



```

FROM sakila.film) AS p1
INNER JOIN
(SELECT a.film_id, name as category
FROM sakila.film_category AS a, sakila.category AS b
WHERE a.category_id = b.category_id) AS p2
ON p1.film_id = p2.film_id);

```

Note: the semantics of rating between the sequel movie and sakila databases do not match (PG rating *vs* presumable movie rating [1-5]). Due to the lack of further information on the disambiguation criteria the two were joined under the same dimension in the view, and further ahead in the data warehouse and all depending data.

### **all\_actors(actor\_id, first\_name, last\_name)**

```

CREATE VIEW all_actors(actor_id, first_name, last_name) AS
(SELECT CONCAT("m_",personID) as actor_id, personFirstName AS first_name, personLastName AS
FROM
(SELECT personID, personFirstName, personLastName
FROM sequelmovie.person) AS a
INNER JOIN
(SELECT p_personID, m_movieID
FROM sequelmovie.role AS r
WHERE r.roleDesc like '%Actor%'
) AS b
ON a.personID = b.p_personID)
UNION
(SELECT CONCAT("s_",actor_id), first_name, last_name
FROM sakila.actor);

```

Both kind of actors are captured by the query from the sequel movie database. In the sakila database actors can only be film actors thus no disambiguation is needed.

### **all\_film\_act(film\_id, actor\_id) film\_id: FK(all\_films) actor\_id: FK(all\_actors)**

```

CREATE VIEW all_film_act(film_id, actor_id) AS
(SELECT film_id,actor_id from
(SELECT film_id from all_films) AS a
INNER JOIN
(SELECT * FROM all_actors as a, sequelmovie.role AS r
WHERE a.actor_id =  CONCAT("m_",r.p_personID)) AS b
ON CONCAT("m_",b.m_movieID) = a.film_id)
UNION
(SELECT a.film_id, actor_id FROM
(SELECT film_id from all_films) AS a
INNER JOIN
(SELECT CONCAT("s_",film_id) AS film_id, CONCAT("s_",r.actor_id) AS actor_id FROM all_actors
WHERE a.actor_id = CONCAT("s_",r.actor_id)) AS b
ON b.film_id = a.film_id);

```

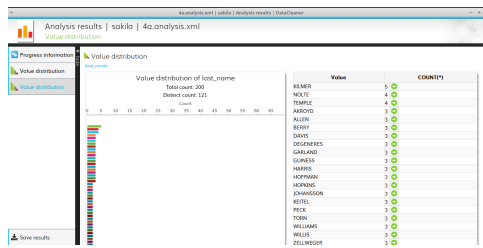
### Question 3

Query: film titles with time  $\geq 60$  mins and category = Science Fiction

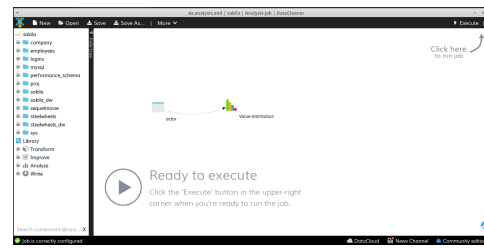
```
SELECT r.first_name, r.last_name, e.title
FROM all_actors AS r, all_film_act AS d, all_films AS e
WHERE e.length  $\geq$  60
AND (category='Science Fiction' OR category ='Sci-Fi')
AND r.actor_id = d.actor_id
AND d.film_id = e.film_id;
```

In the new data source, due to the fact that the rating attribute is ambiguous (either a [1,5] rating or parental guidance rating, namely numeric and categorical) we can state that the data is not homogeneous, as in records do not contain similar/equal information, thus not representing the same entity.

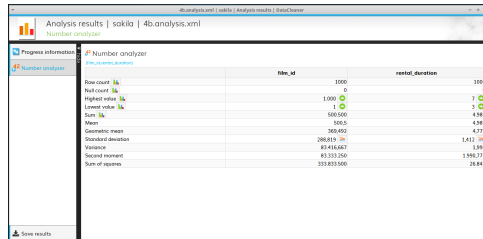
## Question 4



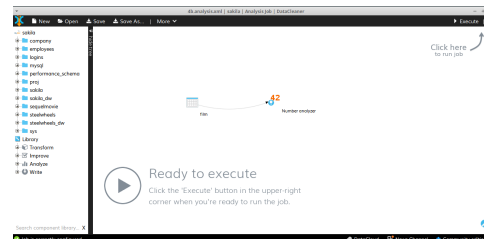
(a)



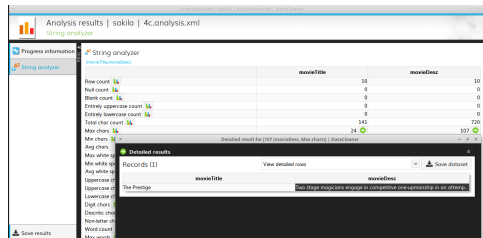
(b)



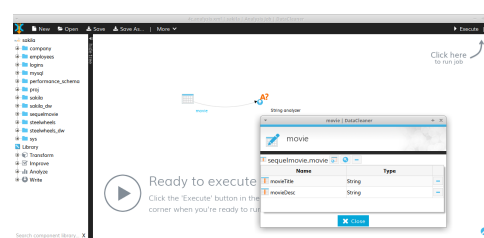
(c)



(d)



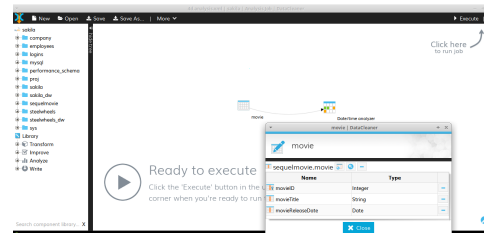
(e)



(f)



(g)



(h)

Figure 2: Figures a) and c) correspond to their question's respective query. In pictures e) and g) despite showing sakila at the top it can be seen that the loading of the table (figs f and h) are done from the sequel movie database. Furthermore, the movie with the longest description in figure d) is "The Prestige" and the date range in figure f) is approx. 45 years (for the entire database)

## Question 5

```
DROP DATABASE IF EXISTS sakila_dw;  
CREATE DATABASE sakila_dw;  
USE sakila_dw;
```

```
CREATE TABLE D_Film (  
  film_id INT,  
  filmid INT,  
  title VARCHAR(255),  
  category VARCHAR(255),  
  PRIMARY KEY (film_id)  
);
```

```
CREATE TABLE D_Customer (  
  customer_id INT,  
  customerid INT,  
  last_name VARCHAR(255),  
  city VARCHAR(255),  
  country VARCHAR(255),  
  VERSION INT,  
  DATE_FROM DATETIME,  
  DATE_TO DATETIME,  
  PRIMARY KEY (customer_id)  
);
```

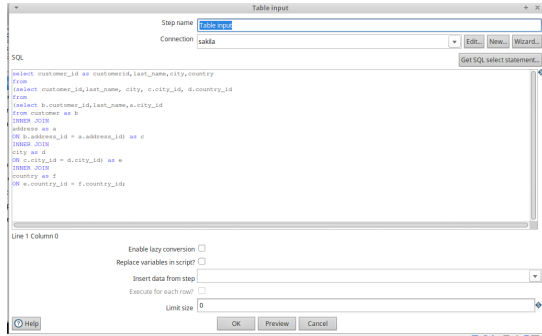
```
CREATE TABLE D_Time (  
  time_id DATETIME,  
  timeid INT NOT NULL AUTO_INCREMENT,  
  day INT,  
  month_id INT,  
  MONTH_NAME VARCHAR(255),  
  year INT,  
  PRIMARY KEY (timeid)  
);
```

```
CREATE TABLE fact_sales (  
  rental_id INT,  
  rentalid INT,  
  measure DOUBLE,  
  customer_id INT,  
  film_id INT,  
  time_id INT,  
  PRIMARY KEY (rental_id),  
  FOREIGN KEY (customer_id) REFERENCES D_Customer (customer_id),  
  FOREIGN KEY (film_id) REFERENCES D_Film (film_id),  
  FOREIGN KEY (time_id) REFERENCES D_Time (timeid)  
);
```

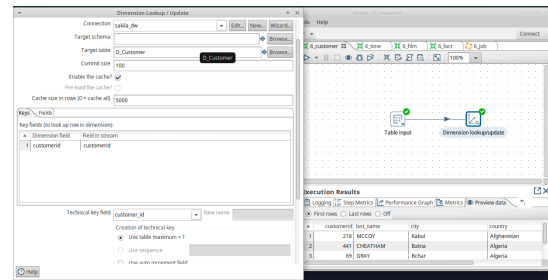
## Question 6

The following subsection contains the screenshots, as required, for each dimension. Some solutions might be overly complex due to the fact of possible ambiguities within the database such as the time to be associated with a fact (rental? payment? return?). Furthermore, while decomposing the dates into single elements (calculator step in 5) a series of null fields were generated and had to be filtered out prior to insertion, while considering all the previous date origins.

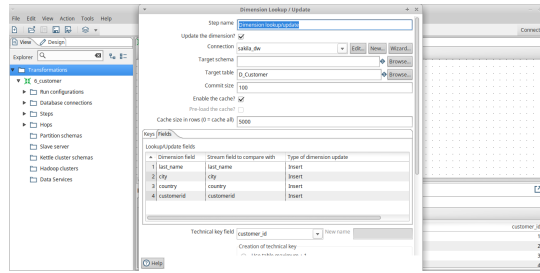
### Customer



(a)



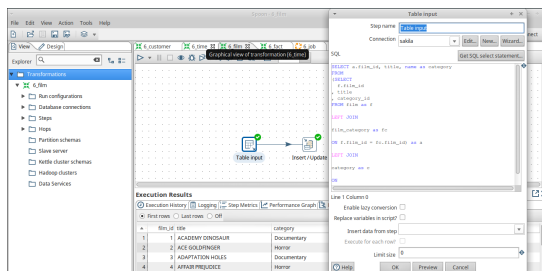
(b)



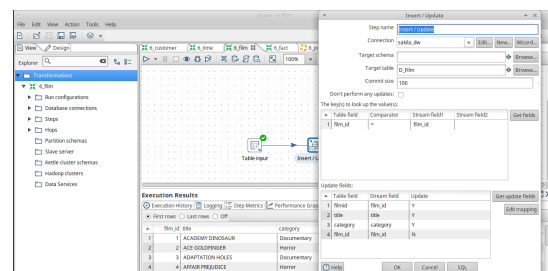
(c)

Figure 3: D\_Customer

### Film



(a)



(b)

Figure 4: D\_Film

Time

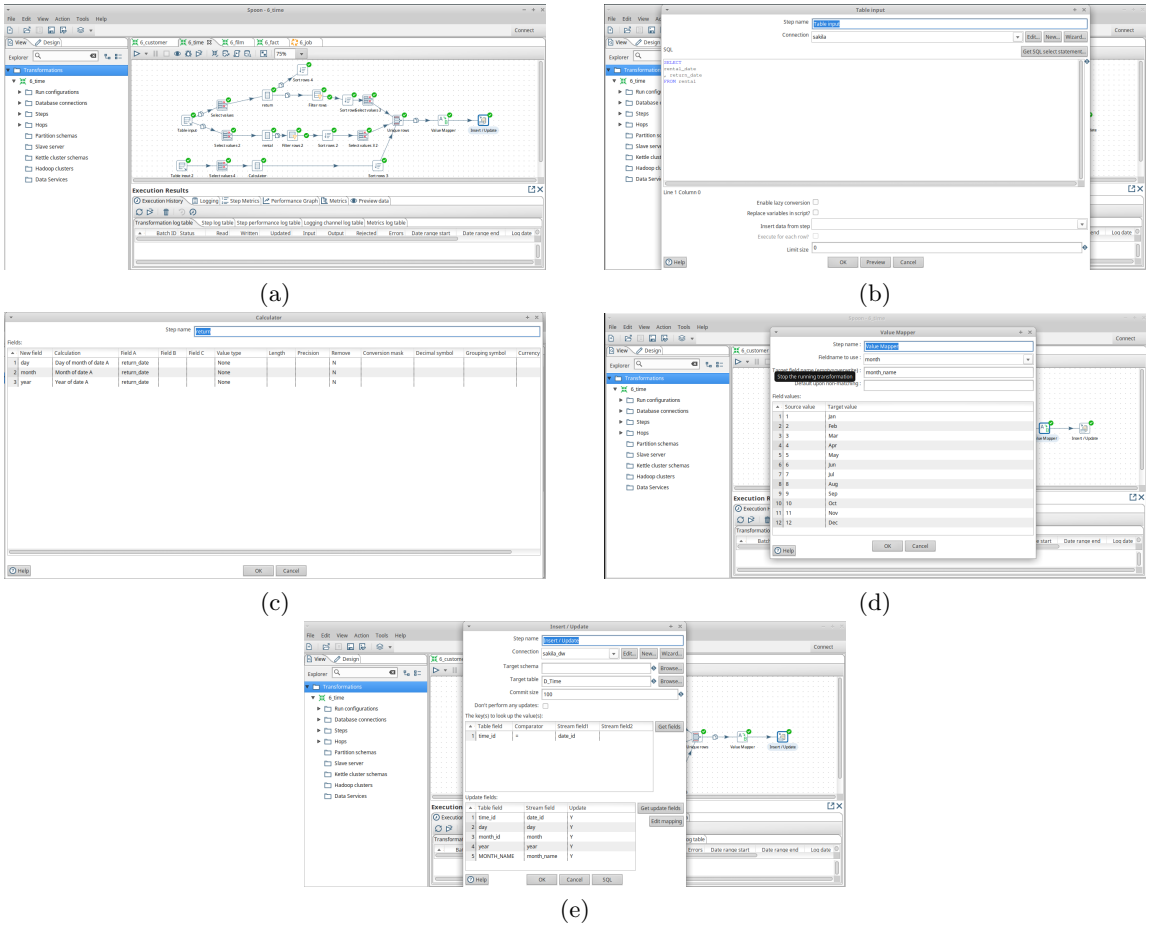
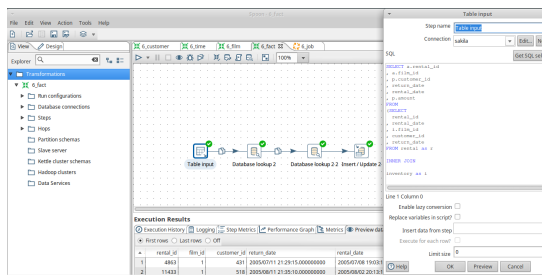
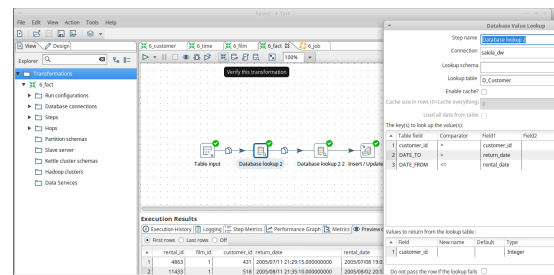


Figure 5: D\_Time

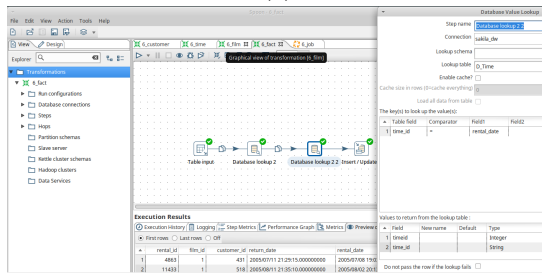
# Fact Rentals



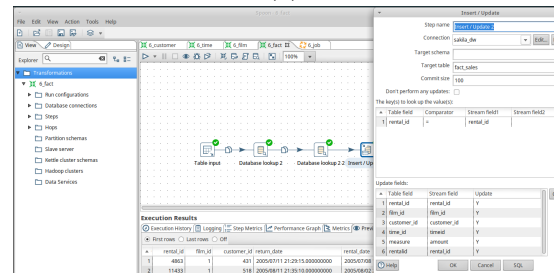
(a)



(b)



(c)



(d)

Figure 6: fact\_sales

# Job

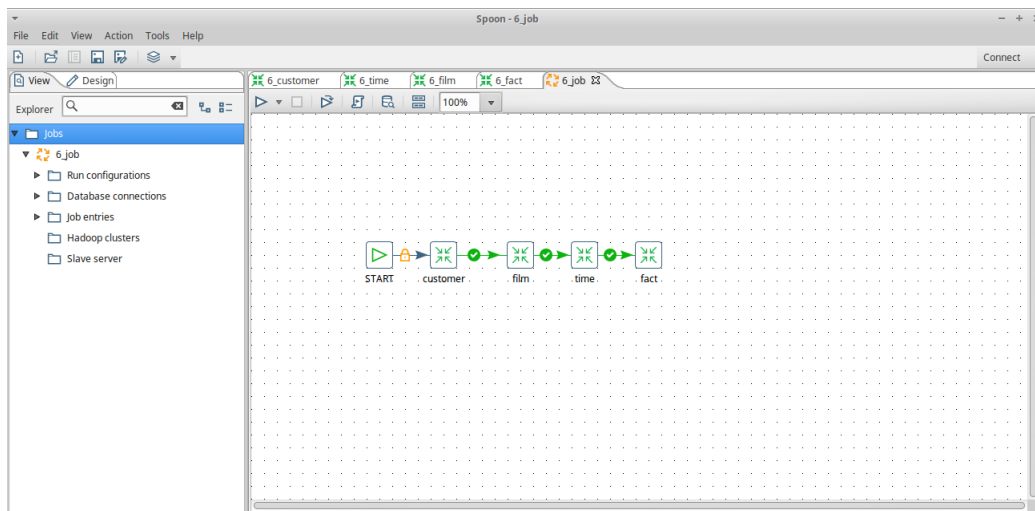


Figure 7: Final job



## Question 7

7a)

Question: total rental payment amount.

Query:

```
select sum(measure) from fact_sales;
```

7b)

Question: total rental payment amount by category name.

Query:

```
SELECT category, sum(a.measure)
FROM
(select measure,category from D_Film as df
INNER JOIN
fact_sales as fs
ON
fs.film_id = df.film_id) as a
GROUP BY
a.category;
```

7c)

The following query takes into account what client was active during the rental corresponding to the date range selection in the *WHERE* clause.

Question: total rental payment amount by category name and by city.

Query:

```
SELECT category,city,sum(measure)
FROM
(SELECT dc.customer_id,dc.city,dc.DATE_FROM, dc.DATE_TO, b.measure,b.category,b.time_id
FROM
(select customer_id, city, DATE_FROM, DATE_TO
FROM D_Customer) as dc
INNER JOIN
(select fs.customer_id, measure,category,fs.time_id from D_Film as df
INNER JOIN
fact_sales as fs
ON
fs.film_id = df.film_id) as b
ON
dc.customer_id = b.customer_id) as g
INNER JOIN
(SELECT time_id, timeid FROM D_Time) as h
ON g.time_id = h.timeid
WHERE DATE_FROM <= h.time_id
and DATE_TO > h.time_id
```

```
GROUP BY
category,city;
```

## Question 8

Query:

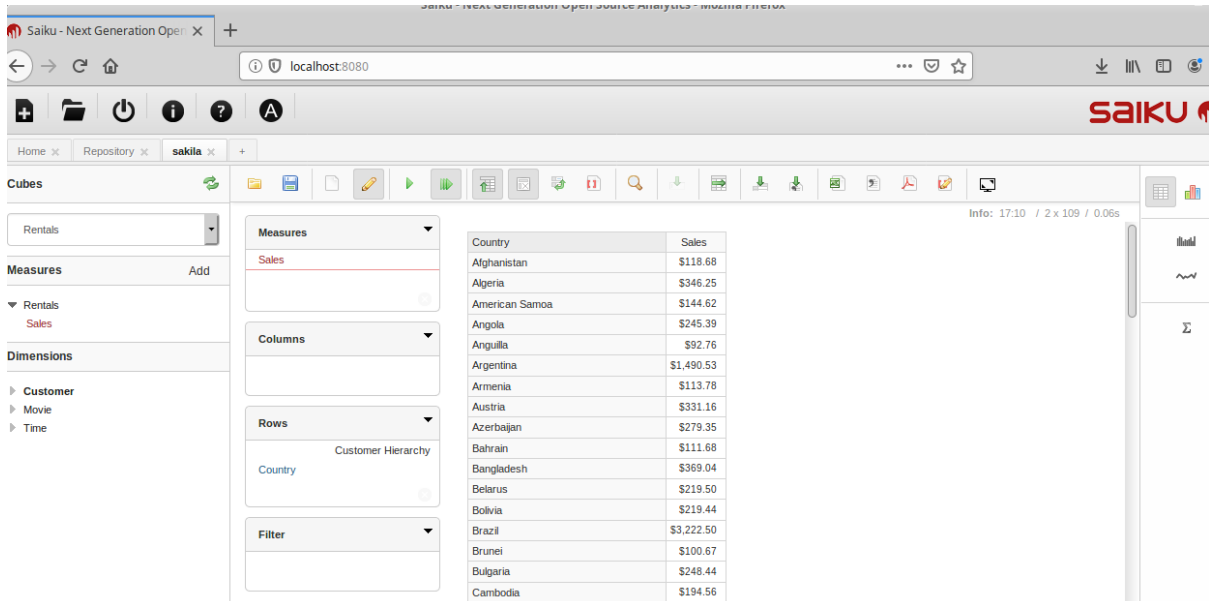
```
SELECT category,city,sum(measure)
FROM
(SELECT dc.customer_id,dc.city,dc.DATE_FROM, dc.DATE_TO, b.measure,b.category,b.time_id
FROM
(select customer_id, city, DATE_FROM, DATE_TO
FROM D_Customer) as dc
INNER JOIN
(select fs.customer_id, measure,category,fs.time_id from D_Film as df
INNER JOIN
fact_sales as fs
ON
fs.film_id = df.film_id) as b
ON
dc.customer_id = b.customer_id) as g
INNER JOIN
(SELECT time_id, timeid FROM D_Time) as h
ON g.time_id = h.timeid
WHERE DATE_FROM <= h.time_id
and DATE_TO > h.time_id
GROUP BY
category,city WITH ROLLUP;
```

## Question 9

In zip file.

## Question 10

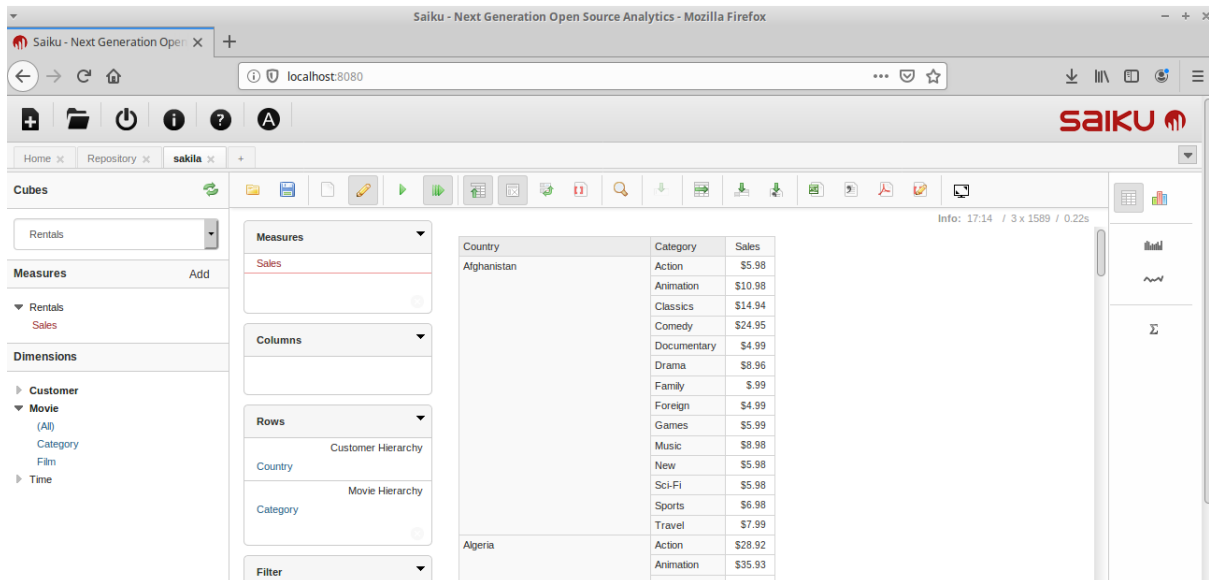
10a)



The screenshot shows the Saiku web application interface. The 'Cubes' panel on the left shows the 'Rentals' cube selected. The 'Measures' panel shows 'Sales' selected. The 'Dimensions' panel shows 'Customer', 'Movie', and 'Time' selected. The main view displays a table of sales data by country.

| Country        | Sales      |
|----------------|------------|
| Afghanistan    | \$118.68   |
| Algeria        | \$346.25   |
| American Samoa | \$144.62   |
| Angola         | \$245.39   |
| Anguilla       | \$92.76    |
| Argentina      | \$1,490.53 |
| Armenia        | \$113.78   |
| Austria        | \$331.16   |
| Azerbaijan     | \$279.35   |
| Bahrain        | \$111.68   |
| Bangladesh     | \$369.04   |
| Belarus        | \$219.50   |
| Bolivia        | \$219.44   |
| Brazil         | \$3,222.50 |
| Brunei         | \$100.67   |
| Bulgaria       | \$248.44   |
| Cambodia       | \$194.56   |

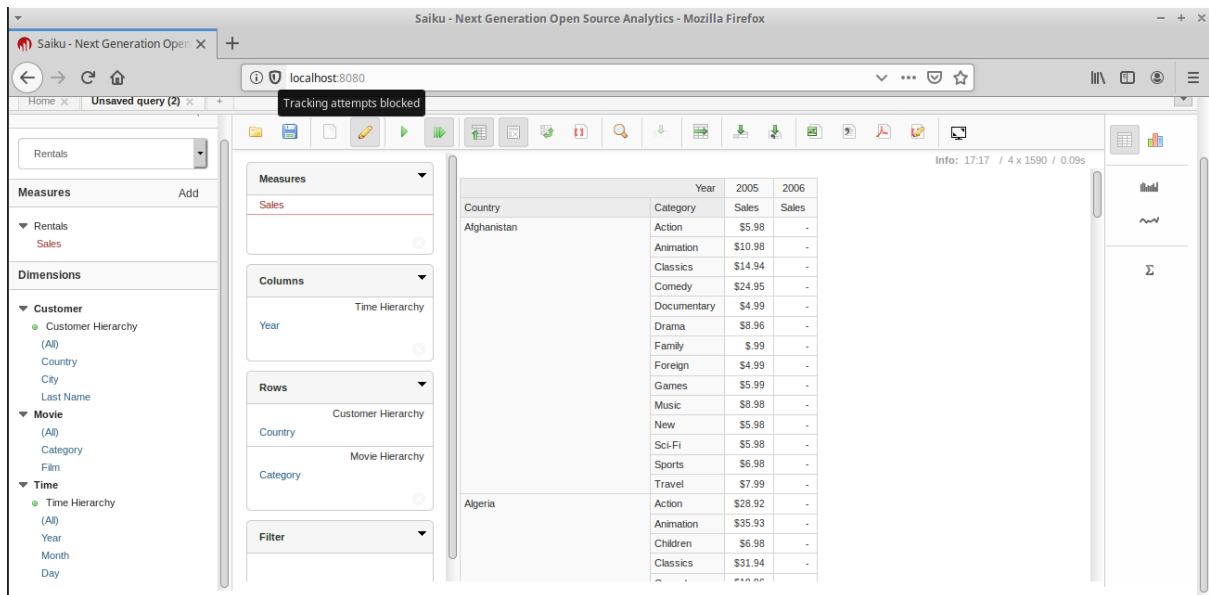
10b)



The screenshot shows the Saiku web application interface. The 'Cubes' panel on the left shows the 'Rentals' cube selected. The 'Measures' panel shows 'Sales' selected. The 'Dimensions' panel shows 'Customer', 'Movie', and 'Time' selected. The main view displays a table of sales data by country and category.

| Country     | Category    | Sales   |
|-------------|-------------|---------|
| Afghanistan | Action      | \$5.98  |
|             | Animation   | \$10.98 |
|             | Classics    | \$14.94 |
|             | Comedy      | \$24.95 |
|             | Documentary | \$4.99  |
|             | Drama       | \$8.96  |
|             | Family      | \$9.99  |
|             | Foreign     | \$4.99  |
|             | Games       | \$5.99  |
|             | Music       | \$8.98  |
|             | New         | \$5.98  |
|             | Sci-Fi      | \$5.98  |
|             | Sports      | \$6.98  |
|             | Travel      | \$7.99  |
| Algeria     | Action      | \$28.92 |
|             | Animation   | \$35.93 |
|             | Children    | \$6.98  |

10c)



## Question 11

**MDXQuery** for: The rental amount growth per month (include previous month) for each South America country existing in the database (i.e., Argentina, Bolivia, Brazil, Chile, Colombia, Ecuador, French Guiana, Paraguay, Peru, and Venezuela).

Query:

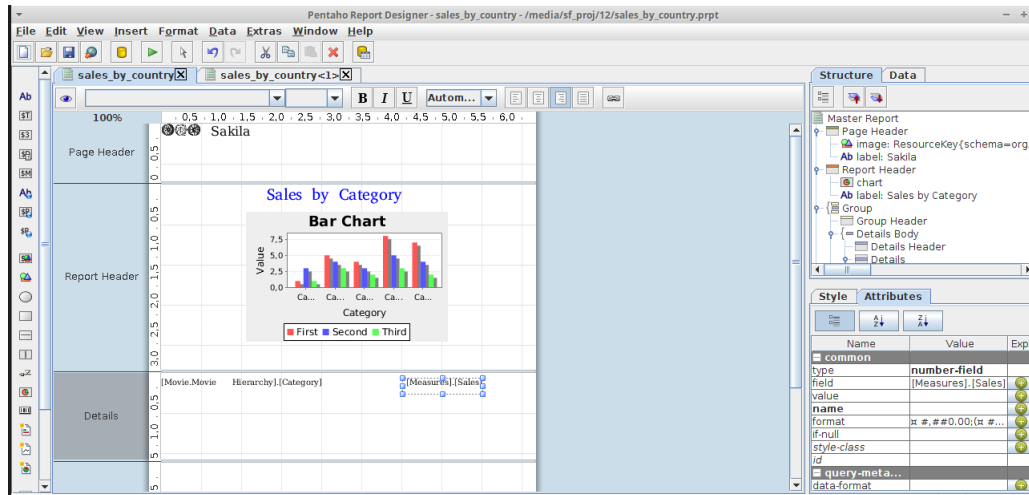
```
WITH
SET [South America Countries] AS {[Customer].[Country].[Argentina],
[Customer].[Country].[Bolivia],
[Customer].[Country].[Brazil],
[Customer].[Country].[Chile],
[Customer].[Country].[Colombia],
[Customer].[Country].[Ecuador],
[Customer].[Country].[Paraguay],
[Customer].[Country].[Peru],
[Customer].[Country].[Venezuela],
[Customer].[Country].[French Guiana]}
MEMBER Measures.[Previous Month] AS Time.Month.CurrentMember.PrevMember
MEMBER Measures.[Sales Growth] AS Measures.Sales - Measures.[Previous Month]
SELECT {Measures.Sales,
Measures.[Previous Month],
Measures.[Sales Growth]} ON COLUMNS,
[South America Countries] * {DRILLDOWNLEVEL(Time.Year.Members)} ON ROWS
FROM Rentals
```

The screenshot shows the Saiku web application interface. The top navigation bar includes the Saiku logo and a search bar. The main content area displays an MDX query in the top pane and its results in the bottom pane. The query is the same as the one provided in the question. The results pane shows a table with columns: Country, Year, Month, Sales, Previous Month, and Sales Growth. The data is grouped by Country (Argentina and Bolivia) and Year (2005 and 2006). The table shows the following data:

| Country   | Year | Month | Sales      | Previous Month | Sales Growth |
|-----------|------|-------|------------|----------------|--------------|
| Argentina | 2005 | May   | \$1,472.59 | -              | \$1,472.59   |
|           |      | Jun   | \$136.66   | -              | \$136.66     |
|           |      | Jul   | \$233.53   | 137            | \$96.87      |
|           |      | Aug   | \$544.69   | 234            | \$311.16     |
|           |      | Sep   | \$557.71   | 545            | \$13.02      |
|           |      | Oct   | -          | 558            | -\$557.71    |
|           |      | Nov   | \$17.94    | 1,473          | -\$1,454.65  |
|           |      | Dec   | \$17.94    | -              | \$17.94      |
|           |      | Jan   | \$219.44   | -              | \$219.44     |
| Bolivia   | 2005 | May   | \$10.98    | -              | \$10.98      |
|           |      | Jun   | \$29.91    | 11             | \$18.93      |
|           |      | Jul   | \$102.75   | 90             | \$12.75      |

## Question 12

### Design Mode



### Preview Mode

