```python
import mysql.connector
from datetime import datetime

# Function to fetch and transform data for agents
def fetch_and_transform_agent_data(conn):
    cursor = conn.cursor()

    query = "SELECT * FROM agent;"
    cursor.execute(query)
    rows = cursor.fetchall()

    transformed_data = []
    for row in rows:
        if None in row:  # Check for NULL values in the row
            print(f"Skipping agent record due to NULL values: {row}")
            continue

        transformed_row = {
            'AgentID': row[0],
            'AgentName': row[1],
            'AgentAddress': row[2],
            'AgentPhone': row[3]
        }
        transformed_data.append(transformed_row)

    cursor.close()
    return transformed_data

# Function to fetch and transform data for directors
def fetch_and_transform_director_data(conn):
    cursor = conn.cursor()

    query = "SELECT * FROM director;"
    cursor.execute(query)
    rows = cursor.fetchall()

    transformed_data = []
    for row in rows:
        if None in row:  # Check for NULL values in the row
            print(f"Skipping director record due to NULL values: {row}")
            continue

        transformed_row = {
            'DirectorID': row[0],
            'DirectorName': row[1],
```

```
            'BirthDate': row[2].strftime('%Y-%m-%d'),  # Format DATETIME to DATE
            'Gender': row[3],
            'AgentID': row[4]
        }
        transformed_data.append(transformed_row)

    cursor.close()
    return transformed_data

# Function to fetch and transform data for actor
def fetch_and_transform_actor_data(conn):
    cursor = conn.cursor()

    query = "SELECT * FROM actor;"
    cursor.execute(query)
    rows = cursor.fetchall()

    transformed_data = []
    for row in rows:
        if None in row:  # Check for NULL values in the row
            print(f"Skipping agent record due to NULL values: {row}")
            continue

        transformed_row = {
            'ActorID': row[0],
            'ActorName': row[1],
            'BirthDate': row[2].strftime('%Y-%m-%d'),
            'Nationality': row[3],
            'AgentID': row[4],
            'Gender': row[5],
        }
        transformed_data.append(transformed_row)

    cursor.close()
    return transformed_data

# Function to fetch and transform data for distributor
def fetch_and_transform_distributor_data(conn):
    cursor = conn.cursor()

    query = "SELECT * FROM distributor;"
    cursor.execute(query)
    rows = cursor.fetchall()

    transformed_data = []
```

```
    for row in rows:
        if None in row:  # Check for NULL values in the row
            print(f"Skipping agent record due to NULL values: {row}")
            continue

        transformed_row = {
            'DistributorID': row[0],
            'DistName': row[1],
            'DistAddress': row[2],
            'DistPhone': row[3]
        }
        transformed_data.append(transformed_row)

    cursor.close()
    return transformed_data

# Function to fetch and transform data for movie
def fetch_and_transform_movie_data(conn):
    cursor = conn.cursor()

    query = """
        SELECT m.MovieID, m.Title, m.DirectorID, m.MovieRating,
               m.ReleaseDate, m.Producer, m.DistributorID, m.ProductionBudget,
               mg.Genre
        FROM movie m
        LEFT JOIN moviegenre mg ON m.MovieID = mg.MovieID;
    """
    cursor.execute(query)
    rows = cursor.fetchall()

    transformed_data = []
    for row in rows:
        if None in row:  # Check for NULL values in the row
            print(f"Skipping agent record due to NULL values: {row}")
            continue

        transformed_row = {
            'MovieID': row[0],
            'Title': row[1],
            'DirectorID': row[2],
            'MovieRating': row[3],
            'ReleaseDate': row[4].strftime('%Y-%m-%d'),
            'Producer': row[5],
            'DistributorID': row[6],
            'ProductionBudget': row[7],
```

```
            'Genre': row[8]
        }
        transformed_data.append(transformed_row)

    cursor.close()
    return transformed_data

# Function to fetch and transform data for movie cast
def fetch_and_transform_moviecast_data(conn):
    cursor = conn.cursor()

    query = "SELECT * FROM moviecast;"
    cursor.execute(query)
    rows = cursor.fetchall()

    transformed_data = []
    for row in rows:
        if None in row:  # Check for NULL values in the row
            print(f"Skipping agent record due to NULL values: {row}")
            continue

        transformed_row = {
            'MovieID': row[0],
            'ActorID': row[1],
            'Salary': row[2]
        }
        transformed_data.append(transformed_row)

    cursor.close()
    return transformed_data

# Function to fetch and transform data for movie genre
def fetch_and_transform_moviegenre_data(conn):
    cursor = conn.cursor()

    query = "SELECT * FROM moviegenre;"
    cursor.execute(query)
    rows = cursor.fetchall()

    transformed_data = []
    for row in rows:
        if None in row:  # Check for NULL values in the row
            print(f"Skipping director record due to NULL values: {row}")
            continue
```

```
        transformed_row = {
            'MovieID': row[0],
            'Genre': row[1]
        }
        transformed_data.append(transformed_row)

    cursor.close()
    return transformed_data

# Function to fetch and transform data for movie reviews
def fetch_and_transform_moviereview_data(conn):
    cursor = conn.cursor()

    query = "SELECT * FROM moviereviews;"
    cursor.execute(query)
    rows = cursor.fetchall()

    transformed_data = []
    for row in rows:
        if None in row:  # Check for NULL values in the row
            print(f"Skipping agent record due to NULL values: {row}")
            continue

        transformed_row = {
            'MovieID': row[0],
            'ReviewerID': row[1],
            'ReviewRating': row[2]
        }
        transformed_data.append(transformed_row)

    cursor.close()
    return transformed_data

# Function to fetch and transform data for reviewer
def fetch_and_transform_reviewer_data(conn):
    cursor = conn.cursor()

    query = "SELECT * FROM reviewer;"
    cursor.execute(query)
    rows = cursor.fetchall()

    transformed_data = []
    for row in rows:
        if None in row:  # Check for NULL values in the row
            print(f"Skipping agent record due to NULL values: {row}")
```

```
            continue

        transformed_row = {
            'ReviewerID': row[0],
            'ReviewerName': row[1],
            'ReviewerClass': row[2],
            'BirthDate': row[3].strftime('%Y-%m-%d')
        }
        transformed_data.append(transformed_row)

    cursor.close()
    return transformed_data

# Function to fetch and transform data for showing
def fetch_and_transform_showing_data(conn):
    cursor = conn.cursor()

    query = "SELECT * FROM showing;"
    cursor.execute(query)
    rows = cursor.fetchall()

    transformed_data = []
    for row in rows:
        if None in row:  # Check for NULL values in the row
            print(f"Skipping agent record due to NULL values: {row}")
            continue

        transformed_row = {
            'MovieID': row[0],
            'TheaterID': row[1],
            'StartDate': row[2].strftime('%Y-%m-%d'),
            'EndDate': row[3].strftime('%Y-%m-%d'),
            'BoxOffice': row[4]
        }
        transformed_data.append(transformed_row)

    cursor.close()
    return transformed_data

# Function to fetch and transform data for theater
def fetch_and_transform_theater_data(conn):
    cursor = conn.cursor()

    query = "SELECT * FROM theater;"
    cursor.execute(query)
```

```
    rows = cursor.fetchall()

    transformed_data = []
    for row in rows:
        if None in row:  # Check for NULL values in the row
            print(f"Skipping agent record due to NULL values: {row}")
            continue

        transformed_row = {
            'TheaterID': row[0],
            'TheaterName': row[1],
            'Location': row[2],
            'NoScreens': row[3],
            'Seats': row[4]
        }
        transformed_data.append(transformed_row)

    cursor.close()
    return transformed_data

# Function to insert agent data
def insert_agent_data(conn, data):
    cursor = conn.cursor()

    insert_query = """
        INSERT INTO dimagent (AgentID, AgentName, AgentAddress, AgentPhone)
        VALUES (%s, %s, %s, %s)
        ON DUPLICATE KEY UPDATE
        AgentName = VALUES(AgentName),
        AgentAddress = VALUES(AgentAddress),
        AgentPhone = VALUES(AgentPhone);
    """

    try:
        for row in data:
            record = (row['AgentID'], row['AgentName'], row['AgentAddress'],
row['AgentPhone'])
            cursor.execute(insert_query, record)

        conn.commit()
        print("Agent data inserted successfully.")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    finally:
        cursor.close()
```

```
# Function to insert director data
def insert_director_data(conn, data):
    cursor = conn.cursor()

    insert_query = """
        INSERT INTO dimdirector (DirectorID, DirectorName, BirthDate, Gender,
AgentID)
        VALUES (%s, %s, %s, %s, %s)
        ON DUPLICATE KEY UPDATE
        BirthDate = VALUES(BirthDate);
    """

    try:
        for row in data:
            record = (row['DirectorID'], row['DirectorName'], row['BirthDate'],
row['Gender'], row['AgentID'])
            cursor.execute(insert_query, record)

        conn.commit()
        print("Director data inserted successfully.")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    finally:
        cursor.close()

# Function to insert actor data
def insert_actor_data(conn, data):
    cursor = conn.cursor()

    insert_query = """
        INSERT INTO dimactor (ActorID, ActorName, BirthDate, Nationality,
AgentID, Gender)
        VALUES (%s, %s, %s, %s, %s, %s)
        ON DUPLICATE KEY UPDATE
        ActorName = VALUES(ActorName),
        BirthDate = VALUES(BirthDate),
        Nationality = VALUES(Nationality),
        AgentID = VALUES(AgentID),
        Gender = VALUES(Gender);
    """

    try:
        for row in data:
```

```
            record = (row['ActorID'], row['ActorName'], row['BirthDate'],
row['Nationality'], row['AgentID'], row['Gender'])
            cursor.execute(insert_query, record)

        conn.commit()
        print("Actor data inserted successfully.")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    finally:
        cursor.close()

# Function to insert distributor data
def insert_distributor_data(conn, data):
    cursor = conn.cursor()

    insert_query = """
        INSERT INTO dimdistributor (DistributorID, DistName, DistAddress,
DistPhone)
        VALUES (%s, %s, %s, %s)
        ON DUPLICATE KEY UPDATE
        DistName = VALUES(DistName),
        DistAddress = VALUES(DistAddress),
        DistPhone = VALUES(DistPhone);
    """

    try:
        for row in data:
            record = (row['DistributorID'], row['DistName'], row['DistAddress'],
row['DistPhone'])
            cursor.execute(insert_query, record)

        conn.commit()
        print("Distributor data inserted successfully.")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    finally:
        cursor.close()

# Function to insert reviewer data
def insert_reviewer_data(conn, data):
    cursor = conn.cursor()

    insert_query = """
        INSERT INTO dimreviewer (ReviewerID, ReviewerName, ReviewerClass,
BirthDate)
```

```
        VALUES (%s, %s, %s, %s)
        ON DUPLICATE KEY UPDATE
        ReviewerName = VALUES(ReviewerName),
        ReviewerClass = VALUES(ReviewerClass),
        BirthDate = VALUES(BirthDate);
    """

    try:
        for row in data:
            record = (row['ReviewerID'], row['ReviewerName'],
row['ReviewerClass'], row['BirthDate'])
            cursor.execute(insert_query, record)

        conn.commit()
        print("Reviewer data inserted successfully.")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    finally:
        cursor.close()

# Function to insert theater data
def insert_theater_data(conn, data):
    cursor = conn.cursor()

    insert_query = """
        INSERT INTO dimtheater (TheaterID, TheaterName, Location, NoScreens,
Seats)
        VALUES (%s, %s, %s, %s, %s)
        ON DUPLICATE KEY UPDATE
        TheaterName = VALUES(TheaterName),
        Location = VALUES(Location),
        NoScreens = VALUES(NoScreens),
        Seats = VALUES(Seats);
    """

    try:
        for row in data:
            record = (row['TheaterID'], row['TheaterName'], row['Location'],
row['NoScreens'], row['Seats'])
            cursor.execute(insert_query, record)

        conn.commit()
        print("Theater data inserted successfully.")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
```

```
    finally:
        cursor.close()

# Function to insert movie data
def insert_movie_data(conn, data):
    cursor = conn.cursor()

    insert_query = """
        INSERT INTO factmovie (MovieID, Title, DirectorID, DistributorID,
ReleaseDate, ProductionBudget, MovieRating, Producer, Genre)
        VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)
        ON DUPLICATE KEY UPDATE
        Title = VALUES(Title),
        DirectorID = VALUES(DirectorID),
        DistributorID = VALUES(DistributorID),
        ReleaseDate = VALUES(ReleaseDate),
        ProductionBudget = VALUES(ProductionBudget),
        MovieRating = VALUES(MovieRating),
        Producer = VALUES(Producer),
        Genre = VALUES(Genre);
    """

    try:
        for row in data:
            record = (row['MovieID'], row['Title'], row['DirectorID'],
row['DistributorID'], row['ReleaseDate'],row['ProductionBudget'],
row['MovieRating'], row['Producer'], row['Genre'] )
            cursor.execute(insert_query, record)

        conn.commit()
        print("Movie data inserted successfully.")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    finally:
        cursor.close()

# Function to insert moviecast data
def insert_moviecast_data(conn, data):
    cursor = conn.cursor()

    insert_query = """INSERT INTO factmoviecast (MovieID, ActorID, Salary)
        VALUES (%s, %s, %s)
        ON DUPLICATE KEY UPDATE
        Salary = VALUES(Salary);
    """
```

```
    try:
        for row in data:
            record = (row['MovieID'], row['ActorID'], row['Salary'])
            cursor.execute(insert_query, record)

        conn.commit()
        print("Movie Cast data inserted successfully.")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    finally:
        cursor.close()


# Function to insert moviereview data
def insert_moviereview_data(conn, data):
    cursor = conn.cursor()

    insert_query = """
        INSERT INTO factmoviereview (MovieID, ReviewerID, ReviewRating)
        VALUES (%s, %s, %s)
        ON DUPLICATE KEY UPDATE
        ReviewRating = VALUES(ReviewRating);
    """

    try:
        for row in data:
            record = (row['MovieID'], row['ReviewerID'], row['ReviewRating'])
            cursor.execute(insert_query, record)

        conn.commit()
        print("Movie Review data inserted successfully.")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    finally:
        cursor.close()

# Function to insert showing data
def insert_showing_data(conn, data):
    cursor = conn.cursor()

    insert_query = """
        INSERT INTO factshowing (MovieID, TheaterID, StartDate, EndDate,
BoxOffice)
        VALUES (%s, %s, %s, %s, %s)
```

```
        ON DUPLICATE KEY UPDATE
        StartDate = VALUES(StartDate),
        EndDate = VALUES(EndDate),
        BoxOffice = VALUES(BoxOffice);
    """

    try:
        for row in data:
            record = (row['MovieID'], row['TheaterID'], row['StartDate'],
row['EndDate'], row['BoxOffice'])
            cursor.execute(insert_query, record)

        conn.commit()
        print("Showing data inserted successfully.")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    finally:
        cursor.close()

def get_row_count(conn, table):
    cursor = conn.cursor()
    query = f"SELECT COUNT(*) FROM {table};"
    cursor.execute(query)
    row_count = cursor.fetchone()[0]
    cursor.close()
    return row_count

def compare_row_counts(conn_source, conn_target, source_tables, target_tables):
    print('-------------------------------------')
    print('For Functional Testing')
    print('-------------------------------------')
    for source_table, target_table in zip(source_tables, target_tables):
        source_count = get_row_count(conn_source, source_table)
        target_count = get_row_count(conn_target, target_table)
        if source_count != target_count:
            print(f"Row counts for table {source_table} and {target_table}
differ:")
            print(f"Source count: {source_count}")
            print(f"Target count: {target_count}")
        else:
            print(f"Row counts for table {source_table} and {target_table} are
equal: {source_count}")
# Main function to handle ETL process
def main():
    try:
```

```
        # Establishing connection
        source_conn = mysql.connector.connect(
            host='localhost',
            user='root',
            password='password',
            database='moviedb'
        )

        destination_conn = mysql.connector.connect(
            host='localhost',
            user='root',
            password='password',
            database='movieoptv3'
        )

        # Fetch and transform data from source
        agent_data = fetch_and_transform_agent_data(source_conn)
        director_data = fetch_and_transform_director_data(source_conn)
        actor_data = fetch_and_transform_actor_data(source_conn)
        distributor_data = fetch_and_transform_distributor_data(source_conn)
        movie_data = fetch_and_transform_movie_data(source_conn)
        moviecast_data = fetch_and_transform_moviecast_data(source_conn)
        moviereview_data = fetch_and_transform_moviereview_data(source_conn)
        reviewer_data = fetch_and_transform_reviewer_data(source_conn)
        showing_data = fetch_and_transform_showing_data(source_conn)
        theater_data = fetch_and_transform_theater_data(source_conn)

        # Insert data into destination
        print('-------------------------------------')
        print('Inserting')
        print('-------------------------------------')
        insert_agent_data(destination_conn, agent_data)
        insert_director_data(destination_conn, director_data)
        insert_actor_data(destination_conn, actor_data)
        insert_distributor_data(destination_conn, distributor_data)
        insert_movie_data(destination_conn, movie_data)
        insert_moviecast_data(destination_conn, moviecast_data)
        insert_moviereview_data(destination_conn, moviereview_data)
        insert_reviewer_data(destination_conn, reviewer_data)
        insert_showing_data(destination_conn, showing_data)
        insert_theater_data(destination_conn, theater_data)

        source_tables = ['agent', 'director', 'actor', 'distributor', 'reviewer',
'theater', 'movie', 'moviecast', 'moviereviews', 'showing']
```

```
        target_tables = ['dimagent', 'dimdirector', 'dimactor', 'dimdistributor',
'dimreviewer', 'dimtheater', 'factmovie', 'factmoviecast', 'factmoviereview',
'factshowing']

        compare_row_counts(source_conn, destination_conn, source_tables,
target_tables)

    except mysql.connector.Error as err:
        print(f"Error: {err}")
    finally:
        # Closing connections
        if source_conn.is_connected():
            source_conn.close()
        if destination_conn.is_connected():
            destination_conn.close()

# Execute the main function
if __name__ == "__main__":
    main()
```