Db2 for i Advanced OLAP Functions

Jim Denton
jldenton@us.ibm.com
IBM Lab Services

OMNI June 2018

Db2 for i ibm.com/systems/power/software/i/db2

IT infrastructure > Power Systems > Software > IBM i >

IBM DB2 for i

Overview | Benefits | Getting started | Products | Resources

DB2 for i (formerly known as DB2 for i5/OS) is an advanced, 64-bit Relational Database Management System (RDBMS) that leverages the high performance, virtualization, and energy efficiency features of IBM's Power Systems. A member of IBM's leading edge family of DB2 products, DB2 for i supports a broad range of applications and development environments at a low cost of ownership due to its unique autonomic computing (self-managing) features.

Download a summary of the DB2 for i 7.3 enhancements

Db2 for i ibm.com/systems/power/software/i/db2

- Getting Started -> Whitepapers
  - Connectivity
    - Accessing DB2 for i from Linux
    - Heterogeneous Data Access from IBM i
  - Data Warehousing and Analytics
    - DB2 for i Star Schema Join Support
  - DB2 and XML
    - Replacing DB2 Extenders with DB2 Built in Support for XML
  - Modernization
    - DDS and SQL: A Winning Combination for Modernization
  - Performance and Scalability
    - Creating and Using Materialized Query Tables
    - Indexing and Statistics Strategies
    - Parallelism with DB2 Symmetric Multi-Processing
    - Table Partitioning Strategies
  - Query Modernization
    - Moving from OPNQRYF to SQL
    - Query/400 Modernization Services
  - SQL
    - Accessing Web Services using SQL

Db2 for i ibm.biz/DB2iWiki

Welcome to DB2 for i

| Updated July 12, 2016 by drmack | Tags: *None*

Page Actions

Welcome to the home page for the DB2 for i Wiki. Here you will find a variety of information from the leading experts for DB2 for i within IBM.

- Articles (Recent)
- Blog: DB2 for i
- Database Modernization
- Data Warehouse and Business Intelligence for IBM i
- DB2 for i Home Page
- DB2 for i (community) Forum
- Redbooks
- Technology refreshes
- Training and Consulting Services
- Whitepapers

New Wiki for Db2 Enhancements via PTF

- Regularly check (or subscribe to) the Db2 for i Updates Wiki!
  - Contains details on new PTFs that deliver new Db2 capabilities
  - Wiki : https://www.ibm.com/developerworks/ibmi/techupdates/db2

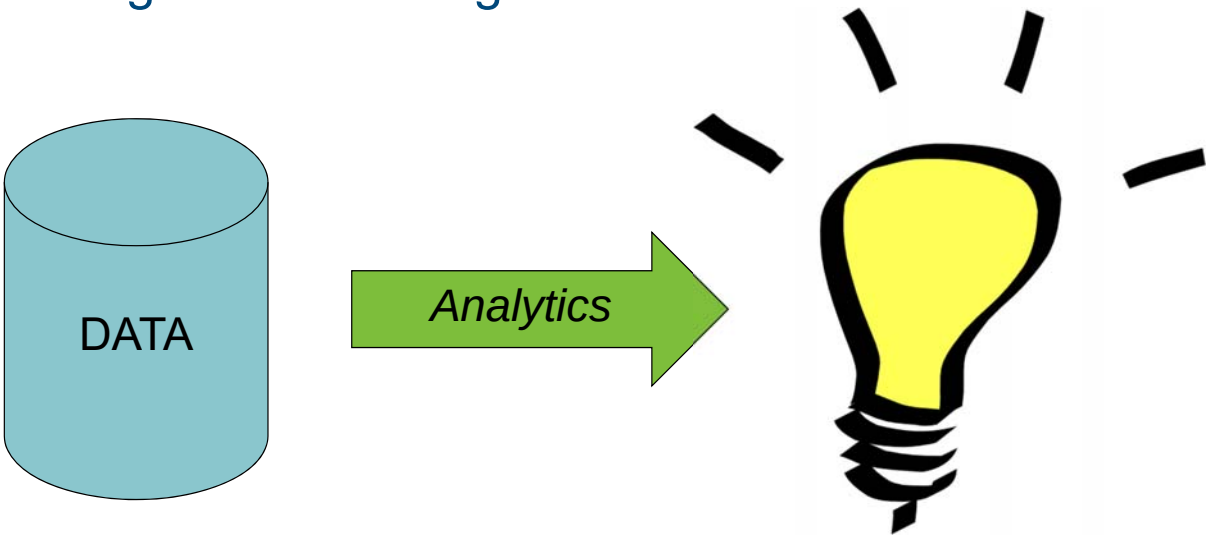
	Database Enhancement Landing Pages
IBM i 7.3	TR1 - Base Enhancements
IBM i 7.2	TR5 - TR4 - TR3 - TR
IBM i 7.1	TR11 - TR10 - TR9 -

DB2 for i updates by PTF Group and year
DB2 for i PTF Groups - 2017
DB2 for i PTF Groups - 2016
DB2 for i PTF Groups - 2015
DB2 for i PTF Groups - 2014
DB2 for i PTF Groups - 2013

What are you doing today for analytics?

- Choosing the platform
  - Hardware / Software / Data Model / Skills
- Managing ETL
- Choosing the reporting tools
  - Legacy tools like Query/400
  - Vendor provided reports
  - HLL reports
  - Spreadsheets
  - Modern tools (Web Query, COGNOS, etc.)
- Isolating the data model from the users
  - Views
  - Metadata

Turning Data into Insight

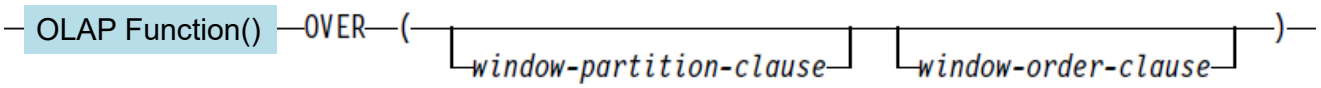


OLAP is Online *Analytical* Processing

What are OLAP Specifications?

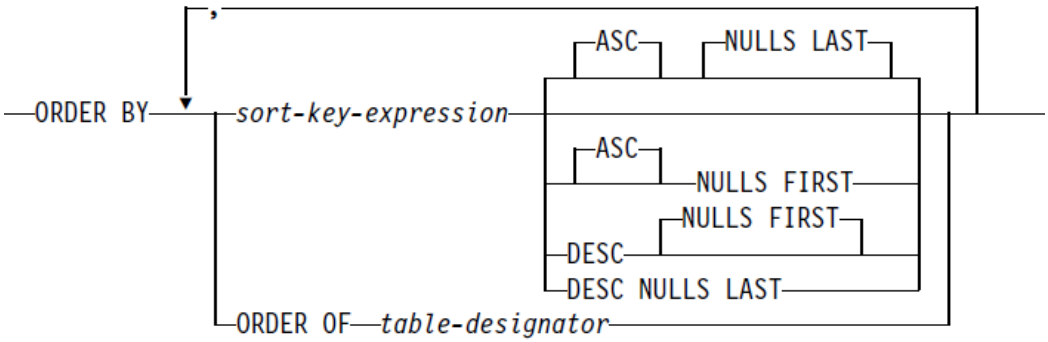
- On-line Analytical Processing functions providing the ability to return ranking, row numbering, aggregates and more as part of a SQL query result
  - Also referred to as window functions
- Can be specified as part of the select-list or ORDER BY clause
- The following is a list of the OLAP function categories:
  - Ordered OLAP specifications
  - Numbering specifications
  - Aggregation specifications
  - Super groups

OLAP Function Syntax Overview



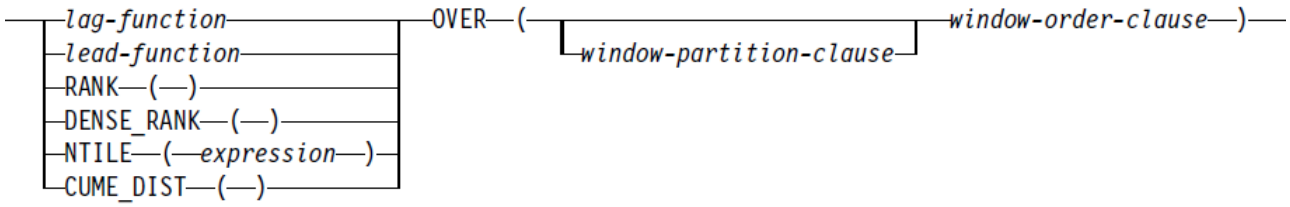
- **All of the OLAP functions allow the window partition clause and the window order clause**
  - The window order clause is sometimes required while the window partition clause is always optional
- **OVER** specifies the definition of the window over the result set
- The ***window-partition-clause*** defines the boundaries between the partitions within the window
- ***window-order-clause*** defines the sort order of the rows within a partition
  - This does not define the ordering of the result set

OLAP Function Window Order Clause



- ORDER BY clause defines the ordering used to calculate the rank or compute the row number
- *sort-key-expression* contains the ordering criteria
- NULLS FIRST or LAST determines whether null values appear before or after all non-null values
  - Nulls come last by default
- Optional ORDER OF specifies the table designator of the sub-select (or full-select) containing an ORDER BY clause

Ordered OLAP Specifications



- The ordered OLAP specifications are as follows:
  - RANK — Ordinal rank within the window
  - DENSE_RANK — Ordinal rank within the window
  - LAG — Reference to a preceding row in the window
  - LEAD — Reference to a following row in the window
  - NTILE — Quantile rank of a row within the window
  - CUME_DIST — Cumulative percentile ranking within the window including the current result set row
- These require the window order clause since the results depend on some kind of ordering

RANK and DENSE_RANK Example

- RANK and DENSE_RANK report values based on the window order clause independent of the result set sorting

```
SELECT empno, lastname, salary+bonus AS TOTAL_SALARY,
  RANK() OVER (ORDER BY salary+bonus DESC) AS Salary_Rank
  FROM employee
  WHERE salary + bonus > 30000
  ORDER BY lastname
```

SELECT empno, lastname, salary+bonus AS TOT...

EMPNO	LASTNAME	TOTAL_SALARY	SALARY_RANK
000050	GEYER	40975.00	5
000010	HAAS	53750.00	1
200010	HEMMINGER	47500.00	2
000090	HENDERSON	30350.00	11
200220	JOHN	30440.00	9
000030	KWAN	39050.00	6
000110	LUCCHESSI	47400.00	3
000220	LUTZ	30440.00	9
000070	PULASKI	36870.00	7
000060	STERN	32750.00	8
000020	THOMPSON	42050.00	4

Dense_Rank() Output

SALARY_RANK
5
1
2
10
9
6
3
9
7
8
4

RANK and DENSE_RANK Example

```
SELECT empno, lastname, salary, bonus,
   DENSE_RANK() OVER (ORDER BY salary+bonus DESC) AS Comp_Rank ,
   DENSE_RANK() OVER (ORDER BY salary DESC) AS Salary_Rank ,
   DENSE_RANK() OVER (ORDER BY bonus DESC) AS Bonus_Rank
   FROM employee
   WHERE salary + bonus > 30000;
```

EMPNO	LASTNAME	SALARY	BONUS	COMP_RANK	SALARY_RANK	BONUS_RANK
000010	HAAS	52750.00	1000.00	1	1	1
200010	HEMMINGER	46500.00	1000.00	2	2	1
000110	LUCCHESSI	46500.00	900.00	3	2	2
000020	THOMPSON	41250.00	800.00	4	3	3
000050	GEYER	40175.00	800.00	5	4	3
000030	KWAN	38250.00	800.00	6	5	3
000070	PULASKI	36170.00	700.00	7	6	4
000060	STERN	32250.00	500.00	8	7	6
000220	LUTZ	29840.00	600.00	9	8	5
200220	JOHN	29840.00	600.00	9	8	5
000090	HENDERSON	29750.00	600.00	10	9	5

PARTITION BY Example

- Rank top entries by department

```
SELECT workdept, empno, lastname, salary+bonus AS TOTAL_SALARY,
  RANK() OVER (PARTITION BY workdept
               ORDER BY salary+bonus DESC) AS Salary_Rank
FROM employee
WHERE salary + bonus > 30000
ORDER BY workdept, lastname
```

WORKDEPT	EMPNO	LASTNAME	TOTAL_SALARY	SALARY_RANK
A00	000010	HAAS	53750.00	1
A00	200010	HEMMINGER ...	47500.00	2
A00	000110	LUCCHESSI	47400.00	3
B01	000020	THOMPSON ...	42050.00	1
C01	000030	KWAN	39050.00	1
D11	200220	JOHN	30440.00	2
D11	000220	LUTZ	30440.00	2
D11	000060	STERN	32750.00	1
D21	000070	PULASKI	36870.00	1
E01	000050	Rockets	40975.00	1
E11	000090	HENDERSON ...	30350.00	1

Ranking is restarted for each department (partition)

LAG and LEAD Example

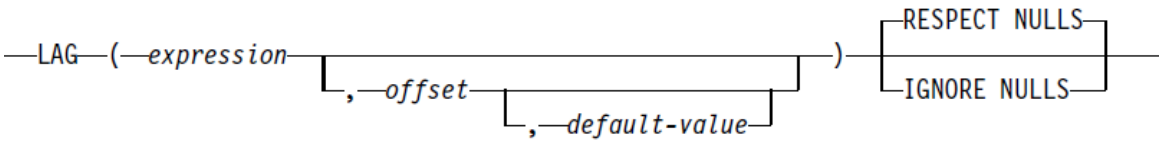
Compare the sales of stores within the same region including comparisons to the stores that were adjacent in terms of better and worse sales

```
SELECT store, region, sales,
       sales - LAG(sales,1) OVER(PARTITION BY region ORDER BY sales)
                 AS prior_diff,
       LEAD(sales,1) OVER(PARTITION BY region ORDER BY sales) - sales
       AS next_diff
  FROM stores ORDER BY region, sales
```

STORE	REGION	SALES	PRIOR_DIFF	NEXT_DIFF
Bobs	NW	100,000.00	-	340,000.00
Toms	NW	440,000.00	340,000.00	60,000.00
Mills	NW	500,000.00	60,000.00	-
Targe	SW	140,000.00	-	260,000.00
Menes	SW	400,000.00	260,000.00	370,000.00
Caining	SW	770,000.00	370,000.00	-

LAG / LEAD Syntax

- LAG and LEAD specifications have additional options to allow greater flexibility:



  - The offset is the offset to the lagging/leading row in the window of the result set. It must be a positive integer and defaults to 1.
  - Default value is what to use if the expression is null.
  - Respect and ignore nulls apply to the expression. If it is null, it is not included in the results.

NTILE Example

Calculate the quartile ranking in terms of highest sales for all stores:

```
SELECT store, region, sales,
        NTILE(4) OVER(ORDER BY sales DESC)
                quartile_rank,
   FROM stores ORDER BY sales DESC
```

STORE	REGION	SALES	QUARTILE_RANK
Caining	SW	770,000.00	1
Mills	NW	500,000.00	1
Toms	NW	440,000.00	1
Menes	SW	400,000.00	2
…	….	….	…
Bobs	NW	100,000.00	4

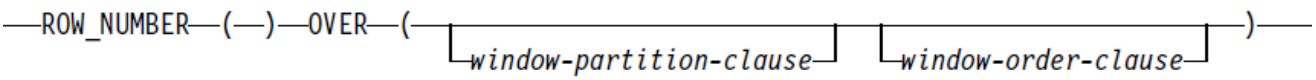
CUME_DIST Example

Select the stores that are in the top 30 percent in terms of sales:

```
WITH t AS
        ( SELECT store, region, sales,
                CUME_DIST() OVER(ORDER BY sales DESC)
                        cume_dist FROM stores )
   SELECT * FROM t WHERE cume_dist <= .30
        ORDER BY sales DESC
```

STORE	REGION	SALES	CUME_DIST
Caining	SW	770,000.00	0.09
Mills	NW	500,000.00	0.18
Toms	NW	440,000.00	0.27

Numbering Specifications



- ROW_NUMBER is the only supported numbering specification.
- The window partition clause is optional.
- The window order clause is optional but defaults to an arbitrary value based on implementation and not to the ORDER BY for the statement's result set.

ROW_NUMBER Example

- ROW_NUMBER can be used to assign a number to query result rows

```
SELECT ROW_NUMBER() OVER
     (ORDER BY workdept, lastname) AS Nbr,
  lastname, salary
 FROM employee
ORDER BY workdept, lastname
```

NBR	LASTNAME	SALARY
1	HAAS	52750.00
2	HEMMINGER	46500.00
3	LUCCHESSI	46500.00
4	O'CONNELL	29250.00
5	ORLANDO	29250.00

```
SELECT workdept, lastname, hiredate,
   ROW_NUMBER() OVER  (PARTITION BY workdept
   ORDER BY hiredate) AS Nbr
FROM employee
ORDER BY workdept, hiredate
```

WORKDEPT	LASTNAME	HIREDATE	NBR
A00	LUCCHESSI	1958-05-16	1
A00	O'CONNELL	1963-12-05	2
A00	HAAS	1965-01-01	3
A00	HEMMINGER	1965-01-01	4
A00	ORLANDO	1972-05-05	5
B01	THOMPSON	1973-10-10	1
C01	QUINTANA	1971-07-28	1
C01	KWAN	1975-04-05	2

Stateful versus Stateless Pagination

- Consuming large result sets in one transaction can result in long response times and unhappy end users
- The concept of pagination or page-at-a-time has been widely used in legacy applications
  - Developers took advantage of stateful, persistent connections
  - Database managed cursor positioning
- Browser based applications tend to be stateless
  - The database connection is not persistent
  - Cursor positioning must be handled within the client application

Example of Stateful Pagination

- Stateful pseudo code
  - Connect, Open
  - ⇒ Fetch First 5 rows
  - ⇒ Fetch Next 5 rows
  - ⇒ Fetch next 5 rows
  - Close, Disconnect
- The connection to the database is persistent during the life of the cursor
- Subsequent fetches start at the next sequential row
- Duplicate data spans pages
- Coding is simple

Result set row ordinal position	Ordering Data	Unique key (Encrypted)
1	Abcd	1234
2	Abdc	3214
3	Acbd	4131
4	Acdb	2143
5	**Bacd**	1243
6	**Bacd**	2341
7	Bcad	4213
8	Bcda	3142
9	Bdac	1423
10	**Bdca**	2431
11	**Bdca**	3412
12	Cadb	1324
13	Cbad	4321

Example of Stateless Pagination

- Stateless pseudo code
  - Connect, Open, Fetch first 5 rows, Close, Disconnect
  - Connect, Open, Fetch first 10 rows, Close, Disconnect
  - Connect, Open, Fetch first 15 rows, Close, Disconnect
- Cursor position is lost after close and disconnect
- Positioning data must be preserved across connections
  - Ordering data and/or unique key may not be suitable for positioning
- Application positioning results in slow response times
  - Previously fetched rows may be retrieved multiple times
  - Copies of result sets are sometimes made
- What if the ordinal position number was part of the result set?

Result set row ordinal position	Ordering Data	Unique key (Encrypted)
1	Abcd	1234
2	Abdc	3214
3	Acbd	4131
4	Acdb	2143
5	**Bacd**	1243
6	**Bacd**	2341
7	Bcad	4213
8	Bcda	3142
9	Bdac	1423
10	**Bdca**	2431
11	**Bdca**	3412
12	Cadb	1324
13	Cbad	4321

ROW_NUMBER To The Rescue!

- Row Number pseudo code

  Connect, Open(row_number>=1), Fetch 5 rows, Close, Disconnect

  Connect, Open(row_number>=6, Fetch 5 rows, Close, Disconnect

  Connect, Open(row_number>=11, Fetch 5 rows, Close, Disconnect

Result set Row Number	Ordering Data	Unique key (Encrypted)
1	Abcd	1234
2	Abdc	3214
3	Acbd	4131
4	Acdb	2143
5	**Bacd**	1243
6	**Bacd**	2341
7	Bcad	4213
8	Bcda	3142
9	Bdac	1423
10	**Bdca**	2431
11	**Bdca**	3412
12	Cadb	1324
13	Cbad	4321

Pagination Using ROW_NUMBER

```
1  WITH rownum_cte AS
     (SELECT empno,
       ROW_NUMBER() OVER
        (ORDER BY lastname, firstnme)
                              AS rownbr
        FROM employee)
   SELECT rownbr, AE.* FROM
     employee AE  INNER JOIN
2          rownum_cte C
     ON AE.empno=C.empno
3  WHERE rownbr >= ?
   ORDER BY rownbr
```

Rownbr >= 1

ROWNBR	EMPNO	FIRSTNME	MIDI...	LASTNAME
1	000150	BRUCE		ADAMSON
2	200340	ROY	R	ALONZO
3	000200	DAVID		BROWN
4	000340	JASON	R	GOUNOT
5	000010	CHRISTINE	N	HAAS

Rownbr >= 6

ROWNBR	EMP...	FIRSTNME	MIDI...	LASTNAME
6	200010	DIAN	J	HEMMINGER ...
7	000090	EILEEN	W	HENDERSON ...
8	000230	JAMES	J	JEFFERSON
9	200220	REBA	K	JOHN
10	000260	SYBIL	P	JOHNSON

Rownbr >= 11

ROWNBR	EM...	FIRSTNME	MIDI...	LASTNAME
11	000...	WILLIAM	T	JONES
12	000...	SALLY	A	KWAN
13	000...	WING		LEE
14	000...	VINCENZO	G	LUCCHESSI
15	000...	JENNIFER	K	LUTZ

- Key steps:
  1. CTE must be used to compute the row number - OLAP specification not allowed on WHERE clause
  2. Computed row number used on WHERE clause to starting row for a page
  3. ORDER BY guarantees the data will be ordered based on the ROW_NUMBER window order

OFFSET and LIMIT for Stateless Pagination

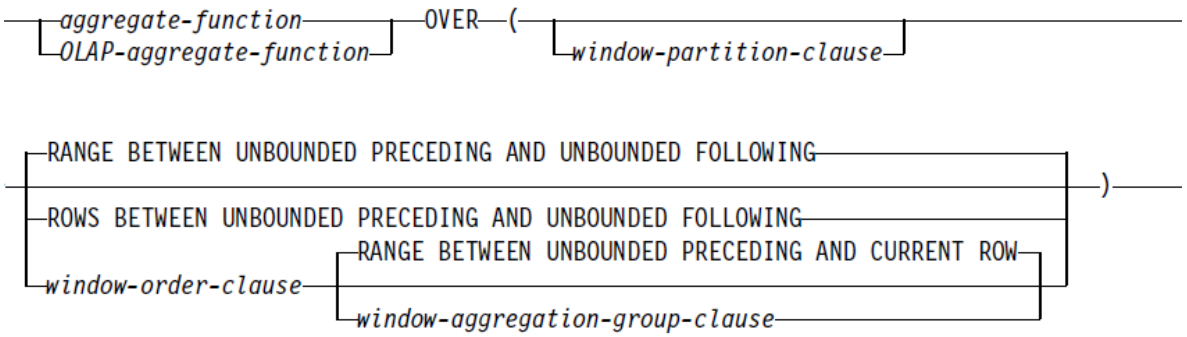
Connect,
SELECT…OFFSET 0 LIMIT 5
Fetch 5 rows, Close, Disconnect

Connect,
SELECT…OFFSET 5 LIMIT 5
Fetch 5 rows, Close, Disconnect

Connect,
SELECT…OFFSET 10 LIMIT 5
Fetch 5 rows, Close, Disconnect

Result set Row Number	Ordering Data	Unique key (Encrypted)
1	Abcd	1234
2	Abdc	3214
3	Acbd	4131
4	Acdb	2143
5	**Bacd**	1243
6	**Bacd**	2341
7	Bcad	4213
8	Bcda	3142
9	Bdac	1423
10	**Bdca**	2431
11	**Bdca**	3412
12	Cadb	1324
13	Cbad	4321

Aggregation Specifications



- Aggregation specifications are very powerful which also means the syntax can be complicated.
- The big differences from ordering and numbering specifications are in the RANGE and ROW clauses

Aggregate Functions

- AVG
- CORRELATION
- COUNT
- COUNT_BIG
- COVARIANCE
- COVARIANCE_SAMP
- MAX
- MEDIUM
- MIN
- PERCENTILE_CONT
- PERCENTILE_DISC
- SUM
- Regression Functions
  - REGR_AVGX
  - REGR_AVGY
  - REGR_COUNT
  - REGR_INTERCEPT
  - REGR_R2
  - REGR_SLOPE
  - REGR_SXX
  - REGR_SXY
  - REGR_SYY
- STDDEV
- STDDEV_SAMP
- VARIANCE
- VARIANCE_SAMP

SUM Aggregate Functions

Return the detail store information and the total sales by region plus the percentage the store contributed to the total for the region:

```
SELECT store, region, sales,
        SUM(sales) OVER(PARTITION BY region) region_total,
        DECIMAL(100*sales / SUM (sales)
           OVER(PARTITION BY region), 5,2) percentage
  FROM stores ORDER BY region,  percentage
```

STORE	REGION	SALES	REGION_TOTAL	PERCENTAGE
Wally	NE	150,000.00	450,000.00	33.33
Pensk	NE	300,000.00	450,000.00	66.66
Bobs	NW	100,000.00	1,040,000.00	9.61
Toms	NW	440,000.00	1,040,000.00	42.30
Mills	NW	500,000.00	1,040,000.00	48.07

SUM Aggregate Example – Rolling Sum

Return the detail store information and the rolling sum of the store sales:

```
SELECT store, region, sales,
        SUM(sales)
              OVER(ORDER BY sales DESC) rolling_sum
FROM stores ORDER BY rolling_sum
```

STORE	REGION	SALES	ROLLING_SUM
Caining	SW	770,000.00	770,000.00
Mills	NW	500,000.00	1,270,000.00
Toms	NW	440,000.00	1,710,000.00
Menes	SW	400,000.00	2,110,000.00
BBB	SE	350,000.00	2,460,000.00

Correlation, Covariance and Covariance_Samp

Use correlation and covariance to analyze the relationship between salary and bonus for each department:

```
SELECT workdept,
        CORRELATION(salary, bonus) correlation,
        COVARIANCE(salary, bonus) covariance,
        COVARIANCE_SAMP(salary, bonus) covariance_samp
FROM employee GROUP BY workdept  ORDER BY workdept
```

WORKDEPT	CORRELATION	COVARIANCE	COVARIANCE_SAMP
A00	0.976023	1,743,000	2,178,750
B01	-	0	-
C01	0.999835	574,437	765,916
D11	0.775424	240,454	264,500
…	….	….	…
E21	0.910221	68,944	82,733

Correlation, Covariance and Covariance_Samp

Use the correlation and covariance aggregate functions in an OLAP expression to further analyze for department ‘A00’ the relationship between salary and bonus:

```
SELECT empno,
   CORRELATION(salary, bonus) OVER(PARTITION BY workdept
        ORDER BY empno) correlation,
   COVARIANCE(salary, bonus)   OVER(PARTITION BY workdept
        ORDER BY empno) covariance
  FROM employee WHERE workdept = ‘A00’ ORDER BY empno
```

EMPNO	CORRELATION	COVARIANCE
000010	-	0
000110	1.000000	156,250
000120	0.999853	1,688,888
200010	0.962723	1,381,250
200120	0.976023	1,743,000

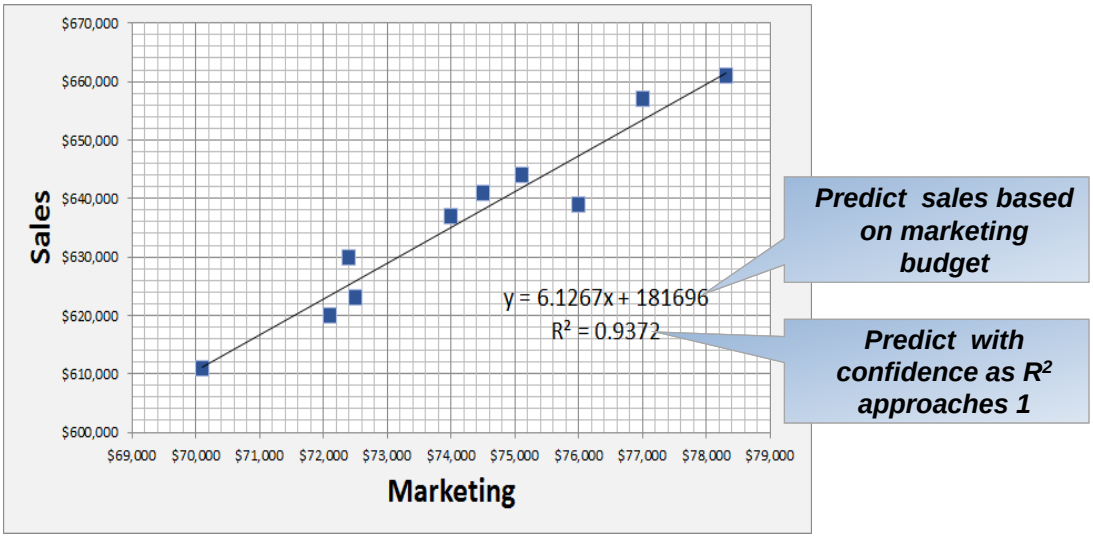
Regression Aggregate Functions

Business questions:

- Is there a correlation between the amount spent on marketing and sales for a product?
- Is the correlation weak or strong?
- Can we predict sales based on the amount spent on marketing?

Year/Quarter	Marketing	Sales
2014 Q1	$70,100	$611,000
2014 Q2	$77,000	$657,000
2014 Q3	$72,100	$620,000
2014 Q4	$72,500	$623,000
2015 Q1	$78,300	$661,000
2015 Q2	$74,500	$641,000
2015 Q3	$74,000	$637,000
2015 Q4	$72,400	$630,000
2016 Q1	$75,100	$644,000
2016 Q2	$76,000	$639,000

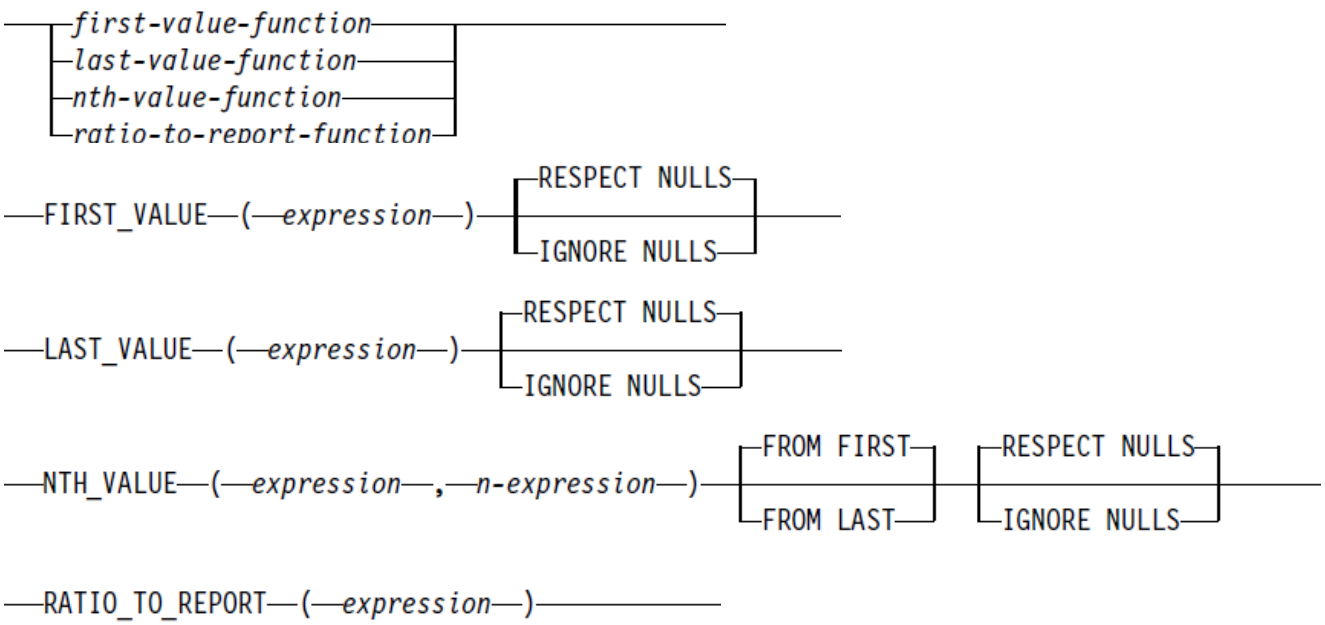
Regression Aggregate Functions



Business results:

- SELECT **REGR_SLOPE** (sales, mktg), **REGR_INTERCEPT** (sales, mktg)
  FROM salesdata
- SELECT POWER (**CORRELATION** (sales, mktg), 2)
  FROM salesdata

OLAP Aggregation Specifications



FIRST, LAST, and NTH Value

Compare the sales of the current store to the store with the best sales, second best sales, and the worst sales results:

```
SELECT store, sales,
   sales - FIRST_VALUE(sales) OVER (ORDER BY sales DESC
         RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)
               behind_1st,
   sales - NTH_VALUE(sales,2) OVER (ORDER BY sales DESC
         RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)
               behind_2nd,
   sales - LAST_VALUE(sales) OVER (ORDER BY sales DESC
         RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)
               compared_to_last
FROM stores ORDER BY sales DESC
```

STORE	SALES	BEHIND_1ST	BEHIND_2ND	COMPARED_TO_LAST
Caining	770,000.00	0.00	270,000.00	670,000.00
Mills	500,000.00	-270,000.00	0.00	400,000.00
..	…	…	…	…
Bobs	100,000.00	-670,000.00	-400,000.00	0.00

RATIO_TO_REPORT Example

Calculate the quartile ranking for all stores and show their overall sales percentage:

```
SELECT store, region, sales,
        DECIMAL(RATIO_TO_REPORT(sales) OVER() *100, 10, 2) percent
  FROM stores ORDER BY sales DESC
```

STORE	REGION	SALES	PERCENT
Caining	SW	770,000.00	21.10
Mills	NW	500,000.00	13.70
Toms	NW	440,000.00	12.05
Menes	SW	400,000.00	10.96
…	….	….	…
Bobs	NW	100,000.00	2.74

Grouping Sets and Super Groups

- Many BI applications and OLAP tools involve hierarchical, multi-dimensional aggregate views of transaction data
  - Users need to view results at multiple levels
  - Users need to view result data from different perspective
  - Current grouping support only allows aggregation data of along a SINGLE dimension

    EXAMPLE: SELECT country region, store, product, SUM(sales)
    FROM trans
    GROUP BY **country, region, store, product**
  - Limitations result in extra coding for programmers

- **6.1** grouping and OLAP capabilities allow data to be grouped in multiple ways with a single SQL request
  - ROLLUP
  - CUBE
  - GROUPING SETS

ROLLUP

- An extension to the GROUP BY clause that produces a result set containing sub-total rows in addition to the "regular" grouped rows
- Sub-total rows are "super-aggregate" rows that contain further aggregates whose values are derived by applying the same column functions that were used to obtain the grouped rows
- ROLLUP on the GROUP BY clause results in DB2 returning aggregates for each level of the hierarchy implicitly represented in the grouping columns

ROLLUP

- ROLLUP(Country, Region) will result in the data being summarized at the following levels
  - (Country, Region)
  - (Country)
  - ( ) << represents Grand Total
- **Example Query**:

```
SELECT country, region, SUM(sales)
 FROM trans
 GROUP BY ROLLUP (country, region)
```

ROLLUP Output Example

SELECT country, region, SUM(sales) FROM trans
GROUP BY ROLLUP (country, region)

GROUP BY country,NULL

GROUP BY NULL, NULL

Country	Region	Sum(Sales)
Canada	-	100,000
Canada	NW	100,000
USA	-	3,250,000
USA	NE	450,000
USA	NW	940,000
USA	SE	550,000
USA	SW	1,310,000
-	-	3,350,000

ROLLUP Output Example

SELECT country, region, SUM(sales) FROM trans
GROUP BY ROLLUP (country, region)
ORDER BY country, region

GROUP BY country,NULL

GROUP BY NULL, NULL

Country	Region	Sum(Sales)
Canada	NW	100,000
Canada	-	100,000
USA	NE	450,000
USA	NW	940,000
USA	SE	550,000
USA	SW	1,310,000
USA	-	3,250,000
-	-	3,350,000

ROLLUP Output Example

```
SELECT      IFNULL(country,'GRAND'),
            IFNULL(region,'TOTAL'),
            SUM(sales) FROM trans
GROUP BY  ROLLUP (country, region)
ORDER BY country, region
```

You can also use COALESCE and CASE for formatting

Country	Region	Sum(Sales)
Canada	NW	100,000
Canada	TOTAL	100,000
USA	NE	450,000
USA	NW	940,000
USA	SE	550,000
USA	SW	1,310,000
USA	TOTAL	3,250,000
GRAND	TOTAL	3,350,000

CUBE

- An extension to the GROUP BY clause that produces a result set that contains all the rows of a ROLLUP aggregation, plus contains "cross-tabulation" rows
- Cross-tabulation rows are additional "super-aggregate" rows that are not part of an aggregation with sub-totals
- CUBE on the GROUP BY clause results in DB2 returning aggregates for all possible distinct combinations represented by the grouping columns

CUBE

- CUBE(Country, Region) will result in the data being summarized at the following levels
  - (Country, Region)
  - (Country)
  - (Region)
  - ( ) << represents Grand Total
- Returns results at multiple intersection points
- **Example Query**:
  SELECT country, region, SUM(sales)
  FROM trans
  GROUP BY CUBE(country, region)

CUBE Output Example

SELECT country,region, SUM(sales) FROM trans
GROUP BY CUBE (country, region)

GROUP BY NULL, region

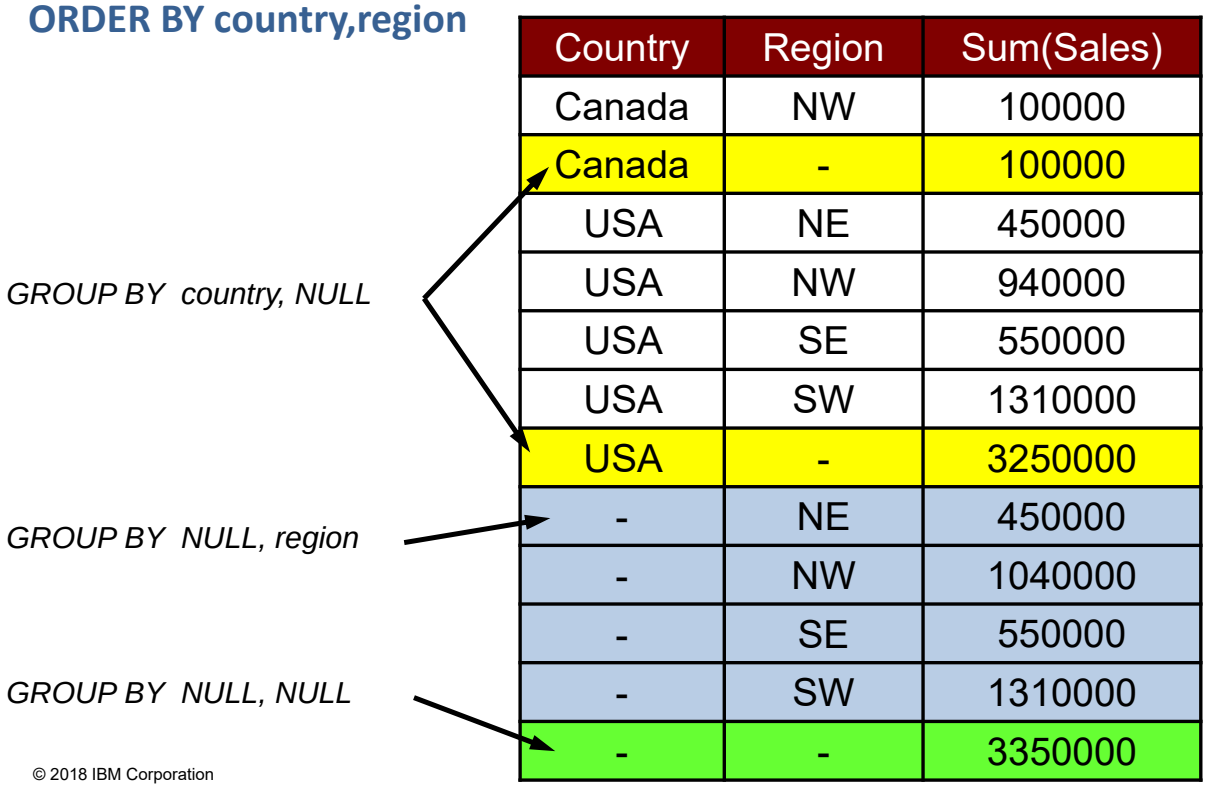
GROUP BY NULL, NULL

GROUP BY country, NULL

Country	Region	Sum(Sales)
-	NE	450000
-	NW	1040000
-	SE	550000
-	SW	1310000
-	-	3350000
Canada	-	100000
USA	-	3250000
Canada	NW	100000
USA	NE	450000
USA	NW	940000
USA	SE	550000
USA	SW	1310000

CUBE Output Example

SELECT country,region, SUM(sales) FROM trans
GROUP BY CUBE (country, region)
ORDER BY country,region



Country	Region	Sum(Sales)
Canada	NW	100000
Canada	-	100000
USA	NE	450000
USA	NW	940000
USA	SE	550000
USA	SW	1310000
USA	-	3250000
-	NE	450000
-	NW	1040000
-	SE	550000
-	SW	1310000
-	-	3350000

GROUPING SETS

- Allows multiple grouping clauses to be specified in a single statement
- This can be thought of as the union of two or more groups of rows into a single result set
- GROUPING SET on the GROUP BY clause enables DB2 to return aggregates for multiple sets of grouping columns

GROUPING SETS

- GROUPING SETS((Country, Region), (Country, Store)) will result in the data being summarized at the following levels
  - (Country, Region)
  - (Country, Store)
- CUBE and ROLLUP can be used in combination with Grouping Sets

  CAUTION: These types of combinations can result in an exponential growth in the number of grouping sets returned by a query, combine carefully
- **Example Query**:

  SELECT country, region, SUM(sales)
  FROM trans
  GROUP BY
  GROUPING SETS((country, region), (country, store))

GROUPING SETS Output Example

```
SELECT country, region, store, SUM(sales)
FROM trans
GROUP BY
GROUPING SETS
((country, region), (country, store))
```

GROUP BY COUNTRY, REGION

GROUP BY COUNTRY, STORE

Country	Region	Store	Sum(Sales)
Canada	NW	-	100,000
USA	NE	-	450,000
USA	NW	-	940,000
USA	SE	-	550,000
USA	SW	-	1,310,000
Canada	-	Dougs	100,000
USA	-	Mariahs	350,000
USA	-	KMs	770,000
USA	-	Jennas	400,000
USA	-	Adrians	500,000
USA	-	Joshs	300,000
USA	-	TZs	200,000
USA	-	Maddies	210,000

GROUPING

- The GROUPING function can be used to determine if null values are from underlying user data or DB2 aggregate processing
  - Function returns **1** if grouping column contains NULL value produced by grouping set or super group processing
  - Function returns **0** if grouping column contains "real" GROUP BY value

***EXAMPLE**:* *SELECT country,region, store, GROUPING(store), SUM(sales)*
FROM trans
WHERE transYear = 2006
GROUP BY
GROUPING SETS ((country, region),(country, store))

Grouping Sets & Super Groups: View Considerations

- Grouping Set & Super Groups produce additional rows not in underlying table. WHERE clause can cause different results
  - Filtering part of View virtual table definition OR…
  - Filtering applied to retrieval of rows from View virtual table

```
CREATE VIEW v1 AS
  SELECT country, region, SUM(sales) FROM trans
   WHERE country = 'USA'
GROUP BY  ROLLUP (country, region)
```

SELECT * FROM v1

Country	Region	Sum(Sales)
USA	NE	450,000
USA	NW	940,000
USA	SE	550,000
USA	SW	1,310,000
USA	-	3,250,000
-	-	3,250,000

```
CREATE VIEW v2 AS
  SELECT country, region, SUM(sales) FROM trans
GROUP BY  ROLLUP (country, region)
```

SELECT * FROM v2 WHERE country='USA'

Country	Region	Sum(Sales)
USA.	NE	450,000
USA	NW	940,000
USA	SE	550,000
USA	SW	1,310,000
USA	-	3,250,000

Where does Db2 Web Query for i fit in?

Salary Analysis

- Your HR department wants to ensure salaries are equitable across the company and across departments and there aren't outliers or other discrepancies
  - What is the employee's salary compared to the average WITHIN their department?
  - What is an employee's ratio of salary within their department and overall company?
  - For each employee, compare their salary to the two closest behind this person's salary, and the two closest ahead of them in salary

The Process – Create SQL Views

View for Salary compared to AVERAGE

```
1  --  Generate SQL
2  --  Version:                  V7R3M0 160422
3  --  Generated on:             01/19/17 09:48:56
4  --  Relational Database:      DB2ICOE2
5  --  Standards Option:         DB2 for i
6  CREATE VIEW SAMPLEDB.CMPTOAVG (
7      EMPNO,
8      WORKDEPT,
9      SALARYL,
10     ABOVE_OR_BELOW_AVG,
11     DEPT_AVG_SALARY,
12                               DELTA)
13     AS
14
15     SELECT  EMPNO, WORKDEPT, SALARY,
16     CASE
17         WHEN (SALARY - (AVG(SALARY) OVER (PARTITION BY WORKDEPT))) > 0
18             THEN '...IS ABOVE AVG'
19         WHEN (SALARY - (AVG(SALARY) OVER (PARTITION BY WORKDEPT))) < 0
20             THEN '...IS BELOW AVG'
21         ELSE '...IS EQUAL TO AVG'
22     END AS ABOVE_OR_BELOW_AVG,
23     DECIMAL(AVG(SALARY) OVER (PARTITION BY WORKDEPT)) AS DEPT_AVG_SALARY,
24     DECIMAL(SALARY - (AVG(SALARY) OVER (PARTITION BY WORKDEPT)) ) AS DELTA
25 FROM    SAMPLEDB.EMPLOYEE;
26
```


The Process – Create SQL Views …

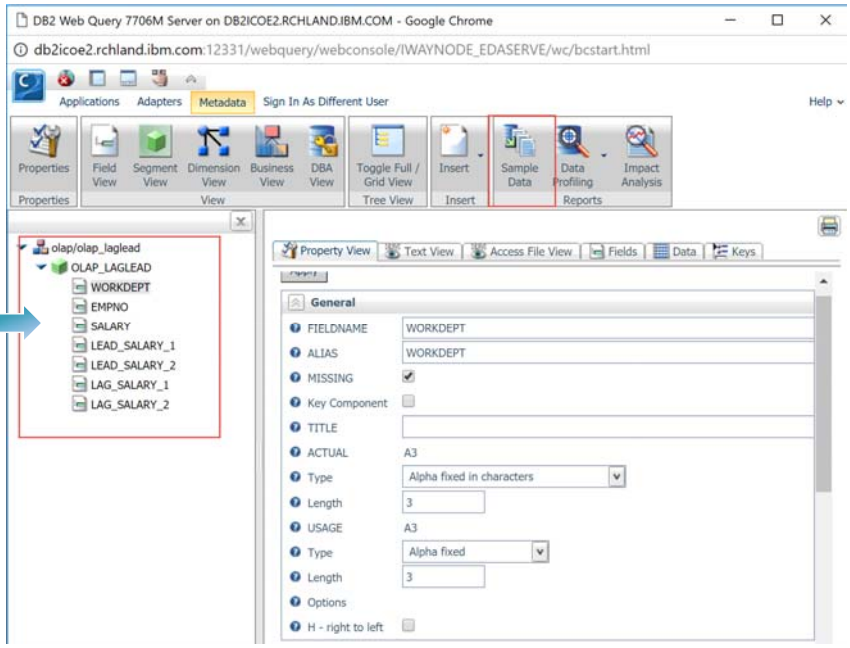
View for 2 ahead and 2 behind

```
1  --  Generate SQL
2  --  Version:                  V7R3M0 160422
3  --  Generated on:             01/19/17 09:48:56
4  --  Relational Database:      DB2ICOE2
5  --  Standards Option:         DB2 for i
6  CREATE VIEW SAMPLEDB.LAGLEAD (
7      WORKDEPT ,
8      EMPNO,
9      SALARY ,
10     LEAD_SALARY_1,
11     LEAD_SALARY_2,
12     LAG_SALARY_1,
13     LAG_SALARY_2  )
14     AS
15       SELECT        WORKDEPT, EMPNO, SALARY,
16        LEAD(SALARY, 1) OVER (PARTITION BY WORKDEPT
17                              ORDER BY SALARY) as LEAD_SALARY_1,
18        LEAD(SALARY, 2) OVER (PARTITION BY WORKDEPT
19                ORDER BY SALARY) as LEAD_SALARY_2,
20        LAG(SALARY, 1)  OVER (PARTITION BY WORKDEPT
21                ORDER BY SALARY) as LAG_SALARY_1,
22        LAG(SALARY, 2)  OVER (PARTITION BY WORKDEPT
23                ORDER BY SALARY) as LAG_SALARY_2
24 FROM        SAMPLEDB.EMPLOYEE
25 WHERE       WORKDEPT = 'D11'
26
27     RCDFMT LAGLEAD      ;
```

The Process – Create Synonyms

- Within Db2 Web Query, create a “synonym” (term for meta data) over each of the SQL Views

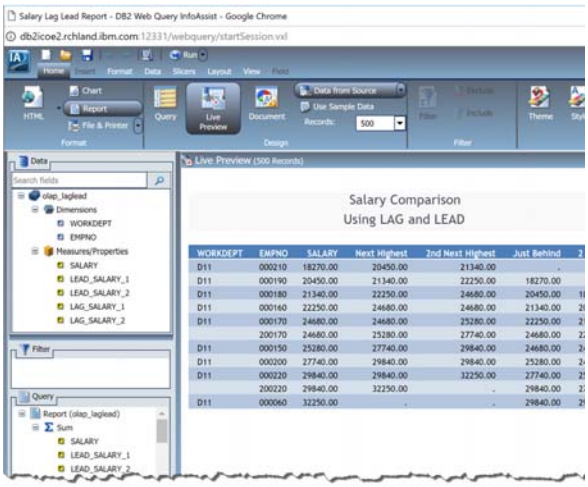
These are fields returned from the view



The Process – Build Reports

- Build Your Reports/Charts/Dashboards with Db2 Web Query InfoAssist
  - Add additional filters or virtual fields
  - Format header/footer/stylesheet
  - Choose output
    - Excel, HTML, mobile
  - Add to dashboard
  - Embed in your app
  - Feed into your data warehouse

These are fields returned from the view and available in your report

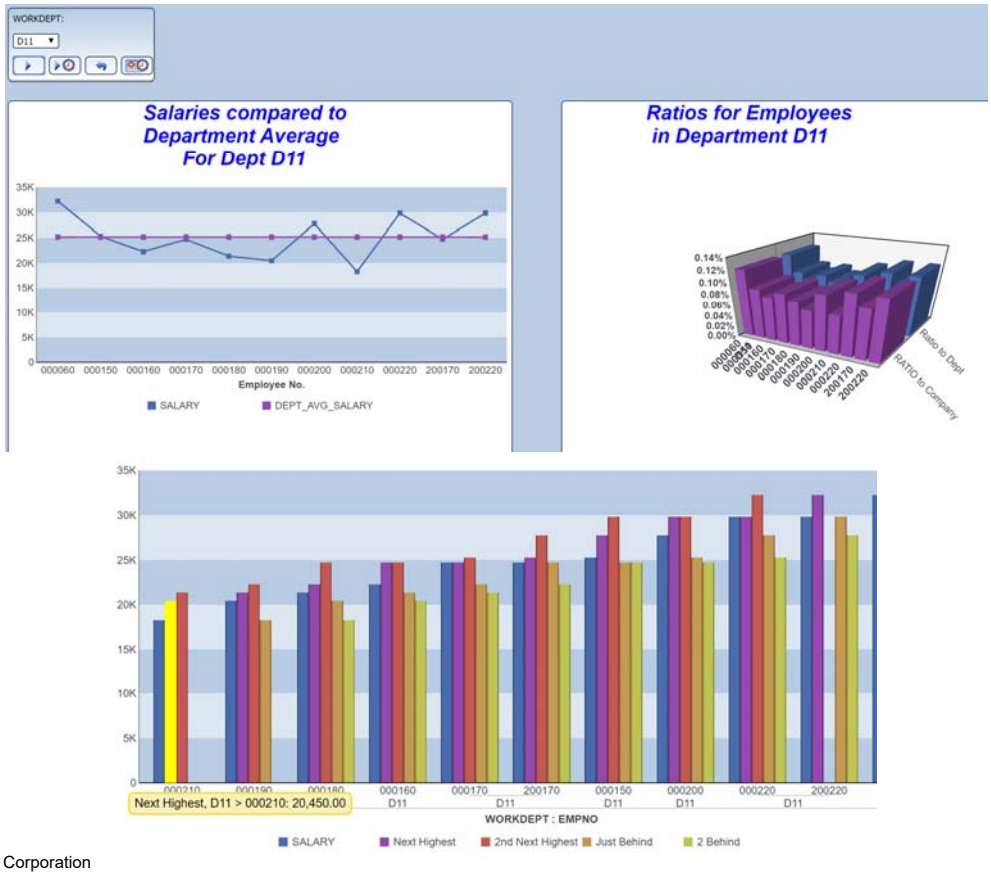


Examples

Comparison to
Average Salary of Dept

Dept	Employee No.	Salary	Comparison to average	Average Dept Salary	DELTA
A00	000010	52750.00	...IS ABOVE AVG	40850	11900
	000110	46500.00	...IS ABOVE AVG	40850	5650
	000120	29250.00	...IS BELOW AVG	40850	-11600
	200010	46500.00	...IS ABOVE AVG	40850	5650
	200120	29250.00	...IS BELOW AVG	40850	-11600
B01	000020	41250.00	...IS EQUAL TO AVG	41250	0
C01	000030	38250.00	...IS ABOVE AVG	29722	8527
	000130	23800.00	...IS BELOW AVG	29722	-5922
	000140	28420.00	...IS BELOW AVG	29722	-1302
	200140	28420.00	...IS BELOW AVG	29722	-1302
D11	000060	32250.00	...IS ABOVE AVG	25147	7102
	000150	25280.00	...IS ABOVE AVG	25147	132
	000160	22250.00	...IS BELOW AVG	25147	-2897
	000170	24680.00	...IS BELOW AVG	25147	-467
	000180	21340.00	...IS BELOW AVG	25147	-3807
	000190	20450.00	...IS BELOW AVG	25147	-4697
	000200	27740.00	...IS ABOVE AVG	25147	2592
	000210	18270.00	...IS BELOW AVG	25147	-6877
	000220	29840.00	...IS ABOVE AVG	25147	4692
	200170	24680.00	...IS BELOW AVG	25147	-467
	200220	29840.00	...IS ABOVE AVG	25147	4692
D21	000070	36170.00	...IS ABOVE AVG	25668	10501

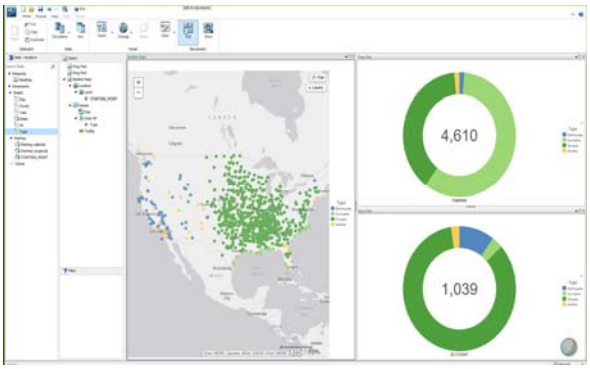
Examples



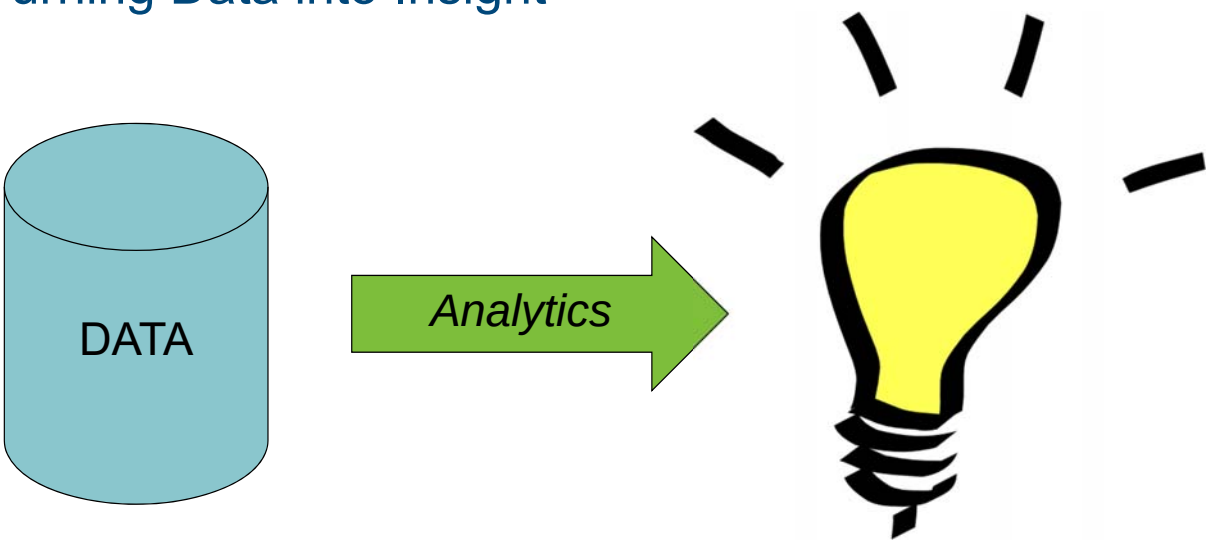
Db2 Web Query Version 2.2.1

- Steps beyond traditional Business Intelligence into **Data Discovery**
  - New data driven **Visualization** empowers:
    → Users, Analysts, and Data scientists
  - **Data layers** (e.g., demographics) for geographic maps
    → What is the average income in this zip code?
- Consolidate, Prepare, and Transform Data with **DataMigrator ETL**
  - Even augment existing data with data from Watson
- Install or upgrade in 15 minutes with the "EZ-Install" Package
  → Includes 100's of sample reports, for the business and I/T

Learn more at
ibm.biz/db2webqueryi
and
db2webqueryi.blogspot.com



Turning Data into Insight



OLAP is Online *Analytical* Processing

Rich function available directly on DB2 for i

- No need to move the data elsewhere!