### ****Logic for Calculating the Happy Path****

1. **Group by Journey**:
   * For each journey\_name, group the dataset so that we can analyze user navigation paths within a single journey.
2. **Calculate Transition Frequencies**:
   * Count the frequency of transitions from one page to the corresponding next\_page.
   * Sort these transitions in descending order of frequency to prioritize the most common paths.
3. **Initialize Happy Path Construction**:
   * Start the happy path with the most common page as the first page.
   * Track the previous\_page to ensure transitions are validated using the page\_referrer.
4. **Iterative Transition Building**:
   * In each iteration, add the current page to the happy\_path.
   * Check the is\_exit flag for the current page:
     + If is\_exit = 1, append "Exit" to the path and terminate the loop.
   * Otherwise, find the most common valid transition using the following criteria:
     + The transition must originate from the current page.
     + The page\_referrer must match the previous\_page.
5. **Handle Empty Transitions**:
   * If no valid next pages exist after validation, assume the user exits the journey and append "Exit" to the path.
6. **Update the Current Page**:
   * Update current\_page to the most frequent valid next\_page for the next iteration.
   * Set previous\_page to the current page.
7. **Return the Final Happy Path**:
   * Construct the final happy path as a string with arrows (->) indicating the sequence of pages.

### # Group by journey and calculate transitions with referrer validation and is\_exit logic

def calculate\_happy\_path(group):

# Calculate transition frequencies for the journey

transitions = group.groupby(['page', 'next\_page']).size().reset\_index(name='count')

transitions = transitions.sort\_values('count', ascending=False)

# Start constructing the happy path

happy\_path = []

current\_page = transitions.iloc[0]['page'] # Start with the most common first page

previous\_page = None

while True:

# Add the current page to the happy path

happy\_path.append(current\_page)

# Check if the current page is an exit page

if group[group['page'] == current\_page]['is\_exit'].iloc[0] == 1:

happy\_path.append('Exit')

break

# Get the most common valid transition from the current page

next\_pages = transitions[(transitions['page'] == current\_page) &

(group['page\_referrer'] == previous\_page)] # Ensure referrer validation

# If no valid next pages exist, terminate the loop

if next\_pages.empty:

happy\_path.append('Exit')

break

# Update the current and previous pages

previous\_page = current\_page

current\_page = next\_pages.iloc[0]['next\_page']

return " -> ".join(happy\_path)

# Apply the logic journey-wise

happy\_paths = df\_sorted.groupby('journey\_name').apply(calculate\_happy\_path).reset\_index(name='happy\_path')

# Display Happy Paths

print("Happy Paths for Each Journey:")

print(happy\_paths)