CALIFORNIA POLYTECHNIC STATE UNIVERSITY POMONA

Dylan Gonzalez

# Pine-valley Furniture SQL Project

March 31, 2024

CIS 3050

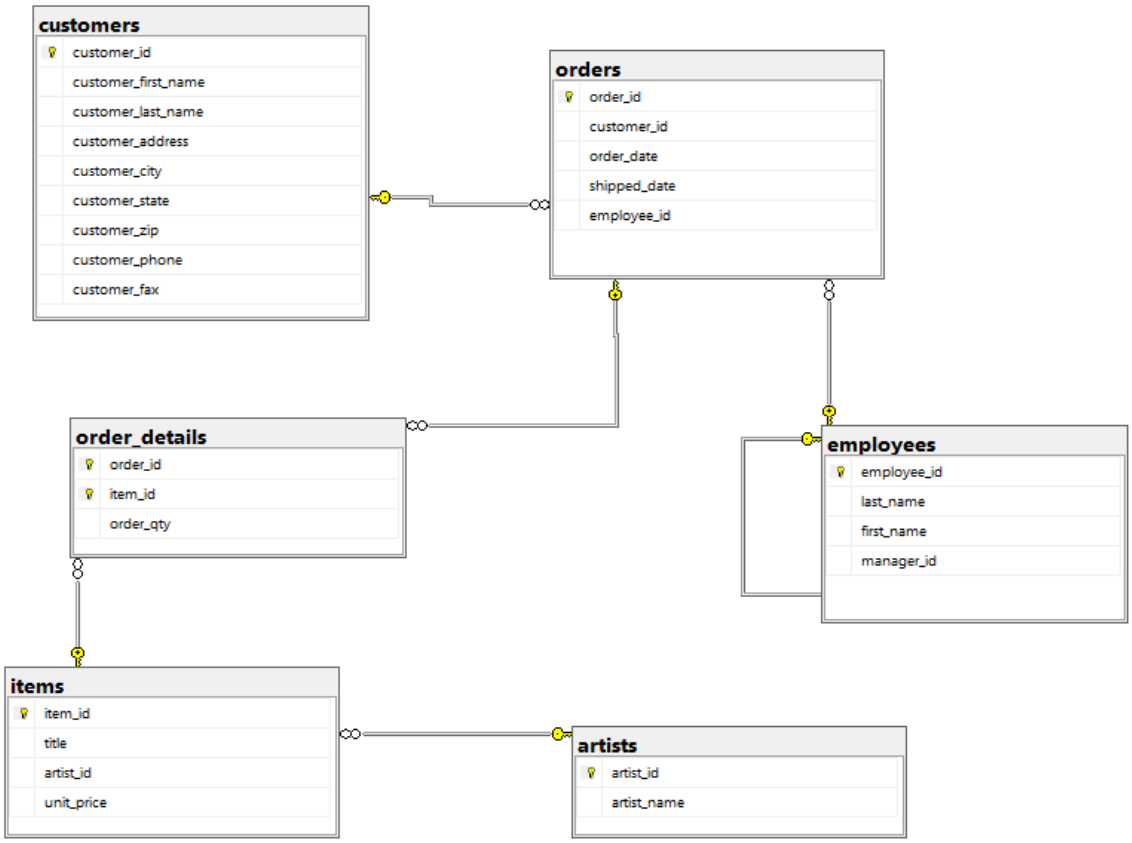
# Table of Contents

## Introduction

The project that I was assigned will dive into the fundamental concepts of database design and logical/physical data modeling. In this project, I will be utilizing MS SQL and its various aspects to retrieve data from the database using statements such as the SELECT, DELETE, UPDATE and more. This database will utilize tables represented by the ERD which will be further discussed in the *Project Description* below.

## Project Description

The database I will be using will be created from the ERD model below. MS SQL will be used to retrieve data to create queries based on the questions asked within this project in order to aid the business to draw meaningful conclusions. Query questions will also have brief descriptions of what is being displayed from the SQL output. There are six tables (customers, orders, order_details, items, artists, and employees) that will be used to pull data to develop queries for each question that the business may have.



# Database Queries

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

*SELECT customer_last_name, customer_state, customer_phone*
*FROM customers*
*ORDER BY customer_state, customer_last_name*

SQLQuery1.sql - D...8NGS54\crxho (88))

```
use Gonzalez

--- CIS3050-05
--- Project 2
--- Dylan Gonzalez
--- 3/31/2024

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

select customer_last_name, customer_state, customer_phone
from customers
order by customer_state, customer_last_name
```

99 %

Results | Messages

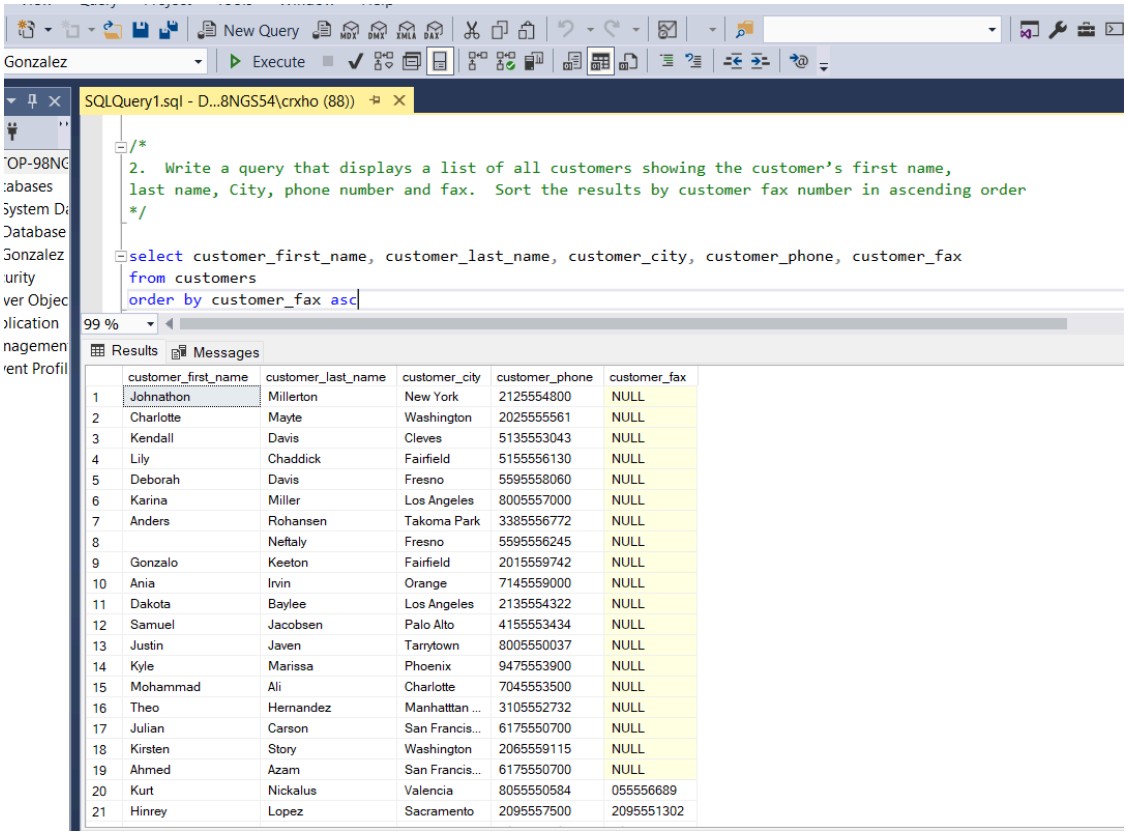
| | customer_last_name | customer_state | customer_phone |
|---|---|---|---|
| 1 | Marissa | AZ | 9475553900 |
| 2 | Azam | CA | 6175550700 |
| 3 | Baylee | CA | 2135554322 |
| 4 | Carson | CA | 6175550700 |
| 5 | Davis | CA | 5595558060 |
| 6 | Hernandez | CA | 3105552732 |
| 7 | Holbrooke | CA | 5595558625 |
| 8 | Irvin | CA | 7145559000 |
| 9 | Jacobsen | CA | 4155553434 |
| 10 | Lopez | CA | 2095557500 |
| 11 | Miller | CA | 8005557000 |
| 12 | Neftaly | CA | 5595556245 |
| 13 | Nickalus | CA | 8055550584 |
| 14 | Mayte | DC | 2025555561 |
| 15 | Story | DC | 2065559115 |

Explanation of Query/ Results

Query displays three columns that show the customer's last name, their state, and phone number from the customers table and have the results ordered by their state and last name.

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

*SELECT customer_first_name, customer_last_name, customer_city, customer_phone, customer_fax*
*FROM customers*
*Order by customer_fax ASC*



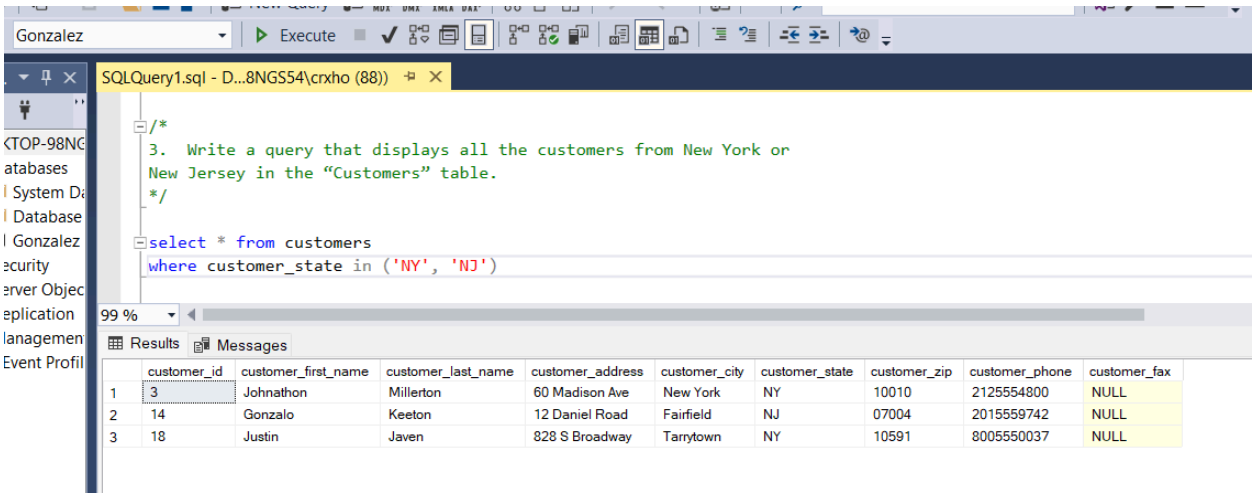

| | customer_first_name | customer_last_name | customer_city | customer_phone | customer_fax |
|---|---|---|---|---|---|
| 1 | Johnathon | Millerton | New York | 2125554800 | NULL |
| 2 | Charlotte | Mayte | Washington | 2025555561 | NULL |
| 3 | Kendall | Davis | Cleves | 5135553043 | NULL |
| 4 | Lily | Chaddick | Fairfield | 5155556130 | NULL |
| 5 | Deborah | Davis | Fresno | 5595558060 | NULL |
| 6 | Karina | Miller | Los Angeles | 8005557000 | NULL |
| 7 | Anders | Rohansen | Takoma Park | 3385556772 | NULL |
| 8 | | Neftaly | Fresno | 5595556245 | NULL |
| 9 | Gonzalo | Keeton | Fairfield | 2015559742 | NULL |
| 10 | Ania | Irvin | Orange | 7145559000 | NULL |
| 11 | Dakota | Baylee | Los Angeles | 2135554322 | NULL |
| 12 | Samuel | Jacobsen | Palo Alto | 4155553434 | NULL |
| 13 | Justin | Javen | Tarrytown | 8005550037 | NULL |
| 14 | Kyle | Marissa | Phoenix | 9475553900 | NULL |
| 15 | Mohammad | Ali | Charlotte | 7045553500 | NULL |
| 16 | Theo | Hernandez | Manhatttan ... | 3105552732 | NULL |
| 17 | Julian | Carson | San Francis... | 6175550700 | NULL |
| 18 | Kirsten | Story | Washington | 2065559115 | NULL |
| 19 | Ahmed | Azam | San Francis... | 6175550700 | NULL |
| 20 | Kurt | Nickalus | Valencia | 8055550584 | 055556689 |
| 21 | Hinrey | Lopez | Sacramento | 2095557500 | 2095551302 |

<u>Explanation of Query/ Results</u>

Query displays columns of all customers by their first and last name, city, phone, and fax number. The order of customers is ascending by their fax numbers.

**<u>Query #3</u>**: Write a query that displays all the customers from New York or New Jersey in the "Customers" table.

*SELECT * FROM customers*
*WHERE customer_state in ('NY', 'NJ')*

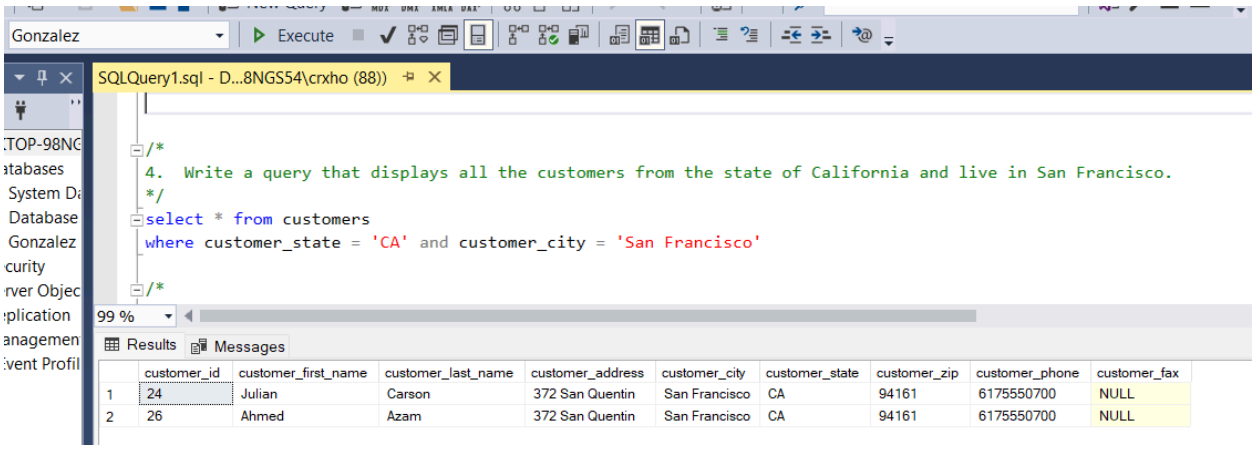



| | customer_id | customer_first_name | customer_last_name | customer_address | customer_city | customer_state | customer_zip | customer_phone | customer_fax |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 3 | Johnathon | Millerton | 60 Madison Ave | New York | NY | 10010 | 2125554800 | NULL |
| 2 | 14 | Gonzalo | Keeton | 12 Daniel Road | Fairfield | NJ | 07004 | 2015559742 | NULL |
| 3 | 18 | Justin | Javen | 828 S Broadway | Tarrytown | NY | 10591 | 8005550037 | NULL |

<u>Explanation of Query/ Results</u>
Query displays all details of customers that are located in New York or New Jersey from the customers table.

**<u>Query #4</u>**: Write a query that displays all the customers from the state of California and live in San Francisco.

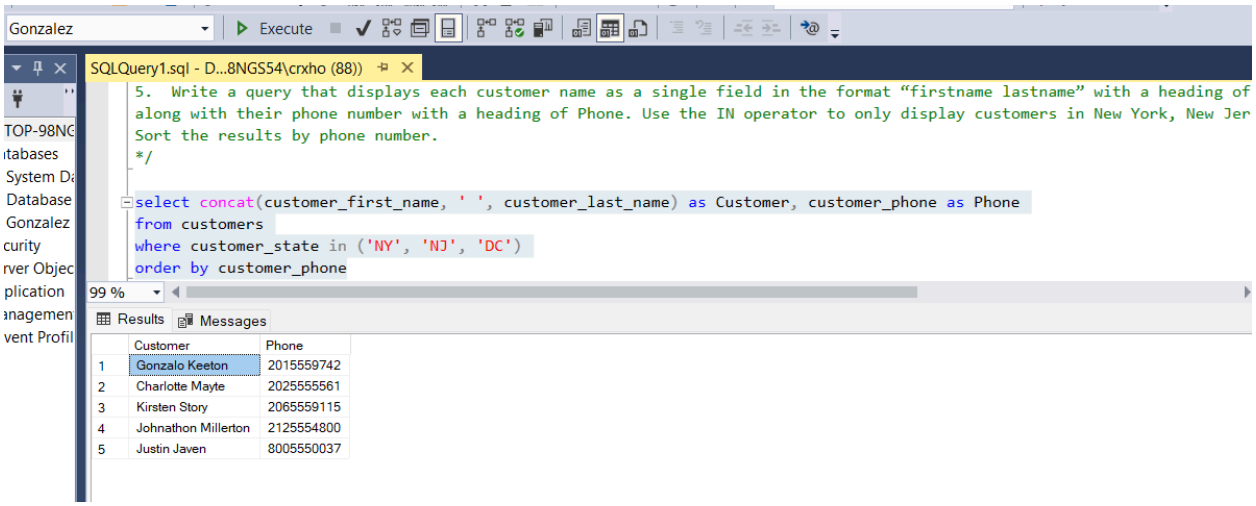
*SELECT * from customers*
*WHERE customer_state = 'CA' and customer_city = 'San Francisco'*




<u>Explanation of Query/ Results</u>
Query displays all details of customers that are located in the state of California and the city San Francisco from the customers table.

**<u>Query #5</u>**: 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 customer_phone*

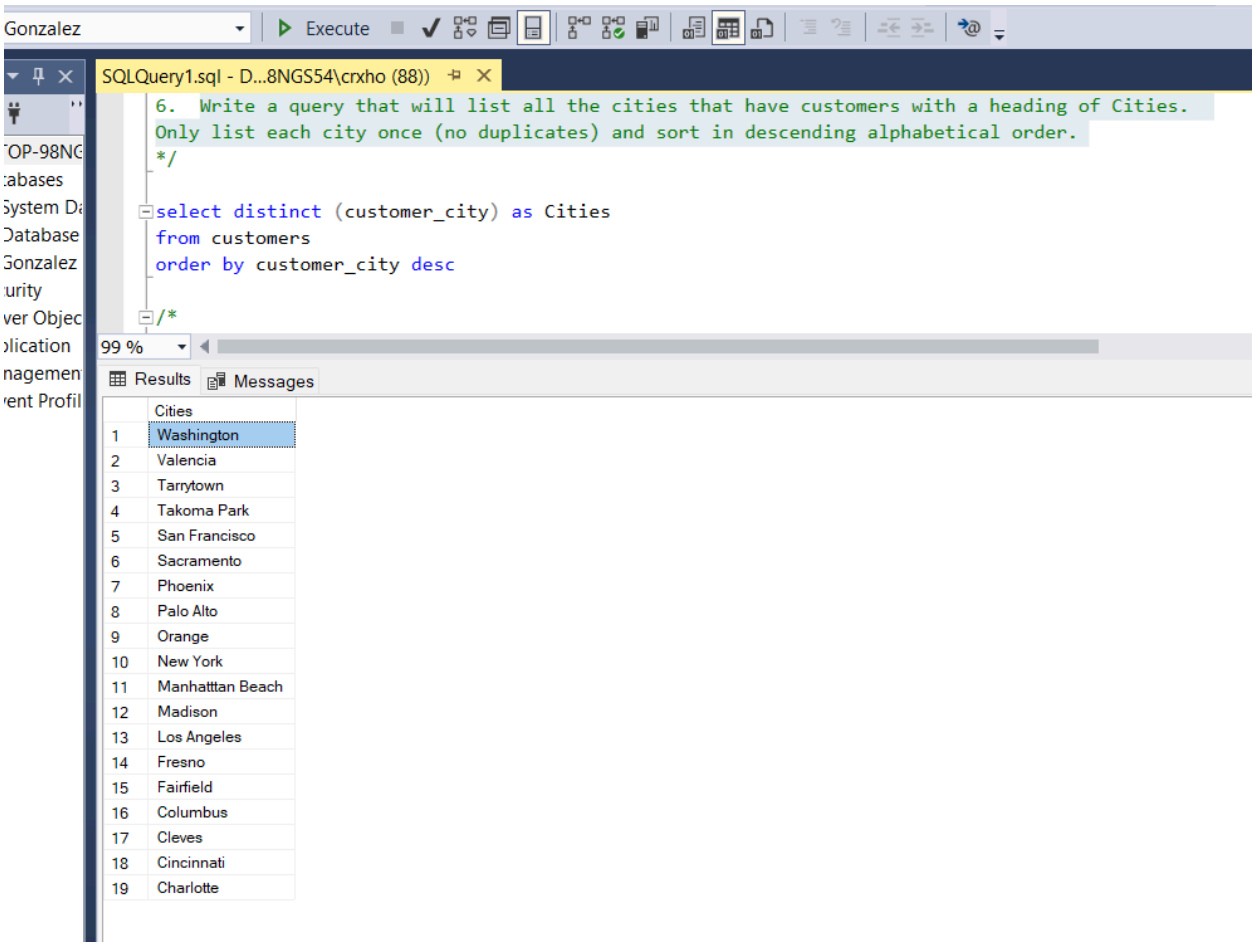



<u>Explanation of Query/ Results</u>

Query displays two columns titled "Customer", which displays the customer's first and last name as a single field using the CONCAT function, and "Phone" as their phone numbers. These customers are located in either New York, New Jersey, or Washington D.C. and results are sorted by phone number.

**<u>Query #6</u>**: 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.

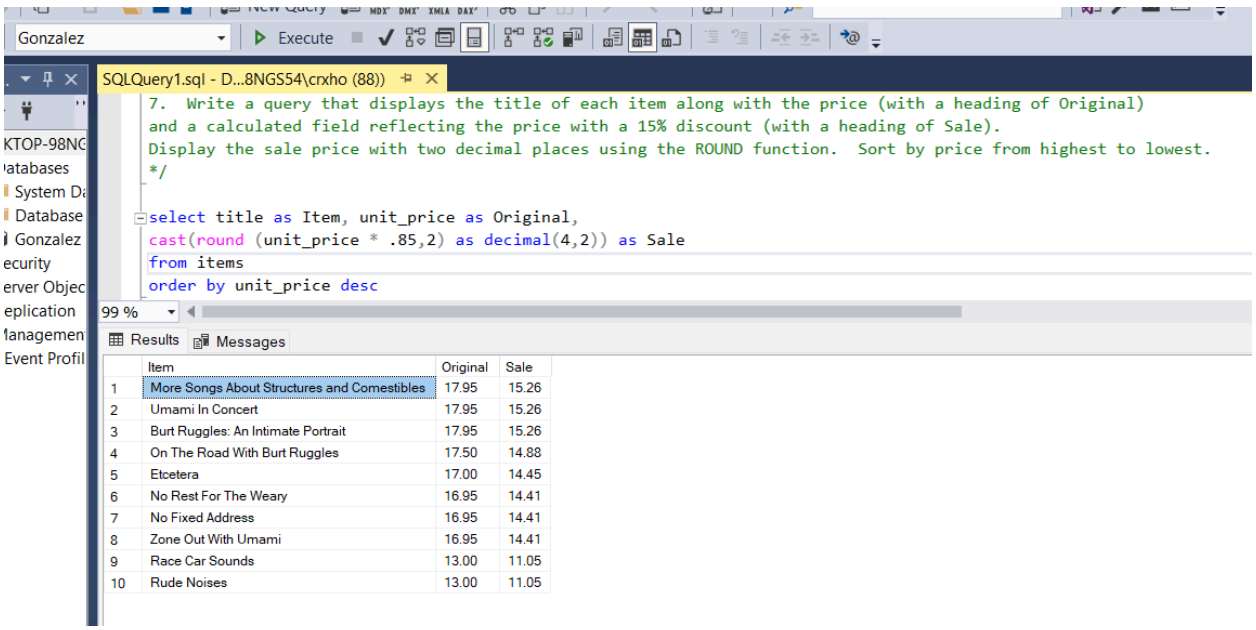
SELECT distinct (customer_city) AS Cities
FROM customers
ORDER BY customer_city desc




<u>Explanation of Query/ Results</u>
Query displays cities that could be associated with a customer and only displays each city once. No duplicate values. Results are in descending alphabetical order from customers table.

**<u>Query #7</u>**: 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 AS Item, unit_price AS Original,*
*CAST(ROUND (unit_price * .85,2) AS decimal(4,2)) AS Sale*
*FROM items*
*ORDER BY unit_price DESC*



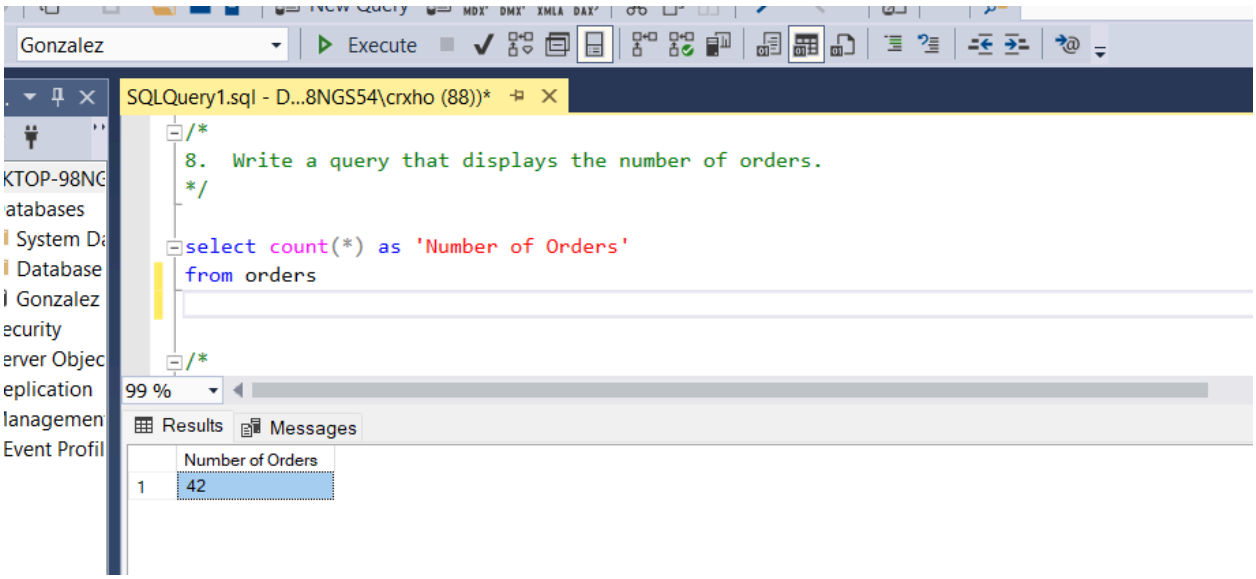

| | Item | Original | Sale |
|---|---|---|---|
| 1 | More Songs About Structures and Comestibles | 17.95 | 15.26 |
| 2 | Umami In Concert | 17.95 | 15.26 |
| 3 | Burt Ruggles: An Intimate Portrait | 17.95 | 15.26 |
| 4 | On The Road With Burt Ruggles | 17.50 | 14.88 |
| 5 | Etcetera | 17.00 | 14.45 |
| 6 | No Rest For The Weary | 16.95 | 14.41 |
| 7 | No Fixed Address | 16.95 | 14.41 |
| 8 | Zone Out With Umami | 16.95 | 14.41 |
| 9 | Race Car Sounds | 13.00 | 11.05 |
| 10 | Rude Noises | 13.00 | 11.05 |

<u>Explanation of Query/ Result</u>

Query displays aliases columns “Item”, “Original”, and “Sale” where the name and unit price for the item is displayed along with its discount price tag. The discount price tag was calculated using the CAST function to get rid of extra zeros after rounding to two decimal places. Results are sorted by the unit price in descending order from the items table.

**<u>Query #8:</u>** Write a query that displays the number of orders.

*SELECT count(*) AS 'Number of Orders'*
*FROM orders*

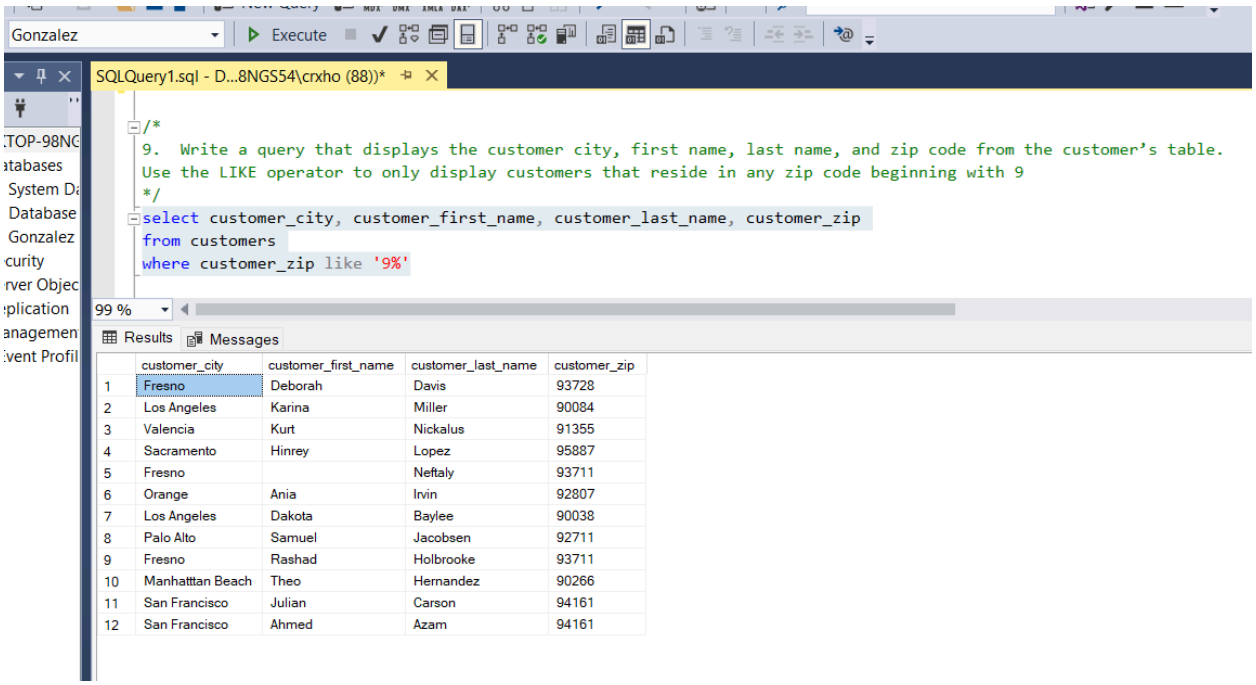



<u>Explanation of Query/ Results</u>

Query displays the total amount of orders from the orders table using the COUNT function and renaming the column as “Number of Orders”.

**<u>Query #9:</u>** 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_city, customer_first_name, customer_last_name, customer_zip*
*FROM customers*
*WHERE customer_zip like '9%'*



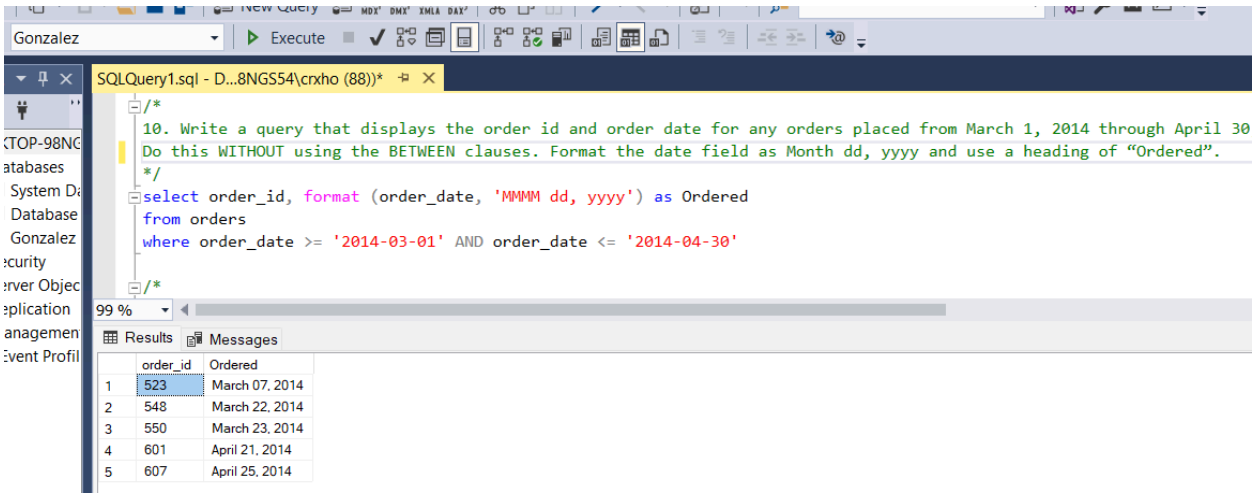

| | customer_city | customer_first_name | customer_last_name | customer_zip |
|---|---|---|---|---|
| 1 | Fresno | Deborah | Davis | 93728 |
| 2 | Los Angeles | Karina | Miller | 90084 |
| 3 | Valencia | Kurt | Nickalus | 91355 |
| 4 | Sacramento | Hinrey | Lopez | 95887 |
| 5 | Fresno | | Neftaly | 93711 |
| 6 | Orange | Ania | Irvin | 92807 |
| 7 | Los Angeles | Dakota | Baylee | 90038 |
| 8 | Palo Alto | Samuel | Jacobsen | 92711 |
| 9 | Fresno | Rashad | Holbrooke | 93711 |
| 10 | Manhatttan Beach | Theo | Hernandez | 90266 |
| 11 | San Francisco | Julian | Carson | 94161 |
| 12 | San Francisco | Ahmed | Azam | 94161 |

<u>Explanation of Query/ Results</u>

Query displays columns of the customer's city, first name, last name, and zip code from the customers table. Results are only customers that have zip codes starting with "9".

**<u>Query #10:</u>** 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 >= '2014-03-01' AND order_date <= '2014-04-30'*

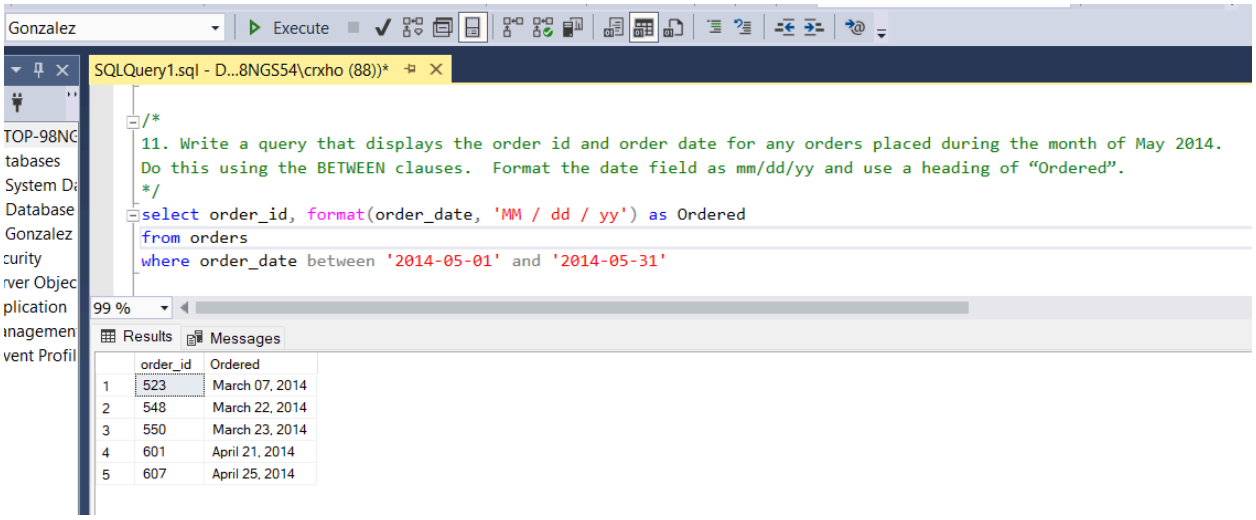



<u>Explanation of Query/ Results</u>

Query displays orders from 03/01/2014 - 04/30/2014 along with the order ID from orders table. The column "Ordered" displays a formatted date displaying the name of the month, the day, and year.

**<u>Query #11:</u>** 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'*

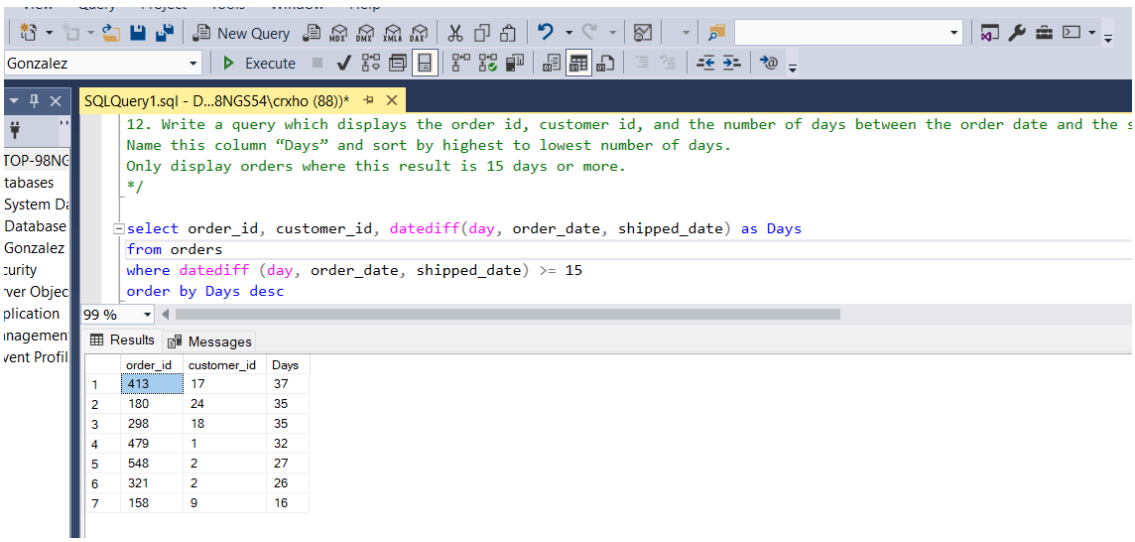



<u>Explanation of Query/ Results</u>

Query displays orders from 03/01/2014 - 04/30/2014 along with the order ID from orders table. The column "Ordered" displays a formatted date displaying the name of the month, the day, and year using the BETWEEN clause.

**<u>Query #12:</u>** 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*
*WHERE datediff (day, order_date, shipped_date) >= 15*
*ORDER BY Days DESC*

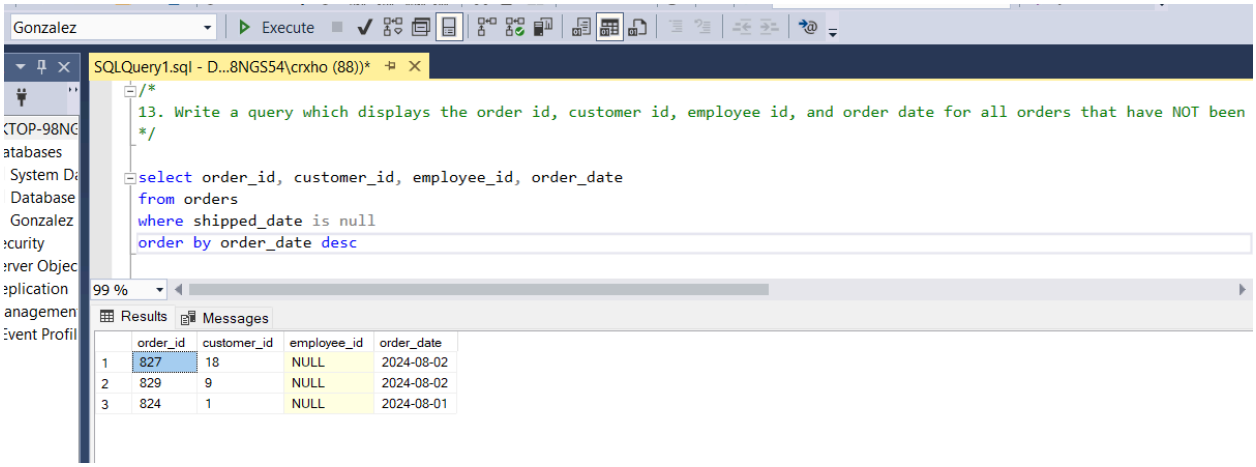



<u>Explanation of Query/ Results</u>

Query displays order ID, the customer ID, and a calculated column, "Days", which displays the amount of days it took for the order to be shipped using the DATEDIFF function. Results are a time frame of 15 days or greater in descending order from the orders table.

**<u>Query #13:</u>** 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*

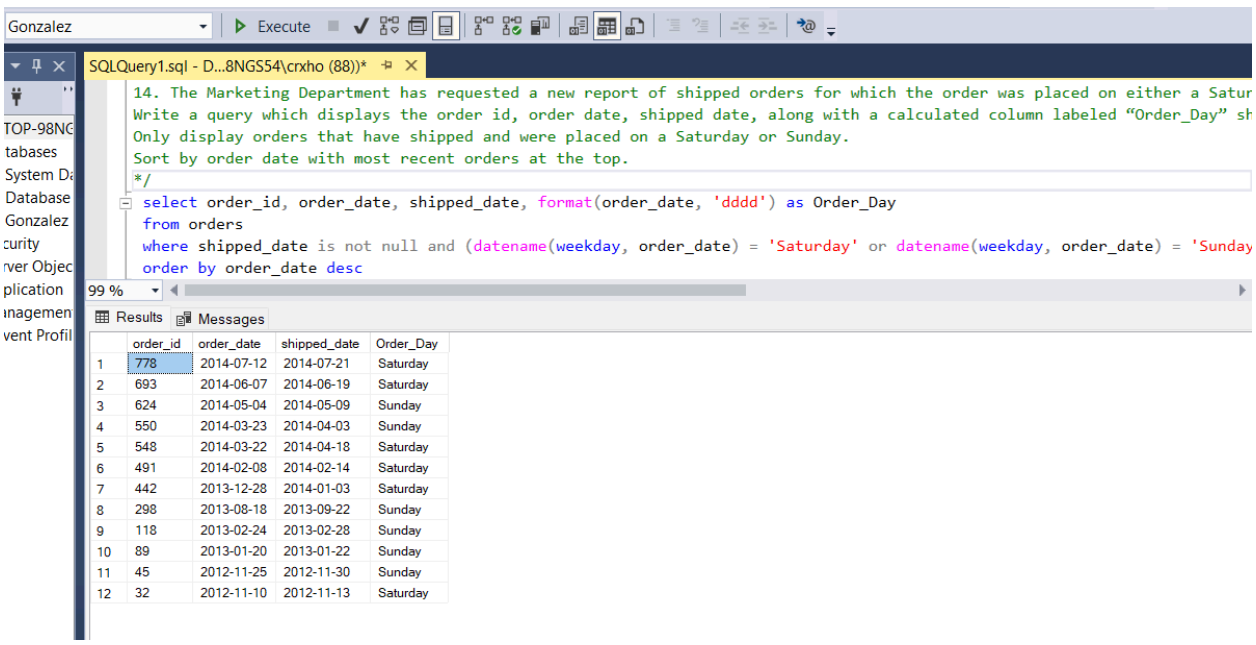



<u>Explanation of Query/ Results</u>

Query displays order ID, customer ID, employee ID, and order date. Results only display orders that have not been shipped with the most recent order at the top from the orders table.

**<u>Query #14:</u>** 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, format(order_date, 'dddd') AS Order_Day*
*FROM orders*
*WHERE shipped_date IS NOT NULL*
*AND (datename(weekday, order_date) = 'Saturday' or datename(weekday, order_date) = 'Sunday')*
*ORDER BY order_date DESC*



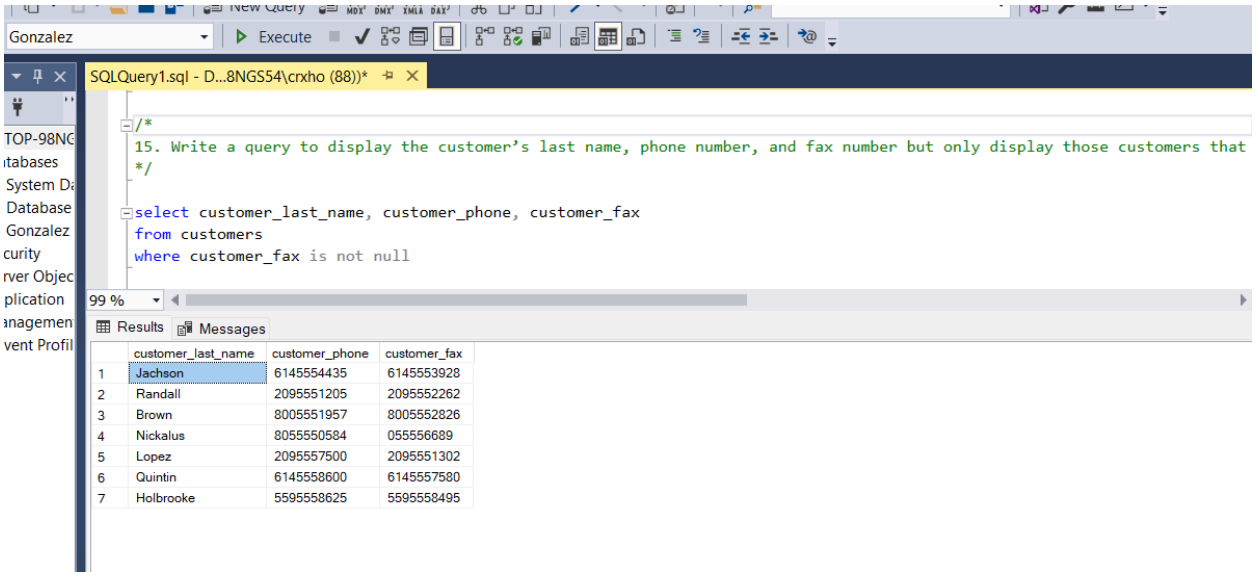

| | order_id | order_date | shipped_date | Order_Day |
|---|---|---|---|---|
| 1 | 778 | 2014-07-12 | 2014-07-21 | Saturday |
| 2 | 693 | 2014-06-07 | 2014-06-19 | Saturday |
| 3 | 624 | 2014-05-04 | 2014-05-09 | Sunday |
| 4 | 550 | 2014-03-23 | 2014-04-03 | Sunday |
| 5 | 548 | 2014-03-22 | 2014-04-18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

<u>Explanation of Query/ Results</u>

Query displays order ID, order date, shipped date, and formatted order day from orders table. Results are only orders that were placed on a Saturday or Sunday and have a shipping date. The most recent order is on top.

**<u>Query #15:</u>** 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*

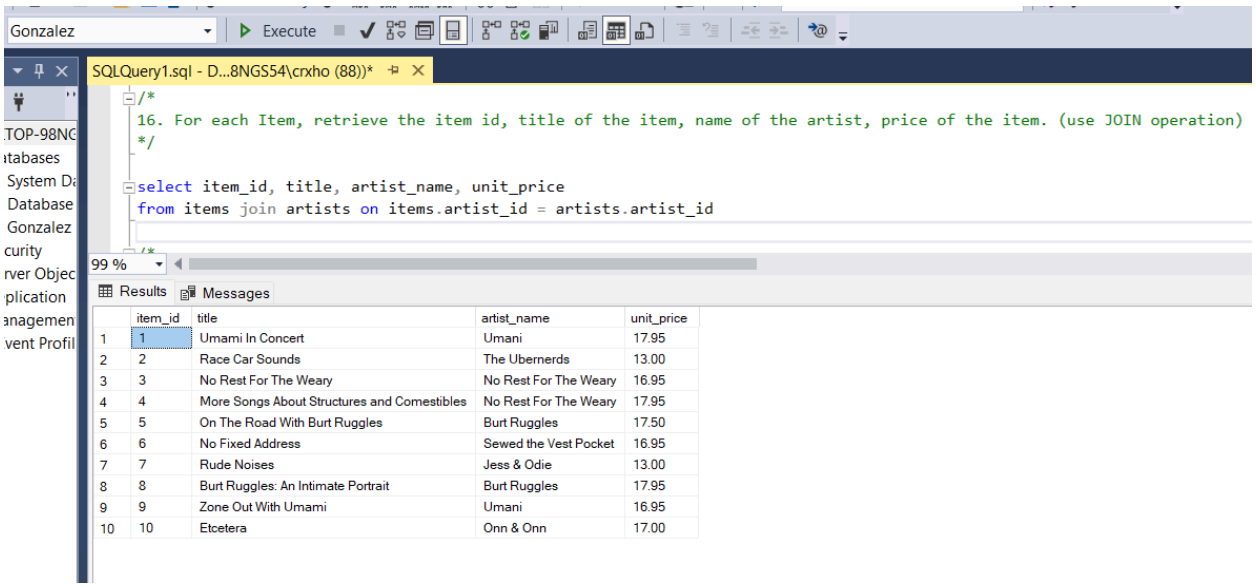



<u>Explanation of Query/ Results</u>

Query displays columns customer's last name, phone number, and fax number from customers table. Results only display customers that have a fax number.

**<u>Query #16:</u>** For each Item, retrieve the item id, title of the item, name of the artist, price of the item. (use JOIN operation)

*SELECT item_id, title, artist_name, unit_price*
*FROM items JOIN artists ON items.artist_id = artists.artist_id*



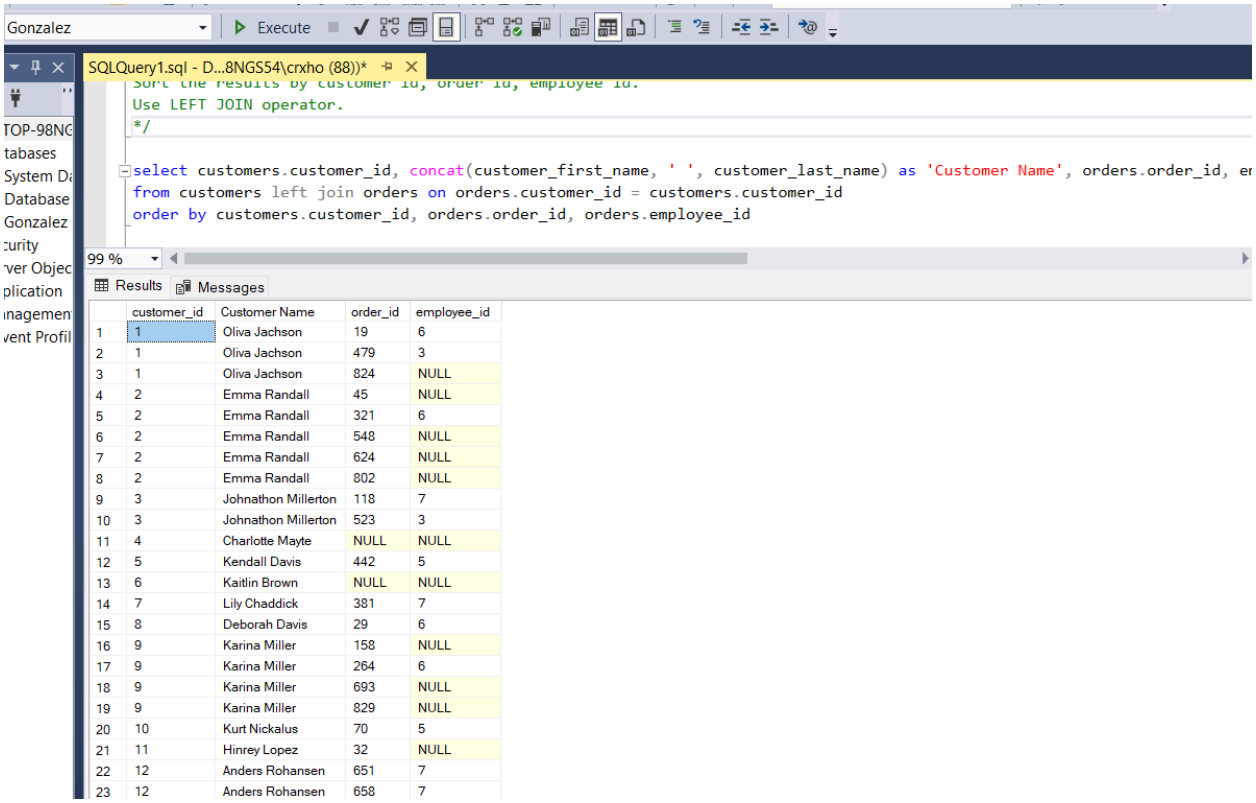

| | item_id | title | artist_name | unit_price |
|---|---|---|---|---|
| 1 | 1 | Umami In Concert | Umani | 17.95 |
| 2 | 2 | Race Car Sounds | The Ubernerds | 13.00 |
| 3 | 3 | No Rest For The Weary | No Rest For The Weary | 16.95 |
| 4 | 4 | More Songs About Structures and Comestibles | No Rest For The Weary | 17.95 |
| 5 | 5 | On The Road With Burt Ruggles | Burt Ruggles | 17.50 |
| 6 | 6 | No Fixed Address | Sewed the Vest Pocket | 16.95 |
| 7 | 7 | Rude Noises | Jess & Odie | 13.00 |
| 8 | 8 | Burt Ruggles: An Intimate Portrait | Burt Ruggles | 17.95 |
| 9 | 9 | Zone Out With Umami | Umani | 16.95 |
| 10 | 10 | Etcetera | Onn & Onn | 17.00 |

<u>Explanation of Query/ Results</u>

Query displays item ID, title of the item, artist name, and unit price. Two tables were used and utilized the JOIN operation to join the items table and artists table. Results are joined on the artist ID.

**<u>Query #17:</u>** Write a query that displays the customer id, customer name, order id, and employee id. Sort the results by customer id, order id, employee id. Use LEFT JOIN operator.

*SELECT customers.customer_id, concat(customer_first_name, ' ', customer_last_name) AS 'Customer Name', orders.order_id, employee_id*
*FROM customers LEFT JOIN orders ON orders.customer_id = customers.customer_id*
*ORDER BY customers.customer_id, orders.order_id, orders.employee_id*



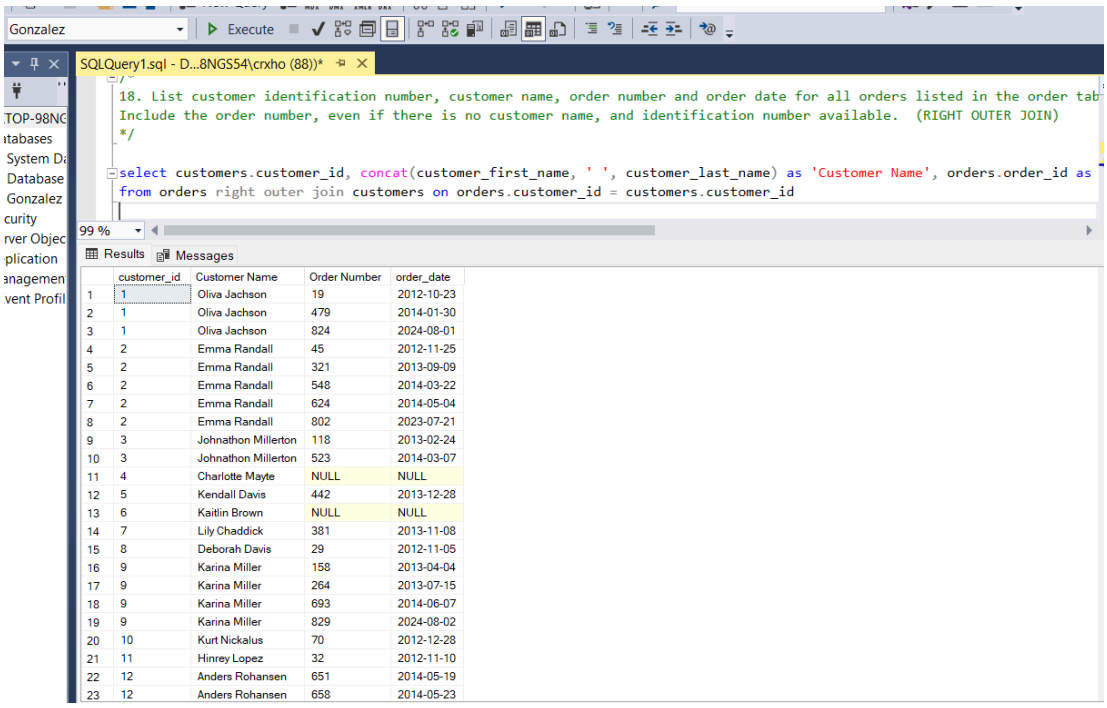

| | customer_id | Customer Name | order_id | employee_id |
|---|---|---|---|---|
| 1 | 1 | Oliva Jachson | 19 | 6 |
| 2 | 1 | Oliva Jachson | 479 | 3 |
| 3 | 1 | Oliva Jachson | 824 | NULL |
| 4 | 2 | Emma Randall | 45 | NULL |
| 5 | 2 | Emma Randall | 321 | 6 |
| 6 | 2 | Emma Randall | 548 | NULL |
| 7 | 2 | Emma Randall | 624 | NULL |
| 8 | 2 | Emma Randall | 802 | NULL |
| 9 | 3 | Johnathon Millerton | 118 | 7 |
| 10 | 3 | Johnathon Millerton | 523 | 3 |
| 11 | 4 | Charlotte Mayte | NULL | NULL |
| 12 | 5 | Kendall Davis | 442 | 5 |
| 13 | 6 | Kaitlin Brown | NULL | NULL |
| 14 | 7 | Lily Chaddick | 381 | 7 |
| 15 | 8 | Deborah Davis | 29 | 6 |
| 16 | 9 | Karina Miller | 158 | NULL |
| 17 | 9 | Karina Miller | 264 | 6 |
| 18 | 9 | Karina Miller | 693 | NULL |
| 19 | 9 | Karina Miller | 829 | NULL |
| 20 | 10 | Kurt Nickalus | 70 | 5 |
| 21 | 11 | Hinrey Lopez | 32 | NULL |
| 22 | 12 | Anders Rohansen | 651 | 7 |
| 23 | 12 | Anders Rohansen | 658 | 7 |

<u>Explanation of Query/ Results</u>

Query displays customer ID, customer name, order ID, and employee ID. Results are displayed using the LEFT JOIN operation on the customers table to match values on the orders table using the customer ID.

**<u>Query #18:</u>** List customer identification number, customer name, order number and order date for all orders listed in the order table. Include the order number, even if there is no customer name, and identification number available. (RIGHT OUTER JOIN)

*SELECT customers.customer_id, concat(customer_first_name, ' ', customer_last_name) AS 'Customer Name', orders.order_id as 'Order Number', orders.order_date*
*FROM orders RIGHT OUTER JOIN customers ON orders.customer_id = customers.customer_id*



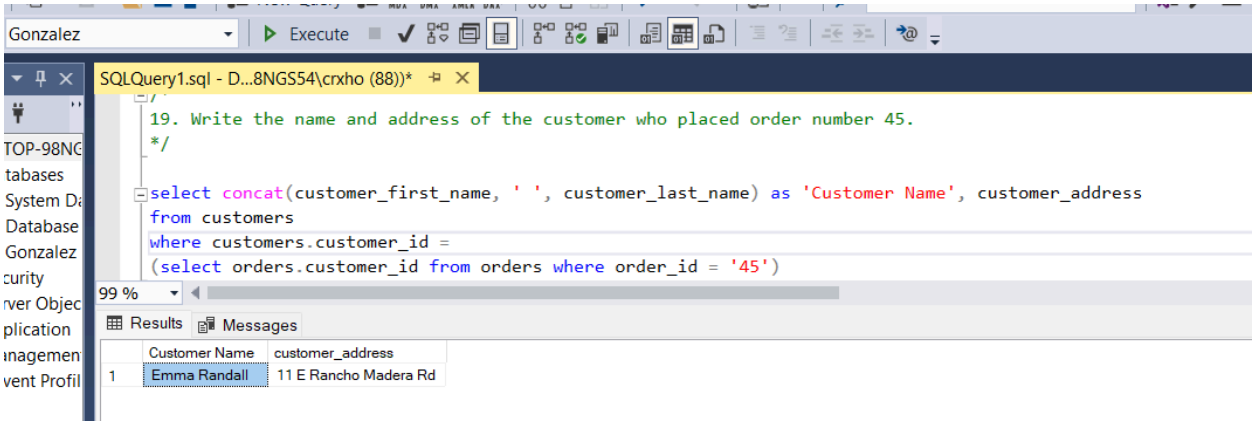

| | customer_id | Customer Name | Order Number | order_date |
|---|---|---|---|---|
| 1 | 1 | Olivia Jachson | 19 | 2012-10-23 |
| 2 | 1 | Olivia Jachson | 479 | 2014-01-30 |
| 3 | 1 | Olivia Jachson | 824 | 2024-08-01 |
| 4 | 2 | Emma Randall | 45 | 2012-11-25 |
| 5 | 2 | Emma Randall | 321 | 2013-09-09 |
| 6 | 2 | Emma Randall | 548 | 2014-03-22 |
| 7 | 2 | Emma Randall | 624 | 2014-05-04 |
| 8 | 2 | Emma Randall | 802 | 2023-07-21 |
| 9 | 3 | Johnathon Millerton | 118 | 2013-02-24 |
| 10 | 3 | Johnathon Millerton | 523 | 2014-03-07 |
| 11 | 4 | Charlotte Mayte | NULL | NULL |
| 12 | 5 | Kendall Davis | 442 | 2013-12-28 |
| 13 | 6 | Kaitlin Brown | NULL | NULL |
| 14 | 7 | Lily Chaddick | 381 | 2013-11-08 |
| 15 | 8 | Deborah Davis | 29 | 2012-11-05 |
| 16 | 9 | Karina Miller | 158 | 2013-04-04 |
| 17 | 9 | Karina Miller | 264 | 2013-07-15 |
| 18 | 9 | Karina Miller | 693 | 2014-06-07 |
| 19 | 9 | Karina Miller | 829 | 2024-08-02 |
| 20 | 10 | Kurt Nickalus | 70 | 2012-12-28 |
| 21 | 11 | Hinrey Lopez | 32 | 2012-11-10 |
| 22 | 12 | Anders Rohansen | 651 | 2014-05-19 |
| 23 | 12 | Anders Rohansen | 658 | 2014-05-23 |

<u>Explanation of Query/ Results</u>

Query displays columns customer ID, customer name, order name, and order date. Using the RIGHT OUTER JOIN operation on orders table on customer ID from customers. Results display all orders with or without customer name and customer ID.

**<u>Query #19:</u>** Write the name and address of the customer who placed order number 45.

*SELECT concat(customer_first_name, ' ', customer_last_name) AS 'Customer Name',*
*customer_address*
*FROM customers*
*WHERE customers.customer_id =*
*(SELECT orders.customer_id FROM orders WHERE order_id = '45')*

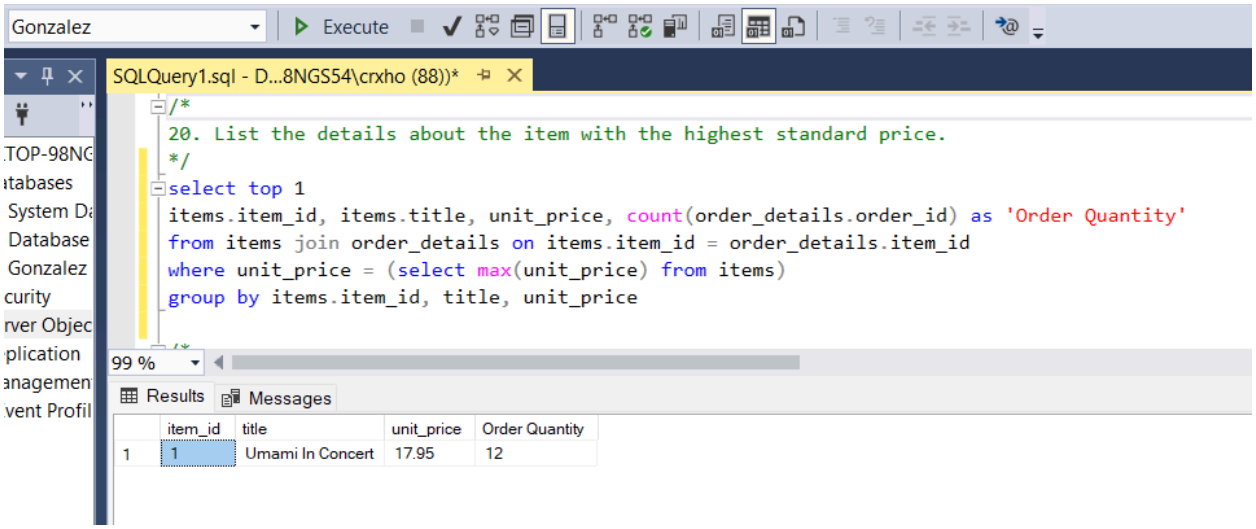



<u>Explanation of Query/ Results</u>

Query displays columns customer name and customer address from customers table. Using a subquery to pull a customer to match the customer ID of 45 from the orders table. Results display the customer as Emma Randall from 11 E Rancho Madera Rd.

**<u>Query #20:</u>** List the details about the item with the highest standard price.

*SELECT top 1*
*items.item_id, items.title, unit_price, count(order_details.order_id) AS 'Order Quantity'*
*FROM items JOIN order_details on items.item_id = order_details.item_id*
*WHERE unit_price = (SELECT max(unit_price) FROM items)*
*GROUP BY items.item_id, title, unit_price*




<u>Explanation of Query/ Results</u>

Query displays columns item ID, item title, unit price, and order quantity. Used the COUNT function to get the quantity of the item with the highest standard price. Items table was used with the JOIN clause with order_details table on the item ID. Subquery was also made to calculate the max price from the items table without creating a new column. Results are grouped by item ID, title, and unit price. The item with the highest standard price is “Umami in Concert” with a price tag of $17.95 and there are 12 orders of this item.

**Query #21:** Create a statement to insert a new record into the item table with the following values:

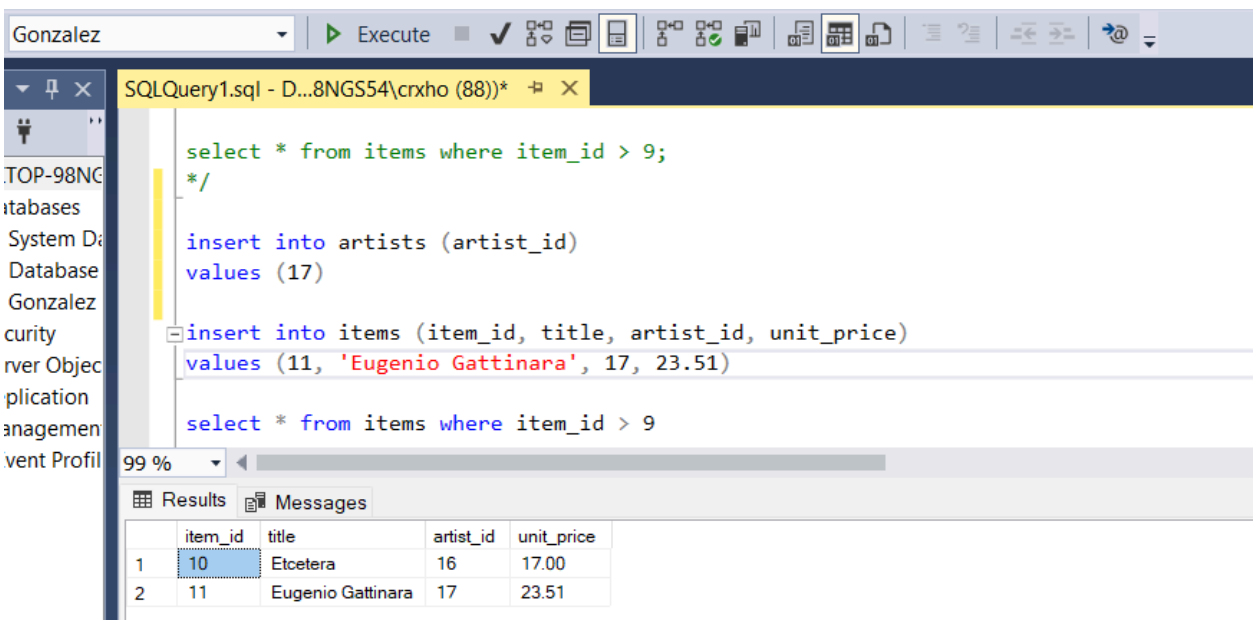
| | |
|---|---|
| **item_id:** | 11 |
| **title:** | Eugenio Gattinara |
| **Artist_id:** | 17 |
| **unit_price:** | 23.51 |

Show your INSERT statement along with the result of the following SELECT query to verify that the insert worked correctly: select * from items where item_id > 9;

*INSERT INTO artists (artist_id)*
*VALUES (17)*

*INSERT INTO items (item_id, title, artist_id, unit_price)*
*VALUES (11, 'Eugenio Gattinara', 17, 23.51)*

*SELECT * FROM items WHERE item_id > 9*



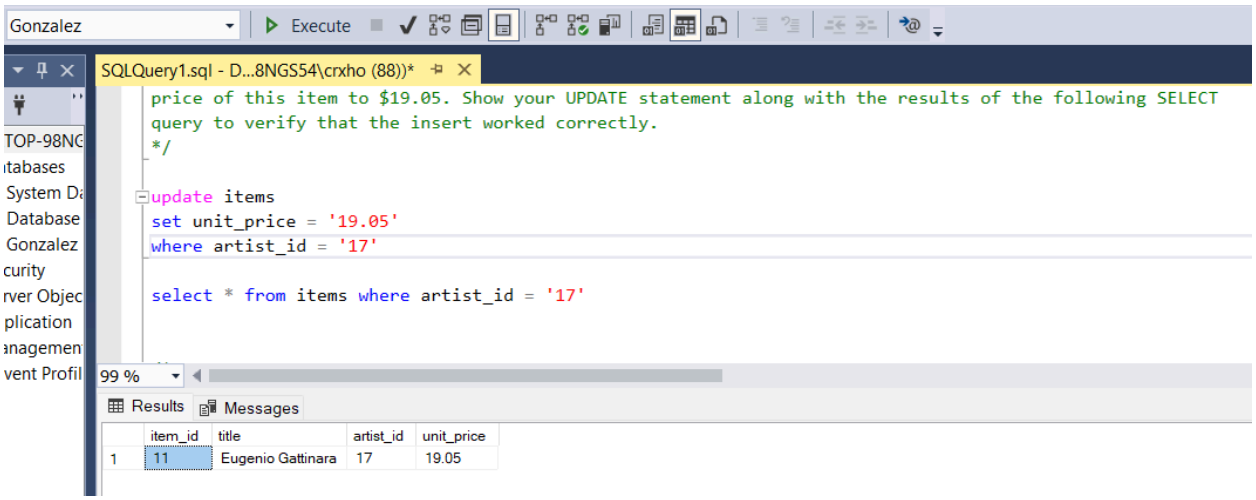

Explanation of Query/ Results

Query displays item ID, title, artist ID, and unit price. A new entry was made into the items tables with the values stated in the question. There was a constraint of a Primary Key and Foreign Key in which the artist ID had to match as well in the artists table. Results display the newly inserted record by using the statement to find item IDs that are greater than 9.

**<u>Query #22:</u>** Create a statement to update the record inserted in the previous step to change the unit price of this item to $19.05. Show your UPDATE statement along with the results of the following SELECT query to verify that the insert worked correctly:
select * from items where artist_id = '17'

*UPDATE items*
*SET unit_price = '19.05'*
*WHERE artist_id = '17'*

*SELECT * FROM items WHERE artist_id = '17'*

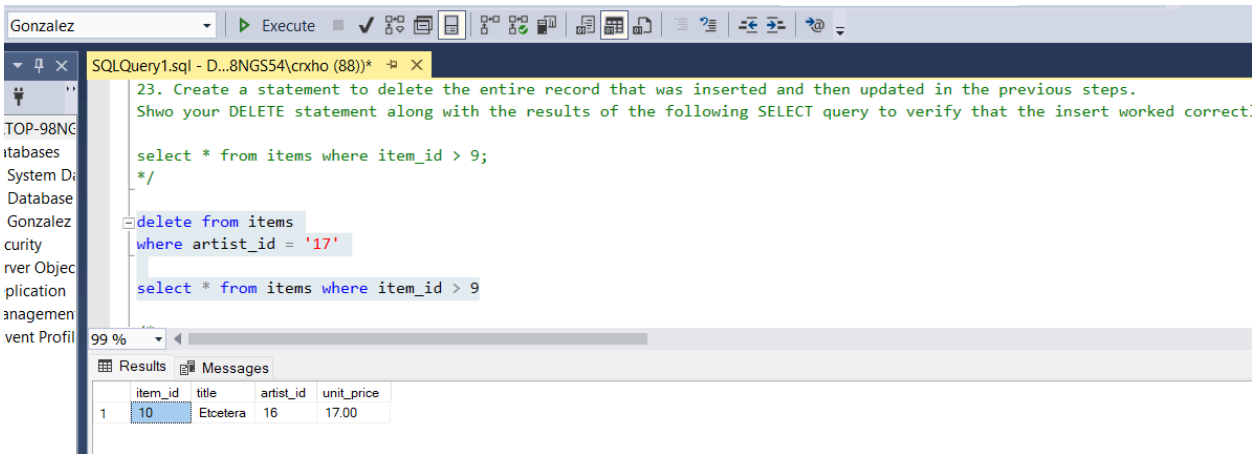



<u>Explanation of Query/ Results</u>
Query displays an updated record from the newly inserted item title in the items table. Updated record by changing the price from $23.51 to $19.05. Checked to make sure the record was updated by using the select statement to find artist ID = 17.

**<u>Query #23:</u>** 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 > 9;

*DELETE from items*
*WHERE artist_id = '17'*

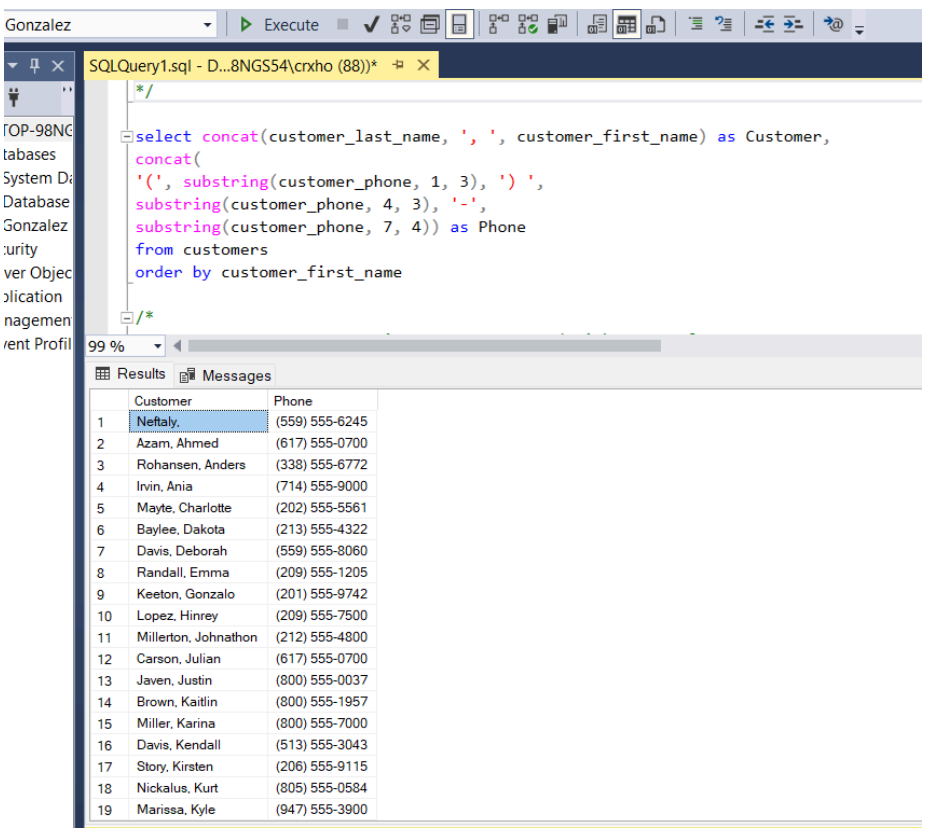
*SELECT * FROM items WHERE item_id > 9*




<u>Explanation of Query/ Results</u>
Query deletes newly inserted record from previous question. Deleted the record from the items table where the artist ID is 17. Checked the items table to make sure the item was removed successfully.

**<u>Query #24:</u>** 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 9095595443 would be output with parentheses, spaces, and hyphens, like this: (909) 559-5443. Sort by first 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, 7, 4)
) AS Phone
FROM customers
ORDER BY customer_first_name
```




| | Customer | Phone |
|---|---|---|
| 1 | Neftaly, | (559) 555-6245 |
| 2 | Azam, Ahmed | (617) 555-0700 |
| 3 | Rohansen, Anders | (338) 555-6772 |
| 4 | Irvin, Ania | (714) 555-9000 |
| 5 | Mayte, Charlotte | (202) 555-5561 |
| 6 | Baylee, Dakota | (213) 555-4322 |
| 7 | Davis, Deborah | (559) 555-8060 |
| 8 | Randall, Emma | (209) 555-1205 |
| 9 | Keeton, Gonzalo | (201) 555-9742 |
| 10 | Lopez, Hinrey | (209) 555-7500 |
| 11 | Millerton, Johnathon | (212) 555-4800 |
| 12 | Carson, Julian | (617) 555-0700 |
| 13 | Javen, Justin | (800) 555-0037 |
| 14 | Brown, Kaitlin | (800) 555-1957 |
| 15 | Miller, Karina | (800) 555-7000 |
| 16 | Davis, Kendall | (513) 555-3043 |
| 17 | Story, Kirsten | (206) 555-9115 |
| 18 | Nickalus, Kurt | (805) 555-0584 |
| 19 | Marissa, Kyle | (947) 555-3900 |

<u>Explanation of Query/ Results</u>
Query displays columns customer and phone. Customer displays customer names in the format (last name, first name). Customer's phone numbers are formatted as well using the SUBSTRING operations to display the numbers in a readable manner. Results are sorted by customer first name using the customers table.

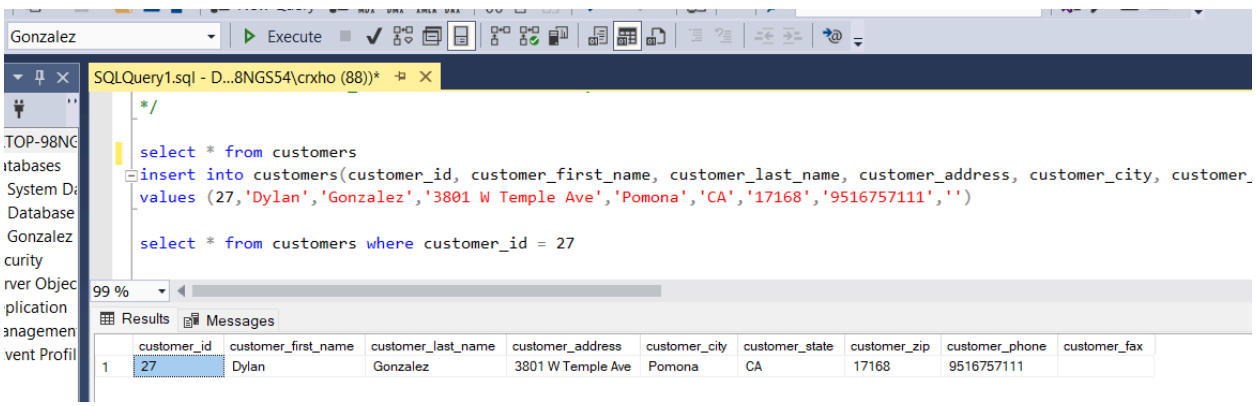
**<u>Query #25:</u>** Create a statement to insert a new record with your values:
your customer id, first name, last name, address, city, state, zip code and fax number.

Note: Use your real name and (3801 W Temple Ave, Pomona, CA 17168) 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 = 27;

*SELECT * FROM customers*
*INSERT INTO customers(customer_id, customer_first_name, customer_last_name, customer_address, customer_city, customer_state, customer_zip, customer_phone, customer_fax)*
*VALUES (27,'Dylan','Gonzalez','3801 W Temple Ave','Pomona','CA','17168','9516757111','')*

*SELECT * FROM customers WHERE customer_id = 27*

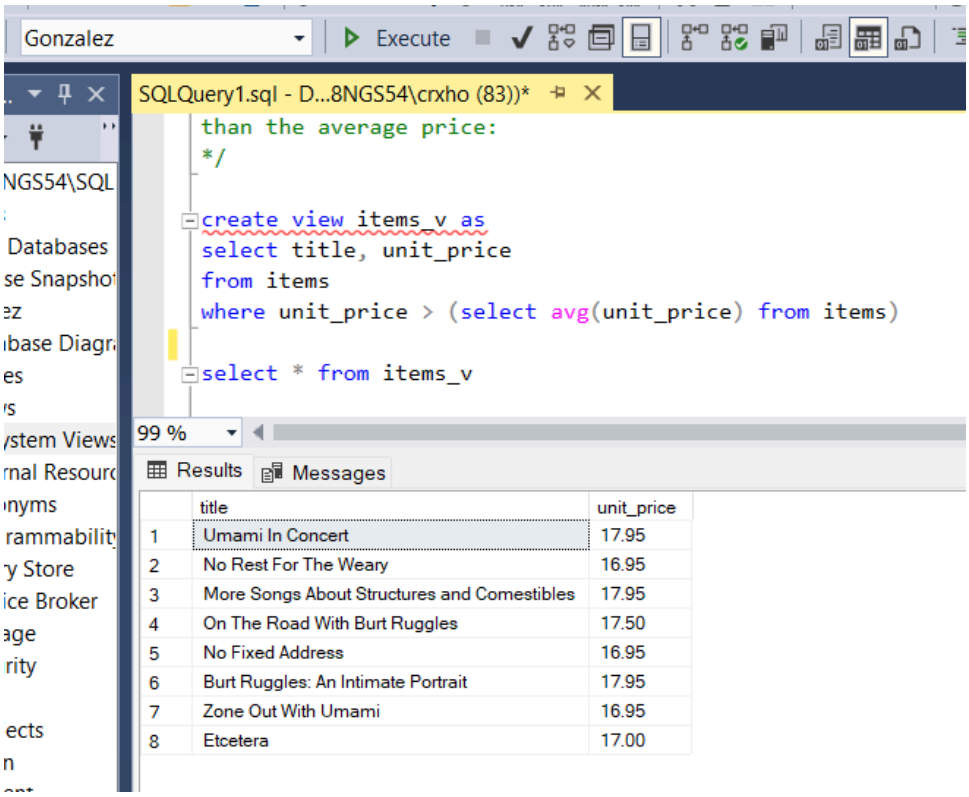



<u>Explanation of Query/ Results</u>
Query displays all details from the customers table. Inserted a new record in the customers table using my information, besides the address which is the Universitys'. Results show detail of newly inserted records by selecting only customer ID = 27.

**<u>Query #26:</u>** Create a view that selects every title in the "item" table with a price higher than the average price

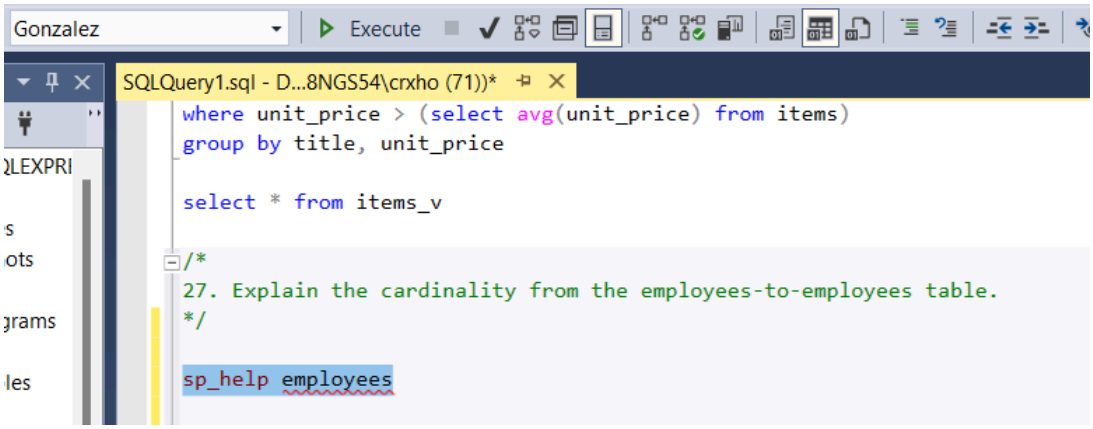
*CREATE VIEW items_v AS*
*SELECT title, unit_price*
*FROM items*
*WHERE unit_price > (SELECT avg(unit_price) FROM items)*

*SELECT * FROM items_v*




<u>Explanation of Query/ Results</u>
Query displays item title and the unit price which is higher than the average standard price. Created a view titled “item_v” where it calculates the average price for all items. Results are only prices that are higher than the average from the items table.

**<u>Query #27</u>:** Explain the cardinality from the employees-to-employees table.




| | constraint_type | constraint_name | delete_action | update_action | status_enabled | status_for_replication | constraint_keys |
|---|---|---|---|---|---|---|---|
| 1 | FOREIGN KEY | emp_fk_mgr | No Action | No Action | Enabled | Is_For_Replication | manager_id |
| 2 | | | | | | | REFERENCES Gonzalez.dbo.employees (employee_id) |
| 3 | PRIMARY KE... | employees_pk | (n/a) | (n/a) | (n/a) | (n/a) | employee_id |

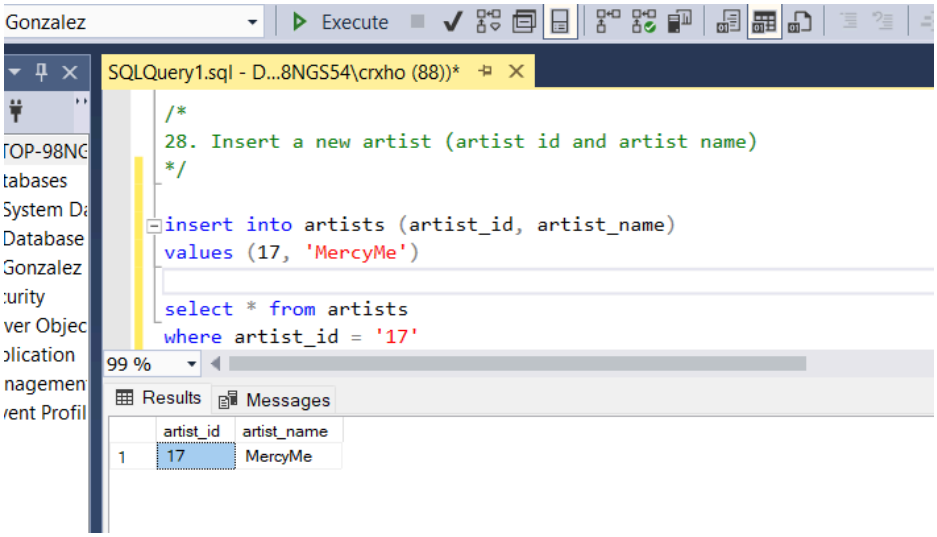
<u>Explanation of Query/ Results</u>

Using the command "sp_help", I was able to view the cardinality of the employees-to-employes table. This is a recusive one-to-many relationship. To further explain, there are employees who have one manager and one manager has many employees. Manager ID is a recursive foreign key, meaning the key refers back to the primary key in the same table where the child and parent table are the same.

**<u>Query #28:</u>** Insert a new artist (artist id and artist name)

*INSERT INTO artists (artist_id, artist_name)*
*VALUES (17, 'MercyMe')*

*SELECT * FROM artists*
*WHERE artist_id = '17'*




<u>Explanation of Query/ Results</u>
Query displays artist ID and artist name. Inserted a new record into the artists table with chosen values as seen in the visual above. Checked to see if the record was inserted into the table by selecting the artist ID = 17.

# Code for Creating Database

```
CREATE TABLE customers
(
customer_id          INT          ,
customer_first_name   VARCHAR(20),
customer_last_name    VARCHAR(20)        NOT NULL,
customer_address      VARCHAR(50)        NOT NULL,
customer_city          VARCHAR(20)       NOT NULL,
customer_state       CHAR(2)          NOT NULL,
customer_zip         CHAR(5)          NOT NULL,
customer_phone       CHAR(10)       NOT NULL,
customer_fax         CHAR(10),
CONSTRAINT customers_pk
  PRIMARY KEY (customer_id)
);


CREATE TABLE artists
(
artist_id          INT                NOT NULL,
artist_name   VARCHAR(30),
CONSTRAINT artist_pk
  PRIMARY KEY (artist_id)
);


CREATE TABLE items
(
item_id          INT                NOT NULL,
title        VARCHAR(50)       NOT NULL,
artist_id          INT     NOT NULL,
unit_price    DECIMAL(9,2)  NOT NULL,
CONSTRAINT items_pk
          PRIMARY KEY (item_id),
CONSTRAINT items_fk_artists
          FOREIGN KEY (artist_id) REFERENCES artists (artist_id)
);
```

```
CREATE TABLE employees
(
employee_id        INT          NOT NULL,
last_name          VARCHAR(20)      NOT NULL,
first_name         VARCHAR(20)      NOT NULL,
manager_id         INT,
CONSTRAINT employees_pk
  PRIMARY KEY (employee_id),
CONSTRAINT emp_fk_mgr FOREIGN KEY (manager_id) REFERENCES
employees(employee_id) );


CREATE TABLE orders
(
order_id         INT     NOT NULL,
customer_id      INT             NOT NULL,
order_date       DATE            NOT NULL,
shipped_date     DATE,
employee_id   INT,
CONSTRAINT orders_pk
        PRIMARY KEY (order_id),
CONSTRAINT orders_fk_customers
        FOREIGN KEY (customer_id) REFERENCES customers (customer_id),
CONSTRAINT orders_fk_employees
        FOREIGN KEY (employee_id) REFERENCES employees (employee_id)
);


CREATE TABLE order_details
(
order_id         INT             NOT NULL,
item_id          INT             NOT NULL,
order_qty        INT             NOT NULL,
CONSTRAINT order_details_pk
        PRIMARY KEY (order_id, item_id),
CONSTRAINT order_details_fk_orders
        FOREIGN KEY (order_id)
        REFERENCES orders (order_id),
CONSTRAINT order_details_fk_items
        FOREIGN KEY (item_id)
```

```
        REFERENCES items (item_id)
);

-- insert rows into tables

INSERT INTO customers VALUES

(1,'Oliva','Jachson','1555 W Lane Ave','Columbus','OH','43221','6145554435','6145553928'),
(2,'Emma','Randall','11 E Rancho Madera
Rd','Madison','WI','53707','2095551205','2095552262'),
(3,'Johnathon','Millerton','60 Madison Ave','New York','NY','10010','2125554800',NULL),
(4,'Charlotte','Mayte','2021 K Street Nw','Washington','DC','20006','2025555561',NULL),
(5,'Kendall','Davis','4775 E Miami River Rd','Cleves','OH','45002','5135553043',NULL),
(6,'Kaitlin','Brown','3250 Spring Grove
Ave','Cincinnati','OH','45225','8005551957','8005552826'),
(7,'Lily','Chaddick','9022 E Merchant Wy','Fairfield','IA','52556','5155556130',NULL),
(8,'Deborah','Davis','415 E Olive Ave','Fresno','CA','93728','5595558060',NULL),
(9,'Karina','Miller','882 W Easton Wy','Los Angeles','CA','90084','8005557000',NULL),
(10,'Kurt','Nickalus','28210 N Avenue
Stanford','Valencia','CA','91355','8055550584','055556689'),
(11,'Hinrey','Lopez','7833 N Ridge Rd','Sacramento','CA','95887','2095557500','2095551302'),
(12,'Anders','Rohansen','12345 E 67th Ave NW','Takoma
Park','MD','24512','3385556772',NULL),
(13,'','Neftaly','2508 W Shaw Ave','Fresno','CA','93711','5595556245',NULL),
(14,'Gonzalo','Keeton','12 Daniel Road','Fairfield','NJ','07004','2015559742',NULL),
(15,'Ania','Irvin','1099 N Farcourt St','Orange','CA','92807','7145559000',NULL),
(16,'Dakota','Baylee','1033 N Sycamore Ave.','Los Angeles','CA','90038','2135554322',NULL),
(17,'Samuel','Jacobsen','3433 E Widget Ave','Palo Alto','CA','92711','4155553434',NULL),
(18,'Justin','Javen','828 S Broadway','Tarrytown','NY','10591','8005550037',NULL),
(19,'Kyle','Marissa','789 E Mercy Ave','Phoenix','AZ','85038','9475553900',NULL),
(20,'Mohammad','Ali','Five Lakepointe Plaza, Ste
500','Charlotte','NC','28217','7045553500',NULL),
(21,'Marvin','Quintin','2677 Industrial Circle
Dr','Columbus','OH','43260','6145558600','6145557580'),
(22,'Rashad','Holbrooke','3467 W Shaw Ave
#103','Fresno','CA','93711','5595558625','5595558495'),
(23,'Theo','Hernandez','627 Aviation Way','Manhatttan Beach','CA','90266','3105552732',NULL),
(24,'Julian','Carson','372 San Quentin','San Francisco','CA','94161','6175550700',NULL),
(25,'Kirsten','Story','2401 Wisconsin Ave NW','Washington','DC','20559','2065559115',NULL),
(26,'Ahmed','Azam','372 San Quentin','San Francisco','CA','94161','6175550700',NULL);
```

```
INSERT INTO artists(artist_id,artist_name) VALUES

(10, 'Umani'),
(11, 'The Ubernerds'),
(12, 'No Rest For The Weary'),
(13, 'Burt Ruggles'),
(14, 'Sewed the Vest Pocket'),
(15, 'Jess & Odie'),
(16, 'Onn & Onn');

INSERT INTO items (item_id,title,artist_id,unit_price) VALUES

(1,'Umami In Concert',10,17.95),
(2,'Race Car Sounds',11,13),
(3,'No Rest For The Weary',12,16.95),
(4,'More Songs About Structures and Comestibles',12,17.95),
(5,'On The Road With Burt Ruggles',13,17.5),
(6,'No Fixed Address',14,16.95),
(7,'Rude Noises',15,13),
(8,'Burt Ruggles: An Intimate Portrait',13,17.95),
(9,'Zone Out With Umami',10,16.95),
(10,'Etcetera',16,17);
 INSERT INTO employees VALUES
(1,'Garcia',      'Mia', null),
(2,'Moore',      'Aria', 1),
(3,'Lee', 'Layla', 2),
(9,'Locario',  'Paulo',1),
(8,'Leary', 'Rhea',9),
(4,'Hernandez','Olivia',9),
(5,'Aaronsen', 'Robert',4),
(6,'White',   'Asher',8),
(7,'clark',    'Thomas',2);

INSERT INTO orders VALUES

(19, 1, '2012-10-23', '2012-10-28', 6),
(29, 8, '2012-11-05', '2012-11-11', 6),
(32, 11, '2012-11-10', '2012-11-13', NULL),
(45, 2, '2012-11-25', '2012-11-30', NULL),
(70, 10, '2012-12-28', '2013-01-07', 5),
```

(89, 22, '2013-01-20', '2013-01-22', 7),
(97, 20, '2013-01-29', '2013-02-02', 5),
(118, 3, '2013-02-24', '2013-02-28', 7),
(144, 17, '2013-03-21', '2013-03-29', NULL),
(158, 9, '2013-04-04', '2013-04-20', NULL),
(165, 14, '2013-04-11', '2013-04-13', NULL),
(180, 24, '2013-04-25', '2013-05-30', NULL),
(231, 15, '2013-06-14', '2013-06-22', NULL),
(242, 23, '2013-06-24', '2013-07-06', 3),
(264, 9, '2013-07-15', '2013-07-18', 6),
(298, 18, '2013-08-18', '2013-09-22', 3),
(321, 2, '2013-09-09', '2013-10-05', 6),
(381, 7, '2013-11-08', '2013-11-16', 7),
(413, 17, '2013-12-05', '2014-01-11', 7),
(442, 5, '2013-12-28', '2014-01-03', 5),
(479, 1, '2014-01-30', '2014-03-03', 3),
(491, 16, '2014-02-08', '2014-02-14', 5),
(523, 3, '2014-03-07', '2014-03-15', 3),
(548, 2, '2014-03-22', '2014-04-18', NULL),
(550, 17, '2014-03-23', '2014-04-03', NULL),
(601, 16, '2014-04-21', '2014-04-27', NULL),
(607, 20, '2014-04-25', '2014-05-04', NULL),
(624, 2, '2014-05-04', '2014-05-09', NULL),
(627, 17, '2014-05-05', '2014-05-10', NULL),
(630, 20, '2014-05-08', '2014-05-18', 7),
(651, 12, '2014-05-19', '2014-06-02', 7),
(658, 12, '2014-05-23', '2014-06-02', 7),
(687, 17, '2014-06-05', '2014-06-08', NULL),
(693, 9, '2014-06-07', '2014-06-19', NULL),
(703, 19, '2014-06-12', '2014-06-19', 7),
(778, 13, '2014-07-12', '2014-07-21', 7),
(796, 17, '2023-07-19', '2014-07-26', 5),
(800, 19, '2023-07-21', '2014-07-28', NULL),
(802, 2, '2023-07-21', '2014-07-31', NULL),
(824, 1, '2024-08-01', NULL, NULL),
(827, 18, '2024-08-02', NULL, NULL),
(829, 9, '2024-08-02', NULL, NULL);

INSERT INTO order_details VALUES

(381,1,1),
(601,9,1),
(442,1,1),
(523,9,1),
(630,5,1),
(778,1,1),
(693,10,1),
(118,1,1),
(264,7,1),
(607,10,1),
(624,7,1),
(658,1,1),
(800,5,1),
(158,3,1),
(321,10,1),
(687,6,1),
(827,6,1),
(144,3,1),
(479,1,2),
(630,6,2),
(796,5,1),
(97,4,1),
(601,5,1),
(800,1,1),
(29,10,1),
(70,1,1),
(165,4,1),
(180,4,1),
(231,10,1),
(413,10,1),
(491,6,1),
(607,3,1),
(651,3,1),
(703,4,1),
(802,3,1),
(824,7,2),
(829,1,1),
(550,4,1),

(796,7,1),
(693,6,1),
(29,3,1),
(32,7,1),
(242,1,1),
(298,1,1),
(479,4,1),
(548,9,1),
(627,9,1),
(778,3,1),
(19,5,1),
(89,4,1),
(242,6,1),
(264,4,1),
(550,1,1),
(693,7,3),
(824,3,1),
(829,5,1),
(829,9,1);