Python Summer Party Challenge

by Interview Master

Day 12 of 15

Walmart

You are a Data Analyst on the Walmart.com Insights team investigating customer return patterns. The team aims to develop a predictive approach to understanding customer return behaviors across different time periods. Your goal is to leverage transaction data to create a comprehensive view of customer return likelihood.

Challenge Questions

Q1:

Identify and list all unique customer IDs who have made returns between July 1st 2024 and June 30th 2025. This will help us understand the base set of customers involved in returns during the specified period.

Q2:

Convert the 'order_date' column to a datetime format and create a MultiIndex with 'customer_id' and 'order_date'. Then, calculate the total number of returns per customer for each month. This will provide insights into monthly return patterns for each customer.

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 12 Python Challenge

```
# Filtering the data to get returned orders only
returns = customer_returns[customer_returns['return_flag'] ==
True]

# Getting all returns between July 1st 2024 and June 30th 2025
returns['order_date'] = pd.to_datetime(returns['order_date'])
jul_jun_returns = returns[
  (returns['order_date'] >= '2024-07-01') &
  (returns['order_date'] <= '2025-06-30')
]

# Identifying unique customers
unique_customers = jul_jun_returns['customer_id'].unique()

print(f"List of unique customer IDs:\n{unique_customers}")
```

InterviewMaster.AI

My Solution - Q2

Day 12 Python Challenge

```
# Converting the order date to a datetime format
customer_returns['order_date'] = pd.to_datetime(
  customer_returns['order_date'],
  format='mixed',
  errors='coerce'
)

# Creating a MultiIndex with customer_id and order_date
multi_indexed = customer_returns.set_index(['customer_id', 'or
der_date'])

# Filtering for returns data only
returns_only = multi_indexed[multi_indexed['return_flag'] == T
rue]

# Calculating the total returns by customer and month
returns_only['month'] = returns_only.index.get_level_values('o
rder_date').month_name()

customer_return_agg = (
  returns_only.groupby(['customer_id','month'])
  .agg(total_returns=('order_id','count'))
  .reset_index()
)

print(customer_return_agg)
```

Ready for your own challenge?

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

www.interviewmaster.ai/python-party

InterviewMaster.AI