THE UNIVERSITY OF UTAH

Advanced SQL Uses

Looking at what is available and how it can be used.

THE UNIVERSITY OF UTAH

Who’s talking here?

Graduated Idaho State University (2013)

Research Analyst

\+ DW Analyst

DG Analyst

Department’s Technology

- Oracle 12.1
- SQL Developer 18.4
- Tableau 2018.2

Goals for this session:

- Analytic Functions
- REGEXP (High level – Talk to Scott Heffron about his presentation for more information)
- Bind Variables

THE UNIVERSITY OF UTAH

Analytic Functions

Quick Review

Aggregate Function

- Compresses Rows
- Summarizes

DEPARTMENT_ID	MAX_DEPT_SALARY
10	4400
20	13000
30	11000
40	6500
50	8200

Analytic Function

- Maintains Rows
- Comparisons

DEPARTMENT_ID	MAX_DEPT_SALARY
10	4400
20	13000
20	13000
30	11000
30	11000
30	11000
30	11000
30	11000
30	11000
40	6500
50	8200

THE UNIVERSITY OF UTAH

Analytic Functions

Quick Review Cont.

Aggregate Function

```
SELECT e.department_id

     , MAX(e.salary)
     AS max_dept_salary

FROM hr.employees e
GROUP BY e.department_id
;
```

Analytic Function

```
SELECT e.department_id

     , MAX(e.salary)
       OVER (PARTITION BY
                  e.department_id
             )
       AS max_dept_salary

FROM hr.employees e
;
```

THE UNIVERSITY OF UTAH

Analytic Functions

Quick Review Cont.

Aggregate Function

```
SELECT e.department_id
     , MAX(e.salary)
     AS max_dept_salary

FROM hr.employees e
GROUP BY e.department_id
;
```

Analytic Function

```
SELECT e.department_id
     , MAX(e.salary)
       OVER (PARTITION BY
                   e.department_id
             )
       AS max_dept_salary

FROM hr.employees e
;
```

THE UNIVERSITY OF UTAH

Analytic Functions

Quick Review Cont.

Aggregate Function

```
SELECT e.department_id

     , MAX(e.salary)
     AS max_dept_salary

FROM hr.employees e
GROUP BY e.department_id
;
```

Analytic Function

```
SELECT e.department_id

     , MAX(e.salary)
       OVER (PARTITION BY
                   e.department_id
             )
       AS max_dept_salary

FROM hr.employees e
;
```

THE UNIVERSITY OF UTAH

Analytic Functions

Quick Review Cont.

Aggregate Function

```
SELECT e.department_id

     , MAX(e.salary)
     AS max_dept_salary

FROM hr.employees e
GROUP BY e.department_id
;
```

Analytic Function

```
SELECT e.department_id

     , MAX(e.salary)
       OVER (PARTITION BY
                    e.department_id
                  )
       AS max_dept_salary

FROM hr.employees e
;
```

THE UNIVERSITY OF UTAH

Analytic Functions

Quick Review Cont.

Aggregate Function

```
SELECT e.department_id
      , MAX(e.salary) AS
max_dept_salary
FROM hr.employees e
GROUP BY e.department_id
;
```

DEPARTMENT_ID	MAX_DEPT_SALARY
10	4400
20	13000
30	11000
40	6500
50	8200

Analytic Function

```
SELECT e.department_id
      , MAX(e.salary)
        OVER (PARTITION BY
        e.department_id) AS
max_dept_salary
FROM hr.employees e
;
```

DEPARTMENT_ID	MAX_DEPT_SALARY
10	4400
20	13000
20	13000
30	11000
30	11000
30	11000
30	11000
30	11000
30	11000
40	6500
50	8200

THE UNIVERSITY OF UTAH

When you use the RIGHT tool for the job you are more EFFICIENT.

Example of this principle:

Request: I want to see the AVERAGE compensation for each job code by year. I want to be able to COMPARE it with a person's current compensation.

Option 1: Use an AGGREGATE function in a NESTED query and JOIN it to the table

Option 2: Use an ANALYTIC function

THE UNIVERSITY OF UTAH

Option 1: Aggregated Nested Query

```
(
SELECT year
        , job_code
        , ROUND(AVG(total_compensation)) AS avg_job_code_ttl_comp
FROM data_gov_employee_compensation
GROUP BY year, job_code
) avg_ttl
```

	YEAR	JOB_CODE	AVG_JOB_CODE_TTL_COMP
1	2019	7371	1960
2	2019	1840	1878
3	2019	Q052	1967
4	2019	8141	1996
5	2019	1823	1917
6	2019	1044	1968
7	2019	1820	1935
8	2019	7335	1880
9	2019	7334	1883
10	2019	3610	1546
11	2019	1634	1893

THE UNIVERSITY OF UTAH

Option 1: Aggregated Nested Query

```
SELECT e.year, e.employee_identifier, e.job_code, e.department_code, e.department

, e.total_compensation
, avg_ttl.avg_job_code_ttl_comp
, ROUND(e.total_compensation) - avg_ttl.avg_job_code_ttl_comp AS diff_ttl_comp_from_avg

FROM data_gov_employee_compensation e
INNER JOIN (    SELECT year, job_code,
                        ROUND(AVG(total_compensation)) AS avg_job_code_ttl_comp
                    FROM data_gov_employee_compensation
                    GROUP BY year, job_code
                ) avg_ttl
ON e.year = avg_ttl.year AND e.job_code = avg_ttl.job_code
ORDER BY e.job_code, e.year DESC, e.total_compensation DESC,
e.department_code;
```

THE UNIVERSITY OF UTAH

Option 1: Aggregated Nested Query

```
SELECT e.year, e.employee_identifier, e.job_code, e.department_code, e.department

, e.total_compensation
, avg_ttl.avg_job_code_ttl_comp
, ROUND(e.total_compensation) - avg_ttl.avg_job_code_ttl_comp AS diff_ttl_comp_from_avg

FROM data_gov_employee_compensation e
INNER JOIN (    SELECT year, job_code,
                        ROUND(AVG(total_compensation)) AS avg_job_code_ttl_comp
                    FROM data_gov_employee_compensation
                    GROUP BY year, job_code
                ) avg_ttl
ON e.year = avg_ttl.year AND e.job_code = avg_ttl.job_code
ORDER BY e.job_code, e.year DESC, e.total_compensation DESC,
e.department_code;
```

THE UNIVERSITY OF UTAH

Option 1: Aggregated Nested Query

```
SELECT e.year, e.employee_identifier, e.job_code, e.department_code, e.department

, e.total_compensation
, avg_ttl.avg_job_code_ttl_comp
, ROUND(e.total_compensation) - avg_ttl.avg_job_code_ttl_comp AS diff_ttl_comp_from_avg

FROM data_gov_employee_compensation e
INNER JOIN (    SELECT year, job_code,
                        ROUND(AVG(total_compensation)) AS avg_job_code_ttl_comp
                    FROM data_gov_employee_compensation
                    GROUP BY year, job_code
                ) avg_ttl
ON e.year = avg_ttl.year AND e.job_code = avg_ttl.job_code
ORDER BY e.job_code, e.year DESC, e.total_compensation DESC,
e.department_code;
```

THE UNIVERSITY OF UTAH

Option 1: Aggregated Nested Query

```
SELECT e.year, e.employee_identifier, e.job_code, e.department_code, e.department

, e.total_compensation
, avg_ttl.avg_job_code_ttl_comp
, ROUND(e.total_compensation) - avg_ttl.avg_job_code_ttl_comp AS diff_ttl_comp_from_avg

FROM data_gov_employee_compensation e
INNER JOIN (    SELECT year, job_code,
                        ROUND(AVG(total_compensation)) AS avg_job_code_ttl_comp
                FROM data_gov_employee_compensation
                GROUP BY year, job_code
            ) avg_ttl
ON e.year = avg_ttl.year AND e.job_code = avg_ttl.job_code
ORDER BY e.job_code, e.year DESC, e.total_compensation DESC,
e.department_code; --489,155 rows
```

	YEAR	EMPLOYEE_IDENTIFIER	JOB_CODE	DEPARTMENT_CODE	DEPARTMENT	TOTAL_COMPENSATION	AVG_JOB_CODE_TTL_COMP	DIFF_TTL_COMP_FROM_AVG
1	2017	47157	0109	BOS	BOS Board Of Supervisors	9697.23	1279	8418
2	2017	25676	0109	BOS	BOS Board Of Supervisors	9228.13	1279	7949
3	2017	43883	0109	BOS	BOS Board Of Supervisors	7854.21	1279	6575
4	2017	24142	0109	BOS	BOS Board Of Supervisors	5854.94	1279	4576
5	2017	14105	0109	BOS	BOS Board Of Supervisors	5763.45	1279	4484
6	2017	47157	0109	BOS	BOS Board Of Supervisors	4901.36	1279	3622
7	2017	43883	0109	BOS	BOS Board Of Supervisors	3439.7	1279	2161
8	2017	12445	0109	BOS	BOS Board Of Supervisors	3316.28	1279	2037

THE UNIVERSITY OF UTAH

Option 2: Average using Analytic Function

```
SELECT e.year, e.employee_identifier, e.job_code,
       e.department_code, e.department

     , e.total_compensation

     , ROUND(AVG(e.total_compensation) OVER (PARTITION BY
       e.year, e.job_code)) AS avg_job_code_ttl_comp

     , ROUND(e.total_compensation) -     ROUND(AVG(e.total_compensation)
       OVER (PARTITION BY e.year, e.job_code)) AS diff_ttl_comp_from_avg

FROM data_gov_employee_compensation e
ORDER BY e.job_code, e.year DESC, e.total_compensation DESC,
e.department_code;
```

THE UNIVERSITY OF UTAH

Option 2: Average using Analytic Function

```
SELECT e.year, e.employee_identifier, e.job_code,
       e.department_code, e.department

    , e.total_compensation

    , ROUND(AVG(e.total_compensation) OVER (PARTITION BY
       e.year, e.job_code)) AS avg_job_code_ttl_comp

    , ROUND(e.total_compensation) -    ROUND(AVG(e.total_compensation)
       OVER (PARTITION BY e.year, e.job_code)) AS diff_ttl_comp_from_avg

FROM data_gov_employee_compensation e
ORDER BY e.job_code, e.year DESC, e.total_compensation DESC,
e.department_code; --489,155 rows
```

	YEAR	EMPLOYEE_IDENTIFIER	JOB_CODE	DEPARTMENT_CODE	DEPARTMENT	TOTAL_COMPENSATION	AVG_JOB_CODE_TTL_COMP	DIFF_TTL_COMP_FROM_AVG
1	2017	47157	0109	BOS	BOS Board Of Supervisors	9697.23	1279	8418
2	2017	25676	0109	BOS	BOS Board Of Supervisors	9228.13	1279	7949
3	2017	43883	0109	BOS	BOS Board Of Supervisors	7854.21	1279	6575
4	2017	24142	0109	BOS	BOS Board Of Supervisors	5854.94	1279	4576
5	2017	14105	0109	BOS	BOS Board Of Supervisors	5763.45	1279	4484
6	2017	47157	0109	BOS	BOS Board Of Supervisors	4901.36	1279	3622
7	2017	43883	0109	BOS	BOS Board Of Supervisors	3439.7	1279	2161
8	2017	12445	0109	BOS	BOS Board Of Supervisors	3316.28	1279	2037

THE UNIVERSITY OF UTAH

Which one was better?

Option 1: Aggregated Nest Query

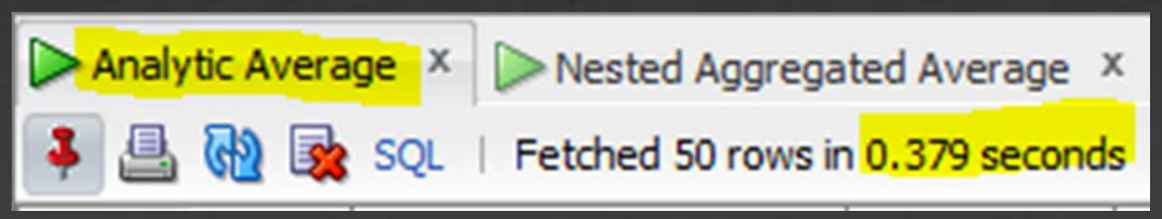
```
SELECT e.year
     , e.employee_identifier
     , e.job_code
     , e.department_code
     , e.department
     , e.total_compensation
     , avg_ttl.avg_job_code_ttl_comp
     ,      ROUND(e.total_compensation)
       -    avg_ttl.avg_job_code_ttl_comp
       AS   diff_ttl_comp_from_avg
FROM data_gov_employee_compensation e
INNER JOIN (     SELECT  year, job_code,
                         ROUND(AVG(total_compensation)) AS avg_job_code_ttl_comp
                 FROM data_gov_employee_compensation
                 GROUP BY year, job_code
           ) avg_ttl
ON e.year = avg_ttl.year AND e.job_code = avg_ttl.job_code
ORDER BY e.job_code, e.year DESC, e.total_compensation DESC, e.department_code
;
```

Option 2: Average using Analytic Function

```
SELECT e.year
     , e.employee_identifier
     , e.job_code
     , e.department_code
     , e.department
     , e.total_compensation

     , ROUND(AVG(e.total_compensation) OVER (PARTITION BY e.year, e.job_code))
       AS avg_job_code_ttl_comp

     ,      ROUND(e.total_compensation)
       -    ROUND(AVG(e.total_compensation) OVER (PARTITION BY e.year, e.job_code))
       AS   diff_ttl_comp_from_avg
FROM data_gov_employee_compensation e
ORDER BY e.job_code, e.year DESC, e.total_compensation DESC, e.department_code
;
```



Basic Analytic Function Uses/Questions Answered

- Difference from the average
- Percentage a field is of the total
- Dynamic current value flag
- Show previous or upcoming rows

THE UNIVERSITY OF UTAH

Percentage a field is of the total

```
SELECT  e.department_id
        , e.employee_id
        , e.salary
        , SUM(e.salary) OVER (PARTITION BY department_id)
              AS dept_salary_total
        , ROUND(e.salary/SUM(e.salary) OVER (PARTITION BY
              department_id) * 100) AS pct_dept_salary_total
FROM hr.employees e
ORDER BY e.department_id
        , e.salary DESC;
```

Percentage a field is of the total

```
SELECT  e.department_id
        , e.employee_id
        , e.salary
        , SUM(e.salary) OVER (PARTITION BY department_id)
              AS dept_salary_total
        , ROUND(e.salary/SUM(e.salary) OVER (PARTITION BY
department_id) * 100) AS pct_dept_salary_total
FROM hr.employees e
ORDER BY e.department_id
        , e.salary DESC;
```

DEPARTMENT_ID	EMPLOYEE_ID	SALARY	DEPT_SALARY_TOTAL	PCT_DEPT_SALARY_TOTAL
10	200	4400	4400	100
20	201	13000	19000	68
20	202	6000	19000	32
30	114	11000	24900	44
30	115	3100	24900	12
30	116	2900	24900	12
30	117	2800	24900	11
30	118	2600	24900	10
30	119	2500	24900	10
40	203	6500	6500	100

THE UNIVERSITY OF UTAH

Percentage a field is of the total

```
SELECT  e.department_id
        , e.employee_id
        , e.salary
        , SUM(e.salary) OVER (PARTITION BY department_id)
              AS dept_salary_total
        , ROUND(e.salary/SUM(e.salary) OVER (PARTITION BY
department_id) * 100) AS pct_dept_salary_total
FROM hr.employees e
ORDER BY e.department_id
        , e.salary DESC;
```

DEPARTMENT_ID	EMPLOYEE_ID	SALARY	DEPT_SALARY_TOTAL	PCT_DEPT_SALARY_TOTAL
10	200	4400	4400	100
20	201	13000	19000	68
20	202	6000	19000	32
30	114	11000	24900	44
30	115	3100	24900	12
30	116	2900	24900	12
30	117	2800	24900	11
30	118	2600	24900	10
30	119	2500	24900	10
40	203	6500	6500	100

THE UNIVERSITY OF UTAH

Dynamic current value

Example: Business glossary application table flags if a definition is the last approved definition, in an approval queue, or definition in a draft.

Problem: Departments want the option to see a definition version that is most recently updated.

Solution: Using MAX as an analytic function

	DEFINITION_NAME	VERSION_STATUS	VERSION_NUMBER	DEFINITION	VERSION_LATEST_APPROVED	VERSION_IN_QUEUE	VERSION_IN_DRAFT
1	Date of Birth	Approved	1	A person's date of birth.	0	0	0
2	Date of Birth	Approved	2	A person's date of birth.	1	0	0
3	Date of Birth	Drafting	3	The date and year a person was born.	0	0	1

THE UNIVERSITY OF UTAH

Dynamic current value

To get a new column that flags the most recent version of a definition without care to what phase it is I used an the MAX analytic function

```
CASE WHEN
      dv.version_number =

      MAX(dv.version_number)
OVER (PARTITION BY dv.definition_id)

THEN 1 ELSE 0 END AS latest_definition_version
```

	DEFINITION_NA...	VERSIO...		DEFINITION	VERSION_LATEST_APPROVED	VERSION_IN_QUEUE	VERSION_IN_DRAFT	LATEST_VERSION
1	Date of Birth	Approved	1	A person's date of birth.	0	0	0	0
2	Date of Birth	Approved	2	A person's date of birth.	1	0	0	0
3	Date of Birth	Drafting	3	The date and year a person was born.	0	0	1	1
4	Sr Administration	Approved	1	The highest level of reporting in the Organization heirarchy....	0	0	0	0
5	Sr Administrator	Approved	2	The highest level of reporting in the Organization heirarchy....	1	0	0	1

THE UNIVERSITY OF UTAH

Show previous or upcoming rows

Example: Students will often switch majors.

Problem 1: Department want to see the current major a student has as well as the previous major a student switched from.

Solution 1: Using LAG as an analytic function (previous row)

Problem 2: Department wants to approved definitions and compare it to any definitions that has a new version being drafted.

Solution 2: Using LEAD as an analytic function (upcoming row)

THE UNIVERSITY OF UTAH

Problem 1: Majors Movement
Show previous rows

LAG(major_group)
OVER (PARTITION BY emplid ORDER BY declare_dt)

EMPLOYEE_ID	PREVIOUS_DECLARE_DT	PREVIOUS_MAJOR_GROUP	DECLARE_DT	MAJOR_GROUP
4347	21-AUG-2017	Consumer & Community Studies	09-NOV-2018	Economics
4427	(null)	(null)	08-FEB-2018	Physician Assistant Studies
4436	(null)	(null)	06-JUN-2017	Non-Matriculated
4443	(null)	(null)	21-AUG-2017	Business Administration

THE UNIVERSITY OF UTAH

Problem 2: Compare current approved definition with new version being drafted Show upcoming rows

```
LEAD(dv.definition_name)
OVER (PARTITION BY dv.definition_id
                ORDER BY dv.version_number)
AS new_version_definition_name

LEAD(dv.definition)
      OVER (PARTITION BY dv.definition_id
                      ORDER BY dv.version_number)
      AS new_version_definition
```

	DEFINITION_NAME	NEW_VERSION_DEFINITION_NAME	DEFINITION	NEW_VERSION_DEFINITION
1	Date of Birth	Date of Birth	A person's date of birth.	The date and year a person was born.
2	Sr Administrator	(null)	The highest level of report...	(null)

THE UNIVERSITY OF UTAH

Problem 2: Compare current approved definition with new version being drafted Show upcoming rows

	DEFINITION_NA...	VERSIO...		DEFINITION	VERSION_LATEST_APPROVED	VERSION_IN_QUEUE	VERSION_IN_DRAFT	LATEST_VERSION
1	Date of Birth	Approved	1	A person's date of birth.	0	0	0	0
2	Date of Birth	Approved	2	A person's date of birth.	1	0	0	0
3	Date of Birth	Drafting	3	The date and year a person was born.	0	0	1	1
4	Sr Administration	Approved	1	The highest level of reporting in the Organization heirarchy....	0	0	0	0
5	Sr Administrator	Approved	2	The highest level of reporting in the Organization heirarchy....	1	0	0	1

	DEFINITION_NAME	NEW_VERSION_DEFINITION_NAME	DEFINITION	NEW_VERSION_DEFINITION
1	Date of Birth	Date of Birth	A person's date of birth.	The date and year a person was born.
2	Sr Administrator	(null)	The highest level of report...	(null)

Running Totals Uses/Questions Answered

- See current account balance with each transaction
- Indicate when expenses/revenue is over budgeted amount
- Rank individual’s performance

See current account balance with each transaction

```
SELECT transaction_id
      , transaction_dt
      , debit_or_credit
       , amount
       , SUM(amount)
          OVER (ORDER BY transaction_dt, transaction_id, amount)
         AS running_ttl_amount
FROM order_account
ORDER BY transaction_dt DESC, transaction_id DESC, amount
DESC;
```

TRANSACTION_ID	TRANSACTION_DT	DEBIT_OR_CREDIT	AMOUNT	RUNNING_TTL_AMOUNT
41694	31-DEC-01	D	68.16	98205831.21
41694	31-DEC-01	D	44.3	98205763.05
41694	31-DEC-01	D	28.77	98205718.75
41694	31-DEC-01	D	9.99	98205689.98
39620	31-DEC-01	D	9.67	98205679.99

THE UNIVERSITY OF UTAH

See current account balance with each transaction

```
SELECT transaction_id
      , transaction_dt
      , debit_or_credit
       , amount
       , SUM(amount)
          OVER (ORDER BY transaction_dt, transaction_id, amount)
         AS running_ttl_amount
FROM order_account
ORDER BY transaction_dt DESC, transaction_id DESC, amount
DESC;
```

TRANSACTION_ID	TRANSACTION_DT	DEBIT_OR_CREDIT	AMOUNT	RUNNING_TTL_AMOUNT
41694	31-DEC-01	D	68.16	98205831.21
41694	31-DEC-01	D	44.3	98205763.05
41694	31-DEC-01	D	28.77	98205718.75
41694	31-DEC-01	D	9.99	98205689.98
39620	31-DEC-01	D	9.67	98205679.99

THE UNIVERSITY OF UTAH

See current account balance with each transaction

```
SELECT transaction_id
        , transaction_dt
        , debit_or_credit
         , amount
         , SUM(amount)
            OVER (ORDER BY transaction_dt, transaction_id, amount)
           AS running_ttl_amount
FROM order_account
ORDER BY transaction_dt, transaction_id, amount;
```

TRANSACTION_ID	TRANSACTION_DT	DEBIT_OR_CREDIT	AMOUNT	RUNNING_TTL_AMOUNT
247	01-JAN-98	D	7.15	7.15
254	01-JAN-98	D	16.63	23.78
254	01-JAN-98	D	30.15	53.93
254	01-JAN-98	D	39.19	93.12
269	01-JAN-98	D	16.84	109.96

See current account balance with each transaction

```
SELECT transaction_id
        , transaction_dt
        , debit_or_credit
         , amount
         , SUM(amount)
            OVER (ORDER BY transaction_dt, transaction_id, amount)
           AS running_ttl_amount
FROM order_account
ORDER BY transaction_dt, transaction_id, amount;
```

TRANSACTION_ID	TRANSACTION_DT	DEBIT_OR_CREDIT	AMOUNT	RUNNING_TTL_AMOUNT
247	01-JAN-98	D	7.15	7.15
254	01-JAN-98	D	16.63	93.12
254	01-JAN-98	D	30.15	93.12
254	01-JAN-98	D	39.19	93.12
269	01-JAN-98	D	16.84	109.96

See current account balance with each transaction

```
SELECT transaction_id
        , transaction_dt
        , debit_or_credit
         , amount
         , SUM(amount)
            OVER (ORDER BY transaction_dt, transaction_id, amount)
           AS running_ttl_amount
FROM order_account
ORDER BY transaction_dt, transaction_id, amount;
```

TRANSACTION_ID	TRANSACTION_DT	DEBIT_OR_CREDIT	AMOUNT	RUNNING_TTL_AMOUNT
247	01-JAN-98	D	7.15	7.15
254	01-JAN-98	D	16.63	93.12
254	01-JAN-98	D	30.15	93.12
254	01-JAN-98	D	39.19	93.12
269	01-JAN-98	D	16.84	109.96

See current account balance with each transaction

```
SELECT transaction_id
        , transaction_dt
        , debit_or_credit
         , amount
         , SUM(amount)
            OVER (ORDER BY transaction_dt, transaction_id, amount)
           AS running_ttl_amount
FROM order_account
ORDER BY transaction_dt, transaction_id, amount;
```

TRANSACTION_ID	TRANSACTION_DT	DEBIT_OR_CREDIT	AMOUNT	RUNNING_TTL_AMOUNT
247	01-JAN-98	D	7.15	7.15
254	01-JAN-98	D	16.63	93.12
254	01-JAN-98	D	30.15	93.12
254	01-JAN-98	D	39.19	93.12
269	01-JAN-98	D	16.84	109.96
269	01-JAN-98	D	16.84	109.96

THE UNIVERSITY OF UTAH

Indicate when expenses/revenue is over budgeted amount

```
SELECT transaction_id
        , transaction_dt
        , debit_or_credit
         , amount
         , SUM(amount)
             OVER (ORDER BY transaction_dt, transaction_id, amount)
            AS running_ttl_amount
FROM order_account
ORDER BY transaction_dt, transaction_id, amount;
```

Indicate when expenses/revenue is over budgeted amount

```
SELECT   transaction_id
        , transaction_dt
        , debit_or_credit
          , amount
          , SUM(amount)
            OVER (ORDER BY transaction_dt, transaction_id, amount)
           AS running_ttl_amount
        , budget
          , CASE WHEN
              SUM(amount)
              OVER (ORDER BY transaction_dt, transaction_id, amount) >
budget
          THEN ‘Over’ END AS over_budget
FROM order_account o
INNER JOIN  budget_order b
ON o.fiscal_year = b.fiscal_year
ORDER BY transaction_dt, transaction_id, amount;
```

Indicate when expenses/revenue is over budgeted amount

```
, SUM(amount)
            OVER (ORDER BY transaction_dt, transaction_id, amount)
           AS running_ttl_amount

, budget

 , CASE WHEN
    SUM(amount)
    OVER (ORDER BY transaction_dt, transaction_id, amount) >
budget
   THEN ‘Over’ END AS over_budget
```

3929	01-JAN-98	D	30.87	993.94	1000	-
3931	02-JAN-98	D	16.63	1010.57	1000	Over
3931	02-JAN-98	D	30.15	1040.72	1000	Over
3960	03-JAN-98	D	16.63	1057.35	1000	Over

THE UNIVERSITY OF UTAH

Rank Examples

RANK

```
RANK()
OVER (PARTITION BY job_code
ORDER BY year, total_compensation
DESC)
```

DENSE_RANK

```
DENSE_RANK()
OVER (PARTITION BY job_code
ORDER BY year, total_compensation
DESC)
```

	YEAR	ORGANIZATION_GROUP_CODE	JOB_CODE	TOTAL_COMPENSATION	COMPENSATION_RANK
1	2019	01	H002	2701.28	1
2	2019	01	H020	2642.71	2
3	2019	01	H010	2525.57	3
4	2019	01	H003	2525.56	4
5	2019	01	H002	2525.56	4
6	2019	01	H002	2525.56	4
7	2019	01	H002	2525.56	4
8	2019	01	H002	2525.56	4
9	2019	01	H002	2525.56	4
10	2019	01	H002	2525.56	4
11	2019	01	H002	2525.56	4
12	2019	01	H002	2525.56	4
13	2019	01	H002	2525.56	4
14	2019	01	H002	2525.56	4
15	2019	01	H002	2525.56	4
16	2019	01	H002	2525.55	16
17	2019	01	H030	2525.55	16

	YEAR	ORGANIZATION_GROUP_CODE	JOB_CODE	TOTAL_COMPENSATION	COMPENSATION_RANK
1	2019	01	H002	2701.28	1
2	2019	01	H020	2642.71	2
3	2019	01	H010	2525.57	3
4	2019	01	H003	2525.56	4
5	2019	01	H002	2525.56	4
6	2019	01	H002	2525.56	4
7	2019	01	H002	2525.56	4
8	2019	01	H002	2525.56	4
9	2019	01	H002	2525.56	4
10	2019	01	H002	2525.56	4
11	2019	01	H002	2525.56	4
12	2019	01	H002	2525.56	4
13	2019	01	H002	2525.56	4
14	2019	01	H002	2525.56	4
15	2019	01	H002	2525.56	4
16	2019	01	H002	2525.55	5
17	2019	01	H030	2525.55	5

Rank Examples

RANK

```
RANK()
OVER (PARTITION BY job_code
ORDER BY year, total_compensation
DESC)
```

DENSE_RANK

```
DENSE_RANK()
OVER (PARTITION BY job_code
ORDER BY year, total_compensation
DESC)
```

	YEAR	ORGANIZATION_GROUP_CODE	JOB_CODE	TOTAL_COMPENSATION	COMPENSATION_RANK
1	2019	01	H002	2701.28	1
2	2019	01	H020	2642.71	2
3	2019	01	H010	2525.57	3
4	2019	01	H003	2525.56	4
5	2019	01	H002	2525.56	4
6	2019	01	H002	2525.56	4
7	2019	01	H002	2525.56	4
8	2019	01	H002	2525.56	4
9	2019	01	H002	2525.56	4
10	2019	01	H002	2525.56	4
11	2019	01	H002	2525.56	4
12	2019	01	H002	2525.56	4
13	2019	01	H002	2525.56	4
14	2019	01	H002	2525.56	4
15	2019	01	H002	2525.56	4
16	2019	01	H002	2525.55	16
17	2019	01	H030	2525.55	16

	YEAR	ORGANIZATION_GROUP_CODE	JOB_CODE	TOTAL_COMPENSATION	COMPENSATION_RANK
1	2019	01	H002	2701.28	1
2	2019	01	H020	2642.71	2
3	2019	01	H010	2525.57	3
4	2019	01	H003	2525.56	4
5	2019	01	H002	2525.56	4
6	2019	01	H002	2525.56	4
7	2019	01	H002	2525.56	4
8	2019	01	H002	2525.56	4
9	2019	01	H002	2525.56	4
10	2019	01	H002	2525.56	4
11	2019	01	H002	2525.56	4
12	2019	01	H002	2525.56	4
13	2019	01	H002	2525.56	4
14	2019	01	H002	2525.56	4
15	2019	01	H002	2525.56	4
16	2019	01	H002	2525.55	5
17	2019	01	H030	2525.55	5

THE UNIVERSITY OF UTAH

Rank Examples

RANK

```
RANK()
OVER (PARTITION BY job_code
ORDER BY year, total_compensation
DESC)
```

DENSE_RANK

```
DENSE_RANK()
OVER (PARTITION BY job_code
ORDER BY year, total_compensation
DESC)
```

	YEAR	ORGANIZATION_GROUP_CODE	JOB_CODE	TOTAL_COMPENSATION	COMPENSATION_RANK
1	2019	01	H002	2701.28	1
2	2019	01	H020	2642.71	2
3	2019	01	H010	2525.57	3
4	2019	01	H003	2525.56	4
5	2019	01	H002	2525.56	4
6	2019	01	H002	2525.56	4
7	2019	01	H002	2525.56	4
8	2019	01	H002	2525.56	4
9	2019	01	H002	2525.56	4
10	2019	01	H002	2525.56	4
11	2019	01	H002	2525.56	4
12	2019	01	H002	2525.56	4
13	2019	01	H002	2525.56	4
14	2019	01	H002	2525.56	4
15	2019	01	H002	2525.56	4
16	2019	01	H002	2525.55	16
17	2019	01	H030	2525.55	16

	YEAR	ORGANIZATION_GROUP_CODE	JOB_CODE	TOTAL_COMPENSATION	COMPENSATION_RANK
1	2019	01	H002	2701.28	1
2	2019	01	H020	2642.71	2
3	2019	01	H010	2525.57	3
4	2019	01	H003	2525.56	4
5	2019	01	H002	2525.56	4
6	2019	01	H002	2525.56	4
7	2019	01	H002	2525.56	4
8	2019	01	H002	2525.56	4
9	2019	01	H002	2525.56	4
10	2019	01	H002	2525.56	4
11	2019	01	H002	2525.56	4
12	2019	01	H002	2525.56	4
13	2019	01	H002	2525.56	4
14	2019	01	H002	2525.56	4
15	2019	01	H002	2525.56	4
16	2019	01	H002	2525.55	5
17	2019	01	H030	2525.55	5

THE UNIVERSITY OF UTAH

LISTAGG

Combining values into one field for a defined group

DEPARTMENT_ID	FIRST_NAME	LAST_NAME
10	Jennifer	Whalen
20	Michael	Hartstein
20	Pat	Fay
30	Den	Raphaely
30	Alexander	Khoo
30	Shelli	Baida
30	Sigal	Tobias

DEPARTMENT_ID	EMPLOYEE
10	Jennifer Whalen
20	Michael Hartstein,Pat Fay
30	Alexander Khoo,Den Raphaely,Guy Himuro,Karen Colmenares,Shelli Baida,Sigal Tobias
40	Susan Mavris

THE UNIVERSITY OF UTAH

LISTAGG

Combining values into one field for a defined group

DEPARTMENT_ID	FIRST_NAME	LAST_NAME
10	Jennifer	Whalen
20	Michael	Hartstein
20	Pat	Fay
30	Den	Raphaely
30	Alexander	Khoo
30	Shelli	Baida
30	Sigal	Tobias

```
SELECT department_id
     , LISTAGG(last_name,',')
       WITHIN GROUP
       (ORDER BY last_name)
         AS employee
FROM hr.employees
GROUP BY department_id
ORDER BY department_id;
```

DEPARTMENT_ID	
10	Whalen
20	Fay,Hartstein
30	Baida,Colmenares,Himuro,Khoo,Raphaely,Tobias
40	Mavris

THE UNIVERSITY OF UTAH

LISTAGG

Combining values into one field for a defined group

DEPARTMENT_ID	FIRST_NAME	LAST_NAME
10	Jennifer	Whalen
20	Michael	Hartstein
20	Pat	Fay
30	Den	Raphaely
30	Alexander	Khoo
30	Shelli	Baida
30	Sigal	Tobias

```
SELECT department_id
     , LISTAGG(last_name,',')
       WITHIN GROUP
       (ORDER BY last_name)
         AS employee
FROM hr.employees
GROUP BY department_id
ORDER BY department_id;
```

DEPARTMENT_ID	
10	Whalen
20	Fay,Hartstein
30	Baida,Colmenares,Himuro,Khoo,Raphaely,Tobias
40	Mavris

THE UNIVERSITY OF UTAH

LISTAGG

Combining values into one field for a defined group

DEPARTMENT_ID	FIRST_NAME	LAST_NAME
10	Jennifer	Whalen
20	Michael	Hartstein
20	Pat	Fay
30	Den	Raphaely
30	Alexander	Khoo
30	Shelli	Baida
30	Sigal	Tobias

```
SELECT department_id
     , LISTAGG(last_name,',')
       WITHIN GROUP
       (ORDER BY last_name)
         AS employee
FROM hr.employees
GROUP BY department_id
ORDER BY department_id;
```

DEPARTMENT_ID	
10	Whalen
20	Fay,Hartstein
30	Baida,Colmenares,Himuro,Khoo,Raphaely,Tobias
40	Mavris

THE UNIVERSITY OF UTAH

LISTAGG

Combining values into one field for a defined group

DEPARTMENT_ID	FIRST_NAME	LAST_NAME
10	Jennifer	Whalen
20	Michael	Hartstein
20	Pat	Fay
30	Den	Raphaely
30	Alexander	Khoo
30	Shelli	Baida
30	Sigal	Tobias

```
SELECT department_id
     , LISTAGG(last_name,',')
       WITHIN GROUP
       (ORDER BY last_name)
         AS employee
FROM hr.employees
GROUP BY department_id
ORDER BY department_id;
```

DEPARTMENT_ID	
10	Whalen
20	Fay,Hartstein
30	Baida,Colmenares,Himuro,Khoo,Raphaely,Tobias
40	Mavris

THE UNIVERSITY OF UTAH

DE-LISTAGG ???

I want the data uncompressed

DEPARTMENT_ID	
10	Whalen
20	Fay,Hartstein
30	Baida,Colmenares,Himuro,Khoo,Raphaely,Tobias
40	Mavris

DE-LISTAGG - Use REGEXP

```
SELECT *FROM (
        WITH occurence AS (
        --Determines the max number of values (employee) in the list seperated by a comma
                SELECT LEVEL AS ocr
                 FROM (   SELECT MAX(REGEXP_COUNT(employee, '[^,]+',1)) AS mx
                          FROM (SELECT department_id                                    ,
                          LISTAGG(last_name,',') WITHIN GROUP (ORDER BY last_name) AS employee
                          FROM hr.employees
                          GROUP BY department_id
                          ORDER BY department_id
                                )
                           CONNECT BY LEVEL <= mx
                        )
 SELECT h.department_id,
             TRIM( REGEXP_SUBSTR( h.employee, '[^,]+', 1, o.ocr)) AS delistagg_employee
FROM (
                SELECT department_id
                        , LISTAGG(last_name,',') WITHIN GROUP (ORDER BY last_name) AS employee
                FROM hr.employees
                GROUP BY department_id
                ORDER BY department_id
           ) h
CROSS JOIN occurence o
                )
WHERE delistagg_employee IS NOT NULL;
```

DE-LISTAGG - Use REGEXP

```
SELECT *FROM (
        WITH occurence AS (
        --Determines the max number of values (employee) in the list seperated by a comma
                SELECT LEVEL AS ocr
                 FROM (   SELECT MAX(REGEXP_COUNT(employee, '[^,]+',1)) AS mx
                          FROM (SELECT department_id                                    ,
                          LISTAGG(last_name,',') WITHIN GROUP (ORDER BY last_name) AS employee
                          FROM hr.employees
                          GROUP BY department_id
                          ORDER BY department_id
                                )
                          CONNECT BY LEVEL <= mx
                      )
 SELECT h.department_id,
             TRIM( REGEXP_SUBSTR( h.employee, '[^,]+', 1, o.ocr)) AS delistagg_employee
FROM (
                SELECT department_id
                          , LISTAGG(last_name,',') WITHIN GROUP (ORDER BY last_name) AS employee
                FROM hr.employees
                GROUP BY department_id
                ORDER BY department_id
           ) h
CROSS JOIN occurence o
                )
WHERE delistagg_employee IS NOT NULL;
```

DE-LISTAGG - Use REGEXP

--Determines the max number of values (employee) in the list separated by a comma

```
SELECT LEVEL AS ocr
FROM (   SELECT
              MAX(REGEXP_COUNT(employee, '[^,]+',1)) AS mx
              FROM (View with LISTAGG column)
              )
CONNECT BY LEVEL <= mx
```

DE-LISTAGG - Use REGEXP

--Determines the max number of values (employee) in the list separated by a comma

```
SELECT LEVEL AS ocr
FROM (   SELECT
             MAX(REGEXP_COUNT(employee, '[^,]+',1)) AS mx
             FROM (View with LISTAGG column)
         )
CONNECT BY LEVEL <= mx
```

OCR
1
2
3
4

DE-LISTAGG - Use REGEXP

```
SELECT *FROM (
        WITH occurence AS (
        --Determines the max number of values (employee) in the list seperated by a comma
                SELECT LEVEL AS ocr
                 FROM (   SELECT MAX(REGEXP_COUNT(employee, '[^,]+',1)) AS mx
                          FROM (SELECT department_id                                    ,
                          LISTAGG(last_name,',') WITHIN GROUP (ORDER BY last_name) AS employee
                          FROM hr.employees
                          GROUP BY department_id
                          ORDER BY department_id
                                )
                          CONNECT BY LEVEL <= mx
                        )
 SELECT h.department_id,
            TRIM( REGEXP_SUBSTR( h.employee, '[^,]+', 1, o.ocr)) AS delistagg_employee
FROM (
                SELECT department_id
                          , LISTAGG(last_name,',') WITHIN GROUP (ORDER BY last_name) AS employee
                FROM hr.employees
                GROUP BY department_id
                ORDER BY department_id
           ) h
CROSS JOIN occurence o
                )
WHERE delistagg_employee IS NOT NULL;
```

DE-LISTAGG - Use REGEXP

```
SELECT h.department_id,
        TRIM( REGEXP_SUBSTR( h.employee, '[^,]+', 1, o.ocr))
AS delistagg_employee
FROM (
    View with LISTAGG column
        ) h
CROSS JOIN occurence o
            )
WHERE delistagg_employee IS NOT NULL;
```

DE-LISTAGG - Use REGEXP

```
SELECT h.department_id,
          TRIM( REGEXP_SUBSTR( h.employee, '[^,]+', 1, o.ocr))
AS delistagg_employee
FROM (
     View with LISTAGG column
        ) h
CROSS JOIN occurence o
          )
WHERE delistagg_employee IS NOT NULL;
```

DE-LISTAGG - Use REGEXP

```
SELECT h.department_id,
          TRIM( REGEXP_SUBSTR( h.employee, '[^,]+', 1, o.ocr))
AS delistagg_employee
FROM (
     View with LISTAGG column
        ) h
CROSS JOIN occurence o
          )
WHERE delistagg_employee IS NOT NULL;
```

DEPARTMENT_ID	DELISTAGG_EMPL
10	Whalen
20	Fay
20	Hartstein
30	Baida
30	Tobias

Bind Variable

Hard Parse

- Syntax Check
- Semantic Check
- Optimizer
- Execution Plan

Soft Parse

- Syntax Check
- Semantic Check
- Skip - Optimizer
- Reuse - Execution Plan

```
select * from orders where order_id = 1234;
select * from orders where order_id = 9876;
```

Bind Variable

Hard Parse

- Syntax Check
- Semantic Check
- Optimizer
- Execution Plan

Soft Parse

- Syntax Check
- Semantic Check
- Skip - Optimizer
- Reuse - Execution Plan

```
select * from orders where order_id = 1234;
select * from orders where order_id = 9876;
```

Bind Variable

Hard Parse

- Syntax Check
- Semantic Check
- Optimizer
- Execution Plan

Soft Parse

- Syntax Check
- Semantic Check
- Skip - Optimizer
- Reuse - Execution Plan

```
select * from orders where order_id = 1234;
select * from orders where order_id = 9876;
```

Bind Variable

Hard Parse

- Syntax Check
- Semantic Check
- Optimizer
- Execution Plan

Soft Parse

- Syntax Check
- Semantic Check
- Skip - Optimizer
- Reuse - Execution Plan

```
select * from orders where order_id = :ord;
```

Bind Variable

Hard Parse

- Syntax Check
- Semantic Check
- Optimizer
- Execution Plan

Soft Parse

- Syntax Check
- Semantic Check
- Skip - Optimizer
- Reuse - Execution Plan

```
select * from orders where order_id = :ord;
```

Bind Variable Uses

- Repeated views that are often filtered down
- Dynamic SQL – User Defined Extension

THE UNIVERSITY OF UTAH

User Defined Extensions

Tools > Preference > Database > User Defined Extensions

https://www.oracle.com/webfolder/technetwork/tutorials/obe/db/sqldev/r30/srccodexmlext/xmlext_otn.htm

QUESTIONS????