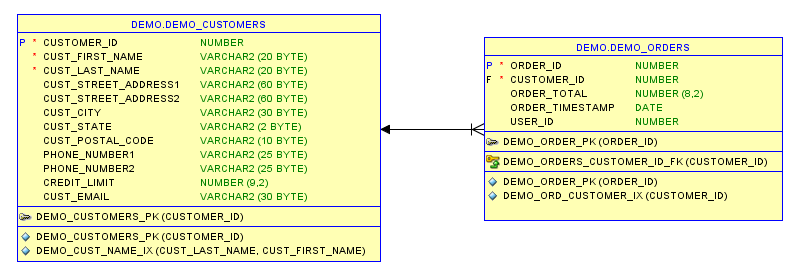
SQL

1. If TableA has 100 rows and TableB has 10 rows, how many rows would be retrieved from the following query:
   1. \_100\_\_\_

**Select** \*

**From** TableA, TableB

1. Given the following ERD:



* 1. What is the cardinality between DEMO\_CUSTOMERS and DEMO\_ORDERS?

12 in DEMO\_CUSTOMERS, 5 in DEMO\_ORDERS

* 1. What is the primary key of DEMO\_CUSTOMERS?

CUSTOMER\_ID

* 1. What foreign key relates DEMO\_CUSTOMERS and DEMO\_ORDERS?

CUSTOMER\_ID

* 1. What type of constraint is DEMO\_CUST\_NAME\_IX?

check

1. Given the above ERD:
   1. Write 15 INSERT statements to create test Customer records. (Do not copy paste. Hand-writing these statements over and over will help you recall the syntax.

INSERT INTO demo\_customers VALUES (10050,’guy’,’dude’,’AZ’,500,’email@email.com’)

INSERT INTO demo\_customers VALUES (10055,’dude’,’man’,’WA’,1500,’emil@email.com’)

INSERT INTO demo\_customers VALUES (10060,’girl’,’girl’,’IN’,1000,’emale@email.com’)

INSERT INTO demo\_customers VALUES (10065,’guy’,’dude’,’IN’,1000,’emall@email.com’)

INSERT INTO demo\_customers VALUES (10070,’guy’,’dude’,’OH’,1500,’emalt@email.com’)

INSERT INTO demo\_customers VALUES (10075,’guy’,’dude’,’OH’,1500,’emael@email.com’)

INSERT INTO demo\_customers VALUES (10080,’guy’,’dude’,’VA’,1500,’emaal@email.com’)

INSERT INTO demo\_customers VALUES (10085,’guy’,’dude’,’VA’,2000,’eail@email.com’)

INSERT INTO demo\_customers VALUES (10090,’guy’,’dude’,’IL’,750,’emil@email.com’)

INSERT INTO demo\_customers VALUES (10095,’guy’,’dude’,’IL’,750,’emal@email.com’)

INSERT INTO demo\_customers VALUES (10100,’guy’,’dude’,’AZ’,500,’emai@email.com’)

INSERT INTO demo\_customers VALUES (10105,’guy’,’dude’,’AZ’,750,’mail@email.com’)

INSERT INTO demo\_customers VALUES (10100,’guy’,’dude’,’AZ’,500,’il@email.com’)

INSERT INTO demo\_customers VALUES (10115,’guy’,’dude’,’WA’,1000,’ail@email.com’)

INSERT INTO demo\_customers VALUES (10120,’guy’,’dude’,’TX’,500,’mlaie@email.com’)

* 1. Write 20 INSERT statements to create test Order data. Make sure some Customers do not reference any orders! (Do not copy paste. Hand-writing these statements over and over will help you recall the syntax.)

INSERT INTO demo\_orders VALUES (10060,10000, 11991,’12-MAY-2017’)

INSERT INTO demo\_orders VALUES (10065,10020, 19542,’12-JUN-2017’)

INSERT INTO demo\_orders VALUES (10070,10020, 15675,’12-MAY-2017’)

INSERT INTO demo\_orders VALUES (10075,10020, 16777,’12-MAY-2017’)

INSERT INTO demo\_orders VALUES (10080,10000, 18776,’12-MAY-2017’)

INSERT INTO demo\_orders VALUES (10085,10000, 18768,’12-MAY-2017’)

INSERT INTO demo\_orders VALUES (10090,10020, 15345,’12-MAY-2017’)

INSERT INTO demo\_orders VALUES (10095,10020, 12344,’04-JUN-2017’)

INSERT INTO demo\_orders VALUES (10100,10015, 14452,’05-JUN-2017’)

INSERT INTO demo\_orders VALUES (10105,10000, 54353,’06-JUN-2017’)

INSERT INTO demo\_orders VALUES (10110,10000, 54535,’07-JUN-2017’)

INSERT INTO demo\_orders VALUES (10115,10000, 15345,’08-JAN-2017’)

INSERT INTO demo\_orders VALUES (10120,10000, 12353,’09-JAN-2017’)

INSERT INTO demo\_orders VALUES (10125,10000, 43434,’10-JAN-2017’)

INSERT INTO demo\_orders VALUES (10130,10020, 44444,’12-MAY-2017’)

INSERT INTO demo\_orders VALUES (10135,10020, 11111,’12-MAY-2017’)

INSERT INTO demo\_orders VALUES (10140,10020, 22222,’12-MAY-2017’)

INSERT INTO demo\_orders VALUES (10145,10015, 33333,’12-MAY-2017’)

INSERT INTO demo\_orders VALUES (10150,10015, 21353,’12-MAY-2017’)

INSERT INTO demo\_orders VALUES (10155,10015, 12345,’12-MAY-2017’)

* 1. Write 10 UPDATE statements to change some of the values is each table.

UPDATE Customer SET cust\_first\_name = 'Luas' WHERE CustomerID = 1;

UPDATE Customer SET cust\_first\_name = 'Lance' WHERE CustomerID = 2;

UPDATE Customer SET cust\_first\_name = 'Lady', credit\_limit = 10000 WHERE CustomerID = 1;

UPDATE Customer SET cust\_first\_name = 'Las' WHERE CustomerID = 3;

UPDATE Customer SET cust\_first\_name = 'Louis' WHERE CustomerID = 3;

UPDATE Customer SET cust\_first\_name = 'Larry' WHERE CustomerID = 2;

UPDATE Customer SET cust\_first\_name = 'Luas', cust\_last\_name = ‘Labe’ WHERE CustomerID = 1;

UPDATE Customer SET cust\_first\_name = 'Luas' WHERE CustomerID = 3;

UPDATE Customer SET cust\_first\_name = 'Luas' WHERE CustomerID = 2;

UPDATE Customer SET cust\_first\_name = 'Luas' WHERE CustomerID = 1;

* 1. Write 5 DELETE statements to remove Customers.

DELETE FROM customer WHERE customerid = 1;

DELETE FROM customer WHERE customerid = 1;

DELETE FROM customer WHERE customerid = 1;

DELETE FROM customer WHERE customerid = 1;

DELETE FROM customer WHERE customerid = 1;

* 1. Write 10 DELETE statements to remove some Orders.

DELETE FROM demo\_orders WHERE customerid = 1040;

DELETE FROM demo\_orders WHERE customerid = 1035;

DELETE FROM demo\_orders WHERE customerid = 1030;

DELETE FROM demo\_orders WHERE customerid = 1025;

DELETE FROM demo\_orders WHERE customerid = 1020;

DELETE FROM demo\_orders WHERE customerid = 1015;

DELETE FROM demo\_orders WHERE customerid = 1010;

DELETE FROM demo\_orders WHERE customerid = 1005;

DELETE FROM demo\_orders WHERE customerid = 995;

DELETE FROM demo\_orders WHERE customerid = 1000;

* 1. Write a query that shows all customers with a credit limit higher than 2,000.

SELECT \* FROM demo\_customers WHERE credit > 2000;

* 1. Write a query that shows all customers who have only one phone number.

SELECT \* FROM customers WHERE secondaryphone IS NULL;

* 1. Write a query that finds any customer who does not have a valid email address format.

SELECT \* FROM customers WHERE email LIKE ‘%@%.com%’;

* 1. Write a query that finds all customers who have spent more than $25,000.

SELECT customerid, SUM(total) FROM invoice WHERE total > 1 GROUP BY customerid HAVING SUM(total) > 25000;

* 1. Write a query that finds all customers who have spent more than $5,000 on a single order.

SELECT customerid, total FROM invoice WHERE total > 5000;

* 1. Write a query that finds the average ticket price for an order.

SELECT AVG(total) FROM invoice;

* 1. Write a query that shows all customers who have never placed an order.

SELECT customer\_id FROM demo\_orders GROUP BY customer\_id;

* 1. Write a query that shows all of the orders from San Antonio, TX within the last 3 months.

SELECT \* FROM demo\_orders WHERE order\_timestamp > '25-FEB-17';

* 1. Write a query that shows the customers ranked in order of their credit limit.

SELECT \* FROM demo\_customers ORDER BY credit\_limit DESC;

* 1. Write a query that shows the top three highest ticket orders.

SELECT \* FROM demo\_orders WHERE ROWNUM <= 3 ORDER BY credit\_limit DESC;

* 1. Write a query that finds the total sales for the last three months.

SELECT SUM(order\_total) FROM demo\_orders WHERE order\_timestamp > '25-FEB-17';

* 1. Write a query that shows all customers, but with their names in only one result column.

SELECT CUST\_FIRST\_NAME FROM demo\_customers

UNION

SELECT CUST\_LAST\_NAME FROM demo\_customers;

* 1. Write a query that shows the total sales last year for each User\_ID (the salesperson).

SELECT customer\_id, sum(order\_total) FROM demo\_orders GROUP BY customer\_id;

* 1. Write a query that finds all customers with the (703) area code.

SELECT customerid FROM customer WHERE phone LIKE '%(703)%';

* 1. Write a query that finds out how many customers from VA that haven’t ordered in 3 months

SELECT customerid FROM invoice WHERE invoicedate > '01-FEB-17';

* 1. Write a query that finds the emails of customers from VA that haven’t ordered in 3 months.

SELECT email FROM customer WHERE customerid IN(SELECT customerid FROM invoice WHERE invoicedate > '01-FEB-11')

* 1. Write a stored procedure that deletes a Customer as well as any of their Order data.

CREATE OR REPLACE PROCEDURE delAll(cust\_id IN NUMBER)

IS

BEGIN

DELETE FROM demo\_orders WHERE customer\_id = cust\_id;

DELETE FROM demo\_customers WHERE customer\_id = cust\_id;

END delAll;

* 1. Write a stored procedure that inserts a new customer along with their very first order.

CREATE OR REPLACE PROCEDURE newCust(fn IN VARCHAR2, ln IN VARCHAR2, cust\_state IN VARCHAR2, credit IN NUMBER, email IN VARCHAR2)

IS

BEGIN

INSERT INTO demo\_customers(cust\_first\_name, cust\_last\_name, cust\_state, credit\_limit, cust\_email)

VALUES (fn, ln, cust\_state, credit, email);

INSERT INTO demo\_orders(customer\_id, order\_total, order\_timestamp)

VALUES (dc\_seq.currval, 0, current\_date);

END newCust;

* 1. Write a trigger that adds $1,000 to a customer credit limit her total sales increments by $10,000.

CREATE OR REPLACE FUNCTION getTotals(cust\_id IN NUMBER)

RETURN NUMERIC AS

the\_total NUMBER(10,0);

BEGIN

SELECT SUM(order\_total)

INTO the\_total

FROM demo\_orders

WHERE customer\_id = cust\_id;

RETURN the\_total;

END getTotals;

/

CREATE OR REPLACE TRIGGER credit\_trigger

BEFORE INSERT ON demo\_orders --on what occasion do you want to trigger an event?

FOR EACH ROW

BEGIN

IF getTotals(:NEW.customer\_id) >= 10000

THEN

UPDATE demo\_customers SET credit\_limit = credit\_limit + 500 WHERE customer\_id = :NEW.customer\_id;

END IF;

END credit\_trigger;

/

* 1. Write an index on Customer phone number.

CREATE INDEX phone\_index ON customer(phone)

TABLESPACE someSpace

STORAGE (INITIAL 20k, NEXT 20k, PCTINCREASE 1);

* 1. Write a view for each join on these tables: inner, left, right, full.

CREATE VIEW inner\_view AS

SELECT \* FROM demo\_orders

INNER JOIN demo\_customers

ON demo\_customers.customer\_id = demo\_orders.customer\_id;

CREATE VIEW inner\_view AS

SELECT \* FROM demo\_orders

LEFT JOIN demo\_customers

ON demo\_customers.customer\_id = demo\_orders.customer\_id;

CREATE VIEW inner\_view AS

SELECT \* FROM demo\_orders

RIGHT JOIN demo\_customers

ON demo\_customers.customer\_id = demo\_orders.customer\_id;

CREATE VIEW inner\_view AS

SELECT \* FROM demo\_orders

FULL OUTER JOIN demo\_customers

ON demo\_customers.customer\_id = demo\_orders.customer\_id;

1. Given the following table:

|  |  |  |
| --- | --- | --- |
| PERSON | | |
| PERSON\_ID | PERSON\_NAME | PERSON\_PARENT |
| 1 | Joseph | Null |
| 2 | Peter | 1 |
| 3 | Matt | 2 |
| 4 | John | 2 |

(FOREIGN KEY **PERSON.PERSON\_PARENT** REFERENCES **PERSON.PERSON\_ID**)

* 1. Write a query that finds every Person record whose name begins with a ‘J’

SELECT \* FROM person WHERE person\_name LIKE ‘J%’;

* 1. Write a query that displays each person’s name followed by their parent’s name.

SELECT e1.person\_name, e2.person\_name

FROM person e1

LEFT OUTER JOIN person e2

ON e1.person\_id <> e2 person\_parent;

* 1. Write a query that displays each person’s name followed by their number of children. For example:

1 Joseph 1

2 Peter 3

3 Matt 0

4 John 0

SELECT person\_name, SUM(person\_parent) FROM person GROUP BY person\_id;