interview-questions

February 14, 2024

#Interview Questions

##General Questions

1. 2 more cow.
2. 4036
3. A
4. 12 meter
5. First Adam and Dave cross (11 min) then Adam turns back (1 min) Second Adam and Clair cross then Adam turns back (1 min) And last Adam and Bob cross (2 min) That's why Adam gets very tired but feels happy to help his friends.

##MySQL Questions:

[ ]:
```
6.        SELECT Source, COUNT(*) TotalTransactions
    FROM transactions
    GROUP BY Source:
```

[ ]:
```
7.         SLECT AVG(Amaount) AS AverageAmount
     FROM transactions
     WHERE MONTH(Timestamp) = 1;
```

[ ]:
```
8.  UPDATE transactions AS t1
    JOIN transactions AS t2
    ON t1.Transactionid = t2.Transactionid
    SET t1.Matchingstatus = 1
    WHERE t1.Source = 'accounting_system' AND t2.Source =␣
 ↪'payment_service_provider';
```

Question

[1]:
```
import pandas as pd
import numpy as np

# Define the data
data = {
    'source': ['web_app', 'web_app', 'web_app', 'web_app', 'web_app',
               'web_app', 'web_app', 'web_app', 'web_app','web_app',
```

```
                'web_app', 'web_app',
                'accounting_system', 'accounting_system', 'accounting_system',␣
 ↪'accounting_system', 'accounting_system',
                'accounting_system', 'accounting_system','accounting_system',␣
 ↪'accounting_system', 'accounting_system',
                'accounting_system', 'accounting_system', 'accounting_system',␣
 ↪'accounting_system',
                'accounting_system', 'payment_service_provider',␣
 ↪'payment_service_provider', 'payment_service_provider',␣
 ↪'payment_service_provider',
                'payment_service_provider', 'payment_service_provider',␣
 ↪'payment_service_provider', 'payment_service_provider',␣
 ↪'payment_service_provider',
                'payment_service_provider', 'payment_service_provider',␣
 ↪'payment_service_provider', 'payment_service_provider'],
    'transactionid': [12345, 12346, 12347, 12348, 12349, 12350, 12353, 12354,␣
 ↪12355, 12356,
                      12357, 12358,
                      12347, 12345, 12352, 12353, 12354, 12351, 12355, 12355,␣
 ↪12356, 12357,
                      12357, 12358, 12350, 12349, 12346,
                      12347, 12348, 12345, 12353, 12358, 12356, 12349, 12357,␣
 ↪12352,
                      12355, 12357, 12350, 12351],
    'amount': [2050, 2100, 2150, 2200, 2250, 2300, 2450, 2500, 2550, 2600,␣
 ↪2650, 2700,
               2150, 2050, 2400, 2450, 2500, 2350, 2550, -2550, 2600, 2650,␣
 ↪-2650, 2700,
               2300, 2250, 2100, 2150, 2200, 2050, 2500, 2700, 2600, 2700,␣
 ↪2650, 2400,
               2550, -2650, 2300, 2350]
}

# Create a DataFrame
df = pd.DataFrame(data)
```

[2]:
```
# Add a new column 'total_amount' by grouping 'source' and 'transactionid' and␣
 ↪summing 'amount'
df['total_amount'] = df.groupby(['source', 'transactionid'])['amount'].
 ↪transform('sum')

# Add a new column 'count' by grouping 'source' and 'transactionid' and␣
 ↪counting occurrences
df['count'] = df.groupby(['source', 'transactionid'])['amount'].
 ↪transform('count')
```

```
# Drop duplicates based on 'source' and 'transactionid'
df = df.drop_duplicates(subset=['source', 'transactionid'])
```

##Comparison web_app – accounting_system

[3]:
```
# Filter for the specified sources
web_app_df = df[df['source'] == 'web_app']
accounting_system_df = df[df['source'] == 'accounting_system']
payment_service_provider_df = df[df['source'] == 'payment_service_provider']
```

[4]:
```
# Merge web_app_df with accounting_system_df using left join
merged_df = pd.merge(web_app_df, accounting_system_df, on='transactionid',␣
 ↪suffixes=('_web_app', '_accounting_system'), how='left')

# Create 'Match_web_app' column
merged_df['Match_web_app'] = (merged_df['amount_web_app'] ==␣
 ↪merged_df['amount_accounting_system']) & (merged_df['total_amount_web_app']␣
 ↪== merged_df['total_amount_accounting_system'])

# Convert boolean values to 1 and 0
merged_df['Match_web_app'] = (merged_df['Match_web_app'].astype(int)) * ␣
 ↪(merged_df['count_web_app'])
```

[5]:
```
# Create 'Unmatch_web_apps' column
# If the transactionid is similar but total_amount is not similar, set to 1,␣
 ↪else set to 0
merged_df['Unmatch_web_apps'] = (merged_df['transactionid'].notnull()) &␣
 ↪(~(merged_df['Match_web_app'].astype(bool))) &␣
 ↪(merged_df['total_amount_web_app'] !=␣
 ↪merged_df['total_amount_accounting_system'])
merged_df['Unmatch_web_apps'] = (merged_df['Unmatch_web_apps'].astype(int)) * ␣
 ↪(merged_df['count_web_app'])

# Select only the desired columns
result_df = merged_df[['Match_web_app', 'Unmatch_web_apps']]

# Calculate the sum of each column
sum_match_web_app = result_df['Match_web_app'].sum()
sum_unmatch_web_apps = result_df['Unmatch_web_apps'].sum()
```

[6]:
```
# Create a DataFrame to display the sums
sum_df = pd.DataFrame({'Match_web_app': [sum_match_web_app],
                       'Unmatch_web_apps': [sum_unmatch_web_apps]})
```

[7]:
```
# Merge accounting_system_df with web_app using left join
merged_df_1 = pd.merge(accounting_system_df, web_app_df, on='transactionid',␣
 ↪suffixes=('_accounting_system', '_web_app'), how='left')
```

```
# Create 'Match_accounting_system' column
merged_df_1['Match_accounting_system'] =␣
 ↪(merged_df_1['amount_accounting_system'] == merged_df_1['amount_web_app']) &␣
 ↪(merged_df_1['total_amount_accounting_system'] ==␣
 ↪merged_df_1['total_amount_web_app'])

# Convert boolean values to 1 and 0
merged_df_1['Match_accounting_system'] =␣
 ↪(merged_df_1['Match_accounting_system'].astype(int))*␣
 ↪(merged_df_1['count_accounting_system'])

# Create 'Unmatch_accounting_system' column
# If the transactionid is similar but total_amount is not similar, set to 1,␣
 ↪else set to 0
merged_df_1['Unmatch_accounting_system'] = (merged_df_1['transactionid'].
 ↪notnull()) & (~(merged_df_1['Match_accounting_system'].astype(bool))) &␣
 ↪(merged_df_1['total_amount_web_app'] !=␣
 ↪merged_df_1['total_amount_accounting_system'])
merged_df_1['Unmatch_accounting_system'] =␣
 ↪(merged_df_1['Unmatch_accounting_system'].astype(int)) *␣
 ↪(merged_df_1['count_accounting_system'])
```

[8]:
```
# Select only the desired columns
result_df_1 = merged_df_1[['Match_accounting_system',␣
 ↪'Unmatch_accounting_system']]


# Calculate the sum of each column
sum_match_accounting_system = result_df_1['Match_accounting_system'].sum()
sum_unmatch_accounting_system = result_df_1['Unmatch_accounting_system'].sum()

# Create a DataFrame to display the sums
sum_df_1 = pd.DataFrame({'Match_accounting_system': [sum_match_web_app],
                         'Unmatch_accounting_system':␣
 ↪[sum_unmatch_accounting_system]})
```

[9]:
```
result = pd.DataFrame({
            'Source': ['web_app', 'accounting_system'],
            'Match' : [sum_df.iloc[0:1, 0][0], sum_df_1.iloc[0:1, 0][0]],
            'Unmatch' :[sum_df.iloc[0:1, 1][0], sum_df_1.iloc[0:1, 1][0]]
})


result
```

[9]:

	Source	Match	Unmatch
0	web_app	9	3
1	accounting_system	9	6

##Comparison accounting_system – payment_service_provider

[10]:
```
# Merge accounting_system_df with payment_service_provider using left join
merged_df_2 = pd.merge(accounting_system_df, payment_service_provider_df,␣
 ↪on='transactionid', suffixes=('_accounting_system',␣
 ↪'_payment_service_provider'), how='left')

# Create 'Match_accounting_syste' column
merged_df_2['Match_accounting_system'] =␣
 ↪(merged_df_2['amount_accounting_system'] ==␣
 ↪merged_df_2['amount_payment_service_provider']) &␣
 ↪(merged_df_2['total_amount_accounting_system'] ==␣
 ↪merged_df_2['total_amount_payment_service_provider'])

# Convert boolean values to 1 and 0
merged_df_2['Match_accounting_system'] =␣
 ↪(merged_df_2['Match_accounting_system'].astype(int)) * ␣
 ↪(merged_df_2['count_accounting_system'])

# Create 'Unmatch_accounting_syste' column
# If the transactionid is similar but total_amount is not similar, set to 1,␣
 ↪else set to 0
merged_df_2['Unmatch_accounting_system'] = (merged_df_2['transactionid'].
 ↪notnull()) & (~(merged_df_2['Match_accounting_system'].astype(bool))) &␣
 ↪(merged_df_2['total_amount_accounting_system'] !=␣
 ↪merged_df_2['total_amount_payment_service_provider'])
merged_df_2['Unmatch_accounting_system'] =␣
 ↪(merged_df_2['Unmatch_accounting_system'].astype(int)) * ␣
 ↪(merged_df_2['count_accounting_system'])
```

[11]:
```
# Select only the desired columns
result_df_2 = merged_df_2[['Match_accounting_system',␣
 ↪'Unmatch_accounting_system']]

# Calculate the sum of each column
sum_match_accounting_system = result_df_2['Match_accounting_system'].sum()
sum_unmatch_accounting_system = result_df_2['Unmatch_accounting_system'].sum()

# Create a DataFrame to display the sums
sum_df_2 = pd.DataFrame({'Match_accounting_system':␣
 ↪[sum_match_accounting_system],
                         'Unmatch_accounting_system':␣
 ↪[sum_unmatch_accounting_system]})
```

[12]:
```
# Merge payment_service_provider_df with accounting_system_df using left join
merged_df_3 = pd.merge(payment_service_provider_df, accounting_system_df,␣
 ↪on='transactionid',␣
 ↪suffixes=('_payment_service_provider','_accounting_system'), how='left')

# Create 'Match_payment_service_provider' column
merged_df_3['Match_payment_service_provider'] =␣
 ↪(merged_df_3['amount_payment_service_provider'] ==␣
 ↪merged_df_3['amount_accounting_system']) &␣
 ↪(merged_df_3['total_amount_payment_service_provider'] ==␣
 ↪merged_df_3['total_amount_accounting_system'])

# Convert boolean values to 1 and 0
merged_df_3['Match_payment_service_provider'] =␣
 ↪(merged_df_3['Match_payment_service_provider'].astype(int))*␣
 ↪(merged_df_3['count_payment_service_provider'])

# Create 'Unmatch_payment_service_provider' column
# If the transactionid is similar but total_amount is not similar, set to 1,␣
 ↪else set to 0
merged_df_3['Unmatch_payment_service_provider'] = (merged_df_3['transactionid'].
 ↪notnull()) & (~(merged_df_3['Match_payment_service_provider'].astype(bool)))␣
 ↪& (merged_df_3['total_amount_accounting_system'] !=␣
 ↪merged_df_3['total_amount_payment_service_provider'])
merged_df_3['Unmatch_payment_service_provider'] =␣
 ↪(merged_df_3['Unmatch_payment_service_provider'].astype(int)) *␣
 ↪(merged_df_3['count_payment_service_provider'])
```

[13]:
```
# Select only the desired columns
result_df_3 = merged_df_3[['Match_payment_service_provider',␣
 ↪'Unmatch_payment_service_provider']]


# Calculate the sum of each column
sum_match_payment_service_provider =␣
 ↪result_df_3['Match_payment_service_provider'].sum()
sum_unmatch_payment_service_provider =␣
 ↪result_df_3['Unmatch_payment_service_provider'].sum()

# Create a DataFrame to display the sums
sum_df_3 = pd.DataFrame({'Match_payment_service_provider':␣
 ↪[sum_match_payment_service_provider],
                         'Unmatch_payment_service_provider':␣
 ↪[sum_unmatch_payment_service_provider]})
```

[14]:
```
result_1 = pd.DataFrame({
            'Source': ['accounting_system', 'payment_service_provider'],
            'Match' : [sum_df_2.iloc[0:1, 0][0], sum_df_3.iloc[0:1, 0][0]],
            'Unmatch' :[sum_df_2.iloc[0:1, 1][0], sum_df_3.iloc[0:1, 1][0]]
})


result_1
```

[14]:

	Source	Match	Unmatch
0	accounting_system	9	6
1	payment_service_provider	9	4