FRANCHISE_DB

- Mason Sayyadi

Connection Info:
server: ix-dev.cs.uoregon.edu
user: guest
password: guest
port: 3585

TABLE OF CONTENTS:

GENERAL

PROJECT LINK	2
INTRODUCTION	2
TABLE SUMMARY	3
APPLICATION INFORMATION	4

TABLES

ORDER_ITEMS	6
ORDERS	7
CUSTOMERS	8
EMPLOYEES	9
FRANCHISE	10
MENU_DETAILS	11
CREDIT_CARD	12

DIAGRAMS/CODE

DATABASE DIAGRAM	13
CODE	14-26
CONCLUSION	26

Link to my project:

https://ix.cs.uoregon.edu/~msayyadi/findData.html

Introductory Summary:

For my database, I decided to make a database of a fictitious fast food franchise. The following database if given 1 week's worth of data information from a "failing" franchise. The "Orders table is one of the main tables that all of the other tables revolve around. Each order has a "Customers" table, where a customer has a first name, last name, credit card number, and an order number which is the primary key that links the Customers table to the Orders table. In addition, Customers, and Orders both share a common variable "card_number" which is referring to a credit card in which the credit card is it's own table that has security details such as a bank, and card number as the primary key that relates it to the Orders table. Each order has an employee that takes the order, and an employees/ customers can have multiple orders that they work on/have. An employee has a first name, last name, age, position, franchise id, manager_id, weekly hours, and salary. An employee's position can be a manager, in which the manager id and employee id would be the same. Multiple items can be a part of a single order, in which the order_id's (order_id is the primary key) of each of the items match that of the same order. This is all done in the item table. Items have a price, a menu_code, and an item_name. In order to get more details (such as description of the items), you would have to look at the Item_details table, which is linked to the items table and contains the description and nutrition code. Menu code is the primary key to this "Item_details" table. The nutrition code of this table is linked to the nutrition table that is linked with the primary key (nutrition code). The nutritions table has all of the calories and health information of the orders.

TABLE SUMMARY

Table: EMPLOYEE

Description: The employee table contains a primary key “employee_id” that is used to link the employees with people’s orders. The employees have a first name (efname), last name (elname), and a franchise code (franchise_code) that can be linked up with the franchise table to give us information. It also has the employees manager ID (manager_id), the number of hours the employee works (weekly_hours), and the employee salary (salary).

Table: ORDERS

Description: The orders table has a primary key (order_id) that gives the unique and specific order information of a particular order. The order number (order_num) is something that resets every week, but is the number that links to order to the customer. Orders also have a code (franchise_code) to lookup the franchise information. Orders also have an employee (employee_id) that gives us server information. Orders also have a card code that links to the CREDIT_CARD table to give us card information.

Table: ORDER_ITEM

Description: The table that contains the primary key “order _id” as the number that links it to the individual orders. The ORDER_ITEM also contains an item code key (item_code) that will give us information on the item. “order_date” and “order_time” are timestamp information that are on each order.

Table: CUSTOMERS

Description: Primary key of “order_num” that links the order to the customer. Each customer has a first name (fname), a last name (lname), a credit card number (card_num), and a credit card code (card_code) that links the credit card to information stored in the bank.

Table: MENU_DETAILS

Description: Primary key is (item_code) that links the menu items with its orders. Each item has a name (item_name), a description(item_desc), and a price (price). In addition, the nutrition information is also stored in this database. Information includes calories (calories), sugar (sugar), and fats (fats).

Table: CREDIT_CARD

Description: Primary key is (card_code) that is linked to a person's credit card to give information about that person. Credit card includes detailed banking information of a cardholder. A credit card has a column for a person's citizenship (country), their credit score (min_credit), and the bank that they are banking from (bank).

Table: FRANCHISE

Description: Primary key is (franchise_code) that is linked to an employee/manager/order to give us information about the location and unique franchise. Each franchise has a location (fr_location) a manager identified by their ID (manager_id), a seating capacity (capacity), an opening time (open_time), and a closing time (close_time).

APPLICATION INFORMATION

Application 1:

This application takes a menu item as its parameter, and will give you a description of that item. The menu is given above the text box, and it's will print the query that was used for the database as well as the results.

Application 2:

This application takes a letter (A-Z) and will return the first names of all people who have their first name start with that letter. It will then show you the total amount of money they have spent at the franchise. The app prints the query that was used for the database as well as the results.

Application 3:

This application takes an employee's first name (Either Mikayla, Alyssa, Olivia, or Jee are acceptable) and it returns a list of customers (first name and last name) who have been served by the specified worker. The app prints the query that was used for the database as well as the results.

Application 4:

This application takes a number between 300-700 and will show you all of the customer's who have credit that's above the specified number. It will display their first and last name. The app prints the query that was used for the database as well as the results.

Application 5:

This application, just like application 1, takes a menu item as an argument, and displays its calorie, sugar, and fat information. The app prints the query that was used for the database as well as the results.

User's Guide:

You type into the text box whatever you are prompted to type in (typically above the text box), and you click the "submit" button.

Tables (Complete):

ORDER_ITEMS

order_id	item_code	order_date	order_time
331456	FRI	2015-02-09	12:43:13
229140	TOT	2015-02-09	12:45:27
983758	DBR	2015-02-09	14:13:07
674839	CFE	2015-02-10	10:55:29
220194	SUN	2015-02-10	11:34:55
528491	NDR	2015-02-10	12:05:44
777494	NDR	2015-02-10	12:09:51
195832	CSW	2015-02-10	13:41:21
405395	JCE	2015-02-11	10:00:53
884893	SBR	2015-02-11	12:39:11
559630	CFE	2015-02-11	12:45:42
273902	CFE	2015-02-12	13:15:17
333953	DBR	2015-02-12	13:18:18
902193	CFE	2015-02-12	15:28:31
774632	TOT	2015-02-12	15:30:31
222224	NDR	2015-02-12	15:33:33
189382	CFE	2015-02-13	10:29:02
293827	DDR	2015-02-13	10:59:15
569382	SBR	2015-02-13	11:03:23
442811	APP	2015-02-13	11:43:56
331456	FRI	2015-02-09	12:43:13
331456	SBR	2015-02-09	12:43:13
331456	NDR	2015-02-09	12:43:13
229140	TOT	2015-02-09	12:45:27
229140	NUG	2015-02-09	12:45:27
983758	DBR	2015-02-09	14:13:07
983758	SHK	2015-02-09	14:13:07
674839	CFE	2015-02-10	10:55:29
220194	SUN	2015-02-10	11:34:55
528491	NDR	2015-02-10	12:05:44
528491	SBR	2015-02-10	12:05:44
528491	CSW	2015-02-10	12:05:44
777494	NDR	2015-02-10	12:09:51
777494	DBR	2015-02-10	12:09:51
195832	CSW	2015-02-10	13:41:21
195832	ICE	2015-02-10	13:41:21
405395	JCE	2015-02-11	10:00:53
405395	APP	2015-02-11	10:00:53
405395	DDR	2015-02-11	10:00:53
884893	SBR	2015-02-11	12:39:11
884893	CSW	2015-02-11	12:39:11
559630	CFE	2015-02-11	12:45:42
559630	SBR	2015-02-11	12:45:42
559630	SHK	2015-02-11	12:45:42
273902	CFE	2015-02-12	13:15:17
333953	DBR	2015-02-12	13:18:18
333953	CSW	2015-02-12	13:18:18
902193	CFE	2015-02-12	15:28:31
774632	TOT	2015-02-12	15:30:31
222224	NDR	2015-02-12	15:33:33
222224	TOT	2015-02-12	15:33:33
189382	CFE	2015-02-13	10:29:02
293827	DDR	2015-02-13	10:59:15
293827	TOT	2015-02-13	10:59:15
293827	APP	2015-02-13	10:59:15
569382	SBR	2015-02-13	11:03:23
569382	SHK	2015-02-13	11:03:23
442811	APP	2015-02-13	11:43:56

ORDERS

order_id	order_num	franchise_code	employee_id	card_code
331456	1	AA	101	USB
229140	2	AA	101	MBX
983758	3	AA	101	CHA
674839	4	AA	103	USB
220194	5	AA	103	BCU
528491	6	AA	111	NFC
777494	7	AA	102	BOA
195832	8	AA	103	BCU
405395	9	AA	102	UMP
884893	10	AA	102	CHA
559630	11	AA	101	BOA
273902	12	AA	103	UMP
333953	13	AA	102	USB
902193	14	AA	102	CHA
774632	15	AA	101	BCU
222224	16	AA	111	BOA
189382	17	AA	102	USB
293827	18	AA	102	CHA
569382	19	AA	103	NFC
442811	20	AA	103	NFC

CUSTOMERS

order_num	fname	lname	card_num	card_code
1	Mason	Sayyadi	7087-8639-0857-4563	USB
2	Alex	Guevara	1238-4564-4535-9767	MBX
3	Jordan	Fraser	8990-6573-3333-5674	CHA
4	Nima	Talebi	3945-2222-4564-6786	USB
5	Alexey	Avodayev	7807-4564-5654-6543	BCU
6	Lucas	Hyatt	4646-6767-2342-8908	NFC
7	Azure	Woodenlegs	9878-5835-7564-2231	BOA
8	Cody	Jacobson	9085-2342-5642-2343	BCU
9	Parsa	Bagheri	3453-2324-4568-5664	UMP
10	James	Kang	5277-6667-9646-4566	CHA
11	Nicholas	Fay	4938-5664-3346-2343	BOA
12	Owen	Kendricks	8700-4256-2318-4234	UMP
13	Brett	Favre	1346-3456-8934-5932	USB
14	Gabrielle	Tor	6811-4592-3405-0365	CHA
15	Olivia	Wix	8355-4579-3447-2453	BCU
16	Downs	Spitler	2746-5673-0786-4565	BOA
17	Logan	Sayyadi	1632-9785-4343-5677	USB
18	Arman	Sayyadi	8954-2651-4235-9986	CHA
19	Arianna	Sayyadi	7867-3324-9890-1325	NFC
20	Garrett	VanSickle	4456-2318-9578-6548	NFC

EMPLOYEE

employee_id	efname	elname	franchise_code	manager_id	weekly_hours	salary
101	Olivia	Pannell	AA	111	40	30000
102	Alyssa	Huque	AA	111	50	45000
103	Mikayla	Campbell	AA	111	40	30000
111	Jee	Choi	AA	111	60	75000
201	Bob	Barger	AB	222	40	30000
202	John	Candy	AB	222	40	30000
203	Jimmy	Smith	AB	222	40	30000
222	Cheryl	Lynn	AB	222	60	75000
301	Cam	Jordan	AC	333	40	30000
302	Mike	Evans	AC	333	40	30000
303	Kyler	Murray	AC	333	40	30000
333	Aaron	Rodgers	AC	333	60	75000
401	Drew	Brees	AD	444	50	45000
402	Alvin	Kamara	AD	444	40	30000
444	Aaron	Jones	AD	444	60	75000
501	Danny	Devito	AE	555	40	30000
502	George	John	AE	555	40	30000
555	Julia	Roberts	AE	555	60	75000
602	Adam	Sandler	AF	666	40	30000
666	Satan	Himself	AF	666	60	75000

FRANCHISE

franchise_code	fr_location	manager_id	capacity	open_time	close_time
AA	Penny Road	111	65	07:00:00	21:00:00
AB	Oak Street	222	40	08:00:00	21:00:00
AC	West Town	333	55	08:00:00	21:00:00
AD	East Bay	444	25	09:00:00	21:30:00
AE	Peterson Way	555	40	09:00:00	21:30:00
AF	Lake City	666	20	09:00:00	21:30:00

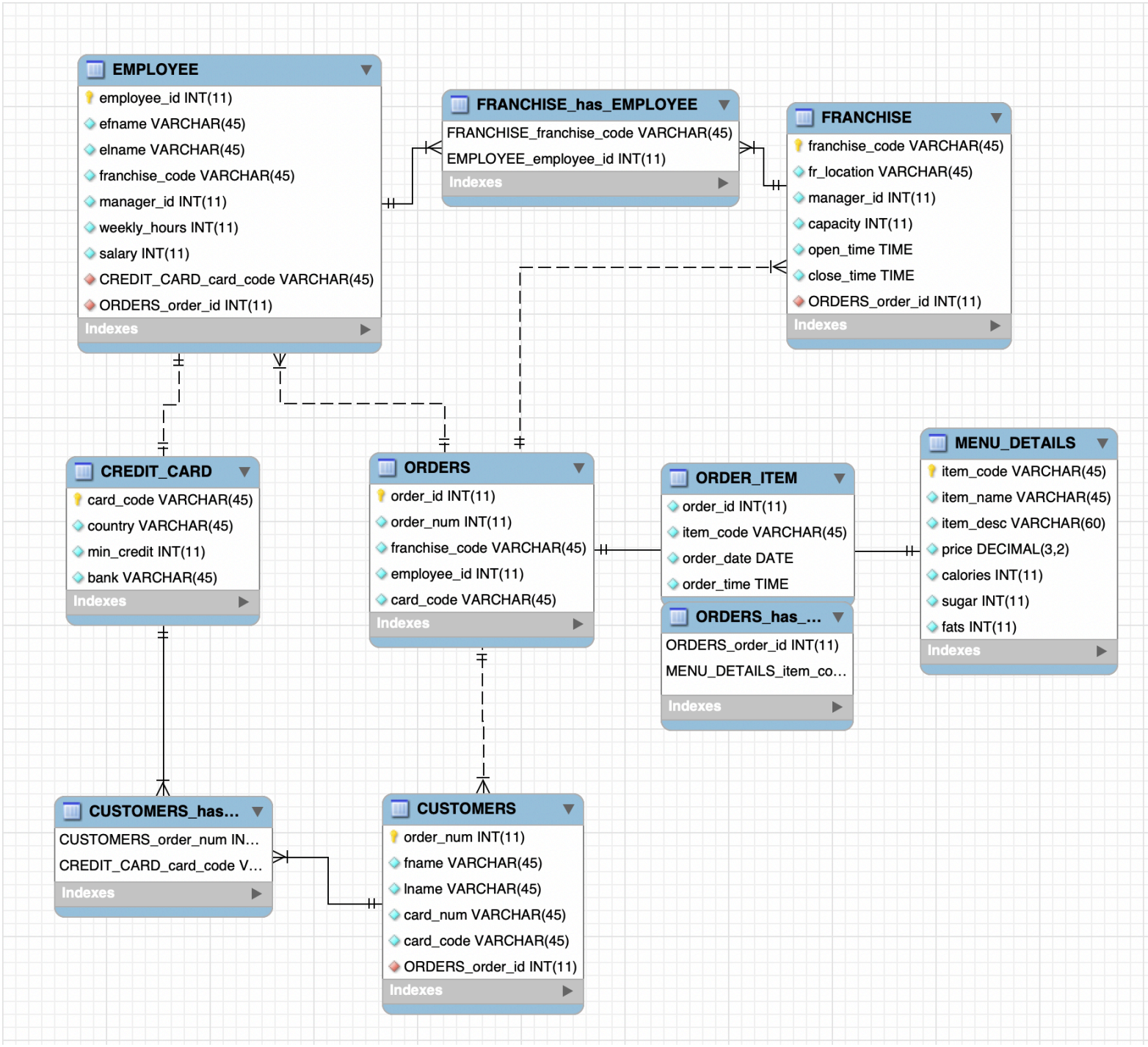
MENU_DETAILS

item_code	item_name	item_desc	price	calories	sugar	fats
JCE	Juice	"Drinkable Fruit Juice"	2.00	136	30	0
CFE	Coffee	"Caffeinated Beverage"	3.00	1	0	0
NDR	Drink	"Fountain Drink Soda"	2.00	150	56	0
DDR	Diet Drink	"Diet Fountain Drink Soda"	2.00	0	0	0
FRI	French Fries	"Deep Fried Potatoes"	3.00	365	10	5
TOT	Tater Tots	"Deep Fried Shredded Potatoes"	3.00	243	10	5
APP	Apple	"A Crunchy Red Fruit"	2.00	95	12	0
NUG	Nuggets	"Fried Chicken Pieces"	5.00	360	30	9
SBR	Cheeseburger	"One Patty Cheeseburger"	6.00	400	40	12
DBR	Double Cheeseburger	"Two Patty Cheeseburger"	7.00	600	40	14
CSW	Chicken Sandwich	"Sandwich of Chicken"	6.00	425	50	11
SHK	Shake	"Drinkable Ice Cream"	4.00	437	145	18
ICE	Ice Cream	"Thick Frozen Cream"	4.00	342	120	20
SUN	Sundae	"Frozen Cream With Toppings"	6.00	484	216	25

CREDIT_CARD

card_code	country	min_credit	bank
USB	USA	500	US Bank
UMP	USA	600	Umpqua Bank
NFC	USA	750	Navy Credit Union
CHA	USA	700	Chase
BOA	USA	450	Bank Of America
BCU	USA	500	BECU
JPC	JAP	400	Japan Bank
CMX	CAN	575	Canada Credit Union
MBX	MEX	380	Bank of Mexico

DATABASE DIAGRAM



LINKS TO CODE

https://ix.cs.uoregon.edu/~msayyadi/sqlStores7.php

https://ix.cs.uoregon.edu/~msayyadi/app2.php

https://ix.cs.uoregon.edu/~msayyadi/app3.php

https://ix.cs.uoregon.edu/~msayyadi/app4.php

https://ix.cs.uoregon.edu/~msayyadi/app5.php

https://ix.cs.uoregon.edu/~msayyadi/sqlConn.php

HTML CODE:

```
<p>

<form action="sqlStores7.php" method="POST">

<input type="text" name="item_name"> <br>
<input type="submit" value="submit">
<input type="reset" value="erase">
</form>

<hr>


<hr>

<p>
Enter a letter ("A","B","C","M" are some that work)
to see the amount of money spent by people with
that letter as the first letter of their first name!

<p>

<form action="app2.php" method="POST">

<input type="text" name="fname"> <br>
<input type="submit" value="submit">
<input type="reset" value="erase">
</form>

<hr>
```

```
<hr>

<p>
Type in an Employee name to see all of the customers they've served!
Employees: Olivia, Alyssa, Mikayla, Jee

<p>

<form action="app3.php" method="POST">

<input type="text" name="efname"> <br>
<input type="submit" value="submit">
<input type="reset" value="erase">
</form>

<hr>


<hr>

<p>
View customers that have a credit score above _____ !
(Numbers between 300-700 will give good results)

<p>

<form action="app4.php" method="POST">

<input type="text" name="min_credit"> <br>
<input type="submit" value="submit">
<input type="reset" value="erase">
</form>

<hr>


<hr>

<p>
Enter the menu item's name to get its nutrition information!
(Refer to the menu at the top of the page)
<p>

<form action="app5.php" method="POST">

<input type="text" name="item_name"> <br>
<input type="submit" value="submit">
<input type="reset" value="erase">
</form>

<hr>
```

```
<p>
<a href="findData.html" >Contents</a>
of this page.

<p>
<a href="sqlStores7.php" >Application1</a>
<a href="app2.php" >Application2</a>
<a href="app3.php" >Appliaction3</a>
<a href="app4.php" >Application4</a>
<a href="app5.php" >Application5</a>
Names of the PHP pages that gets called.
(And the <a href="sqlConn.txt" >connection data</a>,
kept separately for security reasons.)


</body>
</html>
```

APP1:

```
<?php

    // include('connectionData.txt');
    include('sqlConn.txt')

    $conn = mysqli_connect($server, $user, $pass, $dbname, $port)
    or die('Error connecting to MySQL server.')

    ?>

<html>
<head>
<title>Application 1</title>
</head>

<body bgcolor="white">

<hr>
```

```
<?php

    $item_name = $_POST['item_name']

    $item_name = mysqli_real_escape_string($conn, $item_name)
    // this is a small attempt to avoid SQL injection
    // better to use prepared statements

    //$query = "SELECT DISTINCT item_desc FROM ORDER_ITEM o JOIN
MENU_DETAILS m ON o.item_code = m.item_code WHERE item_name = ";
    //$query = $query."'".$item_name."';";
    $query = "SELECT item_desc FROM MENU_DETAILS WHERE item_name = ";
    $query = $query."'".$item_name."';";
    //$query = $query."'".$item_name."';";
    ?>

<p>
The query:
<p>
<?php
    print $query
?>

<hr>

<p>
Information:
<p>

<?php
    // Accesses the connection and the query path
    $result = mysqli_query($conn, $query)
    or die(mysqli_error($conn)) // Terminate if connection fails

    print "<pre>"; //Print the tag
    //While there is data in eacg row, print
    while($row = mysqli_fetch_array($result, MYSQLI_BOTH))
    {
        print "\n";
        print "Item Description: $row[item_desc]";
    }
    // End tag print
    print "</pre>";

    mysqli_free_result($result) //unlink the mySQL data
```

```
    mysqli_close($conn);  // close mySQL connection

    ?>

<p>
<hr>

<p>
<a href="sqlStores7.php" >Contents</a>
of the PHP program that created this page.

</body>
</html>
```

APP 2:

```
<?php

    include('sqlConn.txt');

    $conn = mysqli_connect($server, $user, $pass, $dbname, $port)
    or die('Error connecting to MySQL server.');
```

```
    ?>

<html>
<head>
<title>Application 2</title>
</head>

<body bgcolor="white">

<hr>

<?php

    $fname = $_POST['fname'];

    //$fname = mysqli_real_escape_string($conn, $fname);

    $query = $query."SELECT fname, lname, SUM(price) AS sum_price FROM
franchise_db.ORDER_ITEM oi ";
    $query = $query."JOIN franchise_db.ORDERS o ON oi.order_id = o.order_id ";
    $query = $query."JOIN franchise_db.CUSTOMERS c ON o.order_num =
c.order_num ";
    $query = $query."JOIN franchise_db.MENU_DETAILS m ON oi.item_code =
m.item_code ";
    $query = $query."WHERE fname LIKE  '".$fname."%' ";
    $query = $query."GROUP BY o.order_num;";
    ?>

<p>
The query:
<p>
<?php
    print $query
?>

<hr>

<p>
Customers:
<p>

<?php
    // Accesses the connection and the query path
    $result = mysqli_query($conn  $query)
    or die(mysqli_error($conn))  // Terminate if connection fails
```

```
    print "<pre>"; //Print the tag

    while($row = mysqli_fetch_array($result, MYSQLI_BOTH))
    {
        print "\n";
        print "$row[fname] $row[lname] \t Amount Spent: $row[sum_price]";
    }
    // End tag print
    print "</pre>";

    mysqli_free_result($result); //unlink the mySQL data
    mysqli_close($conn); // close mySQL connection

    ?>

<p>
<hr>

<p>
<a href="app2.php" >Contents</a>
of the PHP program that created this page.

</body>
</html>
```

APP 3:

```
<?php

    // include('connectionData.txt');
    include('sqlConn.txt');

    $conn = mysqli_connect($server, $user, $pass, $dbname, $port)
    or die('Error connecting to MySQL server.');

    ?>

<html>
<head>
<title>Application 3</title>
</head>
```

```
<body bgcolor="white">

<hr>

<?php

   $efname = $_POST['efname'];

   $efname = mysqli_real_escape_string($conn  $efname)
   // this is a small attempt to avoid SQL injection
   // better to use prepared statements

   $query = "SELECT DISTINCT fname, lname FROM franchise_db.ORDER_ITEM oi ";
   $query = $query."JOIN franchise_db.ORDERS o ON oi.order_id = o.order_id ";
   $query = $query."JOIN franchise_db.CUSTOMERS c ON o.order_num =
c.order_num ";
   $query = $query."JOIN franchise_db.EMPLOYEE e ON e.employee_id =
o.employee_id WHERE efname = ";
   $query = $query."'".$efname."';";
   //$query = $query."'".$item_name."';";
   ?>

<p>
The query:
<p>
<?php
   print $query
?>

<hr>

<p>
Customers:
<p>

<?php
   // Accesses the connection and the query path
   $result = mysqli_query($conn  $query)
   or die(mysqli_error($conn))  // Terminate if connection fails

   print "<pre>"  //Print the tag
   //While there is data in eacg row, print
   while($row = mysqli_fetch_array($result  MYSQLI_BOTH))
   {
      print "\n";
```

```
        print "Customer: $row[fname] $row[lname]";
    }
    // End tag print
    print "</pre>";

    mysqli_free_result($result);  //unlink the mySQL data
    mysqli_close($conn);  // close mySQL connection

    ?>

<p>
<hr>

<p>
<a href="app3.php" >Contents</a>
of the PHP program that created this page.

</body>
</html>
```

APP 4:

```
<?php

    // include('connectionData.txt');
    include('sqlConn.txt');

    $conn = mysqli_connect($server, $user, $pass, $dbname, $port)
    or die('Error connecting to MySQL server.');

    ?>

<html>
<head>
<title>Application 4</title>
</head>

<body bgcolor="white">

<hr>
```

```
<?php

    $min_credit = $_POST['min_credit']

    //$min_credit = mysqli_real_escape_string($conn, $min_credit);
    // this is a small attempt to avoid SQL injection
    // better to use prepared statements

    $query = "SELECT fname, lname, min_credit FROM CUSTOMERS c ";
    $query = $query."JOIN CREDIT_CARD cc ON c.card_code = cc.card_code";
    $query = $query." WHERE min_credit >= '".$min_credit."';";
?>

<p>
The query:
<p>
<?php
    print $query
?>

<hr>

<p>
Credit Scores:
<p>

<?php
    // Accesses the connection and the query path
    $result = mysqli_query($conn, $query)
    or die(mysqli_error($conn)); // Terminate if connection fails

    print "<pre>"; //Print the tag
    //While there is data in eacg row, print
    while($row = mysqli_fetch_array($result, MYSQLI_BOTH))
    {
        print "\n";
        print "$row[fname] $row[lname] \t Credit Score: $row[min_credit]";
    }
    // End tag print
    print "</pre>";

    mysqli_free_result($result); //unlink the mySQL data
    mysqli_close($conn); // close mySQL connection

    ?>
```

```
<p>
<hr>

<p>
<a href="app4.php" >Contents</a>
of the PHP program that created this page.

</body>
</html>
```

APP 5

```
<?php

   // include('connectionData.txt');
   include('sqlConn.txt');

   $conn = mysqli_connect($server, $user, $pass, $dbname, $port)
   or die('Error connecting to MySQL server.');

   ?>

<html>
<head>
<title>Application 5</title>
</head>

<body bgcolor="white">
```

```
<hr>

<?php

    $item_name = $_POST['item_name']

    $item_name = mysqli_real_escape_string($conn, $item_name)
    // this is a small attempt to avoid SQL injection
    // better to use prepared statements

    $query = "SELECT item_name, calories, sugar, fats FROM MENU_DETAILS WHERE
item_name = '";
    $query = $query."'".$item_name."';";
    //$query = $query."'".$item_name."';";
    ?>

<p>
The query:
<p>
<?php
    print $query
?>

<hr>

<p>
Nutrition Facts:
<p>

<?php
    // Accesses the connection and the query path
    $result = mysqli_query($conn, $query)
    or die(mysqli_error($conn)); // Terminate if connection fails

    print "<pre>"; //Print the tag
    //While there is data in eacg row, print
    while($row = mysqli_fetch_array($result, MYSQLI_BOTH))
    {
        print "\n";
        print "$row[item_name] \t $row[calories] Calories \t $row[sugar]g Sugar \t
$row[fats]g Fat";
    }
    // End tag print
    print "</pre>";
```

```
    mysqli_free_result($result)  //unlink the mySQL data
    mysqli_close($conn)  // close mySQL connection

    ?>

<p>
<hr>

<p>
<a href="app5.php" >Contents</a>
of the PHP program that created this page.

</body>
</html>
```

CONCLUSION:

What I have done is simulated a fictitious fast food franchise that has encapsulated the overall connections between a customer and their order. Through several different tables that are running on my database, I was able to create an application that could access different types of information about the company and the people who support it. If I had more time, I'd love to add more data. It would be really cool to see different decades worth of data. I'd also create better applications that could perform actions such as "posting" to php pages. Overall, I thought this was a fun project, and I learned a LOT! Thanks for the quarter, prof. Wilson!