

**Syntra Midden Vlaanderen**



**PGA TOUR RENDABILITEIT  
TUSSEN 2001 - 2024**

Een eindwerk voor de opleiding van Data Analist

Leerkracht

door

**Dieter van Itterbeeck**

**Michaël Redant**

2025

## **SAMENVATTING**

Totaal aantal woorden: 10653

Totaal aantal pagina's: 67

Michaël REDANT – PGA Tour Rendabiliteit tussen 2001 - 2024

Eindwerk voor de opleiding van data analist

Lesjaar: 2024 - 2025

## VOORWOORD

Voor u ligt mijn eindwerk, opgesteld in het kader van de opleiding tot data-analist. In dit project ga ik op zoek naar welke spelers op de PGA Tour tussen 2001 en 2024 het meest economisch rendabel waren op basis van hun verdiensten per gespeelde slag.

Mijn interesse in de golfsport heeft een belangrijke rol gespeeld in de keuze voor dit onderwerp. Als golfliefhebber en datadenker leek het me bijzonder boeiend om financiële efficiëntie in de golfsport op een cijfermatige manier in kaart te brengen. Dat ik de kans kreeg om dit thema te analyseren binnen het kader van mijn eindwerk, maakte het des te aangenamer om eraan te werken.

Ik wil graag mijn leerkracht **Dieter van Itterbeeck** bedanken voor zijn deskundige begeleiding en praktische feedback tijdens dit traject. Zijn hulp heeft ertoe bijgedragen dat ik dit eindwerk tot een goed einde kon brengen.

## INHOUDSTABEL

<i>Samenvatting</i> .....	<i>II</i>
<i>Voorwoord</i> .....	<i>III</i>
<i>Inhoudstabel</i> .....	<i>IV</i>
<i>Introductie</i> .....	7
<b>1. Datasetbeschrijving .....</b>	<b>8</b>
§ 1. Overzicht van de dataset .....	8
§ 2. Verduidelijking van gebruikte variabelen .....	8
§ 3. Voorbeeld uit de dataset .....	9
§ 4. Gebruikte datatypen en tools .....	10
§ 5. Uitleg golf technisch jargon .....	11
<b>2. Relatieel datamodel .....</b>	<b>12</b>
§ 1. Conceptueel datamodel .....	12
§ 2. Logisch datamodel .....	13
§ 3. Fysiek datamodel .....	14
a. Tabel: dim_player.....	14
b. Tabel: dim_location.....	14
c. Tabel: dim_tournament.....	14
d. Tabel: dim_date .....	15
e. Tabel: fact_score_round .....	15
§ 4. Relatieel datamodel (RDM) .....	17
<b>3. Data cleansing en voorbereiding ETL.....</b>	<b>19</b>
§ 1. Importeren in een staging-omgeving.....	19
§ 2. Duplicatie naar een werkversie .....	19
§ 3. Splitsingen .....	20
a. Splitslogica namen .....	20
b. Splitslogica locatie.....	20
§ 4. Transformatie ronde gegevens .....	22
a. Creatie van de tussenliggende tabel: round_score_fact .....	22
b. INSERT-query met validatie en conversieoplossing.....	23
§ 5. Validatie en correctie van status_code en valid_score .....	24
a. Corrigeren van status_code.....	25
b. Corrigeren van valid_score.....	25
c. Validatiequeries .....	25
d. Latere SQL aanpassingen en optimalisaties .....	26
Toevoeging van position_numeric (genormaliseerde positie) .....	26
Toevoeging van is_tie-indicator.....	26
Toevoeging van fedex_points .....	27
Herindeling van fact_id als surrogaat sleutel .....	27
<b>4. Dimensioneel model .....</b>	<b>28</b>
§ 1. Van staging naar dimensioneel model .....	28
§ 2. Dimensionele tabellen .....	29

<b>5. SSIS ETL in Visual studio .....</b>	<b>31</b>
<b>§ 1. Doelstelling .....</b>	<b>31</b>
<b>§ 2. Opbouw van de SSIS-structuur .....</b>	<b>31</b>
<b>§ 3. Foutafhandeling via Redirect Rows .....</b>	<b>32</b>
<b>§ 4. Uitdagingen en oplossingen onderweg .....</b>	<b>32</b>
a. SSIS Fase – ETL-uitdagingen .....	32
b. Power BI Fase – Visualisatie & DAX uitdagingen .....	33
<b>§ 5. Opmak SSIS in detail .....</b>	<b>33</b>
a. Voorbereiding: leegmaken van fact_score_round .....	33
b. OLE DB Source: inlezen van round_score_fact .....	34
c. Derived Column: DC_Player.....	35
d. Lookup Player: koppeling met dim_player.....	36
Foutafhandeling: Player_error_file .....	37
e. Derived Column: DC_Tournament .....	38
f. Lookup Tournament: koppeling met dim_tournament .....	39
Foutafhandeling: Tournament_error_file .....	41
g. Lookup Date: koppeling met dim_date .....	41
h. Lookup Location: koppeling met dim_location.....	43
Foutafhandeling: Location_error_file .....	45
i. Sort: voorbereiding op laadinstructie.....	46
j. OLE DB Destination: laden in fact_score_round .....	46
<b>6. Python: automatische profielgeneratie.....</b>	<b>49</b>
<b>§ 1. Doel van het script .....</b>	<b>49</b>
<b>§ 2. Opbouw van het script .....</b>	<b>49</b>
<b>§ 3. Finale python script .....</b>	<b>51</b>
<b>§ 4. Peformantie .....</b>	<b>56</b>
<b>§ 5. Resultaat python script .....</b>	<b>57</b>
<b>7. Visualisatie in Power BI.....</b>	<b>58</b>
<b>§ 1. Structuur van het Power BI-model .....</b>	<b>58</b>
<b>§ 2. Measures en DAX-logica .....</b>	<b>59</b>
a. Scorestatistieken .....	59
b. Verdiensten en efficiëntie .....	59
c. Participatie en frequentie .....	59
d. Cut & success metrics.....	59
e. Seizoensspecifieke analyses .....	59
f. Vergelijkende KPI's via disconnected slicer.....	60
g. Overzicht van meest gebruikte DAX-technieken .....	60
<b>§ 3. Disconnected table voor vergelijkende analyses .....</b>	<b>60</b>
<b>§ 4. Dashboards en Visualisaties.....</b>	<b>61</b>
a. Startpage .....	61
b. Player dashboard.....	62
c. Season dashboard .....	63
d. Golfclub dashboard.....	64
<b>8. Conclusie &amp; reflectie .....</b>	<b>65</b>
<b>Lijst van afbeeldingen .....</b>	<b>66</b>
<b>Lijst van tabellen .....</b>	<b>67</b>



## INTRODUCTIE

De professionele golfsport is uitgegroeid tot een miljardenindustrie, waarbij de PGA Tour (Professional Golfers' Association Tour) geldt als het meest prestigieuze golftoernooi ter wereld. Elk jaar nemen honderden topspelers deel aan tientallen toernooien, waarbij aanzienlijke geldprijzen worden verdeeld op basis van individuele prestaties. Deze geldstromen vormen niet enkel de levensader van spelers, maar ook van sponsors, toernooiorganisatoren en mediapartners.

Initieel in dit eindwerk ging ik op zoek naar de meest economisch rendabele PGA Tour-spelers tussen 2001 en 2024. Niet op basis van overwinningen of totaal verdiende bedragen, maar via een alternatieve maatstaf: verdiensten per gespeelde slag (*earnings per stroke*). Deze KPI biedt een genormaliseerde kijk op efficiëntie en maakt het mogelijk om niet alleen toppers te detecteren, maar ook consistente spelers die op relatief korte tijd een opvallend financieel rendement realiseren.

Het belang van deze analyse ligt in de praktische toepasbaarheid voor stakeholders in de golfwereld. Sponsorfirma's zijn voortdurend op zoek naar spelers die maximale return on investment (ROI) opleveren. Een speler die systematisch hoog scoort qua inkomsten per slag, kan voor merken interessanter zijn dan een meer wisselende, maar bekendere naam. Ook voor sportjournalisten, coaches en teammanagers kan deze informatie waardevol zijn in de context van prestatieanalyse, talentdetectie en carrièreplanning.

De opbouw van dit project verliep in verschillende fasen. In eerste instantie gebruikte ik een volledige ETL-architectuur met SQL Server en SSIS. Op basis van het Kimball-stermodel ontwierp ik een dataplatform waarin ruwe data uit de stagingomgeving stap voor stap opgeschoond, genormaliseerd en omgezet werd naar een analyseerbare structuur. Met technieken zoals conditional splits, surrogate keys, lookup-transformaties en foutafhandeling werd een solide fundament gelegd waarop visualisaties gebouwd konden worden. In de volgende fase voor Python om de ruwe dataset – afkomstig van Kaggle – te verkennen en te controleren op datakwaliteit. Hiermee kon ik snel foutieve of ontbrekende velden detecteren, duplicaten opsporen en alvast enkele eerste inzichten formuleren. Python bood de nodige flexibiliteit om data te verkennen, maar bleek minder geschikt om een gestructureerd en reproduceerbaar datamodel op te bouwen.

Het uiteindelijke doel was om de inzichten niet alleen te berekenen, maar ook begrijpelijk en interactief te presenteren. Met Power BI bouwde ik een reeks dashboards op drie niveaus:

- **Spelersdashboard:** met individuele KPI's zoals earnings per stroke, cut-succesratio, FedEx-punten en een historisch overzicht van meest rendabele toernooien.
- **Seizoensdashboard:** met analyses per jaar, vergelijkingen tussen spelers, moeilijkheidsgraad van toernooien en verdeling van verdiensten doorheen het seizoen.
- **Golfclubdashboard:** met een focus op prestaties per locatie, gemiddelde scores per ronde, laagste gescoorde rondes en winnaars per golfclub.

Dankzij dit model kon ik uiteindelijk veel verder gaan dan de oorspronkelijke onderzoeksverzoek. Zo maakte ik vergelijkingen mogelijk tussen spelers binnen een seizoen of over hun hele carrière, detecteerde ik uitzonderlijk scorende spelers op specifieke locaties en identificeerde ik de meest rendabele maanden en toernooien op basis van gemiddelde inkomsten per slag. Dit eindwerk is daarmee niet alleen een antwoord op een originele onderzoeksverzoek, maar ook een demonstratie van hoe datagedreven inzichten tot stand komen via een robuust model, duidelijke businesslogica en visuele vertaalslagen naar stakeholders.

## 1. DATASETBESCHRIJVING

Voor deze studie werd gebruik gemaakt van de dataset “**PGA Tour Golf Results (2001–2024)**”, beschikbaar via het platform [Kaggle.com](#). De dataset bevat officiële resultaten van alle PGA Tour-toernooien die plaatsvonden tussen januari 2001 en december 2024.

Elke rij in de dataset stelt een unieke combinatie voor van een speler, een toernooi en een seizoen, en bevat informatie over de prestaties, eindpositie en verdiend prijzengeld van de speler.

### § 1. OVERZICHT VAN DE DATASET

De oorspronkelijke dataset bevat **143.289 records** en bevat de volgende kolommen:

KOLOMNAAM	BESCHRIJVING
<b>SEASON</b>	Jaar van het toernooi
<b>START / END</b>	Start- en einddatum van het toernooi
<b>TOURNAMENT</b>	Naam van het toernooi
<b>LOCATION</b>	Locatie van het toernooi
<b>POSITION</b>	Eindpositie van de speler
<b>NAME</b>	Naam van de speler
<b>SCORE</b>	Score tegenover par (bv. -9)
<b>ROUND 1 – ROUND 4</b>	Score per ronde (stroke count)
<b>TOTAL</b>	Totaal aantal gespeelde slagen
<b>EARNINGS</b>	Verdiend prijzengeld in dollar (USD)
<b>FEDEX POINTS</b>	FedEx Cup-punten (niet beschikbaar in de beginjaren)

Tabel 1 Ovezicht dataset

De gegevens zijn gestructureerd op toernooiniveau: elke rij geeft dus de prestaties van één speler weer in één specifiek toernooi. De kolom FedEx points is opgenomen in het databestand, maar bevat lege waarden in de oudere records. Aangezien deze kolom inconsistent gevuld is, werd ze niet opgenomen in de verdere analyse.

### § 2. VERDUIDELIJKING VAN GEBRUIKTE VARIABELEN

De focus van deze analyse ligt op de economische efficiëntie van een speler, uitgedrukt als verdiende dollars per gespeelde slag. Hiervoor werd een afgeleide variabele `earnings_per_stroke` berekend op basis van:

Earnings

$$\text{Earnings per stroke} = \frac{\text{Earnings}}{\text{Total}}$$

Bij de selectie van variabelen werd rekening gehouden met volgende aspecten:

- De kolom score werd bewust genegeerd in deze context, aangezien het gaat om een relatieve score ten opzichte van par, die niet noodzakelijk samenhangend is met verdiende inkomsten.
- Enkel records waarbij zowel earnings als total geldig zijn ingevuld, én waar total > 0 is, werden opgenomen in de analyse.

### § 3. VOORBEELD UIT DE DATASET

Onderstaande tabel toont een fragment van de eerste toernooiresultaten in de dataset:

season	start	end	tournament	location	position	name	score	round1	round2	round3	round4	total	earnings
2001	11/01/2001	14/01/2001	Touchstone Energy Tucson Open	Omni Tucson National Golf Resort and Spa - Tucson, AZ	1	Garrett Willis	-15	71	69	64	69	273	540000
2001	11/01/2001	14/01/2001	Touchstone Energy Tucson Open	Omni Tucson National Golf Resort and Spa - Tucson, AZ	2	Kevin Sutherland	-14	67	72	67	68	274	324000
2001	11/01/2001	14/01/2001	Touchstone Energy Tucson Open	Omni Tucson National Golf Resort and Spa - Tucson, AZ	T3	Bob Tway	-13	73	69	67	66	275	174000
2001	11/01/2001	14/01/2001	Touchstone Energy Tucson Open	Omni Tucson National Golf Resort and Spa - Tucson, AZ	T3	Geoff Ogilvy	-13	67	72	68	68	275	174000
2001	11/01/2001	14/01/2001	Touchstone Energy Tucson Open	Omni Tucson National Golf Resort and Spa - Tucson, AZ	T5	K.J. Choi	-12	70	70	70	66	276	105375
2001	11/01/2001	14/01/2001	Touchstone Energy Tucson Open	Omni Tucson National Golf Resort and Spa - Tucson, AZ	T5	Cliff Kresge	-12	72	67	71	66	276	105375
2001	11/01/2001	14/01/2001	Touchstone Energy Tucson Open	Omni Tucson National Golf Resort and Spa - Tucson, AZ	T5	Greg Kraft	-12	74	65	69	68	276	105375
2001	11/01/2001	14/01/2001	Touchstone Energy Tucson Open	Omni Tucson National Golf Resort and Spa - Tucson, AZ	T5	Mark Wiebe	-12	69	67	66	74	276	105375
2001	11/01/2001	14/01/2001	Touchstone Energy Tucson Open	Omni Tucson National Golf Resort and Spa - Tucson, AZ	T9	Rich Beem	-11	70	69	71	67	277	72000

Tabel 2 Ruwe dataset

Op basis van deze gegevens wordt bijvoorbeeld voor **Garrett Willis** het volgende berekend:

540000

Earnings per stroke = ----- = 1978 \$ / slag

273

Deze berekening vormt de basis voor alle verdere statistische analyse en visualisatie in dit eindwerk.

#### § 4. GEBRUIKTE DATATYPEN EN TOOLS

De originele dataset werd aangeleverd als een tabgescheiden bestand (TSV), waarin de kolommen gescheiden zijn door tabtekens in plaats van komma's. Deze structuur vergemakkelijkt de leesbaarheid en import in verschillende analyse-omgevingen, zonder conflicten bij het gebruik van komma's in tekstvelden.

Bij het inlezen en analyseren van de data werden volgende datatypes geïdentificeerd:

KOLOMNAAM	VERWACHT DATATYPE
SEASON	Integer
START, END	Datum (tekstformaat, later geconverteerd)
TOURNAMENT	Tekst (string)
LOCATION	Tekst (string)
POSITION	Tekst (soms met 'T' in geval van Tie)
NAME	Tekst (string)
SCORE	Tekst of integer (relatief)
ROUND 1-4	Integer (stroke count)
TOTAL	Integer
EARNINGS	Decimaal of float
FEDEX_POINTS	Float / leeg (NULL)

Tabel 3 Datatypes ruwe dataset

Dataconversies zoals het casten van earnings naar numeriek en het parsen van datumvelden zijn uitgevoerd tijdens de data cleaning fase (zie hoofdstuk 2).

Voor de verwerking, opslag en visualisatie van de data werd gebruik gemaakt van volgende tools:

- **Lucidchart:** Voor het opmaken van de diverse datamodellen
- **SQL Server:** Voor de opslag en structureren van de gegevens in tabellen (dimensies en fact-tabel)
- **SSIS (SQL Server Integration Services):** Voor het opzetten van de ETL-stroom van bronbestand naar datawarehouse
- **Power BI:** Voor het berekenen van KPI's, datavisualisatie en rapportage
- **Python:** Voor data exploratie, controle op datakwaliteit
- **Excel** (in het beginstadium): Voor handmatige controle en snelle data-inspectie

## § 5. UITLEG GOLF TECHNISCH JARGON

De dataset bevat naast zuiver numerieke waarden ook termen en afkortingen die specifiek zijn voor de golfsport. Deze termen zijn belangrijk om correct te interpreteren bij het voorbereiden en analyseren van de data. Hieronder volgt een overzicht van de meest voorkomende codes die terug te vinden zijn in de kolom score of position:

AFKORTING	BETEKENIS	UITLEG
DQ	Disqualified	De speler werd gediskwalificeerd wegens een ernstige schending van de golfregels. Scores worden verwijderd en tellen niet mee voor andere competities binnen dezelfde ronde.
WD	Withdrawn	De speler trok zich vrijwillig terug uit het toernooi, vaak wegens blessure. De tot dan toe gespeelde holes blijven zichtbaar en kunnen gebruikt worden in andere (team)toernooien.
NS	No Show	De speler is niet opgedaagd voor het toernooi of de ronde. Er zijn geen scores geregistreerd.
NC	No Card	De speler heeft de ronde wel gespeeld, maar heeft ervoor gekozen geen scorekaart in te dienen. De scores tellen niet mee voor individuele rangschikkingen.
DNF	Did Not Finish	De speler is gestart aan het toernooi, maar heeft het niet afgewerkt. De reeds behaalde scores blijven zichtbaar, vergelijkbaar met WD.
CUT	Niet gehaald	De speler heeft <b>de cut</b> niet gehaald. In de meeste PGA Tour-toernooien wordt na twee rondes een grens gelegd waarbij enkel de beste spelers mogen deelnemen aan het weekendgedeelte van het toernooi. Spelers met "CUT" zijn dus uitgeschakeld na ronde 2.
T	Tied	De letter T in de position-kolom (vb. "T3") duidt aan dat meerdere spelers dezelfde plaats behaald hebben.

Tabel 4 *Golf-technisch jargon*

Deze statussen worden in de analysefase gebruikt om te bepalen welke records geldig zijn voor kwantitatieve evaluatie (zoals het berekenen van standing of earnings\_per\_stroke).

## 2. RELATIONEEL DATAMODEL

In dit hoofdstuk worden de conceptuele, logische en fysieke structuren van het gegevensmodel voor de PGA Tour-analyse toegelicht. Dit model werd opgebouwd op basis van de principes van relationele databases en het dimensioneel modelleren, en vormt de basis voor de verdere data-analyse in Power BI.

### § 1. CONCEPTUEEL DATAMODEL

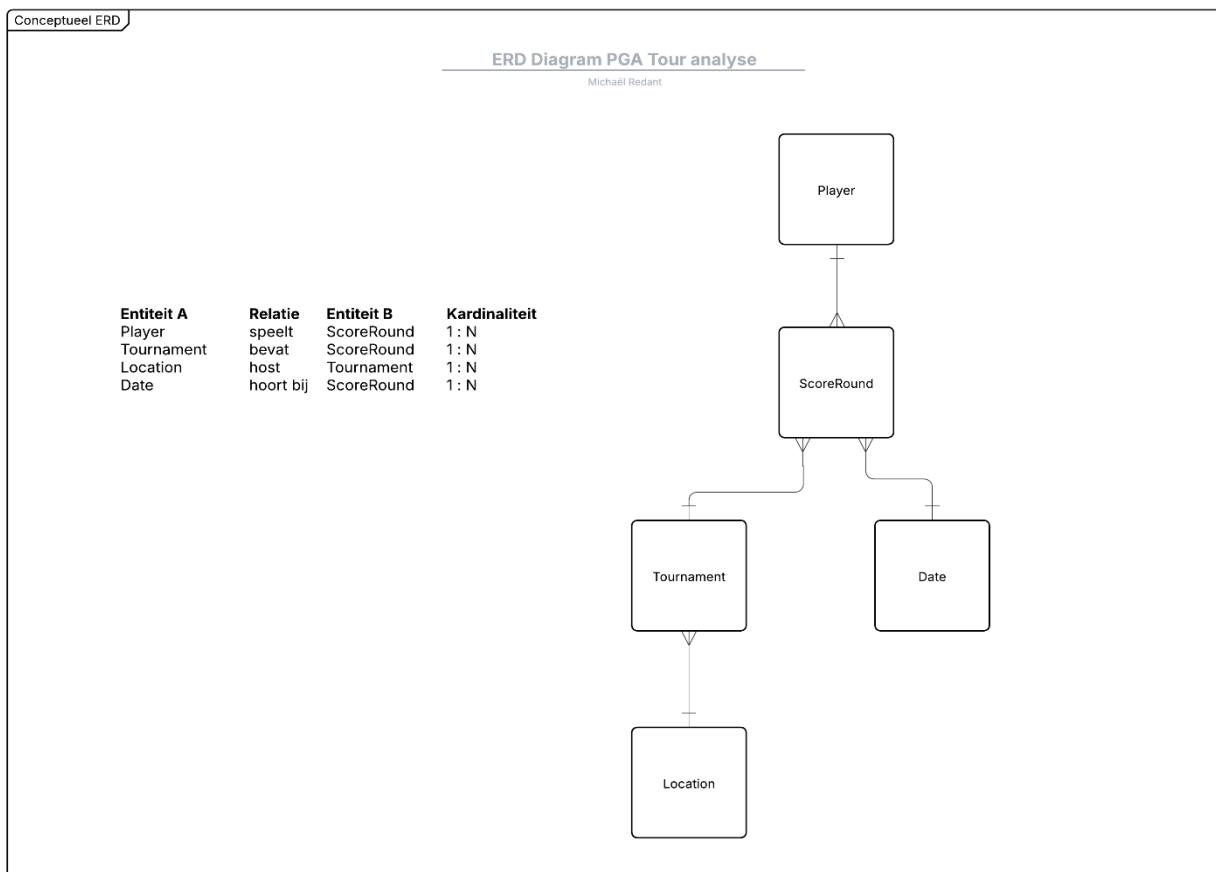
Het conceptueel datamodel (ERD – Entity Relationship Diagram) beschrijft op een abstract niveau de belangrijkste entiteiten in de PGA Tour-analyse en hoe ze met elkaar in relatie staan. De centrale entiteit is een Score Round: dit is een afzonderlijke gespeelde ronde van een speler in een specifiek toernooi, op een bepaalde locatie en datum.

Elke score wordt dus gekenmerkt door vier contextuele dimensies:

- Een **Player** speelt één of meerdere score rondes.
- Een **Tournament** bestaat uit meerdere score rondes.
- Een **Location** is de fysieke plaats waar een toernooi doorgaat.
- Een **Date** representeert de tijdsdimensie en wordt gekoppeld aan elke score ronde.

#### Relaties:

ENTITEIT A	RELATIE	ENTITEIT B	KARDINALITEIT
<b>PLAYER</b>	speelt	ScoreRound	1 : N
<b>TOURNAMENT</b>	bevat	ScoreRound	1 : N
<b>LOCATION</b>	host	Tournament	1 : N
<b>DATE</b>	hoort bij	ScoreRound	1 : N



Figuur 1 Conceptueel ERD diagram

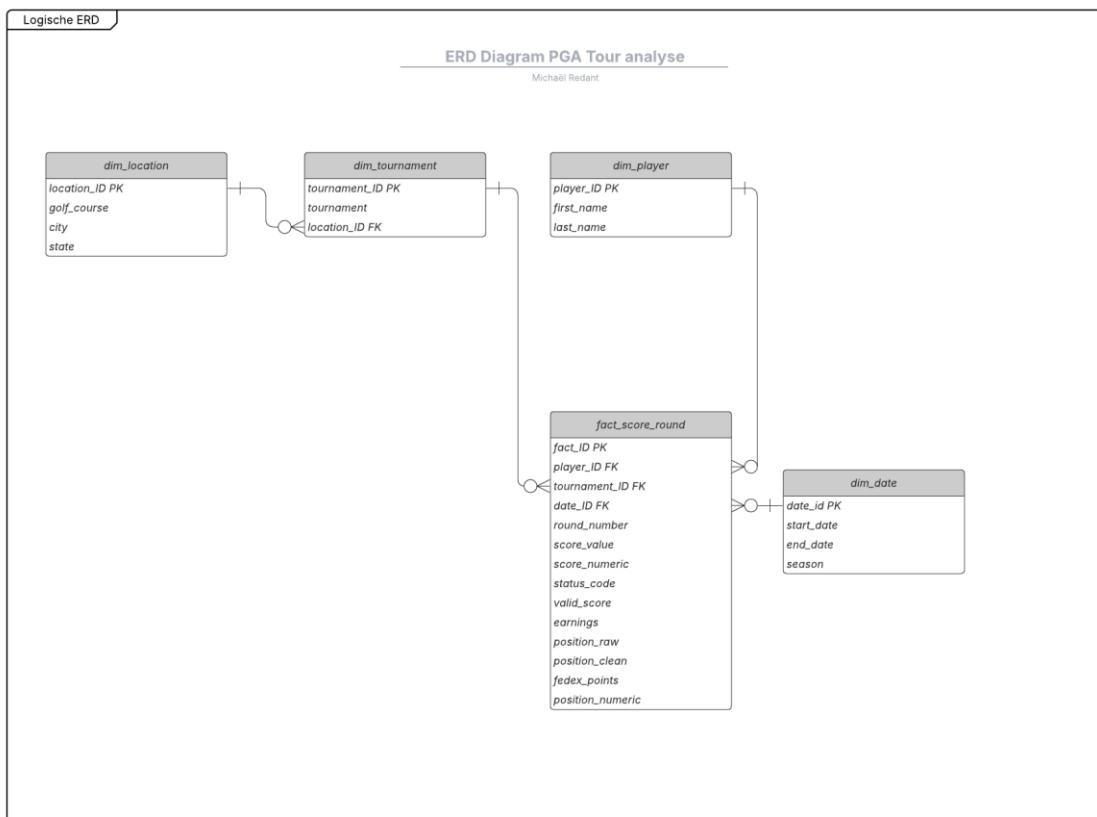
## § 2. LOGISCH DATAMODEL

In het logisch datamodel worden de conceptuele entiteiten vertaald naar tabellen met duidelijke relaties via primaire en vreemde sleutels. Dit model is ontworpen als een **sterstructuur**, waarbij de centrale feitentabel fact\_score\_round wordt omringd door vier dimensietabellen: dim\_player, dim\_tournament, dim\_location en dim\_date.

De feitentabel bevat alle meetbare gegevens (zoals scores, inkomsten, posities en FedEx-punten), terwijl de dimensietabellen de contextuele informatie bevatten die toelaten om te filteren, groeperen en analyseren.

### Belangrijke relaties:

- **Player → fact\_score\_round**  
(1:N)  
Elke speler kan meerdere rondes gespeeld hebben.  
Elke score is gelinkt aan exact één speler via player\_id.
- **Tournament → fact\_score\_round**  
(1:N)  
Een toernooi bevat meerdere scores (van meerdere spelers en ronden).  
Elke score hoort bij exact één toernooi via tournament\_id.
- **Date → fact\_score\_round**  
(1:N)  
Elke score wordt gekoppeld aan een unieke datumcombinatie van startdatum, einddatum en seizoen.  
Deze koppeling gebeurt via date\_id.
- **Location → fact\_score\_round**  
(1:N)  
Elke score vindt plaats op één specifieke locatie (golfclub, stad en staat).  
De koppeling verloopt rechtstreeks via location\_id.



Figuur 2 Logisch ERD diagram

### § 3. FYSIEK DATAMODEL

Het fysiek model vertaalt het logisch model naar implementatie in SQL Server. Hier worden de datatypes, constraints en sleutels gedefinieerd per tabel.

a. Tabel: dim\_player

Kolom	Type	PK	Not Null	Uniek	Opmerking
player_id	INT	X	X	X	Surrogate Key
first_name	NVARCHAR(50)		X		Voornaam
last_name	NVARCHAR(50)		X		Familienaam

Tabel 5 dim\_player

b. Tabel: dim\_location

Kolom	Type	PK	Not Null	Uniek	Opmerking
location_id	INT	X	X	X	Surrogate Key
golf_course	NVARCHAR(100)		X		Naam van de golfclub
city	NVARCHAR(50)		X		Stad
state	CHAR(2)		X		Staatcode (bv. CA, FL, AZ, ...)

Tabel 6 dim\_location

c. Tabel: dim\_tournament

Kolom	Type	PK	Not Null	Opmerking
tournament_id	INT	X	X	Surrogate Key
tournament	NVARCHAR(100)		X	Naam van het toernooi

Tabel 7 dim\_tournament

d. Tabel: dim\_date

Kolom	Type	PK	Not Null	Opmerking
date_id	INT	X	X	Surrogate Key
start_date	DATE		X	Start van het toernooi
end_date	DATE		X	Einddatum van het toernooi
season	INT		X	Kalenderjaar van het event

Tabel 8 dim\_date

**Toelichting:** season als INT: Omdat het jaartal van het seizoen een categorisch gegeven is (bv. 2023), en geen exacte tijdsstempel, is INT een logischere en efficiëntere keuze dan DATE. Dit bevordert ook groepering en filtering in BI-rapportages.

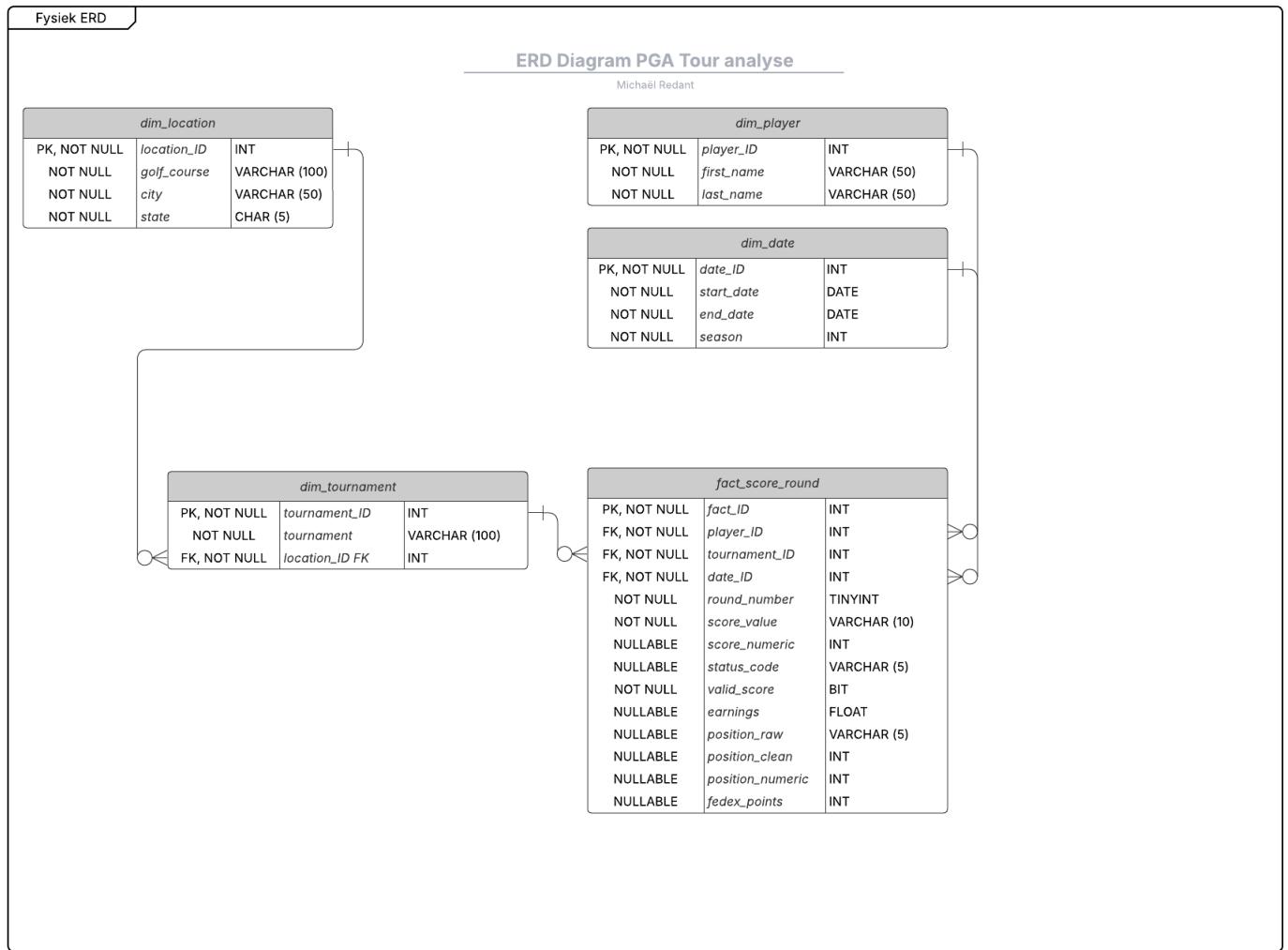
e. Tabel: fact\_score\_round

Kolom	Type	PK	FK	Not Null	Opmerking
fact_id	INT		X	X	Surrogaat ID per score-regel
player_id	INT		X	X	FK naar dim_player
tournament_id	INT		X	X	FK naar dim_tournament
location_id	INT		X	X	FK naar dim_location
date_id	INT		X	X	FK naar dim_date
round_number	TINYINT			X	Waarde 1 t.e.m. 4
score_value	NVARCHAR(10)			X	Originele inputwaarde (vb. 71, CUT, WD...)
score_numeric	INT				Enkel bij numerieke scores
status_code	VARCHAR(5)				CUT, WD, DQ...
valid_score	BIT			X	TRUE als de score bruikbaar is voor analyse
earnings	FLOAT				Verdiende USD (wordt meestal enkel toegekend bij ronde 4)
position	VARCHAR(5)				Originele ranking (vb. "T5", "1", "CUT")
position_numeric	INT				Genormaliseerde plaats (numeriek)
fedex_points	FLOAT				Enkel beschikbaar bij seizoen > 2010

Tabel 9 fact\_score\_round

### Toelichting:

- Scores worden enkel meegenomen in de analyse wanneer valid\_score = TRUE.
- Inkomsten (earnings) en posities (position\_numeric) worden in de meeste gevallen enkel ingevuld voor **ronde 4**, omdat dit de afsluitende ronde is waarop rankings en prijzen worden toegekend.
- fedex\_points is optioneel en afhankelijk van het seizoen; oudere jaren bevatten geen waarden.



Figuur 3 Fysiek ERD diagram

#### § 4. RELATIONEEL DATAMODEL (RDM)

Het relationeel datamodel beschrijft de structuur van het gegevensmodel aan de hand van tabellen, attributen en de onderlinge relaties tussen entiteiten. Elke tabel bevat een unieke primaire sleutel en – waar nodig – vreemde sleutels die verwijzen naar andere tabellen. Het model volgt de principes van een sterstructuur, waarbij de centrale feitentabel fact\_score\_round wordt ondersteund door vier dimensietabellen.

**Tabel: dim\_player**

Attribuut	Type	Sleutel	Constraints
player_id	INT	PK	NOT NULL, uniek
first_name	VARCHAR(50)		NOT NULL
last_name	VARCHAR(50)		NOT NULL

*Tabel 10 RDM - dim\_player*

**Tabel: dim\_location**

Attribuut	Type	Sleutel	Constraints
location_id	INT	PK	NOT NULL, uniek
golf_course	VARCHAR(100)		NOT NULL
city	VARCHAR(50)		NOT NULL
state	CHAR(2)		NOT NULL

*Tabel 11 RDM - dim\_location*

**Tabel: dim\_tournament**

Attribuut	Type	Sleutel	Constraints
tournament_id	INT	PK	NOT NULL, uniek
tournament	NVARCHAR(100)		NOT NULL

*Tabel 12 RDM - dim\_tournament*

**Tabel: dim\_date**

Attribuut	Type	Sleutel	Constraints
date_id	INT	PK	NOT NULL, uniek
start_date	DATE		NOT NULL
end_date	DATE		NOT NULL
season	SMALLINT		NOT NULL

*Tabel 13 RDM - dim\_date*

**Tabel: fact\_score\_round**

<b>Attribuut</b>	<b>Type</b>	<b>Sleutel</b>	<b>Constraints</b>
<i>fact_id</i>	INT	PK	NOT NULL, uniek
<i>player_id</i>	INT	FK	Verwijst naar dim_player(player_id)
<i>tournament_id</i>	INT	FK	Verwijst naar dim_tournament(tournament_id)
<i>location_id</i>	INT	FK	Verwijst naar dim_location(location_id)
<i>date_id</i>	INT	FK	Verwijst naar dim_date(date_id)
<i>round_number</i>	TINYINT		NOT NULL (1-4)
<i>score_value</i>	NVARCHAR(10)		NOT NULL
<i>score_numeric</i>	INT		NULLABLE – indien waarde numeriek is
<i>status_code</i>	VARCHAR(5)		NULLABLE – CUT, WD, DQ, ...
<i>valid_score</i>	BIT		NOT NULL – enkel TRUE voor bruikbare scores
<i>earnings</i>	FLOAT		NULLABLE
<i>position</i>	VARCHAR(5)		NULLABLE
<i>position_numeric</i>	INT		NULLABLE
<i>fedex_points</i>	FLOAT		NULLABLE – alleen beschikbaar bij recentere jaren

Tabel 14 RDM - fact\_score\_round

**Toelichting:**

- De relaties tussen de tabellen worden gewaarborgd via foreign key constraints.
- Alle primaire sleutels zijn surrogate keys die autonummerend gegenereerd worden.
- De centrale feitentabel fact\_score\_round bevat alle meetbare gegevens en vormt het hart van het stermodel.

In tegenstelling tot eerdere gemaakte modellen is de locatie rechtstreeks gekoppeld aan de feitentabel, zonder tussenkomst van de toernooientiteit. Dit laat directe analyses per golfclub toe, zoals gemiddelde score per ronde, aantal overwinningen per club, en laagste gescoorde rondes per locatie.

### 3. DATA CLEANSING EN VOORBEREIDING ETL

In dit hoofdstuk wordt het proces toegelicht dat werd gevuld om de ruwe data uit de Kaggle-dataset om te zetten in een analyseerbare vorm, conform de principes van ETL (Extract – Transform – Load). Hierbij werd gebruik gemaakt van Microsoft SQL Server als opslag- en transformatielaaag.

#### § 1. IMPORTEREN IN EEN STAGING-OMGEVING

De originele dataset werd als TSV-bestand (tab-separated values) geïmporteerd in een staging-tabel binnen een SQL Server-database. De stagingtabel had initieel de naam dbo.pga\_results\_2001-2024.

Aangezien het gebruik van streepjes (-) in objectnamen af te raden is in SQL Server (omwille van foutgevoeligheid bij queries), werd deze hernoemd volgens best practice conventies. Dit gebeurde met behulp van de sp\_rename systeemprocedure:

```
EXEC SP_RENAME 'DBO.PGA_RESULTS_2001-2024', 'PGA_RESULTS_RAW';
```

De tabel pga\_results\_raw fungeerde nadien als bron voor verdere transformaties en opschoning.

#### § 2. DUPLICATIE NAAR EEN WERKVERSIE

Om een scheiding te behouden tussen de originele ruwe data en de bewerkte versie, werd ervoor gekozen om de oorspronkelijke stagingtabel (pga\_results\_raw) onaangetast te laten. Dit is een goede praktijk binnen datawarehousing en voorkomt dat de ruwe brongegevens per ongeluk worden gewijzigd.

Daarom werd een volledige duplicatie uitgevoerd met:

```
SELECT *
INTO STG_PGA_RESULTS_CLEAN
FROM DBO.PGA_RESULTS_RAW;
```

De nieuwe tabel stg\_pga\_results\_clean vormt de werkversie waarop alle verdere data cleaning en transformaties zullen plaatsvinden. In deze fase worden volgende operaties voorzien:

- Het splitsen van samengestelde velden (zoals locatie → golfbaan, stad, staat)
- Validatie en filtering van records met ongeldige waarden (bv. total IS NULL, earnings = 0, of status\_code ≠ geldig)
- Berekening van nieuwe kolommen, zoals earnings\_per\_stroke
- Eventuele conversie van stringwaarden naar numeriek (score, position)

Door deze aanpak wordt de traceerbaarheid van de dataverwerking gewaarborgd, en kan men te allen tijde terugvallen op de originele dataset voor controle of hergebruik.

### § 3. SPLITSINGEN

In de oorspronkelijke dataset is de naam van elke speler opgenomen in één veld (name), waarbij voornaam en familienaam (eventueel aangevuld met tussenvoegsels) samen als één string voorkomen. Om deze informatie correct te kunnen normaliseren en overdragen naar de Player-dimensietabel, heb ik gekozen om de naam op te splitsen in first\_name en last\_name.

#### a. *Splitslogica namen*

De scheiding werd bepaald aan de hand van de **eerste spatie** in het veld name. Het deel vóór de eerste spatie werd toegewezen aan first\_name, terwijl het resterende deel (inclusief eventuele extra spaties of componenten) werd toegewezen aan last\_name.

Deze aanpak houdt correct rekening met samengestelde achternamen en tussenvoegsels, zoals:

"Justin Thomas Jr." → Justin / Thomas Jr.

"Lee van der Walt" → Lee / van der Walt

Dit werd uitgevoerd met:

```
UPDATE STG_PGA_RESULTS_CLEAN
SET
    FIRST_NAME = LEFT(NAME, CHARINDEX(' ', NAME + ' ') - 1),
    LAST_NAME = LTRIM(RIGHT(NAME, LEN(NAME) - CHARINDEX(' ', NAME + ' ')));
```

De volgende validatiequery:

```
SELECT TOP 20 name, first_name, last_name
FROM stg_pga_results_clean;
```

Geeft dan volgende resultaten:

NAME	FIRST_NAME	LAST_NAME
TIGER WOODS	Tiger	Woods
JUSTIN THOMAS JR.	Justin	Thomas Jr.
LEE VAN DER WALT	Lee	van der Walt

Tabel 15 resultaten splitsing query SSMS

#### b. *Splitslogica locatie*

De kolom location in de oorspronkelijke dataset bevat gecombineerde informatie over de naam van het golfcomplex, de stad en de staat, in het volgende formaat:

[golf\_course] - [city], [state]

Om deze informatie correct te kunnen normaliseren en koppelen aan de Location-dimensietabel, werd de data opgesplitst in drie afzonderlijke kolommen: golf\_course, city en state.

De structuur werd aangepast door extra kolommen toe te voegen aan de werkversie van de stagingtabel:

```
ALTER TABLE STG_PGA_RESULTS_CLEAN
ADD GOLF.Course NVARCHAR(100),
    CITY NVARCHAR(50),
    STATE CHAR(2);
```

Om te vermijden dat foutief gestructureerde waarden zouden leiden tot foutmeldingen, werd een WHERE-clausule toegevoegd die enkel werkt op waarden die het verwachte patroon volgen:

```

UPDATE STG_PGA_RESULTS_CLEAN
SET
    GOLF.Course = LTRIM(RTRIM(LEFT(LOCATION, CHARINDEX('-', LOCATION) - 1))),
    CITY      = LTRIM(RTRIM(SUBSTRING(LOCATION, CHARINDEX('-', LOCATION) + 1,
CHARINDEX(',', LOCATION) - CHARINDEX('-', LOCATION) - 1))),
    STATE     = RIGHT(LOCATION, 2)
WHERE
    LOCATION IS NOT NULL AND
    CHARINDEX('-', LOCATION) > 0 AND
    CHARINDEX(',', LOCATION) > CHARINDEX('-', LOCATION);

```

Records die niet aan het verwachte patroon voldeden — zoals internationale locaties of verkeerd ingevoerde waarden — werden opgevangen met standaardwaarden om de analyse niet te verstoren:

```

UPDATE STG_PGA_RESULTS_CLEAN
SET
    GOLF.Course = 'ONBEKEND',
    CITY      = 'ONBEKEND',
    STATE     = '??'
WHERE
    LOCATION IS NULL
    OR CHARINDEX('-', LOCATION) = 0
    OR CHARINDEX(',', LOCATION) <= CHARINDEX('-', LOCATION);

```

Tijdens de verwerking van de kolom location zijn enkel waarden met het patroon [golf\_course] - [city], [state] opgesplitst. Afwijkende of foutief gestructureerde locaties zijn opgevangen met de standaardwaarden 'Onbekend' en '??', zodat de gegevens toch bruikbaar blijven in het datamodel. Deze records kunnen apart gemonitord worden.

Dit geeft dan volgende resultaten:

LOCATION	GOLF.COURSE	CITY	STATE
HAZELTINE NATIONAL GOLF CLUB - CHASKA, MN	Hazeltine National Golf Club	Chaska	MN
OMNI TUCSON NATIONAL GOLF RESORT AND SPA - TUCSON, AZ	Omni Tucson National Golf Resort and Spa	Tucson	AZ
ONBEKEND LOCATIE ZONDER STREEP	Onbekend	Onbekend	??

Tabel 16 resultaten splitsing locatie SSMS

#### § 4. TRANSFORMATIE RONDE GEGEVENS

In de oorspronkelijke dataset werden de scores van de vier speeldagen opgeslagen in vier afzonderlijke kolommen: `round1`, `round2`, `round3` en `round4`. Hoewel deze brede tabelstructuur begrijpelijk is voor ruwe opslag, is ze minder geschikt voor analytische doeleinden.

Voor een performante en flexibele analyse – zoals het berekenen van gemiddelden per ronde, het opvolgen van prestaties in volgorde of het bouwen van dynamische visualisaties in Power BI – is een event-georiënteerde structuur (ook wel "long format") veel geschikter.

Daarom werd de dataset tijdens de ETL-fase getransformeerd naar een genormaliseerde vorm, waarbij elke combinatie van speler en toernooi wordt opgesplitst in vier afzonderlijke records: één per gespeelde ronde. Elk van deze rijen bevat dan expliciet het rondenummer (1 t.e.m. 4) en de bijbehorende score, status, en andere **kenmerken**.

Deze herstructurering vormt de kern van de analyse, en maakte het mogelijk om:

- scores per ronde te analyseren (bv. welke ronde was het moeilijkst),
- spelers onderling te vergelijken op specifieke speeldagen,
- en correcte KPI's op te bouwen zoals *earnings per stroke* of *cut-succesratio*.

##### a. *Creatie van de tussenliggende tabel: round\_score\_fact*

Na de transformatie van brede rondedata naar een event-georiënteerde structuur, werd een tussenliggende facttabel aangemaakt: `round_score_fact`. Deze tabel vormt de eerste genormaliseerde versie van de oorspronkelijke dataset en bevat alle noodzakelijke informatie voor verdere dimensionele modellering.

De tabel `round_score_fact` combineert contextuele gegevens, ronde-informatie, en afgeleide velden in één overzichtelijke structuur. Hiermee werd de brug gelegd tussen de ruwe inputgegevens en de uiteindelijke `fact_score_round`-tabel in het stermodel.

Inhoud van de tabel:

- Contextuele gegevens: seizoen, start- en einddatum, toernooi, golfclub, locatie (stad en staat), naam van de speler
- Ronde-informatie: `round_number`, oorspronkelijke scorewaarde (`score_value`)
- Afgeleide velden: numerieke score (`score_numeric`), statuscode (CUT, WD, DQ...), validatiemarkering (`valid_score`)
- Financiële gegevens: `earnings`, gekopieerd per ronde voor analysegemak

```
CREATE TABLE ROUND_SCORE_FACT (
    SEASON SMALLINT,
```

```

START_DATE DATE,
END_DATE DATE,
TOURNAMENT NVARCHAR(100),
GOLF_COURSE NVARCHAR(100),
CITY NVARCHAR(50),
STATE CHAR(2),
POSITION NVARCHAR(50),
NAME NVARCHAR(100),
FIRST_NAME NVARCHAR(50),
LAST_NAME NVARCHAR(50),
ROUND_NUMBER TINYINT,
SCORE_VALUE NVARCHAR(10),
SCORE_NUMERIC INT,
STATUS_CODE VARCHAR(4),
VALID_SCORE BIT,
EARNINGS FLOAT

```

**Opmerking:** In deze fase werd nog geen gebruik gemaakt van surrogate keys. De focus lag op het herstructureren en verrijken van de data. In latere stappen werd deze tabel opgeladen in het dimensioneel model via SQL Server Integration Services (SSIS), waarbij de correcte sleutelrelaties werden gelegd.

#### b. *INSERT-query met validatie en conversieoplossing*

Tijdens het unpivotproces werden de scores uit de stagingtabel `stg_pga_results_clean` (kolommen `round1` t.e.m. `round4`) geladen in de nieuwe feitentabel `round_score_fact`. Elke combinatie van speler en toernooi resulteert hierbij in **vier afzonderlijke rijen**: één per gespeelde ronde. Deze transformatie is typisch binnen dimensionele modellen en vormt de basis voor verdere analyse per ronde.

#### Fout bij initiële laadtentatief

Bij een eerste poging om de gegevens via een `UNION ALL`-constructie te laden in `round_score_fact`, trad volgende fout op:

```

CONVERSION FAILED WHEN CONVERTING THE VARCHAR VALUE 'CUT' TO DATA
TYPE TINYINT.

```

Deze fout werd veroorzaakt door het datatype van de kolommen `round1-round4` in de stagingtabel, die als `TINYINT` waren gedefinieerd. Hierdoor konden tekstuele waarden zoals 'CUT', 'DQ' of 'WD' niet correct geïnterpreteerd worden, wat leidde tot typeconversiefouten tijdens het laden.

#### Strategische afweging: staging herstructureren of conversie inladen?

Er werden twee oplossingen overwogen:

### **Optie 1: Structuur van de stagingtabel aanpassen**

De datatype van round1 tot round4 wijzigen van TINYINT naar NVARCHAR(10). Hierdoor zouden alle tekstuele en numerieke waarden zonder conversiefout in dezelfde kolom passen.

#### **Voordelen:**

- Geen typeconversie nodig tijdens het laden.
- Tekstuele waarden worden rechtstreeks ondersteund.

#### **NadeLEN:**

- Het stagingmodel wijkt af van best practices: numerieke waarden worden als tekst behandeld.
- SQL-bewerkingen zoals AVG(round1) of SUM(round2) worden moeilijker of foutgevoeliger.
- De stagingtabel wordt “zachter” en minder robuust, wat ze minder geschikt maakt voor hergebruik in andere contexten of integratie in SSAS-modellen.

### **Optie 2: INSERT-query herwerken met expliciete conversie (gekozen optie)**

De tweede en gekozen oplossing bestond erin om de bestaande structuur van stg\_pga\_results\_clean ongemoeid te laten en de conversielogica volledig onder controle te houden in de INSERT INTO round\_score\_fact. Hierbij werden alle rondewaarden per rij expliciet gecast naar NVARCHAR(10), en werd gebruik gemaakt van:

TRY\_CAST() om scores als geheel getallen op te slaan in score\_numeric  
CASE-logica om status\_code en valid\_score correct af te leiden

#### **Voordelen:**

- De oorspronkelijke staging blijft optimaal getypeerd (conform SSIS-importstandaard).
- Je behoudt volledige controle over de unpivotlogica.
- De conversieregels zijn expliciet en transparant, wat de foutgevoeligheid vermindert.
- Je toont als ontwikkelaar of data-analist aan dat je bewust omgaat met datatype-afleidingen en data-integriteit.

#### **Nadeel:**

- De query wordt langer, omdat de logica vier keer moet worden geschreven (voor elke ronde afzonderlijk). Dit werd echter beschouwd als acceptabel in functie van stabiliteit en herhaalbaarheid.

### **Conclusie**

De gekozen oplossing — expliciete conversielogica binnen de INSERT-query — past binnen best practices voor robuuste ETL-processen. Ze garandeert:

- typeveiligheid in de staginglaag;
- controleerbare en reproduceerbare data-unpivoting;
- compatibiliteit met BI-tools zoals Power BI of SSAS.

Deze aanpak toont een bewuste afweging tussen technische nauwkeurigheid en operationele betrouwbaarheid, en sluit perfect aan bij de vereisten van een professioneel datawarehouse-ontwerp.

### **§ 5. VALIDATIE EN CORRECTIE VAN STATUS\_CODE EN VALID\_SCORE**

Na het laden van de round\_score\_fact-tabel bleek bij inspectie dat alle waarden voor valid\_score ontbraken (NULL) en dat status\_code niet altijd correct was afgeleid. Dit werd opgelost via een correctieve update.

*a. Corrigeren van status\_code*

De statuscode (DQ, CUT, etc.) werd afgeleid uit score\_value indien dit een bekende tekstuele waarde bevatte:

```
UPDATE DBO.ROUND_SCORE_FACT
SET STATUS_CODE =
CASE
    WHEN SCORE_VALUE IN ('DQ', 'WD', 'NS', 'NC', 'DNF', 'CUT') THEN SCORE_VALUE
    ELSE NULL
END
WHERE SCORE_VALUE IS NOT NULL;
```

*b. Corrigeren van valid\_score*

Aan de hand van de status\_code en score\_value werd bepaald of een score geldig is voor analyse:

```
UPDATE DBO.ROUND_SCORE_FACT
SET VALID_SCORE =
CASE
    WHEN STATUS_CODE IN ('DQ', 'NS', 'NC') THEN 0
    WHEN STATUS_CODE IN ('WD', 'DNF', 'CUT') THEN 1
    WHEN TRY_CAST(SCORE_VALUE AS INT) IS NOT NULL THEN 1
    ELSE 0
END;
```

*c. Validatiequeries*

Om de correctheid van deze transformaties te controleren, werden diverse query's uitgevoerd:  
Aantal geldige vs. ongeldige scores:

```
SELECT VALID_SCORE, COUNT(*) AS AANTAL
FROM ROUND_SCORE_FACT
GROUP BY VALID_SCORE;
```

Verdeling statuscodes:

```
SELECT STATUS_CODE, COUNT(*) AS AANTAL
FROM ROUND_SCORE_FACT
WHERE VALID_SCORE = 0
GROUP BY STATUS_CODE
ORDER BY AANTAL DESC;
```

Inspectie van ongeldige scores:

```
SELECT *
FROM ROUND_SCORE_FACT
WHERE VALID_SCORE = 0;
```

Door te kiezen voor een aparte feitentabel round\_score\_fact, werd het datamodel vereenvoudigd, schaalbaar gemaakt en klaar voor gebruik in Power BI en SSIS. Fouten tijdens het unpivotten werden correct opgespoord en opgelost via expliciete typecasting en validatieregels, wat aansluit bij de best practices in data engineering.

#### *d. Latere SQL aanpassingen en optimalisaties*

Naast de correctieve updates op status\_code en valid\_score, werden tijdens de SQL-fase nog enkele gerichte aanpassingen doorgevoerd om de datakwaliteit en analyseerbaarheid te verbeteren.

#### **Toevoeging van position\_numeric (genormaliseerde positie)**

De oorspronkelijke position (vb. "T3", "CUT", "1", "WD") werd uitgebreid met een genormaliseerde numerieke variant, zodat posities correct geanalyseerd en gesorteerd kunnen worden:

```
CASE
    WHEN POSITION LIKE 'T%' THEN TRY_CAST(SUBSTRING(POSITION, 2,
LEN(POSITION)) AS INT)
    WHEN TRY_CAST(POSITION AS INT) IS NOT NULL THEN CAST(POSITION
AS INT)
    ELSE -1 -- VOOR CUT, WD, DQ, ENZ.
END AS POSITION_NUMERIC
```

Deze logica werd later geïntegreerd tijdens SSIS-verwerking via een Derived Column.

#### **Toevoeging van is\_tie-indicator**

In een eerdere versie van het datamodel werd het veld is\_tie toegevoegd om aan te geven of een speler een gedeelde plaats behaalde (bijv. "T5"). Dit veld werd afgeleid van het originele position-veld, met behulp van volgende logica:

```
CASE
    WHEN POSITION LIKE 'T%' THEN 1
    ELSE 0
END AS IS_TIE
```

**Opmerking:** In een latere fase werd dit attribuut geschrapt omdat het geen toegevoegde analytische waarde meer had binnen de rapportagestructuur van Power BI.

### Toevoeging van fedex\_points

In een latere fase werd ook de kolom fedex\_points geïntegreerd in de staging en de feitentabel. Deze werd initieel enkel bij de laatste ronde (round\_number = 4) ingevuld in de ruwe data, maar werd via duplicatie over alle rondes verspreid voor consistente analyse:

```
FEDEX_POINTS FLOAT NULL
```

Dit liet toe om **seizoensanalyses** te maken in Power BI en om spelers op basis van puntentotalen te vergelijken over de jaren heen.

### Herindeling van fact\_id als surrogaat sleutel

De finale feitentabel fact\_score\_round kreeg een auto-increment veld fact\_id als primaire sleutel om de tabel compatibel te maken met SSIS-dataloads en latere integratie in Power BI:

```
CREATE TABLE DBO.FACT_SCORE_ROUND (
    FACT_ID INT IDENTITY(1,1) PRIMARY KEY,
    PLAYER_ID INT NOT NULL,
    TOURNAMENT_ID INT NOT NULL,
    DATE_ID INT NOT NULL,
    LOCATION_ID INT NOT NULL,
    ROUND_NUMBER TINYINT NOT NULL,
    SCORE_VALUE NVARCHAR(10) NOT NULL,
    SCORE_NUMERIC INT NULL,
    STATUS_CODE VARCHAR(5) NULL,
    VALID_SCORE BIT NOT NULL,
    EARNINGS FLOAT NULL,
    FEDEX_POINTS FLOAT NULL,
    POSITION_RAW VARCHAR(5) NULL,
    POSITION_NUMERIC INT NULL,
    IS_TIE BIT NULL
);
```

### Toelichting:

- **fact\_id** is de primaire sleutel en wordt automatisch gegenereerd. Dit maakt de tabel SSIS-vriendelijk én uniek indexeerbaar.
- **player\_id, tournament\_id, date\_id, location\_id** zijn foreign keys naar de respectieve dimensietafballen.
- **fedex\_points** werd toegevoegd voor analyses op seizoensniveau en rankingvergelijking.
- **position\_numeric** werd afgeleid om sorteringen en top X-analyses correct uit te voeren.
- **is\_tie** laat je toe om onderscheid te maken tussen gedeelde en exclusieve posities.

## 4. DIMENSIONEEL MODEL

In dit hoofdstuk wordt de opbouw van het uiteindelijke datawarehouse toegelicht, volgens de principes van het dimensioneel modelleren zoals beschreven door Ralph Kimball. Het doel is om de opgeschoonde en getransformeerde data onder te brengen in een ster model, bestaande uit één feitentabel en meerdere dimensietabellen.

### § 1. VAN STAGING NAAR DIMENSIONEEL MODEL

Tijdens de ETL-flow werden verschillende tabellen gebruikt, elk met een specifieke rol:

TABEL	FUNCTIE	STATUS
<b>PGA_RESULTS_RAW</b>	Oorspronkelijke import van de ruwe flatfile	Behouden als bronarchief
<b>STG_PGA_RESULTS_CLEAN</b>	Opschoning en splitsing van velden (bv. naam, locatie)	Behouden als staging
<b>ROUND_SCORE_FACT</b>	Unpivotte scoretabel, 1 rij per ronde	Tussenstap, tijdelijk
<b>FACT_SCORE_ROUND</b>	Eindfacttabel met sleutels naar dimensies	Definitieve feitentabel
<b>DIM_PLAYER</b>	Dimensietabel voor spelers	Definitief
<b>DIM_TOURNAMENT</b>	Dimensietabel voor toernooien	Definitief
<b>DIM_DATE</b>	Dimensietabel voor datum en seizoen	Definitief
<b>DIM_LOCATION</b>	Dimensietabel voor golfbanen en locaties	Definitief

Tabel 17 Rollen van de tabellen ETL

De tabel round\_score\_fact werd aangemaakt als tussenstap om de brede structuur van de oorspronkelijke data om te zetten naar een fijnmazig, rij gebaseerd formaat (één rij per gespeelde ronde). Hoewel deze structuur geschikt was voor validatie en exploratie, bevatte ze nog tekstuele gegevens zoals voornamen en toernooienamen, en geen verwijzingen naar dimensionele sleutels.

Daarom werd ervoor gekozen om de eindfacttabel fact\_score\_round op te bouwen via JOINs op de juiste dimensionele tabellen. Deze tabel bevat uitsluitend sleutels naar dimensies, wat in lijn is met de Kimball-methodiek en zorgt voor een gestandaardiseerde, schaalbare rapportagestructuur.

## § 2. DIMENSIONELE TABELLEN

De volgende dimensietabellen werden aangemaakt:

- **dim\_player:** bevat unieke spelers op basis van voornaam en familienaam.

```
CREATE TABLE DIM_PLAYER (
    PLAYER_ID INT IDENTITY(1,1) PRIMARY KEY,
    FIRST_NAME NVARCHAR(50),
    LAST_NAME NVARCHAR(50)
);
```

*Spelers opladen vanuit round\_score\_fact*

```
INSERT INTO DIM_PLAYER (FIRST_NAME, LAST_NAME)
SELECT DISTINCT FIRST_NAME, LAST_NAME
FROM ROUND_SCORE_FACT
WHERE FIRST_NAME IS NOT NULL AND LAST_NAME IS NOT NULL;
```

- **dim\_tournament:** bevat de unieke namen van de toernooien.

```
CREATE TABLE DIM_TOURNAMENT (
    TOURNAMENT_ID INT IDENTITY(1,1) PRIMARY KEY,
    TOURNAMENT NVARCHAR(100)
);
```

*Gegevens opladen:*

```
INSERT INTO DIM_TOURNAMENT (TOURNAMENT)
SELECT DISTINCT TOURNAMENT
FROM ROUND_SCORE_FACT
WHERE TOURNAMENT IS NOT NULL;
```

- **dim\_date:** bevat unieke combinaties van start- en einddatum en het seizoen.

```
CREATE TABLE DIM_DATE (
    DATE_ID INT IDENTITY(1,1) PRIMARY KEY,
    START_DATE DATE,
    END_DATE DATE,
    SEASON SMALLINT
);
```

*Data opladen:*

```
INSERT INTO DIM_DATE (START_DATE, END_DATE, SEASON)
SELECT DISTINCT START_DATE, END_DATE, SEASON
FROM ROUND_SCORE_FACT
WHERE START_DATE IS NOT NULL;
```

- **dim\_location:** bevat unieke combinaties van golfbaan, stad en staat.

```
CREATE TABLE DIM_LOCATION (
    LOCATION_ID INT IDENTITY(1,1) PRIMARY KEY,
    GOLF_COURSE NVARCHAR(100),
    CITY NVARCHAR(50),
    STATE CHAR(2)
);
```

*Opladen van gegevens:*

```
INSERT INTO DIM_LOCATION (GOLF_COURSE, CITY, STATE)
SELECT DISTINCT GOLF_COURSE, CITY, STATE
FROM ROUND_SCORE_FACT
WHERE GOLF_COURSE IS NOT NULL;
```

Elke tabel werd voorzien van een surrogaat sleutel en werd geladen via een SELECT DISTINCT vanuit round\_score\_fact.

TABEL	SLEUTELS	OPMERKING
DIM_PLAYER	player_id	unieke spelers
DIM_TOURNAMENT	tournament_id	unieke toernooien
DIM_DATE	date_id	unieke datumbereiken
DIM_LOCATION	location_id	unieke locaties

Tabel 18 Dimension tabellen overzicht

## 5. SSIS ETL IN VISUAL STUDIO

In dit hoofdstuk wordt de implementatie van de ETL-stroom in SSIS (SQL Server Integration Services) toegelicht, waarbij de voorbereide gegevens uit round\_score\_fact worden gekoppeld aan de dimensionele sleutels en ingeladen in de finale feitentabel fact\_score\_round. Deze stap vormt de kern van het automatiseringsproces en maakt het mogelijk om een schaalbare, reproduceerbare en foutbestendige datastream op te zetten.

### § 1. DOELSTELLING

De opdracht bestond erin om de dataset uit round\_score\_fact te verrijken met surrogaat sleutels (foreign keys) vanuit de tabellen:

- dim\_player
- dim\_tournament
- dim\_date
- dim\_location

De verwerking verloopt volledig via Visual Studio en SSIS, en maakt gebruik van een Sequence Container waarin het proces opgesplitst is in twee hoofdonderdelen:

- Leegmaken van de doelentabel (fact\_score\_round)
- Laadproces van verrijkte data met lookuptransformaties

### § 2. OPBOUW VAN DE SSIS-STRUCTUUR

De datastream werd als volgt opgebouwd:

<i>Component</i>	<i>Omschrijving</i>
<b><i>SRC_RoundScoreFact</i></b>	OLE DB Source – haalt records op uit stagingtafel round_score_fact
<b><i>DC_Player</i></b>	Derived Column – trims first_name en last_name om lookup-problemen te vermijden
<b><i>LKP_Player</i></b>	Lookup – koppelt first_name + last_name aan player_id in dim_player
<b><i>DC_Tournament</i></b>	Derived Column – trims tournament voor betere matching
<b><i>LKP_Tournament</i></b>	Lookup – koppelt tournament aan tournament_id in dim_tournament
<b><i>LKP_Date</i></b>	Lookup – op basis van start_date, end_date en season naar dim_date
<b><i>LKP_Location</i></b>	Lookup – op basis van golf_course, city, state naar dim_location
<b><i>Sort</i></b>	Sortering vóór de destination (noodzakelijk voor parallelle loads)
<b><i>DST_FactScoreRound</i></b>	OLE DB Destination – schrijft de verrijkte records naar fact_score_round

Tabel 19 Datastream ETL

### § 3. FOUTAFHANDELING VIA REDIRECT ROWS

Elke Lookup Task is voorzien van foutafhandeling:

Indien een record geen match vindt in een dimensietafel, wordt deze via de “Redirect Row”-optie doorgestuurd naar een Flat File Destination (error-logbestand), per dimensie.

Zo worden de volgende foutbestanden gegenereerd:

- Player\_error\_file.csv
- Tournament\_error\_file.csv
- Location\_error\_file.csv

### § 4. UITDAGINGEN EN OPLOSSINGEN ONDERWEG

Tijdens de ontwikkeling van het SSIS-pakket en de uitrol in Power BI werden verschillende problemen gedetecteerd, die telkens werden opgelost met gerichte acties. Hieronder volgt een overzicht van typische ETL-struggles en Power BI-strategische keuzes die gemaakt zijn tijdens het project:

#### a. SSIS Fase – ETL-uitdagingen

<b>PROBLEEM</b>	<b>OORZAAK</b>	<b>OPLOSSING</b>
<i>Lookup Tournament faalde</i>	Lege waarden, spaties of casegevoeligheid	Toevoegen van Derived Column met TRIM(tournament)
<i>Lookup zonder match stopte de flow</i>	Standaardinstelling = "Fail Component"	Aangepast naar "Redirect row" + foutlog-bestanden
<i>Matching waarden werden niet herkend</i>	Verborgen tekens / encodingproblemen	Analyse van foutbestanden → ontbrekende waarden aan dimensie toegevoegd
<i>Dubbele spelers werden niet herkend in dim_player</i>	first_name en last_name hadden afwijkende casing of witruimte	LOWER(TRIM(...)) gebruikt bij insert én lookup
<i>Fout bij insert in destination (null keys)</i>	Missende foreign keys door lookup-failure	Volledige rij redirecten naar error-log voor debug
<i>Insert werkte lokaal maar niet op staging</i>	Culture-instelling of type mismatch op SQL Server	DT_WSTR explicet gedefinieerd bij mapping
<i>SSIS buffer overflow</i>	Te veel records per batch	DefaultBufferMaxRows verhoogd naar 50.000

Tabel 20 ETL uitdagingen in SSIS

b. Power BI Fase – Visualisatie & DAX uitdagingen

PROBLEEM	OORZAAK	OPLOSSING
KPI-vergelijking tussen twee spelers werkte niet  Earnings verschillen tonen -100%	Contextfilter conflict in DAX tussen Selected Player en Compare Player	Measures herschreven met SELECTEDVALUE() + aparte slicers
Sommige filters gaven geen data terug	Eén van de twee measures retourneerde blanco  valid_score = FALSE records werden meegeteld in visuals	F(ISBLANK(...), 0, ...) toegevoegd + fallback-waarden  Filter toegevoegd op valid_score = TRUE in alle rapportpagina's
Kaart visualisatie toonde foute locaties  Seizoen-overzicht onvolledig  Geen matches in matrix	Location-waarden niet correct gegeocoded  Niet alle jaren hadden 4 ronden per speler  Beide slicers (player & year) filterden zonder crossfilter	Locatie-data herzien, city + state opnieuw opgeschoond in SQL  Visuals aangepast naar % completeness per speler en jaar  Measure herschreven met REMOVEFILTERS() en FILTER(ALL(...))

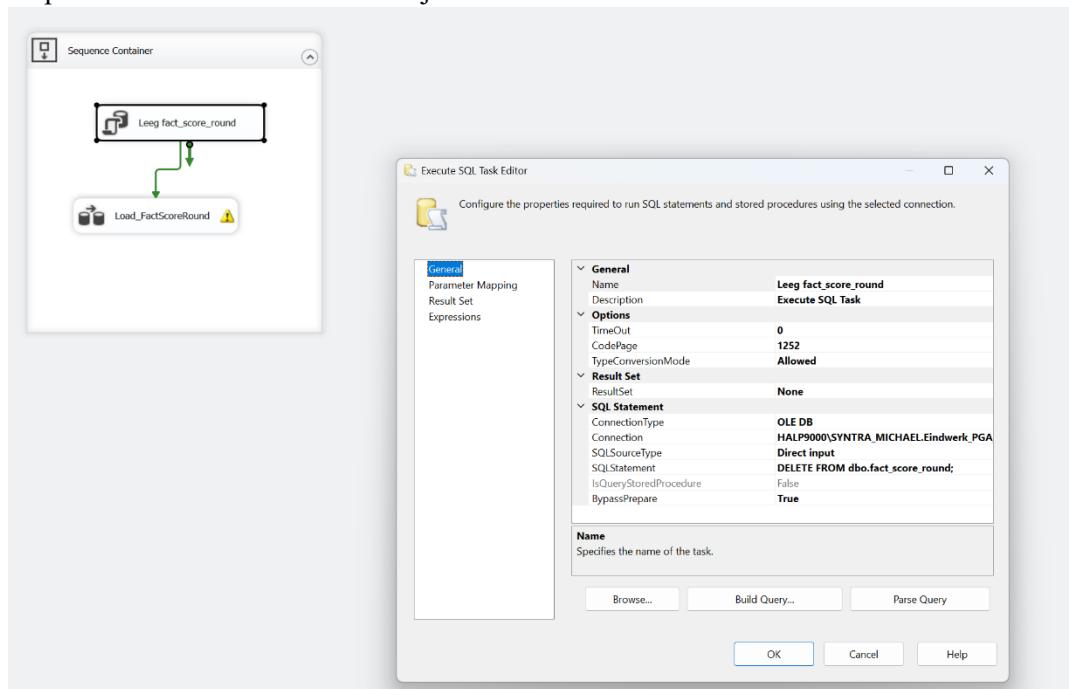
Tabel 21 ETL uitdagingen in Power BI

## § 5. OPMAAK SSIS IN DETAIL

a. Voorbereiding: leegmaken van fact\_score\_round

Vooraleer het laadschema effectief begint met het invullen van de feitentabel, wordt deze eerst volledig leeggemaakt. Dit garandeert dat er bij elke herlading geen dubbele rijen of historische vervuiling ontstaat. De opdracht wordt uitgevoerd via een Execute SQL Task, die als eerste stap binnen de Sequence Container in SSIS is opgenomen.

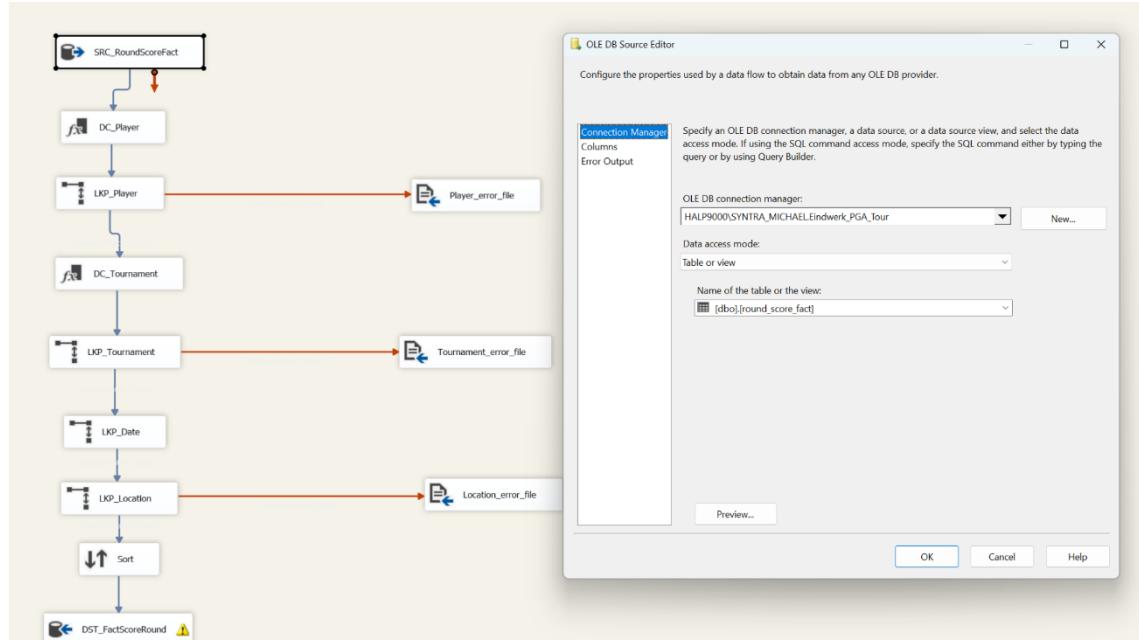
Deze stap is essentieel om reproduceerbare laadsessies mogelijk te maken, zeker bij testladingen of heropbouw van het datawarehouse tijdens de ontwikkelfase.



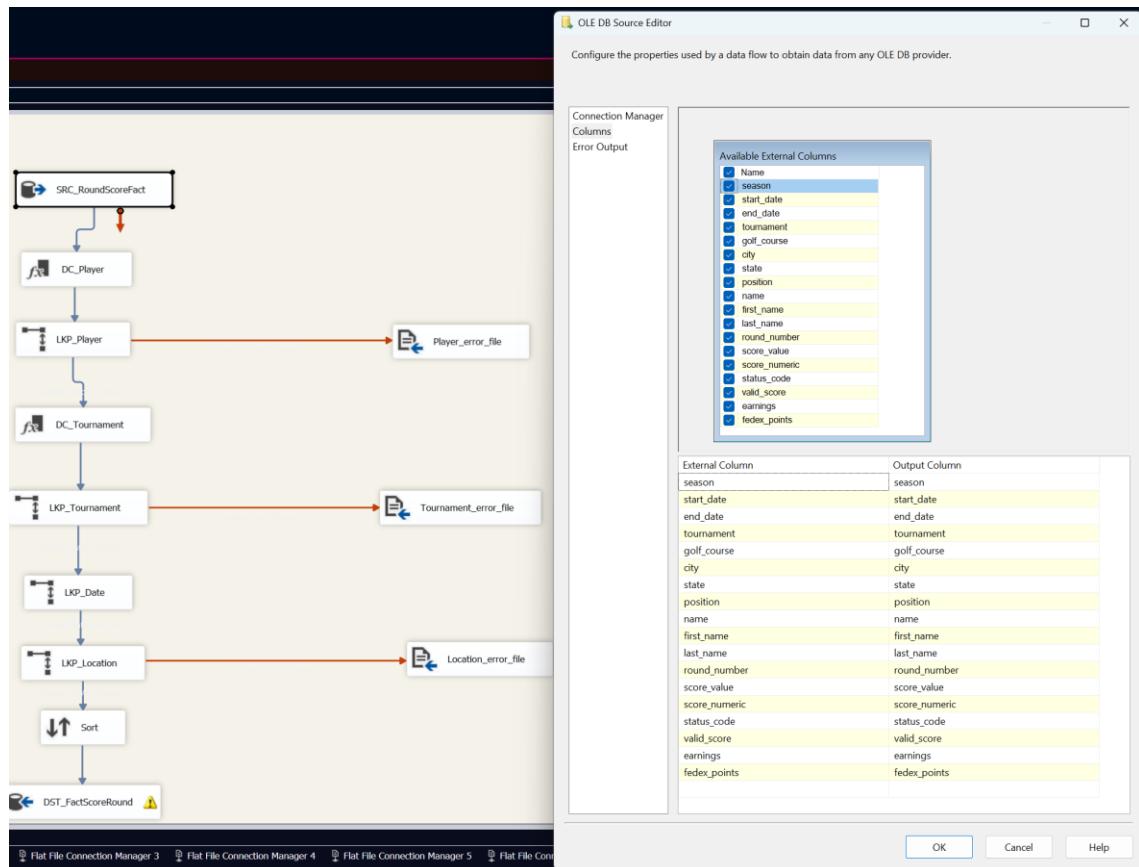
Figuur 4 leegmaken fact\_score\_round SSIS

b. OLE DB Source: inlezen van round\_score\_fact

De eerste stap binnen de Data Flow Task is het inlezen van de ruwe genormaliseerde gegevens uit de stagingtabel round\_score\_fact. Dit gebeurt via een OLE DB Source-component, waarbij alle relevante kolommen worden opgehaald ter voorbereiding van de daaropvolgende lookups en transformaties.



Figuur 5 OLE DB Source file connection manager



Figuur 6 OLE DB Source file mappings

### c. Derived Column: DC\_Player

Om een correcte en foutloze koppeling te maken met de dim\_player-tabel, is het belangrijk dat namen eenduidig, schoon en gestandaardiseerd worden vóór de lookup. In de oorspronkelijke round\_score\_fact konden voor- en familienamen immers spaties, accenten, speciale tekens of verschillen in hoofdlettergebruik bevatten. Om dit te ondervangen werd een Derived Column-transformatie toegepast.

#### Gebruikte expressies:

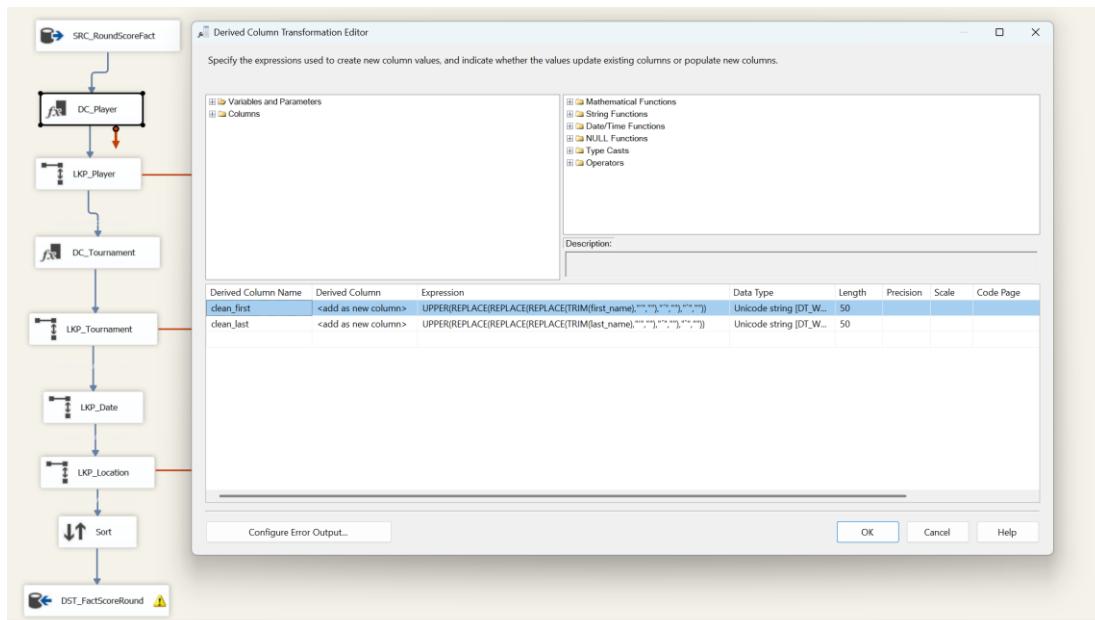
-- VOOR VOORNAAM:

```
UPPER(REPLACE(REPLACE(REPLACE(TRIM(FIRST_NAME), " ", ""), "","", ""), "","", ""))
```

-- VOOR FAMILIENAAM:

```
UPPER(REPLACE(REPLACE(REPLACE(TRIM(LAST_NAME), " ", ""), "","", ""), "","", ""))
```

De REPLACE() wordt hier genest toegepast om verschillende soorten apostrofs en spaties te neutraliseren. De UPPER() zorgt voor case-insensitive vergelijking, wat lookupfouten vermindert.



Figuur 7 Derived Column DC\_Player

#### d. Lookup Player: koppeling met dim\_player

Na de normalisatie van de spelersnamen via DC\_Player, wordt in deze stap een lookup uitgevoerd op de dimensietabel dim\_player. Deze lookup zoekt voor elke rij uit round\_score\_fact de bijhorende player\_id op basis van de gestandaardiseerde naam

- **Doel:** Het koppelen van elke speler aan zijn corresponderende player\_id in dim\_player
- **Reden:** De feitentabel fact\_score\_round mag enkel werken met **surrogaat sleutels**, niet met vrije tekst zoals namen

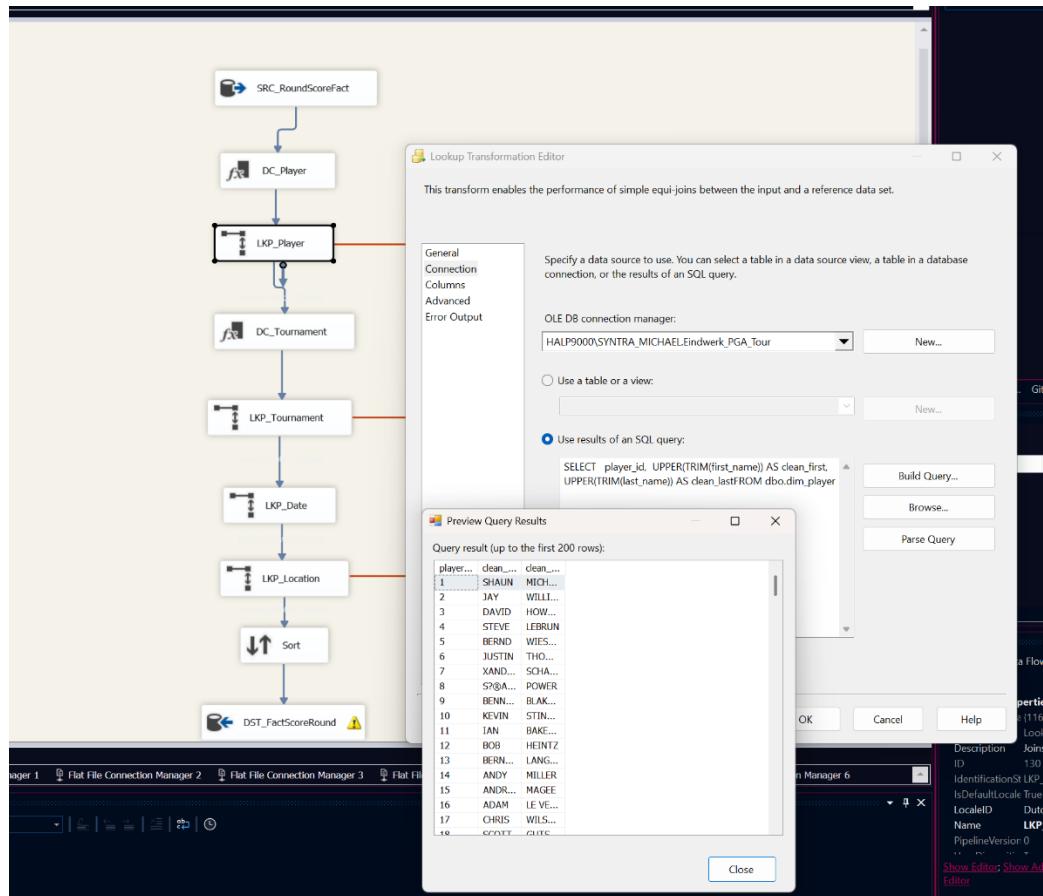
```
SELECT
    PLAYER_ID,
    UPPER(TRIM(FIRST_NAME)) AS CLEAN_FIRST,
    UPPER(TRIM(LAST_NAME)) AS CLEAN_LAST
FROM DBO.DIM_PLAYER
```

<i>Inputkolommen</i>	<i>Lookupkolommen</i>
clean_first	clean_first
clean_last	clean_last

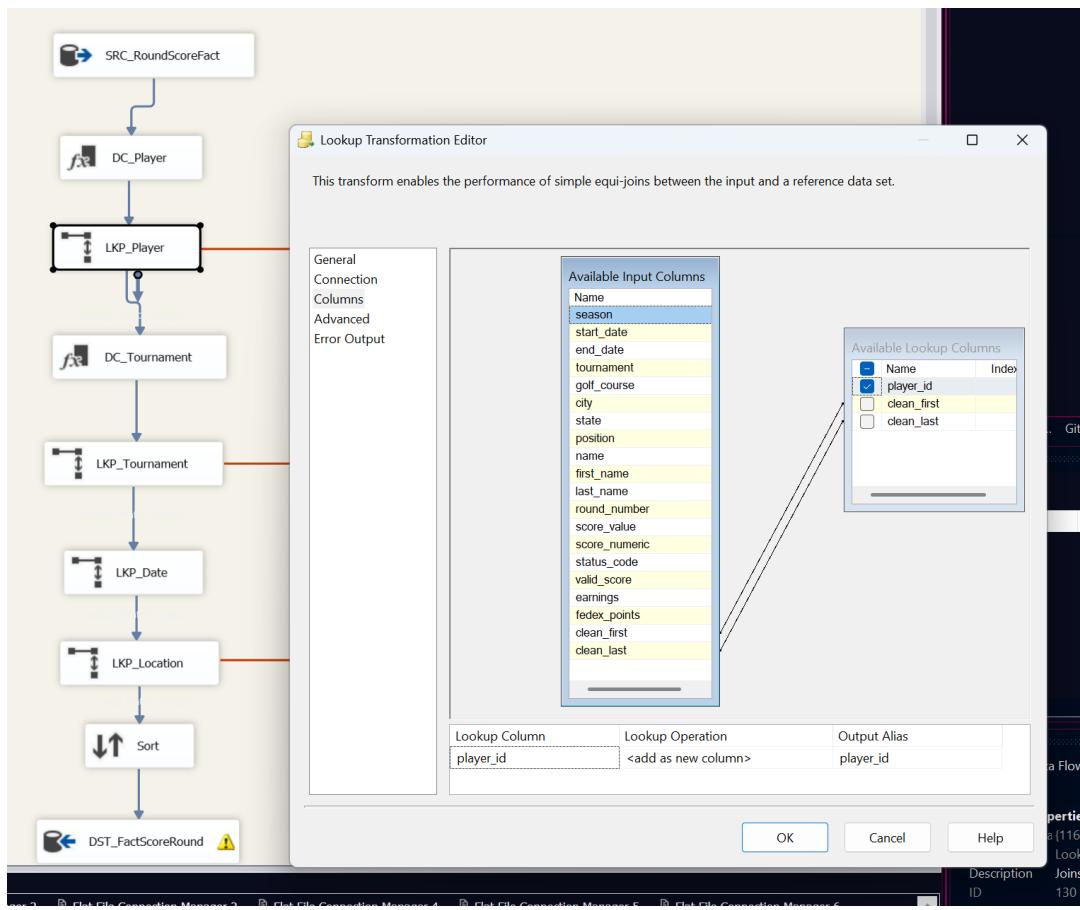
Tabel 22 Input & lookup kolom player

De uitkomst van deze mapping is de kolom player\_id, die vervolgens als foreign key wordt toegevoegd aan de stroom richting de feitentabel.

- Indien een speler niet wordt gevonden in dim\_player, wordt de rij niet afgekeurd, maar doorgestuurd naar een foutbestand (Player\_error\_file) via Redirect Row.
- Dit maakt het mogelijk om ontbrekende spelers achteraf te analyseren en toe te voegen.



Figuur 8 Lookup Player Connection manager

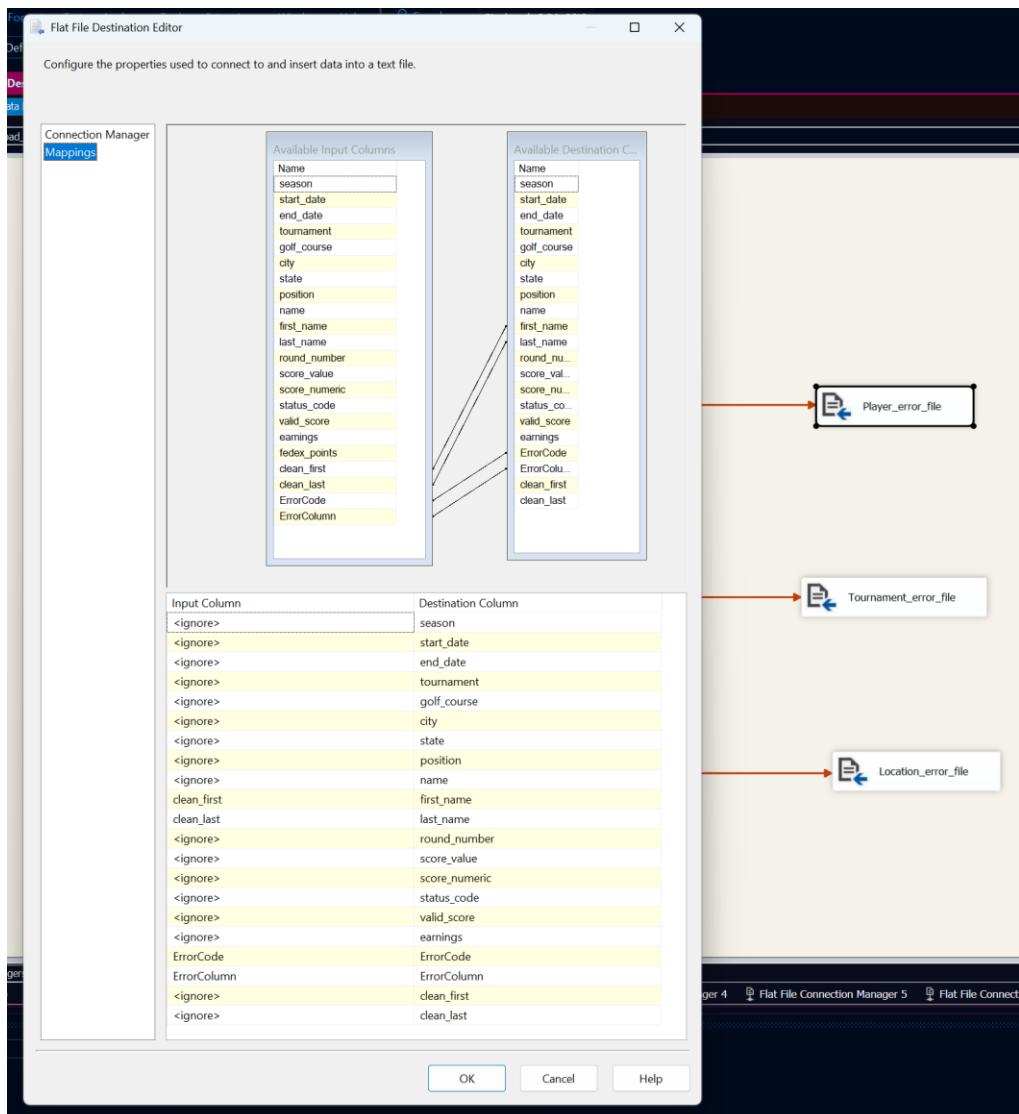


Figuur 9 Lookup Player column mapping

### Foutafhandeling: Player\_error\_file

Wanneer een speler niet correct gematcht wordt via de LKP\_Player component — bijvoorbeeld door afwezigheid in dim\_player, typfouten, of naamafwijkingen — wordt de rij niet geblokkeerd, maar omgeleid naar een foutbestand. Dit gebeurt via Redirect Row en wordt opgevangen in een Flat File Destination: Player\_error\_file.

De belangrijkste velden zoals first\_name, last\_name, score\_value, en tournament worden behouden. Extra debugkolommen zoals ErrorCode en ErrorColumn laten toe om snel de oorzaak van het probleem te achterhalen.



Figuur 10 Player error file mapping

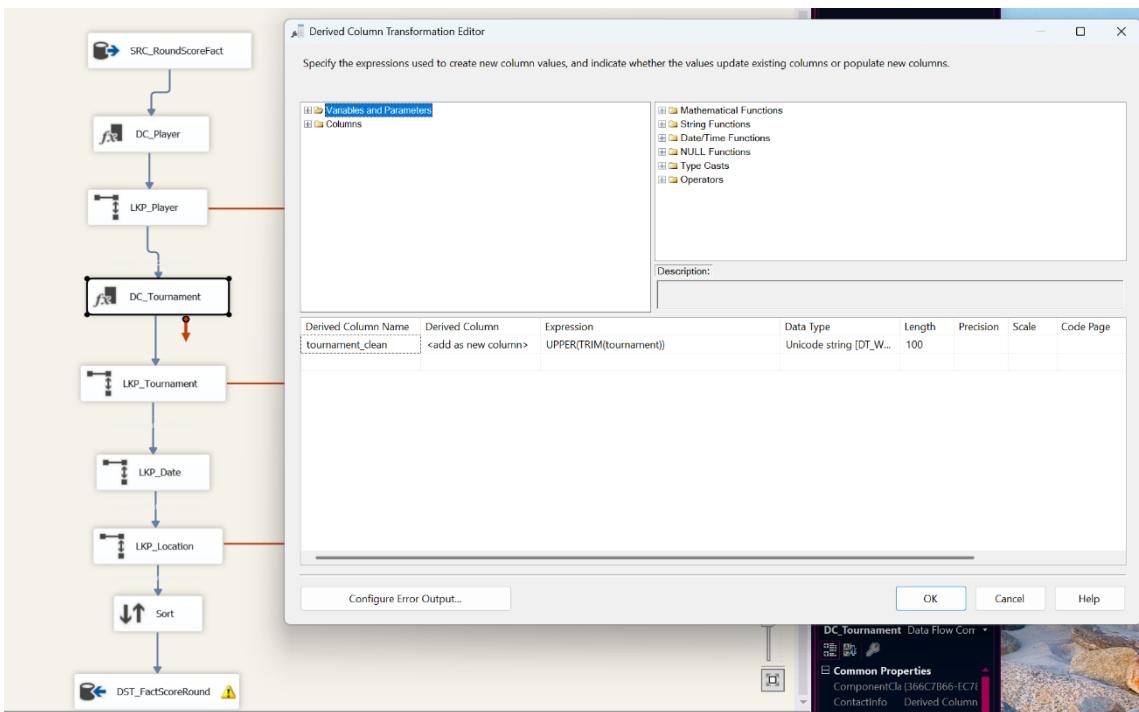
#### e. Derived Column: DC\_Tournament

Net zoals bij spelersnamen, was het ook bij toernooienamen belangrijk om inconsistente schrijfwijzen, hoofdlettergebruik en spaties op te schonen. Daarom werd een Derived Column-transformatie toegevoegd vóór de lookup op dim\_tournament.

- Standaardiseren van de toernooienamen om lookupfouten te vermijden
- Vervangen van kleine tekstuele verschillen door één uniforme waarde
- Vermijden van problemen zoals “The Masters” ≠ “THE MASTERS” of “Genesis Invitational” ≠ “GENESIS INVITATIONAL”

```
UPPER(TRIM(TOURNAMENT))
```

Deze expressie verwijdert spaties aan het begin en einde (TRIM), en zet alle tekens om naar hoofdletters (UPPER). Hierdoor wordt de vergelijking met dim\_tournament case-insensitive en whitespace-neutraal.



Figuur 11 Derived Column Tournament

#### f. Lookup Tournament: koppeling met dim\_tournament

Na het standaardiseren van de toernooinnamen in DC\_Tournament wordt in deze stap een lookup uitgevoerd op dim\_tournament. Het doel is om de tekstuele naam tournament om te zetten naar de corresponderende tournament\_id, conform de Kimball-aanpak waarin de feitentabel enkel surrogaat sleutels bevat.

```

SELECT
    TOURNAMENT_ID,
    UPPER(TRIM(TOURNAMENT)) AS CLEAN_TOURNAMENT
FROM DBO.DIM_TOURNAMENT

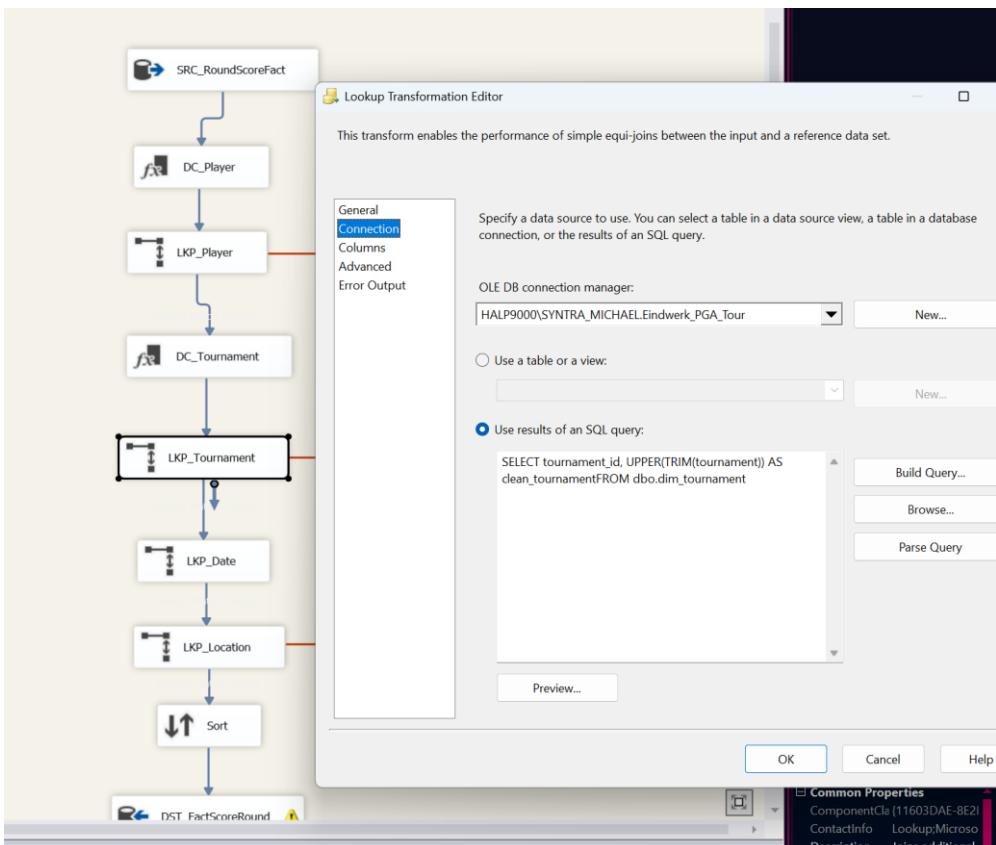
```

De speler–ronde–toernooi–records uit round\_score\_fact worden nu uitgebreid met de correcte tournament\_id afkomstig uit dim\_tournament, wat cruciaal is voor:

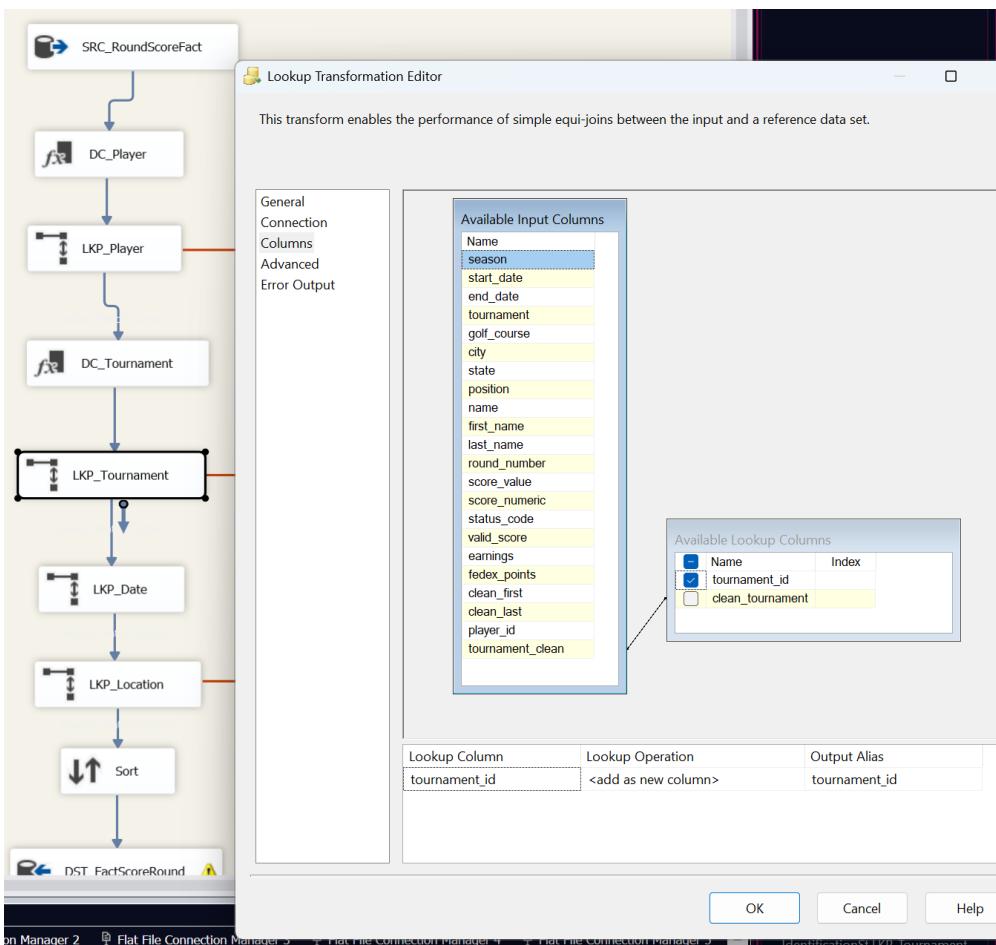
- het correct linken van earnings aan specifieke toernooien,
- seizoensanalyses op toernooiniveau in Power BI,
- het vermijden van duplicatie door inconsistente schrijfwijzen.

<b>Inputkolom</b>	<b>Lookupkolom</b>
tournament_clean	clean_tournament

Tabel 23 Input & Lookup kolom tournament



Figuur 12 Lookup Tournament connection manager

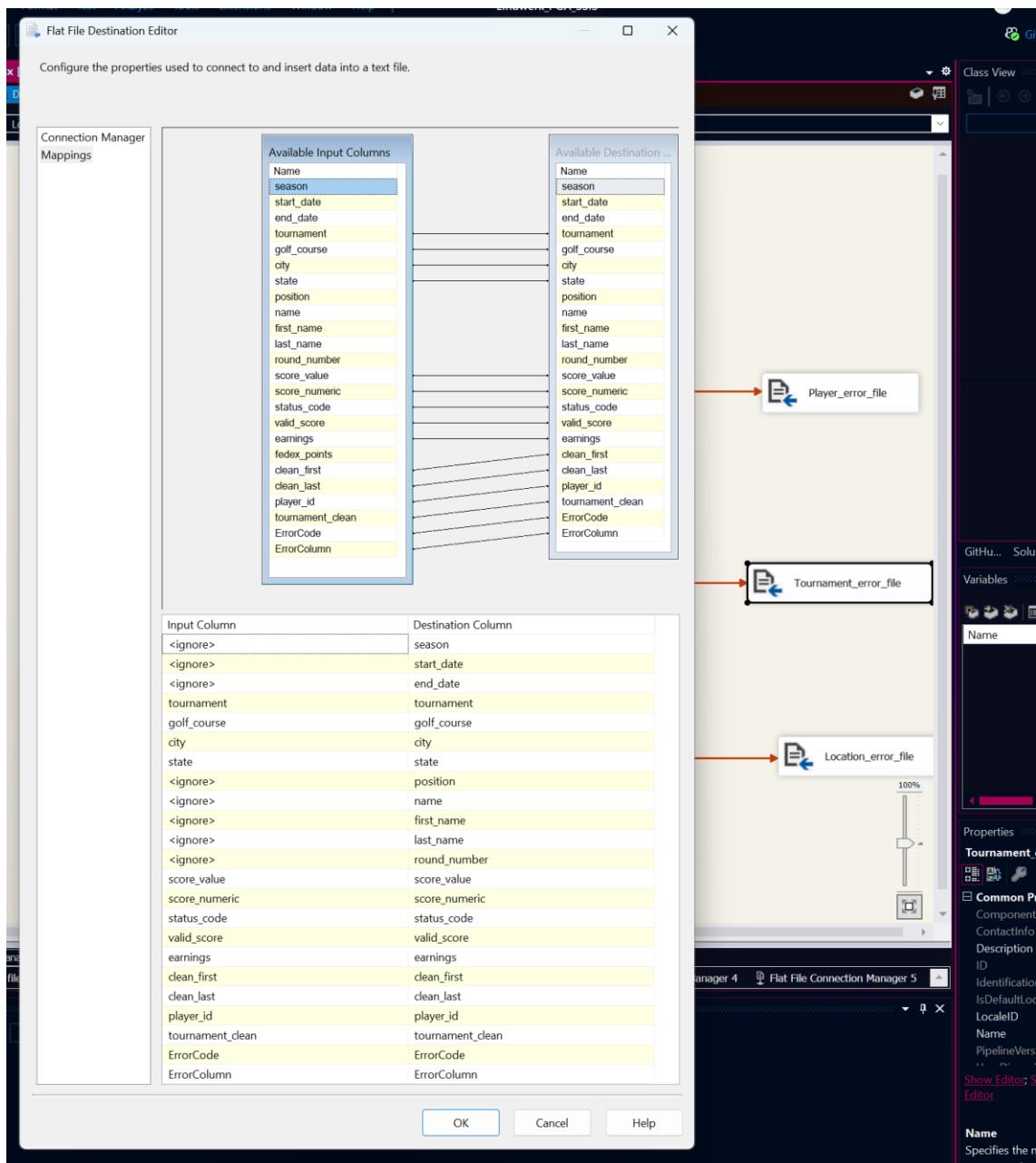


Figuur 13 Lookup Tournament column mapping

### Foutafhandeling: Tournament\_error\_file

Net zoals bij de spelerslookup werd ook bij LKP\_Tournament voorzien in een robuuste foutafhandeling. Als een toernooinaam geen match vindt in dim\_tournament — bijvoorbeeld door typefouten, spaties of ontbrekende records — dan wordt de rij niet geblokkeerd, maar via Redirect Row doorgestuurd naar een foutbestand.

- Rijen zonder correcte tournament\_id loggen voor latere inspectie
- Matchingproblemen traceerbaar maken zonder het ETL-proces te stoppen
- Mogelijkheid bieden om ontbrekende toernooien later toe te voegen aan dim\_tournament



Figuur 14 Tournament error file mapping

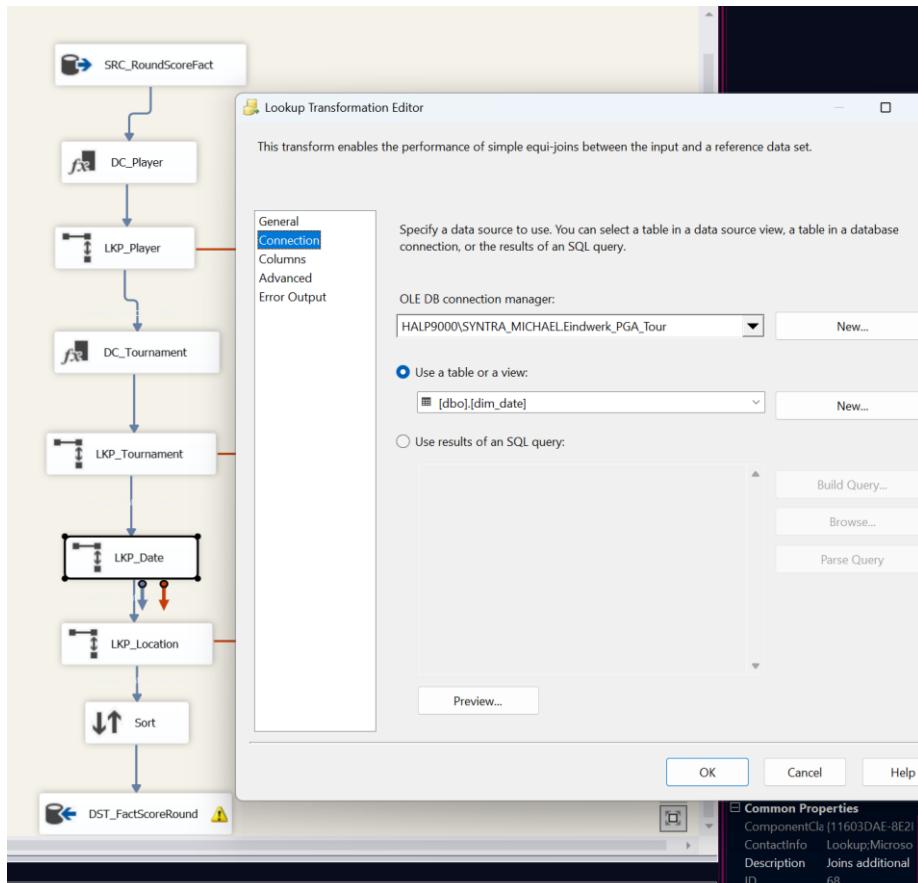
#### g. Lookup Date: koppeling met dim\_date

De dim\_date-tabel in dit model bevat unieke combinaties van startdatum, einddatum en seizoen. In deze stap wordt via een lookup nagegaan welke date\_id overeenkomt met elke combinatie in de stagingtafel round\_score\_fact. Deze sleutel wordt nadien toegevoegd aan de eindfactortabel.

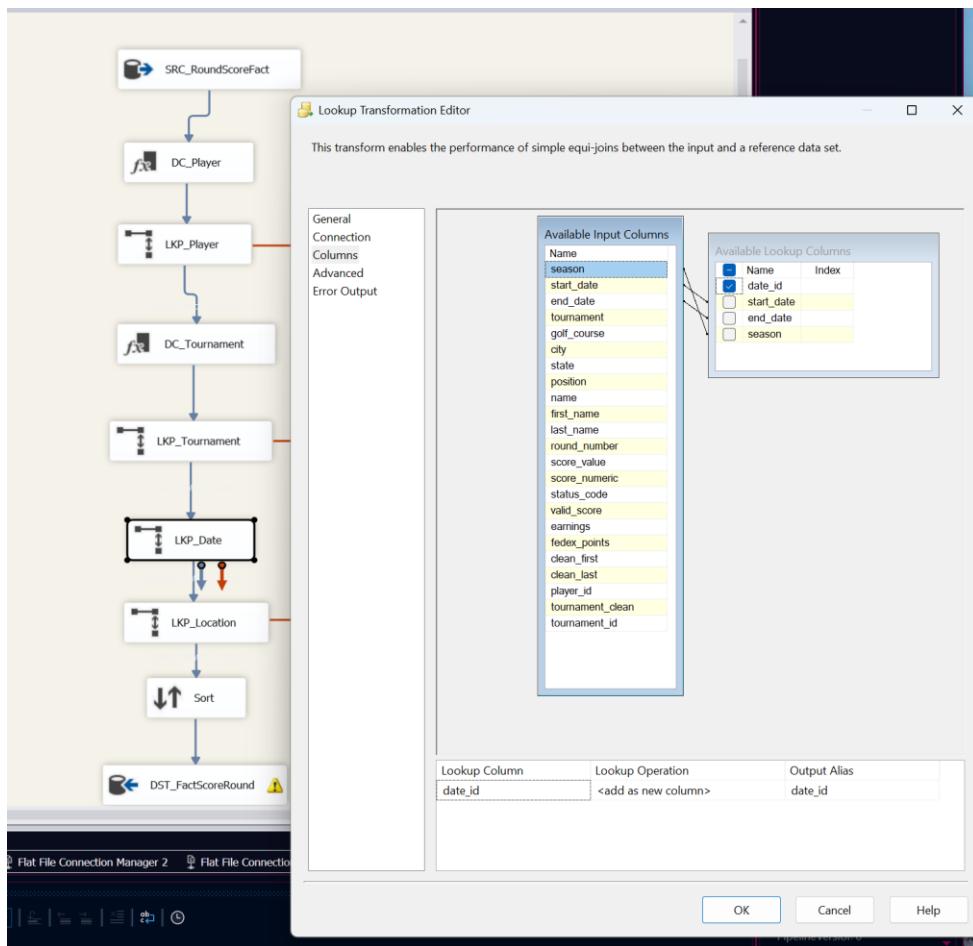
<b>Inputkolommen</b>	<b>Lookupkolommen</b>
<i>start_date</i>	<i>start_date</i>
<i>end_date</i>	<i>end_date</i>
<i>season</i>	<i>season</i>

Tabel 24 Input &amp; Lookup kolom date

- De combinatie van deze drie velden vormt de unieke sleutel binnen dim\_date.
- In tegenstelling tot eerdere lookups (waar een Derived Column nodig was), was hier geen normalisatie vereist, omdat DATE en INT types relatief betrouwbaar zijn voor exacte matching.



Figuur 15 Lookup Date connection manager



Figuur 16 Lookup Date column mapping

#### h. Lookup Location: koppeling met dim\_location

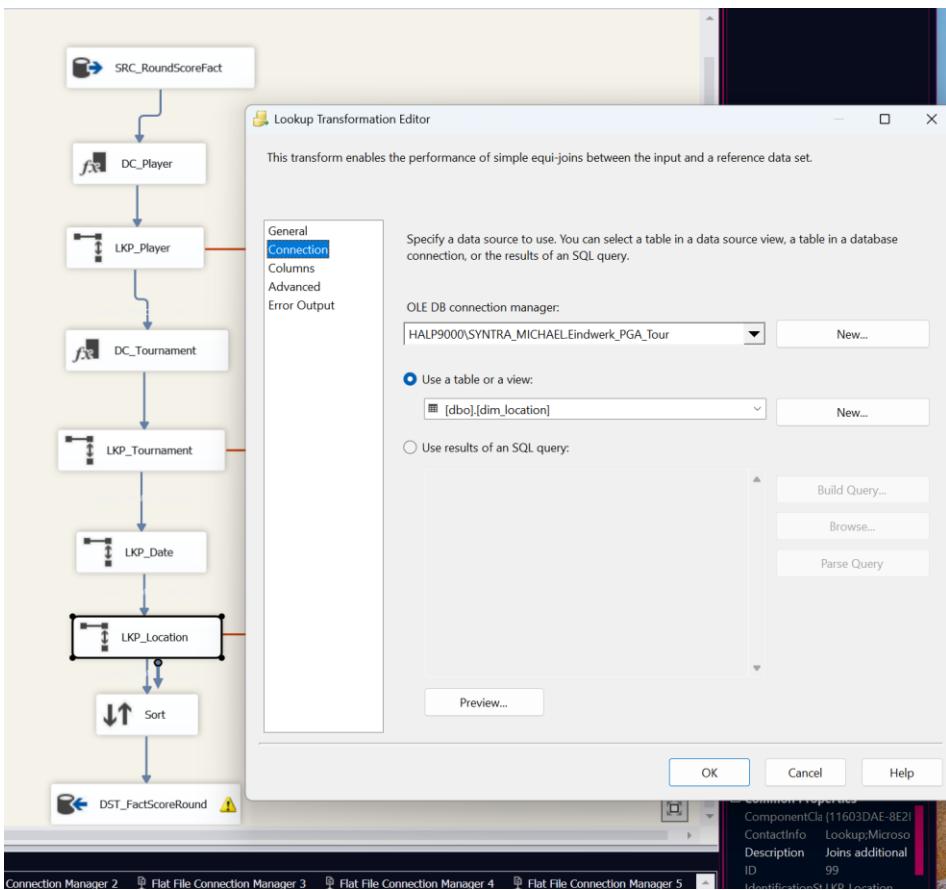
In deze stap wordt een koppeling gelegd tussen de staginggegevens uit round\_score\_fact en de locatiegegevens uit dim\_location. De bedoeling is om op basis van de combinatie van golfbaan, stad en staat de juiste location\_id op te halen voor gebruik in de feitentabel.

Inputkolommen	Lookupkolommen
golf_course	golf_course
city	city
state	state

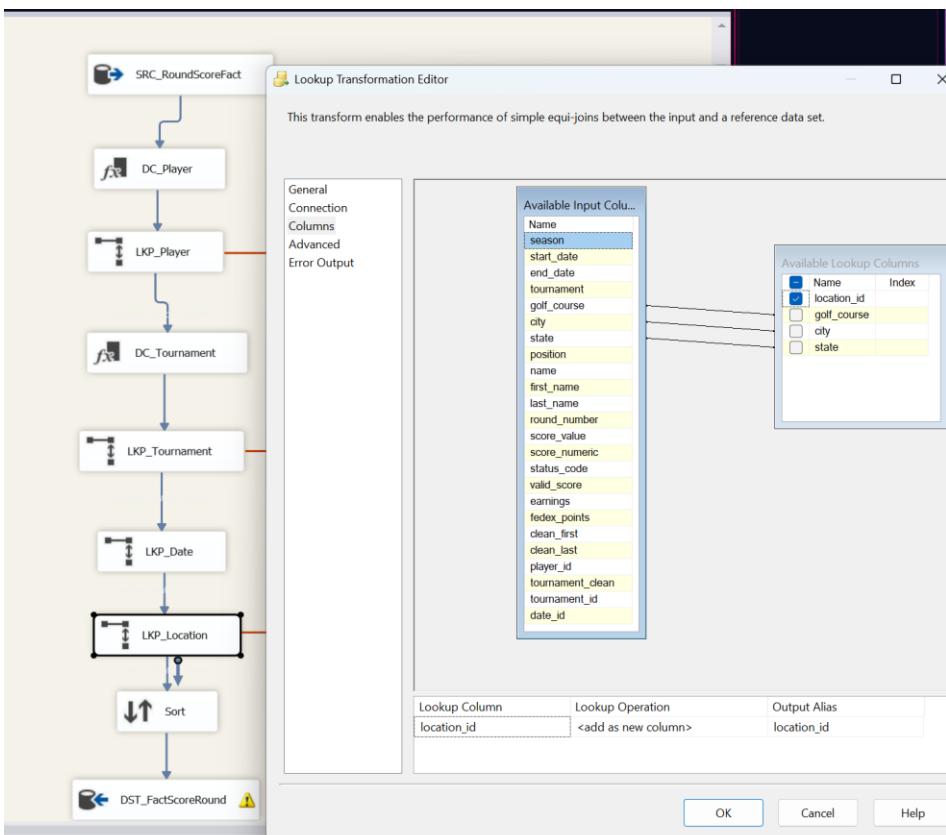
Tabel 25 Input & lookup kolom location

De combinatie van deze drie kolommen vormt een unieke locatie in dim\_location. Wanneer er een match wordt gevonden, wordt de overeenkomstige location\_id toegevoegd als outputkolom.

- Deze lookup is cruciaal voor analyses per golfclub of geografische regio in Power BI.
- Doordat geen afgeleide kolommen werden gebruikt zoals bij speler of toernooi, is het belangrijk dat de gegevens in round\_score\_fact volledig geschoond zijn qua casing en spatiëring vóór deze lookup.



Figuur 17 Lookup Location connection

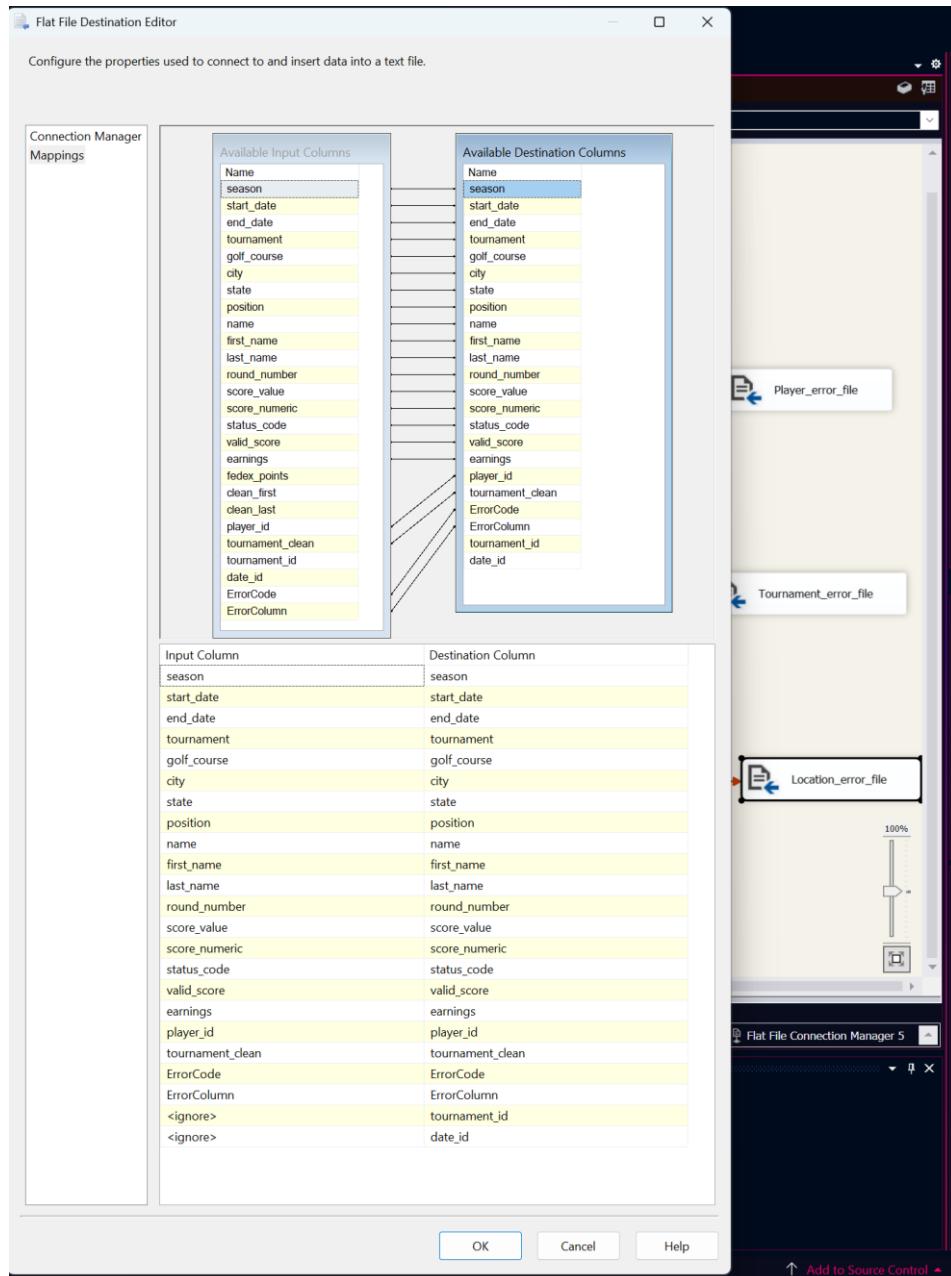


Figuur 18 Lookup location column mapping

### Foutafhandeling: Location\_error\_file

Zoals bij de andere lookups (speler & toernooi), wordt ook voor de LKP\_Location-koppeling een fallbackmechanisme voorzien wanneer een locatie niet correct gematcht wordt met dim\_location. De fout wordt niet genegeerd of geblokkeerd, maar omgeleid naar een foutbestand: Location\_error\_file.

- Alle records waarvoor geen geldige location\_id werd gevonden, loggen naar een apart CSV-bestand.
- Zo behoud je volledige controle over datakwaliteit en kun je:
- ontbrekende locaties aanvullen in dim\_location,
- structurele dataproblemen opsporen (zoals fout gespelde golfbanen of steden).



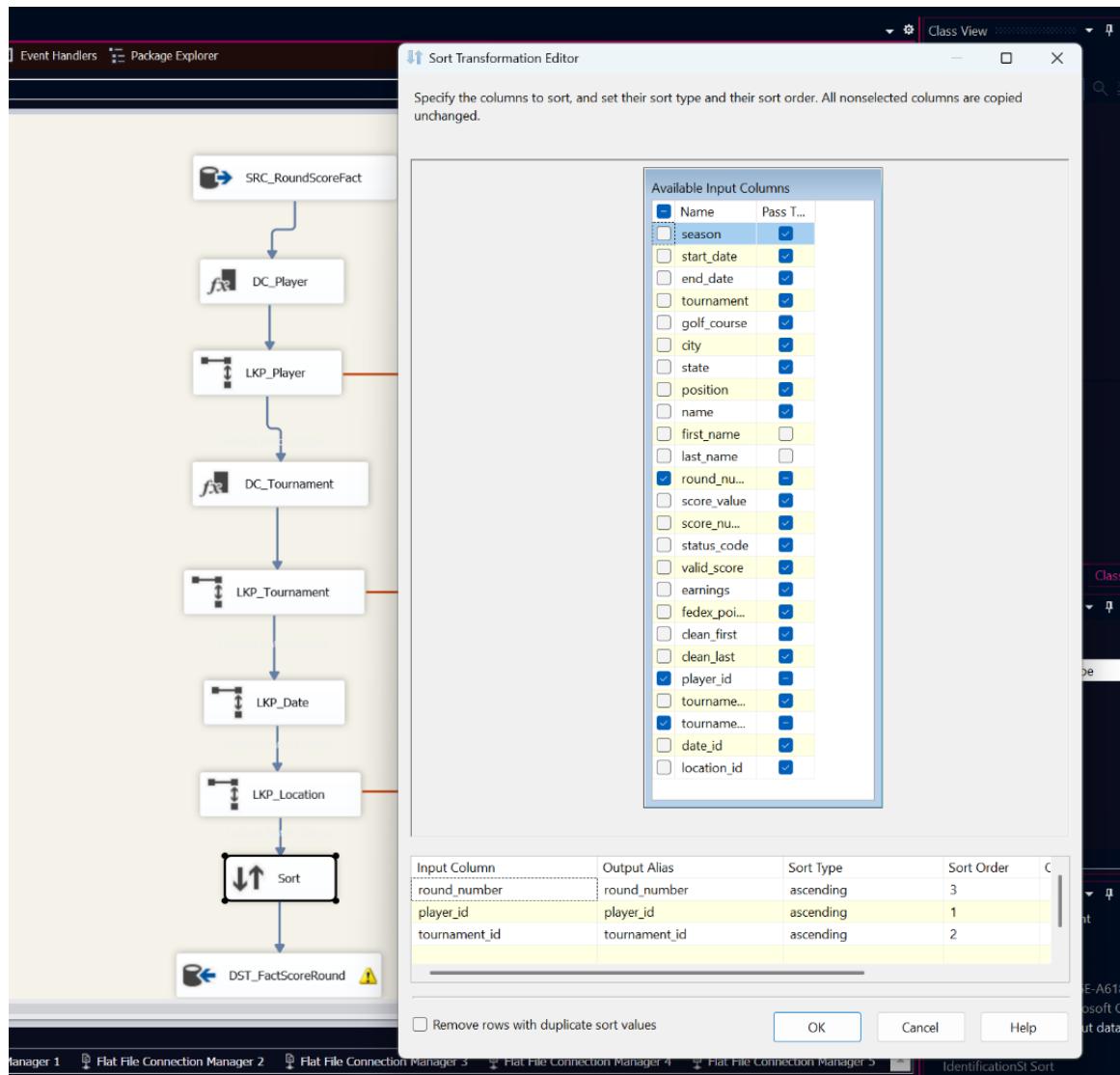
Figuur 19 Location error file mapping

### i. Sort: voorbereiding op laadinstructie

Vooraleer de records geschreven worden naar de fact\_score\_round-tabel via een OLE DB Destination, wordt een Sort Transformation

- Voorbereiding op gestructureerde insert in de eindtabel
- Vermijden van errors in downstream components die gesorteerde input verwachten

De optie “Remove rows with duplicate sort values” is uitgevinkt, wat betekent dat alle records behouden blijven – inclusief identieke rondes binnen dezelfde speler-toernooi-combinatie (wat in dit domein logisch is).



Figuur 20 Sort column mapping

### j. OLE DB Destination: laden in fact\_score\_round

Na het verrijken van alle gegevens met surrogaat sleutels via de lookups, wordt de einddata geschreven naar de feitentabel fact\_score\_round. Dit gebeurt via een OLE DB Destination, het slotpunt van de volledige SSIS-pijplijn.

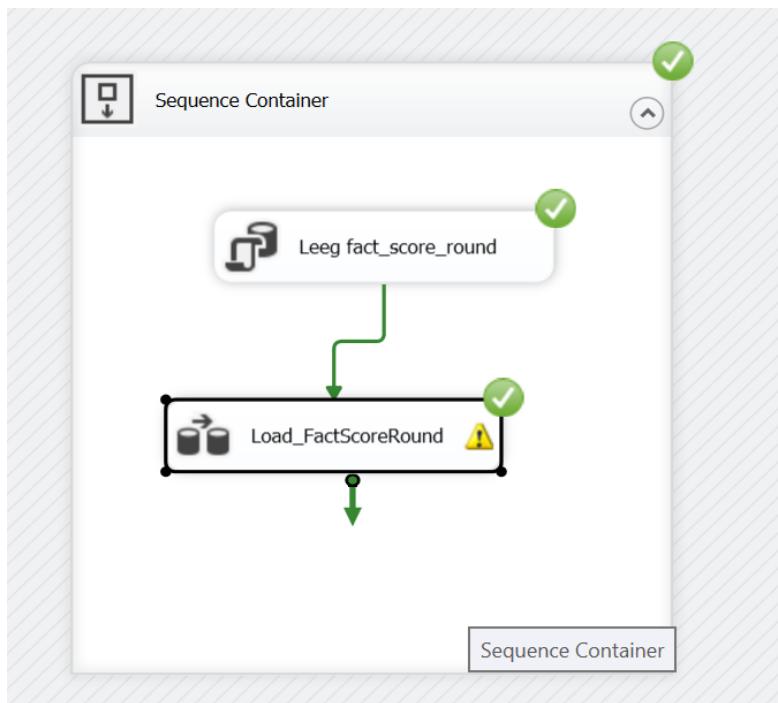
- De gesorteerde en verrijkte records opladen in de centrale feitentabel van het dimensioneel model
- Enkel sleutels, numerieke waarden en KPI's opnemen — volledig conform de Kimball-aanpak

<b>Inputkolom</b>	<b>Doelkolom in feitentabel</b>
<i>player_id</i>	player_id
<i>tournament_id</i>	tournament_id
<i>date_id</i>	date_id
<i>location_id</i>	location_id
<i>round_number</i>	round_number
<i>score_value</i>	score_value
<i>score_numeric</i>	score_numeric
<i>status_code</i>	status_code
<i>valid_score</i>	valid_score
<i>earnings</i>	earnings
<i>position</i>	position
<i>fedex_points</i>	fedex_points

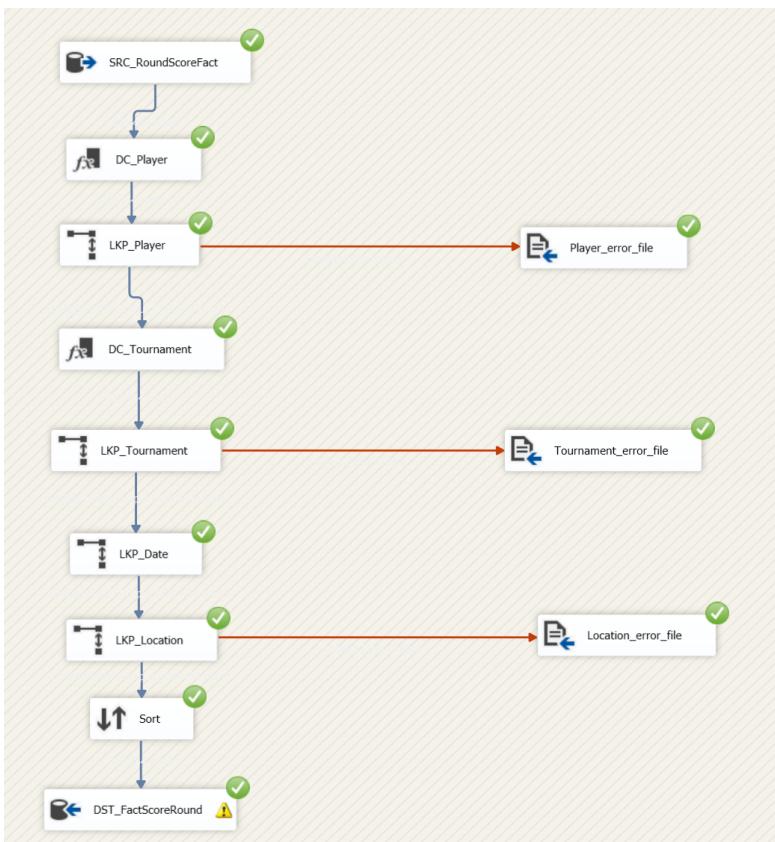
Tabel 26 Input & doelkolom facts table

De kolom fact\_id werd genegeerd, omdat dit een auto-increment kolom is (surrogaat sleutel).

- Door enkel sleutels en numerieke of statuskolommen in te laden, blijft de feitentabel licht, performant en geschikt voor aggregatie en filtering in Power BI.
- Tekstuele kolommen zoals namen, locaties of toernooien worden exclusief via dimensionele relaties opgehaald.
- Deze laadtabel vormt het hart van het datamodel, waarop alle rapporten en visualisaties steunen.



Figuur 21 Werkend flow sequence container



Figuur 22 werkende flow Load\_FactScoreRound

Tijdens het inladen van de gegevens in de feitentabel fact\_score\_round via de OLE DB Destination (DST\_FactScoreRound), werd in SSIS een truncation warning weergegeven:

"TRUNCATION MAY OCCUR DUE TO INSERTING DATA FROM DATA FLOW  
COLUMN 'POSITION' WITH A LENGTH OF 50 TO DESTINATION COLUMN..."

Deze waarschuwing wordt veroorzaakt doordat de kolom position in de dataflow (SSIS) standaard een lengte van 50 tekens meekrijgt, terwijl de kolom position in de doeltabel fact\_score\_round in SQL Server gedefinieerd is met een kortere lengte, met name VARCHAR(10).

Aangezien tijdens analyse van de stagingdata bleek dat alle waarden in position beperkt zijn tot korte statuscodes of numerieke rankings met een maximumlengte van 5 à 6 tekens, werd de waarschuwing als veilig te negeren beschouwd. De kolom in de doeltabel werd niet aangepast, maar de waarschuwing werd gedocumenteerd voor volledigheid en transparantie.

## 6. PYTHON: AUTOMATISCHE PROFIELGENERATIE

Naast de klassieke ETL-pijplijn in SSIS werd aanvullend een Python-script ontwikkeld dat per speler een visueel en informatief HTML-profiel aanmaakt. Dit script demonstreert hoe je met behulp van Python, SQL en visualisaties een interactieve datavisualisatieflow kunt opzetten — volledig los van Power BI.

### § 1. DOEL VAN HET SCRIPT

Het doel is om op basis van de fact\_score\_round-tabel:

- Per speler een individueel HTML-profiel te genereren
- Grafieken toe te voegen over prestaties en verdiensten
- Een indexpagina te maken waarmee alle profielen nagevoerbaar zijn

Deze aanpak simuleert een lichte webapplicatie voor datapresentatie buiten de BI-tooling, nuttig voor rapportages, mobiele weergave of dashboards op schermen.

### § 2. OPBOUW VAN HET SCRIPT

Het script gebruikt volgende Python-modules:

- pandas, pyodbc → SQL-connectie & dataframeverwerking
- matplotlib.pyplot → Grafieken genereren (PNG)
- jinja2 → HTML-rendering met templates
- unicodedata, re → slugify() voor veilige bestandsnamen
- tqdm → Voortgangsbalk bij iteratie

De functie slugify() zorgt ervoor dat spelersnamen als "Rory McIlroy" worden omgezet naar "Rory\_McIlroy" als bestandsnaam.

Er wordt een verbinding gemaakt met de lokale SQL Server-database waarin fact\_score\_round is opgeslagen.

```
CONN = PYODBC.CONNECT(
    'DRIVER={SQL SERVER};'
    'SERVER=HALP9000\SYNTRA_MICHAEL;'
    'DATABASE=EINDWERK_PGA_TOUR;'
    'TRUSTED_CONNECTION=YES;'
)
```

Bestanden worden automatisch gegenereerd als ze nog niet bestaan.

- /spelerprofielen/ - HTML-bestanden
- /spelerprofielen/plots/ -PNG-grafieken
- /templates/ - HTML-template met Jinja2-layout

Per speler wordt:

- **Aantal toernooien** berekend (nunique)
- **Aantal gespeelde rondes** (len)
- **Totaal verdiend** (sum)
- **Earnings per stroke** (earnings / score\_numeric)
- **Min / Max / Gemiddelde scores**

Daarnaast worden grafieken gegenereerd:

- Scoreverloop in de tijd
- Gemiddelde scores per seizoen
- Verdiensten per seizoen

Deze worden opgeslagen als PNG en in het HTML-profiel geïntegreerd.

Een aparte index.html bevat:

- Zoekbalk (via JavaScript)
- Klikbare lijst van alle spelers met links naar hun profielpagina
- Automatisch opgebouwd uit alle gegenereerde bestanden

Je eindigt met een lokale webstructuur bestaande uit:

- HTML-profielen per speler met scores, prestaties en visuals
- Een overzichtspagina met navigatie
- Georganiseerde mapstructuur voor distributie of hosting

### § 3. FINALE PYTHON SCRIPT

```

IMPORT PANDAS AS PD
IMPORT MATPLOTLIB.PYPLOT AS PLT
FROM JINJA2 IMPORT ENVIRONMENT, FILESYSTEMLOADER
IMPORT OS
IMPORT PYODBC
IMPORT RE
IMPORT UNICODEDATA
FROM TQDM IMPORT TQDM

DEF SLUGIFY(VALUE):
    VALUE = STR(VALUE)
    VALUE = UNICODEDATA.NORMALIZE('NFKD', VALUE).ENCODE('ASCII',
'IGNORE').DECODE('ASCII')
    VALUE = RE.SUB(R'^A-ZA-Z0-9_-]+', '_', VALUE)
    RETURN VALUE.STRIPE(_)

# CONNECTIE MET SQL SERVER
CONN = PYODBC.CONNECT(
    'DRIVER={SQL SERVER};'
    'SERVER=HALP9000\SYNTRA_MICHAEL;'
    'DATABASE=EINDWERK_PGA_TOUR;'
    'TRUSTED_CONNECTION=YES;'
)

# DATA OPHALEN
QUERY = """
SELECT
    P.FIRST_NAME + ' ' + P.LAST_NAME AS SPELER_NAAM,
    T.TOURNAMENT,
    D.START_DATE,
    F.SCORE_NUMERIC,
    F.EARNINGS

    FROM FACT_SCORE_ROUND F
    INNER JOIN DIM_PLAYER P ON F.PLAYER_ID = P.PLAYER_ID
    INNER JOIN DIM_TOURNAMENT T ON F.TOURNAMENT_ID = T.TOURNAMENT_ID
    INNER JOIN DIM_DATE D ON F.DATE_ID = D.DATE_ID

    AND F.SCORE_NUMERIC IS NOT NULL
    AND F.EARNINGS IS NOT NULL
"""

```

```

DF = PD.READ_SQL(QUERY, CONN)
DF['START_DATE'] = PD.TO_DATETIME(DF['START_DATE'])
DF['SEASON'] = DF['START_DATE'].DT.YEAR

BASE_DIR = OS.PATH.DIRNAME(OS.PATH.ABS PATH(__FILE__))
OUTPUT_DIR = OS.PATH.JOIN(BASE_DIR, 'SPELERPROFIELEN')
PLOT_DIR = OS.PATH.JOIN(OUTPUT_DIR, 'PLOTS')
TEMPLATE_DIR = OS.PATH.JOIN(BASE_DIR, 'TEMPLATES')

OS.MAKEDIRS(OUTPUT_DIR, EXIST_OK=TRUE)
OS.MAKEDIRS(PLOT_DIR, EXIST_OK=TRUE)
OS.MAKEDIRS(TEMPLATE_DIR, EXIST_OK=TRUE)

TEMPLATE_PATH = OS.PATH.JOIN(TEMPLATE_DIR, 'SPELER_TEMPLATE.HTML')
IF NOT OS.PATH.EXISTS(TEMPLATE_PATH):
    WITH OPEN(TEMPLATE_PATH, 'W', ENCODING='UTF-8') AS F:
        F.WRITE("""<!DOCTYPE HTML>
<HTML LANG='NL'>
<HEAD>
    <META CHARSET='UTF-8'>
    <TITLE>SPELERPROFIEL - {{ NAAM }}</TITLE>
    <STYLE>
        BODY { FONT-FAMILY: ARIAL; MARGIN: 30PX; }
        H1 { COLOR: DARKBLUE; }
        .STATS { MARGIN-BOTTOM: 20PX; }
        IMG { WIDTH: 800PX; HEIGHT: 400PX; BORDER: 1PX SOLID #CCC; MARGIN-
BOTTOM: 20PX; }
        A.BACK-BUTTON {
            DISPLAY: INLINE-BLOCK;
            PADDING: 8PX 12PX;
            BACKGROUND-COLOR: #EEE;
            BORDER: 1PX SOLID #CCC;
            BORDER-RADIUS: 5PX;
            TEXT-DECORATION: NONE;
            COLOR: DARKBLUE;
            FONT-WEIGHT: BOLD;
        }
        A.BACK-BUTTON:HOVER {
            BACKGROUND-COLOR: #DDD;
        }
    </STYLE>
```

```

</HEAD>
<BODY>
    <H1>SPELERPROFIEL: {{ NAAM }}</H1>
    <DIV CLASS="STATS">
        <P><STRONG>AANTAL TOERNOOIEN:</STRONG> {{ TOERNOOIEN }}</P>
        <P><STRONG>AANTAL GESPEELDE RONDEN:</STRONG> {{ RONDEN }}</P>
        <P><STRONG>TOAAL VERDIEND:</STRONG> ${{ ' {:.0F}'.FORMAT(TOTAAL) }}</P>
        <P><STRONG>GEM. VERDIENSTEN PER SLAG:</STRONG> ${{ ' {:.2F}'.FORMAT(PER_SLAG) }}</P>
        <P><STRONG>GEMIDDELDE SCORE:</STRONG> {{ ' {:.2F}'.FORMAT(GEM_SCORE) }}</P>
        <P><STRONG>BESTE SCORE:</STRONG> {{ BESTE_SCORE }}</P>
        <P><STRONG>SLECHTSTE SCORE:</STRONG> {{ SLECHTSTE_SCORE }}</P>
    </DIV>
    <H2>SCOREVERLOOP</H2>
    <IMG SRC="{{ PLOT }}" ALT="SCORE VERLOOP">
    <H2>GEMIDDELDE SCORE PER SEIZOEN</H2>
    <IMG SRC="{{ PLOT_SCORE_SEASON }}" ALT="SEIZOENSSCORE VERLOOP">
    <H2>VERDIENSTEN PER SEIZOEN</H2>
    <IMG SRC="{{ PLOT_EARNINGS_SEASON }}" ALT="SEIZOENSVERDIENSTEN VERLOOP">
    <P><A CLASS="BACK-BUTTON" HREF="../INDEX.HTML">← TERUG NAAR OVERZICHT</A></P>
</BODY>
</HTML>""")
```

```

ENV = ENVIRONMENT(LOADER=FILESYSTEMLOADER(TEMPLATE_DIR))
TEMPLATE = ENV.GET_TEMPLATE('SPELER_TEMPLATE.HTML')
```

```
INDEX_LINKS = []
```

```

FOR NAAM, DATA IN TQDM(DF.GROUPBY('SPELER_NAAM'),
TOTAL=DF['SPELER_NAAM'].NUNIQUE(), DESC="SPELERPROFIELEN"):
    SLUG = SLUGIFY(NAAM)
    TOERNOOIEN = DATA['TOURNAMENT'].NUNIQUE()
    TOTAAL = DATA[EARNINGS].SUM()
    PER_SLAG = (DATA[EARNINGS] / DATA[SCORE_NUMERIC]).MEAN()
    GEM_SCORE = DATA[SCORE_NUMERIC].MEAN()
    BESTE_SCORE = DATA[SCORE_NUMERIC].MIN()
    SLECHTSTE_SCORE = DATA[SCORE_NUMERIC].MAX()
    RONDEN = LEN(DATA)
```

```

# SCORE VERLOOP
DATA_SORTED = DATA.SORT_VALUES('START_DATE')
PLT.FIGURE(figsize=(10, 5))
PLT.PLOT(DATA_SORTED['START_DATE'], DATA_SORTED['SCORE_NUMERIC'],
MARKER='O', linewidth=2)
PLT.TITLE(F'SCOREVERLOOP VAN {NAAM}')
PLT.YLABEL('SCORE')
PLT.XLABEL('DATUM')
PLT.GRID(TRUE)
PLT.XTICKS(rotation=45)
PLT.TIGHT_LAYOUT()
PLOT_FILE = OS.PATH.JOIN(PLOT_DIR, F'{SLUG}.PNG")
PLT.SAVEFIG(PLOT_FILE)
PLT.CLOSE()

# SCORE PER SEIZOEN
SCORE_SEASON =
DATA.GROUPBY('SEASON')['SCORE_NUMERIC'].MEAN().RESET_INDEX()
PLT.FIGURE(figsize=(8, 4))
PLT.PLOT(SCORE_SEASON['SEASON'], SCORE_SEASON['SCORE_NUMERIC'],
MARKER='O', COLOR='STEELBLUE')
PLT.TITLE(F'SCORE PER SEIZOEN - {NAAM}')
PLT.XLABEL('SEIZOEN')
PLT.YLABEL('GEM. SCORE')
PLT.GRID(TRUE)
PLT.TIGHT_LAYOUT()
PLOT_SCORE_SEASON = OS.PATH.JOIN(PLOT_DIR,
F"SCORE_PER_SEASON_{SLUG}.PNG")
PLT.SAVEFIG(PLOT_SCORE_SEASON)
PLT.CLOSE()

# VERDIENSTEN PER SEIZOEN
EARNINGS_SEASON =
DATA.GROUPBY('SEASON')['EARNINGS'].SUM().RESET_INDEX()
PLT.FIGURE(figsize=(8, 4))
PLT.PLOT(EARNINGS_SEASON['SEASON'], EARNINGS_SEASON['EARNINGS'],
MARKER='O', COLOR='GREEN')
PLT.TITLE(F'VERDIENSTEN PER SEIZOEN - {NAAM}')
PLT.XLABEL('SEIZOEN')
PLT.YLABEL('TOTALE VERDIENSTEN ($)')
PLT.GRID(TRUE)

```

```

PLT.TIGHT_LAYOUT()
PLOT_EARNINGS_SEASON = OS.PATH.JOIN(PLOT_DIR,
F"EARNINGS_PER_SEASON_{SLUG}.PNG")
PLT.SAVEFIG(PLOT_EARNINGS_SEASON)
PLT.CLOSE()

HTML = TEMPLATE.RENDER(
    NAAM=NAAM,
    TOERNOOIEN=TOERNOOIEN,
    TOTAAL=TOTAAL,
    PER_SLAG=PER_SLAG,
    GEM_SCORE=GEM_SCORE,
    BESTE_SCORE=BESTE_SCORE,
    SLECHTSTE_SCORE=SLECHTSTE_SCORE,
    RONDES=RONDES,
    PLOT=OS.PATH.JOIN('PLOTS', F"{SLUG}.PNG"),
    PLOT_SCORE_SEASON=OS.PATH.JOIN('PLOTS',
F"SCORE_PER_SEASON_{SLUG}.PNG"),
    PLOT_EARNINGS_SEASON=OS.PATH.JOIN('PLOTS',
F"EARNINGS_PER_SEASON_{SLUG}.PNG")
)

WITH OPEN(OS.PATH.JOIN(OUTPUT_DIR, F"{SLUG}.HTML"), 'W', ENCODING='UTF-
8') AS F:
    F.WRITE(HTML)

INDEX_LINKS.APPEND(F'<LI><A
HREF="SPELERPROFIELEN/{SLUG}.HTML">{NAAM}</A></LI>')

# GENEREER INDEXPAGINA
INDEX_HTML = F"""
<!DOCTYPE HTML>
<HTML LANG="NL">
<HEAD>
    <META CHARSET="UTF-8">
    <TITLE>SPELERPROFIELEN</TITLE>
    <STYLE>
        BODY {{ FONT-FAMILY: ARIAL; MARGIN: 30PX; }}
        H1 {{ COLOR: DARKGREEN; }}
        INPUT {{ WIDTH: 300PX; PADDING: 8PX; MARGIN-BOTTOM: 15PX; }}
        UL {{ LIST-STYLE-TYPE: NONE; PADDING: 0; }}
        LI {{ MARGIN-BOTTOM: 6PX; }}
        A {{ COLOR: DARKGREEN; TEXT-DECORATION: NONE; }}
    </STYLE>
</HEAD>
<BODY>
    <H1>SPELERPROFIELEN</H1>
    <UL>
        {INDEX_LINKS}
    </UL>
</BODY>

```

```

</STYLE>
<SCRIPT>
    FUNCTION ZOEK() {{
        VAR INPUT =
DOCUMENT.GETELEMENTBYID('ZOEK').VALUE.TOLOWERCASE();
        VAR ITEMS = DOCUMENT.QUERYSELECTORALL('UL LI');
        ITEMS.FOREACH(FUNCTION(LI) {{
            LI.STYLE.DISPLAY =
LI.TEXTCONTENT.TOLOWERCASE().INCLUDES(INPUT) ? '' : 'NONE';
        }});
    }}
</SCRIPT>
</HEAD>
<BODY>
    <H1>SPELERPROFIELEN</H1>
    <INPUT TYPE="TEXT" ID="ZOEK" PLACEHOLDER="ZOEK OP NAAM...">
    ONKEYUP="ZOEK()"
    <UL>
        {"JOIN(INDEX_LINKS)}
    </UL>
</BODY>
</HTML>
"""

WITH OPEN(OS.PATH.JOIN(BASE_DIR, 'INDEX.HTML'), 'W', ENCODING='UTF-8') AS F:
    F.WRITE(INDEX_HTML)

PRINT(F"\n{LEN(INDEX_LINKS)} SPELERPROFIELEN GEGENEREERD IN:
{OUTPUT_DIR}")
PRINT("OPEN: INDEX.HTML")

```

#### § 4. PERFORMANTIE

Hoewel het Python-script een hoge mate van controle en maatwerk biedt, moet wel vermeld worden dat het volledige runproces ongeveer 15 minuten in beslag neemt. Dit komt door:

- het verwerken van honderden spelers afzonderlijk,
- het genereren van meerdere grafieken per speler,
- en het renderen en wegschrijven van individuele HTML-bestanden.

In vergelijking met Power BI, waar real-time slicing en filtering mogelijk is binnen seconden, is dit script duidelijk minder concurrentieel op vlak van snelheid of interactie. Het is dus vooral geschikt voor batchgewijze publicatie van statische rapporten, eerder dan interactieve BI-toepassingen.



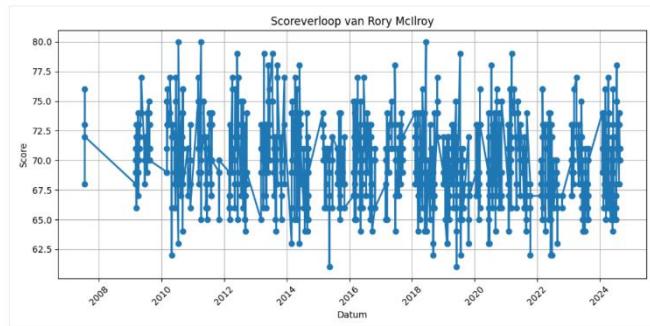
Figuur 23 loading time python script

## § 5. RESULTAAT PYTHON SCRIPT

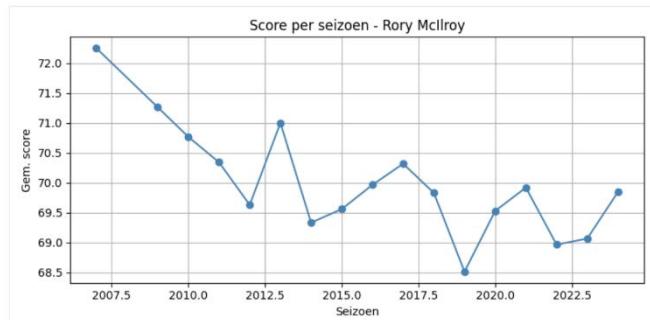
### Spelerprofiel: Rory McIlroy

Aantal toernooien: 69  
 Aantal gespeelde rondes: 920  
 Totaal verdienst: \$329,437,101  
 Gem. verdiensten per slag: \$5,203.99  
 Gemiddelde score: 69.81  
 Beste score: 61  
 Slechtste score: 80

### Scoreverloop

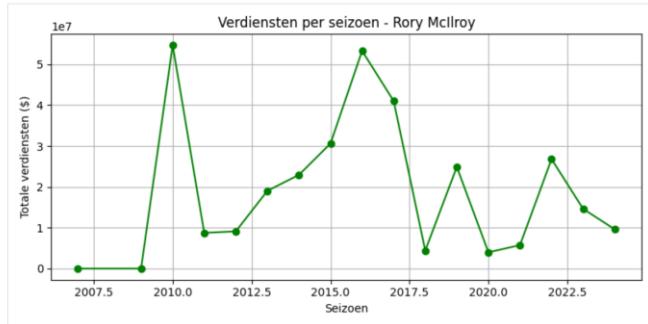


### Gemiddelde score per seizoen



Figuur 24 resultaat python script(1)

### Verdiensten per seizoen



[← Terug naar overzicht](#)

Figuur 25 resultaat python script (2)

## 7. VISUALISATIE IN POWER BI

Na het laden van de gegevens in het stervormig dimensioneel model via SSIS, werd Power BI ingezet om op een interactieve, visuele en datagestuurde manier inzichten te genereren. Power BI biedt via slicers, dynamische measures en gebruikersinteractie een enorme meerwaarde boven statische rapporten of scriptgebaseerde output.

### § 1. STRUCTUUR VAN HET POWER BI-MODEL

Het datamodel bestaat uit vijf hoofdtabellen en twee functionele hulpelementen:

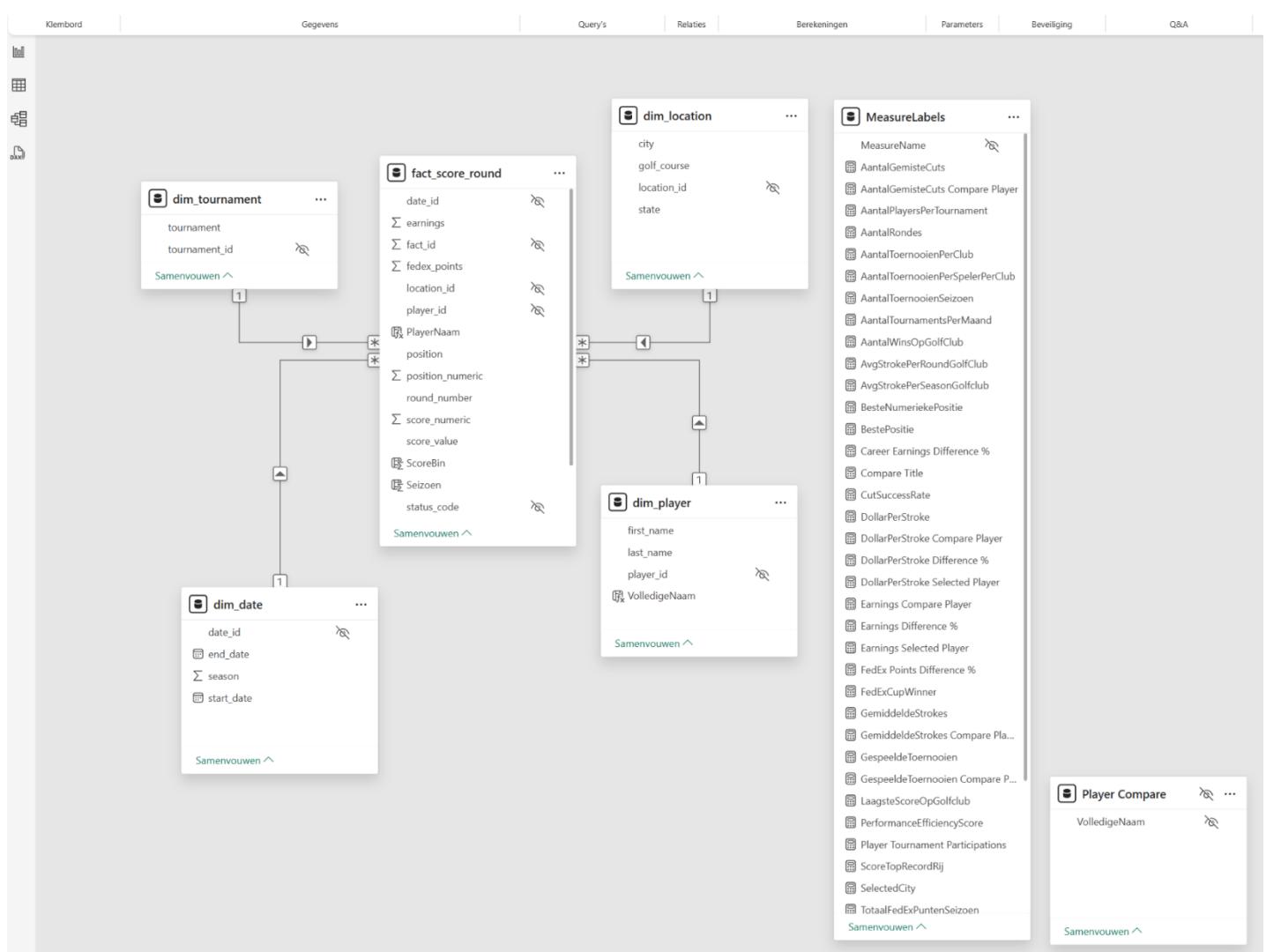
#### Feitentabel

- **fact\_score\_round**: bevat één record per speler-toernooi-ronde, met onder andere:
  - score\_value, score\_numeric, earnings, fedex\_points
  - position & position\_numeric
  - FK's naar de dimensietabellen

#### Dimensietabellen

- **dim\_player**: spelersinfo (first\_name, last\_name, VolledigeNaam)
- **dim\_tournament**: toernooienamen
- **dim\_location**: stad, staat en golfclub
- **dim\_date**: start\_date, end\_date, season

Alle relaties zijn correct gelegd via 1:N-verbanden met de feitentabel.



Figuur 26 Volledig Power BI model

## § 2. MEASURES EN DAX-LOGICA

In dit project werd intensief gebruik gemaakt van DAX-measures om diepgaande inzichten te creëren op basis van het stervormige datamodel. Alle measures zijn logisch gegroepeerd binnen de tabel MeasureLabels (Measures is een beschermd woord en kon niet gebruikt worden), wat zorgt voor overzicht en een scheiding tussen ruwe data (in tabellen) en berekende inzichten.

DAX (Data Analysis Expressions) vormt het berekeningsmechanisme van Power BI en wordt gebruikt om:

- KPI's dynamisch te berekenen op basis van de context (filters, slicers, tijd)
- Spelers met elkaar te vergelijken via virtuele relaties
- Resultaten per ronde, per seizoen of per locatie op te splitsen
- Ranking, filtering en logica toe te passen via measures

De measures kunnen functioneel worden onderverdeeld in volgende categorieën:

### a. Scorestatistieken

- GemiddeldeStrokes  
→ Berekent de gemiddelde score\_numeric van de geselecteerde speler in de context
- BesteNumeriekePositie  
→ Toont de laagste behaalde ranking ( $\text{MIN}(\text{position\_numeric})$ )
- LaagsteScoreOpGolfclub  
→ Meet de minimumscore per locatie en speler

### b. Verdiensten en efficiëntie

- Earnings  
→ Totale verdiensten in geselecteerde context ( $\text{SUM}(\text{earnings})$ )
- DollarPerStroke  
→  $\text{SUM}(\text{earnings}) / \text{SUM}(\text{score\_numeric})$ , als maat voor economische efficiëntie
- Career Earnings Difference %  
→ Verschil tussen geselecteerde speler en vergeleken speler, uitgedrukt in percentage

### c. Participatie en frequentie

- GespeeldeToernooien  
→ Aantal unieke toernooien per speler ( $\text{DISTINCTCOUNT}(\text{tournament\_id})$ )
- AantalRondes  
→ Totale aantal gespeelde rondes ( $\text{COUNTROWS}(\text{fact\_score\_round})$ )
- AantalToernooienPerSpelerPerClub  
→ Gesegmenteerde deelname op locatie-niveau

### d. Cut & success metrics

- AantalGemisteCuts  
→ Aantal ronden met status\_code = CUT
- CutSuccessRate  
→ Percentage toernooien waarin speler **wél door de cut ging**

### e. Seizoensspecifieke analyses

- FedExCupWinner  
→ Dynamische ranking op basis van  $\text{SUM}(\text{fedex\_points})$  binnen een seizoen
- TotaalFedExPuntenSeizoen  
→ Opgebouwde seizoensscore

f. Vergelijkende KPI's via disconnected slicer

Dankzij de Player Compare-tabel worden speciale measures geactiveerd zoals:

- DollarPerStroke Compare Player
- Earnings Compare Player
- FedEx Points Difference %
- BestePositie Vergelijking  
→ Deze gebruiken SELECTEDVALUE('Player Compare'[VolledigeNaam]) in combinatie met CALCULATE, FILTER, REMOVEFILTERS, TREATAS, enz.

g. Overzicht van meest gebruikte DAX-technieken

<b>DAX-techniek</b>	<b>Toepassing in het project</b>
<i>CALCULATE</i>	<b>Context switch</b> voor filtering, bv. totale earnings van een speler binnen een specifiek seizoen of golfclub
<i>REMOVEFILTERS</i>	Verwijdert slicerfilters (bv. om PGA Tour gemiddelden te tonen als benchmark)
<i>SELECTEDVALUE</i>	Detecteert een geselecteerde speler of compare-speler uit slicer of table visual
<i>FILTER + ALL</i>	Maakt het mogelijk om rankings of percentielvergelijkingen te bouwen buiten de huidige context
<i>DIVIDE</i>	Voert veilige delingen uit zonder fouten bij null of 0 (bv. earnings per stroke)
<i>TOPN</i>	Selecteert top X toernooien of spelers op basis van KPI's zoals earnings
<i>SWITCH</i>	Wordt gebruikt voor logische categorisaties (bv. score bins, ranges)
<i>TREATAS</i>	Maakt het mogelijk om waarden uit een niet-gerelateerde tabel (bv. Player Compare) tijdelijk te behandelen alsof ze afkomstig zijn uit een gerelateerde dimensie zoals dim_player, zonder dat er een fysieke relatie bestaat.

Tabel 27 Meest gebruikte DAX technieken

### § 3. DISCONNECTED TABLE VOOR VERGELIJKENDE ANALYSES

In dit project werd een disconnected table toegevoegd met als doel vergelijkende KPI-analyses mogelijk te maken tussen twee spelers, zonder het filtergedrag van het datamodel te verstoren.

#### Wat is een disconnected table?

Een disconnected table is een tabel die geen relatie heeft met het datamodel, maar via DAX expliciet wordt gebruikt in measures om virtueel te filteren. Ik heb dit toegepast op:

- Tabel: Player Compare
- Kolom: VolledigeNaam

Deze tabel bevat een unieke lijst van spelersnamen die de gebruiker kan selecteren via een slicer.

#### Waarom geen fysieke relatie?

Als deze tabel fysiek gekoppeld zou worden aan dim\_player, zou ze:

- alle visuals beïnvloeden bij selectie,
- de slicer “compare player” dezelfde context geven als de “gewone speler”,
- geen onafhankelijke vergelijking mogelijk maken.

Een disconnected table laat daarentegen toe om een speler te vergelijken met zichzelf of met een andere speler, zonder visuele conflicten.

Via de DAX-functie TREATAS() wordt de geselecteerde speler uit de compare-tabel virtueel gekoppeld aan de dim\_player-tabel:

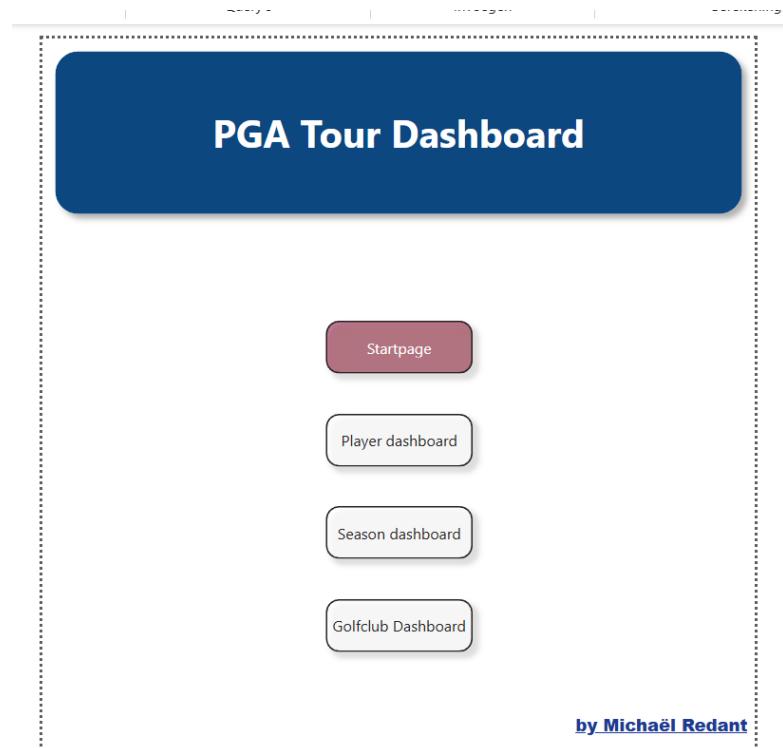
```
EARNINGS COMPARE PLAYER =
CALCULATE(
    [EARNINGS],
    TREATAS(
        VALUES('PLAYER COMPARE'[VOLLEDIGENAAM]),
        DIM_PLAYER[VOLLEDIGENAAM]
    )
)
```

Zo wordt er binnen de measure gerekend alsof er een relatie bestaat, terwijl het datamodel intact en modulair blijft.

## § 4. DASHBOARDS EN VISUALISATIES

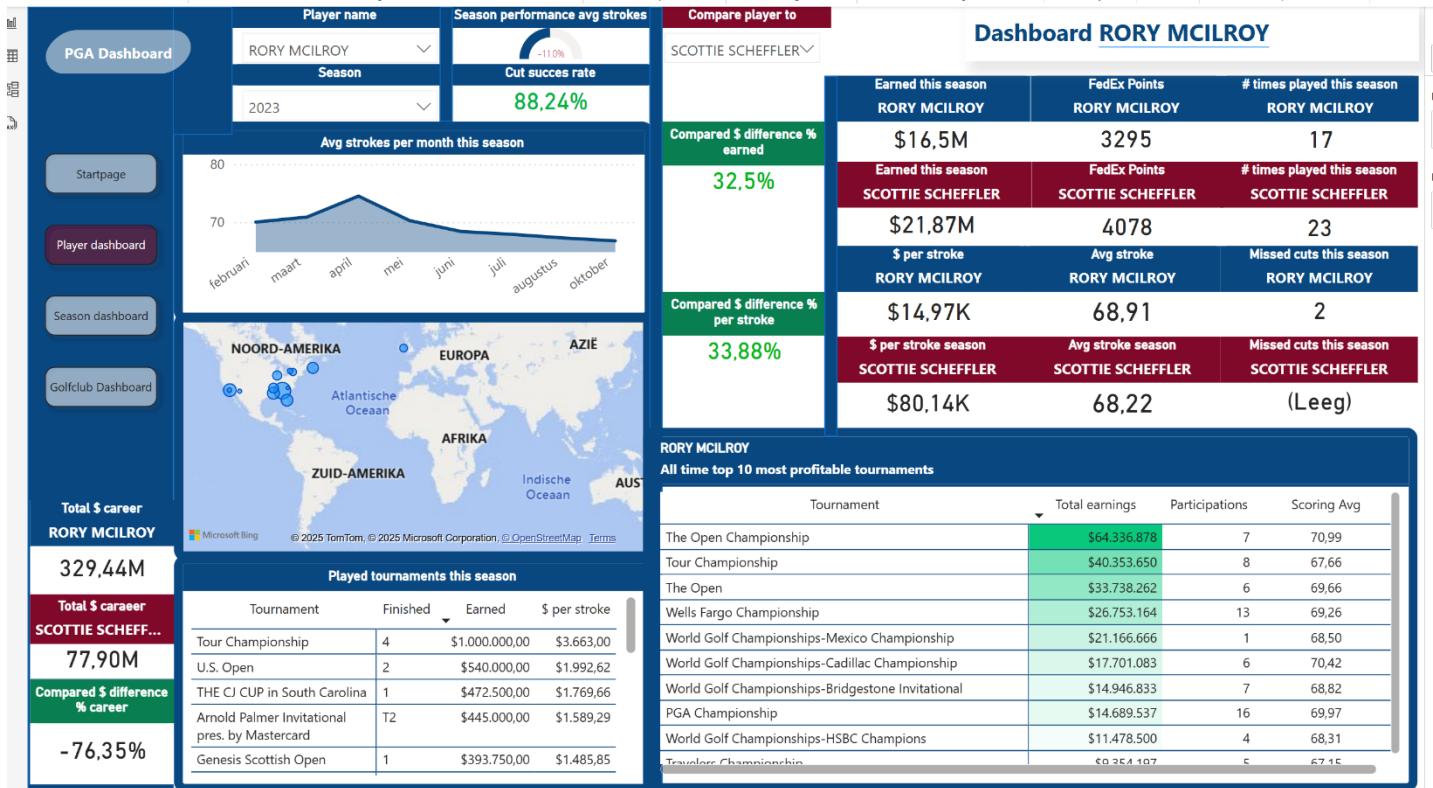
### a. Startpage

Om de gebruikerservaring te verbeteren en de navigatie doorheen het rapport logisch te structureren, werd een startpagina toegevoegd aan het Power BI-dashboard. Deze fungeert als visuele ingangspoort tot de verschillende analysethema's binnen het project.



*Figuur 27 Power BI Startpage*

### b. Player dashboard



Figuur 28 Power BI Player dashboard

Het Player Dashboard vormt het hart van het Power BI-rapport. Op deze pagina krijgt de gebruiker een volledig overzicht van de prestaties van een individuele speler binnen een geselecteerd seizoen, inclusief vergelijking met een andere speler.

Om de gebruikservaring verder te optimaliseren, werd de seisoens-slicer (uit dim\_date[season]) gesynchroniseerd over meerdere pagina's in het rapport.

- In dit project werd de slicer voor Season gedeeld tussen:
  - Player Dashboard
  - Season Dashboard

Dit zorgt ervoor dat de gebruiker het seizoen slechts één keer hoeft te selecteren, en dat alle andere dashboards automatisch volgen. Dit verhoogt de consistentie van de analyses en voorkomt verwarring.

Dit dashboard is ontworpen voor:

- Individuele speleraanalyse:** inzicht in vorm, verdiensten, efficiëntie
- Data-gedreven vergelijking:** wie is economisch of sportief sterker?
- Filterbare interactie:** analyse per seizoen, per speler

c. Season dashboard



Figuur 29 Power BI season dashboard

Het Season Dashboard biedt een breed overzicht van de prestaties in een specifiek PGA-seizoen. In tegenstelling tot het Player Dashboard ligt de focus hier niet op één speler, maar op het geaggregeerde niveau van het seizoen als geheel, met nadruk op verdeling, moeilijkheidsgraad en top performers.

Dit dashboard is ontworpen voor:

- Strategisch inzicht bieden op jaarbasis:
  - Welke maanden waren financieel sterk?
  - Welke toernooien waren zwaar?
  - Welke speler domineerde het veld?
- Zowel bruikbaar voor spelers als coaches

#### d. Golfclub dashboard



Figuur 30 Power BI golfclub dashboard

Het Golfclub Dashboard biedt inzicht in de prestaties van spelers op specifieke golfbanen. Via een filterbare interface kunnen gebruikers per club bekijken hoe moeilijk een locatie is, wie er het meest wint, hoe de scores evolueren, en wat de beste historische prestaties zijn.

Dit dashboard is ontworpen voor:

- Golfclubgericht inzicht in prestaties en moeilijkheid
- Herkenning van favoriete/best presterende spelers op specifieke locaties
- Historisch overzicht van topprestaties en extreme uitschieters

## 8. CONCLUSIE & REFLECTIE

Dit eindwerk bracht een complexe dataset van de PGA Tour, met resultaten van duizenden rondes tussen 2001 en 2024, terug tot een helder, analyseerbaar model. Het project doorliep de volledige data-analysecyclus: van ruwe import via SSIS en SQL-transformaties, tot een geoptimaliseerd Power BI-dashboard met interactieve inzichten per speler, seizoen en golfclub.

Wat dit project onderscheidt, is de combinatie van klassieke data engineeringtechnieken met AI-ondersteunde automatisatie en validatie. Tijdens de ontwikkeling werd gebruik gemaakt van een eigen op maat getrainde GPT, waarin meerdere handboeken en cursusmateriaal werden verwerkt rond:

- SQL-transformaties en stagingpraktijken
- ETL-design en dimensioneel modelleren (Kimball)
- Power BI en DAX-measure structuren

Dankzij deze custom GPT kon ik op elk moment vragen stellen, formules genereren, en scripts of validaties laten controleren op syntaxis en logica. Hierdoor werd de kwaliteit van de gebruikte query's, DAX-measures en transformaties naar een hoger niveau getild, met aandacht voor performantie, consistentie en best practices.

- Earnings per stroke bleek een zeer effectieve KPI om rendabele spelers te identificeren, los van het absolute prijzengeld.
- Het combineren van verschillende tools (SQL Server, SSIS, Power BI, Python) levert een krachtige analyseketen op..
- De inzet van een AI-model als hulplijn toonde aan dat data-analyse ondersteund door taalmodellen niet enkel efficiënt, maar ook kwaliteitsverhogend werkt.
- Door het stermodel zuiver op te bouwen én logisch te laden via lookup-transformaties in SSIS, werd het Power BI-dashboard zeer performant.
- Interactieve visualisaties met gesynchroniseerde slicers en KPI-comparaties maken het rapport niet alleen informatief, maar ook toegankelijk voor eindgebruikers

Dit project toont aan dat een degelijke datamodelarchitectuur in combinatie met moderne analysetools een ruwe dataset kan omvormen tot inzichten. Door de brug te slaan tussen klassieke ETL, krachtige visualisatie én AI-ondersteunde automatisering werd niet enkel een eindwerk afgeleverd, maar ook een volwaardig BI-platform opgebouwd met praktische toepassingen in de sportwereld.

## LIJST VAN AFBEELDINGEN

Figuur 1 Conceptueel ERD diagram .....	12
Figuur 2 Logisch ERD diagram .....	13
Figuur 3 Fysiek ERD diagram .....	16
Figuur 4 leegmaken fact_score_round SSIS .....	33
Figuur 5 OLE DB Source file connection manager .....	34
Figuur 6 OLE DB Source file mappings .....	34
Figuur 7 Derived Column DC_Player .....	35
Figuur 8 Lookup Player Connection manager .....	36
Figuur 9 Lookup Player column mapping .....	37
Figuur 10 Player error file mapping .....	38
Figuur 11 Derived Column Tournament .....	39
Figuur 12 Lookup Tournament connection manager .....	40
Figuur 13 Lookup Tournament column mapping .....	40
Figuur 14 Tournament error file mapping .....	41
Figuur 15 Lookup Date connection manager .....	42
Figuur 16 Lookup Date column mapping .....	43
Figuur 17 Lookup Location connection .....	44
Figuur 18 Lookup location column mapping .....	44
Figuur 19 Location error file mapping .....	45
Figuur 20 Sort column mapping .....	46
Figuur 21 Werkend flow sequence container .....	47
Figuur 22 werkende flow Load_FactScoreRound .....	48
Figuur 23 loading time python script .....	57
Figuur 24 resultaat python script(1) .....	57
Figuur 25 resultaat python script (2) .....	57
Figuur 26 Volledig Power BI model .....	58
Figuur 27 Power BI Startpage .....	61
Figuur 28 Power BI Player dashboard .....	62
Figuur 29 Power BI season dashboard .....	63
Figuur 30 Power BI golfclub dashboard .....	64

## LIJST VAN TABELLEN

Tabel 1 Ovezicht dataset.....	8
Tabel 2 Ruwe dataset .....	9
Tabel 3 Datatypes ruwe dataset.....	10
Tabel 4 Golf-techisch jargon.....	11
Tabel 5 dim_player .....	14
Tabel 6 dim_location .....	14
Tabel 7 dim_tournament .....	14
Tabel 8 dim_date.....	15
Tabel 9 fact_score_round.....	15
Tabel 10 RDM - dim_player .....	17
Tabel 11 RDM - dim_location .....	17
Tabel 12 RDM - dim_tournament.....	17
Tabel 13 RDM - dim_date .....	17
Tabel 14 RDM - fact_score_round .....	18
Tabel 15 resultaten splitsing query SSMS .....	20
Tabel 16 resultaten splitsing locatie SSMS.....	21
Tabel 17 Rollen van de tabellen ETL .....	28
Tabel 18 Dimension tabellen overzicht.....	30
Tabel 19 Datastream ETL.....	31
Tabel 20 ETL uitdagingen in SSIS .....	32
Tabel 21 ETL uitdagingen in Power BI .....	33
Tabel 22 Input & lookup kolom player.....	36
Tabel 23 Input & Lookup kolom tournament .....	39
Tabel 24 Input & Lookup kolom date.....	42
Tabel 25 Input & lookup kolom location.....	43
Tabel 26 Input & doelkolom facts table.....	47
Tabel 27 Meest gebruikte DAX technieken.....	60