**ISM 6208: Data Warehousing**

**Assignment 2: Analytic SQL**

Ankita Patidar

Pranali Kanade

Pranjal Shrivastava

### **Query 1: Aggregations with CUBE and ROLLUP**

This query utilizes the CUBE extension within the FIN S&P data.

Query:

SELECT

sps.COMPANY, sps.GICS\_SECTOR, spesf.TRADE\_DATE,

ROUND(AVG(spf.VOLUME),2) AS Avg\_Vol

FROM

FIN.SP500\_STOCKS sps

INNERJOIN FIN.SP500\_EOD\_STOCK\_FACTS spesf

ON sps.TICKER\_SYMBOL = spesf.TICKER\_SYMBOL

WHERE

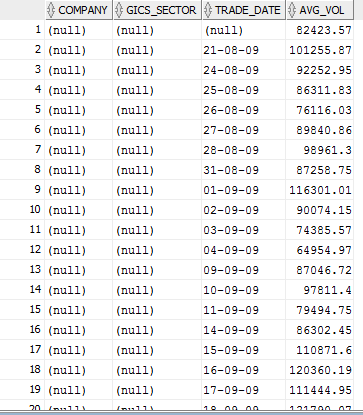
sps.GICS\_SECTOR IS NOT NULL

GROUP BY

CUBE(sps.COMPANY, sps.GICS\_SECTOR, sps.TRADE\_DATE);

The CUBE extension calculates all possible combinations of the COMPANY, GICS\_SECTOR, and TRADE\_DATE dimensions with the AVG TRADING VOLUME. In this circumstance, we can determine which companies/sectors traded substantially above or below its historical average and investigate what events occurred around that time to warrant such increased trading activity.

Output:



### **Query 2: Computing RANKs**

This query utilizes the RANK extension within the FRED\_FEDFUNDS data.

Query:

SELECT

FF\_DATE, FF\_YEAR, FF\_RATE,

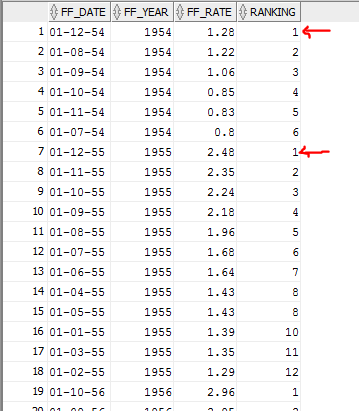
RANK() OVER (PARTITION BY FF\_YEAR ORDER BY FF\_RATE desc) AS RANKING

FROM

FIN.FRED\_FEDFUNDS;

When multiple rows have the same FF\_RATE, they are assigned the same rank but the next rank is not in sequence (like Olympic medaling).

Output:



This query utilizes the DENSE\_RANK extension with the FRED\_FEDFUNDS data.

Query:

SELECT

FF\_DATE, FF\_YEAR, FF\_RATE,

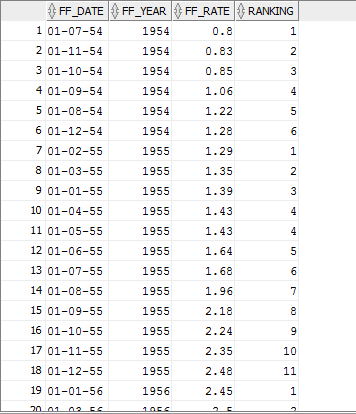
DENSE\_RANK() OVER (PARTITION BY FF\_YEAR ORDER BY FF\_RATE) AS RANKING

FROM

FIN.FRED\_FEDFUNDS;

Using DENSE\_RANK assigns consecutive ranks (not like Olympic medaling).

Output:



**Query 3: Creating Bins with NTILE**

This query utilizes the NTILE extension within the FIN S&P data.

SELECT FF\_DATE, FF\_YEAR, FF\_RATE,

NTILE(5) OVER (PARTITION BY FF\_YEAR ORDER BY FF\_RATE) AS quintile

FROM

FIN.FRED\_FEDFUNDS;

This query uses NTILE to create deciles that bin the different years. The following example divides the values in the year column of the fin.fred\_fedfunds table into 10 buckets. The year column has total 693 values, so the three extra values (the remainder of 693 / 10) are allocated to buckets 1, 2 and 3, which therefore have one more value than buckets 4-10.

SELECT

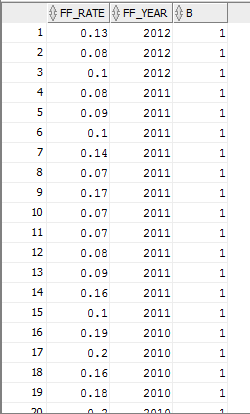
ff\_rate, ff\_year,

NTILE(10) OVER ( ORDER BY ff\_year DESC) AS bins

FROM

fin.fred\_fedfunds;

Output:



In this second query we grouped the data by year and used NTILE within the partitions to assign values.

SELECT

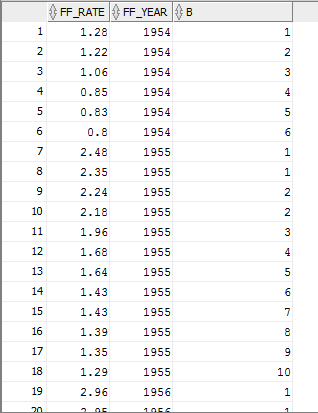
ff\_rate, ff\_year,

NTILE(10) OVER ( PARTITION BY ff\_year ORDER BY ff\_rate DESC ) AS b

FROM

fin.fred\_fedfunds;

Output:



**Query 5: Leading and Lagging Indicators**

This Query to select stock open, stock close, previous close(lag), Nextopen(lead) from sp500 data along with its impact on gdp, with previous gdp into account.

Query:

SELECT

trade\_date, close,

LAG(close, 1, 0) OVER (ORDER BY trade\_date ASC) AS prv\_cls, open AS wkly\_opn,

close AS wkly\_cls,

LEAD(open, 1, 0) OVER (ORDER BY trade\_date ASC) AS nxt\_opn, Gdp\_value,

LAG(Gdp\_value,1,0) OVER (ORDER BY trade\_date ASC) AS Prv\_gdp

from

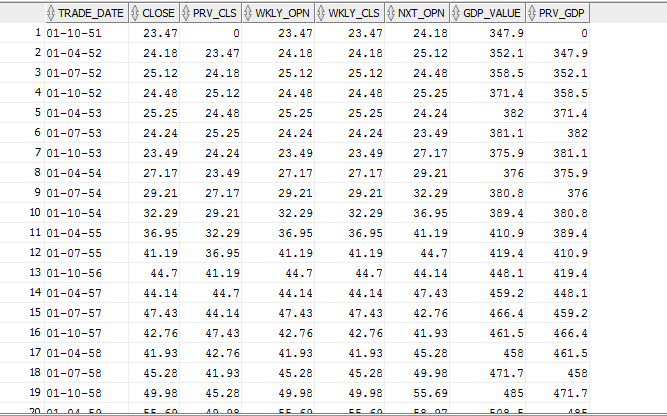
dberndt.sp500\_daily\_facts

INNER JOIN dberndt.fred\_gdp ON (dberndt.sp500\_daily\_facts.Trade\_date = dberndt.fred\_gdp.gdp\_date)

ORDER BY

trade\_date ASC;

Output:



**PART II: Interesting Queries**

**Query 1:**

This query is giving us insight about the revenue generated from different type of product category and their sub categories.

Here the column "Sales Overview" tells us whether there has been a profit or loss in the store.

Query:

SELECT

pdim.product\_subcategory, pdim.product\_category, COUNT(slsfct.order\_id) AS "Order Total", SUM (slsfct.Sales) AS "Sales Total", SUM(slsfct.profit) AS "Total profit or loss",

(CASE

WHEN SUM(slsfct.profit) > 0

THEN 'profit'

ELSE 'loss'

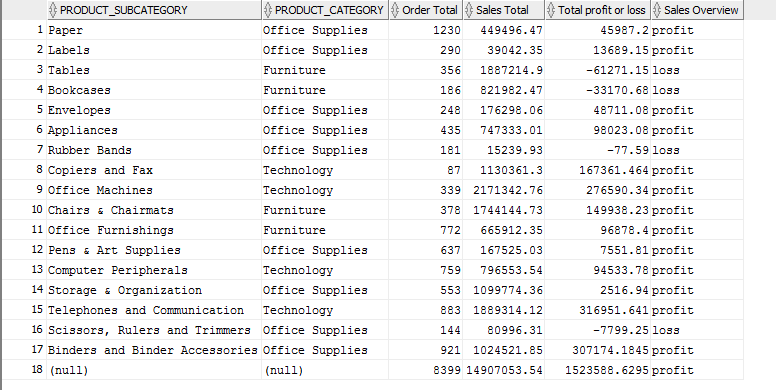
END) "Sales Overview"

from

SuperStore.Product\_Dim pdim INNER JOIN SuperStore.Sales\_Fact slsfct ON pd.product\_key = slsfct.product\_key

GROUP BY ROLLUP ((pdim.product\_subcategory, pdim.product\_category));

Output:



**Query 2:**

This query further expands upon the query created demonstrating RANK. It joins the fed funds rate table with the GDP table.

We then partition the RANK by year and order by GDP value. This can allow us to see if changes in the fed funds rate is a leading indicator to changes in GDP value.

Query:

SELECT

fd.FF\_DATE, fd.FF\_YEAR, fd.FF\_RATE, fg.GDP\_VALUE,

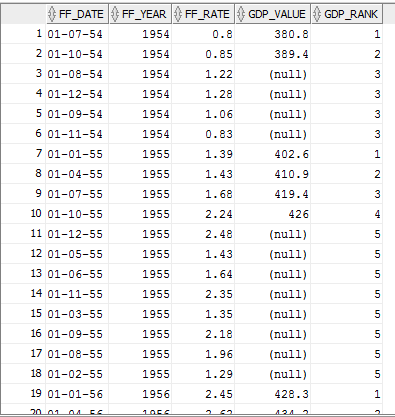
RANK() OVER (PARTITION BY fd.FF\_YEAR ORDER BY fg.GDP\_VALUE) AS GDP\_RANK

FROM

FIN.FRED\_FEDFUNDS fd

LEFT OUTER JOIN FIN.JULIAN\_DAYS jul\_dys ON fd.FF\_DATE = jul\_dys.ACTUAL\_DATE LEFT OUTER JOIN FIN.FRED\_GDP fg ON jul\_dys.JULIAN\_DAY = fg.JULIAN\_DAY;

Output:



**Query 3:**

This query will find the correlations between products and categories within sales.

Query:

SELECT

pdim.product\_key, pdim.product\_name,

EXTRACT(YEAR FROM sf.order\_date), RATIO\_TO\_REPORT(sf.order\_quantity\*sf.unit\_price) OVER(PARTITION BY pdim.product\_name) SALES\_RATIO

from

superstore.product\_dim pdim JOIN superstore.sales\_fact sf

ON pdim.product\_key = sf.product\_key;

Output:

