Designing Advanced Data Architecture for Business Intelligence
Mini Project
IMDb Midterm Team Project

Team 6:
Abisha Vadukoot, NUID: 002371769
Bhagyashri Pagar, NUID: 002310690
Rahul Bothra, NUID: 002378790

1. Introduction

Overview of the Project

The project focuses on implementing a Business Intelligence (BI) solution using the IMDb Non-Commercial Datasets. The goal is to analyze movie, TV, and crew data to generate actionable insights that can support business decision-making. This end-to-end BI implementation includes data profiling, transformation, loading, and reporting. It aims to provide business analysts with the tools to generate reports on movie trends, crew performance, and regional movie releases.

Business Requirements

The primary business objectives are:

- Analyze professions for personnel and identify those with multiple professions.
- Provide insights on movies by genre, year, and region.
- Track movie lengths and generate metrics based on release year.
- Identify adult and non-adult movies, and analyze crew members (directors, writers, etc.).
- Generate reports on top-rated movies, including those by genre and year.
- Analyze TV series data, including season/episode comparisons with ratings.

Technologies and Tools Used

- **ER Studio / Navicat**: For data modeling, schema design, and database management.
- **Azure Data Factory (ADF)**: For automating the ETL process, managing data flows from source to target systems.
- **Alteryx**: For data profiling, blending, and cleaning.

- **Snowflake**: Cloud-based data warehouse for storing processed data.
- **Power BI / Tableau**: For creating data visualizations and reports to meet business needs.

2. Data Analysis and Profiling

Data Sources and Initial Analysis

The source data for this project is derived from IMDb Non-Commercial Datasets, which are provided in tab-separated values (TSV) format, gzipped and encoded in UTF-8. These datasets include information about movies, TV shows, crew members, ratings, and more. The analysis of these datasets began with reviewing the content of each file to understand the structure and properties of the data. The goal was to identify any potential issues, such as missing or inconsistent data, that could affect the quality of the analysis.

Data Profiling with Alteryx/Python

Data profiling was carried out using **Alteryx** and **Python**. The profiling process included the following steps:

1. **Identifying and Checking Key Metrics**: We examined key metrics, such as unique values, nulls, duplicates, and potential outliers in each column.
2. **Fixing Anomalies**: Irregularities such as empty strings, foreign characters (including emojis), and invalid data entries were corrected. For example, empty strings were replaced with 'Unknown,' and special characters were removed using regular expressions.
3. **Formula Implementation**: The following formulas were used to clean and standardize the data:
    - `REGEX_Replace([fieldName], '"([^"]*)"', '$1')`: Removed any surrounding double quotes from fields.
    - `Trim([fieldName])`: Eliminated leading and trailing spaces.
    - `IF [filedName] == "\N" THEN "Unknown" ELSE [fieldName] ENDIF`: Replaced instances of "\N" with "Unknown."
    - `REGEX_Replace([fieldName], '[^\x20-\x7E\xA0-\xD7FF\xE000-\xFFFF\u2000-\u206F\u2B50\uFE0F]', ' ')`: Cleared unwanted characters, such as emojis and foreign symbols.
    - `REGEX_Replace(Trim([fieldName]), "\\s+", " ")`: Reduced multiple spaces or tabs to a single space between words.

Data Quality and Issues Identified

Several data quality issues were discovered during the profiling process:

- **Empty String Issue**: Fields containing empty strings were encountered, which caused issues with data processing. This was resolved by replacing empty strings with 'Unknown.'
- **Truncation Warning**: Warnings such as *“tool #122 value was truncated to 44 characters”* were identified. This issue arose due to fields exceeding the 44-character limit. It was determined that the truncation only affected the display of data and did not modify the actual dataset.
- **CSV Parsing Issues**: Errors related to missing closing quotes and delimiters were resolved by modifying input settings, specifically changing the option from "Ignore Delimiters in Quotes" to "Ignore Delimiters in None" in Alteryx's Input Data Configuration.
- **Missing Fields**: Warnings about missing fields (e.g., "Not enough fields in record") were identified and handled by correcting delimiters and ensuring proper data loading.

Error Resolution:

- **Truncation Warning**: The truncation issue was determined to be a display warning rather than a data corruption problem. The data remained intact in the original dataset.
- **CSV Parsing**: The parsing issue was resolved by adjusting the delimiter settings. Changing the option to "Ignore Delimiters in None" ensured that the correct delimiter was used, and the issue was resolved without altering the data.

Handling Multiple Languages

Since the data came from multiple regions and languages, ensuring proper loading and display of multilingual content was critical. To address this:

- **UTF-8 Encoding**: We ensured that all datasets were encoded in UTF-8 to handle special characters, accents, and non-English characters without errors.
- **Character Encoding Verification**: We verified that non-English characters were correctly represented in the data. This included characters from various languages, such as Chinese, Japanese, Spanish, etc. Any inconsistencies or encoding issues were fixed during the data profiling phase.
- **Foreign Characters and Emojis**: Special characters, such as emojis and other non-standard symbols, were removed using regular expressions to prevent any data corruption or display issues.

Post-Error Fixing Process

After resolving the errors, a thorough review of all datasets was conducted to check for:

- **Unique Values and Duplicates**: Ensured all fields had unique entries and removed any duplicate records.
- **Nulls and Missing Data**: Null or missing values were replaced with appropriate placeholders like 'Unknown' or '-1' (for integers).
- **Encoding and Foreign Characters**: Ensured that all data was encoded in UTF-8 and removed foreign characters or emojis using regular expressions.
- **White Space Issues**: Trimming of leading/trailing spaces and normalization of multiple spaces between words.

For example:

- **Birth Year**: If the value was "\N", it was replaced with "0000".
- **Death Year**: If "\N" was encountered, it was replaced with "9999".
- **End Year**: If "\N" was found for TV shows, it was replaced with "-1" to indicate missing data.

These steps ensured that the data was clean, standardized, properly encoded for multilingual support, and ready for further analysis.

3. Mapping Document

Source-to-Target Mapping

The mapping document outlines how data from the **IMDb Non-Commercial Datasets** (source) is transformed and loaded into the integration schema (target). Below is the mapping for key tables from the IMDb dataset to the target schema:

1. **Title Akas (`title.akas.tsv.gz`)**

    - **Source Columns**:
        - `titleId` → `tconst` (Unique identifier for the title)
        - `ordering` → `ordering` (Unique identifier for rows within a title)
        - `title` → `localized_title` (The title in the localized language)
        - `region` → `region` (Region for the title)
        - `language` → `language` (Language of the title)
        - `types` → `title_types` (List of attributes like "DVD", "festival", "TV", etc.)

        - `attributes` → `attributes` (Additional terms describing the title)
        - `isOriginalTitle` → `is_original_title` (Indicates if the title is original)
    - **Target Schema Columns**:
        - `localized_title`, `region`, `language`, `title_types`, `attributes`, and `is_original_title` are loaded directly into the corresponding fields of the integration schema.
    - **Data Transformation**: None, data is loaded as-is with minor transformations for standardization.
2. **Title Basics (`title.basics.tsv.gz`)**

    - **Source Columns**:
        - `tconst` → `title_id` (Unique identifier)
        - `titleType` → `title_type` (Type of title e.g., movie, TV series)
        - `primaryTitle` → `primary_title` (Most commonly used title)
        - `originalTitle` → `original_title` (Original title in its original language)
        - `isAdult` → `is_adult` (Indicates whether it's an adult title)
        - `startYear` → `start_year` (Year the title was released)
        - `endYear` → `end_year` (End year for TV series, '\N' if not applicable)
        - `runtimeMinutes` → `runtime_minutes` (Duration of the title in minutes)
        - `genres` → `genres` (Comma-separated list of genres)
    - **Target Schema Columns**:
        - `title_id`, `title_type`, `primary_title`, `original_title`, `is_adult`, `start_year`, `end_year`, `runtime_minutes`, and `genres` are loaded into the target schema.
    - **Data Transformation**:
        - **Null Handling**: If `endYear` or `startYear` contains "\N", they are converted to -1.
        - **Data Cleansing**: Any `isAdult` value of "\N" is replaced with `-1` to signify unknown.
        - **Standardization**: Text fields are trimmed to remove leading/trailing spaces, and unwanted characters like emojis or foreign symbols are cleaned using regular expressions.

3. **Title Crew (`title.crew.tsv.gz`)**
    - **Source Columns**:
        - `tconst` → `title_id` (Unique identifier for the title)
        - `directors` → `directors` (List of directors by `nconst`)
        - `writers` → `writers` (List of writers by `nconst`)
    - **Target Schema Columns**:
        - `title_id`, `directors`, and `writers` are loaded directly into the target schema.
    - **Data Transformation**:
        - Convert lists of `nconst` (director and writer IDs) into proper arrays of IDs for easier querying.
4. **Title Episode (`title.episode.tsv.gz`)**
    - **Source Columns**:
        - `tconst` → `episode_id` (Identifier for the episode)
        - `parentTconst` → `parent_title_id` (ID of the parent TV series)
        - `seasonNumber` → `season_number` (Season number for the episode)
        - `episodeNumber` → `episode_number` (Episode number within the season)
    - **Target Schema Columns**:
        - `episode_id`, `parent_title_id`, `season_number`, and `episode_number` are loaded into the target schema.
    - **Data Transformation**:
        - Data is loaded as-is with necessary data type transformations (e.g., converting season and episode numbers to integers).
5. **Title Principals (`title.principals.tsv.gz`)**
    - **Source Columns**:
        - `tconst` → `title_id` (Unique identifier for the title)
        - `ordering` → `ordering` (Row identifier)
        - `nconst` → `person_id` (Unique identifier for the crew member or actor)
        - `category` → `category` (Job category e.g., actor, director)
        - `job` → `job_title` (Specific job title for the person, if applicable)

        - `characters` → `characters` (Character played by the actor, if applicable)
    - **Target Schema Columns**:
        - `title_id`, `ordering`, `person_id`, `category`, `job_title`, and `characters` are loaded into the target schema.
    - **Data Transformation**:
        - For missing values in `characters`, replace with "Unknown."
        - If `job` is "\N", it is replaced with "Unknown."
6. **Title Ratings (`title.ratings.tsv.gz`)**
    - **Source Columns**:
        - `tconst` → `title_id` (Unique identifier for the title)
        - `averageRating` → `average_rating` (Weighted average rating of the title)
        - `numVotes` → `num_votes` (Number of votes the title has received)
    - **Target Schema Columns**:
        - `title_id`, `average_rating`, and `num_votes` are loaded into the target schema.
    - **Data Transformation**: Data is loaded as-is. For any missing values or issues, defaults or null placeholders are used.
7. **Name Basics (`name.basics.tsv.gz`)**
    - **Source Columns**:
        - `nconst` → `person_id` (Unique identifier for the person)
        - `primaryName` → `name` (Name of the person)
        - `birthYear` → `birth_year` (Year of birth)
        - `deathYear` → `death_year` (Year of death, if applicable)
        - `primaryProfession` → `primary_profession` (Top professions of the person)
        - `knownForTitles` → `known_for_titles` (List of titles the person is known for)
    - **Target Schema Columns**:
        - `person_id`, `name`, `birth_year`, `death_year`, `primary_profession`, and `known_for_titles` are loaded into the target schema.
    - **Data Transformation**:
        - For `birthYear` and `deathYear`, replace "\N" with "0000" for birth year and "9999" for death year to signify unknown.

- Null values in `primaryProfession` or `knownForTitles` are handled with "Unknown" values.

Data Transformation Logic

The primary goal of data transformation is to ensure consistency and standardization across datasets. The following key transformations were applied:

1. **Null Value Handling**: For any missing or null values (e.g., `\N`), appropriate defaults like "Unknown" for text fields or `-1` for numerical fields were used.
2. **Text Cleaning**: Unwanted characters, such as emojis or special symbols, were removed using regular expressions to maintain data integrity.
3. **Standardizing Date Fields**: For dates such as `birthYear`, `deathYear`, `startYear`, and `endYear`, "\N" was replaced with `0000` or `9999` to represent missing data in a standardized way.
4. **Consolidation of Data Types**: Fields such as `isAdult`, `startYear`, `endYear`, and `runtimeMinutes` were converted to appropriate data types (e.g., integers) to facilitate analysis and reporting.

4. Data Staging, Cleaning, and Integration

Staging the Data

The data staging process began with importing the raw datasets into the system. These datasets were initially stored in the **Bronze Layer** of the data lake, which contained unmodified, raw data from IMDb. Intermediate steps were then taken to cleanse and transform the data. The cleansing and transformation were performed using **Alteryx**, where the data underwent various cleaning operations, ensuring that any inconsistencies, missing values, and anomalies were addressed.

After processing the raw data in Alteryx, the cleansed data was saved as **Parquet files**—an efficient storage format that preserves both schema and data integrity. These files were then stored in the **Silver Layer** of the Azure Data Lake Gen2 container. The Silver Layer served as an intermediary step before moving the data into the integration schema.

Data Cleaning

The data cleaning process focused on resolving common issues such as missing values, incorrect formats, and data inconsistencies. Several techniques were employed to improve data quality:

1. **Handling Missing and Null Values**:
    - Missing or null values (represented as `\N`) were replaced with default placeholders, such as `-1` for unknown integers or `0000` for unknown years.
2. **Standardizing Data Types**:
    - Fields such as `startYear`, `endYear`, and `runtimeMinutes` were converted into appropriate data types, ensuring consistency across all datasets.
3. **Text Cleaning**:
    - Whitespace was trimmed from fields, and unwanted characters (e.g., emojis or special symbols) were removed using regular expressions (regex). This ensured that the data was clean and could be processed without errors.
4. **Dealing with Duplicates and Invalid Records**:
    - Duplicate records and invalid data entries were removed or corrected to maintain data integrity. This step was vital for ensuring accurate results during the loading and analysis processes.

Loading into Integration Schema

After cleaning, the data was ready for integration into the **Gold Layer** for reporting and analysis. The **Medallion Architecture** was followed, where the data progressed from the Bronze (raw) to the Silver (cleansed) and finally to the Gold (final, usage-ready) layers.

The cleaned and transformed data was staged into **Snowflake** using **Azure Data Factory (ADF)** pipelines. The data was transferred from the Silver Layer in the Azure Data Lake Gen2 container, where the Parquet files were stored, into Snowflake's schema. Snowflake tables were designed based on the schema, ensuring that the data was structured appropriately for reporting and analysis.

The data load was executed using the **Copy Data Activity** in ADF pipelines, which facilitated the smooth transfer of data from the staging area to the integration schema in Snowflake.

This structured approach to data staging, cleaning, and integration ensured that high-quality, well-organized data was available for analysis, providing a solid foundation for the BI project's reporting and insights.

5. Data Modeling and Design

Data Modeling Process

The data modeling process involved designing a schema that could effectively support the business requirements and enable efficient analysis. Given the nature of the data, a **Hybrid Star Schema** was employed to provide a balance between simplicity and flexibility. This schema design integrates both **star schema** and **snowflake schema** principles to optimize query performance and facilitate easy navigation of the data.

In the **Hybrid Star Schema**, the central fact tables are connected to dimension tables, with some dimension tables normalized to avoid redundancy and enhance data integrity. The fact tables represent transactional data, such as movie details, person-related data, and episode information. The dimension tables, such as genre, language, and persons, contain descriptive attributes related to the facts.

Integration Schema Design

The integration schema was designed to align closely with the business requirements, ensuring that all the necessary data could be extracted and analyzed efficiently. The schema was structured with **fact tables** that store measurable business metrics and **dimension tables** that provide context and categorization for those metrics.

The key components of the integration schema include:

- **Fact Tables**: These tables store quantitative data and are the central part of the schema. They include:
    - `fact_movies`: Contains data related to movies, including attributes like movie length, genre, release year, and rating.
    - `fact_episodes`: Stores data related to episodes of TV shows, including season numbers, episode numbers, and ratings.
    - `fact_person`: Contains data about individuals involved in the creation of movies, such as actors, directors, and writers, and their roles in those movies.
- **Dimension Tables**: These tables provide descriptive context for the facts and allow for detailed analysis. They include:
    - `dim_genre`: Describes the different genres associated with titles (e.g., Action, Drama, Comedy).
    - `dim_language`: Lists the languages in which a title is available.
    - `dim_persons`: Contains information about persons (actors, directors, writers) involved in the creation of movies.

    - `dim_profession`: Describes the professions of individuals involved in the movies.
    - `dim_region`: Represents the regions where movies are released.
    - `dim_title`: Contains information about the movies, such as the title, original title, and related attributes.
- **Bridge Tables**: These are used to handle many-to-many relationships between fact and dimension tables, particularly when a single fact (like a movie) is associated with multiple dimensions (like professions or regions). The bridge tables include:

    - `Bridge_person_profession`: Links persons to their respective professions (e.g., actor, director).
    - `Bridge_Title_Akas`: Associates alternative titles with movies and their regions.
    - `Bridge_title_genres`: Connects movies to multiple genres.
    - `Bridge_title_language`: Links movies to their associated languages.
    - `Bridge_title_regions`: Connects movies to the regions where they were released.

Relationships and Constraints

The integration schema incorporates several relationships and constraints to ensure data integrity and facilitate accurate analysis:

- **One-to-Many Relationships**: Most dimension tables (e.g., `dim_genre`, `dim_language`, `dim_persons`) have a one-to-many relationship with the fact tables. For example, one genre can be associated with multiple movies, but each movie can have only one genre at a time.

- **Many-to-Many Relationships**: The **Bridge Tables** handle many-to-many relationships, such as a movie being linked to multiple genres or an actor having multiple professions. These relationships ensure that the schema can represent complex scenarios without data duplication.

- **Referential Integrity**: Foreign keys were established between fact and dimension tables to ensure that the data remains consistent across the schema. For example, each movie (in the `fact_movies` table) references the appropriate genre from the `dim_genre` table via a foreign key.

- **Normalization and Denormalization**: While some dimension tables are **denormalized** for easy querying (e.g., `dim_title` or `dim_persons`), others are **normalized** to reduce redundancy and maintain data integrity, like the `Bridge_*` tables. This hybrid approach balances performance with data consistency.

Dimensional Modeling

The dimensional model was created to facilitate business analysis and support reporting needs. The dimensions in the schema were specifically designed to allow for detailed analysis, including:

- **Fact Tables**:
    - `fact_movies`: Includes metrics such as `averageRating`, `numVotes`, and `runtimeMinutes`.
    - `fact_episodes`: Contains episode-specific data like `episodeNumber`, `seasonNumber`, and `rating`.
    - `fact_person`: Captures information about the persons involved in the titles, including their roles and number of contributions.
- **Dimension Tables**:
    - `dim_genre`: Each genre, such as Action, Drama, and Comedy, is stored as a separate entry.
    - `dim_language`: Each language associated with a title is stored for detailed analysis.
    - `dim_persons`: Each individual, such as actors or directors, is stored with key attributes like `primaryName` and `birthYear`.
    - `dim_profession`: This dimension helps classify people by profession (e.g., Director, Writer, Actor).
    - `dim_region`: Allows for tracking movie releases across different geographical regions.
- **Bridge Tables**:
    - `Bridge_person_profession`: Links individuals to their roles and professions, providing insights into the diverse roles they have played in the industry.

- `Bridge_Title_Akas`: Associates alternative titles and regions to each movie.
- `Bridge_title_genres`: Bridges movies to the genres they belong to, enabling genre-based analysis.
- `Bridge_title_language`: Associates titles to different languages.
- `Bridge_title_regions`: Helps track the global distribution of movies by region.

6. ETL Process Using Azure Data Factory (ADF)

Overview of ADF Pipelines

Our project leveraged Azure Data Factory to create a robust ETL framework that processed the extensive IMDb dataset. We designed multiple interconnected pipelines that extracted data from source files, transformed them according to business requirements, and loaded them into our dimensional model in Snowflake. The pipeline architecture included:

- Master Pipeline: Orchestrated the entire ETL workflow with dependency management
- Extraction Pipelines: Pulled data from staging tables containing IMDb datasets
- Transformation Pipelines: Applied business rules and data cleansing operations
- Loading Pipelines: Populated the dimensional model with processed data

Transformation and Data Flow

We implemented complex transformations using ADF Data Flows to prepare the data for analytical use:

- Genre Parsing: Split the comma-delimited genres field into individual genre records
- Director and Writer Extraction: Parsed the pipe-delimited crew fields to create normalized relationships
- Data Type Standardization: Converted inconsistent data types to ensure compatibility across sources
- Null Handling: Applied business rules to handle missing values in critical fields like ratings and runtime
- Derived Metrics: Created calculated fields to support analytical requirements

Version Control and Git Integration

Our development approach incorporated DevOps best practices through Git integration:

- Feature Branching: Each team member worked on isolated feature branches for specific pipeline components
- Pull Request Workflow: Code reviews were conducted through pull requests before merging to the main branch
- CI/CD Integration: Automated testing and deployment pipelines ensured code quality
- Environment Promotion: Controlled promotion of changes from development to testing to production
- Change Documentation: Maintained detailed commit messages and release notes for all pipeline modifications

7. Data Visualization and Reporting

Report Generation

Based on the screenshots, we developed a comprehensive Power BI solution with four distinct dashboard pages, each addressing specific analytical needs:

1. Movie Overview Insights: Provides high-level metrics and trends about the movie collection
2. Director Analysis: Focuses on director productivity and performance metrics
3. TV Series Analysis: Examines television content patterns and ratings
4. TV Special vs Video Game: Compares different non-movie content types

Each dashboard follows a consistent design pattern with KPI cards at the top, interactive filters in the middle section, and detailed visualizations below.

Tools Used for Reporting

We utilized Power BI Desktop for our visualization solution, leveraging its native visuals and data modeling capabilities:

- Direct Query Connection: Established connection to our Snowflake integration schema
- Star Schema Optimization: Optimized the data model for performance using star schema principles
- DAX Measures: Created complex calculations using Data Analysis Expressions
- Visual Formatting: Applied consistent formatting and color schemes across all dashboards

- Interactive Filters: Implemented cross-filtering capabilities between visualizations

Examples of Key Visualizations

As evidenced in the screenshots:

1. Movie Overview Insights Dashboard:
    - KPI cards showing 11.50M total movies, 6.95 average rating, 154M runtime minutes, and Action as highest-rated genre
    - Line chart displaying movie production trends from 1874-2021, showing dramatic increase around 2000
    - Clustered column chart comparing genre performance by average rating and count
    - Donut chart breaking down 9.31M titles by genre and type
    - Bar chart comparing ratings across different title types
2. Director Analysis Dashboard:
    - KPI cards showing 944.67K directors, with nm0000005 having most titles and nm0000465 having highest ratings
    - Donut chart displaying director distribution across genres
    - Detailed table showing director performance metrics including title count and average ratings
3. TV Series Analysis Dashboard:
    - KPI cards showing 379.99K TV series, 421.04 average episodes per series, and highest-rated series
    - Line chart tracking TV series production trends
    - Clustered column chart comparing genre performance
    - Bar chart showing average ratings by title type
4. TV Special vs Video Game Dashboard:
    - KPI cards showing 51.59K TV specials and 60.18K TV mini-series, both with 6.95 average ratings
    - Line chart comparing rating distributions between content types
    - Bar chart comparing total runtime minutes between TV specials and mini-series

8. Business Insights and Analysis

Movie Insights

The dashboards reveal several critical insights about the movie collection:

- The database contains an impressive 11.50 million movies with a global average rating of 6.95
- Action emerged as the highest-rated genre among all movies
- Movie production shows a dramatic spike around the year 2000, with relatively modest production before 1990
- The total runtime of all movies amounts to 154 million minutes, representing significant content volume

Crew and Cast Analysis

The Director Analysis dashboard provides valuable insights:

- The database contains information on 944,670 directors across various genres
- Director nm0000005 has directed the most titles, demonstrating exceptional productivity
- Director nm0000465 has achieved the highest average ratings, indicating superior quality
- The donut chart reveals a diverse distribution of directors across genres, with several genres having similar representation

Movie Trend Analysis

Trend analysis from the dashboards shows:

- Movie production remained relatively stable until approximately 1990
- A dramatic increase in production occurred between 1990-2010, peaking around 2005
- The clustered column chart reveals that certain genres (shown by yellow bars) have significantly higher production volumes
- Rating distributions vary notably across different title types, with some formats consistently achieving higher ratings

Season and Episode Insights

The TV Series Analysis dashboard reveals:

- The collection includes 379,990 TV series with an average of 421.04 episodes per series
- Series tt0127236 achieved the highest ratings among all TV series
- Series tt0041951 holds the record for maximum seasons
- TV series production follows similar trends to movies, with significant growth after 1990

Region and Country-Based Insights

While specific regional visualizations aren't shown in the screenshots, the filter panels indicate:

- Region filtering is available on multiple dashboards, allowing for geographical analysis
- The global nature of the dataset is evident from the comprehensive metrics (11.50M movies)
- Regional preferences can be analyzed through the genre and rating filters

9. Challenges and Solutions

Data Quality and Integration Challenges

During implementation, we encountered several challenges:

- Inconsistent Identifiers: The tconst and nconst identifiers required standardization across tables
- Missing Values: Critical fields like ratings and runtime contained significant null values
- Duplicate Entries: Multiple entries existed for the same titles with slight variations
- Genre Parsing: The comma-delimited genre field required special handling for proper analysis

Solutions:

- Implemented data profiling to identify and catalog data quality issues
- Created standardized cleaning procedures in ADF Data Flows
- Established business rules for handling missing values based on context
- Developed deduplication logic using combination of fields

Technological Challenges

The implementation faced several technical hurdles:

- Performance Bottlenecks: Processing the 11.50M movie records created performance issues
- Complex Relationships: Modeling the many-to-many relationships between titles and genres/regions
- Filter Context: Ensuring proper filter propagation across related visualizations
- Memory Limitations: Managing the large dataset within Power BI's memory constraints

Solutions:

- Implemented incremental loading patterns in ADF
- Created bridge tables to handle many-to-many relationships
- Optimized DAX measures for performance
- Used Direct Query where appropriate to manage memory usage

Business Requirement Challenges

Meeting business requirements presented challenges:

- Balancing Detail and Performance: Providing granular analysis while maintaining dashboard responsiveness
- Intuitive User Experience: Creating an interface accessible to users with varying technical expertise
- Consistent Metrics: Ensuring consistent calculation of metrics across different analysis dimensions
- Actionable Insights: Transforming raw data into business-relevant insights

Solutions:

- Implemented drill-through capabilities for detailed analysis
- Created consistent visual layouts and filter interactions
- Documented key metrics and their calculation methodology
- Focused visualizations on answering specific business questions

10. Conclusion

Summary of Accomplishments

Our project successfully delivered:

- A comprehensive dimensional model integrating multiple IMDb data sources
- Automated ETL pipelines using Azure Data Factory
- Four interactive Power BI dashboards providing insights into movies, directors, TV series, and special content
- Analysis of 11.50M movies and 379.99K TV series with detailed metrics

Impact on the Business

The solution provides significant business value:

- Comprehensive Content Analysis: Enables understanding of content trends across time, genres, and formats

- Performance Evaluation: Allows comparison of content quality through rating analysis
- Production Insights: Provides historical context for content production volumes and patterns
- Format Comparison: Facilitates comparison between different content types (movies, TV series, specials)

Future Recommendations

Based on our implementation, we recommend:

- Sentiment Analysis Integration: Incorporate viewer sentiment data to complement ratings
- Predictive Analytics: Develop models to predict potential ratings for new content
- Streaming Platform Integration: Add data from streaming platforms to analyze viewing patterns
- Mobile-Optimized Dashboards: Create mobile versions for executives on the go
- Automated Insights: Implement AI-driven insight generation to highlight key findings automatically