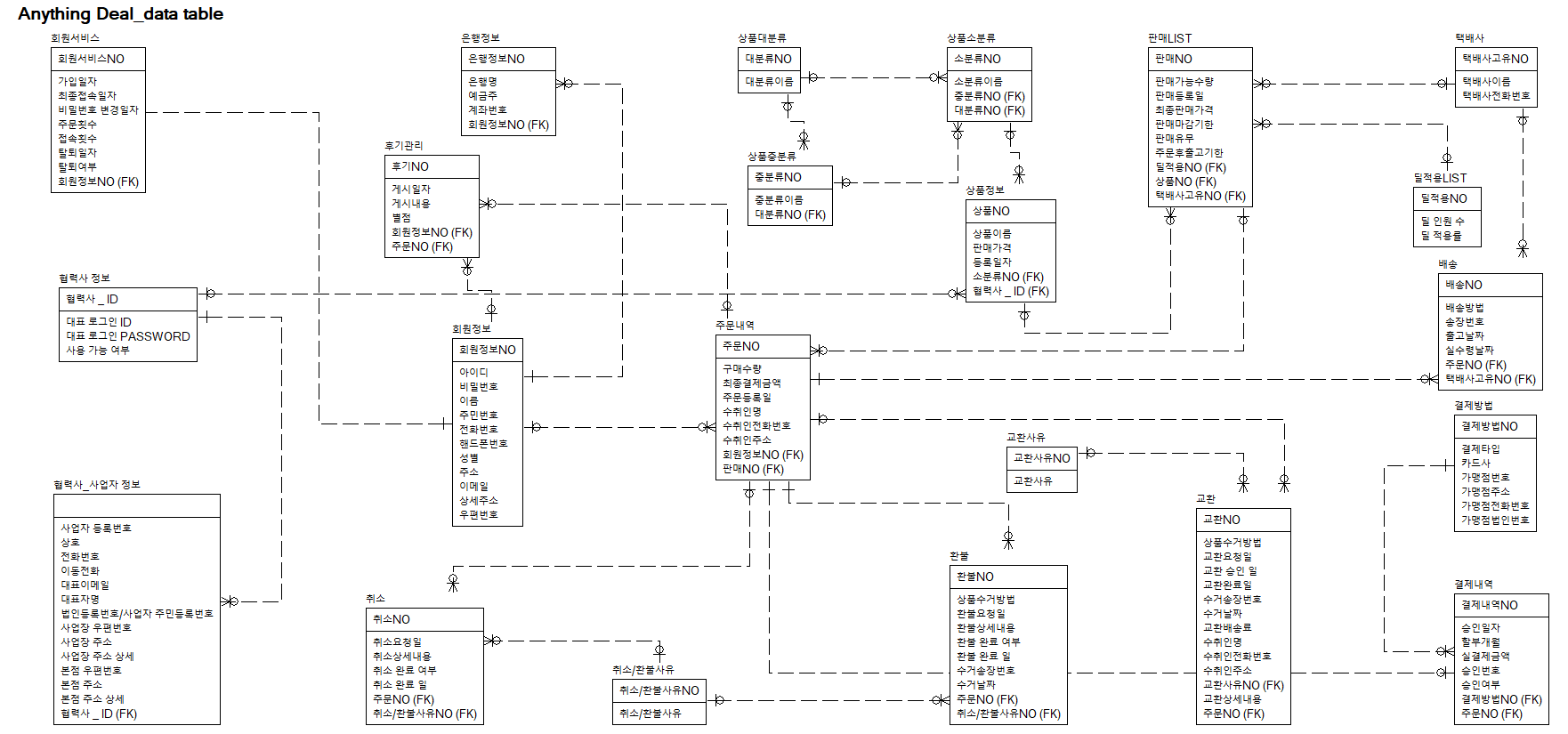
* 판매기간 동안 모인 판매 수량에 따라 판매자가 설정해 놓은 할인율이 마감일에 맞추어 상이하게 적용 됨

1. 선 결제 후 자동 취소 및 재 결제 시스템

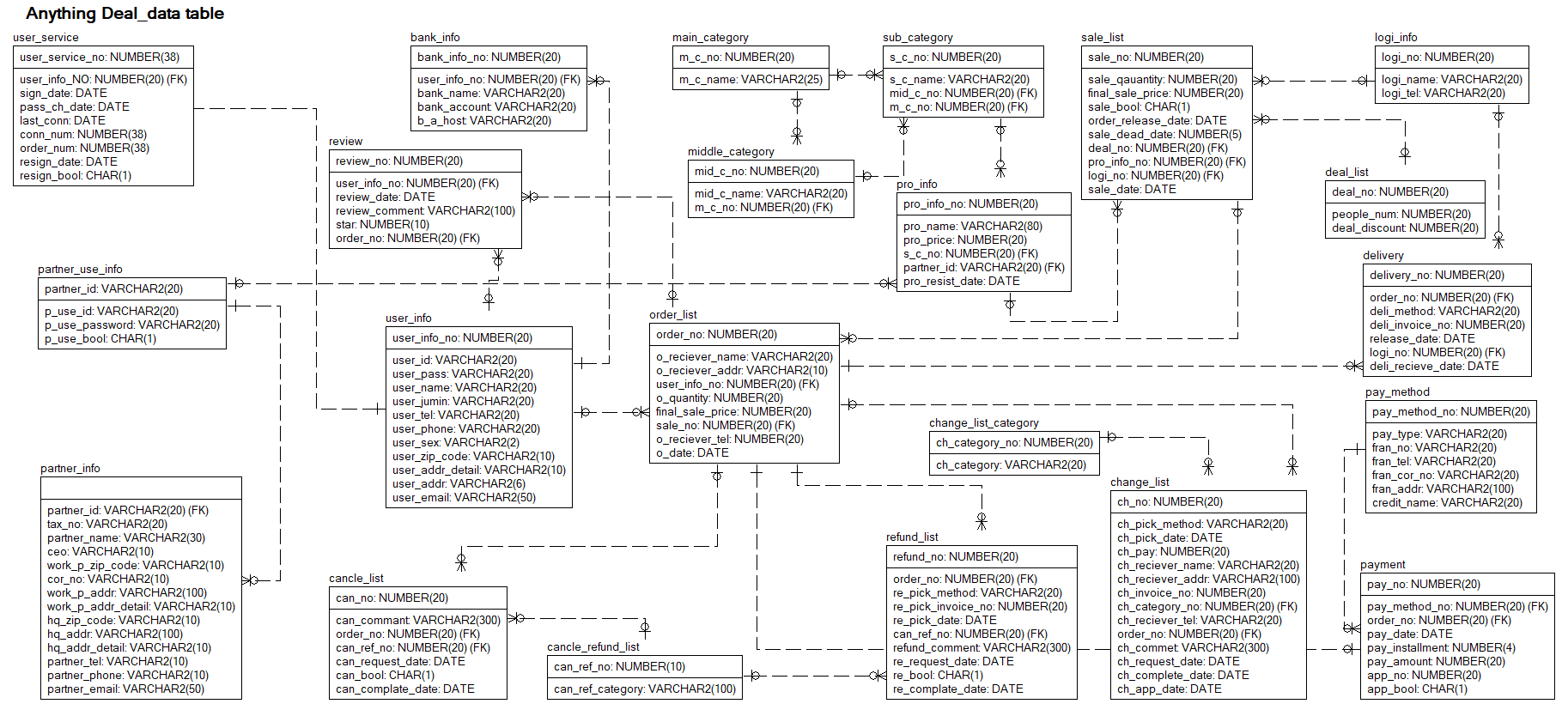
* 구매 일 기준 금액으로 결제가 진행 되지만 마감일 기준 할인율이 다시 적용되어 결제 취소 후 최종 할인율 적용 받아 재 결제
* 소비자의 불편함과 불이익(시점에 따른 금액 차이 피해) 을 없앨 수 있음

1. 업체들에게 꼭 필요한 매출 관련 리포트를 제공

* 분기별, 카테고리별, 상품별, 고객별, 결제수단별 등등 매출 자료 및 예상 자료 분석 기능
* DATA\_TABLE\_LOGICAL



* DATA\_TABLE\_PHYSICAL



* 마케팅

|  |  |  |
| --- | --- | --- |
|  | 리포트 항목 | SQL 쿼리 |
| 1 | 요일 별 상품 구매 현황 | select to\_char(o\_date, 'day') "요일", count(\*) "판매갯수"from order\_list  group by to\_char(o\_date,'day'), to\_char(o\_date - 1, 'D')  order by to\_char(o\_date - 1, 'D'); |
|  |
| 2 | 기간 별 객 단가 변동 현황 | select trunc((sum(final\_sale\_price)/count(user\_info\_no)),-2) "인당평균구매금액" from order\_list; |
|  |
| 3 | 특정 업체의 특정 기간 동안 판매된 상품들의 남녀 비율 현황 | select p.partner\_id "업체번호", u.user\_sex "성별", count(u.user\_sex) "인원수"  from order\_list o, sale\_list s , user\_info u, pro\_info p  where o.sale\_no=s.sale\_no and s.pro\_info\_no=p.pro\_info\_no  and o.user\_info\_no = u.user\_info\_no  and o.o\_date between '2021-01-01' and '2021-01-31'  and p.partner\_id = 19  group by p.partner\_id, u.user\_sex; |
|  |
|  | 리포트 항목 | SQL 쿼리 |
| 4 | 특정 업체의 특정 기간 동안 판매된 상품들의 연령대 비율 현황 | select p.partner\_id "업체번호", nvl((f\_age(user\_jumin)),0) "연령대", count(\*) "판매갯수"  from order\_list o, sale\_list s , user\_info u, pro\_info p  where o.sale\_no=s.sale\_no and s.pro\_info\_no=p.pro\_info\_no and o.user\_info\_no = u.user\_info\_no and o.o\_date between '2021-01-01' and '2021-01-31' and p.partner\_id = 19  group by p.partner\_id, nvl((f\_age(user\_jumin)),0)  having nvl((f\_age(user\_jumin)),0) >= 20 and nvl((f\_age(user\_jumin)),0) <= 70  order by 2; |
|  |
| 5 | 연령별, 카테고리별 주문 현황 | select f\_age(user\_jumin) "연령대",  sum(decode(p.s\_c\_no, 1,1)) 소분류\_1,  sum(decode(p.s\_c\_no, 2,1)) 소분류\_2,  sum(decode(p.s\_c\_no, 3,1)) 소분류\_3  from user\_info u, order\_list o, sale\_list s, pro\_info p  where u.user\_info\_no = o.user\_info\_no and o.sale\_no = s.sale\_no and s.pro\_info\_no = p.pro\_info\_no group by f\_age(user\_jumin) order by 1 asc; |
|  |
|  | 리포트 항목 | SQL 쿼리 |
| 6 | 지역별, 연령별 회원 분포 현황 | select f\_age(user\_jumin) "연령대",  sum(decode(user\_addr,'서울',1)) 서울, sum(decode(user\_addr,'인천',1)) 인천, sum(decode(user\_addr,'광주',1)) 광주, sum(decode(user\_addr,'세종',1)) 세종, sum(decode(user\_addr,'대구',1)) 대구, sum(decode(user\_addr,'부산',1)) 부산, sum(decode(user\_addr,'제주',1)) 제주  from user\_info  group by f\_age(user\_jumin) order by 1; |
|  |
| 7 | 주문 목록 중  카드사별 이용 현황 | select pm.pay\_method\_no "결제방법번호", pm.credit\_name "카드사", count(pm.pay\_method\_no) "이용횟수"  from pay\_method pm, payment pa, order\_list o, sale\_list s  where pm.pay\_method\_no = pa.pay\_method\_no and pa.order\_no = o.order\_no and o.sale\_no = s.sale\_no  having pm.pay\_method\_no <= 8  group by pm.pay\_method\_no, pm.credit\_name  order by 1; |
|  |
|  | 리포트 항목 | SQL 쿼리 |
| 8 | 특정 일 하루 동안 발생한 주문  목록 중 상품별 결제타입 | select o.order\_no "주문번호",  sum(decode(pm.pay\_type,'카드',1)) 카드, sum(decode(pm.pay\_type,'계좌이체',1)) 계좌이체  from pay\_method pm, payment pa, order\_list o  where pm.pay\_method\_no = pa.pay\_method\_no and pa.order\_no = o.order\_no and o.o\_date = '2021-01-01' group by o.order\_no; |
|  |
| 9 | 연령대 별  결제 수단 현황 | select f\_age(user\_jumin) "연령대",  sum(decode(pm.pay\_type,'카드',1)) 카드, sum(decode(pm.pay\_type,'계좌이체',1)) 계좌이체  from pay\_method pm, payment pa, order\_list o, user\_info u  where pm.pay\_method\_no = pa.pay\_method\_no and pa.order\_no = o.order\_no and o.user\_info\_no = u.user\_info\_no group by f\_age(user\_jumin) having f\_age(user\_jumin) >= 20 and f\_age(user\_jumin) <= 70  order by 1 asc; |
|  |
|  | 리포트 항목 | SQL 쿼리 |
| 10 | 연령대 별  선호하는 구매 가격대 | select f\_age(user\_jumin) "연령대",  count(case when final\_sale\_price < 10000 then 1 end) "1만원미만",  count(case when final\_sale\_price >=10000 or final\_sale\_price < 30000 then 1 end) "3만원미만",  count(case when final\_sale\_price >=30000 or final\_sale\_price < 50000 then 1 end) "5만원미만",  count(case when final\_sale\_price >=50000 or final\_sale\_price < 100000 then 1 end) "10만원미만",  count(case when final\_sale\_price >=100000 then 1 end) "10만원이상"  from user\_info u, order\_list o  where u.user\_info\_no = o.user\_info\_no  group by f\_age(user\_jumin) having f\_age(user\_jumin) >= 20 and f\_age(user\_jumin) <= 70  order by 1 asc; |
|  |

|  |  |  |
| --- | --- | --- |
|  | 리포트 항목 | SQL 쿼리 |
| 11 | 카테고리 별  선호하는 구매 가격대 | select s\_c\_no "소분류",  count(case when o.final\_sale\_price < 10000 then 1 end) "1만원미만",  count(case when o.final\_sale\_price >=10000 or o.final\_sale\_price < 30000 then 1 end) "3만원미만",  count(case when o.final\_sale\_price >=30000 or o.final\_sale\_price < 50000 then 1 end) "5만원미만",  count(case when o.final\_sale\_price >=50000 or o.final\_sale\_price < 100000 then 1 end) "10만원미만",  count(case when o.final\_sale\_price >=100000 then 1 end) "10만원이상"  from order\_list o, sale\_list s, pro\_info p  where o.sale\_no = s.sale\_no and s.pro\_info\_no = p.pro\_info\_no  group by s\_c\_no  order by 1 asc; |
|  |

* 회원

|  |  |  |
| --- | --- | --- |
|  | 리포트 항목 | SQL 쿼리 |
| 1 | 기간 별  신규 및 재구매 현황 | select user\_info\_no 고객번호,s.pro\_info\_no 상품코드,(select pro\_name from pro\_info p where p.pro\_info\_no=s.pro\_info\_no ) 상품명,count(O\_QUANTITY) 수량  from order\_list o, sale\_list s  where o.sale\_no=s.sale\_no and o.o\_date between '2021-01-01' and '2021-03-31'  group by o.user\_info\_no,s.pro\_info\_no  having count(O\_QUANTITY) >=2; |
|  |
| 2 | 연령대 별  구매목록의 만족도 현황 | select f\_age(u.user\_jumin) 연령대,lpad('\*',avg(c.star),'\*') avg\_star 별점 from order\_list o, user\_info u, review c  where o.user\_info\_no=u.user\_info\_no and o.user\_info\_no=c.user\_info\_no and o.order\_no=c.order\_no  and o.o\_date between '2020-01-01' and '2021-08-10'  group by u.user\_jumin  order by avg\_star; |
| 3 | 최우수 고객 5명  (최대 구매 고객) | select user\_info\_no "최대구매고객" from user\_info where user\_info\_no in (select user\_info\_no from (select \*from order\_list order by final\_sale\_price desc) where rownum < 6); |
|  |
| 4 | 시간대 별  구매자 접속 현황 | select to\_char(o\_date,'HH') "구매시간" ,sum(final\_sale\_price) "시간대별 구매 총액" from order\_list  where o\_date between '2021-01-01' and '2021-01-15' group by to\_char(o\_date,'HH'); |

* 매출

|  |  |  |
| --- | --- | --- |
|  | 리포트 항목 | SQL 쿼리 |
| 1 | 특정기간, 업체별  매출 현황 | select s.pro\_info\_no 상품코드,(select p.pro\_name from pro\_info p where p.pro\_info\_no=s.pro\_info\_no)상품명, sum(o.o\_quantity) 총수량,sum(o.final\_sale\_price) 총매출 from order\_list o, sale\_list s where o.sale\_no = s.sale\_no and o.o\_date between '2021-01-01' and '2021-01-16' group by s.pro\_info\_no order by s.pro\_info\_no; |
|  |
| 2 | 특정 협력사 별  상품 별 매출 현황 | select m.partner\_id 협력사\_ID, s.pro\_info\_no 상품코드,(select p.pro\_name from pro\_info p where p.pro\_info\_no=s.pro\_info\_no)상품명, sum(o.final\_sale\_price)총매출 from order\_list o, sale\_list s, partner\_info m where o.sale\_no = s.sale\_no and m.partner\_id=15 group by s.pro\_info\_no, m.partner\_id order by 2 desc; |
|  |
|  | 리포트 항목 | SQL 쿼리 |
| 3 | 카테고리 별, 상품 별  매풀 현황 | select s1.s\_c\_no 카테고리, p.pro\_info\_no 상품코드,(select pro\_name from pro\_info pp where pp.pro\_info\_no=p.pro\_info\_no ),sum(o.o\_quantity) 총수량, sum(o.final\_sale\_price) 총매출  from pro\_info p, sale\_list s, order\_list o, sub\_category s1 where p.pro\_info\_no = s.pro\_info\_no and p.s\_c\_no = s1.s\_c\_no and s.sale\_no = o.sale\_no group by s1.s\_c\_no , p.pro\_info\_no order by p.pro\_info\_no desc; |
|  |
| 4 | 카테고리 별  일일 판매 현황 | select s1.s\_c\_no 카테고리, p.pro\_info\_no 상품코드, (select pro\_name from pro\_info pp where pp.pro\_info\_no=p.pro\_info\_no) 상품명,sum(o.o\_quantity) 총수량, sum(o.final\_sale\_price) 총매출  from pro\_info p ,sale\_list s, sub\_category s1,order\_list o  where s.pro\_info\_no = p.pro\_info\_no and s1.s\_c\_no = p.s\_c\_no and s.sale\_no = o.sale\_no and s.sale\_date between '2021-01-01' and '2021-01-01' group by s1.s\_c\_no ,p.pro\_info\_no order by sum(s.final\_sale\_price) desc; |
|  | 리포트 항목 | SQL 쿼리 |
| 4 | 카테고리 별  일일 매출 현황 |  |
| 5 | 특정 일  판매된 상품 별  평균 매출 | select p.pro\_info\_no 상품코드, (select pro\_name from pro\_info pp where pp.pro\_info\_no=p.pro\_info\_no) 상품명 ,avg(o.final\_sale\_price)평균  from pro\_info p, sale\_list s, order\_list o  where p.pro\_info\_no = s.pro\_info\_no  and s.sale\_no = o.sale\_no  and s.sale\_date between '2021-01-01' and '2021-01-01'  group by p.pro\_info\_no  order by avg(o.final\_sale\_price) desc; |
|  |

|  |  |  |
| --- | --- | --- |
|  | 리포트 항목 | SQL 쿼리 |
| 6 | 직전 년도 대비  카테고리별, 분기별  매출 증감율 현황 | with A as  (  select p.s\_c\_no as 올해 , to\_char (o.o\_date,'Q') as 분기 , sum(o.final\_sale\_price) 매출  from sale\_list s, pro\_info p, order\_list o  where s.pro\_info\_no = p.pro\_info\_no  and o.sale\_no = s.sale\_no  and o.o\_date between '2020-01-01' and '2020-12-31'  group by p.s\_c\_no, to\_char(o.o\_date,'Q')  ),  B as  (  select p.s\_c\_no as 작년 , to\_char (o.o\_date-365,'Q') as 분기, sum(o.final\_sale\_price) 매출 from sale\_list s, pro\_info p, order\_list o  where s.pro\_info\_no = p.pro\_info\_no  and o.sale\_no = s.sale\_no  and o.o\_date between '2021-01-01' and '2021-12-31'  group by p.s\_c\_no, to\_char(o.o\_date-365,'Q')  )  select A.올해,(select s\_c\_name from sub\_category s where s.s\_c\_no=A.올해) 분류명 ,A.분기,A.매출,B.작년, B.분기,B.매출 from A , B where A.올해 = B.작년; |
| 6 | 직전 년도 대비  카테고리별, 분기별  매출 증감율 현황 |  |
| 7 | 매출 기준  우수 협력사 순위 | select p1.partner\_ID 협력사,(select partner\_name from partner\_info pp where pp.partner\_id=p1.partner\_id) 파트너명 , sum(o.final\_sale\_price) 총매출  from partner\_use\_info p1, pro\_info p, sale\_list s ,order\_list o  where p1.partner\_ID = p.partner\_ID and p.pro\_info\_no = s.pro\_info\_no and s.sale\_no = o.sale\_no  group by p1.partner\_ID  order by sum(s.final\_sale\_price) desc; |
|  |

|  |  |  |
| --- | --- | --- |
|  | 리포트 항목 | SQL 쿼리 |
| 8 | 할인률 별  상품 판매 현황 | select d.DEAL\_no 딜타입,count(\*) 발생건수 ,(select deal\_discount from deal\_list dd where dd.deal\_no=d.deal\_no) 할인률  from deal\_list d, order\_list o, sale\_list s  where d.deal\_no = s.deal\_no  and s.sale\_no = o.sale\_no  group by d.DEAL\_no; |
|  |

* Function

--------------------------------------------------------------------------------

연령대 구하는 함수

--------------------------------------------------------------------------------

create or replace function f\_age(user\_jumin varchar2)

return number

is

v\_temp number(1);

v\_year number(4);

v\_age number(3);

begin

v\_temp := substr(user\_jumin, 8, 1);

v\_year := to\_char(sysdate,'yyyy');

if v\_temp = 1 or v\_temp = 2 then

v\_age := trunc((v\_year - (1900 + substr(user\_jumin, 1, 2))),-1);

return v\_age;

else

v\_age := trunc((v\_year - (2000 + substr(user\_jumin, 1, 2))),-1);

return v\_age;

end if;

end;

/

--------------------------------------------------------------------------------

기간 중 토일을 제외한 배송일 산출

--------------------------------------------------------------------------------

create or replace function f\_get\_weekdays(v\_sale\_date date,v\_day\_count number)

return date

is

v\_weekend number(3);

v\_return date;

begin

SELECT count(d\_day) into v\_weekend FROM (

SELECT (v\_sale\_date) + LEVEL as d\_date,to\_char(((v\_sale\_date) + LEVEL), 'd') AS d\_day FROM DUAL

CONNECT BY LEVEL <= ((v\_sale\_date + v\_day\_count) - v\_sale\_date)

) B WHERE B.d\_day in (1,7) ;

v\_return:= v\_sale\_date + (v\_day\_count + v\_weekend);

return (v\_return);

end;

/

* Sequence

--------------------------------------------------------------------------------

주문 정보 시퀀스

--------------------------------------------------------------------------------

create Sequence SEQ\_Order\_List increment by 1 start with 1 maxvalue 1000000;

* Procedure

--------------------------------------------------------------------------------

주문 정보 추가 / 업데이트

--------------------------------------------------------------------------------

CREATE OR REPLACE Procedure P\_Order\_List\_In\_Up(

V\_ORDER\_NO ORDER\_LIST.ORDER\_NO%type,

V\_O\_RECIEVER\_NAME ORDER\_LIST.O\_RECIEVER\_NAME%type,

V\_O\_RECIEVER\_ADDR ORDER\_LIST.O\_RECIEVER\_ADDR%type,

V\_USER\_INFO\_NO ORDER\_LIST.USER\_INFO\_NO%type,

V\_O\_QUANTITY ORDER\_LIST.O\_QUANTITY%type,

V\_FINAL\_SALE\_PRICE ORDER\_LIST.FINAL\_SALE\_PRICE%type,

V\_SALE\_NO ORDER\_LIST.SALE\_NO%type,

V\_O\_RECIEVER\_TEL ORDER\_LIST.O\_RECIEVER\_TEL%type,

V\_O\_DATE ORDER\_LIST.O\_DATE%type

)

is

begin

UPDATE ORDER\_LIST SET O\_RECIEVER\_NAME =V\_O\_RECIEVER\_NAME,

O\_RECIEVER\_ADDR=V\_O\_RECIEVER\_ADDR,

USER\_INFO\_NO=V\_USER\_INFO\_NO,

O\_QUANTITY=V\_O\_QUANTITY,

FINAL\_SALE\_PRICE=V\_FINAL\_SALE\_PRICE,

SALE\_NO=V\_SALE\_NO,

O\_RECIEVER\_TEL=V\_O\_RECIEVER\_TEL,

O\_DATE=V\_O\_DATE

WHERE ORDER\_NO=V\_ORDER\_NO;

IF sql%rowcount=0 THEN

INSERT INTO ORDER\_LIST(ORDER\_NO, O\_RECIEVER\_NAME, O\_RECIEVER\_ADDR, USER\_INFO\_NO, O\_QUANTITY, FINAL\_SALE\_PRICE, SALE\_NO, O\_RECIEVER\_TEL, O\_DATE)

VALUES(SEQ\_Order\_List.nextval, V\_O\_RECIEVER\_NAME, V\_O\_RECIEVER\_ADDR, V\_USER\_INFO\_NO, V\_O\_QUANTITY, V\_FINAL\_SALE\_PRICE, V\_SALE\_NO, V\_O\_RECIEVER\_TEL, V\_O\_DATE);

END IF;

END;

/

* Trigger

--------------------------------------------------------------------------------

테이블 백업

--------------------------------------------------------------------------------

create or replace trigger tri\_order\_list\_b

after insert or delete or update on order\_list

for each row

begin

if deleting then

delete order\_list\_b where order\_no=:old.order\_no;

else

update order\_list\_b set O\_RECIEVER\_NAME=:new.O\_RECIEVER\_NAME,

O\_RECIEVER\_ADDR=:new.O\_RECIEVER\_ADDR,

USER\_INFO\_NO=:new.USER\_INFO\_NO,

O\_QUANTITY=:new.O\_QUANTITY,

FINAL\_SALE\_PRICE=:new.FINAL\_SALE\_PRICE,

SALE\_NO=:new.SALE\_NO,

O\_RECIEVER\_TEL=:new.O\_RECIEVER\_TEL,

O\_DATE=:new.O\_DATE

where order\_no=:new.order\_no;

if sql%rowcount=0 then

insert into order\_list\_b(order\_no,O\_RECIEVER\_NAME,O\_RECIEVER\_ADDR,USER\_INFO\_NO,O\_QUANTITY,FINAL\_SALE\_PRICE,SALE\_NO,O\_RECIEVER\_TEL,O\_DATE)

values (:new.order\_no,:new.O\_RECIEVER\_NAME,:new.O\_RECIEVER\_ADDR,:new.USER\_INFO\_NO,:new.O\_QUANTITY,:new.FINAL\_SALE\_PRICE,:new.SALE\_NO,:new.O\_RECIEVER\_TEL,:new.O\_DATE);

end if;

end if;

end;

/

--------------------------------------------------------------------------------

협력사 업무 중지 시 해당되는 판매 상품에 대한 판매 중지 변경

--------------------------------------------------------------------------------

create or replace trigger tri\_partner\_use\_info\_p\_use

after insert or delete or Update of P\_USE\_BOOL on partner\_use\_info

for each row

declare

begin

if :new.P\_USE\_BOOL = '0' then

update sale\_list set sale\_bool='0' where pro\_info\_no in (select pro\_info\_no from pro\_info where partner\_id=:new.partner\_id)

end if;

end;

/

--------------------------------------------------------------------------------

판매 시작 직후 판매 마감일자 자동 업데이트

--------------------------------------------------------------------------------

select to\_char(to\_date('2021-01-01') + level - 1,'YYYY-MM-DD'),to\_char(to\_date('2021-01-31') + level-1, 'D' ) D from dual

--------------------------------------------------------------------------------

회원의 접속 정보 백업

--------------------------------------------------------------------------------

create or replace trigger tri\_user\_info\_last\_conn

after insert or delete or Update of last\_conn on user\_info

for each row

declare

begin

INSERT INTO USER\_Service(User\_Service\_No, Sign\_date, Pass\_Ch\_Date, Last\_Conn, Conn\_num, Order\_Num, User\_Info\_No,Resign\_date,Resign\_bool)

VALUES (SEQ\_UserService.nextval,sysdate,sysdate,:new.last\_conn,0,0,:new.user\_info\_no,sysdate,1);

end;

/

--------------------------------------------------------------------------------

적용 딜 % 정보 변경 / 이전 주문의 딜 적용가 변경

--------------------------------------------------------------------------------

create or replace trigger tri\_order\_info\_deal\_count

after insert or delete or Update of O\_QUANTITY on order\_list

for each row

declare

v\_people deal\_list.people\_num%type;

v\_sale\_bool sale\_list.sale\_bool%type;

v\_deal\_discount deal\_list.deal\_discount%type;

begin

select sale\_bool into v\_sale\_bool from sale\_list where sale\_no=:old.sale\_no;

if v\_sale\_bool='1' then

select count(\*) into v\_people from order\_list where sale\_no=:old.sale\_no;

UPDATE SALE\_LIST SET deal\_no=(SELECT min(deal\_no) FROM deal\_list where people\_num >= v\_people )

WHERE sale\_no=:old.sale\_no;

select deal\_discount into v\_deal\_discount from deal\_list where deal\_no= (SELECT min(deal\_no) FROM deal\_list where people\_num >= v\_people );

update order\_list set final\_sale\_price=(final\_sale\_price \* v\_deal\_discount) / 100

where sale\_no=:old.sale\_no;

end if;

end;

/