Documentation of the pipeline:

The pipeline consists of three main functions - `extract_data`, `transform_data`, and `load_data`. The `extract_data` function retrieves the data from a CSV file or Redis cache. The `transform_data` function performs data cleaning and transformation by converting `call_date` to a datetime object, `call_duration` to `timedelta`, `call_cost` to float, and creating new columns for `call_start_time` and `call_day_of_week`. Finally, the `load_data` function loads the transformed data into a PostgreSQL database table. The `data_pipeline` function calls these three functions in order and prints out the original and transformed dataframes.

Best practices used during the implementation:

1. Error logging: The code includes logging error messages to help identify and troubleshoot issues that may arise during the data processing pipeline.
2. Data caching: The extract_data function caches the data in Redis to improve performance and reduce the number of times the data needs to be read from the CSV file.
3. Modular design: The code is structured in a way that each function performs a specific task, making the pipeline easier to understand and modify.

Recommendations for deployment and running the pipeline with a cloud-based provider:

1. Use a containerization tool like Docker to package the pipeline and its dependencies. This makes it easier to deploy and run the pipeline on different machines without worrying about compatibility issues.
2. Use a managed cloud service for the PostgreSQL database to simplify the process of setting up and managing the database. Some options include Amazon RDS, Google Cloud SQL, and Microsoft Azure Database.
3. Implement an automated deployment process using a Continuous Integration/Continuous Deployment (CI/CD) tool like Jenkins, Travis CI, or CircleCI. This can help ensure that changes to the pipeline are tested and deployed quickly and consistently.