데이터베이스

오라클 환경 설치 및 접속, EXERD, DBEAVER 설치, SQL문

EXERD - ERD를 생성하는 툴

** 포워드 엔지리어닝을 통해 CREATE문을 생성해 낼 수 있다. 대상 DB 오라클로 할 것.

** CTRL 키를 누르고 컬럼을 옮기면 다른 테이블로 컬럼이 옮겨진다

** ctrl 키 + 엔터 필드 추가

** 관계에서 스페이스 누르면 관계 나옴

** 포워드 엔지리어닝

이름 앞에 스키마 표시X

테이블 스페이스 생성 X

뷰 생성 X

트리거 생성 X

테이블 새성 O

물리적 특성들 X

코멘트 - 논리이름

DBEAVER - DB관리 툴

** 툴을 사용할 때 AUTO-COMMIT이 아닌 로컬로 돌리거나 FLASE로 해야 한다. AUTO COMMIT은 커밋을 해버리면 ROLLBACK이 안된다.

주요테이블

테이블명	상세
TSU0204	라벨 관리를 위한 APP_CD, APP_NM 등이 존재 `ex ) SELECT * FROM TSU0204 WHERE APP_CD = 'AN-01-00-003'; 인터페이스 상세 라벨 조회`
TSU0214	프로그램 메뉴 맵핑
TSU0101	사용자 테이블
TAN0201	인터페이스 중심 테이블 상태정보와 업무id를 갖고 있다 tim0301 테이블과 조인해야 업무명을 정확하게 가져올 수 있음.
TAN0101	인터페이스 요건 테이블(TAN0201의 인터페이스 아이디와 FK)
TAN0213	인터페이스 시스템 맵핑 SEQ INTERFACEID
TSU0301	코드명 테이블(한글). 공통코드, 화면에서 쓰는 모든 한글명 송신시스템 수신시스템 담당자 등을 전부 코드화 해 놓은 곳(LEVEL1 : 대분류, LEVEL2 : 소분류, CD : 개발방식, DEL-YN : 삭제 여부)
TIM0301	업무 테이블(BUSINESS). select박스라고 할 수 있다 업무명이 여기에 다 등록되어있고 인터페이스 요건과 연결되어 있음.
TIM0101	시스템 테이블
TAN0213	인터페이스 시스템 맵핑

INTERFACE_ID : 인터페이스 아이디 기본키이며 식별자이다.

INTERFACE_NM - 인터페이스 명(화면에서는 한글로 보인다)

INTEGRATION_ID - 화면에서 보이는 고객이 요청한 아이디

** 고객사 DB를 생성할 때 DB는 모두 같은 DB를 쓴다. 고객사에 맞게 부분을 바꿔주면 된다. CREATE문과 INSERT 문은 기본 베이스 DB와 같다.

SQL

https://www.w3schools.com/sql/sql_join.asp

TO_CHAR

date 타입을 문자열로 변환하는 오라클 문법 (”숫자OR날짜”,”포맷형식”)

날짜 형식으로 지정해 놓으면 변환이 필요하기 때문에 to_ date 시간을 to_char로 문자열 변환

```
SELECT
*
FROM TAN0201
WHERE REG_DATe <= TO_CHAR(sysdate,'yyyymmddhh24miss' ) || 000;
```

```
select to_char(current_timestamp(3),'yyyymmddhh24missff') from dual;
```

SUBSTR

문자열을 자르는 함수

```
SELECT
*
FROM TAN0201
WHERE sysdate >= to_date(SUBSTR (REG_DATE ,1,14),'yyyymmddhh24miss');
```

TAN0201 테이블에서 SELECT할건데 REG_dATE의 1부터 14까지 잘라서 DATE 형식으로 보여준다.

SYSDATE

날짜 조작 함수 현재 날짜와 시간을 가져옴.

emp 테이블 reg_date 컬럼에 현재 날짜를 문자 형식으로 넣기

```
INSERT INTO emp(REG_DATE) values (to_char(sysdate,'yyyymmdd'));
SELECT * FROM EMP;
```

CREATE

```
CREATE TABLE EMP2(
EMP_ID VARCHAR(50) NOT NULL, /* 사원ID */
DEPT_ID VARCHAR(50) NOT NULL, /* 부서ID */
ENP_NM VARCHAR(50) /* 부서명 */
);
```

//테이블 생성 DBEAVER를 통해 테이블 생성문을 가져옴

DELETE

```
DELETE TABLE 테이블명;
```

안에 있는 데이터만 삭제됨 DROP은 테이블 전체를 날리는 것임.

UPDATE

업데이트 문을 쓸 때는 필수로 WHERE 조건을 넣어야 함 아니면 고객 정보가 모두 삭제될 수도 있기 때문에 주의할 것.

```
UPDATE  EMP  SET  REG_DATE = TO_CHAR(SYSDATE,'yyyymmddHH24MISS');
```

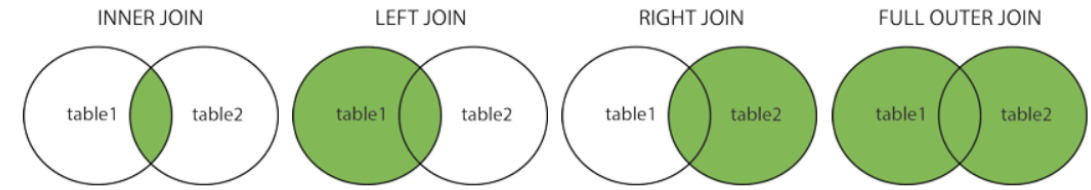
//emp 테이블 reg_date 값 업데이트

INSERT

```
INSERT INTO EMP2(EMP_ID,ENP_NM,DEPT_ID)  VALUES ('uhaha','uhaha','2');
```

//EMP2 테이블에 값과 컬럼 맵핑하여 데이터 넣기

JOIN 문



INNER JOIN과 OUTER JOIN

INNER JOIN : 두 테이블 모두 일치하는 값

LEFT OUTER JOIN : 왼쪽 테이블의 모든 레코드를 반환하고 오른쪽 테이블의 일치하는 값

RIGHT OUTER JOIN : 오른쪽 테이블의 모든 레코드를 반환하고 왼쪽 테이블의 일치하는 값

FULL : 왼쪽 테이블과 오른쪽 테이블의 일치하는 모든 레코드 반환

```
SELECT
a.INTEGRATION_ID
,a.INTERFACE_NM
,b.STATUS
,b.BUSINESS_ID
,b01.BUSINESS_NM  AS "업무명"
,c.SYSTEM_ID
,c01.SYSTEM_NM    AS "송신시스템명"
,d.SYSTEM_ID
,d01.SYSTEM_NM    AS "수신시스템명"
,a.DATA_PR_DIR
,a.APP_PR_METHOD
,a.DATA_PR_DIR
,a.REG_USER
,a01.USER_NM    AS "등록자명"
,a.REG_DATE
,a.MOD_USER
,a02.USER_NM    AS "수정자명"
```

```
,a.MOD_DATE
FROM TAN0201 a
INNER JOIN TAN0101 b
ON a.INTERFACE_ID = b.INTERFACE_ID
INNER JOIN TIM0301 b01
ON b01.BUSINESS_ID = b.BUSINESS_ID
INNER JOIN TAN0213 c
ON a.INTERFACE_ID = c.INTERFACE_ID
AND c.NODE_TYPE = '0'
AND c.SEQ = (SELECT max(SEQ) FROM TAN0213 WHERE INTERFACE_ID = c.INTERFACE_ID AND NODE_TYPE = '0')
INNER JOIN  TIM0101 c01
ON c01.SYSTEM_ID = c.SYSTEM_ID
INNER JOIN TAN0213 d
ON a.INTERFACE_ID = d.INTERFACE_ID
AND d.NODE_TYPE = '2'
AND d.SEQ = (SELECT max(SEQ) FROM TAN0213 WHERE INTERFACE_ID = d.INTERFACE_ID AND NODE_TYPE = '2')
INNER JOIN TIM0101 d01
ON d01.SYSTEM_ID = d.SYSTEM_ID
INNER JOIN tsu0101 a01
ON a01.USER_ID = a.REG_USER
LEFT OUTER JOIN tsu0101 a02
ON a02.USER_ID = a.MOD_USER;
```

한글 NM 가져오기

```
SELECT
a.interface_nm,
a.interface_id,
a.app_pr_method,
a.data_pr_method,
a.data_pr_dir,
cd1.NM AS APP_PR_METHOD_NM, -- 앱처리방식 nm
cd2.NM  AS APP_PR_METHOD, --데이터처리방식 nm
cd3.nm AS data_pr_dir -- 데이터처리방식
frOM tan0201 a
LEFT OUTER JOIN tsu0301 cd1 ON a.APP_PR_METHOD  = cd1.CD AND cd1.LEVEL1  ='IM' AND cd1.LEVEL2= '02' -- 앱처리방식
LEFT OUTER JOIN TSU0301 cd2 ON a.DATA_PR_METHOD = cd2.CD AND cd1.LEVEL1  = 'IM' AND cd2.LEVEL2 = '08' -- 데이터처리방식
LEFT OUTER JOIN TSU0301 cd3 ON a.DATA_PR_DIR = cd3.cd AND cd1.LEVEL1  ='IM' AND CD3.LEVEL2 = '01' --  데이터처리방향
wHERE a.DEL_YN = 'N' – 삭제여부가 No 이고
AND a.REG_USER ='iip' –reg_user가 iip이면서
AND a.INTEGRATION_ID LIKE 'U%' – 인티그레이션 아이디가 u로 시작하고
AND a.APP_PR_METHOD  = '0' – pr 메소드가 0인것
ORDER BY a.INTERFACE_ID ; – 오름차순으로 정렬해서 보여줘라
```

LIKE

‘a%’ - a로 시작하는 것

‘%a’ - a로 끝나는 것

‘%a%’ -모든 위치에 a가 있는 것

'_a%’ - 두번째 위치에 a가 있는 것

'a__%' - a로 시작하고 길이가 3자 이상인 것

'a%o - a로 시작하고 o로 끝나는 것

not like ‘a%’ a로 시작하지 않는 것

MAX 함수

```
SELECT *
FROM tan0201
WHERE
reg_date = (SELECT max(REG_DATE) FROM tan0201 );
```

`SELECT max (DATA_PR_METHOD), min (DATA_PR_METHOD) FROM tan0201;`

등록일자 MAX, MIN

```
SELECT max(reg_date), min(reg_date) FROM tan0201;
```

MAX 가장 최근 날짜

MIN 가장 오래된 날짜

2023년 이후로 등록된 일자별 인터페이스 건수

```
SELECT
SUBSTR(REG_DATE,1,8),
COUNT(interface_id),
MAX(reg_user), –등록한 사람중에 제일 알파벳순이 먼저인 사람
mod_user
FROM tan0201
WHERE REG_DATE >= '20220101' || '00120000'
GROUP BY SUBSTR(reg_date,1,8), mod_user --일자별 GROUB BY 절에 들어간건 위에 나올 수 있음
ORDER BY 1 DESC;
```

COUNT

지정한 행의 갯수를 반환

`SELECT COUNT (REG_DATE) FROM tan0201;`

REG_DATE의 수를 세서 반환해줌

```
SELECT count(*)
FROM tan0201 a
INNER JOIN tan0219 b
ON a.INTERFACE_ID  = b.INTERFACE_ID ;
```

tan0219와 tan0201의 인터페이스 아이디가 같은 것을 세어 반환해줌

UNION

연관 없는 두개의 테이블을 하나로 결합해주는 쿼리문

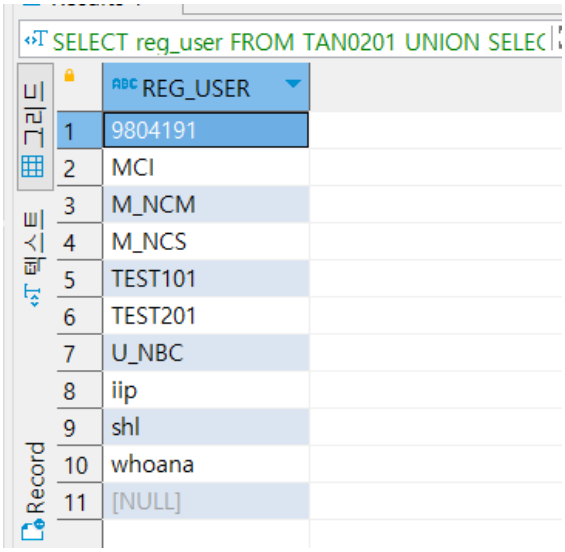
SELECT 컬럼의 갯수와 데이터 타입이 맞아야 결합이 가능함

UNION만 쓸 경우 중복을 걸러주고, UNION ALL을 쓸 경우 중복도 모두 포함함.

```
SELECT
reg_user
```

```
FROM TAN0201
UNION
SELECT MOD_USER
FROM tsu0201;
```

데이터 타입이 일치하여 조회가 가능함.




```
SELECT INTERFACE_ID FROM tan0201 WHERE INTERFACE_ID = 'F@00000483'
UNION
SELECT INTERFACE_ID FROM tan0201 WHERE INTERFACE_ID  IN ('F@00000487','F@00000508');
```

서브쿼리

```
SELECT
         a.INTEGRATION_ID
        ,a.INTERFACE_NM
        ,b.STATUS
        ,b.BUSINESS_ID
        ,b01.BUSINESS_NM    AS "업무명"
        ,c.SYSTEM_ID
        ,c01.SYSTEM_NM      AS "송신시스템명"
        ,a.DATA_PR_DIR
        ,a.APP_PR_METHOD
        ,a.DATA_PR_DIR
        ,a.REG_USER
FROM tan0201 a
INNER JOIN tan0101 b
        ON a.INTERFACE_ID  = b.INTERFACE_ID
INNER JOIN tim0301 b01
        ON b.BUSINESS_ID  = b01.BUSINESS_ID
INNER JOIN tan0213 c
        ON a.INTERFACE_ID = c.INTERFACE_ID
        AND c.NODE_TYPE =0
        AND c.SEQ = (SELECT max(SEQ) FROM TAN0213 WHERE INTERFACE_ID = c.INTERFACE_ID AND NODE_TYPE = '0')
INNER JOIN tim0101 c01
        ON c.SYSTEM_ID =c01.SYSTEM_ID;
```

SELECT문 안 또 다른 SELECT문을 이용하여 중첩하는 것.

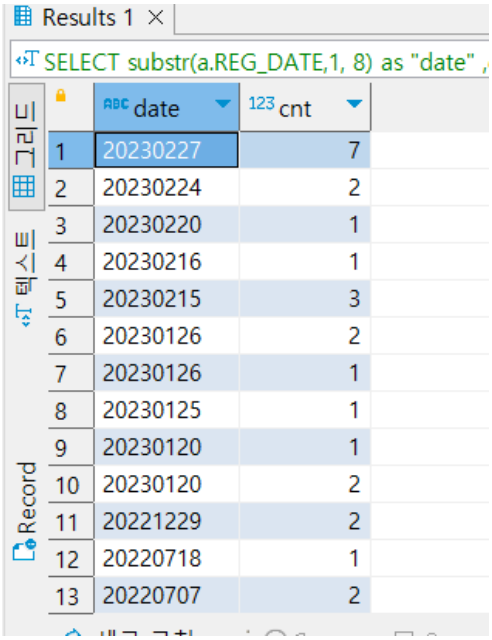
메인쿼리(바깥쪽 SELECT) 안에서 서브쿼리 컬럼을 사용할 수 있으며 서브쿼리에서는 ORDER BY가 불가능하다. GROUP BY는 가능하다.

GROUP BY

특정 열의 값들을 기준으로 그룹화하여 집계함수를 사용한 결과를 반환한다.

** SELECT 절에서 GROUP BY에 사용된 열과 집계함수로(AVG,SUM과 같은) 사용되지 않은 열은 사용이 불가능하다.

```
SELECT
   substr(a.REG_DATE,1, 8)    as "date"
  ,count(a.INTERFACE_ID)      as "cnt"
  FROM tan0201 a
 WHERE a.REG_DATE >= '20220101' || '000000000'
GROUP BY substr(a.REG_DATE,1, 8) , a.MOD_USER
ORDER BY 1 desc;
```



	date	cnt
1	20230227	7
2	20230224	2
3	20230220	1
4	20230216	1
5	20230215	3
6	20230126	2
7	20230126	1
8	20230125	1
9	20230120	1
10	20230120	2
11	20221229	2
12	20220718	1
13	20220707	2

DECODED

IF ELSE 기능을 수행함

`DECODE(YN , 'Y',  'YES, 'N', 'NO', '기타');`

만약 YN이 Y면 YES를 리턴하고 N 이면 NO를 리턴하고 둘 다 아니면 기타를 리턴한다는 의미임.

인터페이스의 상세 조회

```
SELECT
a.INTEGRATION_ID  AS "인터페이스ID"
,a.INTERFACE_NM   AS "인터페이스명"
,cd0.NM       AS "상태명"
,b01.BUSINESS_NM  AS "업무명"
,c01.SYSTEM_NM    AS "송신시스템명"
,d01.SYSTEM_NM    AS "수신시스템명"
,cd1.NM       AS "APP처리방식"
,cd2.NM       AS "데이터처리방식"
,cd3.NM       AS "데이터처리방향"
,a01.USER_NM    AS "등록자명"
,a.REG_DATE     AS "등록일"
,a02.USER_NM    AS "수정자명"
,a.MOD_DATE     AS "수정일"
```

```
,a04.USER_NM || decode(a03.ROLE_TYPE, '0', '(송신담당자)', '2', '(수신담당자)', '(허브담당자') AS "담당자"
FROM TAN0201 a
INNER JOIN TAN0101 b
ON a.INTERFACE_ID = b.INTERFACE_ID
AND a.INTERFACE_ID = 'F@00000421'
INNER JOIN TIM0301 b01
ON b01.BUSINESS_ID = b.BUSINESS_ID
INNER JOIN TAN0213 c
ON a.INTERFACE_ID = c.INTERFACE_ID
AND c.NODE_TYPE = '0'
AND c.SEQ = (SELECT max(SEQ) FROM TAN0213 WHERE INTERFACE_ID = c.INTERFACE_ID AND NODE_TYPE = '0')
INNER JOIN  TIM0101 c01
ON c01.SYSTEM_ID = c.SYSTEM_ID
INNER JOIN TAN0213 d
ON a.INTERFACE_ID = d.INTERFACE_ID
AND d.NODE_TYPE = '2'
AND d.SEQ = (SELECT max(SEQ) FROM TAN0213 WHERE INTERFACE_ID = d.INTERFACE_ID AND NODE_TYPE = '2')
INNER JOIN TIM0101 d01 ON d01.SYSTEM_ID = d.SYSTEM_ID
INNER JOIN TSU0101 a01 ON a01.USER_ID = a.REG_USER
LEFT OUTER JOIN TSU0101 a02 ON a02.USER_ID = a.MOD_USER
LEFT OUTER JOIN TSU0301 cd0 ON b.STATUS         = cd0.CD AND cd0.LEVEL1 = 'AN' AND cd0.LEVEL2 = '01'
LEFT OUTER JOIN TSU0301 cd1 ON a.APP_PR_METHOD  = cd1.CD AND cd1.LEVEL1 = 'IM' AND cd1.LEVEL2 = '02'
LEFT OUTER JOIN TSU0301 cd2 ON a.DATA_PR_METHOD = cd2.CD AND cd2.LEVEL1 = 'IM' AND cd2.LEVEL2 = '12'
LEFT OUTER JOIN TSU0301 cd3 ON a.DATA_PR_DIR    = cd3.CD AND cd3.LEVEL1 = 'IM' AND cd3.LEVEL2 = '01'
LEFT OUTER JOIN TAN0219 a03 ON a.INTERFACE_ID   = a03.INTERFACE_ID AND a03.DEL_YN  = 'N'
LEFT OUTER JOIN TSU0101 a04 ON a03.USER_ID      = a04.USER_ID
;
```

인터페이스ID	인터페이스명	상태명	업무명	송신시스템명	수신시스템명	APP처리방식	데이터처리방식	데이터처리방향	등록자명
CUINCSO00001	테스트0001	등록	고객(처리계)	처리계UI	고객(처리계)	동기	Online	양방향	포탈관리자
CUINCSO00001	테스트0001	등록	고객(처리계)	처리계UI	고객(처리계)	동기	Online	양방향	포탈관리자

DOCKER와 DB

1. docker pull jaspeen/oracle-xe-11g

   오라클 도커 이미지 받기

2. docker run --name oracle11g -d -p 8080:8089 -p 1521:1521 jaspeen/oracle-xe-11g

   컨테이너 실행하기

3. docker exec -it oracle11g sqlplus
   (id/password : system/oralce)

   SQLPLUS 실행

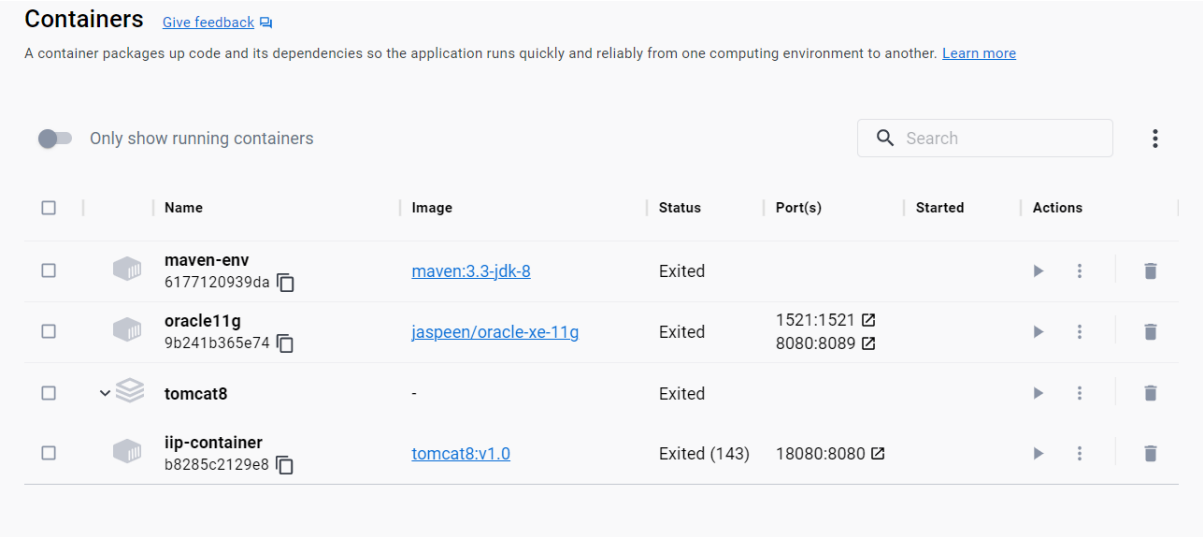
```
C:₩Users₩xxrin>docker exec -it oracle11g sqlplus

SQL*Plus: Release 11.2.0.2.0 Production on Mon Feb 27 13:51:14 2023

Copyright (c) 1982, 2011, Oracle.  All rights reserved.

Enter user-name: system
Enter password:

Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production

SQL> select instance_name, version, status from v$instance;

INSTANCE_NAME    VERSION           STATUS
---------------- ----------------- ------------
XE               11.2.0.2.0        OPEN

SQL>
```

4. docker exec -it oracle11g bash

   컨테이너 bash 접속(내부에 접속에 bash 쉘을 실행하는 것. 즉 컨테이너 내부에서 직접 작업을 할 수 있다. 애플리케이션 실행 로그 확인 등). 여기서 -it 옵션은 쉘을 실행하기 위한 옵션이다.

   `docker exec -it <container_id/container_name> bash`

```
C:₩Users₩xxrin>docker exec -it oracle11g bash
root@9b241b365e74:/#
```



DOCKER에 대해서

컨테이너화 된 애플리케이션을 쉽게 개발 및 배포 실행할 수 있도록 도와주는 오픈소스 컨테이너 플랫폼이다. 도커를 사용하면 개발자는 운영체제나 하드웨어 등의 환경에 독립적인 컨테이너 이미지를 만들어 놓고 이를 배포하면서 각기 다

른 환경에서 실행이 가능하다.

즉, WINDOW를 쓰는 사람이 LINUX 하에서 MAVEN-JAVA 환경에서 작업을 원한다면 그 작업이 가능하도록 CONTAINER를 제공하는 것이다.

1. 이미지 : 컨테이너를 실행하기 위한 파일 시스템과 실행할 애플리케이션에 대한 모든 설정을 담고 있는 패키지 위에서 오라클을 설치하였으니 이미지를 내려 받으면 oracle11g 버전에 대한 환경을 실행할 수 있는 것이다.
2. 컨테이너 : 이미지를 실행한 상태. 격리된 공간에서 애플리케이션을 실행하는데 필요한 라이브러리와 종속성을 갖고 있음
3. 도커 레지스토리 : 도커 이미지를 저장하고 관리하는 서비스. 로컬 레지스토리는 도커 서버에 내장된 저장소이며 공개 레지스트리는 도커 허브 등의 온라인 저장소
4. 도커 엔진 : 도커를 실행하는 핵심 엔진 도커 엔진은 컨테이너를 생성하고 실행하며 이미지를 빌드하고 저장하는 등의 모든 작업 수행