chatterboc

- chat1
- chat1

chat1

```
import polars as pl

chunk_size = 1000  # Adjust based on available memory
dataframes = []

for i in range(0, len(a_list), chunk_size):
    chunk = a_list[i:i + chunk_size]
    dataframes.append(pl.DataFrame(chunk, schema=["stsdf", "abc"]))

combined_df = pl.concat(dataframes)
```

chat1

```
import polars as pl

def flag_offline_periods(df):
    # Count entries per bucket
    counts = df.groupby('timestamp_bucket').agg(pl.count('*').alias('count'))

    # Identify buckets with more than 20 entries
    high_count_buckets = counts.filter(pl.col('count') > 20)['timestamp_bucket']

    # Create a function to generate offline periods
    def generate_offline_periods(high_count_buckets):
        offline_periods = []
        for bucket in high_count_buckets:
            offline_end = bucket + pl.duration(seconds=30)
            offline_periods.append((bucket, offline_end))
        return offline_periods

    # Generate offline periods
    offline_periods = generate_offline_periods(high_count_buckets)

    # Merge overlapping periods
    merged_periods = []
    for start, end in sorted(offline_periods):
        if merged_periods and start <= merged_periods[-1][1]:
            merged_periods[-1] = (merged_periods[-1][0], max(merged_periods[-1][1], end))
        else:
            merged_periods.append((start, end))

    # Create a function to check if a timestamp is within offline periods
    def is_offline(timestamp):
        return any(start <= timestamp < end for start, end in merged_periods)

    # Apply the offline check to each row
    df = df.with_columns(
        pl.when(pl.col('timestamp_bucket').apply(is_offline))
        .then(pl.lit('OFFLINE'))
        .otherwise(pl.lit('ONLINE'))
        .alias('status')
    )
```

```
    return df

# Assuming your DataFrame is called 'df'
df = flag_offline_periods(df)
```