

# NoSQL Databases - Weekly 7 Final Project FINAL

---

## INFO 579 Week 7 Final Project Report

Course: INFO 579: SQL/NoSQL Databases for Data and Information Sciences

Module/Week: 7 - Week of August 18, 2025

### Topic: Week 7 Final Project Report

*NOTE: The assignment document must provide the following information. Up to 5 points may be deducted due to the lack of the below information.*

Student's Full Name: Matthew Qi Lan Thompson

Course Title: INFO 579: SQL/NoSQL Databases for Data and Information Sciences

Term name and year: Summer 2025

Submission Week: Week 7 Final Project Report

Instructor's Name: Dr. Nayem Rahman

Date of Submission: Aug 22, 2025

*NOTE: A 20% deduction of points will be applied for any late submission.*

---

## Week 2 Final Project Update 1: Total Points - 25

Week 2 Final Project Update 1:

1. Come up with a business plan for which you need to collect and store data
2. Come up with the data about your business. Example: like the one I gave you for the assignment.

**Requirements:** As part of the Final Project Update 1, submit a single-spaced one-page report:

1. A brief description of your project.
  2. The data you will use and the source of that data.
  3. Upload the data file (like the one I gave you to do the assignments).
- 

## Business Descriptive Report

1. Business Plan: Hospital Performance & Patient Outcome Analytics System

I am interested in working with data that allows me to assess the following:

- **patient outcomes**,
- **provider performance**, and
- **procedure effectiveness**

I thought about the kind of data I want to work with in the future and I decided to try healthcare-based data and would like to gain experiences with improving healthcare services overall.

---

## 2. Data Collection & Storage Plan

### Data Sources:

I'll be using open-sourced data from **Synthea** (<https://synthetichealth.github.io/synthea/>), which represents realistic hospital EHR data settings. The listed datasets I plan to use are shown below:

Data Tables	Brief Description	Objective
patients.csv	Basic demographic info	Identify patients, stratify risk by age/gender
encounters.csv	Visits per patient	Core fact table; basis for all events
observations.csv	Vitals, labs, metrics	Outcome indicators (e.g., BP, glucose)
conditions.csv	Diagnoses assigned	Track chronic illness and other diagnoses
procedures.csv	Surgeries/treatments	Evaluate procedure volume & quality
providers.csv	Doctors/facilities	Analyze workload, quality, profitability

With these data tables in csv, I'll be able to easily import them to MySQL and use SQL queries to obtain valuable insights about these imported data in a relational schema altogether.

---

## 3. Business Objectives (Driven by Data)

When working with these data using SQL queries, I propose some improvements for the healthcare service quality in a virtual simulation setting using the following objectives I can think of below:

## 1. Increase Profitability

I'll look at procedure outcomes and compare them to the costs. If a procedure has poorer quality but higher cost, that's something worth reviewing and possibly improving.

## 2. Improve Clinical Quality

I'll check the frequency of conditions and assess whether they're being well managed, based on the number of visits for the same condition.

## 3. Optimize Provider Utilization

I can look at providers and see how well they're performing, indicated by the number of encounters they receive under their care.

## 4. Reduce Readmissions

I can assess a patient's health risk based on how often they return for the same condition and how much time passes between each visit. If the time gap is short and the frequency is high, they could be flagged as higher risk, which can help streamline follow-up care.

## 5. Identify Care Strategic Expansion

I can observe the frequency of certain conditions. If some conditions show up very often, I could consider proposing improvements or expansions in that area of care, depending on other factors like treatment outcomes and how often patients return.

However, I may continue to add more business objectives or adjust them as I work with the Synthea data, since its size is enormous and it's designed to mimic a realistic healthcare setting. That gives me plenty of opportunities to practice working with data analytics from different angles beyond what I've proposed so far.

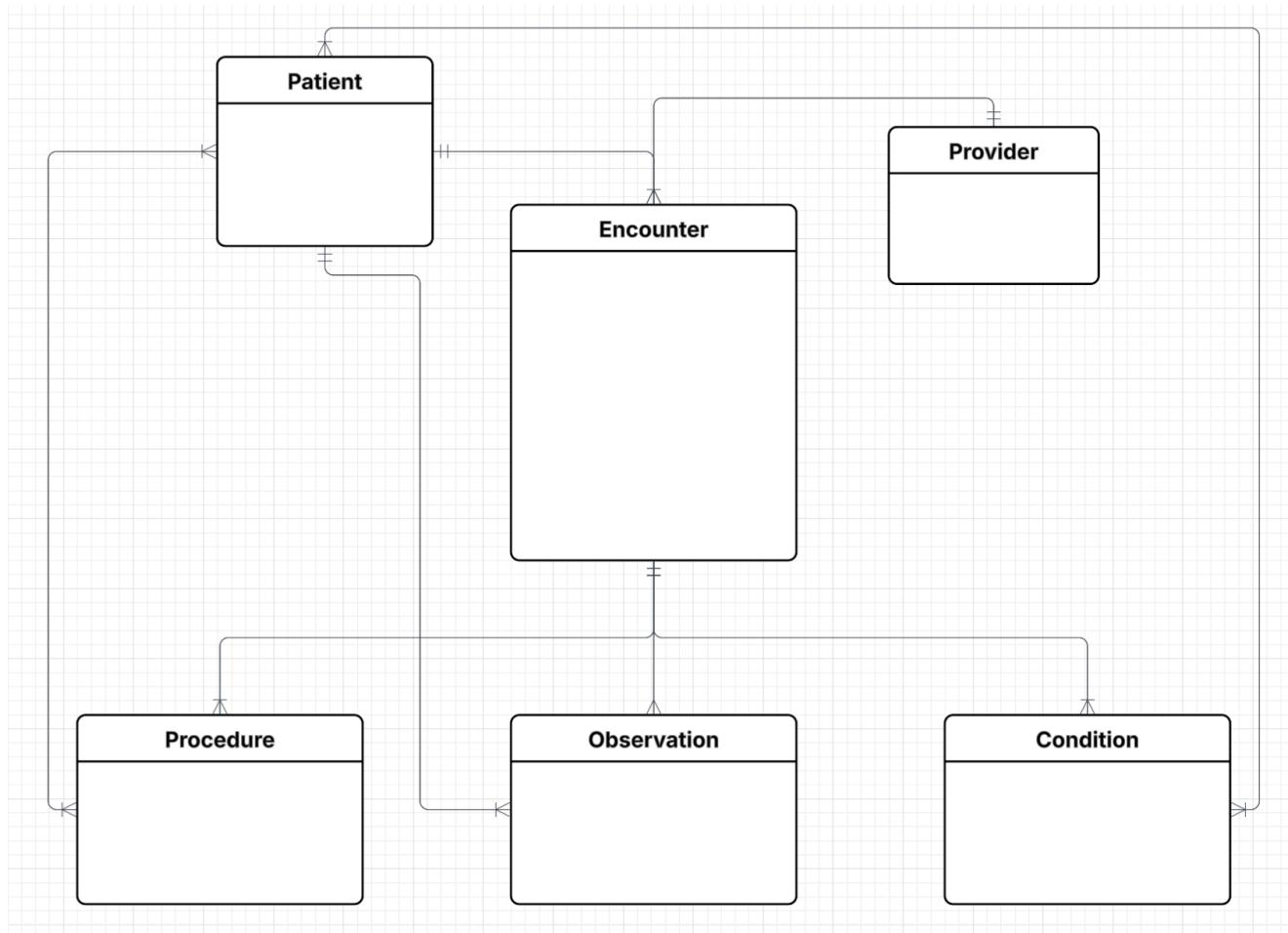
---

I'll upload the data files I plan to use (and mentioned in data collection & storage plan) along with this brief project description in PDF.

---

## Week 4 Final Project Update 2: Total Points - 25

3. Develop a Conceptual Model. Consider 4 or 5 entities. Make sure you have at least one many-to-many relationship. Explain with data why it's a many-to-many relationship.



Two entities have many-to-many relationships: Patient and Procedures & Patient and Condition.

```

7 #Patient # Condition      block comment should start with '# '
8 #how many unique conditions each patient has    block comment should start with '# '
9 unique_conditions_per_patient = conditions.groupby("PATIENT") ["CODE"].nunique()
10
11 #how many patients are linked to each condition   block comment should start with '# '
12 patients_per_condition = conditions.groupby("CODE") ["PATIENT"].nunique()
13
14 #Patient # Procedure    block comment should start with '# '
15 # unique procedures for each patient
16 unique_procedures_per_patient = procedures.groupby("PATIENT") ["CODE"].nunique()
17
18 #how many patients are linked to each procedure   block comment should start with '# '
19 patients_per_procedure = procedures.groupby("CODE") ["PATIENT"].nunique()
20

```

PROBLEMS 14    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    SPELL CHECKER 1

~/Doc/Academics/DS Masters Academics/SQL:NoSQL Databases/Assignments/Week

```

④ > /Users/matthewthompson/.pyenv/versions/3.12.4/bin/python "/Users/matthewthompson/Document
L Databases/Assignments/Weekly 4/relationship_verify.py"
Patient # Condition
Total unique patients: 1152
Patients with >1 condition: 1115
Total unique condition codes: 129
Conditions assigned to >1 patient: 120

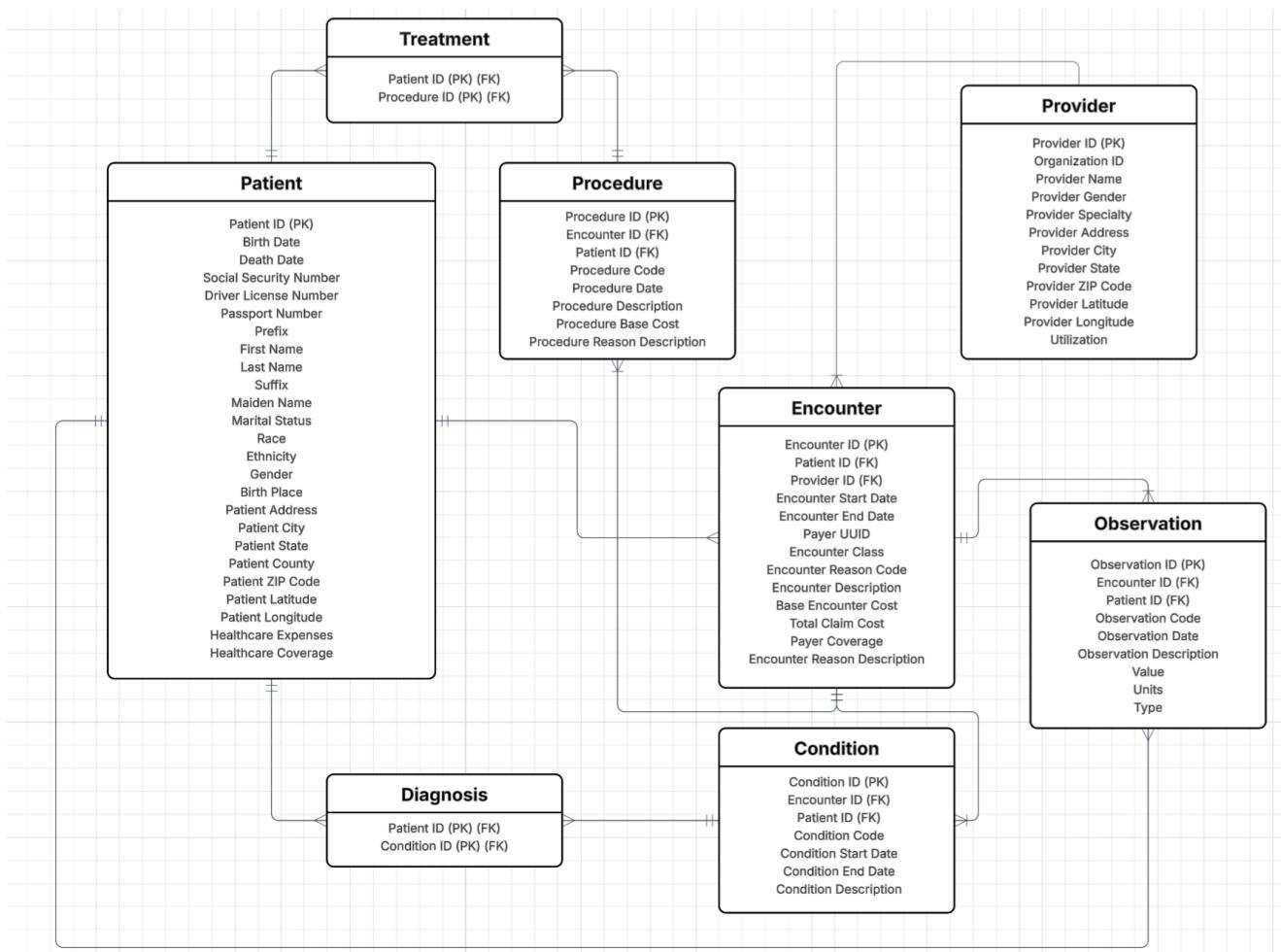
Patient # Procedure
Total unique patients: 1165
Patients with >1 procedure: 994
Total unique procedure codes: 137
Procedures assigned to >1 patient: 130

```

According to the result:

I checked for unique patients and conditions first to justify many-to-many relationships. Because, it shows that conditions are assigned to more than 1 patients, while each patient also has more than one condition. Same is said for the patient & procedures.

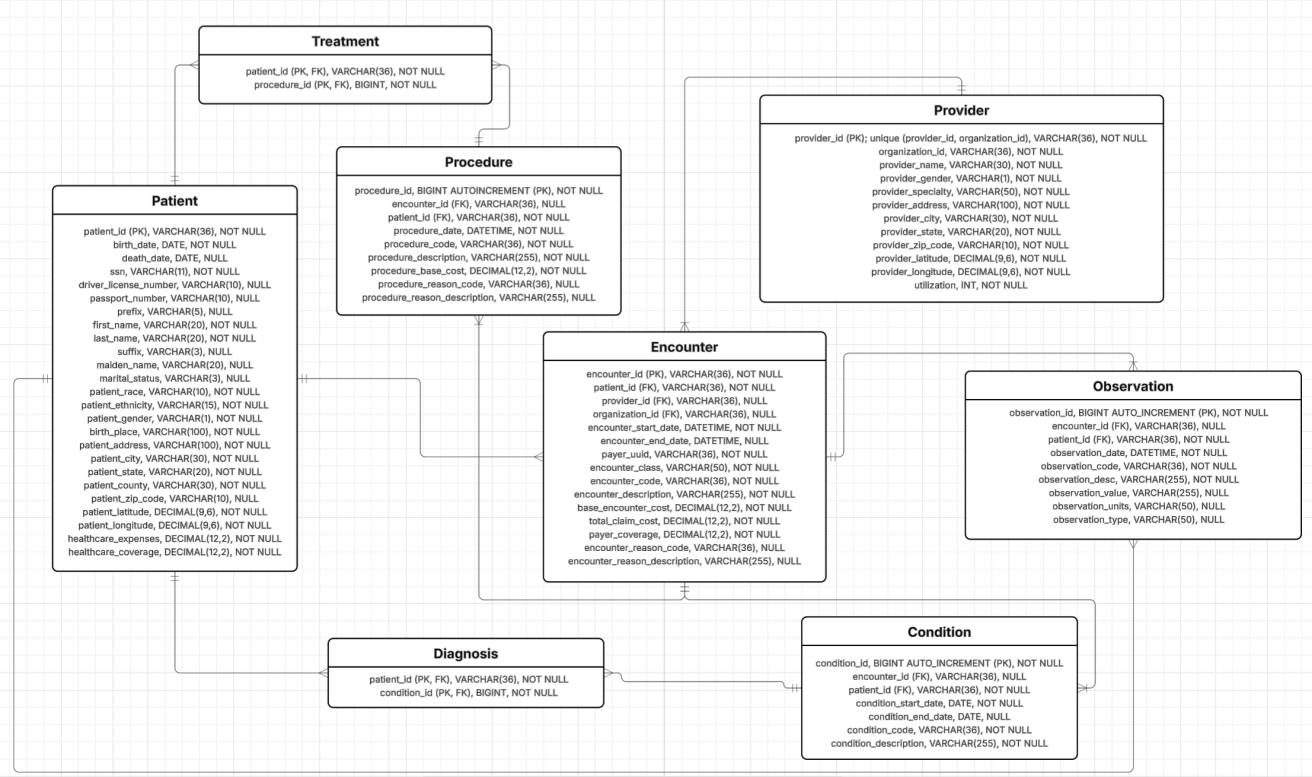
4. Develop a Logical Model using the Conceptual Model. Make sure you come up with a junction entity to resolve the many-to-many relationship.



## Week 7 Final Project Report: Total Points - 100

### Physical Model

5. Develop the physical model based on the Logical Model



6. Create tables using a database system. Insert data into the database tables. You must provide the DDL (CREATE TABLE statements), INSERT statements, and SELECT statements.

Details: Create the tables that you have come up with (the table must be based on the Physical Model).

- (a) Columns, Primary Key (PK), Data Type, and length, and NULL/NOT NULL need to be implemented, per the Physical Model.
- (b) Show the table definition (DDL) that you implemented (not in a graphical view).
- (c) Insert the complete set of data that you have come up with and show the insert statements used.

## Created Tables

### Patient

**IMPORTANT NOTE:** Instead of INSERTING one at a time, I've found a way to import csv files directly into SQL after creating the tables. So, I can bulk insert every single rows and columns by using **LOAD DATA INFILE**. Thankfully, it went well after a few tweaks.

```
CREATE TABLE patient (
```

patient_id	VARCHAR(36)	NOT NULL PRI
birth_date	DATE	NOT NULL,
death_date	DATE	NULL,
ssn	VARCHAR(11)	NOT NULL,
driver_license_number	VARCHAR(10)	NULL,
passport_number	VARCHAR(10)	NULL,
prefix	VARCHAR(5)	NULL,
first_name	VARCHAR(20)	NOT NULL,
last_name	VARCHAR(20)	NOT NULL,
suffix	VARCHAR(3)	NULL,
maiden_name	VARCHAR(20)	NULL,
marital_status	VARCHAR(3)	NULL,
patient_race	VARCHAR(10)	NOT NULL,
patient_ethnicity	VARCHAR(15)	NOT NULL,
patient_gender	VARCHAR(1)	NOT NULL,
birth_place	VARCHAR(100)	NOT NULL,
patient_address	VARCHAR(100)	NOT NULL,
patient_city	VARCHAR(30)	NOT NULL,
patient_state	VARCHAR(20)	NOT NULL,
patient_county	VARCHAR(30)	NOT NULL,
patient_zip_code	VARCHAR(10)	NULL,
patient_latitude	DECIMAL(9,6)	NOT NULL,
patient_longitude	DECIMAL(9,6)	NOT NULL,
healthcare_expenses	DECIMAL(12,2)	NOT NULL,
healthcare_coverage	DECIMAL(12,2)	NOT NULL

```
);
```

```

LOAD DATA INFILE '/var/lib/mysql-files/patients.csv'
INTO TABLE patient
FIELDS TERMINATED BY ',' ENCLOSED BY ""
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(@id, @birthdate, @deathdate, @ssn, @drivers, @passport, @prefix, @first, @last, @suffix,
@maiden, @marital, @race, @ethnicity, @gender, @birthplace, @address, @city, @state, @county,
@zip, @lat, @lon, @expenses, @coverage)
SET
patient_id      = @id,
birth_date       = STR_TO_DATE(@birthdate, '%Y-%m-%d'),
death_date = NULLIF(STR_TO_DATE(NULLIF(@deathdate, ''), '%Y-%m-%d'), NULL), "deathdate": Un
ssn              = @ssn,
driver_license_number = @drivers,
passport_number   = @passport,
prefix            = @prefix,
first_name        = @first,
last_name         = @last,
suffix            = @suffix,
maiden_name       = @maiden,
marital_status    = @marital,
patient_race      = @race,
patient_ethnicity = @ethnicity,
patient_gender    = @gender,
birth_place        = @birthplace,
patient_address   = @address,
patient_city       = @city,
patient_state      = @state,
patient_county    = @county,
patient_zip_code  = @zip,
patient_latitude   = @lat,
patient_longitude = @lon,
healthcare_expenses = @expenses,
healthcare_coverage = @coverage;

```

	SELECT * FROM patient LIMIT 100										
	patient_id	birth_date	death_date	ssn	driver_license_number	passport_number	prefix	first_name	last_name	...	
1	00185faa-2760-4218-9bf5-c	2003-11-18	(NULL)	999-50-8531	S99964760			Eusebio566	Wyman9		
2	0042862c-9889-4a2e-b782-	2009-11-26	(NULL)	999-20-4613				Dewitt635	Feest10:		
3	0047123f-12e7-486c-82df-t	1960-01-20	(NULL)	999-92-5264	S99959789	X2594715X	Mr.	Jordon466	Harber2!		
4	010d4a3a-2316-45ed-ae15-	1998-05-31	(NULL)	999-21-2604	S99974819	X34193837X	Mr.	Patrick786	Hettinge		
5	01207ecd-9dff-4754-8887-4	2019-05-15	(NULL)	999-81-4349				Karyn217	Mueller8		

## Provider

```
✓ CREATE TABLE provider (
    provider_id      VARCHAR(36) NOT NULL PRIMARY KEY,
    organization_id  VARCHAR(36) NOT NULL,
    provider_name    VARCHAR(30) NOT NULL,
    provider_gender   VARCHAR(1)  NOT NULL,
    provider_specialty VARCHAR(50) NOT NULL,
    provider_address  VARCHAR(100) NOT NULL,
    provider_city     VARCHAR(30) NOT NULL,
    provider_state    VARCHAR(20) NOT NULL,
    provider_zip_code VARCHAR(10) NOT NULL,
    provider_latitude DECIMAL(9,6) NOT NULL,
    provider_longitude DECIMAL(9,6) NOT NULL,
    utilization       INT NOT NULL
);

```

▷ Run

```
ALTER TABLE provider
ADD UNIQUE KEY uq_provider_org (provider_id, organization_id);
```

*Note: I created the unique key, because each provider\_id is unique while there are some duplicates of organization\_id. By making an unique key with the paired provider\_id and organization\_id, I would be able to use both provider\_id and organization\_id as foreign keys.*

```
▷ Run | ▷ Select
LOAD DATA INFILE '/var/lib/mysql-files/providers.csv'
INTO TABLE provider
FIELDS TERMINATED BY ',' ENCLOSED BY ''
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(@id, @org, @name, @gender, @speciality, @address, @city, @state, @zip, @lat, @lon, @util)
SET
    provider_id      = @id,
    organization_id  = @org,
    provider_name    = @name,
    provider_gender   = LEFT(@gender,1),          -- enforce single char
    provider_specialty = @speciality,           -- CSV header is "SPECIALITY" "special."
    provider_address  = @address,
    provider_city     = @city,
    provider_state    = @state,
    provider_zip_code = @zip,
    provider_latitude = @lat,
    provider_longitude = @lon,
    utilization       = @util;
```

## Encounter

```

CREATE TABLE encounter (
    encounter_id          VARCHAR(36) NOT NULL,
    patient_id            VARCHAR(36) NOT NULL,
    provider_id           VARCHAR(36) NULL,
    organization_id       VARCHAR(36) NULL,
    encounter_start_date  DATETIME      NOT NULL,
    encounter_end_date    DATETIME      NULL,
    payer_uuid             VARCHAR(36) NULL,
    encounter_class        VARCHAR(50) NOT NULL,
    encounter_code         VARCHAR(36) NOT NULL,
    encounter_description  VARCHAR(255) NOT NULL,
    base_encounter_cost   DECIMAL(12,2) NOT NULL,
    total_claim_cost       DECIMAL(12,2) NOT NULL,
    payer_coverage         DECIMAL(12,2) NOT NULL,
    encounter_reason_code  VARCHAR(36) NULL,
    encounter_reason_desc  VARCHAR(255) NULL,
    CONSTRAINT pk_encounter PRIMARY KEY (encounter_id),
    CONSTRAINT fk_encounter_patient
        FOREIGN KEY (patient_id) REFERENCES patient(patient_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    CONSTRAINT fk_encounter_provider_org
        FOREIGN KEY (provider_id, organization_id)
        REFERENCES provider(provider_id, organization_id)
        ON DELETE SET NULL ON UPDATE CASCADE
);

```

```

▷ Run | Select
LOAD DATA INFILE '/var/lib/mysql-files/encounters.csv'
INTO TABLE encounter
FIELDS TERMINATED BY ',' ENCLOSED BY ""
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(@id, @start, @stop, @patient, @org, @provider, @payer,
@class, @code, @desc, @base_cost, @total_cost, @coverage, @reason_code, @reason_desc)
SET
    encounter_id      = @id,
    patient_id       = @patient,
    provider_id      = NULLIF(@provider,''),
    organization_id  = NULLIF(@org,''),
    encounter_start_date = STR_TO_DATE(REPLACE(REPLACE(@start,'T',' '), 'Z',''), '%Y-%m-%d %H:%i:%s'),
    encounter_end_date = STR_TO_DATE(REPLACE(REPLACE(NULLIF(@stop,''), 'T',' '), 'Z',''), '%Y-%m-%d %H:%i:%s'),
    payer_uuid        = NULLIF(@payer,''),
    encounter_class   = @class,
    encounter_code    = @code,
    encounter_description = @desc,
    base_encounter_cost = @base_cost,
    total_claim_cost  = @total_cost,
    payer_coverage    = @coverage,
    encounter_reason_code = NULLIF(@reason_code,''),
    encounter_reason_description = NULLIF(@reason_desc,'');

```

## Observation

```

▷ Run | Select
CREATE TABLE observation (
    observation_id      BIGINT AUTO_INCREMENT PRIMARY KEY,
    observation_date    DATETIME           NOT NULL,
    patient_id          VARCHAR(36)        NOT NULL,
    encounter_id         VARCHAR(36)        NULL,
    observation_code    VARCHAR(36)        NOT NULL,
    observation_desc    VARCHAR(255)       NOT NULL,
    observation_value   VARCHAR(255)       NULL,
    observation_units   VARCHAR(50)        NULL,
    observation_type    VARCHAR(50)        NULL,
    FOREIGN KEY (patient_id) REFERENCES patient(patient_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (encounter_id) REFERENCES encounter(encounter_id)
        ON DELETE SET NULL ON UPDATE CASCADE
);

```

```

CREATE TABLE st_observation (
    `DATE`        VARCHAR(30),
    `PATIENT`     VARCHAR(36),
    `ENCOUNTER`   VARCHAR(36),
    `CODE`        VARCHAR(36),
    `DESCRIPTION` VARCHAR(255),
    `VALUE`        VARCHAR(255),
    `UNITS`        VARCHAR(50),
    `TYPE`         VARCHAR(50)
);

LOAD DATA INFILE '/var/lib/mysql-files/observations.csv'
INTO TABLE st_observation
FIELDS TERMINATED BY ',' ENCLOSED BY ''
LINES TERMINATED BY '\n'
IGNORE 1 LINES;

INSERT INTO observation
(observation_date, patient_id, encounter_id,
observation_code, observation_desc, observation_value,
observation_units, observation_type)
SELECT
STR_TO_DATE(REPLACE(REPLACE(s.`DATE`,'T',' '), 'Z', ''), '%Y-%m-%d %H:%i:%s') AS observation_date,
s.`PATIENT` AS patient_id,
CASE WHEN e.encounter_id IS NULL THEN NULL ELSE s.`ENCOUNTER` END AS encounter_id,
s.`CODE` AS observation_code,
s.`DESCRIPTION` AS observation_desc,
NULLIF(s.`VALUE`, '') AS observation_value,
NULLIF(s.`UNITS`, '') AS observation_units,
NULLIF(s.`TYPE`, '') AS observation_type
FROM st_observation s
LEFT JOIN encounter e ON e.encounter_id = s.`ENCOUNTER`;

DROP TABLE st_observation;

```

Note: I created this table `st_observation`, because it allows me to load the whole data into that staging table. Then, I `INSERT` and `SELECT` staged observation table into `observation` (intended) table WHILE `LEFT JOIN` on `encounter_id` from `encounter` table. Any found mismatched `encounter_id` will be inserted as `NULL` into `observation` table. Then I dropped `st observation`, because it is not needed after. Apparently, I did a bit of investigation and there are around 10.13% `encounter_id` in `observation` csv file not found in `encounter` csv file. I decided to keep these `NULL` values for future insights later if interested.

```

239 --verifying if this works
240 SELECT
241     COUNT(*) AS total_rows,
242     SUM(encounter_id IS NULL) AS null_encounter_rows,
243     ROUND(SUM(encounter_id IS NULL) * 100.0 / COUNT(*), 2) AS pct_null_encounter
244 FROM observation; 86ms

observation X
Search Results Export Cost: 94ms 1
total_rows null_encounter_rows pct_null_encounter
299697 30363 10.13

```

## Condition

```
▷Run | ▷Select
CREATE TABLE medical_condition (
    condition_id          BIGINT AUTO_INCREMENT PRIMARY KEY,
    condition_start_date  DATE        NOT NULL,
    condition_end_date    DATE        NULL,
    patient_id            VARCHAR(36) NOT NULL,
    encounter_id          VARCHAR(36) NULL,
    condition_code         VARCHAR(36) NOT NULL,
    condition_description VARCHAR(255) NOT NULL,
    FOREIGN KEY (patient_id) REFERENCES patient(patient_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (encounter_id) REFERENCES encounter(encounter_id)
        ON DELETE SET NULL ON UPDATE CASCADE
);
```

```
▷Run | ▷Select
LOAD DATA INFILE '/var/lib/mysql-files/conditions.csv'
INTO TABLE medical_condition
FIELDS TERMINATED BY ',' ENCLOSED BY ""
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(@start, @stop, @patient, @encounter, @code, @desc)
SET
    condition_start_date = STR_TO_DATE(@start, '%Y-%m-%d'),
    condition_end_date   = STR_TO_DATE(NULLIF(@stop,''), '%Y-%m-%d'),
    patient_id           = @patient,
    encounter_id          = NULLIF(@encounter,''),
    condition_code        = @code,
    condition_description = @desc;
```

## Procedure

```

CREATE TABLE procedures (
    procedure_id          BIGINT AUTO_INCREMENT PRIMARY KEY,
    procedure_date        DATETIME      NOT NULL,
    patient_id            VARCHAR(36)  NOT NULL,
    encounter_id          VARCHAR(36)  NULL,
    procedure_code        VARCHAR(36)  NOT NULL,
    procedure_description VARCHAR(255) NOT NULL,
    procedure_base_cost   DECIMAL(12,2) NOT NULL,
    procedure_reason_code VARCHAR(36)  NULL,
    procedure_reason_desc VARCHAR(255) NULL,
    FOREIGN KEY (patient_id) REFERENCES patient(patient_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (encounter_id) REFERENCES encounter(encounter_id)
        ON DELETE SET NULL ON UPDATE CASCADE
);

```

```

▷ Run | ▷Select
LOAD DATA INFILE '/var/lib/mysql-files/procedures.csv'
INTO TABLE procedures
FIELDS TERMINATED BY ',' ENCLOSED BY ""
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(@date, @patient, @encounter, @code, @desc, @base_cost, @reason_code, @reason_desc)
SET
procedure_date      = STR_TO_DATE(REPLACE(REPLACE(@date,'T',' '),'Z',''), '%Y-%m-%d %H:%i:%s'),
patient_id          = @patient,
encounter_id        = NULLIF(@encounter,''),
procedure_code      = @code,
procedure_description = @desc,
procedure_base_cost = CAST(NULLIF(@base_cost,'') AS DECIMAL(12,2)),
procedure_reason_code = NULLIF(@reason_code,''),
procedure_reason_desc = NULLIF(@reason_desc,'');

```

## Junction Tables

```

▷ Run | ▷Select
CREATE TABLE diagnosis (
    patient_id  VARCHAR(36) NOT NULL,
    condition_id BIGINT      NOT NULL,
    PRIMARY KEY (patient_id, condition_id),
    FOREIGN KEY (patient_id) REFERENCES patient(patient_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (condition_id) REFERENCES medical_condition(condition_id)
        ON DELETE CASCADE ON UPDATE CASCADE
);

```

```

CREATE TABLE treatment (
    patient_id  VARCHAR(36) NOT NULL,
    procedure_id BIGINT      NOT NULL,
    PRIMARY KEY (patient_id, procedure_id),
    FOREIGN KEY (patient_id) REFERENCES patient(patient_id)
        ON DELETE RESTRICT ON UPDATE CASCADE,
    FOREIGN KEY (procedure_id) REFERENCES procedures(procedure_id)
        ON DELETE CASCADE ON UPDATE CASCADE
);

```

## Retrieving Data with SQL Queries

### SQL Queries for Business Questions (Mostly)

7. Create a variety of SQL queries to retrieve data from one or many tables:

#### 7.1 Provider utilization (encounters per provider & specialty)

```

1  CREATE TABLE rpt_provider_utilization AS
2  SELECT pr.provider_id,
3  pr.organization_id,
4  pr.provider_name,
5  pr.provider_specialty,
6  COUNT(e.encounter_id) AS n_encounters
7  FROM provider pr
8  LEFT JOIN encounter e
9  ON e.provider_id = pr.provider_id
10 AND e.organization_id = pr.organization_id
11 GROUP BY pr.provider_id, pr.organization_id, pr.provider_name,
12   pr.provider_specialty;
13
14 SELECT * FROM rpt_provider_utilization
15 ORDER BY n_encounters DESC, provider_id;

```

	provider_id	organization_id	provider_name	provider_specialty	n_encounters
1	8f9aea5b-fd01-37c0-8931-18b6d64bdae6	be4c63f3-8d38-3fa9-a183-62045b5c85f6	Gaynell126 Streich926	GENERAL PRACTICE	3217
2	793c18cd-8269-387d-b998-c135459e4248	d311e70d-86e7-3c03-b115-53892bcf7ef1	Gertrudis163 Schaden604	GENERAL PRACTICE	1972
3	f4eb93d1-9187-3cfb-83a4-6d9cd77f7df6	d692e283-0833-3201-8e55-4f868a9c0736	Vern731 Powlowski563	GENERAL PRACTICE	1658
4	e6283e46-fd81-3611-9459-0edb1c3da357	e002090d-4e92-300e-b41e-7d1f21dee4c6	Jeanmarie510 Beatty507	GENERAL PRACTICE	1028
5	4b04cd2f-3f27-35bc-8069-f4ca6339529f	3d10019f-c88e-3de5-9916-6107b9c0263d	Maile198 Frami345	GENERAL PRACTICE	955
6	40d7ff6c-5040-39aa-a876-19f1ccc6740c	3bd5eda0-16da-3ba5-8500-4dfd6ae118b8	Luke971 Rath779	GENERAL PRACTICE	859
7	0a8a9359-7b33-3256-a068-b5a7d18ebe4b	465de31f-3098-365c-af70-48a071e1f5aa	Keri25 Schmidt332	GENERAL PRACTICE	857
8	7bd4e666-a82d-3ad1-bc7c-b49eb726577b	24cb4eab-6166-3530-bddc-a5a8a14a4fc1	Lonna614 Dietrich576	GENERAL PRACTICE	824
9	af01a385-31d3-3c77-8fdb-2867fe88df2f	5d4b9df1-93ae-3bc9-b680-03249990e558	Garth972 Wyman904	GENERAL PRACTICE	783
10	0359f968-d1a6-30eb-b1cc-e6cc0b4d3513	5e765f2b-e908-3888-9fc7-df2cb87beb58	Gaynell126 Streich926	GENERAL PRACTICE	763

#### 7.2 Inpatient length of stay (LOS) per provider

```

1  CREATE TABLE rpt_inpatient_los_provider AS
2  SELECT pr.provider_id,
3  pr.organization_id,
4  pr.provider_name,
5
6  ROUND(AVG(TIMESTAMPDIFF(HOUR, e.encounter_start_date,
e.encounter_end_date))/24, 2) AS avg_los_days,
7  COUNT(*) AS n_inpatient_encounters
8  FROM encounter e
9  JOIN provider pr
10 ON pr.provider_id = e.provider_id
11 AND pr.organization_id = e.organization_id
12 WHERE LOWER(e.encounter_class) = 'inpatient'
13 AND e.encounter_end_date IS NOT NULL
14 GROUP BY pr.provider_id, pr.organization_id, pr.provider_name;
15
16 SELECT * FROM rpt_inpatient_los_provider
17 ORDER BY avg_los_days DESC, provider_id;

```

Screenshot of a database query results table titled "rpt\_inpatient\_los\_provider". The table has 7 columns: provider\_id, organization\_id, provider\_name, avg\_los\_days, and n\_inpatient\_encounters. The results show 72 rows of data.

	* provider_id varchar(36)	* organization_id varchar(36)	* provider_name varchar(30)	avg_los_days decimal(23,2)	* n_inpatient_encounters bigint
1	1045937e-4c7c-3dee-81de- f71ce1a9-cca7-3295-9a1e-c	Jc393 Borer986		578.66	35
2	40d7ff6c-5040-39aa-a876-1	3bd5eda0-16da-3ba5-8500-	Luke971 Rath779	232.84	46
3	77a7881d-6dd5-32e1-9e18- ef6ab57c-ed94-3dbe-9861-8	Philip440 McCullough561		8.38	30
4	1595224e-648a-3615-aae6- 161a799c-4894-36f6-8ca7-1	Quincy153 Heathcote539		2.54	2
5	8f9aea5b-fd01-37c0-8931-1	be4c63f3-8d38-3fa9-a183-6	Gaynell126 Streich926	2.50	2
6	b6c72690-131c-355f-91d0- 8881e3ee-e915-31b3-867f-1	Lynwood354 Monahan736		2.42	52
7	1b75d7a2-3964-340e-b025- 7fb56531-86bd-3e4a-8619-	Jeramy610 Littel644		2.04	6
8	8ee28b4a-9018-3065-9f6b-1	331f4c11-d298-308b-aaa1-c	Malinda718 Cassin499	2.04	1
9	ea53dda3-3bf7-30c9-804b-e	a9f20dc1-5147-3789-bcef-b	Judith460 Johns824	2.02	2
10	69e585d1-1d1f-39ed-a9e8-1	1bd44727-a8e3-322d-9c14-	Suzette512 Monahan736	2.00	1

### 7.3 Top patients by total claim cost

```

1  SELECT p.patient_id,
2  p.first_name, p.last_name,
3  SUM(e.total_claim_cost) AS total_claim_cost
4  FROM patient p
5  JOIN encounter e ON e.patient_id = p.patient_id
6  GROUP BY p.patient_id, p.first_name, p.last_name
7  ORDER BY total_claim_cost DESC, p.patient_id
8  LIMIT 10;

```

Result(RO) ×

Search Results Export Cost: 109ms 1 2 3 4 ... 118 > Total 1171

	patient_id	first_name	last_name	total_claim_cost
	varchar	varchar	varchar	decimal
>	cae10920-f977-48b4-a0d3-4	Logan497	Brekke496	259094.96
>	19d2cfb8-439b-454a-b47e-4	Walker122	Kuhic920	207430.96
>	3f336702-bf73-4fc8-bd59-3	Sanford861	Fritsch593	198648.08
>	3acf9313-1874-4dff-ab2a-3	Earle679	Frami345	106686.16
>	cecb7ece-fb70-4a7f-b51d-6	Isiah14	Fritsch593	58380.32
>	3dd2dd29-7cd0-48f7-b859-	Ira784	Tremblay80	49726.60
>	714b9c18-783d-4f52-aa64-c	Noble66	Sipes176	48435.00
>	2c71dd97-7085-416a-aa07-	Samatha845	Mueller846	47401.72
>	6ec18ddf-e9ee-421a-9033-4	Stanley702	Douglas31	47401.72
>	137acc1b-dbca-473e-84bb-1	Boyd728	Beahan375	44689.36

## 7.4 Procedure volume & average base cost

```

1  CREATE TABLE rpt_procedure_costs AS
2  SELECT prc.procedure_code,
3  prc.procedure_description,
4  COUNT(*) AS n_procedures,
5  ROUND(AVG(prc.procedure_base_cost), 2) AS avg_base_cost
6  FROM procedures prc
7  GROUP BY prc.procedure_code, prc.procedure_description;
8
9  SELECT * FROM rpt_procedure_costs
10 ORDER BY n_procedures DESC, procedure_code
11 LIMIT 25;

```

rpt\_procedure\_costs ×

Search Results Export Cost: 18ms 1 2 3 4 ... 6 > Total 144

	* procedure_code	* procedure_description	* n_procedures	avg_base_cost
	varchar(36)	varchar(255)	bigint	decimal(13,2)
>	430193006	Medication Reconciliation (procedure)	5632	603.26
>	265764009	Renal dialysis (procedure)	3389	516.65
>	225158009	Auscultation of the fetal heart	2705	6371.99
>	274804006	Evaluation of uterine fundal height	2705	6302.03
>	180256009	Subcutaneous immunotherapy	1497	14393.69
>	76601001	Intramuscular injection	1324	3088.66
>	180325003	Electrical cardioversion	981	31721.34
>	703423002	Combined chemotherapy and radiation therapy (procedure)	655	14294.17
>	104091002	Hemoglobin / Hematocrit / Platelet count	602	2400.24
>	73761001	Colonoscopy	527	13877.49

## 7.5 Post-procedure follow-up within 14 days (rate)

```

1  CREATE TABLE rpt_followup_14d_by_proc_code AS
2  WITH proc_flags AS (
3  SELECT prc.procedure_id,
4  prc.procedure_code,

```

```

5   prc.procedure_description,
6   CASE
7     WHEN EXISTS (
8       SELECT 1
9       FROM observation o
10      WHERE o.patient_id = prc.patient_id
11      AND o.observation_date > prc.procedure_date
12      AND o.observation_date <= DATE_ADD(prc.procedure_date, INTERVAL 14
13        DAY)
14    ) THEN 1 ELSE 0
15  END AS has_followup_14d
16  FROM procedures prc
17  )
18  SELECT procedure_code,
19    procedure_description,
20    COUNT(*) AS n_procedures,
21    SUM(has_followup_14d) AS n_with_followup_14d,
22    ROUND(100 * SUM(has_followup_14d) / NULLIF(COUNT(*),0), 2) AS
23    pct_with_followup_14d
24  FROM proc_flags
25  GROUP BY procedure_code, procedure_description;
26
27  SELECT * FROM rpt_followup_14d_by_proc_code
28  ORDER BY pct_with_followup_14d ASC, n_procedures DESC
29  LIMIT 25;

```

Screenshot of a database query results table titled "rpt\_followup\_14d\_by\_proc\_code". The table displays 144 rows of data with the following columns:

procedure_code	procedure_description	n_procedures	n_with_followup_14d	pct_with_followup_14d
180207008	Intravenous blood transfusion of packed cells (procedure)	6	6	100.00
112001000119100	positive screening for PHQ-9	1	1	100.00
173160006	Diagnostic fiberoptic bronchoscopy (procedure)	2	2	100.00
167995008	Sputum microscopy (procedure)	2	2	100.00
112001000119100	positive screening for depression on phq9	5	5	100.00
265764009	Renal dialysis (procedure)	3389	3386	99.91
241055006	Mammogram - symptomatic (procedure)	6	5	83.33
80146002	Appendectomy	9	6	66.67
432231006	Fine needle aspiration biopsy of lung (procedure)	5	3	60.00
69031006	Excision of breast tissue (procedure)	5	3	60.00

## 7.6 High-risk signal: frequent emergency users (>=3 ER visits)

```

1  CREATE TABLE rpt_er_frequenters AS
2  SELECT
3    p.patient_id,
4    p.first_name,
5    p.last_name,
6    COUNT(*) AS er_visits
7  FROM encounter e
8  JOIN patient p

```

```

9   ON p.patient_id = e.patient_id
10  WHERE LOWER(e.encounter_class) = 'emergency'
11  GROUP BY p.patient_id, p.first_name, p.last_name
12  HAVING COUNT(*) >= 3;
13
14  SELECT *
15  FROM rpt_er_frequenter
16  ORDER BY er_visits DESC, last_name, first_name, patient_id;

```

Screenshot of a database query results table titled "rpt\_followup\_14d\_by\_proc\_code". The table has columns: procedure\_code, procedure\_description, n\_procedures, n\_with\_followup\_14d, and pct\_with\_followup\_14d. The data shows various medical procedures and their counts, along with the number and percentage of patients who had a follow-up appointment within 14 days.

	procedure_code	procedure_description	n_procedures	n_with_followup_14d	pct_with_followup_14d
1	180207008	Intravenous blood transfusion of packed cells (procedure)	6	6	100.00
2	112001000119100	positive screening for PHQ-9	1	1	100.00
3	173160006	Diagnostic fiberoptic bronchoscopy (procedure)	2	2	100.00
4	167995008	Sputum microscopy (procedure)	2	2	100.00
5	112001000119100	positive screening for depression on phq9	5	5	100.00
6	265764009	Renal dialysis (procedure)	3389	3386	99.91
7	241055006	Mammogram - symptomatic (procedure)	6	5	83.33
8	80146002	Appendectomy	9	6	66.67
9	432231006	Fine needle aspiration biopsy of lung (procedure)	5	3	60.00
10	69031006	Excision of breast tissue (procedure)	5	3	60.00

## Each Retrieved Tables

8. Retrieve the data from each table by using the `SELECT *` statement and order by PK column(s).
- Show the output. Make sure you show the print screen of the complete set of rows and columns. The rows must be ordered by the PK column(s).

```

1  SELECT * FROM patient ORDER BY patient_id;
2
3  SELECT * FROM provider ORDER BY provider_id;
4
5  SELECT * FROM encounter ORDER BY encounter_id;
6
7  SELECT * FROM observation ORDER BY observation_id;
8
9  SELECT * FROM medical_condition ORDER BY condition_id;
10
11 SELECT * FROM procedures ORDER BY procedure_id;
12
13 SELECT * FROM diagnosis ORDER BY patient_id, condition_id;
14
15 SELECT * FROM treatment ORDER BY patient_id, procedure_id;

```

## Patient Table

The screenshot shows a MySQL Workbench interface with a query editor at the top containing the SQL command:

```
SELECT * FROM patient ORDER BY patient_id; 10ms
```

The results grid displays 1171 rows of data from the patient table, with columns including patient\_id, birth\_date, death\_date, ssn, driver\_license\_number, passport\_number, prefix, first\_name, last\_name, and suffix.

**patient\_columns**

patient\_id , birth\_date , death\_date , ssn ,  
 driver\_license\_number , passport\_number , prefix ,  
 first\_name , last\_name , suffix , maiden\_name ,  
 marital\_status , patient\_race , patient\_ethnicity ,  
 patient\_gender , birth\_place , patient\_address ,  
 patient\_city , patient\_state , patient\_county ,  
 patient\_zip\_code , patient\_latitude , patient\_longitude ,  
 healthcare\_expenses , healthcare\_coverage

**Provider Table**

The screenshot shows a MySQL Workbench interface with a query editor at the top containing the SQL command:

```
SELECT * FROM provider ORDER BY provider_id; 17ms
```

The results grid displays 5855 rows of data from the provider table, with columns including provider\_id, organization\_id, provider\_name, provider\_gender, provider\_specialty, provider\_address, provider\_city, and provider\_state.

<b>provider_columns</b>	provider_id , organization_id , provider_name , provider_gender , provider_specialty , provider_address , provider_city , provider_state , provider_zip_code , provider_latitude , provider_longitude , utilization
-------------------------	---

## Encounter Table

Properties										Data		Log		ER		Monitor																																																																																																																																																							
<code>SELECT * FROM encounter ORDER BY encounter_id</code>																																																																																																																																																																							
307 rows selected.																																																																																																																																																																							
Cost: 28ms																																																																																																																																																																							
1 2 3 4 ... 534 > Total 53346																																																																																																																																																																							
<table border="1"> <thead> <tr> <th>encounter_id</th> <th>patient_id</th> <th>provider_id</th> <th>organization_id</th> <th>encounter_start_date</th> <th>encounter_end_date</th> <th>payer_uuid</th> <th>encounter_class</th> </tr> </thead> <tbody> <tr><td>0000d0b7-937c-498a-9da4-714b9c18-783d-4f52-aa64-a3a65bdb3-12b2-3247-8e8a-5259a506-b80b-3ed5-9c30-1994-09-07 04:48:12</td><td></td><td></td><td></td><td>1994-09-07 05:03:12</td><td></td><td>d47b3510-2895-3b70-9897- wellness</td><td></td></tr> <tr><td>000186d2-1316-4b58-be65-8be62b06-5994-47fe-aad8-56c64467-dd8a-36ca-91c5-8ad64ecf-c817-3753-bee7-c 2014-12-03 13:12:11</td><td></td><td></td><td></td><td>2014-12-03 14:42:11</td><td></td><td>7c4411ce-02f1-39b5-b9ec-c ambulatory</td><td></td></tr> <tr><td>0002adb8-59c3-494a-bb17-1e52e4fe-13c7-41ad-8b3e-5680f4af2-775d-34c4-b213-2 49318f80-bd8b-3fc7-a096-a 2014-12-06 12:52:02</td><td></td><td></td><td></td><td>2014-12-06 13:43:02</td><td></td><td>6e2f1a2d-27bd-3701-8d08- ambulatory</td><td></td></tr> <tr><td>0004e2e7-e3f2-4d25-b3eb-l 8e075015-6b2e-4fcf-bc67-5735b7f6-9d26-3c4a-b6d2-7b5845b8-025a-373d-8bcb-2001-11-11 02:07:45</td><td></td><td></td><td></td><td>2001-11-11 02:37:45</td><td></td><td>7c4411ce-02f1-39b5-b9ec-c wellness</td><td></td></tr> <tr><td>00052e41-7581-46e1-8c7f-i 13a3a783-ecc5-4ec5-b3dc-f61a6a29-7081-3071-8d99-78dbf052-6958-3c67-af45-l 2000-03-03 08:37:29</td><td></td><td></td><td></td><td>2000-03-03 08:52:29</td><td></td><td>6e2f1a2d-27bd-3701-8d08- outpatient</td><td></td></tr> <tr><td>00055b87-0a03-4ca8-a69d-19d2cf08-439b-454a-b47e-l 793c18cd-8269-387d-b998-d311e70d-86e7-3c03-b115-1976-09-07 09:30:42</td><td></td><td></td><td></td><td>1976-09-07 12:45:42</td><td></td><td>b1c428d6-4f07-31e0-90f0- ambulatory</td><td></td></tr> <tr><td>0005b0a0-1b05-40ec-a741-ef0e5433-6dab-4608-8664-! 0f365bed-2aa2-3340-974f-3 d5117822-5756-389d-9547 2018-02-20 01:11:43</td><td></td><td></td><td></td><td>2018-02-20 01:44:43</td><td></td><td>b1c428d6-4f07-31e0-90f0- outpatient</td><td></td></tr> <tr><td>000b55be-23ec-4f4a-a77d- ba190ea7-9318-4b89-a17a-75895e9f-c056-3c5d-b845- a95684bd-b222-3e5e-8d18-1947-08-12 11:08:08</td><td></td><td></td><td></td><td>1947-08-12 11:38:08</td><td></td><td>047f6ec3-6215-35eb-9608- wellness</td><td></td></tr> <tr><td>000c97b3-5832-4ecc-8a47- 3dd2d29-7cd0-48f7-b859- b0404cd2f-3f27-35bc-8069-f 3d10019f-c88e-3de5-9916-l 2018-11-17 22:09:56</td><td></td><td></td><td></td><td>2018-11-17 22:24:56</td><td></td><td>b3221fcf-24fb-339e-823d-b outpatient</td><td></td></tr> <tr><td>000e9d31-d256-4308-91a3- b44b3fde-b4bd-46eb-8ced-c 25ada469-4d5d-3bc4-8a30- 42b37eb8-560c-34e2-186- 2015-09-04 09:49:43</td><td></td><td></td><td></td><td>2015-09-04 10:04:43</td><td></td><td>4d71f845-a6a9-3c39-b242- wellness</td><td></td></tr> <tr><td>000d18f9-4f7f-41be-b874-7 cae10920-f977-48b4-a0d3-4 8f9aea5b-fd01-37c0-8931-1 be4c63f3-8d38-3fa9-a183-6 2000-01-19 23:35:03</td><td></td><td></td><td></td><td>2000-01-19 23:50:03</td><td></td><td>7caa7254-5050-3b5e-9eae-l ambulatory</td><td></td></tr> <tr><td>000fef3b-ba44-4b01-952e-7 13b9a676-7cf7-4b7c-bdb6-4 40d7ff6c-5040-39aa-a876-1 3bd5eda0-16da-3ba5-8500- 1976-07-30 17:27:52</td><td></td><td></td><td></td><td>1976-07-30 18:57:52</td><td></td><td>6e2f1a2d-27bd-3701-8d08- emergency</td><td></td></tr> <tr><td>0010a1d9-1d86-415d-b762- c6fc7fe8-483a-4df7-ba2b-2 2d8632ad-16af-3773-a63e-3 3490b118-b33a-38e3-b763- 1993-09-17 22:29:25</td><td></td><td></td><td></td><td>1993-09-17 22:59:25</td><td></td><td>42c4fcfa7-f8a9-3cd1-982a-d wellness</td><td></td></tr> <tr><td>0013356e-7526-47c9-bf67- 5dccb2a6-668f-41b4-add4-4 4ad998df-d944-37d1-b4dc-c 027988f5-9d2d-367f-a9a4-7 2014-08-24 00:22:39</td><td></td><td></td><td></td><td>2014-08-24 00:37:39</td><td></td><td>b1c428d6-4f07-31e0-90f0- wellness</td><td></td></tr> <tr><td>001443a3-614f-4169-97d5- cae10920-f977-48b4-a0d3-4 8f9aea5b-fd01-37c0-8931-1 be4c63f3-8d38-3fa9-a183-6 1997-10-16 03:35:03</td><td></td><td></td><td></td><td>1997-10-17 02:50:03</td><td></td><td>7caa7254-5050-3b5e-9eae-l ambulatory</td><td></td></tr> <tr><td>00173efd-f51e-4de1-9137-3 04dff6e5-123a-4c13-bd08-a f4eb93d1-9187-3cfb-83a4-E d692e283-0833-3201-8e55- 2010-08-02 00:51:06</td><td></td><td></td><td></td><td>2010-08-02 01:06:06</td><td></td><td>b1c428d6-4f07-31e0-90f0- ambulatory</td><td></td></tr> <tr><td>001b0e21-67f3-4b39-947d- 69b60335-4bd9-4043-ad76- 5d35f370-ec4d-3b67-b43a- 0982bd2c-c422-3e3e-bbf5-1 2018-10-27 10:26:36</td><td></td><td></td><td></td><td>2018-10-27 10:41:36</td><td></td><td>4d71f845-a6a9-3c39-b242- wellness</td><td></td></tr> <tr><td>001ba538-57ed-4f7e-9447-l 86b97fc7-ea8f-4e0d-8e66-d 3180b739-e823-37a0-b307- 6f122869-a856-3d65-8d99- 2017-11-23 02:08:27</td><td></td><td></td><td></td><td>2017-11-24 02:34:27</td><td></td><td>6e2f1a2d-27bd-3701-8d08- inpatient</td><td></td></tr> </tbody> </table>	encounter_id	patient_id	provider_id	organization_id	encounter_start_date	encounter_end_date	payer_uuid	encounter_class	0000d0b7-937c-498a-9da4-714b9c18-783d-4f52-aa64-a3a65bdb3-12b2-3247-8e8a-5259a506-b80b-3ed5-9c30-1994-09-07 04:48:12										1994-09-07 05:03:12		d47b3510-2895-3b70-9897- wellness		000186d2-1316-4b58-be65-8be62b06-5994-47fe-aad8-56c64467-dd8a-36ca-91c5-8ad64ecf-c817-3753-bee7-c 2014-12-03 13:12:11				2014-12-03 14:42:11		7c4411ce-02f1-39b5-b9ec-c ambulatory		0002adb8-59c3-494a-bb17-1e52e4fe-13c7-41ad-8b3e-5680f4af2-775d-34c4-b213-2 49318f80-bd8b-3fc7-a096-a 2014-12-06 12:52:02				2014-12-06 13:43:02		6e2f1a2d-27bd-3701-8d08- ambulatory		0004e2e7-e3f2-4d25-b3eb-l 8e075015-6b2e-4fcf-bc67-5735b7f6-9d26-3c4a-b6d2-7b5845b8-025a-373d-8bcb-2001-11-11 02:07:45				2001-11-11 02:37:45		7c4411ce-02f1-39b5-b9ec-c wellness		00052e41-7581-46e1-8c7f-i 13a3a783-ecc5-4ec5-b3dc-f61a6a29-7081-3071-8d99-78dbf052-6958-3c67-af45-l 2000-03-03 08:37:29				2000-03-03 08:52:29		6e2f1a2d-27bd-3701-8d08- outpatient		00055b87-0a03-4ca8-a69d-19d2cf08-439b-454a-b47e-l 793c18cd-8269-387d-b998-d311e70d-86e7-3c03-b115-1976-09-07 09:30:42				1976-09-07 12:45:42		b1c428d6-4f07-31e0-90f0- ambulatory		0005b0a0-1b05-40ec-a741-ef0e5433-6dab-4608-8664-! 0f365bed-2aa2-3340-974f-3 d5117822-5756-389d-9547 2018-02-20 01:11:43				2018-02-20 01:44:43		b1c428d6-4f07-31e0-90f0- outpatient		000b55be-23ec-4f4a-a77d- ba190ea7-9318-4b89-a17a-75895e9f-c056-3c5d-b845- a95684bd-b222-3e5e-8d18-1947-08-12 11:08:08				1947-08-12 11:38:08		047f6ec3-6215-35eb-9608- wellness		000c97b3-5832-4ecc-8a47- 3dd2d29-7cd0-48f7-b859- b0404cd2f-3f27-35bc-8069-f 3d10019f-c88e-3de5-9916-l 2018-11-17 22:09:56				2018-11-17 22:24:56		b3221fcf-24fb-339e-823d-b outpatient		000e9d31-d256-4308-91a3- b44b3fde-b4bd-46eb-8ced-c 25ada469-4d5d-3bc4-8a30- 42b37eb8-560c-34e2-186- 2015-09-04 09:49:43				2015-09-04 10:04:43		4d71f845-a6a9-3c39-b242- wellness		000d18f9-4f7f-41be-b874-7 cae10920-f977-48b4-a0d3-4 8f9aea5b-fd01-37c0-8931-1 be4c63f3-8d38-3fa9-a183-6 2000-01-19 23:35:03				2000-01-19 23:50:03		7caa7254-5050-3b5e-9eae-l ambulatory		000fef3b-ba44-4b01-952e-7 13b9a676-7cf7-4b7c-bdb6-4 40d7ff6c-5040-39aa-a876-1 3bd5eda0-16da-3ba5-8500- 1976-07-30 17:27:52				1976-07-30 18:57:52		6e2f1a2d-27bd-3701-8d08- emergency		0010a1d9-1d86-415d-b762- c6fc7fe8-483a-4df7-ba2b-2 2d8632ad-16af-3773-a63e-3 3490b118-b33a-38e3-b763- 1993-09-17 22:29:25				1993-09-17 22:59:25		42c4fcfa7-f8a9-3cd1-982a-d wellness		0013356e-7526-47c9-bf67- 5dccb2a6-668f-41b4-add4-4 4ad998df-d944-37d1-b4dc-c 027988f5-9d2d-367f-a9a4-7 2014-08-24 00:22:39				2014-08-24 00:37:39		b1c428d6-4f07-31e0-90f0- wellness		001443a3-614f-4169-97d5- cae10920-f977-48b4-a0d3-4 8f9aea5b-fd01-37c0-8931-1 be4c63f3-8d38-3fa9-a183-6 1997-10-16 03:35:03				1997-10-17 02:50:03		7caa7254-5050-3b5e-9eae-l ambulatory		00173efd-f51e-4de1-9137-3 04dff6e5-123a-4c13-bd08-a f4eb93d1-9187-3cfb-83a4-E d692e283-0833-3201-8e55- 2010-08-02 00:51:06				2010-08-02 01:06:06		b1c428d6-4f07-31e0-90f0- ambulatory		001b0e21-67f3-4b39-947d- 69b60335-4bd9-4043-ad76- 5d35f370-ec4d-3b67-b43a- 0982bd2c-c422-3e3e-bbf5-1 2018-10-27 10:26:36				2018-10-27 10:41:36		4d71f845-a6a9-3c39-b242- wellness		001ba538-57ed-4f7e-9447-l 86b97fc7-ea8f-4e0d-8e66-d 3180b739-e823-37a0-b307- 6f122869-a856-3d65-8d99- 2017-11-23 02:08:27				2017-11-24 02:34:27		6e2f1a2d-27bd-3701-8d08- inpatient										
encounter_id	patient_id	provider_id	organization_id	encounter_start_date	encounter_end_date	payer_uuid	encounter_class																																																																																																																																																																
0000d0b7-937c-498a-9da4-714b9c18-783d-4f52-aa64-a3a65bdb3-12b2-3247-8e8a-5259a506-b80b-3ed5-9c30-1994-09-07 04:48:12				1994-09-07 05:03:12		d47b3510-2895-3b70-9897- wellness																																																																																																																																																																	
000186d2-1316-4b58-be65-8be62b06-5994-47fe-aad8-56c64467-dd8a-36ca-91c5-8ad64ecf-c817-3753-bee7-c 2014-12-03 13:12:11				2014-12-03 14:42:11		7c4411ce-02f1-39b5-b9ec-c ambulatory																																																																																																																																																																	
0002adb8-59c3-494a-bb17-1e52e4fe-13c7-41ad-8b3e-5680f4af2-775d-34c4-b213-2 49318f80-bd8b-3fc7-a096-a 2014-12-06 12:52:02				2014-12-06 13:43:02		6e2f1a2d-27bd-3701-8d08- ambulatory																																																																																																																																																																	
0004e2e7-e3f2-4d25-b3eb-l 8e075015-6b2e-4fcf-bc67-5735b7f6-9d26-3c4a-b6d2-7b5845b8-025a-373d-8bcb-2001-11-11 02:07:45				2001-11-11 02:37:45		7c4411ce-02f1-39b5-b9ec-c wellness																																																																																																																																																																	
00052e41-7581-46e1-8c7f-i 13a3a783-ecc5-4ec5-b3dc-f61a6a29-7081-3071-8d99-78dbf052-6958-3c67-af45-l 2000-03-03 08:37:29				2000-03-03 08:52:29		6e2f1a2d-27bd-3701-8d08- outpatient																																																																																																																																																																	
00055b87-0a03-4ca8-a69d-19d2cf08-439b-454a-b47e-l 793c18cd-8269-387d-b998-d311e70d-86e7-3c03-b115-1976-09-07 09:30:42				1976-09-07 12:45:42		b1c428d6-4f07-31e0-90f0- ambulatory																																																																																																																																																																	
0005b0a0-1b05-40ec-a741-ef0e5433-6dab-4608-8664-! 0f365bed-2aa2-3340-974f-3 d5117822-5756-389d-9547 2018-02-20 01:11:43				2018-02-20 01:44:43		b1c428d6-4f07-31e0-90f0- outpatient																																																																																																																																																																	
000b55be-23ec-4f4a-a77d- ba190ea7-9318-4b89-a17a-75895e9f-c056-3c5d-b845- a95684bd-b222-3e5e-8d18-1947-08-12 11:08:08				1947-08-12 11:38:08		047f6ec3-6215-35eb-9608- wellness																																																																																																																																																																	
000c97b3-5832-4ecc-8a47- 3dd2d29-7cd0-48f7-b859- b0404cd2f-3f27-35bc-8069-f 3d10019f-c88e-3de5-9916-l 2018-11-17 22:09:56				2018-11-17 22:24:56		b3221fcf-24fb-339e-823d-b outpatient																																																																																																																																																																	
000e9d31-d256-4308-91a3- b44b3fde-b4bd-46eb-8ced-c 25ada469-4d5d-3bc4-8a30- 42b37eb8-560c-34e2-186- 2015-09-04 09:49:43				2015-09-04 10:04:43		4d71f845-a6a9-3c39-b242- wellness																																																																																																																																																																	
000d18f9-4f7f-41be-b874-7 cae10920-f977-48b4-a0d3-4 8f9aea5b-fd01-37c0-8931-1 be4c63f3-8d38-3fa9-a183-6 2000-01-19 23:35:03				2000-01-19 23:50:03		7caa7254-5050-3b5e-9eae-l ambulatory																																																																																																																																																																	
000fef3b-ba44-4b01-952e-7 13b9a676-7cf7-4b7c-bdb6-4 40d7ff6c-5040-39aa-a876-1 3bd5eda0-16da-3ba5-8500- 1976-07-30 17:27:52				1976-07-30 18:57:52		6e2f1a2d-27bd-3701-8d08- emergency																																																																																																																																																																	
0010a1d9-1d86-415d-b762- c6fc7fe8-483a-4df7-ba2b-2 2d8632ad-16af-3773-a63e-3 3490b118-b33a-38e3-b763- 1993-09-17 22:29:25				1993-09-17 22:59:25		42c4fcfa7-f8a9-3cd1-982a-d wellness																																																																																																																																																																	
0013356e-7526-47c9-bf67- 5dccb2a6-668f-41b4-add4-4 4ad998df-d944-37d1-b4dc-c 027988f5-9d2d-367f-a9a4-7 2014-08-24 00:22:39				2014-08-24 00:37:39		b1c428d6-4f07-31e0-90f0- wellness																																																																																																																																																																	
001443a3-614f-4169-97d5- cae10920-f977-48b4-a0d3-4 8f9aea5b-fd01-37c0-8931-1 be4c63f3-8d38-3fa9-a183-6 1997-10-16 03:35:03				1997-10-17 02:50:03		7caa7254-5050-3b5e-9eae-l ambulatory																																																																																																																																																																	
00173efd-f51e-4de1-9137-3 04dff6e5-123a-4c13-bd08-a f4eb93d1-9187-3cfb-83a4-E d692e283-0833-3201-8e55- 2010-08-02 00:51:06				2010-08-02 01:06:06		b1c428d6-4f07-31e0-90f0- ambulatory																																																																																																																																																																	
001b0e21-67f3-4b39-947d- 69b60335-4bd9-4043-ad76- 5d35f370-ec4d-3b67-b43a- 0982bd2c-c422-3e3e-bbf5-1 2018-10-27 10:26:36				2018-10-27 10:41:36		4d71f845-a6a9-3c39-b242- wellness																																																																																																																																																																	
001ba538-57ed-4f7e-9447-l 86b97fc7-ea8f-4e0d-8e66-d 3180b739-e823-37a0-b307- 6f122869-a856-3d65-8d99- 2017-11-23 02:08:27				2017-11-24 02:34:27		6e2f1a2d-27bd-3701-8d08- inpatient																																																																																																																																																																	

<b>encounter_columns</b>	encounter_id , patient_id , provider_id , organization_id , encounter_start_date , encounter_end_date , payer_uuid , encounter_class , encounter_code , encounter_description , base_encounter_cost , total_claim_cost , payer_coverage , encounter_reason_code , encounter_reason_description
--------------------------	--

## Observation Table

314 SELECT \* FROM observation ORDER BY observation\_id; 5ms

315

observation ×

Properties DATA Log ER Monitor 🔍 ⚡

SELECT \* FROM observation ORDER BY observation\_id LIMIT 100

Cost: 11ms < 1 2 3 4 ... 2997 > Total 299697

	observation_id	observation_date	patient_id	encounter_id	observation_code	observation_desc	observation_value	observation_units
>	1	2012-01-23 17:45:28	034e9e3b-2def-4559-bb2a-	e88bc3a9-007c-405e-aabc-	8302-2	Body Height	193.3	cm
>	2	2012-01-23 17:45:28	034e9e3b-2def-4559-bb2a-	e88bc3a9-007c-405e-aabc-	72514-3	Pain severity - 0-10 verbal nu	2.0	{score}
>	3	2012-01-23 17:45:28	034e9e3b-2def-4559-bb2a-	e88bc3a9-007c-405e-aabc-	29463-7	Body Weight	87.8	kg
>	4	2012-01-23 17:45:28	034e9e3b-2def-4559-bb2a-	e88bc3a9-007c-405e-aabc-	39156-5	Body Mass Index	23.5	kg/m2
>	5	2012-01-23 17:45:28	034e9e3b-2def-4559-bb2a-	e88bc3a9-007c-405e-aabc-	8462-4	Diastolic Blood Pressure	82.0	mm[Hg]
>	6	2012-01-23 17:45:28	034e9e3b-2def-4559-bb2a-	e88bc3a9-007c-405e-aabc-	8480-6	Systolic Blood Pressure	119.0	mm[Hg]
>	7	2012-01-23 17:45:28	034e9e3b-2def-4559-bb2a-	e88bc3a9-007c-405e-aabc-	8867-4	Heart rate	77.0	/min
>	8	2012-01-23 17:45:28	034e9e3b-2def-4559-bb2a-	e88bc3a9-007c-405e-aabc-	9279-1	Respiratory rate	14.0	/min
>	9	2011-11-17 00:26:23	8d4c4326-e9de-4f45-9a4c-f	ae7555a9-eaff-4c09-98a7-2	8310-5	Body temperature	37.1	Cel
>	10	2010-07-27 12:58:08	10339b10-3cd1-4ac3-ac13-	dae2b7cb-1316-4b78-954f-1	8302-2	Body Height	165.0	cm
>	11	2010-07-27 12:58:08	10339b10-3cd1-4ac3-ac13-	dae2b7cb-1316-4b78-954f-1	72514-3	Pain severity - 0-10 verbal nu	1.0	{score}
>	12	2010-07-27 12:58:08	10339b10-3cd1-4ac3-ac13-	dae2b7cb-1316-4b78-954f-1	29463-7	Body Weight	64.7	kg
>	13	2010-07-27 12:58:08	10339b10-3cd1-4ac3-ac13-	dae2b7cb-1316-4b78-954f-1	39156-5	Body Mass Index	23.8	kg/m2
>	14	2010-07-27 12:58:08	10339b10-3cd1-4ac3-ac13-	dae2b7cb-1316-4b78-954f-1	59576-9	Body mass index (BMI) [Perce	71.3	%
>	15	2010-07-27 12:58:08	10339b10-3cd1-4ac3-ac13-	dae2b7cb-1316-4b78-954f-1	8462-4	Diastolic Blood Pressure	70.0	mm[Hg]
>	16	2012-01-23 17:45:28	034e9e3b-2def-4559-bb2a-	e88bc3a9-007c-405e-aabc-	6690-2	Leukocytes (#/volume] in Bloo	9.5	10*3/uL
>	17	2011-07-28 15:02:18	1d604da9-9a81-4ba9-80c2-	b85c339a-6076-43ed-b9d0-	8302-2	Body Height	181.0	cm
>	18	2010-07-27 12:58:08	10339b10-3cd1-4ac3-ac13-	dae2b7cb-1316-4b78-954f-1	8480-6	Systolic Blood Pressure	121.0	mm[Hg]

**observation\_columns**

observation\_id , observation\_date , patient\_id ,  
 encounter\_id , observation\_code , observation\_desc ,  
 observation\_value , observation\_units ,  
 observation\_type

**Medical Conditions Table**

322 SELECT \* FROM medical\_condition ORDER BY condition\_id; 5ms

323

medical\_condition ×

Properties DATA Log ER Monitor 🔍 ⚡

SELECT \* FROM medical\_condition ORDER BY condition\_id LIMIT 100

Cost: 12ms < 1 2 3 4 ... 84 > Total 8376

	condition_id	condition_start_date	condition_end_date	patient_id	encounter_id	condition_code	condition_descripti
>	1	2001-05-01	(NULL)	1d604da9-9a81-4ba9-80c2-	8f104aa7-4ca9-4473-885a-1	40055000	Chronic sinusitis (disorder)
>	2	2011-08-09	2011-08-16	8d4c4326-e9de-4f45-9a4c-f	9d35ec9f-352a-4629-92ef-3	444814009	Viral sinusitis (disorder)
>	3	2011-11-16	2011-11-26	8d4c4326-e9de-4f45-9a4c-f	ae7555a9-eaff-4c09-98a7-2	195662009	Acute viral pharyngitis (disord
>	4	2011-05-13	2011-05-27	10339b10-3cd1-4ac3-ac13-	e1ab4933-07a1-49f0-b4bd-1	10509002	Acute bronchitis (disorder)
>	5	2011-02-06	2011-02-14	f5dc418-09fe-4a2f-baa0-3c	b8f76eba-7795-4dc4d-a544-f	195662009	Acute viral pharyngitis (disord
>	6	2011-04-18	2011-04-28	f5dc418-09fe-4a2f-baa0-3c	640837d9-845a-433c-9fad-1	195662009	Acute viral pharyngitis (disord
>	7	2011-11-29	2011-12-13	f5dc418-09fe-4a2f-baa0-3c	8c929690-1826-4b84-81c9-	444814009	Viral sinusitis (disorder)
>	8	2011-12-31	2012-01-07	f5dc418-09fe-4a2f-baa0-3c	16300c56-a035-4126-a656-	10509002	Acute bronchitis (disorder)
>	9	2011-12-08	2011-12-22	1d604da9-9a81-4ba9-80c2-	792fae81-a007-44b0-8221-	444814009	Viral sinusitis (disorder)
>	10	2016-12-29	2017-01-05	034e9e3b-2def-4559-bb2a-	3b639086-5fbc-4720-8c31-	10509002	Acute bronchitis (disorder)
>	11	2011-08-11	2011-09-01	10339b10-3cd1-4ac3-ac13-	470ccc46-0d6b-4a60-9f52-f	444814009	Viral sinusitis (disorder)
>	12	2019-03-20	2019-04-10	1d604da9-9a81-4ba9-80c2-	4e595f0c-f50f-461b-a04e-1:	444814009	Viral sinusitis (disorder)
>	13	2015-12-06	2015-12-14	8d4c4326-e9de-4f45-9a4c-f	5818152d-9993-4ed3-a021-	43879008	Streptococcal sore throat (dis
>	14	2017-01-22	2017-02-12	10339b10-3cd1-4ac3-ac13-	4ec8d55b-05fc-42a5-bfa3-1	284551006	Laceration of foot
>	15	2019-04-23	2019-05-07	10339b10-3cd1-4ac3-ac13-	27ff7518-6d93-4308-8a1d-c	10509002	Acute bronchitis (disorder)
>	16	2016-07-03	2016-07-16	f5dc418-09fe-4a2f-baa0-3c	03a1b666-eda5-44e9-b263-	195662009	Acute viral pharyngitis (disord
>	17	2013-09-27	2013-11-22	b1e9b0b9-da6e-4f68-b603-l	004afbc0-d906-4f37-b9d1-5	75498004	Acute bacterial sinusitis (disor
>	18	2013-11-15	(NULL)	b1e9b0b9-da6e-4f68-b603-l	09e946c1-67fd-4234-93e0-i	40055000	Chronic sinusitis (disorder)

<b>medical_condition_columns</b>	condition_id , condition_start_date , condition_end_date , patient_id , encounter_id , condition_code , condition_description
----------------------------------	---

## Procedures Table

329 SELECT \* FROM procedures ORDER BY procedure\_id; 3ms  
330

procedures x Properties DATA Log ER Monitor ( )

SELECT \* FROM procedures ORDER BY procedure\_id LIMIT 100

procedure\_id procedure\_date patient\_id encounter\_id procedure\_code procedure\_description procedure\_base\_cost procedure\_reason\_code

bigint datetime varchar(36) varchar(36) varchar(36) varchar(255) decimal(12,2) varchar(36)

Cost: 11ms < 1 2 3 4 ... 350 > Total 34981

<b>procedures_columns</b>	procedure_id , procedure_date , patient_id , encounter_id , procedure_code , procedure_description , procedure_base_cost , procedure_reason_code , procedure_reason_description
---------------------------	---

## Diagnosis Table

```

337  SELECT * FROM diagnosis      ORDER BY patient_id, condition_id;  3ms
338
diagnosis x
Properties DATA Log ER Monitor 🔍 { } ⏪
SELECT * FROM diagnosis      ORDER BY patient_id, condition_id LIMIT 100
+-----+-----+
| Q   | 🔍 patient_id | 🔍 condition_id |
|     | varchar(36)    | bigint          |
+-----+-----+
| > 00185faa-2760-4218-9bf5-c 5452 |
| > 00185faa-2760-4218-9bf5-c 5453 |
| > 00185faa-2760-4218-9bf5-c 5454 |
| > 00185faa-2760-4218-9bf5-c 5455 |
| > 00185faa-2760-4218-9bf5-c 5456 |
| > 0042862c-9889-4a2e-b782- 5076 |
| > 0042862c-9889-4a2e-b782- 5077 |
| > 0042862c-9889-4a2e-b782- 5078 |
| > 0042862c-9889-4a2e-b782- 5079 |
| > 0047123f-12e7-486c-82df- 7801 |
| > 0047123f-12e7-486c-82df- 7802 |
| > 0047123f-12e7-486c-82df- 7803 |
| > 0047123f-12e7-486c-82df- 7805 |
| > 0047123f-12e7-486c-82df- 7806 |
| > 0047123f-12e7-486c-82df- 7807 |
| > 010d4a3a-2316-45ed-ae15- 1223 |
| > 010d4a3a-2316-45ed-ae15- 1224 |
| > 010d4a3a-2316-45ed-ae15- 1225 |

```

<b>diagnosis_columns</b>	patient_id , condition_id
--------------------------	---------------------------

## Treatment Table

	patient_id	procedure_id
1	00185faa-2760-4218-9bf5-db301acf8274	23030
2	00185faa-2760-4218-9bf5-db301acf8274	23031
3	00185faa-2760-4218-9bf5-db301acf8274	23032
4	00185faa-2760-4218-9bf5-db301acf8274	23033
5	00185faa-2760-4218-9bf5-db301acf8274	23034
6	00185faa-2760-4218-9bf5-db301acf8274	23035
7	00185faa-2760-4218-9bf5-db301acf8274	23036
8	00185faa-2760-4218-9bf5-db301acf8274	23037
9	00185faa-2760-4218-9bf5-db301acf8274	23038
10	00185faa-2760-4218-9bf5-db301acf8274	23039
11	0042862c-9889-4a2e-b782-fac1e540ecb4	20172
12	0042862c-9889-4a2e-b782-fac1e540ecb4	20173
13	0042862c-9889-4a2e-b782-fac1e540ecb4	20174
14	0042862c-9889-4a2e-b782-fac1e540ecb4	20175
15	0042862c-9889-4a2e-b782-fac1e540ecb4	20176
16	0042862c-9889-4a2e-b782-fac1e540ecb4	20177
17	0042862c-9889-4a2e-b782-fac1e540ecb4	20178
18	0042862c-9889-4a2e-b782-fac1e540ecb4	20179

treatment_columns	patient_id , procedure_id
-------------------	---------------------------

## SQL Queries

### 2) INNER JOIN

2. Write an SQL involving the junction table and two other related tables. You must use the INNER JOIN to connect with all three tables. The database that you created must be included in your SQL queries.

```

1  CREATE TABLE Final_Project.rpt_diagnosis_patient_condition AS
2  SELECT
3      d.patient_id,
4      p.first_name,
5      p.last_name,
6      mc.condition_id,
7      mc.condition_code,
```

```

8     mc.condition_description
9     FROM Final_Project.diagnosis AS d
10    INNER JOIN Final_Project.patient AS p
11      ON p.patient_id = d.patient_id
12    INNER JOIN Final_Project.medical_condition AS mc
13      ON mc.condition_id = d.condition_id;
14
15   SELECT *
16   FROM Final_Project.rpt_diagnosis_patient_condition
17  ORDER BY patient_id, condition_id;
18
19 -- top 10 conditions by DISTINCT patients (prevalence)
20   SELECT
21     condition_description,
22     COUNT(DISTINCT patient_id) AS n_patients
23   FROM Final_Project.rpt_diagnosis_patient_condition
24  GROUP BY condition_description
25  ORDER BY n_patients DESC, condition_description
26  LIMIT 10;

```

Final\_Project.rpt\_diagnos... ×

The screenshot shows a table titled "Final\_Project.rpt\_diagnos...". The table has 7 columns: patient\_id, first\_name, last\_name, condition\_id, condition\_code, and condition\_description. The table contains 15 rows of data. The columns are ordered by patient\_id, then condition\_id.

	*patient_id	*first_name	*last_name	*condition_id	*condition_code	*condition_descripti
Q	varchar(36)	varchar(20)	varchar(20)	bigint	varchar(36)	varchar(255)
00185faa-2760-4218-9bf5-c	Eusebio566	Wyman904	5452	195662009	Acute viral pharyngitis (disord	
00185faa-2760-4218-9bf5-c	Eusebio566	Wyman904	5453	43878008	Streptococcal sore throat (disord	
00185faa-2760-4218-9bf5-c	Eusebio566	Wyman904	5454	82423001	Chronic pain	
00185faa-2760-4218-9bf5-c	Eusebio566	Wyman904	5455	124171000119105	Chronic intractable migraine w	
00185faa-2760-4218-9bf5-c	Eusebio566	Wyman904	5456	196416002	Impacted molars	
0042862c-9889-4a2e-b782-	Dewitt635	Feest103	5076	195662009	Acute viral pharyngitis (disord	
0042862c-9889-4a2e-b782-	Dewitt635	Feest103	5077	195662009	Acute viral pharyngitis (disord	
0042862c-9889-4a2e-b782-	Dewitt635	Feest103	5078	65363002	Otitis media	
0042862c-9889-4a2e-b782-	Dewitt635	Feest103	5079	10509002	Acute bronchitis (disorder)	
0047123f-12e7-486c-82df-E	Jordon466	Harber290	7801	444814009	Viral sinusitis (disorder)	
0047123f-12e7-486c-82df-E	Jordon466	Harber290	7802	10509002	Acute bronchitis (disorder)	
0047123f-12e7-486c-82df-E	Jordon466	Harber290	7803	74400008	Appendicitis	
0047123f-12e7-486c-82df-E	Jordon466	Harber290	7805	428251008	History of appendectomy	

	condition_descripti	n_patients
1	Viral sinusitis (disorder)	743
2	Acute viral pharyngitis (disorder)	492
3	Acute bronchitis (disorder)	464
4	Body mass index 30+ - obesity	449
5	Prediabetes	317
6	Hypertension	302
7	Anemia (disorder)	300
8	Chronic sinusitis (disorder)	233
9	Miscarriage in first trimester	221
10	Normal pregnancy	205

Output: I've created the table and INNER JOIN on patient\_id from two tables: medical conditions and patient leaving out NULL values (if applicable), then I can show the number of patients diagnosed with each conditions shown above.

### 3) LEFT OUTER JOIN

3. Write an SQL statement by including two or more tables and using the LEFT OUTER JOIN. Show the results and sort the results by key field(s). Interpret the results compared to what an INNER JOIN does.

```

1  CREATE TABLE rpt_encounter_activity AS
2  SELECT
3    e.encounter_id,
4    e.patient_id,
5    e.encounter_start_date,
6    e.encounter_class,
7    pr.provider_id,
8    pr.organization_id,
9    pr.provider_name,
10   COUNT(pc.procedure_id) AS n_procedures
11  FROM encounter e
12  LEFT JOIN provider pr
13  ON pr.provider_id = e.provider_id
14  AND pr.organization_id = e.organization_id
15  LEFT JOIN procedures pc
16  ON pc.encounter_id = e.encounter_id
17  GROUP BY
18    e.encounter_id, e.patient_id, e.encounter_start_date,
19    e.encounter_class,
20    pr.provider_id, pr.organization_id, pr.provider_name;

```

```

21  SELECT
22    encounter_id,
23    encounter_class,
24    provider_id,
25    provider_name,
26    n_procedures
27  FROM rpt_encounter_activity
28  WHERE provider_id IS NULL
29  ORDER BY encounter_id;
30

```

rpt_encounter_activity					
Cost: 30ms  Total 0					
	* encounter_id	* encounter_class	provider_id	provider_name	* n_procedures
	varchar(36)	varchar(50)	varchar(36)	varchar(30)	bigint

Output: When creating a new table, I LEFT JOIN on provider\_id, organization\_id (unique key), and encounter\_id to find out if there are any encounters that has no recorded providers. Apparently, every encounter always have providers recorded which is good to know.

## 4) SINGLE-ROW SUBQUERY

4. Write a single-row subquery. Show the results and sort the results by key field(s). Interpret the output.

```

1  SELECT patient_id, first_name, last_name, patient_gender,
2    patient_city, patient_state, healthcare_expenses, healthcare_coverage
3  FROM patient
4  WHERE healthcare_expenses = (SELECT MAX(healthcare_expenses) FROM
5    patient)
6  ORDER BY patient_id;

```

Result(R0)								
Cost: 7ms  Total 1								
	patient_id	first_name	last_name	patient_gender	patient_city	patient_state	healthcare_expenses	healthcare_coverage
>	19d2cfb8-439b-454a-b47e-5274c219005b	Walker122	Kuhic920	M	Worcester	Massachusetts	2145924.40	3559.68

Output: I select few necessary columns associated with patient records and find a patient with the highest healthcare expense cost using SELECT MAX. The result is shown above.

## 5) MULTIPLE-ROW SUBQUERY

5. Write a multiple-row subquery. Show the results and sort the results by key field(s). Interpret the output.

```

1  CREATE TABLE rpt_providers_highrisk_er AS
2  WITH provider_tot AS (
3    SELECT provider_id, organization_id, COUNT(*) AS total_encounters
4    FROM encounter
5    GROUP BY provider_id, organization_id
6  ),
7  highrisk_patients AS (
8    SELECT e.patient_id
9    FROM encounter e
10   WHERE LOWER(e.encounter_class) = 'emergency'
11   GROUP BY e.patient_id
12   HAVING COUNT(*) >= 10
13 )
14 SELECT
15   pr.provider_id,
16   pr.organization_id,
17   pr.provider_name,
18   pr.provider_specialty,
19   COUNT(DISTINCT e.patient_id) AS n_highrisk_patients,
20   COUNT(*) AS n_highrisk_encounters,
21   pt.total_encounters,
22   ROUND(100 * COUNT(*) / NULLIF(pt.total_encounters, 0), 2) AS
23     pct_of_provider_encounters
24   FROM provider pr
25   JOIN encounter e
26   ON e.provider_id = pr.provider_id
27   AND e.organization_id = pr.organization_id
28   JOIN provider_tot pt
29   ON pt.provider_id = pr.provider_id
30   AND pt.organization_id = pr.organization_id
31   WHERE e.patient_id IN (SELECT patient_id FROM highrisk_patients) --
32     multi-row subquery use
33   GROUP BY
34   pr.provider_id, pr.organization_id, pr.provider_name,
35   pr.provider_specialty, pt.total_encounters;
36
37   SELECT *
38   FROM rpt_providers_highrisk_er
39   ORDER BY n_highrisk_encounters DESC;

```

provider_id	organization_id	provider_name	provider_specialty	n_highrisk_patient	n_highrisk_encoun	total_encounters	pct_of_provider_end
varchar(36)	varchar(36)	varchar(30)	varchar(50)	bigint	bigint	bigint	decimal(26,2)
8f9aea5b-fd01-37c0-8931-1	be4c63f3-8d38-3fa9-a183-6	Gaynell126 Streich926	GENERAL PRACTICE	2	3059	3217	95.09
4cbaa6db-aa54-3101-b871-1	e0c85f99-d520-3f26-8dcd-c	Garth972 Wyman904	GENERAL PRACTICE	1	489	531	92.09
a83b5619-bcec-3877-9a3e-1	fbf6180e-b800-3ebe-b91d-5	Von197 Mraz590	GENERAL PRACTICE	2	315	534	58.99
93330079-96b3-3b50-8097	53b25c4b-e95f-3dd1-8e69-1	Terry864 Wiegand701	GENERAL PRACTICE	1	235	352	66.76
98a31629-e9a3-304d-a821-1	62fa9127-485f-3032-87cc-6	Veda284 Pfeffer420	GENERAL PRACTICE	1	229	332	68.98
608920ae-c94c-3543-9209-1	7c530dcd-43f3-3f87-b41b-5	Óscar156 Mateo562	GENERAL PRACTICE	1	209	413	50.61
7572448e-118a-3102-84ac-1	b81688f5-bd0e-3c99-963f-5	Malinda718 Cassin499	GENERAL PRACTICE	1	208	326	63.80
43710ea5-5bf8-3fd0-a0ce-3	27f06acd-f732-3871-be87-6	Myong12 Heidenreich818	GENERAL PRACTICE	2	136	142	95.77
5718012f-98aa-3971-ad2e-1	90cf148c-69ed-3d33-aa86-1	Edie35 Howell947	GENERAL PRACTICE	1	134	187	71.66
e6283e46-fd81-3611-9459-1	e002090d-4e92-300e-b41e-1	Jeanmarie510 Beatty507	GENERAL PRACTICE	2	126	1028	12.26

Output: I created a new table that counts the number of encounters (distinct patients) categorized as emergency indicating high risk for each provider and organization ID. Thus, this is a multi-row query with the result shown above ordered by the number of high risk encounters.

## 6) AGGREGATION

6. Write an SQL to aggregate the results by using multiple columns in the SELECT clause. Interpret the output.

```

1  SELECT pr.provider_id,
2    pr.provider_specialty,
3    COUNT(pc.procedure_id) AS total_procedures
4  FROM provider pr
5  LEFT JOIN encounter e
6  ON pr.provider_id = e.provider_id
7  AND pr.organization_id = e.organization_id
8  LEFT JOIN procedures pc
9  ON pc.encounter_id = e.encounter_id
10 GROUP BY pr.provider_id, pr.provider_specialty
11 ORDER BY total_procedures DESC;

```

provider_id	provider_specialty	total_procedures
varchar	varchar	bigint
8f9aea5b-fd01-37c0-8931-1	GENERAL PRACTICE	2398
f4eb93d1-9187-3cfb-83a4-6	GENERAL PRACTICE	1473
793c18cd-8269-387d-b998-	GENERAL PRACTICE	1336
e6283e46-fd81-3611-9459-1	GENERAL PRACTICE	1121
0a8a9359-7b33-3256-a068-	GENERAL PRACTICE	1110
3180b739-e823-37a0-b307-	GENERAL PRACTICE	951
40d7ff6c-5040-39aa-a876-1	GENERAL PRACTICE	936
b08f34d7-3c08-35b9-a2ba-1	GENERAL PRACTICE	855
4b04cd2f-3f27-35bc-8069-f	GENERAL PRACTICE	846
680f4af2-775d-34c4-b213-2	GENERAL PRACTICE	845

Output: Here, I LEFT JOIN encounter\_id (provider -> encounter) allowing me to also LEFT JOIN procedures on encounter\_ID (encounter to procedures). That way, I can group by provider\_id and specialty to find out what specialties are the most common on total procedures performed. Turns out, general practice is very common. I guess good doctors gotta be prepared for any situation, eh?

## 7) NOT IN OPERATOR SUBQUERY

7. Write a subquery using the NOT IN operator. Show the results and sort the results by key field(s). Interpret the output.

```

1  SELECT patient_id, first_name, last_name
2  FROM patient
3  WHERE patient_id NOT IN (SELECT DISTINCT patient_id FROM diagnosis)
4  ORDER BY first_name, last_name;

```

Result(RO)			
	patient_id	first_name	last_name
Q	varchar	varchar	varchar
>	c038adf1-91e4-4d05-a7ce-k	Claude750	Russel238
>	64c260d6-c09b-4280-ab86-	Collin529	Beatty507
>	d9988dce-ec05-4138-8f04-1	Dan465	Walker122
>	f592e861-6e2e-46bc-a19d-5	Dorothea248	Ward668
>	6e852965-91fe-4940-bbf7-5	Eusebio566	Beier427
>	2fa0c2b3-48c0-4fd4-b9f3-d1	Fredric73	Douglas31
>	431971f5-a39f-44f8-9ef9-11	Heide509	Walsh511
>	8a999529-683a-4d3a-b256-	Joycelyn213	Abernathy524
>	01207ecd-9dff-4754-8887-4	Karyn217	Mueller846
>	c8c1bcb3-f787-4d53-9e3f-8	Leo278	Senger904

Output: This is pretty straightforward. I find patients that are not found in diagnosis list using patient table as a reference with NOT IN clause. There are 19 patients who do not have recorded diagnoses.

## 8) CASE STATEMENT QUERY

8. Write a query using a CASE statement. Show the results and sort the results by key field(s). Interpret the output.

```

1  SELECT patient_id,
2  healthcare_coverage,
3  CASE

```

```

4 WHEN healthcare_coverage >= 10000 THEN 'High Coverage'
5 WHEN healthcare_coverage >= 5000 THEN 'Medium Coverage'
6 ELSE 'Low Coverage'
7 END AS coverage_category
8 FROM patient
9 ORDER BY patient_id;
10
11 --printed table of summed counts of each coverage
12 SELECT
13 SUM(CASE WHEN healthcare_coverage >= 10000 THEN 1 ELSE 0 END) AS
highcoverage_count,
14 SUM(CASE WHEN healthcare_coverage >= 5000 AND healthcare_coverage <
10000 THEN 1 ELSE 0 END) AS mediumcoverage_count,
15 SUM(CASE WHEN healthcare_coverage < 5000 THEN 1 ELSE 0 END) AS
lowcoverage_count
16 FROM patient;

```

	patient_id	healthcare_coverage	coverage_category
	00185faa-2760-4218-9bf5-c	9863.36	Medium Coverage
	0042862c-9889-4a2e-b782-	1240.76	Low Coverage
	0047123f-12e7-486c-82df-5	7108.77	Medium Coverage
	010d4a3a-2316-45ed-ae15-	3346.40	Low Coverage
	01207ecd-9dff-4754-8887-4	774.96	Low Coverage
	0149d553-f571-4e99-867e-1	6713.48	Medium Coverage
	01e1f394-7219-4189-bceb-	17017.45	High Coverage
	023a7d29-32b3-4db5-89c8-	32086.31	High Coverage
	0288abb6-633c-40c3-ba0c-	47905.58	High Coverage
	02ea2f1a-ddcf-4809-8279-d	2738.96	Low Coverage

highcoverage_count	326
mediumcoverage_count	309
lowcoverage_count	536

Output: Using CASE WHEN clause, I categorize each healthcare coverage amount based on the criteria as low coverage (less than \$5000), medium coverage (greater than or equal to \$5000), or high coverage (greater than \$10,000) when going through each patient. I also performed a useful SQL query that prints the aggregated count of each coverage shown in the small table above.

## 9) NOT EXISTS OPERATOR QUERY

9. Write a query using the NOT EXISTS operator. Show the results and sort the results by key field(s). Interpret the output.

```

1  CREATE TABLE rpt_inactive_providers_by_specialty AS
2  SELECT
3    pr.provider_specialty,
4    COUNT(*) AS n_inactive_providers
5  FROM provider pr
6  WHERE NOT EXISTS (
7    SELECT 1
8    FROM encounter e
9    WHERE e.provider_id = pr.provider_id
10   AND e.organization_id = pr.organization_id
11  )
12 GROUP BY pr.provider_specialty;
13
14 SELECT *
15 FROM rpt_inactive_providers_by_specialty
16 ORDER BY n_inactive_providers DESC, provider_specialty;

```

	* provider_specialty	* n_inactive_provide
	varchar(50)	bigint
1	> INTERNAL MEDICINE	608
2	> NURSE PRACTITIONER	511
3	> CLINICAL SOCIAL WORKER	367
4	> PHYSICIAN ASSISTANT	311
5	> PHYSICAL THERAPY	308
6	> FAMILY PRACTICE	279
7	> CLINICAL PSYCHOLOGIST	204
8	> DIAGNOSTIC RADIOLOGY	171
9	> CHIROPRACTIC	148
10	> OPTOMETRY	139

Output: I created a new table where it sums the count of inactive providers by specialty to find out most frequent specialties that have no recorded encounters (which are no matching row in encounter for the provider & organization ID).

## 10) NOT NULL OPERATOR SUBQUERY

10. Write a subquery using the NOT NULL operator in the inner query. Show the results and sort the results by key field(s). Interpret the output.

```

1  SELECT
2    e.encounter_class,
3    COUNT(*) AS n_encounters_deceased
4  FROM encounter e
5  WHERE e.patient_id IN (
6    SELECT patient_id
7    FROM patient
8    WHERE death_date IS NOT NULL
9  )
10 GROUP BY e.encounter_class
11 ORDER BY n_encounters_deceased DESC, e.encounter_class;

```

The screenshot shows a database query results interface. At the top, there are various icons for search, refresh, export, and navigation. Below the header, there is a table with two columns: 'encounter\_class' and 'n\_encounters\_deceased'. The 'encounter\_class' column contains values like 'ambulatory', 'wellness', 'outpatient', 'urgentcare', 'inpatient', and 'emergency'. The 'n\_encounters\_deceased' column contains corresponding integers. The table is sorted by the second column in descending order.

	encounter_class	n_encounters_deceased
	varchar	bigint
>	ambulatory	5484
>	wellness	3949
>	outpatient	2180
>	urgentcare	1073
>	inpatient	928
>	emergency	673

Output: Here, I sum the count of dead patients (by `death_date` that is NOT NULL) by encounter class. The result is shown above.

## Answered Business Questions

### 1) Increase Profitability

**Business question:** Who/what drives cost, and where might we tighten up?

- **7.3 Top patients by total claim cost** → flags highest cost.
- **7.4 Procedure volume & average base cost** → identifies costly/high-volume procedures.
- **3) Encounters with zero procedures (LEFT JOIN).**

I've been able to find out who has really high claim costs by joining patient and encounter on `patient_id` then summed them up. This will serve as a recommended system to use for SQL queries to flag patients with high costs and consider strategies to assist them with covering cost, thus less risk of financial issues for both parties.

Procedures performed over 1,000 times are the following:

- Medication Reconciliation (procedure)
- Renal dialysis (procedure)
- Auscultation of the fetal heart
- Evaluation of uterine fundal height
- Subcutaneous immunotherapy
- Intramuscular injection

What they have in common seems to be some kind of interventions through medications, therapies, or evaluations. But they're also non-surgical. Thus the quality of services should focus on the streamlined interventions of these methods. This would give providers sufficient time to cover uncommon procedures. Lastly, encounters with zero procedures showed that every encounter had a recorded provider, which means the services were always provided.

## 2) Improve Clinical Quality

**Business question:** Are patients getting timely follow-up and are inpatient stays efficient?

- **7.5 Post-procedure 14-day follow-up rate** → timely follow-up (quality of care).
- **7.2 Inpatient length of stay (LOS) per provider**.
- **10) Encounters for deceased patients (by class)**.

The table `rpt_inpatient_los_provider` shows the number of encounters that are “inpatient” with averaged length of stay (LOS) by each provider who has inpatient encounters. There are a few who have really high encounter numbers and also longer LOS. Either the provider specializes in lengthy procedures or perhaps there is a reason behind longer stay. Ideally, providers want less averaged LOS for any number of encounters. Higher encounters might be good in this context where they’re a popular choice for patients when receiving treatments. Either way, this table serves as a recommended system for tracking providers who may need interventions and improve their skills, or allocate them when necessary to best alleviate workloads, or anything associated with providers’ ability to provide quality service where it is demanded.

Regarding conditions, procedures, etc. and other causes leading to treatments, top 5 conditions seem to be:

- Viral sinusitis (disorder)
- Acute viral pharyngitis (disorder)
- Acute bronchitis (disorder)
- Body mass index 30+ — obesity (finding)
- Prediabetes

Conditions mainly consist of acute upper-respiratory infections and cardiometabolic risk conditions. The most common after that are chronic conditions, obstetric events (like normal

pregnancy, miscarriages, etc.), and other acute infections. Understanding the most frequent conditions can allow providers or organizations to improve care qualities focusing on these areas.

Finally, deaths by encounter class originally showed the top three as ambulatory, wellness, and outpatient. These are typically scheduled visits, so I re-checked using “death within 30 days of a visit” as a rate per 1,000 encounters.

```

1  SELECT
2      LOWER(e.encounter_class) AS encounter_class,
3      SUM(p.death_date IS NOT NULL
4          AND p.death_date BETWEEN e.encounter_start_date AND
5          DATE_ADD(e.encounter_start_date, INTERVAL 30 DAY)) AS
6      deaths_within_30d,
7      COUNT(*) AS total_encounters,
8      ROUND(1000 * AVG(p.death_date IS NOT NULL
9          AND p.death_date BETWEEN e.encounter_start_date AND
10         DATE_ADD(e.encounter_start_date, INTERVAL 30 DAY)), 2) AS
11     rate_per_1000
12 FROM encounter e
13 JOIN patient p ON p.patient_id = e.patient_id
14 GROUP BY LOWER(e.encounter_class)
15 ORDER BY rate_per_1000 DESC, encounter_class;

```

The result shows as follows:

Result(RO) ×				
	encounter_class	deaths_within_30d	total_encounters	rate_per_1000
Q	varchar	decimal	bigint	decimal
>	inpatient	73	1838	39.72
>	emergency	57	2090	27.27
>	urgentcare	9	2373	3.79
>	ambulatory	36	18936	1.90
>	outpatient	11	9003	1.22
>	wellness	12	19106	0.63

The result shows very low rates for those classes and higher for emergency/inpatient. This is a good example of testing whether encounter class (X) really correlates with death rates (Y). Knowing common encounter types, reasons behind them, and correlations like death rates helps organizations prepare ahead of time and provide timely care. This serves as another recommendation system to track care by encounter class and related correlations to tackle clinical quality improvements.

### 3) Optimize Provider Utilization

**Business question:** Are workloads balanced, and where is capacity underused?

- **7.1 Provider utilization (encounters per provider & specialty)** → workload/volume.
- **9) Inactive providers by specialty (NOT EXISTS)**
- **6) Zero-procedure specialties** → specialties with no procedure activity.

Here, I identified the most popular providers (first page top list):

- Gaynell126 Streich926
- Gertrudis163 Schaden604
- Vern/31 Powlowski563
- Jeanmarie510 Beatty507
- Maile 198 Frami345
- Luke971 Rath779

On the same first page, the workload quickly became unbalanced where Gaynell had over 3,000 encounters while every other provider had less than 2,000 and rapidly decreasing when descending (all of them are under general practice specialty). This indicates very unbalanced allocation and would demand better allocation for workload between providers under similar specialty. This also serves as another recommended system to track workload allocations indicated by the number of encounters for each provider.

Most inactive specialties in the top 5 are:

- INTERNAL MEDICINE
- NURSE PRACTITIONER
- CLINICAL SOCIAL WORKER
- PHYSICIAN ASSISTANT
- PHYSICAL THERAPY

The table `rpt_inactive_providers_by_specialty` showed that providers with zero encounters fall under the most frequent specialties listed above. It looks like they mainly function in ambulatory settings where patients are not admitted overnight (same-day care, outpatient environments). This serves as one other system where it tracks encounters by specialty and may prompt minimizing or removing these specialties to better allocate budgeting into other much-needed care. Less is sometimes more such as faster workflow, less busy lineups, etc.

## 4) Reduce Readmissions

**Business question:** Who is at risk of coming back soon, and what can we do about it?

- **7.6 High-risk signal: frequent emergency users ( $\geq 3$  ER visits)** → likely to revisit the ED.
- **7.5 Post-procedure 14-day follow-up rate** → improving follow-up.

These two tables serve as a recommended system where organizations/providers can track procedures and the quality of their service by the number of encounters under 14 days as well as patients with high emergency visits. If the encounter rate is high under 14 days, it would indicate poorer service quality and may need interventions or investigations for further improvements in that area. These patients with high ER visits will be flagged for more streamlined check-ins allowing for quick interventions and less waiting time when necessary.

## 5) Identify Care Strategic Expansion

**Business question:** Where should we grow services or redesign for demand and cost?

- **7.4 Procedure volume & average base cost** → where services are heavily used (and expensive).
- **2) Top conditions by distinct patients** (`rpt_diagnosis_patient_condition`)
- **9) Inactive providers by specialty + 6) Zero-procedure specialties.**

It looks like therapies, evaluations, and medications are commonly used, but procedures in any form (e.g., therapies, evaluations) seem to be very expensive, usually over \$10,000+. I suppose these service areas would need some extra investments for improved care and higher efficiency at a reduced cost due to better equipment and other factors. Same for other conditions like acute upper-respiratory infections, cardiometabolic risk conditions, chronic conditions, obstetric events (like normal pregnancy, miscarriages, etc.), and other acute infections. To reiterate, inactive providers and zero procedures by specialty allow the organization to either maximize budgeting in the most frequent specialties or increase care services in certain lacking specialties only if the circumstance requires it.

In summary, I laid out practical recommendations for improving care services and paired them with other SQL queries tailored for ongoing tracking.

---

Sources:

“A Review of the Cardiometabolic Risk Experience Among Canadian Métis Populations.” *Canadian Journal of Cardiology*, vol. 29, no. 8, Aug. 2013, pp. 1006–13.  
[www.sciencedirect.com](https://www.sciencedirect.com/science/article/pii/S0820179612005029), <https://doi.org/10.1016/j.cjca.2012.11.029>.

*Cost Reduction Strategies for Health Systems.*

<https://www.compassonehealthcare.com/blog/cost-reduction-strategies-health-systems/>. Accessed 14 Aug. 2025.

Drozdz, Dorota, et al. “Obesity and Cardiometabolic Risk Factors: From Childhood to Adulthood.” *Nutrients*, vol. 13, no. 11, Nov. 2021, p. 4176. *PubMed Central*, <https://doi.org/10.3390/nu13114176>.

“Health Care Costs: What’s the Problem?” AAMC, [https://doi.org/10.15766/rai\\_dozyvh2](https://doi.org/10.15766/rai_dozyvh2). Accessed 14 Aug. 2025.

“Take a Listen: What Auscultation Can Say About Your Health.” *Cleveland Clinic*, <https://my.clevelandclinic.org/health/diagnostics/23080-auscultation>. Accessed 14 Aug. 2025.

“What Is Dialysis?” *Cleveland Clinic*, <https://my.clevelandclinic.org/health/treatments/14618-dialysis>. Accessed 14 Aug. 2025.

“What’s an Upper Respiratory Infection?” *Cleveland Clinic*, <https://my.clevelandclinic.org/health/diseases/4022-upper-respiratory-infection>. Accessed 14 Aug. 2025.

---

The End