1. 아래와 같이 Table 생성,Data를 입력한 후 질문에 대한 SQL을 최소 각각 3개씩 작성하시요!!!!

drop table cust\_status;  
create table cust\_status  
 (cust\_id      char(1)      not null,  
  cust\_id\_seq  number       not null,  
  status       varchar2(10) not null)  
 ;

insert into cust\_status values ('A',1,'정상');  
insert into cust\_status values ('A',2,'위험');  
insert into cust\_status values ('B',1,'정상');  
insert into cust\_status values ('B',2,'정상');  
insert into cust\_status values ('C',1,'위험');  
insert into cust\_status values ('C',2,'위험');  
insert into cust\_status values ('D',1,'위험');  
insert into cust\_status values ('D',2,'위험');  
insert into cust\_status values ('D',3,'정상');  
insert into cust\_status values ('E',1,'정상');  
commit;

Q1>  cust\_id 별로 status 값이  한 종류만 가진  cust\_id 만 출력

* Count,max,min,avg,sum 등 Aggregate Function을 사용.

**1. 접근 방식 : CUST\_ID에 대한 STATUS의 개수를 구해서 그 개수가 1개(같은 STATUS)인 컬럼 조회 ( 키포인트 : CUST\_ID에 대한 STATUS )**

select cust\_id

from

(

select cust\_id,count(distinct status) status\_count

from cust\_status

group by cust\_id

having count(distinct status)=1

);

**2. 접근 방식 : CUST\_ID에 대한 MAX STATUS와 MIN STATUS를 비교해서 같은 값(값이 같으면 하나의 STATUS)의 컬럼 조회 ( 키포인트 : CUST\_ID에 대한 STATUS )**

select cust\_id

from

(

select cust\_id,max(status) max\_status,min(status) min\_status

from cust\_status

group by cust\_id

)

where max\_status = min\_status;

**3. 접근 방식 : CUST\_ID에 대한 STATUS가 ‘정상’ , ‘위험’ 일 경우 1씩 누적 합을 해서**

**SUM\_1 이나 SUM\_2 값이 NULL( STATUS가 다를 경우 각각의 컬럼에 1, 1 이 누적 / 같으면 한쪽 컬럼만 2로 누적 다른쪽은 NULL) 인 컬럼 조회 ( 키포인트 : CUST\_ID에 대한 STATUS )**

select cust\_id

from

(

select cust\_id,

sum(decode(status,'정상',1)) sum\_1,

sum(decode(status,'위험',1)) sum\_2

from cust\_status

group by cust\_id

)

where sum\_1 is null or sum\_2 is null;

**4. 접근 방식 : CUST\_ID에 대한 STATUS를 임의의 한 값으로 정의하고 그 값이 맞을 경우와 다를 경우의 값을 부여한 뒤 AVG로 평균을 구하고 그 평균 값을 부여한 값으로 나눴을 때 나머지가 0인 컬럼 조회 ( 키포인트 : CUST\_ID에 대한 STATUS )**

select cust\_id

from

(

select cust\_id,

avg(decode(status,'정상',2,0)) avg\_status

from cust\_status

group by cust\_id

)

where mod(avg\_status,2) = 0;

Q2> ﻿ cust\_id 별로 status 값이  한 종류만 가진  Row 전체를 출력

* Subquery, In-line View, Scalar Subqury , Outer Join 사용.

**1. 접근 방식 : CUST\_ID에 대한 STATUS가 중복(같은 값) 되면 COUNT가 1이 되고 그 1값을 가지는 컬럼 조회**

select \*

from cust\_status a

where 1 = (select count(distinct status)

from cust\_status

where cust\_id = a.cust\_id);

**2. 접근 방식 : 두 테이블에서 CUST\_ID가 같은 값으로 데이터를 추리고 그 데이터의 STATUS의 값이 같지 않은 컬럼을 조회하고 그 컬럼을 제외한 컬럼 조회**

select \*

from cust\_status a

where not exists (select 1 from cust\_status

where cust\_id = a.cust\_id

and status != a.status);

**3. 접근 방식 : CUST\_ID에 대한 STATUS 값의 수를 구하고 두 테이블의 CUST\_ID가 같은 컬럼에서 COUNT\_STATUS의 수가 1인 컬럼을 조회**

select a.\*

from cust\_status a,

(select cust\_id,count(distinct status) count\_status

from cust\_status

group by cust\_id) b

where a.cust\_id = b.cust\_id

and count\_status = 1;

**4. 접근 방식 : CUST\_ID 별 해당 STATUS의 수를 구하여 컬럼으로 지정하고 그 값이 0인 컬럼을 조회**

select cust\_id,cust\_id\_seq,status

from

(

select a.\*,

(select count(\*) from cust\_status

where cust\_id=a.cust\_id

and status != a.status) count\_status

from cust\_status a

)

where count\_status = 0;

**5. 접근 방식 : 두 테이블을 아우터 조인 하고 조건에 만족하지 않는 컬럼은 아우터 조인 테이블의 컬럼에 NULL로 표시되기 때문에 컬럼이 NULL인 것을 조회**

select a.\*

from cust\_status a,

cust\_status b

where a.cust\_id = b.cust\_id(+)

and a.status != b.status(+)

and b.cust\_id is null;

2. 아래와 같은 Table을 생성 ,Data를 입력한 후 20190101~20191231까지 총 365건 이 조회되도록 일자별집계를 구하시요.

TABLE명 : REPAY\_TEST

**1) 상환 후 잔액 거래 내역**

|  |  |  |  |
| --- | --- | --- | --- |
| **REPAY\_DATE** | **DETR\_NM** | **RBNO** | **LOAN\_BAL\_AMT** |
| 20190103 | 홍길동 | 1234567-1234567 | 1500000 |
| 20190906 | 홍길동 | 1234567-1234567 | 1000000 |
| 20190909 | 홍길동 | 1234567-1234567 | 500000 |

**2) 상환 후 잔액 일별 집계**

|  |  |  |  |
| --- | --- | --- | --- |
| **TOT\_DATE** | **DETR\_NM** | **RBNO** | **LOAN\_BAL\_AMT** |
| 20190101 | 홍길동 | 1234567-1234567 | 0 |
| 20190102 | 홍길동 | 1234567-1234567 | 0 |
| 20190103 | 홍길동 | 1234567-1234567 | 1500000 |
| 20190104 | 홍길동 | 1234567-1234567 | 1500000 |
| 20190105 | 홍길동 | 1234567-1234567 | 1500000 |
| 20190106 | 홍길동 | 1234567-1234567 | 1500000 |
| 상동 | 상동 | 상동 | 상동 |
| 20190905 | 홍길동 | 1234567-1234567 | 1500000 |
| 20190906 | 홍길동 | 1234567-1234567 | 1000000 |
| 20190907 | 홍길동 | 1234567-1234567 | 1000000 |
| 20190908 | 홍길동 | 1234567-1234567 | 1000000 |
| 20190909 | 홍길동 | 1234567-1234567 | 500000 |
| 20190910 | 홍길동 | 1234567-1234567 | 500000 |
| 20190911 | 홍길동 | 1234567-1234567 | 500000 |
| 상동 | 상동 | 상동 | 상동 |

drop table repay\_test;

create table repay\_test

(

repay\_date varchar2(8) not null,

detr\_nm varchar2(100) not null,

rbno varchar2(20) not null ,

loan\_bal\_amt number not null

);

insert into repay\_test values ('20190103','홍길동','1234567-1234567',1500000);

insert into repay\_test values ('20190906','홍길동','1234567-1234567',1000000);

insert into repay\_test values ('20190909','홍길동','1234567-1234567', 500000);

commit;

select a.repay\_date,a.detr\_nm,a.rbno,nvl(loan\_bal\_amt,0) loan\_bal\_amt

from

(

select detr\_nm,rbno,repay\_date

from

(

select distinct detr\_nm,rbno

from repay\_test

) a,

(

select to\_char(to\_date(:from\_date) + level -1,'yyyymmdd') repay\_date

from dual connect by level <= (select to\_date(:to\_date) - to\_date(:from\_date) +1 from dual )

) b

) a,

(

select repay\_date from\_repay\_date,

nvl((lead(repay\_date, 1) over (partition by rbno order by repay\_date))-1, to\_char(sysdate,'yyyymmdd')) to\_repay\_date ,

detr\_nm,rbno,loan\_bal\_amt

from repay\_test

) b

where a.rbno = b.rbno(+)

and a.repay\_date between b.from\_repay\_date(+) and b.to\_repay\_date(+)

order by a.detr\_nm,a.repay\_date;