Unit 3 - Microsoft SQL Server

Design and Explore Geospatial DBs

Shubhangi Singhal (ss100)

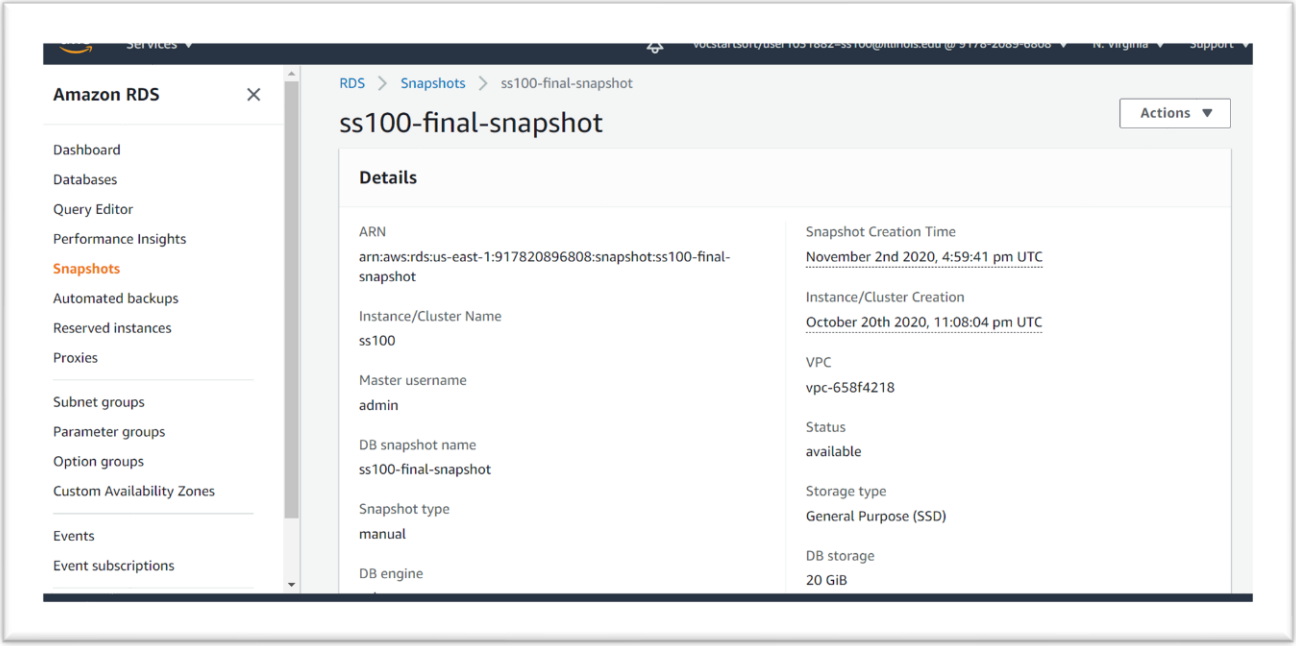
Srishti Rawat (rawat4)

Zainab Zaveri (zainaba2)

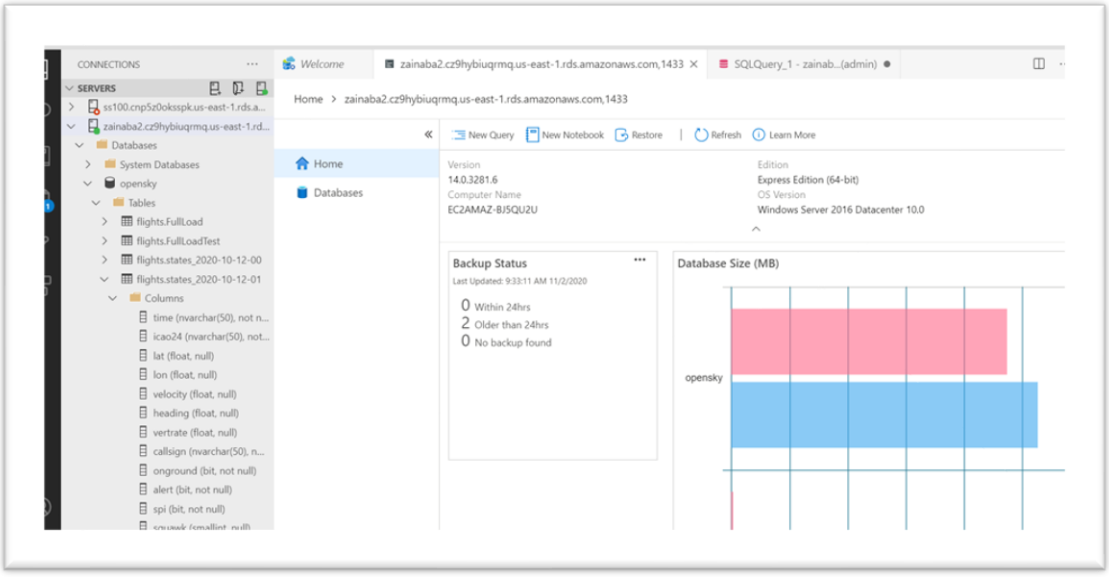
Microsoft SQL Server is a relational database management system developed by Microsoft. As a database server, it is a software product with the primary function of storing and retrieving data as requested by other software applications —which may run either on the same computer or on another computer across a network.

For solving the assignments of Unit 3, we researched the requirements, distributed work for the experiments and collaborated on execution of tasks.

1. We created an Amazon Web Services RDS database instance using MS SQL server. For which we made changes to the inbound rules and configuration settings, allowing our public IP addresses to access the instance, and blocking out the other traffic as our security policy. We also modified network and security settings.

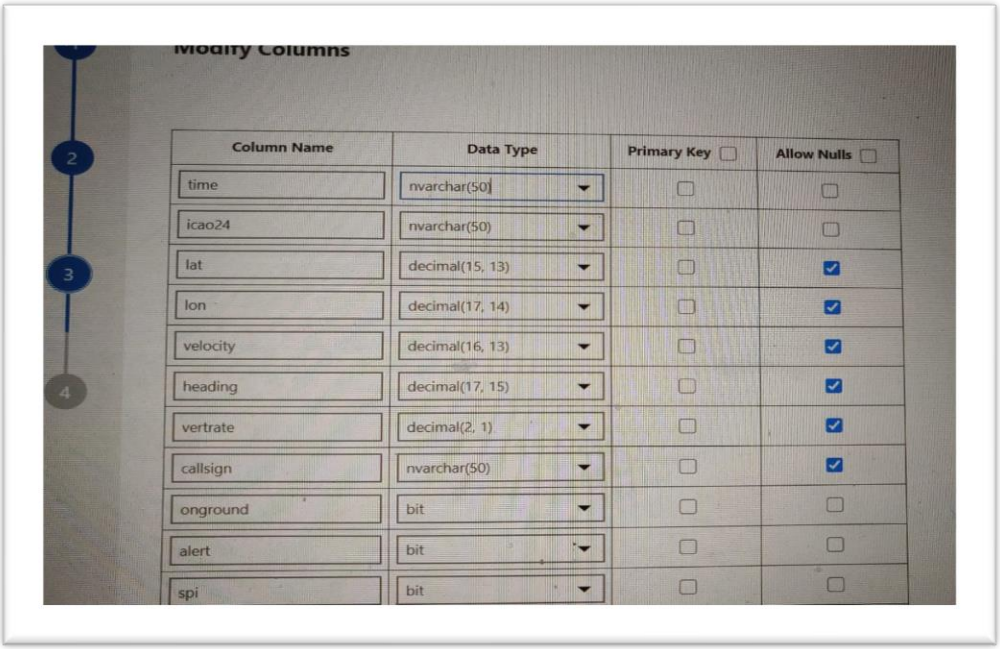


2. Our choice of client for connecting to the server is Microsoft Azure Data Studio



3. We connected all three of our workstations to the RDS instance using port number 1433, connection type - MS SQL server, auth type as SQL login, endpoint and inputting the database credentials.

4. We then decided to use the Import Wizard Extension available in Azure Data Studio to import the OpenSky datasets. Where each of us loaded 8 .csv files. During the process we created a database called **opensky** and schema by the name of **flights** using extension and SQL query editor.

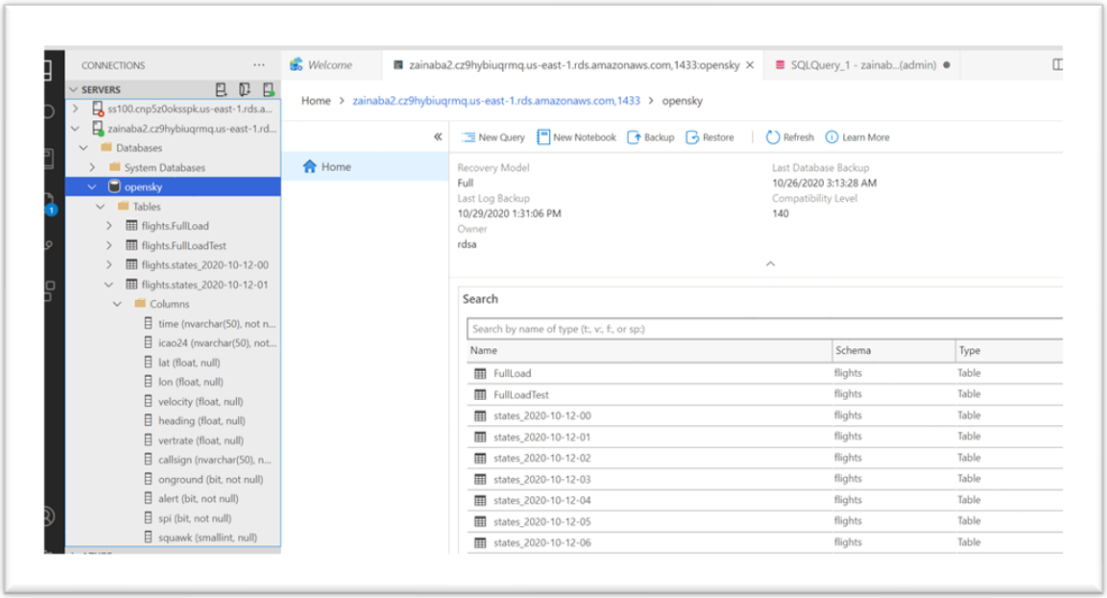


5. SQL for creating database Opensky

   CREATE DATABASE Opensky;

   GO

   Output



6. Each CSV file took about 4 to 5 minutes to get uploaded both on Mac and Windows laptops. We changed the datatype of the Time Column from int to varchar during the import process for all the files.
7. Later we appended individual tables together and stored them in another table called FullLoad using the INTO and UNION commands. Time taken to run this query - 9 minutes.

SQL query to append tables

```
select *

into [flights].[FullLoad]

from(

select * from [flights].[states_2020-10-12-00]

union all

select * from [flights].[states_2020-10-12-01]
```

union all

select * from [flights].[states_2020-10-12-02]

union all

select * from [flights].[states_2020-10-12-03]

union all

select * from [flights].[states_2020-10-12-04]

union all

select * from [flights].[states_2020-10-12-05]

union all

select * from [flights].[states_2020-10-12-06]

union all

select * from [flights].[states_2020-10-12-07]

union all

select * from [flights].[states_2020-10-12-08]

union all

select * from [flights].[states_2020-10-12-09]

union all

select * from [flights].[states_2020-10-12-10]

union all

select * from [flights].[states_2020-10-12-11]

union all

select * from [flights].[states_2020-10-12-12]

union all

select * from [flights].[states_2020-10-12-13]

union all

select * from [flights].[states_2020-10-12-14]

union all

select * from [flights].[states_2020-10-12-15]

union all

select * from [flights].[states_2020-10-12-16]

union all

select * from [flights].[states_2020-10-12-17]

union all

select * from [flights].[states_2020-10-12-18]

union all

select * from [flights].[states_2020-10-12-19]

union all

select * from [flights].[states_2020-10-12-20]

union all

select * from [flights].[states_2020-10-12-21]

union all

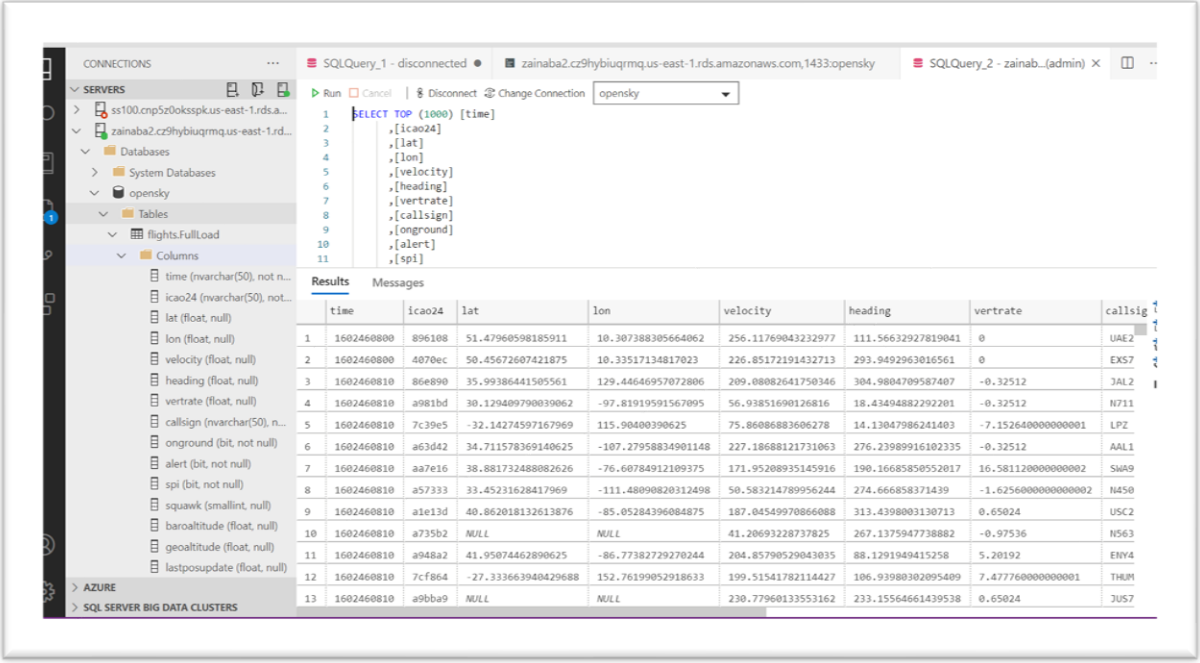
select * from [flights].[states_2020-10-12-22]

union all

select * from [flights].[states_2020-10-12-23]

)t

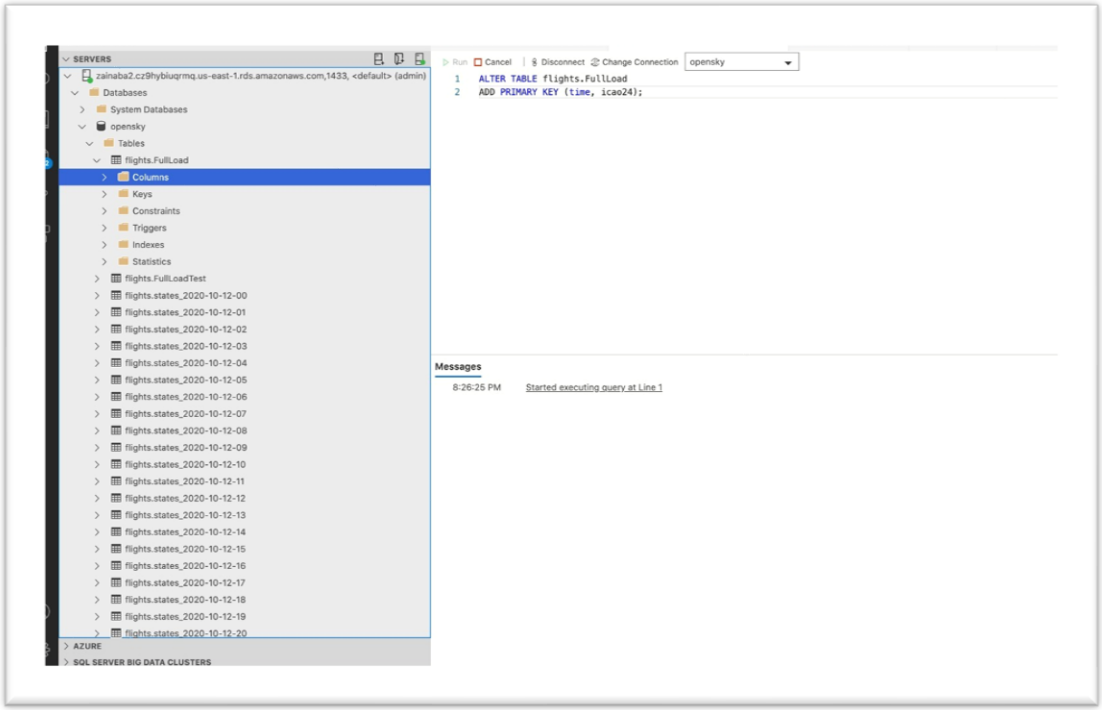
Output



	time	icao24	lat	lon	velocity	heading	vertrate	callsig
1	1602460800	896108	51.479605981859 11	10.307388305664062	256.11769043232977	111.56632927819041	0	UAE2
2	1602460800	4070ec	50.45672607421875	10.33517134817023	226.85172191432713	293.9492963016561	0	EXS7
3	1602460810	86e890	35.99386441505561	129.44646957072806	209.08082641750346	304.9804709587407	-0.32512	JAL2
4	1602460810	a981bd	30.129409790039062	-97.81919591567095	56.93851690126816	18.43494882292201	-0.32512	N711
5	1602460810	7c39e5	-32.14274597167969	115.90400390625	75.86086883606278	14.13047986241403	-7.152640000000001	LPZ
6	1602460810	a63d42	34.711578369140625	-107.27958834901148	227.18688121731063	276.23989916102335	-0.32512	AAL1
7	1602460810	aa7e16	38.881732488082626	-76.60784912109375	171.95208935145916	190.16685850552017	16.581120000000002	SWA9
8	1602460810	a57333	33.45231628417969	-111.48090820312498	50.583214789956244	274.666858371439	-1.6256000000000002	N450
9	1602460810	a1e13d	40.862018132613876	-85.05284396084875	187.04549970866088	313.4398003130713	0.65024	USC2
10	1602460810	a735b2	NULL	NULL	41.20693228737825	267.13759477388 82	-0.97536	N563
11	1602460810	a948a2	41.95074462890625	-86.77382729270244	204.85790529043035	88.1291949415258	5.20192	ENY4
12	1602460810	7cf864	-27.333663940429688	152.76199052918633	199.51541782114427	106.93980302095409	7.477760000000001	THUM
13	1602460810	a9bba9	NULL	NULL	230.77960133553162	233.15564661439538	0.65024	JUS7

8. We decided to keep our primary key as a combination of **time** and **icao24** since for a single transponder ID the data values would be unique at a given point of time.

SQL to set the primary key



Geospatial Queries

1. SELECT COUNT(*) FROM flights.FullLoad WHERE icao24='a0d724' AND time>=1480760100 AND time<=1480764600;

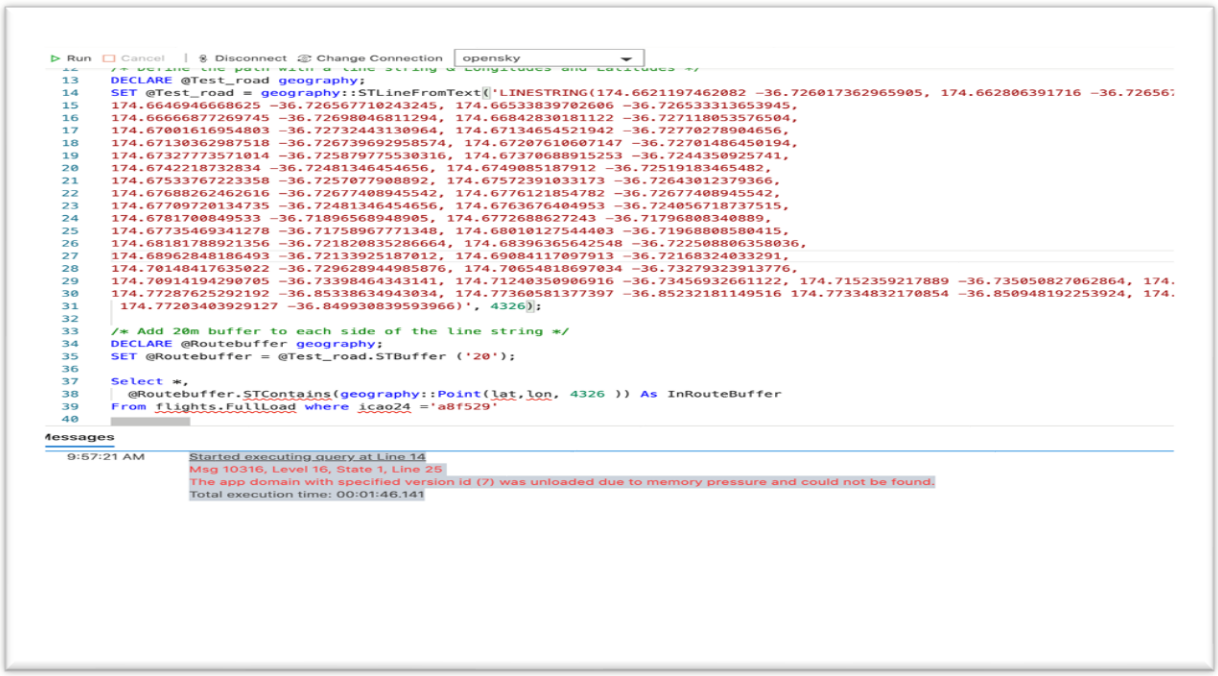
To find how many rows with the transponder ID (a0d724) have been received on Saturday, December 3rd between 10:15 and 11:30 UTC

2. SELECT COUNT(DISTINCT icao24) FROM flights.FullLoad WHERE lat<=50.07 AND lat>=49.98 AND lon<=8.62 AND lon>=8.48 AND time=1493892000;

To count the aircrafts which arrived or departed or crossed Frankfurt during a certain time.

3. We tried to write a geospatial query to create a line type object using latitude and longitudes of a particular route. Then we created a buffer to account for 20 meters in and around the points in the path. We finally tried to find the lat, lon points for a particular icao24 number to see if for that flight, the points followed the path or not. The number

4326 in the query is an SRID(spatial reference identifier) is a unique identifier associated with a specific coordinate system, tolerance, and resolution. How the SRID is populated or what it represents can vary depending on what database you use to store your data.



4.We also tried to create Geom object types from point type. The point took input of one lat, lon pair and the SRID. We first calculated the area of a circle around the location of a particular flight. Next, we stuffed it with a buffer of 2000 meters. We did the same for another flight's locations. Finally, we tried to write a query to get the intersection between the two areas to see if two flights collide with each other or not.

```
/** area around a flight **/

/* Area of Circle */

DECLARE @Location_of_flight geography;
SET @Location_of_flight = geography::STGeomFromText ('POINT(174.77203403929127 -36.849930839593966)', 4326);

/* Add 2000m buffer to the point */
DECLARE @Buffered_Point geography;
SET @Buffered_Point = @Location_of_flight.STBuffer ('2000');

SELECT @Buffered_Point.STArea () AS AREA_around_flight

DECLARE @Location_of_flight2 geography;
SET @Location_of_flight2 = geography::STGeomFromText ('POINT(200.00006544323 -45.786560839593966)', 4326);

/* Add 2000m buffer to the point */
DECLARE @Buffered_Point2 geography;
SET @Buffered_Point2 = @Location_of_flight2.STBuffer ('2000');

SELECT @Buffered_Point2.STArea () AS AREA_around_flight2

DECLARE @Total_area_intersection
geography = @Buffered_Point.STDifference(@Buffered_Point2);
SELECT @Total_area_intersection.STArea () AS total;
```

Challenges Faced and How We Collaborated

The first few challenges we faced was to figure out how to get started with Azure and loading data onto our SQL server. In the beginning, we began by loading each table of our data separately while thinking of ways to load it in bulk. Since it did not take us very long, approximately 4-5 minutes for each table, we never really had to think about bulk loading data. Something for us to work on in the future. We loaded our data successfully. As a comment on the performance, it was evidently good, considering we are talking about millions of rows of data.

Second, since we also encountered log issues. It was full after we ran a few queries and later learned that it had something to do with the setting in azure for our data server instance.

Third, we did not configure any primary key while loading data in the beginning, we tried loading it later. Our query ran for more than 49 hours. This could have been set earlier and that would have been faster.

Overall, geospatial queries looked daunting to us as a concept. We got over that fast. While implementing them, we did take a few moments to understand its application. With the help of

proper resources, we successfully went ahead and got through this challenge as well. As a team, we were continuously in communication with each other and arranged to meet over zoom meetings whenever we felt the need.

References

1. **https://docs.microsoft.com/en-us/sql/azure-data-studio/extensions/sql-server-import-extension?view=sql-server-ver15**
2. **https://www.freecodecamp.org/news/cjn-how-to-connect-your-aws-rds-microsoft-sql-server-using-azure-data-studio/**
3. **https://docs.microsoft.com/en-us/azure/dms/known-issues-troubleshooting-dms-source-connectivity**
4. **https://docs.microsoft.com/en-us/sql/relational-databases/spatial/spatial-data-sql-server?view=sql-server-ver15**
5. **https://www.aws.training/Details/Video?id=16382**
6. **https://database.guide/create-a-sql-server-database-with-azure-data-studio/**