

Module 1: SQL Server ----> Data Source 2: Incurance Claim_State_Health_Data

2nd NORMALIZATION FORM

1. PatientDetails Table (2nd Normalization Form)

The screenshot shows the DBeaver 22.2.0 interface with the SQL Editor open. The SQL script defines the PatientDetails table and inserts data. The Results window shows the data for the PatientDetails table.

```
--1st Normalization Form: Table is already in 1st NF
*****
--2nd Normalization Form

--Incurance Claim_State_Health_Data Table can be Split into 2 by separating PID, GENDER, AGE, HOSPITAL_NAME, SURGURY_CODE in one table and SURGURY_CODE, SURGERY in another
--It is because SURGERY is dependent on SURGURY_CODE and it breaks the 2NF rule

-- PatientDetails Table

CREATE TABLE PatientDetails
(
    PID INT PRIMARY KEY,
    AGE INT,
    GENDER VARCHAR(50),
    HOSPITAL_NAME VARCHAR(50),
    SURGURY_CODE VARCHAR(50)
)

SELECT * FROM PatientDetails

INSERT INTO PatientDetails VALUES
(1, 56, 'Female', 'Rims Govt. General Hospital, Srikakulam', 'M6.5'),
```

PID	AGE	GENDER	HOSPITAL_NAME	SURGURY_CODE
1	56	Female	Rims Govt. General Hospital, Srikakulam	M6.5
2	37	Male	Rims Govt. General Hospital, Srikakulam	M6.5
3	50	Male	Rims Govt. General Hospital, Srikakulam	M6.5
4	45	Male	Rims Govt. General Hospital, Srikakulam	M6.5
5	54	Male	Rims Govt. General Hospital, Srikakulam	M6.5

2. SurgeryDetails Table (2nd Normalization Form)

The screenshot shows the DBeaver 22.2.0 interface with the SQL Editor open. The SQL script defines the SurgeryDetails table and inserts data. The Results window shows the data for the SurgeryDetails table.

```
-- SurgeryDetails Table

CREATE TABLE SurgeryDetails
(
    SURGURY_CODE VARCHAR(50),
    SURGERY VARCHAR(100)
)

SELECT * FROM SurgeryDetails

INSERT INTO SurgeryDetails VALUES
('M6.5', 'Maintenance Hemodialysis For Crf'),
('M6.5', 'Maintenance Hemodialysis For Crf'),
('M6.5', 'Maintenance Hemodialysis For Crf'),
('M6.5', 'Maintenance Hemodialysis For Crf'),
('M6.5', 'Maintenance Hemodialysis For Crf'),
```

SURGURY_CODE	SURGERY
M6.5	Maintenance Hemodialysis For Crf
M6.5	Maintenance Hemodialysis For Crf
M6.5	Maintenance Hemodialysis For Crf
M6.5	Maintenance Hemodialysis For Crf
M6.5	Maintenance Hemodialysis For Crf
M6.5	Maintenance Hemodialysis For Crf
M6.5	Maintenance Hemodialysis For Crf
S7.1.1.1	Coronary Balloon Angioplasty with stent(00.45)

3. PatientDetailss Table (3rd Normalization Form)

DBeaver 22.2.0 - <nehat> NORMALIZATION

File Edit Navigate Search SQL Editor Database Window Help

Auto nehat dbo@nehat

<nehat> NORMALIZATION_neha <nehat> NORMALIZATION <nehat> Script-7

```
--3rd Normalization Form

--Insurance Claim_State_Health_Data Table can be Split into 2 by separating PID, AGE, GENDER, DISTRICT_NAME, CATEGORY_CODE in one table and CATEGORY_CODE, CATEGORY_NAME in another table
--It is because CATEGORY_CODE is dependent on PID that makes non primary attributes (CATEGORY_NAME, SURGERY_CODE, SURGERY) transitively dependent on PID and it breaks 3rd Normalization Form

CREATE TABLE PatientDetailss
(
    PID INT PRIMARY KEY,
    AGE INT,
    GENDER VARCHAR(50),
    DISTRICT_NAME VARCHAR(50),
    CATEGORY_CODE VARCHAR(50)
)

SELECT * FROM PatientDetailss

INSERT INTO PatientDetailss VALUES
(1, 56, 'Female', 'Srikakulam', 'M6'),
(2, 37, 'Male', 'Srikakulam', 'M6'),
(3, 50, 'Male', 'Srikakulam', 'M6'),
(4, 45, 'Male', 'Srikakulam', 'M6'),
(5, 54, 'Male', 'Srikakulam', 'M6'),
(6, 35, 'Male', 'Srikakulam', 'M6')
```

Results 1 x Output

SELECT * FROM PatientDetailss

PID	AGE	GENDER	DISTRICT_NAME	CATEGORY_CODE
1	56	Female	Srikakulam	M6
2	37	Male	Srikakulam	M6
3	50	Male	Srikakulam	M6
4	45	Male	Srikakulam	M6
5	54	Male	Srikakulam	M6
6	35	Male	Srikakulam	M6

IST en Writable Smart Insert 101:55:4377 Set:0|0

4.SurgeryDetailss Table (3rd Normalization Form)

DBeaver 22.2.0 - <nehat> NORMALIZATION

File Edit Navigate Search SQL Editor Database Window Help

Auto nehat dbo@nehat

<nehat> NORMALIZATION_neha <nehat> NORMALIZATION <nehat> Script-7

```
--SurgeryDetailss Table

CREATE TABLE SurgeryDetailss
(
    CATEGORY_CODE VARCHAR(50),
    CATEGORY_NAME VARCHAR(50),
    SURGERY_CODE VARCHAR(50),
    SURGERY VARCHAR(100)
)

SELECT * FROM SurgeryDetailss

INSERT INTO SurgeryDetailss VALUES
('M6', 'NEPHROLOGY', 'M6.5', 'Maintenance Hemodialysis For Crf'),
('M6', 'NEPHROLOGY', 'M6.5', 'Maintenance Hemodialysis For Crf'),
('M6', 'NEPHROLOGY', 'M6.5', 'Maintenance Hemodialysis For Crf'),
('M6', 'NEPHROLOGY', 'M6.5', 'Maintenance Hemodialysis For Crf'),
('M6', 'NEPHROLOGY', 'M6.5', 'Maintenance Hemodialysis For Crf'),
('M6', 'NEPHROLOGY', 'M6.5', 'Maintenance Hemodialysis For Crf'),
('M6', 'NEPHROLOGY', 'M6.5', 'Maintenance Hemodialysis For Crf'),
('M6', 'NEPHROLOGY', 'M6.5', 'Maintenance Hemodialysis For Crf')
```

Results 1 x Output

SELECT * FROM SurgeryDetailss

CATEGORY_CODE	CATEGORY_NAME	SURGERY_CODE	SURGERY
M6	NEPHROLOGY	M6.5	Maintenance Hemodialysis For Crf
M6	NEPHROLOGY	M6.5	Maintenance Hemodialysis For Crf
M6	NEPHROLOGY	M6.5	Maintenance Hemodialysis For Crf
M6	NEPHROLOGY	M6.5	Maintenance Hemodialysis For Crf
M6	NEPHROLOGY	M6.5	Maintenance Hemodialysis For Crf
M6	NEPHROLOGY	M6.5	Maintenance Hemodialysis For Crf
M6	NEPHROLOGY	M6.5	Maintenance Hemodialysis For Crf
M6	NEPHROLOGY	M6.5	Maintenance Hemodialysis For Crf

IST en Writable Smart Insert 148:24:5917 Set:0|0

JOIN

5. INNER JOIN

The screenshot shows the DBeaver 22.2.0 interface with a SQL query editor and a results grid. The query is an INNER JOIN between PatientDetails and SurgeryDetails tables.

```
-- JOINS
-- INNER JOIN
-- (Returns Only Common data from other table)

SELECT Pat.PID, Pat.AGE, Pat.GENDER, Pat.DISTRICT_NAME, Pat.CATEGORY_CODE, Sur.SURGERY
FROM PatientDetails AS Pat
INNER JOIN SurgeryDetails AS Sur
ON Pat.CATEGORY_CODE = Sur.CATEGORY_CODE

-- LEFT JOIN
-- (Returns all rows from the left table (First table) and matching rows from the right table)

SELECT Pat.PID, Pat.AGE, Pat.GENDER, Pat.DISTRICT_NAME, Pat.CATEGORY_CODE, Sur.SURGERY
FROM PatientDetails AS Pat
LEFT JOIN SurgeryDetails AS Sur
ON Pat.CATEGORY_CODE = Sur.CATEGORY_CODE

-- RIGHT JOIN JOIN
-- (Returns all rows from the right table (Second table) and matching rows from the left table)

SELECT Pat.PID, Pat.AGE, Pat.GENDER, Pat.DISTRICT_NAME, Pat.CATEGORY_CODE, Sur.SURGERY
FROM PatientDetails AS Pat
RIGHT JOIN SurgeryDetails AS Sur
ON Pat.CATEGORY_CODE = Sur.CATEGORY_CODE
```

The results grid shows 4 rows of data:

PID	AGE	GENDER	DISTRICT_NAME	CATEGORY_CODE	SURGERY
1	37	Male	Srikakulam	M6	Maintenance Hemodialysis For Crf
2	50	Male	Srikakulam	M6	Maintenance Hemodialysis For Crf
3	45	Male	Srikakulam	M6	Maintenance Hemodialysis For Crf
4	54	Male	Srikakulam	M6	Maintenance Hemodialysis For Crf

6. LEFT JOIN

The screenshot shows the DBeaver 22.2.0 interface with a SQL query editor and a results grid. The query is a LEFT JOIN between PatientDetails and SurgeryDetails tables.

```
-- JOINS
-- INNER JOIN
-- (Returns Only Common data from other table)

SELECT Pat.PID, Pat.AGE, Pat.GENDER, Pat.DISTRICT_NAME, Pat.CATEGORY_CODE, Sur.SURGERY
FROM PatientDetails AS Pat
INNER JOIN SurgeryDetails AS Sur
ON Pat.CATEGORY_CODE = Sur.CATEGORY_CODE

-- LEFT JOIN
-- (Returns all rows from the left table (First table) and matching rows from the right table)

SELECT Pat.PID, Pat.AGE, Pat.GENDER, Pat.DISTRICT_NAME, Pat.CATEGORY_CODE, Sur.SURGERY
FROM PatientDetails AS Pat
LEFT JOIN SurgeryDetails AS Sur
ON Pat.CATEGORY_CODE = Sur.CATEGORY_CODE

-- RIGHT JOIN JOIN
-- (Returns all rows from the right table (Second table) and matching rows from the left table)

SELECT Pat.PID, Pat.AGE, Pat.GENDER, Pat.DISTRICT_NAME, Pat.CATEGORY_CODE, Sur.SURGERY
FROM PatientDetails AS Pat
RIGHT JOIN SurgeryDetails AS Sur
ON Pat.CATEGORY_CODE = Sur.CATEGORY_CODE
```

The results grid shows 4 rows of data:

PID	AGE	GENDER	DISTRICT_NAME	CATEGORY_CODE	SURGERY
1	56	Female	Srikakulam	M5	Management Of Acute MI With An
2	1	56	Female	Srikakulam	Medical Management of Refractor
3	1	56	Female	Srikakulam	Medical Management of Refractor
4	1	56	Female	Srikakulam	Medical Management of Refractor

7. LEFT JOIN

The screenshot shows the DBeaver 22.2.0 interface with a SQL query editor and a results grid. The query is a LEFT JOIN between PatientDetails and SurgeryDetails tables.

```
-- LEFT JOIN
--(Returns all rows from the left table (First table) and matching rows from the right table)

*SELECT Pat.PID, Pat.AGE, Pat.GENDER, Pat.DISTRICT_NAME, Pat.CATEGORY_CODE, Sur.SURGERY
FROM PatientDetails AS Pat
LEFT JOIN SurgeryDetails AS Sur
ON Pat.CATEGORY_CODE = Sur.CATEGORY_CODE

-- RIGHT JOIN JOIN
--(Returns all rows from the right table (Second table) and matching rows from the left table)

*SELECT Pat.PID, Pat.AGE, Pat.GENDER, Pat.DISTRICT_NAME, Pat.CATEGORY_CODE, Sur.SURGERY
FROM PatientDetails AS Pat
RIGHT JOIN SurgeryDetails AS Sur
ON Pat.CATEGORY_CODE = Sur.CATEGORY_CODE

-- FULL JOIN
--(Returns all rows from joined table regardless of matching data)

*SELECT Pat.PID, Pat.AGE, Pat.GENDER, Pat.DISTRICT_NAME, Pat.CATEGORY_CODE, Sur.SURGERY_CODE, Sur.SURGERY
FROM PatientDetails AS Pat
FULL JOIN SurgeryDetails AS Sur
```

The results grid shows 4 rows of data:

PID	AGE	GENDER	DISTRICT_NAME	CATEGORY_CODE	SURGERY
1	37	Male	Srikakulam	M6	Maintenance Hemodialysis For Crf
2	50	Male	Srikakulam	M6	Maintenance Hemodialysis For Crf
3	45	Male	Srikakulam	M6	Maintenance Hemodialysis For Crf
4	54	Male	Srikakulam	M6	Maintenance Hemodialysis For Crf

8. FULL JOIN

The screenshot shows the DBeaver 22.2.0 interface with a SQL query editor and a results grid. The query is a FULL JOIN between PatientDetails and SurgeryDetails tables.

```
-- FULL JOIN
--(Returns all rows from joined table regardless of matching data)

*SELECT Pat.PID, Pat.AGE, Pat.GENDER, Pat.DISTRICT_NAME, Pat.CATEGORY_CODE, Sur.SURGERY_CODE, Sur.SURGERY
FROM PatientDetails AS Pat
FULL JOIN SurgeryDetails AS Sur
ON Pat.CATEGORY_CODE = Sur.CATEGORY_CODE

-- CROSS JOIN
--(Returns matrix 30*30=900 ROWS)

*SELECT Pat.PID, Pat.AGE, Pat.GENDER, Pat.DISTRICT_NAME, Pat.CATEGORY_CODE, Sur.SURGERY_CODE, Sur.SURGERY
FROM PatientDetails AS Pat, SurgeryDetails AS Sur

*****

--ROW_NUMBER, RANK, DENSE_RANK
--(ROW_NUMBER Displays the row number)
--(RANK will be ranking according to the row number leaving the gap. If there is a tie between 1st and 2nd row, it
--(DENSE_RANK never leaves the gap. If there is a tie between 1st and 2nd row, it will go like 1,1,2)

*SELECT
```

The results grid shows 4 rows of data:

PID	AGE	GENDER	DISTRICT_NAME	CATEGORY_CODE	SURGERY_CODE	SURGERY	
1	56	Female	Srikakulam	M5	M5.1.2	Management Of Acute MI V	
2	1	56	Female	Srikakulam	M5	M5.1.5	Medical Management of Re
3	1	56	Female	Srikakulam	M5	M5.1.5	Medical Management of Re
4	1	56	Female	Srikakulam	M5	M5.1.5	Medical Management of Re

9. RANK

The screenshot shows the DBeaver 22.2.0 interface. The SQL Editor contains the following query:

```
--ROW_NUMBER, RANK, DENSE_RANK
--(ROW_NUMBER Displays the row number)
--(RANK will be ranking according to the row number leaving the gap. If there is a tie between 1st and 2nd row, it
--(DENSE_RANK never leaves the gap. If there is a tie between 1st and 2nd row, it will go like 1,1,2)

*SELECT
  CATEGORY_CODE,
  CATEGORY_NAME,
  SURGERY_CODE,
  SURGERY,
  ROW_NUMBER () OVER (ORDER BY CATEGORY_CODE) AS ROW_NUMBER,
  RANK() OVER (ORDER BY CATEGORY_CODE) AS RANK,
  DENSE_RANK() OVER (ORDER BY CATEGORY_CODE) AS DENSE_RANK
FROM SurgeryDetails
```

The Results 1 tab shows the following data grid:

	CATEGORY_CODE	CATEGORY_NAME	SURGERY_CODE	SURGERY	ROW_NUMBER	RANK	DENSE_RANK
10	M5	CARDIOLOGY	M5.1.2	Management Of Acute MI Wit	10	1	1
11	M5	CARDIOLOGY	M5.1.2	Management Of Acute MI Wit	11	1	1
12	M5	CARDIOLOGY	M5.1.2	Management Of Acute MI Wit	12	1	1
13	M5	CARDIOLOGY	M5.1.2	Management Of Acute MI Wit	13	1	1
14	M6	NEPHROLOGY	M6.5	Maintenance Hemodialysis For	14	14	2
15	M6	NEPHROLOGY	M6.5	Maintenance Hemodialysis For	15	14	2
16	M6	NEPHROLOGY	M6.5	Maintenance Hemodialysis For	16	14	2
17	M6	NEPHROLOGY	M6.5	Maintenance Hemodialysis For	17	14	2

10. VIEW

The screenshot shows the DBeaver 22.2.0 interface. The SQL Editor contains the following query:

```
*****
--VIEW
--(It does not store in memory, just hold the table structure which can be viewed and changed)

*CREATE VIEW V_PatientDetails
AS
  SELECT Pat.PID, Pat.AGE, Pat.GENDER, Pat.DISTRICT_NAME, Pat.CATEGORY_CODE, Sur.SURGERY
  FROM PatientDetails AS Pat
  INNER JOIN SurgeryDetails AS Sur
  ON Pat.CATEGORY_CODE = Sur.CATEGORY_CODE

  SELECT * FROM V_PatientDetails

DROP VIEW V_PatientDetails
```

The Results 1 tab shows the following data grid:

	PID	AGE	GENDER	DISTRICT_NAME	CATEGORY_CODE	SURGERY
1	2	37	Male	Srikakulam	M6	Maintenance Hemodialysis For Crf
2	3	50	Male	Srikakulam	M6	Maintenance Hemodialysis For Crf
3	4	45	Male	Srikakulam	M6	Maintenance Hemodialysis For Crf
4	5	54	Male	Srikakulam	M6	Maintenance Hemodialysis For Crf
5	6	35	Male	Srikakulam	M6	Maintenance Hemodialysis For Crf
6	7	52	Male	Kurnool	M6	Maintenance Hemodialysis For Crf
7	8	73	Male	Vizianagaram	M6	Maintenance Hemodialysis For Crf
8	11	52	Male	Vishakhapatnam	M6	Maintenance Hemodialysis For Crf
9	21	55	Male	West Godavari	M6	Maintenance Hemodialysis For Crf