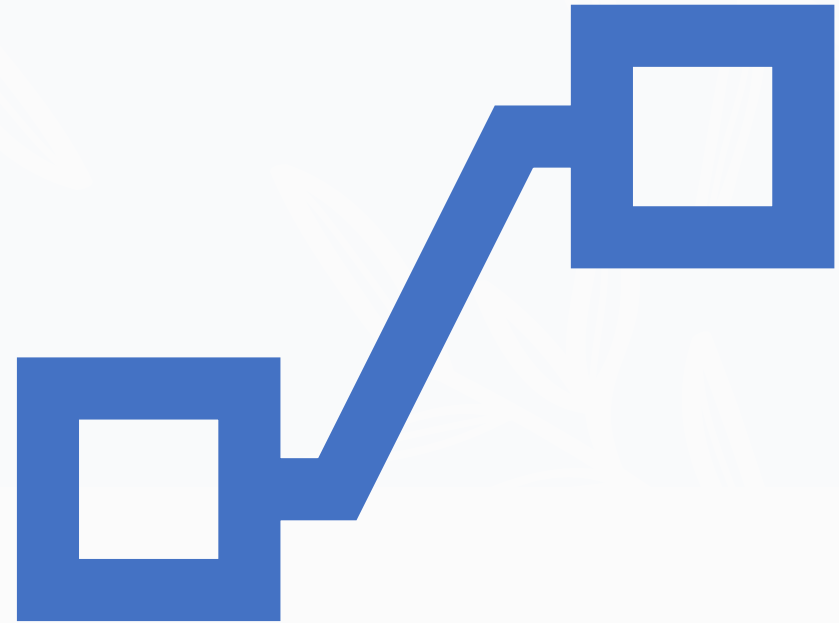


# 삼천리 공공데이터 API 연결 자료

2021.07

SmartMind, Inc.



# INDEX

---

---

한국은행 경제통계시스템 데이터 연결하기

---

DataFrame으로 변환

---

DataFrame TO CSV

---

DataFrame TO SQL Server

---

# 01 | 공공데이터 연결하기 - Azure VM

## 공공데이터 연결 과정 - Python 사용

```
<results>
  <currentCount>1</currentCount>
  <data>
    <item>
      <country_eng_nm>Australia</country_eng_nm>
      <country_iso_alp2>AU</country_iso_alp2>
      <country_nm>호주</country_nm>
      <ctypn_policy_cn/>
      <ecmy_growth_rate/>
      <export_amount/>
      <export_amount_src>('19) World Bank (최근 수정일 : 2021.02.17.)</export_amount_src>
      <ext_debt/>
      <foreign_currency_reserve/>
      <gdp>1,396,567,014,733</gdp>
      <gdp_per_capita>55,060</gdp_per_capita>
      <gdp_src>('19) World Bank (최근 수정일 : 2021.02.17.)</gdp_src>
      <income_amount/>
      <income_amount_src>('19) World Bank (최근 수정일 : 2021.02.17.)</income_amount_src>
      <infltn_rate/>
      <invst_sts_cn>對호주 투자 : 203억불, 對한국 투자 : 47억불</invst_sts_cn>
      <main_indust_cn/>
      <main_resource_cn>철광석, 석탄, LNG, 보크사이트, 우라늄, 원유, 양모, 밀, 쇠고기, 설탕</main_resource_cn>
      <oda_sts_cn/>
      <ptcl_state_cn/>
      <remark/>
      <trade_export_prdnm_cn/>
      <trade_income_prdnm_cn/>
      <trade_year/>
      <unemploy_rate/>
      <unemploy_rate_year/>
      <written_year/>
    </item>
  </data>
  <numOfRows>10</numOfRows>
  <pageNo>1</pageNo>
  <resultCode>0</resultCode>
  <resultMsg>정상</resultMsg>
  <totalCount>1</totalCount>
</results>
```

XML 형식 response

```
{
  "currentCount": 1,
  "data": [
    {
      "country_eng_nm": "Australia",
      "country_iso_alp2": "AU",
      "country_nm": "호주",
      "ctypn_policy_cn": null,
      "ecmy_growth_rate": 2.16,
      "export_amount": 1718341213,
      "export_amount_src": "('19) World Bank (최근 수정일 : 2021.02.17.)",
      "ext_debt": null,
      "foreign_currency_reserve": null,
      "gdp": 1396567014733,
      "gdp_per_capita": 55060,
      "gdp_src": "('19) World Bank (최근 수정일 : 2021.02.17.)",
      "income_amount": 1003698234,
      "income_amount_src": "('19) World Bank (최근 수정일 : 2021.02.17.)",
      "infltn_rate": 1.61,
      "invst_sts_cn": "對호주 투자 : 203억불, 對한국 투자 : 47억불",
      "main_indust_cn": null,
      "main_resource_cn": "철광석, 석탄, LNG, 보크사이트, 우라늄, 원유, 양모, 밀, 쇠고기, 설탕",
      "oda_sts_cn": "",
      "ptcl_state_cn": null,
      "remark": null,
      "trade_export_prdnm_cn": null,
      "trade_income_prdnm_cn": null,
      "trade_year": 2019,
      "unemploy_rate": null,
      "unemploy_rate_year": null,
      "written_year": 2020
    }
  ],
  "numOfRows": 10,
  "pageNo": 1,
  "resultCode": 0,
  "resultMsg": "정상",
  "totalCount": 1
}
```

JSON 형식 response

# 02| 공공데이터 연결

## 한국은행 경제통계시스템 – ecos.bor.or.kr

- 일부 경제 데이터는 한국은행 경제통계시스템에서 API를 통해 데이터를 가져올 수 있음
- 한국은행에서 발급받은 키를 사용하여 필요한 데이터를 url 형식으로 데이터를 요청
- 공공데이터 포털에서 ecos로 시작하는 Url를 제공하는 경우에도 사용



### 오픈API 상세



**XML JSON 한국은행\_물가** [바로가기](#)

생산자물가지수, 국내공급물가지수, 총산출물가지수 등 통계자료

[0](#) [0](#) [관심](#)

[오류신고 및 담당자 문의](#)

### OpenAPI 정보 메타데이터 다운로드

분류체계	재정·세제·금융 - 금융	제공기관	한국은행
관리부서명	커뮤니케이션국	관리부서 전화번호	02-759-5166
API 유형	LINK	데이터포맷	JSON+XML
활용신청	95	키워드	경제통계, 수입물가, 생산자물가
등록	2020-04-27	수정	2020-04-27
URL	<a href="http://ecos.bor.or.kr/jsp/openapi/OpenApiController.jsp?t=main">http://ecos.bor.or.kr/jsp/openapi/OpenApiController.jsp?t=main</a>		
비용부과유무	무료		
이용허락범위	<a href="#">이용허락범위 제한 없음</a>		
참고문서			

## 02| 공공데이터 연결

한국은행 경제통계시스템 – ecos.bor.or.kr

- 한국은행 경제통계시스템 API 서비스 접속 → <http://ecos.bok.or.kr/jsp/openapi/OpenApiController.jsp>
- 메인 메뉴에서 개발 가이드 → 통계코드검색 선택



한국은행 경제통계시스템 – [ecos.bor.or.kr](http://ecos.bor.or.kr)

- 통계코드검색

Figure 1 displays two side-by-side screenshots of a hierarchical tree structure, likely from a data management system, comparing the 'Domestic Product' (국내총생산) and 'Consumer Price' (소비자물가) categories.

The left pane, titled '동계표[코드][주기]' (Domestic Product [Code] [Period]), shows a tree structure under '7.4. 소비자물가지수 (2015=100)'. The item '7.4.1 소비자물가지수 (2015=100)(전국)' is highlighted with a red box, showing its code as '[021Y125][MM,QQ,YY]'. Below it, '7.4.2 소비자물가지수 (2015=100)(전국, 특수분류)' is also visible.

The right pane, titled '동계항목[코드][단위]' (Domestic Product Item [Code] [Unit]), shows a tree structure under '소비자물가지수 (2015=100)(전국)[코드]'. The item '도시가스' is highlighted with a red box, showing its code as '[05201][2015=100]'. Other items listed include '식료품 및 비주류 음료', '주류 및 담배', '의류 및 신발', '주택, 수도, 전기 및 연료', '주거시설 유지·보수', '수도 및 주거관련 서비스', '전기, 가스 및 기타 연료', '전기', '가스', '도시가스', '취사용 LPG', '기타연료 및 에너지', '가정용품 및 가사 서비스', '보건', '교통', '통신', '오락 및 문화', '교육', '음식 및 숙박', and '기타 상품 및 서비스'.

# 02| 공공데이터 연결

## 한국은행 경제통계시스템 – ecos.bor.or.kr

Click to add text

- 개발 명세서 탭으로 이동 후 서비스명을 통계 조회 조건으로 변경
- 통계 조회 조건에서 샘플 테스트를 활용해 데이터가 정상적으로 출력되는지 확인
- 데이터가 정상적으로 출력되면 결과창 상단의 url 복사

### OpenAPI 테스트

OpenAPI 서비스의 요청인자에 값을 입력하고 검색버튼을 클릭하여 해당 서비스의 xml 형태의 응답을 확인해 볼 수 있습니다.

서비스	통계 조회 조건 설정	검색 >
통계코드	서비스명(필수)	StatisticSearch
	인증키(필수)	[REDACTED]
	요청타입(필수)	json
	언어(필수)	kr
	요청시작건수(필수)	1
	요청종료건수(필수)	10
	통계표코드(필수)	021Y125
	주기(필수)	MM
	검색시작일자(필수)	20201001
	검색종료일자(필수)	20210601
	통계항목1코드(선택)	D05201
	통계항목2코드(선택)	
	통계항목3코드(선택)	
	* 통계코드 또는 항목코드는 [개발가이드 > 통계코드검색] 메뉴에서 확인가능합니다.	
<div>http://ecos.bok.or.kr/api/StatisticSearch/[REDACTED]/json/kr/1/10/021Y125/MM/20201001/20210601/D05201/입력/입력/</div> <div>{ "StatisticSearch": { "list_total_count": 7, "row": [ { "UNIT_NAME": "2015=100", "STAT_NAME": "7.4.1 소비자물가지수 (2015=100)" } ] }</div>		

# 02| 한국은행 경제통계시스템 연결하기

Ecos.bor.or.kr

- 복사한 URL을 활용하여 Python에서 출력

## JSON

```
: 1 from urllib.request import urlopen
  2 from urllib.parse import urlencode, unquote, quote_plus
  3 import urllib
  4 import requests
  5 import json
  6 import pandas as pd
  7 from bs4 import BeautifulSoup

: 1 url = "http://ecos.bok.or.kr/api/StatisticSearch/U8WQID0FBFC73"
  2
  3 req=urllib.request.Request(url)
  4 response_body=urlopen(req,timeout=60).read()
  5 data=json.loads(response_body)
  6 data
```

```
: {'StatisticSearch': {'list_total_count': 12,
  'row': [{ 'UNIT_NAME': '% ',
    'STAT_NAME': '18.1.4.1 경제성장률',
    'ITEM_CODE1': 'KOR',
    'STAT_CODE': 'I10Y041',
    'ITEM_CODE2': ' ',
    'ITEM_CODE3': ' ',
    'ITEM_NAME1': '한국',
    'ITEM_NAME2': ' ',
    'DATA_VALUE': '0.8',
    'ITEM_NAME3': ' ',
    'TIME': '2009'},
  { 'UNIT_NAME': '% ',
    'STAT_NAME': '18.1.4.1 경제성장률',
    'ITEM_CODE1': 'KOR',
    'STAT_CODE': 'I10Y041',
    'ITEM_CODE2': ' ',
    'ITEM_CODE3': ' ',
    'ITEM_NAME1': '한국',
    'ITEM_NAME2': ' '}]}}
```



## 02| 한국은행 경제통계시스템 연결하기

Ecos.bor.or.kr

- 복사한 URL을 활용하여 Python에서 출력

```
url = "http://ecos.bok.or.kr/api/StatisticSearch/U8WQID0FBFC73ZSD8RWP/json/kr/1/10/021Y125/MM/20201001/20210601/D05201/"

req=urllib.request.Request(url)
response_body=urlopen(req,timeout=60).read()
data=json.loads(response_body)
data
```

```
atisticSearch': {'list_total_count': 7,
'ow': [{'UNIT_NAME': '2015=100',
'STAT_NAME': '7.4.1 소비자물가지수(2015=100)(전국)',
'ITEM_CODE1': 'D05201',
'STAT_CODE': '021Y125',
'ITEM_CODE2': '',
'ITEM_CODE3': '',
'ITEM_NAME1': '도시가스',
'ITEM_NAME2': '',
'DATA_VALUE': '77.38',
'ITEM_NAME3': '',
'TIME': '202011'},
'UNIT_NAME': '2015=100',
```

# 03| 기타 API 연결

## 한국수출입은행 환율 정보

- URL : <https://www.koreaexim.go.kr/site/program/financial/exchangeJSON>
- Parameter
  - Authkey : 한국수출입 은행의 인증키입니다. 인증키를 발급받아서 사용하시면 됩니다.
  - searchData : 환율을 검색할 날짜입니다
  - Data : 검색할 API 타입입니다.

```
In [35]: 1 url = "https://www.koreaexim.go.kr/site/program/financial/exchangeJSON?authkey=MTEMVPo5Pkk5xQU8gszDBGcUJoTnjTEI&data=AP01"
          2
          3 req=urllib.request.Request(url)
          4 response_body=urlopen(req,timeout=60).read()
          5 data=json.loads(response_body)
          6 data
```

```
Out [35]: [{'result': 1,
            'cur_unit': 'AED',
            'ttb': '305.2',
            'tts': '311.37',
            'deal_bas_r': '308.29',
            'bkpr': '308',
            'yy_efee_r': '0',
            'ten_dd_efee_r': '0',
            'kftc_bkpr': '308',
            'kftc_deal_bas_r': '308.29',
            'cur_nm': '아랍에미리트 디르함'},
            {'result': 1,
            'cur_unit': 'AUD',
            'ttb': '846.8',
            'tts': '863.91',
            'deal_bas_r': '855.36',
            'bkpr': '855',
            'cur_nm': '호주 달러'}
```

# 03| 공공데이터 저장하기

## 공공데이터를 활용하기 위한 Dataframe

### 데이터프레임(DataFrame)

- Python에서 데이터를 저장하기 위한 행과 열이 있는 테이블
- API를 통해 받아온 데이터를 Dataframe에 저장한 후 이를 SQL Database나 CSV 파일로 변환하여 저장
- Python을 통해 데이터를 분석할 경우 Dataframe에 데이터를 불러온 후 사용

	UNIT_NAME	STAT_NAME	ITEM_CODE1	STAT_CODE	ITEM_CODE2	ITEM_CODE3	ITEM_NAME1	ITEM_NAME2	DATA_VALUE	ITEM_NAME3	TIME
0	%	18.1.4.1 경제성장률	KOR	I10Y041			한국		0.8		2009
0	%	18.1.4.1 경제성장률	KOR	I10Y041			한국		6.8		2010
0	%	18.1.4.1 경제성장률	KOR	I10Y041			한국		3.7		2011
0	%	18.1.4.1 경제성장률	KOR	I10Y041			한국		2.4		2012
0	%	18.1.4.1 경제성장률	KOR	I10Y041			한국		3.2		2013
0	%	18.1.4.1 경제성장률	KOR	I10Y041			한국		3.2		2014
0	%	18.1.4.1 경제성장률	KOR	I10Y041			한국		2.8		2015
0	%	18.1.4.1 경제성장률	KOR	I10Y041			한국		2.9		2016
0	%	18.1.4.1 경제성장률	KOR	I10Y041			한국		3.2		2017
0	%	18.1.4.1 경제성장률	KOR	I10Y041			한국		2.9		2018
0	%	18.1.4.1 경제성장률	KOR	I10Y041			한국		2		2019
0	%	18.1.4.1 경제성장률	KOR	I10Y041			한국		-1		2020

# 03| 공공데이터 저장하기

## JSON to DataFrame

---

### JSON 데이터 결합하기

- Pandas = Dataframe을 관리하는 python module
- Pandas의 json\_normalize()를 활용해 json 데이터를 Dataframe으로 변환할 수 있음

# 03| 공공데이터 저장하기

## JSON to DataFrame

Ex ) 국내외\_경제성장률  
JSON을 **Dataframe**으로 변환

```
1 result = pd.DataFrame()
2 for row in data['StatisticSearch']['row'] :
3     if len(result) == 0 :
4         result = pd.json_normalize(row)
5     else :
6         sample = pd.json_normalize(row)
7         result = result.append(sample)
8 result
```

JSON

DataFrame

	UNIT_NAME	STAT_NAME	ITEM_CODE1	STAT_CODE	ITEM_CODE2	ITEM_CODE3
0	%	18.1.4.1 경제성장률	KOR	I10Y041		
0	%	18.1.4.1 경제성장률	KOR	I10Y041		
0	%	18.1.4.1 경제성장률	KOR	I10Y041		
0	%	18.1.4.1 경제성장률	KOR	I10Y041		
0	%	18.1.4.1 경제성장률	KOR	I10Y041		
0	%	18.1.4.1 경제성장률	KOR	I10Y041		
0	%	18.1.4.1 경제성장률	KOR	I10Y041		

# 03| 공공데이터 저장하기

## JSON to DataFrame

### JSON 데이터 결합하기

- Pandas = Dataframe을 관리하는 python module
- Pandas의 json\_normalize()를 활용해 json 데이터를 Dataframe으로 변환할 수 있음

```
{'StatisticSearch': {'list_total_count': 660,  
  'row': [{'UNIT_NAME': '',  
    'STAT_NAME': '9.1.1.1 전국실적',  
    'ITEM_CODE1': '99988',  
    'STAT_CODE': '041Y013',  
    'ITEM_CODE2': 'AA',  
    'ITEM_CODE3': '',  
    'ITEM_NAME1': '전 산 업',  
    'ITEM_NAME2': '업황실적1)',  
    'DATA_VALUE': '89',  
    'ITEM_NAME3': '',  
    'TIME': '201001'},  
    {'UNIT_NAME': '',  
      'STAT_NAME': '9.1.1.1 전국실적',  
      'ITEM_CODE1': '99988',  
      'STAT_CODE': '041Y013',  
      'ITEM_CODE2': 'AA',  
      'ITEM_CODE3': '',  
      'ITEM_NAME1': '전 산 업',  
      'ITEM_NAME2': '업황실적1)'}]}
```



```
1 result = pd.DataFrame()  
2 for row in data['StatisticSearch']['row'] :  
3     if len(result) == 0 :  
4         result = pd.json_normalize(row)  
5     else :  
6         sample = pd.json_normalize(row)  
7         result = result.append(sample)  
8 result
```

	UNIT_NAME	STAT_NAME	ITEM_CODE1	STAT_CODE	ITEM_CODE2	ITEM_CODE3
0		9.1.1.1 전국실적	99988	041Y013	AA	
0		9.1.1.1 전국실적	99988	041Y013	AA	
0		9.1.1.1 전국실적	99988	041Y013	AA	
0		9.1.1.1 전국실적	99988	041Y013	AA	
0		9.1.1.1 전국실적	99988	041Y013	AA	
...	...	...	...	...	...	...
0		9.1.1.1 전국실적	99988	041Y013	AJ	

# 03| 공공데이터 저장하기

## JSON to DataFrame

### JSON 데이터 결합하기

예시 ) 주가지수\_코스닥 평균 데이터 JSON List를 반복문을 활용하여 **Dataframe**으로 변환  
이 경우 필요한 데이터가 **StatisticSearch** → **row**에 존재하므로 반복문도 해당 부분을 불러와서 변환

```
{'StatisticSearch': {'list_total_count': 12,
  'row': [{'UNIT_NAME': '1996.07.01=1000',
    'STAT_NAME': '6.1.2 주식거래 및 주가지수',
    'ITEM_CODE1': '2100000',
    'STAT_CODE': '028Y015',
    'ITEM_CODE2': '',
    'ITEM_CODE3': '',
    'ITEM_NAME1': 'KOSDAQ_평균',
    'ITEM_NAME2': '',
    'DATA_VALUE': '473.64',
    'ITEM_NAME3': '',
    'TIME': '2009'}],
  {'UNIT_NAME': '1996.07.01=1000',
    'STAT_NAME': '6.1.2 주식거래 및 주가지수',
    'ITEM_CODE1': '2100000',
    'STAT_CODE': '028Y015',
    'ITEM_CODE2': '',
    'ITEM_CODE3': '',
    'ITEM_NAME1': 'KOSDAQ_평균',
    'ITEM_NAME2': ''}]}
```



```
1 result = pd.DataFrame()
2 for row in data['StatisticSearch']['row'] :
3     if len(result) == 0 :
4         result = pd.json_normalize(row)
5     else :
6         sample = pd.json_normalize(row)
7         result = result.append(sample)
8 result
```

	UNIT_NAME	STAT_NAME	ITEM_CODE1	STAT_CODE	ITEM_C
0	1996.07.01=1000	6.1.2 주식거래 및 주가지수	2100000	028Y015	
0	1996.07.01=1000	6.1.2 주식거래 및 주가지수	2100000	028Y015	
0	1996.07.01=1000	6.1.2 주식거래 및 주가지수	2100000	028Y015	
0	1996.07.01=1000	6.1.2 주식거래 및 주가지수	2100000	028Y015	
0	1996.07.01=1000	6.1.2 주식거래 및 주가지수	2100000	028Y015	

# 03| 공공데이터 저장하기

## XML to DataFrame

---

### XML 데이터 결합하기

- XML : HTML과 비슷한 문자 기반의 마크업 언어
- Python module의 BeautifulSoup module을 사용해 xml 데이터를 변환한 후 해당 요소를 찾아서 변환



# 03| 공공데이터 저장하기

## XML to DataFrame

Find\_all 사용

예시) 한국 수출입은행 환율정보

```
1 #url for request
2 today = datetime.now().strftime("%Y%m%d")
3
4 url='http://ecos.bok.or.kr/api/StatisticSearch/640DY0DPPCJMGDKX4LNQ/;
5
6 req=requests.get(url)
7
8 soup = BeautifulSoup(req.text, 'html.parser')
9 items = soup.find_all("row")
```

```
1 items
```

```
[<row>
<stat_code>036Y001</stat_code>
<stat_name>8.8.1.1 주요국통화의 대원화 환율</stat_name>
<item_code1>0000001</item_code1>
<item_name1>원/미국달러(매매기준율)</item_name1>
<item_code2> </item_code2>
<item_name2> </item_name2>
<item_code3> </item_code3>
<item_name3> </item_name3>
<unit_name>원 </unit_name>
<time>20210623</time>
<data_value>1132.4</data_value>
</row>]
```

- html을 파이썬에서 읽을 수 있게 파싱합니다. 즉, 파이썬 객체로 변환하는 것입니다.
- html이라는 변수에 저장한 html 소스코드를 .parser를 붙여 변환해줍니다.
- parser는 파이썬의 내장 메소드입니다.
- html.parser (기본 파서, 적당하게 빠른 수준)

XML 데이터 중에 "row"에 지정된 요소 불러옵니다.



# 03| 공공데이터 저장하기

## XML to DataFrame

### Find 사용

예시) 한국 수출입은행 환율정보

```
[<row>
  <stat_code>036Y001</stat_code>
  <stat_name>8.8.1.1 주요국통화의 대원화 환율</stat_name>
  <item_code1>0000001</item_code1>
  <item_name1>원/미국달러(매매기준율)</item_name1>
  <item_code2> </item_code2>
  <item_name2> </item_name2>
  <item_code3> </item_code3>
  <item_name3> </item_name3>
  <unit_name>원 </unit_name>
  <time>20210623</time>
  <data_value>1132.4</data_value>
</row>]
```

Find(): 조건에 해당하는 첫 번째 정보만 보여줍니다.

```
1 ex = []
2 for i in items:
3     ex.append(i.find("data_value").text)
```

```
1 ex
```

```
['1132.4']
```

# 03| 공공데이터 저장하기

## XML to DataFrame

### Select

웹 페이지 요소 중에서 특정 태그의 내용을 전부 선택  
여러 요소가 존재할 경우 List에 내용 저장

```
522.09
</td>

</tr>
</tbody>
<tbody id="tbody2">
<tr>
<td class="nobd_l" scope="col">21년06월24일</td>
<td class="v_hi v_row" style="text-align:right">

73.73
</td>

<td class="v_hi v_row" style="text-align:right">

75.56
</td>

<td class="v_hi v_row" style="text-align:right">

73.30
</td>

</tr>
</tbody>
</table>
```



```
1 tableList = soup.select('#tbody2 > tr')
2 tableList
```

```
<tr>
<td class="nobd_l" scope="col">21년06월24일</td>
<td class="v_hi v_row" style="text-align:right">

73.73
</td>

<td class="v_hi v_row" style="text-align:right">

75.56
</td>

<td class="v_hi v_row" style="text-align:right">

73.30
</td>
```

# 03| 공공데이터 저장하기

## XML to DataFrame

예시) 국가별 지역 경제정보 XML

```
<StatisticSearch>
  <list_total_count>12</list_total_count>
  <row>
    <STAT_CODE>028Y015</STAT_CODE>
    <STAT_NAME>6.1.2 주식거래 및 주가지수</STAT_NAME>
    <ITEM_CODE1>1080000</ITEM_CODE1>
    <ITEM_NAME1>KOSPI_평균</ITEM_NAME1>
    <ITEM_CODE2> </ITEM_CODE2>
    <ITEM_NAME2> </ITEM_NAME2>
    <ITEM_CODE3> </ITEM_CODE3>
    <ITEM_NAME3> </ITEM_NAME3>
    <UNIT_NAME>1980.01.04=100 </UNIT_NAME>
    <TIME>2009</TIME>
    <DATA_VALUE>1429.04</DATA_VALUE>
  </row>
  <row>
    <STAT_CODE>028Y015</STAT_CODE>
    <STAT_NAME>6.1.2 주식거래 및 주가지수</STAT_NAME>
    <ITEM_CODE1>1080000</ITEM_CODE1>
    <ITEM_NAME1>KOSPI_평균</ITEM_NAME1>
    <ITEM_CODE2> </ITEM_CODE2>
    <ITEM_NAME2> </ITEM_NAME2>
    <ITEM_CODE3> </ITEM_CODE3>
    <ITEM_NAME3> </ITEM_NAME3>
    <UNIT_NAME>1980.01.04=100 </UNIT_NAME>
    <TIME>2010</TIME>
    <DATA_VALUE>1764.99</DATA_VALUE>
  </row>
</row>
```

### 주식거래 및 주가지수 XML

#### KOSPI 평균\_XML ¶

```
1 import requests
2 from bs4 import BeautifulSoup
3 import pandas as pd
4 import csv
5 import time
```

모듈 import

```
1 url = 'http://ecos.bok.or.kr/api/StatisticSearch/U8WQID0FBFC73ZSD8'
2 print(url)
```

<http://ecos.bok.or.kr/api/StatisticSearch/U8WQID0FBFC73ZSD8>

```
1 html = requests.get(url)
2 txt = html.text
3 soup = BeautifulSoup(txt, 'lxml-xml') #xml로 파싱
4 items = soup.select('row')
5
```



# 03| 공공데이터 저장하기

## XML to DataFrame


For문으로 컬럼들을 불러옵니다.

```
test = []
for item in items:
    item_list = []
    item_list.append(item.select_one('ITEM_CODE1').text)
    item_list.append(item.select_one('ITEM_NAME1').text)
    item_list.append(item.select_one('ITEM_CODE2').text)
    item_list.append(item.select_one('ITEM_NAME2').text)
    item_list.append(item.select_one('ITEM_CODE3').text)
    item_list.append(item.select_one('ITEM_NAME3').text)
    item_list.append(item.select_one('UNIT_NAME').text)
    item_list.append(item.select_one('TIME').text)
    item_list.append(item.select_one('DATA_VALUE').text)

    test.append(item_list)

print(test)
```

국내외 경제성장률 XML



```
[['KOR', '한국', '', '', '', '', '%', '2009', '0.8'], ['KOR', '한국', '', '', '', '', '%', '2011', '3.7'], ['KOR', '한국', '', '', '', '', '%', '2012', '3.2'], ['KOR', '한국', '', '', '', '', '%', '2013', '3.2'], ['KOR', '한국', '', '', '', '', '%', '2014', '3.2'], ['KOR', '한국', '', '', '', '', '%', '2015', '2.9'], ['KOR', '한국', '', '', '', '', '%', '2016', '2.9'], ['KOR', '한국', '', '', '', '', '%', '2017', '2.9'], ['KOR', '한국', '', '', '', '', '%', '2018', '2.9'], ['KOR', '한국', '', '', '', '', '%', '2019', '2.9'], ['KOR', '한국', '', '', '', '', '%', '2020', '-1']]
```

# 03| 공공데이터 저장하기

## XML to DataFrame

국가별지역 경제정보

```
1 df = pd.DataFrame(test, columns=["ITEM_CODE1",  
2 "ITEM_NAME1",  
3 "ITEM_CODE2",  
4 "ITEM_NAME2",  
5 "ITEM_CODE3",  
6 "ITEM_NAME3",  
7 "UNIT_NAME",  
8 "TIME",  
9 "DATA_VALUE"  
10  
11 ])  
12  
13 df
```

XML to DataFrame

	ITEM_CODE1	ITEM_NAME1	ITEM_CODE2	ITEM_NAME2	ITEM_CODE3	ITEM_NAME3	UNIT_NAME	TIME	DATA_VALUE
0	KOR	한국					%	2011	3.7
1	KOR	한국					%	2012	2.4
2	KOR	한국					%	2013	3.2
3	KOR	한국					%	2014	3.2
4	KOR	한국					%	2015	2.8
5	KOR	한국					%	2016	2.9
6	KOR	한국					%	2017	3.2
7	KOR	한국					%	2018	2.9
8	KOR	한국					%	2019	2

# XML to DataFrame

### 예시)기업경기실사지수 1

## XML

```
</row>,  
<row>  
<STAT_CODE>041Y013</STAT_CODE>  
<STAT_NAME>9.1.1.1 전국실적</STAT_NAME>  
<ITEM_CODE1>99988</ITEM_CODE1>  
<ITEM_NAME1>전 산 업</ITEM_NAME1>  
<ITEM_CODE2>AA</ITEM_CODE2>  
<ITEM_NAME2>업황실적1</ITEM_NAME2>  
<ITEM_CODE3> </ITEM_CODE3>  
<ITEM_NAME3> </ITEM_NAME3>  
<UNIT_NAME/>  
<TIME>201011</TIME>  
<DATA_VALUE>90</DATA_VALUE>  
</row>.
```

# 03| 공공데이터 저장하기

## XML to DataFrame

예시) 기업경기실사지수 2

```
1 test = []
2 for item in items:
3     item_list = []
4     item_list.append(item.select_one('STAT_NAME').text)
5     item_list.append(item.select_one('ITEM_CODE1').text)
6     item_list.append(item.select_one('ITEM_NAME1').text)
7     item_list.append(item.select_one('ITEM_CODE2').text)
8     item_list.append(item.select_one('ITEM_NAME2').text)
9     item_list.append(item.select_one('ITEM_CODE3').text)
10    item_list.append(item.select_one('ITEM_NAME3').text)
11    item_list.append(item.select_one('UNIT_NAME').text)
12    item_list.append(item.select_one('TIME').text)
13    item_list.append(item.select_one('DATA_VALUE').text)
14
15    test.append(item_list)
16
17 print(test)
```

```
국실적', '99988', '전 산 업', 'AA', '업황실적1)', '', '', '201011', '90'], ['9.1.1.1 전국실적', '99988', '전 산 업', 'AA', '업황
실적1)', '', '', '201012', '90'], ['9.1.1.1 전국실적', '99988', '전 산 업', 'AA', '업황실적1)', '', '', '201101', '87'],
['9.1.1.1 전국실적', '99988', '전 산 업', 'AA', '업황실적1)', '', '', '201102', '84'], ['9.1.1.1 전국실적', '99988', '전 산 업',
'AA', '업황실적1)', '', '', '201103', '87'], ['9.1.1.1 전국실적', '99988', '전 산 업', 'AA', '업황실적1)', '', '', '20110
4', '92'], ['9.1.1.1 전국실적', '99988', '전 산 업', 'AA', '업황실적1)', '', '', '201105', '90'], ['9.1.1.1 전국실적', '99988',
'전 산 업', 'AA', '업황실적1)', '', '', '201106', '88'], ['9.1.1.1 전국실적', '99988', '전 산 업', 'AA', '업황실적1)',
'', '', '201107', '87'], ['9.1.1.1 전국실적', '99988', '전 산 업', 'AA', '업황실적1)', '', '', '201108', '81'], ['9.1.1.1 전국실적',
'99988', '전 산 업', 'AA', '업황실적1)', '', '', '201109', '82'], ['9.1.1.1 전국실적', '99988', '전 산 업', 'AA', '업황실적1)',
'', '', '201110', '82'], ['9.1.1.1 전국실적', '99988', '전 산 업', 'AA', '업황실적1)', '', '', '201111', '80'], ['9.1.1.1 전
국실적', '99988', '전 산 업', 'AA', '업황실적1)', '', '', '201112', '81'], ['9.1.1.1 전국실적', '99988', '전 산 업', 'AA', '업황
실적1)', '', '', '201201', '78'], ['9.1.1.1 전국실적', '99988', '전 산 업', 'AA', '업황실적1)', '', '', '201202', '79'],
['9.1.1.1 전국실적', '99988', '전 산 업', 'AA', '업황실적1)', '', '', '201203', '81'], ['9.1.1.1 전국실적', '99988', '전 산 업',
'AA', '업황실적1)', '', '', '201204', '82'], ['9.1.1.1 전국실적', '99988', '전 산 업', 'AA', '업황실적1)', '', '', '20120
5', '82'], ['9.1.1.1 전국실적', '99988', '전 산 업', 'AA', '업황실적1)', '', '', '201206', '78'], ['9.1.1.1 전국실적', '99988',
'전 산 업', 'AA', '업황실적1)', '', '', '201207', '69'], ['9.1.1.1 전국실적', '99988', '전 산 업', 'AA', '업황실적1)',
'', '', '201208', '69'], ['9.1.1.1 전국실적', '99988', '전 산 업', 'AA', '업황실적1)', '', '', '201209', '68'], ['9.1.1.1 전국실적',
'99988', '전 산 업', 'AA', '업황실적1)', '', '', '201210', '68'], ['9.1.1.1 전국실적', '99988', '전 산 업', 'AA', '업황실적1)']
```



# 03| 공공데이터 저장하기

## XML to DataFrame

예시) 기업경기실사지수 3

```
1 df = pd.DataFrame(test, columns=["STAT_CODE",  
2 "ITEM_CODE1",  
3 "ITEM_NAME1",  
4 "ITEM_CODE2",  
5 "ITEM_NAME2",  
6 "ITEM_CODE3",  
7 "ITEM_NAME3",  
8 "UNIT_NAME",  
9 "TIME",  
10 "DATA_VALUE"]  
11 )  
12  
13  
14 df
```

	STAT_CODE	ITEM_CODE1	ITEM_NAME1	ITEM_CODE2	ITEM_NAME2	ITEM_CODE3	ITEM_NAME3	UNIT_NAME	TIME	DATA_VALUE
0	9.1.1.1 전국실적	99988	전 산 업	AA	업황실적1)				201001	89
1	9.1.1.1 전국실적	99988	전 산 업	AA	업황실적1)				201002	89
2	9.1.1.1 전국실적	99988	전 산 업	AA	업황실적1)				201003	89
3	9.1.1.1 전국실적	99988	전 산 업	AA	업황실적1)				201004	94
4	9.1.1.1 전국실적	99988	전 산 업	AA	업황실적1)				201005	95
...	...	...	...	...	...	...	...	...	...	...
655	9.1.1.1 전국실적	99988	전 산 업	AJ	인력사정실적3)				202008	91
656	9.1.1.1 전국실적	99988	전 산 업	AJ	인력사정실적3)				202009	91
657	9.1.1.1 전국실적	99988	전 산 업	AJ	인력사정실적3)				202010	89
658	9.1.1.1 전국실적	99988	전 산 업	AJ	인력사정실적3)				202011	91
659	9.1.1.1 전국실적	99988	전 산 업	AJ	인력사정실적3)				202012	90

660 rows × 10 columns

# 03| 공공데이터 저장하기

## DataFrame to CSV

Pandas 패키지의 `to_csv()`는 Pandas DataFrame(데이터프레임)을 .csv 확장자 파일로 저장해 주는 함수입니다.

다음과 같은 형태로 `to_csv()`를 사용합니다.

```
데이터프레임A.to_csv('저장할_파일이름.csv')
```

기본저장 위치는 작업하는 곳과 동일한 위치입니다.



### 국가지역별\_경제정보 데이터 csv 저장 및 불러오기

```
1 data.to_csv('지역별국가_경제정보.csv')
```

```
1 df = pd.read_csv('지역별국가_경제정보.csv')
2 df
```

Unnamed: 0	country_eng_nm	country_iso_alp2	country_nm	ctypln_policy_cn	ecomy_growth_rate	export_amount	export_amount_src	ext_debt	fore
0	0	Australia	AU	호주	NaN	2.16	1718341213	(19) World Bank (최근 수정일: 2021.02.17.)	NaN

1 rows x 28 columns



# 03| 공공데이터 저장하기

## DataFrame to CSV

예시) 기업경기실사지수 데이터 csv 저장 및 불러오기

```
In [51]: 1 df.to_csv('companyBSI.csv')
```

```
In [53]: 1 data = pd.read_csv('companyBSI.csv')
2 data.drop('Unnamed: 0', axis = 1, inplace = True)
3 data
```

Out [53]:

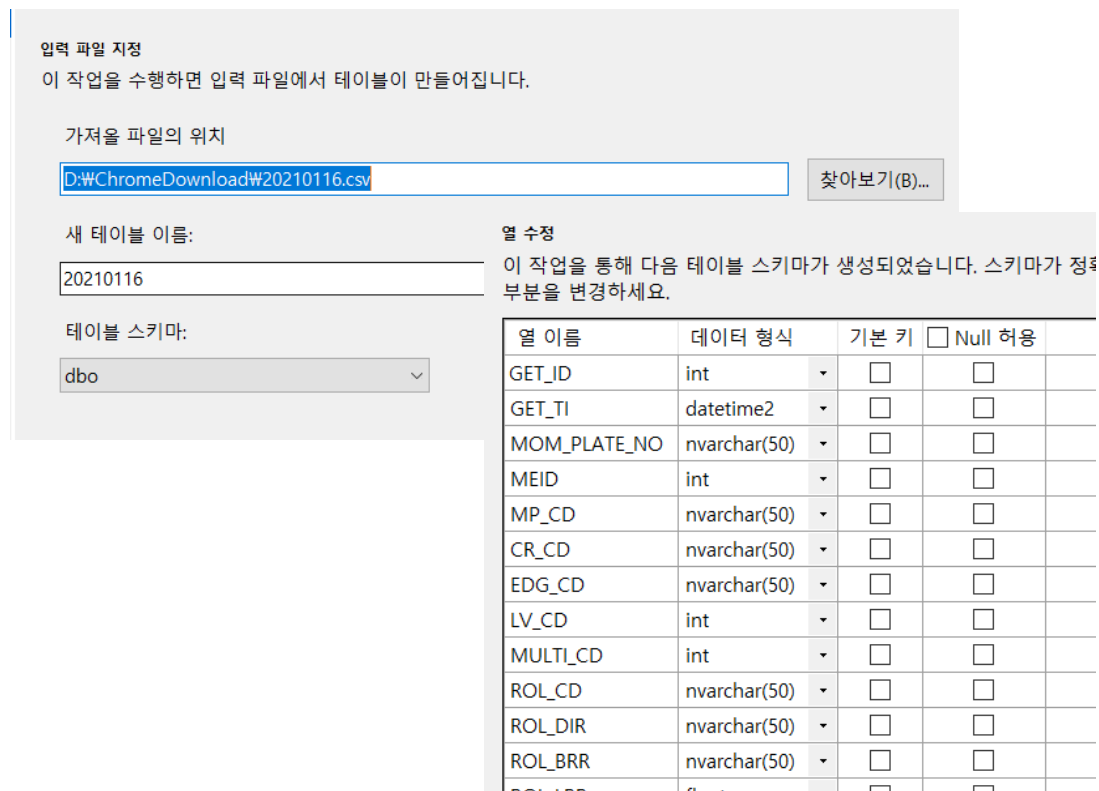
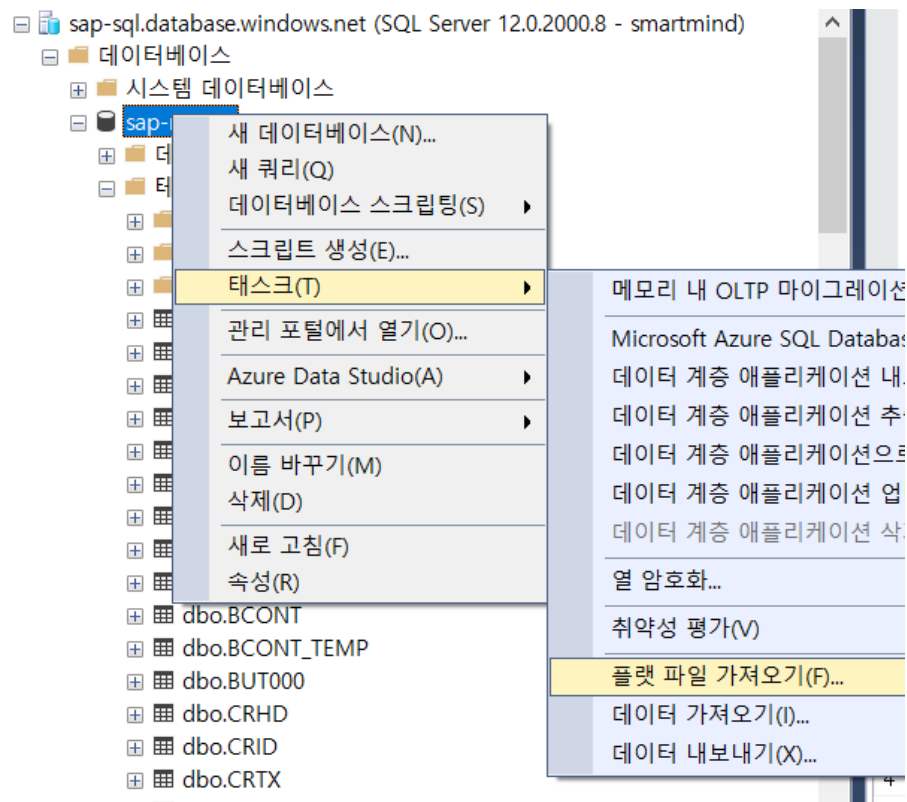
	STAT_CODE	ITEM_CODE1	ITEM_NAME1	ITEM_CODE2	ITEM_NAME2	ITEM_CODE3	ITEM_NAME3	UNIT_NAME	TIME	DATA_VALUE
0	9.1.1.1 전국실적	99988	전 산 업	AA	업황실적1)			NaN	201001	89
1	9.1.1.1 전국실적	99988	전 산 업	AA	업황실적1)			NaN	201002	89
2	9.1.1.1 전국실적	99988	전 산 업	AA	업황실적1)			NaN	201003	89
3	9.1.1.1 전국실적	99988	전 산 업	AA	업황실적1)			NaN	201004	94
4	9.1.1.1 전국실적	99988	전 산 업	AA	업황실적1)			NaN	201005	95
...	...	...	...	...	...	...	...	...	...	...
655	9.1.1.1 전국실적	99988	전 산 업	AJ	인력사정실적3)			NaN	202008	91
656	9.1.1.1 전국실적	99988	전 산 업	AJ	인력사정실적3)			NaN	202009	91
657	9.1.1.1 전국실적	99988	전 산 업	AJ	인력사정실적3)			NaN	202010	89
658	9.1.1.1 전국실적	99988	전 산 업	AJ	인력사정실적3)			NaN	202011	91
659	9.1.1.1 전국실적	99988	전 산 업	AJ	인력사정실적3)			NaN	202012	90

660 rows × 10 columns

# 03| 공공데이터 저장하기

## CSV to SQL

생성된 CSV 파일은 SSMS를 사용하여 DB에 테이블 형식으로 Import 가능  
DB 우클릭 → 태스크 → 플랫폼 파일 가져오기 사용



# 03| 공공데이터 저장하기

## DataFrame to SQL-Server

---

- **DataFrame**을 SQL Server 내 Table에 Import
- 주기적으로 데이터를 저장할 필요가 있을 때 사용
- 이미 생성된 테이블에 데이터를 insert하는 방식으로 진행

# 03| 공공데이터 저장하기

## DataFrame to SQL-Server

데이터베이스 정보 및 모듈 import

### Database

```
1 from datetime import datetime
2 import requests
3 import json
4 import pandas as pd
5 import pyodbc
6 from bs4 import BeautifulSoup
```

```
1 #DB 접속 정보 -> 붙여넣으시면 됩니다.
2 #####
3 import pyodbc
4 server = 'sap-sql.database.windows.net'
5 database = 'sap-master'
6 username = 'smartmind'
7 password = 'qaz!$Xedc!@#'
8 driver = '{ODBC Driver 17 for SQL Server}'
9 cnxn = pyodbc.connect('DRIVER='+driver+';SERVER='+server+';PORT=1433;DATABASE='+database+';UID='+username+';PWD='+ password)
10 cnxn.autocommit = True
11 cursor = cnxn.cursor()
12 cursor.execute("set language Korean")
13 #####
```



# 03| 공공데이터 저장하기

## DataFrame to SQL-Server

예시) 소비자물가지수

데이터베이스 저장 코드

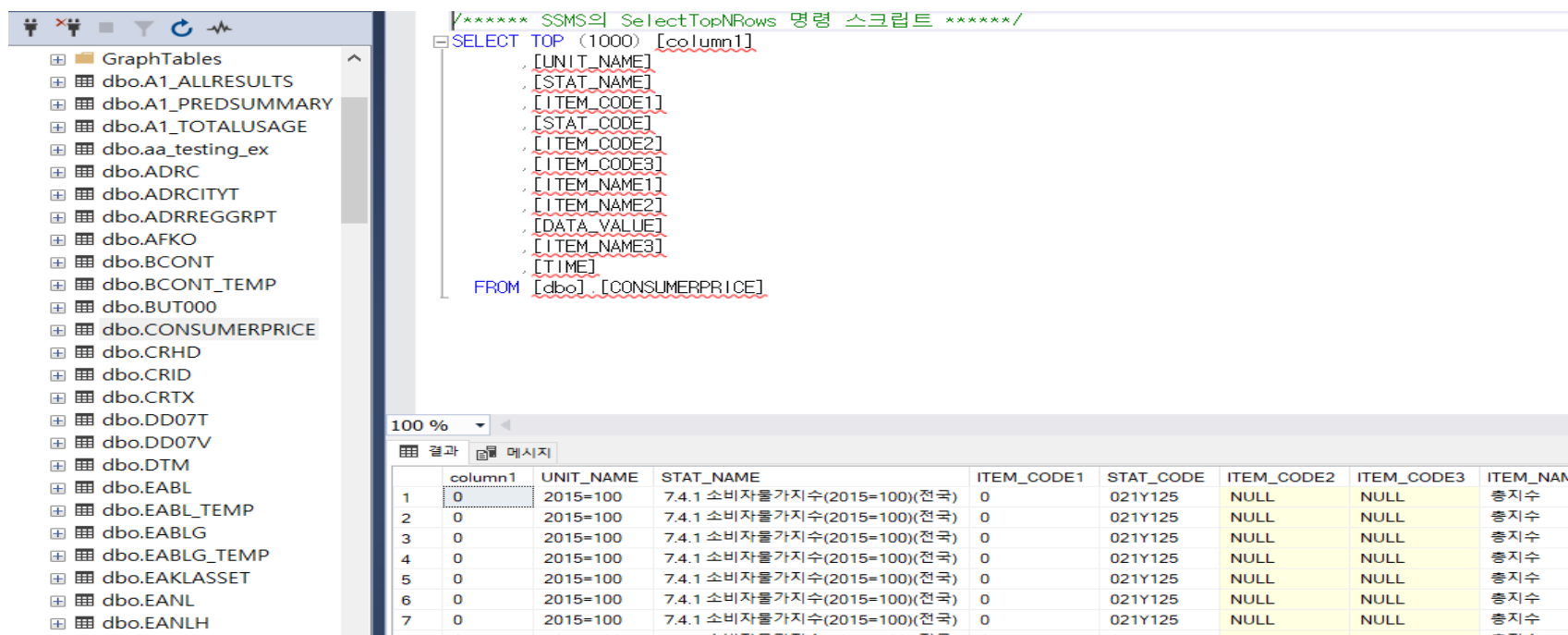
```
1 def insertData(df, cursor, cnxn) :  
2     # Insert Dataframe into SQL Server:  
3     for index, row in df.iterrows():  
4         cursor.execute("INSERT INTO CONSUMERPRICE (UNIT_NAME, STAT_NAME, ITEM_CODE1, STAT_CODE, ITEM_CODE2, ITEM_CODE3, ITEM_NAME1, ITEM_NAME2) VALUES (%s, %s, %s, %s, %s, %s, %s, %s)"  
5             cnxn.commit()
```

# 03| 공공데이터 저장하기

## DataFrame to SQL-Server

MSSQL Server Managment Studio에서 확인 및 테이블결과 조회

예시) 소비자물가지수



The screenshot displays the Microsoft SQL Server Enterprise Manager interface. On the left, a tree view shows the database structure, including tables like `dbo.A1_ALLRESULTS`, `dbo.A1_PREDSUMMARY`, and `dbo.CONSUMERPRICE`. The main pane shows a SQL query in the 'Query' tab:

```
SELECT TOP (1000) [column1],
[UNIT_NAME],
[STAT_NAME],
[ITEM_CODE1],
[STAT_CODE],
[ITEM_CODE2],
[ITEM_CODE3],
[ITEM_NAME1],
[ITEM_NAME2],
[DATA_VALUE],
[ITEM_NAME3],
[TIME]
FROM [dbo].[CONSUMERPRICE]
```

Below the query, the 'Results' tab shows the first 10 rows of the query results. The columns are: `column1`, `UNIT_NAME`, `STAT_NAME`, `ITEM_CODE1`, `STAT_CODE`, `ITEM_CODE2`, `ITEM_CODE3`, and `ITEM_NAM`.

	column1	UNIT_NAME	STAT_NAME	ITEM_CODE1	STAT_CODE	ITEM_CODE2	ITEM_CODE3	ITEM_NAM
1	0	2015=100	7.4.1 소비자물가지수(2015=100)(전국)	0	021Y125	NULL	NULL	총지수
2	0	2015=100	7.4.1 소비자물가지수(2015=100)(전국)	0	021Y125	NULL	NULL	총지수
3	0	2015=100	7.4.1 소비자물가지수(2015=100)(전국)	0	021Y125	NULL	NULL	총지수
4	0	2015=100	7.4.1 소비자물가지수(2015=100)(전국)	0	021Y125	NULL	NULL	총지수
5	0	2015=100	7.4.1 소비자물가지수(2015=100)(전국)	0	021Y125	NULL	NULL	총지수
6	0	2015=100	7.4.1 소비자물가지수(2015=100)(전국)	0	021Y125	NULL	NULL	총지수
7	0	2015=100	7.4.1 소비자물가지수(2015=100)(전국)	0	021Y125	NULL	NULL	총지수



# Thank you for watching

---