**Oleksii Saiun**

Contacts:

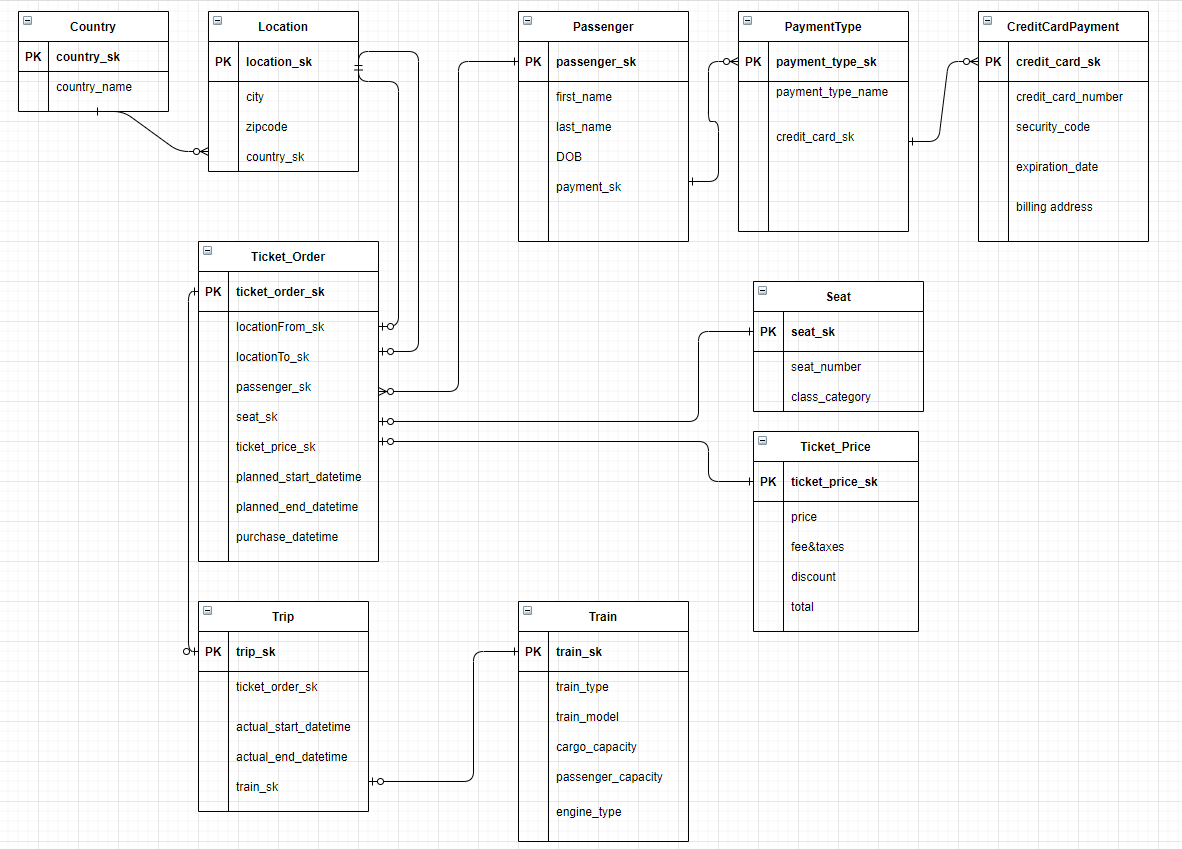
Linkedin: <https://www.linkedin.com/in/oleksii-saiun-37a20b1b0/>

Email: oleksiisaiun@gmail.com;

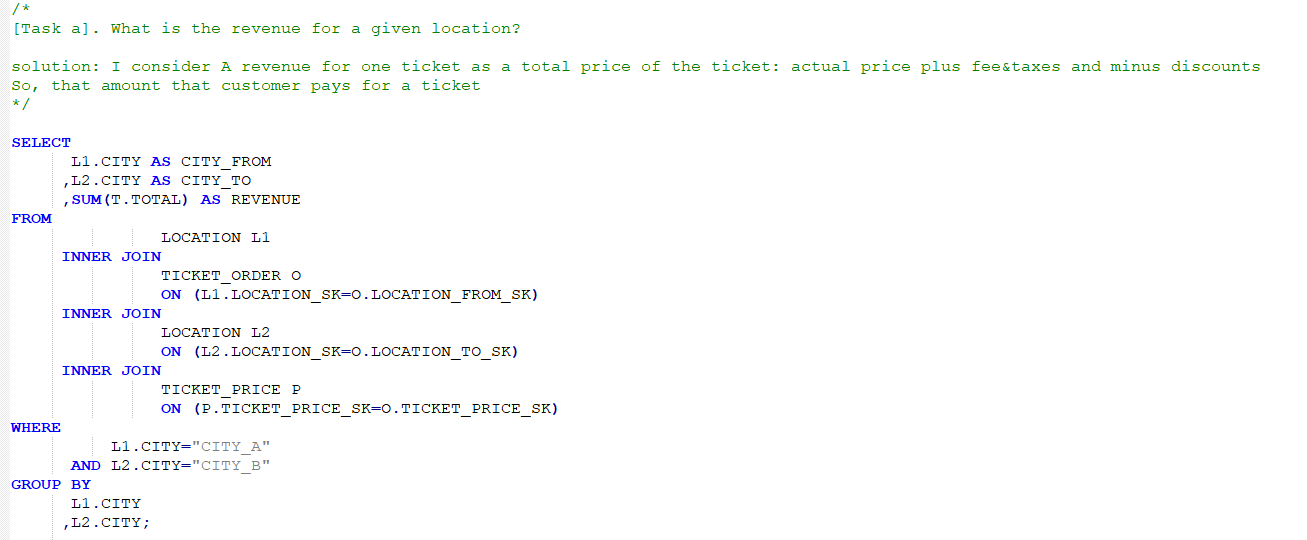
**Link to code**:

<https://github.com/oleksiisaiun/JAVA/tree/master/CODE/INTERVIEW_HOMETASK/TESLA>

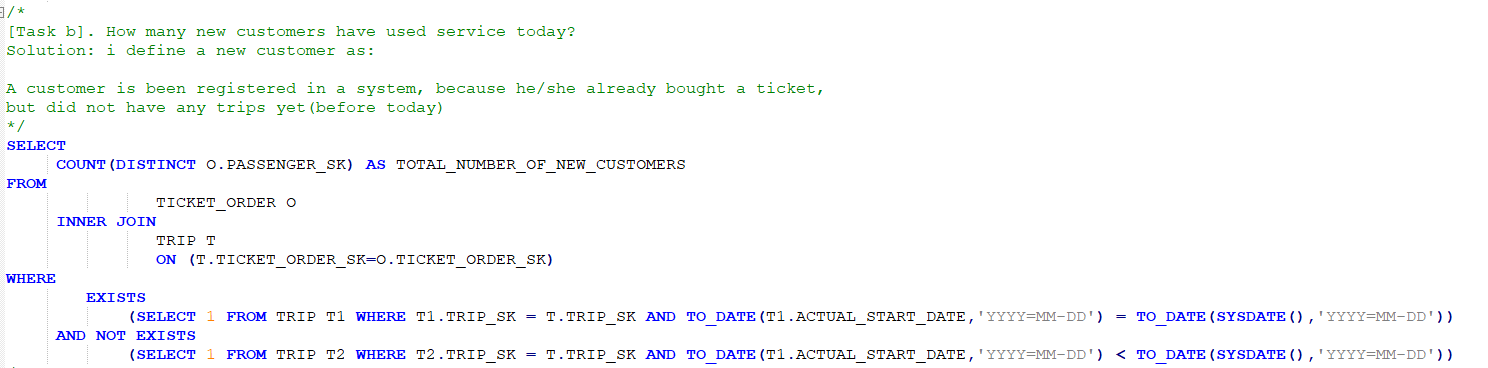
# **TASK1**



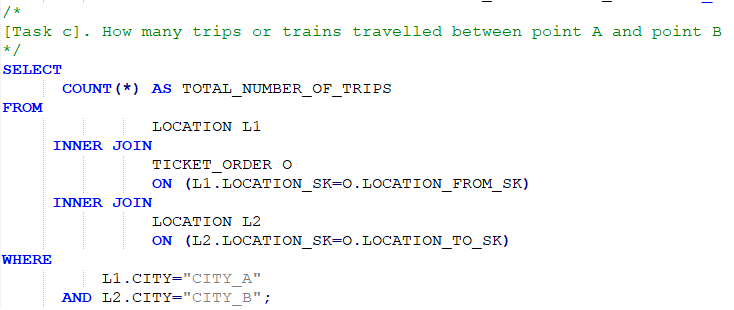
[Task a]. What is the revenue for a given location?



[Task b]. How many new customers have used service today?



[Task c]. How many trips or trains travelled between point A and point B



# **TASK2**

**TASK 2.1-2.3.** *query all events that have occurred during year 2017 and load into database.*

Design’s overview of my solution.

I chose Microsoft Azure Cloud platform to bring data from the url, parse and store data. I use Azure Datarbicks as a processing tool and RDBMS [Azure SQL] (it is a cloud version of SQL Server) to store data

Implementation’s steps:

1. Generate files of json format in Postman and save them on local machine
2. Put them in MS Azure Cloud storage and store them as BLOB files
3. Mount folder in Azure DataBricks for loaded files
4. Create DataFrames using Pyspark in Azure DataBricks
5. In a relational database [Azure SQL], create two schemas: staging and dwh

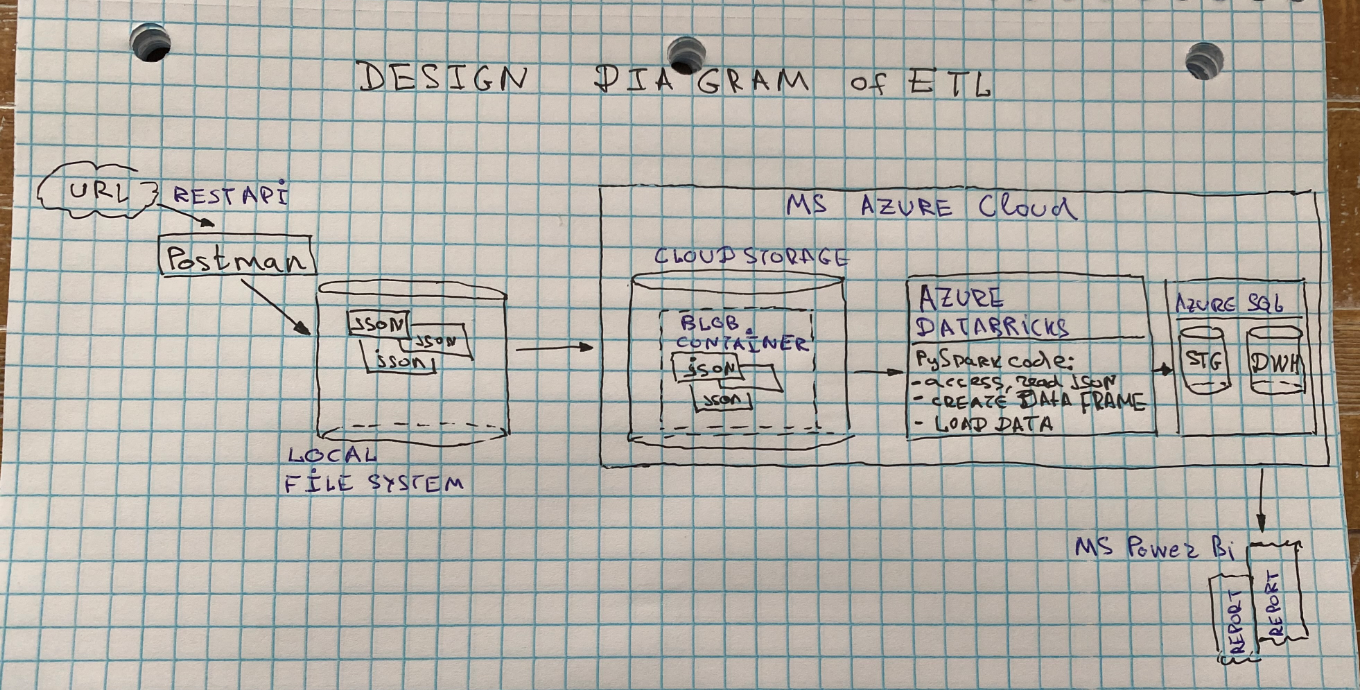
Staging is responsible for loading raw data

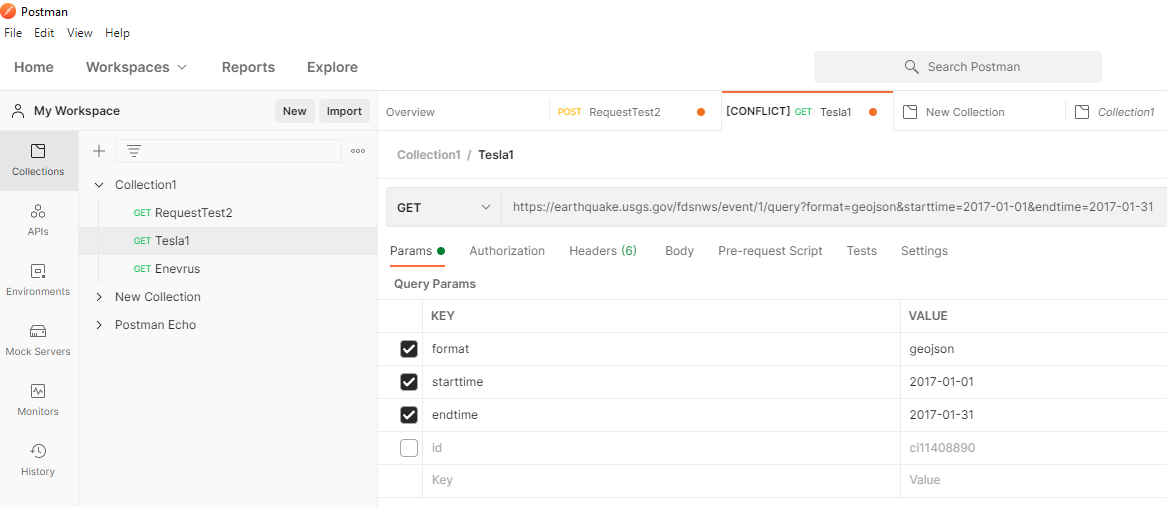
DWH – has cleaned data, generated SK, indexes and foreign keys. I will do queries from DWH’s tables

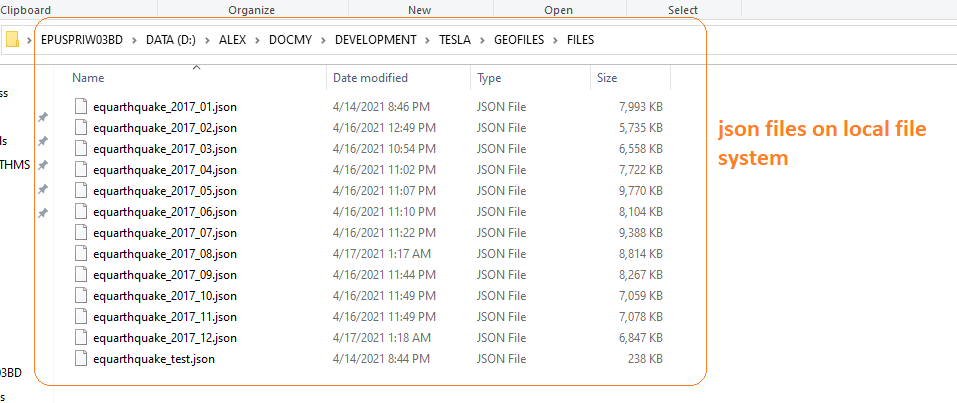
1. Load data from DataFrames into staging
2. Load data from staging to dwh

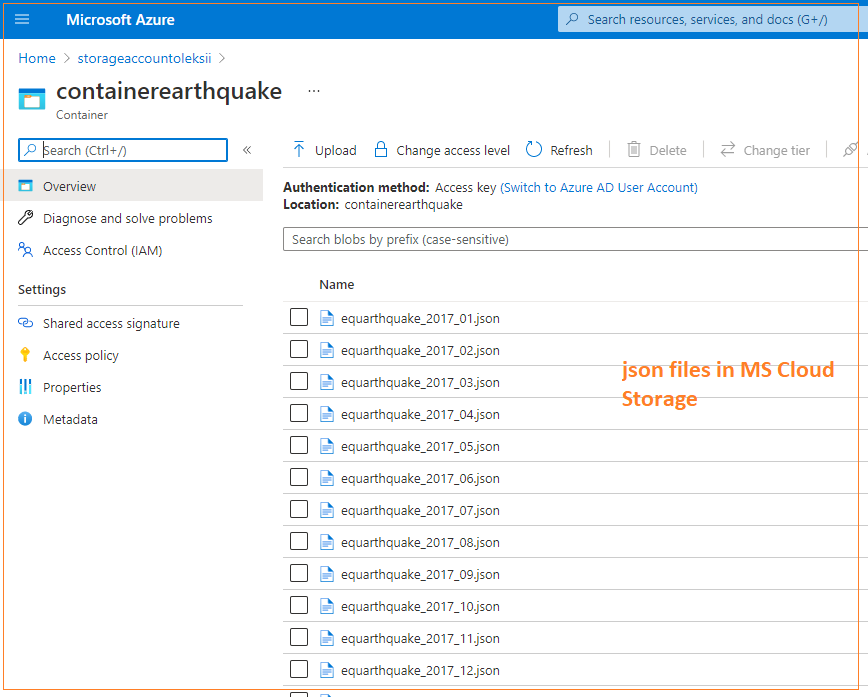
Note: *it is not the optimal solution generate json files from url and then put them on Cloud storage and then proceed. The better solution would be directly access rest api from DataBricks.*

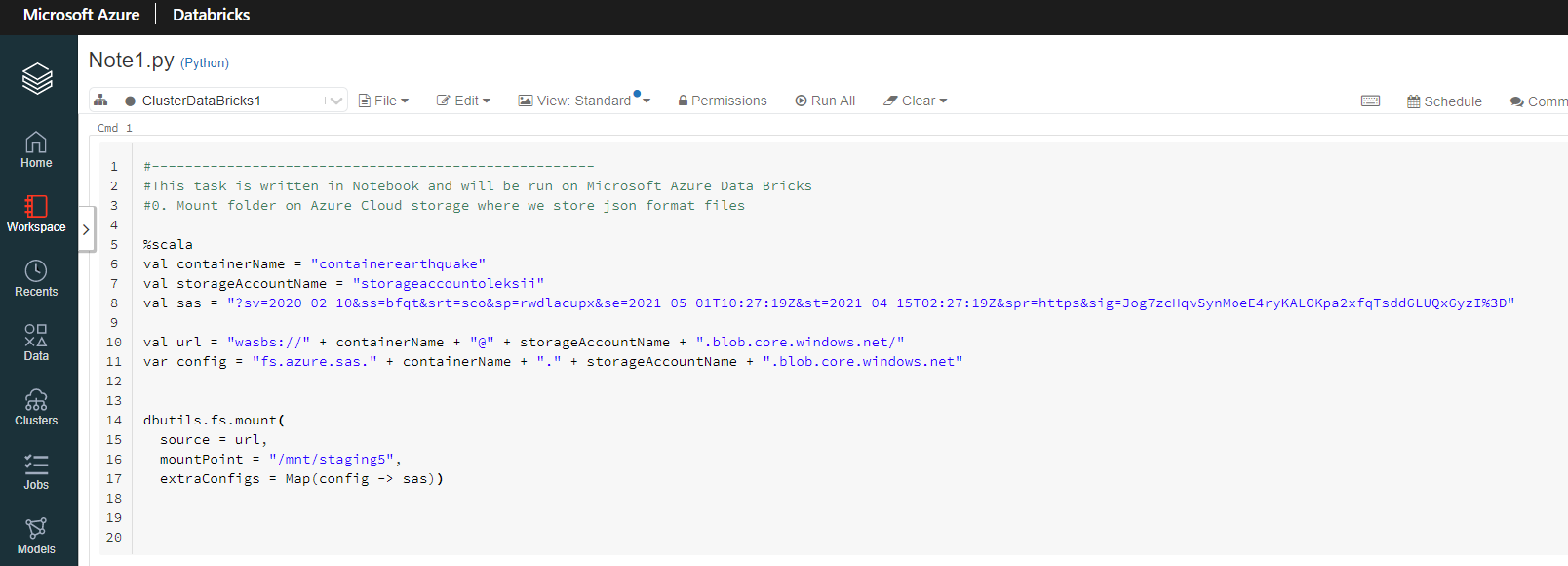
*However, I faced an authentication error to access rest api from DataBricks. My Azure subscription doe not have admin rights and I could not solve that permission error. As an alternative approach, I processed json files*.

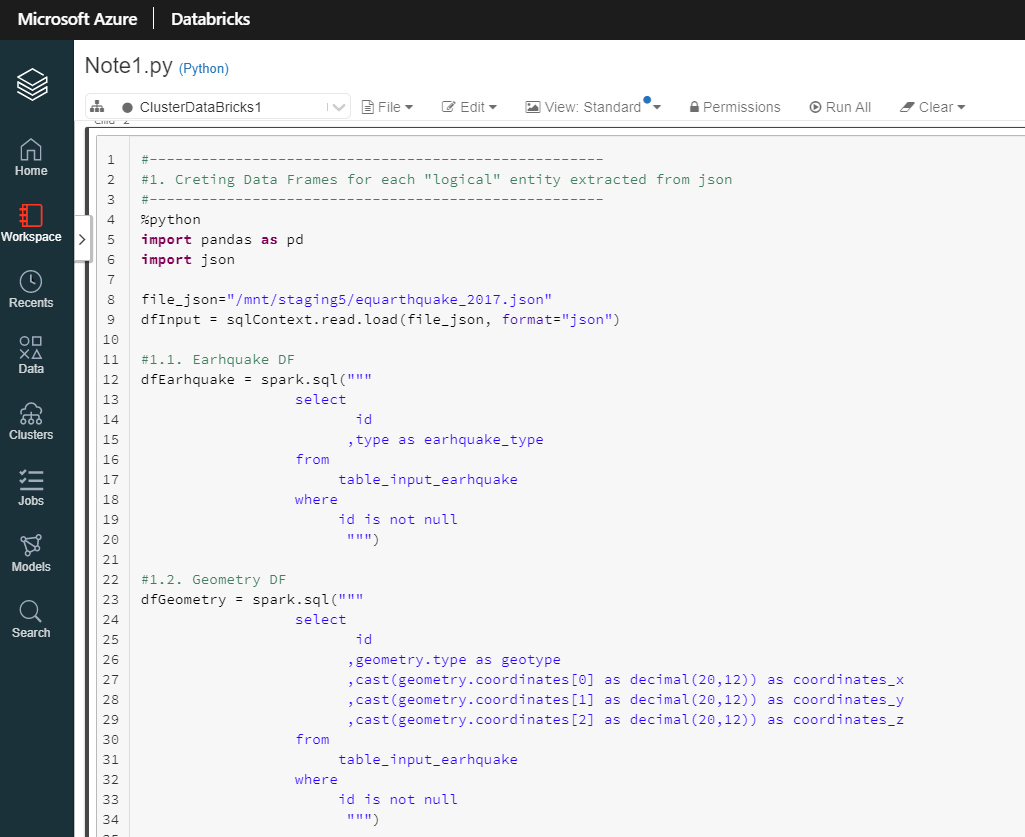


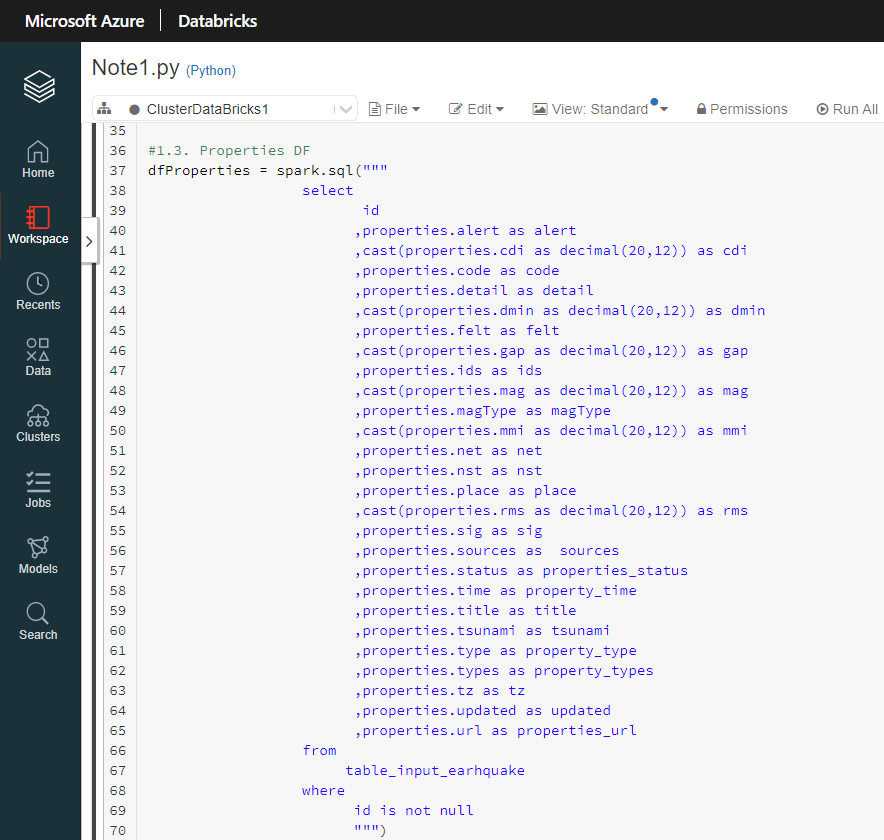


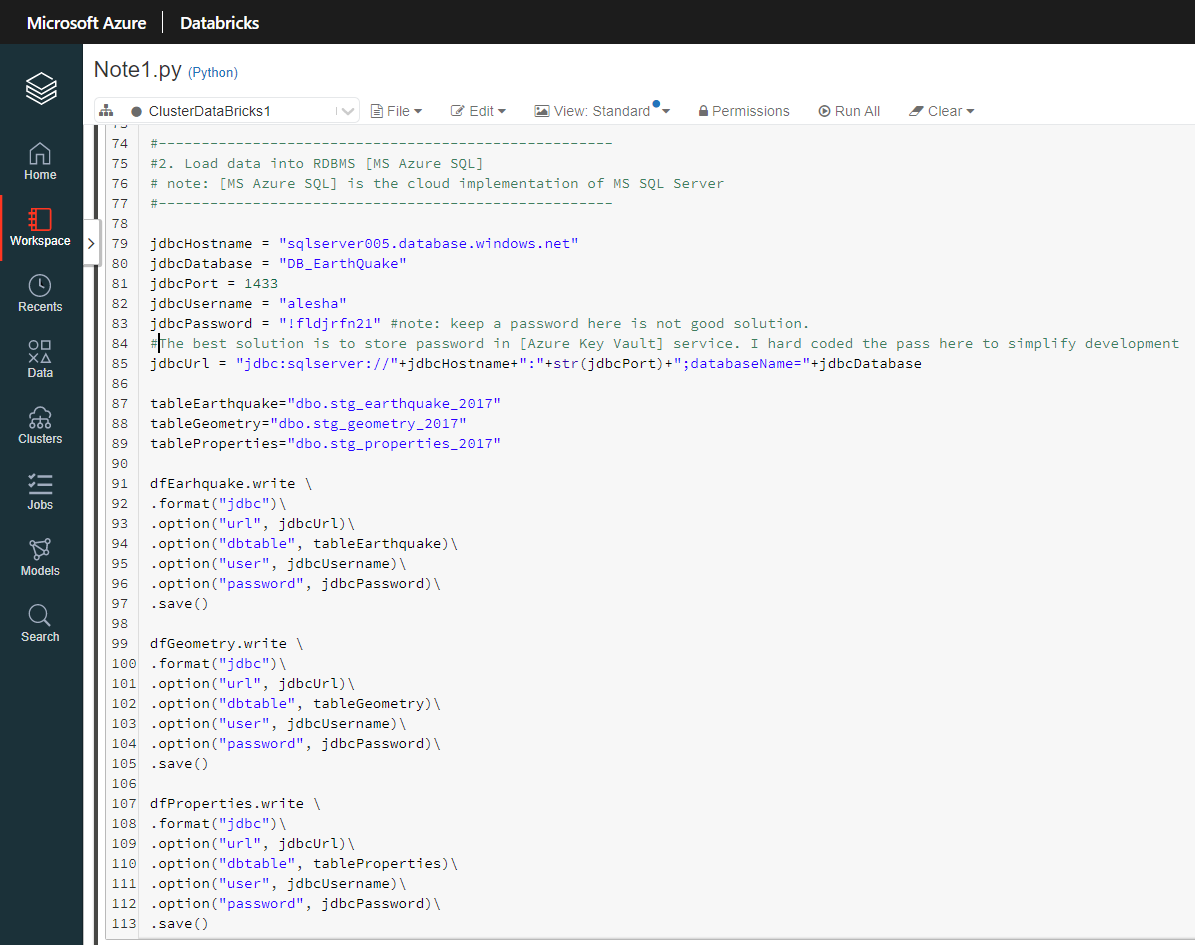


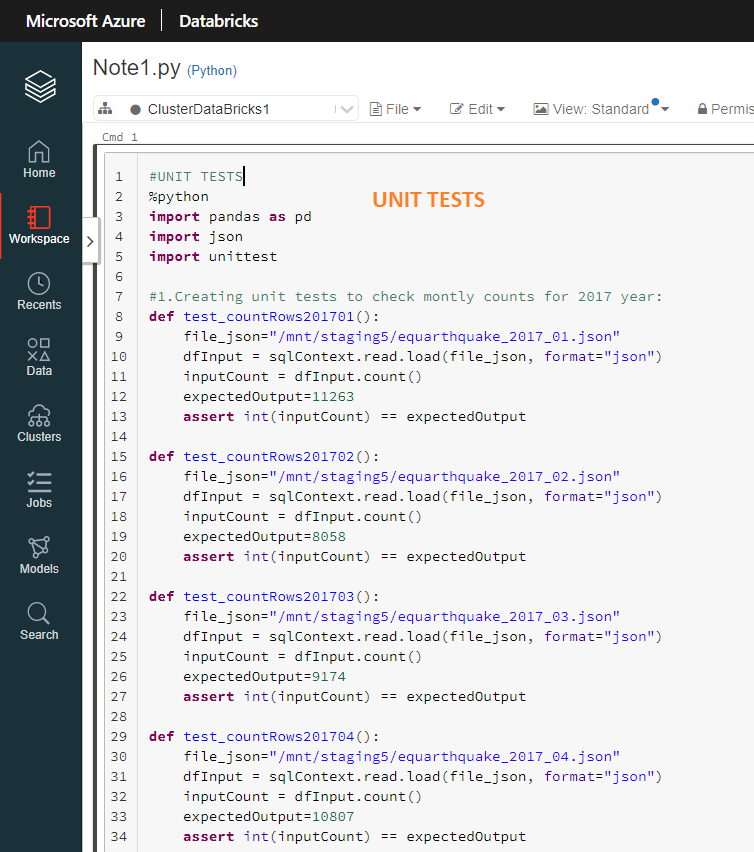


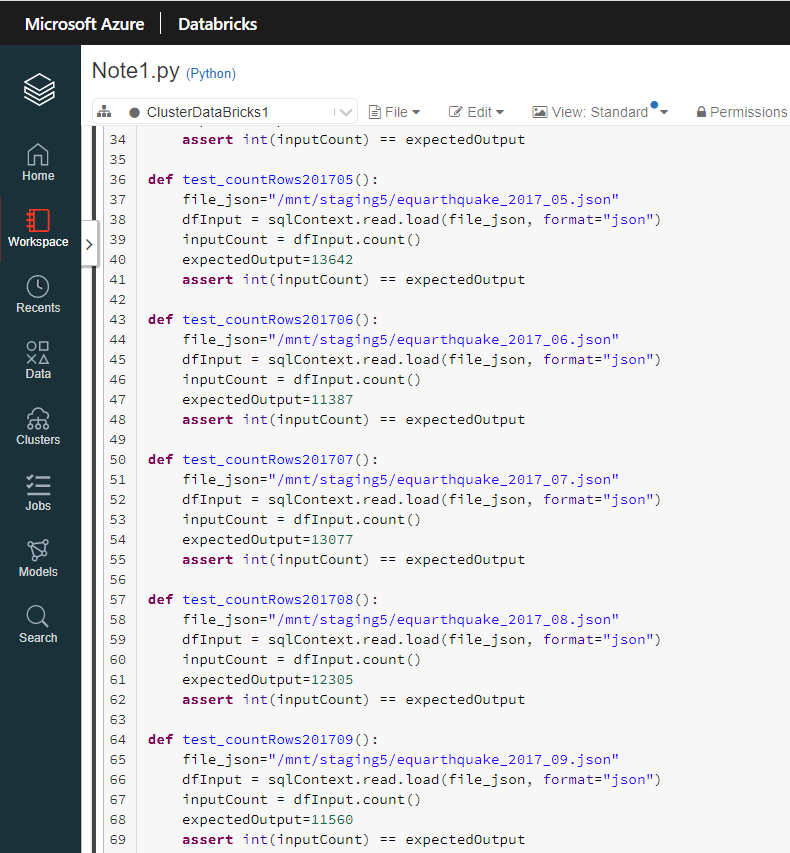


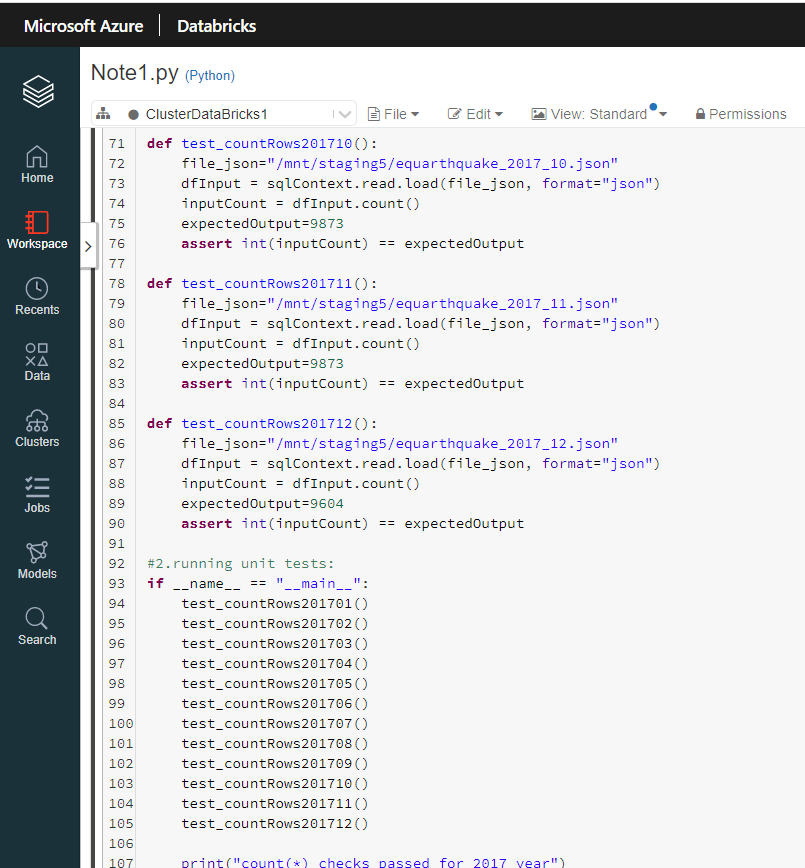




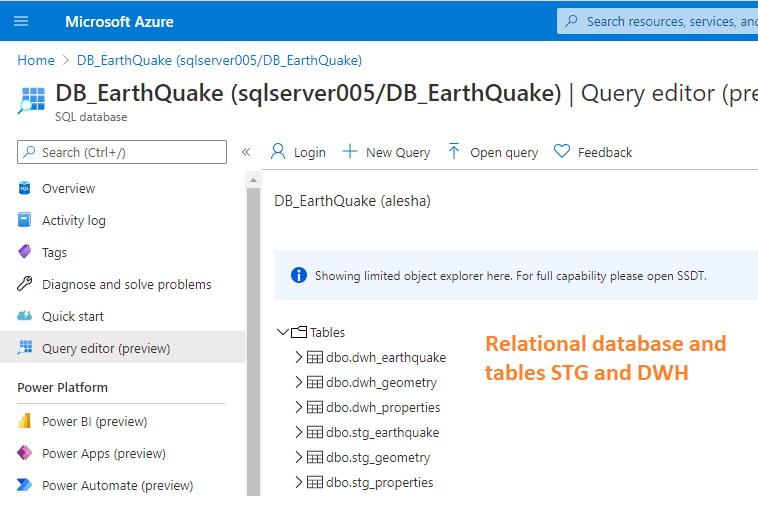


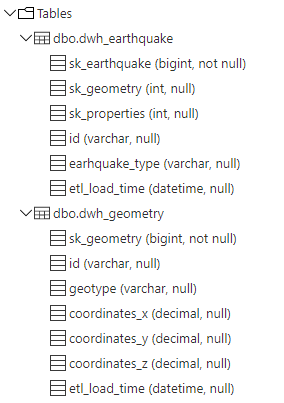


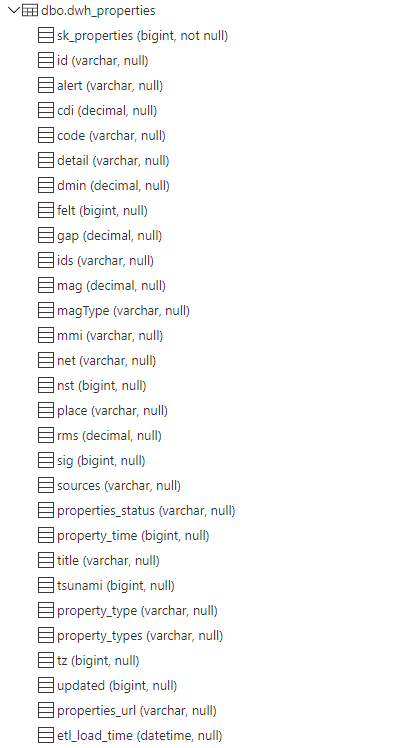




**TASK 2.4.** *Store the response in those database objects.*







**TASK 2.5.SQL.** *Provide query/analysis to give biggest earthquake of 2017*.

--sql:

SELECT

        id

       ,mag

       ,place

       ,dt\_EarthQuake

FROM (

SELECT

        id

       ,mag

       ,place

       ,dateadd(S,convert(int,left(property\_time,10)), '1970-01-01') as dt\_EarthQuake

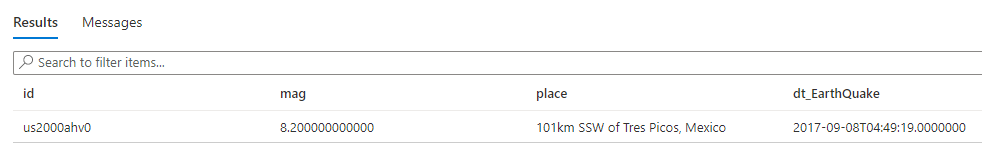
       ,dense\_rank() over(order by mag desc) as rn

FROM

     dwh\_properties) t

WHERE

      rn=1;



**TASK 2.6.SQL.** *Provide query/analysis to give most probable hour of the day for the earthquakes bucketed by the range of magnitude*.

--sql:

WITH

    tmp(bucket,hourOfEarhquake,amount) AS

(

SELECT

               CASE

                   WHEN mag >=0 and mag<1 THEN 0

                   WHEN mag >=1 and mag<2 THEN 1

                   WHEN mag >=2 and mag<3 THEN 3

                   WHEN mag >=3 and mag<4 THEN 4

                   WHEN mag >=4 and mag<5 THEN 5

                   WHEN mag >=6 THEN 6

                   ELSE -1

                END

                   AS bucket

               ,datepart(hour,dateadd(S,convert(int,left(property\_time,10)), '1970-01-01')) hourOfEarhquake

               ,count(\*) as amount

from dwh\_properties

group by

               CASE

                   WHEN mag >=0 and mag<1 THEN 0

                   WHEN mag >=1 and mag<2 THEN 1

                   WHEN mag >=2 and mag<3 THEN 3

                   WHEN mag >=3 and mag<4 THEN 4

                   WHEN mag >=4 and mag<5 THEN 5

                   WHEN mag >=6 THEN 6

                   ELSE -1

                END

               ,datepart(hour,dateadd(S,convert(int,left(property\_time,10)), '1970-01-01'))

),

    tmp2(bucket,hourOfEarhquake,amount,rn)

AS (

SELECT

       bucket

      ,hourOfEarhquake

      ,amount

      ,row\_number() over(partition by bucket order by amount desc) as rn

FROM

     tmp

)

 --main part:

 SELECT

       bucket

      ,hourOfEarhquake

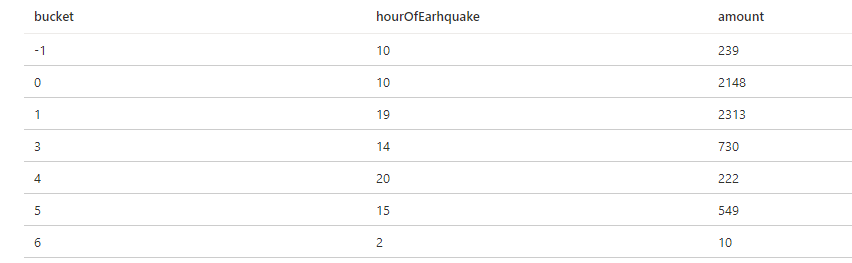
      ,amount

 FROM

     tmp2

 WHERE

      rn=1;

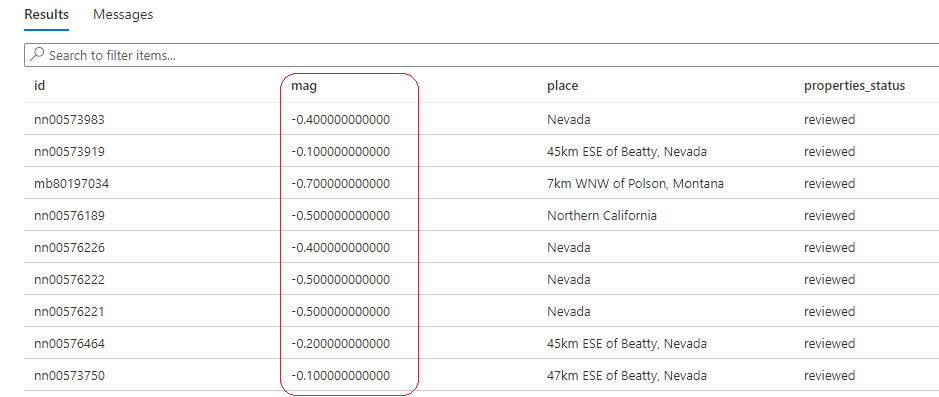


Note: I added the bucket [-1] to keep rows where [mag] has negative values or is null. For, example

--sql

SELECT id, mag, place, properties\_status FROM dwh\_properties

WHERE (mag is null) or (mag < 0);



**TASK 2.7.** *Any interesting visualization*.

I created a dashboard for the sql-query [2.5] in MS Power BI Provide

