

# mag2352: HW0 and HW1

## Step 3: Test File Import

Replace the UNI in the steps with your UNI.

```
In [1]: import mag2352_HW0
```

```
In [2]: mag2352_HW0.t1()
```

```
Out[2]: 'mag2352 says Hello World'
```

The text above should look like my example, but with you UNI.

**Note:** Any time you change the underlying Python file, you must restart the kernel using the menu. You must then re-import and rerun any cells.

## Step 4: Install PyMYSQL and iPython-SQL

- You run the commands below in an Anaconda terminal window.
- [Install](#) pymysql in your Anaconda environment.
- [Install](#) iPython-SQL in your Anaconda environment.
- Restart the notebook Kernel.
- The following cell should execute.

```
In [3]: import pymysql  
pymysql.__version__
```

```
Out[3]: '1.0.2'
```

- In the cell below, replace `dbuser:dbuserdbuser` with your MySQL user ID and password.

```
In [4]: %load_ext sql  
  
%sql mysql+pymysql://root:dbuserdbuser@192.168.1.137
```

- The following is a simple test. You should get similar results, but your might be slightly different.

```
In [5]: %sql show tables from information_schema
```

```
* mysql+pymysql://root:***@192.168.1.137
79 rows affected.
```

Out[5]:

**Tables in information\_schema**

```
ADMINISTRABLE_ROLE_AUTHORIZATIONS
APPLICABLE_ROLES
CHARACTER_SETS
CHECK_CONSTRAINTS
COLLATION_CHARACTER_SET_APPLICABILITY
COLLATIONS
COLUMN_PRIVILEGES
COLUMN_STATISTICS
COLUMNS
COLUMNS_EXTENSIONS
ENABLED_ROLES
ENGINES
EVENTS
FILES
INNODB_BUFFER_PAGE
INNODB_BUFFER_PAGE_LRU
INNODB_BUFFER_POOL_STATS
INNODB_CACHED_INDEXES
INNODB_CMP
INNODB_CMP_PER_INDEX
INNODB_CMP_PER_INDEX_RESET
INNODB_CMP_RESET
INNODB_CMPMEM
INNODB_CMPMEM_RESET
INNODB_COLUMNS
INNODB_DATAFILES
INNODB_FIELDS
INNODB_FOREIGN
INNODB_FOREIGN_COLS
INNODB_FT_BEING_DELETED
INNODB_FT_CONFIG
INNODB_FT_DEFAULT_STOPWORD
INNODB_FT_DELETED
```

INNODB\_FT\_INDEX\_CACHE  
INNODB\_FT\_INDEX\_TABLE  
INNODB\_INDEXES  
INNODB\_METRICS  
INNODB\_SESSION\_TEMP\_TABLESPACES  
INNODB\_TABLES  
INNODB\_TABLESPACES  
INNODB\_TABLESPACES\_BRIEF  
INNODB\_TABLESTATS  
INNODB\_TEMP\_TABLE\_INFO  
INNODB\_TRX  
INNODB\_VIRTUAL  
KEY\_COLUMN\_USAGE  
KEYWORDS  
OPTIMIZER\_TRACE  
PARAMETERS  
PARTITIONS  
PLUGINS  
PROCESSLIST  
PROFILING  
REFERENTIAL\_CONSTRAINTS  
RESOURCE\_GROUPS  
ROLE\_COLUMN\_GRANTS  
ROLE\_ROUTINE\_GRANTS  
ROLE\_TABLE\_GRANTS  
ROUTINES  
SCHEMA\_PRIVILEGES  
SCHEMATA  
SCHEMATA\_EXTENSIONS  
ST\_GEOMETRY\_COLUMNS  
ST\_SPATIAL\_REFERENCE\_SYSTEMS  
ST\_UNITS\_OF\_MEASURE  
STATISTICS  
TABLE\_CONSTRAINTS  
TABLE\_CONSTRAINTS\_EXTENSIONS

TABLE\_PRIVILEGES  
TABLES  
TABLES\_EXTENSIONS  
TABLESPACES  
TABLESPACES\_EXTENSIONS  
TRIGGERS  
USER\_ATTRIBUTES  
USER\_PRIVILEGES  
VIEW\_ROUTINE\_USAGE  
VIEW\_TABLE\_USAGE  
VIEWS

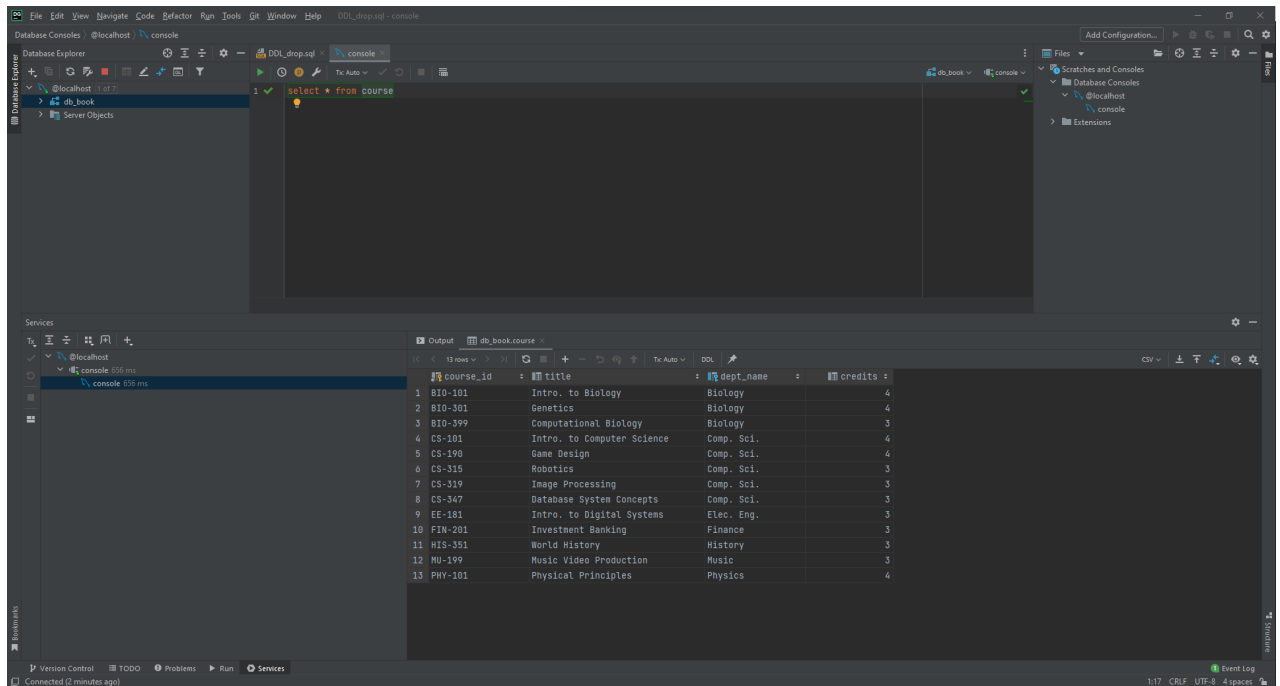
## Step 5: Load Sample Data

- In the directory where you cloned the project, there is a sub-folder `db_book`.
- Start DataGrip.
- In DataGrip, choose `File->New DataSource->MySQL`.
  - Accept the default name for the data source.
  - Set the MySQL user ID and password.
  - You may see a message stating that you need to install database drives. Install the drivers.
- Select the newly created data source. The name will be `Run SQL Script`. Navigate to and choose the file `DDL_drop.sql`.
- Do the same for `smallRelationsInsertFile.sql`.
- You will see an icon/text on the side bar labelled `db_book`. It may be greyed-out. Right click on the entry and choose `New query console`. You may see a message `Current schema not introspected` and `Introspect schema` on the far right. Click on `Introspect schema`.
- Enter `select * from course` in the query console window. Click on the little green arrow to run the query.
- Take a screenshot of your DataGrip window and save the screenshot into the folder of the form `dff9_src` using your UNI. Remember the name of the file.
- Set your file name in the cell below replacing the example and run the cell. You should see your screenshot below. Yours will look a little different from mine. As long as yours shows the query result, you are fine.

```
In [6]: file_name = 'Screenshot 2022-01-24 152824.png'
```

```
print("\n")
from IPython.display import Image
Image(filename=file_name)
```

Out[6]:



## Step 6: Very %sql

- Execute the cell below. Your answer will be similar to mine but may not match exactly.

In [7]:

```
%sql select * from db_book.course
```

```
* mysql+pymysql://root:***@192.168.1.137
13 rows affected.
```

Out[7]:

course_id	title	dept_name	credits
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3

MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

## Step 7: Pandas, CSV and SQL

- Run the cell below.

```
In [8]: import pandas
pandas.__version__
```

```
Out[8]: '1.3.4'
```

- Install [SQLAlchemy](#) using an Anaconda prompt.
- Restart the notebook kernel and rerun all cells. Then run the cell below.

```
In [9]: from sqlalchemy import create_engine
```

- Go into DataGrip. Select your local database, e.g. @localhost .
- Open a query console and execute `create database lahmansdb` . Then execute the cell below.

```
In [10]: %sql show tables from lahmansdb;

* mysql+pymysql://root:***@192.168.1.137
1 rows affected.
```

```
Out[10]: Tables_in_lahmansdb

people
```

- There is a folder `data` in the project you cloned. There is a file in the folder `People.csv` .
- Execute the following code cell. If you are on Windows, you may have to change the path to the file and may have to replace `/` with `\\` in paths.
- You should see a result similar to mine below.

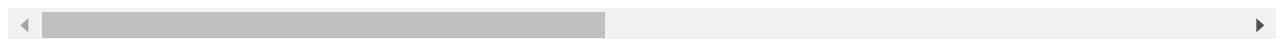
```
In [11]: df = pandas.read_csv('../data/People.csv')
df
```

```
Out[11]:
```

	playerID	birthYear	birthMonth	birthDay	birthCountry	birthState	birthCity	deathYear	deathMonth	deathDay
0	aardsda01	1981.0	12.0	27.0	USA	CO	Denver	NaN	NaN	NaN
1	aaronha01	1934.0	2.0	5.0	USA	AL	Mobile	2021.0	1.0	1.0

	playerID	birthYear	birthMonth	birthDay	birthCountry	birthState	birthCity	deathYear	deathMonth
2	aaronto01	1939.0	8.0	5.0	USA	AL	Mobile	1984.0	
3	aasedo01	1954.0	9.0	8.0	USA	CA	Orange	NaN	
4	abadan01	1972.0	8.0	25.0	USA	FL	Palm Beach	NaN	
...	...	...	...	...	...	...	...	...	...
20353	zupofr01	1939.0	8.0	29.0	USA	CA	San Francisco	2005.0	
20354	zuvelpa01	1958.0	10.0	31.0	USA	CA	San Mateo	NaN	
20355	zuverge01	1924.0	8.0	20.0	USA	MI	Holland	2014.0	
20356	zwilldu01	1888.0	11.0	2.0	USA	MO	St. Louis	1978.0	
20357	zychto01	1990.0	8.0	7.0	USA	IL	Monee	NaN	

20358 rows × 24 columns



- We will now save the data to MySQL. Run the cells below. You will have to change `dbuser:dbuserdbuser` to your MySQL user ID and password.

```
In [12]: engine = create_engine("mysql+pymysql://root:dbuserdbuser@192.168.1.137")
```

```
In [13]: df.to_sql('people', con=engine, index=False, if_exists='replace', schema='lahmansdb')
```

- Test that you wrote the information to the databases.

```
In [14]: %sql select * from lahmansdb.people where nameLast='Williams' and bats='L'
```

```
* mysql+pymysql://root:***@192.168.1.137
19 rows affected.
```

```
Out[14]: playerID  birthYear  birthMonth  birthDay  birthCountry  birthState  birthCity  deathYear  deathMonth
williar01    1877.0        8.0        24.0        USA          MA      Somerville    1941.0        5.0
willibi01    1938.0        6.0        15.0        USA          AL      Whistler      None         None
willibi02    1932.0        6.0        13.0        USA          SC      Newberry      2013.0        6.0
willicy01    1887.0       12.0        21.0        USA          IN      Wadena        1974.0        4.0
```

willida05	1958.0	2.0	28.0	USA	NY	Brooklyn	None	None
willida07	1979.0	3.0	12.0	USA	AK	Anchorage	None	None
willide01	1896.0	12.0	13.0	USA	OR	Portland	1929.0	3.0
willigu02	1888.0	5.0	7.0	USA	NE	Omaha	1964.0	4.0
williju02	1995.0	8.0	20.0	USA	LA	Houma	None	None
willike01	1890.0	6.0	28.0	USA	OR	Grants Pass	1959.0	1.0
willile03	1905.0	12.0	2.0	USA	GA	Macon	1984.0	11.0
willima02	1953.0	7.0	28.0	USA	NY	Elmira	None	None
willima07	1991.0	8.0	21.0	USA	RI	Pawtucket	None	None
willimi02	1964.0	11.0	17.0	USA	CA	Santa Ana	None	None
willini01	1993.0	9.0	8.0	USA	TX	Galveston	None	None
willira01	1975.0	9.0	18.0	USA	TX	Harlingen	None	None
williri02	1893.0	12.0	18.0	USA	CA	Santa Cruz	1966.0	4.0
willist01	1892.0	1.0	31.0	USA	MT	Cascade	1979.0	6.0
willite01	1918.0	8.0	30.0	USA	CA	San Diego	2002.0	7.0

## Step 7: Done

- You are done.

## Programming Track

- Include a screen capture of your PyCharm execution of the web application. Your should look like the one below but may be different.

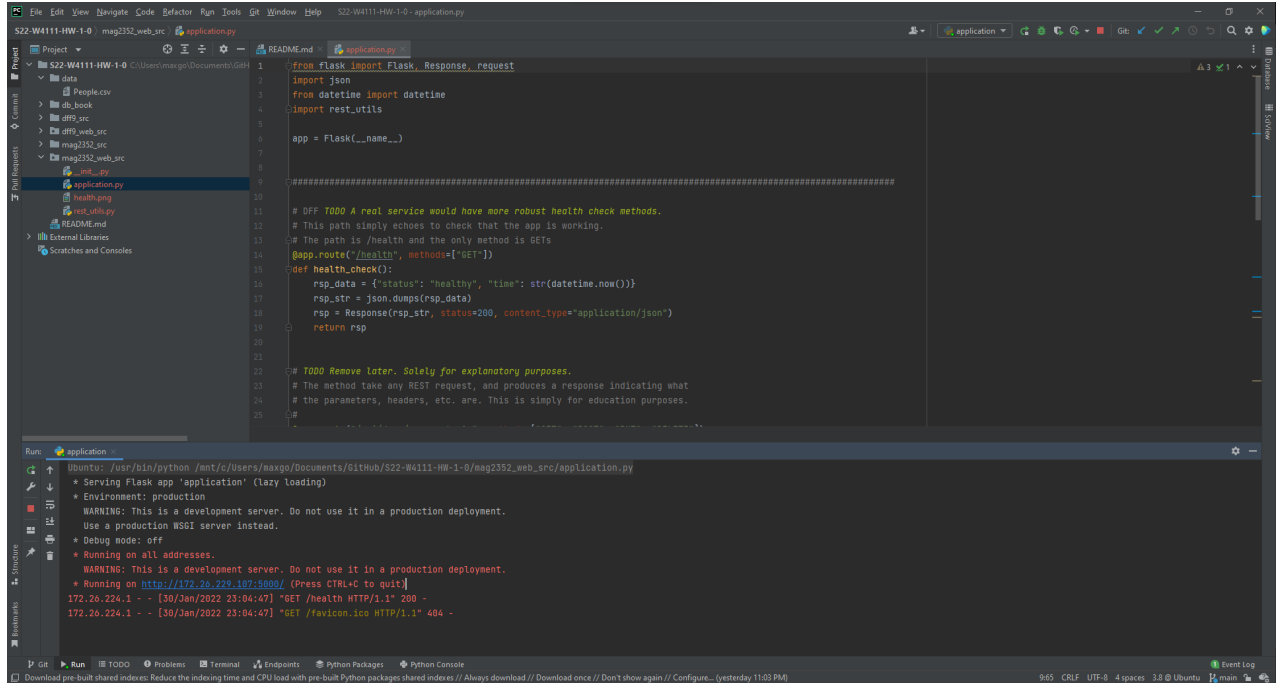
In [16]:

```
file_name = 'pycharm.png'
print("\n")
```



```
from IPython.display import Image
Image(filename=file_name)
```

Out[16]:



The screenshot shows a VS Code editor with a project named 'S22-W4111-HW-1-0'. The file explorer on the left shows a directory structure with files like 'data', 'people.csv', 'app.py', 'app\_src', 'app2352\_src', 'app2352\_web\_src', 'app.py', 'application.py', 'health.png', 'rest\_v0.py', and 'README.md'. The main editor window shows the content of 'application.py', which is a Flask application. The code includes imports for Flask, Response, request, json, datetime, and rest\_utils. It defines a Flask app and a health check endpoint. The terminal at the bottom shows the command to run the application: 'python /mnt/c/Users/maxgo/Documents/GitHub/S22-W4111-HW-1-0/app2352\_web\_src/application.py'. The output shows that the application is running on http://172.26.229.107:5000/ and is serving the health check endpoint.

```
1 from flask import Flask, Response, request
2 import json
3 from datetime import datetime
4 import rest_utils
5 app = Flask(__name__)
6
7
8
9
10
11 # TODO A real service would have more robust health check methods.
12 # This path simply echoes to check that the app is working.
13 # The path is /health and the only method is GETs
14 @app.route('/health', methods=['GET'])
15 def health_check():
16     rsp_data = {'status': 'healthy', 'time': str(datetime.now())}
17     rsp_str = json.dumps(rsp_data)
18     rsp = Response(rsp_str, status=200, content_type='application/json')
19     return rsp
20
21
22 # TODO Remove later. Solely for explanatory purposes.
23 # The method take any REST request, and produces a response indicating what
24 # the parameters, headers, etc. are. This is simply for education purposes.
25
26
```

Runs application -

```
Ubuntu: /usr/bin/python /mnt/c/Users/maxgo/Documents/GitHub/S22-W4111-HW-1-0/app2352_web_src/application.py
* Serving Flask app 'application' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
  * Running on http://172.26.229.107:5000/ (Press CTRL+C to quit)
172.26.224.1 - - [30/Jan/2022 23:04:47] "GET /health HTTP/1.1" 200 -
172.26.224.1 - - [30/Jan/2022 23:04:47] "GET /favicon.ico HTTP/1.1" 404 -
```

- Put a screen capture of access the web page. Yours will look similar to mine but may be slightly different

In [18]:

```
file_name = 'browser.png'

print("\n")
from IPython.display import Image
Image(filename=file_name)
```

Out[18]:

