AZURE PROJECT
NTT DATA SERVICES

By

SHANMUGA ARTHI DHARMARAJ

Guided By

MALLIKARJUNA GANDHAMSETTY

Table of Contents:

==========

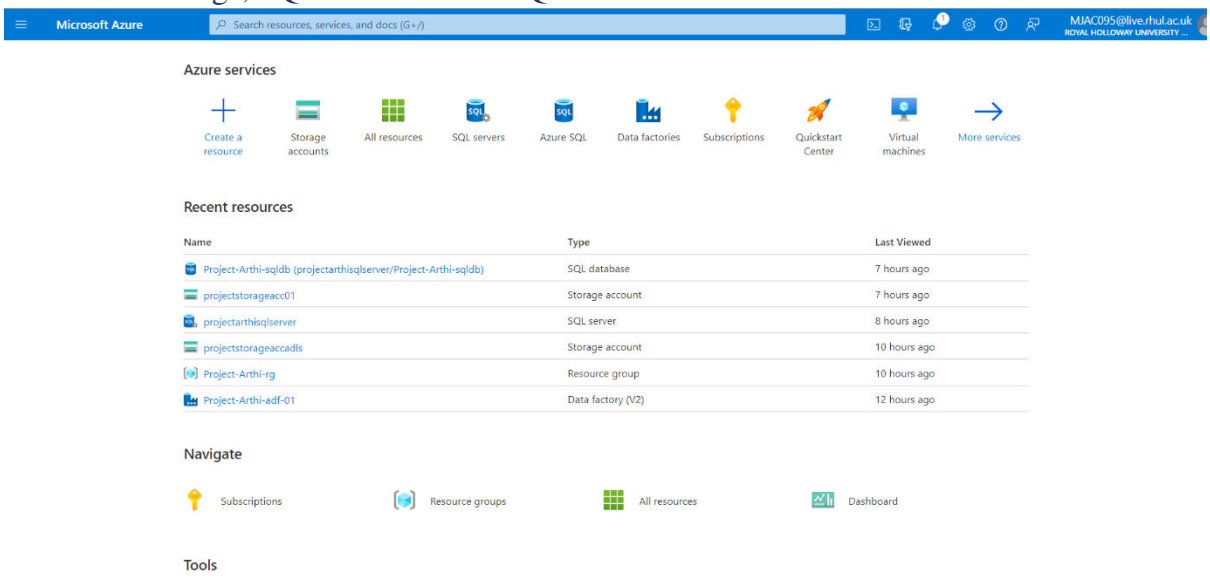
Problem Statement 1: **Ingest data from a single Table in SQL Server database to Azure Blob storage**

==========

Solution :

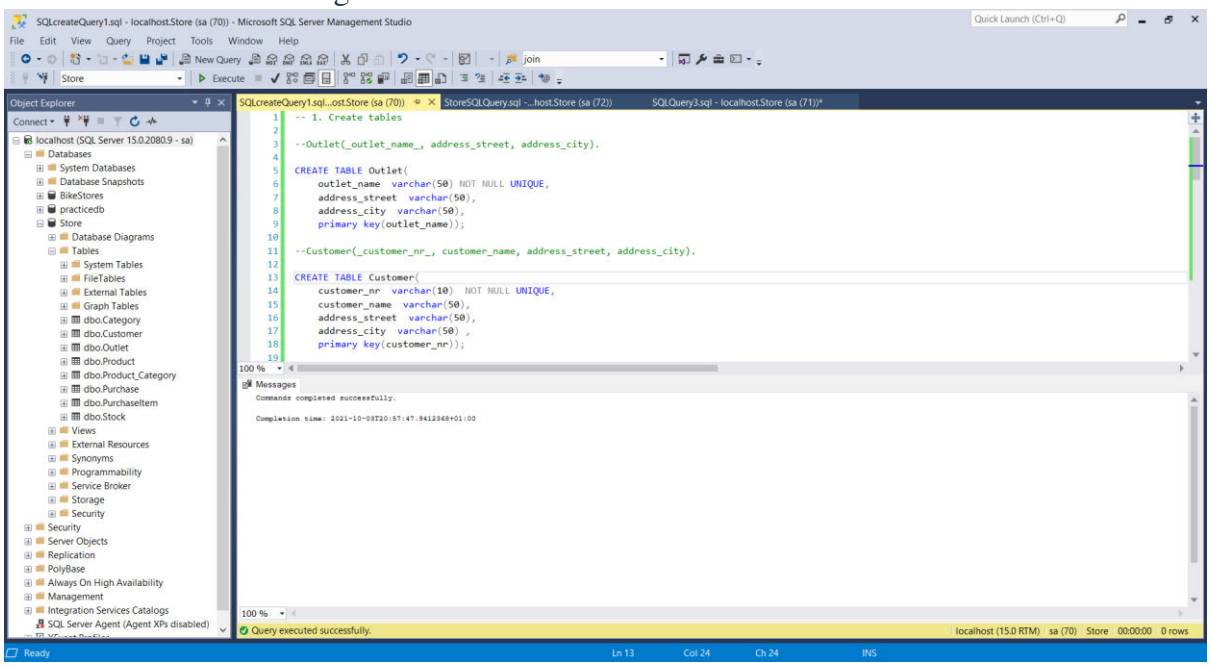
- Created a Resource group, named as: Project-Arthi-rg
- Created a Blob Storage Account, named as: projectstorageacc01
- Created a data factory, named as: Project-Arthi-adf-01
- Created a self-hosted integration runtime named as: IR-1

Screenshot 1.1: Resource Group, Data Factory, Storage Account – Blob Storage and Azure Data Lake Storage, SQL Database and SQL Server – resources created details

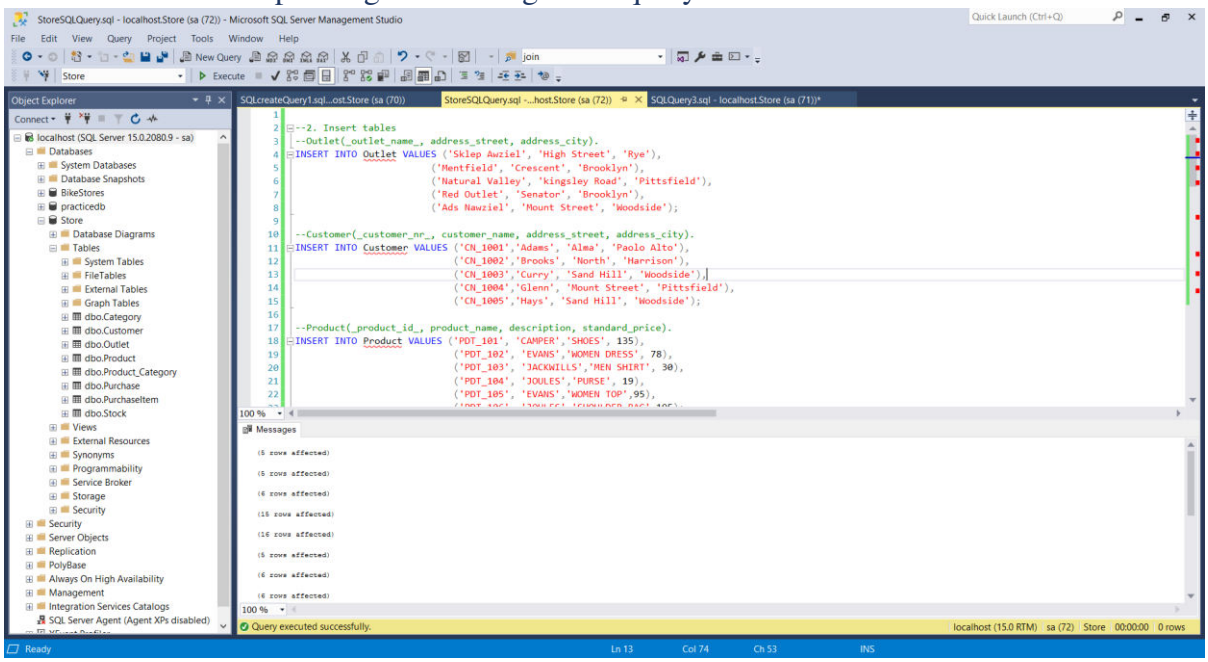


- Create SQL Server and Azure Storage linked services: In SQL Server Management System, created own Database named **Store** containing 7 Tables namely **dbo.Category, dbo.Customer,dbo.Outlet,dbo.Product,dbo.Product_Category, dbo.Purchase, dbo.Purchaseitem,dbo.Stock.**
- Created database and populated the table using insert query values in each table.

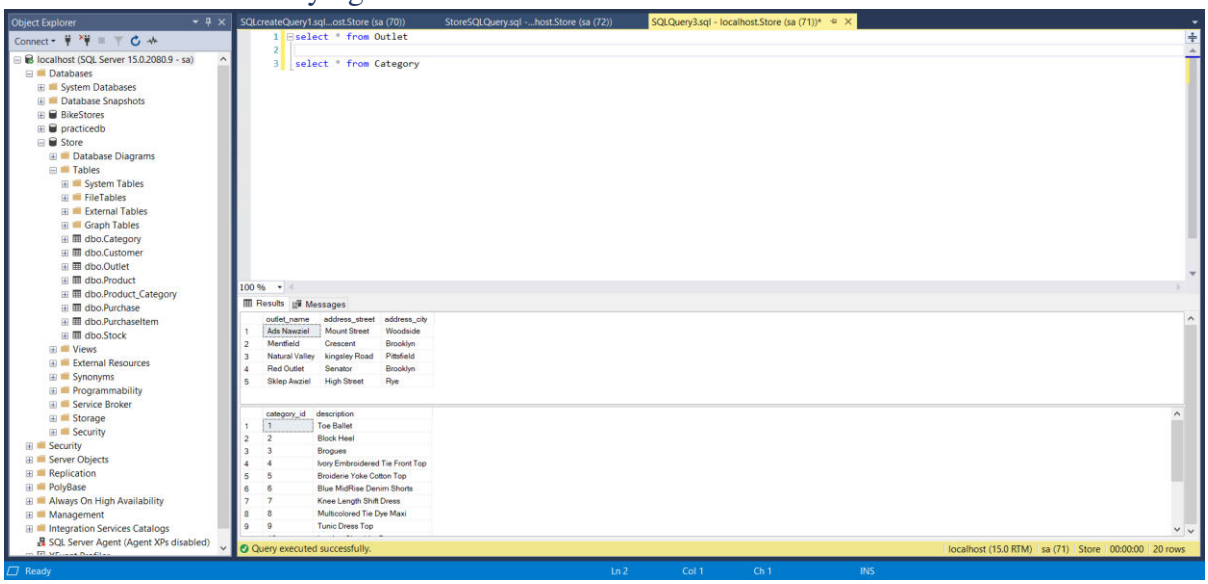
Screenshot 1.2.a: Creating DB in SSMS



Screenshot 1.2.b: Populating Table using insert query in SSMS



Screenshot 1.2.c: Verifying the data inserted in Tables



Sample: **dbo.Outlet**

-- 1. Create Tables

--Outlet(_outlet_name_, address_street, address_city).

```
CREATE TABLE Outlet(
        outlet_name  varchar(50) NOT NULL UNIQUE,
        address_street  varchar(50),
        address_city  varchar(50),
        primary key(outlet_name));
```

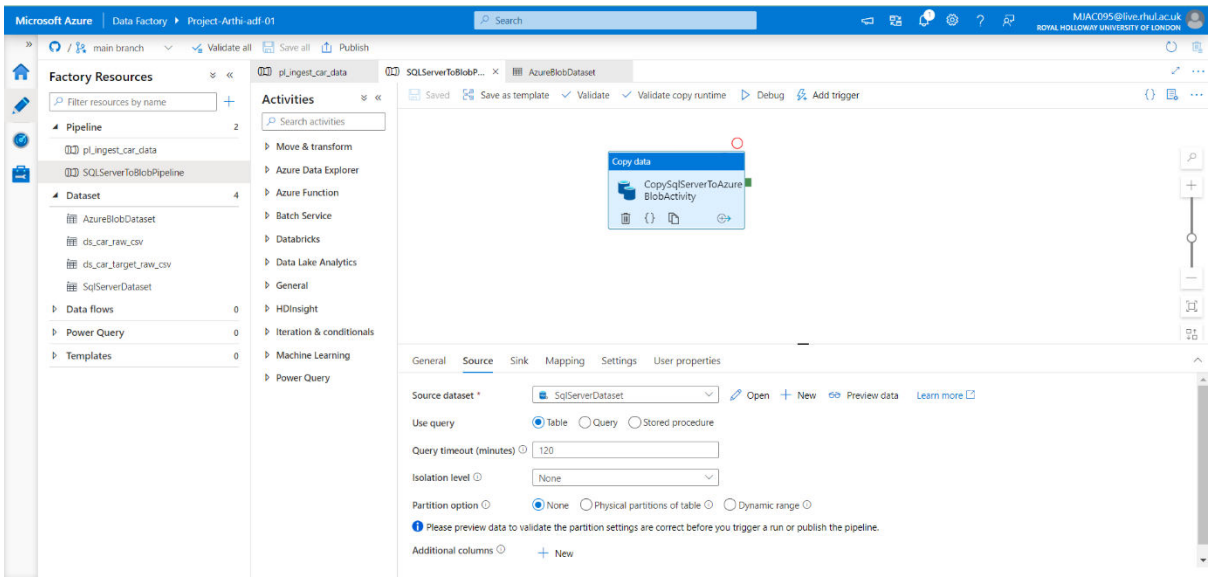
--2. Insert Tables

--Outlet(_outlet_name_, address_street, address_city).

```
INSERT INTO Outlet VALUES ('Sklep Awziel', 'High Street', 'Rye'),
                          ('Mentfield', 'Crescent', 'Brooklyn'),
                          ('Natural Valley', 'kingsley Road',
'Pittsfield'),
                          ('Red Outlet', 'Senator', 'Brooklyn'),
                          ('Ads Nawziel', 'Mount Street', 'Woodside');
```

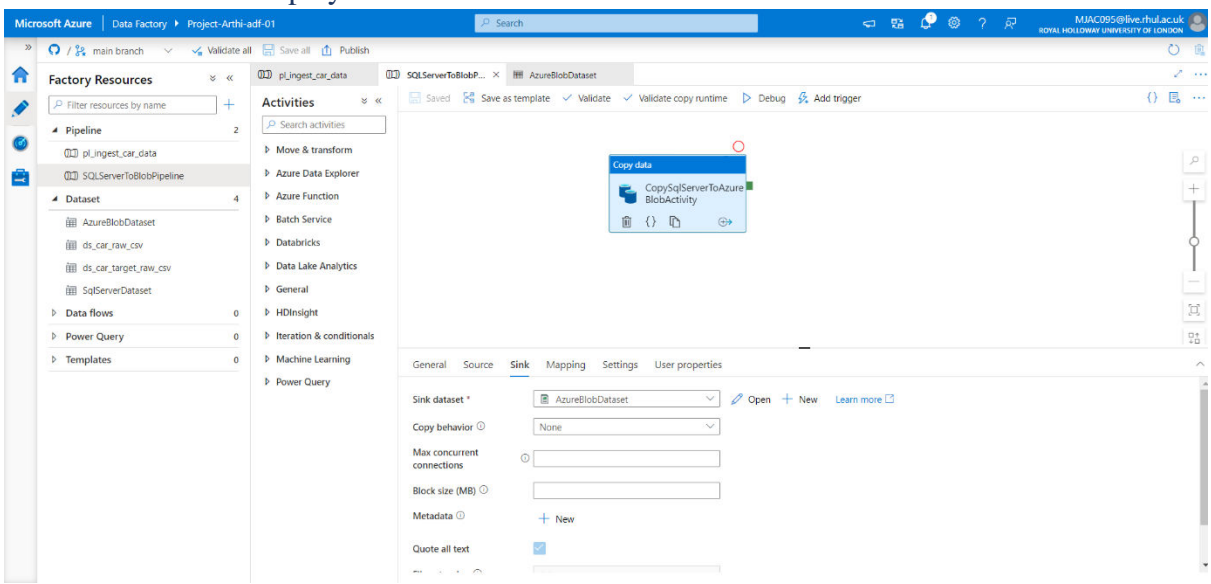
- Created SQL Server and Azure Blob datasets named as **projectstorageacc01 as** showed in Screenshot 1.
- Created a pipeline named **SQLSeverToBlobPipeline** with a copy activity **"CopySqlServerToAzureBlobActivity"** to move the data from on-premises to blob storage. Then set up Source dataset: **SqlServerDataset.**

Screenshot 1.3.a: Displays both Pipelines, Copy Activity and Source Dataset



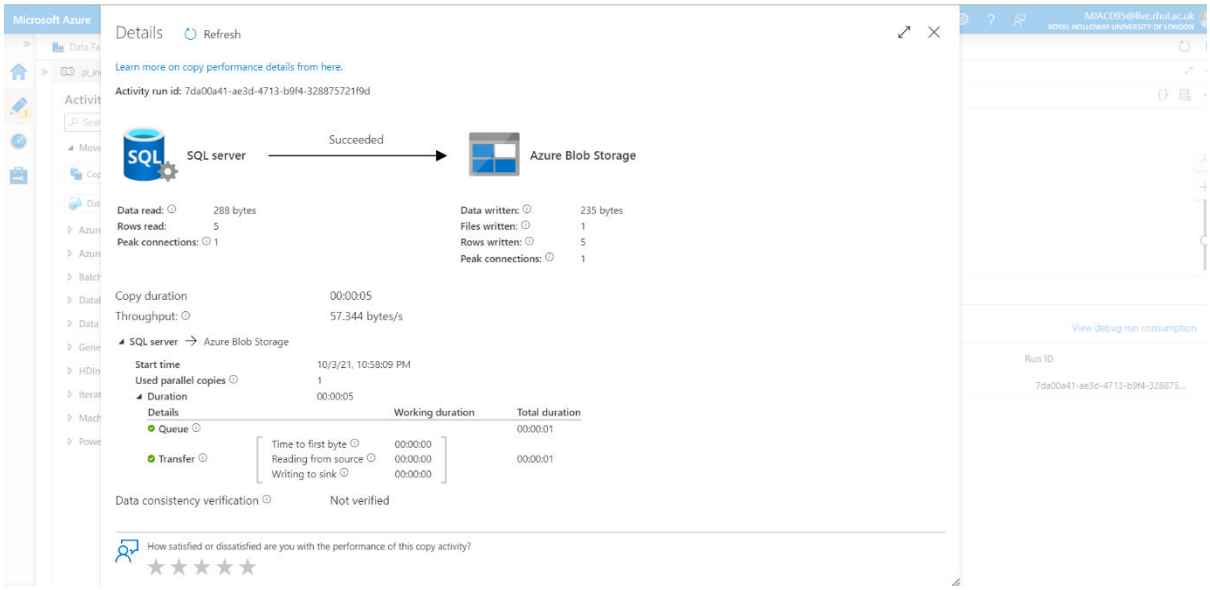
- Sink dataset as: **AzureBlobDataset, created file path as myblob / Directory: fromonprem/ File: @CONCAT(pipeline().RunId, '.txt'),** to rename file as PipelineRunID.txt.

Screenshot 1.3.b: Displays Sink Dataset

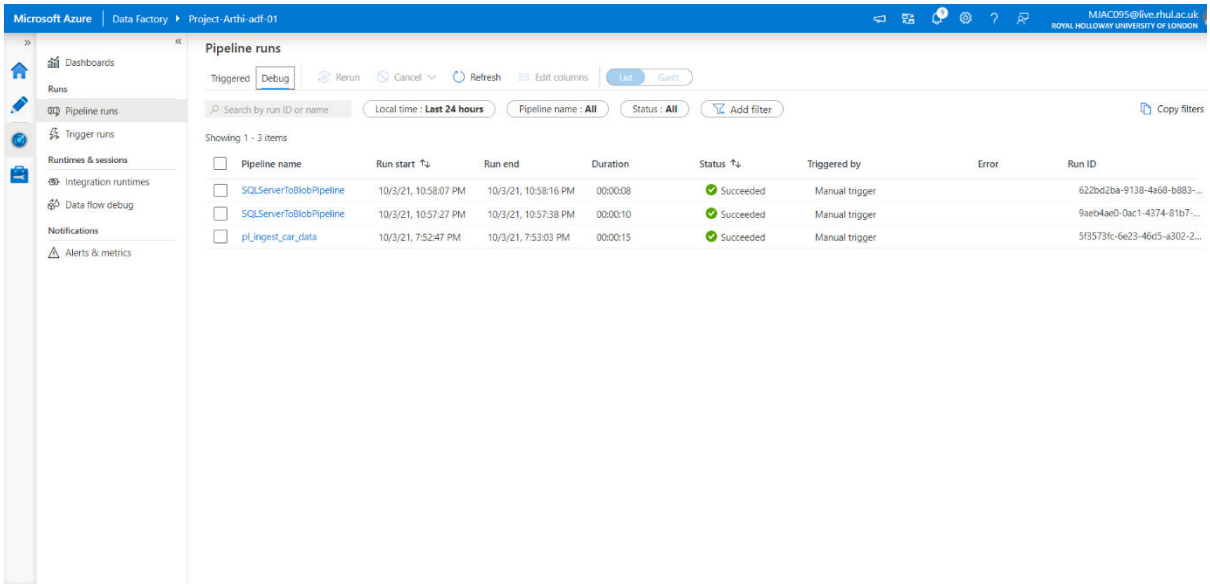


- If the file section adds dynamic content not given user, it will automatically get the filename from the copied data.
- Started a pipeline run, debugged, pipeline validation done. Published All

Screenshot 1.4.a: Pipeline Activity Details

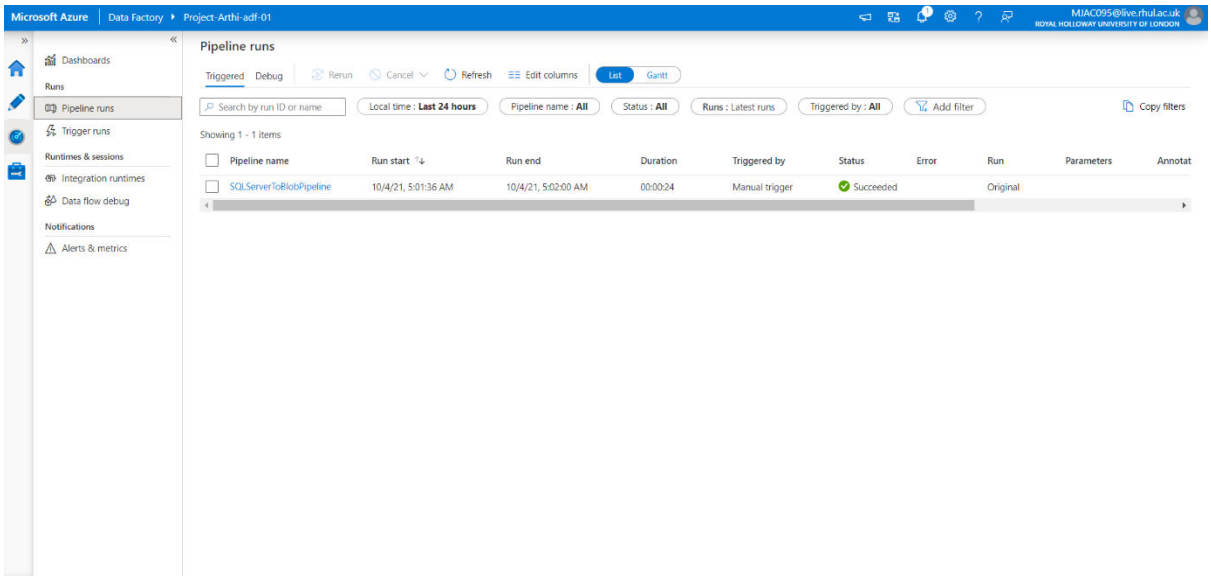


Screenshot 1.4.b: Pipeline Debug Details



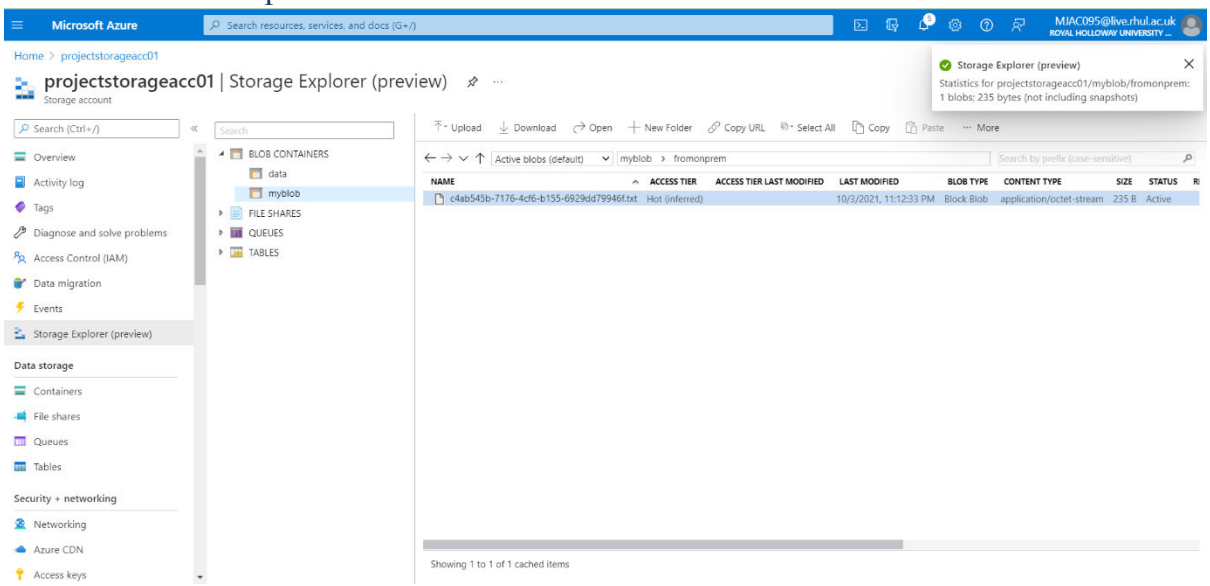
- Added trigger by clicking on trigger now
- Monitoring the relative pipeline SQLServerToBlobPipeline

Screenshot 1.4.c: Pipeline Manual Trigger Details



- Output verified: Table copied to Azure blob storage as .txt file.

Screenshot 1.5: Output Verification



Files: Output text files for this Problem in Solution1 Folder.

==========

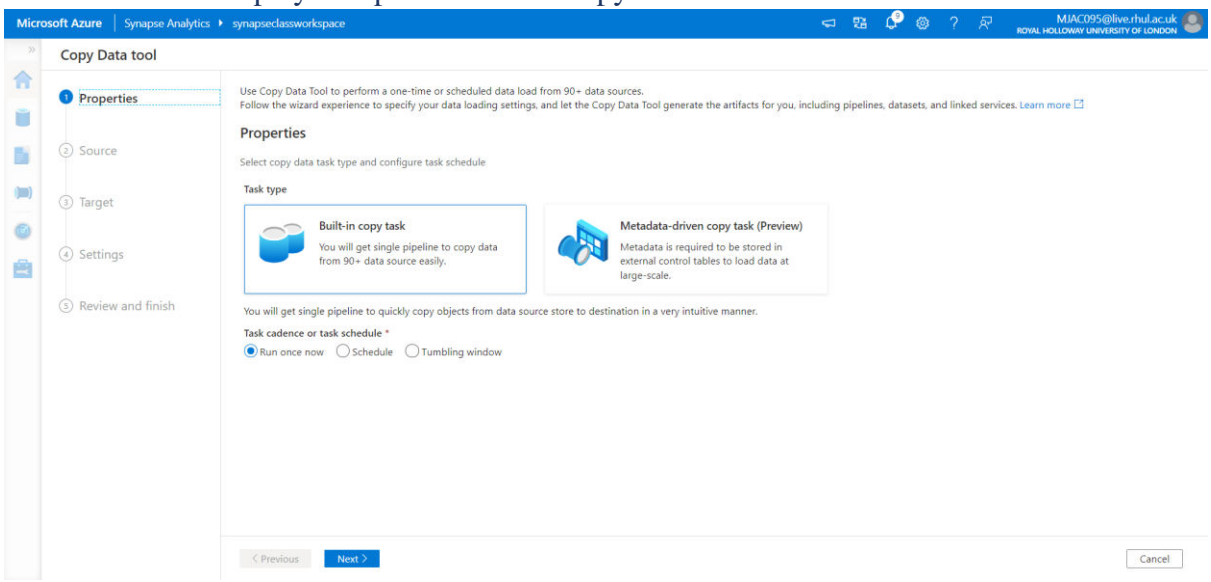
Problem Statement 2: **Ingest data from multiple Tables in SQL Server database to Azure Synapse Analytics**

==========

Solution :

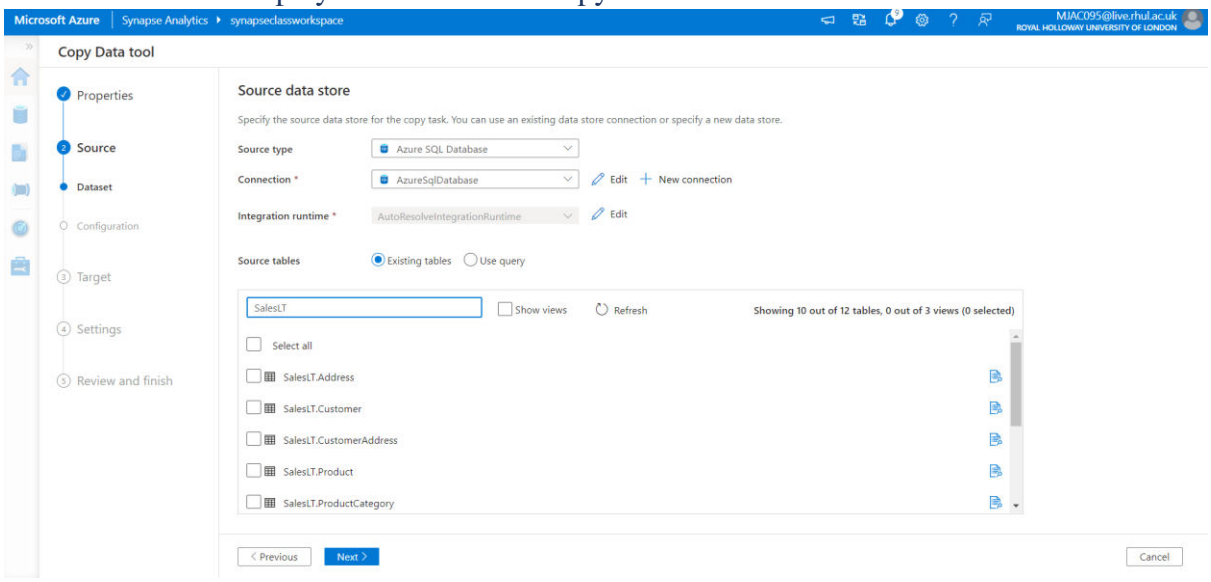
- Created a Resource group, named as: Project-Arthi-rg
- Created a Blob Storage Account, named as: projectstorageacc01
- Created Azure Synapse Workspace named as: synapseclassworkspace
- In the Synapse Workspace, click Ingest:
    - Copy Data Tool is Displayed
    - Chose Built-in copy Task

Screenshot 2.1: Displays Properties Tab of Copy Data Tool

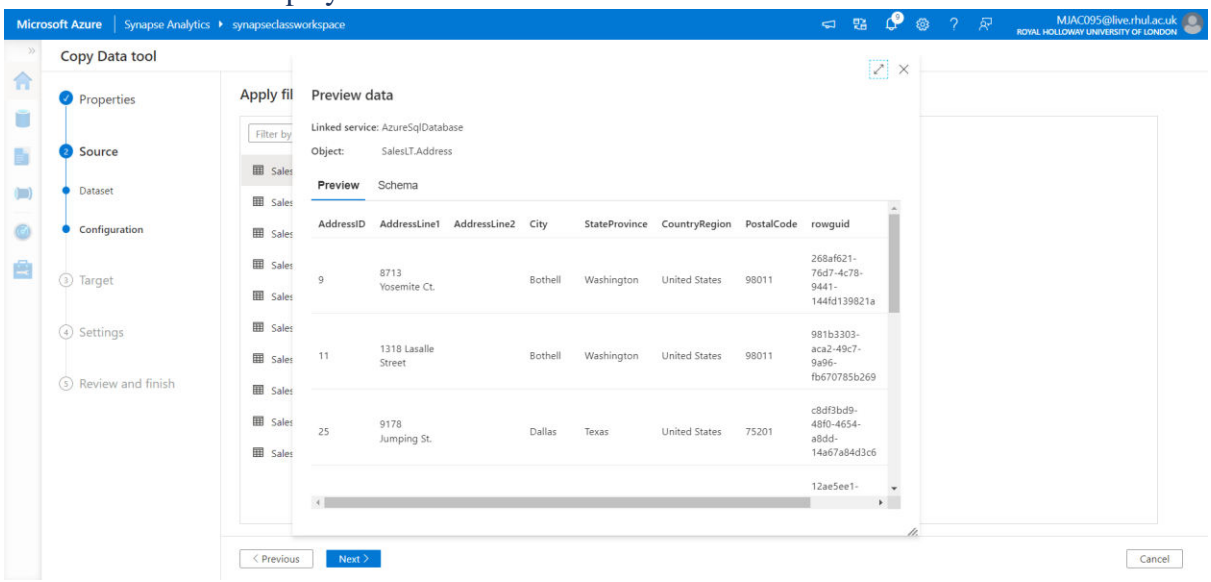


- In the Source Tab,
    - Choose Source Type: Azure SQL Database
    - Choose Connection: AzureSqlDatabse
    - Source Tables: chose existing Tables,
        - entered “SalesLT”
        - Select check box SelectAll

Screenshot 2.2.a: Displays Source Tab of Copy Data Tool



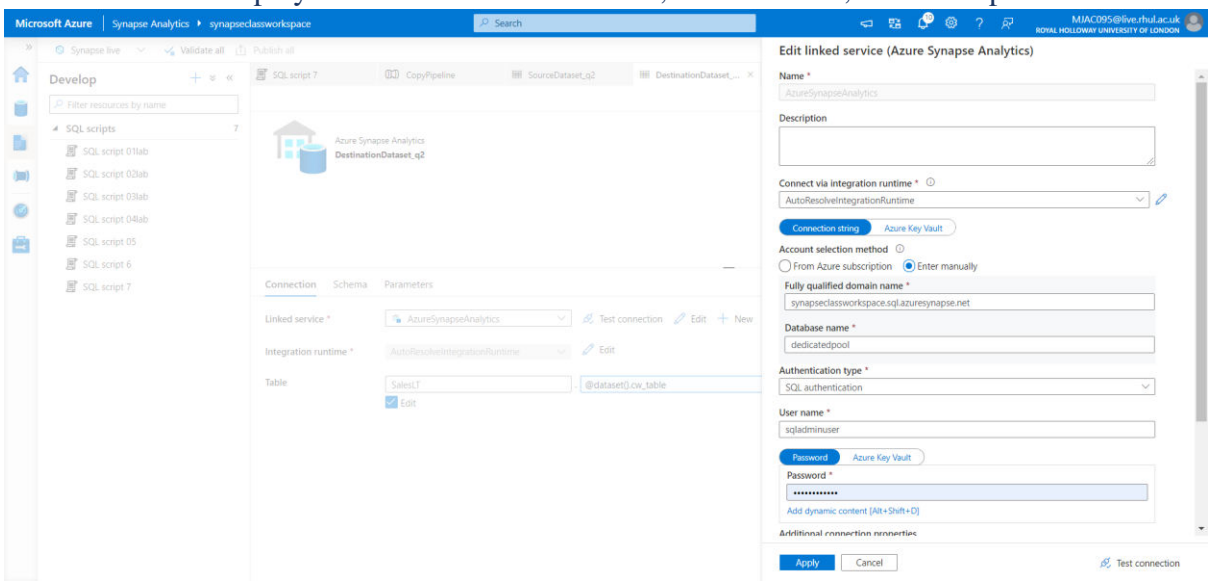
Screenshot 2.2.b.: Displays Preview Data for all the Tables of SalesLT



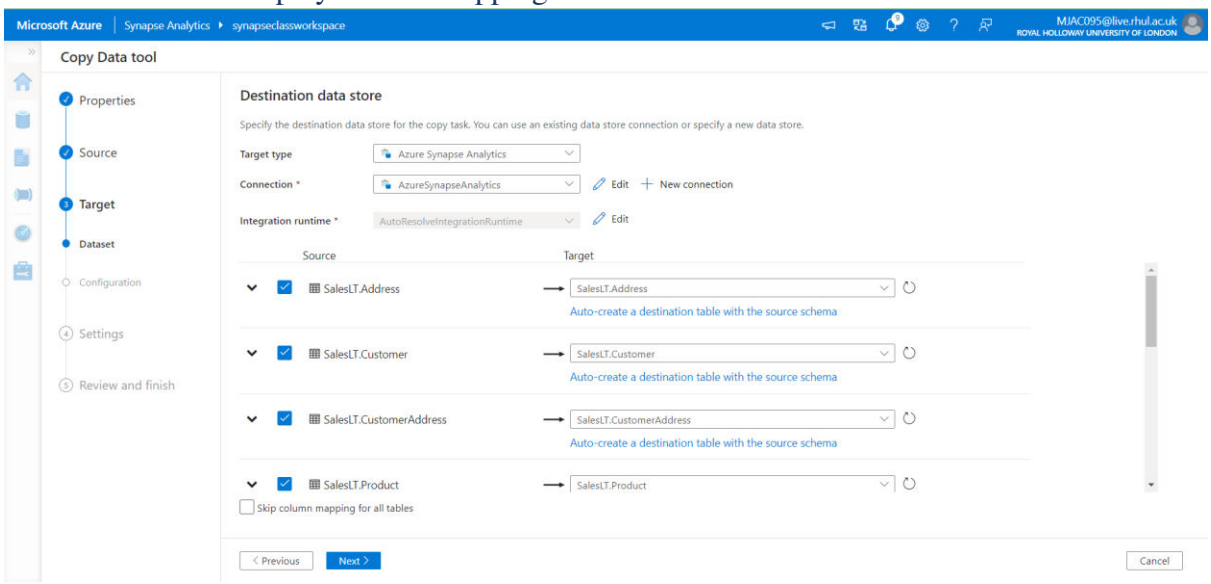
- In the Target Tab,
  - Choose Source Type: Azure Synapse Analytics
  - Choose Connection: AzureSynapseAnalytics
    - Choose server name: synapseclassworkspace.sql.azuresynapse.net
    - Database: dedicatedpool

- In the Destination Data Store:
    - Table mapping should be done,
    - Column mapping has to be done,
    - Deselect Type conversion.

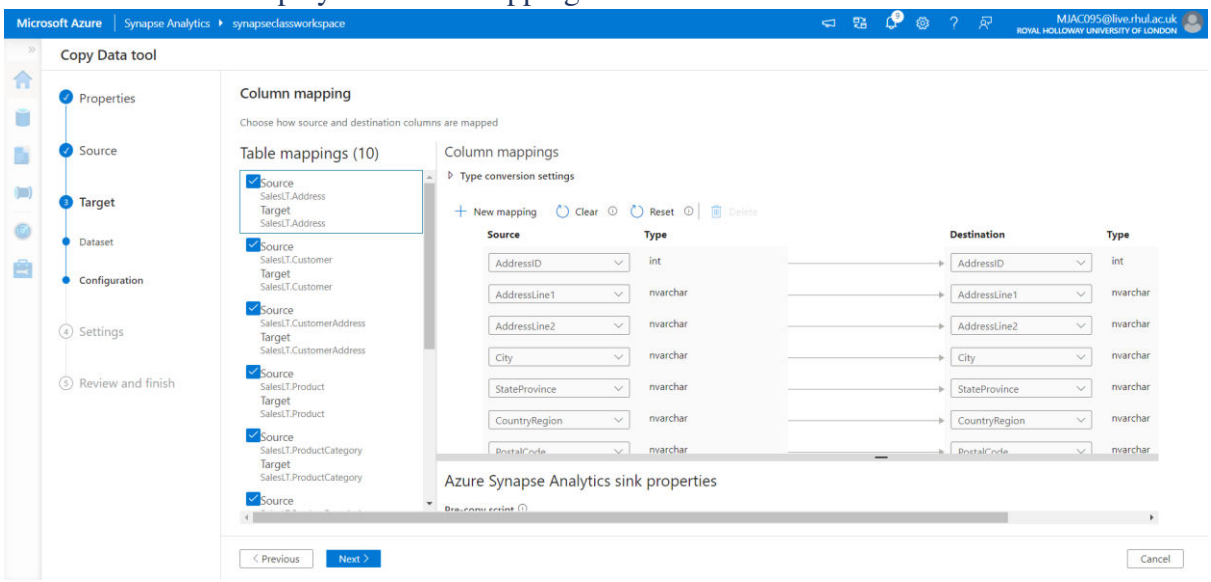
Screenshot 2.3: Displays Sink Data Linked Service, Server Name, Dedicatedpool



Screenshot 2.4.a: Displays Table Mapping Source to Sink

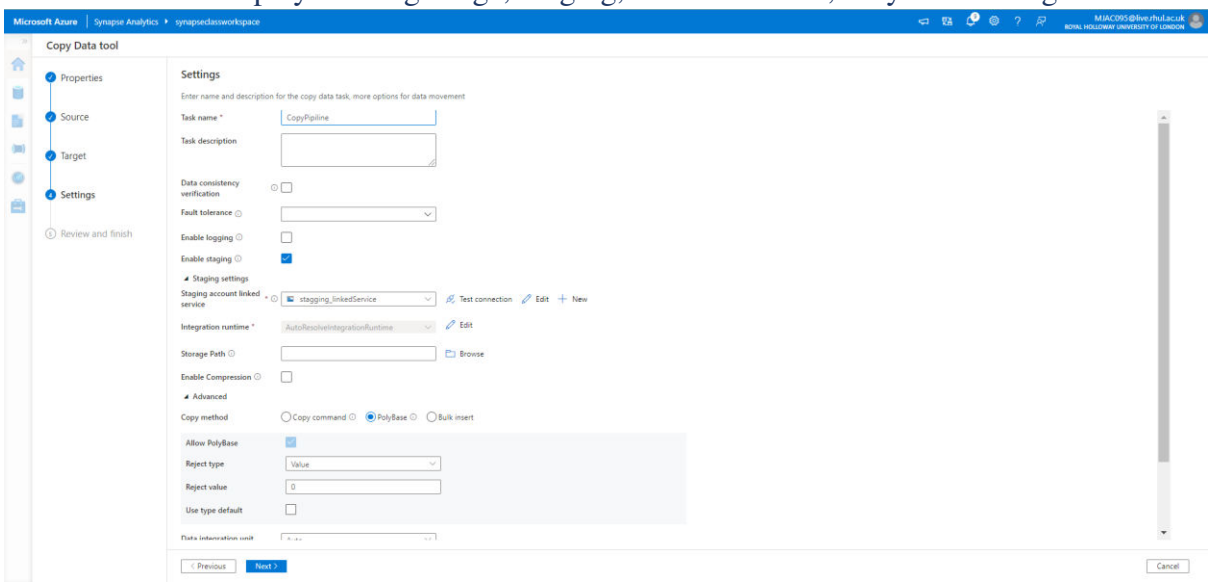


Screenshot 2.4.b: Displays Column Mapping Source to Sink

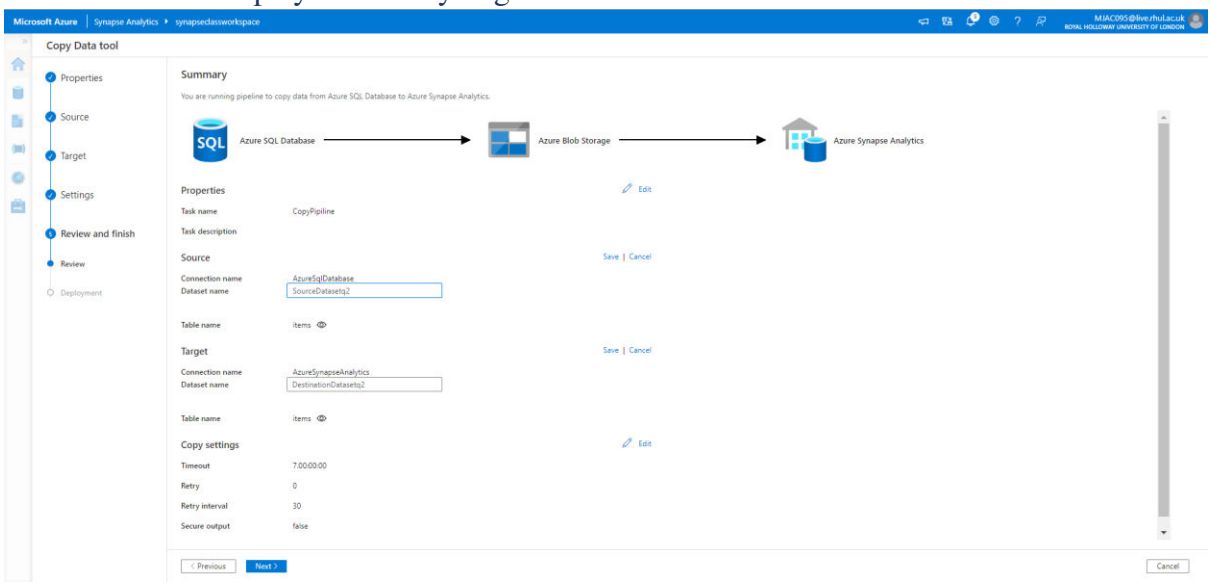


- The storage is used for staging the data before it loads into Azure Synapse Analytics by using PolyBase.
- After the copy is complete, the temporary data in Azure Blob Storage is automatically cleaned up.

Screenshot 2.5: Displays Settings Page, Staging, Linked Service, PolyBase setting

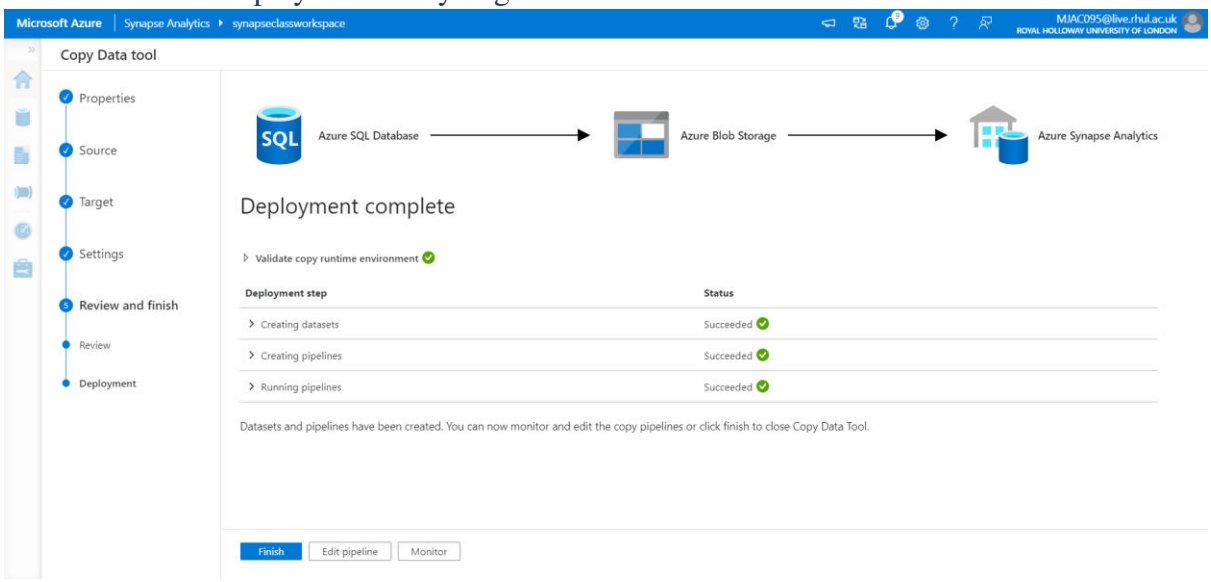


Screenshot 2.6: Displays Summary Page

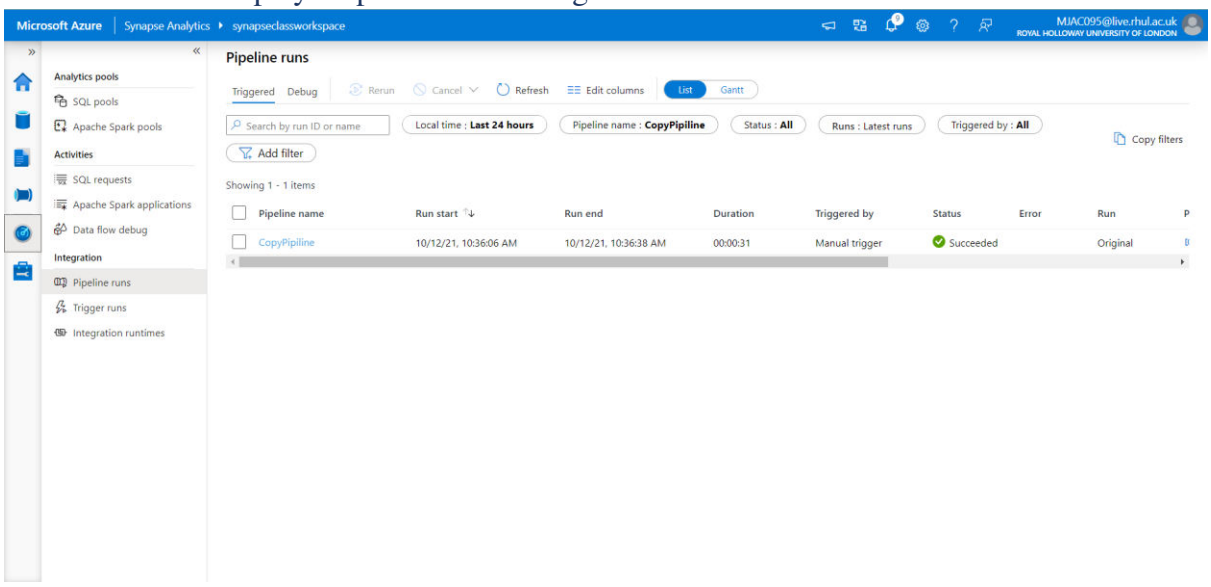


- Now, we have created the required datasets and pipeline settings,

Screenshot 2.7: Displays Summary Page

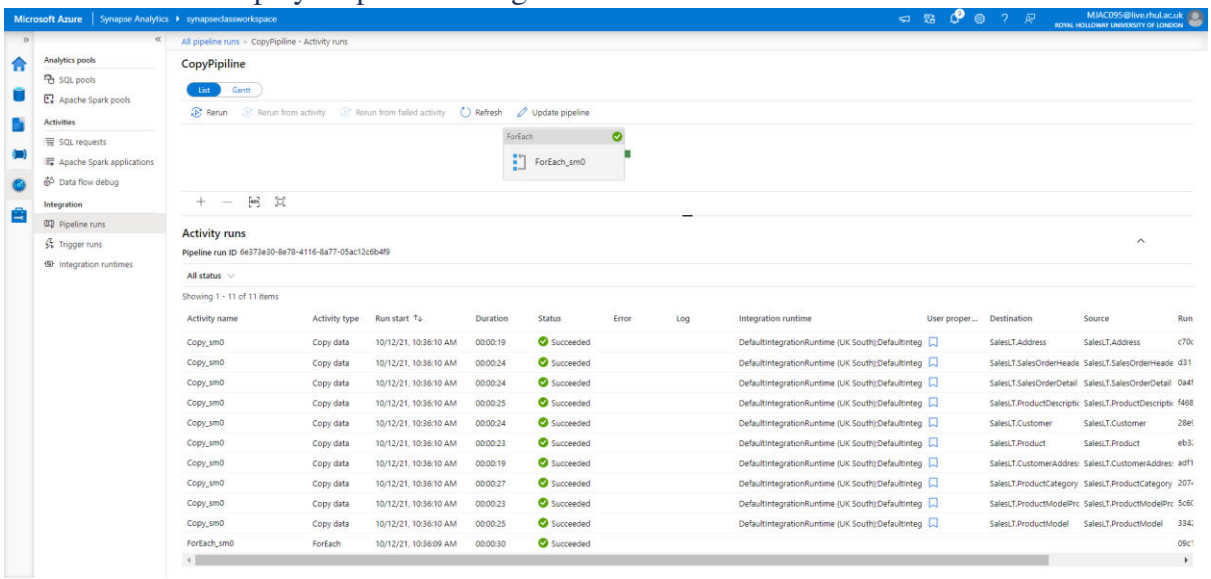


Screenshot 2.8: Displays Pipeline Monitoring



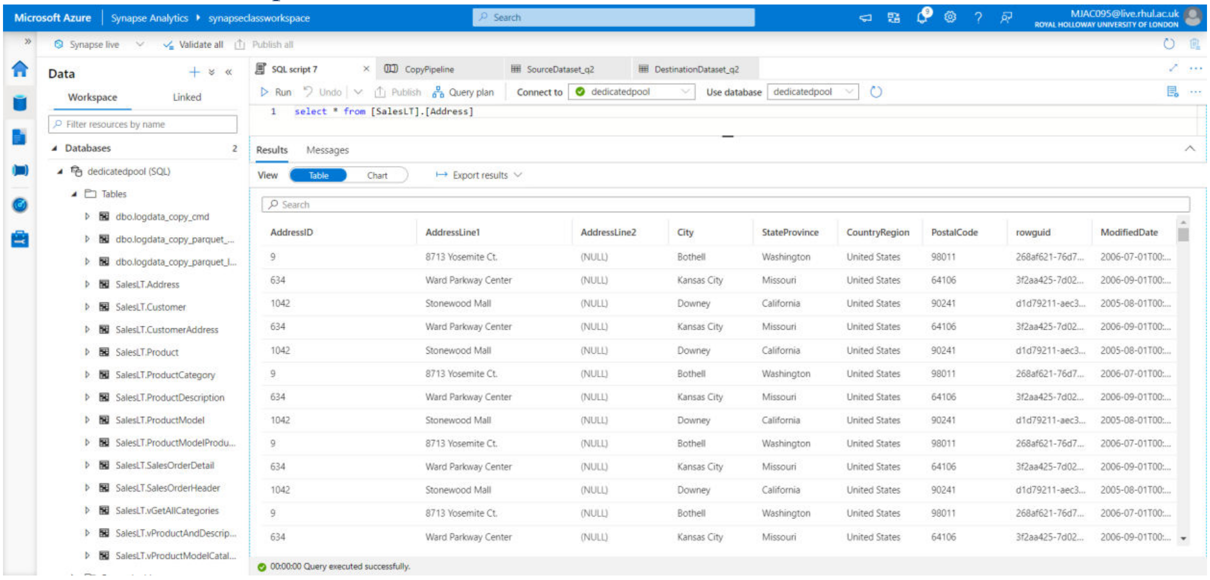
- In the below Screenshot below, we could see all the Tables are copied into the Tables,

Screenshot 2.9: Displays Pipeline Debug Details



- Output verified: Using select query in any one of the tables and view the records

Screenshot 2.10: Output Verification



==========

Problem Statement 3: **Load data from Azure Blob storage to a database in Azure SQL Database by using Azure Data Factory**

==========

Solution:

- Created a Resource group, named as: Project-Arthi-rg
- Created a Blob Storage Account, named as: projectstorageacc01
- Created a data factory, named as: Project-Arthi-adf-01
- Created a self-hosted integration runtime named as: IR-1

as shown in Screenshot1.1.

- Created a Blob Storage and SQL table as **Source**

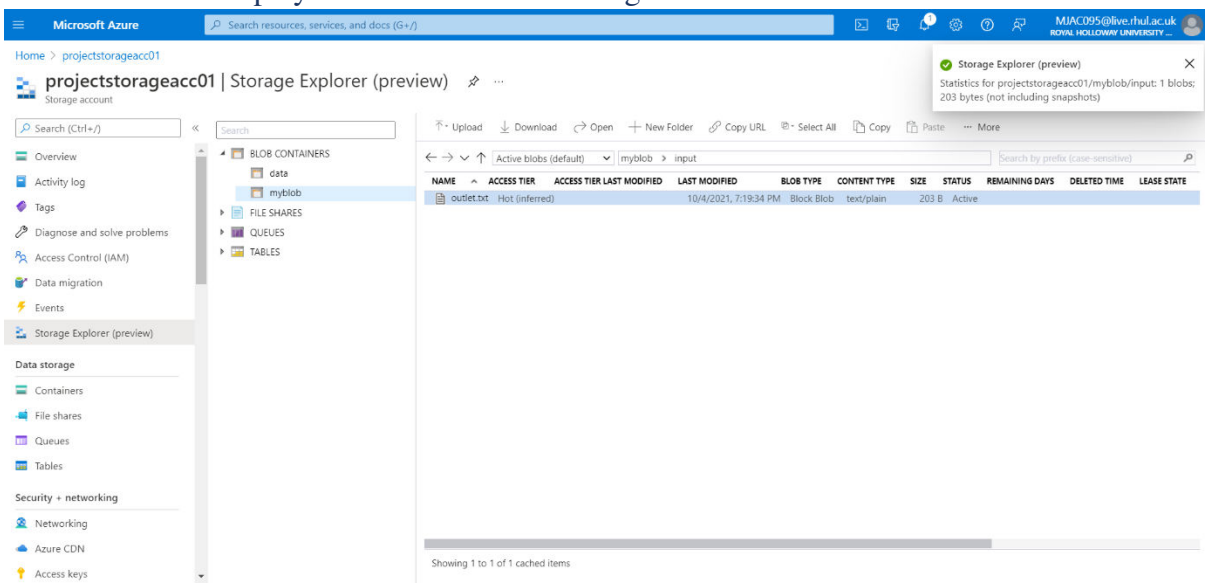
Created a Outel.txt file and uploaded in **myblob – container / input – folder / Outlet.txt**

TextFile :

Outlet.txt

```
outlet_name,address_street,address_city
Sklep Awziel,High Street,Rye
Mentfield,Crescent,Brooklyn
Natural Valley,kingsley Road,Pittsfield
Red Outlet,Senator,Brooklyn
Ads Nawziel,Mount Street,Woodside
```

Screenshot 3.1: Displays Text File in Blob Storage inside Container



- Created SQL table as **Sink, w**ith the following SQL script: **dbo.outlets**

Query: dbo.outlets

```
CREATE TABLE dbo.outlets(
    ID int IDENTITY(1,1) NOT NULL,
    outlet_name  varchar(50) NOT NULL,
    address_street  varchar(50),
    address_city  varchar(50),
    primary key(ID));
GO
 CREATE CLUSTERED INDEX IX_outlet_ID ON dbo.outlets (ID);
```

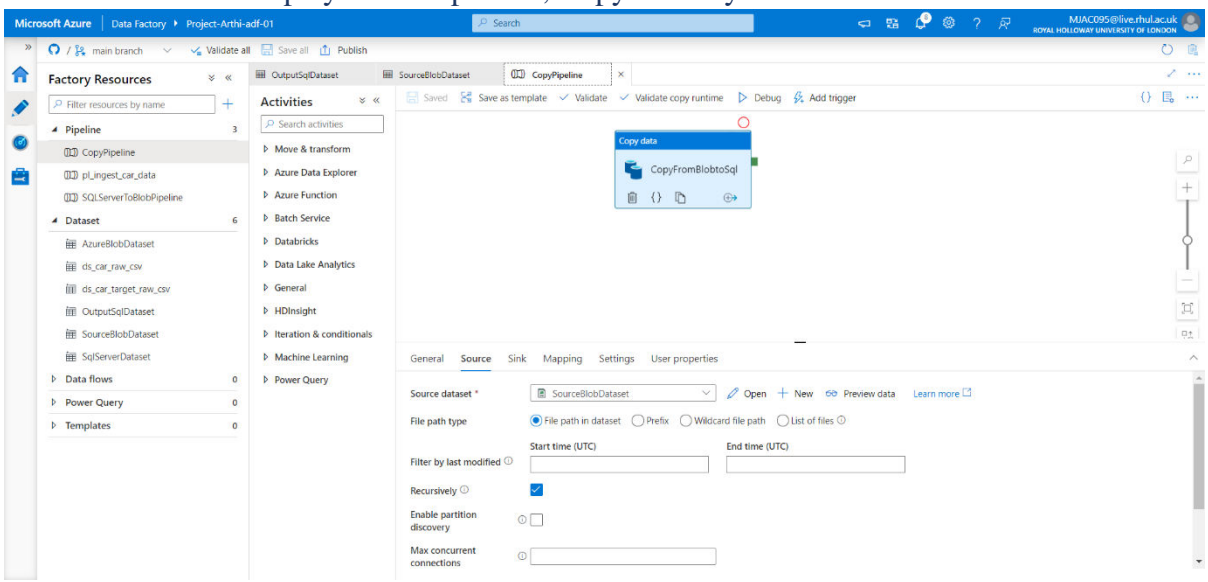
- In case of if any attribute is given unique, it creates integration violation, so we drop and truncate table and create a new one using above SQL script.

  DROP TABLE dbo.outlet1

  TRUNCATE TABLE dbo.outlet1

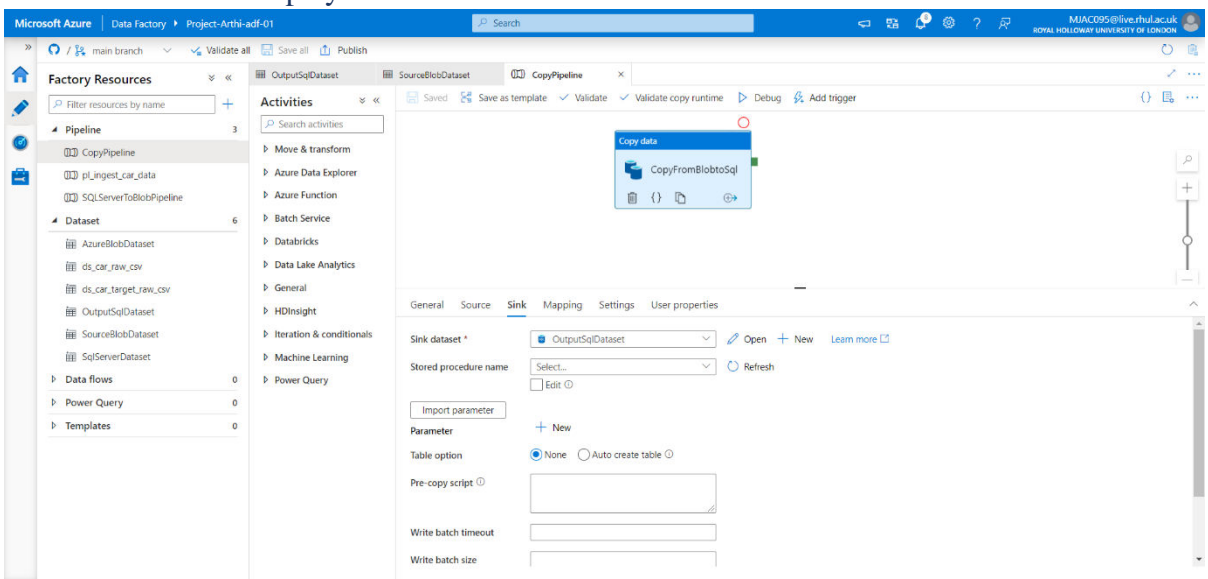
- Created a pipeline named CopyPipeline with a copy activity **“** CopyFromBlobToSql**”** to move the data from blob storage to Azure SQL database. Then set up Source dataset: SourceBlobDataset with linked service AzureStorageLinkedService

Screenshot 3.2.a: Displays both Pipelines, Copy Activity and Source Dataset



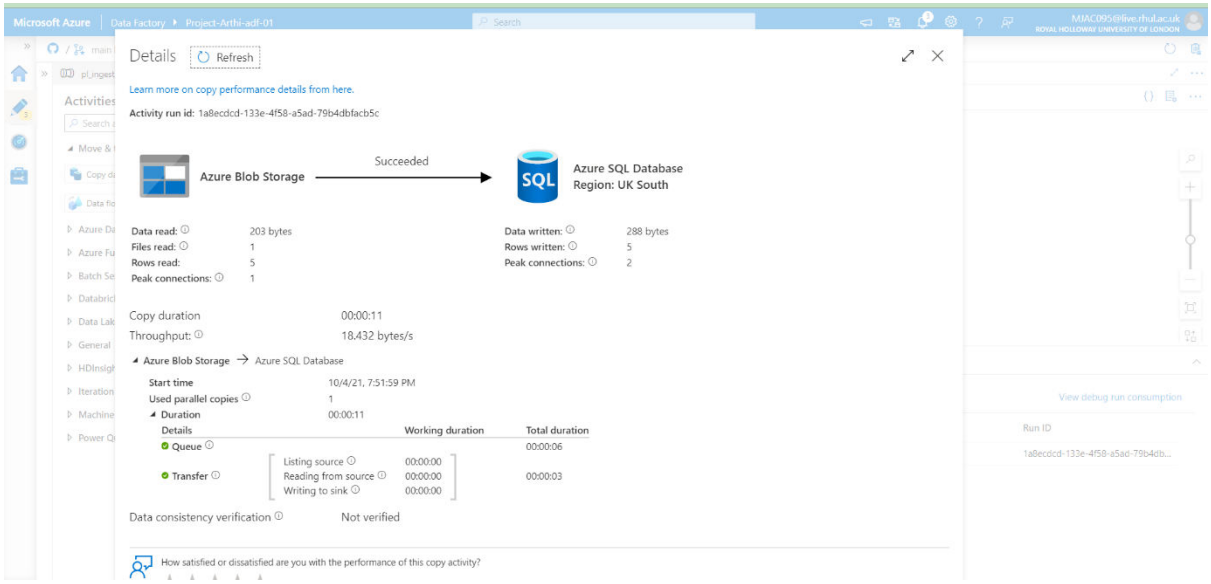
- Sink dataset as: OutputSqlDataset (Azure SQL database) **,** with linked service AzureSqlDatabaseLinkedService

Screenshot 3.2.b: Displays Sink Dataset

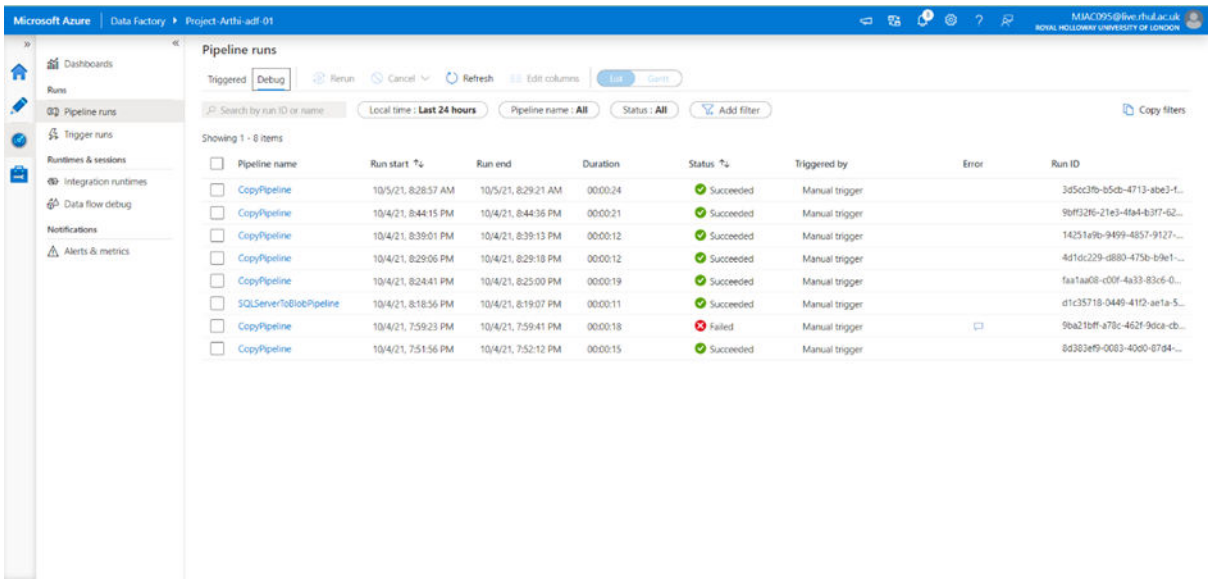


- Started a pipeline run, debugged, pipeline validation done. Published All

Screenshot 3.3.a: Pipeline Activity Detail

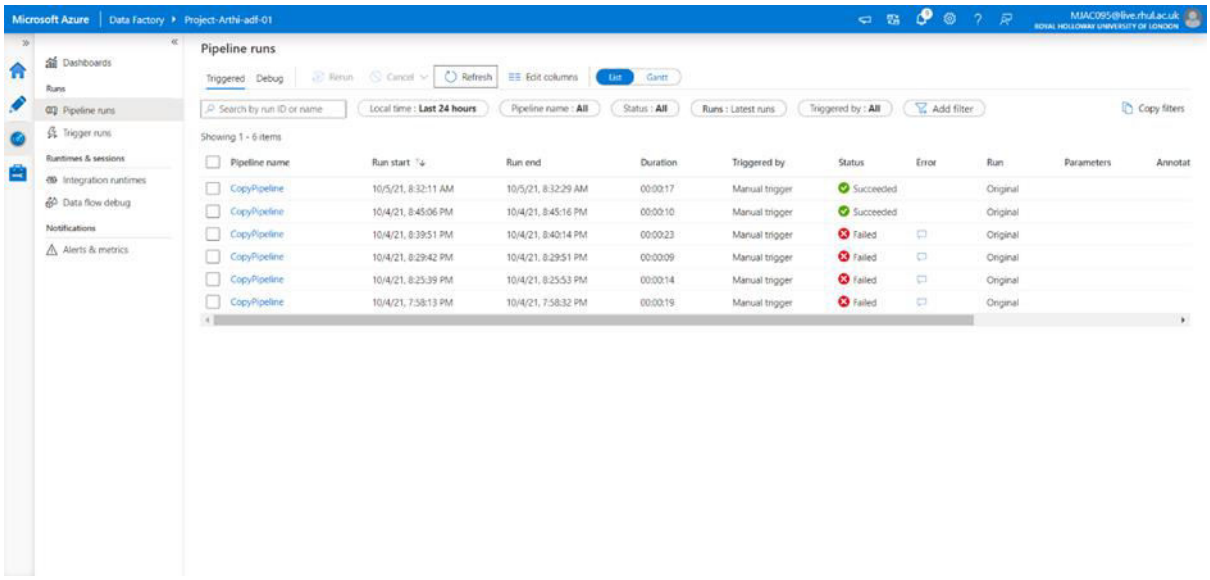


Screenshot 3.3.b: Pipeline Debug Detail



- Added trigger by clicking on trigger now
- Monitoring the relative pipeline CopyPipeline

Screenshot 3.3.c: Pipeline Manual Trigger detail

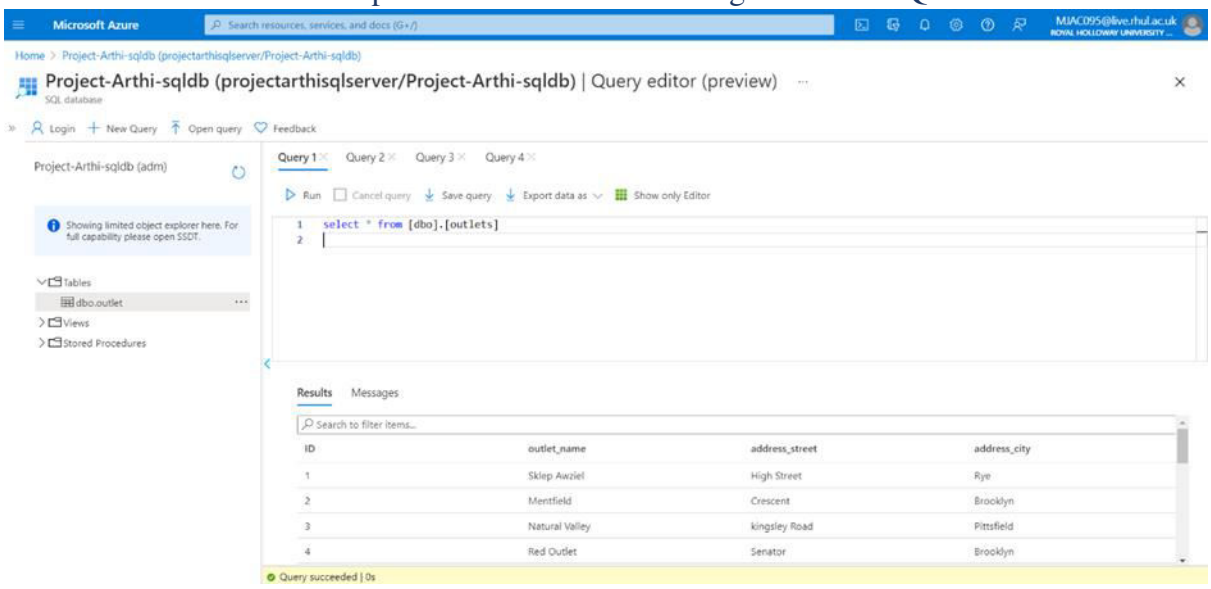


- Output verified: Text File copied from Azure blob storage to a Table format in Azure SQL database.
- Verified using SQL SCRIPT:

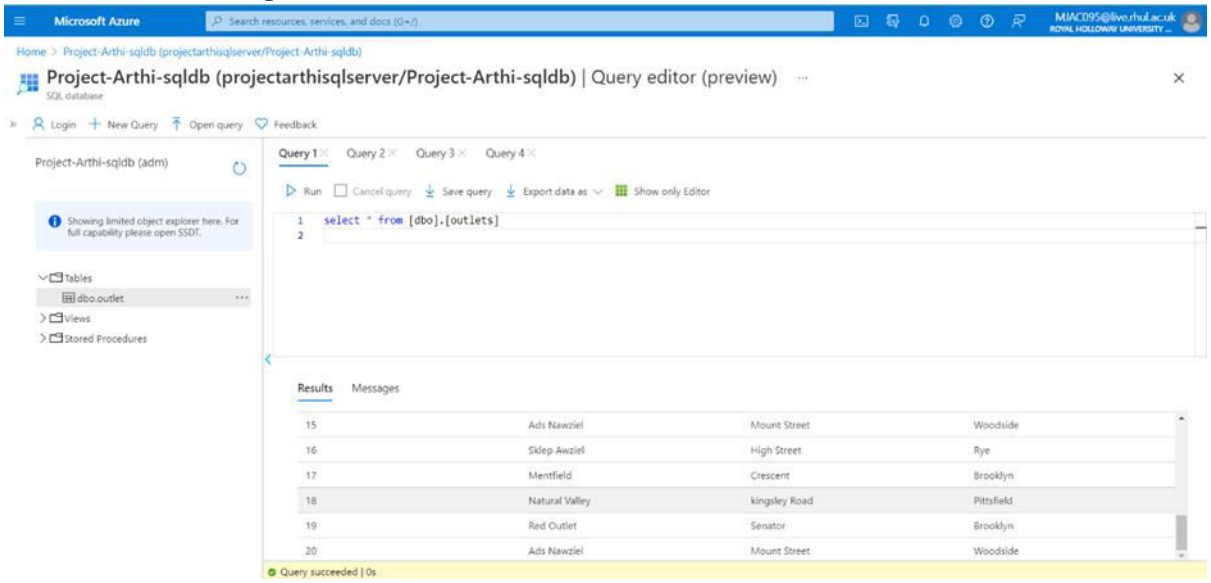
SELECT * FROM [dbo].[outlets]

- When the copy activity performed for first time, text file 5 rows copied into table as first row as header and next 5 rows as 5 records.
- When we triggered, text is copied into table, therefore next 5 recodes are created in the table with ID 6-10.

Screenshot 3.4: Text File copied from Azure Blob Storage to Azure SQL Database



Screenshot 3.5: Output Verification



==========

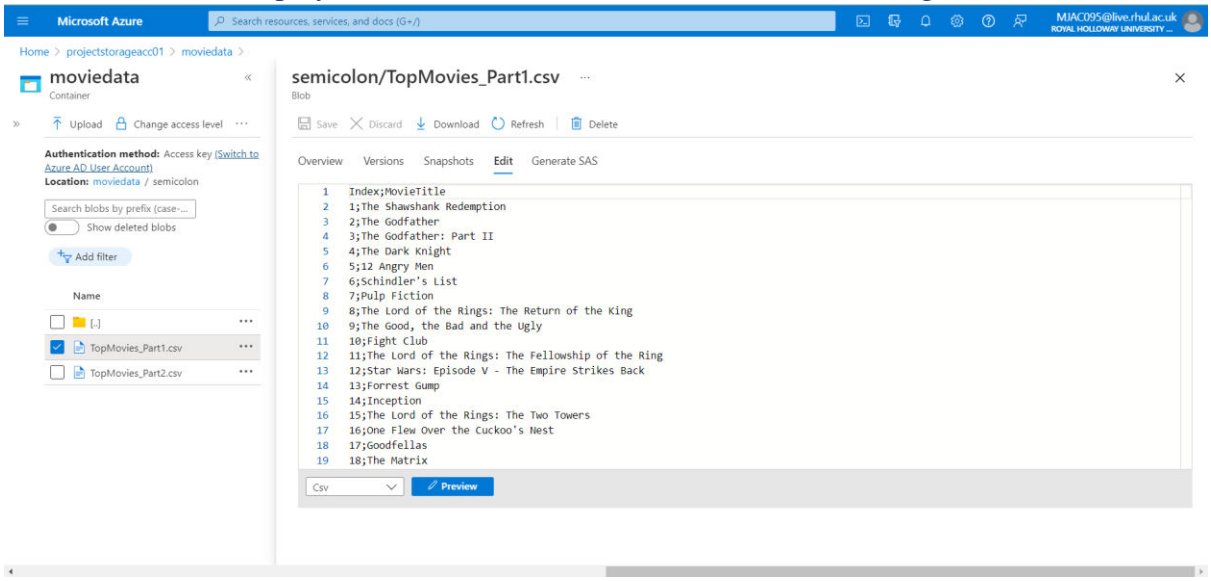
Problem Statement 4: **How to Load Multiple Files in Parallel in Azure Data Factory**

==========

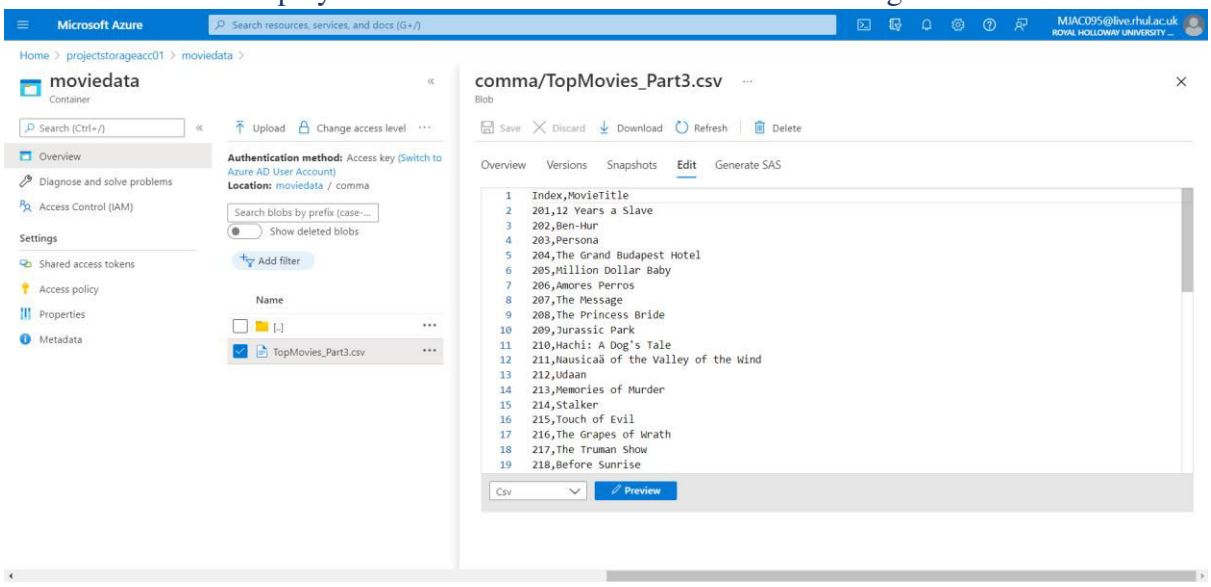
Solution :

- Created a Resource group, named as: Project-Arthi-rg
- Created a Blob Storage Account, named as: projectstorageacc01
- Created a data factory, named as: Project-Arthi-adf-01
- Created a self-hosted integration runtime named as: IR-1 as shown in Screenshot 1.
- My Source file contains 3 Top movie name files, 2 semicolon Delimited with 100 values, 1 comma delimited with 50 values.
- These files uploaded in the Blob storage account
    - projectstorageacc01
        - /Container: moviedata
            - /Folder: semicolon (2 files)
                - comma (1 files)

Screenshot 4.1.a: Displays the Semicolon Delimited Files in Blob Storage

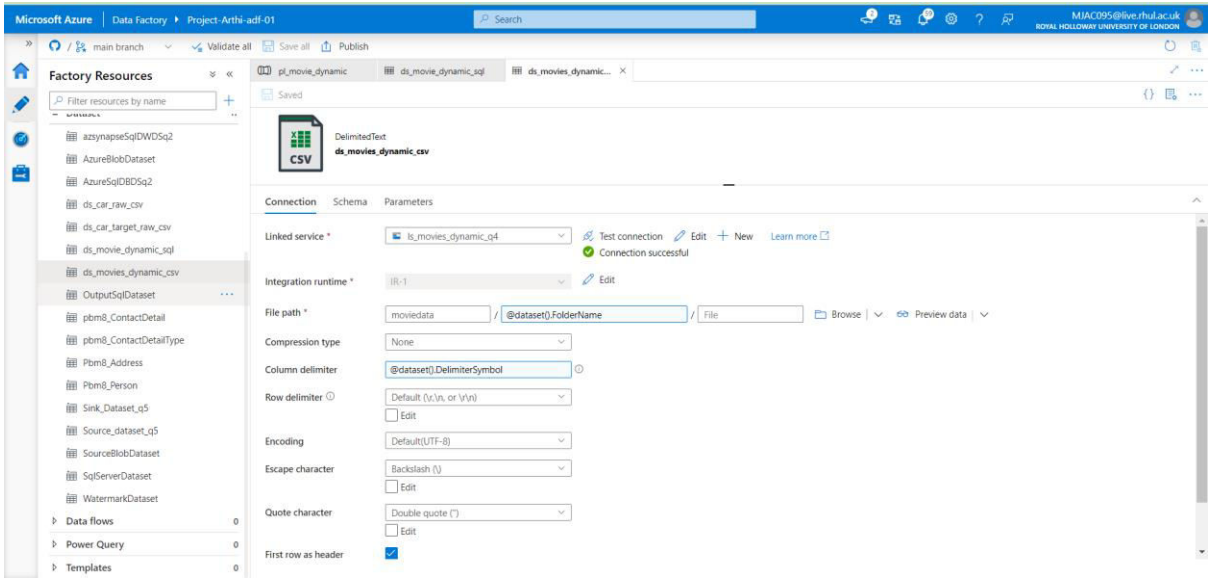


Screenshot 4.1.b: Displays the Comma Delimited Files in Blob Storage



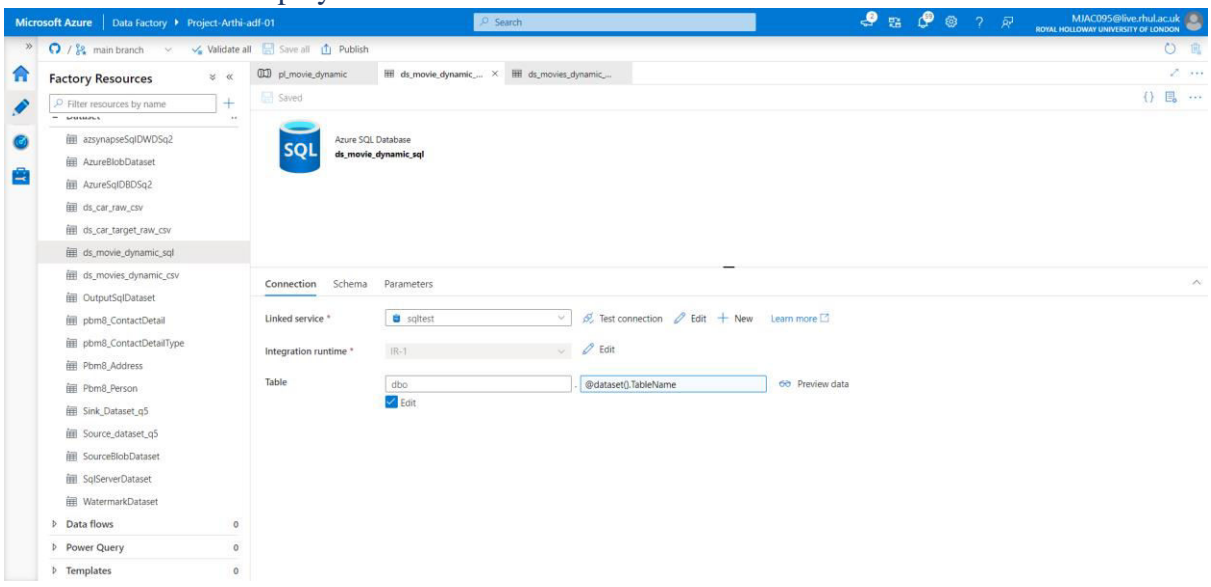
- Metadata for these files are stored in the
    - Azure SQL Database named: Project-Arthi-sqldb,
    - Azure SQL Server named as: projectarthisqlserver
    - Schema for the dbo.Metadata_ADF is stored in a table as key-value pair.
    - Insert queries of the Metadat_ADF, and Destination Table queries is attached with project in the Solution 4 folder.
- Created 2 datasets. One for blob storage and one for SQL Server.
- Source Dataset named as: ds_movies_dynamic_csv
    - Created Linked Service: ls_movies_dynamic_q4
    - Need to add Parameters as FolderName and Delimiter symbol
    - Set the file path:
        - movidata/@dataset()Foldername/File
    - Set the Column delimiter:
        - check edit check box,
        - set: @dataset().DelimiterSymbol

Screenshot 4.2.a: Displays the Source Dataset

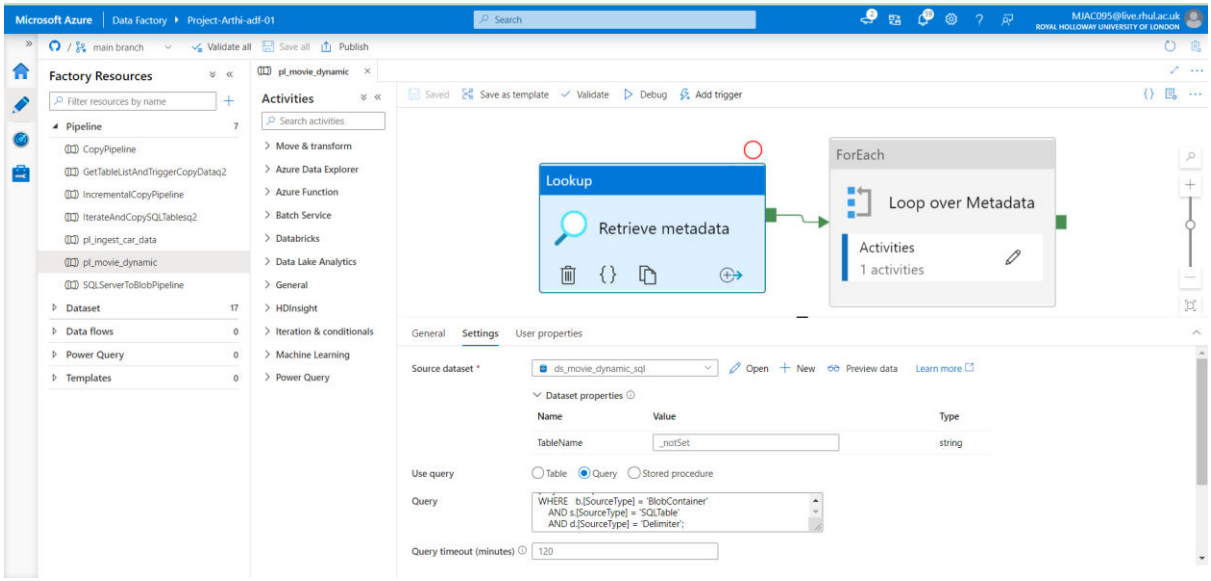


- Sink Dataset named as: ds_movies_dynamic_sql
    - Created linked Service: SQLtest
    - Need to add Parameters as TableName
    - Set theTable:
        - dbo/@dataset()TableName

Screenshot 4.2.b: Displays the Sink Dataset



Screenshot 4.3.a: Displays the Pipeline, Lookup Activity Details

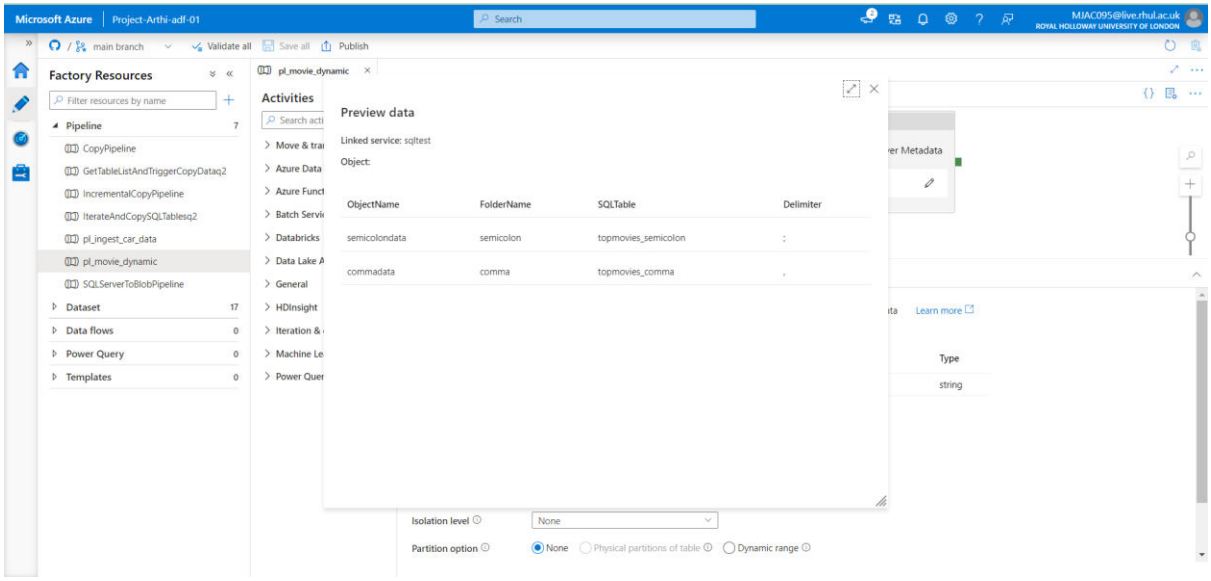


- We connect to the source dataset, set tablename as "_notSet"
- We Use query and deselect the First row only checkbox
- With the following query, we can retrieve the metadata from SQL Server:

Query:

```
SELECT
    b.[ObjectName]
   ,FolderName = b.[ObjectValue]
   ,SQLTable   = s.[ObjectValue]
   ,Delimiter  = d.[ObjectValue]
FROM [dbo].[Metadata_ADF] b
JOIN [dbo].[Metadata_ADF] s ON b.[ObjectName] = s.[ObjectName]
JOIN [dbo].[Metadata_ADF] d ON b.[ObjectName] = d.[ObjectName]
WHERE   b.[SourceType] = 'BlobContainer'
   AND s.[SourceType] = 'SQLTable'
   AND d.[SourceType] = 'Delimiter';
```

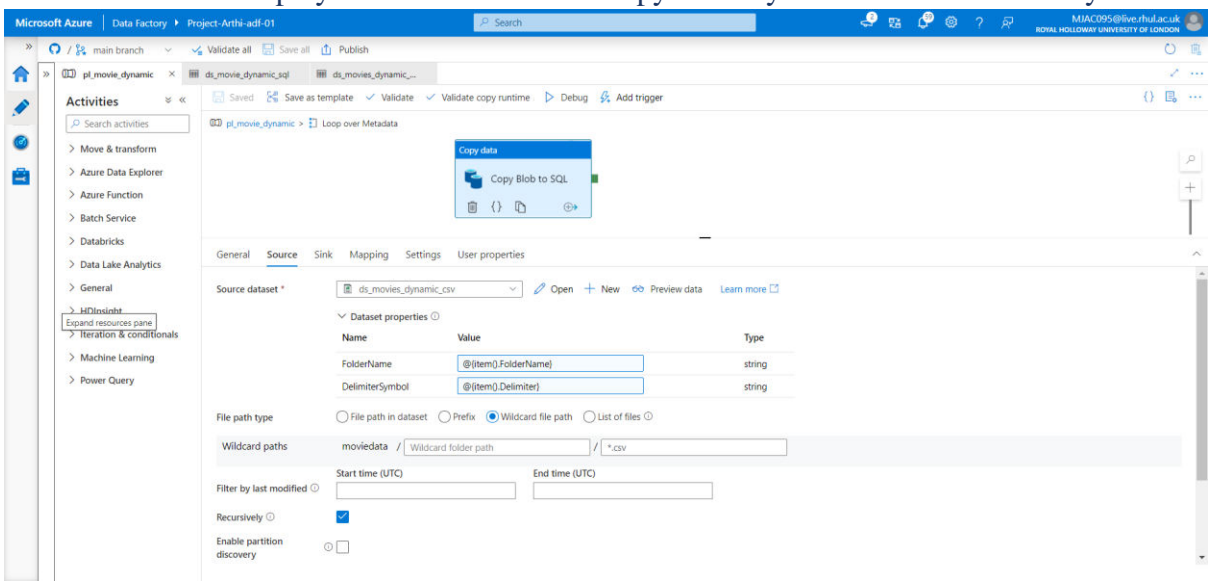
Screenshot 4.3.b: Displays the Preview Data of the Lookup Activity



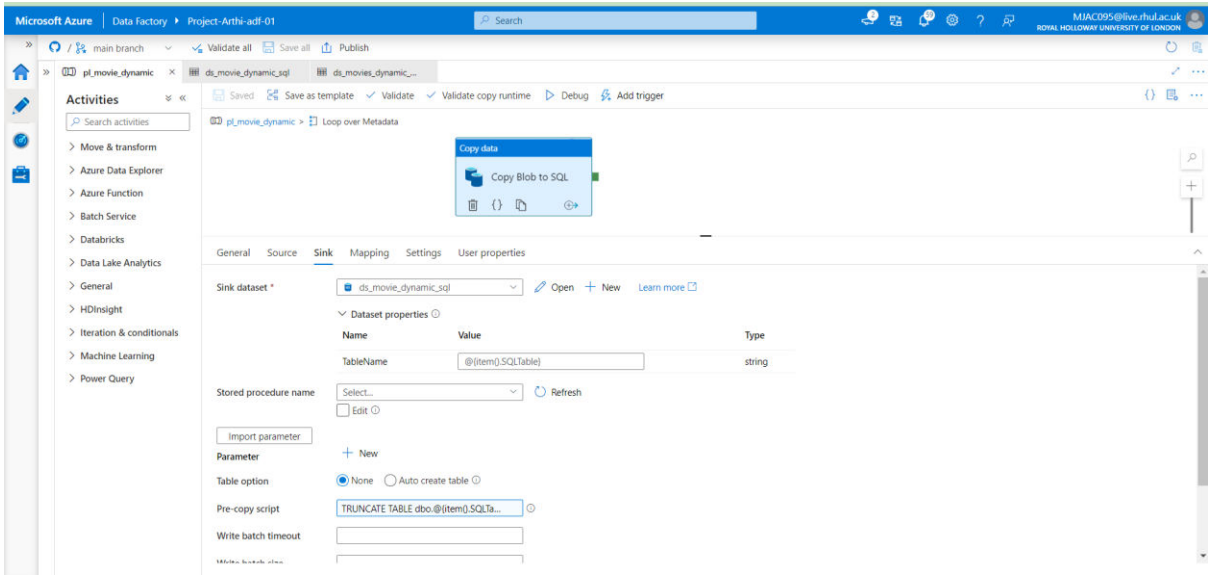
- Next we add ForEach Activity:
    - In Setting tab: set items as : " @activity('Retrieve Metadata').output.value " as dynamic content
    - In Activities tab : Add Copy Activity : Copy Blob to SQl
        - Set Source dataset : ds_movies_dynamic_csv
            - Fill the dataset Properties as @(item().FolderName}, @(item().Delimeter}
            - Choose WildcardPath, set wildcard filename as " *.csv"
        - Set Sink dataset: ds_movies_dynamic_sql
            - Fill the dataset Properties as @(item().TableName}
            - Table Option: None
            - Pre-Copy script:

TRUNCATE TABLE dbo.@{item().SQLTable}

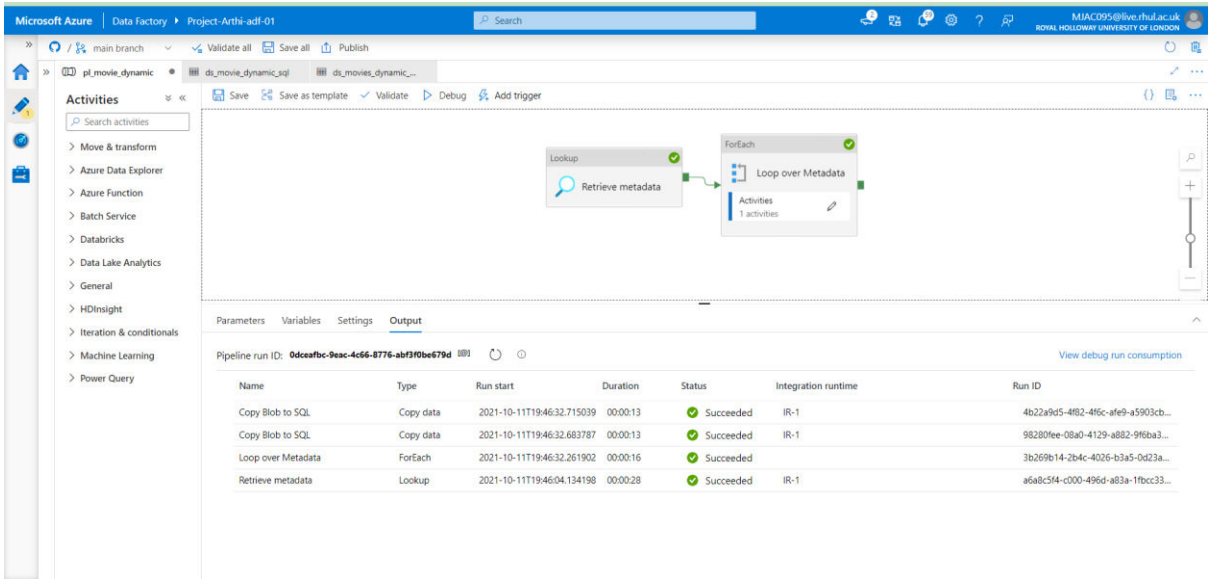
Screenshot 4.3.c: Displays the Source Dataset Copy Activity within ForEach Activity



Screenshot 4.3.d: Displays the Sink Dataset Copy Activity within ForEach Activity

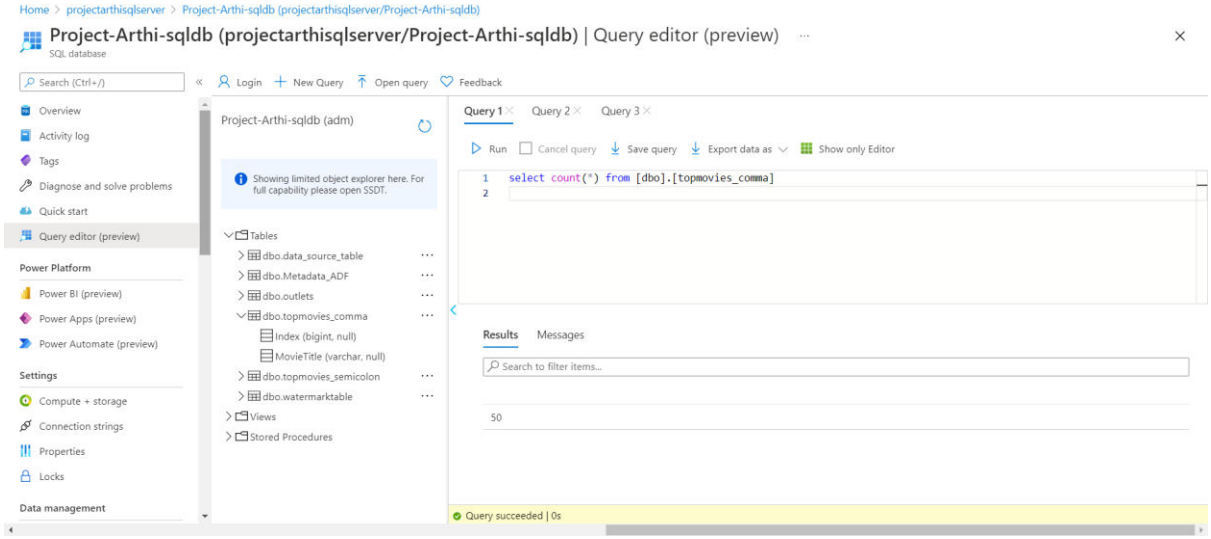


Screenshot 4.4: Pipeline Debug Details

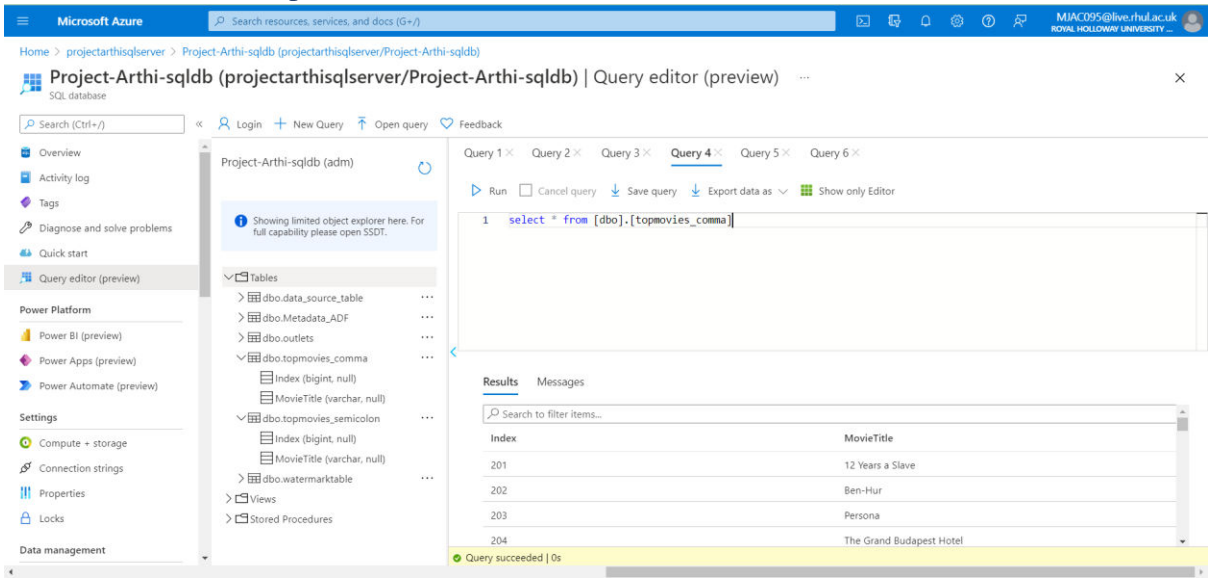


- Output verified: Using SQL Query in the Azure SQL Database, checking the total number of rows in each table and viewing the records.

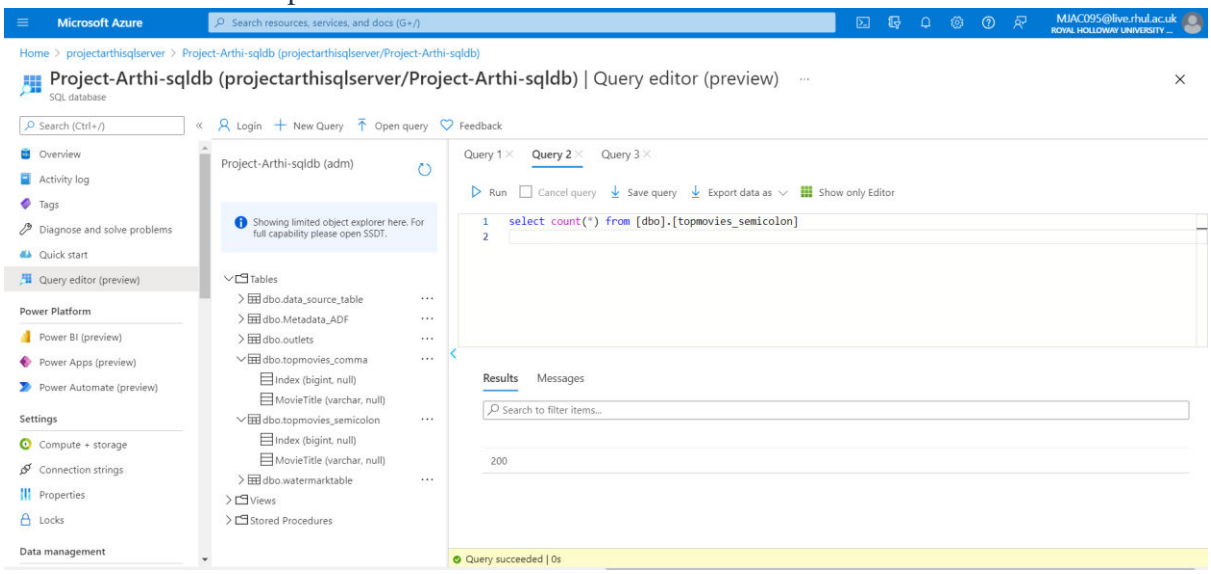
Screenshot 4.5.a: Output Verification I



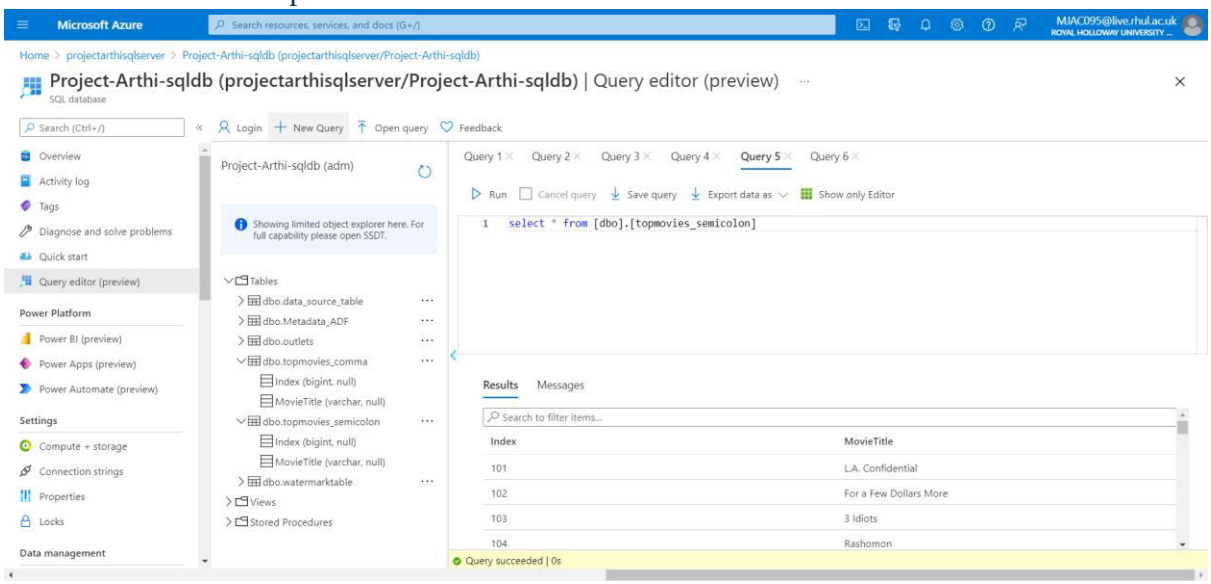
Screenshot 4.5.b: Output Verification II



Screenshot 4.5.c: Output Verification III



Screenshot 4.5.d: Output Verification IV



==========

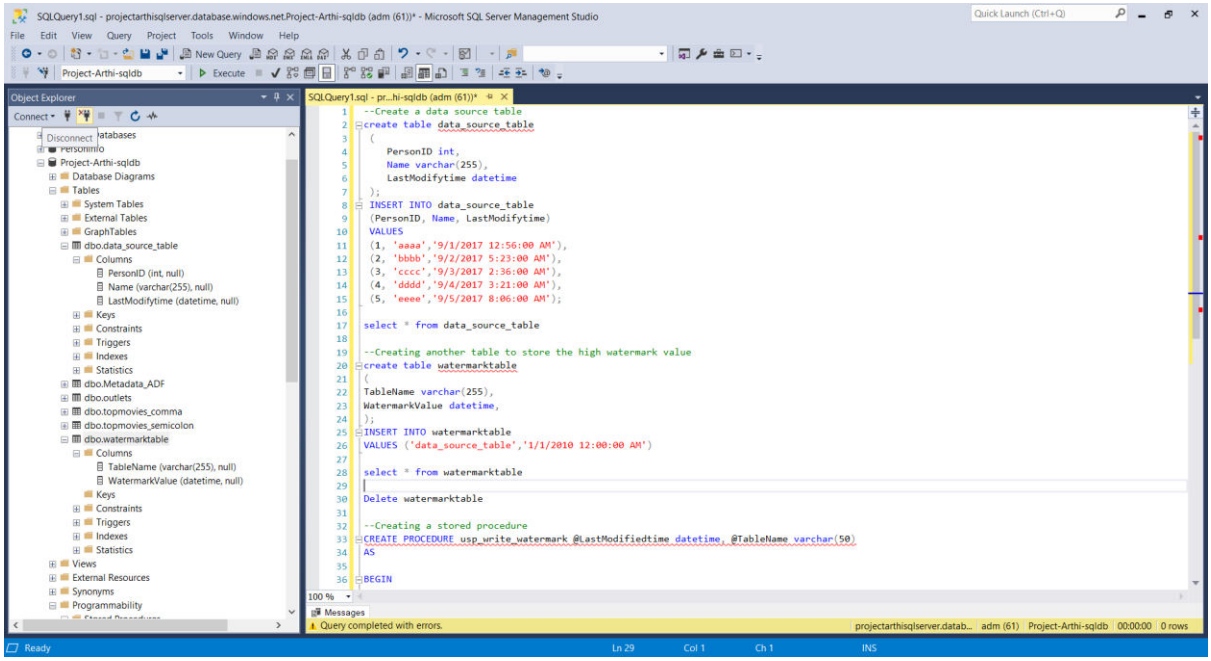
Problem Statement 5: **Incrementally load data from Azure SQL Database to Azure Blob storage using the Azure portal**

==========

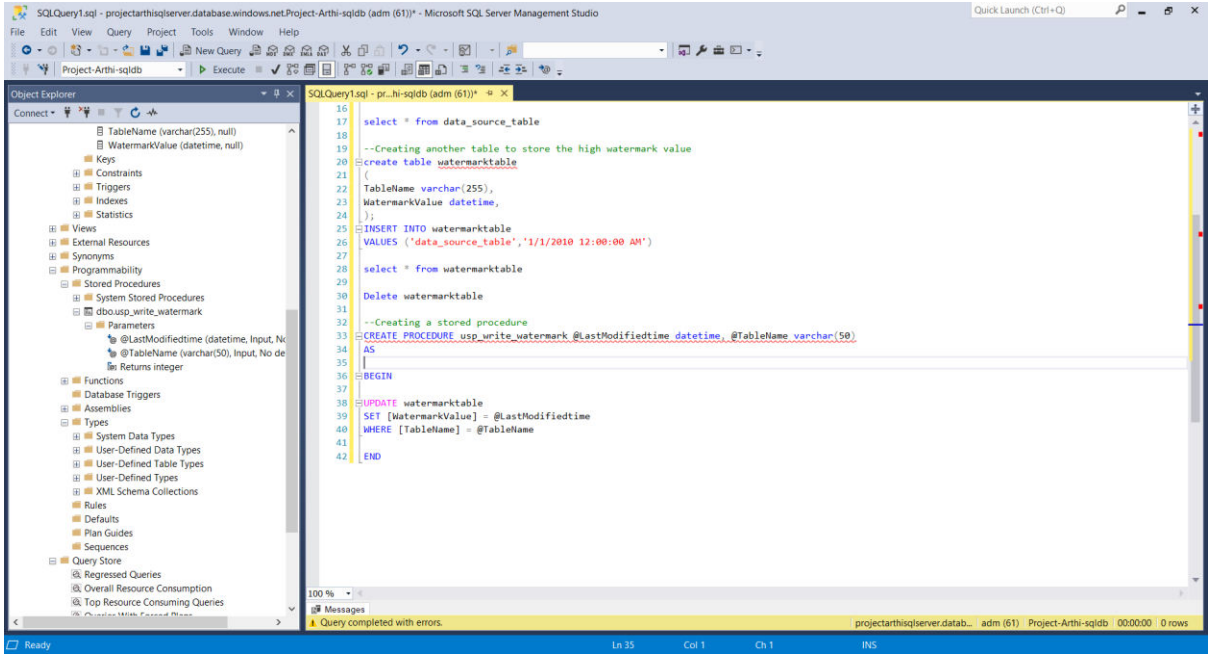
Solution:

- Created a Resource group, named as: Project-Arthi-rg
- Created a data factory, named as: Project-Arthi-adf-01
- Created a self-hosted integration runtime named as: IR-1, as shown in Screenshot1.1.
- Created 2 tables named **data_source_table** as the data source store, **watermarktable** to store the watermark value and a stored procedure named **usp_write_watermark.**

Screenshot 5.1.a: Displays table creation in SSMS

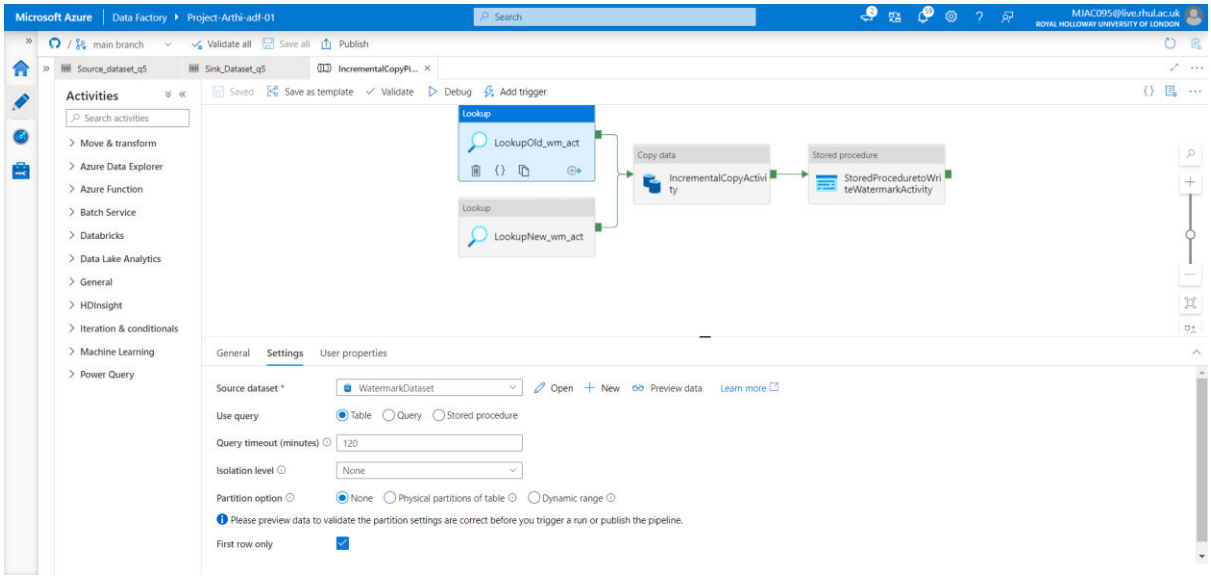


Screenshot 5.1.b: Displays Table Creation in SSMS:

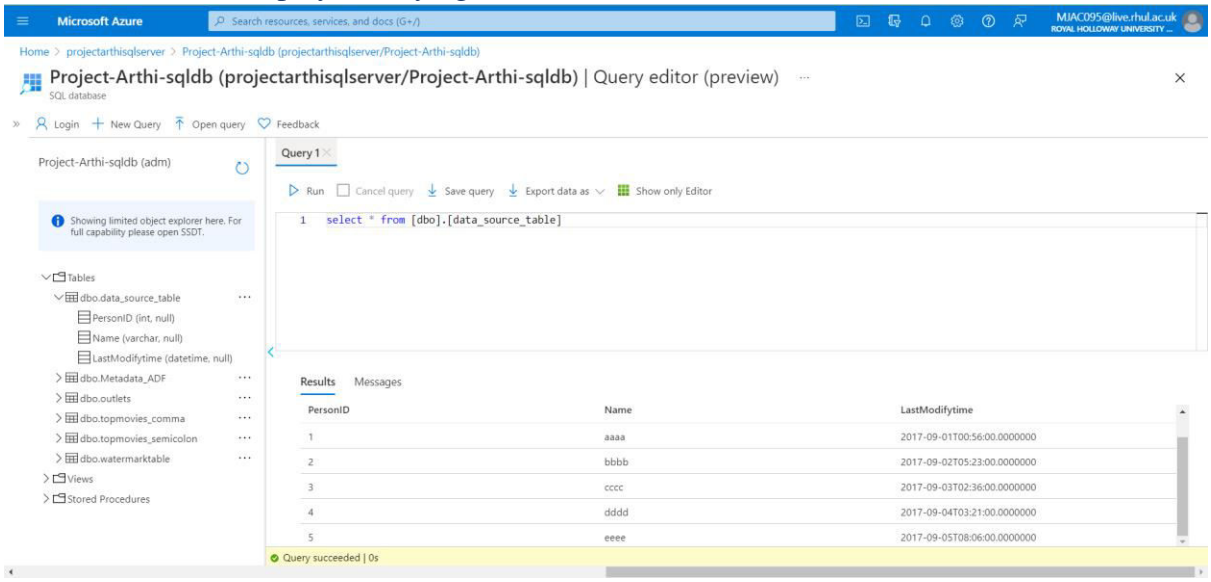


- Created a pipeline named IncrementalCopyPipeline with a Lookup activity **"**LookupOld_wm_act**"** to hold the current watermark data from **watermarktable**. Then set up Source dataset: SourceDataset_q5 with linked service "ls_azsqldb_q5" with data_source_table.

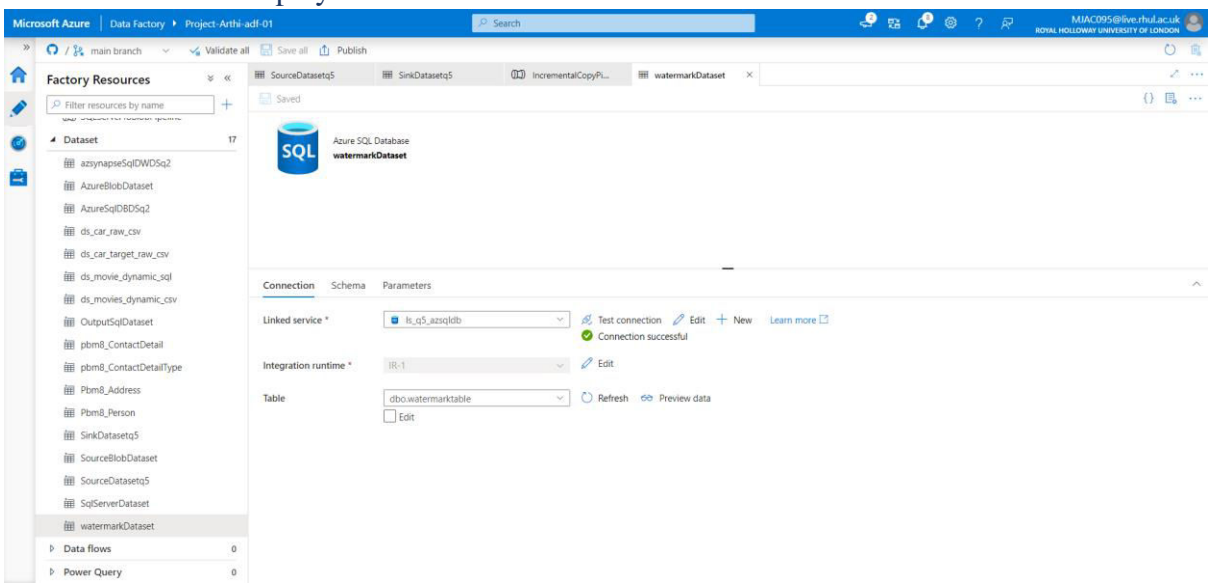
Screenshot 5.2.a: Displays both pipeline, Old Water Mark LookUp Activity and Watermark Dataset



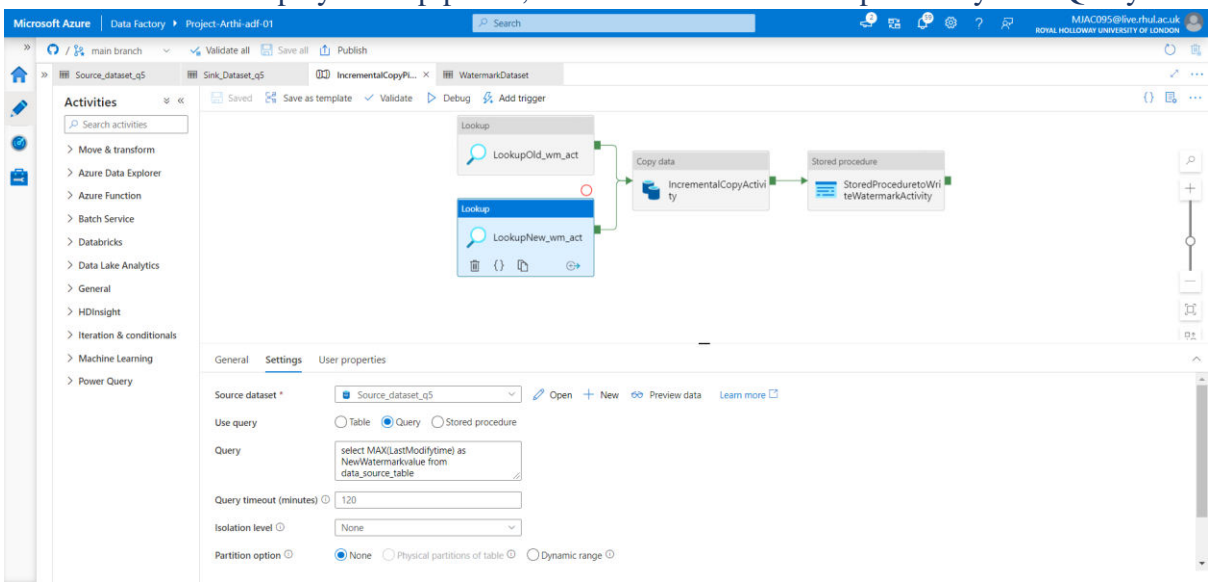
Screenshot 5.2.b: Displays verifying the Data Source Table in Azure SQL Database



Screenshot 5.2.c: Displays Linked Services for Watermark Dataset



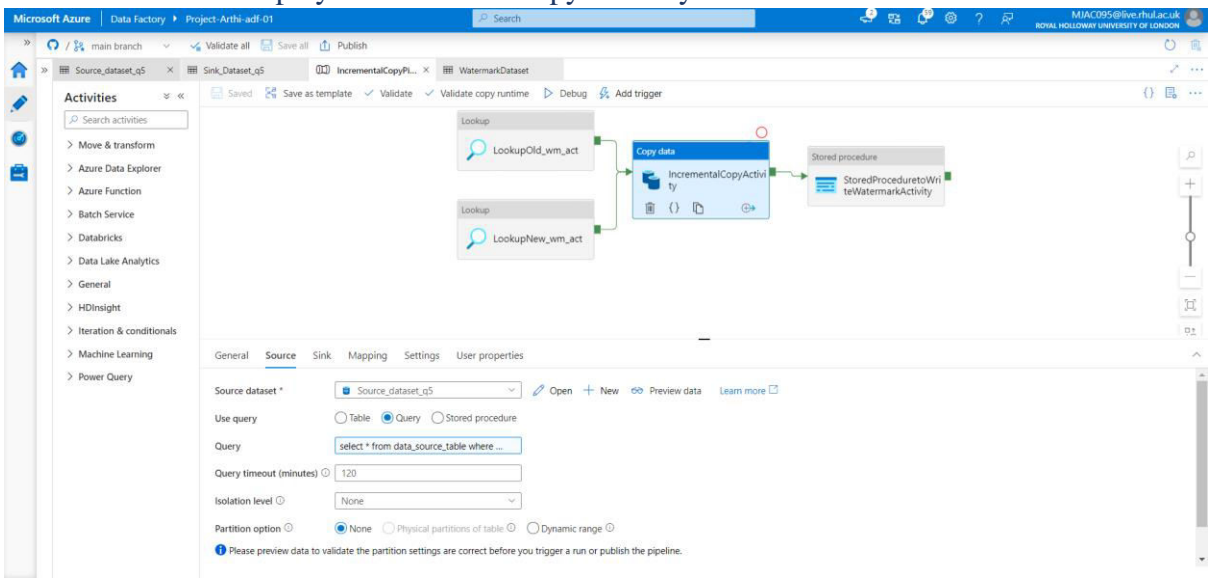
Screenshot 5.2.d: Displays both pipeline, New Water Mark LookUp Activity with Query



Query:

select MAX(LastModifytime) as NewWatermarkvalue from data_source_table

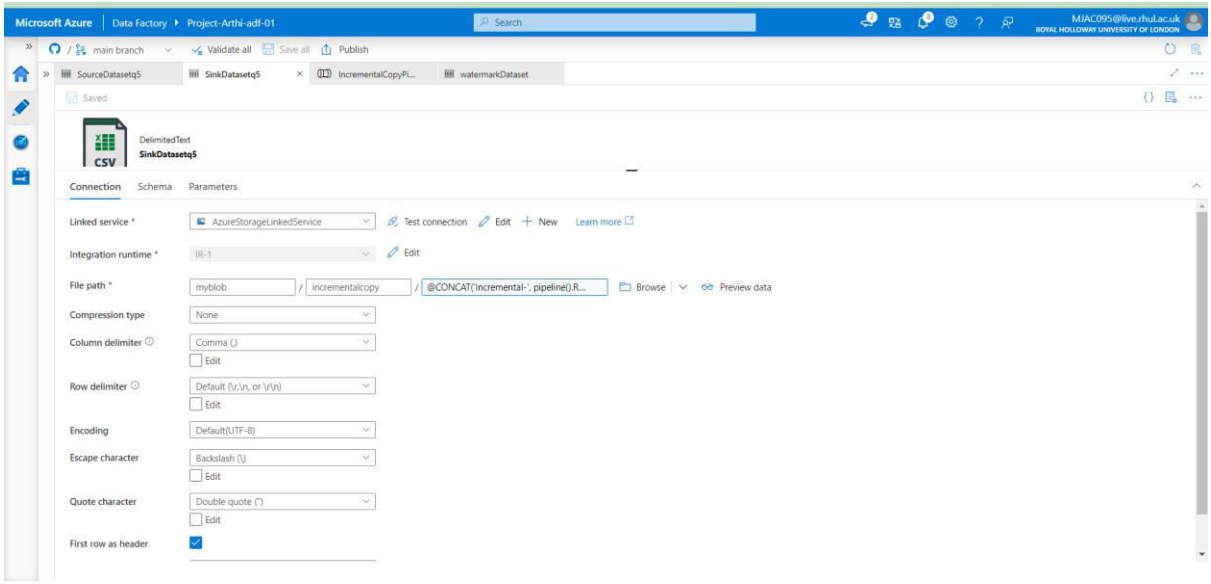
Screenshot 5.2.e: Displays Incremental Copy Activity and Source Dataset Details



Query:

select * from data_source_table where LastModifytime > '@{activity('LookupOldWaterMarkActivity').output.firstRow.WatermarkValue}' and LastModifytime <= '@{activity('LookupNewWaterMarkActivity').output.firstRow.NewWatermarkvalue}'

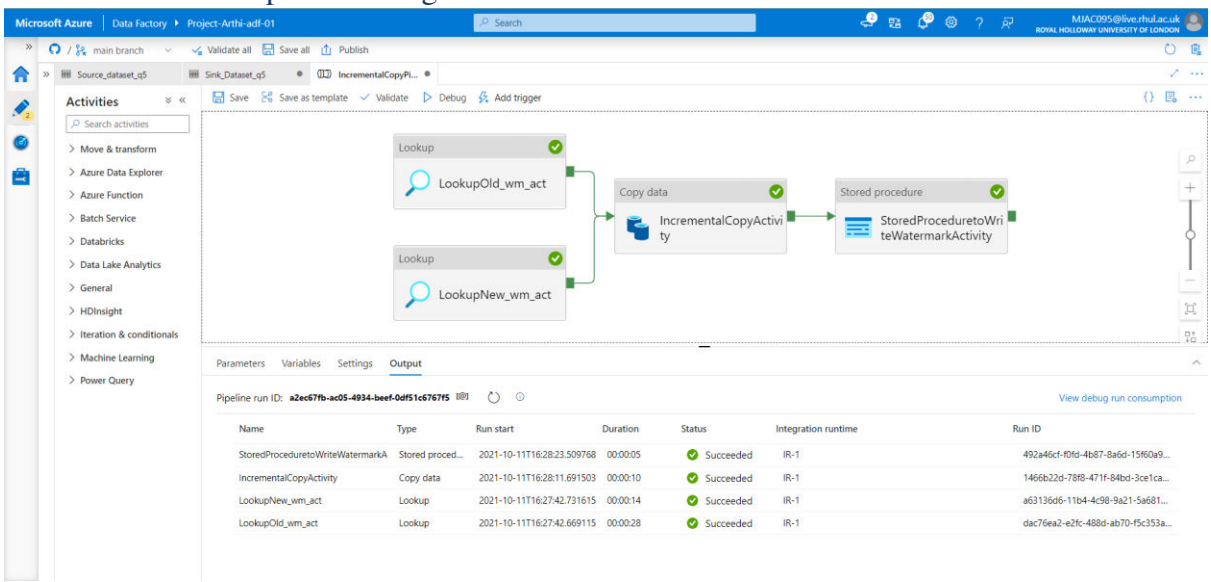
Screenshot 5.2.f: Displays Sink Dataset Details



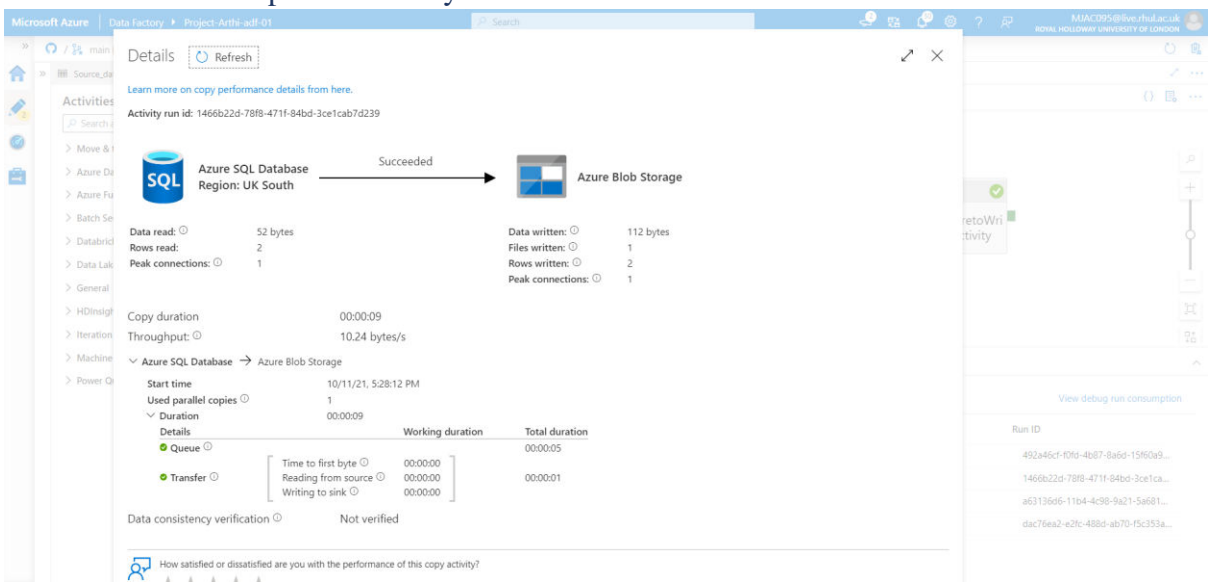
- Azure Data Factory automatically creates the output folder **incrementalcopy** if it does not exist. In the Screenshot, we have browsed the **File path** to navigate to a folder in a **blob container**. In the File path field, add the dynamic content :

@CONCAT ('Incremental-', pipeline (). RunId, '.txt')

Screenshot 5.3.a: Pipeline Debug Details

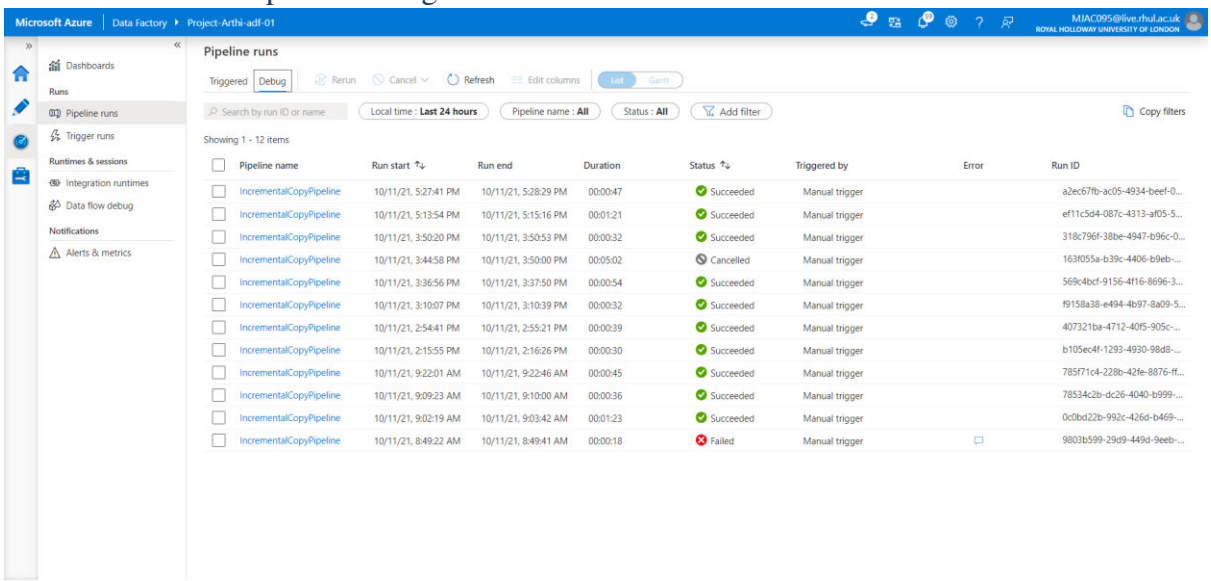


Screenshot 5.3.b: Pipeline Activity Details

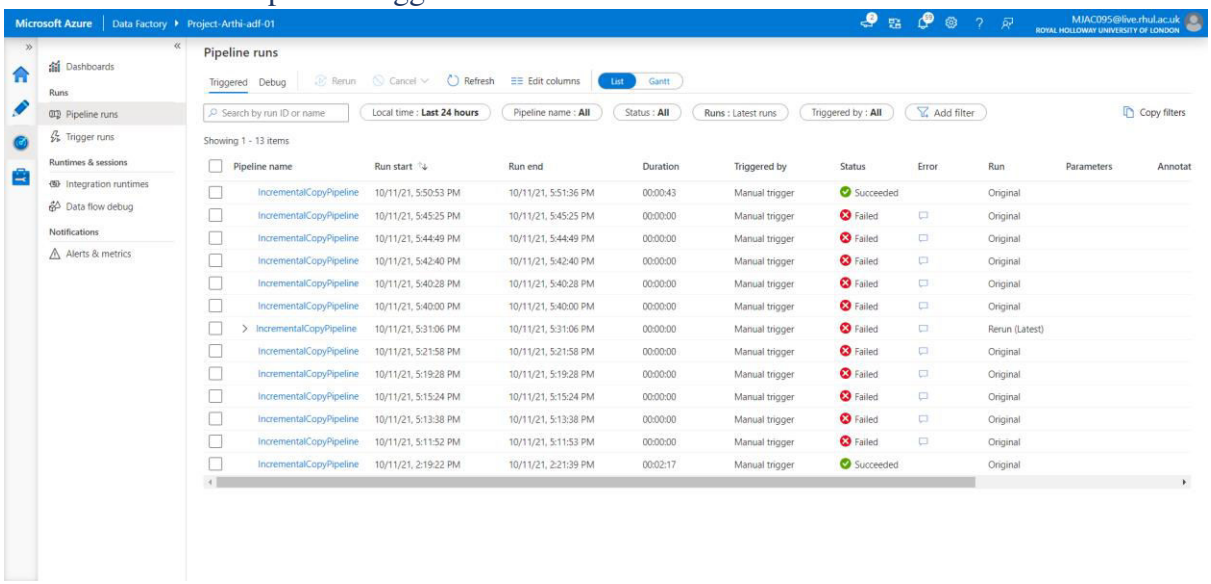


- Added trigger by clicking on trigger now

Screenshot 5.3.c: Pipeline Debug Detail II

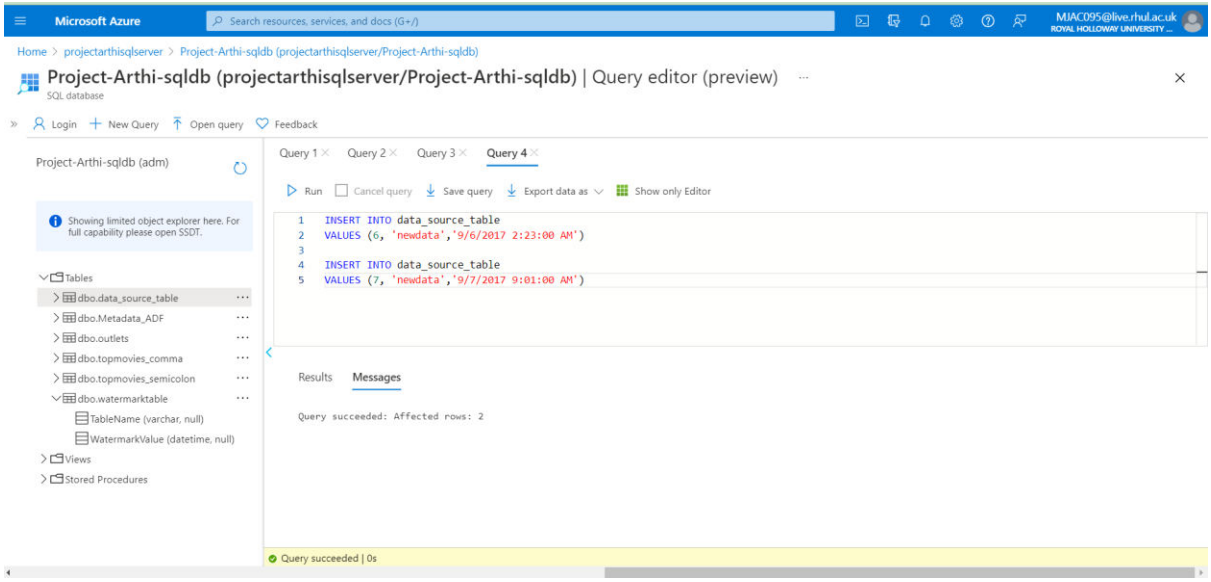


Screenshot 5.3.d: Pipeline Trigger Details

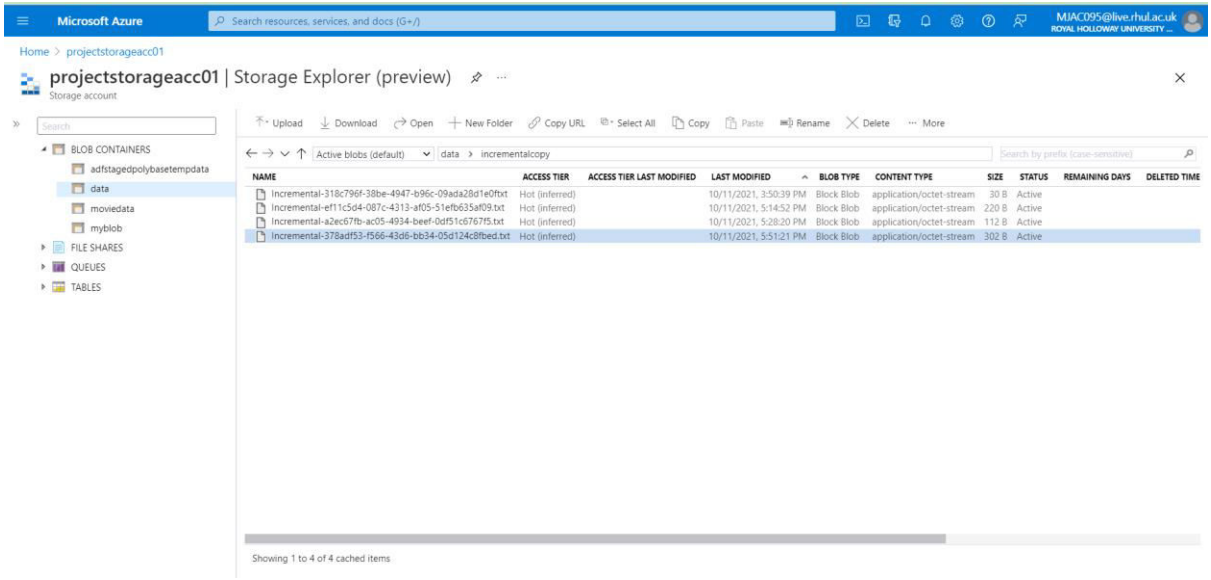


- Output verified: Table values in Azure SQL database copied to Text File in Azure blob storage
- When the copy activity performed for first time, text file 5 rows copied into text file as first row as header and next 5 records as 5 rows.
- New watermark is captured from the table using the query.
- New values 2 records are inserted into to the table
- When we triggered, records are copied into new text file, therefore updated 2 records are created in the table with index value 6,7.
- Again, New watermark is captured from the table using the query.

Screenshot 5.4: New records insertion in Data Source Table



Screenshot 5.5: Output Verification: Files generated after every run



Files: attached queries and output text files for this Problem in Solution5 Folder.

==========

Problem Statement 6: **Design a Relational Database Model for the below scenario in Azure SQL Database**

==========

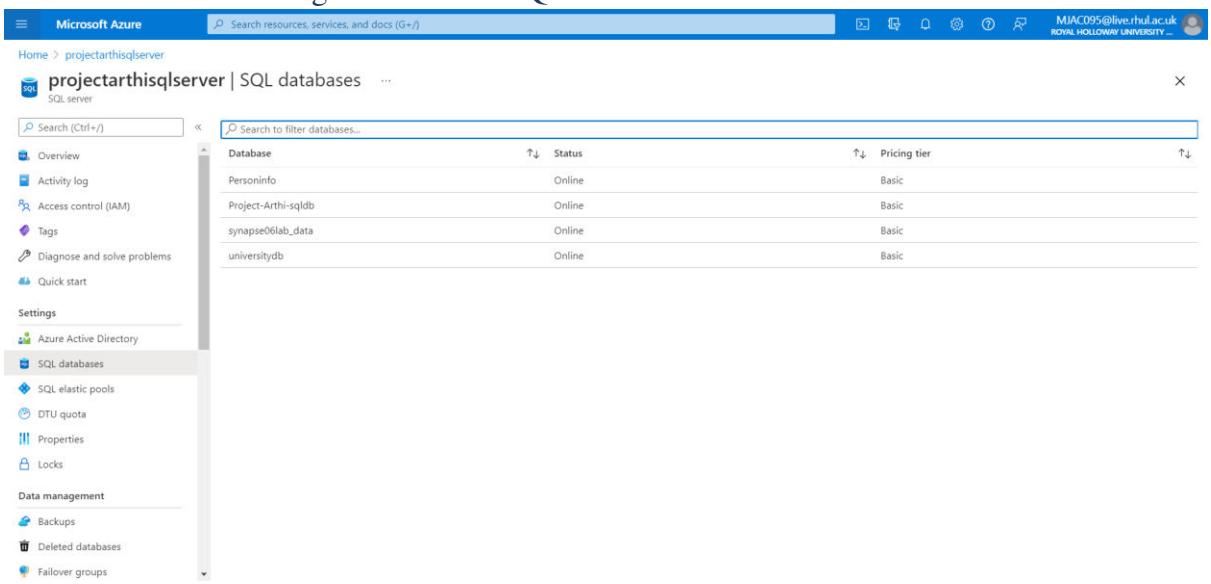
Solution :

- Created a Azure SQL Database named as: **universitydb** containing 4 Tables namely

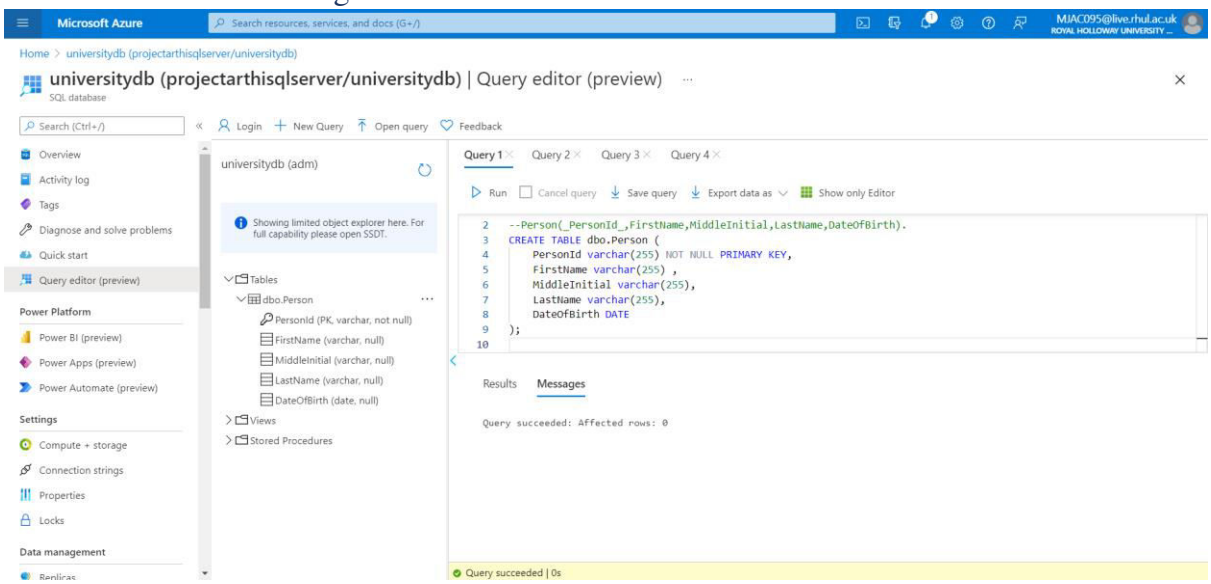
dbo.Person, dbo.Student,dbo.Course, dbo.Credit

- Created tables and relationship among them as given in the problem
- Created database and populated the table using insert query in each table using random values

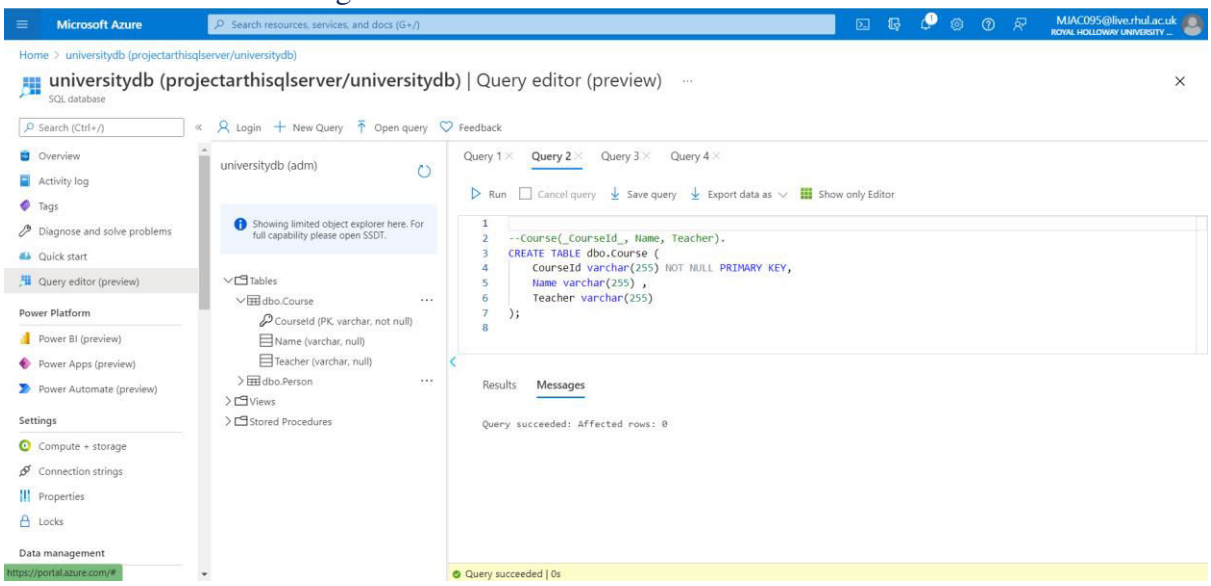
Screenshot 6.1: Creating DB in Azure SQL Database



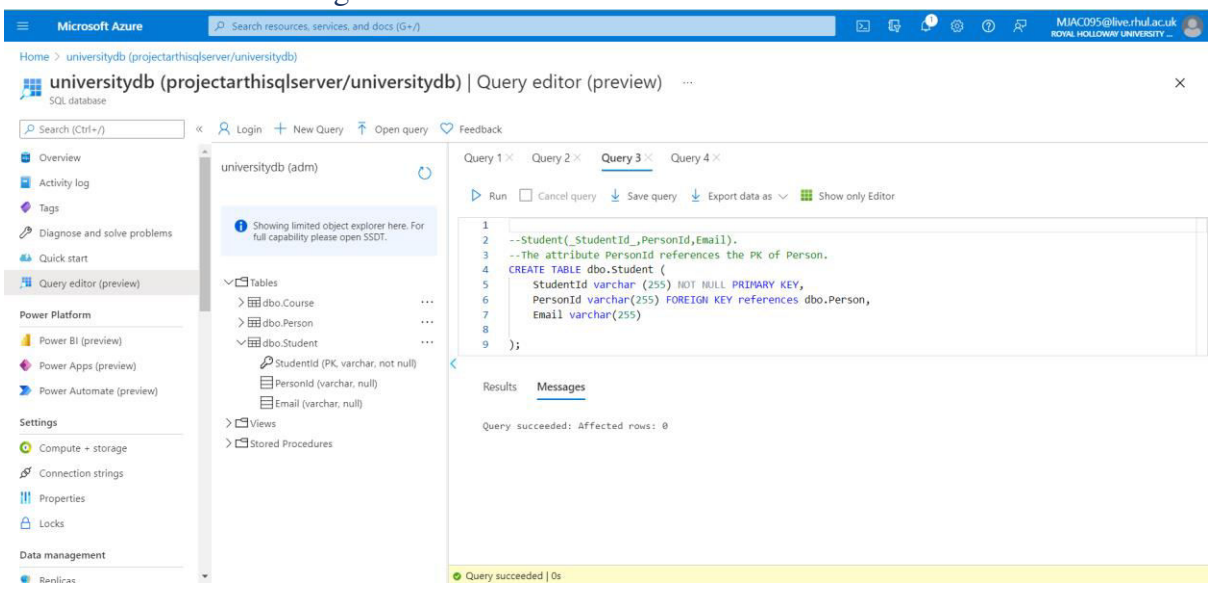
Screenshot 6.2.a: Creating Person Table



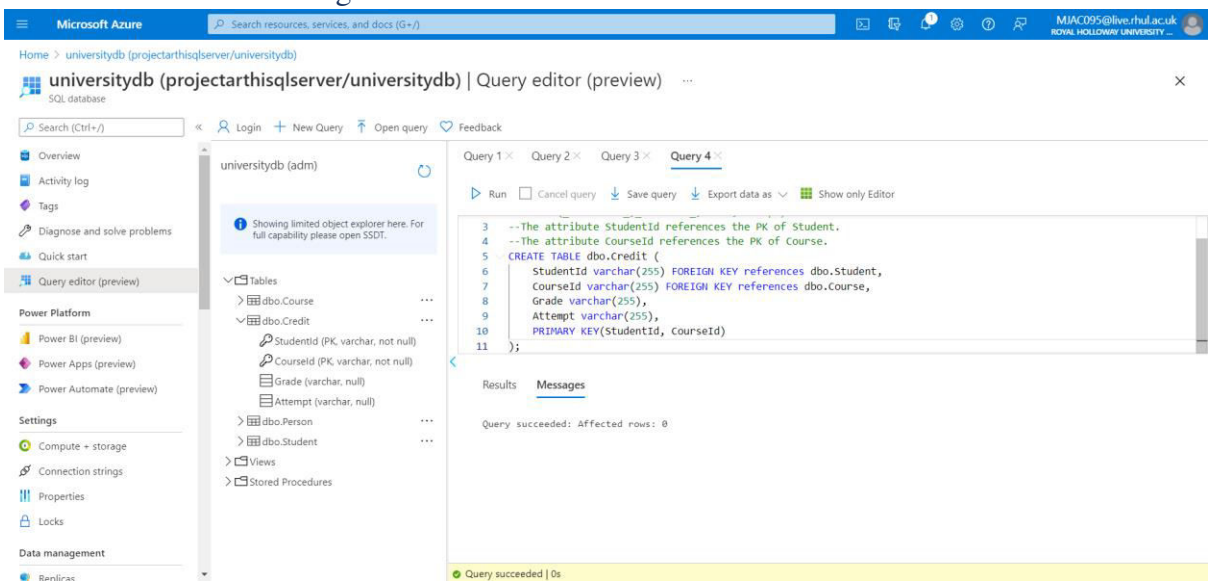
Screenshot 6.2.b: Creating Course Table



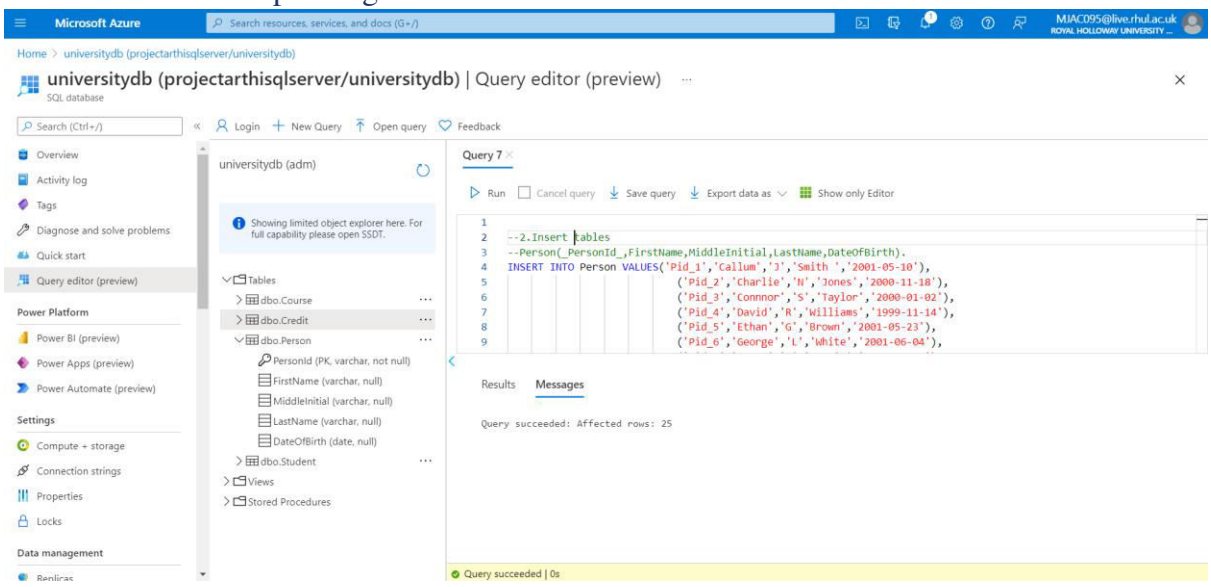
Screenshot 6.2.c: Creating Student Table



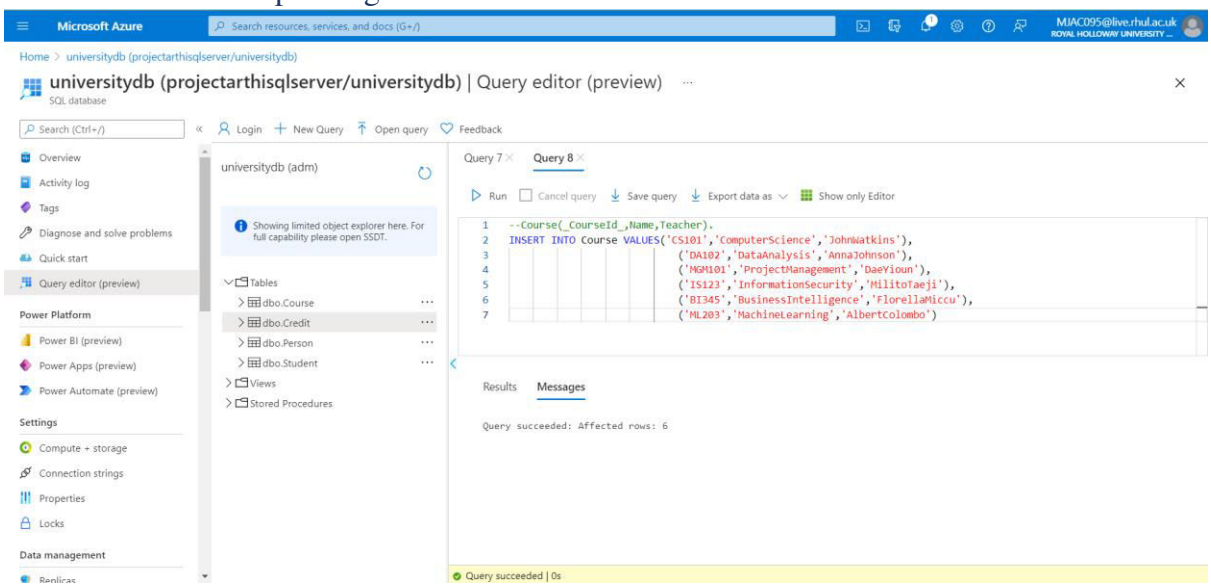
Screenshot 6.2.d: Creating Credit Table



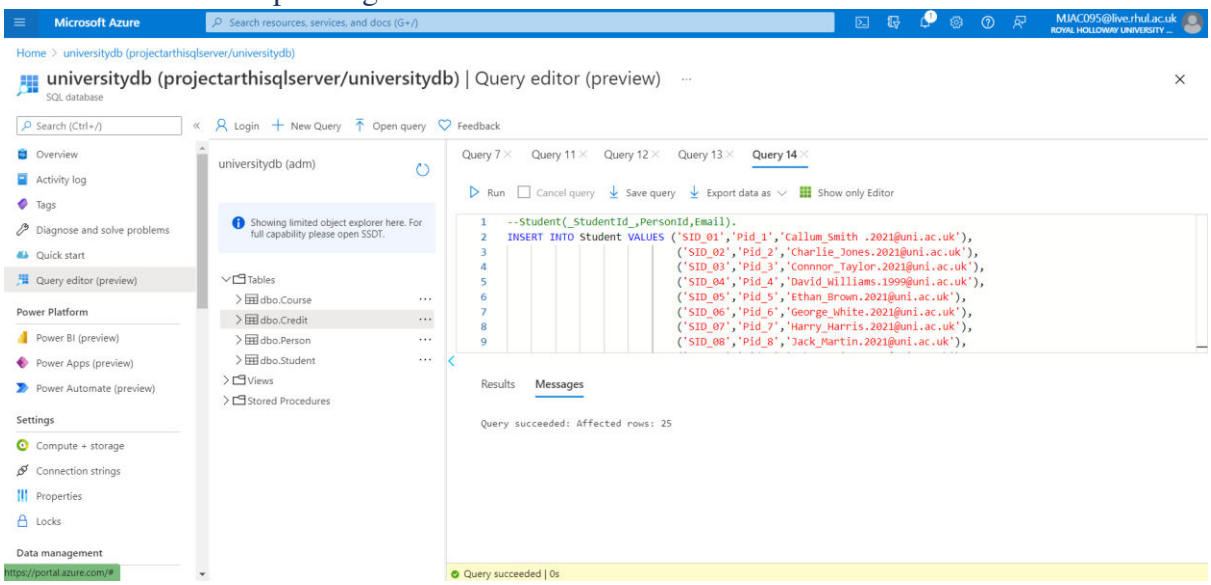
Screenshot 6.3.a: Populating Person Table



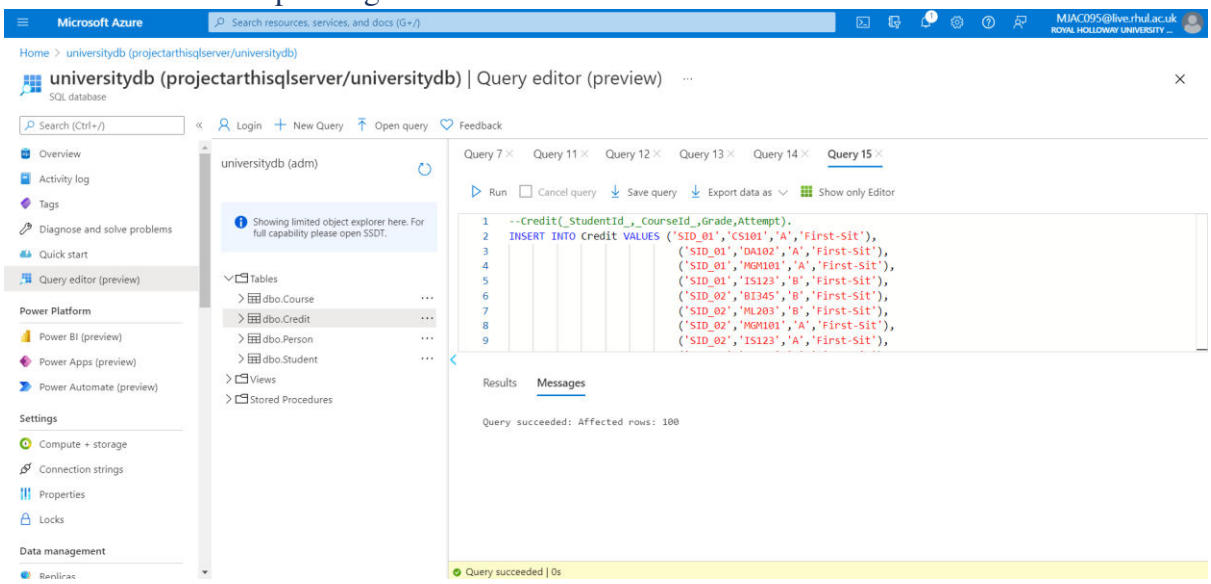
Screenshot 6.3.b: Populating Course Table



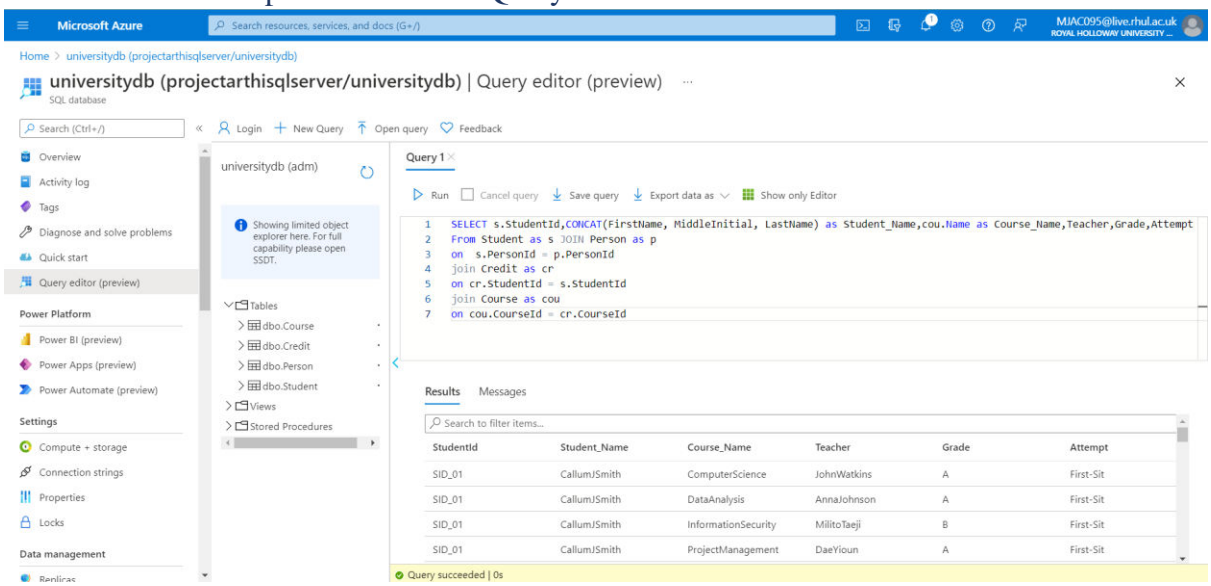
Screenshot 6.3.c: Populating Student Table



Screenshot 6.3.d: Populating Credit Table



Screenshot 6.4: Output Verification: Query to extract the student details



Files: attached create table, insert values, and extracting queries sql file of this Problem is in Solution6 Folder.

==========

Problem Statement 7: **Design a Relational Database Model for the below scenario in Azure SQL Database**

==========

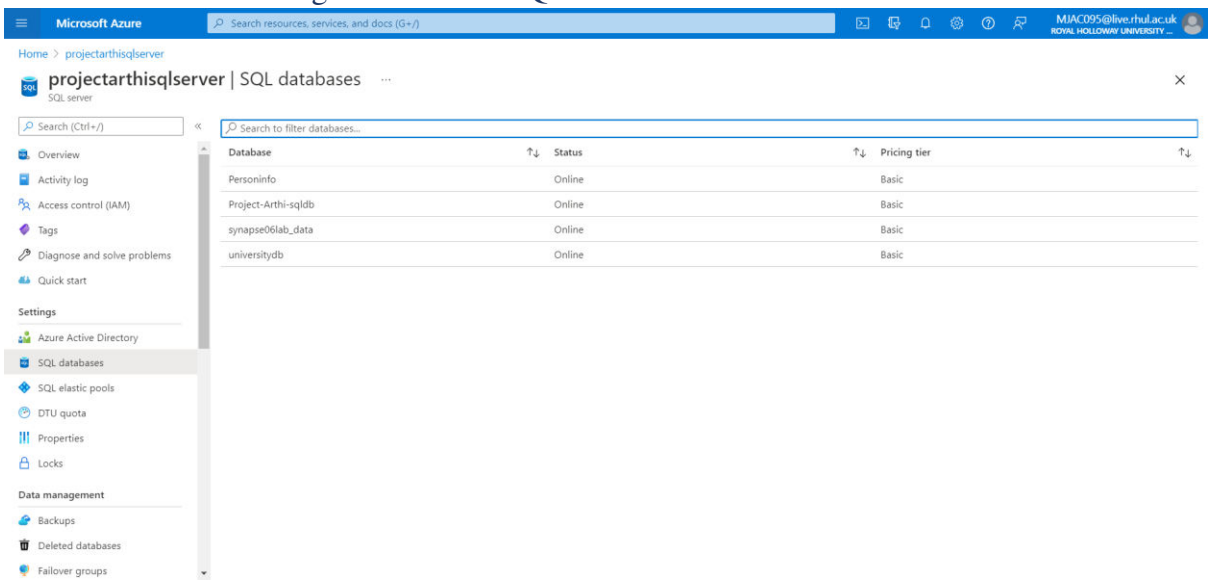
Solution :

- Created an Azure SQL Database named as: **Personinfo** containing 4 Tables namely

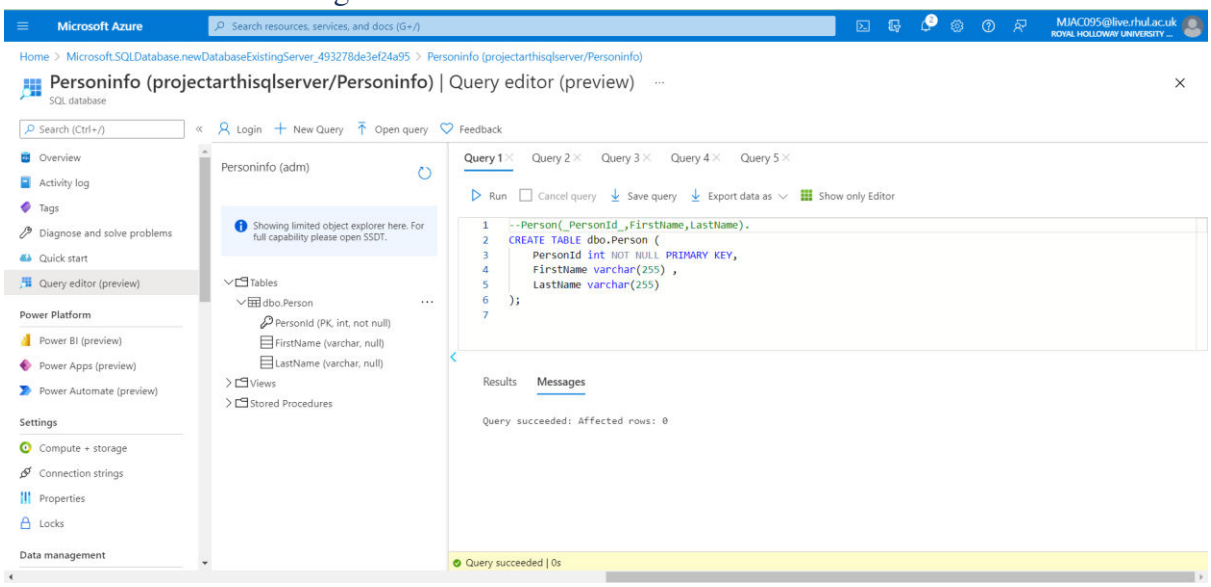
dbo.Person, dbo.Address,dbo.ContactDetail, dbo.ContactDetailType

- Created tables and relationship among them as given in the problem
- Created database and populated the table using insert query in each table using random values.

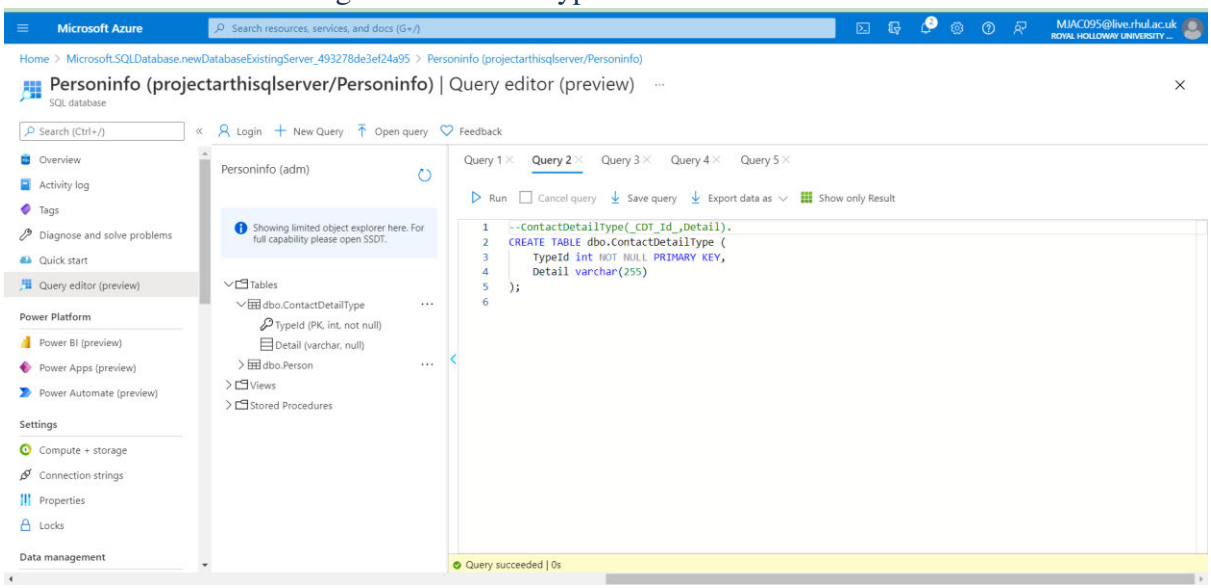
Screenshot 7.1: Creating DB in Azure SQL Database



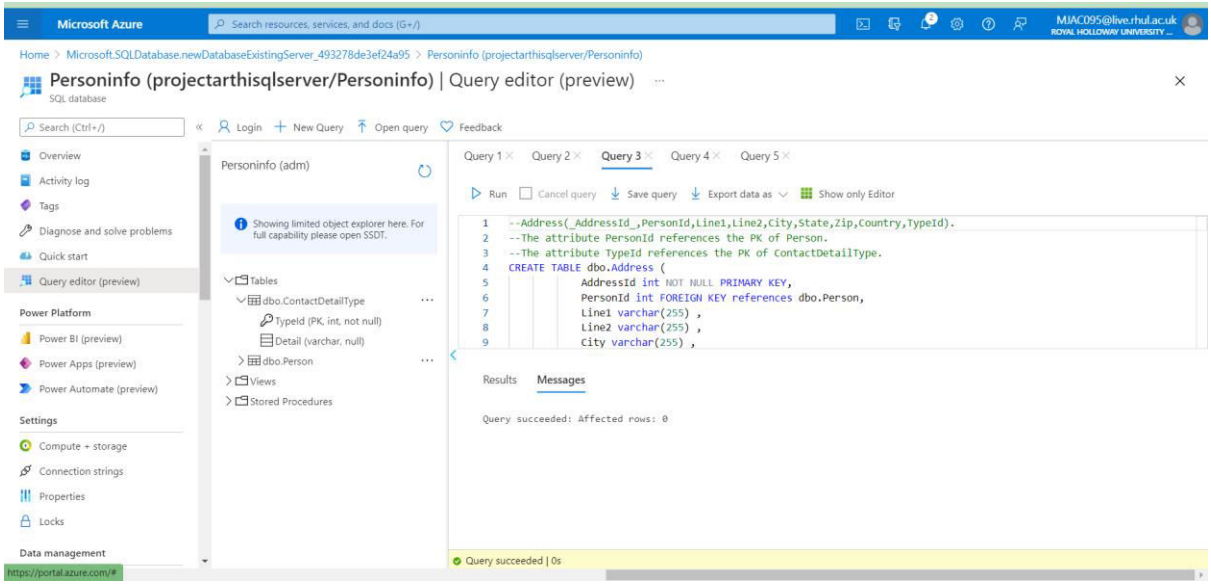
Screenshot 7.1.a: Creating Person Table



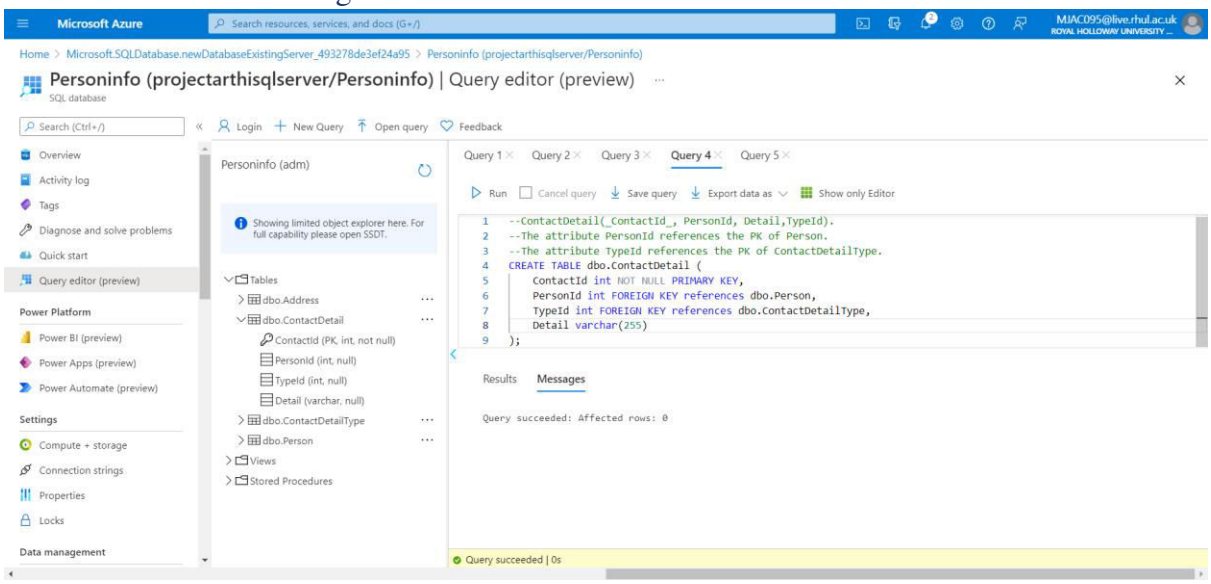
Screenshot 7.1.b: Creating ContactDetailType Table



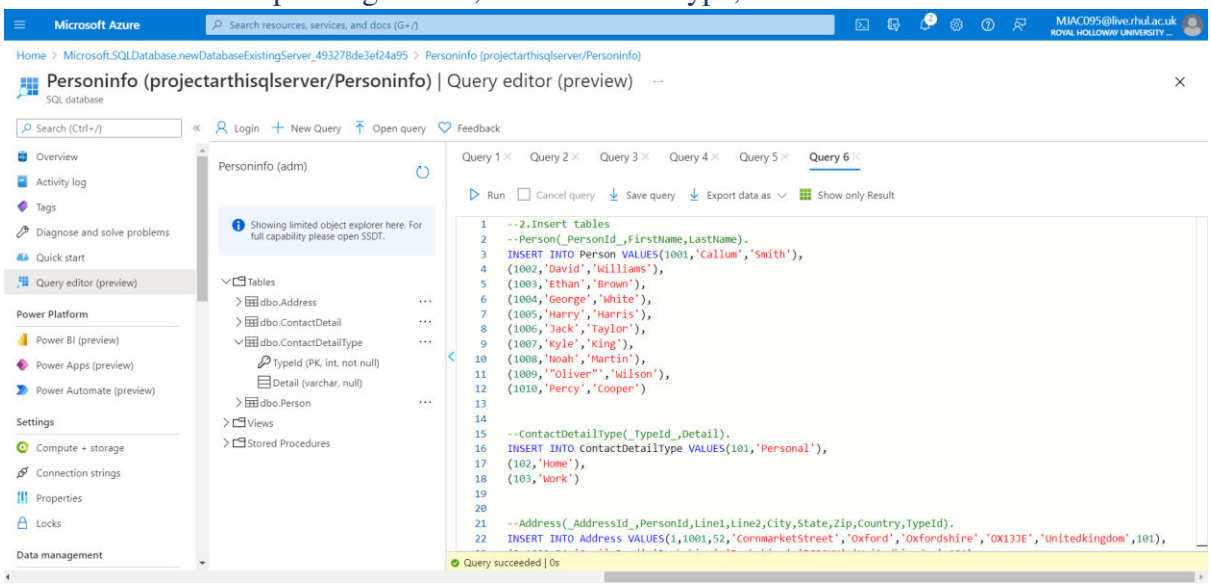
Screenshot 7.1.c: Creating Address Table



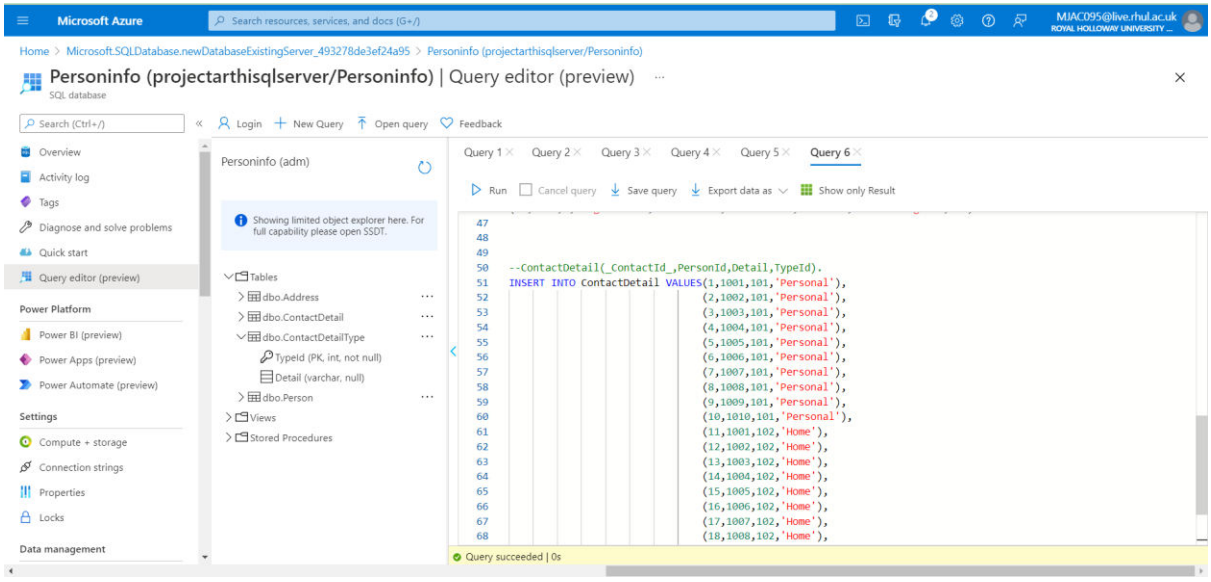
Screenshot 7.1.d: Creating ContactDetail Table



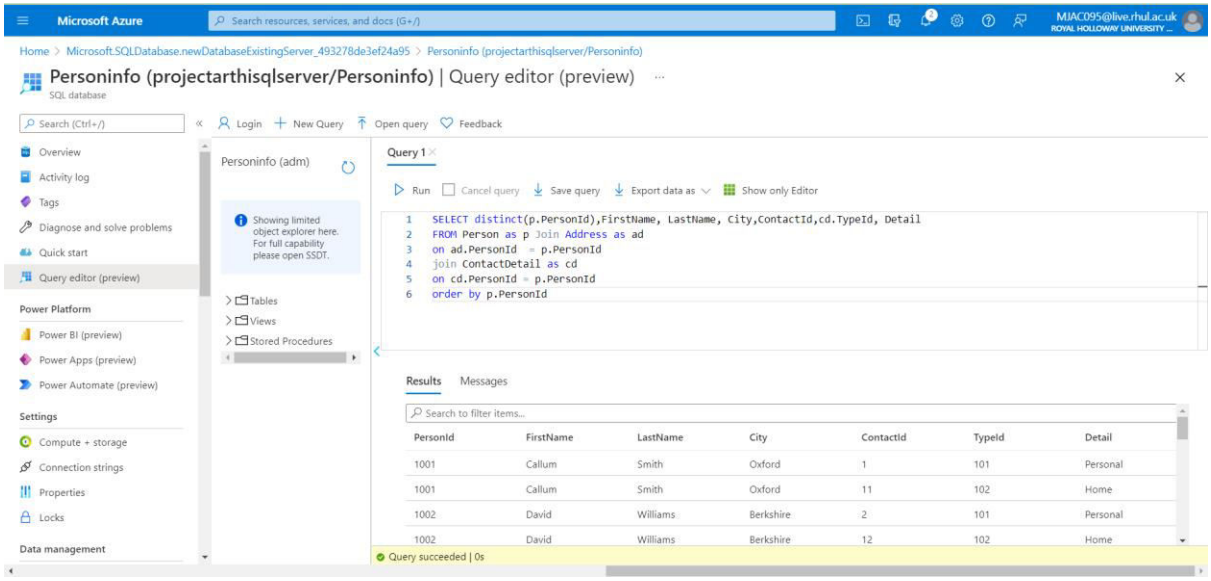
Screenshot 7.2.a: Populating Person, ContactDetailType, Address Table



Screenshot 7.2.b: Populating Table ContactDetail



Screenshot 7.3: Output Verification: Query to extract the Person details



Files: attached create table, insert values, and extracting queries sql file with this Project in Solution7 Folder.

==========

Problem Statement 8: **Design a Non-Relational Database Model equivalent for a Relational Model**

==========

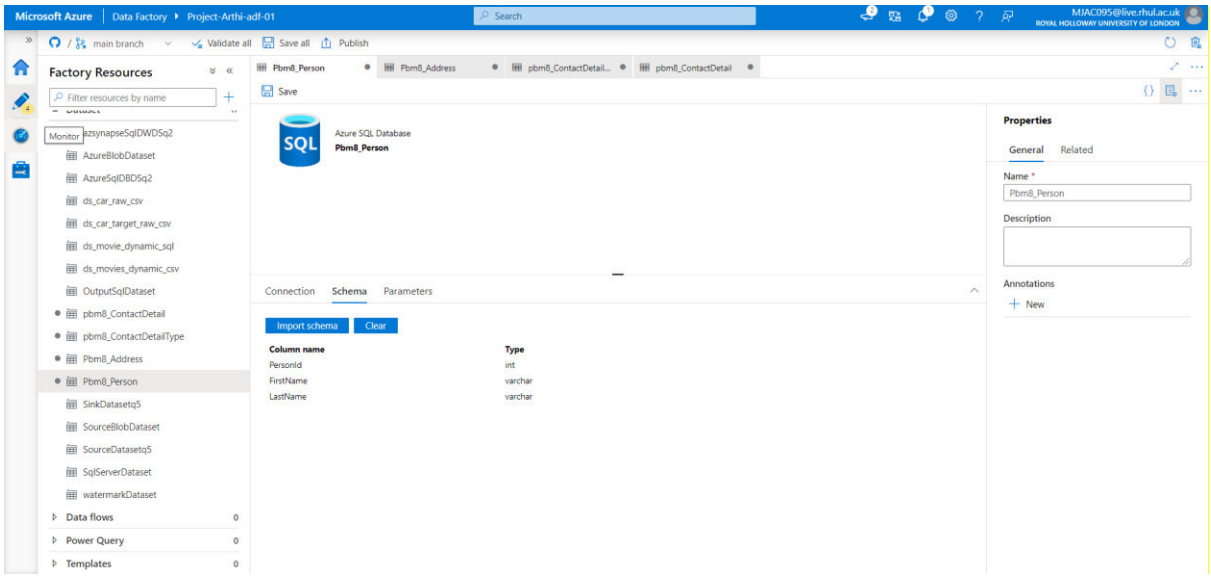
Solution :

- Created an Azure SQL Database named as: **Personinfo** containing 4 Tables namely

dbo.Person, dbo.Address,dbo.ContactDetail, dbo.ContactDetailType

- Created tables and relationship among them as given in the problem
- Created database and populated the table using insert query in each table using random values
- Created a dataset for each table in Azure Data factory to have table in JSON schema

Screenshot 8.1.a: Datasets for PersonInfoDb



Person - Schema:

```
{
   "name": "Pbm8_Person",
   "properties": {
      "linkedServiceName": {
         "referenceName": "ls_pbm8",
         "type": "LinkedServiceReference"
      },
      "annotations": [],
      "type": "AzureSqlTable",
      "schema": [
```

```
            {
                "name": "PersonId",
                "type": "int",
                "precision": 10
            },
            {
                "name": "FirstName",
                "type": "varchar"
            },
            {
                "name": "LastName",
                "type": "varchar"
            }
        ],
        "typeProperties": {
            "schema": "dbo",
            "table": "Person"
        }
    }
}
```

Address - Schema:

```
{
    "name": "Pbm8_Address",
    "properties": {
        "linkedServiceName": {
            "referenceName": "ls_pbm8",
            "type": "LinkedServiceReference"
        },
        "annotations": [],
        "type": "AzureSqlTable",
        "schema": [
            {
                "name": "AddressId",
                "type": "int",
                "precision": 10
            },
            {
                "name": "PersonId",
                "type": "int",
                "precision": 10
            },
            {
                "name": "Line1",
                "type": "varchar"
            },
            {
```

```
            "name": "Line2",
            "type": "varchar"
          },
          {
            "name": "City",
            "type": "varchar"
          },
          {
            "name": "State",
            "type": "varchar"
          },
          {
            "name": "Zip",
            "type": "varchar"
          },
          {
            "name": "Country",
            "type": "varchar"
          },
          {
            "name": "TypeId",
            "type": "int",
            "precision": 10
          }
        ],
        "typeProperties": {
          "schema": "dbo",
          "table": "Address"
        }
    }
}
```


ContactDetailType - Schema:


```
{
    "name": "pbm8_ContactDetailType",
    "properties": {
        "linkedServiceName": {
            "referenceName": "ls_pbm8",
            "type": "LinkedServiceReference"
        },
        "annotations": [],
        "type": "AzureSqlTable",
        "schema": [
            {
                "name": "TypeId",
                "type": "int",
                "precision": 10
```

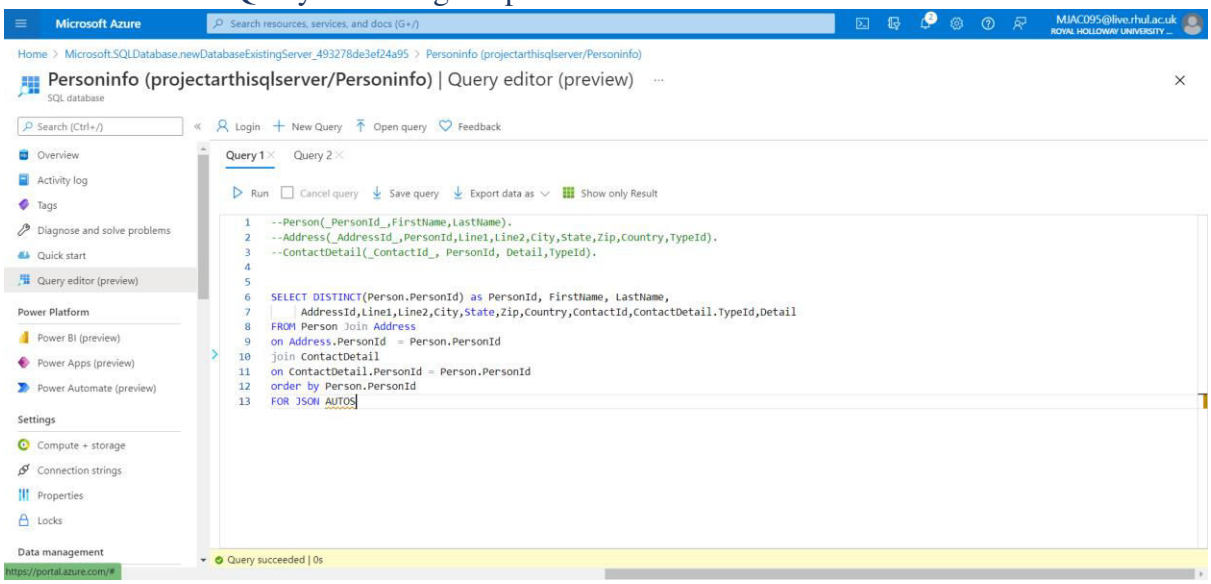
```
            },
            {
                "name": "Detail",
                "type": "varchar"
            }
        ],
        "typeProperties": {
            "schema": "dbo",
            "table": "ContactDetailType"
        }
    }
}
```

ContactDetail - Schema:

```
{
    "name": "pbm8_ContactDetail",
    "properties": {
        "linkedServiceName": {
            "referenceName": "ls_pbm8",
            "type": "LinkedServiceReference"
        },
        "annotations": [],
        "type": "AzureSqlTable",
        "schema": [
            {
                "name": "ContactId",
                "type": "int",
                "precision": 10
            },
            {
                "name": "PersonId",
                "type": "int",
                "precision": 10
            },
            {
                "name": "TypeId",
                "type": "int",
                "precision": 10
            },
            {
                "name": "Detail",
                "type": "varchar"
            }
        ],
```

```
        "typeProperties": {
            "schema": "dbo",
            "table": "ContactDetail"
        }
    }
}
```

Screenshot 8.1.b: Query extracting People full contact info

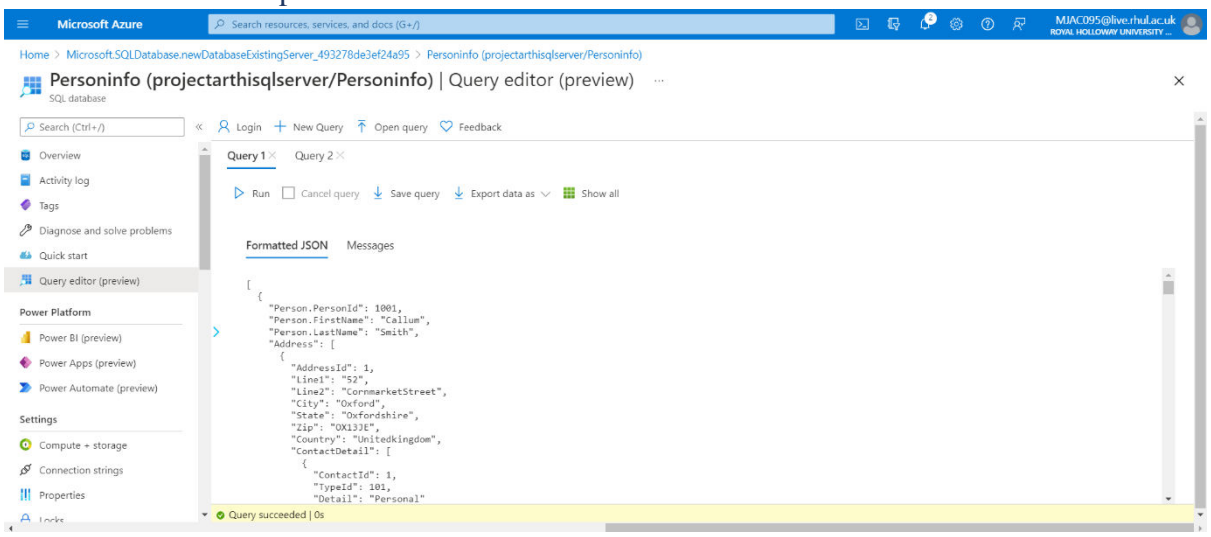


Query:

```
--Person(_PersonId_,FirstName,LastName).
--Address(_AddressId_,PersonId,Line1,Line2,City,State,Zip,Country,TypeId).
--ContactDetail(_ContactId_, PersonId, Detail,TypeId).


SELECT DISTINCT(Person.PersonId) as PersonId, FirstName, LastName,
    AddressId,Line1,Line2,City,State,Zip,Country,ContactId,ContactDetail.TypeId,Detail
FROM Person Join Address
on Address.PersonId  = Person.PersonId
join ContactDetail
on ContactDetail.PersonId = Person.PersonId
order by Person.PersonId
FOR JSON AUTO
```

Screenshot 8.1.c: Output in JSON FORMAT



Sample Output:


```
[
  {
    "Person.PersonId": 1001,
    "Person.FirstName": "Callum",
    "Person.LastName": "Smith",
    "Address": [
      {
        "AddressId": 1,
        "Line1": "52",
        "Line2": "CornmarketStreet",
        "City": "Oxford",
        "State": "Oxfordshire",
        "Zip": "OX13JE",
        "Country": "Unitedkingdom",
        "ContactDetail": [
          {
            "ContactId": 1,
            "TypeId": 101,
            "Detail": "Personal"
          }
        ]
      },
      {
        "AddressId": 11,
        "Line1": "52",
        "Line2": "CornmarketStreet",
        "City": "Oxford",
        "State": "Oxfordshire",
        "Zip": "OX13JE",
        "Country": "Unitedkingdom",
        "ContactDetail": [
          {
            "ContactId": 1,
            "TypeId": 101,
            "Detail": "Personal"
          }
        ]
```

```
    },
    {
     "AddressId": 1,
     "Line1": "52",
     "Line2": "CornmarketStreet",
     "City": "Oxford",
     "State": "Oxfordshire",
     "Zip": "OX13JE",
     "Country": "Unitedkingdom",
     "ContactDetail": [
      {
       "ContactId": 11,
       "TypeId": 102,
       "Detail": "Home"
      }
     ]
    },
    {
     "AddressId": 11,
     "Line1": "52",
     "Line2": "CornmarketStreet",
     "City": "Oxford",
     "State": "Oxfordshire",
     "Zip": "OX13JE",
     "Country": "Unitedkingdom",
     "ContactDetail": [
      {
       "ContactId": 11,
       "TypeId": 102,
       "Detail": "Home"
      }
     ]
    }
   ]
 },
```


Files: attached Complete output with this Project in Solution8 Folder.