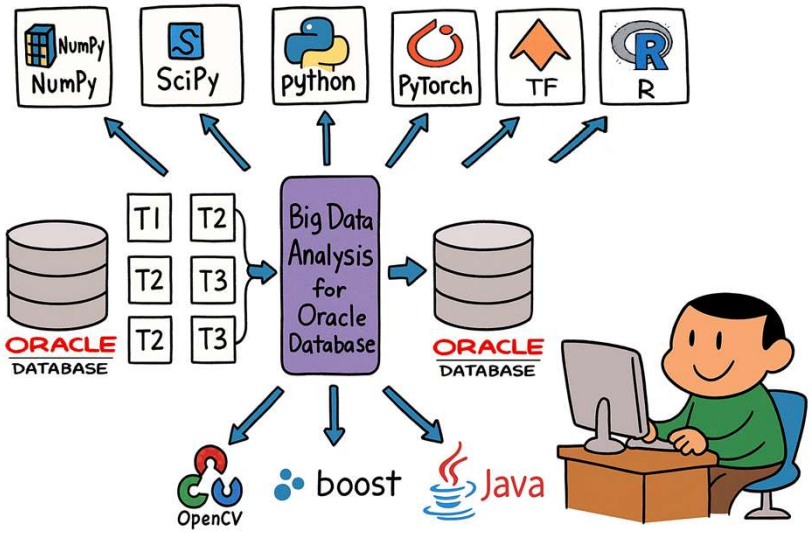
Big Data Analysis Enabler(BDAE™)

매뉴얼 요약본



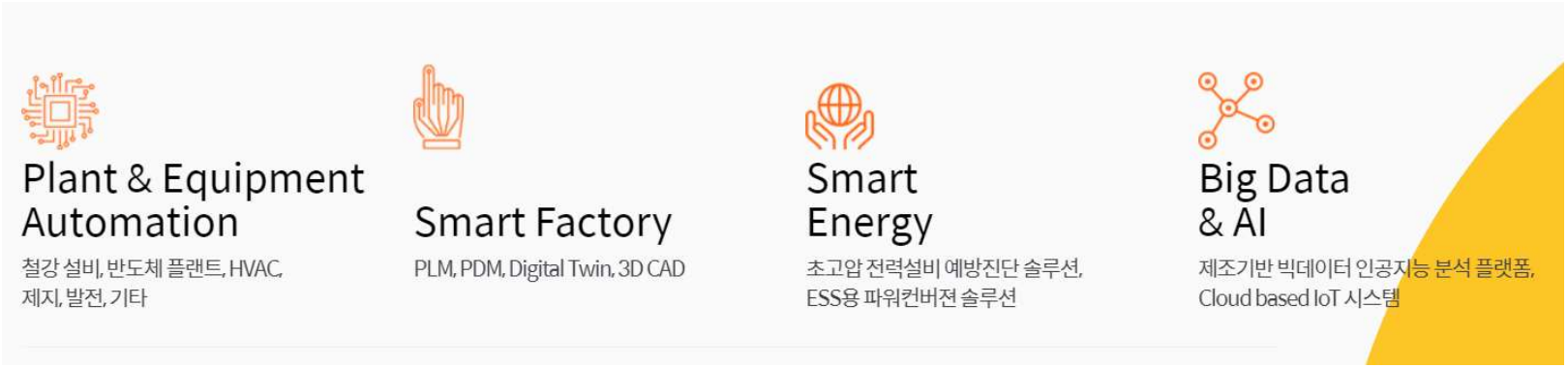
Raymond,

2025/05/16

Copyright @Raymond All rights reserved.

1 개요

BDAE(TM) 는 Oracle Database(TM) 를 사용하는 모든 분야에서 사용 가능 함.
무정지 설치/패치, 알고리즘이나 로직을 즉시 적용하여 기존 솔루션에도 무중단, 바로 적용 가능.



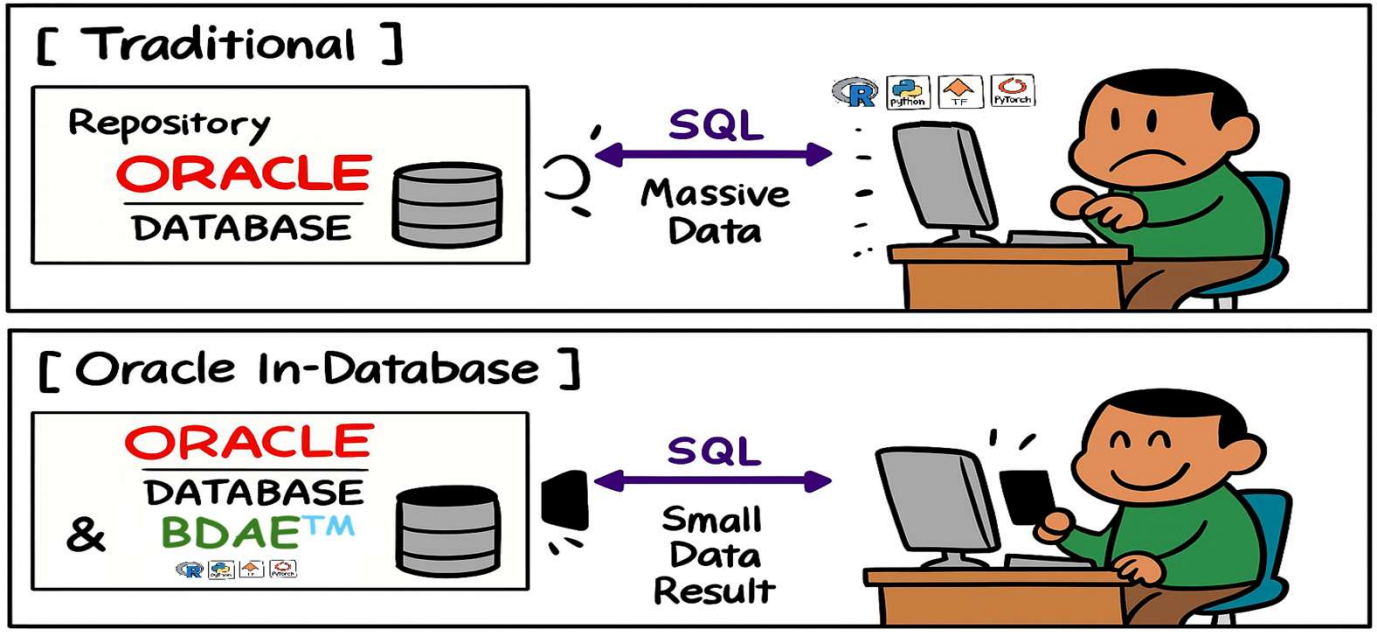
BDAE(TM) 는 SQL 문으로 실행되고 그 결과도 일반 SQL 문의 결과 처럼 조회되므로 별도의 어플리케이션 서버가 불필요하며, Python 과 R 로 모든 로직(백엔드 포함)을 구현할 수 있게 한다.

제조, 금융, 에너지 분야 등의 MES, SPC, FDC, YMS 및 Smart Factory 전 구성 시스템에 사용 가능하며 기존 솔루션에도 간단하게 융합될 수 있는 SQL 방식의 실행 구조를 가짐.

Copyright @Raymond. All rights reserved

2 | BDAE(TM) 의 목적

BDAE(TM) 는 Oracle In-Database 기술을 바탕으로 만들어졌으며 일반적인 AI 업무에서 Oracle Database(TM) 를 단순 저장소 역할로만 사용하는 것을 떠나 학습과 추론 시, 데이터 이동에 대한 Overhead 없이 무정지 운영 환경으로 가져오도록 하는 플랫폼 특징을 가지고 있음.



※ BDAE(TM) 장점은 **개발 생산성(Python, R)**과 **병렬 처리, 성능**을 실시간 영역으로 가져옴.

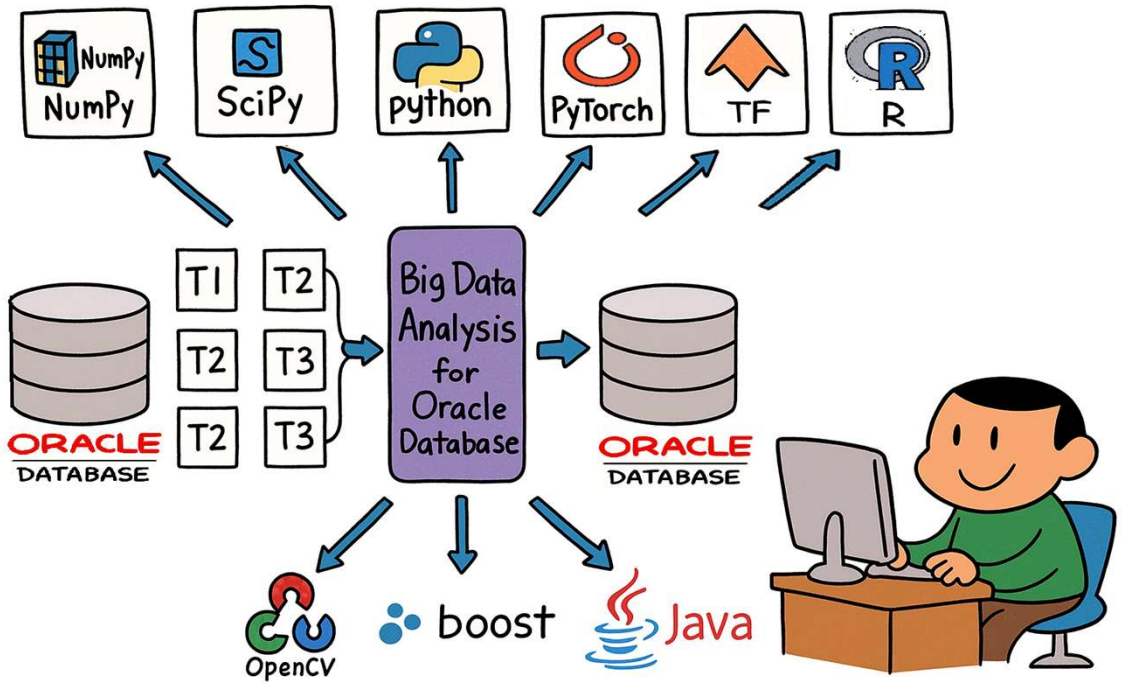
※ 하둡 (Hadoop) 처럼 데이터가 있는 곳에서 프로그램을 실행하는 Concept, Shuffling 없고, 데이터 정렬 등의 비용 최소화, Oracle Database(TM) 의 트랜잭션 능력, 성능, 병렬 처리 이용.

Copyright @Raymond. All rights reserved

3 | BDAE(TM) 의 SW 구성 및 Architecture

Oracle In-Database 형태에서 **BDAE(TM)** 의 동작 위치를 보여 준다.
병렬 분산 처리는 Oracle In-Database 의 기능이며, 분석가들을 자신의 모듈에서 이를 고려할 필요가 없어, **로직의 재 활용성이 증대**된다.

또한 다양한 분석 엔진과 융합 될 수 있다는 점도 **BDAE(TM)** 의 장점이라고 볼 수 있다.
현재는 Python (+Numpy, Scipy, Sklearn, Pytorch, Tensorflow 등) 과 R, 그리고 ETL 용 Java 지원.



Copyright @Raymond. All rights reserved

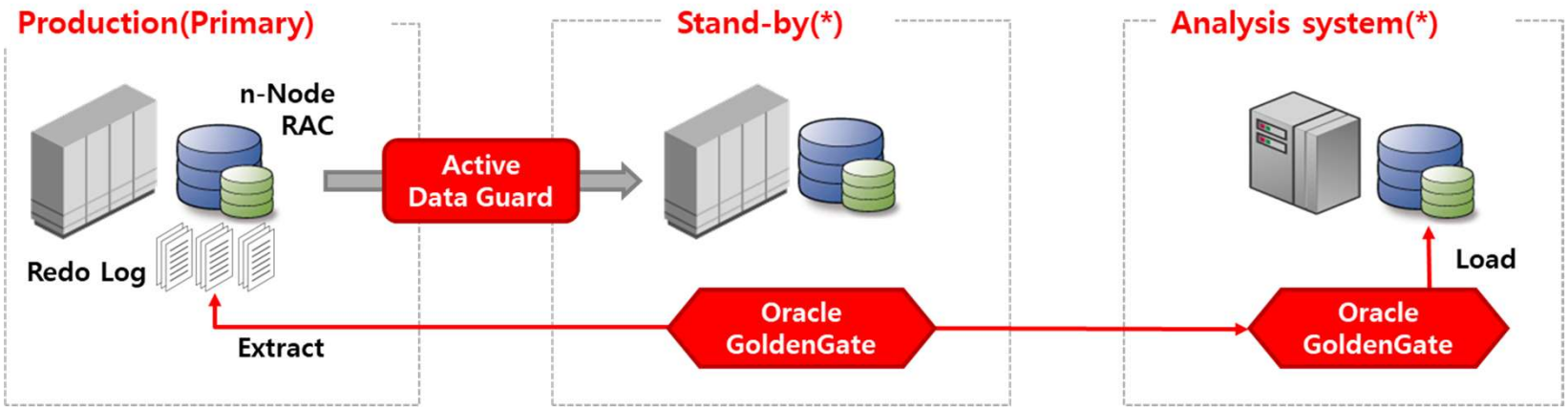
3 | BDAE(TM) 의 SW 구성 및 Architecture

BDAE(TM) 는 Oracle Database(TM) 의 어떠한 아키텍처 구성 형태에서도 동작이 가능하다.

Production 에서는 추론이나, 기술 통계량 등을 구하면 될 것이고, Standby 에서는 모델의 학습을 시키는 방안이나 배치 작업을 하는 것을 추천한다.

별도의 Analysis System 이 있다면 더 좋을 것이다.

Oracle RAC (Real Application Cluster) 기반에서는 BDAE(TM) 는 최고 성능을 보여 줄 수 있으며, 진정한 의미의 Big Data 분석 등이 가능함.



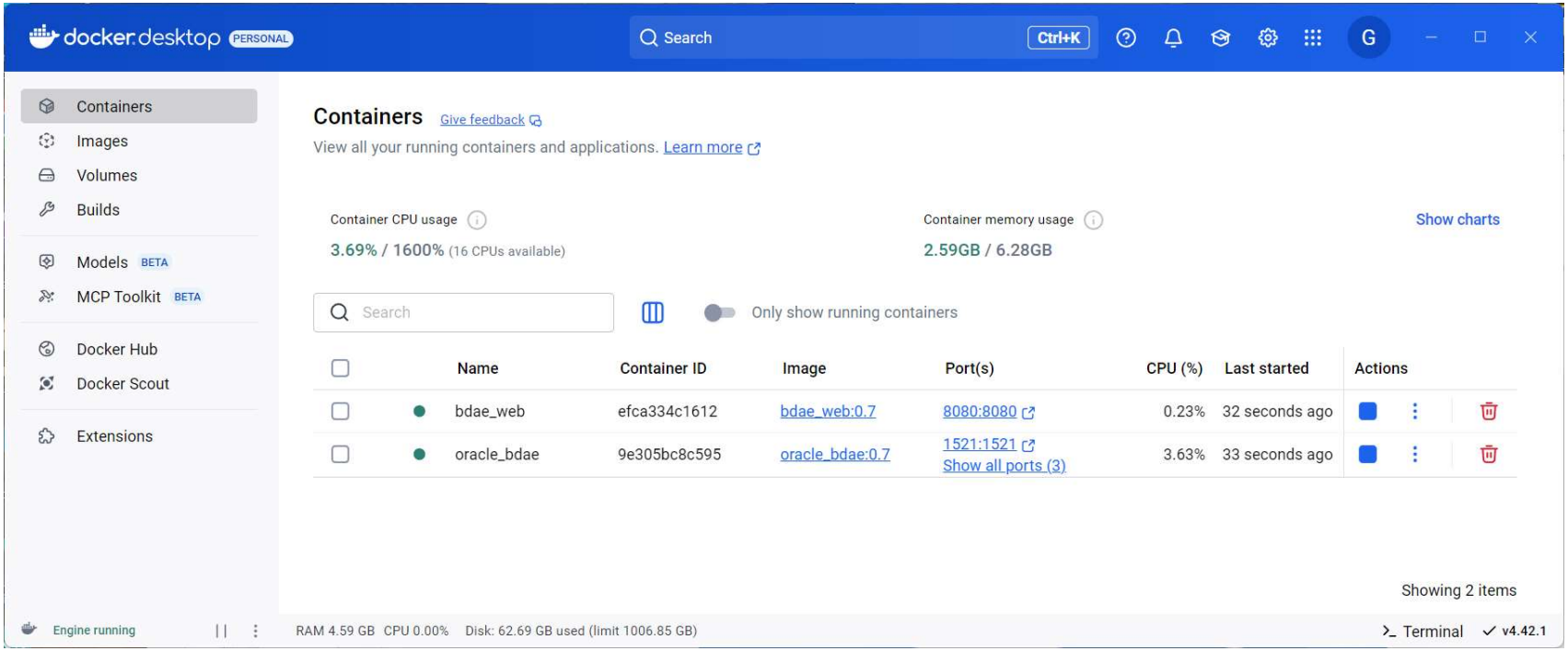
Copyright @Raymond. All rights reserved

4 | BDAE(TM) 클라우드 기반 데모 환경

BDAE(TM) 는 아래 oracle_bdae 와 bdae_web 2개로 구성된다.

bdae_web 은 Spring Boot + JSP 로 되어 있고 oracle_bdae 는 Oracle Database (TM) + Python,R 및 기본 패키지들 + BDAE (TM) 가 설치 되어있으며 Docker export 파일로 배포된다.

(USB 저장장치 또는 클라우드 파일로 전달할 수 있음)



※ 2개의 Docker Image 는 Google Cloud URL 로 메일이 공지. 한달간 공유 예정

Copyright @Raymond. All rights reserved

5 | BDAE(TM) 의 빌드

BDAE(TM) 는 Oracle Linux(TM) (OL, 예전에는 OEL : Oracle Enterprise Linux(TM)) 기반에서 빌드 되었음. Oracle Database(TM) 가 지원하는 어떠한 Linux, UNIX 환경에서 BDAE™ 는 운영 가능.

현재까지 BDAE™ 는 RedHat 계열(Oracle Linux 포함)의 버전 6, 7, 8, 9 에서 모두 빌드가 되어 있으며 한번 빌드되면 Library 파일 형태로 PL/SQL 등과 함께 단 한번 제공되며 설치하면 됨.
(변경 사항은 특별한 경우를 제외하면 없음)

Oracle In-Database 는 반드시 C with PL/SQL, Type, Function 등으로 작성되어야 하기 때문에 빌드가 필요한 것이며 R, Python 등의 엔진도 모두 C 로 되어 있으며, 운영 체제의 디바이스 드라이버와 흡사한 형태로 보면 됨. (Oracle Database 는 OL/OEL 에서 보증되며 GLIBC 버전과 관련 있음)

※ BDAE(TM) 를 직접 설치할 일은 거의 없지만 고객의 알고리즘이 Python 버전에 민감할 경우는 직접 설치해 볼 수 있음.

※Python, R 의 패키지들은 고객이 직접 설치하면 되며 Nexus 기반에서는 Private Network 으로 패키지 위치를 변경 할 수 있음.

※특정 스키마의 병렬 처리 설정은 BDAE™ Oracle Function 만을 만들면 되며 기존 탑재된 예를 참조하면 됨. (향후 도입 시 설명할 사항)

Copyright @Raymond. All rights reserved

6 BDAE(TM) 데모 설치 방법

Windows 환경에서는 Docker Desktop 을 먼저 설치한 후에 다음을 진행한다.
Docker Desktop 설치는 인터넷 해당 문서를 참조하도록 한다.

Oracle Database + BDAE(TM) , BDAE(TM) Web 2개의 Docker Image 를 Import 하고 각각 실행한다.

```
docker load -i oracle_bdae.tar   # Oracle DB + BDAE(TM)
docker load -i bdae_web.tar      # BDAE(TM) Web

# 이미지 Load 되었는지 확인 후 아래를 실행
docker run --name oracle_bdae -p 1521:1521 -p 5500:5500 -p 8888:8888 oracle_bdae:0.5
docker run --name bdae_web -p 8080:8080 bdae_web:0.5
```

Oracle Database 접근 :

- Oracle Service Name : FREE, TNS Port : 1521, IP : 127.0.0.1

Web Server 접근 :

- http://127.0.0.1:8080

Copyright @Raymond. All rights reserved

6 BDAE(TM) 데모 설치 방법 (# NVIDIA GPU)

만약 Docker Desktop 의 PC 또는 서버에 NVIDIA Graphic Card 가 존재하고, 이에 대한 Setup 을 적절하게 한 경우에는 앞서 배포된 이미지를 Load 한 후에 최초 실행 시 다음과 같은 docker 명령을 하게 되면 학습과 추론에 GPU 를 사용할 수 있게 된다.

```
docker load -i oracle_bdae.tar  # Oracle DB + BDAE(TM)

# 이미지 Load 되었는지 확인 후 아래를 실행
docker run -d --ipc=host --name oracle_bdae_gpu --gpus all -p 1521:1521 -p 5500:5500
       -p 8888:8888 oracle_bdae:0.7
```

--ipc=host 를 반드시 주어야 GPU 메모리 오류를 막을 수 있다.

-p 8888:8888 은 jupyter notebook 의 주소이다.

Copyright @Raymond. All rights reserved

7 | BDAE(TM) 의 사용에 앞서 ..

* 상용 제품인 (주)오라클의 Oracle R Enterprise(TM) 은 R 언어만을 지원한다.
물론 훌륭한 분석가들이 주요 알고리즘을 R 패키지로 내장하였지만, 현실은 많은 코드를 짜야 한다.
그리고 짠 코드는 DB 테이블로 저장 되어야 하며 GUI 를 제공하지 않는다.

그런데, 어떠한 UI 도 제공되지 않기 때문에 많은 양의 R 스크립트 코드를 해당 DB 테이블에 넣는 것이 쉽지는 않다.

* 반면, BDAE(TM) 는 Python, R 및 ETL (주로 Hadoop 등)을 목적으로 한 JAVA 를 지원하며, 간단한 Web 화면을 가지고 있고, 이 Web의 Editor 를 통해서 고객의 알고리즘을 DB 로 입력을 하게 된다.

통상 개발은 R의 경우 R studio 나 Python 의 경우 Jupyter notebook 등에서 하는 경우가 많고, 여기에서 개발된 것을 BDAE™ Web 을 통해서 Copy & Paste 해서 입력하면 된다.

※ BDAE(TM) Web 은 번들이며, 기본적인 CRUD 기능과 향후 고객사 나름의 구축 시 참조를 목적으로 함. (Apache NIFI (TM) 와 함께 사용하면 실시간, 배치작업, 스케쥴링 등에 효과적임.)

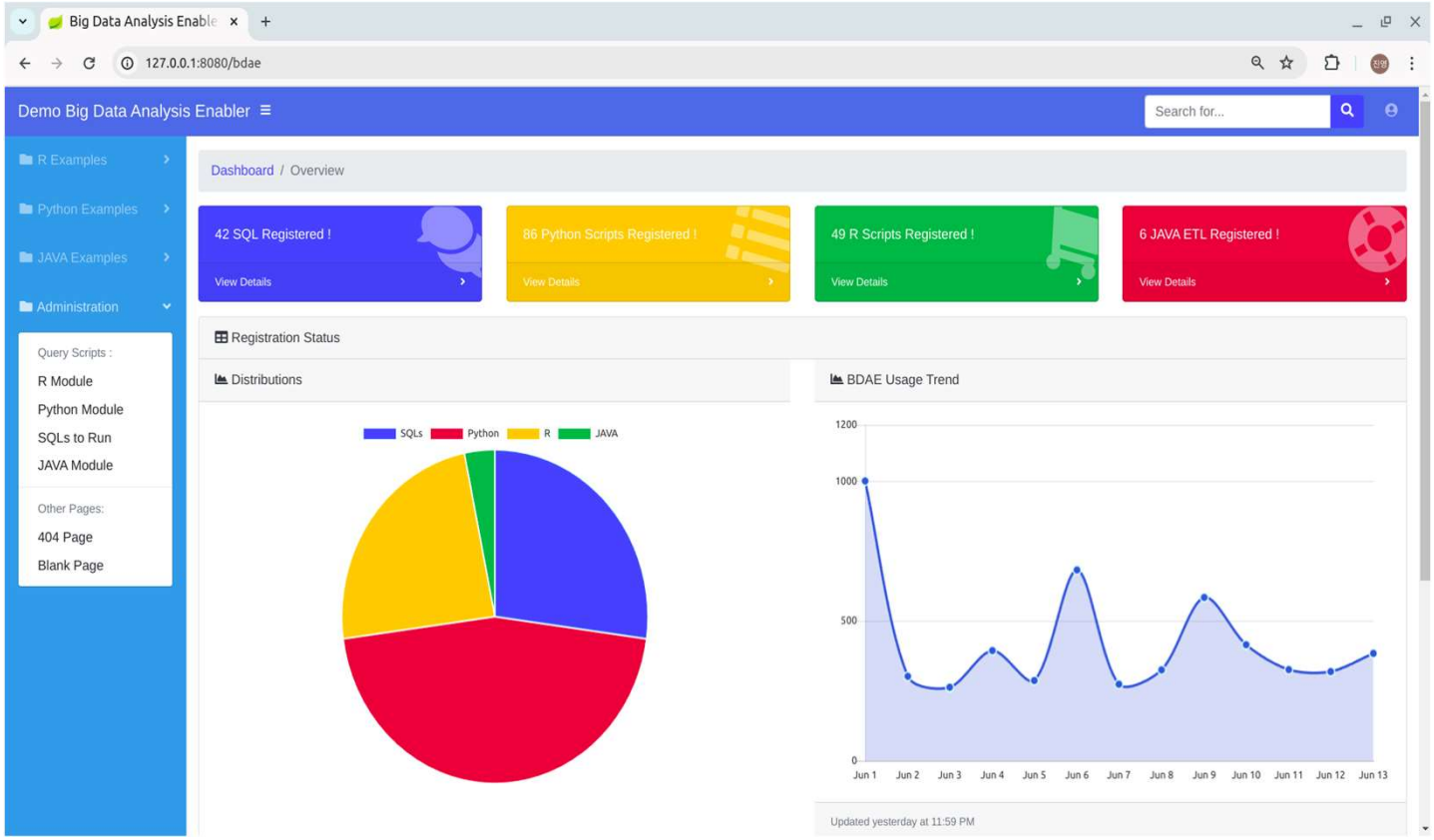
※ Jupyter notebook 에서도 R 은 실행 시킬 수 있으며 BDAE(TM) 설치, 통합 되었다. GitHub 참조
※ 단일 Windows 에서의 R 패키지 설치는 GitHub 의 Troubleshooting 부분을 참조하기 바람.

Copyright @Raymond. All rights reserved

8 | BDAE(TM) Web구성

주로 좌측의 Administration 메뉴를 통해서 R Module, Python Module 을 등록, 조회, 삭제 하고 데이터와 호출을 위한 SQL 을 등록한 뒤 즉시 실행하면서 결과를 확인 해 볼 수 있다.
(※ BDAE(TM) Web 의 기본적인 목적은 Python, R 모듈의 고객의 알고리즘 등록 및 실행이다.)

고객의 자산인 Python, R 의 알고리즘 코드를 DB 에 저장하는 장점은 언제든지 무정지 패치가 가능하고 DB 에서 버전 관리 할 수 있다는데 있다.



Copyright @Raymond. All rights reserved

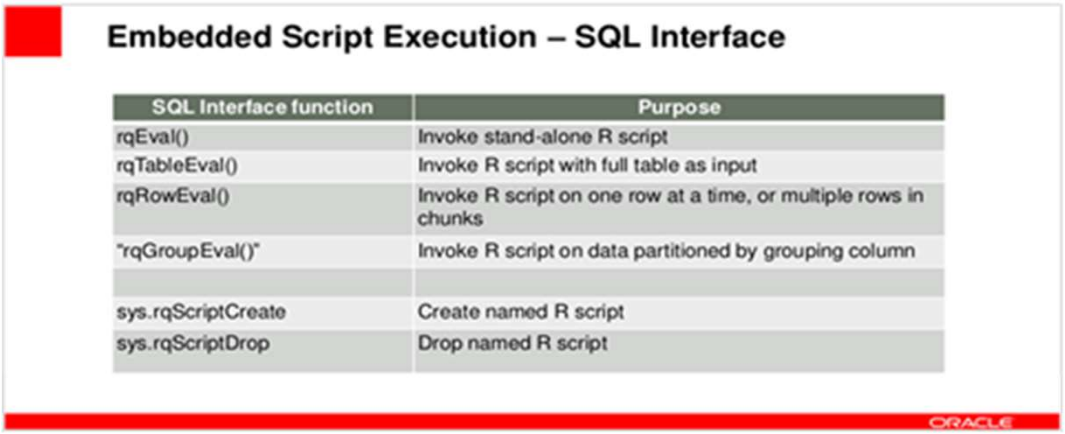
9 Oracle R Enterprise™ SQL 형태

아래는 Oracle R Enterprise™ 에서 제공되는 Embedded Script Execution 형태이다.
맨 아래 2개 (sys.rqScriptCreate, rqScriptDrop) 은 모듈 입력, 삭제에 대한 것이며 좀 불편하다.
BDAE(™) 의 경우 Web Editor 에서 생성, 수정, 삭제가 이뤄지기 때문에 편리하다.

※ Oracle R Enterprise™ 는 R 코드의 버전 관리 기능이 없다.
반면 BDAE(™) 는 버전 관리를 할 수 있는 스키마 구성이 되어 있으며 커스터마이즈 가능하다.

※ rqScriptCreate 내용을 보면 INSERT INTO … 인데 Python 은 들여쓰기가 매우 중요하며 Python, R 모두 " ", ' ' 등을 사용하기 때문에 SQL INSERT 구분을 작성하려면 더 많은 시간이 소모 함.
특히 Python 의 경우 """, "", ' ' 를 섞어 사용하기 때문에, Editor 등의 도움이 없다면 INSERT 구문으로 입력 거의 불가능.

※ Oracle R Enterprise™ 는 제공하는 패키지를 응용하는 정도를 고려하여 코드가 짧을 것으로 고려한 듯,
그러나, 실제 제조현장에서는 거의 쓰지 않는 알고리즘 때문에 구매를 꺼렸다. BDAE(™) 는 따라서 분석가들의
긴 알고리즘 코드를 고려 했으며 Web Editor 를 통해서 쉽게 Python, R 코드를 생성, 수정 할 수 있도록 했음.



Copyright @Raymond. All rights reserved

10 | BDAE(TM) SQL 문의 형태와 R,Python 모듈

BDAE(TM) 는 R, Python 이 모두 되므로 Oracle R Enterprise™ 의 2배, 총 8개의 함수가 제공된다.
그 Argument 는 동일하며 Concept 및 Oracle In-Database 모두 동일한 아키텍처를 따른다.

SQL Interface function	Description (Oracle R Enterprise™ 와 비교)
asEval()	Same rqEval()
asTableEval()	Same rqTableEval()
asRowEval()	Same rqRowEval()
asGroupEval()	Same rqGroupEval()
apEval()	Invoke stand-alone Python module
apTableEval()	Invoke Python module with full table as input
apRowEval()	Invoke Python module on one row at a time, or multiple rows in chunks
apGroupEval()	Invoke Python module on data partitioned by grouping column

Copyright @Raymond. All rights reserved

11 | BDAE(TM) Web 을 이용한 등록 및 실행

BDAE(TM) 는 아래의 3 Step 으로 등록, 실행할 수 있다.

1. R 또는 Python 모듈 등록 (재활용 가능, 분석가의 모듈 수정 거의 없음)
2. DB 데이터와 등록되어 있는 R 또는 Python 모듈과 융합을 위한 앞장의 SQL 문의 작성
3. SQL 문의 실행 및 결과 보기

이것이 전부이다. 실시간에서 즉시 패치하고 기존 버전의 동작 중인 것에 영향을 주지 않는다.

Oracle R Enterprise(TM) 나 BDAE(TM) 모두 분석 및 출력 하고자 하는 데이터는 **Oracle Database(TM)** 내에 존재하기 때문에 모든 것은 SQL 문으로 기술되어야 한다.

즉, 입력과 출력은 모두 Oracle Database(TM) 가 이해할 수 있는 테이블 형태의 데이터 포맷으로 구성되지만, 그 내부의 Temporary 부분은 마음껏 다양한 포맷과 코드로 구성해도 무방하다는 뜻이다.

※ 통상적으로 대용량 분석이나 하둡(Hadoop) 또는 파일 기반의 분석 작업 후 결과를 RDBMS 에 넣는 작업을 하지만 BDAE(TM) 는 자동으로 이러한 작업들이 수행될 수 있다.

※Oracle Optimizer 는 SQL 실행 시 입력과 출력에 대해서 통계 정보 및 Data Dictionary 를 이용하여 확인 후 Plan 을 잡기 때문에, 명시적으로 입.출력에 대해서 반드시 알려 주어야 한다.
Oracle R Enterprise(TM) 나 BDAE(TM) SQL 형태에서 입.출력을 명시한 이유가 바로 이것이다.

Copyright @Raymond. All rights reserved

12 | R 코딩 스타일(#1)

일반적으로 R 코딩은 대략 2가지이며 아래와 같은 함수 형태가 있다.
이때 BDAE(TM) 는 자동으로 함수 argument 인 data, args 이름을 찾아서 그 데이터를 넣어서 호출해 준다. 따라서 data, args 변수 명은 분석가가 임의로 지정해도 무방하다.

```
function(data, args) {
  # 각종 의존성 라이브러리의 등록
  library(xts)
  library(quantmod)
  library(RCurl)
  library(logr)
  library(xts)
  library(quantmod)
  library(RCurl)

  ...
  ... < R 이 지원하는 어떠한 데이터 타입, 모델, 알고리즘 코드를 수용할 수 있음 > ...
  ...

   df <- data.frame(col1=c(...), col2=c(...), ... ,stringsAsFactors=FALSE)
   return (df)
}
```


Copyright @Raymond. All rights reserved

12 | R 코딩 스타일(#2) - BDAE(TM) 만의 특징

또 다른 R 코딩 스타일은 다음과 같은 서술형(순차적으로 실행되는 형태) 이다.
이때는 BDAE(TM) 에서 **data, args** 라는 **명시적인 변수명에 데이터를 넣어서 호출**하기 때문에,
반드시 **data, args** 라는 두가지 변수에 각각 입력 데이터, 부가적인 데이터가 들어간다고 보면 된다.
이 두가지 변수명은 Read-Only 이며 다른 목적으로 사용되어서는 안된다.
맨 마지막에 리턴이 없기 때문에 df 라는 것을 적어주어야 하며, 이 변수명(df)은 임의로 해도 된다.
(※ BDAE(TM) 의 RSCRIPT 테이블의 컬럼 중 STYLE_TYPE=Normal 이 아니어야 한다.)

```
library(xts)
library(quantmod)
library(RCurl)
library(logr)
library(xts)
library(quantmod)
library(RCurl)

x <- data # 입력되는 처리하려는 원시 데이터가 들어 있는 변수는 data 임
y <- args # 부가적인 데이터가 들어 있는 데이터는 args 임
...
... < R 이 지원하는 어떠한 함수, 데이터 타입, 모델, 알고리즘 코드를 수용할 수 있음 > ...
...
df <- data.frame(col1=c(...), col2=c(...), ... ,stringsAsFactors=FALSE)
df
```


Copyright @Raymond. All rights reserved

12 | R 코딩 스타일(#3)

Syntax 오류 체크와 Runtime 오류

분석가는 BDAE(TM) 분석모듈(분석가의 R, Python 코드)에서 예외 처리 하지 않아도 된다.
BDAE(TM) 에서 문제가 되면 SQL 의 오류코드(ORA-20001 ~ 20500)와 R 오류 메시지로 반환된다.

BDAE(TM) 는 R 의 경우 먼저 Syntax 체크를 수행한 후 문제가 없는 경우에만 데이터를 가공해서 해당 모듈을 호출한다. 문제가 되면 분석가는 이를 해결한 후 다시 실행하도록 한다.

BDAE(TM) Python 은 동적으로 결정되며 Trace Back 이 훌륭하기 때문에 따로 Syntax 체크를 별도로 하지 않는다.

※ R 은 Python 에 비해서 예외 처리가 빈약한 편이다. C 언어로 되어 있는 R 의 API 가 한계가 있다.

Copyright @Raymond. All rights reserved

13 | Python 코드(#1)

Python 은 R 과 다르다. Python 은 모듈 이름 (통상 파일 명)과 많은 함수들 중 시작 함수 명이 있어야 호출 가능하다. (__main__ 을 호출하는 일반적인 방식이 아니다.)

```
import pandas as pd
import numpy as np
import seaborn as sns
from StreamToLogger import StreamToLogger
import sys
import logging, logging.handlers

def describe(df):
   logger = logging.getLogger('TitanicDesc:describe')
    logger.setLevel(logging.DEBUG)
    socketHandler = logging.handlers.SocketHandler('localhost',
                    logging.handlers.DEFAULT_TCP_LOGGING_PORT)
    logger.addHandler(socketHandler)

    sys.stdout = StreamToLogger(logger,logging.INFO)
    sys.stderr = StreamToLogger(logger,logging.ERROR)

   print('----- start -----')
   colums_ = df.columns.tolist()
   print(str(colums_))
   df.columns = list(map(str.lower,colums_))
   df_desc = df.describe()
   print(str(df_desc))
   df_desc.reset_index(inplace=True)
   df_desc.columns = ['vars', 'passengerid', 'survived', 'pclass', 'age', 'sibsp', 'parch', 'fare']
   df_melt = pd.melt(df_desc, id_vars=['vars'])
   print('----- end -----')

    socketHandler.close()
    logger.removeHandler(socketHandler)

   return df_melt
```

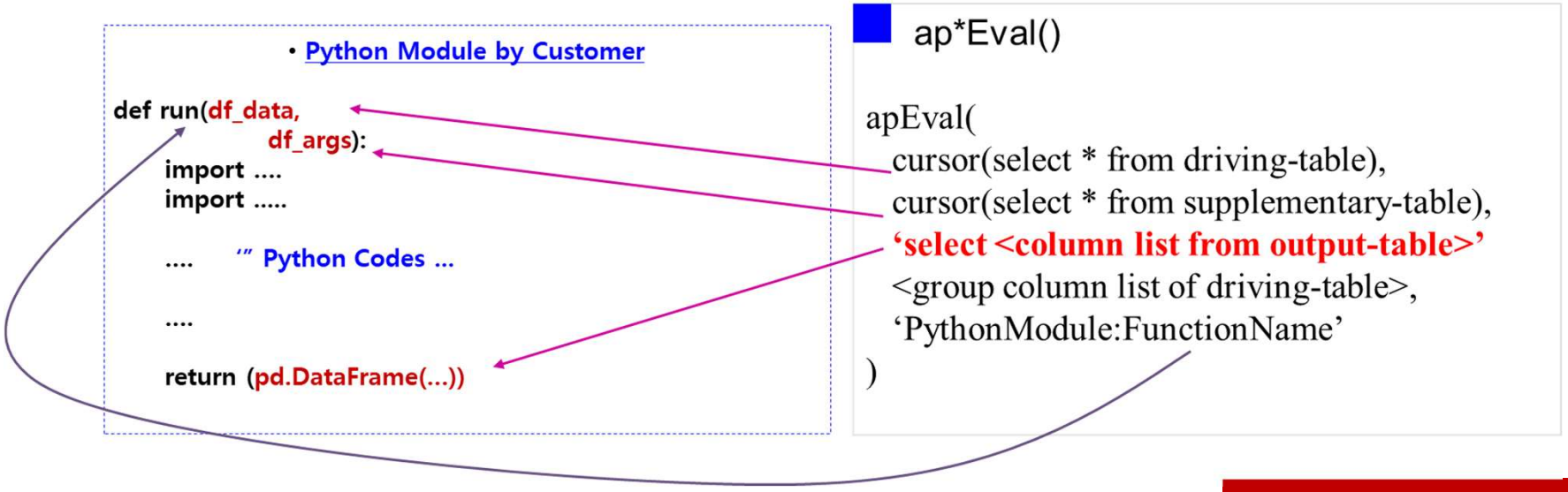

Copyright @Raymond. All rights reserved

13 | Python 코드(#2)

Python 모듈의 시작 함수의 argument 는 R 과 동일하게 2개이다.
앞의 것이 입력 데이터, 뒤의 것이 부가적인 데이터이다. 이름은 마음대로 정하면 된다.

R 과 달리, 서술형태는 지원하지 않는다. 또한 모듈 명은 클래스가 아니다. (주의)

※ 병렬 처리의 경우는 함수 명(ap(s)GroupEval)이 다르고, **병렬처리 컬럼명들**을 **반드시** 제공해야 한다.
※ Oracle Database(TM) 의 SQL 실행 메커니즘을 이해할 필요가 있다. Oracle Optimizer 는 입력과 출력 형태를 보관 중인 통계 정보를 기반으로 Plan 을 만든다. 최종 Python 출력을 Oracle 이 자동으로 인지하지 못하므로 이를 알려 주어야 한다. (우측 빨간 색)
※ 출력이 복잡한 경우 Oracle View 로 만들고 'VIEW_NAME' 만 적어 주면 된다. BDAE(TM) 기능
※ Python, R 출력을 SELECT 구문으로 자동으로 만드는 부분은 GitHub 에 있고, 이를 View 로 만들 것

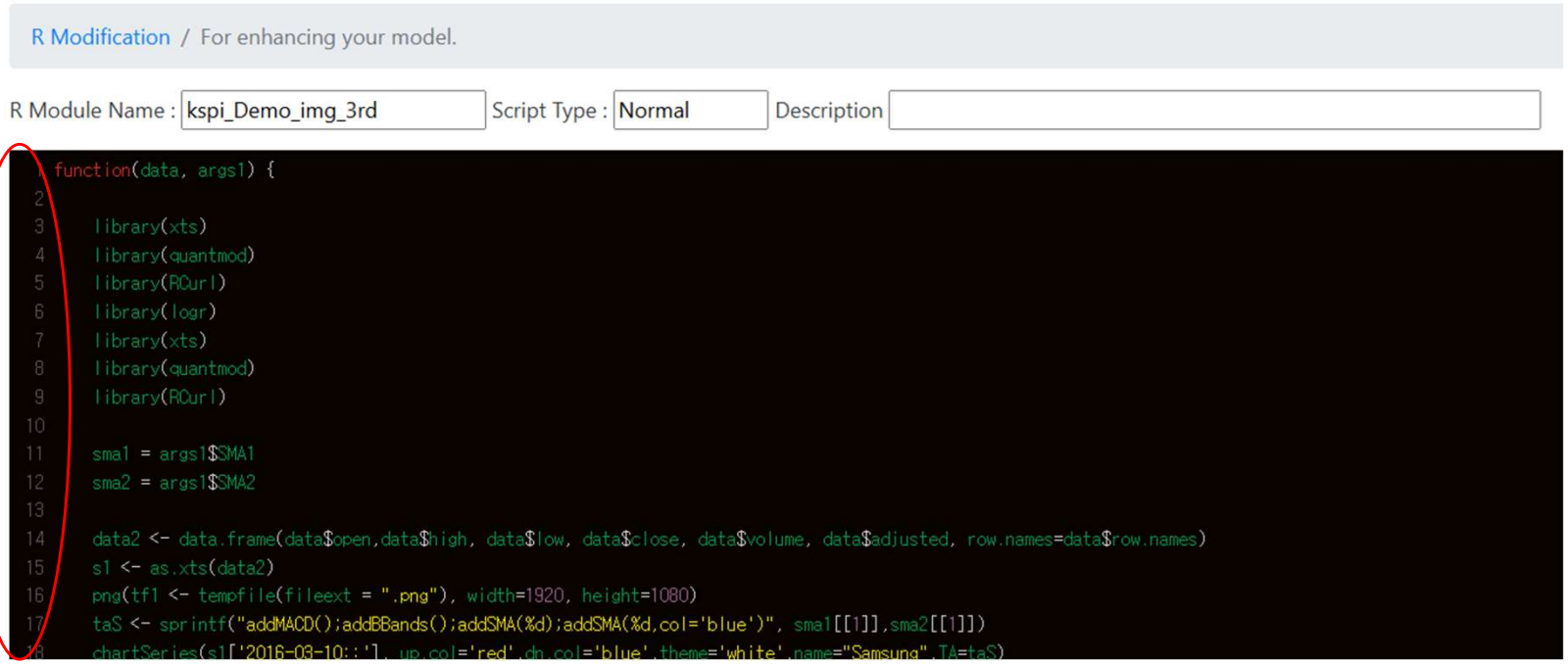


Copyright @Raymond. All rights reserved

14 | BDAE(TM) Web Editor

BDAE(TM) Web Editor 부분은 오픈소스 패키지를 임포트 해서 사용했다. 여기에서 개발하라는 뜻은 아니며, 기존 개발 툴 (R studio, Jupyter notebook 등)에서 개발한 것을 복사해서 넣으면 된다.

※ **Syntax 오류 시에 라인 번호가 있는 경우 이것을 활용하면 좌측의 라인번호가 바로 그것이다.**
BDAE(TM)가 정의한 ORA-20000 대의 Oracle 오류 메시지 안에 포함되어 있다.



※ R 은 Python 에 비하면 예외처리 등이나 디버깅이 매우 불리하다.
Python 은 실행 시의 오류에 대해서 traceback() 이 매우 정확하며 API 가 잘 정리되어 있다.

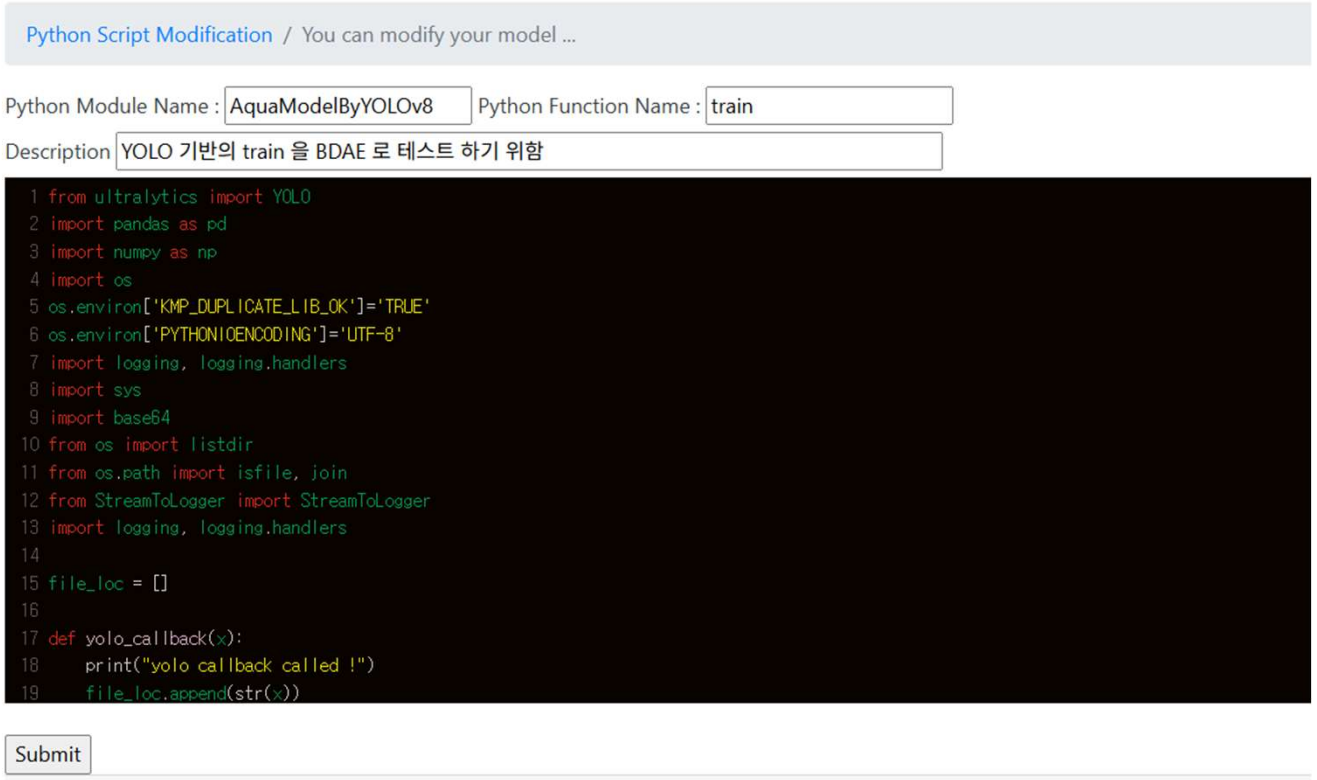
Copyright @Raymond. All rights reserved

15 | BDAE(TM) Python CRUD

Python 의 경우 Module Name 과 Function Name (시작 함수) 두가지가 반드시 존재해야 하며,
BDAE(TM) 는 DB 에 있는 경우와 File 구성 두가지를 모두 실행해 주되, DB 가 우선적으로 실행된다.

즉, DB 에 없다면 PYTHONPATH 위치에서 Module 을 찾는다는 의미이다. (설정은 고객이 함)

※ 파일로 되어 있는 Python 모듈의 문제점은 버전 관리가 취약하다는 점이며, Oracle RAC 환경에서는 모든 Node 에 해당 파일을 설치를 해 주어야 한다는 점이다.



Copyright @Raymond. All rights reserved

16 | Python DataFrame, R data.frame

모든 RDBMS 의 테이블은 2D Tensor 이기 때문에 이것에서 벗어나질 못한다.
따라서 입력 값과 출력 값은 모두 Python Pandas DataFrame 과 R 의 data.frame 포맷으로 작성해 주어야 한다. (**R 의 경우 data.frame, data.table 2가지 출력 모두를 지원한다.**)

입력은 **BDAE(TM)** 가 자동으로 해 주지만, 출력은 분석가가 리턴할 때 만들어서 리턴해 주어야 한다.
입력 시에 대 소문자가 문제인데, 아래 처럼 하면 대소문자가 섞인 컬럼도 가능하다.

SELECT 의 "col1", .. 로 되어 있다면 **대소문자가 섞인 형태**도 그대로 **BDAE™ 가 넘겨줄 것**이다.
이 부분은 분석가들이 알아야 한다. **Pandas DataFrame 컬럼을 이용할 경우 주의를 기울여야 한다.**
차라리 컬럼들을 모두 대문자, 또는 소문자로 바꾼다음에 작업해도 될 것이다.
(※ df['col1'] 등으로 Python, R 등에서 사용하며, 이때 대소문자가 중요하기 때문이다.)

SELECT **"Passenger" as Passenger** 라고 하면 대문자로 나올 것이다.
이건 **BDAE(TM)** 가 하는 것이 아니라 Data Dictionary API 때문이니 알고 넘어가야 한다.

출력의 경우도 "" 로 대소문자를 섞을 경우에는 그대로 나오지만, 아닌 경우는 대문자로 나온다.

※ 이 부분은 **BDAE(TM)** 때문이 아니라 Oracle Database(TM) SQL 엔진 특성이다.
※ SELECT * FROM .. 으로 하면 테이블 생성 당시의 컬럼 대.소문자가 그대로 적용 된다.

Copyright @Raymond. All rights reserved

17 | NA, NaN, Inf (#1)

분석에서는 NA (Not Available), NaN (NULL), Inf (Infinity) 가 중요하다.
그러나, Oracle Database(TM) 에서는 NA, NaN 는 동일하게 값이 NULL 이며,
Inf 는 다음과 같이 제공된다. 아래 R 의 Infinity 는 1.0/0.0 은 양의 Infinity, -1.0/0.0 은 음의 Infinity.

1. R 의 경우

```
X <- c(1,2,3,NA,5)
Y <- c(1.1,-1.0/0,1.0/0,4.0,5.34)

df <- data.frame(X,Y)
df
```

2. Python 의 경우

```
import pandas as pd
import numpy as np

def returnNAN():
    df = pd.DataFrame([['motor type',1, np.inf],
                       [np.nan, 2, 3.2],
                       ['RF', np.nan, 4.5]],
                       columns = list('abc'))
    return df
```


Copyright @Raymond. All rights reserved

17 | NA, NaN, Inf (#2)

Oracle 에서는 Infinity 를 binary_double_infinity 로 리턴하지만, JAVA 에서 이 SQL 을 단순 호출하면 Overflow 가 발생 하니 주의해야 한다. (이 부분은 Oracle Database(TM) 나 BDAE(TM) 문제가 아님.)

```
SELECT A, case when B=-binary_double_infinity
                then '-Infinity'
                when B=binary_double_infinity
                        then '+Infinity'
           else TO_CHAR(B) end AS B
FROM (
       SELECT *
        FROM
        table(asEval(
        NULL,
        'SELECT 1 as A, 1.0 as B FROM dual',
        'R_infinity'))
);
```

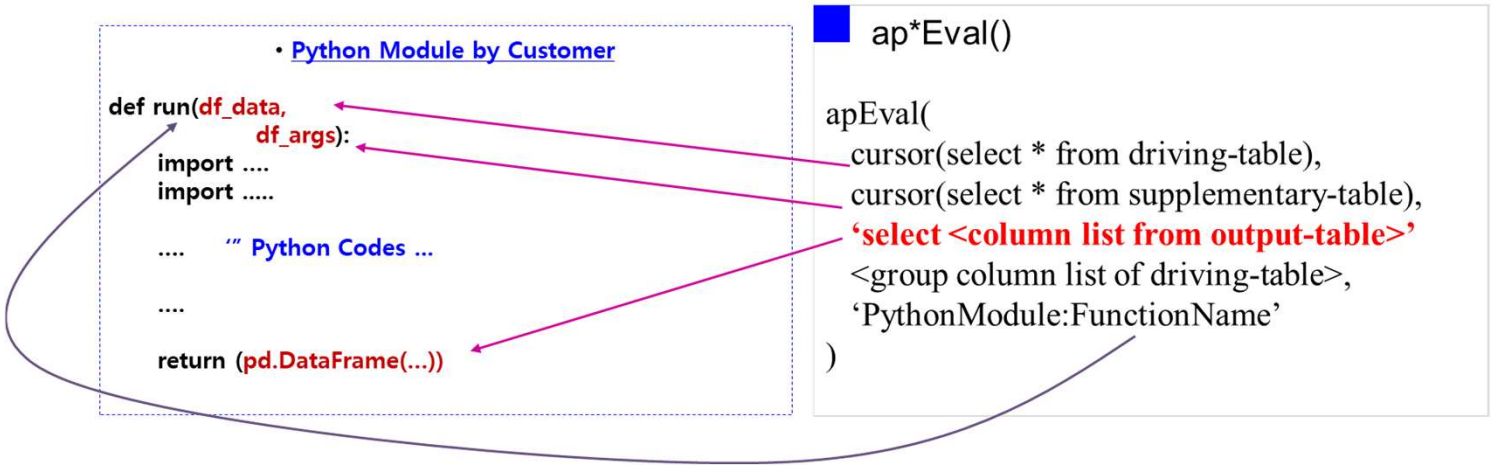

Copyright @Raymond. All rights reserved

18 | 리턴 DataFrame과 Oracle 매칭

우측의 빨간 색 'select <column list from output-table>' 부분은 보통 다음과 같이 dual 을 사용한다. 그런데, 아래와 같이 이 컬럼이 많을 경우에는 짜증 스럽다. 따라서 이를 View 로 만들어 사용하면 편리하다. 'V_OUTPUT1' 으로 입력하면 된다. (※ 이 부분은 BDAE(TM) 의 기능이다.)

```
'SELECT CAST(''AA'' AS VARCHAR2(40)) EQP,
        CAST(''AA'' AS VARCHAR2(40)) UNIT,
        CAST(''AA'' AS VARCHAR2(40)) LOT,
        CAST(''AA'' AS VARCHAR2(40)) WAFER,
        CAST(''AA'' AS VARCHAR2(40)) RECIPE,
        CAST(''AA'' AS VARCHAR2(40)) PARAM,
        CAST(''AA'' AS VARCHAR2(100)) LOCATION
        FROM DUAL'
```

```
CREATE VIEW V_OUTPUT1 AS
SELECT CAST(''AA'' AS VARCHAR2(40)) EQP,
        CAST(''AA'' AS VARCHAR2(40)) UNIT,
        CAST(''AA'' AS VARCHAR2(40)) LOT,
        CAST(''AA'' AS VARCHAR2(40)) WAFER,
        CAST(''AA'' AS VARCHAR2(40)) RECIPE,
        CAST(''AA'' AS VARCHAR2(40)) PARAM,
        CAST(''AA'' AS VARCHAR2(100)) LOCATION
        FROM DUAL;
```



Copyright @Raymond. All rights reserved

19 | 개발 시, 실시간 로깅

시스템 오류가 나면 Python, R 모듈 안의 로그는 당연히 볼 수 없다.

(※ BDAE(TM) 는 Python, R 개발환경 처럼 한 줄씩 실행되는 것이 아니기 때문이다.)

BDAE(TM) 는 Python, R 오류 시에 ORA-20000 대에 정의한 Description 형태로 해당 오류를 전달해 준다.

오류가 없으면서 디버깅을 하려면 파일로 매번 하는 것 보다는 개발 시 원격 로깅을 사용하는 것이 좋다. 이 부분은 GitHub 에 넣어 두었다.

Python 의 경우 LogServer.py 파일을 실행 시키면 된다.

Copyright @Raymond. All rights reserved

20 | Python 모듈 개발 할 때 (Input 에 대한 부분 확인)

BDAE(TM) 없이 개발 환경에서는 Python sqlalchemy 패키지를 사용할 가능성이 있다.
pip install sqlalchemy, cx_Oracle 만 설치한다고 동작하지 않는다.
Oracle Client 가 있어야 하는데, 이것은 Oracle 사이트에서 다운 받으면 된다.
사용법은 아래와 같다. 이것으로 컬럼을 확인하면서 모듈 개발을 하면 된다.

```
import sqlalchemyimport pandas as pd
import pandas as pdimport cx_Oracle
import osfrom sqlalchemy
import create_engine
LOCATION = r"C:\Users\Admin\Downloads\instantclient-basic-windows.x64-23.8.0.25.04\instantclient_23_8"
os.environ["PATH"] = LOCATION + ";" + os.environ["PATH"]

oracle_connection_string = 'oracle+cx_oracle://{username}:{password}@{hostname}:{port}'
DATABASE = "FREE"
SCHEMA = "rquser"
PASSWORD = "0000"
engine = create_engine(    oracle_connection_string.format(username=SCHEMA,
            password= PASSWORD,   hostname='177.175.54.97',    port='1521',    database='FREE',    ))

conn = engine.connect()
SQL = \
'''
SELECT ...
'''
df = pd.read_sql_query(SQL, conn)
```


Copyright @Raymond. All rights reserved

21 | R 모듈 개발 할 때 (Input 에 대한 부분 확인)

BDAE(TM) 없이 개발 환경에서는 ROracle 패키지를 사용할 가능성이 있다.
ROracle 설치는 Github 나 인터넷을 활용하고, 다만 Windows 에서 설치는 GitHub 에 넣어두었음.
Oracle Client 가 있어야 하는데, 이것은 Cloud 에 올려 두었다. (설치 파일 폴더)
사용법은 아래와 같다. 이것으로 **BDAE(TM)** 입력을 예상해서 개발 한다.
(※ Windows 의 ROracle 는 재조립된 ROracle.tar.gz 로 해야 오류를 잡을 수 있으니 GitHub 에서 ..)

```
# 아래는 Oracle 설치 위치 임. PATH 잡혀 있을 경우 무시 가능

Sys.setenv("ORACLE_HOME"="/u01/app/oracle/product/12.2.0.1/db_1")
library(DBI)
library(ROracle)

driv <- dbDriver("Oracle")

connect.string <- paste(
  "(DESCRIPTION=",
  "(ADDRESS=(PROTOCOL = TCP)(HOST = 177.175.54.97)(PORT = 1521))",
  "(CONNECT_DATA=(SERVER = DEDICATED)",
  "(SERVICE_NAME = FREE)))", sep = "")

conn <- dbConnect(driv, username="rquser", password="0000", dbname=connect.string)
df <- dbGetQuery(conn,"SELECT * FROM FDC_TRACE WHERE ROWNUM < 10")
```


Copyright @Raymond. All rights reserved

22 | 모듈 개발 할 때 출력 부분 확인(Python #1)

결국 BDAE(TM) SQL 의 SELECT ... FROM DUAL 부분을 매번 수작업 하기 어렵기 때문에 유틸리티를 만들어 보았다.

Output 부분인데, 아래를 사용하면 만들어 주고, 그것을 다시 View 로 만들면 편리하다.

```
def dtype_to_dbtype(typestr):
    return {
        'int64': lambda: '1',
        'object': lambda: "CAST(''AA'' AS VARCHAR2(40))",
        'float32': lambda: '1.0',
        'float64': lambda: '1.0',
        'datetime64[ns]': lambda: 'TO_TIMESTAMP(NULL)',
        'byte': lambda: 'TO_BLOB(NULL)'
    }.get(typestr, lambda: typestr + "not defined type.")()

def space_fill_underbar(column_name):
    return '_'.join(column_name.split(' '))
```


Copyright @Raymond. All rights reserved

22 | 모듈 개발 할 때 출력 부분 확인(Python #2)

아래 generate_dual_select (df) 을 호출하면 "SELECT ... FROM dual" 문을 만들어준다.

```
def generate_dual_select (df):
    types = df.dtypes
    column_name_list = []
    column_type_list = []
    for i in range(len(types.index.tolist())):
        column_name_list.append(types.index.tolist()[i])
        column_type_list.append(str(types[types.index.tolist()[i]]))
        print("%-40s %s" %(types.index.tolist()[i], str(types[types.index.tolist()[i]])))

    sql = 'SELECT '
    last_index = len(column_name_list) - 1
    for i in range(last_index + 1):
        if i == last_index:
            sql = sql + ' ' + dtype_to_dbtype(column_type_list[i]) + ' ' + space_fill_underbar(column_name_list[i])  + '\nFROM dual'
        else:
            sql = sql + ' ' + dtype_to_dbtype(column_type_list[i]) + ' ' + space_fill_underbar(column_name_list[i]) + ',\n'

    return (sql)
```

```
sql = generate_dual_select(df)
print(sql)
```

Copyright @Raymond. All rights reserved

23 | 모듈 개발 할 때 출력 부분 확인(R)

R의 경우에도 아래 generate_dual_select(df) 을 호출하면 "SELECT ... FROM dual" 문을 만들어 준다.

```
r_to_oracle_type <- function(r_type)
{
 switch(r_type,
       "character" = "VARCHAR2(4000)",
       "integer"   = "NUMBER(10)",
       "numeric"   = "FLOAT",
       "double"    = "FLOAT",
       "logical"   = "CHAR(1)",
       "Date"      = "DATE",
       "POSIXct"   = "TIMESTAMP",
       "factor"    = "VARCHAR2(4000)",
       "list"      = "CLOB",
       "unknown"   = "VARCHAR2(4000)",
       "VARCHAR2(4000)")  # default
}
```

```
# 각 컬럼에 대해 SELECT 구문 생성
generate_dual_select <- function(df)
{
 types <- sapply(df, function(col) class(col)[1])
 oracle_types <- sapply(types, r_to_oracle_type)


 select_parts <- mapply(function(col_name, ora_type) {
  paste0("CAST(NULL AS ", ora_type, ") AS ", col_name)
 }, names(df), oracle_types, USE.NAMES = FALSE)


 paste("SELECT", paste(select_parts, collapse = ", "), "FROM dual")
}


# 실행
query <- generate_dual_select(df)
cat(query)
```

Copyright @Raymond. All rights reserved

24 | SQL 문

* Dynamic SQL 문을 사용하지 않는다면 이는 고객사의 스키마에 의존성을 갖게 되는 것이며 이는 매번 고객사에 맞도록 코드를 변경해야 한다는 뜻이다.

그러나, BDAE(TM) 는 Dynamic SQL 이 가능하도록 구현되었고, 이는 차별화된 핵심적인 기능이다. Oracle R Enterprise (TM) 와 같은 수준으로 BDAE(TM) 도 가능하도록 구현 되었다.

**(※ Database Dictionary 를 완벽하게 지원하는 RDBMS 는 오직 Oracle Database(TM) 뿐이다.
Dynamic SQL 의 핵심은 SubQuery 의 SubQuery .. 가 되어도 최상위 컬럼 정보의 Dictionary 를 투명하게 지원하지 않으면 안되기 때문이다. 아, API 레벨에서 이야기 하는 것이다. SQL 실행이 아닌 ..)**

* 병렬 처리 부분

1. Oracle Optimizer 는 입력 Query 와 출력 Query 및 통계정보를 확인하고 병렬 처리를 결정한다.
2. 따라서, 이미지(BLOB) 이나 4000 바이트 이상의 문자열 (CLOB) 이 포함된 것을 확인 하면 병렬 처리를 Oracle Database(TM) 는 포기한다.
3. BDAE(TM) 를 이용할 때도 마찬가지로 적용 된다. 단, 이것을 회피하고 병렬 처리 가능하도록 꽁수를 부릴 수 있는 방안도 얼마든지 있지만, 일반 SQL 문이 아닌 BDAE(TM) 를 써야 가능하다.
4. BDAE(TM) 의 Python 모듈을 한번 사용하면 PL/SQL 을 잘 사용하지 않게 된다. 개발 생산성 !

Copyright @Raymond. All rights reserved

25 | 분석가들이 준비하는 일

BDAE(TM) SQL 을 만들기 전에 분석가들은 다음과 같이 하면 된다.
병렬 처리, Raw 데이터를 직접 가져오기 등을 고려하지 말고, 단일 모듈, 단일 함수를 만들면 된다.

import sqlalchemy 를 기반으로, 먼저 SQL 로 DataFrame 을 만들어서 개발해 본다.

SQL1 = "SELECT .. " # 이 SQL 문은 분석 하고자 하는 Query 를 만들어 사용한다.
df_data = pd.read_sql_query(SQL1, conn)

만약 만들고자 하는 Class 의 함수가 인자가 필요한 경우는 2가지가 있다.
1) 단순 Scalar 의 조합
SQL2 = "SELECT 1 as ARG1, 0.5 as ARG2 FROM DUAL"

2) 참조되는 데이터 (참조 테이블, 예를 들면 유사도 측정을 위한 Reference 테이블, 추론 테이블)
SQL2 = "SELECT ... FROM TABLE .... "

df_args = pd.read_sql_query(SQL2, conn)

그 두가지를 가지고 목적의 함수를 만들면 되고 그 모듈 명과 함수명을 BDAE(TM) 에 등록 한다.

(※ R 은 ROracle 을 이용해서 작업하면 되고 역시 과정은 비슷하다.)

Copyright @Raymond. All rights reserved

26 | R 예제 참고 사항

R 은 앞서 언급한 바와 같이 코딩 스타일이 두가지 이다.

1. function() 형태 : 입력 변수는 임의의 이름으로 사용 가능, 명시적 리턴 return df
2. 서술 형태는 분석하려는 입력 데이터명은 data 이고, 부가적인 데이터는 args 로 고정되어 있다.
   return 구문 없이 df 를 한번만 써 주면 되며 df 이름 말고 다른 이름도 된다.
   단, 1,2 모두 리턴은 data.frame 포맷이며 **stringsAsFactor=FALSE** 로 해야 한다.

   stringsAsFactor 부분은 R 분석가들은 당연히 잘 알고 있을 것이다.
   무심코 data.frame 을 만들면 Category 성 컬럼은 1,2,3 등의 Factor 로 저장된다는 것을 . . .

3. Python 처럼 리턴되는 data.frame 의 속성은 보편적으로 문자열, 숫자, DateTime 정도이다.
   as.character(), as.numeric(), ..

Copyright @Raymond. All rights reserved

27 | Visualization 과 Serialization/DeSerialization

R/Python 모두 3가지 형태로 차트를 만들 수 있다.

1. 이미지 (GitHub 참조)
2. 이미지, base64 로 인코딩 한 String (GitHub 참조)
3. plotly 를 사용한 Interactive Java Script String (GitHub 참조)

2 번은 추천하지 않는다. 사이즈만 커지기 때문이다. 3 번은 대용량일 경우 추천하지 않지만, 어느 정도 작을 때에는 Mouse 이벤트를 받으면서 Interactive 해서 매우 직관적이다.

이미지나 모델의 바이너리를 data.frame 으로 만드는 것은 익혀두는 것이 좋다.
특히 모델을 학습 시킨 후에 data.frame 으로 만들어 DB 에 저장 후 추후에 이것을 다시 추론에 사용하는 기법은 보편적이다.

AI 학습 시에는 많은 검증 데이터 결과들, 각종 시각화 차트들, 파라미터들이 나오고 이를 보고 싶어 한다. SQL 문의 리턴은 단 1회이기 때문에, 이 모든 것들을 DataFrame, data.frame 에 모두 담을 수 있다.

특히 Model 의 경우는 Serialization 해야 하기 때문에 결국 Oracle 의 BLOB 으로 저장해야 하고, 추후 운영 시 실시간 추론(Inference) 를 할 때 이를 이용해 다시 DeSerialization 해서 사용해야 한다.

Copyright @Raymond. All rights reserved

28 | BDAE(TM) 는 실행 시 저장 공간을 사용하지 않는다.

BDAE(TM) 는 실행 시와 Python, R 엔진을 통해서 분석 후에 리턴 시 별도의 저장 공간이 아닌 모든 것이 메모리에서 이뤄진다.

BDAE(TM) 가 필요한 저장 공간은 오직 고객사의 Python, R 모듈을 관리 할 때만 사용하며 이는 거의 공간을 차지한다고 볼 수 없다. 실제 운영 데이터를 핸들링하고 분석 결과 등은 모두 메모리이다.

BDAE(TM) 의 리턴 결과를 DB 에 저장하는 방법은 Python, R 내에서 분석가가 별도의 세션으로 하는 방법, 그리고 다음과 같은 보편적인 Database 기법을 이용하면 된다.

```
CREATE TABLE RESULT_TEMPORARY_TABLE AS SELECT ... ;
INSERT INTO RESULT_PERMANENT_TABLE SELECT ... ;
```

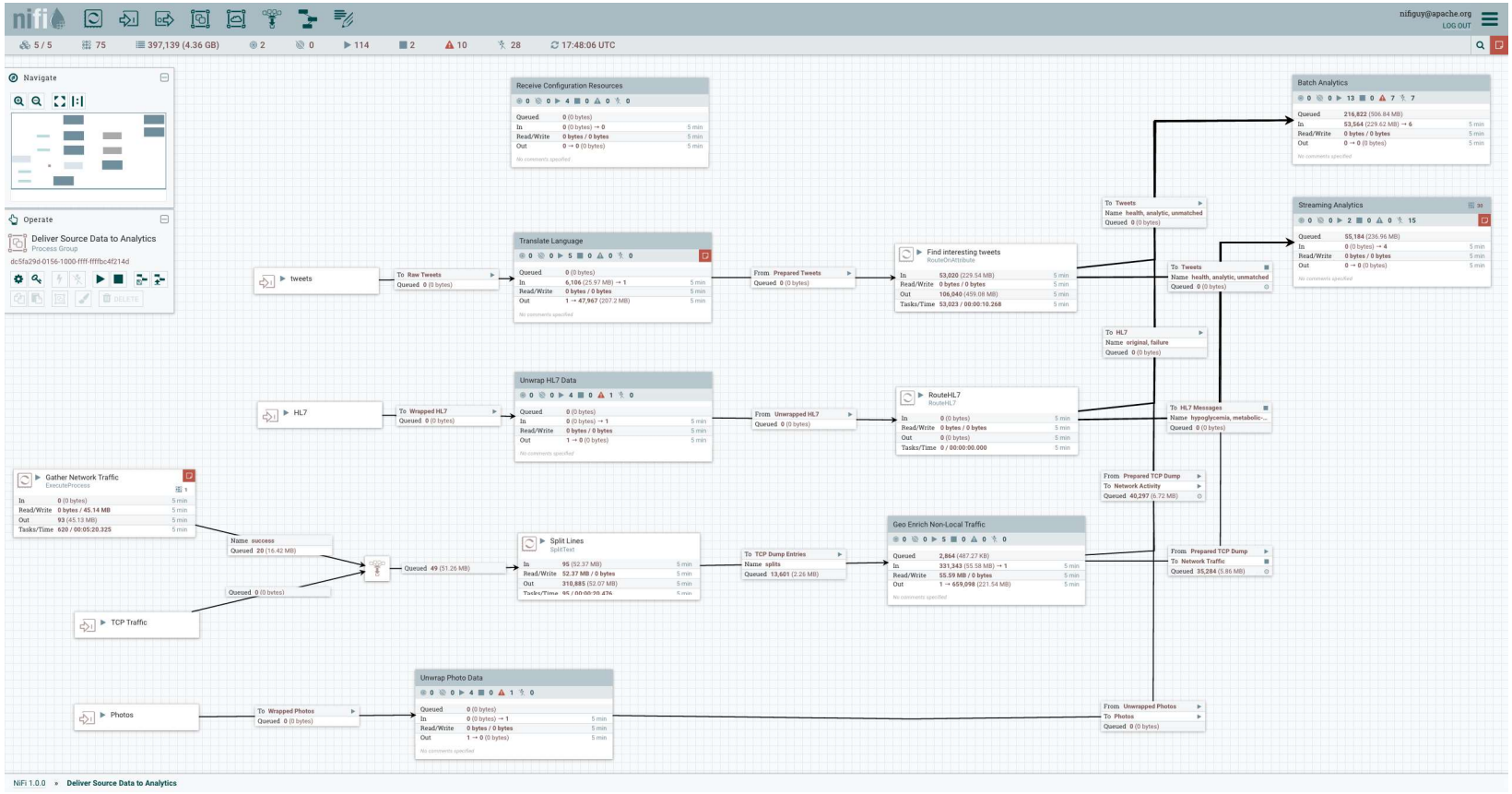
※ Oracle In-Database 프로그램들의 문제점과 BDAE(TM) 의 대책

1. Oracle Database(TM) 의 SGA, PGA 영역 등과는 별개로 관리되기 때문에, Python, R 내부 코드에서 방만하게 많은 메모리를 사용 하게 되면 Oracle Instance 가 있는 서버 자체를 죽일 수 있다. (OS 측면에서 Out of Memory)
2. 실제 Oracle R Enterprise(TM) 운영 시 발생하는 것을 목격했기 때문에, BDAE(TM) 는 하나의 세션 메모리에 대한 최대 가용 메모리를 설정하고, 실행 시 감시하도록 설계했다. (매 10 초마다 해당 세션에서 감시됨.)
3. 또한, Oracle Instance 가 있는 노드의 메모리의 총량이 80 % 이상 일 경우 BDAE(TM) 는 새로운 세션을 거부하도록 되어 있다. 이 부분도 설정할 수 있도록 설계 되어 있다.

Copyright @Raymond. All rights reserved

29 | BDAE(TM) 와 Apache NIFI(TM) 연계

Apache NIFI(TM) 와 BDAE(TM) 를 함께 사용하면 보다 안정적인 Workflow 로 배치 작업을 구성할 수 있다. 특히 파일 기반보다 SQL 기반으로 하면 효과적이며 깔끔하게 정리될 수 있다.



Copyright @Raymond. All rights reserved

BDAE(TM) 활용

Copyright @Raymond. All rights reserved

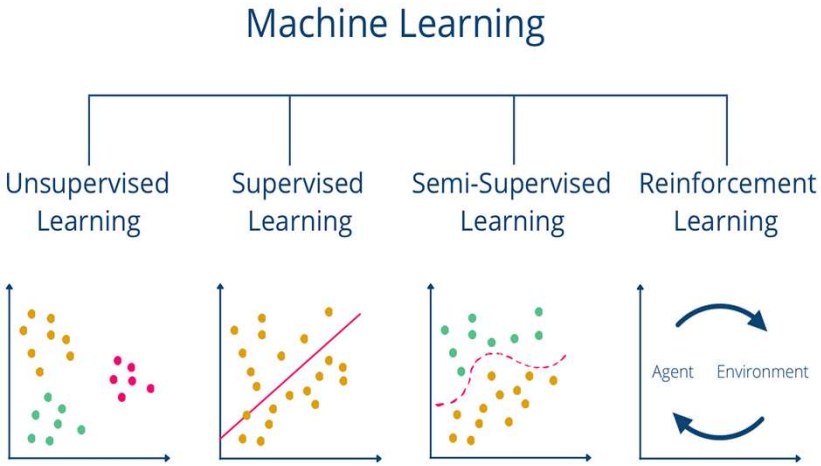
1 | BDAE(TM) 활용 (#1. Python, R 로 수행하는 다양한 분석 업무)

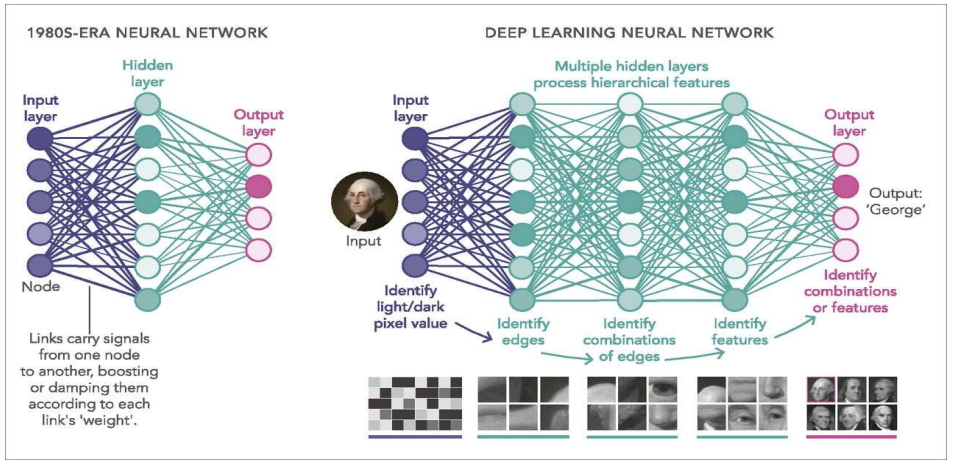
BDAE(TM) 의 최초 개발 목적은 제조 분야의 시스템들, 예를 들면 SPC, FDC, YMS 등에 기존 솔루션에 손쉽게 최신 알고리즘을 적용할 수 있는 방안을 찾는데에 있었다.

특히 패턴 인식을 통한 불량 분석, SPC, 기술 통계량, ANOVA, .. 등을 JAVA 등의 백엔드 등에서 수행하거나, 별도의 고가의 분석 제품을 구입하는 것에 대한 것이 과연 바람직한가 ? 에 대한 의구심.

실제 하이테크 현장에서는 Raw 데이터가 Oracle Database(TM) 이고, R 이나 Python 스크립트를 별도의 어플리케이션 서버에서 돌리는 형태가 보편적이다.

그러나, 별도의 서버에서 알고리즘 소스 관리의 문제, 복잡한 프로토콜 등을 BDAE(TM) 는 해결해 준다.





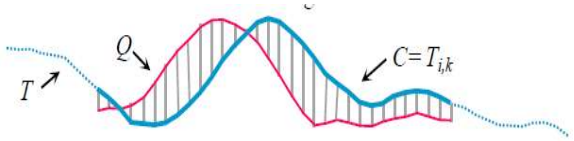
※ Python, R 이 분석 업무 (AI) 에는 가장 적합하며, 생산성이 가장 높은 언어들이다.

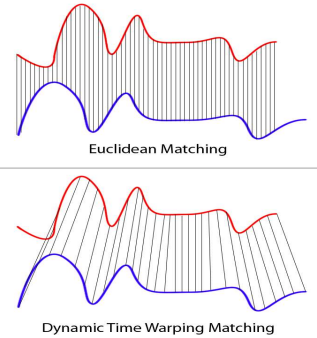
Copyright @Raymond. All rights reserved

2 | BDAE(TM) 활용 (#1. Python, R 로 수행하는 다양한 분석 업무)

실시간의 분석을 빠르게 수행하여 그 결과를 아래와 같은 보편적인 SQL Query 결과로 받는다면 어플리케이션은 매우 간단한 구조가 된다.

※ BDAE(TM) 의 두번째 argument 의 용도는 비교를 위한 Query Data, 추론을 위한 Model 데이타 또는 Python, R 에서 사용하는 함수의 Argument 이다.

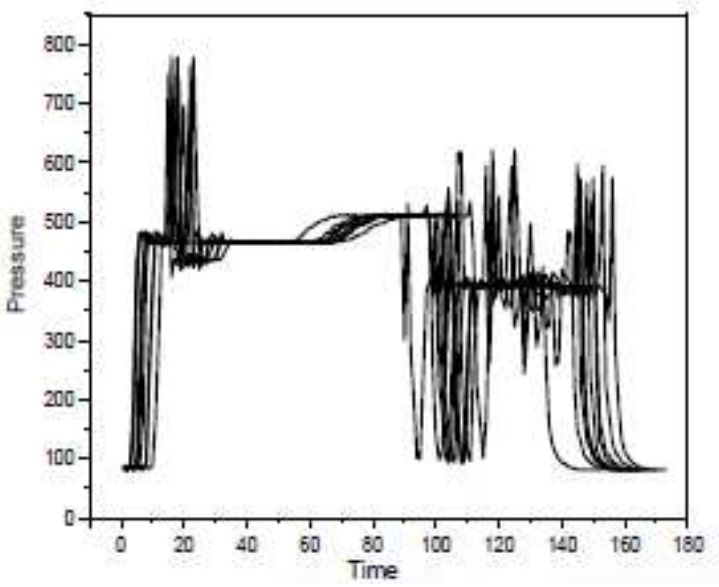





```
SQL> SELECT * /*+ parallel(20) */
  2  FROM TABLE (apGroupEvalParallel (
  3                cursor (
  4                       SELECT *
  5                       FROM TRACE_DATA
  6                       WHERE EQP_ID = 'EPS001'
  7                          AND LOT_ID = 'LOT001'
  8                          AND ETC = '.....'
  9                    ),
 10                cursor(SELECT * FROM GOLDEN_EQUIPMENT ...),
 11                'SELECT  CAST("A" AS VARCHAR2(40)) PARAMETER_ID,
                        1.0 SIMILARITY FROM DUAL',
 11                'EQP_ID, LOT_ID, ....',
 12                'DefectUtil:FastDTW');
```

PARAMETER_ID	SIMILARITY
RF_POWER_1	2.23
O2_PUMP_1	0.5
Ch1_TEMP_1	2.1
……	……
……	……

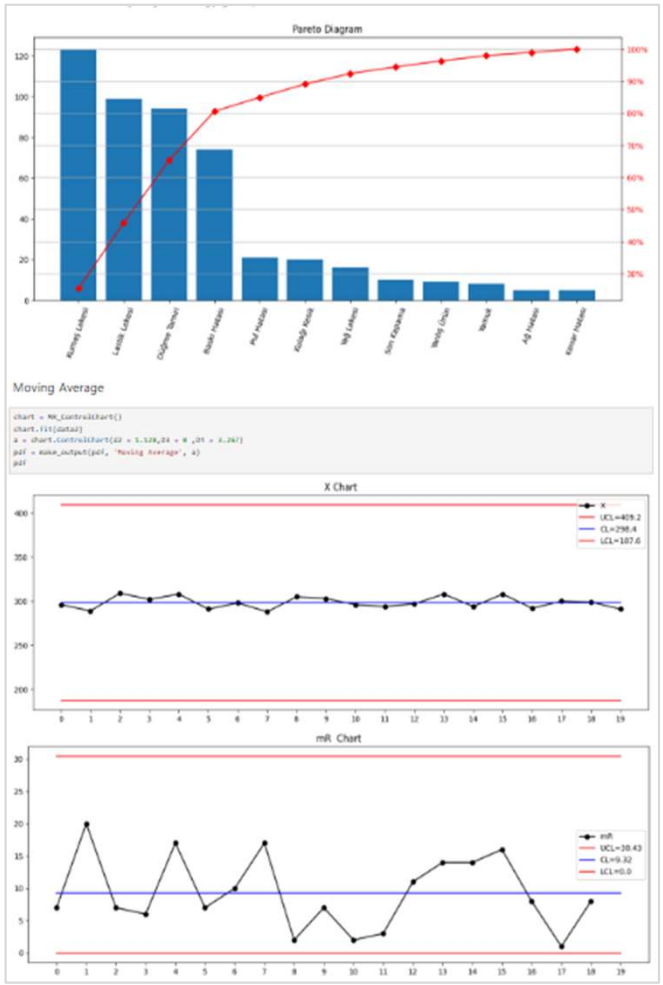
비교가 필요한 알고리즘들

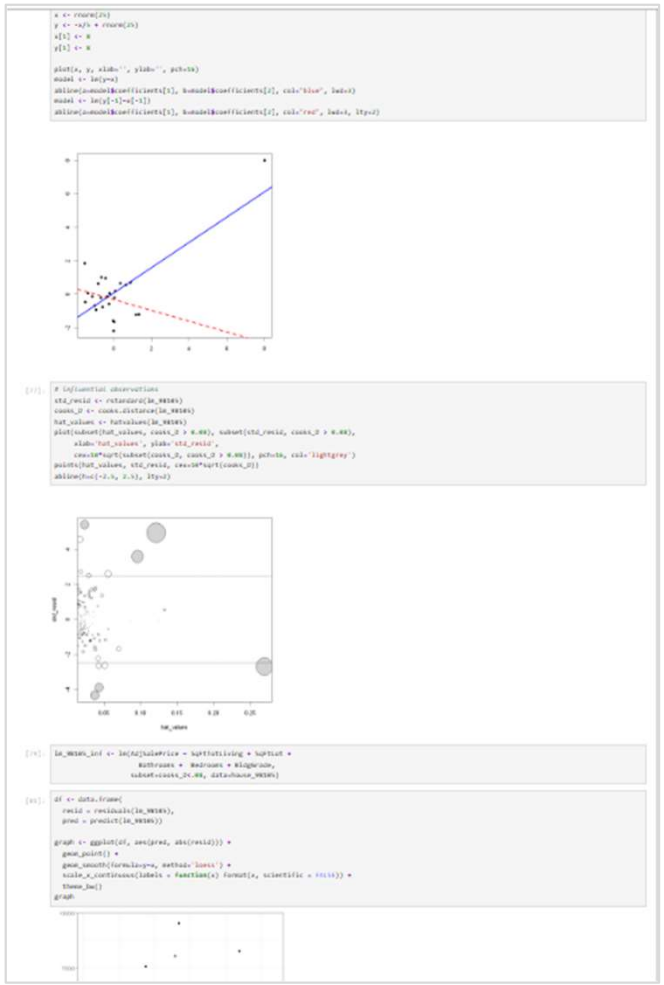


Copyright @Raymond. All rights reserved

3 | BDAE(TM) 활용 (#2. 분석 업무의 배치 작업 및 통합)

앞의 SQL 문을 Python 또는 R 로 Wrapping 하여 클래스로 만들면 다양한 분석 업무의 통합하여 한번에 수행할 수 있지만 병렬 처리가 가능하며, 그 결과 또한 한번에 받을 수 있다.



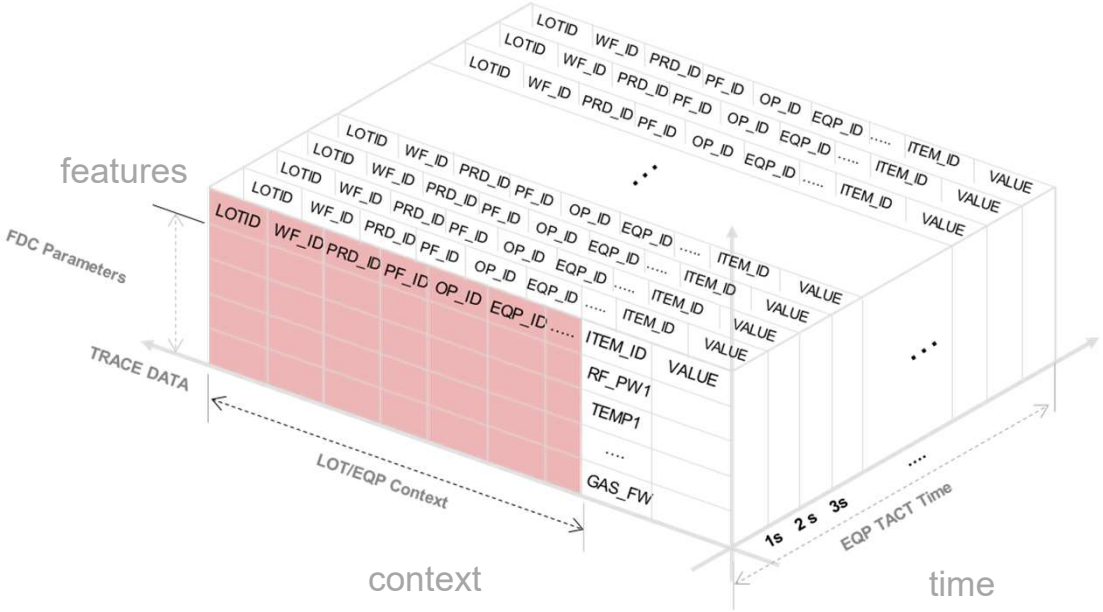


Copyright @Raymond. All rights reserved

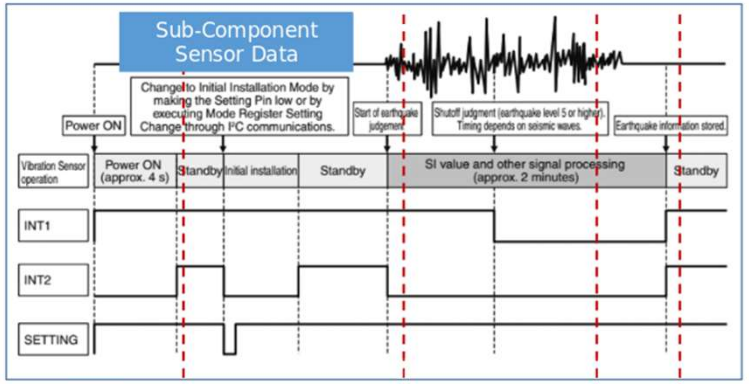
4 | BDAE(TM) 활용 (#4. 병렬 분산 처리)

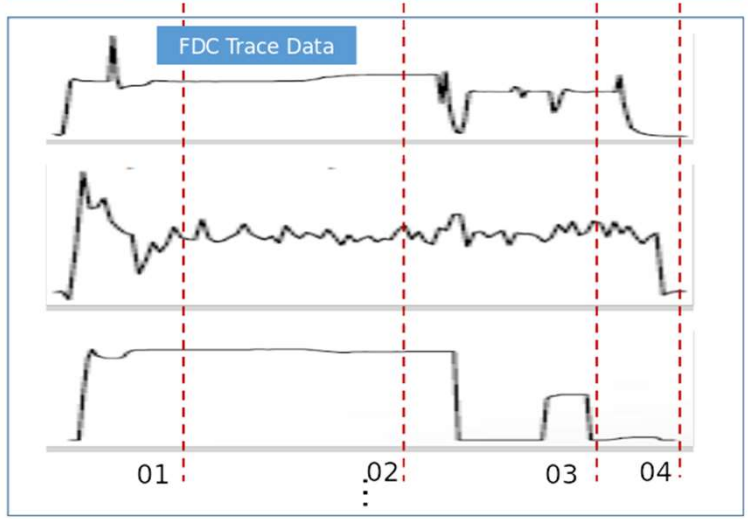
대용량의 테이블을 그룹핑하여 동일한 알고리즘을 한번에 적용하여 결과를 받을 때에는
Python, R 의 병렬 처리를 이용하면 재활용성과 성능, 메모리를 지나치게 많이 사용하게 된다.

이러한 병렬처리 부분은 Oracle In-Database 에 맡기고,
Python, R 모듈은 병렬 처리를 고려하지 않는
단순한 형태를 가지게 하는 것이 BDAE(TM) 의 역할이고
Concept 이며 위의 문제점들은 해결된다.



Trace Data per 1 LOT/1 EQP





Copyright @Raymond. All rights reserved

4 | BDAE(TM) 활용 (#4. 병렬 분산 처리)

다음의 Query 는 대용량 테이블의 병렬 분산 처리를 빨간 글씨의 그룹 컬럼들의 단위로 수행하는 예제이다. Oracle Hint (/*+ parallel(20) */) 부분으로 총 20개로 나누어서 병렬 처리 되며, 그 때 Python 모듈은 병렬 처리를 고려하지 않은 단순한 모듈 형태를 띄게 된다.
BDAE(TM) 는 이를 가능하게 만들어 준다.

```
SELECT /*+ parallel(20) */*
    FROM table(apGroupEvalParallel(
        CURSOR(
            WITH TARGET_TBLE AS
            (
                  SELECT * FROM FDC_TRACE
                  WHERE 1=1
                    AND EQP_ID='EQP-200'
                    AND UNIT_ID='UNIT-02'
            )
           SELECT EQP_ID, UNIT_ID, LOT_ID, WAFER_ID, RECIPE, PARAM_ID,
              (VALUE - (AVG(VALUE) OVER (PARTITION BY PARAM_ID)))
             / (STDDEV(VALUE) OVER (PARTITION BY PARAM_ID)) AS  NORMALIZED_VALUE
                                FROM TARGET_TBLE
        ) ,
        NULL,
         'SELECT CAST(''A'' AS VARCHAR2(40)) EQP_ID,
                CAST(''A'' AS VARCHAR2(40)) UNIT_ID,
                CAST(''A'' AS VARCHAR2(40)) LOT_ID,
                CAST(''A'' AS VARCHAR2(40)) WAFER_ID,
                CAST(''A'' AS VARCHAR2(40)) RECIPE,
                CAST(''A'' AS VARCHAR2(40)) RESULT
                FROM DUAL',
          'EQP_ID,UNIT_ID,LOT_ID,WAFER_ID,RECIPE,PARAM_ID',
         'Standardization:normalize'));
```

Copyright @Raymond. All rights reserved

5 | BDAE(TM) 활용 (#5. 전처리)

많은 입력 데이터 부분의 전처리를 Python, R 이 수행하면 메모리, 성능의 문제가 발생한다.
BDAE(TM) 는 다음과 같은 복잡한 SQL 문을 수용하여 전처리 할 수 있게 하며 이는 Oracle Database(TM) 가 가장 앞선 SQL 엔진을 가지고 있기 때문이다.

타사의 DB 나 오픈소스, 하둡 에코 시스템 (Hive, Phoenix, Impala, . . .) 등은 ANSI-SQL 의 Subset 이기 때문에 이러한 고도화된 전처리를 할 수 없다.

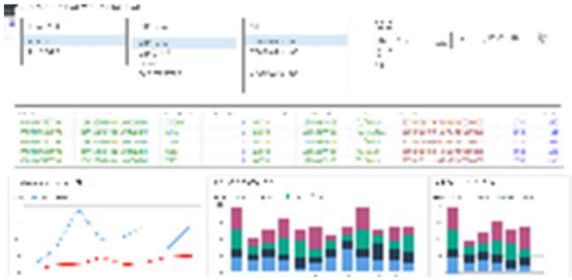
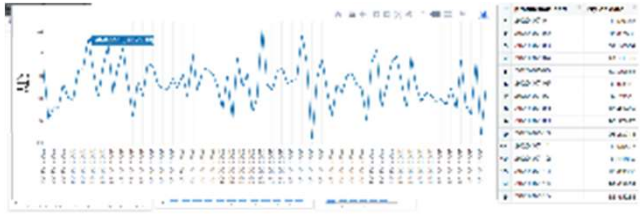
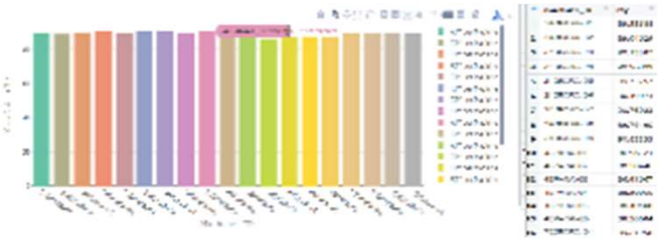
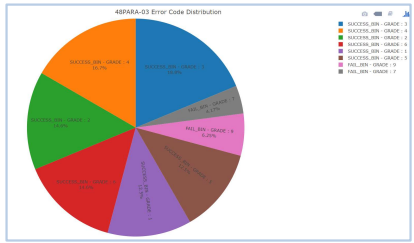
```
SELECT  eqp_id, recipe_id, …, time, parameter_name, sma, ema
FROM    (
            SELECT eqp_id, recipe_id, time, …, parameter_name, parameter_value
            FROM  trace_data
            WHERE 1=1
                AND time between … and …
                AND step_seq = ' …'
                …
            ) a
MODEL
   PARTITION BY (a.parameter_name, 2 / (1 + count(*) over (partition by a.parameter_name))
                smoothing_constant )
   DIMENSION BY (row_number() over (partition by a.parameter_name order by a.time) rn)
   MEASURES (a.time, a.parameter_value, sma, 0 ema)
   (
       ema[1] = a.parameter_value[1],
       ema[rn > 1] order by rn = ( cv(smoothing_constant) * (parameter_value[cv()] - ema[cv() -1]) ) + ema[cv() -
1]
   )
ORDER BY eqp_id, a.recipe_id, …, a.time, a.parameter_name;
```

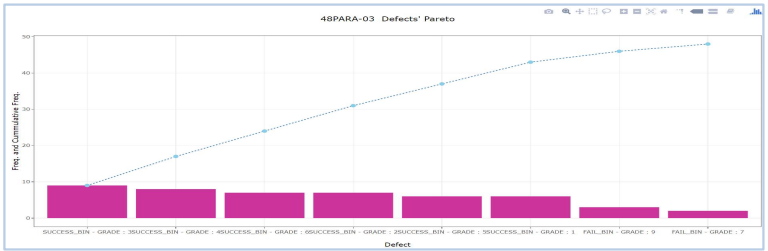

Copyright @Raymond. All rights reserved

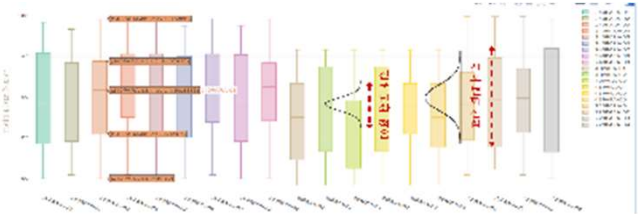
6 | BDAE(TM) 활용 (#6. Smart Factory 분야)

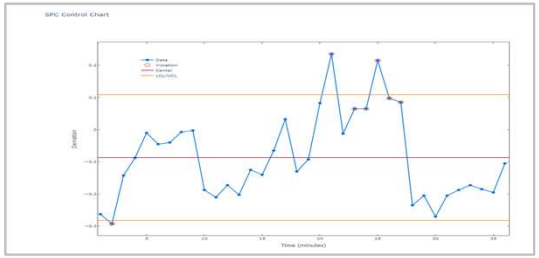
통계적 품질 분석이나, 이상 감지, SPC, . . . 등의 기존 솔루션 기반에서, Python 과 R 이 제공하는 각종 패키지들을 BDAE(TM) 에서 사용하면 별도 비용 없이 유연한 기능들을 SQL 문으로 손쉽게 기존 솔루션에 Embedding 할 수 있다.

※ BDAE(TM) 는 SQL 문으로, 메모리 상에서 수행되기 때문에 어떠한 솔루션에도 추가 될 수 있다.





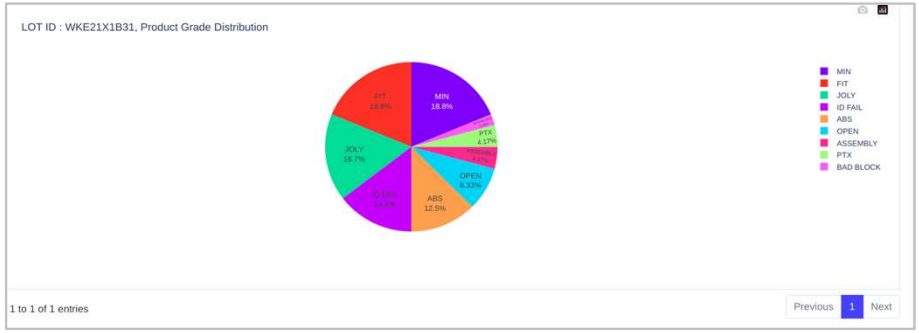



Copyright @Raymond. All rights reserved

6 | BDAE(TM) 활용 (#6. Smart Factory 분야)

다음과 같은 LOT 에 포함된 Product 의 분류 및 시각화에는 단지 5 줄의 분석 코드와 SQL 문이 필요할 뿐이다.

```
WITH LOT_SUM_ONE AS (
  SELECT LOT_ID,
   CASE WHEN PROD_GRADE = 9 THEN 'BAD BLOCK'
         WHEN PROD_GRADE = 8 THEN 'ASSEMBLY'
         WHEN PROD_GRADE = 7 THEN 'PTX'
         WHEN PROD_GRADE = 6 THEN 'ID FAIL'
         WHEN PROD_GRADE = 5 THEN 'OPEN'
         WHEN PROD_GRADE = 4 THEN 'ABS'
         WHEN PROD_GRADE = 3 THEN 'JOLY'
         WHEN PROD_GRADE = 2 THEN 'FIT'
         WHEN PROD_GRADE = 1 THEN 'MIN'
         ELSE
           'N/A'
      END PROD_GRADE_DESC
   , COUNT(PROD_GRADE) CNT FROM table (
     productExplodeEvalCLob(cursor(
       SELECT *
       FROM LOT_SUM
       WHERE LOT_ID = 'WKE21X1B31'
            AND MACHINE_ID = '48PARA-03' ...DURABLE_ID = 'Z718')))
       GROUP BY LOT_ID, PROD_GRADE
)
SELECT *
 FROM table(apTableEval(
  cursor(
    SELECT * FROM LOT_SUM_ONE
  ),
  NULL,
  'XML',
  'LOTSUM_ERR_PIE:display'))
```

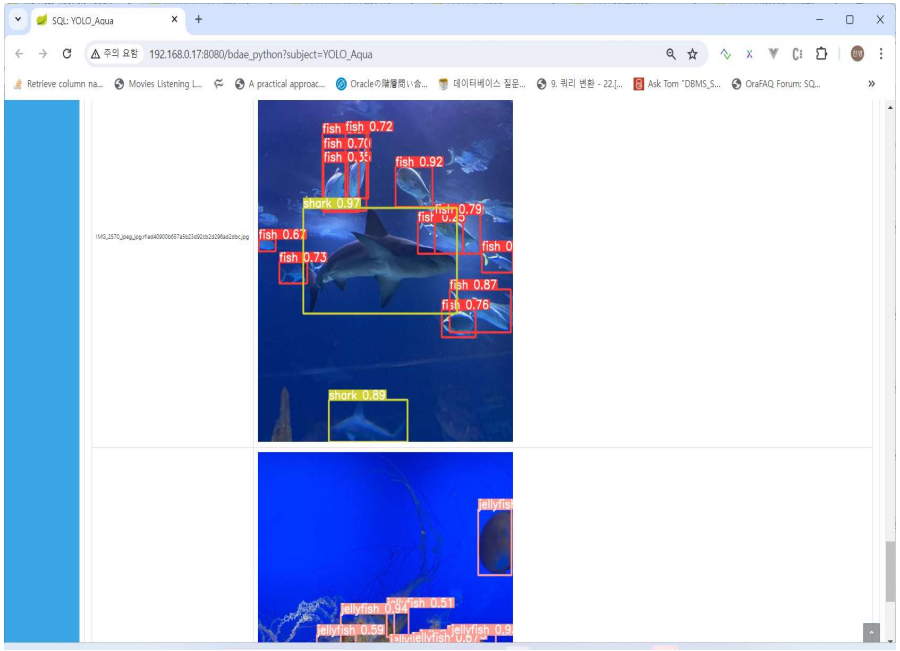


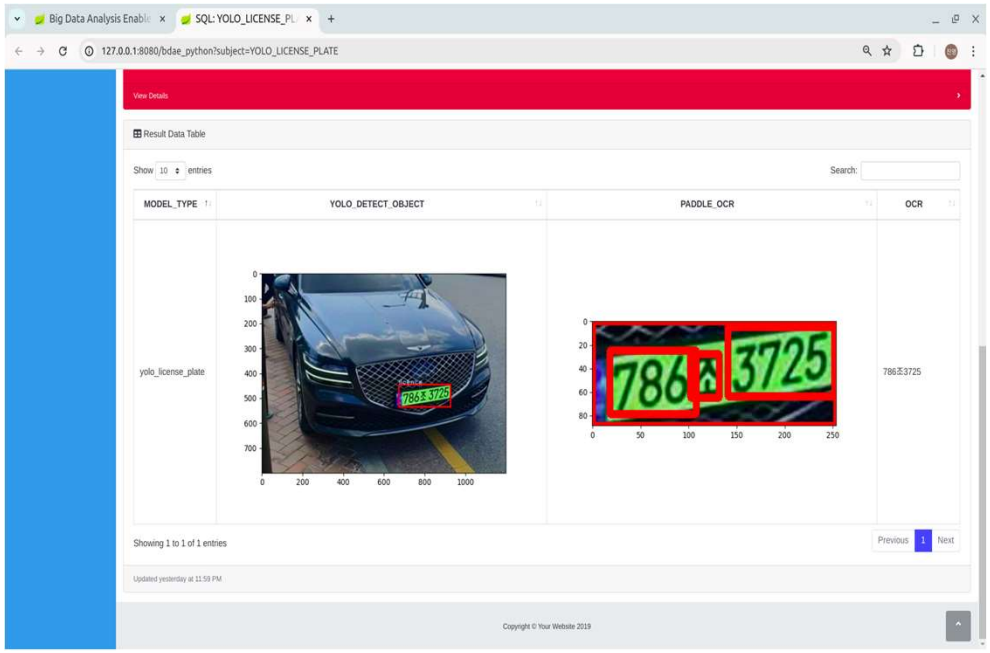
Copyright @Raymond. All rights reserved

7 | BDAE(TM) 활용 (#7. Deep Learning 추론)

비용이나 GPU 에 의존적인 알고리즘의 학습을 위해 BDAE(TM) 가 설치되어 있지 않은 곳에서 Model 을 학습하고 그것을 실시간에 BDAE(TM) 를 이용하여 추론에 사용할 수 있다.
(설비 별, 제품 별 Model 이 다를 수 있기 때문이며 BDAE(TM) 는 DeSerialization 을 지원한다.)

※ 시계열 데이터 (1D Tensor) 를 이미지로 변환 후 이것을 Deep Learning 이미지 합성곱으로 불량 검증을 하는 것이 더 효과적이라는 논문이 있고, 이를 실제로 활용하는 경우가 있음. BDAE(TM) 는 배치작업으로 많은 이미지를 한번에 생성할 수 있다.





Copyright @Raymond. All rights reserved

8 Step by Step

BDAE(TM) 의 실행은 SQL 문으로 시작되며, 이는 Oracle 세션 기반의 어떠한 DB 툴이나 Application 에서 호출될 수 있다.

웹 개발에서 일반적인 백엔드의 역할 보다 그 이상을 할 수 있는 것이 BDAE(TM) 이다.
데이터의 이동 없이, 단 한번의 호출에서 여러 테이블의 형상들의 변형과 단독 SQL 문으로 할 수 없는 것을 BDAE(TM) 는 할 수 있다.

BDAE(TM) 로 이미지, 문서 등의 어떠한 것도 만들 수 있고 이를 SQL 문의 결과로 전달할 수 있고 BLOB 에 저장된 모델을 DeSerialization 하여 추론을 하거나, 인덱싱을 별도로 가져갈 수도 있다.

Oracle In-Database 는 최초에는 검색 엔진으로 자주 활용되었었다. (1990년대 이야기)

본 8 절은 Python, R 모듈과 이를 호출하는 BDAE(TM) SQL 을 만드는 과정을 기술한다.

Copyright @Raymond. All rights reserved

8 R ML Example (#1 함수의 선택)

R 은 4개의 함수가 제공된다. 가장 많이 사용되는 것은 asTableEval() 과 asGroupEval() 이 될 것이다. asEval() 은 테스트 목적으로 사용될 수 있으며 asRowEval 은 정의된 Row 수 마다 무엇을 할 때 유용하다.

이 함수들의 Argument 는 거의 비슷하며 직관적이다.

※ Python 함수는 as 대신 ap 로 되어 있고 정확하게 기능이 동일하며 역시 4개의 함수를 제공한다.

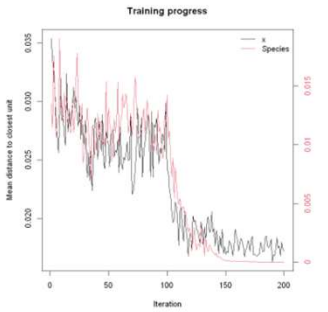
SQL Interface function	Description
asEval()	데이터를 R 모듈에서 자체적으로 공급할 때
asTableEval()	데이터를 BDAE 가 전체를 줄 때
asRowEval()	데이터를 BDAE 가 Row 수 만큼 나누어 줄 때
asGroupEval()	데이터를 BDAE 가 Grouping 하여 공급할 때

Copyright @Raymond. All rights reserved

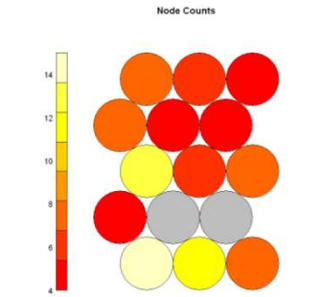
8 | R ML Example (#2 개발 단계)

분석가들은 개발 단계에서는 R studio 나 Jupyter notebook 으로 Interactive 하게 개발한다.

```
[1]: options(warn=-1)
[3]: library(kohonen)
[3]: train  <- sample(1:150, 100)
[5]: train_set  <-  list(x = as.matrix(iris[train,-5]), Species = as.factor(iris[train,5]))
[7]: test_Set <- list(x = as.matrix(iris[-train,-5]), Species = as.factor(iris[-train,5]))
[9]: gr <- somgrid(xdim = 3, ydim = 5, topo = "hexagonal") #grid 갯수 및 모양 설정
     ss <- supersom(train_set, gr, rlen = 200, alpha = c(0.05, 0.01)) #som 학습하기
[25]: str(ss)
List of 14
 $ data           :List of 2
  ..$ x      : num [1:100, 1:4] 5.5 4.6 5.8 5.5 6.3 4.9 5.4 5.2 5.2 7.4 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:100] "34" "48" "143" "54" ...
  .. .. ..$ : chr [1:4] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"
  ..$ Species: num [1:100, 1:3] 1 1 0 0 0 1 1 1 1 0 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:3] "setosa" "versicolor" "virginica"
 $ unit.classif   : num [1:100] 3 2 15 7 11 1 3 1 1 13 ...
 $ distances      : num [1:100] 0.003903 0.000746 0.004022 0.009909 0.006318 ...
 $ grid           :List of 6
  ..$ pts             : num [1:15, 1:2] 1.5 2.5 3.5 1 2 3 1.5 2.5 3.5 1 ...
  .. ..- attr(*, "dimnames")=List of 2
 $ dist.fcts      : chr [1:2] "sumofsquares" "tanimoto"
 - attr(*, "class")= chr "kohonen"
[11]: plot(ss,type="changes")
```

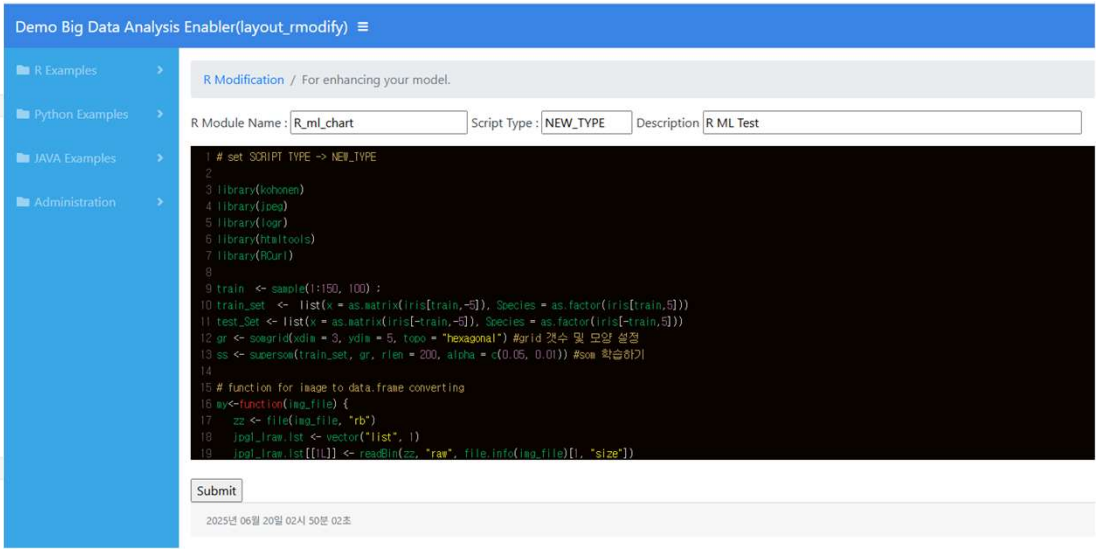



```
[13]: plot(ss, type="count", main="Node Counts")
```



이 개발 작업을 끝낸 뒤에 BDAE(TM) 에서 돌리려면

1) 소스를 정리해서 옮긴다. (Copy & Paste)
   BDAE(TM) Web Editor 에 붙여 넣기를 하면 된다.
2) R 모듈 이름을 정한다.
3) 저장한다.



※ 정의된 R Module 은 재 활용될 수 있다.
※ Python Module 도 비슷하지만 함수명도 넣어야 한다.

Copyright @Raymond. All rights reserved

8 R ML Example (#2 개발 단계)

2) 개발 단계의 R 소스를 BDAE(TM) 로 옮길 때 출력을 고려해야 한다. 결국 Table 형태로 출력 되므로, 모든 데이터는 R 의 data.frame 형태로 변환되어 전달 되어야 한다. 따라서 약간의 소스 수정은 필요하지만, 이 부분은 정형적이므로 재활용 함수들을 만들어 사용하면 어려움이 없을 것이다. 이미지를 data.frame 에 모두 담기 위해서 함수(my)를 하나 만들어서 삽입한다.

```
my<-function(img_file) {
  zz <- file(img_file, "rb")
  jpg1_lraw.lst <- vector("list", 1)
  jpg1_lraw.lst[[1L]] <- readBin(zz, "raw", file.info(img_file)[1, "size"])
  close(zz)
  df <- data.frame(name=img_file,stringsAsFactors=FALSE)
  df$blob <- jpg1_lraw.lst
  unlink(img_file)
  return (df)
}

name <-c("/tmp/kohonen01.jpg","/tmp/kohonen02.jpg","/tmp/kohonen03.jpg","/tmp/kohonen04.jpg", "/tmp/kohonen05.jpg")

jpeg(name[1])
plot(ss,type="changes")
dev.off()
jpeg(name[2])
plot(ss, type="count", main="Node Counts")
dev.off()
```


Copyright @Raymond. All rights reserved

8 R ML Example (#2 개발 단계)

3) rbind 로 각각의 data.frame 을 Row 기반으로 합친 후 최종 data.frame 명을 적어둔다.

```
df = my(name[1])
for (i in 2:length(name)) {
   df_tmp = my(name[i])
   log_print(name[i])
   df = rbind(df, df_tmp)
}

df
```

4) 위의 소스를 BDAE(TM) Web Editor 을 이용하여 이름을 R_ml_chart 로 저장한다. (Copy & Paste)

5) 호출을 위해서는 이제 BDAE(TM) SQL 을 만들어야 한다.
가장 신경 써야 하는 건 출력 부분이다.

앞서 만든 data.frame 은 파일명과 이미지 바이너리로 되어 있으므로 BDAE(TM) 의 출력은

```
SELECT CAST(NULL AS VARCHAR2(40) Name, TO_BLOB) IMG FROM DUAL
```

로 간단히 만들 수 있다. 복잡한 data.frame 의 경우에 이 문서에 있는 출력 함수를 사용하면 된다.
(generate_dual_select)

Copyright @Raymond. All rights reserved

8 R ML Example (#3 BDAE SQL 생성 단계)

최종적으로 asEval() 을 사용하는 SQL 은 다음과 같다.

1) 함수 4 개 중에 데이터를 BDAE(TM) 로부터 받지 않으니, **asEval()** 을 선택 하고,
2) 부가적인 데이터 (예를 들면 R 을 위한 argument 등) 가 없으니 **NULL**,
3) 출력 부분은 이미지 이름과 이미지 바이너리(JPG) 이므로 빨간 색,
4) 마지막으로 알고리즘이 담긴 R 모듈 이름인 R_ml_chart 을 넣는다.

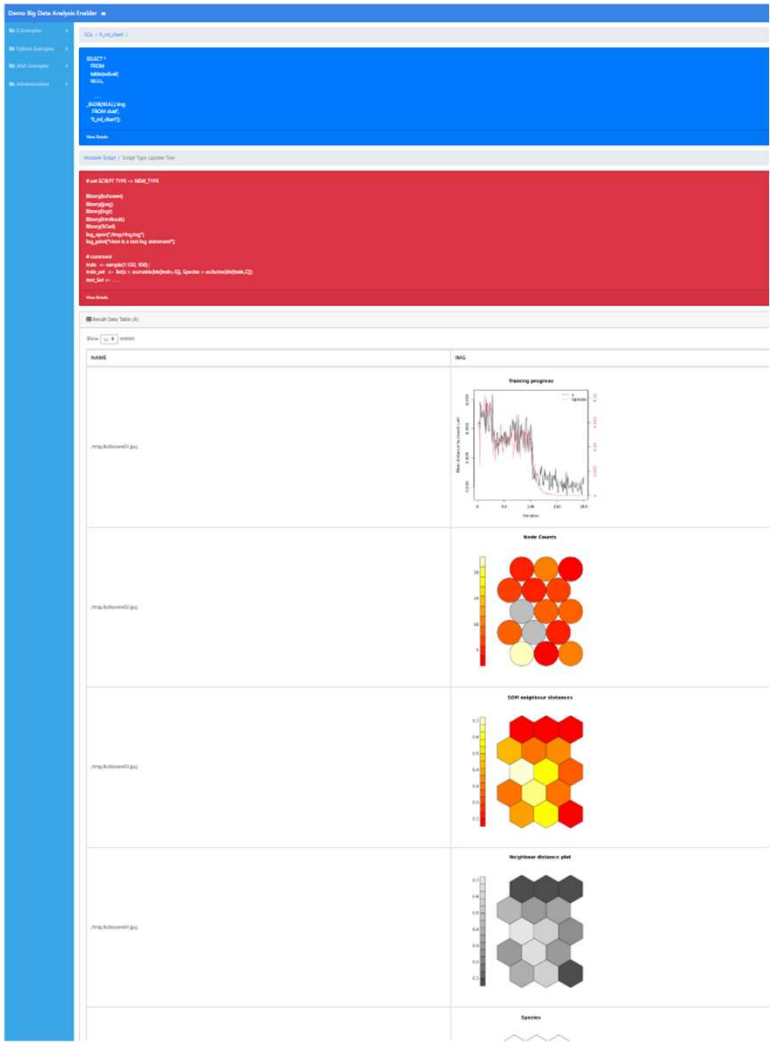
```
SELECT *
  FROM
  table(asEval(
  NULL,
  'SELECT CAST(NULL AS VARCHAR2(40)) name,  TO_BLOB(NULL) img FROM dual',
  'R_ml_chart'))
```

※ Python 의 경우 P_ml_chart:display 등의 start 함수명을 : 로 분리해 넣어주어야 한다.
※ asEval 은 Python 의 경우 apEval 이며 동일한 argument 수를 가진다.

Copyright @Raymond. All rights reserved

8 R ML Example (#4 결과 확인)

BDAE(TM) Web 으로 실행 결과는 다음과 같다.



◁ 호출된 SQL 부분

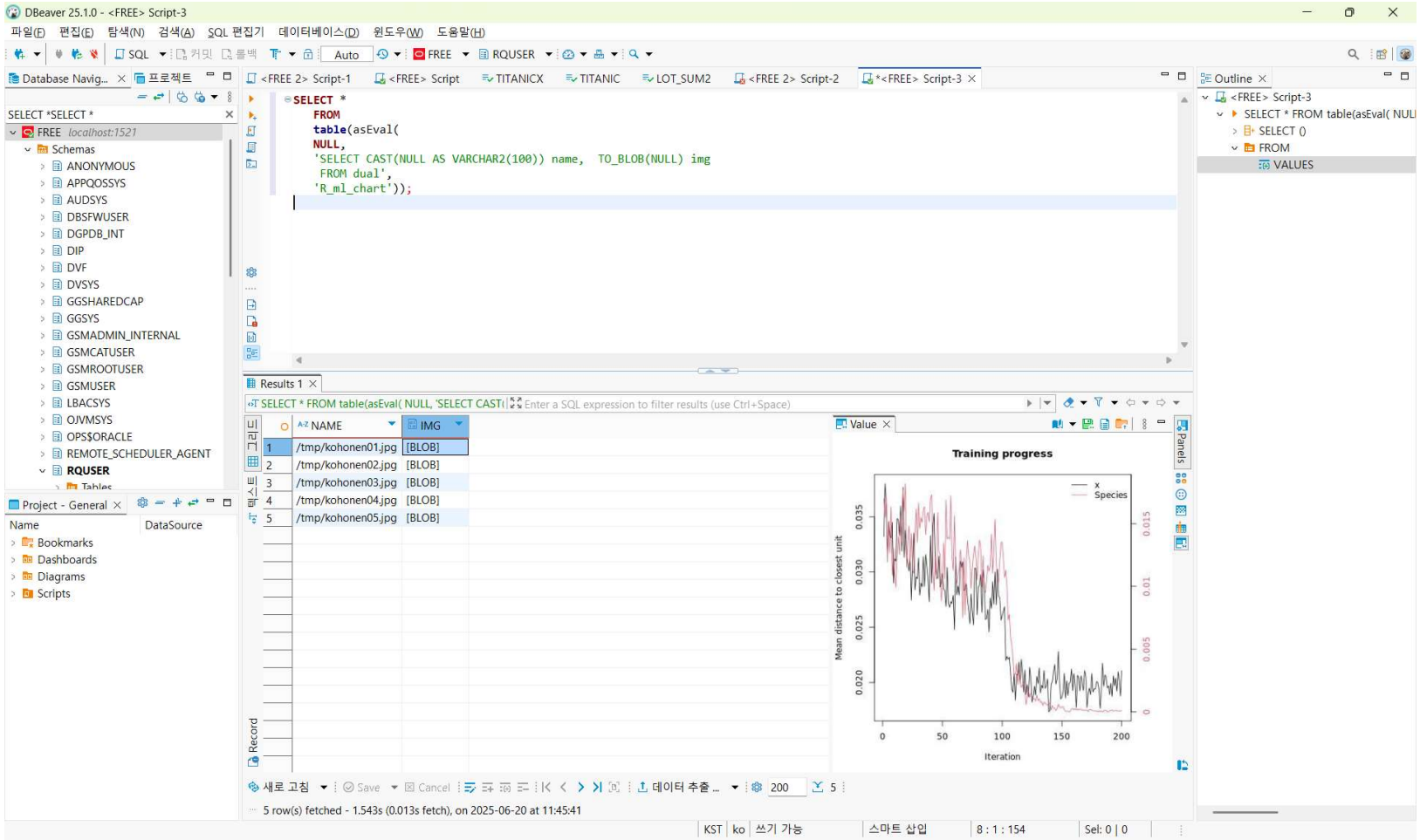
◁ 사용된 R 모듈 부분

◁ SQL 의 결과

Copyright @Raymond. All rights reserved

8 R ML Example (#4 결과 확인)

DBeaver(TM) 툴로 SQL 을 호출하면 다음과 같은 결과를 보여 준다.
이미지 임을 알고 DBeaver(TM) 툴에서도 차트 이미지를 바로 볼 수 있다.



Copyright @Raymond. All rights reserved

8 Python 의 경우

Python 도 R 과 크게 다르지 않다.

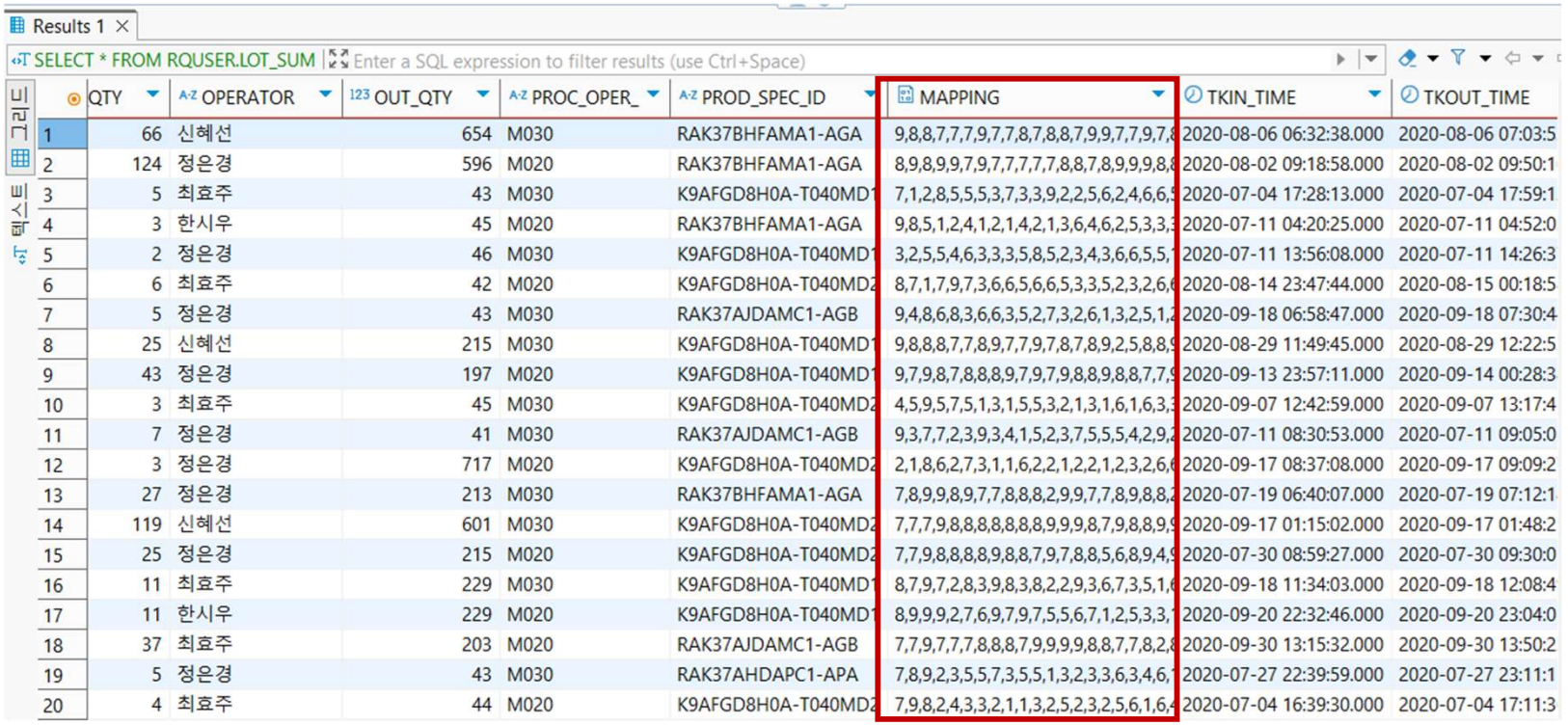
R 은 데이터 해석에 탁월한 언어이고, Python 은 빠른 성능과 GPU 까지 사용하여 AI 의 주력 언어로 이미 자리 잡았다.

BDAE(TM) 입장에서도 Python 의 예외 처리 기법이 좋아서 권장된다.

Copyright @Raymond. All rights reserved

8 Python Smart Factory 관련 Example (#1 테이블 구성)

LOT_SUM 테이블은 원래 아래와 같은 모습이며, 특별한 것은 MAPPING 컬럼의 존재이다.
LOT 안의 PRODUCT 들의 Grade 에 대한 부분이 CLOB 컬럼으로 존재한다.



Results 1

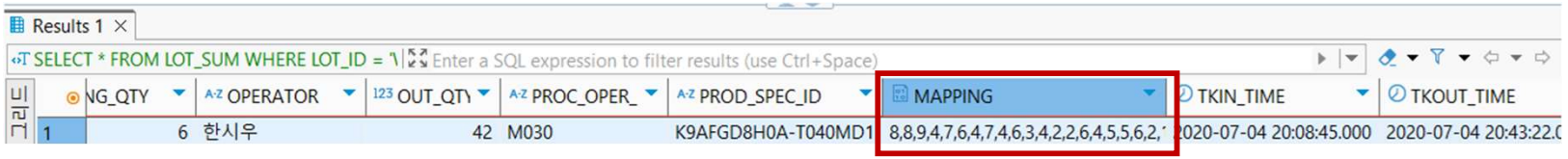
```
SELECT * FROM RQUSER.LOT_SUM
```

Enter a SQL expression to filter results (use Ctrl+Space)

그리드 / 텍스트

	QTY	OPERATOR	OUT_QTY	PROC_OPER_	PROD_SPEC_ID	MAPPING	TKIN_TIME	TKOUT_TIME
1	66	신혜선	654	M030	RAK37BHFAMA1-AGA	9,8,8,7,7,7,9,7,7,8,7,8,8,7,9,9,7,7,9,7,8	2020-08-06 06:32:38.000	2020-08-06 07:03:5
2	124	정은경	596	M020	RAK37BHFAMA1-AGA	8,9,8,9,9,7,9,7,7,7,7,7,8,8,7,8,9,9,9,9,8,8	2020-08-02 09:18:58.000	2020-08-02 09:50:1
3	5	최효주	43	M030	K9AFGD8H0A-T040MD1	7,1,2,8,5,5,5,3,7,3,3,9,2,2,5,6,2,4,6,6,5	2020-07-04 17:28:13.000	2020-07-04 17:59:1
4	3	한시우	45	M020	RAK37BHFAMA1-AGA	9,8,5,1,2,4,1,2,1,4,2,1,3,6,4,6,2,5,3,3,3	2020-07-11 04:20:25.000	2020-07-11 04:52:0
5	2	정은경	46	M030	K9AFGD8H0A-T040MD1	3,2,5,5,4,6,3,3,3,5,8,5,2,3,4,3,6,6,5,5,1	2020-07-11 13:56:08.000	2020-07-11 14:26:3
6	6	최효주	42	M020	K9AFGD8H0A-T040MD2	8,7,1,7,9,7,3,6,6,5,6,6,5,3,3,5,2,3,2,6,6	2020-08-14 23:47:44.000	2020-08-15 00:18:5
7	5	정은경	43	M030	RAK37AJDAMC1-AGB	9,4,8,6,8,3,6,6,3,5,2,7,3,2,6,1,3,2,5,1,2	2020-09-18 06:58:47.000	2020-09-18 07:30:4
8	25	신혜선	215	M030	K9AFGD8H0A-T040MD1	9,8,8,8,7,7,8,9,7,7,9,7,8,7,8,9,2,5,8,8,9	2020-08-29 11:49:45.000	2020-08-29 12:22:5
9	43	정은경	197	M020	K9AFGD8H0A-T040MD1	9,7,9,8,7,8,8,8,9,7,9,7,9,8,8,9,8,8,7,7,9	2020-09-13 23:57:11.000	2020-09-14 00:28:3
10	3	최효주	45	M030	K9AFGD8H0A-T040MD2	4,5,9,5,7,5,1,3,1,5,5,3,2,1,3,1,6,1,6,3,3	2020-09-07 12:42:59.000	2020-09-07 13:17:4
11	7	정은경	41	M030	RAK37AJDAMC1-AGB	9,3,7,7,2,3,9,3,4,1,5,2,3,7,5,5,5,4,2,9,2	2020-07-11 08:30:53.000	2020-07-11 09:05:0
12	3	정은경	717	M020	K9AFGD8H0A-T040MD2	2,1,8,6,2,7,3,1,1,6,2,2,1,2,2,1,2,3,2,6,6	2020-09-17 08:37:08.000	2020-09-17 09:09:2
13	27	정은경	213	M030	RAK37BHFAMA1-AGA	7,8,9,9,8,9,7,7,8,8,8,2,9,9,7,7,8,9,8,8,2	2020-07-19 06:40:07.000	2020-07-19 07:12:1
14	119	신혜선	601	M030	K9AFGD8H0A-T040MD2	7,7,7,9,8,8,8,8,8,8,8,9,9,9,8,7,9,8,8,9,9	2020-09-17 01:15:02.000	2020-09-17 01:48:2
15	25	정은경	215	M020	K9AFGD8H0A-T040MD2	7,7,9,8,8,8,8,9,8,8,7,9,7,8,8,5,6,8,9,4,9	2020-07-30 08:59:27.000	2020-07-30 09:30:0
16	11	최효주	229	M030	K9AFGD8H0A-T040MD1	8,7,9,7,2,8,3,9,8,3,8,2,2,9,3,6,7,3,5,1,6	2020-09-18 11:34:03.000	2020-09-18 12:08:4
17	11	한시우	229	M020	K9AFGD8H0A-T040MD1	8,9,9,9,2,7,6,9,7,9,7,5,5,6,7,1,2,5,3,3,1	2020-09-20 22:32:46.000	2020-09-20 23:04:0
18	37	최효주	203	M020	RAK37AJDAMC1-AGB	7,7,9,7,7,7,8,8,8,7,9,9,9,9,8,8,7,7,8,2,8	2020-09-30 13:15:32.000	2020-09-30 13:50:2
19	5	정은경	43	M030	RAK37AHDAPC1-APA	7,8,9,2,3,5,5,7,3,5,5,1,3,2,3,3,6,3,4,6,1	2020-07-27 22:39:59.000	2020-07-27 23:11:1
20	4	최효주	44	M020	K9AFGD8H0A-T040MD2	7,9,8,2,4,3,3,2,1,1,3,2,5,2,3,2,5,6,1,6,4	2020-07-04 16:39:30.000	2020-07-04 17:11:3

아래 특정 LOT 에 대한 Grade 의 PIE 차트를 그린다고 가정해 보자.



Results 1

```
SELECT * FROM LOT_SUM WHERE LOT_ID = '\
```

Enter a SQL expression to filter results (use Ctrl+Space)

그리드

	NG_QTY	OPERATOR	OUT_QTY	PROC_OPER_	PROD_SPEC_ID	MAPPING	TKIN_TIME	TKOUT_TIME
1	6	한시우	42	M030	K9AFGD8H0A-T040MD1	8,8,9,4,7,6,4,7,4,6,3,4,2,2,6,4,5,5,6,2,'	2020-07-04 20:08:45.000	2020-07-04 20:43:22.0

Copyright @Raymond. All rights reserved

8 Python Smart Factory 관련 Example (#2 모듈의 작성)

Python 으로 Pie Chat 를 그리는 것은 매우 단순하다. 단 2 ~ 3 줄이면 끝난다.
(R 로 해도 별반 차이가 없다. R 도 동일한 패키지를 제공하기 때문이다.)

필요한 것은 해당 LOT 의 정보가 df 에 담겨 넘어와야 한다. 이 Pie Chat 는 JavaScript 로 자동 생성되기 때문에 Interactive 하다. (Mouse 가 움직이면 ..)

다만, 웹 서버 (백엔드)를 위해서 형태로 넘겨 주고 있다.

```
import pandas as pd
import plotly.express as px
import plotly.offline as py

def display(df):
    fig = px.pie(df, values='CNT', names='PROD_GRADE_DESC', title='LOT ID : ' + df['LOT_ID'][0] + ', Product Grade Distribution')
    fig.update_traces(textposition='inside', textinfo='percent+label')
    total = py.offline.plot(fig, output_type='div')
return (total)
```

자, 이제 BDAE Web Editor 를 이용하여 모듈명을 LOTSUM_Error_Pie, 함수명을 display 로 저장하자.

Copyright @Raymond. All rights reserved

8 Python Smart Factory 관련 Example (#3 데이터 탐색)

전처리가 필요하다. 원래 LOT_SUM 의 Mapping 은 CLOB 컬럼이며 이를 Row 기반으로 모두 풀어야 하기 때문이다. 그래서 다음과 같은 모습으로 해야 차트를 그리기 쉽다.

	lot_id	prod_grade_desc	cnt
0	WKE21X1B31	ASSEMBLY	2
1	WKE21X1B31	BAD BLOCK	1
2	WKE21X1B31	ABS	6
3	WKE21X1B31	PTX	2
4	WKE21X1B31	ID FAIL	7
5	WKE21X1B31	JOLY	8
6	WKE21X1B31	FIT	9
7	WKE21X1B31	OPEN	4
8	WKE21X1B31	MIN	9

Mapping 테이블을 Row 기반으로 동적으로 만들 수 있는가 ? 여부이다.

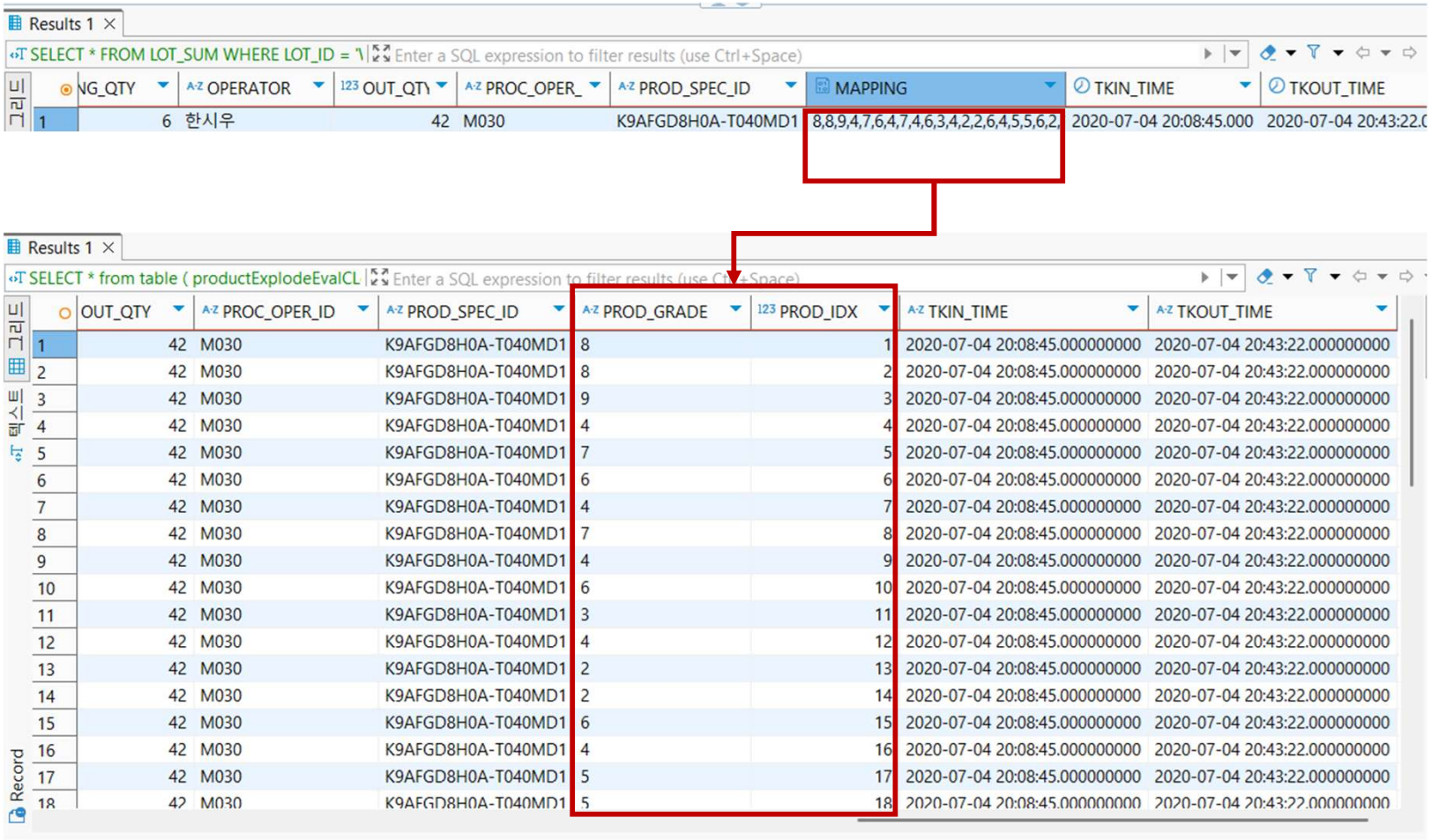
별도의 테이블로 만드는 것은 PL/SQL 등을 사용하면 되지만, 공간이 많이 들 수 밖에 없다.

그렇다면 어떻게 하면 될까 ?

Copyright @Raymond. All rights reserved

8 Python Smart Factory 관련 Example (#4 데이터 전처리)

BDAE(TM) 는 전처리 모듈을 탑재하고 있으며, 동적으로 아래와 같이 만들 수 있다.
동적으로 만들어 주며 이 경우 Python, R 을 호출하지 않고 순수하게 Oracle Native 한 기능을 이용.



Copyright @Raymond. All rights reserved

8 Python Smart Factory 관련 Example (#4 데이터 전처리)

이 productExplodeEvalClob 함수는 BDAE(TM) 에서 제공되는 함수이며 CLob 의 경우 Python, R 없이 만들어진 것이다. CLob 이 4000 바이트가 넘는 경우 일반 SQL 문으로 해결 할 수 없기 때문이다.

※ 문제는 단일 SQL 로 만들 수 없는, 복잡한 스키마들의 구성의 경우이다. BDAE(TM) 는 할 수 있지만, 백엔드에서는 여러 번 DB Query 를 날려야 만들 수 있을 것이다.

※ BLOB 은 어떻게 할까 ? 이때는 BDAE(TM) Python 이나 R 을 써야만 한다. 추론을 생각해 보라.

```
SELECT *
FROM table (
      productExplodeEvalCLob(
       cursor(
         SELECT *
         FROM LOT_SUM
         WHERE 1=1
             AND LOT_ID = 'WKE21X1B31'
             AND MACHINE_ID = '48PARA-03'
             AND DURABLE_ID = 'Z718')
      )
);
```


Copyright @Raymond. All rights reserved

8 Python Smart Factory 관련 Example (#5 함수의 선택)

BDAE(TM) 함수를 골라 보자. 데이터를 BDAE(TM) 가 전처리 한 뒤에 한번에 Python 모듈에게 주어야 함으로 apTableEval() 을 사용해야 한다.

apGroupEval 은 조건에 맞는 여러개의 LOT 들을 한꺼번에 만들 때 사용하면 된다.
이때는 Group Column 들이 추가적으로 필요하다.

apEval 을 억지로 사용할 경우에는 BDAE(TM) 가 데이터를 주지 않기 때문에 안에서 모듈 안에서 스스로 데이터를 Query 한 후 사용하면 된다.

SQL Interface function	Description
apEval()	데이터를 R 모듈에서 자체적으로 공급할 때
apTableEval()	데이터를 BDAE 가 전체를 줄 때
apRowEval()	데이터를 BDAE 가 Row 수 만큼 나누어 줄 때
apGroupEval()	데이터를 BDAE 가 Grouping 하여 공급할 때

Copyright @Raymond. All rights reserved

8 Python Smart Factory 관련 Example (#6 최종 SQL 만들기)

Python 모듈에 맞춰서 입력 SQL 을 만들어 주어야 한다.
빨간 부분이 Pandas DataFrame 컬럼으로 데이터와 함께 만들어져 호출 된다.

```
WITH LOT_SUM_ONE AS (
  SELECT LOT_ID,
    CASE WHEN PROD_GRADE = 9 THEN 'BAD BLOCK'
         WHEN PROD_GRADE = 8 THEN 'ASSEMBLY'
         WHEN PROD_GRADE = 7 THEN 'PTX'
         WHEN PROD_GRADE = 6 THEN 'ID FAIL'
         WHEN PROD_GRADE = 5 THEN 'OPEN'
         WHEN PROD_GRADE = 4 THEN 'ABS'
         WHEN PROD_GRADE = 3 THEN 'JOLY'
         WHEN PROD_GRADE = 2 THEN 'FIT'
         WHEN PROD_GRADE = 1 THEN 'MIN'
         ELSE
           'N/A'
      END PROD_GRADE_DESC
    , COUNT(PROD_GRADE) CNT from table (
      productExplodeEvalClob(cursor(
        SELECT * FROM LOT_SUM WHERE LOT_ID = 'WKE21X1B31' AND MACHINE_ID = '48PARA-03' AND
DURABLE_ID = 'Z718'))) GROUP BY LOT_ID, PROD_GRADE
 )
SELECT *
 FROM table(apTableEval(
  cursor(
    SELECT * FROM LOT_SUM_ONE
  ),
  NULL,
  'XML'  -- XML, PNG, JPG 등 은 BDAE 의 Predefined Keywored 이다.
  'LOTSUM_ERR_PIE:display'))
```

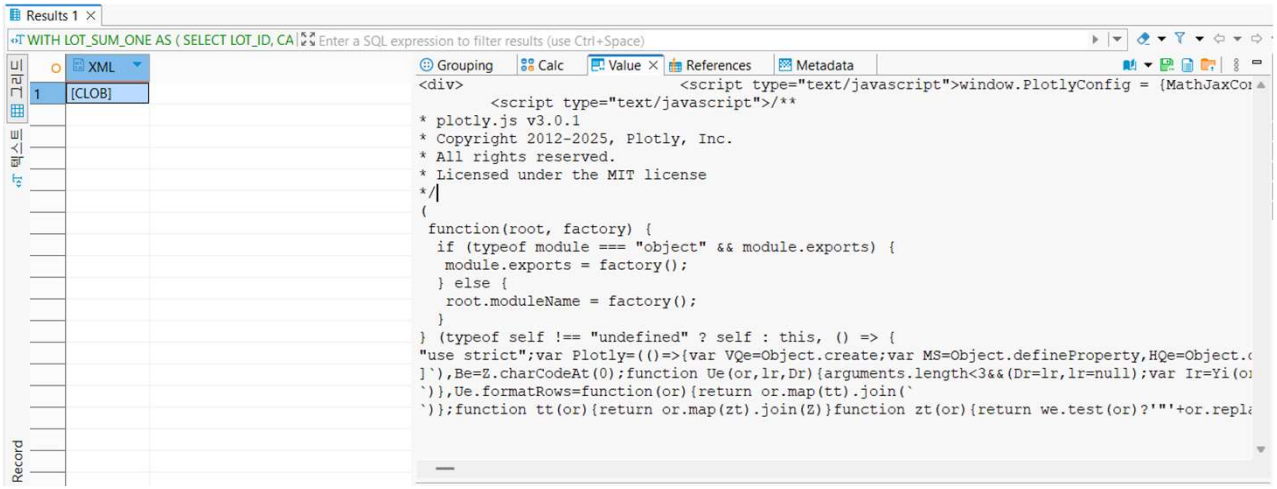
```
import pandas as pd
import plotly.express as px
import plotly.offline as py

def display(df):
    fig = px.pie(df, values='CNT', names='PROD_GRADE_DESC', title='LOT ID : ' + df['LOT_ID'][0] + ', Product Grade Distribution')
    fig.update_traces(textposition='inside', textinfo='percent+label')
    total = py.offline.plot(fig, output_type='div')
return (total)
```

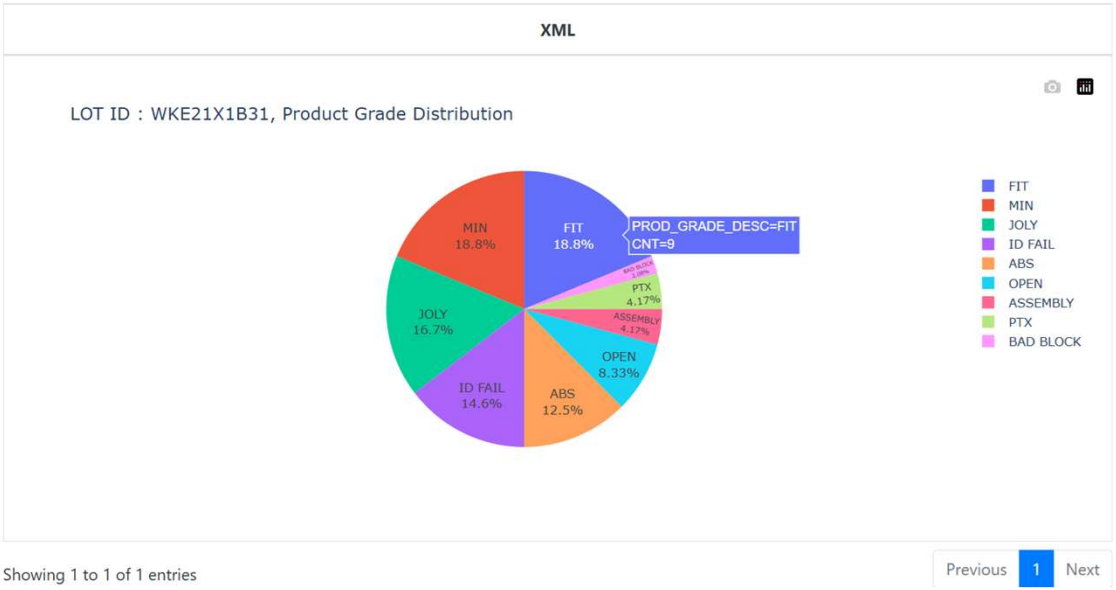

Copyright @Raymond. All rights reserved

8 | Python Smart Factory 관련 Example (#7 결과 확인)

DB 툴로 앞의 SQL 을 실행시키면 결과는 CLOB 형태로 우측의 JavaScript 구성이 보일 것이다.



BDAE Web 으로 보면 의존성 css, javascript 때문에 Interactive 이미지가 동작함을 보게 된다.



Copyright @Raymond. All rights reserved

8 Python Smart Factory 관련 Example (#8 확장)

하나의 LOT 이 아니라 TKIN/OUT 시점의 모든 LOT 에 대한 Pie Chart 를 한번에 받으려면 ?
모듈도 변경 되어야 한다.

apGroupEval 은 Group Column 을 LOT_ID 로만 했을 때 LOT 마다 자동으로 호출된다.
따라서 여러 개의 LOT_ID 묶음이 들어 올 것이라고 생각하지 않아도 된다. 자신의 것만 하면 된다.

사실, 이 모듈이 좀더 재활용성이 원래보다 더 높다. 다만 앞에서는 'XML' 이라는 Predefined Keyword 를 보여주려고 했던 것 뿐이다.

```
import pandas as pd
import plotly.express as px
import plotly.offline as py

def display(df):
    fig = px.pie(df, values='CNT', names='PROD_GRADE_DESC', title='LOT ID : ' + df['LOT_ID'][0] + ', Product Grade Distribution')
    fig.update_traces(textposition='inside', textinfo='percent+label')
    total = py.offline.plot(fig, output_type='div')
    dict = { 'LOT_ID': [df['LOT_ID']], 'XML' : total }

return (pd.DataFrame(dict)
```


Copyright @Raymond. All rights reserved

8 Python Smart Factory 관련 Example (#8 확장)

SQL 문도 변경되어야 한다. 다만, 만약 앞장의 모듈이라면 apTableEval(), apGroupEval() 모두에서 동일한 Python 모듈을 사용할 수 있다. 즉, 재활용 된다는 의미이다.

```
WITH LOT_SUM_ALL AS (
  SELECT LOT_ID,
    CASE WHEN PROD_GRADE = 9 THEN  ' BAD BLOCK '
         WHEN PROD_GRADE = 8 THEN  ' ASSEMBLY '
         WHEN PROD_GRADE = 7 THEN  ' PTX '
         WHEN PROD_GRADE = 6 THEN  ' ID FAIL '
         WHEN PROD_GRADE = 5 THEN  ' OPEN '
         WHEN PROD_GRADE = 4 THEN  ' ABS '
         WHEN PROD_GRADE = 3 THEN  ' JOLY '
         WHEN PROD_GRADE = 2 THEN  ' FIT '
         WHEN PROD_GRADE = 1 THEN  ' MIN '
         ELSE
           'N/A'
      END PROD_GRADE_DESC
    , COUNT(PROD_GRADE) CNT from table (
      productExplodeEvalCLob(cursor(
        SELECT * FROM LOT_SUM WHERE BETWEEN TKIN ... AND ....))) GROUP BY LOT_ID, PROD_GRADE
)
SELECT *
 FROM table(apGroupEval(
  cursor(
    SELECT * FROM LOT_SUM_ALL
  ),
  NULL,
  'SELECT CAST(NULL AS VARCHAR2(40) LOT_ID, TO_CLOB(NULL) XML FROM DUAL ' ,
  LOT_ID,  -- Grouping Column 이다.  (PRODSPEC, LOT_ID, DURABLE_ID 등으로도 할 수 있다.)
 'LOTSUM_ERR_PIE:display'))
```


Copyright @Raymond. All rights reserved

9 | BDAE(TM) Web 에 대해서

BDAE(TM) 의 장점은 SQL 문으로 호출된다는 것이고, 따라서 어떠한 Oracle Connectivity 를 제공하는 클라이언트에서 호출될 수 있다.

BDAE(TM) Web 은 Spring Boot + JSP (Tiles) 기반이며 Sprint Boot 에서 제공하는 jdbcTemplate 을 사용해서 BDAE(TM) SQL 을 호출 한다. (매우 단순하고, 결과는 jQuery, dataTable 하나로 되어 있음.)

그 결과가 String, BLOB, CLOB, .. 인지 호출 결과 즉시 알기 때문에 화면 처리가 매우 간단할 수 있다. 특히 시각화의 경우 형태로 오기 때문에 적절한 GUI 구성이 손쉽다.

다만, 이 경우 Spring Boot 프로젝트의 static 부분에 이미지의 javascript 관련 라이브러리, css 등이 있어야 Interactive 한 것이 동작할 수 있지만, 이 경우는 그리 세팅이 어려운 부분은 아님.

BDAE(TM) Web 은 고객이 직접 커스터마이징 하기 위한 예제라고 생각하면 된다.

Copyright @Raymond. All rights reserved

The Beautiful Times ...

It started at that time, in that place... but it was still a beautiful time.

Copyright @Raymond. All rights reserved