



App Fullstack Developer Program Integrate MySQL to an Express App

Instruction:

For submission,

1. Upload your source code in a zip file to Talentlabs Classroom. Please don't include the node_modules folder.
2. Upload this document as a PDF file as well.

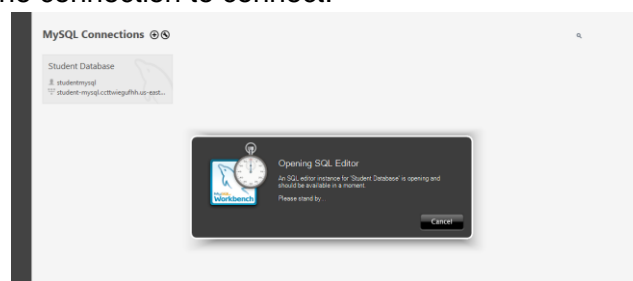
Prepare: Setting up MySQL WorkBench

Instruction

1. Download from here <https://www.mysql.com/products/workbench/>
2. Install the application.
3. Create a new Database connection by clicking the "+" button:

MySQL Connections + ↻

4. Fill in the connection details:
Connection Name: Any name is ok here
Hostname: student-mysql.ccttwiegufhh.us-east-2.rds.amazonaws.com
Username: studentmysql
Password: studentmysql
5. Click "Test connection" and it should work.
6. Click "OK" to store the connection.
7. Double click the connection to connect:



Prepare: Express Application Generator

Steps:

1. Execute the command: **npx express-generator --ejs express-lab-11**
2. Change directory into that new folder: **cd express-lab-11**
3. Install all npm packages: **npm install**
4. Start the server
 - Windows PowerShell:

- **\$env:DEBUG='express-lab-11:*'; npm start**
- Windows Command Prompt:
set DEBUG=express-lab-11:* & npm start
- MacOS or Linux:
SET DEBUG=express-lab-11:* & npm start

Task 1: Seed Data

Before working on this, please make sure you have created the tables via MySQL workbench in lab 10.

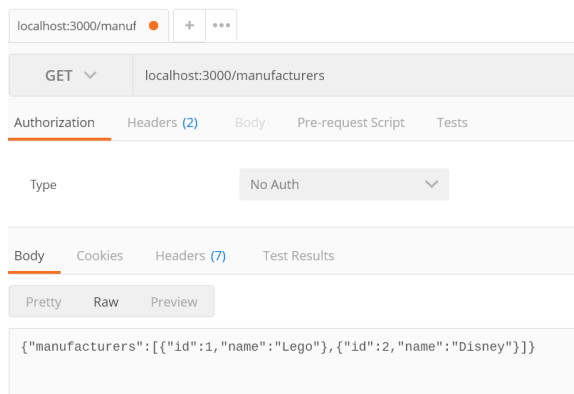
Steps:

- Follow slides 3~5 to set up the **knex.js** file.
- Follow slide 7 to create the **default seed files**.
 - `npx knex seed:make initial-manufacturer`
 - `npx knex seed:make initial-product`
- Follow slides 8~10 to update the seed files to use **upsert**.
- Follow slide 11 to execute the seed files.
- Use MySQL workbench to see if the seed data is created.

Task 2: Get the data from Express

Steps:

- Follow slides 15 to set up the **database.js** file.
- Follow slide 16 ~ 18 to setup the **./routes/index.js** file.
 - This should create the List manufacturers API
- Use Postman to test this API
 - GET localhost:3000/manufacturers



Task 3: List, Retrieve, Create, Update, Delete

Steps:

- Follow slides 20 ~ 30 to set up the **rest of the manufacturers APIs**
- For each List, Retrieve, Create, Update, Delete API
 - Take a screenshot of Postman to show how you test the API.
 - Take a screenshot of MySQL workbench to show the result Manufacturers table.

For example:

List Manufacturer:

The screenshot shows a Postman interface for a GET request to `http://localhost:3000/manufacturers`. The request is successful, returning a 200 OK status with a response time of 1612 ms and a size of 302 B. The response body is displayed in the 'Body' tab, showing a JSON array of two manufacturer objects: `[{"id":1,"name":"Lego"}, {"id":2,"name":"Disney"}]`.

KEY	VALUE	DESCRIPTION
Key	Value	Description

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body | Cookies | Headers (7) | Test Results

Status: 200 OK Time: 1612 ms Size: 302 B Save Response

Pretty Raw Preview Visualize

```
{
  "manufacturers": [
    {
      "id": 1,
      "name": "Lego"
    },
    {
      "id": 2,
      "name": "Disney"
    }
  ]
}
```

</talentlabs>

Query 1 x SQL File 1*

1 select * from manufacturer;

<

Result Grid Filter Rows: Ed

	id	name
▶	1	Lego
	2	Disney
★	NULL	NULL

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap C

	id	name
▶	2	Disney
*	NULL	NULL

</talentlabs>

CREATE

http://localhost:3000/manufacturers

POST http://localhost:3000/manufacturers

Params Authorization Headers (8) Body Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

```
1 {  
2   ... "name": "Marvel"  
3 }
```

Body Cookies Headers (7) Test Results

🌐 200 OK

Pretty Raw Preview Visualize **JSON** ▾



```
1 {  
2   "message": "Done"  
3 }
```

</talentlabs>

Query 1 x SQL File 1*

1 select * from manufacturer;
2

Result Grid

	id	name
▶	1	Lego
	2	Disney
	3	Marvel
•	NULL	NULL

UPDATE

<http://localhost:3000/manufacturers/2>

Save

PUT

<http://localhost:3000/manufacturers/2>

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON

Beautify

```
1 {
2   ... "name": "DC"
3 }
```

Body

Cookies

Headers (7)

Test Results

200 OK 371 ms 253 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "message": "Done"
3 }
```


</talentlabs>

Query 1 x SQL File 1*

Limit to 1000 rows

```
1 select * from manufacturer;  
2
```

Result Grid

Filter Rows:



Edit: Export/Import: Wrap Cell

	id	name
▶	1	Lego
	2	DC
	3	Marvel
•	NULL	NULL

DELETE

http://localhost:3000/manufacturers/3

Save

DELETE

http://localhost:3000/manufacturers/3

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON

Beautify

```
1 {
2   ... "name": "Maivel"
3 }
```

Body

Cookies

Headers (7)

Test Results

200 OK

368 ms

253 B

Save Response

Pretty

Raw

Preview

Visualize

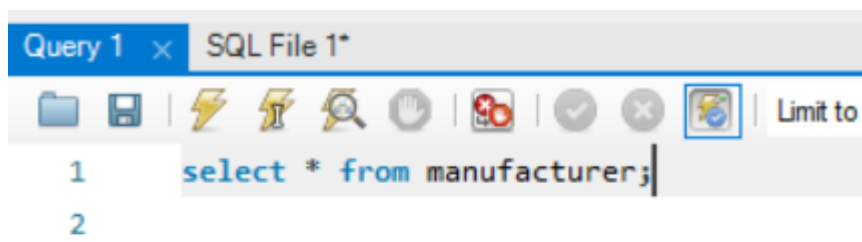
JSON



```
1 {
2   "message": "Done"
3 }
```

</talentlabs>



The screenshot shows the 'Result Grid' view of the query results. The grid has two columns: 'id' and 'name'. The results are as follows:

	id	name
▶	1	Lego
	2	DC
*	NULL	NULL