# Data Normalization Techniques

Estimated time needed: **30** minutes

In this lab, you will focus on data normalization. This includes identifying compensation-related columns, applying normalization techniques, and visualizing the data distributions.

## Objectives

In this lab, you will perform the following:

- Identify duplicate rows and remove them.
- Check and handle missing values in key columns.
- Identify and normalize compensation-related columns.
- Visualize the effect of normalization techniques on data distributions.

---

## Hands on Lab

### Step 1: Install and Import Libraries

In [12]:
```
!pip install pandas
```

```
Requirement already satisfied: pandas in /opt/conda/lib/python3.12/site-packages (2.3.0)
Requirement already satisfied: numpy>=1.26.0 in /opt/conda/lib/python3.12/site-packages (from panda
s) (2.3.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/conda/lib/python3.12/site-packages (fr
om pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.12/site-packages (from pandas)
(2024.2)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.12/site-packages (from panda
s) (2025.2)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-packages (from python-date
util>=2.8.2->pandas) (1.17.0)
```

In [13]:
```
!pip install matplotlib
```

```
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.12/site-packages (3.10.3)
Requirement already satisfied: contourpy>=1.0.1 in /opt/conda/lib/python3.12/site-packages (from mat
plotlib) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.12/site-packages (from matplot
lib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.12/site-packages (from ma
tplotlib) (4.58.4)
Requirement already satisfied: kiwisolver>=1.3.1 in /opt/conda/lib/python3.12/site-packages (from ma
tplotlib) (1.4.8)
Requirement already satisfied: numpy>=1.23 in /opt/conda/lib/python3.12/site-packages (from matplotl
ib) (2.3.0)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.12/site-packages (from matp
lotlib) (24.2)
Requirement already satisfied: pillow>=8 in /opt/conda/lib/python3.12/site-packages (from matplotli
b) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /opt/conda/lib/python3.12/site-packages (from mat
plotlib) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.12/site-packages (from
matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-packages (from python-date
util>=2.7->matplotlib) (1.17.0)
```

In [14]:
```python
import pandas as pd
import matplotlib.pyplot as plt
```

## Step 2: Load the Dataset into a DataFrame

We use the `pandas.read_csv()` function for reading CSV files. However, in this version of the lab, which operates on JupyterLite, the dataset needs to be downloaded to the interface using the provided code below.

The functions below will download the dataset into your browser:

In [15]:
```python
file_path = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/n01PQ9pSmiRX6520flu

df = pd.read_csv(file_path)

# Display the first few rows to check if data is loaded correctly
print(df.head())
```

```
   ResponseId                      MainBranch                 Age  \
0           1  I am a developer by profession  Under 18 years old
1           2  I am a developer by profession    35-44 years old
2           3  I am a developer by profession    45-54 years old
3           4           I am learning to code    18-24 years old
4           5  I am a developer by profession    18-24 years old

            Employment RemoteWork   Check  \
0  Employed, full-time     Remote  Apples
1  Employed, full-time     Remote  Apples
2  Employed, full-time     Remote  Apples
3   Student, full-time        NaN  Apples
4   Student, full-time        NaN  Apples

                                  CodingActivities  \
0                                            Hobby
1  Hobby;Contribute to open-source projects;Other...
2  Hobby;Contribute to open-source projects;Other...
3                                              NaN
4                                              NaN

                                       EdLevel  \
0                       Primary/elementary school
1      Bachelor's degree (B.A., B.S., B.Eng., etc.)
2    Master's degree (M.A., M.S., M.Eng., MBA, etc.)
3  Some college/university study without earning ...
4  Secondary school (e.g. American high school, G...

                                    LearnCode  \
0                           Books / Physical media
1  Books / Physical media;Colleague;On the job tr...
2  Books / Physical media;Colleague;On the job tr...
3  Other online resources (e.g., videos, blogs, f...
4  Other online resources (e.g., videos, blogs, f...

                                  LearnCodeOnline  ... JobSatPoints_6  \
0                                             NaN  ...            NaN
1  Technical documentation;Blogs;Books;Written Tu...  ...            0.0
2  Technical documentation;Blogs;Books;Written Tu...  ...            NaN
3  Stack Overflow;How-to videos;Interactive tutorial  ...            NaN
4  Technical documentation;Blogs;Written Tutorial...  ...            NaN

  JobSatPoints_7 JobSatPoints_8 JobSatPoints_9 JobSatPoints_10  \
0            NaN            NaN            NaN             NaN
1            0.0            0.0            0.0             0.0
2            NaN            NaN            NaN             NaN
3            NaN            NaN            NaN             NaN
4            NaN            NaN            NaN             NaN

  JobSatPoints_11           SurveyLength SurveyEase ConvertedCompYearly JobSat
0             NaN                    NaN        NaN                 NaN    NaN
1             0.0                    NaN        NaN                 NaN    NaN
2             NaN  Appropriate in length       Easy                 NaN    NaN
3             NaN               Too long       Easy                 NaN    NaN
4             NaN             Too short       Easy                 NaN    NaN

[5 rows x 114 columns]
```

In [16]:
```
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/n01PQ9pSmiRX65
```

## Section 1: Handling Duplicates

Task 1: Identify and remove duplicate rows.

In [17]:
```
## Write your code here
# Count and display initial duplicate rows
num_duplicate_rows = df.duplicated().sum()
print(f"Number of duplicate rows in the dataset: {num_duplicate_rows}")

# Remove duplicate rows
df_cleaned = df.drop_duplicates()

# Verify removal
num_duplicates_after_removal = df_cleaned.duplicated().sum()
```

```python
print("\nDataFrame after removing duplicates:")
print(df_cleaned)
print(f"\nNumber of rows after removal: {len(df_cleaned)}")
print(f"Duplicates remaining: {num_duplicates_after_removal}")

if num_duplicates_after_removal == 0:
    print("Verification successful: All exact duplicate rows have been removed.")
else:
    print("Verification failed: Some duplicate rows still exist.")

# If you want to update your original DataFrame:
# df = df_cleaned
```

```
Number of duplicate rows in the dataset: 0

DataFrame after removing duplicates:
       ResponseId                      MainBranch                 Age  \
0               1  I am a developer by profession  Under 18 years old
1               2  I am a developer by profession     35-44 years old
2               3  I am a developer by profession     45-54 years old
3               4           I am learning to code     18-24 years old
4               5  I am a developer by profession     18-24 years old
...           ...                             ...                 ...
65432       65433  I am a developer by profession     18-24 years old
65433       65434  I am a developer by profession     25-34 years old
65434       65435  I am a developer by profession     25-34 years old
65435       65436  I am a developer by profession     18-24 years old
65436       65437     I code primarily as a hobby     18-24 years old

                Employment                                RemoteWork   Check  \
0      Employed, full-time                                    Remote  Apples
1      Employed, full-time                                    Remote  Apples
2      Employed, full-time                                    Remote  Apples
3       Student, full-time                                       NaN  Apples
4       Student, full-time                                       NaN  Apples
...                    ...                                       ...     ...
65432  Employed, full-time                                    Remote  Apples
65433  Employed, full-time                                    Remote  Apples
65434  Employed, full-time                                 In-person  Apples
65435  Employed, full-time  Hybrid (some remote, some in-person)  Apples
65436   Student, full-time                                       NaN  Apples

                                       CodingActivities  \
0                                                 Hobby
1      Hobby;Contribute to open-source projects;Other...
2      Hobby;Contribute to open-source projects;Other...
3                                                   NaN
4                                                   NaN
...                                                 ...
65432                  Hobby;School or academic work
65433        Hobby;Contribute to open-source projects
65434                                             Hobby
65435  Hobby;Contribute to open-source projects;Profe...
65436                                               NaN

                                           EdLevel  \
0                            Primary/elementary school
1        Bachelor’s degree (B.A., B.S., B.Eng., etc.)
2       Master’s degree (M.A., M.S., M.Eng., MBA, etc.)
3      Some college/university study without earning ...
4      Secondary school (e.g. American high school, G...
...                                                 ...
65432    Bachelor’s degree (B.A., B.S., B.Eng., etc.)
65433                                               NaN
65434    Bachelor’s degree (B.A., B.S., B.Eng., etc.)
65435  Secondary school (e.g. American high school, G...
65436                                               NaN

                                         LearnCode  \
0                              Books / Physical media
1      Books / Physical media;Colleague;On the job tr...
2      Books / Physical media;Colleague;On the job tr...
3      Other online resources (e.g., videos, blogs, f...
4      Other online resources (e.g., videos, blogs, f...
...                                                 ...
65432  On the job training;School (i.e., University, ...
65433                                               NaN
65434  Other online resources (e.g., videos, blogs, f...
65435  On the job training;Other online resources (e....
65436                                               NaN

                                      LearnCodeOnline  ... JobSatPoints_6  \
0                                                 NaN  ...            NaN
1      Technical documentation;Blogs;Books;Written Tu...  ...            0.0
2      Technical documentation;Blogs;Books;Written Tu...  ...            NaN
3      Stack Overflow;How-to videos;Interactive tutorial  ...            NaN
4      Technical documentation;Blogs;Written Tutorial...  ...            NaN
...                                                 ...  ...            ...
```

```
65432                                                 NaN  ...           NaN
65433                                                 NaN  ...           NaN
65434  Technical documentation;Stack Overflow;Social ...  ...           NaN
65435  Technical documentation;Blogs;Written Tutorial...  ...           0.0
65436                                                 NaN  ...           NaN

      JobSatPoints_7 JobSatPoints_8 JobSatPoints_9 JobSatPoints_10  \
0                NaN            NaN            NaN             NaN
1                0.0            0.0            0.0             0.0
2                NaN            NaN            NaN             NaN
3                NaN            NaN            NaN             NaN
4                NaN            NaN            NaN             NaN
...              ...            ...            ...             ...
65432            NaN            NaN            NaN             NaN
65433            NaN            NaN            NaN             NaN
65434            NaN            NaN            NaN             NaN
65435            0.0            0.0            0.0             0.0
65436            NaN            NaN            NaN             NaN

      JobSatPoints_11           SurveyLength SurveyEase ConvertedCompYearly  \
0                 NaN                     NaN        NaN                 NaN
1                 0.0                     NaN        NaN                 NaN
2                 NaN  Appropriate in length       Easy                 NaN
3                 NaN               Too long       Easy                 NaN
4                 NaN              Too short       Easy                 NaN
...               ...                     ...        ...                 ...
65432             NaN                     NaN        NaN                 NaN
65433             NaN                     NaN        NaN                 NaN
65434             NaN                     NaN        NaN                 NaN
65435             0.0                     NaN        NaN                 NaN
65436             NaN                     NaN        NaN                 NaN

      JobSat
0        NaN
1        NaN
2        NaN
3        NaN
4        NaN
...      ...
65432    NaN
65433    NaN
65434    NaN
65435    NaN
65436    NaN

[65437 rows x 114 columns]

Number of rows after removal: 65437
Duplicates remaining: 0
Verification successful: All exact duplicate rows have been removed.
```

## Section 2: Handling Missing Values

Task 2: Identify missing values in `CodingActivities`.

In [18]:
```
## Write your code here

# Check current missing values in 'CodingActivities'
initial_missing_count = df['CodingActivities'].isnull().sum()
print(f"\nInitial number of missing values in 'CodingActivities': {initial_missing_count}")
```

```
Initial number of missing values in 'CodingActivities': 10971
```

Task 3: Impute missing values in CodingActivities with forward-fill.

In [19]:
```
## Write your code here
print("--- Task 3: Impute missing values in CodingActivities with forward-fill ---")

# Display initial missing values in 'CodingActivities'
initial_missing_count = df['CodingActivities'].isnull().sum()
print(f"\nInitial missing values in 'CodingActivities': {initial_missing_count}")
print("Original 'CodingActivities' column:")
print(df['CodingActivities'])
```

```python
if initial_missing_count > 0:
    # Impute missing values using forward-fill (ffill)
    # .ffill() propagates the last valid observation forward to next valid observation.
    df['CodingActivities'].ffill(inplace=True)
    print("\nMissing values in 'CodingActivities' imputed using forward-fill.")

    # Verify imputation
    missing_after_imputation = df['CodingActivities'].isnull().sum()
    print(f"Missing values in 'CodingActivities' after imputation: {missing_after_imputation}")

    if missing_after_imputation == 0:
        print("Verification successful: 'CodingActivities' column imputed.")
    else:
        print("Warning: Some missing values might still exist (e.g., if first values were NaN).")
else:
    print("\nNo missing values found in 'CodingActivities'. No imputation needed.")

print("\n'CodingActivities' column after forward-fill imputation:")
print(df['CodingActivities'])
```

```
--- Task 3: Impute missing values in CodingActivities with forward-fill ---

Initial missing values in 'CodingActivities': 10971
Original 'CodingActivities' column:
0                                                   Hobby
1          Hobby;Contribute to open-source projects;Other...
2          Hobby;Contribute to open-source projects;Other...
3                                                     NaN
4                                                     NaN
                               ...
65432                        Hobby;School or academic work
65433              Hobby;Contribute to open-source projects
65434                                               Hobby
65435      Hobby;Contribute to open-source projects;Profe...
65436                                                 NaN
Name: CodingActivities, Length: 65437, dtype: object

Missing values in 'CodingActivities' imputed using forward-fill.
Missing values in 'CodingActivities' after imputation: 0
Verification successful: 'CodingActivities' column imputed.

'CodingActivities' column after forward-fill imputation:
0                                                   Hobby
1          Hobby;Contribute to open-source projects;Other...
2          Hobby;Contribute to open-source projects;Other...
3          Hobby;Contribute to open-source projects;Other...
4          Hobby;Contribute to open-source projects;Other...
                               ...
65432                        Hobby;School or academic work
65433              Hobby;Contribute to open-source projects
65434                                               Hobby
65435      Hobby;Contribute to open-source projects;Profe...
65436      Hobby;Contribute to open-source projects;Profe...
Name: CodingActivities, Length: 65437, dtype: object
```

```
/tmp/ipykernel_1588/3879054661.py:13: FutureWarning: A value is trying to be set on a copy of a Data
Frame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate
object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, in
place=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the ori
ginal object.


  df['CodingActivities'].ffill(inplace=True)
```

**Note**: Before normalizing ConvertedCompYearly, ensure that any missing values (NaN) in this column are handled appropriately. You can choose to either drop the rows containing NaN or replace the missing values with a suitable statistic (e.g., median or mean).

## Section 3: Normalizing Compensation Data

Task 4: Identify compensation-related columns, such as ConvertedCompYearly.

Normalization is commonly applied to compensation data to bring values within a comparable range. Here, you'll identify ConvertedCompYearly or similar columns, which contain compensation information. This column will be used in the subsequent tasks for normalization.

In [20]:
```
## Write your code here
import pandas as pd
import numpy as np

# Ensure 'ConvertedCompYearly' is numeric, handling any non-numeric entries by coercing to NaN
# and then filling NaNs (e.g., with median, as done in a previous task for this column).
if 'ConvertedCompYearly' in df.columns:
    df['ConvertedCompYearly'] = pd.to_numeric(df['ConvertedCompYearly'], errors='coerce')
    if df['ConvertedCompYearly'].isnull().any():
        median_comp = df['ConvertedCompYearly'].median()
        df['ConvertedCompYearly'].fillna(median_comp, inplace=True)
        print(f"Handled missing values in 'ConvertedCompYearly' for demonstration by filling with m

print("--- Section 3, Task 1: Identify columns requiring normalization ---")

# 1. Check if 'ConvertedCompYearly' exists and is numeric
if 'ConvertedCompYearly' in df.columns and pd.api.types.is_numeric_dtype(df['ConvertedCompYearly']):
    print("\n'ConvertedCompYearly' column identified for potential normalization.")
    print("\nDescriptive statistics of 'ConvertedCompYearly':")
    print(df['ConvertedCompYearly'].describe())

    print("\nReason for normalization:")
    print("The 'ConvertedCompYearly' column shows a wide range of values (from min to max) and a si
    print("Normalization is often needed for such columns, especially in machine learning models, t
    print("This helps in equalizing the contribution of all features to the model's performance.")
else:
    print("\n'ConvertedCompYearly' column not found or is not numeric in the DataFrame.")
    print("Please ensure your DataFrame has this column and its data type is numerical.")
```

```
Handled missing values in 'ConvertedCompYearly' for demonstration by filling with median: 65000.00
--- Section 3, Task 1: Identify columns requiring normalization ---

'ConvertedCompYearly' column identified for potential normalization.

Descriptive statistics of 'ConvertedCompYearly':
count    6.543700e+04
mean     7.257636e+04
std      1.122207e+05
min      1.000000e+00
25%      6.500000e+04
50%      6.500000e+04
75%      6.500000e+04
max      1.625660e+07
Name: ConvertedCompYearly, dtype: float64

Reason for normalization:
The 'ConvertedCompYearly' column shows a wide range of values (from min to max) and a significant st
andard deviation.
Normalization is often needed for such columns, especially in machine learning models, to ensure tha
t features with larger numerical ranges do not disproportionately influence the model compared to fe
atures with smaller ranges.
This helps in equalizing the contribution of all features to the model's performance.
/tmp/ipykernel_1588/274937723.py:11: FutureWarning: A value is trying to be set on a copy of a DataF
rame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate
object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, in
place=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the ori
ginal object.


  df['ConvertedCompYearly'].fillna(median_comp, inplace=True)
```

Task 5: Normalize ConvertedCompYearly using Min-Max Scaling.

Min-Max Scaling brings all values in a column to a 0-1 range, making it useful for comparing data across different scales. Here, you will apply Min-Max normalization to the ConvertedCompYearly column, creating a new column ConvertedCompYearly_MinMax with normalized values.

In [23]:
```
## Write your code here
print("--- Task 5: Normalize ConvertedCompYearly using Min-Max Scaling ---")
!pip install scikit-learn
from sklearn.preprocessing import MinMaxScaler
if 'ConvertedCompYearly' in df.columns and pd.api.types.is_numeric_dtype(df['ConvertedCompYearly']):
    # Initialize the MinMaxScaler
    scaler = MinMaxScaler()

    # Reshape the 'ConvertedCompYearly' column to a 2D array as required by scaler
    # .values gets the numpy array, .reshape(-1, 1) converts it to a column vector
    df['ConvertedCompYearly_Normalized'] = scaler.fit_transform(df[['ConvertedCompYearly']])

    print("\n'ConvertedCompYearly' column normalized using Min-Max Scaling.")
    print("\nDescriptive statistics of the normalized 'ConvertedCompYearly_Normalized' column:")
    print(df['ConvertedCompYearly_Normalized'].describe())

    print("\nOriginal and Normalized values (first 5 rows):")
    print(df[['ConvertedCompYearly', 'ConvertedCompYearly_Normalized']].head())

    print("\nVerification: The normalized values should now be between 0 and 1.")
else:
    print("\n'ConvertedCompYearly' column not found or is not numeric. Cannot perform normalization
```

```
--- Task 5: Normalize ConvertedCompYearly using Min-Max Scaling ---
Collecting scikit-learn
  Downloading scikit_learn-1.7.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(17 kB)
Requirement already satisfied: numpy>=1.22.0 in /opt/conda/lib/python3.12/site-packages (from scikit
-learn) (2.3.0)
Collecting scipy>=1.8.0 (from scikit-learn)
  Downloading scipy-1.15.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (61 k
B)
Collecting joblib>=1.2.0 (from scikit-learn)
  Downloading joblib-1.5.1-py3-none-any.whl.metadata (5.6 kB)
Collecting threadpoolctl>=3.1.0 (from scikit-learn)
  Downloading threadpoolctl-3.6.0-py3-none-any.whl.metadata (13 kB)
Downloading scikit_learn-1.7.0-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (12.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.5/12.5 MB 109.1 MB/s eta 0:00:00
Downloading joblib-1.5.1-py3-none-any.whl (307 kB)
Downloading scipy-1.15.3-cp312-cp312-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (37.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 37.3/37.3 MB 177.3 MB/s eta 0:00:0000:01
Downloading threadpoolctl-3.6.0-py3-none-any.whl (18 kB)
Installing collected packages: threadpoolctl, scipy, joblib, scikit-learn
Successfully installed joblib-1.5.1 scikit-learn-1.7.0 scipy-1.15.3 threadpoolctl-3.6.0

'ConvertedCompYearly' column normalized using Min-Max Scaling.

Descriptive statistics of the normalized 'ConvertedCompYearly_Normalized' column:
count    65437.000000
mean         0.004464
std          0.006903
min          0.000000
25%          0.003998
50%          0.003998
75%          0.003998
max          1.000000
Name: ConvertedCompYearly_Normalized, dtype: float64

Original and Normalized values (first 5 rows):
   ConvertedCompYearly  ConvertedCompYearly_Normalized
0              65000.0                        0.003998
1              65000.0                        0.003998
2              65000.0                        0.003998
3              65000.0                        0.003998
4              65000.0                        0.003998

Verification: The normalized values should now be between 0 and 1.
```

Task 6: Apply Z-score Normalization to `ConvertedCompYearly`.

Z-score normalization standardizes values by converting them to a distribution with a mean of 0 and a standard deviation of 1. This method is helpful for datasets with a Gaussian (normal) distribution. Here, you'll calculate Z-scores for the ConvertedCompYearly column, saving the results in a new column ConvertedCompYearly_Zscore.

In [24]:
```
## Write your code here
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler

print("--- Task 6: Apply Z-score Normalization to ConvertedCompYearly ---")

if 'ConvertedCompYearly' in df.columns and pd.api.types.is_numeric_dtype(df['ConvertedCompYearly']):
    # Initialize the StandardScaler
    scaler = StandardScaler()

    # Reshape the 'ConvertedCompYearly' column to a 2D array (required by StandardScaler)
    # and apply Z-score normalization.
    df['ConvertedCompYearly_Zscore'] = scaler.fit_transform(df[['ConvertedCompYearly']])

    print("\n'ConvertedCompYearly' column normalized using Z-score Scaling.")
    print("\nDescriptive statistics of the Z-score normalized 'ConvertedCompYearly_Zscore' column:")
    print(df['ConvertedCompYearly_Zscore'].describe())

    print("\nOriginal, Min-Max Normalized (if applicable), and Z-score Normalized values (first 5 r
    # Display both if Min-Max was also applied, otherwise just original and Z-score
    if 'ConvertedCompYearly_Normalized' in df.columns:
        print(df[['ConvertedCompYearly', 'ConvertedCompYearly_Normalized', 'ConvertedCompYearly_Zsc
    else:
        print(df[['ConvertedCompYearly', 'ConvertedCompYearly_Zscore']].head())

    print("\nVerification: Z-score normalized values should have a mean close to 0 and a standard d
else:
    print("\n'ConvertedCompYearly' column not found or is not numeric. Cannot perform Z-score norma
```

```
--- Task 6: Apply Z-score Normalization to ConvertedCompYearly ---

'ConvertedCompYearly' column normalized using Z-score Scaling.

Descriptive statistics of the Z-score normalized 'ConvertedCompYearly_Zscore' column:
count    6.543700e+04
mean    -4.517105e-17
std      1.000008e+00
min     -6.467249e-01
25%     -6.751355e-02
50%     -6.751355e-02
75%     -6.751355e-02
max      1.442172e+02
Name: ConvertedCompYearly_Zscore, dtype: float64

Original, Min-Max Normalized (if applicable), and Z-score Normalized values (first 5 rows):
   ConvertedCompYearly  ConvertedCompYearly_Normalized  \
0              65000.0                        0.003998
1              65000.0                        0.003998
2              65000.0                        0.003998
3              65000.0                        0.003998
4              65000.0                        0.003998

   ConvertedCompYearly_Zscore
0                   -0.067514
1                   -0.067514
2                   -0.067514
3                   -0.067514
4                   -0.067514

Verification: Z-score normalized values should have a mean close to 0 and a standard deviation close
to 1.
```

## Section 4: Visualization of Normalized Data

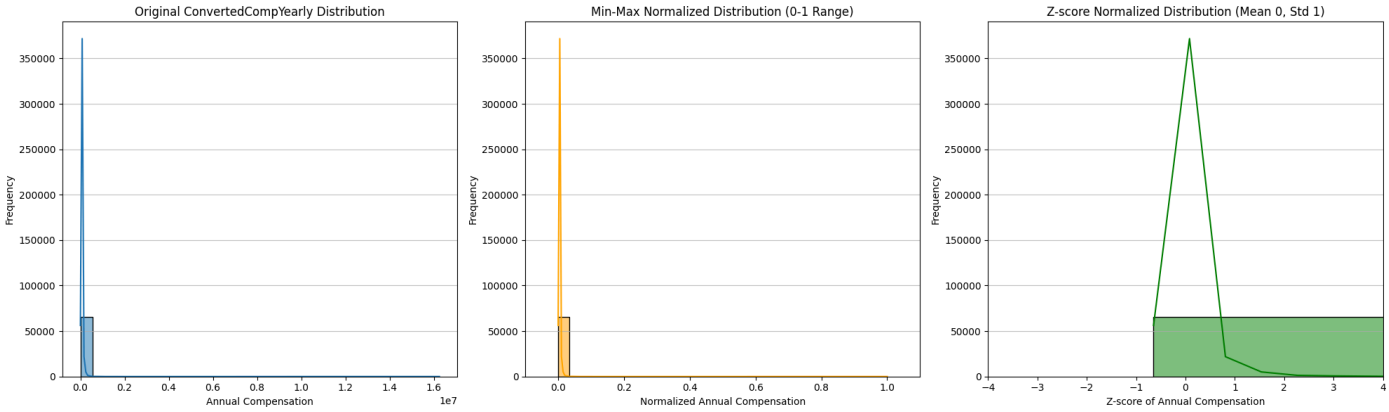
Task 7: Visualize the distribution of `ConvertedCompYearly`, `ConvertedCompYearly_Normalized`, and `ConvertedCompYearly_Zscore`

Visualization helps you understand how normalization changes the data distribution. In this task, create histograms for the original ConvertedCompYearly, as well as its normalized versions (ConvertedCompYearly_MinMax and ConvertedCompYearly_Zscore). This will help you compare how each normalization technique affects the data range and distribution.

In [26]:
```
## Write your code here

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler, StandardScaler

# Ensure 'ConvertedCompYearly' is numeric and handle NaNs for robustness
df['ConvertedCompYearly'] = pd.to_numeric(df['ConvertedCompYearly'], errors='coerce')
if df['ConvertedCompYearly'].isnull().any():
    median_comp = df['ConvertedCompYearly'].median()
    df['ConvertedCompYearly'].fillna(median_comp, inplace=True)
    print(f"Handled missing values in 'ConvertedCompYearly' for visualization demo by filling with

# Min-Max Normalize 'ConvertedCompYearly'
# This creates the 'ConvertedCompYearly_Normalized' column (equivalent to your 'ConvertedCompYearly
scaler_minmax = MinMaxScaler()
df['ConvertedCompYearly_Normalized'] = scaler_minmax.fit_transform(df[['ConvertedCompYearly']])
print("Min-Max Normalized 'ConvertedCompYearly' data created as 'ConvertedCompYearly_Normalized'.")


# Z-score Normalize 'ConvertedCompYearly'
scaler_zscore = StandardScaler()
df['ConvertedCompYearly_Zscore'] = scaler_zscore.fit_transform(df[['ConvertedCompYearly']])
print("Z-score Normalized 'ConvertedCompYearly' data created as 'ConvertedCompYearly_Zscore'.")


print("\n--- Task 7: Visualize the distribution of ConvertedCompYearly, ConvertedCompYearly_Normali

# Create a figure with three subplots for comparison
# Each subplot will contain a histogram with a Kernel Density Estimate (KDE)
fig, axes = plt.subplots(1, 3, figsize=(20, 6)) # 1 row, 3 columns for side-by-side comparison

# Plot 1: Original ConvertedCompYearly
sns.histplot(df['ConvertedCompYearly'], kde=True, bins=30, ax=axes[0])
axes[0].set_title('Original ConvertedCompYearly Distribution')
axes[0].set_xlabel('Annual Compensation')
axes[0].set_ylabel('Frequency')
axes[0].grid(axis='y', alpha=0.75) # Add grid for readability

# Plot 2: Min-Max Normalized ConvertedCompYearly
sns.histplot(df['ConvertedCompYearly_Normalized'], kde=True, bins=30, ax=axes[1], color='orange')
axes[1].set_title('Min-Max Normalized Distribution (0-1 Range)')
axes[1].set_xlabel('Normalized Annual Compensation')
axes[1].set_ylabel('Frequency')
axes[1].set_xlim([-0.1, 1.1]) # Set consistent x-axis for 0-1 range
axes[1].grid(axis='y', alpha=0.75)

# Plot 3: Z-score Normalized ConvertedCompYearly
sns.histplot(df['ConvertedCompYearly_Zscore'], kde=True, bins=30, ax=axes[2], color='green')
axes[2].set_title('Z-score Normalized Distribution (Mean 0, Std 1)')
axes[2].set_xlabel('Z-score of Annual Compensation')
axes[2].set_ylabel('Frequency')
axes[2].set_xlim([-4, 4]) # Set consistent x-axis for typical Z-score range
axes[2].grid(axis='y', alpha=0.75)

plt.tight_layout() # Adjusts subplot params for a tight layout
plt.show()

print("\n--- Visualization Interpretation ---")
print("The plots visually demonstrate the effect of each normalization technique:")
```

```
print("1.  **Original Distribution:** Shows the raw spread of compensation values.")
print("2.  **Min-Max Normalized Distribution:** The data is scaled to a fixed range, typically betw
print("3.  **Z-score Normalized Distribution:** The data is transformed to have a mean of approxima
print("This helps in ensuring that features with larger numerical values do not dominate machine le
```

```
Min-Max Normalized 'ConvertedCompYearly' data created as 'ConvertedCompYearly_Normalized'.
Z-score Normalized 'ConvertedCompYearly' data created as 'ConvertedCompYearly_Zscore'.

--- Task 7: Visualize the distribution of ConvertedCompYearly, ConvertedCompYearly_Normalized, and C
onvertedCompYearly_Zscore ---
```




```
--- Visualization Interpretation ---
The plots visually demonstrate the effect of each normalization technique:
1.  **Original Distribution:** Shows the raw spread of compensation values.
2.  **Min-Max Normalized Distribution:** The data is scaled to a fixed range, typically between 0 an
d 1. The shape of the distribution remains the same, but the values are compressed or expanded to fi
t this new range.
3.  **Z-score Normalized Distribution:** The data is transformed to have a mean of approximately 0 a
nd a standard deviation of approximately 1. The shape is preserved, but the distribution is centered
and scaled based on its statistical properties, rather than a fixed min/max.
This helps in ensuring that features with larger numerical values do not dominate machine learning m
odels and that they contribute equally during training.
```

## Summary

In this lab, you practiced essential normalization techniques, including:

- Identifying and handling duplicate rows.
- Checking for and imputing missing values.
- Applying Min-Max scaling and Z-score normalization to compensation data.
- Visualizing the impact of normalization on data distribution.