Create a new Data Table and add columns by defining its data types

(CREATE, ALTER – UPDATE, VARCHAR, SELECT clauses)

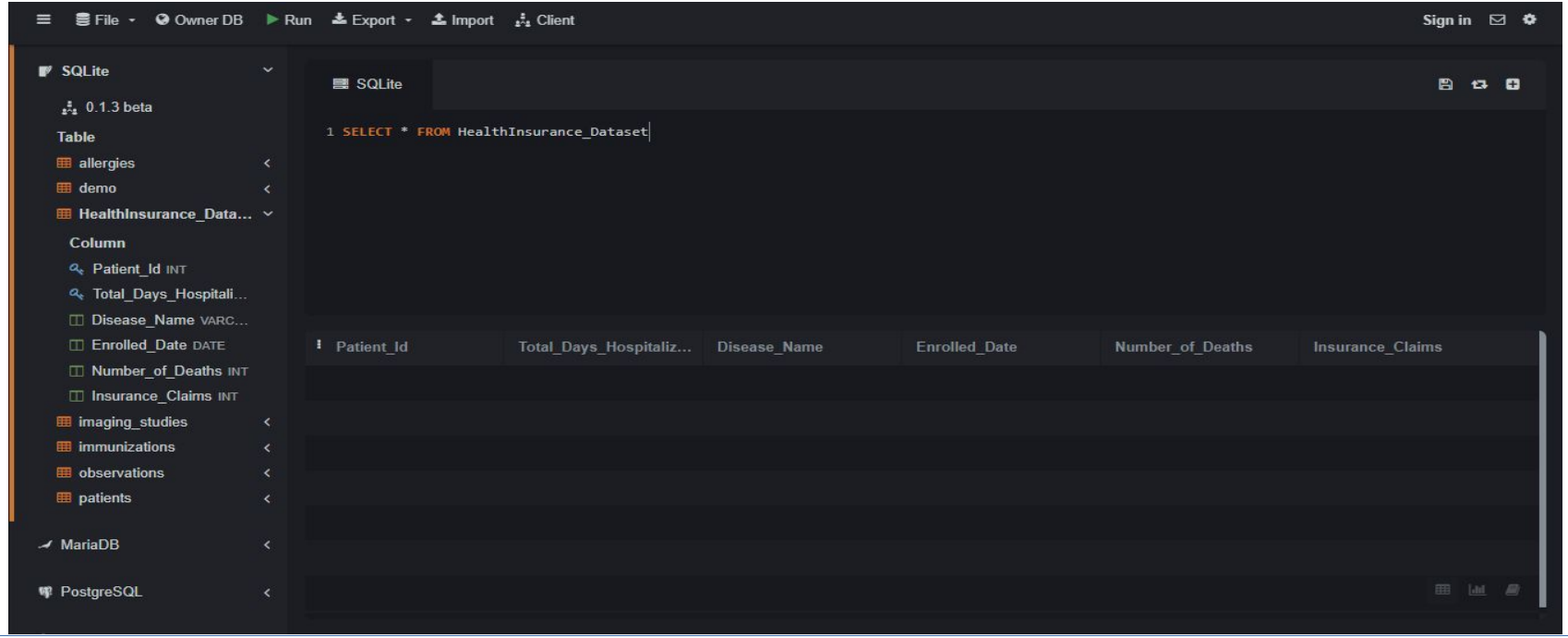
```
CREATE TABLE HealthInsurance_Dataset
(Patient_Id INT, Total_Days_Hospitalized INT, Disease_Name VARCHAR(50) NOT NULL, Enrolled_Date Date);
```

Add new columns to the table created and set the rule for Number of Deaths as 0 if the value is Null

```
ALTER TABLE HealthInsurance_Dataset
ADD COLUMN Number_of_Deaths INT;
ADD COLUMN Insurance_Claims INT;
UPDATE HealthInsurance_Dataset SET Number_of_Deaths = 0 WHERE Number_of_Deaths IS NULL;
```

Now retrieve the column details in the newly created data table

```
SELECT * FROM HealthInsurance_Dataset;
```



Add only 8 rows into all the columns of the data table created

(INSERT INTO Clause)

```
INSERT INTO HealthInsurance_Dataset (Patient_Id, Total_Days_Hospitalized, Disease_Name, Enrolled_Date, Number_of_Deaths, insurance_claims)
VALUES
(1, 30, 'Alzimers Disease', '2024-05-01', 0, '153000'),
(2, 45, 'Atrial Fibrillation', '2024-05-02', 0, '250350'),
(3, 22, 'Septic Shock', '2024-05-03', 0, '1545000'),
(4, 33, 'Brain Haemorrhage', '2024-05-04', 0, '2545321'),
(5, 50, 'Addisons Disease', '2024-05-05', 0, '90000'),
(6, 17, 'Pneumonia', '2024-05-06', 0, '36455'),
(7, 70, 'Cirrhosis', '2024-05-07', 0, '85956'),
(8, 88, 'Thrombocytopenia', '2024-05-08', 0, '48743');
```

Patient_Id	Total_Days_Hospitalized	Disease_Name	Enrolled_Date	Number_of_Deaths	Insurance_Claims
1	30	Alzimers Disease	2024-05-01	0	153000
2	45	Atrial Fibrillation	2024-05-02	0	250350
3	22	Septic Shock	2024-05-03	0	1545000
4	33	Brain Haemorrhage	2024-05-04	0	2545321
5	50	Addisons Disease	2024-05-05	0	90000
6	17	Pneumonia	2024-05-06	0	36455
7	70	Cirrhosis	2024-05-07	0	85956
8	88	Thrombocytopenia	2024-05-08	0	48743

Create a Temporary Table from above logic to rename an existing column

(ALTER - RENAME, CREATE TABLE, SELECT clauses)

Rename current data table as Temporary Table or Backup Table

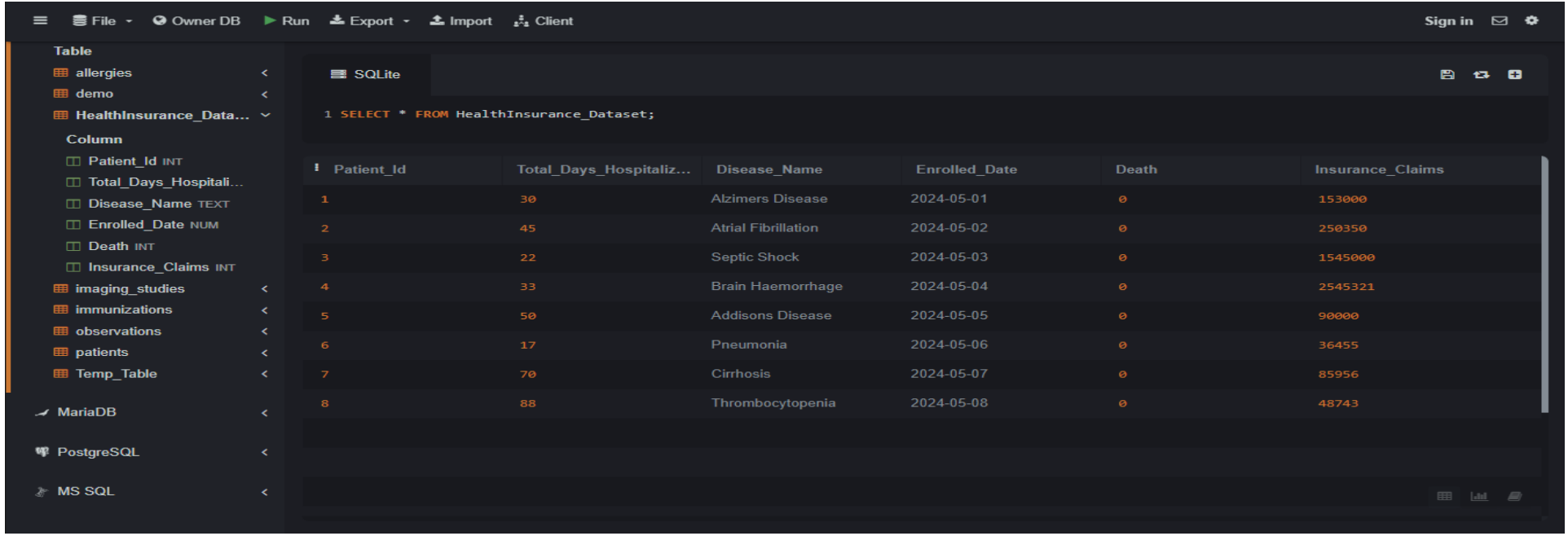
ALTER TABLE HealthInsurance_Dataset RENAME TO Temp_Table;

Create new data table again with the previous data table name, attributes, records

CREATE TABLE HealthInsurance_Dataset AS SELECT
Patient_Id, Total_Days_Hospitalized, Disease_Name, Enrolled_Date, Number_of_Deaths AS Death, Insurance_Claims FROM Temp_Table;

Now retrieve the column details in the newly created data table

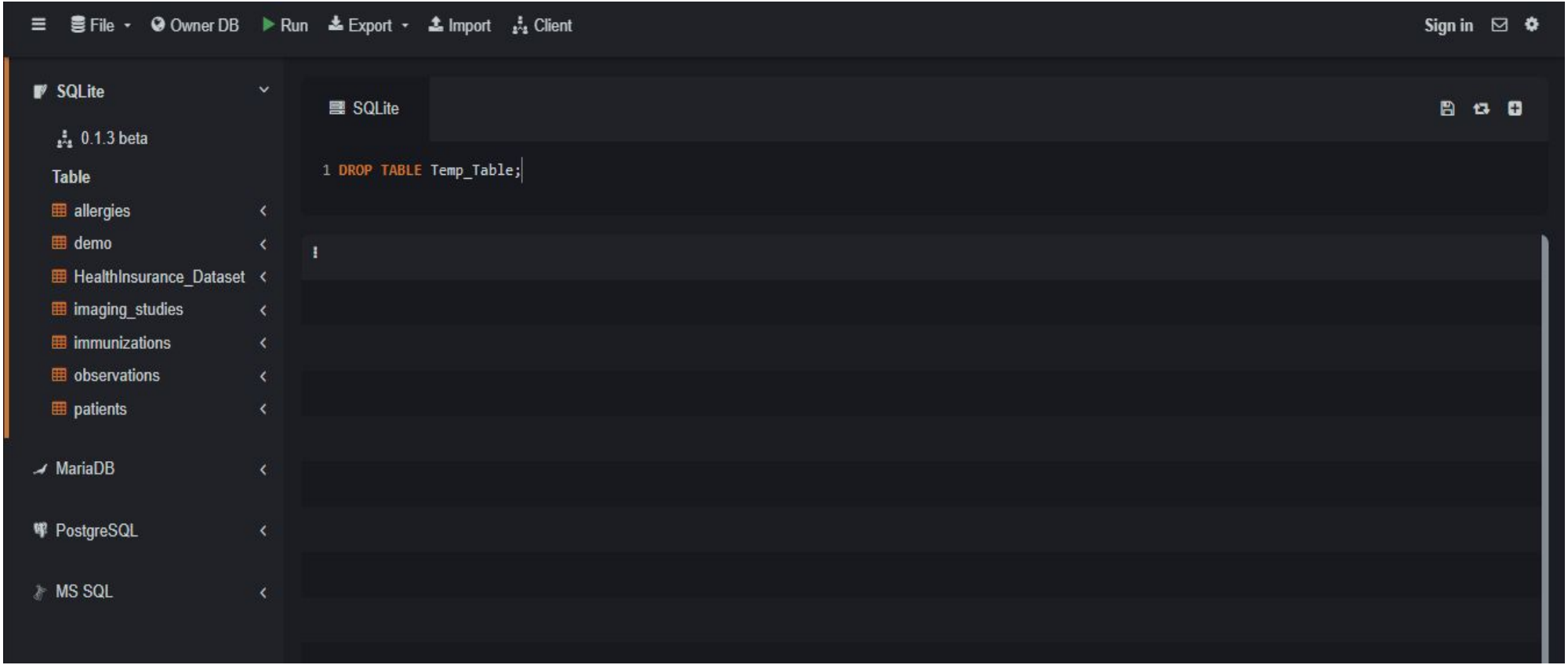
SELECT * FROM HealthInsurance_Dataset;



Patient_Id	Total_Days_Hospitaliz...	Disease_Name	Enrolled_Date	Death	Insurance_Claims
1	30	Alzimers Disease	2024-05-01	0	153000
2	45	Atrial Fibrillation	2024-05-02	0	250350
3	22	Septic Shock	2024-05-03	0	1545000
4	33	Brain Haemorrhage	2024-05-04	0	2545321
5	50	Addisons Disease	2024-05-05	0	90000
6	17	Pneumonia	2024-05-06	0	36455
7	70	Cirrhosis	2024-05-07	0	85956
8	88	Thrombocytopenia	2024-05-08	0	48743

Delete the Temporary Table ensuring there is no data loss in the parent table
(DELETE TABLE)

DROP TABLE Temp_Table;



Delete ‘ID’ column from the Imaging_Studies data table

(ALTER – DROP clause)

```
ALTER TABLE imaging_studies
DROP COLUMN
ID;
```

File | Owner DB | Run | Export | Import | Client | Sign in

SQLite

0.1.3 beta

Table

- allergies
- demo
- HealthInsurance_Dataset
- imaging_studies
  - Column
    - Date TEXT
    - PatientNumber TEXT
    - Encounter TEXT
    - Series_UID TEXT
    - BodySiteCode TEXT
    - Body_Site TEXT
    - Image_Code TEXT
    - Image_Description T...
    - Instance_UID TEXT
    - Storage_Code TEXT
    - Storage_Description...
    - Procedure_Code TEXT
- immunizations
- observations

SQLite

```
1 SELECT * FROM imaging_studies;
```

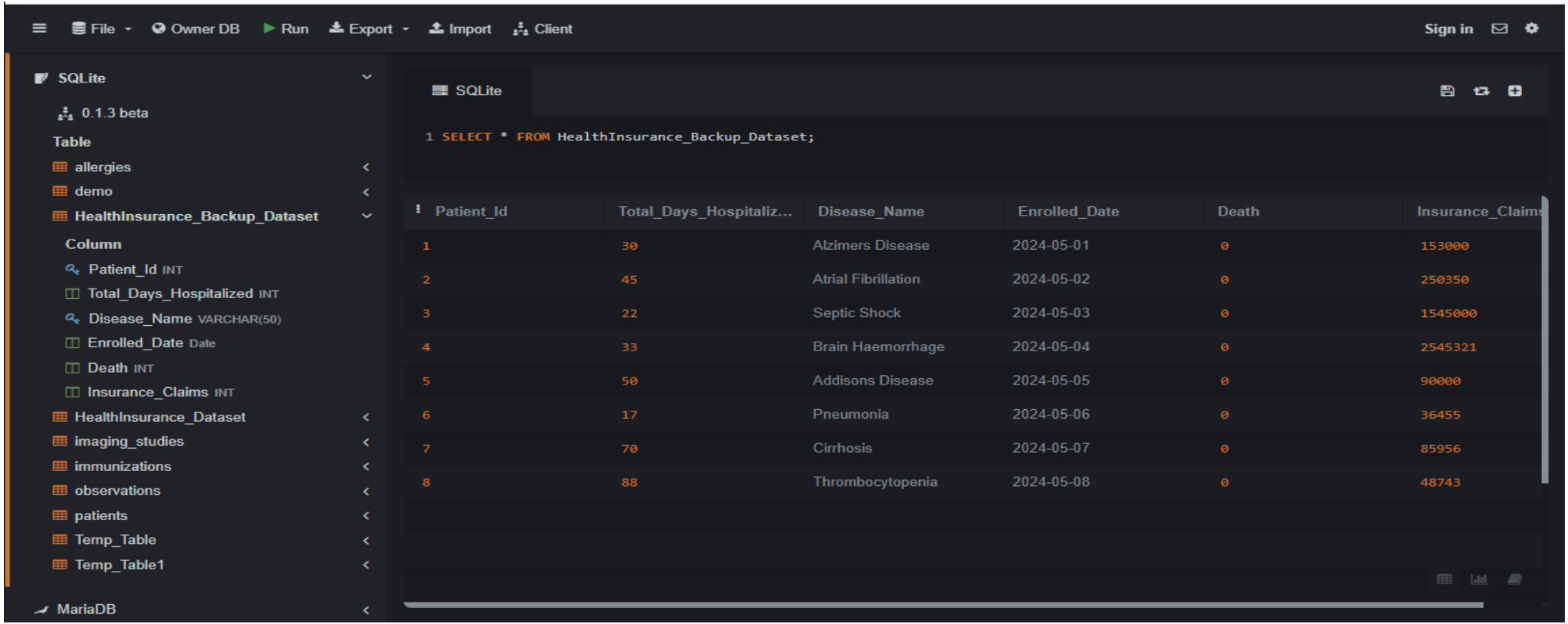
Date	Patient...	Encounter	Series_...	BodySit...	Body_Site	Image_...	Image_...	Instanc...	Storage...	Storage...	Procedure_C...
DATE	PATIENT	ENCOUN...	SERIES_...	BODYSI...	BODYSI...	MODALI...	MODALI...	INSTANC...	SOP_CO...	SOP_DE...	PROCEDURE_...
2017-02-...	c1f1fcaa-...	6474f606...	1.2.840.9...	344001	Ankle	DX	Digital Ra...	1.2.840.9...	1.2.840.1...	Digital X-...	19490002
2014-07-...	dc6c06d0...	1997235...	1.2.840.9...	344001	Ankle	DX	Digital Ra...	1.2.840.9...	1.2.840.1...	Digital X-...	19490002
2006-06-...	55a6a46...	0ef24554...	1.2.840.9...	40983000	Arm	DX	Digital Ra...	1.2.840.9...	1.2.840.1...	Digital X-...	1225002
2016-08-...	98e4223...	9a066f56...	1.2.840.9...	40983000	Arm	DX	Digital Ra...	1.2.840.9...	1.2.840.1...	Digital X-...	1225002
2017-11-...	98e4223...	3071e5d...	1.2.840.9...	51299004	Clavicle	DX	Digital Ra...	1.2.840.9...	1.2.840.1...	Digital X-...	168594001
2018-05-...	57b5331...	241d998f...	1.2.840.9...	50519007	Structure ...	US	Ultrasound	1.2.840.9...	1.2.840.1...	Ultrasoun...	426701000119...
2015-12-...	8bf6aa92...	4898e96...	1.2.840.9...	51299004	Clavicle	DX	Digital Ra...	1.2.840.9...	1.2.840.1...	Digital X-...	168594001
2013-01-...	7d28d76...	710bb3f4...	1.2.840.9...	50519007	Structure ...	US	Ultrasound	1.2.840.9...	1.2.840.1...	Ultrasoun...	426701000119...
2013-12-...	f93c78a3...	ccb84217...	1.2.840.9...	51299004	Clavicle	DX	Digital Ra...	1.2.840.9...	1.2.840.1...	Digital X-...	168594001
2015-07-...	5b2f8ddd...	7a3ae84...	1.2.840.9...	344001	Ankle	DX	Digital Ra...	1.2.840.9...	1.2.840.1...	Digital X-...	19490002

Create Primary Key and Foreign Key in a Backup Data Table by linking it to the Parent table

(PRIMARY KEY, FOREIGN KEY clauses)

Create a Backup table using Patient ID as Primary Key and Disease Name as Foreign Key

```
CREATE TABLE HealthInsurance_Backup_Dataset
(Patient_Id INT, Total_Days_Hospitalized INT,  Disease_Name VARCHAR(50) NOT NULL,  Enrolled_Date Date,  Death INT,  Insurance_Claims INT,  PRIMARY KEY (Patient_Id),
FOREIGN KEY (Disease_Name) REFERENCES HealthInsurance_Dataset(Disease_Name));
```



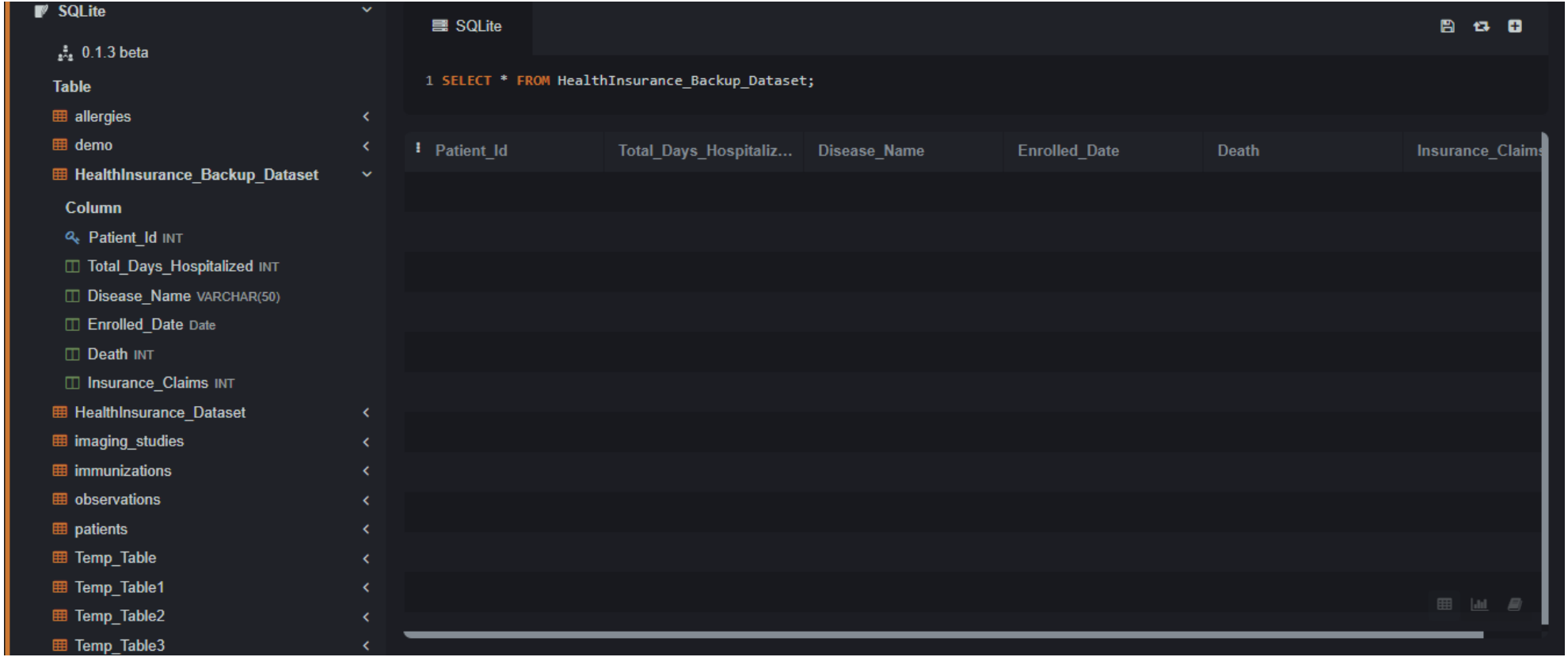
Patient_Id	Total_Days_Hospitaliz...	Disease_Name	Enrolled_Date	Death	Insurance_Claims
1	30	Alzimers Disease	2024-05-01	0	153000
2	45	Atrial Fibrillation	2024-05-02	0	250350
3	22	Septic Shock	2024-05-03	0	1545000
4	33	Brain Haemorrhage	2024-05-04	0	2545321
5	50	Addisons Disease	2024-05-05	0	90000
6	17	Pneumonia	2024-05-06	0	36455
7	70	Cirrhosis	2024-05-07	0	85956
8	88	Thrombocytopenia	2024-05-08	0	48743

Delete all rows in the backup data table created

(TRUNCATE clauses)

Delete all rows in the back up table created using above logic and then retrieve the back up table to confirm if logic is getting executed correctly

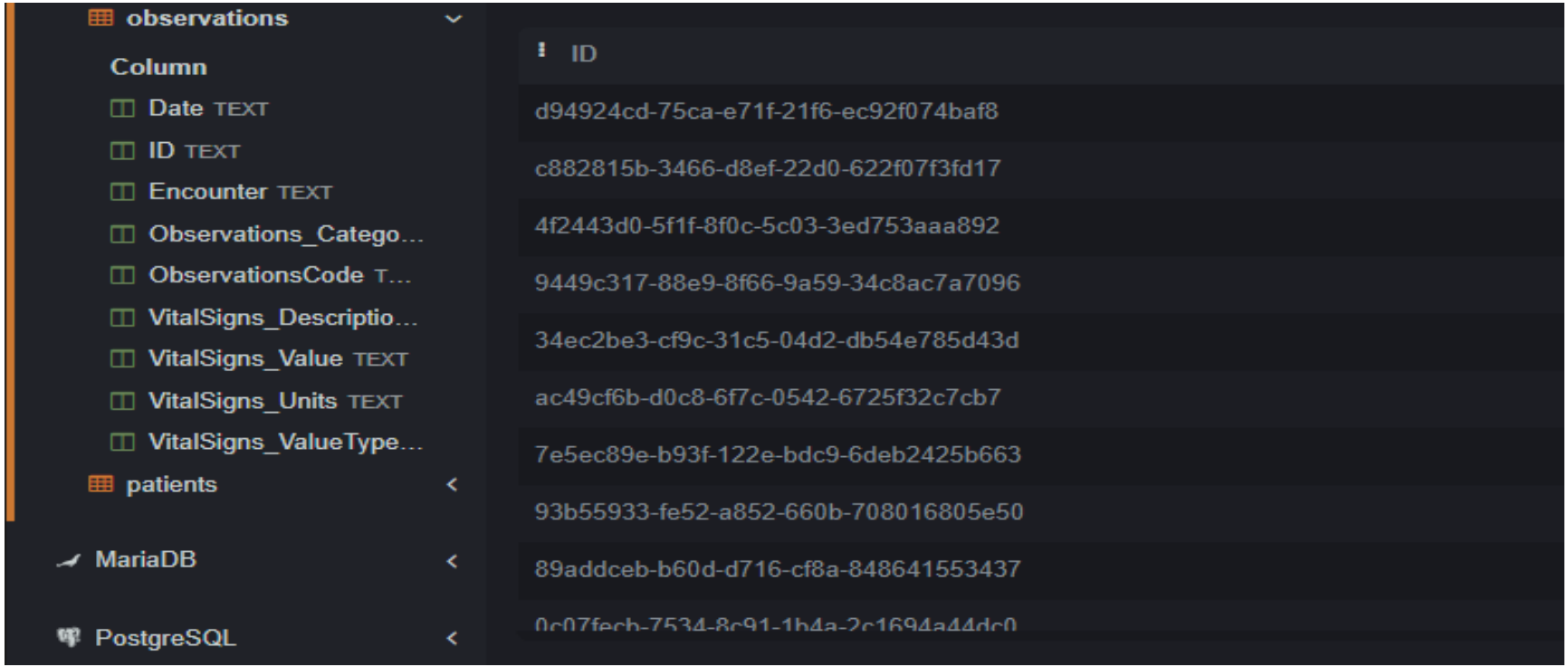
TRUNCATE HealthInsurance_Backup_Dataset;



Fetch IDs of the patients whose Image Description includes ‘Digital’ and Vital signs are related to ‘Heart’

(UNION clause to merge records from 2 different tables of same data type columns. It also pulls duplicate records from both tables in final result)

```
SELECT ID FROM imaging_studies where imaging_studies.Image_Description like '%Digital%'
UNION ALL
SELECT ID FROM observations where observations.vitalsigns_description like '%Heart%';
```



Fetch common records between IDs, encounter number in Imaging table, observations table

(Intersect clause to pull same records between 2 different tables with same data type columns)

```
SELECT patientnumber, encounter FROM imaging_studies
INTERSECT
SELECT ID, encounter FROM observations;
```

PatientNumber	Encounter
03e502b6-b810-06c1-7d65-83db077ed3ee	4e26a71c-1358-adeb-542e-0d1617da4f78
03e502b6-b810-06c1-7d65-83db077ed3ee	4eb71790-9084-f515-74a0-27ed49d450a9
03e502b6-b810-06c1-7d65-83db077ed3ee	568c0432-30fa-d68c-efec-65b213f3d0d5
03e502b6-b810-06c1-7d65-83db077ed3ee	56cd7cef-225a-c6da-8603-9053ba39a512
03e502b6-b810-06c1-7d65-83db077ed3ee	57a42a04-f4f4-ef51-1f09-f0492410ba18
03e502b6-b810-06c1-7d65-83db077ed3ee	5b32f94c-add8-5818-b229-fb1bf9970bf7
03e502b6-b810-06c1-7d65-83db077ed3ee	5f8a287a-7c29-ea40-8055-4a1723e62143
03e502b6-b810-06c1-7d65-83db077ed3ee	691920c6-ea1a-0128-8140-a1997fbc12f0
03e502b6-b810-06c1-7d65-83db077ed3ee	6d89d0bd-7349-ac9e-fe6f-7d9ced20e243
03e502b6-b810-06c1-7d65-83db077ed3ee	75453926-6e04-ac94-7bde-40be83951f42

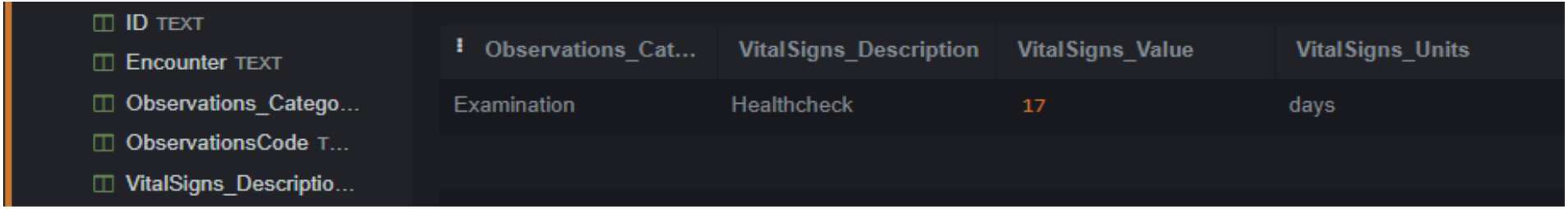
Add a single row to the columns in Observations data table

(Insert clause is used to add a row in the data table. Then write Select, Where clause to retrieve added row)

```
INSERT INTO observations
        (observations_category,
         vitalsigns_description,
            vitalsigns_value,
            vitalsigns_units)
             VALUES
                    (
                 'Examination',
                 'Healthcheck',
                       '17',
                      'days'
                    );
```

<u>Fetch results using below query:-</u>

```
SELECT observations_category, vitalsigns_description, vitalsigns_value, vitalsigns_units
from observations
where vitalsigns_description in ('Healthcheck')
```



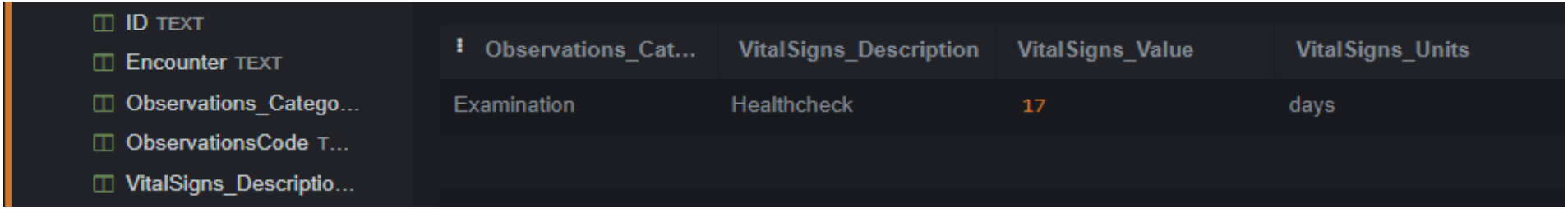
Add a single row to the columns in Observations data table

(Insert clause is used to add a row in the data table. Then write Select, Where clause to retrieve added row)

```
INSERT INTO observations
        (observations_category,
         vitalsigns_description,
            vitalsigns_value,
            vitalsigns_units)
             VALUES
                    (
                 'Examination',
                 'Healthcheck',
                       '17',
                      'days'
                    );
```

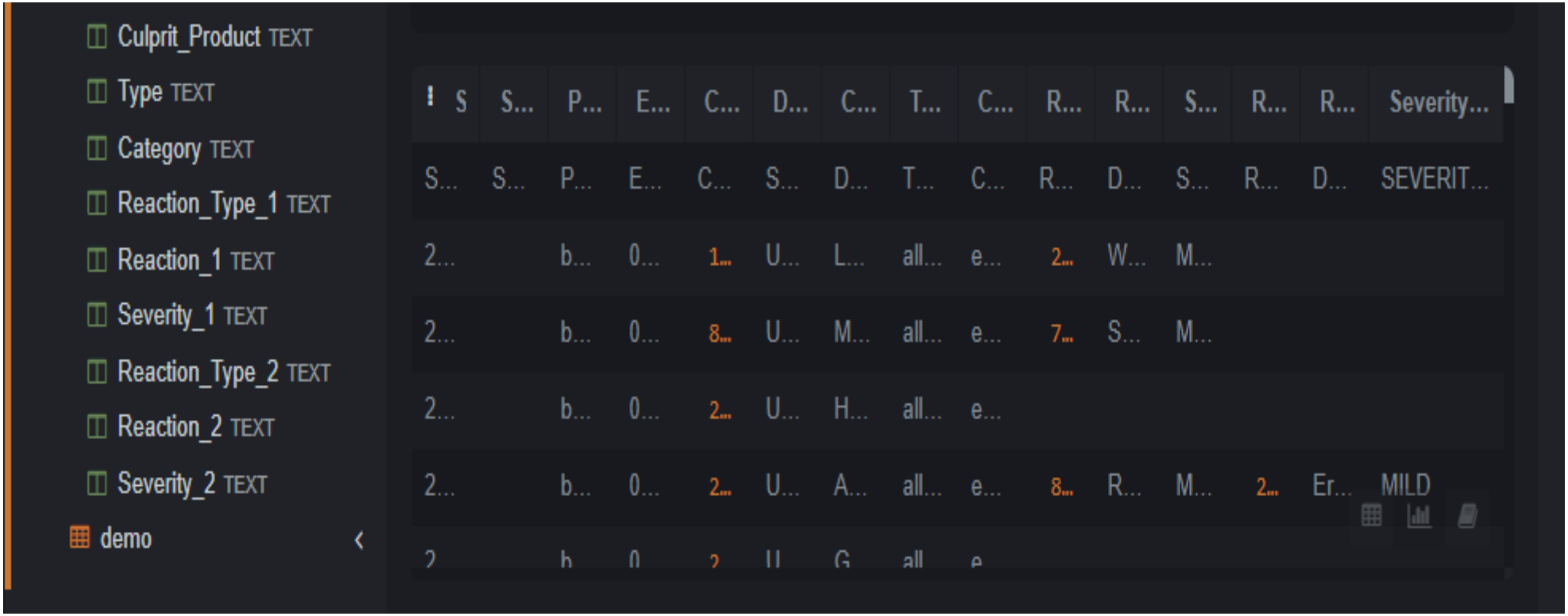
<u>Fetch results using below query:-</u>

```
SELECT observations_category, vitalsigns_description, vitalsigns_value, vitalsigns_units
from observations
where vitalsigns_description in ('Healthcheck')
```



Query all rows and columns from a table

- SELECT * FROM allergies;



Retrieve all Start Dates for the Category with food allergy

- select Start, Category from allergies

  where Category='food';

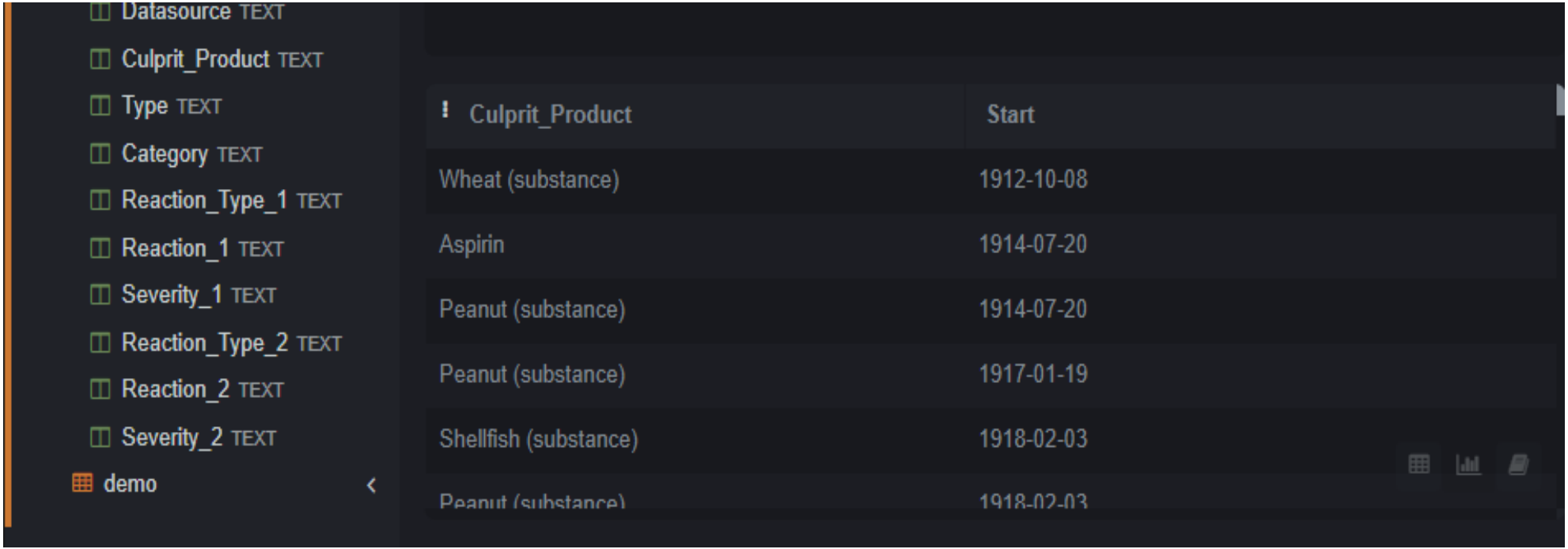
Retrieve distinct (Unique) Start Dates for ‘medication allergy’ Category

select DISTINCT Start from allergies

where Category='medication';

Retrieve Start Date for all Culprit Products in Ascending Order

```
select culprit_product,Start from allergies
order BY Start;
```



Culprit_Product	Start
Wheat (substance)	1912-10-08
Aspirin	1914-07-20
Peanut (substance)	1914-07-20
Peanut (substance)	1917-01-19
Shellfish (substance)	1918-02-03
Peanut (substance)	1918-02-03

Retrieve Start Date for all Culprit Products in Descending Order

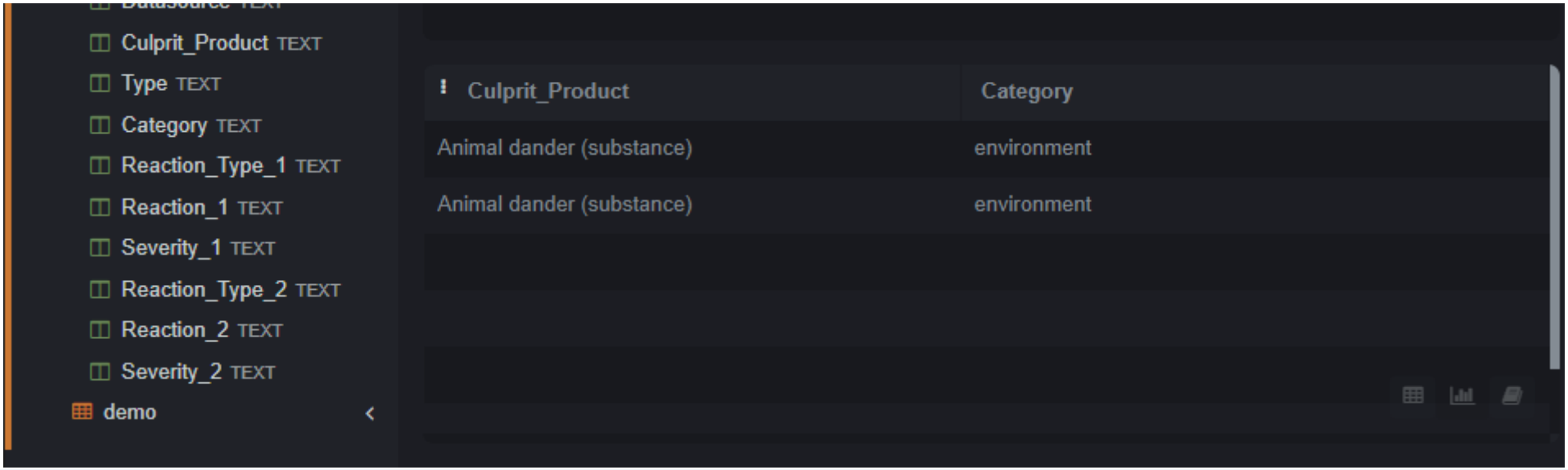
```
select culprit_product,Start from allergies
order BY Start DESC;
```

- Culprit_Product TEXT
- Type TEXT
- Category TEXT
- Reaction_Type_1 TEXT
- Reaction_1 TEXT
- Severity_1 TEXT
- Reaction_Type_2 TEXT
- Reaction_2 TEXT
- Severity_2 TEXT
- demo

Culprit_Product	Start
DESCRIPTION	START
Mold (organism)	2021-02-15
House dust mite (organism)	2021-02-15
Animal dander (substance)	2021-02-15
Grass pollen (substance)	2021-02-15
Tree pollen (substance)	2021-02-15

Retrieve only first 2 rows for Culprit Products, Category in Ascending Order

```
SELECT culprit_product, Category
FROM allergies
ORDER BY culprit_product, Category
LIMIT 2 ;
```



Retrieve top 2nd, 3rd unique Start Dates by excluding top 1st row

- SELECT DISTINCT Start

  FROM allergies

  LIMIT 2 OFFSET 1;

Modify all Column names in the data table

ALTER TABLE allergies RENAME COLUMN c1 TO Start; ALTER TABLE allergies RENAME COLUMN c2 TO Stop;

ALTER TABLE allergies RENAME COLUMN c3 TO Patient; ALTER TABLE allergies RENAME COLUMN c4 TO Encounter;

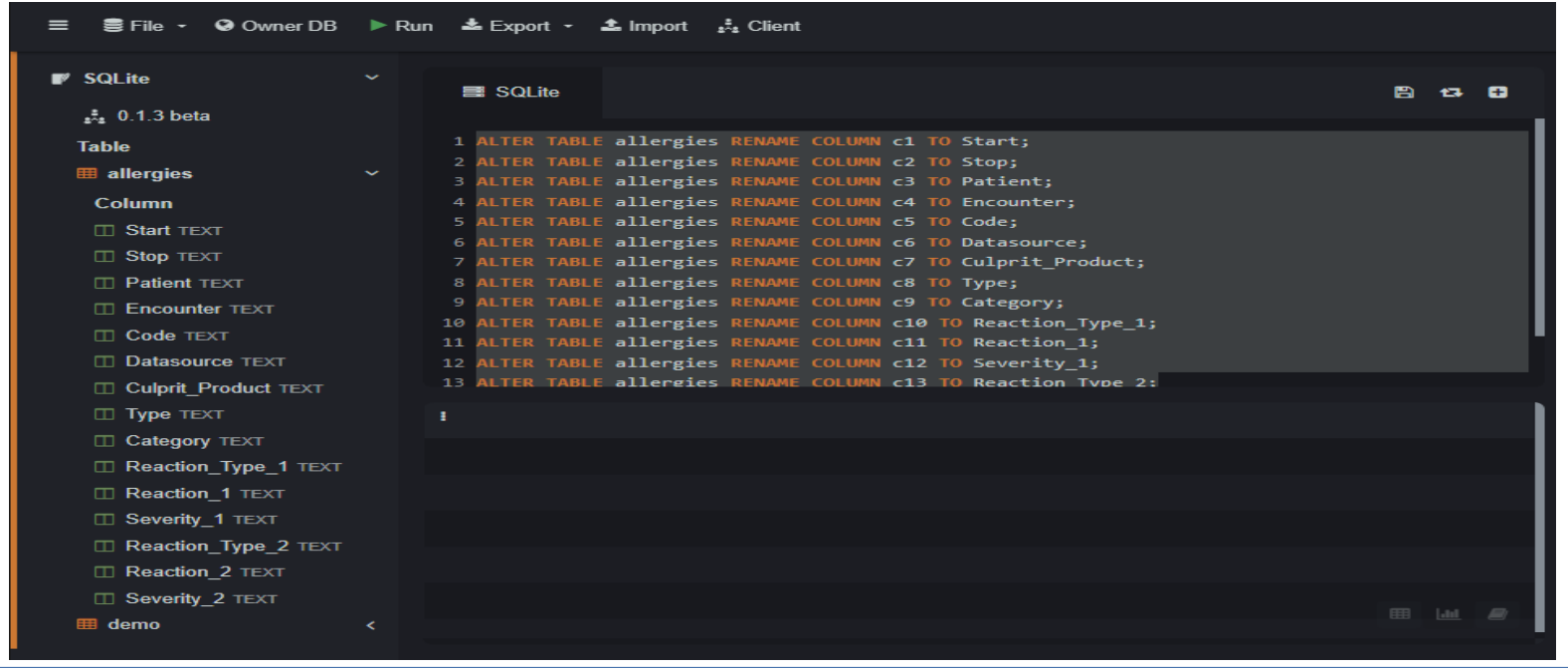
ALTER TABLE allergies RENAME COLUMN c5 TO Code; ALTER TABLE allergies RENAME COLUMN c6 TO Data_source;

ALTER TABLE allergies RENAME COLUMN c7 TO Culprit_Product; ALTER TABLE allergies RENAME COLUMN c8 TO Type;

ALTER TABLE allergies RENAME COLUMN c9 TO Category; ALTER TABLE allergies RENAME COLUMN c10 TO Reaction_Type_1;

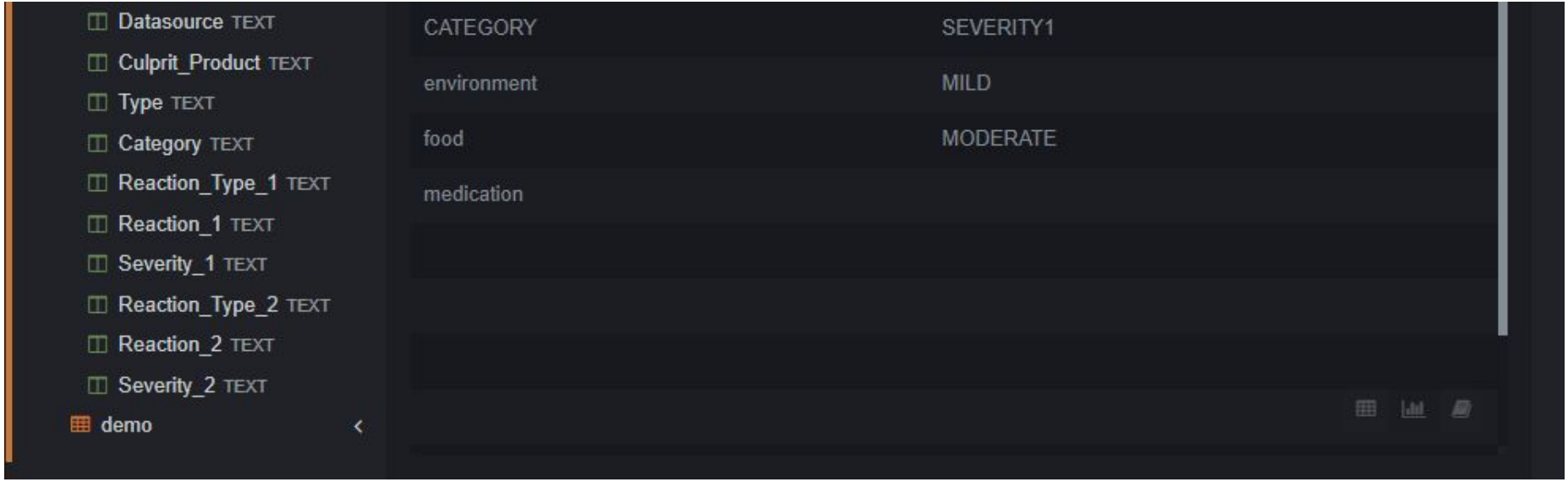
ALTER TABLE allergies RENAME COLUMN c11 TO Reaction_1; ALTER TABLE allergies RENAME COLUMN c12 TO Severity_1;

ALTER TABLE allergies RENAME COLUMN c13 TO Reaction_Type_2; ALTER TABLE allergies RENAME COLUMN c14 TO Reaction_2; ALTER TABLE allergies RENAME COLUMN c15 TO Severity_2;



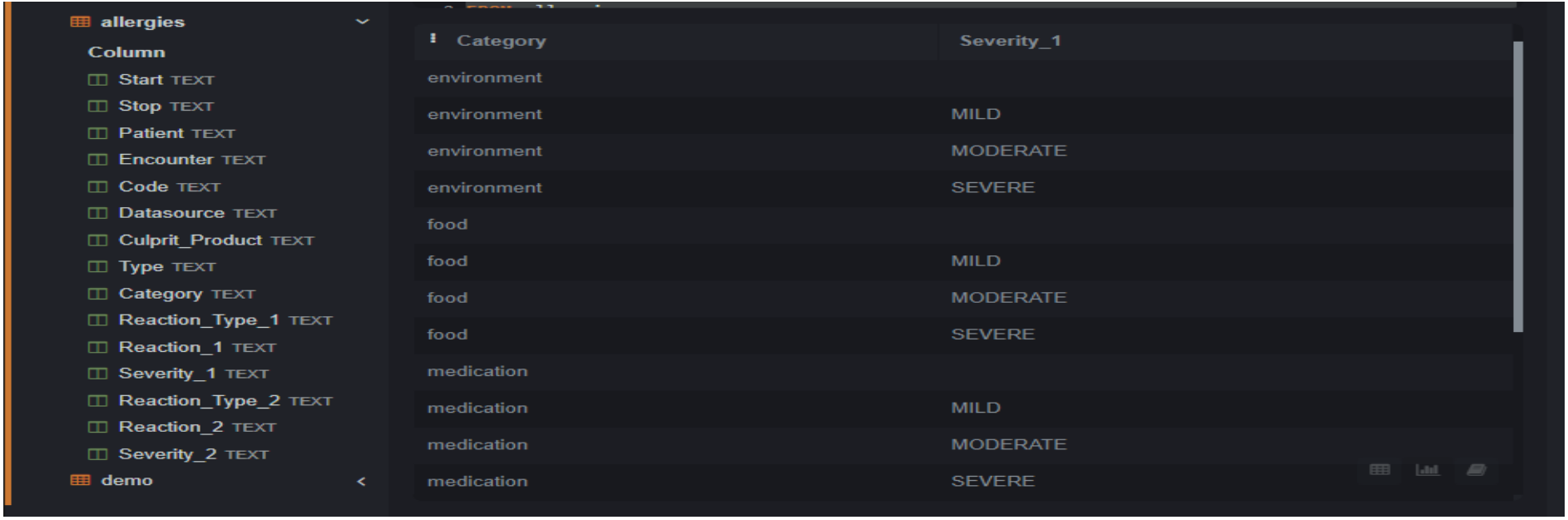
Aggregate various types of allergic Categories based on their Severity grade (Group By Clause)

- SELECT category, severity_1
  FROM allergies
  GROUP BY category;



Aggregate both allergic Categories and Severity grade (Group By Clause)

- SELECT category, severity_1
  FROM allergies
  GROUP BY category, severity_1;



Category	Severity_1
environment	
environment	MILD
environment	MODERATE
environment	SEVERE
food	
food	MILD
food	MODERATE
food	SEVERE
medication	
medication	MILD
medication	MODERATE
medication	SEVERE

Retrieve all Culprit Products, Severity grades with highest Code values (Group By Clause along with Aggregate function)

```
SELECT culprit_product, severity_1, code
FROM allergies
group BY severity_1 HAVING max (code) ;
```

- Stop TEXT
- Patient TEXT
- Encounter TEXT
- Code TEXT
- Datasource TEXT
- Culprit_Product TEXT
- Type TEXT
- Category TEXT
- Reaction_Type_1 TEXT

Culprit_Product	Severity_1	Code
Mold (organism)		84489001
Mold (organism)	MILD	84489001
Penicillin V	MODERATE	7984
Peanut (substance)	SEVERE	762952008

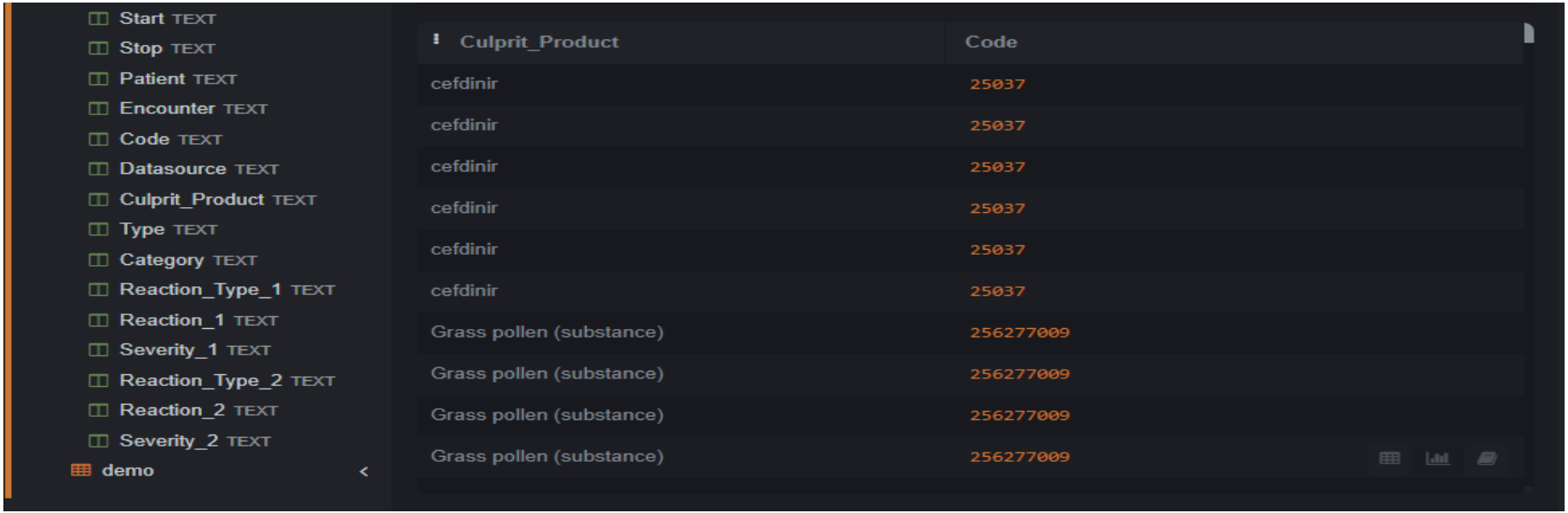
Retrieve all Culprit Products with ‘Mild’ & ‘Moderate’ Severity grades only (Where Clause)

```
SELECT culprit_product, severity_1
FROM allergies
WHERE severity_1 BETWEEN 'MILD' AND 'MODERATE';
```

Culprit_Product	Severity_1
Latex (substance)	MILD
Mold (organism)	MILD
Animal dander (substance)	MODERATE
Cow's milk (substance)	MODERATE
Soya bean (substance)	MODERATE
Shellfish (substance)	MODERATE
Shellfish (substance)	MODERATE
Mold (organism)	MILD
Animal dander (substance)	MODERATE
Aspirin	MODERATE

Retrieve all Culprit Products with Codes greater than 20000 by Sorting in Ascending Order (Combine Where & Order By clauses)

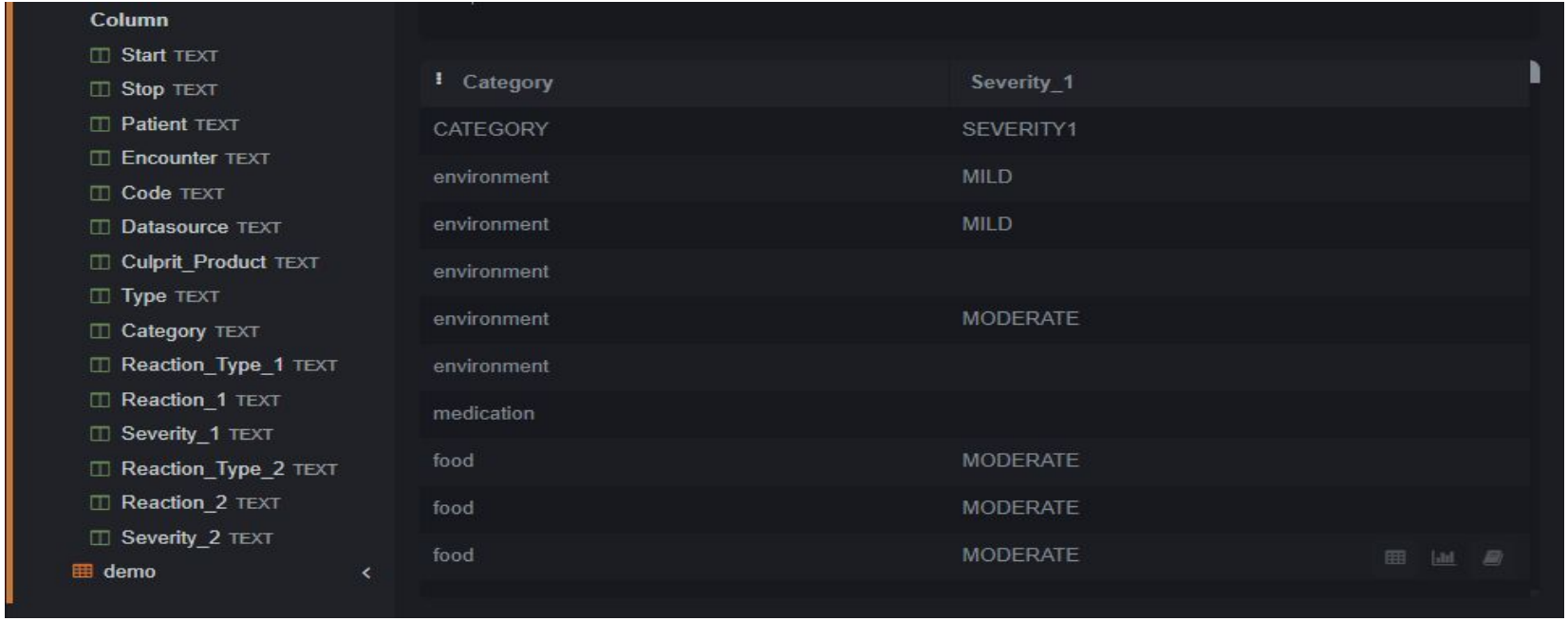
```
SELECT culprit_product, code
FROM allergies
WHERE code >20000 order by code;
```



Culprit_Product	Code
cefdinir	25037
cefdinir	25037
cefdinir	25037
cefdinir	25037
cefdinir	25037
cefdinir	25037
Grass pollen (substance)	256277009
Grass pollen (substance)	256277009
Grass pollen (substance)	256277009
Grass pollen (substance)	256277009

Retrieve all Severity grades where Category is not blank (Not a Null clause)

SELECT category, severity_1 FROM allergies
WHERE category is NOT NULL;



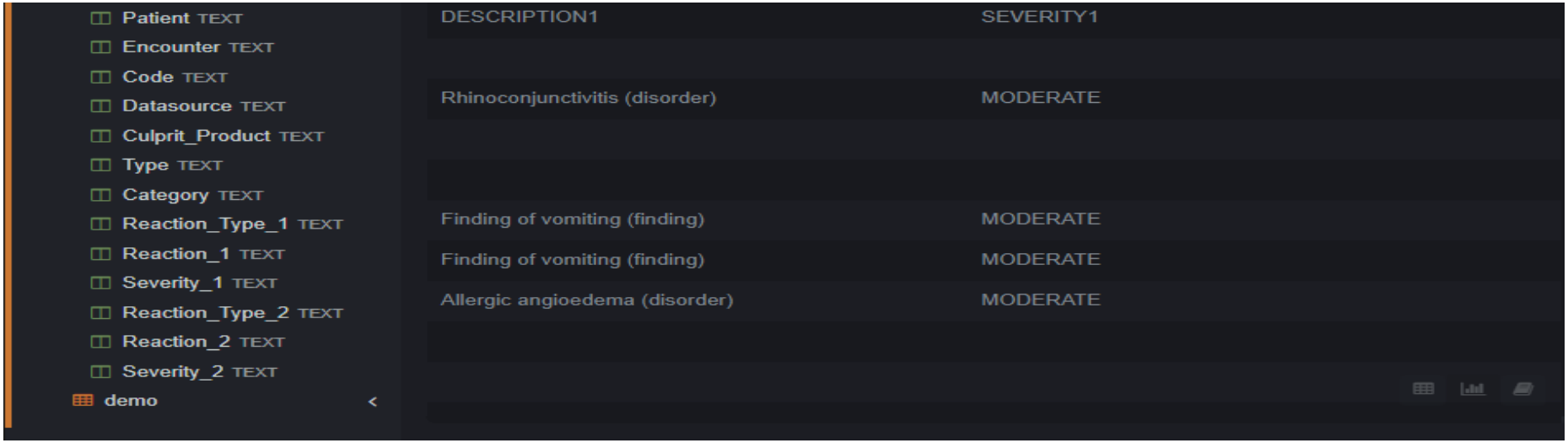
Category	Severity_1
CATEGORY	SEVERITY1
environment	MILD
environment	MILD
environment	
environment	MODERATE
environment	
medication	
food	MODERATE
food	MODERATE
food	MODERATE

Retrieve all Categories with empty Severity grade (Null clause)

```
SELECT category, severity_1 FROM allergies
WHERE severity_1 = '' or severity_1 is NULL;
```

Retrieve allergic reactions which are not ‘Mild’ in severity (Like clause to find specific characters/search patterns)

```
SELECT reaction_1, severity_1
FROM allergies
WHERE severity_1 NOT LIKE '%mil%';
```



Retrieve allergic reactions codes which is not in the range of 35000 (Where clause to exclude single specific character and pull rest of the rows)

```
SELECT reaction_1, code
FROM allergies
WHERE code <> 35000;
```

- Patient TEXT
- Encounter TEXT
- Code TEXT
- Datasource TEXT
- Culprit_Product TEXT
- Type TEXT
- Category TEXT
- Reaction_Type_1 TEXT
- Reaction_1 TEXT
- Severity_1 TEXT
- Reaction_Type_2 TEXT
- Reaction_2 TEXT
- Severity_2 TEXT
- demo

DESCRIPTION1	CODE
Wheal (finding)	111088007
Sneezing	84489001
	260147004
Rhinoconjunctivitis (disorder)	264287008
	256277009
	1191
Finding of vomiting (finding)	3718001
Finding of vomiting (finding)	256355007
Allergic angioedema (disorder)	735029006

Retrieve allergic reactions codes which is not in the range of 35000 (Where clause to exclude multiple characters and pull other rows)

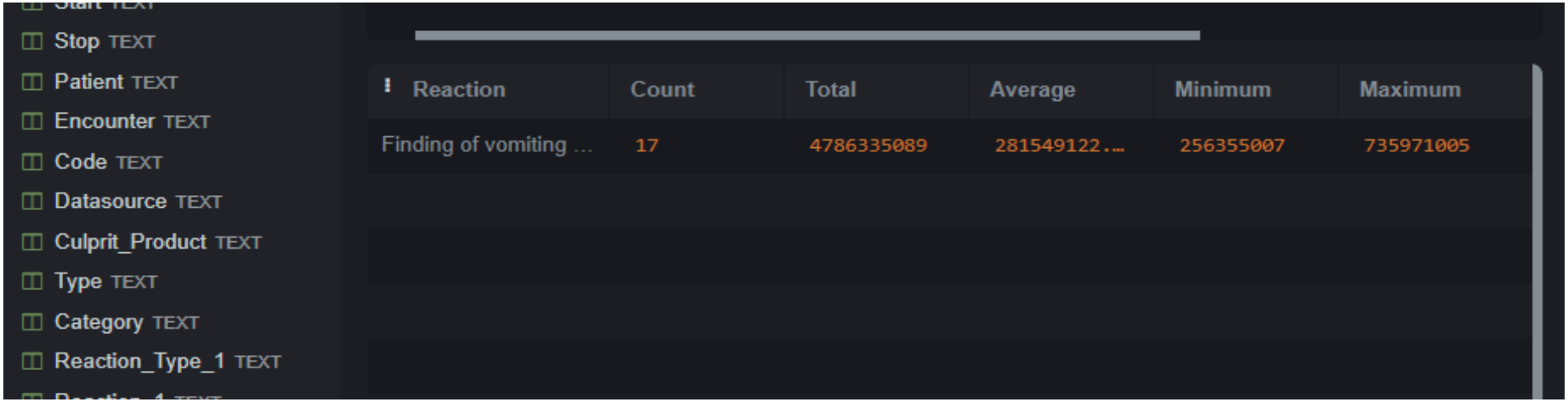
```
SELECT reaction_1, code
FROM allergies
WHERE code NOT IN (35000, 36000, 37000);
```

- Patient TEXT
- Encounter TEXT
- Code TEXT
- Datasource TEXT
- Culprit_Product TEXT
- Type TEXT
- Category TEXT
- Reaction_Type_1 TEXT
- Reaction_1 TEXT
- Severity_1 TEXT
- Reaction_Type_2 TEXT
- Reaction_2 TEXT
- Severity_2 TEXT
- demo

DESCRIPTION1	CODE
Wheal (finding)	111088007
Sneezing	84489001
	260147004
Rhinoconjunctivitis (disorder)	264287008
	256277009
	1191
Finding of vomiting (finding)	3718001
Finding of vomiting (finding)	256355007
Allergic angioedema (disorder)	735029006

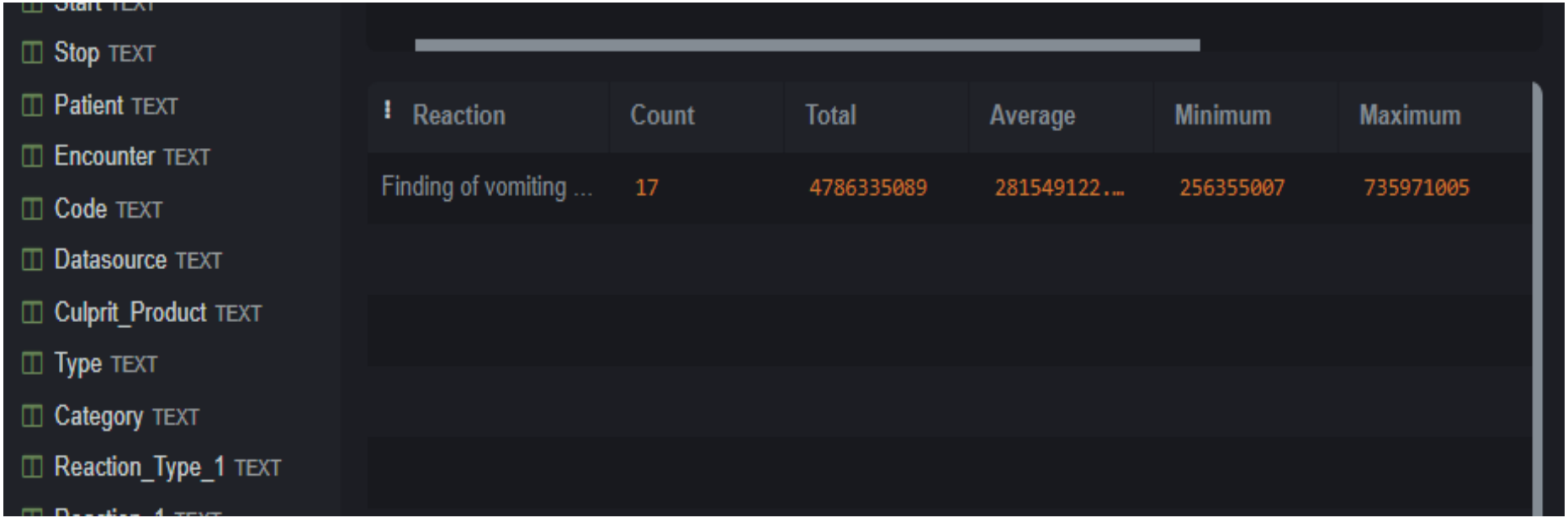
Retrieve all Aggregation code values for ‘Vomiting’ allergic reaction (Count, Sum, Average, Minimum, Maximum and name the new columns)

```
SELECT reaction_1 as Reaction, Count(code) as Count, SUM(code) as Total, AVG(code) as Average, MIN(code)
as Minimum, MAX(code) as Maximum
FROM allergies
WHERE reaction_1 LIKE '%vomit%';
```



Skip the first 5 records and retrieve the next 5 records alone for Reaction, culprit products with respect to severity
(LIMIT 5 limits the results to 5 rows, OFFSET 5 skips the first 5 rows of the result)

```
SELECT reaction_1, severity_1, culprit_product
FROM allergies
ORDER BY severity_1 DESC
LIMIT 5 OFFSET 5;
```



SQL Comparison Operators
(=, <>, >, <, >=, <=)

=Equal, <>Not equal to, > Greater than, >=Greater than or equal to, < Less than, <=Less than or equal to

```
SELECT reaction_1, severity_1, culprit_product, code
FROM allergies
WHERE code <> ('3000, 55000, 7500');
```

Reaction_1	Severity_1	Culprit_Product	Code
DESCRIPTION1	SEVERITY1	DESCRIPTION	CODE
Wheal (finding)	MILD	Latex (substance)	111088007
Sneezing	MILD	Mold (organism)	84489001
		House dust mite (orga...	260147004
Rhinoconjunctivitis (dis...	MODERATE	Animal dander (substa...	264287008
		Grass pollen (substance)	256277009
		Aspirin	1191
Finding of vomiting (fin...	MODERATE	Cow's milk (substance)	3718001
Finding of vomiting (fin...	MODERATE	Soya bean (substance)	256355007

SQL Logical Operators
(All, And, Any, Between, Exists, In, Like, Not in, Or, Some Clauses)

(Not In Clause)

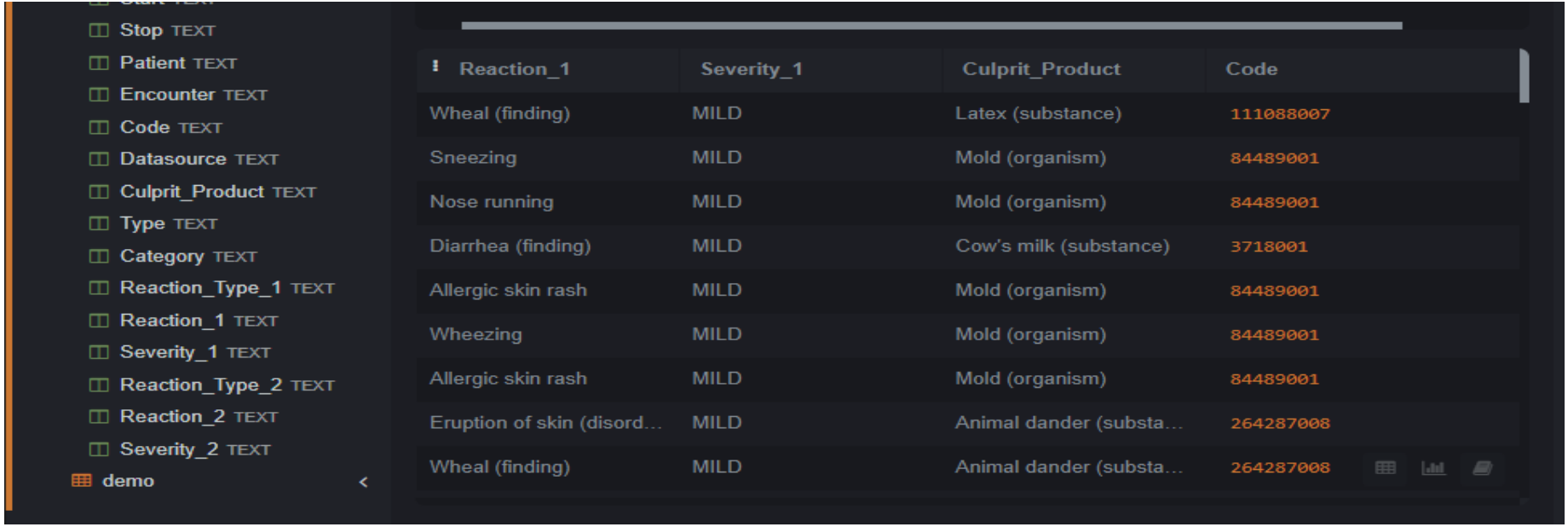
```
SELECT reaction_1, severity_1, culprit_product
FROM allergies
WHERE reaction_1 not IN ('Anaphylaxis', 'Finding of vomiting (finding)', 'Sneezing');
```

- Stop TEXT
- Patient TEXT
- Encounter TEXT
- Code TEXT
- Datasource TEXT
- Culprit_Product TEXT
- Type TEXT
- Category TEXT
- Reaction_Type_1 TEXT
- Reaction_1 TEXT
- Severity_1 TEXT
- Reaction_Type_2 TEXT
- Reaction_2 TEXT
- Severity_2 TEXT
- demo

Reaction_1	Severity_1	Culprit_Product
DESCRIPTION1	SEVERITY1	DESCRIPTION
Wheal (finding)	MILD	Latex (substance)
		House dust mite (organism)
Rhinoconjunctivitis (disorder)	MODERATE	Animal dander (substance)
		Grass pollen (substance)
		Aspirin
Allergic angioedema (disorder)	MODERATE	Shellfish (substance)
		Cow's milk (substance)
		Mold (organism)

Sample code for SQL Logical Operators along with syntax rules (AND, OR, NOT, NULL)

```
SELECT reaction_1, severity_1, culprit_product, code
FROM allergies
WHERE code >= 1000 AND (severity_1 = 'MILD' OR reaction_1 = 'Sneezing') AND NOT (culprit_product IS NULL);
```

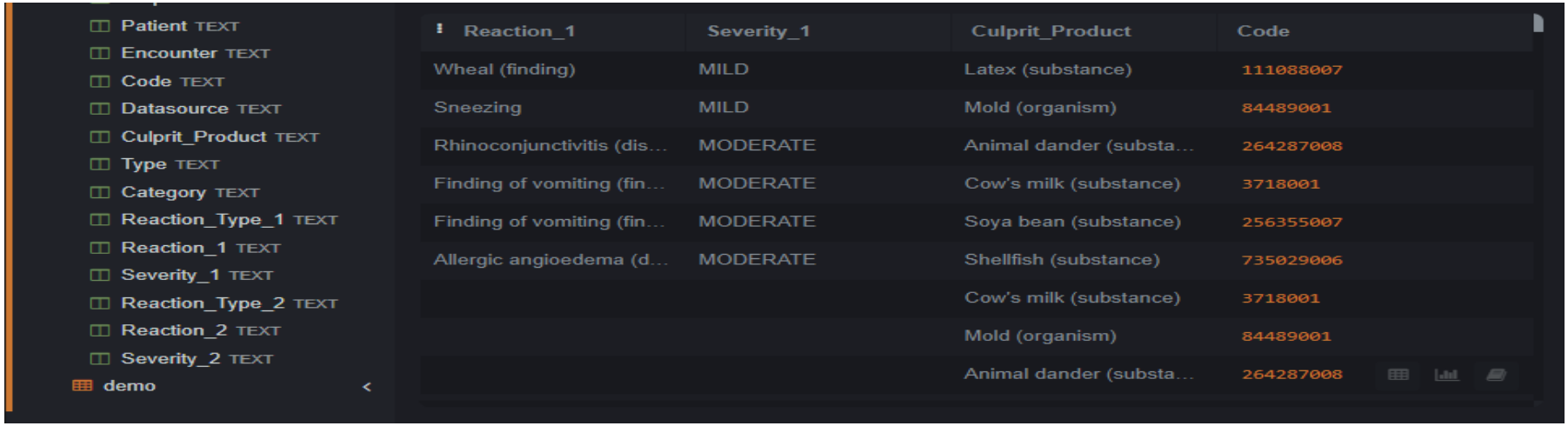


Reaction_1	Severity_1	Culprit_Product	Code
Wheal (finding)	MILD	Latex (substance)	111088007
Sneezing	MILD	Mold (organism)	84489001
Nose running	MILD	Mold (organism)	84489001
Diarrhea (finding)	MILD	Cow's milk (substance)	3718001
Allergic skin rash	MILD	Mold (organism)	84489001
Wheezing	MILD	Mold (organism)	84489001
Allergic skin rash	MILD	Mold (organism)	84489001
Eruption of skin (disord...	MILD	Animal dander (substa...	264287008
Wheal (finding)	MILD	Animal dander (substa...	264287008

Provide a list of reactions, severity, products that caused them, and their codes from the allergies table, but only for those records where the code is greater than or equal to 1000, the severity is ‘MILD’ or the reaction is ‘Sneezing’, and the culprit product is not null?

(combination of Select, Where, Exists, AND, AS, OR, NOT, NULL, >=, = clauses)

```
SELECT reaction_1, severity_1, culprit_product, code
FROM allergies
WHERE EXISTS (
   SELECT 1
   FROM allergies AS a
   WHERE a.code = allergies.code AND a.code >= 1000 AND (a.severity_1 = 'MILD' OR a.reaction_1 = 'Sneezing')) AND NOT (culprit_product IS NULL);SELECT
reaction_1, severity_1, culprit_product, code
FROM allergies
WHERE code >= 1000 AND (severity_1 = 'MILD' OR reaction_1 = 'Sneezing') AND NOT (culprit_product IS NULL);
```

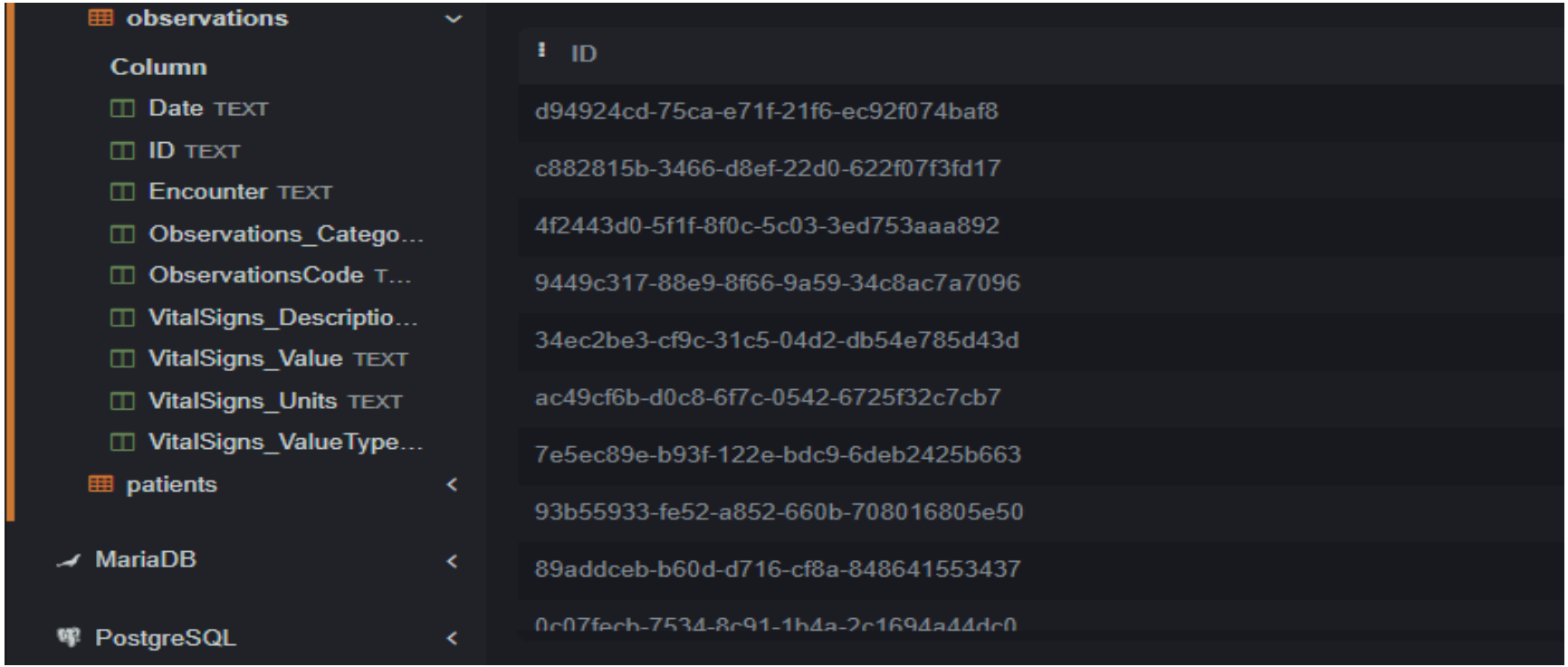


Reaction_1	Severity_1	Culprit_Product	Code
Wheal (finding)	MILD	Latex (substance)	111088007
Sneezing	MILD	Mold (organism)	84489001
Rhinoconjunctivitis (dis...	MODERATE	Animal dander (substa...	264287008
Finding of vomiting (fin...	MODERATE	Cow's milk (substance)	3718001
Finding of vomiting (fin...	MODERATE	Soya bean (substance)	256355007
Allergic angioedema (d...	MODERATE	Shellfish (substance)	735029006
		Cow's milk (substance)	3718001
		Mold (organism)	84489001
		Animal dander (substa...	264287008

Fetch IDs of the patients whose Image Description includes ‘Digital’ and Vital signs are related to ‘Heart’

(UNION clause to merge records from 2 different tables of same data type columns. It also pulls duplicate records from both tables in final result)

```
SELECT ID FROM imaging_studies where imaging_studies.Image_Description like '%Digital%'
UNION ALL
SELECT ID FROM observations where observations.vitalsigns_description like '%Heart%';
```



Fetch common records between IDs, encounter number in Imaging table, observations table

(Intersect clause to pull same records between 2 different tables with same data type columns)

```
SELECT patientnumber, encounter FROM imaging_studies
INTERSECT
SELECT ID, encounter FROM observations;
```

PatientNumber	Encounter
03e502b6-b810-06c1-7d65-83db077ed3ee	4e26a71c-1358-adeb-542e-0d1617da4f78
03e502b6-b810-06c1-7d65-83db077ed3ee	4eb71790-9084-f515-74a0-27ed49d450a9
03e502b6-b810-06c1-7d65-83db077ed3ee	568c0432-30fa-d68c-efec-65b213f3d0d5
03e502b6-b810-06c1-7d65-83db077ed3ee	56cd7cef-225a-c6da-8603-9053ba39a512
03e502b6-b810-06c1-7d65-83db077ed3ee	57a42a04-f4f4-ef51-1f09-f0492410ba18
03e502b6-b810-06c1-7d65-83db077ed3ee	5b32f94c-add8-5818-b229-fb1bf9970bf7
03e502b6-b810-06c1-7d65-83db077ed3ee	5f8a287a-7c29-ea40-8055-4a1723e62143
03e502b6-b810-06c1-7d65-83db077ed3ee	691920c6-ea1a-0128-8140-a1997fbc12f0
03e502b6-b810-06c1-7d65-83db077ed3ee	6d89d0bd-7349-ac9e-fe6f-7d9ced20e243
03e502b6-b810-06c1-7d65-83db077ed3ee	75453926-6e04-ac94-7bde-40be83951f42

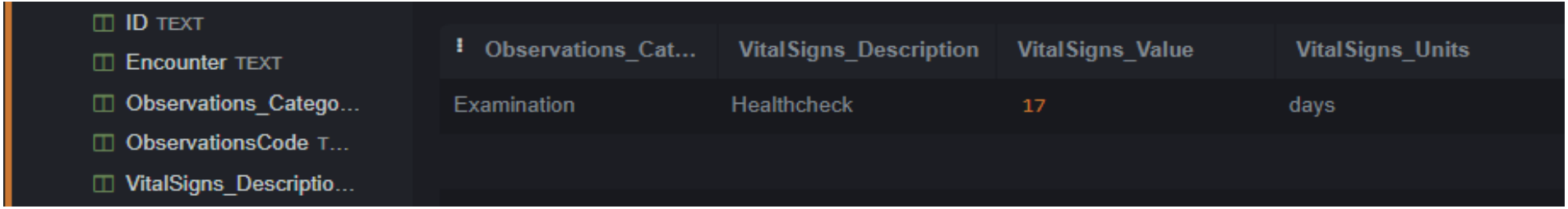
Add a single row to the columns in Observations data table

(Insert clause is used to add a row in the data table. Then write Select, Where clause to retrieve added row)

```
INSERT INTO observations
        (observations_category,
         vitalsigns_description,
            vitalsigns_value,
            vitalsigns_units)
             VALUES
                    (
                 'Examination',
                 'Healthcheck',
                       '17',
                      'days'
                    );
```

<u>Fetch results using below query:-</u>

```
SELECT observations_category, vitalsigns_description, vitalsigns_value, vitalsigns_units
from observations
where vitalsigns_description in ('Healthcheck')
```



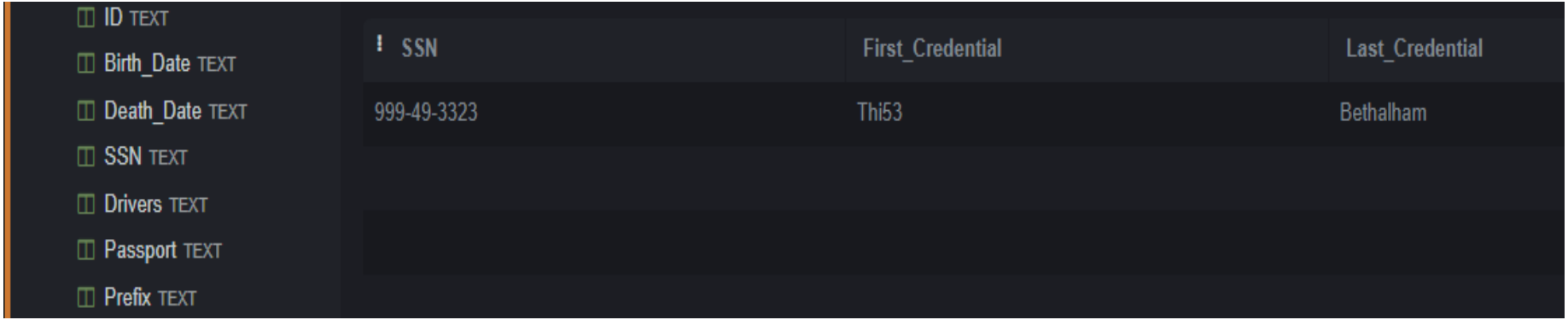
Modify Last Name of the patient based on SSN number and retrieve the result (UPDATE clause to modify the record)

Modify a record in the patients table

```
UPDATE patients
SET
last_credential = 'Bethalham'
WHERE
SSN = 999-49-3323;
```

<u>Retrieve the results using below query</u>

```
SELECT SSN, first_credential, last_credential
from patients
where SSN = '999-49-3323';
```



Based on previous tab query, make changes made in patients data table (child table) available in Master data table (parent table)

If a parent data table was available with Last Credential column based on SSN number, then the changes made in child data table should exist in parent table as well.

```
UPDATE Master Table
SET last_credential = (
SELECT
last_credential
FROM
patients
WHERE
patients_SSN = observations_SSN
```

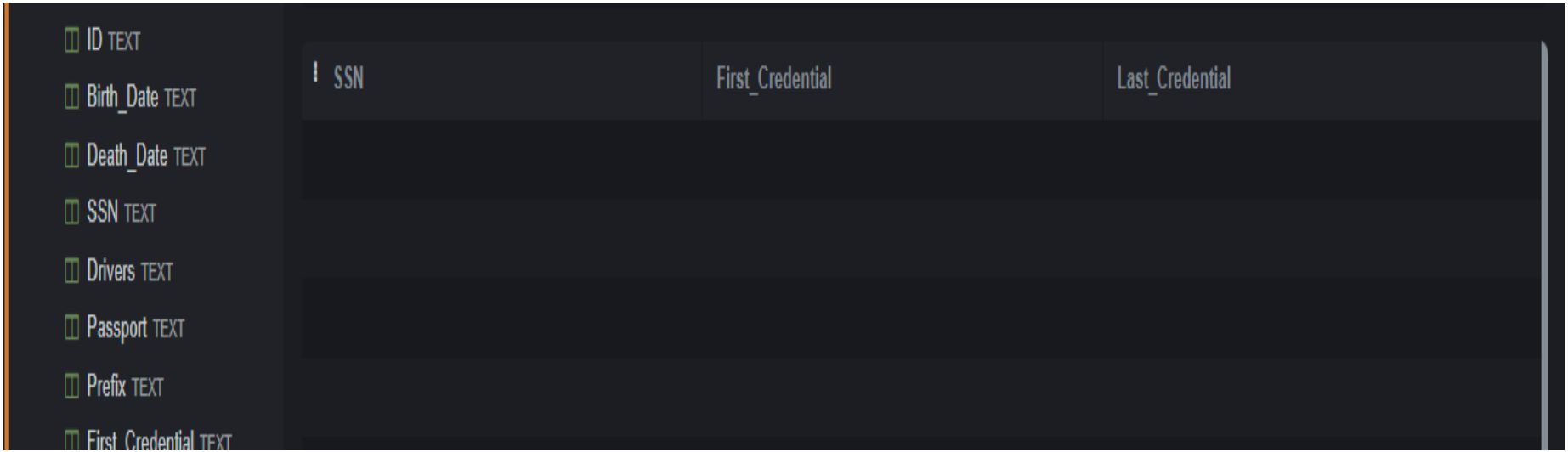
ID TEXT	SSN	First_Credential	Last_Credential
Birth_Date TEXT			
Death_Date TEXT	999-49-3323	Thi53	Bethalham
SSN TEXT			
Drivers TEXT			
Passport TEXT			
Prefix TEXT			

Delete 2 patients records based on their SSN numbers

```
DELETE FROM patients
WHERE
SSN IN ('999-69-5975' , '999-62-6022');
```

<u>Retrieve the results using below query</u>

```
select SSN, first_credential, last_credential from patients
where SSN = '999-69-5975' and SSN = '999-62-6022';
```



Fetch First, Last credentials of patient whose ID is available in both patients and observations data tables (Sub-Query)

```
SELECT
id, first_credential, last_credential
FROM
patients
WHERE
id IN (SELECT
id
FROM
observations
WHERE
id = 'b9c610cd-28a6-4636-ccb6-c7a0d2a4cb85')
ORDER BY first_credential , last_credential;
```

ObservationsCode T...
VitalSigns_Descriptio...
VitalSigns_Value TEXT
VitalSigns_Units TEXT
VitalSigns_ValueType...
patients

ID	First_Credential	Last_Credential
b9c610cd-28a6-4636-ccb6-c7a0d2a4cb85	Damon455	Langosh790