**Problem Solution 1.**

With this first task, I provisioned several resources through my azure portal to achieve an expected result.

A screenshot of a computer

Description automatically generated

Figure 1a. Provisioned resources to perform various parts of the Project

First deployment was the data factory which would made it possible to create different pipelines, dataset and run them throughout the course of this Practice Lab.

Graphical user interface, text, application, email, website

Description automatically generated

Figure 1b. Deployed Data Factory

Second task was to create a self-hosted integration runtime which is instrumental in the running of the data factory such as data flows and data movements. For a Mac user as myself, I was not able to perform this task on my system, so I had to provision a virtual machine in my azure portal and connect through a Microsoft Remote Desktop to be able to utilise the capabilities of self-hosted integration runtime. My next task was to provision an SQL server through my portal, also very important in holding different databases deployed throughout the course of this Project.

Graphical user interface, text, application

Description automatically generated

Figure 1c. Copying data from SQL Server to Blob Storage Container Pipeline

Through my data factory, I was able to deploy linked service for my data source and its destination of storage and create a Sql dataset through the author tab. Finally, a pipeline was created to facilitate the movement of data from the SQL server to the created blob storage container (Copy Data activity was used here).

Graphical user interface, text, application, email

Description automatically generated

Figure 1d. Created Blob Storage and Adftutorial Container

**Problem Solution 2.**

The second task was to ingest data from multiple tables to an azure blob storage using the loop activity from our data factory. As most of the resources needed have been provisioned, it was quite straight forward to work on this task. A linked service for the source data was deployed. This had already been created from performing Problem Statement 1 above. The next was to deploy a linked service for sink data store which is the Azure Synapse Analytics and then finally deploy another linked service for the staging storage which is going to be the Blob storage. The next step was to deploy datasets for both the source and sink pointing to location of stored data. It was very important to have a naming convention and the right naming format throughout that makes it easier to connect the linked services to the right datasets as I was going through this task. Under the connection tab of the source dataset, a dummy table had to be used as query was going to be used to extract the data stored in my database.

In the connection tab of the sink dataset, a dynamic content had to be added to the table section to help reduce the number of pipelines and activities needed to accomplish this task. The next is to create a pipeline to perform continues copying of tables from our Sql tables. A parameter was added with a tableList as it name and Array was used as it type.

Graphical user interface, text, application, email

Description automatically generated

Figure 2a. Copy Data activity inside Foreach activity

Foreach activity is used but also different dynamic contents were used throughout this task to reduce pipelines and activities to be deployed. A copy data activity was used inside the foreach activity for the source and sink datasets but also polybase was selected as a copy method because of its scalability and speed of loading data.

Graphical user interface, text, application, email

Description automatically generated

Figure 2b. Lookup and Execution Pipeline to trigger and copy data from SQL table

Another pipeline (execution pipeline) was also deployed in addition to the earlier lookup activity to get the list of tables from the database to be copied and that makes it possible to trigger the continues copying from the Sql tables.

**Problem Solution 3.**

This task involves loading data from the blob storage created previously to a database also created at the beginning of this project Lab. Data factory is already deployed also in the previous task so the next step would be to start with creating pipelines to perform the task at hand. Copy data activity is then dragged onto the canvas area of the pipeline.

A screenshot of a computer

Description automatically generated

Figure 3a. Configuring Copy data Activity both Source and Sink to perform data migration

A new dataset is also created and connected with a linked service but also its important specify the file that is been used, which is in the blob storage to populate the table in the database. This task is done for the source of our copy data. The sink tab is also configured by creating another dataset and associating it a linked service.

A screenshot of a computer

Description automatically generated

Figure 3b. Successful run of the trigger for the copy data activity to perform data movement.

The next step is validation and debugging of the pipeline to check for errors. Once successful, the pipeline is published and then triggered to perform the copy activity.

Graphical user interface

Description automatically generated

Figure 3c. Results of the copy activity from the trigger run

Problem Solution 4.

The next task was to load files from different locations into my created database. First task was to download the source file and upload them into a new container and different folders based on the downloaded files in the blob storage. Tables are then created in the database to hold metadata, running another script to insert the metadata into the created table and finally the sink tables are also created. The next step is to create blob dataset for our source data, ensuring the right parameters are specified and including dynamic content for the file path, column delimiter as they help reduce the level of pipeline and activities that must be deployed. The next dataset is for the sink data, which is going to be an Azure Sql Database, making sure the right parameter is specified again and the correct dynamic contents are added to the dataset connection.

Next is deploying pipelines to move flat data from blob container to table in the Sql database. Lookup activity is first used, and queries are used to connect to the database making it possible to use it as a place holder for the metadata. Next is to add foreach activity, making sure to add a dynamic content to the item section in settings. This will make it possible to select the required data out of the lookup activity.

Graphical user interface, text

Description automatically generated

Figure 4a. Successful run of the pipeline for Lookup, Foreach and Copy activity

Last step is to add copy data (from blob to Sql) activity inside the foreach activity making sure the right value is provided for parameters specified during the provision of the sink data and selecting the right dataset to use for the activity. The last step is to debug the pipeline. The results are displayed above.

Graphical user interface, application, table, email

Description automatically generated

Figure 4b. Results from moving our flat data(metadata) to table in SQL table.

One issue encountered was the naming format (Used lowercase letters) when I created my tables in my database and then used (Uppercase letters) for folder names created inside the Blob storage. That meant my copy activity could not locate or connect to the right folder to perform it task, resulting in errors. Carefully monitoring the error helped me resolve it very quickly.

**Problem 5 Solution.**

With this task, a table is created, and data is inserted as the data source. Another table is then created, and data inserted for the watermark. The next step is to create a stored procedure in the database. Next is to use data factory to create pipeline to hold the first activity, which is the lookup activity and then creating a dataset for source data (watermarkdataset). It is also important to connect a linked service to the dataset and ensuring the right table is selected under the connection tab. The next is to add another lookup activity to get the new watermark value to facilitate data movement from the source to the sink database.

A new data set is then created to set direction to the new watermark value created. Here, it is also very important to select the right table making it possible to run query on this table. Also, it’s important select the correct dataset for this activity.

Graphical user interface, application

Description automatically generated

Figure 5a. Pipeline to execute Lookup to Stored procedure activities

The next activity is dragging the copy data activity onto the canvas with the two lookup activity. Selecting the source data, the correct dataset must be selected, and query checked for running of the select query. Next is to select the sink tab and making sure blob storage is used in this section as the dataset, making sure the right blob storage, container and folder is selected. The next activity to use the stored procedure and connecting the copy data to it. For the stored procedure, the right parameters must be imported, and the correct values inserted to get the right results from the pipeline activities designed. The next is to validate the pipeline and add a trigger to run the activities.

Graphical user interface, text, application, email

Description automatically generated

Figure 5b. Successful file creation and migration of incremental data from Sql to adftutorial blob Storage

**Problem Solution 6.**

With this task, a database was created in Azure data studio with a deployed Sql server attached to it. The next step involved creating four different relational tables to hold random information about students. Each table had to be defined making sure the right column is used for both primary keys and foreign keys and setting up the correct constraints to deal with when data must be updated or deleted and establish a relation between the different tables. Other columns of the table had to also be defined based on either it been an INT, Char, Varchar and Date. Each table was then populated, and a few queries run to ensure the table can be used for any intended purpose. Joins were also used to combine multiple tables to view results from executed query of SELECT. Below are the codes used to achieve this task.

Creating and Populating Database

1.(PersonId Int Identity Primary key,

FirstName varchar (255) not null,

MiddleInitial varchar (255) not null,

LastName varchar (255) not null,

DateofBirth Int not null

—Insert Data into Person Table

Insert into Person values ('Spider', 'W', 'Man', '1990-01-01')

Insert into Person values ('Super', 'S', 'Man', '1989-02-01')

Insert into Person values ('Harley', 'C', 'Quinn', '1987-03-01')

Insert into Person values ('Wonder', 'W', 'Woman', '1991-04-01')

select \* from Person

1. Create table Student

(

StudentId Int IDENTITY Primary key not null,

Personid Int not null,

Email varchar(255) not null ,

Constraint FK1 foreign key (Personid) references

Person

—Insert Data into Student Table

Insert into Student values (1, 'Spider@gmail.com')

Insert into Student values (2, 'Super@gmail.com')

Insert into Student values (3, 'Harley@gmail.com')

Insert into Student values (4, 'Wonder@gmail.com')

select \* from Person

3. Create table Course

(

CourseId Int IDENTITY Primary key not null,

CourseName varchar (255) not null,

Teacher varchar (255) not null

);

-- Insert Data into Course Table

Insert into Course values ('Math', 'Ms Grace White')

Insert into Course values ('English', 'Mr Mark Taylor')

Insert into Course values ('Physics', 'Ms Char Brown')

Insert into Course values ('Art', 'Mr Enzo Rice')

select \* from Course

4.Create table Credit

(

CreditId Int identity Primary key not null,

StudentId Int not null,

Grade Int not null,

Attempt Int not null,

Constraint FK2 foreign key (StudentId) references

Student,

Constraint FK3 foreign key (CourseId) references

Course

)

-- Insert Data into Credit Table

Insert into Credit values (1,70, 1)

Insert into Credit values (2,72, 1)

Insert into Credit values (3,74, 1)

Insert into Credit values (4,76, 1)

select \* from Credit

Select FirstName, LastName, Teacher, Grade, Attempt

From Person

inner join Student on Person.PersonId = Student.PersonId

inner join Credit on Student.StudentId = Credit.StudentId

inner join Course on Credit.CourseId = Course.CourseId

One of the issues I encountered working on this task was that I couldn’t run my lines of code altogether as it took more than 30min to do so twice. Best solution to tackle that was to cancel the query operation, drop my entire database to eliminate errors and then a setup a new one. Moving forward, I ensured each block of code was executed separately to optimize time and operation on the database.

**Problem Solution 7**.

This task was similar to that of Project 6 above.

1. Connected my sql server to my azure data studio as a Mac user.
2. Created a database to store my tables.
3. Created tables and defined data type and constraints for each table to establish relationships between them for ease of query.
4. Fake data is then inserted into each table.
5. Finally, table joins where used to combine several tables to show the required data from the executed query.

As noted in project 6, each block of code was executed separately to ensure a fast running and efficient running of the databasese.

Below are the lines of code used to achieve results of this task.

1. Create database Personinfo
2. Create table Person

(

PersonId Int Identity Primary key,

FirstName varchar (255),

LastName varchar (255)

);

1. Create table Address

(

AddressId Int Identity Primary Key,

PersonId Int,

Line1 varchar (255) not null,

Line2 varchar (255) not null,

City varchar (255) not null,

State varchar (255) not null,

Zip Int,

Country varchar (255) not null,

TypeId Int,

Constraint FK1 foreign key (PersonId) references

Person

);

1. Create table ContactDetailType

(

TypeId Int Primary Key,

Detail varchar (255)

);

1. Create table ContactDetails

(

ContactId Int Identity Primary key,

Personid Int,

Details1 varchar(255) not null,

Details2 varchar(255) not null,

Details3 varchar(255) not null,

TypeId Int

Constraint FK2 foreign key (TypeId) references

ContactDetailType

);

6a. From here you insert data into the created tables using multiple rows insert method. Again, the purpose is to explore different method of inserting into tables different from the approach used in Project Solution 6.

Insert into Person (

FirstName,

LastName

)

Values

('Kaden',

'Munoz'),

('Dayton',

'Barr'),

('Lamont',

'Avery'),

('Allisson',

'Green'

);

Select \* from Person

6b.

Insert into Address (

PersonId,

Line1,

Line2,

City,

State,

Zip,

Country,

TypeId

)

VALUES

(

1, '124 Hindhead Road', 'Easington Lane', 'London',

'DH5 3QW', 000, 'United Kingdom', 1

)

(

2, '85 Newmarket Road', 'Hayscastle', 'London',

'SA62 2RX' ,000, 'United Kingdom', 2

),

(

3, '130 Scarcroft Road', 'Porthrhyd', 'London',

'SA32 6RW' ,000, 'United Kingdom', 3

),

(

4, '86 Overton Circle', 'littleworth', 'London',

'SN7 2ZZ',000, 'United Kingdom', 4

);

Select \* from Address

6c.

Insert into ContactDetailType (

TypeId ,

Detail

)

VALUES

(

1, 'Contact through email, alternatively through mobile'

),

(

2, 'Contact through post, alternatively through email'

),

(

3,'Contact through mobile only'

),

(

4, 'Contact through post only '

);

Select \* from ContactDetailType

6d.

Insert into ContactDetails

(

PersonId,

Details1,

Details2,

Details3,

TypeId

)

Values

(

1, 'Kadenmz@temporary-mail.net', 'DH5 3QW', 07447134958, 1

),

(

2, 'Daytonbarr@temporary-mail.net','SA62 2RX', 07773465839, 2

),

(

3, 'LamontAy@temporary-mail.net','SA32 6RW', 07899934554, 3

),

(

4, 'Allissongreen@temporary-mail.net','SN7 2ZZ', 07885938598,4

);

1. Use these queries to look at the data you have inserted into your tables

Select \* from ContactDetail.

Also looking at the schema of the database and it tables, it was important to create ContactDetailType table first before creating ContactDetail because of the foreign key based on ContactDetailType needed to establish a relationship with ContactDetail. Regardless of this information, I tried to create the ContactDetail table first and the query returned many errors, which then reaffirmed my earlier method of table creation.

1. Then you run this query to retrieve the information from the tables created as per the task

select FirstName, LastName, City, Details1,Details2,Details3 from

Person

inner join ContactDetails on Person.PersonId = ContactDetails.PersonId

inner join Address on Person.PersonId = Address.Personid

1. Problem Solution 8

This task involved taking several steps which involved various research and learning to aid in the solution that was required as I had little knowledge on non-relational database. Most of the knowledge to accomplish this task was from the Microsoft documents available on their website and reading blog posts. To iterate the steps taking.

Graphical user interface, application

Description automatically generated

8a. Deployed Cosmos DB hosting Database, Container, and Item (JSON Document)

1. New to Nosql database, queries, and creation of documents.
2. Had to research on how to perform this task which I found Microsoft documentation and several blog post to be very useful.
3. I had to create an Azure Cosmos DB and deploy to write in Sql rather than using Mongo DB to be able to write my statements and insert all my fake data.
4. A new database, container and partition key were created to support hosting of non-relational data.
5. Within the created container, new items were added to store person information and made it possible to store multiple address for some individuals.
6. Finally, a query was executed against this container to retrieve person information based on the partition key of firstName deployed at an earlier. One challenge I found here was how to query non-relational database, so I had to learn the basic for now and then work on learning the others for future reference and work.

Below is information inserted into the JSON document created.

{

"id": "1",

"firstName": "Chris",

"lastName": "Tuck",

"addresses": [

{

"line1": "124 Hindhead Road'",

"line2": "Easington Lane'",

"city": "London'",

"Postcode": "DH5 3QW",

"Country": "United Kingdom"

}

],

"contactDetails": [

{

"email": "christ@andersen.com"

},

{

"phone": "07334345345"

}

{

"id": "2",

"firstName": "Kaden",

"lastName": "Munoz",

"addresses": [

{

"line1": "85 Newmarket Road''",

"line2": "Hayscastle",

"city": "London",

"Postcode": "SA62 2RX",

"Country": "United Kingdom"

}

],

"contactDetails": [

{"email": "kadenmun@gmail.com"},

{"phone": "07334568345"}

]

}

{

"id": "3",

"firstName": "Dayton",

"lastName": "Bar",

"addresses": [

{

"line1": "130 Scarcroft Road",

"line2": "Porthrhyd",

"city": "London",

"Postcode": "SA32 6RW",

"Country": "United Kingdom"

}

],

"contactDetails": [

{"email": "Daybar@gmail.com"},

{"phone": "07334341225"}

]

}

{

"id": "4",

"firstName": "Allisson",

"lastName": "Green",

"addresses": [

{

"line1": "86 Overton Circle",

"line2": "littleworth",

"city": "London",

"Postcode": "SN7 6RW",

"Country": "United Kingdom"

}

],

"contactDetails": [

{

"email": "Greenall@gmail.com"

},

{

"phone": "077364341225"

}

]

**Final Thoughts**

This project has really taken me through different parts of Azure, re-enforcing and bringing to the forefront of my learning path many of the topics covered during the academy. It also has highlighted several aspects of azure that needs extra attention and mastering to ensure full competency of the Azure platform. The learning still continuous.

Regardless, it was a Project that was fun, collaborative and challenging to have delt with and fully complete on time and with confidence.