Interview Questions :-

1. What are missing values and how do you handle them?

- **Definition**: Entries where a value is absent (NaN/None in Pandas).
- **Why they matter**: Can bias analyses or break downstream code.
- **Handling strategies**:
  - **Identify** with df.isnull().sum().
  - **Drop** rows or columns if the missing rate is very high—for example, we dropped any rows still missing date_added or duration.
  - **Impute** with a constant or a statistic: in our script we filled director, cast, country, and rating with "Unknown", and imputed missing duration_int with its median.
  - **Forward-/back-fill** for time series fields—e.g., we used .ffill().bfill() on date_added.

---

2. How do you treat duplicate records?

- **Why remove?** Duplicates can exaggerate counts or distort averages.
- **Detection**: df.duplicated() or df.drop_duplicates() in Pandas.
- **Treatment**:
  - Use df.drop_duplicates() to remove exact duplicates (we printed how many rows were dropped).
  - Optionally, identify near-duplicates (e.g., same title & year) and decide whether to merge or remove.

---

3. Difference between dropna() and fillna() in Pandas?

- **dropna()**

- **Purpose**: Remove any rows (or columns) containing missing values.
- **Use case**: When missingness is rare or cannot be imputed reliably—for example, dropping rows still missing critical fields like date_added.

- **fillna()**
  - **Purpose**: Replace missing values with a specified value or method.
  - **Use case**: When you want to preserve row count and can reasonably impute—e.g., filling director with "Unknown" or numeric columns with median.

---

4. What is outlier treatment and why is it important?

- **Outliers** are data points far outside the typical range (e.g., a "duration_int" of 1,000 minutes).
- **Importance**: They can skew summary statistics and model training.
- **Treatment methods**:
  - **Detection** via boxplots, z-scores, or IQR rule.
  - **Handling**:
    - **Cap or floor** them to a percentile (e.g., 1st–99th).
    - **Remove** extreme values if they are clearly erroneous.
    - **Transform** variables (e.g., log transform) to reduce skew.

---

5. Explain the process of standardizing data.

- **Goal**: Ensure consistency in text or numeric formats so analyses aren't fragmented.
- **Text standardization** (we did):
  - Trim whitespace: .str.strip()

    - Consistent casing: .str.title() or .str.lower()
    - Uniform delimiters in multi-value fields (e.g., genres).
- **Numeric standardization** (if needed):
    - Scaling to zero-mean/unit-variance (StandardScaler) or min-max scaling.
    - Useful before clustering or more advanced modeling.

---

6. How do you handle inconsistent data formats (e.g., date/time)?

- **Parsing**: Use a robust parser—e.g., pd.to_datetime(..., errors='coerce', dayfirst=True) to convert strings into datetime64 objects.
- **Imputation**: After parsing, forward/back-fill or drop remaining nulls.
- **Reformatting**: Store dates in ISO format (YYYY-MM-DD) or extract components (.dt.year, .dt.month) for analysis.

---

7. What are common data cleaning challenges?

1. **High missingness** in critical fields.
2. **Inconsistent encoding** or delimiters (e.g., mixed comma/semicolon lists).
3. **Non-standard text** (typos, varying case).
4. **Date/time quirks** (multiple formats, time zones).
5. **Hidden duplicates** (near-duplicates requiring fuzzy matching).
6. **Unbalanced classes** or skewed numeric distributions.

---

8. How can you check data quality?

- **Quantitative checks**:
    - Missing-value counts (df.isnull().sum())
    - Duplicate counts (df.duplicated().sum())

  - DataType consistency (df.dtypes)
  - Summary statistics (df.describe())
- **Visual checks**:
  - Missing-value heatmaps or bar charts (we plotted missing_values.png).
  - Histograms and boxplots to spot outliers.
- **Business-rule validations**:
  - Ensure release_year ≤ current year.
  - Check that duration_int > 0.
  - Validate ratings against a known set (e.g., ['G','PG','PG-13','R','TV-MA', …]).