Third Program

__/__/____

we can extract the .csv, .excel data using read_csv() method which is avialable in pandas.

import pandas as pd

df = pd. read_csv(r'C:\User\Admin\Desktop\Child Labour.csv')

$\downarrow$ $\underline{\text{file Path}}$

raw string, to correctly handled backslashes in file path.

If python program and corresponding .csv file present in the same folder no need to give the file path. we will directly uses filename in the program.

ex:- df = pd. read_csv('Child Labour.csv')

df.shape → it will give row count & column count.

df.columns → it will display the column names.

df.dtypes → it gives list of column names & data types.

df.info() → it gives column names, data types, not null count, rows count, column count, range index(0...24), size

df.describe() → it gives count, mean, standard deviation, min, max, 25%, 50%, 75% of all the data types except object

df.head() → it gives top 5 records.

df.tail() → it gives bottom 5 records.

~~df.h~~ df.head(1) → it gives top 1 record

df.tail() → it gives bottom 1 record.

df.duplicated() → it gives True/False

True means duplicates are present in the file.

df.duplicated().sum() → it gives no of duplicate records

duplicated = df[df.duplicated()] → it displays the ~~duplicat~~ duplicated data.

_/_/_

df['States'].duplicated() → it gives column wise duplicates
df['States'].duplicated().sum() → it gives column wise duplicated count.
duplicated_states = df[df['States'].duplicated()]
duplicated-States → it prints the column wise duplicated data. (State column)
df.drop_duplicates() → it deletes the duplicate records.
df['States'].drop-duplicates() → it deletes the duplicated data from the state column.
a=df['States'].drop-duplicates()
df['States'] → it gives all the data which is present in the state column.
After execution of drop command on States columns you can execute df['States'] we can see duplicates which means that duplicated records still exist in the Data frame.
df-drop-duplicates(Subset=['category of States'])
It deletes the duplicated records based on category of States column.
df → it will give all records. which means that not impact the data frame.
df.drop_duplicates(Subset=['category of States'], inplace=True)

df
inplace=True means modifications directly to the dataframe. no need to assign result to the

any variable. but changes are irreversible unless you have to copy of the original dataframe.

df.isnull() → it gives True | False. True indicates NULL.

df.isnull().sum() → it gives column wise null value count

df.isnull().sum().sum() → it gives total null value count

dfnull = pd.read_csv('Child Labour NULL')

~~df-null().sum.sum()~~ df_null.isnull().sum().sum() → 22

~~df-null~~ df2 = df null.fillna(1) → Filling all null values to 1

~~df-null~~ df2.isnull().sum().sum() → Now it shows 0

df_null

df_null.fillna({'Agriculture': 0, 'Construction': 1})

df_null.isnull().sum() → You can see null values for Agriculture and Construction columns.

To overcome this we need to use below command

df_null_update = df_null.fillna({'Agriculture': 0, 'Construction': 1})

df_null_update

df_null_update.isnull().sum() → Now, we didn't see any null values for Agriculture & Construction

df_null

df_null.fillna({'Agriculture': 0, 'Construction': 1}, inplace=True)

df_null

df_null.isnull().sum()

df_null.dropna(how='all') → drop all null values

df_null.dropna(how='any') → drop any null value

df_null.isnull.sum() → see null values.

df_null_delete = df_null.dropna(how='any')

df-null-delete

'\W' → Replace all non-word characters with an empty string

__/__/____

```
df_null_delete.isnull().sum() → no null values
df_sp = pd.read_csv('child labour special.csv')
df_sp
df_sp['Category of States']
Category_of_States = df_sp['Category of States'].
                        replace(r'\W', '', regex=True)
Category_of_States
df_sp['States']
States = df_sp['States'].replace(r'\W', '', regex=True)
States
States = [States for States in df['States']
                        if States.strip()]
States
print('the total no of rows {}' '\n'
      'the total no of collums {}'.
      format(df.shape[0], df.shape[1]))
df.drop('Category of States', axis=1) → drop the
      Category of States Column.
df
df.drop('category of States', axis=1, inplace=True)
df
df.drop(1, axis=0) → drop the first row.
df
df.drop(1, axis=0, inplace=True)
df
```

```
rename Columns = ['category' if 'Category of States in colnam
                  else 'Agri' if 'Agriculture' in colname
                  else colname for colname in df]
States_match = df['States'] == 'Andhra Pradesh'
States_match → gives True or False.
States_match = df[df['States'] == 'Andhra Pradesh']
States_match
States_not_match = df[df['States'] != 'Andhra Pradesh']
States_not_match
df['States'].value_counts() → gives the count of States
df['States'].value_counts().plot()
df['States'].value_counts().plot(kind = bar)
df['States'].unique() → gives unique records
df['States'].nunique() → gives unique record count
df['Manufacturing'] = df['Manufacturing'].replace('9.9', '99
df['Manufacturing'] = df['Manufacturing'].astype('float')
df['Manufacturing']
df.dtypes
```

Converting the datatype from object to float.