

In [1]:

```
# Carregando Pandas para uso nos calculos das planilhas
import pandas as pd
from datetime import datetime
# Carregando arquivo.csv em memória (para manipulação)
# Em meu caso, eu estou modificando o "df = pandas.read_csv('../Database/dataset.csv')"
# Eu estou utilizando uma função lambda para usar um parse no campo datas
dateparse = lambda x: datetime.strptime(x, '%H:%M:%S')
df = pd.read_csv('../Database/dataset.csv', parse_dates=True, date_parser=dateparse)
df.head(31)
```

Out[1]:

	tempo_inicio_recebimento_mercadoria	tempo_fim_recebimento_mercadoria	tempo_inicio_pre_triagem	tempo_fim_pre_triagem	tempo_inicio_raio_x	tempo_fim_raio_x
0	05:16:21	05:30:21	18:53:43	18:57:43	16:43:04	16:44:04
1	07:51:07	08:05:07	01:12:47	01:19:47	17:55:17	17:59:17
2	03:28:10	03:37:10	16:28:54	16:34:54	05:45:50	05:45:50
3	14:54:25	14:56:25	17:58:10	18:07:10	14:33:26	14:48:26
4	14:49:53	14:57:53	06:19:11	06:23:11	00:40:29	00:43:29
5	04:24:12	04:27:12	21:25:22	21:35:22	06:37:33	06:51:33
6	04:33:30	04:47:30	02:30:53	02:40:53	08:01:21	08:05:21
7	22:50:12	22:53:12	13:24:58	13:35:58	10:29:12	10:36:12
8	13:44:59	13:48:59	09:16:28	09:29:28	09:26:00	09:36:00
9	14:50:50	14:52:50	13:37:44	13:40:44	10:48:19	10:52:19
10	12:32:16	12:45:16	15:15:16	15:30:16	05:41:14	05:52:14
11	10:54:34	10:54:34	16:26:16	16:38:16	20:19:28	20:30:28
12	03:16:24	03:28:24	13:05:14	13:17:14	22:33:44	22:43:44
13	17:09:17	17:19:17	13:54:10	13:59:10	20:21:11	20:36:11
14	07:32:06	07:45:06	12:40:02	12:47:02	11:38:40	11:52:40
15	01:07:34	01:11:34	23:58:09	00:03:09	05:11:24	05:14:24
16	22:23:57	22:30:57	09:43:03	09:50:03	09:33:34	09:38:34
17	12:28:29	12:39:29	14:01:29	14:06:29	12:05:57	12:11:57
18	21:55:29	21:58:29	07:52:06	07:59:06	05:16:42	05:26:42

	tempo_inicio_recebimento_mercadoria	tempo_fim_recebimento_mercadoria	tempo_inicio_pre_triagem	tempo_fim_pre_triagem	tempo_inicio_raio_x	tempo_fim_raio_x
19	09:38:08	09:46:08	20:29:04	20:39:04	05:29:26	05:40:26
20	03:53:04	04:06:04	16:15:00	16:23:00	13:11:08	13:23:08
21	17:33:54	17:48:54	15:00:34	15:06:34	22:46:14	22:51:14
22	18:11:42	18:18:42	16:30:01	16:40:01	13:57:02	14:01:02
23	18:26:26	18:37:26	01:54:48	02:09:48	04:18:59	04:31:59
24	04:42:27	04:47:27	15:06:39	15:20:39	08:54:14	09:06:14
25	04:48:58	04:52:58	04:44:45	04:47:45	23:08:56	23:18:56
26	16:57:28	17:02:28	08:40:10	08:54:10	05:27:54	05:35:54
27	16:41:53	16:55:53	16:24:08	16:34:08	03:03:40	03:05:40
28	10:04:45	10:13:45	18:51:15	18:05:15	13:04:02	13:10:02

```
In [2]: #  
# Etapas para tratar os dados e realizar os cálculos - MTTF, MTTR, MTBF e Disponibilidade - Tempo de Recebimento de Mercadoria  
#
```

```
In [3]: # Primeiramente, iremos Ordenar o valores de recebimento, para criar uma sequência de dados crescente (tempo_inicio_recebimento_mercad
```

```
In [4]: df.sort_values(by=['tempo_inicio_recebimento_mercadoria'])
```

	tempo_inicio_recebimento_mercadoria	tempo_fim_recebimento_mercadoria	tempo_inicio_pre_triagem	tempo_fim_pre_triagem	tempo_inicio_raio_x	tempo_fim_raio_x
15	01:07:34	01:11:34	23:58:09	00:03:09	05:11:24	05:14:24
12	03:16:24	03:28:24	13:05:14	13:17:14	22:33:44	22:43:44
2	03:28:10	03:37:10	16:28:54	16:34:54	05:45:50	05:45:50
20	03:53:04	04:06:04	16:15:00	16:23:00	13:11:08	13:23:08
5	04:24:12	04:27:12	21:25:22	21:35:22	06:37:33	06:51:33
6	04:33:30	04:47:30	02:30:53	02:40:53	08:01:21	08:05:21
24	04:42:27	04:47:27	15:06:39	15:20:39	08:54:14	09:06:14
25	04:48:58	04:52:58	04:44:45	04:47:45	23:08:56	23:18:56
0	05:16:21	05:30:21	18:53:43	18:57:43	16:43:04	16:44:04

	tempo_inicio_recebimento_mercadoria	tempo_fim_recebimento_mercadoria	tempo_inicio_pre_triagem	tempo_fim_pre_triagem	tempo_inicio_raio_x	tempo_fim_raio_x
14	07:32:06	07:45:06	12:40:02	12:47:02	11:38:40	11:52:40
1	07:51:07	08:05:07	01:12:47	01:19:47	17:55:17	17:59:17
19	09:38:08	09:46:08	20:29:04	20:39:04	05:29:26	05:40:26
28	10:04:45	10:13:45	18:51:15	19:05:15	13:04:02	13:19:02
11	10:54:34	10:54:34	16:26:16	16:38:16	20:19:28	20:30:28
17	12:28:29	12:39:29	14:01:29	14:06:29	12:05:57	12:11:57
10	12:32:16	12:45:16	15:15:16	15:30:16	05:41:14	05:52:14
8	13:44:59	13:48:59	09:16:28	09:29:28	09:26:00	09:36:00
4	14:49:53	14:57:53	06:19:11	06:23:11	00:40:29	00:43:29
9	14:50:50	14:52:50	13:37:44	13:40:44	10:48:19	10:52:19
3	14:54:25	14:56:25	17:58:10	18:07:10	14:33:26	14:48:26
27	16:41:53	16:55:53	16:24:08	16:34:08	03:03:40	03:05:40
26	16:57:28	17:02:28	08:40:10	08:54:10	05:27:54	05:35:54
13	17:09:17	17:19:17	13:54:10	13:59:10	20:21:11	20:36:11
21	17:33:54	17:48:54	15:00:34	15:06:34	22:46:14	22:51:14
22	18:11:42	18:18:42	16:30:01	16:40:01	13:57:02	14:01:02
23	18:26:26	18:37:26	01:54:48	02:09:48	04:18:59	04:31:59
29	19:26:42	19:37:42	20:57:19	21:09:19	23:06:30	23:13:30
18	21:55:29	21:58:29	07:52:06	07:59:06	05:16:42	05:26:42
16	22:23:57	22:30:57	09:43:03	09:50:03	09:33:34	09:38:34

```
In [5]: tempo_inicio_recebimento_mercadoria = df['tempo_inicio_recebimento_mercadoria']
```

```
In [6]: tempo_fim_recebimento_mercadoria = df['tempo_fim_recebimento_mercadoria']
```

```
In [7]: tempo_inicio_recebimento_mercadoria = pd.to_datetime(tempo_inicio_recebimento_mercadoria)
```

```
In [8]: tempo_fim_recebimento_mercadoria = pd.to_datetime(tempo_fim_recebimento_mercadoria)
```

```
In [9]: # Usando uma variável para armazenar o valores de Tempo Total Disponível (1ª tempo inicial - último tempo final)  
# Com a subtração acima, é possível ter um valor de tempo disponível
```

```
In [10]: tempo_total_disponivel_recebimento_mercadoria = tempo_fim_recebimento_mercadoria[7] - tempo_inicio_recebimento_mercadoria[15]
```

```
In [11]: tempo_total_disponivel_recebimento_mercadoria
```

```
Out[11]: Timedelta('0 days 21:45:38')
```

```
In [12]: tempo_duracao_recebimento_mercadoria = tempo_fim_recebimento_mercadoria - tempo_inicio_recebimento_mercadoria
```

```
In [13]: tempo_duracao_recebimento_mercadoria
```

```
Out[13]: 0    0 days 00:14:00  
1    0 days 00:14:00  
2    0 days 00:09:00  
3    0 days 00:02:00  
4    0 days 00:08:00  
5    0 days 00:03:00  
6    0 days 00:14:00  
7    0 days 00:03:00  
8    0 days 00:04:00  
9    0 days 00:02:00  
10   0 days 00:13:00  
11   0 days 00:00:00  
12   0 days 00:12:00  
13   0 days 00:10:00  
14   0 days 00:13:00  
15   0 days 00:04:00  
16   0 days 00:07:00  
17   0 days 00:11:00  
18   0 days 00:03:00  
19   0 days 00:08:00  
20   0 days 00:13:00  
21   0 days 00:15:00  
22   0 days 00:07:00  
23   0 days 00:11:00  
24   0 days 00:05:00  
25   0 days 00:04:00
```

```
26    0 days 00:05:00
27    0 days 00:14:00
28    0 days 00:09:00
29    0 days 00:11:00
dtype: timedelta64[ns]
```

```
In [14]: # Usando uma variável para armazenar Tempo Total Utilizado (somando todos os valores da sequência de tempo de duração)
# Com esta operação, é possível ter o valor final de tempo de atividade (produção)
```

```
In [15]: tempo_total_utilizado_recebimento_mercadoria = tempo_duracao_recebimento_mercadoria.sum()
```

```
In [16]: tempo_total_utilizado_recebimento_mercadoria
```

```
Out[16]: Timedelta('0 days 04:08:00')
```

```
In [17]: # Usando uma variável para armazenar o Tempo Total de Paradas (subtraindo do Tempo Total Disponível o Tempo Total Utilizado)
```

```
In [18]: tempo_total_parada_recebimento_mercadoria = tempo_total_disponivel_recebimento_mercadoria - tempo_total_utilizado_recebimento_mercador
```

```
In [19]: tempo_total_parada_recebimento_mercadoria
```

```
Out[19]: Timedelta('0 days 17:37:38')
```

```
In [20]: # Usando uma variável para armazenar as ocorrências de paradas (quantidade de registro -1, nos mostra todas as paradas da série)
```

```
In [21]: qtde_paradas_recebimento_mercadorias = tempo_duracao_recebimento_mercadoria.count() - 1
```

```
In [22]: qtde_paradas_recebimento_mercadorias
```

```
Out[22]: 29
```

```
In [23]: # Calculando o MTTF das Colunas de Recebimento
```

```
In [24]: mttf_recebimento_mercadoria = tempo_duracao_recebimento_mercadoria.mean()
```

```
In [25]: mttf_recebimento_mercadoria
```

```
Out[25]: Timedelta('0 days 00:08:16')
```

```
In [26]: # Calculando o MTTR das Colunas de Recebimento
```

```
In [27]: mtrr_recebimento_mercadoria = tempo_total_parada_recebimento_mercadoria / qtde_paradas_recebimento_mercadorias
```

```
In [28]: mtrr_recebimento_mercadoria
```

```
Out[28]: Timedelta('0 days 00:36:28.206896551')
```

```
In [29]: # Calculando o MTBF das Colunas de Recebimento
```

```
In [30]: mtbf_recebimento_mercadoria = (tempo_total_disponivel_recebimento_mercadoria - tempo_total_parada_recebimento_mercadoria) / qtde_parad
```

```
In [31]: mtbf_recebimento_mercadoria
```

```
Out[31]: Timedelta('0 days 00:08:33.103448275')
```

```
In [32]: # Calculando a Disponibilidade das Colunas de Recebimento
```

```
In [33]: disponibilidade_recebimento_mercadoria = mtbf_recebimento_mercadoria / (mtrr_recebimento_mercadoria + mtbf_recebimento_mercadoria)
```

```
In [34]: disponibilidade_recebimento_mercadoria
```

```
Out[34]: 0.18994613086859172
```

```
In [35]: #  
# Etapas para tratar os dados e realizar os cálculos - MTTF, MTTR, MTBF e Disponibilidade - Tempo de Pré Triagem  
#
```

```
In [36]: # Primeiramente, iremos Ordenar o valores de recebimento, para criar uma sequência de dados crescente (tempo_inicio_pre_triagem)
```

```
In [37]: df.sort_values(by=['tempo_inicio_pre_triagem'])
```

```
Out[37]:
```

	tempo_inicio_recebimento_mercadoria	tempo_fim_recebimento_mercadoria	tempo_inicio_pre_triagem	tempo_fim_pre_triagem	tempo_inicio_raio_x	tempo_fim_raio_x
1	07:51:07	08:05:07	01:12:47	01:19:47	17:55:17	17:59:17
23	18:26:26	18:37:26	01:54:48	02:09:48	04:18:59	04:31:59
6	04:33:30	04:47:30	02:30:53	02:40:53	08:01:21	08:05:21
25	04:48:58	04:52:58	04:44:45	04:47:45	23:08:56	23:18:56
4	14:49:53	14:57:53	06:19:11	06:23:11	00:40:29	00:43:29
18	21:55:29	21:58:29	07:52:06	07:59:06	05:16:42	05:26:42
26	16:57:28	17:02:28	08:40:10	08:54:10	05:27:54	05:35:54
8	13:44:59	13:48:59	09:16:28	09:29:28	09:26:00	09:36:00
16	22:23:57	22:30:57	09:43:03	09:50:03	09:33:34	09:38:34
14	07:32:06	07:45:06	12:40:02	12:47:02	11:38:40	11:52:40
12	03:16:24	03:28:24	13:05:14	13:17:14	22:33:44	22:43:44
7	22:50:12	22:53:12	13:24:58	13:35:58	10:29:12	10:36:12
9	14:50:50	14:52:50	13:37:44	13:40:44	10:48:19	10:52:19
13	17:09:17	17:19:17	13:54:10	13:59:10	20:21:11	20:36:11
17	12:28:29	12:39:29	14:01:29	14:06:29	12:05:57	12:11:57
21	17:33:54	17:48:54	15:00:34	15:06:34	22:46:14	22:51:14
24	04:42:27	04:47:27	15:06:39	15:20:39	08:54:14	09:06:14
10	12:32:16	12:45:16	15:15:16	15:30:16	05:41:14	05:52:14
20	03:53:04	04:06:04	16:15:00	16:23:00	13:11:08	13:23:08
27	16:41:53	16:55:53	16:24:08	16:34:08	03:03:40	03:05:40
11	10:54:34	10:54:34	16:26:16	16:38:16	20:19:28	20:30:28
2	03:28:10	03:37:10	16:28:54	16:34:54	05:45:50	05:45:50

	tempo_inicio_recebimento_mercadoria	tempo_fim_recebimento_mercadoria	tempo_inicio_pre_triagem	tempo_fim_pre_triagem	tempo_inicio_raio_x	tempo_fim_raio_x
22	18:11:42	18:18:42	16:30:01	16:40:01	13:57:02	14:01:02
3	14:54:25	14:56:25	17:58:10	18:07:10	14:33:26	14:48:26
28	10:04:45	10:13:45	18:51:15	19:05:15	13:04:02	13:19:02
0	05:16:21	05:30:21	18:53:43	18:57:43	16:43:04	16:44:04
19	09:38:08	09:46:08	20:29:04	20:39:04	05:29:26	05:40:26
29	19:26:42	19:37:42	20:57:19	21:09:19	23:06:30	23:13:30
5	04:24:12	04:27:12	21:25:22	21:25:22	06:27:22	06:51:22

```
In [38]: tempo_inicio_pre_triagem = df['tempo_inicio_pre_triagem']
```

```
In [39]: tempo_fim_pre_triagem = df['tempo_fim_pre_triagem']
```

```
In [40]: tempo_inicio_pre_triagem = pd.to_datetime(tempo_inicio_pre_triagem)
```

```
In [41]: tempo_fim_pre_triagem = pd.to_datetime(tempo_fim_pre_triagem)
```

```
In [42]: # Usando uma variável para armazenar o valores de Tempo Total Disponível (1ª tempo inicial - último tempo final)
# Com a subtração acima, é possível ter um valor de tempo disponível
# O uso da operação "+ pd.Timedelta("1 days")" é para corrigir erro do valor negativo.
```

```
In [43]: tempo_total_disponivel_pre_triagem = (tempo_fim_pre_triagem[15] - tempo_inicio_pre_triagem[1]) + pd.Timedelta("1 days")
```

```
In [44]: tempo_total_disponivel_pre_triagem
```

```
Out[44]: Timedelta('0 days 22:50:22')
```

```
In [45]: tempo_duracao_pre_triagem = tempo_fim_pre_triagem - tempo_inicio_pre_triagem
```

```
In [46]: tempo_duracao_pre_triagem
```

```
Out[46]: 0      0 days 00:04:00
```



```
1      0 days 00:07:00
2      0 days 00:06:00
3      0 days 00:09:00
4      0 days 00:04:00
5      0 days 00:10:00
6      0 days 00:10:00
7      0 days 00:11:00
8      0 days 00:13:00
9      0 days 00:03:00
10     0 days 00:15:00
11     0 days 00:12:00
12     0 days 00:12:00
13     0 days 00:05:00
14     0 days 00:07:00
15    -1 days +00:05:00
16     0 days 00:07:00
17     0 days 00:05:00
18     0 days 00:07:00
19     0 days 00:10:00
20     0 days 00:08:00
21     0 days 00:06:00
22     0 days 00:10:00
23     0 days 00:15:00
24     0 days 00:14:00
25     0 days 00:03:00
26     0 days 00:14:00
27     0 days 00:10:00
28     0 days 00:14:00
29     0 days 00:12:00
dtype: timedelta64[ns]
```

```
In [47]: # Usando uma variável para armazenar Tempo Total Utilizado (somando todos os valores da sequência de tempo de duração)
# Com esta operação, é possível ter o valor final de tempo de atividade (produção)
```

```
In [48]: tempo_total_utilizado_pre_triagem = tempo_duracao_pre_triagem.sum() + pd.Timedelta("1 days")
```

```
In [49]: tempo_total_utilizado_pre_triagem
```

```
Out[49]: Timedelta('0 days 04:28:00')
```

```
In [50]: # Usando uma variável para armazenar o Tempo Total de Paradas (subtraindo do Tempo Total Disponível o Tempo Total Utilizado)
```

```
In [51]: tempo_total_parada_pre_triagem = tempo_total_disponivel_pre_triagem - tempo_total_utilizado_pre_triagem
```

```
In [52]: tempo_total_parada_pre_triagem
```

```
Out[52]: Timedelta('0 days 18:22:22')
```

```
In [53]: # Usando uma variável para armazenar as ocorrências de paradas (quantidade de registro -1, nos mostra todas as paradas da série)
```

```
In [54]: qtde_paradas_pre_triagem = tempo_duracao_pre_triagem.count() - 1
```

```
In [55]: qtde_paradas_pre_triagem
```

```
Out[55]: 29
```

```
In [56]: # Calculando o MTTF das Colunas de Pré Triagem
```

```
In [57]: # Correção do item 15 da nossa lista (-1 days +00:05:00).  
# A correção é necessária para desprezar o valor de "-1 days" do resultado. Correto é "00:05:00"  
# Observação-1: a multiplicação por -1 (*-1) não resolvia o problema, por que estamos trabalhando com tipo data/hora  
# Observação-2: esta solução deve ser executada uma única vez. Caso necessário uma segunda ou mais execuções  
#          será necessário "limpar todas as saídas" e reexecutar todos os calculos novamente.
```

```
In [58]: x_tempo_duracao_pre_triagem = tempo_duracao_pre_triagem
```

```
In [59]: x_tempo_duracao_pre_triagem[15] = x_tempo_duracao_pre_triagem[15] + pd.Timedelta("1 days")
```

```
In [60]: mttf_pre_triagem = x_tempo_duracao_pre_triagem.mean()
```

```
In [61]: mttf_pre_triagem
```

```
Out[61]: Timedelta('0 days 00:08:56')
```

```
In [62]: # Calculando o MTTR das Colunas de Pré Triagem
```

```
In [63]: mttr_pre_triagem = tempo_total_parada_pre_triagem / qtde_paradas_pre_triagem
```

```
In [64]: mttr_pre_triagem
```

```
Out[64]: Timedelta('0 days 00:38:00.758620689')
```

```
In [65]: # Calculando o MTBF das Colunas de Pré Triagem
```

```
In [66]: mtbf_pre_triagem = (tempo_total_disponivel_pre_triagem - tempo_total_parada_pre_triagem) / qtde_paradas_pre_triagem
```

```
In [67]: mtbf_pre_triagem
```

```
Out[67]: Timedelta('0 days 00:09:14.482758620')
```

```
In [68]: # Calculando a Disponibilidade das Colunas de Pré Triagem
```

```
In [69]: disponibilidade_pre_triagem = mtbf_pre_triagem / (mttr_pre_triagem + mtbf_pre_triagem)
```

```
In [70]: disponibilidade_pre_triagem
```

```
Out[70]: 0.19556809612983905
```

```
In [71]: #  
# Etapas para tratar os dados e realizar os cálculos - MTTF, MTTR, MTBF e Disponibilidade - Tempo de Raio X  
#
```

```
In [72]: # Primeiramente, iremos Ordenar o valores de recebimento, para criar uma sequência de dados crescente (tempo_inicio_raio_x)
```

```
In [73]: df.sort_values(by=['tempo_inicio_raio_x'])
```

Out[73]:

	tempo_inicio_recebimento_mercadoria	tempo_fim_recebimento_mercadoria	tempo_inicio_pre_triagem	tempo_fim_pre_triagem	tempo_inicio_raio_x	tempo_fim_raio_x
4	14:49:53	14:57:53	06:19:11	06:23:11	00:40:29	00:43:29
27	16:41:53	16:55:53	16:24:08	16:34:08	03:03:40	03:05:40
23	18:26:26	18:37:26	01:54:48	02:09:48	04:18:59	04:31:59
15	01:07:34	01:11:34	23:58:09	00:03:09	05:11:24	05:14:24
18	21:55:29	21:58:29	07:52:06	07:59:06	05:16:42	05:26:42
26	16:57:28	17:02:28	08:40:10	08:54:10	05:27:54	05:35:54
19	09:38:08	09:46:08	20:29:04	20:39:04	05:29:26	05:40:26
10	12:32:16	12:45:16	15:15:16	15:30:16	05:41:14	05:52:14
2	03:28:10	03:37:10	16:28:54	16:34:54	05:45:50	05:45:50
5	04:24:12	04:27:12	21:25:22	21:35:22	06:37:33	06:51:33
6	04:33:30	04:47:30	02:30:53	02:40:53	08:01:21	08:05:21
24	04:42:27	04:47:27	15:06:39	15:20:39	08:54:14	09:06:14
8	13:44:59	13:48:59	09:16:28	09:29:28	09:26:00	09:36:00
16	22:23:57	22:30:57	09:43:03	09:50:03	09:33:34	09:38:34
7	22:50:12	22:53:12	13:24:58	13:35:58	10:29:12	10:36:12
9	14:50:50	14:52:50	13:37:44	13:40:44	10:48:19	10:52:19
14	07:32:06	07:45:06	12:40:02	12:47:02	11:38:40	11:52:40
17	12:28:29	12:39:29	14:01:29	14:06:29	12:05:57	12:11:57
28	10:04:45	10:13:45	18:51:15	19:05:15	13:04:02	13:19:02
20	03:53:04	04:06:04	16:15:00	16:23:00	13:11:08	13:23:08
22	18:11:42	18:18:42	16:30:01	16:40:01	13:57:02	14:01:02
3	14:54:25	14:56:25	17:58:10	18:07:10	14:33:26	14:48:26
0	05:16:21	05:30:21	18:53:43	18:57:43	16:43:04	16:44:04
1	07:51:07	08:05:07	01:12:47	01:19:47	17:55:17	17:59:17
11	10:54:34	10:54:34	16:26:16	16:38:16	20:19:28	20:30:28
13	17:09:17	17:19:17	13:54:10	13:59:10	20:21:11	20:36:11

	tempo_inicio_recebimento_mercadoria	tempo_fim_recebimento_mercadoria	tempo_inicio_pre_triagem	tempo_fim_pre_triagem	tempo_inicio_raio_x	tempo_fim_raio_x
12	03:16:24	03:28:24	13:05:14	13:17:14	22:33:44	22:43:44
21	17:33:54	17:48:54	15:00:34	15:06:34	22:46:14	22:51:14
29	19:26:42	19:37:42	20:57:19	21:09:19	23:06:30	23:13:30

```
In [74]: tempo_inicio_raio_x = df['tempo_inicio_raio_x']
```

```
In [75]: tempo_fim_raio_x = df['tempo_fim_raio_x']
```

```
In [76]: tempo_inicio_raio_x = pd.to_datetime(tempo_inicio_raio_x)
```

```
In [77]: tempo_fim_raio_x = pd.to_datetime(tempo_fim_raio_x)
```

```
In [78]: # Usando uma variável para armazenar o valores de Tempo Total Disponível (1ª tempo inicial - último tempo final)  
# Com a subtração acima, é possível ter um valor de tempo disponível
```

```
In [79]: tempo_total_disponivel_raio_x = (tempo_fim_raio_x[4] - tempo_inicio_raio_x[27]) + pd.Timedelta("1 days")
```

```
In [80]: tempo_total_disponivel_raio_x
```

```
Out[80]: Timedelta('0 days 21:39:49')
```

```
In [81]: tempo_duracao_raio_x = tempo_fim_raio_x - tempo_inicio_raio_x
```

```
In [82]: # Usando uma variável para armazenar Tempo Total Utilizado (somando todos os valores da sequência de tempo de duração)  
# Com esta operação, é possível ter o valor final de tempo de atividade (produção)
```

```
In [83]: tempo_total_utilizado_raio_x = tempo_duracao_raio_x.sum()
```

```
In [84]: tempo_total_utilizado_raio_x
```

```
Out[84]: Timedelta('0 days 04:06:00')
```

```
In [85]: # Usando uma variável para armazenar o Tempo Total de Paradas (subtraindo do Tempo Total Disponível o Tempo Total Utilizado)
```

```
In [86]: tempo_total_parada_raio_x = tempo_total_disponivel_raio_x - tempo_total_utilizado_raio_x
```

```
In [87]: tempo_total_parada_raio_x
```

```
Out[87]: Timedelta('0 days 17:33:49')
```

```
In [88]: # Usando uma variável para armazenar as ocorrências de paradas (quantidade de registro -1, nos mostra todas as paradas da série)
```

```
In [89]: qtde_paradas_raio_x = tempo_duracao_raio_x.count() - 1
```

```
In [90]: qtde_paradas_raio_x
```

```
Out[90]: 29
```

```
In [91]: # Calculando o MTTF das Colunas de Raio X
```

```
In [92]: mttf_raio_x = tempo_duracao_raio_x.mean()
```

```
In [93]: mttf_raio_x
```

```
Out[93]: Timedelta('0 days 00:08:12')
```

```
In [94]: # Calculando o MTTR das Colunas de Raio X
```

```
In [95]: mttr_raio_x = tempo_total_parada_raio_x / qtde_paradas_raio_x
```

```
In [96]: mttr_raio_x
```

```
Out[96]: Timedelta('0 days 00:36:20.310344827')
```

```
In [97]: # Calculando o MTBF das Colunas de Raio X
```

```
In [98]: mtbf_raio_x = (tempo_total_disponivel_raio_x - tempo_total_parada_raio_x) / qtde_paradas_raio_x
```

```
In [99]: mtbf_raio_x
```

```
Out[99]: Timedelta('0 days 00:08:28.965517241')
```

```
In [1... # Calculando a Disponibilidade das Colunas de Raio X
```

```
In [1... disponibilidade_raio_x = mtbf_raio_x / (mttr_raio_x + mtbf_raio_x)
```

```
In [1... disponibilidade_raio_x
```

```
Out[1... 0.18925745938522484
```

```
In [ ]:
```