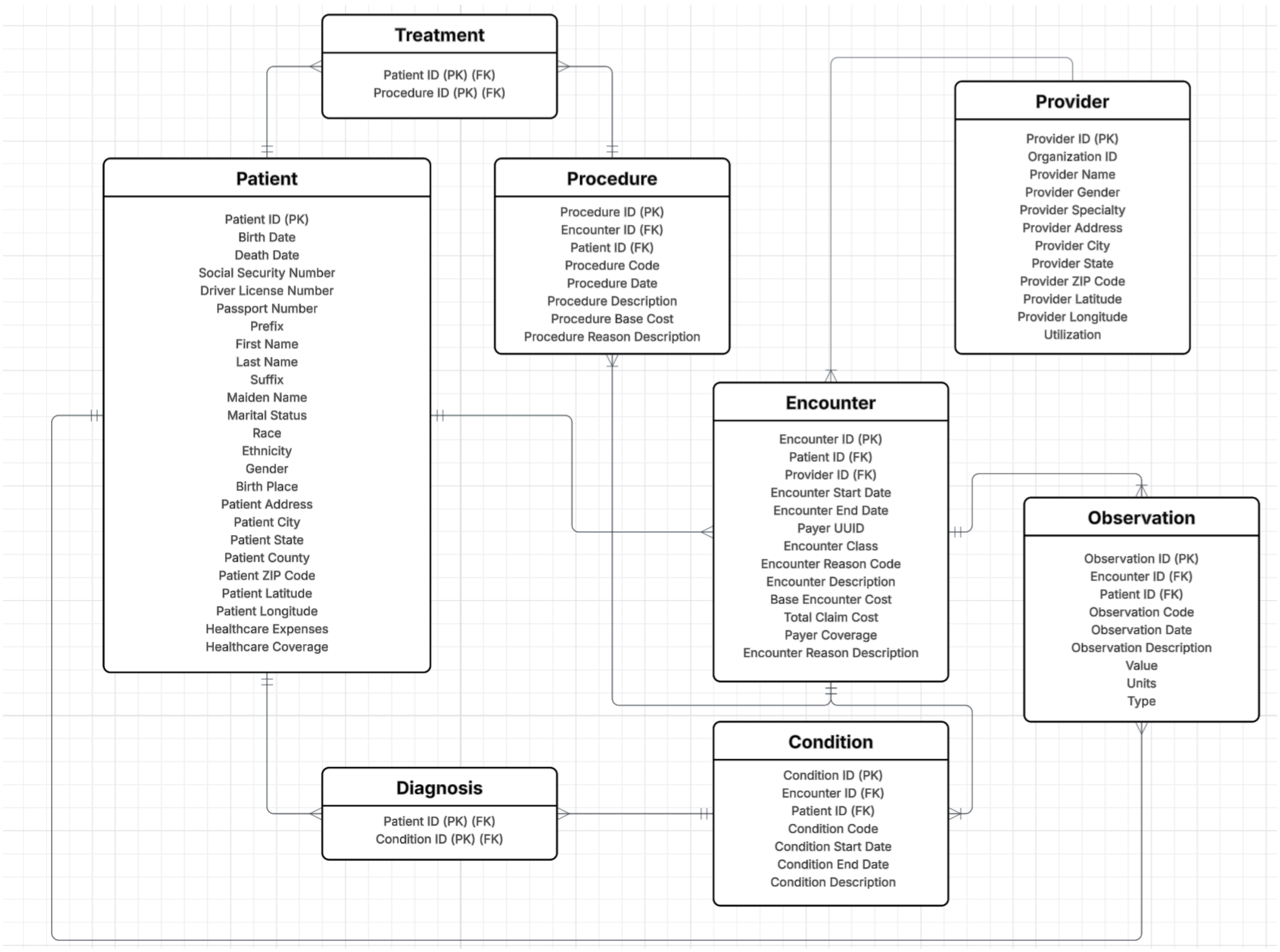
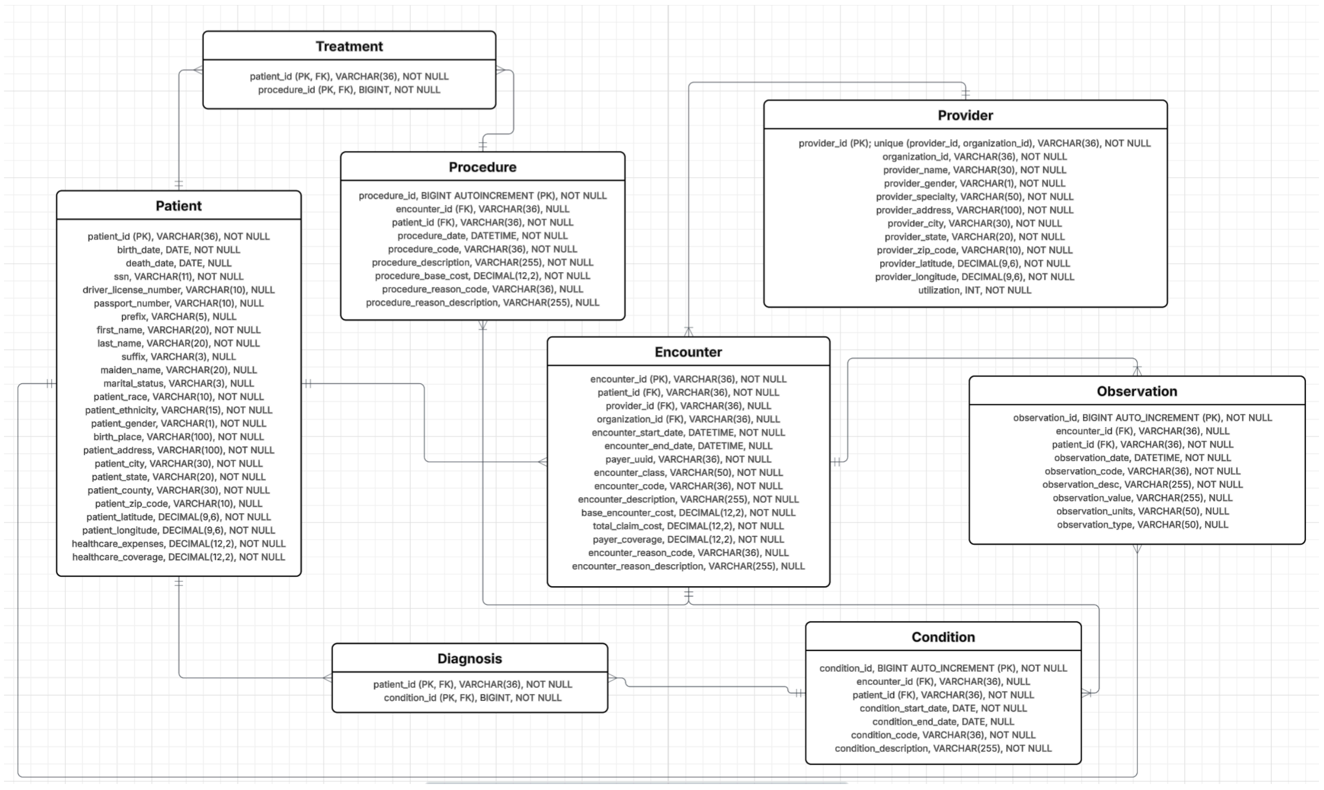


Week 7 Final Project Report: Total Points - 100

Physical Model

5. Develop the physical model based on the Logical Model


Treatment
patient_id (PK, FK), VARCHAR(36), NOT NULL
procedure_id (PK, FK), BIGINT, NOT NULL
Procedure
procedure_id, BIGINT AUTOINCREMENT (PK), NOT NULL
encounter_id (FK), VARCHAR(36), NULL
patient_id (FK), VARCHAR(36), NOT NULL
procedure_date, DATETIME, NOT 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_description, VARCHAR(255), NULL
Provider
provider_id (PK); unique (provider_id, organization_id), VARCHAR(36), NOT NULL
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
Patient
patient_id (PK), VARCHAR(36), NOT NULL
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
Encounter
encounter_id (PK), VARCHAR(36), NOT NULL
patient_id (FK), VARCHAR(36), NOT NULL
provider_id (FK), VARCHAR(36), NULL
organization_id (FK), VARCHAR(36), NULL
encounter_start_date, DATETIME, NOT NULL
encounter_end_date, DATETIME, NULL
payer_uuid, VARCHAR(36), NOT 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_description, VARCHAR(255), NULL
Observation
observation_id, BIGINT AUTO_INCREMENT (PK), NOT NULL
encounter_id (FK), VARCHAR(36), NULL
patient_id (FK), VARCHAR(36), NOT NULL
observation_date, DATETIME, NOT 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
Diagnosis
patient_id (PK, FK), VARCHAR(36), NOT NULL
condition_id (PK, FK), BIGINT, NOT NULL
Condition
condition_id, BIGINT AUTO_INCREMENT (PK), NOT NULL
encounter_id (FK), VARCHAR(36), NULL
patient_id (FK), VARCHAR(36), NOT NULL
condition_start_date, DATE, NOT NULL
condition_end_date, DATE, NULL
condition_code, VARCHAR(36), NOT NULL
condition_description, VARCHAR(255), NOT NULL

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.

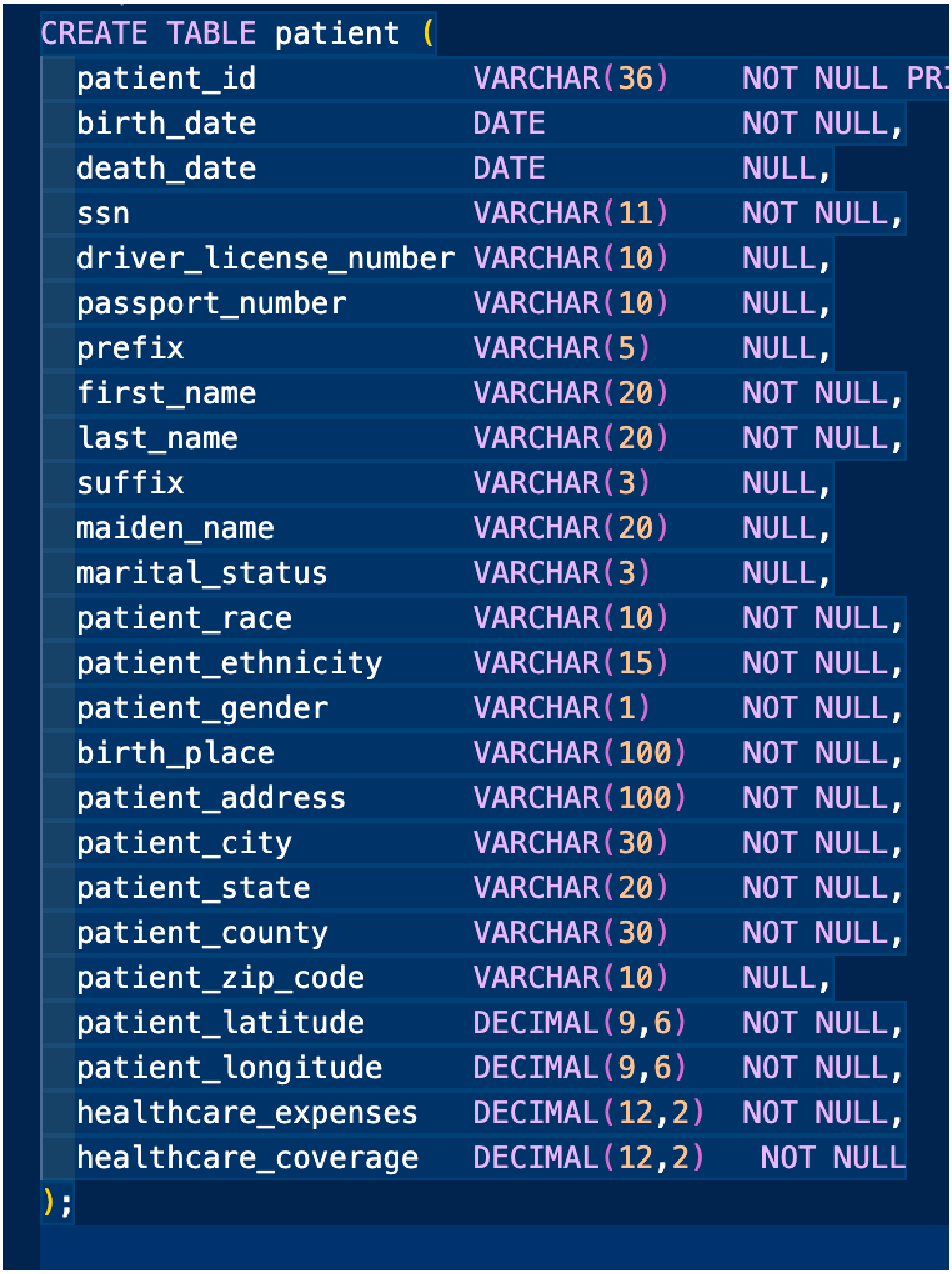
Details: Create the tables that you have come up with (the table must be based on the Physical Model).

- (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.

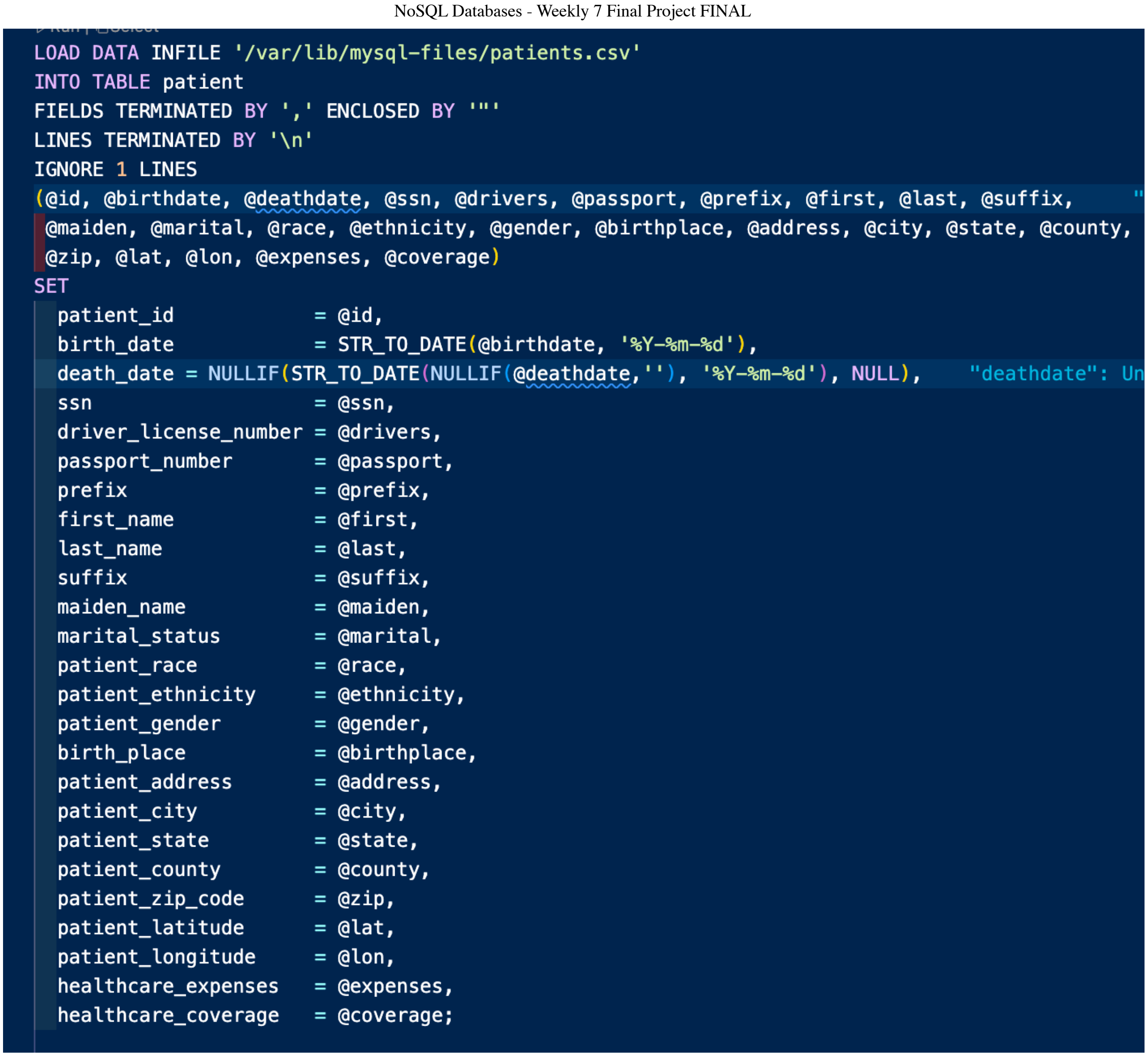
Created Tables

Patient

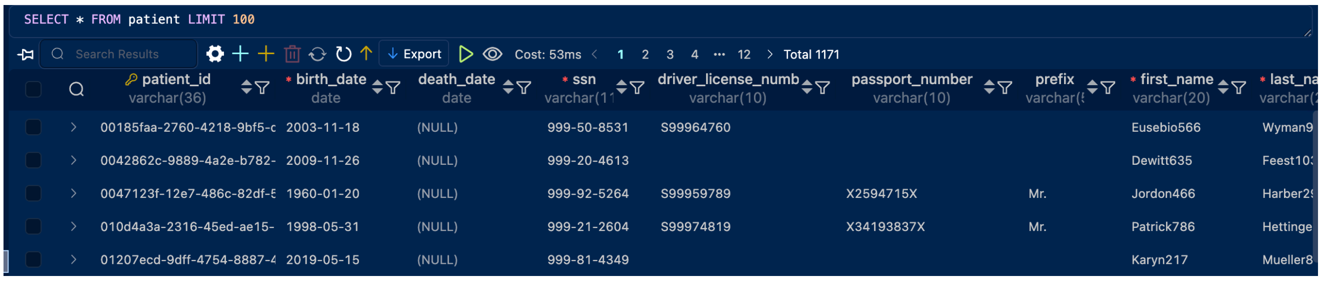
IMPORTANT NOTE: Instead of INSERTING one at a time, I've found a way to import csv files directly into SQL after creating the tables. So, I can bulk insert every single rows and columns by using **LOAD DATA INFILE**. Thankfully, it went well after a few tweaks.



```
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;
```

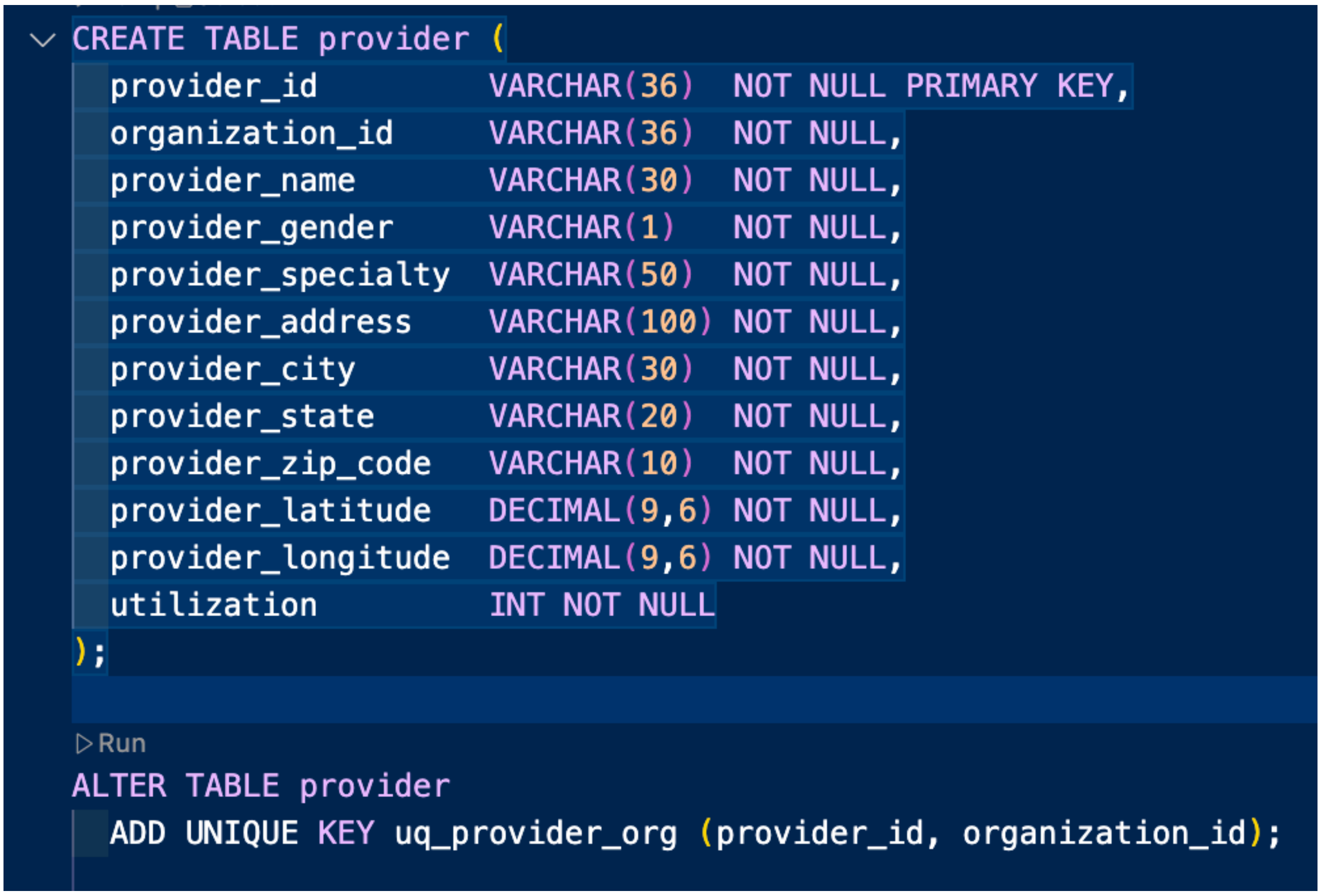


```
SELECT * FROM patient LIMIT 100
```

Cost: 53ms 1 2 3 4 ... 12 Total 1171

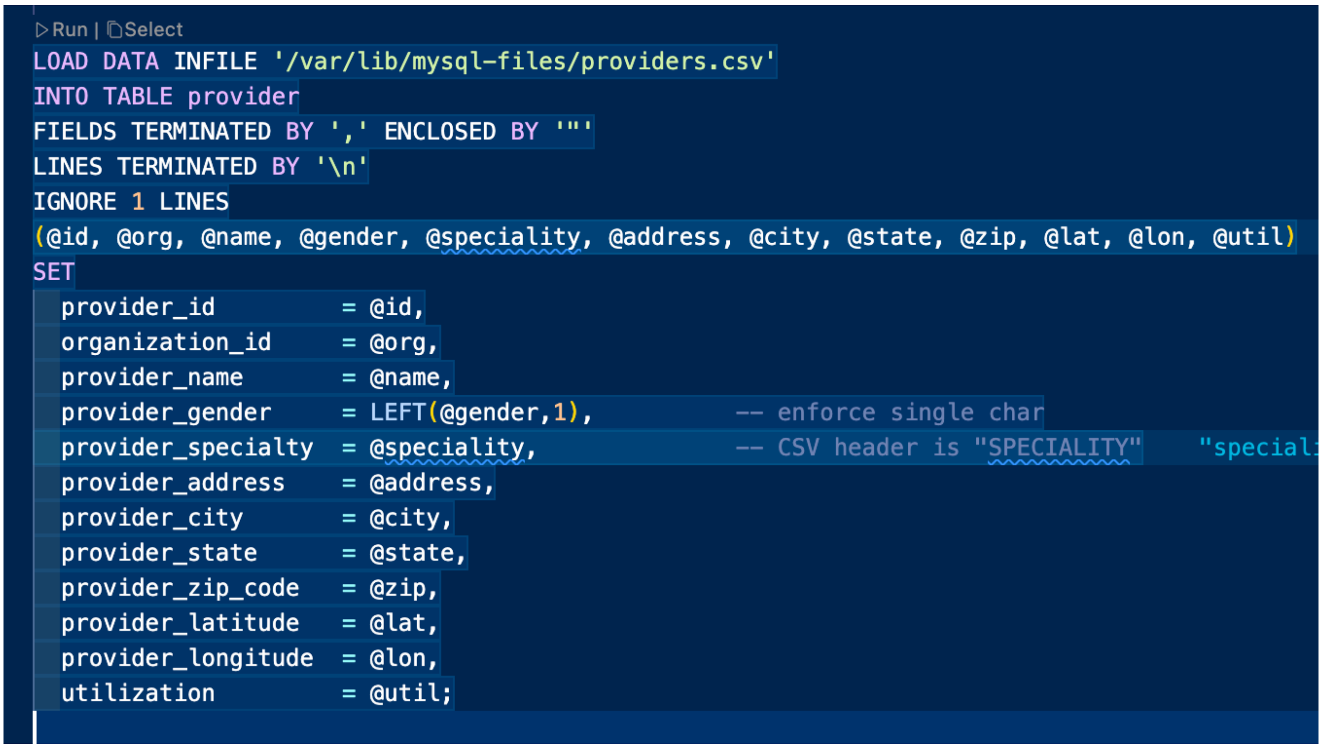
patient_id varchar(36)	* birth_date date	death_date date	* ssn varchar(1	driver_license_numb varchar(10)	passport_number varchar(10)	prefix varchar(	* first_name varchar(20)	* last_na varchar(
00185faa-2760-4218-9bf5-c	2003-11-18	(NULL)	999-50-8531	S99964760			Eusebio566	Wyman9
0042862c-9889-4a2e-b782-	2009-11-26	(NULL)	999-20-4613				Dewitt635	Feest10
0047123f-12e7-486c-82df-5	1960-01-20	(NULL)	999-92-5264	S99959789	X2594715X	Mr.	Jordon466	Harber2
010d4a3a-2316-45ed-ae15-	1998-05-31	(NULL)	999-21-2604	S99974819	X34193837X	Mr.	Patrick786	Hettinge
01207ecd-9dff-4754-8887-4	2019-05-15	(NULL)	999-81-4349				Karyn217	Mueller8

Provider



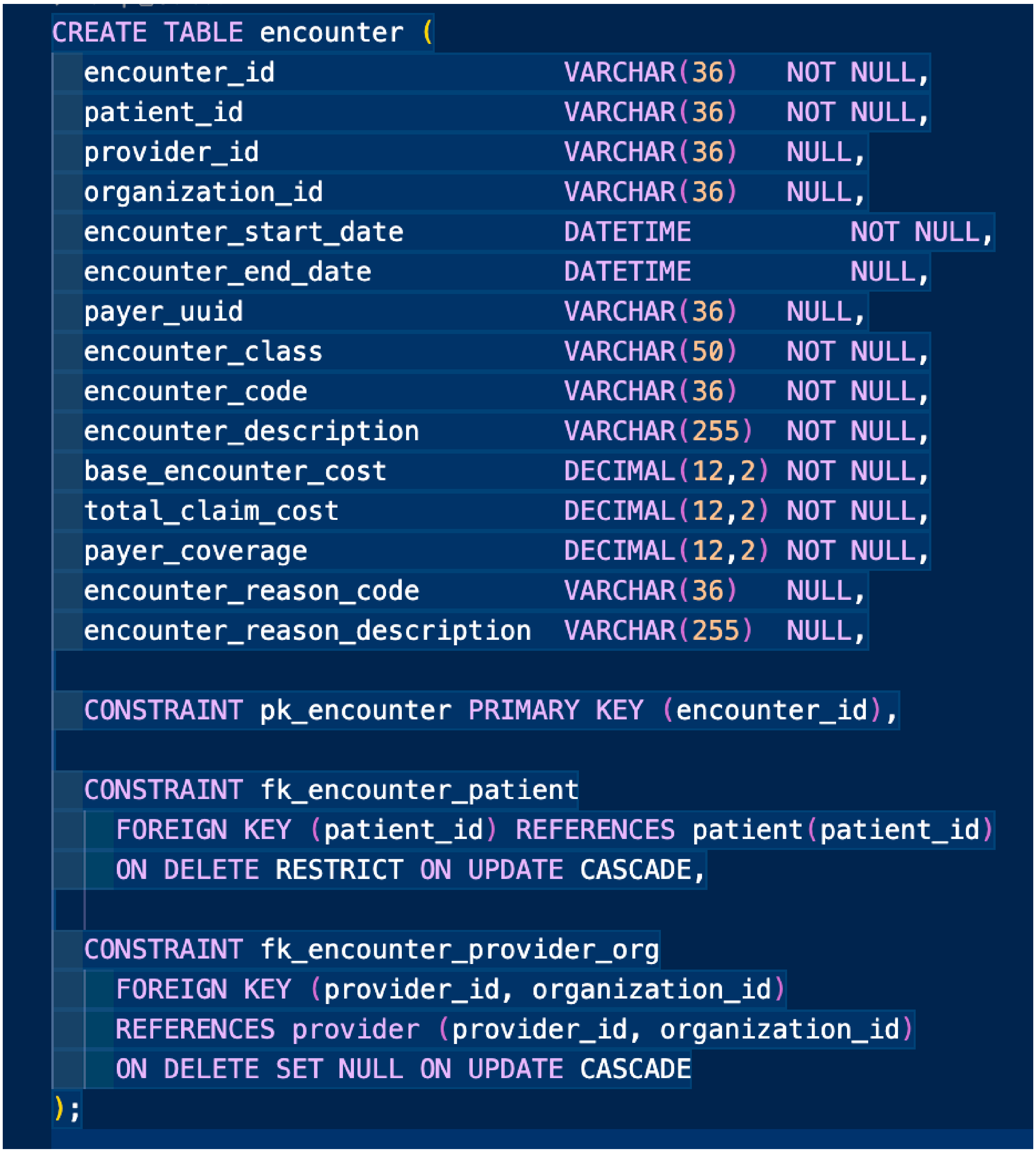
```
CREATE TABLE provider (
    provider_id         VARCHAR(36)  NOT NULL PRIMARY KEY,
    organization_id     VARCHAR(36)  NOT NULL,
    provider_name       VARCHAR(30)  NOT NULL,
    provider_gender     VARCHAR(1)   NOT NULL,
    provider_specialty  VARCHAR(50)  NOT NULL,
    provider_address    VARCHAR(100) 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.

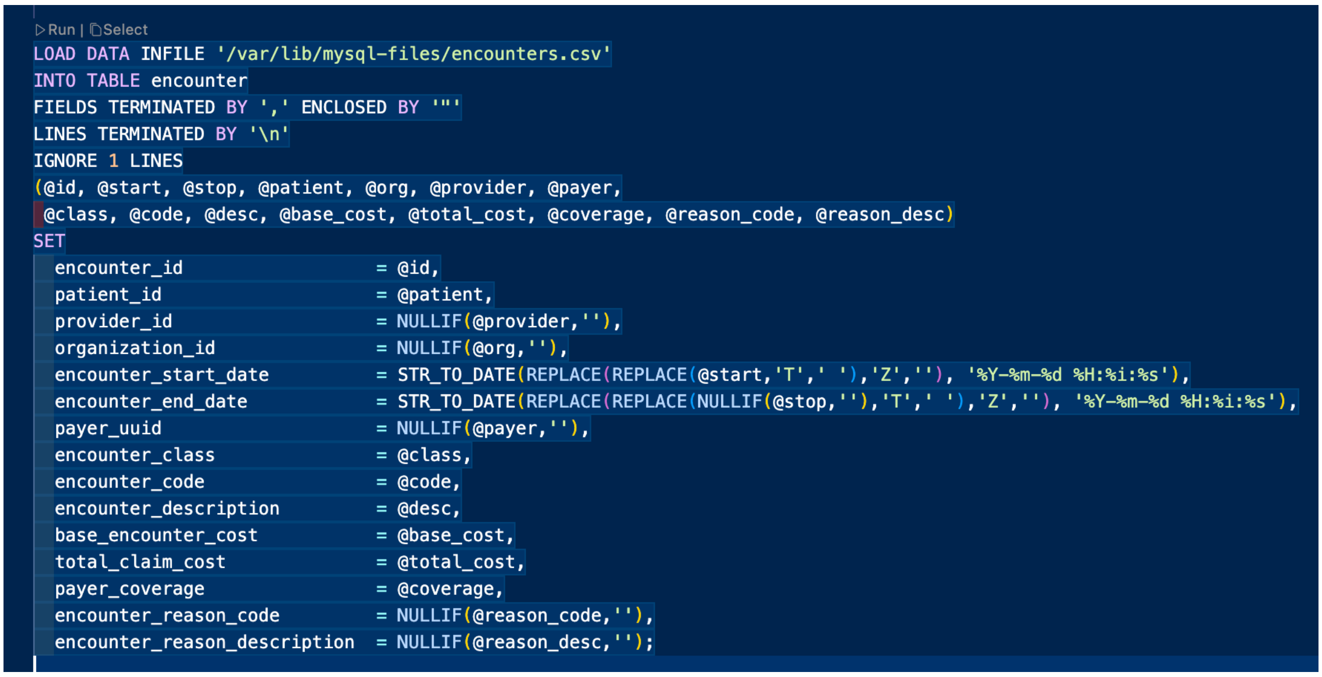


```
▷Run | ⎘Select
LOAD DATA INFILE '/var/lib/mysql-files/providers.csv'
INTO TABLE provider
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(@id, @org, @name, @gender, @speciality, @address, @city, @state, @zip, @lat, @lon, @util)
SET
  provider_id         = @id,
  organization_id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

Encounter

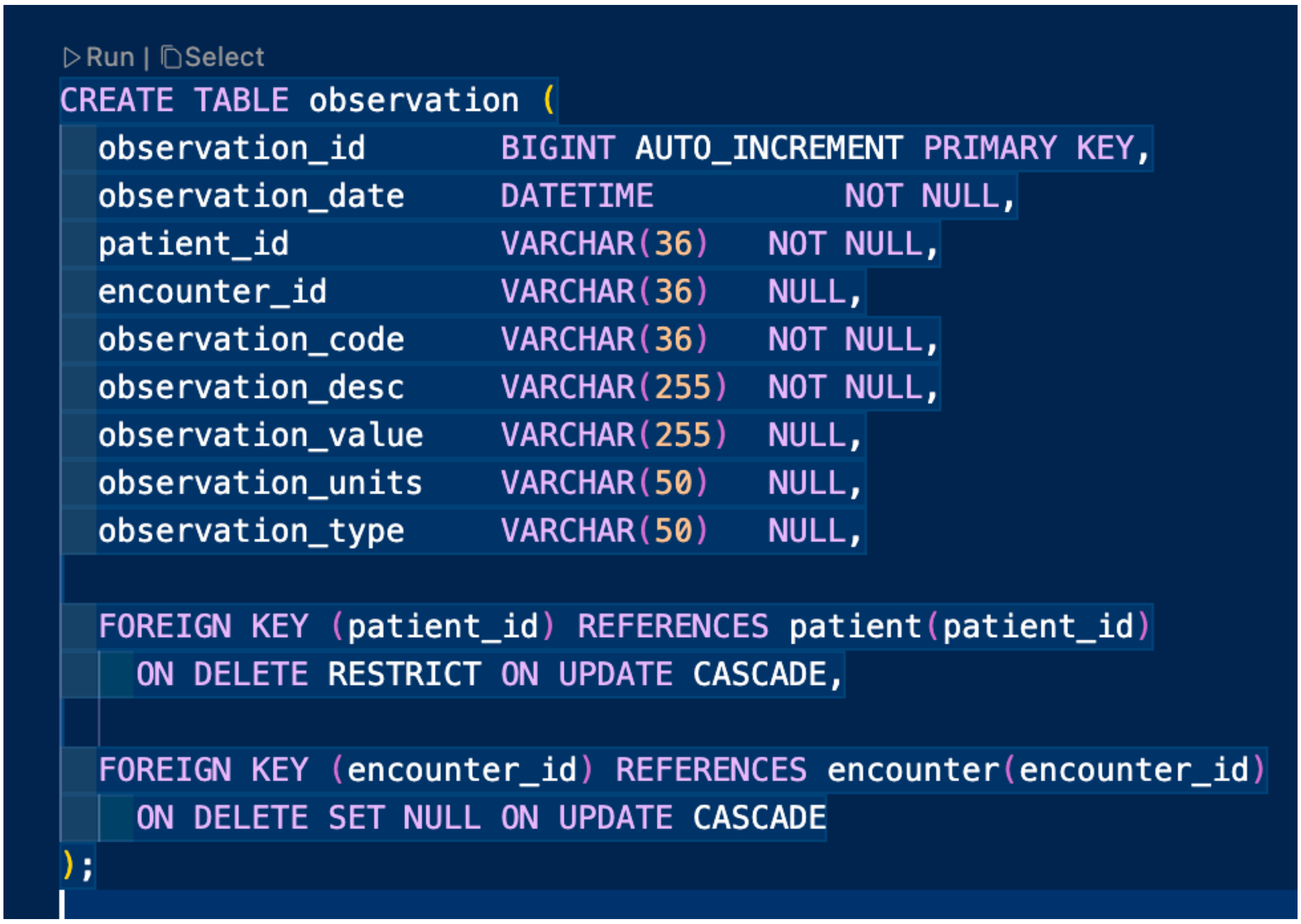


```
CREATE TABLE encounter (
    encounter_id                 VARCHAR(36)    NOT NULL,
    patient_id                   VARCHAR(36)    NOT NULL,
    provider_id                  VARCHAR(36)    NULL,
    organization_id              VARCHAR(36)    NULL,
    encounter_start_date         DATETIME           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_description 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
);
```

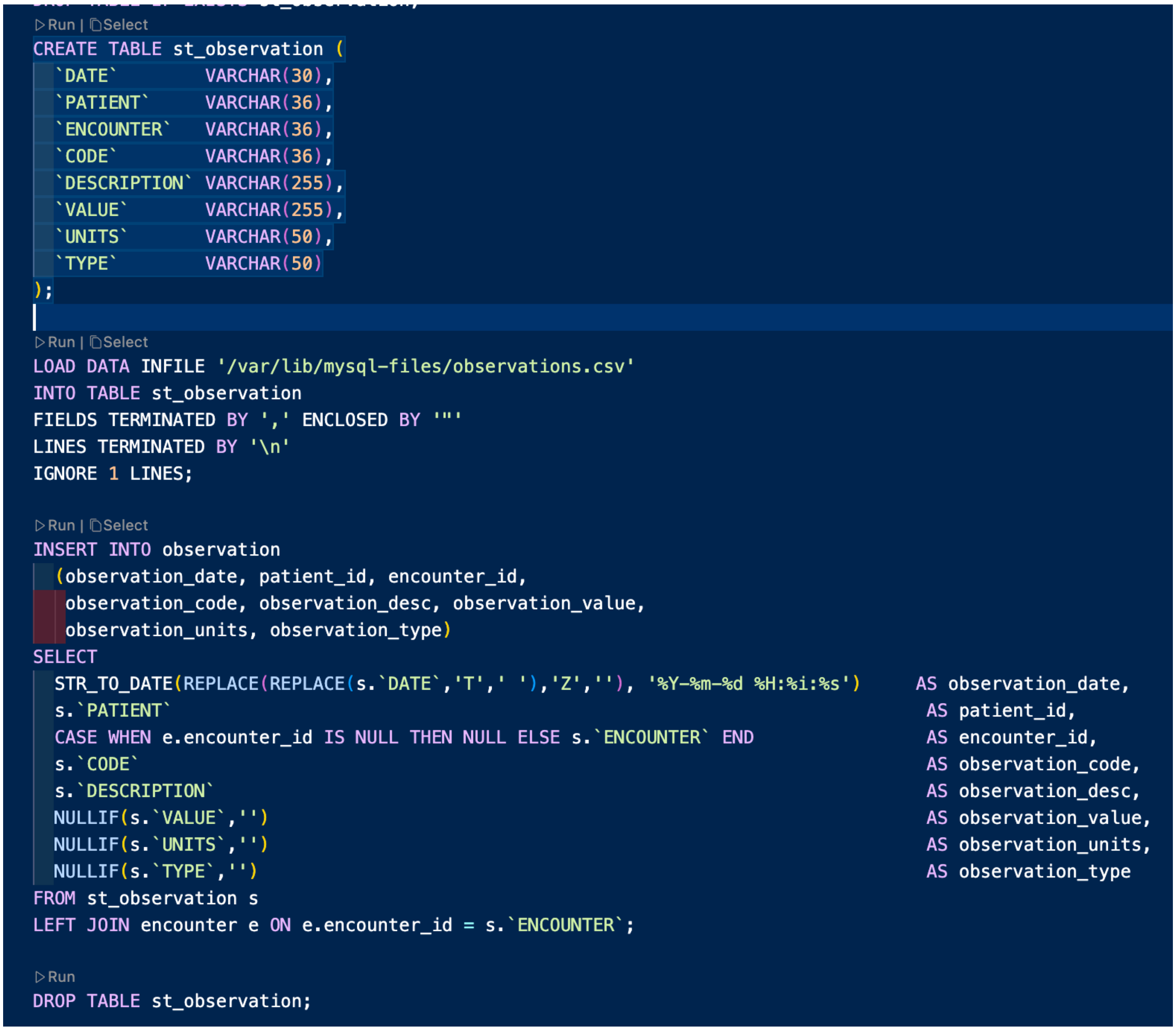


```
▷Run | ⧉Select
LOAD DATA INFILE '/var/lib/mysql-files/encounters.csv'
INTO TABLE encounter
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(@id, @start, @stop, @patient, @org, @provider, @payer,
 @class, @code, @desc, @base_cost, @total_cost, @coverage, @reason_code, @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

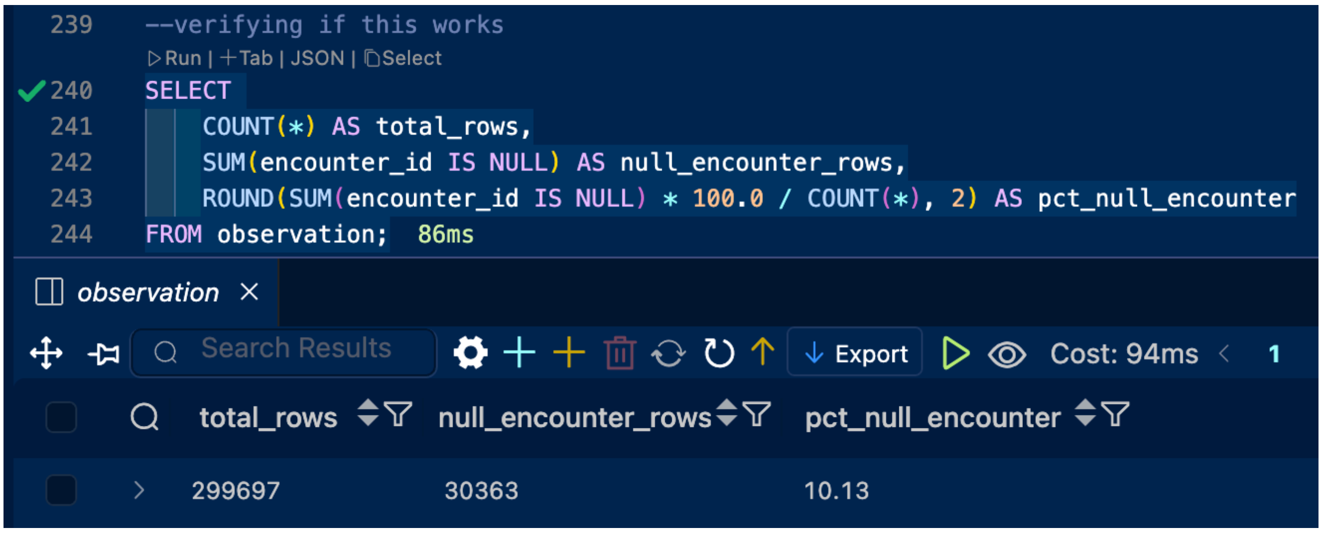


```
▷Run | ⧉Select
CREATE TABLE observation (
  observation_id      BIGINT AUTO_INCREMENT PRIMARY KEY,
  observation_date    DATETIME          NOT NULL,
  patient_id          VARCHAR(36)   NOT NULL,
  encounter_id        VARCHAR(36)   NULL,
  observation_code    VARCHAR(36)   NOT NULL,
  observation_desc    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
);
```



```sql
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;
```

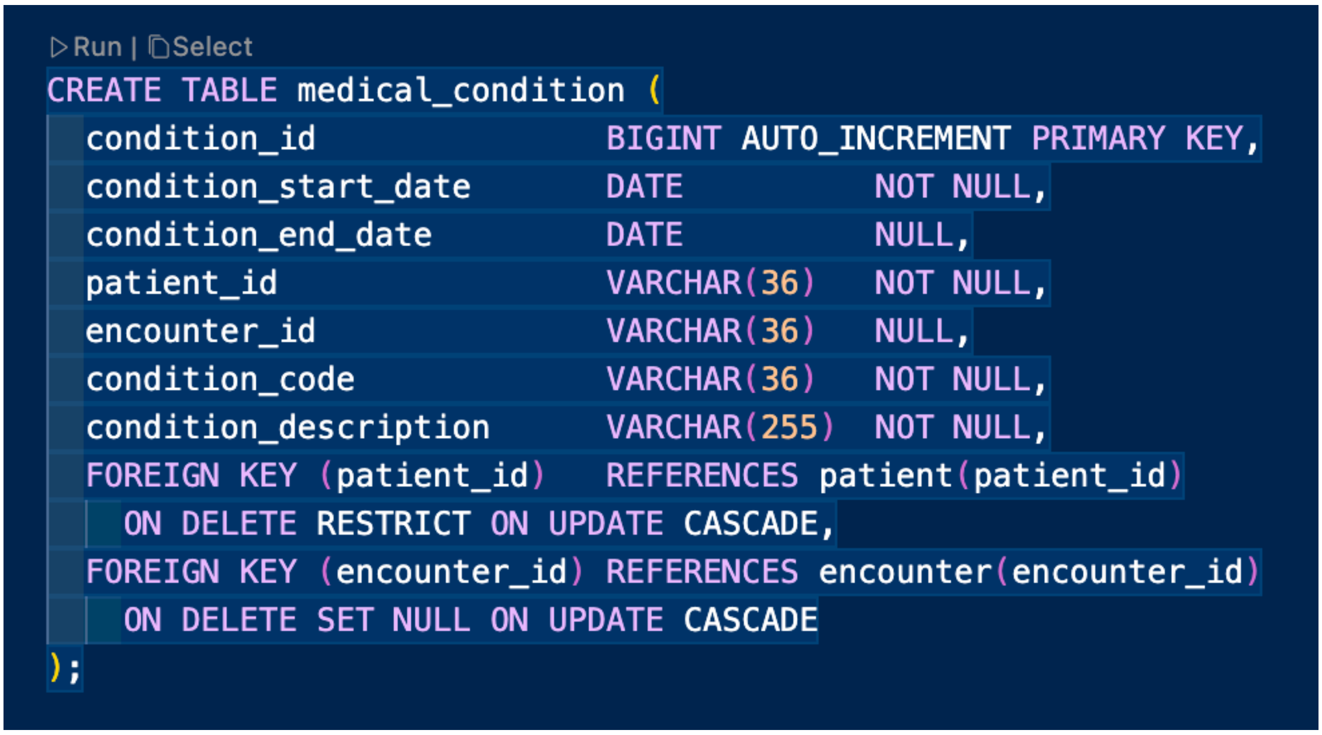
Note: I created this table st_observation, because it allows me to load the whole data into that staging table. Then, I INSERT and SELECT staged observation table into observation (intended) table WHILE LEFT JOIN on encounter_id from encounter table. Any found mismatched encounter_id will be inserted 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.



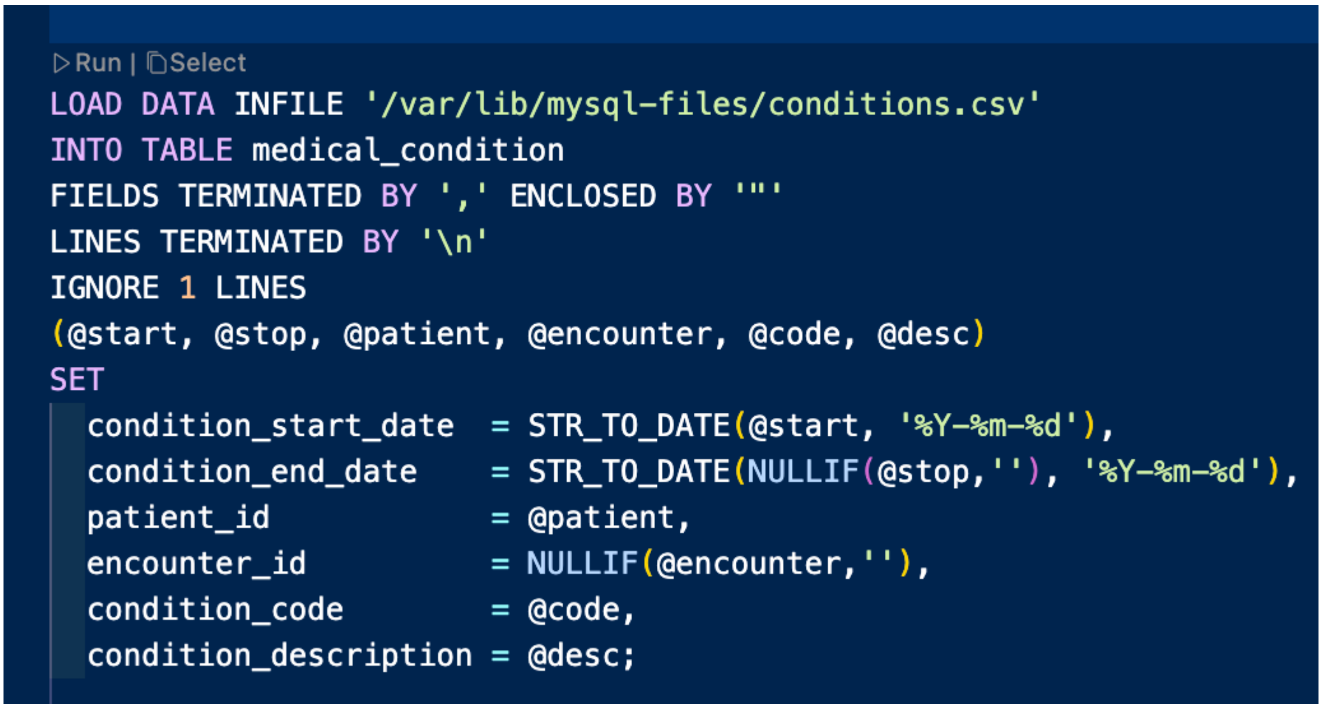
```sql
--verifying if this works
SELECT
    COUNT(*) AS total_rows,
    SUM(encounter_id IS NULL) AS null_encounter_rows,
    ROUND(SUM(encounter_id IS NULL) * 100.0 / COUNT(*), 2) AS pct_null_encounter
FROM observation;  86ms
```

```
total_rows    null_encounter_rows    pct_null_encounter
299697        30363                  10.13
```

Condition

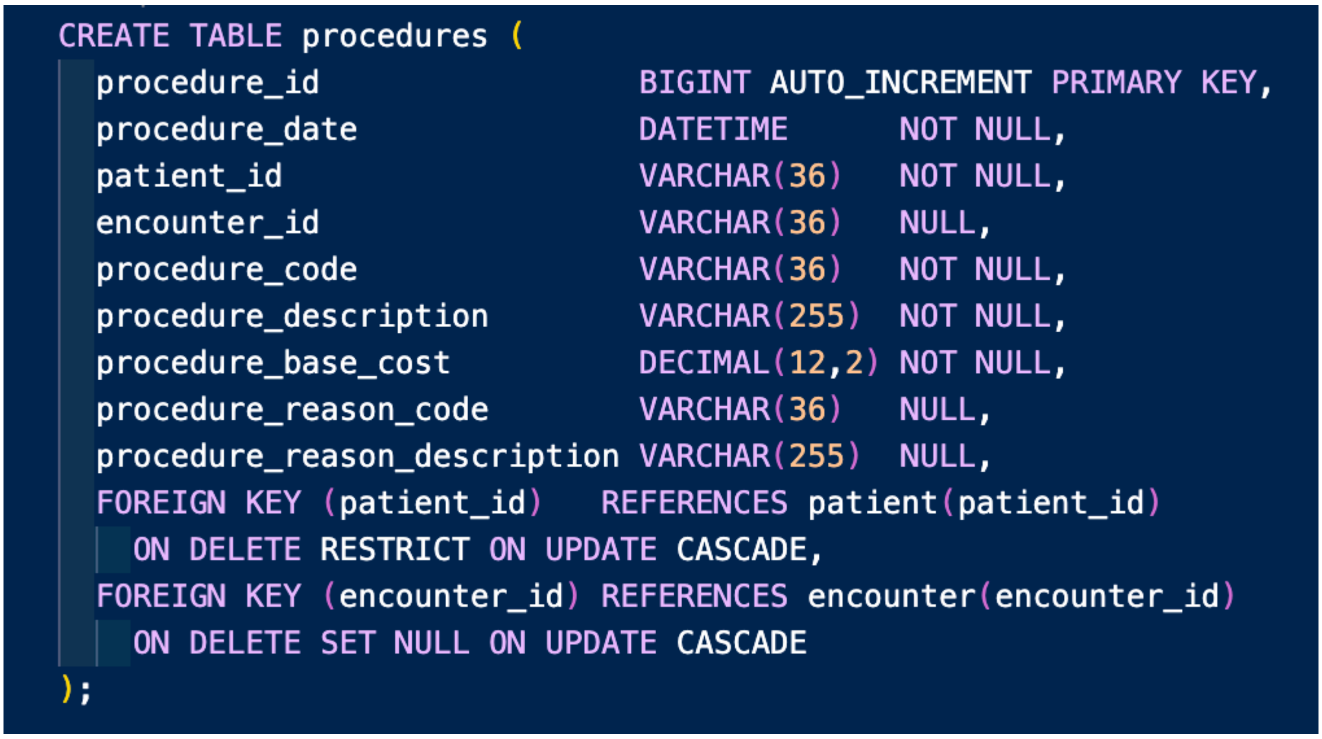


```sql
CREATE TABLE medical_condition (
    condition_id                BIGINT AUTO_INCREMENT PRIMARY KEY,
    condition_start_date        DATE          NOT NULL,
    condition_end_date          DATE          NULL,
    patient_id                  VARCHAR(36)   NOT NULL,
    encounter_id                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
);
```

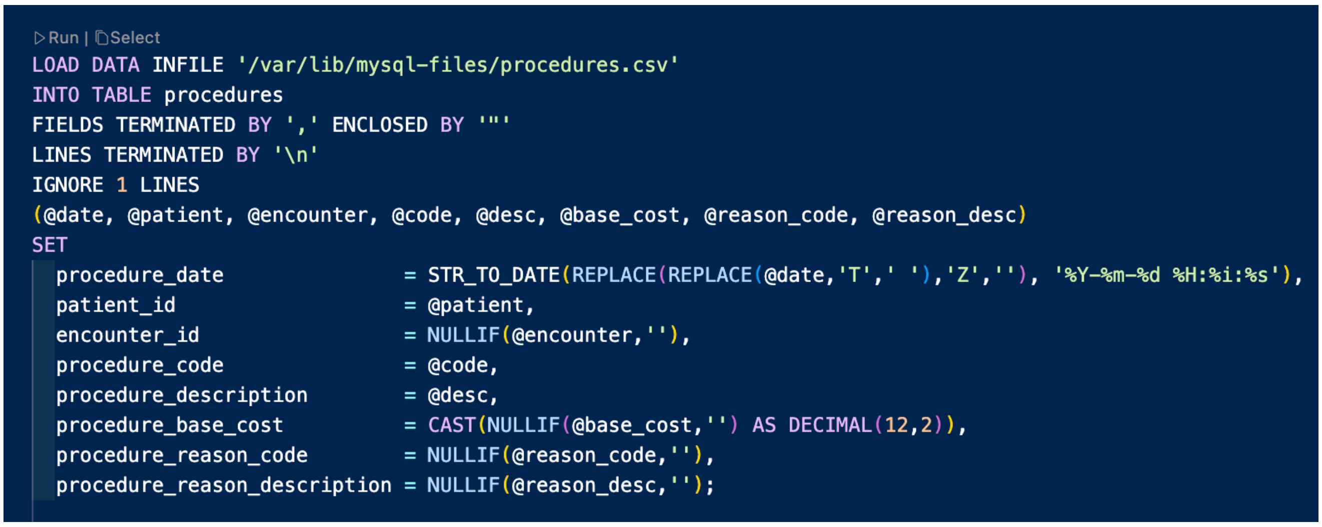


```sql
LOAD DATA INFILE '/var/lib/mysql-files/conditions.csv'
INTO TABLE medical_condition
FIELDS TERMINATED BY ',' ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(@start, @stop, @patient, @encounter, @code, @desc)
SET
  condition_start_date  = STR_TO_DATE(@start, '%Y-%m-%d'),
  condition_end_date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

Procedure

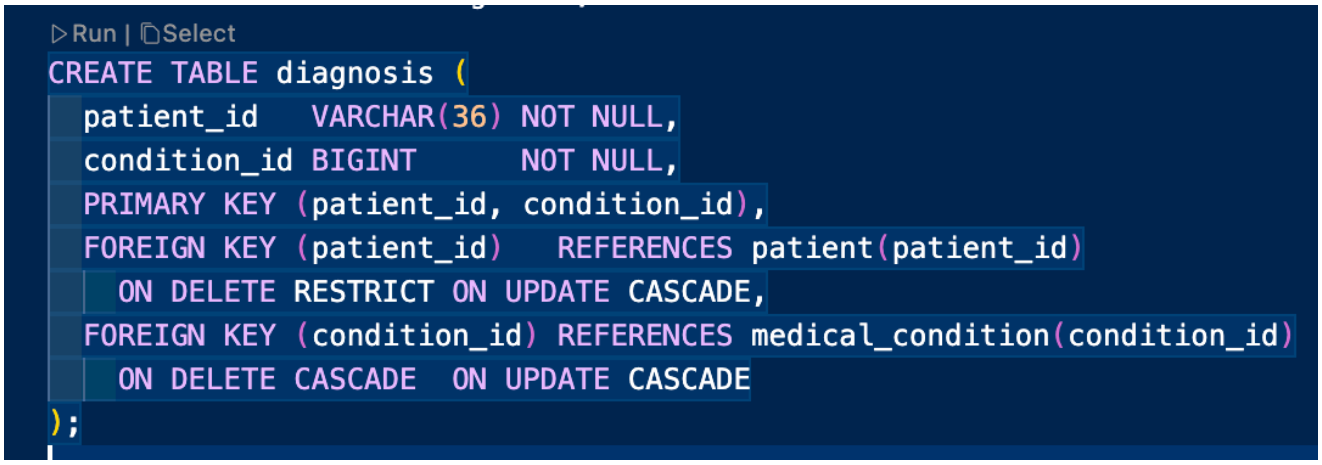


```sql
CREATE TABLE procedures (
  procedure_id                  BIGINT AUTO_INCREMENT PRIMARY KEY,
  procedure_date                DATETIME      NOT NULL,
  patient_id                    VARCHAR(36)   NOT NULL,
  encounter_id                  VARCHAR(36)   NULL,
  procedure_code                VARCHAR(36)   NOT NULL,
  procedure_description         VARCHAR(255)  NOT NULL,
  procedure_base_cost           DECIMAL(12,2) NOT NULL,
  procedure_reason_code         VARCHAR(36)   NULL,
  procedure_reason_description  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
);
```



```sql
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_description = NULLIF(@reason_desc,'');
```

Junction Tables



```sql
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
);
```