Streaming Service Data- Data Cleaning and Preprocessing

In this notebook, we focus on data preparation, cleaning, and preprocessing for a Streaming Service Dataset, which contains user interaction logs with video content. This dataset is useful for understanding user behavior, content performance, and engagement patterns.

Proper data cleaning and transformation are essential for generating reliable insights and building predictive models. In this notebook, we address common data issues such as missing values, duplicates, and inconsistent formats, and we create new derived features to support further analysis.

We begin by importing the key Python libraries required for handling and exploring the dataset:

pandas for working with structured data.

numpy for numerical and array-based operations.

os for accessing and managing file paths.

```
import pandas as pd
import numpy as np
import os
```

Project Directory Setup (Text Description)

To ensure an organized structure for the streaming service data project, we first determine the current working directory and navigate one level up to identify the project's root directory. From there, we define a clear folder structure for handling data, results, and documentation.

The following folders are defined:

data – This is the main directory for all datasets used in the project.

raw – Contains the original, unprocessed (raw) streaming service data.

processed – Stores the cleaned and transformed datasets after preprocessing steps are completed.

results – This folder is used to store output files such as visualizations, reports, and any result files generated during analysis.

docs– Contains project documentation, including notes, markdown files, or final written reports.

After defining the paths for each folder, the program ensures that all these directories exist. If any of them are missing, they are created automatically using os.makedirs() with the exist_ok=True parameter to avoid overwriting existing folders.

```
# Get working directory
current_dir=os.getcwd()

# Go one directory up to root directory
project_root_dir=os.path.dirname(current_dir)

# Define paths to the data folders
data_dir=os.path.join(project_root_dir,'data')
raw_dir=os.path.join(data_dir,'raw')
processed_dir=os.path.join(data_dir,'processed')

# Define paths to results folders
result_dir=os.path.join(project_root_dir,'results')

# Define paths to Docs folder
docs_dir=os.path.join(project_root_dir,'docs')

# Create a directories if they do not exists

os.makedirs(raw_dir,exist_ok=True)
os.makedirs(processed_dir,exist_ok=True)
os.makedirs(result_dir,exist_ok=True)
os.makedirs(docs_dir,exist_ok=True)
```

Loading the Streaming Data

Now that the folders are ready, we move on to loading the actual data into our program.

We start by creating the full path to the dataset file called streaming_service_data.csv, which is stored in the **raw folder**. After that, we use a function from the Pandas library to read the

data file and store it in a table-like format called a DataFrame, which makes it easier to work with.

Finally, we display the first few rows of the data to make sure everything loaded correctly and to get a quick look at what's inside—such as column names and sample values.

This step helps us confirm that the data is available and ready to be cleaned and analyzed.

```
stream_data_filename=os.path.join(raw_dir,"streaming_service_data.csv")
stream_df=pd.read_csv(stream_data_filename)
stream_df.head()
```

	User_ID	User_Name	Join_Date	Last_Login	Monthly_Price	Watch_Hours	Favorite_Genre	A
0	2518	Amber	5/15/2023	12/13/2024	7.99	49	Action	3
1	6430	Patrick	4/3/2023	12/15/2024	7.99	161	Drama	1
2	1798	Robert	8/2/2023	12/14/2024	11.99	87	Action	2
3	5255	Cole	1/31/2023	12/2/2024	15.99	321	Sci-Fi	1
4	2854	Jamie	6/6/2023	12/15/2024	11.99	386	Documentary	1

Check the Size of the Data

This shows how many rows and how many columns the data has.

```
stream_df.shape
```

```
(1000, 23)
```

Check Data Information

How many rows and columns there are

What type of data is in each column (like numbers or text)

How many values are missing in each column

```
stream_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 23 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   User_ID                    1000 non-null   int64
 1   User_Name                  1000 non-null   object
 2   Join_Date                  1000 non-null   object
 3   Last_Login                 1000 non-null   object
 4   Monthly_Price              1000 non-null   float64
 5   Watch_Hours                1000 non-null   int64
 6   Favorite_Genre             1000 non-null   object
 7   Active_Devices             1000 non-null   int64
 8   Profile_Count              1000 non-null   int64
 9   Parental_Controls          1000 non-null   bool
 10  Total_Movies_Watched       1000 non-null   int64
 11  Total_Series_Watched       1000 non-null   int64
 12  Country                    1000 non-null   object
 13  Payment_Method             1000 non-null   object
 14  Language_Preference        1000 non-null   object
 15  Recommended_Content_Count  1000 non-null   int64
 16  Average_Rating_Given       1000 non-null   float64
 17  Has_Downloaded_Content     1000 non-null   bool
 18  Membership_Status          1000 non-null   object
 19  Loyalty_Points             1000 non-null   int64
 20  First_Device_Used          1000 non-null   object
 21  Age_Group                  1000 non-null   object
 22  Primary_Watch_Time         1000 non-null   object
dtypes: bool(2), float64(2), int64(8), object(11)
memory usage: 166.1+ KB
```

Data cleaning

understanding data set

```
stream_df.columns
```

```
Index(['User_ID', 'User_Name', 'Join_Date', 'Last_Login', 'Monthly_Price',
       'Watch_Hours', 'Favorite_Genre', 'Active_Devices', 'Profile_Count',
       'Parental_Controls', 'Total_Movies_Watched', 'Total_Series_Watched',
       'Country', 'Payment_Method', 'Language_Preference',
```

```
 'Recommended_Content_Count', 'Average_Rating_Given',
 'Has_Downloaded_Content', 'Membership_Status', 'Loyalty_Points',
 'First_Device_Used', 'Age_Group', 'Primary_Watch_Time'],
dtype='object')
```

Dataset Description

This dataset contains user information from a streaming service. Below is a description of each column:

Column Name	Description
User_ID	Unique identifier for each user.
User_Name	The name or username of the user.
Join_Date	The date when the user created their account.
Last_Login	The most recent date the user logged into the platform.
Monthly_Price	The monthly subscription price paid by the user.
Watch_Hours	Total number of hours the user has spent watching content.
Favorite_Genre	The genre most frequently watched by the user such as Action, Comedy, Documentary, Drama, Horror, Romance,Sci-Fi.
Active_Devices	Number of devices currently linked to the user account.
Profile_Count	Number of user profiles under one account.
Parental_Controls	Whether parental control settings are enabled such as False, True.
Total_Movies_Watched	Total number of movies watched by the user.
Total_Series_Watched	Total number of series watched by the user.
Country	The country of residence of the user are Australia, Canada, France, Germany, India, UK, USA.
Payment_Method	The payment method used (e.g., Credit Card, Cryptocurrency, Debit Card, PayPal).
Language_Preference	The preferred language of content/interface i have is English, French, German, Hindi, Mandarin, Spanish.
Recommended_Content_Count	Number of recommended contents the user has received.
Average_Rating_Given	Average rating the user has given to content.

Column Name	Description
Has_Downloaded_Content	Whether the user has downloaded content false and true.
Membership_Status	Indicates if the user has an active or inactive membership.
Loyalty_Points	Loyalty points accumulated by the user.
First_Device_Used	The first device used to access the platform used are Desktop, Laptop, Smart TV, Smartphone, Tablet.
Age_Group	The age group the user falls into (e.g., Teen, Adult, Senior).
Primary_Watch_Time	The time of day the user most frequently watches content used was Afternoon', Evening, Late Night, Morning.

```
np.unique(stream_df.Monthly_Price.to_list())
```

```
array([ 7.99, 11.99, 15.99])
```

```
np.unique(stream_df.Watch_Hours.to_list())
```

```
array([ 10,  11,  12,  13,  14,  15,  16,  17,  18,  19,  20,  21,  22,
        23,  24,  25,  26,  27,  28,  29,  30,  32,  33,  34,  35,  36,
        37,  38,  39,  40,  42,  43,  44,  45,  46,  47,  48,  49,  50,
        52,  53,  54,  55,  56,  57,  58,  59,  61,  62,  63,  64,  65,
        68,  69,  70,  71,  73,  74,  75,  76,  77,  78,  81,  82,  83,
        84,  85,  87,  88,  89,  90,  91,  92,  93,  96,  97,  98,  99,
       100, 101, 102, 103, 104, 105, 106, 108, 109, 110, 112, 113, 114,
       115, 116, 118, 119, 120, 121, 122, 123, 124, 125, 127, 128, 129,
       130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 144,
       145, 146, 147, 148, 150, 152, 153, 154, 155, 157, 158, 159, 160,
       161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173,
       174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186,
       187, 188, 190, 191, 192, 193, 195, 196, 197, 198, 200, 201, 202,
       203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 214, 215, 216,
       217, 219, 220, 221, 222, 223, 224, 225, 227, 231, 233, 234, 235,
       236, 237, 238, 239, 240, 241, 242, 243, 245, 246, 247, 248, 249,
       250, 251, 252, 253, 254, 256, 257, 258, 259, 262, 263, 264, 266,
       267, 268, 270, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281,
       282, 283, 285, 286, 287, 289, 290, 291, 292, 293, 294, 295, 296,
```

```
       297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309,
       311, 312, 313, 315, 316, 317, 318, 319, 321, 322, 325, 326, 327,
       328, 329, 330, 331, 332, 335, 336, 337, 338, 339, 341, 342, 343,
       344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356,
       357, 358, 359, 360, 361, 362, 364, 365, 366, 367, 368, 369, 370,
       371, 373, 374, 375, 376, 378, 379, 380, 381, 382, 383, 384, 385,
       386, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 399, 401,
       404, 405, 406, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417,
       418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 429, 432, 433,
       435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447,
       448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 459, 460, 461,
       462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474,
       475, 477, 478, 479, 480, 482, 483, 484, 485, 486, 487, 490, 491,
       492, 493, 494, 495, 496, 498, 499, 500])
```

```
np.unique(stream_df.Favorite_Genre.to_list())
```

```
array(['Action', 'Comedy', 'Documentary', 'Drama', 'Horror', 'Romance',
       'Sci-Fi'], dtype='<U11')
```

```
np.unique(stream_df.Active_Devices.to_list())
```

```
array([1, 2, 3, 4, 5])
```

```
np.unique(stream_df.Profile_Count.to_list())
```

```
array([1, 2, 3, 4, 5, 6])
```

```
np.unique(stream_df.Parental_Controls.to_list())
```

```
array([False,  True])
```

```
np.unique(stream_df.Total_Movies_Watched.to_list())
```

```
array([  12,   13,   15,   17,   18,   20,   22,   25,   26,   27,   28,
         29,   30,   31,   33,   38,   39,   40,   42,   44,   49,   50,
         51,   52,   53,   56,   57,   59,   60,   61,   62,   63,   65,
         67,   69,   70,   73,   75,   76,   77,   79,   80,   84,   85,
```

```
 89,   90,   91,   92,   93,   95,   96,   97,  102,  103,  104,
105,  107,  108,  109,  112,  113,  115,  118,  120,  123,  127,
131,  132,  133,  135,  138,  139,  141,  142,  144,  146,  148,
150,  151,  154,  155,  156,  158,  159,  160,  161,  162,  163,
168,  170,  174,  175,  177,  178,  179,  180,  181,  182,  183,
184,  185,  186,  187,  188,  190,  191,  192,  194,  196,  198,
201,  202,  203,  205,  206,  207,  208,  209,  213,  214,  216,
217,  218,  221,  222,  225,  226,  228,  229,  230,  231,  232,
233,  235,  236,  237,  238,  239,  240,  242,  244,  245,  246,
247,  253,  255,  256,  257,  260,  263,  264,  265,  266,  267,
268,  271,  273,  274,  276,  277,  280,  284,  285,  286,  287,
291,  292,  295,  297,  302,  304,  305,  308,  309,  310,  312,
313,  314,  315,  316,  317,  318,  319,  321,  322,  323,  324,
326,  327,  328,  330,  331,  334,  336,  338,  340,  341,  343,
344,  346,  347,  348,  349,  350,  352,  354,  355,  356,  357,
358,  359,  361,  362,  364,  366,  367,  369,  370,  374,  376,
377,  378,  379,  380,  381,  382,  385,  386,  387,  389,  390,
391,  392,  393,  394,  396,  399,  400,  404,  406,  407,  409,
410,  411,  412,  413,  415,  416,  417,  418,  420,  421,  423,
424,  426,  428,  429,  430,  431,  432,  434,  435,  439,  442,
445,  446,  450,  451,  453,  455,  456,  458,  459,  461,  463,
464,  465,  466,  467,  468,  469,  471,  472,  474,  475,  476,
477,  479,  481,  482,  483,  484,  485,  487,  488,  489,  490,
491,  492,  493,  494,  498,  499,  501,  502,  503,  505,  506,
507,  509,  510,  512,  513,  514,  515,  517,  518,  520,  521,
522,  523,  524,  525,  526,  527,  528,  532,  535,  536,  537,
539,  541,  542,  543,  544,  546,  549,  551,  552,  557,  559,
560,  561,  562,  563,  564,  568,  569,  570,  571,  573,  577,
578,  579,  580,  581,  582,  583,  585,  586,  587,  588,  589,
590,  591,  592,  593,  594,  597,  600,  601,  603,  604,  605,
606,  607,  608,  609,  611,  614,  615,  616,  617,  619,  623,
624,  625,  626,  631,  632,  633,  634,  636,  637,  641,  645,
646,  647,  648,  651,  653,  655,  657,  659,  660,  664,  665,
666,  667,  668,  669,  670,  672,  676,  679,  680,  682,  683,
684,  685,  687,  688,  693,  694,  695,  696,  697,  698,  699,
700,  701,  702,  704,  706,  707,  708,  709,  710,  711,  713,
714,  717,  718,  720,  722,  724,  725,  726,  727,  730,  732,
734,  735,  737,  738,  741,  742,  743,  744,  746,  748,  749,
750,  751,  752,  753,  754,  755,  757,  758,  759,  760,  763,
765,  767,  769,  770,  773,  774,  775,  776,  778,  779,  780,
781,  782,  783,  784,  785,  786,  788,  791,  792,  793,  796,
797,  799,  800,  801,  802,  803,  804,  805,  807,  808,  811,
813,  817,  818,  819,  820,  821,  824,  826,  827,  829,  830,
```

```
       831,  832,  833,  836,  837,  838,  841,  842,  843,  844,  848,
       849,  851,  853,  855,  857,  858,  860,  861,  862,  865,  866,
       869,  871,  872,  874,  875,  879,  881,  882,  883,  885,  886,
       887,  889,  890,  891,  895,  896,  897,  899,  900,  902,  903,
       905,  906,  907,  908,  909,  914,  918,  919,  920,  922,  923,
       925,  928,  929,  932,  936,  938,  939,  940,  942,  943,  944,
       945,  946,  950,  951,  952,  958,  959,  961,  962,  964,  968,
       969,  970,  973,  975,  976,  977,  981,  983,  985,  987,  988,
       989,  990,  994,  995,  996,  997,  998,  999, 1000])
```

```
np.unique(stream_df.Total_Series_Watched.to_list())
```

```
array([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,  13,
        14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,  26,
        27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,  39,
        40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,  52,
        53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,  65,
        66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,  78,
        79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,  91,
        92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103, 104,
       105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117,
       118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130,
       131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143,
       144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156,
       157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 168, 169, 170,
       171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183,
       184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196,
       197, 198, 199, 200])
```

```
np.unique(stream_df.Country.to_list())
```

```
array(['Australia', 'Canada', 'France', 'Germany', 'India', 'UK', 'USA'],
      dtype='<U9')
```

```
np.unique(stream_df.Payment_Method.to_list())
```

```
array(['Credit Card', 'Cryptocurrency', 'Debit Card', 'PayPal'],
      dtype='<U14')
```

```
np.unique(stream_df.Language_Preference.to_list())
```

```
array(['English', 'French', 'German', 'Hindi', 'Mandarin', 'Spanish'],
      dtype='<U8')
```

```
np.unique(stream_df.Recommended_Content_Count.to_list())
```

```
array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,
        13,  14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,
        26,  27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,
        52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,
        65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,
        78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,
        91,  92,  93,  94,  95,  96,  97,  98,  99, 100])
```

```
np.unique(stream_df.Average_Rating_Given.to_list())
```

```
array([3. , 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 4. , 4.1, 4.2,
       4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 5. ])
```

```
np.unique(stream_df.Has_Downloaded_Content.to_list())
```

```
array([False,  True])
```

```
np.unique(stream_df.Average_Rating_Given.to_list())
```

```
array([3. , 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 4. , 4.1, 4.2,
       4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9, 5. ])
```

```
np.unique(stream_df.Membership_Status.to_list())
```

```
array(['Active'], dtype='<U6')
```

```
np.unique(stream_df.Loyalty_Points.to_list())
```

```
array([   3,    4,   15,   17,   28,   30,   33,   47,   48,   53,   55,
         60,   62,   68,   72,   73,   74,   77,   92,   96,   97,   99,
        105,  106,  110,  111,  119,  125,  130,  136,  144,  147,  150,
        164,  165,  167,  168,  172,  173,  185,  188,  212,  213,  214,
        215,  218,  223,  225,  228,  234,  237,  239,  242,  244,  245,
        249,  255,  256,  260,  274,  290,  292,  296,  314,  318,  333,
        336,  340,  342,  344,  353,  354,  368,  371,  373,  380,  388,
        396,  398,  402,  412,  414,  416,  421,  423,  424,  428,  433,
        444,  447,  448,  450,  459,  460,  473,  476,  484,  486,  494,
        496,  510,  513,  525,  527,  540,  544,  546,  547,  548,  561,
        564,  571,  581,  583,  585,  595,  599,  608,  613,  615,  628,
        631,  633,  639,  647,  650,  658,  668,  670,  674,  681,  708,
        710,  711,  718,  727,  728,  732,  745,  746,  747,  749,  754,
        755,  756,  771,  772,  773,  775,  782,  783,  784,  786,  790,
        804,  808,  809,  827,  833,  836,  837,  851,  860,  863,  868,
        874,  876,  877,  882,  888,  905,  906,  913,  933,  935,  941,
        944,  945,  947,  948,  952,  954,  957,  959,  965,  967,  987,
        993,  995,  996, 1000, 1005, 1013, 1016, 1017, 1025, 1037, 1040,
       1042, 1050, 1051, 1058, 1065, 1068, 1072, 1080, 1094, 1095, 1099,
       1108, 1110, 1121, 1127, 1129, 1135, 1150, 1153, 1155, 1159, 1172,
       1176, 1180, 1185, 1201, 1206, 1216, 1228, 1231, 1238, 1243, 1250,
       1252, 1254, 1256, 1258, 1261, 1271, 1275, 1281, 1285, 1289, 1290,
       1293, 1294, 1298, 1299, 1303, 1309, 1311, 1312, 1317, 1325, 1327,
       1330, 1340, 1341, 1348, 1351, 1357, 1360, 1363, 1364, 1365, 1367,
       1368, 1375, 1382, 1392, 1394, 1404, 1408, 1411, 1413, 1416, 1422,
       1423, 1426, 1428, 1429, 1431, 1436, 1442, 1454, 1456, 1457, 1461,
       1485, 1486, 1500, 1504, 1506, 1508, 1510, 1512, 1518, 1520, 1522,
       1525, 1526, 1529, 1531, 1535, 1536, 1538, 1539, 1542, 1544, 1546,
       1547, 1556, 1559, 1563, 1572, 1577, 1580, 1581, 1584, 1587, 1594,
       1610, 1626, 1628, 1634, 1638, 1651, 1652, 1656, 1657, 1665, 1672,
       1674, 1683, 1690, 1692, 1697, 1702, 1704, 1708, 1713, 1721, 1734,
       1744, 1752, 1756, 1758, 1764, 1765, 1782, 1784, 1785, 1786, 1790,
       1792, 1808, 1813, 1821, 1823, 1824, 1828, 1835, 1836, 1849, 1850,
       1856, 1858, 1870, 1872, 1876, 1882, 1901, 1906, 1910, 1911, 1912,
       1917, 1925, 1926, 1934, 1938, 1946, 1952, 1955, 1960, 1961, 1965,
       1972, 1976, 1982, 1984, 1986, 1989, 1991, 2002, 2015, 2022, 2023,
       2029, 2039, 2043, 2050, 2067, 2070, 2083, 2084, 2086, 2088, 2089,
       2097, 2098, 2102, 2113, 2124, 2130, 2131, 2132, 2138, 2141, 2156,
       2159, 2163, 2164, 2175, 2193, 2209, 2213, 2218, 2229, 2234, 2238,
       2242, 2259, 2261, 2284, 2288, 2291, 2299, 2318, 2323, 2327, 2328,
       2330, 2331, 2336, 2341, 2344, 2346, 2356, 2370, 2377, 2381, 2384,
       2385, 2387, 2388, 2390, 2394, 2395, 2397, 2400, 2406, 2407, 2409,
       2416, 2418, 2428, 2432, 2440, 2444, 2448, 2460, 2461, 2465, 2488,
```

2508, 2517, 2521, 2523, 2534, 2535, 2536, 2538, 2542, 2547, 2554,
2557, 2559, 2560, 2561, 2562, 2568, 2571, 2572, 2575, 2576, 2580,
2583, 2596, 2597, 2600, 2610, 2615, 2620, 2624, 2634, 2636, 2643,
2644, 2647, 2652, 2657, 2666, 2670, 2676, 2678, 2695, 2706, 2714,
2725, 2727, 2728, 2731, 2741, 2749, 2757, 2759, 2760, 2761, 2779,
2780, 2785, 2790, 2798, 2805, 2812, 2821, 2824, 2830, 2835, 2841,
2845, 2853, 2856, 2858, 2864, 2866, 2867, 2871, 2872, 2878, 2886,
2891, 2897, 2901, 2910, 2914, 2919, 2922, 2925, 2927, 2928, 2932,
2933, 2936, 2938, 2941, 2953, 2964, 2965, 2969, 2978, 2984, 2989,
3003, 3007, 3009, 3015, 3020, 3027, 3028, 3037, 3039, 3040, 3053,
3059, 3062, 3069, 3078, 3079, 3081, 3083, 3085, 3091, 3100, 3113,
3118, 3124, 3130, 3138, 3151, 3152, 3157, 3161, 3165, 3173, 3178,
3179, 3182, 3183, 3197, 3199, 3201, 3206, 3211, 3213, 3221, 3247,
3254, 3264, 3276, 3277, 3278, 3281, 3282, 3288, 3289, 3290, 3307,
3308, 3313, 3314, 3325, 3332, 3334, 3349, 3354, 3356, 3366, 3379,
3398, 3408, 3411, 3421, 3424, 3425, 3426, 3428, 3432, 3433, 3437,
3441, 3445, 3448, 3452, 3455, 3462, 3463, 3468, 3476, 3488, 3496,
3499, 3505, 3508, 3510, 3515, 3516, 3517, 3520, 3542, 3552, 3556,
3566, 3568, 3574, 3578, 3589, 3596, 3599, 3604, 3607, 3610, 3616,
3617, 3626, 3630, 3633, 3641, 3645, 3648, 3655, 3659, 3663, 3674,
3679, 3689, 3696, 3697, 3698, 3702, 3708, 3711, 3712, 3714, 3726,
3730, 3741, 3751, 3758, 3761, 3763, 3773, 3775, 3787, 3788, 3791,
3807, 3815, 3817, 3823, 3828, 3834, 3836, 3840, 3842, 3843, 3847,
3848, 3849, 3859, 3865, 3898, 3910, 3913, 3918, 3923, 3925, 3928,
3930, 3935, 3941, 3942, 3953, 3966, 3975, 3980, 3983, 4004, 4008,
4012, 4015, 4020, 4031, 4037, 4048, 4056, 4062, 4068, 4070, 4072,
4076, 4083, 4085, 4108, 4111, 4114, 4116, 4117, 4125, 4127, 4130,
4131, 4133, 4134, 4147, 4155, 4159, 4162, 4164, 4170, 4176, 4177,
4185, 4193, 4194, 4200, 4201, 4204, 4216, 4219, 4221, 4235, 4236,
4243, 4245, 4260, 4264, 4269, 4273, 4291, 4293, 4307, 4308, 4311,
4317, 4322, 4327, 4329, 4332, 4333, 4337, 4338, 4348, 4356, 4358,
4361, 4377, 4378, 4380, 4397, 4400, 4409, 4414, 4421, 4422, 4426,
4435, 4438, 4445, 4450, 4456, 4465, 4477, 4486, 4497, 4501, 4503,
4504, 4505, 4509, 4510, 4511, 4513, 4517, 4518, 4528, 4531, 4537,
4542, 4552, 4555, 4556, 4565, 4566, 4569, 4570, 4575, 4586, 4588,
4589, 4599, 4602, 4626, 4632, 4633, 4635, 4641, 4644, 4646, 4647,
4650, 4651, 4655, 4659, 4662, 4672, 4673, 4674, 4685, 4691, 4702,
4714, 4719, 4726, 4729, 4732, 4742, 4763, 4777, 4783, 4789, 4792,
4798, 4799, 4800, 4808, 4815, 4817, 4820, 4835, 4851, 4867, 4868,
4873, 4879, 4883, 4884, 4889, 4905, 4906, 4919, 4920, 4922, 4927,
4928, 4934, 4935, 4938, 4942, 4963, 4971, 4972, 4976, 4980, 4990])

Grouping Watch Hours into Ranges

We want to group users based on how many hours they watched. To do this:

We create bins that define hour ranges: 0–50, 51–100, 101–200, 201–300, and 301–500.

Using these bins, we make a new column called Watch_Hour_Group in the data. Each user is assigned to one of these groups based on their total watch hours.

Then, we count how many users fall into each watch hour group.

Finally, we print out the counts to see how users are distributed across these groups.

```
bins = [0, 50, 100, 200, 300, 500]
labels = ['0-50 ', '51-100 ', '101-200 ', '201-300 ', '301-500 ']
# Create a new column for hour groups
stream_df['Watch_Hour_Group'] = pd.cut(stream_df['Watch_Hours'], bins=bins, labels=labels, i

# Count users in each group
hrs_counts = stream_df['Watch_Hour_Group'].value_counts().sort_index()

# Display the counts
print(hrs_counts)
```

```
Watch_Hour_Group
0-50          93
51-100        88
101-200      210
201-300      201
301-500      408
Name: count, dtype: int64
```

```
loyalty_bins = [0, 1000, 2000, 3000, 4000, 4990]
loyalty_labels = ['0-999 ', '1000-1999 ', '2000-2999 ', '3000-3999 ', '4000-4990 ']

# Group Loyalty Points
stream_df['Loyalty_Point_Group'] = pd.cut(
    stream_df['Loyalty_Points'],
    bins=loyalty_bins,
    labels=loyalty_labels,
    include_lowest=True
)

# Count users per group
```

```
loyalty_counts = stream_df['Loyalty_Point_Group'].value_counts().sort_index()
loyalty_counts
```

```
Loyalty_Point_Group
0-999         207
1000-1999     215
2000-2999     196
3000-3999     191
4000-4990     191
Name: count, dtype: int64
```

```
stream_df.columns
```

```
Index(['User_ID', 'User_Name', 'Join_Date', 'Last_Login', 'Monthly_Price',
       'Watch_Hours', 'Favorite_Genre', 'Active_Devices', 'Profile_Count',
       'Parental_Controls', 'Total_Movies_Watched', 'Total_Series_Watched',
       'Country', 'Payment_Method', 'Language_Preference',
       'Recommended_Content_Count', 'Average_Rating_Given',
       'Has_Downloaded_Content', 'Membership_Status', 'Loyalty_Points',
       'First_Device_Used', 'Age_Group', 'Primary_Watch_Time',
       'Watch_Hour_Group', 'Loyalty_Point_Group'],
      dtype='object')
```

Grouping Users by Age Category

We want to give friendly names to different age groups of users.

We create a map that links each age range to a label:

18-24→ Young Streamers

25-34 → Adult Streamers

35-44 → Mature Streamers

45-54 → Older Streamers

55+ → Senior Streamers

We use this map to create a new column called Age_Category in the data, replacing age ranges with these easy-to-understand labels.

```
age_map = {
    '18-24': 'Young Streamers',
    '25-34': 'Adult Streamers',
    '35-44': 'Mature Streamers',
    '45-54': 'Older Streamers',
    '55+': 'Senior Streamers'
}

stream_df['Age_Category'] = stream_df['Age_Group'].map(age_map)
stream_df['Age_Category'].unique()
```

```
array(['Mature Streamers', 'Adult Streamers', 'Young Streamers',
       'Older Streamers', 'Senior Streamers'], dtype=object)
```

Final dataset and Save cleaning stream data set in csv

```
stream_df
```

	User_ID	User_Name	Join_Date	Last_Login	Monthly_Price	Watch_Hours	Favorite_Genre
0	2518	Amber	5/15/2023	12/13/2024	7.99	49	Action
1	6430	Patrick	4/3/2023	12/15/2024	7.99	161	Drama
2	1798	Robert	8/2/2023	12/14/2024	11.99	87	Action
3	5255	Cole	1/31/2023	12/2/2024	15.99	321	Sci-Fi
4	2854	Jamie	6/6/2023	12/15/2024	11.99	386	Documentary
...	...	...	...	...	...	...	...
995	6878	Mary	4/26/2024	11/27/2024	7.99	136	Sci-Fi
996	5681	Ryan	5/8/2024	12/8/2024	11.99	159	Romance
997	4448	David	3/23/2024	12/16/2024	11.99	99	Sci-Fi
998	5795	John	11/25/2023	12/13/2024	11.99	157	Action
999	5320	Katherine	9/28/2023	12/9/2024	11.99	123	Documentary

```
stream_df['Watch_Hours'] = pd.to_numeric(stream_df['Watch_Hours'], errors='coerce')
stream_df['Loyalty_Points'] = pd.to_numeric(stream_df['Loyalty_Points'], errors='coerce')
```

```
stream_df.to_csv("cleaned_stream_data.csv", index=False, encoding='utf-8')
```