Practical 2

Aim: - Data Preprocessing and Cleaning for Electronic Health Records (EHR) Data

Theory: -

Electronic Health Records

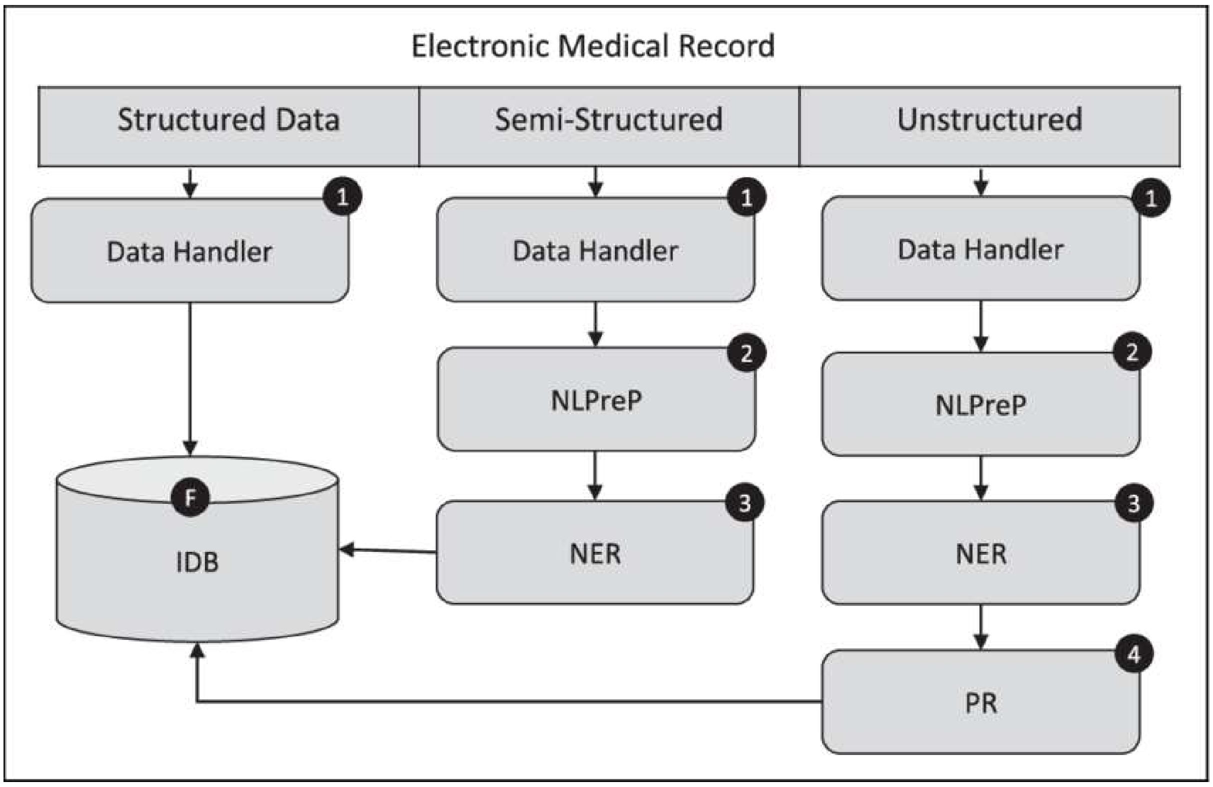
An Electronic Health Record (EHR) is an electronic version of a patients medical history, that is maintained by the provider over time, and may include all of the key administrative clinical data relevant to that persons care under a particular provider, including demographics, progress notes, problems, medications, vital signs, past medical history, immunizations, laboratory data and radiology reports The EHR automates access to information and has the potential to streamline the clinician's workflow. The EHR also has the ability to support other care-related activities directly or indirectly through various interfaces, including evidence-based decision support, quality management, and outcomes reporting.

EHRs are the next step in the continued progress of healthcare that can strengthen the relationship between patients and clinicians. The data, and the timeliness and availability of it, will enable providers to make better decisions and provide better care.

For example, the EHR can improve patient care by:

- Reducing the incidence of medical error by improving the accuracy and clarity of medical records.
- Making the health information available, reducing duplication of tests, reducing delays in treatment, and patients well informed to take better decisions.
- Reducing medical error by improving the accuracy and clarity of medical records.

Block Diagram: -



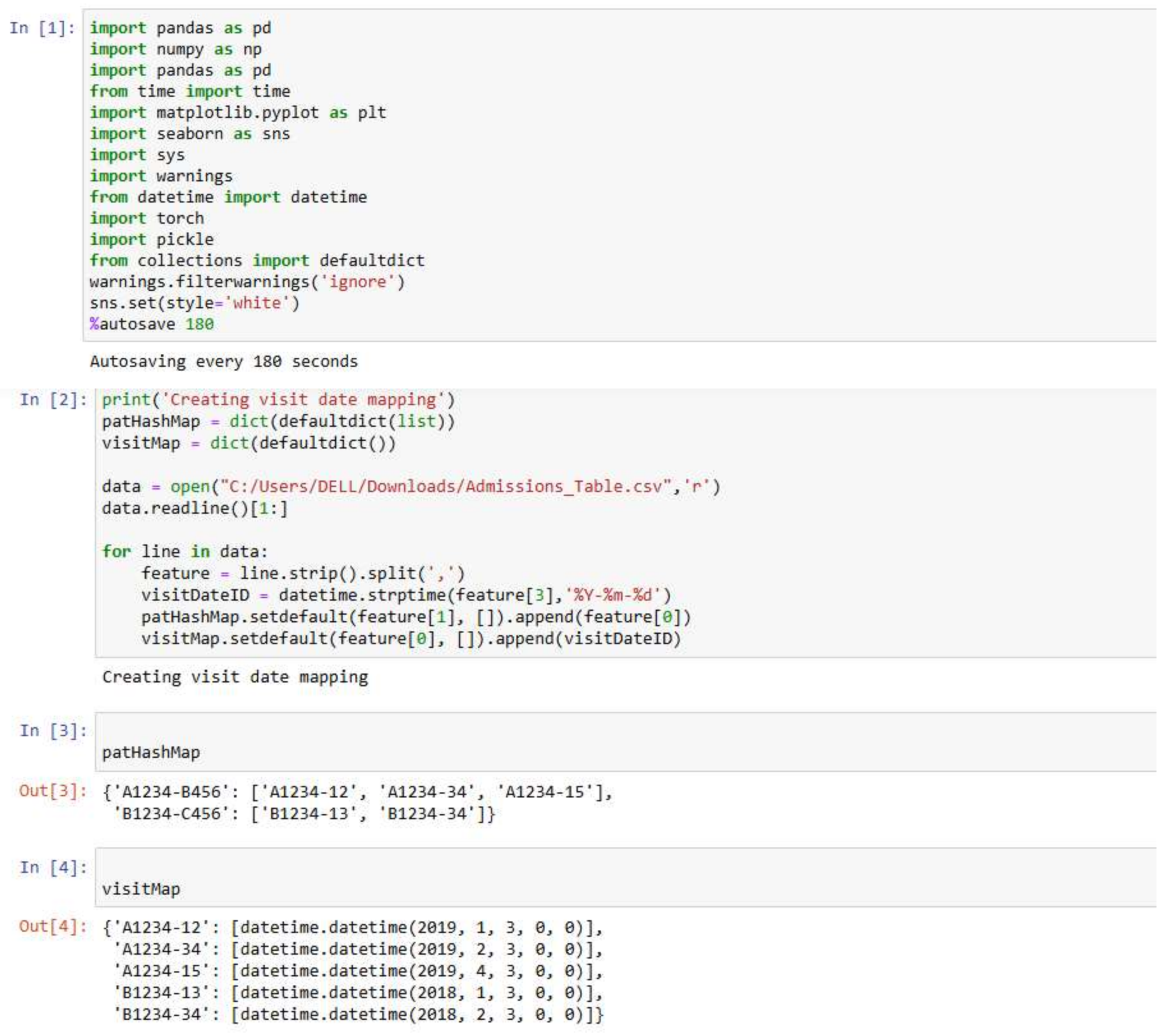
Dataset Description: -

The **Admission Table** captures information about patient admissions in a healthcare facility. Each record represents a unique admission event. Key attributes include:

- **HashKey**: A unique identifier for each record in the table, ensuring data integrity and uniqueness.
- **PatientID**: The unique identifier assigned to a patient.
- **Admission ID**: A unique identifier for a specific admission event tied to a patient.
- **AdmissionStartDate**: The date when the patient was admitted.
- **AdmissionEndDate**: The date when the patient was discharged or the admission concluded.
- The **Diagnosis Table** provides details about medical diagnoses assigned during patient admissions. Each record corresponds to a specific diagnosis event. Key attributes include:
- **HashKey**: A unique identifier for each record in the table, ensuring data integrity and uniqueness.
- **PatientID**: The unique identifier assigned to a patient.

- **Admission ID**: A unique identifier linking the diagnosis to a specific admission event.
- **CodingSystem**: The classification system used for the diagnosis code, such as ICD-10 or SNOMED.
- **PrimaryDiagnosisCode**: The primary diagnosis code representing the most significant condition addressed during the admission.
- **DiagnosisCodeDescription**: A descriptive text explaining the diagnosis associated with the code.

Source Code and Output: -

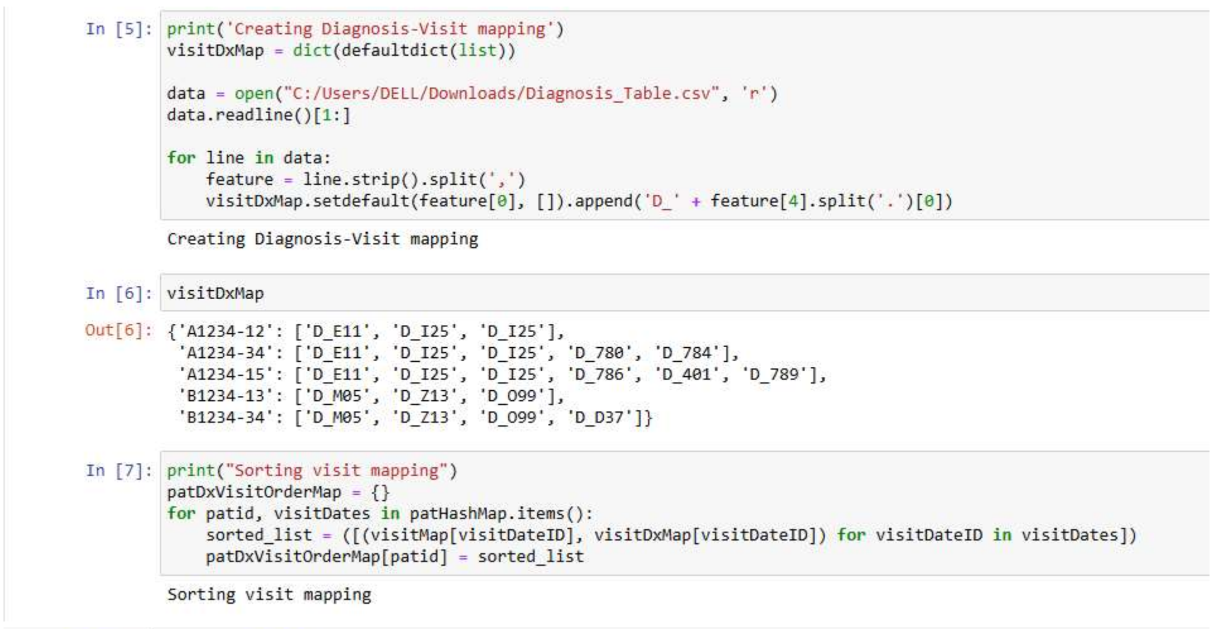


```
In [1]: import pandas as pd
        import numpy as np
        import pandas as pd
        from time import time
        import matplotlib.pyplot as plt
        import seaborn as sns
        import sys
        import warnings
        from datetime import datetime
        import torch
        import pickle
        from collections import defaultdict
        warnings.filterwarnings('ignore')
        sns.set(style='white')
        %autosave 180

        Autosaving every 180 seconds
```

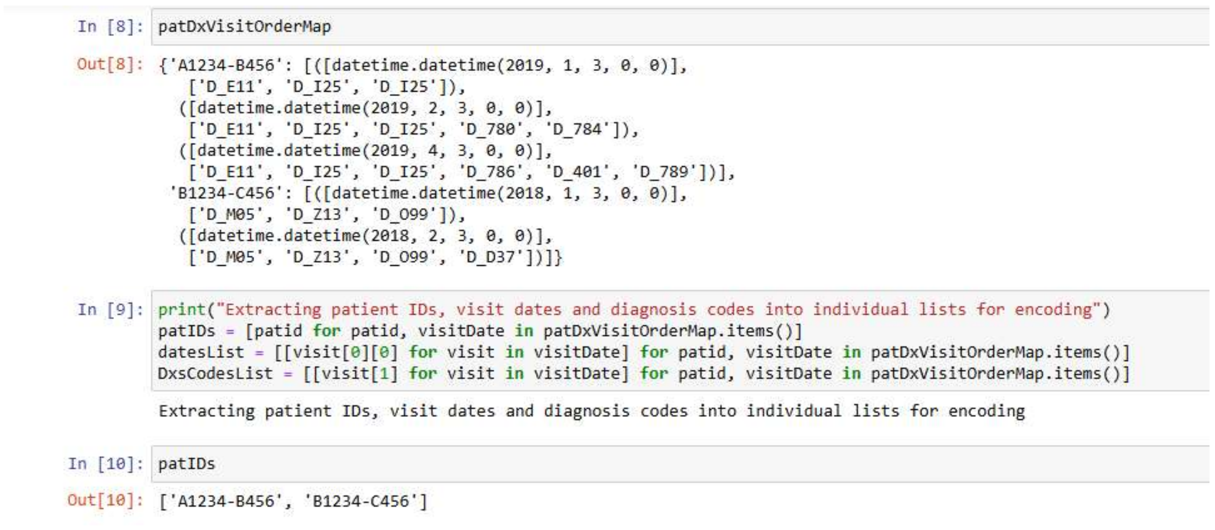
```
In [2]: print('Creating visit date mapping')
        patHashMap = dict(defaultdict(list))
        visitMap = dict(defaultdict())

        data = open("C:/Users/DELL/Downloads/Admissions_Table.csv",'r')
        data.readline()[1:]

        for line in data:
            feature = line.strip().split(',')
            visitDateID = datetime.strptime(feature[3],'%Y-%m-%d')
            patHashMap.setdefault(feature[1], []).append(feature[0])
            visitMap.setdefault(feature[0], []).append(visitDateID)

        Creating visit date mapping
```

```
In [3]:
        patHashMap

Out[3]: {'A1234-B456': ['A1234-12', 'A1234-34', 'A1234-15'],
         'B1234-C456': ['B1234-13', 'B1234-34']}
```

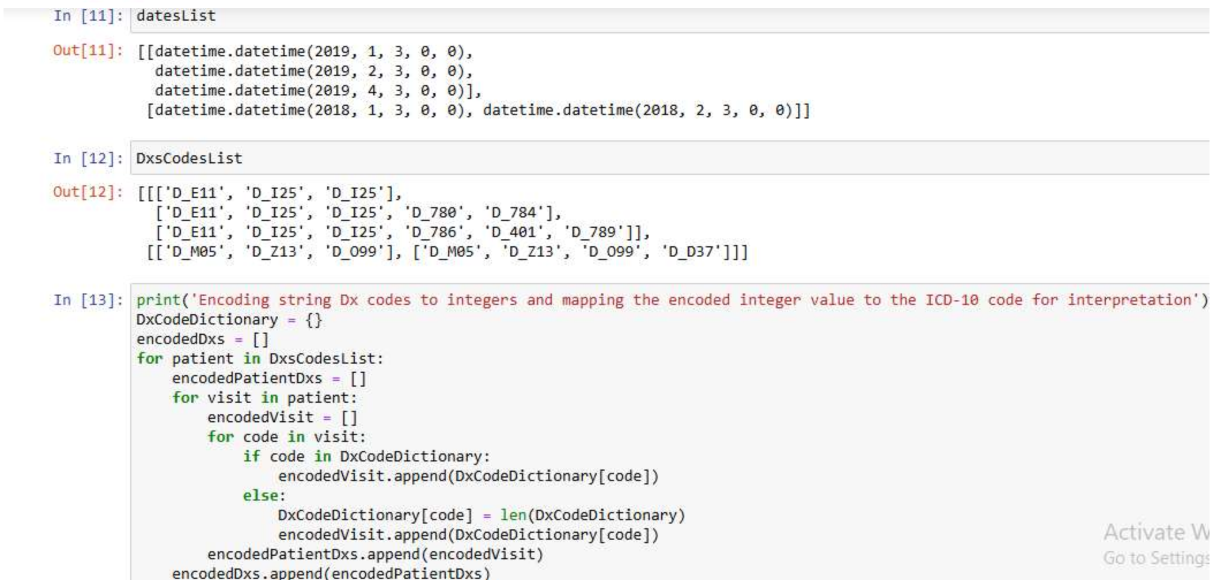
```
In [4]:
        visitMap

Out[4]: {'A1234-12': [datetime.datetime(2019, 1, 3, 0, 0)],
         'A1234-34': [datetime.datetime(2019, 2, 3, 0, 0)],
         'A1234-15': [datetime.datetime(2019, 4, 3, 0, 0)],
         'B1234-13': [datetime.datetime(2018, 1, 3, 0, 0)],
         'B1234-34': [datetime.datetime(2018, 2, 3, 0, 0)]}
```



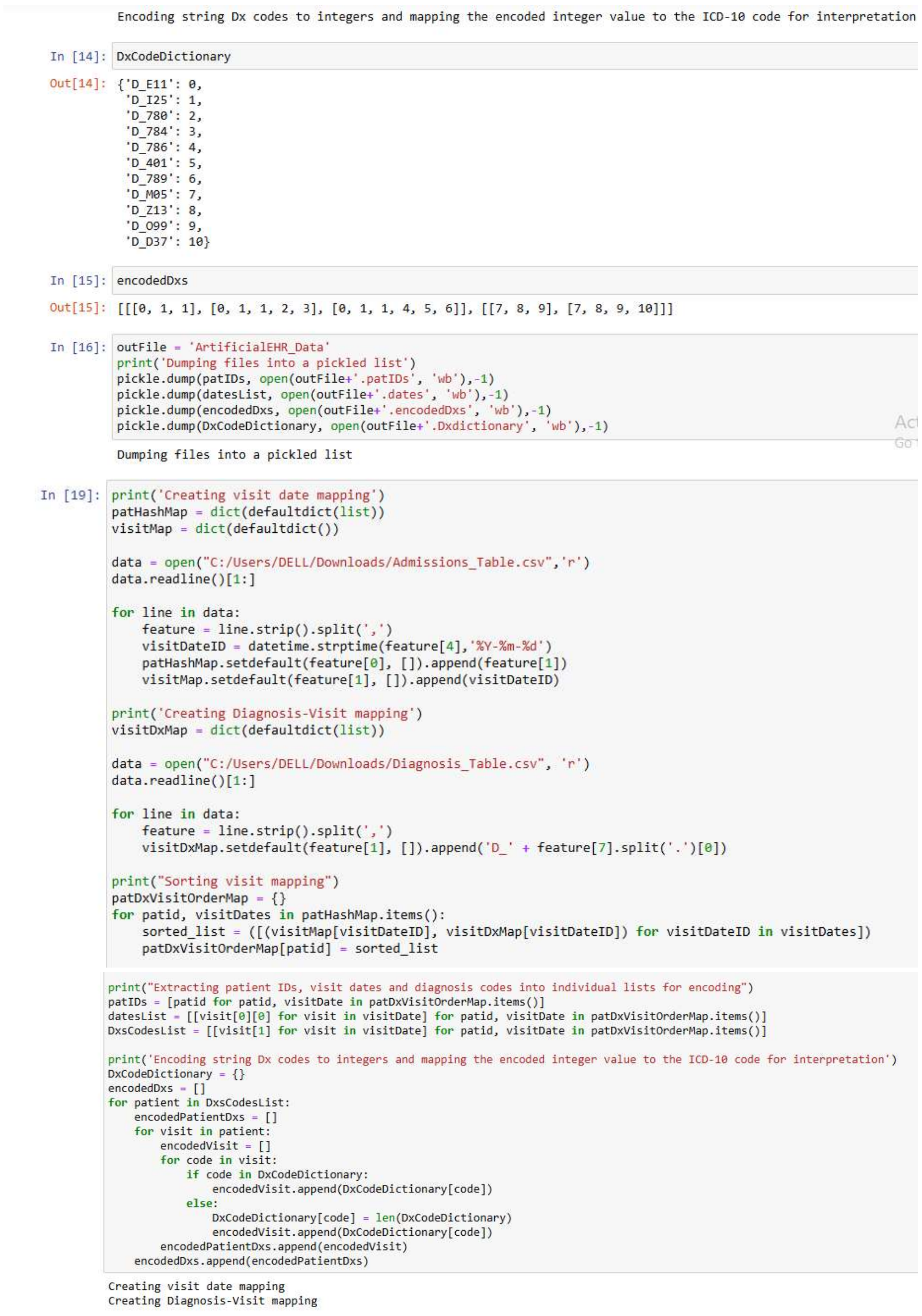
```
In [5]: print('Creating Diagnosis-Visit mapping')
        visitDxMap = dict(defaultdict(list))

        data = open("C:/Users/DELL/Downloads/Diagnosis_Table.csv", 'r')
        data.readline()[1:]

        for line in data:
            feature = line.strip().split(',')
            visitDxMap.setdefault(feature[0], []).append('D_' + feature[4].split('.')[0])

        Creating Diagnosis-Visit mapping

In [6]: visitDxMap

Out[6]: {'A1234-12': ['D_E11', 'D_I25', 'D_I25'],
         'A1234-34': ['D_E11', 'D_I25', 'D_I25', 'D_780', 'D_784'],
         'A1234-15': ['D_E11', 'D_I25', 'D_I25', 'D_786', 'D_401', 'D_789'],
         'B1234-13': ['D_M05', 'D_Z13', 'D_O99'],
         'B1234-34': ['D_M05', 'D_Z13', 'D_O99', 'D_D37']}

In [7]: print("Sorting visit mapping")
        patDxVisitOrderMap = {}
        for patid, visitDates in patHashMap.items():
            sorted_list = ([(visitMap[visitDateID], visitDxMap[visitDateID]) for visitDateID in visitDates])
            patDxVisitOrderMap[patid] = sorted_list

        Sorting visit mapping
```



```
In [8]: patDxVisitOrderMap

Out[8]: {'A1234-B456': [([datetime.datetime(2019, 1, 3, 0, 0)],
           ['D_E11', 'D_I25', 'D_I25']),
          ([datetime.datetime(2019, 2, 3, 0, 0)],
           ['D_E11', 'D_I25', 'D_I25', 'D_780', 'D_784']),
          ([datetime.datetime(2019, 4, 3, 0, 0)],
           ['D_E11', 'D_I25', 'D_I25', 'D_786', 'D_401', 'D_789'])],
         'B1234-C456': [([datetime.datetime(2018, 1, 3, 0, 0)],
           ['D_M05', 'D_Z13', 'D_O99']),
          ([datetime.datetime(2018, 2, 3, 0, 0)],
           ['D_M05', 'D_Z13', 'D_O99', 'D_D37'])]}

In [9]: print("Extracting patient IDs, visit dates and diagnosis codes into individual lists for encoding")
        patIDs = [patid for patid, visitDate in patDxVisitOrderMap.items()]
        datesList = [[visit[0][0] for visit in visitDate] for patid, visitDate in patDxVisitOrderMap.items()]
        DxsCodesList = [[visit[1] for visit in visitDate] for patid, visitDate in patDxVisitOrderMap.items()]

        Extracting patient IDs, visit dates and diagnosis codes into individual lists for encoding

In [10]: patIDs

Out[10]: ['A1234-B456', 'B1234-C456']
```



```
In [11]: datesList

Out[11]: [[datetime.datetime(2019, 1, 3, 0, 0),
           datetime.datetime(2019, 2, 3, 0, 0),
           datetime.datetime(2019, 4, 3, 0, 0)],
          [datetime.datetime(2018, 1, 3, 0, 0), datetime.datetime(2018, 2, 3, 0, 0)]]

In [12]: DxsCodesList

Out[12]: [[['D_E11', 'D_I25', 'D_I25'],
           ['D_E11', 'D_I25', 'D_I25', 'D_780', 'D_784'],
           ['D_E11', 'D_I25', 'D_I25', 'D_786', 'D_401', 'D_789']],
          [['D_M05', 'D_Z13', 'D_O99'], ['D_M05', 'D_Z13', 'D_O99', 'D_D37']]]

In [13]: print('Encoding string Dx codes to integers and mapping the encoded integer value to the ICD-10 code for interpretation')
         DxCodeDictionary = {}
         encodedDxs = []
         for patient in DxsCodesList:
             encodedPatientDxs = []
             for visit in patient:
                 encodedVisit = []
                 for code in visit:
                     if code in DxCodeDictionary:
                         encodedVisit.append(DxCodeDictionary[code])
                     else:
                         DxCodeDictionary[code] = len(DxCodeDictionary)
                         encodedVisit.append(DxCodeDictionary[code])
                 encodedPatientDxs.append(encodedVisit)
             encodedDxs.append(encodedPatientDxs)
```



```
Encoding string Dx codes to integers and mapping the encoded integer value to the ICD-10 code for interpretation
```

```
In [14]: DxCodeDictionary
```

```
Out[14]: {'D_E11': 0,
          'D_I25': 1,
          'D_780': 2,
          'D_784': 3,
          'D_786': 4,
          'D_401': 5,
          'D_789': 6,
          'D_M05': 7,
          'D_Z13': 8,
          'D_O99': 9,
          'D_D37': 10}
```

```
In [15]: encodedDxs
```

```
Out[15]: [[[0, 1, 1], [0, 1, 1, 2, 3], [0, 1, 1, 4, 5, 6]], [[7, 8, 9], [7, 8, 9, 10]]]
```

```
In [16]: outFile = 'ArtificialEHR_Data'
         print('Dumping files into a pickled list')
         pickle.dump(patIDs, open(outFile+'.patIDs', 'wb'),-1)
         pickle.dump(datesList, open(outFile+'.dates', 'wb'),-1)
         pickle.dump(encodedDxs, open(outFile+'.encodedDxs', 'wb'),-1)
         pickle.dump(DxCodeDictionary, open(outFile+'.Dxdictionary', 'wb'),-1)
```

```
Dumping files into a pickled list
```

```
In [19]: print('Creating visit date mapping')
         patHashMap = dict(defaultdict(list))
         visitMap = dict(defaultdict())

         data = open("C:/Users/DELL/Downloads/Admissions_Table.csv",'r')
         data.readline()[1:]

         for line in data:
             feature = line.strip().split(',')
             visitDateID = datetime.strptime(feature[4],'%Y-%m-%d')
             patHashMap.setdefault(feature[0], []).append(feature[1])
             visitMap.setdefault(feature[1], []).append(visitDateID)

         print('Creating Diagnosis-Visit mapping')
         visitDxMap = dict(defaultdict(list))

         data = open("C:/Users/DELL/Downloads/Diagnosis_Table.csv", 'r')
         data.readline()[1:]

         for line in data:
             feature = line.strip().split(',')
             visitDxMap.setdefault(feature[1], []).append('D_' + feature[7].split('.')[0])

         print("Sorting visit mapping")
         patDxVisitOrderMap = {}
         for patid, visitDates in patHashMap.items():
             sorted_list = ([(visitMap[visitDateID], visitDxMap[visitDateID]) for visitDateID in visitDates])
             patDxVisitOrderMap[patid] = sorted_list
```

```
print("Extracting patient IDs, visit dates and diagnosis codes into individual lists for encoding")
patIDs = [patid for patid, visitDate in patDxVisitOrderMap.items()]
datesList = [[visit[0][0] for visit in visitDate] for patid, visitDate in patDxVisitOrderMap.items()]
DxsCodesList = [[visit[1] for visit in visitDate] for patid, visitDate in patDxVisitOrderMap.items()]

print('Encoding string Dx codes to integers and mapping the encoded integer value to the ICD-10 code for interpretation')
DxCodeDictionary = {}
encodedDxs = []
for patient in DxsCodesList:
    encodedPatientDxs = []
    for visit in patient:
        encodedVisit = []
        for code in visit:
            if code in DxCodeDictionary:
                encodedVisit.append(DxCodeDictionary[code])
            else:
                DxCodeDictionary[code] = len(DxCodeDictionary)
                encodedVisit.append(DxCodeDictionary[code])
        encodedPatientDxs.append(encodedVisit)
    encodedDxs.append(encodedPatientDxs)
```

```
Creating visit date mapping
Creating Diagnosis-Visit mapping
```

Conclusion: - The experiment successfully preprocesses electronic health record (EHR) data, integrating admission and diagnosis datasets to create a cohesive, clean, and structured dataset that facilitates accurate analysis and supports informed decision-making in healthcare analytics.