```
import pandas as pd
import numpy as np
from scipy.stats import ttest_ind
from scipy import stats


from google.colab import drive
drive.mount("/content/drive")
%cd /content/drive/MyDrive/bq_results
df = pd.read_csv("ab_test_result.csv" )
df.head()
```

```
Mounted at /content/drive
/content/drive/MyDrive/bq_results
```

	date	country	device	continent	channel	test	test_group	event_name	value
0	2020-11-01	Lithuania	mobile	Europe	Organic Search	2	2	new account	1
1	2020-11-01	El Salvador	desktop	Americas	Social Search	2	1	new account	1
2	2020-11-01	Slovakia	mobile	Europe	Paid Search	2	2	new account	1
3	2020-11-01	Lithuania	desktop	Europe	Paid Search	2	2	new account	1
4	2020-11-02	North Macedonia	desktop	Europe	Direct	2	1	new account	1

```
# Конвертуємо колонку 'date' в тип datetime
df['date'] = pd.to_datetime(df['date'], errors='coerce')



print(df['date'].min(), df['date'].max())
```

```
2020-11-01 00:00:00 2021-01-27 00:00:00
```

```
print(df['event_name'].unique())
```

```
['new account' 'session with orders' 'sessions' 'user_engagement'
 'first_visit' 'view_item' 'page_view' 'session_start' 'view_promotion'
 'scroll' 'add_payment_info' 'begin_checkout' 'add_to_cart' 'select_item'
 'view_search_results' 'add_shipping_info' 'select_promotion' 'click'
 'view_item_list']
```

Створення основного дф

Основні кроки: Оголошення метрик та сегментів:

Визначаються метрики (add_payment_info_per_session, add_shipping_info_per_session, тощо), які обчислюються як відношення подій (event_name) до загальної кількості сесій (sessions). Встановлюються категорії для групування даних: test, device, channel. Обхід груп даних:

Дані групуються за заданими сегментами (test, device, channel). Для кожної групи розраховуються чисельники (кількість подій), знаменники (кількість сесій) та коефіцієнти конверсій (CTR) для тестової та контрольної груп. Розрахунок CTR та різниці в конверсіях:

CTR для кожної групи обчислюється як відношення чисельника до знаменника. Різниця в конверсіях (ctr_difference) обчислюється у відсотках відносно контрольної групи. Розрахунок статистичної значущості:

Використовується z-тест для перевірки статистичної значущості змін між контрольними та тестовими групами: Обчислюється pooled_ctr (загальний CTR). Розраховується стандартна помилка (standard_error). Виконується обчислення z_stat та відповідного p_value. Результат порівнюється з порогом значущості (p_value < 0.05), щоб визначити, чи є зміни значущими. Збереження результатів:

Результати додаються в список у вигляді словників. Після завершення циклів створюється таблиця (results_df) із даними для подальшого аналізу. Вихідні дані: Таблиця (DataFrame) містить такі ключові стовпці:

> *test:* Номер тесту.
> *metric:* Назва метрики.
> *numerator/denominator:* Подія та кількість сесій.
> *ctr_1/ctr_2:* Конверсії для контрольної та тестової груп.
> *ctr_difference:* Відсоткова різниця між групами.
> z_stat/p_value: Результати статистичного тесту.
> *significant:* Вказує, чи є зміни статистично значущими.

```
metrics = {
    "add_payment_info_per_session": lambda df: df[df['event_name'] == "add_payment_info"]['value'].sum() / df[df['event_name'] == "sess
```

```
    "add_shipping_info_per_session": lambda df: df[df['event_name'] == "add_shipping_info"]['value'].sum() / df[df['event_name'] == "ses
    "begin_checkout_per_session": lambda df: df[df['event_name'] == "begin_checkout"]['value'].sum() / df[df['event_name'] == "sessions'
    "new account_per_session": lambda df: df[df['event_name'] == "new account"]['value'].sum() / df[df['event_name'] == "sessions"]['va
}

segments = ['test', 'device', 'channel']
events = ["add_payment_info", "add_shipping_info", "begin_checkout", "new account"] # Список подій

results = []

for metric_name, metric_formula in metrics.items():
    event = metric_name.split('_per_session')[0]
    for _, group_data in df.groupby(segments):
        control = group_data[group_data["test_group"] == 1]
        test = group_data[group_data["test_group"] == 2]

        # Витягуємо значення для чисельника та знаменника для контрольної групи
        numerator_value_1 = control[control['event_name'] == event]['value'].sum()
        denominator_value_1 = control[control['event_name'] == 'sessions']['value'].sum()
        ctr_1 = numerator_value_1 / denominator_value_1 if denominator_value_1 != 0 else None

        # Витягуємо значення для чисельника та знаменника для тестової групи
        numerator_value_2 = test[test['event_name'] == event]['value'].sum()
        denominator_value_2 = test[test['event_name'] == 'sessions']['value'].sum() # Виправлено: використовуємо загальну кількість сес
        ctr_2 = numerator_value_2 / denominator_value_2 if denominator_value_2 != 0 else None

        # Різниця у конверсіях
        ctr_difference = ((ctr_2 - ctr_1) / ctr_1) * 100 if ctr_1 is not None and ctr_2 is not None and ctr_1 != 0 else None

        # Перевірка наявності даних
        if (denominator_value_1 + denominator_value_2) != 0:
            pooled_ctr = (numerator_value_1 + numerator_value_2) / (denominator_value_1 + denominator_value_2)
        else:
            pooled_ctr = None

        if pooled_ctr is not None and 0 <= pooled_ctr <= 1 and denominator_value_1 != 0 and denominator_value_2 != 0:
            standard_error = ((pooled_ctr * (1 - pooled_ctr)) * (1 / denominator_value_1 + 1 / denominator_value_2)) ** 0.5
        else:
            standard_error = None

        z_stat = (ctr_2 - ctr_1) / standard_error if standard_error is not None and standard_error != 0 else None

        if numerator_value_1 == 0 or denominator_value_1 == 0 or numerator_value_2 == 0 or denominator_value_2 == 0:
            z_stat, p_value = None, None
            significant = False
        else:
            p_value = 2 * (1 - stats.norm.cdf(abs(z_stat))) if z_stat is not None else None
            significant = p_value < 0.05 if p_value is not None else False

        results.append({
            "test": group_data['test'].iloc[0] if 'test' in group_data.columns else None,
            #"date": group_data['date'].iloc[0] if 'date' in group_data.columns else None,
            #"country": group_data['country'].iloc[0] if 'country' in group_data.columns else None,
            #"continent": group_data['continent'].iloc[0] if 'continent' in group_data.columns else None,
            "device": group_data['device'].iloc[0] if 'device' in group_data.columns else None,
            "channel": group_data['channel'].iloc[0] if 'channel' in group_data.columns else None,
            "metric": metric_name,
            "numerator": event,
            "denominator": 'session',
            "numerator_value_1": numerator_value_1,
            "denominator_value_1": denominator_value_1,
            "ctr_1": ctr_1,
            "numerator_value_2": numerator_value_2,
            "denominator_value_2": denominator_value_2,
            "ctr_2": ctr_2,
            "ctr_difference": ctr_difference,
            "z_stat": z_stat,
            "p_value": p_value,
            "significant": significant,
        })

results_df = pd.DataFrame(results)
results_df.head()
```

	test	device	channel	metric	numerator	denominator	numerator_value_1	denominator_value_1	ctr_
0	1	desktop	Direct	add_payment_info_per_session	add_payment_info	session	227	6258	0.03627
1	1	desktop	Organic Search	add_payment_info_per_session	add_payment_info	session	334	9221	0.03622
2	1	desktop	Paid Search	add_payment_info_per_session	add_payment_info	session	288	6843	0.04208
3	1	desktop	Social Search	add_payment_info_per_session	add_payment_info	session	126	2232	0.05645
4	1	desktop	Undefined	add_payment_info_per_session	add_payment_info	session	155	1913	0.08102

```
results_df.to_excel('test_data.xlsx', index=False)
```

Дф для розрахунку статистичної значущості загалом по тестах

Так як ми не можемо ніяк агрегувати показники отримані зі статистичних тестів, для візуалізації даних виконую код повторно і формую ще один дф з групуванням тільки по колонці 'test'.

```
metrics = {
    "add_payment_info_per_session": lambda df: df[df['event_name'] == "add_payment_info"]['value'].sum() / df[df['event_name'] == "sess
    "add_shipping_info_per_session": lambda df: df[df['event_name'] == "add_shipping_info"]['value'].sum() / df[df['event_name'] == "se
    "begin_checkout_per_session": lambda df: df[df['event_name'] == "begin_checkout"]['value'].sum() / df[df['event_name'] == "sessions'
    "new account_per_session": lambda df: df[df['event_name'] == "new account"]['value'].sum() / df[df['event_name'] == "sessions"]['va
}

segments = ['test']
events = ["add_payment_info", "add_shipping_info", "begin_checkout", "new account"] # Список подій

results = []

for metric_name, metric_formula in metrics.items():
    event = metric_name.split('_per_session')[0]
    for _, group_data in df.groupby(segments):
        control = group_data[group_data["test_group"] == 1]
        test = group_data[group_data["test_group"] == 2]

        # Витягуємо значення для чисельника та знаменника для контрольної групи
        numerator_value_1 = control[control['event_name'] == event]['value'].sum()
        denominator_value_1 = control[control['event_name'] == 'sessions']['value'].sum()
        ctr_1 = numerator_value_1 / denominator_value_1 if denominator_value_1 != 0 else None

        # Витягуємо значення для чисельника та знаменника для тестової групи
        numerator_value_2 = test[test['event_name'] == event]['value'].sum()
        denominator_value_2 = test[test['event_name'] == 'sessions']['value'].sum() # Виправлено: використовуємо загальну кількість сес
        ctr_2 = numerator_value_2 / denominator_value_2 if denominator_value_2 != 0 else None

        # Різниця у конверсіях
        ctr_difference = ((ctr_2 - ctr_1) / ctr_1) * 100 if ctr_1 is not None and ctr_2 is not None and ctr_1 != 0 else None

        # Перевірка наявності даних
        if (denominator_value_1 + denominator_value_2) != 0:
            pooled_ctr = (numerator_value_1 + numerator_value_2) / (denominator_value_1 + denominator_value_2)
        else:
            pooled_ctr = None

        if pooled_ctr is not None and 0 <= pooled_ctr <= 1 and denominator_value_1 != 0 and denominator_value_2 != 0:
            standard_error = ((pooled_ctr * (1 - pooled_ctr)) * (1 / denominator_value_1 + 1 / denominator_value_2)) ** 0.5
        else:
            standard_error = None

        z_stat = (ctr_2 - ctr_1) / standard_error if standard_error is not None and standard_error != 0 else None

        if numerator_value_1 == 0 or denominator_value_1 == 0 or numerator_value_2 == 0 or denominator_value_2 == 0:
            z_stat, p_value = None, None
            significant = False
        else:
            p_value = 2 * (1 - stats.norm.cdf(abs(z_stat))) if z_stat is not None else None
            significant = p_value < 0.05 if p_value is not None else False

        results.append({
            "test": group_data['test'].iloc[0] if 'test' in group_data.columns else None,
            #"date": group_data['date'].iloc[0] if 'date' in group_data.columns else None,
            #"country": group_data['country'].iloc[0] if 'country' in group_data.columns else None,
            #"continent": group_data['continent'].iloc[0] if 'continent' in group_data.columns else None,
```

```
            #"device": group_data['device'].iloc[0] if 'device' in group_data.columns else None,
            #"channel": group_data['channel'].iloc[0] if 'channel' in group_data.columns else None,
            "metric": metric_name,
            "numerator": event,
            "denominator": 'session',
            "numerator_value_1": numerator_value_1,
            "denominator_value_1": denominator_value_1,
            "ctr_1": ctr_1,
            "numerator_value_2": numerator_value_2,
            "denominator_value_2": denominator_value_2,
            "ctr_2": ctr_2,
            "ctr_difference": ctr_difference,
            "z_stat": z_stat,
            "p_value": p_value,
            "significant": significant,
        })

results_df = pd.DataFrame(results)
results_df.head()
```

	test	metric	numerator	denominator	numerator_value_1	denominator_value_1	ctr_1	numerator_value_
0	1	add_payment_info_per_session	add_payment_info	session	1988	45362	0.043825	22
1	2	add_payment_info_per_session	add_payment_info	session	2344	50637	0.046290	24(
2	3	add_payment_info_per_session	add_payment_info	session	3623	70047	0.051722	36
3	4	add_payment_info_per_session	add_payment_info	session	3731	105079	0.035507	36(
4	1	add_shipping_info_per_session	add_shipping_info	session	3034	45362	0.066884	32

```
results_df.to_excel('total_test_data.xlsx', index=False)
```

Створення дф для додаткової візуалізації

Для того, щоб віазулізувати динаміку ключових метрик по датах розраховую 4 ключові метрики, а також залишаю значення необхідних івентів для того, щоб не доводилося агрегувати CTR в Tableau. Цей дф доданий до робочої книги як ще одне джерело даних і дозволяє розраховувати показники CTR, за необхідності, також і в Tableau, групуючи дані в будь-якій варіації. Оскільки дані сегментовані по днях та всіх можливих показниках, проводити тести статистичної значущості не має сенсу.

```
metrics = {
    "add_payment_info_per_session": lambda df: df[df['event_name'] == "add_payment_info"]['value'].sum() / df[df['event_name'] == "sessio
    "add_shipping_info_per_session": lambda df: df[df['event_name'] == "add_shipping_info"]['value'].sum() / df[df['event_name'] == "sess
    "begin_checkout_per_session": lambda df: df[df['event_name'] == "begin_checkout"]['value'].sum() / df[df['event_name'] == "sessions"]
    "new account_per_session": lambda df: df[df['event_name'] == "new account"]['value'].sum() / df[df['event_name'] == "sessions"]['valu
}

segments = ['test', 'country', 'device', 'continent', 'channel', 'date']
events = ["add_payment_info", "add_shipping_info", "begin_checkout", "new account"] # Список подій

results = []

for metric_name, metric_formula in metrics.items():
    event = metric_name.split('_per_session')[0]
    for _, group_data in df.groupby(segments):
        control = group_data[group_data["test_group"] == 1]
        test = group_data[group_data["test_group"] == 2]

        # Витягуємо значення для чисельника та знаменника для контрольної групи
        numerator_value_1 = control[control['event_name'] == event]['value'].sum()
        denominator_value_1 = control[control['event_name'] == 'sessions']['value'].sum()
        ctr_1 = numerator_value_1 / denominator_value_1 if denominator_value_1 != 0 else None

        # Витягуємо значення для чисельника та знаменника для тестової групи
        numerator_value_2 = test[test['event_name'] == event]['value'].sum()
        denominator_value_2 = test[test['event_name'] == 'sessions']['value'].sum() # Виправлено: використовуємо загальну кількість сесій
        ctr_2 = numerator_value_2 / denominator_value_2 if denominator_value_2 != 0 else None

        # Різниця у конверсіях
        ctr_difference = ((ctr_2 - ctr_1) / ctr_1) * 100 if ctr_1 is not None and ctr_2 is not None and ctr_1 != 0 else None

        # Перевірка наявності даних
        if (denominator_value_1 + denominator_value_2) != 0:
            pooled_ctr = (numerator_value_1 + numerator_value_2) / (denominator_value_1 + denominator_value_2)
        else:
            pooled_ctr = None
```

```
        if pooled_ctr is not None and 0 <= pooled_ctr <= 1 and denominator_value_1 != 0 and denominator_value_2 != 0:
            standard_error = ((pooled_ctr * (1 - pooled_ctr)) * (1 / denominator_value_1 + 1 / denominator_value_2)) ** 0.5
        else:
            standard_error = None

        results.append({
            "test": group_data['test'].iloc[0] if 'test' in group_data.columns else None,
            "date": group_data['date'].iloc[0] if 'date' in group_data.columns else None,
            "country": group_data['country'].iloc[0] if 'country' in group_data.columns else None,
            "continent": group_data['continent'].iloc[0] if 'continent' in group_data.columns else None,
            "device": group_data['device'].iloc[0] if 'device' in group_data.columns else None,
            "channel": group_data['channel'].iloc[0] if 'channel' in group_data.columns else None,
            "metric": metric_name,
            "numerator": event,
            "denominator": 'session',
            "numerator_value_1": numerator_value_1,
            "denominator_value_1": denominator_value_1,
            "ctr_1": ctr_1,
            "numerator_value_2": numerator_value_2,
            "denominator_value_2": denominator_value_2,
            "ctr_2": ctr_2,
            "ctr_difference": ctr_difference,

        })

results_df = pd.DataFrame(results)
results_df.head()
```

	test	date	country	continent	device	channel	metric	numerator	denominator	numerator_value_1	der
0	1	2020-11-03	(not set)	(not set)	desktop	Direct	add_payment_info_per_session	add_payment_info	session	0	
1	1	2020-11-04	(not set)	(not set)	desktop	Direct	add_payment_info_per_session	add_payment_info	session	0	
2	1	2020-11-05	(not set)	(not set)	desktop	Direct	add_payment_info_per_session	add_payment_info	session	0	
3	1	2020-11-07	(not set)	(not set)	desktop	Direct	add_payment_info_per_session	add_payment_info	session	0	
4	1	2020-11-10	(not set)	(not set)	desktop	Direct	add_payment_info_per_session	add_payment_info	session	0	

```
results_df.to_excel('test.xlsx', index=False)
```

Посилання на робочу книгу в Tableau > https://public.tableau.com/views/Book1_17357529637370/CTRsignificance?:language=en-US&publish=yes&:sid=&:redirect=auth&:display_count=n&:origin=viz_share_link

Посилання на основний файл > https://docs.google.com/spreadsheets/d/17lsMDF4udfYeAxC6lQeB0ihF7ikTPZ5Y/edit?usp=sharing&ouid=100825818221053864338&rtpof=true&sd=true

Посилання на файл для додаткової візуалізації > https://docs.google.com/spreadsheets/d/1OyySs4K5nKuv1Sm7Czx62W2i0iyYGftJ/edit?usp=sharing&ouid=100825818221053864338&rtpof=true&sd=true