Аналитика поведенческой воронки вебинара
Аналитик: [Сороквашина Алена]
Дата: [25.06.2025]

📌 Цель исследования

Проанализировать воронку вебинара для селлеров маркетплейсов, выявить причины низкой конверсии и предложить улучшения.

🚩 Основные этапы воронки:
- Пользователь зашел в Telegram-бота
- Подключился к вебинару
- Отключился от вебинара
- Оставил заявку на продукт

📌 1. Основные результаты (TL;DR)

- Всего пользователей: **1000**
- До вебинара дошли: **38.2%**
- Общая конверсия в заявку: **4.7%**
- ⭐ Конверсия достигает **11.1%** в группе участников с участием **40–50 минут**
- 📉 До 40-й минуты уходит около **65%** пользователей
- 🔁 Повторные подключения повышают конверсию в **2.6 раза**
- 🧩 Гипотеза: оффер озвучен поздно (около 40-й минуты), большинство не доживает

🔷 2. Импорт библиотек и палитра

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

# Палитра
palette_custom = ["#dad7cd", "#a3b18a", "#588157", "#3a5a40", "#344e41"]
```

🔷 3. Загрузка данных

```
df = pd.read_csv("webinar_user_events.csv", sep=";")
df['created_at'] = pd.to_datetime(df['created_at'], format="%d.%m.%Y %H:%M")
```

🔷 4. Анализ основной воронки

```
total_users = df['user_id'].nunique()
entered_bot = df[df['action'] == 'enter_bot']['user_id'].nunique()
entered_webinar = df[df['action'] == 'enter_webinar_room']['user_id'].nunique()
made_order = df[df['action'] == 'make_order']['user_id'].nunique()

print("==== ВОРОНКА ====")
print(f"Всего пользователей: {total_users}")
print(f"Зашли в бота: {entered_bot} ({entered_bot / total_users:.1%})")
print(f"Подключились к вебинару: {entered_webinar} ({entered_webinar / entered_bot:.1%})")
print(f"Оставили заявку: {made_order} ({made_order / entered_webinar:.1%})")
```

```
==== ВОРОНКА ====
Всего пользователей: 1000
Зашли в бота: 1000 (100.0%)
Подключились к вебинару: 382 (38.2%)
Оставили заявку: 18 (4.7%)
```

```
stages = ['Вошли в бота', 'Подключились', 'Оставили заявку']
values = [entered_bot, entered_webinar, made_order]
conversion = [100, entered_webinar / entered_bot * 100, made_order / entered_webinar * 100]

plt.figure(figsize=(8, 5))
sns.barplot(x=stages, y=conversion, palette=palette_custom[:3])
plt.title('Конверсия по этапам воронки')
plt.ylabel('Конверсия (%)')
for i, val in enumerate(values):
    plt.text(i, conversion[i] / 2, f'n={val}', ha='center', color='black', weight='bold')
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-34-3735446822.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`

  sns.barplot(x=stages, y=conversion, palette=palette_custom[:3])
```

🔷 5. Подготовка данных по пользователям

```
# Подготовка сессий
webinar_start = df[df['action'] == 'enter_webinar_room']['created_at'].min()

user_stats = []
for user_id in df['user_id'].unique():
    user_data = df[df['user_id'] == user_id]
    if {'enter_webinar_room', 'leave_webinar_room'}.issubset(set(user_data['action'])):
        enter = user_data[user_data['action'] == 'enter_webinar_room']['created_at'].min()
        leave = user_data[user_data['action'] == 'leave_webinar_room']['created_at'].max()
        duration = (leave - enter).total_seconds() / 60
        order = 1 if 'make_order' in user_data['action'].values else 0
        user_stats.append({
            'user_id': user_id,
            'duration_min': duration,
            'made_order': order
        })

webinar_time = pd.DataFrame(user_stats)
```

🔷 6. Конверсия по длительности участия

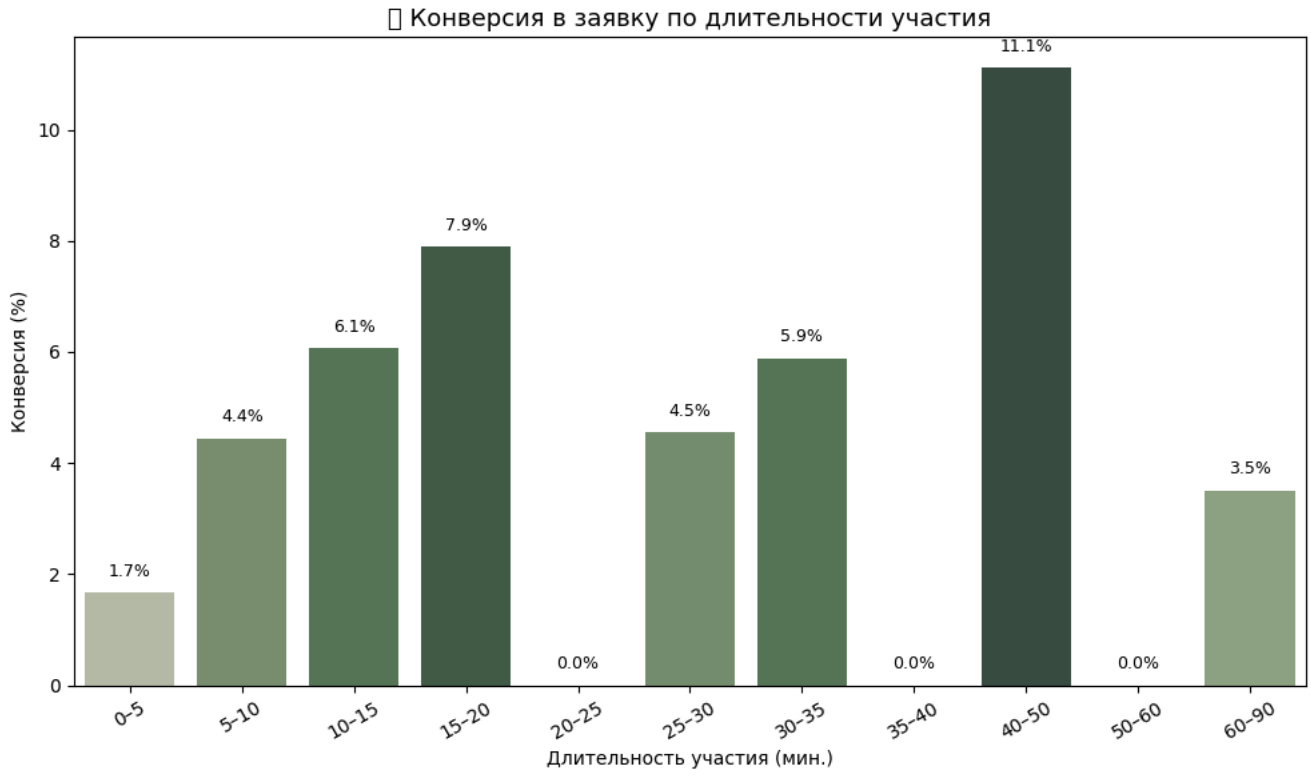
```
bins = [0, 5, 10, 15, 20, 25, 30, 35, 40, 50, 60, 90]
labels = ["0–5", "5–10", "10–15", "15–20", "20–25", "25–30",
          "30–35", "35–40", "40–50", "50–60", "60–90"]

webinar_time['duration_group'] = pd.cut(webinar_time['duration_min'], bins=bins, labels=labels, right=False)
grouped = webinar_time.groupby('duration_group')['made_order'].agg(['count', 'sum'])
grouped['conversion (%)'] = (grouped['sum'] / grouped['count'] * 100).round(2)
grouped = grouped.rename(columns={'count': 'Users', 'sum': 'Orders'})

print("\n📊 Конверсия в зависимости от длительности участия:")
print(grouped)
```

```
📊 Конверсия в зависимости от длительности участия:
                Users  Orders  conversion (%)
duration_group
0–5                60       1            1.67
5–10               45       2            4.44
10–15              33       2            6.06
15–20              38       3            7.89
20–25              18       0            0.00
25–30              22       1            4.55
30–35              17       1            5.88
35–40              15       0            0.00
40–50              27       3           11.11
50–60              20       0            0.00
```

```
60–90              57       2           3.51
/tmp/ipython-input-7-2123199912.py:7: FutureWarning: The default of observed=False is deprecated and will be changed to
  grouped = webinar_time.groupby('duration_group')['made_order'].agg(['count', 'sum'])
```

```
from matplotlib.colors import LinearSegmentedColormap

# Градиент по значениям конверсии — от min к max
colorscale = LinearSegmentedColormap.from_list("custom_green", palette_custom)

norm_values = (grouped["conversion (%)"] - grouped["conversion (%)"].min()) / (
    grouped["conversion (%)"].max() - grouped["conversion (%)"].min()
)
colors = [colorscale(val) for val in norm_values.fillna(0)]  # защита от NaN

# Построение графика
plt.figure(figsize=(10, 6))
bars = sns.barplot(x=grouped.index, y=grouped['conversion (%)'], palette=colors)

plt.title('📊 Конверсия в заявку по длительности участия', fontsize=13)
plt.xlabel('Длительность участия (мин.)')
plt.ylabel('Конверсия (%)')

# Подписи
for i, v in enumerate(grouped['conversion (%)']):
    bars.text(i, v + 0.3, f"{v:.1f}%", ha='center', fontsize=9)

plt.xticks(rotation=30)
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-8-2756991944.py:13: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`

  bars = sns.barplot(x=grouped.index, y=grouped['conversion (%)'], palette=colors)
/tmp/ipython-input-8-2756991944.py:24: UserWarning: Glyph 128202 (\N{BAR CHART}) missing from font(s) DejaVu Sans.
  plt.tight_layout()
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 128202 (\N{BAR CHART}) missin
  fig.canvas.print_figure(bytes_io, **kw)
```



🔷 7. Сегментация по поведению

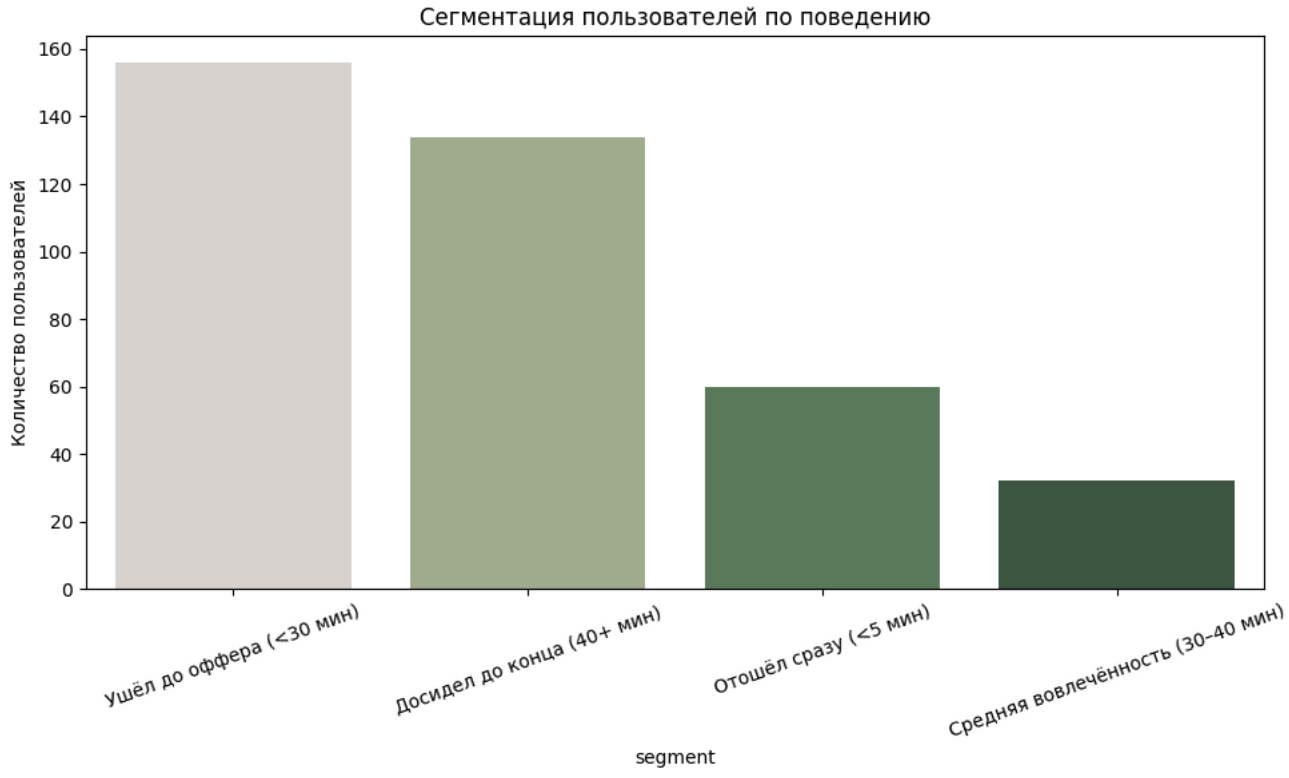
```
def behavior_segment(row):
    if row['duration_min'] < 5:
        return 'Отошёл сразу (<5 мин)'
    elif row['duration_min'] < 30:
        return 'Ушёл до оффера (<30 мин)'
    elif row['duration_min'] >= 40:
        return 'Досидел до конца (40+ мин)'
    else:
```

```
        return 'Средняя вовлечённость (30–40 мин)'

webinar_time['segment'] = webinar_time.apply(behavior_segment, axis=1)

behavior_summary = webinar_time.groupby('segment').agg(
    Users=('user_id', 'count'),
    Orders=('made_order', 'sum')
).sort_values(by='Users', ascending=False)

behavior_summary['conversion (%)'] = (behavior_summary['Orders'] / behavior_summary['Users'] * 100).round(2)
behavior_summary['percent_of_total'] = (behavior_summary['Users'] / webinar_time.shape[0] * 100).round(1)

print("\n📊 Сегментация по активности пользователей:\n")
print(behavior_summary)
```

```
📊 Сегментация по активности пользователей:

                                  Users  Orders  conversion (%)  \
segment
Ушёл до оффера (<30 мин)            156       8            5.13
Досидел до конца (40+ мин)          134       8            5.97
Отошёл сразу (<5 мин)                60       1            1.67
Средняя вовлечённость (30–40 мин)    32       1            3.12

                                  percent_of_total
segment
Ушёл до оффера (<30 мин)                      40.8
Досидел до конца (40+ мин)                    35.1
Отошёл сразу (<5 мин)                         15.7
Средняя вовлечённость (30–40 мин)              8.4
```

```
plt.figure(figsize=(10, 6))
sns.barplot(x=behavior_summary.index, y=behavior_summary['Users'], palette=palette_custom)
plt.title('Сегментация пользователей по поведению')
plt.ylabel('Количество пользователей')
plt.xticks(rotation=20)
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-10-67218628.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`

  sns.barplot(x=behavior_summary.index, y=behavior_summary['Users'], palette=palette_custom)
/tmp/ipython-input-10-67218628.py:2: UserWarning: The palette list has more values (5) than needed (4), which may not be
  sns.barplot(x=behavior_summary.index, y=behavior_summary['Users'], palette=palette_custom)
```



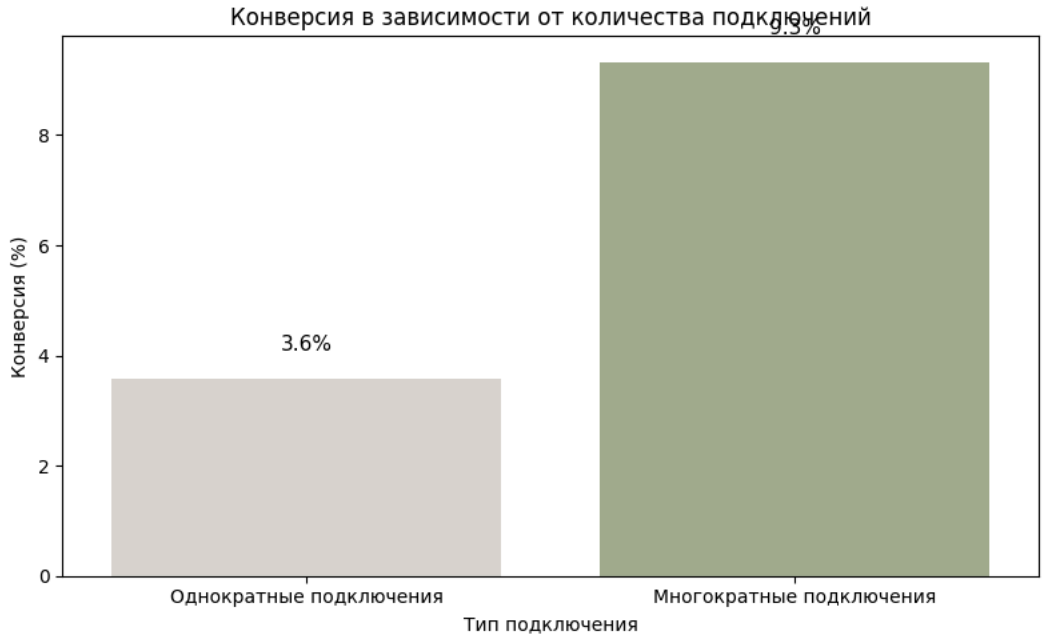
🔷 8. Повторные подключения

```
enter_counts = df[df['action'] == 'enter_webinar_room'].groupby('user_id').size().reset_index(name='enter_count')
webinar_time = webinar_time.merge(enter_counts, on='user_id', how='left')
webinar_time['multi_attempt'] = webinar_time['enter_count'] > 1

multi_summary = webinar_time.groupby('multi_attempt').agg(
    Users=('user_id', 'count'),
    Orders=('made_order', 'sum')
).reset_index()
multi_summary['conversion (%)'] = (multi_summary['Orders'] / multi_summary['Users'] * 100).round(2)
multi_summary['group_label'] = multi_summary['multi_attempt'].map({
    True: 'Многократные подключения',
    False: 'Однократные подключения'
})

print("\n🔁 Анализ по количеству подключений:\n")
print(multi_summary[['group_label', 'Users', 'conversion (%)']].rename(columns={'group_label': 'Тип подключения'}))
```

```
🔁 Анализ по количеству подключений:

             Тип подключения  Users  conversion (%)
0   Однократные подключения    307            3.58
1  Многократные подключения     75            9.33
```

```
plt.figure(figsize=(8, 5))
sns.barplot(data=multi_summary, x='group_label', y='conversion (%)', palette=palette_custom[:2])
plt.title('Конверсия в зависимости от количества подключений')
plt.xlabel('Тип подключения')
plt.ylabel('Конверсия (%)')
for i, val in enumerate(multi_summary['conversion (%)']):
    plt.text(i, val + 0.5, f"{val:.1f}%", ha='center', fontsize=11)
plt.tight_layout()
plt.show()
```

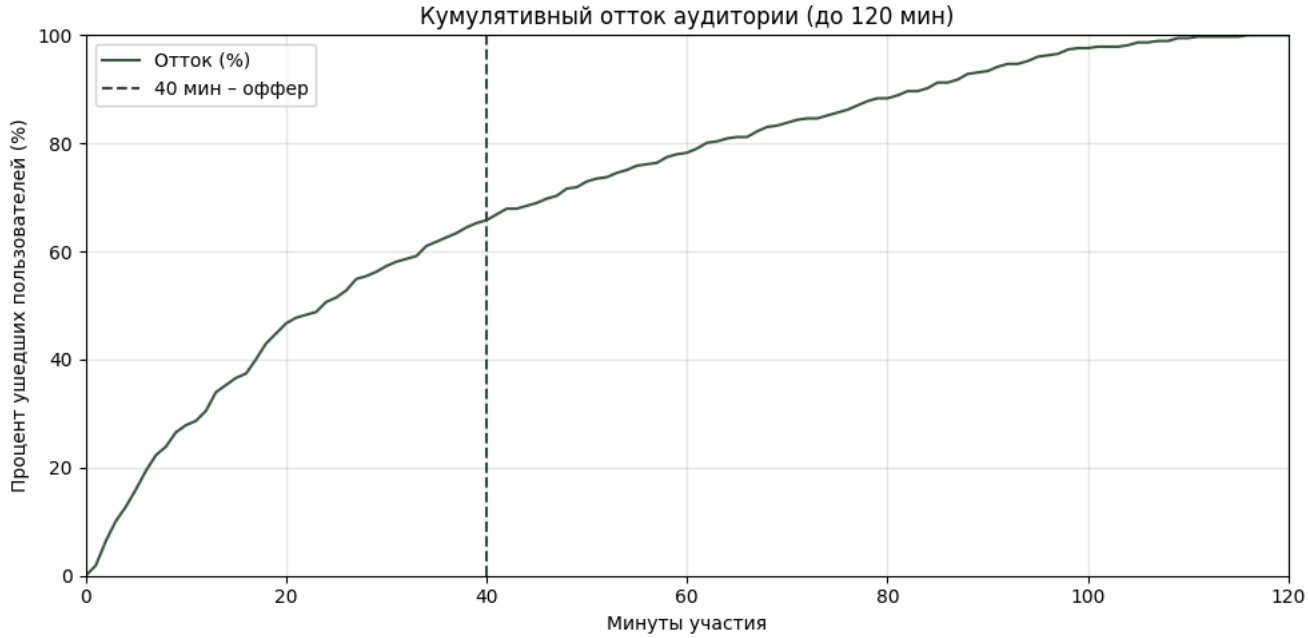
```
/tmp/ipython-input-12-4269214082.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue`

  sns.barplot(data=multi_summary, x='group_label', y='conversion (%)', palette=palette_custom[:2])
```



🔷 9. Retention и заявки после 40 мин

```
filtered_webinar_time = webinar_time[webinar_time['duration_min'] <= 120].copy()

# RETENTION CURVE
retention_curve = {}
for minute in range(0, 121):
    remained = (filtered_webinar_time['duration_min'] >= minute).sum()
    retention_curve[minute] = remained

retention_df = pd.DataFrame.from_dict(retention_curve, orient="index", columns=["remained"])
retention_df['retention_%'] = retention_df['remained'] / retention_df['remained'].iloc[0] * 100
retention_df['dropoff_%'] = 100 - retention_df['retention_%']
retention_df.index.name = 'minute'
```

```
dropoff_40 = retention_df.loc[40, 'dropoff_%']


plt.figure(figsize=(10, 5))
plt.plot(retention_df.index, retention_df['dropoff_%'], color=palette_custom[3], label='Отток (%)')
plt.axvline(40, color=palette_custom[4], linestyle='--', label='40 мин — оффер')
plt.title('Кумулятивный отток аудитории (до 120 мин)')
plt.xlabel('Минуты участия')
plt.ylabel('Процент ушедших пользователей (%)')
plt.xlim(0, 120)
plt.ylim(0, 100)
plt.legend()
plt.grid(alpha=0.3)
plt.tight_layout()
plt.show()
```




```
# Заявки после 40-й
orders_after_40 = webinar_time[webinar_time['duration_min'] >= 40]['made_order'].sum()
total_orders = webinar_time['made_order'].sum()
share_after_40 = orders_after_40 / total_orders * 100 if total_orders > 0 else 0

# Переменные для вывода
seg_ended = behavior_summary.loc['Досидел до конца (40+ мин)']
seg_left_early = behavior_summary.loc['Ушёл до оффера (<30 мин)']
multi_yes = multi_summary[multi_summary['multi_attempt'] == True].iloc[0]
multi_no = multi_summary[multi_summary['multi_attempt'] == False].iloc[0]
```

🔷 10. Выводы и рекомендации

```
# print("\n=== ВЫВОДЫ ===\n")

print(f"""
📌 📈 Сегмент «Досидел до конца (40+ мин)»:
   • Размер: {seg_ended['percent_of_total']}% аудитории
   • Конверсия: {seg_ended['conversion (%)']}%

📌 ⏱️ Сегмент «Ушёл до 30-й минуты»:
   • Размер: {seg_left_early['percent_of_total']}%
   • Конверсия: {seg_left_early['conversion (%)']}%

📌 🔁 Повторные подключения:
   • Однократные: {multi_no['Users']} — конверсия {multi_no['conversion (%)']}%
   • Многократные: {multi_yes['Users']} — конверсия {multi_yes['conversion (%)']}% (в ~{round(multi_yes['conversion (%)'] /

📉 К 40-й минуте уходит ≈ {dropoff_40:.1f}% пользователей
📊 Из всех заявок: {orders_after_40} из {total_orders} (≈{share_after_40:.1f}%) пришлись на период после 40-й минуты
""")
```

```
📌 📈 Сегмент «Досидел до конца (40+ мин)»:
   • Размер: 35.1% аудитории
   • Конверсия: 5.97%
```

📌 ⏱ Сегмент «Ушёл до 30-й минуты»:
• Размер: 40.8%
• Конверсия: 5.13%

📌 🔁 Повторные подключения:
• Однократные: 307 — конверсия 3.58%
• Многократные: 75 — конверсия 9.33% (в ~2.6 раза выше)

📉 К 40-й минуте уходит ≈ 65.8% пользователей
📊 Из всех заявок: 8 из 18 (≈44.4%) пришлись на период после 40-й минуты

```
# print("\n=== ГИПОТЕЗЫ И РЕКОМЕНДАЦИИ ===\n")

print(f"""
🧩 Гипотезы:
▶ На 40-й минуте находится оффер — рост заявок после этой точки это подтверждает.
▶ Однако ~{dropoff_40:.1f}% пользователей уходят раньше — не слышат о предложении.
▶ Повторные подключения — индикатор интереса, конверсия в 2–3 раза выше обычной.

💡 Рекомендации:
• Озвучить «тизер оффера» уже на 15–20 минуте.
• Повторить ключевые преимущества на 25–30 минуте.
• Досидевшим 30+ мин предложить бонус.
• Повторников — отловить и обработать отдельно.
• Ушедших до 10-й — ретаргетить повторной воронкой через TG/email.
""")
```

📥
🧩 Гипотезы:
▶ На 40-й минуте находится оффер — рост заявок после этой точки это подтверждает.
▶ Однако ~65.8% пользователей уходят раньше — не слышат о предложении.
▶ Повторные подключения — индикатор интереса, конверсия в 2–3 раза выше обычной.

💡 Рекомендации:
• Озвучить «тизер оффера» уже на 15–20 минуте.
• Повторить ключевые преимущества на 25–30 минуте.
• Досидевшим 30+ мин предложить бонус.
• Повторников — отловить и обработать отдельно.
• Ушедших до 10-й — ретаргетить повторной воронкой через TG/email.