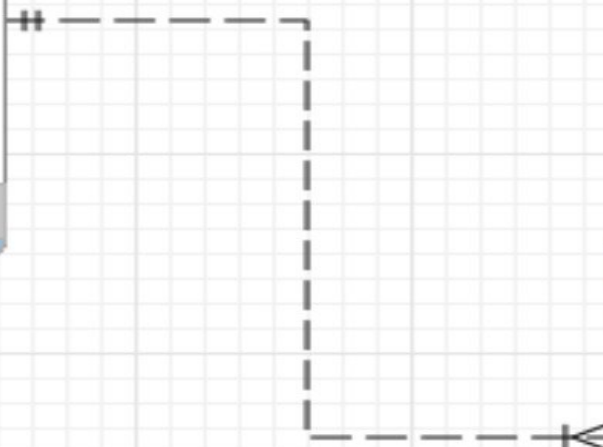
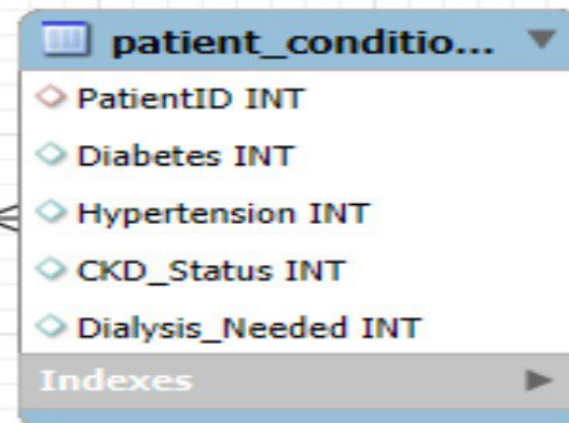
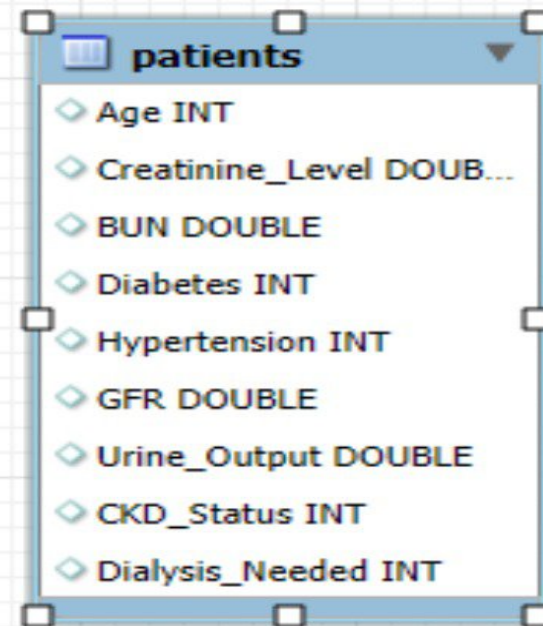
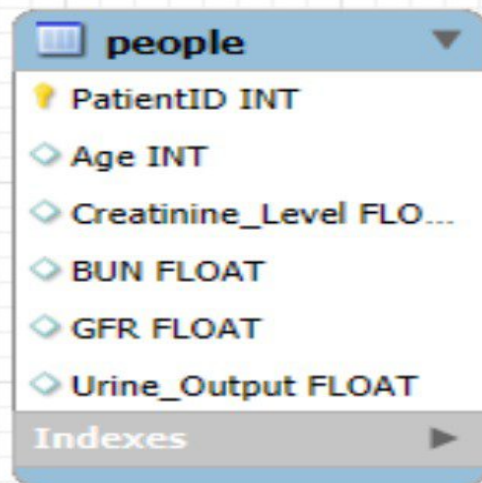




Filtering Health Through Data

Turning patients data into insights,
and insights into healthier lives.



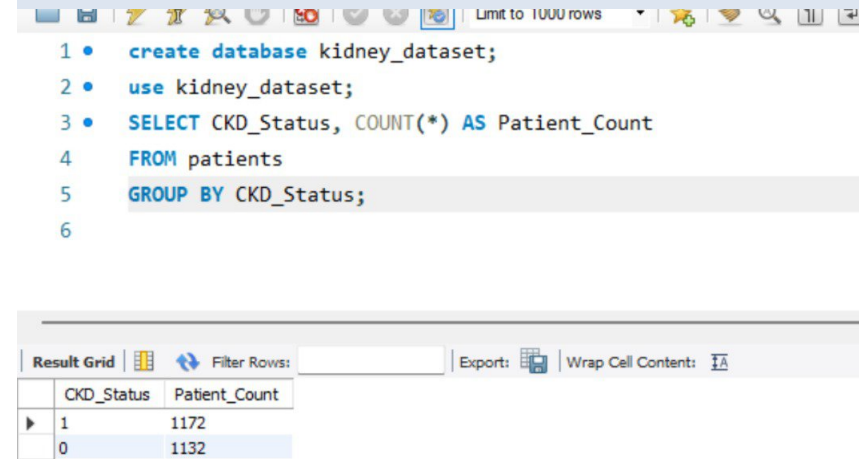
CKD

Chronic Kidney Disease

This a long-term condition where the kidneys gradually lose their ability to

filter waste and excess fluids from the blood.

It usually develops due to diabetes, high blood pressure, or other kidney disorders, and can progress to kidney failure if untreated.



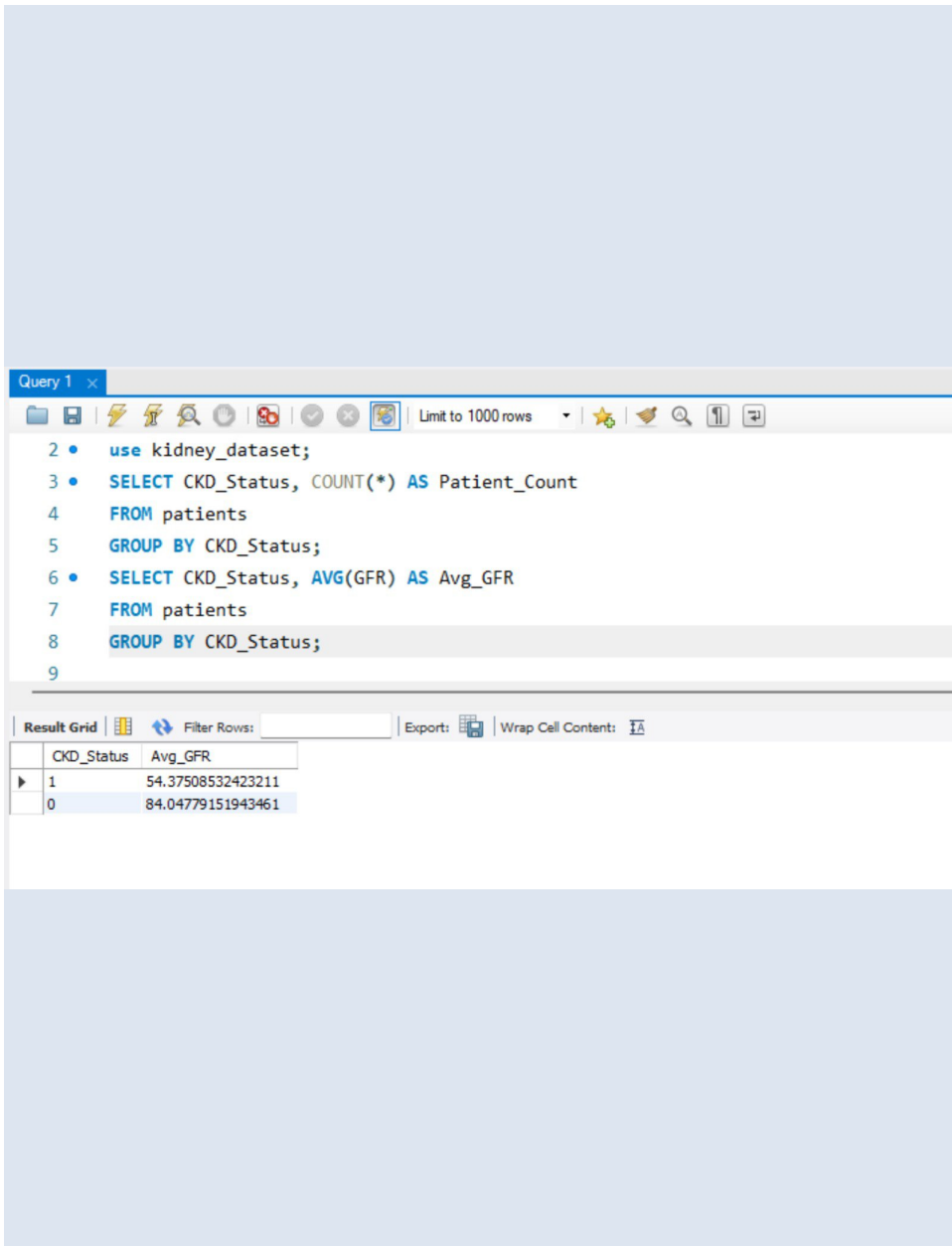
The screenshot shows a SQL query editor with the following code:

```
1 • create database kidney_dataset;  
2 • use kidney_dataset;  
3 • SELECT CKD_Status, COUNT(*) AS Patient_Count  
4   FROM patients  
5   GROUP BY CKD_Status;  
6
```

Below the query editor is a results window titled "Result Grid". It displays the following data:

	CKD_Status	Patient_Count
▶	1	1172
	0	1132

The results window also includes a "Filter Rows:" field, an "Export:" button, and a "Wrap Cell Content:" checkbox.



The screenshot displays the Arc Interior Studio interface. At the top, there is a toolbar with various icons for file operations, search, and execution. Below the toolbar, a SQL query is entered in a text area. The query is as follows:

```
2 • use kidney_dataset;  
3 • SELECT CKD_Status, COUNT(*) AS Patient_Count  
4 FROM patients  
5 GROUP BY CKD_Status;  
6 • SELECT CKD_Status, AVG(GFR) AS Avg_GFR  
7 FROM patients  
8 GROUP BY CKD_Status;  
9
```

Below the query editor, there is a 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The result grid shows two columns: 'CKD_Status' and 'Avg_GFR'. The data is as follows:

CKD_Status	Avg_GFR
1	54.37508532423211
0	84.04779151943461

GFR

Glomerular Filtration Rate

This a test that measures how well your kidneys filter blood by checking the rate at which fluid passes through the glomeruli (tiny filters in the kidneys).

It is used to assess kidney function and classify stages of Chronic Kidney Disease (CKD).

Diabetes & Hypertension

Used Count and Group By Function

Query 1

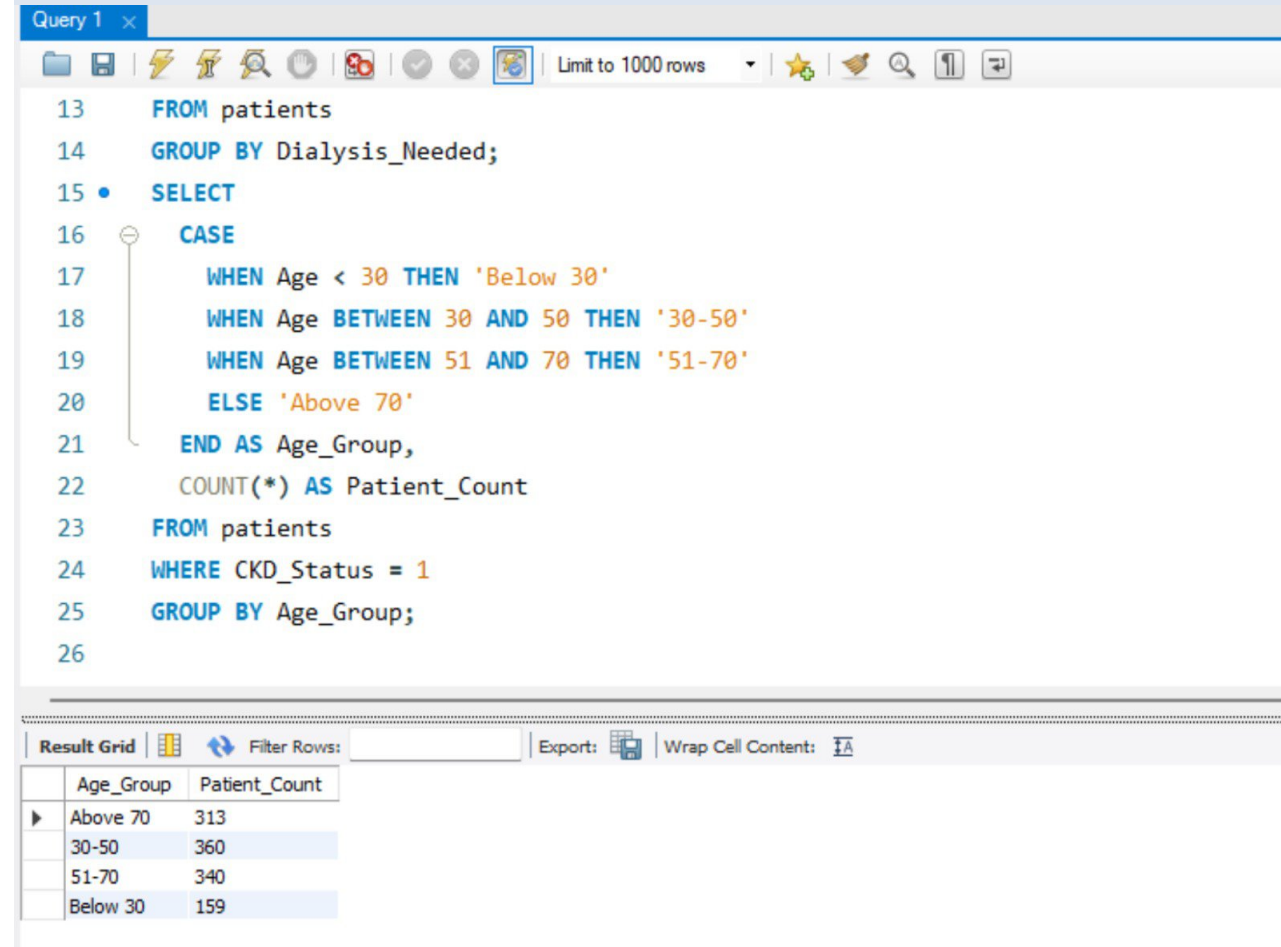
```
5 GROUP BY CKD_Status;
6 • SELECT CKD_Status, AVG(GFR) AS Avg_GFR
7 FROM patients
8 GROUP BY CKD_Status;
9 • SELECT Diabetes, Hypertension, COUNT(*) AS Patient_Count
10 FROM patients
11 GROUP BY Diabetes, Hypertension;
12
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	Diabetes	Hypertension	Patient_Count
▶	0	1	677
	0	0	690
	1	1	471
	1	0	466

Patients of different Age Group.

Used Where Clause &
Case Statement



Query 1

Limit to 1000 rows

```
13 FROM patients
14 GROUP BY Dialysis_Needed;
15 • SELECT
16 CASE
17     WHEN Age < 30 THEN 'Below 30'
18     WHEN Age BETWEEN 30 AND 50 THEN '30-50'
19     WHEN Age BETWEEN 51 AND 70 THEN '51-70'
20     ELSE 'Above 70'
21 END AS Age_Group,
22 COUNT(*) AS Patient_Count
23 FROM patients
24 WHERE CKD_Status = 1
25 GROUP BY Age_Group;
26
```

Result Grid

	Age_Group	Patient_Count
▶	Above 70	313
	30-50	360
	51-70	340
	Below 30	159

Query 1

Limit to 1000 rows

8

GROUP BY CKD_Status;

9

SELECT Diabetes, Hypertension, COUNT(*) AS Patient_Count

10

FROM patients

11

GROUP BY Diabetes, Hypertension;

12

SELECT Dialysis_Needed, AVG(Creatinine_Level) AS Avg_Creatinine

13

FROM patients

14

GROUP BY Dialysis_Needed;

15

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

Dialysis_Needed	Avg_Creatinine
0	1.3040299164100366
1	1.423548387096774

Creatinine

Creatinine is a waste product formed from the normal breakdown of muscle. The kidneys normally filter it out of the blood through urine, so high blood creatinine levels usually indicate reduced kidney function.

Top 5 High Creatinine Patients

Used Order By & Limit Function

Query 1

```
29 • SELECT
30     SUM(CASE WHEN Dialysis_Needed = 1 THEN 1 ELSE 0 END) * 100.0 / COUNT(*) AS Percent_Dialysis
31 FROM patients
32 WHERE CKD_Status = 1;
33 • SELECT Age, Creatinine_Level, GFR, CKD_Status, Dialysis_Needed
34 FROM patients
35 ORDER BY Creatinine_Level DESC
36 LIMIT 5;
```

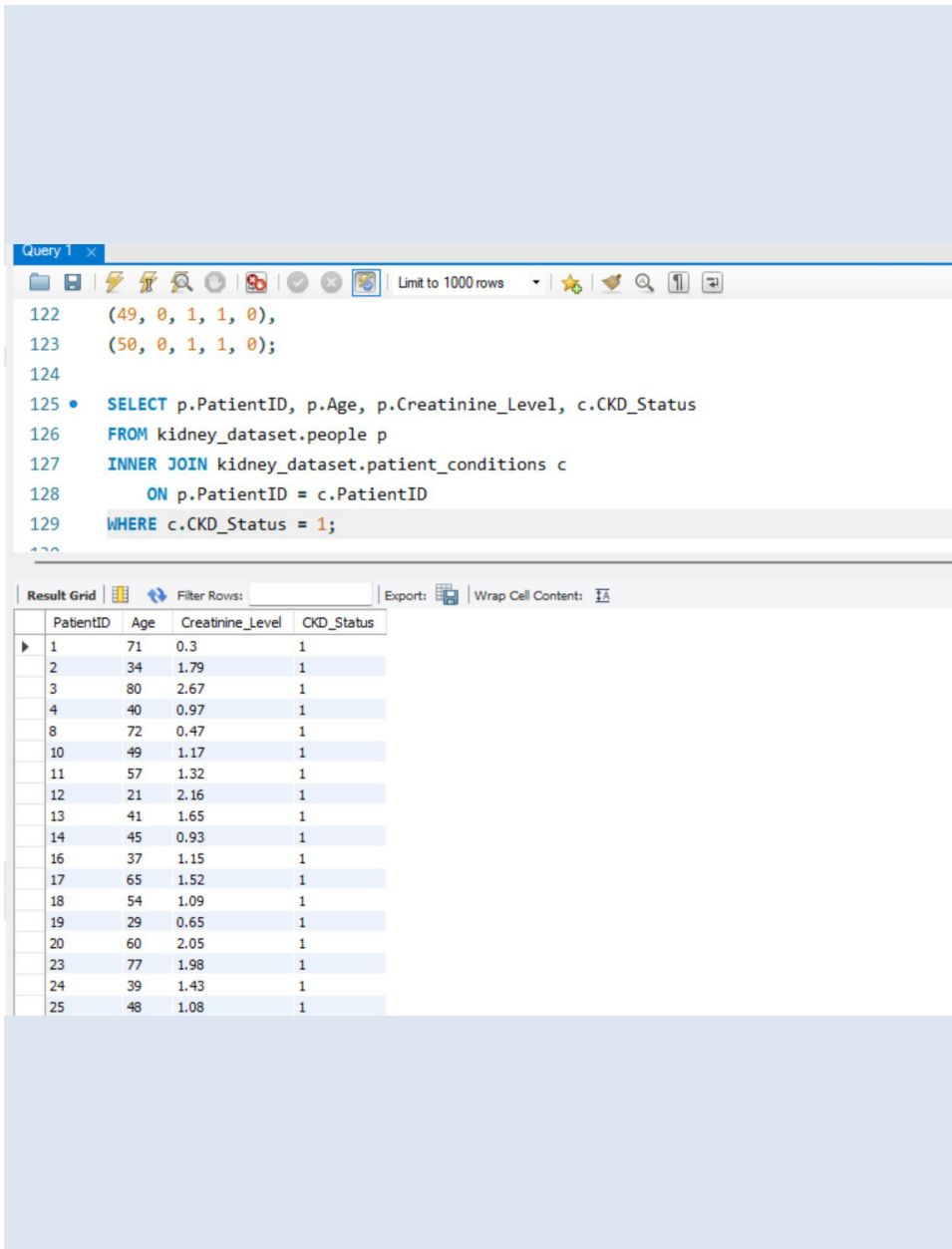
Result Grid

	Age	Creatinine_Level	GFR	CKD_Status	Dialysis_Needed
▶	25	4.13	106.4	1	0
	81	4.13	114.5	1	0
	47	4.01	86.5	1	0
	73	3.98	70.1	1	0
	36	3.83	104.1	1	0

Creation of Two
Tables Using

Primary Key & Foreign Key

```
Query 1 x
Limit to 1000 rows
1 • create database kidney_dataset;
2 • use kidney_dataset;
3 • CREATE TABLE people (
4     PatientID INT PRIMARY KEY,
5     Age INT,
6     Creatinine_Level FLOAT,
7     BUN FLOAT,
8     GFR FLOAT,
9     Urine_Output FLOAT
10 );
11
12 -- Patient conditions table
13 • CREATE TABLE patient_conditions (
14     PatientID INT,
15     Diabetes INT,
16     Hypertension INT,
17     CKD_Status INT,
18     Dialysis_Needed INT,
19     FOREIGN KEY (PatientID) REFERENCES people(PatientID)
20 );
```



The screenshot shows a SQL query editor window with a query that performs an inner join between two tables in a database. The query is as follows:

```
122 (49, 0, 1, 1, 0),  
123 (50, 0, 1, 1, 0);  
124  
125 • SELECT p.PatientID, p.Age, p.Creatinine_Level, c.CKD_Status  
126 FROM kidney_dataset.people p  
127 INNER JOIN kidney_dataset.patient_conditions c  
128 ON p.PatientID = c.PatientID  
129 WHERE c.CKD_Status = 1;
```

Below the query editor, the 'Result Grid' is displayed, showing the results of the query. The grid has four columns: PatientID, Age, Creatinine_Level, and CKD_Status. The results are as follows:

	PatientID	Age	Creatinine_Level	CKD_Status
1	71	0.3	1	1
2	34	1.79	1	1
3	80	2.67	1	1
4	40	0.97	1	1
8	72	0.47	1	1
10	49	1.17	1	1
11	57	1.32	1	1
12	21	2.16	1	1
13	41	1.65	1	1
14	45	0.93	1	1
16	37	1.15	1	1
17	65	1.52	1	1
18	54	1.09	1	1
19	29	0.65	1	1
20	60	2.05	1	1
23	77	1.98	1	1
24	39	1.43	1	1
25	48	1.08	1	1

Inner Join

An **INNER JOIN** in SQL returns only the rows that have matching values in both tables based on a specified condition.

Query 1 x

Limit to 1000 rows

```
127 INNER JOIN kidney_dataset.patient_conditions c
128     ON p.PatientID = c.PatientID
129 WHERE c.CKD_Status = 1;
130
131 • SELECT p.PatientID, p.Age, c.Diabetes
132 FROM kidney_dataset.people p
133 LEFT JOIN kidney_dataset.patient_conditions c
134     ON p.PatientID = c.PatientID;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	PatientID	Age	Diabetes
▶	1	71	0
	2	34	0
	3	80	0
	4	40	0
	5	43	1
	6	22	0
	7	41	1
	8	72	1
	9	21	1
	10	49	0
	11	57	1
	12	21	1
	13	41	1
	14	45	0
	15	27	0
	16	37	1
	17	65	0
	18	54	0

Left Join

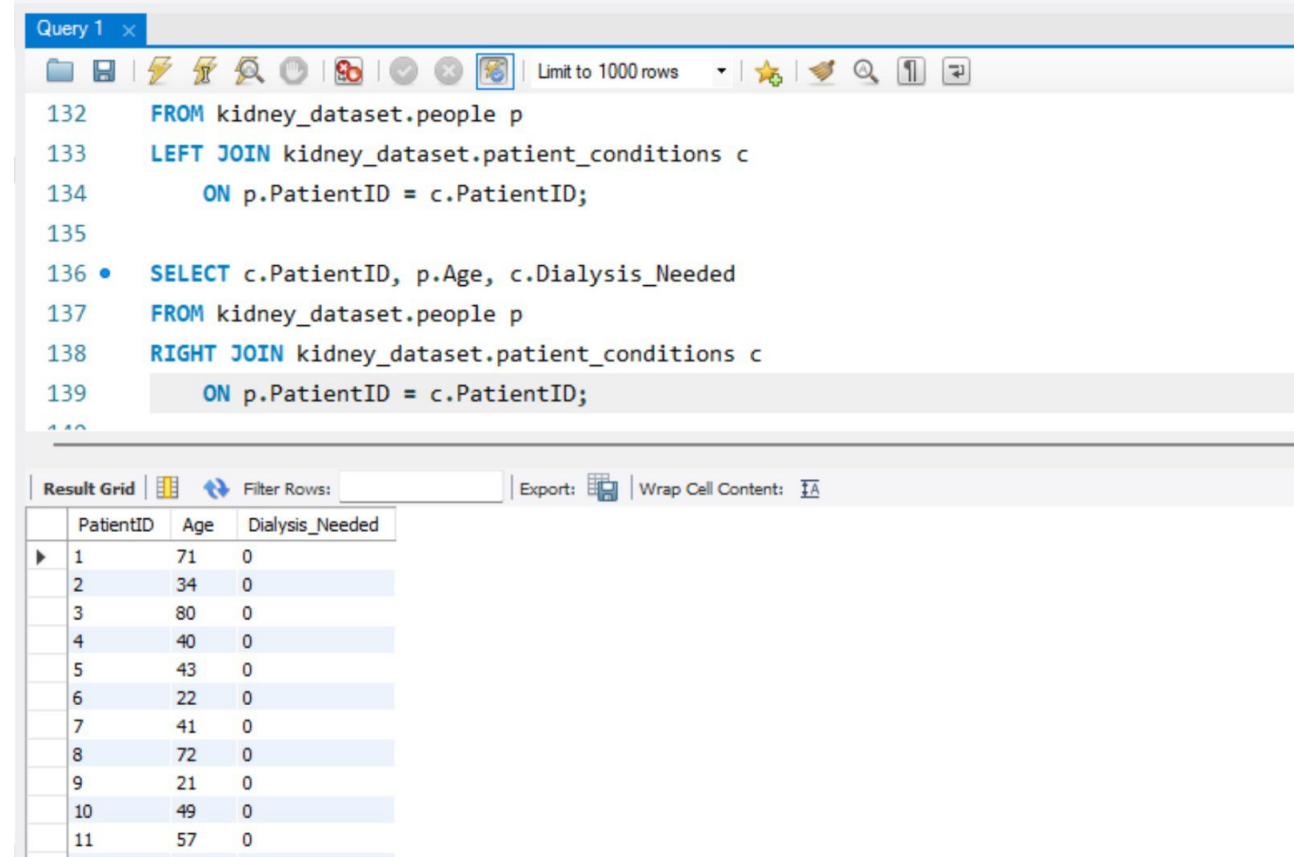
A **LEFT JOIN** in SQL returns **all rows from the left table** and the **matching rows from the right table**.

If there's no match, it still shows the left table row but fills the right table columns with **NULL**.

Right Join

A **RIGHT JOIN** in SQL returns **all rows from the right table** and the **matching rows from the left table**.

If there's no match, it still shows the right table row but fills the left table columns with **NULL**.

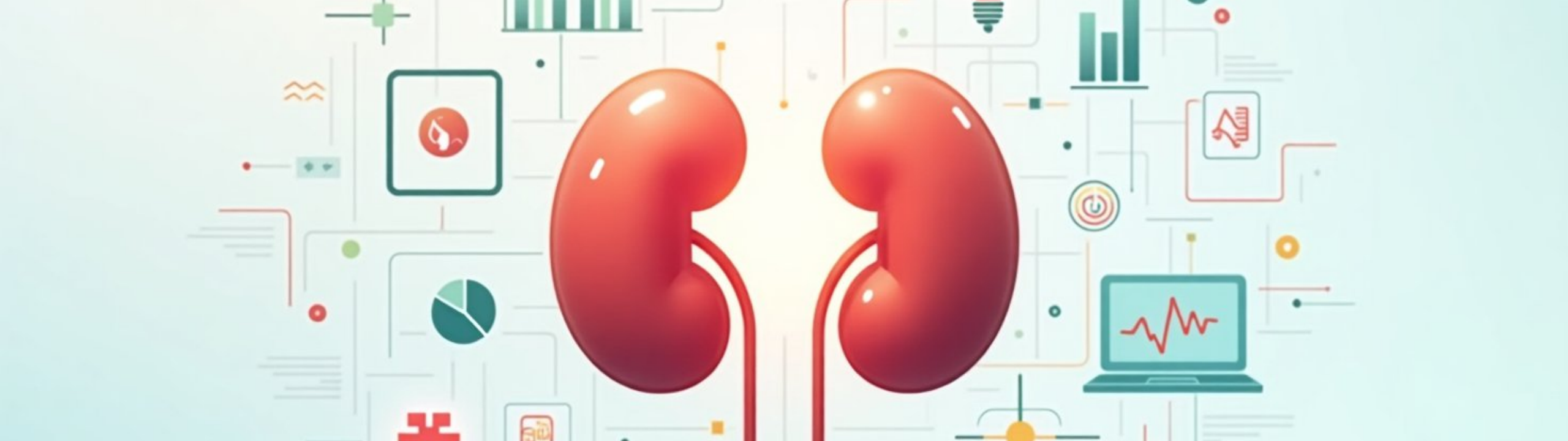


The screenshot displays a SQL query editor window titled "Query 1". The query is as follows:

```
132 FROM kidney_dataset.people p
133 LEFT JOIN kidney_dataset.patient_conditions c
134     ON p.PatientID = c.PatientID;
135
136 • SELECT c.PatientID, p.Age, c.Dialysis_Needed
137 FROM kidney_dataset.people p
138 RIGHT JOIN kidney_dataset.patient_conditions c
139     ON p.PatientID = c.PatientID;
```

Below the query editor is the "Result Grid" showing the output of the query. The grid has three columns: PatientID, Age, and Dialysis_Needed. It displays 11 rows of data, where PatientID 1 through 10 have matching entries in the patient_conditions table, and PatientID 11 does not.

	PatientID	Age	Dialysis_Needed
▶	1	71	0
	2	34	0
	3	80	0
	4	40	0
	5	43	0
	6	22	0
	7	41	0
	8	72	0
	9	21	0
	10	49	0
	11	57	0



Thank you