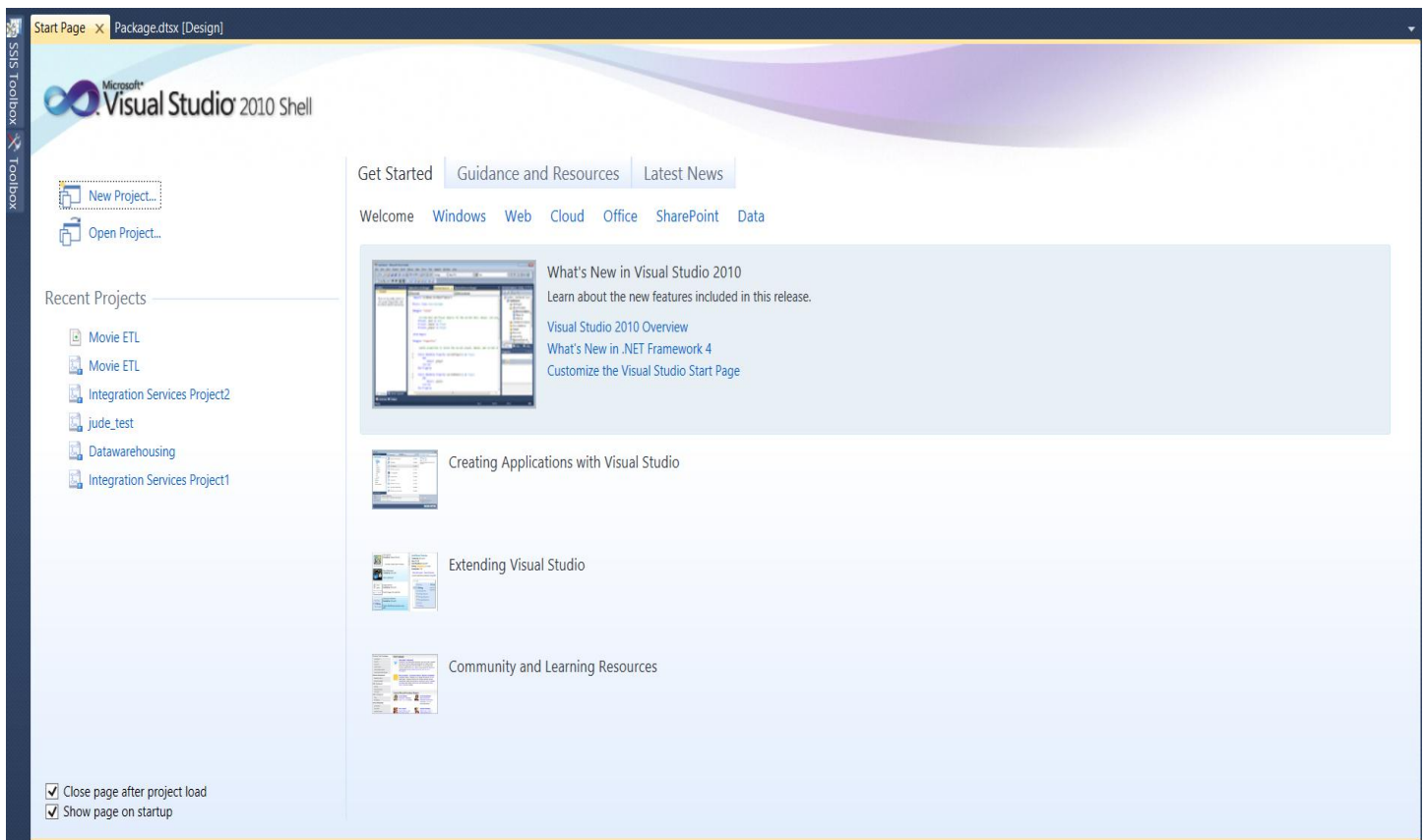


# “Movie Management System”

## OLAP of the Application

---

*Mohamed Niyaz*



## 1. OLAP Introduction

In computing, online analytical processing, or OLAP an approach to answering multi-dimensional analytical (MDA) queries swiftly. OLAP is part of the broader category of business intelligence, which also encompasses relational database report writing and data mining. Typical applications of OLAP include business reporting for sales, marketing, management reporting, business process management budgeting and forecasting, financial reporting and similar areas, with new applications coming up, such as agriculture. The term OLAP was created as a slight modification of the traditional database term OLTP (Online Transaction Processing).

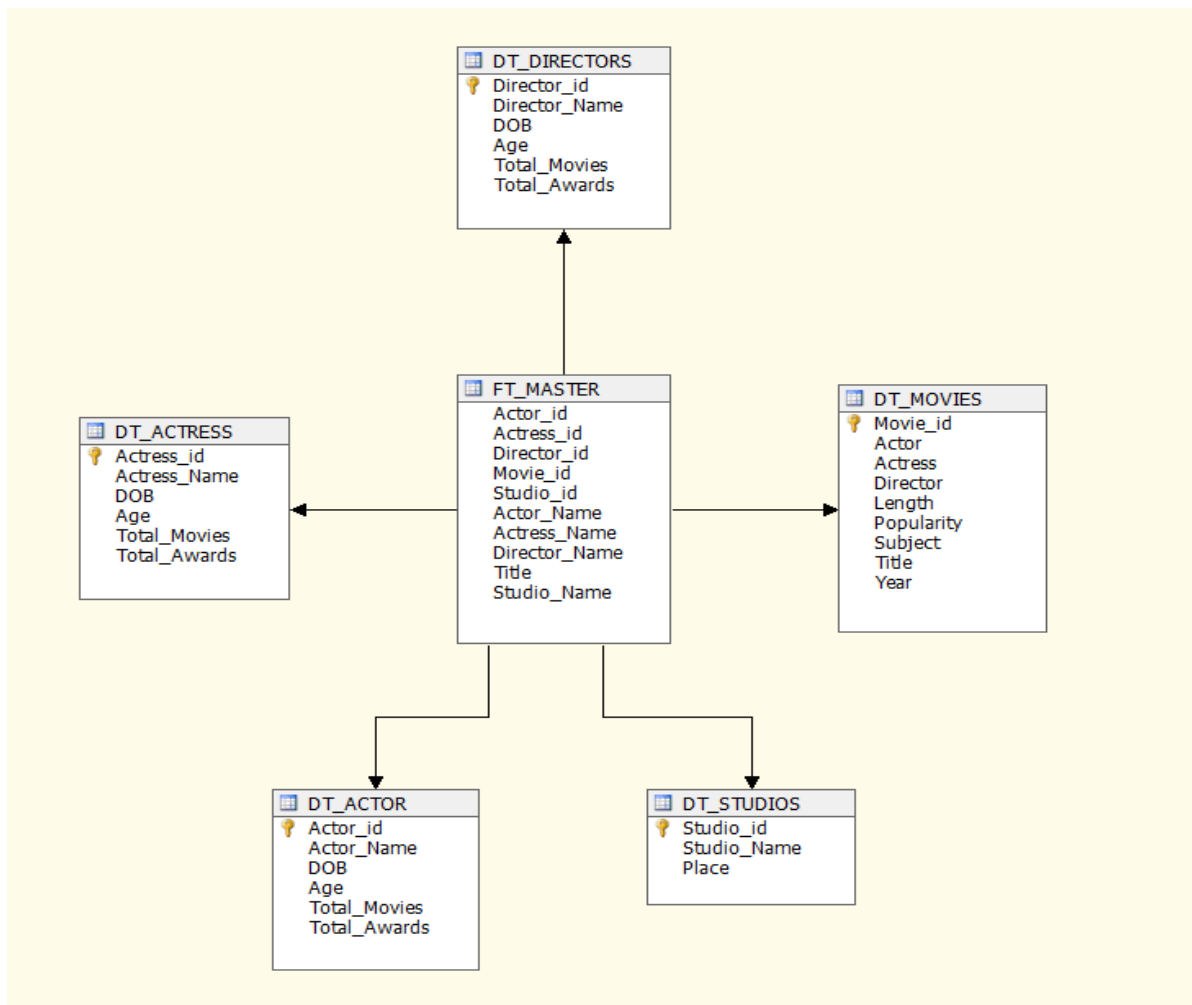
OLAP tools enable users to analyze multidimensional data interactively from multiple perspectives. OLAP consists of three basic analytical operations: consolidation (roll-up), drill-down, and slicing and dicing. Consolidation involves the aggregation of data that can be accumulated and computed in one or more dimensions. Databases configured for OLAP use a multidimensional data model, allowing for complex analytical and ad-hoc queries with a rapid execution time. They borrow aspects of navigational databases, hierarchical databases and relational databases.

## 2. OLAP Design



Our OLAP Process is carried through **Microsoft BI** tool and below screen shot explains the high level overview of the **Schema**

The below Screenshot depicts **data flow task** which is part of ETL process



### 3. OLAP Process

- **Roll-up** : A roll-up involves summarizing the data along a dimension. The summarization rule might be computing totals along a hierarchy or applying a set of formulas such as movie rating = Total Rating/Number of Votes.
- **Drill-Down** : Drill Down allows the user to navigate among levels of data ranging from the most summarized (up) to the most detailed (down).
- **Slicing** : Slice is the act of picking a rectangular subset of a cube by choosing a single value for one of its dimensions, creating a new cube with one fewer dimension. Finding out the movie that released in the year of 2004 are "sliced" out of the data cube.
- **Dicing** : The dice operation produces a sub cube by allowing the analyst to pick specific values of multiple dimensions. The new cube will show limited number of movies, the time and region dimensions cover the same range as before.

## 4. Table Hierarchies

- Actor table : Actor id < Actor Name < DOB < Age
- Actress table : Actress id < Actress Name < DOB < Age
- Director table : Director id < Director Name < DOB < Age
- Movies table : Movie id < Title < Genre < Popularity < Year
- Studio table : Studio id < Studio Name < Place
- Master table : Actor Name < Actress Name < Director Name < Title < Studio Name

## 5. OLAP Operations (Includes Screenshot and Sql implementation)

Consider a base cuboid Master table and Movies table [Master, Movies]

**Step1:** Create a Base cuboid with Year and Subject = [Year, Subject]

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Movies' folder is expanded, showing a hierarchy of tables: System Tables, FileTables, and various data tables (dbo.DT\_ACTOR, dbo.DT\_ACTRESS, dbo.DT\_DIRECTORS, dbo.DT\_MOVIES, dbo.DT\_STUDIOS, dbo.FT\_MASTER, dbo.STG\_ACTOR, dbo.STG\_ACTRESS, dbo.STG\_DIRECTORS, dbo.STG\_MOVIES, dbo.STG\_STUDIOS). The 'Results' pane on the right displays the output of the following query:

```
select b.Year,b.Subject from dbo.FT_MASTER a , dbo.DT_MOVIES b where a.Movie_id = b.Movie_id order by b.Year asc;
```

	Year	Subject
1	1920	Drama
2	1924	Drama
3	1925	Drama
4	1926	Action
5	1926	Mystery
6	1926	Science Fiction
7	1927	Comedy
8	1927	Drama
9	1927	Mystery
10	1928	Drama
11	1928	War

**Sql implementation:**

```
select b.Year,b.Subject from dbo.FT_MASTER a , dbo.DT_MOVIES b where a.Movie_id = b.Movie_id order by b.Year asc;
```

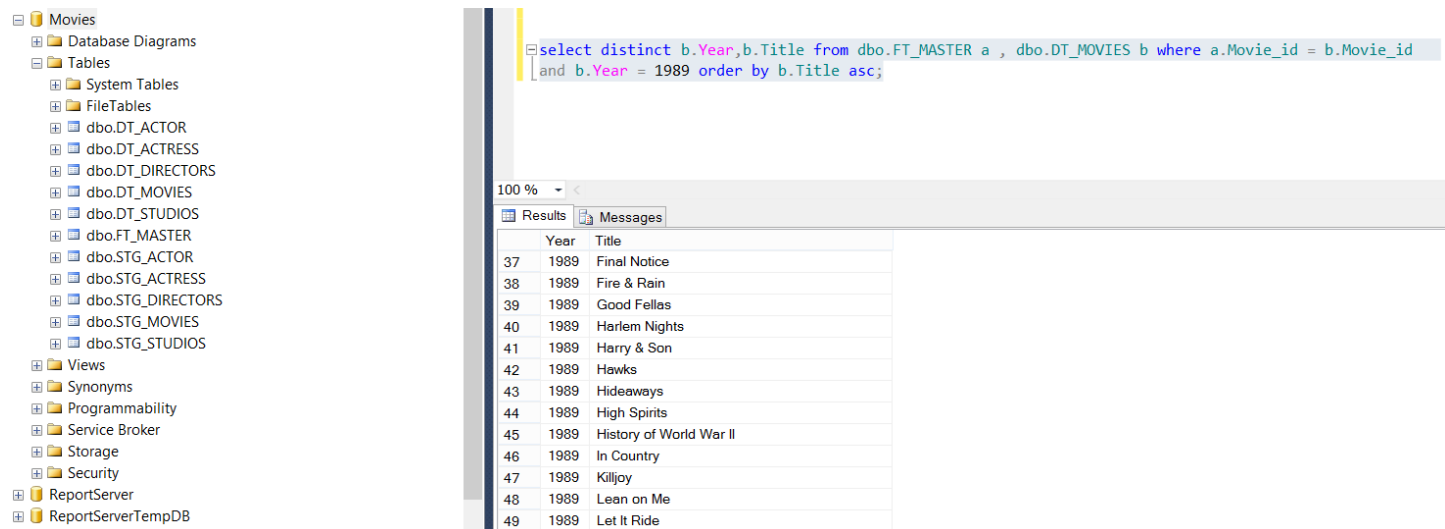
**Step2:** Slice on the Year by 1989

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Movies' folder is expanded, showing the same hierarchy of tables as in the previous screenshot. The 'Results' pane on the right displays the output of the following query:

```
select distinct b.Year,b.Subject from dbo.FT_MASTER a , dbo.DT_MOVIES b where a.Movie_id = b.Movie_id and b.Year = 1989 order by b.Subject asc;
```

	Year	Subject
1	1989	Action
2	1989	Comedy
3	1989	Drama
4	1989	Horror
5	1989	Music
6	1989	Mystery
7	1989	Science Fiction
8	1989	War
9	1989	Westerns

```
select distinct b.Year,b.Subject from dbo.FT_MASTER a , dbo.DT_MOVIES b where a.Movie_id = b.Movie_id and b.Year = 1989 order by b.Subject asc;
```

**Step3:** Rollup on the Subject to obtain the list of titles for the year 1989


The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Movies' folder is expanded, showing a hierarchy of tables including 'dbo.DT\_ACTOR', 'dbo.DT\_ACTRESS', 'dbo.DT\_DIRECTORS', 'dbo.DT\_MOVIES', 'dbo.DT\_STUDIOS', 'dbo.FT\_MASTER', 'dbo.STG\_ACTOR', 'dbo.STG\_ACTRESS', 'dbo.STG\_DIRECTORS', 'dbo.STG\_MOVIES', and 'dbo.STG\_STUDIOS'. The main window displays a SQL query:

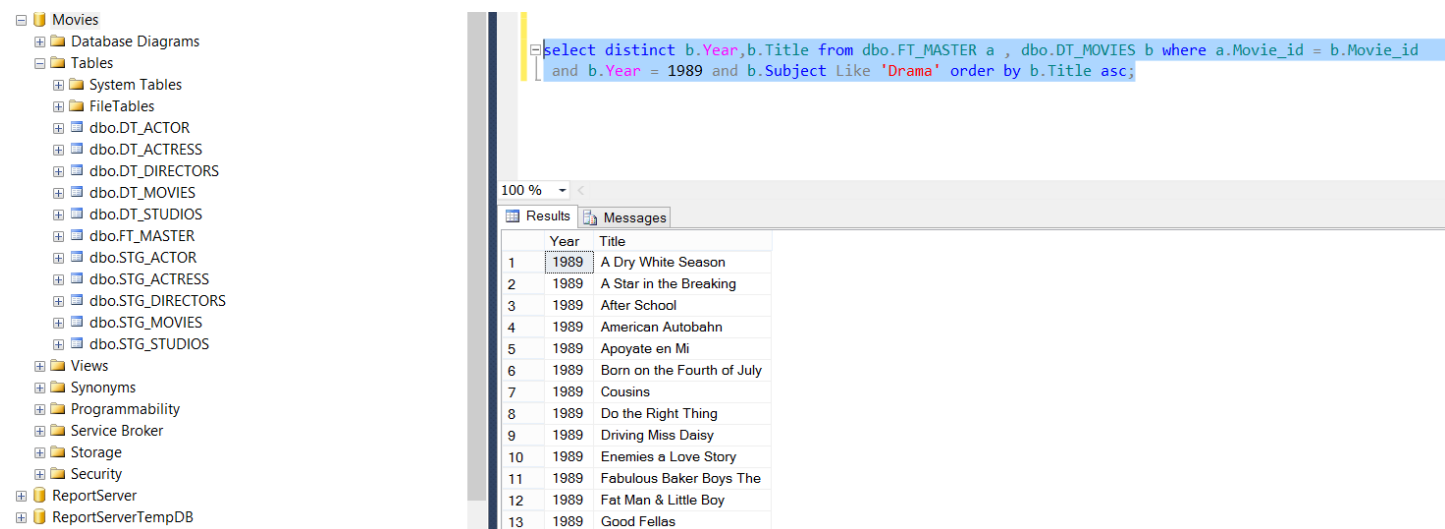
```
select distinct b.Year,b.Title from dbo.FT_MASTER a , dbo.DT_MOVIES b where a.Movie_id = b.Movie_id
and b.Year = 1989 order by b.Title asc;
```

The 'Results' tab shows the following data:

	Year	Title
37	1989	Final Notice
38	1989	Fire & Rain
39	1989	Good Fellas
40	1989	Harlem Nights
41	1989	Harry & Son
42	1989	Hawks
43	1989	Hideaways
44	1989	High Spirits
45	1989	History of World War II
46	1989	In Country
47	1989	Killjoy
48	1989	Lean on Me
49	1989	Let It Ride

**Sql implementation:**

```
select distinct b.Year,b.Title from dbo.FT_MASTER a , dbo.DT_MOVIES b where a.Movie_id = b.Movie_id
and b.Year = 1989 order by b.Title asc;
```

**Step4:** Slice on the Subject to Drama to obtain the list of movies for 1989 with genre Drama


The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Movies' folder is expanded, showing a hierarchy of tables including 'dbo.DT\_ACTOR', 'dbo.DT\_ACTRESS', 'dbo.DT\_DIRECTORS', 'dbo.DT\_MOVIES', 'dbo.DT\_STUDIOS', 'dbo.FT\_MASTER', 'dbo.STG\_ACTOR', 'dbo.STG\_ACTRESS', 'dbo.STG\_DIRECTORS', 'dbo.STG\_MOVIES', and 'dbo.STG\_STUDIOS'. The main window displays a SQL query:

```
select distinct b.Year,b.Title from dbo.FT_MASTER a , dbo.DT_MOVIES b where a.Movie_id = b.Movie_id
and b.Year = 1989 and b.Subject Like 'Drama' order by b.Title asc;
```

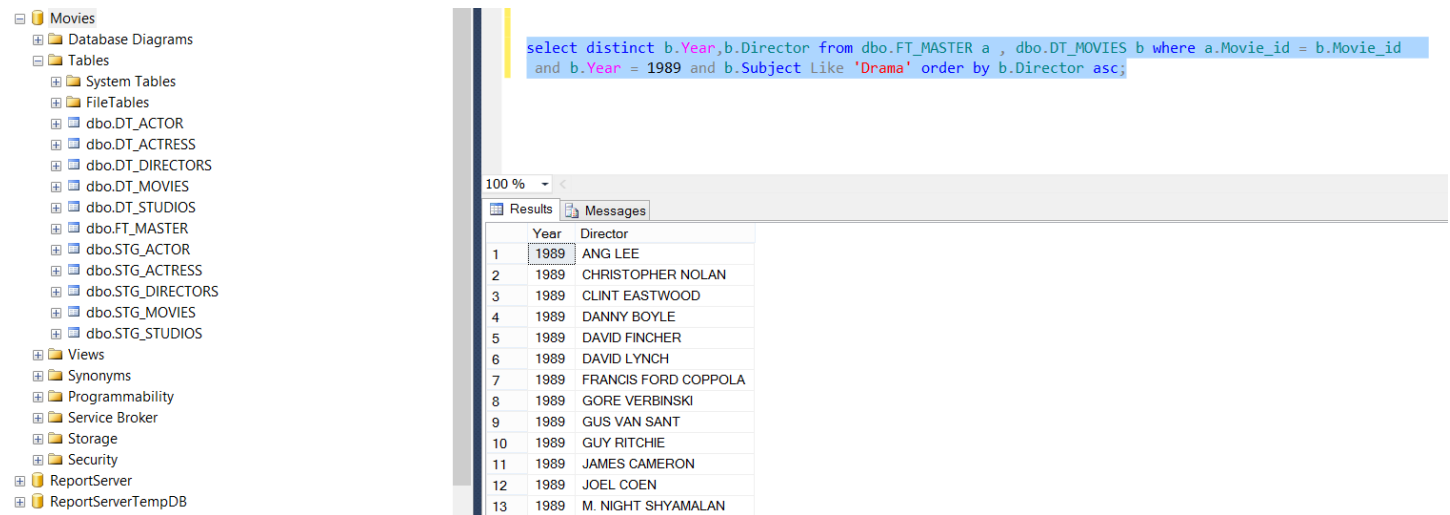
The 'Results' tab shows the following data:

	Year	Title
1	1989	A Dry White Season
2	1989	A Star in the Breaking
3	1989	After School
4	1989	American Autobahn
5	1989	Apayate en Mi
6	1989	Born on the Fourth of July
7	1989	Cousins
8	1989	Do the Right Thing
9	1989	Driving Miss Daisy
10	1989	Enemies a Love Story
11	1989	Fabulous Baker Boys The
12	1989	Fat Man & Little Boy
13	1989	Good Fellas

**Sql implementation:**

```
select distinct b.Year,b.Title from dbo.FT_MASTER a , dbo.DT_MOVIES b where a.Movie_id = b.Movie_id
and b.Year = 1989 and b.Subject Like 'Drama' order by b.Title asc;
```

**Step5:** Rollup on the Title to obtain list of director who has directed in the year 1989 with genre Drama



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Movies' database is expanded, showing various tables and views. The central pane displays a SQL query:

```
select distinct b.Year,b.Director from dbo.FT_MASTER a , dbo.DT_MOVIES b where a.Movie_id = b.Movie_id
and b.Year = 1989 and b.Subject Like 'Drama' order by b.Director asc;
```

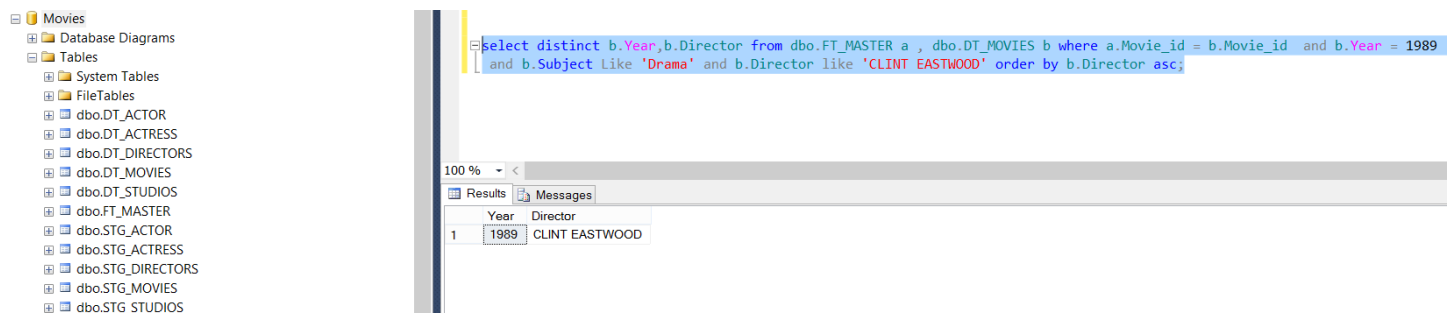
Below the query, the 'Results' tab shows the output of the query:

	Year	Director
1	1989	ANG LEE
2	1989	CHRISTOPHER NOLAN
3	1989	CLINT EASTWOOD
4	1989	DANNY BOYLE
5	1989	DAVID FINCHER
6	1989	DAVID LYNCH
7	1989	FRANCIS FORD COPPOLA
8	1989	GORE VERBINSKI
9	1989	GUS VAN SANT
10	1989	GUY RITCHIE
11	1989	JAMES CAMERON
12	1989	JOEL COEN
13	1989	M. NIGHT SHYAMALAN

### Sql implementation:

```
select distinct b.Year,b.Director from dbo.FT_MASTER a , dbo.DT_MOVIES b where a.Movie_id = b.Movie_id
and b.Year = 1989 and b.Subject Like 'Drama' order by b.Director asc;
```

**Step6:** Slice on the director to Clint Eastwood to obtain the list of movies for 1989 with genre Drama



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Movies' database is expanded. The central pane displays a SQL query:

```
select distinct b.Year,b.Director from dbo.FT_MASTER a , dbo.DT_MOVIES b where a.Movie_id = b.Movie_id and b.Year = 1989
and b.Subject Like 'Drama' and b.Director like 'CLINT EASTWOOD' order by b.Director asc;
```

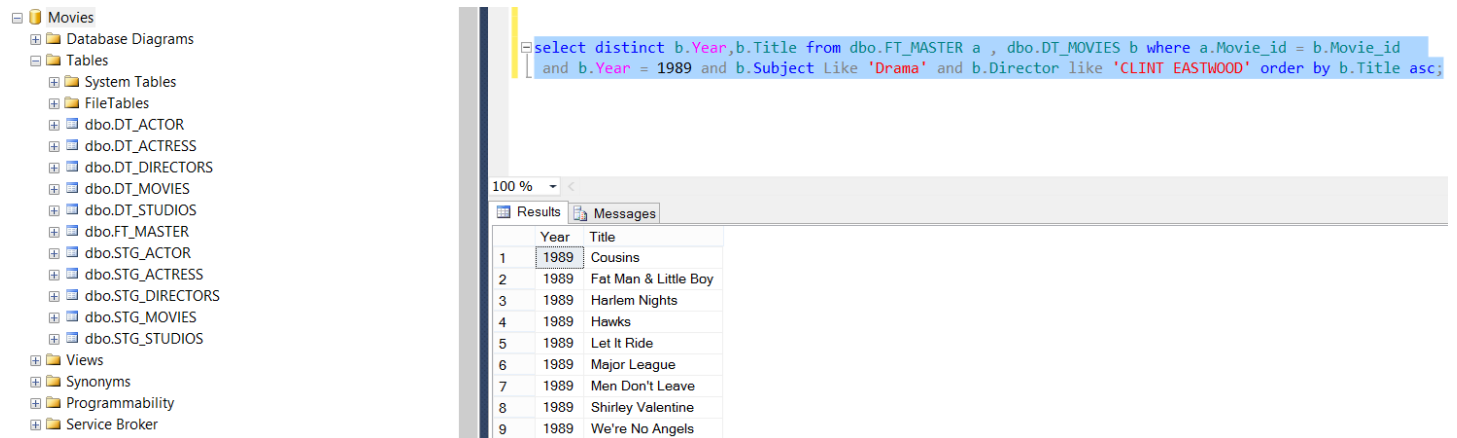
Below the query, the 'Results' tab shows the output of the query:

	Year	Director
1	1989	CLINT EASTWOOD

### Sql implementation:

```
select distinct b.Year,b.Director from dbo.FT_MASTER a , dbo.DT_MOVIES b where a.Movie_id = b.Movie_id
and b.Year = 1989 and b.Subject Like 'Drama' and b.Director like 'CLINT EASTWOOD' order by b.Director
asc;
```

**Step7:** Rollup on the title to obtain list of movies for 1989 with genre Drama directed by Clint Eastwood



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Movies' folder is expanded, showing a hierarchy of tables: System Tables, FileTables, and various tables in the 'dbo' schema including DT\_ACTOR, DT\_ACTRESS, DT\_DIRECTORS, DT\_MOVIES, DT\_STUDIOS, FT\_MASTER, STG\_ACTOR, STG\_ACTRESS, STG\_DIRECTORS, STG\_MOVIES, and STG\_STUDIOS. The main pane displays a SQL query:

```
select distinct b.Year,b.Title from dbo.FT_MASTER a , dbo.DT_MOVIES b where a.Movie_id = b.Movie_id
and b.Year = 1989 and b.Subject Like 'Drama' and b.Director like 'CLINT EASTWOOD' order by b.Title asc;
```

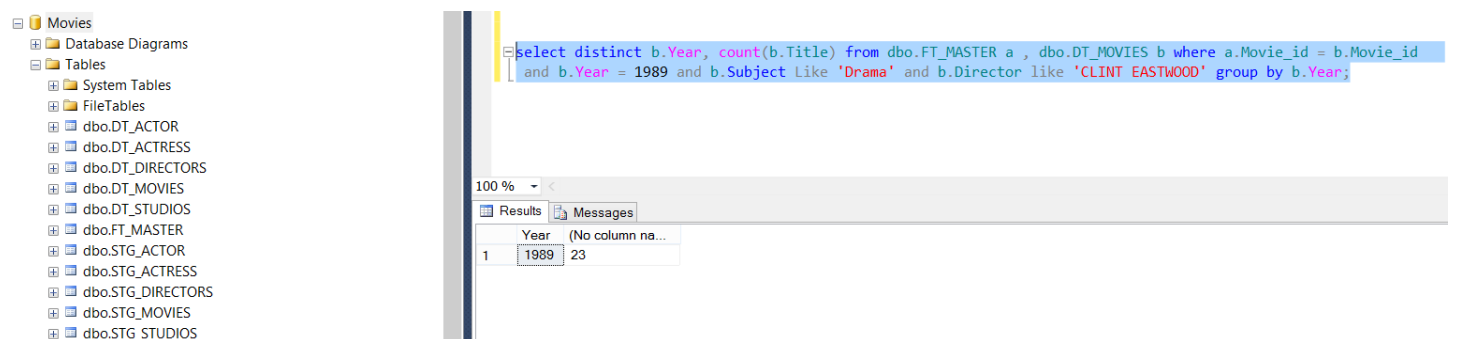
Below the query, the 'Results' tab shows the following data:

	Year	Title
1	1989	Cousins
2	1989	Fat Men & Little Boy
3	1989	Harlem Nights
4	1989	Hawks
5	1989	Let It Ride
6	1989	Major League
7	1989	Men Don't Leave
8	1989	Shirley Valentine
9	1989	We're No Angels

### Sql implementation:

```
select distinct b.Year,b.Title from dbo.FT_MASTER a , dbo.DT_MOVIES b where a.Movie_id = b.Movie_id
and b.Year = 1989 and b.Subject Like 'Drama' and b.Director like 'CLINT EASTWOOD' order by b.Title asc;
```

**Step8:** Dice on the title to obtain the number of movies for 1989 with genre Drama directed by Clint Eastwood



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Movies' folder is expanded, showing a hierarchy of tables: System Tables, FileTables, and various tables in the 'dbo' schema including DT\_ACTOR, DT\_ACTRESS, DT\_DIRECTORS, DT\_MOVIES, DT\_STUDIOS, FT\_MASTER, STG\_ACTOR, STG\_ACTRESS, STG\_DIRECTORS, STG\_MOVIES, and STG\_STUDIOS. The main pane displays a SQL query:

```
select distinct b.Year, count(b.Title) from dbo.FT_MASTER a , dbo.DT_MOVIES b where a.Movie_id = b.Movie_id
and b.Year = 1989 and b.Subject Like 'Drama' and b.Director like 'CLINT EASTWOOD' group by b.Year;
```

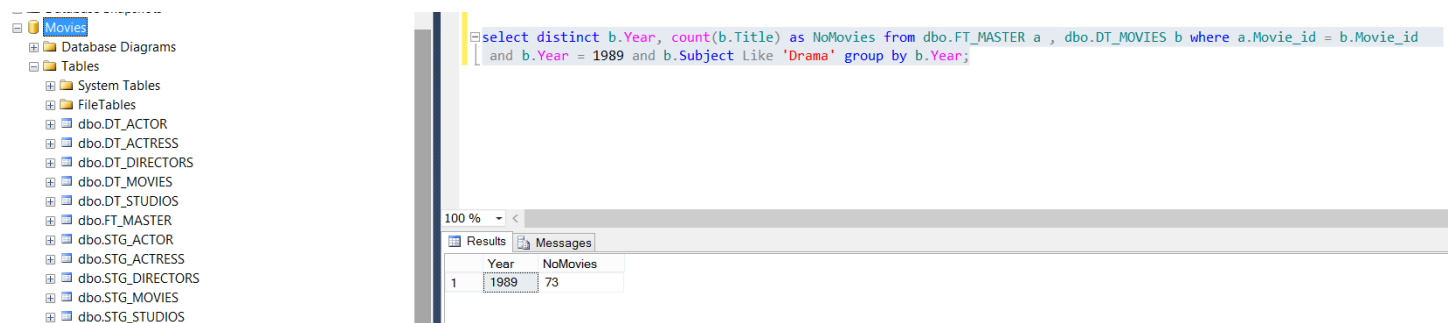
Below the query, the 'Results' tab shows the following data:

	Year	(No column na...
1	1989	23

### Sql implementation:

```
select distinct b.Year, count(b.Title) from dbo.FT_MASTER a , dbo.DT_MOVIES b where a.Movie_id =
b.Movie_id and b.Year = 1989 and b.Subject Like 'Drama' and b.Director like 'CLINT EASTWOOD' group by
b.Year;
```

**Step9:** Drill down on director to find the number of movies directed for the year 1989 with genre drama



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Movies' database is expanded, showing the 'Tables' folder. The 'dbo.DT\_DIRECTORS' table is selected. The main pane displays the following SQL query:

```
select distinct b.Year, count(b.Title) as NoMovies from dbo.FT_MASTER a , dbo.DT_MOVIES b where a.Movie_id = b.Movie_id  
and b.Year = 1989 and b.Subject Like 'Drama' group by b.Year;
```

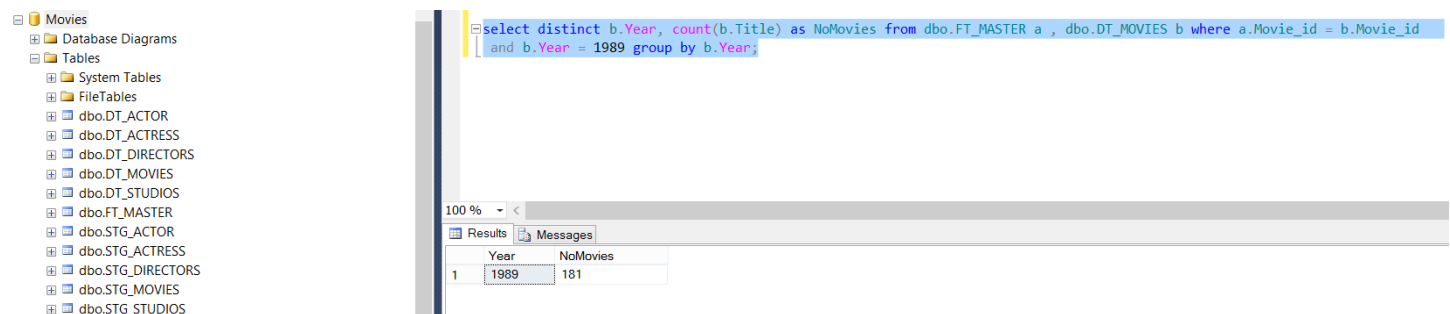
The 'Results' tab shows the following data:

Year	NoMovies
1 1989	73

### Sql implementation:

```
select distinct b.Year, count(b.Title) from dbo.FT_MASTER a , dbo.DT_MOVIES b where a.Movie_id =  
b.Movie_id and b.Year = 1989 and b.Subject Like 'Drama' group by b.Year;
```

**Step10:** Drill down on subject to find the number of movies directed for the year 1989



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Movies' database is expanded, showing the 'Tables' folder. The 'dbo.DT\_DIRECTORS' table is selected. The main pane displays the following SQL query:

```
select distinct b.Year, count(b.Title) as NoMovies from dbo.FT_MASTER a , dbo.DT_MOVIES b where a.Movie_id = b.Movie_id  
and b.Year = 1989 group by b.Year;
```

The 'Results' tab shows the following data:

Year	NoMovies
1 1989	181

### Sql implementation:

```
select distinct b.Year, count(b.Title) as NoMovies from dbo.FT_MASTER a , dbo.DT_MOVIES b where  
a.Movie_id = b.Movie_id and b.Year = 1989 group by b.Year;
```



**Step11:** Drill down on Year to find the number of movies directed so far.

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Database Diagrams' pane is expanded, showing a hierarchy of tables including 'dbo.DT\_ACTOR', 'dbo.DT\_ACTRESS', 'dbo.DT\_DIRECTORS', 'dbo.DT\_MOVIES', 'dbo.DT\_STUDIOS', 'dbo.FT\_MASTER', 'dbo.FT\_ACTOR', 'dbo.STG\_ACTRESS', 'dbo.STG\_DIRECTORS', 'dbo.STG\_MOVIES', and 'dbo.STG\_STUDIOS'. The main window displays a SQL query in the 'Query Editor' pane:

```
select distinct b.Year, count(b.Title) as NoMovies from dbo.FT_MASTER a , dbo.DT_MOVIES b where a.Movie_id = b.Movie_id
group by b.Year;
```

Below the query editor, the 'Results' pane shows the output of the query as a table with two columns: 'Year' and 'NoMovies'. The results are as follows:

	Year	NoMovies
1	1920	2
2	1924	3
3	1925	1
4	1926	5
5	1927	7
6	1928	6
7	1929	6
8	1930	2
9	1931	10
10	1932	11
11	1933	4

### Sql implementation:

```
select distinct b.Year, count(b.Title) as NoMovies from dbo.FT_MASTER a , dbo.DT_MOVIES b where
a.Movie_id = b.Movie_id group by b.Year;
```

## 6. Tools Used

- **Microsoft SQL Enterprise Edition**- The tool has inbuilt SQL server, Repository service, Integration Service, Analytical Services , SQL Database Management Studio. It's a tool which has both the ETL and the OLAP functionality with it.
- **Pl-Sql** – This is an Oracle commands which will be used to provide a flexible database activity in order to promote best achievements in obtaining an elegant working system.
- **SSIS** – Used for ETL (Extract transform Load)
- **SSAS** – Used for OLAP(Roll up , drill down , slicing, dicing , cube creation)
- **Hardware's**- Windows 8 64 bit operating , 16 GB DDR3 Ram, i7 Intel Processor, 2 GB DDR5

## 7. Challenges

- OLAP operations can be performed only the cleanse data, so getting the cleanse data was one among the challenges.
- OLAP operations have to be performed on fact and dimension tables, forming multiple relations and hierarchy was really challenging.
- Forming the cuboid was the biggest challenge among all these, we have to make sure the entity relation is perfectly right and data are fully cleansed.

No refinement to the warehouse design.