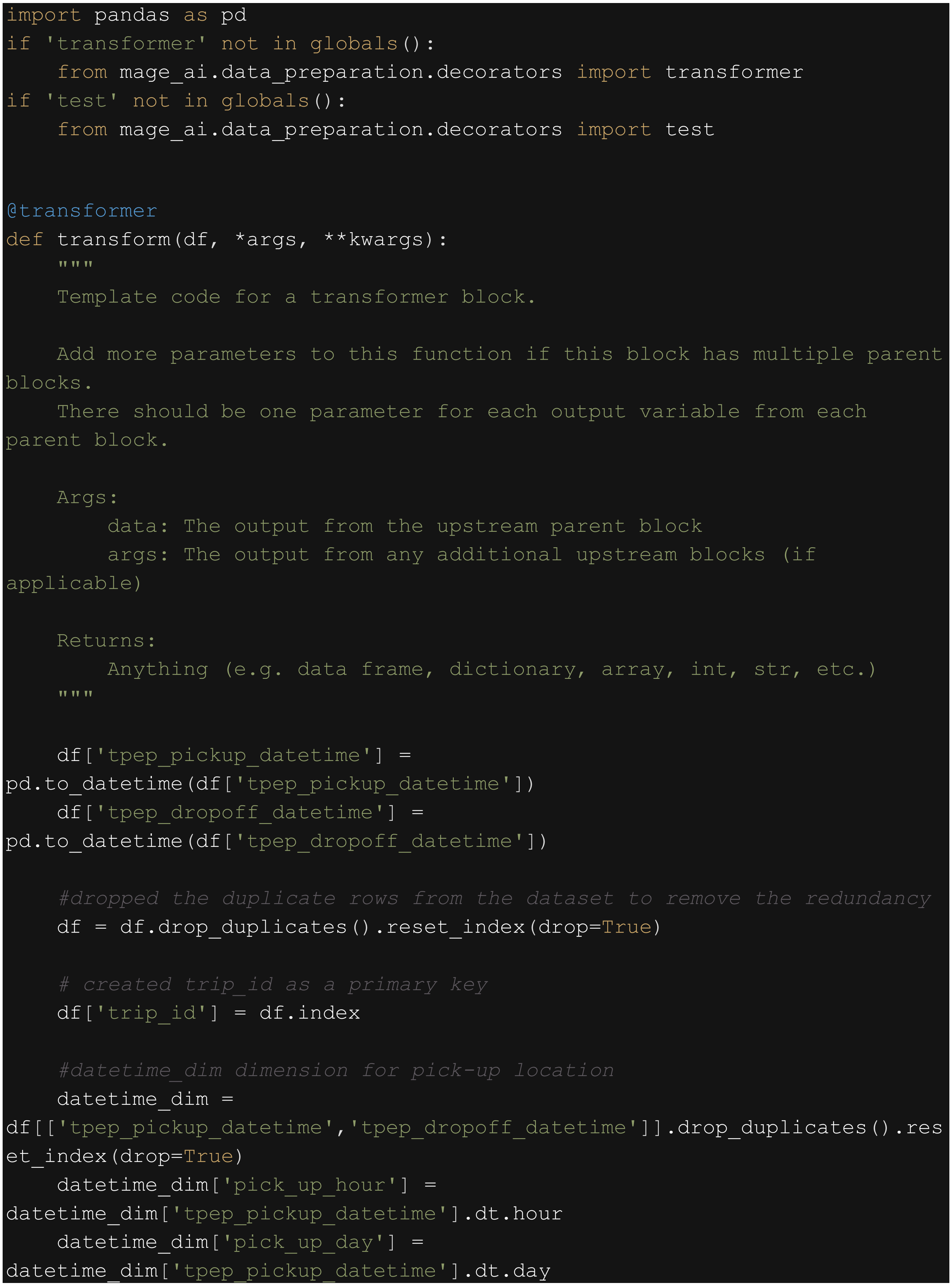


```
import pandas as pd
if 'transformer' not in globals():
    from mage_ai.data_preparation.decorators import transformer
if 'test' not in globals():
    from mage_ai.data_preparation.decorators import test


@transformer
def transform(df, *args, **kwargs):
    """
    Template code for a transformer block.

    Add more parameters to this function if this block has multiple parent
blocks.
    There should be one parameter for each output variable from each
parent block.

    Args:
        data: The output from the upstream parent block
        args: The output from any additional upstream blocks (if
applicable)

    Returns:
        Anything (e.g. data frame, dictionary, array, int, str, etc.)
    """

    df['tpep_pickup_datetime'] =
pd.to_datetime(df['tpep_pickup_datetime'])
    df['tpep_dropoff_datetime'] =
pd.to_datetime(df['tpep_dropoff_datetime'])

    #dropped the duplicate rows from the dataset to remove the redundancy
    df = df.drop_duplicates().reset_index(drop=True)

    # created trip_id as a primary key
    df['trip_id'] = df.index

    #datetime_dim dimension for pick-up location
    datetime_dim =
df[['tpep_pickup_datetime','tpep_dropoff_datetime']].drop_duplicates().res
et_index(drop=True)
    datetime_dim['pick_up_hour'] =
datetime_dim['tpep_pickup_datetime'].dt.hour
    datetime_dim['pick_up_day'] =
datetime_dim['tpep_pickup_datetime'].dt.day
```

```
    datetime_dim['pick_up_month'] =
datetime_dim['tpep_pickup_datetime'].dt.month
    datetime_dim['pick_up_year'] =
datetime_dim['tpep_pickup_datetime'].dt.year
    datetime_dim['pick_up_weekday'] =
datetime_dim['tpep_pickup_datetime'].dt.weekday

    #datetime_dim dimension for drop-off location

    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_id -> primary key
    datetime_dim['datetime_id'] = datetime_dim.index

    datetime_dim=datetime_dim[['datetime_id','tpep_pickup_datetime',
'tpep_dropoff_datetime', 'pick_up_hour',
        'pick_up_day', 'pick_up_month', 'pick_up_year','pick_up_weekday',
'drop_hour', 'drop_day',
        'drop_month', 'drop_year','drop_weekday',
        ]]
    # Passenger dimension
    passenger_count_dim =
df[['passenger_count']].drop_duplicates().reset_index(drop=True)
    passenger_count_dim['passenger_count_id'] =
passenger_count_dim['passenger_count'].index
    passenger_count_dim =
passenger_count_dim[['passenger_count_id','passenger_count' ]]

    #Trip distance dimension
    Trip_distance_dim
=df[['trip_distance']].drop_duplicates().reset_index(drop=True)
    Trip_distance_dim['trip_distance_id']=Trip_distance_dim['trip_distance
'].index
    Trip_distance_dim = Trip_distance_dim[['trip_distance_id',
'trip_distance']]
```

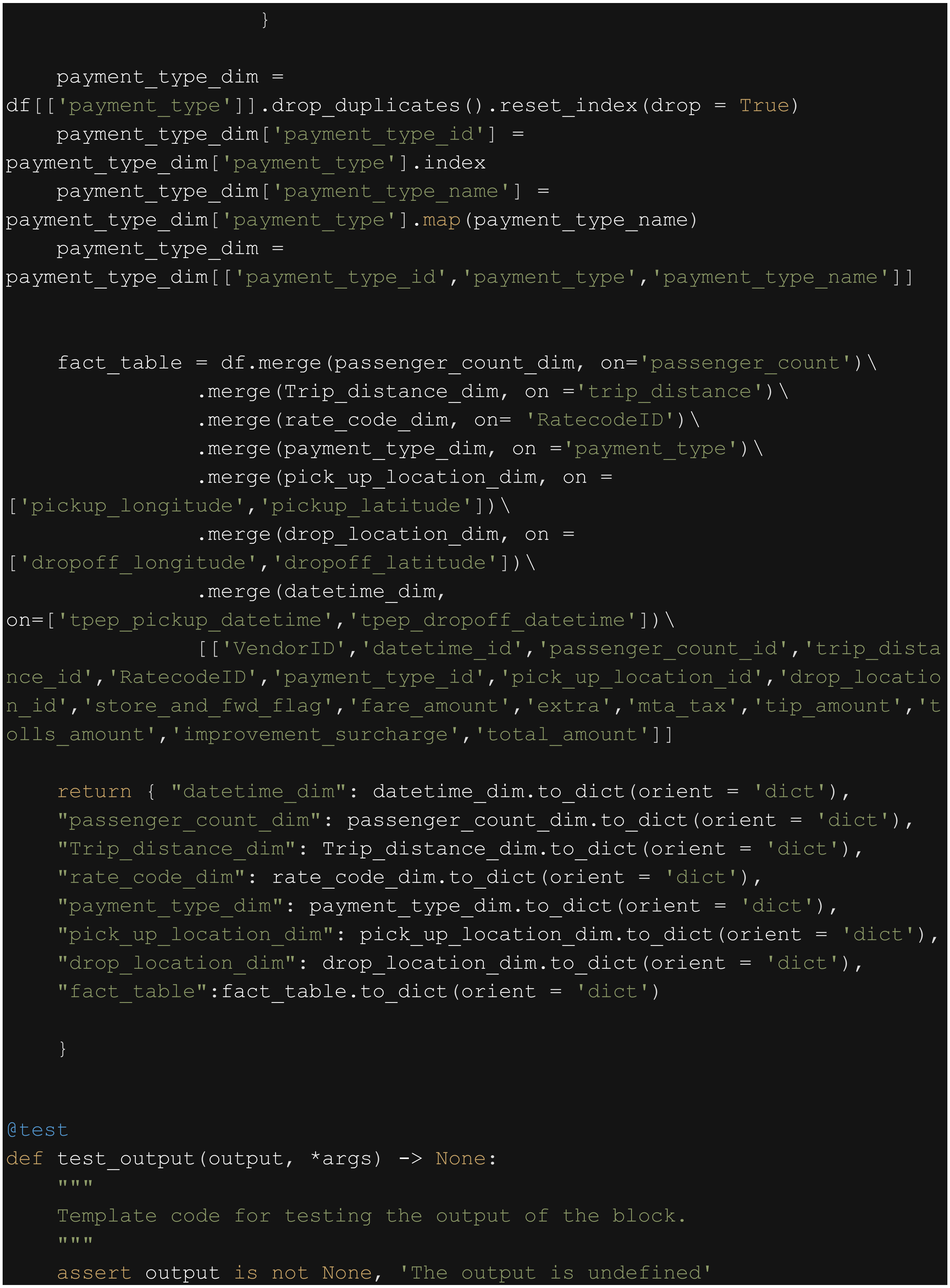
```
    #Rate code dimension
    rate_code_type ={
                    1: "Standard rate",
                    2: "JFK",
                    3: "Newark",
                    4: "Nassau or Westchester",
                    5: "Negotiated fare",
                    6: "Group ride",
                }

    rate_code_dim
=df[['RatecodeID']].drop_duplicates().reset_index(drop=True)
    rate_code_dim['Rate_code_ID'] = rate_code_dim['RatecodeID'].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']]


    #Pick Up Location dimension
    pick_up_location_dim=df[['pickup_longitude','pickup_latitude']].drop_d
uplicates().reset_index(drop=True)
    pick_up_location_dim['pick_up_location_id'] =
pick_up_location_dim.index
    pick_up_location_dim =
pick_up_location_dim[['pick_up_location_id','pickup_longitude','pickup_lat
itude']]

    #drop off location dimesnion
    drop_location_dim=df[['dropoff_longitude','dropoff_latitude']].drop_du
plicates().reset_index(drop=True)
    drop_location_dim['drop_location_id'] = drop_location_dim.index
    drop_location_dim =
drop_location_dim[['drop_location_id','dropoff_longitude','dropoff_latitud
e']]


    #Payment Dimension
    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['payment_type'].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(payment_type_dim, on ='payment_type')\
               .merge(pick_up_location_dim, on =
['pickup_longitude','pickup_latitude'])\
               .merge(drop_location_dim, on =
['dropoff_longitude','dropoff_latitude'])\
               .merge(datetime_dim,
on=['tpep_pickup_datetime','tpep_dropoff_datetime'])\
               [['VendorID','datetime_id','passenger_count_id','trip_dista
nce_id','RatecodeID','payment_type_id','pick_up_location_id','drop_locatio
n_id','store_and_fwd_flag','fare_amount','extra','mta_tax','tip_amount','t
olls_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'),
    "payment_type_dim": payment_type_dim.to_dict(orient = 'dict'),
    "pick_up_location_dim": pick_up_location_dim.to_dict(orient = 'dict'),
    "drop_location_dim": drop_location_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'
```