DATA WAREHOUSING

In--Class Dimensional Modeling and Analytical SQL Exercise
ISM6208.001S17

Professor : Dr. Donald Berndt

SUBMITTED BY :

GAURAV GIRISH KULKARNI

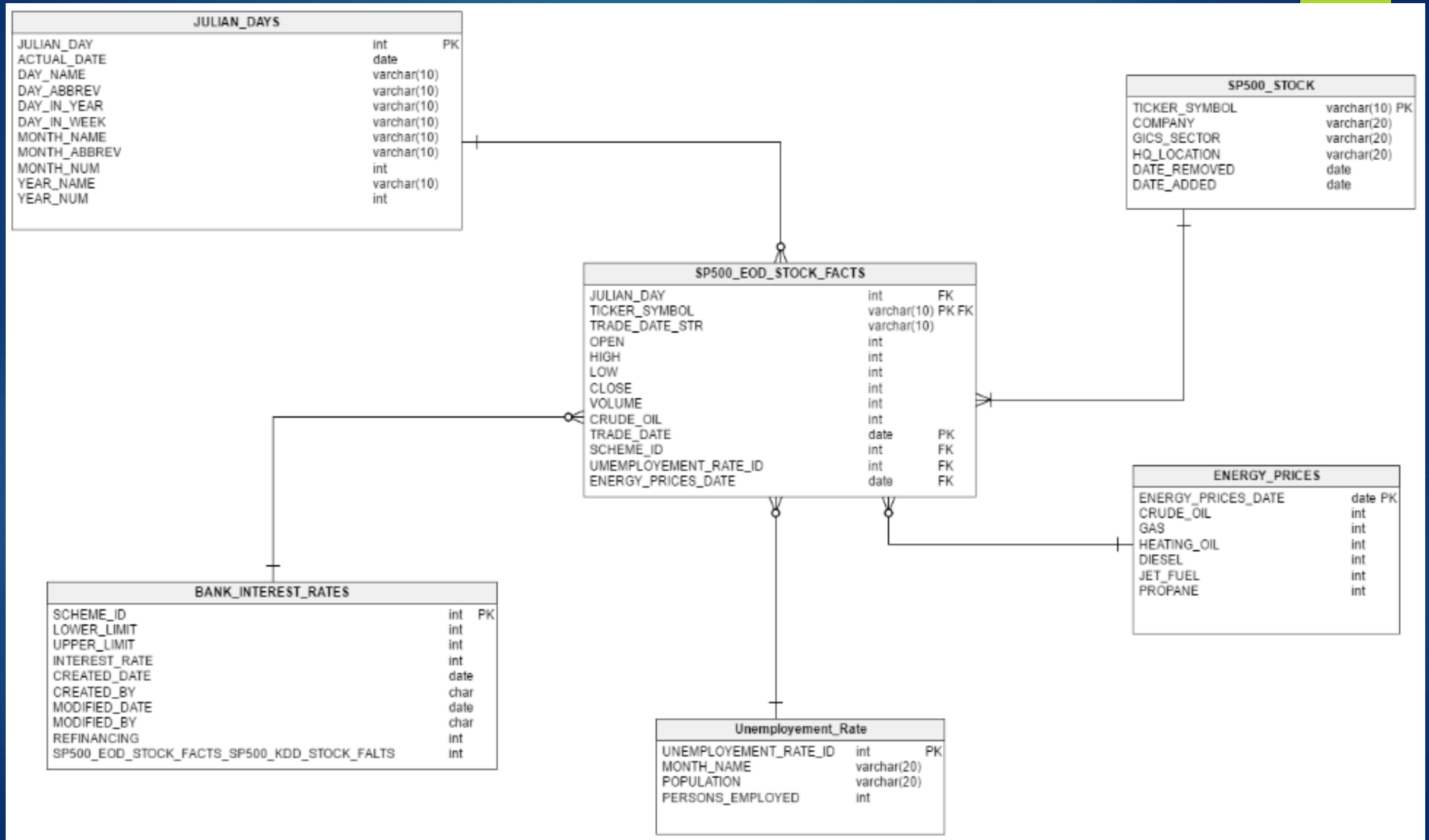
PRATUL SUNIL AGARWAL

SHIFA CHAUDHARY

SHRIRENGAPRIYA SREETHARAN

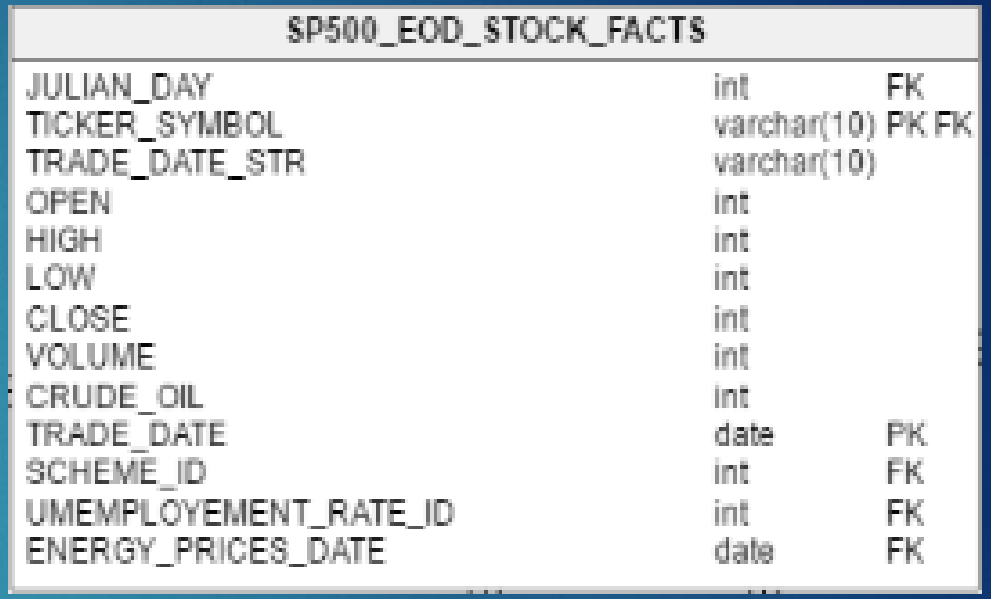
VAGEESHA MAIYA

ER DIAGRAM


JULIAN_DAYS
JULIAN_DAY int PK
ACTUAL_DATE date
DAY_NAME varchar(10)
DAY_ABBREV varchar(10)
DAY_IN_YEAR varchar(10)
DAY_IN_WEEK varchar(10)
MONTH_NAME varchar(10)
MONTH_ABBREV varchar(10)
MONTH_NUM int
YEAR_NAME varchar(10)
YEAR_NUM int
SP500_STOCK
TICKER_SYMBOL varchar(10) PK
COMPANY varchar(20)
GICS_SECTOR varchar(20)
HQ_LOCATION varchar(20)
DATE_REMOVED date
DATE_ADDED date
SP500_EOD_STOCK_FACTS
JULIAN_DAY int FK
TICKER_SYMBOL varchar(10) PK FK
TRADE_DATE_STR varchar(10)
OPEN int
HIGH int
LOW int
CLOSE int
VOLUME int
CRUDE_OIL int
TRADE_DATE date PK
SCHEME_ID int FK
UMEMPLOYEMENT_RATE_ID int FK
ENERGY_PRICES_DATE date FK
ENERGY_PRICES
ENERGY_PRICES_DATE date PK
CRUDE_OIL int
GAS int
HEATING_OIL int
DIESEL int
JET_FUEL int
PROPANE int
BANK_INTEREST_RATES
SCHEME_ID int PK
LOWER_LIMIT int
UPPER_LIMIT int
INTEREST_RATE int
CREATED_DATE date
CREATED_BY char
MODIFIED_DATE date
MODIFIED_BY char
REFINANCING int
SP500_EOD_STOCK_FACTS_SP500_KDD_STOCK_FALTS int
Unemployement_Rate
UNEMPLOYEMENT_RATE_ID int PK
MONTH_NAME varchar(20)
POPULATION varchar(20)
PERSONS_EMPLOYED int

Fact Table

- SP500_EOD_STOCK_FACTS :
- Fact table forms a Star Schema with five tables connecting dimensions such as JULIAN_DAYS, Unemployment_rate, BANK_INTEREST_RATES, ENERGY_PRICES, SP500_STOCK through their relevant primary key.



SP500_EOD_STOCK_FACTS		
JULIAN_DAY	int	FK
TICKER_SYMBOL	varchar(10)	PK FK
TRADE_DATE_STR	varchar(10)	
OPEN	int	
HIGH	int	
LOW	int	
CLOSE	int	
VOLUME	int	
CRUDE_OIL	int	
TRADE_DATE	date	PK
SCHEME_ID	int	FK
UMEMPLOYEMENT_RATE_ID	int	FK
ENERGY_PRICES_DATE	date	FK

Dimension Tables

- SP500_STOCK : Contains information about the stocks of a company with details about the headquarters, location and dates the stocks were added or removed.
- JULIAN_DAYS : tabulates date and years
- ENERGY_PRICES : table updates energy prices of various sectors on an everyday basis.
- BANK_INTEREST_RATES : variation in interest rates triggers fluctuation in stock prices.
- Unemployment_Rate : Unemployment Rate influences the economy of the United States. We have considered this attribute since the rate is declared by the government institution per month.

Assumptions

- The current scope is to track one stock exchange only.
- Server can support daily facts about each stock.
- Energy resources like Crude Oil, Propane, Diesel, Jet Fuel etc. play vital role in fluctuations of S&P Index Values.
- Values Mentioned in High, Low, Open, Close are dollar values.
- New external data doesn't erase history data.
- All prices and predictions are recorded in one currency, which does not account for rapid inflation or deflation

QUERY 1: Moving Averages

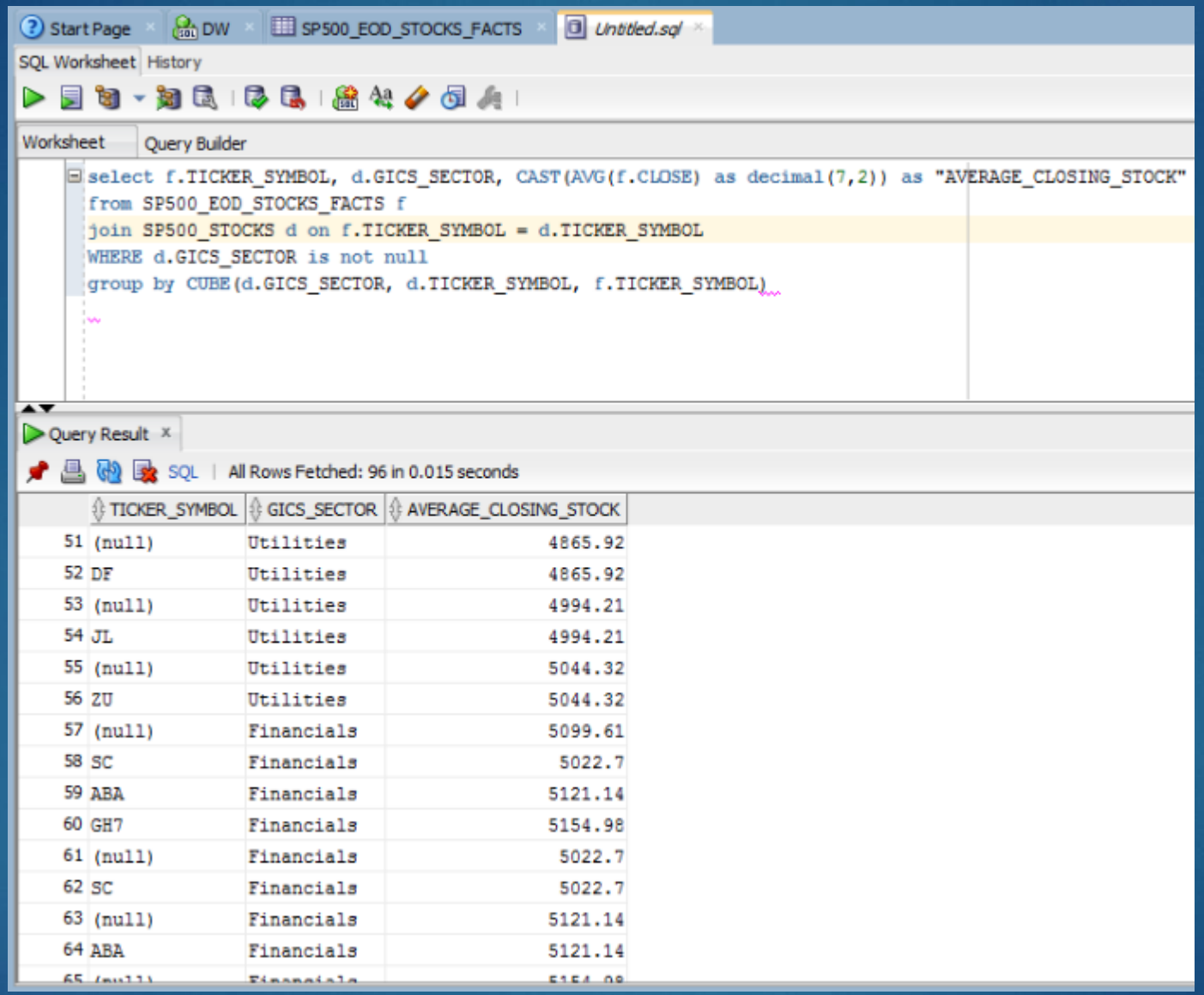
```
SELECT TRADE_DATE, TICKER_SYMBOL,
CAST(AVG(VOLUME) as DECIMAL(8,2)) OVERALL  VOL  MOVING  AVG,
CAST(AVG(VOLUME) OVER(ORDER BY TRADE_DATE ASC ROWS BETWEEN 7 PRECEDING AND
CURRENT ROW) AS DECIMAL(8,2)) 7DAY  VOL  MOVING  AVG
FROM SP500_EOD_STOCKS_FACTS
WHERE TICKER  SYMBOL like '%G%'
GROUP BY TRADE_DATE, TICKER_SYMBOL
ORDER BY TICKER  SYMBOL, TRADE  DATE ASC;
```

16	15-09-09	2009	GOOG	3	23986	21651.25	20222.57
17	16-09-09	2009	GOOG	3	25873	21899.59	21566.86
18	17-09-09	2009	GOOG	3	44834	23173.72	25829.14
19	18-09-09	2009	GOOG	3	32842	23682.58	27384.57
20	21-09-09	2009	GOOG	3	21175	23557.2	26787.71
21	22-09-09	2009	GOOG	3	30418	23883.9	28414.86
22	23-09-09	2009	GOOG	3	27046	24027.64	29453.43
23	24-09-09	2009	GOOG	3	25286	24082.35	29639.14
24	25-09-09	2009	GOOG	3	20520	23933.92	28874.43
25	28-09-09	2009	GOOG	3	18431	23713.8	25102.57
26	29-09-09	2009	GOOG	3	20993	23609.15	23409.86
27	30-09-09	2009	GOOG	3	31417	23898.33	24873
28	01-10-09	2009	GOOG	3	28162	24050.61	24550.71
29	02-10-09	2009	GOOG	3	26008	24118.1	24402.43
30	05-10-09	2009	GOOG	3	21264	24022.97	23827.86
31	06-10-09	2009	GOOG	3	27329	24129.61	24800.57
32	07-10-09	2009	GOOG	3	48776	24899.81	29135.57
33	08-10-09	2009	GOOG	3	43065	25450.27	32288.71

QUERY 2: Aggregations with CUBE and ROLLUP

- Compute for each sector and for each ticker symbol, the average value for the closing stock amount
- Use CUBE since we need the aggregate value for each ticker symbol and each sector

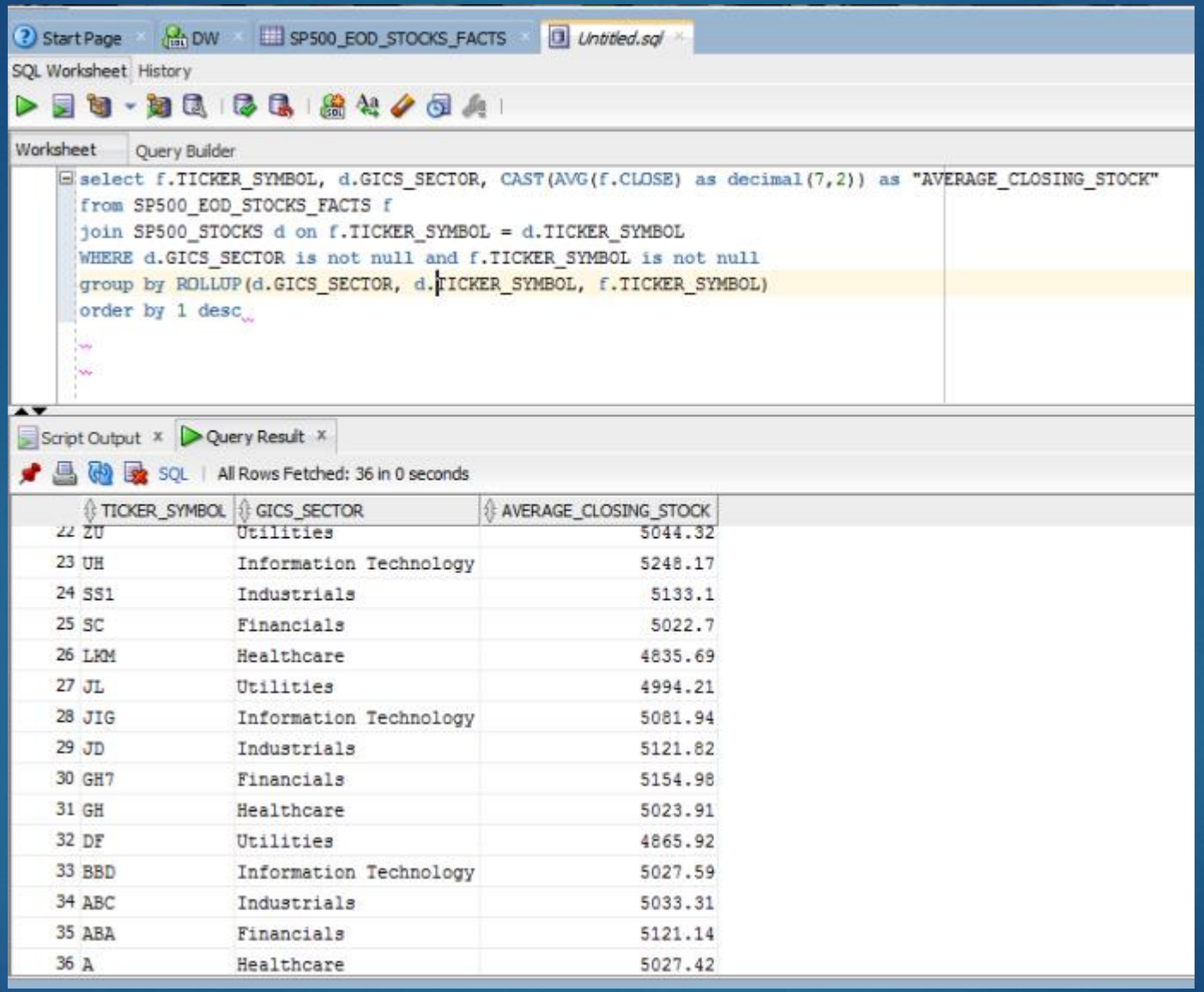
```
SELECT
d.TICKER_SYMBOL,
d.GICS_SECTOR,
AVG(f.CLOSE) "Avg. Closing Stock"
FROM  SP500_STOCK d,
SP500_EOD_STOCK_FACTS f
JOIN ON d.TICKER_SYMBOL=f.TICKER_SYMBOL
WHERE d.GICS_SECTOR IS NOT NULL
GROUP BY  CUBE (d.GICS_SECTOR, d.TICKER_SYMBOL)
```



	TICKER_SYMBOL	GICS_SECTOR	AVERAGE_CLOSING_STOCK
51	(null)	Utilities	4865.92
52	DF	Utilities	4865.92
53	(null)	Utilities	4994.21
54	JL	Utilities	4994.21
55	(null)	Utilities	5044.32
56	ZU	Utilities	5044.32
57	(null)	Financials	5099.61
58	SC	Financials	5022.7
59	ABA	Financials	5121.14
60	GH7	Financials	5154.98
61	(null)	Financials	5022.7
62	SC	Financials	5022.7
63	(null)	Financials	5121.14
64	ABA	Financials	5121.14
65	(null)	Financials	5154.98

- Calculate the overall average value of closing stock amount aggregated on a hierarchical fashion (using ROLLUP)

```
SELECT
d.TICKER_SYMBOL,
d.GICS_SECTOR,
AVG(f.OPEN)
FROM  SP500_STOCK d, SP500_EOD_STOCK_FACTS f
JOIN ON d.TICKER_SYMBOL=f.TICKER_SYMBOL
WHERE d.GICS_SECTOR IS NOT NULL
GROUP BY  ROLLUP (d.GICS_SECTOR, f.TICKER_SYMBOL
```

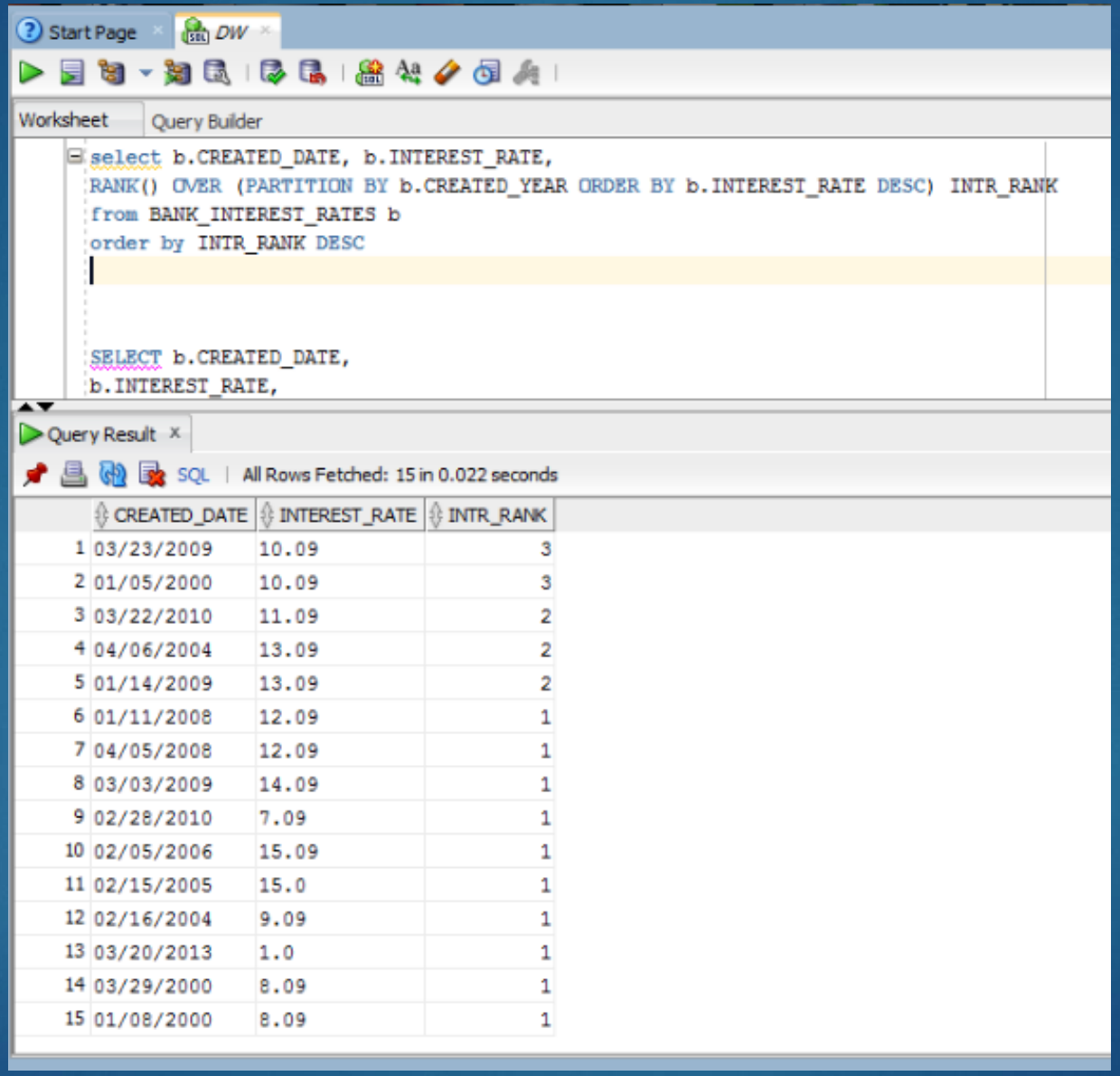


	TICKER_SYMBOL	GICS_SECTOR	AVERAGE_CLOSING_STOCK
22	ZU	Utilities	5044.32
23	UH	Information Technology	5248.17
24	SS1	Industrials	5133.1
25	SC	Financials	5022.7
26	LKM	Healthcare	4835.69
27	JL	Utilities	4994.21
28	JIG	Information Technology	5081.94
29	JD	Industrials	5121.82
30	GH7	Financials	5154.98
31	GH	Healthcare	5023.91
32	DF	Utilities	4865.92
33	BBD	Information Technology	5027.59
34	ABC	Industrials	5033.31
35	ABA	Financials	5121.14
36	A	Healthcare	5027.42

QUERY 3: Computing RANKS

- Rank the schemes based on the interest rates offered

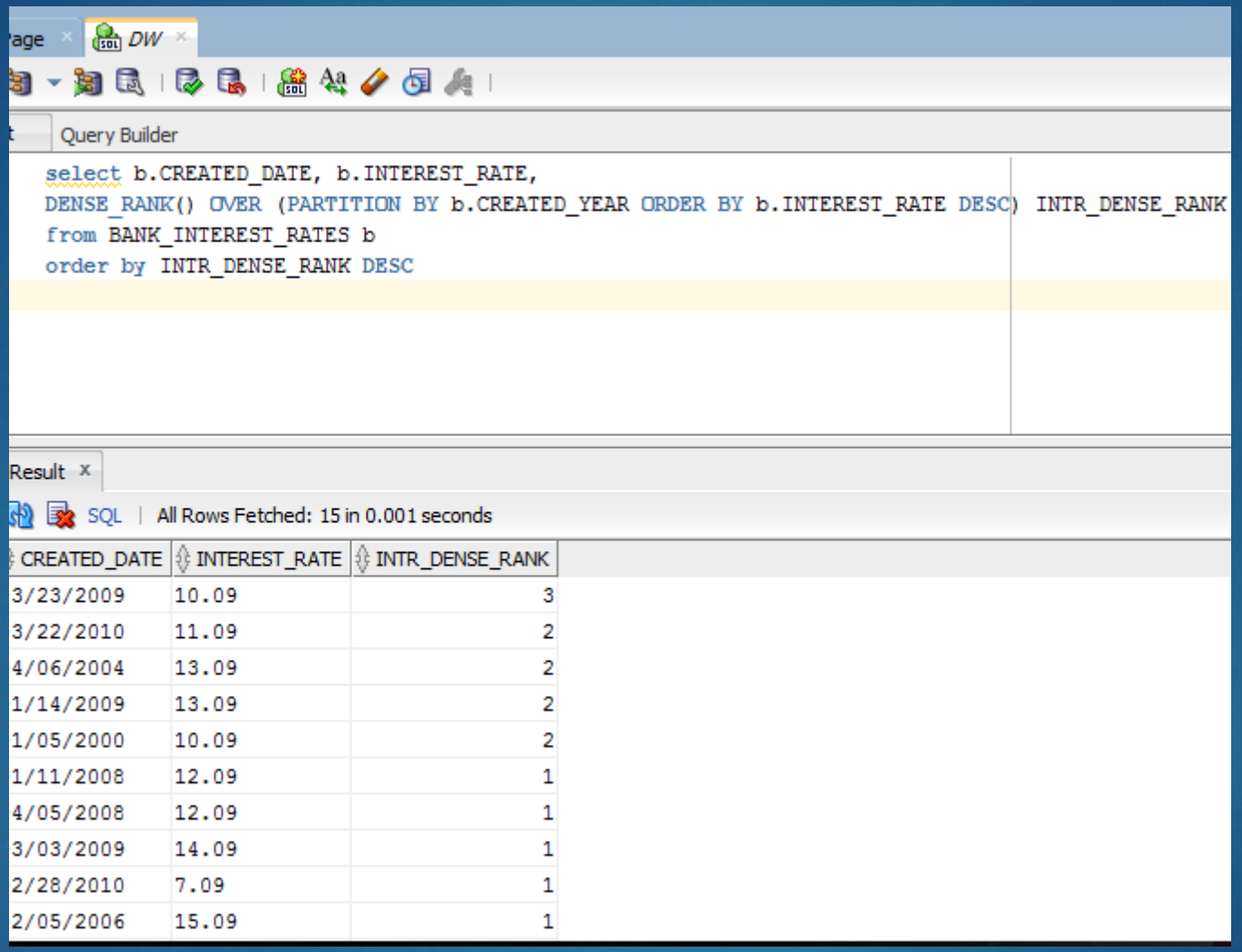
```
SELECT b.CREATED_DATE,
b.INTEREST_RATE,
RANK() OVER (PARTITION BY b.CREATED_DATE ORDER BY
b.INTEREST_RATE DESC) INTR_RANK
FROM  BANK_INTEREST_RATES b
ORDER BY FF_YEAR;
```



	CREATED_DATE	INTEREST_RATE	INTR_RANK
1	03/23/2009	10.09	3
2	01/05/2000	10.09	3
3	03/22/2010	11.09	2
4	04/06/2004	13.09	2
5	01/14/2009	13.09	2
6	01/11/2008	12.09	1
7	04/05/2008	12.09	1
8	03/03/2009	14.09	1
9	02/28/2010	7.09	1
10	02/05/2006	15.09	1
11	02/15/2005	15.0	1
12	02/16/2004	9.09	1
13	03/20/2013	1.0	1
14	03/29/2000	8.09	1
15	01/08/2000	8.09	1

- Rank the schemes based on interest rates offered but ensure the ranks are consecutive in order

```
SELECT b.CREATED_DATE,
b.INTEREST_RATE,
DENSE_RANK() OVER (PARTITION BY b.CREATED_DATE ORDER BY b.INTEREST_RATE DESC) INTR_DENSE_RANK
FROM  BANK_INTEREST_RATES b
ORDER BY FF_YEAR;
```



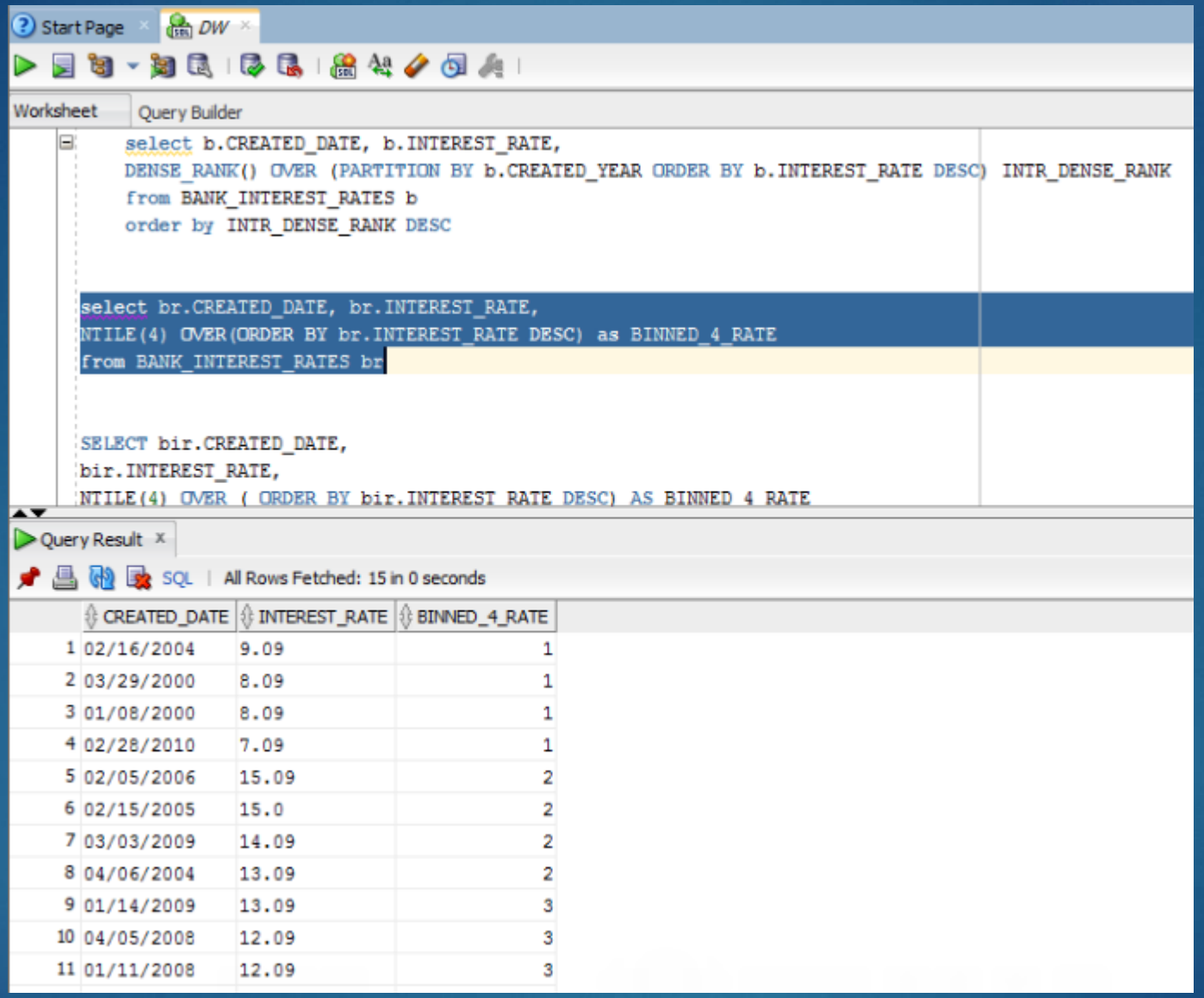
CREATED_DATE	INTEREST_RATE	INTR_DENSE_RANK
3/23/2009	10.09	3
3/22/2010	11.09	2
4/06/2004	13.09	2
1/14/2009	13.09	2
1/05/2000	10.09	2
1/11/2008	12.09	1
4/05/2008	12.09	1
3/03/2009	14.09	1
2/28/2010	7.09	1
2/05/2006	15.09	1

QUERY 4: CREATING BINS USING NTILE

- Use NTILE functions to bin interest rates offered

- Use QUARTILE to bin data into 4 different buckets:

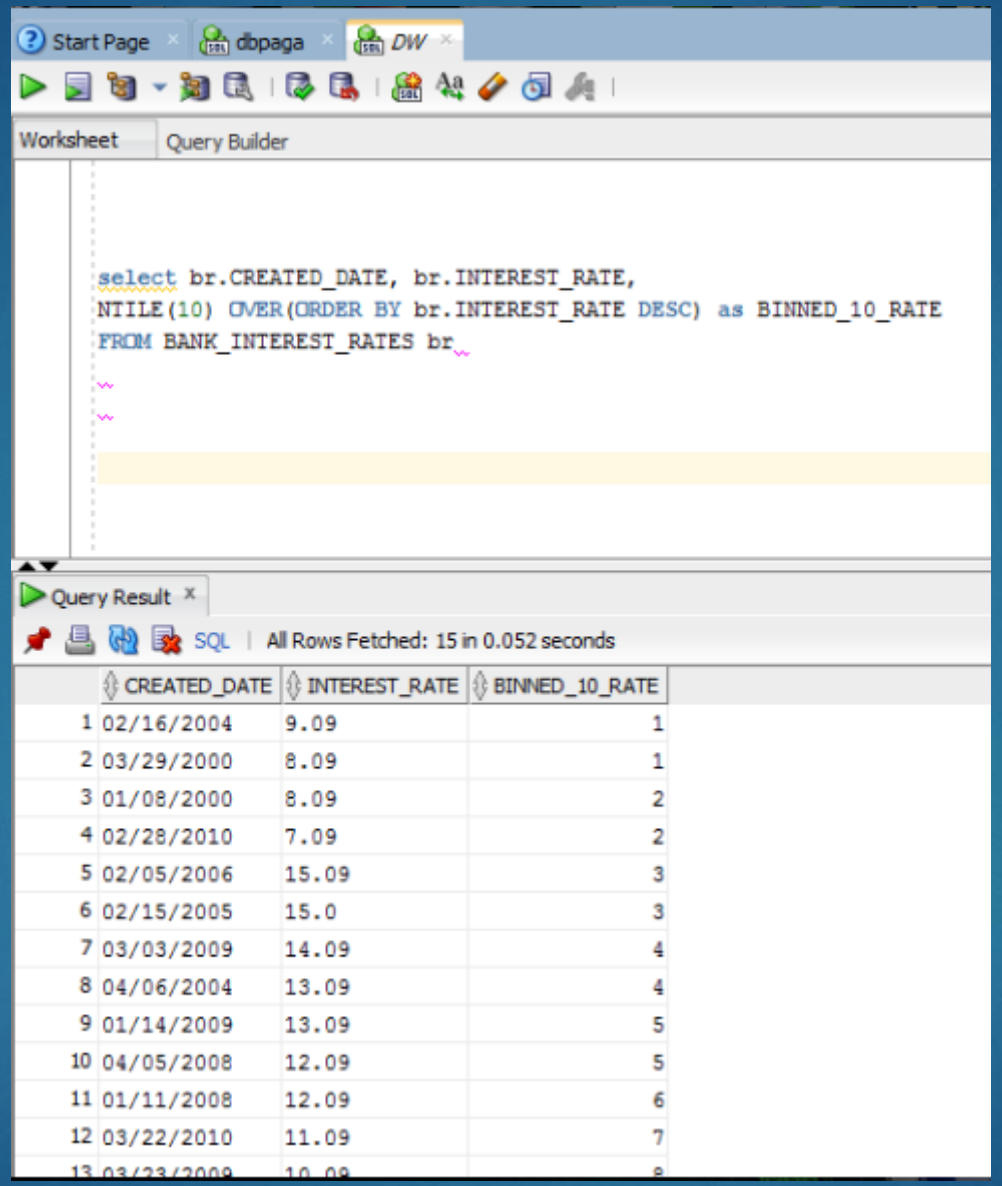
```
SELECT bir.CREATED_DATE,
bir.INTEREST_RATE,
NTILE(4) OVER ( ORDER BY bir.INTEREST_RATE DESC) AS BINNED_4_RATE
FROM  BANK_INTEREST_RATES bir;
```



	CREATED_DATE	INTEREST_RATE	BINNED_4_RATE
1	02/16/2004	9.09	1
2	03/29/2000	8.09	1
3	01/08/2000	8.09	1
4	02/28/2010	7.09	1
5	02/05/2006	15.09	2
6	02/15/2005	15.0	2
7	03/03/2009	14.09	2
8	04/06/2004	13.09	2
9	01/14/2009	13.09	3
10	04/05/2008	12.09	3
11	01/11/2008	12.09	3

- Use NTILE and bucket the records in 10 different bins in their decreasing order of interest rates

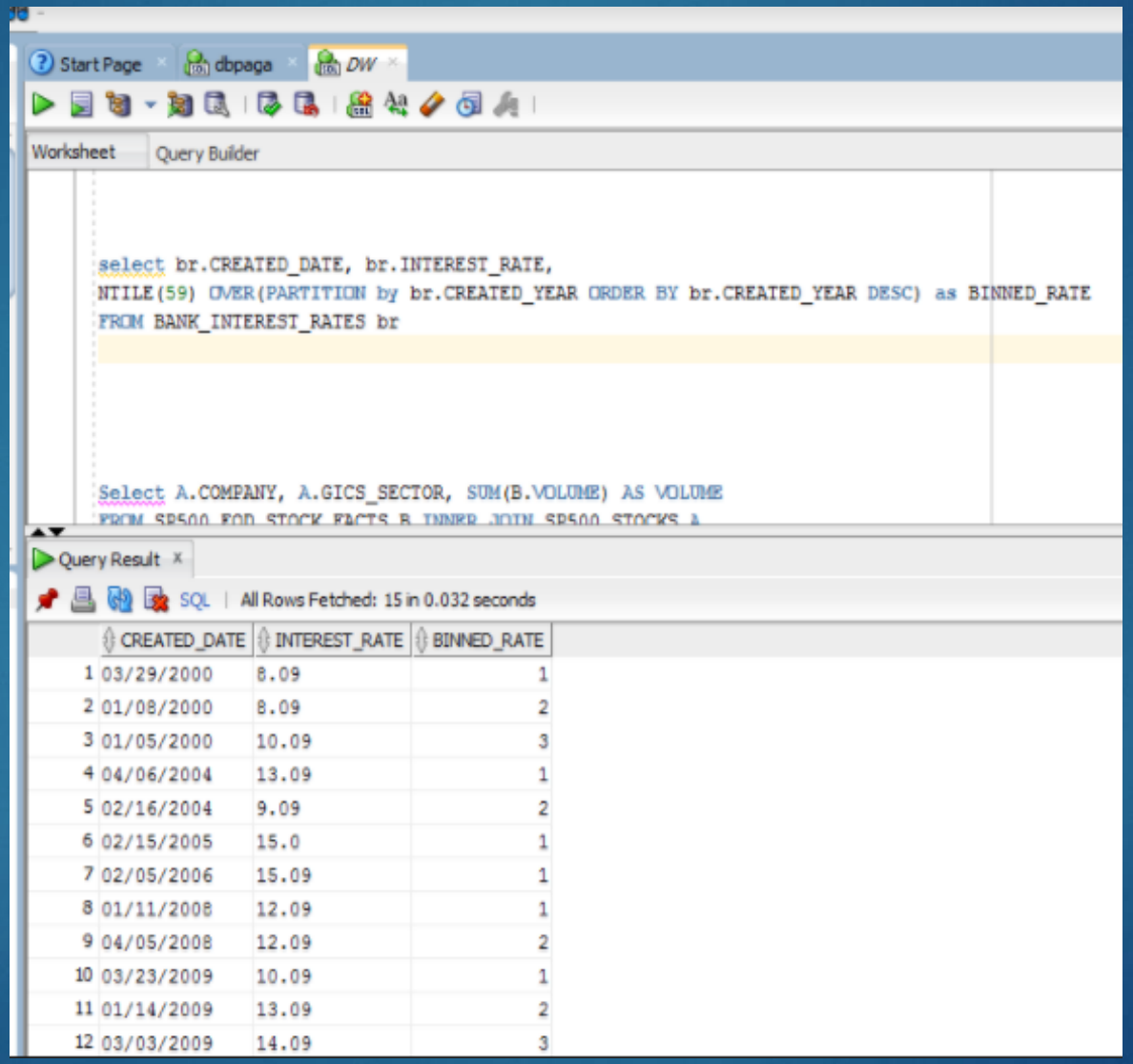
- SELECT bir.CREATED_DATE,
- bir.INTEREST_RATE,
- NTILE(10) OVER (ORDER BY bir.INTEREST_RATE DESC) AS BINNED_10_RATE
- FROM BANK_INTEREST_RATES bir;



	CREATED_DATE	INTEREST_RATE	BINNED_10_RATE
1	02/16/2004	9.09	1
2	03/29/2000	8.09	1
3	01/08/2000	8.09	2
4	02/28/2010	7.09	2
5	02/05/2006	15.09	3
6	02/15/2005	15.0	3
7	03/03/2009	14.09	4
8	04/06/2004	13.09	4
9	01/14/2009	13.09	5
10	04/05/2008	12.09	5
11	01/11/2008	12.09	6
12	03/22/2010	11.09	7
13	03/23/2009	10.09	8

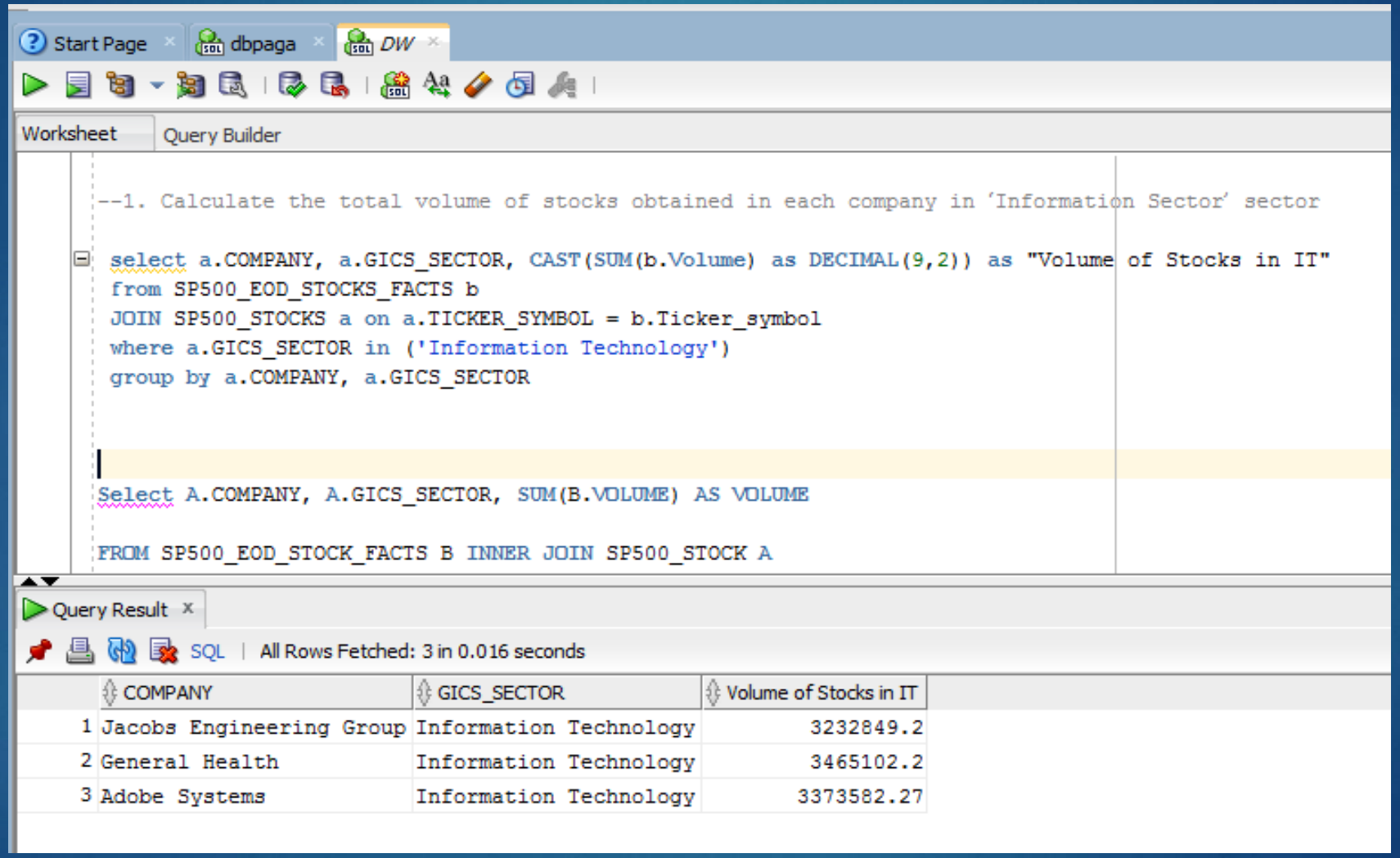
- Group the data by year or decade and use NTILE within the partitions to assign values

```
SELECT bir.CREATED_DATE,
bir.INTEREST_RATE,
NTILE(59) OVER ( PARTITION BY bir.CREATE_DATE ORDER BY bir.
CREATE_DATE DESC) AS BINNED_RATE
FROM  BANK_INTEREST_RATES bir
GROUP BY bir.CREATED_DATE,
bir.CREATED_DATE, bir.INTEREST_RATE;
```



	CREATED_DATE	INTEREST_RATE	BINNED_RATE
1	03/29/2000	8.09	1
2	01/08/2000	8.09	2
3	01/05/2000	10.09	3
4	04/06/2004	13.09	1
5	02/16/2004	9.09	2
6	02/15/2005	15.0	1
7	02/05/2006	15.09	1
8	01/11/2008	12.09	1
9	04/05/2008	12.09	2
10	03/23/2009	10.09	1
11	01/14/2009	13.09	2
12	03/03/2009	14.09	3

Interesting Query - 1



Interesting Query - 2

```
select a.TICKER_SYMBOL, b.CREATED_YEAR, MAX(b.INTEREST_RATE),
AVG(b.REFINANCING),
b.LOWER_LIMIT,
b.UPPER_LIMIT
from SP500_EOD_STOCKS_FACTS a INNER JOIN BANK_INTEREST_RATES b
on a.SCHEME_ID = b.SCHEME_ID
group by b.CREATED_YEAR, b.LOWER_LIMIT, b.UPPER_LIMIT
```