SA-AKI

In [1]:
```
import dask.dataframe as dd
import pandas as pd
import numpy as np
```

In [2]:
```
sepsis_data = pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\sepsis3.parquet')
```

In [ ]:
```
sepsis_data
```

In [4]:
```
sepsis_data.loc[:,'suspected_infection_time'] = pd.to_datetime(sepsis_data['suspected_infection_time'], dayfirst=True)
sepsis_data['suspected_infection_time'] = sepsis_data['suspected_infection_time'].astype('datetime64[ns]')

sepsis_data.loc[:,'sofa_time'] = pd.to_datetime(sepsis_data['sofa_time'], dayfirst=True)
sepsis_data['sofa_time'] = sepsis_data['sofa_time'].astype('datetime64[ns]')

sepsis_data.loc[:,'antibiotic_time'] = pd.to_datetime(sepsis_data['antibiotic_time'], dayfirst=True)
sepsis_data['antibiotic_time'] = sepsis_data['antibiotic_time'].astype('datetime64[ns]')

sepsis_data.loc[:,'culture_time'] = pd.to_datetime(sepsis_data['culture_time'], dayfirst=True)
sepsis_data['culture_time'] = sepsis_data['culture_time'].astype('datetime64[ns]')
```

In [5]:
```
sepsis_data['infection_time'] = sepsis_data.apply(lambda row: min(row['antibiotic_time'], row['suspected_infection_time'], row['culture_time']), axis=1)
```

In [6]:
```
sepsis_data['los_anti-infe'] = sepsis_data['antibiotic_time'] - sepsis_data['infection_time']
```

In [7]:
```
sepsis_data.head()
```

Out[7]:

	subject_id	stay_id	antibiotic_time	culture_time	suspected_infection_time	sofa_time	sofa_score	respiration	coagulation	liver	cardiovascular	cns	renal	sepsis3	infection_time	los_anti-infe
0	18421337	30000484	2136-01-14 21:00:00	2136-01-14 18:10:00	2136-01-14 18:10:00	2136-01-14 19:00:00	3	0	0	0	0	3	0	t	2136-01-14 18:10:00	0 days 02:50:00
1	12207593	30000646	2194-04-29 07:00:00	2194-04-29 01:00:00	2194-04-29 01:00:00	2194-04-29 11:00:00	3	2	0	0	1	0	0	t	2194-04-29 01:00:00	0 days 06:00:00
2	15726459	30000831	2140-04-18 18:00:00	2140-04-18 05:11:00	2140-04-18 05:11:00	2140-04-17 22:00:00	5	0	0	0	0	3	2	t	2140-04-18 05:11:00	0 days 12:49:00
3	16513856	30001446	2186-04-12 04:00:00	2186-04-11 08:20:00	2186-04-11 08:20:00	2186-04-12 04:00:00	8	0	3	3	0	0	2	t	2186-04-11 08:20:00	0 days 19:40:00
4	10656173	30001555	2177-09-27 16:00:00	2177-09-27 07:21:00	2177-09-27 07:21:00	2177-09-27 12:00:00	8	0	3	4	0	1	0	t	2177-09-27 07:21:00	0 days 08:39:00

In [8]:
```
sepsis_data = sepsis_data.drop(['culture_time', 'antibiotic_time', 'suspected_infection_time'], axis=1)
```

In [9]:
```
sepsis_data.columns
```

Out[9]:
```
Index(['subject_id', 'stay_id', 'sofa_time', 'sofa_score', 'respiration',
       'coagulation', 'liver', 'cardiovascular', 'cns', 'renal', 'sepsis3',
       'infection_time', 'los_anti-infe'],
      dtype='object')
```

In [10]:
```
sepsis_data1 = sepsis_data.drop(['respiration', 'coagulation', 'liver', 'cardiovascular', 'cns', 'renal'], axis=1)
```

In [11]:
```
sepsis_data1.shape
```

Out[11]:
```
(41295, 7)
```

In [12]:
```
sepsis_data1.stay_id.nunique()
```

Out[12]:
```
41295
```

In [13]:
```
sepsis_data1.subject_id.nunique()
```

Out[13]:
```
31910
```

In [14]:
```
result = sepsis_data1.sort_values(by='stay_id')
```

In [15]:
```
result.head()
```

Out[15]:

	subject_id	stay_id	sofa_time	sofa_score	sepsis3	infection_time	los_anti-infe
0	18421337	30000484	2136-01-14 19:00:00	3	t	2136-01-14 18:10:00	0 days 02:50:00
1	12207593	30000646	2194-04-29 11:00:00	3	t	2194-04-29 01:00:00	0 days 06:00:00
2	15726459	30000831	2140-04-17 22:00:00	5	t	2140-04-18 05:11:00	0 days 12:49:00
3	16513856	30001446	2186-04-12 04:00:00	8	t	2186-04-11 08:20:00	0 days 19:40:00
4	10656173	30001555	2177-09-27 12:00:00	8	t	2177-09-27 07:21:00	0 days 08:39:00

In [ ]:
```
result.groupby('subject_id')['stay_id'].value_counts()
```

In [17]:
```
first_sta_id = result.groupby('subject_id')['stay_id'].min().reset_index()
```

In [ ]:
```
first_sta_id.head()
```

In [19]:
```
first_sta_id.stay_id.nunique()
```

Out[19]:
```
31910
```

In [20]:
```
first_sta_id.subject_id.nunique()
```

Out[20]:
```
31910
```

合并两个表格共同的部分

In [21]:
```
final_result = result.merge(first_sta_id, on=['stay_id','subject_id'])
```

In [22]:
```
final_result.head()
```

Out[22]:

	subject_id	stay_id	sofa_time	sofa_score	sepsis3	infection_time	los_anti-infe
0	18421337	30000484	2136-01-14 19:00:00	3	t	2136-01-14 18:10:00	0 days 02:50:00
1	12207593	30000646	2194-04-29 11:00:00	3	t	2194-04-29 01:00:00	0 days 06:00:00
2	15726459	30000831	2140-04-17 22:00:00	5	t	2140-04-18 05:11:00	0 days 12:49:00
3	16513856	30001446	2186-04-12 04:00:00	8	t	2186-04-11 08:20:00	0 days 19:40:00
4	10656173	30001555	2177-09-27 12:00:00	8	t	2177-09-27 07:21:00	0 days 08:39:00

In [23]:
```
final_result.shape
```

Out[23]: (31910, 7)

In [24]:
```
final_result = final_result.query('sofa_score> =2')
```

In [25]:
```
final_result.shape
```

Out[25]: (31910, 7)

In [26]:
```
final_result=final_result.drop('sepsis3',axis=1)
```

In [27]:
```
final_result.shape
```

Out[27]: (31910, 6)

以上数据为因脓毒症第一次住院治疗的患者

1

获取基线资料

In [28]:
```
icu_stays = pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\icustay_detail.parquet')
```

In [ ]:
```
icu_stays.head()
```

In [30]:
```
icu_stays['admittime'] = pd.to_datetime(icu_stays['admittime'], dayfirst=True)
icu_stays['dischtime'] = pd.to_datetime(icu_stays['dischtime'], dayfirst=True)
icu_stays['icu_intime'] = pd.to_datetime(icu_stays['icu_intime'], dayfirst=True)
icu_stays['icu_outtime'] = pd.to_datetime(icu_stays['icu_outtime'], dayfirst=True)
icu_stays['admittime'] = icu_stays['admittime'].astype('datetime64[ns]')
icu_stays['dischtime'] = icu_stays['dischtime'].astype('datetime64[ns]')
icu_stays['icu_intime'] = icu_stays['icu_intime'].astype('datetime64[ns]')
icu_stays['icu_outtime'] = icu_stays['icu_outtime'].astype('datetime64[ns]')
icu_stays['dod'] = pd.to_datetime(icu_stays['dod'], dayfirst=True)
icu_stays['dod'] = icu_stays['dod'].astype('datetime64[ns]')
```

In [31]:
```
df = pd.merge(icu_stays, final_result, on=['subject_id','stay_id'])
```

In [32]:
```
df.shape
```

Out[32]: (31910, 22)

In [33]:
```
df.stay_id.nunique()
```

Out[33]: 31910

In [34]:
```
df.subject_id.nunique()
```

Out[34]: 31910

In [35]:
```
df.hadm_id.nunique()
```

Out[35]: 31910

In [36]:
```
df['first_icu_stay'].value_counts()
```

Out[36]:
```
first_icu_stay
t    29878
f     2032
Name: count, dtype: int64
```

In [37]:
```
df0 = df.query('first_icu_stay == "t"')
```

In [38]:
```
df0.shape
```

Out[38]: (29878, 22)

In [39]:
```
df0 = df0.sort_values(by='admittime')
```

In [ ]:
```
df0.head()
```

In [41]:
```
first_index = df0.groupby('hadm_id')['admittime'].min().reset_index()
```

In [ ]:
```
first_index
```

In [43]:
```
df0 = pd.merge(df0,first_index,on=['hadm_id','admittime'])
```

In [44]:
```
df0.shape
```

Out[44]: (29878, 22)

In [45]:
```
df_icu = df0.query('los_icu >= 0')
```

In [46]:
```
df_icu.subject_id.nunique()
```

Out[46]: 29878

In [47]:
```
df_icu.stay_id.nunique()
```

Out[47]: 29878

In [48]:
```
df_icu.shape
```

Out[48]: (29878, 22)

年龄大于18岁

In [49]:
```
df1 = df_icu.query('admission_age > 18')
```

In [50]:
```
df1.shape
```

Out[50]: (29878, 22)

In [51]:
```
df1 = df1.drop(['icustay_seq','first_icu_stay','first_hosp_stay'], axis=1)
```

In [52]:
```
df1.columns
```

Out[52]:
```
Index(['subject_id', 'hadm_id', 'stay_id', 'gender', 'dod', 'admittime',
       'dischtime', 'los_hospital', 'admission_age', 'race',
       'hospital_expire_flag', 'hospstay_seq', 'icu_intime', 'icu_outtime',
       'los_icu', 'sofa_time', 'sofa_score', 'infection_time',
       'los_anti-infe'],
      dtype='object')
```

In [53]:
```
df1 = df1.drop(['los_hospital','hospital_expire_flag','hospstay_seq'], axis=1)
```

In [54]:
```
df1.head()
```

Out[54]:

	subject_id	hadm_id	stay_id	gender	dod	admittime	dischtime	admission_age	race	icu_intime	icu_outtime	los_icu	sofa_time	sofa_score	infection_time	los_anti-infe
0	18106347	24305596	30588857	F	NaT	2110-01-11 10:14:00	2110-01-15 17:31:00	48.028547	WHITE	2110-01-11 10:16:06	2110-01-12 17:17:47	1.29	2110-01-11 17:00:00	3	2110-01-11 12:00:00	0 days 00:00:00
1	16800952	22641185	31337458	M	2112-04-12	2110-01-16 04:04:00	2110-02-04 17:50:00	26.041533	OTHER	2110-01-16 04:28:00	2110-01-17 17:51:08	1.56	2110-01-16 07:00:00	3	2110-01-16 02:17:00	0 days 04:43:00
2	13201095	28453791	39953418	F	2110-01-25	2110-01-18 14:46:00	2110-01-25 09:40:00	88.048229	UNKNOWN	2110-01-18 14:46:27	2110-01-25 12:42:11	6.91	2110-01-18 16:00:00	5	2110-01-18 15:21:00	0 days 01:39:00
3	12770182	20446666	34901199	M	NaT	2110-01-18 17:46:00	2110-01-21 16:30:00	53.048571	WHITE	2110-01-18 17:47:47	2110-01-20 22:25:09	2.19	2110-01-19 00:00:00	2	2110-01-19 00:00:00	0 days 00:00:00
4	12106780	25661012	37081503	M	2110-09-08	2110-01-31 00:00:00	2110-02-12 12:15:00	91.082137	WHITE	2110-02-04 10:31:02	2110-02-06 18:33:23	2.33	2110-02-04 13:00:00	6	2110-02-03 17:04:00	0 days 20:56:00

AKI

In [55]:
```
kdigo_stages = pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\kdigo_stages.parquet')
kdigo_stages.head()
```

Out[55]:

	subject_id	hadm_id	stay_id	charttime	creat_low_past_7day	creat_low_past_48hr	creat	aki_stage_creat	uo_rt_6hr	uo_rt_12hr	uo_rt_24hr	aki_stage_uo	aki_stage_crrt	aki_stage	aki_stage_sr
0	10000032	29079034	39553978	23/7/2180 06:39:00	NaN	NaN	0.7	0.0	NaN	NaN	NaN	NaN	NaN	0	
1	10000032	29079034	39553978	23/7/2180 15:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0	
2	10000032	29079034	39553978	23/7/2180 21:45:00	0.7	0.7	0.5	0.0	NaN	NaN	NaN	NaN	NaN	0	
3	10000690	25860671	37081114	2/11/2150 12:10:00	NaN	NaN	1.0	0.0	NaN	NaN	NaN	NaN	NaN	0	
4	10000690	25860671	37081114	2/11/2150 21:27:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0	

In [56]:
```
kdigo_stages.shape
```

Out[56]:
```
(5099899, 15)
```

In [57]:
```
kdigo_stages =kdigo_stages.drop(['creat_low_past_7day','creat_low_past_48hr','aki_stage_creat','uo_rt_6hr','uo_rt_12hr','uo_rt_24hr'], axis=1)
```

In [58]:
```
kdigo_stages.head()
```

Out[58]:

	subject_id	hadm_id	stay_id	charttime	creat	aki_stage_uo	aki_stage_crrt	aki_stage	aki_stage_smoothed
0	10000032	29079034	39553978	23/7/2180 06:39:00	0.7	NaN	NaN	0	0
1	10000032	29079034	39553978	23/7/2180 15:00:00	NaN	NaN	NaN	0	0
2	10000032	29079034	39553978	23/7/2180 21:45:00	0.5	NaN	NaN	0	0
3	10000690	25860671	37081114	2/11/2150 12:10:00	1.0	NaN	NaN	0	0
4	10000690	25860671	37081114	2/11/2150 21:27:00	NaN	NaN	NaN	0	0

In [59]:
```
kdigo_stages.head()
```

Out[59]:

	subject_id	hadm_id	stay_id	charttime	creat	aki_stage_uo	aki_stage_crrt	aki_stage	aki_stage_smoothed
0	10000032	29079034	39553978	23/7/2180 06:39:00	0.7	NaN	NaN	0	0
1	10000032	29079034	39553978	23/7/2180 15:00:00	NaN	NaN	NaN	0	0
2	10000032	29079034	39553978	23/7/2180 21:45:00	0.5	NaN	NaN	0	0
3	10000690	25860671	37081114	2/11/2150 12:10:00	1.0	NaN	NaN	0	0
4	10000690	25860671	37081114	2/11/2150 21:27:00	NaN	NaN	NaN	0	0

In [60]:
```
kdigo_stages =kdigo_stages.drop(['creat','aki_stage_uo','aki_stage_crrt','aki_stage'], axis=1)
```

In [61]:
```
mask = kdigo_stages[['subject_id', 'hadm_id', 'stay_id']].isin(df1[['subject_id', 'hadm_id', 'stay_id']].to_dict('list')).all(axis=1)
kdigo_stages1 = kdigo_stages[mask]
```

In [62]:
```
kdigo_stages1.shape
```

Out[62]:
```
(2519230, 5)
```

In [63]:
```
kdigo_stages1.head()
```

Out[63]:

	subject_id	hadm_id	stay_id	charttime	aki_stage_smoothed
3	10000690	25860671	37081114	2/11/2150 12:10:00	0
4	10000690	25860671	37081114	2/11/2150 21:27:00	0
5	10000690	25860671	37081114	2/11/2150 22:00:00	0
6	10000690	25860671	37081114	2/11/2150 23:00:00	0
7	10000690	25860671	37081114	3/11/2150 00:00:00	0

In [ ]:
```
kdigo_stages1.rename(columns={'aki_stage_smoothed': 'aki_stage'}, inplace=True)
```

In [65]:
```
kdigo_stages1.head()
```

Out[65]:

	subject_id	hadm_id	stay_id	charttime	aki_stage
3	10000690	25860671	37081114	2/11/2150 12:10:00	0
4	10000690	25860671	37081114	2/11/2150 21:27:00	0
5	10000690	25860671	37081114	2/11/2150 22:00:00	0
6	10000690	25860671	37081114	2/11/2150 23:00:00	0
7	10000690	25860671	37081114	3/11/2150 00:00:00	0

In [66]:
```
kdigo_stages1.subject_id.nunique()
```

Out[66]: 29878

In [67]:
```
kdigo_stages2= kdigo_stages1.groupby('stay_id').filter(lambda x: (x['aki_stage'] != 0).any())
```

In [68]:
```
kdigo_stages2.subject_id.nunique()
```

Out[68]: 24364

In [69]:
```
kdigo_stages2.head()
```

Out[69]:

	subject_id	hadm_id	stay_id	charttime	aki_stage
3	10000690	25860671	37081114	2/11/2150 12:10:00	0
4	10000690	25860671	37081114	2/11/2150 21:27:00	0
5	10000690	25860671	37081114	2/11/2150 22:00:00	0
6	10000690	25860671	37081114	2/11/2150 23:00:00	0
7	10000690	25860671	37081114	3/11/2150 00:00:00	0

全是AKI时间

In [70]:
```
kdigo_stages3= kdigo_stages2.query('aki_stage >=1')
```

In [71]:
```
kdigo_stages= kdigo_stages3
```

In [72]:
```
kdigo_stages.shape
```

Out[72]: (1102255, 5)

In [73]:
```
kdigo_stages.head()
```

Out[73]:

	subject_id	hadm_id	stay_id	charttime	aki_stage
13	10000690	25860671	37081114	3/11/2150 09:00:00	1
14	10000690	25860671	37081114	3/11/2150 10:00:00	1
15	10000690	25860671	37081114	3/11/2150 12:00:00	2
16	10000690	25860671	37081114	3/11/2150 13:00:00	2
17	10000690	25860671	37081114	3/11/2150 14:00:00	2

In [74]:
```
kdigo_stages=kdigo_stages.drop('hadm_id',axis=1)
```

In [75]:
```
kdigo_stages.head()
```

Out[75]:

	subject_id	stay_id	charttime	aki_stage
13	10000690	37081114	3/11/2150 09:00:00	1
14	10000690	37081114	3/11/2150 10:00:00	1
15	10000690	37081114	3/11/2150 12:00:00	2
16	10000690	37081114	3/11/2150 13:00:00	2
17	10000690	37081114	3/11/2150 14:00:00	2

In [76]:
```
kdigo_stages1 = pd.merge(kdigo_stages, df1, on=['subject_id','stay_id'])
```

In [77]:
```
kdigo_stages1.shape
```

Out[77]: (1102255, 18)

In [78]:
```
kdigo_stages1.head()
```

Out[78]:

	subject_id	stay_id	charttime	aki_stage	hadm_id	gender	dod	admittime	dischtime	admission_age	race	icu_intime	icu_outtime	los_icu	sofa_time	sofa_score	infection_time	los_ant inf
0	10000690	37081114	3/11/2150 09:00:00	1	25860671	F	2152-01-30	2150-11-02 18:02:00	2150-11-12 13:45:00	86.83712	WHITE	2150-11-02 19:37:00	2150-11-06 17:03:17	3.89	2150-11-02 20:00:00	3	2150-11-02 12:10:00	0 day 12:50:0
1	10000690	37081114	3/11/2150 10:00:00	1	25860671	F	2152-01-30	2150-11-02 18:02:00	2150-11-12 13:45:00	86.83712	WHITE	2150-11-02 19:37:00	2150-11-06 17:03:17	3.89	2150-11-02 20:00:00	3	2150-11-02 12:10:00	0 day 12:50:0
2	10000690	37081114	3/11/2150 12:00:00	2	25860671	F	2152-01-30	2150-11-02 18:02:00	2150-11-12 13:45:00	86.83712	WHITE	2150-11-02 19:37:00	2150-11-06 17:03:17	3.89	2150-11-02 20:00:00	3	2150-11-02 12:10:00	0 day 12:50:0
3	10000690	37081114	3/11/2150 13:00:00	2	25860671	F	2152-01-30	2150-11-02 18:02:00	2150-11-12 13:45:00	86.83712	WHITE	2150-11-02 19:37:00	2150-11-06 17:03:17	3.89	2150-11-02 20:00:00	3	2150-11-02 12:10:00	0 day 12:50:0
4	10000690	37081114	3/11/2150 14:00:00	2	25860671	F	2152-01-30	2150-11-02 18:02:00	2150-11-12 13:45:00	86.83712	WHITE	2150-11-02 19:37:00	2150-11-06 17:03:17	3.89	2150-11-02 20:00:00	3	2150-11-02 12:10:00	0 day 12:50:0

In [79]:
```
kdigo_stages1.loc[:,'charttime'] = pd.to_datetime(kdigo_stages1['charttime'], dayfirst=True)
kdigo_stages1['charttime'] = kdigo_stages1['charttime'].astype('datetime64[ns]')
```

In [80]:
```
kdigo_stages1.rename(columns={'charttime':'aki_charttime'},inplace=True)
```

In [81]:
```
kdigo_stages1.shape
```

Out[81]: (1102255, 18)

In [82]:
```
kdigo_stages1.subject_id.nunique()
```

Out[82]: 24364

In [83]:
```
kdigo_stages1.stay_id.nunique()
```

Out[83]: 24364

In [84]:
```
kdigo_stages1.hadm_id.nunique()
```

Out[84]: 24364

SAKI诊断标准：1.SA-AKI定义为在脓毒症确诊后7天内发生的AKI；2.脓毒症发生后48小时内确诊的AKI为早期SA-AKI，而48小时至7天内确诊的AKI应视为晚期SA-AKI

In [85]:
```
result = kdigo_stages1.query('aki_charttime >= infection_time')
```

In [86]:
```
result = result.query('aki_charttime >=sofa_time')
```

In [87]:
```
result.shape
```

Out[87]: (1023726, 18)

In [88]:
```
result.subject_id.nunique()
```

Out[88]: 23920

In [89]:
```
result.stay_id.nunique()
```

Out[89]: 23920

In [90]:
```
result.hadm_id.nunique()
```

Out[90]: 23920

In [91]:
```
aki= result.query('aki_stage >=1')
```

In [92]:
```
aki.shape
```

Out[92]: (1023726, 18)

In [93]:
```
aki.stay_id.nunique()
```

Out[93]: 23920

In [94]:
```
aki['aki_charttime'] = pd.to_datetime(aki['aki_charttime'])
aki['infection_time'] = pd.to_datetime(aki['infection_time'])
aki['sofa_time'] = pd.to_datetime(aki['sofa_time'])

aki['max_time'] = aki[['infection_time', 'sofa_time']].max(axis=1)#确定脓毒症时间

aki['window_end'] = aki['max_time'] + pd.Timedelta(days=7)

filtered_aki = aki[(aki['aki_charttime'] > aki['max_time']) & (aki['aki_charttime'] <= aki['window_end'])]
```

In [95]:
```
filtered_aki.shape#脓毒症相关急性肾损伤数据
```

Out[95]: (757223, 20)

In [96]:
```
filtered_aki.stay_id.nunique()
```

Out[96]: 23729

In [97]:
```
min_indices = filtered_aki.groupby(['stay_id'])['aki_charttime'].idxmin()
min_aki = filtered_aki.loc[min_indices]
```

In [98]:
```
min_aki.shape
```

Out[98]: (23729, 20)

In [ ]:
```
min_aki.head()
```

In [100…
```
min_aki.subject_id.nunique()
```

Out[100… 23729

In [101…
```
min_aki.hadm_id.nunique()
```

Out[101… 23729

In [102…
```
min_aki.stay_id.nunique()
```

Out[102… 23729

In [103…
```
def assign_saki(row):
    max_time = row['max_time']
    aki_charttime = row['aki_charttime']

    if aki_charttime <= max_time + pd.Timedelta(days=2):
        return 'early'
    elif aki_charttime <= max_time + pd.Timedelta(days=7):
        return 'late'
    else:
        return None

min_aki['saki'] = min_aki.apply(assign_saki, axis=1)
```

In [104…
```
min_aki.stay_id.nunique()
```

Out[104… 23729

In [105…
```
min_aki.shape
```

Out[105… (23729, 21)

In [106…
```
min_aki.saki.value_counts()
```

Out[106…
```
saki
early    22524
late      1205
Name: count, dtype: int64
```

In [ ]:
```
min_aki.head()
```

持续性AKI

In [108…
```
cxxakimax=filtered_aki.groupby(['stay_id'])['aki_charttime'].idxmax()
cxxaki = filtered_aki.loc[cxxakimax]
```

In [109…
```
cxxaki=cxxaki[['stay_id','subject_id','aki_charttime']]
```

In [110…
```
cxxaki.shape
```

Out[110… (23729, 3)

In [111…
```
cxxaki.rename(columns={'aki_charttime':'aki_charttime_max'},inplace=True)
```

In [112…
```
akimax = pd.merge(cxxaki,min_aki,on=['stay_id','subject_id'])
```

In [113…
```
akimax.shape
```

Out[113… (23729, 22)

In [ ]:
```
akimax.head()
```

In [115…
```
akimax['los_aki'] = akimax['aki_charttime_max'] - akimax['aki_charttime']
```

In [116…
```
akimax.shape
```

Out[116… (23729, 23)

In [ ]:
```
akimax.head()
```

排除sepsis之前就已经发生AKI的数据

In [118…
```
aki_se = akimax[['subject_id', 'stay_id', 'max_time']]
```

In [119…
```
aki_se1= pd.merge(kdigo_stages1, aki_se, on=['subject_id','stay_id'])
```

In [120…
```
aki_se1.shape
```

Out[120… (1093839, 19)

In [121…
```
aki_se1 = aki_se1.query('aki_charttime < max_time')
```

In [122…
```
aki_se1.shape
```

Out[122… (74188, 19)

In [123…
```
aki_se1.subject_id.nunique()
```

Out[123… 7169

In [124…
```
aki_se1max = aki_se1.groupby(['stay_id'])['aki_charttime'].idxmax()
aki_sep = aki_se1.loc[aki_se1max]
```

In [125…
```
aki_sep.shape
```

Out[125… (7169, 19)

In [126…
```
aki_sep= aki_sep.query('aki_stage >=1')
```

In [127…
```
aki_sep.shape
```

Out[127… (7169, 19)

In [128…
```
aki_sep = aki_sep[['subject_id', 'stay_id']]
```

In [129…
```
aki_sep.shape
```

Out[129… (7169, 2)

In [130…
```
aki_sep.head()
```

Out[130…

	subject_id	stay_id
917564	18421337	30000484
624368	15726459	30000831
715541	16513856	30001446
194656	11823540	30002012
153921	11423795	30003226

In [131…
```
akimax.shape
```

Out[131… (23729, 23)

In [132…
```
min_aki_filtered = pd.merge(akimax, aki_sep, on=['subject_id', 'stay_id'], how='outer', indicator=True)
min_aki_filtered = min_aki_filtered[min_aki_filtered['_merge'] == 'left_only']
min_aki_filtered = min_aki_filtered.drop(columns=['_merge'])
min_aki_filtered.shape
```

Out[132… (16560, 23)

排除标准

In [133…
```
charlson = pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\charlson.parquet')
charlson.head()
```

Out[133…

	subject_id	hadm_id	age_score	myocardial_infarct	congestive_heart_failure	peripheral_vascular_disease	cerebrovascular_disease	dementia	chronic_pulmonary_disease	rheumatic_disease	…	mil
0	10467237	20000019	3	0	0	0	0	0	1	0	…	
1	16925328	20000024	4	0	0	0	0	0	0	0	…	
2	19430048	20000034	3	0	0	0	0	0	1	0	…	
3	18910522	20000041	2	0	0	0	0	0	0	0	…	
4	13413708	20000045	1	0	0	0	0	0	0	0	…	

5 rows × 21 columns

In [134…
```
charlson.columns
```

Out[134…
```
Index(['subject_id', 'hadm_id', 'age_score', 'myocardial_infarct',
       'congestive_heart_failure', 'peripheral_vascular_disease',
       'cerebrovascular_disease', 'dementia', 'chronic_pulmonary_disease',
       'rheumatic_disease', 'peptic_ulcer_disease', 'mild_liver_disease',
       'diabetes_without_cc', 'diabetes_with_cc', 'paraplegia',
       'renal_disease', 'malignant_cancer', 'severe_liver_disease',
       'metastatic_solid_tumor', 'aids', 'charlson_comorbidity_index'],
      dtype='object')
```

In [135…
```
print(np.intersect1d(min_aki_filtered.columns,charlson.columns))
```

```
['hadm_id' 'subject_id']
```

In [ ]:
```
dfa = pd.merge(charlson, min_aki_filtered, on=['subject_id','hadm_id'])
dfa.head()
```

In [137…
```
dfa.shape
```

Out[137…
```
(16560, 42)
```

心肌梗死，充血性心力衰竭，周围血管疾病，脑血管疾病，痴呆，慢性肺病，类风湿性疾病，消化性溃疡病，轻度肝病，糖尿病无并发症，糖尿病有并发症，截瘫，肾疾病，恶性肿瘤，严重肝病，转移性实体瘤，艾滋病，查尔森合并症指数

In [148…
```
d_icd_diagnoses1 = pd.read_parquet(r'D:\MIMIC\MIMICIV3.1_Parquet\MIMICIV_HOSP\d_icd_diagnoses.parquet')
d_icd_diagnoses1
```

Out[148…

	icd_code	icd_version	long_title
0	0010	9	Cholera due to vibrio cholerae
1	0011	9	Cholera due to vibrio cholerae el tor
2	0019	9	Cholera, unspecified
3	0020	9	Typhoid fever
4	0021	9	Paratyphoid fever A
…	…	…	…
112102	Z992	10	Dependence on renal dialysis
112103	Z993	10	Dependence on wheelchair
112104	Z998	10	Dependence on other enabling machines and devices
112105	Z9981	10	Dependence on supplemental oxygen
112106	Z9989	10	Dependence on other enabling machines and devices

112107 rows × 3 columns

In [149…
```
def id_search_mimic(disease_select_longtitle): #确定符合疾病的subject_id和hadm_id

    rows_result = d_icd_diagnoses1[d_icd_diagnoses1['long_title'].str.contains(disease_select_longtitle, case=False)]
# 获取这些行的行号
    row_indices_icd = rows_result.index.tolist()

    print("请注意，根据您提供的疾病查询关键字，\n搜索到的如果不匹配，需进一步筛选，或者重新筛选\n",d_icd_diagnoses1.loc[row_indices_icd,'long_title'])

    return d_icd_diagnoses1.loc[row_indices_icd,'icd_code']
```

In [150…
```
diagnoses = pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMICIV_HOSP\diagnoses_icd.parquet')
```

In [151…
```
diagnoses.head()
```

Out[151…

	subject_id	hadm_id	seq_num	icd_code	icd_version
0	10000032	22595853	1	5723	9
1	10000032	22595853	2	78959	9
2	10000032	22595853	3	5715	9
3	10000032	22595853	4	07070	9
4	10000032	22595853	5	496	9

In [152…
```
diagnoses.shape
```

Out[152…
```
(6364488, 5)
```

def data_filter_disease(icd_code_exclue,diagnoses_label): result = diagnoses.icd_code[~diagnoses.icd_code.isin(icd_code_exclue)].index subject_id = diagnoses.loc[result,'subject_id'] return subject_id

In [153…
```
def data_filter_disease(icd_code_exclue, diagnoses_label):
    result = diagnoses_label.icd_code[~diagnoses_label.icd_code.isin(icd_code_exclue)].index
    subject_id = diagnoses_label.loc[result, 'subject_id']
    return subject_id
```

In [154…
```
icd_code_hyper = id_search_mimic('hypertension')
```

```
请注意，根据您提供的疾病查询关键字，
搜索到的如果不匹配，需进一步筛选，或者重新筛选
 3527                        Benign intracranial hypertension
3853                                Ocular hypertension
4651                       Malignant essential hypertension
4652                          Benign essential hypertension
4653                       Unspecified essential hypertension
                                ...
39076     Unspecified maternal hypertension, unspecified...
42059                                 Neonatal hypertension
42061                     Pulmonary hypertension of newborn
43372     Elevated blood-pressure reading, without diagn...
103946                           Screening for hypertension
Name: long_title, Length: 160, dtype: object
```

In [155…
```
def retain_disease_data(icd_code_retain, diagnoses):
    # 筛选出那些包含在 icd_code_retain 列表中的 ICD 代码的行
    result = diagnoses[diagnoses['icd_code'].isin(icd_code_retain)]
    return result
```

In [156…
```
retain_hyper=retain_disease_data(icd_code_hyper, diagnoses)
```

In [157…
```
retain_hyper
```

Out[157...

	subject_id	hadm_id	seq_num	icd_code	icd_version
0	10000032	22595853	1	5723	9
91	10000635	20642640	4	I10	10
99	10000635	26134563	3	4019	9
111	10000690	23280645	12	4019	9
135	10000690	25860671	21	4019	9
...	...	...	...	...	...
6364408	19999784	29889147	8	I10	10
6364425	19999828	25744818	9	I10	10
6364444	19999828	29734428	9	I10	10
6364465	19999840	21033226	8	4019	9
6364474	19999840	26071774	5	4019	9

217853 rows × 5 columns

In [158...
```
hyper=retain_hyper[['subject_id','hadm_id']]
```

In [159...
```
hyper= hyper.drop_duplicates()
```

In [160...
```
hyper['HTN'] = 1

# 将 df1 和 hyper 根据 subject_id 和 hadm_id 进行合并
dfa = pd.merge(dfa, hyper[['subject_id', 'hadm_id', 'HTN']], on=['subject_id', 'hadm_id'], how='left')

# 填充缺失值为0，表示这些患者没有高血压
dfa['HTN'] = dfa['HTN'].fillna(0)
```

In [ ]:
```
dfa.head()
```

In [162...
```
dfa.HTN.value_counts()
```

Out[162...
```
HTN
0.0    8474
1.0    8086
Name: count, dtype: int64
```

sepsis诊断标准：1.确诊感染或疑似感染；2.SOFA评分较基线增加>=2分

In [ ]:
```
icd_code_rs = id_search_mimic('pregnancy')
subject_id_exclude_rs = data_filter_disease(icd_code_rs,diagnoses)
```

In [ ]:
```
subject_id_exclude_rs
```

In [165...
```
df_merged = dfa[dfa['subject_id'].isin(subject_id_exclude_rs)]
```

In [166...
```
df_merged.shape
```

Out[166...
```
(16557, 43)
```

In [167...
```
df_merged.columns
```

Out[167...
```
Index(['subject_id', 'hadm_id', 'age_score', 'myocardial_infarct',
       'congestive_heart_failure', 'peripheral_vascular_disease',
       'cerebrovascular_disease', 'dementia', 'chronic_pulmonary_disease',
       'rheumatic_disease', 'peptic_ulcer_disease', 'mild_liver_disease',
       'diabetes_without_cc', 'diabetes_with_cc', 'paraplegia',
       'renal_disease', 'malignant_cancer', 'severe_liver_disease',
       'metastatic_solid_tumor', 'aids', 'charlson_comorbidity_index',
       'stay_id', 'aki_charttime_max', 'aki_charttime', 'aki_stage', 'gender',
       'dod', 'admittime', 'dischtime', 'admission_age', 'race', 'icu_intime',
       'icu_outtime', 'los_icu', 'sofa_time', 'sofa_score', 'infection_time',
       'los_anti-infe', 'max_time', 'window_end', 'saki', 'los_aki', 'HTN'],
      dtype='object')
```

Chronic kidney disease, unspecified End stage renal disease Dependent on dialysis Kidney transplant status

In [ ]:
```
# 将疾病描述列表作为单个参数传递
icd_code_esrd = id_search_mimic('End stage renal disease')
subject_id_exclude_esrd = data_filter_disease(icd_code_esrd, diagnoses)
```

In [169...
```
df_merged1 = df_merged[df_merged['subject_id'].isin(subject_id_exclude_esrd)]
```

In [170...
```
df_merged1.shape
```

Out[170...
```
(16557, 43)
```

In [ ]:
```
icd_code_esrd = id_search_mimic('Chronic kidney disease, stage 5')
subject_id_exclude_esrd = data_filter_disease(icd_code_esrd, diagnoses)
```

In [172...
```
df_merged2 = df_merged1[df_merged1['subject_id'].isin(subject_id_exclude_esrd)]
```

In [173...
```
df_merged2.shape
```

Out[173...
```
(16557, 43)
```

In [ ]:
```
icd_code_esrd = id_search_mimic('uremia')
subject_id_exclude_esrd = data_filter_disease(icd_code_esrd, diagnoses)
```

In [175...
```
df_merged2 = df_merged2[df_merged2['subject_id'].isin(subject_id_exclude_esrd)]
```

In [176...
```
df_merged2.shape
```

Out[176...
```
(16557, 43)
```

In [ ]:
```
icd_code_esrd = id_search_mimic('uremic')
subject_id_exclude_esrd = data_filter_disease(icd_code_esrd, diagnoses)
```

In [178...
```
df_merged2 = df_merged2[df_merged2['subject_id'].isin(subject_id_exclude_esrd)]
```

In [179...
```
df_merged2.shape
```

Out[179...
```
(16557, 43)
```

AKD

In [180...
```
aki.shape
```

Out[180… (1023726, 20)

In [181…
```
aki.stay_id.nunique()
```

Out[181… 23920

In [182…
```
aki.columns
```

Out[182…
```
Index(['subject_id', 'stay_id', 'aki_charttime', 'aki_stage', 'hadm_id',
       'gender', 'dod', 'admittime', 'dischtime', 'admission_age', 'race',
       'icu_intime', 'icu_outtime', 'los_icu', 'sofa_time', 'sofa_score',
       'infection_time', 'los_anti-infe', 'max_time', 'window_end'],
      dtype='object')
```

In [183…
```
df_merged2.shape
```

Out[183… (16557, 43)

In [184…
```
df_merged2.stay_id.nunique()
```

Out[184… 16557

In [185…
```
print(list(df_merged2.columns))
```

```
['subject_id', 'hadm_id', 'age_score', 'myocardial_infarct', 'congestive_heart_failure', 'peripheral_vascular_disease', 'cerebrovascular_disease', 'dementia', 'chronic_pulmonary_disease', 'rheumatic_disease', 'peptic_ulcer_disease', 'mild_liver_disease', 'diabetes_without_cc', 'diabetes_with_cc', 'paraplegia', 'renal_disease', 'malignant_cancer', 'severe_liver_disease', 'metastatic_solid_tumor', 'aids', 'charlson_comorbidity_index', 'stay_id', 'aki_charttime_max', 'aki_charttime', 'aki_stage', 'gender', 'dod', 'admittime', 'dischtime', 'admission_age', 'race', 'icu_intime', 'icu_outtime', 'los_icu', 'sofa_time', 'sofa_score', 'infection_time', 'los_anti-infe', 'max_time', 'window_end', 'saki', 'los_aki', 'HTN']
```

In [ ]:
```
aki_min = df_merged2.groupby(['subject_id', 'stay_id']).agg({'aki_charttime': 'min'}).reset_index()
aki_min.rename(columns={'aki_charttime': 'min_aki_charttime'}, inplace=True)

# 步骤2: 筛选出在最小值之后7天至90天内的数据
aki_min['window_start7'] = aki_min['min_aki_charttime'] + pd.Timedelta(days=7)
aki_min['window_end90'] = aki_min['min_aki_charttime'] + pd.Timedelta(days=90)

# 合并aki_min和aki以便进行筛选
aki_merged = pd.merge(aki, aki_min, on=['subject_id', 'stay_id'])

# 筛选条件
final_filtered_aki = aki_merged[(aki_merged['aki_charttime'] > aki_merged['window_start7'])]

# 显示结果
print(final_filtered_aki.head())
```

In [187…
```
aki_min.head()
```

Out[187…

	subject_id	stay_id	min_aki_charttime	window_start7	window_end90
0	10000690	37081114	2150-11-03 09:00:00	2150-11-10 09:00:00	2151-02-01 09:00:00
1	10001843	39698942	2134-12-06 03:29:00	2134-12-13 03:29:00	2135-03-06 03:29:00
2	10001884	37510196	2131-01-12 19:00:00	2131-01-19 19:00:00	2131-04-12 19:00:00
3	10002013	39060235	2160-05-19 04:00:00	2160-05-26 04:00:00	2160-08-17 04:00:00
4	10002155	31090461	2130-09-24 18:00:00	2130-10-01 18:00:00	2130-12-23 18:00:00

In [188…
```
final_filtered_aki.shape
```

Out[188… (146716, 23)

In [ ]:
```
final_filtered_aki.head()
```

In [ ]:
```
final_filtered_aki['AKD'] = 1
```

In [ ]:
```
final_filtered_aki.head()
```

In [192…
```
akd=final_filtered_aki[['stay_id','AKD']]
```

In [193…
```
akd.shape
```

Out[193… (146716, 2)

In [ ]:
```
akd.head()
```

In [195…
```
akd.stay_id.nunique()
```

Out[195… 2498

In [196…
```
akd= akd.drop_duplicates(subset='stay_id')
```

In [197…
```
akd.shape
```

Out[197… (2498, 2)

In [198…
```
df_merged2.shape
```

Out[198… (16557, 43)

SAKI

In [199…
```
saki = df_merged2.merge(akd, on='stay_id', how='left').fillna({'AKD': 0})
```

In [200…
```
saki.shape
```

Out[200… (16557, 44)

In [201…
```
saki.AKD.value_counts()
```

Out[201…
```
AKD
0.0    14059
1.0     2498
Name: count, dtype: int64
```

In [202…
```
saki.saki.value_counts()
```

Out[202…
```
saki
early    15534
late      1023
Name: count, dtype: int64
```

In [203…
```
zakd = saki[['AKD', 'saki']]
```

In [ ]:
```
zakd.head()
```

In [205…
```
zakd.shape
```

Out[205… (16557, 2)

In [206…
```
zakd=zakd.query('AKD==1.0')
```

In [207…
```
zakd.shape
```

Out[207… (2498, 2)

In [208…
```
zakd = zakd.query('saki == "early"')
```

In [209…
```
zakd.shape
```

Out[209… (2309, 2)

In [210…
```
saki= saki.query('los_icu >= 2')
```

In [211…
```
saki.AKD.value_counts()
```

Out[211…
```
AKD
0.0    9271
1.0    2498
Name: count, dtype: int64
```

In [212…
```
saki.shape
```

Out[212… (11769, 44)

In [213…
```
saki.columns
```

Out[213…
```
Index(['subject_id', 'hadm_id', 'age_score', 'myocardial_infarct',
       'congestive_heart_failure', 'peripheral_vascular_disease',
       'cerebrovascular_disease', 'dementia', 'chronic_pulmonary_disease',
       'rheumatic_disease', 'peptic_ulcer_disease', 'mild_liver_disease',
       'diabetes_without_cc', 'diabetes_with_cc', 'paraplegia',
       'renal_disease', 'malignant_cancer', 'severe_liver_disease',
       'metastatic_solid_tumor', 'aids', 'charlson_comorbidity_index',
       'stay_id', 'aki_charttime_max', 'aki_charttime', 'aki_stage', 'gender',
       'dod', 'admittime', 'dischtime', 'admission_age', 'race', 'icu_intime',
       'icu_outtime', 'los_icu', 'sofa_time', 'sofa_score', 'infection_time',
       'los_anti-infe', 'max_time', 'window_end', 'saki', 'los_aki', 'HTN',
       'AKD'],
      dtype='object')
```

In [ ]:
```
saki.head()
```

In [215…
```
saki['los_icu-aki'] = saki['aki_charttime'] - saki['icu_intime']
```

In [216…
```
saki['window_start7'] = saki['aki_charttime'] + pd.Timedelta(days=7)
```

In [217…
```
saki['window_start-1'] = saki['aki_charttime'] - pd.Timedelta(days=1)
```

In [ ]:
```
saki.head()
```

In [ ]:
```
saki['icu_intime']
```

In [220…
```
saki['los_dod'] = saki['dod'] - saki['aki_charttime']
```

In [221…
```
def convert_to_hours(time_str):
    try:
        # 将字符串转为 Timedelta 类型
        td = pd.to_timedelta(time_str)
        # 计算总秒数，再除以 3600 得到小时
        return td.total_seconds() / 3600
    except:
        return None  # 异常值处理（如空值或无效格式）

# 应用转换函数到指定列
saki['los_dod'] = saki['los_dod'].apply(convert_to_hours)/24
```

In [222…
```
saki = saki.query('los_dod > 7 or los_dod.isnull()')
```

In [223…
```
saki.shape
```

Out[223… (10269, 48)

In [224…
```
saki.subject_id.nunique()
```

Out[224… 10269

In [225…
```
saki.stay_id.nunique()
```

Out[225… 10269

In [226…
```
saki.hadm_id.nunique()
```

Out[226… 10269

In [227…
```
saki.AKD.value_counts()
```

Out[227…
```
AKD
0.0    7795
1.0    2474
Name: count, dtype: int64
```

实验室指标

临床参数：体温、心率、呼吸频率、收缩压、脉压、是否用升压药、吸入氧浓度、吸氧流量、GCS评分、SOFA评分（或qSOFA评分）

In [ ]:
```
bg = pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\bg.parquet')
bg.head()
```

In [229…
```
bg.shape
```

Out[229… (697418, 27)

In [230…
```
bg1 = bg.query('specimen == "ART."')
bg1.shape
```

Out[230… (514972, 27)

In [231…
```
bg1 =bg1.drop(['specimen'], axis=1)
```

In [232…
```
bg1 = bg1.dropna(subset=['hadm_id'])
bg1['hadm_id'] = bg1['hadm_id'].astype('int64')
```

In [233…
```
bg1['charttime'] = pd.to_datetime(bg1['charttime'], dayfirst=True)
bg1['charttime'] = bg1['charttime'].astype('datetime64[ns]')
```

In [234…
```
print(np.intersect1d(saki.columns,bg1.columns))
```

```
['hadm_id' 'subject_id']
```

In [235…
```
print(list(bg1.columns))
```

```
['subject_id', 'hadm_id', 'charttime', 'so2', 'po2', 'pco2', 'fio2_chartevents', 'fio2', 'aado2', 'aado2_calc', 'pao2fio2ratio', 'ph', 'baseexcess', 'bicarbonate', 'totalco2', 'hematocrit', 'h
emoglobin', 'carboxyhemoglobin', 'methemoglobin', 'chloride', 'calcium', 'temperature', 'potassium', 'sodium', 'lactate', 'glucose']
```

In [ ]:
```
import matplotlib.pyplot as plt
from missingno import missingno as msno
msno.bar(bg1)
```

In [237…
```
# 计算每列的缺失率
missing_rate = bg1.isnull().mean()

# 找出缺失率大于30%的列
cols_to_drop = missing_rate[missing_rate > 0.50].index

cols_to_drop
# 显示结果
```

Out[237…
```
Index(['so2', 'fio2', 'aado2', 'bicarbonate', 'hematocrit', 'hemoglobin',
       'carboxyhemoglobin', 'methemoglobin', 'chloride', 'temperature',
       'potassium', 'sodium', 'glucose'],
      dtype='object')
```

In [238…
```
bg1.drop(cols_to_drop, axis=1, inplace=True)
```

In [239…
```
bg1.drop(['fio2_chartevents','calcium'], axis=1, inplace=True)
```

In [240…
```
bg1.head()
```

Out[240…

	subject_id	hadm_id	charttime	po2	pco2	aado2_calc	pao2fio2ratio	ph	baseexcess	totalco2	lactate
1	10000690	25860671	2150-11-08 05:43:00	68	52.0	NaN	NaN	7.45	9.0	37.0	NaN
2	10000935	25849114	2187-10-22 15:40:00	86	33.0	NaN	NaN	7.41	-2.0	22.0	2.8
14	10001884	26184834	2131-01-10 13:15:00	72	49.0	NaN	NaN	7.42	5.0	33.0	2.0
17	10001884	26184834	2131-01-11 03:42:00	74	94.0	NaN	NaN	7.22	6.0	40.0	NaN
23	10001884	26184834	2131-01-12 21:04:00	65	60.0	145.2	162.5	7.38	7.0	37.0	1.1

In [241…
```
saki1=saki[['subject_id','hadm_id','window_start-1','window_start7']]
```

In [242…
```
bg2 = pd.merge(bg1, saki1, on=['subject_id','hadm_id'],how='inner')
```

In [243…
```
bg2.head()
```

Out[243…

	subject_id	hadm_id	charttime	po2	pco2	aado2_calc	pao2fio2ratio	ph	baseexcess	totalco2	lactate	window_start-1	window_start7
0	10000690	25860671	2150-11-08 05:43:00	68	52.0	NaN	NaN	7.45	9.0	37.0	NaN	2150-11-02 09:00:00	2150-11-10 09:00:00
1	10001884	26184834	2131-01-10 13:15:00	72	49.0	NaN	NaN	7.42	5.0	33.0	2.0	2131-01-11 19:00:00	2131-01-19 19:00:00
2	10001884	26184834	2131-01-11 03:42:00	74	94.0	NaN	NaN	7.22	6.0	40.0	NaN	2131-01-11 19:00:00	2131-01-19 19:00:00
3	10001884	26184834	2131-01-12 21:04:00	65	60.0	145.20	162.5	7.38	7.0	37.0	1.1	2131-01-11 19:00:00	2131-01-19 19:00:00
4	10001884	26184834	2131-01-13 02:28:00	69	53.0	149.95	172.5	7.42	7.0	36.0	1.2	2131-01-11 19:00:00	2131-01-19 19:00:00

In [244…
```
bg2.hadm_id.nunique()
```

Out[244…
```
8075
```

In [245…
```
condition = (bg2['window_start-1']  < bg2['charttime']) & \
            (bg2['charttime'] < bg2['window_start7'])
bg3 = bg2[condition]
```

In [246…
```
bg3.hadm_id.nunique()
```

Out[246…
```
7388
```

In [247…
```
bg3.head()
```

Out[247…

	subject_id	hadm_id	charttime	po2	pco2	aado2_calc	pao2fio2ratio	ph	baseexcess	totalco2	lactate	window_start-1	window_start7
0	10000690	25860671	2150-11-08 05:43:00	68	52.0	NaN	NaN	7.45	9.0	37.0	NaN	2150-11-02 09:00:00	2150-11-10 09:00:00
3	10001884	26184834	2131-01-12 21:04:00	65	60.0	145.20	162.500000	7.38	7.0	37.0	1.1	2131-01-11 19:00:00	2131-01-19 19:00:00
4	10001884	26184834	2131-01-13 02:28:00	69	53.0	149.95	172.500000	7.42	7.0	36.0	1.2	2131-01-11 19:00:00	2131-01-19 19:00:00
5	10001884	26184834	2131-01-14 07:05:00	91	49.0	132.95	227.500000	7.46	9.0	36.0	NaN	2131-01-11 19:00:00	2131-01-19 19:00:00
6	10002155	28994087	2130-09-24 09:25:00	69	37.0	547.84	74.193548	7.37	-3.0	22.0	NaN	2130-09-23 18:00:00	2130-10-01 18:00:00

In [ ]:
```
min_chartoffsets = bg3.groupby('hadm_id')['charttime'].min().reset_index()

# 步骤2和3: 检查缺失值并替换
def fill_missing_values(row, df):
    for col in ['po2','pco2','aado2_calc','pao2fio2ratio','ph','baseexcess','totalco2','lactate']:
        if pd.isna(row[col]):
            # 获取该patientunitstayid的下一个chartoffset的行
            next_rows = df[(df['hadm_id'] == row['hadm_id']) & (df['charttime'] > row['charttime'])]
            for _, next_row in next_rows.iterrows():
                if not pd.isna(next_row[col]):
                    row[col] = next_row[col]
                    break
    return row

# 应用函数
result_list = []

for _, row in min_chartoffsets.iterrows():
    patient_data =bg3[bg3['hadm_id'] == row['hadm_id']]
    min_chartoffset_row = patient_data[patient_data['charttime'] == row['charttime']].iloc[0]
    filled_row = fill_missing_values(min_chartoffset_row, patient_data)
    result_list.append(filled_row)

# 使用concat合并DataFrame
bg4= pd.concat(result_list, axis=1).transpose()
```

In [ ]:
```
bg4.hadm_id.nunique()
```

Out[ ]: 7388

In [ ]:
```
bg5 = bg4.drop(columns=['charttime', 'window_start-1', 'window_start7'])
```

In [ ]:
```
print(np.intersect1d(saki.columns,bg5.columns))
```

```
['hadm_id' 'subject_id']
```

In [ ]:
```
df2= saki.merge(bg5, on=['subject_id','hadm_id'], how='left')
```

In [ ]:
```
df2.hadm_id.nunique()
```

Out[ ]: 10269

In [ ]:
```
df2.shape
```

Out[ ]: (10269, 55)

In [ ]:
```
blood_differential = pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\blood_differential.parquet')
blood_differential.head()
```

In [ ]:
```
blood_differential =blood_differential.drop(['specimen_id'], axis=1)
blood_differential = blood_differential.dropna(subset=['hadm_id'])
blood_differential['hadm_id'] = blood_differential['hadm_id'].astype('int64')
blood_differential['charttime'] = pd.to_datetime(blood_differential['charttime'], dayfirst=True)
blood_differential['charttime'] = blood_differential['charttime'].astype('datetime64[ns]')
print(np.intersect1d(df2.columns,blood_differential.columns))
```

```
['hadm_id' 'subject_id']
```

In [ ]:
```
blood_differential2=blood_differential[['subject_id','hadm_id','charttime','wbc','basophils_abs','eosinophils_abs','lymphocytes_abs','monocytes_abs','neutrophils_abs']]
```

In [ ]:
```
blood_differential2.head()
```

Out[ ]:

	subject_id	hadm_id	charttime	wbc	basophils_abs	eosinophils_abs	lymphocytes_abs	monocytes_abs	neutrophils_abs
3	15637323	21539156	2128-10-07 05:16:00	27.9	NaN	NaN	NaN	NaN	NaN
4	11375469	26645160	2128-07-05 06:21:00	4.8	NaN	NaN	NaN	NaN	NaN
5	16182726	26238489	2171-09-05 06:30:00	7.7	NaN	NaN	NaN	NaN	NaN
7	19881575	29284557	2124-06-09 06:15:00	6.3	NaN	NaN	NaN	NaN	NaN
8	12288954	24694399	2124-02-15 13:20:00	10.6	NaN	NaN	NaN	NaN	NaN

In [ ]:
```
missing_per_row = blood_differential2.isnull().mean(axis=1)

# 保留缺失值比例<=50%的行
blood_differentialv= blood_differential2[missing_per_row <= 0.5]
```

In [ ]:
```
blood_differentialv.head()
```

Out[ ]:

	subject_id	hadm_id	charttime	wbc	basophils_abs	eosinophils_abs	lymphocytes_abs	monocytes_abs	neutrophils_abs
14	19381331	27606319	2125-06-21 06:50:00	13.6	0.0136	2.9784	1.1968	0.408	9.0168
16	11111901	26267238	2164-04-12 00:57:00	0.1	0.0000	0.0000	0.0300	0.000	0.0700
27	17468647	21124037	2123-11-18 06:30:00	7.9	0.0200	0.0700	0.9200	0.680	6.1100
31	18286929	23683319	2153-04-29 05:45:00	13.3	0.0700	0.0300	1.4400	0.730	10.8000
34	19933583	29391119	2144-01-13 16:28:00	19.0	0.0760	0.1900	1.2730	0.608	16.8530

In [ ]:
```
blood_differentialv.hadm_id.nunique()
```

Out[ ]: 183919

In [ ]:
```
blood_differentialv1= pd.merge(blood_differentialv, saki1, on=['subject_id','hadm_id'],how='inner')
```

In [ ]:
```
condition = (blood_differentialv1['window_start-1']  <blood_differentialv1['charttime']) & \
            (blood_differentialv1['charttime'] < blood_differentialv1['window_start7'])
blood_differentialv2 = blood_differentialv1[condition]
```

In [ ]:
```
blood_differentialv3 =blood_differentialv2.groupby('hadm_id')['charttime'].idxmin()
blood_differentialv4= blood_differentialv2.loc[blood_differentialv3]
blood_differentialv4.shape
```

Out[ ]: (6254, 11)

In [ ]:
```
blood_differentialv5= blood_differentialv4.drop(columns=['charttime', 'window_start-1', 'window_start7'])
print(np.intersect1d(df2.columns,blood_differentialv5.columns))
```

```
['hadm_id' 'subject_id']
```

In [ ]:
```
df3= df2.merge(blood_differentialv5, on=['subject_id','hadm_id'], how='left')
```

In [ ]:
```
cardiac_marker = pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\cardiac_marker.parquet')
cardiac_marker.head()
```

Out[ ]:

	subject_id	hadm_id	charttime	specimen_id	troponin_t	ck_mb	ntprobnp
0	13448204	24480863.0	1/12/2164 13:10:00	538	NaN	3.0	NaN
1	12559662	27181478.0	19/9/2151 06:10:00	551	NaN	2.0	NaN
2	17932295	27965144.0	29/5/2125 16:50:00	672	0.09	6.0	NaN
3	17838140	26310561.0	25/6/2122 17:09:00	898	NaN	5.0	NaN
4	14361990	NaN	28/11/2161 13:40:00	1125	0.04	3.0	NaN

In [ ]:
```
cardiac_marker =cardiac_marker.drop(['specimen_id'], axis=1)
cardiac_marker = cardiac_marker.dropna(subset=['hadm_id'])
cardiac_marker['hadm_id'] = cardiac_marker['hadm_id'].astype('int64')
cardiac_marker['charttime'] = pd.to_datetime(cardiac_marker['charttime'], dayfirst=True)
cardiac_marker['charttime'] = cardiac_marker['charttime'].astype('datetime64[ns]')
print(np.intersect1d(df3.columns,cardiac_marker.columns))
```

```
['hadm_id' 'subject_id']
```

In [ ]:
```
merged_df = pd.merge(cardiac_marker, df3, on=['subject_id','hadm_id'],how='left')
condition = (merged_df['window_start-1']  < merged_df['charttime']) & \
            (merged_df['charttime'] < merged_df['window_start7'])
filtered_df = merged_df[condition]
filtered_df= filtered_df.groupby('hadm_id')['charttime'].min().reset_index()
cardiac_marker1= filtered_df.merge(cardiac_marker, on=['hadm_id','charttime'], how='left')
cardiac_marker1 = cardiac_marker1.drop_duplicates(subset=['hadm_id', 'charttime'])
```

In [ ]:
```
cardiac_marker1 =cardiac_marker1 .drop(['charttime'], axis=1)
df4 = df3.merge(cardiac_marker1 , on=['subject_id','hadm_id'], how='left')
df4.shape
```

Out[ ]: (10269, 64)

In [ ]:
```
df4.head()
```

Out[ ]:

	subject_id	hadm_id	age_score	myocardial_infarct	congestive_heart_failure	peripheral_vascular_disease	cerebrovascular_disease	dementia	chronic_pulmonary_disease	rheumatic_disease	...	lact
0	14577567	20001361	0	0	0	0	0	0	0	0	...	
1	10117812	20001770	0	0	0	0	0	0	0	0	...	N
2	18953418	20003543	2	0	0	0	0	0	0	0	...	N
3	19657904	20004357	3	0	1	0	0	0	1	0	...	
4	17912752	20005024	3	0	0	0	0	0	0	0	...	

5 rows × 64 columns

In [ ]:
```
chemistry = pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\chemistry.parquet')
chemistry.head()
```

Out[ ]:

	subject_id	hadm_id	charttime	specimen_id	albumin	globulin	total_protein	aniongap	bicarbonate	bun	calcium	chloride	creatinine	glucose	sodium	potassium
0	17518964	20219416.0	11/2/2143 06:07:00	15	NaN	NaN	NaN	14.0	24.0	13.0	8.8	103.0	1.3	134.0	141.0	3.0
1	17208408	NaN	16/8/2190 17:50:00	18	NaN	NaN	NaN	13.0	23.0	23.0	NaN	106.0	0.7	100.0	142.0	4.6
2	15314300	25539343.0	17/12/2126 16:15:00	22	NaN	NaN	NaN	14.0	22.0	10.0	7.8	103.0	0.7	96.0	139.0	4.8
3	19681724	NaN	17/11/2182 16:30:00	69	NaN	NaN	NaN	18.0	21.0	29.0	NaN	100.0	1.2	115.0	139.0	4.7
4	19810410	NaN	21/1/2170 12:30:00	80	NaN	NaN	NaN	18.0	26.0	54.0	9.4	99.0	1.6	NaN	140.0	3.4

In [ ]:
```
chemistry =chemistry.drop(['specimen_id'], axis=1)
chemistry = chemistry.dropna(subset=['hadm_id'])
chemistry['hadm_id'] = chemistry['hadm_id'].astype('int64')
chemistry['charttime'] = pd.to_datetime(chemistry['charttime'], dayfirst=True)
chemistry['charttime'] = chemistry['charttime'].astype('datetime64[ns]')
print(np.intersect1d(df4.columns,chemistry.columns))
```

```
['hadm_id' 'subject_id']
```

In [ ]:
```
msno.bar(chemistry)
```

In [ ]:
```
missing_rate = chemistry.isnull().mean()

# 找出缺失率大于30%的列
cols_to_drop = missing_rate[missing_rate > 0.50].index

cols_to_drop
```

Out[ ]: Index(['albumin', 'globulin', 'total_protein'], dtype='object')

In [ ]:
```
chemistry.drop(cols_to_drop, axis=1, inplace=True)
```

In [ ]:
```
chemistry2 = pd.merge(chemistry,saki1, on=['subject_id','hadm_id'],how='inner')
```

In [ ]:
```
chemistry2.hadm_id.nunique()
```

Out[ ]: 10269

In [ ]:
```
condition = (chemistry2['window_start-1']  < chemistry2['charttime']) & \
            (chemistry2['charttime'] < chemistry2['window_start7'])
chemistry3 = chemistry2[condition]
```

In [ ]:
```
chemistry3.hadm_id.nunique()
```

Out[ ]: 10262

In [ ]:
```
min_chartoffsets = chemistry3.groupby('hadm_id')['charttime'].min().reset_index()

# 步骤2和3: 检查缺失值并替换
def fill_missing_values(row, df):
    for col in ['aniongap','bicarbonate','bun','calcium','chloride','creatinine','glucose','sodium','potassium']:
        if pd.isna(row[col]):
            # 获取该patientunitstayid的下一个chartoffset的行
            next_rows = df[(df['hadm_id'] == row['hadm_id']) & (df['charttime'] > row['charttime'])]
            for _, next_row in next_rows.iterrows():
                if not pd.isna(next_row[col]):
                    row[col] = next_row[col]
                    break
    return row

# 应用函数
result_list = []

for _, row in min_chartoffsets.iterrows():
    patient_data =chemistry3[chemistry3['hadm_id'] == row['hadm_id']]
    min_chartoffset_row = patient_data[patient_data['charttime'] == row['charttime']].iloc[0]
    filled_row = fill_missing_values(min_chartoffset_row, patient_data)
    result_list.append(filled_row)

# 使用concat合并DataFrame
chemistry4= pd.concat(result_list, axis=1).transpose()
```

In [ ]:
```
chemistry4.hadm_id.nunique()
```

Out[ ]: 10262

In [ ]:
```
chemistry5= chemistry4.drop(columns=['charttime', 'window_start-1', 'window_start7'])
df5 = df4.merge(chemistry5 , on=['subject_id','hadm_id'], how='left')
df5.shape
```

Out[ ]: (10269, 73)

In [ ]:
```
coagulation = pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\coagulation.parquet')
coagulation.head()
```

Out[ ]:

	subject_id	hadm_id	charttime	specimen_id	d_dimer	fibrinogen	thrombin	inr	pt	ptt
0	10522669	NaN	28/7/2169 16:59:00	21	NaN	NaN	NaN	1.0	11.2	NaN
1	15560336	NaN	11/12/2170 09:24:00	120	NaN	NaN	NaN	1.0	11.1	NaN
2	12664423	28224503.0	29/10/2165 13:58:00	146	NaN	NaN	NaN	1.2	13.2	27.5
3	14169173	NaN	30/12/2175 03:34:00	184	NaN	NaN	NaN	1.5	15.9	25.2
4	10397642	NaN	7/6/2173 09:35:00	206	NaN	NaN	NaN	1.1	11.7	32.3

In [ ]:
```
coagulation =coagulation.drop(['specimen_id'], axis=1)
coagulation = coagulation.dropna(subset=['hadm_id'])
```

```
coagulation['hadm_id'] = coagulation['hadm_id'].astype('int64')
coagulation['charttime'] = pd.to_datetime(coagulation['charttime'], dayfirst=True)
coagulation['charttime'] = coagulation['charttime'].astype('datetime64[ns]')
print(np.intersect1d(df5.columns,coagulation.columns))
```

```
['hadm_id' 'subject_id']
```

In [ ]:
```
msno.bar(coagulation)
```

In [ ]:
```
missing_rate = coagulation.isnull().mean()
cols_to_drop = missing_rate[missing_rate > 0.50].index
cols_to_drop
```

Out[ ]:
```
Index(['d_dimer', 'fibrinogen', 'thrombin'], dtype='object')
```

In [ ]:
```
coagulation.drop(cols_to_drop, axis=1, inplace=True)
```

In [ ]:
```
coagulation2 = pd.merge(coagulation,saki1, on=['subject_id','hadm_id'],how='inner')
```

In [ ]:
```
coagulation2 .hadm_id.nunique()
```

Out[ ]:
```
10079
```

In [ ]:
```
condition = (coagulation2 ['window_start-1']  < coagulation2 ['charttime']) & \
            (coagulation2 ['charttime'] < coagulation2 ['window_start7'])
coagulation3 = coagulation2 [condition]
```

In [ ]:
```
coagulation3.hadm_id.nunique()
```

Out[ ]:
```
9816
```

In [ ]:
```
min_chartoffsets = coagulation3.groupby('hadm_id')['charttime'].min().reset_index()

# 步骤2和3: 检查缺失值并替换
def fill_missing_values(row, df):
    for col in ['inr','pt','ptt']:
        if pd.isna(row[col]):
            # 获取该patientunitstayid的下一个chartoffset的行
            next_rows = df[(df['hadm_id'] == row['hadm_id']) & (df['charttime'] > row['charttime'])]
            for _, next_row in next_rows.iterrows():
                if not pd.isna(next_row[col]):
                    row[col] = next_row[col]
                    break
    return row

# 应用函数
result_list = []

for _, row in min_chartoffsets.iterrows():
    patient_data =coagulation3[coagulation3['hadm_id'] == row['hadm_id']]
    min_chartoffset_row = patient_data[patient_data['charttime'] == row['charttime']].iloc[0]
    filled_row = fill_missing_values(min_chartoffset_row, patient_data)
    result_list.append(filled_row)

# 使用concat合并DataFrame
coagulation4= pd.concat(result_list, axis=1).transpose()
```

In [ ]:
```
coagulation4.hadm_id.nunique()
```

Out[ ]:
```
9816
```

In [ ]:
```
coagulation5= coagulation4.drop(columns=['charttime', 'window_start-1', 'window_start7'])
df6= df5.merge(coagulation5, on=['subject_id','hadm_id'], how='left')
df6.shape
```

Out[ ]:
```
(10269, 76)
```

In [ ]:
```
complete_blood = pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\complete_blood_count.parquet')
complete_blood.head()
```

Out[ ]:

	subject_id	hadm_id	charttime	specimen_id	hematocrit	hemoglobin	mch	mchc	mcv	platelet	rbc	rdw	rdwsd	wbc
0	19050791	NaN	25/2/2151 13:35:00	31	38.1	12.8	32.7	33.6	97.0	97.0	3.92	16.6	None	4.2
1	19903398	NaN	24/7/2149 12:50:00	41	38.8	12.6	28.2	32.4	87.0	314.0	4.46	12.7	None	9.1
2	11641458	NaN	19/9/2146 09:10:00	47	35.8	12.3	30.8	34.4	90.0	121.0	3.99	12.9	None	4.5
3	15637323	21539156.0	7/10/2128 05:16:00	48	24.9	8.3	31.7	33.3	95.0	86.0	2.62	18.6	None	27.9
4	11375469	26645160.0	5/7/2128 06:21:00	57	28.5	8.9	39.6	31.2	127.0	45.0	2.25	17.4	None	4.8

In [ ]:
```
complete_blood =complete_blood.drop(['specimen_id'], axis=1)
complete_blood = complete_blood.dropna(subset=['hadm_id'])
complete_blood['hadm_id'] = complete_blood['hadm_id'].astype('int64')
complete_blood['charttime'] = pd.to_datetime(complete_blood['charttime'], dayfirst=True)
complete_blood['charttime'] = complete_blood['charttime'].astype('datetime64[ns]')
print(np.intersect1d(df6.columns,complete_blood.columns))
```

```
['hadm_id' 'subject_id' 'wbc']
```

In [ ]:
```
complete_blood.rename(columns={'wbc': 'WBC1'}, inplace=True)
```

In [ ]:
```
msno.bar(complete_blood)
```

In [ ]:
```
complete_blood1=complete_blood.drop(['rdwsd'], axis=1)
```

In [ ]:
```
merged_df = pd.merge(complete_blood1, df6, on=['subject_id','hadm_id'],how='left')
condition = (merged_df['window_start-1']  < merged_df['charttime']) & \
            (merged_df['charttime'] < merged_df['window_start7'])
filtered_df = merged_df[condition]
filtered_df= filtered_df.groupby('hadm_id')['charttime'].min().reset_index()
complete_blood1= filtered_df.merge(complete_blood1, on=['hadm_id','charttime'], how='left')
complete_blood1 = complete_blood1.drop_duplicates(subset=['hadm_id', 'charttime'])
```

In [ ]:
```
complete_blood1 =complete_blood1.drop(['charttime'], axis=1)
df7= df6.merge(complete_blood1, on=['subject_id','hadm_id'], how='left')
df7.shape
```

Out[ ]:
```
(10269, 85)
```

In [ ]:
```
gcs = pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\gcs.parquet')
gcs.head()
```

Out[ ]:

	subject_id	stay_id	charttime	gcs	gcs_motor	gcs_verbal	gcs_eyes	gcs_unable
0	12466550	30000153	29/9/2174 12:45:00	15	5.0	0.0	3.0	1
1	12466550	30000153	29/9/2174 16:26:00	15	6.0	0.0	4.0	1
2	12466550	30000153	29/9/2174 17:37:00	15	6.0	0.0	4.0	1
3	12466550	30000153	29/9/2174 18:00:00	9	5.0	1.0	3.0	0
4	12466550	30000153	29/9/2174 19:00:00	9	5.0	1.0	3.0	0

In [ ]:
```
gcs1 =gcs.drop(['gcs_motor','gcs_verbal','gcs_eyes','gcs_unable'], axis=1)
```

In [ ]:
```
gcs1['charttime'] = pd.to_datetime(gcs1['charttime'], dayfirst=True)
gcs1['charttime'] = gcs1['charttime'].astype('datetime64[ns]')
print(np.intersect1d(df7.columns,gcs1.columns))
```

```
['stay_id' 'subject_id']
```

In [ ]:
```
merged_df = pd.merge(gcs1, df7, on=['subject_id','stay_id'],how='left')
condition = (merged_df['window_start-1']  < merged_df['charttime']) & \
            (merged_df['charttime'] < merged_df['window_start7'])
filtered_df = merged_df[condition]
filtered_df= filtered_df.groupby('stay_id')['charttime'].min().reset_index()
gcs12= filtered_df.merge(gcs1, on=['stay_id','charttime'], how='left')
gcs12= gcs12.drop_duplicates(subset=['stay_id', 'charttime'])
```

In [ ]:
```
gcs12=gcs12.drop(['charttime'], axis=1)
df8= df7.merge(gcs12, on=['subject_id','stay_id'], how='left')
df8.shape
```

Out[ ]:
```
(10269, 86)
```

In [ ]:
```
inflammation= pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\inflammation.parquet')
inflammation.head()
```

Out[ ]:

	subject_id	hadm_id	charttime	specimen_id	crp
0	19681724	NaN	17/11/2182 16:30:00	69	197.4
1	10725595	NaN	4/11/2163 15:35:00	619	12.7
2	11932181	NaN	8/4/2141 12:20:00	841	43.0
3	17847714	NaN	16/9/2136 15:54:00	1484	127.4
4	19010070	NaN	16/10/2125 13:45:00	1570	16.9

In [ ]:
```
inflammation =inflammation.drop(['specimen_id'], axis=1)
inflammation = inflammation.dropna(subset=['hadm_id'])
inflammation['hadm_id'] = inflammation['hadm_id'].astype('int64')
inflammation['charttime'] = pd.to_datetime(inflammation['charttime'], dayfirst=True)
inflammation['charttime'] = inflammation['charttime'].astype('datetime64[ns]')
print(np.intersect1d(df8.columns,inflammation.columns))
```

```
['hadm_id' 'subject_id']
```

In [ ]:
```
merged_df = pd.merge(inflammation, df8, on=['subject_id','hadm_id'],how='left')
condition = (merged_df['window_start-1']  < merged_df['charttime']) & \
            (merged_df['charttime'] < merged_df['window_start7'])
filtered_df = merged_df[condition]
filtered_df= filtered_df.groupby('hadm_id')['charttime'].min().reset_index()
inflammation1= filtered_df.merge(inflammation, on=['hadm_id','charttime'], how='left')
inflammation1 = inflammation1.drop_duplicates(subset=['hadm_id', 'charttime'])
```

In [ ]:
```
inflammation1=inflammation1.drop(['charttime'], axis=1)
df9= df8.merge(inflammation1, on=['subject_id','hadm_id'], how='left')
df9.shape
```

Out[ ]:
```
(10269, 87)
```

In [ ]:
```
vitalsign= pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\vitalsign.parquet')
vitalsign.head()
```

Out[ ]:

	subject_id	stay_id	charttime	heart_rate	sbp	dbp	mbp	sbp_ni	dbp_ni	mbp_ni	resp_rate	temperature	temperature_site	spo2	glucose
0	10000032	39553978	23/7/2180 14:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	37.06	Oral	NaN	NaN
1	10000032	39553978	23/7/2180 14:11:00	NaN	84.0	48.0	56.0	84.0	48.0	56.0	NaN	NaN	None	NaN	NaN
2	10000032	39553978	23/7/2180 14:12:00	91.0	NaN	NaN	NaN	NaN	NaN	NaN	24.0	NaN	None	NaN	NaN
3	10000032	39553978	23/7/2180 14:13:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	None	98.0	NaN
4	10000032	39553978	23/7/2180 14:30:00	93.0	95.0	59.0	67.0	95.0	59.0	67.0	21.0	NaN	None	97.0	NaN

In [ ]:
```
vitalsign.shape
```

Out[ ]:
```
(13519533, 15)
```

In [ ]:
```
vitalsign=vitalsign.drop(['glucose','sbp_ni','dbp_ni','mbp_ni','temperature_site'], axis=1)
```

In [ ]:
```
vitalsign.head()
```

Out[ ]:

	subject_id	stay_id	charttime	heart_rate	sbp	dbp	mbp	resp_rate	temperature	spo2
0	10000032	39553978	23/7/2180 14:00:00	NaN	NaN	NaN	NaN	NaN	37.06	NaN
1	10000032	39553978	23/7/2180 14:11:00	NaN	84.0	48.0	56.0	NaN	NaN	NaN
2	10000032	39553978	23/7/2180 14:12:00	91.0	NaN	NaN	NaN	24.0	NaN	NaN
3	10000032	39553978	23/7/2180 14:13:00	NaN	NaN	NaN	NaN	NaN	NaN	98.0
4	10000032	39553978	23/7/2180 14:30:00	93.0	95.0	59.0	67.0	21.0	NaN	97.0

In [ ]:
```
unique_patientunitstayid_df14 = df9['stay_id'].unique()
vitalsign= vitalsign[vitalsign['stay_id'].isin(unique_patientunitstayid_df14)]
vitalsign.shape
```

Out[ ]:
```
(2967455, 10)
```

In [ ]:
```
vitalsign_cleaned = vitalsign
```

In [ ]:
```
vitalsign_cleaned.shape
```

Out[ ]:
```
(2967455, 10)
```

In [ ]:
```
print(np.intersect1d(df9.columns,vitalsign_cleaned.columns))
```

```
['stay_id' 'subject_id']
```

In [ ]:
```
vitalsign_cleaned['charttime'] = pd.to_datetime(vitalsign_cleaned['charttime'], dayfirst=True)
vitalsign_cleaned['charttime'] = vitalsign_cleaned['charttime'].astype('datetime64[ns]')
```

```
print(np.intersect1d(df9.columns,vitalsign_cleaned.columns))
```

```
['stay_id' 'subject_id']
```

In [ ]:
```
merged_df = pd.merge(vitalsign_cleaned, df9, on=['subject_id','stay_id'],how='left')
condition = (merged_df['window_start-1']  < merged_df['charttime']) & \
            (merged_df['charttime'] < merged_df['window_start7'])
filtered_df = merged_df[condition]
```

In [ ]:
```
min_chartoffsets = filtered_df.groupby('stay_id')['charttime'].min().reset_index()

# 步骤2和3: 检查缺失值并替换
def fill_missing_values(row, df):
    for col in ['heart_rate', 'sbp', 'dbp', 'mbp', 'resp_rate', 'temperature','spo2']:
        if pd.isna(row[col]):
            # 获取该patientunitstayid的下一个chartoffset的行
            next_rows = df[(df['stay_id'] == row['stay_id']) & (df['charttime'] > row['charttime'])]
            for _, next_row in next_rows.iterrows():
                if not pd.isna(next_row[col]):
                    row[col] = next_row[col]
                    break
    return row

# 应用函数
result_list = []

for _, row in min_chartoffsets.iterrows():
    patient_data =filtered_df[filtered_df['stay_id'] == row['stay_id']]
    min_chartoffset_row = patient_data[patient_data['charttime'] == row['charttime']].iloc[0]
    filled_row = fill_missing_values(min_chartoffset_row, patient_data)
    result_list.append(filled_row)

# 使用concat合并DataFrame
df10= pd.concat(result_list, axis=1).transpose()
```

In [ ]:
```
df10.shape
```

Out[ ]:
```
(10269, 95)
```

In [ ]:
```
df11=df10.drop(['charttime'], axis=1)
```

In [ ]:
```
height= pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\height.parquet')
height.head()
```

Out[ ]:

	subject_id	stay_id	charttime	height
0	10000032	39553978	23/7/2180 12:36:00	152.0
1	10001725	31205490	11/4/2110 15:52:00	157.0
2	10001884	37510196	11/1/2131 04:20:00	157.0
3	10002013	39060235	18/5/2160 10:00:00	157.0
4	10002114	34672098	17/2/2162 22:33:00	173.0

In [ ]:
```
height1= height.drop(['charttime'], axis=1)
```

In [ ]:
```
height1= height1.drop_duplicates(subset=['stay_id'])
```

In [ ]:
```
df12 = pd.merge(height1,df11, on=['subject_id','stay_id'],how='right')
df12.shape
```

Out[ ]:
```
(10269, 95)
```

In [ ]:
```
weight= pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\weight_durations.parquet')
weight.head()
```

Out[ ]:

	stay_id	starttime	endtime	weight	weight_type
0	30000153	29/9/2174 10:09:00	29/9/2174 16:00:00	70.0	admit
1	30000153	29/9/2174 16:00:00	1/10/2174 05:26:10	73.0	daily
2	30000213	21/6/2162 03:38:00	22/6/2162 00:00:00	84.7	admit
3	30000213	22/6/2162 00:00:00	22/6/2162 22:52:48	73.7	daily
4	30000484	14/1/2136 15:23:32	17/1/2136 06:53:08	68.5	admit

In [ ]:
```
weight1= weight.drop(['starttime','endtime','weight_type'], axis=1)
```

In [ ]:
```
weight1=weight1.drop_duplicates(subset=['stay_id'])
```

In [ ]:
```
weight1.shape
```

Out[ ]:
```
(91163, 2)
```

In [ ]:
```
weight1['stay_id'] = weight1['stay_id'].astype(str)
df12['stay_id'] = df12['stay_id'].astype(str)
df13 = pd.merge(weight1, df12, on=['stay_id'], how='right')
df13.shape
```

Out[ ]:
```
(10269, 96)
```

In [ ]:
```
df13.head()
```

Out[ ]:

	stay_id	weight	subject_id	height	heart_rate	sbp	dbp	mbp	resp_rate	temperature	...	hemoglobin	mch	mchc	mcv	platelet	rbc	rdw	WBC1	gcs	crp
0	30002654	75.0	15978672	NaN	75.0	116.0	65.0	77.0	15.0	37.11	...	10.2	33.2	32.6	102.0	28.0	3.07	16.2	18.2	15.0	NaN
1	30003598	136.0	15332791	NaN	70.0	96.0	44.0	63.0	15.0	36.06	...	10.3	28.4	32.4	88.0	449.0	3.62	17.6	17.6	15.0	NaN
2	30004144	80.0	10369174	NaN	60.0	132.0	53.0	78.0	27.0	36.39	...	13.0	32.2	35.8	90.0	191.0	4.02	13.0	5.1	13.0	NaN
3	30004391	58.8	18730522	178.0	118.0	108.0	79.0	85.0	17.0	37.17	...	8.5	36.5	33.5	109.0	130.0	2.33	12.5	8.1	15.0	NaN
4	30005085	101.4	14289094	183.0	84.0	123.0	59.0	81.0	14.0	36.0	...	13.0	30.8	33.0	94.0	202.0	4.21	13.6	6.5	15.0	NaN

5 rows × 96 columns

In [ ]:
```
df13['height_m'] = df13['height'] / 100

# 计算BMI: 体重（公斤） / (身高（米）^2)
df13['BMI'] = df13['weight'] / (df13['height_m'] ** 2)
df13['BMI'] = df13['BMI'].round(2)
# 如果你不需要保留转换后的身高列，可以将其删除
df13.drop('height_m', axis=1, inplace=True)
```

In [ ]:
```
df13.head()
```

Out[ ]:

	stay_id	weight	subject_id	height	heart_rate	sbp	dbp	mbp	resp_rate	temperature	...	mch	mchc	mcv	platelet	rbc	rdw	WBC1	gcs	crp	BMI
0	30002654	75.0	15978672	NaN	75.0	116.0	65.0	77.0	15.0	37.11	...	33.2	32.6	102.0	28.0	3.07	16.2	18.2	15.0	NaN	NaN
1	30003598	136.0	15332791	NaN	70.0	96.0	44.0	63.0	15.0	36.06	...	28.4	32.4	88.0	449.0	3.62	17.6	17.6	15.0	NaN	NaN
2	30004144	80.0	10369174	NaN	60.0	132.0	53.0	78.0	27.0	36.39	...	32.2	35.8	90.0	191.0	4.02	13.0	5.1	13.0	NaN	NaN
3	30004391	58.8	18730522	178.0	118.0	108.0	79.0	85.0	17.0	37.17	...	36.5	33.5	109.0	130.0	2.33	12.5	8.1	15.0	NaN	18.56
4	30005085	101.4	14289094	183.0	84.0	123.0	59.0	81.0	14.0	36.0	...	30.8	33.0	94.0	202.0	4.21	13.6	6.5	15.0	NaN	30.28

5 rows × 97 columns

In [ ]:
```
urine_output= pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\urine_output_rate.parquet')
urine_output.head()
```

Out[ ]:

	stay_id	charttime	weight	uo	urineoutput_6hr	urineoutput_12hr	urineoutput_24hr	uo_mlkghr_6hr	uo_mlkghr_12hr	uo_mlkghr_24hr	uo_tm_6hr	uo_tm_12hr	uo_tm_24hr
0	30000153	29/9/2174 12:12:00	70.0	280.0	280.0	280.0	280.0	NaN	NaN	NaN	0.1	0.1	0.1
1	30000153	29/9/2174 14:00:00	70.0	45.0	325.0	325.0	325.0	NaN	NaN	NaN	1.9	1.9	1.9
2	30000153	29/9/2174 15:00:00	70.0	50.0	375.0	375.0	375.0	NaN	NaN	NaN	2.9	2.9	2.9
3	30000153	29/9/2174 16:00:00	70.0	50.0	425.0	425.0	425.0	NaN	NaN	NaN	3.9	3.9	3.9
4	30000153	29/9/2174 17:00:00	73.0	45.0	470.0	470.0	470.0	NaN	NaN	NaN	4.9	4.9	4.9

In [ ]:
```
urine_output1=urine_output[['stay_id','charttime', 'urineoutput_24hr']]
```

In [ ]:
```
urine_output1['charttime'] = pd.to_datetime(urine_output1['charttime'], dayfirst=True)
urine_output1['charttime'] = urine_output1['charttime'].astype('datetime64[ns]')
print(np.intersect1d(df13.columns,urine_output1.columns))
```

In [ ]:
```
merged_df = pd.merge(urine_output1, df13, on=['stay_id'],how='left')
condition = (merged_df['window_start-1'] < merged_df['charttime']) & \
            (merged_df['charttime'] < merged_df['aki_charttime']+ pd.Timedelta(days=1))
filtered_df = merged_df[condition]

filtered_df = filtered_df.groupby('stay_id').agg({'urineoutput_24hr': 'max', 'charttime': 'first'}).reset_index()
urine_output12 = filtered_df.merge(urine_output1, on=['stay_id', 'charttime'], how='left')
urine_output12 = urine_output12.drop_duplicates(subset=['stay_id', 'charttime'])
```

In [ ]:
```
urine_output12=urine_output12.drop(['charttime'], axis=1)
df14= df13.merge(urine_output12, on=['stay_id'], how='left')
df14.shape
```

Out[ ]: (10269, 99)

In [ ]:
```
df14.head()
```

Out[ ]:

	stay_id	weight	subject_id	height	heart_rate	sbp	dbp	mbp	resp_rate	temperature	...	mcv	platelet	rbc	rdw	WBC1	gcs	crp	BMI	urineoutput_24hr_x	urineoutput_24hr_y
0	30002654	75.0	15978672	NaN	75.0	116.0	65.0	77.0	15.0	37.11	...	102.0	28.0	3.07	16.2	18.2	15.0	NaN	NaN	1370.0	60.0
1	30003598	136.0	15332791	NaN	70.0	96.0	44.0	63.0	15.0	36.06	...	88.0	449.0	3.62	17.6	17.6	15.0	NaN	NaN	1370.0	30.0
2	30004144	80.0	10369174	NaN	60.0	132.0	53.0	78.0	27.0	36.39	...	90.0	191.0	4.02	13.0	5.1	13.0	NaN	NaN	1652.0	240.0
3	30004391	58.8	18730522	178.0	118.0	108.0	79.0	85.0	17.0	37.17	...	109.0	130.0	2.33	12.5	8.1	15.0	NaN	18.56	2732.0	1065.0
4	30005085	101.4	14289094	183.0	84.0	123.0	59.0	81.0	14.0	36.0	...	94.0	202.0	4.21	13.6	6.5	15.0	NaN	30.28	2002.0	60.0

5 rows × 99 columns

In [ ]:
```
df14.drop('urineoutput_24hr_y', axis=1, inplace=True)
```

In [ ]:
```
df14.rename(columns={'urineoutput_24hr_x': 'urineoutput_24hr'}, inplace=True)
```

In [ ]:
```
df14.shape
```

Out[ ]: (10269, 98)

In [ ]:
```
aspiii = pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\apsiii.parquet')
aspiii.head()
```

Out[ ]:

	subject_id	hadm_id	stay_id	apsiii	apsiii_prob	hr_score	mbp_score	temp_score	resp_rate_score	pao2_aado2_score	...	wbc_score	creatinine_score	uo_score	bun_score	sodium_score	albu
0	14007982	25442966	36796773	29	0.044555	5.0	7.0	0.0	6.0	NaN	...	0.0	0.0	8.0	0.0	0.0	
1	13630588	26564297	32994851	23	0.033927	0.0	7.0	0.0	8.0	NaN	...	0.0	0.0	5.0	0.0	0.0	
2	14155916	20422432	34317467	41	0.075975	1.0	15.0	0.0	6.0	NaN	...	0.0	0.0	4.0	7.0	2.0	
3	13264941	27604262	37567861	28	0.042586	1.0	7.0	0.0	8.0	NaN	...	5.0	0.0	NaN	0.0	0.0	
4	10649299	23334742	35217255	45	0.090356	7.0	7.0	0.0	8.0	NaN	...	0.0	0.0	0.0	7.0	0.0	

5 rows × 21 columns

In [ ]:
```
print(np.intersect1d(df14.columns,aspiii.columns))
```

```
['hadm_id' 'stay_id' 'subject_id']
```

In [ ]:
```
aspiii = aspiii.drop('hadm_id', axis=1)
```

In [ ]:
```
aspiii1=aspiii[['subject_id', 'stay_id','apsiii']]
```

In [ ]:
```
df14['stay_id'] = df14['stay_id'].astype(int)
```

In [ ]:
```
df15= pd.merge(aspiii1, df14, on=['subject_id','stay_id'],how='inner')
df15.shape
```

Out[ ]: (10269, 99)

In [ ]:
```
sirs = pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\sirs.parquet')
sirs.head()
```

Out[ ]:

	subject_id	hadm_id	stay_id	sirs	temp_score	heart_rate_score	resp_score	wbc_score
0	10000032	29079034	39553978	2	0.0	1.0	1.0	0.0
1	10000690	25860671	37081114	3	1.0	1.0	1.0	0.0
2	10000980	26913865	39765666	1	0.0	0.0	1.0	0.0
3	10001217	27703517	34592300	2	0.0	1.0	1.0	0.0
4	10001217	24597018	37067082	4	1.0	1.0	1.0	1.0

In [ ]:
```
sirs=sirs[['subject_id', 'stay_id','sirs']]
```

In [ ]:
```
df16= pd.merge(sirs, df15, on=['subject_id','stay_id'],how='inner')
df16.shape
```

Out[ ]: (10269, 100)

In [ ]:
```
df16.head()
```

Out[ ]:

	subject_id	stay_id	sirs	apsiii	weight	height	heart_rate	sbp	dbp	mbp	...	mchc	mcv	platelet	rbc	rdw	WBC1	gcs	crp	BMI	urineoutput_24hr
0	10000690	37081114	3	52	55.3	NaN	79.0	107.0	63.0	71.0	...	33.2	93.0	199.0	3.07	16.4	7.5	15.0	NaN	NaN	815.0
1	10001884	37510196	3	51	65.0	157.0	76.0	127.0	73.0	87.0	...	31.0	91.0	149.0	3.68	17.6	12.0	15.0	NaN	26.37	1655.0
2	10002155	31090461	3	52	48.0	NaN	94.0	118.0	51.0	68.0	...	31.1	74.0	263.0	2.61	16.2	9.5	15.0	NaN	NaN	3500.0
3	10002443	35044219	4	41	156.1	178.0	92.0	121.0	79.0	92.0	...	32.7	93.0	219.0	4.51	13.1	15.9	15.0	NaN	49.27	3450.0
4	10002495	36753294	3	54	64.1	170.0	114.0	160.0	78.0	94.0	...	34.5	92.0	176.0	4.4	12.5	36.8	15.0	NaN	22.18	3400.0

5 rows × 100 columns

In [ ]:
```
acei = pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\acei.parquet')
acei.head()
```

Out[ ]:

	subject_id	hadm_id	acei	starttime	stoptime
0	10000690	23280645	Lisinopril	17/9/2150 10:00:00	19/9/2150 17:00:00
1	10000690	23280645	Lisinopril	24/9/2150 10:00:00	24/9/2150 18:00:00
2	10000690	23280645	Lisinopril	22/9/2150 10:00:00	24/9/2150 10:00:00
3	10000690	23280645	Lisinopril	17/9/2150 10:00:00	17/9/2150 11:00:00
4	10000690	23280645	Lisinopril	24/9/2150 11:00:00	24/9/2150 18:00:00

In [ ]:
```
acei['starttime'] = pd.to_datetime(acei['starttime'], dayfirst=True)
acei['starttime'] = acei['starttime'].astype('datetime64[ns]')
```

In [ ]:
```
acei = acei.drop(['stoptime','acei'], axis=1)
```

In [ ]:
```
acei.head()
```

Out[ ]:

	subject_id	hadm_id	starttime
0	10000690	23280645	2150-09-17 10:00:00
1	10000690	23280645	2150-09-24 10:00:00
2	10000690	23280645	2150-09-22 10:00:00
3	10000690	23280645	2150-09-17 10:00:00
4	10000690	23280645	2150-09-24 11:00:00

In [ ]:
```
acei= acei.drop_duplicates()
```

In [ ]:
```
acei.shape
```

Out[ ]: (125747, 3)

In [ ]:
```
aceia = acei.groupby('hadm_id')['starttime'].min().reset_index()
```

In [ ]:
```
aceia.head()
```

Out[ ]:

	hadm_id	starttime
0	20000041	2143-09-03 08:00:00
1	20000057	2190-01-15 22:00:00
2	20000180	2165-11-07 08:00:00
3	20000239	2170-04-13 10:00:00
4	20000298	2183-06-20 10:00:00

In [ ]:
```
aceia.shape
```

Out[ ]: (84369, 2)

In [ ]:
```
merged_df = df11.merge(aceia, on=['hadm_id'], how='left')

# 检查时间条件
merged_df['starttime'] = pd.to_datetime(merged_df['starttime'])
merged_df['window_start7'] = pd.to_datetime(merged_df['window_start7'])
merged_df['acei'] = (merged_df['starttime'] < merged_df['window_start7']).astype(int)

# 保留 df11 原本的数据
df17 = df16.merge(merged_df[['hadm_id', 'acei']], on=['hadm_id'], how='left')

df17.head()
```

Out[ ]:

	subject_id	stay_id	sirs	apsiii	weight	height	heart_rate	sbp	dbp	mbp	...	mcv	platelet	rbc	rdw	WBC1	gcs	crp	BMI	urineoutput_24hr	acei
0	10000690	37081114	3	52	55.3	NaN	79.0	107.0	63.0	71.0	...	93.0	199.0	3.07	16.4	7.5	15.0	NaN	NaN	815.0	1
1	10001884	37510196	3	51	65.0	157.0	76.0	127.0	73.0	87.0	...	91.0	149.0	3.68	17.6	12.0	15.0	NaN	26.37	1655.0	0
2	10002155	31090461	3	52	48.0	NaN	94.0	118.0	51.0	68.0	...	74.0	263.0	2.61	16.2	9.5	15.0	NaN	NaN	3500.0	0
3	10002443	35044219	4	41	156.1	178.0	92.0	121.0	79.0	92.0	...	93.0	219.0	4.51	13.1	15.9	15.0	NaN	49.27	3450.0	0
4	10002495	36753294	3	54	64.1	170.0	114.0	160.0	78.0	94.0	...	92.0	176.0	4.4	12.5	36.8	15.0	NaN	22.18	3400.0	1

5 rows × 101 columns

In [ ]:
```
df17.shape
```

Out[ ]: (10269, 101)

In [ ]:
```
df17.subject_id.nunique()
```

Out[ ]: 10269

In [ ]:
```
df17.stay_id.nunique()
```

Out[ ]: 10269

In [ ]:
```
df17.hadm_id.nunique()
```

Out[ ]: 10269

In [ ]:
```
sum(df17['acei'] == 1)
```

Out[ ]: 2000

In [ ]:
```
arb = pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\arb.parquet')
arb.head()
```

Out[ ]:

	subject_id	hadm_id	arb	starttime	stoptime
0	10001401	20144849	Losartan Potassium	20/11/2136 15:00:00	23/11/2136 18:00:00
1	10001401	29250371	Losartan Potassium	5/1/2137 08:00:00	5/1/2137 20:00:00
2	10001667	22672901	Losartan Potassium	22/8/2173 08:00:00	24/8/2173 22:00:00
3	10001843	21728396	Losartan Potassium	9/11/2131 21:00:00	11/11/2131 15:00:00
4	10002013	21516558	Losartan Potassium	4/12/2169 08:00:00	5/12/2169 21:00:00

In [ ]:
```
arb['starttime'] = pd.to_datetime(arb['starttime'], dayfirst=True)
arb['starttime'] = arb['starttime'].astype('datetime64[ns]')
```

In [ ]:
```
arb = arb.drop(['stoptime','arb'], axis=1)
```

In [ ]:
```
arb.head()
```

Out[ ]:

	subject_id	hadm_id	starttime
0	10001401	20144849	2136-11-20 15:00:00
1	10001401	29250371	2137-01-05 08:00:00
2	10001667	22672901	2173-08-22 08:00:00
3	10001843	21728396	2131-11-09 21:00:00
4	10002013	21516558	2169-12-04 08:00:00

In [ ]:
```
arb= arb.drop_duplicates()
```

In [ ]:
```
arb.shape
```

Out[ ]: (50516, 3)

In [ ]:
```
arba = arb.groupby('hadm_id')['starttime'].min().reset_index()
```

In [ ]:
```
arba.head()
```

Out[ ]:

	hadm_id	starttime
0	20000240	2148-12-17 20:00:00
1	20000343	2137-01-28 08:00:00
2	20000351	2145-06-15 17:00:00
3	20001135	2151-10-31 07:00:00
4	20001363	2177-09-08 18:00:00

In [ ]:
```
arba.shape
```

Out[ ]: (36594, 2)

In [ ]:
```
merged_df1 = df17.merge(arba, on=['hadm_id'], how='left')

# 检查时间条件
merged_df1['starttime'] = pd.to_datetime(merged_df1['starttime'])
merged_df1['window_start7'] = pd.to_datetime(merged_df1['window_start7'])
merged_df1['arb'] = (merged_df1['starttime'] < merged_df1['window_start7']).astype(int)

# 保留 df11 原本的数据
df18= df17.merge(merged_df1[['hadm_id', 'arb']], on=['hadm_id'], how='left')

df18.head()
```

Out[ ]:

	subject_id	stay_id	sirs	apsiii	weight	height	heart_rate	sbp	dbp	mbp	...	platelet	rbc	rdw	WBC1	gcs	crp	BMI	urineoutput_24hr	acei	arb
0	10000690	37081114	3	52	55.3	NaN	79.0	107.0	63.0	71.0	...	199.0	3.07	16.4	7.5	15.0	NaN	NaN	815.0	1	0
1	10001884	37510196	3	51	65.0	157.0	76.0	127.0	73.0	87.0	...	149.0	3.68	17.6	12.0	15.0	NaN	26.37	1655.0	0	0
2	10002155	31090461	3	52	48.0	NaN	94.0	118.0	51.0	68.0	...	263.0	2.61	16.2	9.5	15.0	NaN	NaN	3500.0	0	0
3	10002443	35044219	4	41	156.1	178.0	92.0	121.0	79.0	92.0	...	219.0	4.51	13.1	15.9	15.0	NaN	49.27	3450.0	0	0
4	10002495	36753294	3	54	64.1	170.0	114.0	160.0	78.0	94.0	...	176.0	4.4	12.5	36.8	15.0	NaN	22.18	3400.0	1	0

5 rows × 102 columns

In [ ]:
```
df18.shape
```

Out[ ]: (10269, 102)

In [ ]:
```
df18.hadm_id.nunique()
```

Out[ ]: 10269

In [ ]:
```
df18.stay_id.nunique()
```

Out[ ]: 10269

In [ ]:
```
sum(df18['arb'] == 1)
```

Out[ ]: 590

In [ ]:
```
df18.shape
```

Out[ ]: (10269, 102)

In [ ]:
```
df18['ACEI/ARB'] = (df18['acei'] == 1) | (df18['arb'] == 1)
df18['ACEI/ARB'] = df18['ACEI/ARB'].astype(int)
# 删除原本的'ACEI'和'arb'列
df18.drop(['acei', 'arb'], axis=1, inplace=True)
```

In [ ]:
```
sum(df18['ACEI/ARB'] == 1)
```

Out[ ]: 2533

In [ ]:
```
df18.stay_id.nunique()
```

Out[ ]: 10269

In [ ]:
```
df18.subject_id.nunique()
```

Out[ ]: 10269

In [ ]:
```
df18.hadm_id.nunique()
```

Out[ ]: 10269

In [ ]:
```
creatinine_baselines = pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\creatinine_baseline.parquet')
creatinine_baselines.head()
```

Out[ ]:

	hadm_id	gender	age	scr_min	ckd	mdrd_est	scr_baseline
0	24016413	F	46.803035	NaN	0	0.565994	0.565994
1	22725460	F	77.917014	0.6	0	0.627694	0.600000
2	20940957	F	75.483055	0.6	0	0.623663	0.600000
3	21476780	F	75.156656	0.9	0	0.623114	0.900000
4	21743184	F	65.360090	0.8	0	0.605696	0.800000

In [ ]:
```
print(np.intersect1d(creatinine_baselines.columns,df18.columns))
```

```
['gender' 'hadm_id']
```

In [ ]:
```
creatinine_baselines = creatinine_baselines.drop(['gender','scr_baseline','age','mdrd_est'], axis=1)
```

In [ ]:
```
creatinine_baselines['hadm_id'] = creatinine_baselines['hadm_id'].astype('int64')
df19 = pd.merge(df18, creatinine_baselines, on=['hadm_id'])
```

In [ ]:
```
df19.shape
```

Out[ ]: (10269, 103)

In [ ]:
```
oasis = pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\oasis.parquet')
oasis.head()
```

Out[ ]:

	subject_id	hadm_id	stay_id	oasis	oasis_prob	age	age_score	preiculos	preiculos_score	gcs	...	resprate	resp_rate_score	temp	temp_score	urineoutput	urineoutput_score	mec
0	10000032	29079034	39553978	31	0.097783	52.559969	3	85.000000	3	14.0	...	24.0	1.0	37.50	2.0	175.0	10.0	
1	10000690	25860671	37081114	41	0.279468	86.837120	9	95.000000	3	12.0	...	35.0	6.0	35.56	4.0	695.0	5.0	
2	10000980	26913865	39765666	18	0.020241	76.486231	6	64.000000	3	15.0	...	25.5	1.0	37.06	2.0	3900.0	0.0	
3	10001217	27703517	34592300	17	0.017861	55.962942	6	1364.400000	0	15.0	...	11.0	1.0	37.00	2.0	2475.0	1.0	
4	10001217	24597018	37067082	18	0.020241	55.881486	6	2662.033333	2	15.0	...	27.0	1.0	38.22	2.0	2645.0	0.0	

5 rows × 25 columns

In [ ]:
```
print(list(oasis.columns))
```

```
['subject_id', 'hadm_id', 'stay_id', 'oasis', 'oasis_prob', 'age', 'age_score', 'preiculos', 'preiculos_score', 'gcs', 'gcs_score', 'heartrate', 'heart_rate_score', 'meanbp', 'mbp_score', 'res
prate', 'resp_rate_score', 'temp', 'temp_score', 'urineoutput', 'urineoutput_score', 'mechvent', 'mechvent_score', 'electivesurgery', 'electivesurgery_score']
```

In [ ]:
```
oasis=oasis[['subject_id','stay_id','oasis']]
```

In [ ]:
```
print(np.intersect1d(oasis.columns,df15.columns))
```

```
['stay_id' 'subject_id']
```

In [ ]:
```
df20= pd.merge(oasis, df19, on=['subject_id','stay_id'],how='right')
df20.shape
```

Out[ ]: (10269, 104)

In [ ]:
```
df20.stay_id.nunique()
```

Out[ ]: 10269

In [ ]:
```
df20.subject_id.nunique()
```

Out[ ]: 10269

In [ ]:
```
df20.head()
```

Out[ ]:

	subject_id	stay_id	oasis	sirs	apsiii	weight	height	heart_rate	sbp	dbp	...	rbc	rdw	WBC1	gcs	crp	BMI	urineoutput_24hr	ACEI/ARB	scr_min	ckd
0	10000690	37081114	41	3	52	55.3	NaN	79.0	107.0	63.0	...	3.07	16.4	7.5	15.0	NaN	NaN	815.0	1	0.6	0
1	10001884	37510196	35	3	51	65.0	157.0	76.0	127.0	73.0	...	3.68	17.6	12.0	15.0	NaN	26.37	1655.0	0	0.5	0
2	10002155	31090461	26	3	52	48.0	NaN	94.0	118.0	51.0	...	2.61	16.2	9.5	15.0	NaN	NaN	3500.0	0	2.1	1
3	10002443	35044219	24	4	41	156.1	178.0	92.0	121.0	79.0	...	4.51	13.1	15.9	15.0	NaN	49.27	3450.0	0	0.6	0
4	10002495	36753294	32	3	54	64.1	170.0	114.0	160.0	78.0	...	4.4	12.5	36.8	15.0	NaN	22.18	3400.0	1	1.1	0

5 rows × 104 columns

In [ ]:
```
crrt = pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\crrt.parquet')
crrt.head()
```

Out[ ]:

	stay_id	charttime	crrt_mode	access_pressure	blood_flow	citrate	current_goal	dialysate_fluid	dialysate_rate	effluent_pressure	...	prefilter_replacement_rate	postfilter_replacement_rate	rep
0	30003226	27/2/2123 08:26:00	CVVHDF	-44.0	120.0	180.0	0.0	Prismasate K2	1000.0	0.0	...	1800.0	NaN	
1	30003226	27/2/2123 08:27:00	CVVHDF	NaN	NaN	1810.0	0.0	Prismasate K2	1000.0	NaN	...	1800.0	NaN	
2	30003226	27/2/2123 09:00:00	None	NaN	NaN	NaN	NaN	None	NaN	NaN	...	NaN	NaN	
3	30003226	27/2/2123 10:00:00	None	-45.0	120.0	180.0	NaN	Prismasate K2	1000.0	-25.0	...	1800.0	NaN	
4	30003226	27/2/2123 11:00:00	None	NaN	NaN	NaN	NaN	None	NaN	NaN	...	NaN	NaN	

5 rows × 24 columns

In [ ]:
```
crrt.shape
```

Out[ ]: (475214, 24)

In [ ]:
```
crrt['charttime'] = pd.to_datetime(crrt['charttime'], dayfirst=True)
crrt['charttime'] = crrt['charttime'].astype('datetime64[ns]')
```

In [ ]:
```
crrt=crrt[['stay_id','charttime']]
```

In [ ]:
```
crrt= crrt.drop_duplicates()
```

In [ ]:
```
crrt.head()
```

Out[ ]:

	stay_id	charttime
0	30003226	2123-02-27 08:26:00
1	30003226	2123-02-27 08:27:00
2	30003226	2123-02-27 09:00:00
3	30003226	2123-02-27 10:00:00
4	30003226	2123-02-27 11:00:00

In [ ]:
```
crrt.shape
```

Out[ ]:
```
(475214, 2)
```

In [ ]:
```
crrt['stay_id'] = crrt['stay_id'].astype(int)
```

In [ ]:
```
df20a=df20[['stay_id','aki_charttime','window_start7']]
```

In [ ]:
```
crrta = df20a.merge(crrt, on='stay_id', how='inner')
```

In [ ]:
```
crrta.head()
```

Out[ ]:

	stay_id	aki_charttime	window_start7	charttime
0	34100191	2196-02-25 00:00:00	2196-03-03 00:00:00	2196-02-26 01:00:00
1	34100191	2196-02-25 00:00:00	2196-03-03 00:00:00	2196-02-26 02:00:00
2	34100191	2196-02-25 00:00:00	2196-03-03 00:00:00	2196-02-26 02:03:00
3	34100191	2196-02-25 00:00:00	2196-03-03 00:00:00	2196-02-26 03:00:00
4	34100191	2196-02-25 00:00:00	2196-03-03 00:00:00	2196-02-26 04:00:00

In [ ]:
```
def assign_crrt_value(row):
    if row['aki_charttime'] <= row['charttime'] < row['window_start7']:
        return 1
    elif row['charttime'] < row['aki_charttime']:
        return -1
    else:
        return 0

crrta['CRRT'] = crrta.apply(assign_crrt_value, axis=1)
```

In [ ]:
```
crrta.head()
```

Out[ ]:

	stay_id	aki_charttime	window_start7	charttime	CRRT
0	34100191	2196-02-25 00:00:00	2196-03-03 00:00:00	2196-02-26 01:00:00	1
1	34100191	2196-02-25 00:00:00	2196-03-03 00:00:00	2196-02-26 02:00:00	1
2	34100191	2196-02-25 00:00:00	2196-03-03 00:00:00	2196-02-26 02:03:00	1
3	34100191	2196-02-25 00:00:00	2196-03-03 00:00:00	2196-02-26 03:00:00	1
4	34100191	2196-02-25 00:00:00	2196-03-03 00:00:00	2196-02-26 04:00:00	1

In [ ]:
```
crrt1= crrta.query('CRRT == 1')
```

In [ ]:
```
crrt2=crrt1[['stay_id','CRRT']]
```

In [ ]:
```
crrt3= crrt2.drop_duplicates()
```

In [ ]:
```
crrt3.shape
```

Out[ ]:
```
(530, 2)
```

In [ ]:
```
df21 = df20.merge(crrt3, on=['stay_id'], how='left')
```

In [ ]:
```
df21['CRRT'].fillna(0, inplace=True)
```

In [ ]:
```
df21.shape
```

Out[ ]:
```
(10269, 105)
```

In [ ]:
```
df21['CRRT'].value_counts()
```

Out[ ]:
```
CRRT
0.0    9739
1.0     530
Name: count, dtype: int64
```

In [ ]:
```
ventilator= pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\ventilation.parquet')
ventilator.head()
```

Out[ ]:

	stay_id	starttime	endtime	ventilation_status
0	32324375	28/1/2135 00:13:00	28/1/2135 01:00:00	SupplementalOxygen
1	31109006	24/8/2187 00:00:00	24/8/2187 08:00:00	SupplementalOxygen
2	30982821	12/7/2138 19:00:00	13/7/2138 12:00:00	InvasiveVent
3	31612054	12/9/2116 07:00:00	13/9/2116 05:00:00	SupplementalOxygen
4	31045843	22/1/2130 08:24:00	5/2/2130 10:00:00	InvasiveVent

In [ ]:
```
ventilator['starttime'] = pd.to_datetime(ventilator['starttime'], dayfirst=True)
ventilator['starttime'] = ventilator['starttime'].astype('datetime64[ns]')
ventilator['endtime'] = pd.to_datetime(ventilator['endtime'], dayfirst=True)
ventilator['endtime'] = ventilator['endtime'].astype('datetime64[ns]')
```

In [ ]:
```
ventilator = ventilator.drop_duplicates(subset=['stay_id', 'starttime'])
```

In [ ]:
```
ventilator.head()
```

Out[ ]:

	stay_id	starttime	endtime	ventilation_status
0	32324375	2135-01-28 00:13:00	2135-01-28 01:00:00	SupplementalOxygen
1	31109006	2187-08-24 00:00:00	2187-08-24 08:00:00	SupplementalOxygen
2	30982821	2138-07-12 19:00:00	2138-07-13 12:00:00	InvasiveVent
3	31612054	2116-09-12 07:00:00	2116-09-13 05:00:00	SupplementalOxygen
4	31045843	2130-01-22 08:24:00	2130-02-05 10:00:00	InvasiveVent

In [ ]:
```
unique_ventilation_statuses = ventilator['ventilation_status'].unique()
unique_ventilation_statuses.tolist()
```

Out[ ]:
```
['SupplementalOxygen',
 'InvasiveVent',
 'NonInvasiveVent',
 'HFNC',
 'Tracheostomy',
 'None']
```

In [ ]:
```
ventilator= ventilator[ventilator['ventilation_status'] != 'SupplementalOxygen']
ventilator= ventilator[ventilator['ventilation_status'] != 'HFNC']
ventilator= ventilator[ventilator['ventilation_status'] != 'Tracheostomy']
ventilator= ventilator[ventilator['ventilation_status'] != 'None']
```

In [ ]:
```
ventilator.head()
```

Out[ ]:

	stay_id	starttime	endtime	ventilation_status
2	30982821	2138-07-12 19:00:00	2138-07-13 12:00:00	InvasiveVent
4	31045843	2130-01-22 08:24:00	2130-02-05 10:00:00	InvasiveVent
6	30965732	2159-01-22 23:00:00	2159-01-23 13:00:00	NonInvasiveVent
8	30599145	2155-01-14 11:00:00	2155-01-23 08:00:00	InvasiveVent
14	30299191	2154-04-21 22:20:00	2154-04-23 12:11:00	InvasiveVent

In [ ]:
```
ventilator.shape
```

Out[ ]:
```
(50837, 4)
```

In [ ]:
```
ventilator['stay_id'] = ventilator['stay_id'].astype(int)
```

In [ ]:
```
df21a=df21[['stay_id','aki_charttime','window_start7']]
```

In [ ]:
```
ventilator1 = df21a.merge(ventilator, on='stay_id', how='inner')
```

In [ ]:
```
ventilator1['Mechanical ventilation'] = (
    (ventilator1['aki_charttime'] < ventilator1['starttime']) & (ventilator1['starttime'] < ventilator1['window_start7'])
) | (
    (ventilator1['aki_charttime'] < ventilator1['endtime']) & (ventilator1['endtime'] < ventilator1['window_start7'])
) | (
    (ventilator1['starttime'] < ventilator1['aki_charttime']) & (ventilator1['window_start7'] < ventilator1['endtime'])
)
```

In [ ]:
```
ventilator2 = ventilator1[ventilator1['Mechanical ventilation'] == True]
```

In [ ]:
```
ventilator2.head()
```

Out[ ]:

	stay_id	aki_charttime	window_start7	starttime	endtime	ventilation_status	Mechanical ventilation
0	37510196	2131-01-12 19:00:00	2131-01-19 19:00:00	2131-01-12 21:00:00	2131-01-13 04:00:00	NonInvasiveVent	True
1	37510196	2131-01-12 19:00:00	2131-01-19 19:00:00	2131-01-16 10:00:00	2131-01-19 18:40:00	InvasiveVent	True
2	37510196	2131-01-12 19:00:00	2131-01-19 19:00:00	2131-01-13 04:00:00	2131-01-14 07:00:00	InvasiveVent	True
4	36753294	2141-05-23 17:00:00	2141-05-30 17:00:00	2141-05-23 20:22:00	2141-05-24 05:00:00	NonInvasiveVent	True
5	32128372	2137-02-26 06:00:00	2137-03-05 06:00:00	2137-02-25 23:37:00	2137-02-28 11:00:00	InvasiveVent	True

In [ ]:
```
ventilator3= ventilator2.groupby('stay_id')['starttime'].idxmin()
ventilator4= ventilator2.loc[ventilator3]
```

In [ ]:
```
ventilator4.shape
```

Out[ ]:
```
(5845, 7)
```

In [ ]:
```
ventilator4.stay_id.nunique()
```

Out[ ]:
```
5845
```

In [ ]:
```
df22 = df21.merge(ventilator4[['stay_id','Mechanical ventilation']], on=['stay_id'], how='left')
df22.head()
```

Out[ ]:

	subject_id	stay_id	oasis	sirs	apsiii	weight	height	heart_rate	sbp	dbp	...	WBC1	gcs	crp	BMI	urineoutput_24hr	ACEI/ARB	scr_min	ckd	CRRT	Mechanical ventilation
0	10000690	37081114	41	3	52	55.3	NaN	79.0	107.0	63.0	...	7.5	15.0	NaN	NaN	815.0	1	0.6	0	0.0	NaN
1	10001884	37510196	35	3	51	65.0	157.0	76.0	127.0	73.0	...	12.0	15.0	NaN	26.37	1655.0	0	0.5	0	0.0	True
2	10002155	31090461	26	3	52	48.0	NaN	94.0	118.0	51.0	...	9.5	15.0	NaN	NaN	3500.0	0	2.1	1	0.0	NaN
3	10002443	35044219	24	4	41	156.1	178.0	92.0	121.0	79.0	...	15.9	15.0	NaN	49.27	3450.0	0	0.6	0	0.0	NaN
4	10002495	36753294	32	3	54	64.1	170.0	114.0	160.0	78.0	...	36.8	15.0	NaN	22.18	3400.0	1	1.1	0	0.0	True

5 rows × 106 columns

In [ ]:
```
df22['Mechanical ventilation'] = df22['Mechanical ventilation'].fillna(False)
```

In [ ]:
```
df22.shape
```

Out[ ]:
```
(10269, 106)
```

In [ ]:
```
sum(df22['Mechanical ventilation'] == 1)
```

Out[ ]:
```
5845
```

In [ ]:
```
vasoactive_agent= pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\vasoactive_agent.parquet')
vasoactive_agent.head()
```

Out[ ]:

	stay_id	starttime	endtime	dopamine	epinephrine	norepinephrine	phenylephrine	vasopressin	dobutamine	milrinone
0	30000484	15/1/2136 06:39:00	15/1/2136 07:00:00	NaN	NaN	NaN	NaN	NaN	1.000174	NaN
1	30000484	15/1/2136 07:00:00	15/1/2136 09:15:00	NaN	NaN	NaN	NaN	NaN	2.000348	NaN
2	30000484	15/1/2136 09:15:00	15/1/2136 09:40:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	30000484	15/1/2136 09:40:00	15/1/2136 17:42:00	5.003784	NaN	NaN	NaN	NaN	NaN	NaN
4	30000484	15/1/2136 17:42:00	15/1/2136 21:02:00	2.501892	NaN	NaN	NaN	NaN	NaN	NaN

In [ ]:
```
vasoactive_agent['starttime'] = pd.to_datetime(vasoactive_agent['starttime'], dayfirst=True)
vasoactive_agent['starttime'] = vasoactive_agent['starttime'].astype('datetime64[ns]')
vasoactive_agent['endtime'] = pd.to_datetime(vasoactive_agent['endtime'], dayfirst=True)
vasoactive_agent['endtime'] = vasoactive_agent['endtime'].astype('datetime64[ns]')
```

In [ ]:
```
vasoactive_agent = vasoactive_agent.drop_duplicates(subset=['stay_id', 'starttime'])
```

In [ ]:
```
vasoactive_agent['stay_id'] = vasoactive_agent['stay_id'].astype(int)
```

In [ ]:
```
vasoactive_agent1 = df21a.merge(vasoactive_agent, on='stay_id', how='inner')
```

In [ ]:
```
vasoactive_agent1.head()
```

Out[ ]:

	stay_id	aki_charttime	window_start7	starttime	endtime	dopamine	epinephrine	norepinephrine	phenylephrine	vasopressin	dobutamine	milrinone
0	37081114	2150-11-03 09:00:00	2150-11-10 09:00:00	2150-11-02 20:00:00	2150-11-02 22:45:00	NaN	NaN	NaN	0.600106	NaN	NaN	NaN
1	37081114	2150-11-03 09:00:00	2150-11-10 09:00:00	2150-11-02 22:45:00	2150-11-02 23:36:00	NaN	NaN	NaN	0.499987	NaN	NaN	NaN
2	37081114	2150-11-03 09:00:00	2150-11-10 09:00:00	2150-11-02 23:36:00	2150-11-03 00:35:00	NaN	NaN	NaN	0.400031	NaN	NaN	NaN
3	37081114	2150-11-03 09:00:00	2150-11-10 09:00:00	2150-11-03 00:35:00	2150-11-03 01:07:00	NaN	NaN	NaN	0.299991	NaN	NaN	NaN
4	37081114	2150-11-03 09:00:00	2150-11-10 09:00:00	2150-11-03 01:07:00	2150-11-03 02:03:00	NaN	NaN	NaN	0.200016	NaN	NaN	NaN

In [ ]:
```
vasoactive_agent1['vasoactive_agent'] = (
    (vasoactive_agent1['aki_charttime'] < vasoactive_agent1['starttime']) & (vasoactive_agent1['starttime'] < vasoactive_agent1['window_start7'])
) | (
    (vasoactive_agent1['aki_charttime'] < vasoactive_agent1['endtime']) & (vasoactive_agent1['endtime'] < vasoactive_agent1['window_start7'])
) | (
    (vasoactive_agent1['starttime'] < vasoactive_agent1['aki_charttime']) & (vasoactive_agent1['window_start7'] < vasoactive_agent1['endtime'])
)
```

In [ ]:
```
vasoactive_agent2 = vasoactive_agent1[vasoactive_agent1['vasoactive_agent'] == True]
```

In [ ]:
```
vasoactive_agent3= vasoactive_agent2.groupby('stay_id')['starttime'].idxmin()
vasoactive_agent4= vasoactive_agent2.loc[vasoactive_agent3]
```

In [ ]:
```
df23= df22.merge(vasoactive_agent4[['stay_id','vasoactive_agent']], on=['stay_id'], how='left')
df23.head()
```

Out[ ]:

	subject_id	stay_id	oasis	sirs	apsiii	weight	height	heart_rate	sbp	dbp	...	gcs	crp	BMI	urineoutput_24hr	ACEI/ARB	scr_min	ckd	CRRT	Mechanical ventilation	vasoactive_agent
0	10000690	37081114	41	3	52	55.3	NaN	79.0	107.0	63.0	...	15.0	NaN	NaN	815.0	1	0.6	0	0.0	False	NaN
1	10001884	37510196	35	3	51	65.0	157.0	76.0	127.0	73.0	...	15.0	NaN	26.37	1655.0	0	0.5	0	0.0	True	NaN
2	10002155	31090461	26	3	52	48.0	NaN	94.0	118.0	51.0	...	15.0	NaN	NaN	3500.0	0	2.1	1	0.0	False	NaN
3	10002443	35044219	24	4	41	156.1	178.0	92.0	121.0	79.0	...	15.0	NaN	49.27	3450.0	0	0.6	0	0.0	False	NaN
4	10002495	36753294	32	3	54	64.1	170.0	114.0	160.0	78.0	...	15.0	NaN	22.18	3400.0	1	1.1	0	0.0	True	True

5 rows × 107 columns

In [ ]:
```
df23['vasoactive_agent'] = df23['vasoactive_agent'].fillna(False)
```

In [ ]:
```
df23.shape
```

Out[ ]:
```
(10269, 107)
```

In [ ]:
```
sum(df23['vasoactive_agent'] == 1)
```

Out[ ]:
```
4337
```

In [ ]:
```
lods= pd.read_parquet(r'E:\MIMIC\MIMICIV3.1_Parquet\MIMIC_derived\lods.parquet')
lods.head()
```

Out[ ]:

	subject_id	hadm_id	stay_id	lods	neurologic	cardiovascular	renal	pulmonary	hematologic	hepatic
0	12466550	23998182	30000153	3	1.0	0.0	1.0	1	0.0	0.0
1	13180007	27543152	30000213	8	0.0	0.0	5.0	3	0.0	0.0
2	18421337	22413411	30000484	10	5.0	1.0	3.0	0	0.0	1.0
3	12207593	22795209	30000646	2	0.0	1.0	1.0	0	0.0	0.0
4	15726459	22744101	30000831	10	3.0	1.0	5.0	1	0.0	0.0

In [ ]:
```
lods=lods[['subject_id','stay_id','lods']]
```

In [ ]:
```
lods.head()
```

Out[ ]:

	subject_id	stay_id	lods
0	12466550	30000153	3
1	13180007	30000213	8
2	18421337	30000484	10
3	12207593	30000646	2
4	15726459	30000831	10

In [ ]:
```
lods.shape
```

Out[ ]:
```
(94458, 3)
```

In [ ]:
```
lods.stay_id.nunique()
```

Out[ ]:
```
94458
```

In [ ]:
```
df23['stay_id'] = df23['stay_id'].astype('Int64')
lods['stay_id'] = lods['stay_id'].astype('Int64')
```

In [ ]:
```
df24= df23.merge(lods, on=['subject_id', 'stay_id'], how='left')
```

In [ ]:
```
df24.shape
```

Out[ ]: (10269, 108)

In [ ]:
```
df24.stay_id.nunique()
```

Out[ ]: 10269

In [ ]:
```
df24.subject_id.nunique()
```

Out[ ]: 10269

In [ ]:
```
df24.hadm_id.nunique()
```

Out[ ]: 10269

In [ ]:
```
df24.columns
```

In [ ]:
```
print(df24['scr_min'].value_counts().to_string())
```

In [ ]:
```
msno.bar(df24)
```

数据筛选

In [ ]:
```
final1=df24
```

In [ ]:
```
final1.shape
```

Out[ ]: (10269, 108)

In [ ]:
```
final1.subject_id.nunique()
```

Out[ ]: 10269

In [ ]:
```
final1.stay_id.nunique()
```

Out[ ]: 10269

In [ ]:
```
final1.hadm_id.nunique()
```

Out[ ]: 10269

In [ ]:
```
print(list(final1.columns))
```

```
['subject_id', 'stay_id', 'oasis', 'sirs', 'apsiii', 'weight', 'height', 'heart_rate', 'sbp', 'dbp', 'mbp', 'resp_rate', 'temperature', 'spo2', 'hadm_id', 'age_score', 'myocardial_infarct', 'congestive_heart_failure', 'peripheral_vascular_disease', 'cerebrovascular_disease', 'dementia', 'chronic_pulmonary_disease', 'rheumatic_disease', 'peptic_ulcer_disease', 'mild_liver_disease', 'diabetes_without_cc', 'diabetes_with_cc', 'paraplegia', 'renal_disease', 'malignant_cancer', 'severe_liver_disease', 'metastatic_solid_tumor', 'aids', 'charlson_comorbidity_index', 'aki_charttime_max', 'aki_charttime', 'aki_stage', 'gender', 'dod', 'admittime', 'dischtime', 'admission_age', 'race', 'icu_intime', 'icu_outtime', 'sofa_time', 'sofa_score', 'infection_time', 'los_anti-infe', 'max_time', 'window_end', 'saki', 'los_aki', 'HTN', 'AKD', 'los_icu-aki', 'window_start7', 'window_start-1', 'los_dod', 'po2', 'pco2', 'aado2_calc', 'pao2fio2ratio', 'ph', 'baseexcess', 'totalco2', 'lactate', 'wbc', 'basophils_abs', 'eosinophils_abs', 'lymphocytes_abs', 'monocytes_abs', 'neutrophils_abs', 'troponin_t', 'ck_mb', 'ntprobnp', 'aniongap', 'bicarbonate', 'bun', 'calcium', 'chloride', 'creatinine', 'glucose', 'sodium', 'potassium', 'inr', 'pt', 'ptt', 'hematocrit', 'hemoglobin', 'mch', 'mchc', 'mcv', 'platelet', 'rbc', 'rdw', 'WBC1', 'gcs', 'crp', 'BMI', 'urineoutput_24hr', 'ACEI/ARB', 'scr_min', 'ckd', 'CRRT', 'Mechanical ventilation', 'vasoactive_agent', 'lods']
```

In [ ]:
```
missing_percentages = final1.isnull().mean()

# 找出缺失数据超过50%的列
columns_to_drop = missing_percentages[missing_percentages > 0.30].index

# 删除这些列
final2 = final1.drop(columns=columns_to_drop)
```

In [ ]:
```
final2.shape
```

Out[ ]: (10269, 93)

In [ ]:
```
final2 = final2.drop(['height', 'BMI'], axis=1)
```

In [ ]:
```
msno.bar(final2)
```

In [ ]:
```
# 计算每行的缺失值比例
missing_ratio_per_row = final2.isna().mean(axis=1)

# 保留缺失比例≤10%的行
final3 = final2[missing_ratio_per_row <= 0.15]
```

In [ ]:
```
final3.shape
```

Out[ ]: (10234, 91)

In [ ]:
```
msno.bar(final3)
```

In [ ]:
```
final3.AKD.value_counts()
```

Out[ ]:
```
AKD
0.0    7765
1.0    2469
Name: count, dtype: int64
```

In [ ]:
```
print(list(final3.columns))
```

```
['subject_id', 'stay_id', 'oasis', 'sirs', 'apsiii', 'weight', 'heart_rate', 'sbp', 'dbp', 'mbp', 'resp_rate', 'temperature', 'spo2', 'hadm_id', 'age_score', 'myocardial_infarct', 'congestive_heart_failure', 'peripheral_vascular_disease', 'cerebrovascular_disease', 'dementia', 'chronic_pulmonary_disease', 'rheumatic_disease', 'peptic_ulcer_disease', 'mild_liver_disease', 'diabetes_without_cc', 'diabetes_with_cc', 'paraplegia', 'renal_disease', 'malignant_cancer', 'severe_liver_disease', 'metastatic_solid_tumor', 'aids', 'charlson_comorbidity_index', 'aki_charttime_max', 'aki_charttime', 'aki_stage', 'gender', 'admittime', 'dischtime', 'admission_age', 'race', 'icu_intime', 'icu_outtime', 'sofa_time', 'sofa_score', 'infection_time', 'los_anti-infe', 'max_time', 'window_end', 'saki', 'los_aki', 'HTN', 'AKD', 'los_icu-aki', 'window_start7', 'window_start-1', 'po2', 'pco2', 'ph', 'baseexcess', 'totalco2', 'aniongap', 'bicarbonate', 'bun', 'calcium', 'chloride', 'creatinine', 'glucose', 'sodium', 'potassium', 'inr', 'pt', 'ptt', 'hematocrit', 'hemoglobin', 'mch', 'mchc', 'mcv', 'platelet', 'rbc', 'rdw', 'WBC1', 'gcs', 'urineoutput_24hr', 'ACEI/ARB', 'scr_min', 'ckd', 'CRRT', 'Mechanical ventilation', 'vasoactive_agent', 'lods']
```

In [ ]:
```
columns_to_drop = [col for col in final3.columns if '_charttime' in col]
final4 = final3.drop(columns=columns_to_drop)
```

In [ ]:
```
final4.shape
```

Out[ ]: (10234, 89)

In [ ]:
```
columns_to_drop = [col for col in final4.columns if '_time' in col]
final5 = final4.drop(columns=columns_to_drop)
```

In [ ]:
```
final5.shape
```

Out[ ]: (10234, 86)

In [ ]:
```
final5.head()
```

Out[ ]:

	subject_id	stay_id	oasis	sirs	apsiii	weight	heart_rate	sbp	dbp	mbp	...	WBC1	gcs	urineoutput_24hr	ACEI/ARB	scr_min	ckd	CRRT	Mechanical ventilation	vasoactive_agent	lods
0	10000690	37081114	41	3	52	55.3	79.0	107.0	63.0	71.0	...	7.5	15.0	815.0	1	0.6	0	0.0	False	False	5
1	10001884	37510196	35	3	51	65.0	76.0	127.0	73.0	87.0	...	12.0	15.0	1655.0	0	0.5	0	0.0	True	False	7
2	10002155	31090461	26	3	52	48.0	94.0	118.0	51.0	68.0	...	9.5	15.0	3500.0	0	2.1	1	0.0	False	False	6
3	10002443	35044219	24	4	41	156.1	92.0	121.0	79.0	92.0	...	15.9	15.0	3450.0	0	0.6	0	0.0	False	False	2
4	10002495	36753294	32	3	54	64.1	114.0	160.0	78.0	94.0	...	36.8	15.0	3400.0	1	1.1	0	0.0	True	True	4

5 rows × 86 columns

In [ ]:
```
print(list(final5.columns))
```

```
['subject_id', 'stay_id', 'oasis', 'sirs', 'apsiii', 'weight', 'heart_rate', 'sbp', 'dbp', 'mbp', 'resp_rate', 'temperature', 'spo2', 'hadm_id', 'age_score', 'myocardial_infarct', 'congestive_
heart_failure', 'peripheral_vascular_disease', 'cerebrovascular_disease', 'dementia', 'chronic_pulmonary_disease', 'rheumatic_disease', 'peptic_ulcer_disease', 'mild_liver_disease', 'diabetes_
without_cc', 'diabetes_with_cc', 'paraplegia', 'renal_disease', 'malignant_cancer', 'severe_liver_disease', 'metastatic_solid_tumor', 'aids', 'charlson_comorbidity_index', 'aki_stage', 'gende
r', 'admittime', 'dischtime', 'admission_age', 'race', 'icu_intime', 'icu_outtime', 'sofa_score', 'los_anti-infe', 'window_end', 'saki', 'los_aki', 'HTN', 'AKD', 'los_icu-aki', 'window_start
7', 'window_start-1', 'po2', 'pco2', 'ph', 'baseexcess', 'totalco2', 'aniongap', 'bicarbonate', 'bun', 'calcium', 'chloride', 'creatinine', 'glucose', 'sodium', 'potassium', 'inr', 'pt', 'pt
t', 'hematocrit', 'hemoglobin', 'mch', 'mchc', 'mcv', 'platelet', 'rbc', 'rdw', 'WBC1', 'gcs', 'urineoutput_24hr', 'ACEI/ARB', 'scr_min', 'ckd', 'CRRT', 'Mechanical ventilation', 'vasoactive_a
gent', 'lods']
```

In [ ]:
```
final6 = final5.drop(['admittime', 'dischtime','race', 'icu_intime', 'icu_outtime','window_end','window_start7','window_start-1','WBC1'], axis=1)
```

In [ ]:
```
print(list(final6.columns))
```

```
['subject_id', 'stay_id', 'oasis', 'sirs', 'apsiii', 'weight', 'heart_rate', 'sbp', 'dbp', 'mbp', 'resp_rate', 'temperature', 'spo2', 'hadm_id', 'age_score', 'myocardial_infarct', 'congestive_
heart_failure', 'peripheral_vascular_disease', 'cerebrovascular_disease', 'dementia', 'chronic_pulmonary_disease', 'rheumatic_disease', 'peptic_ulcer_disease', 'mild_liver_disease', 'diabetes_
without_cc', 'diabetes_with_cc', 'paraplegia', 'renal_disease', 'malignant_cancer', 'severe_liver_disease', 'metastatic_solid_tumor', 'aids', 'charlson_comorbidity_index', 'aki_stage', 'gende
r', 'admission_age', 'sofa_score', 'los_anti-infe', 'saki', 'los_aki', 'HTN', 'AKD', 'los_icu-aki', 'po2', 'pco2', 'ph', 'baseexcess', 'totalco2', 'aniongap', 'bicarbonate', 'bun', 'calcium',
'chloride', 'creatinine', 'glucose', 'sodium', 'potassium', 'inr', 'pt', 'ptt', 'hematocrit', 'hemoglobin', 'mch', 'mchc', 'mcv', 'platelet', 'rbc', 'rdw', 'gcs', 'urineoutput_24hr', 'ACEI/AR
B', 'scr_min', 'ckd', 'CRRT', 'Mechanical ventilation', 'vasoactive_agent', 'lods']
```

In [ ]:
```
final6['diabetes'] = final6.apply(lambda row: 1 if 1 in [row['diabetes_without_cc'], row['diabetes_with_cc']] else 0, axis=1)

# 删除原本的两列
final6.drop(['diabetes_without_cc', 'diabetes_with_cc'], axis=1, inplace=True)
```

In [ ]:
```
final6.shape
```

Out[ ]:
```
(10234, 76)
```

In [ ]:
```
final6['liver_disease'] = final6.apply(lambda row: 1 if 1 in [row['severe_liver_disease'], row['mild_liver_disease']] else 0, axis=1)

# 删除原本的两列
final6.drop(['severe_liver_disease', 'mild_liver_disease'], axis=1, inplace=True)
```

In [ ]:
```
final6.malignant_cancer.value_counts()
```

Out[ ]:
```
malignant_cancer
0    9003
1    1231
Name: count, dtype: int64
```

In [ ]:
```
final6.shape
```

Out[ ]:
```
(10234, 75)
```

In [ ]:
```
final6.rename(columns={
    'saki': 'SA-AKI',
    'admission_age': 'age',
    'crrt': 'CRRT',
    'oasis': 'OASIS',
    'sirs': 'SIRS',
    'apsiii': 'APS III',
    'diabetes': 'DM',
    'sofa_score': 'SOFA score',
    'aki_stage': 'AKI stage',
    'ckd': 'CKD',
    'aids': 'AIDS',
    'lods': 'LODS',
}, inplace=True)
```

In [ ]:
```
print(list(final6.columns))
```

```
['subject_id', 'stay_id', 'OASIS', 'SIRS', 'APS III', 'weight', 'heart_rate', 'sbp', 'dbp', 'mbp', 'resp_rate', 'temperature', 'spo2', 'hadm_id', 'age_score', 'myocardial_infarct', 'congestive
_heart_failure', 'peripheral_vascular_disease', 'cerebrovascular_disease', 'dementia', 'chronic_pulmonary_disease', 'rheumatic_disease', 'peptic_ulcer_disease', 'paraplegia', 'renal_disease',
'malignant_cancer', 'metastatic_solid_tumor', 'AIDS', 'charlson_comorbidity_index', 'AKI stage', 'gender', 'age', 'SOFA score', 'los_anti-infe', 'SA-AKI', 'los_aki', 'HTN', 'AKD', 'los_icu-ak
i', 'po2', 'pco2', 'ph', 'baseexcess', 'totalco2', 'aniongap', 'bicarbonate', 'bun', 'calcium', 'chloride', 'creatinine', 'glucose', 'sodium', 'potassium', 'inr', 'pt', 'ptt', 'hematocrit', 'h
emoglobin', 'mch', 'mchc', 'mcv', 'platelet', 'rbc', 'rdw', 'gcs', 'urineoutput_24hr', 'ACEI/ARB', 'scr_min', 'CKD', 'CRRT', 'Mechanical ventilation', 'vasoactive_agent', 'LODS', 'DM', 'liver_
disease']
```

In [ ]:
```
final6.drop(['los_aki','age_score','creatinine'], axis=1, inplace=True)
```

In [ ]:
```
final6.shape
```

Out[ ]:
```
(10234, 72)
```

In [ ]:
```
def convert_to_hours(time_str):
    try:
        # 将字符串转为 Timedelta 类型
        td = pd.to_timedelta(time_str)
        # 计算总秒数，再除以 3600 得到小时
        return td.total_seconds() / 3600
    except:
        return None  # 异常值处理（如空值或无效格式）

# 应用转换函数到指定列
final6['Los_inf._AB'] = final6['los_anti-infe'].apply(convert_to_hours)
final6['Los_icu_aki'] = final6['los_icu-aki'].apply(convert_to_hours)

final6['Los_inf._AB'] = final6['Los_inf._AB'].apply(lambda x: 0 if x < 0 else x)
final6['Los_icu_aki'] = final6['Los_icu_aki'].apply(lambda x: 0 if x < 0 else x)
# 删除原始列（可选）
final6.drop(['los_anti-infe', 'los_icu-aki'], axis=1, inplace=True)
```

In [ ]:
```
final6.shape
```

Out[ ]:
```
(10234, 72)
```

In [ ]:
```
print(list(final6.columns))
```

```
['subject_id', 'stay_id', 'OASIS', 'SIRS', 'APS III', 'weight', 'heart_rate', 'sbp', 'dbp', 'mbp', 'resp_rate', 'temperature', 'spo2', 'hadm_id', 'myocardial_infarct', 'congestive_heart_failur
e', 'peripheral_vascular_disease', 'cerebrovascular_disease', 'dementia', 'chronic_pulmonary_disease', 'rheumatic_disease', 'peptic_ulcer_disease', 'paraplegia', 'renal_disease', 'malignant_ca
ncer', 'metastatic_solid_tumor', 'AIDS', 'charlson_comorbidity_index', 'AKI stage', 'gender', 'age', 'SOFA score', 'SA-AKI', 'HTN', 'AKD', 'po2', 'pco2', 'ph', 'baseexcess', 'totalco2', 'anion
gap', 'bicarbonate', 'bun', 'calcium', 'chloride', 'glucose', 'sodium', 'potassium', 'inr', 'pt', 'ptt', 'hematocrit', 'hemoglobin', 'mch', 'mchc', 'mcv', 'platelet', 'rbc', 'rdw', 'gcs', 'uri
neoutput_24hr', 'ACEI/ARB', 'scr_min', 'CKD', 'CRRT', 'Mechanical ventilation', 'vasoactive_agent', 'LODS', 'DM', 'liver_disease', 'Los_inf._AB', 'Los_icu_aki']
```

In [ ]:
```
binary_cols = ['liver_disease', 'DM', 'vasoactive_agent','Mechanical ventilation','CRRT','CKD','ACEI/ARB',
               'myocardial_infarct', 'congestive_heart_failure', 'peripheral_vascular_disease', 'cerebrovascular_disease', 'dementia',
               'chronic_pulmonary_disease', 'rheumatic_disease', 'peptic_ulcer_disease', 'paraplegia', 'renal_disease', 'malignant_cancer',
```

```
                'metastatic_solid_tumor', 'AIDS','HTN', 'AKD',]
final6.replace({col: {0: 'NO', 1: 'YES'} for col in binary_cols}, inplace=True)
```

In [ ]:
```
binary_cols = [ 'vasoactive_agent','Mechanical ventilation',]
final6.replace({col: {False: 'NO', True: 'YES'} for col in binary_cols}, inplace=True)
```

In [ ]:
```
final6.rename(columns={'baseexcess':'Base Excess'}, inplace=True)
```

In [ ]:
```
final6['calcium'] = final6['calcium']  / 2.54
```

In [ ]:
```
final6.columns = [col[0].upper() + col[1:] if col else col for col in final6.columns]
```

In [ ]:
```
print(list(final6.columns))
```

```
['Subject_id', 'Stay_id', 'OASIS', 'SIRS', 'APS III', 'Weight', 'Heart_rate', 'Sbp', 'Dbp', 'Mbp', 'Resp_rate', 'Temperature', 'Spo2', 'Hadm_id', 'Myocardial_infarct', 'Congestive_heart_failur
e', 'Peripheral_vascular_disease', 'Cerebrovascular_disease', 'Dementia', 'Chronic_pulmonary_disease', 'Rheumatic_disease', 'Peptic_ulcer_disease', 'Paraplegia', 'Renal_disease', 'Malignant_ca
ncer', 'Metastatic_solid_tumor', 'AIDS', 'Charlson_comorbidity_index', 'AKI stage', 'Gender', 'Age', 'SOFA score', 'SA-AKI', 'HTN', 'AKD', 'Po2', 'Pco2', 'Ph', 'Base Excess', 'Totalco2', 'Anio
ngap', 'Bicarbonate', 'Bun', 'Calcium', 'Chloride', 'Glucose', 'Sodium', 'Potassium', 'Inr', 'Pt', 'Ptt', 'Hematocrit', 'Hemoglobin', 'Mch', 'Mchc', 'Mcv', 'Platelet', 'Rbc', 'Rdw', 'Gcs', 'Ur
ineoutput_24hr', 'ACEI/ARB', 'Scr_min', 'CKD', 'CRRT', 'Mechanical ventilation', 'Vasoactive_agent', 'LODS', 'DM', 'Liver_disease', 'Los_inf._AB', 'Los_icu_aki']
```

In [ ]:
```
column_names_to_upper = ['Gcs','Inr','Pt','Ptt','Mch','Mchc', 'Mcv', 'Rbc', 'Rdw', 'Wbc', 'Sbp', 'Dbp', 'Mbp','Ph','Bun']
final6.columns = [col.upper() if col in column_names_to_upper else col for col in final6.columns]
```

In [ ]:
```
final6['Hemoglobin'] = final6['Hemoglobin'] * 10
```

In [ ]:
```
final6['Glucose'] = final6['Glucose'] / 18.0182
```

In [ ]:
```
final6['Scr_min'] = final6['Scr_min'] * 88.4
```

final6['NLR'] = final6['Neutrophils_abs'] / final6['Lymphocytes_abs'].replace(0, np.nan)

计算 PIV，避免除以零

final6['PIV'] = (final6['Neutrophils_abs'] * final6['Platelet'] * final6['Monocytes_abs']) / final6['Lymphocytes_abs'].replace(0, np.nan)

检查并处理无穷大或过大的值

final6.replace([np.inf, -np.inf], np.nan, inplace=True)

In [ ]:
```
print(list(final6.columns))
```

```
['Subject_id', 'Stay_id', 'OASIS', 'SIRS', 'APS III', 'Weight', 'Heart_rate', 'SBP', 'DBP', 'MBP', 'Resp_rate', 'Temperature', 'Spo2', 'Hadm_id', 'Myocardial_infarct', 'Congestive_heart_failur
e', 'Peripheral_vascular_disease', 'Cerebrovascular_disease', 'Dementia', 'Chronic_pulmonary_disease', 'Rheumatic_disease', 'Peptic_ulcer_disease', 'Paraplegia', 'Renal_disease', 'Malignant_ca
ncer', 'Metastatic_solid_tumor', 'AIDS', 'Charlson_comorbidity_index', 'AKI stage', 'Gender', 'Age', 'SOFA score', 'SA-AKI', 'HTN', 'AKD', 'Po2', 'Pco2', 'PH', 'Base Excess', 'Totalco2', 'Anio
ngap', 'Bicarbonate', 'BUN', 'Calcium', 'Chloride', 'Glucose', 'Sodium', 'Potassium', 'INR', 'PT', 'PTT', 'Hematocrit', 'Hemoglobin', 'MCH', 'MCHC', 'MCV', 'Platelet', 'RBC', 'RDW', 'GCS', 'Ur
ineoutput_24hr', 'ACEI/ARB', 'Scr_min', 'CKD', 'CRRT', 'Mechanical ventilation', 'Vasoactive_agent', 'LODS', 'DM', 'Liver_disease', 'Los_inf._AB', 'Los_icu_aki']
```

In [ ]:
```
final6['BUN'] = final6['BUN'] * 0.357
```

In [ ]:
```
# 以下是修改指定列，保留小数点后2位的代码
columns_to_round = ['Los_inf._AB','Los_icu_aki','Age','PH','Scr_min','Calcium','Chloride','Glucose','BUN','Pao2fio2ratio']
for col in columns_to_round:
    if col in final6.columns:  # 确保列名存在于 DataFrame 中
        final6[col] = final6[col].round(2)

# 现在 df 中的指定列已经保留了小数点后2位
```

In [ ]:
```
msno.bar(final6)
```

In [ ]:
```
final6.shape
```

Out[ ]:
```
(10234, 72)
```

final6.to_csv('250501重制版清洗后数据v1 saki.csv')