Python 101

Using Pandas

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Reading a CSV File

Loading data from a CSV file into a DataFrame.

```
df = pd.read_csv('data.csv')
```

The pd.read_csv() function reads a CSV file and loads it into a Pandas DataFrame. The string 'data.csv' is the file path.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

DataFrame Creation

Creating a DataFrame from a dictionary of lists.

data = {'Name': ['Alice', 'Bob'], 'Age': [25, 30]} \n
df = pd.DataFrame(data)

The pd.DataFrame() function creates a DataFrame from a dictionary, where the keys are column names and the values are lists of data.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Selecting Columns

Accessing specific columns from a DataFrame.

```
ages = df['Age']
```

The code df['Age'] selects the 'Age' column from the DataFrame df and assigns it to the variable ages.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Filtering Rows

Filtering rows based on a condition.

```
filtered_df = df[df['Age'] > 26]
```

The code df[df['Age'] > 26] filters the DataFrame df to include only rows where the 'Age' column value is greater than 26.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Adding a Column

Adding a new column to an existing DataFrame.

```
df['Salary'] = [50000, 60000]
```

The code df['Salary'] = [50000, 60000] adds a new column 'Salary' to the DataFrame df, assigning values to each row.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Dropping a Column

Removing a column from a DataFrame.

```
df = df.drop('Salary', axis=1)
```

The df.drop('Salary', axis=1) removes the 'Salary' column from the DataFrame df. The axis=1 specifies that a column is being dropped.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Handling Missing Data

Filling missing values in a DataFrame.

df['Age'] = df['Age'].fillna(0)

The df['Age'].fillna(0) replaces any NaN (missing) values in the 'Age' column with 0.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

GroupBy Operation

Grouping data by one or more columns and applying an aggregate function.

grouped_df = df.groupby('Name')['Age'].mean()

The df.groupby('Name')['Age'].mean() groups the DataFrame df by the 'Name' column and calculates the mean of the 'Age' column for each group.

pandas

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Merging DataFrames

Combining two DataFrames based on a common column (similar to SQL join).

```
merged_df = pd.merge(df1, df2, on='ID')
```

The pd.merge(df1, df2, on='ID') merges DataFrames df1 and df2 on the 'ID' column, combining rows where the 'ID' matches in both DataFrames.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Pivot Table

Creating a pivot table to summarize data by grouping and applying an aggregate function.

```
pivot_df = df.pivot_table(values='Sales',
index='Region', columns='Product',
aggfunc='sum')
```

The df.pivot_table() creates a pivot table summarizing the 'Sales' column, grouped by 'Region' and 'Product', with the sum as the aggregation function.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Date Parsing

Converting strings to datetime objects in a DataFrame.

```
df['Date'] = pd.to_datetime(df['Date'])
```

The pd.to_datetime(df['Date']) converts the 'Date' column from a string format to a datetime object, allowing for date-specific operations.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

DataFrame Join

Joining two DataFrames on a common index or column.

joined_df = df1.join(df2, on='ID', how='inner')

The df1.join(df2, on='ID', how='inner') joins DataFrame df1 with df2 on the 'ID' column using an inner join, keeping only rows that match in both DataFrames.

pandas

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Applying Functions

Applying a function to each element in a DataFrame column.

```
df['Age_in_Days'] =
df['Age'].apply(lambda x: x*365)
```

The df['Age'].apply(lambda x: x*365) applies a lambda function to each value in the 'Age' column, creating a new column 'Age_in_Days' with the results.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Sorting Data

Sorting the DataFrame by one or more columns.

```
sorted_df = df.sort_values(by='Age',
ascending=False)
```

The df.sort_values(by='Age', ascending=False) sorts the DataFrame df by the 'Age' column in descending order.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

DataFrame Iteration

Iterating over rows of a DataFrame.

```
for index, row in df.iterrows():
print(row['Name'], row['Age'])
```

The df.iterrows() method allows iteration over DataFrame rows. The code prints the 'Name' and 'Age' for each row.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Concatenating DataFrames

Concatenating multiple DataFrames along a particular axis (rows or columns).

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

The pd.concat([df1, df2], axis=0) concatenates DataFrames df1 and df2 along the rows (axis=0).

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Removing Duplicates

Dropping duplicate rows in a DataFrame.

df = df.drop_duplicates()

The df.drop_duplicates() removes any duplicate rows from the DataFrame df.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Reshaping DataFrame

Reshaping a DataFrame from wide to long format (or vice versa).

```
melted_df = df.melt(id_vars=['ID'],
value_vars=['Score1', 'Score2'],
var_name='Exam', value_name='Score')
```

The df.melt() reshapes df from wide to long format by transforming 'Score1' and 'Score2' into a single column 'Score', with a corresponding 'Exam' column indicating the exam type.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

DataFrame Transpose

Transposing rows to columns and vice versa in a DataFrame.

transposed_df = df.T

The df.T transposes the DataFrame df, switching its rows and columns.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Descriptive Statistics

Calculating descriptive statistics like mean, median, min, max, etc., for a DataFrame's columns.

stats = df.describe()

The df.describe() generates summary statistics (count, mean, std, min, 25th percentile, median, 75th percentile, max) for numeric columns in the DataFrame df.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Renaming Columns

Renaming columns in a DataFrame.

df = df.rename(columns={'OldName': 'NewName'})

The df.rename(columns={'OldName': 'NewName'}) renames the column 'OldName' to 'NewName' in the DataFrame df.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Handling NaN Values

Checking for missing values in a DataFrame.

nan_count = df.isna().sum()

The df.isna().sum() returns the count of missing values (NaN) for each column in the DataFrame

df.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Filling NaN with Mean

Filling missing values with the mean of the column.

```
df['Age'] =
df['Age'].fillna(df['Age'].mean())
```

The df['Age'].fillna(df['Age'].mean()) fills missing values in the 'Age' column with the mean of that column.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Dropping Rows with NaN

Dropping rows that contain any missing values.

df = df.dropna()

The df.dropna() removes all rows that contain NaN values in any column from the DataFrame df.

pandas

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Dropping Rows by Condition

Dropping rows based on a condition.

```
df = df[df['Age'] > 20]
```

The df[df['Age'] > 20] filters the DataFrame df to include only rows where the 'Age' column value is greater than 20.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Resetting Index

Resetting the index of a DataFrame after row operations.

df = df.reset_index(drop=True)

The df.reset_index(drop=True) resets the index of the DataFrame df, dropping the old index.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Unique Values

Finding unique values in a column.

unique_ages = df['Age'].unique()

The df['Age'].unique() returns the unique values present in the 'Age' column.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Value Counts

Counting occurrences of each unique value in a column.

age_counts = df['Age'].value_counts()

The df['Age'].value_counts() counts the number of occurrences of each unique value in the 'Age' column.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

String Operations

Applying string operations on a column.

df['Name'] = df['Name'].str.upper()

The df['Name'].str.upper() converts all values in the 'Name' column to uppercase.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Chaining Methods

Chaining multiple DataFrame operations together.

```
df = df[df['Age'] > 20].sort_values(by='Age').reset_index(drop=True)
```

The code filters df for rows where 'Age' is greater than 20, sorts by 'Age', and then resets the index, all in a single line.

pandas

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Datetime Indexing

Setting a datetime column as the index of a DataFrame.

```
df['Date'] = pd.to_datetime(df['Date']) \n
df.set_index('Date', inplace=True)
```

The pd.to_datetime(df['Date']) converts the 'Date' column to datetime format and df.set_index('Date', inplace=True) sets it as the index.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Resampling Time Series

Resampling time series data to a different frequency.

df_resampled = df.resample('M').mean()

The df.resample('M').mean() resamples the DataFrame df to a monthly frequency, calculating the mean for each month.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Rolling Windows

Applying rolling window calculations on time series data.

```
df['RollingMean'] =
df['Sales'].rolling(window=3).mean()
```

The df['Sales'].rolling(window=3).mean() calculates the rolling mean with a window size of 3 for the 'Sales' column, adding the result to a new column 'RollingMean'.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Expanding Window

Applying expanding window calculations on time series data.

```
df['ExpandingSum'] =
df['Sales'].expanding().sum()
```

The df['Sales'].expanding().sum() calculates the cumulative sum over the expanding window for the 'Sales' column, adding the result to a new column 'ExpandingSum'.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Cumulative Operations

Performing cumulative operations such as sum or product.

df['CumulativeSum'] = df['Sales'].cumsum()

The df['Sales'].cumsum() calculates the cumulative sum of the 'Sales' column, storing the result in a new column 'CumulativeSum'.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Correlation Matrix

Calculating the correlation matrix between numerical columns.

correlation = df.corr()

The df.corr() calculates the pairwise correlation between numerical columns in the DataFrame df.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Conditional Replacement

Replacing values in a column based on a condition.

```
df['Status'] = df['Age'].apply(lambda x: 'Adult' if x >= 18 else 'Minor')
```

The code df['Age'].apply(lambda x: 'Adult' if x >= 18 else 'Minor') creates a new column 'Status' that labels each row as 'Adult' or 'Minor' based on the value in the 'Age' column.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

DataFrame Slicing

Selecting a subset of rows and columns from a DataFrame.

subset_df = df.loc[0:5, ['Name', 'Age']]

The df.loc[0:5, ['Name', 'Age']] selects rows 0 to 5 and the columns 'Name' and 'Age' from the DataFrame df.

pandas

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Cross Tabulation

Creating a cross-tabulation (contingency table) of two columns.

cross_tab = pd.crosstab(df['Gender'], df['Purchased'])

The pd.crosstab(df['Gender'], df['Purchased']) creates a cross-tabulation table showing the frequency of occurrences of 'Gender' and 'Purchased' combinations.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

DataFrame Map Function

Mapping values of a Series to another set of values using a mapping function or dictionary.

```
df['Category'] = df['Code'].map({'A': 'Category 1', 'B': 'Category 2'})
```

The df['Code'].map({'A': 'Category 1', 'B': 'Category 2'}) maps values in the 'Code' column to new categories in the 'Category' column based on the provided dictionary.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Merging on Multiple Keys

Merging two DataFrames on multiple keys.

merged_df = pd.merge(df1, df2, on=['Key1', 'Key2'])

The pd.merge(df1, df2, on=['Key1', 'Key2']) merges DataFrames df1 and df2 based on the values of both 'Key1' and 'Key2' columns.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Stacking and Unstacking

Stacking or unstacking (pivoting) the level of a hierarchical index.

```
stacked_df = df.stack() \n
unstacked_df = stacked_df.unstack()
```

The df.stack() compresses a level in the DataFrame columns into the index (stacking), and stacked_df.unstack() reverses this operation (unstacking).

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Window Functions

Applying window functions such as rank, shift, or diff on DataFrame columns.

```
df['Rank'] =
df['Sales'].rank(method='dense',
ascending=False)
```

The df['Sales'].rank(method='dense', ascending=False) assigns ranks to the 'Sales' column with no gaps in rank values (dense rank), ordering them in descending order.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Querying DataFrame

Querying the DataFrame using a SQL-like syntax.

```
filtered_df = df.query('Age > 25 & Gender == "Male"')
```

The df.query('Age > 25 & Gender == "Male"') filters rows based on a condition using SQL-like syntax, selecting rows where 'Age' is greater than 25 and 'Gender' is 'Male'.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Melt Function

Unpivot a DataFrame from wide format to long format using melt.

```
melted_df = df.melt(id_vars='ID',
value_vars=['Q1', 'Q2'], var_name='Quarter',
value_name='Score')
```

The df.melt() function unpivots the DataFrame df by transforming 'Q1' and 'Q2' columns into a single 'Score' column, while the original column names are stored in the 'Quarter' column.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Pivot with Multiple Aggregations

Creating a pivot table with multiple aggregation functions.

```
pivot_df = df.pivot_table(values='Sales',
index='Region', \n
columns='Product', aggfunc=[np.sum, np.mean])
```

The df.pivot_table() creates a pivot table with both sum and mean aggregation functions applied to the 'Sales' column, grouped by 'Region' and 'Product'.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Categorical Data

Converting a column to a categorical data type.

df['Category'] =
df['Category'].astype('category')

The df['Category'].astype('category') converts the 'Category' column to a categorical data type, which is useful for memory efficiency and statistical modeling.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Expanding Mean

Calculating the expanding mean of a column.

```
df['ExpandingMean'] =
df['Sales'].expanding().mean()
```

The df['Sales'].expanding().mean() calculates the expanding mean, which is a cumulative average, of the 'Sales' column and stores it in the 'ExpandingMean' column.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Lag and Lead Operations

Performing lag or lead operations on DataFrame columns.

df['Prev_Sales'] = df['Sales'].shift(1)

The df['Sales'].shift(1) shifts the 'Sales' column down by one row, effectively creating a lag column called 'Prev_Sales' that contains the previous row's sales value.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Merging DataFrames with Indicator

Merging DataFrames while adding an indicator to show the origin of each row.

merged_df = pd.merge(df1, df2, on='ID', how='outer', indicator=True)

The pd.merge(df1, df2, on='ID', how='outer', indicator=True) performs an outer join on 'ID' and adds an indicator column that shows whether a row is from df1, df2, or both.

Shwetank Singh
GritSetGrow - GSGLearn.com

Python 101: Using Pandas

Exploding Lists into Rows

Expanding lists within cells into separate rows.

df = df.explode('Tags')

The df.explode('Tags') function takes a column 'Tags' that contains lists and expands each element of the lists into its own row, replicating the other column values.

Shwetank Singh
GritSetGrow - GSGLearn.com


Thank
you