Università di Pisa

Dipartimento di Informatica

Laurea magistrale in Data Science and Business Informatics

LDS REPORT

Tennis matches Data Warehouse

2021/2022

Autori

Mario Proia 616679
Giada Traina 616682

Contents

1 Parte 1

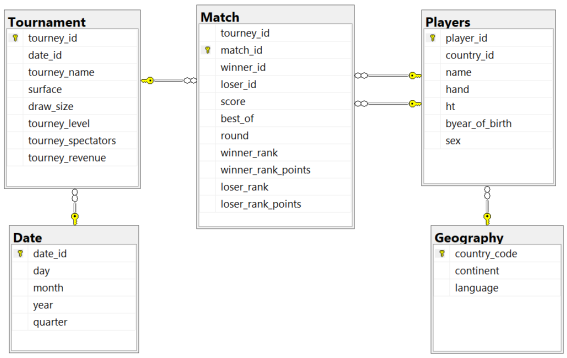
1.1 Creazione dello schema

In primo luogo, per poter realizzare lo schema in SQL Server Management Studio, abbiamo analizzato i diversi tipi di attributi presenti nei file csv, al fine di comprenderne la tipologia. In seguito, è stata utilizzata l'interfaccia grafica del software per inserire le chiavi primarie ed esterne dove necessario. In ognuno dei campi delle tabelle è stato specificato il data type e sono stati posti vincoli NOT NULL sia per le chiavi esterne sia per le chiavi primarie.

È opportuno specificare le chiavi primarie ed esterne:

- Match (PK_match_id, FK_tourney_id, FK_winner_id, FK_loser_id)
- Tournament (PK_tourney_id, FK_date_id)
- Players (PK_player_id, FK_country_id)
- Date (PK_date_id)
- Geography (PK_country_code)

Nella figura seguente è riportato lo schema risultante:



1.2 Manipolazione dei file csv

Per la realizzazione degli script di split sono state usate le funzionalità basilari di Python e la libreria csv.

1.2.1 Creazione della fact table Match e preprocessing per l'eliminazione dei duplicati

- Per realizzare la tabella Match, è stato necessario creare il "match_id" tramite la concatenazione del match number e del tourney_id, in modo tale da ottenere una chiave univoca. Questo è stato fatto estraendo "match_num" e "tourney_id" da tennis.csv, e appendendo in una lista la concatenazione dei due, in modo tale da preservare la posizione.
- Una volta ottenuto il match_id è stato possibile scrivere il file match.csv, ovvero un file grezzo in cui sono stati messi insieme tutti gli attributi necessari alla creazione della tabella Match, ma con la presenza di duplicati ed errori. Il file è stato creato iterando sulle righe di tennis.csv, e la colonna "tourney_id" è stata creata concatenando il tourney_id corrispondente al match, il tourney_level e il tourney_name; mentre il match_id è stato aggiunto riga per riga iterando sulla lista precedentemente creata.
- A questo punto, dopo aver ottenuto il file grezzo, sono state tolte le righe con match_id duplicati tramite iterazione sul file match.csv, tenendo traccia dei match_id incontrati tramite l'utilizzo di un set, e scrivendo la riga senza id duplicato sul file match_finale.csv. Allo stesso momento, in un altro set chiamato "errori", è stata tenuta traccia delle righe che avevano un id già incontrato, ma con almeno un valore diverso nelle righe: questo è servito a trovare i match diversi ma che avevano lo stesso id.

- Al termine della rimozione dei duplicati e della scrittura del file finale, le righe contenute nel set "errori" sono state aggiunte in coda al file finale, ma concatenando un numero incrementale intero al match_id, così da renderlo univoco.
- A questo punto la tabella è composta da: 185765 righe e 30 colonne.

1.2.2 Creazione della tabella dimensionale Players

- Per creare la tabella players, è stato necessario assegnare il Sex ai giocatori che provenivano dai due file separati "male_players.csv" e "female_players.csv". Questo step è stato eseguito leggendo prima il file con i males, da cui sono stati presi nome e cognome, per poi unirli in un unico valore per creare l'attributo "name". Se uno dei due fosse stato missing o errato, sarebbe stato preso solo quello corretto. I nomi ottenuti sono stati scritti su un nuovo file "output_males.csv", in cui è stata aggiunta la colonna "Sex" con valore "M", in quanto questi players erano tutti provenienti dal file con i males.
- Lo stesso step è stato ripetuto per le females provenienti dal file "females_players.csv", ma questa volta aggiungendo il valore "F" nella nuova colonna "Sex", scrivendo il file "output_females.csv".
- Una volta ottenuto il sex, sono state create due liste: una contenente le righe dei males e il relativo sex, e una lista per females, questa volta però saltando i duplicati (così da non dover rileggere i file più volte), e allo stesso momento è stata tenuta traccia dei nomi già incontrati tramite dei set (uno per nomi maschili e uno per nomi femminili).
- Per verificare la presenza dei players che sono sia in males che in females, è stata eseguita un'intersezione tra i due set di nomi, così da ottenere circa 30 persone di cui il sex era incerto. Per queste persone, il sex è stato verificato in base al sex dell'avversario contro cui hanno giocato almeno una partita, tramite il file tennis.csv.
- Avendo quindi ottenuto tutti i possibili players con sex corretto, è stato scritto un file unico chiamato "people.csv", in cui prima sono stati inseriti tutti i males, poi tutte le females e infine i players con sex appena verificato.
- In seguito, per creare la vera e propria tabella players.csv sono state inserite in una lista tutte le righe e gli attributi relativi ai winners e in un'altra lista tutte le righe e gli attributi relativi ai losers (per evitare più letture del file tennis.csv).
- A questo punto sono stati scritti i winners sul file players.csv tenendo traccia dei winner_id visitati (tramite un set apposito), aggiungendo il sex tramite il file people.csv, e l'anno di nascita calcolato come differenza fra la data in cui il player ha giocato una qualsiasi partita e l'età che aveva al momento di quella partita. Lo stesso è stato eseguito per i losers, ma facendo in modo tale da inserire solo quegli id non visti nel set contenente i winner_id, così da evitare players ridondanti.
- Inoltre, sono stati individuati degli errori che sono stati corretti manualmente: un giocatore italiano (nazionalità verificata tramite Wikipedia) presentava country_ioc = ITF, che è quindi stato corretto con ITA.
- In totale la tabella è composta da: 10123 righe e 7 colonne.

1.2.3 Creazione della tabella dimensionale Tournament

- Per creare questa tabella, sono state estratte le colonne necessarie direttamente dal file tennis.csv.
- I campi richiesti sono stati riempiti scrivendo Tournament, usando la libreria csv e il comando csv.DictReader.
- Per scrivere il campo "tourney_id", poiché si tratta di una chiave primaria, al fine di evitare duplicati, il valore è stato creato concatenando "tourney_id", "tourney_level" e "tourney_name". In tal modo vengono creati degli id univoci, senza duplicati.
- In totale la tabella è composta da: 4911 righe e 8 colonne.

1.2.4 Creazione della tabella dimensionale Date

- Dalla stringa "date_id" presente nel file tennis.csv sono stati ricavati i campi "anno", "mese" e "giorno", tramite slicing della stringa, considerando rispettivamente : dal primo al quarto valore

della stringa, quinto e sesto, settimo e ottavo.
- Successivamente, i mesi sono stati suddivisi in intervalli trimestrali, grazie ai quali sono state generate le relative stringhe Quarter1, Quarter2, Quarter3, Quarter4, che andranno a popolare il campo "quarter".
- In totale la tabella è composta da: 375 righe e 3 colonne.

1.2.5 Creazione della tabella dimensionale Geography

- I file utilizzati per la creazione di tale tabella Geography con attributi dimensionali quali "country_code", "continent" e "language", sono stati: tennis.csv, players.csv, country.csv e country_list.csv. Quest'ultimo sarà utilizzato per effettuare una data integration. Fonte: `www.fullstacks.io/2016/07/countries-and-their-spoken-languages.html`.
- Come primo passo sono state estratte le colonne "country_name" e "language" dal file country_list.csv, in modo tale da poter creare un dizionario con country_name come chiave, e language come valore.
- Poiché alcuni countries non avevano un nome corrispondente o presentavano degli errori di battitura, è stato necessario correggere gli errori tramite uno script che assegna il linguaggio ai countries mancanti/errati.
- Infine sono stati integrati circa 30 countries_ioc univoci dal file players.csv, che non erano già presenti nei file countries e country_list. Poiché i country_ioc non erano numerosi, sono stati recuperati i dati mancanti da wikipedia cercando il codice che identifica univocamente il country, e scrivendo nel file geography.csv il country_code, continent e language.
- In totale la tabella è composta da: 153 righe e 3 colonne.

1.3 Gestione dei missing values

Le tabelle dimensionali "Geography" e "Date" non presentano missing values o righe o id duplicati.

1.3.1 Preprocessing tabella Players

- Dopo la creazione della tabella players, è stata notata la presenza di alcune righe con player_id uguale e nome diverso. Per correggere questa incongruenza, è stato creato un dataset di players ma con tutti i possibili duplicati riutilizzando lo script per creare la tabella Players, in modo tale da non perdere nessuna persona.
- Con un'iterazione sul nuovo dataset con duplicati, è stata tenuta traccia delle righe che avevano id uguali e nome diversi, così da eliminare le righe duplicate riguardanti le stesse persone. Queste righe uniche sono state scritte su un file csv (chiamato players_no_dup_nomi.csv) per rendere più semplice la rilettura.
- Sull'ultimo dataset creato, è stata tenuta traccia delle righe con id uguale e almeno un valore di riga* (*si fa riferimento a tutti gli altri attributi presenti nella riga) diverso come già visto con il file match_finale.csv, tramite un set chiamato "errori". Le righe contenute in questo set sono state aggiunte in coda a players_finale, ma con player_id che partiva dal massimo valore di id trovato dai precedenti players + 1, e ad ogni inserimento l'id è stato incrementato di 1.
- A questo punto sono state aggiunte a match_finale.csv le colonne "winner_name" e "loser_name", per poter sostituire gli id dei giocatori errati, tramite il nome del giocatore. Per fare ciò è stato utilizzato un dizionario con chiave il nome del giocatore, e come valore il nuovo id estratto da players. Al termine della sostituzione sono state rimosse le colonne winner_name e loser_name da match.

1.3.2 Missing values tabella Match

I missing values presenti in Match riguardavano in particolare un gran numero di colonne, quali *'minutes', 'w_ace', 'w_df', 'w_svpt', 'w_1stIn', 'w_1stWon', 'w_2ndWon', 'w_SvGms', 'w_bpSaved', 'w_bpFaced', 'l_ace', 'l_df', 'l_svpt', 'l_1stIn', 'l_1stWon', 'l_2ndWon', 'l_SvGms', 'l_bpSaved','l_bpFaced'*, le quali presentano circa di 103000 missing values, su un totale di 185764 records. Per questo motivo è stato deciso

di eliminare tali colonne.
Anche i missing values (i records) presenti nella colonna score sono stati eliminati, in quanto sarebbe stato scorretto risalire ai valori tramite una moda, poiché ciò avrebbe potuto portare ad una scorretta attribuzione degli score, per esempio, mettendo score vincenti a giocatori perdenti e viceversa.
Per la gestione dei missing values in winner_rank, loser_rank, winner_rank_points e loser_rank_points, è stato deciso di effettuare una media, raggruppando per anno e id del giocatore. Questo perchè il rank rappresenta un insieme di punteggi ottenuti nell'anno passato da parte di un giocatore. Successivamente, tramite questa sostituzione, sono comunque rimasti dei valori mancanti (circa 12000 per winner_rank e winner_rank_points e 26000 per loser_rank e loser_rank_points), i quali sono stati sostituiti con valore "-1". Questo perchè, tramite un'eliminazione delle suddette righe, avremmo rischiato di perdere una grande quantità di dati. Inoltre, la sostituzione è stata applicata al fine di non avere campi vuoti nel DW.
Il totale delle righe 185590 righe.

1.3.3 Missing values tabella Tournament

Nella tabella Tournament è stata riscontrata la presenza di 62 missing values all'interno della colonna "surface". Per gestire tali campi nulli:

1. è stata analizzata la distribuzione della variabile, la quale risulta così ripartita: Hard = 2600, Clay = 2052, Grass = 110 e Carpet = 88;
2. i valori nulli sono stati sostituiti con la moda, ovvero con il valore "Hard".

1.3.4 Missing values tabella Players

I missing values riscontrati in players sono stati i seguenti: 33 in hand, 9599 in ht, 2110 in byear_of_birth e 34 in sex. I missing values di sex sono stati controllati manualmente dal sito `https://peoplepill.com/people`. Quest'ultimo ha confermato che l'unica donna presente fosse "Alona Fomina". Perciò si è proceduto a sostituire solo quest'ultima con il valore "F".

Il valore hand è composto da 6286 campi "U", 3402 campi "R" e 402 campi "L". Poiché è stato verificato che il valore "U" presenta il significato di "undefined", tali valori sono stati convertiti in nan. In seguito è stata analizzata la distribuzione percentuale di "L" e "R" nell'intero dataset. Il risultato ottenuto ha mostrato come il 90% dei valori fossero "R" e il 10% "L". Per tal ragione si è proceduto a sostituire i missing values in modo randomico, ma mantenendo questa distribuzione percentuale.

I missing values relativi al valore HT sono stati sostituiti effettuando prima un raggruppamento per "genere" e "continent" e poi una media. Per tale task è stato necessario effettuare un join con la tabella "geography" per ricavare la colonna "continent", assente in players.

In seguito è stato posto in essere un controllo relativo alle altezze, per verificare se ci fossero giocatori con altezza minore di 150cm. Così facendo sono state trovate due righe con due giocatori alti rispettivamente 2cm (probabilmente dovuto ad un errore di trascrizione) e 145cm. Successivamente è stato ricercato il loro nome sul sito `https://it.wikipedia.org/wiki/Pemra_%C3%96zgen` per sostituire il campo con le reali altezza dei tennisti.

Per i restanti missing values nel campo "byear_of_birth", in primo luogo, si è cercato di individuare all'interno del file tennis.csv se vi fossero delle età non considerate dal codice, ma senza alcun riscontro. In seguito è stata effettuata una ricerca per individuare dei dataset con cui porre in essere una data integration, ma anche in questo caso, senza alcun riscontro. Per tal motivo, le righe che presentavano quei valori nulli sono state sostituite con valore "-1"; in quanto, la loro eliminazione avrebbe creato delle inconsistenze con le chiavi esterne presenti in match (winner_id e loser_id).
Il totale delle righe sarà adesso di 10123.

1.4 Popolamento del DB

Per l'inserimento dei dati nelle tabelle nel DB, sono stati eseguiti degli script separati per ogni tabella. Ogni script effettua una connessione al DB da remoto; poi, grazie all'oggetto Cursor, esegue la query INSERT nella tabella corrispondente, per ogni riga del file, tramite un ciclo FOR. In particolare, per i file di grandi dimensioni come match, players e tournament, è stato effettuato uno script per dividerli ad intervalli di circa 1000 righe. Questo ha permesso di effettuare commit con file più piccoli, in modo da evitare interruzioni di scrittura dovute a cali di connessione.

2 Parte 2: Soluzioni SSIS

2.1 Assignment 0

L'Assignment richiede di ottenere per ogni anno, i "tournaments" ordinati per numero di spettatori.

La soluzione proposta (figura 1) ha previsto inizialmente di leggere dal DB la tabella Tournament, dalla quale sono state selezionate le colonne "tourney_id", "tourney_name", "date_id" e "tourney_spectators". Inoltre, tramite lookup con la tabella Date, è stata inserita anche la colonna year.
Successivamente è stato utilizzato il nodo "Aggregazione" per eseguire un raggruppamento su "tourney_id", "tourney_name" e "year" ed una sommatoria su "tourney_spectators". Il risultato dell'aggregazione è stato poi ordinato per year in maniera crescente e per tourney_spectators in maniera decrescente.
Infine, il risultato ottenuto è stato scritto su un file txt tramite il nodo "Destinazione file flat".
La soluzione di riferimento è Assignment_0_alt.sln.

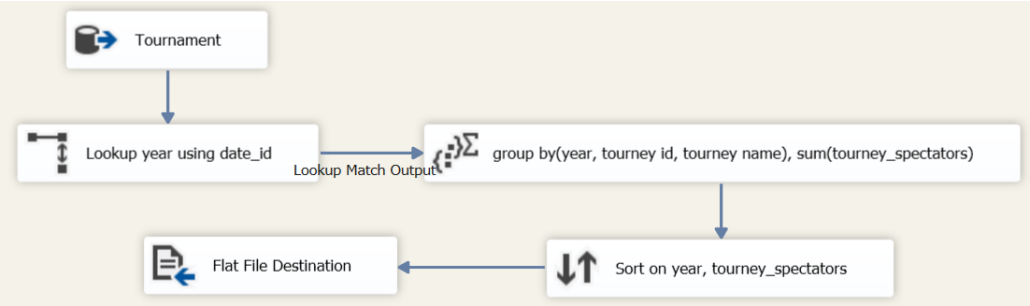


Fig. 1: Soluzione Assignment 0

Inoltre, poiché il raggruppamento viene effettuato su tourney_id, il quale è univoco (essendo una chiave primaria), è stata proposta una seconda soluzione per l'Assignment (figura 2). Quest'ultima omette il raggruppamento, proponendo direttamente un'operazione di sorting in seguito al lookup.
Tale alternativa è presentata in Assignment_0_nogroup.sln.

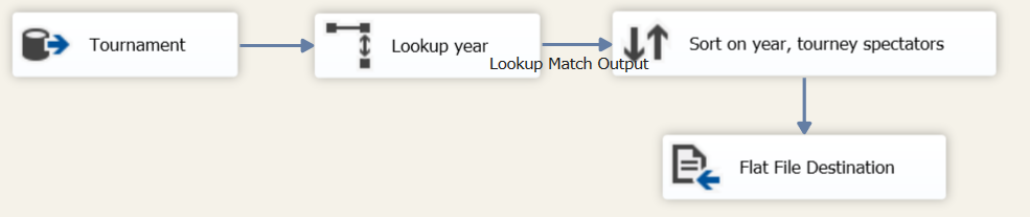


Fig. 2: Soluzione Assignment 0 senza raggruppamento

2.2 Assignment 1

L'Assignment richiede di calcolare per ogni player, la sua "nemesi", la quale rappresenta il player contro cui lui/lei ha perso più match. Elencando, inoltre, ogni player con la rispettiva nemesi e il numero dei match persi.

In questo flusso (figura 3) è stata inizialmente letta la tabella Match, recuperando le colonne "loser_name" e "winner_name" dalla tabella Players, tramite lookup.
Al fine di ottenere il numero di match giocati da un giocatore e un qualsiasi avversario è stata eseguita un'aggregazione, raggruppando per winner_id, loser_id, winner_name, loser_name ed eseguendo un count(*).
Successivamente è stato utilizzato un nodo "Multicast", per dividere il flusso originale in due flussi uguali su cui lavorare separatamente. Sul primo è stato effettuato un raggruppamento, aggregando su loser_name e contando per ogni "loser" il numero massimo di match persi.

In seguito è stato effettuato un ordinamento di entrambi i flussi sul loser_name, prima dell'operazione di merge join.
Infine, è stato utilizzato un nodo di suddivisione condizionale per ottenere solo le righe in cui il numero dei match giocati da un "loser_name" fosse uguale al MaxMatchLost, in modo tale da selezionare solo i loser_name con la rispettiva Nemesi, ovvero l'avversario contro cui lui/lei ha perso il maggior numero di match.
La query è stata conclusa scrivendo i risultati ottenuti su un file txt.
La soluzione di riferimento è Assignment_1_nuovo.sln.

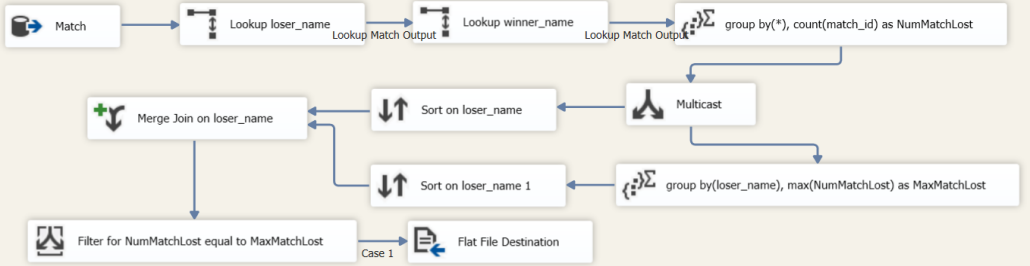


Fig. 3: Soluzione Assignment 1

2.3 Assignment 2

L'Assignment richiede di calcolare per ogni anno, il nome del torneo con il maggior numero di match giocati.

Per eseguire la query (figura 4) , prima di tutto, è stata effettuata la connessione alla tabella Match. In seguito sono state inserite le colonne "tourney_id" e "date_id", ottenute tramite un'operazione di lookup eseguita rispettivamente sulle tabelle Tournament e Date.
In seguito è stato effettuato un raggruppamento sui campi "match_id", "tourney_id", "tourney_name" e "year", al fine di contare il numero dei match giocati durante ogni torneo.
Successivamente il flusso è stato diviso in due parti tramite il nodo "Multicast". Per il flusso di sinistra è stato eseguito un raggruppamento per year, al fine di calcolare il numero massimo dei match giocati per ogni anno. Il flusso si è poi concluso con un ordinamento crescente su year.
Nel flusso di destra è stato eseguito semplicemente un ordinamento crescente su year. I due flussi sono stati, in seguito, uniti tramite l'operatore merge join.
Infine, tramite un nodo di suddivisione condizionale, sono state filtrate solo le righe dove il numero dei match giocati fosse uguale al numero massimo dei match per ogni anno.
La query è stata conclusa scrivendo i risultati ottenuti su un file txt.
La soluzione di riferimento è Assignment_2.sln.

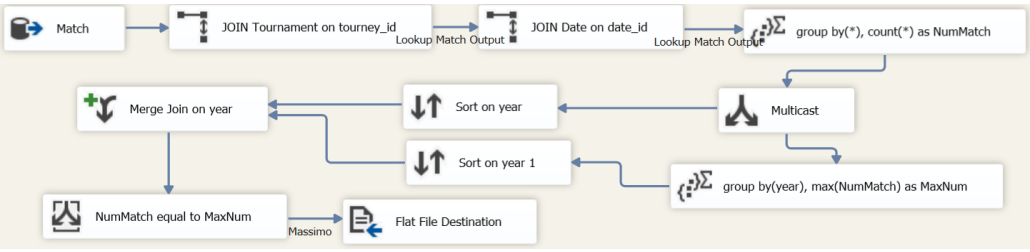


Fig. 4: Soluzione Assignment 2

3 Parte 3: OLAP Cube

3.1 Creazione del cubo

Il primo passo effettuato per la creazione del cubo, è stato quello di creare una Data Source View sul Database Group_2_DB. Dopodiché è stata creata la dimensione Tournament, in cui sono stati inclusi sia gli attributi di Tournament sia gli attributi di Date in modo tale da creare la gerarchia "DayMonthQuarterYear": Year → Quarter → Month → Day → Tourney Id. Per ogni attributo della gerarchia è stato imposto l'ordinamento per "Key", in quanto si tratta di valori interi e non di stringhe, in modo da garantire l'ordinamento. Lo stesso è stato fatto per la tabella Players, creando la gerarchia "Geography": Continent → Country Code → Player Id. Una volta create le dimensioni è stato creato il cubo importando tutti i dati e creando le apposite misure nella fact table.
Più nel dettaglio, è stata creata una misura aggiuntiva chiamata "n_distinct_winners", ovvero il numero di winner_id distinti che verrà utilizzato per le successive query in MDX. Per tutte le misure è stato selezionato "Standard" come "FormatString" dato che non sono presenti misure con formati particolari. Inoltre, è stato verificato che la "AggregateFunction" fosse corretta per ogni misura: per n_distinct_winners è stata settata la funzione "DistinctCount", mentre per winner_rank, winner_rank_points, loser_rank, loser_rank_points è stata settata la funzione "Sum".

3.2 Query MDX 1

La prima query richiede di mostrare il numero dei n_distinct_winners in relazione al paese e il totale rispetto ad ogni continente.

Per risolvere la query sono state utilizzate le funzioni "Children" su Continent e "Members" sul Country_code per ottenere le possibili combinazioni dei country_code e dei continenti nelle righe ed il numero dei rispettivi winners distinti nelle colonne, utilizzando la funzione "nonempty" per evitare di ottenere celle vuote nel risultato. In questo modo è stato possibile ottenere in output prima il totale dei n_distinct_winners per continente e poi i singoli n_distinct_winners per ogni country_code.
La query risultante è la seguente:

```
select [Measures].[n_distinct_winners] on columns,
nonempty((
[Winner].[Geography].children,
[Winner].[Country Code].members
)) on rows
from [Group 2 DB]
```

Fig. 5: Query MDX 1

3.3 Query MDX 2

La seconda query richiede di mostrare il totale dei winner_rank_points per ogni anno e la somma cumulata annuale dei winner rank points dei giocatori Europei.

Per prima cosa, nella select è stata utilizzata la gerarchia "DayMonthQuarterYear" nelle righe per accedere all'anno, evitando di mostrare in output valori vuoti grazie all'utilizzo della funzione "nonempty". In seguito sono stati selezionati i tre output da inserire nelle colonne, quali: i winner_rank_points totali per ogni anno, winner_rank_points totali per ogni cittadino membro di un paese dell'UE e le somme cumulate dei winner_rank_points per i cittadini dei paesi membri dell'UE. Dopodiché è stato creato il membro "EU_running_total" in cui sono state impiegate le funzioni Sum() e Periodstodate() al fine di ottenere per ogni anno la somma tra i winner rank points dell'anno corrente e quella di tutti gli anni precedenti. Inoltre, sono stati selezionati solo i winner rank points relativi ai giocatori europei. Infine, è stato creato il membro "European_Members" per mostrare solo il totale annuale dei winner rank points dei giocatori europei, per completezza, al fine di rendere l'output più leggibile.

```
with member EU_running_total as
(
sum( PERIODSTODATE ([Tournament].[DayMonthQuarterYear].[All].level,
[Tournament].[DayMonthQuarterYear].currentmember),
(([Winner].[Geography].[Continent].&[Europe],[Measures].[Winner Rank Points]))
)
)

member European_Members as
(([Winner].[Geography].[Continent].&[Europe],[Measures].[Winner Rank Points])
)

select {[Measures].[Winner Rank Points], European_Members, EU_running_total} on columns,
nonempty(([Tournament].[DayMonthQuarterYear].[Year])) on rows
from [Group 2 DB]
```

Fig. 6: Query MDX 2

3.4 Query MDX 3

Questa query richiede di mostrare il rapporto tra il totale annuale dei winner rank points, rispetto al totale dell'anno precedente.

Inizialmente nella clausola select è stata utilizzata la gerarchia "DayMonthQuarterYear" nelle righe per accedere all'anno, evitando di mostrare in output valori vuoti grazie all'utilizzo della funzione "nonempty". In seguito sono stati selezionati i due output da inserire nelle colonne, rispettivamente: winner_rank_points e il Previous_year_ratio.
In seguito è stato creato il membro "Previous_year_ratio" in cui viene calcolato il rapporto tra il totale annuale dei winner rank points dell'anno corrente e il totale dell'anno precedente (tramite la funzione "lag(1)"), utilizzando la funzione "round" al fine di ottenere risultati con 4 cifre decimali dopo la virgola. Nel caso in cui l'anno precedente risulti assente, il ratio sarà pari a 0 automaticamente, per evitare di avere "Inf" come output.

```
with member Previous_year_ratio as (
    iif (([Tournament].[DayMonthQuarterYear].currentmember.lag(1), [Measures].[Winner Rank Points])=0, 0,
        Round([Measures].[Winner Rank Points] / ([Tournament].[DayMonthQuarterYear].currentmember.lag(1),
        [Measures].[Winner Rank Points]), 4)
    )
)

select {[Measures].[Winner Rank Points], Previous_year_ratio} on columns,
nonempty([Tournament].[DayMonthQuarterYear].[Year].members) on rows
from [Group 2 DB]
```

Fig. 7: Query MDX 3

3.5 Dashboard

Per ogni dashboard realizzata tramite PowerBI è possibile navigare le gerarchie o filtrare i dati. Per tale ragione, le immagini che verranno riportate rappresentano solo un'estrazione di una specifica granularità.

3.5.1 Assignment 4

Distribuzione geografica dei winner rank points e loser rank points.
Per lo svolgimento di questo task è stato scelto di utilizzare la visualizzazione "mappa", disponendo la gerarchia "Geography" in "località" e le rispettive misure winner_rank_points e loser_rank_points in "dimensioni". Dal grafico sottostante è possibile vedere i risultati ottenuti per loser_rank_points a sinistra e winner_rank_points a destra. Inoltre, sono stati aggiunti dei line plot che mostrano il medesimo risultato calcolato sopra, ma con una leggenda contenente il sesso dei giocatori.

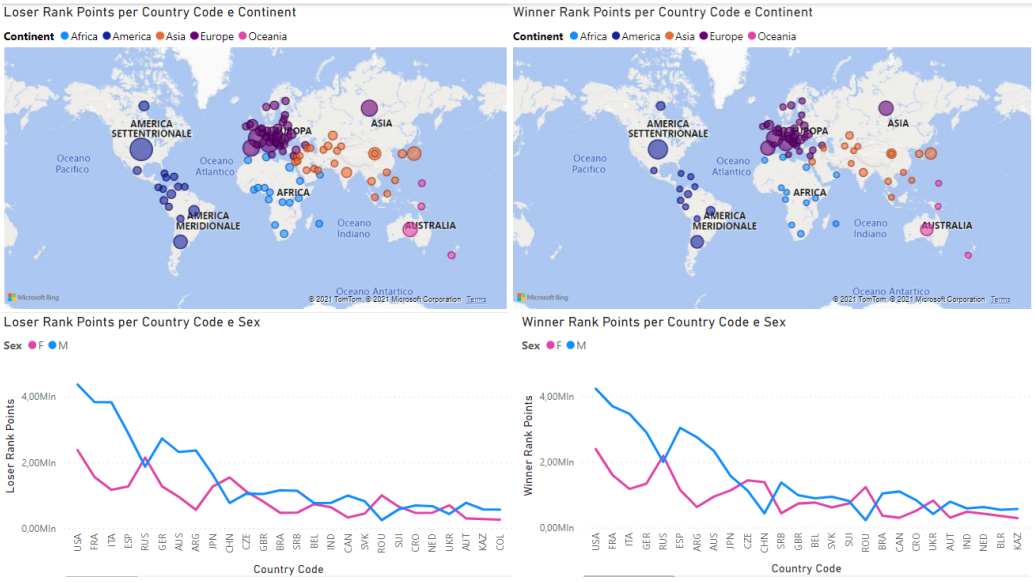


Fig. 8: Dashboard

3.5.2 Assignment 5

Tramite PowerBI è stato possibile creare dei report riguardanti l'andamento dei winner_rank_points e dei n_distinct_winners divisi per sesso e anno, sesso e continente, sesso e livello del torneo, sesso e country code.
Come è possibile vedere dal grafico a colonne in pila (in basso a destra), il numero dei giocatori vincitori di sesso femminile è notevolmente superiore rispetto ai vincitori di sesso maschile. D'altro canto, osservando il grafico a barre in pila percentuale (in basso a sinistra), è possibile notare che i winner rank points dei giocatori di sesso maschile sono, in quattro continenti su cinque, superiori a quelli dei giocatori di sesso femminile.
Inoltre, grazie al grafico a torta (in alto a sinistra) è possibile notare come, negli anni relativi alla diffusione della pandemia (2020-2021), siano stati ottenuti un numero di winner rank points notevolmente inferiore, probabilmente per il minor numero di tornei giocati. Infine, grazie all'utilizzo della mappa ad albero, è possibile notare come la maggior parte dei winner rank points sia contenuta nei tornei di livello C ed A.

Fig. 9: Dashboard a scelta

Grazie all'interattività dei plot del software, cliccando su un dato specifico è possibile adattare i

grafici presenti, ottenendo, come nell'esempio sottostante, una panoramica su come sono distribuiti i winner rank points per tornei di livello C.

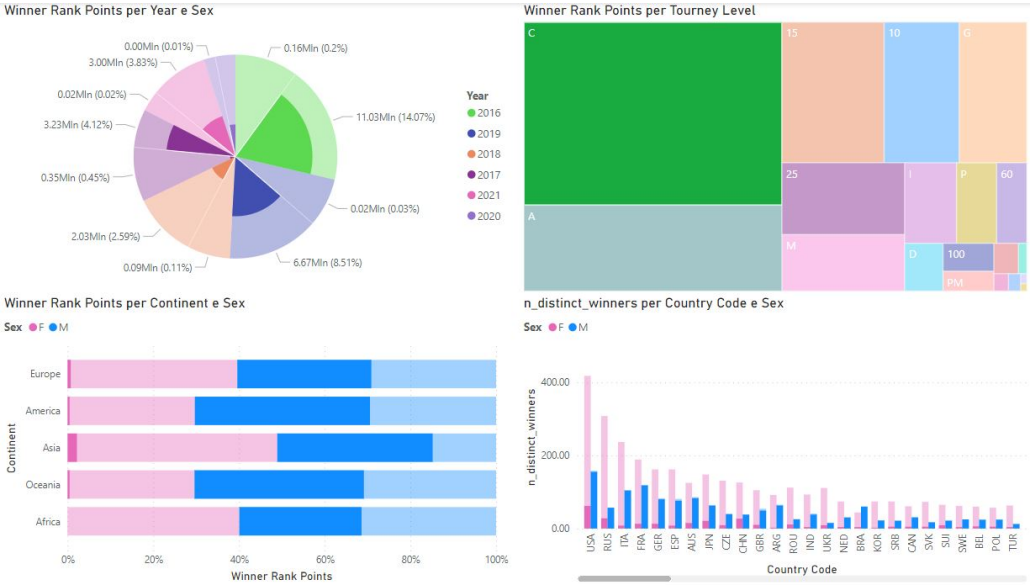


Fig. 10: Dashboard a scelta con filtro