SurveyMoneky API

January 29, 2025

1 Survey Monkey ETL Pipline

[ ]:
```
import pandas as pd
import requests
```

1.0.1 Establish Link Between Python and Survey Monkey

[ ]:
```
import requests

# Provide Access Token and Survey Monkey URL
ACCESS_TOKEN = "Access_Token"
BASE_URL = "https://api.surveymonkey.com/v3"

headers = {
    "Authorization": f"Bearer {ACCESS_TOKEN}",
    "Content-Type": "application/json"
}

# Fetch all surveys in your account
response = requests.get(f"{BASE_URL}/surveys", headers=headers)

if response.status_code == 200:
    print(response.json())  # List of surveys
else:
    print(f"Error: {response.status_code} - {response.text}")
```

1.1 Fetch Responses for the Survey

1.1.1 Pick a survey ID

[ ]:
```
# Use access token
ACCESS_TOKEN = "Access_Token"
BASE_URL = "https://api.surveymonkey.com/v3"
SURVEY_ID = "313731387"  # Illness Survey ID

headers = {
    "Authorization": f"Bearer {ACCESS_TOKEN}",
    "Content-Type": "application/json"
```

```
}

# Get responses
def get_survey_responses(survey_id, page=1, per_page=100):
    """Fetch responses for a given survey ID"""
    endpoint = f"{BASE_URL}/surveys/{survey_id}/responses/bulk?
 ↪page={page}&per_page={per_page}"
    response = requests.get(endpoint, headers=headers)

    if response.status_code == 200:
        return response.json()
    else:
        print(f"Error {response.status_code}: {response.text}")
        return None

responses = get_survey_responses(SURVEY_ID)
if responses:
    print(responses)
```

1.1.2 Match Survey details to Question ID's

[ ]:
```
def get_survey_details(survey_id):
    """Fetch survey questions and structure"""
    endpoint = f"{BASE_URL}/surveys/{survey_id}/details"
    response = requests.get(endpoint, headers=headers)

    if response.status_code == 200:
        return response.json()
    else:
        print(f"Error {response.status_code}: {response.text}")
        return None

details = get_survey_details(SURVEY_ID)
if details:
    print(details)  # View the question structure
```

1.1.3 Extract Responses for the Survey

[ ]:
```
def get_survey_responses(survey_id, page=1, per_page=100):
    """Fetch all responses for a given survey ID"""
    endpoint = f"{BASE_URL}/surveys/{survey_id}/responses/bulk?
 ↪page={page}&per_page={per_page}"
    response = requests.get(endpoint, headers=headers)

    if response.status_code == 200:
        return response.json()
    else:
```

```
        print(f"Error {response.status_code}: {response.text}")
        return None

responses = get_survey_responses(SURVEY_ID)
if responses:
    print(responses)  # Inspect the response structure
```

1.1.4 Extract Question and Answer Text from Survey Details

[ ]:
```
def extract_question_mapping(survey_details):
    """Create a dictionary mapping question IDs to question text and answer␣
 ↪choices"""
    question_map = {}

    for page in survey_details.get("pages", []):
        for question in page.get("questions", []):
            question_id = question["id"]
            question_text = question["headings"][0]["heading"]

            # Store question text
            question_map[question_id] = {"question_text": question_text,␣
 ↪"choices": {}}

            # If the question has multiple choice answers, store answer choices
            if "answers" in question and "choices" in question["answers"]:
                for choice in question["answers"]["choices"]:
                    choice_id = choice["id"]
                    choice_text = choice["text"]
                    question_map[question_id]["choices"][choice_id] =␣
 ↪choice_text

    return question_map

question_map = extract_question_mapping(details)
print(question_map)  # View the question mapping
```

1.1.5 Convert Responses to a Readable Table

[ ]:
```
print(question_map)
```

[ ]:
```
import requests
import pandas as pd
import json

# --- Question Map ---
question_map = {
    "722712604": {"question_text": "What is your name?"},
```

```
    "722713064": {
        "question_text": "What floor do you work on?",
        "choices": {
            "4753019026": "1st Floor",
            "4753019027": "2nd Floor",
            "4753019028": "3rd Floor",
            "4753019029": "4th Floor",
            "4753019030": "5th Floor",
            "4753019031": "Float"
        }
    },
    "722714894": {
        "question_text": "Please indicate your position or department.",
        "choices": {
            "4753032503": "MHW",
            "4753032504": "Nurse",
            "4753032505": "LSI",
            "4753032506": "TRS",
            "4753032507": "Social Worker",
            "4753032508": "MDS Coordinator",
            "4753032509": "Clinical Management",
            "4753032510": "Housekeeping",
            "4753032511": "Dietary",
            "4753032512": "Maintenance",
            "4753032513": "Vocational",
            "4753032514": "Human Resources",
            "4753032515": "Front Office",
            "4753032516": "Financial",
            "4753032517": "Other (please specify)"
        }
    },
    "722733269": {
        "question_text": "What symptoms are you experiencing?",
        "choices": {
            "4753152481": "Fever (Temperature 100.0F and above)",
            "4753152482": "Cough",
            "4753152483": "Shortness of breath",
            "4753152484": "Diarrhea",
            "4753152485": "Vomiting",
            "4753152486": "Nausea",
            "4753152487": "Nasal Congestion",
            "4813579695": "Sore throat",
            "4753152488": "Body aches",
            "4753152489": "Headache",
            "4753152490": "Fatigue",
            "4753152491": "Runny nose",
            "4753152492": "New loss of taste or smell",
```

```
            "4753152493": "None of the above",
            "4753152494": "Other (please specify)"
        }
    },
    "775672031": {"question_text": "Please indicate approximate date and time␣
 ↪of symptoms onset."},
    "775672030": {"question_text": "Have you been in close contact with anyone␣
 ↪who has exhibited the above symptoms or known to be COVID or Flu Positive?"},
    "775672028": {"question_text": "If you answered 'Yes' to the previous␣
 ↪question, Indicate the person(s)."},
    "775672034": {
        "question_text": "Have you completed at home or in clinic testing or␣
 ↪been seen by a provider?",
        "choices": {
            "5137922448": "COVID-19 Test",
            "5137922449": "Doctors appointment",
            "5137922450": "Flu Test",
            "5137922451": "Other (please specify)"
        }
    },
    "775672037": {"question_text": "If applicable, please indicate the results␣
 ↪of testing."},
    "775672035": {"question_text": "If you underwent testing, please provide␣
 ↪the date."},
    "723185890": {"question_text": "What was the last shift you worked?"},
    "723186244": {"question_text": "What is your phone number?"},
    "723186672": {
        "question_text": "Is the number provided able to receive and send texts?
 ↪",
        "choices": {
            "4756101513": "Yes",
            "4756101514": "No"
        }
    },
    "723187096": {"question_text": "If applicable, please provide any␣
 ↪additional details that have not been addressed above."}
}



# --- Fetch Survey Responses ---
def fetch_all_survey_responses(api_url, start_date="2024-01-01",␣
 ↪end_date="2025-12-31"):
    """Fetch all paginated survey responses from the API within the given date␣
 ↪range."""
```

```
    all_responses = []
    page = 1

    while True:
        params = {
            "start_created_at": start_date,
            "end_created_at": end_date,
            "page": page,
            "per_page": 1000
        }

        response = requests.get(api_url, headers=headers, params=params)

        # Handle errors
        if response.status_code != 200:
            print(f"  Error: {response.status_code} - {response.text}")
            return None

        data = response.json()
        if "data" not in data or not data["data"]:
            break  # Stop when no more data is returned

        all_responses.extend(data["data"])
        page += 1

    print(f"  Total responses fetched: {len(all_responses)}")
    return {"data": all_responses}

# --- Process Responses ---
def process_responses(responses, question_map):
    """Convert API responses into a structured DataFrame, ensuring all mapped␣
 ↪questions appear."""
    processed_data = []

    for response in responses.get("data", []):
        row = {
            "response_id": response["id"],
            "date_created": response["date_created"]
        }

        # Initialize all expected columns as empty strings
        for question_id, question_info in question_map.items():
            row[question_info["question_text"]] = ""

        # Populate responses
        for page in response.get("pages", []):
            for question in page.get("questions", []):
```

```
                question_id = question["id"]

                if question_id in question_map:
                    question_text = question_map[question_id]["question_text"]

                    if "answers" in question and question["answers"]:
                        answer_data = question["answers"][0]

                        # Handle text responses
                        if "text" in answer_data:
                            row[question_text] = answer_data["text"]

                        # Handle single-choice responses
                        elif "choice_id" in answer_data:
                            choice_id = answer_data["choice_id"]
                            row[question_text] =␣
 ↪question_map[question_id]["choices"].get(choice_id, "Unknown Choice")

                        # Handle multiple-choice responses
                        elif "choice_ids" in answer_data:
                            choice_ids = answer_data["choice_ids"]
                            row[question_text] = ", ".join(
                                question_map[question_id]["choices"].get(cid,␣
 ↪"Unknown Choice") for cid in choice_ids
                            )

        processed_data.append(row)

    return pd.DataFrame(processed_data)


# --- Main Script ---
ACCESS_TOKEN = "Access_Token"
SURVEY_ID = "313731387"  # Add Survey ID
API_URL = f"https://api.surveymonkey.com/v3/surveys/{SURVEY_ID}/responses/bulk"

headers = {
    "Authorization": f"Bearer {ACCESS_TOKEN}",
    "Content-Type": "application/json"
}

# Fetch all responses (paginated)
responses = fetch_all_survey_responses(API_URL)

if responses:
    # Process responses
    df_responses = process_responses(responses, question_map)
```

```
    # Debugging: Print the first rows of the DataFrame
    print(df_responses.head())

    # Convert 'date_created' to datetime and remove timezone
    df_responses["date_created"] = pd.to_datetime(df_responses["date_created"],␣
 ↪errors="coerce").dt.tz_localize(None)

    # Filter for 2024+ responses
    df_responses = df_responses[df_responses["date_created"] >= "2024-01-01"]

    # Save to Excel (ensuring file is not open)
    output_file = r"K:\INFECTION CONTROL\Infection␣
 ↪Reports\SurveyMonkeyIllCallData\SurveyMonkeyToExcelPipeline.xlsx"
    with pd.ExcelWriter(output_file, mode="w", engine="openpyxl") as writer:
        df_responses.to_excel(writer, index=False)

    print("  File successfully saved to:", output_file)
else:
    print("  No responses fetched. Check your API token, Survey ID, or date␣
 ↪range.")
```

[ ]:
```
import requests
import json

#  Add Access Token
ACCESS_TOKEN = "Access_Token"  # Add API token
SURVEY_ID = "313731387"  # Add Survey ID

#  API Endpoint for Fetching Survey Questions
API_URL = f"https://api.surveymonkey.com/v3/surveys/{SURVEY_ID}/details"

#  API Headers
headers = {
    "Authorization": f"Bearer {ACCESS_TOKEN}",
    "Content-Type": "application/json"
}

#  Make API Request
response = requests.get(API_URL, headers=headers)

#  Check Response
if response.status_code == 200:
    data = response.json()
    print(json.dumps(data, indent=2))  # Pretty print JSON response
else:
    print(f"  Error: {response.status_code} - {response.text}")
```