



# Fire Up Your Base

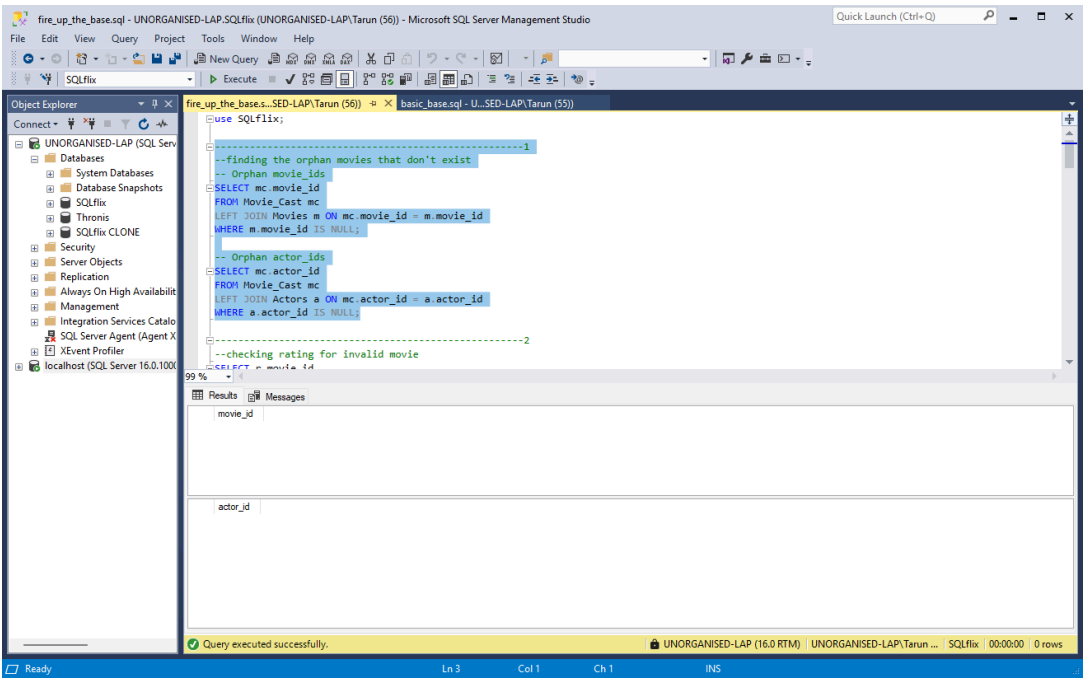


Refer the SQLflix SSIS Integration before performing the warm up queries in the base.

## Movie to Actor Relation

```
-----1
--finding the orphan movies that don't exist
-- Orphan movie_ids
SELECT mc.movie_id
FROM Movie_Cast mc
LEFT JOIN Movies m ON mc.movie_id = m.movie_id
WHERE m.movie_id IS NULL;

-- Orphan actor_ids
SELECT mc.actor_id
FROM Movie_Cast mc
LEFT JOIN Actors a ON mc.actor_id = a.actor_id
WHERE a.actor_id IS NULL;
```

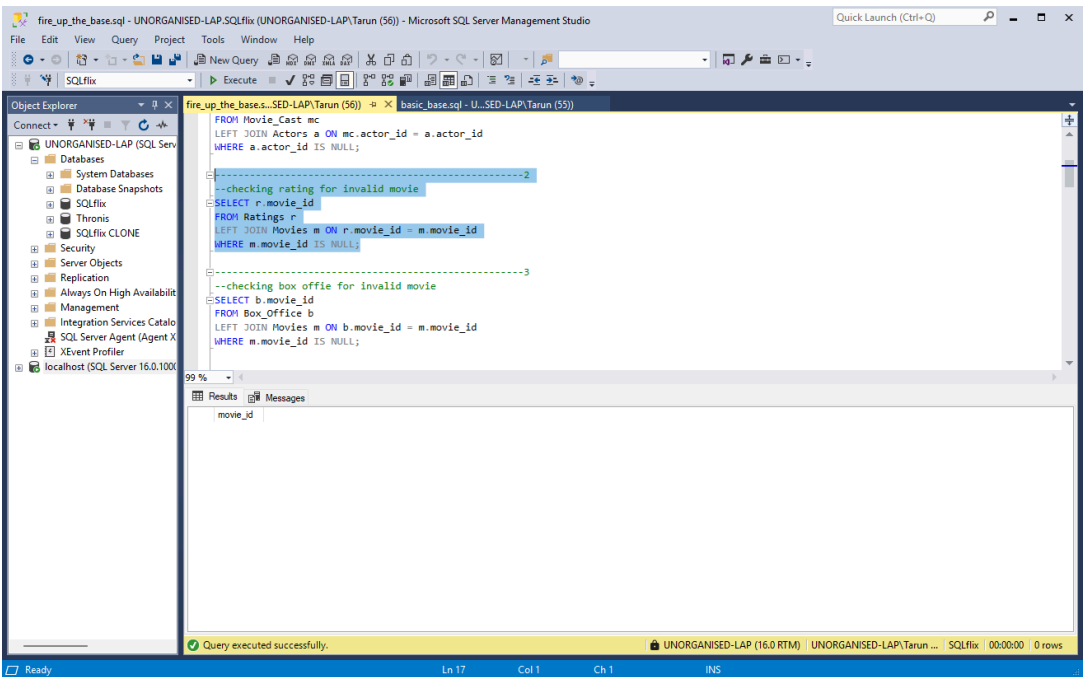


## Output

NO RECORDS

## Invalid Rating

```
-----2
--checking rating for invalid movie
SELECT r.movie_id
FROM Ratings r
LEFT JOIN Movies m ON r.movie_id = m.movie_id
WHERE m.movie_id IS NULL;
```

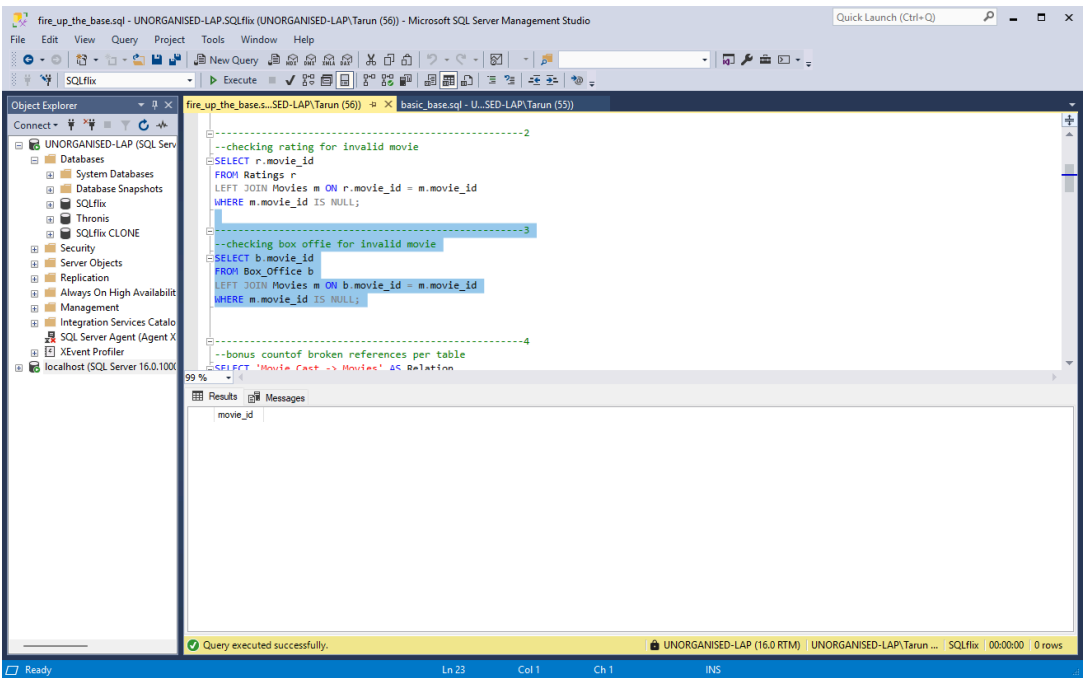


## Output

NO RECORDS

### Invalid box office revenue

```
-----3
--checking box offie for invalid movie
SELECT b.movie_id
FROM Box_Office b
LEFT JOIN Movies m ON b.movie_id = m.movie_id
WHERE m.movie_id IS NULL;
```



## Output

NO RECORDS

### Count broken references

```
-----4
--bonus countof broken references per table
SELECT 'Movie_Cast → Movies' AS Relation,
      COUNT(*) AS BrokenLinks
FROM Movie_Cast mc
LEFT JOIN Movies m ON mc.movie_id = m.movie_id
WHERE m.movie_id IS NULL

UNION ALL

SELECT 'Movie_Cast → Actors', COUNT(*)
FROM Movie_Cast mc
```

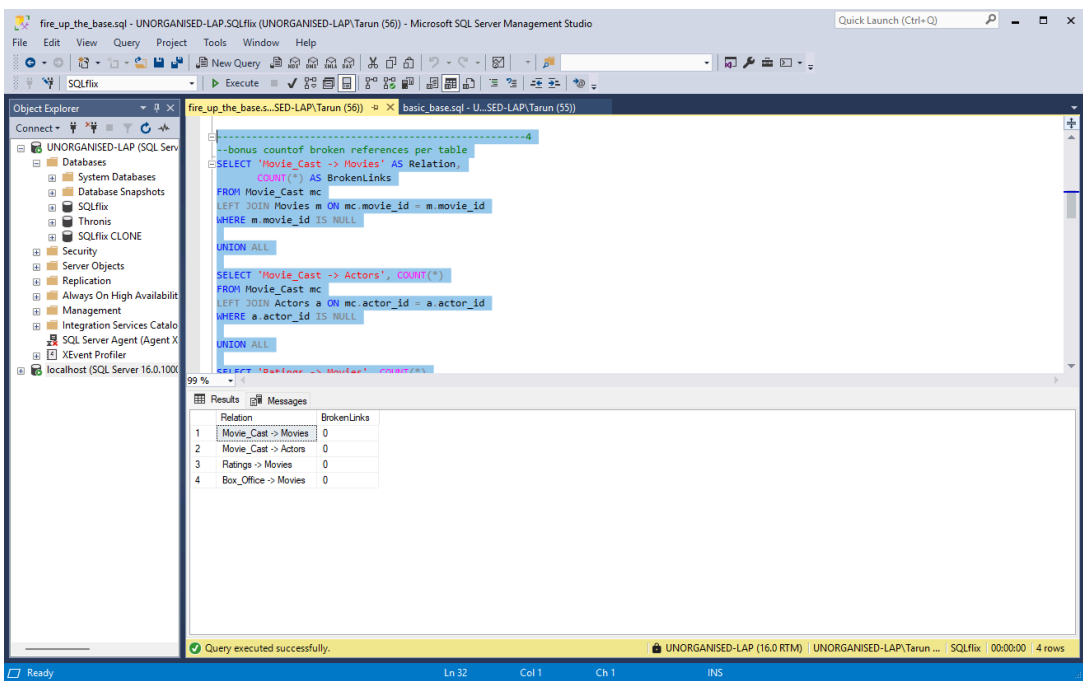
```
LEFT JOIN Actors a ON mc.actor_id = a.actor_id
WHERE a.actor_id IS NULL
```

UNION ALL

```
SELECT 'Ratings → Movies', COUNT(*)
FROM Ratings r
LEFT JOIN Movies m ON r.movie_id = m.movie_id
WHERE m.movie_id IS NULL
```

UNION ALL

```
SELECT 'Box_Office → Movies', COUNT(*)
FROM Box_Office b
LEFT JOIN Movies m ON b.movie_id = m.movie_id
WHERE m.movie_id IS NULL;
```



## Output

All the columns  
returning zeros.

## PK FK Relation

-----5

--duplicate PK check

-- Movies PK check

```
SELECT movie_id, COUNT(*) AS occurrences
FROM Movies
GROUP BY movie_id
HAVING COUNT(*) > 1;
```

-- Actors PK check

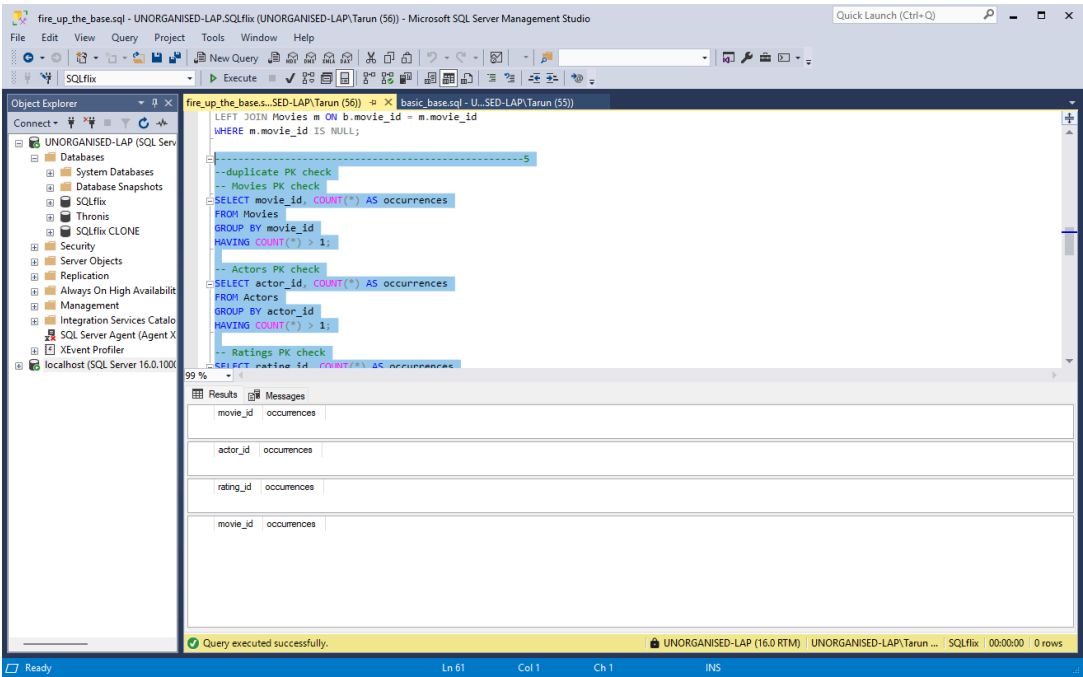
```
SELECT actor_id, COUNT(*) AS occurrences
FROM Actors
GROUP BY actor_id
HAVING COUNT(*) > 1;
```

-- Ratings PK check

```
SELECT rating_id, COUNT(*) AS occurrences
FROM Ratings
GROUP BY rating_id
HAVING COUNT(*) > 1;
```

-- Box\_Office PK check

```
SELECT movie_id, COUNT(*) AS occurrences
FROM Box_Office
GROUP BY movie_id
HAVING COUNT(*) > 1;
```



Output

NO RECORDS

## Count the number of records

```
-----6
SELECT 'Actors' AS "Table", COUNT(*) AS Records
FROM Actors

UNION ALL

SELECT 'Box_Office', COUNT(*)
FROM Box_Office

UNION ALL

SELECT 'Movie_Cast', COUNT(*)
FROM Movie_Cast

UNION ALL

SELECT 'Movies', COUNT(*)
FROM Movies

UNION ALL

SELECT 'Ratings', COUNT(*)
FROM Ratings;
```

Query executed successfully.

Table	Records
Actors	1000
Box_Office	5000
Movie_Cast	9830
Movies	5000
Ratings	138406

## Output

Number of records  
in each table.  
NOT NULL.

## Schema Information

```
-----7
SELECT TABLE_NAME, COLUMN_NAME, DATA_TYPE, IS_NULLABLE, CHARACTER_MAXIMUM_LENGTH
FROM INFORMATION_SCHEMA.COLUMNS
ORDER BY TABLE_NAME, ORDINAL_POSITION;
```

Query executed successfully.

TABLE_NAME	COLUMN_NAME	DATA_TYPE	IS_NULLABLE	CHARACTER_MAXIMUM_LENGTH
Actors	actor_id	int	NO	NULL
Actors	name	nvarchar	NO	255
Actors	birth_year	int	YES	NULL
Actors	nationality	nvarchar	YES	100
Box_Office	movie_id	int	NO	NULL
Box_Office	budget_million	decimal	YES	NULL
Box_Office	revenue_million	decimal	YES	NULL
Movie_Cast	movie_id	int	NO	NULL
Movie_Cast	actor_id	int	NO	NULL
Movie_Cast	role_name	nvarchar	YES	255
Movies	movie_id	int	NO	NULL
Movies	title	nvarchar	NO	255
Movies	release_year	int	YES	NULL

## Output

The Schema  
Information.