**Database Design – 2023S**   
**Student ID:** 901142 **Student Name:** Roshan Shrestha **Practical Activity#8 – Sub Queries / Multiple Table Queries using Where.**

---------------------------------------------------------------------------------------------------------------------

The list of all the created tables after modifying and executing the SQL scripts provided are listed below as following:

**A screenshot of a computer

Description automatically generatedTable: REP\_142**

A screenshot of a chat

Description automatically generated**A screenshot of a computer

Description automatically generatedTable:** **CUSTOMER\_142**  
**Table: ORDERS\_142**

A screenshot of a computer

Description automatically generated**A screenshot of a computer

Description automatically generatedTable: PART\_142  
  
Table: ORDER\_LINE\_142**

**Practical Activity#1**

**1. For each order, list the order number and order date along with the number and name of the customer that placed the order.**To retrieve the above mentioned data, we can use the script below:  
**SELECT  
 O.ORDER\_NUM AS order\_id\_142,  
 O.ORDER\_DATE,  
 C.CUSTOMER\_NUM AS customer\_id\_142,  
 C.CUSTOMER\_NAME AS customer\_name  
FROM  
 ORDERS\_142 O  
JOIN  
 CUSTOMER\_142 C ON O.CUSTOMER\_NUM = C.CUSTOMER\_NUM;**

A screenshot of a computer

Description automatically generatedThis SQL query retrieves data from two tables, **ORDERS\_142** and **CUSTOMER\_142**, using a **JOIN** operation. It selects specific columns from each table and aliases them as **order\_id\_142**, **ORDER\_DATE**, **customer\_id\_142**, and **customer\_name**. The **JOIN** is performed based on the matching **CUSTOMER\_NUM** in both tables. The output of the query is as below:

**2. For each order placed on October 23, 2010, list the order number along with the number and name of the customer that placed the order.**To get the list of order that was placed on October 23,2010, we can we the SQL script below:

**SELECT**  
 **O.ORDER\_NUM AS order\_id\_142,**  
 **C.CUSTOMER\_NUM AS customer\_id\_142,**  
 **C.CUSTOMER\_NAME AS customer\_name**  
**FROM  
 ORDERS\_142 O**  
**JOIN**  
 **CUSTOMER\_142 C ON O.CUSTOMER\_NUM = C.CUSTOMER\_NUM**  
**WHERE**  
 **O.ORDER\_DATE = TO\_DATE('2010-10-23', 'YYYY-MM-DD');**

A screenshot of a computer

Description automatically generatedThis SQL query performs a **JOIN** operation on the tables **ORDERS\_142** and **CUSTOMER\_142** to get specified columns (**order\_id\_142**, **customer\_id\_142**, and **customer\_name**). Based on the matching **CUSTOMER\_NUM** in both tables, the **JOIN** is carried out. The **WHERE** clause in the query limits the records that are displayed to those where the **ORDER\_DATE** equals **“2010-10-23”**. The result of the query is below:

**3. For each order, list the order number, order date, part number, number of units ordered, and quoted price for each order line that makes up the order. (Hint :order , orderline).**To get the above mentioned set of data we can execute the query below:  
**SELECT  
 O.ORDER\_NUM AS order\_id\_142,  
 O.ORDER\_DATE,  
 OL.PART\_NUM,  
 OL.NUM\_ORDERED,  
 OL.QUOTED\_PRICE  
FROM  
 ORDERS\_142 O  
JOIN  
 ORDER\_LINE\_142 OL ON O.ORDER\_NUM = OL.ORDER\_NUM  
JOIN  
 PART\_142 P ON OL.PART\_NUM = P.PART\_NUM;**

A screenshot of a computer

Description automatically generated  
Using two **JOIN** procedures, this SQL query pulls information from the three tables **ORDERS\_142**, **ORDER\_LINE\_142**, and **PART\_142**. Order\_Num (also known as **order\_id\_142**), Order Date, Part Number, Number Ordered, and Quoted Price are the columns that are chosen. The matching **ORDER\_NUM** between **ORDERS\_142** and **ORDER\_LINE\_142**, as well as the matching **PART\_NUM** between **ORDER\_LINE\_142** and **PART\_142**, are used to perform the **JOINS**. The output is below:

**Practical Activity#2**

**4. Use the IN operator to find the number and name of each customer that placed an order on October 23, 2010.**We can use the query below to find the number and name of each customer that places an order on October 23, 2010, using **IN** operator:  
**SELECT DISTINCT  
 C.CUSTOMER\_NUM AS customer\_id\_142,  
 C.CUSTOMER\_NAME AS customer\_name  
FROM  
 CUSTOMER\_142 C  
WHERE  
 C.CUSTOMER\_NUM IN (  
 SELECT O.CUSTOMER\_NUM  
 FROM ORDERS\_142 O  
 WHERE O.ORDER\_DATE = TO\_DATE('2010-10-23', 'YYYY-MM-DD') );**

A screenshot of a computer

Description automatically generatedThe **CUSTOMER\_142** table is queried using SQL to retrieve specific customer data. Columns **CUSTOMER\_NUM** (also known as **customer\_id\_142**) and **CUSTOMER\_NAME** is chosen. The **WHERE** clause in the query limits the list of customers in the results to those whose **CUSTOMER\_NUM** appears in the list of **CUSTOMER\_NUM** values from the **ORDERS\_142** table where the **ORDER\_DATE** is **2010-10-23**. Simply said, it retrieves specific customer information for consumers who made an order on the designated date. The output from executing the query is below:

**5. Repeat Exercise 4, but this time use the EXISTS operator in your answer.**Here we are going to find the number and name of each customer that places an order on October 23, 2010, but this time using the **EXISTS** operator, the SQL query can be found below:  
**SELECT DISTINCT  
 C.CUSTOMER\_NUM AS customer\_id\_142,  
 C.CUSTOMER\_NAME AS customer\_name  
FROM  
 CUSTOMER\_142 C  
WHERE  
 EXISTS (  
 SELECT 1  
 FROM ORDERS\_142 O  
 WHERE O.CUSTOMER\_NUM = C.CUSTOMER\_NUM  
 AND O.ORDER\_DATE = TO\_DATE('2010-10-23', 'YYYY-MM-DD'));**

A screenshot of a computer

Description automatically generatedThe **CUSTOMER\_142** table is queried using SQL to retrieve specific customer data. Columns **CUSTOMER\_NUM** (also known as **customer\_id\_142**) and **CUSTOMER\_NAME** are chosen. The query has an **EXISTS** subquery and a **WHERE** clause. The **EXISTS** subquery determines whether the **ORDERS\_142** table contains at least one record with the **CUSTOMER\_NUMBER** matching the one in the primary query and the **ORDER\_DATE** being **2010-10-23** Simply said, it retrieves specific customer information for consumers who made an order on the designated date. The output is as below:

**6. Find the number and name of each customer that did not place an order on October 23, 2010.**In order to retrieve the data including name and number of each customer that did not place an order on October 23, 2010, we can use the below SQL query:  
**SELECT DISTINCT  
 C.CUSTOMER\_NUM AS customer\_id\_142,  
 C.CUSTOMER\_NAME AS customer\_name  
FROM  
 CUSTOMER\_142 C  
WHERE  
 NOT EXISTS (  
 SELECT 1  
 FROM ORDERS\_142 O  
 WHERE O.CUSTOMER\_NUM = C.CUSTOMER\_NUM  
 AND O.ORDER\_DATE = TO\_DATE('2010-10-23', 'YYYY-MM-DD'));**

A screenshot of a computer

Description automatically generatedThe **CUSTOMER\_142** table is queried for unique customer information in this SQL query. Columns **CUSTOMER\_NUM** (aliased as **customer\_id\_142**) and **CUSTOMER\_NAME** is chosen. The query has a **WHERE** clause as well as a **NOT EXISTS** subquery. The **NOT EXISTS** subquery checks to see if there are any records in the **ORDERS\_142** table with the same **CUSTOMER\_NUM** as the main query and the **ORDER\_DATE** of **2010-10-23**. In a nutshell, it retrieves unique customer information for consumers who did not place a purchase on the provided date. The output can be visualized below:

**Practical Activity#3   
7. For each order, list the order number, order date, part number, part description, and item class for each part that makes up the order.**To get the desired data we can use the query below:  
**SELECT**

**O.ORDER\_NUM AS order\_id\_142,**

**O.ORDER\_DATE,**

**OL.PART\_NUM,**

**P.DESCRIPTION AS part\_description,**

**P.CLASS AS item\_class**

**FROM**

**ORDERS\_142 O**

**JOIN**

**ORDER\_LINE\_142 OL ON O.ORDER\_NUM = OL.ORDER\_NUM**

**JOIN  
 PART\_142 P ON OL.PART\_NUM = P.PART\_NUM;**

A screenshot of a computer

Description automatically generatedUsing **JOIN** procedures, this SQL query obtains specified columns from three tables: **ORDERS\_142**, **ORDER\_LINE\_142**, and **PART\_142**. It pulls **ORDER\_NUM** (aliased as **order\_id\_142**), **ORDER\_DATE**, and **PART\_NUM** from the **PART\_142** database, as well as **DESCRIPTION** (aliased as **part\_description**) and **CLASS** (aliased as **item\_class**). Matching **ORDER\_NUM** between **ORDERS\_142** and **ORDER\_LINE\_142**, as well as matching **PART\_NUM** between **ORDER\_LINE\_142** and **PART\_142**, is used to perform the **JOINS**. For each order line, it integrates order information with corresponding part details. The output is below:

**8. Repeat Exercise 7, but this time order the rows by item class and then by order number.**

We can achieve the similar result as for exercise 7, and order the rows by item class and then by order number using the script below:

**SELECT**

**O.ORDER\_NUM AS order\_id\_142,**

**O.ORDER\_DATE,**

**OL.PART\_NUM,**

**P.DESCRIPTION AS part\_description,**

**P.CLASS AS item\_class**

**FROM**

**ORDERS\_142 O**

**JOIN**

**ORDER\_LINE\_142 OL ON O.ORDER\_NUM = OL.ORDER\_NUM**

**JOIN**

**PART\_142 P ON OL.PART\_NUM = P.PART\_NUM**

**ORDER BY**

**P.CLASS,**

**O.ORDER\_NUM;**

A screenshot of a computer

Description automatically generatedUsing **JOIN** procedures, this SQL query obtains specified columns from the three tables **ORDERS\_142**, **ORDER\_LINE\_142**, and **PART\_142**. From the **PART\_142** table, it chooses **ORDER\_NUM** (also known as **order\_id\_142**), **ORDER\_DATE**, **PART\_NUM**, and also inserts **DESCRIPTION** (also known as **part\_description**) and **CLASS** (also known as **item\_class**). Both matching **ORDER\_NUM** between **ORDERS\_142** and **ORDER\_LINE\_142** and matching **PART\_NUM** between **ORDER\_LINE\_142** and **PART\_142** are used to conduct the **JOINS**. Following that, the query sorts the outcomes by **order\_id\_142** and **item\_class** in ascending order. In essence, it receives order and part information, combines it, and then groups the results by item class and order number, in that order. The output is below:

**Practical Activity#4**9. Use a subquery to find the rep number, last name, and first name of each sales rep who represents at least one customer with a credit limit of $10,000. List each sales rep only once in the results.   
**SELECT**

**REP.REP\_NUM AS rep\_number\_142,**

**REP.LAST\_NAME,**

**REP.FIRST\_NAME**

**FROM**

**REP\_142 REP**

**WHERE**

**REP.REP\_NUM IN (**

**SELECT DISTINCT**

**C.REP\_NUM**

**FROM**

**CUSTOMER\_142 C**

**WHERE**

**A screenshot of a computer

Description automatically generated C.CREDIT\_LIMIT >= 10000);**From **the REP\_142** table, this SQL query retrieves representative data. It chooses the columns **LAST\_NAME**, **FIRST\_NAME**, and **REP\_NUM** (also known as **rep\_number\_142**). The subquery that verifies if the **REP\_NUM** is present in the list of different **REP\_NUM** values from the **CUSTOMER\_142** database where the **CREDIT\_LIMIT** is more than or equal to 10000 is included in the query's **WHERE** clause. It retrieves representative information for those who have clients with a credit limit of at least $10,000, to put it briefly.

**10. Repeat Exercise 9, but this time do not use a subquery.**

The same exercise 9 without using subquery can be represented by below SQL script:

**SELECT DISTINCT**

**R.REP\_NUM AS rep\_number\_142,**

**R.LAST\_NAME,**

**R.FIRST\_NAME**

**FROM**

**REP\_142 R**

**JOIN**

**CUSTOMER\_142 C ON R.REP\_NUM = C.REP\_NUM**

**WHERE**

**C.CREDIT\_LIMIT >= 10000;**

This SQL query obtains representative data from the **REP\_142** table by specifying different **REP\_NUM** (aliased as **rep\_number\_142**), **LAST\_NAME**, and **FIRST\_NAME** values. Based on the matching **REP\_NUM**, it executes a **JOIN** transaction with the **CUSTOMER\_142** table. A **WHERE** clause is used to restrict the results to reveal only reps who have customers with a credit limit of at least $10,000. In summary, it retrieves representative information for those who have customers with credit limits of $10,000 or greater. The output is below:

A screenshot of a computer

Description automatically generated