SAKILA DATA WAREHOUSE

By

Name: SWAPNIL SHEKHAR CHAUDHARI
Student No:10505342
Module: B9DA102 Data Storage Solutions for Data Analytics
BATCH C- DATA ANALYTICS
DUBLIN BUSINESS SCHOOL

CONTENTS

BACKGROUND FOR THE DEVELOPMENT OF CHOSEN SYSTEM

Data warehousing is An advance technique which aid in quick decision making for business growth. The idea of a data warehouse is to develop a permanent storage for the data needed to support reporting, analytical , and other Business Intelligence functions. It may seem improper to store data in multiple places (source systems and the data warehouse), but it provides many benefits of doing that.

Data warehouses dwells on servers used for running a database management system (DBMS) such as SQL Server and performing Extract, Transform, and Load (ETL) software such as SQL Server Integration Services (SSIS) to fetch data from the source systems and push into the data warehouse.

There are routine needs to provide reports, dashboards, analytic applications and ad-hoc queries inside any company, Hence Data Warehousing can be implemented to support decision making and analytical approaches.

PART 1:

Develop a proof of concept data warehouse with more than one fact table by capturing data from an existing data source(s). Document your reasons for selecting the subject area(s), identify key stakeholders, formalise the vision and goals and requirements for developing the data warehouse.

➢ 1.1 MENTION OF SOURCE DATABASE

Sakila Sample Database is used as a source database to develop a concept data warehouse.

➢ 1.2 ABOUT SAKILA DATABASE

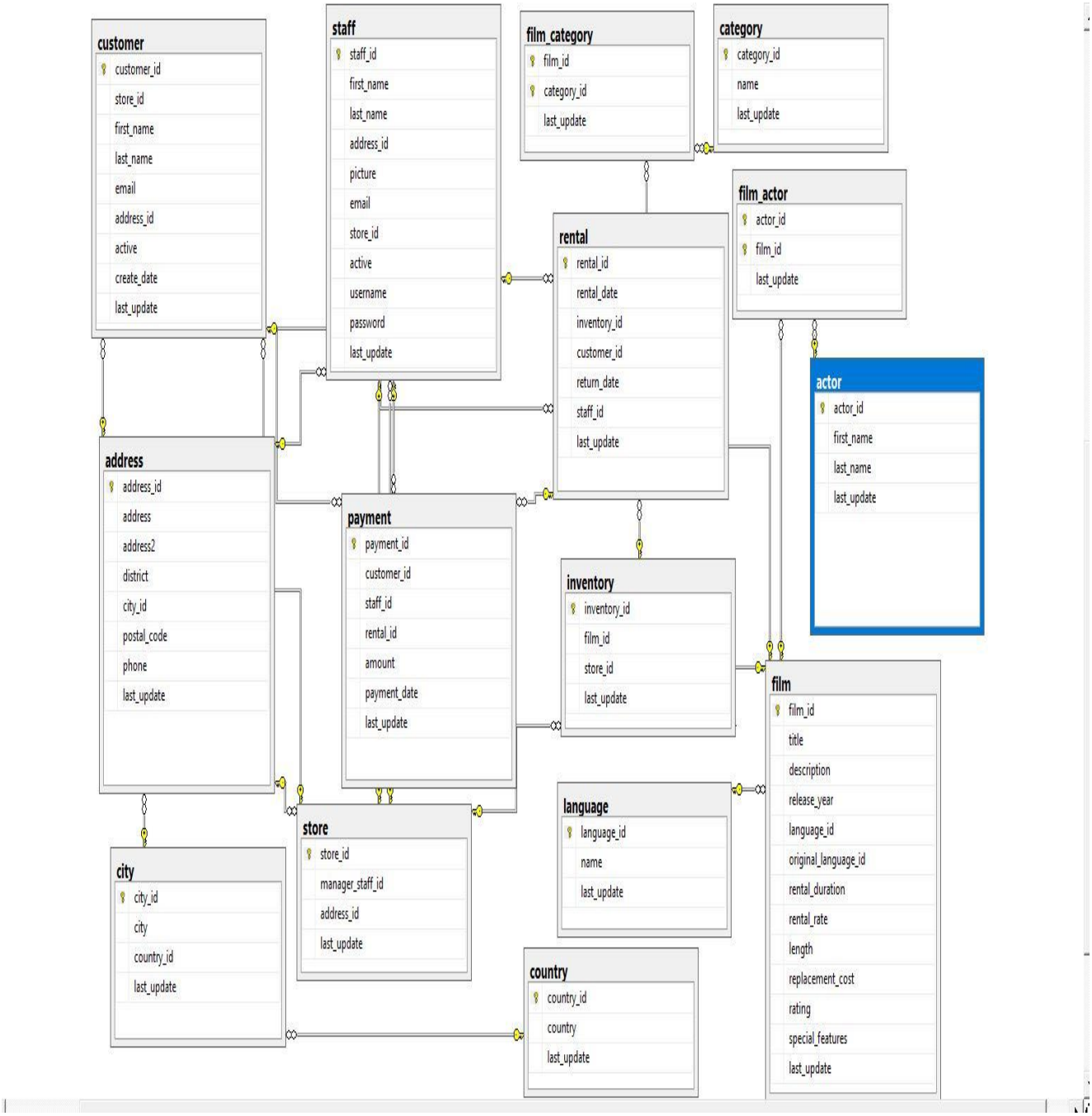
Sakila for Microsoft SQL Server is a part of the example of Sakila database from MySQL. Originally it was created by Mike Hillyer of the MySQL AB documentation team.

The Sakila sample database was developed and designed as a substitute to the world sample database, also provided by Oracle.

the Sakila sample database is designed to understand a DVD rental store.

➢ 1.3 SAKILA ENTITY-RELATIONSHIP MODEL

ER model below represents the relationship between various entities and relationship among them in the Sakila sample database.



➢ 1.4 ENTITIES AND ATTRIBUTES OF SAKILA

There are 15 entities in the Sakila database : city, country, language, store, film, address, payment, inventory, actor, rental, staff, customer, film category, category, film actor. All are used to create a suitable Data Warehouse.

➢ 1.5 SUBJECT AREAS OF ANALYSIS

Based on this ER diagram, Sakila database can be analyzed in following ways to improve store's business:

1. Available Total Movies By each Actor

2. Available Total DVDS Of each Actor in Inventory.

3.Earning By Date

4.Monthly Earning By Category

5.Number of Times DVDs OF each Categories Rented in different Years AND Relevant Total Rent Earnings for the same Period.

6. Available Total DVDS of each Actors in different Film Category.

7. Available Total Actor Movies By Each Category.

8. Number of DVDS for Each Movies

➢ 1.6 REASONS FOR SELECTING SUBJECT AREAS

Reasons for selecting the subject areas are as below:

1.Financial aspect:

- From subject areas we can get idea about in which direction stores revenue is performing. In other words growth or plunge.

2.Customer Preferences:

- Management can observe which movie dvds are more frequently rented to customers and which actors' dvds are the most famous and the least popular among customers across the globe. which kind of movie are most popular and the least popular among customers

3.Inventory Management:

- After knowing customer preferences , to satisfy customer need, management can increase or decrease the stock of dvds depending on popularity of film category, films and actors.

➢ 1.7 KEY STAKEHOLDERS

- **Customers** – Play the significant role in revenue of the store. May be internal or external who pays rents for dvds of their choice.
- **Employees -**They are part of work force of the organization.
- **Managers** – They are responsible for all the work of the employees who reports to them.
- **Management** – They are the part of staff who set the business vision and rules to be obeyed throughout the organization for growth.

➢ 1.8 VISION , GOALS AND REQUIREMENTS

➢ 1.8 VISION, GOALS AND REQUIREMENTS

Vision:
To produce effective analysis based on routine transactions to grow the business further .

Goals:

- Sakila contains huge amount of available data of routine transactions in the dvd store, observing a limited portion of data can lead the business to significant growth. Hence , for better understanding of business, the data and the whole organization is divided into several parts . Here Sakila database has a lot of dimensions , but not all drives the business growth.

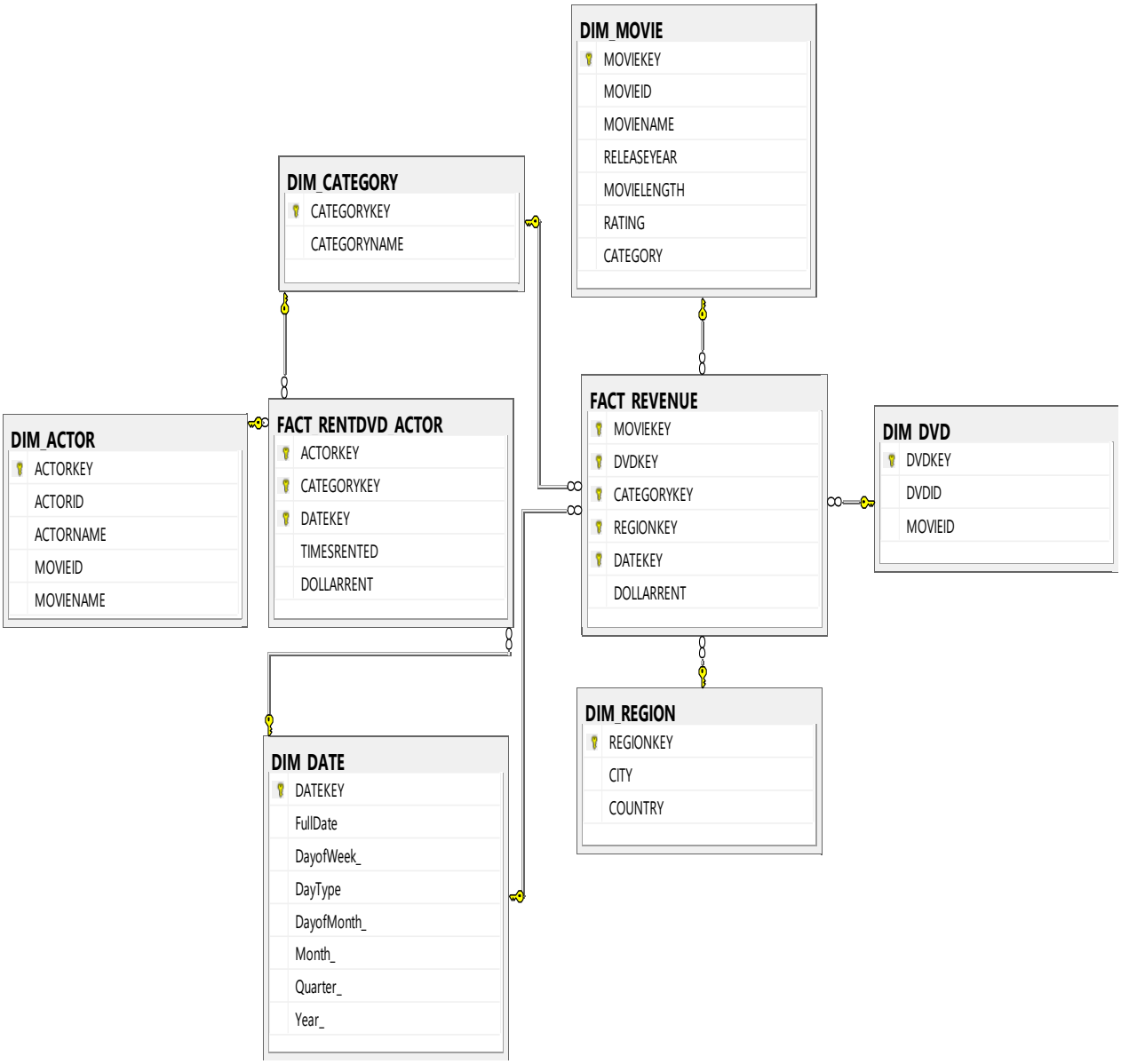
- Data warehouse acutize consistency of information by removal the all data duplications from various sources. Moreover it prevent storage space loss due to redundancy. Thereby, information derived from the data warehouse is of better use . Each dimension in data warehouse model contains primary key constraint which reduces the data duplication to save storage space. Lastly, the details in fact tables represents the major facts of analysis.

PART2:

Develop and present a suitable schema for the data warehouse. Discuss your reasons for the design.

➢ 2.1. SAKILA WAREHOUSE SCHEMA

Below is the *schema for Sakila data warehouse "SakilaDBS"*. This star schema has two fact table and six dimension table screen shots here.



➢ 2.2. DIMENSIONS AND FACT TABLES OF SAKILA WAREHOUSE

- **Dim_Category**: This dimension table contains Type of movie category.
- **Dim_Actor :** This Table contains all information about actors and actresses.
- **Dim_Movies:** This Table contains data of each movies.
- **Dim_Date:** This table contains all the transaction dates, quarter, day of week, month, year.
- **Dim_Region:** This table contains all information about customer location across the glob.
- **Dim_DVD:** This Table contains all information about stock of dvds of different movies.
- **Fact_RENT_DVD_ACTOR:** To accumulate information of actor, their kind of movies and their total contribution in rent income of the store.

  Fact measures: **TimesRented, DollarRent**

  Connects Actors, film category and rent date. **TimesRented*** shows number of dvds rented for particular actor's any movie belonging to particular category on the mentioned date. And **DollarRent*** represents the total dollar rent earned for the same.

- **FACT_REVENUE:**

  To accumulate information of movies, their category and their total contribution in rent income of the store.

  Fact measures: **DollarRent***

  Connects movie, dvds, category , region, and date. It represents dvd with dvd number belonging to particular movie of related category rented to customer of particular region on the mentioned date. And **DollarRent*** represents rent income for the same.

➢ 2.3. REASONS FOR SELECTING DESIGN

- Firstly, Sakila Warehouse follows the OLAP model. And it's data never alters.
- Secondly, It Provides cost effective performance for all the queries
- Thirdly, It describes all relationship among all dimensions.
- Lastly, The model has a simple structure so, it is easy to modify dimensions and fact tables according to requirements

PART 3- EXTRACT ,TRANSFORM AND LOAD

Using Microsoft SQL Server, implement your tables and extract, transform and load data from the operational source(s) into the data warehouse. This can be done using any available tool or by writing SQL statements.

Tool used: Microsoft SQL Server Management Studio, SQL Server Data Tools

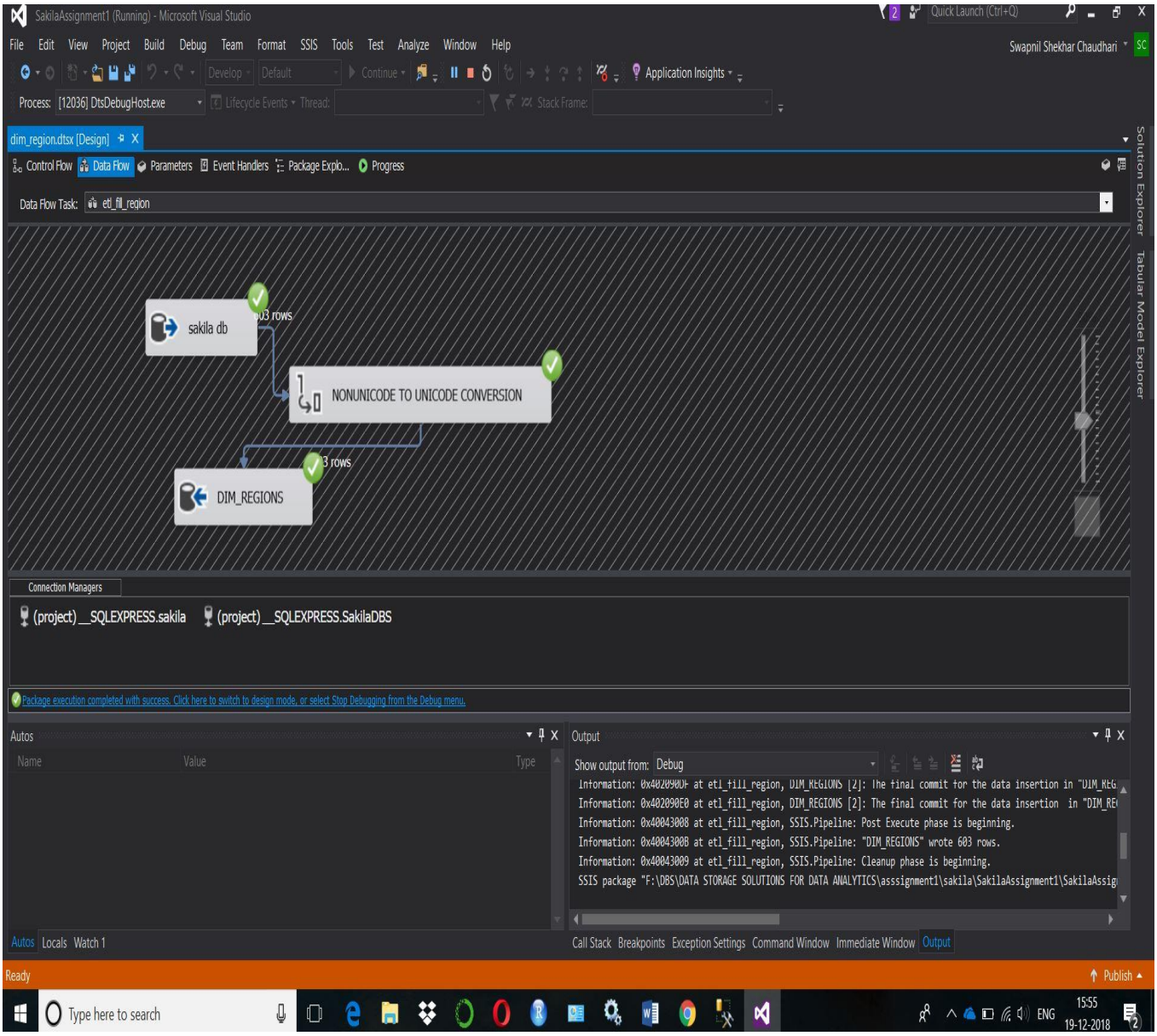
Data source: Sakila database.

ETL stands for :Extract ,Transform ,Load.

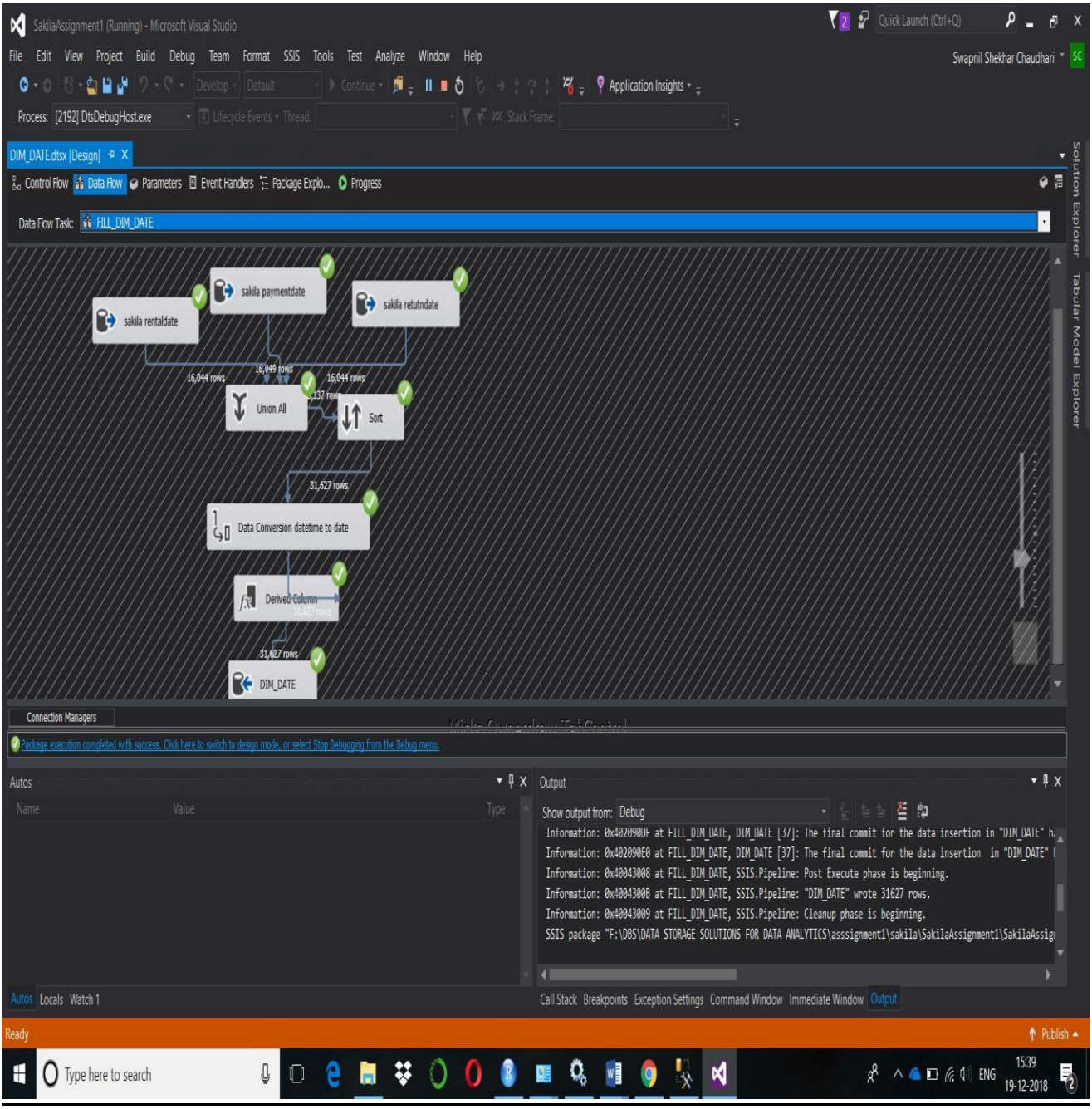
First step is the extracting the data from soure database “Sakila” with the aid of SQL Server Management studio. After creating the Dimension tables such as dim_category, dim_Date, dim_actor, dim_movie and Fact tables such as fact_revenue, Fact_rentdvd_actor. Once creating the table with their column names, we perform the following operation in the data tools which can be seen in the picture below.

➢ 3.1 ETL PROCESS FOR POPULATING DATA- SCREENSHOTS

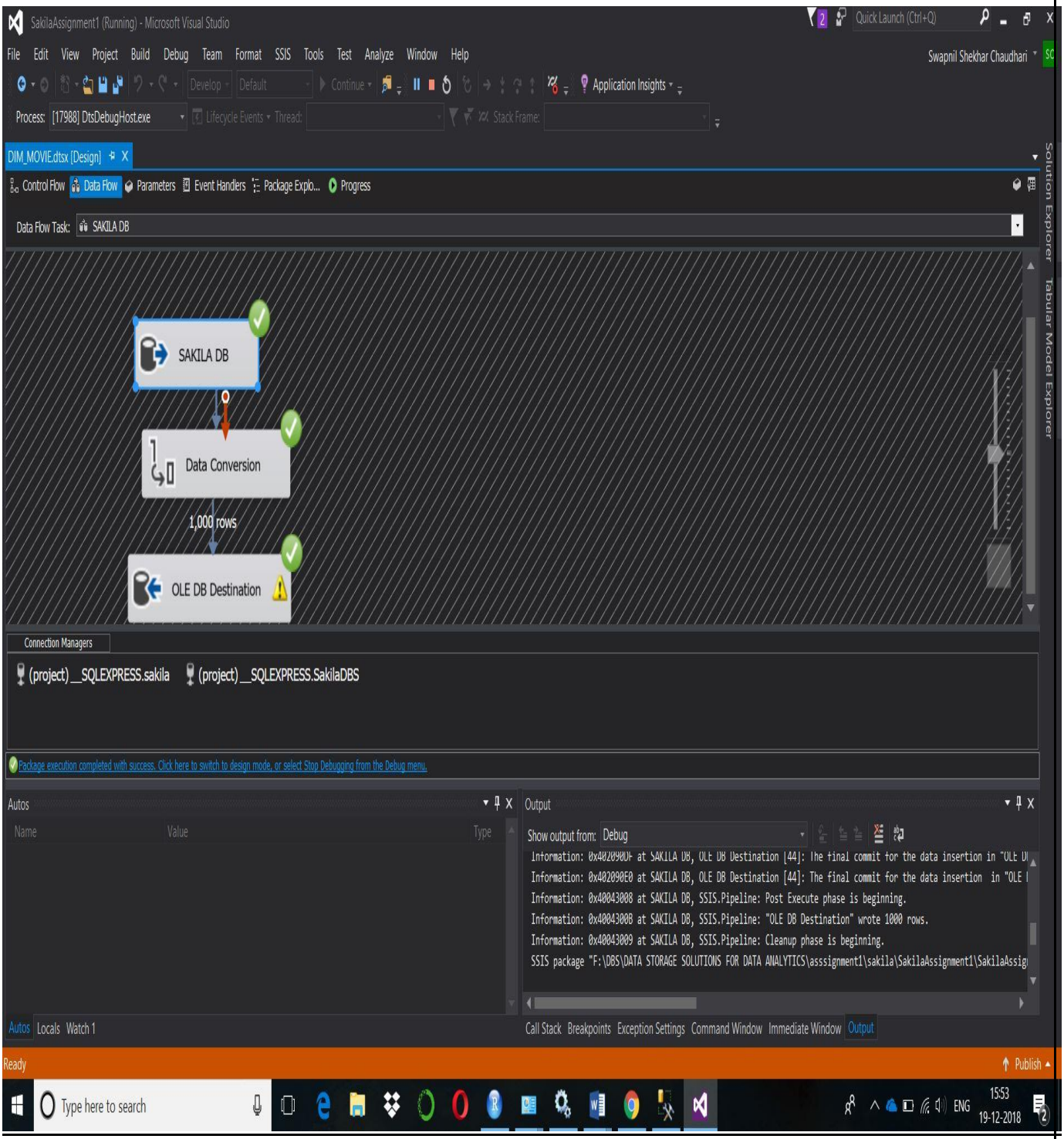
ETL PROCESS FOR DIMENSION REGION



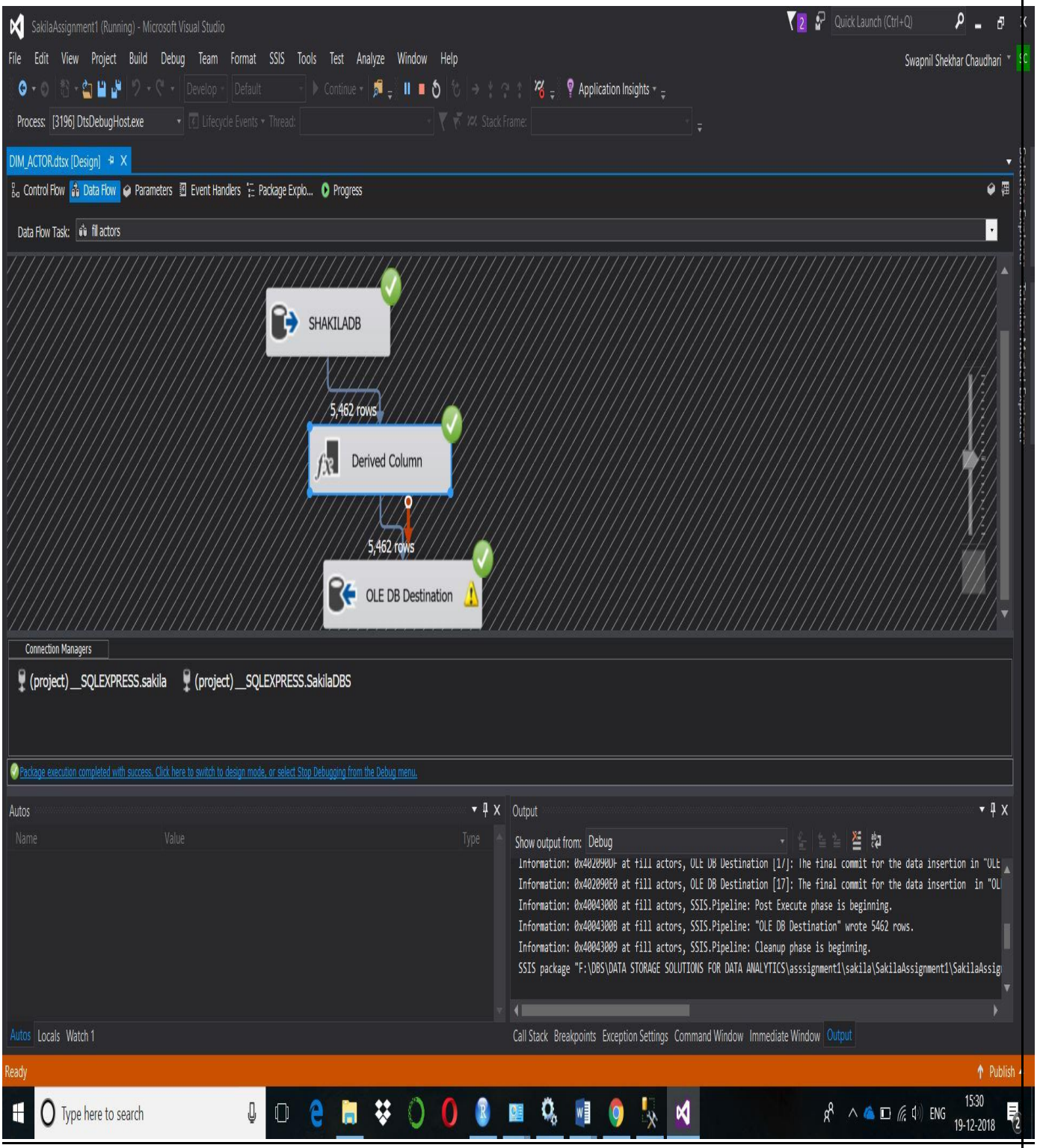
ETL PROCESS FOR DIMENSION DATE



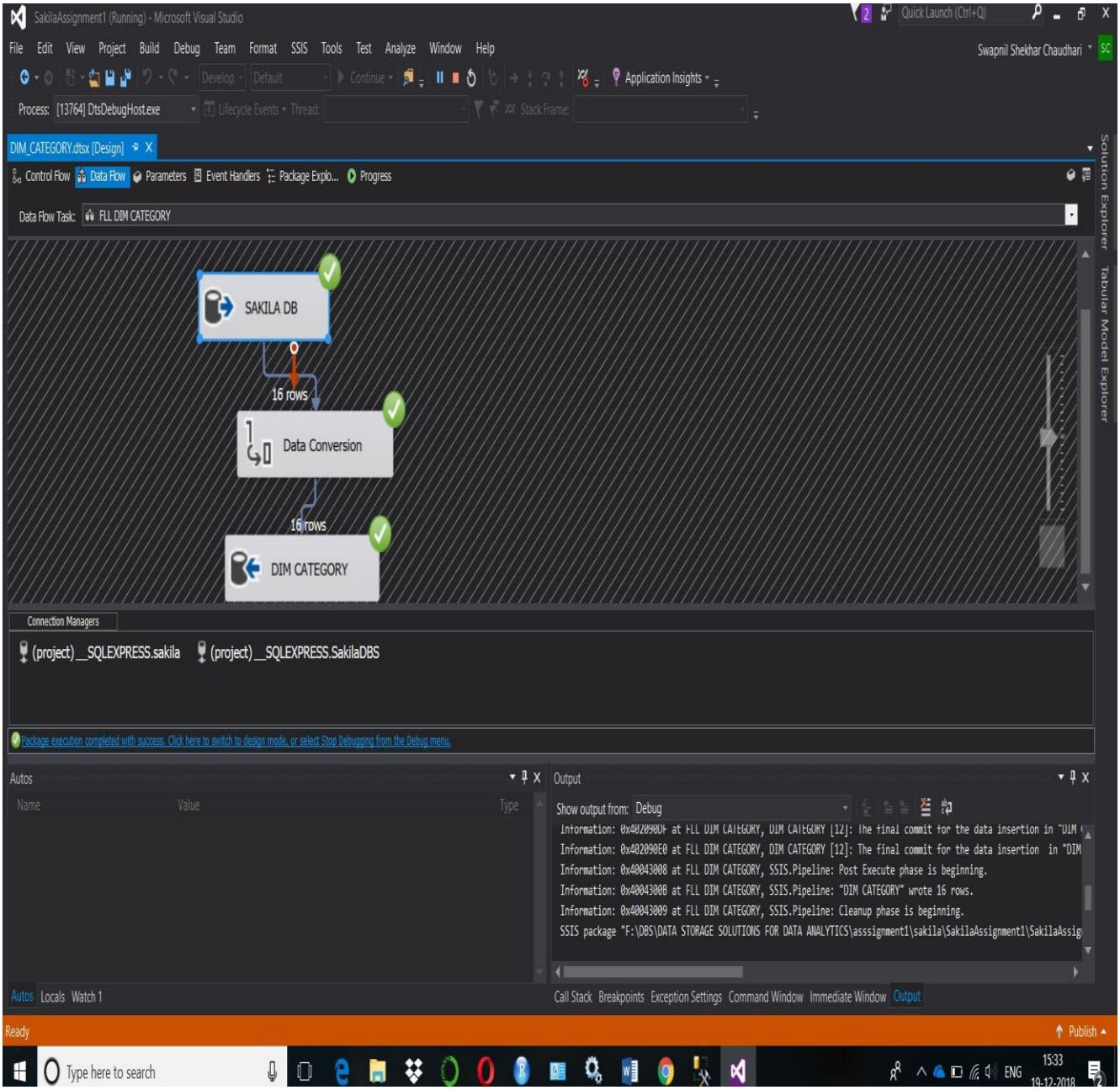
ETL PROCESS FOR DIMENSION MOVIE



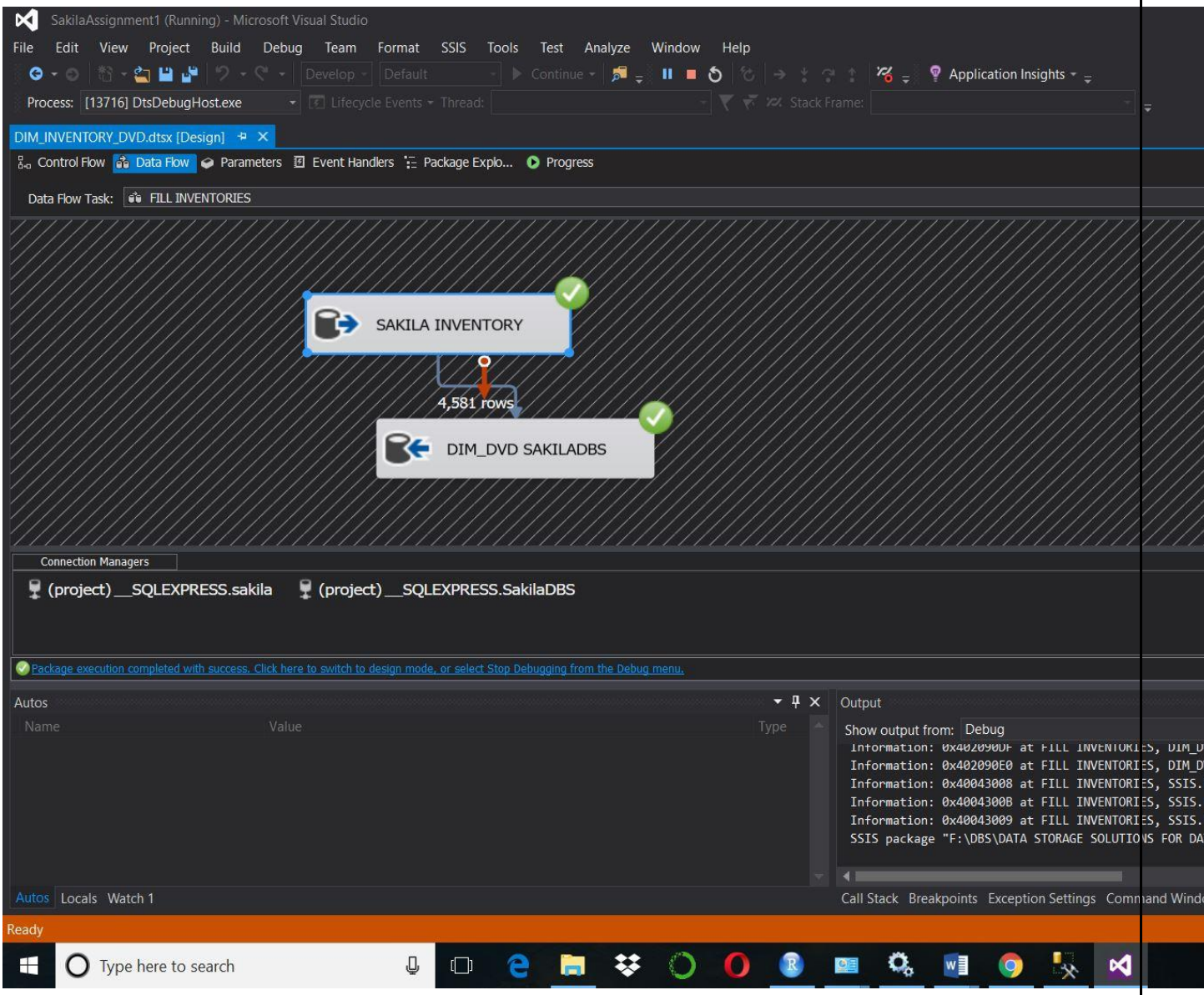
ETL PROCESS FOR DIMENSION ACTOR



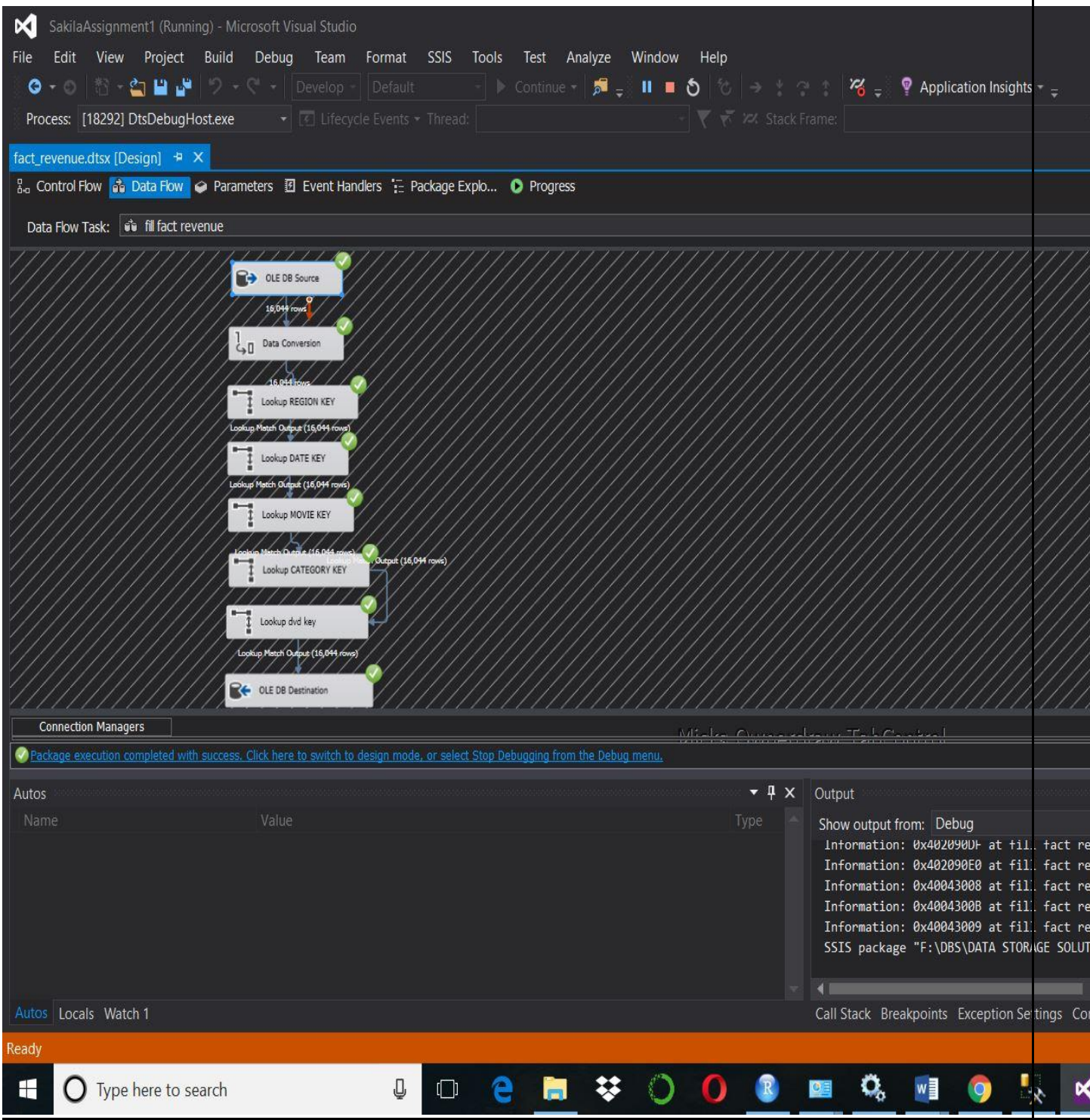
ETL PROCESS FOR DIMENSION CATEGORY



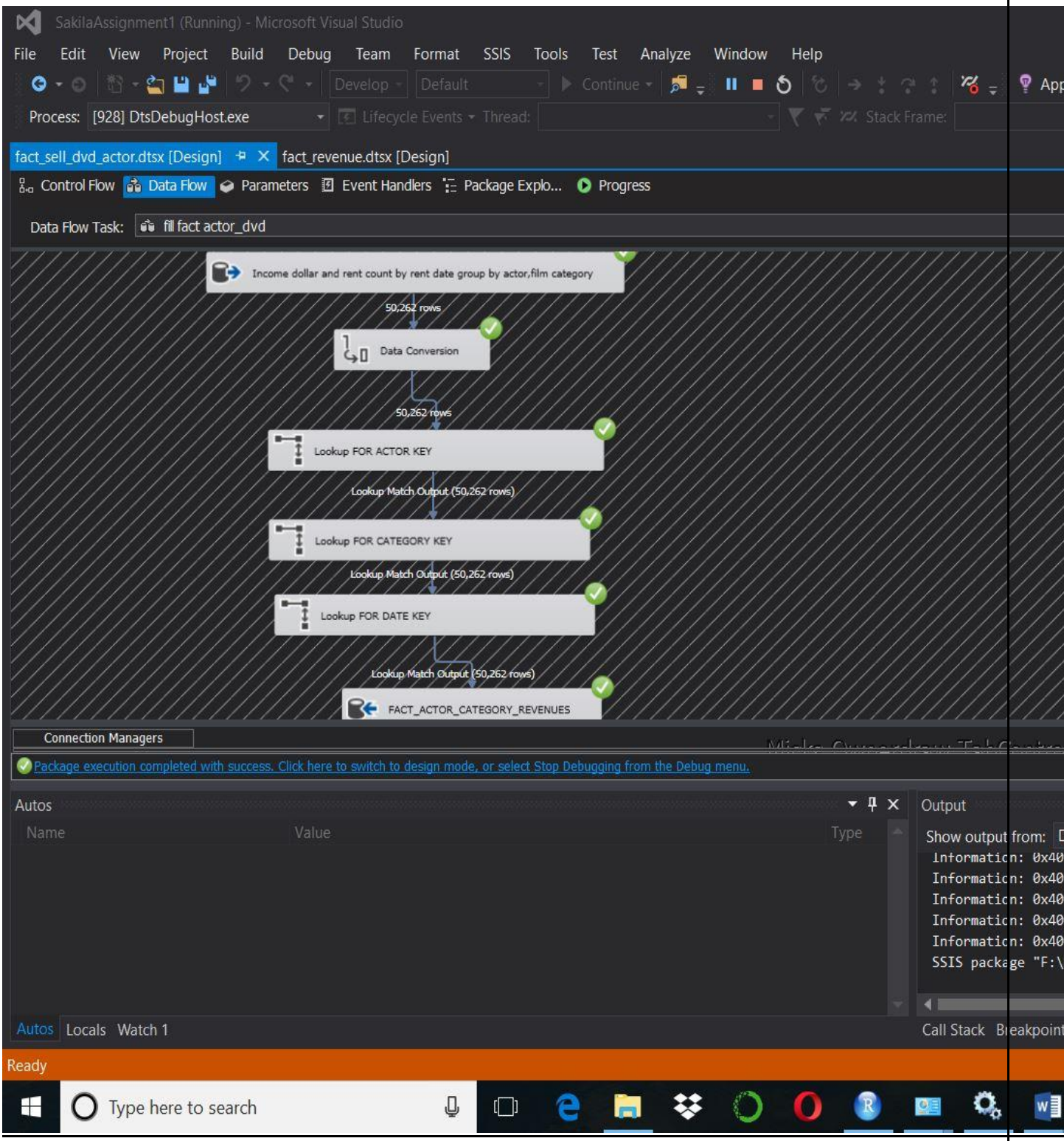
ETL PROCESS FOR DIMENSION DVD



ETL PROCESS FOR FACT REVENUE



ETL PROCESS FOR FACT RENTDVD ACTOR



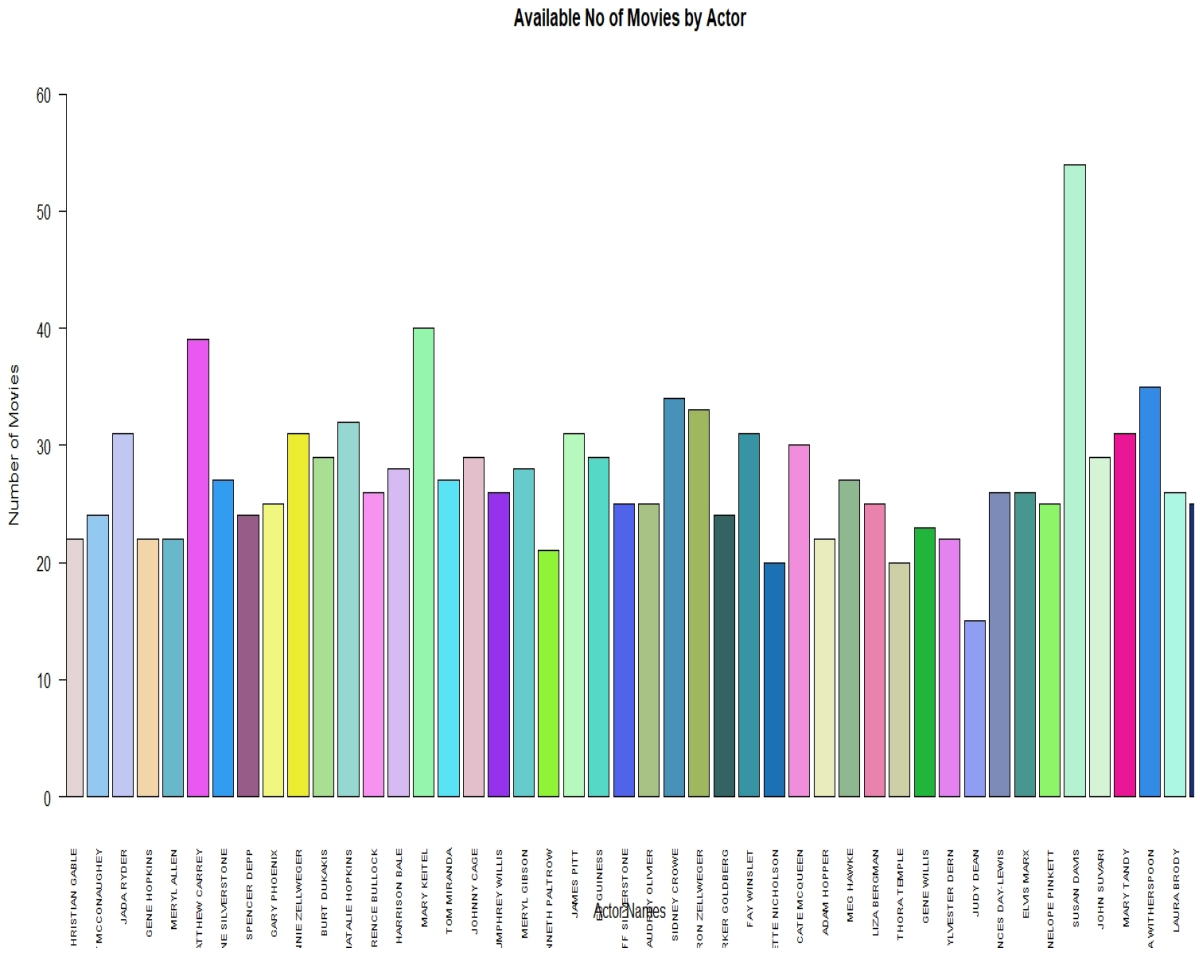
PART 4. REPORT GENERATION AND VISUALIZATION

Produce four reports in support of the requirements outlined in section 1 using SSRS. Also produce four visualisations using R programming language and discuss them.

Solution:

➢ 4.1 VISUALIZATION USING R STUDIO

REPORT 1:AVAILABLE NUMBER OF MOVIES BY EACH ACTOR



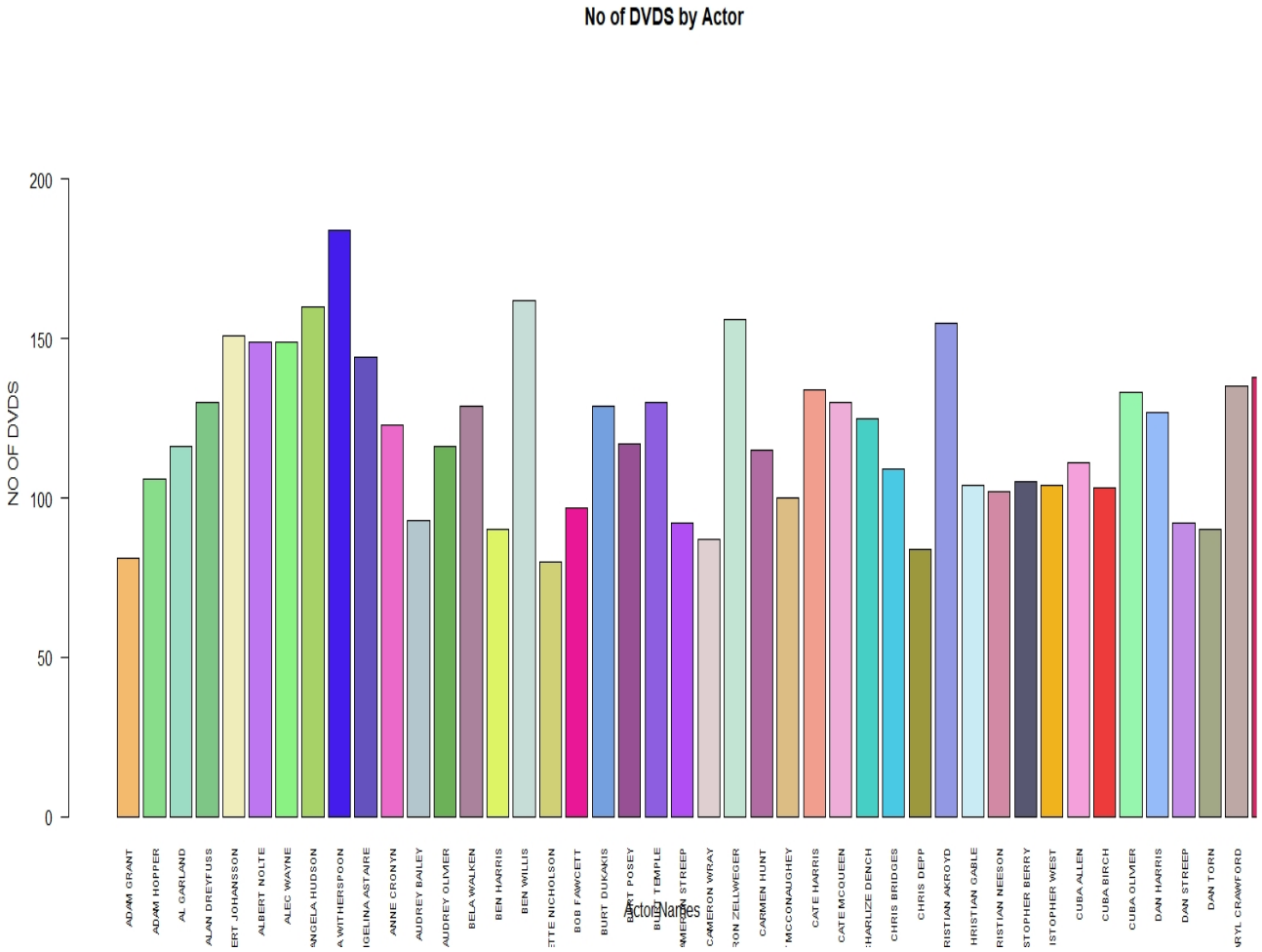
Report: Gives information about available Actor and their total movies

(Note: this is just a part of an actual graph as there are 199 columns which are impossible to accommodate in a single screen shot. But we can observe all in R Studio)

- From visualization graph we can observe 199 actors and their total movies .
- Each bar represents number of movies for each single actor.

- Among all the movies , store contains maximum number of movies of an actress ***"SUSAN DAVIS"*** that is ***54***.
- Minimum Number of movies are ***14*** for *"**EMILY DEE**"*

<u>REPORT 2:</u> AVAILABLE DVDS BY EACH ACTOR



Report: Gives information about Actors and their total dvds

(Note: this is just a part of an actual graph as there are 199 columns which are impossible to accommodate in a single screen shot. But we can observe all in R Studio)

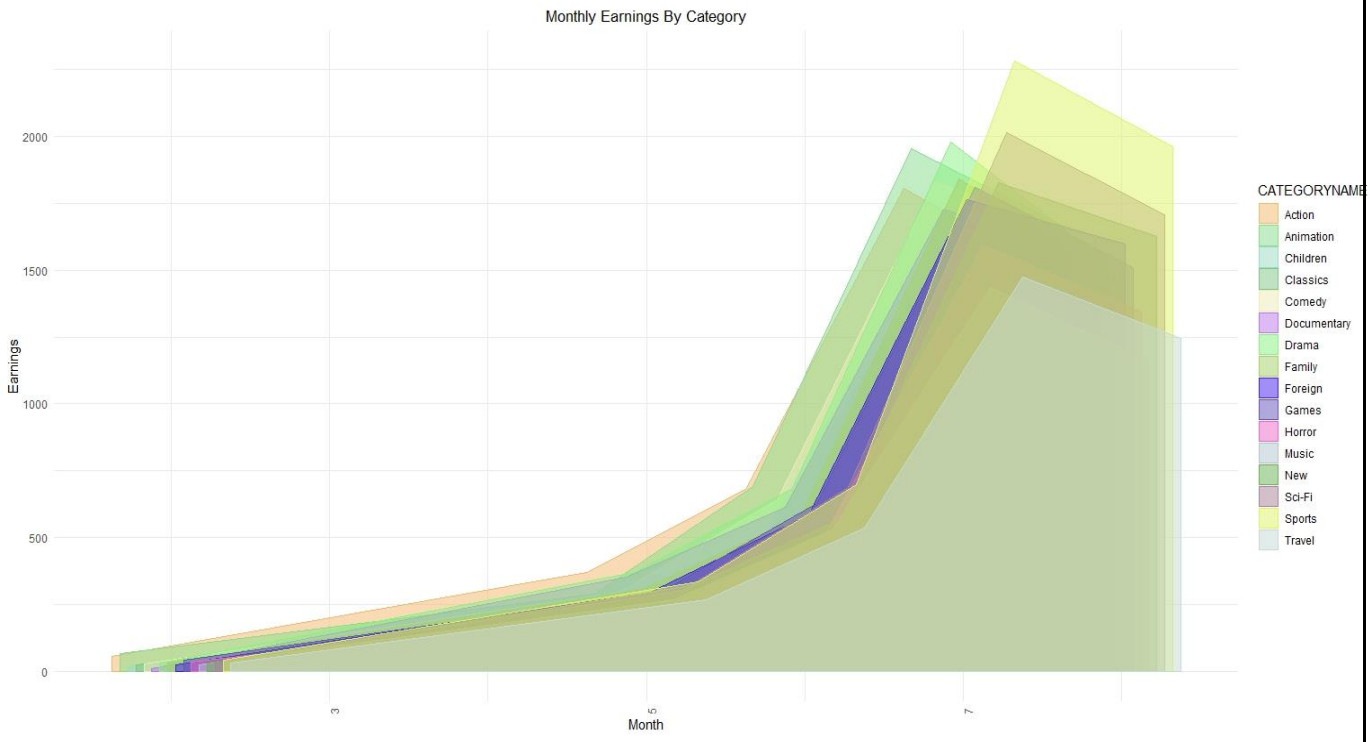
- From visualization graph we can observe 199 actors and their total movies .
- Each bar represents number of movies for each single actor.
- Among all the movies , store contains maximum number of movie dvds of an actress ***"SUSAN DAVIS"*** that is ***236***.
- Minimum Number of movie dvds are ***62*** for *"**EMILY DEE**"*.

REPORT 3: Earnings by Date

Report: Gives information about income pattern of the store

- From visualization graph we can observe the pattern of income.
- A steep decrease and a sudden increase is observed in each week
- Observed earning ranges between 0 to maximum of about $2800 .

REPORT 4: Earnings By Month and Film Category



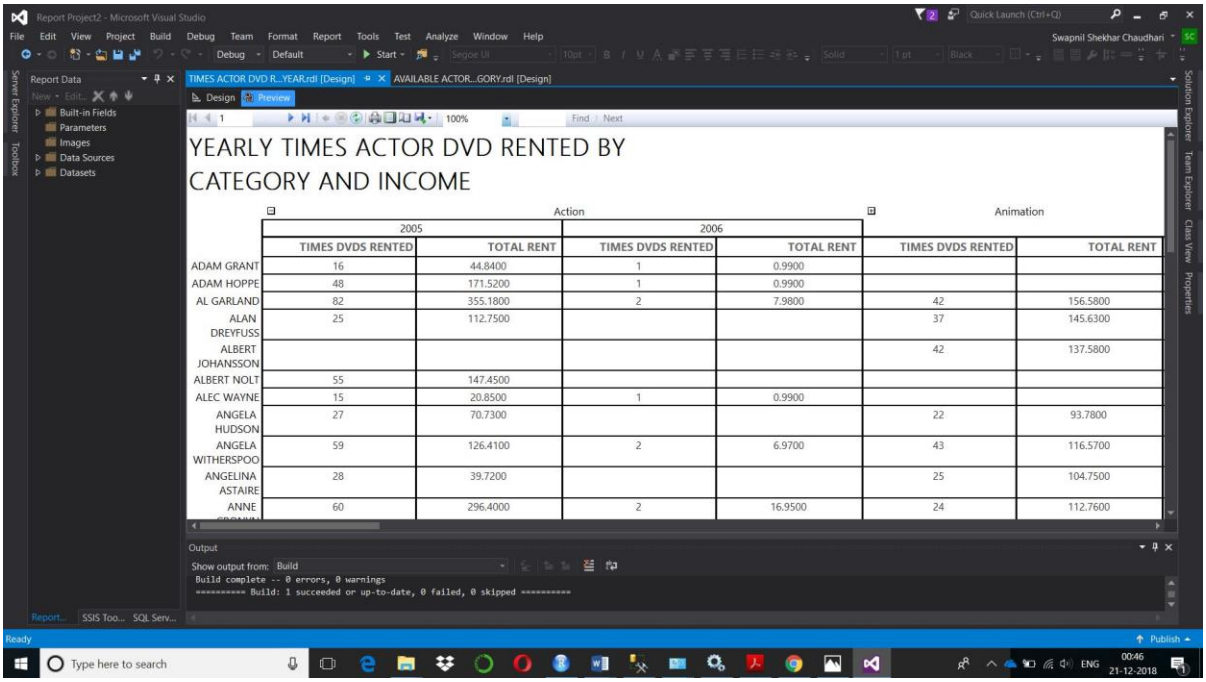
Report: Gives information about monthly income by film category

- From visualization graph depicts income contribution by each film category.
- It is observed that maximum rent is earned from sports category movies, that is about $2300.
- Hence, store can decide to increase movies and dvds related to sport categories.

➢ 4.2 REPORTS USING SSRS

Tool: SQL Server Data Tools:

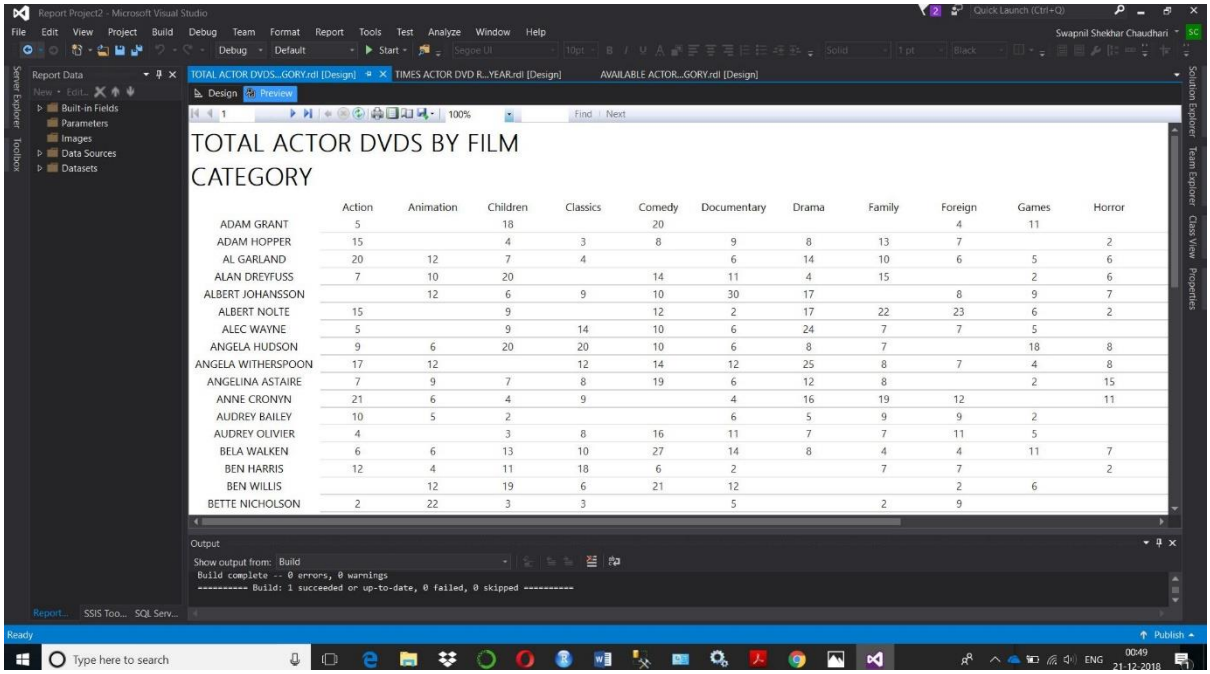
<u>REPORT 5</u>: NUMBER OF TIMES ACTOR DVD RENTED BY FILM CATEGORY AND RELATED INCOME



Report: Gives information about total yearly income by actor and film category

- This report shows yearly earning by contribution of actors and film category.
- From this report management can decide which actors' which kind of movies are contributing majorly and minimally in the store earnings.
- Management can also take step to reduce movies of particular actor in particular less contributing film category. *For example travel movies of ELLEN PRESLEY has generated income of mere 16 euros.*
- Based on this it is observed that , in overall sports category dominates in the stores total income.

REPORT 6:TOTAL ACTOR DVDS BY FILM CATEGORY



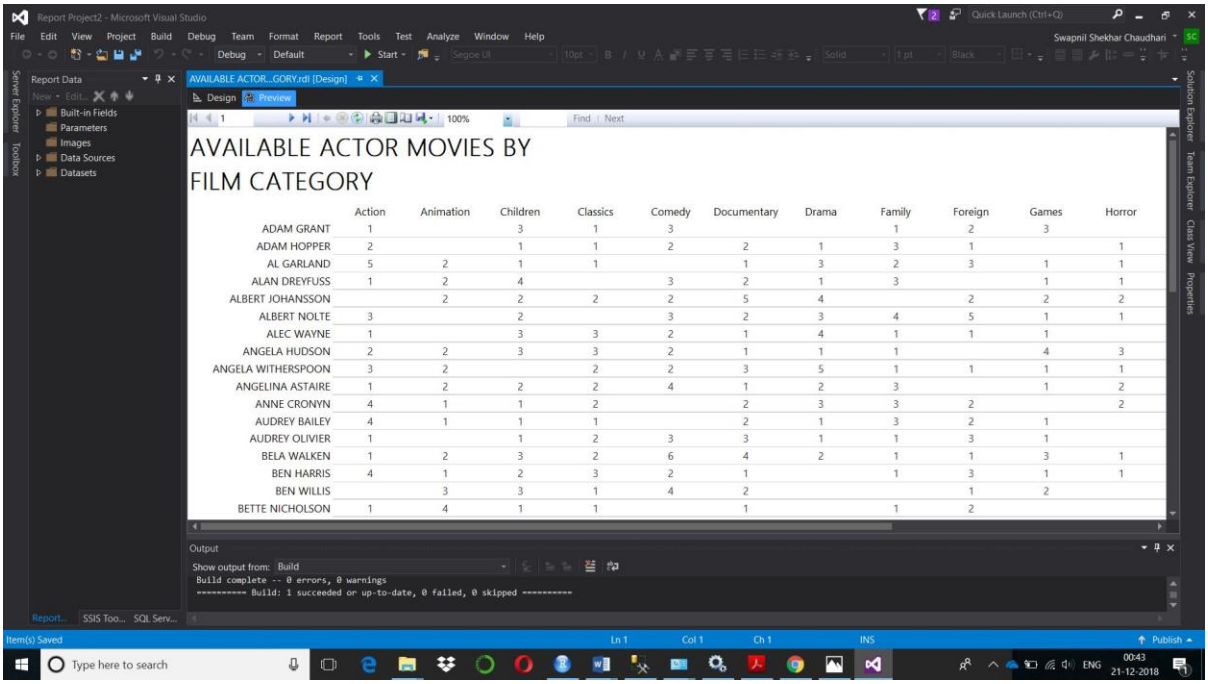
Report:

Gives information about total actor dvds in each film category

	Action	Animation	Children	Classics	Comedy	Documentary	Drama	Family	Foreign	Games	Horror
ADAM GRANT	5		18		20				4	11	
ADAM HOPPER	15		4	3	8	9	8	13	7		2
AL GARLAND	20	12	7	4		6	14	10	6	5	6
ALAN DREYFUSS	7	10	20		14	11	4	15		2	6
ALBERT JOHANSSON		12	6	9	10	30	17		8	9	7
ALBERT NOLTE	15		9		12	2	17	22	23	6	2
ALEC WAYNE	5		9	14	10	6	24	7	7	5	
ANGELA HUDSON	9	6	20	20	10	6	8	7		18	8
ANGELA WITHERSPOON	17	12		12	14	12	25	8	7	4	8
ANGELINA ASTAIRE	7	9	7	8	19	6	12	8		2	15
ANNE CRONYN	21	6	4	9		4	16	19	12		11
AUDREY BAILEY	10	5	2			6	5	9	9	2	
AUDREY OLIVIER	4		3	8	16	11	7	7	11	5	
BELA WALKEN	6	6	13	10	27	14	8	4	4	11	7
BEN HARRIS	12	4	11	18	6	2		7	7		2
BEN WILLIS		12	19	6	21	12			2	6	
BETTE NICHOLSON	2	22	3	3		5		2	9		

- Previous report aids to decide which actors particular kind of movies are rented how many times. Accordingly, decision should be taken from this report to increase or decrease related movies for particular film category and actor .

REPORT 7:AVAILABLE ACTOR MOVIES BY FILM CATEGORY

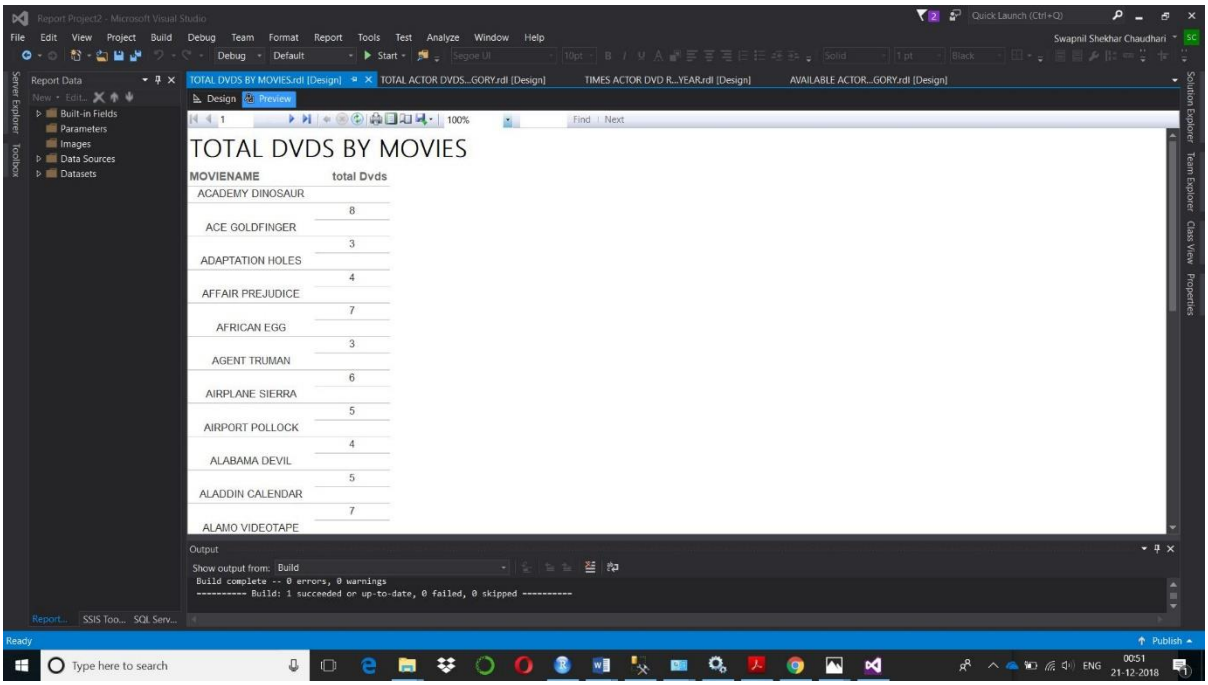


	Action	Animation	Children	Classics	Comedy	Documentary	Drama	Family	Foreign	Games	Horror
ADAM GRANT	1		3	1	3			1	2	3	
ADAM HOPPER	2		1	1	2	2	1	3	1		1
AL GARLAND	5	2	1	1		1	3	2	3	1	1
ALAN DREYFUSS	1	2	4		3	2	1	3		1	1
ALBERT JOHANSSON		2	2	2	2	5	4		2	2	2
ALBERT NOLTE	3		2		3	2	3	4	5	1	1
ALEC WAYNE	1		3	3	2	1	4	1	1	1	
ANGELA HUDSON	2	2	3	3	2	1	1	1		4	3
ANGELA WITHERSPOON	3	2		2	2	3	5	1	1	1	1
ANGELINA ASTAIRE	1	2	2	2	4	1	2	3		1	2
ANNE CRONYN	4	1	1	2		2	3	3	2		2
AUDREY BAILEY	4	1	1	1		2	1	3	2	1	
AUDREY OLIVIER	1		1	2	3	3	1	1	3	1	
BELA WALKEN	1	2	3	2	6	4	2	1	1	3	1
BEN HARRIS	4	1	2	3	2	1		1	3	1	1
BEN WILLIS		3	3	1	4	2			1	2	
BETTE NICHOLSON	1	4	1	1		1		1	2		

Report: Available actor movies by film category

- This reports elicits availability of number of movies by each actor in various category.

REPORT 8: TOTAL DVDS FOR EACH NOVIES



Report: Available dvds for each movie

- This reports displays inventory of dvds for each movies.

PART 5.: XML AND XSD OF DATA WAREHOUSE CUBE

In this spirit and to understand XML in a better way, develop an XML Schema based on the data warehouse data that a cube may capture and apply it to develop an XML document.

Solution:

Choosen Fact Table: FACT REVENUE

Choosen Dimensions: DIMENSION MOVIE, DIMENSION CATEGORY, DIMENSION REGION

Xsd: revenue.xsd

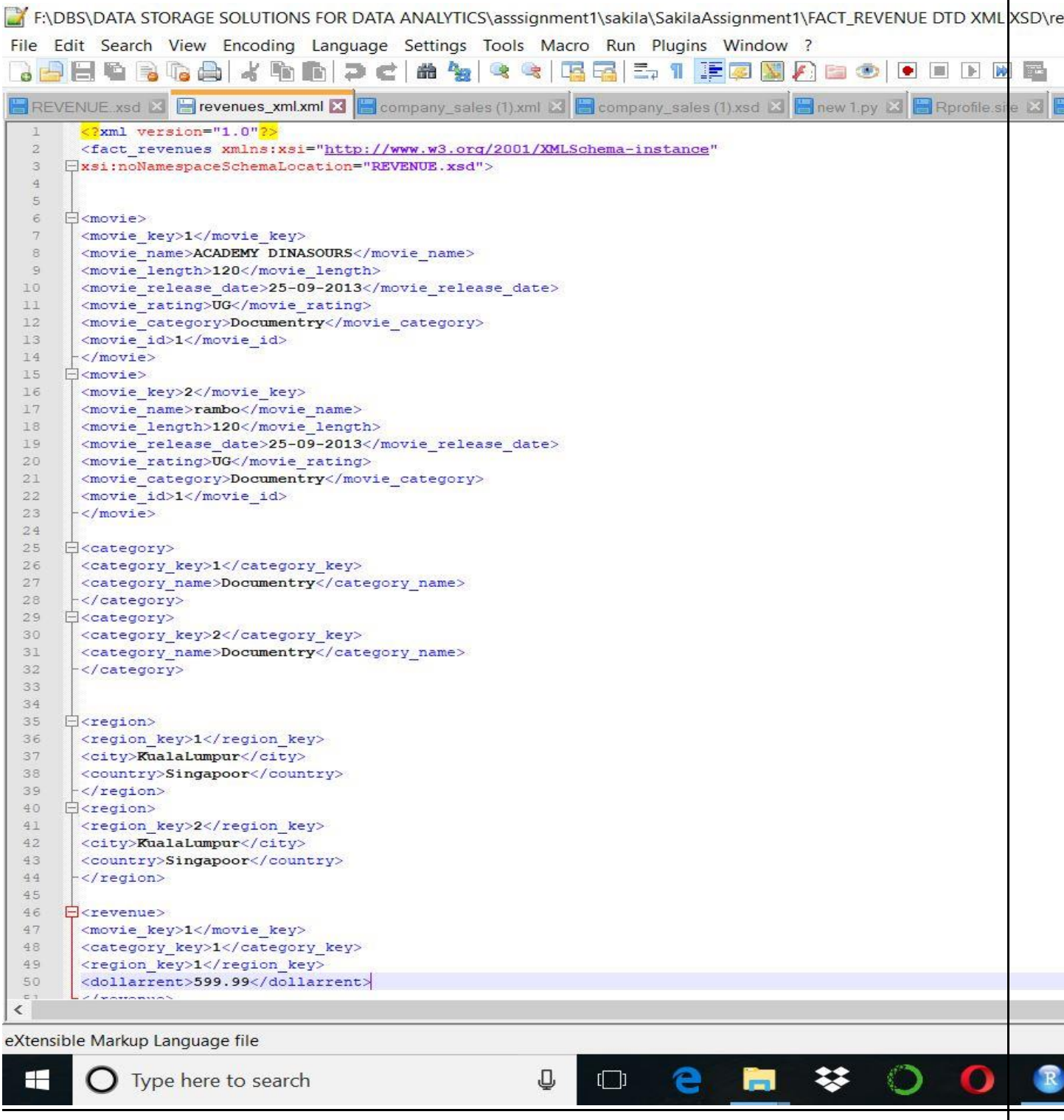
F:\DBS\DATA STORAGE SOLUTIONS FOR DATA ANALYTICS\asssignment1\sakila\SakilaAssignment1\FACT_REVENUE DTD XML XSD\REVENUE..xsd - Notepad++

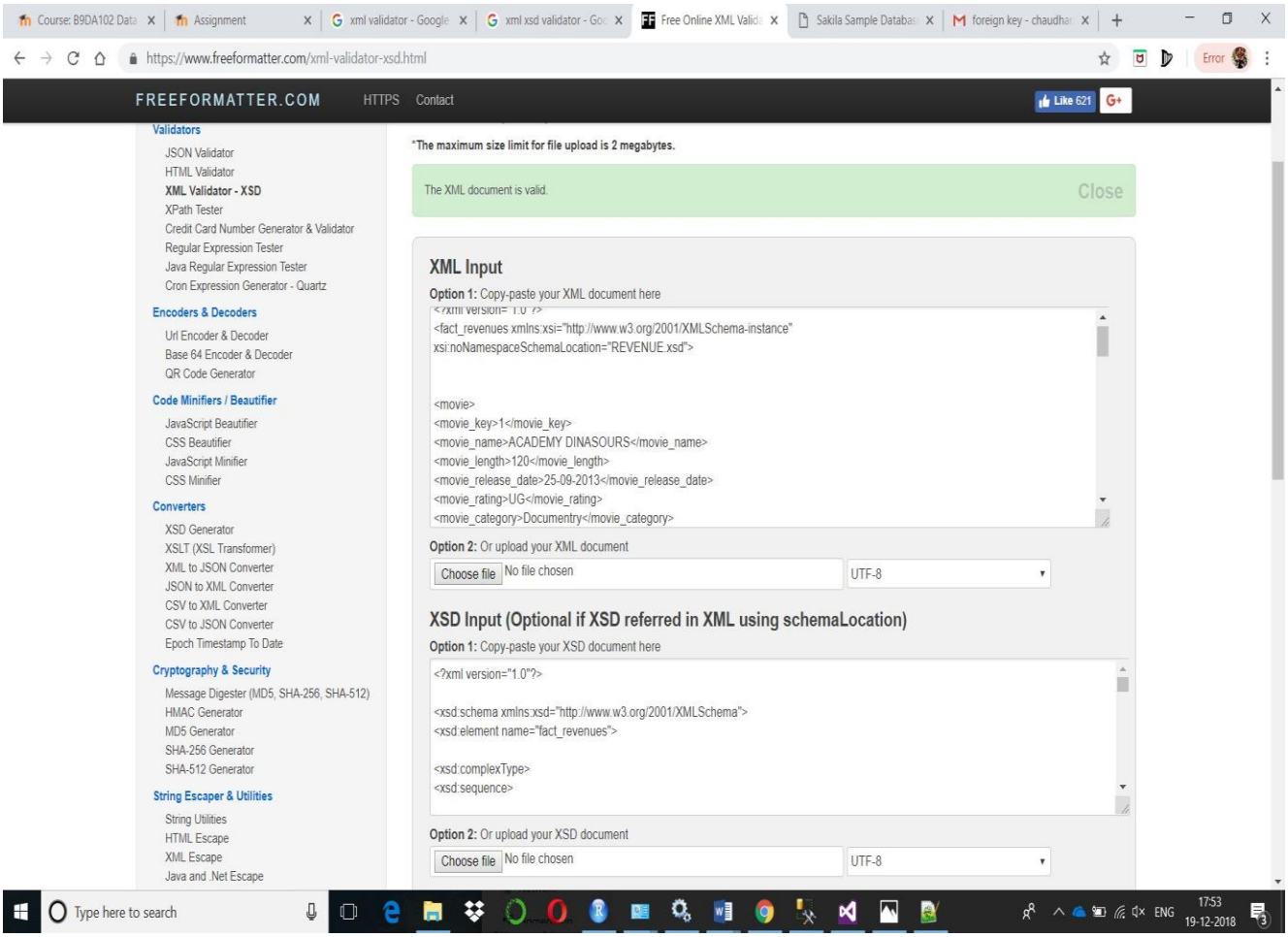
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

REVENUE..xsd | revenues_xml.xml | company_sales (1).xml | company_sales (1).xsd | new 1.py | Rprofile.site | imdb.txt | imdb.txt

```
<?xml version="1.0"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="fact_revenues">

<xsd:complexType>
<xsd:sequence>


<xsd:element ref="movie" minOccurs="1" maxOccurs="unbounded" />
<xsd:element ref="category" minOccurs="1" maxOccurs="unbounded" />
<xsd:element ref="region" minOccurs="1" maxOccurs="unbounded" />
<xsd:element ref="revenue" minOccurs="1" maxOccurs="unbounded" />
</xsd:sequence>
</xsd:complexType>

<xsd:key name="movie_key">
<xsd:selector xpath="movie/movie_key" />
<xsd:field xpath="." />
</xsd:key>
<xsd:keyref name="movie_key_ref" refer="movie_key">
<xsd:selector xpath="revenue/employee_id" />
<xsd:field xpath="." />
</xsd:keyref>

<xsd:key name="category_key">
<xsd:selector xpath="category/category_key" />
<xsd:field xpath="." />
</xsd:key>
<xsd:keyref name="category_key_ref" refer="category_key">
<xsd:selector xpath="revenue/customer_id" />
<xsd:field xpath="." />
</xsd:keyref>

<xsd:key name="region_key">
<xsd:selector xpath="region/region_key" />
<xsd:field xpath="." />
</xsd:key>
<xsd:keyref name="region_key_ref" refer="region_key">
<xsd:selector xpath="revenue/region_key" />
<xsd:field xpath="." />
</xsd:keyref>
</xsd:element>

```

Xml:revenue_xml.xml





xml-xsd Validator: https://www.freeformatter.com/xml-validator-xsd.html

PART 6: GRAPH DATABASE

Implement a part of your source database or data warehouse as graph database using Neo4j technologies. Discuss the use of graph databases in comparison to relational databases.

Solution:

Tool: Neo4j

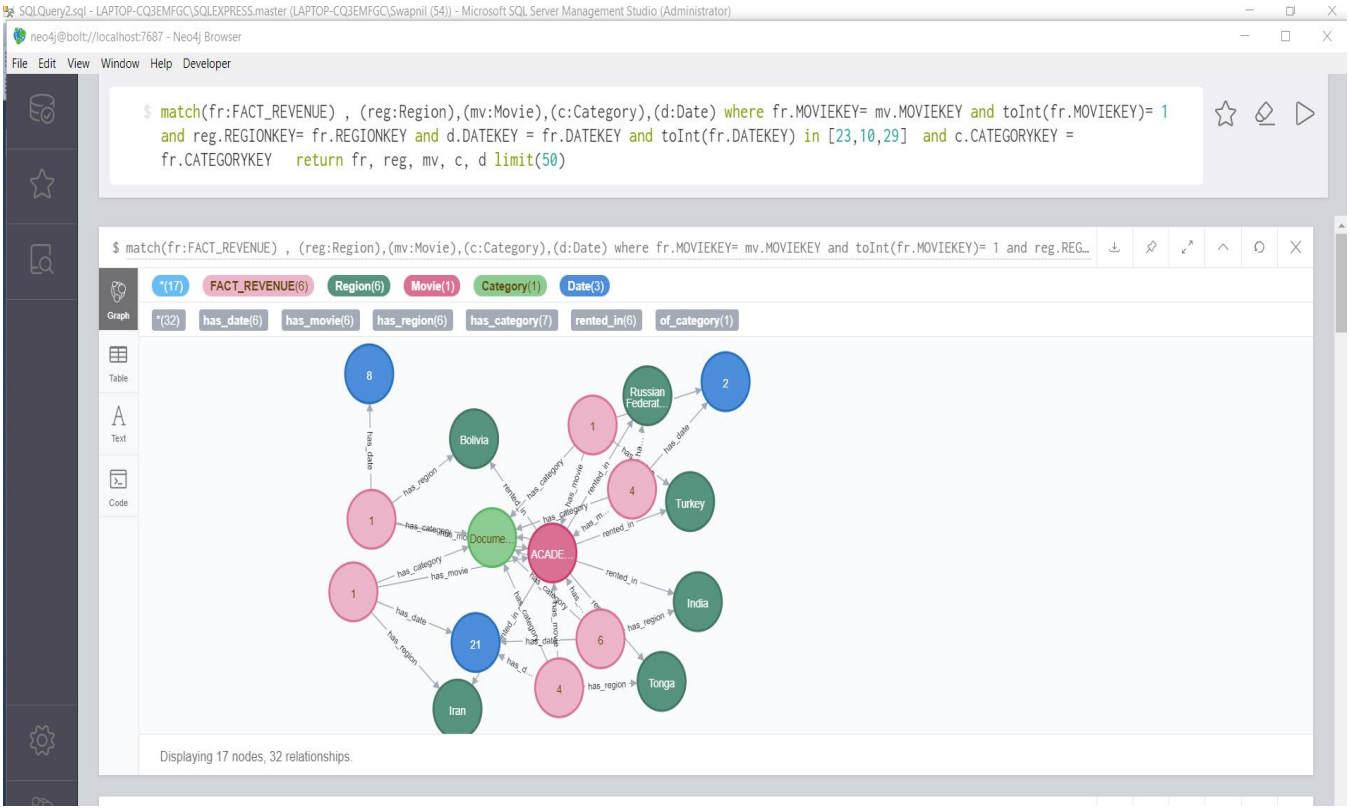
Language: Cypher Query Language

Choosen Fact Table: FACT REVENUE

Choosen Dimensions: DIMENSION MOVIE, DIMENSION CATEGORY, DIMENSION DATE

Comparison of Graph Database And Relational Database

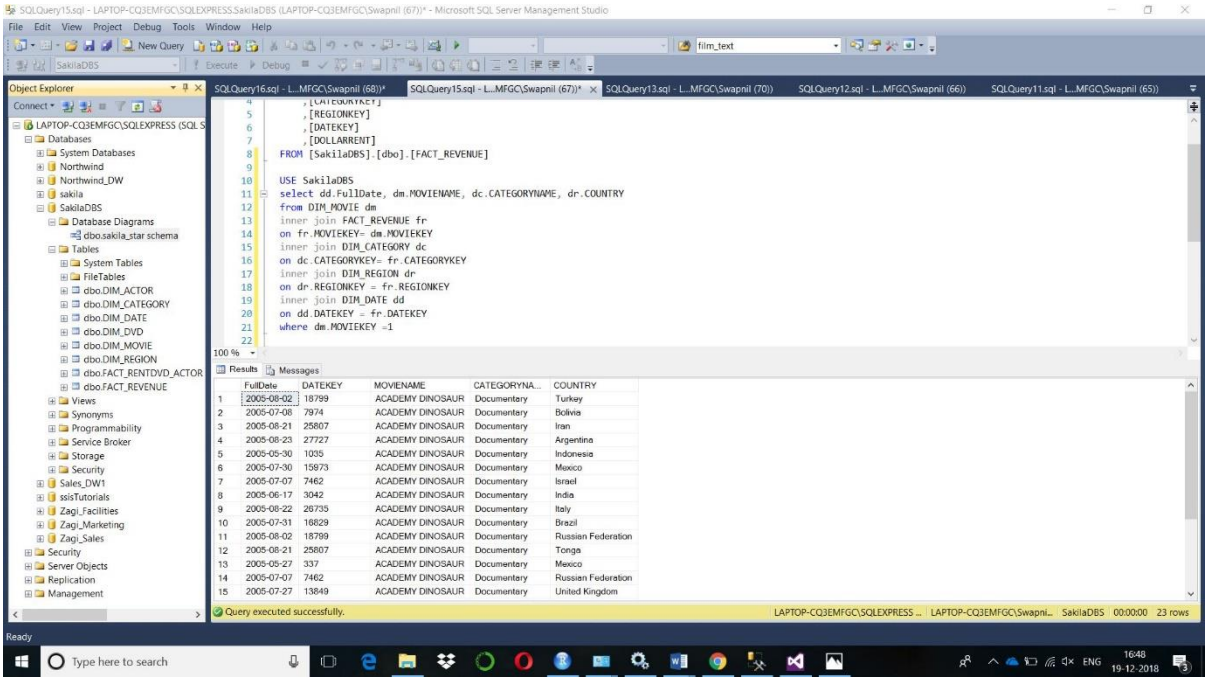
- Because of rigid structure, relational databases compels developers and applications to follow the structure their applications. While Graph database have no strict structure. Key value concept to store value for each property of nodes. Hence resulting in better performance and small queries.
- In relational databases, rows and tables are referred by primary key attributes via foreign key columns. Joins are used to query by matching primary and foreign keys of all rows in the related tables. However these operations are memory-intensive and have significant cost. the relationships are stored at the individual record level in graph database. Hence
- When many-to-many relationships is observed in the database, we use joins, which further increasing join operation costs.
- Those kind of time consuming join operations are addressed by denormalizing the data to reduce the number of joins necessary, thereby breaking the data integrity of a relational database.



Explanation:

- All rows of fact_revenues related to moviekey1(***"ACADEMY DINASOURS"***.
- Blue nodes are date nodes (datekey, full date,…) which represents movie dvd rent date .
- pink nodes are the fact_revenue node that connects all other nodes with relations.
- Dark green nodes represents country of customer
- lite green nodes are film category nodes ("Documentary").

Same data represented in developed data warehouse “SakilaDBS”



BIBLIOGRAPHY

Dev.mysql.com. (2018). *MySQL :: Sakila Sample Database*. [online] Available at: https://dev.mysql.com/doc/sakila/en/ [Accessed 22 Dec. 2018].

Freeformatter.com. (2018). *Free Online XML Validator Against XSD Schema - FreeFormatter.com*. [online] Available at: https://www.freeformatter.com/xml-validator-xsd.html [Accessed 23 Dec. 2018].

Neo4j Graph Database Platform. (2018). *Relational Databases vs. Graph Databases: A Comparison*. [online] Available at: https://neo4j.com/developer/graph-db-vs-rdbms/ [Accessed 23 Dec. 2018].

Serra, J. (2018). *Why You Need a Data Warehouse*. [online] James Serra's Blog. Available at: https://www.jamesserra.com/archive/2013/07/why-you-need-a-data-warehouse/ [Accessed 23 Dec. 2018].

APPENDIX A – SQL QUERIES

Creating Dimension Region

```
CREATE TABLE DIM_REGION
(
REGIONKEY  INT IDENTITY(1,1) NOT NULL,

CITY NVARCHAR(255) NOT NULL ,

COUNTRY NVARCHAR(255) ,

CONSTRAINT PK_DIM_REGION PRIMARY KEY (REGIONKEY)

);
```

Creating Dimension Date

```
CREATE TABLE DIM_DATE
(
DATEKEY INT  IDENTITY(1,1) NOT NULL,
[FullDate] [date] NULL,
[DayofWeek_] NVARCHAR(15) NULL,
[DayType] NVARCHAR(20) NULL,
[DayofMonth_] [int] NULL,
[Month_] NVARCHAR(10) NULL,
[Quarter_] NVARCHAR(2) NULL,
[Year_] [int] NULL,

CONSTRAINT PK_DIM_DATE PRIMARY KEY (DATEKEY)

);
```

Creating Dimension Movie

```
CREATE TABLE DIM_MOVIE
(
MOVIEKEY INT IDENTITY(1,1) NOT NULL,
MOVIEID INT NOT NULL,
MOVIENAME NVARCHAR(255),
RELEASEYEAR DATE,
MOVIELENGTH INT,
RATING NVARCHAR(5),
CATEGORY NVARCHAR(100)

CONSTRAINT PK_DIM_MOVIE PRIMARY KEY (MOVIEKEY)
```

```
);
```

Creating Dimension Actor

```
CREATE TABLE DIM_ACTOR
(
ACTORKEY INT IDENTITY(1,1) NOT NULL,
ACTORID INT NOT NULL,
ACTORNAME NVARCHAR(255) NOT NULL,
MOVIEID INT NOT NULL,
MOVIENAME NVARCHAR(255) NOT NULL,

CONSTRAINT PK_DIM_ACTOR PRIMARY KEY(ACTORKEY)
);
```

Creating Dimension Category

```
CREATE TABLE DIM_CATEGORY
(
CATEGORYKEY INT IDENTITY(1,1) NOT NULL,
CATEGORYNAME NVARCHAR(255) NOT NULL,

CONSTRAINT PK_DIM_CATEGORY PRIMARY KEY(CATEGORYKEY)
);
```

Creating Dimension DVD

```
CREATE TABLE DIM_DVD
```

```
(
DVDKEY INT IDENTITY(1,1),
DVDID INT NOT NULL,
MOVIEID INT NOT NULL,

CONSTRAINT PK_DIM_DVD PRIMARY KEY(DVDKEY)
);
```

Creating Fact Revenue

```
CREATE TABLE FACT_REVENUE
(
MOVIEKEY INT NOT NULL,
DVDKEY INT NOT NULL,
CATEGORYKEY INT NOT NULL,
REGIONKEY INT NOT NULL,

DATEKEY INT NOT NULL,
DOLLARRENT MONEY,

CONSTRAINT FK_DIM_REVENUE_MOVIE FOREIGN KEY(MOVIEKEY)
REFERENCES  DIM_MOVIE (MOVIEKEY)
ON DELETE NO ACTION,

CONSTRAINT FK_DIM_REVENUE_DVD FOREIGN KEY(DVDKEY)
REFERENCES  DIM_DVD (DVDKEY)
ON DELETE NO ACTION,
```

```
CONSTRAINT FK_DIM_REVENUE_CATEGRY FOREIGN
KEY(CATEGORYKEY) REFERENCES  DIM_CATEGORY (CATEGORYKEY)

ON DELETE NO ACTION,


CONSTRAINT FK_DIM_REVENUE_REFION FOREIGN KEY(REGIONKEY)
REFERENCES  DIM_REGION (REGIONKEY)

ON DELETE NO ACTION,


CONSTRAINT FK_DIM_REVENUE_DATE FOREIGN KEY(DATEKEY)
REFERENCES  DIM_DATE (DATEKEY)

ON DELETE NO ACTION,


CONSTRAINT PK_FACT_REVENUE PRIMARY
KEY(MOVIEKEY,DVDKEY,CATEGORYKEY, REGIONKEY,DATEKEY )

);
```

Creating Fact RentDvd Actor

```
CREATE TABLE FACT_RENTDVD_ACTOR

(


ACTORKEY INT  NOT NULL,

CATEGORYKEY INT NOT NULL,

DATEKEY INT NOT NULL,


TIMESRENTED INT NOT NULL,

DOLLARRENT MONEY,


CONSTRAINT PK_FACT_RENTDVD_ACTOR PRIMARY KEY(ACTORKEY,
DATEKEY, CATEGORYKEY),
```

```
CONSTRAINT FK_FACT_RENTDVD_ACTOR_DIMCATEGORY FOREIGN
KEY(CATEGORYKEY) REFERENCES  DIM_CATEGORY(CATEGORYKEY)

ON DELETE NO ACTION,


CONSTRAINT FK_FACT_RENTDVD_ACTOR_DIMDATE FOREIGN
KEY(DATEKEY) REFERENCES  DIM_DATE (DATEKEY)

ON DELETE NO ACTION,


CONSTRAINT FK_FACT_RENTDVD_ACTOR_DIMACTOR FOREIGN
KEY(ACTORKEY) REFERENCES  DIM_ACTOR (ACTORKEY)

ON DELETE NO ACTION,


);
```

APPENDIX B- SSRS QUERIES

ETL to populate Date Dimension

QUERY RENTAL DATE

SELECT rental_date

FROM rental

QUERY PAYMENT DATE

SELECT payment_date

FROM payment

QUERY RETURN DATE

SELECT return_date

FROM rental

After applying union on all these dates, all are sorted.

Now extracting other details from Date_ in derived column through the following expressions

---

EXTRACTING DAY OF WEEK

(DATEPART("dw",[Data Conversion datetime to date].DATE_) == 1 ? "Sunday" :
(DATEPART("dw",[Data Conversion datetime to date].DATE_) == 2 ? "Monday" :
(DATEPART("dw",[Data Conversion datetime to date].DATE_) == 3 ? "Tuesday" :
(DATEPART("dw",[Data Conversion datetime to date].DATE_) == 4 ? "Wednesday" :
(DATEPART("dw",[Data Conversion datetime to date].DATE_) == 5 ? "Thursday" :
(DATEPART("dw",[Data Conversion datetime to date].DATE_) == 6 ? "Friday" :
(DATEPART("dw",[Data Conversion datetime to date].DATE_) == 7 ? "Saturday" : "")))))))

EXTRACTING TYPE OF DAY: WEEKDAYS OR WEEKEND

(DATEPART("dw",[Data Conversion datetime to date].DATE_) == 1 ? "Weekend" :
(DATEPART("dw",[Data Conversion datetime to date].DATE_) == 2 ? "Weekday" :
(DATEPART("dw",[Data Conversion datetime to date].DATE_) == 3 ? "Weekday" :
(DATEPART("dw",[Data Conversion datetime to date].DATE_) == 4 ? "Weekday" :
(DATEPART("dw",[Data Conversion datetime to date].DATE_) == 5 ? "Weekday" :
(DATEPART("dw",[Data Conversion datetime to date].DATE_) == 6 ? "Weekday" :
(DATEPART("dw",[Data Conversion datetime to date].DATE_) == 7 ? "Weekend" : "")))))))

EXTRACTING QUARTER

(DATEPART("qq",[Data Conversion datetime to date].DATE_) == 1 ? "Q1" :
(DATEPART("qq",[Data Conversion datetime to date].DATE_) == 2 ? "Q2" :

(DATEPART("qq",[Data Conversion datetime to date].DATE_) == 3 ? "Q3" :
(DATEPART("qq",[Data Conversion datetime to date].DATE_) == 4 ? "Q4" : ""))))

EXTRACTING DAY OF MONTH

DAY([Data Conversion datetime to date].DATE_)

EXTRACTING YEAR

YEAR([Data Conversion datetime to date].DATE_)

EXTRACTING MONTH

MONTH([Data Conversion datetime to date].DATE_)

QUERY MOVIE DETAILS

SELECT film.film_id, film.title, film.description, film.release_year, film.original_language_id, film.rental_duration, film.rental_rate, film.length, film.rating, category.name

FROM film INNER JOIN

film_category ON film.film_id = film_category.film_id INNER JOIN

category ON film_category.category_id = category.category_id

QUERY FACT REVENUE

select

f.title,

i.inventory_id as DvdId ,

cat.name,

convert(date,r.rental_date) as rented_date,

ct.city,

co.country,

count(convert(date,r.rental_date)) as count_dvd_rented,

sum (p.amount) as dollarRented ,

from film f

```
inner join film_category fc
        on fc.film_id = f.film_id
inner join category cat
        on cat.category_id = fc.category_id
inner join inventory i
        on i.film_id = f.film_id
inner join rental r
        on r.inventory_id = i.inventory_id
inner join payment p
        on p.rental_id = r.rental_id
inner join customer c
        on c.customer_id = r.customer_id
inner join [dbo].[address] a
        on a.address_id = c.address_id
inner join city ct
        on ct.city_id = a.city_id
inner join country co
        on co.country_id = ct.country_id
group by
f.title, cat.name, i.inventory_id ,convert(date,r.rental_date), ct.city , co.country
order by f.title
```

QUERY FOR FACT RENTDVD_ACTOR

```
select cat.name as categoryname,
        (a.first_name +' '+a.last_name) as actorname,
        convert(date,(r.rental_date)) as date_rented,
```

```
        sum(p.amount) as rentIncome,
        count(convert(date,r.rental_date)) as times_rented
FROM category cat
inner join film_category fc
        on fc.category_id = cat.category_id
inner join film f
        on f.film_id = fc.film_id
inner join film_actor fa
        on fa.film_id = f.film_id
inner join actor a
        on a.actor_id = fa.actor_id
inner join inventory i
        on i.film_id = fa.film_id
inner join rental
        on r.inventory_id = i.inventory_id
inner join payment p
        on p.rental_id = r.rental_id
group by
 cat.name ,(a.first_name +' '+a.last_name) ,  convert(date,r.rental_date)
```

APPENDIX C- XML AND XSD

Xsd: revenue.xsd

```
<?xml version="1.0"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="fact_revenues">

<xsd:complexType>
<xsd:sequence>

<xsd:element ref="movie" minOccurs="1" maxOccurs="unbounded" />
<xsd:element ref="category" minOccurs="1" maxOccurs="unbounded" />
<xsd:element ref="region" minOccurs="1" maxOccurs="unbounded" />
<xsd:element ref="revenue" minOccurs="1" maxOccurs="unbounded" />
</xsd:sequence>
</xsd:complexType>

<xsd:key name="movie_key">
<xsd:selector xpath="movie/movie_key" />
<xsd:field xpath="." />
</xsd:key>
<xsd:keyref name="movie_key_ref" refer="movie_key">
<xsd:selector xpath="revenue/employee_id" />
```

```
<xsd:field xpath="." />
</xsd:keyref>

<xsd:key name="category_key">
<xsd:selector xpath="category/category_key" />
<xsd:field xpath="." />
</xsd:key>
<xsd:keyref name="category_key_ref" refer="category_key">
<xsd:selector xpath="revenue/customer_id" />
<xsd:field xpath="." />
</xsd:keyref>

<xsd:key name="region_key">
<xsd:selector xpath="region/region_key" />
<xsd:field xpath="." />
</xsd:key>
<xsd:keyref name="region_key_ref" refer="region_key">
<xsd:selector xpath="revenue/region_key" />
<xsd:field xpath="." />
</xsd:keyref>
</xsd:element>

<xsd:element name="movie">
<xsd:complexType>
<xsd:sequence>
```

<xsd:element name="movie_key" type="xsd:int" />

<xsd:element name="movie_name" type="xsd:string" />

<xsd:element name="movie_length" type="xsd:string" />

<xsd:element name="movie_release_date" type="xsd:string" />

<xsd:element name="movie_rating" type="xsd:string" />

<xsd:element name="movie_category" type="xsd:string" />

<xsd:element name="movie_id" type="xsd:int" />

</xsd:sequence>

</xsd:complexType>

</xsd:element>

<xsd:element name="category">

<xsd:complexType>

<xsd:sequence>

<xsd:element name="category_key" type="xsd:int" />

<xsd:element name="category_name" type="xsd:string" />

</xsd:sequence>

</xsd:complexType>

</xsd:element>

<xsd:element name="region">

<xsd:complexType>

```
<xsd:sequence>
<xsd:element name="region_key" type="xsd:int" />
<xsd:element name="city" type="xsd:string" />
<xsd:element name="country" type="xsd:string" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>


<xsd:element name="revenue">
<xsd:complexType>
<xsd:all>
<xsd:element name="movie_key" type="xsd:int" />
<xsd:element name="category_key" type="xsd:int" />
<xsd:element name="region_key" type="xsd:int" />
<xsd:element name="dollarrent" type="xsd:decimal" />
</xsd:all>
</xsd:complexType>
</xsd:element>

</xsd:schema>
```

Xml:revenue_xml.xml

```
<?xml version="1.0"?>
<fact_revenues xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="REVENUE.xsd">

<movie>
<movie_key>1</movie_key>
<movie_name>ACADEMY DINASOURS</movie_name>
<movie_length>120</movie_length>
<movie_release_date>25-09-2013</movie_release_date>
<movie_rating>UG</movie_rating>
<movie_category>Documentry</movie_category>
<movie_id>1</movie_id>
</movie>
<movie>
<movie_key>2</movie_key>
<movie_name>rambo</movie_name>
<movie_length>120</movie_length>
<movie_release_date>25-09-2013</movie_release_date>
<movie_rating>UG</movie_rating>
<movie_category>Documentry</movie_category>
<movie_id>1</movie_id>
</movie>
```

<category>

<category_key>1</category_key>

<category_name>Documentry</category_name>

</category>

<category>

<category_key>2</category_key>

<category_name>Documentry</category_name>

</category>

<region>

<region_key>1</region_key>

<city>KualaLumpur</city>

<country>Singapoor</country>

</region>

<region>

<region_key>2</region_key>

<city>KualaLumpur</city>

<country>Singapoor</country>

</region>

<revenue>

<movie_key>1</movie_key>

<category_key>1</category_key>

<region_key>1</region_key>

<dollarrent>599.99</dollarrent>

</revenue>

<revenue>

<movie_key>2</movie_key>

<category_key>2</category_key>

<region_key>2</region_key>

<dollarrent>599.99</dollarrent>

</revenue>

</fact_revenues>

APPENDIX D– NEO4J QUERIES

Loading table fact revenue from csv to neo4j.

LOAD CSV WITH HEADERS FROM "file:///factrevenue.csv"

AS row create (fr:FACT_REVENUE)

set fr = row {

MOVIEKEY:row.

MOVIEKEY,

DVDKEY:row.DVDKEY,

CATEGORYKEY:row.

CATEGORYKEY,

REGIONKEY:row.REGIONKEY,

DATEKEY:row.DATEKEY,

DOLLARRENT:row.DOLLARRENT

}

return fr

Similarly loaded other dimensional tables: category, date, movie, region

---

Creating constraint:

```
CONSTRAINT ON ( category:Category ) ASSERT category.CATEGORYKEY IS UNIQUE

CONSTRAINT ON ( date:Date ) ASSERT date.DATEKEY IS UNIQUE

CONSTRAINT ON ( movie:Movie ) ASSERT movie.MOVIEKEY IS UNIQUE

CONSTRAINT ON ( region:Region ) ASSERT region.REGIONKEY IS UNIQUE
```

Create relation between movie label nodes and category label nodes

MATCH (mv:Movie), (ct:Category)

WHERE mv.CATEGORY = ct.CATEGORYNAME

CREATE (mv)-[r:has_category]->(ct)

RETURN mv,ct

---

to create relation between label fact revenue nodes and label category nodes

match(fr:FACT_REVENUE),(ct:Category)

where fr.CATEGORYKEY = ct.CATEGORYKEY

create(fr)-[r:has_category]->(ct)

return fr,ct,r

Create relation between label fact revenue nodes and label category nodes

---

match(fr:FACT_REVENUE), (ct:Category)

where fr.CATEGORYKEY = ct.CATEGORYKEY

create(fr)-[r:has_category]->(ct)

return fr,ct,r

Create relation between label fact revenue nodes and label date nodes

---

match (fr:FACT_REVENUE), (dt:Date)

where fr.DATEKEY = dt.DATEKEY

create(fr)-[r:has_date]->(dt)

return fr,dt,r

Create relation between label fact revenue nodes and label category nodes

---

Match (mv:Movie), (dt:Date), (fr:FACT_REVENUE)

Where (mv.MOVIEKEY= fr.MOVIEKEY

and

dt.DATEKEY = mv.DATEKEY)

create (mv)-[r:rented_on]-(dt) return

to create relation “rented on dates” between all movies nodes and date nodes

match (fr:FACT_REVENUE), (c:Category), (d:Date), (mv:Movie), (rg:Region)

where (fr.CATEGORYKEY= c.CATEGORYKEY

AND fr.MOVIEKEY = mv.MOVIEKEY

and

d.DATEKEY = fr.DATEKEY,

rg.REGIONKEYREG = fr.REGIONKEY)

create (mv)-[r:rented_on_date]-(d)

Display all movie and category nodes related to each

match (c:Category), (mv:Movie)

where c.CATEGORYNAME = mv.CATEGORY

create (mv)-[r:of_category]->(c)

return c,mv,r

Create relation between all movies nodes and region nodes

Match (c:Category), (mv:Movie), (reg:Region), (fr:FACT_REVENUE)

where fr.MOVIEKEY= mv.MOVIEKEY

and reg.REGIONKEY = fr.REGIONKEY

create (mv)-[r:rented_in]->(reg)

return r,reg,mv limit(25)

Display nodes with relation to each other

match (fr:FACT_REVENUE), (reg:Region), (mv:Movie), (c:Category), (d:Date)

```
 where fr.MOVIEKEY= mv.MOVIEKEY
and toInt(fr.MOVIEKEY)= 1 and reg.REGIONKEY= fr.REGIONKEY
and d.DATEKEY = fr.DATEKEY
and toInt(fr.DATEKEY) in [23,10,29]
  and c.CATEGORYKEY = fr.CATEGORYKEY
return fr, reg, mv, c, d limit(50)
```

APPENDIX E– R VISUALIZATION CODE

```
#RCODE ASSIGNMENT REPORT ONE

#install.packages("RODBC")

require(RODBC)

#install.packages("zoom")

library(zoom)

#install.packages("dplyr")

library(dplyr)

#install.packages("tidyr")

library(tidyr)

#install.packages("reshape2")

library(reshape2)

library(randomcoloR)

#zm()

#creaing connection

conn = odbcDriverConnect('driver={SQL Server};server=.\\SQLEXPRESS;

                         database=SakilaDBS; trusted_connection=true;')

conn


################################ available movies by actors
#####################################################

#bar plot

queryMoviesByAcctor ="select da.ACTORNAME ,count(da.MOVIEID) as
TOTLMOVIESBYACTOR

from DIM_ACTOR da

GROUP BY DA.ACTORNAME"
```

```
MoviesByActorData =sqlQuery(conn, queryMoviesByAcctor)

summary(data.frame(MoviesByActorData))

length(MoviesByActorData$TOTLMOVIESBYACTOR)



n <- length(MoviesByActorData$ACTORNAME)

palette <- distinctColorPalette(n)


#limiting xlim = 50 bins

barplot(MoviesByActorData$TOTLMOVIESBYACTOR, main= "Available No of
Movies by Actor",

    ylab="Number of Movies",

    xlab="Actor Names",

    col= palette,

    names.arg = MoviesByActorData$ACTORNAME,

    cex.axis = 1,

    cex.names = 0.6,

    horiz = FALSE,

    las= 2,

    ylim = c(0,60),

    xlim = c(0,50)


    )
```

```
zm()


#install.packages("ggplot")


########################################## 2.total dvds by actor
############

#bar plots

queryActorDvdCounts = "SELECT da.ACTORNAME, count(dd.DVDID) AS
TOTALDVDS

from DIM_ACTOR da

        inner join DIM_MOVIE dm

        on dm.MOVIEID = da.MOVIEID

        inner join DIM_DVD dd

        on dd.MOVIEID = dm.MOVIEID

        group by da.ACTORNAME

        order by da.ACTORNAME"



dataActorDvdCounts= sqlQuery(conn, queryActorDvdCounts)

summary(data.frame(dataActorDvdCounts))


#length(MoviesByActorData$TOTLMOVIESBYACTOR)


barplot(dataActorDvdCounts$TOTALDVDS, main= " No of DVDS by Actor",

    ylab="NO OF DVDS",
```

```
    xlab="Actor Names",
    col= palette,
    names.arg = dataActorDvdCounts$ACTORNAME,
    cex.axis = 1,
    cex.names = 0.6,
    horiz = FALSE,
    las= 2,
    xlim = c(0,50)

)
zm()

############################################################ 3.earnings
by date #####
library(ggplot2)
theme_set(theme_minimal())

querySTring = "select dd.FullDate, sum(fr.DOLLARRENT) as EARNINGRENT
 from FACT_REVENUE fr
 inner join DIM_DATE dd
 on dd.DATEKEY= fr.DATEKEY
 group by dd.FullDate
"
data = sqlQuery(conn, querySTring)
```

```
# Basic line plot

ggplot(data = data, aes(x = FullDate, y = EARNINGRENT, group = 1))+

  geom_line(color = "#00FF00", size = 2)+

theme(plot.title = element_text(hjust = 0.5),axis.text.x = element_text(angle = 90,
hjust = 1))+

  labs(title="Earnings By Date",x ="Date", y = "Earnings")


# Plot a subset of the data

zm()


################################ 4.RENT EARNED BY DATE AND
CATEGORY ##################


# qrySTring = "select  dd.full, dc.CATEGORYNAME, sum(fr.DOLLARRENT) as
RENTEARNED

#  from FACT_REVENUE fr

#  inner join DIM_DATE dd

#  on dd.DATEKEY= fr.DATEKEY

#  inner join DIM_CATEGORY dc

#  on dc.CATEGORYKEY = fr.CATEGORYKEY

#  group by dd.Month_, dd.Year_, dc.CATEGORYNAME

# "


data = sqlQuery(conn, "select  dd.Month_, dc.CATEGORYNAME,
sum(fr.DOLLARRENT) as RENTEARNED

 from FACT_REVENUE fr

            inner join DIM_DATE dd
```

```
        on dd.DATEKEY= fr.DATEKEY
        inner join DIM_CATEGORY dc
        on dc.CATEGORYKEY = fr.CATEGORYKEY
        group by dd.Month_, dc.CATEGORYNAME
        ")


ggplot(data, aes(x = Month_, y = RENTEARNED)) +
  geom_area(aes(color = CATEGORYNAME, fill = CATEGORYNAME),
            alpha = 0.5, position = position_dodge(0.8)) +
  scale_color_manual(values = palette) +
  scale_fill_manual(values = palette)+
theme(plot.title = element_text(hjust = 0.5),axis.text.x = element_text(angle = 90,
hjust = 1))+
  labs(title="Monthly Earnings By Category",x ="Month", y = "Earnings")
zm()
```