In this session, I am recreating the scd lecture step to step to understand it more deeply. This covers day2 of dimensional data modelling.

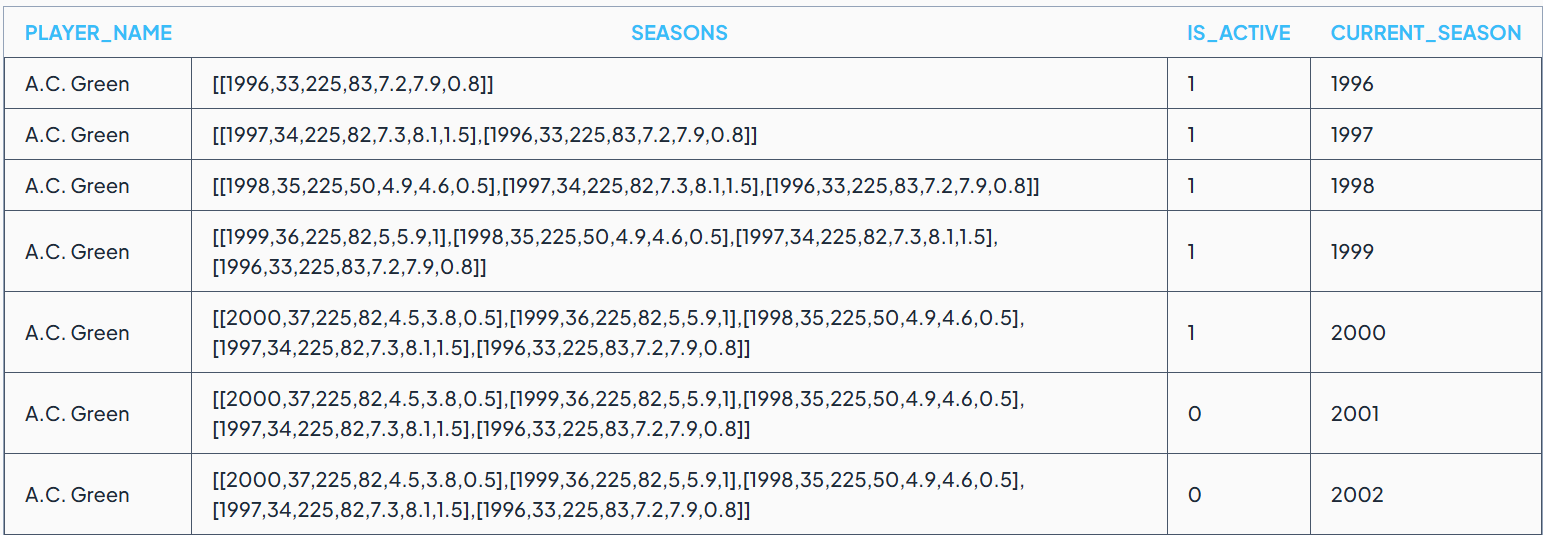
A slowly changing dimension can be modelled in different ways, but we are choosing the best way, SCD type 2 which maintains the history of the dimension from start to finish and hence, idempotent.

Recap from the first session, we created Prithvi.nba\_players, a cumulative table.

SELECT player\_name,seasons,is\_active,current\_season FROM prithvi.nba\_players

order by player\_name, current\_season

LIMIT 50



If you look at a sample entry for ac green, seasons, is\_active are the changing columns. We would like to understand for a particular player, how his dimensions are changing with time, here season.

Seasons column changes every season, changes very often and could be modelled using yearly snapshot. It does not fit the criteria for how often the dimension is changing. It should be slowly changing.

Now, is\_active column for A.C.Green is changing only twice for his entire history of his time as a player. It fits the bill for scd, so, we chose is\_active to scd.

The final result should have 2 rows for A.C. Green, with 1 row’s is\_active column set to 1 and another row’s set to 0. We would need two additional columns: start\_season and end\_season to track the starting and ending point for is\_active column. These two columns help maintain the entire history.

We will start with creating a schema for implementing this scd.

Schema:

CREATE TABLE prithvi.nba\_player\_scd (

player\_name VARCHAR,

is\_active INTEGER,

start\_season INTEGER,

end\_season INTEGER,

current\_season INTEGER

)

WITH (

format ='PARQUET',

partitioning = ARRAY['current\_season']

)

Player\_name – the dimension which is changing over time

Is\_active – the column which changes slowly over time

Start\_season- the column which maintains the starting season when the is\_active is set to current value

End\_season- the column which maintains the ending season when the is\_active is set to current\_value. The is\_active flag changes next season.

Current\_season- the variable which tracks the time as season rolls on.

SCD can be implemented either in 1 load query or incrementally. Lets do the 1 load query first.

1 load Thought process:

First, lets see whether if the player was active last season or not. We use lag function to add an additional column is\_active\_last\_season. If there is a change we now know that its changed from its previous season. This is of utmost importance for implementing this scd on is\_active column.

SELECT player\_name,

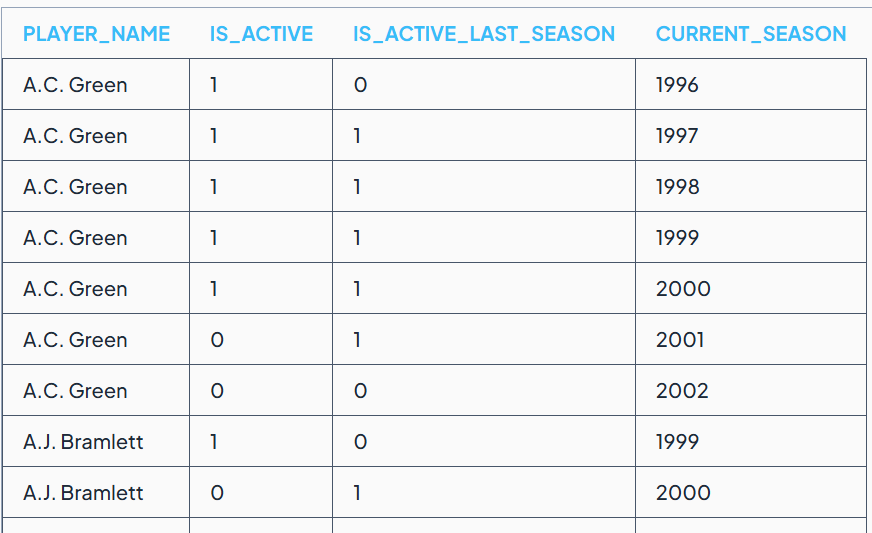
is\_active,

COALESCE(LAG(is\_active,1) OVER(PARTITION BY player\_name ORDER BY current\_season),0) as is\_active\_last\_season,

current\_season

FROM prithvi.nba\_players

The coalesce statement handles the nulls for the very first season.



Add a new column called did\_change to track the difference between the present and previous season.

WITH lagged AS(

SELECT player\_name,

is\_active,

COALESCE(LAG(is\_active,1) OVER(PARTITION BY player\_name ORDER BY current\_season),0) as is\_active\_last\_season,

current\_season

FROM prithvi.nba\_players

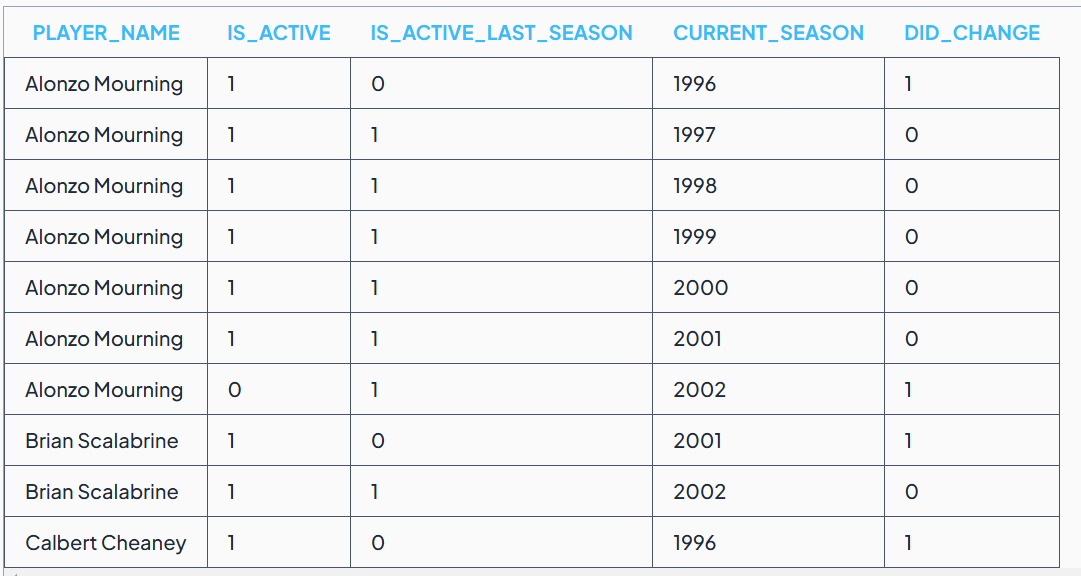
)

SELECT \*,

CASE WHEN

is\_active <> is\_active\_last\_season THEN 1 ELSE 0 END AS did\_change

FROM lagged limit 50



We now count the lifetime ‘did\_change’ for a particular player. We can do this by running sum over the did\_change column as follows:

WITH lagged AS(

SELECT player\_name,

is\_active,

COALESCE(LAG(is\_active,1) OVER(PARTITION BY player\_name ORDER BY current\_season),0) as is\_active\_last\_season,

current\_season

FROM prithvi.nba\_players

)

SELECT \*,

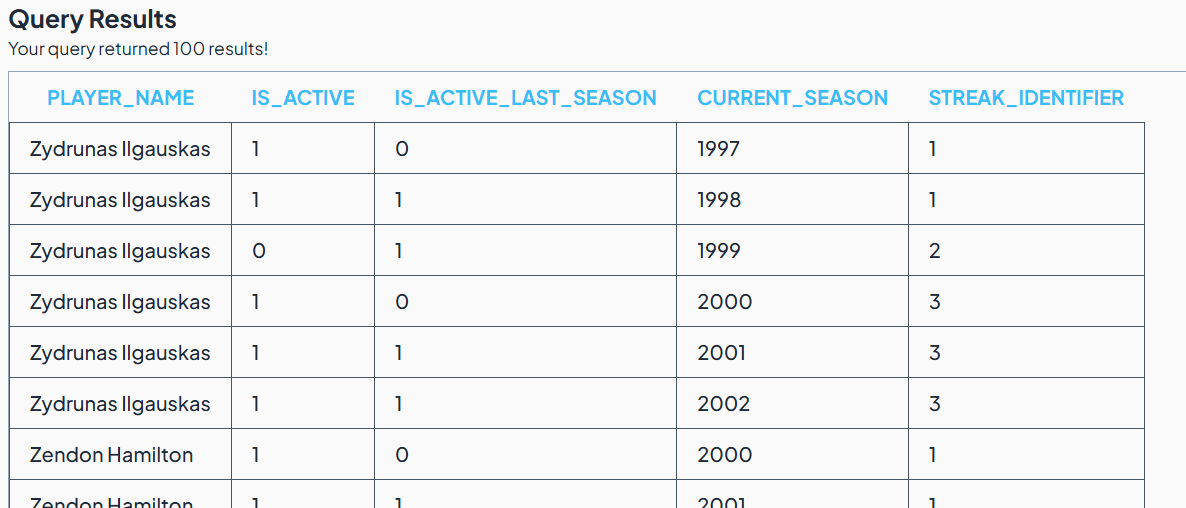
SUM(CASE WHEN

is\_active <> is\_active\_last\_season THEN 1 ELSE 0 END) OVER(PARTITION BY player\_name ORDER BY current\_season)AS streak\_identifier

FROM lagged

ORDER BY player\_name desc,current\_season

limit 100



The streak\_identifier column now accurately depicts the players behaviour vs time. If there’s a change it increments, if not, it stay the same.

The pair of values: (player\_name,streak\_identifier) are unique for each streak. In the above example:

* Zydrunas IIgauskas + 1 (streak\_identifier) indicates his streak of playing for the seasons of 1997-1998
* Zydrunas IIgauskas + 2 (streak\_identifier) indicates his inactive strek for year 1999
* Zydrunas IIgauskas + 3 (streak\_identifier) indicates his active streak from 2000 to 2002

We can write a query using group by player\_name,streak\_identifier to get the final scd table

WITH lagged AS(

SELECT player\_name,

is\_active,

COALESCE(LAG(is\_active,1) OVER(PARTITION BY player\_name ORDER BY current\_season),0) as is\_active\_last\_season,

current\_season

FROM prithvi.nba\_players

),

streaked AS(

SELECT \*,

SUM(CASE WHEN

is\_active <> is\_active\_last\_season THEN 1 ELSE 0 END) OVER(PARTITION BY player\_name ORDER BY current\_season)AS streak\_identifier

FROM lagged

)

SELECT player\_name,

MAX(is\_active) AS is\_active,

MIN(current\_season) as start\_season,

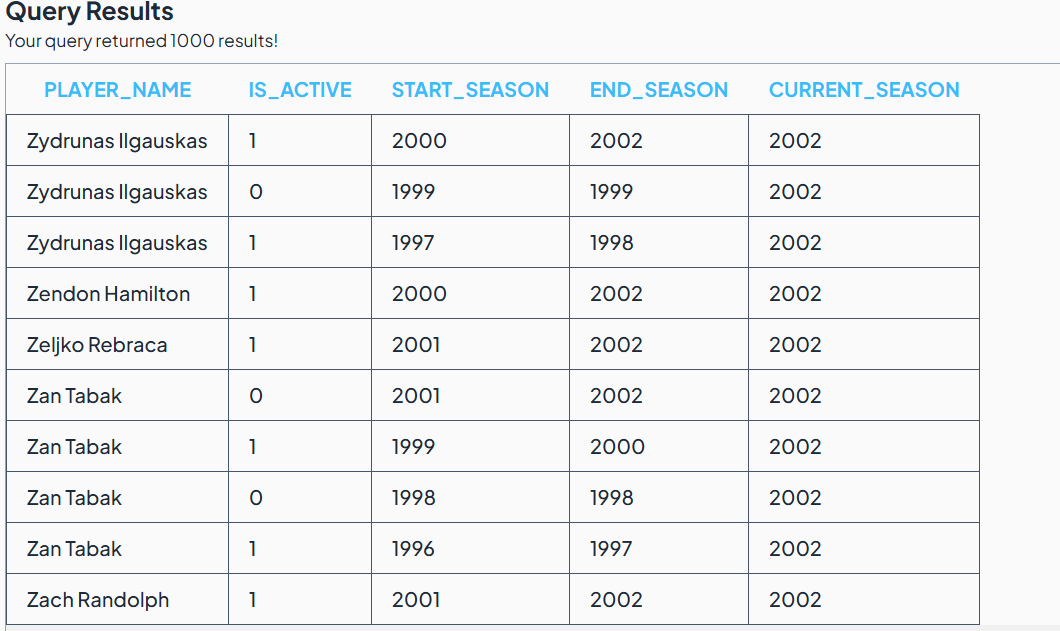
MAX(current\_season) as end\_season,

'2002' as current\_season

FROM streaked

GROUP BY player\_name,streak\_identifier

order by player\_name desc



Time to load the data into our earlier schema :Prithvi.nba\_player\_scd

Insert into Prithvi.nba\_player\_scd

WITH lagged AS(

SELECT player\_name,

is\_active,

COALESCE(LAG(is\_active,1) OVER(PARTITION BY player\_name ORDER BY current\_season),0) as is\_active\_last\_season,

current\_season

FROM prithvi.nba\_players

),

streaked AS(

SELECT \*,

SUM(CASE WHEN

is\_active <> is\_active\_last\_season THEN 1 ELSE 0 END) OVER(PARTITION BY player\_name ORDER BY current\_season)AS streak\_identifier

FROM lagged

)

SELECT player\_name,

MAX(is\_active) AS is\_active,

MIN(current\_season) as start\_season,

MAX(current\_season) as end\_season,

'2002' as current\_season

FROM streaked

GROUP BY player\_name,streak\_identifier

This is the 1-time load for the scd table, now let’s look at the incremental approach.

Incremental Load thought process

Lets load the data only till the season: 2001 and incrementally load the data of the season: 2002

Editing the above full load query as follows to recreate the scenario:

Insert into Prithvi.nba\_player\_scd

WITH lagged AS(

SELECT player\_name,

is\_active,

COALESCE(LAG(is\_active,1) OVER(PARTITION BY player\_name ORDER BY current\_season),0) as is\_active\_last\_season,

current\_season

FROM prithvi.nba\_players

WHERE current\_season<=2001

),

streaked AS(

SELECT \*,

SUM(CASE WHEN

is\_active <> is\_active\_last\_season THEN 1 ELSE 0 END) OVER(PARTITION BY player\_name ORDER BY current\_season)AS streak\_identifier

FROM lagged

)

SELECT player\_name,

MAX(is\_active) AS is\_active,

MIN(current\_season) as start\_season,

MAX(current\_season) as end\_season,

'2001' as current\_season

FROM streaked

GROUP BY player\_name,streak\_identifier

1278 rows inserted

Now the incremental logic:

WITH last\_season\_scd AS (

SELECT \* FROM prithvi.nba\_player\_scd

WHERE current\_season = 2001

),

current\_season\_scd AS (

SELECT \* FROM prithvi.nba\_players

WHERE current\_season = 2002

)

SELECT

COALESCE(ls.player\_name,cs.player\_name) AS player\_name ,

COALESCE(ls.start\_season,cs.current\_season) AS start\_season,

COALESCE(ls.end\_season,cs.current\_season) AS end\_season,

ls.is\_active as is\_active\_last\_season,

cs.is\_active as is\_active\_this\_season,

CASE WHEN ls.is\_active <> cs.is\_active THEN 1

WHEN ls.is\_active = cs.is\_active THEN 0 END AS did\_change,

2002 as current\_season

FROM last\_season\_scd ls

FULL OUTER JOIN current\_season\_scd cs

ON ls.player\_name=cs.player\_name

AND ls.end\_season+1 = cs.current\_season

Explanation for the above: We take the values from the last season scd, for this current season scd, we need to get the data from Prithvi.nba\_players for year 2002.

* COALESCE(ls.player\_name,cs.player\_name) AS player\_name

For the new players in season 2002, ls.player\_name is null, hence it takes cs.player\_name as the player\_name for new players

Similarly for the start\_season, end\_season for the new players in 2002.

* ls.is\_active as is\_active\_last\_season

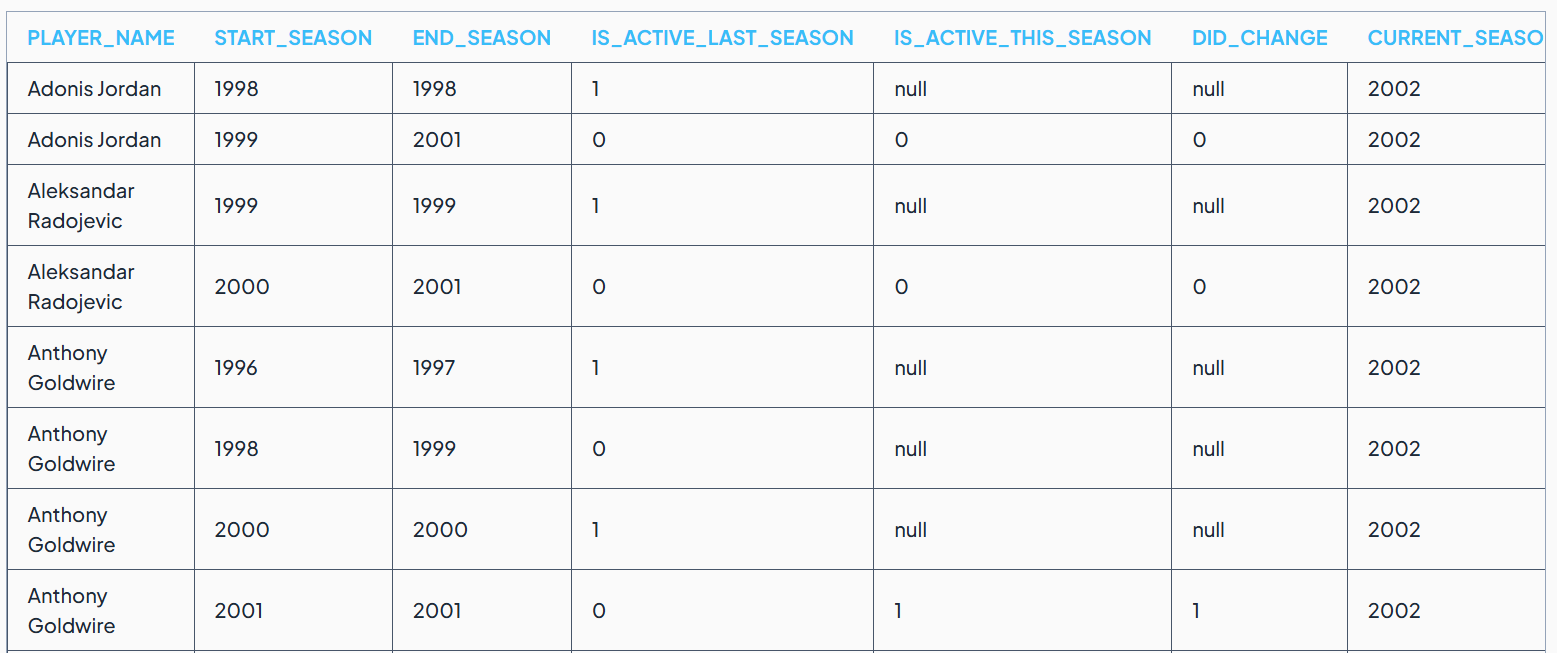
is\_active flag for the last season

* cs.is\_active as is\_active\_this\_season

is\_active flag for current season

* CASE WHEN ls.is\_active <> cs.is\_active THEN 1 WHEN ls.is\_active = cs.is\_active THEN 0 END AS did\_change

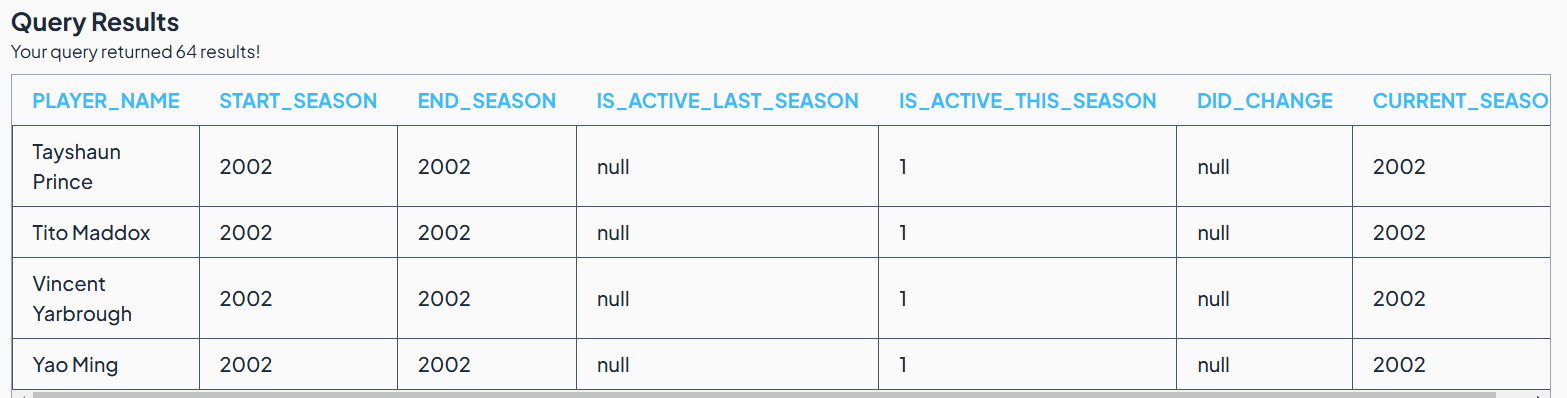
Did\_change flag to track if the is\_active flag is changing from 2001 to 2002



Three values for did\_change flag: 1,0,NULL.

NULL happens in 2 cases:

1. For a new player



1. For old players where end\_season is less than 2001. These records don’t change as it happened in the past

We will handle all the scenarios with did\_change column in the next step, by adding an array element as follows:

WITH last\_season\_scd AS (

SELECT \* FROM prithvi.nba\_player\_scd

WHERE current\_season = 2001

),

current\_season\_scd AS (

SELECT \* FROM prithvi.nba\_players

WHERE current\_season = 2002

),

combined as(

SELECT COALESCE(ls.player\_name,cs.player\_name) AS player\_name ,

COALESCE(ls.start\_season,cs.current\_season) AS start\_season,

COALESCE(ls.end\_season,cs.current\_season) AS end\_season,

ls.is\_active as is\_active\_last\_season,

cs.is\_active as is\_active\_this\_season,

CASE WHEN ls.is\_active <> cs.is\_active THEN 1

WHEN ls.is\_active = cs.is\_active THEN 0 END AS did\_change,

2002 as current\_season

FROM last\_season\_scd ls

FULL OUTER JOIN current\_season\_scd cs

ON ls.player\_name=cs.player\_name

AND ls.end\_season+1 = cs.current\_season

)

SELECT player\_name,

did\_change,

CASE

WHEN did\_change = 1

THEN ARRAY[ROW(is\_active\_last\_season,start\_season,end\_season),ROW(is\_active\_this\_season,current\_season,current\_season)]

WHEN did\_change = 0

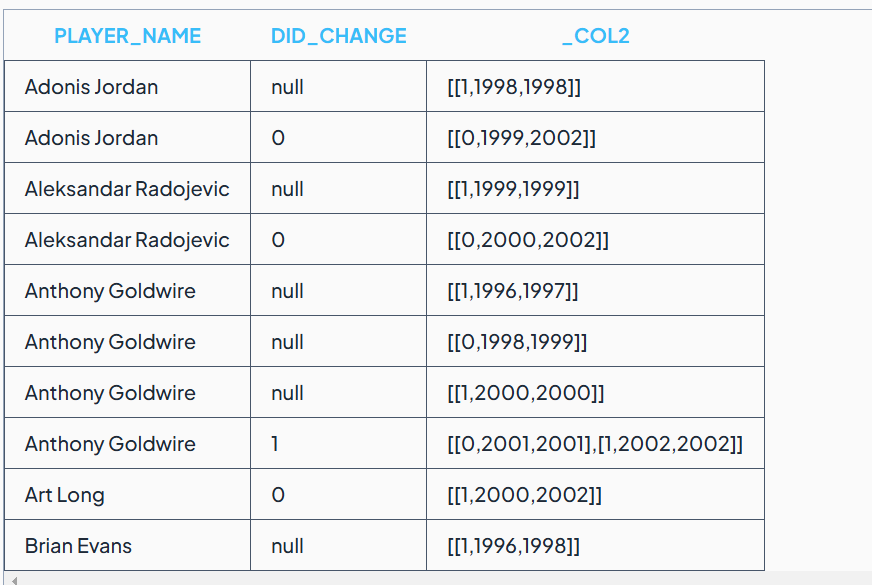
THEN ARRAY[ROW(is\_active\_last\_season,start\_season,end\_season+1)]

WHEN did\_change is NULL

THEN ARRAY[ROW(COALESCE(is\_active\_last\_season,is\_active\_this\_season),start\_season,end\_season)]

END

FROM combined



Final query:

INSERT INTO prithvi.nba\_player\_scd

WITH

last\_season\_scd AS (

SELECT

\*

FROM

prithvi.nba\_player\_scd

WHERE

current\_season = 2001

),

current\_season\_scd AS (

SELECT

\*

FROM

prithvi.nba\_players

WHERE

current\_season = 2002

),

combined AS (

SELECT

COALESCE(ls.player\_name, cs.player\_name) AS player\_name,

COALESCE(ls.start\_season, cs.current\_season) AS start\_season,

COALESCE(ls.end\_season, cs.current\_season) AS end\_season,

CASE

WHEN ls.is\_active <> cs.is\_active THEN 1

WHEN ls.is\_active = cs.is\_active THEN 0

END AS did\_change,

ls.is\_active AS is\_active\_last\_season,

cs.is\_active AS is\_active\_this\_season,

2002 AS current\_season

FROM

last\_season\_scd ls

FULL OUTER JOIN current\_season\_scd cs ON ls.player\_name = cs.player\_name

AND ls.end\_season + 1 = cs.current\_season

),

changes AS (

SELECT

player\_name,

current\_season,

CASE

WHEN did\_change = 0 THEN ARRAY[

CAST(

ROW(

is\_active\_last\_season,

start\_season,

end\_season + 1

) AS ROW(

is\_active integer,

start\_season integer,

end\_season integer

)

)

]

WHEN did\_change = 1 THEN ARRAY[

CAST(

ROW(is\_active\_last\_season, start\_season, end\_season) AS ROW(

is\_active integer,

start\_season integer,

end\_season integer

)

),

CAST(

ROW(

is\_active\_this\_season,

current\_season,

current\_season

) AS ROW(

is\_active integer,

start\_season integer,

end\_season integer

)

)

]

WHEN did\_change IS NULL THEN ARRAY[

CAST(

ROW(

COALESCE(is\_active\_last\_season, is\_active\_this\_season),

start\_season,

end\_season

) AS ROW(

is\_active integer,

start\_season integer,

end\_season integer

)

)

]

END AS change\_array

FROM

combined

)

SELECT

player\_name,

arr.is\_active,

arr.start\_season,

arr.end\_season,

current\_season

FROM

changes

CROSS JOIN UNNEST (change\_array) AS arr