SQL CODE
WRITE-UP

MOST EXPENSIVE ORDER

NOV 2023

Joins, Group By & Order By

LinkedIn Coding Challenge

Intermediate

A database with many tables has foreign keys that are designed to join the tables during SQL queries.

Find the most expensive order by joining the following tables:

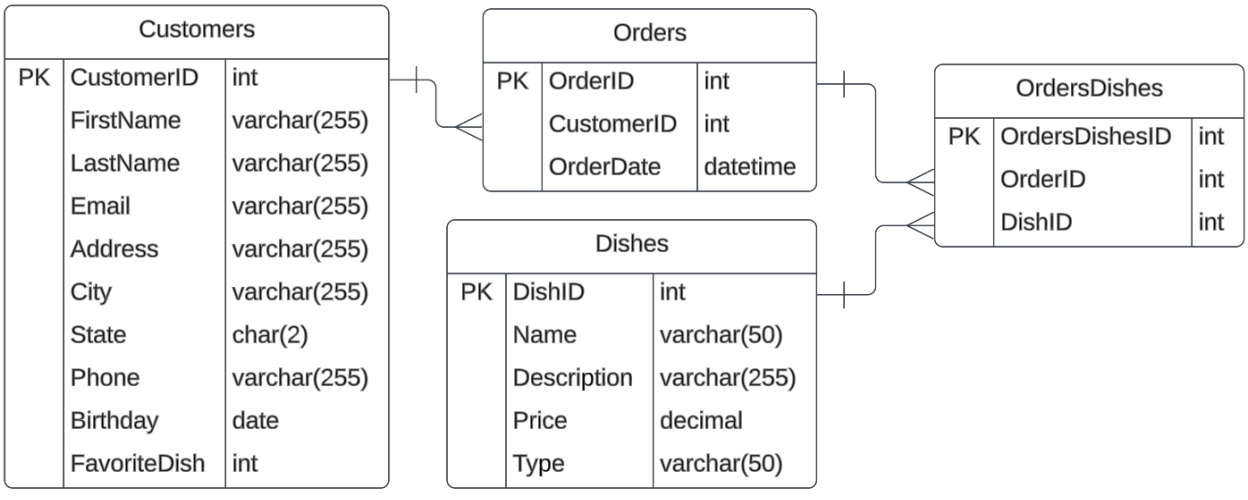
Orders.OrderID → OrdersDishes.OrderID

OrdersDishes.DishID → Dishes.DishID

Orders.CustomerID → Customers.CustomerID

INPUT FORMAT

The source tables are ***ORDERS, ORDERSDISHES, DISHES*** and ***CUSTOMERS.*** The tables are connected as follows:



CONSTRAINTS

The ***DISHES*** table doesn't have a quantity column so each dish in an order has a quantity of one

CODE SOLUTION

```
SELECT Orders.ORDERID, Customers.FIRSTNAME, Customers.LASTNAME,
SUM(Dishes.PRICE) AS ORDERTOTAL

FROM Orders

JOIN OrdersDishes

ON Orders.ORDERID = OrdersDishes.ORDERID

JOIN Dishes

ON OrdersDishes.DISHID = Dishes.DISHID

JOIN Customers

ON Orders.CUSTOMERID = Customers.CUSTOMERID

GROUP BY Orders.ORDERID

ORDER BY ORDERTOTAL DESC

LIMIT 1
```

SOLUTION PROCESS

- Select function: ***ORDERS*** is the primary table choice because it has the greatest many-to-one relationships connected to it. Join functions allow extraction of customer names and summed dish price data renamed as ORDERTOTAL.
- Group By: Results are grouped by first column in Select function to ensure that ORDERTOTAL is summed per each individual order
- Limit function: Only top result is requested. Couple with Order by function in descending order, the code produces such a result. Alternative, the same output is derived from using a `SELECT TOP [#]` function.

OUTPUT

```
_____________________________________________

| ORDERID | FIRSTNAME | LASTNAME    | ORDERTOTAL |

_____________________________________________

| 787     | Yves      | Dell'Abbate | 58.95      |

_____________________________________________
```