```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import random
```

Creacion de los dataframe

```
lotes = pd.DataFrame(columns = ['id_lote', 'fecha_programada','id_destino', 'desc', 'id_categoria', 'cantidad_unidades'], dty
destinos = pd.DataFrame(columns = ['id_destino', 'nombre','atencion_critica', 'region', 'latitud', 'longitud'], dtype = int)
categorias_insumos = pd.DataFrame(columns = ['id_categoria', 'nombre','uso_esencial'], dtype = int)


for i in range(800):
    lotes = lotes.append({
        'id_lote': random.randrange(50),
        'fecha_programada': random.choice(['2020-12-14', '2019-12-14', '2018-10-12', '2020-09-16', '2020-09-10']),
        'id_destino': random.randrange(15),
        'desc': random.choice(['desc1', 'desc2', 'desc3']),
        'id_categoria': random.randrange(10),
        'cantidad_unidades': random.randrange(100)
    }, ignore_index = True)

for i in range(15):
  destinos = destinos.append({
      'id_destino': i,
      'nombre': random.choice(['nombre1', 'nombre2', 'nombre3', 'nombre4']),
      'atencion_critica': random.choice([True, False]),
      'region': random.choice(['CABA', 'region2', 'region3', 'region4']),
      'latitud': random.randrange(150),
      'longitud': random.randrange(150)
  }, ignore_index=True)

for i in range(10):
```

```
for i in range(10):
  categorias_insumos = categorias_insumos.append({
      'id_categoria': i,
      'nombre': random.choice(['nombre1', 'nombre2', 'nombre3', 'nombre4']),
      'uso_esencial': random.choice([True, False])
  }, ignore_index=True)
```

Manipulando los tipos de datos

```
lotes['fecha_programada'] = pd.to_datetime( lotes['fecha_programada'], errors='coerce' )
lotes
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 6 columns):
 #   Column             Non-Null Count  Dtype         
---  ------             --------------  -----         
 0   id_lote            800 non-null    int64         
 1   fecha_programada   800 non-null    datetime64[ns]
 2   id_destino         800 non-null    int64         
 3   desc               800 non-null    object        
 4   id_categoria       800 non-null    int64         
 5   cantidad_unidades  800 non-null    int64         
dtypes: datetime64[ns](1), int64(4), object(1)
memory usage: 37.6+ KB
```

```
destinos['atencion_critica'] = destinos['atencion_critica'] == 1
destinos
```

	id_destino	nombre	atencion_critica	region	latitud	longitud
0	0	nombre1	True	region4	140	10
1	1	nombre2	False	region2	5	45
2	2	nombre1	False	region2	81	4
3	3	nombre2	True	region3	0	71
4	4	nombre1	False	region2	110	134
5	5	nombre2	True	region4	132	124
6	6	nombre3	False	CABA	59	134
7	7	nombre4	False	region3	131	91
8	8	nombre2	False	CABA	59	102
9	9	nombre3	True	region2	133	125
10	10	nombre1	True	CABA	75	60

```
categorias_insumos['uso_esencial'] = categorias_insumos['uso_esencial'] == 1
categorias_insumos
```

	id_categoria	nombre	uso_esencial
0	0	nombre1	False
5	5	nombre2	True
7	7	nombre4	True

a) Listar los nombres de 10 destinos de la región de “CABA” que durante el mes de septiembre recibieron la mayor cantidad de lotes de insumos de uso esencial.

Con "cantidad de lotes de insumos" entiendo que hace referencia a la columna "cantidad_de_unidades_insumo" del dataframe lotes

Arrancamos a Filtrar

```
lotes_septiembre = lotes[ (lotes['fecha_programada'].dt.month == 9) ]
lotes_septiembre
```

```
destinos_caba = destinos[ destinos['region'] == 'CABA' ]
nombres_destinos_caba = destinos_caba[['id_destino', 'nombre']]
nombres_destinos_caba
```

```
uso_esencial = categorias_insumos[ categorias_insumos['uso_esencial'] ]
ids_uso_esencial = uso_esencial['id_categoria']
ids_uso_esencial.to_frame
# Como ya esta filtrado, lo unico que necesitamos son las categorias.
```

```
<bound method Series.to_frame of 1    1
5    5
6    6
7    7
Name: id_categoria, dtype: int64>
```

Mergeando los 3 dataframes

```
merged = lotes_septiembre.merge( nombres_destinos_caba, on='id_destino' )
merged.head()
```

```
merged = merged.merge( ids_uso_esencial, on='id_categoria' )
merged.head()
```

Resultado

```
merged.nlargest(10, 'cantidad_unidades')['nombre']
```

```
0     nombre4
22    nombre3
28    nombre2
6     nombre2
29    nombre3
15    nombre3
31    nombre2
20    nombre4
5     nombre2
16    nombre2
Name: nombre, dtype: object
```

B) Listar el resumen del plan de entregas para la transportadora del dia de hoy.

Este debe tener el siguiente formato (fecha_programada_entrega, nombre_destino, cantidad_de_lotes_a_entregar). A su vez debe estar ordenado de tal forma que primero estén listados aquellos destinos que son de atención crítica.

```
fecha_de_hoy = '2020-12-14'
```

Filtrando

```
destinos_criticos = destinos[ destinos['atencion_critica'] ]
destinos_criticos = destinos_criticos[['id_destino', 'nombre']]
destinos_criticos
```

	id_destino	nombre
0	0	nombre1
3	3	nombre2
5	5	nombre2
9	9	nombre3
10	10	nombre1
13	13	nombre4
14	14	nombre4

```
destinos_no_criticos = destinos[ destinos['atencion_critica'] == False ]
destinos_no_criticos = destinos_no_criticos[['id_destino', 'nombre']]
destinos_no_criticos
```

	id_destino	nombre
1	1	nombre2
2	2	nombre1
4	4	nombre1
6	6	nombre2

```
resumen = lotes[ lotes['fecha_programada'] == fecha_de_hoy ]
resumen = resumen[['id_destino', 'fecha_programada']]
resumen
```

	id_destino	fecha_programada
12	12	2020-12-14
16	14	2020-12-14
22	5	2020-12-14
30	9	2020-12-14
31	4	2020-12-14
...	...	...
762	0	2020-12-14
766	10	2020-12-14
776	4	2020-12-14
788	12	2020-12-14
799	0	2020-12-14

150 rows × 2 columns

Agrupando

Criticos

```
resumen_criticos = resumen.merge( destinos_criticos, on='id_destino' )
resumen_criticos
```

	id_destino	fecha_programada	nombre
0	14	2020-12-14	nombre4
1	14	2020-12-14	nombre4
2	14	2020-12-14	nombre4
3	14	2020-12-14	nombre4
4	14	2020-12-14	nombre4
...	...	...	...
71	3	2020-12-14	nombre2
72	3	2020-12-14	nombre2
73	3	2020-12-14	nombre2
74	3	2020-12-14	nombre2
75	3	2020-12-14	nombre2

76 rows × 3 columns

```
resumen_criticos = resumen_criticos.groupby('nombre').agg({'nombre':'count'})
resumen_criticos.columns = ['cantidad_lotes']
resumen_criticos
```

nombre	cantidad_lotes
nombre1	31
nombre2	17

No criticos

nombre	16

```
resumen_no_criticos = resumen.merge( destinos_no_criticos, on='id_destino')
resumen_no_criticos
```

	id_destino	fecha_programada	nombre
0	12	2020-12-14	nombre4
1	12	2020-12-14	nombre4
2	12	2020-12-14	nombre4
3	12	2020-12-14	nombre4
4	12	2020-12-14	nombre4
...	...	...	...
69	6	2020-12-14	nombre3
70	6	2020-12-14	nombre3
71	2	2020-12-14	nombre1
72	2	2020-12-14	nombre1
73	2	2020-12-14	nombre1

74 rows × 3 columns

```
resumen_no_criticos = resumen_no_criticos.groupby('nombre').agg({'nombre':'count'})
resumen_no_criticos.columns = ['cantidad_lotes']
resumen_no_criticos
```

nombre	cantidad_lotes
nombre1	19
nombre2	21
nombre3	12
nombre4	22

Juntando todo

```
resultado = resumen_criticos.append(resumen_no_criticos).reset_index()
resultado
```

	nombre	cantidad_lotes
0	nombre1	31
1	nombre2	17
2	nombre3	12
3	nombre4	16
4	nombre1	19
5	nombre2	21
6	nombre3	12
7	nombre4	22

```
resultado.insert( loc=0, column='fecha_programada', value=fecha_de_hoy )
resultado
```

	fecha_programada	nombre	cantidad_lotes
0	2020-12-14	nombre1	31
1	2020-12-14	nombre2	17
2	2020-12-14	nombre3	12
3	2020-12-14	nombre4	16
4	2020-12-14	nombre1	19
5	2020-12-14	nombre2	21
6	2020-12-14	nombre3	12
7	2020-12-14	nombre4	22