more_tables

October 14, 2025

0.1 Connect to DB

[1]:
```
from dotenv import load_dotenv
import os
load_dotenv('../.env')
password = os.getenv("PASS")
```

[2]:
```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password=password,
  database="data1050f25"
)

print(mydb)

if mydb.is_connected():
    print("CONNECTION SUCCESSFUL")
```

```
<mysql.connector.connection_cext.CMySQLConnection object at 0x1083730e0>
CONNECTION SUCCESSFUL
```

0.2 Create Tables

[53]:
```
mycursor = mydb.cursor()

# create time_slot, section, teaches, and takes
create_time_slot = """CREATE TABLE time_slot (
time_slot_id INT PRIMARY KEY,
day VARCHAR(128) NOT NULL,
start_time TIME NOT NULL,
end_time TIME NOT NULL
)
"""

mycursor.execute(create_time_slot)
```

[54]:
```
create_section = """CREATE TABLE section (
course_id VARCHAR(10),
sec_id INT,
semester VARCHAR(128),
year INT,
building VARCHAR(255),
room_no INT,
time_slot_id INT,
PRIMARY KEY (course_id, sec_id, semester, year),
FOREIGN KEY(time_slot_id) REFERENCES time_slot(time_slot_id),
FOREIGN KEY(course_id) REFERENCES course(course_id)
)
"""

mycursor = mydb.cursor()
mycursor.execute(create_section)
```

[55]:
```
create_teaches = """CREATE TABLE teaches (
ID VARCHAR(128),
course_id VARCHAR(10),
sec_id INT,
semester VARCHAR(128),
year INT,
FOREIGN KEY(ID) REFERENCES instructor(ID),
FOREIGN KEY(course_id, sec_id, semester, year) REFERENCES
    section(course_id, sec_id, semester, year),
PRIMARY KEY(ID, course_id, sec_id, semester, year)
)"""

mycursor = mydb.cursor()
mycursor.execute(create_teaches)
```

[56]:
```
create_takes = """CREATE TABLE takes (
ID VARCHAR(128),
course_id VARCHAR(10),
sec_id INT,
semester VARCHAR(128),
year INT,
grade VARCHAR(1),
FOREIGN KEY(ID) REFERENCES student(ID),
FOREIGN KEY(course_id, sec_id, semester, year) REFERENCES
    section(course_id, sec_id, semester, year),
PRIMARY KEY(ID, course_id, sec_id, semester, year)
)"""

mycursor = mydb.cursor()
mycursor.execute(create_takes)
```

[57]:
```
mycursor = mydb.cursor()
mycursor.execute("SHOW TABLES")

for x in mycursor:
    print(x)
```

```
('advisor',)
('classroom',)
('course',)
('department',)
('instructor',)
('prereq',)
('section',)
('student',)
('takes',)
('teaches',)
('time_slot',)
```

0.3 Add Check Constrains

[65]:
```
# check that semester in section is in Fall, Spring, Summer, Winter
check_semester = "ALTER TABLE section " \
"ADD CONSTRAINT check_section_semester " \
"CHECK (semester IN ('Fall','Spring','Summer','Winter'))"


print(check_semester)
mycursor = mydb.cursor()
mycursor.execute(check_semester)
```

```
ALTER TABLE section ADD CONSTRAINT check_section_semester CHECK (semester IN
('Fall','Spring','Summer','Winter'))
```

[66]:
```
# check that the year is at least >1764 (founding of university)
check_year = "ALTER TABLE section " \
"ADD CONSTRAINT check_section_year " \
"CHECK (year > 1764)"

print(check_year)
mycursor = mydb.cursor()
mycursor.execute(check_year)
```

```
ALTER TABLE section ADD CONSTRAINT check_section_year CHECK (year > 1764)
```

[67]:
```
# check that the time_slot days are in the week
check_day = "ALTER TABLE time_slot " \
"ADD CONSTRAINT check_day_in_week " \
"CHECK (day IN␣
 ↪('Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday'))"
```

```
print(check_day)
mycursor = mydb.cursor()
mycursor.execute(check_day)
```

```
ALTER TABLE time_slot ADD CONSTRAINT check_day_in_week CHECK (day IN
('Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday'))
```

0.4 Populate Tables

[58]:
```
import pandas as pd
import numpy as np

# populate tables time_slot, section, teaches, and takes
time_slot_data = pd.read_csv("../data/sql-hw/time_slot.csv")
section_data = pd.read_csv("../data/sql-hw/section.csv")
teaches_data = pd.read_csv("../data/sql-hw/teaches.csv")
takes_data = pd.read_csv("../data/sql-hw/takes.csv")

all_data = {"time_slot":time_slot_data,
            "section":section_data,
            "teaches":teaches_data,
            "takes":takes_data}

for name, data in all_data.items():
    print(f"inserting {data.shape[0]} rows into {name}")
    data = data.replace({np.nan: None, pd.NA: None})
    placeholders = "%s," * (data.shape[1]-1) + "%s"
    for i,row in data.iterrows():
                sql = f"INSERT INTO {name} VALUES ({placeholders})"
                mycursor.execute(sql, tuple(row))
                print("Record inserted")
                mydb.commit()
```

```
inserting 18 rows into time_slot
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
```

```
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
inserting 14 rows into section
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
inserting 11 rows into teaches
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
inserting 23 rows into takes
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
```

```
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
Record inserted
```

0.5 View Records

[60]:
```python
# intro :)
print("peak at records from... \n")
# iter over list of tables
for table in ["time_slot","section","teaches","takes"]:
    # print title
    print(f"{table.upper()}")
    # select 5 records and print
    mycursor = mydb.cursor()
    mycursor.execute(f"SELECT * FROM {table} LIMIT 5")
    for x in mycursor:
        print(x)
    print()
```

```
peak at records from…

TIME_SLOT
(1, 'Monday', datetime.timedelta(seconds=32400),
datetime.timedelta(seconds=39600))
(2, 'Monday', datetime.timedelta(seconds=39600),
datetime.timedelta(seconds=3600))
(3, 'Monday', datetime.timedelta(seconds=3600),
datetime.timedelta(seconds=10800))
(4, 'Monday', datetime.timedelta(seconds=10800),
datetime.timedelta(seconds=18000))
(5, 'Tuesday', datetime.timedelta(seconds=32400),
datetime.timedelta(seconds=39600))

SECTION
('APMA1650', 1, 'Spring', 2023, 'Sayles', 128, 3)
('CSCI1270', 1, 'Fall', 2023, 'CIT', 312, 6)
('CSCI1270', 2, 'Fall', 2023, None, None, 6)
('Data1030', 1, 'Fall', 2022, 'Franklin', 212, 6)
('Data1030', 1, 'Fall', 2023, 'Franklin', 212, 6)

TEACHES
('113', 'APMA1650', 1, 'Spring', 2023)
('112', 'CSCI1270', 1, 'Fall', 2023)
```

```
('118', 'CSCI1270', 2, 'Fall', 2023)
('117', 'Data1030', 1, 'Fall', 2023)
('117', 'Data1050', 1, 'Fall', 2022)

TAKES
('1122', 'CSCI1270', 2, 'Fall', 2023, None)
('1122', 'Data1050', 1, 'Fall', 2023, None)
('1238', 'CSCI1270', 1, 'Fall', 2023, None)
('1238', 'Data1030', 1, 'Fall', 2023, None)
('1238', 'Data1050', 1, 'Fall', 2022, 'A')
```

0.6 Run Query

*Find `ID` and **`name`** for all DSI students taking Data 1050 in Fall, 2023.*

[78]:
```
query1 = "SELECT student.ID, student.name " \
"FROM student " \
"INNER JOIN takes " \
"ON student.ID = takes.ID " \
"WHERE student.dept_name = 'DSI' " \
"AND takes.course_id = 'Data1050' " \
"AND takes.semester = 'Fall' " \
"AND takes.year = 2023"

mycursor = mydb.cursor()
mycursor.execute(query1)

for x in mycursor:
    print(x)
```

```
('1845', 'Jane Chen')
('2142', 'Sabrina Zhou')
```