```
import json
import base64
import boto3
from botocore.exceptions import ClientError
import os

# Initialize DynamoDB client
dynamodb = boto3.resource('dynamodb')
table_name = os.environ.get('DYNAMODB_TABLE_NAME',
'SimpleLearnProject2-Adult-Data-DynamoDB')
table = dynamodb.Table(table_name)

def lambda_handler(event, context):
    print("Received event: " + json.dumps(event, indent=2))

    if 'Records' not in event:
        print("Event does not contain 'Records' key.")
        return {
            'statusCode': 500,
            'body': json.dumps("Event does not contain 'Records' key.")
        }

    for record in event['Records']:
        base64_encoded_data = record['kinesis']['data']
        try:
            # Decode the base64 data from Kinesis (first layer)
            decoded_data = base64.b64decode(base64_encoded_data).decode('utf-8').strip()

            if not decoded_data:
                print("Decoded data is empty, skipping record.")
                continue

            print(f"First decoded payload: '{decoded_data}'")

            # Since the payload is still base64-encoded JSON, decode it again (second layer)
            try:
                second_decoded_data = base64.b64decode(decoded_data).decode('utf-8').strip()
                print(f"Second decoded payload: '{second_decoded_data}'")

                # Parse the JSON data
                try:
                    data = json.loads(second_decoded_data)
                    print(f"Parsed JSON data: {data}")

                    if isinstance(data, list) and len(data) == 15:
                        print(f"Parsed data: {data}")

                        # Extract and trim values from the array
                        try:
                            age = int(data[0].strip())
                            workclass = data[1].strip()
                            fnlwgt = int(data[2].strip())
                            education = data[3].strip()
                            education_num = int(data[4].strip())
                            marital_status = data[5].strip()
```

```
                    occupation = data[6].strip()
                    relationship = data[7].strip()
                    race = data[8].strip()
                    sex = data[9].strip()
                    capital_gain = int(data[10].strip())
                    capital_loss = int(data[11].strip())
                    hours_per_week = int(data[12].strip())
                    native_country = data[13].strip()
                    income = data[14].strip()

                    # Generate a unique record ID (customize if needed)
                    record_id = f"{age}_{fnlwgt}_{education.replace(' ', '')}"  # Ensure no spaces in the ID

                    # Insert into DynamoDB
                    table.put_item(
                        Item={
                            'record_id': record_id,
                            'age': age,
                            'workclass': workclass,
                            'fnlwgt': fnlwgt,
                            'education': education,
                            'education_num': education_num,
                            'marital_status': marital_status,
                            'occupation': occupation,
                            'relationship': relationship,
                            'race': race,
                            'sex': sex,
                            'capital_gain': capital_gain,
                            'capital_loss': capital_loss,
                            'hours_per_week': hours_per_week,
                            'native_country': native_country,
                            'income': income
                        }
                    )
                    print(f"Successfully put item: {record_id}")

                except ValueError as e:
                    print(f"Error converting data types: {e}")

            else:
                print("Data is not in expected format or has incorrect length.")

        except json.JSONDecodeError as e:
            print(f"Error decoding JSON data: {e}")

    except (base64.binascii.Error, UnicodeDecodeError) as e:
        print(f"Error decoding Base64 data: {e}")

except (base64.binascii.Error, UnicodeDecodeError) as e:
    print(f"Error decoding Base64 data: {e}")
except ClientError as e:
    print(f"Error putting item to DynamoDB: {e}")

return {
    'statusCode': 200,
```

```
    'body': json.dumps('Data processed successfully')
}
```