```
%run /Workspace/Users/vm2001kk@gmail.com/path

from pyspark.sql import SparkSession

#%run /Workspace/Users/vm2001kk@gmail.com


#### Mounting of containers

# dbutils.fs.mount(source='wasbs://raw@projectbsa.blob.core.windows.net',

#                  mount_point='/mnt/blobstore',

# extra_configs={'fs.azure.account.key.projectbsa.blob.core.windows.net':'z910CsRPXiS4d4TzsPzK2Khfus2uZaBNtibYt85mHrThoK5HEtvoopkLEqvMm2SLhyWFp4Ey2E0X+ASts1svMQ=='})

#### Raw Layer

%python

# Unmount the existing mount point

dbutils.fs.unmount('/mnt/blobstore')


# Mount the storage again

dbutils.fs.mount(

  source='wasbs://raw@projectbsa.blob.core.windows.net',

  mount_point='/mnt/blobstore',

  extra_configs={'fs.azure.account.key.projectbsa.blob.core.windows.net': 'z910CsRPXiS4d4TzsPzK2Khfus2uZaBNtibYt85mHrThoK5HEtvoopkLEqvMm2SLhyWFp4Ey2E0X+ASts1svMQ=='}

)

display(dbutils.fs.ls("/mnt/blobstore"))

dbutils.fs.unmount('/mnt/bronze')

## Bronze Layer

# Unmount the existing mount point



# Mount the storage again
```

```
dbutils.fs.mount(
   source='wasbs://bronze@projectdls.blob.core.windows.net',
   mount_point='/mnt/bronze',
   extra_configs={'fs.azure.account.key.projectdls.blob.core.windows.net':
'NsRyNKa/hXt+6YClzXJseOmlEQV72nFTzZUmg4Fj7HliUp0pjEpzK0wFb7CjxNwW25KL+hNub0yU+AStySga
5w==''}
)
# display(dbutils.fs.ls("/mnt/bronze"))
# dbutils.fs.mount(source='wasbs://silver@projectdls.blob.core.windows.net',
#               mount_point='/mnt/silver',
#
extra_configs={'fs.azure.account.key.projectdls.blob.core.windows.net':'nMMDc1P9OJjtr6dT+A0vEtN6nJ
dCHs5p/GGBtblBf27Bqz3MmnMXHBDEOvL+rgBNPiCJBgEjiiX9+AStfgqxtw==''})
display(dbutils.fs.ls("/mnt/silver"))
# dbutils.fs.mount(source='wasbs://gold@projectdls.blob.core.windows.net',
#               mount_point='/mnt/gold',
#
extra_configs={'fs.azure.account.key.projectdls.blob.core.windows.net':'nMMDc1P9OJjtr6dT+A0vEtN6nJ
dCHs5p/GGBtblBf27Bqz3MmnMXHBDEOvL+rgBNPiCJBgEjiiX9+AStfgqxtw==''})
#display(dbutils.fs.ls("/mnt/gold"))
df_d=spark.read.csv('dbfs:/mnt/blobstore/department_Data.csv',header=True,inferSchema=True)
df_d.show()
df_d.printSchema()
df_e=spark.read.csv("dbfs:/mnt/blobstore/employee_Data.csv",header=True,inferSchema=True)
df_e.show()
df_s=spark.read.csv("dbfs:/mnt/blobstore/salary_Data.csv",header=True,inferSchema=True)
df_s.show()
%python
df_s.write.parquet('/mnt/bronze/salary_info', mode='overwrite')
display(df_s)
```

```
%python
df_e.write.parquet('/mnt/bronze/employee_info', mode='overwrite')
df_d.write.parquet('/mnt/bronze/department_info',mode='overwrite')
salary= spark.read.parquet("/mnt/bronze/salary_info")

#salary = spark.read.parquet("/mnt/bronze/employee_info.parquet")
salary.printSchema()
%python
employee = spark.read.parquet("/mnt/bronze/employee_info")
employee.printSchema()
department= spark.read.parquet("/mnt/bronze/department_info")
department.printSchema()

from pyspark.sql.functions import current_date, current_timestamp

salary=salary.withColumn('ingestion_date',current_timestamp())
salary.show(n=4)
from pyspark.sql.functions import *
employee=employee.withColumn('ingestion_date',current_timestamp())
employee=employee.withColumn(' source_file',lit('employee.csv'))
employee.show(n=4)
department=department.withColumn('ingestion_date',current_timestamp())\
   .withColumn('source_file',lit("department.csv"))
department.show(n=4)
display(dbutils.fs.ls("/mnt/bronze"))
## Silver Work
df_sal=spark.read.parquet('dbfs:/mnt/bronze/salary_info/')
df_emp=spark.read.parquet('dbfs:/mnt/bronze/employee_info/')
```

```
df_dep=spark.read.parquet('dbfs:/mnt/bronze/department_info/')
### check Null values
for i in df_sal.columns:
   print(i,df_sal.filter(col(i).isNull()).count())
for i in df_emp.columns:
   print(i,df_emp.filter(col(i).isNull()).count())
for i in df_dep.columns:
   print(i,df_dep.filter(col(i).isNull()).count())
## Check for duplicates
df_dep.groupBy(df_dep.columns).count().filter(col('count')>1).display()
df_sal.groupBy(df_sal.columns).count().filter(col('count')>1).display()
df_emp.groupBy(df_emp.columns).count().filter(col('count')>1).display()
## drop Null values

df_sal=df_sal.dropna(subset=['department_id','salary_amount'])
df_sal.count()
df_emp=df_emp.dropna(subset=['job_title'])
#df_emp.count()
## Computing Additional Fields
## month wise bonus
## Minimum bonus
df_sal=df_sal.withColumn('min_sal',col('salary_amount') *(8.33 / 100))
df_sal=df_sal.withColumn('min_sal',round(df_sal['min_sal'],2))
df_sal.show(n=4)

## Maximum bonus
df_sal=df_sal.withColumn('max_sal',col('salary_amount') *(20 / 100))
df_sal=df_sal.withColumn('max_sal',round(df_sal['max_sal'],2))
df_sal.show(n=4)
```

```
df_sal=df_sal.withColumn(colName='total_salary_value',col=col('salary_amount').cast('float'))

df_sal.show()

df_sal=df_sal.withColumn('processed_date',current_date())

df_sal.show()

df_sal.printSchema()

## Joining Related Tables

df_join=df_sal.join(df_emp,on='employee_id',how='inner')\

   .join(df_dep,on='department_id',how='left')

df_join.display()

salary_par=df_join.select('salary_id','employee_name','job_title','department_name','salary_amount','t
otal_salary_value','salary_date' ,'processed_date')



employee_par=df_join.select('employee_name','job_title')

department_par=df_join.select('department_name')

salary_par.write.mode('overwrite').parquet('/mnt/silver/salary_parquet')

employee_par.write.mode('overwrite').parquet('/mnt/silver/employee_parquet')

department_par.write.mode('overwrite').parquet('/mnt/silver/department_parquet')

display(dbutils.fs.ls("/mnt/silver"))

%python

salary_par.write \

   .format("jdbc") \

   .option("url",
"jdbc:sqlserver://servernew011.database.windows.net:1433;databaseName=project_1") \

   .option("dbtable", "silver_db.salary_silver") \

   .option("user", connectionProperties["user"]) \

   .option("password", connectionProperties["password"]) \

   .mode("OVERWRITE") \

   .save()
```

```
# <\\df\\>.write \
#     .format("jdbc") \
#     .option("url", "jdbc:sqlserver://<\\servername\\>.database.windows.net:1433;databaseName=<\\dbname\\>") \
#     .option("dbtable", "silver_db.<\\tablename\\>") \
#     .options(**connectionProperties) \
#     .mode("OVERWRITE") \
#     .save()
connectionProperties = {
  "user": "sush_chaudhari",
  "password": "mita@3602",
  "driver": "com.microsoft.sqlserver.jdbc.SQLServerDriver"
}
salary_par.write \
    .format("jdbc") \
    .option("url", "jdbc:sqlserver://servernew011.database.windows.net:1433;databaseName=project_1") \
    .option("dbtable", "silver_db.salary_silver") \
    .options(**connectionProperties) \
    .mode("OVERWRITE") \
    .save()
employee_par.write \
    .format("jdbc") \
    .option("url", "jdbc:sqlserver://servernew011.database.windows.net:1433;databaseName=project_1") \
    .option("dbtable", "silver_db.employee_silver") \
    .options(**connectionProperties) \
    .mode("OVERWRITE") \
    .save()
department_par.write \
```

```
    .format("jdbc") \
    .option("url",
"jdbc:sqlserver://servernew011.database.windows.net:1433;databaseName=project_1") \
    .option("dbtable", "silver_db.department_silver") \
    .options(**connectionProperties) \
    .mode("OVERWRITE") \
    .save()
d=spark.read.parquet('/mnt/silver/employee_parquet')
d.show()
df_read = spark.read \
    .format("jdbc") \
    .option("url", "jdbc:sqlserver://surajsat01.database.windows.net:1433;databaseName=serversql") \
    .option("dbtable", "silver_db.department_silver") \
    .options(**connectionProperties) \
    .load()

df_read.display()
%python
connectionProperties = {
    "user": "sush_chaudhari",
    "password": "mita@3602",
    "driver": "com.microsoft.sqlserver.jdbc.SQLServerDriver"
}

df_read = spark.read \
    .format("jdbc") \
    .option("url",
"jdbc:sqlserver://servernew011.database.windows.net:1433;databaseName=project_1") \
    .option("dbtable", "silver_db.department_silver") \
```

```
    .options(**connectionProperties) \
    .load()

df_read.display()
```

SQL Scripts

Gold Layer

```
%sql
create schema gold_db;

create table salary_gold(
department_name varchar(50),
total_salary_value decimal,
avg_salary decimal,
report_date date
);

insert into salary_gold (department_name,total_salary_value,avg_salary,report_date)
select s.department_name,sum(s.salary_amount) as total_salary_value,avg(salary_amount) as average_salary,getDate() as report_date
from silver_db.salary_silver as s
join [silver_db].[department_silver] as d on s.department_name = d.department_name
group by s.department_name

select * from dbo.salary_gold
```