import pandas as pd

def validate\_dataframe(incoming\_df: pd.DataFrame, reference\_df: pd.DataFrame):

# Создаем датафрейм для хранения расхождений

discrepancies\_df = pd.DataFrame(columns=["row\_index", "column", "error\_description", "incoming\_value", "reference\_range"])

# 1. Проверка уникальности строк

duplicate\_rows = incoming\_df[incoming\_df.duplicated()]

if not duplicate\_rows.empty:

for idx in duplicate\_rows.index:

discrepancies\_df = discrepancies\_df.append({

"row\_index": idx,

"column": "all",

"error\_description": "Duplicate row",

"incoming\_value": incoming\_df.iloc[idx].to\_dict(),

"reference\_range": "Unique rows expected"

}, ignore\_index=True)

# 2. Проверка уникальности поля id

duplicate\_ids = incoming\_df[incoming\_df.duplicated(subset=['id'])]

if not duplicate\_ids.empty:

for idx in duplicate\_ids.index:

discrepancies\_df = discrepancies\_df.append({

"row\_index": idx,

"column": "id",

"error\_description": "Duplicate id",

"incoming\_value": incoming\_df.at[idx, 'id'],

"reference\_range": "Unique ids expected"

}, ignore\_index=True)

# 3. Проверка значений колонок на соответствие диапазону эталонного датафрейма

for column in incoming\_df.columns:

if column in reference\_df.columns:

# Проверка числовых колонок

if pd.api.types.is\_numeric\_dtype(incoming\_df[column]):

min\_value = reference\_df[column].min()

max\_value = reference\_df[column].max()

out\_of\_range = incoming\_df[(incoming\_df[column] < min\_value) | (incoming\_df[column] > max\_value)]

if not out\_of\_range.empty:

for idx in out\_of\_range.index:

discrepancies\_df = discrepancies\_df.append({

"row\_index": idx,

"column": column,

"error\_description": "Value out of range",

"incoming\_value": incoming\_df.at[idx, column],

"reference\_range": f"{min\_value} - {max\_value}"

}, ignore\_index=True)

# Проверка категориальных колонок

elif pd.api.types.is\_object\_dtype(incoming\_df[column]) or pd.api.types.is\_categorical\_dtype(incoming\_df[column]):

reference\_values = reference\_df[column].unique()

out\_of\_range = incoming\_df[~incoming\_df[column].isin(reference\_values)]

if not out\_of\_range.empty:

for idx in out\_of\_range.index:

discrepancies\_df = discrepancies\_df.append({

"row\_index": idx,

"column": column,

"error\_description": "Invalid categorical value",

"incoming\_value": incoming\_df.at[idx, column],

"reference\_range": f"Expected one of {reference\_values}"

}, ignore\_index=True)

# Проверяем, есть ли расхождения

if discrepancies\_df.empty:

print("No discrepancies found. Report not generated.")

else:

print("Discrepancies found. Displaying rows with discrepancies...")

# Извлекаем уникальные индексы строк, где найдены расхождения

unique\_row\_indices = discrepancies\_df['row\_index'].unique()

# Выводим только те строки входящего датафрейма, где есть расхождения

rows\_with\_discrepancies = incoming\_df.loc[unique\_row\_indices]

return rows\_with\_discrepancies

# Пример использования

incoming\_data = {

'id': [1, 2, 2, 4, 5],

'age': [25, 150, 30, 22, 19],

'salary': [5000, 6000, 7000, 3000, 10000],

'status': ['active', 'inactive', 'deleted', 'active', 'unknown'] # 'unknown' - некорректное значение

}

reference\_data = {

'id': [1, 2, 3, 4, 5],

'age': [18, 35, 28, 24, 21],

'salary': [4000, 5500, 6500, 2800, 9000],

'status': ['active', 'inactive'] # Эталонные значения для категориальной переменной

}

incoming\_df = pd.DataFrame(incoming\_data)

reference\_df = pd.DataFrame(reference\_data)

# Выполняем проверку

rows\_with\_discrepancies = validate\_dataframe(incoming\_df, reference\_df)

# Выводим строки с расхождениями, если они найдены

if rows\_with\_discrepancies is not None:

print(rows\_with\_discrepancies)

Вариант 2

import pandas as pd

def find\_discrepancies(incoming\_df: pd.DataFrame, reference\_df: pd.DataFrame):

# 1. Проверка на дубликаты строк

duplicate\_rows = incoming\_df[incoming\_df.duplicated()]

# 2. Проверка уникальности поля id

duplicate\_ids = incoming\_df[incoming\_df.duplicated(subset=['id'])]

# 3. Проверка значений колонок на соответствие диапазонам и категориальным значениям

out\_of\_range\_rows = pd.DataFrame()

for column in incoming\_df.columns:

if column in reference\_df.columns:

if pd.api.types.is\_numeric\_dtype(incoming\_df[column]):

min\_value = reference\_df[column].min()

max\_value = reference\_df[column].max()

out\_of\_range\_rows = pd.concat([

out\_of\_range\_rows,

incoming\_df[(incoming\_df[column] < min\_value) | (incoming\_df[column] > max\_value)]

])

elif pd.api.types.is\_object\_dtype(incoming\_df[column]) or pd.api.types.is\_categorical\_dtype(incoming\_df[column]):

reference\_values = reference\_df[column].unique()

out\_of\_range\_rows = pd.concat([

out\_of\_range\_rows,

incoming\_df[~incoming\_df[column].isin(reference\_values)]

])

# Объединяем все строки с ошибками

discrepancies\_df = pd.concat([duplicate\_rows, duplicate\_ids, out\_of\_range\_rows]).drop\_duplicates()

# Если найдены расхождения, выводим их

if discrepancies\_df.empty:

print("No discrepancies found. Report not generated.")

else:

print("Discrepancies found:")

return discrepancies\_df

# Пример использования

incoming\_data = {

'id': [1, 2, 2, 4, 5],

'age': [25, 150, 30, 22, 19],

'salary': [5000, 6000, 7000, 3000, 10000],

'status': ['active', 'inactive', 'deleted', 'active', 'unknown']

}

reference\_data = {

'id': [1, 2, 3, 4, 5],

'age': [18, 35, 28, 24, 21],

'salary': [4000, 5500, 6500, 2800, 9000],

'status': ['active', 'inactive']

}

incoming\_df = pd.DataFrame(incoming\_data)

reference\_df = pd.DataFrame(reference\_data)

# Выполняем проверку

discrepancies\_df = find\_discrepancies(incoming\_df, reference\_df)

# Выводим строки с расхождениями, если они найдены

if discrepancies\_df is not None:

print(discrepancies\_df)

Вариант 3

import pandas as pd

import great\_expectations as ge

def validate\_dataframe(incoming\_df: pd.DataFrame, reference\_df: pd.DataFrame):

# Преобразуем входящий DataFrame в объект great\_expectations

incoming\_ge = ge.from\_pandas(incoming\_df)

# 1. Проверка уникальности строки

result = incoming\_ge.expect\_table\_row\_count\_to\_be\_between(min\_value=0, max\_value=len(incoming\_df.drop\_duplicates()))

if not result.success:

print("Duplicate rows found.")

# 2. Проверка уникальности поля 'id'

result = incoming\_ge.expect\_column\_values\_to\_be\_unique("id")

if not result.success:

print("Duplicate ids found.")

# 3. Проверка значений на соответствие диапазонам числовых полей

for column in ['age', 'salary']:

min\_value = reference\_df[column].min()

max\_value = reference\_df[column].max()

result = incoming\_ge.expect\_column\_values\_to\_be\_between(column, min\_value=min\_value, max\_value=max\_value)

if not result.success:

print(f"Values in column {column} are out of range.")

# 4. Проверка категориальных значений

for column in ['status']:

reference\_values = reference\_df[column].unique()

result = incoming\_ge.expect\_column\_values\_to\_be\_in\_set(column, value\_set=reference\_values)

if not result.success:

print(f"Invalid categorical values found in column {column}.")

# Пример использования

incoming\_data = {

'id': [1, 2, 2, 4, 5],

'age': [25, 150, 30, 22, 19],

'salary': [5000, 6000, 7000, 3000, 10000],

'status': ['active', 'inactive', 'deleted', 'active', 'unknown'] # Некорректное значение

}

reference\_data = {

'id': [1, 2, 3, 4, 5],

'age': [18, 35, 28, 24, 21],

'salary': [4000, 5500, 6500, 2800, 9000],

'status': ['active', 'inactive'] # Эталонные значения для категориальных переменных

}

incoming\_df = pd.DataFrame(incoming\_data)

reference\_df = pd.DataFrame(reference\_data)

# Выполняем проверку

validate\_dataframe(incoming\_df, reference\_df)

Вариант 4 с отправкой сообщения в ТГ

from airflow import DAG

from airflow.operators.python\_operator import PythonOperator

from airflow.utils.dates import days\_ago

import pandas as pd

import great\_expectations as ge

import requests

# Telegram bot token and chat ID

TELEGRAM\_BOT\_TOKEN = 'YOUR\_TELEGRAM\_BOT\_TOKEN'

CHAT\_ID = 'YOUR\_CHAT\_ID'

# Функция для отправки сообщения в Telegram

def send\_telegram\_message(message):

url = f"https://api.telegram.org/bot{TELEGRAM\_BOT\_TOKEN}/sendMessage"

payload = {

'chat\_id': CHAT\_ID,

'text': message,

'parse\_mode': 'HTML'

}

requests.post(url, json=payload)

# Создаем DAG

default\_args = {

'owner': 'airflow',

'start\_date': days\_ago(1),

}

dag = DAG(

'data\_validation\_pipeline',

default\_args=default\_args,

description='A simple data validation pipeline with Great Expectations',

schedule\_interval=None,

)

# Пример данных

incoming\_data = {

'id': [1, 2, 2, 4, 5],

'age': [25, 150, 30, 22, 19],

'salary': [5000, 6000, 7000, 3000, 10000],

'status': ['active', 'inactive', 'deleted', 'active', 'unknown'] # Некорректное значение

}

reference\_data = {

'id': [1, 2, 3, 4, 5],

'age': [18, 35, 28, 24, 21],

'salary': [4000, 5500, 6500, 2800, 9000],

'status': ['active', 'inactive'] # Эталонные значения для категориальных переменных

}

# Функция загрузки данных

def load\_data(\*\*kwargs):

incoming\_df = pd.DataFrame(incoming\_data)

reference\_df = pd.DataFrame(reference\_data)

# Передаем данные через XCom

kwargs['ti'].xcom\_push(key='incoming\_df', value=incoming\_df)

kwargs['ti'].xcom\_push(key='reference\_df', value=reference\_df)

# Функция валидации данных с использованием great\_expectations

def validate\_data(\*\*kwargs):

# Получаем данные из XCom

incoming\_df = kwargs['ti'].xcom\_pull(key='incoming\_df')

reference\_df = kwargs['ti'].xcom\_pull(key='reference\_df')

# Преобразуем входящий DataFrame в объект great\_expectations

incoming\_ge = ge.from\_pandas(incoming\_df)

discrepancies = []

for column in incoming\_df.columns:

if column in reference\_df.columns:

# 1. Проверка уникальности

if incoming\_df[column].dtype in [int, float]:

result = incoming\_ge.expect\_column\_values\_to\_be\_unique(column)

if not result.success:

discrepancies.append(f"Duplicate values found in column '{column}'.")

# 2. Проверка диапазонов для числовых данных

min\_value = reference\_df[column].min()

max\_value = reference\_df[column].max()

result = incoming\_ge.expect\_column\_values\_to\_be\_between(column, min\_value=min\_value, max\_value=max\_value)

if not result.success:

discrepancies.append(f"Values in column '{column}' are out of range.")

else:

# 3. Проверка категориальных значений

reference\_values = reference\_df[column].unique()

result = incoming\_ge.expect\_column\_values\_to\_be\_in\_set(column, value\_set=reference\_values)

if not result.success:

discrepancies.append(f"Invalid categorical values found in column '{column}'.")

if discrepancies:

message = "Discrepancies found:\n" + "\n".join(discrepancies)

send\_telegram\_message(message)

else:

send\_telegram\_message("All data validations passed successfully.")

# Задача загрузки данных

load\_data\_task = PythonOperator(

task\_id='load\_data',

python\_callable=load\_data,

provide\_context=True,

dag=dag,

)

# Задача валидации данных

validate\_data\_task = PythonOperator(

task\_id='validate\_data',

python\_callable=validate\_data,

provide\_context=True,

dag=dag,

)

# Определяем порядок задач

load\_data\_task >> validate\_data\_task

Вариант 4

from airflow import DAG

from airflow.operators.python\_operator import PythonOperator

from airflow.utils.dates import days\_ago

import pandas as pd

import great\_expectations as ge

import requests

# Telegram bot token and chat ID

TELEGRAM\_BOT\_TOKEN = 'YOUR\_TELEGRAM\_BOT\_TOKEN'

CHAT\_ID = 'YOUR\_CHAT\_ID'

# Функция для отправки сообщения в Telegram

def send\_telegram\_message(message):

url = f"https://api.telegram.org/bot{TELEGRAM\_BOT\_TOKEN}/sendMessage"

payload = {

'chat\_id': CHAT\_ID,

'text': message,

'parse\_mode': 'HTML'

}

requests.post(url, json=payload)

# Создаем DAG

default\_args = {

'owner': 'airflow',

'start\_date': days\_ago(1),

}

dag = DAG(

'data\_validation\_pipeline',

default\_args=default\_args,

description='A simple data validation pipeline with Great Expectations',

schedule\_interval=None,

)

# Пример данных

incoming\_data = {

'id': [1, 2, 2, 4, 5],

'age': [25, 150, 30, 22, 19],

'salary': [5000, 6000, 7000, 3000, 10000],

'status': ['active', 'inactive', 'deleted', 'active', 'unknown'] # Некорректное значение

}

reference\_data = {

'id': [1, 2, 3, 4, 5],

'age': [18, 35, 28, 24, 21],

'salary': [4000, 5500, 6500, 2800, 9000],

'status': ['active', 'inactive'] # Эталонные значения для категориальных переменных

}

# Функция загрузки данных

def load\_data(\*\*kwargs):

incoming\_df = pd.DataFrame(incoming\_data)

reference\_df = pd.DataFrame(reference\_data)

# Передаем данные через XCom

kwargs['ti'].xcom\_push(key='incoming\_df', value=incoming\_df)

kwargs['ti'].xcom\_push(key='reference\_df', value=reference\_df)

# Функция валидации данных с использованием great\_expectations

def validate\_data(\*\*kwargs):

# Получаем данные из XCom

incoming\_df = kwargs['ti'].xcom\_pull(key='incoming\_df')

reference\_df = kwargs['ti'].xcom\_pull(key='reference\_df')

# Преобразуем входящий DataFrame в объект great\_expectations

incoming\_ge = ge.from\_pandas(incoming\_df)

discrepancies = []

for column in incoming\_df.columns:

if column in reference\_df.columns:

# 1. Проверка уникальности

if incoming\_df[column].dtype in [int, float]:

result = incoming\_ge.expect\_column\_values\_to\_be\_unique(column)

if not result.success:

discrepancies.append(f"Duplicate values found in column '{column}'.")

# 2. Проверка диапазонов для числовых данных

min\_value = reference\_df[column].min()

max\_value = reference\_df[column].max()

result = incoming\_ge.expect\_column\_values\_to\_be\_between(column, min\_value=min\_value, max\_value=max\_value)

if not result.success:

discrepancies.append(f"Values in column '{column}' are out of range.")

else:

# 3. Проверка категориальных значений

reference\_values = reference\_df[column].unique()

result = incoming\_ge.expect\_column\_values\_to\_be\_in\_set(column, value\_set=reference\_values)

if not result.success:

discrepancies.append(f"Invalid categorical values found in column '{column}'.")

# Отправляем сообщение только если есть расхождения

if discrepancies:

message = "Discrepancies found:\n" + "\n".join(discrepancies)

send\_telegram\_message(message)

else:

print("All data validations passed successfully.")

# Задача загрузки данных

load\_data\_task = PythonOperator(

task\_id='load\_data',

python\_callable=load\_data,

provide\_context=True,

dag=dag,

)

# Задача валидации данных

validate\_data\_task = PythonOperator(

task\_id='validate\_data',

python\_callable=validate\_data,

provide\_context=True,

dag=dag,

)

# Определяем порядок задач

load\_data\_task >> validate\_data\_task