Athens University of Economics and Business
Department of Management Science and Technology

# Project Report

Module: Business Intelligence and Big Data Analytics

Students:
Danai Kape (8160038, danaykap13@gmail.com)
Eirini Kolimatsi (8160047, irinikolimatsh@gmail.com)

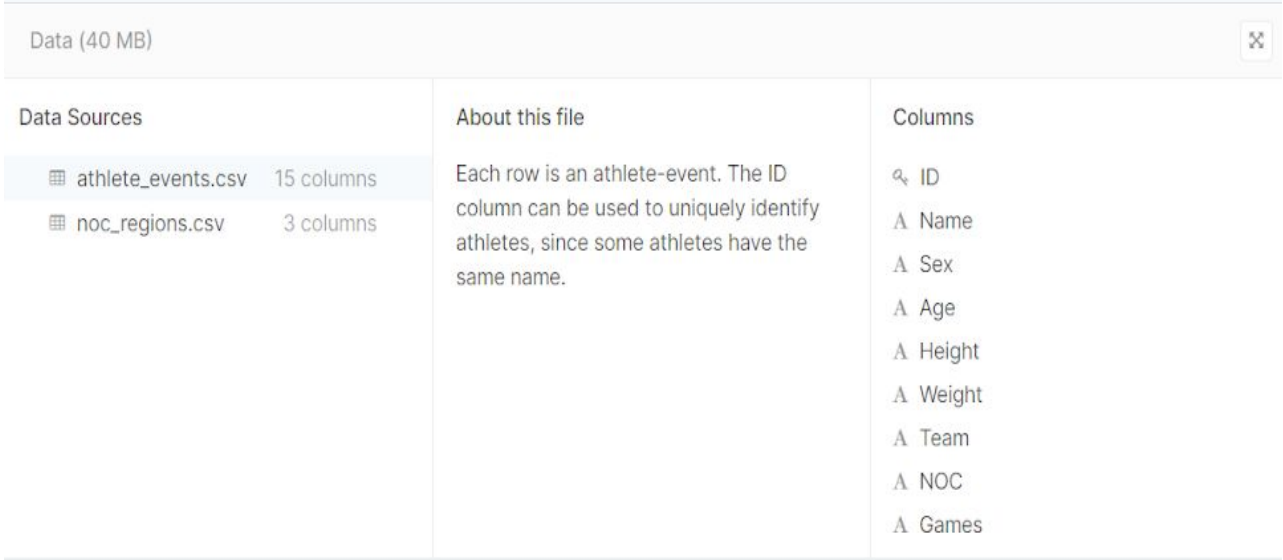
Athens, January 2020

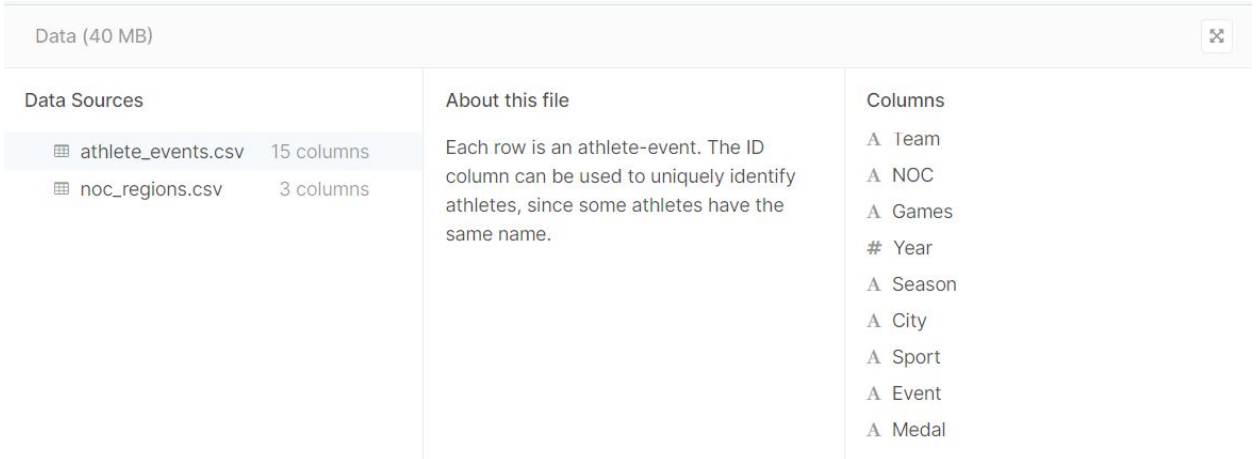
# Table of Contents

# 1. Data

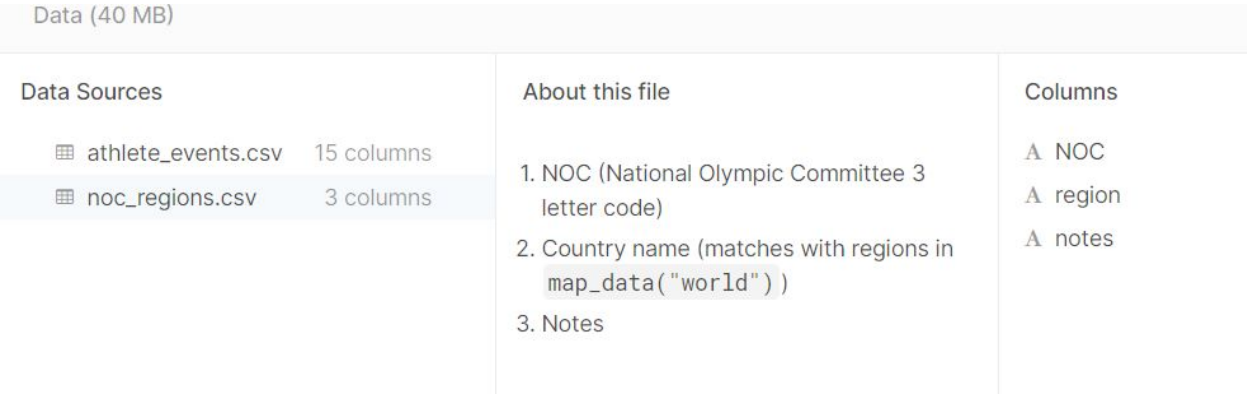
The source of the main data used for this project is Kaggle and can be found in the following link: https://www.kaggle.com/heesoo37/120-years-of-olympic-history-athletes-and-results
It contains 2 csv files (athlete_events.csv and noc_regions.csv).

- **athlete_events.csv** :



- **Noc_regions.csv :**




For the purposes of our data analysis, we needed to have for every country, existing or former country, the continent and the sub region that belongs to. Hence, we looked around the web for such a dataset but we didn't find exactly what we were in need of. Based on this dataset: https://github.com/lukes/ISO-3166-Countries-with-Regional-Codes/blob/master/all/all.csv we updated the alpha-3 codes that were different from the codes that are allocated from the International Olympic Committee (IOC), the countries that have some differences on their names, eg "Bahamas" vs "Bahamas, the" and most importantly added former countries, their IOC country codes, continent, sub-region and intermediate region. To add this info we conducted manual internet search for all the cases that we needed data. Furthermore, it should be noted that in the dataset there are 3 cases that are not related with former countries, but are teams for refugee participants (Refugee Olympic Team) or don't identify as participants of a country (Individual Olympic Athletes) or participants whose country is not known (NA). The final version of this dataset can be found on our project's Github repository, in the following link: https://github.com/Danaykap/Olympics-Data-Mining-Project/blob/master/data/Areas.csv

Furthermore, we used data for the host countries that show year, season, city, country, longitude and latitude for all Olympics. It can be found on the following link: https://github.com/rgriff23/Olympic_history_v2/blob/master/Data/d_hostcity.csv

Finally, we used data containing the longitude and latitude of the participant countries. For former countries, we considered them as the country that is right now on their geographic area, eg Crete was considered as Greece and this part of data was manually added to the

dataset. The original dataset can be found on Kaggle through: https://www.kaggle.com/folaraz/world-countries-and-continents-details#Countries%20Longitude%20and%20Latitude.csv

# 2. Importing Data

To import the data to an SQL Server Server database we used Azure Data Studio and screenshots from this process can be found below.

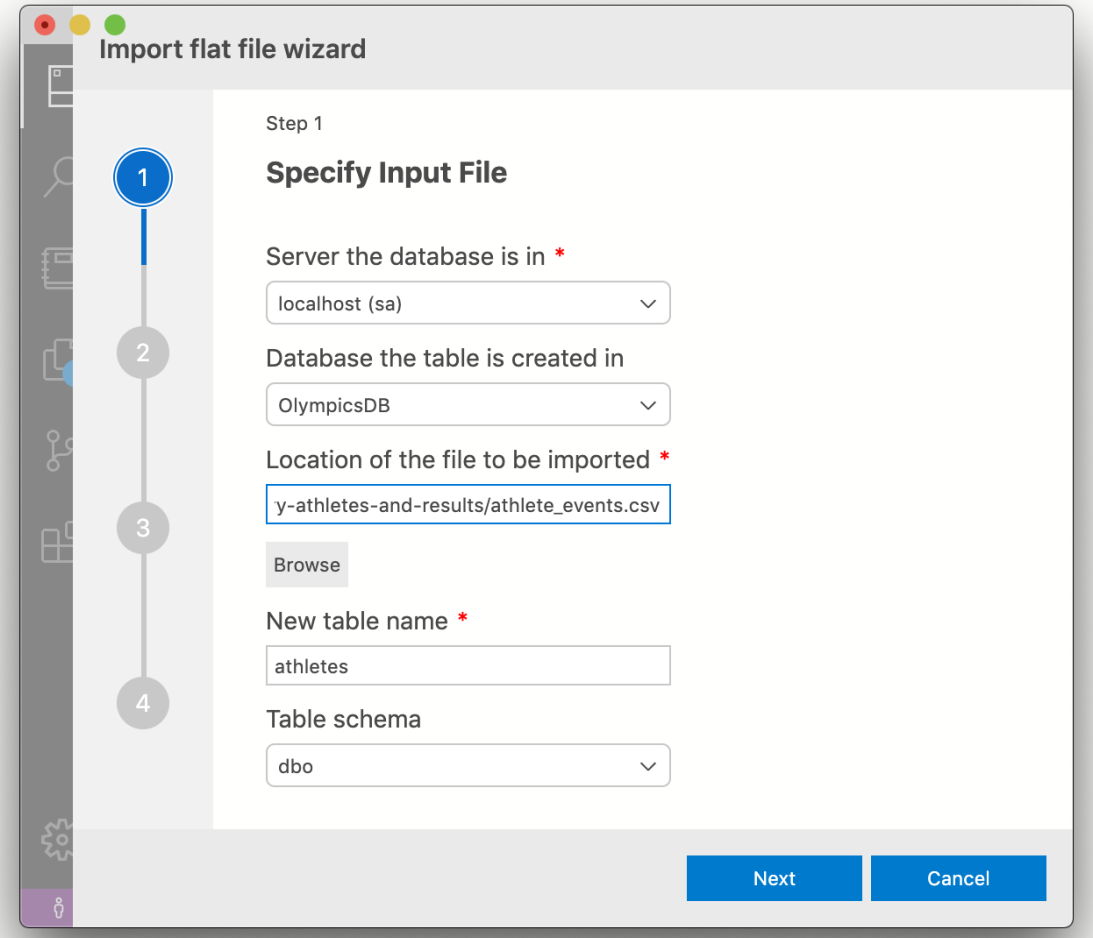
Firstly, to create a database we run the following SQL Server query:

```
create database OlympicsDB;
```

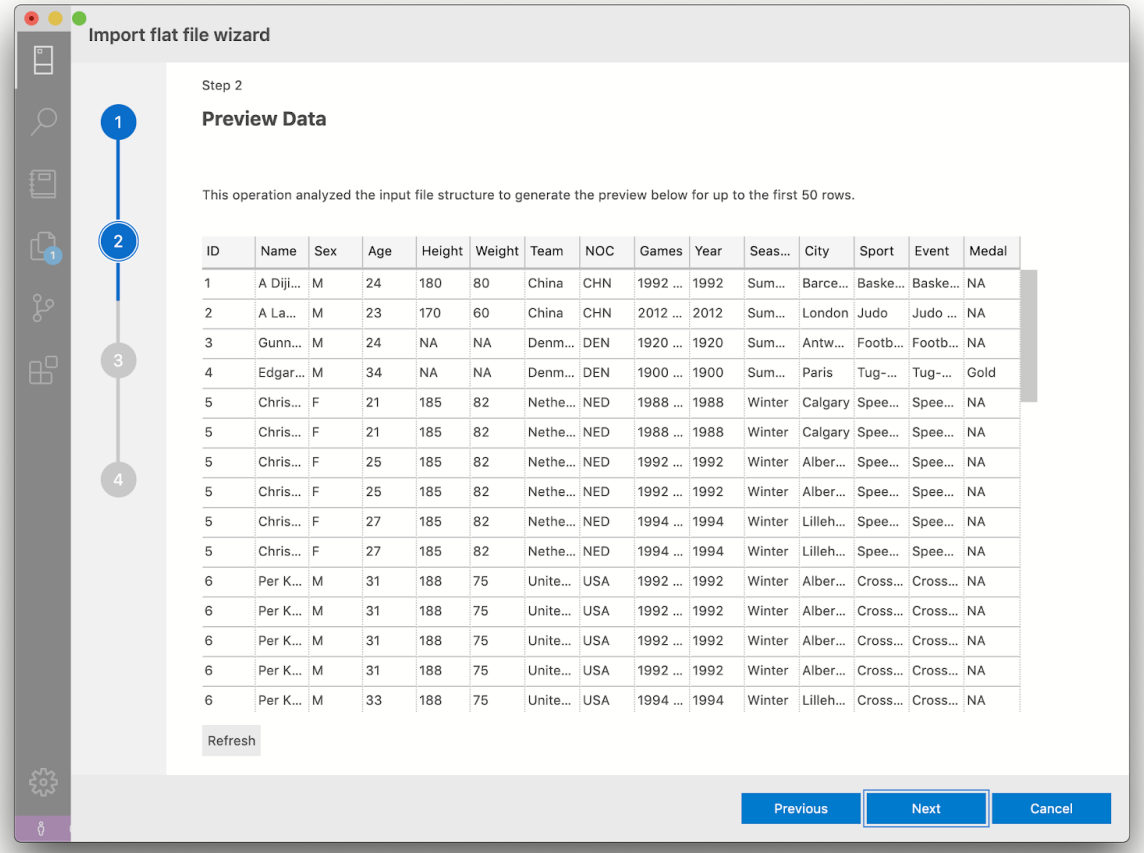
Then, using the flat file import wizard (extension for Azure Data Studio that should be installed before importing the data) we imported the dataset.To import a flat file using Azure Data Studio, requires selecting the database and right-clicking on it, then choosing Import wizard from the available options.

The steps followed to import the athlete_events.csv file can be found below in screenshots:

**Step 1:** Choose the server, the database, browse to find the file, enter table name and choose schema. After doing so, click on next to go to step 2.

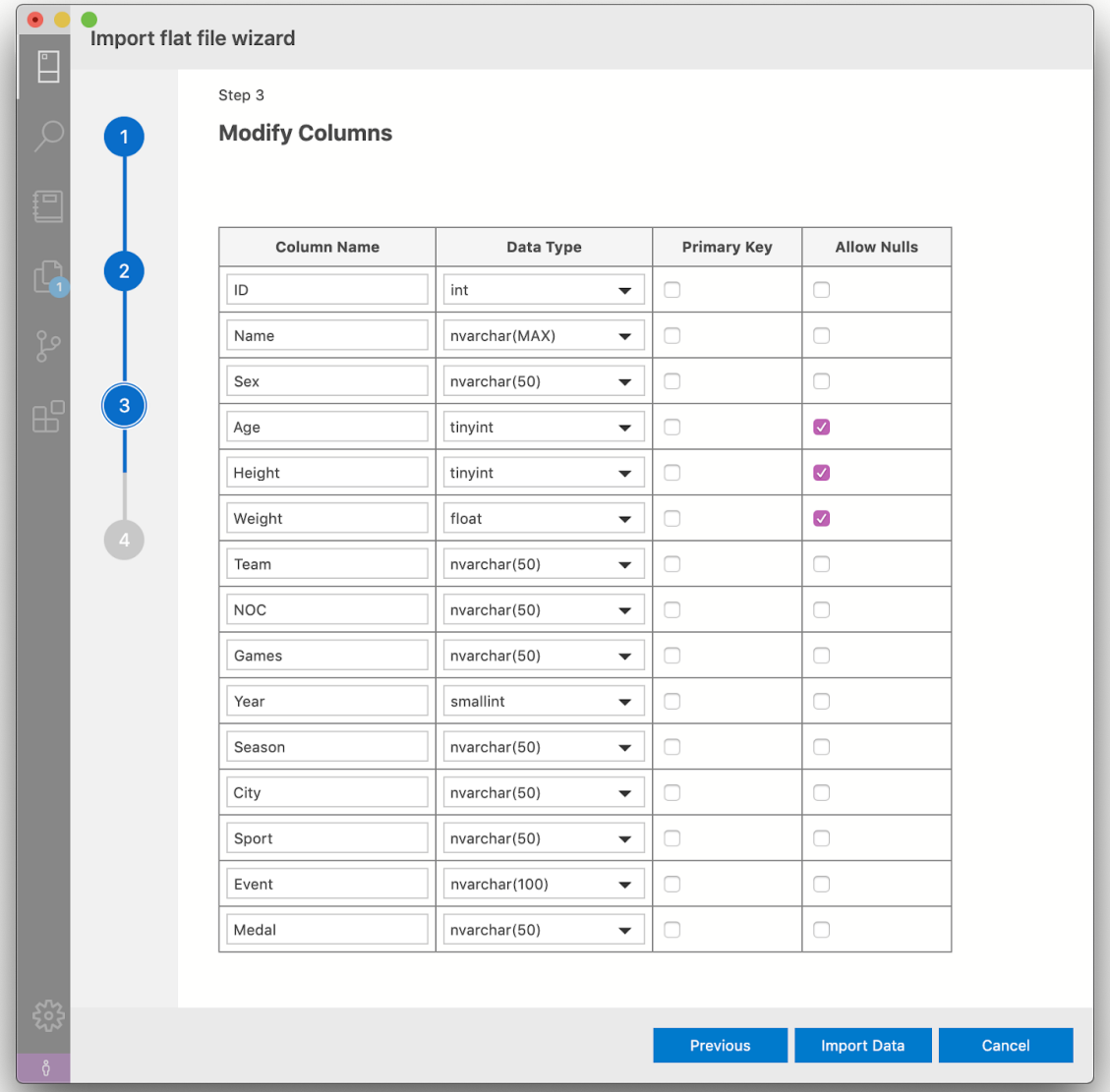


**Step 2:** This window comes up after clicking Next (from Step 1). Should check that data are imported properly. Then, should click on Next to proceed on the Step 3.

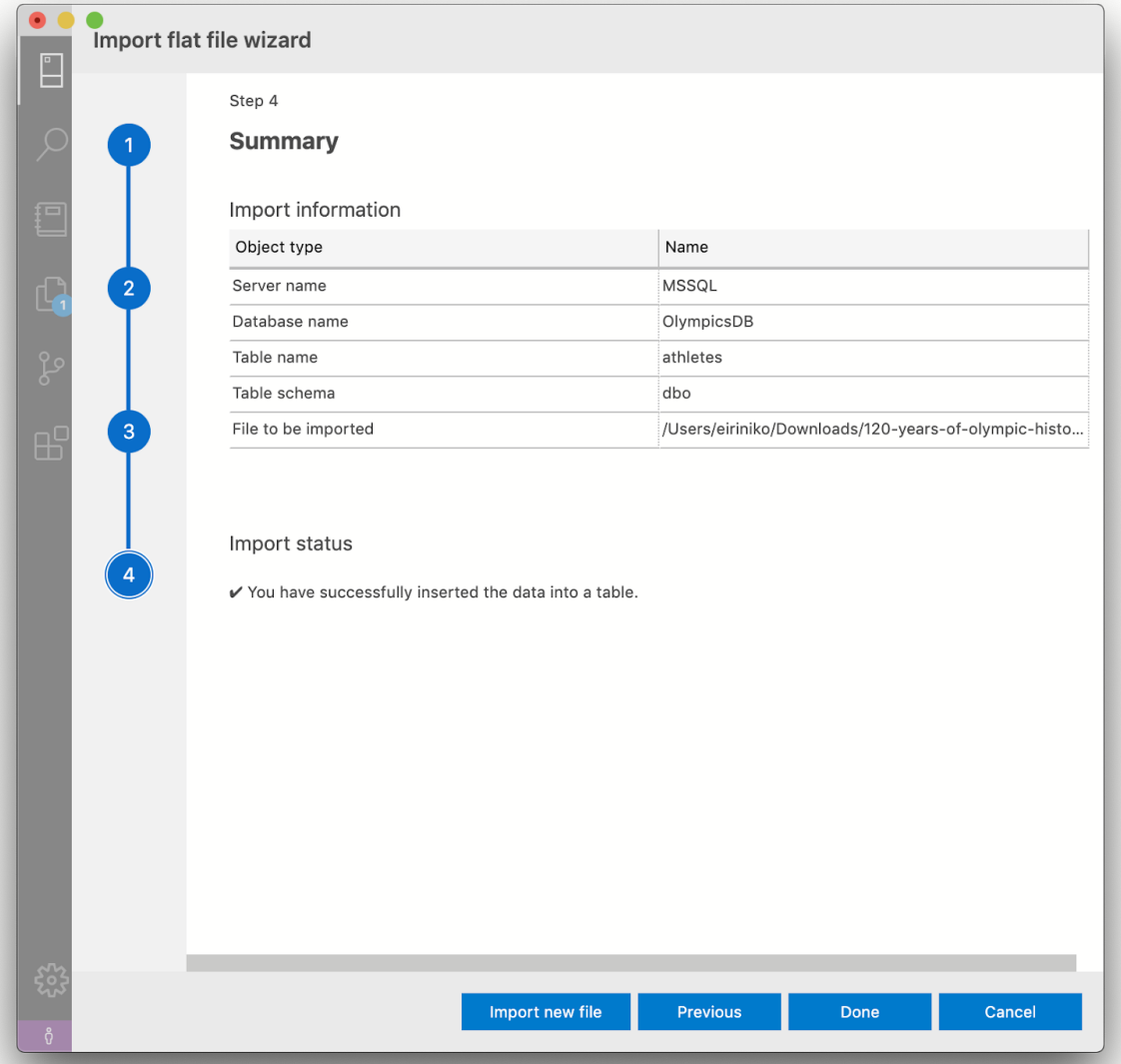



| ID | Name | Sex | Age | Height | Weight | Team | NOC | Games | Year | Seas... | City | Sport | Event | Medal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | A Diji... | M | 24 | 180 | 80 | China | CHN | 1992 ... | 1992 | Sum... | Barce... | Baske... | Baske... | NA |
| 2 | A La... | M | 23 | 170 | 60 | China | CHN | 2012 ... | 2012 | Sum... | London | Judo | Judo ... | NA |
| 3 | Gunn... | M | 24 | NA | NA | Denm... | DEN | 1920 ... | 1920 | Sum... | Antw... | Footb... | Footb... | NA |
| 4 | Edgar... | M | 34 | NA | NA | Denm... | DEN | 1900 ... | 1900 | Sum... | Paris | Tug-... | Tug-... | Gold |
| 5 | Chris... | F | 21 | 185 | 82 | Nethe... | NED | 1988 ... | 1988 | Winter | Calgary | Spee... | Spee... | NA |
| 5 | Chris... | F | 21 | 185 | 82 | Nethe... | NED | 1988 ... | 1988 | Winter | Calgary | Spee... | Spee... | NA |
| 5 | Chris... | F | 25 | 185 | 82 | Nethe... | NED | 1992 ... | 1992 | Winter | Alber... | Spee... | Spee... | NA |
| 5 | Chris... | F | 25 | 185 | 82 | Nethe... | NED | 1992 ... | 1992 | Winter | Alber... | Spee... | Spee... | NA |
| 5 | Chris... | F | 27 | 185 | 82 | Nethe... | NED | 1994 ... | 1994 | Winter | Lilleh... | Spee... | Spee... | NA |
| 5 | Chris... | F | 27 | 185 | 82 | Nethe... | NED | 1994 ... | 1994 | Winter | Lilleh... | Spee... | Spee... | NA |
| 6 | Per K... | M | 31 | 188 | 75 | Unite... | USA | 1992 ... | 1992 | Winter | Alber... | Cross... | Cross... | NA |
| 6 | Per K... | M | 31 | 188 | 75 | Unite... | USA | 1992 ... | 1992 | Winter | Alber... | Cross... | Cross... | NA |
| 6 | Per K... | M | 31 | 188 | 75 | Unite... | USA | 1992 ... | 1992 | Winter | Alber... | Cross... | Cross... | NA |
| 6 | Per K... | M | 31 | 188 | 75 | Unite... | USA | 1992 ... | 1992 | Winter | Alber... | Cross... | Cross... | NA |
| 6 | Per K... | M | 33 | 188 | 75 | Unite... | USA | 1994 ... | 1994 | Winter | Lilleh... | Cross... | Cross... | NA |

**Step 3:** Column types should be changed to fit data needs.

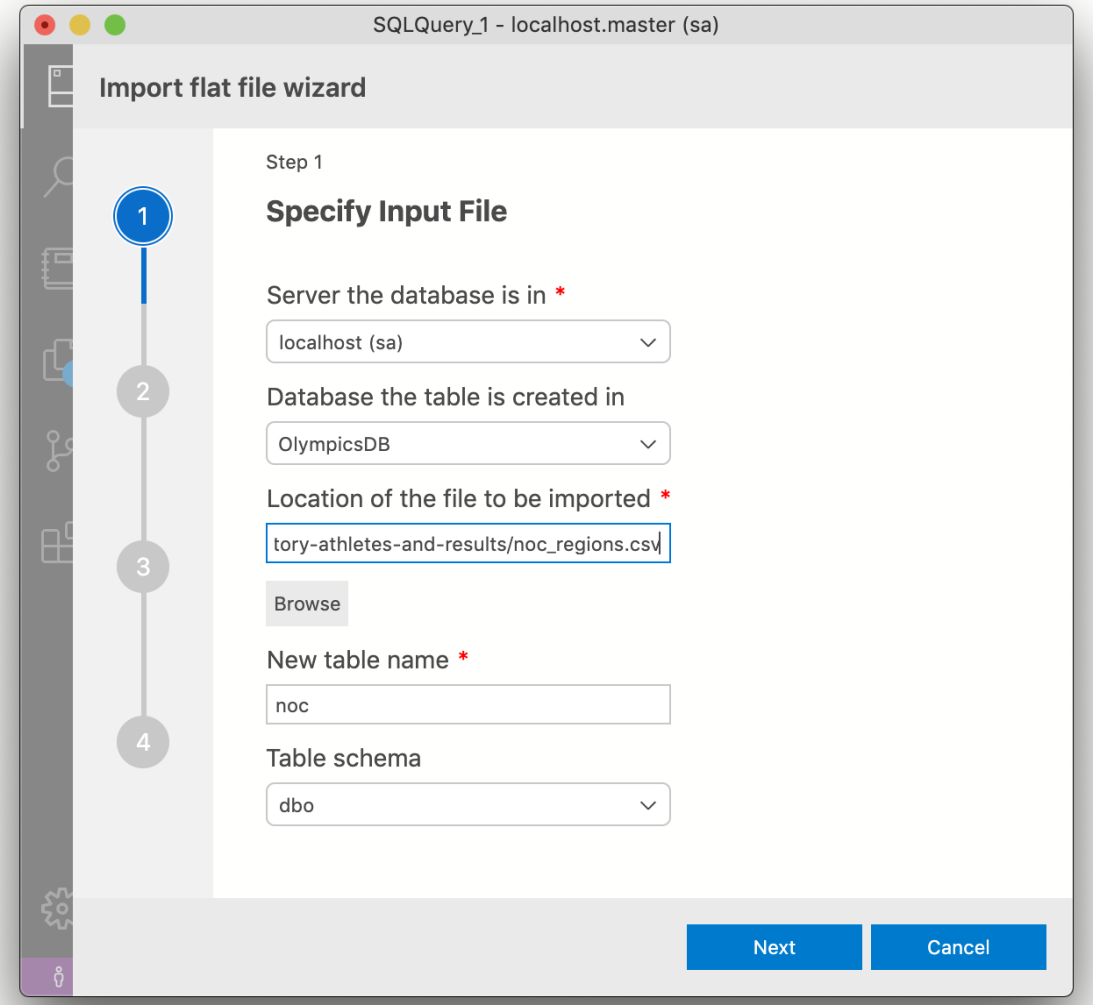


**Step 4:** If the process is successful the following message should be shown.Samess

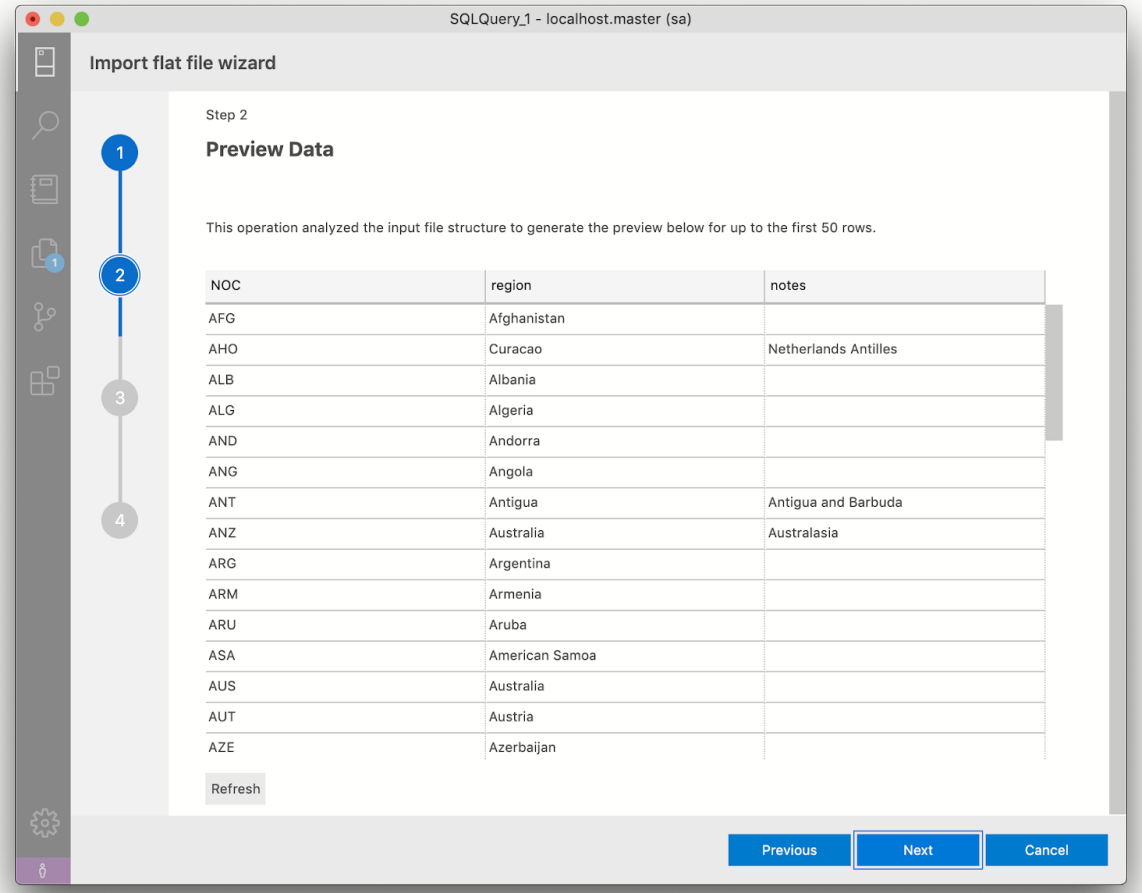



The exact same process should be followed to import all other tables to the database.

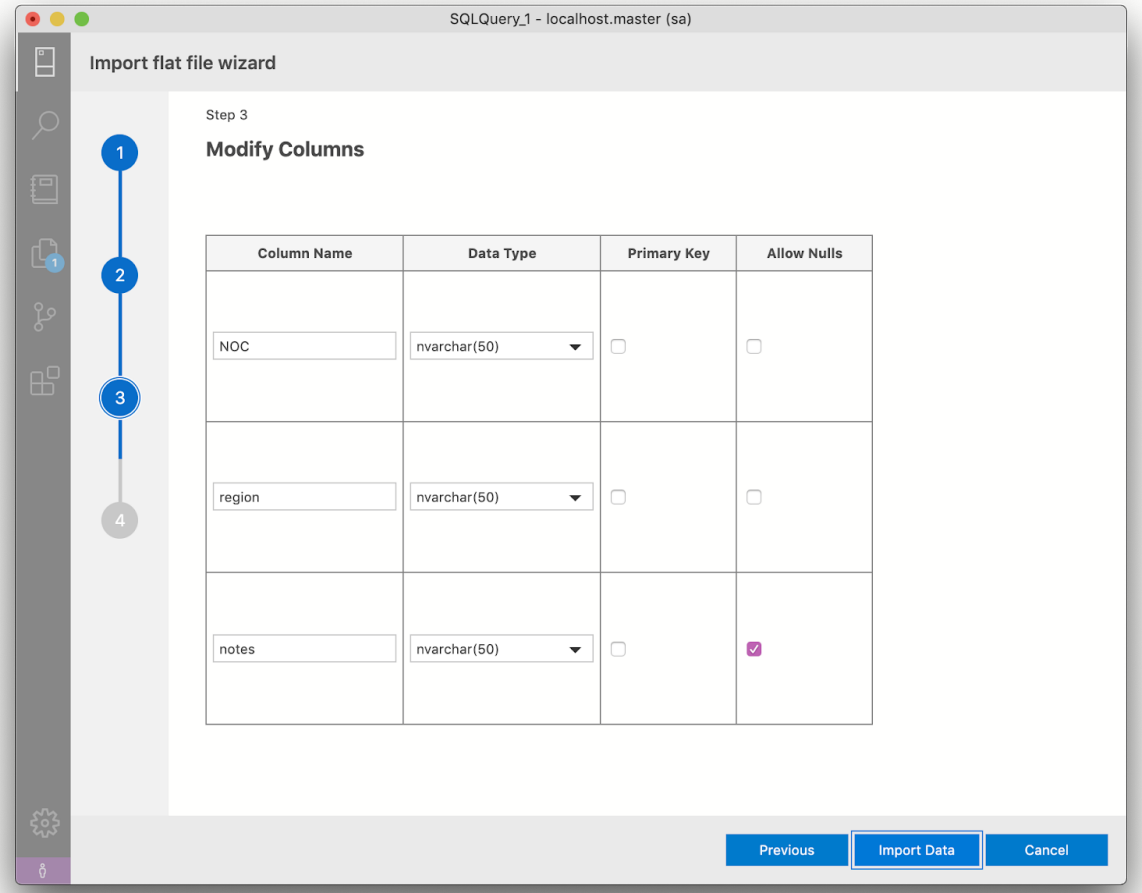
**Step 1**: NOC table



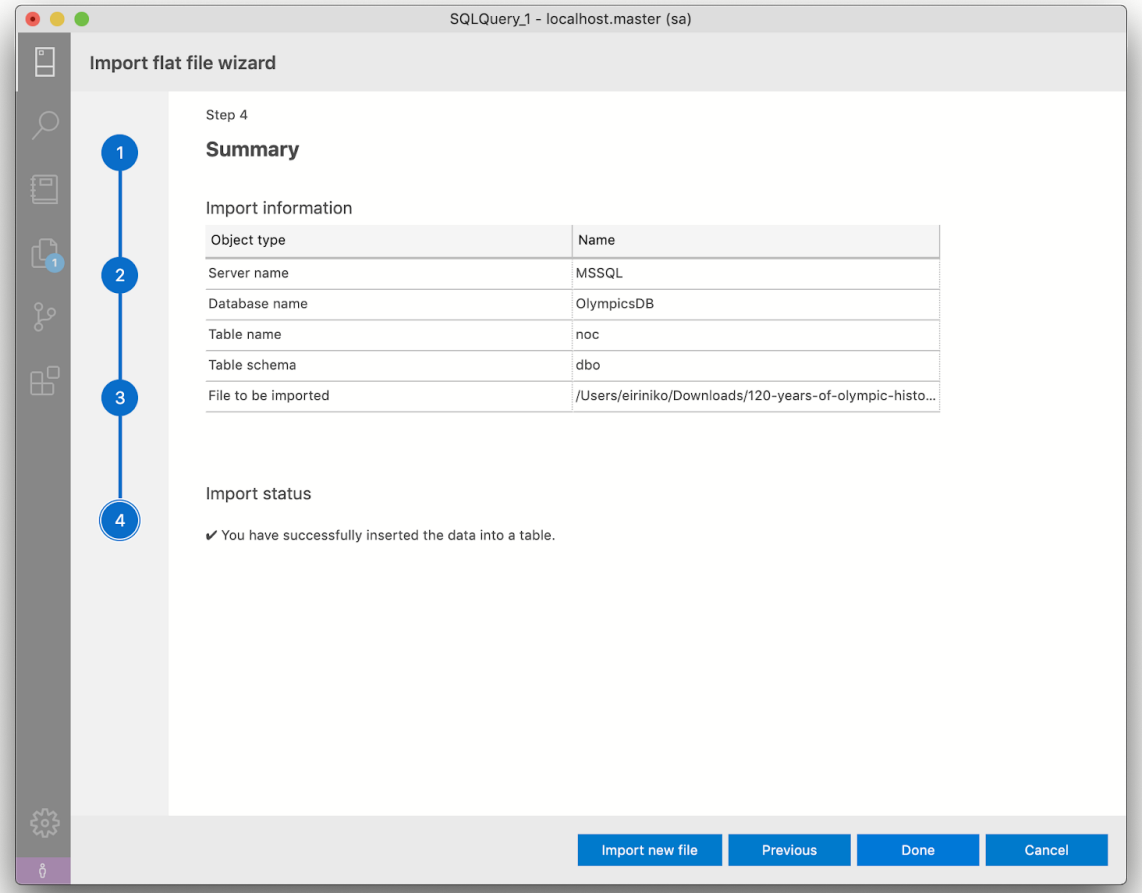
**Step 2**: NOC table



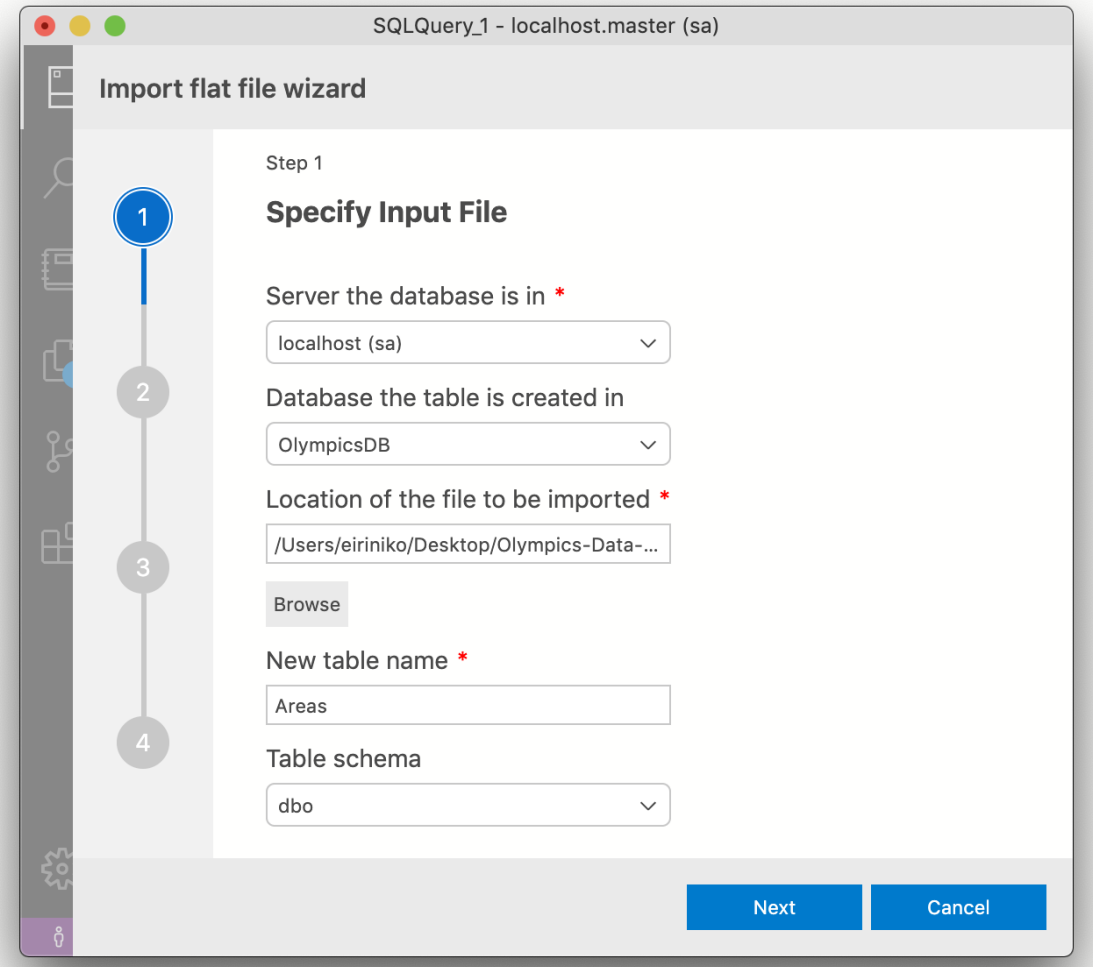
**Step 3**: NOC table



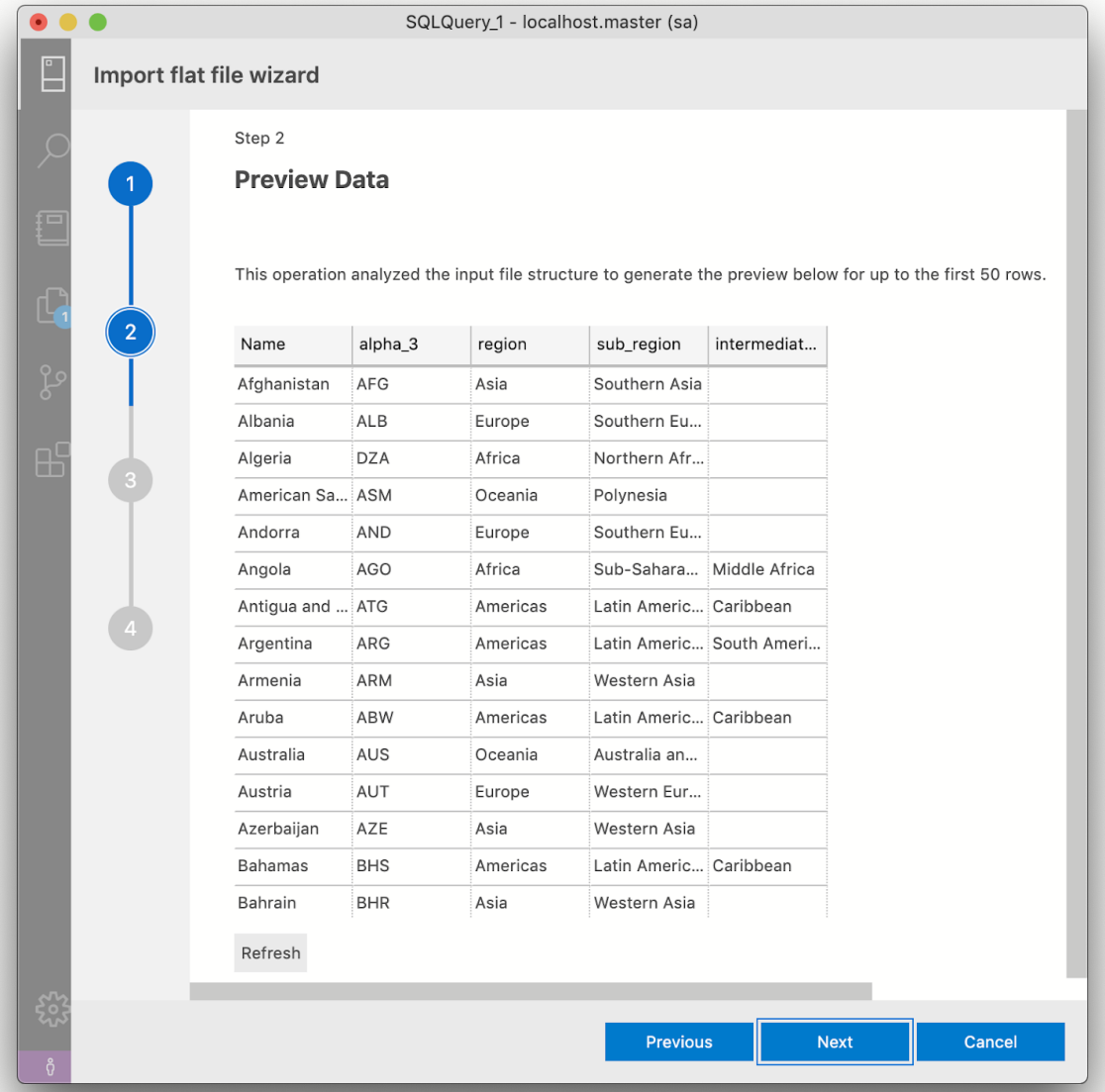
**Step 4**: NOC table



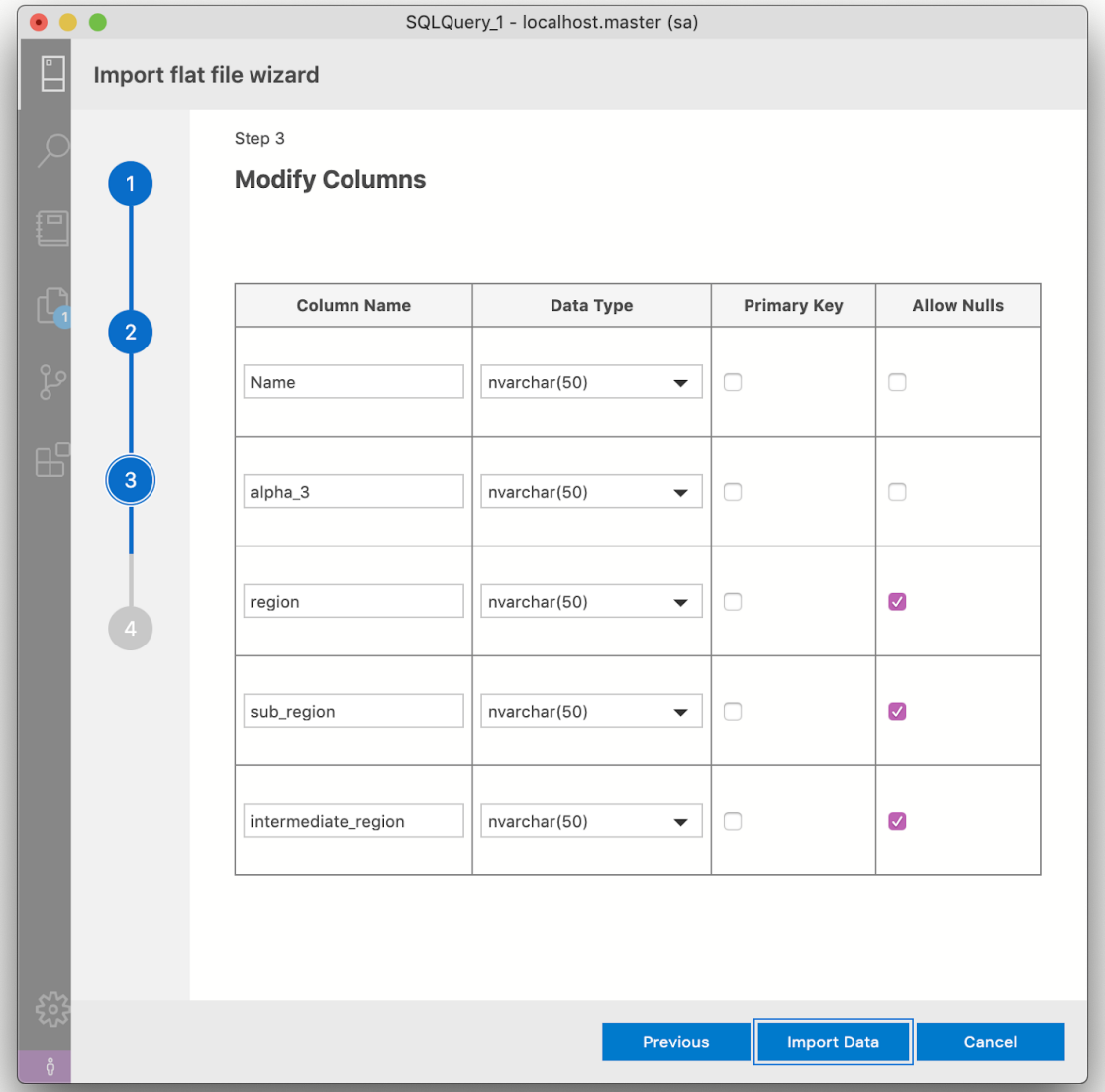
**Step 1**: Areas table



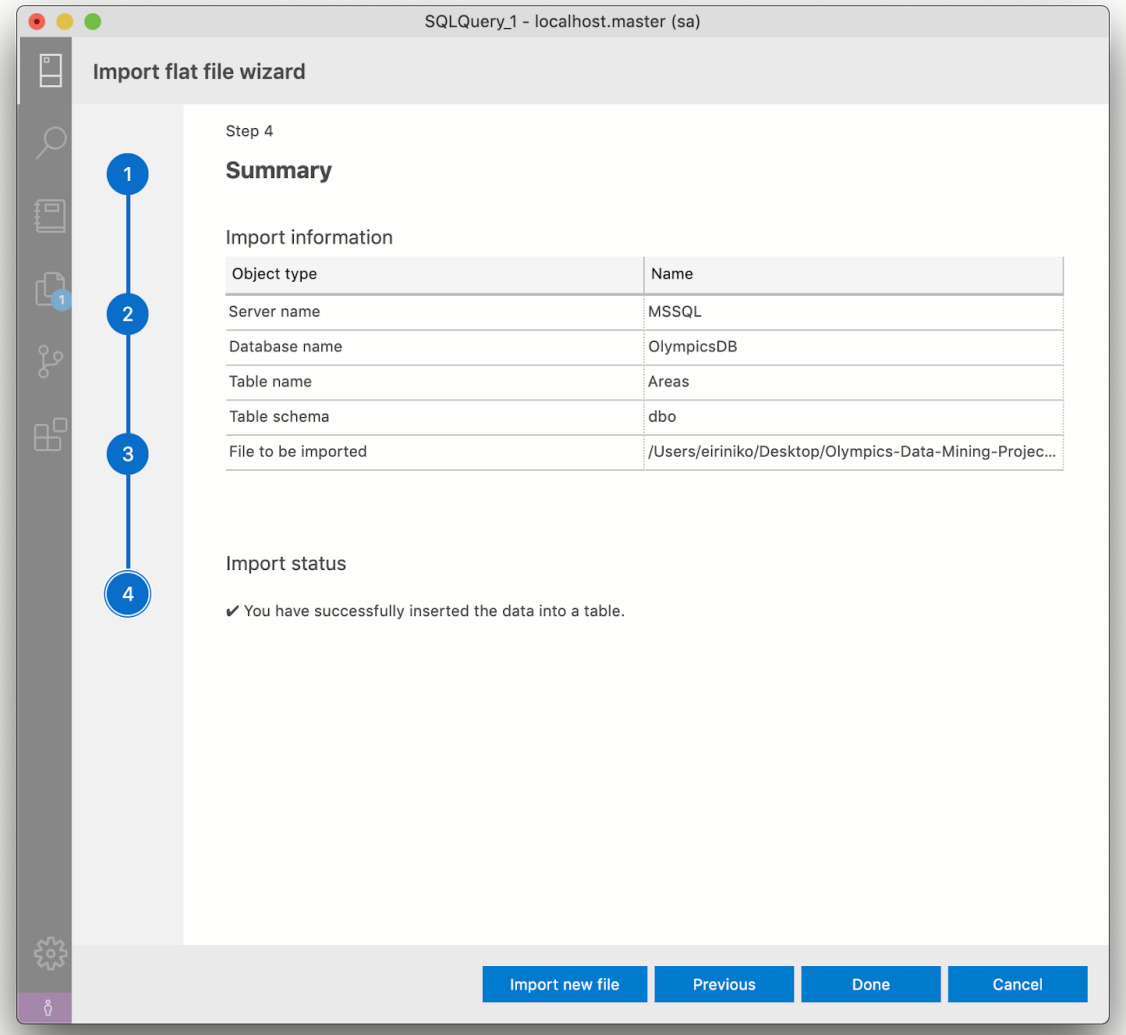
**Step 2**: Areas table



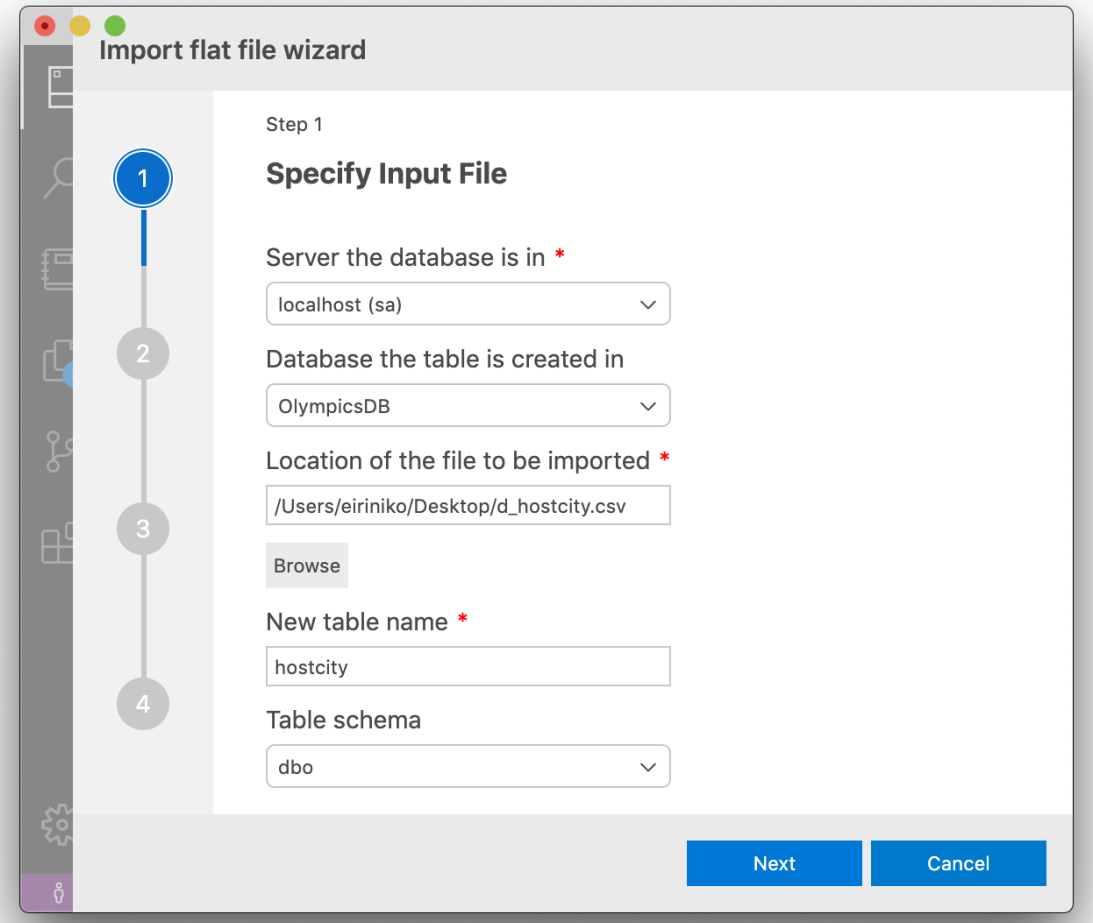
**Step 3**: Areas table



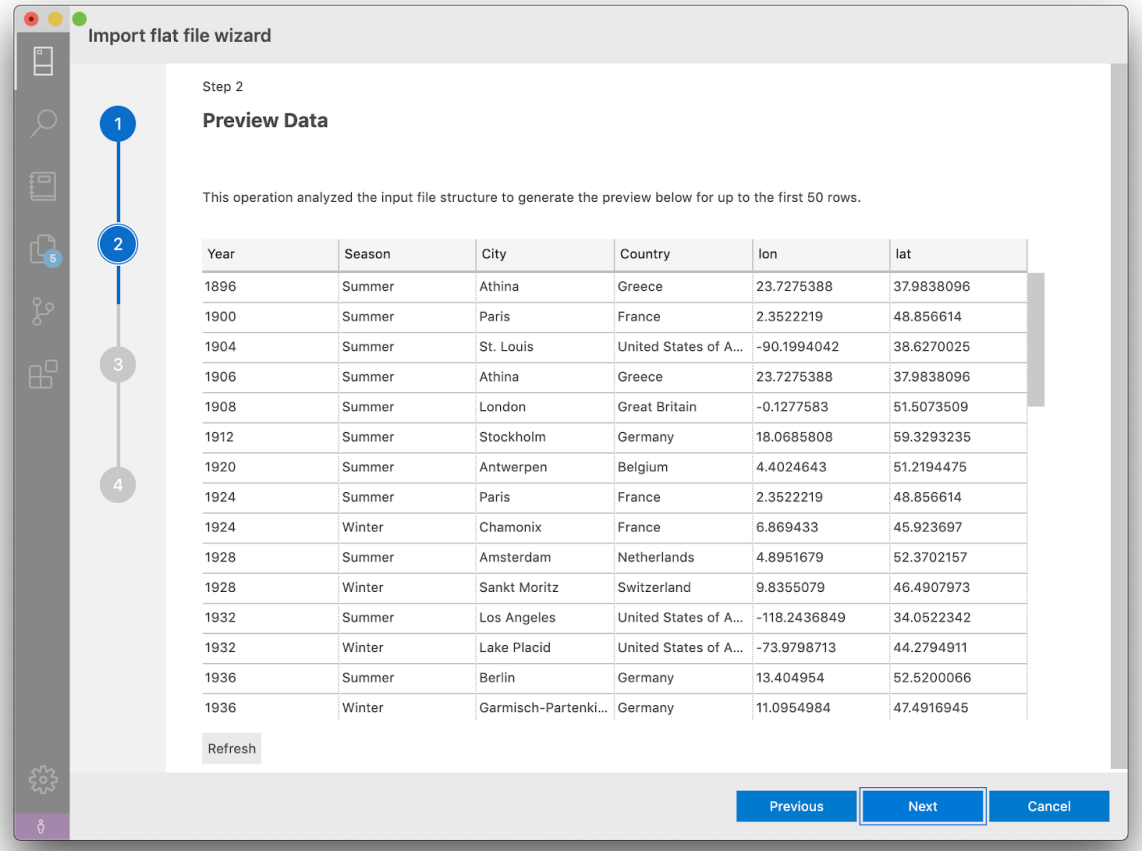
**Step 4**: Areas table



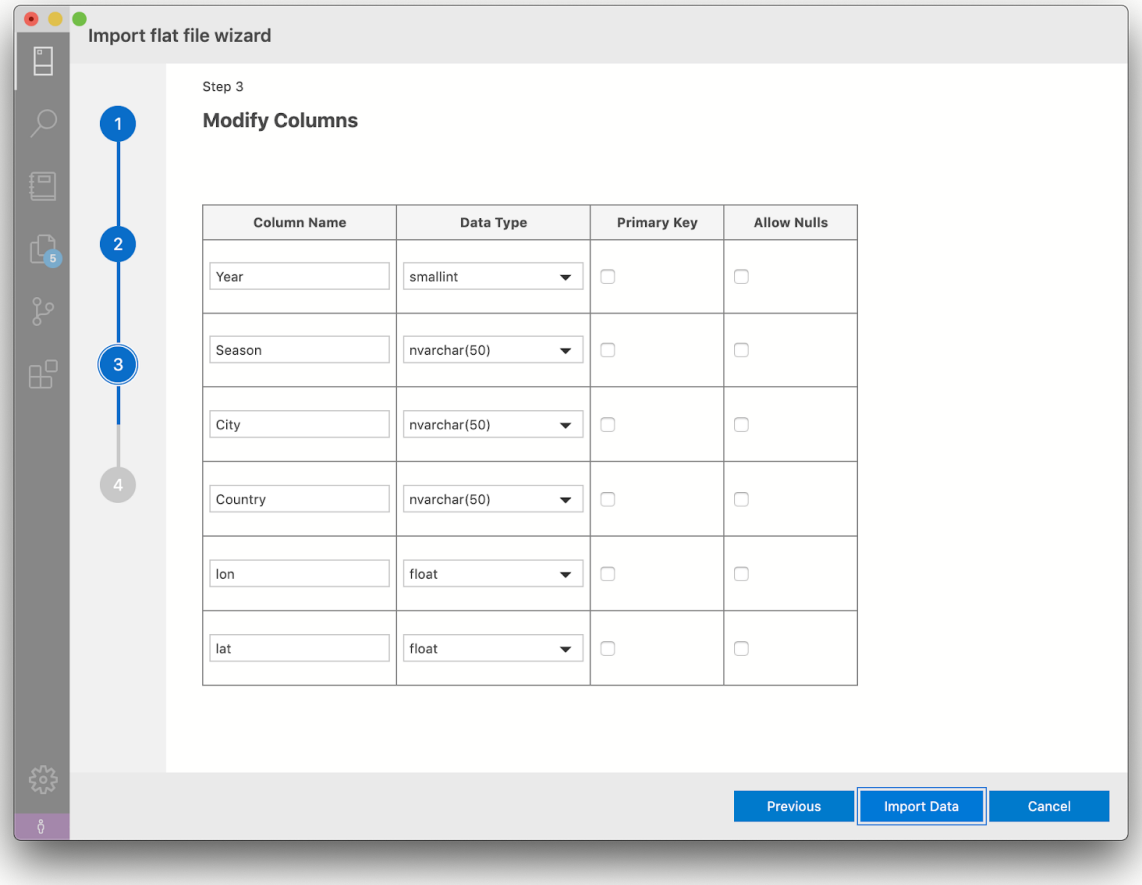
**Step 1**: Host City  table



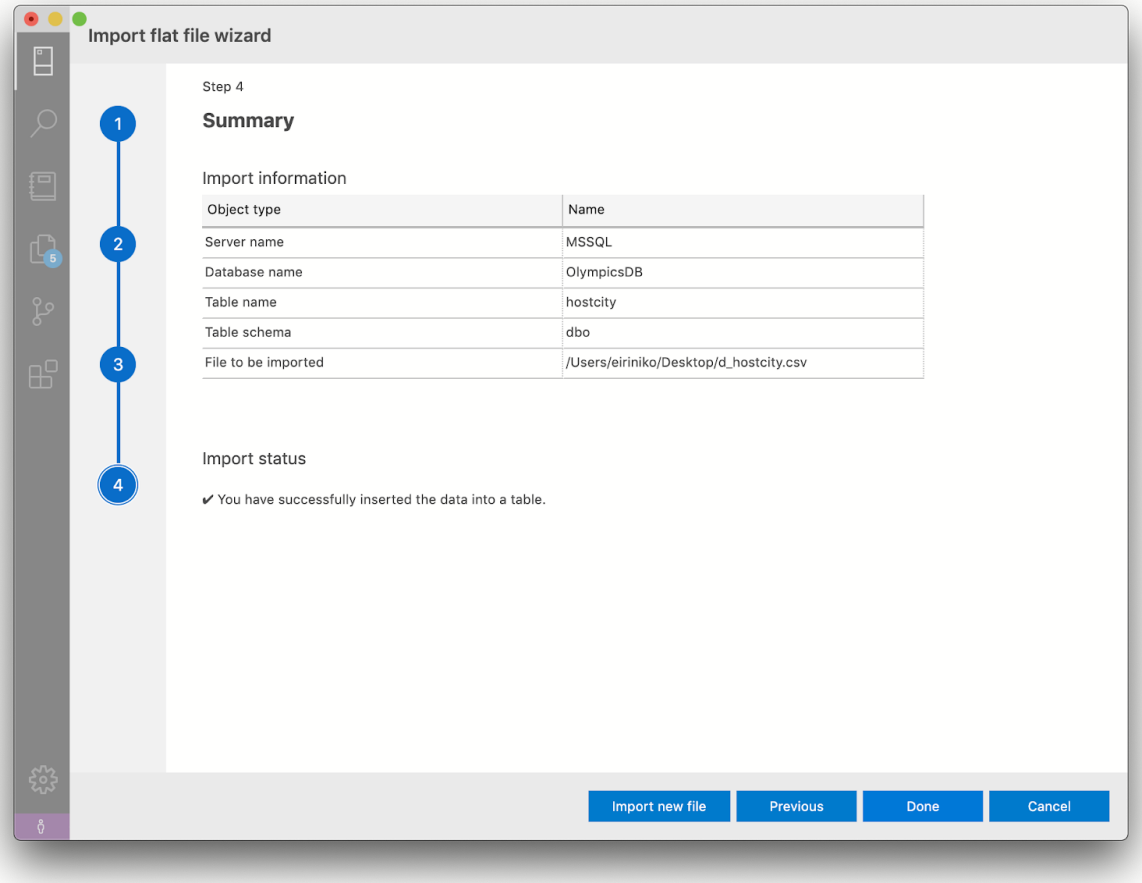
**Step 2**: Host City  table



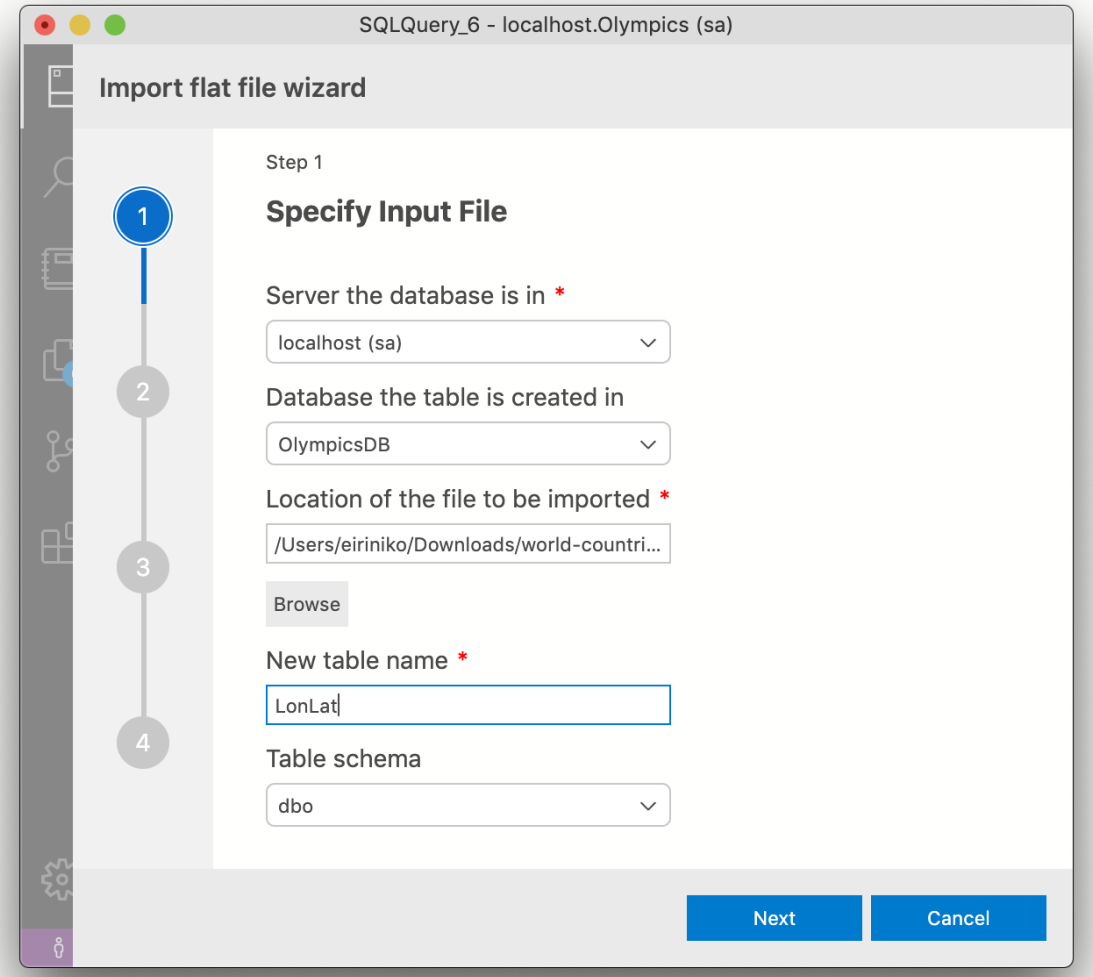
**Step 3**: Host City  table



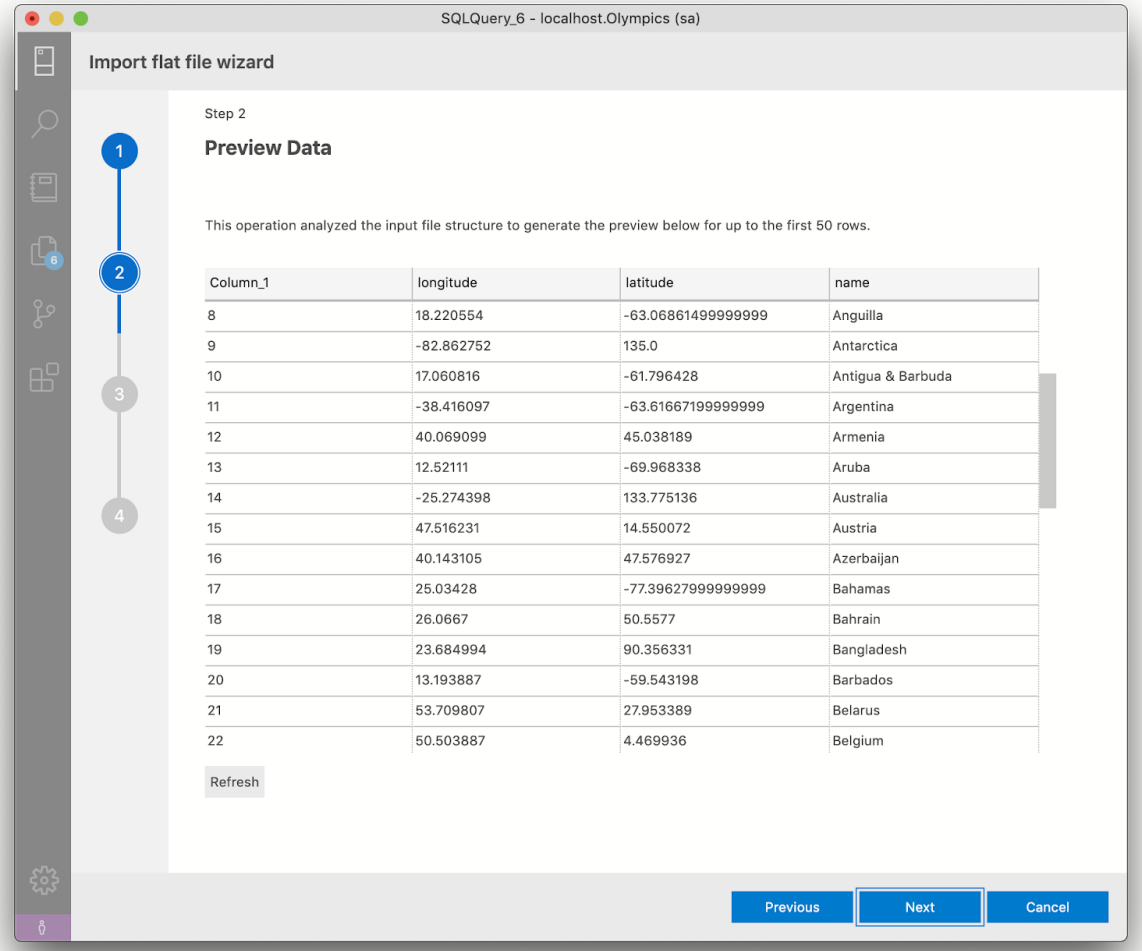
**Step 4**: Host City  table



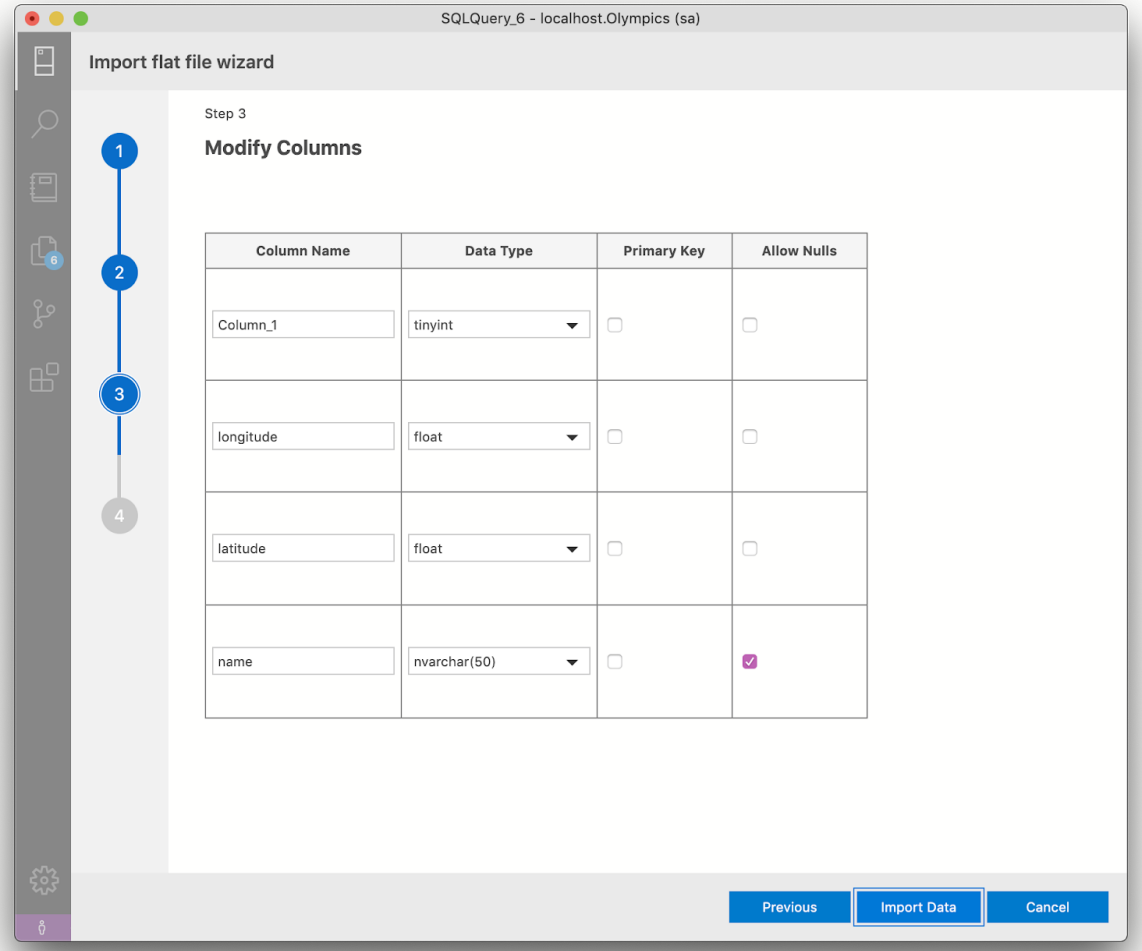
**Step 1**: Longitude-Latitude table



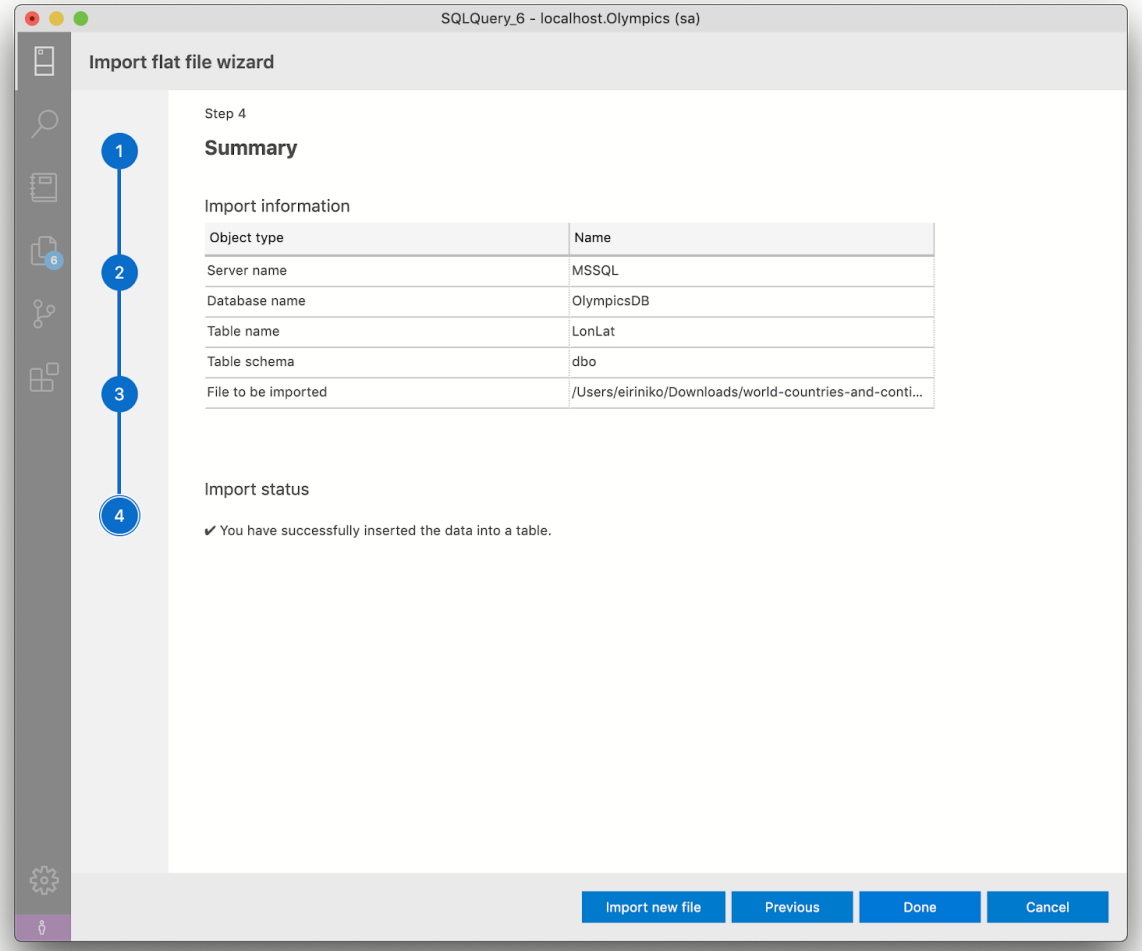
**Step 2**: Longitude-Latitude table



**Step 3**: Longitude-Latitude table



**Step 4**: Longitude-Latitude table




# 3. Data Cleansing

The data chosen for the project were mostly clean. So, the main problem found with data was on noc_regions.csv (noc table from now on). On noc table there are 3 columns: NOC code, country and notes. The problem is on former countries, that many times appear with the same but different NOC code. For example, Greece appears with the code CRT and GRC. For the CRT NOC code on the notes was “Crete”. Also, for some countries there was a different NOC

code, but there was the same country name without notes. So, this would be a problem for the Data Analysis and we changed that, by using the name of the country to the one that those countries had by the time of the games that they competed. The steps followed to solve this problem are:

**Step 1:** At the end of the notes strings there were spaces, that couldn't be removed with the `RTRIM ( character_expression )` function because of the existence of some non readable (non ASCII) characters. Thus, as the number of lines to be affected was very small (18 lines) we manually changed it.

**Step 2:** Change the country name for the countries that have notes and are problematic. This cleansing step doesn't apply for all countries, as Virgin Islands, US and St. Kitts didn't require any changes on their names.

```
1  select * from noc where notes is not null;
2
```

Results Messages

| | NOC | region | notes |
|---|---|---|---|
| 1 | AHO | Curacao | Netherlands Antilles |
| 2 | ANT | Antigua | Antigua and Barbuda |
| 3 | ANZ | Australia | Australasia |
| 4 | BOH | Czech Republic | Bohemia |
| 5 | CRT | Greece | Crete |
| 6 | HKG | China | Hong Kong |
| 7 | IOA | Individual Olympic Athlet… | Individual Olympic Athlet… |
| 8 | ISV | Virgin Islands, US | Virgin Islands |
| 9 | NBO | Malaysia | North Borneo |
| … | NFL | Canada | Newfoundland |
| … | ROT | NA | Refugee Olympic Team |
| … | SCG | Serbia | Serbia and Montenegro |
| … | SKN | Saint Kitts | Turks and Caicos Islands |
| … | TTO | Trinidad | Trinidad and Tobago |
| … | TUV | NA | Tuvalu |
| … | UAR | Syria | United Arab Republic |
| … | UNK | NA | Unknown |
| … | WIF | Trinidad | West Indies Federation |
| … | YAR | Yemen | North Yemen |
| … | YMD | Yemen | South Yemen |
| … | YUG | Serbia | Yugoslavia |

```
UPDATE noc SET region='Netherlands Antilles' WHERE notes ='Netherlands Antilles';
```

```
UPDATE noc SET region='Antigua and Barbuda' WHERE notes='Antigua and Barbuda';
UPDATE noc SET region='Australasia' WHERE notes='Australasia';
UPDATE noc SET region='Bohemia' WHERE notes='Bohemia';
UPDATE noc SET region='Crete' WHERE notes='Crete';
UPDATE noc SET region='Hong Kong' WHERE notes='Hong Kong';
UPDATE noc SET region='North Borneo' WHERE notes='North Borneo';
UPDATE noc SET region='Newfoundland' WHERE notes='Newfoundland';
UPDATE noc SET region='Refugee Olympic Team' WHERE notes='Refugee Olympic Team';
UPDATE noc SET region='Serbia and Montenegro' WHERE notes='Serbia and Montenegro';
UPDATE noc SET region='Trinidad and Tobago' WHERE notes='Trinidad and Tobago';
UPDATE noc SET region='Tuvalu' WHERE notes='Tuvalu';
UPDATE noc SET region='United Arab Republic' WHERE notes='United Arab Republic';
UPDATE noc SET region='West Indies Federation' WHERE notes='West Indies Federation';
UPDATE noc SET region='North Yemen' WHERE notes='North Yemen';
UPDATE noc SET region='South Yemen' WHERE notes='South Yemen';
UPDATE noc SET region='Yugoslavia' WHERE notes='Yugoslavia';
```

**Step 3:** Change countries that their name is not following their NOC code (eg TCH: Czech Republic but it should be Czechoslovakia) + change the name of Bolivia which is misspelled.

```
UPDATE noc SET region='Bolivia' WHERE region='Boliva';
UPDATE noc SET region='Czechoslovakia' WHERE NOC='TCH';
UPDATE noc SET region='West Germany' WHERE NOC='FRG';
UPDATE noc SET region='East Germany' WHERE NOC='GDR';
UPDATE noc SET region='Saar' WHERE NOC='SAA';
UPDATE noc SET region='Malaya' WHERE NOC='MAL';
```

```
UPDATE noc SET region='Soviet Union' WHERE NOC='URS';
UPDATE noc SET region='Unified Team' WHERE NOC='EUN';
UPDATE noc SET region='South Vietnam' WHERE NOC='VNM';
UPDATE noc SET region='Rhodesia' WHERE NOC='RHO';
```

The main problem with the data was the different sources for data about countries and their characteristics. For example, there is tension between noc and athletes for the country code on Singapore;

```
UPDATE noc SET NOC='SGP' WHERE region='Singapore';
```

There are different country names for some countries between LonLat and noc;

```
UPDATE LonLat SET name='Antigua and Barbuda' WHERE name='Antigua & Barbuda';
UPDATE LonLat SET name='Bosnia and Herzegovina' WHERE name='Bosnia';
UPDATE LonLat SET name='Virgin Islands, British' WHERE name='British Virgin Islands';
UPDATE LonLat SET name='Democratic Republic of the Congo' WHERE name='Congo - Kinshasa';
UPDATE LonLat SET name='Republic of Congo' WHERE name='Congo - Brazzaville';
UPDATE LonLat SET name='Netherlands Antilles' WHERE name='Curaçao';
UPDATE LonLat SET name='Ivory Coast' WHERE name='Côte d'Ivoire';
UPDATE LonLat SET name='Virgin Islands, US' WHERE name='U.S. Virgin Islands';
UPDATE LonLat SET name='Trinidad and Tobago' WHERE name='Trinidad & Tobago';
UPDATE LonLat SET name='Sao Tome and Principe' WHERE name='São Tomé & Príncipe';
UPDATE LonLat SET name='Saint Vincent' WHERE name='St. Vincent & Grenadines';
UPDATE LonLat SET name='Saint Lucia' WHERE name='St. Lucia';
UPDATE LonLat SET name='Saint Kitts' WHERE name='St. Kitts & Nevis';
UPDATE LonLat SET name='USA' WHERE name='US';
```

```
UPDATE noc SET NOC='SGP' WHERE region='Singapore';
```

Finally, we have a different name for the Moscow in hostcity and athletes;

```
UPDATE hostcity SET city='Moskva' WHERE city='Moscow';
```

**Step 4:** For the LonLat table that contains the longitude and the latitude of the countries, we should add the former countries that have participated. The longitude and latitude is the same as the countries that are today in their geographic location. Also, for participants that played in the Olympics independently or as part of the refugee team or there is no information of their country, longitude and latitude will be set as 0.

```
insert into LonLat (Column_1,longitude,latitude,name) values (
   252,
   (select longitude from LonLat where name='Greece'),
   (select latitude from LonLat where name='Greece'),'Crete');

insert into LonLat (Column_1,longitude,latitude,name) values (
   251,
   (select longitude from LonLat where name='Zimbabwe'),
   (select latitude from LonLat where name='Zimbabwe'),'Rhodesia');

insert into LonLat (Column_1,longitude,latitude,name) values (
   253,
   (select longitude from LonLat where name='Australia'),
                (select     latitude     from     LonLat     where
name='Australia'),'Australasia');

insert into LonLat (Column_1,longitude,latitude,name) values (
   254,
   (select longitude from LonLat where name='Czech Republic'),
          (select    latitude    from    LonLat    where    name='Czech
Republic'),'Bohemia');
```

```
insert into LonLat (Column_1,longitude,latitude,name) values (
   255,
   (select longitude from LonLat where name='Czech Republic'),
           (select    latitude    from    LonLat    where    name='Czech
Republic'),'Czechoslovakia');

insert into LonLat (Column_1,longitude,latitude,name) values (
   256,
   (select longitude from LonLat where name='Germany'),
       (select   latitude   from   LonLat   where   name='Germany'),'East
Germany');

insert into LonLat (Column_1,longitude,latitude,name) values (
   257,
   (select longitude from LonLat where name='Germany'),
       (select   latitude   from   LonLat   where   name='Germany'),'West
Germany');

insert into LonLat (Column_1,longitude,latitude,name) values (
   258,
   (select longitude from LonLat where name='Germany'),
   (select latitude from LonLat where name='Germany'),'Saar');

insert into LonLat (Column_1,longitude,latitude,name) values (
   259,
   (select longitude from LonLat where name='Serbia'),
     (select  latitude  from  LonLat  where  name='Serbia'),'Serbia  and
Montenegro');

insert into LonLat (Column_1,longitude,latitude,name) values (
   260,
   (select longitude from LonLat where name='Serbia'),
   (select latitude from LonLat where name='Serbia'),'Kosovo');
```

```
insert into LonLat (Column_1,longitude,latitude,name) values (
   261,
   (select longitude from LonLat where name='Serbia'),
   (select latitude from LonLat where name='Serbia'),'Yugoslavia');

insert into LonLat (Column_1,longitude,latitude,name) values (
   262,
   (select longitude from LonLat where name='Malaysia'),
   (select latitude from LonLat where name='Malaysia'),'Malaya');

insert into LonLat (Column_1,longitude,latitude,name) values (
   263,
   (select longitude from LonLat where name='Canada'),
   (select latitude from LonLat where name='Canada'),'Newfoundland');

insert into LonLat (Column_1,longitude,latitude,name) values (
   264,
   (select longitude from LonLat where name='Yemen'),
   (select latitude from LonLat where name='Yemen'),'North Yemen');

insert into LonLat (Column_1,longitude,latitude,name) values (
   265,
   (select longitude from LonLat where name='Yemen'),
   (select latitude from LonLat where name='Yemen'),'South Yemen');

insert into LonLat (Column_1,longitude,latitude,name) values (
   266,
   (select longitude from LonLat where name='Vietnam'),
      (select  latitude  from  LonLat  where  name='Vietnam'),'South
Vietnam');

insert into LonLat (Column_1,longitude,latitude,name) values (
```

```
   267,
   (select longitude from LonLat where name='Syria'),
     (select latitude from LonLat where name='Syria'),'United Arab
Republic');

insert into LonLat (Column_1,longitude,latitude,name) values (
   268,
   (select longitude from LonLat where name='Russia'),
   (select latitude from LonLat where name='Russia'),'Soviet Union');

insert into LonLat (Column_1,longitude,latitude,name) values (
   269,
   (select longitude from LonLat where name='Russia'),
   (select latitude from LonLat where name='Russia'),'Unified Team');

insert into LonLat (Column_1,longitude,latitude,name) values (
   270,0,0,'NA');

insert into LonLat (Column_1,longitude,latitude,name) values (
   271,0,0,'Refugee Olympic Team');

insert into LonLat (Column_1,longitude,latitude,name) values (
   272,0,0,'Individual Olympic Athletes');

insert into LonLat (Column_1,longitude,latitude,name) values (
   273,
   (select longitude from LonLat where name='Malaysia'),
      (select latitude from LonLat where name='Malaysia'),'North
Borneo');

insert into LonLat (Column_1,longitude,latitude,name) values (
   274,
   (select longitude from LonLat where name='Jamaica'),
```

```
    (select latitude from LonLat where name='Jamaica'),'West Indies
Federation');
```

# 4. Data Transformation

## 4.1. Transformation

To create a table with all data inside. Originally, the dataset from Kaggle had in each line athletes and their medal result for each Olympics, sport and event they participated in. The main transformation done, was to change the data from each row one athlete at an event to each row one country at an event, sport and Olympics. So, instead of showing athlete X winning the gold medal, the table is showing country Y, its participants in the event, the results and their athletes details (eg average Weight).

More specifically, each row of the fw table has an id, the Games (eg. Athens 2004), the season (eg. Summer), the city (eg. Athens), the year (eg. 2004), the sport (eg. Athletics), the event (eg. Athletics Men's 5,000 metres), the country of the team (eg. France), its NOC code (eg. FRA) and the host country (eg. Greece). In the table there are and some metrics, which are presented and explained in 4.2 part of this report.

```
drop table if exists fw;
Go

create table fw(
   id INT IDENTITY(1,1) NOT NULL PRIMARY KEY,
   [Games] [nvarchar](50) NOT NULL,
   [Season] [nvarchar](50) NOT NULL,
   [City] [nvarchar](50) NOT NULL,
   [Year] [int] NOT NULL,
   [Sport] [nvarchar](50) NOT NULL,
   [Event] [nvarchar](100) NOT NULL,
   [Country] [nvarchar](50) NOT NULL,
   [NOC] [nvarchar](50) NOT NULL,
   [countF] INT,
```

```
    [countM] INT,
    [countParticipants] INT,
    [countGold] INT,
    [countSilver] INT,
    [countBronze] INT,
    [avgAge] FLOAT(24),
    [avgHeight] FLOAT(24),
    [avgWeight] FLOAT(24),
    [HC] [nvarchar](50),
    [Longitude] FLOAT,
    [Latitude] FLOAT
);
Go

insert into fw
(Games,Season,City,Year,Sport,Event,Country,NOC,HC,Longitude,Latitude
,avgAge,avgHeight,avgWeight,countParticipants,countM,
countF,countGold,countSilver,countBronze)
SELECT
games,Athletes.Season,Athletes.City,Athletes.year,Athletes.sport,even
t,noc.region,noc.NOC,hostcity.Country,LonLat.Longitude,LonLat.Latitud
e,Avg(Age) as AvgAge,Avg(Height) as AvgHgt,Avg(Weight) as AvgWgt,
count(Athletes.Name) As Participants,
    sum(case when Sex = 'M' then 1 else 0 end) AS Male,
    sum(case when Sex = 'F' then 1 else 0 end) AS Female,
    sum(case when Medal = 'Gold' then 1 else 0 end) AS Gold,
    sum(case when Medal = 'Silver' then 1 else 0 end) AS Silver,
    sum(case when Medal = 'Bronze' then 1 else 0 end) AS Bronze
FROM Athletes,noc,hostcity,LonLat
Where noc.NOC=athletes.NOC AND Athletes.City=hostcity.City AND
noc.NOC=Athletes.NOC AND LonLat.name=noc.region
GROUP BY
Athletes.games,Athletes.season,Athletes.city,Athletes.year,Athletes.s
```

```
port,Athletes.event,noc.region,noc.NOC,hostcity.Country,LonLat.Longit
ude,LonLat.Latitude;Go
```

## 4.2. Metrics

The metrics used are count of female participants (countF), count of male participants (countM), count of total participants (countParticipants), count of gold medals (countGold), count of silver medals (countSilver), count of bronze medals (countBronze), average age (avgAge), average height (avgHeight), average weight (avgWeight), Longitude, Latitude. All the metrics are calculated by row, which means that for each team participating at each sport, event and Olympics it is calculated differently. Probably, longitude and latitude are not calculated in a specific way, but they are just the metrics for this country.

## 4.3. Dimensions

The dimensions created are the following: Sport (eg. Rowing), City (eg. Los Angeles), Event (eg. Men's 100M), Region (eg. France), Season (eg. Summer), Year (eg. 2012), Games (eg. Beijing 2008), Host Country (eg. Greece), Greater Region (eg. Western Europe) and NOC (eg. AUS). In all dimensions, there is the value and an assigned id to it.

```
--(1)create Sport dimension
drop table if exists Sport_Dimension;
Create table Sport_Dimension(
   sd_id INT IDENTITY(1,1) PRIMARY KEY,
   sport nvarchar(max) NOT NULL
);
INSERT INTO Sport_Dimension (sport)
SELECT distinct Sport
FROM fw;
```

```sql
--(2)create City dimension
drop table if exists City_Dimension;
Create table City_Dimension(
   ct_id INT IDENTITY(1,1) PRIMARY KEY,
   city nvarchar(50) NOT NULL
);
INSERT INTO City_Dimension (city)
SELECT distinct City
FROM fw;

--(3)create Event dimension
drop table if exists Event_Dimension;
Create table Event_Dimension(
   ev_id INT IDENTITY(1,1) PRIMARY KEY,
   event nvarchar(100) NOT NULL
);
INSERT INTO Event_Dimension (event)
SELECT distinct Event
FROM fw;

--(4)create Region dimension
drop table if exists Country_Dimension;
Create table Country_Dimension(
   cn_id INT IDENTITY(1,1) PRIMARY KEY,
   country nvarchar(50) NOT NULL
);
INSERT INTO Country_Dimension (country)
SELECT distinct Country
FROM fw;

--(5)create Season dimension
drop table if exists Season_Dimension;
Create table Season_Dimension(
```

```
   ss_id INT IDENTITY(1,1) PRIMARY KEY,
   season nvarchar(50) NOT NULL
);
INSERT INTO Season_Dimension (season)
SELECT distinct Season
FROM fw;

--(6)create Year dimension
drop table if exists Year_Dimension;
Create table Year_Dimension(
   yr_id INT IDENTITY(1,1) PRIMARY KEY,
   year int NOT NULL
);
INSERT INTO Year_Dimension (year)
SELECT distinct Year
FROM fw;

--(7)create Games dimension
drop table if exists Games_Dimension;
Create table Games_Dimension(
   gm_id INT IDENTITY(1,1) PRIMARY KEY,
   games nvarchar(50) NOT NULL
);
INSERT INTO Games_Dimension (games)
SELECT distinct Games
FROM fw;

--(8)create HC (Host Country)dimension
drop table if exists HC_Dimension;
Create table HC_Dimension(
   hc_id INT IDENTITY(1,1) PRIMARY KEY,
   hostCountry nvarchar(50) NOT NULL
);
```

```
INSERT INTO HC_Dimension (hostCountry)
SELECT distinct HC
FROM fw;

drop table if exists NOC_Dimension;

drop table if exists GreaterRegion_Dimension;

--(9)create Greater Region dimension
Create table GreaterRegion_Dimension(
   gr_id INT IDENTITY(1,1) PRIMARY KEY,
   sub_region nvarchar(80),
);

INSERT INTO GreaterRegion_Dimension (sub_region)
SELECT Distinct sub_region
FROM Areas;
GO

INSERT INTO GreaterRegion_Dimension (sub_region) VALUES ('None')

drop table if exists NOC_Dimension;

--(10)create NOC dimension
Create table NOC_Dimension(
   nc_id INT IDENTITY(1,1) PRIMARY KEY,
   NOC nvarchar(50) NOT NULL,
   gr_id int FOREIGN KEY REFERENCES GreaterRegion_Dimension(gr_id)
);

drop table if exists regtemp;
create table regtemp(
   rgtid INT IDENTITY(1,1) PRIMARY KEY,
```

```
    region nvarchar(80),
    noc nvarchar(50) NOT NULL
)

INSERT INTO regtemp
SELECT distinct Country, NOC from fw;

DECLARE @tmax INT;
SET @tmax = (SELECT COUNT(Distinct Country) FROM fw)+1;
DECLARE @t INT = 1;
WHILE @t<=@tmax
BEGIN
    DECLARE @tgrid int;
    SET @tgrid =
        (SELECT gr_id
        FROM GreaterRegion_Dimension
        WHERE sub_region = (
            SELECT sub_region
            FROM Areas
            Where Name = (
                SELECT region
                FROM regtemp
                WHERE rgtid = @t
            )
        )
    );
    DECLARE @tncid nvarchar(50);
    SET @tncid = (SELECT NOC FROM regtemp WHERE rgtid = @t);
    INSERT INTO NOC_Dimension (NOC,gr_id) VALUES (@tncid,@tgrid);
    SET @t = @t+1;
END

UPDATE NOC_Dimension
```

```
SET gr_id = (SELECT gr_id FROM GreaterRegion_Dimension WHERE
sub_region = 'None')
WHERE gr_id is null;

GO

drop table if exists regtemp;

GO
```

## 4.4. Fact Table

Για την δημιουργία του fact table χρησιμοποιήθηκε ο παρακάτω κώδικας:

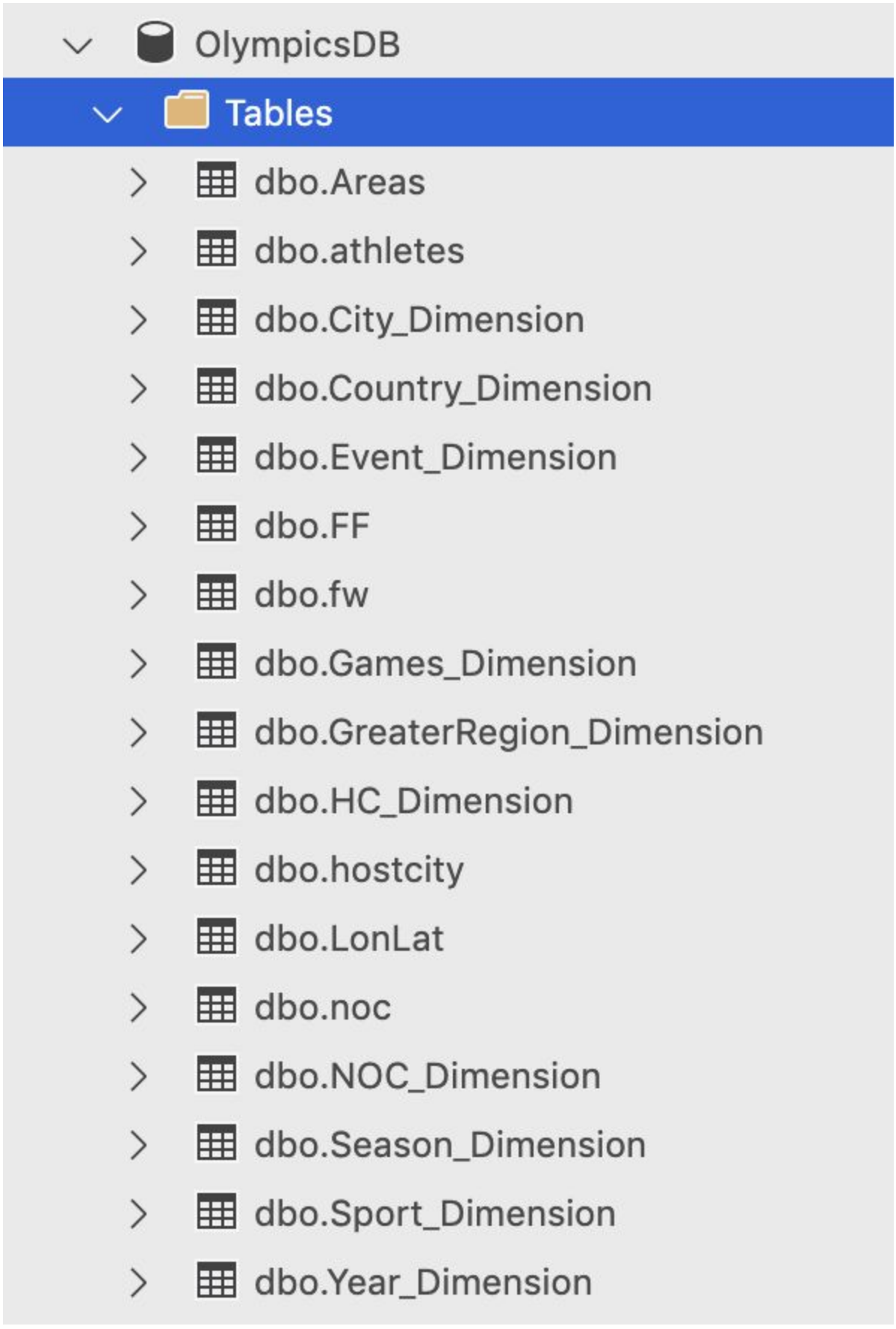
```
drop table if exists FF;
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[FF](
   [id] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,
   [Games] int FOREIGN KEY REFERENCES Games_Dimension(gm_id),
   [Season] int FOREIGN KEY REFERENCES Season_Dimension(ss_id),
   [City] int FOREIGN KEY REFERENCES City_Dimension(ct_id),
   [HC] int FOREIGN KEY REFERENCES HC_Dimension(hc_id),
   [lon] FLOAT,
   [lat] FLOAT,
   [Year] int FOREIGN KEY REFERENCES Year_Dimension(yr_id),
   [Sport] int FOREIGN KEY REFERENCES Sport_Dimension(sd_id),
   [Event] int FOREIGN KEY REFERENCES Event_Dimension(ev_id),
   [Country] int FOREIGN KEY REFERENCES Country_Dimension(cn_id),
```

```
   [NOC] int FOREIGN KEY REFERENCES NOC_Dimension(nc_id),
[Greater_Region]       int       FOREIGN       KEY       REFERENCES
GreaterRegion_Dimension(gr_id),
   [countF] INT,
   [countM] INT,
   [countParticipants] INT,
   [countGold] INT,
   [countSilver] INT,
   [countBronze] INT,
   [avgAge] FLOAT(24),
   [avgHeight] FLOAT(24),
   [avgWeight] FLOAT(24)
);
go
INSERT INTO FF (Games, Season, City, HC, lon, lat, Year, Sport,
Event, Country, NOC, Greater_Region, countF, countM,
countParticipants, countGold, countSilver, countBronze, avgAge,
avgHeight, avgWeight)
SELECT
   Games_Dimension.gm_id,
   Season_Dimension.ss_id,
   City_Dimension.ct_id,
   HC_Dimension.hc_id, fw.Longitude, fw.Latitude,
   Year_Dimension.yr_id,
   Sport_Dimension.sd_id,
   Event_Dimension.ev_id,
   Country_Dimension.cn_id,
   NOC_Dimension.nc_id,
   GreaterRegion_Dimension.gr_id,

fw.[countF],fw.[countM],fw.[countParticipants],fw.[countGold],fw.[cou
ntSilver],fw.[countBronze],fw.[avgAge],
   fw.[avgHeight],fw.[avgWeight]
```

```
FROM
      fw,  NOC_Dimension,  GreaterRegion_Dimension,  Sport_Dimension,
Country_Dimension, HC_Dimension, Games_Dimension,
   Season_Dimension, Year_Dimension, City_Dimension, Event_Dimension
WHERE
   fw.[NOC] =  NOC_Dimension.NOC AND
   GreaterRegion_Dimension.gr_id = NOC_Dimension.gr_id AND
   fw.[Sport] = Sport_Dimension.sport AND
   fw.[Country] = Country_Dimension.country AND
   fw.[Games] = Games_Dimension.games AND
   fw.[Season] = Season_Dimension.season AND
   fw.[Year] = Year_Dimension.[year] AND
   fw.[City] = City_Dimension.city AND
   fw.[HC] = HC_Dimension.hostCountry AND
   fw.[Event] = Event_Dimension.event;
```

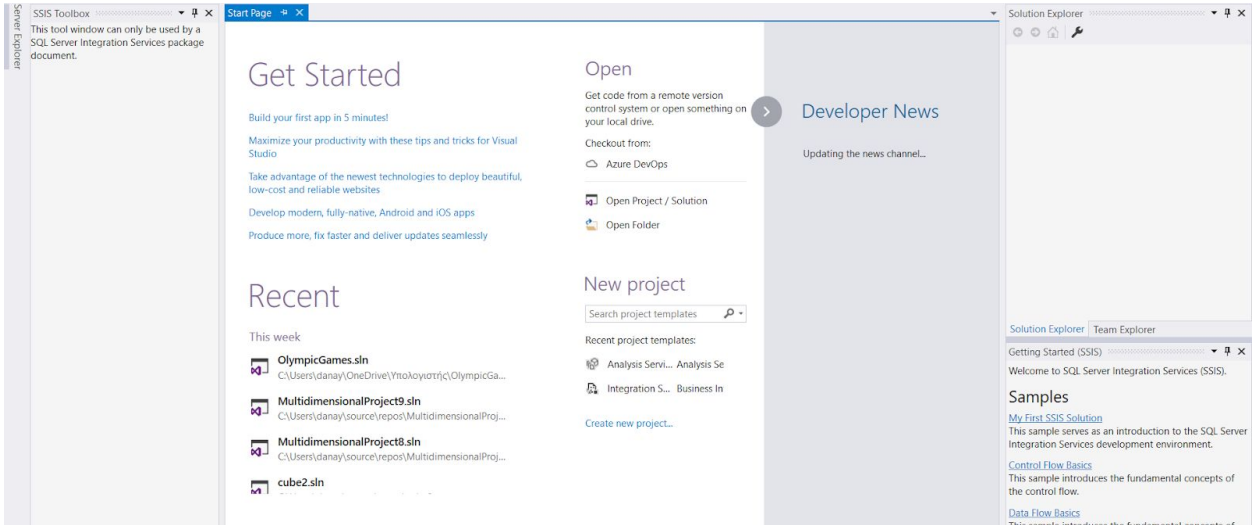
After creating all tables, including the fact table:

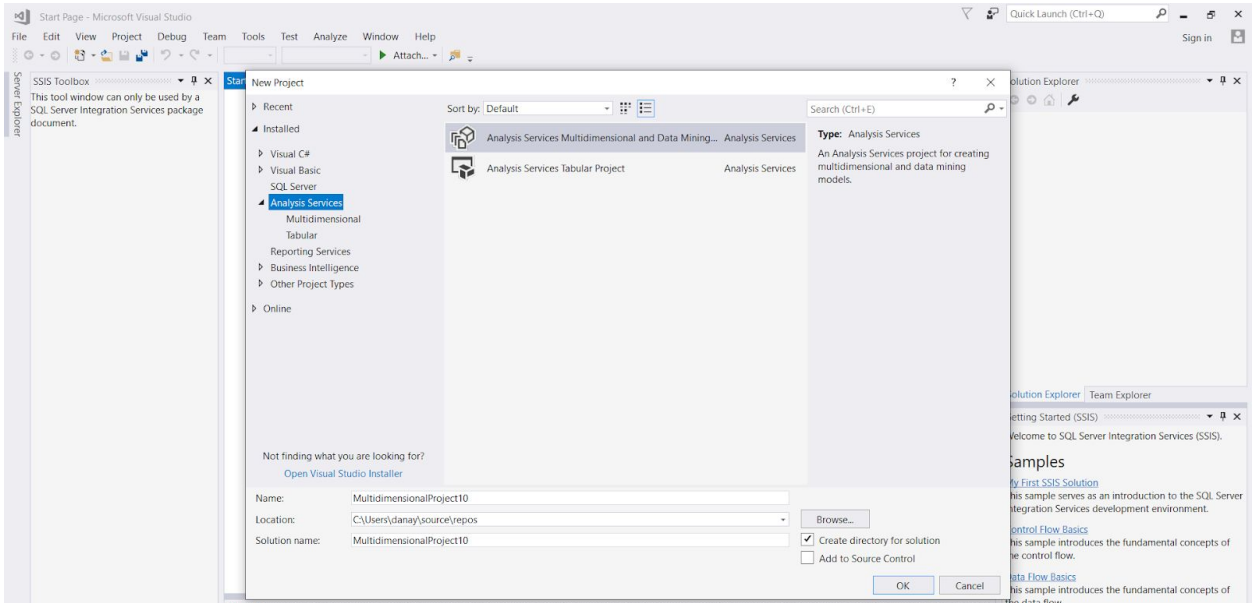

OlympicsDB
Tables
dbo.Areas
dbo.athletes
dbo.City_Dimension
dbo.Country_Dimension
dbo.Event_Dimension
dbo.FF
dbo.fw
dbo.Games_Dimension
dbo.GreaterRegion_Dimension
dbo.HC_Dimension
dbo.hostcity
dbo.LonLat
dbo.noc
dbo.NOC_Dimension
dbo.Season_Dimension
dbo.Sport_Dimension
dbo.Year_Dimension

# 5. Data Cube (Visual Studio 2017)

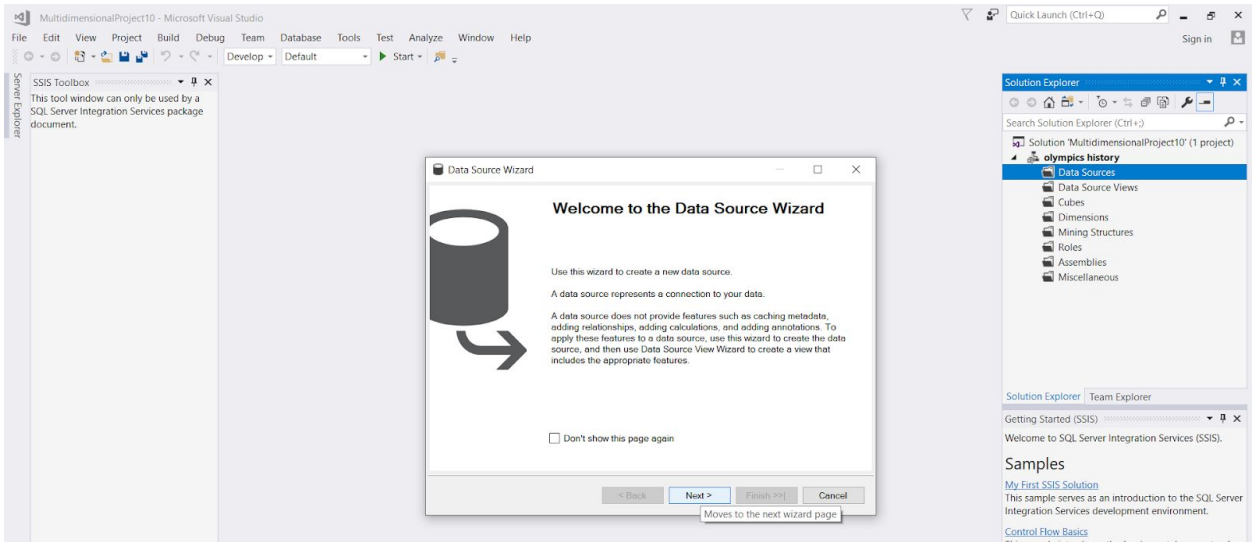
**Step 1:** We create the cube with Visual Studio 2017.

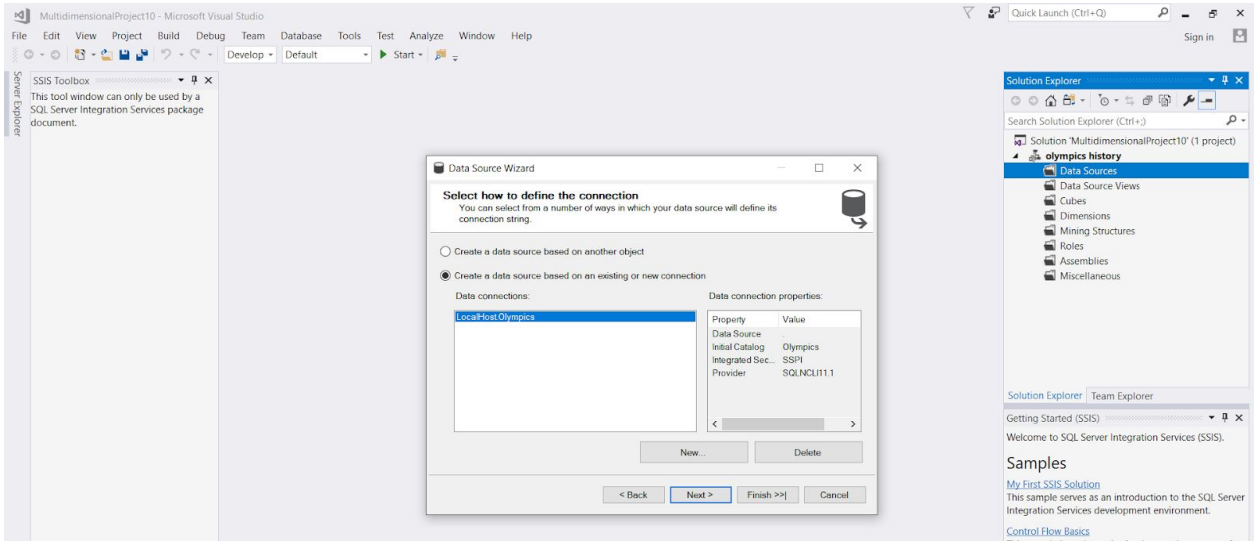
Select “Create new project” → “Analysis Services” → “Analysis Services Multidimensional Data Mining”.

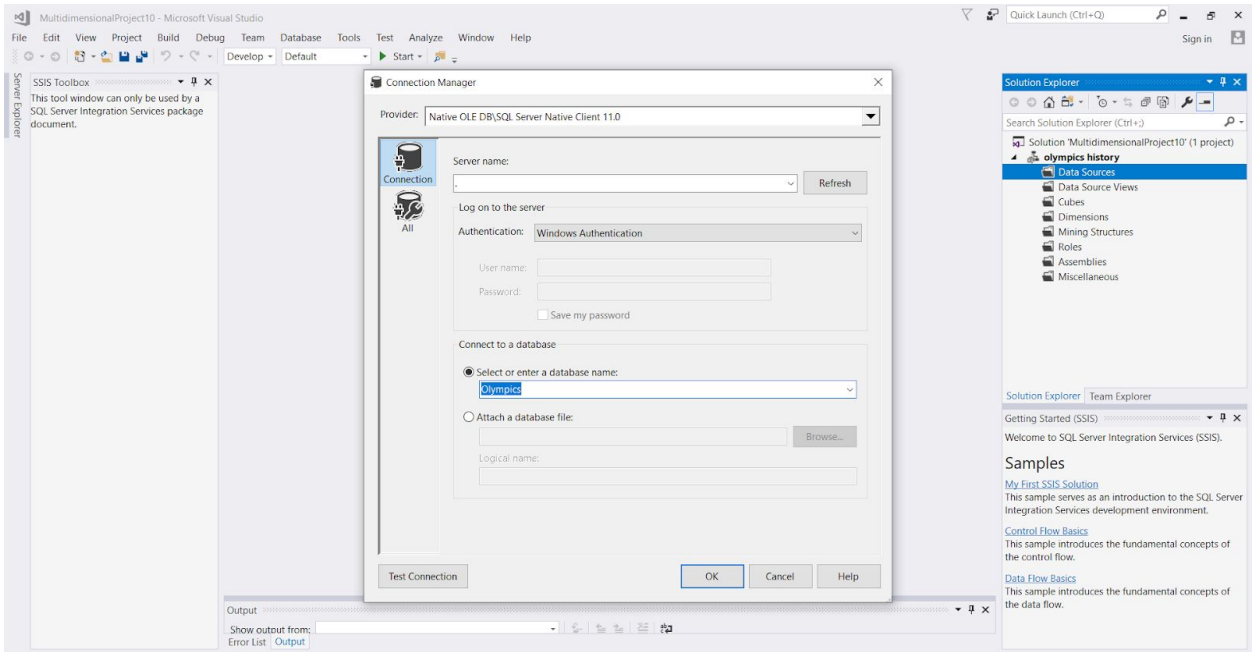


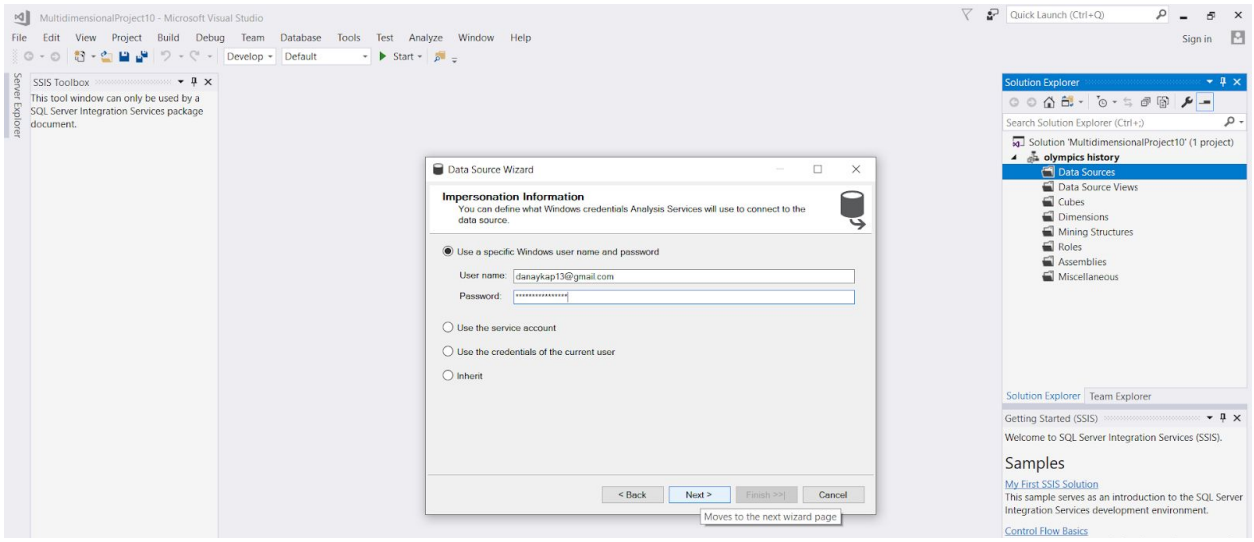
**Step 2:** We will open our database following the steps below.

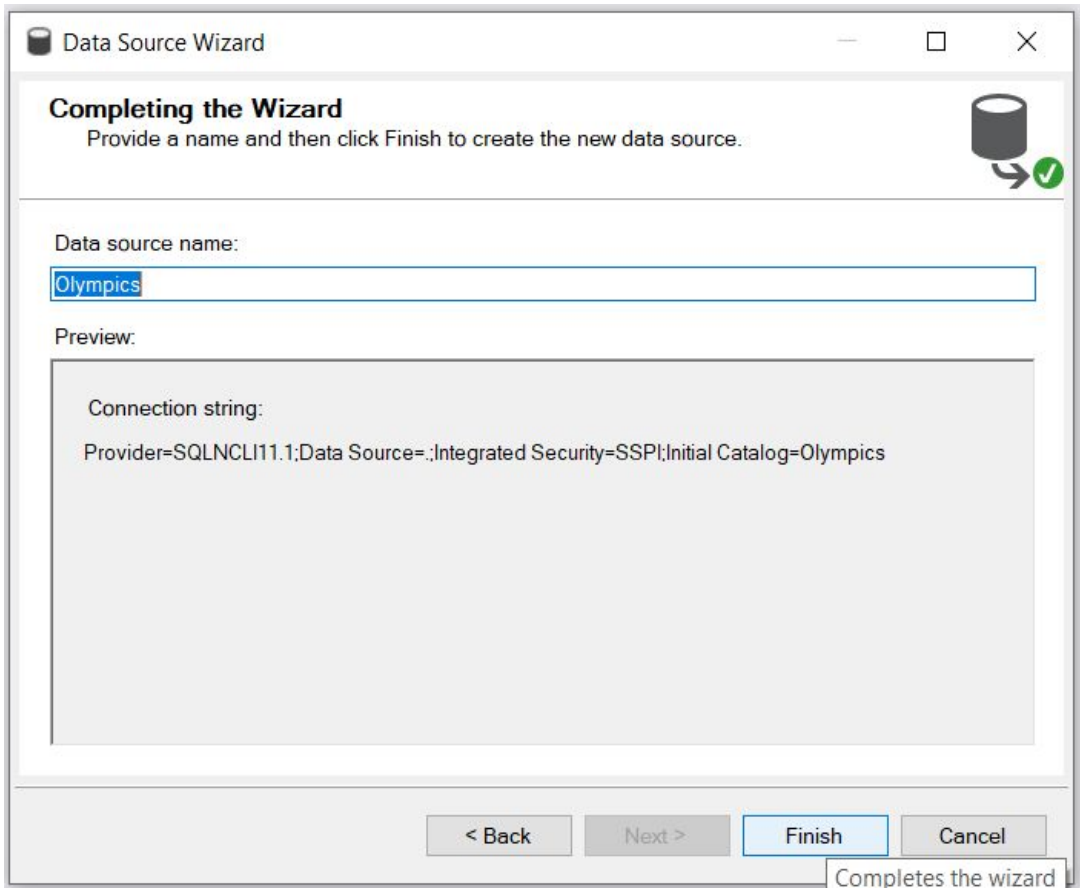


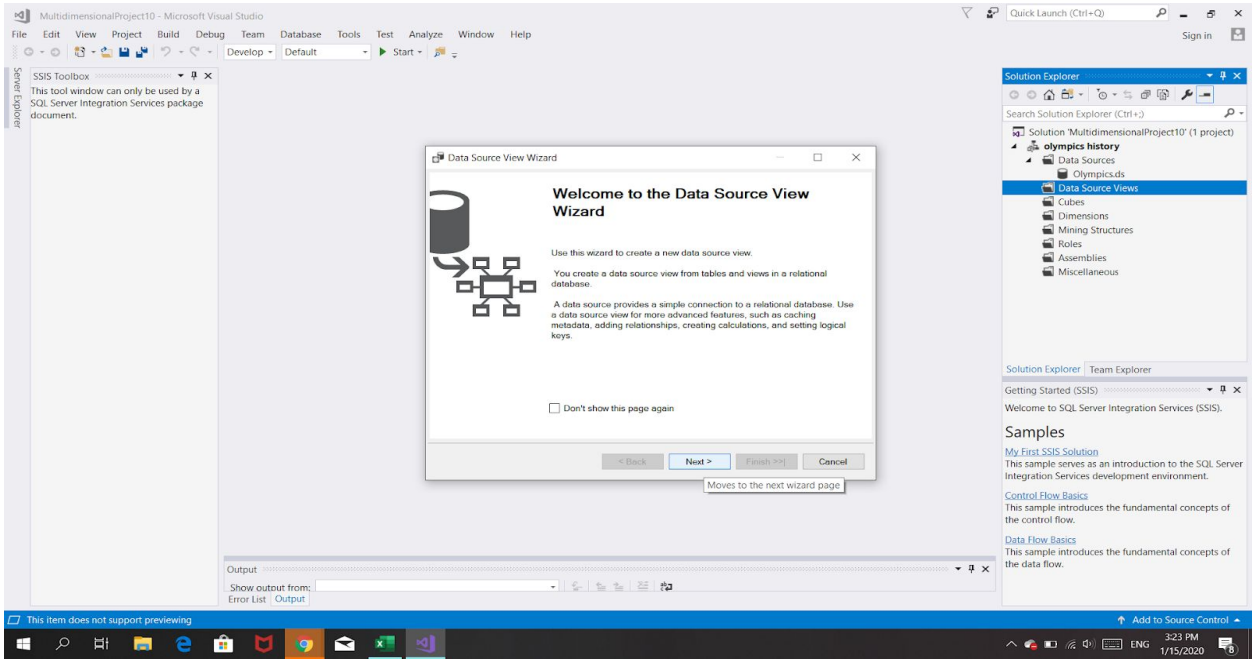

MultidimensionalProject10 - Microsoft Visual Studio
File Edit View Project Build Debug Team Database Tools Test Analyze Window Help
Quick Launch (Ctrl+Q)
Sign in
Develop
Default
Start
SSIS Toolbox
This tool window can only be used by a SQL Server Integration Services package document.
Connection Manager
Provider:
Native OLE DB\SQL Server Native Client 11.0
Connection
All
Server name:
Refresh
Log on to the server
Authentication:
Windows Authentication
User name:
Password:
Save my password
Connect to a database
Select or enter a database name:
Olympics
Attach a database file:
Browse...
Logical name:
Test Connection
OK
Cancel
Help
Output
Show output from:
Error List
Output
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MultidimensionalProject10' (1 project)
olympics history
Data Sources
Data Source Views
Cubes
Dimensions
Mining Structures
Roles
Assemblies
Miscellaneous
Solution Explorer
Team Explorer
Getting Started (SSIS)
Welcome to SQL Server Integration Services (SSIS).
Samples
My First SSIS Solution
This sample serves as an introduction to the SQL Server Integration Services development environment.
Control Flow Basics
This sample introduces the fundamental concepts of the control flow.
Data Flow Basics
This sample introduces the fundamental concepts of the data flow.

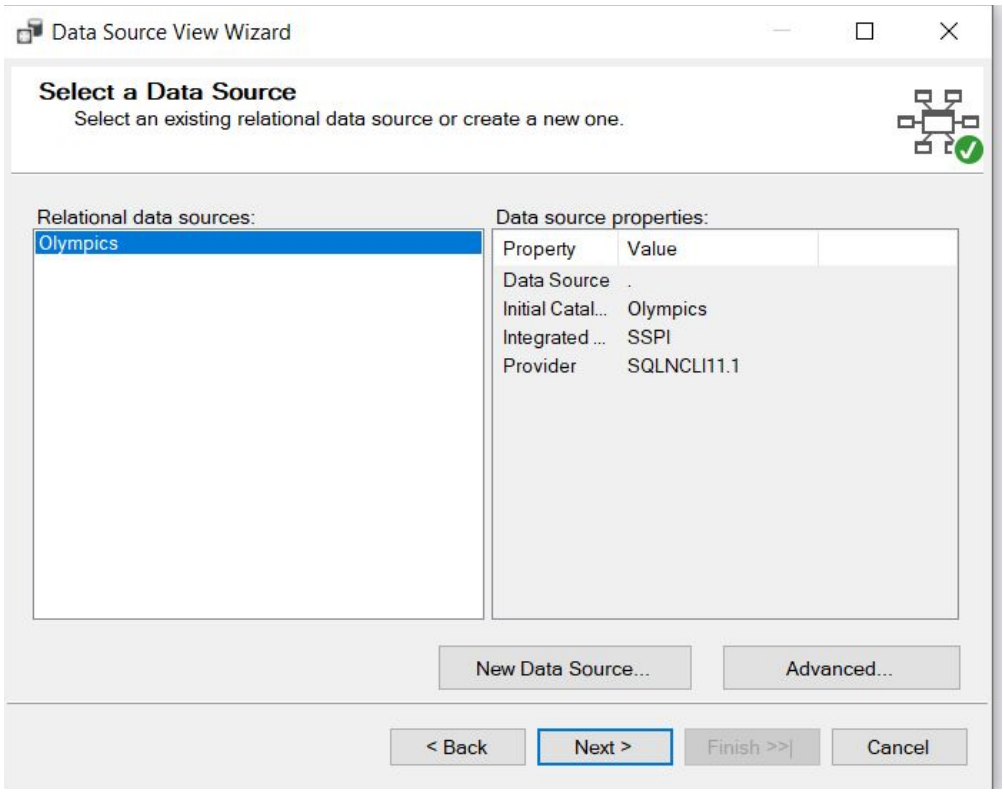
**Important :** We need to sign in with our windows credentials.

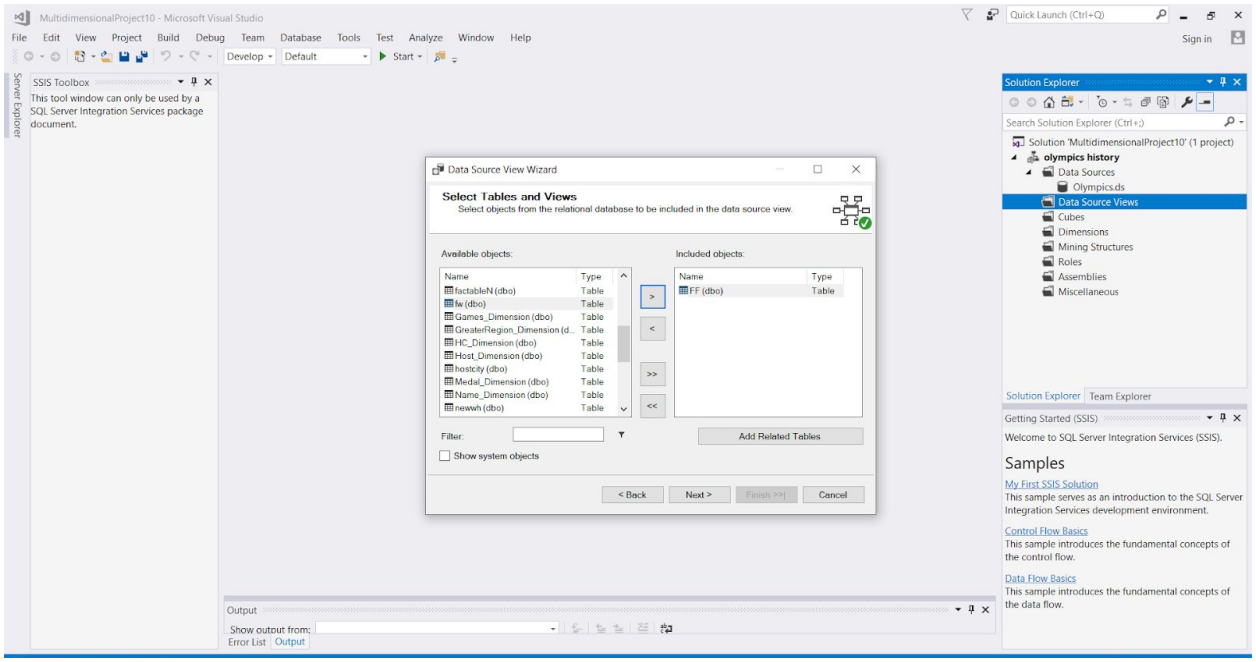


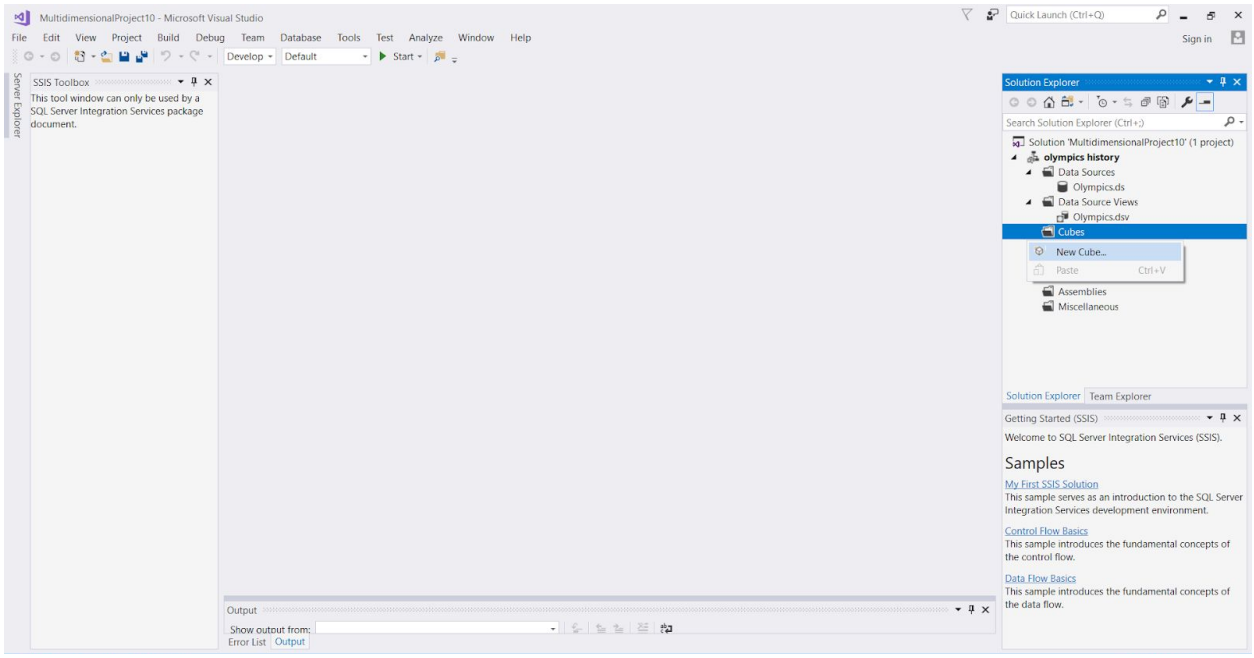
**Step 3:** We create a data source view from our database, by importing our fact table and all other related tables.

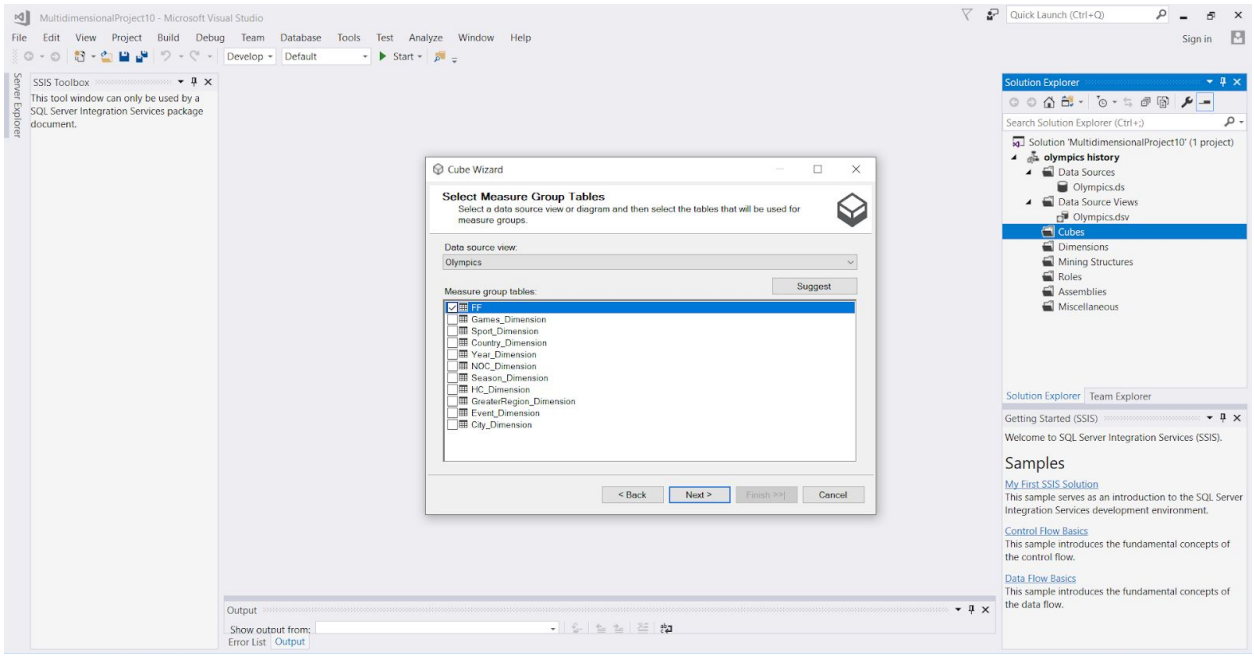


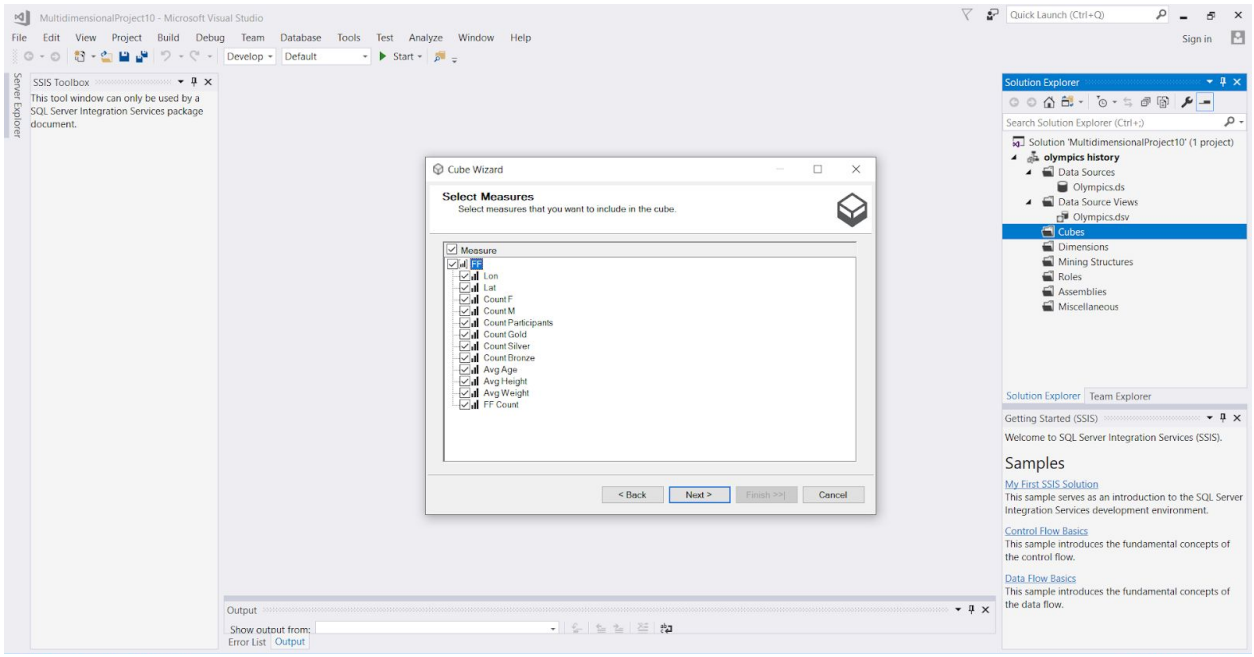



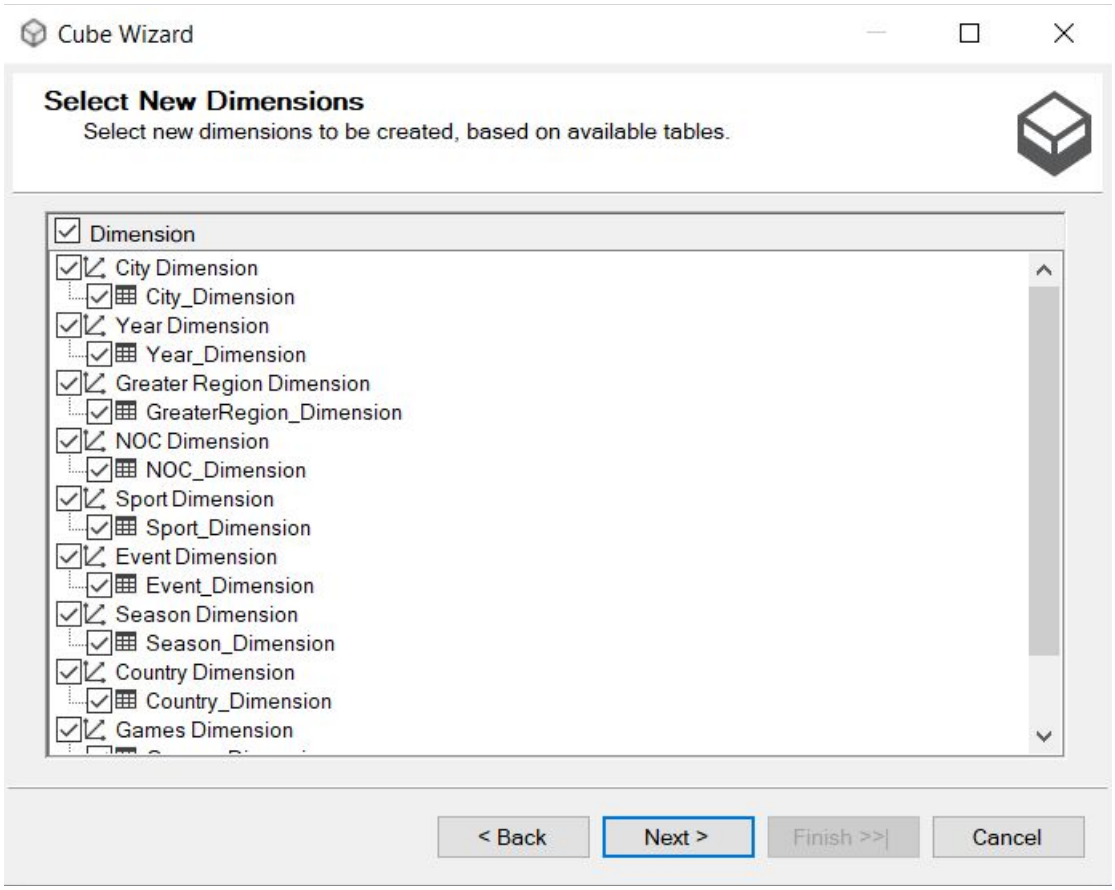

MultidimensionalProject10 - Microsoft Visual Studio
File Edit View Project Build Debug Team Database Tools Test Analyze Window Help
Quick Launch (Ctrl+Q)
Sign in
Develop
Default
Start
SSIS Toolbox
This tool window can only be used by a SQL Server Integration Services package document.
Data Source View Wizard
Select Tables and Views
Select objects from the relational database to be included in the data source view.
Available objects:
Name
Type
factableN (dbo)
fw (dbo)
Games_Dimension (dbo)
GreaterRegion_Dimension (d...
HC_Dimension (dbo)
Host_Dimension (dbo)
hostcity (dbo)
Medal_Dimension (dbo)
Name_Dimension (dbo)
newwh (dbo)
Table
Included objects:
FF (dbo)
Filter:
Show system objects
Add Related Tables
< Back
Next >
Finish >>|
Cancel
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MultidimensionalProject10' (1 project)
olympics history
Data Sources
Olympics.ds
Data Source Views
Cubes
Dimensions
Mining Structures
Roles
Assemblies
Miscellaneous
Solution Explorer
Team Explorer
Getting Started (SSIS)
Welcome to SQL Server Integration Services (SSIS).
Samples
My First SSIS Solution
This sample serves as an introduction to the SQL Server Integration Services development environment.
Control Flow Basics
This sample introduces the fundamental concepts of the control flow.
Data Flow Basics
This sample introduces the fundamental concepts of the data flow.
Output
Show output from:
Error List
Output

**Step 4:** We will create our cube, by importing all dimensions and measures of the fact table.

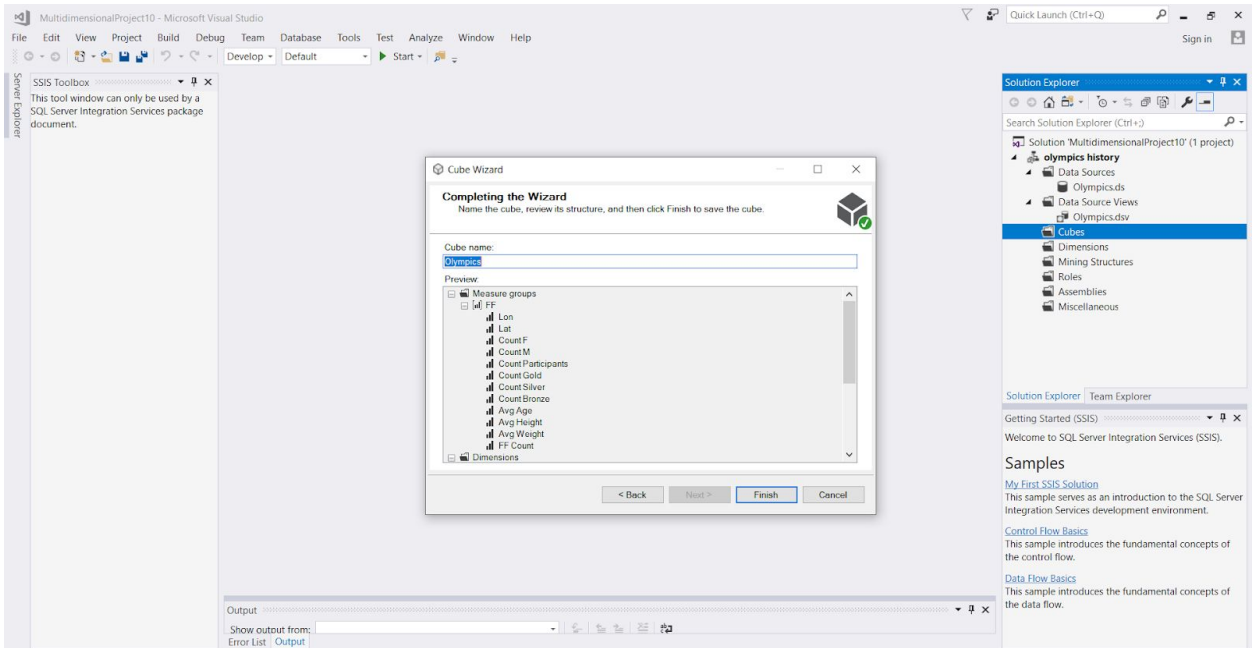


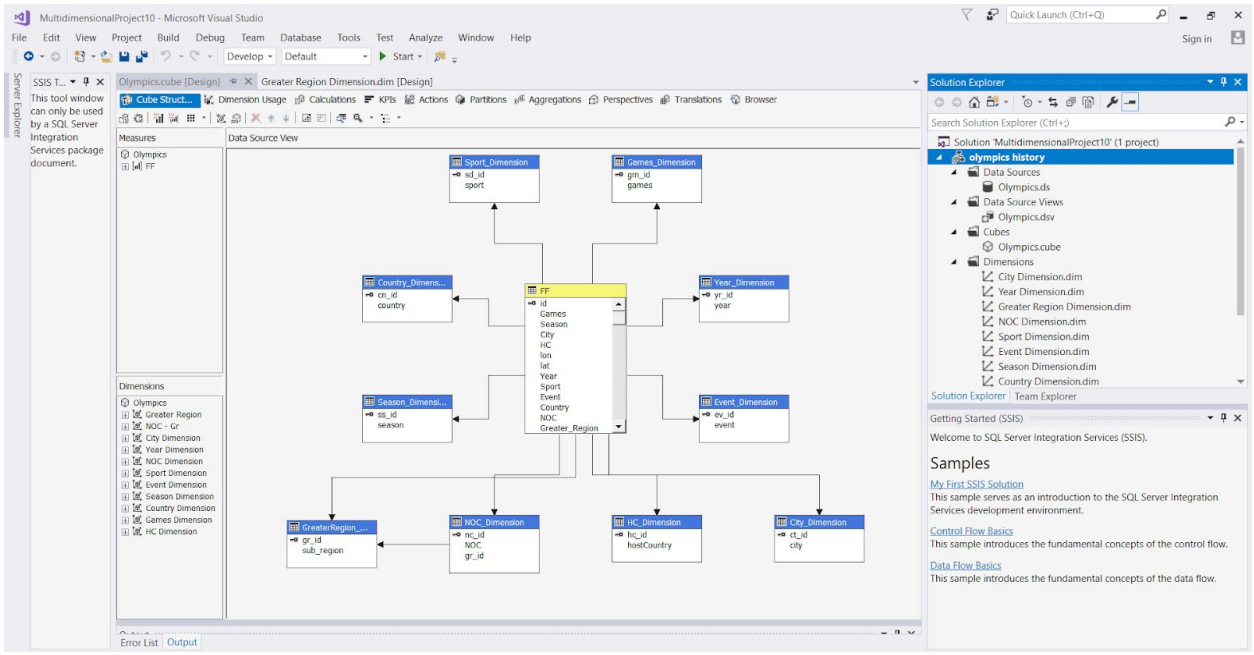

MultidimensionalProject10 - Microsoft Visual Studio
File Edit View Project Build Debug Team Database Tools Test Analyze Window Help
SSIS Toolbox
This tool window can only be used by a SQL Server Integration Services package document.
Cube Wizard
Select Measures
Select measures that you want to include in the cube.
Measure
FF
Lon
Lat
Count F
Count M
Count Participants
Count Gold
Count Silver
Count Bronze
Avg Age
Avg Height
Avg Weight
FF Count
< Back
Next >
Finish >>|
Cancel
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MultidimensionalProject10' (1 project)
olympics history
Data Sources
Olympics.ds
Data Source Views
Olympics.dsv
Cubes
Dimensions
Mining Structures
Roles
Assemblies
Miscellaneous
Solution Explorer
Team Explorer
Getting Started (SSIS)
Welcome to SQL Server Integration Services (SSIS).
Samples
My First SSIS Solution
This sample serves as an introduction to the SQL Server Integration Services development environment.
Control Flow Basics
This sample introduces the fundamental concepts of the control flow.
Data Flow Basics
This sample introduces the fundamental concepts of the data flow.
Output
Show output from:
Error List
Output


Cube Wizard
Select New Dimensions
Select new dimensions to be created, based on available tables.
Dimension
City Dimension
City_Dimension
Year Dimension
Year_Dimension
Greater Region Dimension
GreaterRegion_Dimension
NOC Dimension
NOC_Dimension
Sport Dimension
Sport_Dimension
Event Dimension
Event_Dimension
Season Dimension
Season_Dimension
Country Dimension
Country_Dimension
Games Dimension
< Back
Next >
Finish >>|
Cancel

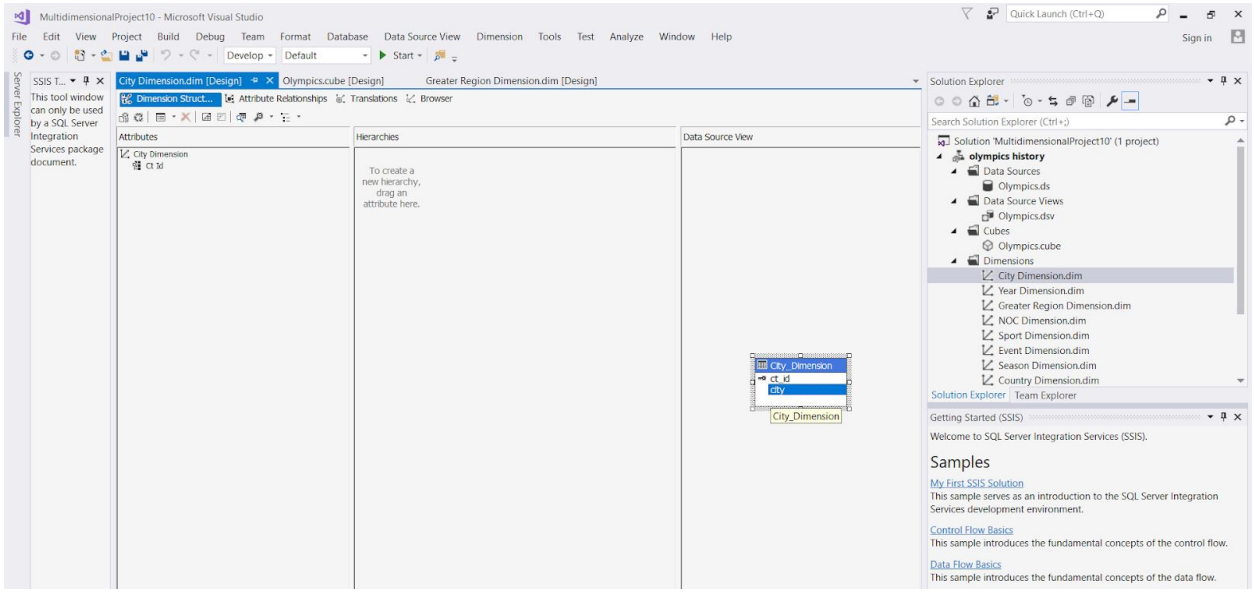
lv

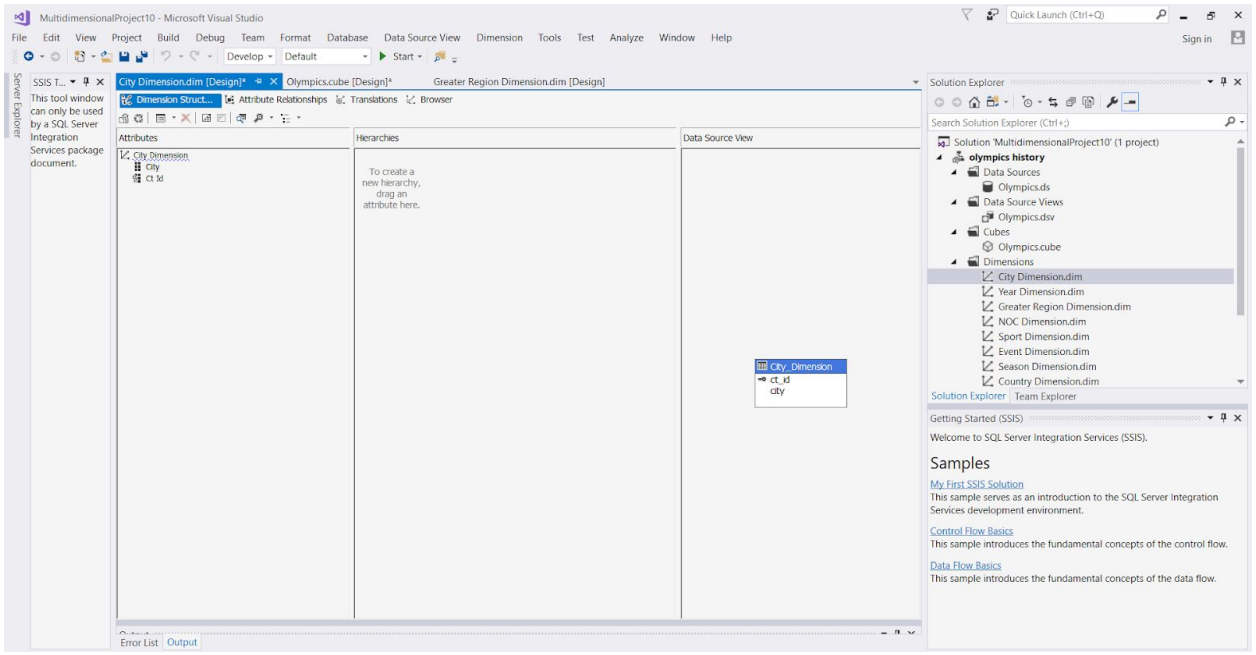


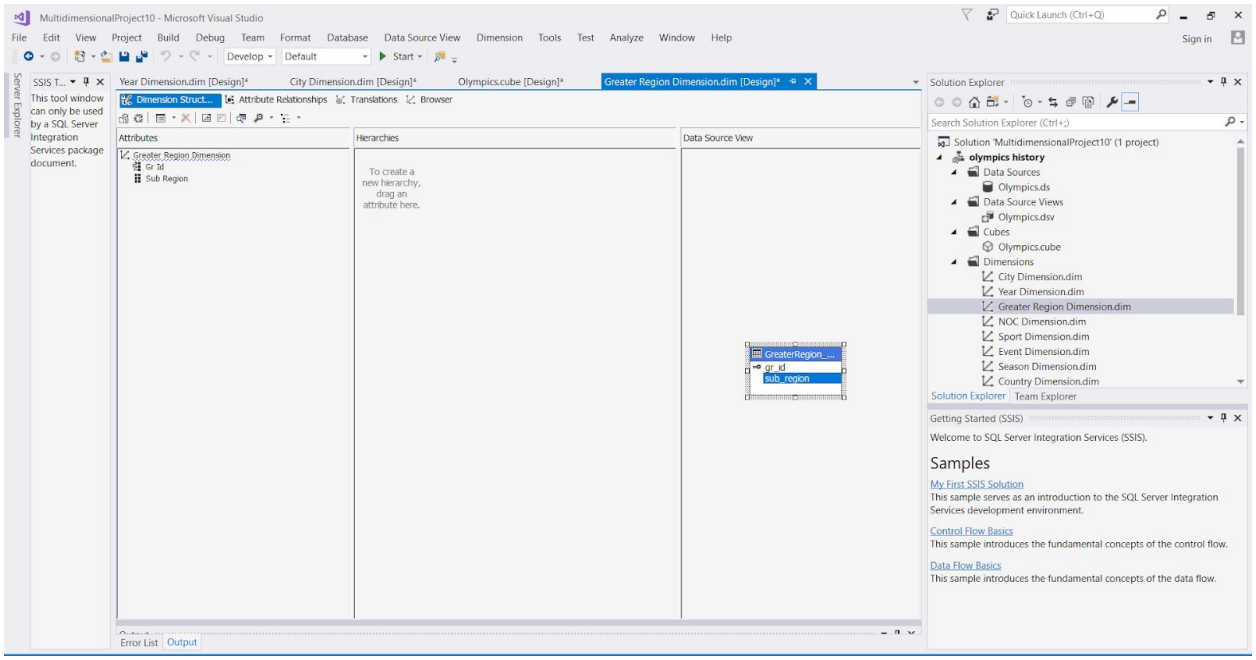

In the following picture, there is our cube.

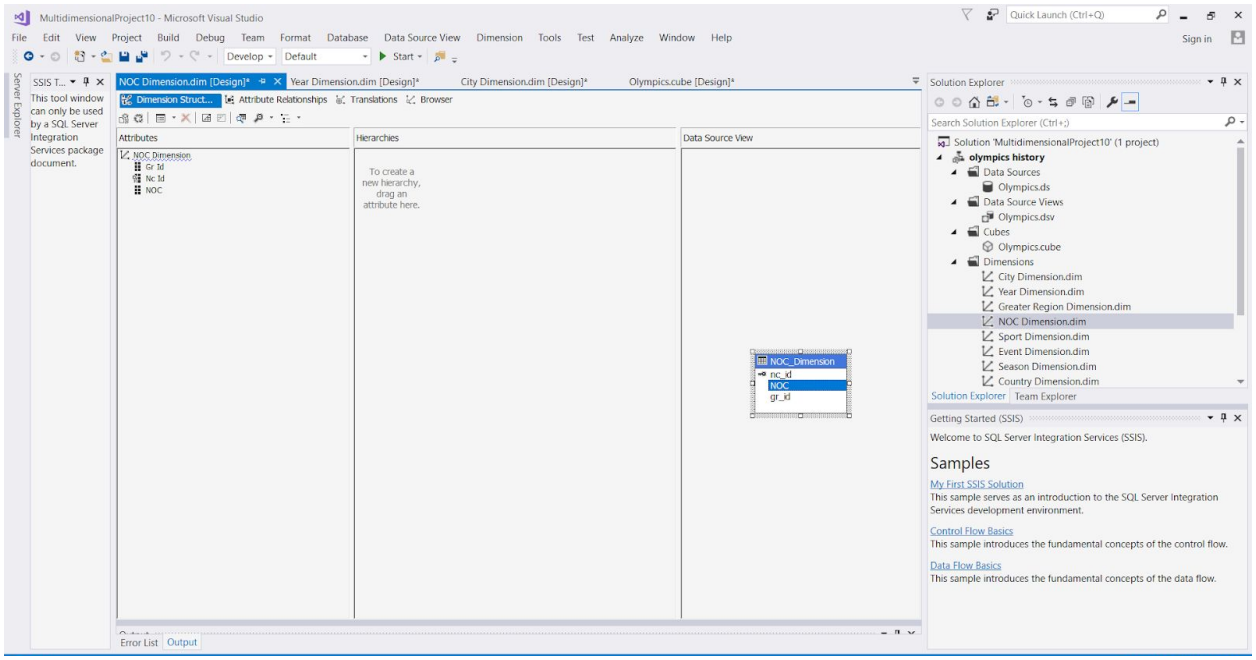


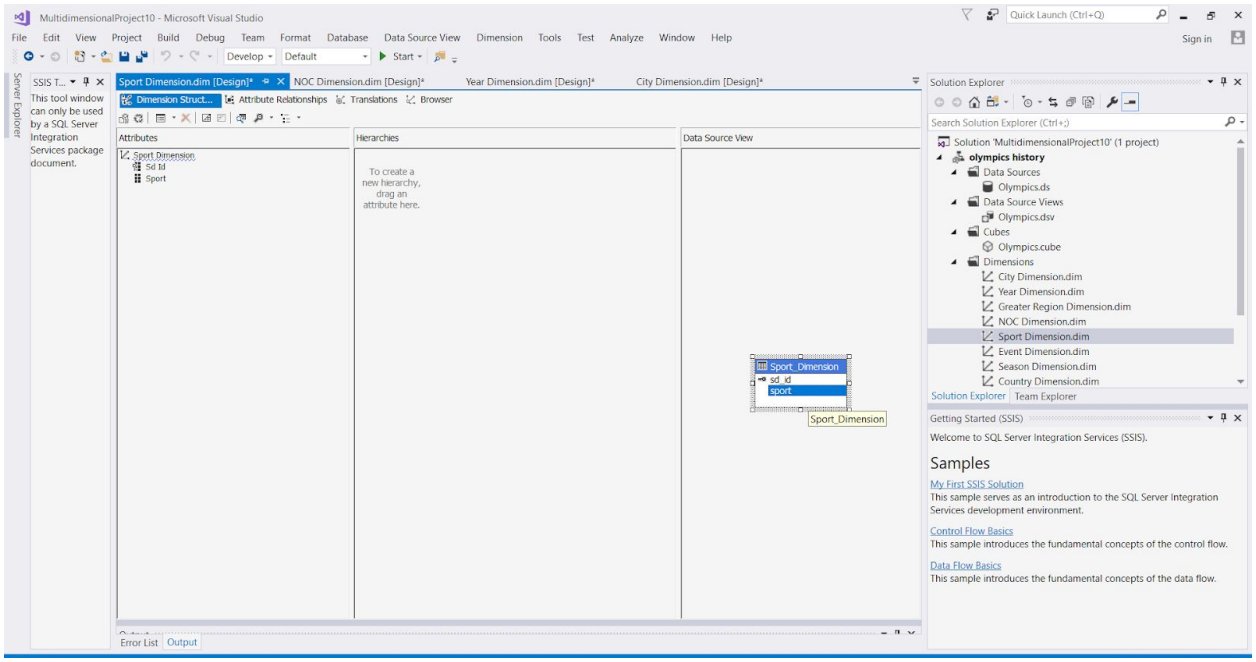
At the beginning, the results of the cube were we the IDs of dimensions. To change that and get back values of our fields instead of IDs we followed these steps:

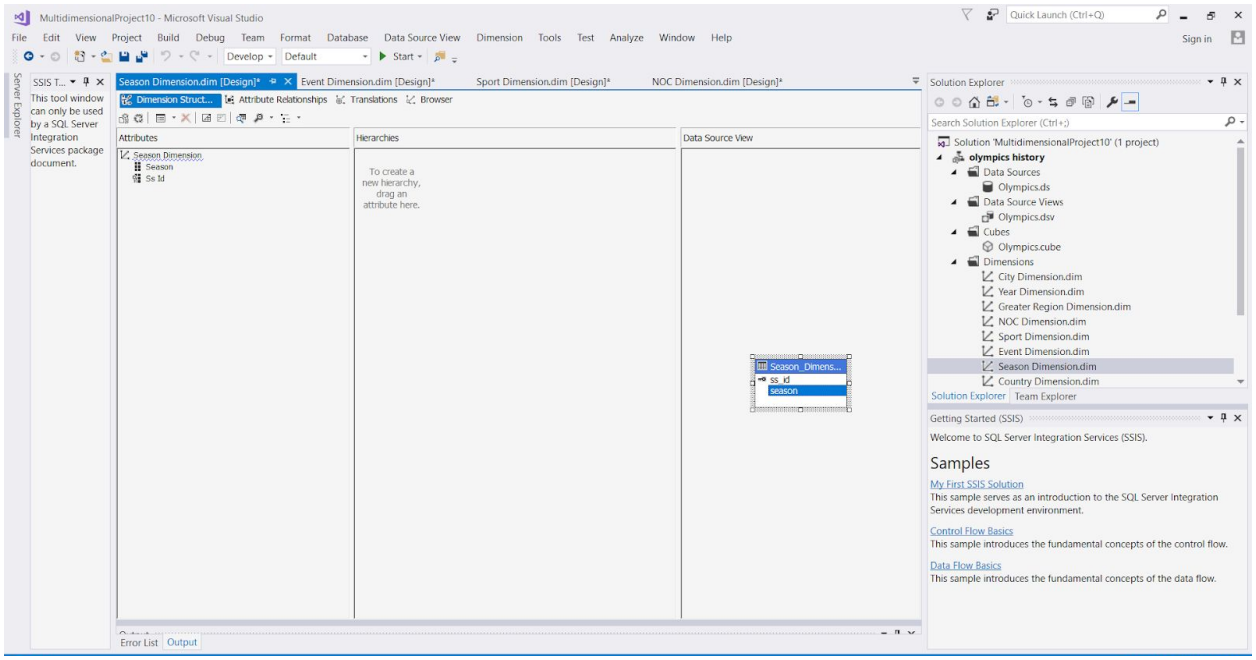


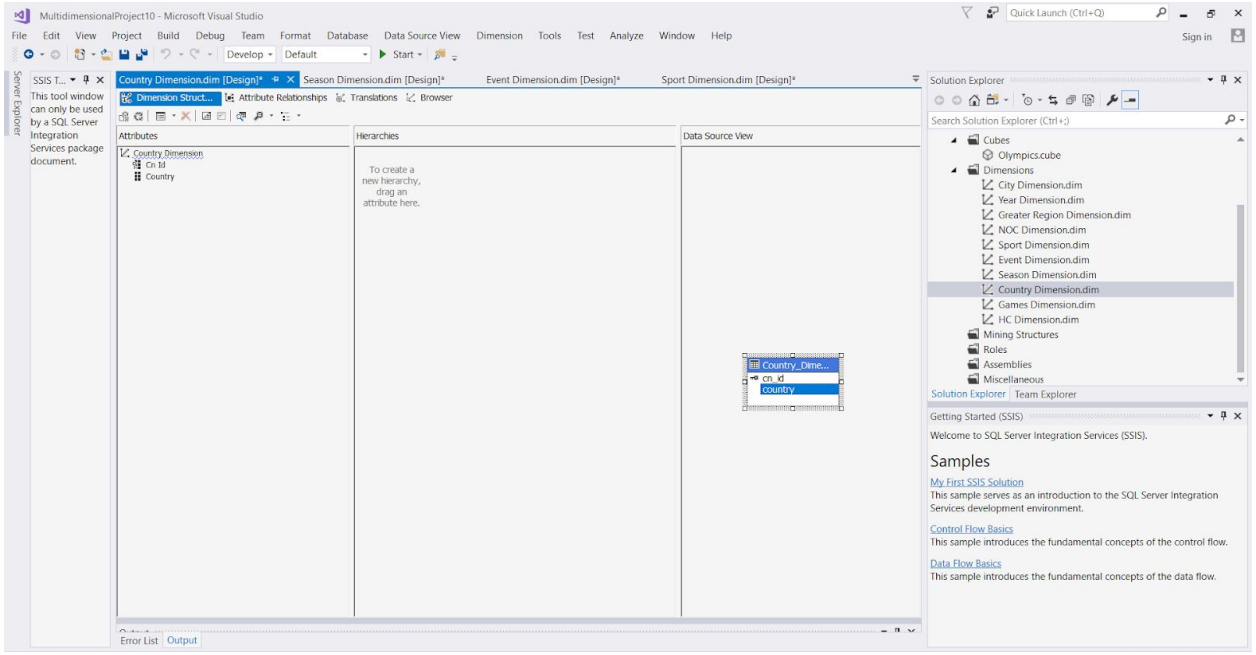

MultidimensionalProject10 - Microsoft Visual Studio
File Edit View Project Build Debug Team Format Database Data Source View Dimension Tools Test Analyze Window Help
Sign in
Quick Launch (Ctrl+Q)
Develop
Default
Start
SSIS T...
This tool window can only be used by a SQL Server Integration Services package document.
City Dimension.dim [Design]*
Olympics.cube [Design]*
Greater Region Dimension.dim [Design]
Dimension Struct...
Attribute Relationships
Translations
Browser
Attributes
Hierarchies
Data Source View
City Dimension
City
Ct Id
To create a new hierarchy, drag an attribute here.
City_Dimension
ct_id
city
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MultidimensionalProject10' (1 project)
olympics history
Data Sources
Olympics.ds
Data Source Views
Olympics.dsv
Cubes
Olympics.cube
Dimensions
City Dimension.dim
Year Dimension.dim
Greater Region Dimension.dim
NOC Dimension.dim
Sport Dimension.dim
Event Dimension.dim
Season Dimension.dim
Country Dimension.dim
Solution Explorer
Team Explorer
Getting Started (SSIS)
Welcome to SQL Server Integration Services (SSIS).
Samples
My First SSIS Solution
This sample serves as an introduction to the SQL Server Integration Services development environment.
Control Flow Basics
This sample introduces the fundamental concepts of the control flow.
Data Flow Basics
This sample introduces the fundamental concepts of the data flow.
Error List
Output

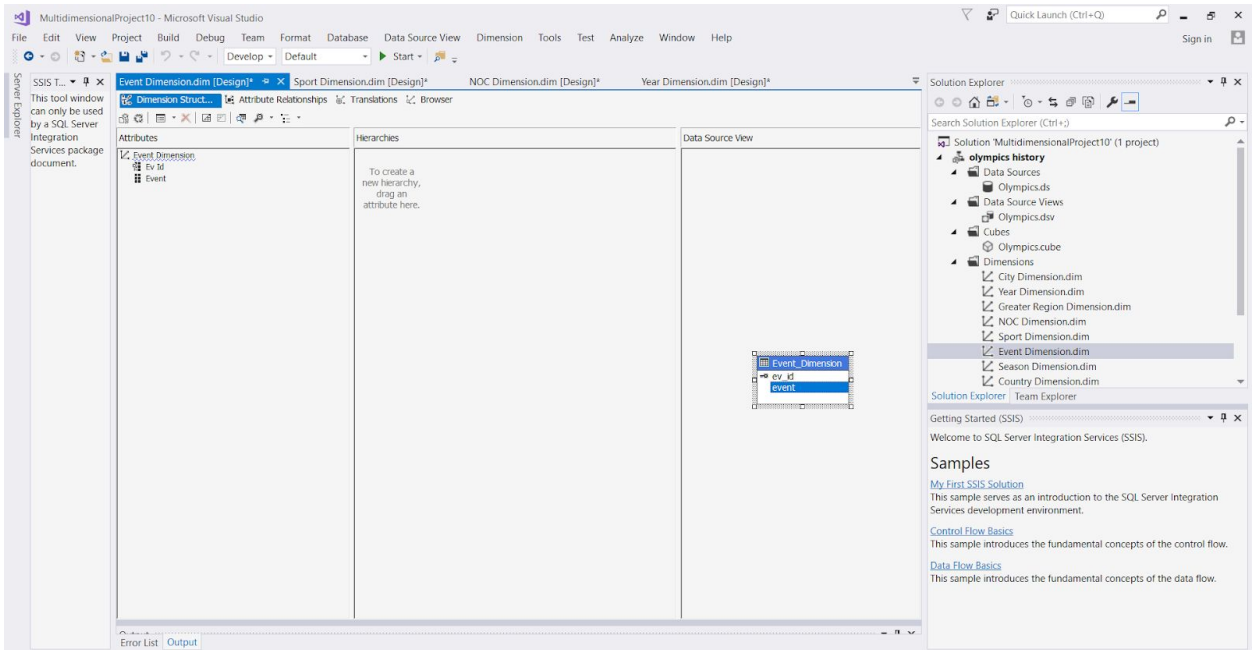

MultidimensionalProject10 - Microsoft Visual Studio
File Edit View Project Build Debug Team Format Database Data Source View Dimension Tools Test Analyze Window Help
Sign in
Quick Launch (Ctrl+Q)
Develop
Default
Start
SSIS T...
This tool window can only be used by a SQL Server Integration Services package document.
Year Dimension.dim [Design]*
City Dimension.dim [Design]*
Olympics.cube [Design]*
Greater Region Dimension.dim [Design]*
Dimension Struct...
Attribute Relationships
Translations
Browser
Attributes
Hierarchies
Data Source View
Greater Region Dimension
Gr Id
Sub Region
To create a new hierarchy, drag an attribute here.
GreaterRegion_...
gr_id
sub_region
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MultidimensionalProject10' (1 project)
olympics history
Data Sources
Olympics.ds
Data Source Views
Olympics.dsv
Cubes
Olympics.cube
Dimensions
City Dimension.dim
Year Dimension.dim
Greater Region Dimension.dim
NOC Dimension.dim
Sport Dimension.dim
Event Dimension.dim
Season Dimension.dim
Country Dimension.dim
Solution Explorer
Team Explorer
Getting Started (SSIS)
Welcome to SQL Server Integration Services (SSIS).
Samples
My First SSIS Solution
This sample serves as an introduction to the SQL Server Integration Services development environment.
Control Flow Basics
This sample introduces the fundamental concepts of the control flow.
Data Flow Basics
This sample introduces the fundamental concepts of the data flow.
Error List
Output

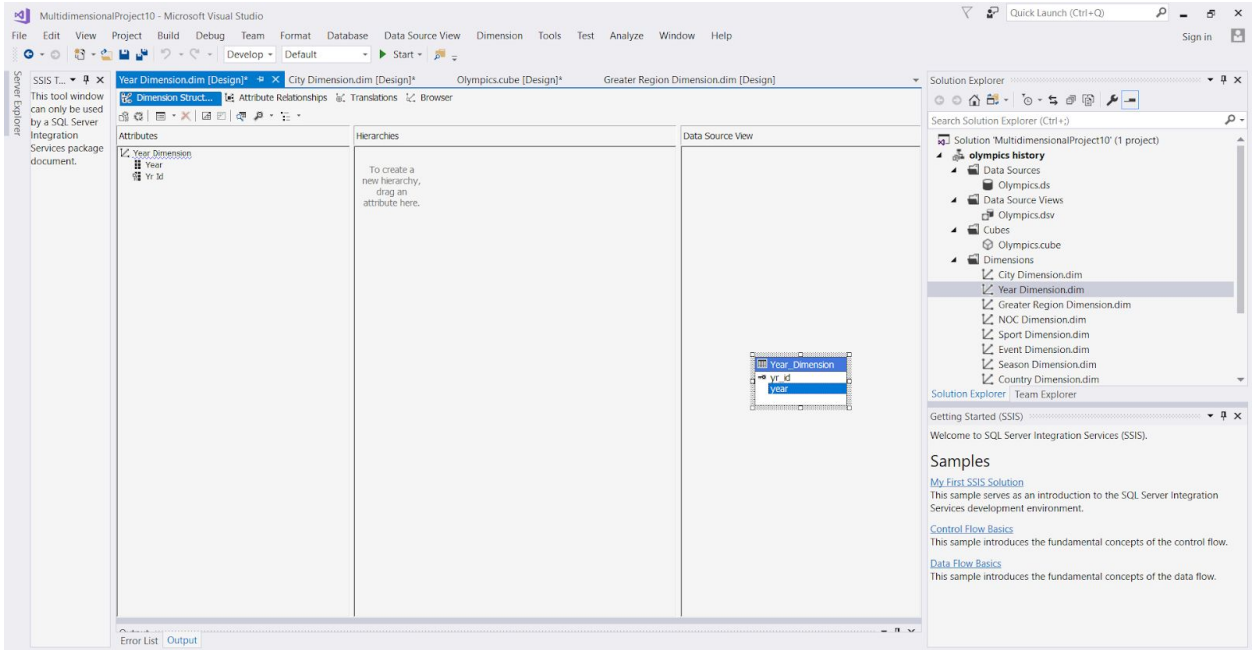

MultidimensionalProject10 - Microsoft Visual Studio
File Edit View Project Build Debug Team Format Database Data Source View Dimension Tools Test Analyze Window Help
Sign in
Develop Default Start
NOC Dimension.dim [Design]* Year Dimension.dim [Design]* City Dimension.dim [Design]* Olympics.cube [Design]*
Dimension Struct... Attribute Relationships Translations Browser
Attributes
Hierarchies
Data Source View
NOC Dimension
Gr Id
Nc Id
NOC
To create a new hierarchy, drag an attribute here.
NOC_Dimension
nc_id
NOC
gr_id
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MultidimensionalProject10' (1 project)
olympics history
Data Sources
Olympics.ds
Data Source Views
Olympics.dsv
Cubes
Olympics.cube
Dimensions
City Dimension.dim
Year Dimension.dim
Greater Region Dimension.dim
NOC Dimension.dim
Sport Dimension.dim
Event Dimension.dim
Season Dimension.dim
Country Dimension.dim
Solution Explorer Team Explorer
Getting Started (SSIS)
Welcome to SQL Server Integration Services (SSIS).
Samples
My First SSIS Solution
This sample serves as an introduction to the SQL Server Integration Services development environment.
Control Flow Basics
This sample introduces the fundamental concepts of the control flow.
Data Flow Basics
This sample introduces the fundamental concepts of the data flow.
Error List Output


MultidimensionalProject10 - Microsoft Visual Studio
File Edit View Project Build Debug Team Format Database Data Source View Dimension Tools Test Analyze Window Help
Sign in
Develop Default Start
Sport Dimension.dim [Design]* NOC Dimension.dim [Design]* Year Dimension.dim [Design]* City Dimension.dim [Design]*
Dimension Struct... Attribute Relationships Translations Browser
Attributes
Hierarchies
Data Source View
Sport Dimension
Sd Id
Sport
To create a new hierarchy, drag an attribute here.
Sport_Dimension
sd_id
sport
Sport_Dimension
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'MultidimensionalProject10' (1 project)
olympics history
Data Sources
Olympics.ds
Data Source Views
Olympics.dsv
Cubes
Olympics.cube
Dimensions
City Dimension.dim
Year Dimension.dim
Greater Region Dimension.dim
NOC Dimension.dim
Sport Dimension.dim
Event Dimension.dim
Season Dimension.dim
Country Dimension.dim
Solution Explorer Team Explorer
Getting Started (SSIS)
Welcome to SQL Server Integration Services (SSIS).
Samples
My First SSIS Solution
This sample serves as an introduction to the SQL Server Integration Services development environment.
Control Flow Basics
This sample introduces the fundamental concepts of the control flow.
Data Flow Basics
This sample introduces the fundamental concepts of the data flow.
Error List Output

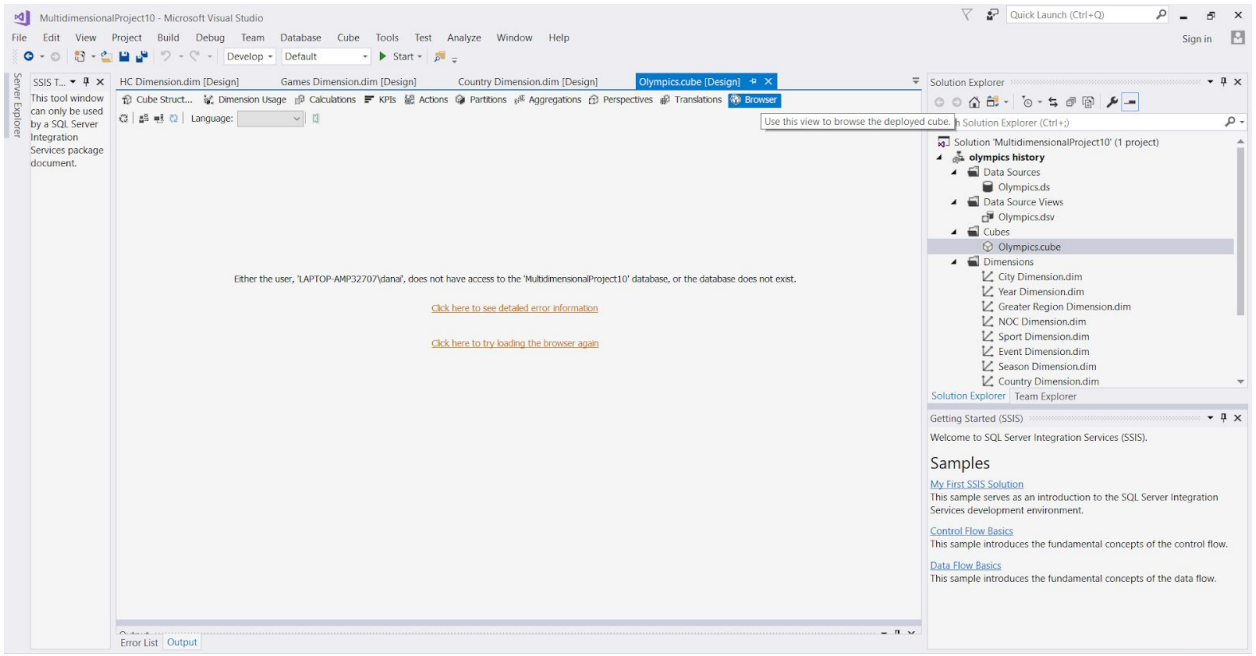

MultidimensionalProject10 - Microsoft Visual Studio
Season Dimension.dim [Design]*
Event Dimension.dim [Design]*
Sport Dimension.dim [Design]*
NOC Dimension.dim [Design]*
Attributes
Hierarchies
Data Source View
Season Dimension
Season
Ss Id
To create a new hierarchy, drag an attribute here.
Season_Dimens...
ss_id
season
Solution Explorer
olympics history
Data Sources
Olympics.ds
Data Source Views
Olympics.dsv
Cubes
Olympics.cube
Dimensions
City Dimension.dim
Year Dimension.dim
Greater Region Dimension.dim
NOC Dimension.dim
Sport Dimension.dim
Event Dimension.dim
Season Dimension.dim
Country Dimension.dim
Getting Started (SSIS)
Samples
My First SSIS Solution
Control Flow Basics
Data Flow Basics

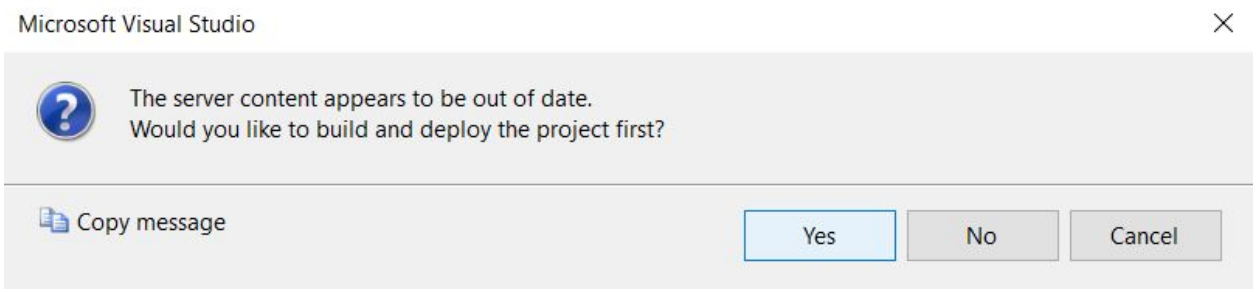

MultidimensionalProject10 - Microsoft Visual Studio
Country Dimension.dim [Design]*
Season Dimension.dim [Design]*
Event Dimension.dim [Design]*
Sport Dimension.dim [Design]*
Attributes
Hierarchies
Data Source View
Country Dimension
Cn Id
Country
To create a new hierarchy, drag an attribute here.
Country_Dime...
cn_id
country
Solution Explorer
Cubes
Olympics.cube
Dimensions
City Dimension.dim
Year Dimension.dim
Greater Region Dimension.dim
NOC Dimension.dim
Sport Dimension.dim
Event Dimension.dim
Season Dimension.dim
Country Dimension.dim
Games Dimension.dim
HC Dimension.dim
Mining Structures
Roles
Assemblies
Miscellaneous
Getting Started (SSIS)
Samples
My First SSIS Solution
Control Flow Basics
Data Flow Basics

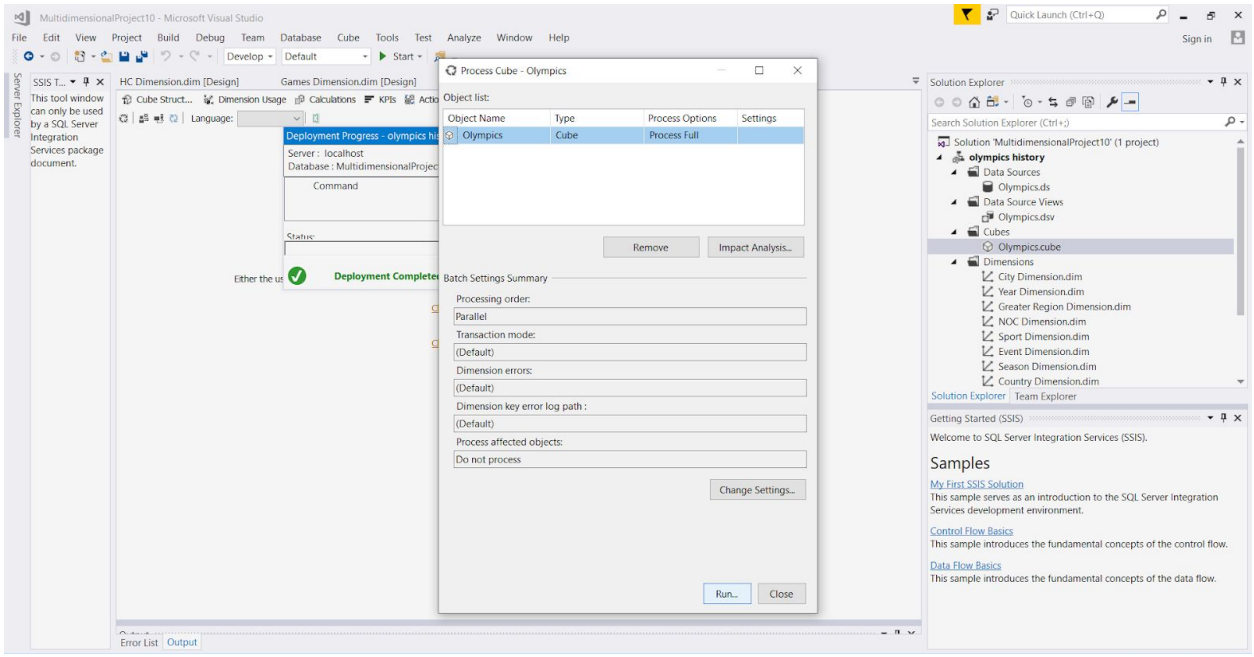

MultidimensionalProject10 - Microsoft Visual Studio
Event Dimension.dim [Design]*
Sport Dimension.dim [Design]*
NOC Dimension.dim [Design]*
Year Dimension.dim [Design]*
Dimension Struct...
Attribute Relationships
Translations
Browser
Attributes
Hierarchies
Data Source View
Event Dimension
Ev Id
Event
To create a new hierarchy, drag an attribute here.
Event_Dimension
ev_id
event
Solution Explorer
olympics history
Data Sources
Olympics.ds
Data Source Views
Olympics.dsv
Cubes
Olympics.cube
Dimensions
City Dimension.dim
Year Dimension.dim
Greater Region Dimension.dim
NOC Dimension.dim
Sport Dimension.dim
Event Dimension.dim
Season Dimension.dim
Country Dimension.dim
Getting Started (SSIS)
Welcome to SQL Server Integration Services (SSIS).
Samples
My First SSIS Solution
This sample serves as an introduction to the SQL Server Integration Services development environment.
Control Flow Basics
This sample introduces the fundamental concepts of the control flow.
Data Flow Basics
This sample introduces the fundamental concepts of the data flow.
Error List
Output

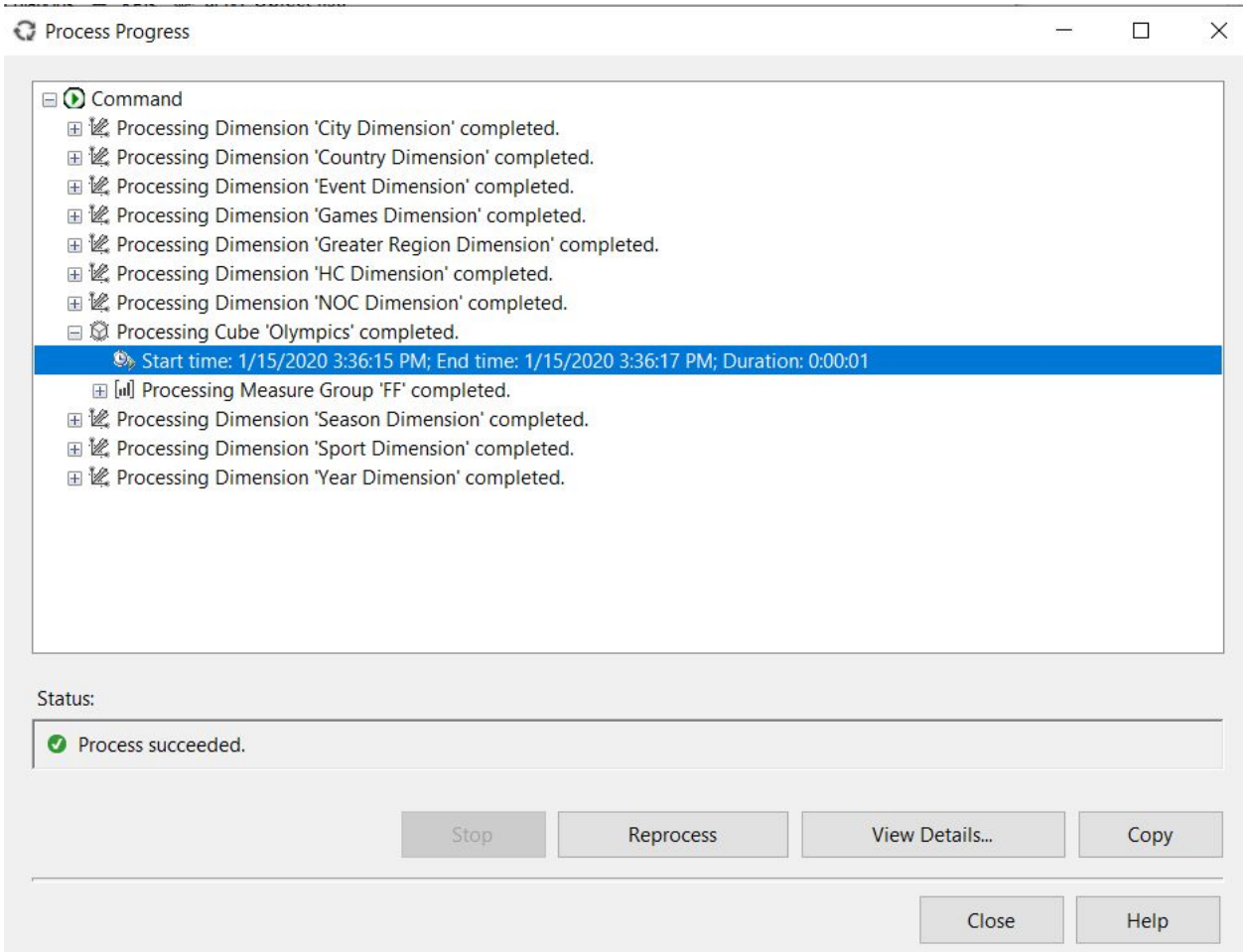

MultidimensionalProject10 - Microsoft Visual Studio
Year Dimension.dim [Design]*
City Dimension.dim [Design]*
Olympics.cube [Design]*
Greater Region Dimension.dim [Design]
Dimension Struct...
Attribute Relationships
Translations
Browser
Attributes
Hierarchies
Data Source View
Year Dimension
Year
Yr Id
To create a new hierarchy, drag an attribute here.
Year_Dimension
yr_id
year
Solution Explorer
olympics history
Data Sources
Olympics.ds
Data Source Views
Olympics.dsv
Cubes
Olympics.cube
Dimensions
City Dimension.dim
Year Dimension.dim
Greater Region Dimension.dim
NOC Dimension.dim
Sport Dimension.dim
Event Dimension.dim
Season Dimension.dim
Country Dimension.dim
Getting Started (SSIS)
Welcome to SQL Server Integration Services (SSIS).
Samples
My First SSIS Solution
This sample serves as an introduction to the SQL Server Integration Services development environment.
Control Flow Basics
This sample introduces the fundamental concepts of the control flow.
Data Flow Basics
This sample introduces the fundamental concepts of the data flow.
Error List
Output

After fixing the values, we deployed the cube.

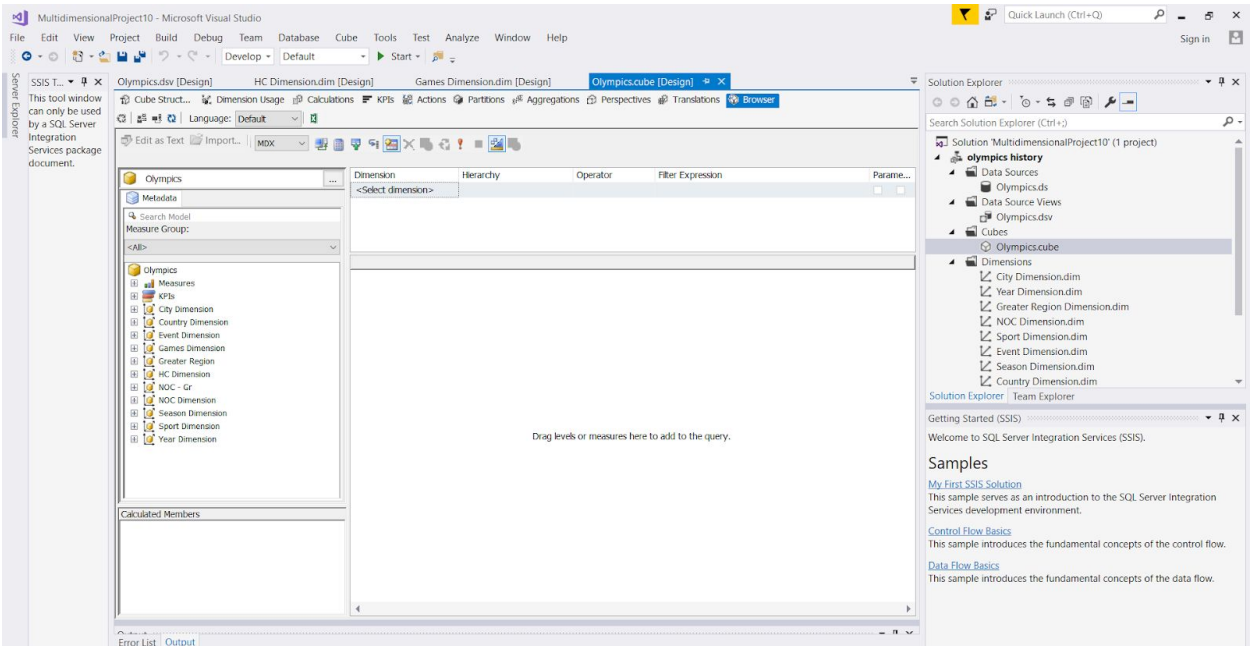


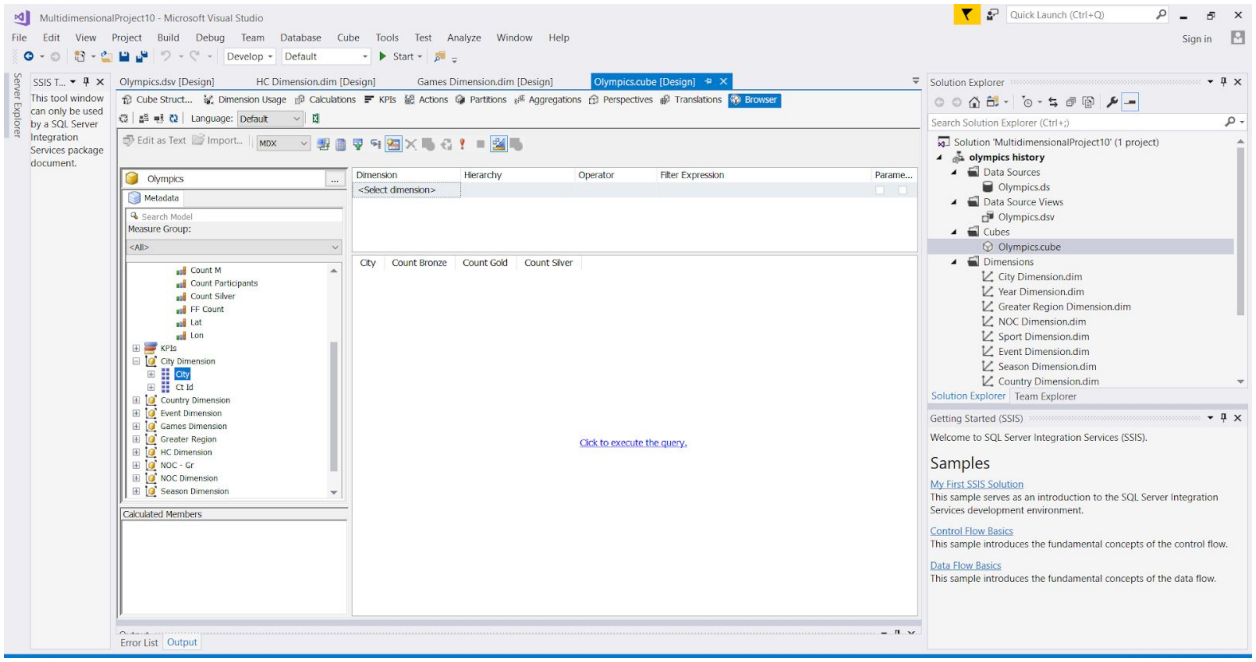



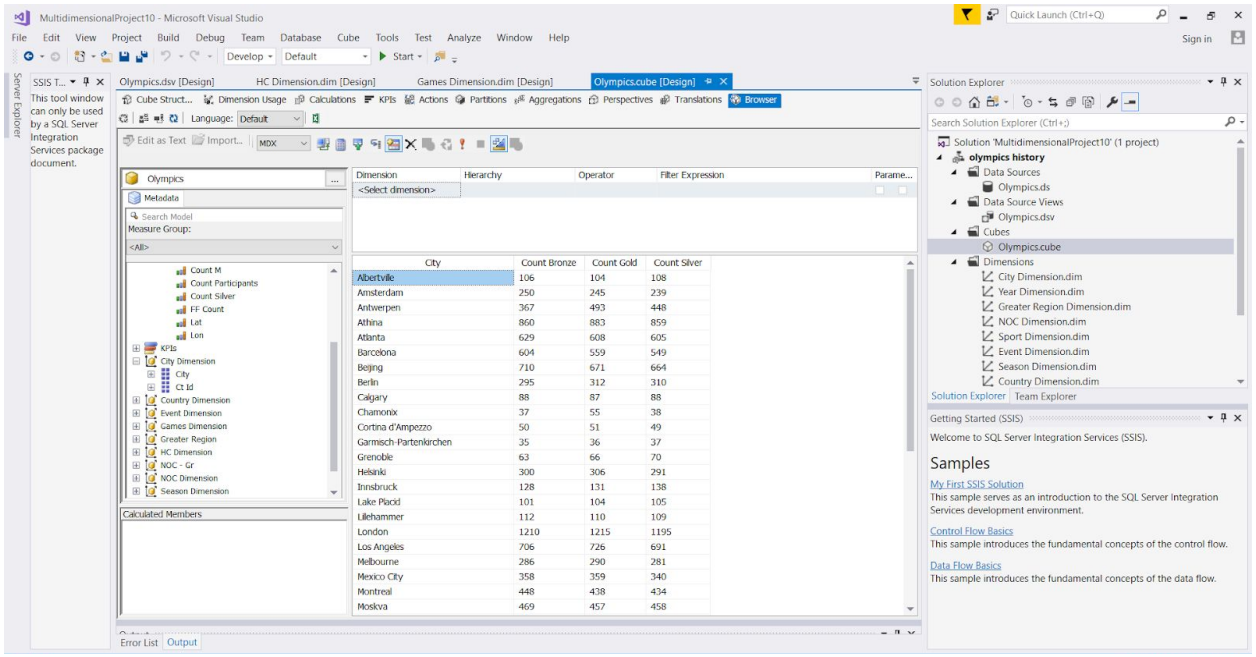

MultidimensionalProject10 - Microsoft Visual Studio
Process Cube - Olympics
Object list:
Object Name
Type
Process Options
Settings
Olympics
Cube
Process Full
Remove
Impact Analysis...
Batch Settings Summary
Processing order:
Parallel
Transaction mode:
(Default)
Dimension errors:
(Default)
Dimension key error log path :
(Default)
Process affected objects:
Do not process
Change Settings...
Run...
Close
Solution Explorer
Solution 'MultidimensionalProject10' (1 project)
olympics history
Data Sources
Olympics.ds
Data Source Views
Olympics.dsv
Cubes
Olympics.cube
Dimensions
City Dimension.dim
Year Dimension.dim
Greater Region Dimension.dim
NOC Dimension.dim
Sport Dimension.dim
Event Dimension.dim
Season Dimension.dim
Country Dimension.dim
Getting Started (SSIS)
Welcome to SQL Server Integration Services (SSIS).
Samples
My First SSIS Solution
This sample serves as an introduction to the SQL Server Integration Services development environment.
Control Flow Basics
This sample introduces the fundamental concepts of the control flow.
Data Flow Basics
This sample introduces the fundamental concepts of the data flow.


Process Progress
Command
Processing Dimension 'City Dimension' completed.
Processing Dimension 'Country Dimension' completed.
Processing Dimension 'Event Dimension' completed.
Processing Dimension 'Games Dimension' completed.
Processing Dimension 'Greater Region Dimension' completed.
Processing Dimension 'HC Dimension' completed.
Processing Dimension 'NOC Dimension' completed.
Processing Cube 'Olympics' completed.
Start time: 1/15/2020 3:36:15 PM; End time: 1/15/2020 3:36:17 PM; Duration: 0:00:01
Processing Measure Group 'FF' completed.
Processing Dimension 'Season Dimension' completed.
Processing Dimension 'Sport Dimension' completed.
Processing Dimension 'Year Dimension' completed.
Status:
Process succeeded.
Stop
Reprocess
View Details...
Copy
Close
Help

**Step 5 :** In order to check if our cube is working we counted the number of gold, silver and bronze medals per country.



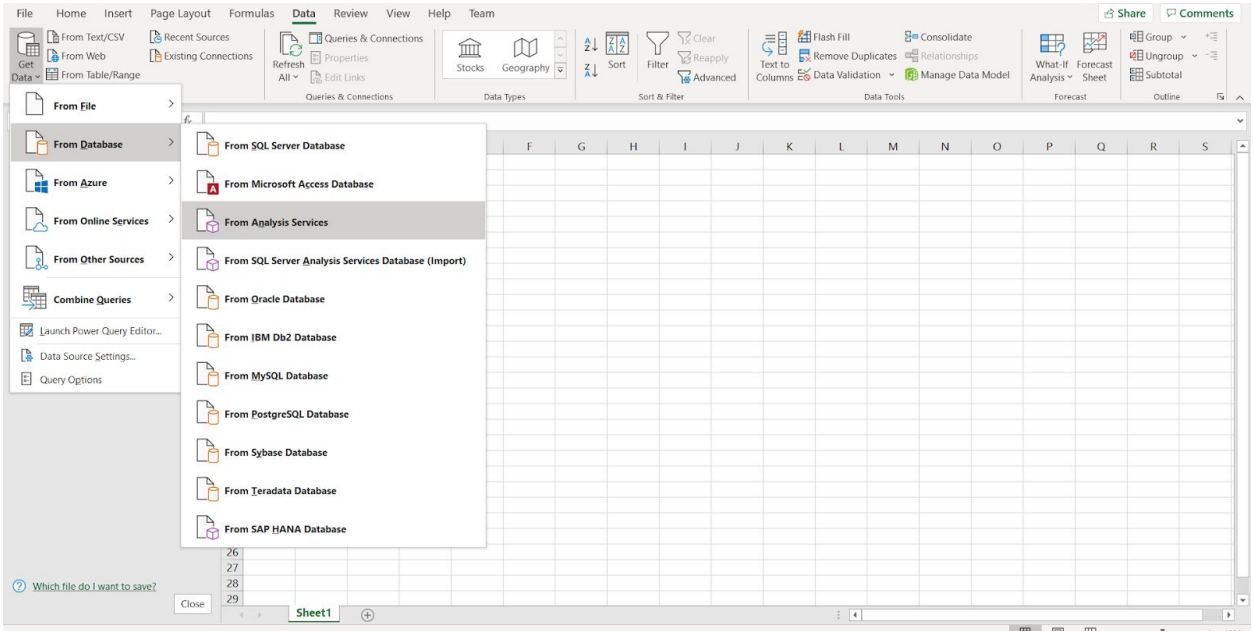

MultidimensionalProject10 - Microsoft Visual Studio
Olympics.dsv [Design]
HC Dimension.dim [Design]
Games Dimension.dim [Design]
Olympics.cube [Design]
Cube Struct...
Dimension Usage
Calculations
KPIs
Actions
Partitions
Aggregations
Perspectives
Translations
Browser
Solution Explorer
olympics history
Data Sources
Olympics.ds
Data Source Views
Olympics.dsv
Cubes
Olympics.cube
Dimensions
City Dimension.dim
Year Dimension.dim
Greater Region Dimension.dim
NOC Dimension.dim
Sport Dimension.dim
Event Dimension.dim
Season Dimension.dim
Country Dimension.dim
Click to execute the query.
Getting Started (SSIS)
Samples
My First SSIS Solution
Control Flow Basics
Data Flow Basics

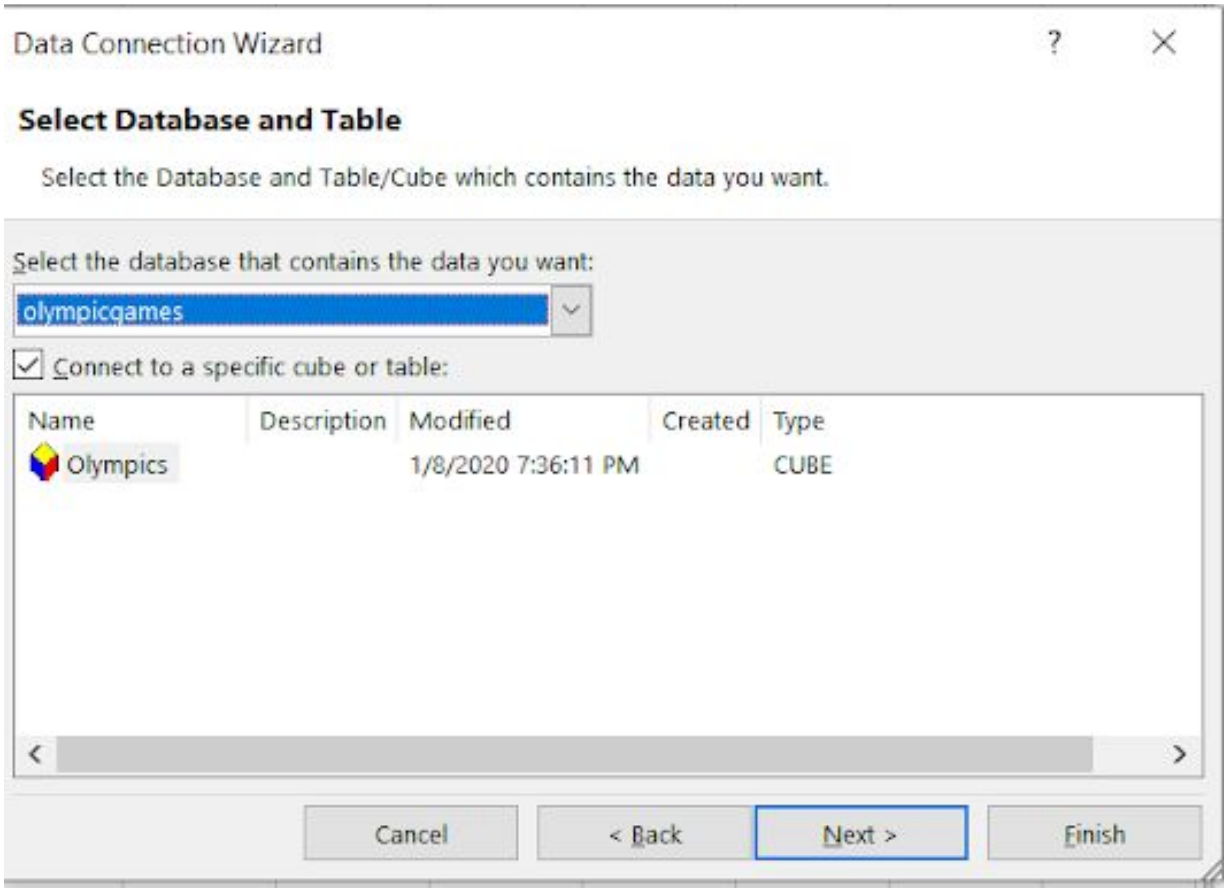

MultidimensionalProject10 - Microsoft Visual Studio
Olympics.dsv [Design]
HC Dimension.dim [Design]
Games Dimension.dim [Design]
Olympics.cube [Design]
Solution Explorer
olympics history
Data Sources
Olympics.ds
Data Source Views
Olympics.dsv
Cubes
Olympics.cube
Dimensions
City Dimension.dim
Year Dimension.dim
Greater Region Dimension.dim
NOC Dimension.dim
Sport Dimension.dim
Event Dimension.dim
Season Dimension.dim
Country Dimension.dim
City Count Bronze Count Gold Count Silver
Albertville 106 104 108
Amsterdam 250 245 239
Antwerpen 367 493 448
Athina 860 883 859
Atlanta 629 608 605
Barcelona 604 559 549
Beijing 710 671 664
Berlin 295 312 310
Calgary 88 87 88
Chamonix 37 55 38
Cortina d'Ampezzo 50 51 49
Garmisch-Partenkirchen 35 36 37
Grenoble 63 66 70
Helsinki 300 306 291
Innsbruck 128 131 138
Lake Placid 101 104 105
Lillehammer 112 110 109
London 1210 1215 1195
Los Angeles 706 726 691
Melbourne 286 290 281
Mexico City 358 359 340
Montreal 448 438 434
Moskva 469 457 458
Getting Started (SSIS)
Samples
My First SSIS Solution
Control Flow Basics
Data Flow Basics

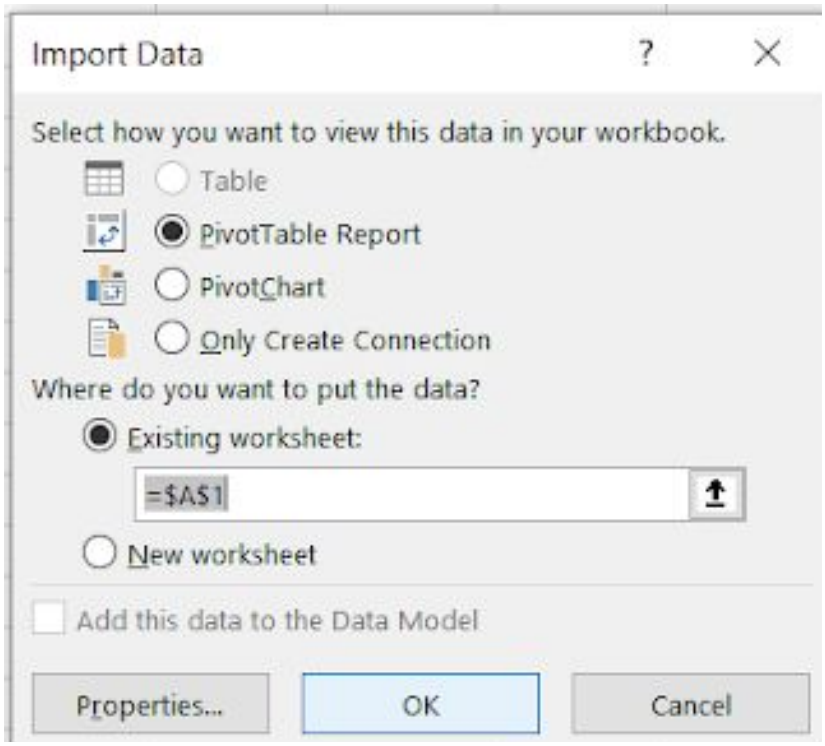
# 6. Data Visualisation

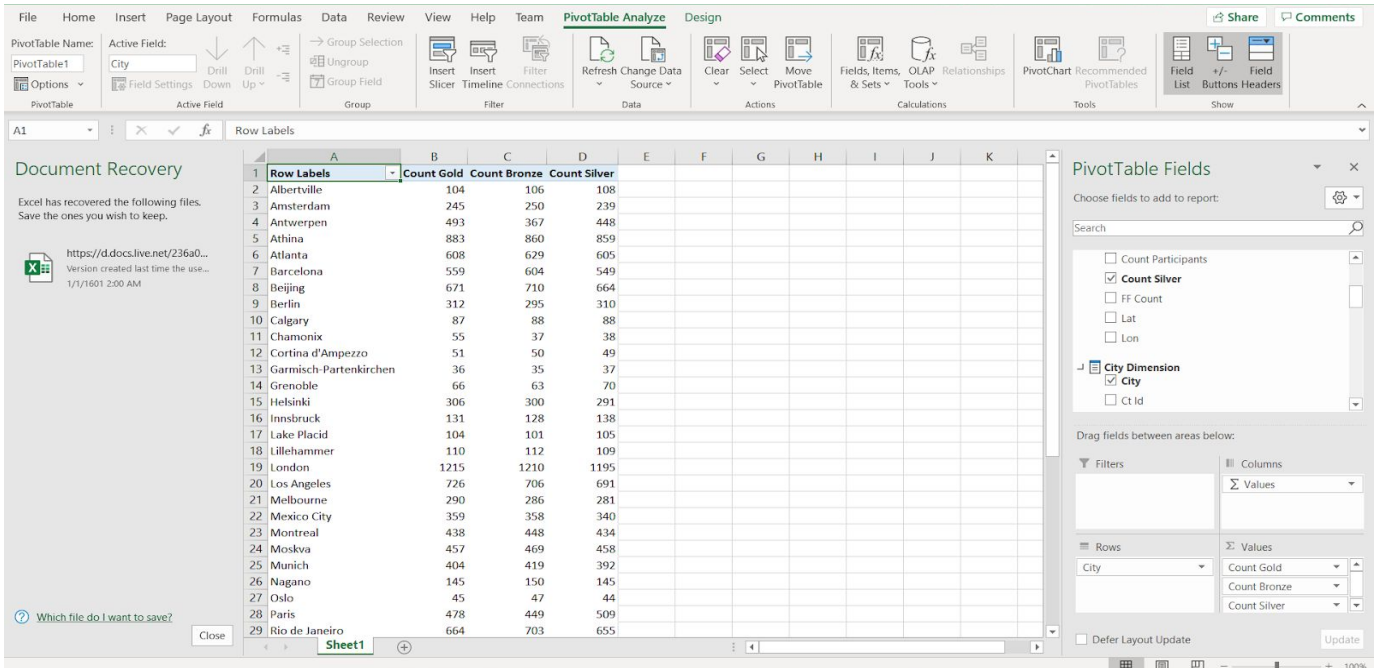
## 6.1. Import Cube in Excel

In order to connect the cube with Microsoft Excel, we followed the steps below:



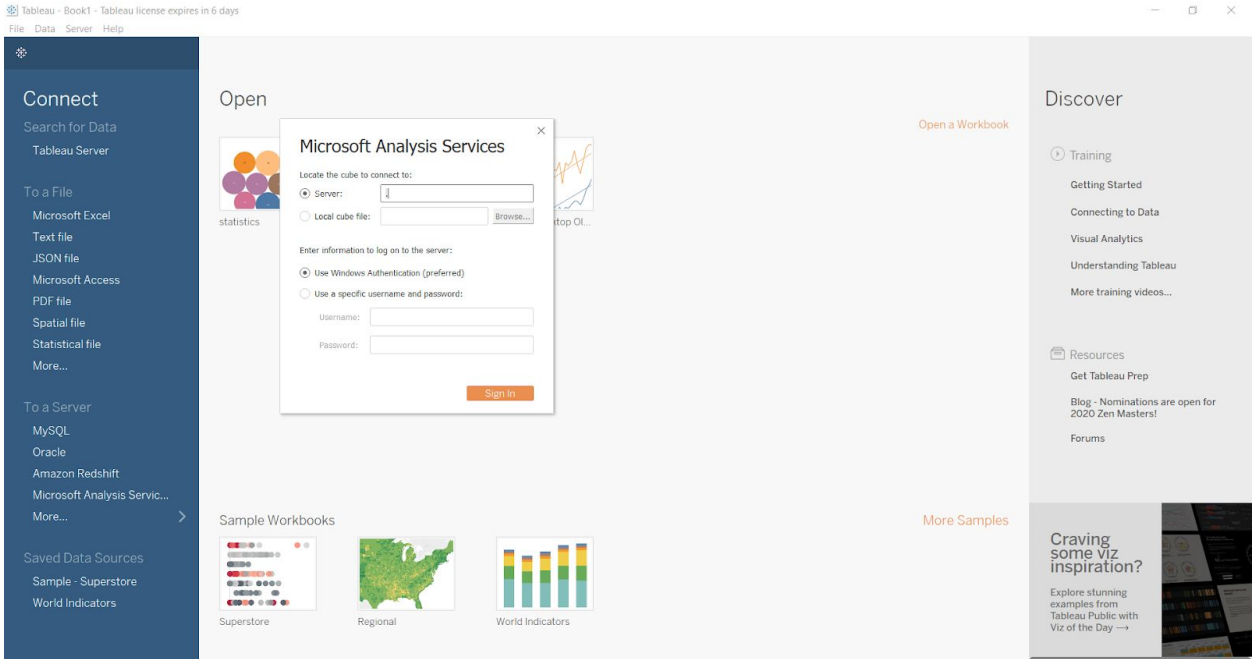

Data Connection Wizard
Select Database and Table
Select the Database and Table/Cube which contains the data you want.
Select the database that contains the data you want:
olympicgames
Connect to a specific cube or table:
Name
Description
Modified
Created
Type
Olympics
1/8/2020 7:36:11 PM
CUBE
Cancel
< Back
Next >
Finish


Import Data
Select how you want to view this data in your workbook.
Table
PivotTable Report
PivotChart
Only Create Connection
Where do you want to put the data?
Existing worksheet:
=$A$1
New worksheet
Add this data to the Data Model
Properties...
OK
Cancel


Document Recovery
Excel has recovered the following files. Save the ones you wish to keep.
Which file do I want to save?
Close
Sheet1
PivotTable Fields
Choose fields to add to report:
Drag fields between areas below:

## 6.2. Import Cube in Tableau

In order to connect the cube with Tableau, we followed the steps below:

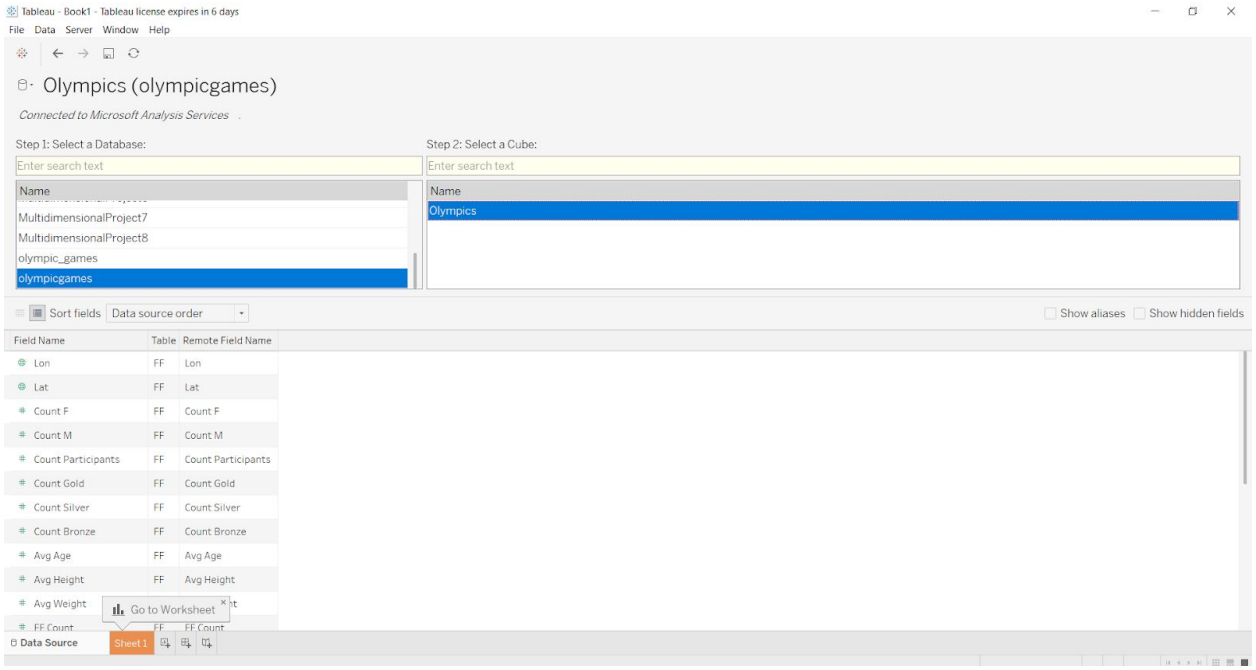


Tableau - Book1 - Tableau license expires in 6 days
File Data Server Window Help
Olympics (olympicgames)
Connected to Microsoft Analysis Services
Step 1: Select a Database:
Enter search text
Name
MultidimensionalProject7
MultidimensionalProject8
olympic_games
olympicgames
Step 2: Select a Cube:
Enter search text
Name
Olympics
Sort fields
Data source order
Show aliases
Show hidden fields
Field Name
Table
Remote Field Name
Lon
Lat
Count F
Count M
Count Participants
Count Gold
Count Silver
Count Bronze
Avg Age
Avg Height
Avg Weight
FF Count
Go to Worksheet
Data Source
Sheet 1

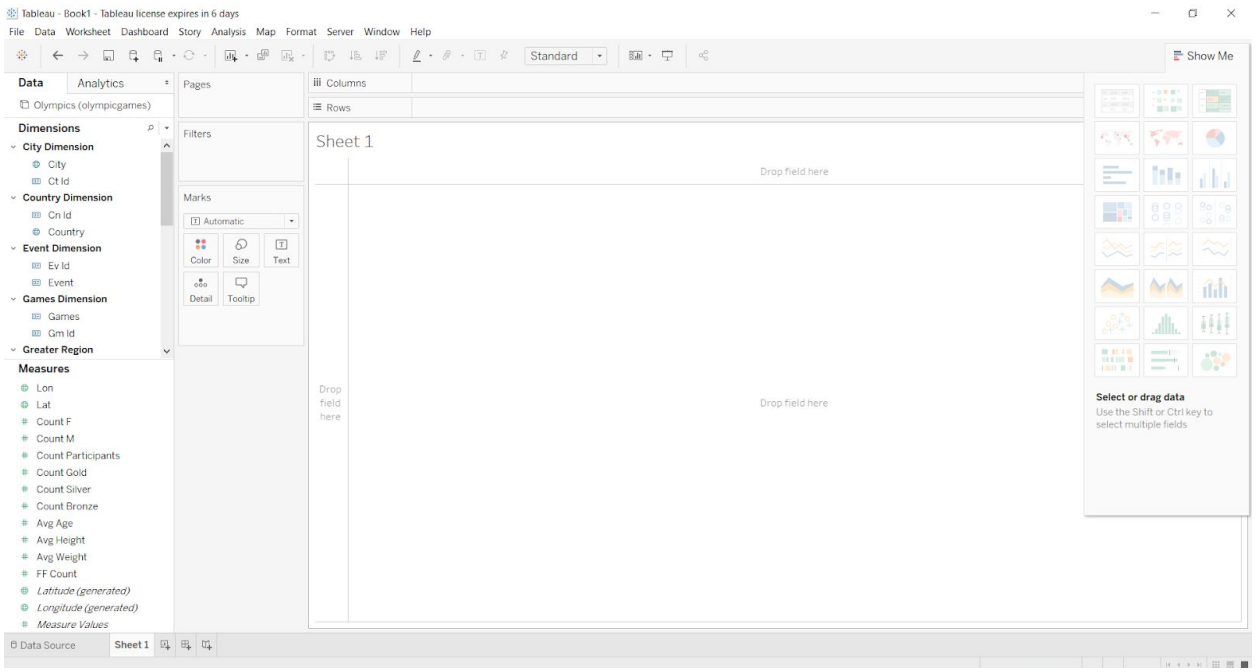
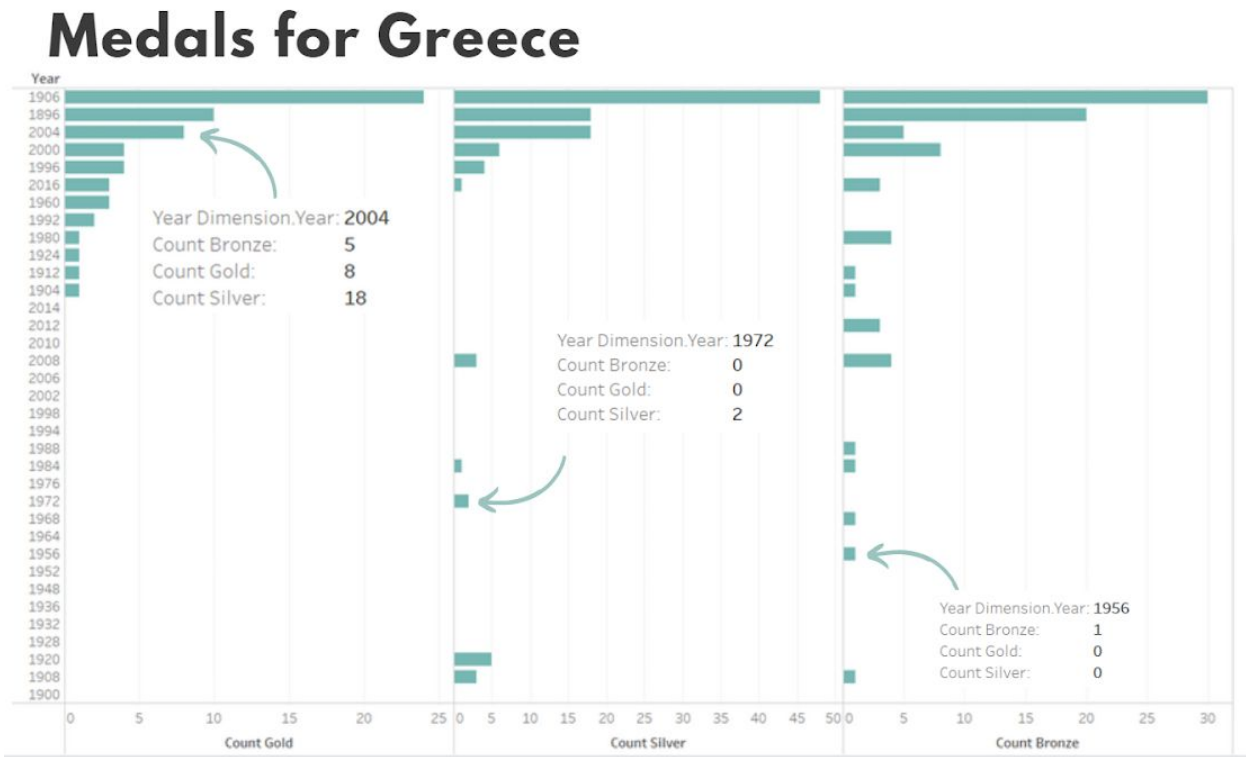
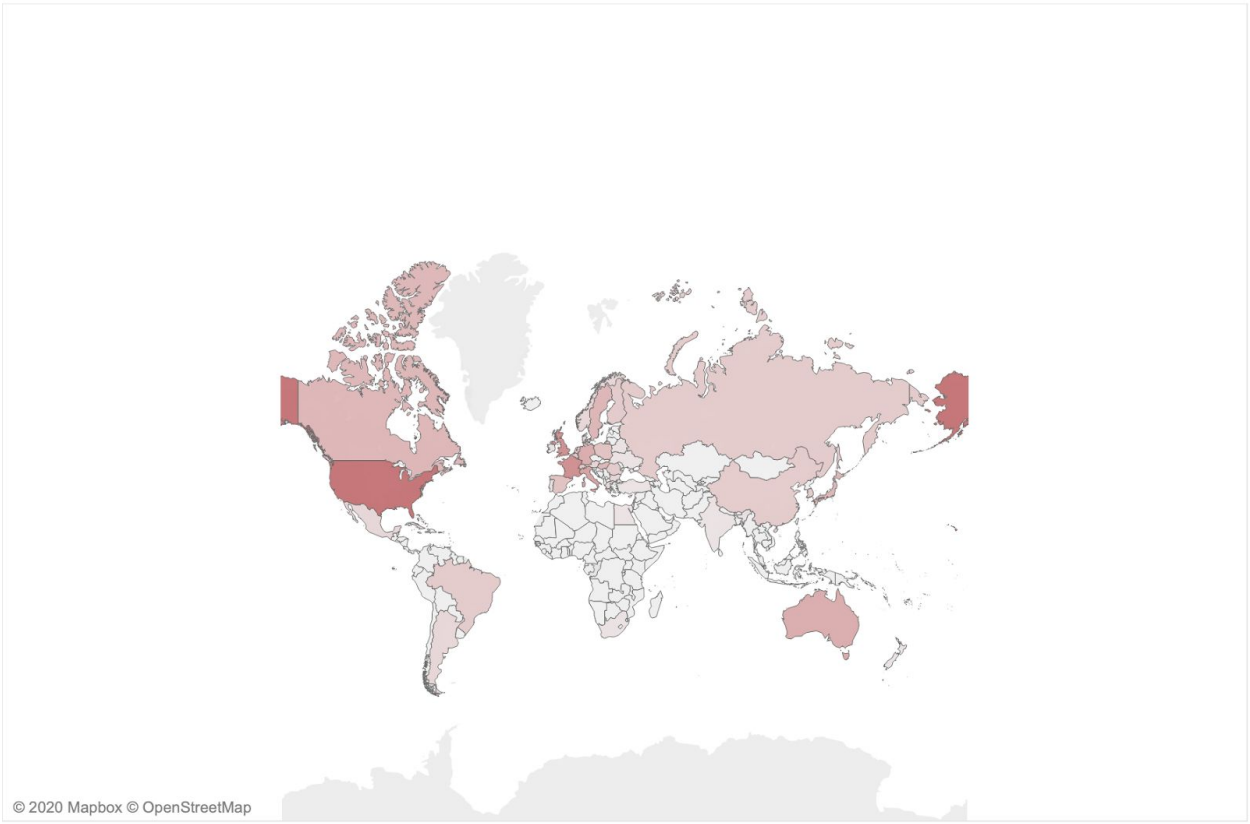


Tableau - Book1 - Tableau license expires in 6 days
File Data Worksheet Dashboard Story Analysis Map Format Server Window Help
Standard
Show Me
Data
Analytics
Olympics (olympicgames)
Dimensions
City Dimension
City
Ct Id
Country Dimension
Cn Id
Country
Event Dimension
Ev Id
Event
Games Dimension
Games
Gm Id
Greater Region
Measures
Lon
Lat
Count F
Count M
Count Participants
Count Gold
Count Silver
Count Bronze
Avg Age
Avg Height
Avg Weight
FF Count
Latitude (generated)
Longitude (generated)
Measure Values
Pages
Filters
Marks
Automatic
Color
Size
Text
Detail
Tooltip
Columns
Rows
Sheet 1
Drop field here
Drop field here
Drop field here
Select or drag data
Use the Shift or Ctrl key to select multiple fields
Data Source
Sheet 1

## 6.3. Data Visualisation



## Summary: Participants by Country (Summer)

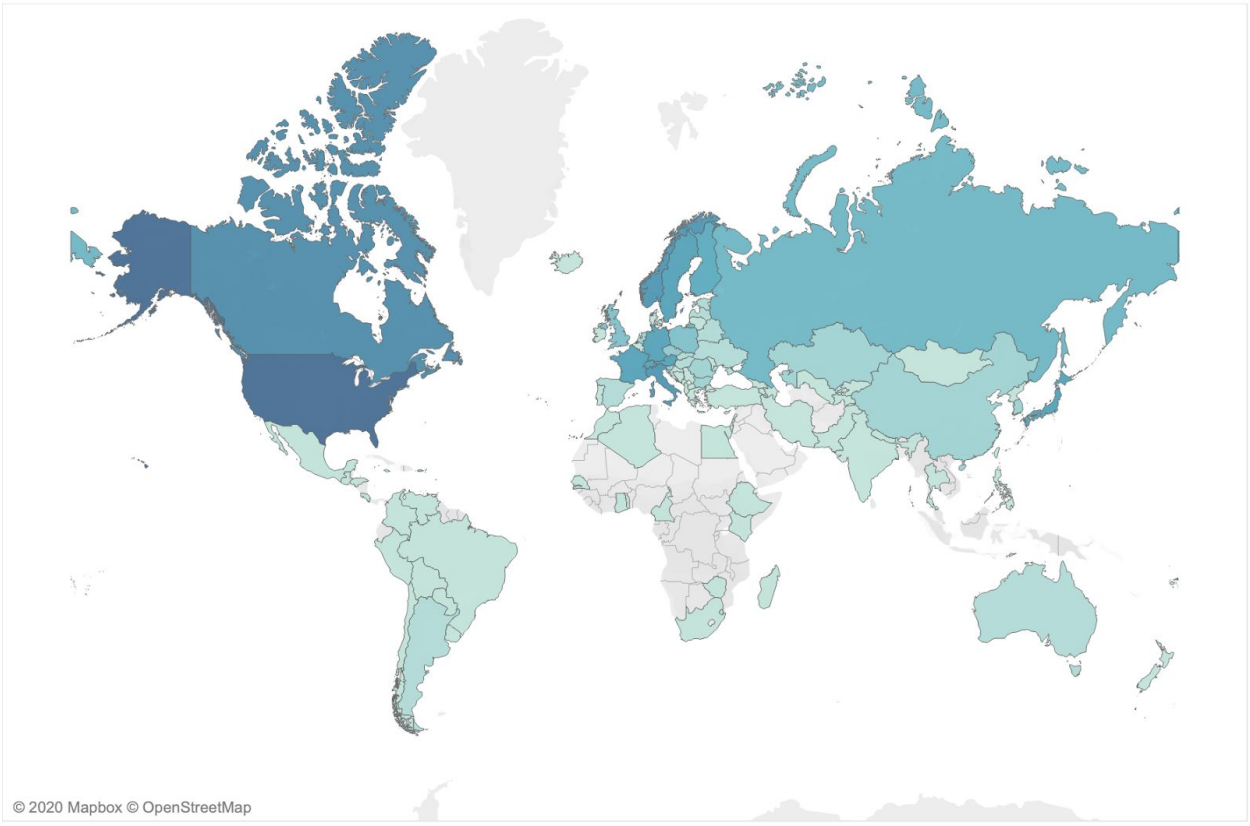



Map based on Longitude (generated) and Latitude (generated). Color shows sum of Count Participants. Details are shown for Country. The data is filtered on Season, which keeps Summer.

Count Participants

1 15.064

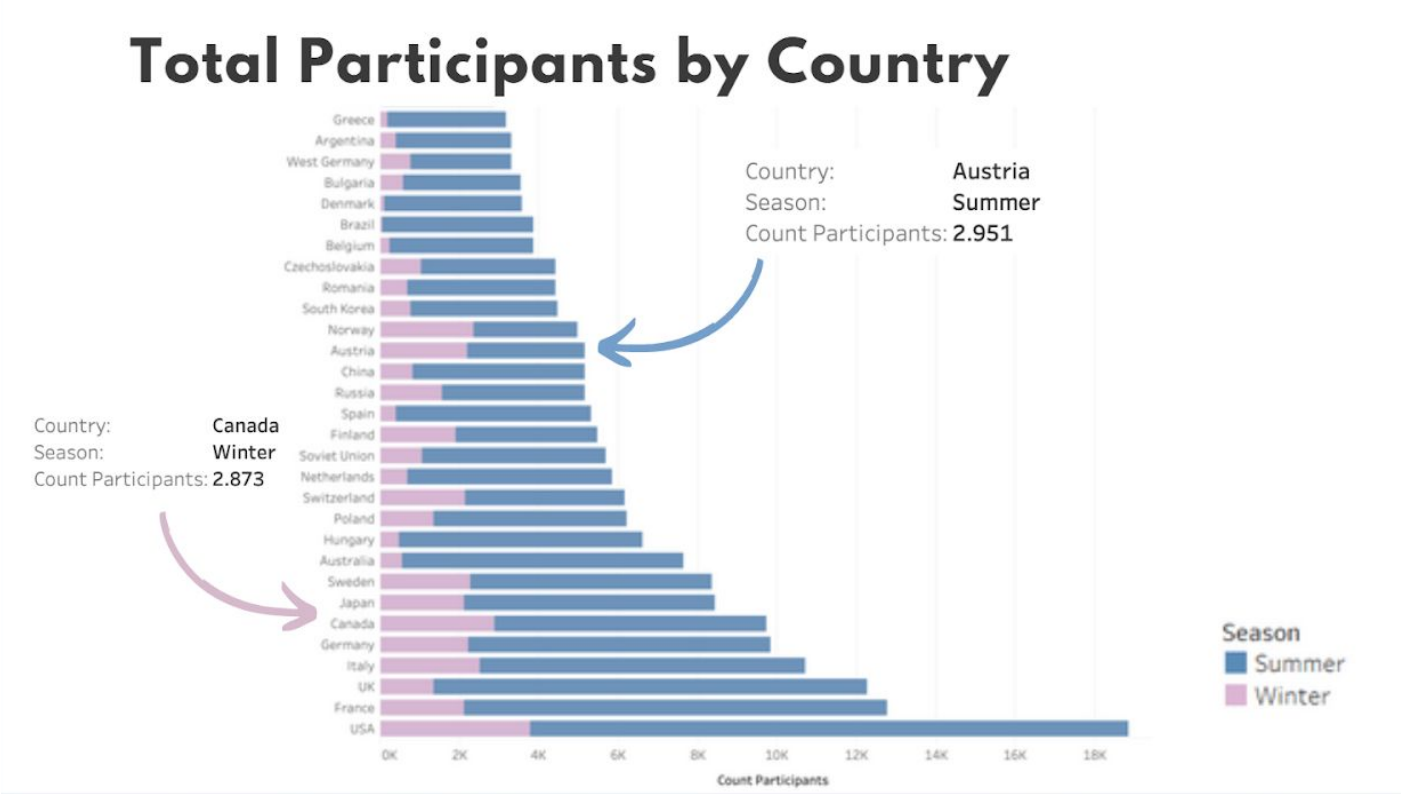
Summary: Participants by Country (Winter)

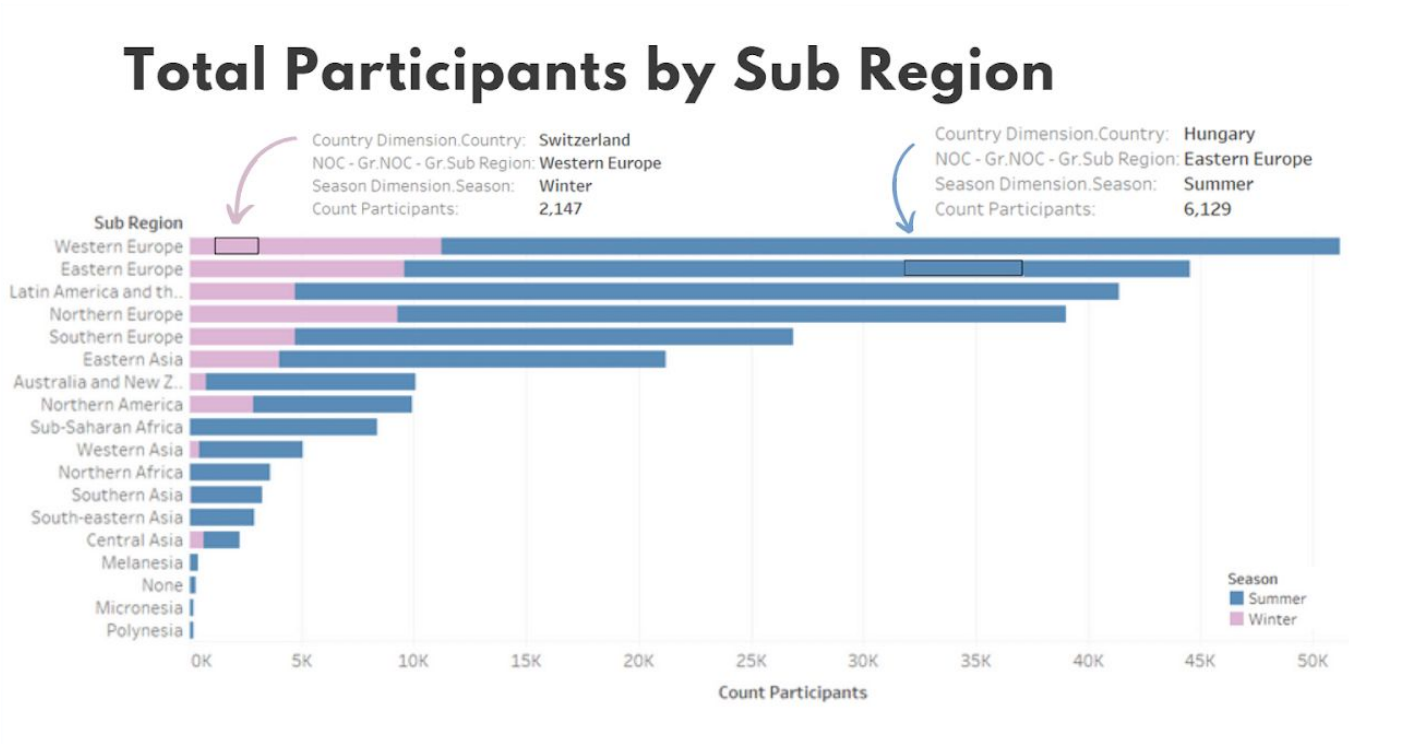



Map based on Longitude (generated) and Latitude (generated). Color shows sum of Count Participants. Details are shown for Country. The data is filtered on Season, which keeps Winter.

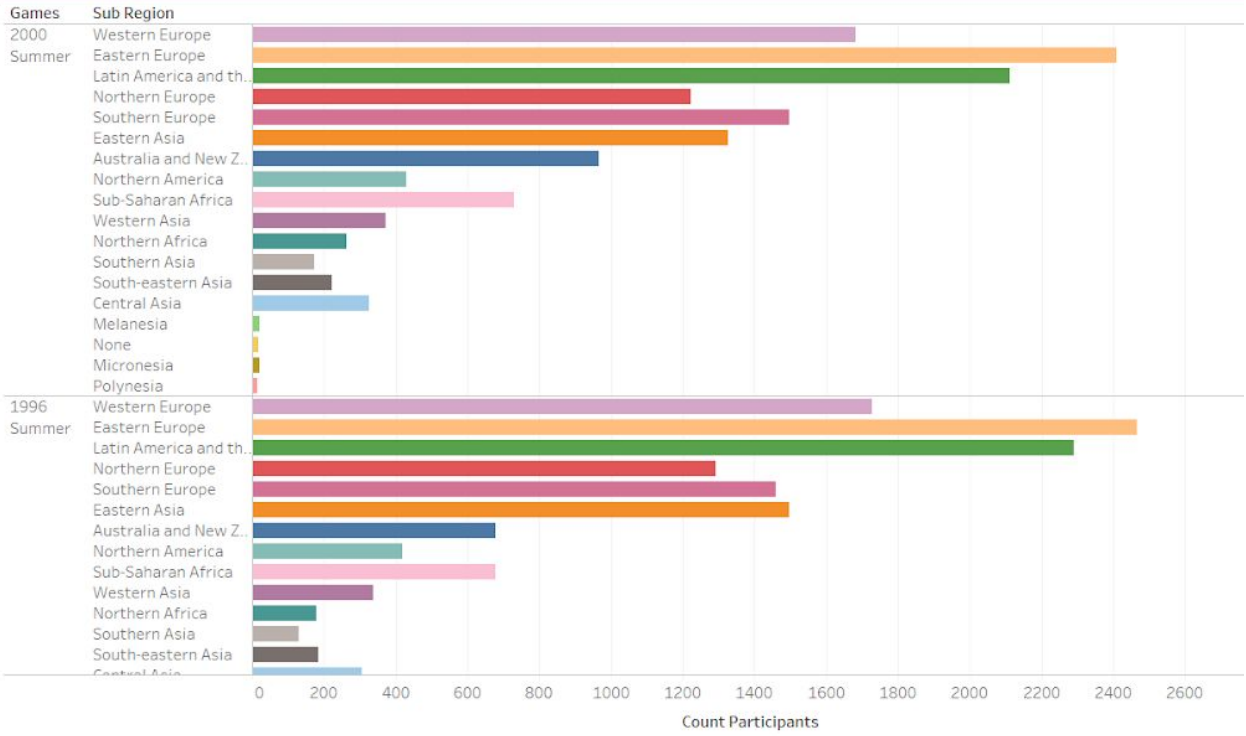
Count Participants

1 3.789

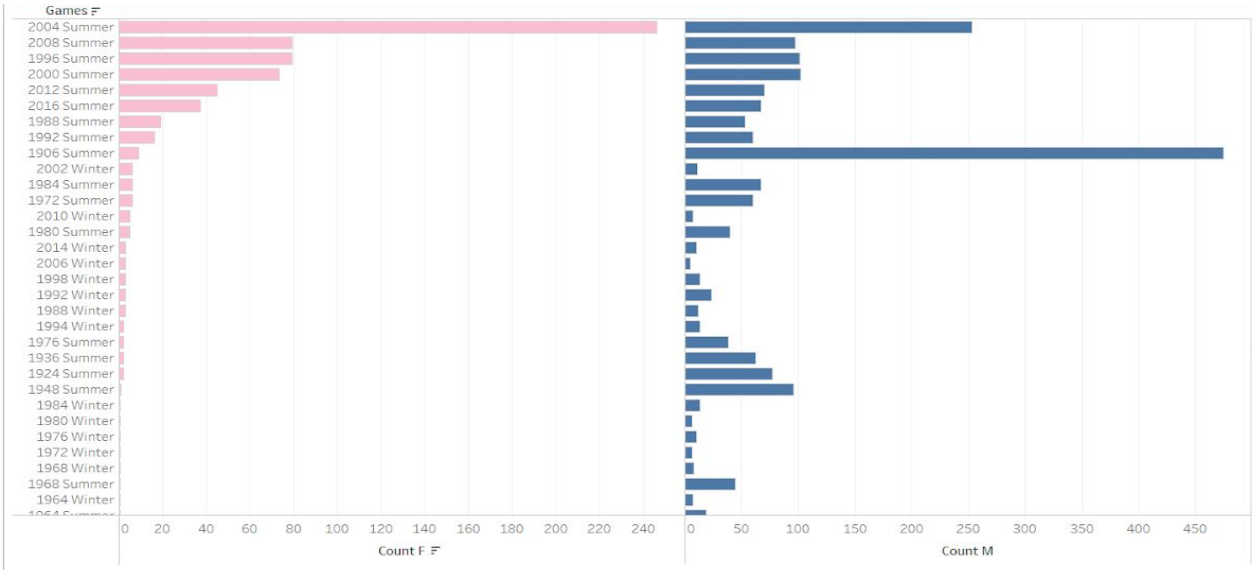

Total Participants by Country
Country: Austria
Season: Summer
Count Participants: 2.951
Country: Canada
Season: Winter
Count Participants: 2.873
Greece
Argentina
West Germany
Bulgaria
Denmark
Brazil
Belgium
Czechoslovakia
Romania
South Korea
Norway
Austria
China
Russia
Spain
Finland
Soviet Union
Netherlands
Switzerland
Poland
Hungary
Australia
Sweden
Japan
Canada
Germany
Italy
UK
France
USA
0K 2K 4K 6K 8K 10K 12K 14K 16K 18K
Count Participants
Season
Summer
Winter


Total Participants by Sub Region
Country Dimension.Country: Switzerland
NOC - Gr.NOC - Gr.Sub Region: Western Europe
Season Dimension.Season: Winter
Count Participants: 2,147
Country Dimension.Country: Hungary
NOC - Gr.NOC - Gr.Sub Region: Eastern Europe
Season Dimension.Season: Summer
Count Participants: 6,129
Sub Region
Western Europe
Eastern Europe
Latin America and th..
Northern Europe
Southern Europe
Eastern Asia
Australia and New Z..
Northern America
Sub-Saharan Africa
Western Asia
Northern Africa
Southern Asia
South-eastern Asia
Central Asia
Melanesia
None
Micronesia
Polynesia
0K 5K 10K 15K 20K 25K 30K 35K 40K 45K 50K
Count Participants
Season
Summer
Winter

## Total Participants per Game and Sub Region



## Females and Males per Game



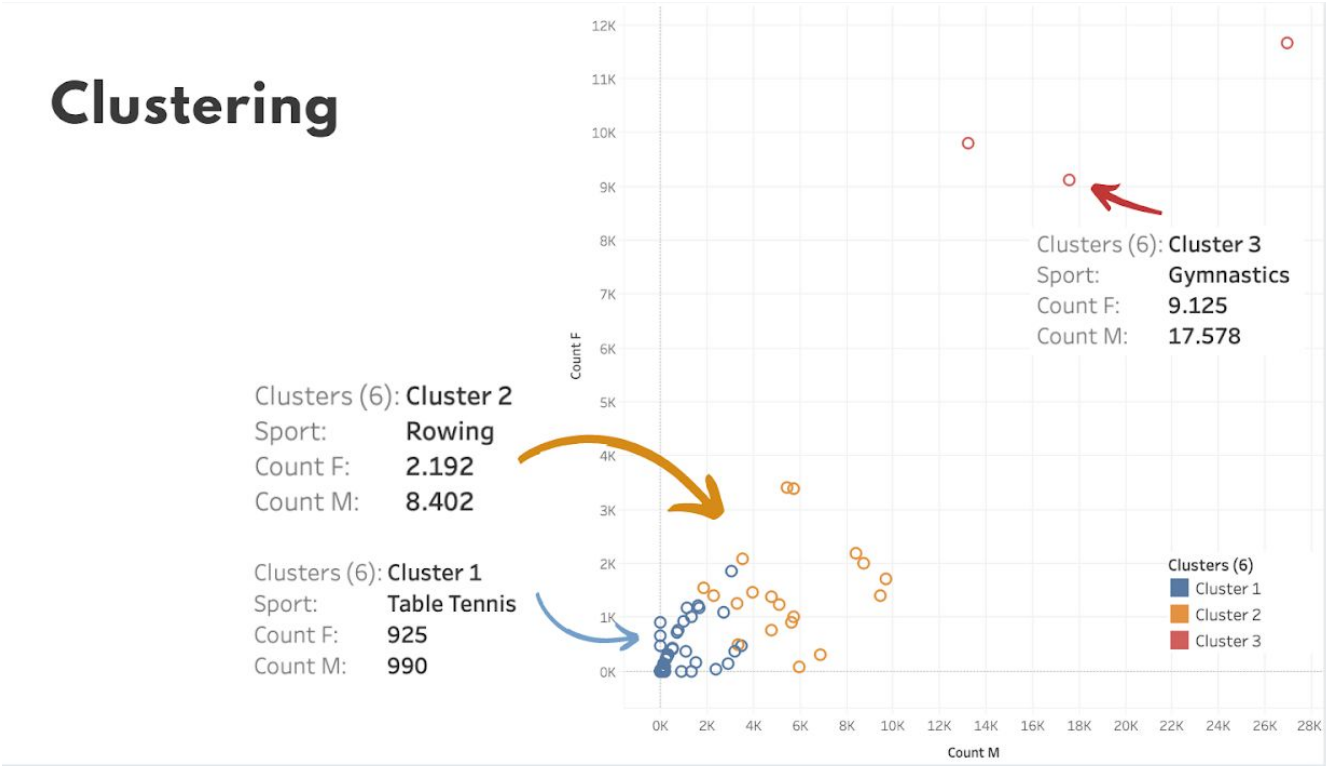
# 7. Data Mining

## 7.1. Clustering

Data clustering was done using Tableau, through the Data Analytics Tab. The number of clusters was automatically adjusted by Tableau's algorithm. The clustering results can be found below.

What does the clustering show? It shows by sport, the number of total male and female competitors through the years considering gold medals as well. Thus, it has grouped the competitors in 3 main groups: well known sports (like swimming and gymnastics), mid-known sports and sports with few competitors. Through this clustering, we can also understand what the analogy of male and female competitors is, given that through the clustering results, in sports with fewer competitors there are higher chances for equal participation.

# Clustering




# Clustering

## Inputs for Clustering

| | |
|---|---|
| **Variables:** | Sum of Count F |
| | Sum of Count Gold |
| | Sum of Count M |
| **Level of Detail:** | Sport |
| **Scaling:** | Normalized |

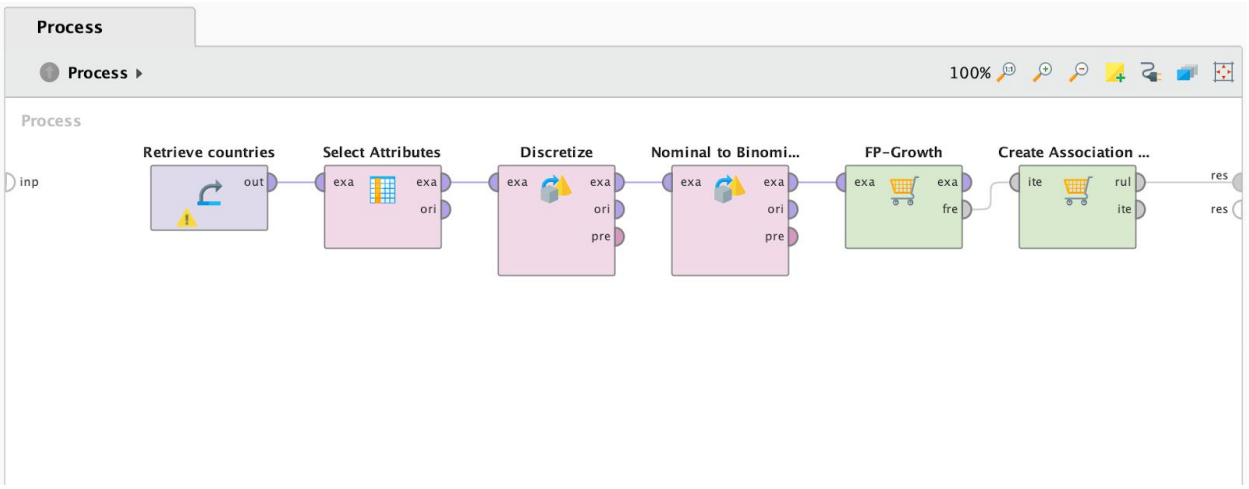
## Summary Diagnostics

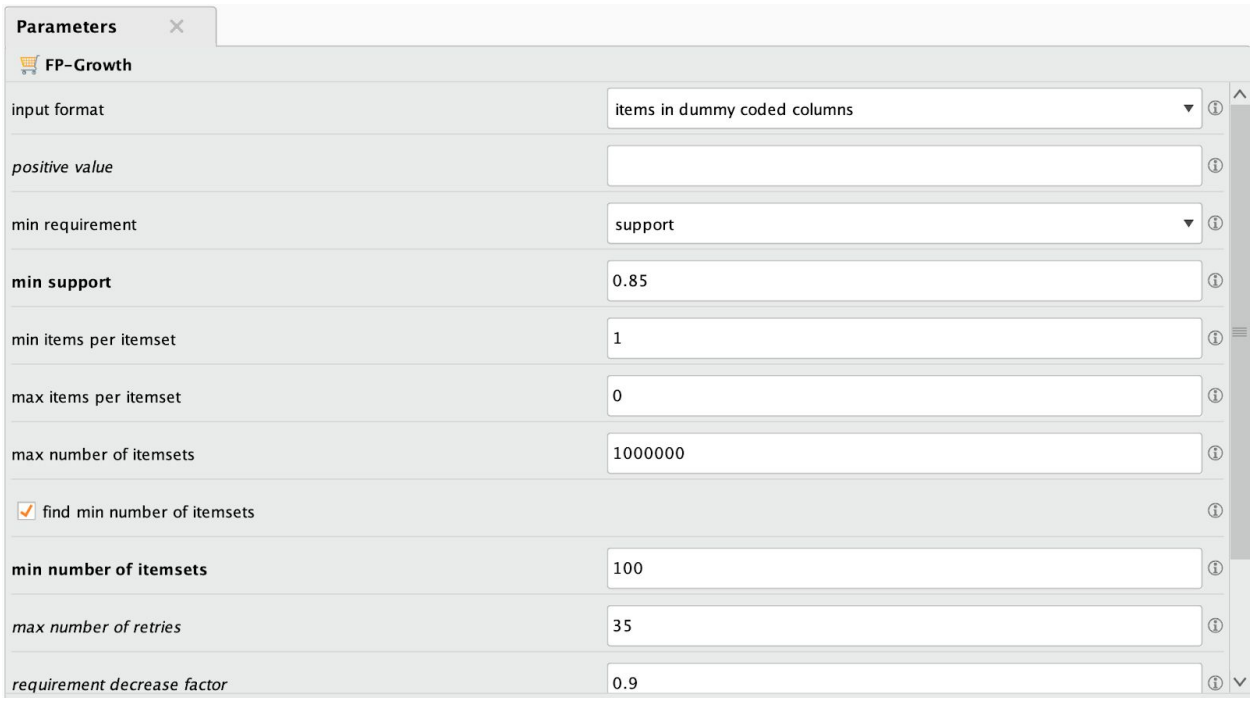
| | |
|---|---|
| **Number of Clusters:** | 3 |
| **Number of Points:** | 66 |
| **Between-group Sum of Squares:** | 5.8459 |
| **Within-group Sum of Squares:** | 0.99337 |
| **Total Sum of Squares:** | 6.8393 |

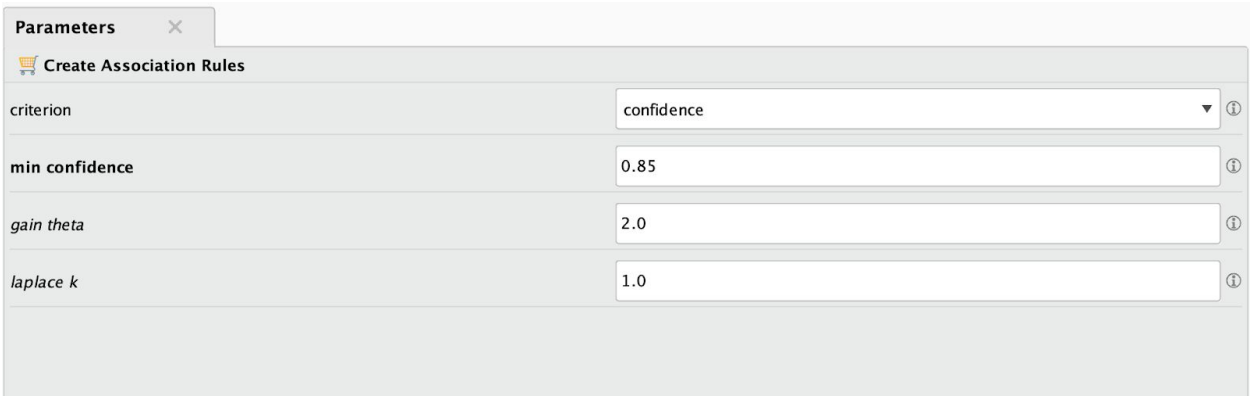
| | | Centers | | |
|---|---|---|---|---|
| **Clusters** | **Number of Items** | **Sum of Count F** | **Sum of Count Gold** | **Sum of Count M** |
| **Cluster 1** | 44 | 359.55 | 54.341 | 773.93 |
| **Cluster 2** | 19 | 1473.3 | 408.0 | 5502.6 |
| **Cluster 3** | 3 | 10193.0 | 1076.0 | 19257.0 |
| **Not Clustered** | 0 | | | |

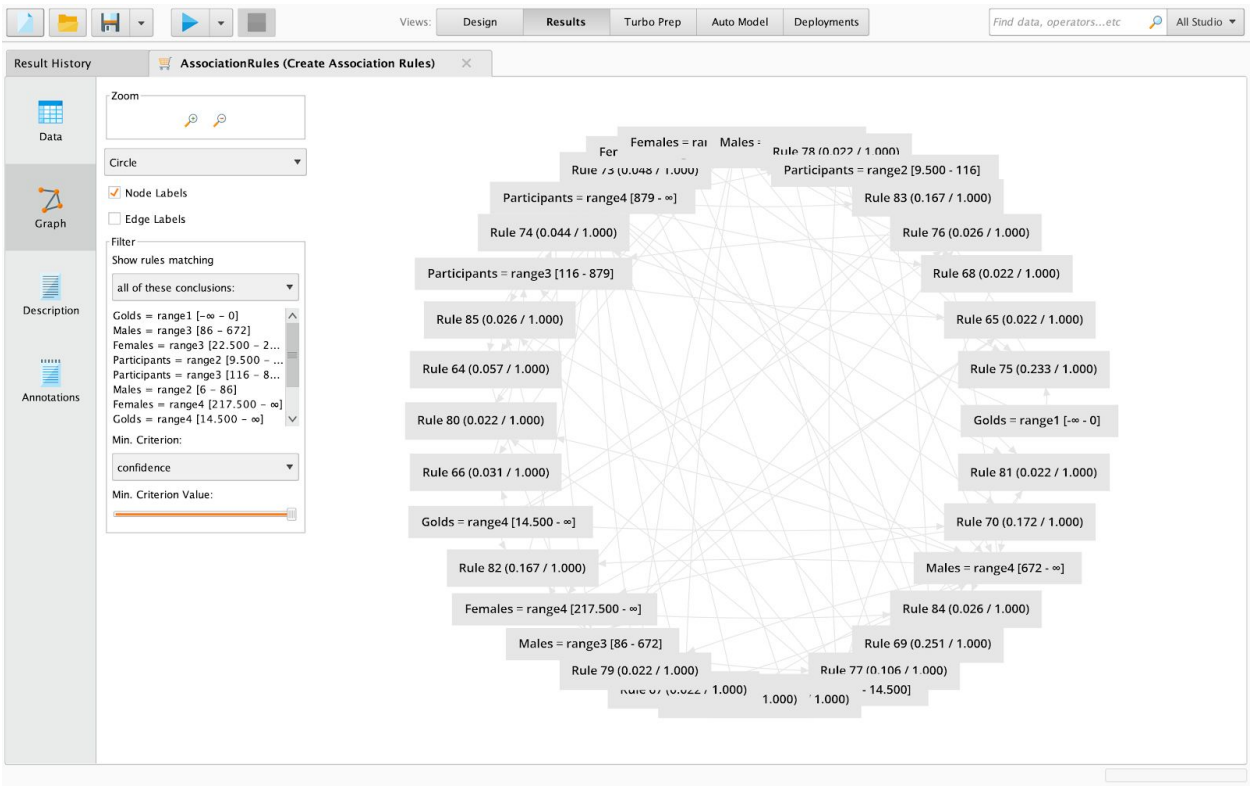
## 7.2. Association Rules

For the creation of association rules, we used RapidMiner Studio. The process followed can be found in screenshots below. Our goal was to understand and predict the number of male and female participants and its connection to the gold medals that a country has.





Parameters
FP-Growth
input format
items in dummy coded columns
positive value
min requirement
support
min support
0.85
min items per itemset
1
max items per itemset
0
max number of itemsets
1000000
find min number of itemsets
min number of itemsets
100
max number of retries
35
requirement decrease factor
0.9




The result are the following:

Association Rules

[Females = range3 [22.500 - 217.500], Golds = range4 [14.500 - ∞]] --> [Males = range3 [86 - 672]] (confidence: 0.857)

[Females = range3 [22.500 - 217.500], Golds = range4 [14.500 - ∞]] --> [Males = range3 [86 - 672], Participants = range3 [116 - 879]] (confidence: 0.857)

[Females = range3 [22.500 - 217.500], Participants = range3 [116 - 879], Golds = range4 [14.500 - ∞]] --> [Males = range3 [86 - 672]] (confidence: 0.857)

[Males = range3 [86 - 672], Participants = range3 [116 - 879], Golds = range3 [0 - 14.500]] --> [Females = range3 [22.500 - 217.500]] (confidence: 0.857)

[Participants = range3 [116 - 879], Golds = range3 [0 - 14.500]] --> [Females = range3 [22.500 - 217.500]] (confidence: 0.862)

[Females = range4 [217.500 - ∞], Males = range4 [672 - ∞]] --> [Golds = range4 [14.500 - ∞]] (confidence: 0.864)

[Females = range4 [217.500 - ∞], Males = range4 [672 - ∞]] --> [Golds = range4 [14.500 - ∞], Participants = range4 [879 - ∞]] (confidence: 0.864)

[Females = range4 [217.500 - ∞], Males = range4 [672 - ∞], Participants = range4 [879 - ∞]] --> [Golds = range4 [14.500 - ∞]] (confidence: 0.864)

[Males = range3 [86 - 672], Females = range2 [2 - 22.500]] --> [Participants = range3 [116 - 879]] (confidence: 0.867)

[Females = range4 [217.500 - ∞], Participants = range4 [879 - ∞]] --> [Golds = range4 [14.500 - ∞]] (confidence: 0.867)

[Males = range2 [6 - 86]] --> [Golds = range1 [-∞ - 0], Participants = range2 [9.500 - 116]] (confidence: 0.875)

[Participants = range2 [9.500 - 116], Golds = range3 [0 - 14.500]] --> [Males = range2 [6 - 86]] (confidence: 0.875)

[Males = range2 [6 - 86], Golds = range3 [0 - 14.500]] --> [Participants = range2 [9.500 - 116]] (confidence: 0.875)

[Females = range4 [217.500 - ∞]] --> [Males = range4 [672 - ∞]] (confidence: 0.880)

[Males = range4 [672 - ∞]] --> [Females = range4 [217.500 - ∞]] (confidence: 0.880)

[Females = range4 [217.500 - ∞]] --> [Males = range4 [672 - ∞], Participants = range4 [879 - ∞]] (confidence: 0.880)

[Males = range4 [672 - ∞]] --> [Females = range4 [217.500 - ∞], Participants = range4 [879 - ∞]] (confidence: 0.880)

[Participants = range4 [879 - ∞]] --> [Females = range4 [217.500 - ∞], Males = range4 [672 - ∞]] (confidence: 0.880)

[Males = range2 [6 - 86]] --> [Golds = range1 [-∞ - 0]] (confidence: 0.889)

[Golds = range1 [-∞ - 0], Males = range3 [86 - 672], Females = range3 [22.500 - 217.500]] --> [Participants = range3 [116 - 879]] (confidence: 0.889)

[Golds = range1 [-∞ - 0], Females = range3 [22.500 - 217.500], Males = range2 [6 - 86]] --> [Participants = range2 [9.500 - 116]] (confidence: 0.889)

[Females = range2 [2 - 22.500]] --> [Golds = range1 [-∞ - 0]] (confidence: 0.890)

[Participants = range2 [9.500 - 116]] --> [Golds = range1 [-∞ - 0]] (confidence: 0.893)

[Golds = range1 [-∞ - 0], Males = range3 [86 - 672]] --> [Participants = range3 [116 - 879]] (confidence: 0.895)

[Males = range4 [672 - ∞], Participants = range4 [879 - ∞]] --> [Females = range4 [217.500 - ∞]] (confidence: 0.898)

[Participants = range2 [9.500 - 116], Females = range2 [2 - 22.500]] --> [Golds = range1 [-∞ - 0], Males = range2 [6 - 86]] (confidence: 0.898)

[Females = range4 [217.500 - ∞]] --> [Participants = range4 [879 - ∞]] (confidence: 0.900)

[Participants = range4 [879 - ∞]] --> [Females = range4 [217.500 - ∞]] (confidence: 0.900)

[Participants = range2 [9.500 - 116], Males = range2 [6 - 86]] --> [Golds = range1 [-∞ - 0]] (confidence: 0.900)

[Females = range4 [217.500 - ∞], Golds = range4 [14.500 - ∞]] --> [Males = range4 [672 - ∞]] (confidence: 0.905)

[Females = range4 [217.500 - ∞], Golds = range4 [14.500 - ∞]] --> [Males = range4 [672 - ∞], Participants = range4 [879 - ∞]] (confidence: 0.905)

[Males = range3 [86 - 672], Golds = range4 [14.500 - ∞]] --> [Participants = range3 [116 - 879]] (confidence: 0.909)

[Participants = range3 [116 - 879], Golds = range4 [14.500 - ∞]] --> [Males = range3 [86 - 672]] (confidence: 0.909)

[Golds = range1 [-∞ - 0], Males = range3 [86 - 672], Females = range2 [2 - 22.500]] --> [Participants = range3 [116 - 879]] (confidence: 0.909)

[Participants = range2 [9.500 - 116], Females = range2 [2 - 22.500]] --> [Golds = range1 [-∞ - 0]] (confidence: 0.915)

[Males = range3 [86 - 672]] --> [Participants = range3 [116 - 879]] (confidence: 0.923)

[Females = range4 [217.500 - ∞], Golds = range4 [14.500 - ∞]] --> [Participants = range4 [879 - ∞]] (confidence: 0.929)

[Females = range2 [2 - 22.500], Males = range2 [6 - 86]] --> [Golds = range1 [-∞ - 0]] (confidence: 0.930)

[Females = range2 [2 - 22.500], Males = range2 [6 - 86]] --> [Golds = range1 [-∞ - 0], Participants = range2 [9.500 - 116]] (confidence: 0.930)

[Participants = range2 [9.500 - 116], Females = range2 [2 - 22.500], Males = range2 [6 - 86]] --> [Golds = range1 [-∞ - 0]] (confidence: 0.930)

[Participants = range2 [9.500 - 116]] --> [Males = range2 [6 - 86]] (confidence: 0.933)

[Golds = range1 [-∞ - 0], Participants = range2 [9.500 - 116]] --> [Males = range2 [6 - 86]] (confidence: 0.940)

[Males = range3 [86 - 672], Females = range3 [22.500 - 217.500]] --> [Participants = range3 [116 - 879]] (confidence: 0.947)

[Females = range3 [22.500 - 217.500], Participants = range3 [116 - 879]] --> [Males = range3 [86 - 672]] (confidence: 0.947)

[Participants = range3 [116 - 879]] --> [Males = range3 [86 - 672]] (confidence: 0.960)

[Golds = range1 [-∞ - 0], Females = range3 [22.500 - 217.500], Participants = range3 [116 - 879]] --> [Males = range3 [86 - 672]] (confidence: 0.960)

[Females = range3 [22.500 - 217.500], Participants = range3 [116 - 879], Golds = range3 [0 - 14.500]] --> [Males = range3 [86 - 672]] (confidence: 0.960)

[Males = range3 [86 - 672], Golds = range3 [0 - 14.500]] --> [Participants = range3 [116 - 879]] (confidence: 0.966)

[Participants = range3 [116 - 879], Golds = range3 [0 - 14.500]] --> [Males = range3 [86 - 672]] (confidence: 0.966)

[Participants = range2 [9.500 - 116], Females = range2 [2 - 22.500]] --> [Males = range2 [6 - 86]] (confidence: 0.966)

[Golds = range1 [-∞ - 0], Participants = range3 [116 - 879]] --> [Males = range3 [86 - 672]] (confidence: 0.971)

[Males = range2 [6 - 86]] --> [Participants = range2 [9.500 - 116]] (confidence: 0.972)

[Golds = range4 [14.500 - ∞], Males = range4 [672 - ∞]] --> [Females = range4 [217.500 - ∞]] (confidence: 0.974)

[Golds = range4 [14.500 - ∞], Males = range4 [672 - ∞]] --> [Participants = range4 [879 - ∞]] (confidence: 0.974)

[Golds = range4 [14.500 - ∞], Participants = range4 [879 - ∞]] --> [Males = range4 [672 - ∞]] (confidence: 0.974)

[Golds = range4 [14.500 - ∞], Males = range4 [672 - ∞]] --> [Females = range4 [217.500 - ∞], Participants = range4 [879 - ∞]] (confidence: 0.974)

[Golds = range4 [14.500 - ∞], Participants = range4 [879 - ∞]] --> [Females = range4 [217.500 - ∞], Males = range4 [672 - ∞]] (confidence: 0.974)

[Females = range4 [217.500 - ∞], Golds = range4 [14.500 - ∞], Participants = range4 [879 - ∞]] --> [Males = range4 [672 - ∞]] (confidence: 0.974)

[Females = range4 [217.500 - ∞], Participants = range4 [879 - ∞]] --> [Males = range4 [672 - ∞]] (confidence: 0.978)

[Males = range4 [672 - ∞]] --> [Participants = range4 [879 - ∞]] (confidence: 0.980)

[Participants = range4 [879 - ∞]] --> [Males = range4 [672 - ∞]] (confidence: 0.980)

[Golds = range1 [-∞ - 0], Participants = range2 [9.500 - 116], Females = range2 [2 - 22.500]] --> [Males = range2 [6 - 86]] (confidence: 0.981)

[Golds = range1 [-∞ - 0], Males = range2 [6 - 86]] --> [Participants = range2 [9.500 - 116]] (confidence: 0.984)

[Participants = range3 [116 - 879], Females = range2 [2 - 22.500]] --> [Males = range3 [86 - 672]] (confidence: 1.000)

[Participants = range3 [116 - 879], Females = range4 [217.500 - ∞]] --> [Males = range3 [86 - 672]] (confidence: 1.000)

[Females = range3 [22.500 - 217.500], Golds = range4 [14.500 - ∞]] --> [Participants = range3 [116 - 879]] (confidence: 1.000)

[Females = range3 [22.500 - 217.500], Participants = range4 [879 - ∞]] --> [Males = range4 [672 - ∞]] (confidence: 1.000)

[Females = range3 [22.500 - 217.500], Participants = range4 [879 - ∞]] --> [Golds = range3 [0 - 14.500]] (confidence: 1.000)

[Females = range2 [2 - 22.500], Males = range2 [6 - 86]] --> [Participants = range2 [9.500 - 116]] (confidence: 1.000)

[Golds = range4 [14.500 - ∞], Participants = range4 [879 - ∞]] --> [Females = range4 [217.500 - ∞]] (confidence: 1.000)

[Females = range4 [217.500 - ∞], Males = range4 [672 - ∞]] --> [Participants = range4 [879 - ∞]] (confidence: 1.000)

[Males = range4 [672 - ∞], Golds = range3 [0 - 14.500]] --> [Participants = range4 [879 - ∞]] (confidence: 1.000)

[Participants = range4 [879 - ∞], Golds = range3 [0 - 14.500]] --> [Males = range4 [672 - ∞]] (confidence: 1.000)

[Golds = range1 [-∞ - 0], Participants = range3 [116 - 879], Females = range2 [2 - 22.500]] --> [Males = range3 [86 - 672]] (confidence: 1.000)

[Golds = range1 [-∞ - 0], Females = range2 [2 - 22.500], Males = range2 [6 - 86]] --> [Participants = range2 [9.500 - 116]] (confidence: 1.000)

[Males = range3 [86 - 672], Females = range3 [22.500 - 217.500], Golds = range4 [14.500 - ∞]] --> [Participants = range3 [116 - 879]] (confidence: 1.000)

[Males = range3 [86 - 672], Females = range3 [22.500 - 217.500], Golds = range3 [0 - 14.500]] --> [Participants = range3 [116 - 879]] (confidence: 1.000)

[Females = range3 [22.500 - 217.500], Participants = range4 [879 - ∞]] --> [Males = range4 [672 - ∞], Golds = range3 [0 - 14.500]] (confidence: 1.000)

[Females = range3 [22.500 - 217.500], Males = range4 [672 - ∞], Participants = range4 [879 - ∞]] --> [Golds = range3 [0 - 14.500]] (confidence: 1.000)

[Females = range3 [22.500 - 217.500], Males = range4 [672 - ∞], Golds = range3 [0 - 14.500]] --> [Participants = range4 [879 - ∞]] (confidence: 1.000)

[Females = range3 [22.500 - 217.500], Participants = range4 [879 - ∞], Golds = range3 [0 - 14.500]] --> [Males = range4 [672 - ∞]] (confidence: 1.000)

[Females = range4 [217.500 - ∞], Golds = range4 [14.500 - ∞], Males = range4 [672 - ∞]] --> [Participants = range4 [879 - ∞]] (confidence: 1.000)

[Golds = range4 [14.500 - ∞], Males = range4 [672 - ∞], Participants = range4 [879 - ∞]] --> [Females = range4 [217.500 - ∞]] (confidence: 1.000)

[Females = range4 [217.500 - ∞], Males = range4 [672 - ∞], Golds = range3 [0 - 14.500]] --> [Participants = range4 [879 - ∞]] (confidence: 1.000)

[Females = range4 [217.500 - ∞], Participants = range4 [879 - ∞], Golds = range3 [0 - 14.500]] --> [Males = range4 [672 - ∞]] (confidence: 1.000)