**SQL & Python Practice**

**Date:-10-12-2024 -Task**

**1.** **SQL Queries:-**

**🡪 New Column Creation:-**

**🡪 Syntax:-**

-- Alter table name

--Add column name

**🡪 Value Updates:-**

**🡪 Syntax:-**

--update table name

--set column name (new column name)

--where column name (it’s already column name)

**2.Python Syntax:-**

**🡪 New Column Creation:-**

**🡪 Syntax:-**

Df=df.copy()

Df[‘New\_Column’]=’Value’

Print(Df)

**🡪 Value Updates:-**

**🡪 Syntax:-**

Df[‘New\_Column’]=[‘Value1’, ‘Value2’, ‘Value3’, ‘Value4’, ‘Value5’]

**Date:-11-12-2024 \_Task**

**1.** **SQL Queries:-**

**🡪 Case When queries:- (single value update)**

**🡪 Syntax:-**

SELECT column\_name1, column\_name2, (it’s already column names in table)

CASE  
     WHEN *condition1* THEN *result1*  
    WHEN *condition2* THEN *result2*  
     ELSE default\_result

END AS Column Nema (New Creation Column Name)    
 FROM Table Name

**2.Python Syntax:-**

**🡪 Np Where condition:- (single value update)**

**🡪 Syntax:-**

Df[‘New\_Column’]=np.where(df[‘condition’], value\_if\_true, value\_if\_false)

**Date:-12-12-2024 \_Task**

**1.** **SQL Queries:-**

**🡪 Case When queries:- (multiple values updates)**

**🡪 Syntax:-**

SELECT column\_name1, column\_name2, column\_name3, (it’s already column names in table)

CASE

WHEN condition1 THEN result1

WHEN condition2 THEN result2

WHEN condition2 THEN result2

ELSE default\_result

END AS Column Name (New Creation Column Name)

FROM Table Name

**2.Python Syntax:-**

**🡪Np Where condition:- (multiple value updates)**

**🡪 Syntax:-**

Df[‘New\_Column’] = np.where (df[‘Score’] >= 50, ‘Pass’, ‘Fail’)

**Date:-13-12-2024 \_Task**

**1.SQL Join Queries:-**

**🡪 Inner Joins:-**

**🡪 Syntax:-**

Select\*

From table name (Base)

Inner join table name (Child)

On table name (Base). Column name (Matched column)=table name (Child). Column name (Matched column)

**🡪 Left Joins:-**

**🡪 Syntax:-**

Select\*

From table name (Base)

Left join table name (Child)

On table name (Base). Column name (Matched column)=table name (Child). Column name (Matched column)

**2.Python Syntax:-**

**🡪 Inner Joins:-**

**🡪 Syntax:-**

Df\_table name (Base).Merge(df\_ table name (Child),how=’inner’,on=’key’)

**🡪 Left Joins:-**

**🡪 Syntax:-**

Df\_table name (Base).Merge(df\_table name (Child),how=’left’,on=’key’)

**🡪 Right Joins:-**

**🡪 Syntax:-**

Df\_table name (Base).Merge(df\_table name (Child),how=’right’,on=’key’)

**🡪 Outer Joins:-**

**🡪 Syntax:-**

Df\_table name (Base).Merge(df\_table name (Child),how=’outer’,on=’key’)

**Date:-16-12-2024 -Task**

**1.** **SQL Queries:-**

**🡪 Filters:- (And, Or & Multiple Conditions)**

**🡪 1.And Condition**

**🡪 Syntax:-**

Select\*

From Table Name

Where Column Name=’Value’(Value in Column)

**🡪2. Syntax:-**

Select\*

From Table Name

Where Column Name=’Value’(Value in Column) and Column Name=’Value’(Value in Column)

\*\*\*\*\*\*\*\*\*\*And Condition Column & Row are Matched.\*\*\*\*\*\*\*\*\*\*

**🡪3. OR Condition**

**🡪 Syntax:-**

Select\*

From Table Name

Where (Column Name=’Value’(Value in Column) or Column Name=’Value’(Value in Column)

\*\*\*\*\*Or Condition Column & Row Two Conditions are Satisfy then Display Output.\*\*\*\*\*

🡪4. **Multiple Conditions:-**

**🡪 Syntax:-**

Select\*

From Table Name

Where (Column Name=’Value’(Value in Column) and Column Name=’Value’(Value in Column)) or Column Name =’Value’ (Value in Column)

\*\*\*\*\*Or Condition Column & Row Two Conditions are Satisfy then Display Output.\*\*\*\*\*

**2.Python Syntax:-**

**🡪 Filters:-**

**1.Equal Condition**

**2.Not Equal Condition**

**3.And Condition**

**4. OR Condition**

**5.Isin Condition**

**6. Not Isin Condition**

**1.Equal Condition:-**

**🡪 Syntax:-**

df[df['Column Name']=='Value'(Value in Column Name )]

**2.Not Equal Condition:-**

**🡪 Syntax:-**

df[df[' Column Name’]!= 'Value'(Value in Column Name )]

**3.And Condition:-**

**🡪 Syntax:-**

df[(df[' Column Name’]==''Value'(Value in Column Name )]) & (df[‘Column Name’]== 'Value'(Value in Column Name )])]

\*\*\*\*\*\*\*\*\*\*And Condition Column & Row are Matched.\*\*\*\*\*\*\*\*\*\*

**4. OR Condition:-**

**🡪 Syntax:-**

df[(df[' Column Name’]=='Value'(Value in Column Name )]) | (df[‘Column Name’]== 'Value'(Value in Column Name )])]

\*\*\*\*\*Or Condition Column & Row Two Conditions are Satisfy then Display Output.\*\*\*\*\*

**5.Isin Condition:-**

**🡪 Syntax:-**

lst=['Value1', 'Value2,'Value’ (Values in Column Name )]

df[df.Column Name.isin(lst)]

**6. Not Isin Condition:-**

**🡪 Syntax:-**

lst=['Value1', 'Value2,'Value’ (Values in Column Name )]

df[~df.Column Name.isin(lst)]

**Date:-17-12-2024 -Task**

**1.** **SQL Queries:-**

**🡪 In Condition:- (Particular Values)**

**🡪 Syntax:-**

Select\*

From Table Name

Where Column Name In (‘Value1’,’Value2’,’Value3’,’Value4’)

**🡪 Between Condition:- (Between Values)**

**🡪 Syntax:-**

Select\*

From Table Name

Where Column Name Between (‘Value1’,’Value2’,’Value3’,’Value4’)

**2.Python Syntax:-**

**🡪 In Condition:- (Particular Values)**

**🡪 Syntax:-**

df[df['Column\_Name'].isin([‘Value1’,’Value2’,’Value3’,’Value4’])]

**Date:-18-12-2024 -Task**

**1.SQL Queries:-**

**🡪 Limit Offset:-**

* **Syntax:-**

Select\*

From table name

Limit[offset] row\_count

**2.Python Syntax:-**

**🡪 1.def function:-**

* **Syntax:-**

Def Function\_Name(Parameter1, Parameter2, Parameter3,):

< Statements >

< Logic >

Return Value

Return (Value1, Value2, Value3,)

* **2. Concat or Appending:-**

🡪 **One By One Date Append:-**

🡪 **Syntax:-**

**df=pd.**concat([ df1,df2],axis=0)

🡪 **Said By Said Date Append:-**

🡪 **Syntax:-**

df=pd.concat([ df1,df2],axis=1)

**Status:-**

1.Understanding the project work process

2.what is dataware house in snowflake

3.how many dataware house types in snowflake

4.how to crate dataware house in snowflake