Análisis de Datos

Integrantes del Grupo 3 - Aprodatos:

Micaela Rosa Dorado

Guillermo German Jalil

Nicolás Paul Ramírez Moreale

Gabriel Iván Jofre

Alan Aramayo

Camila Funes

Quienes somos

Somos Aprodatos, una empresa innovadora y de vanguardia en el análisis de datos. Aunque somos nuevos en el mercado, nuestro equipo está compuesto por profesionales con experiencia en ingeniería de datos, estadística avanzada, y visualización de información, todos comprometidos en transformar datos en insights poderosos que impulsan la toma de decisiones estratégicas. Nos especializamos en procesos de extracción, transformación y carga (ETL) de datos públicos y privados, con un enfoque único en la creación de soluciones personalizadas para nuestros clientes. En Aprodatos creemos en la democratización de los datos y trabajamos con plataformas de última generación que permiten a nuestros clientes tener acceso a visualizaciones interactivas y fáciles de entender. Gracias a esto, logramos que la información más compleja se convierta en un activo accesible para todos.

Nuestra misión

Ayudamos a empresas y organizaciones a comprender mejor sus entornos de negocio, identificar patrones y oportunidades, y optimizar sus operaciones a través del poder de los datos. Con Aprodatos, nuestros clientes pueden contar con un aliado confiable y visionario en la era del big data.

Presentación del problema

Primera comunicación con el cliente:

Estimado equipo de Aprodatos,

Mi nombre es Michael Reed, representante de EasyJet, una compañía multinacional de aerolíneas low-cost interesada en expandir nuestras operaciones al mercado de vuelos internos en Argentina. Consideramos que el mercado argentino de transporte aéreo está listo para la innovación y la competitividad que caracterizan nuestro modelo de negocio, el cual busca ofrecer vuelos accesibles y convenientes para todos.

Para lograr una inserción exitosa, necesitamos un análisis detallado de los patrones de consumo y la dinámica de los vuelos en el ámbito nacional, específicamente en vuelos entre provincias. Nos interesa comprender profundamente el comportamiento de los pasajeros: los meses de mayor demanda, los días y horarios en los que se realizan más vuelos, y las provincias que concentran un mayor volumen de tráfico aéreo. Esta información es crucial para definir nuestras estrategias de rutas, precios y promoción.

Creemos que su enfoque innovador y dinámico en el manejo de datos puede brindarnos la perspectiva fresca que buscamos para evaluar este mercado en profundidad. Quedamos a la espera de su propuesta para una presentación de estos datos en un tablero visual que nos permita analizar los indicadores de manera rápida y efectiva.

Atentamente,

Michael Reed

EasyJet

Conclusión de la problemática planteada:

EasyJet, una aerolínea de bajo costo con presencia multinacional, busca expandir sus operaciones al mercado argentino de vuelos internos. Solicitan un análisis detallado de:

- Patrones de demanda estacional y mensual.
- Días y horarios pico de vuelos entre provincias.
- Provincias con mayor tráfico aéreo en el ámbito nacional.

Objetivo de la Solución

Proporcionar un análisis exhaustivo de patrones de consumo y dinámica en el mercado de vuelos internos en Argentina, centrándonos en:

- Identificación de patrones de demanda estacional y mensual.
- Detección de días y horarios pico para vuelos entre provincias.
- Análisis del tráfico aéreo entre provincias, destacando las de mayor volumen de pasajeros

---

Metodología y Estrategia de Análisis

- **Recopilación de Datos**
  Utilizaremos datos obtenidos de un archivo CSV con carga manual, el cual será actualizado periódicamente para reflejar patrones recientes en vuelos provinciales.
- **Procesamiento y Análisis de Datos (ETL)**
  A través de un flujo de ETL, transformaremos los datos en un formato adecuado, focalizándonos en vuelos interprovinciales. Este procesamiento incluirá:
    - Clasificación de vuelos por origen y destino.
    - Agrupación por fechas, horarios y períodos de alta demanda.
    - Detección de patrones estacionales y picos de actividad.

- **Visualización de Datos en Tableau**
  Crearemos un tablero interactivo en Tableau, donde se podrán analizar los indicadores de manera intuitiva, aunque no en tiempo real. Tableau permitirá explorar patrones históricos con filtros y vistas personalizables.

Indicadores Clave del Análisis

- **Demanda por Mes y Estación del Año**
  Permitirá a EasyJet ajustar su oferta según el comportamiento de los pasajeros en diferentes épocas del año.
- **Días y Horarios de Mayor Actividad**
  Mostraremos patrones de vuelos por día de la semana y horario para ayudar a planificar la programación de vuelos en horarios de alta demanda.
- **Análisis por Provincia**
  Identificaremos las provincias con mayor movimiento aéreo para facilitar la priorización de destinos y la definición de rutas.

Tecnología y Herramientas Utilizadas

- **ETL para Procesamiento de Datos**
  Implementaremos un proceso de ETL para la integración y transformación de datos, asegurando la consistencia en el análisis.
- **Tableau para Visualización**
  Dado que somos agnósticos a la tecnología decidimos utilizar tableau,herramienta BI propuesta por EasyJet . Este tablero permitirá explorar los datos históricos con gráficos intuitivos y filtros personalizables, facilitando el análisis de rutas, demanda y tendencias.

Beneficios de la Solución

- **Acceso a Información Estratégica**
  El tablero en Tableau proporciona una vista clara y estructurada del mercado de vuelos internos en Argentina, permitiendo un análisis detallado de la demanda histórica.
- **Soporte para la Planificación y Optimización de Rutas**
  La identificación de patrones de demanda y tráfico facilita la optimización de rutas y la planificación de vuelos en los horarios de mayor actividad.
- **Flexibilidad y Accesibilidad de los Datos**
  La interfaz interactiva de Tableau permite a los usuarios acceder y analizar los datos de manera flexible, generando reportes para distintas necesidades empresariales.

Obtención y Descripción de los Datos

La fuente de datos utilizada en este análisis de vuelos en Argentina proviene de la plataforma de datos abiertos del gobierno argentino, específicamente de la página [Datos.gob.ar](https://www.datos.gob.ar/dataset/transporte-lista-aeropuertos/archivo/transporte_eb54e49e-9a5a-4614-91f4-526c650d0105), que proporciona información detallada sobre los aeropuertos y los vuelos en el país.

Limpieza y Preprocesamiento de Datos

La limpieza de datos se realizó para los años 2019 a 2023, con el fin de estructurar y estandarizar la información en un formato consistente. Este proceso incluye la unificación de nombres de columnas, la corrección de formatos de fechas y horas, la eliminación de registros nulos o redundantes y la conversión de ciertas variables al formato numérico adecuado. Además, se ajustaron las zonas horarias a UTC-3 (Hora de Argentina) para asegurar la coherencia temporal de los registros y facilitar el análisis.

Paso 1: Importación y Configuración Inicial

1. **Carga de archivos CSV**:
    - Se importa la biblioteca pandas como pd para trabajar con los archivos CSV y manipular los datos en DataFrames.
    - Cada archivo CSV, de los años 2019 a 2023, se carga en un DataFrame específico utilizando pd.read_csv(), definiendo sep=';' o sep=',' según el delimitador, y low_memory=False para optimizar el uso de memoria.

```
data_2019 = pd.read_csv('2019.csv' ,sep=';', low_memory=False )
# Cargar el CSV
data_2020 = pd.read_csv('2020.csv' ,sep=';', low_memory=False )
# Cargar el CSV
data_2021 = pd.read_csv('2021.csv' ,sep=',', low_memory=False )
# Cargar el CSV
data_2022 = pd.read_csv('2022.csv' ,sep=';' , low_memory=False)
# Cargar el CSV
data_2023 = pd.read_csv('2023.csv' ,sep=';', low_memory=False )
```

Paso 2: Verificación de Carga y Columnas

- Para confirmar la correcta carga de datos, se imprimen las primeras cinco filas de cada DataFrame usando .head(5).
- También se imprimen los nombres de las columnas en cada DataFrame para analizar posibles discrepancias entre los archivos.

```
#verifico que se carguen bien los dataframe
print(data_2019.head(5))
print(data_2020.head(5))
print(data_2021.head(5))
print(data_2022.head(5))
print(data_2023.head(5))

#verifico columnas de dataframe
print(data_2019.columns)
print(data_2020.columns)
print(data_2021.columns)
print(data_2022.columns)
print(data_2023.columns)
```

Paso 3: Renombrar Columnas y Consolidación de Datos

1. **Estandarización de nombres de columnas**:
    - Dado que los nombres de columnas varían ligeramente entre archivos, se define un diccionario column_mapping para unificar estos nombres.

}

2. **Aplicación de renombrado**:
    - Cada DataFrame aplica rename() para estandarizar sus nombres de columnas según el diccionario. Luego, se imprime para verificar la estandarización.
3. **Unión de DataFrames**:
    - Se concatenan todos los DataFrames en uno solo (datos_combinados) usando pd.concat() y ignore_index=True para reasignar índices.

```
"""LOS ARCHIVOS POSEEN LAS MISMAS COLUMNAS CON VARIANTES DE NOMBRES ,
PERO TODAS RESPETAN LA POSICION DE LOS DATOS DONDE SE ENCUENTRAN .
SE VA A REALIZAR UN APPEND PARA TENER TODOS LOS DATOS EN UN DATAFRAME Y
EN UN CSV
"""

# Diccionario de mapeo para renombrar las columnas
column_mapping = {
    'Fecha': 'Fecha_utc',
    'Hora UTC': 'Hora_utc',
    'Clase de vuelos (todos los vuelos)': 'Clase_vuelos',
    'Clasificacion Vuelo': 'Clasificacion_vuelo',
```

```
    'Tipo Movimiento': 'Tipo_movimiento',
    'Tipo de Movimiento': 'Tipo_movimiento',
    'Aeropuerto': 'Aeropuerto',
    'Origen/Destino': 'Origen_destino',
    'Aerolinea Nombre': 'Aerolinea_nombre',
    'Aeronave': 'Aeronave',
    'Pasajeros': 'Pasajeros',
    'PAX': 'Pax',
    'Calidad del dato': 'Calidad_del_dato',
    'Fecha UTC': 'Fecha_utc',
    'Origen / Destino': 'Origen_destino',
    'Calidad dato': 'Calidad_del_dato',
    'Clasificación Vuelo': 'Clasificacion_vuelo',
    'Clase de Vuelo (todos los vuelos)': 'Clase_vuelos'


}

print ("columnas 2019")
data_2019.rename(columns=column_mapping, inplace=True)
print(data_2019.columns)

print ("columnas 2020")
data_2020.rename(columns=column_mapping, inplace=True)
print(data_2020.columns)


print ("columnas 2021")
data_2021.rename(columns=column_mapping, inplace=True)
print(data_2021.columns)


print ("columnas 2022")
data_2022.rename(columns=column_mapping, inplace=True)
print(data_2022.columns)

print ("columnas 2023")
data_2023.rename(columns=column_mapping, inplace=True)
print(data_2023.columns)

# Hacer el append de todos los DataFrames
datos_combinados = pd.concat([data_2019,data_2020, data_2021,
data_2022,data_2023], ignore_index=True)


print(datos_combinados.head(5))
print(datos_combinados.columns)
```

Paso 5: Normalización de Fechas y Tiempos

1. **Corrección de horas**:

- Se estandariza el formato HH:MM para la columna Hora_utc, eliminando los segundos si están presentes.

2. **Combinación de Fecha y Hora**:
   - Fecha_utc y Hora_utc se combinan en una columna FechaHora_utc de tipo datetime en UTC, lo cual facilita la conversión a zonas horarias específicas.
3. **Conversión de UTC a UTC-3**:
   - Para adaptar los datos a la zona horaria de Argentina, se convierte FechaHora_utc a FechaHora_arg.
4. **Separación de Fecha y Hora**:
   - FechaHora_arg se descompone en Fecha_arg y Hora_arg, estandarizadas sin segundos.

```
"""NORMALIZAR CAMPO FECHA YA QUE SE ENCUENTRA EN UTC . NECESITAMOS PASARLO A UTC-3 QUE ES EL QUE SE MANEJA EN ARGENTINA PARA LUEGO HACER ANALISIS"""

# Paso 1: Asegurarse de que todas las horas tengan formato HH:MM (remover segundos si están presentes)
datos_combinados['Hora_utc'] = datos_combinados['Hora_utc'].apply(lambda x: x[:5] if len(x) >= 5 else x)

# Paso 2: Combinar Fecha y Hora en un solo campo de tipo datetime en UTC
datos_combinados['FechaHora_utc'] = pd.to_datetime(datos_combinados['Fecha_utc'] + ' ' + datos_combinados['Hora_utc'], utc=True, dayfirst=True, errors='coerce')

# Paso 3: Convertir de UTC a UTC-3 (Hora de Argentina)
datos_combinados['FechaHora_arg'] = datos_combinados['FechaHora_utc'].dt.tz_convert('America/Argentina/Buenos_Aires')

# Paso 4: Separar en Fecha y Hora en la zona horaria de Argentina, sin segundos
datos_combinados['Fecha_arg'] = datos_combinados['FechaHora_arg'].dt.strftime('%d/%m/%Y')  # Formato dd/mm/yyyy
datos_combinados['Hora_arg'] = datos_combinados['FechaHora_arg'].dt.strftime('%H:%M')  # Formato sin segundos

print(datos_combinados[['Fecha_utc', 'Hora_utc', 'FechaHora_utc', 'FechaHora_arg', 'Fecha_arg', 'Hora_arg']])
```

Paso 6: Estandarización de Datos para Análisis

1. **Estandarización de valores de columnas**:
    - Se identifican y mapean valores únicos en las columnas relevantes (Clase_vuelos y Clasificacion_vuelo) para mantener consistencia.
2. **Conversión de valores a mayúsculas**:
    - Las columnas clave se convierten a mayúsculas (str.upper()), eliminando posibles diferencias por capitalización.
3. **Conversión de datos numéricos**:
    - La columna Pasajeros se convierte a tipo numérico mediante pd.to_numeric(), asignando NaN a valores no válidos y reemplazando NaN con 0 para simplificar cálculos posteriores.

```
"""SIGUIENTE PASO ESTANDARIZAR Y CONVERTIR DATOS DE LOS CAMPOS QUE SE VAN A UTILIZAR PARA HACER ANALISIS"""

# Paso 1: Obtener los valores únicos
print("Valores únicos en 'clase_vuelos':", datos_combinados['Clase_vuelos'].unique())

# Paso 2: Definir un mapeo de estandarización
clase_vuelos_mapping = {
    'REGULAR': 'REGULAR',
    'Regular': 'Regular',

    'VUELOS PRIVADOS NACIONALES': 'VUELOS PRIVADOS NACIONALES',
    'Vuelo Privado con Matrícula Nacional': 'VUELOS PRIVADOS NACIONALES',

    'VUELOS OFICIALES NACIONALES': 'VUELOS OFICIALES NACIONALES',
    'Vuelo Oficial Nacional': 'VUELOS OFICIALES NACIONALES',

    'NO REGULAR': 'REGULAR',
    'No Regular': 'NO REGULAR',

    'VUELOS PRIVADO CON MATRICULA EXTRANJERA': 'VUELOS PRIVADO CON MATRICULA EXTRANJERA',
    'Vuelo Privado con Matrícula Extranjera': 'VUELOS PRIVADO CON MATRICULA EXTRANJERA',

    'VUELOS DE ADIESTRAMIENTO': 'VUELOS DE ADIESTRAMIENTO',
    'Vuelo de Adiestramiento': 'VUELOS DE ADIESTRAMIENTO',

    'VUELOS ESCUELA': 'VUELOS ESCUELA',
    'Vuelo Escuela': 'VUELOS ESCUELA',

    'VUELOS OFICIALES EXTRANJEROS': 'VUELOS OFICIALES EXTRANJEROS',
    'Vuelo Oficial Extranjero': 'VUELOS OFICIALES EXTRANJEROS',

    'TRABAJO AEREO': 'TRABAJO AEREO',
    'Trabajo Aéreo': 'TRABAJO AEREO',
```

```
    'ESCUELA (NO VIGENTE)': 'ESCUELA (NO VIGENTE)'
}

# Paso 3: Aplicar el mapeo a la columna
datos_combinados['Clase_vuelos'] = datos_combinados['Clase_vuelos'].replace(clase_vuelos_mapping)

# Paso 4: Convertir a mayúsculas (en caso de que no se haya hecho)
datos_combinados['Clase_vuelos'] = datos_combinados['Clase_vuelos'].str.upper()

# Mostrar el DataFrame estandarizado
#print("\nDataFrame estandarizado:\n", datos_combinados)
print("Valores únicos en 'clase_vuelos':", datos_combinados['Clase_vuelos'].unique())

#clasificacion vuelos
print("Valores únicos en 'Clasificacion_vuelo':", datos_combinados['Clasificacion_vuelo'].unique())

# Por ejemplo para 'clasificacion_vuelos'
clasificacion_mapping = {
    'Doméstico': 'DOMESTICO',
    'Internacional': 'INTERNACIONAL',  # Corrigiendo acento
    'Dom': 'DOMESTICO',
    'Inter': 'INTERNACIONAL'
}

datos_combinados['Clasificacion_vuelo'] = datos_combinados['Clasificacion_vuelo'].replace(clasificacion_mapping).str.upper()

datos_combinados['Clasificacion_vuelo'] = datos_combinados['Clasificacion_vuelo'].str.upper()
print("Valores únicos en 'Clasificacion_vuelo':", datos_combinados['Clasificacion_vuelo'].unique())



#pasar a mayuscula

datos_combinados['Tipo_movimiento'] = datos_combinados['Tipo_movimiento'].str.upper()
print("Valores únicos en 'Tipo_movimiento':", datos_combinados['Tipo_movimiento'].unique())

#hay 3 registros que estan en nan nomas

# Mostrar el DataFrame final
```

```
#print("\nDataFrame final:\n", datos_combinados)

"""PARSEAR A NUMERICO EL CAMPO NUMERICO"""

# Utiliza errors='coerce' para convertir valores no válidos en NaN
datos_combinados['Pasajeros'] =
   pd.to_numeric(datos_combinados['Pasajeros'], errors='coerce')

# Si deseas convertir NaN a 0 (o a otro valor), puedes usar fillna
datos_combinados['Pasajeros'] =
   datos_combinados['Pasajeros'].fillna(0).astype(int)

"""MOSTRAR TODOS LOS TIPOS DE DATOS DE LOS CAMPOS"""

# También puedes mostrar el tipo de dato de todo el DataFrame
tipos_datos_completos = datos_combinados.dtypes

# Imprimir los resultados

print("\nTipos de datos de todas las columnas:\n",
   tipos_datos_completos)
```

Paso 7: Ajustes Finales en Formato de Columnas

1. **Conversión de fechas y horas**:
    - La columna Fecha_utc se formatea a DD/MM/AAAA, y Hora_utc a HH:MM. Lo mismo aplica a Fecha_arg.
2. **Estandarización de campos adicionales**:
    - Columnas como Aeropuerto, Origen_destino, Aerolinea_nombre, Aeronave, Pax, y Calidad_del_dato también se estandarizan eliminando espacios en blanco y aplicando mayúsculas.

```
 Limpiar espacios en blanco
datos_combinados['Fecha_utc'] = datos_combinados['Fecha_utc'].str.strip()
datos_combinados['Hora_utc'] = datos_combinados['Hora_utc'].str.strip()
datos_combinados['Fecha_arg'] = datos_combinados['Fecha_arg'].str.strip()
# Paso 1: Convertir Fecha_utc a formato DD/MM/AAAA
datos_combinados['Fecha_utc'] =
pd.to_datetime(datos_combinados['Fecha_utc'], dayfirst=True,
errors='coerce').dt.strftime('%d/%m/%Y')

# Paso 2: Convertir Hora_utc a formato HH:MM
datos_combinados['Hora_utc'] =
pd.to_datetime(datos_combinados['Hora_utc'], format='%H:%M',
errors='coerce').dt.strftime('%H:%M')
```

```python
# Paso 4: Convertir Fecha_arg a formato DD/MM/AAAA
datos_combinados['Fecha_arg'] =
pd.to_datetime(datos_combinados['Fecha_arg'], errors='coerce',
dayfirst=True)
datos_combinados['Fecha_arg'] =
datos_combinados['Fecha_arg'].dt.strftime('%d/%m/%Y')

# Mostrar el tipo de dato actualizado
print(datos_combinados.dtypes)

"""CONVERTIR TIPOS DE DATOS DE LOS CAMPOS"""

# Estandarización de la columna 'Clasificacion_vuelo'
datos_combinados['Clasificacion_vuelo'] =
datos_combinados['Clasificacion_vuelo'].str.strip().str.upper()

# Estandarización de la columna 'Tipo_movimiento'
datos_combinados['Tipo_movimiento'] =
datos_combinados['Tipo_movimiento'].str.strip().str.upper()

# Estandarización de la columna 'Aeropuerto'
datos_combinados['Aeropuerto'] =
datos_combinados['Aeropuerto'].str.strip().str.upper()

# Estandarización de la columna 'Origen_destino'
datos_combinados['Origen_destino'] =
datos_combinados['Origen_destino'].str.strip().str.upper()

# Estandarización de la columna 'Aerolinea_nombre'
datos_combinados['Aerolinea_nombre'] =
datos_combinados['Aerolinea_nombre'].str.strip().str.upper()

# Estandarización de la columna 'Aeronave'
datos_combinados['Aeronave'] =
datos_combinados['Aeronave'].str.strip().str.upper()

# Estandarización de la columna 'Pax'
datos_combinados['Pax'] = datos_combinados['Pax'].str.strip().str.upper()

# Estandarización de la columna 'Calidad_del_dato'
datos_combinados['Calidad_del_dato'] =
datos_combinados['Calidad_del_dato'].str.strip().str.upper()


# Verificar los tipos de datos
print(datos_combinados.dtypes)

print(datos_combinados.head())

print("Valores únicos en 'Aerolinea_nombre':",
datos_combinados['Aerolinea_nombre'].unique())
```

```
"""Paso 1: Preparar los datos
Primero, asegúrate de que las columnas de fecha y hora estén en el
formato correcto. Luego, extrae los días de la semana y las horas de tus
datos. Finalmente, organiza los datos para el gráfico.
"""

!pip install babel


from babel.dates import format_datetime


# Asegúrate de que 'Fecha_arg' sea un objeto datetime
datos_combinados['Fecha_arg'] =
pd.to_datetime(datos_combinados['Fecha_arg'], format='%d/%m/%Y')

# Crear una columna con el nombre del día en español
datos_combinados['Dia'] = datos_combinados['Fecha_arg'].apply(lambda x:
format_datetime(x, 'EEEE', locale='es_ES'))

# Extraer el año para cada registro
datos_combinados['Anio'] = datos_combinados['Fecha_arg'].dt.year

# Crear una tabla de conteo para las horas y días
conteo = datos_combinados.groupby(['Anio', 'Dia',
'Hora_arg']).size().reset_index(name='Conteo')

print (datos_combinados.head())
```

Paso 8: Filtrado de registros

1. **Filtrado de registros**:
    - Se filtran los registros para conservar únicamente vuelos regulares y aquellos con más de 0 pasajeros.

```
2. # Mostrar la cantidad de registros (filas)
3. print("Cantidad de registros:", datos_combinados.shape[0])
4. #Cantidad de registros: 2155873
```

```
5.
6.
7. #Filtrar los registros donde Clase_vuelos sea igual a 'REGULAR'
8. datos_combinados =
   datos_combinados[datos_combinados['Clase_vuelos'] == 'REGULAR']
9.
10.print("Cantidad de registros:", datos_combinados.shape[0])
11.# Cantidad de registros: 1190322
12.
13.# Filtrar los registros donde Pasajeros no sea igual a 0
14.datos_combinados = datos_combinados[datos_combinados['Pasajeros']
   != 0]
15.
16.print("Cantidad de registros:", datos_combinados.shape[0])
17.# Cantidad de registros: 1153281
```

Paso 9: Exportación del DataFrame Final

- El DataFrame consolidado y limpio se exporta a un archivo vuelos_bd_total.csv para su uso en análisis futuros.

```
# Exporta a CSV con punto y coma como delimitador
datos_combinados.to_csv('vuelos_bd_total.csv', sep=';', index=False,
encoding='utf-8')
```

Conclusión

Este script es un ejemplo completo de preparación de datos, que abarca desde la consolidación y estandarización hasta la limpieza y exportación, preparando los datos de vuelos para un análisis eficiente en ciencia de datos.

LEVANTO CSV A DATAFRAME SEPARADOS POR PUNTO Y COMA

```python
#IMPORTO LIBRERIA PANDAS
import pandas as pd

# Cargar el CSV
# low_memory: Esto permite a pandas usar más memoria al procesar el archivo
data_2019 = pd.read_csv('2019.csv' ,sep=';', low_memory=False )
# Cargar el CSV
data_2020 = pd.read_csv('2020.csv' ,sep=';', low_memory=False )
# Cargar el CSV
data_2021 = pd.read_csv('2021.csv' ,sep=',', low_memory=False )
# Cargar el CSV
data_2022 = pd.read_csv('2022.csv' ,sep=';' , low_memory=False)
# Cargar el CSV
data_2023 = pd.read_csv('2023.csv' ,sep=';', low_memory=False )


#verifico que se carguen bien los dataframe
print(data_2019.head(5))
print(data_2020.head(5))
print(data_2021.head(5))
print(data_2022.head(5))
print(data_2023.head(5))

#verifico columnas de dataframe
print(data_2019.columns)
print(data_2020.columns)
print(data_2021.columns)
print(data_2022.columns)
print(data_2023.columns)
```

```
      Fecha  Hora UTC Clase de vuelos (todos los vuelos) Clasificacion Vuelo  \
0  1/1/2019  00:01:00                            REGULAR           Doméstico
1  1/1/2019  00:01:00                            REGULAR       Internacional
2  1/1/2019  00:03:00                            REGULAR           Doméstico
3  1/1/2019  00:04:00                            REGULAR       Internacional
4  1/1/2019  00:06:00                            REGULAR       Internacional
```

```
  Tipo Movimiento Aeropuerto Origen/Destino  \
0      Aterrizaje        EZE            SAL
1      Aterrizaje        EZE           SBGL
2      Aterrizaje        AER            SIS
3      Aterrizaje        EZE           SBGR
4      Aterrizaje        AER           SBGR

                          Aerolinea Nombre          Aeronave  Pasajeros  PAX  \
0                 AEROLINEAS ARGENTINAS SA      BO-B-737-76N         88   44
1           TRANSPORTES AEREOS DEL MERCOSUR               NaN        165  165
2  AUSTRAL LINEAS AEREAS-CIELOS DEL SUR S.A  EMB-ERJ190100IGW         22   11
3           TRANSPORTES AEREOS DEL MERCOSUR               NaN         69   69
4       LAN ARGENTINA S.A. (LATAM AIRLINES)     AIB-A-320-233         53   53

  Calidad del dato
0       DEFINITIVO
1       DEFINITIVO
2       DEFINITIVO
3       DEFINITIVO
4       DEFINITIVO
     Fecha Hora UTC Clase de Vuelo (todos los vuelos) Clasificación Vuelo  \
0  1/1/2020    00:06                          Regular       Internacional
1  1/1/2020    00:08                          Regular       Internacional
2  1/1/2020    00:10                          Regular           Doméstico
3  1/1/2020    00:13                          Regular       Internacional
4  1/1/2020    00:13                          Regular           Doméstico

  Tipo de Movimiento Aeropuerto Origen / Destino  \
0         Aterrizaje        EZE             LEMD
1           Despegue        EZE             SCEL
2         Aterrizaje        PAL              BAR
3           Despegue        EZE             KDFW
4         Aterrizaje        PAL              DOZ

                     Aerolinea Nombre Aeronave  Pasajeros  PAX Calidad dato
0     IBERIA - LINEAS AÉREAS DE ESPAÑA        0        239  239   DEFINITIVO
1  LAN ARGENTINA S.A. (LATAM AIRLINES)        0        152  152   DEFINITIVO
2               JETSMART AIRLINES S.A.        0        116   58   DEFINITIVO
3               AMERICAN AIRLINES INC.        0        255  255   DEFINITIVO
4               JETSMART AIRLINES S.A.        0        146   73   DEFINITIVO
       Fecha Hora UTC     Clase de Vuelo (todos los vuelos)  \
0  01/01/2021    00:02  Vuelo Privado con Matrícula Nacional
1  01/01/2021    00:24                               Regular
2  01/01/2021    00:26                               Regular
3  01/01/2021    00:29                               Regular
4  01/01/2021    00:37                               Regular

  Clasificación Vuelo Tipo de Movimiento Aeropuerto Origen / Destino  \
0           Doméstico           Despegue        PAR              ROS
1           Doméstico         Aterrizaje        EZE              GRA
2           Doméstico         Aterrizaje        EZE              ECA
3           Doméstico         Aterrizaje        EZE              SAL
4           Doméstico         Aterrizaje        EZE              TUC
```

```
           Aerolinea Nombre          Aeronave  Pasajeros  PAX Calidad dato
0                         0      PA-PA-28-181          0    0   DEFINITIVO
1  AEROLINEAS ARGENTINAS SA       BO-B737-8MB        140   70   DEFINITIVO
2  AEROLINEAS ARGENTINAS SA        BO-737-800        140   70   DEFINITIVO
3  AEROLINEAS ARGENTINAS SA      BO-B-737-76N         24   12   DEFINITIVO
4  AEROLINEAS ARGENTINAS SA  EMB-ERJ190100IGW         52   26   DEFINITIVO
    Fecha UTC Hora UTC Clase de Vuelo (todos los vuelos) Clasificación Vuelo  \
0  01/01/2022    00:01                            Regular           Doméstico
1  01/01/2022    00:05                            Regular           Doméstico
2  01/01/2022    00:05                            Regular           Doméstico
3  01/01/2022    00:09                            Regular           Doméstico
4  01/01/2022    00:09                            Regular       Internacional

  Tipo de Movimiento Aeropuerto Origen / Destino          Aerolinea Nombre  \
0         Aterrizaje        AER              ECA  AEROLINEAS ARGENTINAS SA
1         Aterrizaje        AER              SAL  AEROLINEAS ARGENTINAS SA
2           Despegue        IGU              AER    JETSMART AIRLINES S.A.
3         Aterrizaje        AER              GAL  AEROLINEAS ARGENTINAS SA
4           Despegue        EZE             KDFW    AMERICAN AIRLINES INC.

       Aeronave  Pasajeros  PAX Calidad dato
0    BO-737-8SH        138   69   DEFINITIVO
1     BO-B737-8        129   65   DEFINITIVO
2  AIB-A320-232         82   41   DEFINITIVO
3   BO-B737-81D        145   73   DEFINITIVO
4             0        261  261   DEFINITIVO
    Fecha UTC Hora UTC Clase de Vuelo (todos los vuelos) Clasificación Vuelo  \
0  01/01/2023     0:01                            Regular       Internacional
1  01/01/2023     0:10                            Regular           Doméstico
2  01/01/2023     0:15                            Regular       Internacional
3  01/01/2023     0:17                            Regular           Doméstico
4  01/01/2023     0:19                            Regular           Doméstico

  Tipo de Movimiento Aeropuerto Origen / Destino  \
0         Aterrizaje        EZE             LEMD
1           Despegue        IGU              EZE
2         Aterrizaje        AER             SBPA
3         Aterrizaje        SAL              AER
4         Aterrizaje        AER              OSA

                   Aerolinea Nombre          Aeronave  Pasajeros  PAX  \
0  IBERIA - LINEAS AÉREAS DE ESPAÑA                 0        199  199
1            JETSMART AIRLINES S.A.      AIB-A320-232         67   34
2          AEROLINEAS ARGENTINAS SA      BO-B-737-76N         36   36
3            JETSMART AIRLINES S.A.      AIB-A320-232        168   84
4          AEROLINEAS ARGENTINAS SA  EMB-ERJ190100IGW         17    9
```

```
  Calidad dato
0   DEFINITIVO
1   DEFINITIVO
2   DEFINITIVO
3   DEFINITIVO
4   DEFINITIVO
Index(['Fecha', 'Hora UTC', 'Clase de vuelos (todos los vuelos)',
       'Clasificacion Vuelo', 'Tipo Movimiento', 'Aeropuerto',
       'Origen/Destino', 'Aerolinea Nombre', 'Aeronave', 'Pasajeros', 'PAX',
       'Calidad del dato'],
      dtype='object')
Index(['Fecha', 'Hora UTC', 'Clase de Vuelo (todos los vuelos)',
       'Clasificación Vuelo', 'Tipo de Movimiento', 'Aeropuerto',
       'Origen / Destino', 'Aerolinea Nombre', 'Aeronave', 'Pasajeros', 'PAX',
       'Calidad dato'],
      dtype='object')
Index(['Fecha', 'Hora UTC', 'Clase de Vuelo (todos los vuelos)',
       'Clasificación Vuelo', 'Tipo de Movimiento', 'Aeropuerto',
       'Origen / Destino', 'Aerolinea Nombre', 'Aeronave', 'Pasajeros', 'PAX',
       'Calidad dato'],
      dtype='object')
Index(['Fecha UTC', 'Hora UTC', 'Clase de Vuelo (todos los vuelos)',
       'Clasificación Vuelo', 'Tipo de Movimiento', 'Aeropuerto',
       'Origen / Destino', 'Aerolinea Nombre', 'Aeronave', 'Pasajeros', 'PAX',
       'Calidad dato'],
      dtype='object')
Index(['Fecha UTC', 'Hora UTC', 'Clase de Vuelo (todos los vuelos)',
       'Clasificación Vuelo', 'Tipo de Movimiento', 'Aeropuerto',
       'Origen / Destino', 'Aerolinea Nombre', 'Aeronave', 'Pasajeros', 'PAX',
       'Calidad dato'],
      dtype='object')
```

LOS ARCHIVOS POSEEN LAS MISMAS COLUMNAS CON VARIANTES DE NOMBRES , PERO TODAS RESPETAN LA POSICION DE LOS DATOS DONDE SE ENCUENTRAN . SE VA A REALIZAR UN APPEND PARA TENER TODOS LOS DATOS EN UN DATAFRAME Y EN UN CSV

```python
# Diccionario de mapeo para renombrar las columnas
column_mapping = {
    'Fecha': 'Fecha_utc',
    'Hora UTC': 'Hora_utc',
    'Clase de vuelos (todos los vuelos)': 'Clase_vuelos',
    'Clasificacion Vuelo': 'Clasificacion_vuelo',
    'Tipo Movimiento': 'Tipo_movimiento',
    'Tipo de Movimiento': 'Tipo_movimiento',
    'Aeropuerto': 'Aeropuerto',
    'Origen/Destino': 'Origen_destino',
    'Aerolinea Nombre': 'Aerolinea_nombre',
    'Aeronave': 'Aeronave',
    'Pasajeros': 'Pasajeros',
    'PAX': 'Pax',
    'Calidad del dato': 'Calidad_del_dato',
    'Fecha UTC': 'Fecha_utc',
    'Origen / Destino': 'Origen_destino',
    'Calidad dato': 'Calidad_del_dato',
    'Clasificación Vuelo': 'Clasificacion_vuelo',
    'Clase de Vuelo (todos los vuelos)': 'Clase_vuelos'


}

print ("columnas 2019")
data_2019.rename(columns=column_mapping, inplace=True)
print(data_2019.columns)

print ("columnas 2020")
data_2020.rename(columns=column_mapping, inplace=True)
print(data_2020.columns)


print ("columnas 2021")
data_2021.rename(columns=column_mapping, inplace=True)
print(data_2021.columns)


print ("columnas 2022")
data_2022.rename(columns=column_mapping, inplace=True)
print(data_2022.columns)

print ("columnas 2023")
data_2023.rename(columns=column_mapping, inplace=True)
print(data_2023.columns)
```

```
           Aeronave  Pasajeros  Pax Calidad_del_dato
0      BO-B-737-76N         88   44       DEFINITIVO
1               NaN        165  165       DEFINITIVO
2  EMB-ERJ190100IGW         22   11       DEFINITIVO
3               NaN         69   69       DEFINITIVO
4     AIB-A-320-233         53   53       DEFINITIVO
Index(['Fecha_utc', 'Hora_utc', 'Clase_vuelos', 'Clasificacion_vuelo',
       'Tipo_movimiento', 'Aeropuerto', 'Origen_destino', 'Aerolinea_nombre',
       'Aeronave', 'Pasajeros', 'Pax', 'Calidad_del_dato'],
      dtype='object')
```

AHORA QUE ESTAN LOS CAMPOS ESTANDARIZADOS Y TODO EN UN SOLO DATAFRAME SE VA A EXPORTAR A CSV DELIMITADO POR PUNTO Y COMA

NORMALIZAR CAMPO FECHA YA QUE SE ENCUENTRA EN UTC . NECESITAMOS PASARLO A UTC-3 QUE ES EL QUE SE MANEJA EN ARGENTINA PARA LUEGO HACER ANALISIS

```
# Paso 1: Asegurarse de que todas las horas tengan formato HH:MM (remover segundos si están presentes)
datos_combinados['Hora_utc'] = datos_combinados['Hora_utc'].apply(lambda x: x[:5] if len(x) >= 5 else x)

# Paso 2: Combinar Fecha y Hora en un solo campo de tipo datetime en UTC
datos_combinados['FechaHora_utc'] = pd.to_datetime(datos_combinados['Fecha_utc'] + ' ' + datos_combinados['Hora_utc'], utc=True, dayfirst=True, errors='coerce')

# Paso 3: Convertir de UTC a UTC-3 (Hora de Argentina)
datos_combinados['FechaHora_arg'] = datos_combinados['FechaHora_utc'].dt.tz_convert('America/Argentina/Buenos_Aires')

# Paso 4: Separar en Fecha y Hora en la zona horaria de Argentina, sin segundos
datos_combinados['Fecha_arg'] = datos_combinados['FechaHora_arg'].dt.strftime('%d/%m/%Y')  # Formato dd/mm/yyyy
datos_combinados['Hora_arg'] = datos_combinados['FechaHora_arg'].dt.strftime('%H:%M')  # Formato sin segundos

print(datos_combinados[['Fecha_utc', 'Hora_utc', 'FechaHora_utc', 'FechaHora_arg', 'Fecha_arg', 'Hora_arg']])
```

```
          Fecha_utc Hora_utc            FechaHora_utc  \
0          1/1/2019    00:01 2019-01-01 00:01:00+00:00
1          1/1/2019    00:01 2019-01-01 00:01:00+00:00
2          1/1/2019    00:03 2019-01-01 00:03:00+00:00
3          1/1/2019    00:04 2019-01-01 00:04:00+00:00
4          1/1/2019    00:06 2019-01-01 00:06:00+00:00
...             ...       ...                       ...
2155868  31/12/2023    23:45 2023-12-31 23:45:00+00:00
2155869  31/12/2023    23:50 2023-12-31 23:50:00+00:00
2155870  31/12/2023    23:51 2023-12-31 23:51:00+00:00
2155871  31/12/2023    23:55 2023-12-31 23:55:00+00:00
2155872  31/12/2023    23:58 2023-12-31 23:58:00+00:00

                   FechaHora_arg   Fecha_arg Hora_arg
0       2018-12-31 21:01:00-03:00  31/12/2018    21:01
1       2018-12-31 21:01:00-03:00  31/12/2018    21:01
2       2018-12-31 21:03:00-03:00  31/12/2018    21:03
3       2018-12-31 21:04:00-03:00  31/12/2018    21:04
4       2018-12-31 21:06:00-03:00  31/12/2018    21:06
...                           ...         ...      ...
2155868 2023-12-31 20:45:00-03:00  31/12/2023    20:45
2155869 2023-12-31 20:50:00-03:00  31/12/2023    20:50
2155870 2023-12-31 20:51:00-03:00  31/12/2023    20:51
2155871 2023-12-31 20:55:00-03:00  31/12/2023    20:55
2155872 2023-12-31 20:58:00-03:00  31/12/2023    20:58

[2155873 rows x 6 columns]
```

```python
# Hacer el append de todos los DataFrames
datos_combinados = pd.concat([data_2019,data_2020, data_2021, data_2022,data_2023], ignore_index=True)


print(datos_combinados.head(5))
print(datos_combinados.columns)
```

```
columnas 2019
Index(['Fecha_utc', 'Hora_utc', 'Clase_vuelos', 'Clasificacion_vuelo',
       'Tipo_movimiento', 'Aeropuerto', 'Origen_destino', 'Aerolinea_nombre',
       'Aeronave', 'Pasajeros', 'Pax', 'Calidad_del_dato'],
      dtype='object')
columnas 2020
Index(['Fecha_utc', 'Hora_utc', 'Clase_vuelos', 'Clasificacion_vuelo',
       'Tipo_movimiento', 'Aeropuerto', 'Origen_destino', 'Aerolinea_nombre',
       'Aeronave', 'Pasajeros', 'Pax', 'Calidad_del_dato'],
      dtype='object')
columnas 2021
Index(['Fecha_utc', 'Hora_utc', 'Clase_vuelos', 'Clasificacion_vuelo',
       'Tipo_movimiento', 'Aeropuerto', 'Origen_destino', 'Aerolinea_nombre',
       'Aeronave', 'Pasajeros', 'Pax', 'Calidad_del_dato'],
      dtype='object')
columnas 2022
Index(['Fecha_utc', 'Hora_utc', 'Clase_vuelos', 'Clasificacion_vuelo',
       'Tipo_movimiento', 'Aeropuerto', 'Origen_destino', 'Aerolinea_nombre',
       'Aeronave', 'Pasajeros', 'Pax', 'Calidad_del_dato'],
      dtype='object')
columnas 2023
Index(['Fecha_utc', 'Hora_utc', 'Clase_vuelos', 'Clasificacion_vuelo',
       'Tipo_movimiento', 'Aeropuerto', 'Origen_destino', 'Aerolinea_nombre',
       'Aeronave', 'Pasajeros', 'Pax', 'Calidad_del_dato'],
      dtype='object')
  Fecha_utc  Hora_utc Clase_vuelos Clasificacion_vuelo Tipo_movimiento  \
0  1/1/2019  00:01:00      REGULAR           Doméstico      Aterrizaje
1  1/1/2019  00:01:00      REGULAR       Internacional      Aterrizaje
2  1/1/2019  00:03:00      REGULAR           Doméstico      Aterrizaje
3  1/1/2019  00:04:00      REGULAR       Internacional      Aterrizaje
4  1/1/2019  00:06:00      REGULAR       Internacional      Aterrizaje

  Aeropuerto Origen_destino                       Aerolinea_nombre  \
0        EZE            SAL                  AEROLINEAS ARGENTINAS SA
1        EZE           SBGL            TRANSPORTES AEREOS DEL MERCOSUR
2        AER            SIS  AUSTRAL LINEAS AEREAS-CIELOS DEL SUR S.A
3        EZE           SBGR            TRANSPORTES AEREOS DEL MERCOSUR
4        AER           SBGR       LAN ARGENTINA S.A. (LATAM AIRLINES)
```

SIGUIENTE PASO ESTANDARIZAR Y CONVERTIR DATOS DE LOS CAMPOS QUE SE VAN A UTILIZAR PARA HACER ANALISIS

```python
# Paso 1: Obtener los valores únicos
print("Valores únicos en 'clase_vuelos':", datos_combinados['Clase_vuelos'].unique())

# Paso 2: Definir un mapeo de estandarización
clase_vuelos_mapping = {
    'REGULAR': 'REGULAR',
    'Regular': 'Regular',

    'VUELOS PRIVADOS NACIONALES': 'VUELOS PRIVADOS NACIONALES',
    'Vuelo Privado con Matrícula Nacional': 'VUELOS PRIVADOS NACIONALES',

    'VUELOS OFICIALES NACIONALES': 'VUELOS OFICIALES NACIONALES',
    'Vuelo Oficial Nacional': 'VUELOS OFICIALES NACIONALES',

    'NO REGULAR': 'REGULAR',
    'No Regular': 'NO REGULAR',

    'VUELOS PRIVADO CON MATRICULA EXTRANJERA': 'VUELOS PRIVADO CON MATRICULA EXTRANJERA',
    'Vuelo Privado con Matrícula Extranjera': 'VUELOS PRIVADO CON MATRICULA EXTRANJERA',

    'VUELOS DE ADIESTRAMIENTO': 'VUELOS DE ADIESTRAMIENTO',
    'Vuelo de Adiestramiento': 'VUELOS DE ADIESTRAMIENTO',

    'VUELOS ESCUELA': 'VUELOS ESCUELA',
    'Vuelo Escuela': 'VUELOS ESCUELA',

    'VUELOS OFICIALES EXTRANJEROS': 'VUELOS OFICIALES EXTRANJEROS',
    'Vuelo Oficial Extranjero': 'VUELOS OFICIALES EXTRANJEROS',

    'TRABAJO AEREO': 'TRABAJO AEREO',
    'Trabajo Aéreo': 'TRABAJO AEREO',

    'ESCUELA (NO VIGENTE)': 'ESCUELA (NO VIGENTE)'
}



# Paso 3: Aplicar el mapeo a la columna
datos_combinados['Clase_vuelos'] = datos_combinados['Clase_vuelos'].replace(clase_vuelos_mapping)

# Paso 4: Convertir a mayúsculas (en caso de que no se haya hecho)
datos_combinados['Clase_vuelos'] = datos_combinados['Clase_vuelos'].str.upper()



# Mostrar el DataFrame estandarizado
#print("\nDataFrame estandarizado:\n", datos_combinados)
print("Valores únicos en 'clase_vuelos':", datos_combinados['Clase_vuelos'].unique())
```

```python
#clasificacion vuelos
print("Valores únicos en 'Clasificacion_vuelo':", datos_combinados['Clasificacion_vuelo'].unique())

# Por ejemplo para 'clasificacion_vuelos'
clasificacion_mapping = {
    'Doméstico': 'DOMESTICO',
    'Internacional': 'INTERNACIONAL',  # Corrigiendo acento
    'Dom': 'DOMESTICO',
    'Inter': 'INTERNACIONAL'
}

datos_combinados['Clasificacion_vuelo'] = datos_combinados['Clasificacion_vuelo'].replace(clasificacion_mapping).str.upper()

datos_combinados['Clasificacion_vuelo'] = datos_combinados['Clasificacion_vuelo'].str.upper()
print("Valores únicos en 'Clasificacion_vuelo':", datos_combinados['Clasificacion_vuelo'].unique())



#pasar a mayuscula

datos_combinados['Tipo_movimiento'] = datos_combinados['Tipo_movimiento'].str.upper()
print("Valores únicos en 'Tipo_movimiento':", datos_combinados['Tipo_movimiento'].unique())

#hay 3 registros que estan en nan nomas

# Mostrar el DataFrame final
#print("\nDataFrame final:\n", datos_combinados)
```

```
Valores únicos en 'clase_vuelos': ['REGULAR' 'VUELOS PRIVADOS NACIONALES' 'VUELOS OFICIALES NACIONALES'
 'NO REGULAR' 'VUELOS PRIVADO CON MATRICULA EXTRANJERA' 'TRABAJO AEREO'
 'VUELOS DE ADIESTRAMIENTO' 'VUELOS ESCUELA'
 'VUELOS OFICIALES EXTRANJEROS' 'ESCUELA (NO VIGENTE)' 'Regular'
 'Vuelo Privado con Matrícula Nacional' 'No Regular'
 'Vuelo Privado con Matrícula Extranjera' 'Trabajo Aéreo'
 'Vuelo de Adiestramiento' 'Vuelo Oficial Extranjero'
 'Vuelo Oficial Nacional' 'Vuelo Escuela']
Valores únicos en 'clase_vuelos': ['REGULAR' 'VUELOS PRIVADOS NACIONALES' 'VUELOS OFICIALES NACIONALES'
 'VUELOS PRIVADO CON MATRICULA EXTRANJERA' 'TRABAJO AEREO'
 'VUELOS DE ADIESTRAMIENTO' 'VUELOS ESCUELA'
 'VUELOS OFICIALES EXTRANJEROS' 'ESCUELA (NO VIGENTE)' 'NO REGULAR']
Valores únicos en 'Clasificacion_vuelo': ['Doméstico' 'Internacional' 'Dom' 'Inter']
Valores únicos en 'Clasificacion_vuelo': ['DOMESTICO' 'INTERNACIONAL']
Valores únicos en 'Tipo_movimiento': ['ATERRIZAJE' 'DESPEGUE' nan]
```

PARSEAR A NUMERICO EL CAMPO NUMERICO

```
[ ] # Utiliza errors='coerce' para convertir valores no válidos en NaN
    datos_combinados['Pasajeros'] = pd.to_numeric(datos_combinados['Pasajeros'], errors='coerce')

    # Si deseas convertir NaN a 0 (o a otro valor), puedes usar fillna
    datos_combinados['Pasajeros'] = datos_combinados['Pasajeros'].fillna(0).astype(int)
```

MOSTRAR TODOS LOS TIPOS DE DATOS DE LOS CAMPOS

```
# También puedes mostrar el tipo de dato de todo el DataFrame
tipos_datos_completos = datos_combinados.dtypes

# Imprimir los resultados

print("\nTipos de datos de todas las columnas:\n", tipos_datos_completos)
```

```
Tipos de datos de todas las columnas:
 Fecha_utc                                                 object
Hora_utc                                                   object
Clase_vuelos                                               object
Clasificacion_vuelo                                        object
Tipo_movimiento                                            object
Aeropuerto                                                 object
Origen_destino                                             object
Aerolinea_nombre                                           object
Aeronave                                                   object
Pasajeros                                                   int64
Pax                                                        object
Calidad_del_dato                                           object
FechaHora_utc                                 datetime64[ns, UTC]
FechaHora_arg         datetime64[ns, America/Argentina/Buenos_Aires]
Fecha_arg                                                  object
Hora_arg                                                   object
dtype: object
```

```
# Limpiar espacios en blanco
datos_combinados['Fecha_utc'] = datos_combinados['Fecha_utc'].str.strip()
datos_combinados['Hora_utc'] = datos_combinados['Hora_utc'].str.strip()
datos_combinados['Fecha_arg'] = datos_combinados['Fecha_arg'].str.strip()
# Paso 1: Convertir Fecha_utc a formato DD/MM/AAAA
datos_combinados['Fecha_utc'] = pd.to_datetime(datos_combinados['Fecha_utc'], dayfirst=True, errors='coerce').dt.strftime('%d/%m/%Y')

# Paso 2: Convertir Hora_utc a formato HH:MM
datos_combinados['Hora_utc'] = pd.to_datetime(datos_combinados['Hora_utc'], format='%H:%M', errors='coerce').dt.strftime('%H:%M')

# Paso 4: Convertir Fecha_arg a formato DD/MM/AAAA
datos_combinados['Fecha_arg'] = pd.to_datetime(datos_combinados['Fecha_arg'], errors='coerce', dayfirst=True)
datos_combinados['Fecha_arg'] = datos_combinados['Fecha_arg'].dt.strftime('%d/%m/%Y')

# Mostrar el tipo de dato actualizado
print(datos_combinados.dtypes)
```

```
Fecha_utc                                                 object
Hora_utc                                                  object
Clase_vuelos                                              object
Clasificacion_vuelo                                       object
Tipo_movimiento                                           object
Aeropuerto                                                object
Origen_destino                                            object
Aerolinea_nombre                                          object
Aeronave                                                  object
Pasajeros                                                  int64
Pax                                                       object
Calidad_del_dato                                          object
FechaHora_utc                                 datetime64[ns, UTC]
FechaHora_arg         datetime64[ns, America/Argentina/Buenos_Aires]
Fecha_arg                                                 object
Hora_arg                                                  object
dtype: object
```

CONVERTIR TIPOS DE DATOS DE LOS CAMPOS

```python
# Estandarización de la columna 'Clasificacion_vuelo'
datos_combinados['Clasificacion_vuelo'] = datos_combinados['Clasificacion_vuelo'].str.strip().str.upper()

# Estandarización de la columna 'Tipo_movimiento'
datos_combinados['Tipo_movimiento'] = datos_combinados['Tipo_movimiento'].str.strip().str.upper()

# Estandarización de la columna 'Aeropuerto'
datos_combinados['Aeropuerto'] = datos_combinados['Aeropuerto'].str.strip().str.upper()

# Estandarización de la columna 'Origen_destino'
datos_combinados['Origen_destino'] = datos_combinados['Origen_destino'].str.strip().str.upper()

# Estandarización de la columna 'Aerolinea_nombre'
datos_combinados['Aerolinea_nombre'] = datos_combinados['Aerolinea_nombre'].str.strip().str.upper()

# Estandarización de la columna 'Aeronave'
datos_combinados['Aeronave'] = datos_combinados['Aeronave'].str.strip().str.upper()

# Estandarización de la columna 'Pax'
datos_combinados['Pax'] = datos_combinados['Pax'].str.strip().str.upper()

# Estandarización de la columna 'Calidad_del_dato'
datos_combinados['Calidad_del_dato'] = datos_combinados['Calidad_del_dato'].str.strip().str.upper()


# Verificar los tipos de datos
print(datos_combinados.dtypes)
```

```
Fecha_utc                                                 object
Hora_utc                                                  object
Clase_vuelos                                              object
Clasificacion_vuelo                                       object
Tipo_movimiento                                           object
Aeropuerto                                                object
Origen_destino                                            object
Aerolinea_nombre                                          object
Aeronave                                                  object
Pasajeros                                                  int64
Pax                                                       object
Calidad_del_dato                                          object
FechaHora_utc                                 datetime64[ns, UTC]
FechaHora_arg          datetime64[ns, America/Argentina/Buenos_Aires]
Fecha_arg                                                 object
Hora_arg                                                  object
dtype: object
```

```
[ ] print(datos_combinados.head())
```

```
    Fecha_utc Hora_utc Clase_vuelos Clasificacion_vuelo Tipo_movimiento  \
0  01/01/2019    00:01      REGULAR           DOMESTICO      ATERRIZAJE
1  01/01/2019    00:01      REGULAR       INTERNACIONAL      ATERRIZAJE
2  01/01/2019    00:03      REGULAR           DOMESTICO      ATERRIZAJE
3  01/01/2019    00:04      REGULAR       INTERNACIONAL      ATERRIZAJE
4  01/01/2019    00:06      REGULAR       INTERNACIONAL      ATERRIZAJE

  Aeropuerto Origen_destino                       Aerolinea_nombre  \
0        EZE            SAL                 AEROLINEAS ARGENTINAS SA
1        EZE           SBGL           TRANSPORTES AEREOS DEL MERCOSUR
2        AER            SIS  AUSTRAL LINEAS AEREAS-CIELOS DEL SUR S.A
3        EZE           SBGR           TRANSPORTES AEREOS DEL MERCOSUR
4        AER           SBGR       LAN ARGENTINA S.A. (LATAM AIRLINES)

           Aeronave  Pasajeros  Pax Calidad_del_dato  \
0      BO-B-737-76N         88  NaN       DEFINITIVO
1               NaN        165  NaN       DEFINITIVO
2  EMB-ERJ190100IGW         22  NaN       DEFINITIVO
3               NaN         69  NaN       DEFINITIVO
4     AIB-A-320-233         53  NaN       DEFINITIVO

              FechaHora_utc             FechaHora_arg   Fecha_arg Hora_arg
0 2019-01-01 00:01:00+00:00 2018-12-31 21:01:00-03:00  31/12/2018    21:01
1 2019-01-01 00:01:00+00:00 2018-12-31 21:01:00-03:00  31/12/2018    21:01
2 2019-01-01 00:03:00+00:00 2018-12-31 21:03:00-03:00  31/12/2018    21:03
3 2019-01-01 00:04:00+00:00 2018-12-31 21:04:00-03:00  31/12/2018    21:04
4 2019-01-01 00:06:00+00:00 2018-12-31 21:06:00-03:00  31/12/2018    21:06
```

```
print("Valores únicos en 'Aerolinea_nombre':", datos_combinados['Aerolinea_nombre'].unique())

# Imprimir todos los valores únicos de 'Aerolinea_nombre' sin abreviación
##aerolineas_unicas = datos_combinados['Aerolinea_nombre'].unique()
####print("Valores únicos en 'Aerolinea_nombre':", list(aerolineas_unicas))



# Usar pd.Series para una mejor visualización
##print(pd.Series(aerolineas_unicas).to_string(index=False))
```

```
Valores únicos en 'Aerolinea_nombre': ['AEROLINEAS ARGENTINAS SA' 'TRANSPORTES AEREOS DEL MERCOSUR'
 'AUSTRAL LINEAS AEREAS-CIELOS DEL SUR S.A' ...
 'NELSON FERNANDES LAMARTINE NOGUEIRA' 'ABX AIR'
 'S.T.A.R. S.R.L. - BUSH PILOTS']
```

```
!pip install babel


from babel.dates import format_datetime


# Asegúrate de que 'Fecha_arg' sea un objeto datetime
datos_combinados['Fecha_arg'] = pd.to_datetime(datos_combinados['Fecha_arg'], format='%d/%m/%Y')

# Crear una columna con el nombre del día en español
datos_combinados['Dia'] = datos_combinados['Fecha_arg'].apply(lambda x: format_datetime(x, 'EEEE', locale='es_ES'))

# Extraer el año para cada registro
datos_combinados['Anio'] = datos_combinados['Fecha_arg'].dt.year

# Crear una tabla de conteo para las horas y días
conteo = datos_combinados.groupby(['Anio', 'Dia', 'Hora_arg']).size().reset_index(name='Conteo')
```

```
Requirement already satisfied: babel in /usr/local/lib/python3.10/dist-packages (2.16.0)
```

```
print (datos_combinados.head())
```

```
    Fecha_utc Hora_utc Clase_vuelos Clasificacion_vuelo Tipo_movimiento  \
0  01/01/2019    00:01      REGULAR           DOMESTICO      ATERRIZAJE
1  01/01/2019    00:01      REGULAR       INTERNACIONAL      ATERRIZAJE
2  01/01/2019    00:03      REGULAR           DOMESTICO      ATERRIZAJE
3  01/01/2019    00:04      REGULAR       INTERNACIONAL      ATERRIZAJE
4  01/01/2019    00:06      REGULAR       INTERNACIONAL      ATERRIZAJE

  Aeropuerto Origen_destino                       Aerolinea_nombre  \
0        EZE            SAL                AEROLINEAS ARGENTINAS SA
1        EZE           SBGL          TRANSPORTES AEREOS DEL MERCOSUR
2        AER            SIS  AUSTRAL LINEAS AEREAS-CIELOS DEL SUR S.A
3        EZE           SBGR          TRANSPORTES AEREOS DEL MERCOSUR
4        AER           SBGR       LAN ARGENTINA S.A. (LATAM AIRLINES)

           Aeronave  Pasajeros  Pax Calidad_del_dato  \
0      BO-B-737-76N         88  NaN       DEFINITIVO
1               NaN        165  NaN       DEFINITIVO
2  EMB-ERJ190100IGW         22  NaN       DEFINITIVO
3               NaN         69  NaN       DEFINITIVO
4     AIB-A-320-233         53  NaN       DEFINITIVO

              FechaHora_utc             FechaHora_arg  Fecha_arg Hora_arg  \
0 2019-01-01 00:01:00+00:00 2018-12-31 21:01:00-03:00 2018-12-31    21:01
1 2019-01-01 00:01:00+00:00 2018-12-31 21:01:00-03:00 2018-12-31    21:01
2 2019-01-01 00:03:00+00:00 2018-12-31 21:03:00-03:00 2018-12-31    21:03
3 2019-01-01 00:04:00+00:00 2018-12-31 21:04:00-03:00 2018-12-31    21:04
4 2019-01-01 00:06:00+00:00 2018-12-31 21:06:00-03:00 2018-12-31    21:06
```

```
     Dia  Anio
0  lunes  2018
1  lunes  2018
2  lunes  2018
3  lunes  2018
4  lunes  2018
```

```
[ ] # Mostrar la cantidad de registros (filas)
    print("Cantidad de registros:", datos_combinados.shape[0])
```

```
Cantidad de registros: 2155873
```

```
[ ] #Filtrar los registros donde Clase_vuelos sea igual a 'REGULAR'
    datos_combinados = datos_combinados[datos_combinados['Clase_vuelos'] == 'REGULAR']

    print("Cantidad de registros:", datos_combinados.shape[0])
```

```
Cantidad de registros: 1190322
```

```
[ ] # Filtrar los registros donde Pasajeros no sea igual a 0
    datos_combinados = datos_combinados[datos_combinados['Pasajeros'] != 0]

    print("Cantidad de registros:", datos_combinados.shape[0])
```

```
Cantidad de registros: 1153281
```

HACER MERGE CON ID DE DESTINO Y ID

```
[ ] # Exporta a CSV con punto y coma como delimitador
    datos_combinados.to_csv('vuelos_bd_total.csv', sep=';', index=False, encoding='utf-8')
```

Recomendaciones

1. **Aumentar frecuencia en rutas de alta demanda:**
    - **Propuesta:** Más vuelos en rutas y horarios pico (ej., Buenos Aires a provincias del norte en verano).
    - **Impacto:** Mejora la disponibilidad y sube la ocupación en momentos clave.
2. **Ajuste de precios según temporada:**
    - **Propuesta:** Precios dinámicos: subir tarifas en temporada alta y ofrecer promos en temporada baja.
    - **Impacto:** Incremento de ingresos en momentos de alta demanda y atracción de pasajeros en baja.
3. **Campañas de marketing en provincias con potencial:**
    - **Propuesta:** Enfocar publicidad en regiones de crecimiento (ej., el sur de Argentina).
    - **Impacto:** Atrae nuevos pasajeros en áreas menos competitivas.
4. **Optimizar programación en horas pico:**
    - **Propuesta:** Reforzar horarios con alta demanda y reducir vuelos en horas valle.
    - **Impacto:** Mejor uso de recursos y costos más bajos en momentos de baja ocupación.
5. **Evaluación continua de demanda:**
    - **Propuesta:** Revisar tráfico y demanda cada tres meses para ajustar rutas y precios.
    - **Impacto:** Mantenerse competitivo y responder rápido a cambios del mercado.

Conclusiones

La solución le da a EasyJet datos clave para planificar su expansión en Argentina. El análisis de demanda y tráfico permite definir rutas, precios y promos estratégicas. La visualización en Tableau hace que los datos sean fáciles de interpretar y permite decisiones rápidas.

Impacto Potencial:

- **Optimización de costos:** Reducción de hasta un 15% al ajustar la flota en horarios con menor demanda.
- **Aumento de demanda:** Incremento de hasta un 20% en temporada baja con campañas y ajustes de precio.
- **Mejora en satisfacción del cliente:** Más disponibilidad y horarios convenientes aumentan la fidelización y mejoran la experiencia.

Estas recomendaciones permiten que EasyJet optimice recursos y capte más pasajeros con una estrategia clara basada en datos.