```
from google.colab import drive
drive.mount('/content/drive/')
```

```
     Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive.mount("/content/drive/", force_remount=True).
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime , timedelta
```

```
data1=pd.read_csv(r"/content/sampledata turnaround time analysis.csv")
```

```
data1.head(10)
```

	FLIGHT	SCHEDULED ARRIVAL	EXPECTED ARRIVAL	CREW READY	JET BRIDGE ON	UNLOADING OF PASSENGERS START	UNLOADING OF PASSENGERS END	UNL BAGS	UNL BAGS END	REFUEL START	REFUEL ENDS	CLEANING STARTS	CLEANING ENDS	RESTOCK FOOD	RESTOCK FOOD ENDS	RELOADING BAG STR	RELOADING BAGS END	RELOADING PASSENGERS	RELOADING PASSENGERS END	JE BRIDG OF
0	I52467	11:12	11:01	10:54	11:03	11:04	11:13	11:05	11:10	11:05	11:16	11:14	11:20	11:08	11:12	11:13	11:18	11:23	11:26	11:2
1	UK812	11:30	11:30	11:20	11:33	11:34	11:45	11:35	11:40	11:36	11:50	11:47	11:53	11:37	11:43	11:43	11:49	11:58	12:07	12:1
2	CX44	11:45	11:50	11:42	11:53	11:55	12:07	11:56	12:02	11:57	12:11	12:02	12:09	11:58	12:03	12:04	12:10	12:13	12:18	12:2
3	6E2133	12:02	12:05	11:55	12:07	12:09	12:21	12:11	12:17	12:10	12:25	12:18	12:25	12:12	12:16	12:20	12:24	12:28	12:34	12:3
4	AI565	12:20	12:18	12:10	12:21	12:23	12:30	12:25	12:31	12:22	12:37	12:31	12:37	12:26	12:33	12:33	12:37	12:39	12:45	12:4
5	6E388	12:45	12:45	12:35	12:47	12:50	01:03	12:50	12:56	12:50	01:10	12:54	01:01	12:54	12:59	12:58	01:05	01:05	01:11	01:1
6	9I513	01:00	01:00	12:55	01:05	01:08	01:20	01:07	01:15	01:09	01:26	01:22	01:28	01:10	01:16	01:16	01:22	01:30	01:35	01:3
7	6E435	01:20	01:27	01:18	01:30	01:32	01:42	01:33	01:37	01:34	01:50	01:43	01:50	01:34	01:40	01:38	01:44	01:44	01:52	01:5
8	9I536	01:45	01:40	01:33	01:43	01:45	01:52	01:44	01:50	01:45	02:00	01:53	02:00	01:48	01:55	01:52	01:59	01:56	02:04	02:0
9	9I898	02:00	02:08	02:00	02:10	02:13	02:20	02:13	02:19	02:12	02:28	02:20	02:26	02:18	02:24	02:22	02:26	02:28	02:38	02:4

```
DATA=pd.read_csv(r'/content/sampledata turnaround time analysis.csv', index_col='FLIGHT')
```

```
DATA.drop(['6E388'], inplace= True)
```

```
DATA.drop(['6E8239'], inplace= True)
```

```
DATA.head(10)
```

	SCHEDULED ARRIVAL	EXPECTED ARRIVAL	CREW READY	JET BRIDGE ON	UNLOADING OF PASSENGERS START	UNLOADING OF PASSENGERS END	UNL BAGS	UNL BAGS END	REFUEL START	REFUEL ENDS	CLEANING STARTS	CLEANING ENDS	RESTOCK FOOD	RE
FLIGHT														
I52467	11:12	11:01	10:54	11:03	11:04	11:13	11:05	11:10	11:05	11:16	11:14	11:20	11:08	
UK812	11:30	11:30	11:20	11:33	11:34	11:45	11:35	11:40	11:36	11:50	11:47	11:53	11:37	
CX44	11:45	11:50	11:42	11:53	11:55	12:07	11:56	12:02	11:57	12:11	12:02	12:09	11:58	
6E2133	12:02	12:05	11:55	12:07	12:09	12:21	12:11	12:17	12:10	12:25	12:18	12:25	12:12	
AI565	12:20	12:18	12:10	12:21	12:23	12:30	12:25	12:31	12:22	12:37	12:31	12:37	12:26	
9I513	01:00	01:00	12:55	01:05	01:08	01:20	01:07	01:15	01:09	01:26	01:22	01:28	01:10	
6E435	01:20	01:27	01:18	01:30	01:32	01:42	01:33	01:37	01:34	01:50	01:43	01:50	01:34	
9I536	01:45	01:40	01:33	01:43	01:45	01:52	01:44	01:50	01:45	02:00	01:53	02:00	01:48	
9I898	02:00	02:08	02:00	02:10	02:13	02:20	02:13	02:19	02:12	02:28	02:20	02:26	02:18	

```
DATA.dropna(inplace=True)
```

```
DATA['SCHEDULED ARRIVAL'].mean
```

```
<bound method Series.mean of FLIGHT
I52467    11:12
UK812     11:30
CX44      11:45
6E2133    12:02
AI565     12:20
9I513     01:00
6E435     01:20
9I536     01:45
9I898     02:00
9I876     02:20
I52990    02:30
6E734     02:50
6E379     03:10
6E587     03:30
SG706     03:50
SG3008    04:10
6E847     04:30
G8834     04:45
6E477     05:00
SG7010    05:20
6E332     05:45
G8805     06:10
6E6188    06:30
AI503     07:05
I51624    07:20
G8406     07:45
6E2487    08:00
6E1456    08:20
Name: SCHEDULED ARRIVAL, dtype: object>
```

DATA["SCHEDULED ARRIVAL"]=DATA["SCHEDULED ARRIVAL"].astype("datetime64[ns]") DATA["EXPECTED ARRIVAL"]=DATA["EXPECTED ARRIVAL"].astype("datetime64[ns]") DATA['CREW READY']=DATA["CREW READY"].astype("datetime64[ns]") DATA['JET BRIDGE ON']=DATA['JET BRIDGE ON'].astype("datetime64[ns]") DATA['UNLOADING OF PASSENGERS START']=DATA['UNLOADING OF PASSENGERS START'].astype("datetime64[ns]") DATA['UNLOADING OF PASSENGERS END']=DATA['UNLOADING OF PASSENGERS END'].astype("datetime64[ns]") DATA['UNL BAGS ']=DATA['UNL BAGS '].astype("datetime64[ns]") DATA['UNL BAGS END']=DATA['UNL BAGS END'].astype("datetime64[ns]") DATA['REFUEL START']=DATA['REFUEL START'].astype("datetime64[ns]") DATA['REFUEL ENDS']=DATA['REFUEL ENDS'].astype("datetime64[ns]") DATA['CLEANING STARTS']=DATA['CLEANING STARTS'].astype("datetime64[ns]") DATA['CLEANING ENDS']=DATA['CLEANING ENDS'].astype("datetime64[ns]") DATA['RESTOCK FOOD']=DATA['RESTOCK FOOD'].astype("datetime64[ns]") DATA['RESTOCK FOOD ENDS']=DATA['RESTOCK FOOD ENDS'].astype("datetime64[ns]") DATA['RELOADING BAG STR']=DATA['RELOADING BAG STR'].astype("datetime64[ns]") DATA['RELOADING BAGS END']=DATA['RELOADING BAGS END'].astype("datetime64[ns]") DATA['RELOADING PASSENGERS']=DATA['RELOADING PASSENGERS'].astype("datetime64[ns]") DATA['RELOADING PASSENGERS END']=DATA['RELOADING PASSENGERS END'].astype("datetime64[ns]") DATA['JET BRIDGE OFF']=DATA['JET BRIDGE OFF'].astype("datetime64[ns]") DATA['DOORS CLOSED']=DATA['DOORS CLOSED'].astype("datetime64[ns]") DATA['PUSH BACK']=DATA['PUSH BACK'].astype("datetime64[ns]") DATA['TAKE OFF']=DATA['TAKE OFF'].astype("datetime64[ns]")

```
DATA["SCHEDULED ARRIVAL"]=DATA["SCHEDULED ARRIVAL"].astype("datetime64[ns]")
DATA["EXPECTED ARRIVAL"]=DATA["EXPECTED ARRIVAL"].astype("datetime64[ns]")
DATA['CREW READY']=DATA["CREW READY"].astype("datetime64[ns]")
DATA['JET BRIDGE ON']=DATA['JET BRIDGE ON'].astype("datetime64[ns]")
DATA['UNLOADING OF PASSENGERS START']=DATA['UNLOADING OF PASSENGERS START'].astype("datetime64[ns]")
DATA['UNLOADING OF PASSENGERS END']=DATA['UNLOADING OF PASSENGERS END'].astype("datetime64[ns]")
DATA['UNL BAGS ']=DATA['UNL BAGS '].astype("datetime64[ns]")
DATA['UNL BAGS END']=DATA['UNL BAGS END'].astype("datetime64[ns]")
DATA['REFUEL START']=DATA['REFUEL START'].astype("datetime64[ns]")
DATA['REFUEL ENDS']=DATA['REFUEL ENDS'].astype("datetime64[ns]")
DATA['CLEANING STARTS']=DATA['CLEANING STARTS'].astype("datetime64[ns]")
DATA['CLEANING ENDS']=DATA['CLEANING ENDS'].astype("datetime64[ns]")
DATA['RESTOCK FOOD']=DATA['RESTOCK FOOD'].astype("datetime64[ns]")
DATA['RESTOCK FOOD ENDS']=DATA['RESTOCK FOOD ENDS'].astype("datetime64[ns]")
DATA['RELOADING BAG STR']=DATA['RELOADING BAG STR'].astype("datetime64[ns]")
DATA['RELOADING BAGS END']=DATA['RELOADING BAGS END'].astype("datetime64[ns]")
DATA['RELOADING PASSENGERS']=DATA['RELOADING PASSENGERS'].astype("datetime64[ns]")
DATA['RELOADING PASSENGERS END']=DATA['RELOADING PASSENGERS END'].astype("datetime64[ns]")
DATA['JET BRIDGE OFF']=DATA['JET BRIDGE OFF'].astype("datetime64[ns]")
DATA['DOORS CLOSED']=DATA['DOORS CLOSED'].astype("datetime64[ns]")
DATA['PUSH BACK']=DATA['PUSH BACK'].astype("datetime64[ns]")
DATA['TAKE OFF']=DATA['TAKE OFF'].astype("datetime64[ns]")
DATA['EXPECTED TAKE OFF']=DATA['EXPECTED TAKE OFF'].astype("datetime64[ns]")
```

```
DATA.head(10)
```

	SCHEDULED ARRIVAL	EXPECTED ARRIVAL	CREW READY	JET BRIDGE ON	UNLOADING OF PASSENGERS START	UNLOADING OF PASSENGERS END	UNL BAGS	UNL BAGS END	REFUEL START	REFUEL ENDS	CLEANING STARTS	CLEANING ENDS	F
FLIGHT													
I52467	2021-05-08 11:12:00	2021-05-08 11:01:00	2021-05-08 10:54:00	2021-05-08 11:03:00	2021-05-08 11:04:00	2021-05-08 11:13:00	2021-05-08 11:05:00	2021-05-08 11:10:00	2021-05-08 11:05:00	2021-05-08 11:16:00	2021-05-08 11:14:00	2021-05-08 11:20:00	1
UK812	2021-05-08 11:30:00	2021-05-08 11:30:00	2021-05-08 11:20:00	2021-05-08 11:33:00	2021-05-08 11:34:00	2021-05-08 11:45:00	2021-05-08 11:35:00	2021-05-08 11:40:00	2021-05-08 11:36:00	2021-05-08 11:50:00	2021-05-08 11:47:00	2021-05-08 11:53:00	1
CX44	2021-05-08 11:45:00	2021-05-08 11:50:00	2021-05-08 11:42:00	2021-05-08 11:53:00	2021-05-08 11:55:00	2021-05-08 12:07:00	2021-05-08 11:56:00	2021-05-08 12:02:00	2021-05-08 11:57:00	2021-05-08 12:11:00	2021-05-08 12:02:00	2021-05-08 12:09:00	1
6E2133	2021-05-08 12:02:00	2021-05-08 12:05:00	2021-05-08 11:55:00	2021-05-08 12:07:00	2021-05-08 12:09:00	2021-05-08 12:21:00	2021-05-08 12:11:00	2021-05-08 12:17:00	2021-05-08 12:10:00	2021-05-08 12:25:00	2021-05-08 12:18:00	2021-05-08 12:25:00	1
AI565	2021-05-08 12:20:00	2021-05-08 12:18:00	2021-05-08 12:10:00	2021-05-08 12:21:00	2021-05-08 12:23:00	2021-05-08 12:30:00	2021-05-08 12:25:00	2021-05-08 12:31:00	2021-05-08 12:22:00	2021-05-08 12:37:00	2021-05-08 12:31:00	2021-05-08 12:37:00	1
9I513	2021-05-08 01:00:00	2021-05-08 01:00:00	2021-05-08 12:55:00	2021-05-08 01:05:00	2021-05-08 01:08:00	2021-05-08 01:20:00	2021-05-08 01:07:00	2021-05-08 01:15:00	2021-05-08 01:09:00	2021-05-08 01:26:00	2021-05-08 01:22:00	2021-05-08 01:28:00	0
6E435	2021-05-08 01:20:00	2021-05-08 01:27:00	2021-05-08 01:18:00	2021-05-08 01:30:00	2021-05-08 01:32:00	2021-05-08 01:42:00	2021-05-08 01:33:00	2021-05-08 01:37:00	2021-05-08 01:34:00	2021-05-08 01:50:00	2021-05-08 01:43:00	2021-05-08 01:50:00	0
9I536	2021-05-08 01:45:00	2021-05-08 01:40:00	2021-05-08 01:33:00	2021-05-08 01:43:00	2021-05-08 01:45:00	2021-05-08 01:52:00	2021-05-08 01:44:00	2021-05-08 01:50:00	2021-05-08 01:45:00	2021-05-08 02:00:00	2021-05-08 01:53:00	2021-05-08 02:00:00	0
	2021-05-	2021-05-	2021-	2021-	[illegible]	[illegible]	2021-	2021-	2021-	2021-	2021-05-	2021-05-	

```
DATA.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 28 entries, I52467 to 6E1456
Data columns (total 23 columns):
 #   Column                         Non-Null Count  Dtype         
---  ------                         --------------  -----         
 0   SCHEDULED ARRIVAL              28 non-null     datetime64[ns]
 1   EXPECTED ARRIVAL               28 non-null     datetime64[ns]
 2   CREW READY                     28 non-null     datetime64[ns]
 3   JET BRIDGE ON                  28 non-null     datetime64[ns]
 4   UNLOADING OF PASSENGERS START  28 non-null     datetime64[ns]
 5   UNLOADING OF PASSENGERS END    28 non-null     datetime64[ns]
 6   UNL BAGS                       28 non-null     datetime64[ns]
 7   UNL BAGS END                   28 non-null     datetime64[ns]
 8   REFUEL START                   28 non-null     datetime64[ns]
 9   REFUEL ENDS                    28 non-null     datetime64[ns]
 10  CLEANING STARTS                28 non-null     datetime64[ns]
 11  CLEANING ENDS                  28 non-null     datetime64[ns]
 12  RESTOCK FOOD                   28 non-null     datetime64[ns]
 13  RESTOCK FOOD ENDS              28 non-null     datetime64[ns]
 14  RELOADING BAG STR             28 non-null     datetime64[ns]
 15  RELOADING BAGS END            28 non-null     datetime64[ns]
 16  RELOADING PASSENGERS          28 non-null     datetime64[ns]
 17  RELOADING PASSENGERS END      28 non-null     datetime64[ns]
 18  JET BRIDGE OFF                28 non-null     datetime64[ns]
 19  DOORS CLOSED                  28 non-null     datetime64[ns]
 20  PUSH BACK                     28 non-null     datetime64[ns]
 21  TAKE OFF                      28 non-null     datetime64[ns]
 22  EXPECTED TAKE OFF             28 non-null     datetime64[ns]
dtypes: datetime64[ns](23)
memory usage: 5.2+ KB
```

```
DATA.describe
```

```
<bound method NDFrame.describe of          SCHEDULED ARRIVAL  ...   EXPECTED TAKE OFF
FLIGHT                      ...
I52467 2021-05-08 11:12:00  ... 2021-05-08 11:33:00
UK812  2021-05-08 11:30:00  ... 2021-05-08 12:10:00
CX44   2021-05-08 11:45:00  ... 2021-05-08 12:25:00
6E2133 2021-05-08 12:02:00  ... 2021-05-08 12:40:00
AI565  2021-05-08 12:20:00  ... 2021-05-08 12:52:00
9I513  2021-05-08 01:00:00  ... 2021-05-08 01:40:00
6E435  2021-05-08 01:20:00  ... 2021-05-08 02:00:00
9I536  2021-05-08 01:45:00  ... 2021-05-08 02:15:00
9I898  2021-05-08 02:00:00  ... 2021-05-08 02:40:00
9I876  2021-05-08 02:20:00  ... 2021-05-08 02:45:00
I52990 2021-05-08 02:30:00  ... 2021-05-08 03:15:00
6E734  2021-05-08 02:50:00  ... 2021-05-08 03:30:00
6E379  2021-05-08 03:10:00  ... 2021-05-08 03:40:00
6E587  2021-05-08 03:30:00  ... 2021-05-08 04:05:00
SG706  2021-05-08 03:50:00  ... 2021-05-08 04:35:00
SG3008 2021-05-08 04:10:00  ... 2021-05-08 04:50:00
6E847  2021-05-08 04:30:00  ... 2021-05-08 05:00:00
G8834  2021-05-08 04:45:00  ... 2021-05-08 05:20:00
6E477  2021-05-08 05:00:00  ... 2021-05-08 05:45:00
SG7010 2021-05-08 05:20:00  ... 2021-05-08 06:00:00
6E332  2021-05-08 05:45:00  ... 2021-05-08 06:20:00
G8805  2021-05-08 06:10:00  ... 2021-05-08 06:40:00
6E6188 2021-05-08 06:30:00  ... 2021-05-08 07:10:00
AI503  2021-05-08 07:05:00  ... 2021-05-08 07:45:00
I51624 2021-05-08 07:20:00  ... 2021-05-08 08:00:00
G8406  2021-05-08 07:45:00  ... 2021-05-08 08:20:00
6E2487 2021-05-08 08:00:00  ... 2021-05-08 08:35:00
6E1456 2021-05-08 08:20:00  ... 2021-05-08 08:50:00

[28 rows x 23 columns]>
```

```
DATA.count()
```

```
SCHEDULED ARRIVAL                28
EXPECTED ARRIVAL                 28
CREW READY                       28
JET BRIDGE ON                    28
UNLOADING OF PASSENGERS START    28
UNLOADING OF PASSENGERS END      28
UNL BAGS                         28
UNL BAGS END                     28
REFUEL START                     28
REFUEL ENDS                      28
```

```
CLEANING STARTS                  28
CLEANING ENDS                    28
RESTOCK FOOD                     28
RESTOCK FOOD ENDS                28
RELOADING BAG STR                28
RELOADING BAGS END               28
RELOADING PASSENGERS             28
RELOADING PASSENGERS END         28
JET BRIDGE OFF                   28
DOORS CLOSED                     28
PUSH BACK                        28
TAKE OFF                         28
EXPECTED TAKE OFF                28
dtype: int64
```

```
DATA['SCHEDULED ARRIVAL'].mean()
```

```
Timestamp('2021-05-08 05:50:51.428571648')
```

```
df=pd.DataFrame(DATA, columns=['SCHEDULED ARRIVAL','EXPECTED ARRIVAL','CREW READY','JET BRIDGE ON','UNLOADING OF PASSENGERS START','UNLOADING OF PASSENGERS END','UNL BAGS','UNL BAGS E
```

```
df1=pd.DataFrame(DATA, columns=['EXPECTED ARRIVAL' , 'TAKE OFF' , 'FLIGHT'])
```

```
data2=pd.DataFrame(df1, columns=['EXPECTED ARRIVAL', 'TAKE OFF' , 'SCHEDULED TURNAROUND TIME'])
```

```
for item in df1:
  df1['SCHEDULED TURNAROUND TIME']= df1['TAKE OFF'] - df1['EXPECTED ARRIVAL']
print(df1)
```

```
          EXPECTED ARRIVAL  ... SCHEDULED TURNAROUND TIME
FLIGHT                      ...
I52467 2021-05-08 11:01:00  ...           0 days 00:34:00
UK812  2021-05-08 11:30:00  ...           0 days 00:50:00
CX44   2021-05-08 11:50:00  ...           0 days 00:37:00
6E2133 2021-05-08 12:05:00  ...           0 days 00:38:00
AI565  2021-05-08 12:18:00  ...           0 days 00:34:00
9I513  2021-05-08 01:00:00  ...           0 days 00:43:00
6E435  2021-05-08 01:27:00  ...           0 days 00:33:00
9I536  2021-05-08 01:40:00  ...           0 days 00:37:00
9I898  2021-05-08 02:08:00  ...           0 days 00:37:00
9I876  2021-05-08 02:15:00  ...           0 days 00:30:00
I52990 2021-05-08 02:40:00  ...           0 days 00:43:00
6E734  2021-05-08 02:50:00  ...           0 days 00:45:00
6E379  2021-05-08 03:10:00  ...           0 days 00:48:00
6E587  2021-05-08 03:30:00  ...           0 days 00:50:00
SG706  2021-05-08 03:50:00  ...           0 days 00:45:00
SG3008 2021-05-08 04:10:00  ...           0 days 00:41:00
6E847  2021-05-08 04:27:00  ...           0 days 00:38:00
G8834  2021-05-08 04:45:00  ...           0 days 00:35:00
6E477  2021-05-08 05:10:00  ...           0 days 00:38:00
SG7010 2021-05-08 05:20:00  ...           0 days 00:40:00
6E332  2021-05-08 05:45:00  ...           0 days 00:40:00
G8805  2021-05-08 06:10:00  ...           0 days 00:32:00
6E6188 2021-05-08 06:30:00  ...           0 days 00:40:00
AI503  2021-05-08 07:05:00  ...           0 days 00:40:00
I51624 2021-05-08 07:20:00  ...           0 days 00:40:00
```

```
G8406  2021-05-08 07:45:00  ...           0 days 00:45:00
6E2487 2021-05-08 08:00:00  ...           0 days 00:45:00
6E1456 2021-05-08 08:18:00  ...           0 days 00:46:00

[28 rows x 4 columns]
```

```
df1['SCHEDULED TURNAROUND TIME'].head(10)
```

```
FLIGHT
I52467   0 days 00:34:00
UK812    0 days 00:50:00
CX44     0 days 00:37:00
6E2133   0 days 00:38:00
AI565    0 days 00:34:00
9I513    0 days 00:43:00
6E435    0 days 00:33:00
9I536    0 days 00:37:00
9I898    0 days 00:37:00
9I876    0 days 00:30:00
Name: SCHEDULED TURNAROUND TIME, dtype: timedelta64[ns]
```

```
df1['SCHEDULED TURNAROUND TIME'].median()
```

```
Timedelta('0 days 00:40:00')
```

```
df1['SCHEDULED TURNAROUND TIME'].mean() #insight 1 : the average turnaround time for an airbus aircraft is 40 minutes.
```

```
Timedelta('0 days 00:40:08.571428571')
```

```
df1['SCHEDULED TURNAROUND TIME'].mode()
```

```
0   0 days 00:40:00
dtype: timedelta64[ns]
```

```
df1.head()
```

	EXPECTED ARRIVAL	TAKE OFF	FLIGHT	SCHEDULED TURNAROUND TIME
FLIGHT				
I52467	2021-05-08 11:01:00	2021-05-08 11:35:00	NaN	0 days 00:34:00
UK812	2021-05-08 11:30:00	2021-05-08 12:20:00	NaN	0 days 00:50:00
CX44	2021-05-08 11:50:00	2021-05-08 12:27:00	NaN	0 days 00:37:00
6E2133	2021-05-08 12:05:00	2021-05-08 12:43:00	NaN	0 days 00:38:00
AI565	2021-05-08 12:18:00	2021-05-08 12:52:00	NaN	0 days 00:34:00

```
df1
```

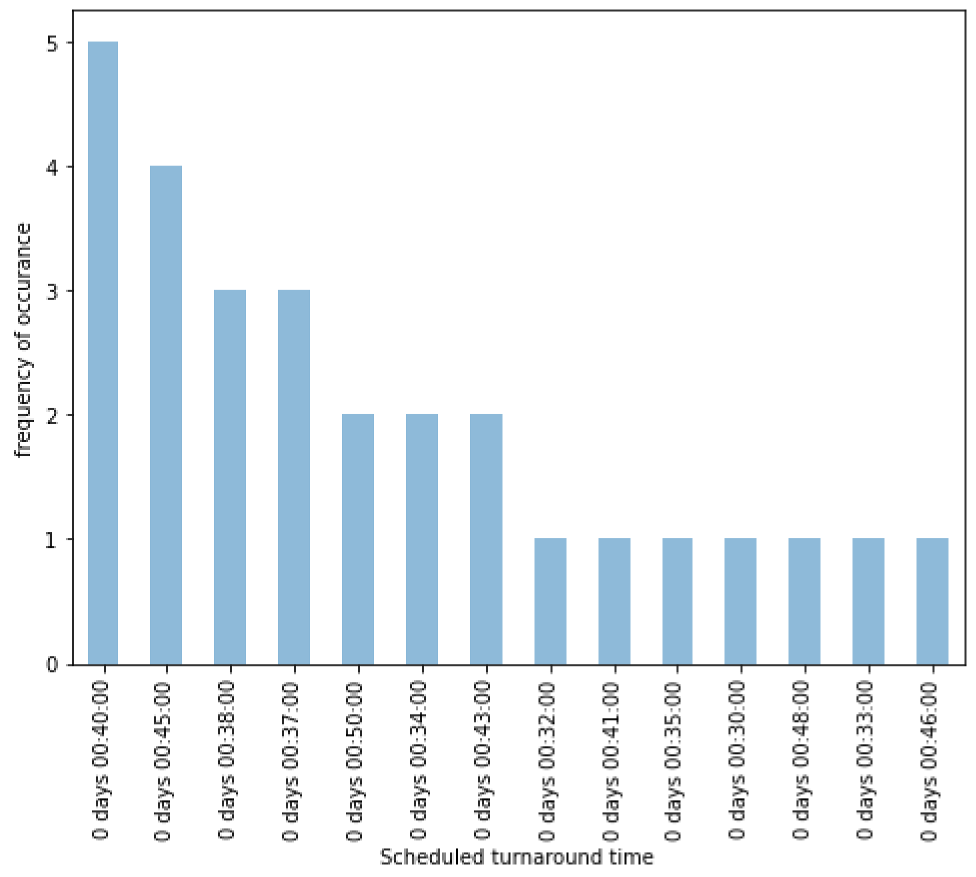
	EXPECTED ARRIVAL	TAKE OFF	FLIGHT	SCHEDULED TURNAROUND TIME
FLIGHT				
I52467	2021-05-08 11:01:00	2021-05-08 11:35:00	NaN	0 days 00:34:00
UK812	2021-05-08 11:30:00	2021-05-08 12:20:00	NaN	0 days 00:50:00
CX44	2021-05-08 11:50:00	2021-05-08 12:27:00	NaN	0 days 00:37:00
6E2133	2021-05-08 12:05:00	2021-05-08 12:43:00	NaN	0 days 00:38:00
AI565	2021-05-08 12:18:00	2021-05-08 12:52:00	NaN	0 days 00:34:00
9I513	2021-05-08 01:00:00	2021-05-08 01:43:00	NaN	0 days 00:43:00
6E435	2021-05-08 01:27:00	2021-05-08 02:00:00	NaN	0 days 00:33:00
9I536	2021-05-08 01:40:00	2021-05-08 02:17:00	NaN	0 days 00:37:00
9I898	2021-05-08 02:08:00	2021-05-08 02:45:00	NaN	0 days 00:37:00
9I876	2021-05-08 02:15:00	2021-05-08 02:45:00	NaN	0 days 00:30:00
I52990	2021-05-08 02:40:00	2021-05-08 03:23:00	NaN	0 days 00:43:00
6E734	2021-05-08 02:50:00	2021-05-08 03:35:00	NaN	0 days 00:45:00
6E379	2021-05-08 03:10:00	2021-05-08 03:58:00	NaN	0 days 00:48:00
6E587	2021-05-08 03:30:00	2021-05-08 04:20:00	NaN	0 days 00:50:00
SG706	2021-05-08 03:50:00	2021-05-08 04:35:00	NaN	0 days 00:45:00
SG3008	2021-05-08 04:10:00	2021-05-08 04:51:00	NaN	0 days 00:41:00
6E847	2021-05-08 04:27:00	2021-05-08 05:05:00	NaN	0 days 00:38:00
G8834	2021-05-08 04:45:00	2021-05-08 05:20:00	NaN	0 days 00:35:00
6E477	2021-05-08 05:10:00	2021-05-08 05:48:00	NaN	0 days 00:38:00
SG7010	2021-05-08 05:20:00	2021-05-08 06:00:00	NaN	0 days 00:40:00
6E332	2021-05-08 05:45:00	2021-05-08 06:25:00	NaN	0 days 00:40:00
G8805	2021-05-08 06:10:00	2021-05-08 06:42:00	NaN	0 days 00:32:00
6E6188	2021-05-08 06:30:00	2021-05-08 07:10:00	NaN	0 days 00:40:00
AI503	2021-05-08 07:05:00	2021-05-08 07:45:00	NaN	0 days 00:40:00
I51624	2021-05-08 07:20:00	2021-05-08 08:00:00	NaN	0 days 00:40:00
G8406	2021-05-08 07:45:00	2021-05-08 08:30:00	NaN	0 days 00:45:00
6E2487	2021-05-08 08:00:00	2021-05-08 08:45:00	NaN	0 days 00:45:00

```
my_tab = pd.crosstab(index=df1["SCHEDULED TURNAROUND TIME"],  # Make a crosstab
                     columns="count")                  # Name the count column

my_tab
```

col_0	count
SCHEDULED TURNAROUND TIME	
0 days 00:30:00	1
0 days 00:32:00	1
0 days 00:33:00	1
0 days 00:34:00	2
0 days 00:35:00	1
0 days 00:37:00	3
0 days 00:38:00	3
0 days 00:40:00	5
0 days 00:41:00	1
0 days 00:43:00	2
0 days 00:45:00	4
0 days 00:46:00	1

```
plt.figure(figsize=(8,6))
df1["SCHEDULED TURNAROUND TIME"].value_counts().plot(kind='bar',alpha=0.5)
plt.xlabel('Scheduled turnaround time')
plt.ylabel('frequency of occurance')
plt.show()
```



```
# insight 2 : 5 out of 28 aircrafts took 40 minutes 4 out of 28 aircrafts took 45 minutes and 6 aircrafts took more than 37 minutes for the turnaround process.
# 14 out of 28 aircrafts, i.e 50% of the aircrafts took more than 40 minutes but less than 50 minutes, 2 aircrafts took 50 minutes i.e 7%.
# the ideal time for the turnaround process is considered anywhere between 30 to 40 minutes and 12 out of 28 aircrafts i.e 42% of the aircrafts fall under this category.
```

```
df2=pd.DataFrame(DATA, columns=[ "UNLOADING OF PASSENGERS START" , "UNLOADING OF PASSENGERS END" , "FLIGHT"])
```

```
for item in df2:
  df2['critical path 1']= df2['UNLOADING OF PASSENGERS END'] - df2['UNLOADING OF PASSENGERS START']
print(df2)
```

```
           UNLOADING OF PASSENGERS START  ... critical path 1
FLIGHT                                    ...
I52467           2021-05-08 11:04:00  ... 0 days 00:09:00
UK812            2021-05-08 11:34:00  ... 0 days 00:11:00
CX44             2021-05-08 11:55:00  ... 0 days 00:12:00
6E2133           2021-05-08 12:09:00  ... 0 days 00:12:00
AI565            2021-05-08 12:23:00  ... 0 days 00:07:00
9I513            2021-05-08 01:08:00  ... 0 days 00:12:00
6E435            2021-05-08 01:32:00  ... 0 days 00:10:00
9I536            2021-05-08 01:45:00  ... 0 days 00:07:00
9I898            2021-05-08 02:13:00  ... 0 days 00:07:00
9I876            2021-05-08 02:20:00  ... 0 days 00:10:00
I52990           2021-05-08 02:45:00  ... 0 days 00:10:00
6E734            2021-05-08 02:56:00  ... 0 days 00:11:00
6E379            2021-05-08 03:15:00  ... 0 days 00:09:00
6E587            2021-05-08 03:35:00  ... 0 days 00:11:00
SG706            2021-05-08 03:54:00  ... 0 days 00:10:00
SG3008           2021-05-08 04:15:00  ... 0 days 00:08:00
6E847            2021-05-08 04:30:00  ... 0 days 00:10:00
G8834            2021-05-08 04:49:00  ... 0 days 00:11:00
6E477            2021-05-08 05:15:00  ... 0 days 00:07:00
SG7010           2021-05-08 05:25:00  ... 0 days 00:09:00
6E332            2021-05-08 05:49:00  ... 0 days 00:11:00
G8805            2021-05-08 06:14:00  ... 0 days 00:08:00
6E6188           2021-05-08 06:35:00  ... 0 days 00:08:00
AI503            2021-05-08 07:10:00  ... 0 days 00:10:00
I51624           2021-05-08 07:25:00  ... 0 days 00:09:00
G8406            2021-05-08 07:50:00  ... 0 days 00:10:00
6E2487           2021-05-08 08:04:00  ... 0 days 00:11:00
6E1456           2021-05-08 08:23:00  ... 0 days 00:10:00

[28 rows x 4 columns]
```

```
df2['critical path 1'].mean()
```

```
Timedelta('0 days 00:09:38.571428571')
```

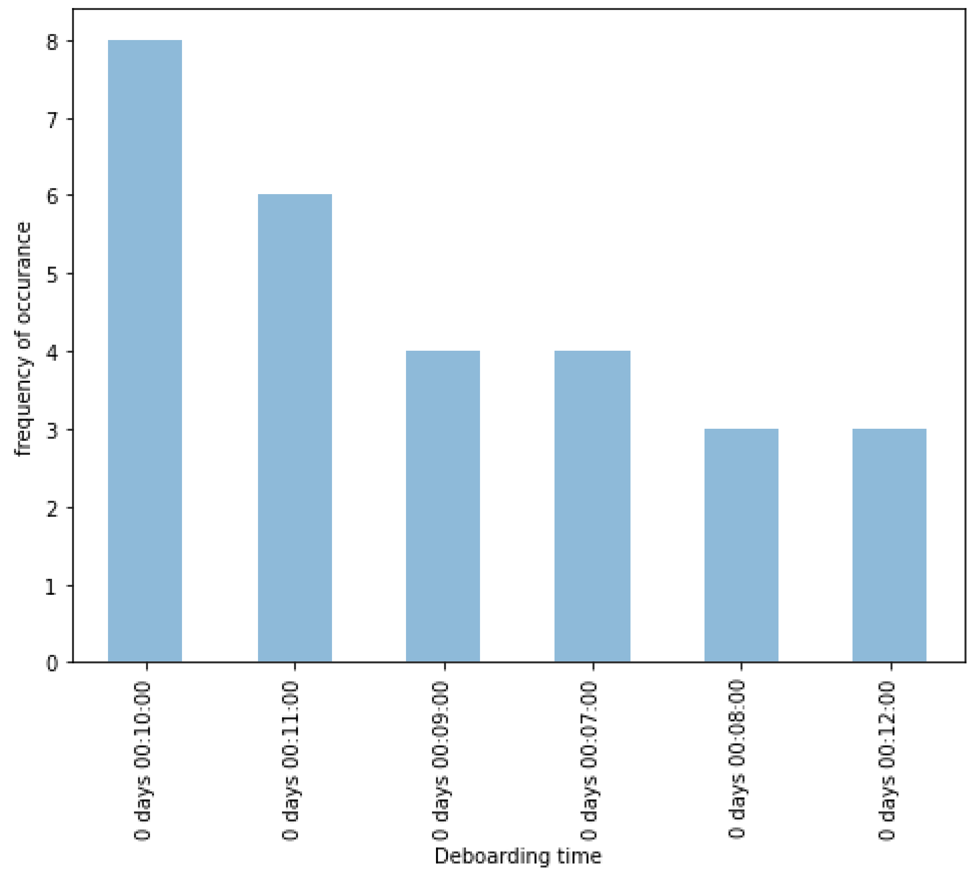
```
df2['critical path 1'].median()
```

```
Timedelta('0 days 00:10:00')
```

```
df2['critical path 1'].mode()
```

```
0   0 days 00:10:00
dtype: timedelta64[ns]
```

```
plt.figure(figsize=(8,6))
df2["critical path 1"].value_counts().plot(kind='bar',alpha=0.5)
plt.xlabel('Deboarding time')
plt.ylabel('frequency of occurance')
plt.show()
```



```
#insight 3 : deboarding being the critical path 1, takes a mean time of 9 minutes for the completion of the process.
#8 out of 28 took 10 minutes (28%), 6 out of 28 took 11 minutes (21%), 3 out of 28 took 12 minutes (9%), only 11 out of 28 took 9 or less than 9 minutes (40%)
```

```
df3=pd.DataFrame(DATA, columns=["REFUEL START" , "REFUEL ENDS" , "FLIGHT"])
```

```
for item in df3:
  df3['critical path 2']= df3['REFUEL ENDS'] - df3['REFUEL START']
print(df3)
```

```
               REFUEL START         REFUEL ENDS  FLIGHT critical path 2
FLIGHT
I52467 2021-05-08 11:05:00 2021-05-08 11:16:00     NaN 0 days 00:11:00
UK812  2021-05-08 11:36:00 2021-05-08 11:50:00     NaN 0 days 00:14:00
CX44   2021-05-08 11:57:00 2021-05-08 12:11:00     NaN 0 days 00:14:00
6E2133 2021-05-08 12:10:00 2021-05-08 12:25:00     NaN 0 days 00:15:00
AI565  2021-05-08 12:22:00 2021-05-08 12:37:00     NaN 0 days 00:15:00
9I513  2021-05-08 01:09:00 2021-05-08 01:26:00     NaN 0 days 00:17:00
6E435  2021-05-08 01:34:00 2021-05-08 01:50:00     NaN 0 days 00:16:00
9I536  2021-05-08 01:45:00 2021-05-08 02:00:00     NaN 0 days 00:15:00
```

```
9I898  2021-05-08 02:12:00 2021-05-08 02:28:00     NaN 0 days 00:16:00
9I876  2021-05-08 02:18:00 2021-05-08 02:33:00     NaN 0 days 00:15:00
I52990 2021-05-08 02:44:00 2021-05-08 03:00:00     NaN 0 days 00:16:00
6E734  2021-05-08 02:55:00 2021-05-08 03:11:00     NaN 0 days 00:16:00
6E379  2021-05-08 03:15:00 2021-05-08 03:31:00     NaN 0 days 00:16:00
6E587  2021-05-08 03:37:00 2021-05-08 03:53:00     NaN 0 days 00:16:00
SG706  2021-05-08 03:55:00 2021-05-08 04:12:00     NaN 0 days 00:17:00
SG3008 2021-05-08 04:15:00 2021-05-08 04:35:00     NaN 0 days 00:20:00
6E847  2021-05-08 04:30:00 2021-05-08 04:46:00     NaN 0 days 00:16:00
G8834  2021-05-08 04:50:00 2021-05-08 05:08:00     NaN 0 days 00:18:00
6E477  2021-05-08 05:15:00 2021-05-08 05:32:00     NaN 0 days 00:17:00
SG7010 2021-05-08 05:25:00 2021-05-08 05:40:00     NaN 0 days 00:15:00
6E332  2021-05-08 05:55:00 2021-05-08 06:12:00     NaN 0 days 00:17:00
G8805  2021-05-08 06:15:00 2021-05-08 06:33:00     NaN 0 days 00:18:00
6E6188 2021-05-08 06:35:00 2021-05-08 06:52:00     NaN 0 days 00:17:00
AI503  2021-05-08 07:08:00 2021-05-08 07:23:00     NaN 0 days 00:15:00
I51624 2021-05-08 07:30:00 2021-05-08 07:49:00     NaN 0 days 00:19:00
G8406  2021-05-08 07:50:00 2021-05-08 08:07:00     NaN 0 days 00:17:00
6E2487 2021-05-08 08:04:00 2021-05-08 08:20:00     NaN 0 days 00:16:00
6E1456 2021-05-08 08:25:00 2021-05-08 08:42:00     NaN 0 days 00:17:00
```

```
df3['critical path 2'].mean()
```

```
Timedelta('0 days 00:16:06.428571428')
```

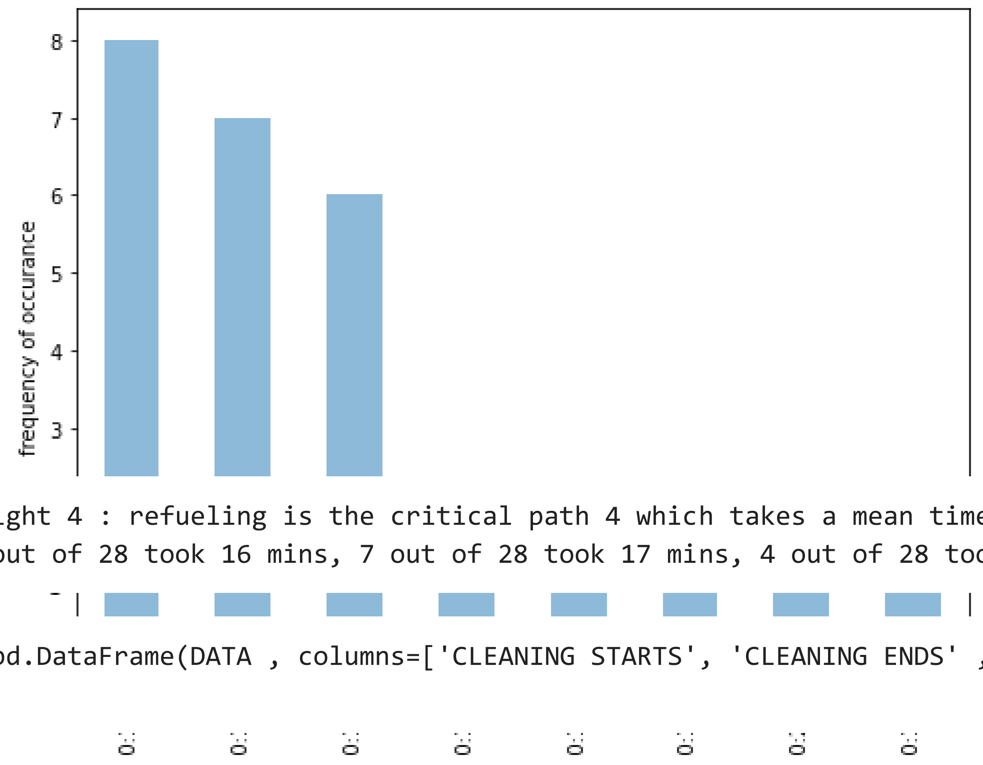
```
df3['critical path 2'].median()
```

```
Timedelta('0 days 00:16:00')
```

```
df3['critical path 2'].mode()
```

```
0   0 days 00:16:00
dtype: timedelta64[ns]
```

```
plt.figure(figsize=(8,6))
df3["critical path 2"].value_counts().plot(kind='bar',alpha=0.5)
plt.xlabel('fueling time')
plt.ylabel('frequency of occurance')
plt.show()
```



```
#insight 4 : refueling is the critical path 4 which takes a mean time of 16 minutes.
# 8 out of 28 took 16 mins, 7 out of 28 took 17 mins, 4 out of 28 took more than 18 minutes. 17 out of 28 took 16 minutes which r on time.
```

```
df4=pd.DataFrame(DATA , columns=['CLEANING STARTS', 'CLEANING ENDS' , 'FLIGHT'])
```

```
for item in df4:
  df4['critical path 3']= df4['CLEANING ENDS'] - df4['CLEANING STARTS']
print(df4)
```

```
                CLEANING STARTS       CLEANING ENDS  FLIGHT critical path 3
FLIGHT
I52467 2021-05-08 11:14:00 2021-05-08 11:20:00     NaN 0 days 00:06:00
UK812  2021-05-08 11:47:00 2021-05-08 11:53:00     NaN 0 days 00:06:00
CX44   2021-05-08 12:02:00 2021-05-08 12:09:00     NaN 0 days 00:07:00
6E2133 2021-05-08 12:18:00 2021-05-08 12:25:00     NaN 0 days 00:07:00
AI565  2021-05-08 12:31:00 2021-05-08 12:37:00     NaN 0 days 00:06:00
9I513  2021-05-08 01:22:00 2021-05-08 01:28:00     NaN 0 days 00:06:00
6E435  2021-05-08 01:43:00 2021-05-08 01:50:00     NaN 0 days 00:07:00
9I536  2021-05-08 01:53:00 2021-05-08 02:00:00     NaN 0 days 00:07:00
9I898  2021-05-08 02:20:00 2021-05-08 02:26:00     NaN 0 days 00:06:00
9I876  2021-05-08 02:22:00 2021-05-08 02:28:00     NaN 0 days 00:06:00
I52990 2021-05-08 02:58:00 2021-05-08 03:04:00     NaN 0 days 00:06:00
6E734  2021-05-08 03:10:00 2021-05-08 03:18:00     NaN 0 days 00:08:00
6E379  2021-05-08 03:27:00 2021-05-08 03:37:00     NaN 0 days 00:10:00
6E587  2021-05-08 03:49:00 2021-05-08 03:57:00     NaN 0 days 00:08:00
SG706  2021-05-08 04:06:00 2021-05-08 04:13:00     NaN 0 days 00:07:00
SG3008 2021-05-08 04:25:00 2021-05-08 04:32:00     NaN 0 days 00:07:00
6E847  2021-05-08 04:42:00 2021-05-08 04:49:00     NaN 0 days 00:07:00
G8834  2021-05-08 04:55:00 2021-05-08 05:00:00     NaN 0 days 00:05:00
6E477  2021-05-08 05:25:00 2021-05-08 05:32:00     NaN 0 days 00:07:00
SG7010 2021-05-08 05:36:00 2021-05-08 05:43:00     NaN 0 days 00:07:00
6E332  2021-05-08 06:02:00 2021-05-08 06:10:00     NaN 0 days 00:08:00
G8805  2021-05-08 06:23:00 2021-05-08 06:30:00     NaN 0 days 00:07:00
6E6188 2021-05-08 06:45:00 2021-05-08 06:53:00     NaN 0 days 00:08:00
AI503  2021-05-08 07:22:00 2021-05-08 07:28:00     NaN 0 days 00:06:00
I51624 2021-05-08 07:35:00 2021-05-08 07:43:00     NaN 0 days 00:08:00
G8406  2021-05-08 08:01:00 2021-05-08 08:09:00     NaN 0 days 00:08:00
6E2487 2021-05-08 08:17:00 2021-05-08 08:26:00     NaN 0 days 00:09:00
6E1456 2021-05-08 08:35:00 2021-05-08 08:44:00     NaN 0 days 00:09:00
```

```
df4['critical path 3'].mean()
```

```
    Timedelta('0 days 00:07:06.428571428')
```

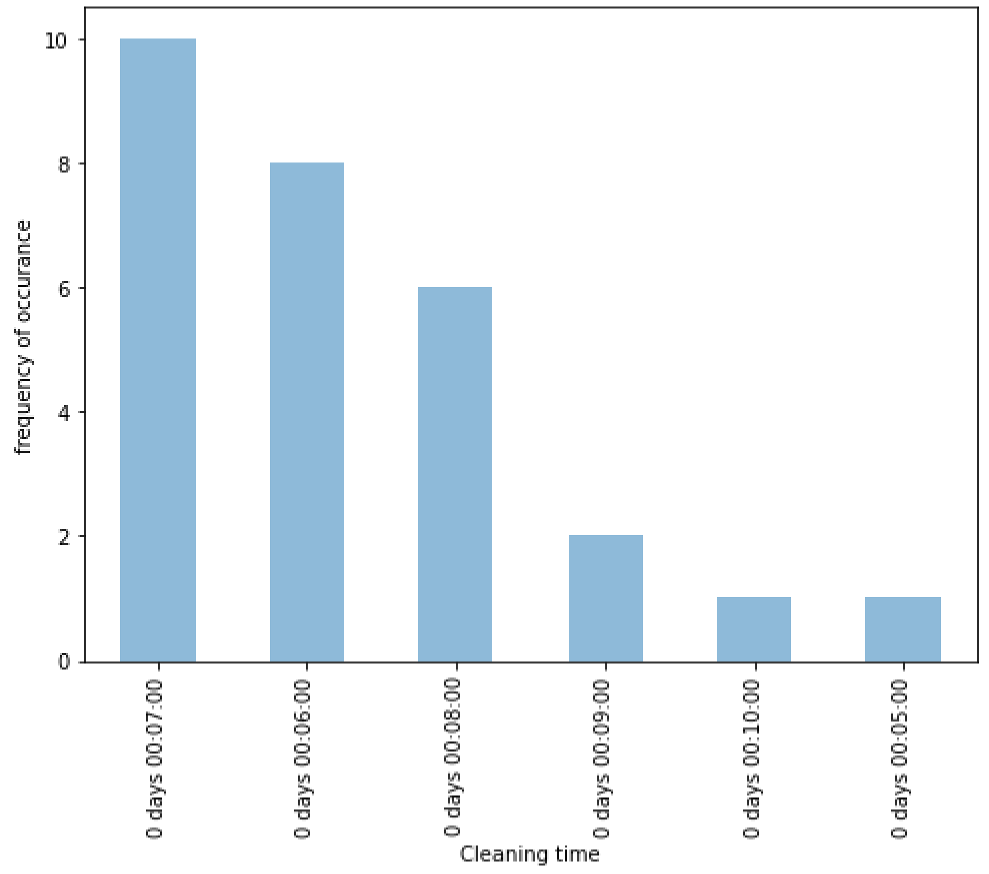
```
df4['critical path 3'].median()
```

```
    Timedelta('0 days 00:07:00')
```

```
df4['critical path 3'].mode()
```

```
    0   0 days 00:07:00
    dtype: timedelta64[ns]
```

```
plt.figure(figsize=(8,6))
df4["critical path 3"].value_counts().plot(kind='bar',alpha=0.5)
plt.xlabel('Cleaning time')
plt.ylabel('frequency of occurance')
plt.show()
```



```
#insight 5: the mean time for cleaning was 7 minutes.
#10 out of 28 took 7 mins, 8 out of 28 took 6 mins 9 out of 28 took more than 8 mins. 19 out of 28 took 7 or lesser time.
```

```
df5=pd.DataFrame(DATA, columns=['JET BRIDGE ON' , 'JET BRIDGE OFF' , 'FLIGHT'])
```

```
for item in df5:
  df5['critical path 4']= df5['JET BRIDGE OFF'] - df5['JET BRIDGE ON']
print(df5)
```

```
                 JET BRIDGE ON      JET BRIDGE OFF  FLIGHT critical path 4
FLIGHT
I52467 2021-05-08 11:03:00 2021-05-08 11:28:00     NaN 0 days 00:25:00
UK812  2021-05-08 11:33:00 2021-05-08 12:10:00     NaN 0 days 00:37:00
CX44   2021-05-08 11:53:00 2021-05-08 12:20:00     NaN 0 days 00:27:00
6E2133 2021-05-08 12:07:00 2021-05-08 12:36:00     NaN 0 days 00:29:00
AI565  2021-05-08 12:21:00 2021-05-08 12:46:00     NaN 0 days 00:25:00
9I513  2021-05-08 01:05:00 2021-05-08 01:37:00     NaN 0 days 00:32:00
6E435  2021-05-08 01:30:00 2021-05-08 01:54:00     NaN 0 days 00:24:00
9I536  2021-05-08 01:43:00 2021-05-08 02:08:00     NaN 0 days 00:25:00
9I898  2021-05-08 02:10:00 2021-05-08 02:40:00     NaN 0 days 00:30:00
9I876  2021-05-08 02:18:00 2021-05-08 02:40:00     NaN 0 days 00:22:00
I52990 2021-05-08 02:43:00 2021-05-08 03:16:00     NaN 0 days 00:33:00
6E734  2021-05-08 02:52:00 2021-05-08 03:30:00     NaN 0 days 00:38:00
6E379  2021-05-08 03:12:00 2021-05-08 03:52:00     NaN 0 days 00:40:00
6E587  2021-05-08 03:32:00 2021-05-08 04:14:00     NaN 0 days 00:42:00
SG706  2021-05-08 03:52:00 2021-05-08 04:26:00     NaN 0 days 00:34:00
SG3008 2021-05-08 04:13:00 2021-05-08 04:45:00     NaN 0 days 00:32:00
6E847  2021-05-08 04:28:00 2021-05-08 05:01:00     NaN 0 days 00:33:00
G8834  2021-05-08 04:47:00 2021-05-08 05:15:00     NaN 0 days 00:28:00
6E477  2021-05-08 05:12:00 2021-05-08 05:45:00     NaN 0 days 00:33:00
SG7010 2021-05-08 05:23:00 2021-05-08 05:57:00     NaN 0 days 00:34:00
6E332  2021-05-08 05:47:00 2021-05-08 06:21:00     NaN 0 days 00:34:00
G8805  2021-05-08 06:12:00 2021-05-08 06:40:00     NaN 0 days 00:28:00
6E6188 2021-05-08 06:33:00 2021-05-08 07:05:00     NaN 0 days 00:32:00
AI503  2021-05-08 07:08:00 2021-05-08 07:41:00     NaN 0 days 00:33:00
I51624 2021-05-08 07:22:00 2021-05-08 07:56:00     NaN 0 days 00:34:00
G8406  2021-05-08 07:47:00 2021-05-08 08:24:00     NaN 0 days 00:37:00
6E2487 2021-05-08 08:02:00 2021-05-08 08:37:00     NaN 0 days 00:35:00
6E1456 2021-05-08 08:20:00 2021-05-08 08:58:00     NaN 0 days 00:38:00
```

```
df5['critical path 4'].mean()
```

```
Timedelta('0 days 00:31:55.714285714')
```

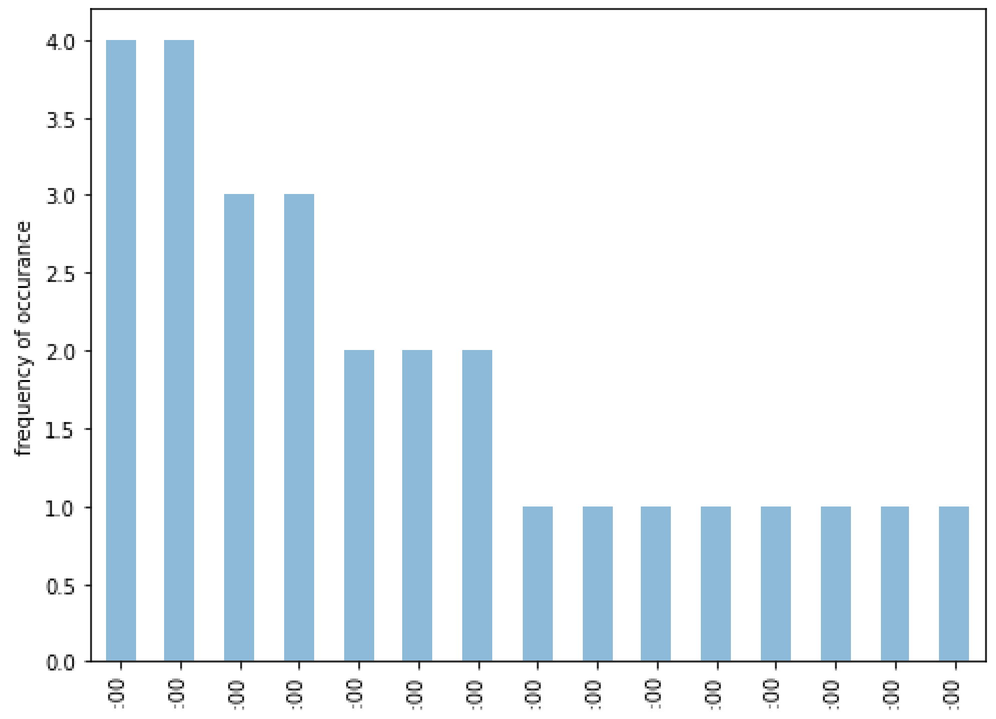
```
df5['critical path 4'].median()
```

```
Timedelta('0 days 00:33:00')
```

```
df5['critical path 4'].mode()
```

```
0   0 days 00:33:00
1   0 days 00:34:00
dtype: timedelta64[ns]
```

```
plt.figure(figsize=(8,6))
df5["critical path 4"].value_counts().plot(kind='bar',alpha=0.5)
plt.xlabel('Passenger bridge time')
plt.ylabel('frequency of occurance')
plt.show()
```



```
# the mean time for jet bridge is 32 minutes and 13 out of 28 aircrafts are on or below 32 minute mark and 4 are on 33 minute mark which isn't considered as delay.
#majority of the delay time is caused by the jet bridge as the time interval between the starting point and the ending point depends on various factors and there isn't much of data tc
# 11 out of 28 take more than 33 minutes.
```

```
df6=pd.DataFrame(DATA, columns=['RELOADING PASSENGERS', 'RELOADING PASSENGERS END', 'FLIGHT'])


for item in df6:
  df6['critical path 5']= df6['RELOADING PASSENGERS END'] - df6['RELOADING PASSENGERS']
print(df6)
```

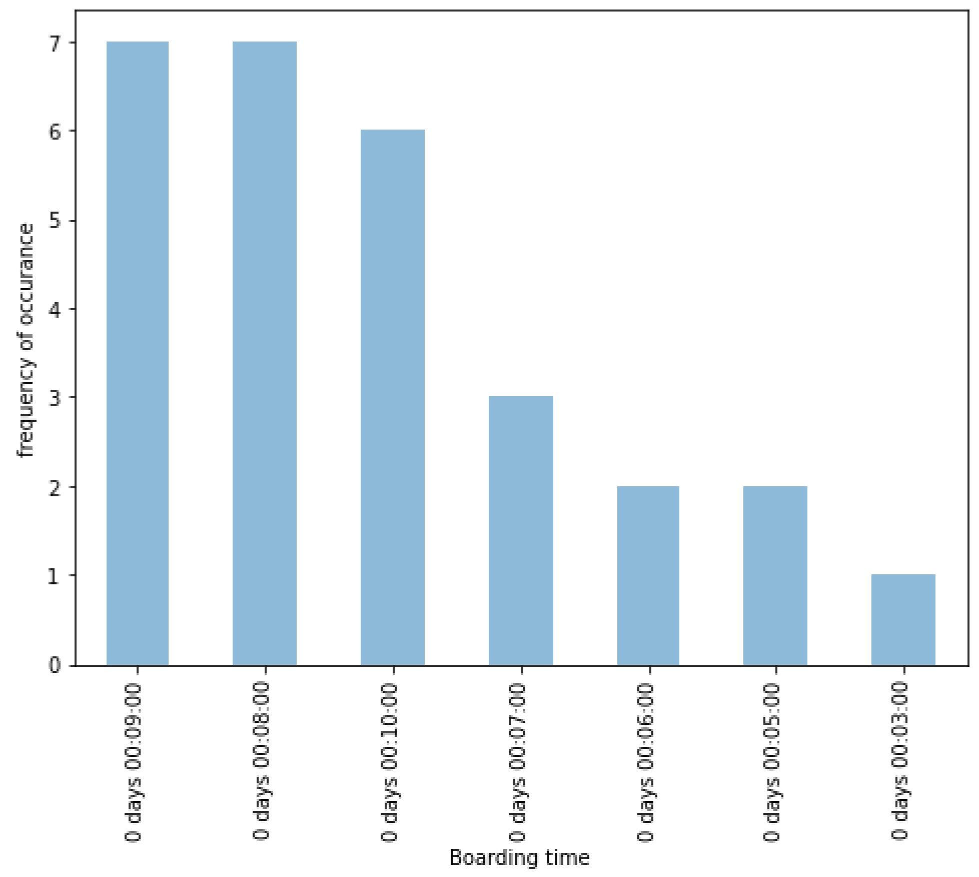
```
            RELOADING PASSENGERS RELOADING PASSENGERS END  FLIGHT critical path 5
     FLIGHT
     I52467  2021-05-08 11:23:00      2021-05-08 11:26:00     NaN 0 days 00:03:00
     UK812   2021-05-08 11:58:00      2021-05-08 12:07:00     NaN 0 days 00:09:00
     CX44    2021-05-08 12:13:00      2021-05-08 12:18:00     NaN 0 days 00:05:00
     6E2133  2021-05-08 12:28:00      2021-05-08 12:34:00     NaN 0 days 00:06:00
     AI565   2021-05-08 12:39:00      2021-05-08 12:45:00     NaN 0 days 00:06:00
     9I513   2021-05-08 01:30:00      2021-05-08 01:35:00     NaN 0 days 00:05:00
     6E435   2021-05-08 01:44:00      2021-05-08 01:52:00     NaN 0 days 00:08:00
     9I536   2021-05-08 01:56:00      2021-05-08 02:04:00     NaN 0 days 00:08:00
     9I898   2021-05-08 02:28:00      2021-05-08 02:38:00     NaN 0 days 00:10:00
     9I876   2021-05-08 02:30:00      2021-05-08 02:38:00     NaN 0 days 00:08:00
     I52990  2021-05-08 03:06:00      2021-05-08 03:14:00     NaN 0 days 00:08:00
     6E734   2021-05-08 03:20:00      2021-05-08 03:28:00     NaN 0 days 00:08:00
     6E379   2021-05-08 03:40:00      2021-05-08 03:50:00     NaN 0 days 00:10:00
     6E587   2021-05-08 04:00:00      2021-05-08 04:10:00     NaN 0 days 00:10:00
     SG706   2021-05-08 04:15:00      2021-05-08 04:24:00     NaN 0 days 00:09:00
     SG3008  2021-05-08 04:34:00      2021-05-08 04:44:00     NaN 0 days 00:10:00
     6E847   2021-05-08 04:53:00      2021-05-08 05:00:00     NaN 0 days 00:07:00
     G8834   2021-05-08 05:03:00      2021-05-08 05:12:00     NaN 0 days 00:09:00
     6E477   2021-05-08 05:34:00      2021-05-08 05:43:00     NaN 0 days 00:09:00
     SG7010  2021-05-08 05:45:00      2021-05-08 05:55:00     NaN 0 days 00:10:00
     6E332   2021-05-08 06:12:00      2021-05-08 06:20:00     NaN 0 days 00:08:00
     G8805   2021-05-08 06:32:00      2021-05-08 06:39:00     NaN 0 days 00:07:00
     6E6188  2021-05-08 06:55:00      2021-05-08 07:03:00     NaN 0 days 00:08:00
     AI503   2021-05-08 07:30:00      2021-05-08 07:39:00     NaN 0 days 00:09:00
```

```
I51624  2021-05-08 07:45:00      2021-05-08 07:54:00     NaN 0 days 00:09:00
G8406   2021-05-08 08:11:00      2021-05-08 08:20:00     NaN 0 days 00:09:00
6E2487  2021-05-08 08:28:00      2021-05-08 08:35:00     NaN 0 days 00:07:00
6E1456  2021-05-08 08:46:00      2021-05-08 08:56:00     NaN 0 days 00:10:00
```

```
df6['critical path 5'].mean()
```

```
Timedelta('0 days 00:08:02.142857142')
```

```
plt.figure(figsize=(8,6))
df6["critical path 5"].value_counts().plot(kind='bar',alpha=0.5)
plt.xlabel('Boarding time')
plt.ylabel('frequency of occurance')
plt.show()
```



```
#boarding is directly co-related to the cleaning process so sooner the cleaning process lesser the delay time for boarding time.
#the boarding mean time is 8 to 9 minutes.
#14 aircrafts take 8 to 9 minutes for boarding. 6 take 10 minutes and 8 take less than 8 minutes.
```

```
df7=pd.DataFrame(DATA, columns=['RESTOCK FOOD', 'RESTOCK FOOD ENDS', 'FLIGHT'])
```

```
for item in df7:
  df7['critical path 6']= df7['RESTOCK FOOD ENDS'] - df7['RESTOCK FOOD']
print(df7)
```

```
                   RESTOCK FOOD   RESTOCK FOOD ENDS  FLIGHT critical path 6
FLIGHT
I52467 2021-05-08 11:08:00 2021-05-08 11:12:00     NaN 0 days 00:04:00
```

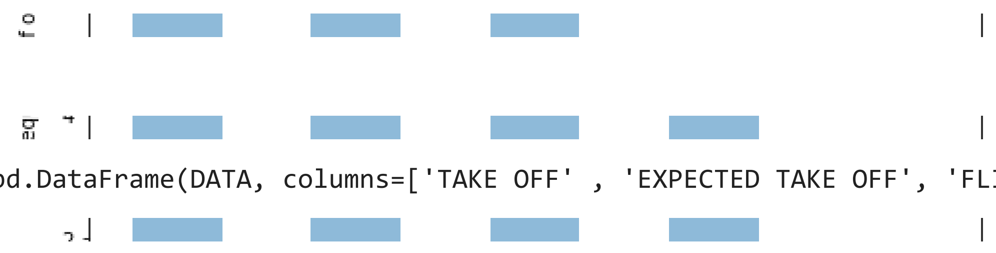
```
     UK812  2021-05-08 11:37:00 2021-05-08 11:43:00     NaN 0 days 00:06:00
     CX44   2021-05-08 11:58:00 2021-05-08 12:03:00     NaN 0 days 00:05:00
     6E2133 2021-05-08 12:12:00 2021-05-08 12:16:00     NaN 0 days 00:04:00
     AI565  2021-05-08 12:26:00 2021-05-08 12:33:00     NaN 0 days 00:07:00
     9I513  2021-05-08 01:10:00 2021-05-08 01:16:00     NaN 0 days 00:06:00
     6E435  2021-05-08 01:34:00 2021-05-08 01:40:00     NaN 0 days 00:06:00
     9I536  2021-05-08 01:48:00 2021-05-08 01:55:00     NaN 0 days 00:07:00
     9I898  2021-05-08 02:18:00 2021-05-08 02:24:00     NaN 0 days 00:06:00
     9I876  2021-05-08 02:23:00 2021-05-08 02:27:00     NaN 0 days 00:04:00
     I52990 2021-05-08 02:54:00 2021-05-08 03:00:00     NaN 0 days 00:06:00
     6E734  2021-05-08 03:00:00 2021-05-08 03:04:00     NaN 0 days 00:04:00
     6E379  2021-05-08 03:16:00 2021-05-08 03:23:00     NaN 0 days 00:07:00
     6E587  2021-05-08 03:37:00 2021-05-08 03:43:00     NaN 0 days 00:06:00
     SG706  2021-05-08 04:00:00 2021-05-08 04:05:00     NaN 0 days 00:05:00
     SG3008 2021-05-08 04:20:00 2021-05-08 04:25:00     NaN 0 days 00:05:00
     6E847  2021-05-08 04:43:00 2021-05-08 04:48:00     NaN 0 days 00:05:00
     G8834  2021-05-08 04:43:00 2021-05-08 04:48:00     NaN 0 days 00:05:00
     6E477  2021-05-08 05:23:00 2021-05-08 05:29:00     NaN 0 days 00:06:00
     SG7010 2021-05-08 05:32:00 2021-05-08 05:39:00     NaN 0 days 00:07:00
     6E332  2021-05-08 05:55:00 2021-05-08 06:00:00     NaN 0 days 00:05:00
     G8805  2021-05-08 06:17:00 2021-05-08 06:23:00     NaN 0 days 00:06:00
     6E6188 2021-05-08 06:45:00 2021-05-08 06:53:00     NaN 0 days 00:08:00
     AI503  2021-05-08 07:23:00 2021-05-08 07:30:00     NaN 0 days 00:07:00
     I51624 2021-05-08 07:38:00 2021-05-08 07:44:00     NaN 0 days 00:06:00
     G8406  2021-05-08 08:00:00 2021-05-08 08:07:00     NaN 0 days 00:07:00
     6E2487 2021-05-08 08:20:00 2021-05-08 08:26:00     NaN 0 days 00:06:00
     6E1456 2021-05-08 08:33:00 2021-05-08 08:40:00     NaN 0 days 00:07:00
```

```
df7['critical path 6'].mean()
```

```
     Timedelta('0 days 00:05:49.285714285')
```

```
plt.figure(figsize=(8,6))
df7["critical path 6"].value_counts().plot(kind='bar',alpha=0.5)
plt.xlabel('CATERING time')
plt.ylabel('frequency of occurance')
plt.show()
```

```
#restocking of food mean time is 5 to 6 minutes.
#20 aircrafts take 6 minutes or lesser time for restocking.
```

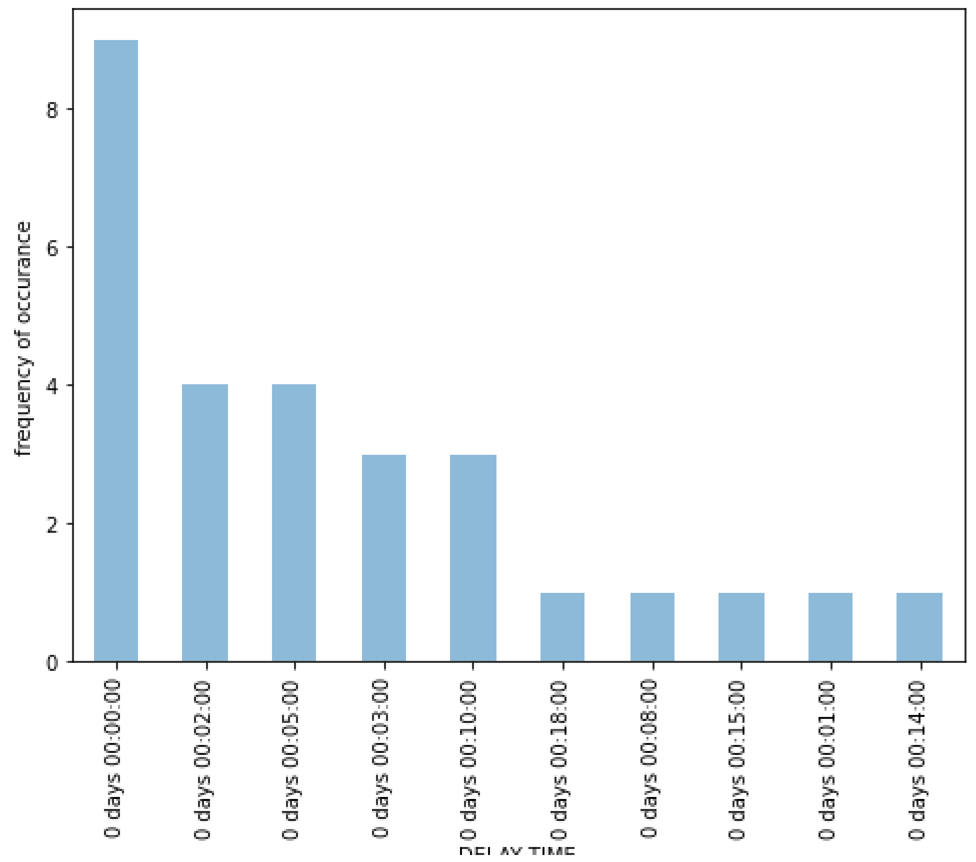


```
df8=pd.DataFrame(DATA, columns=['TAKE OFF' , 'EXPECTED TAKE OFF', 'FLIGHT'])
```

```
for item in df8:
  df8['DELAY TIME']= df8['TAKE OFF'] - df8['EXPECTED TAKE OFF']
print(df8)
```

```
                   TAKE OFF   EXPECTED TAKE OFF  FLIGHT      DELAY TIME
FLIGHT
I52467 2021-05-08 11:35:00 2021-05-08 11:33:00     NaN 0 days 00:02:00
UK812  2021-05-08 12:20:00 2021-05-08 12:10:00     NaN 0 days 00:10:00
CX44   2021-05-08 12:27:00 2021-05-08 12:25:00     NaN 0 days 00:02:00
6E2133 2021-05-08 12:43:00 2021-05-08 12:40:00     NaN 0 days 00:03:00
AI565  2021-05-08 12:52:00 2021-05-08 12:52:00     NaN 0 days 00:00:00
9I513  2021-05-08 01:43:00 2021-05-08 01:40:00     NaN 0 days 00:03:00
6E435  2021-05-08 02:00:00 2021-05-08 02:00:00     NaN 0 days 00:00:00
9I536  2021-05-08 02:17:00 2021-05-08 02:15:00     NaN 0 days 00:02:00
9I898  2021-05-08 02:45:00 2021-05-08 02:40:00     NaN 0 days 00:05:00
9I876  2021-05-08 02:45:00 2021-05-08 02:45:00     NaN 0 days 00:00:00
I52990 2021-05-08 03:23:00 2021-05-08 03:15:00     NaN 0 days 00:08:00
6E734  2021-05-08 03:35:00 2021-05-08 03:30:00     NaN 0 days 00:05:00
6E379  2021-05-08 03:58:00 2021-05-08 03:40:00     NaN 0 days 00:18:00
6E587  2021-05-08 04:20:00 2021-05-08 04:05:00     NaN 0 days 00:15:00
SG706  2021-05-08 04:35:00 2021-05-08 04:35:00     NaN 0 days 00:00:00
SG3008 2021-05-08 04:51:00 2021-05-08 04:50:00     NaN 0 days 00:01:00
6E847  2021-05-08 05:05:00 2021-05-08 05:00:00     NaN 0 days 00:05:00
G8834  2021-05-08 05:20:00 2021-05-08 05:20:00     NaN 0 days 00:00:00
6E477  2021-05-08 05:48:00 2021-05-08 05:45:00     NaN 0 days 00:03:00
SG7010 2021-05-08 06:00:00 2021-05-08 06:00:00     NaN 0 days 00:00:00
6E332  2021-05-08 06:25:00 2021-05-08 06:20:00     NaN 0 days 00:05:00
G8805  2021-05-08 06:42:00 2021-05-08 06:40:00     NaN 0 days 00:02:00
6E6188 2021-05-08 07:10:00 2021-05-08 07:10:00     NaN 0 days 00:00:00
AI503  2021-05-08 07:45:00 2021-05-08 07:45:00     NaN 0 days 00:00:00
I51624 2021-05-08 08:00:00 2021-05-08 08:00:00     NaN 0 days 00:00:00
G8406  2021-05-08 08:30:00 2021-05-08 08:20:00     NaN 0 days 00:10:00
6E2487 2021-05-08 08:45:00 2021-05-08 08:35:00     NaN 0 days 00:10:00
6E1456 2021-05-08 09:04:00 2021-05-08 08:50:00     NaN 0 days 00:14:00
```

```
plt.figure(figsize=(8,6))
df8["DELAY TIME"].value_counts().plot(kind='bar',alpha=0.5)
plt.xlabel('DELAY TIME')
plt.ylabel('frequency of occurance')
plt.show()
```



```
#delay time more than 5 minutes is taken into consideration and the flight is officially declared as delayed.
#21 out of 28 have 5 or less than 5 minutes of delay.
```

```
delay_counts=df8['DELAY TIME'].value_counts()
```

```
delay_counts #9 are exactly on time 12 are within in the delay time 7 are delayed
```

```
    0 days 00:00:00    9
    0 days 00:02:00    4
    0 days 00:05:00    4
    0 days 00:03:00    3
    0 days 00:10:00    3
    0 days 00:18:00    1
    0 days 00:08:00    1
    0 days 00:15:00    1
    0 days 00:01:00    1
    0 days 00:14:00    1
    Name: DELAY TIME, dtype: int64
```

```
from datetime import timedelta
```

```
time_delta = timedelta(hours=0, minutes=5, seconds=0, microseconds=0)
```

```
delayed_Flights=delay_counts.loc[delay_counts.index >  time_delta] #7 out of 28 flights are delayed
```

```
delayed_Flights
```

```
     0 days 00:10:00    3
     0 days 00:18:00    1
     0 days 00:08:00    1
     0 days 00:15:00    1
     0 days 00:14:00    1
     Name: DELAY TIME, dtype: int64
```

```
delayed_data=df8['DELAY TIME'].loc[df8['DELAY TIME'] > time_delta]
```

```
delayed_data
```

```
     FLIGHT
     UK812    0 days 00:10:00
     I52990   0 days 00:08:00
     6E379    0 days 00:18:00
     6E587    0 days 00:15:00
     G8406    0 days 00:10:00
     6E2487   0 days 00:10:00
     6E1456   0 days 00:14:00
     Name: DELAY TIME, dtype: timedelta64[ns]
```

```
#these are the flight numbers which are delayed
# now we going to check how critical paths are affecting the delay time.
```

```
#since we know the number of delayed flights we can have the data for only those flights and do analysis.
DATA.loc['UK812',['SCHEDULED ARRIVAL','EXPECTED ARRIVAL']]
```

```
     SCHEDULED ARRIVAL   2021-05-08 11:30:00
     EXPECTED ARRIVAL    2021-05-08 11:30:00
     Name: UK812, dtype: datetime64[ns]
```

```
DATA.loc['I52990',['SCHEDULED ARRIVAL','EXPECTED ARRIVAL']]
```

```
     SCHEDULED ARRIVAL   2021-05-08 02:30:00
     EXPECTED ARRIVAL    2021-05-08 02:40:00
     Name: I52990, dtype: datetime64[ns]
```

```
DATA.loc['6E379',['SCHEDULED ARRIVAL','EXPECTED ARRIVAL']]
```

```
     SCHEDULED ARRIVAL   2021-05-08 03:10:00
     EXPECTED ARRIVAL    2021-05-08 03:10:00
     Name: 6E379, dtype: datetime64[ns]
```

```
DATA.loc['6E587',['SCHEDULED ARRIVAL','EXPECTED ARRIVAL']]
```

```
     SCHEDULED ARRIVAL   2021-05-08 03:30:00
     EXPECTED ARRIVAL    2021-05-08 03:30:00
     Name: 6E587, dtype: datetime64[ns]
```

```
DATA.loc['G8406',['SCHEDULED ARRIVAL','EXPECTED ARRIVAL']]
```

```
     SCHEDULED ARRIVAL   2021-05-08 07:45:00
     EXPECTED ARRIVAL    2021-05-08 07:45:00
     Name: G8406, dtype: datetime64[ns]
```

```
DATA.loc['6E2487',['SCHEDULED ARRIVAL','EXPECTED ARRIVAL']]
```

```
     SCHEDULED ARRIVAL   2021-05-08 08:00:00
     EXPECTED ARRIVAL    2021-05-08 08:00:00
     Name: 6E2487, dtype: datetime64[ns]
```

```
DATA.loc['6E1456',['SCHEDULED ARRIVAL','EXPECTED ARRIVAL']]
```

```
     SCHEDULED ARRIVAL   2021-05-08 08:20:00
     EXPECTED ARRIVAL    2021-05-08 08:18:00
     Name: 6E1456, dtype: datetime64[ns]
```

```
#only in 1 of the cases there is a different expected arrival which means most of the delay reasons are coming from the critical path
#flight I52990 is the only flight affected by arrival delay
```

```
data3=[df2['critical path 1'],df3['critical path 2'],df4['critical path 3'],df5['critical path 4'],df6['critical path 5'],df7['critical path 6']]
```

```
data4=pd.DataFrame(data3, columns=['UK812','I52990','6E379','6E587','G8406','6E2487','6E1456'])
```

```
data4
```

	UK812	I52990	6E379	6E587	G8406	6E2487	6E1456
critical path 1	0 days 00:11:00	0 days 00:10:00	0 days 00:09:00	0 days 00:11:00	0 days 00:10:00	0 days 00:11:00	0 days 00:10:00
critical path 2	0 days 00:14:00	0 days 00:16:00	0 days 00:16:00	0 days 00:16:00	0 days 00:17:00	0 days 00:16:00	0 days 00:17:00
critical path 3	0 days 00:06:00	0 days 00:06:00	0 days 00:10:00	0 days 00:08:00	0 days 00:08:00	0 days 00:09:00	0 days 00:09:00
critical path 4	0 days 00:37:00	0 days 00:33:00	0 days 00:40:00	0 days 00:42:00	0 days 00:37:00	0 days 00:35:00	0 days 00:38:00
critical path 5	0 days 00:09:00	0 days 00:08:00	0 days 00:10:00	0 days 00:10:00	0 days 00:09:00	0 days 00:07:00	0 days 00:10:00
critical path 6	0 days 00:06:00	0 days 00:06:00	0 days 00:07:00	0 days 00:06:00	0 days 00:07:00	0 days 00:06:00	0 days 00:07:00

```
# ideal time for critical path 1 is 9 minutes
# ideal time for critical path 2 is 16 minutes
# ideal time for critical path 3 is 7 minutes
# ideal time for critical path 4 is 31 minutes
# ideal time for critical path 5 is 8 minutes
# ideal time for critical path 6 is 5 minutes


#the mean time is taken into consideration to compare the delay time
#flight UK812 11-9 + (14-16) + (6-7) + (37-31) + (9-8) + (6-5) = 7 minutes out of the 10 minutes delay is caused by the critical paths which is 70% of the delay time.
#flight I52990 10-9 + 16-16 + (6-7) + 33- 31 + 8-8 + 6-5 = 3 minutes of the 8 minutes delay is caused by the critical paths which is 37.5% of the delay time.
#flight 6E379 9-9 + 16-16 + 10-7 + 40-31 + 10-8 + 7-5 = 16 out 18 minutes of delay is caused by critical path which is 88% of the delay time.
#flight 6E587 11-9 + 16-16 + 8-7 + 42- 31 + 10-8 + 7-5 = 16 minutes of delay is caused by the critical path but here the delay is only 15 minutes, that means the other processes which
#flight g8406 10-9 + 17-16 + 8-7 +37-31 +9-8 +7-5 = 12 minutes of the delay is caused by the critical path but here the delay is only 10 minutes and the same conclusion is applied as
#flight 6E2487 11-9 + 16-16 + 9-7 +35-31 + (7-8) +6-5 = 8 minutes of the 10 minutes delay is caused by the critical paths which is 80% of the delay time.
#flight 6E1456 10-9 + 17-16 + 9-7 +38 - 33 + 10-8 + 7-5 = 13 minutes out of 14 minutes of delay is caused by critical path which is 92% of the delay time.


#if we calculate the percentage of delay caused by the critical paths for each flight
# UK812 = 70%
# I52990 = 37.5%
# 6E379 = 88%
# 6E587 = 100%
# G8406 = 100%
# 6E2487 = 80%
# 6E1456 = 92%
```

```
delay_mean_time= 70 + 37.5 + 88 + 100 + 100 + 80 + 92
```

```
delay_mean_time / 7
```

```
81.07142857142857
```

```
# with the small dataset we have we can say that 81% of the delays are casued by the critical paths.
# we do not have a large dataset to strongly support the statement and the accuracy of this analysis is not precise but assuming the sample of this dataset to be somewhere near to acc
```