



데이터 수집 및 가공 실습

빅데이터 오케스트레이션 및 시각화 실습 - 4일차

학습 내용

1. 데이터 시각화 개요
2. Mysql
3. Node-Red와 DB 연결 실습
4. Web 시각화
→ 테이블 및 차트 시각화
5. DDP 플레이어 실습



_Open API 불러오기 실습

실습 문제 : 다음 URL에서 데이터를 읽어오는 Node-red 노드를 구성하고 AJAX로 불러오기

→ <http://make-random.com/MakeRandom/api/userInfo.get>

- 1) 노드 레드 API를 불러오는 웹페이지 작성 (AJAX 호출)
- 2) 'http://make-random.com/MakeRandom/api/userInfo.get' 읽어 오는 노드 작성

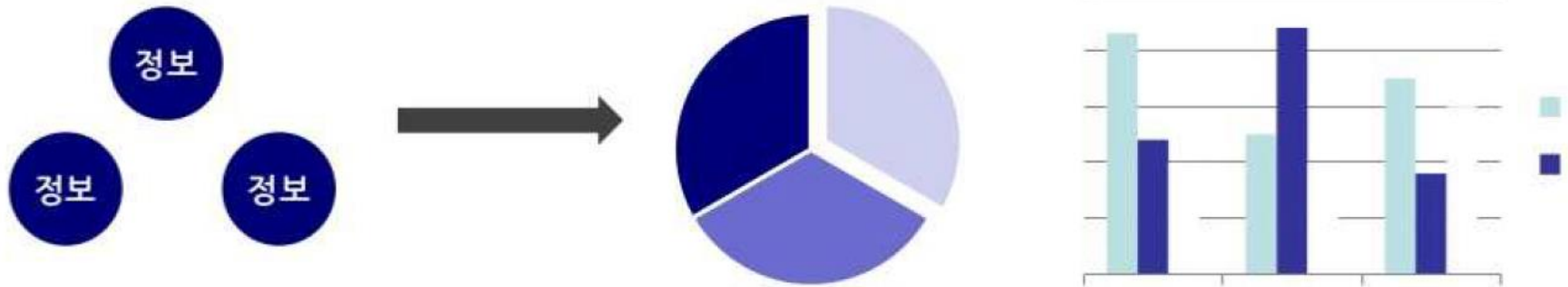


데이터 시각화 개요

_데이터 시각화

■ 데이터 시각화란

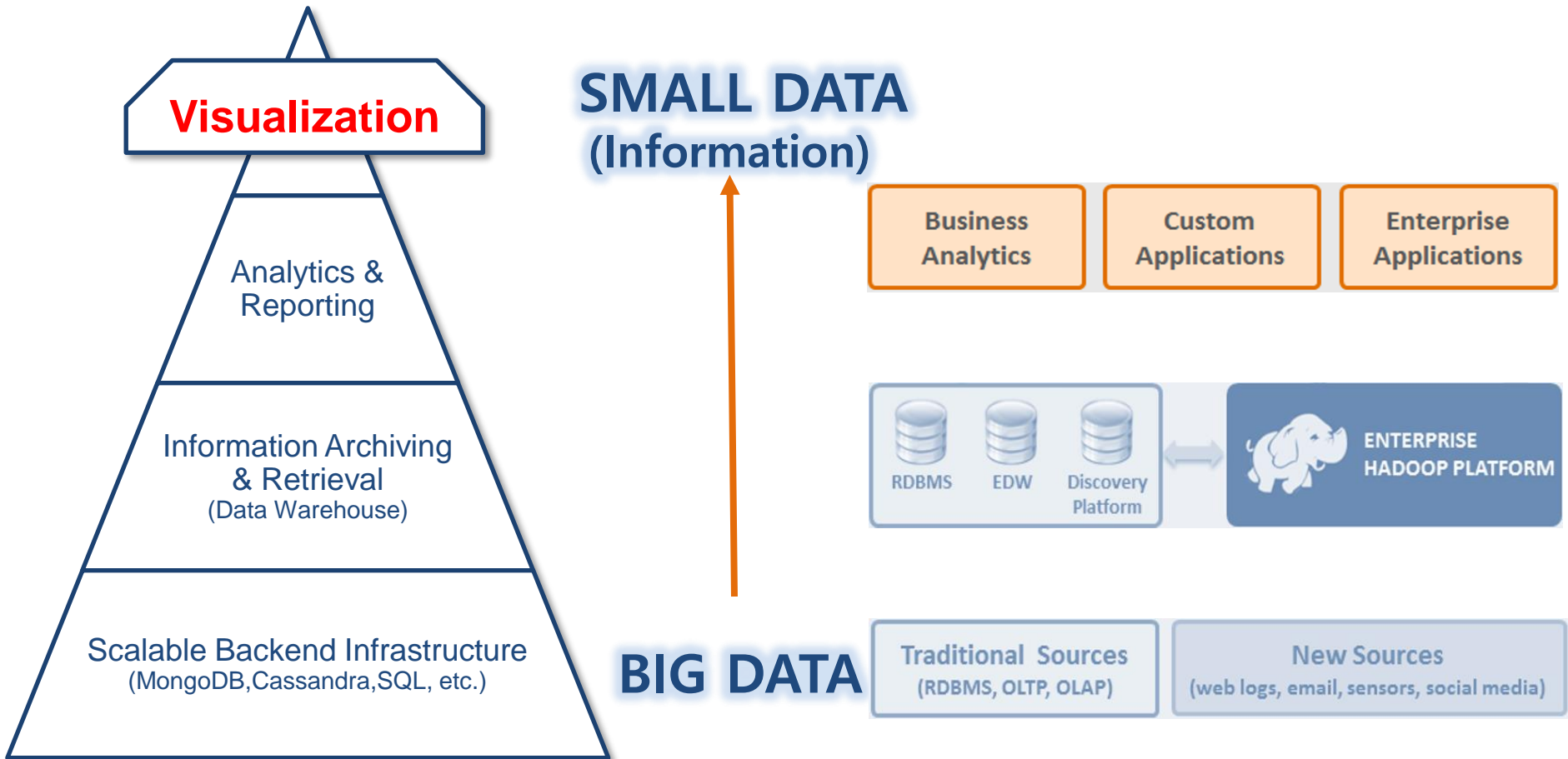
- 광범위하게 분산된 방대한 양의 자료를 분석해 한눈에 볼 수 있도록 도표나 차트 등으로 정리하는 것



■ 얻을수 있는 효과

- 자료로부터 정보를 습득하는 시간 절감으로 즉각적인 상황 판단이 가능
- 자료를 습득하는 사람들의 흥미를 유발하고 정보의 빠른 확산을 촉진
- 자료를 기억하는데에도 도움

_데이터 시각화



<http://www.datasciencecentral.com/profiles/blogs/small-data-is-beautiful>



데이터 시각화

빅데이터 시각화 프로세스

빅데이터 시각화 프로세스	
1단계: 정보 구조화	데이터를 수집하고 정제하는 과정으로 데이터세트를 만들기 위한 분석 도구 필요
2단계: 정보 시각화	주로 분석 도구에서 제공하는 그래프나 분석 도구의 특성에 따른 시각화
3단계: 정보 시각표현	시각화의 의도를 강화해 전달하기 위해 분석 도구에서 만든 결과물에 별도 그래픽 요소를 추가해 완성



_데이터 시각화

- 1단계 정보 구조화
 - 사실 빅데이터의 시각 표현에 있어서 많은 데이터를 수작업으로 조직화하거나 분류하기 위와 같은 다양한 **시각화 툴을 사용해 개발**하는 경우가 대부분
 - 아예 데이터를 갖고 인공지능의 한 분야인 기계학습 알고리즘을 접목해 데이터를 파악하고 이를 통해 사람이 파악할 수 없었던 결과들을 자동으로 시각 추출하는 방법도 활발히 연구되고 있음
- 2단계 정보 시각화
 - 빅데이터의 시각화를 위해서는 **전문가 수준의 툴을 이용해 다양한 데이터를 시각화하는 것이 필요**
 - 특히 빅데이터에서의 '데이터 시각화 툴'은 일반적으로 자신의 데이터를 분석하기 위한 플랫폼을 의미
 - 정보 시각화 단계에서는 시각화 툴에서 제공하는 다양한 그래프를 어떤 이유로, 왜 쓰는지, 어떻게 표현해야 하는지에 대해 설명하고자 함
 - 각각의 그래프에 대한 가이드와 예제를 통해 시각화 툴에서 제공하는 **그래프를 효율적으로 이용하게 하는 데 목표**를 두고 있음



_데이터 시각화

- 3단계 정보 시각표현
 - 최종적으로 시각적인 완성을 하는 단계로, 시각화 툴로 선택한 그래프를 시각적으로 더 다듬거나 시각 표현을 극대화하는 방안을 실험하면서 완성
 - 세부적인 데이터를 시각화하기 위한 그래픽의 7요소, 전체적인 시각화를 완성하기 위한 그래픽 디자인의 기본 원리, 인터랙션 디자인을 통해 방대한 양의 데이터 시각화를 탐험할 수 있게 하는 다양한 방법을 습득할 필요가 있음



_Context 시각화 (현장 시각화)

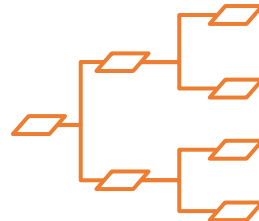
Massive Domain Data



- 현장 필요한 수많은 데이터가 산재되어 있음
- 효과적으로 통합되어야 함
(문제해결과 의사결정을 위해)

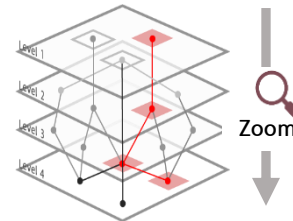
Data Visualization

Fast interactive visualization of large data structures organized in a tree



맵 구조(현장)

(1) Tree형태 구조화
(요소간 관계 및 Map
위계관계)



Drill-down View

(2) 전자지도식 Drill-down
Interaction

Replay from past to now with video



영상 융합처리

(3) 데이터&영상 Playback

Insight Driven Operation

시나리오에 따른
현장운영 및 통제



Database : Mysql

_Database

1. mysql 설치

<https://downloads.mysql.com/archives/installer/>

- 버전은 5.7.xx 로 선택

MySQL Product Archives

MySQL Installer (Archived Versions)

Please note that these are old versions. New releases will have recent bug fixes and features!
To download the latest release of MySQL Installer, please visit [MySQL Downloads](#).

Product Version:

Operating System:

Windows (x86, 32-bit), MSI Installer

Apr 6, 2022

(mysql-installer-web-community-5.7.38.0.msi)

Windows (x86, 32-bit), MSI Installer

Apr 6, 2022

(mysql-installer-community-5.7.38.0.msi)

i We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

2. workbanch 설치

<https://downloads.mysql.com/archives/workbench/>

- 버전은 6.3.xx 로 선택

MySQL Product Archives

MySQL Workbench (Archived Versions)

Please note that these are old versions. New releases will have recent bug fixes and features!
To download the latest release of MySQL Workbench, please visit [MySQL Downloads](#).

Product Version:

Operating System:

Windows (x86, 32-bit), MSI Installer

Feb 15, 2013

(mysql-workbench-gpl-5.2.47-win32.msi)

Windows (x86, 32-bit), ZIP Archive

Feb 15, 2013

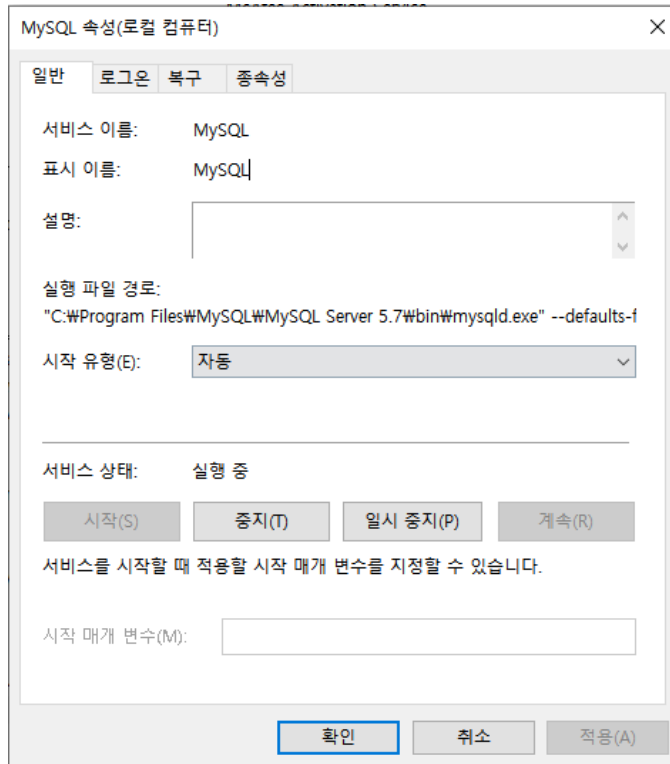
(mysql-workbench-gpl-5.2.47-win32-noinstall.zip)

i We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.



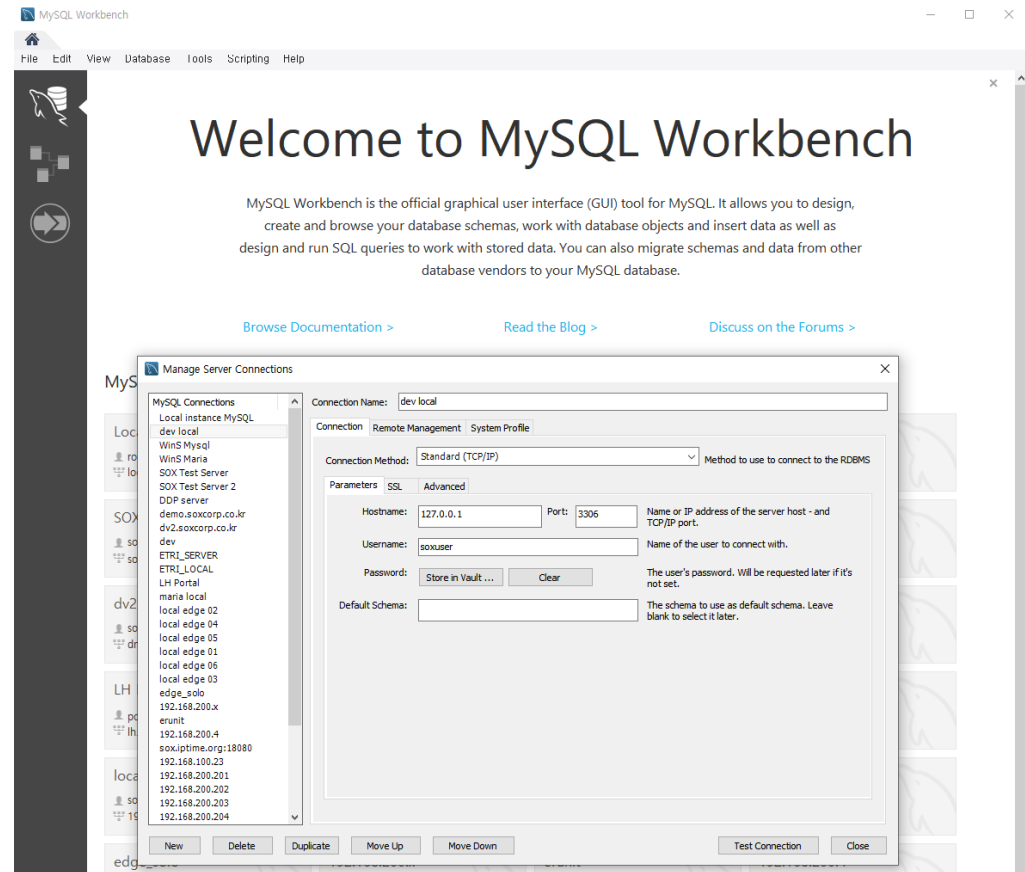
_Database

1. mysql 실행 : 서비스에서 실행



2. workbench 실행

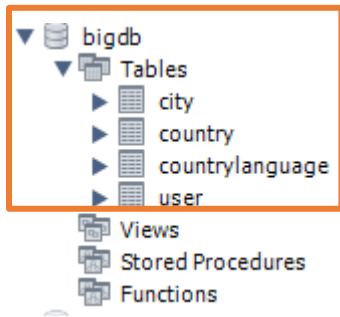
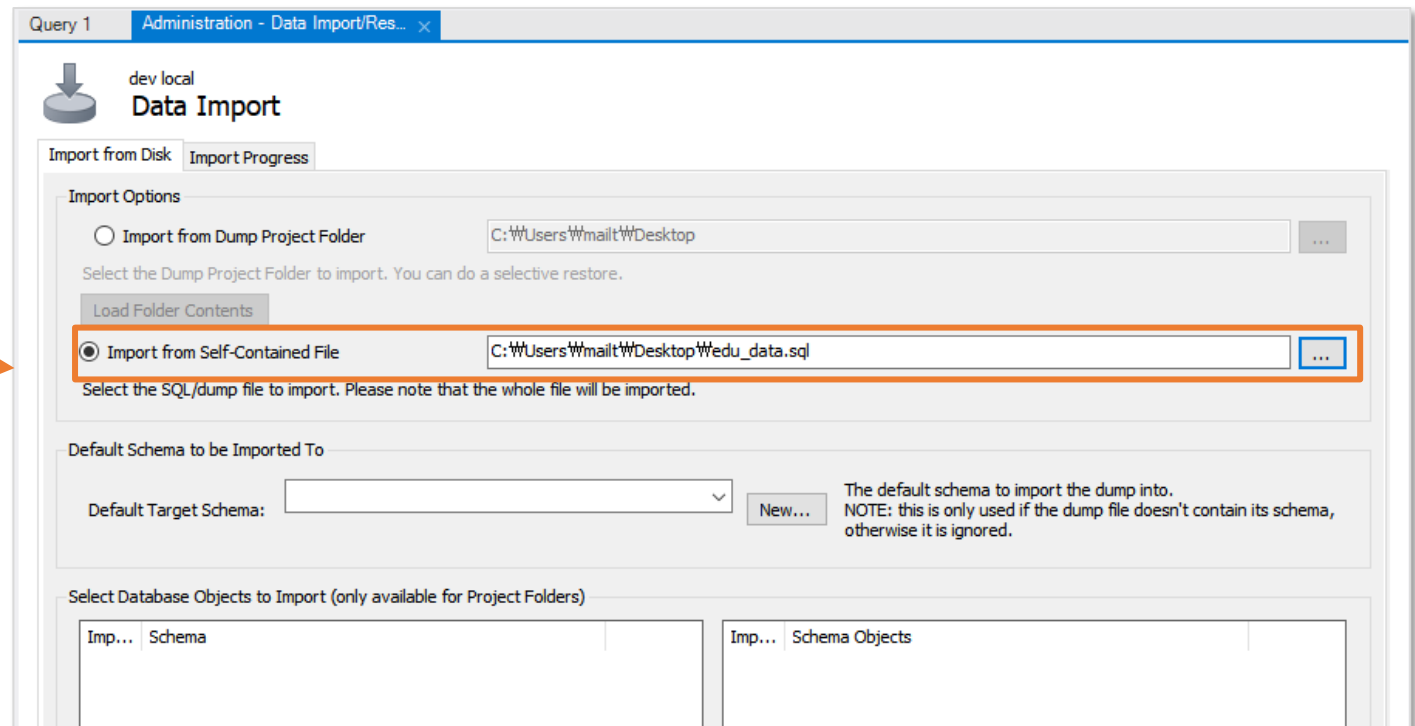
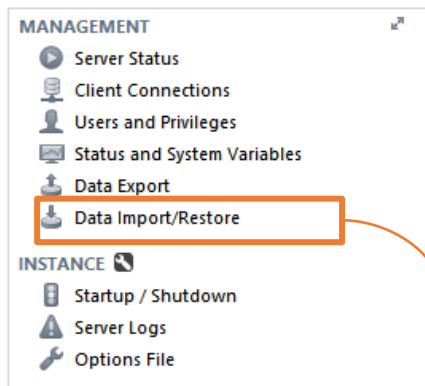
: 접속 정보 입력 후 더블클릭



_Database

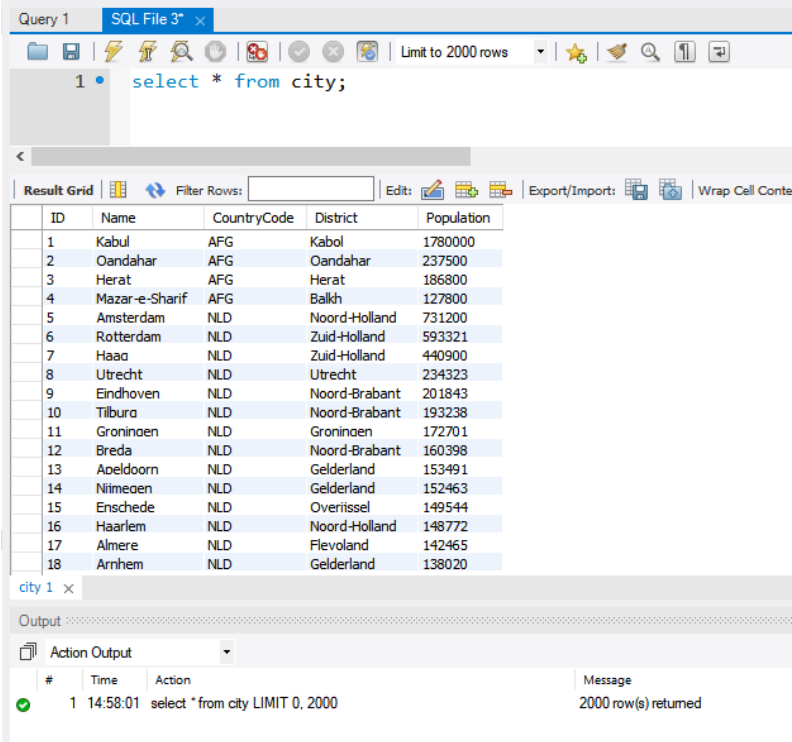
실습 데이터 임포트 하기 : 'edu_data.sql '

→ bigdb 스키마 생성



_Database

- SELECT는 테이블의 데이터를 읽어 출력
- SELECT 필드목록
FROM 테이블
[WHERE 조건]
[ORDER BY 정렬기준]
- SELECT와 FROM 사이의 필드 목록에 출력할
필드의 이름을 지정하되 * 기호는 모든 필드 출력
 - SELECT * FROM city;
- WHERE 절은 읽을 레코드의 조건을 지정한다.
 - 필드와 특정값을 비교하는 조건문 형식으로 작성한다.
 - SELECT * FROM city WHERE Population > 700000;
 - 특정 레코드의 특정 필드만 표시
 - SELECT Name, Population FROM city WHERE Population > 700000;
 - 문자열이나 날짜는 작은 따옴표로 감싼다.
 - AND는 두 조건이 모두 참인 레코드를 검색하며 OR는 두 조건 중 하나라도 참인 레코드를 검사
 - SELECT * FROM city WHERE Population >= 100000 AND Population >= 700000;



The screenshot shows a SQL query editor with a query window and a results window. The query window contains the query: `select * from city;`. The results window displays a table with 18 rows of data. The table has columns: ID, Name, CountryCode, District, and Population. The data includes cities from Afghanistan (Kabul, Oandahar, Herat, Mazar-e-Sharif) and the Netherlands (Amsterdam, Rotterdam, Haao, Utrecht, Eindhoven, Tilbura, Groninoen, Breda, Apeldoorn, Niimeoen, Enschede, Haarlem, Almere, Arnhem).

ID	Name	CountryCode	District	Population
1	Kabul	AFG	Kabul	1780000
2	Oandahar	AFG	Oandahar	237500
3	Herat	AFG	Herat	186800
4	Mazar-e-Sharif	AFG	Balkh	127800
5	Amsterdam	NLD	Noord-Holland	731200
6	Rotterdam	NLD	Zuid-Holland	593321
7	Haao	NLD	Zuid-Holland	440900
8	Utrecht	NLD	Utrecht	234323
9	Eindhoven	NLD	Noord-Brabant	201843
10	Tilbura	NLD	Noord-Brabant	193238
11	Groninoen	NLD	Groninoen	172701
12	Breda	NLD	Noord-Brabant	160398
13	Apeldoorn	NLD	Gelderland	153491
14	Niimeoen	NLD	Gelderland	152463
15	Enschede	NLD	Overijssel	149544
16	Haarlem	NLD	Noord-Holland	148772
17	Almere	NLD	Flevoland	142465
18	Arnhem	NLD	Gelderland	138020

city 1 x

Output

Action Output

#	Time	Action	Message
1	14:58:01	select * from city LIMIT 0, 2000	2000 row(s) returned



_Database

■ WHERE 절은 읽을 레코드의 조건을 지정한다.

■ LIKE 연산자는 패턴으로 부분 문자열을 검색한다.

- SELECT * FROM city WHERE name LIKE 'Over%';

문자	설명
%	복수개의 문자와 대응한다. 도스의 *와 동일한 의미를 가지며 %자리에는 임의 개수의 임의 문자가 올 수 있다.
_	하나의 문자와 대응한다. 도스의 ?와 동일한 의미를 가지며 _자리에 하나의 임의 문자가 올 수 있다.
[]	[]안에 포함된 문자 리스트 중 하나의 문자와 대응한다.
[^]	[^]안에 포함된 문자 리스트에 포함되지 않은 하나의 문자와 대응한다.

■ “BETWEEN 최소값 AND 최대값” 형식으로 두 값 사이의 범위를 제한한다.

- SELECT * FROM city WHERE population BETWEEN 500000 AND 700000;
- SELECT * FROM tStaff WHERE joindate BETWEEN '20150101' AND '20180101';

■ IN 연산자는 불연속적인 값 여러 개의 목록을 제공하여 이 목록과 일치하는 레코드를 검색한다.

- SELECT * FROM city WHERE name IN ('경상', '전라');

■ ORDER BY 절로 정렬 순서를 지정한다.

ORDER BY 필드 [ASC | DESC]

- 오름차순 ASC가 디폴트이며 생략시 ASC를 적용한다.
- SELECT * FROM city ORDER BY population;
- SELECT * FROM city ORDER BY population DESC;



_Database

- 중복된 값을 제거할 때는 DISTINCT 키워드를 붙인다.
 - SELECT DISTINCT CountryCode FROM city;
 - SELECT count(DISTINCT CountryCode) FROM city;
- SELECT ... LIMIT 총개수
 - 인구 상위 4개 도시 구하기
SELECT * FROM city ORDER BY population DESC LIMIT 4;
- 집계 함수(Aggregate Function)는 복수개의 레코드에 대해 집합적인 계산을 수행하여 합계, 평균, 분산 같은 통계값을 산출한다.
 - SELECT COUNT(*) as 'CityCount', CountryCode FROM city GROUP BY CountryCode ;

함수	설명
SUM	총합을 구한다.
AVG	평균을 구한다.
MIN	최소값을 구한다.
MAX	최대값을 구한다.
STDDEV	표준 편차를 구한다. MSSQL은 함수명이 STDEV이다.
VARIANCE	분산을 구한다. MSSQL은 함수명이 VAR이다.



_Database

- 레코드를 추가하는 명령은 INSERT이다.
 - INSERT INTO 테이블 (필드목록) VALUES (값목록)
 - INSERT INTO city (Name, CountryCode, Population) VALUES ('Seoul', 'KOR', 10000000);
- 명령문과 필드 목록은 딱 한 번만 밝히고 실제 삽입할 데이터만 나열할 수 있다.
 - INSERT INTO city (Name, CountryCode, Population) VALUES ('Busan', 'KOR', 5000000), ('Daejeon', 'KOR', 4000000);
- 레코드를 삭제할 때는 DELETE 명령을 사용한다.
 - DELETE FROM 테이블 WHERE 조건
 - DELETE FROM city WHERE Name = 'Busan';
- 레코드의 필드 값을 변경할 때는 UPDATE 명령을 사용한다. SET 키워드 뒤에 필드에 값을 대입하는 대입문이 오며 콤마로 끊어 복수개의 필드를 한꺼번에 변경한다.
 - UPDATE 테이블 SET 필드=값 [,필드=값] WHERE 조건
 - 서울의 인구를 1100만명으로 변경
UPDATE city SET Population = 11000000 WHERE Name = 'Seoul';



_Database 실습

■ Workbench 사용

The screenshot displays the MySQL Workbench interface. The top pane shows a list of SQL queries. The second query, `SELECT * FROM city;`, is highlighted with an orange box. Below the queries, the 'Result Grid' shows the output of this query, also highlighted with an orange box. The grid contains columns: ID, Name, CountryCode, District, and Population. The bottom pane shows the 'Action Output' log, which records the execution of the query and the number of rows returned.

ID	Name	CountryCode	District	Population
4064	Odessa	USA	Texas	89293
4065	Carson	USA	California	89089
4066	Charleston	USA	South Carolina	89063
4067	Charlotte Amalie	VIR	St Thomas	13000
4068	Harare	ZWE	Harare	1410000
4069	Bulawavo	ZWE	Bulawavo	621742
4070	Chitungwiza	ZWE	Harare	274912
4071	Mount Darwin	ZWE	Harare	164362
4072	Mutare	ZWE	Manicaland	131367
4073	Gweru	ZWE	Midlands	128037
4074	Gaza	PSE	Gaza	353632
4075	Khan Yunis	PSE	Khan Yunis	123175
4076	Hebron	PSE	Hebron	119401
4077	Jabalva	PSE	North Gaza	113901
4078	Nablus	PSE	Nablus	100231
4079	Rafah	PSE	Rafah	92020
4080	Seoul	KOR		10000000
NULL	NULL	NULL	NULL	NULL

city 17 x

Output

Action Output

#	Time	Action	Message
✓ 12	21:30:22	SELECT * FROM city LIMIT 0, 100	100 row(s) returned
✓ 13	21:30:49	SELECT DISTINCT CountryCode FROM city LIMIT 0, 100	100 row(s) returned
✓ 14	21:31:12	SELECT count(DISTINCT CountryCode) FROM city LIMIT 0, 100	1 row(s) returned
✓ 15	21:33:36	SELECT * FROM city LIMIT 0, 100	100 row(s) returned
✓ 16	21:34:40	SELECT COUNT(*) as 'CityCount' CountryCode FROM city GROUP BY CountryCode LIMIT 0, 100	100 row(s) returned



_Database 정리

■ 데이터 조회

- SELECT 필드목록
FROM 테이블
[WHERE 조건]
[ORDER BY 정렬기준]
- SELECT * FROM city;

■ 레코드를 추가하는 명령은 INSERT

- INSERT INTO 테이블 (필드목록) VALUES (값목록)
- INSERT INTO city (Name, CountryCode, Population) VALUES ('Seoul', 'KOR', 10000000);

■ 레코드를 삭제할 때는 DELETE 명령

- DELETE FROM 테이블 WHERE 조건
- DELETE FROM city WHERE Name = 'Busan';

■ 레코드의 필드 값을 변경할 때는 UPDATE 명령

- UPDATE 테이블 SET 필드=값 [,필드=값] WHERE 조건
- 서울의 인구를 1000만명으로, 지역을 충청도로 변경한다.
UPDATE city SET Population = 11000000 WHERE Name = 'Seoul';



Node-Red DB 연결

_Node-Red Database

데이터 베이스 연결 노드 추가 (커스텀 노드 추가)

- 1) 메뉴에서 '팔레트 관리' 선택
- 2) '설치가 가능한 노드'에서 mysql로 검색
- 3) 추가 노드 설치 → 저장 팔레트에 'mysql'노드 추가 됨

The image shows the Node-RED web interface with three main components highlighted to illustrate the steps:

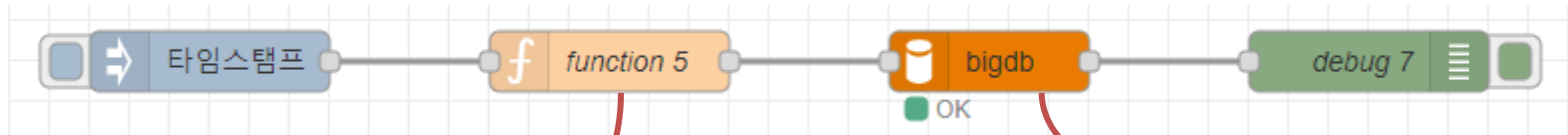
- Left Panel (Menu):** The 'palette management' (팔레트 관리) option is selected and highlighted with a red box. A red arrow points from this menu item to the right panel.
- Right Panel (Node Search):** The '설치 가능한 노드' (Installable Nodes) tab is active. The search bar contains 'mysql'. The search results show two nodes: 'node-red-node-mysql' and 'node-red-node-mysql-test'. The '설치됨' (Installed) button for the first node is highlighted with a red box.
- Bottom Panel (Saved Palette):** The '저장' (Saved) palette is shown, containing nodes like 'write file', 'read file', 'watch', and 'mysql'. The 'mysql' node is highlighted with a red box, indicating it has been successfully added to the saved palette.



_Node-Red Database

데이터 베이스 연결 노드 만들기

- 1) 쿼리 노드 만들기 : 함수 노드 혹은 템플릿 노드로 만든다
- 2) 데이터베이스 노드에 연결할 데이터베이스 설정을 입력



속성

이름: function 5

Setup On Start 코드 On Stop

```
1 msg.payload = {};  
2 msg.topic =  
3 "SELECT * FROM city limit 10 ;";  
4  
5 return msg;
```

- * msg.topic 에 쿼리를 담음
- * 함수 노드 대신에 template 노드를 활용 해도 됨

속성

이름: 이름

프로퍼티: msg. topic

템플릿

```
1 SELECT * FROM city limit 10 ;
```

구문: mustache

속성

Database: bigdb

이름: 이름

속성

Host: 127.0.0.1

Port: 3306

User: soxuser

Password:

Database: bigdb

Timezone: ±hh:mm

Charset: UTF8

* Database 접속정보 입력

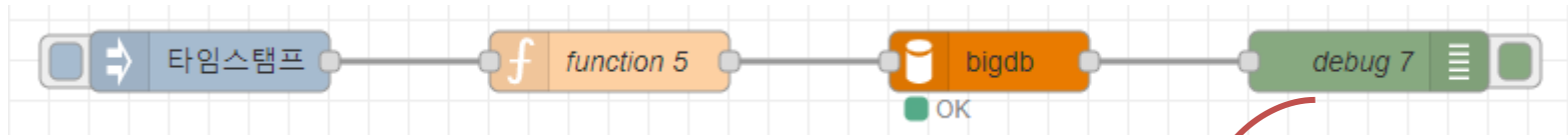


_Node-Red Database

데이터 베이스 연결 노드 만들기

3) 쿼리 결과 가공

4) http로 보내거나 로그로 출력

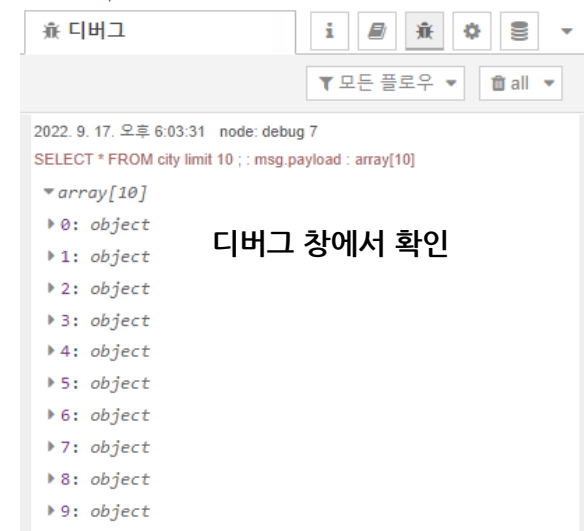


쿼리 결과를 가공하려면 함수노드를 둔다.

```
1 var query_result = msg.payload;
2 var resultlist = [];
3
4 for(var i=0 ; i < query_result.length ; i++){
5     var item = {};
6     for(var k in query_result[i]) {
7         item[k] = query_result[i][k];
8     }
9     resultlist.push(item);
10 }
11 msg.payload = resultlist;
12
13
14
15
16 return msg;
```

엘리먼트에 적용할 수 있도록 데이터 가공

* 쿼리 결과는 msg.payload로 넘어옴



_DB 불러오기 실습

실습 문제 : city 데이터에서 인구 70만 이상인 도시만 쿼리하는 Node-red 노드를 구성하고
AJAX로 불러오기




테이블 및 차트 시각화


_DataTable


데이터 테이블 만들기

- 1) 라이브러리 추가 : JQuery, DataTable
- 2) 테이블이 위치할 지점에 <table> 태그 정의
- 3) 자바스트립트로 내용 및 옵션 정의 → `$("#example").DataTable({ ... });`

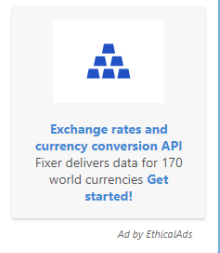
<https://datatables.net/examples/ajax/objects.html>

 CloudTables

 DataTables

 Editor

[Manual](#) [Download](#) [Login / Register](#)



Exchange rates and currency conversion API
Fixer delivers data for 170 world currencies [Get started!](#)
Ad by EthicalAds

Add advanced interaction controls to your HTML tables *the free & easy way*

1 - Include these two files ↴

CSS

`//cdn.datatables.net/1.12.1/css/jquery.dataTables.min.css`

JS

`//cdn.datatables.net/1.12.1/js/jquery.dataTables.min.js`

2 - Call this single function ↴

```
1 $(document).ready( function () {  
2   $('#myTable').DataTable();  
3 } );
```

3 - You get a fully interactive table →

Full Getting Started Guide

Show entries

Search:

Name	Position	Office	Age	Start date
Aliri Satou	Accountant	Tokyo	33	2008. 11. 28.
Angelica Ramos	Chief Executive Officer (CEO)	London	47	2009. 10. 9.
Ashton Cox	Junior Technical Author	San Francisco	66	2009. 1. 12.
Bradley Greer	Software Engineer	London	41	2012. 10. 13.
Brenden Wagner	Software Engineer	San Francisco	28	2011. 6. 7.
Brielle Williamson	Integration Specialist	New York	61	2012. 12. 2.
Bruno Nash	Software Engineer	London	38	2011. 5. 3.
Caesar Vance	Pre-Sales Support	New York	21	2011. 12. 12.
Cara Stevens	Sales Assistant	New York	46	2011. 12. 6.
Cedric Kelly	Senior Javascript Developer	Edinburgh	22	2012. 3. 29.

NamePositionOfficeAgeStart date

Showing 1 to 10 of 57 entries

Previous123456Next



_DataTable 실습

```
<script src="https://code.jquery.com/jquery-3.5.1.js"></script>
<link href="https://cdn.datatables.net/1.12.1/css/jquery.dataTables.min.css" rel="stylesheet" />
<script src="https://cdn.datatables.net/1.12.1/js/jquery.dataTables.min.js"></script>

<table id="example" class="display" style="width: 100%">
  <thead>
    <tr>
      <th>Name</th>
      <th>Position</th>
      <th>Office</th>
      <th>Extn.</th>
      <th>Start date</th>
      <th>Salary</th>
    </tr>
  </thead>
</table>
```



_DataTable 실습

```
<script>
$(document).ready(function () {
    $("#example").DataTable({
        ajax: "tabledata.json",
        columns: [
            { data: "name" },
            { data: "position" },
            { data: "office" },
            { data: "extn" },
            { data: "start_date" },
            { data: "salary" },
        ],
    });
});
</script>
```

Show entries

Search:

Name	Position	Office	Extn.	Start date	Salary
Airi Satou	Accountant	Tokyo	5407	2008/11/28	162700
Ashton Cox	Junior Technical Author	San Francisco	1562	2009/01/12	86000
Bradley Greer	Software Engineer	London	2558	2012/10/13	132000
Brielle Williamson	Integration Specialist	New York	4804	2012/12/02	372000
Cedric Kelly	Senior Javascript Developer	Edinburgh	6224	2012/03/29	433060
Charde Marshall	Regional Director	San Francisco	6741	2008/10/16	470600
Colleen Hurst	Javascript Developer	San Francisco	2360	2009/09/15	205500
Dai Rios	Personnel Lead	Edinburgh	2290	2012/09/26	217500
Garrett Winters	Accountant	Tokyo	8422	2011/07/25	170750
Gloria Little	Systems Administrator	New York	1721	2009/04/10	237500
Name	Position	Office	Extn.	Start date	Salary

Showing 1 to 10 of 20 entries

Previous 2 Next

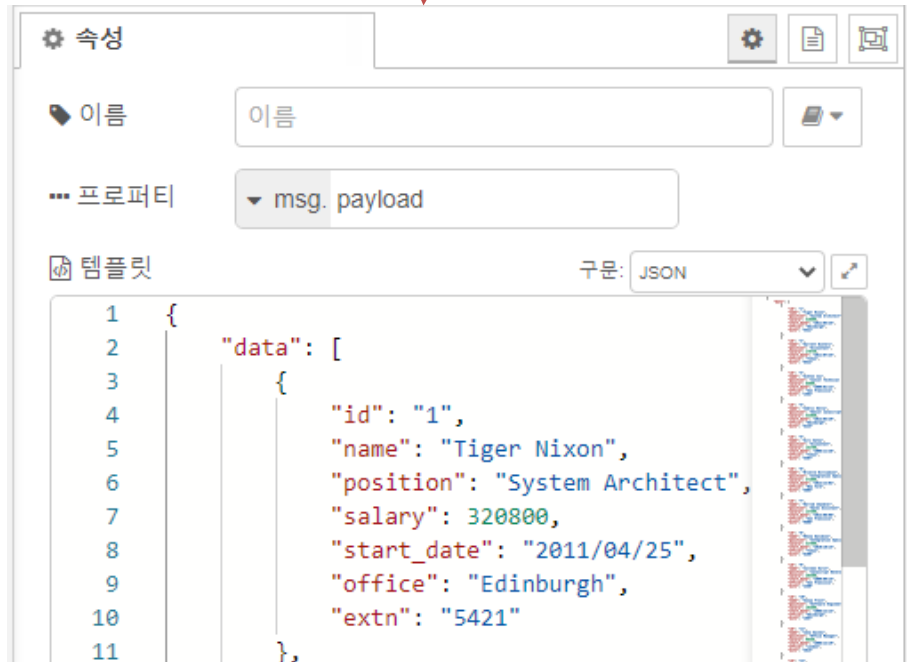


_DataTable 실습

실습 문제 : Node-Red에서 데이터 불러오기



```
<script>
$(document).ready(function () {
  $("#example").DataTable({
    ajax: "http://localhost:1880/tabledata",
    columns: [
      { data: "name" },
      { data: "position" },
      { data: "office" },
      { data: "extn" },
      { data: "start_date" },
      { data: "salary" },
    ],
  });
});
</script>
```



_DataTable 실습

실습 문제 : 다음 URL에서 데이터를 읽어서 일부 데이터만 테이블에 표기하기

→ <http://make-random.com/MakeRandom/api/userInfo.get>

The diagram illustrates a workflow for fetching and processing data from a specific URL. The workflow consists of four nodes: a '[get] /tabledata2' node, an 'http request' node, a 'function 4' node, and an 'http' node. Red arrows indicate the configuration panels for the 'http request' and 'function 4' nodes.

http request configuration:

- 속성 (Properties): GET
- URL: <http://make-random.com/MakeRandom/api/userInfo.get>
- 페이로드 (Payload): ignore
- SSL/TLS접속을 유효화 (Enable SSL/TLS): ☐
- 인증을 사용 (Use authentication): ☐
- Enable connection keep-alive: ☐
- 프록시를 사용 (Use proxy): ☐
- Only send non-2xx responses to Catch node: ☐
- Disable strict HTTP parsing: ☐
- 출력형식 (Output format): JSON오브젝트 (JSON object)

function 4 configuration:

```
1 var data = msg.payload;
2
3 var convert = [];
4
5 for(let line of data){
6   let obj = {};
7   obj.id = line.id;
8   obj.website = line.website;
9   obj.address = line.address.address;
10  obj.phone = line.phone;
11  obj.name = line.name;
12  obj.company = line.company.name;
13  obj.email = line.email;
14
15  convert.push(obj);
16 }
17
18 msg.payload = {};
19 msg.payload.data = convert;
20
21 return msg;
```



_DataTable 실습

```
<table id="example" class="display" style="width: 100%">
  <thead>
    <tr>
      <th>ID</th>
      <th>Name</th>
      <th>website</th>
      <th>address</th>
      <th>phone</th>
      <th>company</th>
      <th>email</th>
    </tr>
  </thead>
</table>
```

```
<script>
  $(document).ready(function () {
    $("#example").DataTable({
      ajax: "http://localhost:1880/tabledata2",
      columns: [
        { data: "id" },
        { data: "name" },
        { data: "website" },
        { data: "address" },
        { data: "phone" },
        { data: "company" },
        { data: "email" },
      ],
    });
  });
</script>
```



_e-charts


차트 만들기

- 1) 라이브러리 추가 : echarts → <https://fastly.jsdelivrivr.net/npm/echarts@5.3.3/dist/echarts.min.js>
- 2) 테이블이 위치할 지점에 <div> 태그 정의
- 3) 자바스크립트로 내용 및 옵션 정의 → `var myChart = echarts.init(...); myChart.setOption(option);`

<https://echarts.apache.org/examples/en/index.html>

Apache ECharts

An Open Source JavaScript Visualization Library

 Get Started

 Demo



e-charts

```
<script src="https://fastly.jsdelivrivr.net/npm/echarts@5.3.3/dist/echarts.min.js"></script>

<div id="container" style="height: 100%"></div>
<script type="text/javascript">
  var dom = document.getElementById("container");
  var myChart = echarts.init(dom, null, {
    renderer: "canvas",
    useDirtyRect: false,
  });
  var option = {
    xAxis: {
      type: "category",
      data: ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"],
    },
    yAxis: {
      type: "value",
    },
    series: [
      {
        data: [150, 230, 224, 218, 135, 147, 260],
        type: "line",
      },
    ],
  };
  if (option && typeof option === "object") {
    myChart.setOption(option);
  }
</script>
```



e-charts 예제 공통 코드

```
<script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>
<script src="https://fastly.jsdelivrivr.net/npm/echarts@5.3.3/dist/echarts.min.js"></script>
<div id="container" style="height: 100%"></div>
<script type="text/javascript">
    var dom = document.getElementById("container");
    var myChart = echarts.init(dom, null, {
        renderer: "canvas",
        useDirtyRect: false,
    });

    var option;

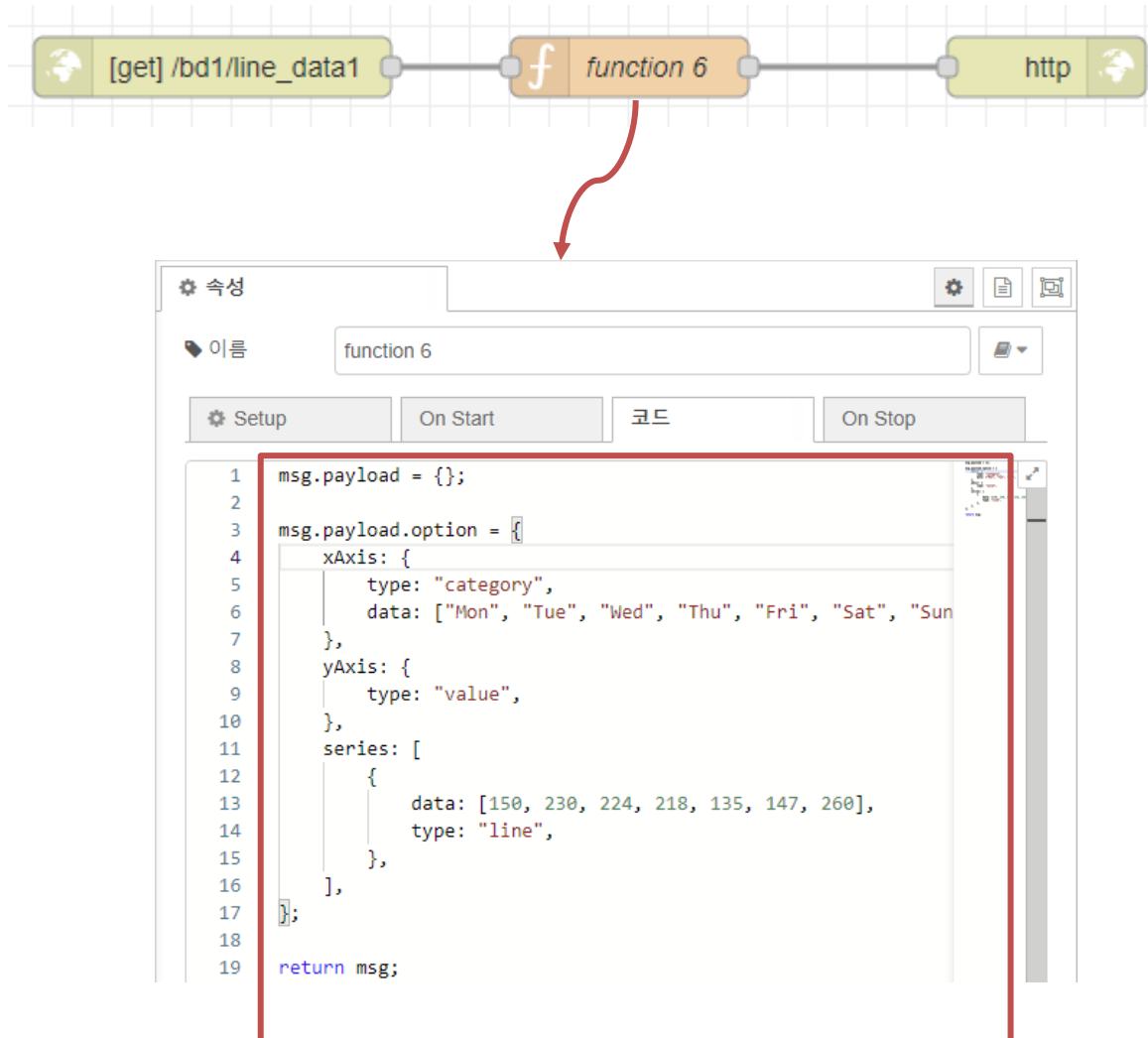
    $.ajax({
        type: "GET",
        url: "http://127.0.0.1:1880/bd1/line_data1", // → 이 부분만 계속 바꾸면 됨
        dataType: "json",
        success: function (data) {
            option = data.option;
            if (option && typeof option === "object") {
                myChart.setOption(option);
            }
        },
        error: function () {
            console.log("통신실패!!!!");
        },
    });

</script>
```



_e-charts & Node-Red

1. Line 차트 예제



_e-charts & Node-Red

2. Bar 차트 예제



속성

이름: function 7

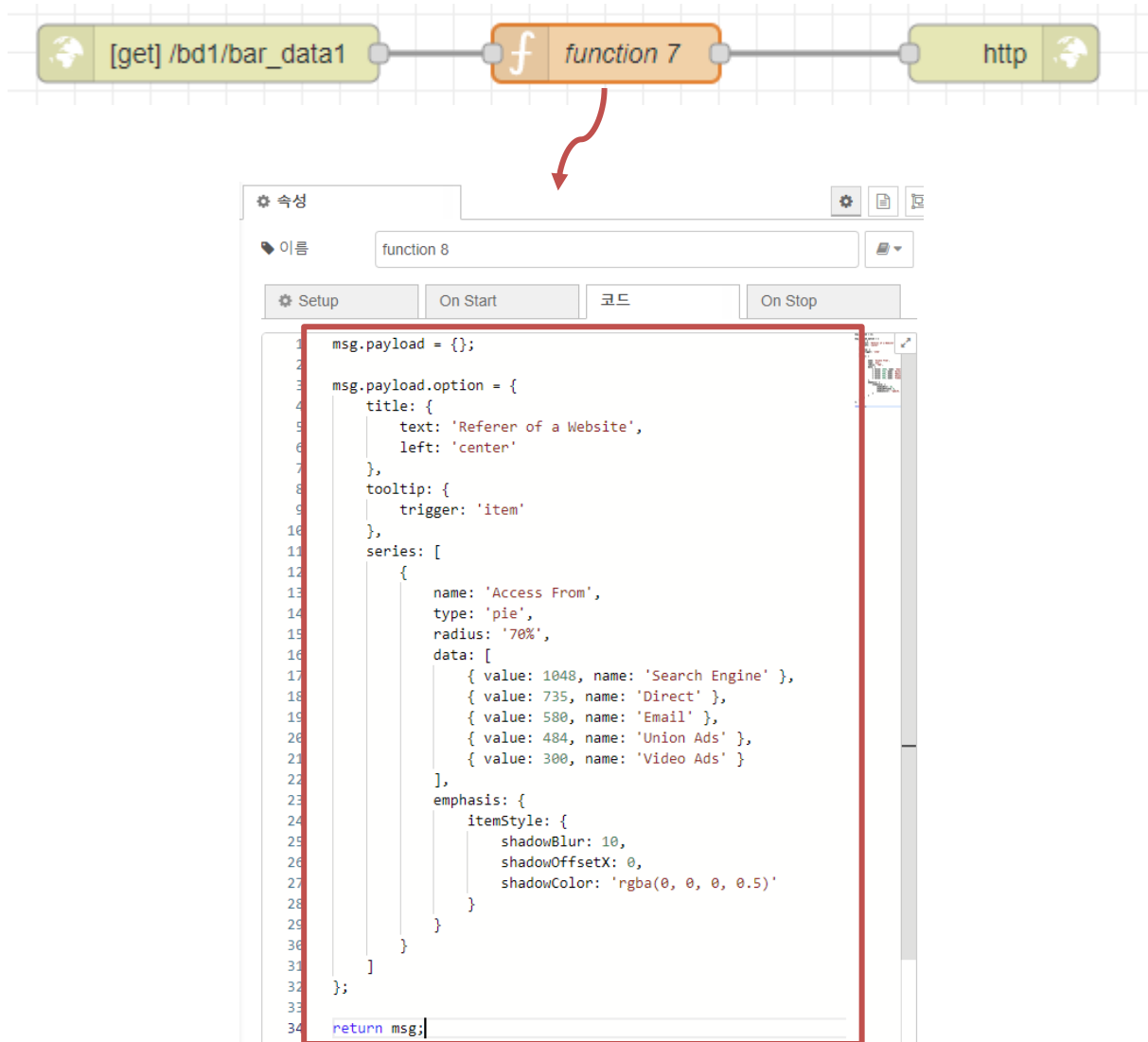
Setup On Start 코드 On Stop

```
1 msg.payload = {};  
2  
3 msg.payload.option = {  
4   xAxis: {  
5     type: 'category',  
6     data: ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun'],  
7   },  
8   yAxis: {  
9     type: 'value'  
10  },  
11  series: [  
12    {  
13      data: [120, 200, 150, 80, 70, 110, 130],  
14      type: 'bar'  
15    }  
16  ]  
17 };  
18  
19 return msg;
```



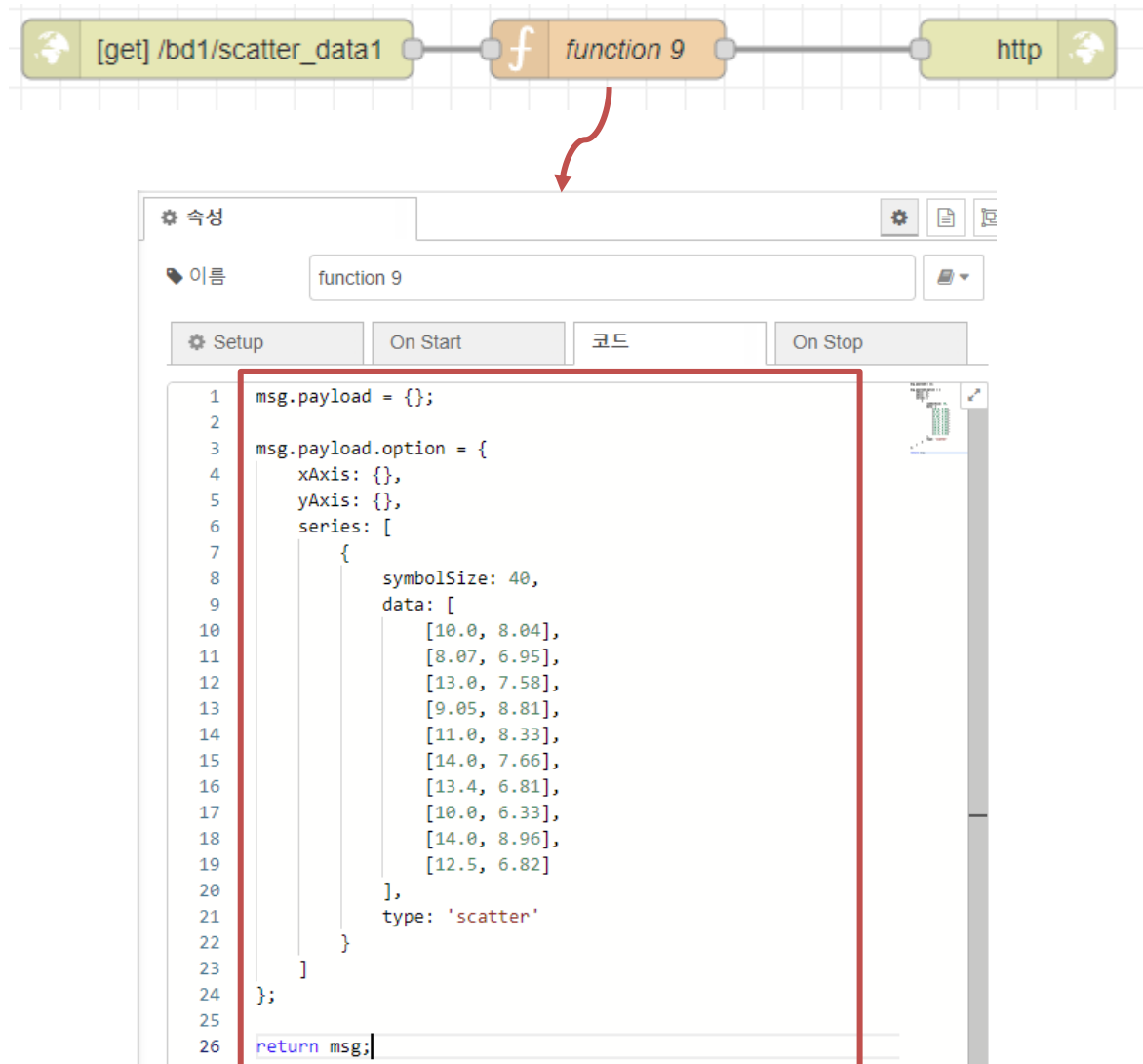
_e-charts & Node-Red

3. Pie 차트 예제



_e-charts & Node-Red

4. Scatter 차트 예제



_e-charts & Node-Red

5. Gauge 차트 예제

The image shows a Node-Red flow diagram at the top with three nodes: a `[get] /bd1/gauge_data1` node, a `function 10` node, and an `http` node. A red arrow points from the `function 10` node to its configuration editor below.

The configuration editor for the `function 10` node is shown. It has tabs for `속성` (Properties), `이름` (Name), `Setup`, `On Start`, `코드` (Code), and `On Stop`. The `코드` tab is selected, showing the following JavaScript code:

```
1 msg.payload = {};  
2  
3 var value = 45;  
4  
5 msg.payload.option = {  
6   tooltip: {  
7     formatter: '{a} <br/>{b} : {c}%'  
8   },  
9   series: [  
10    {  
11      name: 'Progress',  
12      type: 'gauge',  
13      min: 0,  
14      max: 50,  
15      startAngle: 180,  
16      endAngle: 0,  
17      title: {  
18        fontWeight: 'bolder',  
19        fontSize: 20,  
20        fontStyle: 'italic'  
21      },  
22      detail: { formatter: '{value} C' },  
23      data: [{ value: value, name: '온도' }]  
24    }  
25  ]  
26 };  
27  
28 return msg;
```



시각화 Player

_Drilldown Play Player

1

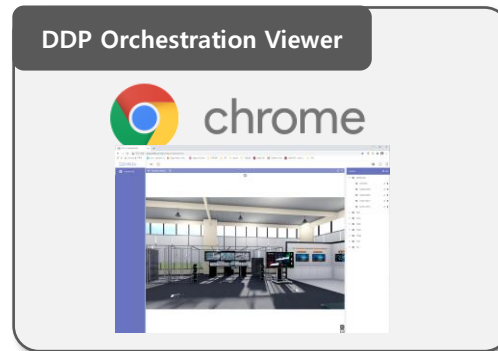
- 배경 및 엘리먼트 생성 등에 대한 그래픽 편집
- 정적인 스타일은 여기에서 설정 (테이블, 차트는 제외)



로컬 PC

2

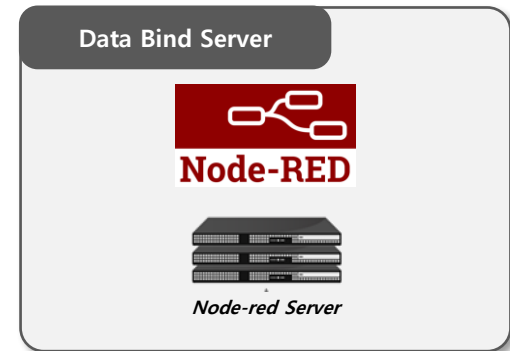
- 엘리먼트와 데이터 노드 연결 설정
- URL을 지정함



<http://dev2.soxcorp.co.kr/DrilldownPlay>

3

- 외부 데이터(데이터 소스)와 연결
- 엘리먼트에 데이터를 전송할 인터페이스 설정
- Rect, Alarm 동적 스타일 지정
- Table, Chart 스타일 지정



[http:// dev2.soxcorp.co.kr :1880](http://dev2.soxcorp.co.kr:1880)



_Drilldown Play Player

- 로그인
- 사용자 계정은 발급 받아야 합니다.



_Drilldown Play 데이터 객체 편집

- 대시보드 업로드 아이콘을 누르면 등록창이 팝업됨
- 이름을 지정하고 대시보드 파일을 선택
- Guid등의 정보는 자동으로 채워짐

temp line

scatter_test

신규 콘텐츠 등록

Example : 0 ~ 1000 GO

INFORMATION

Guid
dee4e3cf-56ed-458b-8ace-cdbc4ce46b7d

Name
이름 지정

Content Type
타입 선택

Data URLs
바인딩 노드에 대한 URL

Reload Time
리프레시 주기

Resource File

SAVE

Contents

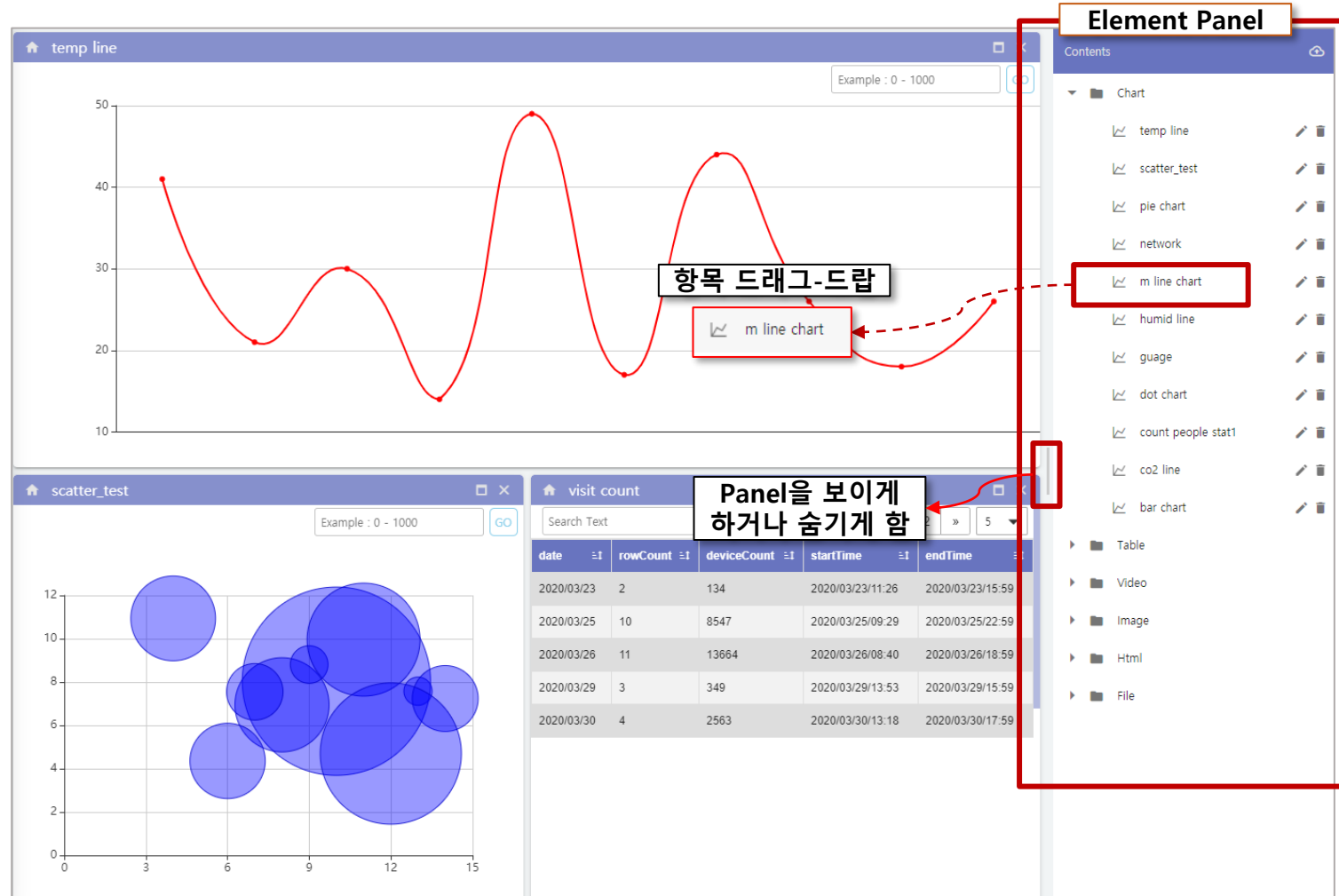
- Chart
 - temp_line
 - scatter_test
 - pie chart
 - network
 - m line chart
 - humid line
 - guage
 - dot chart
 - count people stat1
 - co2 line
 - bar chart
- Table
- Video
- Image
- Html
- File

2020/03/25	10	8547	2020/03/25/09:29	2020/03/25/22:59
2020/03/26	11	13664	2020/03/26/08:40	2020/03/26/18:59
2020/03/29	3	349	2020/03/29/13:53	2020/03/29/15:59
2020/03/30	4	2563	2020/03/30/13:18	2020/03/30/17:59



_Drilldown Play 데이터 객체 편집

- Element Panel에서 항목을 대상 Cell에 드래그&드랍하여 Play
- 대시보드 아이콘을 다시 누르면 팝업된 패널이 숨겨짐



_Drilldown Play 데이터 객체 편집

테이블 객체

naon table	
온도	습도
32	70%
34	60%
55	80%

Node-Red 편집 내용 - 함수 노드

```
var headerlist = [  
  {text:'온도', value:'temp'},  
  {text:'습도', value:'humid'}  
];
```

헤더형식

```
var items = [  
  {temp:32, humid:'70%'},  
  {temp:34, humid:'60%'},  
  {temp:55, humid:'80%'},  
];
```

항목데이터 형식

```
msg.payload = {  
  headers: headerlist,  
  items:items  
};
```

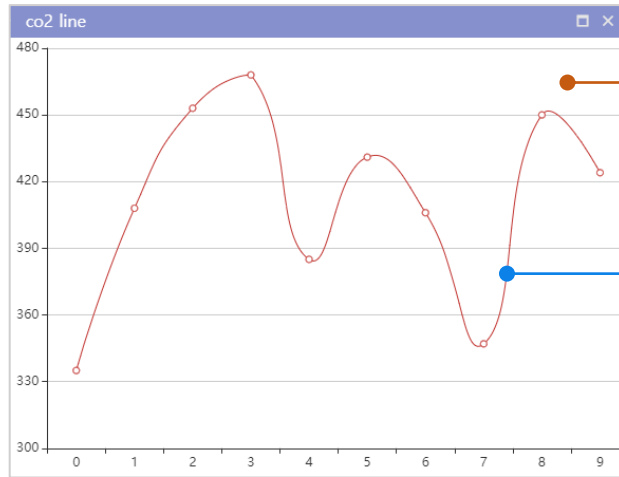
전송형식

```
return msg;
```



_Drilldown Play 데이터 객체 편집

차트 객체 - 라인차트



* dataList에 표현할 데이터를 1차원 배열로 담아 주면 됨

Node-Red 편집 내용 - 함수 노드

```
var dataList = [26,30,21,15,14,19,25,30,45,55];  
var title = '타이틀';  
  
msg.payload = {  
  grid: {  
    top:10,  
    left:'10%',  
    right:'0%',  
    bottom:'10%'  
  },  
  xAxis:{  
    type: "category",  
    data:[]  
  },  
  yAxis:{  
    type:'value',  
    scale: true,  
  },  
  series: [{  
    name:title,  
    animation: false,  
    data: dataList,  
    type: 'line',  
    smooth: true,  
    lineStyle: {  
      color: '#5470C6',  
      width: 2,  
    },  
    symbol: 'circle',  
    symbolSize: 5,  
    itemStyle: {  
      borderWidth: 1,  
      borderColor: '#5470C6',  
      color: '#5470C6'  
    }  
  }  
]  
  
return msg;
```

데이터 형식

차트 옵션 : 건드리지 않음

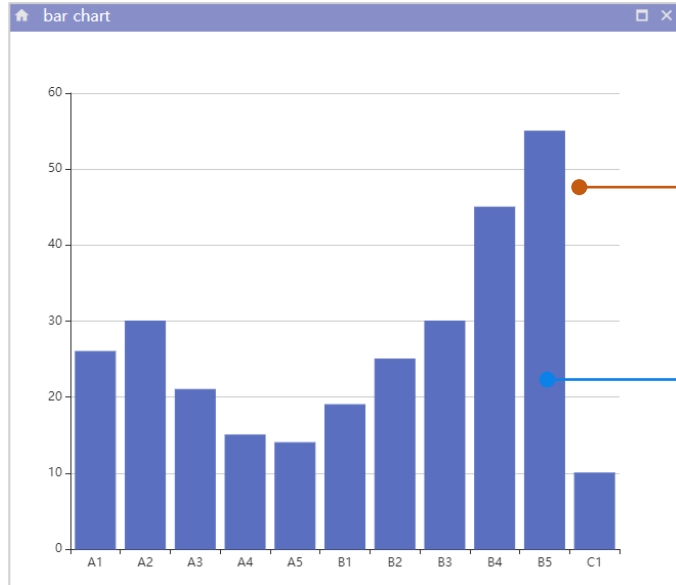
true : 곡선 / false : 직선

라인스타일 옵션



_Drilldown Play 데이터 객체 편집

차트 객체 - 바차트



* dataList에 표현할 데이터를 1차원 배열로 담고
X축에 표시할 내용을 xRowList에 담음

Node-Red 편집 내용 - 함수 노드

```
var xRowList = ["A1","A2","A3","A4","A5","B1","B2","B3","B4","B5","C1"]; X축 표시값
var dataList = [26,30,21,15,14,19,25,30,45,55,10]; 데이터

msg.payload = {
  tooltip: {},
  xAxis: {
    type: 'category',
    data: xRowList
  },
  yAxis: {
    type: 'value'
  },
  series: [
    {
      name: 'data1',
      data: dataList,
      type: 'bar',
      barWidth: '80%',
      barGap: 0,
      itemStyle: {
        borderWidth: 1,
        borderColor: '#5470C6',
        color: '#5470C6'
      }
    }
  ]
};

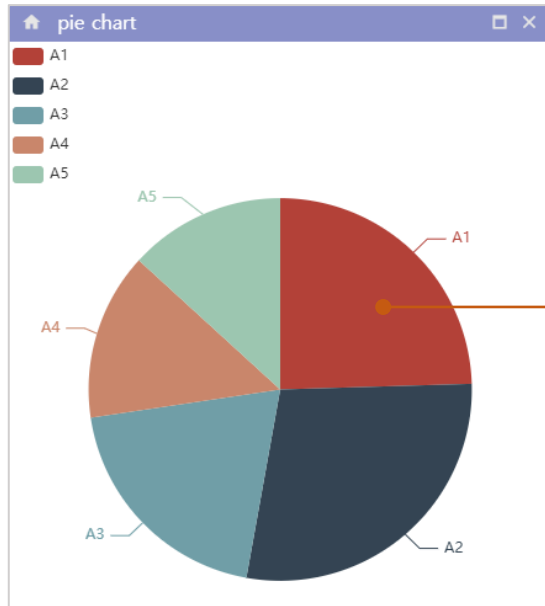
return msg;
```

바 스타일 옵션



_Drilldown Play 데이터 객체 편집

차트 객체 - 파이차트



* dataList에 표현할 데이터를 1차원 배열로 담고
라벨에 표시할 내용을 RowList에 담음

Node-Red 편집 내용 - 함수 노드

```
var rowList = ["A1","A2","A3","A4","A5"];  
var dataList = [26,30,21,15,14];  
var items = [];
```

데이터

```
// add items  
for(var i=0 ; i < dataList.length ; i++){  
  var item = {};  
  item["value"] = dataList[i];  
  item["name"] = rowList[i];  
  items.push(item);  
}
```

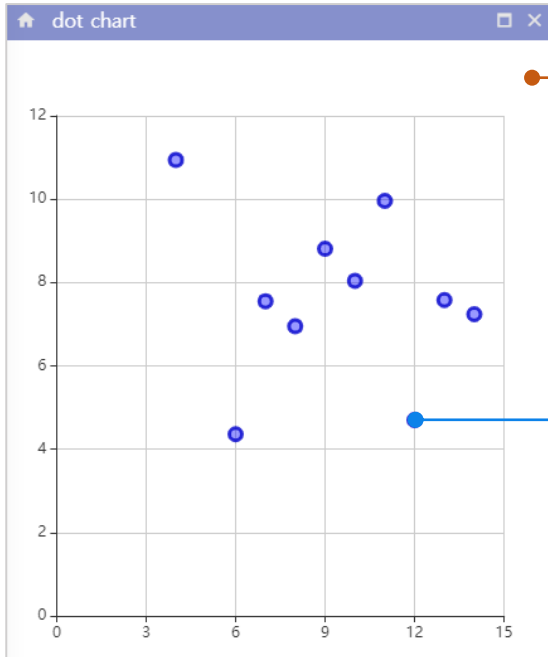
데이터를 2차원으로 구성

```
msg.payload = {  
  legend: {  
    orient: 'vertical',  
    left: 'left',  
  },  
  tooltip: {  
    trigger: 'item'  
  },  
  series: [  
    {  
      name: '점유율',  
      type: 'pie',  
      radius: '70%',  
      data: items,  
      emphasis: {  
        itemStyle: {  
          shadowBlur: 10,  
          shadowOffsetX: 0,  
          shadowColor: 'rgba(0, 0, 0, 0.5)'  
        }  
      }  
    }  
  ]  
};  
return msg;
```



_Drilldown Play 데이터 객체 편집

차트 객체 - 도트차트



- * 데이터 항목 형식은
[x좌표, y좌표, 값, id]
- x좌표 List와 y좌표 List는 필수적임

Node-Red 편집 내용 - 함수 노드

```
var rowList = ["A1","A2","A3","A4","A5","B1","B2","B3","B4","B5"];  
var xList = [10, 8, 13, 9, 11, 14, 6, 4, 12, 7];  
var yList = [8.04, 6.95, 7.58, 8.81, 9.96, 7.24, 4.36, 10.94, 4.7, 7.55];  
var dataList = [200, 100, 30, 40, 120, 70, 80, 90, 150, 60];
```

데이터

```
var items = [];  
for(var i=0 ; i < dataList.length ; i++){  
  var item = [];  
  item.push(xList[i]);  
  item.push(yList[i]);  
  item.push(dataList[i]);  
  item.push(rowList[i]);  
  items.push(item);  
}
```

데이터를 2차원으로 구성

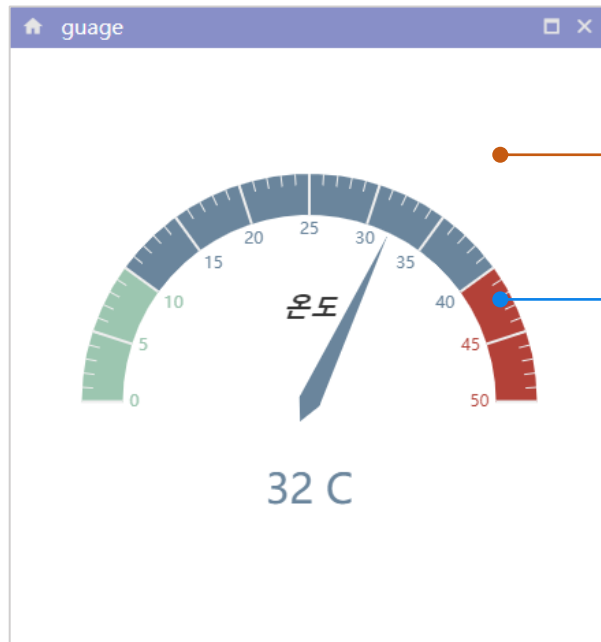
```
msg.payload = {  
  xAxis: {},  
  yAxis: {},  
  series: [{  
    color: 'rgba(0, 0, 255, 0.5)',  
    itemStyle: {  
      borderColor: '#0000cc',  
      borderWidth: 3  
    },  
    symbolSize: function (data) {  
      return data[2];  
    },  
    emphasis: {  
      label: {  
        show: true,  
        formatter: function (param) {  
          return param.data[3] + " : " + param.data[2];  
        },  
        position: 'top'  
      }  
    },  
    data: items,  
    type: 'scatter'  
  }]  
};  
return msg;
```

도트 스타일 옵션



_Drilldown Play 데이터 객체 편집

차트 객체 - 게이지차트



Node-Red 편집 내용 - 함수 노드

var value = 10; 데이터

```
msg.payload = {  
  tooltip: {  
    formatter: '{a} <br/>{b} : {c}%'  
  },  
  series: [  
    {  
      name: 'Progress',  
      type: 'gauge',  
      min: 0,  
      max: 50,  
      startAngle: 180,  
      endAngle: 0,  
      title: {  
        fontWeight: 'bolder',  
        fontSize: 20,  
        fontStyle: 'italic'  
      },  
      detail: {formatter: '{value} C'},  
      data: [{value: value, name: '온도'}]  
    }  
  ]  
};  
return msg;
```

게이지 범위 및 각도 조정

데이터 표시 형식



응용 예제

응용 예제

'country' 데이터에서 GNP 상위 10개 나라에 대한 데이터를 출력하는 프로그램

1. DB는 bigdb의 country 테이블을 활용
2. Node-Red로 GNP 상위 10개 나라를 쿼리하시오.
3. GNP데이터에 대해 바차트를 그릴 수 있도록 옵션을 만드는 API를 작성하시오
4. ajax를 통해 GNP 상위 10개 나라에 대한 데이터를 테이블로 도출하시오.
5. ajax를 통해 3번에서 작성한 API와 연결하여 바차트를 그리시오.



