Test the Explainability of the model for sample data extracted from the database

This cell builds a set of stringification helper functions that convert structured clinical data — such as ICD diagnosis codes, lab results, medication orders, and admission metadata — into natural-language sentences. These functions form the foundation of the Explainability Agent, allowing the system to describe patient encounters in human-readable text rather than raw database codes.

Concretely:

- It defines templates to turn ICD codes (diagnoses), lab results, medication orders, and admission details into narrative text.
- The stringify_visit() function integrates these elements into a full patient summary, providing a coherent paragraph describing an encounter.
- The final block (under if **name** == '**main**':) demonstrates how each function generates readable summaries from sample data.

In the context of the assignment:

This code operationalizes the Explainability Agent's role — translating structured hospital data into clinician-friendly narratives that explain what the system "sees" and "thinks."

It prepares the groundwork for generating explanations agent.

In [2]:
```
# import numpy as np
# example ICD loading
def stringify_icd(code_type, code_value, code_text):
    return f"ICD-{str(code_type)} code: [{str(code_value).strip().replace('.', '').

# Patient {str(pid)} visited with encounter_id {enc_id} and

def stringify_visit_codes(codes):
    return_list = [f"The patient received the following diagnostic codes:"]
    for code in codes:
        return_list.append(stringify_icd(**code))
    return " ".join(return_list)
# example lab loading

def stringify_lab(lab_name, lab_value, datetime=None, valueuom=None, flag=None):
    return_list = []
    if flag == 'abnormal':
        return_list.append('abnormal')
```

```
    return_list.append(f"{str(lab_name).strip().lower()} of {str(lab_value).strip()

    if valueuom is not None and str(valueuom) != 'nan':
        return_list.append(str(valueuom).strip().lower())

    if datetime is not None and str(datetime) != 'nan':
        return_list.append(f"on {str(datetime).strip().lower()}")

    return " ".join(return_list) + "."


def stringify_visit_labs(labs):
    return_list = [f"The patient had the following labs:"]
    for lab in labs:
        return_list.append(stringify_lab(**lab))
    return " ".join(return_list)

# example medication loading
def stringify_med(drug, starttime, route=None, endtime=None):

    return_list = [f"{str(drug).strip().lower()} ordered"]
    if route is not None:
        return_list.append(f"via {str(route).strip().lower()}")
    return_list.append(f"at {str(starttime).strip().lower()}")

    if endtime is not None:
        return_list.append(f"and ended at {str(endtime).strip().lower()}")
    return " ".join(return_list) + "."

def stringify_visit_meds(meds):
    return_list = [f"The patient was ordered the following medications:"]
    for med in meds:
        return_list.append(stringify_med(**med))
    return " ".join(return_list)


# patient meta from admissions.csv


def stringify_visit_meta(subject_id, hadm_id, admittime,
                    insurance=None, language=None, marital_status=None, race=None,
                    dischtime=None, deathtime=None,
                    admission_type=None,
                    admission_location=None, discharge_location=None):
    return_list = [f"Patient {str(subject_id).strip().lower()} was seen at {str(adm

    if admission_type is not None:
        return_list.append(f"The admission type was {str(admission_type).strip().lo
    if admission_location is not None:
        return_list.append(f"The means of arrival was {str(admission_location).stri
    if language is not None:
        return_list.append(f"The patient's primary language was {str(language).stri
    if race is not None:
        return_list.append(f"The patient's race was {str(race).strip().lower()}.")
    if marital_status is not None:
```

```
        return_list.append(f"The patient's marital status was {str(marital_status).
    if insurance is not None:
        return_list.append(f"The patient's insurance was {str(insurance).strip().lo

    # if we want to include discharge information leave these in
    if dischtime is not None:
        return_list.append(f"The patient was discharged on {str(dischtime).strip().
    if deathtime is not None and str(deathtime) != 'nan':
        return_list.append(f"The patient is deceased as of {str(deathtime).strip().
    return " ".join(return_list)




def stringify_visit(admission_details, meds=None, labs=None, codes=None):
    return_string = [stringify_visit_meta(**admission_details)]

    if meds is None:
        return_string.append("No medications were ordered.")
    else:
        return_string.append(stringify_visit_meds(meds))

    if labs is None:
        return_string.append("No labs were ordered.")
    else:
        return_string.append(stringify_visit_labs(labs))

    if codes is None:
        return_string.append("No diagnostic codes were assigned.")
    else:
        return_string.append(stringify_visit_codes(codes))

    return " ".join(return_string)



# %%
if __name__ == '__main__':
    admission_details = dict(
        subject_id = 10004235,
        hadm_id = 24181354,
        admittime='08/09/2023',
        dischtime=None,
        deathtime=None,
        admission_type='URGENT',
        admission_location='TRANSFER FROM HOSPITAL',
        discharge_location=None,
        insurance='Medicaid',
        language='ENGLISH',
        marital_status='SINGLE',
        race='WHITE'
    )

    med_dict = dict(
        drug = "multivitamins",
```

```
        route = "IV",
        starttime = "08/09/2023",
        endtime = "08/09/2023"
    )

    lab_dict = dict(
        lab_name = "% Ionized Calcium", # from d_labitems
        lab_value = 15.4,  # from labs table
        valueuom = '%',  # from labs table
        flag = 'abnormal', # from labs table
        datetime = '08/09/2023'
    )

    code_dict = dict(
        code_type = 9,
        code_value = '4170',
        # note: need to merge with d_icd to get the code text
        code_text = "Arteriovenous fistula of pulmonary vessels")


    print(stringify_visit_meta(**admission_details))
    print(stringify_icd(**code_dict))
    print(stringify_med(**med_dict))
    print(stringify_lab(**lab_dict))


    meds = [med_dict, med_dict]
    print(stringify_visit_meds(meds))

    codes = [code_dict, code_dict]
    stringify_visit_codes(codes)

    labs = [lab_dict, lab_dict]
    print(stringify_visit_labs(labs))

    stringify_visit(admission_details, meds, labs, codes)
```

```
Patient 10004235 was seen at 08/09/2023 and given admission id 24181354. The admissi
on type was urgent. The means of arrival was transfer from hospital. The patient's p
rimary language was english. The patient's race was white. The patient's marital sta
tus was single. The patient's insurance was medicaid.
ICD-9 code: [4170], arteriovenous fistula of pulmonary vessels.
multivitamins ordered via iv at 08/09/2023 and ended at 08/09/2023.
abnormal % ionized calcium of 15.4 % on 08/09/2023.
The patient was ordered the following medications: multivitamins ordered via iv at 0
8/09/2023 and ended at 08/09/2023. multivitamins ordered via iv at 08/09/2023 and en
ded at 08/09/2023.
The patient had the following labs: abnormal % ionized calcium of 15.4 % on 08/09/20
23. abnormal % ionized calcium of 15.4 % on 08/09/2023.
```