Project, 911

Project Documentation

Project, 911

Project Documentation

Project topic

911 is a database project with a main aim to cover an emergency quick response system in which has 3 parts: Healthcare, Fire Department & Emergency Reception Department (Police).

Table Of Contents

01

Healthcare

Patient records and medical history.

02

Fire Department

Fire incidents & firefighters records

03

Quick Response

Incident management system

04

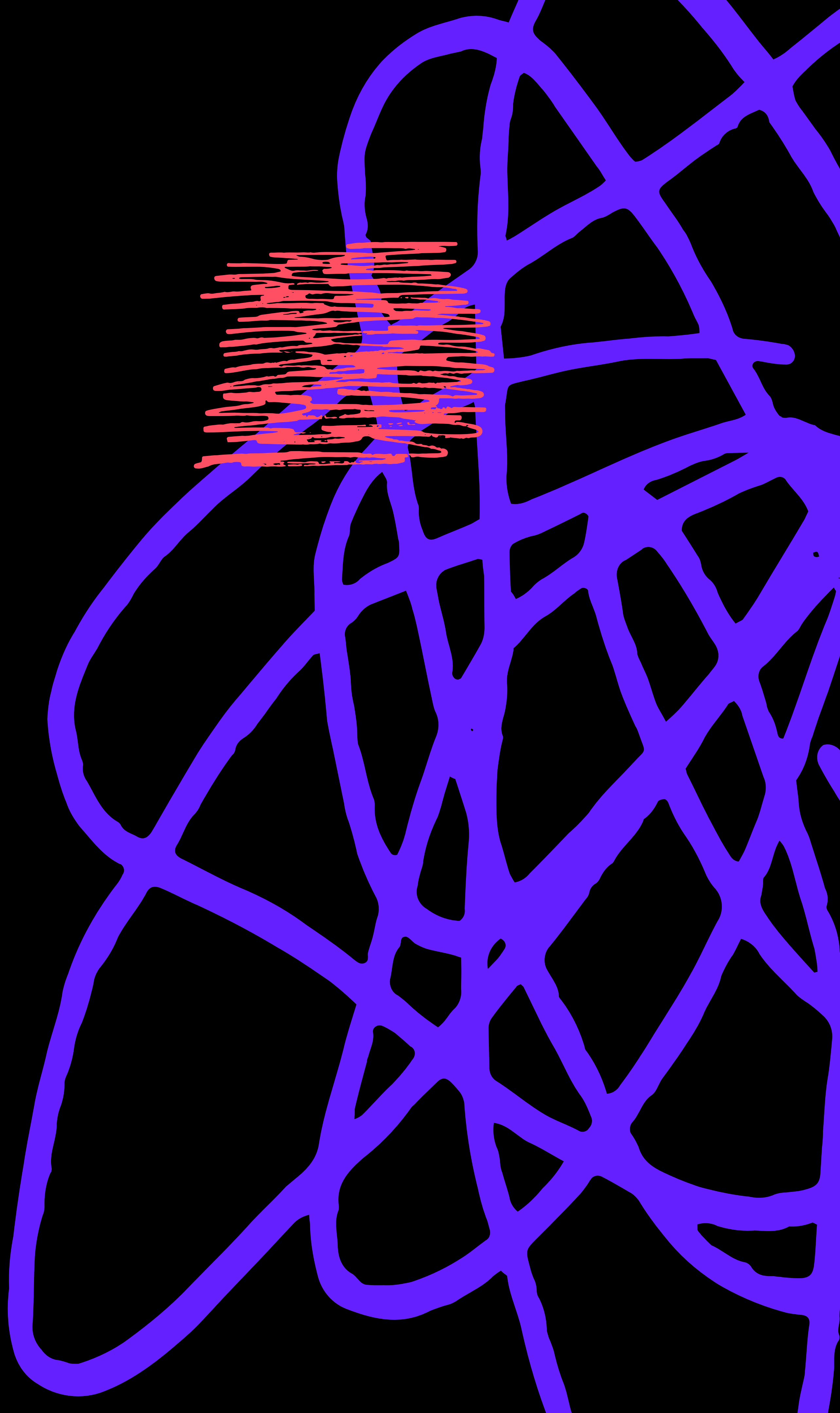
Inserts & Queries

Main Queries sum up

01

Healthcare

The part that involves patient records and medical history.



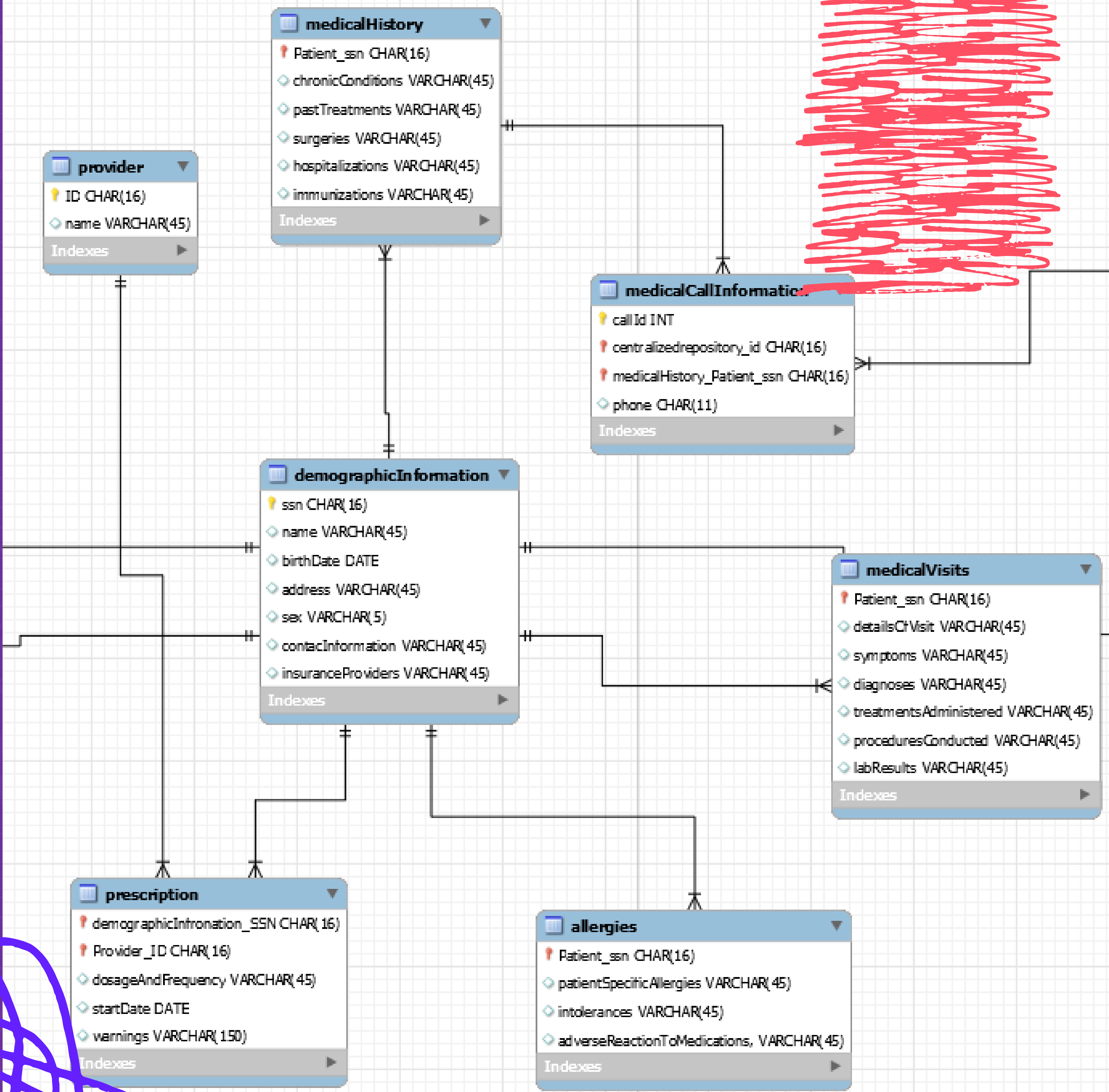
Functionality

Patient Records: Detailed information about patients, including personal details, medical history, allergies, and prescribed medications.

Appointments: Records of scheduled appointments for patients, with associated doctors and timings.

Medical History: Chronic conditions, past treatments, surgeries, and ongoing medical conditions.

SOURCE: 911 Project Brief



[redacted]

medicalHistory
- Patient_ssn CHAR(16)
- chronicConditions VARCHAR(45)
- pastTreatments VARCHAR(45)
- surgeries VARCHAR(45)
- hospitalizations VARCHAR(45)
- immunizations VARCHAR(45)

Indexes

provider
- ID CHAR(16)
- name VARCHAR(45)

Indexes

medicalCallInformatic
- callId INT
- centralizedrepository_id CHAR(16)
- medicalHistory_Patient_ssn CHAR(16)
- phone CHAR(11)

Indexes

demographicInformation
- ssn CHAR(16)
- name VARCHAR(45)
- birthDate DATE
- address VARCHAR(45)
- sex VARCHAR(5)
- contacInformation VARCHAR(45)
- insuranceProviders VARCHAR(45)

Indexes

medicalVisits
- Patient_ssn CHAR(16)
- detailsOfVisit VARCHAR(45)
- symptoms VARCHAR(45)
- diagnoses VARCHAR(45)
- treatmentsAdministered VARCHAR(45)
- proceduresConducted VARCHAR(45)
- labResults VARCHAR(45)

Indexes

prescription
- demographicInfronation_SSN CHAR(16)
- Provider_ID CHAR(16)
- dosageAndFrequency VARCHAR(45)
- startDate DATE
- warnings VARCHAR(150)

Indexes

allergies
- Patient_ssn CHAR(16)
- patientSpecificAllergies VARCHAR(45)
- intolerances VARCHAR(45)
- adverseReactionToMedications, VARCHAR(45)

Indexes

02

Fire Department

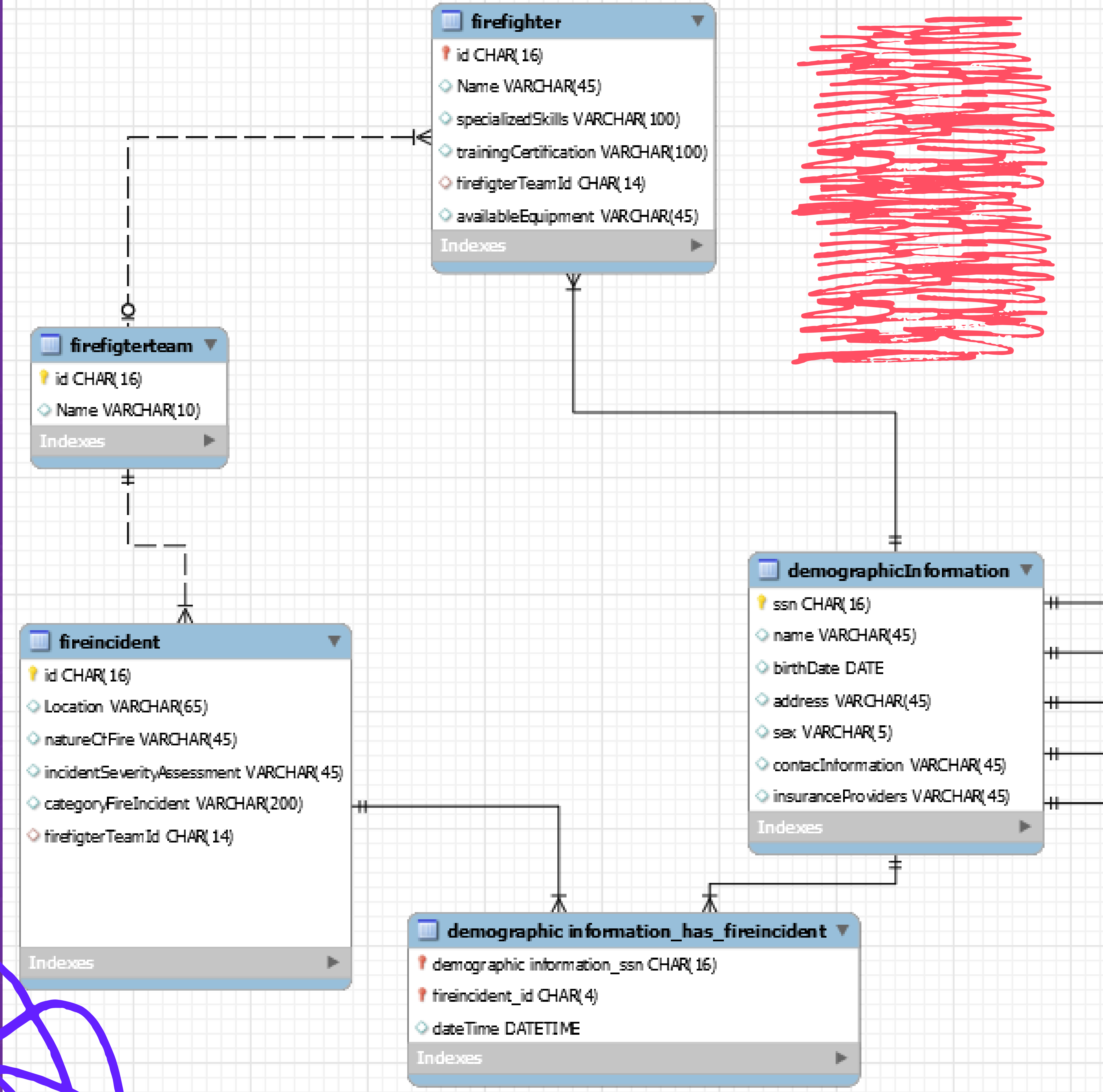
The part that involves fire incidents & firefighters records .

Functionality

Fire Incidents: Details about fire-related incidents, their locations, severity, and types of equipment required.

Firefighter Details: Information about firefighters, their teams, available equipment, and their proximity to reported incidents.

SOURCE: 911 Project Brief



firefighter
- id CHAR(16)
- Name VARCHAR(45)
- specializedSkills VARCHAR(100)
- trainingCertification VARCHAR(100)
- firefigterTeamId CHAR(14)
- availableEquipment VARCHAR(45)
- Indexes

[redacted]

firefigterteam
- id CHAR(16)
- Name VARCHAR(10)
- Indexes

fireincident
- id CHAR(16)
- Location VARCHAR(65)
- natureOfFire VARCHAR(45)
- incidentSeverityAssessment VARCHAR(45)
- categoryFireIncident VARCHAR(200)
- firefigterTeamId CHAR(14)
- Indexes

demographicInformation
- ssn CHAR(16)
- name VARCHAR(45)
- birthDate DATE
- address VARCHAR(45)
- sex VARCHAR(5)
- contacInformation VARCHAR(45)
- insuranceProviders VARCHAR(45)
- Indexes

demographic information_has_fireincident
- demographic information_ssn CHAR(16)
- fireincident_id CHAR(4)
- dateTime DATETIME
- Indexes

03

Quick Response

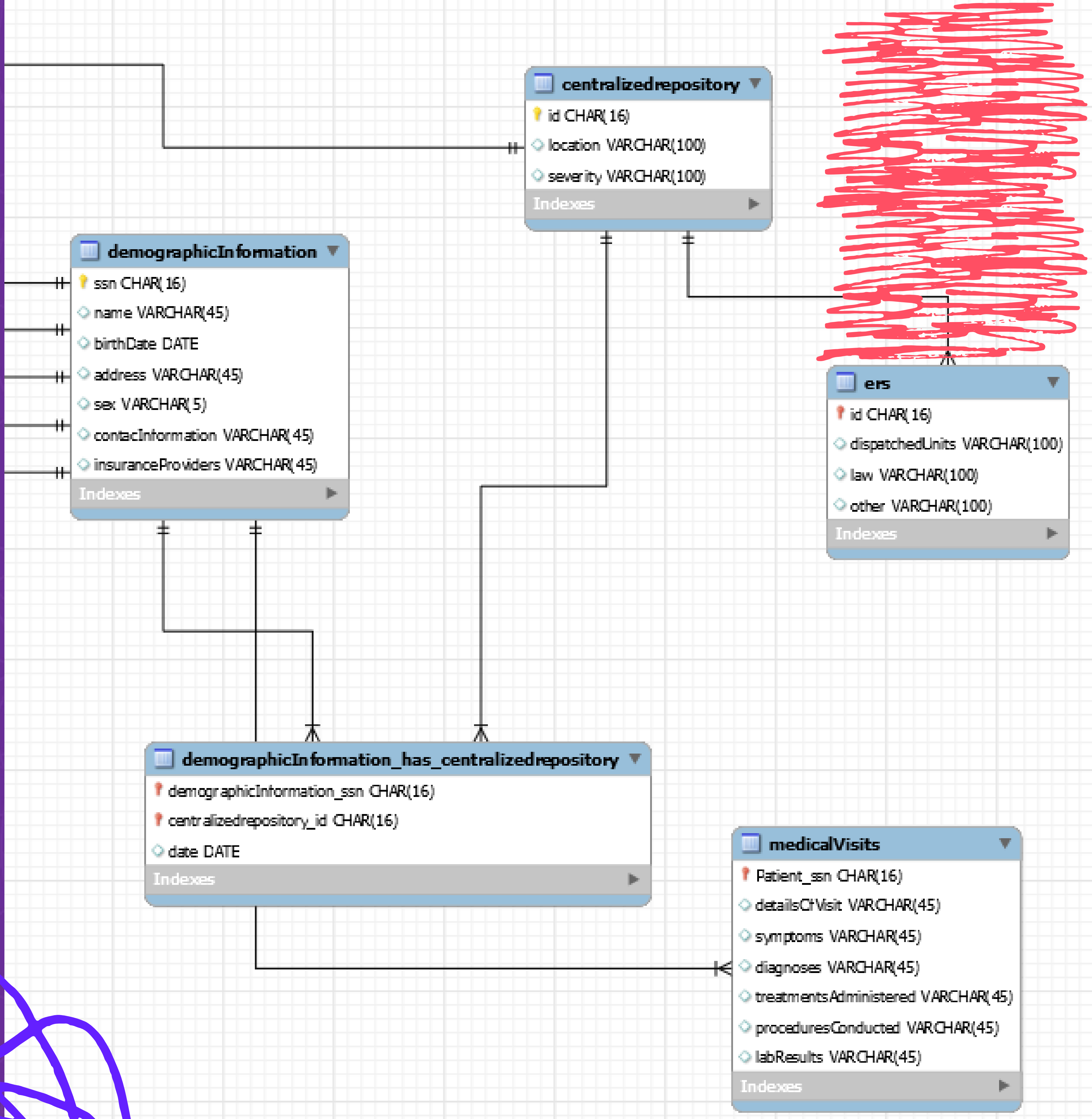
Our incident management system.

Functionality

Call Reception and Logging: A system to log and track incoming emergency calls, including caller details, time of call, and brief descriptions of the emergencies reported.

Unique Incident Identifiers: Creating unique identifiers for each reported incident for efficient tracking.

SOURCE: 911 Project Brief



centralizedrepository
- id CHAR(16)
- location VARCHAR(100)
- severity VARCHAR(100)

Indexes

[redacted]

demographicInformation
- ssn CHAR(16)
- name VARCHAR(45)
- birthDate DATE
- address VARCHAR(45)
- sex VARCHAR(5)
- contacInformation VARCHAR(45)
- insuranceProviders VARCHAR(45)

Indexes

ers
- id CHAR(16)
- dispatchedUnits VARCHAR(100)
- law VARCHAR(100)
- other VARCHAR(100)

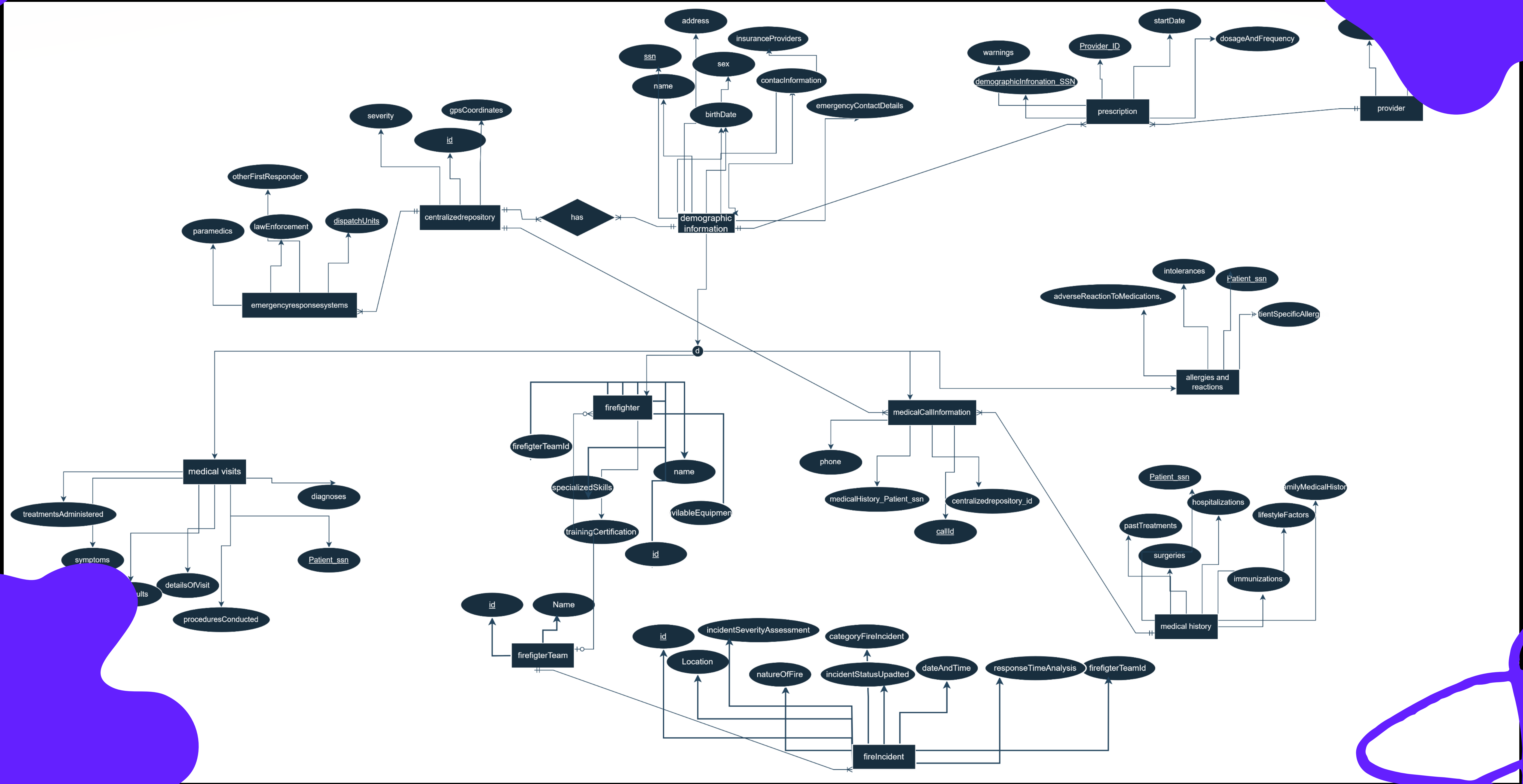
Indexes

demographicInformation_has_centralizedrepository
- demographicInformation_ssn CHAR(16)
- centralizedrepository_id CHAR(16)
- date DATE

Indexes

medicalVisits
- Patient_ssn CHAR(16)
- detailsOfVisit VARCHAR(45)
- symptoms VARCHAR(45)
- diagnoses VARCHAR(45)
- treatmentsAdministered VARCHAR(45)
- proceduresConducted VARCHAR(45)
- labResults VARCHAR(45)

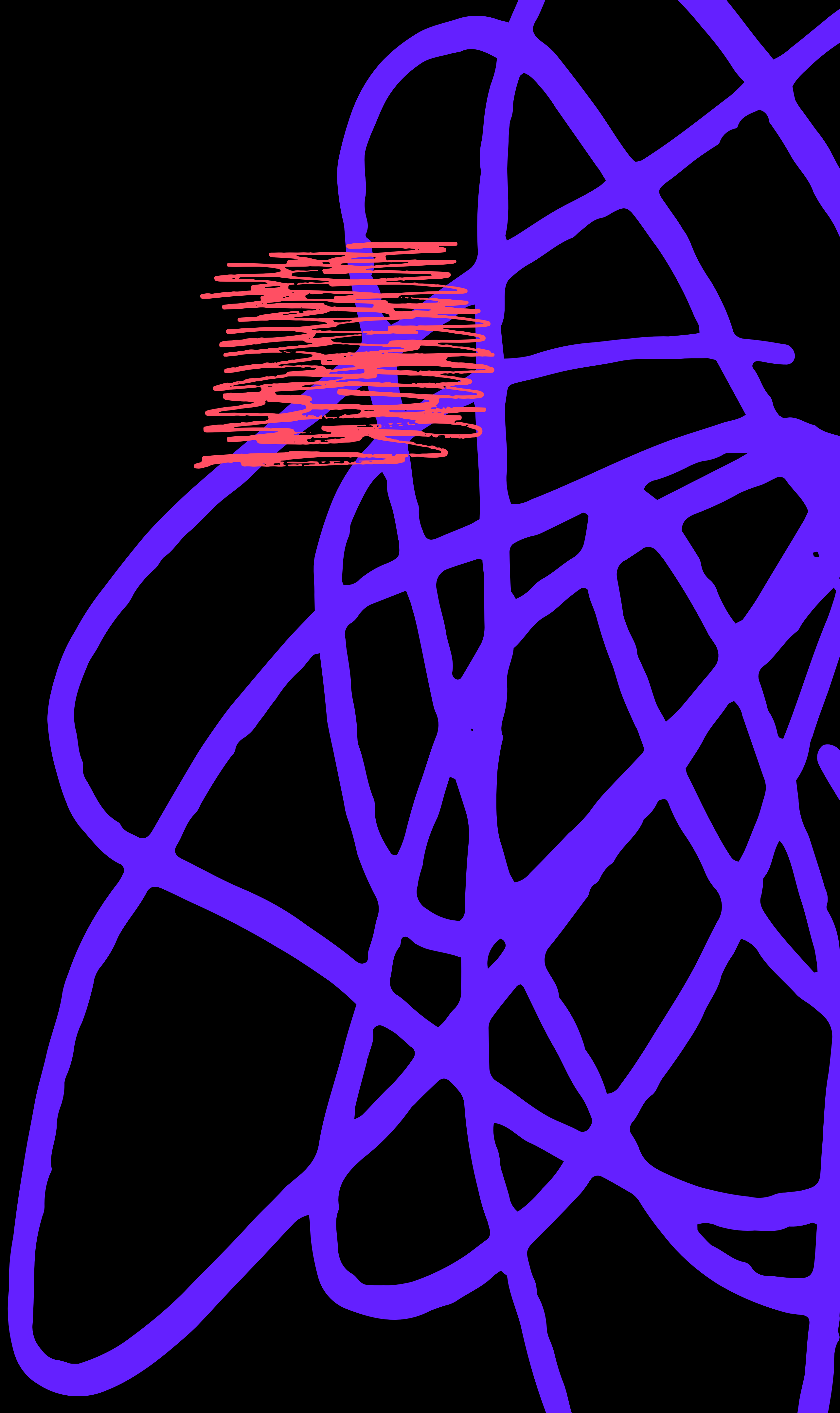
Indexes


address
insuranceProviders
ssn
sex
name
contacInformation
birthDate
emergencyContactDetails
severity
gpsCoordinates
id
otherFirstResponder
paramedics
lawEnforcement
dispatchUnits
centralizedrepository
has
demographic information
emergencyresponsesystems
warnings
Provider_ID
startDate
dosageAndFrequency
demographicInfronation_SSN
prescription
provider
intolerances
Patient_ssn
adverseReactionToMedications,
tientSpecificAllerg
allergies and reactions
d
firefighter
firefigterTeamId
name
specializedSkills
vilableEquipmen
trainingCertification
id
medicalCallInformation
phone
medicalHistory_Patient_ssn
centralizedrepository_id
callId
medical visits
diagnoses
treatmentsAdministered
symptoms
Patient_ssn
ults
detailsOfVisit
proceduresConducted
Patient_ssn
hospitalizations
amilyMedicalHistor
pastTreatments
lifestyleFactors
surgeries
immunizations
medical history
id
Name
firefigterTeam
id
incidentSeverityAssessment
categoryFireIncident
Location
natureOfFire
incidentStatusUpadted
dateAndTime
responseTimeAnalysis
firefigterTeamId
fireIncident

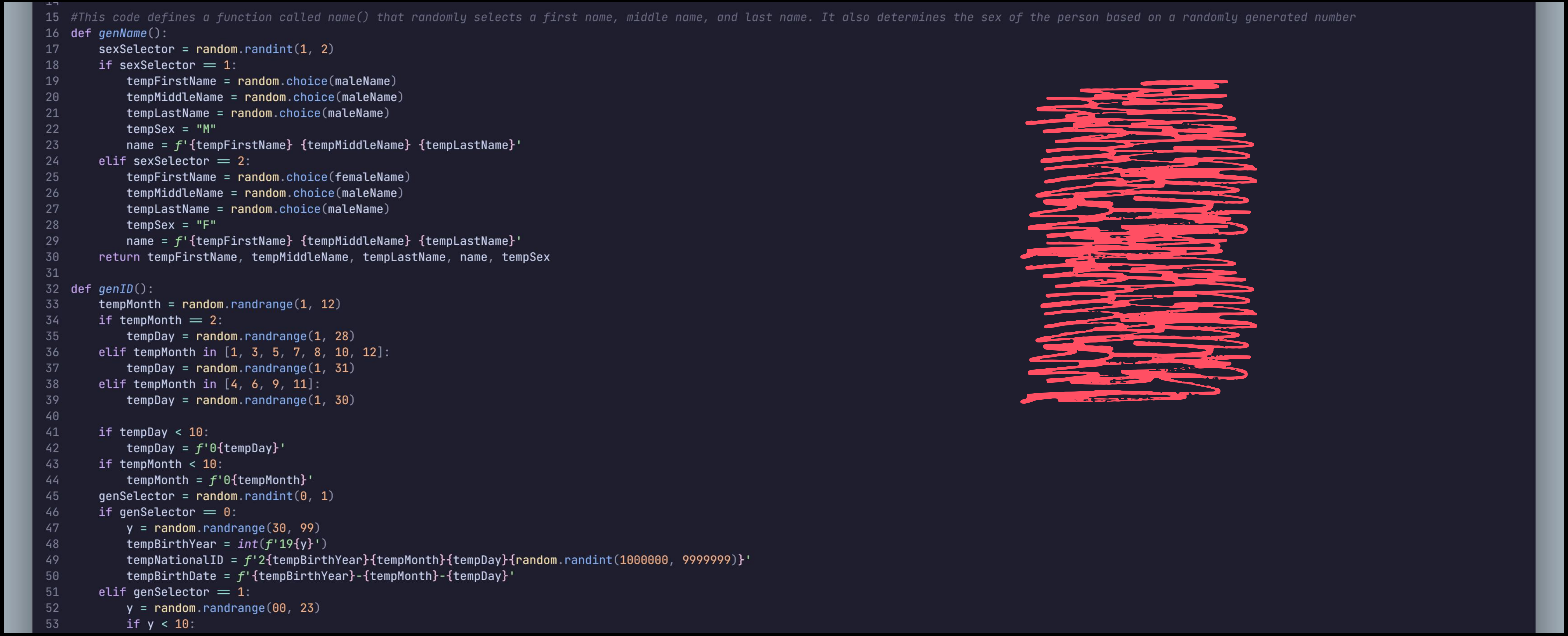
04

Inserts & Queries

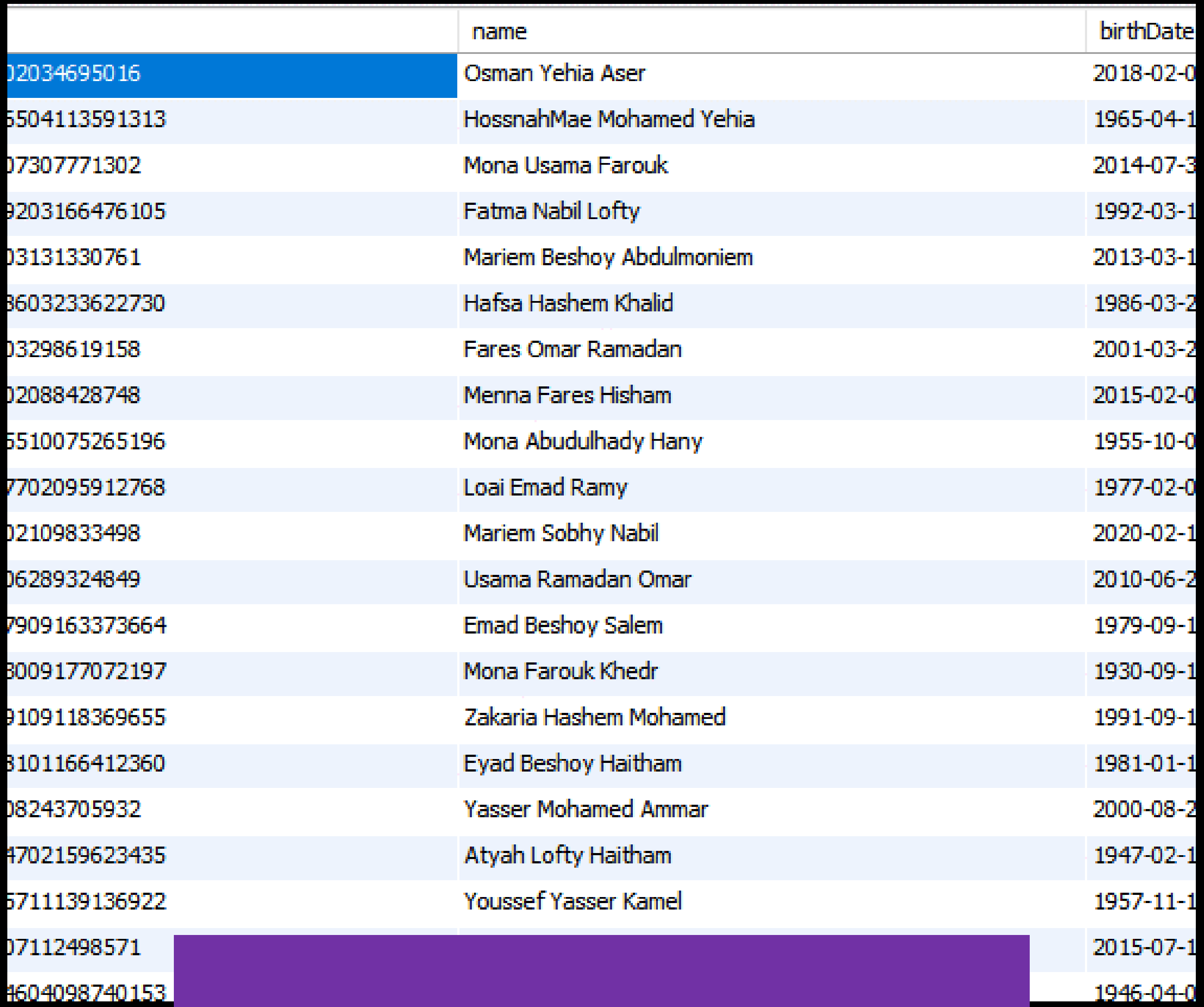
Basic summary for every one of our queries.



The Culprit Behind Insertions



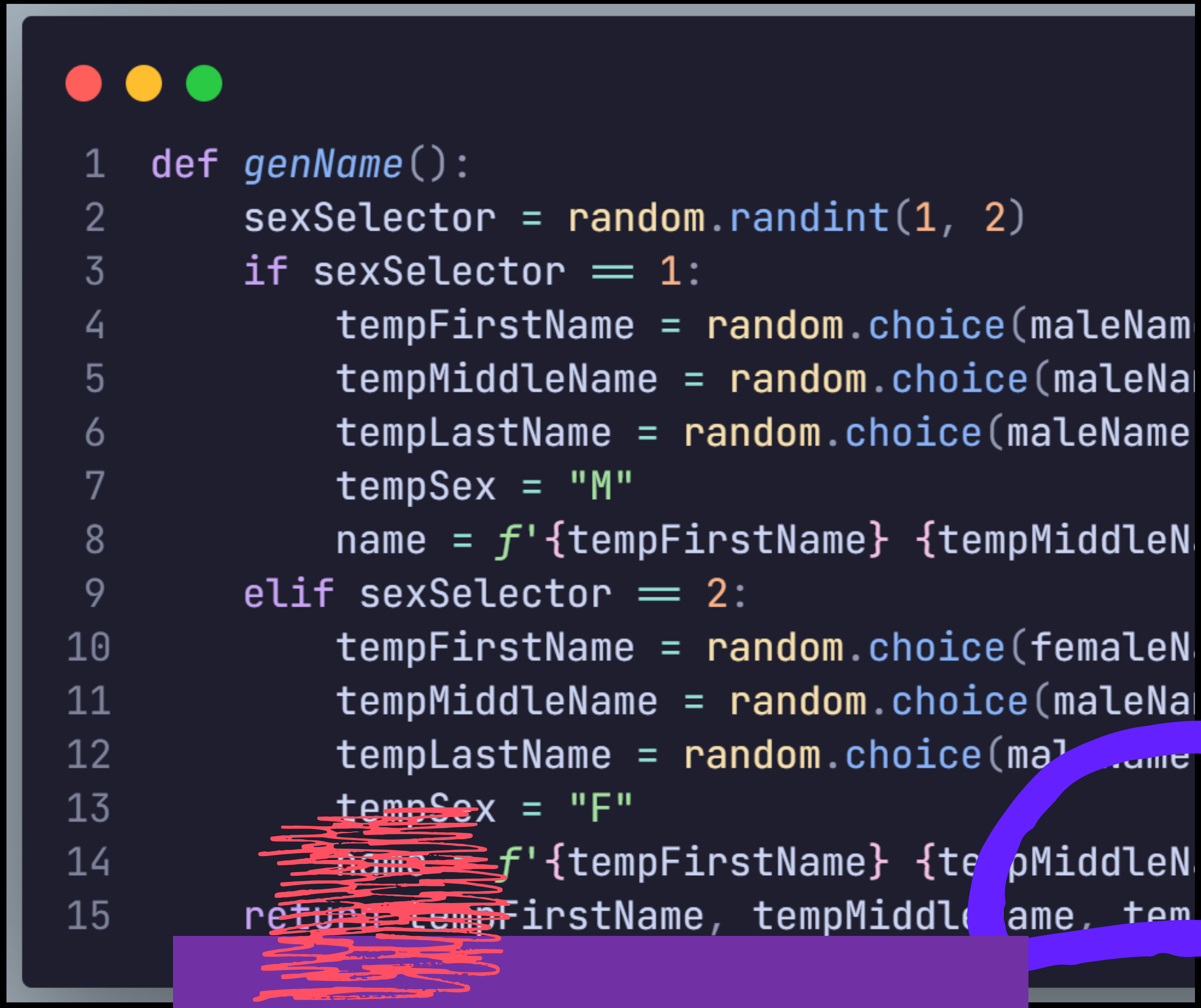
```python
#This code defines a function called name() that randomly selects a first name, middle name, and last name. It also determines the sex of the person based on a randomly generated number
def genName():
    sexSelector = random.randint(1, 2)
    if sexSelector == 1:
        tempFirstName = random.choice(maleName)
        tempMiddleName = random.choice(maleName)
        tempLastName = random.choice(maleName)
        tempSex = "M"
        name = f'{tempFirstName} {tempMiddleName} {tempLastName}'
    elif sexSelector == 2:
        tempFirstName = random.choice(femaleName)
        tempMiddleName = random.choice(maleName)
        tempLastName = random.choice(maleName)
        tempSex = "F"
        name = f'{tempFirstName} {tempMiddleName} {tempLastName}'
    return tempFirstName, tempMiddleName, tempLastName, name, tempSex

def genID():
    tempMonth = random.randrange(1, 12)
    if tempMonth == 2:
        tempDay = random.randrange(1, 28)
    elif tempMonth in [1, 3, 5, 7, 8, 10, 12]:
        tempDay = random.randrange(1, 31)
    elif tempMonth in [4, 6, 9, 11]:
        tempDay = random.randrange(1, 30)

    if tempDay < 10:
        tempDay = f'0{tempDay}'
    if tempMonth < 10:
        tempMonth = f'0{tempMonth}'
    genSelector = random.randint(0, 1)
    if genSelector == 0:
        y = random.randrange(30, 99)
        tempBirthYear = int(f'19{y}')
        tempNationalID = f'2{tempBirthYear}{tempMonth}{tempDay}{random.randint(1000000, 9999999)}'
        tempBirthDate = f'{tempBirthYear}-{tempMonth}-{tempDay}'
    elif genSelector == 1:
        y = random.randrange(00, 23)
        if y < 10:
```



	name	birthDate
02034695016	Osman Yehia Aser	2018-02-0
5504113591313	HossnahMae Mohamed Yehia	1965-04-1
07307771302	Mona Usama Farouk	2014-07-3
9203166476105	Fatma Nabil Lofty	1992-03-1
03131330761	Mariem Beshoy Abdulmoniem	2013-03-1
8603233622730	Hafsa Hashem Khalid	1986-03-2
03298619158	Fares Omar Ramadan	2001-03-2
02088428748	Menna Fares Hisham	2015-02-0
5510075265196	Mona Abudulhady Hany	1955-10-0
7702095912768	Loai Emad Ramy	1977-02-0
02109833498	Mariem Sobhy Nabil	2020-02-1
06289324849	Usama Ramadan Omar	2010-06-2
7909163373664	Emad Beshoy Salem	1979-09-1
3009177072197	Mona Farouk Khedr	1930-09-1
9109118369655	Zakaria Hashem Mohamed	1991-09-1
8101166412360	Eyad Beshoy Haitham	1981-01-1
08243705932	Yasser Mohamed Ammar	2000-08-2
4702159623435	Atyah Lofty Haitham	1947-02-1
5711139136922	Youssef Yasser Kamel	1957-11-1
07112498571		2015-07-1
4604098740153		1946-04-0

Data (CSV)s

This helped us achieve workbench side inserts.



```
def genName():
    sexSelector = random.randint(1, 2)
    if sexSelector == 1:
        tempFirstName = random.choice(maleNam
        tempMiddleName = random.choice(maleNa
        tempLastName = random.choice(maleName
        tempSex = "M"
        name = f'{tempFirstName} {tempMiddleN
    elif sexSelector == 2:
        tempFirstName = random.choice(femaleN
        tempMiddleName = random.choice(maleNa
        tempLastName = random.choice(maleName
        tempSex = "F"
        [redacted] f'{tempFirstName} {tempMiddleN
    [redacted]FirstName, tempMiddleName, tem
```

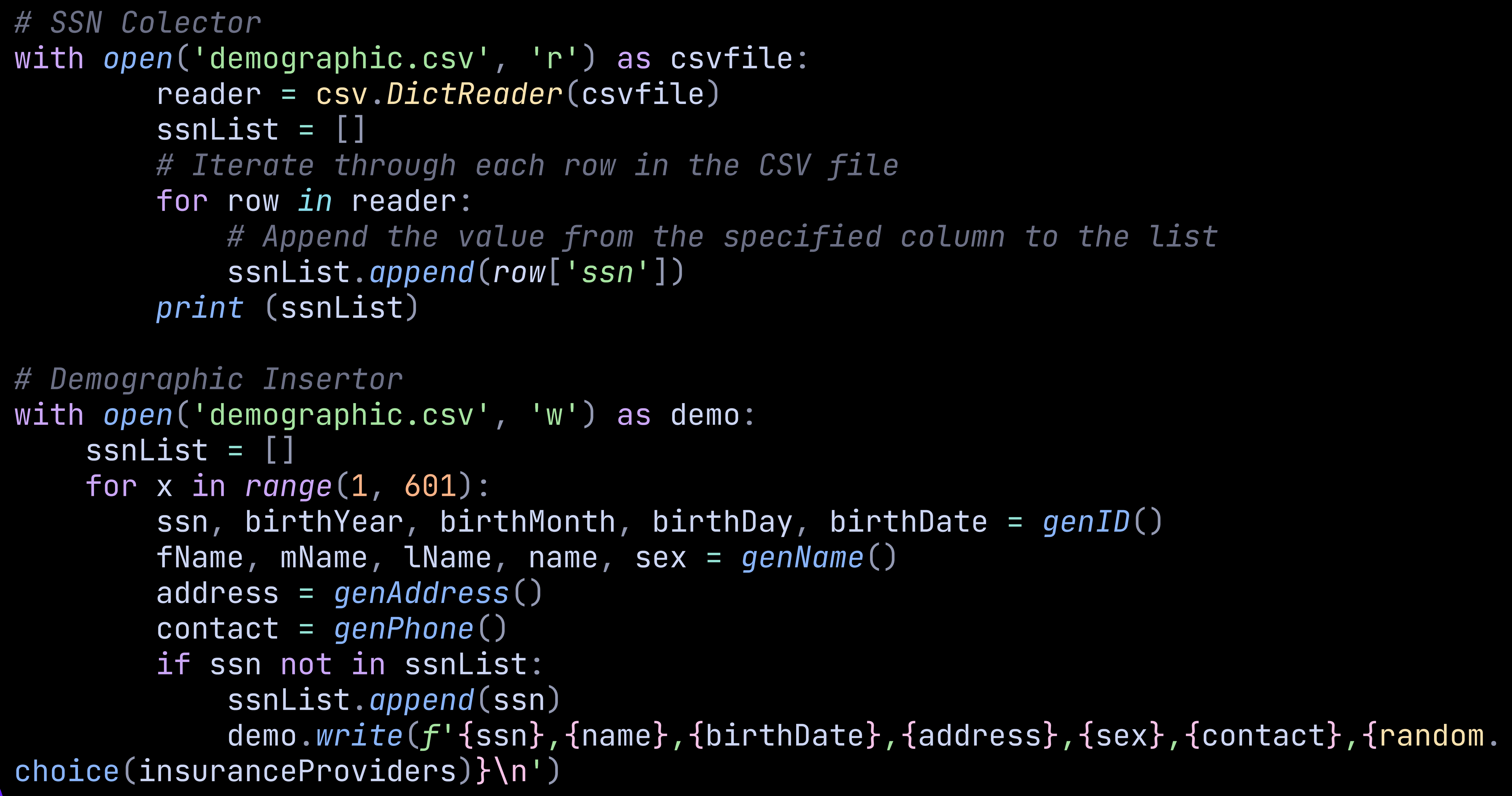
Python Scripts

Main way to make our records.

```python
maleName = ['Abdulaziz', 'Abdulmoniem', 'Abdulrahman', 'Abudulhady', 'Ahmed', 'Ali',
'Amir', 'Ammar', 'Aser', 'Ashraf', 'Atyah', 'Awad', 'Beshoy', 'Boles', 'Emad',
'Eyad', 'Ezzat', 'Fares', 'Farouk', 'Fathy', 'Haitham', 'Hany', 'Hashem', 'Hisham',
'Ibrahim', 'Ismael', 'Kamel', 'Kariem', 'Khalid', 'Khedr', 'Loai', 'Lofty',
'Mahmoud', 'Mamdouh', 'Moamen', 'Mohamed', 'Motaz', 'Nabil', 'Omar', 'Osman',
'Rafat', 'Ramadan', 'Ramez', 'Ramy', 'Salem', 'Salem', 'Soliman', 'Sameh', 'Seif',
'Sherif', 'Sobhy', 'Taha', 'Tamer', 'Usama', 'Walid', 'Yasser', 'Yassin', 'Yehia',
'Yonis', 'Youssef', 'Zakaria', 'Zeyad']

femaleName = ['Aya', 'Eman', 'Esraa', 'Fatma', 'Ghadeer', 'Hafsa', 'Hala', 'Hana',
'Hanaa', 'Hanaa', 'Heba', 'Hend', 'HossnahMae', 'Jana', 'Jomana', 'Malak', 'Mariem',
'Mariem', 'Menna', 'Mona', 'Moreen', 'Nada', 'Noureen', 'Perry', 'Rawia', 'Reem',
'Salma', 'Sara', 'Shahd', 'Shereen', 'Somaya', 'Verina', 'iNour’]

insuranceProviders = [
    "Allianz Egypt",
    "AXA Egypt",
    "MetLife Egypt",
    "Delta Insurance Company",
    "Wethaq Takaful Insurance",
    "Arope Insurance Egypt",
    "Misr Insurance Company (MIC)",
    "Suez Canal Insurance Company (SCIC)",
    "Arab Misr Insurance Group (AMIG)",
    "Egyptian Saudi Insurance House (ESIH)"
]
```

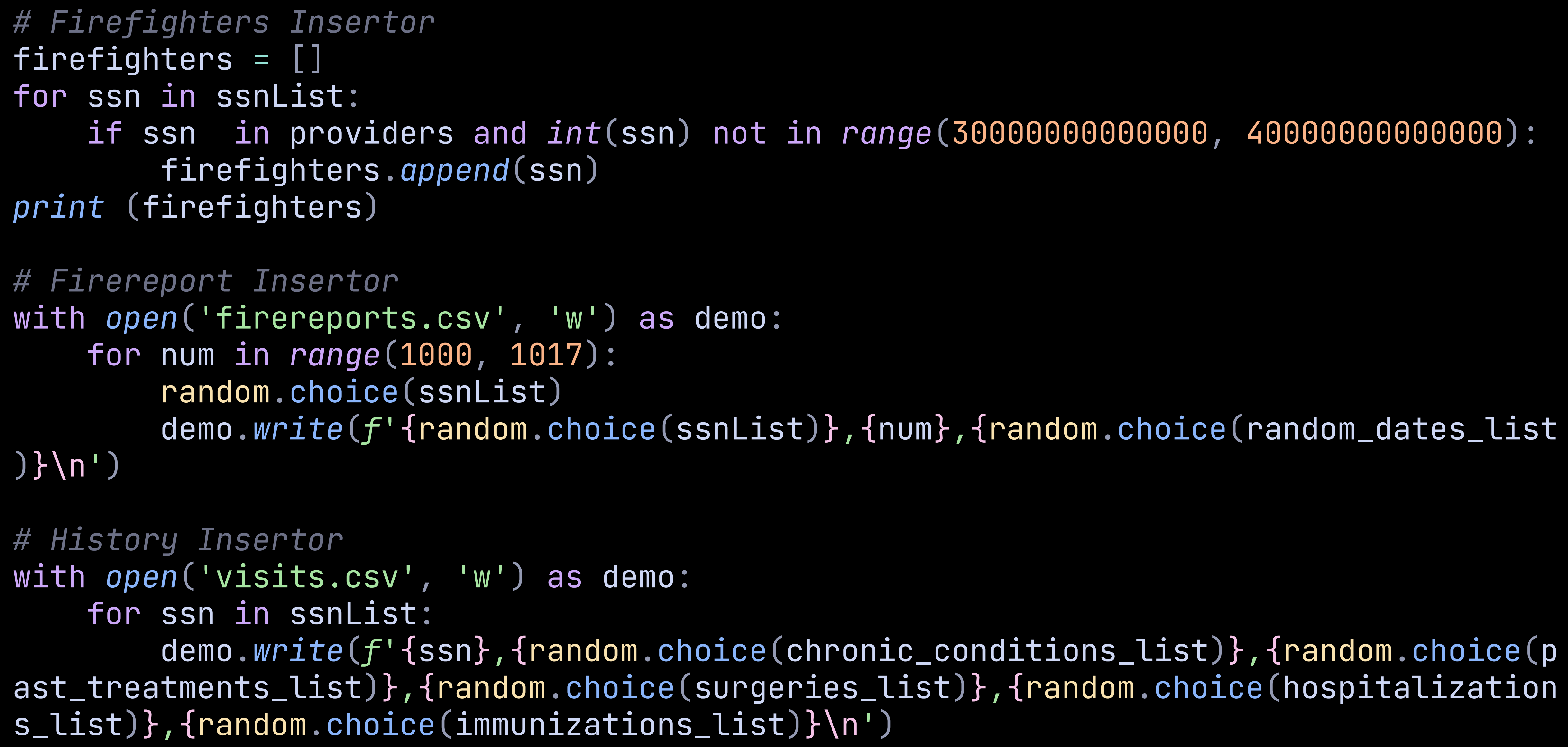
```
medication_warnings = [
    "May cause drowsiness",
    "Avoid alcohol consumption",
    "Take with food",
    "Do not exceed recommended dosage",
    "Consult a doctor before use",
    "May cause allergic reactions",
    "Keep out of reach of children",
    "May cause stomach upset",
    "Avoid driving or operating machinery",
    "Do not stop abruptly",
    "May cause headache",
    "May increase blood pressure",
    "Avoid exposure to sunlight",
    "May cause nausea",
    "May lead to dizziness",
    "Do not use if pregnant",
    "May cause diarrhea",
    "Keep in a cool, dry place",
    "May affect kidney function",
    "Avoid prolonged use"
]
```

```
specialized_skills_firefighters = [
    "Hazardous materials handling",
    "Technical rescue operations",
    "Wildland firefighting",
    "Emergency medical services",
    "Fire investigation",
    "Vehicle extrication",
    "Incident command system",
    "Rope rescue",
    "Structural collapse rescue",
    "Swiftwater rescue",
    "Confined space rescue",
    "High-angle rescue",
    "Trench rescue",
    "Fire behavior",
    "Fire prevention and inspection",
    "Public education and outreach",
    "Firefighting tactics and strategy",
    "Fire equipment maintenance",
    "Radio communication procedures",
    "Physical fitness and endurance training"
]
```

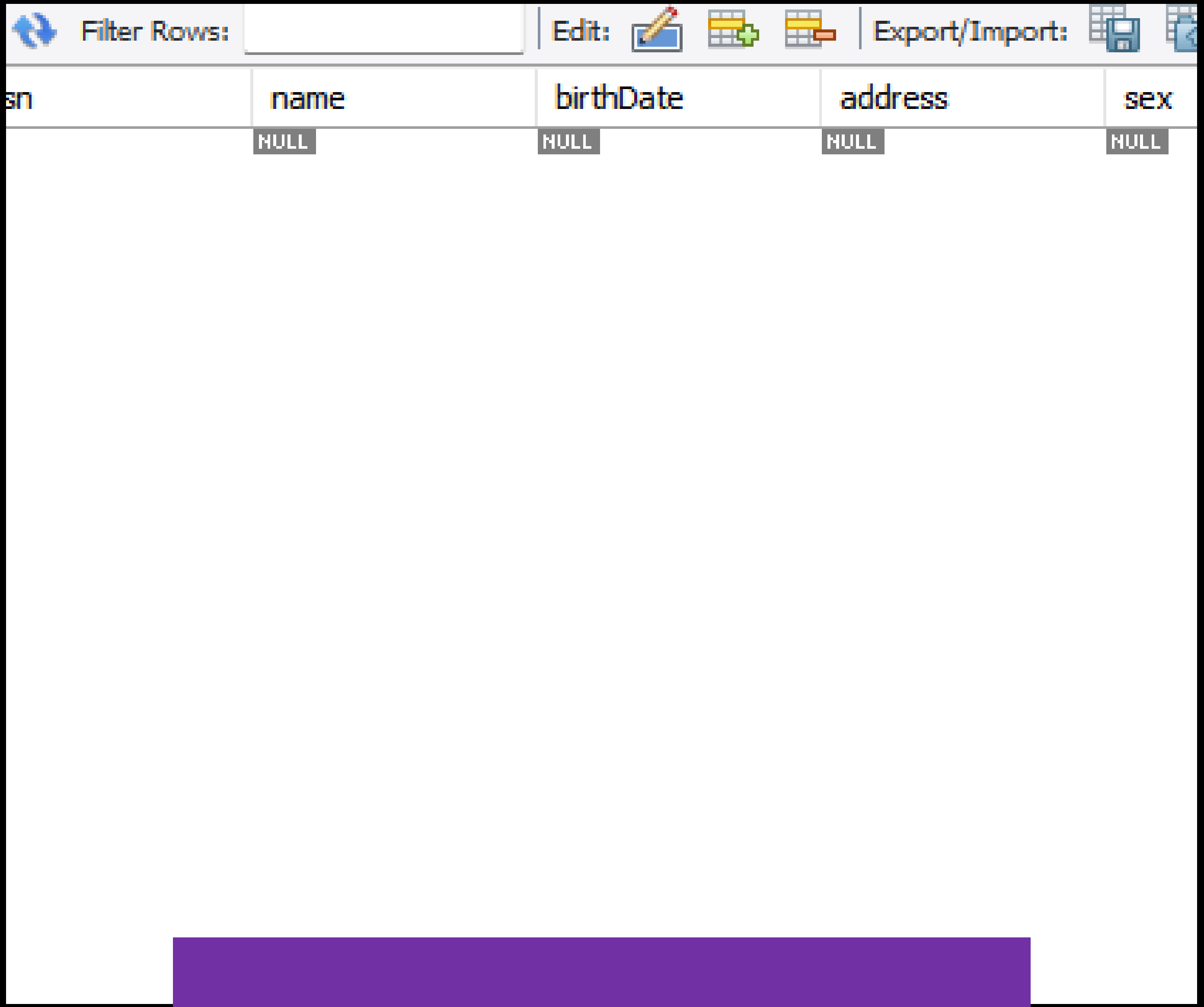


```
# SSN Colector
with open('demographic.csv', 'r') as csvfile:
        reader = csv.DictReader(csvfile)
        ssnList = []
        # Iterate through each row in the CSV file
        for row in reader:
            # Append the value from the specified column to the list
            ssnList.append(row['ssn'])
        print (ssnList)

# Demographic Insertor
with open('demographic.csv', 'w') as demo:
    ssnList = []
    for x in range(1, 601):
        ssn, birthYear, birthMonth, birthDay, birthDate = genID()
        fName, mName, lName, name, sex = genName()
        address = genAddress()
        contact = genPhone()
        if ssn not in ssnList:
            ssnList.append(ssn)
            demo.write(f'{ssn},{name},{birthDate},{address},{sex},{contact},{random.
choice(insuranceProviders)}\n')
```

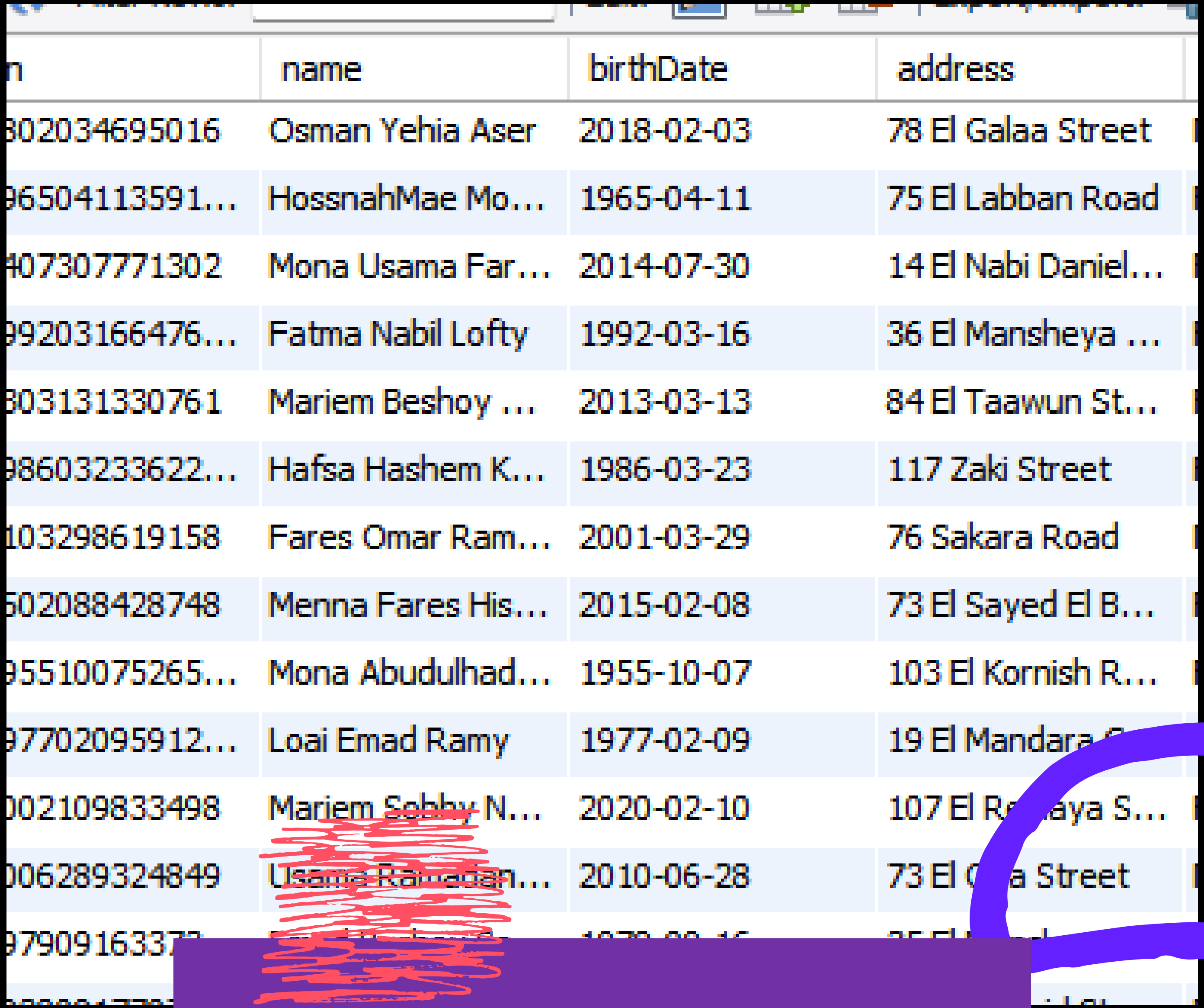
```
# Allergies Insertor
with open('allergies.csv', 'w') as demo:
    for ssn in ssnList:
        demo.write(f'{ssn},{random.choice(allergies)},{random.choice(intolerances)},
{random.choice(adverse_reactions_to_medication)}\n')

# Provider Insertor
with open('provider.csv', 'w') as demo:
    for name in nameList:
        ssn = int(ssnList[nameList.index(name)])
        if ssn not in range(300000000000000, 400000000000000):
            demo.write(f'{ssn},{name}\n')

# Prescriptions Insertor
with open('prescriptions.csv', 'w') as demo:
    for ssn in ssnList:
        demo.write(f'{ssn},{random.choice(providers)},{random.choice(medications)}
{random.choice(dosages)},{random.choice(random_dates_list)},{random.choice(medicatio
n_warnings)} & {random.choice(medication_warnings)} &
{random.choice(medication_warnings)}\n')
```



```
# Firefighters Insertor
firefighters = []
for ssn in ssnList:
    if ssn  in providers and int(ssn) not in range(30000000000000, 40000000000000):
        firefighters.append(ssn)
print (firefighters)

# Firereport Insertor
with open('firereports.csv', 'w') as demo:
    for num in range(1000, 1017):
        random.choice(ssnList)
        demo.write(f'{random.choice(ssnList)},{num},{random.choice(random_dates_list
)}\n')

# History Insertor
with open('visits.csv', 'w') as demo:
    for ssn in ssnList:
        demo.write(f'{ssn},{random.choice(chronic_conditions_list)},{random.choice(p
ast_treatments_list)},{random.choice(surgeries_list)},{random.choice(hospitalization
s_list)},{random.choice(immunizations_list)}\n')
```



sn	name	birthDate	address	sex
	NULL	NULL	NULL	NULL



n	name	birthDate	address
302034695016	Osman Yehia Aser	2018-02-03	78 El Galaa Street
96504113591…	HossnahMae Mo…	1965-04-11	75 El Labban Road
407307771302	Mona Usama Far…	2014-07-30	14 El Nabi Daniel…
99203166476…	Fatma Nabil Lofty	1992-03-16	36 El Mansheya …
303131330761	Mariem Beshoy …	2013-03-13	84 El Taawun St…
98603233622…	Hafsa Hashem K…	1986-03-23	117 Zaki Street
103298619158	Fares Omar Ram…	2001-03-29	76 Sakara Road
502088428748	Menna Fares His…	2015-02-08	73 El Sayed El B…
95510075265…	Mona Abudulhad…	1955-10-07	103 El Kornish R…
97702095912…	Loai Emad Ramy	1977-02-09	19 El Mandara [illegible]
002109833498	Mariem Sobhy N…	2020-02-10	107 El R[illegible]aya S…
006289324849	[redacted]	2010-06-28	73 El [illegible]a Street
97909163373	[redacted]	[illegible]	[illegible]

Before

This helped us achieve workbench side inserts.

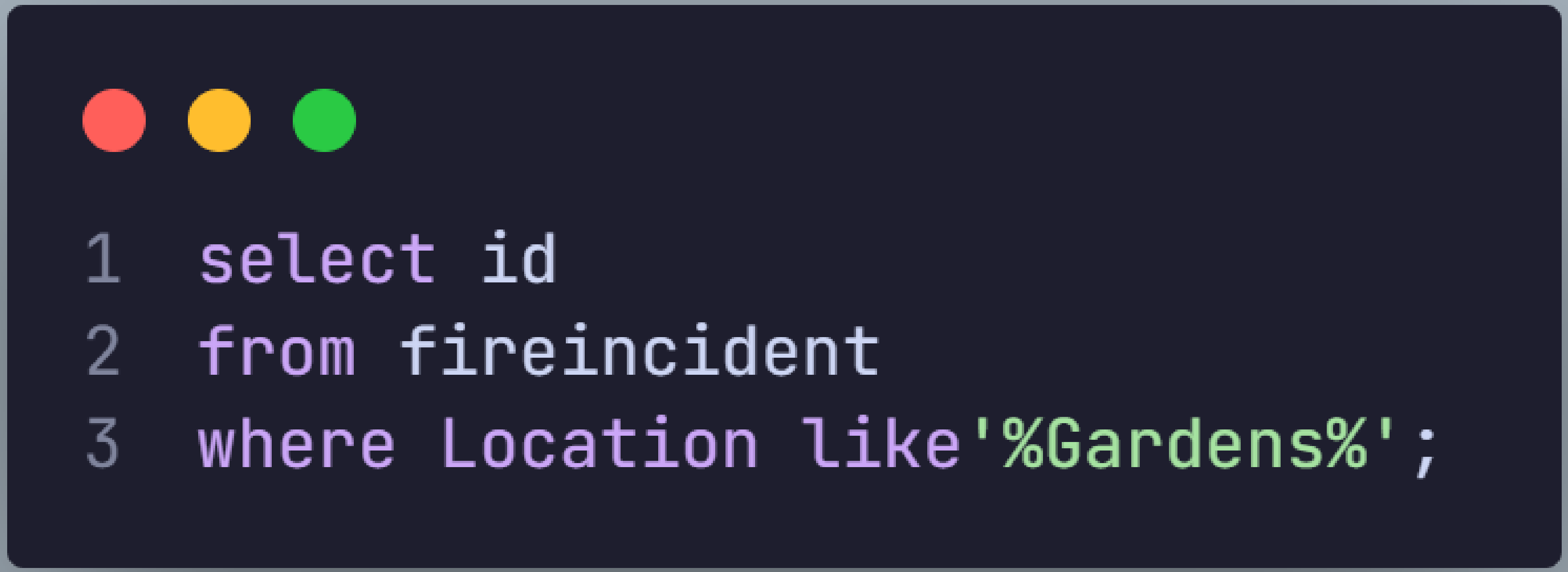
After

Saved us a lot of time for insertions.

Query 1

What does it do?

Return all incidents that happened in a garden

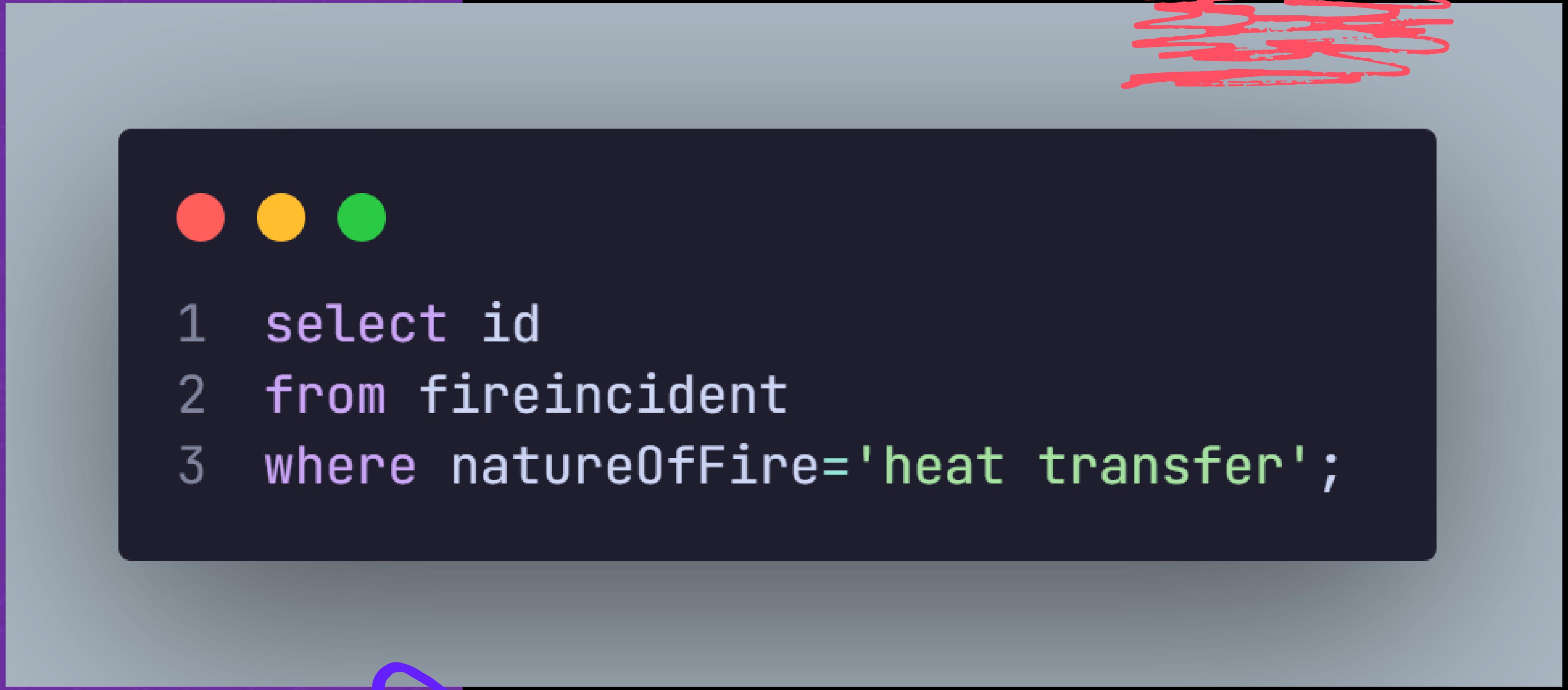


```sql
select id
from fireincident
where Location like'%Gardens%';
```

Query 2

What does it do?

Return incidents where the nature of fire is heat transfer.

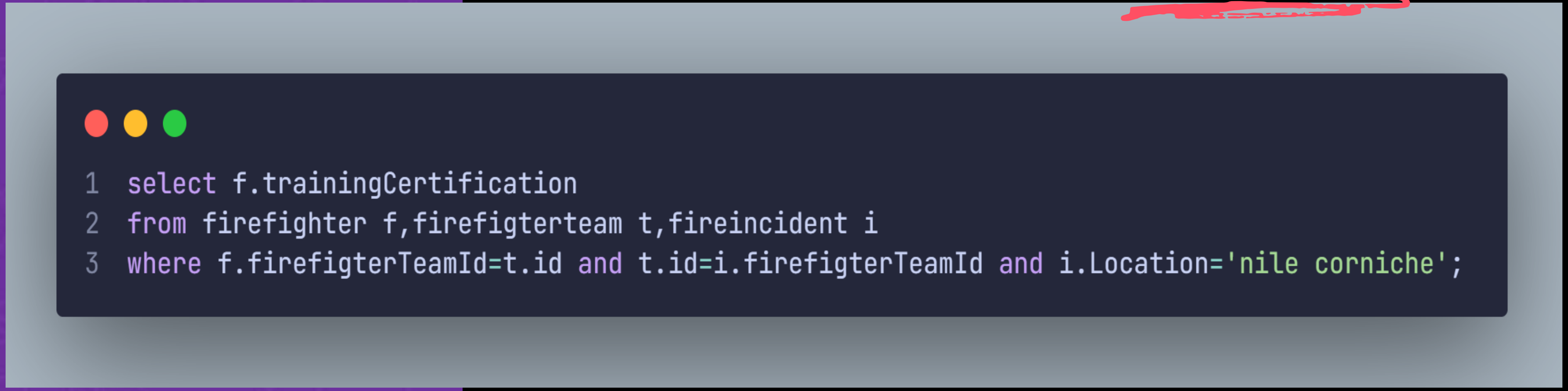


```sql
select id
from fireincident
where natureOfFire='heat transfer';
```

Query 3

What does it do?

Return the training certification.

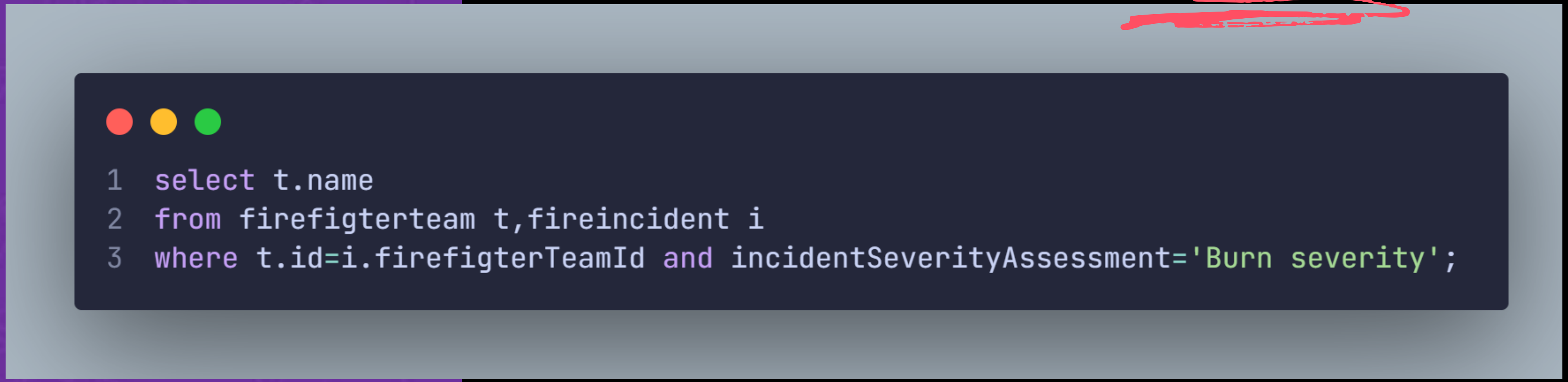


```sql
select f.trainingCertification
from firefighter f,firefigterteam t,fireincident i
where f.firefigterTeamId=t.id and t.id=i.firefigterTeamId and i.Location='nile corniche';
```

Query 4

- What does it do?

  Return the name of firefighter team with incident burn sensitivity.

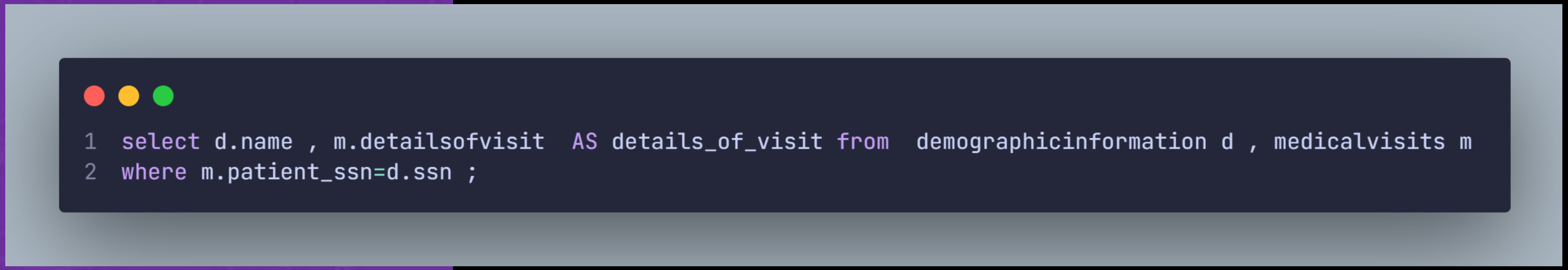


```sql
select t.name
from firefigterteam t,fireincident i
where t.id=i.firefigterTeamId and incidentSeverityAssessment='Burn severity';
```

Query 5

- What does it do?

Return the data of each visit.



```
select d.name , m.detailsofvisit  AS details_of_visit from  demographicinformation d , medicalvisits m
where m.patient_ssn=d.ssn ;
```

S

STRENGTHS

We have 5000+ inserts into our database.
Our database is readable from just Select *

W

WEAKNESSES

Maybe the ER Diagram is not the best in the world.

O

OPPORTUNITIES

Learned a lot about teamwork and how to manage a project as all members participated equally.

T

THREATS

Time was tight as we had a lot of practical exams parallel to the project.

Thank you!

We are DONE!