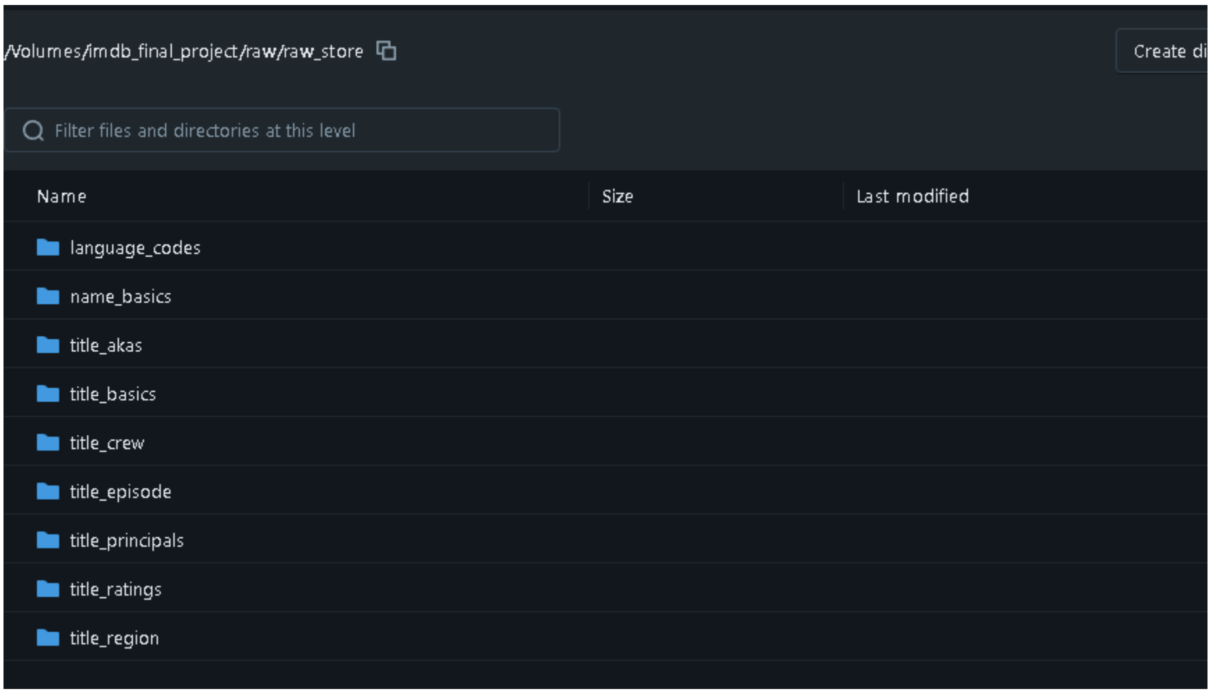
Data Cleaning

Data cleaning has been done in databricks where the files have been uploaded in volume called raw_store


/Volumes/imdb_final_project/raw/raw_store

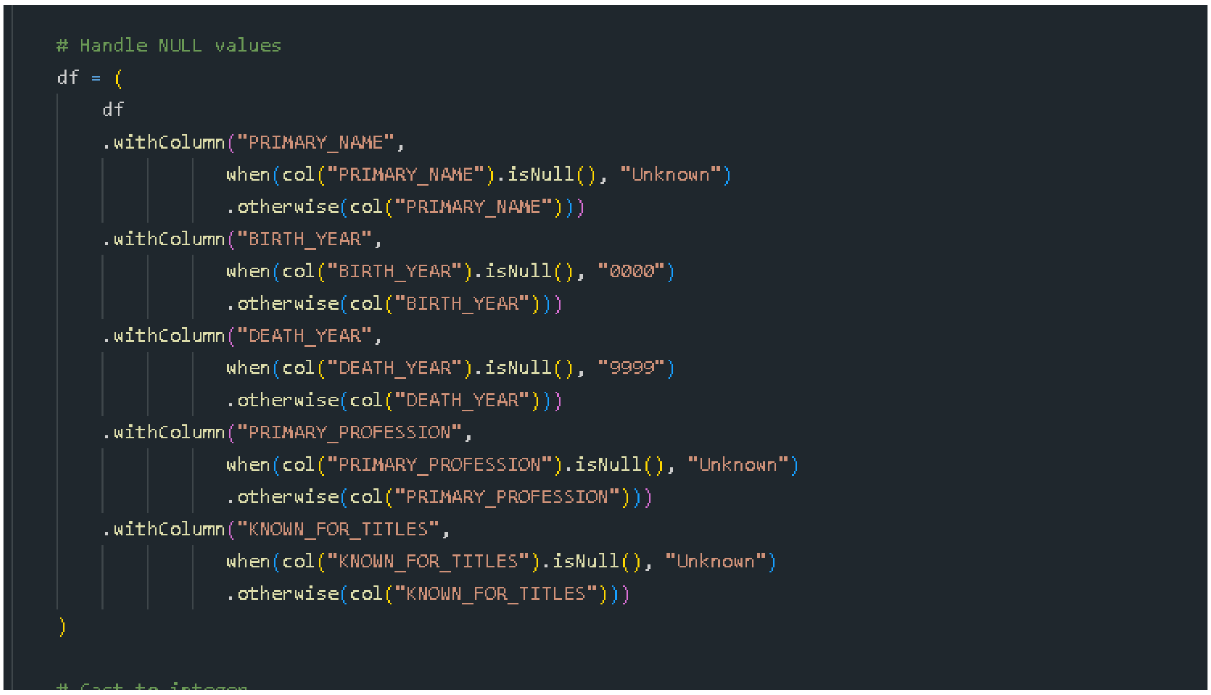
Name	Size	Last modified
language_codes		
name_basics		
title_akas		
title_basics		
title_crew		
title_episode		
title_principals		
title_ratings		
title_region		

1. Name_Basics

1. Expectations

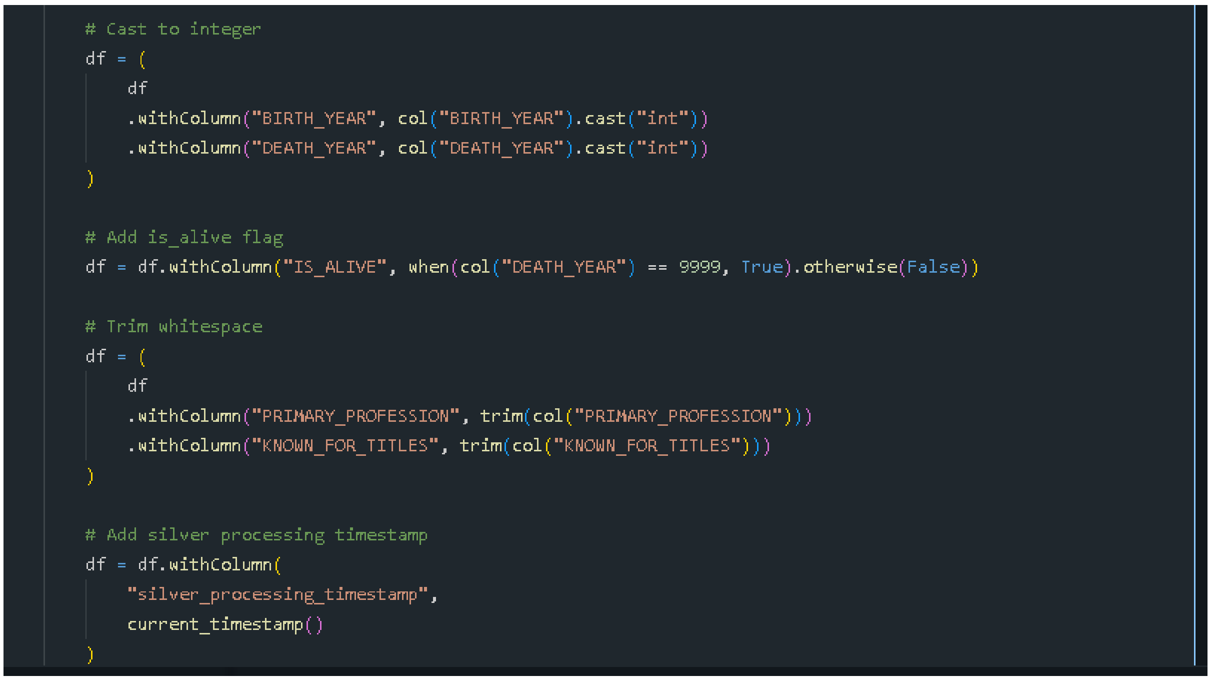
Put the expectations as drop if nconst is not Null

2. Replacing Nulls


```
# Handle NULL values
df = (
    df
    .withColumn("PRIMARY_NAME",
                when(col("PRIMARY_NAME").isNull(), "Unknown")
                .otherwise(col("PRIMARY_NAME")))
    .withColumn("BIRTH_YEAR",
                when(col("BIRTH_YEAR").isNull(), "0000")
                .otherwise(col("BIRTH_YEAR")))
    .withColumn("DEATH_YEAR",
                when(col("DEATH_YEAR").isNull(), "9999")
                .otherwise(col("DEATH_YEAR")))
    .withColumn("PRIMARY_PROFESSION",
                when(col("PRIMARY_PROFESSION").isNull(), "Unknown")
                .otherwise(col("PRIMARY_PROFESSION")))
    .withColumn("KNOWN_FOR_TITLES",
                when(col("KNOWN_FOR_TITLES").isNull(), "Unknown")
                .otherwise(col("KNOWN_FOR_TITLES")))
)

# Cast to integer
```

Replaced the nulls with values according to their respective datatypes

3. Datatype changes:

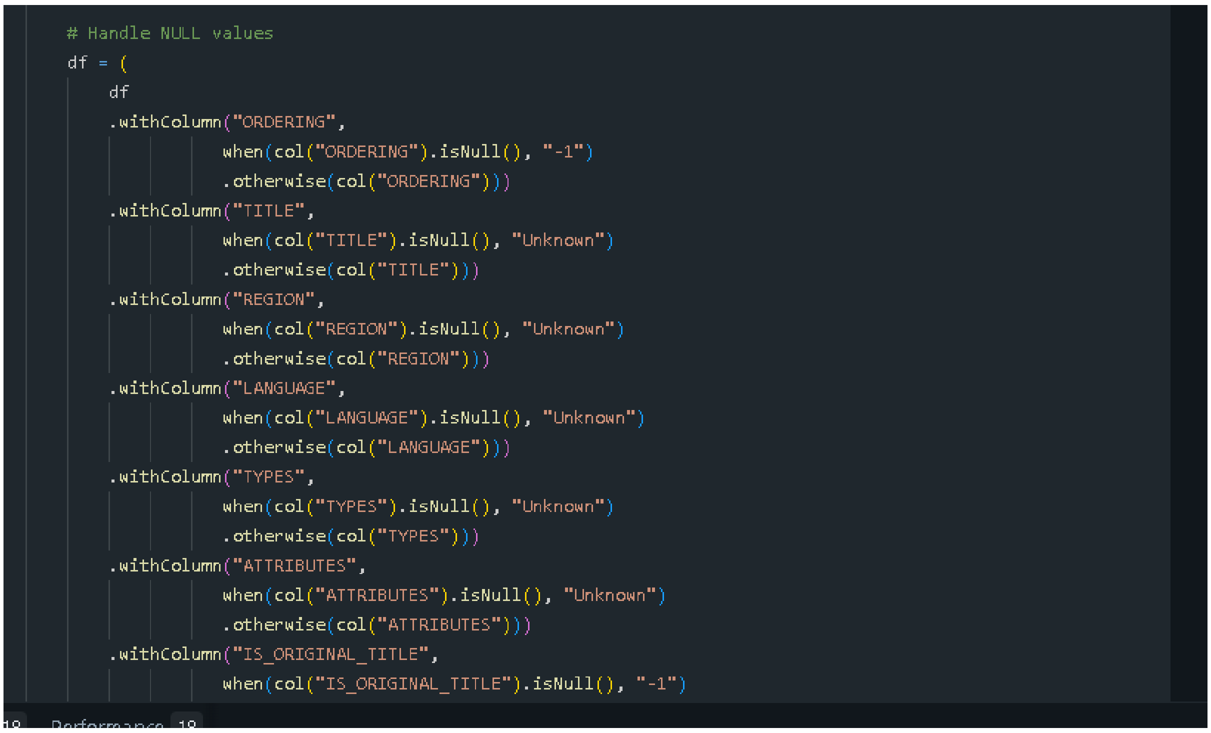


```
# Cast to integer
df = (
    df
    .withColumn("BIRTH_YEAR", col("BIRTH_YEAR").cast("int"))
    .withColumn("DEATH_YEAR", col("DEATH_YEAR").cast("int"))
)

# Add is_alive flag
df = df.withColumn("IS_ALIVE", when(col("DEATH_YEAR") == 9999, True).otherwise(False))

# Trim whitespace
df = (
    df
    .withColumn("PRIMARY_PROFESSION", trim(col("PRIMARY_PROFESSION")))
    .withColumn("KNOWN_FOR_TITLES", trim(col("KNOWN_FOR_TITLES")))
)

# Add silver processing timestamp
df = df.withColumn(
    "silver_processing_timestamp",
    current_timestamp()
)
```

2. Title_akas

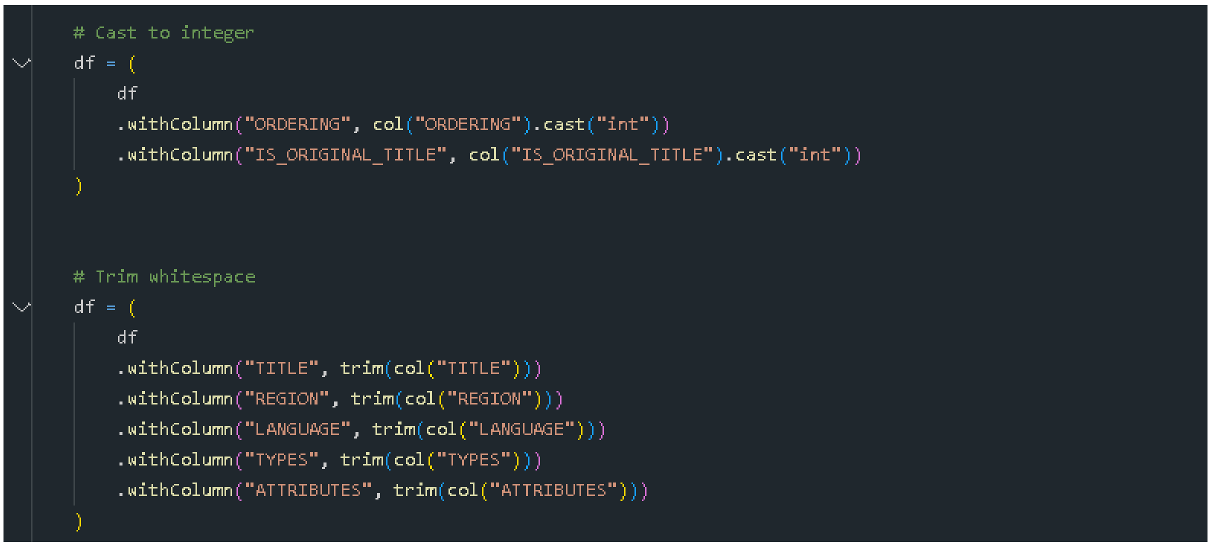
1. Replacing Nulls

Replaced all the nulls with respective datatype values



```python
# Handle NULL values
df = (
    df
    .withColumn("ORDERING",
                when(col("ORDERING").isNull(), "-1")
                .otherwise(col("ORDERING")))
    .withColumn("TITLE",
                when(col("TITLE").isNull(), "Unknown")
                .otherwise(col("TITLE")))
    .withColumn("REGION",
                when(col("REGION").isNull(), "Unknown")
                .otherwise(col("REGION")))
    .withColumn("LANGUAGE",
                when(col("LANGUAGE").isNull(), "Unknown")
                .otherwise(col("LANGUAGE")))
    .withColumn("TYPES",
                when(col("TYPES").isNull(), "Unknown")
                .otherwise(col("TYPES")))
    .withColumn("ATTRIBUTES",
                when(col("ATTRIBUTES").isNull(), "Unknown")
                .otherwise(col("ATTRIBUTES")))
    .withColumn("IS_ORIGINAL_TITLE",
                when(col("IS_ORIGINAL_TITLE").isNull(), "-1")
```

2. Changed the datatypes and fixed the whitespaces using trim function



```python
# Cast to integer
df = (
    df
    .withColumn("ORDERING", col("ORDERING").cast("int"))
    .withColumn("IS_ORIGINAL_TITLE", col("IS_ORIGINAL_TITLE").cast("int"))
)


# Trim whitespace
df = (
    df
    .withColumn("TITLE", trim(col("TITLE")))
    .withColumn("REGION", trim(col("REGION")))
    .withColumn("LANGUAGE", trim(col("LANGUAGE")))
    .withColumn("TYPES", trim(col("TYPES")))
    .withColumn("ATTRIBUTES", trim(col("ATTRIBUTES")))
)
```

3. Title_Language_codes

No cleaning as the table was created and it was ran from bronze to silver layer

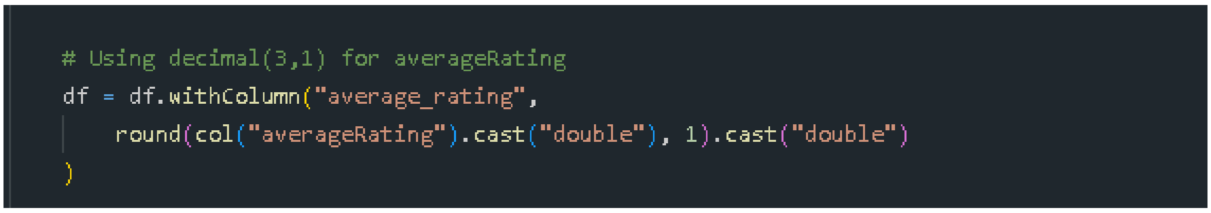
4. Title_ratings

1. Expectations

Put the expectation as tconst is not null ie drop if nulls are in tconst as it the identifer

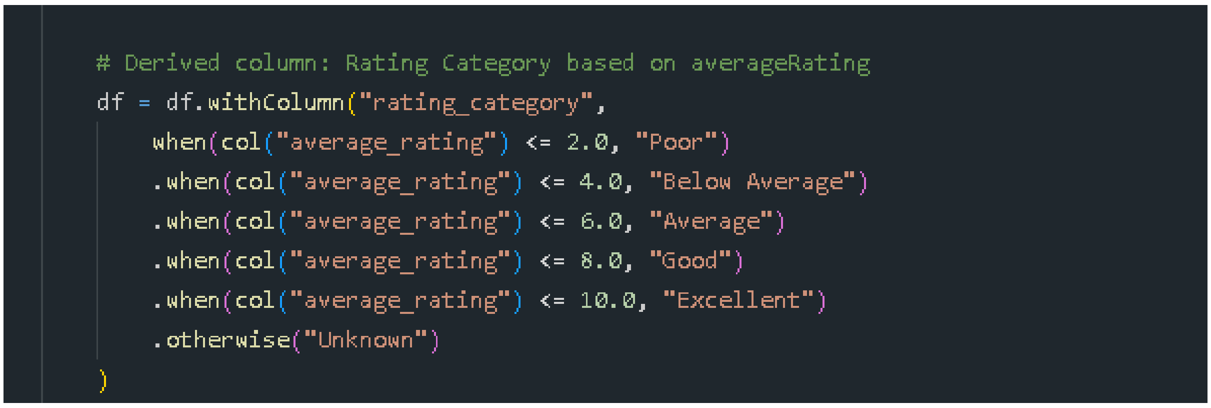
2. Datatype conversions :

averageRating has been changed to double from string and numvotes has been made integer as well



```
# Using decimal(3,1) for averageRating
df = df.withColumn("average_rating",
    round(col("averageRating").cast("double"), 1).cast("double")
)
```

3. Created a new Derived column ‘rating category’ based on the ratings



```
# Derived column: Rating Category based on averageRating
df = df.withColumn("rating_category",
    when(col("average_rating") <= 2.0, "Poor")
    .when(col("average_rating") <= 4.0, "Below Average")
    .when(col("average_rating") <= 6.0, "Average")
    .when(col("average_rating") <= 8.0, "Good")
    .when(col("average_rating") <= 10.0, "Excellent")
    .otherwise("Unknown")
)
```

5 .Title_basics

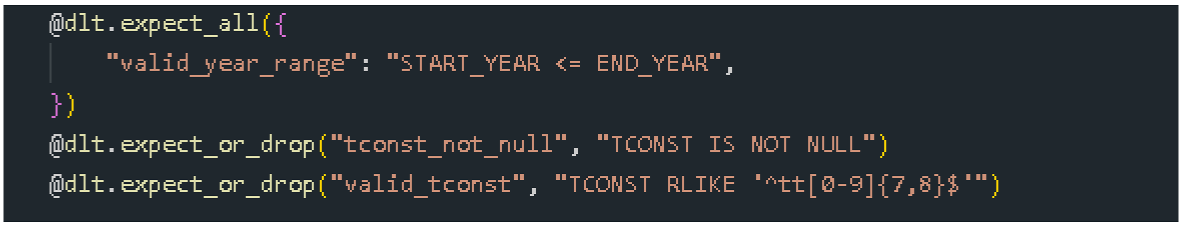
1.Expectations:

There are three expectations:

Tconst_not_null

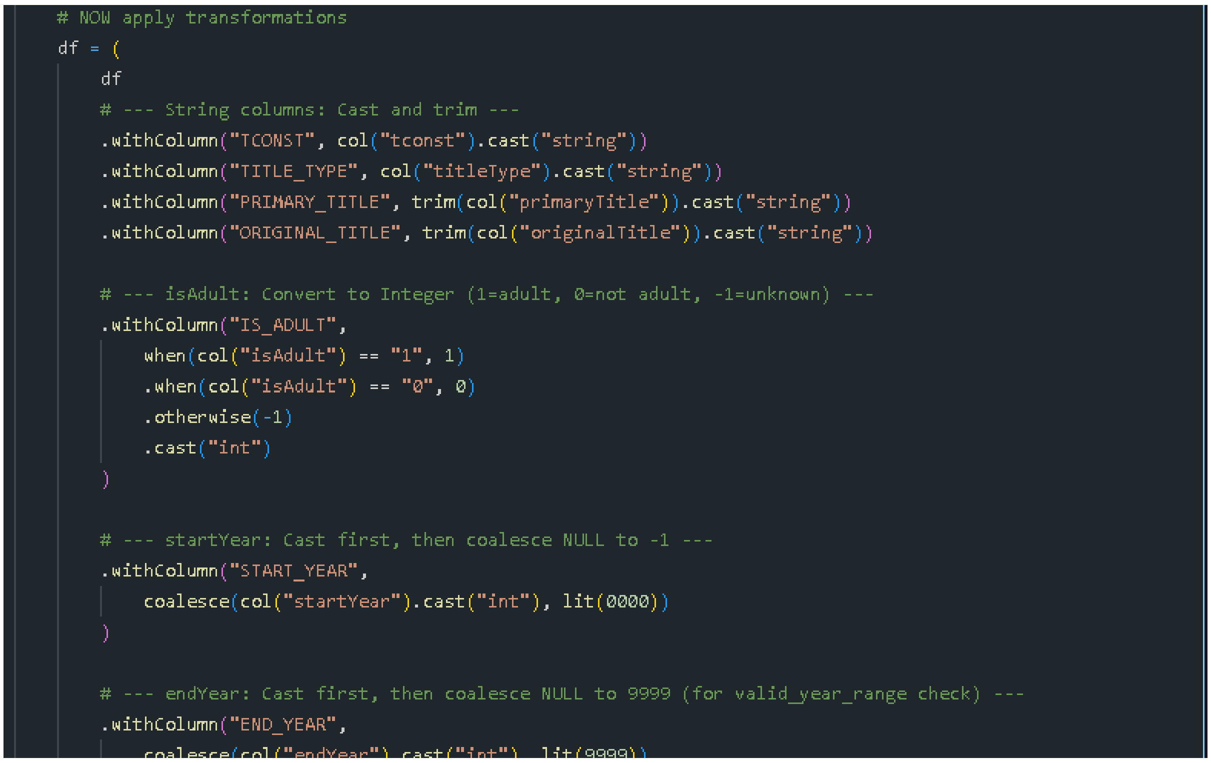
Valid_tconst

Valid_year range to make sure there are no fields which have start year greater than end year

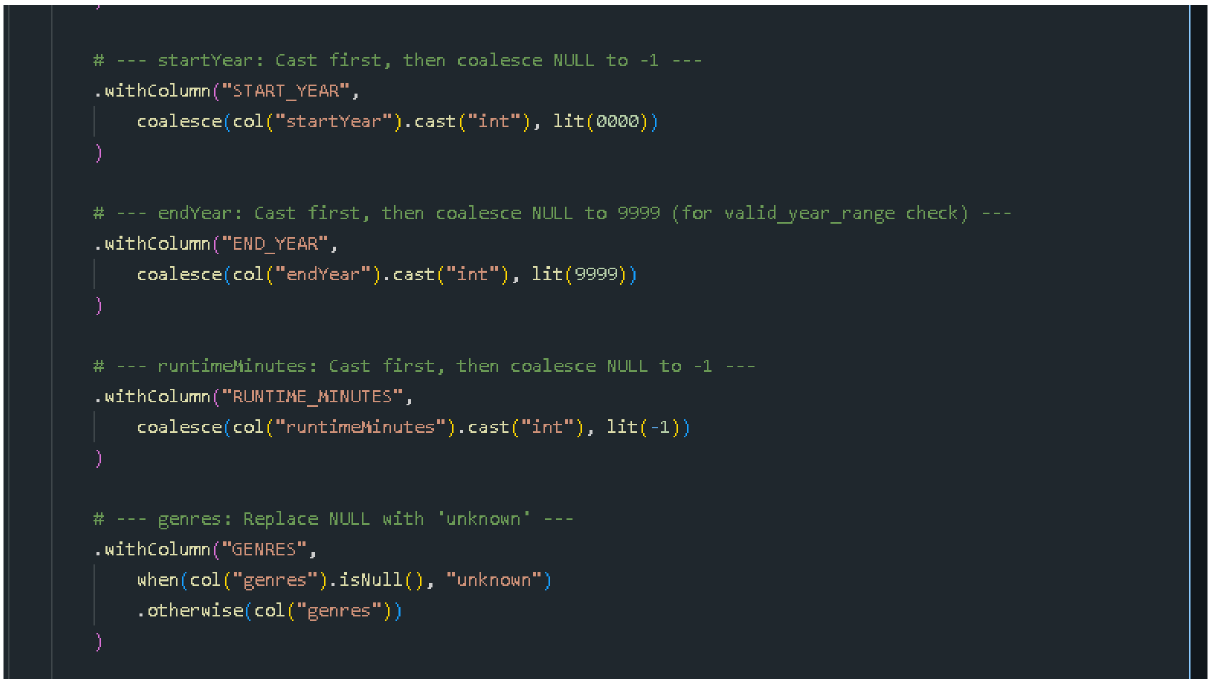


```
@dlt.expect_all({
    "valid_year_range": "START_YEAR <= END_YEAR",
})
@dlt.expect_or_drop("tconst_not_null", "TCONST IS NOT NULL")
@dlt.expect_or_drop("valid_tconst", "TCONST RLIKE '^tt[0-9]{7,8}$'")
```

2. Datatypes have been casted and all respective nulls / odd values have been changed with respective unknown category



```
# NOW apply transformations
df = (
    df
    # --- String columns: Cast and trim ---
    .withColumn("TCONST", col("tconst").cast("string"))
    .withColumn("TITLE_TYPE", col("titleType").cast("string"))
    .withColumn("PRIMARY_TITLE", trim(col("primaryTitle")).cast("string"))
    .withColumn("ORIGINAL_TITLE", trim(col("originalTitle")).cast("string"))

    # --- isAdult: Convert to Integer (1=adult, 0=not adult, -1=unknown) ---
    .withColumn("IS_ADULT",
        when(col("isAdult") == "1", 1)
        .when(col("isAdult") == "0", 0)
        .otherwise(-1)
        .cast("int")
    )

    # --- startYear: Cast first, then coalesce NULL to -1 ---
    .withColumn("START_YEAR",
        coalesce(col("startYear").cast("int"), lit(0000))
    )

    # --- endYear: Cast first, then coalesce NULL to 9999 (for valid_year_range check) ---
    .withColumn("END_YEAR",
        coalesce(col("endYear").cast("int"), lit(9999))
```

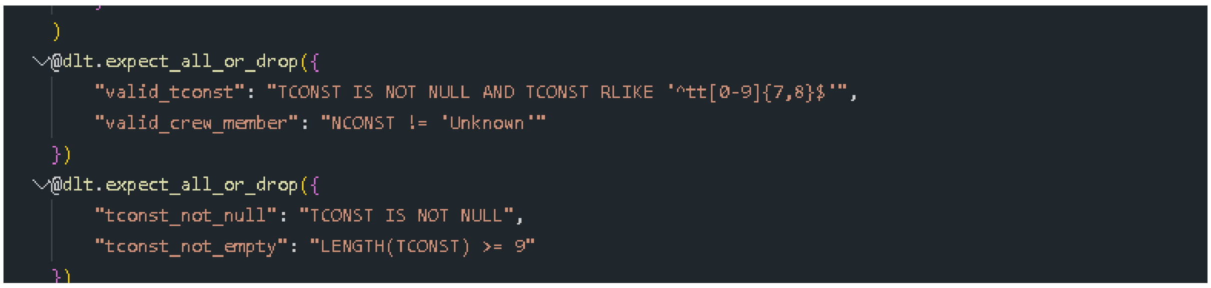


```
    )

    # --- startYear: Cast first, then coalesce NULL to -1 ---
    .withColumn("START_YEAR",
        coalesce(col("startYear").cast("int"), lit(0000))
    )

    # --- endYear: Cast first, then coalesce NULL to 9999 (for valid_year_range check) ---
    .withColumn("END_YEAR",
        coalesce(col("endYear").cast("int"), lit(9999))
    )

    # --- runtimeMinutes: Cast first, then coalesce NULL to -1 ---
    .withColumn("RUNTIME_MINUTES",
        coalesce(col("runtimeMinutes").cast("int"), lit(-1))
    )

    # --- genres: Replace NULL with 'unknown' ---
    .withColumn("GENRES",
        when(col("genres").isNull(), "unknown")
        .otherwise(col("genres"))
    )
```

6. Title_crew

1. Expectations:

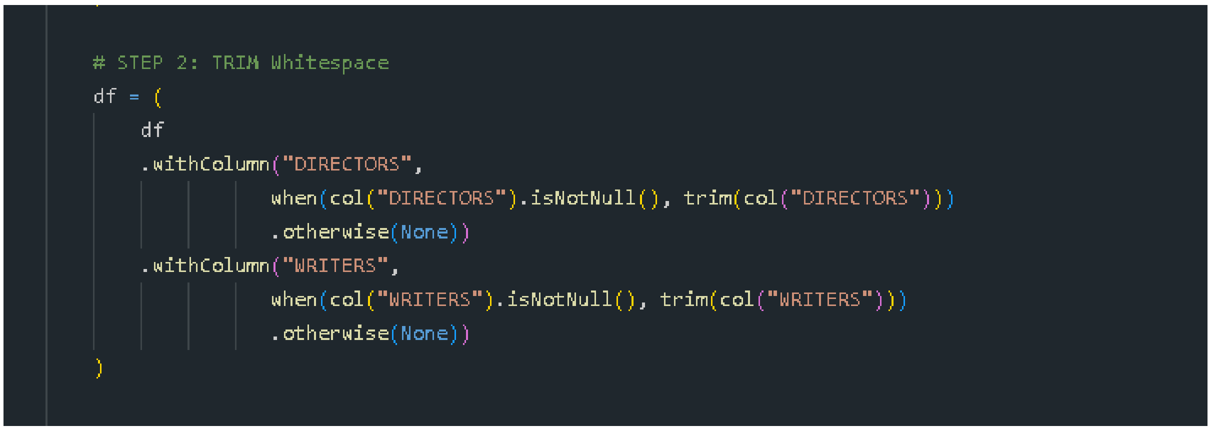
Tconst should not be null and should be valid



```
    )
@dlt.expect_all_or_drop({
    "valid_tconst": "TCONST IS NOT NULL AND TCONST RLIKE '^tt[0-9]{7,8}$'",
    "valid_crew_member": "NCONST != 'Unknown'"
})
@dlt.expect_all_or_drop({
    "tconst_not_null": "TCONST IS NOT NULL",
    "tconst_not_empty": "LENGTH(TCONST) >= 9"
})
```

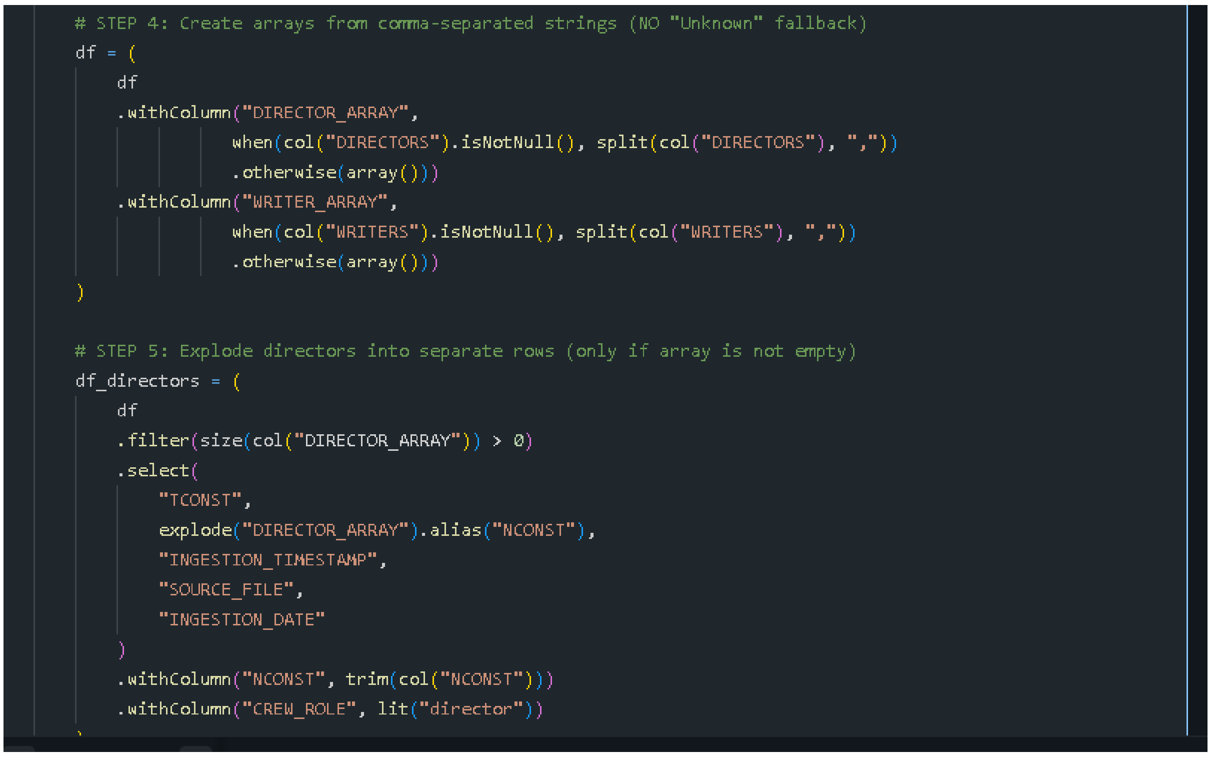
2. Whitespaces issues

Whitespaces have been fixed using trim in directors and writers



```
    # STEP 2: TRIM Whitespace
    df = (
        df
        .withColumn("DIRECTORS",
                    when(col("DIRECTORS").isNotNull(), trim(col("DIRECTORS")))
                    .otherwise(None))
        .withColumn("WRITERS",
                    when(col("WRITERS").isNotNull(), trim(col("WRITERS")))
                    .otherwise(None))
    )
```

Exploding the comma seperated values into directors and writers as separate rows and create a field called crew_role to store the directors and writers to in seperated rows and nconst as seperate



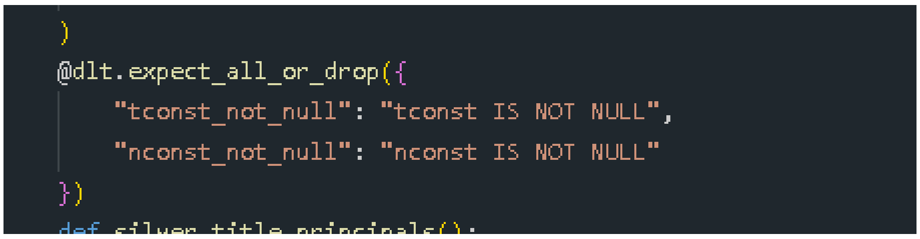
```
# STEP 4: Create arrays from comma-separated strings (NO "Unknown" fallback)
df = (
    df
    .withColumn("DIRECTOR_ARRAY",
                when(col("DIRECTORS").isNotNull(), split(col("DIRECTORS"), ","))
                .otherwise(array()))
    .withColumn("WRITER_ARRAY",
                when(col("WRITERS").isNotNull(), split(col("WRITERS"), ","))
                .otherwise(array()))
)

# STEP 5: Explode directors into separate rows (only if array is not empty)
df_directors = (
    df
    .filter(size(col("DIRECTOR_ARRAY")) > 0)
    .select(
        "TCONST",
        explode("DIRECTOR_ARRAY").alias("NCONST"),
        "INGESTION_TIMESTAMP",
        "SOURCE_FILE",
        "INGESTION_DATE"
    )
    .withColumn("NCONST", trim(col("NCONST")))
    .withColumn("CREW_ROLE", lit("director"))
```

7. Title_Principals

Expectations:

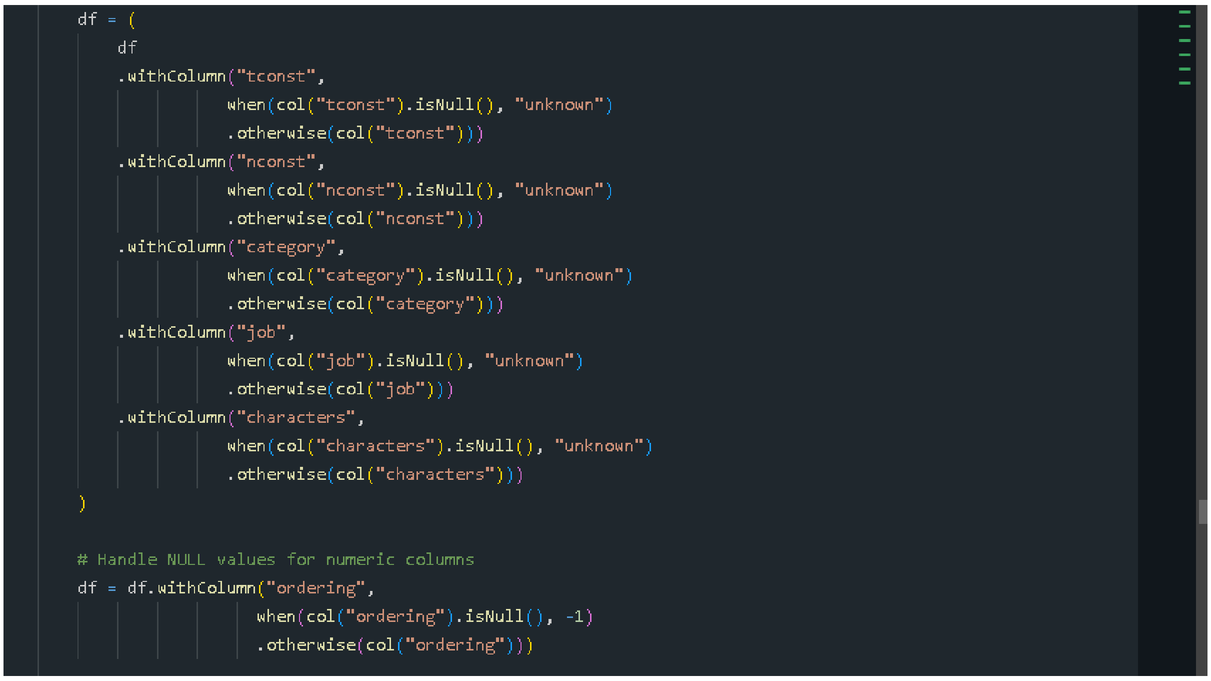
Tconst not null or else drop it

Ncosnt not null or else drop it



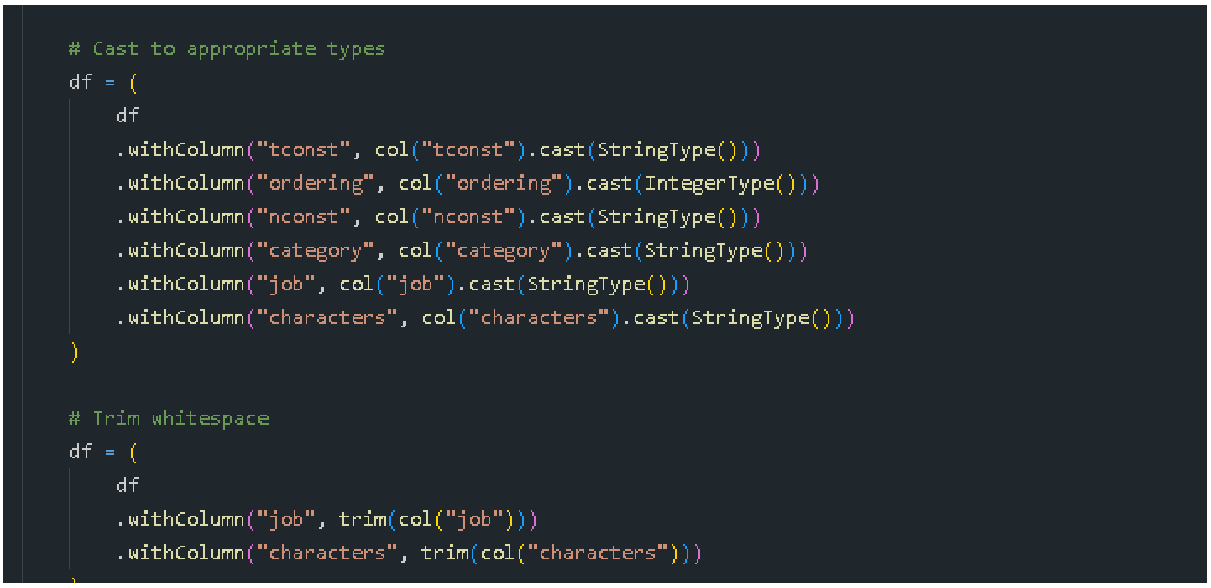
```
)
@dlt.expect_all_or_drop({
    "tconst_not_null": "tconst IS NOT NULL",
    "nconst_not_null": "nconst IS NOT NULL"
})
def silver_title_principals():
```

Replaced the nulls with respective datatype according values


```
df = (
    df
    .withColumn("tconst",
                when(col("tconst").isNull(), "unknown")
                .otherwise(col("tconst")))
    .withColumn("nconst",
                when(col("nconst").isNull(), "unknown")
                .otherwise(col("nconst")))
    .withColumn("category",
                when(col("category").isNull(), "unknown")
                .otherwise(col("category")))
    .withColumn("job",
                when(col("job").isNull(), "unknown")
                .otherwise(col("job")))
    .withColumn("characters",
                when(col("characters").isNull(), "unknown")
                .otherwise(col("characters")))
)

# Handle NULL values for numeric columns
df = df.withColumn("ordering",
                   when(col("ordering").isNull(), -1)
                   .otherwise(col("ordering")))
```

4. Datatypes cast and whitespace

Did the casting and trimmed the whitespaces wherever required


```
# Cast to appropriate types
df = (
    df
    .withColumn("tconst", col("tconst").cast(StringType()))
    .withColumn("ordering", col("ordering").cast(IntegerType()))
    .withColumn("nconst", col("nconst").cast(StringType()))
    .withColumn("category", col("category").cast(StringType()))
    .withColumn("job", col("job").cast(StringType()))
    .withColumn("characters", col("characters").cast(StringType()))
)

# Trim whitespace
df = (
    df
    .withColumn("job", trim(col("job")))
    .withColumn("characters", trim(col("characters")))
```

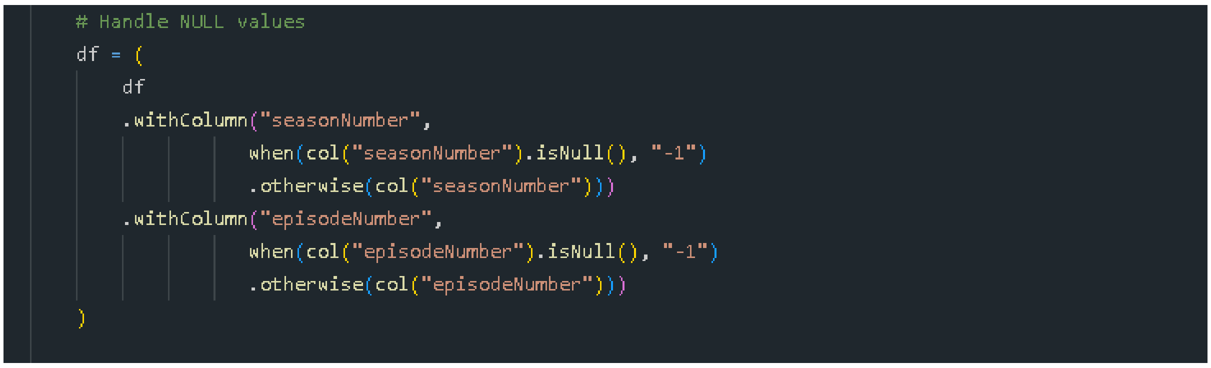
8. Title_Episode

1.Expectations:

Tconst and parentTconst should not be else drop it

2. Handling Nulls

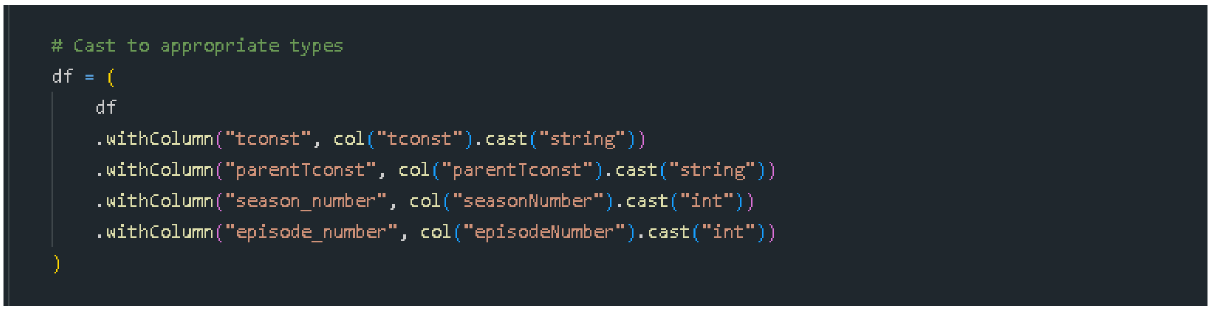
Replaced the nulls in the field with respective datatype specifc values



```
# Handle NULL values
df = (
    df
    .withColumn("seasonNumber",
                when(col("seasonNumber").isNull(), "-1")
                .otherwise(col("seasonNumber")))
    .withColumn("episodeNumber",
                when(col("episodeNumber").isNull(), "-1")
                .otherwise(col("episodeNumber")))
)
```

3. Datatype conversion:

Casted the datatype as defined in profiling



```
# Cast to appropriate types
df = (
    df
    .withColumn("tconst", col("tconst").cast("string"))
    .withColumn("parentTconst", col("parentTconst").cast("string"))
    .withColumn("season_number", col("seasonNumber").cast("int"))
    .withColumn("episode_number", col("episodeNumber").cast("int"))
)
```

9. Title_Region

No cleaning required as the tables were created and it was run from bronze to silver layer