SELECT

KUNDEN\_NR,

FAM\_NAME,

VOR\_NAME,

GEB\_DAT,

ROW\_NUMBER() OVER (ORDER BY GEB\_DAT ASC) AS RowNum

FROM D\_KUST\_FA;

SELECT

KUNDEN\_NR,

FAM\_NAME,

VOR\_NAME,

GEB\_DAT,

PLZ,

RANK() OVER (ORDER BY GEB\_DAT DESC, PLZ) AS Customer\_Rank

FROM D\_KUST\_FA;

WITH AgeCTE AS (

SELECT

KUNDEN\_NR,

ORT,

EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM GEB\_DAT) AS Age

FROM D\_KUST\_FA

)

SELECT

ORT,

AVG(Age) AS Avg\_Age

FROM AgeCTE

GROUP BY ORT

ORDER BY Avg\_Age DESC;

SELECT

C1.KUNDEN\_NR AS Customer\_1,

C1.FAM\_NAME AS Family\_Name\_1,

C2.KUNDEN\_NR AS Customer\_2,

C2.FAM\_NAME AS Family\_Name\_2,

C1.PLZ

FROM D\_KUST\_FA C1

JOIN D\_KUST\_FA C2

ON C1.PLZ = C2.PLZ

AND C1.KUNDEN\_NR != C2.KUNDEN\_NR

ORDER BY C1.PLZ;

SELECT KUNDEN\_NR, FIRMA, KTO\_NR

FROM D\_KUST\_FA

WHERE KTO\_NR = (

SELECT KTO\_NR

FROM D\_KUST\_FA

GROUP BY KTO\_NR

ORDER BY COUNT(\*) DESC

FETCH FIRST 1 ROWS ONLY

);

SELECT

C1.KUNDEN\_NR AS Customer\_With\_Mobile,

C2.KUNDEN\_NR AS Customer\_Without\_Mobile

FROM D\_KUST\_FA C1

FULL OUTER JOIN D\_KUST\_FA C2

ON C1.TELEFON\_MOBIL = C2.TELEFON\_MOBIL

WHERE C1.TELEFON\_MOBIL IS NOT NULL

OR C2.TELEFON\_MOBIL IS NULL;

SELECT KUNDEN\_NR, FAM\_NAME, VOR\_NAME

FROM D\_KUST\_FA C1

WHERE EXISTS (

SELECT 1

FROM D\_KUST\_FA C2

WHERE C1.KUNDEN\_NR = C2.KUNDEN\_NR

AND C2.ORT = 'HAMBURG'

AND C2.EMAIL IS NOT NULL

);

SELECT

KUNDEN\_NR,

FAM\_NAME,

VOR\_NAME,

GEB\_DAT,

CASE

WHEN EXTRACT(YEAR FROM SYSDATE) - EXTRACT(YEAR FROM GEB\_DAT) < 60 THEN 'Young'

ELSE 'Senior'

END AS Customer\_Status

FROM D\_KUST\_FA;

SELECT KUNDEN\_NR, FAM\_NAME, VOR\_NAME, ORT

FROM D\_KUST\_FA

WHERE ORT IN (

SELECT ORT

FROM D\_KUST\_FA

WHERE GEB\_DAT < TO\_DATE('01-JAN-1950', 'DD-MON-YYYY')

);

Let me explain the concept of **C1** and **C2** in very simple terms.

**What Are C1 and C2?**

In SQL, **C1** and **C2** are just **aliases** (nicknames) given to two **instances** (or copies) of the same table.

Let me break it down with an analogy:

* Imagine you have a **list of customers** in front of you. Now, imagine that you want to compare **two different customers** from the same list.
* To do that, you **copy** the list twice. One copy of the list is called **C1** and the other copy is called **C2**.
* Now, when you look at **C1**, you're looking at the list of customers from one perspective, and when you look at **C2**, you're looking at the same list but from another perspective.
* So, **C1** and **C2** are just **two versions** of the same list, and you can compare data from these two "versions" (or "copies") of the list.

**Why Do We Use C1 and C2?**

We use **C1** and **C2** when we want to **compare rows** within the **same table**. This is called a **self-join**. It’s useful when you want to compare or match data that is in the same table, but from different rows.

For example, in your query:

* **C1** represents the **first instance** of the table D\_KUST\_FA (the first copy of the customer list).
* **C2** represents the **second instance** of the same table D\_KUST\_FA (the second copy of the customer list).

**The Join Between C1 and C2**

The query uses a **JOIN** between **C1** and **C2**. This means the query is asking the database to **combine** the data from **C1** and **C2** based on certain conditions (like the postal code being the same and the customer numbers being different).

**Example to Understand C1 and C2 Better**

Let’s say the table D\_KUST\_FA looks like this:

| **KUNDEN\_NR** | **FAM\_NAME** | **PLZ** |
| --- | --- | --- |
| 1001 | Schmidt | 70123 |
| 1002 | Müller | 70123 |
| 1003 | Weber | 70124 |
| 1004 | Meyer | 70123 |

Now, if you want to find **pairs of customers** that live in the same postal code, you would use **C1** and **C2** like this:

* **C1**: Imagine **C1** as the first copy of the table. So, when we say C1.PLZ, we are talking about the **postal code** from the first copy of the table.
* **C2**: **C2** is the second copy of the same table. When we say C2.PLZ, we are talking about the **postal code** from the second copy of the table.

By doing this, the query can **compare** the **postal code** of one customer (from **C1**) with the **postal code** of another customer (from **C2**) to see if they match.

**What Happens in the Query?**

* We are looking for **pairs of customers** that live in the **same postal code**.
* We **exclude** pairs where the **customer number** (KUNDEN\_NR) is the same (i.e., we don’t want to compare a customer with themselves).

**Simplified Explanation of the Query**

SELECT

C1.KUNDEN\_NR AS Customer\_1, -- Customer number from the first list (C1)

C1.FAM\_NAME AS Family\_Name\_1, -- Family name from the first list (C1)

C2.KUNDEN\_NR AS Customer\_2, -- Customer number from the second list (C2)

C2.FAM\_NAME AS Family\_Name\_2, -- Family name from the second list (C2)

C1.PLZ -- Postal code from the first list (C1)

FROM D\_KUST\_FA C1 -- First copy of the customers table (C1)

JOIN D\_KUST\_FA C2 -- Second copy of the customers table (C2)

ON C1.PLZ = C2.PLZ -- Find customers with the same postal code

AND C1.KUNDEN\_NR != C2.KUNDEN\_NR -- Make sure it's not the same customer (C1 != C2)

ORDER BY C1.PLZ; -- Order by postal code

**In Very Simple Terms:**

* **C1** and **C2** are just **two copies** of the same list (the D\_KUST\_FA table).
* We want to **compare** data from both lists to find pairs of customers who live in the **same postal code** but are **not the same customer**.
* **C1** gives us data from one "copy" of the table, and **C2** gives us data from the other "copy" of the table.
* We're looking for rows where **C1** and **C2** have the same postal code but different customer numbers.

**Why Use This Technique?**

This technique of **self-join** with **C1** and **C2** is useful when you want to find pairs or compare data within the same table (for example, comparing customers who live in the same location).

**Final Summary:**

* **C1** and **C2** are just **two copies** of the same table. One is called **C1**, and the other is called **C2**.
* We use them to **compare** data in the same table, like finding **pairs of customers** who live in the same postal code.
* This technique is called a **self-join**, and it helps us compare rows from the same table.

I hope this clears things up! Let me know if you have more questions.