**Group and Calculate Transitions**:****

* + Group the data by page and next\_page to calculate the frequency of transitions.
  + Sort transitions by count in descending order to prioritize the most common paths.

1. **Start with the Most Common First Page**:
   * Initialize the current\_page with the page that has the highest transition frequency.
2. **Construct the Path Iteratively**:
   * Add the current\_page to the happy\_path.
   * Check the is\_exit column to determine if the current\_page is an exit page. If is\_exit == 1, append "Exit" and terminate.
3. **Validate Transitions Using** page\_referrer:
   * Ensure that the page\_referrer matches the previous\_page to validate the transition.
   * Use this condition to filter valid next pages:

python

CopyEdit

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

(group['page\_referrer'] == previous\_page)]

1. **Handle Empty Transitions**:
   * If no valid next pages exist, append "Exit" to the happy\_path and break the loop.
2. **Return the Happy Path**:
   * Convert the happy\_path list into a string with arrows (->) indicating the sequence of pages.

### # Function to calculate the happy path

def calculate\_happy\_path(group):

# Check if the group is empty

if group.empty:

return "No Data"

# Calculate transition frequencies for the journey

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

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

# Ensure transitions are valid

if transitions.empty:

return "No Valid Transitions"

# 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

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

if is\_exit == 1: # Terminate if the current page is marked as exit

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)] # Validate page\_referrer

# 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)

# Sort data by user journey

df\_sorted = df.sort\_values(['journey\_name', 'account\_num', 'exit\_time'])

# Apply the logic journey-wise

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

# Display the happy paths

print("Happy Paths for Each Journey:")

print(happy\_paths)