



## WINDOW FUNCTIONS IN SQL

```
USE DATABASE DEMO_DATABASE;
```

```
DROP TABLE TOP_SCORERS;
```

```
--another way to insert data
```

```
CREATE OR REPLACE TABLE TOP_SCORERS AS
```

```
SELECT
```

```
    'James Harden' AS player,
```

```
    2335 AS points,
```

```
    2020 AS season
```

```
UNION ALL
```

```
(SELECT
```

```
    'Damian Lillard' AS player,
```

```
    1978 AS points,
```

```
    2020 AS season)
```

```
UNION ALL
```

```
(SELECT
```

```
    'Devin Booker' AS player,
```

```
    1863 AS points,
```

```
    2020 AS season)
```

```
UNION ALL
```

```
(SELECT
```

```
    'James Harden' AS player,
```

```
    2818 AS points,
```

```
    2019 AS season)
```

```
UNION ALL
```

```
(SELECT
```

```
    'Paul George' AS player,
```

```
    1978 AS points,
```

```
    2019 AS season)
```



## WINDOW FUNCTIONS IN SQL

UNION ALL

(SELECT

'Kemba Walker' AS player,

2102 AS points,

2019 AS season)

UNION ALL

(SELECT

'Damian Lillard' AS player,

2067 AS points,

2019 AS season)

UNION ALL

( SELECT

'Richard Bartner' AS player,

2067 AS points,

2019 AS season)

UNION ALL

(SELECT

'Devin Booker' AS player,

1700 AS points,

2019 AS season)

UNION ALL

(SELECT

'Paul George' AS player,

1033 AS points,

2020 AS season)

UNION ALL

(SELECT

'Kemba Walker' AS player,



## WINDOW FUNCTIONS IN SQL

1145 AS points,

2020 AS season)

UNION ALL

(SELECT

'Adam Gilchrist' AS player,

1145 AS points,

2020 AS season);

SELECT \* FROM TOP\_SCORERS ORDER BY SEASON;

SELECT distinct season FROM TOP\_SCORERS;

SELECT SEASON,

MAX(POINTS) AS MAXM\_PNTS,

MIN(POINTS) AS MINM\_PNTS

FROM TOP\_SCORERS

GROUP BY 1

ORDER BY 1;

--MAXM PNTS - 2,818 MINM PNTS - 1700 -- 2019

--MAXM PNTS - 2,335 MINM PNTS - 1033 - 2020

DESCRIBE TABLE AJ\_WINDOW\_DEMO;

---YEAR-OVER-YEAR CHANGE

SELECT DISTINCT

player,

FIRST\_VALUE(POINTS) OVER (ORDER BY SEASON DESC) AS first\_season\_2019, -- 2019



## WINDOW FUNCTIONS IN SQL

```
LAST_VALUE(POINTS) OVER (ORDER BY SEASON DESC) AS last_season_2020, --20209
((last_season_2020 - first_season_2019) / first_season_2019) * 100 AS PER_CHANGE
--(100 * ((LAST_VALUE(points) OVER (PARTITION BY player ORDER BY season ASC) -
FIRST_VALUE(points) OVER (PARTITION BY player ORDER BY season ASC)) / FIRST_VALUE(points)
OVER (PARTITION BY player ORDER BY season ASC))) AS per_change
FROM
TOP_SCORERS
ORDER BY 1;

SELECT DISTINCT
player,
FIRST_VALUE(POINTS) OVER (PARTITION BY PLAYER ORDER BY SEASON ) AS first_season,
LAST_VALUE(POINTS) OVER (PARTITION BY PLAYER ORDER BY SEASON ) AS last_season
FROM TOP_SCORERS
ORDER BY 1;

--We used FIRST_VALUE and LAST_VALUE to find the scores for each player in the
--earliest and most recent seasons of data.

-- Then we computed the percent difference using:

-- 100 * ((new value - old value) / old value) per_difference

--- How to get top 3 results for each group?

SELECT
season,
ROW_NUMBER() OVER (PARTITION BY season ORDER BY points DESC) AS
ROW_NUMBER_points_rank,
RANK() OVER (PARTITION BY season ORDER BY points DESC) AS RANK_points_rank,
```



## WINDOW FUNCTIONS IN SQL

```
DENSE_RANK() OVER (PARTITION BY season ORDER BY points DESC) AS  
DENSE_RANK_points_rank,  
  
    player,  
  
    points  
FROM  
TOP_SCORERS;
```

```
--153 ms - row_number  
--179 ms - rank  
-- 132 ms - dense_rank
```

```
SELECT  
  
    season,  
  
    --ROW_NUMBER() OVER (PARTITION BY season ORDER BY points DESC) AS  
ROW_NUMBER_points_rank,  
  
    --RANK() OVER (PARTITION BY season ORDER BY points DESC) AS RANK_points_rank,  
  
    DENSE_RANK() OVER (PARTITION BY season ORDER BY points DESC) AS  
DENSE_RANK_points_rank,  
  
    player,  
  
    points  
FROM  
TOP_SCORERS;
```

```
SELECT  
  
    *  
FROM  
  
(  
  
    SELECT  
  
        season,
```



## WINDOW FUNCTIONS IN SQL

```
ROW_NUMBER() OVER (PARTITION BY season ORDER BY points DESC) AS  
ROW_NUMBER_points_rank,  
  
RANK() OVER (PARTITION BY season ORDER BY points DESC) AS RANK_points_rank,  
  
DENSE_RANK() OVER (PARTITION BY season ORDER BY points DESC) AS  
DENSE_RANK_points_rank,  
  
player,  
  
points  
FROM  
  
TOP_SCORERS  
  
)  
WHERE  
  
(points_rank <= 3);
```

-- In this example, we used RANK to rank each player by points over each season.

-- Then we used a subquery to then return only the top 3 ranked players for each season.

-- How to find a running total?

```
select  
  
season,  
  
player,  
  
points,  
  
--SUM( <expr1> ) OVER ( [ PARTITION BY <expr2> ] [ ORDER BY <expr3> [ ASC | DESC ] [ <window_frame> ] ] )  
  
SUM(top_scorers.points) OVER (PARTITION BY player ORDER BY season ASC) AS  
running_total_points  
FROM  
  
TOP_SCORERS  
  
ORDER BY PLAYER ASC, SEASON ASC;
```



## WINDOW FUNCTIONS IN SQL

--To find the running total simply use SUM with an OVER clause where you specify your groupings (PARTITION BY),

-- and the order in which to add them (ORDER BY).

ANALYTICSWITHANAND