Python Summer Party Challenge

by Interview Master

Day 4 of 15

Google

You are a Product Analyst on the Google Search team investigating user engagement with search result pages. The team wants to understand how different numbers of search results impact user interaction time. Your analysis will help optimize the current search results presentation strategy.

Challenge Questions

Q1:

Identify and remove any duplicate entries in the dataset to ensure data quality. How many duplicates were found and removed?

Q2:

After dropping duplicates, aggregate the data to find the average user interaction time for each number of search results displayed per page. What are the average interaction times?

Q3:

Sort the aggregated results from Q2 to determine which number of search results per page has the highest average user interaction time. What is the optimal number of search results per page?

Want to try this yourself?

Join the Challenge

Sign up for the Python Summer Party Challenge and solve 21 days of data science problems

www.interviewmaster.ai/python-party

Or keep scrolling to see my solutions

My Solution - Q1

Day 4 Python Challenge

```
# Remove duplicates and get total removed
no_duplicates = user_engagement_data.drop_duplicates()
print(len(user_engagement_data) - len(no_duplicates))
```

InterviewMaster.AI

My Solution - Q2

Day 4 Python Challenge

```
# Remove duplicates
no_duplicates = user_engagement_data.drop_duplicates()

# Calculate the average user interaction time
avg_interaction = (
  no_duplicates.groupby('search_results_displayed')
  .agg(avg_interaction_time=('interaction_time','mean'))
  .reset_index()
)
print(avg_interaction)
```

InterviewMaster.AI

My Solution - Q3

Day 4 Python Challenge

```
# Remove duplicates
no_duplicates = user_engagement_data.drop_duplicates()

# Calculate the average user interaction time
avg_interaction = (
  no_duplicates.groupby('search_results_displayed')
  .agg(avg_interaction_time=('interaction_time','mean'))
  .reset_index()
)

# Sort the result to get the highest average user interaction
time
sorted_avg_interaction = avg_interaction.sort_values(by='avg_i
nteraction_time', ascending=False)
print(sorted_avg_interaction)
```

Ready for your own challenge?

Try this yourself by signing up for the Python Summer Party challenge:

www.interviewmaster.ai/python-party

InterviewMaster.AI