COMP 2350

Lab Report:

SELECTs Lab

By Yeseol (Sol) Kim

Submitted to: Patrick Guichon

Introduction:

Today, in the lab, I practiced creating MySQL databases and tables using commands. I was able to create new files using "create database" or "create table" and delete them using "drop database" or "drop table." It was very convenient as we could also specify the values and data types for each column all at once. Additionally, I learned to create backup files for the database, and I could later import these backup files. During this process, I was able to generate complex database Entity-Relationship Diagrams (ERDs) for a quick overview.

Next, I focused on scenarios, where we could manipulate data based on specific commands. For example, I could sort data in ascending or descending order, and combine sorting by last name and first name to display data by names. This allowed us to organize the data according to our needs using various commands.

Finally, I connected MySQL with Render that I worked on last week, enabling us to directly send data to Render. However, modifying values in each EJS and JS file proved to be a bit challenging. Through multiple Git pushes, I identified and rectified errors in the process.

Screenshot #1:

This is how to create database and give the primary key and value of each column.

While commend can add datatype as well.

Filename: 01_Create_Pet_Table_Command.png

```
CREATE DATABASE intro_to_sql_y_kim;
use Intro_to_sql_y_kim;
CREATE TABLE pet (
pet_id int auto_increment NOT NULL,
pet_name varchar(50) NULL,
 PRIMARY KEY (pet_id)
);
```

Screenshot #2:

INSERT INTO person (person_id, first_name, last_name, phone_number, birth_date)

VALUES

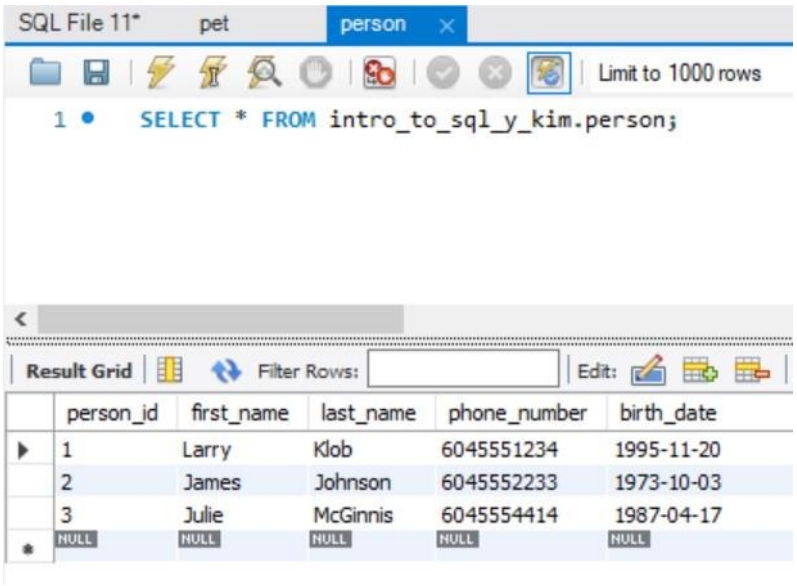
(1, 'Larry', 'Klob', '6045551234', '1995-11-20'),

(2, 'James', 'Johnson', '6045552233', '1973-10-03'),

(3, 'Julie', 'McGinnis', '6045554414', '1987-04-17');

This commend allowed to insert the value in columns in person table.

Filename: 02_Select_From_Person.png



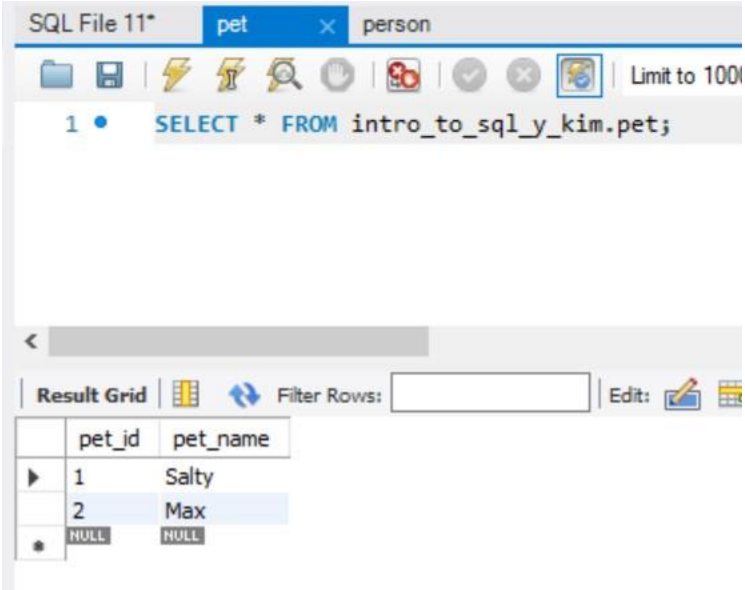
Screenshot #3:

The SQL SELECT command is used to retrieve data from one or more tables in a database. It does not select the table itself but rather selects specific columns or rows from the table.

SELECT column1, column2 FROM your_table;

*SELECT * FROM your_table;*

Filename: 03_Select_From_Pet.jpg



Screenshot #4:

This is the file I export the pet and person tables as a backup.

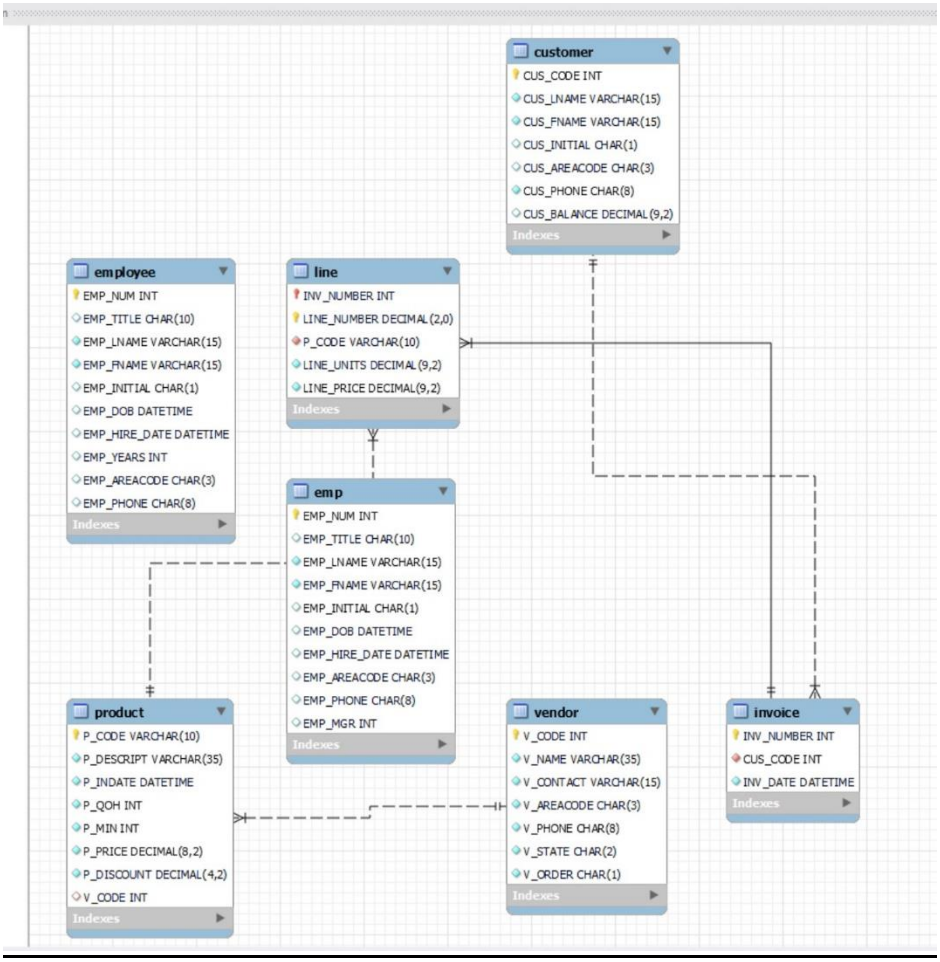
Filename: 04_Person and Pet Backup (Logical).sql

04_Person and Pet Backup (Logica

Screenshot #5:

This is after import the sql file to Mysql workbench and show the ERD.

Filename: 05_SaleCo_ERD.jpg



Screenshot #6:

Screenshot 6 to 19, each representing different methods to retrieve data from an imported file, and you want to comment on the SQL SELECT statements with specific column names for displaying the data as a table.

Filename: 06_Scenario_2.jpg

```
1 •  SELECT v_name, v_areacode, v_phone, v_state
2    FROM vendor;
```

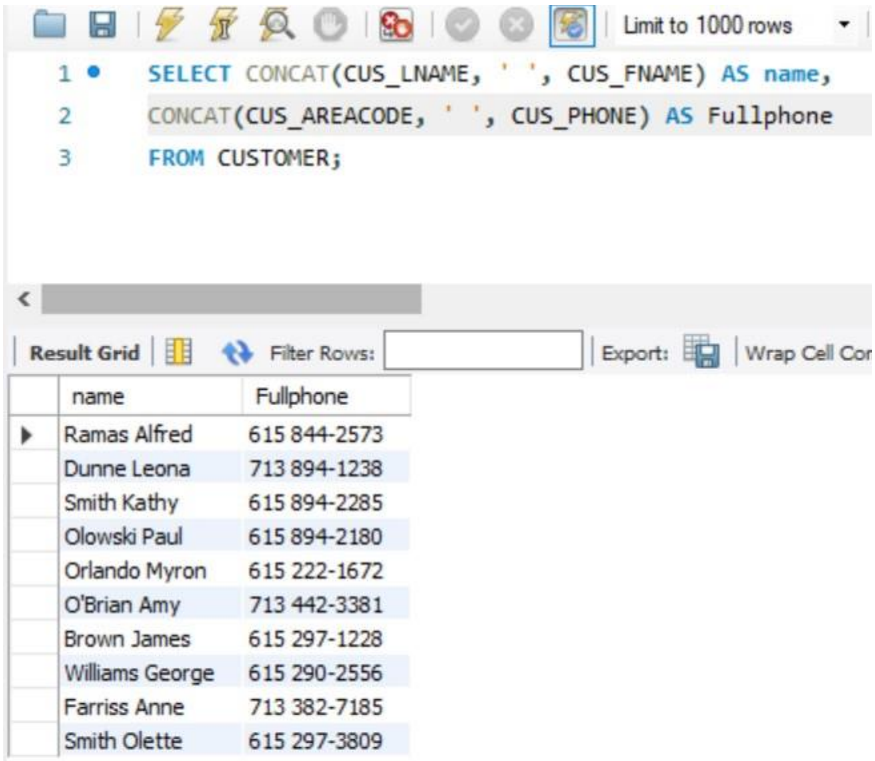
Result Grid | Filter Rows: | Export:

v_name	v_areacode	v_phone	v_state
Bryson, Inc.	615	223-3234	TN
SuperLoo, Inc.	904	215-8995	FL
D&E Supply	615	228-3245	TN
Gomez Bros.	615	889-2546	KY
Dome Supply	901	678-1419	GA
Randsets Ltd.	901	678-3998	GA
Brackman Bros.	615	228-1410	TN
ORDVA, Inc.	615	898-1234	TN
B&K, Inc.	904	227-0093	FL
Damal Supplies	615	890-3529	TN
Rubicon Systems	904	456-0092	FL

Screenshot #7:

The CONCAT function is used to combine multiple columns into a unified column. This results in a single column that incorporates the values from the specified columns.

Filename: 07_Scenario_3.jpg



name	Fullphone
Ramas Alfred	615 844-2573
Dunne Leona	713 894-1238
Smith Kathy	615 894-2285
Olowski Paul	615 894-2180
Orlando Myron	615 222-1672
O'Brian Amy	713 442-3381
Brown James	615 297-1228
Williams George	615 290-2556
Farriss Anne	713 382-7185
Smith Olette	615 297-3809

Screenshot #8:

It also has the capability to sort the data according to your preference. The default order is ascending. DESC means decending.

Filename: 08_Scenario_4.jpg

```
SELECT p_CODE, p_descript, p_price
FROM product
ORDER BY p_price DESC;
```

Result Grid | Filter Rows: | Export:

p_CODE	p_descript	p_price
89-WRE-Q	Hicut chain saw, 16 in.	256.99
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	119.95
11QER/31	Power painter, 15 psi., 3-nozzle	109.99
2232/QTY	B&D jigsaw, 12-in. blade	109.92
2232/QWE	B&D jigsaw, 8-in. blade	99.87
1558-QW1	Hrd. cloth, 1/2-in., 3x50	43.99
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	39.95
2238/QPD	B&D cordless drill, 1/2-in.	38.95
14-Q1/L3	9.00-in. pwr. saw blade	17.49
13-Q2/P2	7.25-in. pwr. saw blade	14.99
23114-AA	Sledge hammer, 12 lb.	14.40
23109-HB	Claw hammer	9.95
SW-23116	2.5-in. wd. screw, 50	8.45
SM-18277	1.25-in. metal screw, 25	6.99
PVC23DRT	PVC pipe, 3.5-in., 8-ft	5.87
54778-2T	Rat-tail file, 1/8-in. fine	4.99

Screenshot #9:

I selected employee first names, initials, and employee last names to create a table, and sorted them in ascending order. The default in MySQL is ascending order, but I specified 'ASC' to organize them in ascending order explicitly.

Filename: 09_Scenario_5.jpg

```
1 •  SELECt emp_FNAME, emp_INITIAL, Emp_LNAME
2    FROM employee
3    ORDER BY emp_FNAME, Emp_LNAME asc;
```

Result Grid | Filter Rows: | Export:

emp_FNAME	emp_INITIAL	Emp_LNAME
Anne	M	Jones
Edward	E	Johnson
George	D	Kolmycz
George	K	Smith
George	A	Smith
Hermine	R	Saranda
Jeanine	K	Smith
John	P	Lange
Jorge	D	Diante
Leighla	W	Genkazi
Marie	G	Brandon
Melanie	P	Smythe
Paul	R	Wiesenbach
Rhett	NULL	Vandam
Rhonda	G	Lewis
Robert	D	Williams
Rupert	E	Washington

Screenshot #10:

I selected product code, product description, and product price from the product table, and instructed to sort the prices in descending order. Using the LIMIT, I displayed the top three products.

Filename: 10_Scenario_6.jpg

```
SELECT P_code, P_descript, P_price
FROM product
ORDER BY p_price DESC
LIMIT 3;
```

Result Grid | Filter Rows: | Export:

P_code	P_descript	P_price
89-WRE-Q	Hicut chain saw, 16 in.	256.99
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	119.95
11QER/31	Power painter, 15 psi., 3-nozzle	109.99

Screenshot #11:

Using WHERE allows for detailed filtering based on a specific column. By employing LIKE '% %', you can extract table values that include the specified pattern

Filename: 11_Scenario_7A.jpg

```
SELECT P_CODE, P_DESCRIPT, P_PRICE, P_DISCOUNT, V_CODE
FROM PRODUCT
WHERE P_DESCRIPT LIKE '%hammer%';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

P_CODE	P_DESCRIPT	P_PRICE	P_DISCOUNT	V_CODE
23109-HB	Claw hammer	9.95	0.10	21225
23114-AA	Sledge hammer, 12 lb.	14.40	0.05	NULL

Screenshot #12:

This is an additional item in the screenshot labeled 11. 'IS NOT NULL' indicates a table with existing values. It displays only the products where the V_code is present, and the description includes the term 'hammer.

Filename: 12_Scenario_7B.jpg

```
SELECT P_CODE, P_DESCRIPT, P_PRICE, P_DISCOUNT, V_CODE
FROM PRODUCT
WHERE P_DESCRIPT LIKE '%hammer%' and v_code is not null;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

P_CODE	P_DESCRIPT	P_PRICE	P_DISCOUNT	V_CODE
23109-HB	Claw hammer	9.95	0.10	21225

Screenshot #13:

The utilization of the IN operator showcases the table that includes the specified values. It effectively filters and presents the relevant data based on the specified criteria.

Filename: 13_Scenario_7C.jpg

```
SELECT EMP_TITLE, EMP_LNAME, EMP_FNAME
FROM EMPLOYEE
WHERE EMP_TITLE IN ('Ms.','Mrs.');
```

Result Grid | Filter Rows: | Export:

EMP_TITLE	EMP_LNAME	EMP_FNAME
Ms.	Lewis	Rhonda
Ms.	Jones	Anne
Mrs.	Smith	Jeanine
Mrs.	Genkazi	Leighla
Ms.	Smythe	Melanie
Ms.	Brandon	Marie
Mrs.	Saranda	Hermine

Screenshot #14:

The BETWEEN operator defines a range of values by specifying a minimum and maximum. It displays values within that range, inclusive of both the minimum and maximum values.

Filename: 14_Scenario_7D.jpg

```
SELECT P_CODE, P_DESCRIPT, P_PRICE, V_CODE
FROM PRODUCT
WHERE P_PRICE BETWEEN '20' AND '50'
```

Result Grid | Filter Rows: | Export: | Wr

P_CODE	P_DESCRIPT	P_PRICE	V_CODE
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	39.95	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	43.99	23119
2238/QPD	B&D cordless drill, 1/2-in.	38.95	25595

Screenshot #15:

Alternatively, assigning the minimum and maximum values as new column values allows the creation of a new table. It looks advantageous when dealing with extensive datasets.

Filename: 15_Scenario_8A.jpg

```
SELECT min(cus_code) AS "Fisrt Customer Code" ,
MAX(cus_code) AS "Last Customer Code"
FROM customer
```

Result Grid | Filter Rows: | Export: | Wrap C

Fisrt Customer Code	Last Customer Code
10010	10019

Screenshot #16:

COUNT reveals the quantity of specific values. In this example, it counts the number of products where the V_code is present.

Filename: 16_Scenario_8B.jpg

```
SELECT COUNT(*)
FROM product
WHERE v_code IS not NULL;
```

Result Grid | Filter Rows:

COUNT(*)
14

Screenshot #17:

Using GROUP BY, you can count the occurrences of each value and present the counts for each group in individual rows.

Filename: 17_Scenario_9A.jpg

Limit to 1000 rows

```
SELECT cus_areacode, COUNT(*) AS 'Count by Areacode'
FROM customer
GROUP BY cus_areacode;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

cus_areacode	Count by Areacode
615	7
713	3

Screenshot #18:

Alternatively, counting based on titles is also possible. This can be particularly useful when dealing with a table that contains repeated titles.

Filename: 18_Scenario_9B.jpg

```
SELECT emp_title, COUNT(*)
FROM employee
GROUP BY emp_title;
```

Result Grid | Filter Rows:

emp_title	COUNT(*)
Mr.	10
Ms.	4
Mrs.	3

Screenshot #19:

In the final screenshot, the total price for each invoice was calculated, automatically identifying values exceeding $100. This proved to be a much more concise and visually appealing method compared to calculating in Excel.

Filename: 19_Scenario_10.jpg

Limit to 500 rows

```
SELECT inv_number, SUM(Line_units * Line_price) AS "Invoice Total"
FROM line
GROUP BY inv_number
HAVING SUM(LINE_PRICE * LINE_UNITS) >= 100;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

inv_number	Invoice Total
1003	153.8500
1006	397.8300
1008	399.1500

Screenshot #20:

This txt includes my Render site URL.

20_Render_Web_
Users_Site.txt

Filename: 20_Render_Web_Users_Site.txt

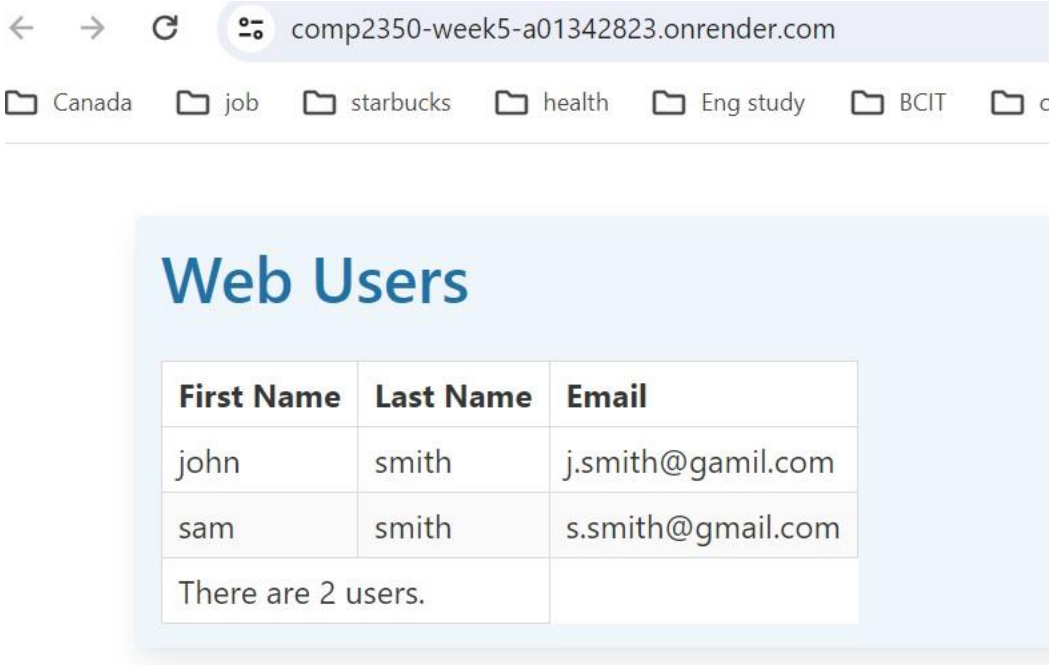
Screenshot #21:

I have modified all of my code to cater to web users transitioning from a to-do list

21_Qoddi_Web_Users_Code.zip

Filename: 21_Qoddi_Web_Users_Code.zip

Screenshot #22:

After changed to web users, I created a table in MySQL. This allows me to automatically update my contents and send them to the Render server.

Filename: 22_Qoddi_Web_Users_Screenshot.jpg