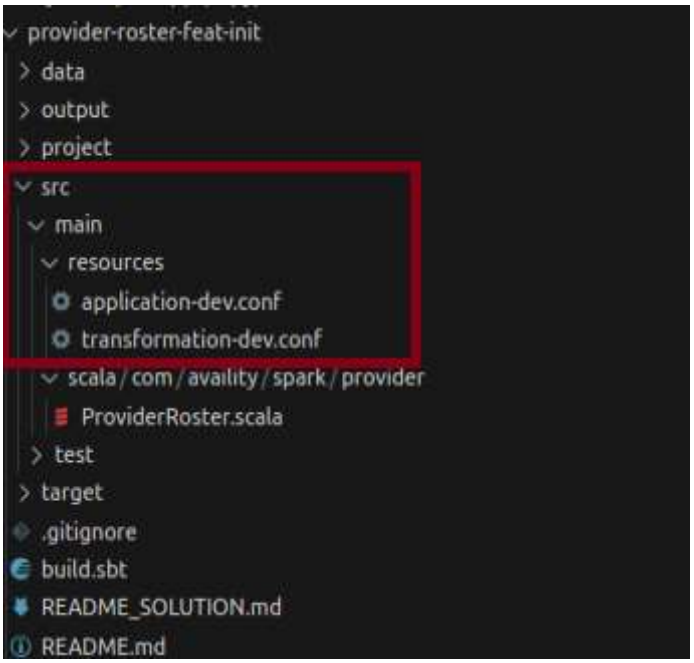
I. DEPLOYMENT

1. Environment Preparation

- Created an Ubuntu 22.04 VM using VirtualBox
- Installed java, scala, spark, code
- Downloaded the zip file from https://gitlab.com/dane.allen.3/provider-roster/-/blob/feat/init/README.md?ref_type=heads
- extracted the zip file to local folder in ubuntu as "provider-roster-feat-init"

2. Application Preparation

- Updated provider-roster-feat-init/build.sbt
- Updated provider-roster-feat-init/src/main/scala/com/availity/spark/provider/ProviderRoster.scala
    - conf file location



- provider-roster-feat-init/src/main/resources/application-dev.conf

provider-roster-feat-init > src > main > resources > application-dev.conf

```
app {
    spark_session {
        master  = "local[*]"
        appName = "ProviderAnalytics"
        timeout = "300"
        # you can add your additional config here
    }

    provider_csv {
        csv_path = "data/providers.csv"
        csv_header = "true"
        csv_delimiter = "|"
        csv_schema = [
            { name = "provider_id", type = "LongType", nullable = false },
            { name = "provider_specialty", type = "StringType", nullable = false },
            { name = "first_name", type = "StringType", nullable = false },
            { name = "middle_name", type = "StringType", nullable = false },
            { name = "last_name", type = "StringType", nullable = false }
        ]
        }

    visit_csv {
        csv_path  = "data/visits.csv"
        csv_header = "false"
        csv_delimiter = ","
        csv_schema = [
            { name = "visit_id", type = "LongType", nullable = false },
            { name = "provider_id", type = "LongType", nullable = false },
            { name = "date_of_visit", type = "DateType", nullable = false }
        ]
        }

    output_paths {
        answer_to_question_1_path = "output/answer_to_question_1"
        answer_to_question_2_path = "output/answer_to_question_2"
    }
}

```

- provider-roster-feat-init/src/main/resources/transformation-dev.conf

provider-roster-feat-init > src > main > resources > transformation-dev.conf

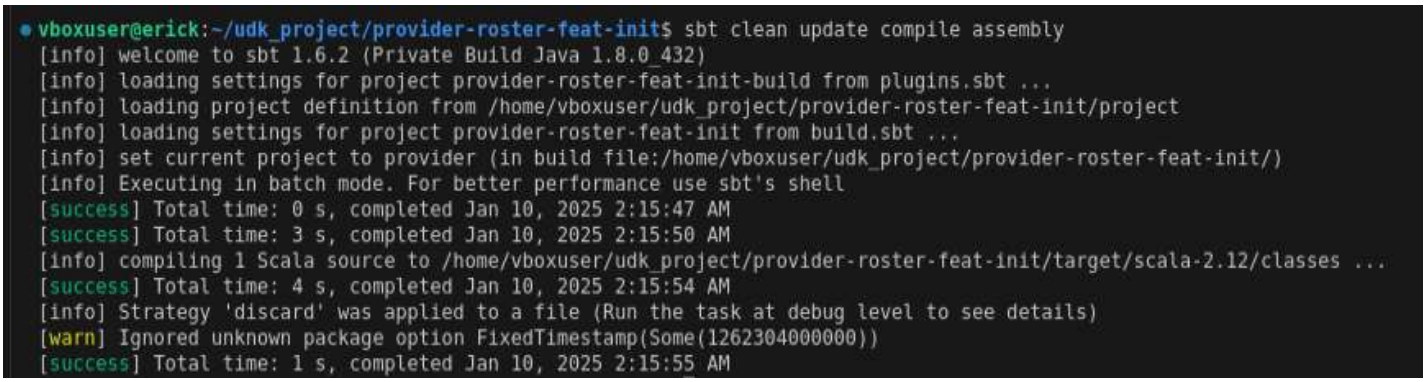
```
transformations {
  question_1_cte = """
    -- CTE 1: Create `visited_provider_table` by joining provider_table and visit_table
    WITH visited_provider_table AS (
      SELECT
        p.provider_id,
        p.provider_specialty,
        CONCAT(p.first_name, ' ', p.middle_name, ' ', p.last_name) AS full_name,
        DATE_FORMAT(v.date_of_visit, 'yyyy_MM') AS year_month,
        v.visit_id
      FROM provider_table p
      LEFT JOIN visit_table v
      ON p.provider_id = v.provider_id
    ),

    -- CTE 2: Aggregate visit counts for each provider and specialty
    visited_provider_aggregated AS (
      SELECT
        provider_specialty,
        provider_id,
        full_name,
        COUNT(provider_id) AS count_of_visit
      FROM visited_provider_table
      GROUP BY provider_specialty, provider_id, full_name
      ORDER BY provider_specialty, provider_id, full_name
    ),

```

3. Application Logics

- The application is consisted of 7 steps
- Step 1: Read application and transformation configs
- Step 2: Initialize SparkSession
- Step 3: Read input files (providers.csv and visits.csv) to DataFrames
- Step 4: Register the DataFrames as tables
- Step 5: Extract the sql transformation scripts from transformation.conf file
- Step 6: create dataframes from the query strings representing the result (solution)
- Step 7: write the dataframes as json files

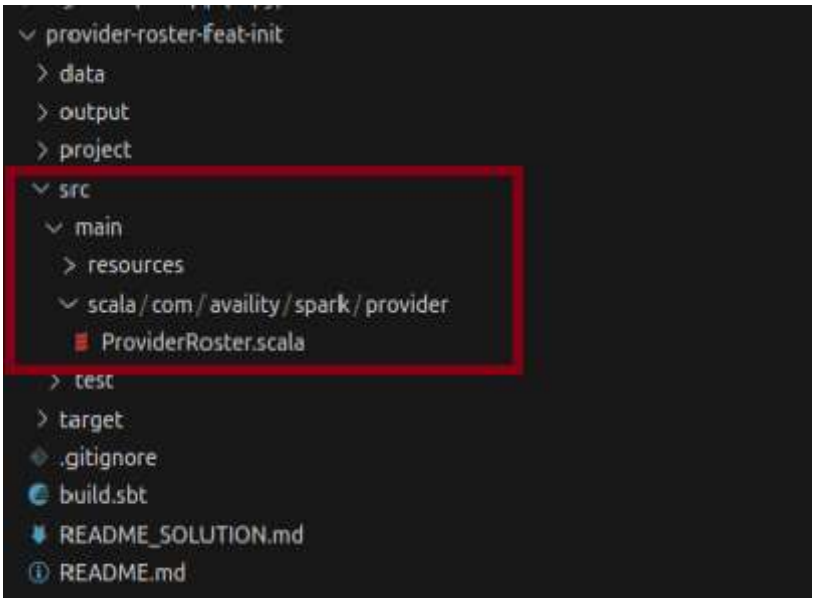
4. Compile + Create JAR

- Open vs code in ubuntu
- Open the folder provider-roster-feat-init
- Open the vs code integrated terminal
- Run the following from the command line "sbt clean update compile assembly"



5. JSON Output

- Once the uber jar is created in the target/scala-2.12/provider.jar, then run the following command line
- "spark-submit --class com.availity.spark.provider.ProviderRoster target/scala-2.12/provider.jar"



- The above step will produce 2 output json folders representing the output files for the 2 questions
  - Question 1:

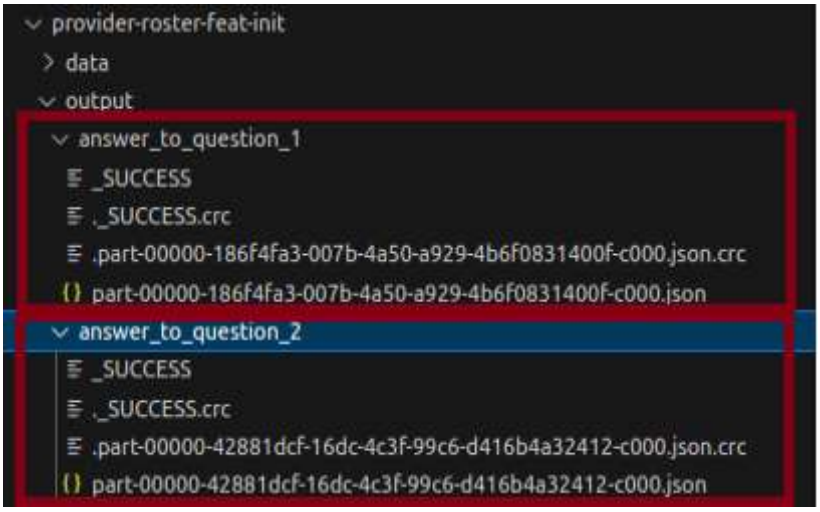
    Given the two data datasets, calculate the total number of visits per provider. The resulting set should contain the provider's ID, name, specialty, along with the number of visits. Output the report in json, partitioned by the provider's specialty.

- Question 2:

  Given the two datasets, calculate the total number of visits per provider per month. The resulting set should contain the provider's ID, the month, and total number of visits. Output the result set in json.


```
{
  "provider_id": {
    "57": [],
    "145": [],
    "204": [],
    "402": [],
    "410": [],
    "467": [],
    "499": [],
    "505": [],
    "509": [],
    "548": [],
    "574": [],
    "642": [],
    "667": [],
    "669": [
      {
        "year_month": "2021_09",
        "count_of_visit": 3
      },
      {
        "year_month": "2021_10",
        "count_of_visit": 2
      },
      {
        "year_month": "2021_12",
        "count_of_visit": 1
      },
      {
        "year_month": "2022_01",
        "count_of_visit": 2
```


- Here are the location of the files


provider-roster-feat-init
data
output
answer_to_question_1
_SUCCESS
._SUCCESS.crc
.part-00000-186f4fa3-007b-4a50-a929-4b6f0831400f-c000.json.crc
part-00000-186f4fa3-007b-4a50-a929-4b6f0831400f-c000.json
answer_to_question_2
_SUCCESS
._SUCCESS.crc
.part-00000-42881dcf-16dc-4c3f-99c6-d416b4a32412-c000.json.crc
part-00000-42881dcf-16dc-4c3f-99c6-d416b4a32412-c000.json

II. ASSUMPTIONS

1. Implemented 2 conf files, i.e.,
    - src/main/resources/application-dev.conf
        - defines the application details pertaining to
        - a. spark_session (contains the info about the spark session)
        - b. provider.csv (source location, schema, delimiter, etc)
        - c. visit.csv (source location, schema, delimiter, etc)
        - d. output_paths (contains the location of the output json files)
    - src/main/resources/transformation-dev.conf
        - defines the transformation logic to join the provider with the visit data
        - uses CTEs to handle the joining, aggregation and transformation logics to match the required JSON structure
2. Added "com.typesafe" % "config" % "1.4.2" on build.sbt to simplify the parsing of conf files
3. Implemented logging, i.e. import org.slf4j.LoggerFactory
4. Imeplemented try, catch, finally
5. Method testing not implemented

III. DATA OBSERVATION

1. In the provider.csv files,

- there are a total of 1000 rows, with 0 null values
- unique provider_id is 992
- 8 provider_id have 2 provider_specialty. See below:
  - 86440
  - 86202
  - 82338
  - 99487
  - 55504
  - 56273
  - 63596
  - 25817

2. In the visit.csv,

- there are a total of 22,348 rows, with 0 null values
- no duplicate on visit_id
- there is provider_id but provider_specialty is not available,

3. Important Notice

- NO ADJUSTMENT WAS MADE TO CORRECT "THE IMPACT OF PROVIDER_ID WITH 2 PROVIDER_SPECIALTY" SINCE THERE IS NO SPECIFIC INSTRUCTION OR DESCRIPTION ABOUT IT.
- The impact will be double counting the visits for those 8 provider_id, which can be translated as a VISIT on a provider_id will mean visiting both PROVIDER_SPECIALTY which can skew the total number of visits when re-aggregated by provider_specialty by provider_id.

IV. DOCUMENTATION

- See the provider-roster-feat-init/README_SOLUTION.md.
- It describes the entire solution
- See the 7 images on provider-roster-feat-init/images