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?

InterviewMaster.AI

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

```
# Note: pandas and numpy are already imported as pd and np
# The following tables are loaded as pandas DataFrames with th
e same names: user_engagement_data
# Please print your final result or dataframe

first_count = user_engagement_data.shape[0]
duplicates = user_engagement_data.duplicated()
num_of_duplicates = duplicates.sum()
df = user_engagement_data.drop_duplicates()
new_count = df.shape[0]

print(f'Duplicates found and removed: {num_of_duplicates}')
```

InterviewMaster.AI

My Solution - Q2

Day 4 Python Challenge

```
df = user_engagement_data.drop_duplicates()

avg_interaction = df.groupby('search_results_displayed')['interaction_time'].mean().reset_index()

print(avg_interaction)
```

InterviewMaster.AI

My Solution - Q3

Day 4 Python Challenge

```python
df = user_engagement_data.drop_duplicates()

avg_interaction = df.groupby('search_results_displayed')['interaction_time'].mean().reset_index()

sorted_df = avg_interaction.sort_values(by= 'interaction_time', ascending= False)

print(sorted_df.iloc[0])
```