Agenda

1. Review
2. Multi Row Functions
3. aggregating data using group functions
4. obtaining summary information (such as averages) for groups of rows.
5. Leveraging the GROUP BY ….. HAVING clauses to create and filter groups
6. how to group rows in a table into smaller sets and - how to specify search criteria for groups of rows.

SQL Functions

- What is another name for Single-row functions?
- Scaler functions
- What is another name for Multiple-row functions?
- Aggregate functions
- What is the difference between the two?
- A single-row function returns one result for each row. These functions operate on single rows only and return one result for every row acted on.
- A multiple-row function returns one result per set of rows. Functions can manipulate groups of rows to give one result per group of rows. These functions are also called group functions.

Single Row Functions Categories

- Character Functions
- Numeric Functions
- Datetime Functions
- Conversion Functions

Using Character Functions

- Provide the SQL to produce a column that extracts the first name and the intials from LREMPLOYEE. The NAME column exists, FNAMEFINITIAL is virtual

EMPNUM	NAME	FNAMEFINITIAL
513,333,333	Badilla, Guy G.	Guy G.
272,333,333	Bailey, Evelyn S.	Evelyn S.
966,666,666	Baker, Pandora M.	Pandora M.
334,333,333	Bender, Alexander O.	Alexander O.
956,789,000	Benjamin, Hillary R.	Hillary R.
912,345,678	Berg, Hadley B.	Hadley B.
243,243,433	Bolton, Tana J.	Tana J.
333,333,333	Bowers, Berk I.	Berk I.
344,444,444	Brady, Fitzgerald S.	Fitzgerald S.
622,222,222	Cabrera, Vivien W.	Vivien W.
476,666,666	Calhoun, Lael U.	Lael U.
755,555,555	Callahan, Leila K.	Leila K.

Using Character Functions

- SUBSTR is used
- INSTR is used

```
SELECT EMPNUM,
       NAME,
       SUBSTR(NAME, INSTR(NAME, ',') + 2) FNAMEFINITIAL
FROM   LREMPLOYEE
```

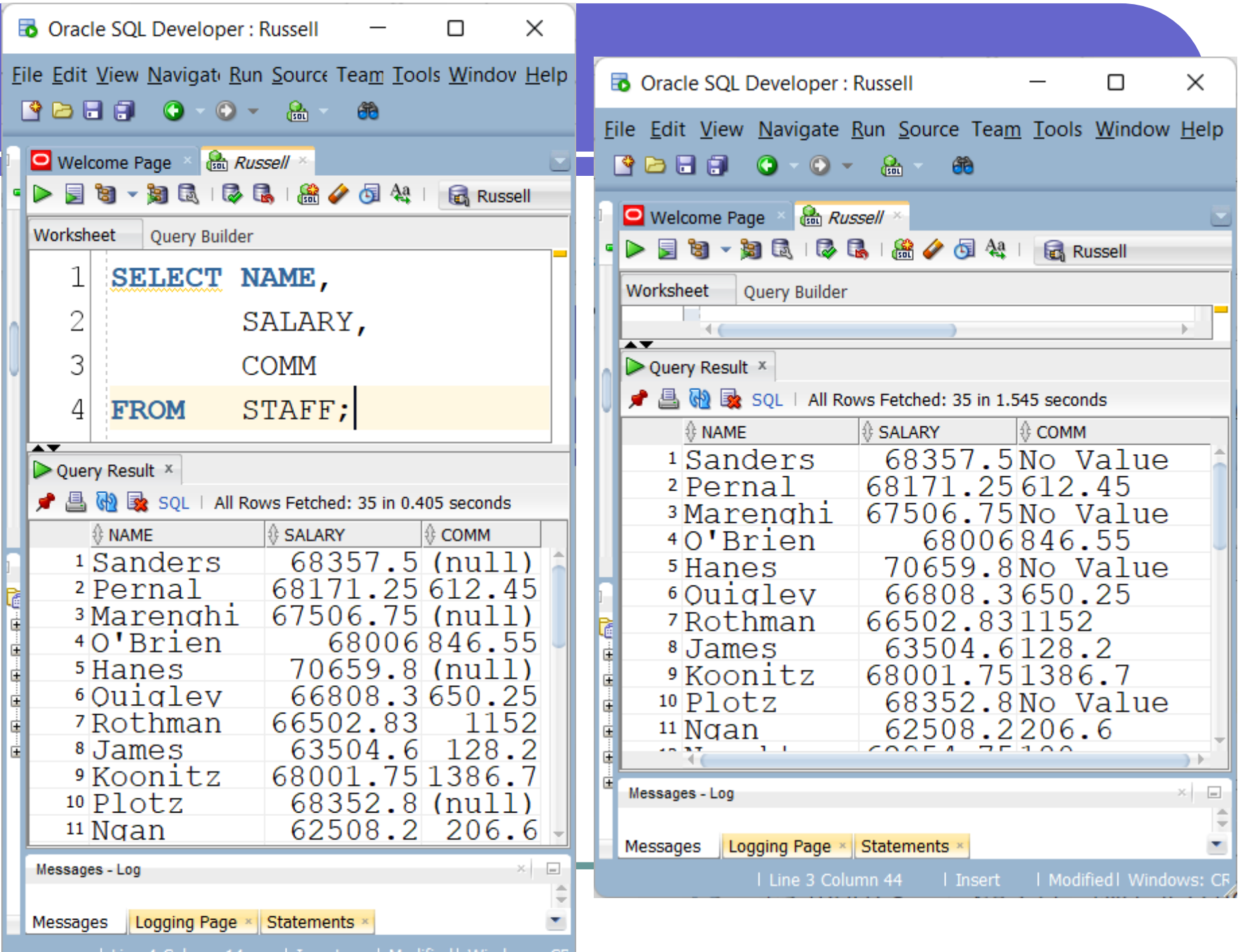
Using Character Functions

- Extract the last name into an additional virtual column and convert the 50 character NAME column to a 20 character field and only grab the first 20 characters of FNAMEFINITIAL

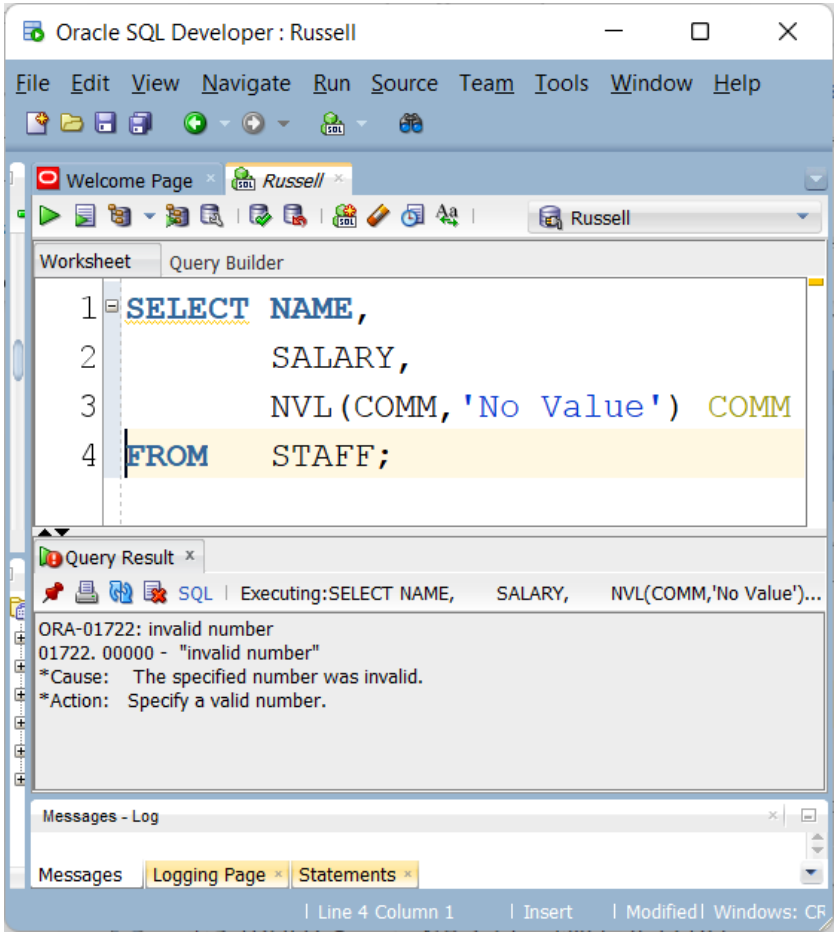
EMPNUM	NAME	FNAMEFINITIAL	LASTNAME
513,333,333	Badilla, Guy G.	Guy G.	Badilla
272,333,333	Bailey, Evelyn S.	Evelyn S.	Bailey
966,666,666	Baker, Pandora M.	Pandora M.	Baker
334,333,333	Bender, Alexander O.	Alexander O.	Bender
956,789,000	Benjamin, Hillary R.	Hillary R.	Benjamin
912,345,678	Berg, Hadley B.	Hadley B.	Berg
243,243,433	Bolton, Tana J.	Tana J.	Bolton
333,333,333	Bowers, Berk I.	Berk I.	Bowers
344,444,444	Brady, Fitzgerald S.	Fitzgerald S.	Brady
622,222,222	Cabrera, Vivien W.	Vivien W.	Cabrera
476,666,666	Calhoun, Lael U.	Lael U.	Calhoun
755,555,555	Callahan, Leila K.	Leila K.	Callahan

- Two different methods to shorten the column width on the screen
- SUBSTR
- CAST

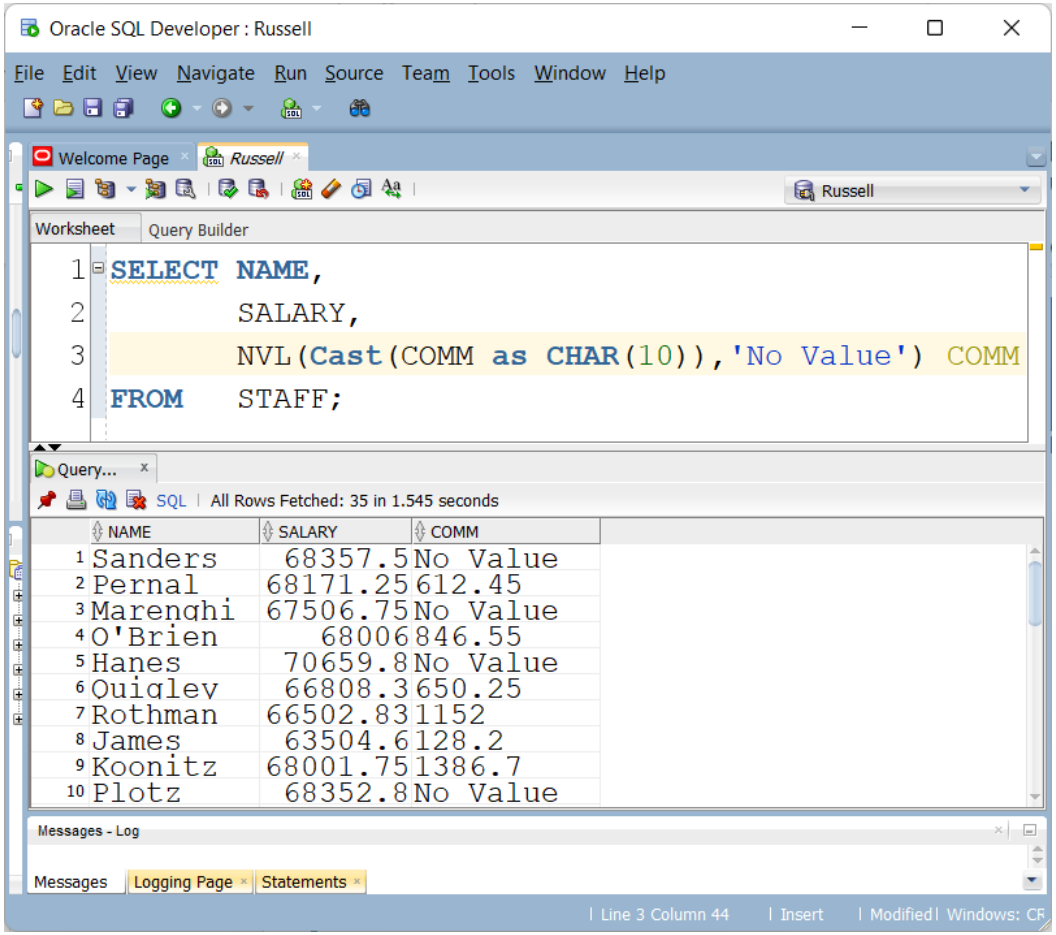
```
SELECT EMPNUM,
       CAST(NAME AS CHAR(20) ) NAME,
       SUBSTR(NAME, INSTR(NAME, ',') + 2)  FNAMEFINITIAL,
       SUBSTR(NAME,1, INSTR(NAME, ',') -1) LASTNAME
FROM   LREMPLOYEE
```


Oracle SQL Developer : Russell
File Edit View Navigat Run Source Team Tools Windov Help
Welcome Page
Russell
Russell
Worksheet
Query Builder
1 SELECT NAME,
2 SALARY,
3 COMM
4 FROM STAFF;
Query Result
SQL | All Rows Fetched: 35 in 0.405 seconds
NAME SALARY COMM
1 Sanders 68357.5 (null)
2 Pernal 68171.25 612.45
3 Marenghi 67506.75 (null)
4 O'Brien 68006 846.55
5 Hanes 70659.8 (null)
6 Quigley 66808.3 650.25
7 Rothman 66502.83 1152
8 James 63504.6 128.2
9 Koonitz 68001.75 1386.7
10 Plotz 68352.8 (null)
11 Ngan 62508.2 206.6
Messages - Log
Messages
Logging Page
Statements
Oracle SQL Developer : Russell
File Edit View Navigate Run Source Team Tools Window Help
Welcome Page
Russell
Russell
Worksheet
Query Builder
Query Result
SQL | All Rows Fetched: 35 in 1.545 seconds
NAME SALARY COMM
1 Sanders 68357.5 No Value
2 Pernal 68171.25 612.45
3 Marenghi 67506.75 No Value
4 O'Brien 68006 846.55
5 Hanes 70659.8 No Value
6 Quigley 66808.3 650.25
7 Rothman 66502.83 1152
8 James 63504.6 128.2
9 Koonitz 68001.75 1386.7
10 Plotz 68352.8 No Value
11 Ngan 62508.2 206.6
Messages - Log
Messages
Logging Page
Statements
Line 3 Column 44
Insert
Modified
Windows: CR

NVL on a numeric field



Solve using Cast



Adding Salary and Comm

ID	NAME	SALARY	COMM	TOTALSALARY
10	Sanders	68,357.50	-	-
20	Pernal	68,171.25	612.45	68,783.70
30	Marenghi	67,506.75	-	-
40	O'Brien	68,006.00	846.55	68,852.55
50	Hanes	70,659.80	-	-
60	Quigley	66,808.30	650.25	67,458.55
70	Rothman	66,502.83	1,152.00	67,654.83
80	James	63,504.60	128.20	63,632.80
90	Koonitz	68,001.75	1,386.70	69,388.45
00	Plotz	68,352.80	-	-
10	Ngan	62,508.20	206.60	62,714.80
20	Naughton	62,954.75	180.00	63,134.75
30	Yamaguchi	60,505.90	75.60	60,581.50
40	Fraye	71,150.00	-	-
50	Williams	69,456.50	637.65	70,094.15
60	Molinare	72,959.20	-	-
70	Kermisch	62,258.50	110.10	62,368.60
80	Abrahams	62,009.75	236.50	62,246.25
90	Sneider	64,252.75	126.50	64,379.25

- To get this

ID	NAME	SALARY	COMM	TOTALSALARY
10	Sanders	68,357.50	-	68,357.50
20	Pernal	68,171.25	612.45	68,783.70
30	Marenghi	67,506.75	-	67,506.75
40	O'Brien	68,006.00	846.55	68,852.55
50	Hanes	70,659.80	-	70,659.80
60	Quigley	66,808.30	650.25	67,458.55

```
SELECT "ID",
       NAME,
       SALARY,
       COMM,
       SALARY + NVL(COMM,0) TOTALSALARY
FROM STAFF
```

Conversion Functions

- Conversions are about changing the data type of a column.
- This may be required to bridge between one SQL statement and another – or – to bridge between the way an application works versus the way a database is designed (for example: application performs an arithmetic operation on numerical values stored as a CHAR or VARCHAR in the datebase)
- This cannot be performed in an ALTER TABLE
- This is about changing the data type in a result set
- Some data types allow for conversions, some conditionally allow for conversions and some do not.
- An example of a valid data type conversion, which will always work: INTEGER to CHAR/VARCHAR
- An example of a valid data type conversion, which may not work: CHAR/VARCHAR to INTEGER
- An example of a valid data type conversion, which may produce a warning: VARCHAR to CHAR
- An example of an invalid data type conversion: INTEGER to BINARY

CAST

```
SELECT SALARY * 1.15,
       CAST(ROUND(SALARY * 1.15,1) AS DECIMAL(7,1)) SALARYROUND
FROM EMPLOYEE
```

SALARY * 1.15	SALARYROUND
83,662.5000	83,662.5
70,437.5000	70,437.5
66,987.5000	66,987.5
69,201.2500	69,201.3
60,087.5000	60,087.5
64,595.5000	64,595.5
57,212.5000	57,212.5

CAST

- EMPNO is character, change to decimal

```
select empno,
       cast(empno as decimal (10,2) )
from   employee
```

```
EMPNO   CAST function
000010          10.00
000020          20.00
000030          30.00
000050          50.00
000060          60.00
000070          70.00
```

CAST

- A cast fails

```
select workdept,
       cast(workdept as decimal (10,2) )
from   employee

 WORKDEPT  CAST function
   A00     ++++++++++++++
   B01     ++++++++++++++
   C01     ++++++++++++++
   E01     ++++++++++++++
```

General Comparison Functions

- Comparison functions allow for two or more values to be compared during a SQL statement to help define the desired result set. There is a large set of comparison functions.
- >, <=, >=, =, !=, <>
- AND
- OR
- IN
- BETWEEN
- NOT BETWEEN
- LIKE
- NOT LIKE
- IS
- IS NOT

Not equal to

```
SELECT * FROM PATIENT
WHERE PATIENTNO <> 123
```

```
SELECT * FROM PATIENT
WHERE PATIENTNO != 123
```

- Results in

PATIENTNO	FIRSTNAME	LASTNAME	PINSURNUM
456	Bill	Trimble	666
789	Tom	Seaver	888
246	John	Howard	0

- What is the difference?

```
SELECT EMPNO, FIRSTNME,
       LASTNAME, SEX,
       SALARY, salary + comm
FROM employee
WHERE sex = 'F' and (salary + comm) > 60000
```

```
SELECT EMPNO, FIRSTNME,
       LASTNAME, SEX,
       SALARY, salary + comm
FROM employee
WHERE sex = 'F' or  (salary + comm) > 60000
```

Only include depts 20, 38 & 51

```
SELECT DEPT,
       NAME,
       SALARY,
       NVL(CAST(COMM AS CHAR(10)),'No Value') COMM,
       SALARY + NVL(COMM,0) AS TOTALSAL
FROM   STAFF
WHERE DEPT IN(20,51,38)
```

- Use IN

DEPT	NAME	SALARY	COMM	TOTALSAL
20	Sanders	68,357.50	No Value	68,357.50
20	Pernal	68,171.25	612.45	68,783.70
38	Marenghi	67,506.75	No Value	67,506.75
38	O'Brien	68,006.00	846.55	68,852.55
38	Quigley	66,808.30	650.25	67,458.55
20	James	63,504.60	128.20	63,632.80
38	Naughton	62,954.75	180.00	63,134.75
51	Fraye	71,150.00	No Value	71,150.00
51	Williams	69,456.50	637.65	70,094.15
38	Abrahams	62,009.75	236.50	62,246.25
20	Sneider	64,252.75	126.50	64,379.25
51	Smith	67,654.50	992.80	68,647.30
51	Lundquist	63,369.80	189.65	63,559.45

All Depts except 20, 38 & 51

```
SELECT DEPT,
       NAME,
       SALARY,
       NVL(CAST(COMM AS CHAR(10)),'No Value') COMM,
       SALARY + NVL(COMM,0) AS TOTALSAL
FROM   STAFF
WHERE  NOT DEPT IN(20,51,38)
```

- Use Not In

DEPT	NAME	SALARY	COMM	TOTALSAL
15	Hanes	70,659.80	No Value	70,659.80
15	Rothman	66,502.83	1152.00	67,654.83
42	Koonitz	68,001.75	1386.70	69,388.45
42	Plotz	68,352.80	No Value	68,352.80
15	Ngan	62,508.20	206.60	62,714.80
42	Yamaguchi	60,505.90	75.60	60,581.50
10	Molinare	72,959.20	No Value	72,959.20
15	Kermisch	62,258.50	110.10	62,368.60
42	Scoutten	61,508.60	84.20	61,592.80
10	Lu	70,010.00	No Value	70,010.00
10	Daniels	69,260.25	No Value	69,260.25
10	Jones	71,234.00	No Value	71,234.00
66	Lea	68,555.50	No Value	68,555.50
66	Wilson	68,674.50	811.50	69,486.00

BETWEEN

```
SELECT * FROM EMPLOYEE
WHERE  SALARY BETWEEN 40000 AND 50000
```

```
SELECT * FROM EMPLOYEE
WHERE  SALARY NOT BETWEEN 40000 AND 50000
```

LIKE and %

SELECT LASTNAME FROM EMPLOYEE
WHERE LASTNAME LIKE '%A’

```
LASTNAME
QUINTANA
PIANKA
YOSHIMURA
MEHTA
```

```
SELECT LASTNAME FROM EMPLOYEE
WHERE  LASTNAME NOT LIKE '%A'
```

```
LASTNAME
HAAS
THOMPSON
KWAN
GEYER
STERN
PULASKI
HENDERSON
SPENSER
```

LIKE and %

- Anybody that has a last name that begins with an A

```
SELECT LASTNAME FROM EMPLOYEE
WHERE  LASTNAME  LIKE 'A%'
```

- Anybody that has a last name that has an A somewhere in the name

```
SELECT LASTNAME FROM EMPLOYEE
WHERE  LASTNAME  LIKE '%A%'
```

- If you are uncertain about the case for all the names – could be a variety of combinations

```
SELECT * FROM EMPLOYEE
WHERE  LOWER(LASTNAME) LIKE '%a'
```

NULL

```
SELECT NAME,
       COMM
FROM   STAFF
WHERE  COMM IS NULL
```

NAME	COMM
Sanders	-
Marenghi	-
Hanes	-
Plotz	-
Fraye	-
Molinare	-
Lu	-
Daniels	-
Jones	-
Lea	-
Quill	-

Group Functions

- Group functions (multi-row functions)
- Operate on sets of rows to give one result per group.
- These sets may comprise the entire table or the table split into groups
- SELECT AVG(SALARY) FROM STAFF …..
- Every row with SALARY value is summed up and then total divided by number of rows, producing a SINGLE result

Group Functions Examples

- AVG
- COUNT
- MAX
- MIN
- SUM
- STDDEV
- VARIANCE

Number of countries in LREMPLOYEE

```
SELECT COUNT(COUNTRY) TOTALCOUNTRY FROM LREMPLOYEE

 TOTALCOUNTRY
          100
******** End of data ********

SELECT COUNTRY FROM LREMPLOYEE
ORDER BY COUNTRY

   COUNTRY
   Afghanistan  SELECT COUNT(DISTINCT COUNTRY) TOTALCOUNTRY
   Afghanistan  FROM LREMPLOYEE
   Albania
   Argentina                    TOTALCOUNTRY
   Argentina                             81
                               ******* End of data ********
```

Group Functions Guidelines

- DISTINCT - Makes the function consider only non-duplicate values
- ALL - Makes function consider every value
  - DEFAULT value is ALL and does not need to be specified
- The DATA TYPES with the syntax expr argument may be CHAR, VARCHAR2, NUMBER, DATE
- All group functions ignore null values.

Group Functions Examples

- AVG ([distinct | ALL] expression)
  - Average value of n, ignoring null values
- COUNT ({ * [distinct | ALL] })
  - Number of rows where expr evaluates to something other than null - count all selected rows using * including duplicates and nulls unless you use distinct
- MAX([DISTINCT|ALL]expression)
  - Maximum value of expr, ignoring null values
- MIN([DISTINCT|ALL]expression)
  - Minimum value of expr, ignoring null values
- SUM([DISTINCT|ALL] expression)
  - Sum values of n, ignoring null values

Group Functions Examples

- STDDEV([DISTINCT|ALL]n)
  - Standard deviation of n, ignoring null values
- VARIANCE ([DISTINCT|ALL]n)
  - Variance of n, ignoring null values
- You can have distinct or all

```
select count(dept) from staff

COUNT ( DEPT )
            35
select count(distinct dept) from staff

       COUNT
           8
********  End of data  ********
```

Group Functions Examples

- PROBLEM: President wants to know data about salaries, such as the average salary, what the highest and lowest paid person's salary is and the company's total salary payout.

```
SELECT  AVG(SALARY),
        MAX(SALARY),
        MIN(SALARY),
        SUM(SALARY)
FROM    EMPLOYEE
```

AVG (SALARY)	MAX (SALARY)	MIN (SALARY)	SUM (SALARY)
47,441.071428571428571428571428	72,750.00	35,340.00	1,992,525.00

```
SELECT  CAST(AVG(SALARY) AS DECIMAL(10,2)) AVGSALARY,
        MAX(SALARY),
        MIN(SALARY),
        SUM(SALARY)
FROM    EMPLOYEE
```

AVGSALARY	MAX (SALARY)	MIN (SALARY)	SUM (SALARY)
47,441.07	72,750.00	35,340.00	1,992,525.00

Functions ignore null values

```
SELECT * FROM STAFF
WHERE COMM IS NULL
```

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
10	Sanders	20	Mgr	7	68,357.50	-
30	Marenghi	38	Mgr	5	67,506.75	-
50	Hanes	15	Mgr	10	70,659.80	-
100	Plotz	42	Mgr	7	68,352.80	-
140	Fraye	51	Mgr	6	71,150.00	-
160	Molinare	10	Mgr	7	72,959.20	-
210	Lu	10	Mgr	10	70,010.00	-
240	Daniels	10	Mgr	5	69,260.25	-
260	Jones	10	Mgr	12	71,234.00	-
270	Lea	66	Mgr	9	68,555.50	-
290	Quill	84	Mgr	10	69,818.00	-

```
SELECT MIN(COMM) FROM STAFF
```

MIN (COMM)
55.50

Group Functions – MIN / MAX

- You can use MIN and MAX for the following
- Numeric
- Character
- Date
- PROBLEM: Find the most junior (newest) and most senior (longest employed) employee.
- Is MIN the most years or the least years with the company ?

```
MOSTYEARS  LEASTYEARS
05/05/67    09/30/00

SELECT MIN(HIREDATE) MOSTYEARS,
       MAX(HIREDATE) LEASTYEARS
FROM EMPLOYEE
```

Group Functions – MIN / MAX

- PROBLEM: Find the first person alphabetically by last name and find the last employee by last name
- Applied to character columns

```
FIRSTINLINE      LASTINLINE
ADAMSON          YOSHIMURA
```

```
SELECT   MIN(LASTNAME) FIRSTINLINE,
         MAX(LASTNAME) LASTINLINE
FROM EMPLOYEE
```

- Note: You may want to use UPPER or LOWER if you are unsure about the case consistency of the data in the table. You can still use INITCAP to format your result set.

Using less space to show more columns

NAME	ADDRESS	COUNTRY	PHONE	EMPNUM
Yancey, William P.	P.O. Box 572, 5480 Eget St.	Guinea-bissau	70827680	111,111,111
Guthrie, Larissa B.	583 Rutrum Ave	Palau	94860680	122,222,222
Martinez, Yetta W.	569-4890 Dignissim. Street	Saint Kitts and Nevi	22761694	123,123,123
Landry, Thaddeus P.	Ap 804-4147 Nulla Road	Libyan Arab Jamahiri	82960776	133,333,333
Edwards, Brittany G.	Ap 373-3470 Quam, St.	Eritrea	82363520	143,444,433
Logan, Kaitlin J.	Ap 386-3761 Blandit Street	Bahrain	60253164	144,444,444
Morrison, Basia E.	3127 Nostra, Rd.	United Arab Emirates	36584232	155,555,555

```
SELECT substr(NAME,1,20) name,
       substr(ADDRESS,1,30) address,
       substr(COUNTRY,1,20) country,
       PHONE,
       EMPNUM
FROM lremployee
```

Group Functions – NULLS

- PROBLEM: What is the average commission paid out to employees ?
- Two interpretations:
  - Average commission payment for those employees who receive commission
  - Average commission payment for all employees, including those who receive an unknown commission.
- Two situations:
  - Is the unknown commission represented as 0 in our table ?
  - Is the unknown commission represented as NULL in our table ?

AVG with NULL values

```
SELECT COUNT(*)
FROM STAFF

 COUNT ( * )
          35

 SELECT COUNT(*)
 FROM STAFF
 WHERE COMM IS NOT NULL

   COUNT ( * )
            24
```

- SELECT SUM(COMM) FROM STAFF returns 12319.45
- 12319.45/35 = 351.98
- 12319.45/24 = 513.31

```
SELECT AVG(COMM)                          AVG ( COMM )
FROM STAFF          513.3104166666666666666666666
```

Group Functions – Groups of Data

- All group functions have treated the table so far as one large group
- Sometimes the information needs to be divided into smaller groups
- Example: Average by department

```
SELECT DEPT, AVG(SALARY)
FROM STAFF
```

- This will FAIL !!

```
Column DEPT or expression in SELECT list not valid.
```

Group Functions – Groups of Data

```
SELECT DEPT, AVG(SALARY)
FROM STAFF
```

- The use of the department id results in a row of output for each row in the employee table
- The AVG(SALARY) wants a single result for the entire table.
- There is no sensible way to display that.

Group Functions – Groups of Data

- the GROUP BY clause will resolve this

AVG with Group By

```
SELECT DEPT,
       AVG(SALARY)
FROM   STAFF
GROUP BY DEPT
 DEPT                          AVG ( SALARY )
   10    70,865.86250000000000000000000
   15    65,482.33250000000000000000000
   20    66,071.52500000000000000000000
   38    65,457.11000000000000000000000
   42    64,592.26250000000000000000000
   51    67,218.16000000000000000000000
   66    67,215.24000000000000000000000
   84    66,536.75000000000000000000000
```

- To clean up the output, there are a few options including ROUND

AVG with ROUND

```
SELECT DEPT,
       ROUND(AVG(SALARY),0)
FROM   STAFF
GROUP BY DEPT

DEPT                                    ROUND
  10   70,866.00000000000000000000000000
  15   65,482.00000000000000000000000000
  20   66,072.00000000000000000000000000
  38   65,457.00000000000000000000000000
  42   64,592.00000000000000000000000000
  51   67,218.00000000000000000000000000
  66   67,215.00000000000000000000000000
  84   66,537.00000000000000000000000000
```

- To clean up the output, there are a few options including CAST

AVG

```
SELECT DEPT,
       CAST(ROUND(AVG(SALARY),0) AS INTEGER) AVG_SALARY
FROM   STAFF
GROUP BY DEPT
```

DEPT	AVG_SALARY
10	70,866
15	65,482
20	66,072
38	65,457
42	64,592
51	67,218
66	67,215
84	66,537

- We included the group by column in the result set, usually a good idea but not always necessary

AVG with sorting

- How do you get this result?

```
DEPT       AVG_SALARY
  84           66,537
  66           67,215
  51           67,218
  42           64,592
  38           65,457
  20           66,072
  15           65,482
  10           70,866
```

```
SELECT DEPT,
       CAST(ROUND(AVG(SALARY),0) AS INTEGER) AVG_SALARY
FROM   STAFF
GROUP BY DEPT
ORDER BY DEPT DESC
```

AVG – order by average salary

- How do you get this result?

```
DEPT       AVG_SALARY
  42            64,592
  38            65,457
  15            65,482
  20            66,072
  84            66,537
  66            67,215
  51            67,218
  10            70,866
```

```
SELECT DEPT,
       CAST(ROUND(AVG(SALARY),0) AS INTEGER) AVG_SALARY
FROM   STAFF
GROUP BY DEPT
ORDER BY 2
```

Group Functions Examples

- For STAFF, how do you get this result:

DEPT	AVGSALARY	MAX (SALARY)	MIN (SALARY)	SUM (SALARY)
10	70,865.86	72,959.20	69,260.25	283,463.45
15	65,482.33	70,659.80	62,258.50	261,929.33
20	66,071.52	68,357.50	63,504.60	264,286.10
38	65,457.11	68,006.00	62,009.75	327,285.55
42	64,592.26	68,352.80	60,505.90	258,369.05
51	67,218.16	71,150.00	63,369.80	336,090.80
66	67,215.24	71,000.00	60,988.00	336,076.20
84	66,536.75	69,818.00	63,030.50	266,147.00

```
SELECT DEPT,
       CAST(AVG(SALARY) AS DECIMAL(10,2)) AVGSALARY,
       MAX(SALARY),
       MIN(SALARY),
       SUM(SALARY)
FROM   STAFF
GROUP BY DEPT
```

Group Functions Examples

- Fof STAFF, how do you get this result:

```
JOB          AVGSALRY    HIGHSALARY    LOWSALARY                    TOTALSALARY
Sales       67,869.36     71,000.00    65,454.50                     814,432.33
Mgr         69,805.80     72,959.20    67,506.75                     767,863.80
Clerk       72,004.50     74,129.00    69,581.78                     864,054.03
********  End of data  ********
```

- How do you just report on Clerks?

```
JOB          AVGSALRY    HIGHSALARY    LOWSALARY                    TOTALSALARY
Clerk       72,004.50     74,129.00    69,581.78                     864,054.03
```

```
SELECT   JOB,
         CAST(AVG(SALARY) AS DEC (10,2)) AVGSALRY,
         MAX(SALARY) HIGHSALARY,
         MIN(SALARY) LOWSALARY,
         SUM(SALARY) TOTALSALARY
FROM STAFF
GROUP BY JOB
HAVING JOB = 'Clerk'
```

Group Functions Examples

- Fof STAFF you are only interested in some of the departments

```
DEPT       AVGSALARY   MAX ( SALARY )  MIN ( SALARY )                SUM ( SALARY )
  42       64,592.26       68,352.80       60,505.90                     258,369.05
  51       67,218.16       71,150.00       63,369.80                     336,090.80
  66       67,215.24       71,000.00       60,988.00                     336,076.20
  84       66,536.75       69,818.00       63,030.50                     266,147.00
******  End of data  ********
```

```
SELECT DEPT,
       CAST(AVG(SALARY) AS DECIMAL(10,2)) AVGSALARY,
       MAX(SALARY),
       MIN(SALARY),
       SUM(SALARY)
FROM   STAFF
GROUP BY DEPT
HAVING DEPT > 40
```

- The Where clause filters out rows that don’t meet a search condition
- The Having clause filters out entire groups that don’t meet a search condition

Group Functions – Groups of Data

- Grouping by more than 1 column
- Leverage groups within groups
- PROBLEM: Display the total salary paid to each job title within each department

Group Functions – Groups of Data

DEPT	JOB	SUM (SALARY)
10	Mgr	283,463.45
15	Clerk	124,766.70
15	Mgr	70,659.80
15	Sales	66,502.83
20	Clerk	127,757.35
20	Mgr	68,357.50
20	Sales	68,171.25
38	Clerk	124,964.50
38	Mgr	67,506.75
38	Sales	134,814.30
42	Clerk	122,014.50
42	Mgr	68,352.80
42	Sales	68,001.75
51	Clerk	127,829.80
51	Mgr	71,150.00
51	Sales	137,111.00
66	Clerk	60,988.00
66	Mgr	68,555.50
66	Sales	206,532.70

Group Functions – Groups of Data

```
SELECT   DEPT,
         JOB,
         SUM(SALARY)
FROM     STAFF
GROUP BY DEPT, JOB
ORDER BY DEPT, JOB
```

- Limit this to where salaries are greater than 200000

```
DEPT   JOB                                    SUM ( SALARY )
  10   Mgr                                        283,463.45
  66   Sales                                      206,532.70
******  End of data  ********
```

```
    SELECT   DEPT,
             JOB,
             SUM(SALARY)
    FROM     STAFF
    GROUP BY DEPT, JOB
    HAVING SUM(SALARY) > 200000
    ORDER BY DEPT, JOB
```

DEPT	JOB	TOTALSALARY	TOTALINDEPT
10	Mgr	283,463.45	4
15	Clerk	124,766.70	2
15	Mgr	70,659.80	1
15	Sales	66,502.83	1
20	Clerk	127,757.35	2
20	Mgr	68,357.50	1
20	Sales	68,171.25	1
38	Clerk	124,964.50	2
38	Mgr	67,506.75	1
38	Sales	134,814.30	2
42	Clerk	122,014.50	2
42	Mgr	68,352.80	1
42	Sales	68,001.75	1
51	Clerk	127,829.80	2
51	Mgr	71,150.00	1
51	Sales	137,111.00	2
66	Clerk	60,988.00	1
66	Mgr	68,555.50	1
66	Sales	206,532.70	3

```
SELECT DEPT,
       JOB,
       CAST(SUM(SALARY) AS DECIMAL (10,2)) TOTALSALARY,
       COUNT(*) TOTALINDEPT
FROM   STAFF
GROUP BY DEPT, JOB
ORDER BY DEPT, JOB
```

Group Functions – Groups of Data

- Can you use a Where and a Group By in the same statement?
- What is the average salary for all job types excluding manager?

```
SELECT JOB,AVG(SALARY)
FROM STAFF
WHERE JOB <> 'Mgr'
GROUP BY JOB
```

```
 JOB                          AVG ( SALARY )
 Sales   67,869.360833333333333333333333
 Clerk   62,612.612500000000000000000000
```

Group Functions – Groups of Data

```
SELECT JOB,AVG(SALARY)
FROM STAFF
GROUP BY JOB
HAVING JOB <> 'Mgr'
```

```
JOB                          AVG ( SALARY )
Sales   67,869.360833333333333333333333
Clerk   62,612.612500000000000000000000
```

```
SELECT JOB,AVG(SALARY)
FROM STAFF
where AVG(salary) > 63000
GROUP BY JOB
HAVING JOB <> 'Mgr'
```

Use of function AVG not valid.

Group Functions – Groups of Data

```
SELECT JOB,AVG(SALARY)
FROM   STAFF
where  JOB <> 'Mgr'
GROUP  BY JOB
HAVING AVG(salary) > 63000
```

```
  JOB                          AVG ( SALARY )
  Sales  67,869.360833333333333333333333
```

PATIENT3

PATIENTNO	FIRSTNAME	LASTNAME	PINSURNUM	BIRTHDATE	CHARGE
123	Karen	Wong	555	12/25/67	6,000.00
456	Bill	Trimble	666	07/01/78	80,000.00
789	Tom	Seaver	888	02/22/84	39,999.00
246	John	Howard	444	04/15/98	12,000.00
333	Mandy	Suarez	444	11/25/77	3,200.00
555	Kumar	Rajendra	222	04/04/88	3,300.00
777	Wendy	Thomas	222	01/24/93	50,000.00
888	Casey	Stengal	111	02/18/66	44,000.00
999	Ted	Mendez	111	11/11/68	2,200.00
321	Mary	Worth	666	07/01/78	8,000.00
432	Jerry	Lowell	888	04/30/61	5,999.00
654	Wei	Tsung	444	05/15/77	900.00

```
alter table patient3 add column charge decimal(9,2)
```

INSURANCE3

INSNUM	INSURENAME	INSPHONE	MAXPAYOUT
555	Manulife	7,056,663,344	1,000,000.00
666	Royal Insurance	4,167,774,444	2,500,000.00
888	Cut Rate Insurers	9,058,883,333	100,000.00
444	SureHealth Ins	6,132,225,555	4,000,000.00

```
alter table insurance3 add column maxpayout decimal (9,2)
```

Problem

- Show each insurance company in insurance number order, how many patients have insurance with that company, what is the highest and lowest patient charges the companies covered and what would be the maximum amount they would ever pay out.

Produce this output

PINSURNUM	INSURENAME	TOTALPATIENTS	HIGHCHARGE	LOWCHARGE	MAXPAYOUT
444	SureHealth Ins	3	12,000.00	900.00	4,000,000.00
555	Manulife	1	6,000.00	6,000.00	1,000,000.00
666	Royal Insurance	2	80,000.00	8,000.00	2,500,000.00
888	Cut Rate Insurers	2	39,999.00	5,999.00	100,000.00

```
SELECT     PINSURNUM,
           INSURENAME,
           COUNT(PINSURNUM) TOTALPATIENTS,
           MAX(CHARGE) HIGHCHARGE,
           MIN(CHARGE) LOWCHARGE,
           MAXPAYOUT
FROM       PATIENT3 INNER JOIN INSURANCE3
ON         PINSURNUM = INSNUM
GROUP BY   PINSURNUM, MAXPAYOUT,  INSURENAME
ORDER BY   PINSURNUM
```

Problem

- Show each insurance company, how many patients have insurance with that company what is the highest and lowest patient charges the companies covered and what would be the maximum amount they would ever pay out.
- Remove the maximum payout any company can make and add the average payout they made

Produce this output

PINSURNUM	INSURENAME	TOTALPATIENTS	HIGHCHARGE	LOWCHARGE	AVGCHARGE
444	SureHealth Ins	3	12,000.00	900.00	5,366.66
555	Manulife	1	6,000.00	6,000.00	6,000.00
666	Royal Insurance	2	80,000.00	8,000.00	44,000.00
888	Cut Rate Insurers	2	39,999.00	5,999.00	22,999.00

```
SELECT     PINSURNUM,
           INSURENAME,
           COUNT(PINSURNUM) TOTALPATIENTS,
           MAX(CHARGE) HIGHCHARGE,
           MIN(CHARGE) LOWCHARGE,
           CAST (AVG(CHARGE)AS DECIMAL(9,2)) AVGCHARGE
FROM       PATIENT3, INSURANCE3
WHERE      PINSURNUM = INSNUM
GROUP BY   PINSURNUM, INSURENAME
ORDER BY   PINSURNUM
```

- Don’t include any company that has the same amounts for high charge and low charge
- Show an average charge of 5,366.666666 as 5,366

Exclude where the totalof = 1

PINSURNUM	INSURENAME	TOTALPATIENTS	HIGHCHARGE	LOWCHARGE	AVGCHARGE
444	SureHealth Ins	3	12,000.00	900.00	5,366
666	Royal Insurance	2	80,000.00	8,000.00	44,000
888	Cut Rate Insurers	2	39,999.00	5,999.00	22,999

```
SELECT    PINSURNUM,
          INSURENAME,
          COUNT(PINSURNUM) TOTALPATIENTS,
          MAX(CHARGE) HIGHCHARGE,
          MIN(CHARGE) LOWCHARGE,
          CAST(AVG(CHARGE) AS INTEGER) AVGCHARGE
FROM      PATIENT3 INNER JOIN INSURANCE3
ON        PINSURNUM = INSNUM
GROUP BY PINSURNUM, INSURENAME
HAVING    COUNT(PINSURNUM) <> 1
ORDER BY PINSURNUM
```

Doesn't cover the situation with only two patients and both have the same charge

A better solution than <> 1 answer

PINSURNUM	INSURENAME	TOTALPATIENTS	HIGHCHARGE	LOWCHARGE	AVGCHARGE
444	SureHealth Ins	3	12,000.00	900.00	5,366
666	TD Insurance	2	80,000.00	8,000.00	44,000
888	Cut Rate Insurers	2	39,999.00	5,999.00	22,999

```
SELECT    PINSURNUM,
          INSURENAME,
          COUNT(PINSURNUM) TOTALPATIENTS,
          MAX(CHARGE) HIGHCHARGE,
          MIN(CHARGE) LOWCHARGE,
          CAST(AVG(CHARGE) AS INTEGER) AVGCHARGE
FROM      PATIENT3 INNER JOIN INSURANCE3
ON        PINSURNUM = INSNUM
GROUP BY PINSURNUM, INSURENAME
HAVING MAX(CHARGE) <> MIN(CHARGE)
ORDER BY PINSURNUM
```

Does this work?

- Can you use TOTALPATIENTS in having clause?

```
SELECT    PINSURNUM,
          INSURENAME,
          COUNT(PINSURNUM) TOTALPATIENTS,
          MAX(CHARGE) HIGHCHARGE,
          MIN(CHARGE) LOWCHARGE,
          CAST(AVG(CHARGE) AS INTEGER) AVGCHARGE
FROM      PATIENT3 INNER JOIN INSURANCE3
ON        PINSURNUM = INSNUM
GROUP BY PINSURNUM, INSURENAME
HAVING    TOTALPATIENTS   <> 1
ORDER BY PINSURNUM

Column or global variable TOTALPATIENTS not found.
```

Will This Work?

```
SELECT     PINSURNUM,
           INSURENAME,
           COUNT(PINSURNUM)  TOTALPATIENTS,
           MAX(CHARGE)  HIGHCHARGE,
           MIN(CHARGE)  LOWCHARGE,
           CAST  (AVG(CHARGE)AS  INTEGER)  AVGCHARGE
FROM       PATIENT3 JOIN INSURANCE3
ON       PINSURNUM = INSNUM
GROUP BY   PINSURNUM,   INSURENAME
HAVING     COUNT(PINSURNUM) <> 1
ORDER BY   TOTALPATIENTS DESC
```

PINSURNUM	INSURENAME	TOTALPATIENTS	HIGHCHARGE	LOWCHARGE	AVGCHARGE
444	SureHealth Ins	3	12,000.00	900.00	5,366
666	Royal Insurance	2	80,000.00	8,000.00	44,000
888	Cut Rate Insurers	2	39,999.00	5,999.00	22,999

Lab1

- Lab 1 demonstrated to me in this weeks lab period
- In the lab period this week I am only marking lab 1
- If you want to stay ahead, show me lab 2 in an office hour the same week the lab is due
- Send an email when the office hour starts saying you want to show me lab 2 and your class
- I will bring you into the waiting room and see everyone in the order they appear in the waiting room.
- Office hours Monday – 8:55 AM, Mon – 10:45 AM, Thur 10:45 AM
- Have your statements in SQL Developer when starting your lab demonstration

Lab 2 & Assignment 1

- Lab 2 already available and Lecture 2 Powerpoint will be uploaded sometime tonight.
- Assignment 1 posted on Thursday.
- Only assignment task this week is to inform me who is in your group by Saturday at 9:00 pm
- After that time I place members in groups and the groups will not be changed