Epic 1:

- Epic1_DataFactoryPipeline.png
  - A jpg of the pipeline to get data into the staging table.
  - The image shows that the pipeline worked/succeeded .
  - If you want this turned in another way let me know, I could not figure out how to export it.
- Epic1_ERDiagram.png
  - Shows new tables (award, ceremony, academy_award, cast_acclaim, crew_acclaim, person_acclaim) in relation to old tables (movie, movie_crew, movie_cast, person)
  - Acclaim pct (percent) is calculated as number of nominations divided by number of appearances (for person, for cast order, for job)
  - Acclaim tables (cast_acclaim, job_acclaim, person_acclaim) are each tied to their respective tables (movie_cast, movie_crew, person)
  - ceremony_id and award_id are unique identifiers for their respective tables
- Epic1_View.png
  - Image showing that the view does work.
- Epic1_SchemaScript.sql
  - Creates staging table for data factory pipeline
  - Creates tables for the schema
- Epic1_LoadScript.sql
  - Loads data from queries into schema tables created by Epic1_SchemaScript.sql
  - Also updates null values to zero for the purpose of calculating acclaim pct
- Epic1_View.sql
  - Creates the view for movies and their acclaim percentages

Epic 3:

- Epic3_ERDiagram.png
  - Very simple schema
  - Users can be associated with multiple stream services and stream services can be associated to multiple users
  - Review table holds ratings and comments (reviews) for a movie, acts as a many to many bridge between user and movie
  - User table has variables like sex and age to determine user demographics
- Epic3_SchemaLoadScript.sql
  - Acts as both schema and load script
  - Builds staging table and data to be inserted into it
  - Builds schema tables
  - Loads data into schema from staging table through queries
- Epic3_StoredProcedures.sql
  - Creates the 12 required procedures and test code underneath them
- Epic3_Views.sql
  - Creates the 3 required views with test code underneath each one