/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* \*\*

\*\* CHAPTER 8. Practical 1. \*\*

\*\* Aggregates and summarised queries \*\*

\*\* \*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Use database0

SELECT \* FROM dept

SELECT \* FROM salesperson

SELECT \* FROM sale

SELECT \* FROM contact

SELECT \* FROM company

-- a reminder of Join syntax from yesterday

From tableA as One Join tableB as Many

ON One.pk = Many.fk

--p1) Display the sum and average of the sales\_targets as well as a count

-- of the number of salespeople. (1 row 3 columns!)

SELECT sum(sp.sales\_target) Total, avg(sp.sales\_target) Average , count(sp.emp\_no) NoPeople

from salesperson sp

Total Average No of sales people

---------------- ------------------- ------------------

66.00 11.000000 6

--p2) Copy/Paste 1) and amend to display the same columns per dept number.

-- Ensure that you also include dept\_no at the start of the SELECT list (3 rows, 4 columns).

SELECT sp.dept\_no, sum(sp.sales\_target) Total, avg(sp.sales\_target) Average , count(sp.emp\_no) NoPeople

from salesperson sp

group by sp.dept\_no

dept\_no Total Average No of sales people

----------- --------------- ---------------------- ------------------

1 9.00 9.000000 1

2 21.00 10.500000 2

3 36.00 12.000000 3

--p3) Users of the output from 2) say "great report but I am not very good on dept numbers

-- can you include dept names AS WELL, and I would like the dept name to appear

-- just to the right of the dept\_no".

-- So copy/paste from 2), add dept\_name (now it is broken), then repair it.

SELECT sp.dept\_no, d.dept\_name,sum(sp.sales\_target) Total, avg(sp.sales\_target) Average , count(\*) NoPeople

from salesperson sp

join dept d on sp.dept\_no = d.dept\_no

group by sp.dept\_no, d.dept\_name

dept\_no dept\_name Total Average No of sales people

----------- -------------------- -------------- ------------------- ------------------

1 Animal Products 9.00 9.000000 1

2 Business Systems 21.00 10.500000 2

3 Credit Control 36.00 12.000000 3

-- Do NOTE!

-- All you have done is added 1 column to the display. How much did you have to change?

-- welcome to SQL!

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* \*\*

\*\* END OF CHAPTER 8. Practical 1 \*\*

\*\* \*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* \*\*

\*\* CHAPTER 8. Practical 2. \*\*

\*\* More Aggregates and summarised queries \*\*

\*\* \*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

--p1) Look at your output for part 3) of the previous lab and answer this question.

-- Which depts (names) have more than 1 person?

-- Copy/paste your code from part 3) above and amend it to show just the answer to that

-- question (2 rows, 1 column). You do need to use your DELETE key in this one!

dept\_name

--------------------

Business Systems

Credit Control

--p2) Read this carefully, for a new query.

-- Display a COUNT of the number of contacts per company name (4 rows).

-- You might like to run the code that you wrote yesterday in Chapter 5 (JOINs)

-- here it is, to help you picture what your output is going to look like.

SELECT CONT.name 'Contact', COMP.name 'Company'

FROM contact CONT INNER JOIN company COMP

ON CONT.company\_no = COMP.company\_no

Contact Company

-------------------- --------------------

Munching Mike Happy Heaters PLC

Naughty Nick Icicle Igloos PLC

Ollie Octopus Icicle Igloos PLC

Purposeful Peter Judo Jeans PLC

Quentin Quail Judo Jeans PLC

Robber Red Judo Jeans PLC

Marvellous Marvin Kipper Kickers Inc

Ricky Rambo Kipper Kickers Inc

Terrible Tim Kipper Kickers Inc

Uppy Umbrella Kipper Kickers Inc

SELECT comp.name, count(\*) 'TotalContacts'

FROM company comp

JOIN contact cont ON comp.company\_no = cont.company\_no

GROUP BY comp.name

name TotalContacts

-------------------- -------------

Happy Heaters PLC 1

Icicle Igloos PLC 2

Judo Jeans PLC 3

Kipper Kickers Inc 4

-- p3) You’re lucky that at the moment all companies have contacts,

-- but in the future some companies may exist (in 'company') to which we have not yet

-- allocated any 'contacts'.

-- Copy/Paste your code from p2) and ensure that those 'Companies without contacts'

-- also get included in the report by doing an OUTER JOIN (you decide whether it is LEFT or RIGHT)

-- To ensure that your output now changes, run this INSERT first.

INSERT INTO company

VALUES( 5000, 'ABC Ltd(no contacts)', '(01456)346782', 'Dorset' , 'ST8 3RG' )

--<<<< Paste the code in here

SELECT comp.name, count(CONT.company\_no) 'TotalContacts'

FROM company comp

LEFT JOIN contact cont ON comp.company\_no = cont.company\_no

GROUP BY comp.name

-- You might get this output!

name TotalContacts

-------------------- -------------

ABC Ltd(no contacts) 1 -- this might say 0 if so good, but read on

Happy Heaters PLC 1

Icicle Igloos Inc 2

Judo Jeans PLC 3

Kipper Kickers Inc 4

-- Did the ‘TotalContacts’ correctly display as '0', or did it say '1'?

-- If '1' then solve your problem.

name TotalContacts

-------------------- -------------

ABC Ltd(no contacts) 0

Happy Heaters PLC 1

Icicle Igloos Inc 2

Judo Jeans PLC 3

Kipper Kickers Inc 4

--p4) CREATE a VIEW called 'NoOfContacts' that contains the SELECT statement of 3).

CREATE VIEW NoOfContacts

AS

SELECT comp.name, count(CONT.company\_no) 'TotalContacts'

FROM company comp

LEFT JOIN contact cont ON comp.company\_no = cont.company\_no

GROUP BY comp.name

--p5) Be the 'user' who uses the VIEW from 4).

-- ORDER BY descending number of contacts.

SELECT \* FROM NoOfContacts

ORDER BY 'TotalContacts' desc

name TotalContacts

-------------------- -------------

Kipper Kickers Inc 4

Judo Jeans PLC 3

Icicle Igloos Inc 2

Happy Heaters PLC 1

ABC Ltd(no contacts) 0

-- Recognise how little the 'user' needs to know of SQL, compared to the person

-- who wrote the underlying SQL.

-- Clean up:

DELETE FROM company WHERE company\_no = 5000

-- If you are finished, here is some extra stuff to look at.

-- p6) TUTORIAL PORTION If you have time or review after the course

-- ================

-- For your consideration, NO CODE NEEDS TO BE WRITTEN HERE.

SELECT company\_no, emp\_no, SUM(order\_value) 'Total sales'

FROM sale

GROUP BY company\_no, emp\_no

ORDER BY company\_no, emp\_no

-- produces this standard summarised report.

company\_no emp\_no Total sales

----------- ----------- -----------

1000 10 5

1000 60 7

2000 10 3

2000 60 12

3000 50 27

3000 60 21

4000 60 2

-- SQL Server/Sybase/Oracle/Access all support variations on the following syntax.

-- The ROLLUP clause

-- =================

SELECT company\_no, emp\_no, SUM(order\_value) 'Total sales'

FROM sale

GROUP BY company\_no, emp\_no WITH ROLLUP -- rolling up to higher totals like a balance sheet

-- produces

company\_no emp\_no Total sales

----------- ----------- -----------

1000 10 5 <-- this row was in result set above

1000 60 7 <-- this row was in result set above

1000 NULL 12 <-- extra total row for company\_no 1000

2000 10 3 <-- this row was in result set above

2000 60 12 <-- this row was in result set above

2000 NULL 15 <-- extra total row for company\_no 2000

3000 50 27 <-- this row was in result set above

3000 60 21 <-- this row was in result set above

3000 NULL 48 <-- extra total row for company\_no 3000

4000 60 2 <-- this row was in result set above

4000 NULL 2 <-- extra total row for company\_no 4000

NULL NULL 77 <-- extra grand total row for all companies

-- and

-- the CUBE clause

-- ===============

SELECT company\_no, emp\_no, SUM(order\_value) 'Total sales'

FROM sale

GROUP BY company\_no, emp\_no WITH CUBE -- rolling up company\_no totals and showing emp\_no totals

-- produces

company\_no emp\_no Total sales

----------- ----------- -----------

1000 10 5

1000 60 7

1000 NULL 12 <-- total row for company\_no 1000

2000 10 3

2000 60 12

2000 NULL 15 <-- total row for company\_no 2000

3000 50 27

3000 60 21

3000 NULL 48 <-- total row for company\_no 3000

4000 60 2

4000 NULL 2 <-- total row for company\_no 4000

NULL NULL 77 <-- grand total row for all companies

NULL 10 8 <-- extra total row for emp\_no 10

NULL 50 27 <-- extra total row for emp\_no 50

NULL 60 42 <-- extra total row for emp\_no 60

-- So with a little coalescing,

SELECT COALESCE(STR(company\_no,4),'All Companies ') AS company\_no,

COALESCE(STR(emp\_no,3),'All employees ') AS emp\_no,

SUM(order\_value) 'Total sales'

FROM sale

GROUP BY company\_no, emp\_no WITH CUBE -- rolling up company\_no totals and showing emp\_no totals

-- we can get

company\_no emp\_no Total sales

--------------- --------------- -----------

1000 10 5

1000 60 7

1000 All employees 12

2000 10 3

2000 60 12

2000 All employees 15

3000 50 27

3000 60 21

3000 All employees 48

4000 60 2

4000 All employees 2

All Companies All employees 77 -- Grand total line from rollup

All Companies 10 8 --|

All Companies 50 27 --|these 3 lines because of CUBE

All Companies 60 42 --|

-- GROUPING SECTION

-- ================

-- The grouping function is useful as it returns '1' if it is 'super-aggregate' row

-- and '0' if it is a normal total row.

-- This is useful if there are 'NULL' values in the original data because now you are not

-- sure whether the NULL is from the data or a 'NULL' is generated in a super-aggregate row

-- because you are using rollup or cube.

SELECT COALESCE(STR(company\_no,4),'All Companies ') AS company\_no,

COALESCE(STR(emp\_no,3),'All employees ') AS emp\_no,

SUM(order\_value) 'Total sales',

GROUPING(company\_no) AS Grp\_CompNo,

GROUPING(emp\_no) AS Grp\_EmpNo

FROM sale

GROUP BY company\_no, emp\_no WITH CUBE -- rolling up company\_no totals and showing emp\_no totals

company\_no emp\_no Total sales Grp\_CompNo Grp\_EmpNo

--------------- --------------- ----------- ---------- ---------

1000 10 5 0 0

1000 60 7 0 0 -- normal summarised row both 0

1000 All employees 12 0 1 -- 1 = 'total for all employees'

2000 10 3 0 0

2000 60 12 0 0

2000 All employees 15 0 1 -- 1 = 'total for all employees'

3000 50 27 0 0

3000 60 9 0 0

3000 All employees 36 0 1 -- 1 = 'total for all employees'

4000 60 2 0 0

4000 All employees 2 0 1 -- 1 = 'total for all employees'

All Companies All employees 65 1 1 -- both '1' as it is grand total

All Companies 10 8 1 0

All Companies 50 27 1 0 -- 1 = 'total for all companies'

All Companies 60 30 1 0

-- Grouping in conjunction with a Case statement enables the following

SELECT

-- to determine what appears in 1st column

CASE

WHEN GROUPING(company\_no) = 1 AND GROUPING(emp\_no) = 1

THEN 'Grand Total' --Total of both

ELSE

CASE

WHEN GROUPING(company\_no) = 1 --But Grouping(emp\_no) = 0

THEN 'All Companies ' --Total for one employee

ELSE STR(company\_no,4) --Normal Total line

END

END AS Company, -- end of column 1

-- to determine what appears in 2nd column

CASE

WHEN GROUPING(company\_no) = 1 AND GROUPING(emp\_no) = 1

THEN '' --Total of both

ELSE

CASE

WHEN GROUPING(emp\_no) = 1

THEN 'All Employees ' --Total for one company

ELSE STR(emp\_no,2) --Normal Total line

END

END AS Employee, -- end of column 2

SUM(order\_value) AS 'Total sales'

FROM sale

GROUP BY company\_no, emp\_no WITH CUBE -- rolling up company\_no totals and showing emp\_no totals

-- produces this elegant output

Company Employee Total sales

--------------- --------------- -----------

1000 10 5

1000 60 7

1000 All Employees 12

2000 10 3

2000 60 12

2000 All Employees 15

3000 50 27

3000 60 21

3000 All Employees 48

4000 60 2

4000 All Employees 2

Grand Total 77

All Companies 10 8

All Companies 50 27

All Companies 60 42

-- There is little variation in this syntax between the major players.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* \*\*

\*\* END OF CHAPTER 8. Practical 2 \*\*

\*\* \*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/