

데이터베이스와 SQL

10장

Python과 MySQL 연동

빅데이터 분석가 과정

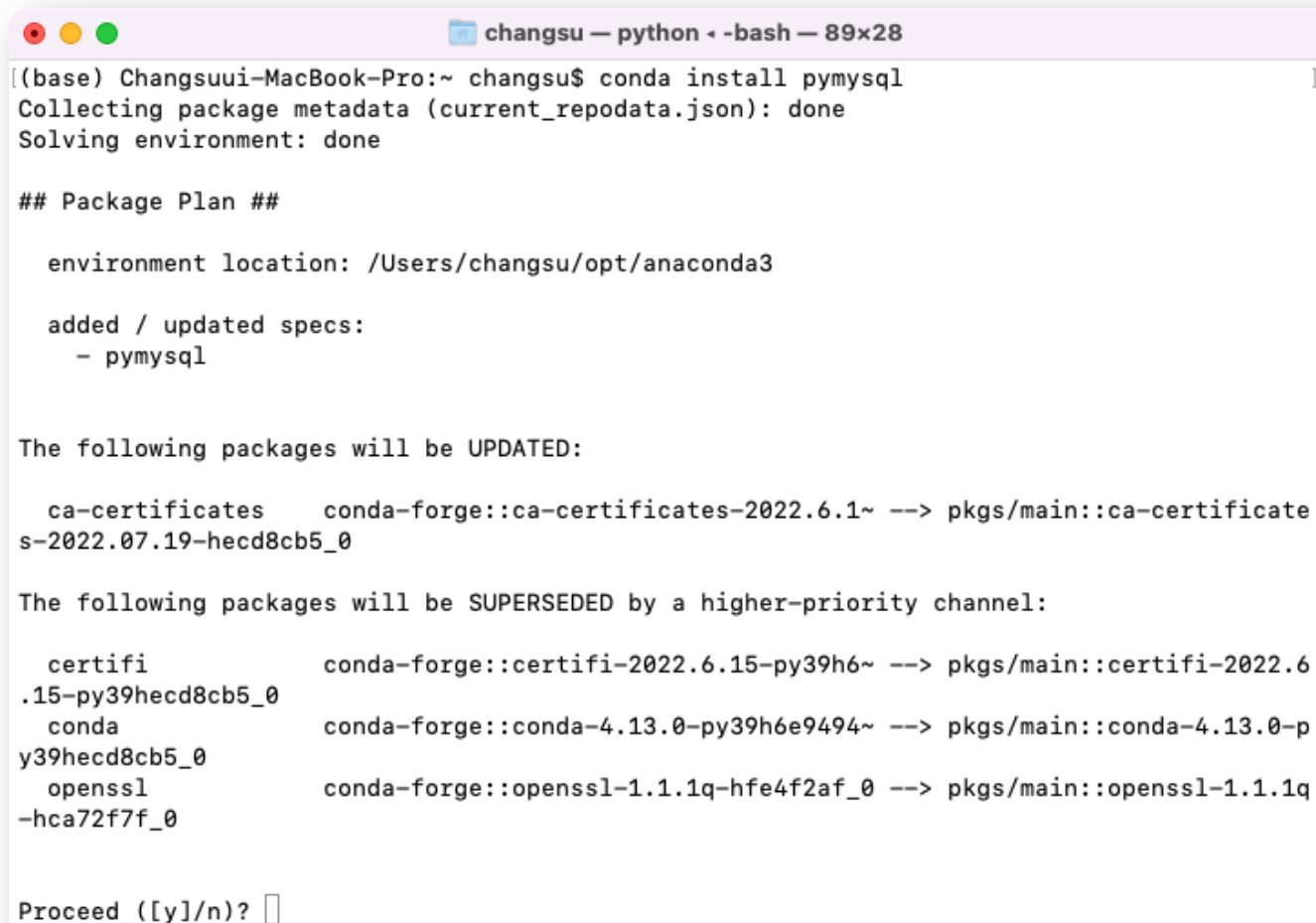
목차

- 라이브러리 설치
- MySQL 연동

MySQL과 Python 연동하기

■ PyMySQL 라이브러리 설치

```
$ conda install pymysql
```



```
changsu — python — bash — 89x28
[(base) Changsuui-MacBook-Pro:~ changsu$ conda install pymysql
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /Users/changsu/opt/anaconda3

  added / updated specs:
    - pymysql

The following packages will be UPDATED:

  ca-certificates    conda-forge::ca-certificates-2022.6.1~ --> pkgs/main::ca-certificates-2022.07.19-hecd8cb5_0

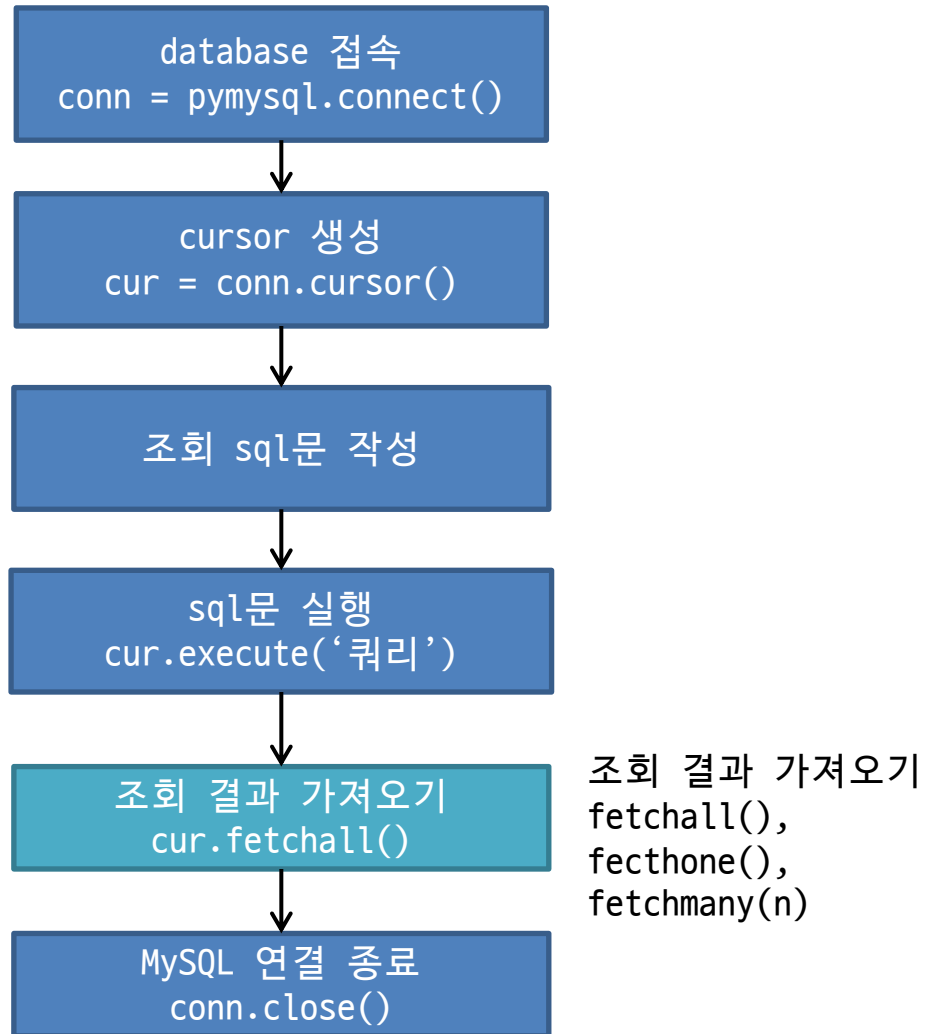
The following packages will be SUPERSEDED by a higher-priority channel:

  certifi            conda-forge::certifi-2022.6.15-py39h6~ --> pkgs/main::certifi-2022.6.15-py39hecd8cb5_0
  conda               conda-forge::conda-4.13.0-py39h6e9494~ --> pkgs/main::conda-4.13.0-py39hecd8cb5_0
  openssl            conda-forge::openssl-1.1.1q-hfe4f2af_0 --> pkgs/main::openssl-1.1.1q-hca72f7f_0

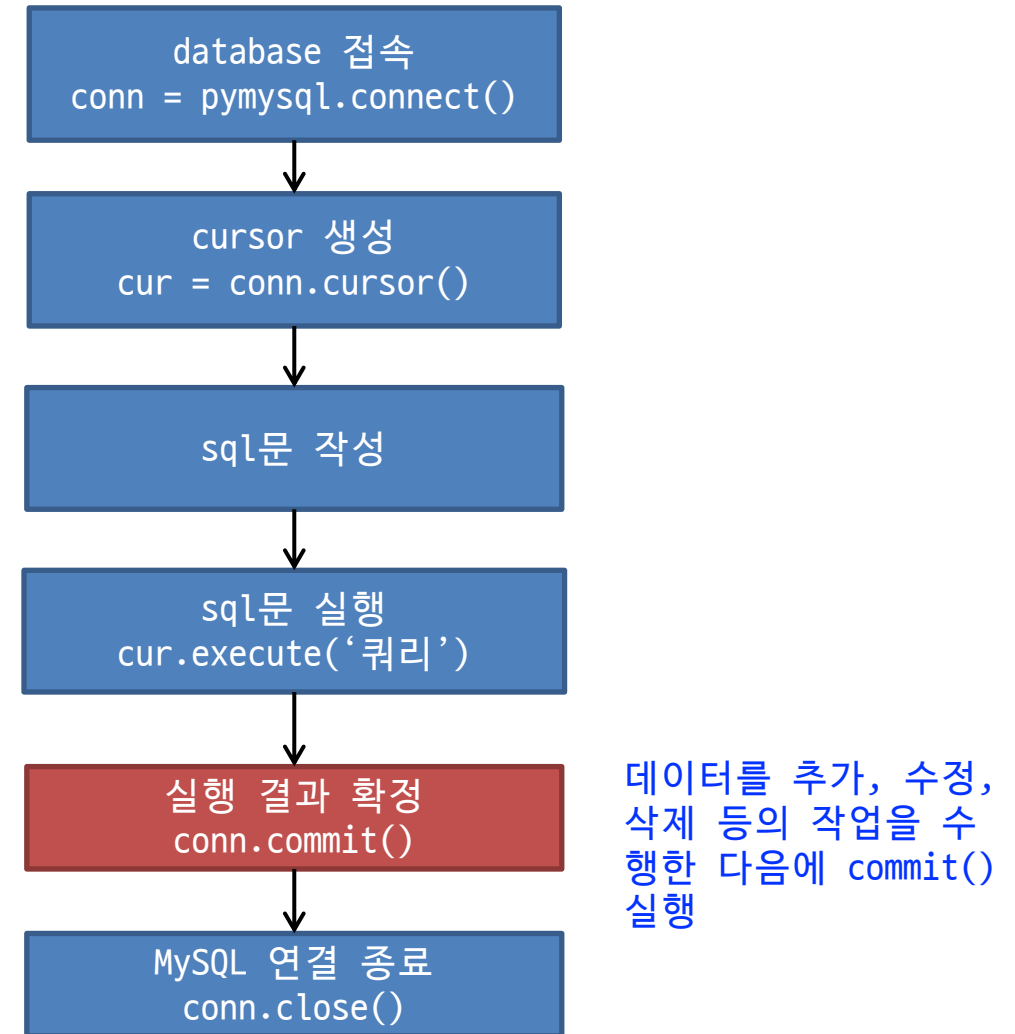
Proceed ([y]/n)?
```

MySQL과 Python 연동하기 (pymysql)

■ 데이터 조회 방법



■ 데이터 추가, 수정, 삭제 방법



MySQL과 Python 연동하기

- `Connection.cursor()` 함수
 - cursor 객체 생성
 - cursor: 쿼리문에 의해 반환되는 결과값을 저장하는 공간
- `cursor.execute('쿼리문장', args=None)` 함수
 - 작성한 쿼리를 실행
- `cursor.executemany('쿼리문장', 튜플 데이터)`
 - 한 번에 여러 개의 데이터(튜플 데이터) 처리
- cursor 객체의 fetch 관련 함수
 - `cursor.fetchall()`: 모든 데이터를 한 번에 가져옴 -> Pandas의 DataFrame 으로 변환 가능
 - `cursor.fetchone()`: 한 번 호출에 하나의 행만 가져옴
 - `cursor.fetchmany(n)`: n개 만큼의 데이터를 가져옴
- `Connection.commit()` 함수
 - 데이터를 추가, 수정, 삭제 등의 작업을 수행한 다음에 실행
- `cursor.close()` 함수
 - cursor 객체 종료

PyMySQL의 cursor 정보

■ cursor.description 속성

- 전체 컬럼의 이름이 description의 각 인덱스[0]에 저장되어 있음

```
import pymysql
import pandas as pd
import csv

conn = pymysql.connect(host='localhost', user='root', password='비밀번호',
                       db='sakila', charset='utf8')

cur = conn.cursor()
cur.execute('select * from language')
```

```
desc = cur.description # 헤더 정보를 가져옴
for i in range(len(desc)):
    print(desc[i][0], end=' ')
print()
```

[i][0]에 column 이름을
가지고 있음

```
rows = cur.fetchall() # 모든 데이터를 가져옴
for data in rows:
    print(data)
print()

cur.close()
conn.close() # 데이터베이스 연결 종료
```

language_id	name	last_update
(1, 'English', datetime.datetime(2006, 2, 15, 5, 2, 19))		
(2, 'Italian', datetime.datetime(2006, 2, 15, 5, 2, 19))		
(3, 'Japanese', datetime.datetime(2006, 2, 15, 5, 2, 19))		
(4, 'Mandarin', datetime.datetime(2006, 2, 15, 5, 2, 19))		
(5, 'French', datetime.datetime(2006, 2, 15, 5, 2, 19))		
(6, 'German', datetime.datetime(2006, 2, 15, 5, 2, 19))		

MySQL과 Python 연동하기

■ pymysql.connect() 함수

- host: DB가 존재하는 서버의 주소(`localhost` 또는 `IP주소`)
- user: 사용자 ID, db: 연결할 데이터베이스 이름
- 리턴: Connection 객체

<mysql1.py>

```
import pymysql
import pandas as pd

conn = pymysql.connect(host='localhost', user='root',
                       password='비밀번호',
                       db = 'sakila', charset='utf8')

cur = conn.cursor()
cur.execute('select * from language')
rows = cur.fetchall() # 모든 데이터를 가져옴
print(rows)

language_df = pd.DataFrame(rows)
print(language_df)

cur.close()
conn.close() # 데이터베이스 연결 종료
```

```
((1, 'English', datetime.datetime(2006, 2, 15, 5, 2, 19)),
(2, 'Italian', datetime.datetime(2006, 2, 15, 5, 2, 19)),
(3, 'Japanese', datetime.datetime(2006, 2, 15, 5, 2, 19)),
(4, 'Mandarin', datetime.datetime(2006, 2, 15, 5, 2, 19)),
(5, 'French', datetime.datetime(2006, 2, 15, 5, 2, 19)),
(6, 'German', datetime.datetime(2006, 2, 15, 5, 2, 19)))
```

DataFrame 형태로 변환

	0	1	2
0	1	English	2006-02-15 05:02:19
1	2	Italian	2006-02-15 05:02:19
2	3	Japanese	2006-02-15 05:02:19
3	4	Mandarin	2006-02-15 05:02:19
4	5	French	2006-02-15 05:02:19
5	6	German	2006-02-15 05:02:19

MySQL과 Python 연동하기

■ DictCursor 사용

- pymysql.connect(, cursorclass=pymysql.cursors.DictCursor)
- conn.cursor(pymysql.cursors.DictCursor)
 - DataFrame의 column들을 같이 리턴함

```
import pymysql
import pandas as pd

conn = pymysql.connect(host='localhost', user='root', password='비밀번호',
                       db='sakila', charset='utf8')
cur = conn.cursor(pymysql.cursors.DictCursor)
cur.execute('select * from language')
rows = cur.fetchall() # 모든 데이터를 가져옴

language_df = pd.DataFrame(rows) # DataFrame 형태로 변환
print(language_df)
#print(language_df.iloc[0:3])
print()
print(language_df['name'])
cur.close()
conn.close() # 데이터베이스 연결 종료
```

DataFrame의 column 정보도
같이 출력됨

	language_id	name	last_update
0	1	English	2006-02-15 05:02:19
1	2	Italian	2006-02-15 05:02:19
2	3	Japanese	2006-02-15 05:02:19
3	4	Mandarin	2006-02-15 05:02:19
4	5	French	2006-02-15 05:02:19
5	6	German	2006-02-15 05:02:19

0	English
1	Italian
2	Japanese
3	Mandarin
4	French
5	German

MySQL과 Python 연동하기

- 복잡한 쿼리 실행
 - inner join 내용

<mysql2.py>

```
import pymysql

conn = pymysql.connect(host='localhost', user='root', password='비밀번호',
                        db='sakila', charset='utf8')
cur = conn.cursor()

query = """
select c.email
from customer as c
      inner join rental as r
      on c.customer_id = r.customer_id
where date(r.rental_date) = (%s)"""

cur.execute(query, ('2005-06-14'))
rows = cur.fetchall() # 모든 데이터를 가져옴
for row in rows:
    print(row)

cur.close()
conn.close()
```

실제 쿼리와 동일한 문자열
전달 (따옴표 주의)

쿼리에 전달된 값
(%s): 문자열

MySQL과 Python 연동하기

■ 실행 결과

```
('JEFFERY.PINSON@sakilacustomer.org',)
('ELMER.NOE@sakilacustomer.org',)
('MINNIE.ROMERO@sakilacustomer.org',)
('MIRIAM.MCKINNEY@sakilacustomer.org',)
('DANIEL.CABRAL@sakilacustomer.org',)
('TERRANCE.ROUSH@sakilacustomer.org',)
('JOYCE.EDWARDS@sakilacustomer.org',)
('GWENDOLYN.MAY@sakilacustomer.org',)
('CATHERINE.CAMPBELL@sakilacustomer.org',)
('MATTHEW.MAHAN@sakilacustomer.org',)
('HERMAN.DEVORE@sakilacustomer.org',)
('AMBER.DIXON@sakilacustomer.org',)
('TERRENCE.GUNDERSON@sakilacustomer.org',)
('SONIA.GREGORY@sakilacustomer.org',)
('CHARLES.KOWALSKI@sakilacustomer.org',)
('JEANETTE.GREENE@sakilacustomer.org',)
```

테이블 생성

<mysql3.py>

```
import pymysql

def create_table(conn, cur):
    try:
        query1 = "drop table if exists customer"
        query2 = """
            create table customer
            (name varchar(10),
            category smallint,
            region varchar(10))
        """
        cur.execute(query1)
        cur.execute(query2)
        conn.commit()
        print('Table 생성 완료')
    except Exception as e:
        print(e)
```

customer 테이블
생성

```
def main():
    conn = pymysql.connect(host='localhost', user='root',
                           password='비밀번호',
                           db='sqlclass_db',
                           charset='utf8')

    cur = conn.cursor()

    # 테이블 생성 함수 호출
    create_table(conn, cur)

    # 연결 종료
    cur.close()
    conn.close()
    print('Database 연결 종료')
```

main()

DBaver에서
customer테이블 생성 확인

컬럼명	#	Data Type	Not Null	Auto Increment	Key
ABC name	1	varchar(10)	[]	[]	
123 category	2	smallint	[]	[]	
ABC region	3	varchar(10)	[]	[]	

데이터 추가: INSERT

■ execute() 예제

<mysql4.py>

```
import pymysql

conn = pymysql.connect(host='localhost', user='root', password= ' 비밀번호',
                        db='sqlclass_db', charset='utf8')

curs = conn.cursor()
sql = """insert into customer(name, category, region)
        values (%s, %s, %s)"""

curs.execute(sql, ('홍길동', 1, '서울'))
curs.execute(sql, ('이연수', 2, '서울'))
conn.commit()

print('INSERT 완료')

curs.execute('select * from customer')
rows = curs.fetchall() # 모든 데이터를 가져옴
print(rows)

curs.close()
conn.close()
```

customer 테이블에
데이터 추가

INSERT 완료
(('홍길동', 1, '서울'), ('이연수', 2, '서울'))

데이터 추가

■ executemany()

- 여러 개의 tuple 데이터를 처리

<mysql5.py>

```
import pymysql

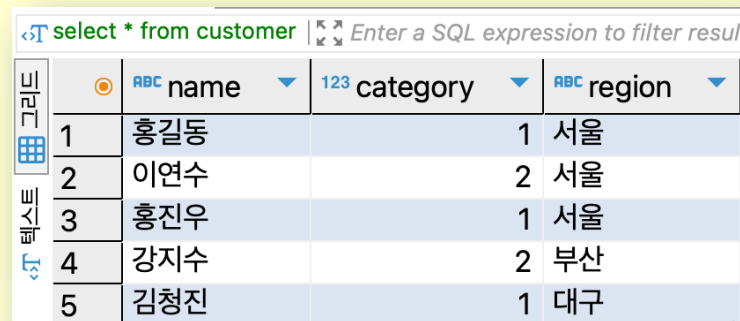
conn = pymysql.connect(host='localhost', user='root', password='비밀번호',
                        db='sqlclass_db', charset='utf8')
curs = conn.cursor()

sql = """insert into customer(name, category, region)
        values (%s, %s, %s)"""

data = (
    ('홍진우', 1, '서울'),
    ('강지수', 2, '부산'),
    ('김청진', 1, '대구'),
)

curs.executemany(sql, data)

conn.commit()
print('executemany() 완료')
curs.close()
conn.close()
```



The screenshot shows a MySQL database window with the query 'select * from customer' entered. The results are displayed in a table with columns: name, category, and region. The data rows are:

	name	category	region
1	홍길동	1	서울
2	이연수	2	서울
3	홍진우	1	서울
4	강지수	2	부산
5	김청진	1	대구

UPDATE, DELETE

<mysql6.py>

```
'''  
UPDATE, DELETE  
'''  
  
import pymysql  
  
conn = pymysql.connect(host='localhost', user='root', password= '비밀번호',  
                        db='sqlclass_db', charset='utf8')  
curs = conn.cursor()
```

```
sql = """  
    update customer  
    set region = '서울특별시'  
    where region='서울'  
    """
```

```
curs.execute(sql)  
print('update 완료')
```

```
sql = "delete from customer where name=%s"  
curs.execute(sql, '홍길동')  
print('delete 홍길동')
```

```
conn.commit()  
conn.close()
```

select * from customer | Enter a SQL expression to filter result

	ABC name	123 category	ABC region
1	이연수	2	서울특별시
2	홍진우	1	서울특별시
3	강지수	2	부산
4	김청진	1	대구

참고 사이트

- PyMySQL documentation

- <https://pymysql.readthedocs.io/en/latest/#>

- PyMySQL과 DataFrame 연결

- <https://ssoonidev.tistory.com/71>

- MySQLCursor.description 정보

- <https://dev.mysql.com/doc/connector-python/en/connector-python-api-mysqldb-cursor-description.html>



Questions?