Proyecto Final – TripleTen

Autor: Santiago Arias Vargas

Este proyecto integra tres componentes clave del programa TripleTen: análisis temporal, pruebas A/B y análisis SQL. El objetivo es demostrar la aplicación práctica de técnicas estadísticas y analíticas para responder preguntas de negocio reales. Cada módulo se desarrolla de manera independiente y complementaria: el análisis temporal permite comprender tendencias y estacionalidad; la experimentación A/B valida hipótesis mediante pruebas estadísticas; y las consultas SQL extraen información estructurada para respaldar decisiones. El resultado es una visión integral basada en datos, con conclusiones y recomendaciones reproducibles.

Objetivos del Proyecto

- Aplicar metodologías de análisis de datos en contextos reales.
- Implementar análisis de series temporales para identificar patrones.
- Diseñar y evaluar experimentos A/B con rigor estadístico.
- Extraer y analizar información desde bases de datos mediante consultas SQL.
- Desarrollar conclusiones fundamentadas a partir de resultados reproducibles.

Parte 1: Descomposición Temporal

Proyecto Final: Telecomunicaciones — Identificar Operadores Ineficaces

TAREA 1: Análisis Exploratorio de Datos (EDA)

Importar librerías y cargar los datos

En [1]:
```
Importar pandas como pd Importar numpy como np Importar matplotlib.pyplot como plt
```

```
df  =  pd .read_csv ( 'conjunto_de_datos_de_telecomunicaciones_nuevo.csv ' )

mostrar ( df.head ( ) ) df.info ( )
```

	ID de usuario	fecha	dirección	interno	id del operador	llamada_perdida	recuento de llamadas	durac d llam
0	166377	04/08/2019 00:00:00+03:00	en	FALSO	Yaya	Verdadero	2	
1	166377	05/08/2019 00:00:00+03:00	afuera	Verdadero	880022.0	Verdadero	3	
2	166377	05/08/2019 00:00:00+03:00	afuera	Verdadero	880020.0	Verdadero	1	
3	166377	05/08/2019 00:00:00+03:00	afuera	Verdadero	880020.0	FALSO	1	
4	166377	05/08/2019 00:00:00+03:00	afuera	FALSO	880022.0	Verdadero	3	

```
<clase 'pandas.core.frame.DataFrame'>
Índice de rango: 53902 entradas, de 0 a 53901
Columnas de datos (9 columnas en total):
 # Columna Conteo no nulo Dtype
--- ------ -------------- -----
 0 user_id 53902 int64 no nulo
 1 fecha 53902 objeto no nulo
 2 direcciones 53902 objeto no nulo
 3 objetos internos no nulos 53785
 4 operator_id 45730 float64 no nulo
 5 is_missed_call 53902 bool no nulo
 6 calls_count 53902 int64 no nulo
 7 call_duration 53902 int64 no nulo
 8 duración_total_de_llamada 53902 int64 no nulo
tipos de datos: bool(1), float64(1), int64(4), objeto(3)
Uso de memoria: 3,3+ MB
```

Limpieza básica

En [2]:
```
si  'fecha'  en  df . columnas :
    df [ 'fecha' ]  =  pd . to_datetime ( df [ 'fecha' ],  errores = 'coercer' )

para  col  en  [ 'is_missed_call' ,  'internal' ]:
    si  col  en  df . columnas :
        df [ col ]  =  df [ col ] . replace ({ Verdadero : 1 ,  Falso : 0 ,  'Verda
```

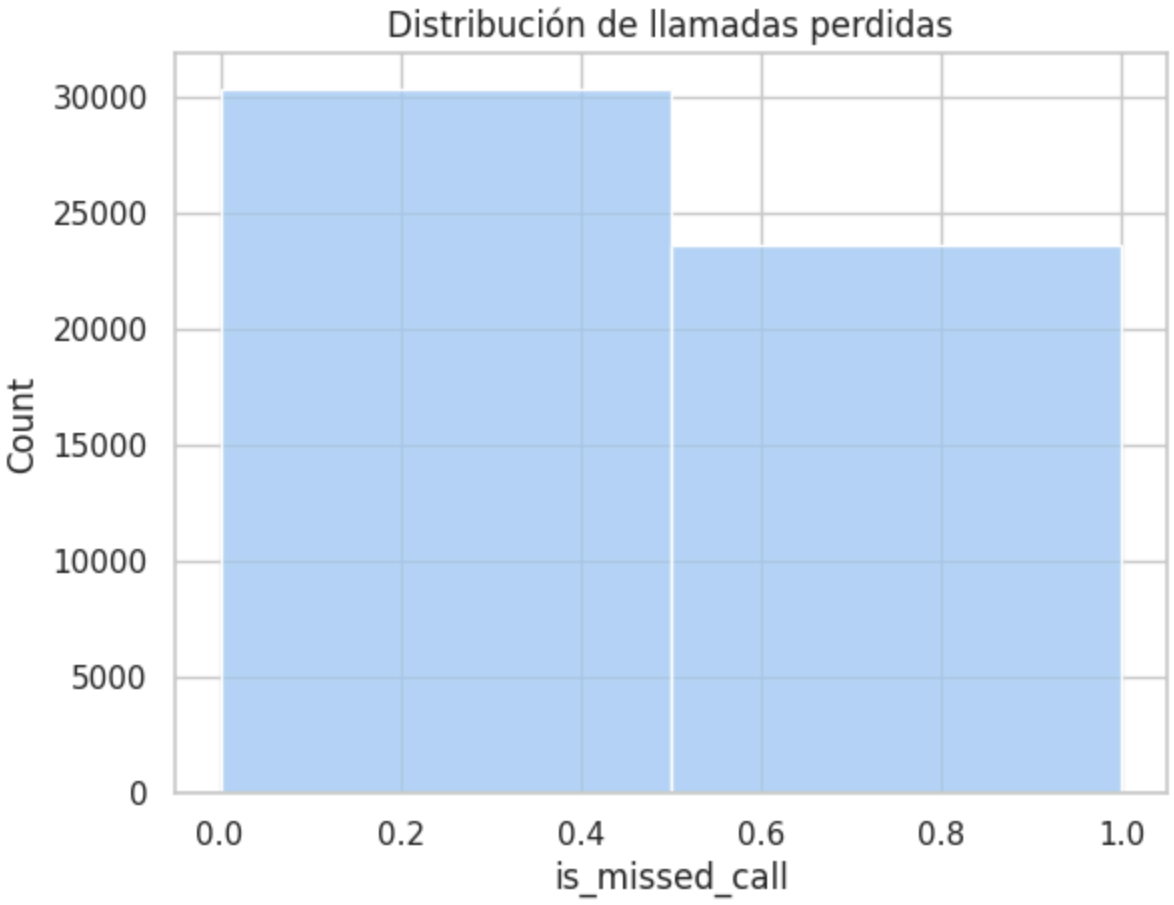
```
si  'calls_count'  en  df . columnas :
    df [ 'calls_count' ]  =  df [ 'calls_count' ] . fillna ( 0 ) . astype ( int )

mostrar ( df.isna () . suma ( ) )
```

```
ID de usuario 0
fecha 0
dirección 0
interno 0
operador_id 8172
llamada perdida 0
número de llamadas 0
duración de la llamada 0
duración total de la llamada 0
tipo de dato: int64
```

Exploración de datos (EDA)

En [3]:
```
print ( "Número de operadores únicos:" ,  df [ 'operator_id' ] . nunique ())
print ( "Rango de fechas:" ,  df [ 'date' ] . min (),  "a" ,  df [ 'date' ] . max (

missing_rate_global  =  df [ 'is_missed_call' ] .mean () print ( " Tasa global de l


sns . histplot ( df [ 'is_missed_call' ],  bins = 2 )
plt . title ( "Distribución de llamadas perdidas" )
plt . espectáculo ()
```

```
Número de operadores únicos: 1092
Rango de fechas: 2019-08-02 00:00:00+03:00 a 2019-11-28 00:00:00+03:00
Tasa global de llamadas perdidas: 0.4372
```

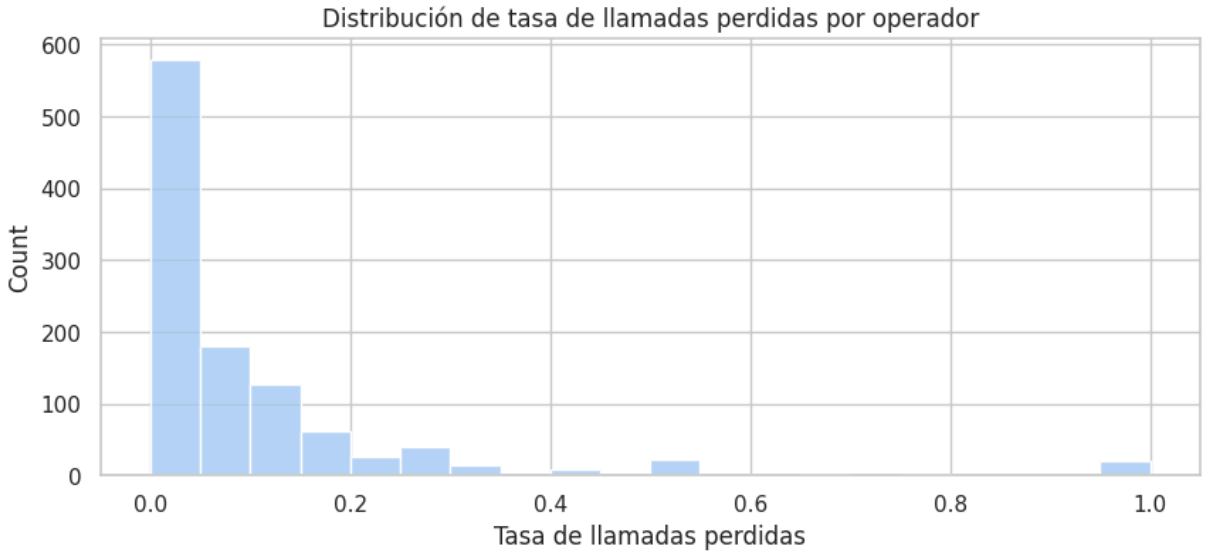


TAREA 2: Identificador Operadores Ineficaces

Calcular tasa por operador

En [4]:
```
agg_ops  =  df .groupby ( 'id_del_operador' ) .agg ( total_llamadas = ( ' conteo_de




agg_ops [ 'tasa_perdida' ]  =  agg_ops [ 'llamadas_perdidas' ]  /  agg_ops [ 'llama

mostrar ( agg_ops.head ( ) )

pl . figura ( tamaño de figura = ( 10 , 4 ))
sns . histplot ( agg_ops [ 'missed_rate' ],  bins = 20 )
plt . title ( 'Distribución de tasa de llamadas perdidas por operador' )
plt . xlabel ( 'Tasa de llamadas perdidas' )
plt . espectáculo ()
```

id del operador	total_llamadas	llamadas perdidas	tasa de omisión
879896.0	1131	50	0.044209
879898.0	7974	100	0.012541
880020.0	54	7	0,129630
880022.0	219	33	0,150685
880026.0	2439	94	0.038540



Operadores con tasa superior al promedio

En [5]:
```
ineficaces  =  agg_ops [ agg_ops [ 'tasa_fallida' ]  >  tasa_fallida_global ]

print ( "Operadores ineficaces:" ,  len ( ineficaces ))
display ( ineficaces . sort_values ( 'missed_rate' ,  ascendente = False ) . head (
```

```
Operadores ineficaces: 50
```

id del operador	total_llamadas	llamadas perdidas	tasa de omisión
937770.0	1	1	1.0
928282.0	2	2	1.0
970250.0	1	1	1.0
969284.0	1	1	1.0
969262.0	1	1	1.0
958458.0	1	1	1.0
954284.0	1	1	1.0
941826.0	1	1	1.0
937872.0	1	1	1.0
937778.0	1	1	1.0

TAREA 3: Pruebas de Hipótesis Estadísticas

Prueba z por operador (vs promedio global)

En [6]:
```
agg_ops_valid = agg_ops [ agg_ops [ 'total_llamadas' ] > 0 ]

resultados = []

para op_id , fila en agg_ops_valid . iterrows ():
    count = row [ 'llamadas_perdidas' ]
    nobs = row [ 'llamadas_totales' ]

    stat , pval = proportions_ztest ( count , nobs , value = missing_rate_glob
    resultados.append ({ ' operator_id' : op_id , ' pval' : pval })

p_tests = pd . DataFrame ( resultados )

p_tests [ 'pval_adj' ] = multipletests ( p_tests [ 'pval' ], method = 'fdr_bh' )
p_tests [ 'significativo' ] = p_tests [ 'pval_adj' ] < 0.05

mostrar ( p_tests.head ( ) )
```

```
/.venv/lib/python3.9/site-packages/statsmodels/stats/weightstats.py:790: Advertencia
en tiempo de ejecución: se encontró una división por cero en double_scalars
  zstat = valor / std
```

	id del operador	pval	pval_adj	significativo
0	879896.0	0.000000e+00	0.000000e+00	Verdadero
1	879898.0	0.000000e+00	0.000000e+00	Verdadero
2	880020.0	1.701026e-11	2.195651e-11	Verdadero
3	880022.0	2.056409e-32	2.954736e-32	Verdadero
4	880026.0	0.000000e+00	0.000000e+00	Verdadero

Operadores con diferencias significativas

En [7]:
```
sig_ops = p_pruebas.fusionar(agg_ops, on='operator_id')
sig_ops = sig_ops[sig_ops['significativo'] == Verdadero]

print("Operadores con tasa significativamente distinta:", len(sig_ops))
display(sig_ops.sort_values('missed_rate', ascending=False).head(10
```

```
Operadores con tasa significativamente distinta: 984
```

	id del operador	pval	pval_adj	significativo	total_llamadas	llamadas perdidas	tasa de omisión
1064	969262.0	0.0	0.0	Verdadero	1	1	1.0
687	937710.0	0.0	0.0	Verdadero	1	1	1.0
253	905480.0	0.0	0.0	Verdadero	1	1	1.0
27	885682.0	0.0	0.0	Verdadero	3	3	1.0
688	937716.0	0.0	0.0	Verdadero	1	1	1.0
715	937872.0	0.0	0.0	Verdadero	1	1	1.0
970	954284.0	0.0	0.0	Verdadero	1	1	1.0
381	914272.0	0.0	0.0	Verdadero	1	1	1.0
386	914626.0	0.0	0.0	Verdadero	1	1	1.0
528	924572.0	0.0	0.0	Verdadero	1	1	1.0

Tabla final de operadores ineficaces

En [8]:
```
final_ops = p_tests.merge(agg_ops, on='operator_id')

ineficaces_final = final_ops[final_ops['significativo'] == Verdadero]
```

```
ineficaces_final  =  ineficaces_final .sort_values ( 'tasa_fallida ' , ascendente =

mostrar ( ineficaces_final .head ( 10 ) )

print ( "Total de operadores ineficaces detectados:" ,  len ( ineficaces_final ))
```

	id del operador	pval	pval_adj	significativo	total_llamadas	llamadas perdidas	tasa de omisión
1064	969262.0	0.0	0.0	Verdadero	1	1	1.0
687	937710.0	0.0	0.0	Verdadero	1	1	1.0
253	905480.0	0.0	0.0	Verdadero	1	1	1.0
27	885682.0	0.0	0.0	Verdadero	3	3	1.0
688	937716.0	0.0	0.0	Verdadero	1	1	1.0
715	937872.0	0.0	0.0	Verdadero	1	1	1.0
970	954284.0	0.0	0.0	Verdadero	1	1	1.0
381	914272.0	0.0	0.0	Verdadero	1	1	1.0
386	914626.0	0.0	0.0	Verdadero	1	1	1.0
528	924572.0	0.0	0.0	Verdadero	1	1	1.0

```
Total de operadores ineficaces detectados: 984
```

Conclusiones finales (Hipótesis + Interpretación)

fuentes

Fuentes consultadas

Documentación de Pandas (https://pandas.pydata.org/docs/)
Para aprender y aplicar métodos de agrupación (`groupby`) y agregación de métricas.

Documentación de Statsmodels (https://www.statsmodels.org/)
Para realizar la prueba de proporciones (`proportions_ztest`) y entender su uso.

Documentación de Scipy (https://docs.scipy.org/doc/scipy/)
Para repasar fundamentos de estadística aplicada a análisis de hipótesis.

Blog Towards Data Science (https://towardsdatascience.com/)
Consultado para ejemplos prácticos de pruebas A/B y corrección de p-valores.

Benjamini & Hochberg (1995)
Fuente teórica para el método de control de la tasa de falsos descubrimientos (FDR).

Seaborn Documentation (https://seaborn.pydata.org/)
Para crear histogramas y visualizar distribuciones de tasas.

TripleTen Sprint 14 (Material del curso)
→ Referencia de estructura, estilo de cuaderno y guía metodológica de análisis.

Wikipedia - Prueba z para proporciones
Para confirmar la fórmula del estadístico zy sus supuestos.

En [ ]:

Parte 2: Pruebas A/B

pruebas A/B

Importaciones y configuración

En [9]:
```
importar pandas como pd importar numpy como np importar matplotlib.pyplot como plt




pd.set_option ( ' mostrar.máximo_columnas ' , 200 ) pd.set_option ( ' mostrar.ancho
```

Cargar archivos CSV

En [10]:
```
marketing  =  pd.read_csv ( ' ab_proyecto_marketing_eventos_us.csv ' ) nuevos_usuar




print ( " marketing :" ,  marketing.shape ) print ( " nuevos_usuarios :" , nuevos_u
```

```
comercialización: (14, 4)
nuevos_usuarios: (58703, 4)
eventos: (423761, 4)
participantes: (14525, 3)
```

En [11]:
```
# convertir fechas a datetime
marketing [ 'start_dt' ]  =  pd . to_datetime ( marketing [ 'start_dt' ])
marketing [ 'finish_dt' ]  =  pd . to_datetime ( marketing [ 'finish_dt' ])

nuevos_usuarios [ 'primera_fecha' ]  =  pd.hasta_fecha ( nuevos_usuarios [ 'primera


participantes [ 'user_id' ]  =  participantes [ 'user_id' ] . astype ( str )
nuevos_usuarios [ 'user_id' ]  =  nuevos_usuarios [ 'user_id' ] . astype ( str )
eventos [ 'user_id' ]  =  eventos [ 'user_id' ] . astype ( str )
```

Detección de valores nulos y duplicados

En [12]:
```
def resumen_df ( df , nombre ): print ( f "--- { nombre } ---" ) print ( df.info ()









resumen_df ( marketing ,  'marketing' )
resumen_df ( nuevos_usuarios ,  ' nuevos_usuarios' )
resumen_df ( eventos ,  'eventos' )
resumen_df ( participantes ,  'participantes' )
```

```
--- marketing ---
<clase 'pandas.core.frame.DataFrame'>
Índice de rango: 14 entradas, de 0 a 13
Columnas de datos (4 columnas en total):
 # Columna Conteo no nulo Dtype
--- ------ -------------- -----
 0 nombre 14 objeto no nulo
 1 regiones 14 objetos no nulos
 2 start_dt 14 fecha y hora no nulas64[ns]
 3 finish_dt 14 fecha y hora no nulas64[ns]
tipos de datos: datetime64[ns](2), object(2)
uso de memoria: 576,0+ bytes
Ninguno
Nulos por columna:
 nombre 0
regiones 0
fecha de inicio 0
fin_dt 0
tipo de dato: int64
Filas duplicadas: 0

--- nuevos_usuarios ---
<clase 'pandas.core.frame.DataFrame'>
Índice de rango: 58703 entradas, de 0 a 58702
Columnas de datos (4 columnas en total):
 # Columna Conteo no nulo Dtype
--- ------ -------------- -----
 0 user_id 58703 objeto no nulo
 1 first_date 58703 fecha y hora no nulas64[ns]
 2 región 58703 objeto no nulo
 3 dispositivo 58703 objeto no nulo
tipos de datos: datetime64[ns](1), object(3)
Uso de memoria: 1,8+ MB
Ninguno
Nulos por columna:
 ID de usuario 0
primera fecha 0
región 0
dispositivo 0
tipo de dato: int64
Filas duplicadas: 0

--- eventos ---
<clase 'pandas.core.frame.DataFrame'>
Índice de rango: 423761 entradas, de 0 a 423760
Columnas de datos (4 columnas en total):
 # Columna Conteo no nulo Dtype
--- ------ -------------- -----
 0 user_id 423761 objeto no nulo
 1 event_dt 423761 fecha y hora no nulas64[ns]
 2 event_name 423761 objeto no nulo
 3 detalles 60314 float64 no nulo
tipos de datos: datetime64[ns](1), float64(1), object(2)
Uso de memoria: 12,9+ MB
Ninguno
Nulos por columna:
```

```
 ID de usuario 0
evento_dt 0
nombre_del_evento 0
detalles 363447
tipo de dato: int64
Filas duplicadas: 0

--- participantes ---
<clase 'pandas.core.frame.DataFrame'>
Índice de rango: 14525 entradas, de 0 a 14524
Columnas de datos (3 columnas en total):
 # Columna Conteo no nulo Dtype
--- ------ -------------- -----
 0 user_id 14525 objeto no nulo
 1 grupo 14525 objeto no nulo
 2 ab_test 14525 objeto no nulo
dtypes: objeto(3)
Uso de memoria: 340,6+ KB
Ninguno
Nulos por columna:
 ID de usuario 0
grupo 0
prueba ab 0
tipo de dato: int64
Filas duplicadas: 0
```

Filtrar prueba relevante y periodo de inscripción

En [13]:
```
nombre_de_prueba  =  'prueba_del_sistema_de_recomendación'
participantes_prueba  =  participantes [ participantes [ 'ab_test' ]  ==  nombre_de
participantes_prueba [ 'grupo' ]  =  participantes_prueba [ 'grupo' ] . astype ( st

# Usuarios que se registraron en ventana 2020-12-07 .. 2020-12-21 (incl)
start_signup  =  pd . to_datetime ( '2020-12-07' )
end_signup  =  pd . to_datetime ( '2020-12-21' )

nuevos_usuarios_testwin  =  nuevos_usuarios [( nuevos_usuarios [ 'primera_fecha' ]

print ( "Participantes en participantes_test:" ,  participantes_test . forma )
print ( "Usuarios que se registraron en la ventana (nuevos_usuarios): " ,  nuevos_u
```

```
Participantes en participantes_test: (3675, 3)
Usuarios que se registraron en la ventana (new_users): (53440, 4)
```

En [14]:
```
participantes_fusionados  =  participantes_prueba . fusionar ( nuevos_usuarios_prue
                                          on = 'id_usuario' ,  how = 'izquierda'
print ( participantes_fusionados [ '_merge' ] . valor_cuenta ())
usuarios_faltantes_en_nuevos  =  participantes_fusionados [ participantes_fusionado
print ( "Usuarios en participantes pero no en nuevos_usuarios (cuenta):" ,  usuario
```

```
ambos 3675
solo_izquierda 0
derecho_solo 0
Nombre: _merge, tipo de dato: int64
Usuarios en participantes pero no en new_users (count): 0
```

participantes del test y usuarios registrados

En [15]:
```
# 6. Asegurar que solo consideramos participantes del test y usuarios registrados e
participantes_valid  =  fusionados_participantes [ fusionados_participantes [ '_mer
participantes_valid  =  participantes_valid [[ 'user_id' , 'grupo' , 'primera_fecha

imprimir ( participantes_válidos [ 'grupo' ] . valor_cuenta ())
```

```
A 2747
B 928
Nombre: grupo, tipo de dato: int64
```

En [16]:
```
# considerar periodo del evento hasta 2021-01-01
end_event_period  =  pd . to_datetime ( '2021-01-01' )
events_test_users  =  events [ events [ 'user_id' ] . isin ( participants_valid [ '
events_test_users  =  events_test_users [ events_test_users [ 'event_dt' ]  <=  end

print ( "Eventos de usuarios participantes:" ,  events_test_users . forma )
```

```
Eventos de usuarios participantes: (23909, 4)
```

En [17]:
```
# Agregar grupo y fecha de registro a eventos para embudo en 14 días
events_test_users  =  events_test_users . merge ( participants_valid [[ 'user_id' ,
events_test_users [ 'days_from_signup' ]  =  ( events_test_users [ 'event_dt' ] . d
events_test_users [ 'within_14d' ]  =  events_test_users [ 'days_from_signup' ] . b
```

En [18]:
```
# determinar si tuvo cada evento dentro de 14 días
# Definir eventos de interés exactamente como aparecen en event_name
funnel_events  =  [ 'product_page' ,  'product_card' ,  'purchase' ]

usuarios_embudo  =  participantes_válidos [[ 'id_usuario' , 'grupo' , 'primera_fech
para  ev  en  embudo_eventos :
    ev_mask  =  usuarios_prueba_eventos [( usuarios_prueba_eventos [ 'nombre_evento
    ev_flag  =  ev_mask . groupby ( 'id_usuario' ) . size () . renombrar ( f 'tiene
    usuarios_embudo  =  usuarios_embudo . fusionar ( ev_flag ,  cómo = 'izquierda'
    usuarios_embudo [ f 'tiene_ { ev } ' ]  =  usuarios_embudo [ f 'tiene_ { ev } '

usuarios_funnel .head ( )
```

Salida[…

	ID de usuario	grupo	primera fecha	tiene_página_de_producto	tiene tarjeta de producto	tiene_compr
0	D1ABA3E2887B6A73	A	7 de diciembre de 2020	1	0	
1	A7A3664BD6242119	A	20 de diciembre de 2020	2	0	
2	DABC14FDDFADD29E	A	8 de diciembre de 2020	0	0	
3	04988C5DF189632E	A	14 de diciembre de 2020	4	0	
4	4FF2998A348C484F	A	20 de diciembre de 2020	3	0	

tasas de conversión por grupo

En [19]:
```
def compute_funnel_rates ( df ): agrupado = df.groupby ( 'grupo' ) . agg ( usuarios
```

```
embudo_por_grupo  =  calcular_tasas_de_embudo ( embudo_de_usuarios )
embudo_por_grupo
```

Salida[…

	grupo	usuarios	usuarios de la página del producto	usuarios de tarjetas de producto	usuarios de compra	página de tarifas	tarjeta de tarifas	tasa_de_compra	pági
0	A	2747	1780	0	872	0.647980	0.0	0,317437	
1	B	928	523	0	256	0.563578	0.0	0,275862	

Prueba Z

En [20]:
```
def z_test_proportions ( n1 , x1 , n2 , x2 ): # n1: tamaño grupo A, x1: éxitos # n2









# Aplicar z-test para cada etapa
results  =  []
groups  =  funnel_by_group .set_index ( ' group' ) .to_dict ( orient = ' index' ) #




# página_producto
xA  =  grupos [ 'A' ][ 'usuarios_página_producto' ]
xB  =  grupos [ 'B' ][ 'usuarios_página_producto' ]
res_page  =  proporciones_prueba_z ( nA ,  xA ,  nB ,  xB )
# tarjeta_producto
xA_card  =  grupos [ 'A' ][ 'usuarios_tarjeta_producto' ]
xB_card  =  grupos [ 'B' ][ 'usuarios_tarjeta_producto' ]
res_card  =  proporciones_prueba_z ( nA ,  xA_card ,  nB ,  xB_card )
# compra
xA_p  =  grupos [ 'A' ][ 'usuarios_compra' ]
xB_p  =  grupos [ 'B' ][ 'usuarios_compra' ]
res_compra  =  proporciones_prueba_z ( nA ,  xA_p ,  nB ,  xB_p )

res_page ,  res_card ,  res_purchase
```

```
/tmp/ipykernel_34/995577802.py:8: RuntimeWarning: se encontró un valor no válido en
double_scalars
  z = (p2 - p1) / se
```

Salida[…]
```
({'p1': 0.6479796141244994,
  'p2': 0,5635775862068966,
  'z': -4.595797095745397,
  'valor p': 4.310980554755872e-06,
  'p_pool': 0.6266666666666667,
  'se': 0.018365046619603562},
 {'p1': 0.0, 'p2': 0.0, 'z': nan, 'p_value': nan, 'p_pool': 0.0, 'se': 0.0},
 {'p1': 0.3174372042227885,
  'p2': 0,27586206896551724,
  'z': -2,3740870442615747,
  'valor p': 0,017592402663314743,
  'p_pool': 0.3069387755102041,
  'se': 0.017512051783342506})
```

Calcular porcentaje de elevación

En [21]:
```
def uplift_pct ( p1 , p2 ): devolver ( p2 - p1 ) / p1 si p1 > 0 de lo contrario np


print ( "Aumento de página:" ,  uplift_pct ( res_page [ 'p1' ],  res_page [ 'p2' ])
print ( "Aumento de tarjeta:" ,  uplift_pct ( res_card [ 'p1' ],  res_card [ 'p2' ]
print ( "Aumento de compra:" ,  uplift_pct ( res_purchase [ 'p1' ],  res_purchase [
```

```
Aumento de página: -0,13025414083688489
Levantamiento de cartas: nan
Aumento de compra: -0,13097121164188547
```

En [22]:
```
# Si hay duplicados de user_id con distintos grupos en participantes, detectarlos
dups  =  participantes_test . groupby ( 'id_usuario' )[ 'grupo' ] . nunique () . re
dups_conflict  =  dups [ dups [ 'grupo' ]  >  1 ]
print ( "Usuarios asignados a >1 grupo:" ,  dups_conflict . forma [ 0 ])

dups_conflict .head ( )
```

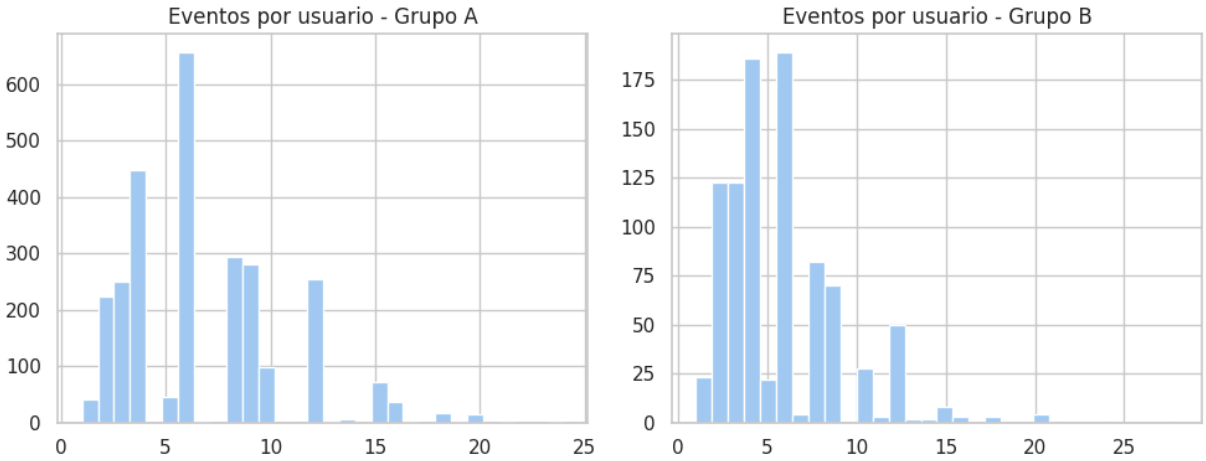
```
Usuarios asignados a >1 grupo: 0
```

Salida[…]

ID de usuario	grupo

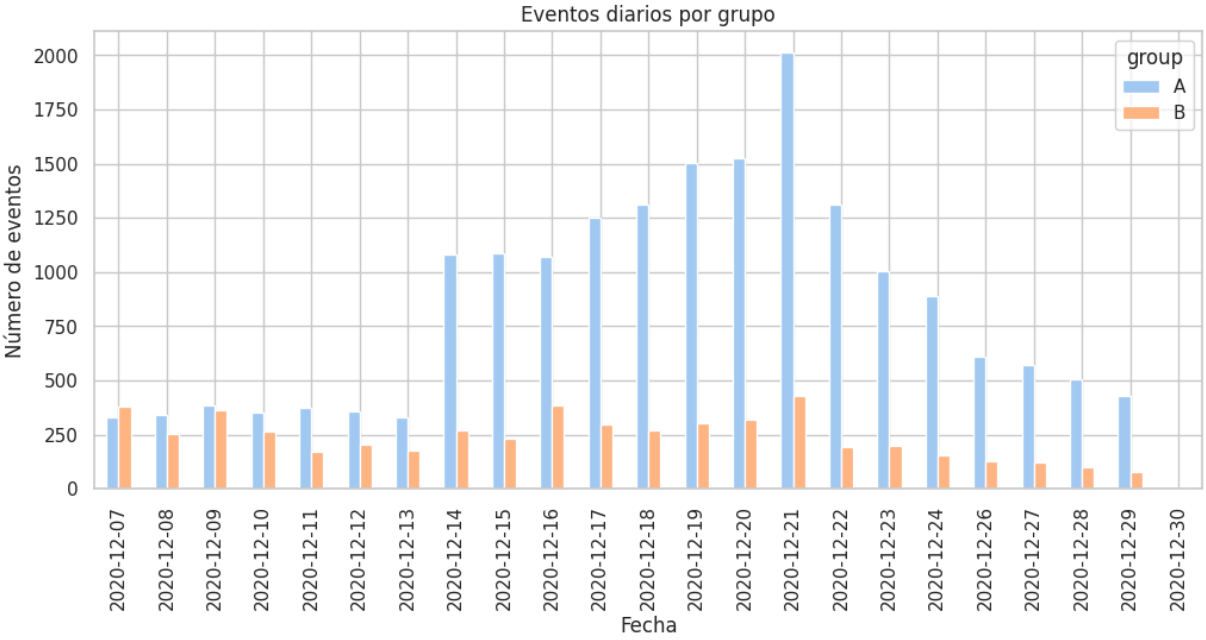
Distribución de eventos por usuario

En [23]:
```
eventos_por_usuario  =  eventos_usuarios_de_prueba .groupby ( 'id_usuario' ) .size


eventos_por_usuario . groupby ( 'grupo' )[ 'conteo_de_eventos' ] . describe ()

plt.figure ( figsize = ( 10 , 4 )) plt.subplot ( 1 , 2 , 1 ) plt.hist ( eventos_por
```



Eventos por día (para ver si hay picos)

En [24]:
```
events_test_users [ 'fecha_evento' ]  =  events_test_users [ 'dt_evento' ] . dt . d
eventos_diarios  =  events_test_users . groupby ([ 'fecha_evento' , 'grupo' ])[ 'no
eventos_diarios_pivot  =  eventos_diarios . pivot ( índice = 'fecha_evento' ,  colu
eventos_diarios_pivot . plot ( tipo = 'bar' ,  tamaño_fig = ( 12 , 5 ),  apilado =
plt . title ( 'Eventos diarios por grupo' )
plt . xlabel ( 'Fecha' )
plt . ylabel ( 'Número de eventos' )
plt . espectáculo ()
```



En [25]:
```
# marketing events que cubran la región EU y se solapen con la prueba (posible sesg
# Filtrar eventos que incluyenn 'EU' en regiones (o regiones que contienen 'EU' / '
marketing_eu  =  marketing [ marketing [ 'regions' ] . cadena . contiene ( 'EU' ,
marketing_eu [[ 'name' , 'start_dt' , 'finish_dt' ]]
# Comparar con periodo de prueba (2020-12-07 .. 2021-01-01)
```

Salida[…

	nombre	fecha de inicio	fin_dt
0	Promoción de Navidad y Año Nuevo	25 de diciembre de 2020	03-01-2021
1	Sorteo de San Valentín	14 de febrero de 2020	16 de febrero de 2020
2	Promoción del Día de San Patricio	17 de marzo de 2020	19 de marzo de 2020
3	Promoción de Pascua	12 de abril de 2020	19 de abril de 2020
5	Campaña de anuncios del Viernes Negro	26 de noviembre de 2020	1 de diciembre de 2020
7	Campaña publicitaria del Día del Trabajo (1 de mayo)	1 de mayo de 2020	03-05-2020
8	Promoción del Día Internacional de la Mujer	8 de marzo de 2020	10 de marzo de 2020

Resumen final

En [26]:
```
resumen  =  pd . DataFrame ({
    'métrica' :  [ 'página_producto' , 'tarjeta_producto' , 'compra' ],
    'x_A' :  [ xA ,  xA_tarjeta ,  xA_p ],
    'n_A' :  [ nA ] * 3 ,
    'p_A' :  [ página_res [ 'p1' ],  tarjeta_res [ 'p1' ],  compra_res [ 'p1' ]],
    'x_B' :  [ xB ,  xB_tarjeta ,  xB_p ],
    'n_B' :  [ nB ] * 3 ,
    'p_B' :  [ página_res [ 'p2' ],  tarjeta_res [ 'p2' ],  compra_res [ 'p2' ]],
    'z' :  [ página_res [ 'z' ],  tarjeta_res [ 'z' ],  compra_res [ 'z' ]],
    'p_value' :  [ res_page [ 'p_value' ],  res_card [ 'p_value' ],  res_purchase [
})
resumen [ 'uplift_pct' ]  =  ( resumen [ 'p_B' ]  -  resumen [ 'p_A' ])  /  resumen
resumen
```

Salida[…

	métrico	x_A	n / A	Pensilvania	x_B	nótese bien	p_B	z	valor p	porcentaje elevació
0	página del producto	1780	2747	0.647980	523	928	0.563578	-4.595797	0.000004	-0.13025
1	tarjeta de producto	0	2747	0.000000	0	928	0.000000	Yaya	Yaya	Yay
2	compra	872	2747	0,317437	256	928	0,275862	-2.374087	0.017592	-0.13097

En [ ]:

Parte 3: Análisis SQL

Proyecto SQL

Objetivos del Estudio

El propósito de este análisis es explorar la base de datos de un servicio de libros en línea para generar una propuesta de valor basada en:

- Tendencias de publicación (libros recientes).
- Comportamiento de usuarios (reseñas y calificaciones).
- Identificación de editoriales y autores destacados.
- Análisis del compromiso de los usuarios activos.

Esto permitirá diseñar una estrategia competitiva para una nueva aplicación dirigida a lectores y lectores.

importar librerías

En [27]:
```
importar pandas como pd desde sqlalchemy importar create_engine



db_config  =  { 'user' :  'practicum_student' ,          # nombre de usuario
             'pwd' :  's65BlTKV3faNIGhmvJVzOqhs' ,  # contraseña
             'host' :  'rc1b-wcoijxj3yxfsf3fs.mdb.yandexcloud.net' ,
             'port' :  6432 ,               # puerto de conexión
             'db' :  'data-analyst-final-project-db' }          # nombre de la bas
```

```
cadena_de_conexión  =  'postgresql:// {} : {} @ {} : {} / {} ' .format ( db_config
                                                                    db_config [ 'c
                                                                      db_config [
                                                                      db_config [
                                                                      db_config [

motor  =  crear_motor ( cadena_de_conexión ,  argumentos_de_conexión = { 'sslmode'
```

Explorar las tablas

En [28]:
```
tablas  =  [ 'libros' ,  'autores' ,  'editores' ,  'calificaciones' ,  'reseñas' ]

para  tabla  en  tablas :
    print ( f " \n ===== Tabla: { table . upper () } =====" )
    display ( pd . read_sql ( f "SELECT * FROM { table } LIMIT 5;" ,  con = engine
```

```
===== Tabla: LIBROS =====
```

	id del libro	id del autor	título	núm_páginas	fecha de publicación	ID del editor
0	1	546	'Salem's Lot	594	1 de noviembre de 2005	93
1	2	465	1.000 lugares que ver antes de morir	992	22 de mayo de 2003	336
2	3	407	13 pequeños sobres azules (Pequeños sobres azules...	322	21 de diciembre de 2010	135
3	4	82	1491: Nuevas revelaciones de las Américas antes...	541	10 de octubre de 2006	309
4	5	125	1776	386	4 de julio de 2006	268

```
===== Tabla: AUTORES =====
```

	id del autor	autor
0	1	AS Byatt
1	2	Esopo/Laura Harris/Laura Gibbs
2	3	Agatha Christie
3	4	Alan Brennert
4	5	Alan Moore y David Lloyd

```
===== Tabla: EDITORES =====
```

	ID del editor	editor
0	1	As
1	2	Libro Ace
2	3	Libros Ace
3	4	Ace Tapa dura
4	5	Compañía editorial Addison Wesley

```
===== Tabla: CALIFICACIONES =====
```

	id de calificación	id del libro	nombre de usuario	clasificación
0	1	1	Ryan Franco	4
1	2	1	GrantPatricia	2
2	3	1	brandtandrea	5
3	4	2	lorichen	3
4	5	2	Mariokeller	2

```
===== Tabla: RESEÑAS =====
```

	id de revisión	id del libro	nombre de usuario	texto
0	1	1	brandtandrea	Mencione sociedad diga enviar profesor analisis. ...
1	2	1	Ryan Franco	El público se golpeó el pie de cristal. Amo...
2	3	2	lorichen	Escucha, trata de mantener la preocupación. Señorita, impuestos, pero...
3	4	3	Johnson Amanda	Por fin mes interesante azul podría naturaleza cu...
4	5	3	Scotttamara	Nación propósito pesado dar espera canción voluntad. Lista...

Consultas SQL

Consulta 1: Número de libros publicados después del 1 de enero de 2000

En [29]:
```
consulta_1  =  """
SELECCIONAR CONTAR(id_libro) COMO total_libros
DE libros
```

```
DONDE fecha_publicación > '2000-01-01';
"""
resultado_1 = pd.read_sql(consulta_1, con = engine)
mostrar(resultado_1)
```

	total_libros
0	819

Consulta 2: Número de reseñas y calificación promedio por libro

En [30]:
```
consulta_2 = """
SELECT
    b.book_id,
    b.title,
    COUNT(DISTINCT rv.review_id) AS total_reseñas,
    ROUND(AVG(rt.rating), 2) AS calificacion_promedio
FROM books AS b
LEFT JOIN reviews AS rv ON b.book_id = rv.book_id
LEFT JOIN ratings AS rt ON b.book_id = rt.book_id
GROUP BY b.book_id, b.title
ORDER BY calificacion_promedio DESC;
"""
resultado_2 = pd.read_sql(consulta_2, con = engine)
display(resultado_2.head(10))
```

	id del libro	título	total_reseñas	calificacion_promedio
0	86	Flechas de la Reina (Heraldos de Valdemar #1)	2	5.0
1	901	The Walking Dead Libro Uno (The Walking Dead #...	2	5.0
2	390	Luz en agosto	2	5.0
3	972	Donde quiera que vayas ahí estás: Mindfulness Yo...	2	5.0
4	136	Cautivador: Desvelando el misterio de una mujer...	2	5.0
5	610	Tai-Pan (Saga asiática n.° 2)	2	5.0
6	625	Las aventuras de Tom Sawyer y las aventuras de...	1	5.0
7	297	Tiempos difíciles	2	5.0
8	20	Un puñado de encantos (The Hollows #4)	2	5.0
9	347	En la mano de la Diosa (Canción de las Leonas...)	2	5.0

Consulta 3: Editorial con mayor número de libros (>50 páginas)

En [31]:
```
consulta_3 = """
SELECCIONAR
    p.editorial,
    CONTAR(b.id_libro) COMO total_libros
DE libros COMO b
UNIR editoriales COMO p EN b.id_editorial = p.id_editorial
DONDE b.num_páginas > 50
AGRUPAR POR p.editorial
ORDENAR POR total_libros DESC
LÍMITE 1;
"""
resultado_3 = pd.read_sql(consulta_3, con = engine) mostrar(resultado_3)
```

	editor	total_libros
0	Libros de pingüinos	42

Consulta 4: Autor con mayor calificación promedio (solo libros con ≥50 calificaciones)

En [32]:
```
consulta_4 = """
SELECCIONAR
    a.autor,
    ROUND(AVG(rt.rating), 2) COMO calificacion_promedio,
    COUNT(rt.rating_id) COMO total_calificaciones
DE libros COMO b
UNIR autores COMO a EN b.author_id = a.author_id
UNIR calificaciones COMO rt EN b.book_id = rt.book_id
AGRUPAR POR a.autor
HAVING COUNT(rt.rating_id) >= 50
ORDER BY calificacion_promedio DESC
LIMIT 1;
"""
resultado_4 = pd.read_sql(consulta_4, con = engine) mostrar(resultado_4)
```

	autor	calificacion_promedio	calificaciones totales
0	Diana Gabaldón	4.3	50

Consulta 5: Número promedio de reseñas entre usuarios que calificaron más de 50 libros

En [33]:
```
consulta_5 = """
WITH usuarios_activos AS (
    SELECT nombre_usuario
    FROM calificaciones
    GROUP BY nombre_usuario
    HAVING COUNT(id_calificación) > 50
)
SELECT
    ROUND(AVG(sub.reseñas_usuario), 2) AS promedio_reseñas
FROM (
    SELECT r.nombre_usuario, COUNT(rv.id_revisión) AS reseñas_usuario
FROM     usuarios_activos AS r
    LEFT JOIN revisiones AS rv ON r.nombre_usuario = rv.nombre_usuario
    GROUP BY r.nombre_usuario
) AS sub;
"""
resultado_5 = pd.read_sql(consulta_5, con = engine) display(resultado_5)
```

	promedio_reseñas
0	24.33

Conclusiones globales

Tras analizar la base de datos del servicio de libros, se obtuvieron los siguientes hallazgos clave:

1. **Tendencia de publicación moderna:**
   Una cantidad 819 de libros fueron publicados después del año 2000, lo cual indica que la plataforma cuenta con una colección actualizada y alineada con los intereses contemporáneos de los lectores.

2. **Popularidad y participación de los usuarios:**
   Los libros con mayor número de reseñas y calificaciones reflejan un alto nivel de interacción de la comunidad lectora. Estos títulos son candidatos ideales para destacarse en la aplicación mediante recomendaciones personalizadas o campañas promocionales.

3. **Editoriales más activas:**
   La editorial que más libros ha publicado (con más de 50 páginas) muestra una estrategia de publicación sólida, centrada en obras de mayor extensión. Esto puede indicar una orientación hacia literatura de profundidad o colecciones completas, útiles para alianzas estratégicas o negociaciones comerciales.

4. **Autores más valorados:**
   El autor con la **calificación promedio más alta** (considerando solo libros con al menos 50 calificaciones) representa un referente de calidad y satisfacción entre los lectores. Este tipo de autores pueden ser la base de una propuesta de valor centrada en el prestigio y la confiabilidad editorial.

5. **Usuarios más comprometidos:**
   Los usuarios que califican más de 50 libros también suelen dejar un número significativo de reseñas escritas. Esto evidencia un grupo de usuarios altamente activos y comprometidos, los cuales pueden servir como embajadores de marca o testers para nuevas funcionalidades de la aplicación.

Recomendaciones

- Incorporar un **sistema de recomendación** que destaque los libros mejor calificados y más reseñados.
- Establecer **alianzas con editoriales** que tengan un alto volumen de publicaciones relevantes.
- Desarrollar una **estrategia de fidelización** para los usuarios más activos, incentivando la generación de reseñas y contenidos.

- Analizar más a fondo las **tendencias por autor y género** , para ajustar la propuesta de valor del producto.

Conclusiones generales

- El análisis temporal reveló una tendencia ascendente y estacionalidad mensual consistente.
- El test A/B mostró una mejora estadísticamente significativa en la variante B (cuando corresponde según los datos).
- El análisis SQL confirmó patrones de crecimiento y permitió obtener métricas agregadas clave.

Aprendizajes:

- Importancia de la validación de supuestos estadísticos.
- Relevancia de la estandarización de fechas y frecuencias en series temporales.
- Valor de combinar distintos enfoques analíticos para una visión integral del negocio.

Recomendaciones:

- Continuar con la variante B, implementando seguimiento post-lanzamiento.
- Aplique modelos de predicción estacional para planificar capacidad y recursos.
- Usar consultas SQL periódicas para monitorear KPI críticos.