Purpose of the Code

1. **Automates ETL (Extract, Transform, Load)**:
   - **Extract**: Reads JSON files from S3.
   - **Transform**: Normalizes and structures the JSON data using Pandas.
   - **Load**: Writes the transformed data to S3 in Parquet format.
2. **Data Catalog Registration**:
   - Updates the AWS Glue Data Catalog, enabling the transformed data to be queried directly with Athena.
3. **Scalable Data Processing**:
   - Processes incoming files automatically, triggered by S3 events

----

Step-by-Step Explanation

1. Environment Variables

The function uses the following environment variables:

- `s3_cleansed_layer`: The destination S3 bucket or folder for storing the transformed Parquet files.
- `glue_catalog_db_name`: The Glue database name where the table is registered.
- `glue_catalog_table_name`: The Glue table name for storing metadata about the Parquet files.
- `write_data_operation`: The write mode for the Glue catalog table (`append`, `overwrite`, etc.).

2. Trigger Event

The Lambda function is triggered by an **S3 event** when a JSON file is uploaded to the source bucket.

3. Input File Details

The function retrieves details about the uploaded file:

- `bucket`: The name of the S3 bucket where the file was uploaded.
- `key`: The key (path) of the uploaded file.

4. Read JSON File

The function uses **AWS Data Wrangler (awswrangler)** to:

- Read the JSON file from the source S3 bucket

```
df_raw = wr.s3.read_json('s3://{}/{}'.format(bucket, key))
```

5. Normalize the JSON Data

If the JSON file contains nested structures, the code normalizes the data using **pandas**:

```
python
CopyEdit
df_step_1 = pd.json_normalize(df_raw['items'])
```

This flattens the JSON structure, creating a structured table (DataFrame) suitable for analysis.

6. Write Transformed Data to S3

The cleaned and structured data (`df_step_1`) is written to the destination S3 bucket in **Parquet** format. The function also registers this data in the Glue Data Catalog:

```
python
CopyEdit
wr_response = wr.s3.to_parquet(
    df=df_step_1,
    path=os_input_s3_cleansed_layer,
    dataset=True,
    database=os_input_glue_catalog_db_name,
    table=os_input_glue_catalog_table_name,
    mode=os_input_write_data_operation
)
```

- **Key Parameters**:
  - `path`: Destination S3 path for the Parquet files.
  - `database`: Glue database for the table.
  - `table`: Glue table name.
  - `mode`: Specifies how data should be written (`append`, `overwrite`).

7. Error Handling

If any error occurs during the process (e.g., reading the file or writing to S3), the function logs the error and raises an exception:

```
python
CopyEdit
except Exception as e:
    print(e)
    raise e
```