AZURE DATA ENGINEERING

Difference between Data Engineer/Science vs DevOps vs Analyst:

Data Engineer:
Moving all data from different sources to one possible location(using data pipelines)—(aggregation, migration –(ADF), analytics(using spark or DB))
work: create pipelines, monitor, debug, analytical jobs

Data Science:
Training models based on the data(clean data), mostly works on prediction and forecasting, AI/ML

DevOps:
CI/CD(continuous implementation and continuous development), they create DevOps pipelines moving code from one environment to another environment (dev to UAT(testing))

Data Analyst:
Creating reports based on the data using power bi, tableau whatever tool is available

ON-Premises – owning own server or machine rather than working through cloud,, the problem is we need to buy our own service machines which is not very cost effective for an organization.

Cloud :
It provides services(storage,software) and resources(compute power) over the internet

Services:
IAAS: infrastructure as a service taking entire machine ex: virtual machine, aws, azure
Virtual Machine: Machine that you create on cloud.
PAAS: platform as a service, we don't worry about machine I just need one platform to code, develop and deploy my program ex: google app engine, Heroku
SAAS: software as a service, provide software via the internet ex: microsoft365,gmail

Advantages:
Cost saving:
cost-effective for organization just rent it and use whenever required and pay only for amount of time you use it, which is much better than owning it
Scalability(increase or decrease at any time):
vertical: increasing the power of same machine ex: if it was 16gb, making it it 32gb
Horizantal: increasing the number of machines in horizontally
High availability:
you will have the immediate backup to your servers you have rented if anything wents down then you will get immediate backup server, nothing is going to happen your data its fucking their own cloud service provider problem
Highly agile:
Resource allocation is very fast, which speed up the whole development process

Types:
Public: anybody can use it(creating account and using cloud) ex: google cloud, azure, aws
Private: many big banking try to have their own private cloud it is most secured type. Ex: IBM cloud

Hybrid: some companies opt for hybrid where they store most critical data in private and less critical in public which does makes sense
we don't fucking care about which kind of cloud type it is we are using it is a problem of infrastructure team or admin team.

you can manage the azure cloud using UI(Portal – mostly we use), power shell/ bash(command way), rest Api(cloud developers), mobile app(just checking status)

Azure Storage Account:

An Azure Storage Account is a cloud storage service provided by Microsoft Azure that allows you to store data and access it from anywhere in the world.

Subscription:

can be on team basis, like HR, operations, sales, and market or can be on environment base as well dev, uat,prod one project can have multiple subscriptions but one azure account may be the organization.

Resource group:

it is a logical separation so we can identify the particular resource which group it belongs to.

Storage account naming convention:

storage account names should be unique since it was used in URL that's why it always recommended to use small letters and numbers no special characters or capital letters are allowed.

Redundancy level:(Duplication of data):

Local redundant: In same city the duplicate servers are available
Zone redundant: In same Zone(East or west) the duplicate servers(3) are available
Zeo redundant: In same region(India) the data is present at different zones(east,west,north,south)

Access tier:

Hot(if data is used frequently helps to access data quickly), cool(if file is not used frequently),archive(you keep the data just for compliance and very very rarely used)

Types:

Blob storage/Container:

Stores unstructured data like text, images, videos which are considered as binary large object. It's often used for serving documents, media files, or backups,,
landing folder/Raw(where data lands from different sources—starting point)

Access level: public(only data should be allowed to public If it is used for any 3rd party) else mostly it is in private(closed access)

ADLS: High throughput and parallel processing can be done, Hierarchical namespace enabled, allows folders to create and good for big data analytics.

File share:

Can be used as common file sharing in local computer just map network drive with file share url and use credentials as storage account user name and password as the keys, can also used by 2 applications common sharing data(not our work)

Queue services:

Stores large numbers of messages that can be accessed from anywhere via authenticated HTTP or HTTPS calls. It's designed for reliable messaging between different systems.
ex: order processing system(once ordered msg is created and vender check and validate it and then process and after completing deletes it from queue).,,,,,,,

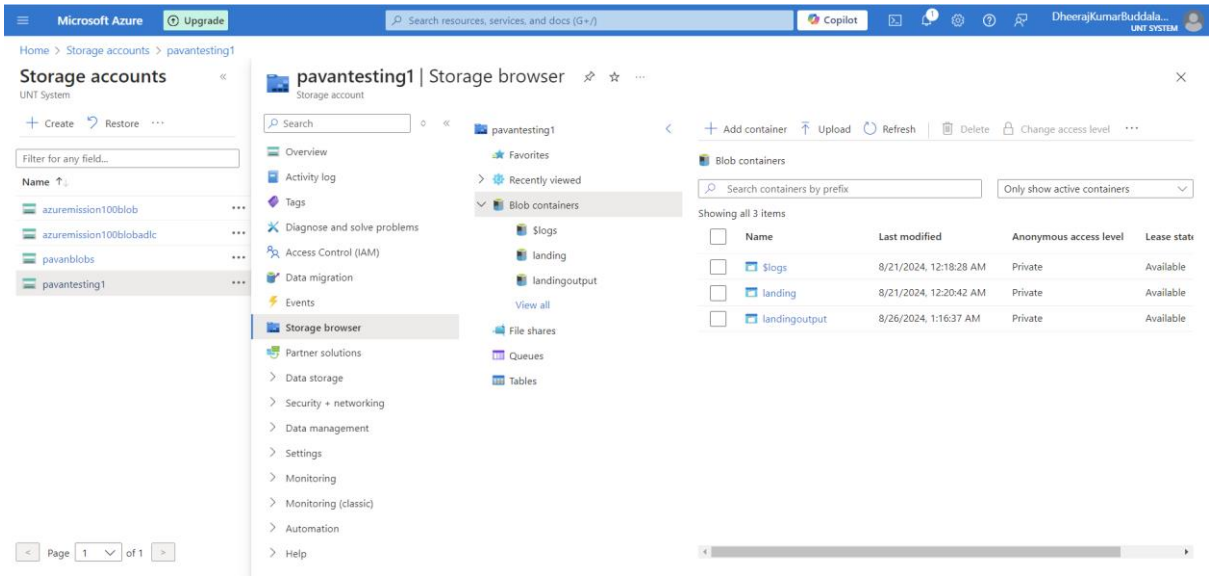
application generates a message which is intended to be used by the other application. However both of these applications are using different programming languages

Tables:

Azure Storage Explorer is a easy way to access the storage account, give good intereaction to work with storage accounts.

can be connected via subscription or storage account,

Azure table is not a database, it follows NoSql (columnar format every new data inserted can be in kind of different format) generally used to store any data that might be essential in future use.



Note:

Soft delete helps to restore the data upto 7 days even if deleted

Authentication: for checking right person or not

Authorization: for checking if this right person has access to use this service

Authentication can be done using access keys and SAS tokens but access keys give entire authorization to user for that storage account(exception case if storage account doesn't need access to keys while creation)

SAS(shared access signature):

helps us to give access to particular storage account with particular authorizations for the authenticated user

2 keys:

we have 2 keys since if we want to change the existing key we can ask everyone to change to key2 and then gradually change our existing 1st key

Data encryption:

whenever data is uploaded in azure or transferred to azure, data is encrypted if anybody accessed data they cant read it because of encryption,,,

2types encryption keys to read the data mmk(Microsoft managed keys), cmk(customer managed keys)

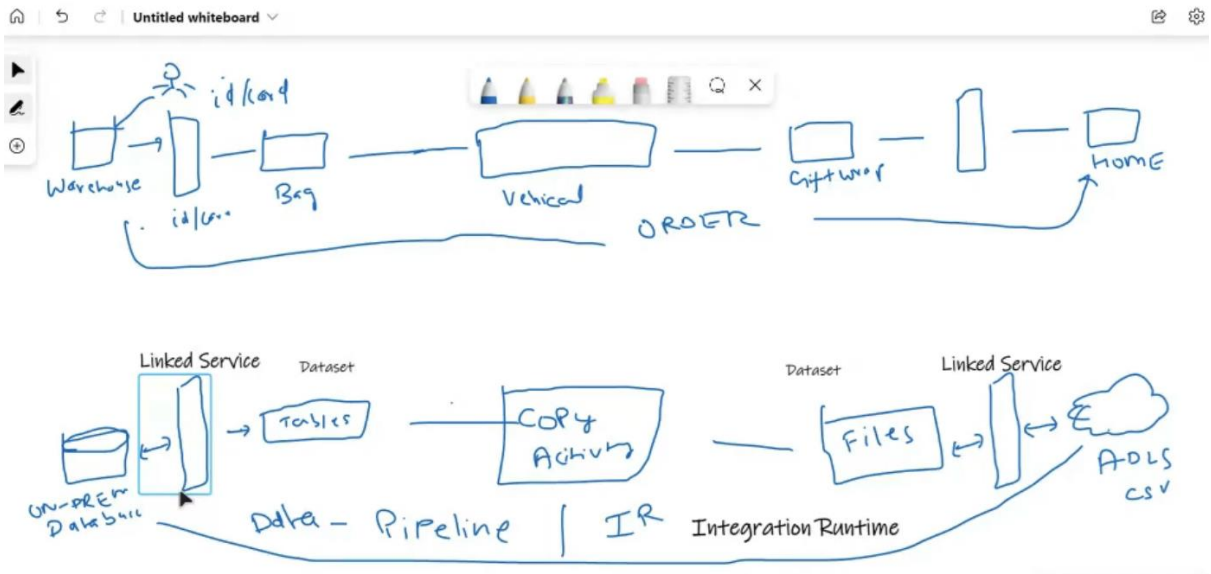
Azure SQL Server:
SQL server into the cloud,, we can create multiple databases inside the SQL server
while creating the database u can give authentication mechanism required for the database

AZURE DATA FACTORY

ADF is used to create, schedule and manage data pipelines or orchestrate(code free) data pipelines, it is a ETL(Extraction of data, Transforming and loading the data at destination) tool or can also be considered more powerful than ETL.
data transfer can be one time lift and shift or Incremental
ex: migrating data from on-prem to cloud
Data lake: place where all the organization data is stored at one point, created mostly for analytical purpose of data.(data engineer are one creating data lake)

Creation process same as the azure blob storage

Data Pipeline attributes:
we need source(to extract data) and sink(to load data), LS(Link service) to authenticate both source and sink and dataset to identify type of dataset are source and sink and finally the activities help to transform the data as we required.
"Publish it to save your changes"
Cost is based on how much time we running the pipeline



Integration runtime: it is the power house that gives run time power to the system
one Linked service can have multiple datasets related to it,, since it pretty common link service is only used to establish connection but dataset is type of data we need to extract or load using that link service

Pipeline debugging: after executing pipeline using debug option in bottom you will get options of

input and output data of the current pipeline executed which can be cross verified

copy behaviour:
Preserve: it presevers the same heirarcy of data it has, like same folder heirarcy, don't forgot to mark wild card path as *

parameterization of pipeline:
These are the common parameters than can be accessed inside the entire pipeline activities, it asks in every run time to give parameter value

Variables:(pipeline)
Values(in string format) are stored in variable name for temporary purpose or something we can use multiple times, unlike pipeline parameters variable doesn't ask parameter value every time pipeline runs.

the variable value can be changed dynamically by using a set variable activity while pipeline is running, whereas pipeline parameters can't be changed dynamically while pipeline running once their values are given.

Global Parameters:
These parameters can be used across any pipeline can be created under the manage tab
use case: something we feel can pass across all piplines or across the ADF account then we can use this
parametarization can happen in 4 ways → global>pipeline>Link service or dataset (whole concept is to make more dynamic without hardcoding)

Link Service:
used to create link for source and sink, basically we try to connect the source or sink by authenticating them with keys or key vault or user id and password whatever it works

Parameterization of link service:
in link service there are parameters section where you can add parameters and then can use them dynamically.
use case: creating dynamic link service with parameter as user name, now this can be usefull for many people who want to connect via different user names

Dataset:
used to recognize type of data we are going to use, has an option to mark first line as a header in dataset, helpful for csv files
for table selection from database it helps to select the table we wanted from database.

Parameterization of dataset:
every database has 100's of tables it will become very difficult to create many dataset for many tables, so parametarization is solution while creating dataset select none for table selection.
After creating dataset add parameters inside the dataset and use them as dynamic in connection to the dataset, these parameters are asked everytime while we running the pipelines

ex:

helps us to connect to directory dynamically, use case if we want to create folder on sink side with foldername/year/m/dd/time/file_name dynamically which changes every time pipeline runs.

The idea is to no to overwrite existing file and making sure the exact file from date

Activities:

Copy Activity:
copies the data from source to destination, the activity properties of source and sink changes based on type of dataset we are using at source and sink.
Mapping helpful to map the schema between source and destination side example mapping between 2 different data bases and matching the schema
write behaviour:
insert(just inserting), upsert(update and insert) need to give the key column helps to identify the unique id
Stored Procedure: This is useful for more advanced scenarios where you need custom logic to process or manipulate the data

Lookup:
helps to read the data(looking upon the data), This activity is particularly useful when you need to extract some information and use them in subsequent activities within the pipeline.
ex: select count(*) as record_count from table_name
If-else:
It functions similarly to an if-else statement in programming, making it a powerful tool for creating dynamic and conditional workflows within your data pipelines.
Nested if-else is not allowed at this point of time, so if you want to use nested if use the new pipeline and create if-else there and use execute pipeline to connect to it

Execute pipeline:
Helps to call another pipeline inside one pipeline, use case is breaking problem into smaller solutions and then chaining them.
to check the activity status of this pipeline follow the link obtained in the output section of this executed pipeline
For-each Activity:
it performs similar to the for loop in python programming, where here it takes array/list as an input and iterate one by one until list is done
the for each is going to run on count of the list count
The for-each current item can be accessible dynamically inside for each activity.
it can perform parllel and sequential for each operations.
Batch count helps no of tables(list items) need to copied at a time(control parllel operation items), helps the database to not to get slow down quickly
Default: sequential is off, batch count is 5, and max batch count is 50
use case: copying 3 tables using one copy activity using for each,
Nested for each is not available

Get-meta data activity:
it doesn't perform anything on its own, but helps to get metadata information about various types of files, folders, and databases. This activity is especially useful when you need to make decisions based on the properties of your data before performing further operations in your data pipeline.

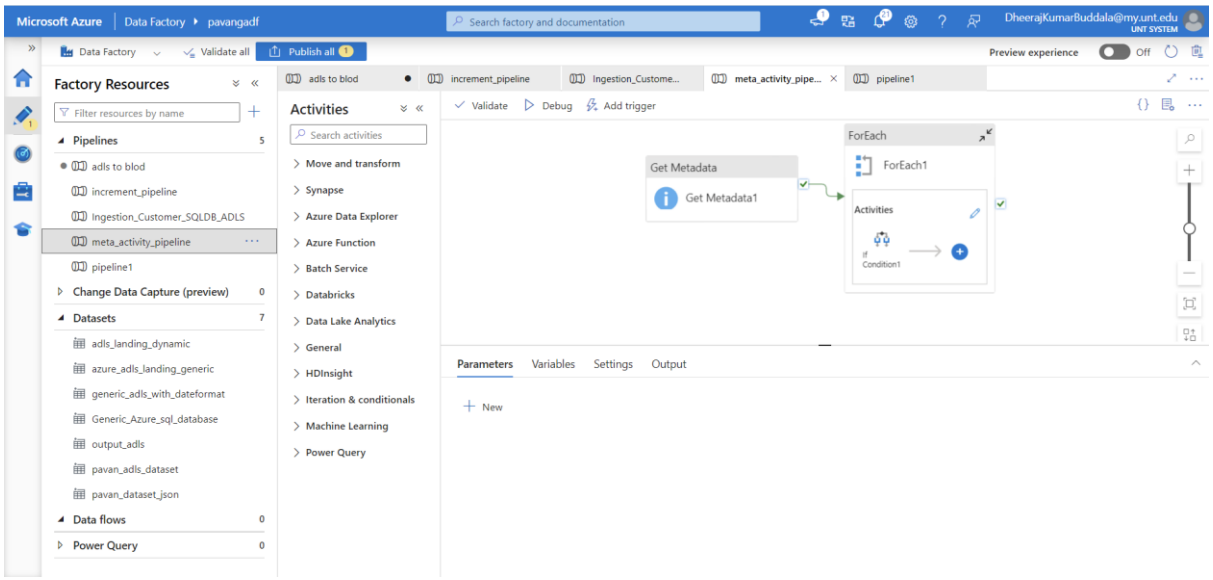
Field List:

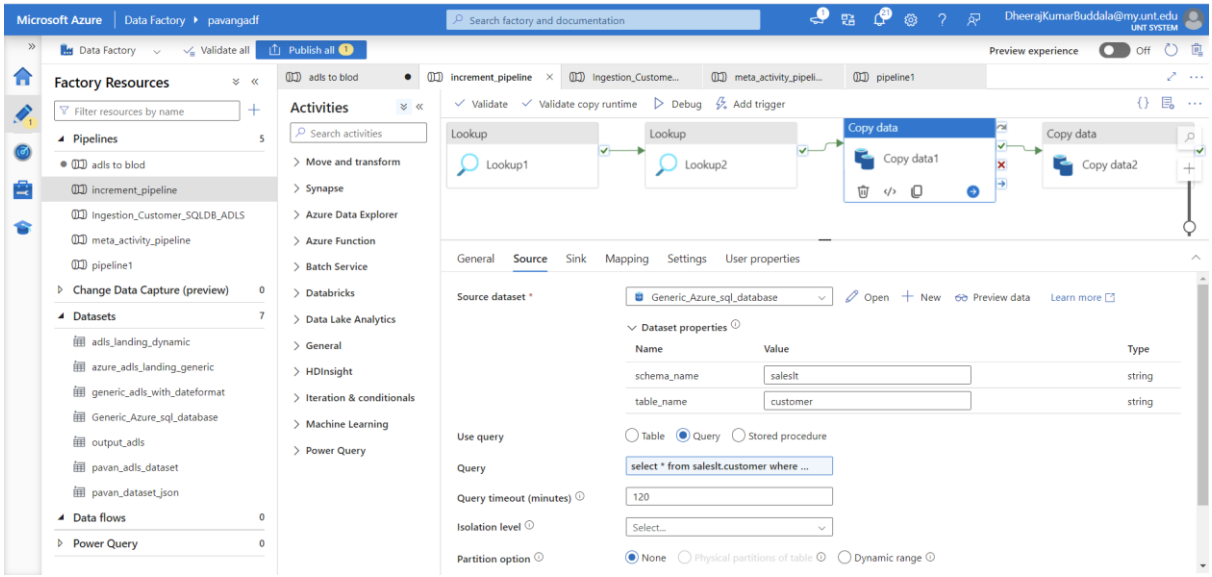
The options will get changed based on type of selection of input source
Exists: returns True if path exists else false
Childitems: Give you the child information in form of array like type of file and name of file.
Itemname,Itemtype,Lastmodified: helps to get the current item name and item type and last modified time or date
Columncount: only works for csv
ContentMD5: some string used for encryption not data engineer role
Structure: can get structure of the file or table(schema)

Usecase:
if we want to copy only type files to another container, or check if file exists and copy that file
if we want to find type of file use split activity to split the name of file

<u>Set Variable activity:</u>
Used to change the value of variable dynamically, use case can be used as to count the no of tables we done for each like that and then use value of variable later like that
self referencing cannot be done like x = x + 1

<u>dataflow activity:</u>
this activity helps us to select the dataflow which we want to run in our current pipeline





Pipeline Expression builder(available whenever we want to add dynamic expressions):

This helps us to connect activities in the pipelines, allows us to use functions, activity outputs, parameters, etc.,

to give the string dynamically '@{}' (use case if you want to pass any value in sql statement ex:Incremental pipeline)

Functions:

Logical functions: greater,smaller,equals and etc., helps to do logical operations, check function parameters what it is expecting, some cases lookup give output an unwanted string but greater is expected an integer

Activity:

it has Activity output, Pipeline output

Activity output returns the output of the activity

Parameters:

Show's the available parameters that can be used for that activity

Collection Functions:

CreateArray: helps to create the array with values of what we need

Date collections:

utcnow(): helps to get date,time,year and will get extra information also

formatDateTime: helps to extract the information we want

String Functions:

Concat: used to concat two strings and form complete string

@concat(item(),'/',formatDateTime(utcnow(), 'yyyy')),'MM', 'dd' for date and month and 'hh','mm' hour and minute

int(): converts to int

split: used to split based on the separator

add(): helps to do app operation

Azure Key Vault:

it contains 3 things keys,secret,certificate(ex:http certificate in browser)→ key vault is the place where we store secret information, and secret information can be keys, secret passwords or any certificates

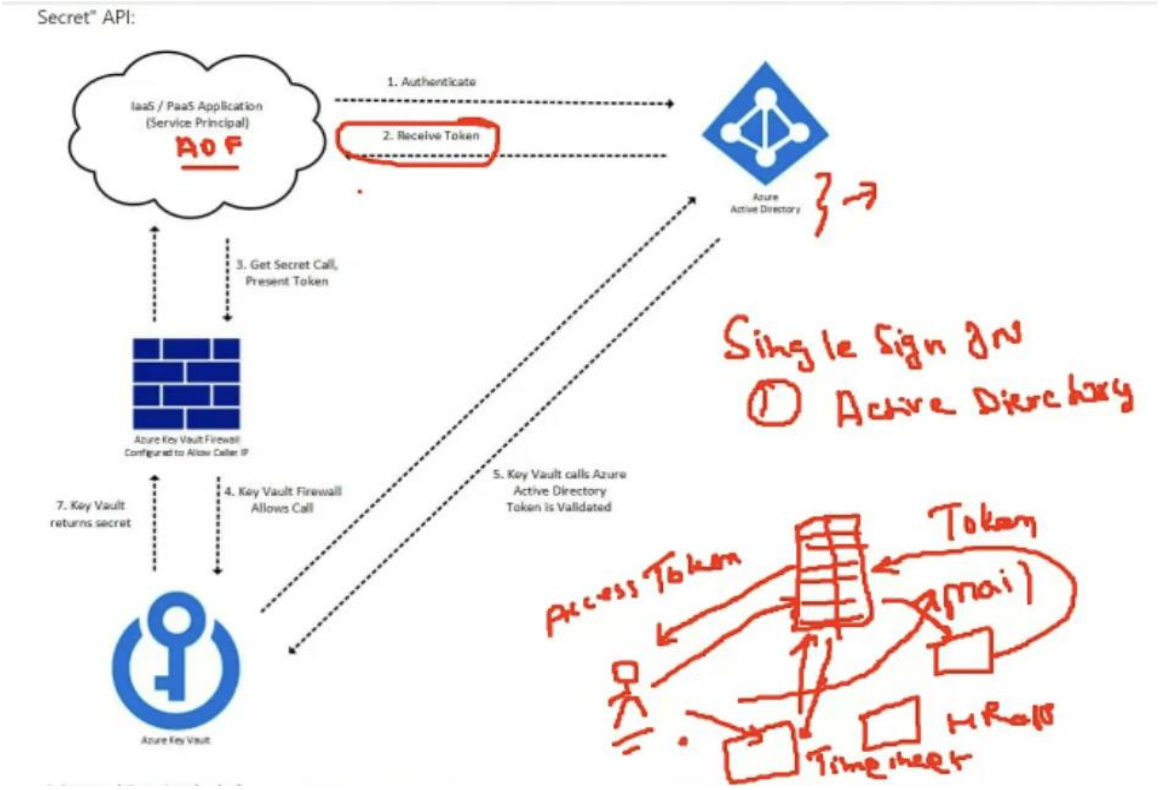
can be created same as storage account and ADF, we can give expire date, enable disable options as well

after creating we actually need to give permission for adf account to access this azure key vault this can be done in networking section, after authorization is done link service is needed to connect to this azure key vault and then we can connect the passwords in the key vault using secret name.

version helps us to identify which password or secret is latest

Active Directory:

It is of 2 types Active Directory(ON-premise) and Azure active directory,,,, these are used for single sign on operations



working mechanism:

azure active directory account is created when we create an azure account and token will be generated under the hood when we logged In and then if we want to sign on to adf, storage accounts, key vault ,,, under the hood it tries to validate or authenticate the token with the azure active directory if token is correct we are being auto signed in to the ADF remaining azure services

Incremental Pipeline:

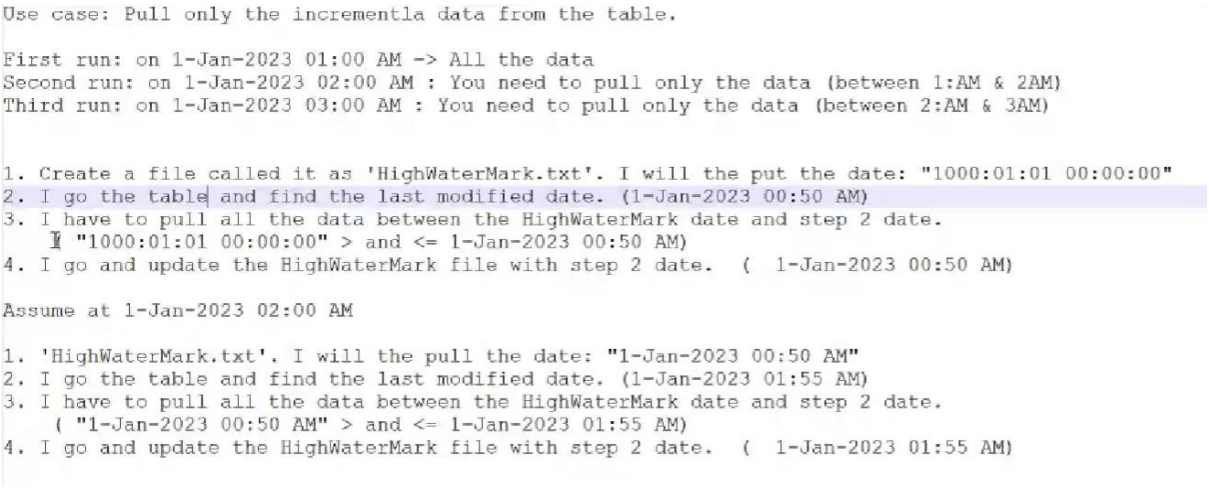
in general there are 2 types of piplines

Full load: creating pipeline we are doing kind of full load where taking all data from source and loading it in sink

Incremental: instead of taking all data from source pipeline will take the changed data from source

and load it into the sink(which make sense instead of pulling millions of data you just pull the changed data)
well the sinking of changing data depends on the business req whether it is daily,hourly or weekly

Working:



string interpolation: if we want to give string dynamically in the run time we can give it using '@{}'

we can doing using db table also like create high water mark table In database and its columns are table name and lastmodifieddate, but problem is sometime we don't have write access to the database so using high water mark file is the preferred way

Trigger: helps to schedule the pipeline (auto scheduling to run)

DATA FLOW:

Data Flow is a powerful feature that enables you to build visually designed data transformation logics without writing any code ,, the drag and drop flows under the hood it convert dataflow into spark code and run the code in the spark cluster

is it a spark replacement:

Not entirerly though you can do most of the basic or moderate work replacing spark but for complex queries writing code in spark is much more convenient than using Data Flow

Data flow debug: tries to allocate the spark cluster by default it is 4 machines

source:

helps to select the source file can be anything just like we selected source while creating data pipelines

source settings and source options:

name of the source(stream name), type of dataset(inline(this dataset created here will be taken as temporary and will not be reflected at dataset names just like originals) or existing)
input: table, query or stored procedure
most of them as same as source settings while creating data pipeline
Projection: this will help us to project existing schema and allows to rewrite it

inspect: helps to inspect the schema used in final data preview(updated schema)
data preview: helps to see the output sample data
projection and data preview kind of interlinked before importing schema try to run the data preview

Sink:

same as datapipeline it also used here to store data at last but here more options are allowed like mapping,data preview,inspect etc.,
here in sink by default we may got 6 files instead of 1 file, because spark runs in parallel mode making your file into pieces and then processing it
to control this select single partition as option in sink

if there are more than 1 sink in settings we can assign ordering which sink will run first

Note:
Dataflow isn't a activity we need a data pipeline to start it(in datapipline select dataflow activity and select our data flow and debug to start)
we can check in the inspect whether everything in dataflow executed correctly once pipeline is executed

options for data Transformation:

Schema modifiers:

select:
works same as in the SQL used to select the columns and o allows us to rename the column name, also have some more features like skip duplicate input or output columns

Derived column:
Helps to insert the new column or change the existing column and allows us to use dynamic expression builder for expression of new column(ex: Flagging Any country giving output as true if country is present)

Aggregate:
can be used as Groupby in sql so here in output you get output columns based on group by(column_name) and output(dynamic expression used to group the column) (ex:grouping by country and expression is summing the sales)
ex: aggregate functions sum,min,max,count()

Row modifiers:

Filter:
works as the where in the SQL used to filter the row data, here it allows us to use dynamic expression builder , same kind of functionality as pipeline expression builder

Sort by:
helps to sort in ascending or decreasing any particular column (with nulls first or last) it is basically a order by functionality in SQL

Alter row:
used to set the policies like which rows are eligible to update,upsert,delete etc.,

Multiple inputs:

Join:
similar to SQL join here we have options to join 2 streams(input) you can do inner,outer, right,left

whatever you want, but need to specify on what basis you are doing join
for example: inner join ,, condition: if both customer id are equal

Union:

combining 2 tables or any 2 data sources.

exists:

if we want to check if particular data exists in 2 tables or 2 data sources

conditional split:

splits the data into 2or more parts based on particular condition ,, its like if it qualifies the condition one data output is created and not another data output is created from the existing same data

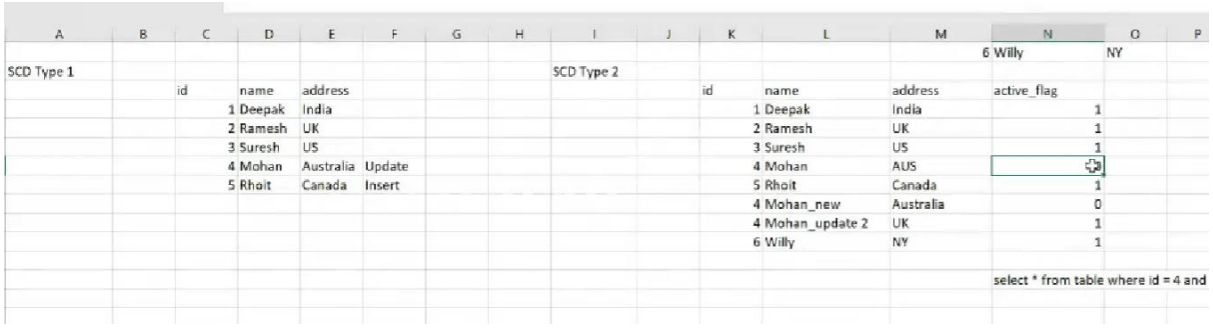
Lookup:

It works as a left outer join, on LHS will get all values, RHS if row exist we will get value else null

dynamic expression builder(data flow):

sum(), ==(all operators direct use), isnull()

SCD(Slowly changing Dimension - database concept):



													6 Willy	NY	
SCD Type 1								SCD Type 2							
		id	name	address					id	name	address	active_flag			
			1 Deepak	India						1 Deepak	India	1			
			2 Ramesh	UK						2 Ramesh	UK	1			
			3 Suresh	US						3 Suresh	US	1			
			4 Mohan	Australia	Update					4 Mohan	AUS				
			5 Rhoit	Canada	Insert					5 Rhoit	Canada	1			
										4 Mohan_new	Australia	0			
										4 Mohan_update 2	UK	1			
										6 Willy	NY	1			
													select * from table where id = 4 and		

it has 2 main types:

Type 1:

SCD (Slowly Changing Dimension) Type 1 is a data warehousing concept used to manage changes in dimensional data.

1.Overwrite the old data: When an update occurs, the existing record is modified directly. If record is new and not present in existing data table, then new record is inserted

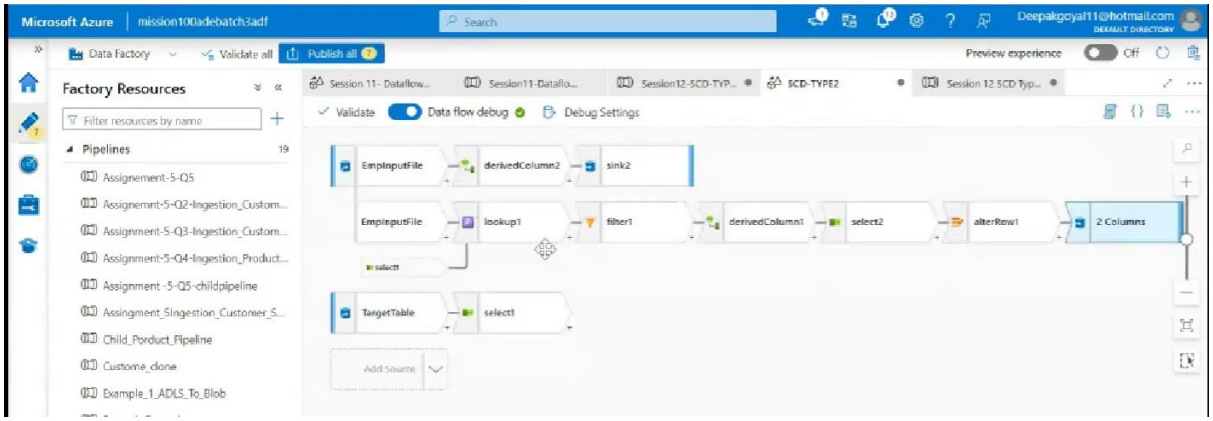
2.No history: Previous values are lost, as the update does not preserve the old state

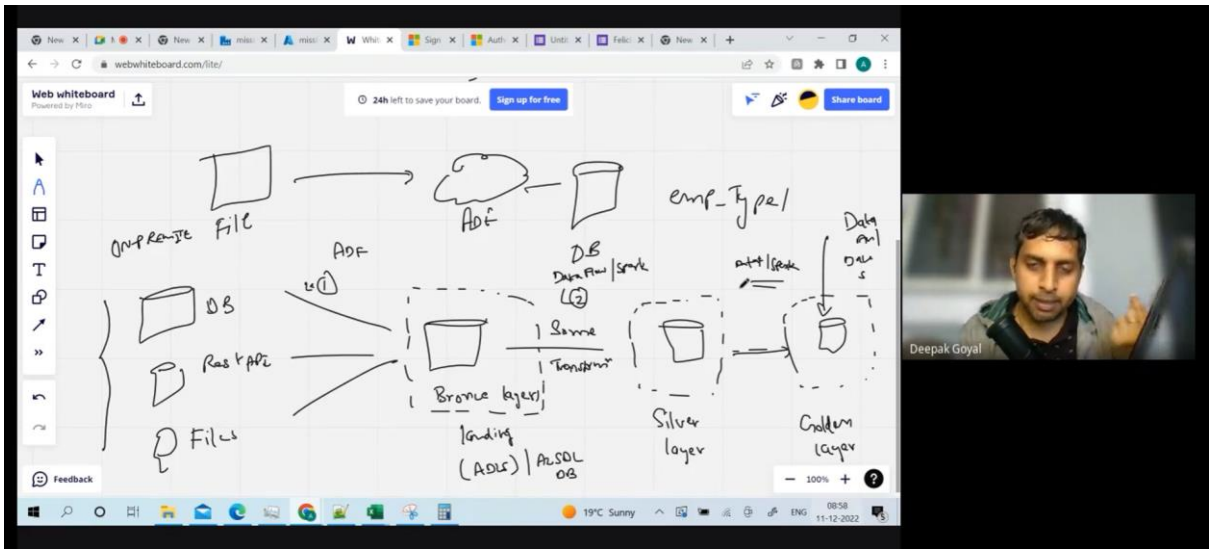
Implementation: copy activity with upsert at sink side is enough for SCD type 1

Type 2:

This will helps us to track the changed data(old record is also tracked along with the updated record)
To track the changes create new column like active flag,,, and it will be updated to 1 for new record and 0 for old record
ex: tracking of old passwords

implementation:
we need 2 sinks one to update active_flag in database 0 if record is already there and another to insert the new records to the database





(BRONZE → SILVER → GOLD)

Bronze Layer:

Dumping data from different sources to one area without changing any data(here we basically have raw data)

Silver Layer:

here we are going to do transformations from bronze layer data(removing unwanted data) ex: removing inconsistent columns, making null

Gold layer:

this data is very accurate it was transformed from silver layer(may be here we will do joins) and is transformed to be perfect data → data science or data analyst can use this data for their analysis and algorithm running