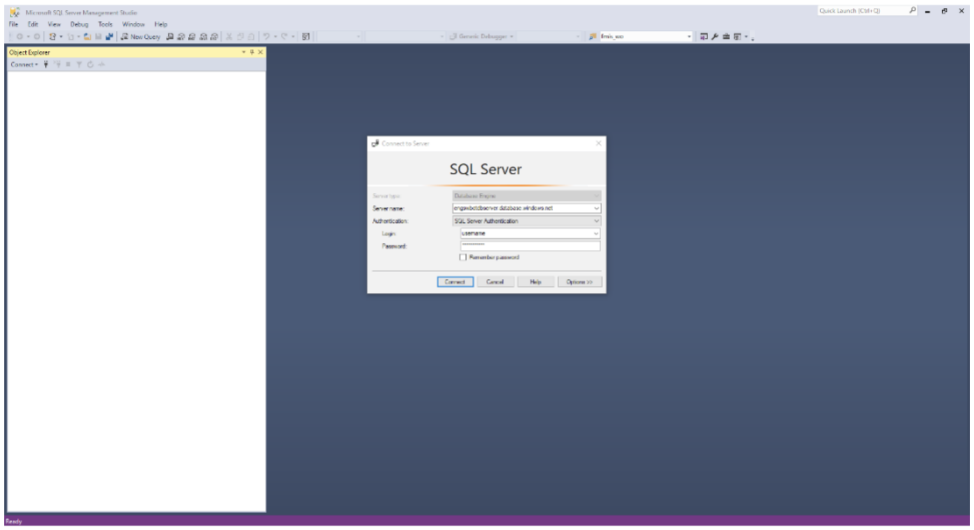
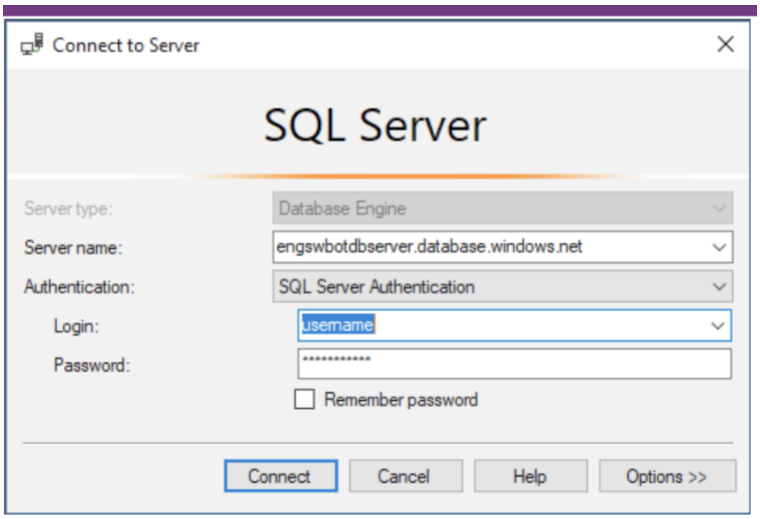
Senior Design 2021 - Database Implementation

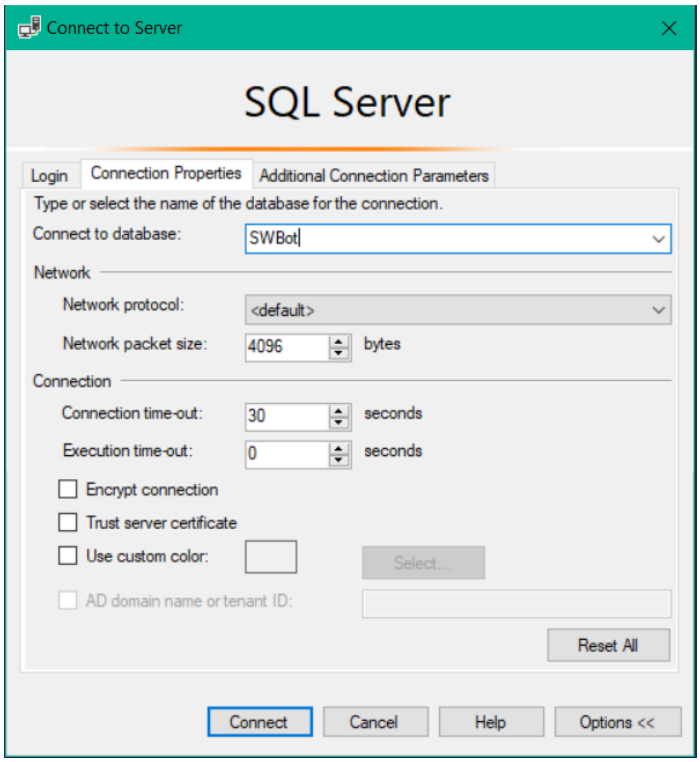
1. Backend database
    a. The backend database ties together the rest of the software used on the Leo Rover. Allowing us to collect, store, and read the data needed to accomplish the rovers mission. We're working with an Azure SQL DB and utilizing other tools such as Azure Blob Storage for automation and image storage.

2. Access
    a. Connect to Azure SQL Database Using Microsoft SQL Server Management Studio (SSMS)
        i. Open Microsoft SQL Server Management Studio.
        ii. In the Connect to Server dialog box, log in using the Server name and credentials provided.NOTE: Make sure to select Authentication as 'SQL Server Authentication'.

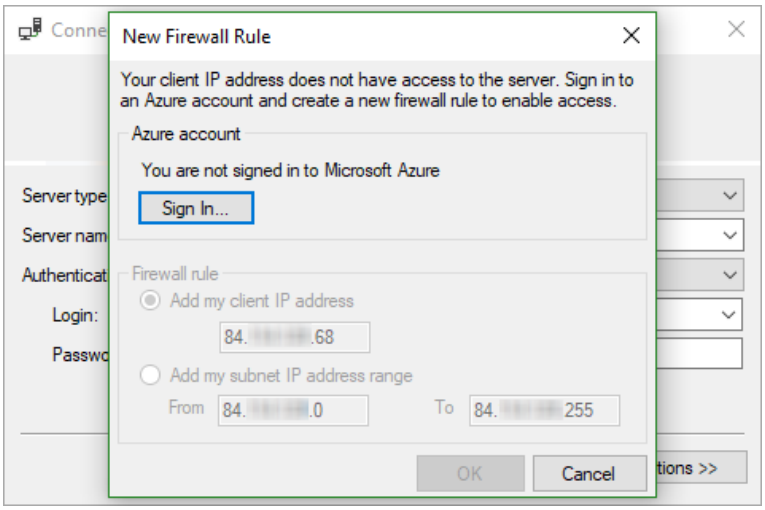




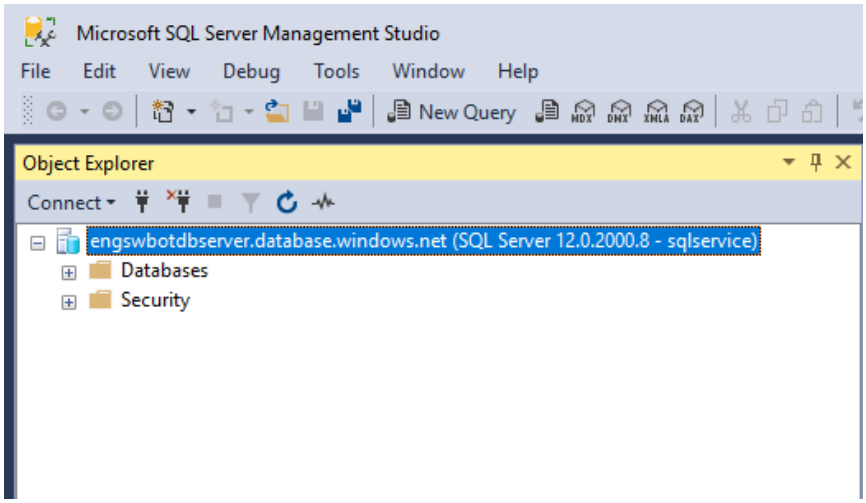
iii. Before connecting, click on **Options>>**. Under *Connection Properties*, *Connect to database*: enter **SWBot** (see screenshot below).



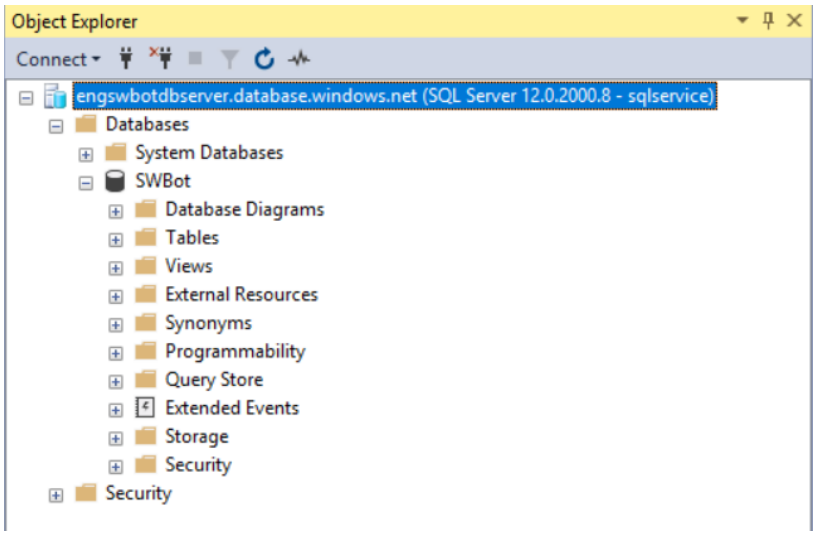
iv. Click “Connect”. You may receive a message as shown below requesting to add your IP address to the allowed firewall rules. Please send your IP address to email: fabiola.desantiago@lacity.org so that we may add the IP to the rules. Once this have been completed you should be able to connect successfully to the server.



v. Once you have successfully connected to the server, Object Explorer should display like the screenshot below.



vi. In **Object Explorer** (tab on the left-hand side), expand **Databases** and then expand '**SWBot**' to view the objects in the database as shown below.



3. Data
    a. Data is collected from the various hardware components on the Rover--such as a gps module, level, GoPro, Depth Cameras, and more.
    The data collected consists of :
        i. Raw GPS
        ii. Time stamps
        iii. Latitude
        iv. Latitude Reference
        v. Longitude
        vi. Longitude Reference
        vii. X Slope
        viii. Y Slope
        ix. Angle
        x. PinID
        xi. Image URL
    b. Data is collected from the Image / GoPro metadata and is recorded onto the GoProImageData and EXIF DATA table within Azure. The data collected consists of :
        i. Latitude Reference
        ii. Latitude
        iii. Longitude
        iv. Longitude Reference
        v. ImageType
        vi. TimeStamps
        vii. Dates
        viii. Orientation

ix. PinID
x. ImageID
xi. URL Front
xii. URL Back

4. Tables (Code) - SQL Code used to create our DB tables and staging tables.
   a. GoProImageData

```
CREATE TABLE [dbo].[GoProImageData](
    [LatitudeRef] [char](1) NULL,
    [Latitude] [nvarchar](128) NOT NULL,
    [LongitudeRef] [char](1) NULL,
    [Longitude] [nvarchar](128) NOT NULL,
    [ImageType] [varchar](128) NULL,
    [Timestamps] [nvarchar](128) NULL,
    [Dates] [nvarchar](68) NOT NULL,
    [Orientation] [nvarchar](50) NULL,
    [PinID] [nvarchar](14) NULL,
    [ImageID] [nvarchar](50) NULL,
    [UrlFront] [varchar](1000) NULL,
    [UrlBack] [varchar](1000) NULL,
    [UrlRendered] [varchar](1000) NULL
) ON [PRIMARY]
```

   b. RoverData

```
CREATE TABLE [dbo].[RoverData](
    [RawGPS] [nvarchar](512) NOT NULL,
    [Timestamps] [datetime2](7) NOT NULL,
    [Latitude] [float] NOT NULL,
    [LatitudeRef] [char](1) NULL,
    [Longitude] [float] NOT NULL,
    [LongitudeRef] [char](1) NULL,
    [X_Slope] [nvarchar](50) NOT NULL,
    [Y_Slope] [nvarchar](50) NOT NULL,
    [Angle] [decimal](18, 0) NULL,
    [PinID] [nvarchar](14) NULL,
    [ImageURL] [nvarchar](1000) NULL
) ON [PRIMARY]
```

   c. EXIF_DATA

```
CREATE TABLE [stage].[EXIF_DATA](
    [LatitiudeRef] [nvarchar](50) NOT NULL,
    [Latititude] [float] NOT NULL,
    [LongitudeRef] [nvarchar](50) NOT NULL,
    [Longitude] [float] NOT NULL,
    [Image_Type] [nvarchar](50) NOT NULL,
    [Dates] [datetime2](7) NOT NULL,
    [Timestamps] [nvarchar](50) NOT NULL,
    [Orientation] [nvarchar](50) NOT NULL,
    [ImageID] [nvarchar](50) NOT NULL
) ON [PRIMARY]
```

   d. GOPROEXIF

```
CREATE TABLE [stage].[GOPROEXIF](
    [Latititude] [float] NOT NULL,
    [Longitude] [float] NOT NULL,
    [Image_Type] [nvarchar](50) NOT NULL,
    [Dates] [datetime2](7) NOT NULL,
    [Timestamps] [nvarchar](50) NOT NULL,
    [Orientation] [nvarchar](50) NOT NULL,
    [ImageID] [nvarchar](50) NOT NULL
) ON [PRIMARY]
```

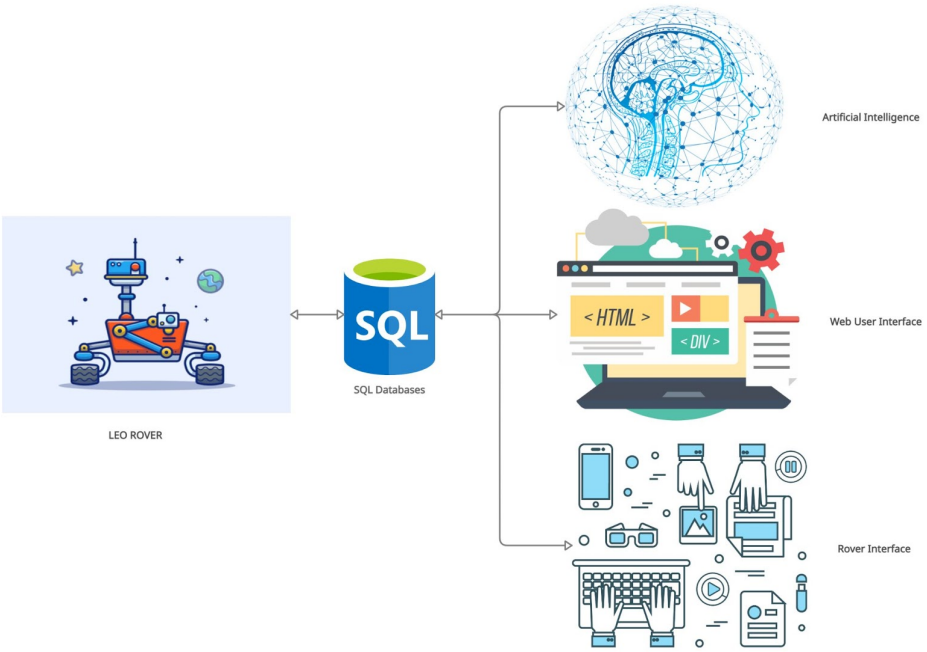
e. ROVERDATA

```
CREATE TABLE [stage].[ROVERDATA](
    [RawGPS] [nvarchar](100) NOT NULL,
    [Timestamp] [datetime2](7) NOT NULL,
    [Latitude] [float] NOT NULL,
    [Longitude] [float] NOT NULL,
    [X_slope] [nvarchar](50) NULL,
    [Y_slope] [nvarchar](50) NULL
) ON [PRIMARY]
```

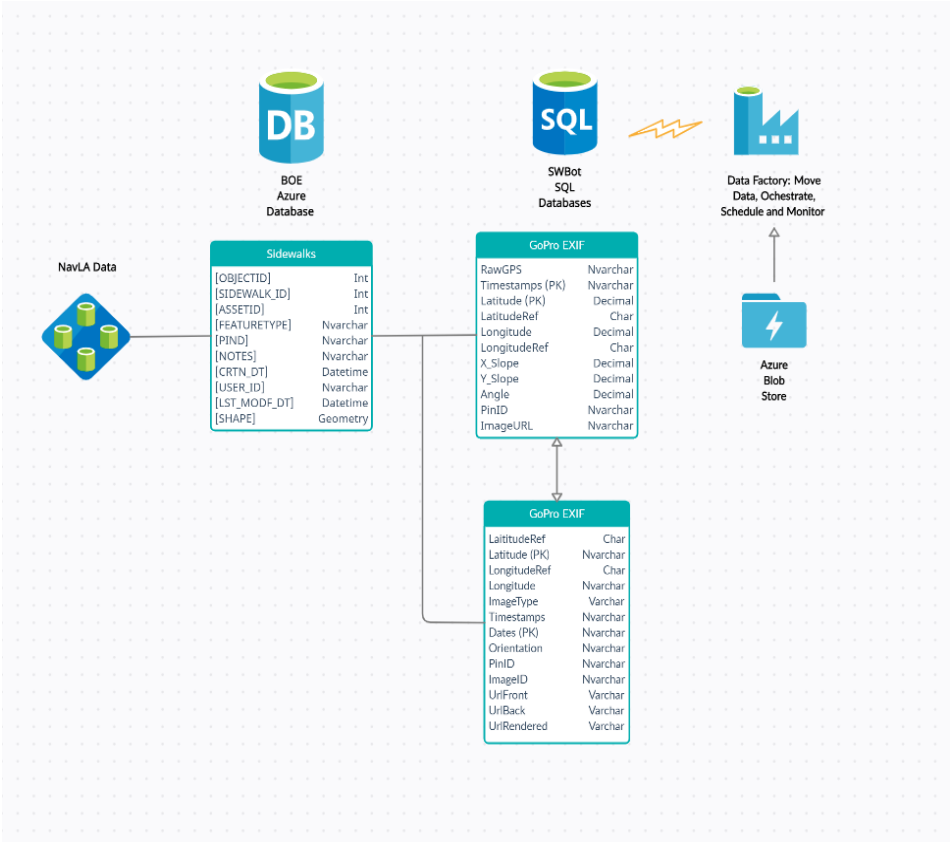
f. SLOPE_DATA

```
CREATE TABLE [stage].[SLOPE_DATA](
    [Raw_Data] [nvarchar](100) NOT NULL,
    [Date_Time] [datetime2](7) NOT NULL,
    [Latitude] [float] NOT NULL,
    [Lat_Ref] [nvarchar](50) NOT NULL,
    [Longitude] [float] NOT NULL,
    [Long_Ref] [nvarchar](50) NOT NULL,
    [X_slope] [nvarchar](50) NULL,
    [Y_slope] [nvarchar](50) NULL
) ON [PRIMARY]
```

5. Schema



Note: Top table should be labeled: “Rover Data” not “GoPro EXIF”



6. Linking our Data to the Navigate Los Angeles Data using GPS Coordinates
    a. Correlating our data to the data in existing BOE Database tables is a work in progress but we have a temporary solution which we were not able to implement because of time constraints.

       Our plan was to correlate our data to the data in the NavLa Sidewalks table using PinID as our identifier. By taking our Lat/Long coordinates which we pull from our GPS module and GoPro Exif, we can transform those coordinates into WGS Web Mercator format and once you've updated your table with Web Mercator coordinate fields, create a SHAPE field, then create the point geometries. This will allow us to grab the closest PinID to the coordinates provided--allowing us to tie our data to that of its surroundings.

       BOE Nathans approach Code:

       --SQL Statement to get the polygon coordinates from the sidewalk table (not needed now):

       SELECT TOP(1000) OBJECTID, SIDEWALK_ID, ASSETID, FEATURETYPE, PIND, shape.STSrid as SRID,
       replace(replace(SHAPE.STAsText(),'))',''),'POLYGON ((','') AS COORDINATES
       FROM dbo.sdwkapp_sidewalks_wm

       --Once you've updated your table with Web Mercator coordinate fields, create a SHAPE field, then create the point geometries:

       UPDATE {your_table_here} SET SHAPE =
       geometry::STGeomFromText('POINT(' + cast({your_table_X_field} as varchar) + '
       ' + cast (your_table_Y_field} as varchar) + ')', 3857))

       --Create the spatial index for your table (increases speed):

       Create spatial Index [FDO_Shape] on [{your_table_here}]
       (
              [Shape]
) using geometry_grid with
       (bounding_box = (-13224000,3985000,-13138000,4079000),
       grids = (level_1 = Medium, level_2 = Medium,level_3 = Medium,level_4 =
       Medium),
       cells_per_object = 16, pad_index = off, sort_in_tempDB = off, drop_existing = off,
       allow_row_locks = on, allow_page_locks = on) on [PRIMARY]

       --Get the nearest SIDEWALK_ID (or PIND, etc.) to your points:

```
SELECT a.{your_table_unique_id},
(select top(1) b.sidewalk_id from sdwkapp_sidewalks as b
order by b.shape.STDistance(a.Shape)) as SIDEWALK_ID
FROM {your_table_here} as a
where a.{your_table_unique_id} = 9999 -- include 'where' clause to get one at a
time, replacing 9999 with desired id.
```

7. Blob Storage
    a. Blob storage is necessary for saving the thousands of Images coming from the rover such as the front, rear, and rendered Images. An account recording the rovers data, holds an image container, which records a link to the blob storage holding all image files.
    b. Azure Storage Explorer Connection Guide

1. Navigate to
https://azure.microsoft.com/en-us/features/storage-explorer/

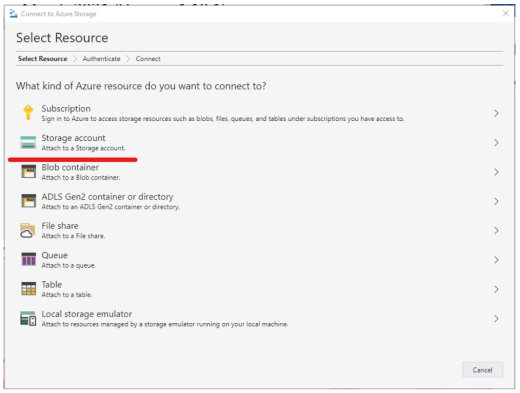
and download Azure Storage Explorer.

2. Run StorageExplorer.exe to install the application.

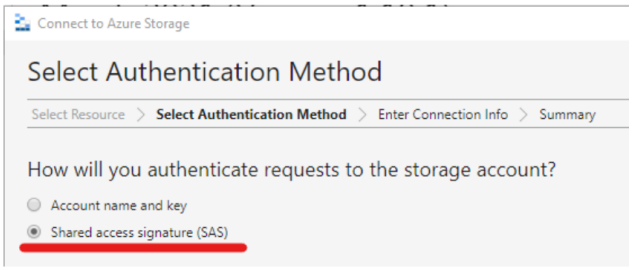
3. Launch Azure Storage Explorer.

4. Click on the “Open Connect Dialog” icon.

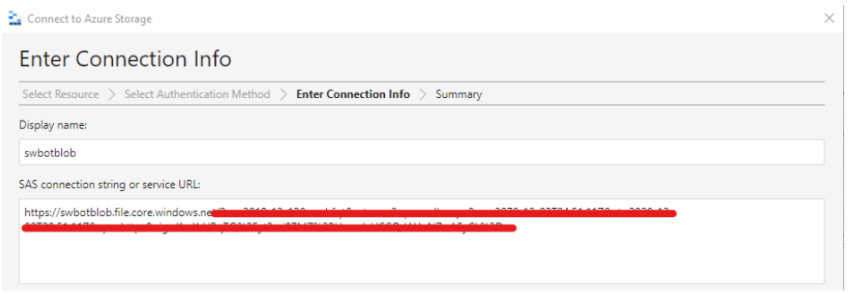
5. Click on Storage Account



6. Select Shared access signature (SAS)

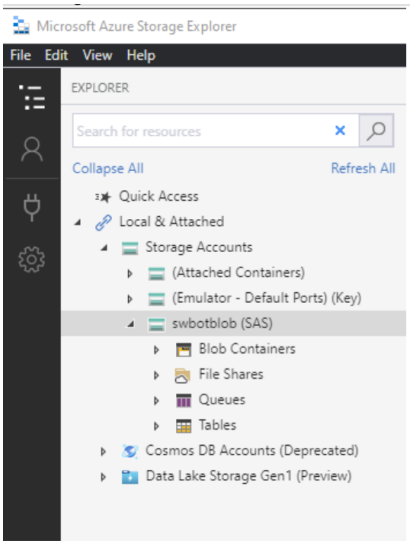


7. Paste the SAS connection string that is shared with you. The display name should automatically populate. (SAS connection string is to be provided by BOE).



8. Click connect on the summary page.

9. The storage container should now show on the left-hand side.



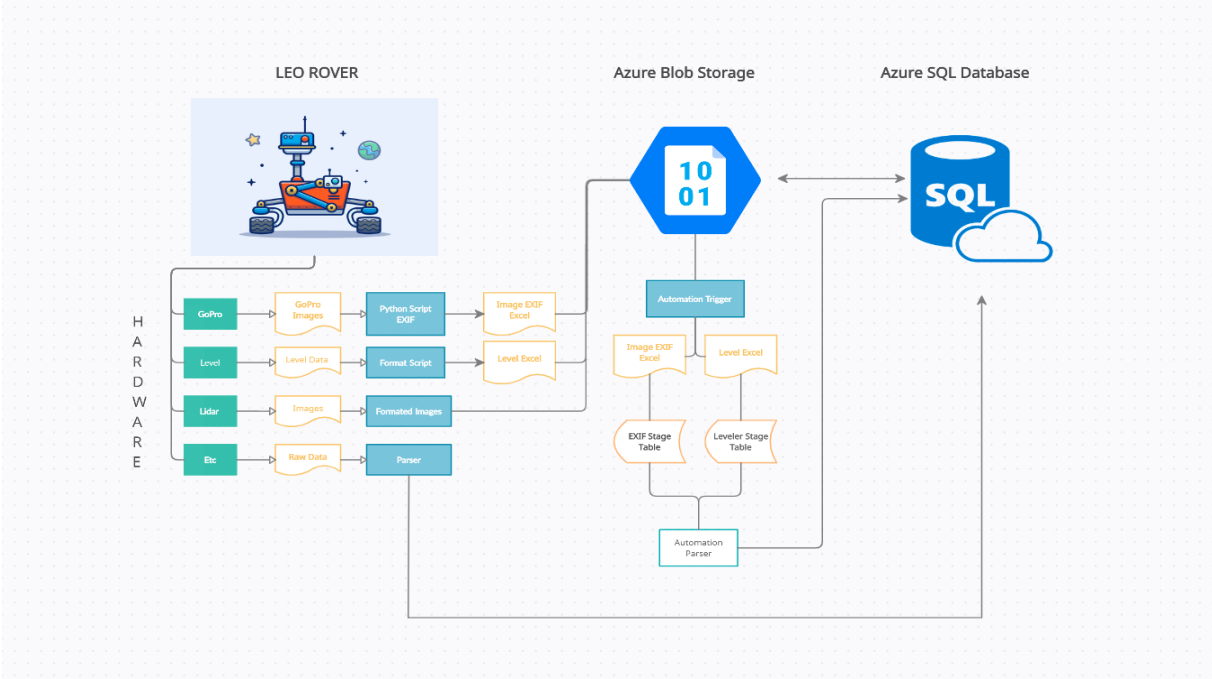
8. Containers

   The Blob storage entails two containers which are the Images and the CSV information.

9. Images/CSV
   a. CSV table GoProImage Data with links to images in Blob Storage

	LatitudeRef	Latitude	LongitudeRef	Longitude	ImageType	Timestamps	Dates	Orientation	PinID	ImageID	UrlFront	UrlBack	UrlRendered
1	N	34.0399319	W	118.2632594	GoPro	2020-10-15 15:03:00.00000…	2020:10:15	DCIM\100GFRNT\GPFR1772.JPG	NULL	DCIM\100GFRNT\GPFR1772.JPG	NULL	NULL	NULL
2	NULL	4.00625e+006	NULL	-1.31625e+007	GoPro	2021:03:12	2021-03-12 11:46:00.00000…	DCIM\100GFRNT\GPFR1888.JPG	NULL	GPFR1888.JPG	NULL	NULL	NULL
3	NULL	4.00625e+006	NULL	-1.31626e+007	GoPro	2021:03:12	2021-03-12 11:47:00.00000…	DCIM\100GFRNT\GPFR1889.JPG	NULL	GPFR1889.JPG	NULL	NULL	NULL
4	NULL	4.00625e+006	NULL	-1.31626e+007	GoPro	2021:03:12	2021-03-12 11:48:00.00000…	DCIM\100GFRNT\GPFR1890.JPG	NULL	GPFR1890.JPG	NULL	NULL	NULL
5	NULL	4.00625e+006	NULL	-1.31628e+007	GoPro	2021:03:12	2021-03-12 11:50:00.00000…	DCIM\100GFRNT\GPFR1891.JPG	NULL	GPFR1891.JPG	NULL	NULL	NULL
6	NULL	4.00628e+006	NULL	-1.31629e+007	GoPro	2021:03:12	2021-03-12 10:54:00.00000…	DCIM\100GFRNT\GPFR1871.JPG	NULL	GPFR1871.JPG	NULL	NULL	NULL
7	NULL	4.00639e+006	NULL	-1.31623e+007	GoPro	2021:03:12	2021-03-12 11:43:00.00000…	DCIM\100GFRNT\GPFR1887.JPG	NULL	GPFR1887.JPG	NULL	NULL	NULL
8	NULL	4.00648e+006	NULL	-1.31623e+007	GoPro	2021:03:12	2021-03-12 11:41:00.00000…	DCIM\100GFRNT\GPFR1885.JPG	NULL	GPFR1885.JPG	NULL	NULL	NULL
9	NULL	4.0064e+006	NULL	-1.31623e+007	GoPro	2021:03:12	2021-03-12 11:42:00.00000…	DCIM\100GFRNT\GPFR1886.JPG	NULL	GPFR1886.JPG	NULL	NULL	NULL
1…	NULL	4.00656e+006	NULL	1.31623e+007	GoPro	2021:03:12	2021-03-12 11:40:00.00000…	DCIM\100GFRNT\GPFR1884.JPG	NULL	GPFR1884.JPG	NULL	NULL	NULL
1…	NULL	4.00665e+006	NULL	-1.31622e+007	GoPro	2021:03:12	2021-03-12 11:38:00.00000…	DCIM\100GFRNT\GPFR1883.JPG	NULL	GPFR1883.JPG	NULL	NULL	NULL
1…	NULL	4.00672e+006	NULL	-1.31623e+007	GoPro	2021:03:12	2021-03-12 11:36:00.00000…	DCIM\100GFRNT\GPFR1882.JPG	NULL	GPFR1882.JPG	NULL	NULL	NULL
1…	NULL	4.00674e+006	NULL	-1.31624e+007	GoPro	2021:03:12	2021-03-12 11:34:00.00000…	DCIM\100GFRNT\GPFR1881.JPG	NULL	GPFR1881.JPG	NULL	NULL	NULL
1…	NULL	4.00678e+006	NULL	-1.31625e+007	GoPro	2021:03:12	2021-03-12 11:32:00.00000…	DCIM\100GFRNT\GPFR1880.JPG	NULL	GPFR1880.JPG	NULL	NULL	NULL
1…	NULL	4.00687e+006	NULL	-1.31628e+007	GoPro	2021:03:12	2021-03-12 11:22:00.00000…	DCIM\100GFRNT\GPFR1878.JPG	NULL	GPFR1878.JPG	NULL	NULL	NULL
1…	NULL	4.00687e+006	NULL	1.31627e+007	GoPro	2021:03:12	2021-03-12 11:24:00.00000…	DCIM\100GFRNT\GPFR1879.JPG	NULL	GPFR1879.JPG	NULL	NULL	NULL

10. Automation
    a. Rover collects data, data is saved in CSV format, csv is saved within Azure Blob CSV container, a trigger grabs the data from the csv and creates temporary staging tables within the sql DB with the data from the csv which we can then use within a View to transfer our data from the staging tables to our SQL tables.



11. Liaisons - POC
    a. Fabiola Desantiago
        i. fabiola.desantiago@lacity.org
        ii. Contact for Azure connection access and inquiries.
    b. Irvin Nguyen
        i. irvin.nguyen@lacity.org
        ii. Contact to schedule meetings and connect with other points of interest
    c. Omar Haikal
        i. omar.haikal@lacity.org
        ii. Contact for Azure SQL and automation inquiries.
    d. Miguel Grajeda
        i. miguel.grajeda@lacity.org
        ii. Contact for Navigate LA training and inquiries.
    e. Nathan Neumann
        i. nathan.neumann@lacity.org
        ii. Contact for Azure SQL and Navigate LA inquiries.
    f. Alexis Pena
        i. alexis.pena020@gmail.com
        ii. Senior Design Database Lead
    g. Abigail Garcia
        i. garciaabigail23@gmail.com
        ii. Senior Design Database team member