1 download and save and keep your one access key id and secret access key password 2 create a role for your redshift to access s3 . (RedhshiftS3Role) 3 next is have your cluster.config file ready , in this case it is at open main directory itself , create it in your local and then upload here

In [ ]:

```
```

In [1]:

```
import boto3import pandas as pd
import psycopg2
```

In [2]:

```
import pandas as pd
import psycopg2
```

In [3]:

```
import json
```

In [4]:

```
import configparser
config = configparser.ConfigParser()
config.read_file(open('cluster.config'))
```

In [5]:

```
config
```

Out[5]:

```
<configparser.ConfigParser at 0x7fa268a829e0>
```

In [6]:

```
config.get("AWS","KEY")
```

Out[6]:

```
'AKIA32HRISHKK5HBRU5D'
```

In [7]:

```
config.get("DWH","DWH_DB_PASSWORD")
```

Out[7]:

```
'Password1!'
```

In [10]:

```
config.get("DWH","DWH_CLUSTER_TYPE")
```

Out[10]:

```
'single-node'
```

In [ ]:

```
```

In [ ]:

```
```

In [18]:
```
KEY= config.get('AWS','KEY')
SECRET=config.get('AWS','SECRET')


DWH_CLUSTER_TYPE=config.get('DWH','DWH_CLUSTER_TYPE')
DWH_NUM_NODES=config.get('DWH','DWH_NUM_NODES')
DWH_NODE_TYPE=config.get('DWH','DWH_NODE_TYPE')
DWH_CLUSTER_IDENTIFIER=config.get('DWH','DWH_CLUSTER_IDENTIFIER')
DWH_DB=config.get('DWH','DWH_DB')
DWH_DB_USER=config.get('DWH','DWH_DB_USER')
DWH_DB_PASSWORD=config.get('DWH','DWH_DB_PASSWORD')
DWH_PORT=config.get('DWH','DWH_PORT')
DWH_IAM_ROLE_NAME=config.get('DWH','DWH_IAM_ROLE_NAME')


(DWH_IAM_ROLE_NAME)
```

Out[18]:
```
'sagemakerS3RdsRedshiftMainRole'
```

In [19]:
```
DWH_CLUSTER_TYPE
```

Out[19]:
```
'single-node'
```

In [20]:
```
pd.DataFrame({"Param":["DWH_CLUSTER_TYPE","DWH_NUM_NODES","DWH_NODE_TYPE","DWH_
              "Value":[DWH_CLUSTER_TYPE,DWH_NUM_NODES,DWH_NODE_TYPE,DWH_CLUSTE

})
```

Out[20]:

	Param	Value
0	DWH_CLUSTER_TYPE	single-node
1	DWH_NUM_NODES	1
2	DWH_NODE_TYPE	dc2.large
3	DWH_CLUSTER_IDENTIFIER	redshift-google201-vishtej-cluster-1
4	DWH_DB	dev
5	DWH_DB_USER	admin
6	DWH_DB_PASSWORD	Password1!
7	DWH_PORT	5439
8	DWH_IAM_ROLE_NAME	sagemakerS3RdsRedshiftMainRole

In [21]:
```
s3 = boto3.resource('s3',
                    region_name='ap-south-1',
                    aws_access_key_id=KEY,
                    aws_secret_access_key=SECRET
                   )
```

In [22]:
```
bucket = s3.Bucket("stratacent-vish-tej-main")
```

In [23]:
```
log_data_files = [filename.key for filename in bucket.objects.filter(Prefix='/
#gives the last alphabatical file
```

In [24]:
```
log_data_files
```

Out[24]:
```
['/temp/sample.csv']
```

In [25]:
```
log_data_files = [filename.key for filename in bucket.objects.filter(Prefix=''
#all the files in bucket
```

In [26]:
```
log_data_files
```

Out[26]:
```
['/temp/sample.csv',
 'datasets/',
 'datasets/project3/',
 'datasets/project3/passengers.csv',
 'datasets/project3/survival.csv',
 'datasets/project3/titanic.csv',
 'datasets/project3/trip_info.csv',
 'datasets/project4/',
 'datasets/project4/2018_Financial_Data.csv',
 'datasets/project4/electronicsData1.xlsx',
 'datasets/project4/excel/',
 'datasets/project4/excel/Financial_Data2018a.xlsx',
 'datasets/project4/temp1/',
 'datasets/project4/temp1/Financial_Data2018a.csv',
 'datasets/project4/temp1/Financial_Data2018adf11.csv',
 'ec2.txt',
 'ec2_1.txt',
 'ec2_2.txt',
 'imp_scripts/',
 'imp_scripts/script13.py',
 'temp/',
 'temp/sample.csv']
```

In [27]:
```
# if i just want all the files inside a dataset

log_data_files = [filename.key for filename in bucket.objects.filter(Prefix='d
```

In [28]:
```
log_data_files
```

Out[28]:
```
['datasets/',
 'datasets/project3/',
 'datasets/project3/passengers.csv',
 'datasets/project3/survival.csv',
 'datasets/project3/titanic.csv',
 'datasets/project3/trip_info.csv',
 'datasets/project4/',
 'datasets/project4/2018_Financial_Data.csv',
 'datasets/project4/electronicsData1.xlsx',
 'datasets/project4/excel/',
 'datasets/project4/excel/Financial_Data2018a.xlsx',
 'datasets/project4/temp1/',
 'datasets/project4/temp1/Financial_Data2018a.csv',
 'datasets/project4/temp1/Financial_Data2018adf11.csv']
```

In [ ]:
```
# getting iam role details
```

In [30]:
```
iam = boto3.client('iam',
                   region_name='ap-south-1',
                   aws_access_key_id=KEY,
                   aws_secret_access_key=SECRET
                  )
```

In [31]:
```
roleARN = iam.get_role(RoleName=DWH_IAM_ROLE_NAME)['Role']['Arn']
```

In [32]:
```
roleARN
```

Out[32]:
```
'arn:aws:iam::812254794196:role/sagemakerS3RdsRedshiftMainRole'
```

In [33]:
```
redshift = boto3.client('redshift',
                   region_name='ap-south-1',
                   aws_access_key_id=KEY,
                   aws_secret_access_key=SECRET
                  )
```

In [34]:

```
try:
    respond = redshift.create_cluster(
    ClusterType=DWH_CLUSTER_TYPE,
    NodeType=DWH_NODE_TYPE,

    #Identifiers & Credentials
    DBName=DWH_DB,
    ClusterIdentifier=DWH_CLUSTER_IDENTIFIER,
    MasterUsername=DWH_DB_USER,
    MasterUserPassword=DWH_DB_PASSWORD,

    #Roles (for s3 access)
    IamRoles=[roleARN]

    )
except Exception as e:
    print(e)
```

In [35]:
```
redshift.describe_clusters(ClusterIdentifier=DWH_CLUSTER_IDENTIFIER)
```

Out[35]:

```
{'Clusters': [{'ClusterIdentifier': 'redshift-google201-vishtej-cluster-1',
   'NodeType': 'dc2.large',
   'ClusterStatus': 'available',
   'ClusterAvailabilityStatus': 'Available',
   'MasterUsername': 'admin',
   'DBName': 'dev',
   'Endpoint': {'Address': 'redshift-google201-vishtej-cluster-1.c0uba31fdzr
s.ap-south-1.redshift.amazonaws.com',
    'Port': 5439},
   'ClusterCreateTime': datetime.datetime(2023, 2, 21, 14, 43, 55, 850000, tz
info=tzlocal()),
   'AutomatedSnapshotRetentionPeriod': 1,
   'ManualSnapshotRetentionPeriod': -1,
   'ClusterSecurityGroups': [],
   'VpcSecurityGroups': [{'VpcSecurityGroupId': 'sg-0a7797d2164306075',
     'Status': 'active'}],
   'ClusterParameterGroups': [{'ParameterGroupName': 'default.redshift-1.0',
     'ParameterApplyStatus': 'in-sync'}],
   'ClusterSubnetGroupName': 'default',
   'VpcId': 'vpc-00f3b19ae4198a911',
   'AvailabilityZone': 'ap-south-1b',
   'PreferredMaintenanceWindow': 'sat:07:30-sat:08:00',
   'PendingModifiedValues': {},
   'ClusterVersion': '1.0',
   'AllowVersionUpgrade': True,
   'NumberOfNodes': 1,
   'PubliclyAccessible': True,
   'Encrypted': False,
   'ClusterPublicKey': 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDRJKPHPOjQhZzNd
Sn9OHlaecQ4d3nip1It/tJEOk+vRznPuakdCFGqPlmqWj34VymIQlLr6444cPNw+lM8KUlWfvcJj9
bKxGxhtE1zwne2LqQMLw0BZ34pFqyjbv/j/r9NZ1PVbbnjE02CecgZ18V1OErYf/G2SVhaUuCLDSc
Hx4ajTIChYnE+pR1LjpsLkhjef56CYxZv6yjtkPU8FIILX4YUjRjmiF8qdtroceWjmKJKHSno77YO
gjawVLKYKncgTgZJ1CMNFnLzlysgbOLz748U2h5zvHOWwucMEQTNmRQu62Xy+H2YeIlBTOMP5tp2G
PBEUQHh1ZLTm4E6ozEr Amazon-Redshift\n',
   'ClusterNodes': [{'NodeRole': 'SHARED',
     'PrivateIPAddress': '172.31.7.244',
     'PublicIPAddress': '13.232.36.144'}],
   'ClusterRevisionNumber': '46806',
   'Tags': [],
   'EnhancedVpcRouting': False,
   'IamRoles': [{'IamRoleArn': 'arn:aws:iam::812254794196:role/sagemakerS3Rds
RedshiftMainRole',
     'ApplyStatus': 'in-sync'}],
   'MaintenanceTrackName': 'current',
   'DeferredMaintenanceWindows': [],
   'NextMaintenanceWindowStartTime': datetime.datetime(2023, 2, 25, 7, 30, tz
info=tzlocal()),
   'AvailabilityZoneRelocationStatus': 'disabled',
   'ClusterNamespaceArn': 'arn:aws:redshift:ap-south-1:812254794196:namespac
e:d61c3016-d658-47e2-8c8a-9cbeebebab2f',
   'TotalStorageCapacityInMegaBytes': 400000,
   'AquaConfiguration': {'AquaStatus': 'disabled',
    'AquaConfigurationStatus': 'auto'}}],
 'ResponseMetadata': {'RequestId': '1deddd4b-4115-40ec-a174-8a09ae007834',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '1deddd4b-4115-40ec-a174-8a09ae007834',
   'content-type': 'text/xml',
```

```
 'content-length': '4098',
 'date': 'Tue, 21 Feb 2023 14:49:10 GMT'},
'RetryAttempts': 0}}
```

In [36]:
```
['Cluster'][0]
```

Out[36]:
```
'Cluster'
```

In [37]:
```
redshift.describe_clusters(ClusterIdentifier=DWH_CLUSTER_IDENTIFIER)['Clusters
```

Out[37]:
```
{'ClusterIdentifier': 'redshift-google201-vishtej-cluster-1',
 'NodeType': 'dc2.large',
 'ClusterStatus': 'available',
 'ClusterAvailabilityStatus': 'Available',
 'MasterUsername': 'admin',
 'DBName': 'dev',
 'Endpoint': {'Address': 'redshift-google201-vishtej-cluster-1.c0uba31fdzrs.a
p-south-1.redshift.amazonaws.com',
  'Port': 5439},
 'ClusterCreateTime': datetime.datetime(2023, 2, 21, 14, 43, 55, 850000, tzin
fo=tzlocal()),
 'AutomatedSnapshotRetentionPeriod': 1,
 'ManualSnapshotRetentionPeriod': -1,
 'ClusterSecurityGroups': [],
 'VpcSecurityGroups': [{'VpcSecurityGroupId': 'sg-0a7797d2164306075',
   'Status': 'active'}],
 'ClusterParameterGroups': [{'ParameterGroupName': 'default.redshift-1.0',
   'ParameterApplyStatus': 'in-sync'}],
 'ClusterSubnetGroupName': 'default',
 'VpcId': 'vpc-00f3b19ae4198a911',
 'AvailabilityZone': 'ap-south-1b',
 'PreferredMaintenanceWindow': 'sat:07:30-sat:08:00',
 'PendingModifiedValues': {},
 'ClusterVersion': '1.0',
 'AllowVersionUpgrade': True,
 'NumberOfNodes': 1,
 'PubliclyAccessible': True,
 'Encrypted': False,
 'ClusterPublicKey': 'ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDRJKPHPOjQhZzNdSn
9OHlaecQ4d3nip1It/tJEOk+vRznPuakdCFGqPlmqWj34VymIQlLr6444cPNw+lM8KUlWfvcJj9bK
xGxhtE1zwne2LqQMLw0BZ34pFqyjbv/j/r9NZ1PVbbnjE02CecgZ18V1OErYf/G2SVhaUuCLDScHx
4ajTIChYnE+pR1LjpsLkhjef56CYxZv6yjtkPU8FIILX4YUjRjmiF8qdtroceWjmKJKHSno77YOgj
awVLKYKncgTgZJ1CMNFnLzlysgbOLz748U2h5zvHOWwucMEQTNmRQu62Xy+H2YeIlBTOMP5tp2GPB
EUQHh1ZLTm4E6ozEr Amazon-Redshift\n',
 'ClusterNodes': [{'NodeRole': 'SHARED',
   'PrivateIPAddress': '172.31.7.244',
   'PublicIPAddress': '13.232.36.144'}],
 'ClusterRevisionNumber': '46806',
 'Tags': [],
 'EnhancedVpcRouting': False,
 'IamRoles': [{'IamRoleArn': 'arn:aws:iam::812254794196:role/sagemakerS3RdsRe
dshiftMainRole',
   'ApplyStatus': 'in-sync'}],
 'MaintenanceTrackName': 'current',
 'DeferredMaintenanceWindows': [],
 'NextMaintenanceWindowStartTime': datetime.datetime(2023, 2, 25, 7, 30, tzin
fo=tzlocal()),
 'AvailabilityZoneRelocationStatus': 'disabled',
 'ClusterNamespaceArn': 'arn:aws:redshift:ap-south-1:812254794196:namespace:d
61c3016-d658-47e2-8c8a-9cbeebebab2f',
 'TotalStorageCapacityInMegaBytes': 400000,
 'AquaConfiguration': {'AquaStatus': 'disabled',
  'AquaConfigurationStatus': 'auto'}}
```

In [38]:
```
def prettyRedshiftProps(props):
    pd.set_option('display.max_colwidth',-1)
    keysToShow = ["ClusterIdentifier","NodeType","ClusterStatus","MasterUsernar
    x = [(k,v) for k,v in props.items() if k in keysToShow]
    return pd.DataFrame(data=x,columns=["Key","Value"])

myClusterProps = redshift.describe_clusters(ClusterIdentifier=DWH_CLUSTER_IDEN
prettyRedshiftProps(myClusterProps)
```

```
/tmp/ipykernel_10904/2941255731.py:2: FutureWarning: Passing a negative integ
er is deprecated in version 1.0 and will not be supported in future version.
Instead, use None to not limit the column width.
  pd.set_option('display.max_colwidth',-1)
```

Out[38]:

	Key	Value
0	ClusterIdentifier	redshift-google201-vishtej-cluster-1
1	NodeType	dc2.large
2	ClusterStatus	available
3	MasterUsername	admin
4	DBName	dev
5	Endpoint	{'Address': 'redshift-google201-vishtej-cluster-1.c0uba31fdzrs.ap-south-1.redshift.amazonaws.com', 'Port': 5439}
6	VpcId	vpc-00f3b19ae4198a911

In [39]:
```
DWH_ENDPOINT = myClusterProps['Endpoint']['Address']
DWH_ROLE_ARN = myClusterProps['IamRoles'][0]['IamRoleArn']
DB_NAME = myClusterProps['DBName']
DB_USER = myClusterProps['MasterUsername']
```

In [40]:
```
DB_NAME
```

Out[40]:
```
'dev'
```

In [41]:
```
DB_USER
```

Out[41]:
```
'admin'
```

In [42]:
```
DWH_ROLE_ARN
```

Out[42]:
```
'arn:aws:iam::812254794196:role/sagemakerS3RdsRedshiftMainRole'
```

In [43]:
```
DWH_ENDPOINT
```

Out[43]:
```
'redshift-google201-vishtej-cluster-1.c0uba31fdzrs.ap-south-1.redshift.amazon
aws.com'
```

In [ ]:

In [50]:
```
try:
   conn = psycopg2.connect(host=DWH_ENDPOINT,dbname=DB_NAME,user=DB_USER,passw
except psycopy2.Error as e:
    print(e)

conn.set_session(autocommit=True)
print("successfull connected to redshift")
```

```
successfull connected to redshift
```

In [51]:
```
try:
    cur = conn.cursor()
except psycopg2.Error as e :
    print(e)

print("cursor for redshift established successfully")
```

```
cursor for redshift established successfully
```

In [46]:
```
try:
   cur.execute("""
   create table employees(jobId varchar(50),companyId varchar(50),jobType varcl

   """)
except psycopg2.Error as e:
    print(e)
```

In [ ]:
```
query = copy employees from 's3://stratacent-vish-tej-main/datasets/project16_
   credentials 'aws_iam_role=arn:aws:iam::812254794196:role/sagemakerS3RdsRedsh
   delimiter ','
   region 'ap-south-1'
```

In [60]:
```
try:
   cur.execute("""
   copy employees from 's3://stratacent-vish-tej-main/datasets/project16_redsh
   credentials 'aws_iam_role=arn:aws:iam::812254794196:role/sagemakerS3RdsRedsh
   delimiter ','
   region 'ap-south-1'

   """)
except psycopg2.Error as e:
    print(e)
```

```
exception name : UnauthorizedException, error type : 135, message: Not author
ized to get credentials of role arn:aws:iam::812254794196:role/sagemakerS3Rds
RedshiftMainRole, should retry : 0
DETAIL:
  -----------------------------------------------
  error:  exception name : UnauthorizedException, error type : 135, message:
Not authorized to get credentials of role arn:aws:iam::812254794196:role/sage
makerS3RdsRedshiftMainRole, should retry : 0
  code:      30000
  context:
  query:     364
  location:  xen_aws_credentials_mgr.cpp:411
  process:   padbmaster [pid=32027]
  -----------------------------------------------
```

In [54]:
```
import pandas
import sqlalchemy
```

In [55]:
```
engine = sqlalchemy.create_engine('postgresql://postgsuperuser:Password1!@redsh
#postgres://postgsuperuser:password1!@ola201-vishtejpostgre01.cluster-caddif7r
```

In [58]:
```
data = pandas.read_csv('s3://stratacent-vish-tej-main/datasets/project16_redsh
```

In [59]:
```
data.to_sql('employees2',engine)
```

```
---------------------------------------------------------------------------
OperationalError                          Traceback (most recent call last)
~/anaconda3/envs/python3/lib/python3.10/site-packages/sqlalchemy/engine/base.py in _wrap_pool_connect(self, fn, connection)
   3360         try:
-> 3361             return fn()
   3362         except dialect.dbapi.Error as e:

~/anaconda3/envs/python3/lib/python3.10/site-packages/sqlalchemy/pool/base.py in connect(self)
    319         """
--> 320         return _ConnectionFairy._checkout(self)
    321

~/anaconda3/envs/python3/lib/python3.10/site-packages/sqlalchemy/pool/base.py in _checkout(cls, pool, threadconns, fairy)
    883         if not fairy:
--> 884             fairy = _ConnectionRecord.checkout(pool)
    885
```

In [ ]: