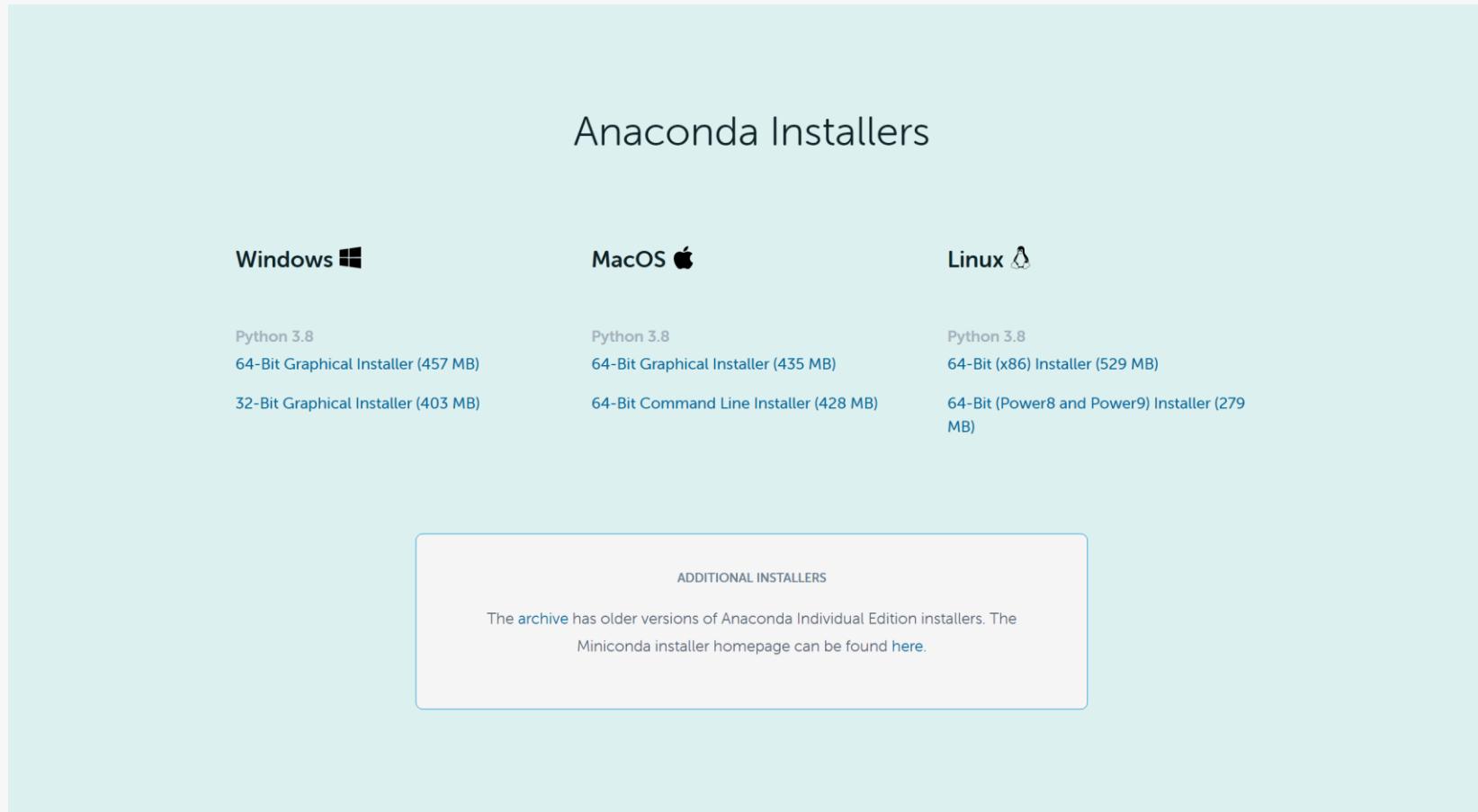


# 개발환경설정

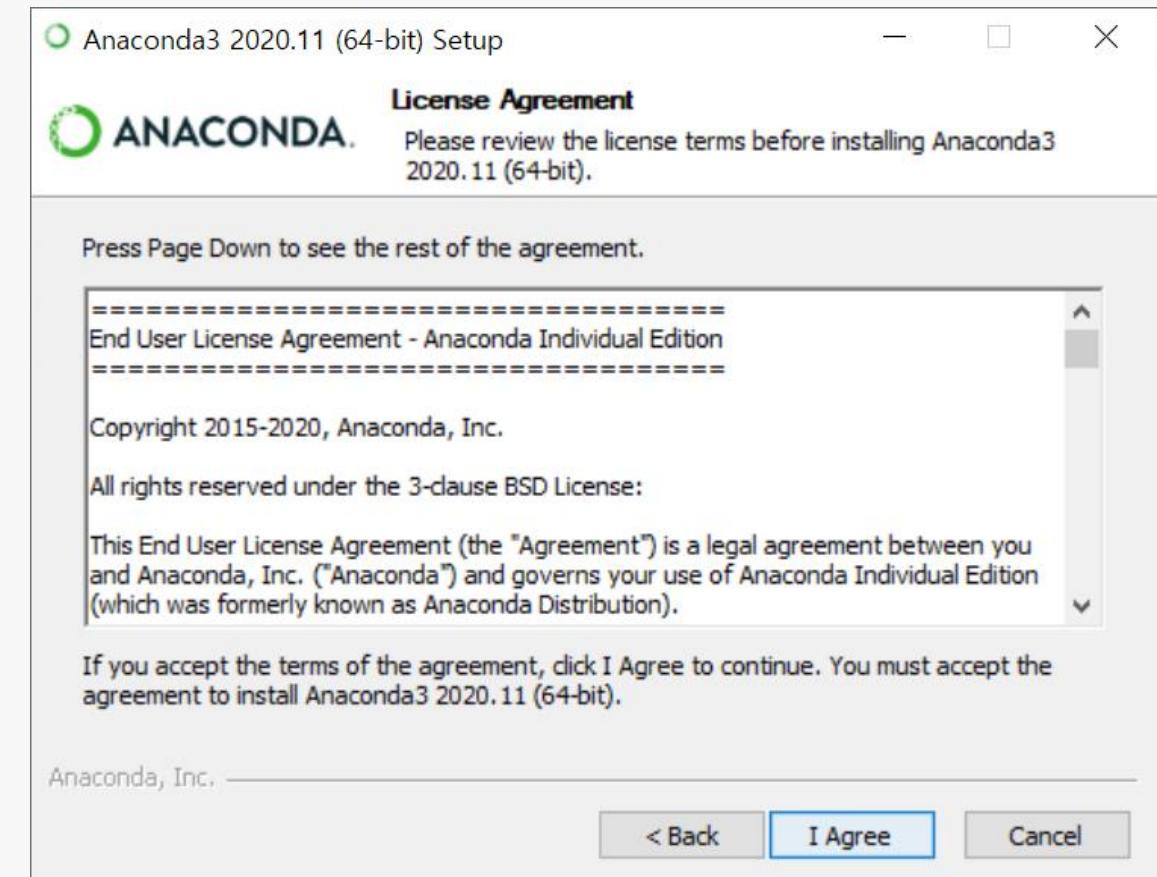
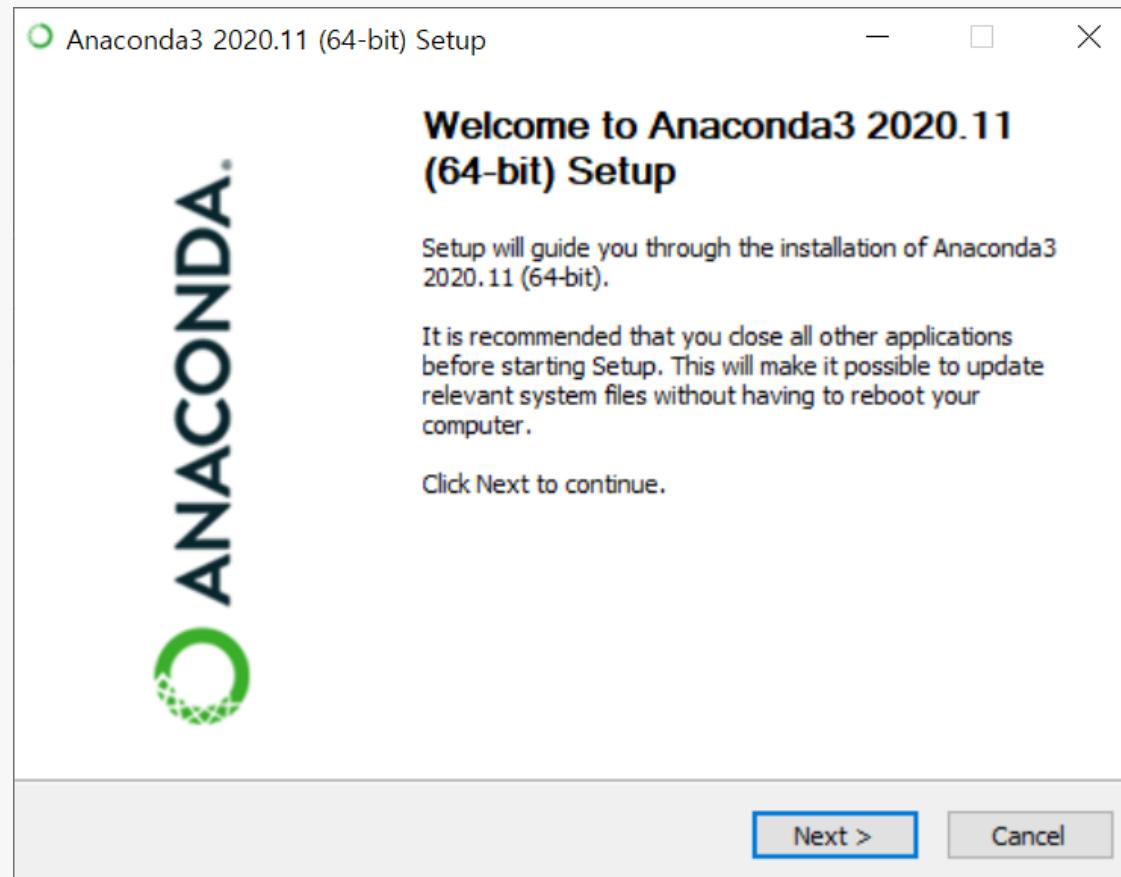
with  python™

OS별 머신러닝 & 딥러닝을 구현할 수 있도록 도와주는 패키지  
(웹사이트 주소: <https://www.anaconda.com/products/individual>)



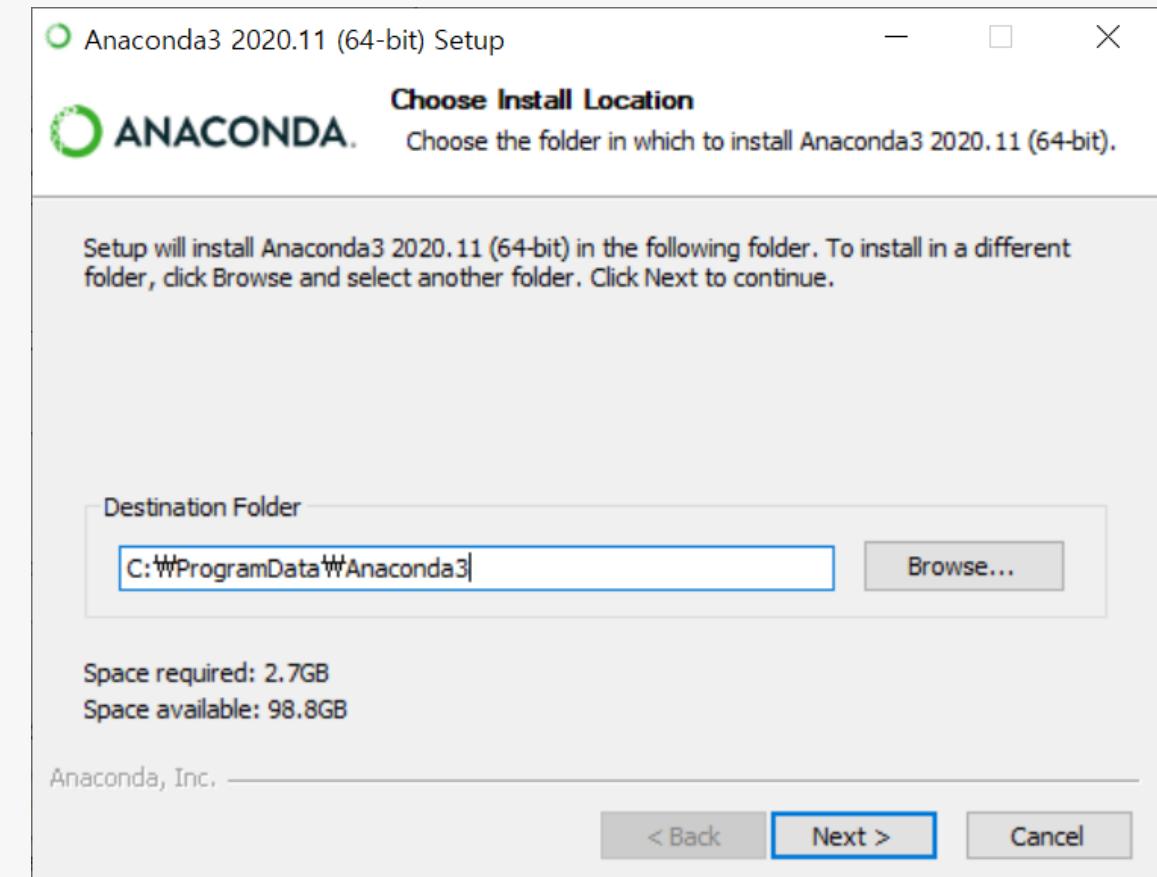
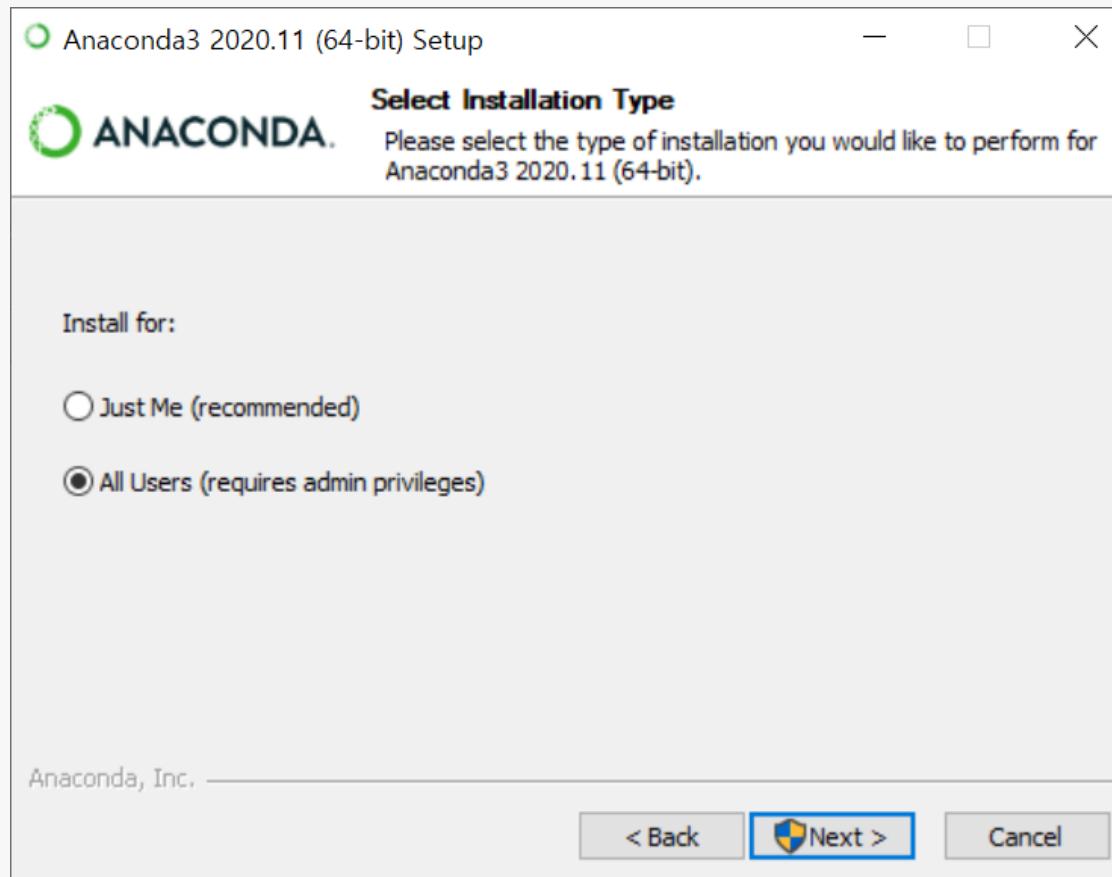
# 00. 아나콘다 – Windows 10

(2020년 12월 기준)



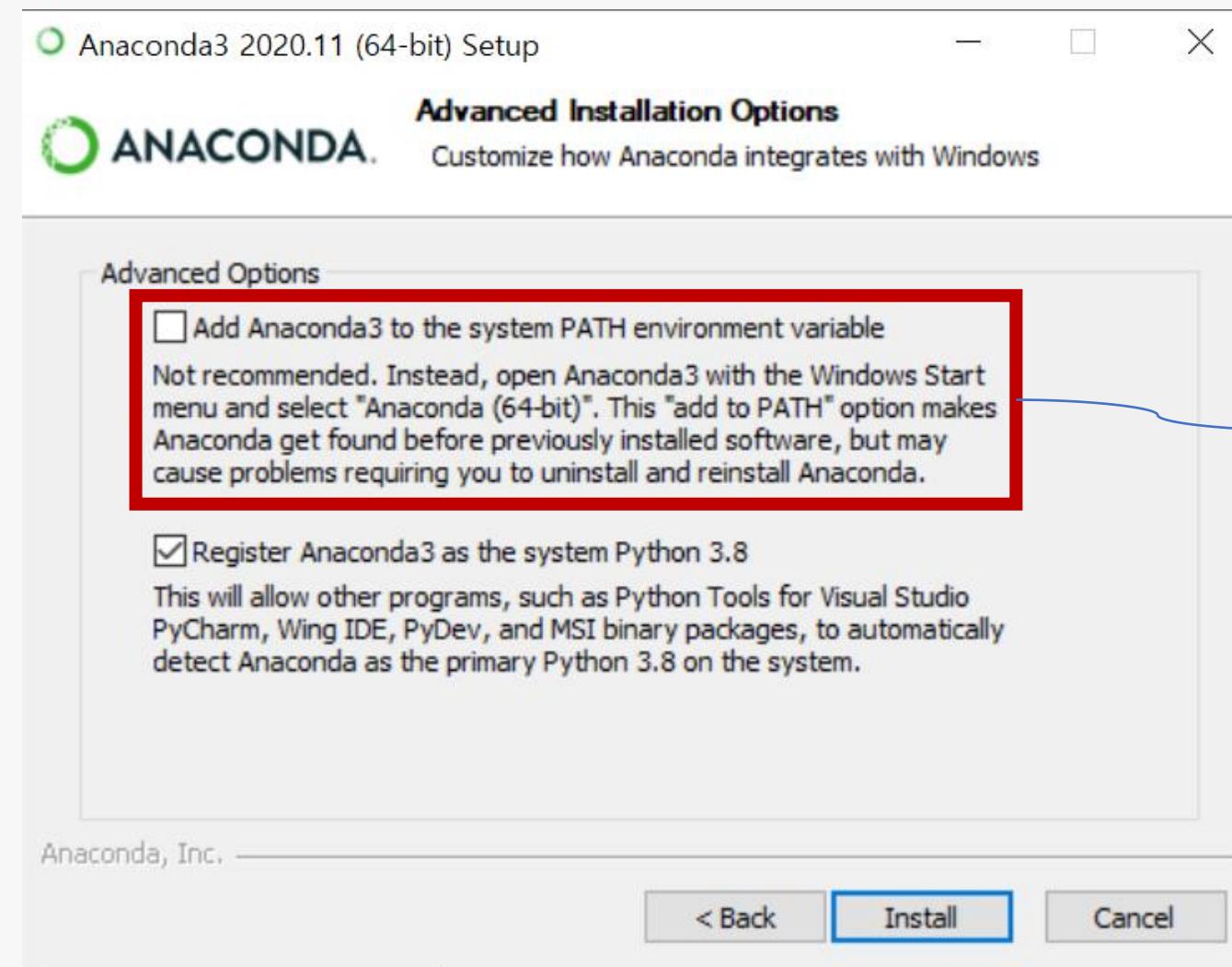
# 00. 아나콘다 – Windows 10

(2020년 12월 기준)



# 00. 아나콘다 – Windows 10

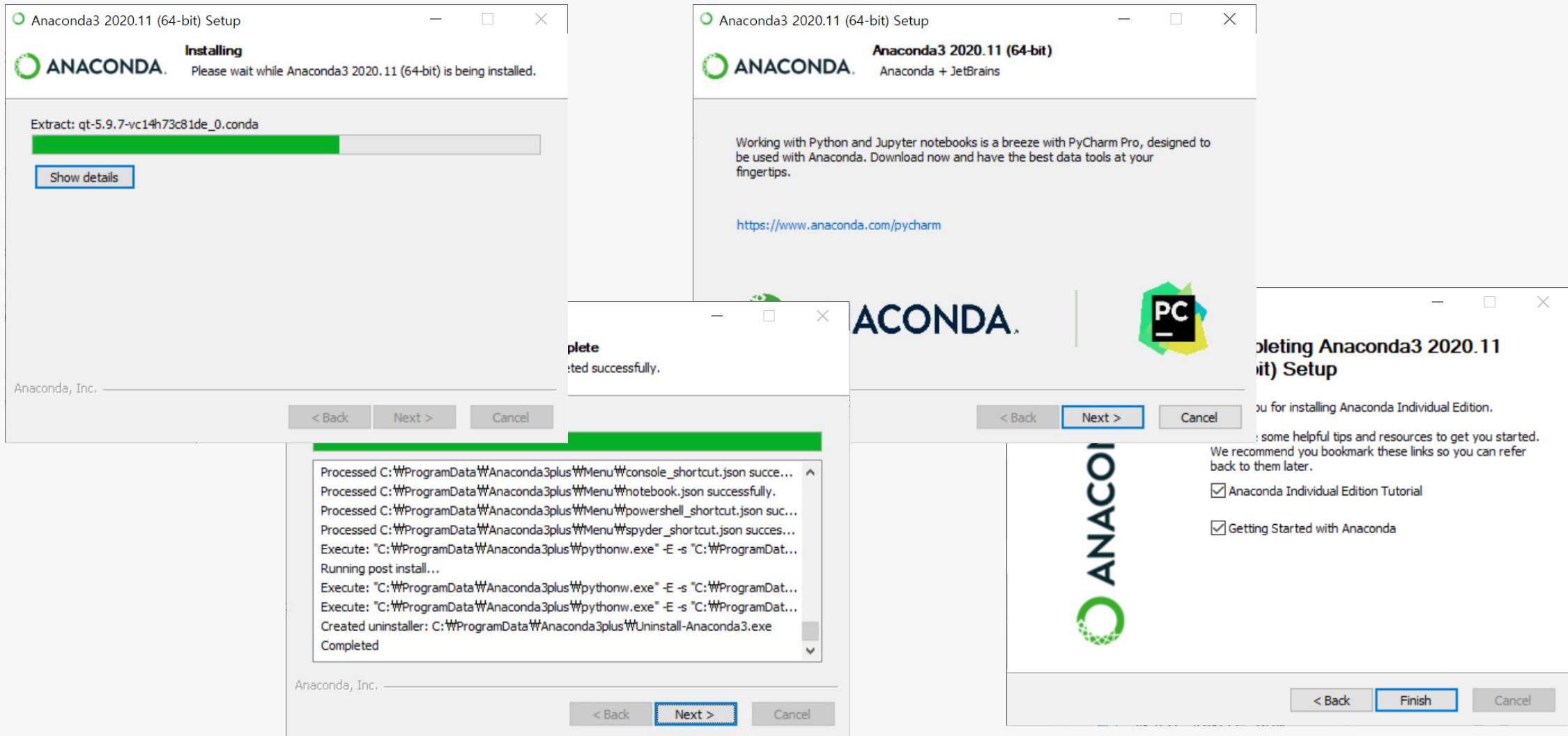
(2020년 12월 기준)



- 환경 변수의 개념을 잘 모른다면 추가하지 않는다.
- 환경 변수는 언제든지 추가가 가능하다.

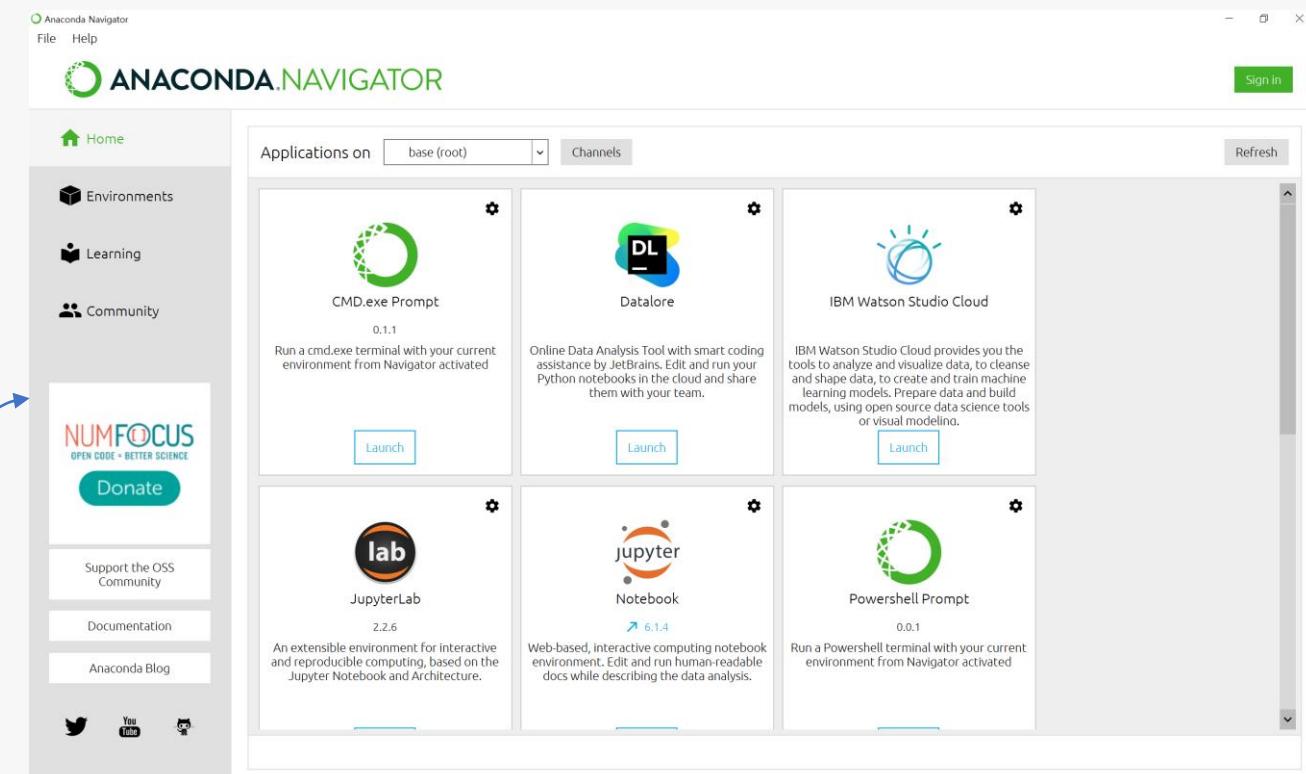
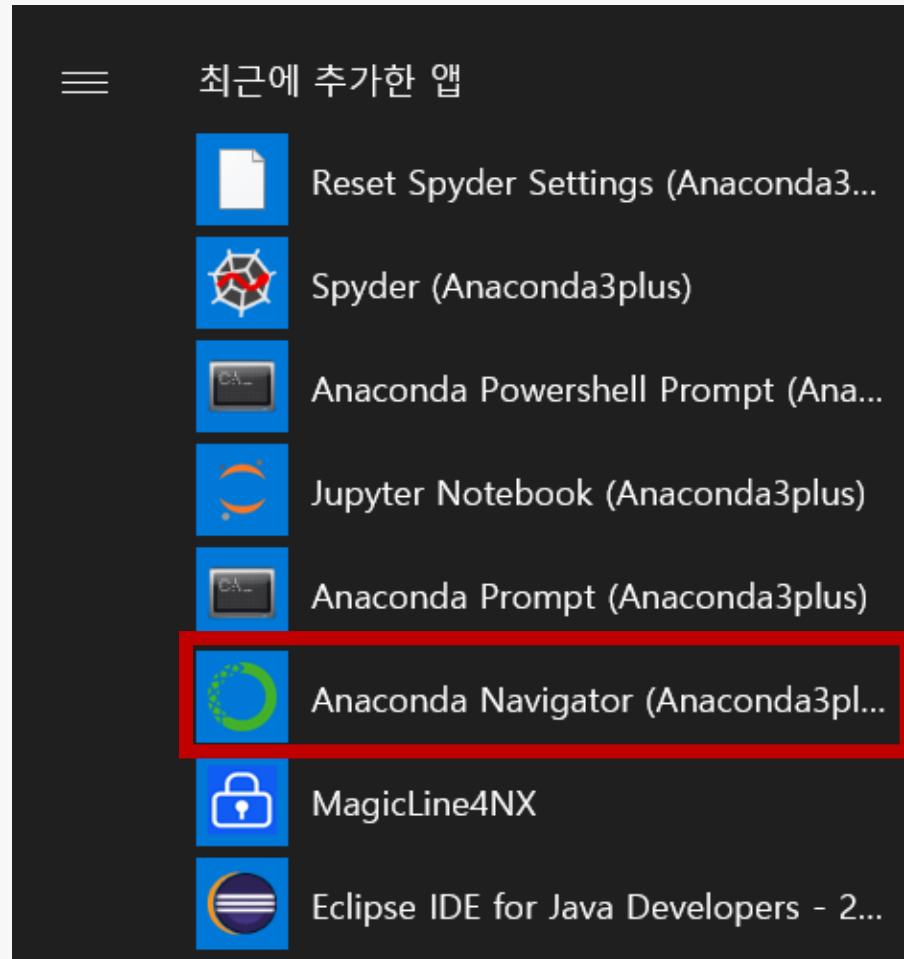
# 00. 아나콘다 – Windows 10

(2020년 12월 기준)

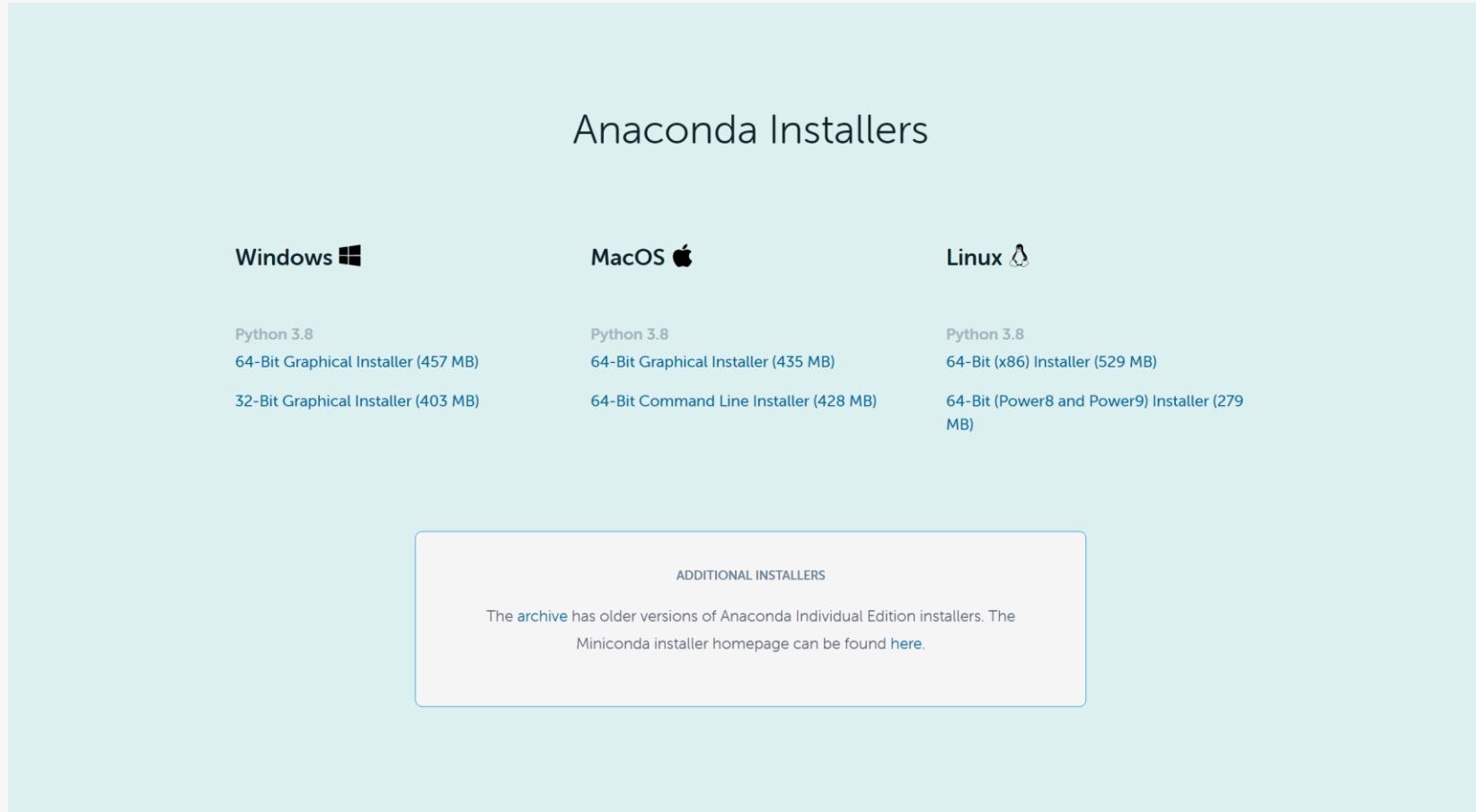


# 00. 아나콘다 – Windows 10

(2020년 12월 기준)

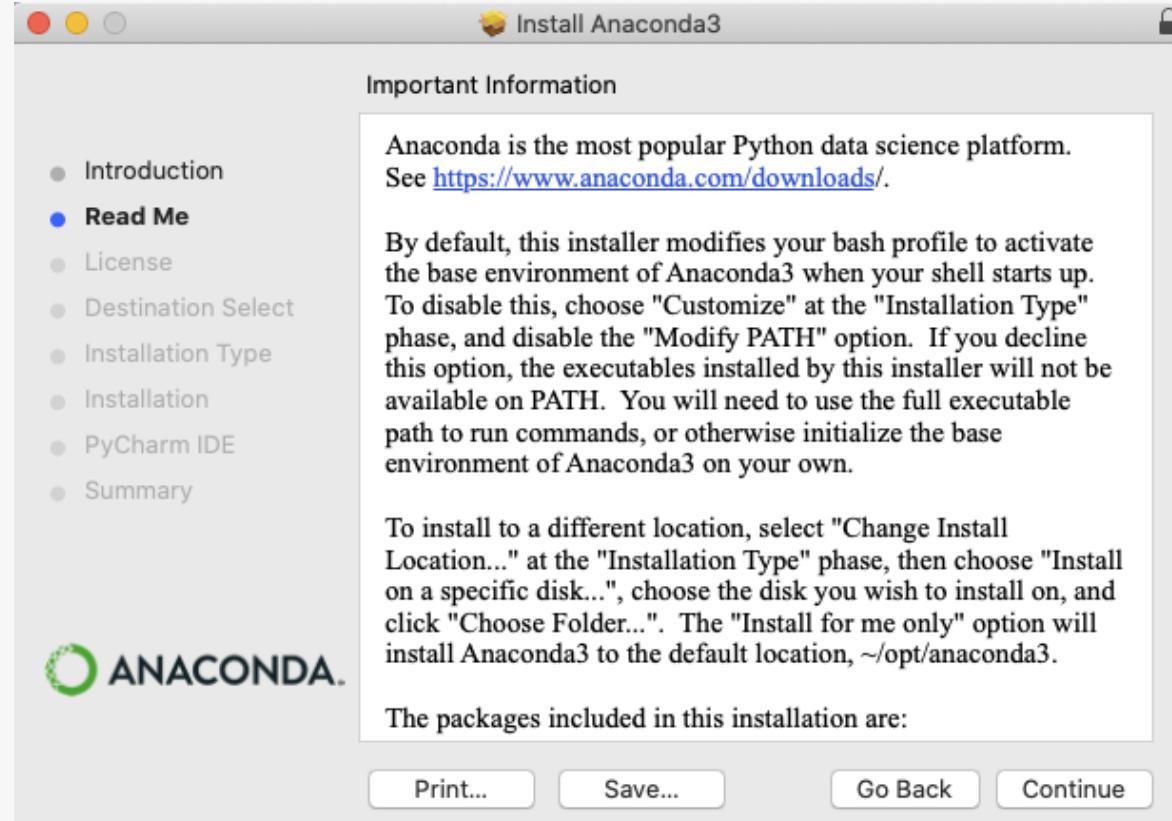
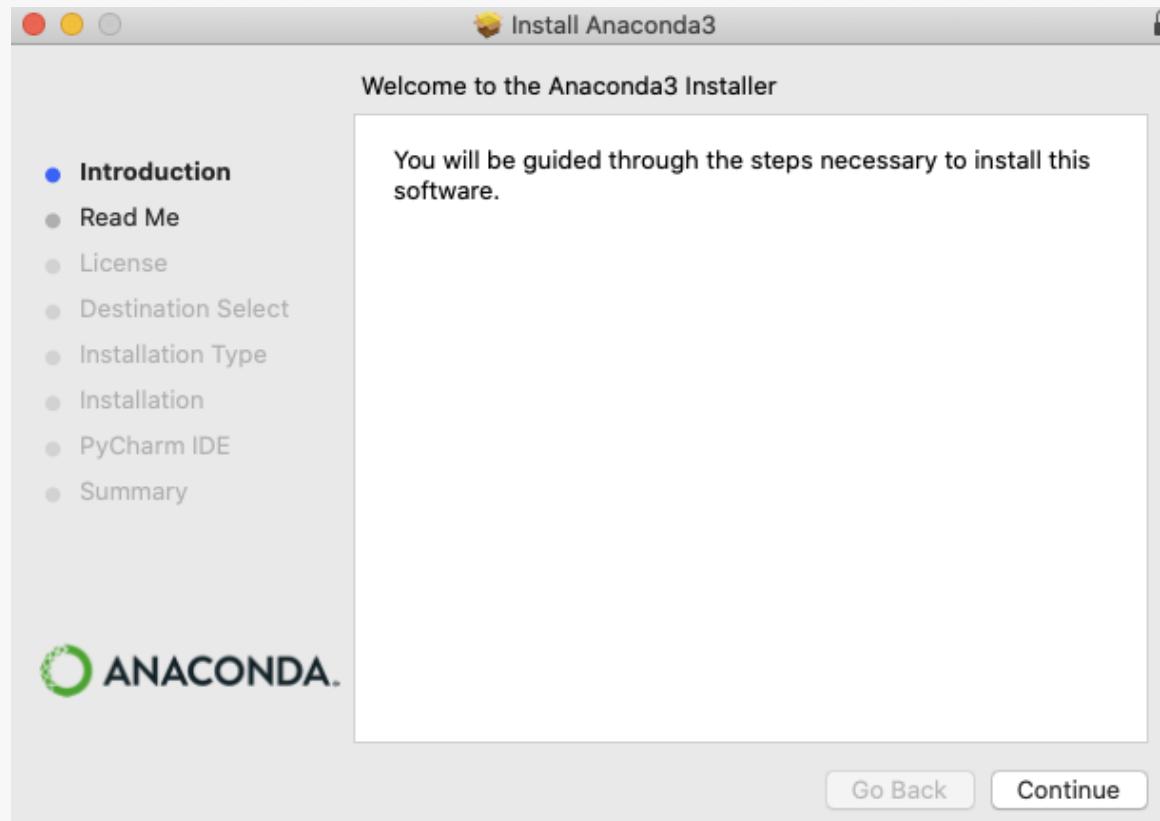


Shell Script 실행 방법 모른다면, 64-Bit Graphical Installer 선택  
Shell Script 실행 방법 알고 있다면, 64-Bit Command Line Installer 선택



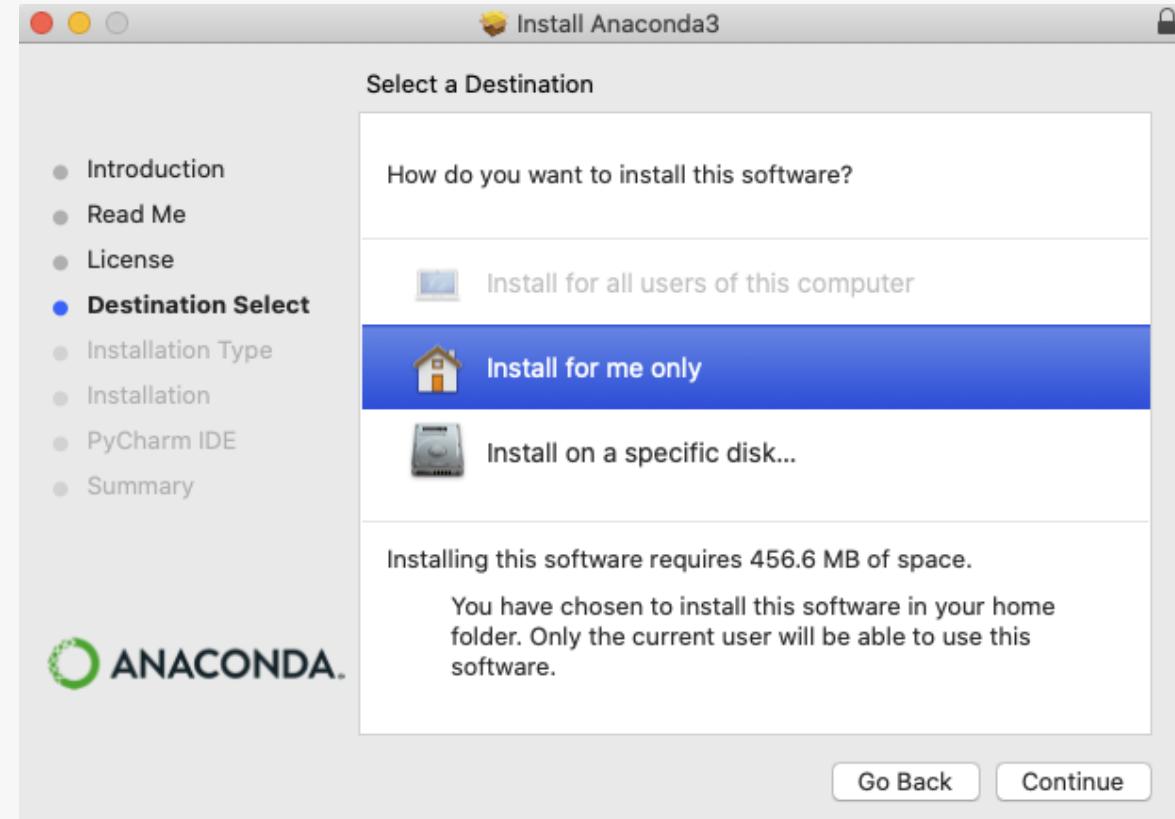
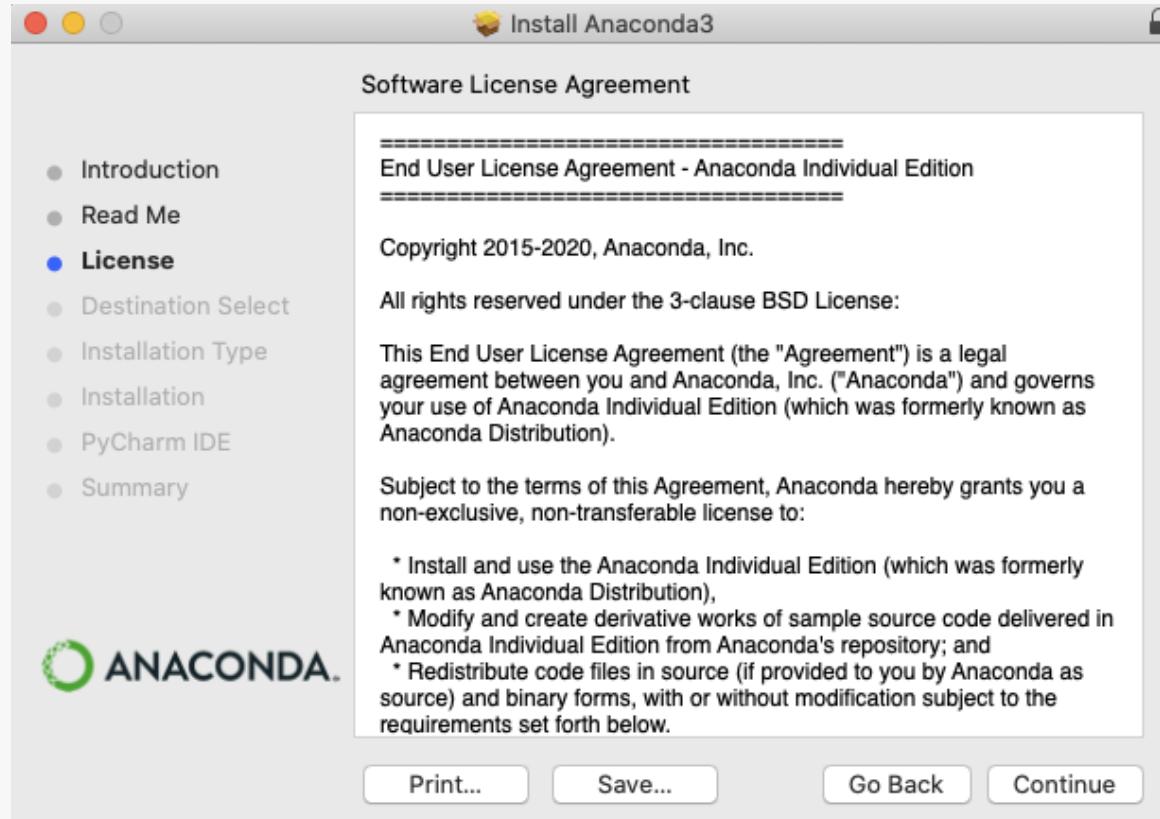
# 01. 아나콘다 – MacOS

(2020년 12월 기준)



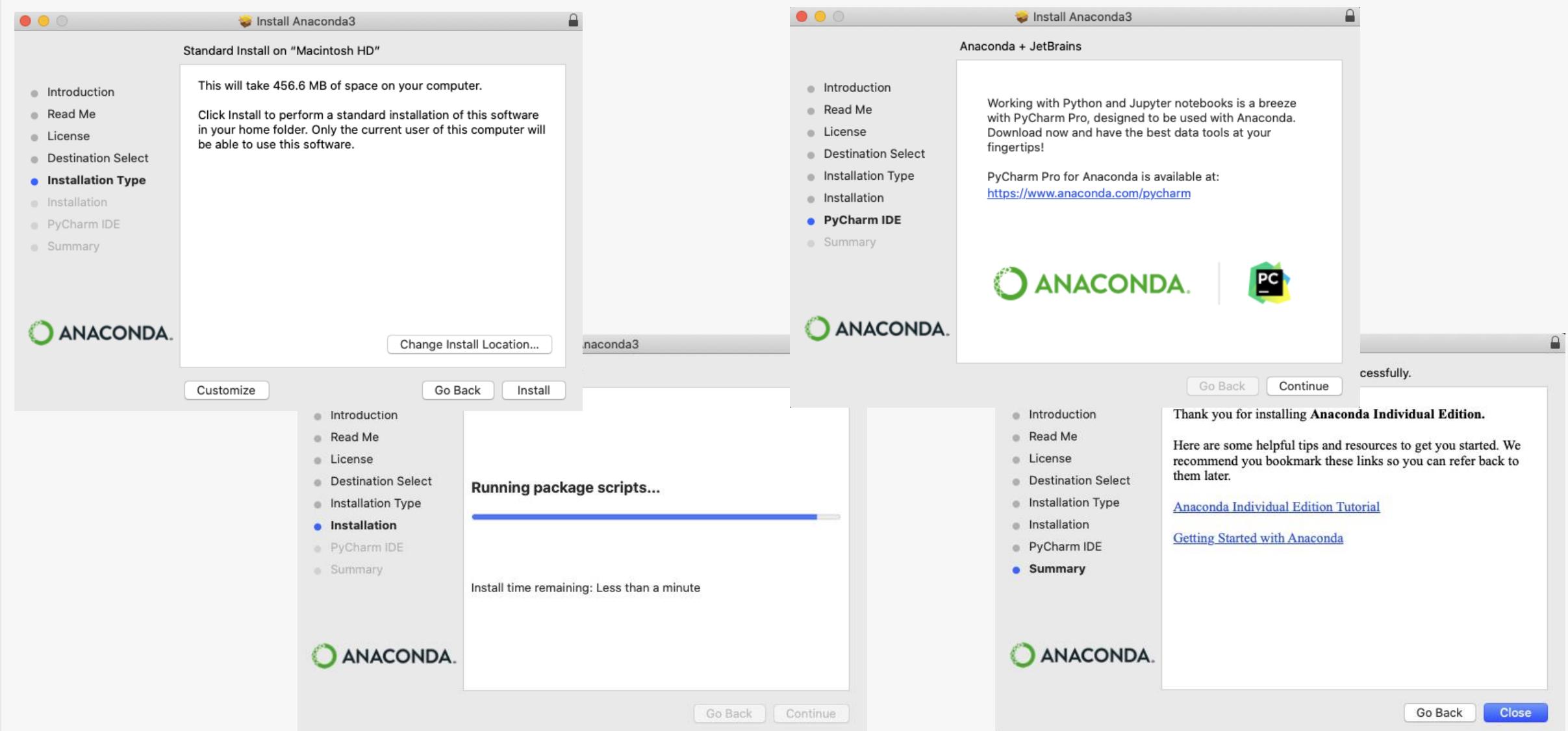
# 01. 아나콘다 – MacOS

(2020년 12월 기준)



# 01. 아나콘다 – MacOS

(2020년 12월 기준)



# 01. 아나콘다 – MacOS

(2020년 12월 기준)

The screenshot shows the Anaconda Navigator interface. On the left is a sidebar with links for Home, Environments, Learning, and Community, along with a NUMFOCUS donation button and social media links. The main area displays a grid of applications:

Application	Description	Version	Action
Datalore	Online Data Analysis Tool with smart coding assistance by JetBrains.	2.2.6	Launch
IBM Watson Studio Cloud	IBM Watson Studio Cloud provides tools for data analysis, machine learning, and AI development.	6.1.4	Launch
JupyterLab	An extensible environment for interactive and reproducible computing.	2.2.6	Launch
Jupyter Notebook	Web-based, interactive computing notebook environment.	2020.2.1	Launch
PyCharm Community	An IDE by JetBrains for pure Python development.	4.7.7	Launch
Qt Console	PyQt GUI that supports inline figures, multiline editing, and graphical calltips.	4.7.7	Launch
Spyder	Scientific Python Development Environment.	4.1.5	Launch
VS Code	Streamlined code editor with support for development operations like debugging, task running, and version control.	1.52.1	Launch
Glueviz	Multidimensional data visualization across files.	1.0.0	Install
Orange 3	Component based data mining framework.	3.26.0	Install
PyCharm Professional	A full-fledged IDE for both Scientific and Web Python development.	1.1.456	Install
RStudio	A set of integrated tools designed to help you be more productive with R.	1.1.456	Install

# 데이터 분석 강의 Intro

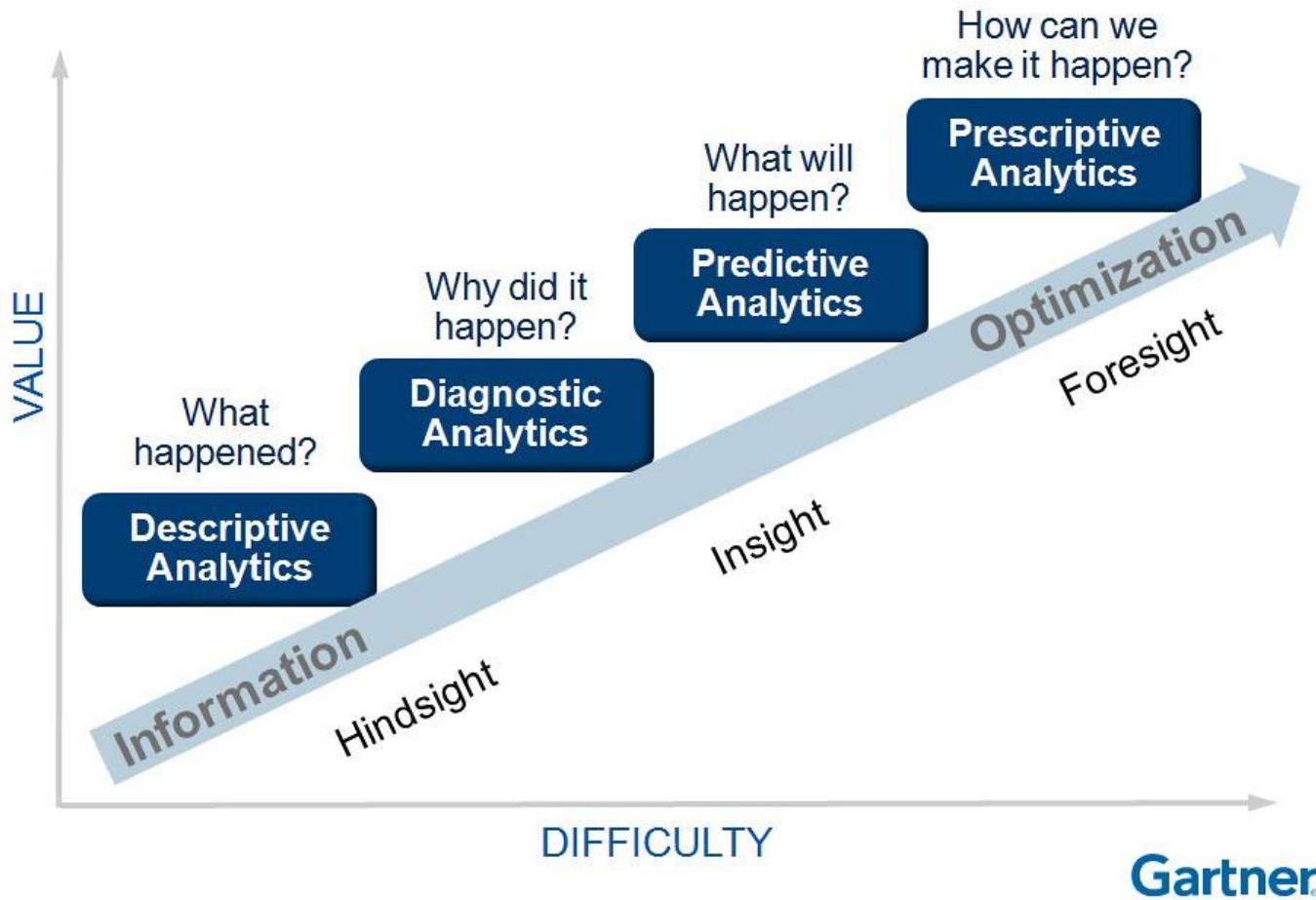
with



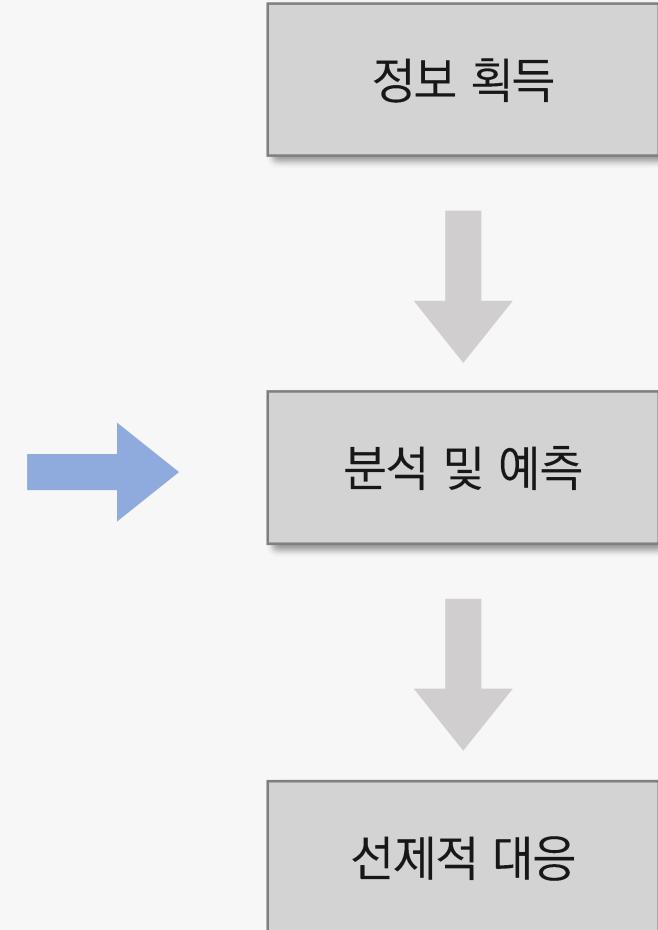
PL/SQL  
**ORACLE®**

# 01. 데이터 분석 가치 4단계

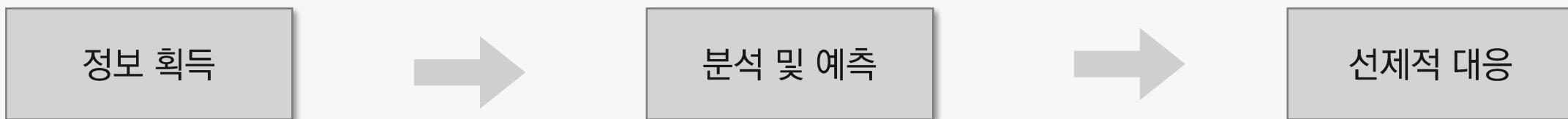
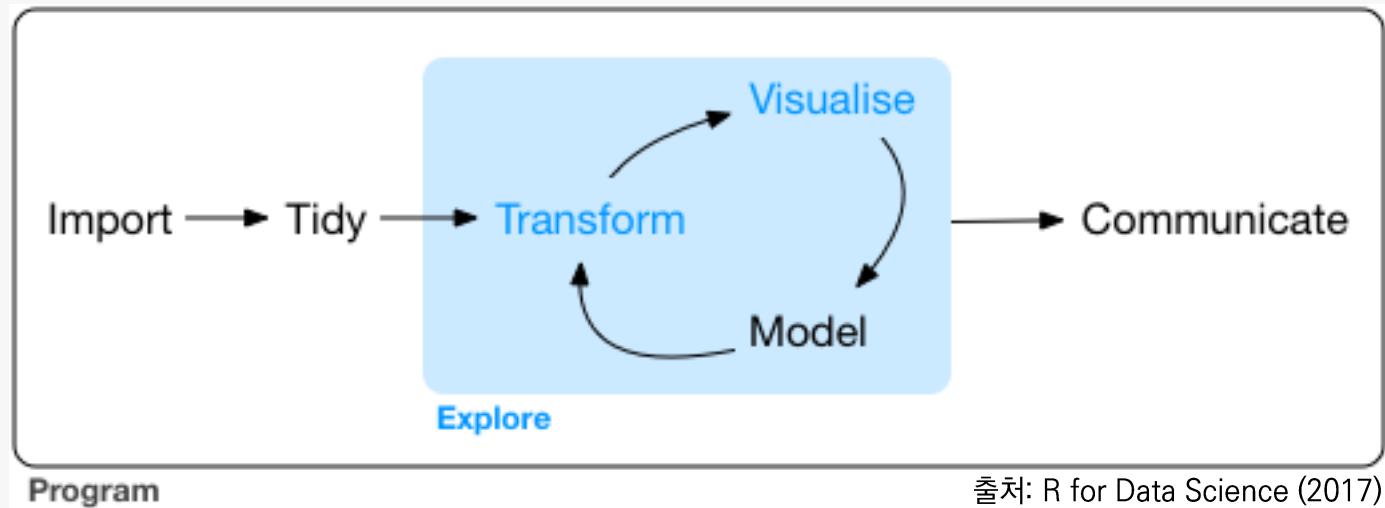
## Analytic Value Escalator



출처: Gartner (2012)



# 01. 데이터 분석 4단계



# 개발환경설정

with



PL/SQL  
**ORACLE®**

# 01. Oracle



**DBMS**  
(Database Management System)

- 정의 (Definition) : 응용 프로그램이 요구하는 DB구조, 변경, 제거
- 조작 (Manipulation): 삽입, 갱신, 삭제
- 제어 (Control): DB 접근할 수 있는 사용자 제한 및 성능 관리

**관계형 데이터베이스**  
(Relational Database)

- 통상적으로 RDB라고 함
- 행과 열로 구분하는 2차원 테이블 형태로 구성
- 오라클 DBMS, MS SQL SERVER, MYSQL, PostgreSQL

**Oracle의 인기**

- DB Engines Ranking 1위 (350개 DB) (2020 ~ 2021)<sup>1</sup>
- 국내 시장 점유율 70%<sup>2</sup>

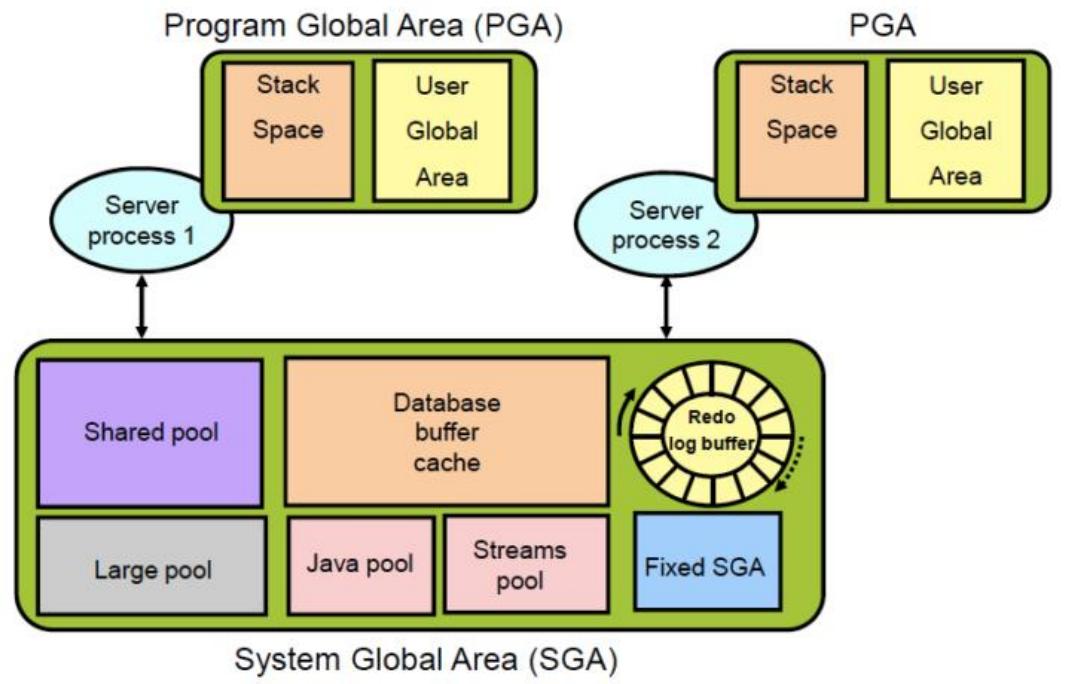
**ORACLE의 장점**

- 중앙 집중 방식, 쿼리 최적화 프로그램 등
- 다양한 플랫폼 지원 (Windows, MacOS, Linux 등)

출처 1: <https://db-engines.com/en/ranking>

출처 2: <https://www.sedaily.com/NewsView/1YXT5Z0V09>

## 02. 오라클 데이터베이스의 구조



출처 : 황선영 (2019). 오라클 데이터베이스의 구조

- PGA(Program Global Areas)
  - : 비공유 메모리, 각 서버 프로세스에 대한 데이터 및 정보 포함
  - : 유저의 글로벌 데이터나 세션 상태정보, 참조되는 포인터 정보
- SGA(System Global Area)
  - : Shared Pool – 공유 SQL 영역 및 라이브러리 캐시 옵티마이저, 하드 파싱, 소프트 파싱
  - : Large Pool – 백업이나 Recovery 작업 시 메모리 할당
  - : Java Pool – 오라클 DB에서 자바 이용할 수 있도록 도와줌

# 03. 오라클 설치 (Windows 10)

(2021년 6월 기준)

## 오라클 다운로드 (19c / 19.3 Enterprise Edition)

(웹사이트 주소: <https://www.oracle.com/database/technologies/oracle-database-software-downloads.html#19c>)

### Oracle Database 19c

Oracle Database 19c is the latest Long Term Release with the widest window of support duration. For details about database releases and their support timeframes, refer to Oracle Support Document 742060.1 (Release Schedule of Current Database Releases) on My Oracle Support.

#### 19.5 - Enterprise Edition (also includes Standard Edition 2)

Name	Download	Note
Oracle Solaris (x86 systems, 64-bit)	 ZIP (2.7 GB)	<a href="#">See All</a>

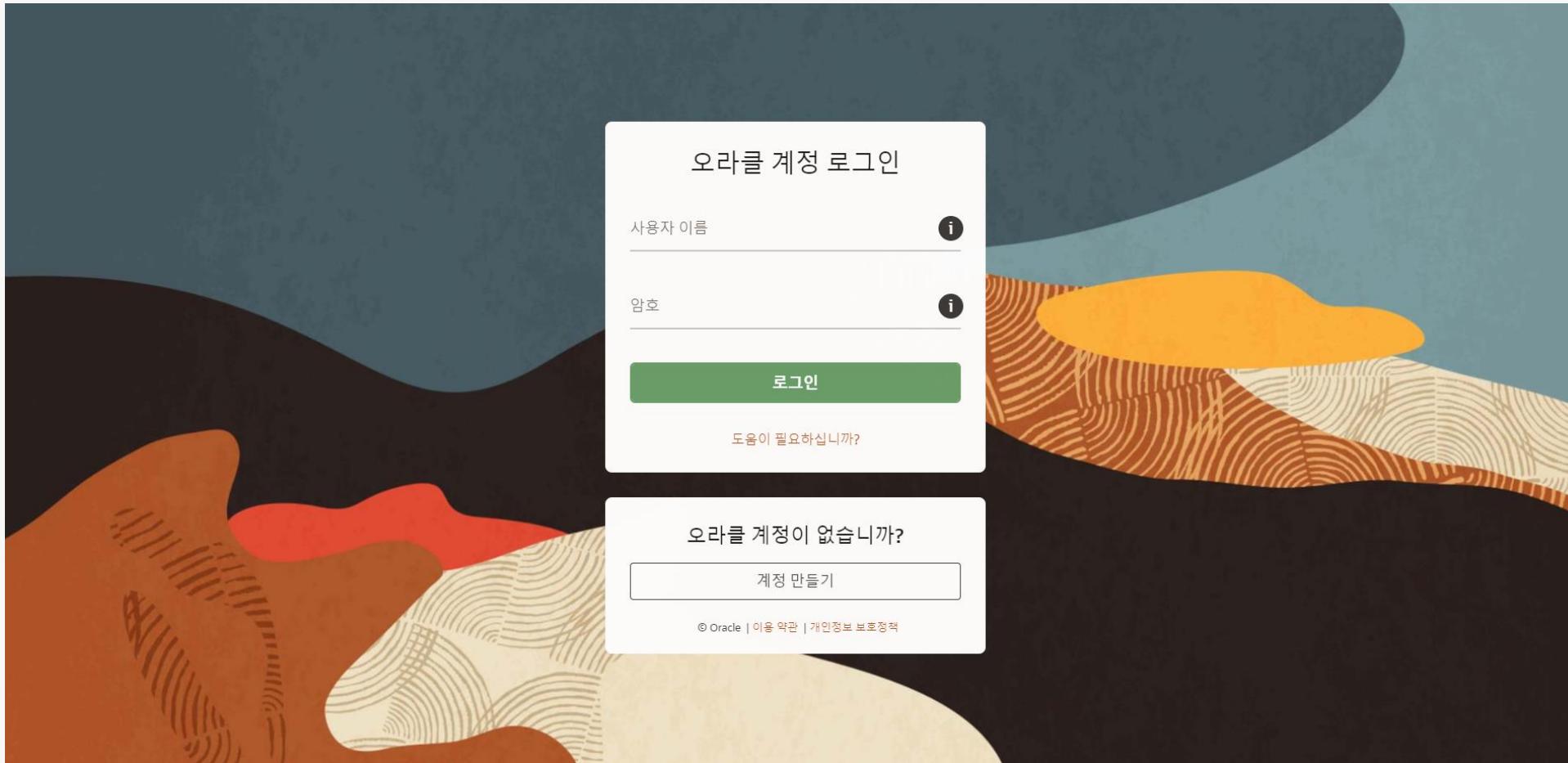
#### 19.3 - Enterprise Edition (also includes Standard Edition 2)

Name	Download	Note
Microsoft Windows x64 (64-bit)	 ZIP (2.9 GB)	<a href="#">See All</a>
Linux x86-64	 ZIP (2.8 GB)    RPM (2.5 GB)	<a href="#">See All</a>
Oracle Solaris (SPARC systems, 64-bit)	 ZIP (2.8 GB)	<a href="#">See All</a>
IBM AIX	 ZIP (4.1 GB)	<a href="#">See All</a>
HP-UX ia64	 ZIP (4.7 GB)	<a href="#">See All</a>
Linux on System z (64-bit)	 ZIP (2.6 GB)	<a href="#">See All</a>

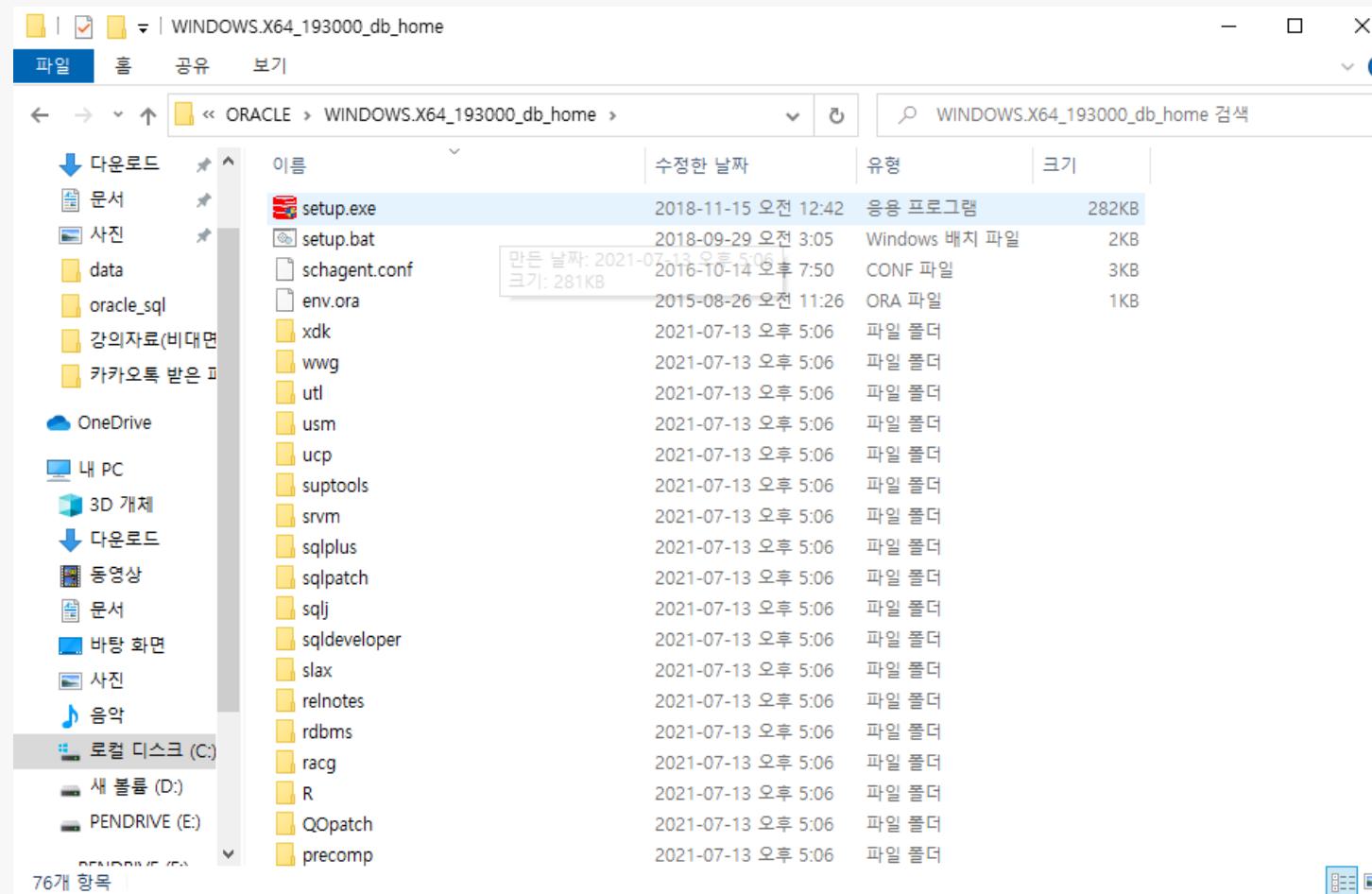
# 03. 오라클 설치 (Windows 10)

(2021년 6월 기준)

## 오라클 계정 로그인



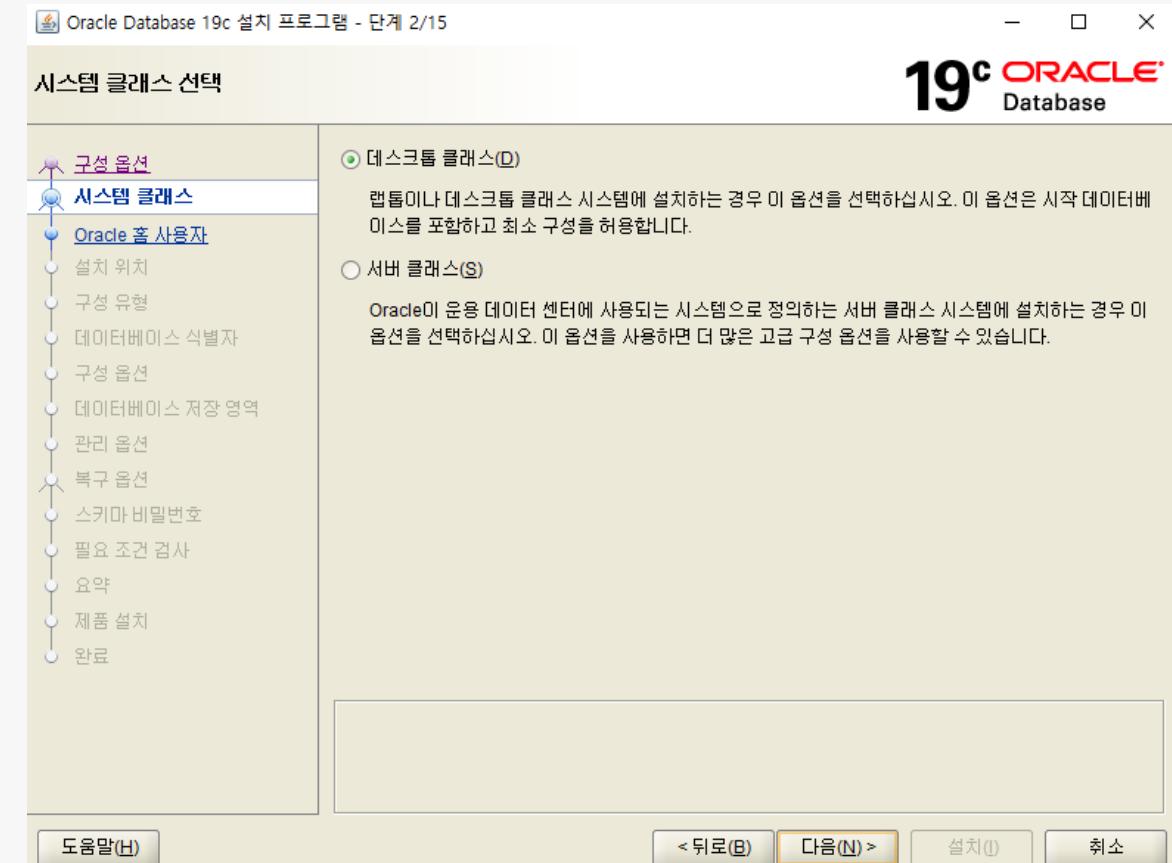
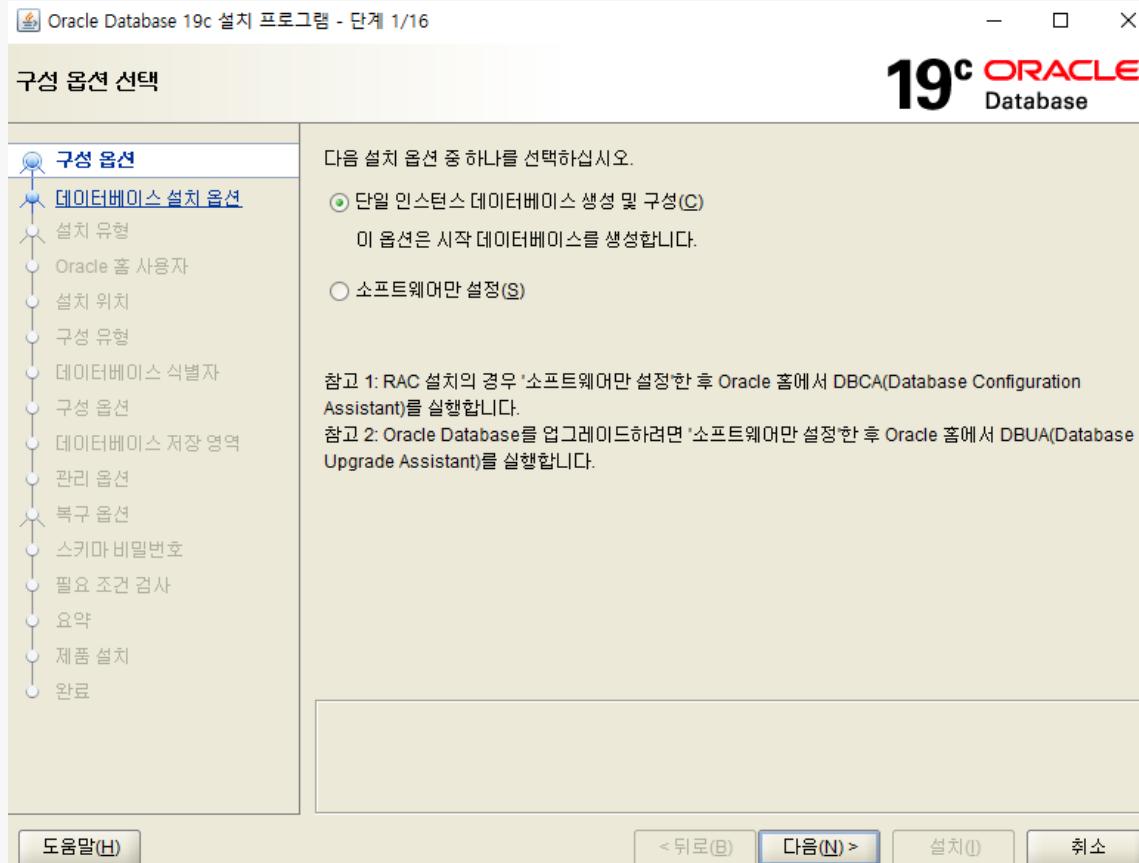
## 압축파일 해제 – 설치 프로그램 실행



# 03. 오라클 설치 (Windows 10)

(2021년 6월 기준)

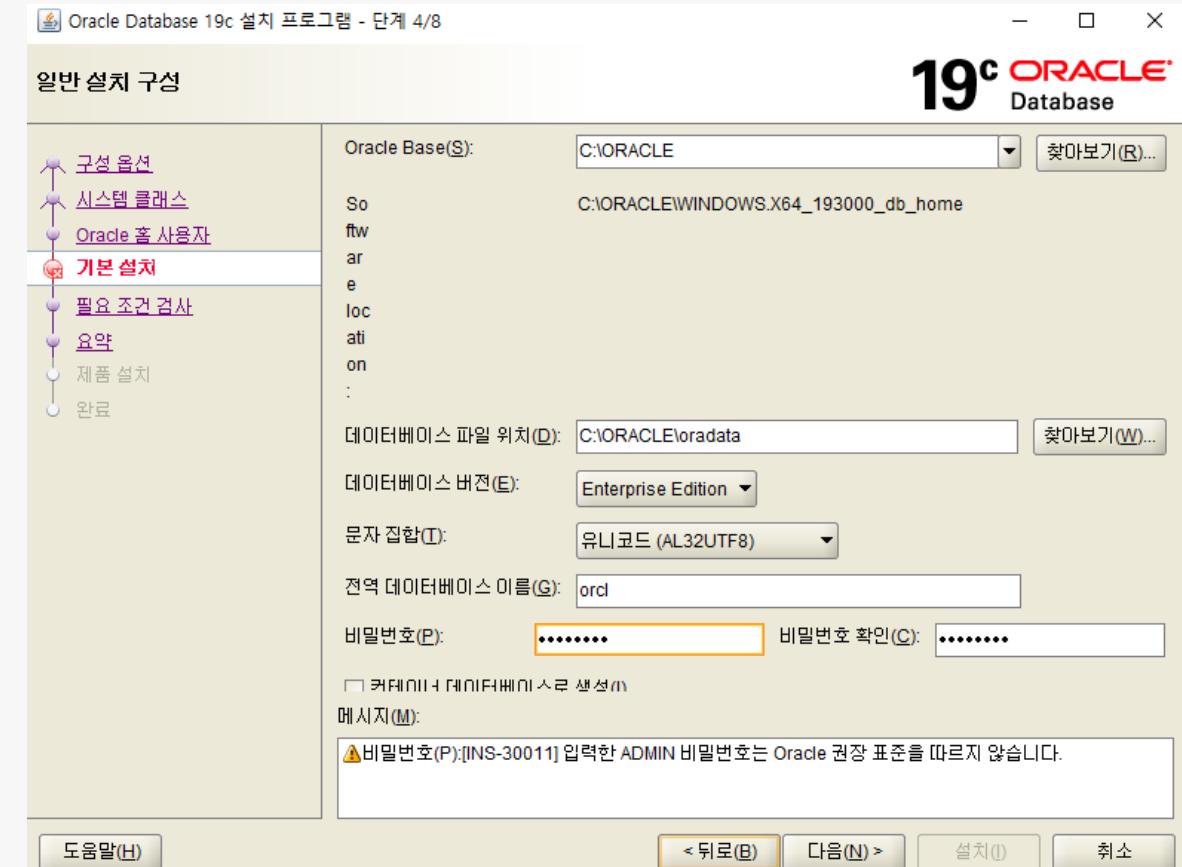
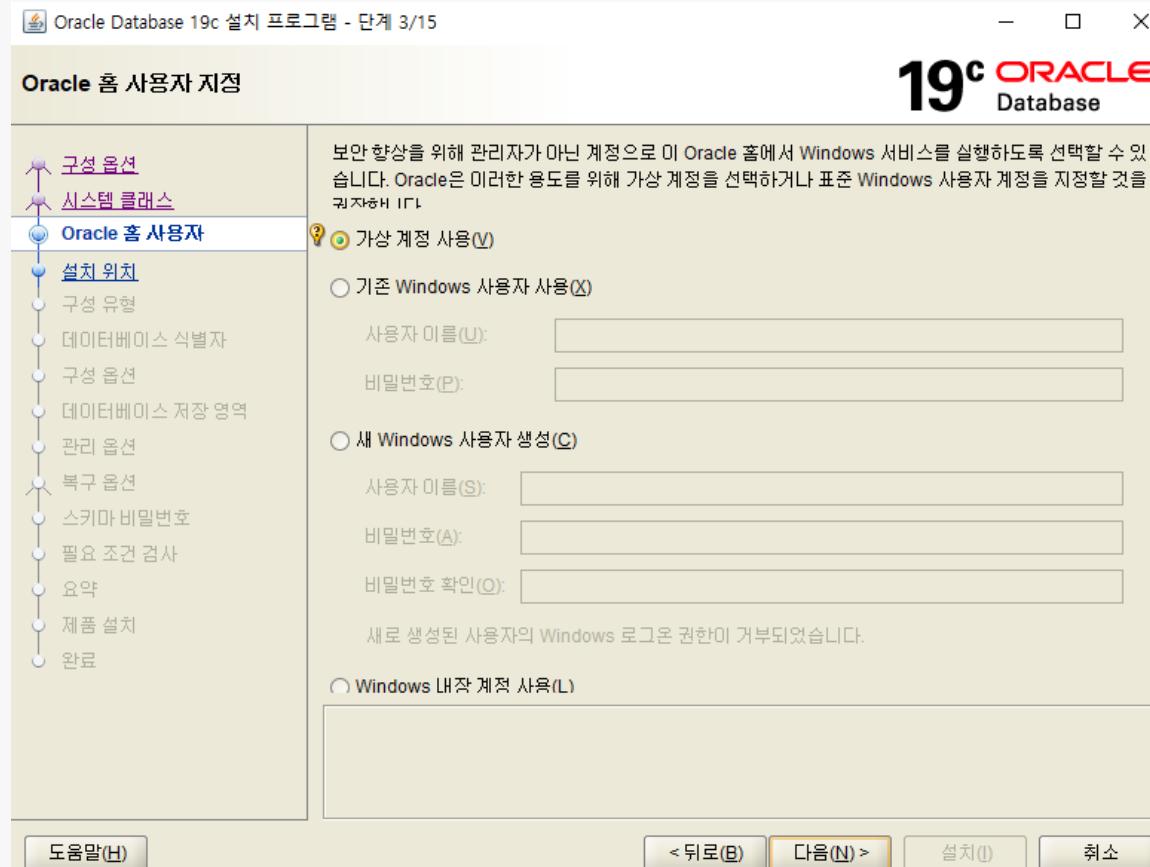
순차적으로 진행 (왼쪽에서 오른쪽 순으로)



# 03. 오라클 설치 (Windows 10)

(2021년 6월 기준)

순차적으로 진행 (왼쪽에서 오른쪽 순으로)



# 03. 오라클 설치 (Windows 10)

(2021년 6월 기준)

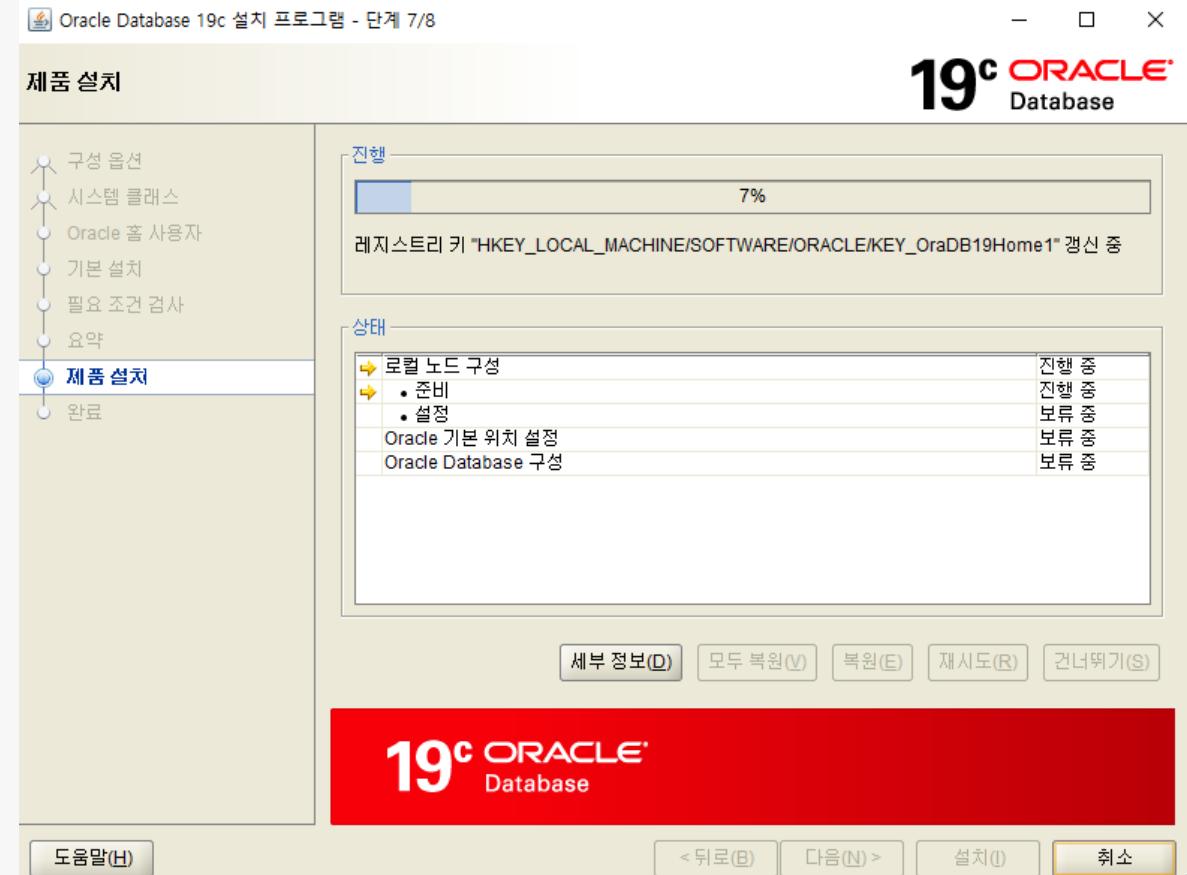
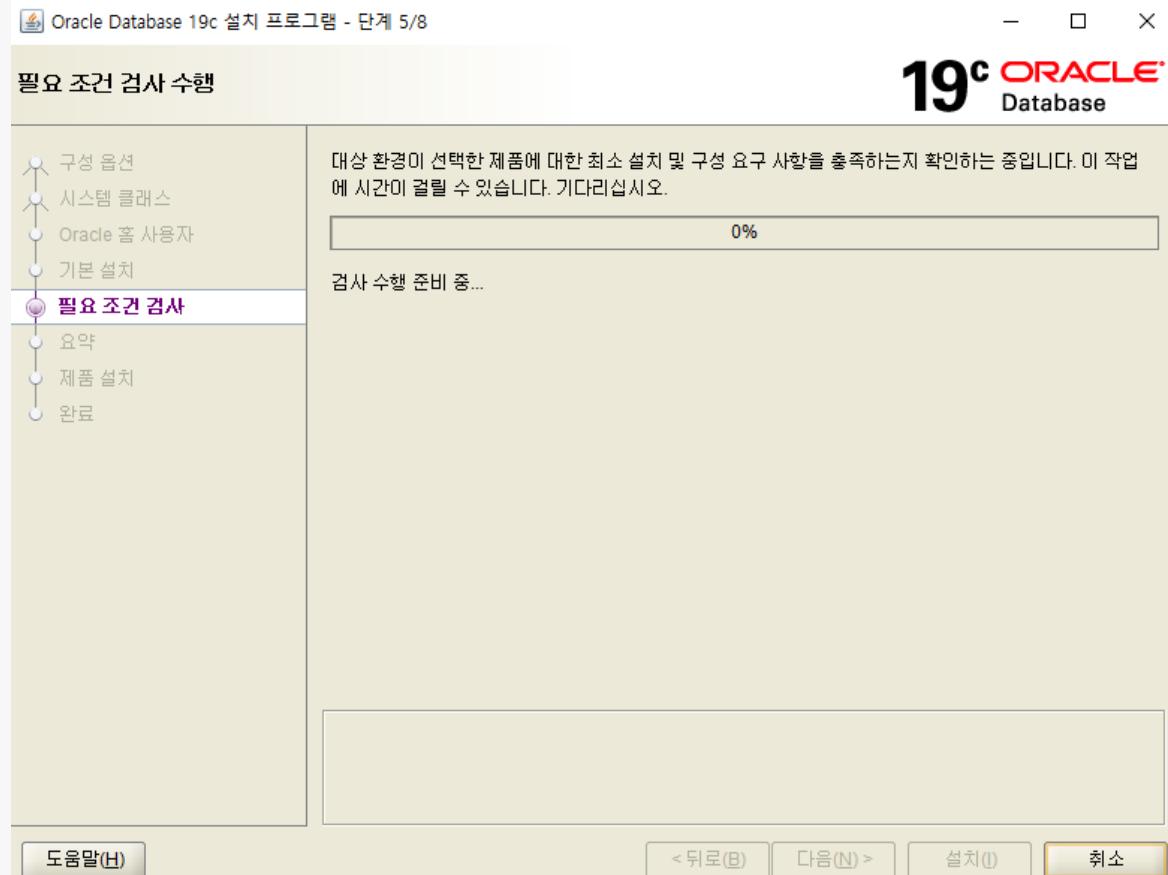
순차적으로 진행 (왼쪽에서 오른쪽 순으로)



# 03. 오라클 설치 (Windows 10)

(2021년 6월 기준)

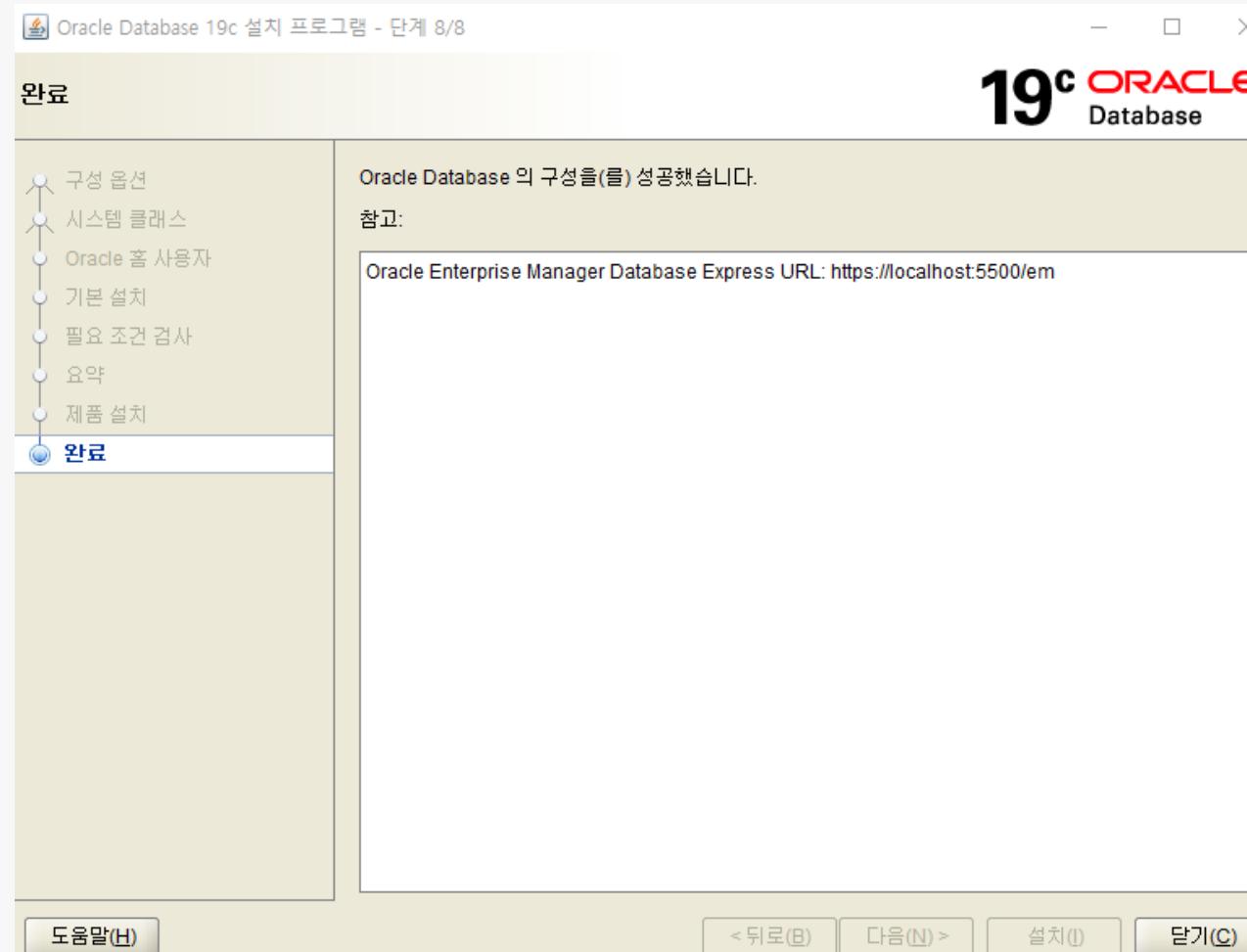
순차적으로 진행 (왼쪽에서 오른쪽 순으로)



# 03. 오라클 설치 (Windows 10)

(2021년 6월 기준)

## 순차적으로 진행



- 명령 프롬프트 실행

```
Microsoft Windows [Version 10.0.19042.1083]
(c) Microsoft Corporation. All rights reserved.
```

```
C:\Windows\system32>sqlplus "/ as sysdba"
SQL*Plus: Release 19.0.0.0.0 - Production on 수 7월 14 12:06:33 2021
```

```
Version 19.3.0.0.0 Copyright (c) 1982, 2019, Oracle. All rights reserved.
다음에 접속됨:
```

```
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
```

```
SQL>
```

- 유저 생성

```
SQL> create user scott identified by tiger;  
사용자가 생성되었습니다.
```

```
SQL> grant dba to scott;  
권한이 부여되었습니다.
```

```
SQL> connect scott/tiger;  
연결되었습니다.
```

```
SQL> show user;  
USER은 “SCOTT”입니다.
```

## SQL Developer 설치

<https://www.oracle.com/tools/downloads/sqldev-downloads.html>

The screenshot shows the Oracle website's "Tools / Downloads / SQL Developer Downloads" section. The main heading is "SQL Developer 20.4.1 Downloads". Below it, it says "Version 20.4.1.407.0006 - February 22, 2021". There are links for "Release Notes" and "Documentation". The download table has three columns: "Platform", "Download", and "Notes". Two rows are shown:

Platform	Download	Notes
Windows 64-bit with JDK 8 included	<a href="#">Download (423 MB)</a>	<ul style="list-style-type: none"><li>MD5: 73e7180ec8b494868c2fce4d99114630</li><li>SHA1: 0ad1cc75a28c0ac3eb9797bfc167bfc6fea3212e</li><li><a href="#">Installation Notes</a></li></ul>
Windows 32-bit/64-bit	<a href="#">Download (432 MB)</a>	<ul style="list-style-type: none"><li>MD5: 041709f01de2c6d176f37132089b61b8</li><li>SHA1: df90320a3a6e15df90fafb9d0c603317f3a68b84</li><li><a href="#">Installation Notes</a></li><li>JDK 8 or 11 required</li></ul>

# 04. SQL Developer 설치

(2021년 6월 기준)

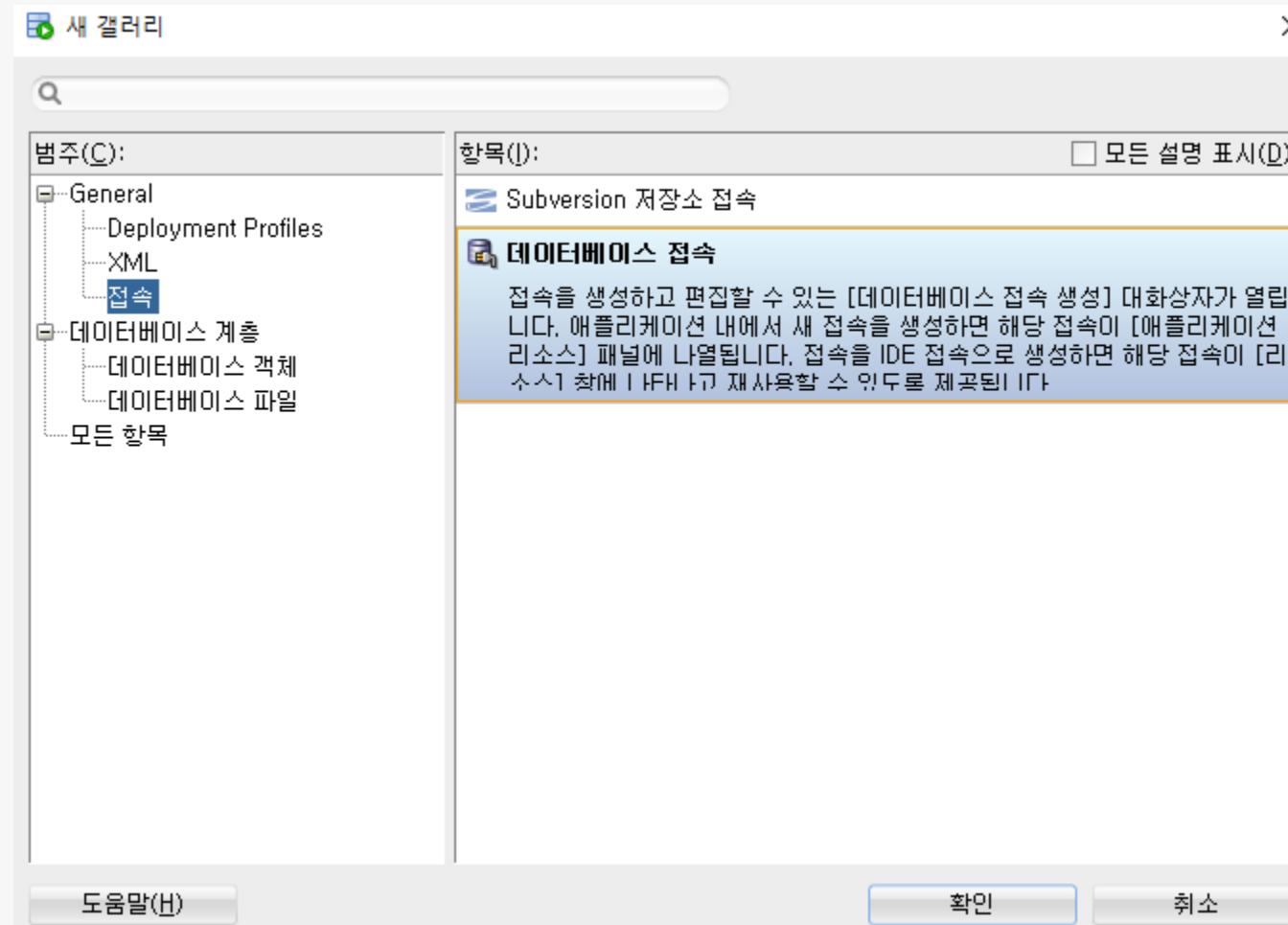
순차적으로 진행



# 04. SQL Developer 설치

(2021년 6월 기준)

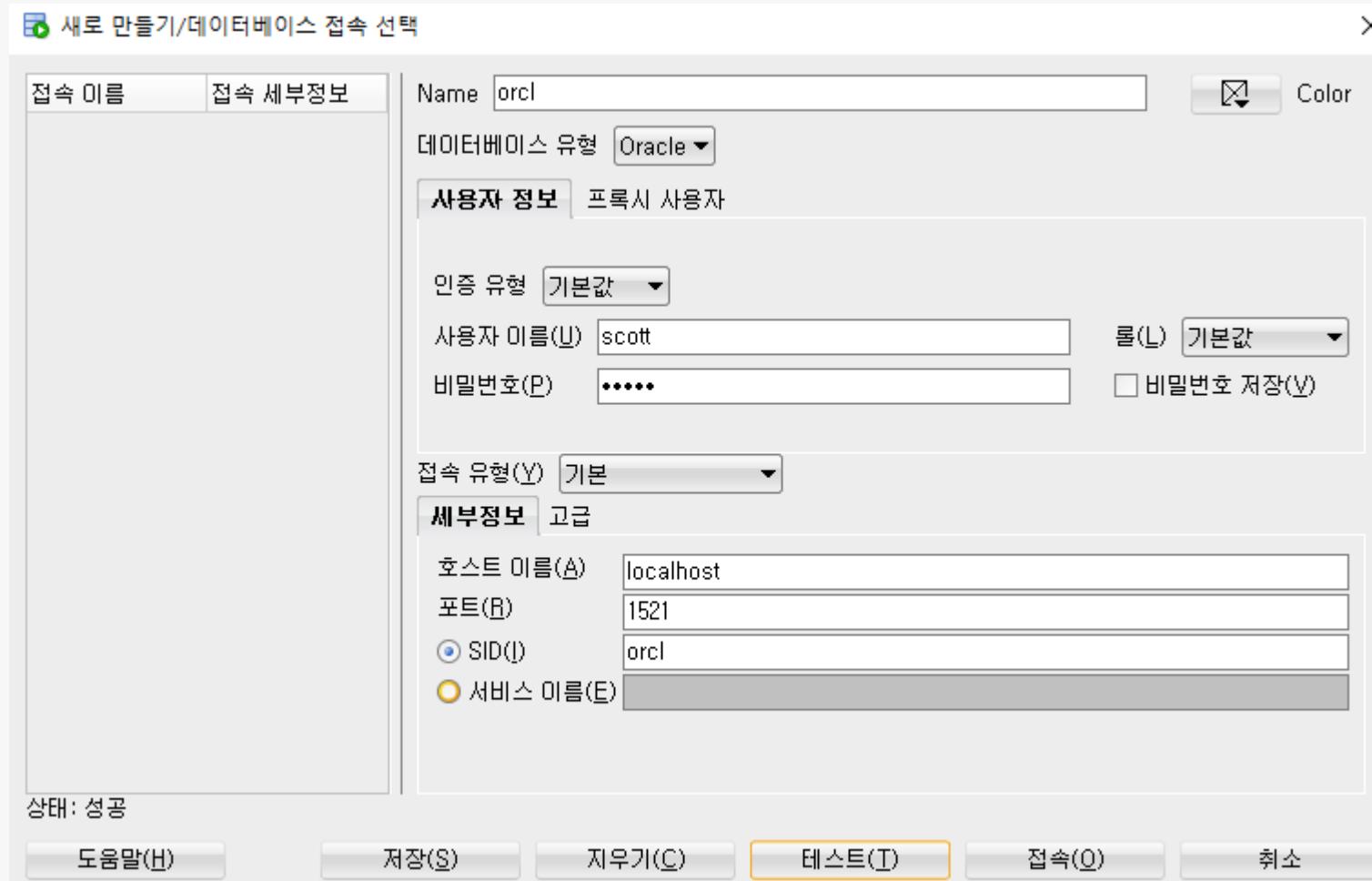
## 순차적으로 진행



# 04. SQL Developer 설치

(2021년 6월 기준)

## 순차적으로 진행



# 04. SQL Developer 설치

(2021년 6월 기준)

해당 코드 실행 (Ctrl + Enter)

The screenshot shows the Oracle SQL Developer interface with the following components:

- Toolbar:** Includes standard file operations like File, Edit, View, Tools, Database, Window, and Help.
- Left Sidebar:** Shows the connection tree under "접속" (Connections) with "Oracle 접속" expanded, showing "orcl". It also includes "데이터베이스 스키마 서비스 접속".
- Central Area:** A large "워크시트" (Worksheet) pane containing the following SQL code:

```
commit;

drop table salgrade;

create table salgrade
(
    grade   number(10),
    losal   number(10),
    hisal   number(10)
);

insert into salgrade values(1,700,1200);
insert into salgrade values(2,1201,1400);
insert into salgrade values(3,1401,2000);
insert into salgrade values(4,2001,3000);
insert into salgrade values(5,3001,9999);

commit;
```
- Bottom Right Panel:** A "스크립트 출력" (Script Output) window showing the results of the execution:

```
작업이 완료되었습니다.(0.437초)

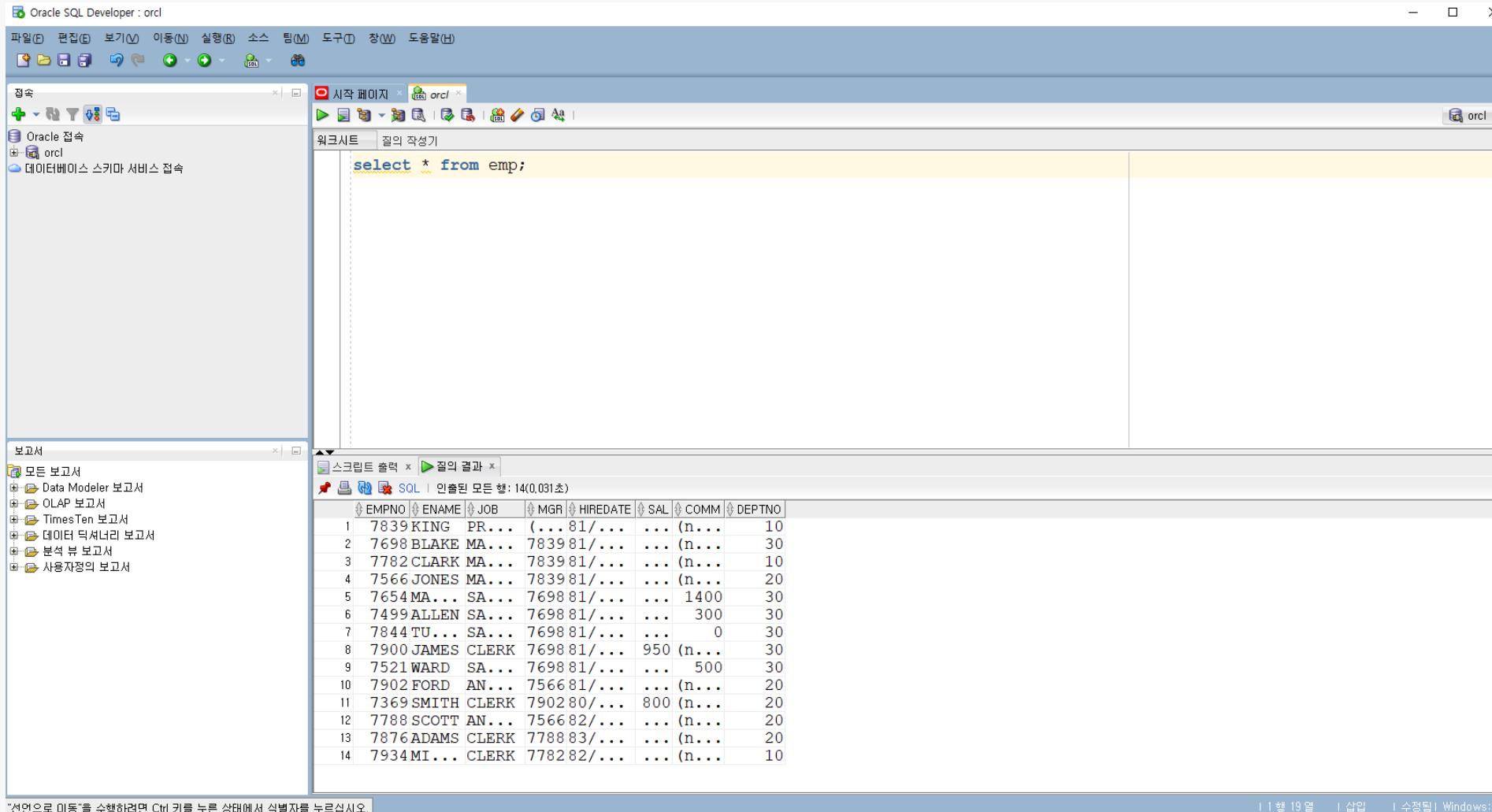
1 행 미 (가) 삽입되었습니다.

커밋 완료.
```
- Bottom Status Bar:** Displays "155 행 1 열" (155 rows 1 column), "삽입" (Insert), and "수정됨 Windows: CR".

# 04. SQL Developer 설치

(2021년 6월 기준)

해당 코드 실행 (Ctrl + Enter)



# 오라클 SQL 입문 문법

with  PL/SQL  
ORACLE®

# 01. 연산자

- 비교 연산자

연산자	의미
>	크다
<	작다

- 기타 비교 연산자

연산자	의미
BETWEEN AND	~ 사이에 있는
LIKE	일치하는 문자 패턴 검색

- 기타 비교 연산자

연산자	의미
>=	크거나 같다
<=	작거나 같다
=	같다
=	같지 않다
^=	같지 않다
<>	같지 않다

- 기타 비교 연산자

연산자	의미
IS NULL	NULL 값인지 여부
IN	값 리스트 중 일치하는 값 검색

## 02. 날짜 형식

- 현재 날짜 형식 확인

```
SQL> SELECT *
      FROM NLS_SESSION_PARAMETERS
     WHERE PARAMETER = 'NLS_DATE_FORMAT';
```

## PARAMETER

## VALUE

NLS\_DATE\_FORMAT  
RR/MM/DD

## 02. 날짜 형식

- 날짜 형식 수정

```
SQL> ALTER SESSION SET NLS_DATE_FORMAT='YY/MM/DD';
세션이 변경되었습니다.
```

### 03. ADD OR NOT 연산자

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

OR	TRUE	FALSE	NULL
TRUE	FALSE	TRUE	NULL

# 오라클 SQL 기초 문법

with



PL/SQL  
**ORACLE®**

# 016. 대소문자 변환 함수 배우기

- INTERVAL 표현식

함수의 종류	설명	
단일행 함수	정의	하나의 행을 입력받아 하나의 행을 반환하는 함수
	종류	문자, 숫자, 날짜, 변환, 일반 함수
다중 행 함수	정의	여러 개의 행을 입력받아 하나의 행을 반환하는 함수
	종류	그룹 함수

단일행 함수	함수
문자 함수	UPPER, LOWER, INITCAP, SUBSTR, LENGTH, CONCAT, INSTR, TRIM, LPAD, RPAD

# 027. INTERVAL 표현식

- INTERVAL 표현식

표현식	설명
INTERVAL '4' YEAR	An interval of 4 years 0 months
INTERVAL '123' YEAR	An interval of 123 years 0 months
INTERVAL '123' YEAR	An interval of 6 months
INTERVAL '600' MONTH(3)	An interval of 600 months
INTERVAL '400' DAY(3)	400 days
INTERVAL '10' HOUR	10 hours
INTERVAL '10' MINUTE	10 minutes
INTERVAL '4' DAY	4 days
INTERVAL '25' HOUR	25 hours
INTERVAL '40' MINUTE	'40' minute
INTERVAL '120' HOUR(3)	120 hours

## 030. 날짜 표현식

- 날짜를 문자로 출력할 때 사용할 수 있는 날짜 포맷

연도	RRRR, YYYY, RR, YY
월	MM, MON
일	DD
요일	DAY, DY

주	WW, IW, W
시간	HH, HH24
분	MI
초	SS

# 050. 데이터 분석 함수로 누적 데이터 출력하기(SUM OVER)

- ROWS

원도우 기준	원도우 방식	설명
ROWS	UNBOUNDED PRECEDING	맨 첫 번째 행을 가리킵니다.
	UNBOUNDED FOLLOWING	맨 마지막 행을 가리킵니다.
	CURRENT ROW	현재 행을 가리킵니다.

# 054. 데이터 분석 함수로 집계 결과 출력하기(GROUPING SETS)

- ROWS

GROUPING SETS	출력 결과
GROUPING SETS((deptno), (job), ())	부서 번호별 집계, 직업별 집계, 전체 집계
GROUPING SETS((deptno), (job))	부서 번호별 집계, 직업별 집계
GROUPING SETS((deptno, job), ())	부서 번호와 직업별 집계, 전체 집계
GROUPING SETS((deptno, job))	부서 번호와 직업별 집계

# 오라클 SQL 조인 문법

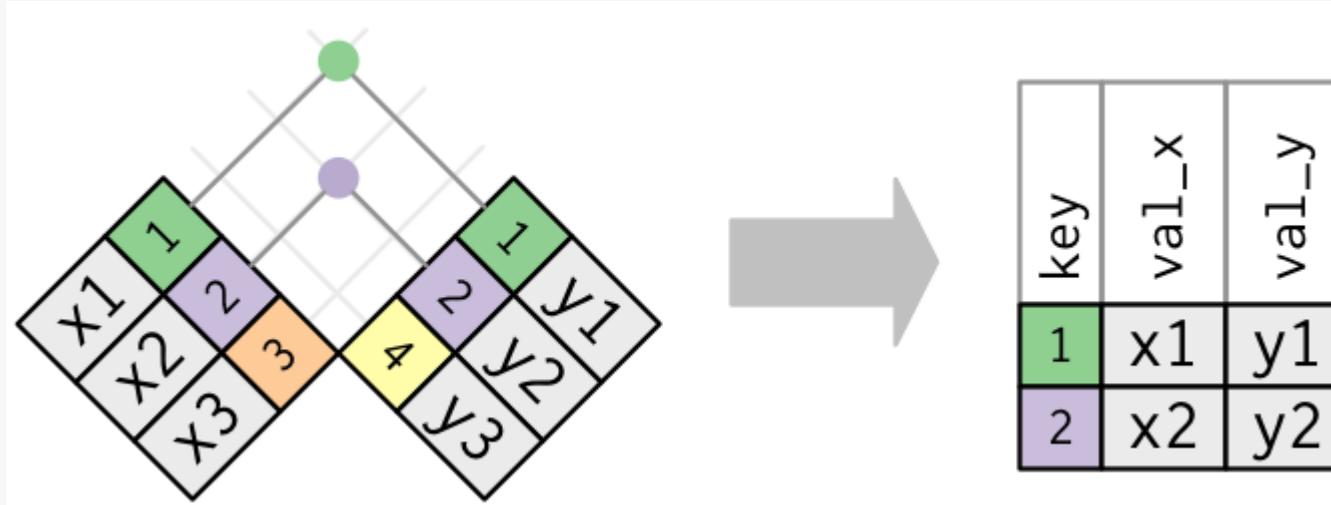
with



PL/SQL  
**ORACLE®**

# 01. EQUI JOIN

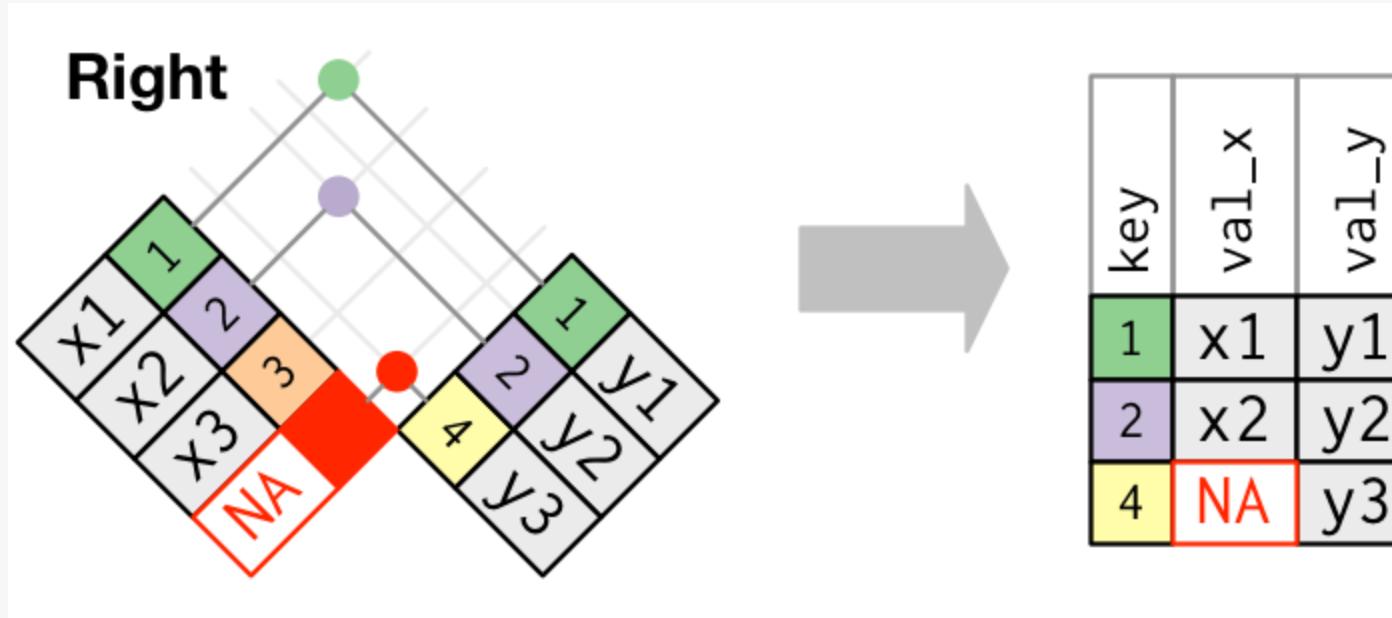
- INNER EQUI JOIN



출처: <https://r4ds.had.co.nz/relational-data.html>

## 02. RIGHT OUTER JOIN

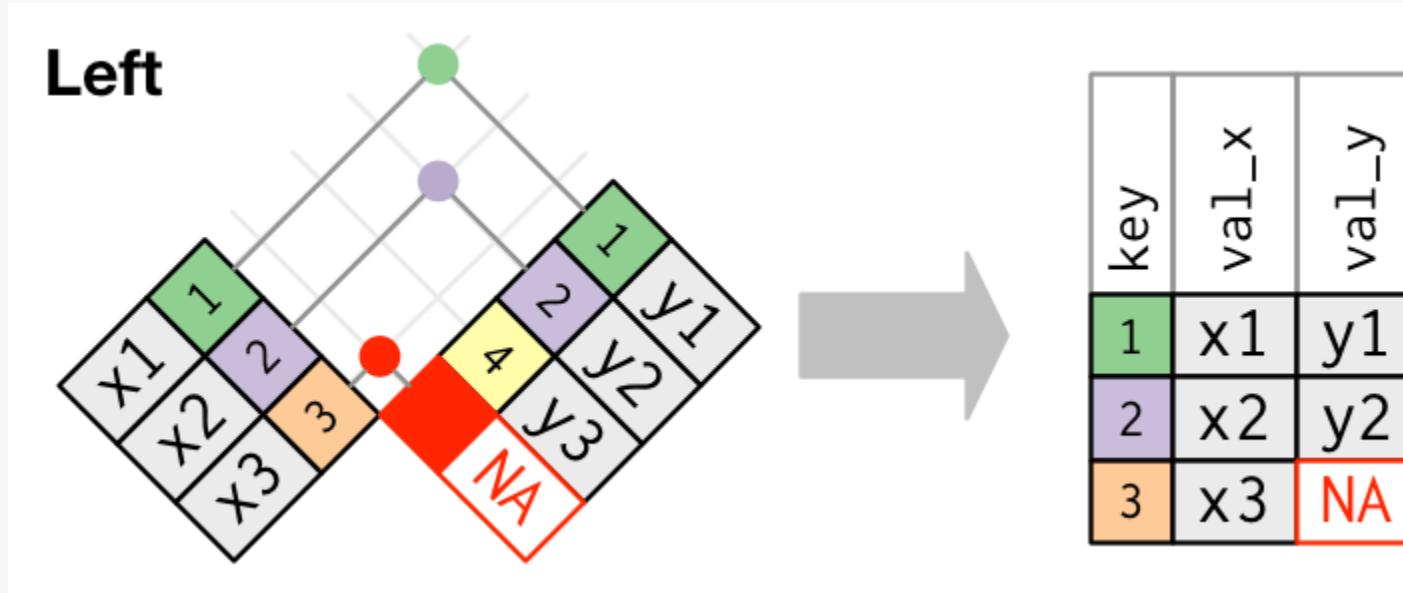
- RIGHT OUTER JOIN



출처: <https://r4ds.had.co.nz/relational-data.html>

# 03. LEFT OUTER JOIN

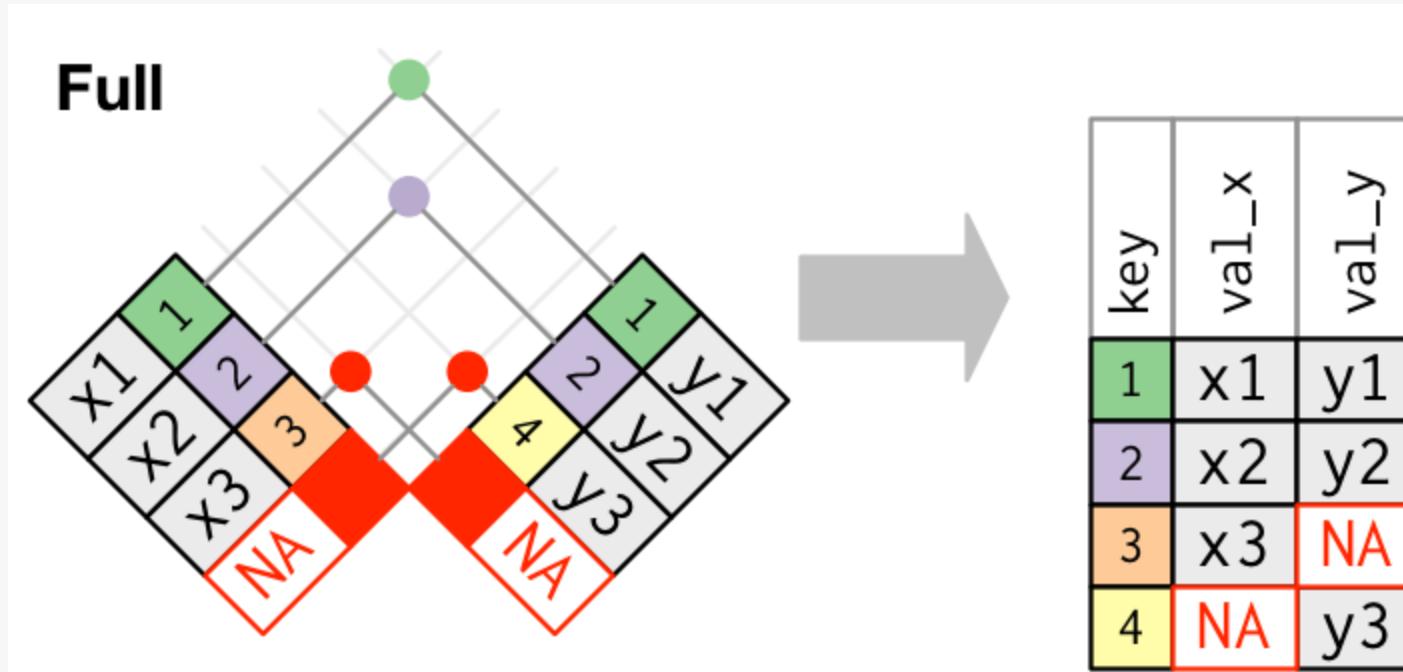
- LEFT OUTER JOIN



출처: <https://r4ds.had.co.nz/relational-data.html>

# 04. FULL JOIN

- FULL JOIN



출처: <https://r4ds.had.co.nz/relational-data.html>

# 05. 조인 문법의 종류

조인 문법의 종류	조인의 종류
ANSI/ISO SQL: 1999 standards	ON절을 사용한 JOIN
	LEFT/RIGHT/FULL OUTER JOIN
	USING절을 사용한 JOIN
	NATURAL JOIN
	CROSS JOIN

조인 문법의 종류	조인의 종류
오라클 조인	EQUI JOIN
	NON EQUI JOIN
	OUTER JOIN
	SELF JOIN

# 06. 집합 연산자

## 집합 연산자 작성 시 주의사항

UNION ALL 위쪽 쿼리와 아래쪽 쿼리 컬럼의 개수가 동일해야 한다.

UNION ALL 위쪽 쿼리와 아래쪽 쿼리 컬럼의 데이터 타입이 동일해야 한다.

결과로 출력되는 컬럼명은 위쪽 쿼리의 컬럼명으로 출력된다.

ORDER BY절은 제일 아래쪽 쿼리에만 작성할 수 있다.

## UNION 연산자가 UNION ALL과 다른 점

중복된 데이터를 하나의 고유한 값으로 출력한다.

첫 번째 컬럼의 데이터를 기준으로 내림차순으로 정렬하여 출력한다.

# 오라클 SQL 서브쿼리 문법

with



# 01. 서브쿼리의 종류 및 연산자

단일 행  
서브쿼리

다중 행  
서브쿼리

- 비교연산자(=, <, <=, >=, <>)
- IN, ANY, ALL, EXISTS 사용

## 02. 서브쿼리 종류 및 연산자

종류	설명
단일 행 서브 쿼리	서브 쿼리에서 메인 쿼리로 하나의 값이 반환됨
다중 행 서브 쿼리	서브 쿼리에서 메인 쿼리로 여러 개의 값이 반환됨
다중 컬럼 서브 쿼리	서브 쿼리에서 메인 쿼리로 여러 개의 컬럼 값이 반환됨

종류	연산자
단일 행 서브 쿼리	=, !=, >, <, >=, <=
다중 행 서브 쿼리	In, not in, >any, <any, >all, <all

### 03. 연산자 설명

연산자	설명
In	리스트의 값과 동일하다.
Not In	리스트의 값과 동일하지 않다.
>all	리스트에서 가장 큰 값보다 크다.
>any	리스트에서 가장 작은 값보다 크다
<all	리스트에서 가장 작은 값보다 작다
<any	리스트에서 가장 큰 값보다 작다.

# 04. SELECT문 6가지 절

SELECT문의 6가지 절	서브 쿼리 사용 여부	서브 쿼리 이름
SELECT	가능	스칼라(Scalar) 서브 쿼리
FROM	가능	IN LINE VIEW
WHERE	가능	서브 쿼리
GROUP BY	불가능	
HAVING	가능	서브 쿼리
ORDER BY	가능	스칼라(Scalar) 서브 쿼리

## 05. 쿼리 연습

- 2015년 전체 국가의 평균 수명을 계산하십시오.

# 05. 쿼리 연습

- 모든 데이터를 조회한다.
- 2015년 평균 기대 수명의 1.15배보다 높도록 설정한다.

▶ 질의 결과 ×

SQL | 인출된 모든 행: 5(0,002초)

	POP_ID	COUNTRY_CODE	YEAR	FERTILITY RATE	LIFE EXPECTANCY
1	21	AUS	2015	1.833	82.4512195121951
2	376	CHE	2015	1.54	83.1975609756098
3	356	ESP	2015	1.32	83.3804878048781
4	134	FRA	2015	2.01	82.6707317073171
5	170	HKG	2015	1.195	84.2780487804878

# 05. 쿼리 연습

- 서브 쿼리의 국가 테이블에 있는 capital 필드를 사용한다.
- cities 테이블에서 name, country\_code, urbanarea\_pop 필드를 조회한다.

질의 결과 ×

SQL | 50개의 행이 인출됨(0.01초)

	NAME	COUNTRY_CODE	URBANAREA_POP
1	Beijing	CHN	21516000
2	Dhaka	BGD	14543124
3	Tokyo	JPN	13513734
4	Moscow	RUS	12197596
5	Cairo	EGY	10230350
6	Kinshasa	COD	10130000
7	Jakarta	IDN	10075310
8	Seoul	KOR	9995784
9	Mexico City	MEX	8974724
10	Lima	PER	8852000

## 05. 쿼리 연습

- economies 데이터에서 country code, inflation rate, unemployment rate를 조회한다.
- Inflation rate 오름차순으로 정렬한다. 연도는 2015년이다.
- Alias 사용하지 않는다.
- countries 테이블 내 gov\_form 칼럼에서 Constitutional Monarchy 또는 'Republic'이 들어간 국가는 제외한다.

	CODE	INFLATION_RATE	UNEMPLOYMENT_RATE
1	AFG	-1.549	(null)
2	CHE	-1.14	3.178
3	PRI	-0.751	12
4	ROU	-0.596	6.812
5	BRN	-0.423	6.9
6	TON	-0.283	(null)
7	OMN	0.065	(null)
8	TLS	0.553	(null)
9	BEL	0.62	8.492
10	CAN	1.132	6.9

## 05. 쿼리 연습

- 첫번째 주석을 풀고 실행합니다.
- GROUP BY 코드를 변환하여 SELECT 내부의 하위 쿼리를 사용한다. 첫번째, 쿼리에서 GROUP BY 코드를 사용하여 주어진 결과와 일치하는 코드를 작성한다.
- 다시 city\_num 내림차순으로 결과를 정렬한 후, code 오름차순으로 정렬한다.



The screenshot shows a database query results window with the following details:

- WindowTitle: 질의 결과
- Toolbar Buttons: Refresh, Stop, SQL, and a status message: 50개의 행이 인출됨(0.009초)
- Table Headers: COUNTRY and CITIES\_NUM
- Table Data:

COUNTRY	CITIES_NUM
1 China	36
2 India	18
3 Japan	11
4 Brazil	10
5 Pakistan	9
6 United States	9
7 Indonesia	7
8 Russian Federation	7
9 South Korea	7
10 Iran	6

## 05. 쿼리 연습

- 2015년의 continent 그룹별로 하여 continent, inflation\_rate의 최댓값을 조회한다.
- 각 대륙별 inflation\_rate가 가장 높은 나라를 추출하는 코드를 작성하도록 한다.

#	NAME	CONTINENT	INFLATION_RATE
1	Afghanistan	Asia	-1.549
2	Netherlands	Europe	0.22
3	Albania	Europe	1.896
4	Algeria	Africa	4.784
5	Angola	Africa	10.287
6	Antigua and Barbuda	North America	0.969
7	United Arab Emirates	Asia	4.07
8	Argentina	South America	(null)
9	Armenia	Asia	3.731
10	Australia	Oceania	1.461

#	CONTINENT	MAX_INF
1	North America	7.524
2	Oceania	9.784
3	Africa	21.858
4	Asia	39.403
5	Europe	48.684
6	South America	121.738

#	NAME	CONTINENT	INFLATION_RATE
1	Haiti	North America	7.524
2	Yemen	Asia	39.403
3	Malawi	Africa	21.858
4	Nauru	Oceania	9.784
5	Ukraine	Europe	48.684
6	Venezuela	South America	121.738

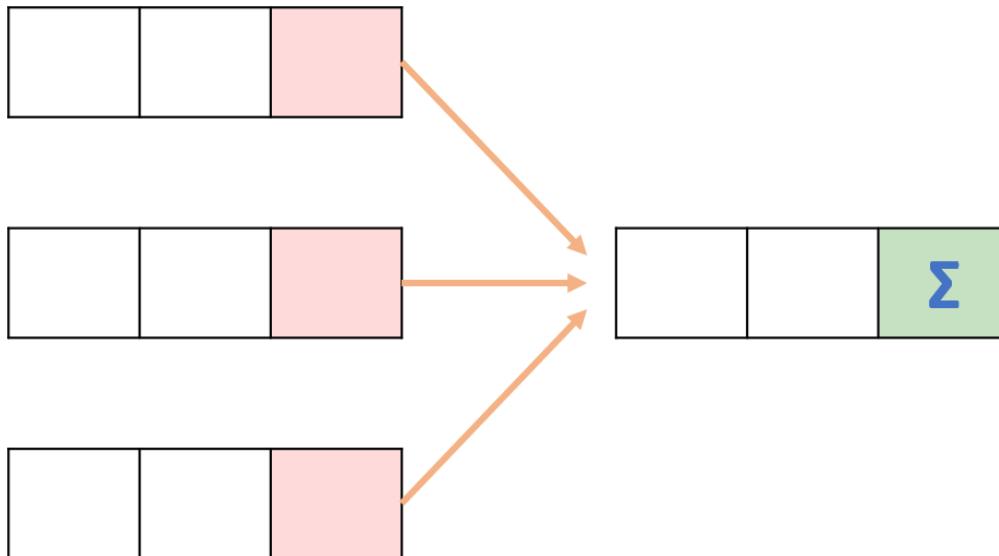
# 오라클 SQL 윈도우 함수 문법

with

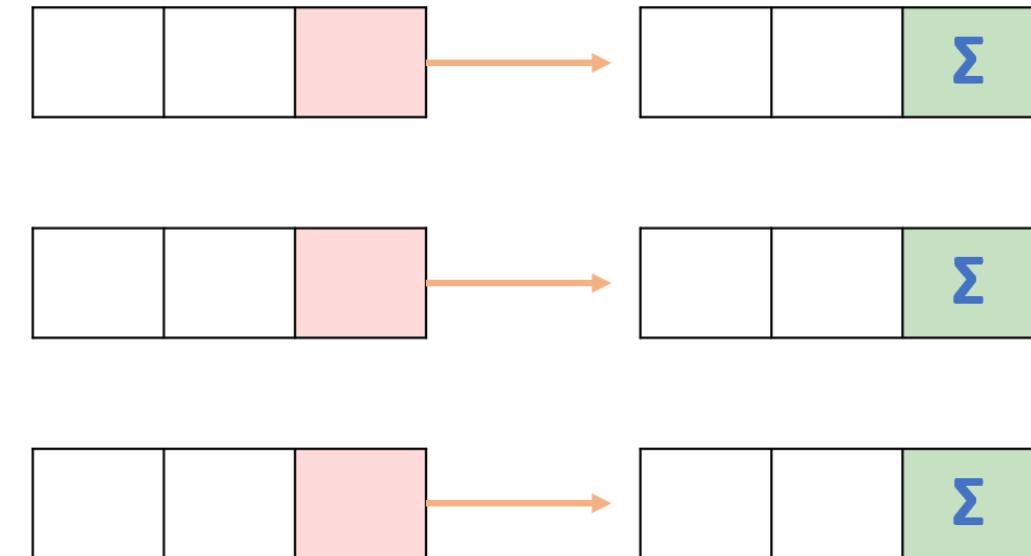


# 01. 기본 개념

## Aggregation



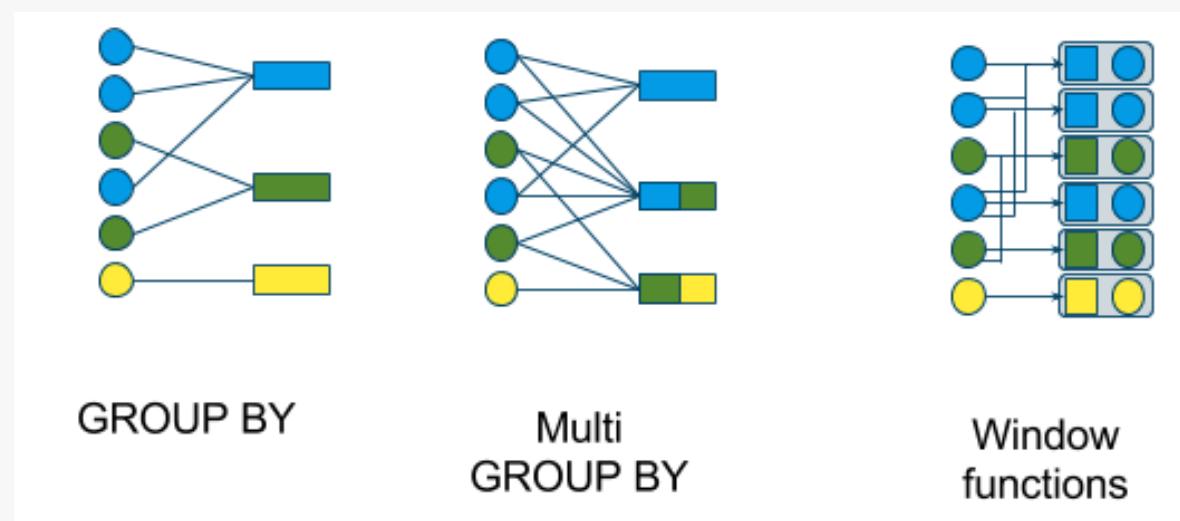
## Window Function



출처: <https://towardsdatascience.com/sql-window-function-demonstrated-with-real-interview-questions-from-leetcode-e83e28edaabc>

# 01. 기본 개념

- 현재 행과 관련된 행 집합에서 쿼리 작업 수행
- GROUP BY 집계 함수와 유사하지만, 결괏값에서 모든 행이 존재 할 수
- Ranking, 누적 계산, 이동 평균 계산 진행 가능



출처: <https://sundaskhalid.medium.com/sql-window-function-vs-group-by-b246d14223d2>

## 02. 쿼리 연습

- 각 행에 숫자를 1, 2, 3, …, N 행태로 추가한다.
- ROW\_N 기준으로 오름차순으로 진행한다.

▶ 질의 결과 x

SQL | 50개의 행이 인출됨(0.002초)

ROW_N	YEAR	CITY	SPORT	DISCIPLINE	ATHLETE	COUNTRY	GENDER	EVENT	MEDAL
1	1900	Paris	Athletics	Athletics	SHELDON Richard	USA	Men	Discus Throw	Bronze
2	1900	Paris	Athletics	Athletics	BAUER Rudolf	HUN	Men	Discus Throw	Gold
3	1900	Paris	Athletics	Athletics	JANDA Frantisek	BOH	Men	Discus Throw	Silver
4	1900	Paris	Athletics	Athletics	MCCRACKEN Josiah	USA	Men	Hammer Throw	Bronze
5	1900	Paris	Athletics	Athletics	FLANAGAN John Jesus	USA	Men	Hammer Throw	Gold
6	1900	Paris	Athletics	Athletics	HARE Thomas Truxton	USA	Men	Hammer Throw	Silver
7	1900	Paris	Athletics	Athletics	GNCZY Lajos	HUN	Men	High Jump	Bronze
8	1900	Paris	Athletics	Athletics	BAXTER Irving	USA	Men	High Jump	Gold
9	1900	Paris	Athletics	Athletics	LEAHY Patrick Joseph	GBR	Men	High Jump	Silver
10	1900	Paris	Athletics	Athletics	SHELDON Lewis Pendleton	USA	Men	High Jump Standing	Bronze

## 02. 쿼리 연습

- 하계 올림픽이 열린 각 연도에 번호를 할당한다.
- 가장 최근 연도를 가진 행이 더 낮은 숫자를 갖도록 한다.

질의 결과 x

SQL | 인출!

	YEAR	ROW_N
1	1896	1
2	1900	2
3	1904	3
4	1908	4
5	1912	5
6	1920	6
7	1924	7
8	1928	8
9	1932	9
10	1936	10

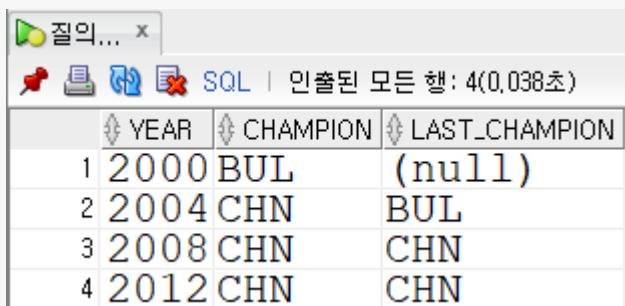
## 02. 쿼리 연습

- 각 운동선수들이 획득한 메달 개수를 내림차순으로 정렬 한다.
- 각 선수들의 랭킹을 추가한다.

	MEDALS	ATHLETE	ROW_N
1	22	PHELPS Michael	1
2	18	LATYNINA Larisa	2
3	15	ANDRIANOV Nikolay	3
4	13	ONO Takashi	4
5	13	SHAKHLIN Boris	5
6	13	MANGIAROTTI Edoardo	6
7	12	NURMI Paavo	7
8	12	COUGHLIN Natalie	8
9	12	THOMPSON Jenny	9
10	12	FISCHER Birgit	10

## 02. 쿼리 연습

- 남자 69KG 역도 경기에서 매년 금메달리스트 조회하도록 한다.
  - Discipline: Weightlifting
  - Event: 69kg
  - Gender: Men
  - Medal: Gold
- 기존 쿼리에서 매년 전년도 챔피언도 같이 조회하도록 한다
  - 이 때, LAG( ) 함수를 사용한다.



The screenshot shows a MySQL Workbench interface with a query editor window. The title bar says '질의...' (Query). The toolbar includes icons for connection, schema, and SQL. The status bar indicates 'SQL | 인출된 모든 행: 4(0.038초)'. The results pane displays a table with three columns: YEAR, CHAMPION, and LAST\_CHAMPION. The data is as follows:

	YEAR	CHAMPION	LAST_CHAMPION
1	2000	BUL	(null)
2	2004	CHN	BUL
3	2008	CHN	CHN
4	2012	CHN	CHN

## 03. PARTITION BY

- PARTITION BY는 테이블을 각 칼럼의 값에 따라서 분할을 합니다.
- ROW\_NUMBER는 각 파티션에 따라 새롭게 정의를 내립니다.
- LAG는 이전 행이 동일한 파티션에 있는 경우에만 행의 이전 값을 가져옵니다.

# 03. PARTITION BY

## Query

```
WITH Discus_Gold_Medal AS (
    SELECT
        Year, Event, Country AS Champion
    FROM summer_medals
    WHERE
        Year IN (2004, 2008, 2012)
        AND Gender = 'Men' AND Medal = 'Gold'
        AND Event IN ('Discus Throw', 'Triple Jump')
        AND Gender = 'Men')
```

```
SELECT
    YEAR, Event, Champion,
    LAG(Champion) OVER
        (ORDER BY Event ASC, Year ASC) AS Last_Champion
FROM Discus_Gold_Medal
ORDER BY Event ASC, Year ASC;
```

## Result

YEAR	EVENT	CHAMPION	LAST_CHAMPION
1 2004	Discus Throw	LTU	(null)
2 2008	Discus Throw	EST	LTU
3 2012	Discus Throw	GER	EST
4 2004	Triple Jump	SWE	GER
5 2008	Triple Jump	POR	SWE
6 2012	Triple Jump	USA	POR

# 03. PARTITION BY

## Query

```
WITH Discus_Gold_Medal AS (
    SELECT
        Year, Event, Country AS Champion
    FROM summer_medals
    WHERE
        Year IN (2004, 2008, 2012)
        AND Gender = 'Men' AND Medal = 'Gold'
        AND Event IN ('Discus Throw', 'Triple Jump')
        AND Gender = 'Men')
```

```
SELECT
    YEAR, Event, Champion,
    LAG(Champion) OVER
    (PARTITION BY Event ORDER BY Event ASC, Year ASC) AS
    Last_Champion
FROM Discus_Gold_Medal
ORDER BY Event ASC, Year ASC;
```

## Result

YEAR	EVENT	CHAMPION	LAST_CHAMPION
1 2004	Discus Throw	LTU	(null)
2 2008	Discus Throw	EST	LTU
3 2012	Discus Throw	GER	EST
4 2004	Triple Jump	SWE	(null)
5 2008	Triple Jump	POR	SWE
6 2012	Triple Jump	USA	POR

## 04. 쿼리 연습

- 성별에 따라 각 이벤트의 이전 챔피언을 반환한다.
- 성별에 따른 현재 챔피언과 이전 챔피언을 반환한다.
- 이번에는 경기종목을 2개 추가한다.
- ('100M', '10000M')
- 이 때에는 IN() 함수를 사용한다.

질의 결과 x

SQL | 인출된 모든 행: 15(0,012초)

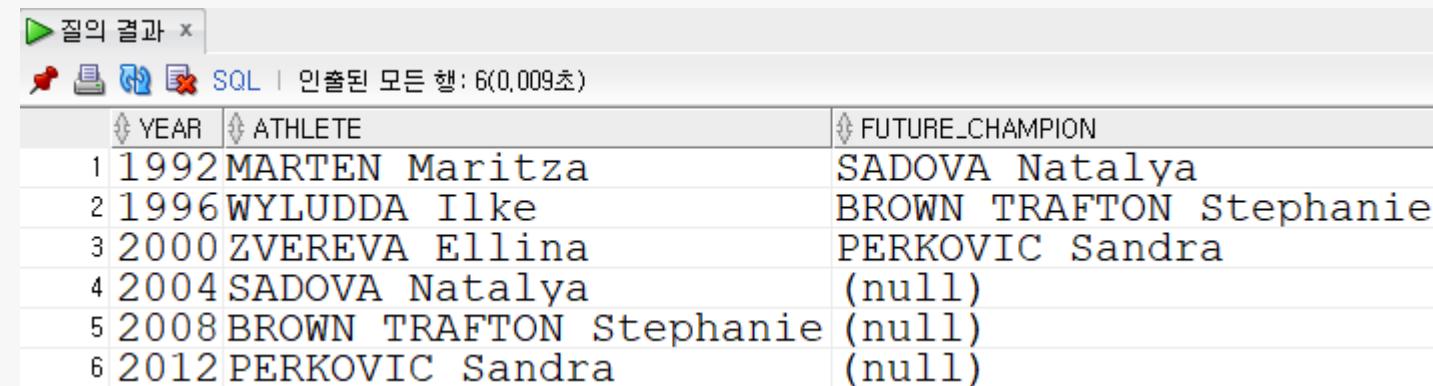
	GENDER	YEAR	EVENT	CHAMPION	LAST_CHAMPION
1	Men	2000	10000M	ETH	(null)
2	Men	2004	10000M	ETH	ETH
3	Men	2008	10000M	ETH	ETH
4	Men	2012	10000M	GBR	ETH
5	Women	2000	10000M	ETH	(null)
6	Women	2004	10000M	CHN	ETH
7	Women	2008	10000M	ETH	CHN
8	Women	2012	10000M	ETH	ETH
9	Men	2000	100M	USA	(null)
10	Men	2004	100M	USA	USA
11	Men	2008	100M	JAM	USA
12	Men	2012	100M	JAM	JAM
13	Women	2004	100M	BLR	(null)
14	Women	2008	100M	JAM	BLR
15	Women	2012	100M	JAM	JAM

# 05. Relative vs Absolute

- **Relative**
  - ✓ LAG(column, n) LAG()는 현재 행 앞의 행에 있는 열의 값을 반환합니다.
  - ✓ Lead(column, n) 현재 행 뒤의 행에 있는 열의 값을 반환합니다.
- **Absolute**
  - ✓ FIRST\_VALUE(column) 테이블 또는 파티션의 첫번째 값을 반환합니다.
  - ✓ LAST\_VALUE(column) 테이블 또는 파티션의 마지막 값을 반환합니다.

## 06. 쿼리 연습

- 각 연도마다 현재 금메달리스트와 3개 대회 이후의 메달리스트를 같이 조회합니다.



The screenshot shows a database query results window with the following data:

YEAR	ATHLETE	FUTURE_CHAMPION
1 1992	MARTEN Maritza	SADOVA Natalya
2 1996	WYLUDDA Ilke	BROWN TRAFTON Stephanie
3 2000	ZVEREVA Ellina	PERKOVIC Sandra
4 2004	SADOVA Natalya	(null)
5 2008	BROWN TRAFTON Stephanie	(null)
6 2012	PERKOVIC Sandra	(null)

## 06. 쿼리 연습

- 이번에는 FIRST\_VALUE를 활용하여 모든 선수를 조회한다.
- 동시에 첫번째 선수를 조회하도록 합니다.
- 모든 선수 조회 시, 알파벳 순으로 정렬합니다.
- Medal = 'Gold'
- Gender = 'Men'

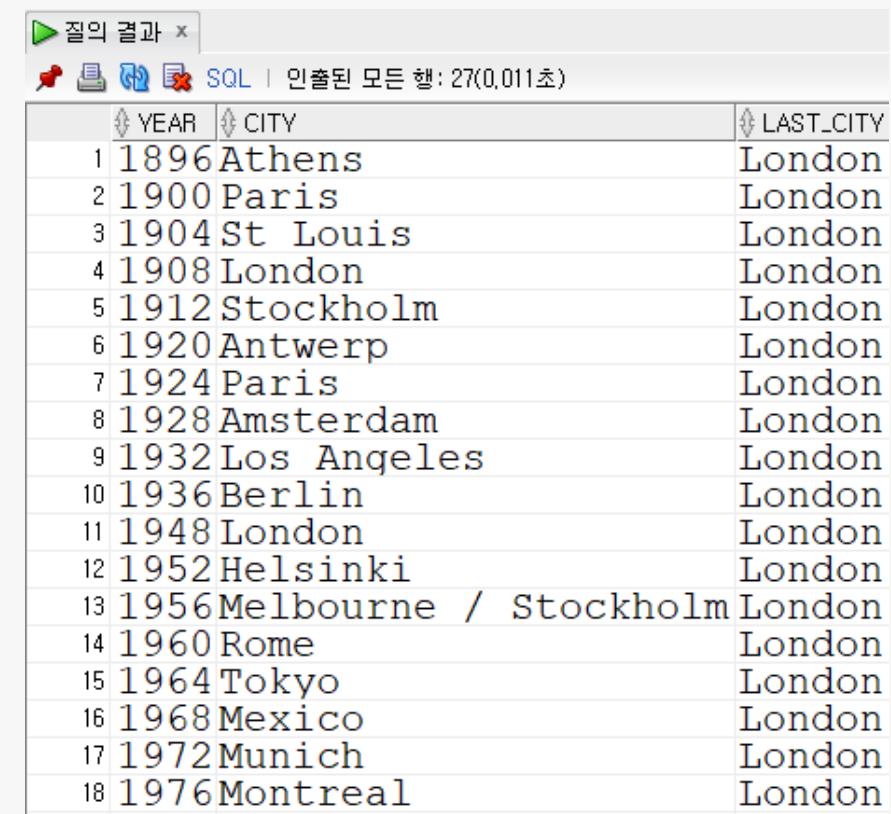
질의 결과 x

SQL | 50개의 행이 인출됨(0,018초)

ATHLETE	FIRST_ATHLETE
1 AABYE Edgar	AABYE Edgar
2 AALTONEN Paavo Johannes	AABYE Edgar
3 AAS Thomas Valentin	AABYE Edgar
4 ABALMASAU Aliaksei	AABYE Edgar
5 ABALO Luc	AABYE Edgar
6 ABANDA ETONG Patrice	AABYE Edgar
7 ABARCA Jose Maria	AABYE Edgar
8 ABASCAL GARCIA Alejandro	AABYE Edgar
9 ABATI Joel	AABYE Edgar
10 ABBAGNALE Agostino	AABYE Edgar
11 ABBAGNALE Carmine	AABYE Edgar
12 ABBAGNALE Giuseppe	AABYE Edgar
13 ABDOULLAYEV Mahamatkodir	AABYE Edgar
14 ABDUL Hamid	AABYE Edgar
15 ABDUL Rashid I	AABYE Edgar
16 ABDUL Rashid III	AABYE Edgar
17 ABDUL Rashid IV	AABYE Edgar
18 ABDUL Waheed	AABYE Edgar

## 06. 쿼리 연습

- 각 올림픽 경기가 열렸던 해외 도시를 조회합니다.
- 본 데이터에서 올림픽 경기가 열렸던 마지막 도시를 조회합니다.
- LAST\_VALUE()



The screenshot shows a database query results window with the following details:

- WindowTitle: 질의 결과 x
- Toolbar Buttons: Refresh, Save, Print, SQL (selected), Exit
- Text: SQL | 인출된 모든 행: 27(0,011초)
- Table Headers: YEAR, CITY, LAST\_CITY
- Table Data:

YEAR	CITY	LAST_CITY
1	1896 Athens	London
2	1900 Paris	London
3	1904 St Louis	London
4	1908 London	London
5	1912 Stockholm	London
6	1920 Antwerp	London
7	1924 Paris	London
8	1928 Amsterdam	London
9	1932 Los Angeles	London
10	1936 Berlin	London
11	1948 London	London
12	1952 Helsinki	London
13	1956 Melbourne / Stockholm	London
14	1960 Rome	London
15	1964 Tokyo	London
16	1968 Mexico	London
17	1972 Munich	London
18	1976 Montreal	London

## 07. Ranking 함수

- ROW\_NUMBER()는 두 행의 값이 동일한 경우에도 항상 고유한 번호를 할당한다.
- RANK() 동일한 값을 가진 행에 같은 번호를 할당하고 이러한 경우 다음 숫자를 건너뛴다.
- DENSE\_RANK() 값이 동일한 행에 같은 숫자를 할당함. 다음 숫자를 건너뛰지 않는다.

```
SELECT
    Country, COUNT(DISTINCT Year) AS Games
FROM summer_medals
Where
    Country IN ('GBR', 'DEN', 'FRA',
                 'ITA', 'AUT', 'BEL',
                 'NOR', 'JPN', 'KOR')
GROUP BY Country
ORDER BY GAMES DESC;
```

	COUNTRY	GAMES
1	GBR	27
2	FRA	26
3	DEN	26
4	ITA	25
5	BEL	24
6	AUT	24
7	NOR	22
8	JPN	20
9	KOR	15

# 07. Ranking 함수

```
WITH Country_Games AS (
    SELECT
        Country, COUNT(DISTINCT Year) AS Games
    FROM summer_medals
    WHERE
        Country IN ('GBR', 'DEN', 'FRA',
                    'ITA', 'AUT', 'BEL',
                    'NOR', 'JPN', 'KOR')
    GROUP BY Country
    ORDER BY GAMES DESC)

SELECT
    Country, Games,
    ROW_NUMBER()
        OVER (ORDER BY Games DESC) AS Row_N,
    RANK()
        OVER (ORDER BY Games DESC) AS Rank_N,
    DENSE_RANK()
        OVER (ORDER BY Games DESC) AS Dense_Rank_N
FROM Country_Games
ORDER BY Games DESC, Country ASC;
```

질의 결과 x

SQL | 인출된 모든 행: 9(0.009초)

COUNTRY	GAMES	ROW_N	RANK_N	DENSE_RANK_N
1 GBR	27	1	1	1
2 DEN	26	2	2	2
3 FRA	26	3	2	2
4 ITA	25	4	4	3
5 AUT	24	5	5	4
6 BEL	24	6	5	4
7 NOR	22	7	7	5
8 JPN	20	8	8	6
9 KOR	15	9	9	7

## 08. 쿼리연습

- 이번에는 국가별 Gold 메달 개수를 알파벳 순서대로 작업을 하도록 한다.
- 이 때, Gold 메달 누적 개수를 같이 조회하도록 한다.

질의 결과 ×

SQL | 50개의 행이 인출됨(0.005초)

	COUNTRY	MEDALS	CUM_MEDALS
1	ALG	2	2
2	ARG	47	49
3	AUS	159	208
4	AUT	6	214
5	AZE	6	220
6	BAH	11	231
7	BEL	2	233
8	BLR	16	249
9	BRA	46	295
10	BUL	8	303
11	CAN	19	322
12	CHI	3	325
13	CHN	221	546
14	CMR	20	566
15	COL	2	568
16	CRO	30	598

## 08. 쿼리연습

- 이번에는 한국과 일본 국가만 조회될 수 있도록 쿼리 작성한다.
- 조회된 기록 중, 가장 많은 메달을 기록 갯수가 나타나도록 한다.

▶ 질의 결과 x

SQL | 인출된 모든 행: 8(0.007초)

	COUNTRY	YEAR	MEDALS	MAX_MEDALS
1	JPN	2000	5	5
2	JPN	2004	21	21
3	JPN	2008	23	23
4	JPN	2012	7	23
5	KOR	2000	12	12
6	KOR	2004	14	14
7	KOR	2008	41	41
8	KOR	2012	18	41

## 08. 쿼리연습

- 국가별 금메달을 기준으로 랭킹을 구하는 쿼리 작성

▶ 질의 결과 X

SQL | 인출된 모든 행: 9(0.011초)

	COUNTRY	YEAR	MEDAL_CNT	RANK
1	FRA	2004	21	2
2	FRA	2008	25	3
3	FRA	2012	30	3
4	GBR	2004	17	3
5	GBR	2008	31	2
6	GBR	2012	48	1
7	GER	2004	41	1
8	GER	2008	42	1
9	GER	2012	45	2

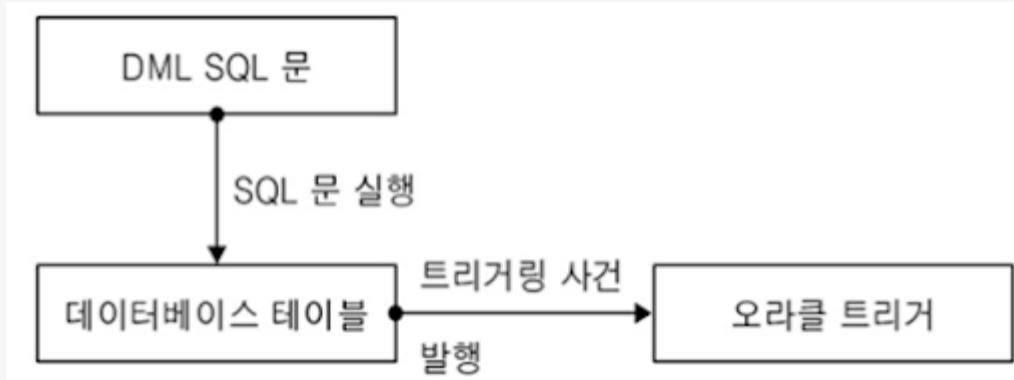
# 오라클 트리거 개념

with



# 01. 기본 개념

- 특정 사건이 발생했을 때, 자동으로 호출되는 코드
- 시스템이 호출하므로 인수를 전달할 수 없고 리턴값도 반환할 수 없음



< 트리거 일반적인 흐름도 >

# 01. 기본 개념

CREATE [OR REPLACE] TRIGGER 트리거 이름

BEFORE | AFTER | INSTEAD OF

INSERT OR UPDATE OR DELETE [OF 컬럼]

ON 테이블명

[FOR EACH ROW]

BEGIN

명령;

END

- 트리거 이름은 TR\_ 접두어를 붙임

# 01. 기본 개념

- :NEW and :OLD 구문 비교

	INSERT	UPDATE	DELETE
:NEW	VALID	VALID	INVALID
:OLD	INVALID	VALID	VALID

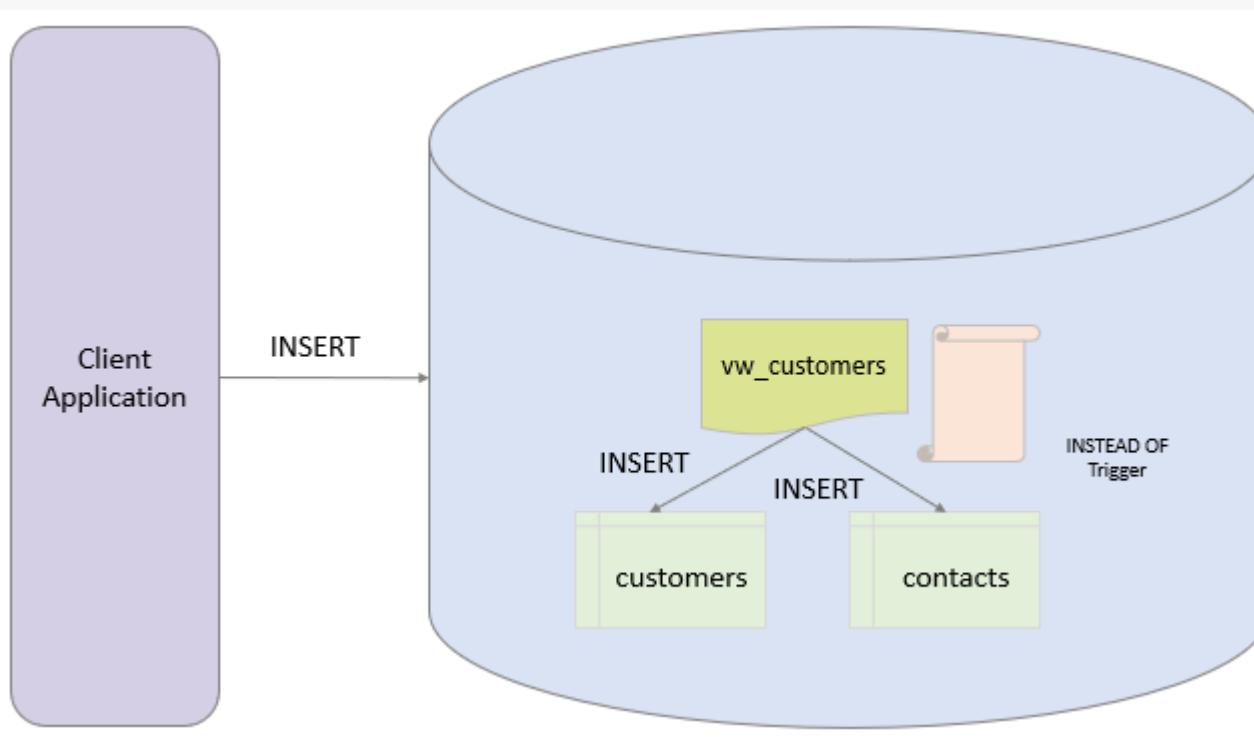
- :NEW – INSERT, UPDATE 시 변경된 후의 레코드를 가짐
- :OLD – UPDATE DELETE 시 삭제 또는 변경 전의 레코드를 가짐

## 02. AFTER, BEFORE, INSTEAD OF TRIGGER

- 트리거 실행 시점
  - AFTER 트리거: 갱신이 완료된 상태에서 호출됨
  - BEFORE 트리거: 삽입전에 발생하는 것이며, 따라서 수정 기회가 있음
  - INSTEAD OF 트리거: 복합적인 VIEW에서 DML 이벤트가 발생할 때 사용함
    - 지정된 이벤트에 VIEW 테이블을 수정하는 대신 기본 테이블을 직접 수정 가능

## 02. INSTEAD OF TRIGGER

- INSTEAD OF 트리거: 복합적인 VIEW에서 DML 이벤트가 발생할 때 사용함
  - 지정된 이벤트에 VIEW 테이블을 수정하는 대신 기본 테이블을 직접 수정 가능



출처: <https://www.oracletutorial.com/plsql-tutorial/oracle-instead-of-triggers/>