빅데이터플랫폼부 인턴행원 염승원 개인과제

1. 데이터 불러오기

In [173…
```
import sys
import os
import teradataml as tdml
from teradataml import *
import pandas as pd
import pandas as read_pd
import getpass
sys.path.append(os.path.abspath('/home/woori/JupyterLabRoot/'))  # 지우지 않고 사
from Config.EduInfoModule import getDBHost, getDBUserId, getUserPass

DatabaseHost = str(getDBHost())
SVC = str(getDBUserId())
SVCP = str(getUserPass())

## DB 연결
create_context(host=DatabaseHost, username=SVC, password=SVCP)

## EDUSVC(교육용 DB)에 있는 DIGITAL_LOG 데이터를 조회하고, 결과를 데이터프레임에
dt_log = DataFrame.from_query(""" select * from DIGITAL_LOG """).to_pandas(all_r
step_def = DataFrame.from_query(""" select * from STEP_DEFINITION """).to_pandas
ctns_scrn_list = DataFrame.from_query(""" select * from CTNS_SCRN_LIST """).to_p
```

```
/usr/local/lib/python3.8/dist-packages/teradataml/context/context.py:462: UserWar
ning:

[Teradata][teradataml](TDML_2002) Overwriting an existing context associated with
Teradata Vantage Connection. Most of the operations on any teradataml DataFrames
created before this will not work.

/usr/local/lib/python3.8/dist-packages/teradataml/context/context.py:484: Teradat
aMlRuntimeWarning:

Warning: Password is URL encoded.
```

2. 데이터 살펴보기

dt_log

In [173…
```
#dt_log.head()
```

In [173…
```
dt_log.shape
```

Out[173… (601247, 11)

In [173…
```
dt_log.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 601247 entries, 0 to 601246
Data columns (total 11 columns):
 #   Column               Non-Null Count   Dtype 
---  ------               --------------   ----- 
 0   bas_dt               601247 non-null  int64 
 1   dt_log_sess_id       601247 non-null  object
 2   dt_log_evnt_id       601247 non-null  object
 3   page_id              601247 non-null  object
 4   plm_pdcd             178788 non-null  object
 5   prd_nm               178655 non-null  object
 6   dt_log_evnt_no       4218 non-null    object
 7   infw_parm_id         9306 non-null    object
 8   clck_btn_txt         301316 non-null  object
 9   clck_btn_id          43852 non-null   object
 10  user_log_occ_dtm_txt  601247 non-null  object
dtypes: int64(1), object(10)
memory usage: 50.5+ MB
```

step_def

In [173…
```
#step_def.head()
```

In [173…
```
step_def.shape
```

Out[173… (761, 9)

In [174…
```
step_def.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 761 entries, 0 to 760
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   seq             761 non-null    int64 
 1   tgt_dscd        761 non-null    object
 2   tgt_id          761 non-null    object
 3   tgt_nm          761 non-null    object
 4   step_seq        761 non-null    int64 
 5   step_nm         761 non-null    object
 6   step_map_seq    761 non-null    int64 
 7   dt_log_evnt_id  761 non-null    object
 8   page_id         761 non-null    object
dtypes: int64(3), object(6)
memory usage: 53.6+ KB
```

ctns_scrn_list

In [174…
```
#ctns_scrn_list.head()
```

In [174…
```
ctns_scrn_list.shape
```

Out[174… (3585, 68)

3. 데이터 전처리

In [174…
```
# 라이브러리 불러오기
import pandas as pd
import numpy as np
from datetime import datetime
```

dt_log

In [174…
```
# 필요한 컬럼만 추출
dt_log = dt_log.drop(['dt_log_evnt_no', 'infw_parm_id',
                      'clck_btn_txt', 'clck_btn_id'], axis=1)
dt_log.shape
```

Out[174… (601247, 7)

In [174…
```
#dt_log.head()
```

In [174…
```
# dt_log에서 비정상적인 데이터를 지닌 세션ID 통채로 제거
delete_sessions = ["3C3913D5-9036-46D1-8658-8479285A937E", "264D03DE-42CE-42AA-9
"54064FA7-7ED2-4BAE-9BDB-29F2364E441D", "9529F1E8-573D-4DF2-B9F8-8A545EE50689",
"52E3F71A-F741-48A4-8D61-1A6A0D84D4FF", "B928ACD1-6C73-4357-ABC6-71057D67A183",
"7F9939F9-DCD6-4E29-AF49-E7B970D298A2", "3F57388E-026B-4546-9B4F-8A9C01E1CD22"]
dt_log = dt_log[~dt_log["dt_log_sess_id"].isin(delete_sessions)]
```

In [174…
```
# 상품 중심의 분석 -> 상품코드가 존재하는 것만 추출
valid_sessions = dt_log[dt_log['plm_pdcd'].notna()]['dt_log_sess_id'].unique()
dt_log = dt_log[dt_log['dt_log_sess_id'].isin(valid_sessions)]
#dt_log.head()
```

In [174…
```
# page 기준으로 분석
dt_log = dt_log[dt_log['dt_log_evnt_id'] == 'page']
#dt_log.head()
```

In [174…
```
# 시간 데이터 변환 및 정렬
dt_log['user_log_occ_dtm_txt'] = pd.to_datetime(dt_log['user_log_occ_dtm_txt'],
dt_log = dt_log.sort_values(by=['bas_dt', 'dt_log_sess_id', 'user_log_occ_dtm_tx
```

In [175…
```
#dt_log.head()
```

In [175…
```
print(dt_log.shape)
```

```
(239436, 7)
```

step_def

In [175…
```
# 필요한 데이터만 추출 # 상품에 초점 맞춰서 볼 것이기 때문에 'tgt_dscd'가 'PR'(상
step_def = step_def[step_def['tgt_dscd'] == 'PR']
#step_def.head()
```

In [175…
```
# dt_log에서의 'plm_pdcd'가 P110000041이라면 step_def에서의 'tgt_id'는 P11000004
step_def['tgt_id'] = step_def['tgt_id'].str.replace(r'\d$', '', regex=True)
#step_def.head()
```

In [175…
```
print(step_def.shape)
```

```
(520, 9)
```

In [175…
```
# dt_log에는 있는 상품이 step_def에는 없는 경우 발생 => 단계가 없는 것
dt_log_products = dt_log['plm_pdcd'].dropna().unique()
step_def_products = step_def['tgt_id'].unique()
missing_products = set(dt_log_products) - set(step_def_products)
print('단계정의 데이터에 없는 상품 코드 목록 : ', missing_products)

dt_log = dt_log[~dt_log['plm_pdcd'].isin(missing_products)]
```

```
단계정의 데이터에 없는 상품 코드 목록 :  {'P050000682', 'P040001871', 'P05000069
4', 'P030000019', 'P030000013', 'P020006574', 'P050000646', 'P050000677', 'P02000
0121', 'P130008715', 'P040002443', 'P040011200', 'P020006425', 'P050000593', 'P05
0000039', 'P040003168', 'P030000082', 'P030000074', 'P040008221', 'P020006666',
'P010002359', 'P130011592', 'P040005245', 'P110000044', 'P030000015', 'P01000229
3', 'P040009898', 'P040008100', 'P040003286', 'P030000089', 'P120000159', 'P04000
8899', 'P110000045', 'P030000014', 'P130012550', 'P040003659', 'P020006500', 'P04
0006068', 'P040001351', 'P050000684', 'P130011611', 'P010000013', 'P040007265',
'P040010804', 'P050000040', 'P030000017', 'P130013337', 'P010002283', 'P01000240
3', 'P010002373', 'P100000001', 'P020000118', 'P010002482'}
```

In [175…
```
dt_log.shape
```

Out[175…
```
(238592, 7)
```

ctns_scrn_list

In [175…
```
ctns_scrn_list = ctns_scrn_list[['scrn_id','exps_scrn_nm']]
```

4. 데이터 병합 및 최종 도달 여부 설정

In [175…
```
# 단계 정의 데이터 병합 및 최종 도달 여부 설정
def merge_with_steps(dt_log, step_def, ctns_scrn_list):
    # 1. 상품코드가 있는 경우 처리
    with_product = dt_log[dt_log['plm_pdcd'].notna()].copy()
    step_def_subset = step_def[['tgt_dscd','tgt_id', 'tgt_nm', 'step_seq', 'step
    with_product = pd.merge(with_product, step_def_subset, how='left', left_on =
    with_product['page_nm'] = with_product['tgt_nm']

    # 2. 상품코드가 없는 경우 처리
    without_product = dt_log[dt_log['plm_pdcd'].isna()].copy()
    without_product = pd.merge(without_product, ctns_scrn_list, how='left', left
    without_product['page_nm'] = without_product['exps_scrn_nm']

    merged_data = pd.concat([with_product, without_product], ignore_index=True)

    # 최종 단계 여부 설정
    final_step = merged_data.groupby('tgt_id', dropna=True)['step_seq'].max().re
    final_step.rename(columns={'step_seq': 'final_step_seq'}, inplace=True)
    merged_data = pd.merge(merged_data, final_step, on='tgt_id', how='left')
    merged_data['is_final_stage'] = merged_data['step_seq'] == merged_data['fina

    return merged_data
```

In [175…
```
#merge_with_steps(dt_log, step_def, ctns_scrn_list).head()
```

- 단계 정의가 된 데이터만 필터링

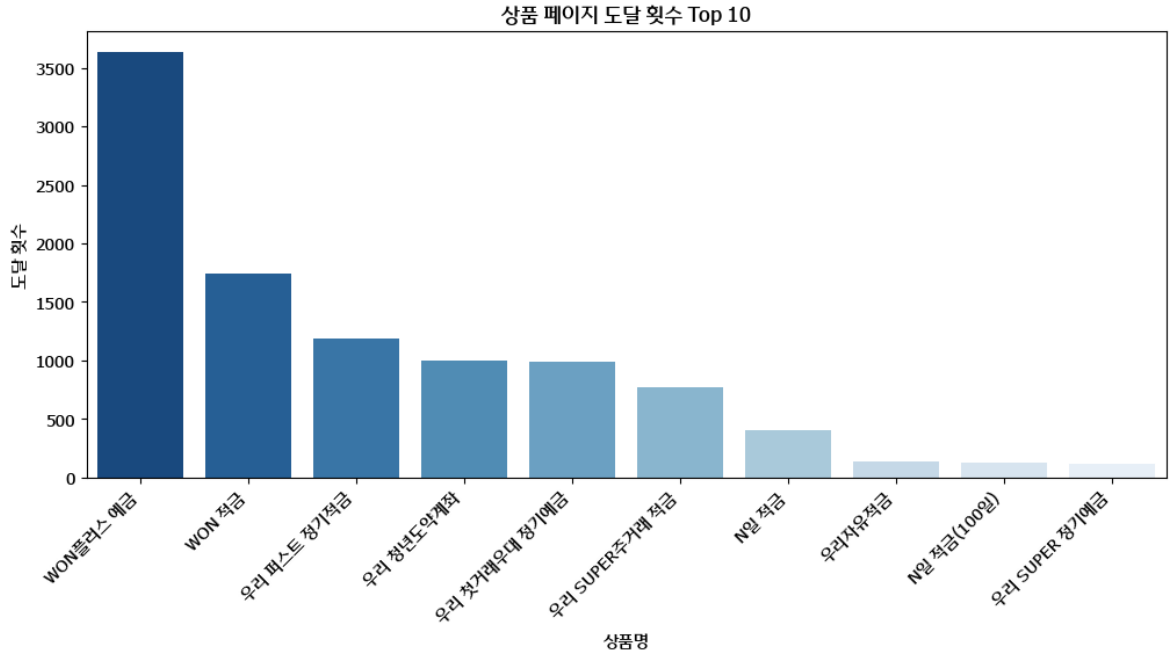
In [176…
```
merged_data = merge_with_steps(dt_log, step_def, ctns_scrn_list)
merged_data = merged_data[merged_data['step_seq'].notna()]
merged_data.shape
```

Out[176… (51839, 17)

5. 상품 페이지별 도달 횟수

In [176…
```
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.font_manager as fm
plt.rcParams['font.family']='Wooridaum'

def plot_product_reach_distribution(merged_data,top_n = 10):
    product_counts = merged_data.groupby('tgt_nm')['dt_log_sess_id'].nunique().r
    product_counts.rename(columns={'dt_log_sess_id':'reach_count'}, inplace=True

    product_counts = product_counts.sort_values(by='reach_count', ascending=Fals
    if top_n:
        product_counts = product_counts.head(top_n)

    plt.figure(figsize=(12,5))
    sns.barplot(x='tgt_nm', y='reach_count', data=product_counts,palette='Blues_
    plt.xlabel('상품명')
    plt.ylabel('도달 횟수')
    plt.title('상품 페이지 도달 횟수 Top 10')
    plt.xticks(rotation=45, ha='right')
    plt.show()
```

In [176…
```
plot_product_reach_distribution(merged_data,10)
```



In [176…
```
product_counts = merged_data.groupby('tgt_nm')['dt_log_sess_id'].nunique().reset
product_counts.rename(columns={'dt_log_sess_id':'reach_count'}, inplace=True)
```

```
product_counts.sort_values(by='reach_count', ascending=False).head(10)
```

Out[176…

	tgt_nm	reach_count
9	WON플러스 예금	3633
7	WON 적금	1746
34	우리 퍼스트 정기적금	1192
31	우리 청년도약계좌	995
29	우리 첫거래우대 정기예금	992
18	우리 SUPER주거래 적금	767
0	N일 적금	404
40	우리자유적금	142
1	N일 적금(100일)	132
17	우리 SUPER 정기예금	120

6. 단계별 고객 이탈률 분석

In [176…

```
df_sumy = merged_data.groupby(['dt_log_sess_id', 'plm_pdcd', 'prd_nm', 'step_sec
df_sumy = merged_data.groupby(['dt_log_sess_id', 'plm_pdcd', 'prd_nm']).agg(max_

#df_sumy.head()
```

In [176…

```
df_sumy.shape
```

Out[176…

```
(11001, 4)
```

In [176…

```
df_sumy1 = df_sumy.groupby(['plm_pdcd', 'prd_nm', 'max_step_id']).size().reset_i
df_sumy1.sort_values(by = ['plm_pdcd', 'prd_nm', 'max_step_id'],
                     ignore_index = True,
                     inplace= True)
df_sumy1.shape
```

Out[176…

```
(141, 4)
```

In [176…

```
df_sumy1[:5]
```

Out[176…

	plm_pdcd	prd_nm	max_step_id	0
0	P010000011	우리 SUPER주거래 통장	5.0	37
1	P010000011	우리 SUPER주거래 통장	6.0	2
2	P010000034	우리CUBE통장	1.0	2
3	P010000109	우리 SUPER주거래 적금	1.0	2
4	P010000109	우리 SUPER주거래 적금	2.0	338

In [176…

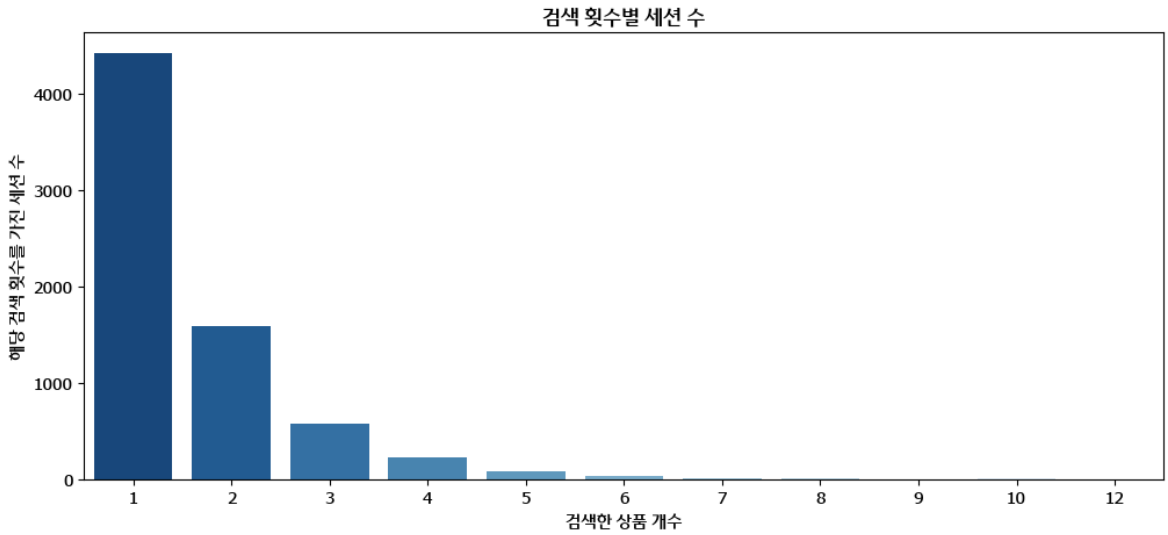
```
df_sumy1.to_csv('/home/woori/JupyterLabRoot/test_file.txt', sep = '\t', index =
```

7. 상품 검색 횟수별 분석

- 상품 검색 횟수별 세션 수

In [176…

```
# 각 세션별 상품 검색 수 계산
session_search_counts = merged_data.groupby('dt_log_sess_id')['plm_pdcd'].nuniqu
session_search_counts.rename(columns={'plm_pdcd': 'search_count'}, inplace=True)

# 검색 횟수별 전체 세션 수 집계
search_total_sessions = session_search_counts.groupby('search_count')['dt_log_se
search_total_sessions.rename(columns={'dt_log_sess_id': 'total_sessions'}, inpla

plt.figure(figsize=(12,5))
sns.barplot(x='search_count', y='total_sessions', data=search_total_sessions, pa
plt.xlabel('검색한 상품 개수')
plt.ylabel('해당 검색 횟수를 가진 세션 수')
plt.title('검색 횟수별 세션 수')
plt.show()
```



In [177…

```
search_total_sessions
```

Out[177…

	search_count	total_sessions
0	1	4424
1	2	1594
2	3	575
3	4	226
4	5	78
5	6	31
6	7	11
7	8	7
8	9	1
9	10	3
10	12	1

- 상품 검색 횟수별 전환율

In [177…

```
# 검색 횟수별 최종 단계 도달 세션 수 집계
final_stage_sessions = merged_data[merged_data['is_final_stage']].groupby('dt_lo
final_stage_sessions.rename(columns={'plm_pdcd': 'search_count'}, inplace=True)

final_stage_counts = final_stage_sessions.groupby('search_count')['dt_log_sess_i
final_stage_counts.rename(columns={'dt_log_sess_id': 'final_stage_sessions'}, in

final_stage_counts
```

Out[177…

	search_count	final_stage_sessions
0	1	3952
1	2	480
2	3	52
3	4	8
4	5	2

In [177…

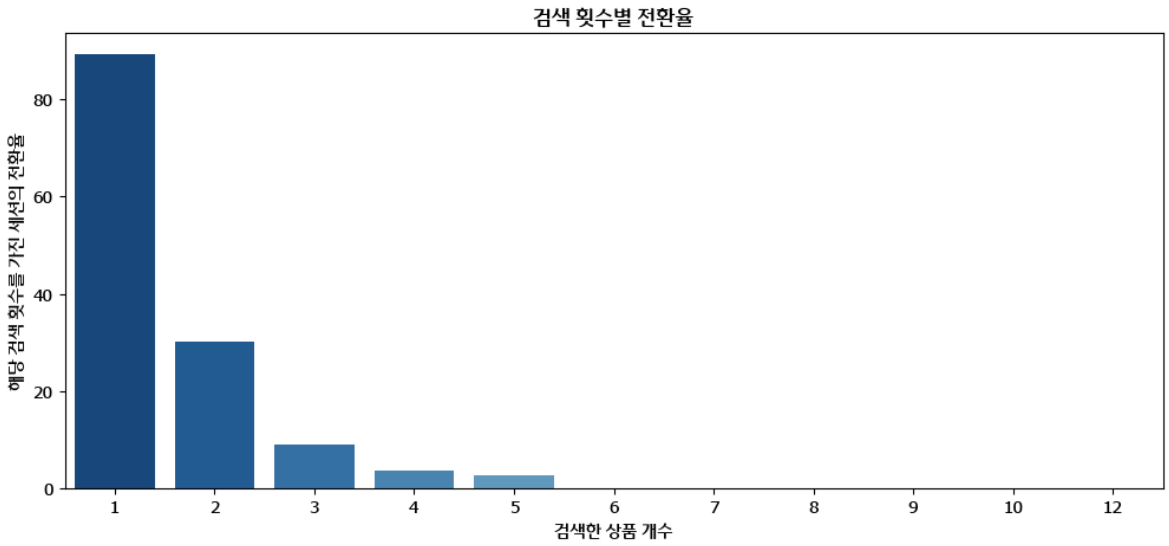
```
# 전환율 계산
conversion_by_search_df = search_total_sessions.merge(final_stage_counts, on='se
conversion_by_search_df['final_stage_sessions'].fillna(0, inplace=True)
conversion_by_search_df['conversion_rate'] = (conversion_by_search_df['final_sta

conversion_by_search_df.sort_values(by='search_count')
```

Out[177…

	search_count	total_sessions	final_stage_sessions	conversion_rate
0	1	4424	3952.0	89.330922
1	2	1594	480.0	30.112923
2	3	575	52.0	9.043478
3	4	226	8.0	3.539823
4	5	78	2.0	2.564103
5	6	31	0.0	0.000000
6	7	11	0.0	0.000000
7	8	7	0.0	0.000000
8	9	1	0.0	0.000000
9	10	3	0.0	0.000000
10	12	1	0.0	0.000000

In [177…

```
plt.figure(figsize=(12,5))
sns.barplot(x='search_count', y='conversion_rate', data=conversion_by_search_df,
plt.xlabel('검색한 상품 개수')
plt.ylabel('해당 검색 횟수를 가진 세션의 전환율')
plt.title('검색 횟수별 전환율')
plt.show()
```



부록

- 경로 생성

세션별 이전/다음 page_nm과 page_id를 식별해 유입 경로 파악

In [177…

```
def generate_paths(merged_data):
    merged_data['prev_nm'] = merged_data.groupby('dt_log_sess_id')['page_nm'].sh
```

```
    merged_data['next_nm'] = merged_data.groupby('dt_log_sess_id')['page_nm'].sh
    merged_data['prev_page_id'] = merged_data.groupby('dt_log_sess_id')['page_ic
    merged_data['next_page_id'] = merged_data.groupby('dt_log_sess_id')['page_ic
    return merged_data
```

In [177…
```
#generate_paths(merged_data).head()
```

In [177…
```
merged_data = generate_paths(merged_data)
```

- 상품별 유입 경로 분석

특정 상품코드를 본 세션들만 추려, 해당 세션에서 (prev_nm -> page_nm) 경로 파악.

In [177…
```
# 상품 분석
### 특정 상품 세션만 골라서 유입 경로 분석
def analyze_inflow_by_product(merged_data, product_name, top_n=20):
    # 1) 해당 상품을 본 세션ID
    product_sessions = merged_data.loc[merged_data['page_nm'] == product_name, '
    # 2) 그 세션 내 모든 로그 (상품코드 없는 페이지도 포함)
    product_flow = merged_data[merged_data['dt_log_sess_id'].isin(product_sessic
    # 3) (prev_nm, page_nm) 집계
    transitions = product_flow.groupby(['prev_nm','page_nm']).size().reset_index
    transitions = transitions.nlargest(top_n, 'count')
    return transitions

analyze_inflow_by_product(merged_data, '우리 퍼스트 정기적금', 10)
```

Out[177…

	prev_nm	page_nm	count
213	우리 퍼스트 정기적금	우리 퍼스트 정기적금	3203
69	WON플러스 예금	WON플러스 예금	1262
36	WON 적금	WON 적금	1153
124	우리 SUPER주거래 적금	우리 SUPER주거래 적금	556
173	우리 첫거래우대 정기예금	우리 첫거래우대 정기예금	524
199	우리 퍼스트 정기적금	WON 적금	305
49	WON 적금	우리 퍼스트 정기적금	284
236	우리자유적금	우리자유적금	227
185	우리 청년도약계좌	우리 청년도약계좌	173
210	우리 퍼스트 정기적금	우리 첫거래우대 정기예금	162

In [177…
```
def analyze_pre_inflow(merged_data, target_product, top_n=20):
    """
    특정 상품 페이지에 도달하기 전의 유입 경로 분석
    - 해당 상품을 방문한 세션을 찾고
    - 그 세션에서 특정 상품 페이지 직전(`prev_nm`)을 분석
    - Sankey Diagram으로 시각화
    Parameters:
        merged_data  : 전처리된 병합 데이터 (dt_log + step_def + ctns_scrn_list)
        target_product : 분석할 상품명 (예: 'N일 적금')
        top_n        : 상위 N개의 유입 경로를 표시 (기본 20개)
```

```
    Returns:
        fig : plotly Sankey Diagram
    """
    # 1) 특정 상품이 등장한 세션 찾기
    target_sessions = merged_data.loc[merged_data['tgt_nm'] == target_product, '
    # 2) 그 세션에서 특정 상품 페이지가 등장하기 전의 데이터 찾기
    pre_inflow_data = merged_data[
        (merged_data['dt_log_sess_id'].isin(target_sessions)) &  # 해당 상품을 본
        (merged_data['page_nm'] == target_product)               # 특정 상품 페이
    ]
    # 3) 유입 경로 집계 (prev_nm → page_nm 전환 횟수)
    inflow_transitions = pre_inflow_data.groupby(['prev_nm', 'page_nm']).size().
    # NaN 값 제거 (처음 방문한 경우 prev_nm이 NaN일 수 있음)
    inflow_transitions.dropna(subset=['prev_nm', 'page_nm'], inplace=True)
    # 4) 상위 N개 경로만 선택
    inflow_transitions = inflow_transitions.nlargest(top_n, 'count')
    return inflow_transitions

analyze_pre_inflow(merged_data, '우리 퍼스트 정기적금', 10)
```

Out[177…

	prev_nm	page_nm	count
17	우리 퍼스트 정기적금	우리 퍼스트 정기적금	3203
4	WON 적금	우리 퍼스트 정기적금	284
15	우리 첫거래우대 정기예금	우리 퍼스트 정기적금	95
6	WON플러스 예금	우리 퍼스트 정기적금	80
11	우리 SUPER주거래 적금	우리 퍼스트 정기적금	52
16	우리 청년도약계좌	우리 퍼스트 정기적금	44
0	N일 적금	우리 퍼스트 정기적금	18
5	WON 통장	우리 퍼스트 정기적금	10
21	우리페이 적금	우리 퍼스트 정기적금	8
14	우리 국민연금 우대 통장	우리 퍼스트 정기적금	7

- 상품 중심 전환율 분석

In [177…

```
### 상품 중심 전환율 분석
def analyze_product_conversion(merged_data):
    """
    상품코드가 있는 세션만 필터링해서
    단계별 전환율 등을 계산.
    """

    # 세션 중 상품코드가 1개라도 있으면 전체 세션 유지
    product_data = merged_data[merged_data['plm_pdcd'].notna()].copy()

    # (tgt_id, dt_log_sess_id)별 최대 step_seq -> 최종단계
    final_step = product_data.groupby(['tgt_id','dt_log_sess_id'])['step_seq'].m
    final_step.rename(columns={'step_seq':'max_step_seq'}, inplace=True)
    product_data = pd.merge(product_data, final_step, on=['tgt_id','dt_log_sess_
    product_data['is_final'] = product_data['step_seq'] == product_data['max_ste
```

```
    # 상품단계별 세션 수
    step_counts = product_data.groupby(['tgt_id','tgt_nm','step_seq','step_nm'])
    step_counts.rename(columns={'dt_log_sess_id':'user_count'}, inplace=True)
    step_counts = step_counts[step_counts['user_count']>=10]
    step_counts.sort_values(['tgt_id','step_seq'], inplace=True)

    # 전환율 계산 (연속 단계)
    step_counts['next_user_count'] = step_counts.groupby(['tgt_id'])['user_count
    step_counts['conversion_rate'] = step_counts['next_user_count'] / step_count
    return step_counts

product_step_counts = analyze_product_conversion(merged_data)
product_step_counts.sort_values('conversion_rate', ascending=True).head(10)
```

Out[177…

	tgt_id	tgt_nm	step_seq	step_nm	user_count	next_user_count	conversion_
114	P010002512	우리 청년도약계좌	3.0	가입자격 안내	711	128.0	18.00
158	P060000049	개인형 퇴직연금(IRP)	1.0	유입	37	11.0	29.72
82	P010002483	우리 WON 파킹통장	1.0	상품 상세	37	12.0	32.43
60	P010002406	우리 Magic 적금 by 롯데카드	1.0	상품 상세	47	17.0	36.17
55	P010002402	우리 SUPER 정기예금	1.0	이자 시뮬레이션	118	43.0	36.44
71	P010002425	우리페이 적금	1.0	상품 상세	50	20.0	40.00
92	P010002487	우리 첫거래우대 정기예금	4.0	약관동의	511	205.0	40.11
89	P010002487	우리 첫거래우대 정기예금	1.0	이자 시뮬레이션	966	415.0	42.96
66	P010002408	첫급여 우리적금	1.0	이자 시뮬레이션	66	29.0	43.93
108	P010002507	우리 퍼스트 정기적금	2.0	가입 준비중	1143	505.0	44.18

- Sankey Diagram 시각화

In [178…

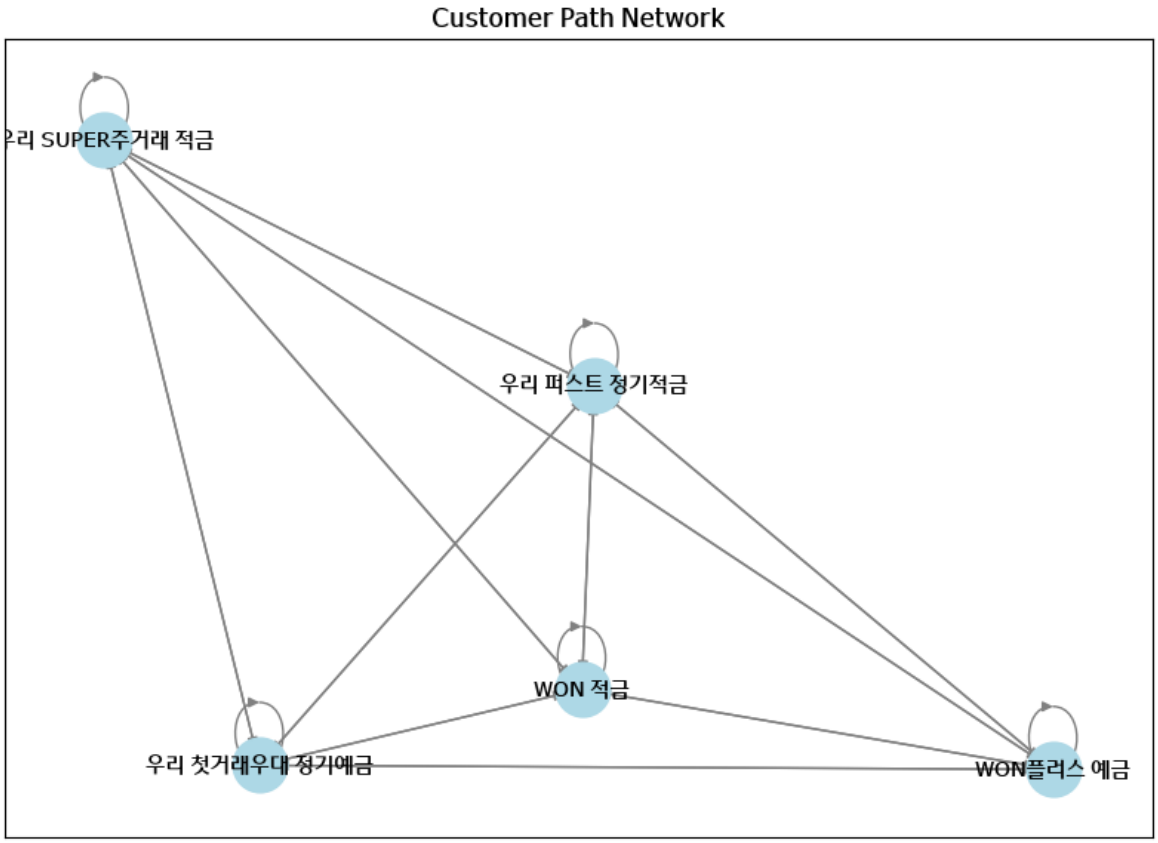
```
import plotly.graph_objects as go
def create_sankey_from_pagenm(merged_data, product_name, top_n=30):
    # 1) 특정 상품명을 본 세션ID 추출
    #    (tgt_nm == product_name) & plm_pdcd.notna()  # 상품 로그만
    target_sessions = merged_data.loc[
    (merged_data['tgt_nm'] == product_name) & (merged_data['plm_pdcd'].notna()),
```

```
    'dt_log_sess_id'
    ].unique()

    # 2) 해당 세션의 전체 로그 -> 상품코드 없는 페이지도 포함 (유입경로 보기)
    product_flow = merged_data[merged_data['dt_log_sess_id'].isin(target_sessior
    # 3) prev_nm -> page_nm 경로의 등장 횟수 계산
    transitions = product_flow.groupby(['prev_nm','page_nm']).size().reset_index
    # prev_nm 또는 page_nm이 NaN인 행 제거 (세션 첫 페이지 등)
    transitions.dropna(subset=['prev_nm','page_nm'], inplace=True)
    # 4) 상위 N개 경로만 표시
    transitions = transitions.nlargest(top_n, 'count')
    # 5) Sankey Diagram 생성
    all_nodes = list(set(transitions['prev_nm']).union(set(transitions['page_nm'
    node_indices = {node: i for i, node in enumerate(all_nodes)}
    transitions['source'] = transitions['prev_nm'].map(node_indices)
    transitions['target'] = transitions['page_nm'].map(node_indices)
    transitions['value'] = transitions['count']
    fig = go.Figure(go.Sankey(
        node=dict(
            pad=15,
            thickness=20,
            line=dict(color='black', width=0.5),
            label=all_nodes
        ),
        link=dict(
            source=transitions['source'],
            target=transitions['target'],
            value=transitions['value']
        )
    ))
    fig.update_layout(
        title_text=f"Sankey Diagram for '{product_name}' (Page Flow)",
        font_size=10
    )
    return fig

# 특정 상품명 예: 'N일 적금'
product_name = "우리 퍼스트 정기적금"
# Sankey 생성
fig = create_sankey_from_pagenm(merged_data, product_name, top_n=20)
fig.show()
```

- 네트워크 그래프 시각화

In [178…
```
import networkx as nx
import matplotlib.pyplot as plt
import matplotlib.font_manager as fm

# 네트워크 그래프 시각화
def visualize_path_graph(transitions):
    top_pages = transitions['prev_nm'].value_counts().nlargest(5).index
    transitions = transitions[transitions['prev_nm'].isin(top_pages) & transitio

    G = nx.DiGraph()
    for _, row in transitions.iterrows():
        G.add_edge(row['prev_nm'], row['page_nm'], weight=row['count'])

    pos = nx.spring_layout(G)
    plt.figure(figsize=(10, 7))
    nx.draw_networkx_nodes(G, pos, node_size=700, node_color='lightblue')
    nx.draw_networkx_edges(G, pos, edge_color='gray', arrows=True)
    nx.draw_networkx_labels(G, pos,  font_family='Wooridaum', font_size=10, font
    plt.title('Customer Path Network')
    plt.show()
```

In [178…
```
visualize_path_graph(transitions)
```


Customer Path Network
우리 SUPER주거래 적금
우리 퍼스트 정기적금
WON 적금
우리 첫거래우대정기예금
WON플러스 예금