Guadalupe Alvarado

Project 2

11/1/22

CIS 3050-05

Fall 2022

Table of Contents

P1: Cover Page

P2: Table of Contents

P3: Student's Academic Honesty

P4: Introduction/Description and Requirements

P5: Query 1- Screen shots (query and results, explanations)

P6: Query 2- Screen shots (query and results, explanations)

P7: Query 3- Screen shots (query and results, explanations)

P8: Query 4- Screen shots (query and results, explanations)

P9: Query 5- Screen shots (query and results, explanations)

P10: Query 6- Screen shots (query and results, explanations)

P11: Query 7- Screen shots (query and results, explanations)

P12: Query 8- Screen shots (query and results, explanations)

P13: Query 9- Screen shots (query and results, explanations)

P14: Query 10- Screen shots (query and results, explanations)

P15: Query 11- Screen shots (query and results, explanations)

P16: Query 12- Screen shots (query and results, explanations)

P17: Query 13- Screen shots (query and results, explanations)

P18: Query 14- Screen shots (query and results, explanations)

P19: Query 15- Screen shots (query and results, explanations)

P20: Query 16- Screen shots (query and results, explanations)

P21: Query 17- Screen shots (query and results, explanations)

P22: Query 18- Screen shots (query and results, explanations)

P23: Query 19- Screen shots (query and results, explanations)

P24: Query 20- Screen shots (query and results, explanations)

P25: Query 21- Screen shots (query and results, explanations)

P26-27: Report Analysis, Results, & Discussion/ Lessons Learned / Conclusion

P28: References

Student's Academic Honesty

My name is: Guadalupe Alvarado, I declare that, except where fully referenced no aspect of this project has been copied from any other source. I understand that any act of Academic Dishonesty such as plagiarism or collusion may result in serious offense and punishments. I promise not to lie about my academic work, to cheat, or to steal the words or ideas of others, nor will I help fellow students to violate the Code of Academic Honesty.

Name: Guadalupe Alvarado

Date: 11/1/22

Signature: 

Introduction

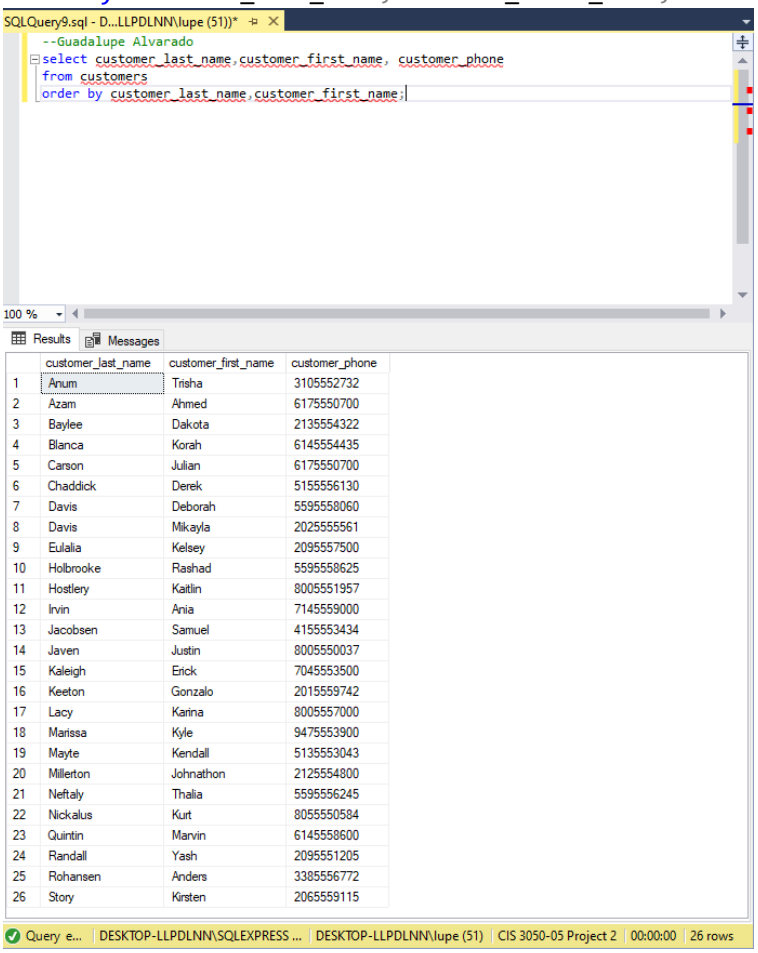
In Project #2, I'm designing, developing, and demonstrating the functionality of a database created based on a set of business specifications. I'm creating logical and physical data models by using Microsoft SQL Server 2019, MS SQL Server Management Tool, and the class textbook. The tables will be updated, deleted, and populated using script files. The logical and physical schema's relationships will be shown using cardinalities. The goal of this project is to create a detailed and correct database management system.

Project Description and Requirements

The project is designed to introduce the various aspects of the SQL SELECT statement and the methods of retrieving data from the database tables. My database system should be designed to perform general information management tasks such as systematic collection, update, and retrieval of information for a small organization. The aim of Project#2 is to utilize a set of tables that are represented by the ERD and are created and populated by the script file. A customers, artists, items, employees, orders, and order_details tables will be created. Information provided by the supplier(instructor) will be inserted to the various tables previously mentioned. The important data fields are the names of the customers, artists, items, employees- the Id's of orders, and order_details. In addition to these important requirements for tables, I'm provided with several business rules. These rules will range from different entity types, and if its null or not.

Query #1: Write a query that displays a list of all customers showing the customer's ***last name***, ***first name,*** and ***phone number***. Sort the results by customer last name, then first name.

```
select customer_last_name,customer_first_name, customer_phone
from customers
order by customer_last_name,customer_first_name;
```

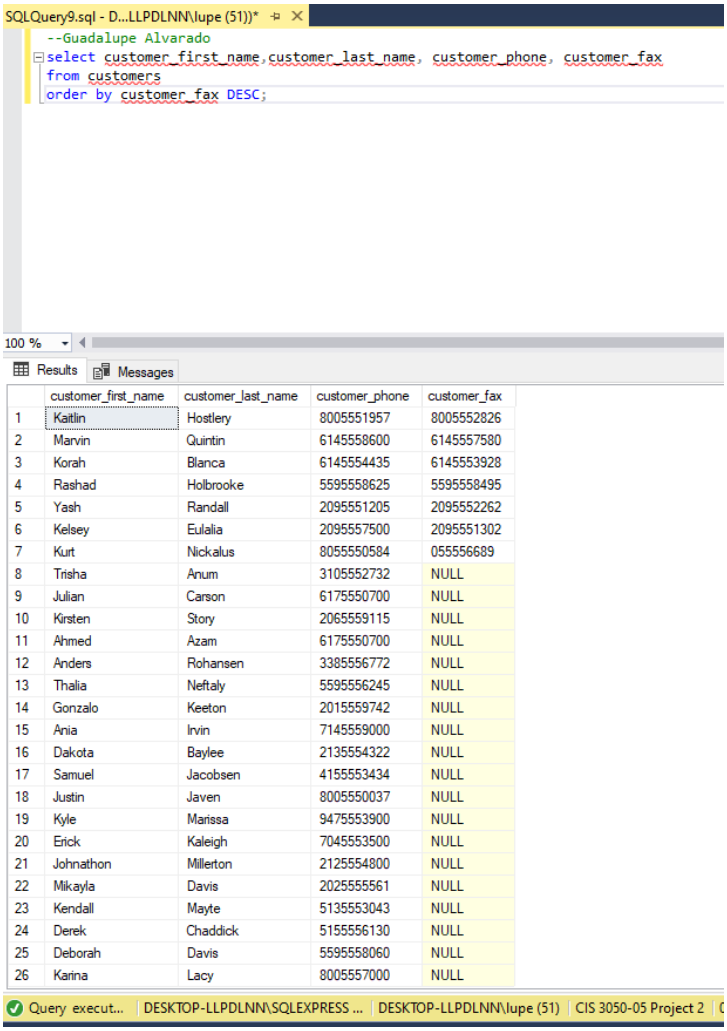


	customer_last_name	customer_first_name	customer_phone
1	Anum	Trisha	3105552732
2	Azam	Ahmed	6175550700
3	Baylee	Dakota	2135554322
4	Blanca	Korah	6145554435
5	Carson	Julian	6175550700
6	Chaddick	Derek	5155556130
7	Davis	Deborah	5595558060
8	Davis	Mikayla	2025555561
9	Eulalia	Kelsey	2095557500
10	Holbrooke	Rashad	5595558625
11	Hostlery	Kaitlin	8005551957
12	Irvin	Ania	7145559000
13	Jacobsen	Samuel	4155553434
14	Javen	Justin	8005550037
15	Kaleigh	Erick	7045553500
16	Keeton	Gonzalo	2015559742
17	Lacy	Karina	8005557000
18	Marissa	Kyle	9475553900
19	Mayte	Kendall	5135553043
20	Millerton	Johnathon	2125554800
21	Neftaly	Thalia	5595556245
22	Nickalus	Kurt	8055550584
23	Quintin	Marvin	6145558600
24	Randall	Yash	2095551205
25	Rohansen	Anders	3385556772
26	Story	Kirsten	2065559115

Explanation: Selected three fields(columns) from the customers table to show the Customer's last name, first name and phone number. The columns were ordered by last name then first name.

Query #2: Write a query that displays a list of all customers showing the customer's ***first name***, ***last name,*** phone number and fax. Sort the results by customer fax number in a descending order.

```
select customer_first_name,customer_last_name, customer_phone, customer_fax
from customers
order by customer_fax DESC;
```

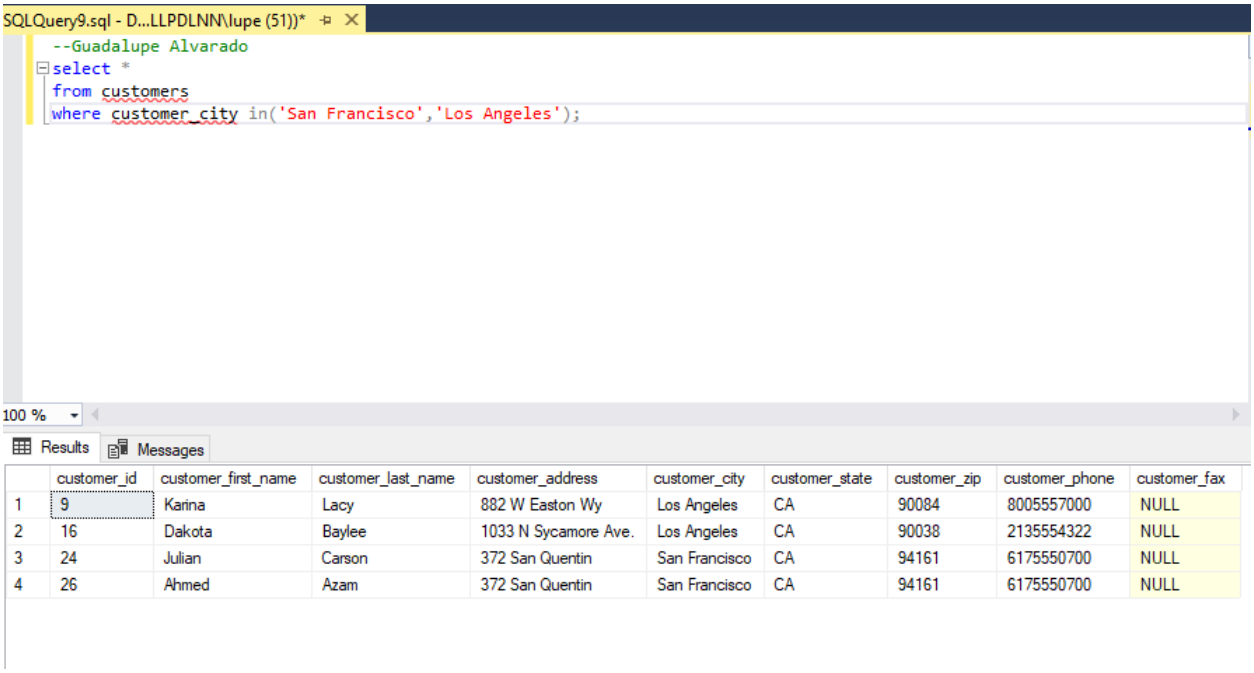


	customer_first_name	customer_last_name	customer_phone	customer_fax
1	Kaitlin	Hostlery	8005551957	8005552826
2	Marvin	Quintin	6145558600	6145557580
3	Korah	Blanca	6145554435	6145553928
4	Rashad	Holbrooke	5595558625	5595558495
5	Yash	Randall	2095551205	2095552262
6	Kelsey	Eulalia	2095557500	2095551302
7	Kurt	Nickalus	8055550584	055556689
8	Trisha	Anum	3105552732	NULL
9	Julian	Carson	6175550700	NULL
10	Kirsten	Story	2065559115	NULL
11	Ahmed	Azam	6175550700	NULL
12	Anders	Rohansen	3385556772	NULL
13	Thalia	Neftaly	5595556245	NULL
14	Gonzalo	Keeton	2015559742	NULL
15	Ania	Irvin	7145559000	NULL
16	Dakota	Baylee	2135554322	NULL
17	Samuel	Jacobsen	4155553434	NULL
18	Justin	Javen	8005550037	NULL
19	Kyle	Marissa	9475553900	NULL
20	Erick	Kaleigh	7045553500	NULL
21	Johnathon	Millerton	2125554800	NULL
22	Mikayla	Davis	2025555561	NULL
23	Kendall	Mayte	5135553043	NULL
24	Derek	Chaddick	5155556130	NULL
25	Deborah	Davis	5595558060	NULL
26	Karina	Lacy	8005557000	NULL

Explanation: Selected four fields(columns) from the customers table to show the Customer's first name, last name, phone number, and fax number. The columns were ordered by fax number in descending order. More than half of the customers do not have fax numbers on file.

Query #3: Write a query that displays all the customers from San Francisco or Los Angeles in the “Customers” table.

```
select *
from customers
where customer_city in('San Francisco','Los Angeles');
```



	customer_id	customer_first_name	customer_last_name	customer_address	customer_city	customer_state	customer_zip	customer_phone	customer_fax
1	9	Karina	Lacy	882 W Easton Wy	Los Angeles	CA	90084	8005557000	NULL
2	16	Dakota	Baylee	1033 N Sycamore Ave.	Los Angeles	CA	90038	2135554322	NULL
3	24	Julian	Carson	372 San Quentin	San Francisco	CA	94161	6175550700	NULL
4	26	Ahmed	Azam	372 San Quentin	San Francisco	CA	94161	6175550700	NULL

Explanation: Selected all fields(columns) from the customers table to show all the Customers who are from San Francisco or Los Angeles.

Query #4: Write a query that displays all the customers from the state of California and live in San Francisco or Los Angeles.

```
select *
from customers
where customer_city in('San Francisco','Los Angeles');
```

SQLQuery9.sql - D...LLPDLNN\lupe (51))*

```
--Guadalupe Alvarado
select *
from customers
where customer_city in('San Francisco','Los Angeles');
```

100 %

Results | Messages

	customer_id	customer_first_name	customer_last_name	customer_address	customer_city	customer_state	customer_zip	customer_phone	customer_fax
1	9	Karina	Lacy	882 W Easton Wy	Los Angeles	CA	90084	8005557000	NULL
2	16	Dakota	Baylee	1033 N Sycamore Ave.	Los Angeles	CA	90038	2135554322	NULL
3	24	Julian	Carson	372 San Quentin	San Francisco	CA	94161	6175550700	NULL
4	26	Ahmed	Azam	372 San Quentin	San Francisco	CA	94161	6175550700	NULL

Explanation: Selected all fields(columns) from the customers table to show all the Customers who are from San Francisco or Los Angeles.

Query #5: Write a query that displays each customer name as a single field in the format "firstname lastname" with a heading of Customer, along with their phone number with a heading of Phone. Use the IN operator to only display customers in New York, New Jersey, or Washington D.C. Sort the results by phone number.

```
select concat(customer_first_name,customer_last_name) as "Customer" ,customer_phone as
"phone"
from customers
where customer_state in('NY', 'NJ', 'DC')
order by phone;
```

SQLQuery10.sql -...LLPDLNN\lupe (70))*

```
--Guadalupe Alvarado
select concat(customer_first_name,customer_last_name) as "Customer" ,customer_phone as "phone"
from customers
where customer_state in('NY', 'NJ', 'DC')
order by phone;
```

100 %

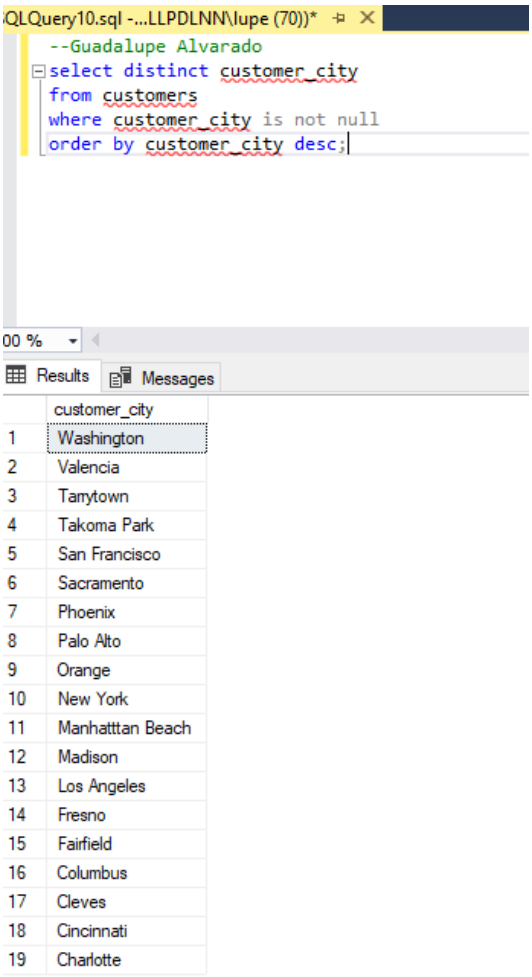
Results | Messages

	Customer	phone
1	GonzaloKeeton	2015559742
2	MikaylaDavis	2025555561
3	KirstenStory	2065559115
4	JohnathonMillerton	2125554800
5	JustinJaven	8005550037

Explanation: Added two strings together from the fields(columns) from the customers table to show the Customer's first name, last name as one field and also have the phone number. The columns were ordered by phone number.

Query #6: Write a query that will list all the cities that have customers with a heading of Cities. Only list each city once (no duplicates) and sort in descending alphabetical order.

```
select distinct customer_city
from customers
where customer_city is not null
order by customer_city desc;
```



	customer_city
1	Washington
2	Valencia
3	Tarrytown
4	Takoma Park
5	San Francisco
6	Sacramento
7	Phoenix
8	Palo Alto
9	Orange
10	New York
11	Manhatttan Beach
12	Madison
13	Los Angeles
14	Fresno
15	Fairfield
16	Columbus
17	Cleves
18	Cincinnati
19	Charlotte

Explanation: Selected distinct customer_city from the customers table to show all the states customers are in. Used selected distinct command to avoid duplicate states from the customers table records.

Query #7: Write a query that displays the title of each item along with the price (with a heading of Original) and a calculated field reflecting the price with a 15% discount (with a heading of Sale). Display the sale price with two decimal places using the ROUND function. Sort by price from highest to lowest.

```
select title,unit_price as "Original" , round(0.15*unit_price, 2) as "Sale"
from items
order by unit_price desc;
```

SQLQuery10.sql -...LLPDLNN\lupe (70))*

```
--Guadalupe Alvarado
select title,unit_price as "Original" ,round(0.15*unit_price, 2) as "Sale"
from items
order by unit_price desc;
```

100 %

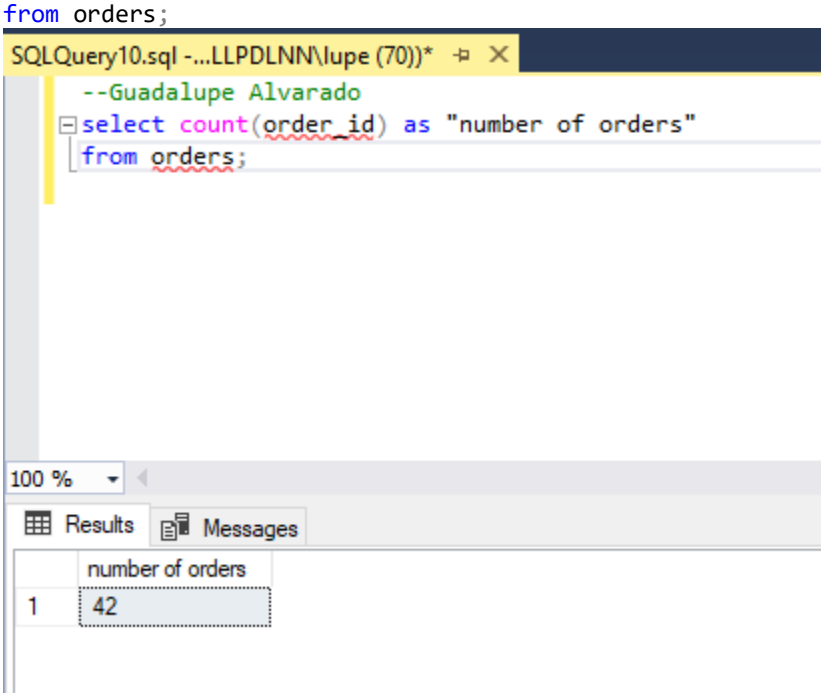
Results | Messages

	title	Original	Sale
1	More Songs About Structures and Comestibles	17.95	2.6900
2	Umami In Concert	17.95	2.6900
3	Burt Ruggles: An Intimate Portrait	17.95	2.6900
4	On The Road With Burt Ruggles	17.50	2.6300
5	Etcetera	17.00	2.5500
6	No Rest For The Weary	16.95	2.5400
7	No Fixed Address	16.95	2.5400
8	Zone Out With Umami	16.95	2.5400
9	Race Car Sounds	13.00	1.9500
10	Rude Noises	13.00	1.9500

Explanation: Selected title and unit_price from the items table and categorized the unit price as Original. Used the round function to create a Sale column with .15 multiplied by the unit_price-rounded to the nearest two decimals. Three columns in total.

Query #8: Write a query that displays the number of orders.

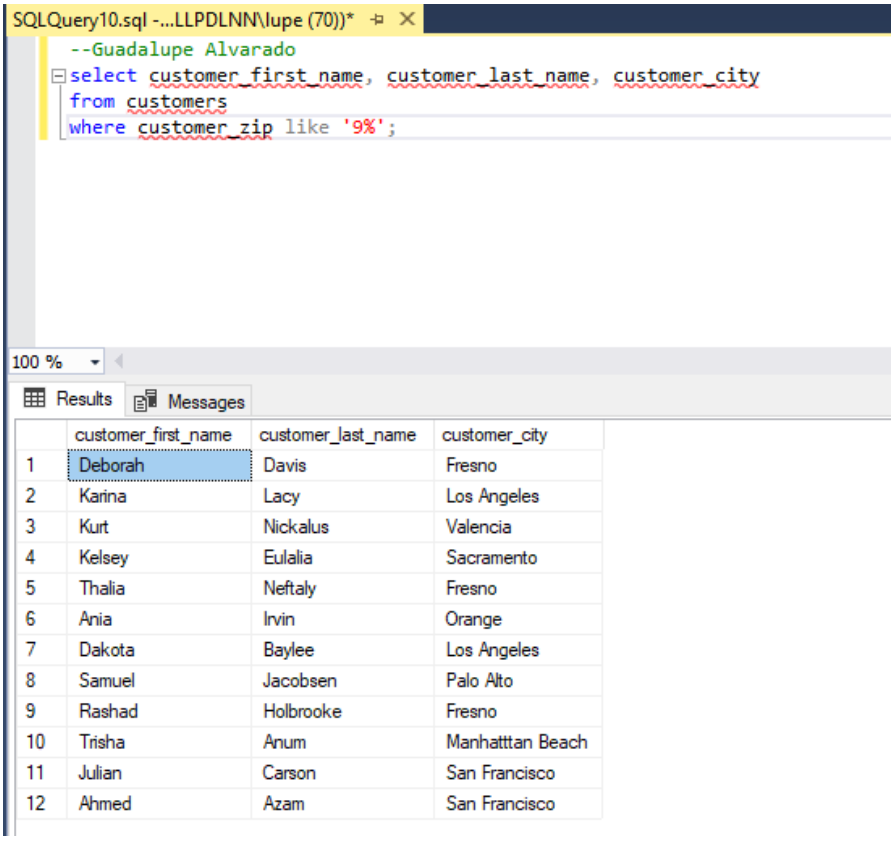
```
select count(order_id) as "number of orders"
from orders;
```



Explanation: Selected a count from the order table and labelled the column as number of orders. Its optional to choose order_id or order_date. Both options will produce the same result.

Query #9: Write a query that displays the customer city, first name, last name, and zip code from the customer's table. Use the LIKE operator to only display customers that reside in any zip code beginning with 9.

```
select customer_first_name, customer_last_name, customer_city
from customers
where customer_zip like '9%';
```



	customer_first_name	customer_last_name	customer_city
1	Deborah	Davis	Fresno
2	Karina	Lacy	Los Angeles
3	Kurt	Nickalus	Valencia
4	Kelsey	Eulalia	Sacramento
5	Thalia	Neftaly	Fresno
6	Ania	Irvin	Orange
7	Dakota	Baylee	Los Angeles
8	Samuel	Jacobsen	Palo Alto
9	Rashad	Holbrooke	Fresno
10	Trisha	Anum	Manhatttan Beach
11	Julian	Carson	San Francisco
12	Ahmed	Azam	San Francisco

Explanation: Selected customer's first name, last name, and city from the customer table- three columns in total. Used a condition where the only customers who showed were the ones who resided in a city zip code that starts with 9. The zip code column is not shown, but is used in the backend for the query.

Query #10: Write a query that displays the order id and order date for any orders placed from March 1, 2014 through April 30, 2014. Do this WITHOUT using the BETWEEN clauses. Format the date field as Month dd, yyyy and use a heading of "Ordered".

```
select order_id, FORMAT(order_date, 'MMMM dd, yyyy') as 'Ordered'
from Orders
where order_date >= 'March 1, 2014' and order_date <= 'April 30, 2014';
```

SQLQuery10.sql -...LLPDLNN\lupe (70))*

```
--Guadalupe Alvarado
select order_id, FORMAT(order_date, 'MMMM dd, yyyy') as 'Ordered'
from Orders
where order_date >= 'March 1, 2014' and order_date <= 'April 30, 2014';
```

100 %

Results | Messages

	order_id	Ordered
1	523	March 07, 2014
2	548	March 22, 2014
3	550	March 23, 2014
4	601	April 21, 2014
5	607	April 25, 2014

Explanation: Selected order id and order date but formatted order date as month, day, and year- also labelled it as Ordered. Order id and order date were pulled from the orders table. Order date had a set range from March 1 2014 – April 30 2014 using comparison operators.

Query #11: Write a query that displays the order id and order date for any orders placed during the month of May 2014. Do this using the BETWEEN clauses. Format the date field as mm/dd/yy and use a heading of "Ordered".

```
select order_id, FORMAT(order_date, 'MM/dd/yy') as 'Ordered'
from Orders
where order_date between '2014-05-01' and '2014-05-31';
```

SQLQuery10.sql -...LLPDLNN\lupe (70))*

```
--Guadalupe Alvarado
select order_id, FORMAT(order_date, 'MM/dd/yy') as 'Ordered'
from Orders
where order_date between '2014-05-01' and '2014-05-31';
```

100 %

Results | Messages

	order_id	Ordered
1	624	05/04/14
2	627	05/05/14
3	630	05/08/14
4	651	05/19/14
5	658	05/23/14

Explanation: Selected order id and order date but formatted order date as mm/dd/yy-also labelled it as Ordered. Order id and order date were pulled from the orders table. Order date had a set range from May 1 2014 – May 31 2014 using the between operator.

Query #12: Write a query which displays the order id, customer id, and the number of days between the order date and the ship date (use the DATEDIFF function). Name this column "Days" and sort by highest to lowest number of days. Only display orders where this result is 15 days or more.

```
select order_id, customer_id, DATEDIFF(day, order_date, shipped_date) as 'Days'
from Orders -- from Orders table
where DATEDIFF(day, order_date, shipped_date) >= 15
order by DATEDIFF(day, order_date, shipped_date) desc;
```

SQLQuery10.sql -...LLPDLNN\lupe (70))*

```
--Guadalupe Alvarado
select order_id, customer_id, DATEDIFF(day, order_date, shipped_date) as 'Days'
from Orders -- from Orders table
where DATEDIFF(day, order_date, shipped_date) >= 15
order by DATEDIFF(day, order_date, shipped_date) desc;
```

100 %

Results | Messages

	order_id	customer_id	Days
1	413	17	37
2	180	24	35
3	298	18	35
4	479	1	32
5	548	2	27
6	321	2	26
7	158	9	16

Explanation: Selected columns order_id, customer_id. Used operator DATEDIFF(day, order_date, shipped_date) and label it as 'Days'. Order id and customer id were pulled from the orders table. DATEDDIFF has a clause where only 15 days or more are shown and its ordered in descending order.

Query #13: Write a query which displays the order id, customer id, employee id, and order date for all orders that have NOT been shipped, sorted by order date with the most recent order at the top.

```
select order_id, customer_id, employee_id, order_date
from orders
where shipped_date is null order by order_Date desc;
```

SQLQuery10.sql -...LLPDLNN\lupe (70))*

```
--Guadalupe Alvarado
select order_id, customer_id, employee_id, order_date
from orders
where shipped_date is null order by order_Date desc;
```

100 %

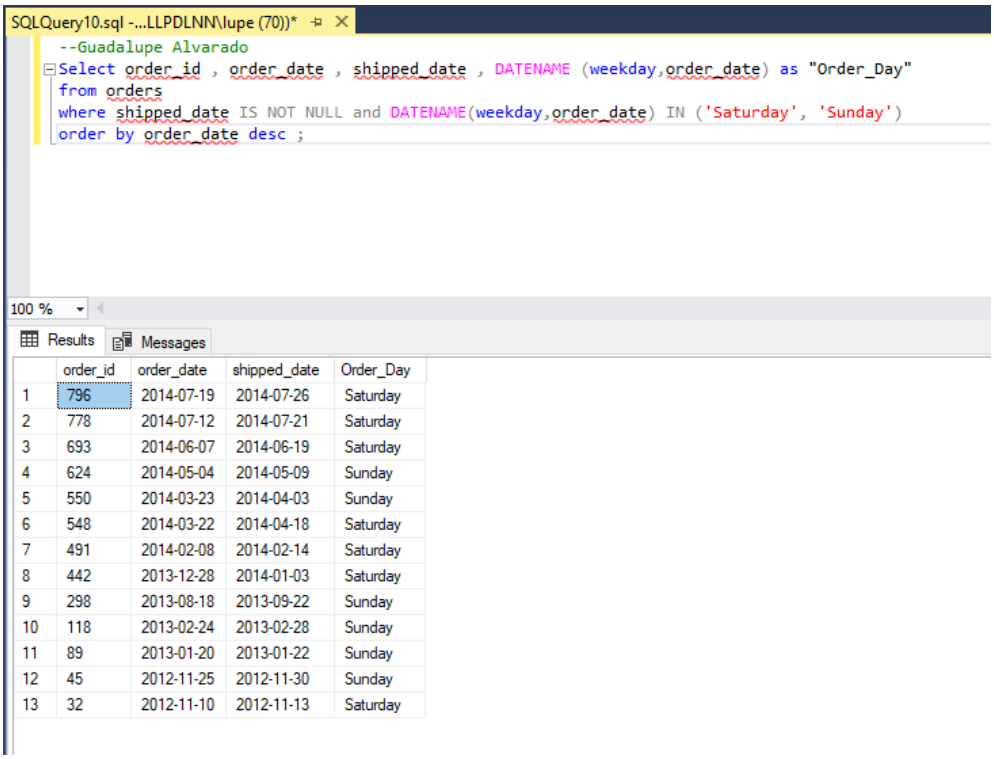
Results | Messages

	order_id	customer_id	employee_id	order_date
1	827	18	NULL	2014-08-02
2	829	9	NULL	2014-08-02
3	824	1	NULL	2014-08-01

Explanation: Selected columns order_id, customer_id, employee_id, and order_date from the orders table. Used where clause to only show the orders that have not been shipped yet. Ordered the orders by the latest one.

Query #14: The Marketing Department has requested a new report of shipped orders for which the order was placed on either a Saturday or a Sunday. Write a query which displays the order id, order date, shipped date, along with a calculated column labeled “Order_Day” showing the day of the week the order was placed (use the DAYNAME function). Only display orders that have shipped and were placed on a Saturday or Sunday. Sort by order date with most recent orders at the top.

```
Select order_id , order_date , shipped_date , DATENAME (weekday,order_date) as
"Order_Day"
from orders
where shipped_date IS NOT NULL and DATENAME(weekday,order_date) IN ('Saturday', 'Sunday')
order by order_date desc ;
```



	order_id	order_date	shipped_date	Order_Day
1	796	2014-07-19	2014-07-26	Saturday
2	778	2014-07-12	2014-07-21	Saturday
3	693	2014-06-07	2014-06-19	Saturday
4	624	2014-05-04	2014-05-09	Sunday
5	550	2014-03-23	2014-04-03	Sunday
6	548	2014-03-22	2014-04-18	Saturday
7	491	2014-02-08	2014-02-14	Saturday
8	442	2013-12-28	2014-01-03	Saturday
9	298	2013-08-18	2013-09-22	Sunday
10	118	2013-02-24	2013-02-28	Sunday
11	89	2013-01-20	2013-01-22	Sunday
12	45	2012-11-25	2012-11-30	Sunday
13	32	2012-11-10	2012-11-13	Saturday

Explanation: Selected columns order_id, order_date, shipped_date, and order_date but formatted it into weekdays only and labelled it Order_day, from the orders table. Used where clause to only show the orders that have not been shipped yet and only Saturday and Sunday. Ordered the orders by the latest one.

Query 15: Write a query to display the customer's last name, phone number, and fax number but only display those customers that have a fax number.

```
select customer_last_name, customer_phone, customer_fax
from customers
where customer_fax is not null;
```

SQLQuery10.sql -...LLPDLNN\lupe (70))*

```
--Guadalupe Alvarado
select customer_last_name, customer_phone, customer_fax
from customers
where customer_fax is not null;
```

100 %

Results | Messages

	customer_last_name	customer_phone	customer_fax
1	Blanca	6145554435	6145553928
2	Randall	2095551205	2095552262
3	Hostlery	8005551957	8005552826
4	Nickalus	8055550584	055556689
5	Eulalia	2095557500	2095551302
6	Quintin	6145558600	6145557580
7	Holbrooke	5595558625	5595558495

Explanation: Selected columns customer's last name, phone number, and fax number from the customer's table. Used where clause to only show the customers that have a fax number on file.

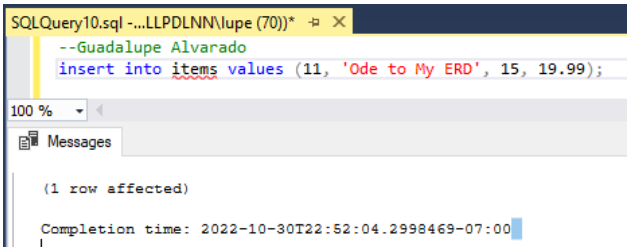
Query 16: Create a statement to insert a new record into the items table with the following values:

item_id:	11
title:	Ode To My ERD
Artist_id:	15
unit_price:	19.99

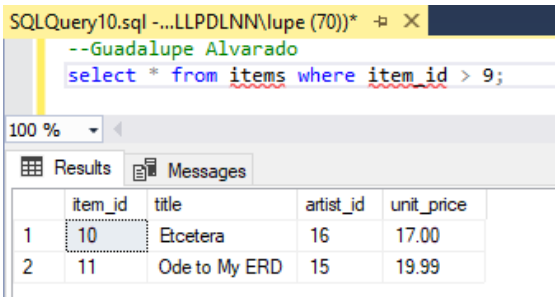
Show your INSERT statement along with the results of the following SELECT query to verify that the insert worked correctly.

select * from items where item_id > 9;

```
insert into items values (11, 'Ode to My ERD', 15, 19.99);
```




```
select * from items where item_id > 9;
```



Explanation: Inserted into the items table the values, “11, 'Ode to My ERD', 15, 19.99”. Verified that the insert worked correctly by selecting all the item ids in the items table, where the item id was greater than 9. Two items were selected since there is an item with an id of 10.

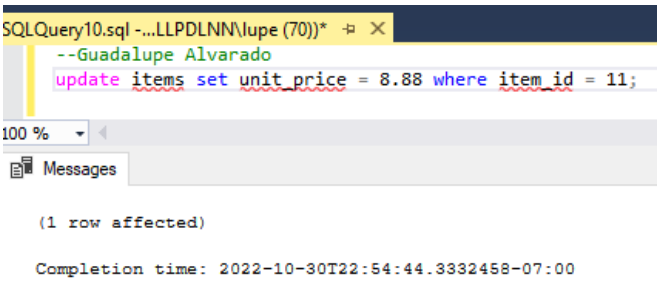
Query 17: Create a statement to update the record inserted in the previous step to change the unit price of this item to 8.99.

item_id:	11
title:	Ode To My ERD
artist:	15
unit_price:	8.88

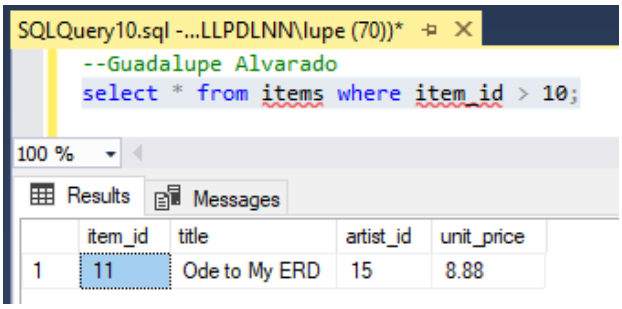
Show your UPDATE statement along with the results of the following SELECT query to verify that the insert worked correctly.

select * from items where item_id > 10;

```
update items set unit_price = 8.88 where item_id = 11;
```




```
select * from items where item_id > 10;
```



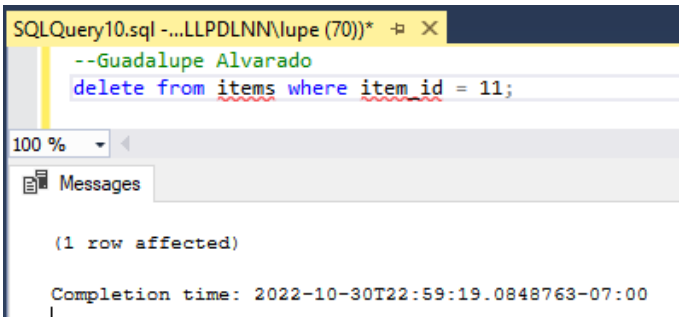
Explanation: Updated the item in items table that had the item value of 11. Verified that the update worked correctly by selecting all the item ids in the items table, where the item id was greater than 10.

Query #18: Create a statement to delete the entire record that was inserted and then updated in the previous steps.

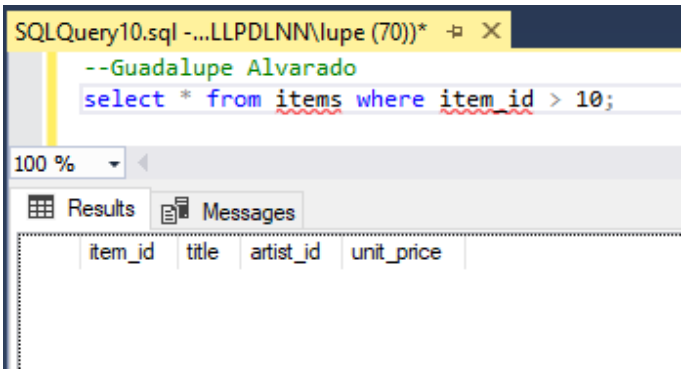
Show your DELETE statement along with the results of the following SELECT query to verify that the insert worked correctly.

select * from items where item_id > 10;

```
delete from items where item_id = 11;
```



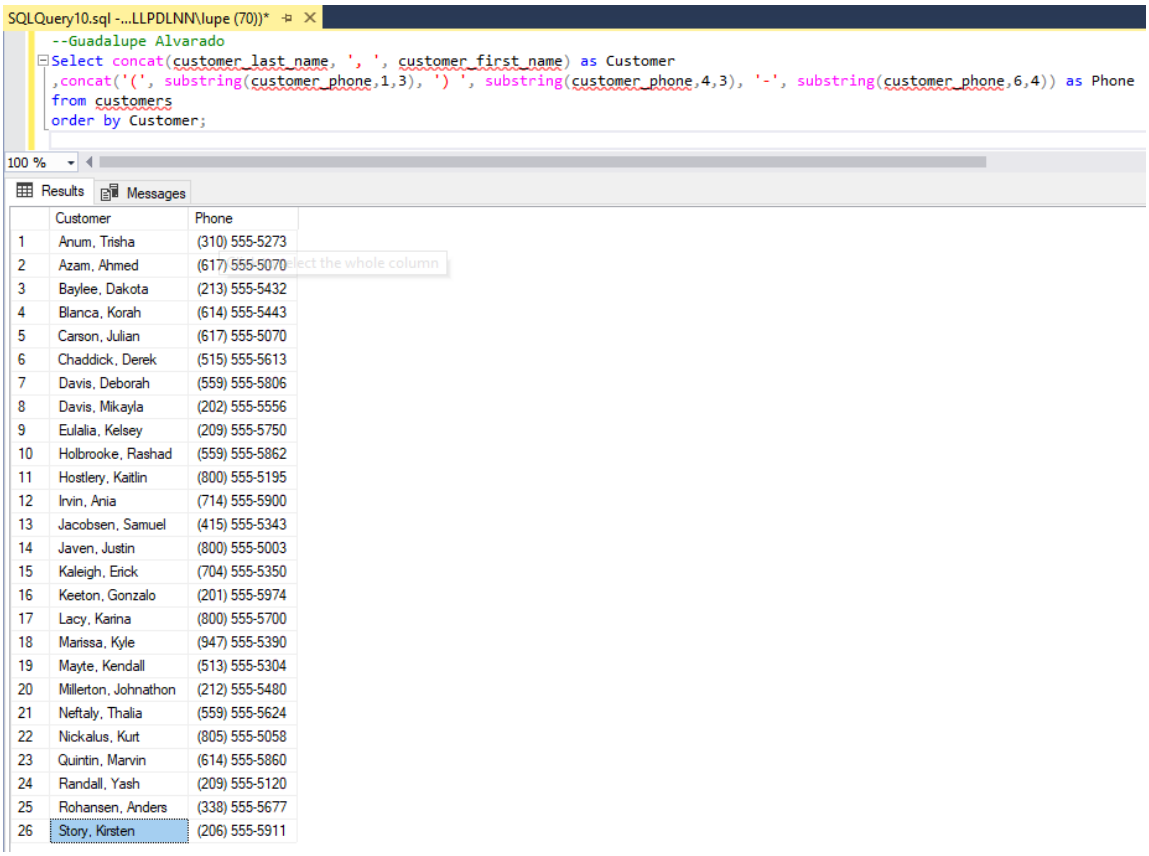

```
select * from items where item_id > 10;
```



Explanation: Deleted the row from the items table which item id = 11. Verified that the item was deleted by selecting an item_id that was greater than 10.

Query 19: Using the SUBSTRING and CONCAT functions, write a query to display each customer name as a single field in the format “Jones, Tom” with a heading of Customer along with the customer_phone field in a nicely formatted calculated column named Phone. For example, a record containing the customer_phone value 6145535443 would be output with parentheses, spaces, and hyphens, like this: (614) 555-5443. Sort by last name.

```
Select concat(customer_last_name, ', ', customer_first_name) as Customer
,concat('(', substring(customer_phone,1,3), ') ', substring(customer_phone,4,3), '-',
substring(customer_phone,6,4)) as Phone
from customers
order by Customer;
```



	Customer	Phone
1	Anum, Trisha	(310) 555-5273
2	Azam, Ahmed	(617) 555-5070
3	Baylee, Dakota	(213) 555-5432
4	Blanca, Korah	(614) 555-5443
5	Carson, Julian	(617) 555-5070
6	Chaddick, Derek	(515) 555-5613
7	Davis, Deborah	(559) 555-5806
8	Davis, Mikayla	(202) 555-5556
9	Eulalia, Kelsey	(209) 555-5750
10	Holbrooke, Rashad	(559) 555-5862
11	Hostlery, Kaitlin	(800) 555-5195
12	Irvin, Ania	(714) 555-5900
13	Jacobsen, Samuel	(415) 555-5343
14	Javen, Justin	(800) 555-5003
15	Kaleigh, Erick	(704) 555-5350
16	Keeton, Gonzalo	(201) 555-5974
17	Lacy, Karina	(800) 555-5700
18	Marissa, Kyle	(947) 555-5390
19	Mayte, Kendall	(513) 555-5304
20	Millerton, Johnathon	(212) 555-5480
21	Neftaly, Thalia	(559) 555-5624
22	Nickalus, Kurt	(805) 555-5058
23	Quintin, Marvin	(614) 555-5860
24	Randall, Yash	(209) 555-5120
25	Rohansen, Anders	(338) 555-5677
26	Story, Kirsten	(206) 555-5911

Explanation: Added three strings together from the fields(columns) from the customers table to show the Customer’s first name, last name, and an extra ‘,’ as one field. Added four strings together from the fields(columns) from the customers table to show the Customer’s phone number, strings ‘(‘,’)’,’-‘, and the customers0 phone number. The table was sorted by customer last name.

Query 20: Create a statement to insert a new record with your values: your customer id, first name, last name, address, city, state, zip code p and fax number.

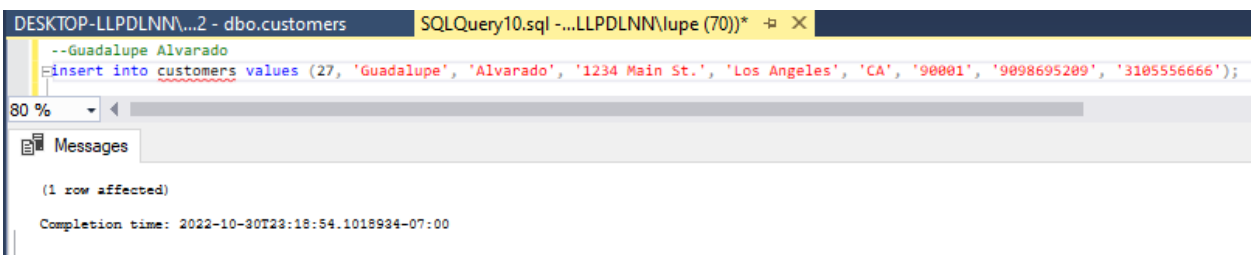
Customer id:	26
Your First Name	Your first Name
Your Last Name	Your last name
Your Address	3801 West Temple Avenue
Your City	Pomona
Your State	CA
Your Zip Code	91768
Your Phone	(909) 869-5209
Your fax	(310)555-6666

Note: Use your real name and (1234 Main St, Los Angeles, CA 90001) as your address:

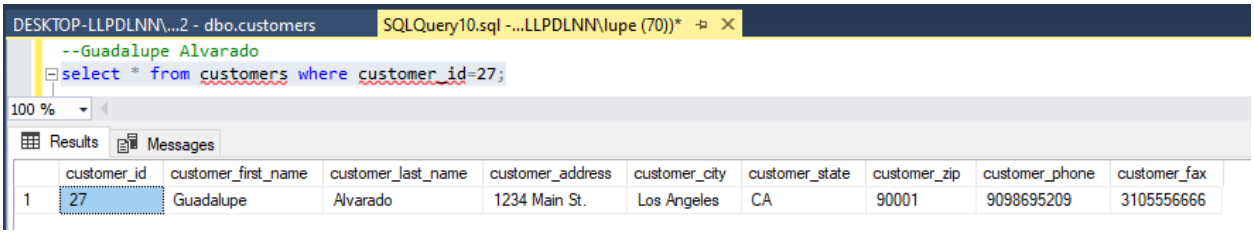
Show your INSERT statement along with the results of the following SELECT query to verify that the insert worked correctly.

*select * from Customer_T where Customer ID = 26*;

```
insert into customers values (27, 'Guadalupe', 'Alvarado', '1234 Main St.', 'Los
Angeles', 'CA', '90001', '9098695209', '3105556666');
```



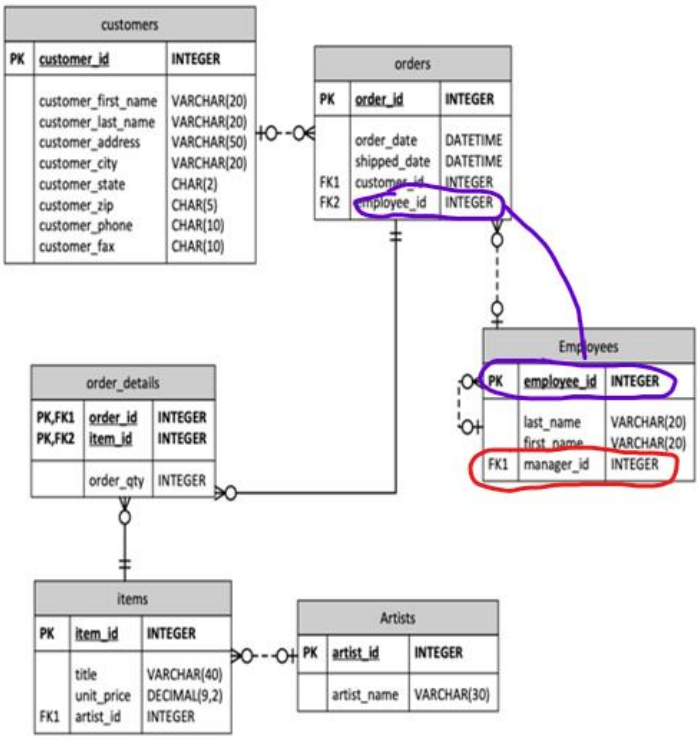

```
select * from customers where customer_id=27;
```



Explanation: Inserted into the customers table the values. Customer id 26 was taken already so I used 27. Verified that the insert worked correctly by selecting the customer id that equaled to 27.

Query 21: ***Explain the cardinality from the employees-to-employees table.***

An employee might have their employee_id as a foreign key on the orders table because there might be an employee assigned to a certain order. Although the field is null so an employee might not be on an order field. Additionally, there are managerids that are treated as foreign keys on the employee tables. The manager_id is null because not every employee is a manager and it's a foreign key because there is a different table with managers. Although the managers table is not listed in this project, we assume there is one because of the cardinality between the employees table and manager_id.



Results, Discussion, & Report Analysis

As I was working on the tables and fields, I found the need to correctly format my sql code to make it more readable. Each query had a specific purpose, and it was intended to give certain results for the producer. A lot of the queries were in the customers table and orders table. For query 7, the company wanted to put all the items on sale for 15% off and I used the round operator to achieve that. For query 10-13, it seemed more order specific like if the company wanted to know the status of certain orders. There were some queries that seemed more beneficial than others like the ones stated above but I believe all the queries served a purpose.

Lessons Learned

In the project documentation, there was certain operators that were not correct for MS SQL. At least that's what I read according to documentation on the Microsoft website. For example, the DAYNAME operator is used for MySQL and not MS SQL, so I had to work around this and use a different operator called DATENAME. This operator was almost the same, it just needs one more variable , 'weekday'. Another problem I ran into was when I went to insert my own name and details for the customer table. There was already a person who was using the specific customer ID the documentation wanted me to use. I had two options, delete the person's records that were already there or just use a different customer ID and I chose to do the latter. Lastly, the most important lesson I learned was to handle the SQL language with more organization. It's easy to continue to write the query in one big line, but I realized this would be hard to read in the future if anyone wanted to use the same query.

Conclusion

In Project #2, I designed, developed, and demonstrated the functionality of a database created based on a set of business specifications. I created logical and physical data models by using Microsoft SQL Server 2019, MS SQL Server Management Tool, and the class textbook. The tables will were updated, deleted, and populated using script files. The logical and physical schema's relationships were shown using cardinalities. The goal of this project was to create a detailed and correct database management system and I believe I achieved that with all the queries required of me.

References: https://learn.microsoft.com/en-us/sql/t-sql/functions/datename-transact-sql?view=sql-server-ver16

https://learn.microsoft.com/en-us/sql/t-sql/functions/format-transact-sql?view=sql-server-ver16

https://learn.microsoft.com/en-us/sql/t-sql/functions/concat-transact-sql?view=sql-server-ver16