**Azure SQL Database Notes:**

## **Configuring authentication**

SQL Database supports two types of authentications, SQL Authentication and Azure Active Directory Authentication (Azure AD Authentication).

### **SQL Authentication**

With SQL Authentication, when you create a SQL Database, you also create a login that is the server-level principal account for your SQL Database server. The login is analogous to the SA login for on-premises SQL Server. This principal account manages all server- and database-level security and allows the creation of other accounts to manage logins and databases in SQL Database.

### **Azure Active Directory Authentication**

Azure AD authentication uses identities managed by Azure Active Directory and is supported for managed and integrated domains. To use Azure AD authentication, you must create a second server-level principal account called “Azure AD Admin” to administer Azure AD users and groups. This admin can also perform all operations the regular (SQL) SA can.

With Azure AD authentication, Azure SQL Database extends existing authentication mechanism allowing Azure AD users to login to the database using three methods:

* Principal name/password
  + Works for Azure AD managed and federated domains
  + The easiest way to adopt Azure AD Authentication in existing applications
* Integrated Windows Authentication
  + Works for Azure AD federated domains and clients on domain-joined machines
  + Eliminates storing password and enables single sign-on
* Token-based authentication (will be released later during public preview)
  + Gives application full control over access token acquisition
  + Enables authentication using certificates

**Managing Permissions**

**Server-level roles**

While you can use the server-level principal account to manage server-level security, you also have the option to assign logins to other SQL Database security roles. Simply use the ***login manager***role to grant permission to create logins (similar to the ***securityadmin***role in an on-premises instance of SQL Server). The ***dB manager***role is used to create databases in the server and is comparable to the *dB creator* role in on-premises SQL Server (using the ***database manager***role to grant permission to create a database).

Because these roles are powerful, you should assign them only to a limited set of administrative personnel and not generically to user accounts.

You can also create these roles within the master database. The roles will scope to the server, so a user with permissions in the master database also has access to data in any SQL Database on that server through server-scoped dynamic management views (DMVs).

**Database-level roles**

The built-in security roles at the database level are similar to on-premises SQL Server security roles. You can implement database-level security by using fixed database roles (such as ***db\_datareader***or ***db\_datawriter***), or you can create custom roles for your application to grant explicit permissions to selected database objects. The use of role-based security for database access is considered a best practice

for all databases. For more information about role-based security best practices, visit [Server and Database](https://msdn.microsoft.com/en-us/library/bb669065(v%3Dvs.110).aspx) [Roles in SQL Server.](https://msdn.microsoft.com/en-us/library/bb669065(v%3Dvs.110).aspx)

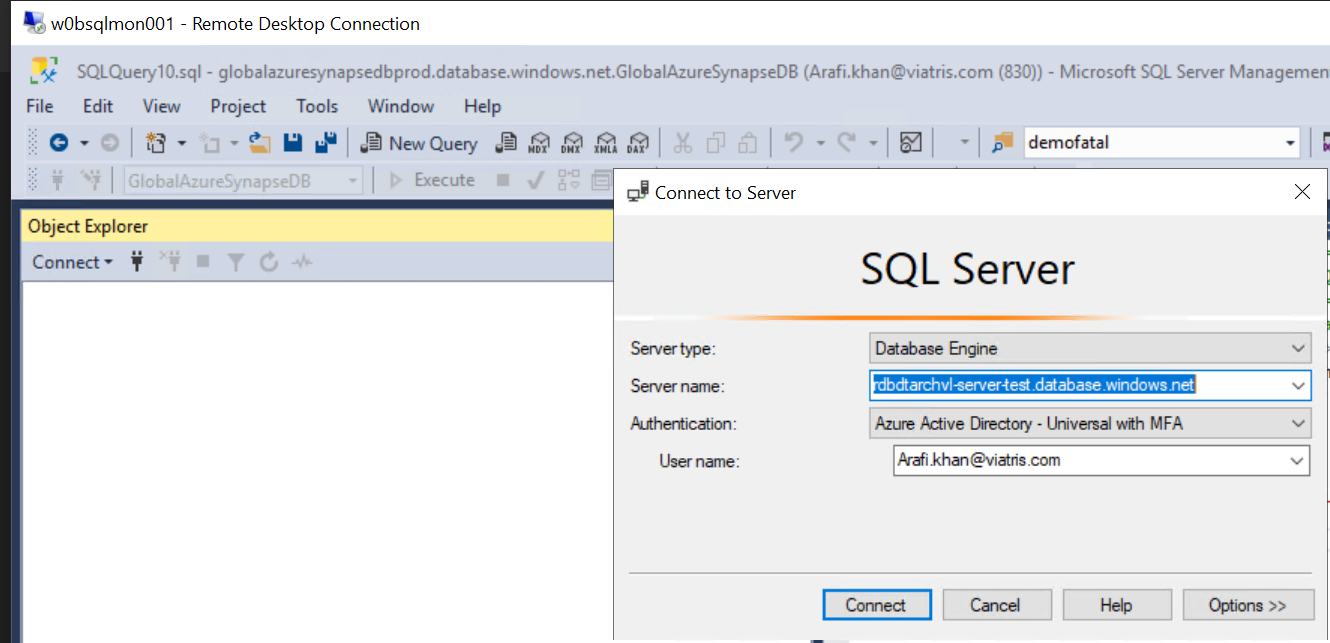
How to connect Azure Database in SSMS:

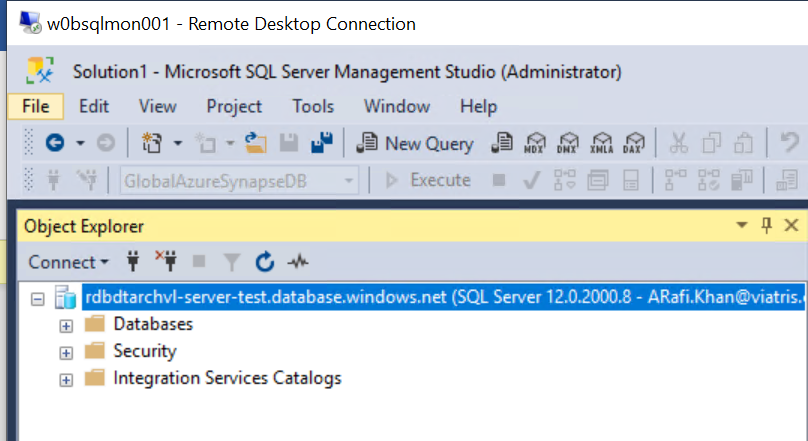
Step1: server name: “ rdbdtarchvl-server-test.database.windows.net”

Step2: Authentication: azure active Directory-Universal with MFA

Step3: User name: [yourmailid@Viaaaaa.com](mailto:yourmailid@Viaaaaa.com).

Step4: Connect.

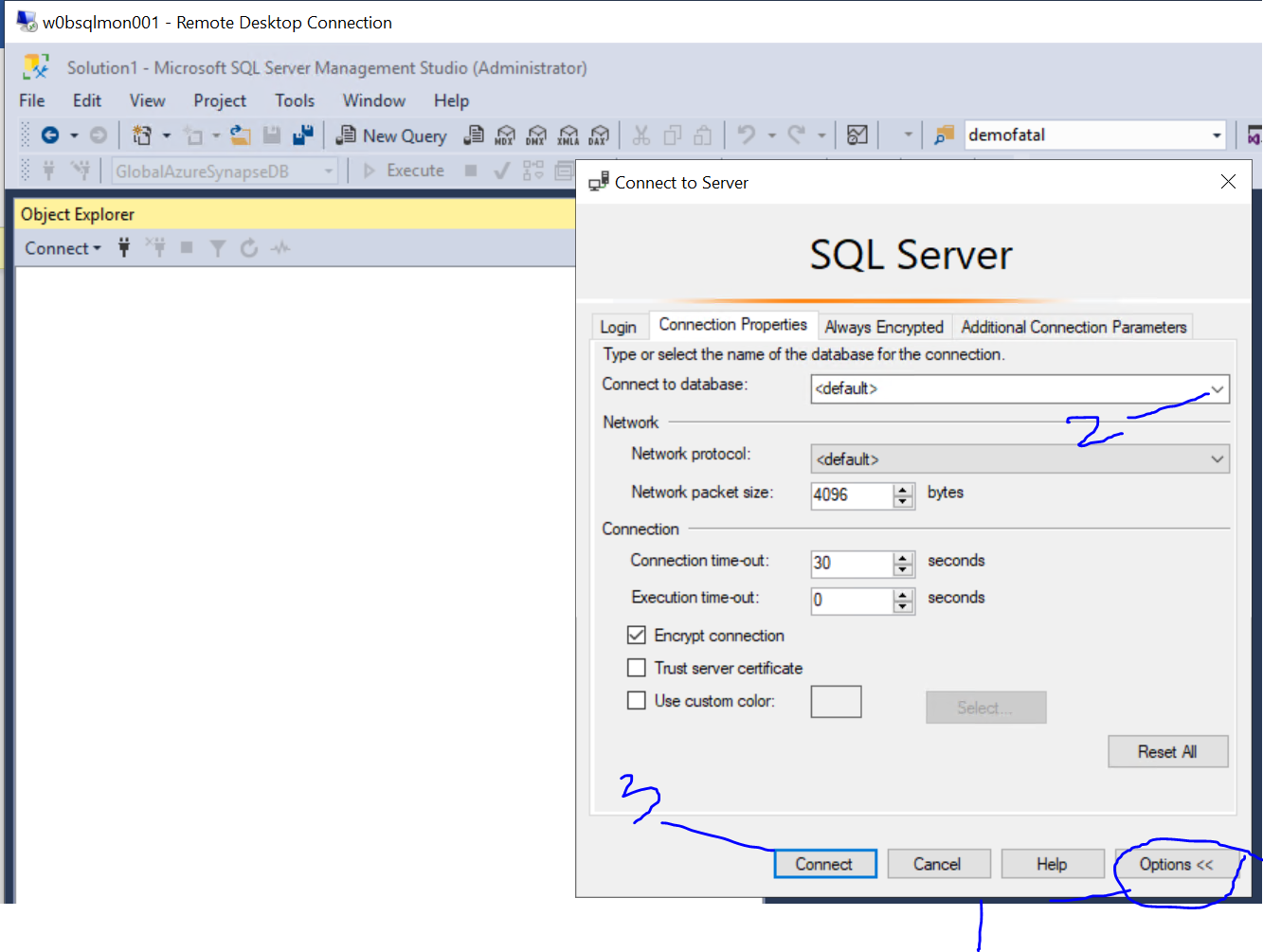




If get error, try below steps:

* Select Option
* Under step2 select Database name
* Connect

Refer below Screen shot:

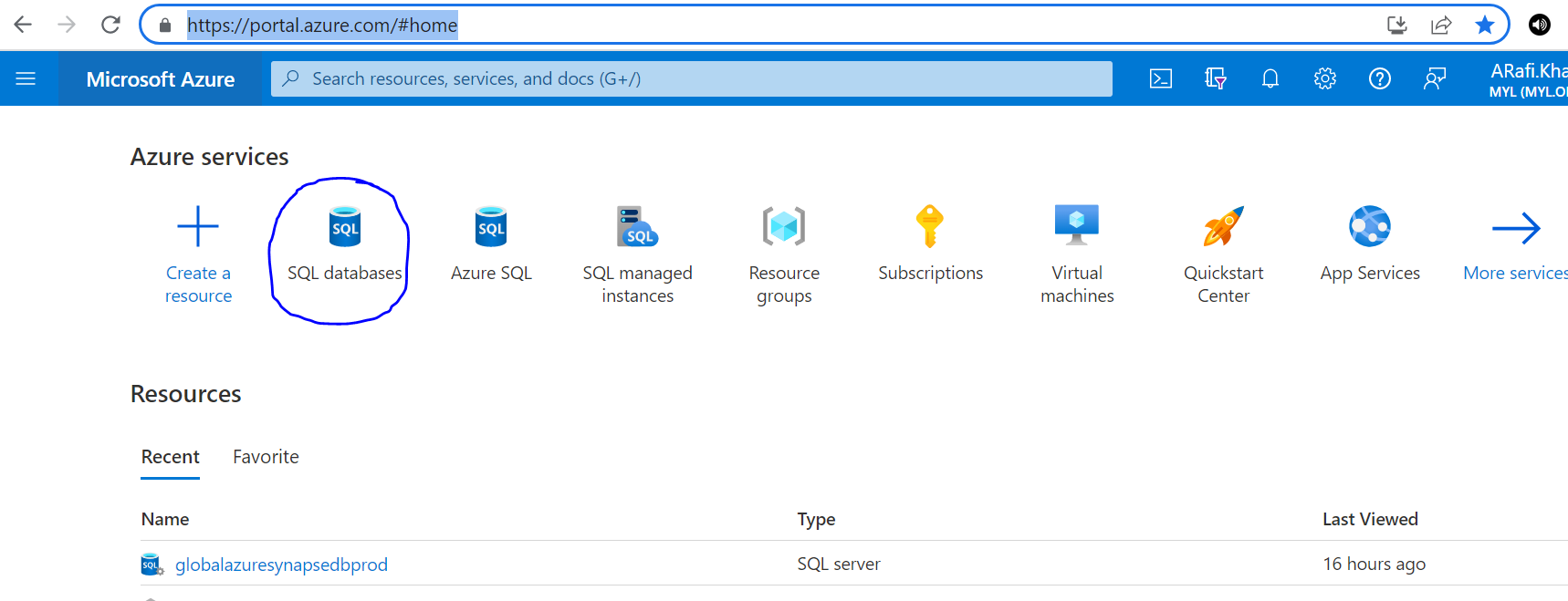


How to verify Azure instance available or not:

Connect Azure Portal:

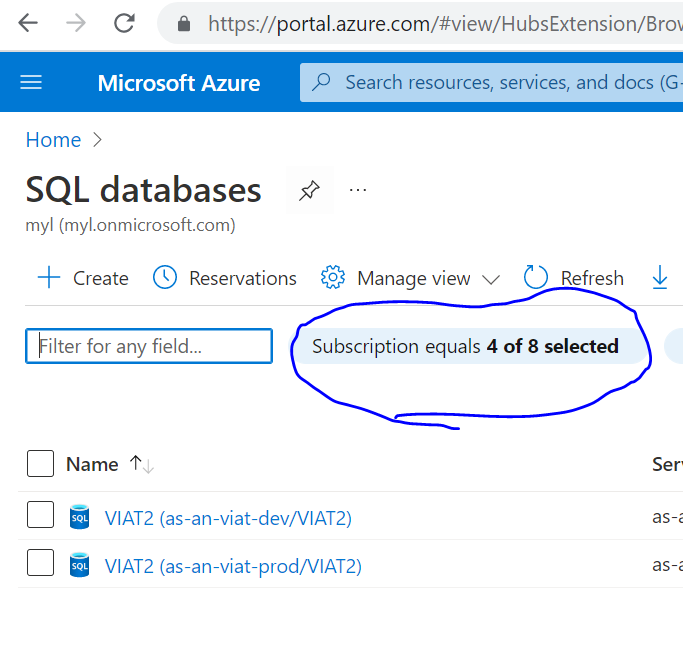
<https://portal.azure.com/#home>

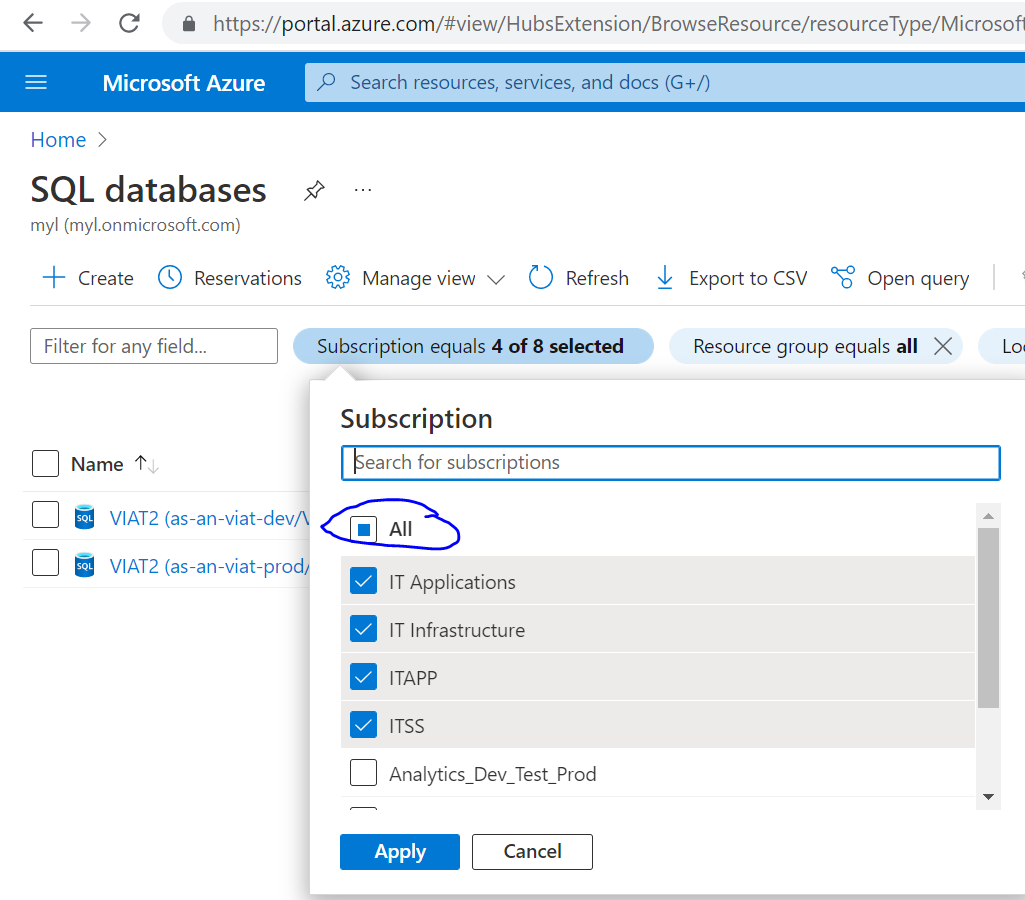
select SQL Database



* Select subscription equals
* Select all
* Apply

Refer below ss

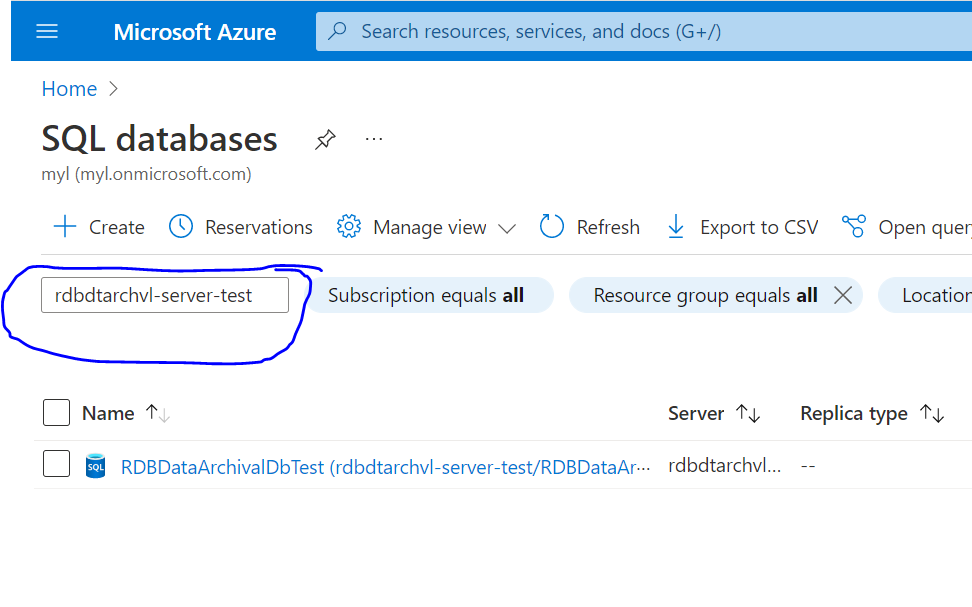
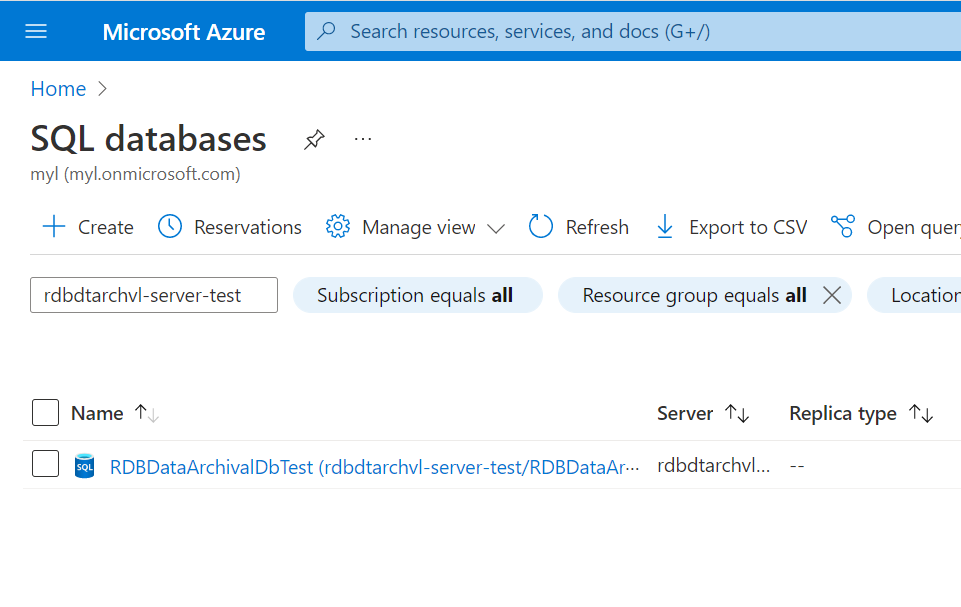




**Under Highlighted one type of server name like below Don’t type full server name:**

**Full name:** rdbdtarchvl-server-test.database.windows.net

**Search only highlighted one is Instance name:** rdbdtarchvl-server-test

**Database level access:**

**User Create command:**

* create user [user\_name] from external provider

**Example:**

* create user Analytics\_NorthAmerica\_Sales\_CanadaContributorMgr from external provider
* sp\_addrolemember 'db\_datareader', 'Analytics\_NorthAmerica\_Sales\_CanadaContributorMgr'

**User access on Database level:**

**Command:**

sp\_addrolemember 'db\_datawriter', 'user\_name'

**Example:**

* sp\_addrolemember 'db\_datawriter', 'NorthAmericaAdfTest'
* sp\_addrolemember 'db\_datareader', 'NorthAmericaAdfTest'
* sp\_addrolemember 'db\_ddladmin', 'NorthAmericaAdfTest' --> to delete the temporary staging tables.

**SQL authentication login creation command:**

CREATE LOGIN “Login\_name ” WITH PASSWORD = <Login\_password >;

CREATE USER “User\_name ” FOR LOGIN “Login\_name ”;

Grant specific rights on schema to AD group:

**Type of Schema access:**

1. CONTROL
2. ALTER
3. SELECT
4. View DEFINITION
5. INSERT
6. UPDATE
7. DELETE
8. EXECUTE ON

**Command to grant access:**

GRANT Access\_name SCHEMA::shcema\_name TO user\_name ;

**Example:**

GRANT CONTROL, ALTER, SELECT, View DEFINITION, INSERT, UPDATE, DELETE, EXECUTE ON SCHEMA::emex TO Analytics\_UAIT\_Developers;

**Grant specific rights on database to AD group:**

ALTER ANY EXTERNAL FILE FORMAT,

ALTER ANY EXTERNAL DATA SOURCE,

ALTER ANY DATASPACE, CREATE TABLE,

CREATE PROCEDURE,

CREATE FUNCTION,

CREATE VIEW

**Command:**

GRANT Access\_type TO AD\_Group\_name ;

**Example:**

GRANT ALTER ANY EXTERNAL FILE FORMAT, ALTER ANY EXTERNAL DATA SOURCE, ALTER ANY DATASPACE, CREATE TABLE, CREATE PROCEDURE, CREATE FUNCTION, CREATE VIEW TO Analytics\_UAIT\_Developers;

**To restrict the** resource classes(smallrc, mediumrc, largerc, xlargerc **level:**

**Revoke alter any role from [AG-ANLY\_ADG\_SUPT\_DE]**

**Note:** Who is having the control access they can modify the resource class, so we need to “revoke alter any role”

**To check the permission user level:**

----user name 'aaduser1@microsoft.com’ change the User name to check details.

SELECT prm.permission\_name

   , prm.class\_desc

   , prm.state\_desc

   , p2.name as 'Database role'

   , p3.name as 'Additional database role'

FROM sys.database\_principals AS p

JOIN sys.database\_permissions AS prm

   ON p.principal\_id = prm.grantee\_principal\_id

   LEFT JOIN sys.database\_principals AS p2

   ON prm.major\_id = p2.principal\_id

   LEFT JOIN sys.database\_role\_members r

   ON p.principal\_id = r.member\_principal\_id

   LEFT JOIN sys.database\_principals AS p3

   ON r.role\_principal\_id = p3.principal\_id

WHERE p.name = 'aaduser1@microsoft.com';

-- Get current users with assigned roles in DB level

SELECT usrs.name                               as UserName,

              usrs.principal\_id         as UserPrincipalId,

              usrs.type                                as UserType,

              usrs.type\_desc                       as UserDesc,

              usrs.sid                                              as UserSID,

              usrs.authentication\_type\_desc as UserAuthTypeDesc,

              roles.name                             as RoleName,

              roles.type\_desc                      as RoleTypeDesc,

              roles.is\_fixed\_role  as RoleFixed,

              roles.principal\_id         as RolePrincipalId

FROM sys.database\_principals AS usrs

left join sys.database\_role\_members AS assign

           on usrs.principal\_id = assign.member\_principal\_id

left join sys.database\_principals as roles

           on assign.role\_principal\_id = roles.principal\_id

  and roles.type = 'R'

WHERE usrs.type <> 'R'

order by 1;

Get current users with assigned roles in Schema level

SELECT pr.principal\_id, pr.name, pr.type\_desc,

    pr.authentication\_type\_desc, pe.state\_desc, pe.permission\_name, pe.class\_desc, pe.major\_id, s.[name]

FROM sys.database\_principals AS pr

JOIN sys.database\_permissions AS pe

   ON pe.grantee\_principal\_id = pr.principal\_id

LEFT JOIN sys.schemas s

   ON pe.major\_id = s.schema\_id

WHERE pr.name not in ('public', 'dbo')

 AND pe.permission\_name <> 'CONNECT'

order by 1;