

주말과제 - DBMS Connection & 시각화

DBMS Connection & 시각화

[개요](#)

[요구사항](#)

[DBMS Connection 개념 및 ODBC/JDBC 용어 정리](#)

[Connection을 맺기 위한 주요 구성 요소](#)

[Client - DBMS Server간 Connection을 맺는 과정](#)

[ODBC\(Open Database Connectivity\), JDBC\(Java Database Connectivity\) 란?](#)

[데이터 분석 주제](#)

[SQLDEV 쿼리 설계](#)

[Excel - ODBC 연동 및 시각화](#)

[Oracle Instant Client 설치](#)

[윈도우 ODBC Driver Configuration 설정](#)

[Excel ODBC 연동](#)

[R - JDBC 연동 및 시각화](#)

[JDBC 설치](#)

[R을 활용한 시각화](#)

[Java - JDBC 연동 및 Chart FX이용 시각화](#)

[JavaFX 환경 설정 \(1\) - JavaFX 라이브러리 다운로드](#)

[JavaFX 환경 설정 \(2\) - JavaFX가 내장된 JDK 배포판 사용](#)

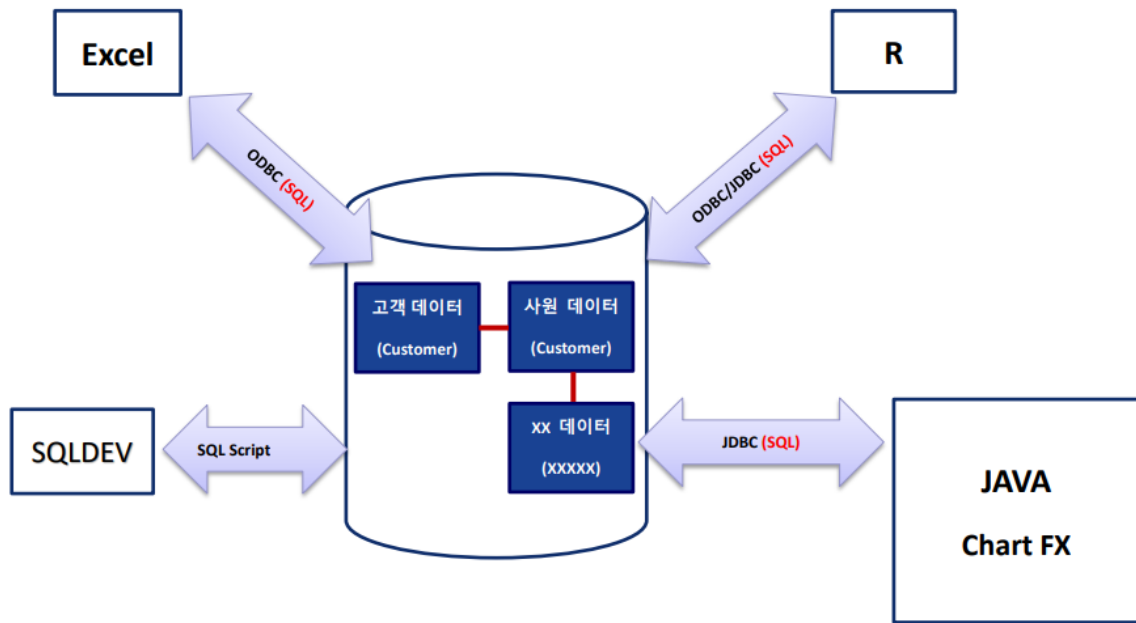
[JavaFX를 활용한 시각화](#)

[소스 코드](#)

SQL의 결과 데이터 Spool을 사용하여 ~.csv 로 출력

DBMS Connection & 시각화

개요



1. SQLDEV 에서 데이터 분석 Query 를 설계한다.
2. ODBC/JDBC 를 활용하여 Excel , R , JAVA 에서 DBMS 에 Connection 을 맺는다.
3. 해당 SQL 쿼리를 실행시키고 해당 데이터를 시각화한다.

과제를 통해 ODBC/JDBC 의 정의, 개요, 용도, 연결과정을 정리하고 핵심 내용을 이해한다.

요구사항



1. ODBC/ JDBC 정의, 개요, 용도, 연결과정 정리, 결과 화면캡처 포트폴리오 작성
2. 기술 질문시 답변할수 있도록 각자 이해한후 간결하고 핵심적인 내용으로 정리
3. 데이터 분석 결과 시각화 & 시사점(의견)
4. 3개 이상의 임의의 테이블 사용 , Join , SubQuery 사용 (option)
5. SQLDEV에서 SQL작성하여 기능 검증후 해당 SQL을 EXCEL,R,JAVA에서 재사용
R,JAVA 소스 첨부 , 소스내 간단한 주석
6. 해당 SQL의 결과 데이터 Spool을 사용하여 ~.csv 로 출력하여 제출
column header를 csv 파일에 포함 (노가다 입력 금지 !!!)

DBMS Connection 개념 및 ODBC/JDBC 용어 정리

Connection을 맺기 위한 주요 구성 요소

IP (Internet Protocol) 주소



DBMS 서버의 IP 주소는 **DBMS 서버를 식별하는데 사용**

Port 번호



DBMS 서버는 일반적으로 **특정 포트 번호에서 실행되며, 클라이언트는 이 포트 번호를 사용하여 해당 DBMS 서버에 연결한다.**

Listener



리스너: DBMS 서버내에서 **사용자의 신규접속 요청을 처리하는 프로세스** (default listening port :1521)

리스너는 **서버 프로세스를 생성(fork)** 하고 서버 프로세스가 사용하는 **메모리 영역(PGA : Program Global Area) 할당**

Protocol



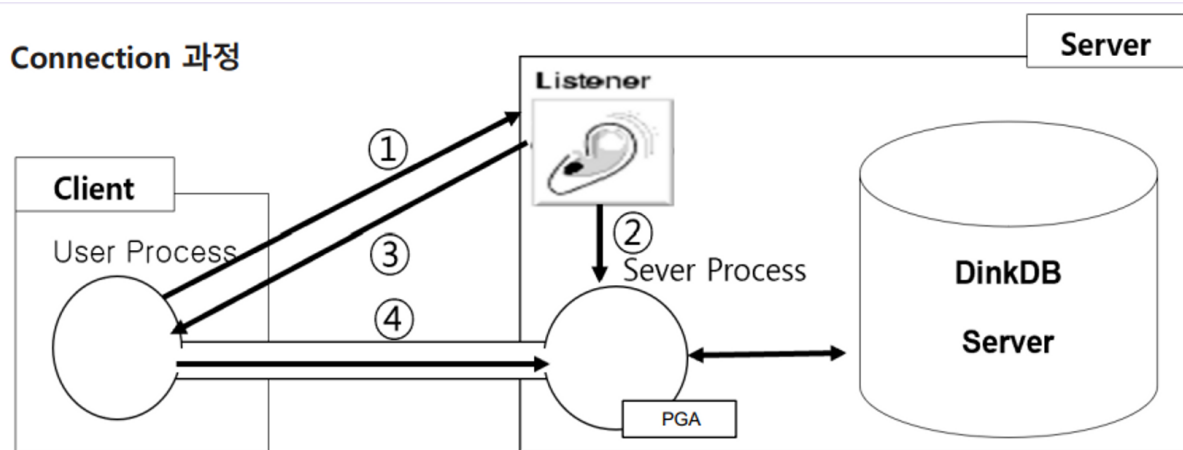
프로토콜은 DBMS 서버와 통신하는 데 사용됩니다. 대부분의 DBMS 서버는 **TCP/IP(Transmission Control Protocol/Internet Protocol) 프로토콜을 사용합니다.**

서비스 명



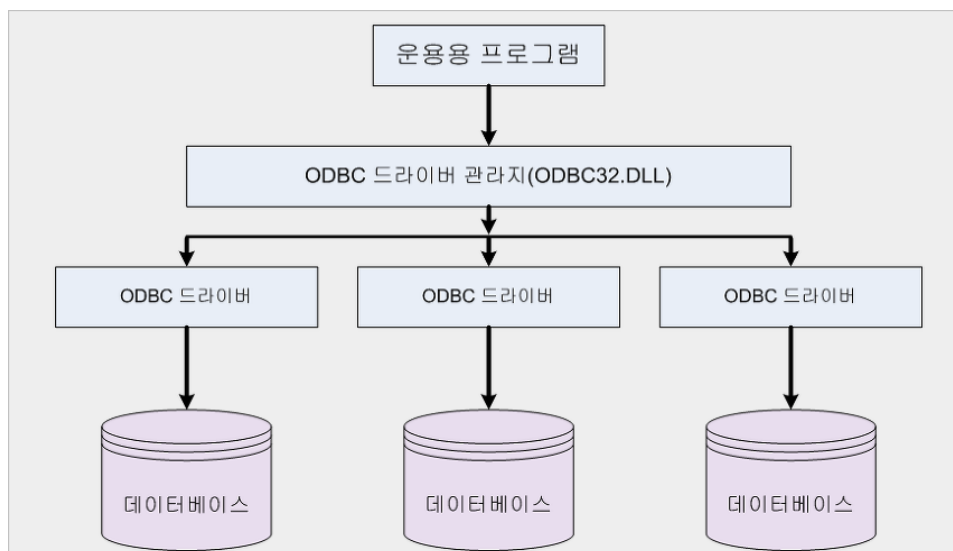
네트워크 상에서 **DBMS 인스턴스에 접속하기 위한 서비스 식별자로 사용**

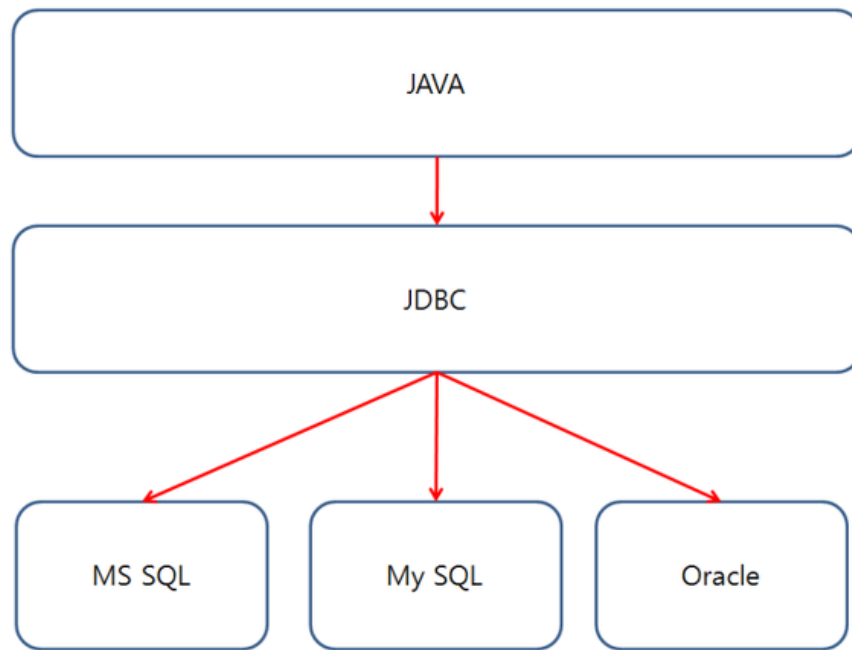
Client - DBMS Server간 Connection을 맺는 과정



1. 클라이언트 애플리케이션은 DBMS 서버의 IP 주소와 포트 번호를 사용하여 listener에 연결 요청
2. 리스너는 **서버 프로세스를 생성(fork)** 하고 서버 프로세스가 사용하는 **메모리 영역(PGA : Program Global Area) 할당**
3. 리스너가 새로 생성된 서버 프로세스의 주소를 사용자 프로세스에게 전달한다.
4. 사용자 프로세스가 서버 프로세스와 직접 Connection 형성(생성)

ODBC(Open Database Connectivity), JDBC(Java Database Connectivity) 란?





데이터베이스와 연결하여 데이터를 관리하고 조작하는 데 사용되는 프로그래밍 인터페이스

ODBC(Open Database Connectivity)

- ODBC는 데이터베이스에서 데이터를 가져오기 위한 표준 인터페이스
- ODBC 드라이버를 사용하여 데이터베이스에 연결하고 SQL(Structured Query Language)을 사용하여 데이터를 조회하거나 수정할 수 있다.
- ODBC는 C, C++, C#, VB 등의 프로그래밍 언어에서 사용되며 Excel에서 ODBC를 사용하여 데이터베이스에서 데이터를 가져올 수 있다.

JDBC(Java Database Connectivity)

- JDBC는 Oracle에서 개발된 자바 프로그래밍 언어를 위한 데이터베이스 연결 API이다.
- JDBC API는 Java SE의 일부로 Java 언어의 일부로 내장되어 있다.
- Java로 개발된 애플리케이션과 데이터베이스 간의 통신을 위한 공식적인 방법이다.

데이터 분석 주제



부서별 급여 지급 현황을 알아보려고 한다.

1. 부서별 급여의 총 액

→ 어떤 부서가 가장 많은 급여를 받고 있는지

2. 부서별 급여 지급 평균

→ 그 부서의 평균적인 급여 수준 확인

3. 부서별 급여 등급(SALGRADE)에 대한 평균

→ 부서 내 직원들의 SALGRADE의 평균 수준 확인

4. 부서 내 최대 급여의 액수

→ 최대 급여를 받는 사원의 급여 액수 확인

5. 최대 급여를 받는 사원의 수

→ 최대 급여를 받는 사원의 수 확인

위 정보를 통해 부서 별로 급여 지급 현황을 파악해보고자 한다.

SQLDEV 쿼리 설계

```
-- 주말 과제 2번 --
-- 사용할 데이터 테이블 --
SELECT * FROM DEPT;
SELECT * FROM EMP;
SELECT * FROM SALGRADE;

-- 부서별 평균급여와 급여등급의 평균, 최대 급여, 최대급여를 받는 사원 수 출력
SELECT d.deptno AS 부서번호,
       ROUND(AVG(e.sal)) AS 부서별_평균급여,
       SUM(e.sal) AS 부서별_급여총액,
       ROUND(AVG(s.grade)) AS 부서별_급여등급평균,
       MAX(e.sal) AS 최대급여,
       e_max.cnt AS 최대급여_사원수

-- 부서(DEPT), 사원(EMP), 급여등급(SALGRADE), 최대급여를 받는 사원 수 (sub query) Join
FROM dept d
INNER JOIN EMP e
ON d.deptno = e.deptno
INNER JOIN salgrade s
ON e.sal BETWEEN s.losal AND s.hisal -- 사원의 급여가 어느 등급에 포함되는지
INNER JOIN (
    SELECT deptno, count(deptno) AS cnt -- 최대 급여를 받는 사원 수
    FROM EMP
    WHERE (deptno, sal) IN (
```

```

SELECT deptno, MAX(sal) AS max_sal -- 그룹 별 최대 급여
FROM EMP
GROUP BY deptno
) group by deptno
) e_max
ON d.deptno = e_max.deptno -- Join 조건
GROUP BY d.deptno, e_max.cnt
ORDER BY d.DEPTNO;

```

질의 결과 x

SQL | 인출된 모든 행: 3(0.017초)

	부서번호	부서별_평균급여	부서별_급여총액	부서별_급여등급평균	최대급여	최대급여_사원수
1	10	2917	8750	4	5000	1
2	20	2175	10875	3	3000	2
3	30	1567	9400	3	2850	1



SQL Developer를 통해 SQL Script 쿼리를 설계했다.

* 사용한 테이블 : DEPT, EMP, SALGRADE

Excel - ODBC 연동 및 시각화







Oracle Instant Client 설치

Oracle Instant Client Downloads | Oracle 대한민국

Download links for Oracle Instant Client

<https://www.oracle.com/kr/database/technologies/instant-client/downloads.html>

Version 11.2.0.4.0

Name	Download	Description
Instant Client Package - Basic	 instantclient-basic-windows.x64-11.2.0.4.0.zip	All files required to run OCI, OCCI, and JDBC-OCI applications (54,956,947 bytes)
Instant Client Package - Basic Light	 instantclient-basclite-windows.x64-11.2.0.4.0.zip	Smaller version of the Basic, with only English error messages and Unicode, ASCII, and Western European character set support (10.2 only) (23,504,640 bytes)
Instant Client Package - JDBC Supplement	 instantclient-jdbc-windows.x64-11.2.0.4.0.zip	Additional support for XA, Internationalization, and RowSet operations under JDBC (1,565,996 bytes)
Instant Client Package - SQL*Plus	 instantclient-sqlplus-windows.x64-11.2.0.4.0.zip	Additional libraries and executable for running SQL*Plus with Instant Client (821,172 bytes)
Instant Client Package - SDK	 instantclient-sdk-windows.x64-11.2.0.4.0.zip	Additional header files and an example makefile for developing Oracle applications with Instant Client (1,446,625 bytes)
Instant Client Package - ODBC	 instantclient-odbc-windows.x64-11.2.0.4.0.zip	Additional libraries for enabling ODBC applications (1,358,385 bytes)



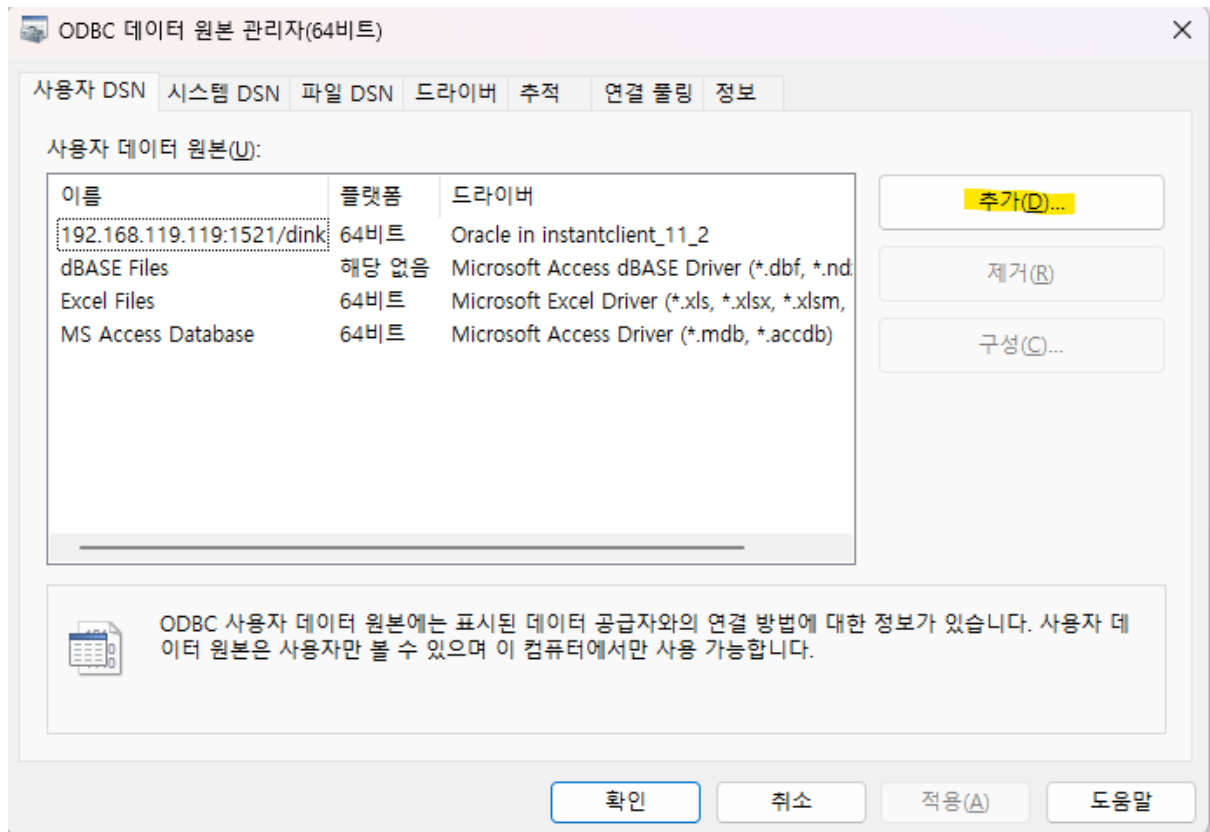
Excel 에서 ODBC 를 이용하여 Oracle DBMS에 접속하기 위한 클라이언트 프로그램 설치

```
rudej@gram MINGW64 /c/oracle/instantclient_11_2
$ ./odbc_install.exe
Oracle ODBC Driver with same name already exists.
```



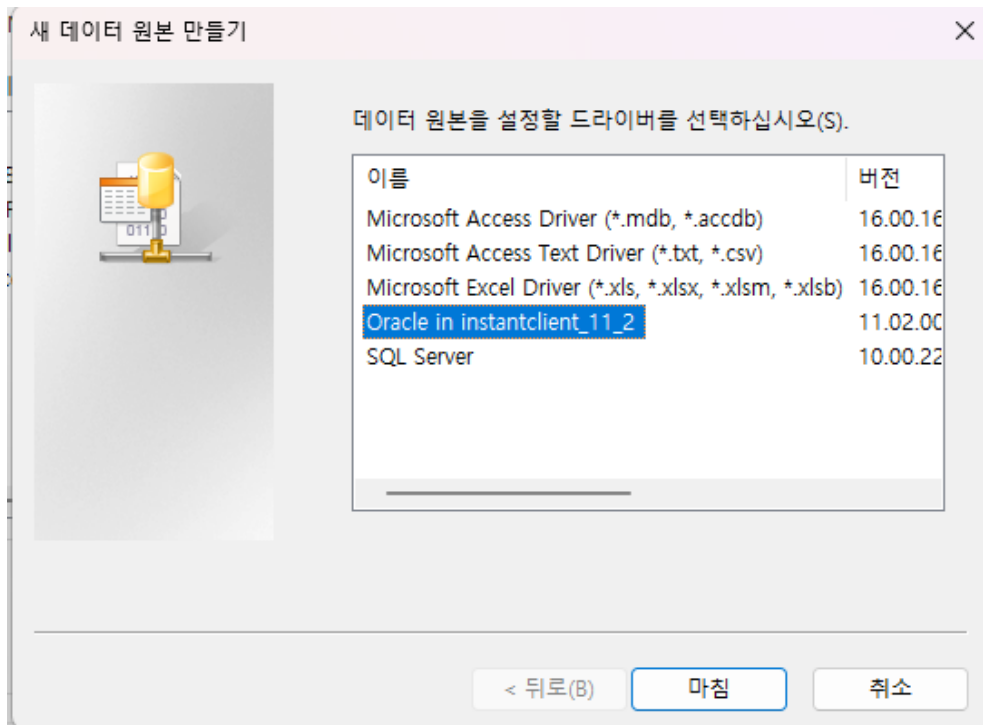
C:\Oracle\instantclient_11_2 에 압축을 풀고, odbc_install.exe 실행 파일을 실행 했다.

윈도우 ODBC Driver Configuration 설정

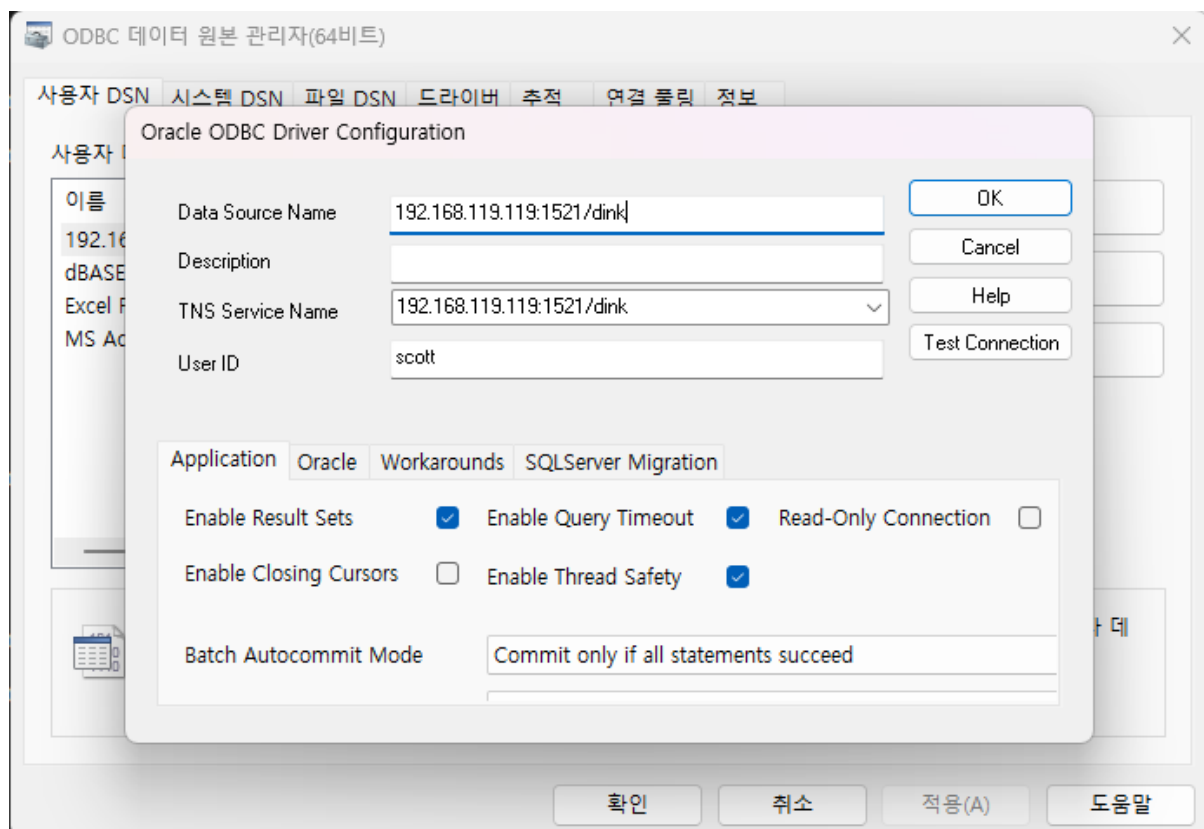


앞서 Oracle Instant Client 프로그램을 설치한 후,
Window 환경에서 ODBC 데이터 원본 관리자(64비트) 에서 사용자 DSN 에서 추가 를 누른다.

사용자 DSN은 사용자 계정에서 사용할 수 있는 데이터 원본을 만들 수 있다.



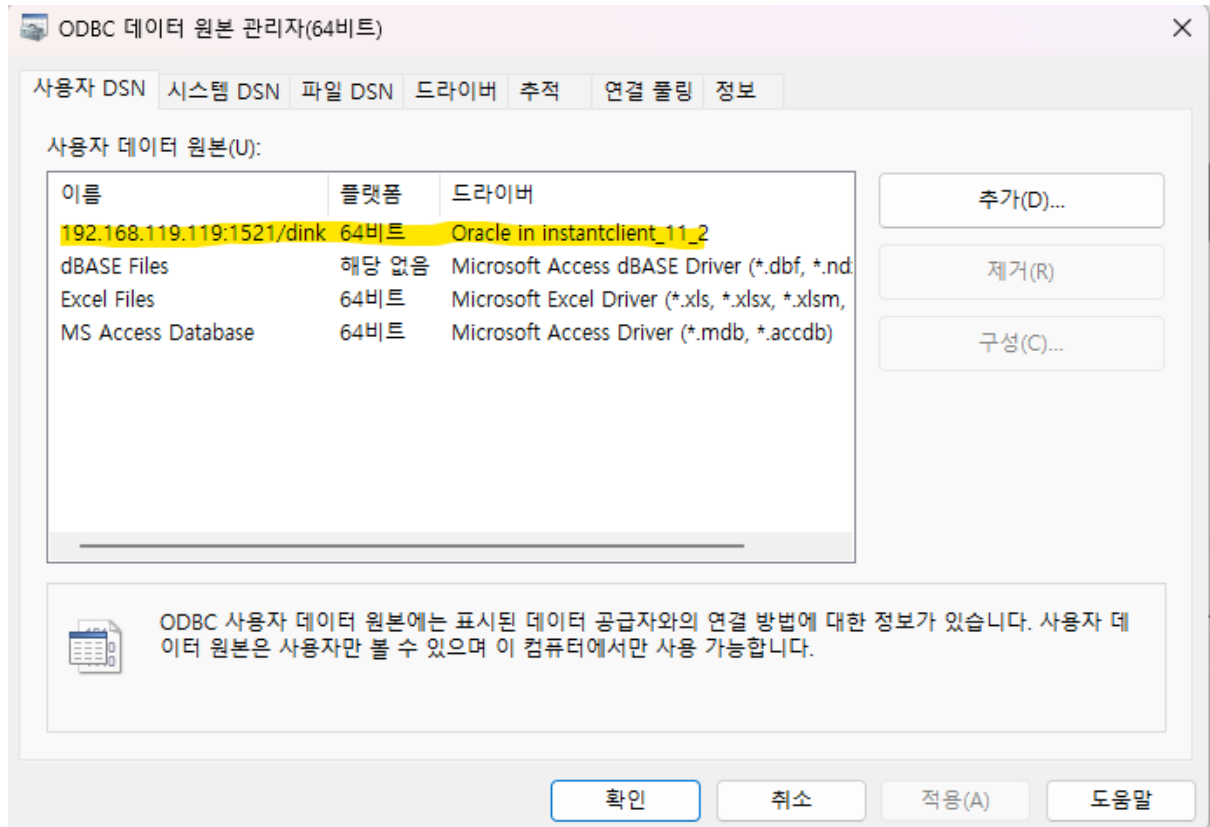
새 데이터 원본을 앞서 설치한 `Oracle in instantclient_11_2` 를 선택한다.





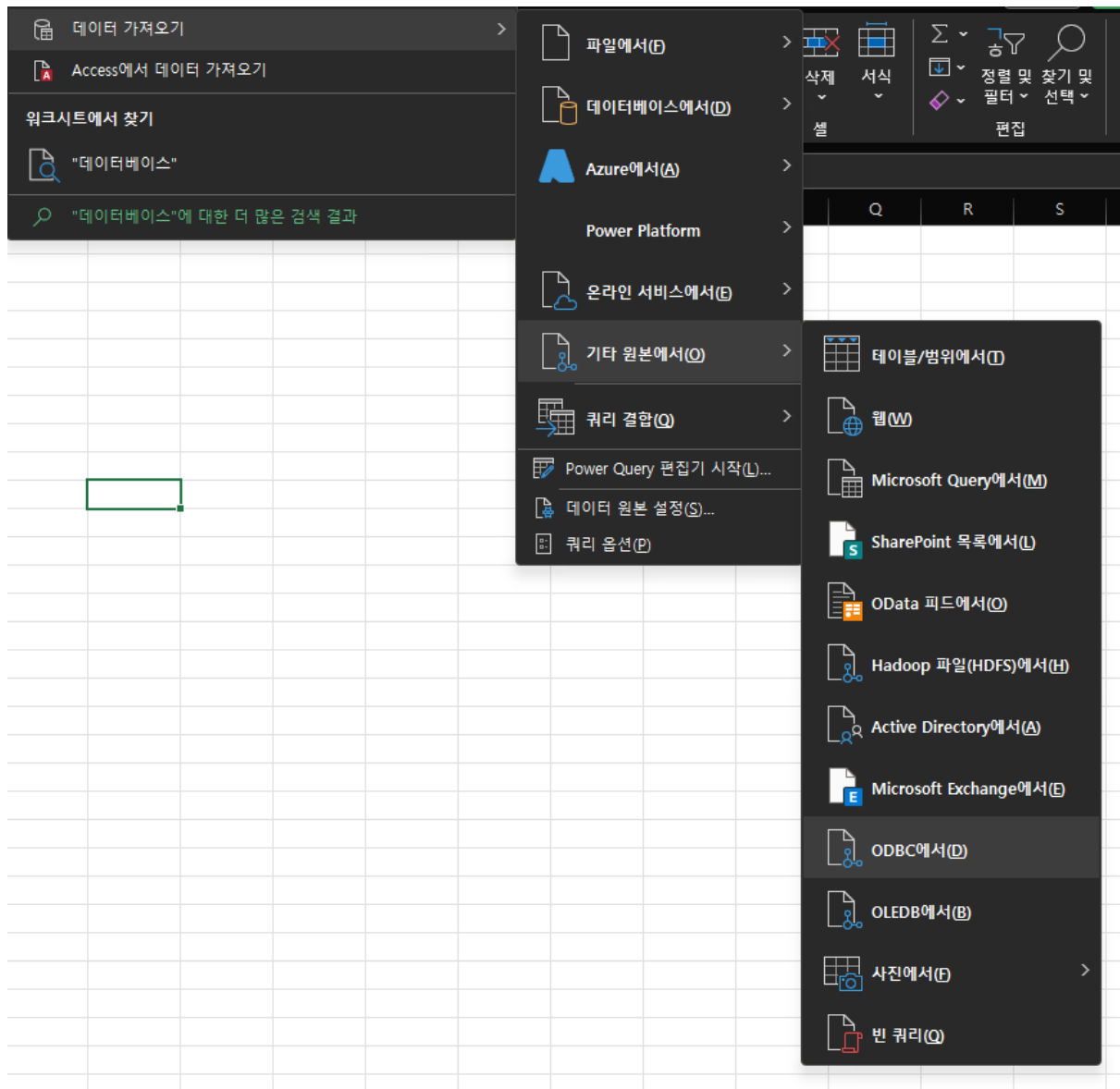
위 그림과 같이 (가상머신에 설치된 Oracle DBMS의 설정 정보)

Data Source Name, **TNS Service Name** (DBMS 서버 IP 번호:포트번호/서비스명), **UserID** 를 기입하고 OK를 누른다.



정상적으로 설정된 것을 볼 수 있다.

Excel ODBC 연동



Excel에서 데이터 가져오기 - 기타 원본에서 - ODBC에서 선택.

ODBC(dsn=192.168.119.119:1521/dink -- 부서별 평균급여와 급여등급의 평균, 최대 급여, 최대...

부서번호	부서별_평균급여	부서별_급여등급평균	최대급여	최대급여_사원수
10	2917	4	5000	1
20	2175	3	3000	2
30	1567	3	2850	1

로드 ▾

데이터 변환

취소



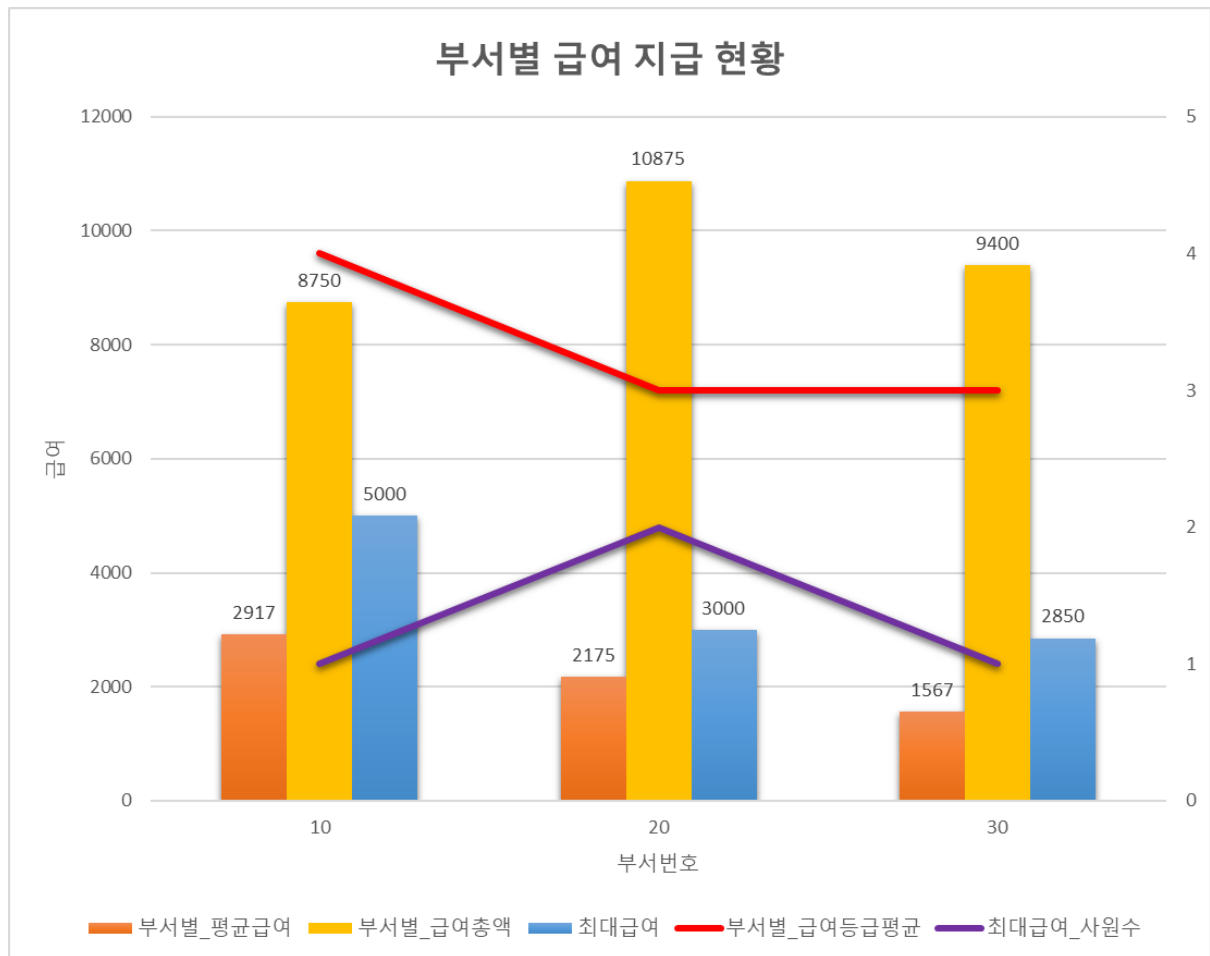
위와 같이, DBMS 서버에 정상적으로 접속이 되었다.

SQL 쿼리문의 결과 Table이 생성되었고 Excel에 로드할 수 있다.

	A	B	C	D	E	F
1	부서번호 ▾	부서별_평균급여 ▾	부서별_급여총액 ▾	부서별_급여등급평균 ▾	최대급여 ▾	최대급여_사원수 ▾
2	10	2917	8750	4	5000	1
3	20	2175	10875	3	3000	2
4	30	1567	9400	3	2850	1



정상적으로 SQL 쿼리문의 결과 Table이 로드되었다.



SQL 쿼리를 통해 얻은 Table을 차트로 시각화 했다.

20번 부서의 급여 총액이 가장 높았지만, 평균 급여 수준은 10번 부서가 가장 높음을 볼 수 있다.

또한 10번 부서에 가장 많은 급여를 받는 사원이 존재하는 것을 시각적으로 파악할 수 있다.

R - JDBC 연동 및 시각화

JDBC 설치

JDBC and UCP Downloads page | Oracle 대한민국

This page lists JDBC driver , UCP and other necessary jar files for various supported versions of Oracle Database.

<https://www.oracle.com/kr/database/technologies/appdev/jdbc-downloads.html>

Oracle Database 23c Free - Developer Release (23.2.0.0) JDBC Driver & UCP Downloads

Supports Oracle Database versions - 23c, 21c, 19c, 18c, and 12.2.

Name	Download	JDK Supported	Description
Oracle JDBC driver	ojdbc11.jar	Implements JDBC 4.3 spec and certified with JDK11 and JDK17	Oracle JDBC driver except classes for NLS support in Oracle Object and Collection types. (6,971,601 bytes) - (SHA1: a19cc23a15caea0914883e938907129354239e75)
Oracle JDBC Driver	ojdbc8.jar	Implements JDBC 4.2 spec and certified with JDK8 and JDK11	Oracle JDBC driver except classes for NLS support in Oracle Object and Collection types. (6,844,991 bytes) - (SHA1: 49acfb33ee776e43d2085e2fcc838778202a9128)
Universal Connection Pool (UCP) - ucp11.jar	ucp11.jar	Certified with JDK11 and JDK17	Universal Connection Pool (UCP) to be used with ojdbc11.jar (1,513,648 bytes) - (SHA1: 0638ee573f2b7bf873702fae53f291099ac7681)
Universal Connection Pool (UCP)	ucp.jar	Certified with JDK8 and JDK11	Universal Connection Pool (UCP) to be used with ojdbc8.jar (1,471,956 bytes) - (SHA1: 1b5300b30d54d091f7af7dc9070681c62fdad04c)
Zipped JDBC driver (ojdbc11.jar) and Companion Jars	ojdbc11-full.tar.gz	Certified with JDK11 and JDK17	This archive contains ojdbc11.jar , ucp11.jar , Reactive Streams Ingest (rsi.jar), companion jars ¹ , JDBC, UCP, RSI Javadoc, and their Readmes. Refer to README.txt in the zip for details. (26,562,560 bytes) - (SHA1: 41204e77573e9d36397016a18b0abc932580c44f)
Zipped JDBC driver (ojdbc8.jar) and Companion Jars	ojdbc8-full.tar.gz	Certified with JDK8 and JDK11	This archive contains ojdbc8.jar , ucp.jar , Reactive Streams Ingest (rsi.jar), companion jars ¹ , JDBC, UCP, RSI Javadoc, and their Readmes. Refer to README.txt in the zip for details. (25,907,200 bytes) - (SHA1: fed6c77364cb34cdd2e8236ec0a292a77d736e8d)

```
rudej@gram MINGW64 /d/SQLDEV
$ ls
0_virtual_machine/      dept_ext.csv
1_program/              instantclient-basic-windows.x64-11.2.0.4.0.zip
2_수업환경구성/        instantclient-odbc-windows.x64-11.2.0.4.0.zip
3_intro/                instantclient_11_2/
ODAC122010xcopy_x64/    ojdbc11.jar
ODP.NET_Managed_ODAC122cr1/ ojdbc8.jar
Readme.txt
```



JDBC 파일을 D://SQLDEV 에 다운 받았다.

R을 활용한 시각화

```
#JDBC 패키지 설치 및 부착
install.packages('rJava')
install.packages('RJDBC')
library(rJava)
library(RJDBC)

# 기초통계량 패키지
install.packages('Hmisc') #패키지 설치 한번만 하면 됨
library(Hmisc) #패키지 부착
install.packages('plyr')
library(plyr)

# 그래프 패키지
install.packages('ggplot2') #패키지 설치 한번만 하면 됨
library(ggplot2) #패키지 부착

# JDBC 드라이버에 driverClass, classPath 위치 대입
```



```

jdbcDriver <- JDBC(driverClass = "oracle.jdbc.OracleDriver",
                    classPath = "D://SQLDEV//ojdbc11.jar")

# 해당 Oracle DBMS 서버에 대한 정보를 통해 커넥션을 맺음
conn <- dbConnect(jdbcDriver,
                  "jdbc:oracle:thin:@192.168.119.119:1521/dink", "scott", "tiger")

# 커넥션을 맺은 서버에 SQL 쿼리문을 Get방식으로 요청 -> 결과 테이블을 데이터프레임 형식으로 얻을 수 있다.
test <- dbGetQuery(conn,
"SELECT d.deptno AS 부서번호,
ROUND(AVG(e.sal)) AS 부서별_평균급여,
SUM(e.sal) AS 부서별_급여총액,
ROUND(AVG(s.grade)) AS 부서별_급여등급평균,
MAX(e.sal) AS 최대급여,
e_max.cnt AS 최대급여_사원수
FROM dept d
INNER JOIN EMP e
ON d.deptno = e.deptno
INNER JOIN salgrade s
ON e.sal BETWEEN s.losal AND s.hisal
INNER JOIN (
  SELECT deptno, count(deptno) AS cnt
  FROM EMP
  WHERE (deptno, sal) IN (
    SELECT deptno, MAX(sal) AS max_sal
    FROM EMP
    GROUP BY deptno
  ) group by deptno
) e_max
ON d.deptno = e_max.deptno
GROUP BY d.deptno, e_max.cnt
ORDER BY d.DEPTNO")

# SQL 쿼리 결과 테이블 확인
test

# 부서별 평균급여, 급여총액, 최대급여에 대해 막대 그래프로 시각화 (ggplot, geom_rect 함수 이용)
# test dataframe의 '부서번호'를 x축, '부서별_평균급여'를 y축으로 설정
ggplot(test, aes(x = 부서번호, y = 부서별_평균급여)) +
  scale_x_continuous(breaks = c(10, 20, 30)) +
  geom_rect(aes(xmin = 부서번호 - 1, xmax = 부서번호 - 0.5, ymin = 0, ymax = 부서별_평균급여),
            fill = "lightblue", color = "black", alpha = 0.5) +

  geom_rect(aes(xmin = 부서번호 - 0.2, xmax = 부서번호 + 0.3, ymin = 0, ymax = 부서별_급여총액),
            fill = "pink", color = "black", alpha = 0.5) +

  geom_rect(aes(xmin = 부서번호 + 0.6, xmax = 부서번호 + 1.1, ymin = 0, ymax = 최대급여),
            fill = "green", color = "black", alpha = 0.5) +

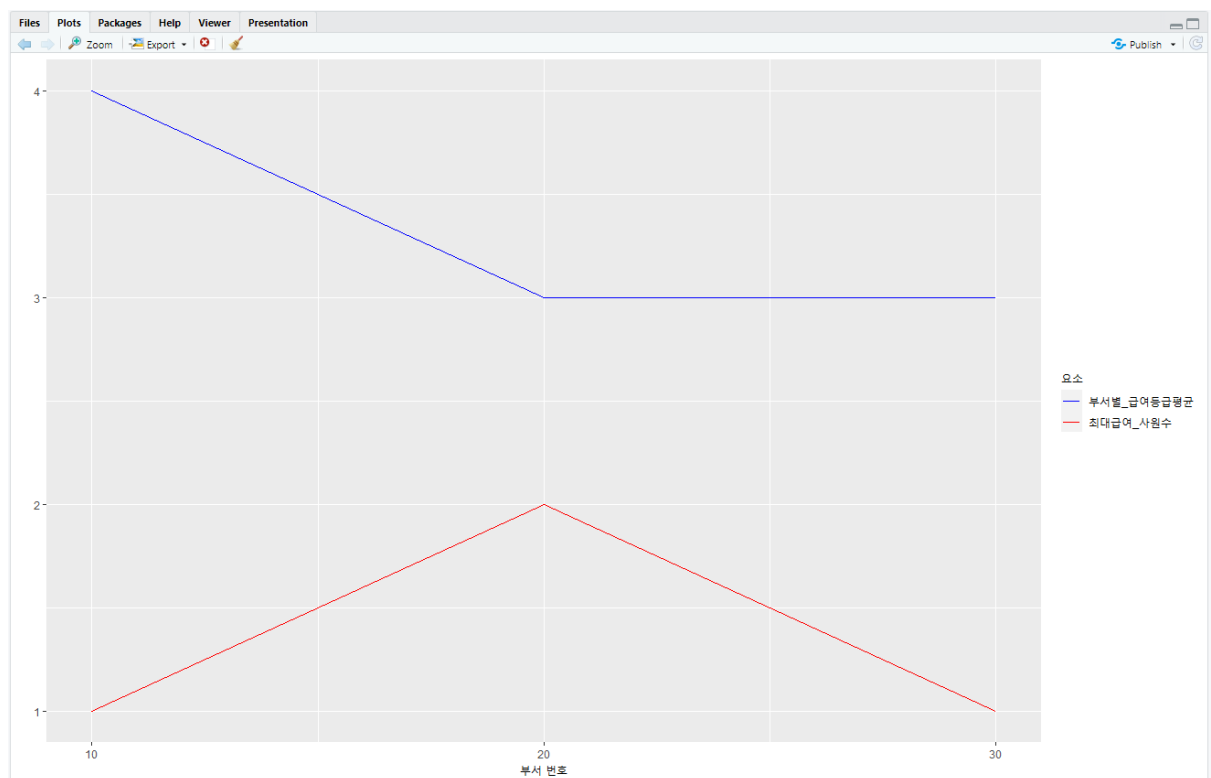
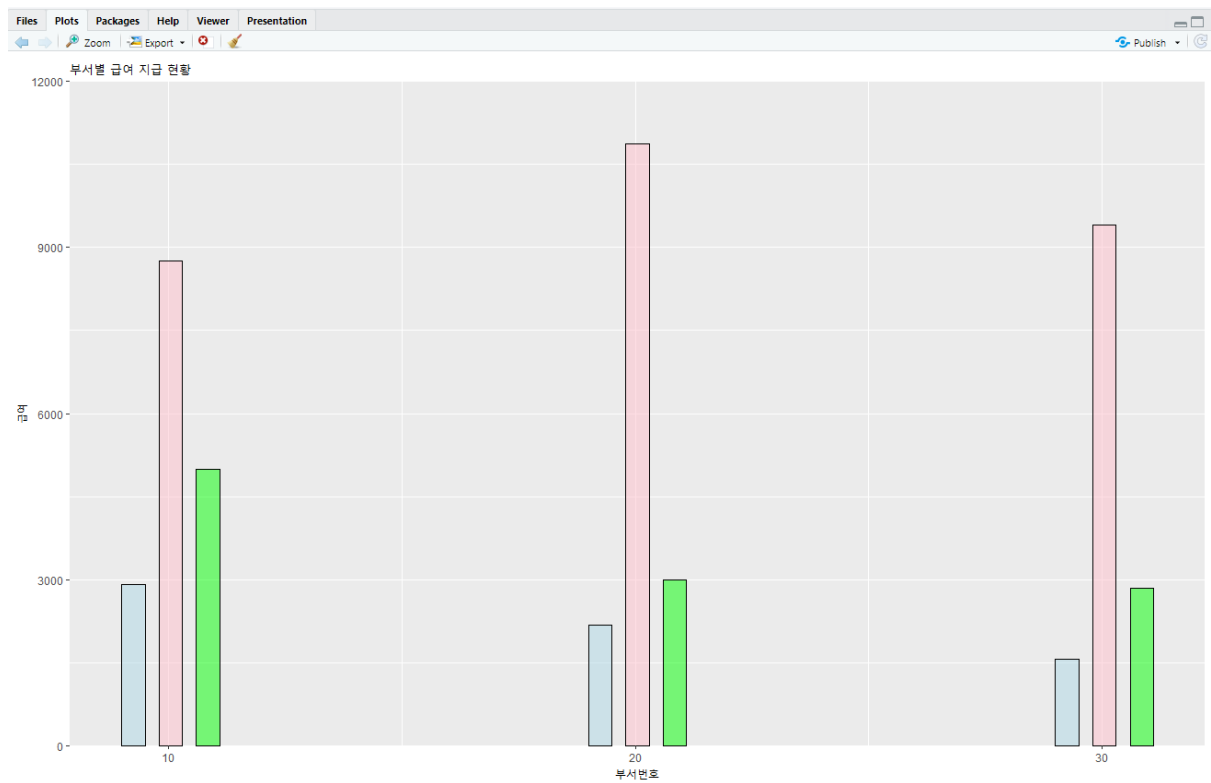
  # y축의 범위를 0부터 12000까지 지정
  # expand는 그래프와 축 사이에 여백을 주는 값으로, x축과 y축 각각 0으로 지정하여 여백 제거
  scale_y_continuous(limits = c(0, 12000), expand = c(0, 0)) +

  # 제목 및 x, y축에 라벨 부착
  labs(title = "부서별 급여 지급 현황", x = "부서번호", y = "급여")

# 부서별 급여등급평균, 최대급여_사원수에 대해 꺾은선 그래프로 시각화 (ggplot, geom_line 함수 이용)
ggplot(test, aes(x = 부서번호)) +
  scale_x_continuous(breaks = c(10, 20, 30)) +
  geom_line(aes(y = 부서별_급여등급평균, colour = "부서별_급여등급평균")) +
  geom_line(aes(y = 최대급여_사원수, colour = "최대급여_사원수")) +
  xlab("부서 번호") +

```

```
ylab("") +
  scale_colour_manual(name = "요소", values = c("부서별_급여등급평균" = "blue", "최대급여_사원수" = "red"))
```





R에서 JDBC를 이용, SQL 쿼리를 통해 얻은 Table (Dataframe)을 차트로 시각화 했다.

`ggplot`, `geom_rect` 함수 이용하여, 부서별 평균급여, 급여총액, 최대급여에 대해 막대 그래프로 시각화 했다.

`ggplot`, `geom_line` 함수를 이용하여, 부서별 급여등급평균, 최대급여_사원수에 대해 꺾은선 그래프로 시각화 했다.

마찬가지로,
20번 부서의 급여 총액이 가장 높았지만, 평균 급여 수준은 10번 부서가 가장 높음을 볼 수 있으며, 10번 부서에 가장 많은 급여를 받는 사원이 존재하는 것을 시각적으로 파악할 수 있다.

Java - JDBC 연동 및 Chart FX이용 시각화

JavaFX 환경 설정 (1) - JavaFX 라이브러리 다운로드

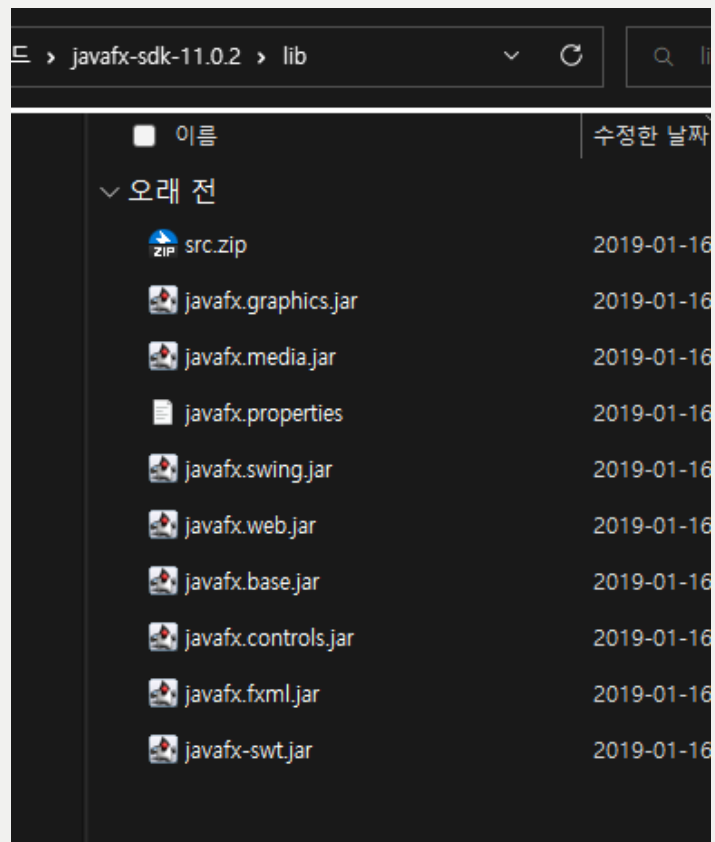
Download JavaFX 11

 https://www.reddit.com/r/JavaFX/comments/r729tw/download_javafx_11/

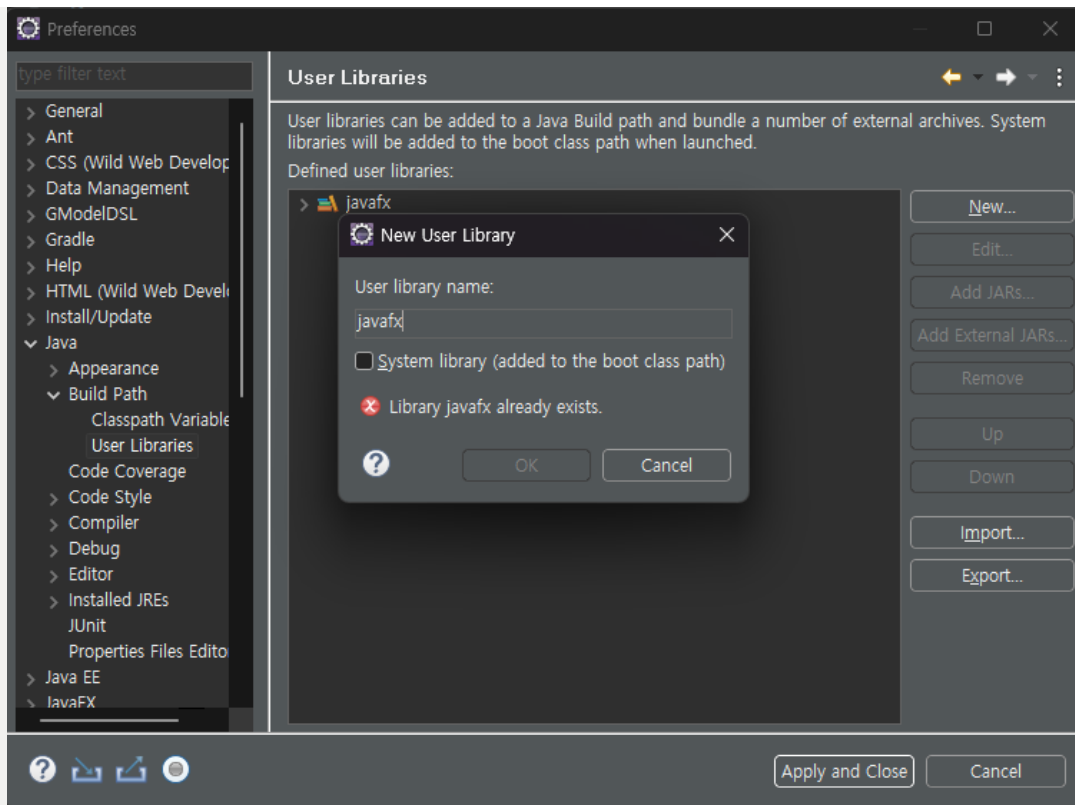


JavaFX 11 라이브러리 다운로드

C:\Users\Downloads\javafx-sdk-11.0.2\lib

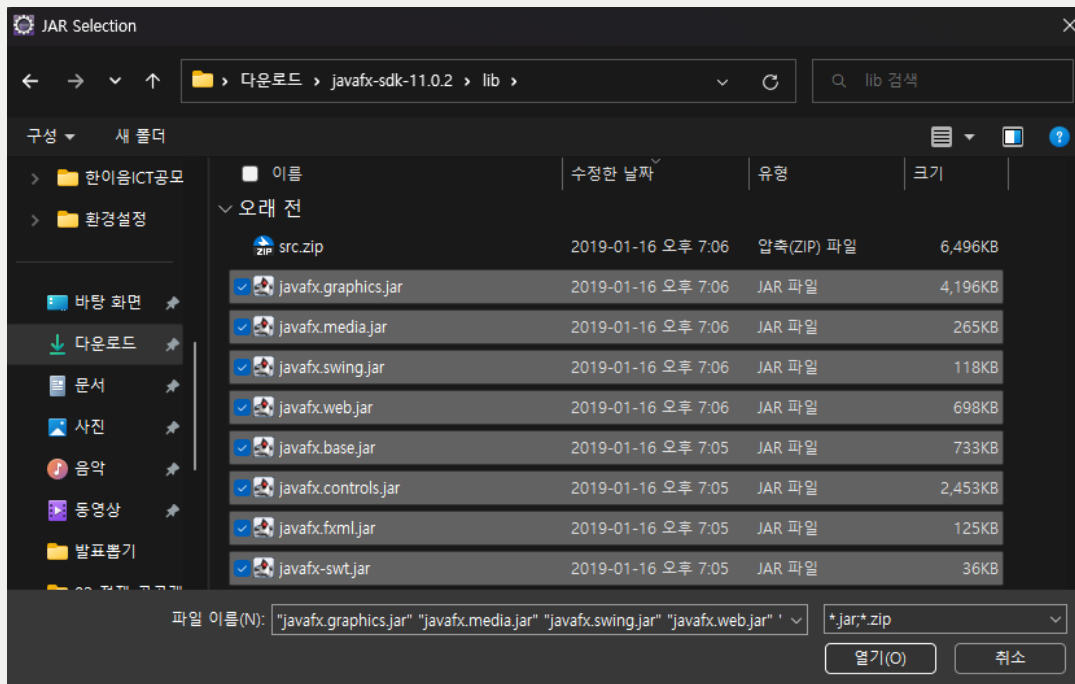


우선, 라이브러리 다운로드 후 압축을 풀고 **javaFX 라이브러리 .jar** 파일을 얻었다.

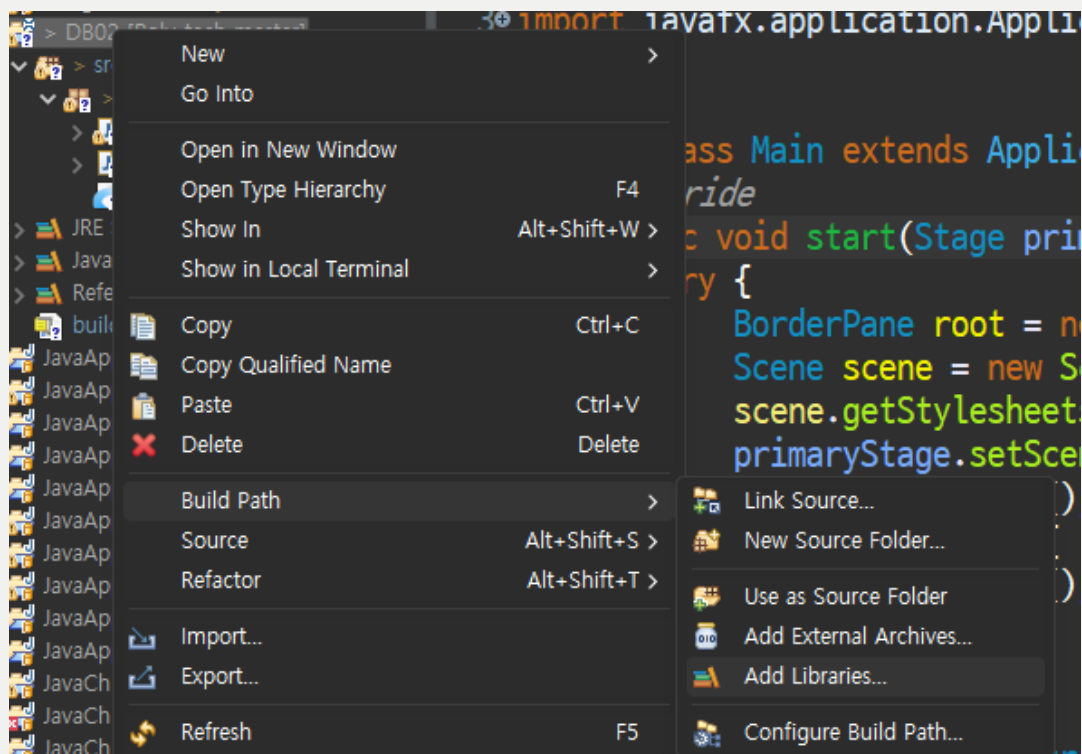


javafx 사용자 라이브러리 정의

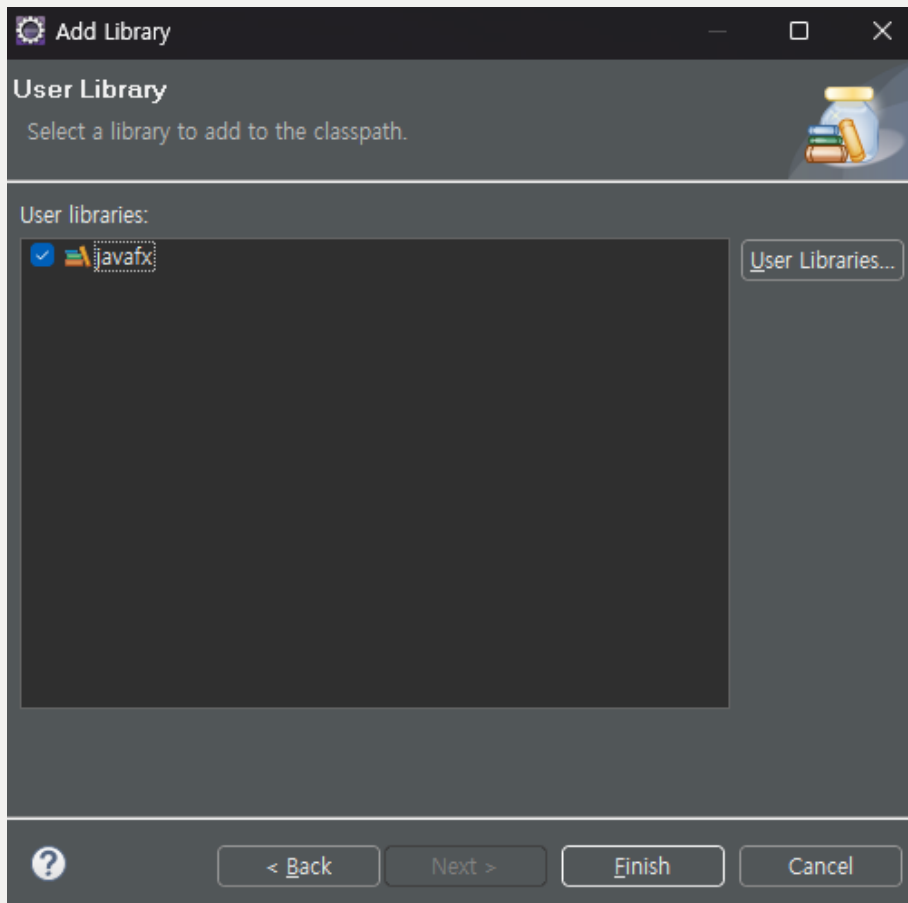
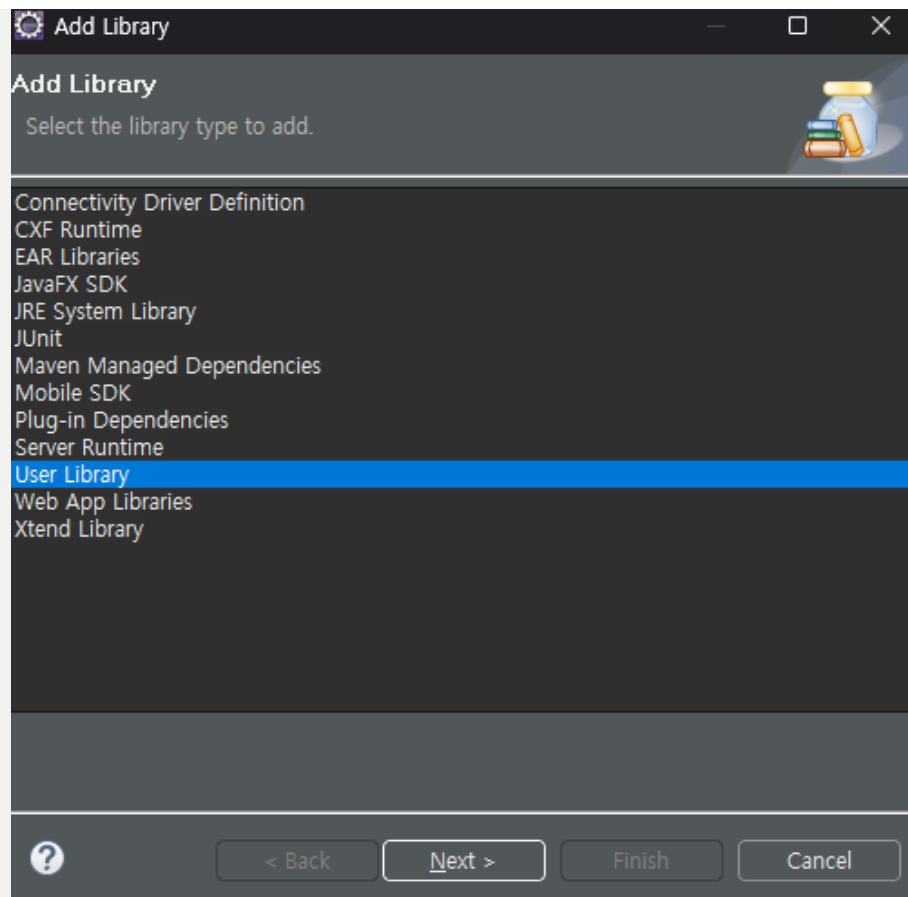
Windows - Preference - Java - Build Path - User Libraries - New...



ADD External JARS - 앞서, 다운 받은 javaFX 라이브러리 jar 파일 선택 이후 Apply and Close




JavaFX 프로젝트 우클릭 - Build Path - Add Libraries...



해당 라이브러리 선택 후 완료

JavaFX 환경 설정 (2) - JavaFX가 내장된 JDK 배포판 사용

Azul Downloads

 <https://www.azul.com/downloads/?version=java-11-lts&os=windows&architecture=x86-64-bit&package=jdk-fx>

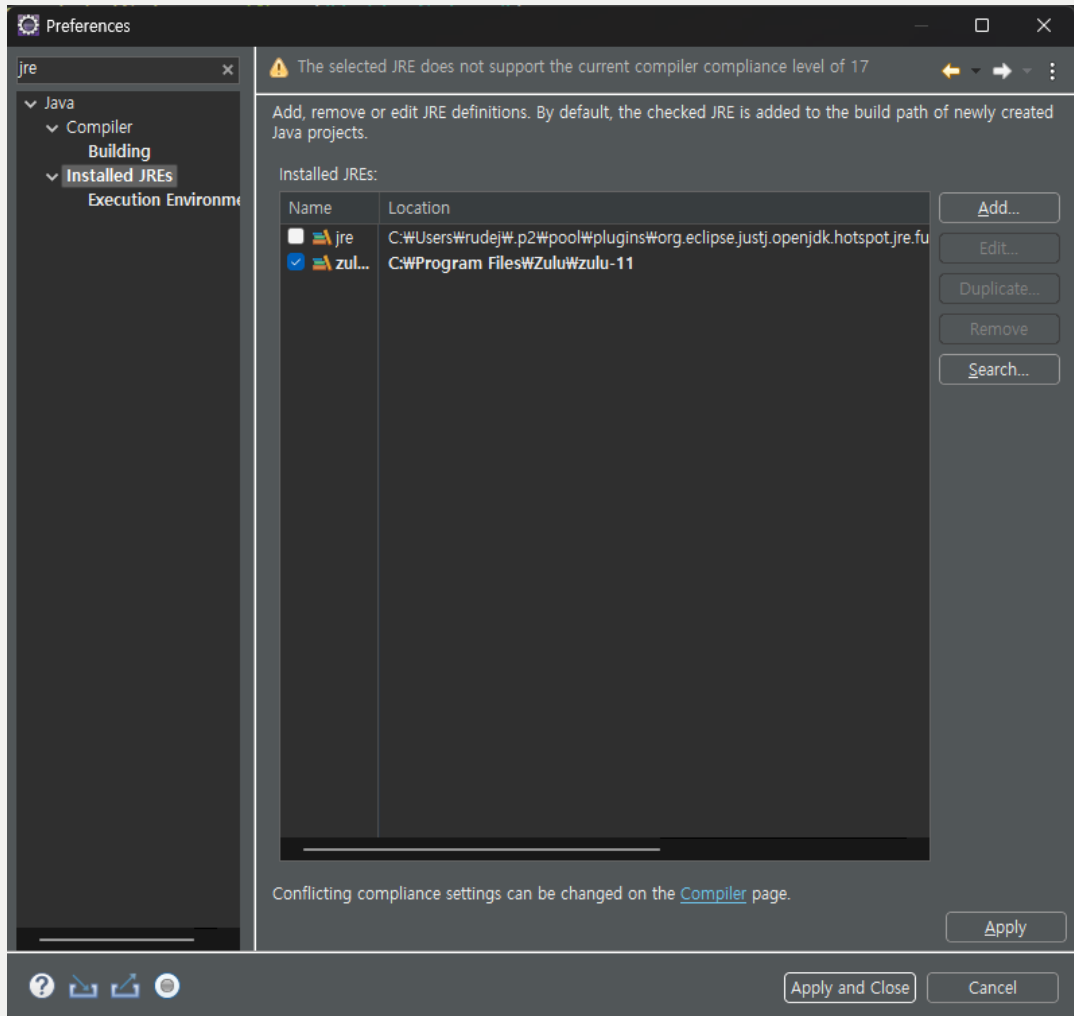
azul

Download the Azul Zulu Builds of OpenJDK and Azul Platform Prime

Download Free

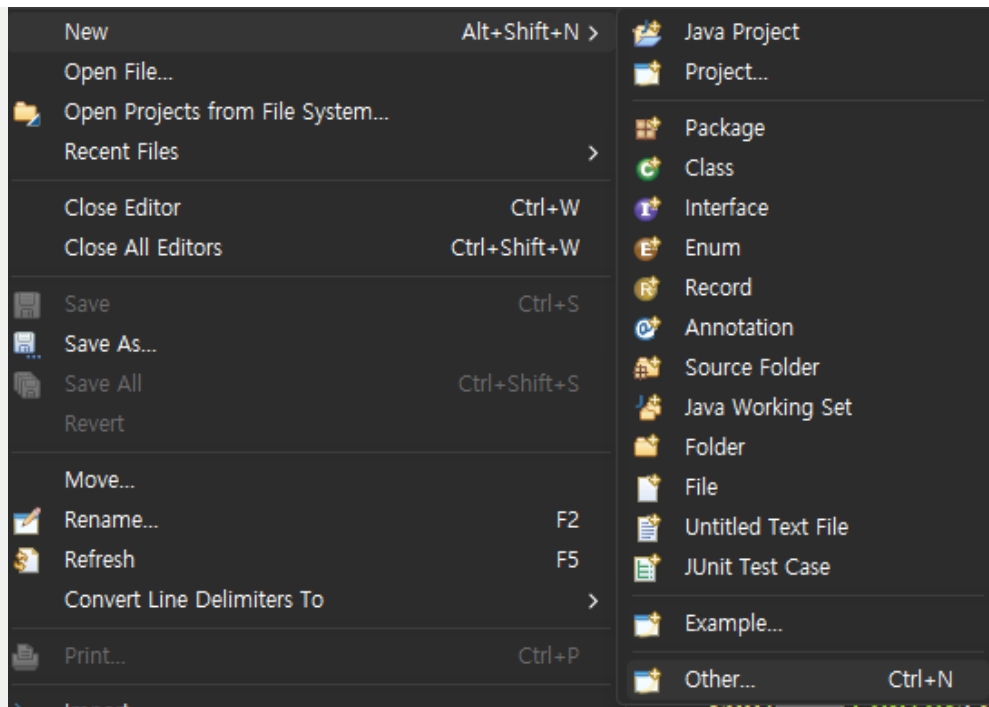


JavaFX를 간편하게 사용하기 위해서, **JavaFX 라이브러리가 내장된 JDK 배포판- zulu JDK 11.msi 설치**

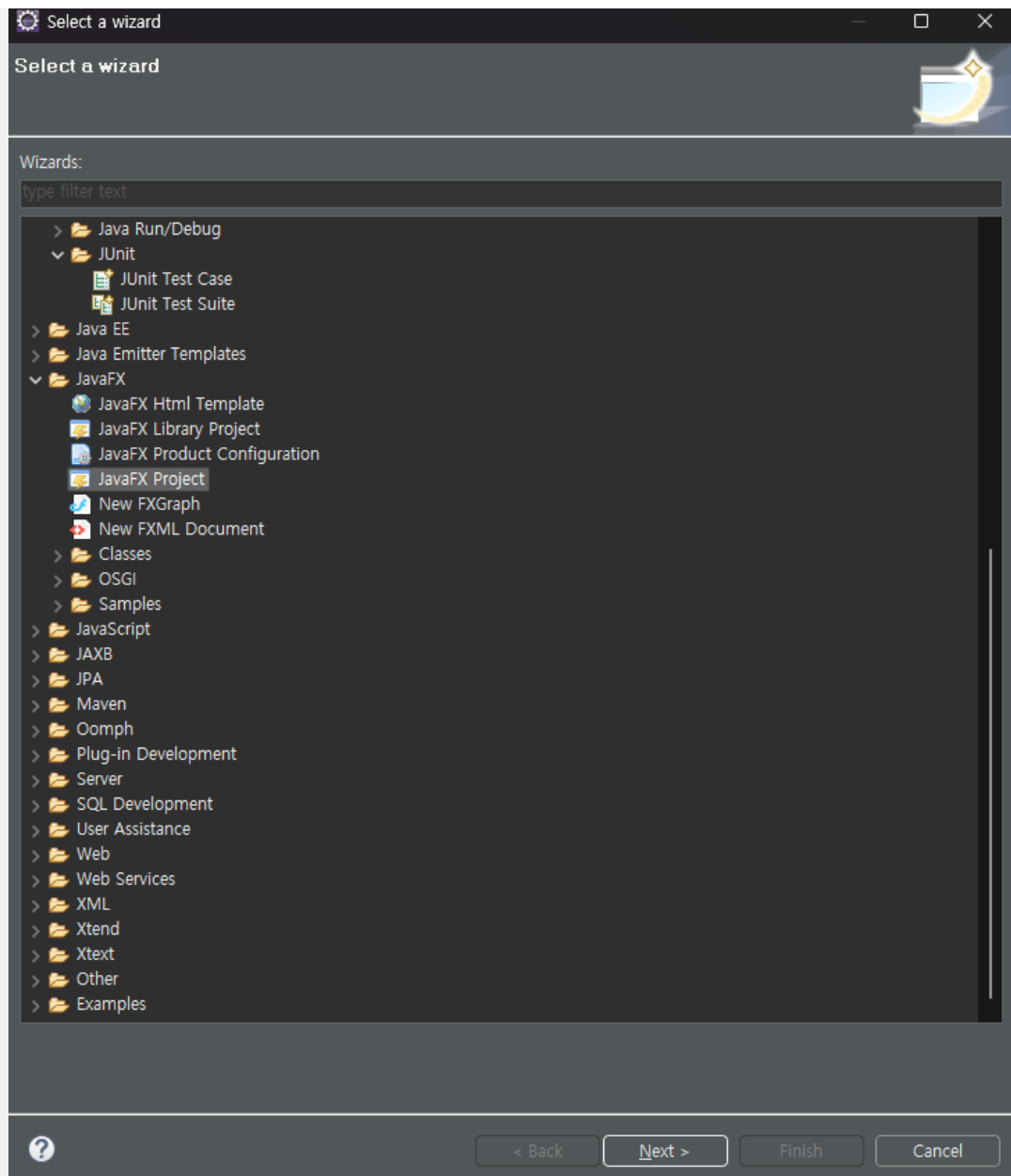


Zulu JDK 설치 후 Eclipse - Window - Preferences - Installed JREs 에 zulu JDK Add

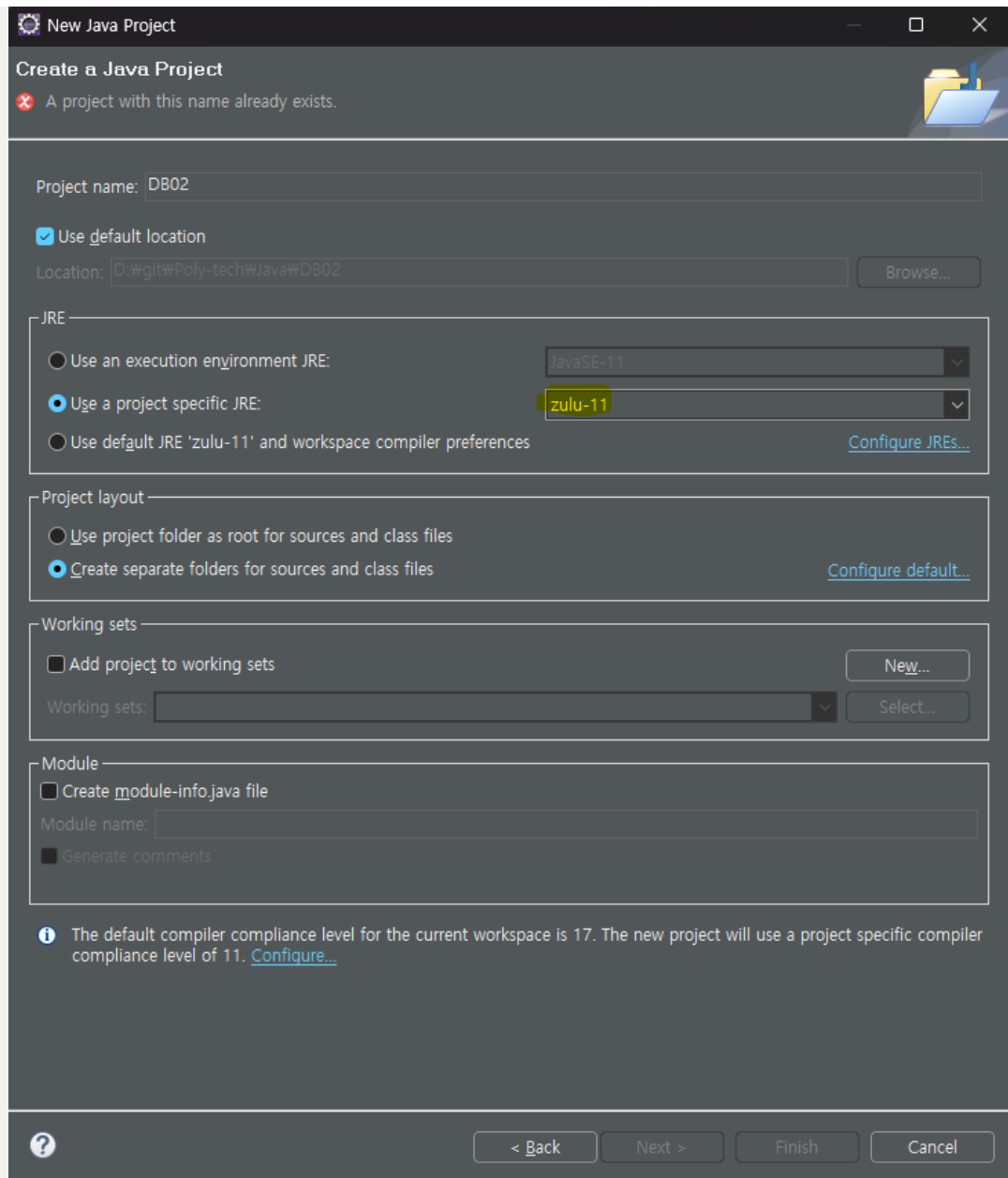
zulu-11 체크 후 Apply and Close



File - New - Other...



JavaFX Project 생성



프로젝트 생성시, JRE를 **zulu-11**로 선택 완료

JavaFX를 활용한 시각화

소스 코드

```
package application;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
```

```

import java.sql.SQLException;
import java.sql.Statement;
import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.Scene;
import javafx.scene.chart.BarChart;
import javafx.scene.chart.CategoryAxis;
import javafx.scene.chart.NumberAxis;
import javafx.scene.chart.XYChart;
import javafx.scene.layout.BorderPane;
import javafx.stage.Stage;

public class JDBC extends Application {

    // 그래프의 데이터를 저장하기 위한 ObservableList
    private ObservableList<XYChart.Data<String, Number>> dataAvgSalary =
        FXCollections.observableArrayList();
    private ObservableList<XYChart.Data<String, Number>> dataTotalSalary =
        FXCollections.observableArrayList();
    private ObservableList<XYChart.Data<String, Number>> dataMaxSalary =
        FXCollections.observableArrayList();

    @Override
    public void start(Stage primaryStage) throws Exception {
        // UI를 구성하기 위한 루트 노드 생성
        BorderPane root = new BorderPane();

        // BarChart 생성
        CategoryAxis xAxis = new CategoryAxis(); // x축
        NumberAxis yAxis = new NumberAxis(); // y축
        BarChart<String, Number> barChart = new BarChart<>(xAxis, yAxis); // BarChart 생성

        barChart.setTitle("부서별 급여 지급 현황"); // 그래프 제목 설정

        // 데이터베이스 연결 정보 설정
        String url = "jdbc:oracle:thin:@192.168.119.119:1521/dink";
        String user = "scott";
        String passwd = "tiger";

        // 질의문 작성
        String query = "SELECT d.deptno AS 부서번호,\r\n"
            + "        ROUND(AVG(e.sal)) AS 부서별_평균급여,\r\n"
            + "        SUM(e.sal) AS 부서별_급여총액,\r\n"
            + "        ROUND(AVG(s.grade)) AS 부서별_급여등급평균,\r\n"
            + "        MAX(e.sal) AS 최대급여,\r\n"
            + "        e_max.cnt AS 최대급여_사원수\r\n"
            + "        \r\n"
            + "FROM dept d\r\n"
            + "INNER JOIN EMP e\r\n"
            + "ON d.deptno = e.deptno\r\n"
            + "INNER JOIN salgrade s\r\n"
            + "ON e.sal BETWEEN s.losal AND s.hisal\r\n"
            + "INNER JOIN (\r\n"
            + "    SELECT deptno, count(deptno) AS cnt\r\n"
            + "    FROM EMP\r\n"
            + "    WHERE (deptno, sal) IN (\r\n"
            + "        SELECT deptno, MAX(sal) AS max_sal\r\n"
            + "        FROM EMP\r\n"
            + "        GROUP BY deptno\r\n"
            + "    ) group by deptno\r\n"
            + ") e_max\r\n"

```

```

        + "ON d.deptno = e_max.deptno\r\n"
        + "GROUP BY d.deptno, e_max.cnt\r\n"
        + "ORDER BY d.DEPTNO";

// 데이터베이스에 연결하여 질의문 실행
try (Connection con = DriverManager.getConnection(url, user, passwd);
    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery(query)) {

    // 결과셋에서 데이터를 읽어와서 그래프에 데이터 추가
    while (rs.next()) {
        String deptNo = rs.getString("부서번호"); // 부서명
        String grade = rs.getString("부서별_급여등급평균"); // 평균 등급
        Double avgSal = rs.getDouble("부서별_평균급여"); // 평균 급여
        Double maxSalNumber = rs.getDouble("부서별_급여총액"); // 급여 총액
        Double maxSal = rs.getDouble("최대급여"); // 최대 급여

        // 데이터 추가
        dataAvgSalary.add(new XYChart.Data<>(deptNo + "-" + grade, avgSal));
        dataTotalSalary.add(new XYChart.Data<>(deptNo + "-" + grade, maxSalNumber));
        dataMaxSalary.add(new XYChart.Data<>(deptNo + "-" + grade, maxSal));
    }
} catch (SQLException e) {
    e.printStackTrace(); // 예외 처리
}

// 데이터를 그래프에 추가
XYChart.Series<String, Number> seriesAvgSalary = new XYChart.Series<>();
XYChart.Series<String, Number> seriesTotalSalary = new XYChart.Series<>();
XYChart.Series<String, Number> seriesMaxSalary = new XYChart.Series<>();

seriesAvgSalary.setName("Average Salary");
seriesTotalSalary.setName("Total Salary");
seriesMaxSalary.setName("Maximum Salary");

seriesAvgSalary.setData(dataAvgSalary);
seriesTotalSalary.setData(dataTotalSalary);
seriesMaxSalary.setData(dataMaxSalary);

barChart.getData().addAll(seriesAvgSalary, seriesTotalSalary, seriesMaxSalary);

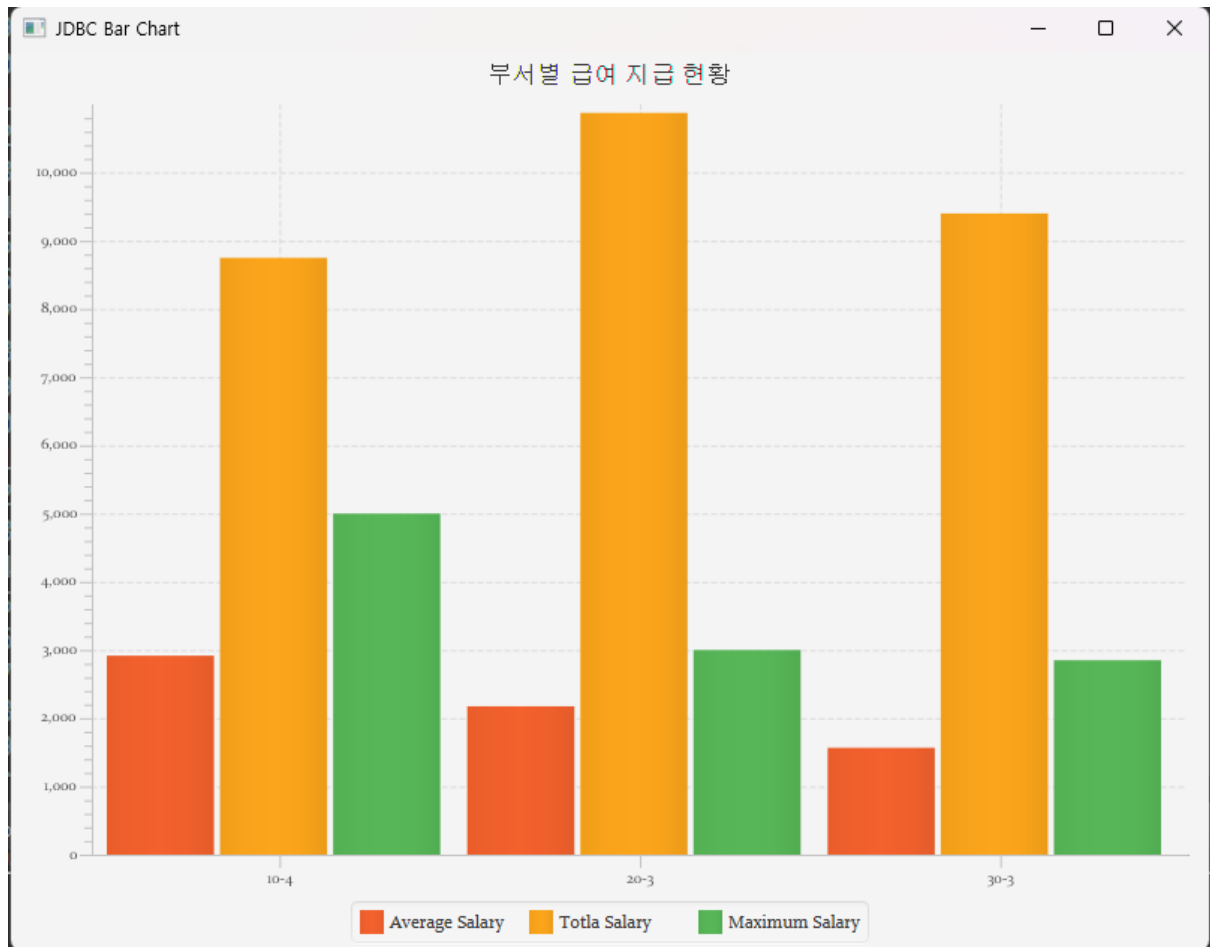
// 그래프를 루트 노드에 추가
root.setCenter(barChart);

// Scene 생성
Scene scene = new Scene(root, 800, 600);

// Stage에 Scene 추가 후 보여주기
primaryStage.setScene(scene);
primaryStage.setTitle("JDBC Bar Chart");
primaryStage.show();
}

public static void main(String[] args) {
    launch(args);
}
}

```



Java에서 JDBC를 이용, SQL 쿼리를 통해 얻은 테이블 객체를 차트로 시각화 했다.

시각화 할 데이터를 `observableList`에 저장하고,
`BarChart<>` 제네릭 클래스를 활용해 막대 차트로 시각화 했다.

마찬가지로,
 20번 부서의 급여 총액이 가장 높았지만, 평균 급여 수준은 10번 부서가 가장 높음을 볼 수 있으며, 10번 부서에 가장 많은 급여를 받는 사원이 존재하는 것을 시각적으로 파악할 수 있다.

SQL의 결과 데이터 Spool을 사용하여 ~.csv 로 출력

```
set heading off
set feedback off
set echo off
set term off
set timing off

SPPOOL assignment02.csv
```

```

SELECT d.deptno AS 부서번호,
       ROUND(AVG(e.sal)) AS 부서별_평균급여,
       SUM(e.sal) AS 부서별_급여총액,
       ROUND(AVG(s.grade)) AS 부서별_급여등급평균,
       MAX(e.sal) AS 최대급여,
       e_max.cnt AS 최대급여_사원수
FROM dept d
INNER JOIN EMP e
ON d.deptno = e.deptno
INNER JOIN salgrade s
ON e.sal BETWEEN s.losal AND s.hisal
INNER JOIN (
    SELECT deptno, count(deptno) AS cnt
    FROM EMP
    WHERE (deptno, sal) IN (
        SELECT deptno, MAX(sal) AS max_sal
        FROM EMP
        GROUP BY deptno
    ) group by deptno
) e_max
ON d.deptno = e_max.deptno
GROUP BY d.deptno, e_max.cnt
ORDER BY d.DEPTNO;

SPOOL OFF

```

	1	2	3	4	5	6	7
1	부서번호	부서별_평균	부서별_급여	부서별_급여	최대급여	최대급여_사원수	
2	10	2917	8750	4	5000	1	
3	20	2175	10875	3	3000	2	
4	30	1567	9400	3	2850	1	
5							



이번 과제를 통해 3개의 테이블을 Join 및 여러 서브 쿼리를 미리 경험했다. 또한, 해당 쿼리 결과를 통해 **부서별 급여 지급 현황**이라는 의미 있는 분석을 도출해보기도 했다.

그리고 SQL Developer에서의 쿼리 작성 뿐만 아니라 JDBC/ODBC를 활용하여 Excel, R, Java에서의 각 환경 구축 및 Embedded SQL 그리고 데이터 시각화까지 실습해보는 경험이 되었다.

이번 경험으로 단순 SQL 쿼리 작성을 넘어 SQL Client 프로그램을 통한 SQL 기능 검증 후, Java와 같은 프로그래밍 언어에서 해당 SQL을 재사용하는 개발 프로세스 과정을 익히게 되는 계기가 되었다.