```
import pandas as pd

if 'transformer' not in globals():
    from mage_ai.data_preparation.decorators import transformer
if 'test' not in globals():
    from mage_ai.data_preparation.decorators import test


@transformer
def transform(df, *args, **kwargs):

    # Specify your transformation logic here
    df['tpep_pickup_datetime'] = pd.to_datetime(df['tpep_pickup_datetime'])
    df['tpep_dropoff_datetime'] = pd.to_datetime(df['tpep_dropoff_datetime'])

    datetime_dim =
df[['tpep_pickup_datetime','tpep_dropoff_datetime']].drop_duplicates().reset_index(dr
op=True)
    datetime_dim['pick_hour'] = datetime_dim['tpep_pickup_datetime'].dt.hour
    datetime_dim['pick_day'] = datetime_dim['tpep_pickup_datetime'].dt.day
    datetime_dim['pick_month'] = datetime_dim['tpep_pickup_datetime'].dt.month
    datetime_dim['pick_year'] = datetime_dim['tpep_pickup_datetime'].dt.year
    datetime_dim['pick_weekday'] = datetime_dim['tpep_pickup_datetime'].dt.weekday


    datetime_dim['drop_hour'] = datetime_dim['tpep_dropoff_datetime'].dt.hour
    datetime_dim['drop_day'] = datetime_dim['tpep_dropoff_datetime'].dt.day
    datetime_dim['drop_month'] = datetime_dim['tpep_dropoff_datetime'].dt.month
    datetime_dim['drop_year'] = datetime_dim['tpep_dropoff_datetime'].dt.year
    datetime_dim['drop_weekday'] =
datetime_dim['tpep_dropoff_datetime'].dt.weekday
    datetime_dim['datetime_id'] = datetime_dim.index
    datetime_dim = datetime_dim[['datetime_id', 'tpep_pickup_datetime', 'pick_hour',
'pick_day', 'pick_month', 'pick_year', 'pick_weekday',
                         'tpep_dropoff_datetime', 'drop_hour', 'drop_day', 'drop_month',
'drop_year', 'drop_weekday']]

    passenger_count_dim =
df[['passenger_count']].drop_duplicates().reset_index(drop=True)
    passenger_count_dim['passenger_count_id'] = passenger_count_dim.index
```

```
    passenger_count_dim =
passenger_count_dim[['passenger_count_id','passenger_count']]

    trip_distance_dim = df[['trip_distance']].drop_duplicates().reset_index(drop=True)
    trip_distance_dim['trip_distance_id'] = trip_distance_dim.index
    trip_distance_dim = trip_distance_dim[['trip_distance_id','trip_distance']]

    rate_code_type = {
    1:"Standard rate",
    2:"JFK",
    3:"Newark",
    4:"Nassau or Westchester",
    5:"Negotiated fare",
    6:"Group ride"
    }

    rate_code_dim = df[['RatecodeID']].drop_duplicates().reset_index(drop=True)
    rate_code_dim['rate_code_id'] = rate_code_dim.index
    rate_code_dim['rate_code_name'] =
rate_code_dim['RatecodeID'].map(rate_code_type)
    rate_code_dim = rate_code_dim[['rate_code_id','RatecodeID','rate_code_name']]

    pickup_location_dim = df[['pickup_longitude',
'pickup_latitude']].drop_duplicates().reset_index(drop=True)
    pickup_location_dim['pickup_location_id'] = pickup_location_dim.index
    pickup_location_dim =
pickup_location_dim[['pickup_location_id','pickup_latitude','pickup_longitude']]


    dropoff_location_dim = df[['dropoff_longitude',
'dropoff_latitude']].drop_duplicates().reset_index(drop=True)
    dropoff_location_dim['dropoff_location_id'] = dropoff_location_dim.index
    dropoff_location_dim =
dropoff_location_dim[['dropoff_location_id','dropoff_latitude','dropoff_longitude']]

    payment_type_name = {
    1:"Credit card",
    2:"Cash",
    3:"No charge",
    4:"Dispute",
```

```
    5:"Unknown",
    6:"Voided trip"
    }
    payment_type_dim = df[['payment_type']].drop_duplicates().reset_index(drop=True)
    payment_type_dim['payment_type_id'] = payment_type_dim.index
    payment_type_dim['payment_type_name'] =
payment_type_dim['payment_type'].map(payment_type_name)
    payment_type_dim =
payment_type_dim[['payment_type_id','payment_type','payment_type_name']]


    fact_table = df.merge(passenger_count_dim, on='passenger_count') \
            .merge(trip_distance_dim,on='trip_distance') \
            .merge(rate_code_dim, on='RatecodeID') \
            .merge(pickup_location_dim, on=['pickup_longitude','pickup_latitude']) \
            .merge(dropoff_location_dim, on=['dropoff_longitude','dropoff_latitude'])\
            .merge(datetime_dim, on=['tpep_pickup_datetime','tpep_dropoff_datetime'])
\
            .merge(payment_type_dim, on='payment_type') \
            [['VendorID', 'datetime_id', 'passenger_count_id',
            'trip_distance_id', 'rate_code_id', 'store_and_fwd_flag', 'pickup_location_id',
'dropoff_location_id',
            'payment_type_id', 'fare_amount', 'extra', 'mta_tax', 'tip_amount',
'tolls_amount',
            'improvement_surcharge', 'total_amount']]

    return {"datetime_dim":datetime_dim.to_dict(orient="dict"),
    "passenger_count_dim":passenger_count_dim.to_dict(orient="dict"),
    "trip_distance_dim":trip_distance_dim.to_dict(orient="dict"),
    "rate_code_dim":rate_code_dim.to_dict(orient="dict"),
    "pickup_location_dim":pickup_location_dim.to_dict(orient="dict"),
    "dropoff_location_dim":dropoff_location_dim.to_dict(orient="dict"),
    "payment_type_dim":payment_type_dim.to_dict(orient="dict"),
    "fact_table":fact_table.to_dict(orient="dict"),
    }


@test
def test_output(output, *args) -> None:
    """
```

Template code for testing the output of the block.
"""
assert output is not None, 'The output is undefined'