# DAD 220 Cardinality and Targeted Data Template

Replace the bracketed text in this template with your screenshots and responses. Then submit it to the Module Four Lab for submission, grading, and feedback. Screenshots should be sized to approximately one quarter of a page. Written responses should be in complete sentences. Rename this document by adding your last name to the file name before you submit.

1. **Retrieve employee tuples and identify the number of employees** in San Francisco and New York.

A screenshot of a computer

Description automatically generated

SELECT firstName, lastName, jobTitle, offices.city

FROM employees

INNER JOIN offices ON employees.officeCode = offices.officeCode WHERE state='CA';

&

SELECT firstName, lastName, jobTitle, offices.city

FROM employees

INNER JOIN offices ON employees.officeCode = offices.officeCode WHERE state='NY';

1. **Retrieve order details** for orderNumber 10330, 10338, and 10194 and **identify** what **type of cardinality** this represents in the entity relationship model.

One-to-One (1:1): Each order number should uniquely identify an order, and each order should have a unique order number.

One-to-Many (1:N): If, for example, you have a relationship between the orders table and another table (e.g., orderDetails), where each order can have multiple order details, it would be a one-to-many relationship. Each order in the orders table could correspond to multiple records in the orderDetails table.

Many-to-Many (M:N): If there's a many-to-many relationship, it would typically involve an associative table. For example, if each order can have multiple products, and each product can be in multiple orders, you might have an associative table like orderProducts linking orders and products.

A black screen with white text

Description automatically generated

SELECT \*

FROM orders

WHERE orderNumber IN (10330, 10338, 10194);

1. **Delete records** from the payments table where the customer number equals 103.

A screen shot of a computer

Description automatically generated

SELECT \*

FROM payments

WHERE customerNumber = 103;

A black screen with white text

Description automatically generated

DELETE FROM payments

WHERE customerNumber = 103;

&

SELECT \*

FROM payments

WHERE customerNumber = 103;

1. **Retrieve customer records** for sales representative Barry Jones and **identify** if the **relationships** are one-to-one or one-to-many**.**

The results below show that Barry jones is repeated but each row has a different customer. This indicates a one-to-many relationship so basically Barry Jones is associated with multiple customers but only one sales rep.

A screen shot of a computer

Description automatically generated

SELECT CONCAT(employees.firstName, '', employees.lastName) AS SALES\_REP, customers.customerName AS CUSTOMER

FROM employees

INNER JOIN customers ON employees.employeeNumber=customers.salesRepEmployeeNumber

WHERE employees.firstName = 'Barry' AND employees.lastName = 'Jones';

1. **Retrieve records** for customers who reside in Massachusetts and **identify** **their sales rep and the relationship of entities**. Identify if these entities demonstrate one-to-one or many-to-many relationships.

The relation ship between the sales representatives seem to be one-to-many because for each sales rep there are several customers but only one sales rep per customer.

A black screen with white text

Description automatically generated

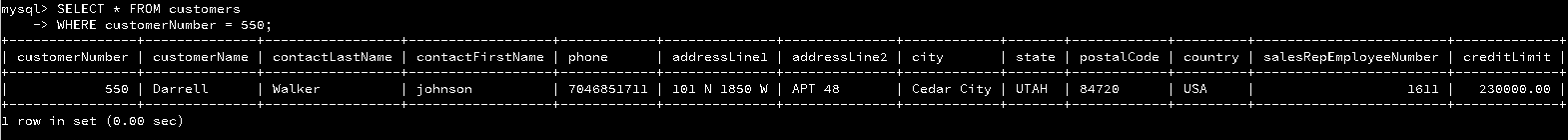
SELECT CONCAT (employees.firstName, '', employees.lastName) AS SALES\_REP, customers.customerName AS CUSTOMER, customers.state

FROM employees

INNER JOIN customers ON employees.employeeNumber = customers.salesRepEmployeeNumber

WHERE UPPER(customers.state) = 'MA';

1. **Add one customer record** with your last name using an INSERT statement. You may use the name of a celebrity or fictional character if you don’t use your own name.



SELECT \* FROM customers

WHERE customerNumber = 550;

1. **Reflection**
   1. **Define how cardinality is applied** to the databases you’ve been working with and why different numbers of records returned from the different offices.

In the databases we're talking about, there's a thing called "cardinality," which is like a way of saying how different types of information are connected. Imagine you have a list of workers (employees) and a list of offices where they work. The relationship between these lists is like this: for every worker, there can be one or more offices they belong to, but each office is linked to only one worker. This means it's not a one-to-one connection; it's more like one worker to many offices. When we look at these lists using SQL queries, we might see different numbers of records because one worker can be part of the same office as other workers, and that's why we get various results when we check the information. It's like how one teacher (employee) can be in charge of many students (offices), but each student is assigned to only one teacher.

* 1. **Compare and contrast** the different **queries** you ran and how cardinality applies to them.

Query 1: Employee and Office Information for San Francisco

In this query, we're exploring the connection between employees and the offices they work in. Each worker is tied to a specific office, but an office can house multiple workers. It's a bit like how people in a company might be connected to different offices. The comparison to the first question is evident – we're still looking at workers and offices, but this time, it's narrowed down to San Francisco. Rather than focusing on one person, we're examining the whole location and the employees associated with it.

Query 2: Order Details for Specific Numbers

This query is straightforward – we're simply looking at specific orders. Unlike the other questions, there isn't a big connection between these orders; each order is kind of its own thing. It's comparable to looking for specific items on a list without considering their relationships. While other queries explored connections between groups or locations, this one is more about identifying particular orders.

Query 3: Customer Records for Barry Jones

Moving on to customer records, this query revolves around the link between customers and their sales representatives. Each customer is assigned to a single salesperson, but a salesperson may have several customers. Drawing a parallel to the first question, it's akin to examining workers and offices, but this time, we're focused on customers and a specific sales representative – Barry Jones. Rather than exploring a place, we're investigating customers tied to a particular salesperson.

Query 4: Customer Records in Massachusetts

Similar to the third query, we're still looking at customers and their sales representatives. Each customer remains associated with one salesperson, while a salesperson might oversee multiple customers. Comparing this to the third question, we're now interested in customers from a specific location – Massachusetts. Instead of concentrating on one person (Barry Jones), the emphasis shifts to customers from a particular area, showcasing the flexibility of these queries in extracting meaningful information based on different criteria.

* 1. **Describe two** of the crucial **benefits** **of cardinality** in this type of database.

Cardinality is super important in databases for a couple of big reasons. First off, it helps keep our data in check – it ensures that when we say one thing is connected to another, it really is. Like, if we're talking about workers and offices, it makes sure each worker is truly linked to one office, but an office can have lots of workers. This keeps our data reliable and consistent. Another cool thing is that it helps us search for info in a smart way. If we know how things are connected, we can ask the database questions in a way that gets us exactly what we need without wasting time or getting extra stuff we don't want. So, it's like the guardian of our data honesty and the guide for asking the database the right questions.