CreateUser.sql

ACCEPT username VARCHAR2 PROMPT "User Name: "

ACCEPT password VARCHAR2 PROMPT "Password: "

CREATE USER &username IDENTIFIED BY &password

DEFAULT TABLESPACE users

/

GRANT CONNECT TO &&username;

GRANT RESOURCE TO &&username;

GRANT CREATE VIEW TO &&username;

GRANT UNLIMITED TABLESPACE TO &&user;

ALTER PROFILE default LIMIT PASSWORD\_LIFE\_TIME UNLIMITED;

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

\* HRGrants.sql

\* CIS-2720

\*

\* Grants access to all the HR tables to all users of the database (PUBLIC)

\*

\* Must be run by HR, SYSTEM, or and account with the DBA role

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

GRANT SELECT ON hr.regions TO public;

GRANT SELECT ON hr.locations TO public;

GRANT SELECT ON hr.departments TO public;

GRANT SELECT ON hr.jobs TO public;

GRANT SELECT ON hr.employees TO public;

GRANT SELECT ON hr.job\_history TO public;

GRANT SELECT ON hr.countries TO public;

-- Try running with "F5" (the green arrow icon)

-- This is the "text" version (or "MS Word" like version)

-- Highlight one query and "Ctrl-Enter" (the green arrow on top of

-- document)

-- This is the "grid" version (or "Excel" like version)

-- Try making the font larger (tools -> preferences -> code editor -> fonts)

**2 Getting Information from a Table(SELECT, WHERE)**

SELECT employee\_id AS employee\_number,

phone\_number AS extension,

last\_name,

'EXCELLENT WORKER' AS evaluation,

10 AS rating

FROM l\_employees;

SELECT DISTINCT manager\_id

FROM l\_employees

WHERE employee\_id IN (201, 208, 210)

OR first\_name = 'PAULA';

SELECT dept\_code,

last\_name

FROM l\_employees

WHERE NOT (employee\_id = 209)

ORDER BY dept\_code,

last\_name DESC;

SELECT employee\_id,

first\_name,

last\_name,

manager\_id

FROM l\_employees

WHERE NOT (manager\_id = 203);

SELECT employee\_id,

first\_name,

last\_name,

hire\_date

FROM l\_employees

WHERE hire\_date BETWEEN '16-AUG-1999' AND '01-JUL-2008';

SELECT employee\_id,

first\_name,

last\_name

FROM l\_employees

WHERE last\_name LIKE '%N%';

**3 Compound Conditions in the Where Clause**

-- A \* B + C \* D

SELECT \*

FROM zip\_codes

WHERE state = 'IL'

AND city = 'ELGIN'

OR state = 'AZ'

AND city = 'WINSLOW';

-- (A \* B) + (C \* D)

-- Equivalent to the previous query - "implied parenthesis"

SELECT \*

FROM zip\_codes

WHERE (state = 'IL'

AND city = 'ELGIN')

OR (state = 'AZ'

AND city = 'WINSLOW');

SELECT \*

FROM l\_employees

WHERE NOT (dept\_code IN ('ACT', 'MKT'))

AND NOT (last\_name BETWEEN 'J' AND 'M')

AND NOT (last\_name LIKE '%S')

AND NOT (manager\_id IS NULL);

SELECT \*

FROM l\_employees

WHERE employee\_id = 203

OR employee\_id = 204

OR employee\_id = 205

AND dept\_code = 'SAL';

--equel

SELECT \*

FROM l\_employees

WHERE employee\_id = 203

OR employee\_id = 204

OR (employee\_id = 205 AND dept\_code = 'SAL');

SELECT \*

FROM l\_suppliers

WHERE supplier\_name = 'ALICE ' || &|| ' RAY''S RESTAURANT';

**4 Saving Your Results(CREAT, INSERT, DROP, UPDATE)**

DROP TABLE academy\_awards;

CREATE TABLE academy\_awards (

ceremony\_nbr NUMBER(3,0),

ceremony VARCHAR2(30),

ceremony\_date DATE,

winner VARCHAR2(60),

broadcast\_length VARCHAR2(20),

viewers NUMBER(9,0),

rating NUMBER(4,1),

hosts VARCHAR2(100)

)

-- Insert one row in the table

-- Note that values for all columns must be specified, in the correct -- order

-- If a value is to be left out, a NULL must be used

INSERT INTO academy\_awards VALUES(1, '1st Academy Awards', '16-MAY-1929', 'Wings', 'No broadcast', NULL, NULL, 'Douglas Fairbanks, William C. deMille');

-- COMMIT "saves" your work

COMMIT;

-- Insert one row, Empty columns are not listed, Columns can be in any -- order

INSERT INTO academy\_awards

(

hosts,

winner,

ceremony\_date,

ceremony,

ceremony\_nbr

)

VALUES(

'William C. deMille',

'The Broadway Melody',

'03-APR-1930',

'2nd Academy Awards',

2

);

COMMIT;

-- Update original table

-- Note the / instead of a ;

UPDATE academy\_awards

SET broadcast\_length = 'Unknown'

WHERE broadcast\_length IS NULL

/

CREATE OR REPLACE VIEW aa\_1960s\_view AS

SELECT ceremony\_nbr, ceremony, ceremony\_date, winner, hosts

FROM academy\_awards

WHERE ceremony\_date BETWEEN '01-JAN-1960' AND '31-DEC-1969';

UPDATE sec0411\_foods

SET price = price + 0.10,

price\_increase = price\_increase + 0.10

WHERE supplier\_id IN ('JBR', 'FRV');

DELETE FROM sec0412\_foods; --delete the whole data

ROLLBACK; --get the data back

**5 The Data Dictionary and Other Oracle Topics**

-- 5-5 Example 2 Step 1

CREATE OR REPLACE VIEW sec0505b\_shipping\_dept\_view AS

SELECT employee\_id,

first\_name,

last\_name,

dept\_code,

credit\_limit

FROM sec0505b\_employees

WHERE dept\_code = 'SHP';

-- 5-5 Example 2 Step 2

UPDATE sec0505b\_shipping\_dept\_view

SET dept\_code = 'MKT'

WHERE employee\_id = 207;

-- 5-5 Example 2 Step 3

INSERT INTO sec0505b\_shipping\_dept\_view

VALUES

(211, 'SUSAN', 'MANNING', 'EXE', 50.00);

-- 5-5 Example 2 Step 4

SELECT \* FROM sec0505b\_shipping\_dept\_view;

-- 5-5 Example 2 Step 5

SELECT \* FROM sec0505b\_employees;

DESC user\_tables

SELECT \* FROM user\_tables

ORDER BY table\_name;

DESC user\_views

SELECT view\_name FROM user\_views;

- 5-14 Describe command IN Oracle

-- This works IN the SQL Commands page

-- But it does NOT work IN an SQL Script

DESCRIBE l\_employees;

DESC l\_employees;

-- 5-14 Column names of tables and views

SELECT table\_name,

column\_name,

column\_id

FROM user\_tab\_columns

WHERE table\_name = 'L\_EMPLOYEES'

ORDER BY column\_id;

-- 5-15 step 1

DESC user\_constraints

SELECT table\_name,

constraint\_type,

constraint\_name

FROM user\_constraints

WHERE table\_name = 'L\_FOODS';

-- 5-15 step 2 ---try to find primry key

DESC user\_cons\_columns

SELECT \*

FROM user\_cons\_columns

WHERE table\_name = 'L\_FOODS';

**6 Creating Your Own Tables(Prime Key, Alter, add or modify a column)**

SELECT z.zip\_code,

z.ROWID,

dbms\_rowid.rowid\_object(z.ROWID) AS object\_id,

dbms\_rowid.rowid\_to\_absolute\_fno(z.rowid, o.owner, o.object\_name) as file\_id,

dbms\_rowid.rowid\_block\_number(z.ROWID) AS block\_number,

dbms\_rowid.ROWID\_ROW\_NUMBER(Z.rowid) as ROW\_NUMBER,

o.object\_name as table\_name,

f.file\_name,

f.tablespace\_name

FROM kevin.zip\_codes z

JOIN dba\_objects o

ON (o.object\_id = dbms\_rowid.rowid\_object(z.ROWID))

JOIN dba\_data\_files f

ON (f.file\_id = dbms\_rowid.rowid\_to\_absolute\_fno(z.ROWID, o.owner, o.object\_name))

ORDER BY ROWID;

DELETE FROM rowid\_test WHERE ROWID = 'AAAIrKAAEAAAA4vAAA';

SELECT row\_id,

variable\_length\_string || 'X',

LENGTH(variable\_length\_string) AS length\_of\_vl\_string,

fixed\_length\_string || 'X',

LENGTH(fixed\_length\_string) AS length\_of\_fl\_string,

long\_string || 'X',

LENGTH(long\_string) AS length\_of\_long\_string

FROM sec0603\_text\_datatypes;

ALTER TABLE sec0609\_foods

ADD CONSTRAINT pk\_sec0609\_foods

PRIMARY KEY (supplier\_id, product\_code);

-- 6-10 Method 1 - Using the name of the CONSTRAINT to DROP it

ALTER TABLE sec0610\_foods

DROP CONSTRAINT pk\_sec0610\_foods;

-- 6-10 Method 2 - NOT using the name of the CONSTRAINT to DROP it

ALTER TABLE sec0610b\_foods

DROP PRIMARY KEY;

-- 6-11 ADD a column to the TABLE

ALTER TABLE sec0611\_foods

ADD date\_introduced DATE;

-- 6-12

ALTER TABLE sec0612\_foods

MODIFY description VARCHAR2(25);

ALTER TABLE sec0612\_foods

MODIFY price NUMBER(7,2);

-- 6-13 DELETE a column FROM the TABLE

ALTER TABLE sec0613\_foods

DROP COLUMN price\_increase;

-- 6-16

CREATE TABLE sec0616\_no\_duplicate\_rows AS

SELECT DISTINCT \*

FROM sec0615b;

**7 Formats, Sequences, and Indexes**

**-- Date Examples.sql**

ALTER SESSION SET NLS\_DATE\_FORMAT = 'DD-MON-YY';

-- Validate that the signup date is before last access date

-- Should return zero rows

SELECT \* FROM customer\_web\_access

WHERE signup\_date >= last\_access\_datetime;

SELECT \*

FROM customer\_web\_access

WHERE signup\_date >= TO\_DATE('6/1/2011', 'mm/dd/yyyy');

-- Columns of DATE type contain both date and time

SELECT cust\_nbr,

last\_name,

first\_name,

signup\_date,

TO\_CHAR(signup\_date, 'mm/dd/yyyy hh24:mi:ss') AS signup\_date\_string,

last\_access\_datetime,

TO\_CHAR(last\_access\_datetime, 'mm/dd/yyyy hh24:mi:ss') AS last\_access\_datetime\_string

FROM customer\_web\_access

ORDER BY cust\_nbr;

-- This does what we wanted

SELECT \*

FROM customer\_web\_access

WHERE TRUNC(last\_access\_datetime) = '13-AUG-2011';

-- Correct method #1

SELECT \*

FROM customer\_web\_access

WHERE last\_access\_datetime BETWEEN '01-JAN-2010'

AND TO\_DATE('31-DEC-2010 23:59:59', 'DD-MON-YYYY hh24:mi:ss');

-- Correct method #2

SELECT \*

FROM customer\_web\_access

WHERE last\_access\_datetime >= '01-JAN-2010'

AND last\_access\_datetime < '01-JAN-2011';

--correct method #3

SELECT \*

FROM customer\_web\_access

WHERE TRUNC (last\_access\_datetime) BETWEEN '01-JAN-2010' AND '01-JAN-2011';

ALTER TABLE customer\_web\_access

ADD CONSTRAINT ck\_signup\_date CHECK

(signup\_date = TRUNC(signup\_date));

-- Part 5: Miscellaneous date techniques

-- Current date/time on database server

SELECT SYSDATE FROM DUAL;

SELECT SYSDATE FROM l\_employees;

SELECT SYSDATE,

TO\_CHAR(SYSDATE, 'DD-MON-YYYY hh24:mi:ss') AS sysdate\_datetime

FROM DUAL;

-- Yesterday, same time (24 hours ago)

SELECT TO\_CHAR(SYSDATE, 'DD-MON-YYYY hh24:mi:ss') AS sysdate\_datetime,

TO\_CHAR(SYSDATE - 1, 'DD-MON-YYYY hh24:mi:ss') AS yesterday\_datetime

FROM DUAL;

-- Yesterday, (25 hours ago)

SELECT TO\_CHAR(SYSDATE, 'DD-MON-YYYY hh24:mi:ss') AS sysdate\_datetime,

TO\_CHAR(SYSDATE - 1/24, 'DD-MON-YYYY hh24:mi:ss') AS yesterday\_datetime

FROM DUAL;

-- Yesterday, (23 hours and 59 minutes ago)

SELECT TO\_CHAR(SYSDATE, 'DD-MON-YYYY hh24:mi:ss') AS sysdate\_datetime,

TO\_CHAR(SYSDATE - 1/24/60, 'DD-MON-YYYY hh24:mi:ss') AS yesterday\_datetime

FROM DUAL;

-- Yesterday, no time

SELECT TO\_CHAR(SYSDATE, 'DD-MON-YYYY hh24:mi:ss') AS sysdate\_datetime,

TO\_CHAR(TRUNC(SYSDATE) - 1, 'DD-MON-YYYY hh24:mi:ss') AS yesterday\_date

FROM DUAL;

-- Two and a half hours ago

SELECT TO\_CHAR(SYSDATE, 'DD-MON-YYYY hh24:mi:ss') AS sysdate\_datetime,

TO\_CHAR(SYSDATE - 2.5/24, 'DD-MON-YYYY hh24:mi:ss') AS time\_150\_min\_ago

FROM DUAL;

-- Five minutes ago

SELECT TO\_CHAR(SYSDATE, 'DD-MON-YYYY hh24:mi:ss') AS sysdate\_datetime,

TO\_CHAR(SYSDATE - 5/24/60, 'DD-MON-YYYY hh24:mi:ss') AS time\_5\_min\_ago

FROM DUAL;

-- One year from now

SELECT TO\_CHAR(SYSDATE, 'DD-MON-YYYY hh24:mi:ss') AS sysdate\_datetime,

TO\_CHAR(ADD\_MONTHS(SYSDATE, 12), 'DD-MON-YYYY hh24:mi:ss') AS one\_year

FROM DUAL;

-- Raw data

SELECT TO\_CHAR(SYSDATE, 'DD-MON-YYYY hh24:mi:ss') AS sysdate\_datetime,

DUMP(SYSDATE) AS raw\_sysdate

FROM DUAL;

-- Last day of current month

SELECT TO\_CHAR(SYSDATE, 'DD-MON-YYYY hh24:mi:ss') AS sysdate\_datetime,

TO\_CHAR(LAST\_DAY(SYSDATE), 'DD-MON-YYYY hh24:mi:ss') AS last\_day

FROM DUAL;

-- Next Friday

SELECT TO\_CHAR(SYSDATE, 'DD-MON-YYYY hh24:mi:ss') AS sysdate\_datetime,

TO\_CHAR(NEXT\_DAY(SYSDATE, 'FRI'), 'DD-MON-YYYY hh24:mi:ss') AS next\_fri

FROM DUAL;

-- Round (instead of TRUNC)

SELECT TO\_CHAR(SYSDATE, 'DD-MON-YYYY hh24:mi:ss') AS sysdate\_datetime,

TO\_CHAR(ROUND(SYSDATE), 'DD-MON-YYYY hh24:mi:ss') AS sysdate\_round,

TO\_CHAR(ROUND(SYSDATE + 0.5), 'DD-MON-YYYY hh24:mi:ss') AS sysdate\_round2,

TO\_CHAR(ROUND(SYSDATE - 0.5), 'DD-MON-YYYY hh24:mi:ss') AS sysdate\_round3

FROM DUAL;

-- Show tables

SELECT s.owner, table\_name, segment\_type, bytes/1024/1024 megs

FROM dba\_segments s

INNER JOIN dba\_tables t

ON s.owner = t.owner

AND s.segment\_name = t.table\_name

WHERE s.owner = 'BIG'

ORDER BY s.owner, table\_name;

-- Show indexes

SELECT s.owner, table\_name, index\_name, segment\_type, bytes/1024/1024 megs

FROM dba\_segments s

INNER JOIN dba\_indexes i

ON s.owner = i.owner

AND s.segment\_name = i.index\_name

WHERE s.owner = 'BIG'

ORDER BY s.owner, table\_name, index\_name;

SELECT employee\_id,

first\_name,

hire\_date,

TO\_CHAR(hire\_date, 'MM-DD-YYYY HH:MI AM') AS formatted\_date

FROM l\_employees

ORDER BY employee\_id;

-- 7-6

-- DELETE command for a SEQUENCE

-- This IS a preventative DELETE for the SEQUENCE we

-- are about to CREATE.

DROP SEQUENCE sec0706\_seq\_lunch\_id;

-- 7-6 CREATE a new SEQUENCE

CREATE SEQUENCE sec0706\_seq\_lunch\_id

START WITH 23

INCREMENT BY 1;

-- 7-7 Get the current value of a SEQUENCE

SELECT sec0707\_seq\_lunch\_id.currval

FROM DUAL;

-- 7-7b: Get the next value of a SEQUENCE

SELECT sec0707\_seq\_lunch\_id.nextval

FROM DUAL;

-- 7-9: Book index example: "Run Button"

CREATE INDEX ix\_sec0709\_employees\_name

ON sec0709\_employees

(last\_name, first\_name);

-- 7-16

SELECT \*

FROM dictionary

WHERE table\_name LIKE '%SEQ%';

**8 Data Integrity(Check, Unique, Prime Key, Foreign Key )**

-- 8-2

ALTER TABLE sec0802\_foods

ADD CONSTRAINT sec0802\_foods\_max\_price

CHECK (price < 10.00);

-- 8-3 method 1

ALTER TABLE sec0803\_employees

ADD CONSTRAINT unique\_sec0803\_emp\_phone\_num

UNIQUE (phone\_number);

-- 8-3 method 2

CREATE UNIQUE INDEX uix\_sec0803b\_emp\_phone

ON sec0803b\_employees (phone\_number);

-- 8-5

ALTER TABLE sec0805\_employees

ADD CONSTRAINT pk\_sec0805\_employees

PRIMARY KEY (employee\_id);

-- 8-8

ALTER TABLE sec0808\_clients

ADD CONSTRAINT fk\_sec0808\_clients\_state\_code

FOREIGN KEY (state\_code)

REFERENCES sec0808\_states (state\_code);

-- 8-17 task 1

ALTER TABLE sec0817\_clients

ADD CONSTRAINT fk\_sec0817\_clients\_state\_code

FOREIGN KEY (state\_code)

REFERENCES sec0817\_states (state\_code)

ON DELETE SET NULL;

-- 8-17 task 2

DELETE FROM sec0817\_states

WHERE state\_code = 'CA';

-- 8-21

ALTER TABLE sec0821\_lunch\_items

ADD CONSTRAINT fk\_sec0821\_lunch\_items\_foods

FOREIGN KEY (supplier\_id, product\_code)

REFERENCES sec0821\_foods (supplier\_id, product\_code);

-- 8-23 method 1

-- Un-named "In-line" constraints

CREATE TABLE sec0823a\_employees (

employee\_id NUMBER(3) PRIMARY KEY,

first\_name VARCHAR2(10) NOT NULL,

last\_name VARCHAR2(20) NOT NULL,

dept\_code VARCHAR2(3) REFERENCES sec0823a\_departments(dept\_code),

hire\_date DATE,

credit\_limit NUMBER(4,2) CHECK (credit\_limit < 50),

phone\_number VARCHAR2(4) UNIQUE,

manager\_id NUMBER(3) REFERENCES sec0823a\_employees(employee\_id),

UNIQUE (first\_name, last\_name)

);

ALTER TABLE sec0823a\_employees DROP CONSTRAINT SYS\_C0038505;

-- 8-23 method 2

-- Named "In-line" constraints

CREATE TABLE sec0823b\_employees (

employee\_id NUMBER(3) CONSTRAINT pk\_employee\_id PRIMARY KEY,

first\_name VARCHAR2(10) CONSTRAINT nn\_first\_name

CHECK (first\_name IS NOT NULL),

last\_name VARCHAR2(20) CONSTRAINT nn\_last\_name

CHECK (last\_name IS NOT NULL),

dept\_code VARCHAR2(3) CONSTRAINT fk\_dept\_code

REFERENCES sec0823b\_departments(dept\_code),

hire\_date DATE,

credit\_limit NUMBER(4,2) CONSTRAINT max\_limit

CHECK (credit\_limit < 50),

phone\_number VARCHAR2(4) CONSTRAINT unique\_phone\_num

UNIQUE,

manager\_id NUMBER(3) CONSTRAINT fk\_manager\_id

REFERENCES sec0823b\_employees(employee\_id),

CONSTRAINT unique\_name

UNIQUE (first\_name, last\_name)

);

**9 Row Functions**

-- 9-2

CREATE TABLE sec0902\_foods AS

SELECT l\_foods.\*,

price + price\_increase AS new\_price

FROM l\_foods;

-- 9-7

SELECT 3 \* 4 FROM DUAL;

-- 9-8

SELECT n,

mod(n, 3)

FROM sec0908\_test\_numbers

ORDER BY n;

-- 9-10

SELECT employee\_id,

first\_name || ' ' || last\_name AS full\_name

FROM l\_employees;

-- 9-12

SELECT employee\_id,

first\_name,

last\_name,

'(415) 643-' || phone\_number AS phone\_number2

FROM l\_employees;

-- 9-13 Added by KF - "years between"

-- Use MONTHS\_BETWEEN() to determine whole number of years

SELECT MONTHS\_BETWEEN(TO\_DATE('01-MAR-1958'),TO\_DATE('01-MAR-1999')) / 12 AS d1,

(TO\_DATE('01-MAR-1958') - TO\_DATE('01-MAR-1999')) / 365.25 AS d2

FROM l\_employees;

SELECT TRUNC(MONTHS\_BETWEEN(TO\_DATE('01-MAR-1958'),TO\_DATE('01-MAR-1999')) / 12) AS d1,

TRUNC((TO\_DATE('01-MAR-1958') - TO\_DATE('01-MAR-1999')) / 365.25) AS d2

FROM l\_employees;

-- 9-14

SELECT first\_name, last\_name, hire\_date,

sysdate - hire\_date AS days\_with\_company,

floor((TO\_DATE('01-JAN-2011') - hire\_date)/30) AS months\_with\_the\_company

FROM l\_employees;

-- 9-15 showing the times

SELECT lunch\_id, lunch\_date,

employee\_id,

TO\_CHAR(date\_entered, 'DD-MON-YYYY HH:MI AM') AS date\_entered

FROM l\_lunches;

-- 9-15 deleting the times

SELECT lunch\_id, lunch\_date,

employee\_id,

TO\_CHAR(trunc(date\_entered), 'DD-MON-YYYY HH:MI AM') AS date\_entered

FROM l\_lunches;

**10 Using Row Functions**

-- 10-2

SELECT user,

TO\_CHAR(SYSDATE, 'DAY MONTH DD, YYYY HH:MI AM') AS date\_time

FROM DUAL;

-- 10-3 task 1, Oracle style - Replacement value has the same datatype -- AS the column

SELECT pkey,

NVL(num\_col,0) AS num\_col2,

NVL(text\_col, 'ZILCH') AS text\_col2,

NVL(date\_col, '01-JAN-1900') AS date\_col2

FROM sec1003;

-- 10-3 task 2, Access style - Replacement value IS text

SELECT pkey,

NVL(TO\_CHAR(num\_col), 'NO NUMBER') AS num\_col2,

NVL(text\_col, 'NO TEXT') AS text\_col2,

NVL(TO\_CHAR(date\_col), 'NO DATE') AS date\_col2

FROM sec1003;

-- Added by KF - NVL2()

SELECT last\_name,

first\_name,

NVL(phone\_number, 'NO PHONE') AS phone\_flag1,

NVL2(phone\_number, 'HAS A PHONE', 'NO PHONE') AS phone\_flag2

FROM l\_employees;

-- 10-4 Shows the problem IN Access

CREATE TABLE xyz AS

SELECT pkey,

text\_1,

text\_2,

text\_1 + text\_2 AS text\_add,

text\_1 - text\_2 AS text\_subtract,

text\_1 \* text\_2 AS text\_multiply,

text\_1 / text\_2 AS text\_divide

FROM sec1004;

-- 10-8

-- 10-8 step 1

CREATE OR REPLACE VIEW sec1008\_view AS

SELECT n,

3 \* n AS multiple\_of\_3

FROM numbers\_0\_to\_99;

-- 10-8 step 2

SELECT multiple\_of\_3

FROM sec1008\_view

WHERE multiple\_of\_3 BETWEEN 50 AND 250

ORDER BY multiple\_of\_3;

-- 10-9

SELECT n AS prime\_number

FROM numbers\_0\_to\_99

WHERE n > 10

AND NOT (MOD(n, 2) = 0)

AND NOT (MOD(n, 3) = 0)

AND NOT (MOD(n, 5) = 0)

AND NOT (MOD(n, 7) = 0)

ORDER BY n;

-- 10-10

-- 10-10 Step 1 - CREATE a TABLE of constants

CREATE TABLE sec1010\_constants (

begin\_date DATE

);

INSERT INTO sec1010\_constants VALUES ('24-FEB-2010');

-- 10-10 Step 2 - CREATE a VIEW containing seven dates

CREATE OR REPLACE VIEW sec1010\_view AS

SELECT begin\_date + digit AS days

FROM numbers\_0\_to\_9,

sec1010\_constants

WHERE digit < 7;

-- 10-10 Step 3 - List the days formatted IN three ways

SELECT days,

TO\_CHAR(days, 'DY') AS abbreviated\_day,

TO\_CHAR(days, 'DAY') AS full\_day

FROM sec1010\_view

ORDER BY days;

-- 10-12

SELECT SYSDATE - TO\_DATE('21-MAR-1978') AS days\_old

FROM DUAL;

-- 10-13

SELECT TO\_DATE('21-MAR-1978') + 10000 AS celebration\_day

FROM DUAL;

**11 Summarizing Data**

-- 11-3

SELECT MIN(credit\_limit),

MAX(credit\_limit),

MAX(first\_name),

MAX(last\_name),

MAX(hire\_date)

FROM l\_employees;

-- 11-5 Method 2

SELECT \* FROM l\_employees;

SELECT employee\_id,

first\_name,

last\_name,

credit\_limit

FROM l\_employees

WHERE credit\_limit = (SELECT MIN(credit\_limit)

FROM l\_employees)

ORDER BY employee\_id;

-- 11-6

SELECT COUNT(\*),

COUNT(last\_name),

COUNT(hire\_date),

COUNT(manager\_id)

FROM l\_employees;

SELECT COUNT(\*),

COUNT(last\_name),

COUNT(hire\_date),

COUNT(DISTINCT manager\_id)

FROM l\_employees;

-- 11-7

-- Modified BY KF - added 2 to aliases so they were not keywords words

-- (these are "keywords", not reserved words, so technically it is okay

-- to use them as aliases - but it isn't a good idea).

-- Note that SUM AND AVG functions covered IN section 11-10

SELECT COUNT(col\_2) AS count\_col,

COUNT(\*) AS count\_rows,

MAX(col\_2) AS max2,

MIN(col\_2) AS min2,

SUM(col\_2) AS sum2,

AVG(col\_2) AS avg2

FROM sec1107;

-- 11-9

SELECT \* FROM l\_employees;

SELECT manager\_id || '\*' || credit\_limit FROM l\_employees;

SELECT DISTINCT manager\_id || '\*' || credit\_limit FROM l\_employees;

-- 11-11 Method 1 - Using a row function CORRECT --?????

SELECT SUM(NVL(col\_2, 0)) + SUM(NVL(col\_3, 0)) AS columns\_added\_first,

SUM(NVL(col\_2, 0) + NVL(col\_3, 0)) AS rows\_added\_first

FROM sec1111;

-- 11-12

SELECT price \* quantity AS total,

NVL(tax, 0.07 \* price \* quantity) AS tax,

NVL(shipping, 0.12 \* price \* quantity) AS shipping

FROM sec1112\_shipping;

SELECT SUM((price \* quantity)

+ NVL(tax, 0.07 \* price \* quantity)

+ NVL(shipping, 0.12 \* price \* quantity)) AS total\_invoices

FROM sec1112\_shipping;

**12 Controlling the Level of Summarization**

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

\* AggregationSingleTable.sql

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

-- Number of descriptions in each language

SELECT language\_id, COUNT(\*)

FROM oe.product\_descriptions

GROUP BY language\_id

ORDER BY language\_id;

-- Sales statistics, by day

SELECT \* FROM oe.orders;

SELECT TRUNC(order\_date),

COUNT(\*) AS nbr\_orders,

SUM(order\_total) AS total\_sales,

ROUND(AVG(order\_total),2) AS avg\_sales

FROM oe.orders

GROUP BY TRUNC(order\_date);

-- Sales statistics, by month

SELECT TO\_CHAR(order\_date, 'MM/YY') AS month,

COUNT(\*) AS nbr\_orders,

SUM(order\_total) AS total\_sales,

ROUND(AVG(order\_total),2) AS avg\_sales

FROM oe.orders

GROUP BY TO\_CHAR(order\_date, 'MM/YY'),

TO\_CHAR(order\_date, 'YYYY-MM')

ORDER BY TO\_CHAR(order\_date, 'YYYY-MM');

-- Customers email domain count

SELECT SUBSTR(cust\_email, INSTR(cust\_email, '@') + 1) AS Domain,

COUNT(\*)

FROM oe.customers

GROUP BY SUBSTR(cust\_email, INSTR(cust\_email, '@') + 1)

ORDER BY COUNT(\*) DESC;

-- Count number of zip codes in each states

SELECT state,

COUNT(\*) nbr\_zips

FROM zip\_codes

WHERE latitude <> 0

AND longitude <> 0

GROUP BY state

ORDER BY COUNT(\*) DESC;

-- Only those states entirely north of Mason-Dixon line

SELECT state,

COUNT(\*) nbr\_zips

FROM zip\_codes

WHERE latitude <> 0

AND longitude <> 0

GROUP BY state

HAVING MIN(latitude) > 39 + 43/60 + 20/60/60

ORDER BY COUNT(\*);

**13 Inner Joins**

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

\* ClassRoomsJoins.sql

\* CIS-2720 and CIS-2725

\*

\* Note: this script will run on either Oracle or SQL Server

\* (except where noted).

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

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

SELECT \* FROM Course\_Sections;

SELECT \* FROM Classrooms;

-- Section #1

--

-- Cross join (a.k.a. Cartesian product or Cartesian join)

-- Implicitly defined join (no "join" keyword)

-- Without table aliases

-- select "star" used - thus table names not specified for each column

SELECT \*

FROM Course\_Sections,

Classrooms;

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

-- Section #2

--

-- Cross join (a.k.a. Cartesian product or Cartesian join)

-- Implicitly defined join (no "join" keyword)

-- Without table aliases

-- select "star" used for Course\_Sections

-- individual columns selected for Classrooms

SELECT Course\_Sections.\*,

Classrooms.Room\_Nbr,

Classrooms.Capacity,

Classrooms.Projector\_Flag

FROM Course\_Sections,

Classrooms;

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

-- Section #3

--

-- Cross join (a.k.a. Cartesian product or Cartesian join)

-- Implicitly defined join (no "join" keyword)

-- With table aliases

-- select "star" used for Course\_Sections

-- individual columns selected (no "star")

SELECT s.Section,

s.Class\_Name,

r.Room\_Nbr,

s.Students\_Registered,

r.Capacity AS Room\_Capacity

FROM Course\_Sections s, -- Cannot use "AS" here in Oracle

Classrooms r; -- Cannot use "AS" here in Oracle

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

-- Section #4

--

-- Inner join

-- Implicitly defined (old-school)

-- With table aliases

SELECT s.Section,

s.Class\_Name,

r.Room\_Nbr,

s.Students\_Registered,

r.Capacity AS Room\_Capacity

FROM Course\_Sections s,

Classrooms r

WHERE s.Room\_Nbr = r.Room\_Nbr

ORDER BY s.section;

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

-- Section #5

--

-- Inner join

-- Implicitly defined (old-school)

-- with table aliases

-- with extra filter in where clause

SELECT s.Section,

s.Class\_Name,

r.Room\_Nbr,

s.Students\_Registered,

r.Capacity AS Room\_Capacity

FROM Course\_Sections s,

Classrooms r

WHERE s.Room\_Nbr = r.Room\_Nbr

AND r.Room\_Nbr = 'SCC-135';

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

-- Section #6

--

-- Inner join

-- Explicitly defined

-- With table aliases

SELECT s.Section,

s.Class\_Name,

r.Room\_Nbr,

s.Students\_Registered,

r.Capacity AS Room\_Capacity

FROM Course\_Sections s

INNER JOIN Classrooms r -- "INNER" is optional (it's the default)

ON s.Room\_Nbr = r.Room\_Nbr;

SELECT s.Section,

s.Class\_Name,

r.Room\_Nbr,

s.Students\_Registered,

r.Capacity AS Room\_Capacity

FROM Course\_Sections s

JOIN Classrooms r -- "INNER" is optional (it's the default)

ON s.Room\_Nbr = r.Room\_Nbr

WHERE r.Room\_Nbr = 'SCC-135';

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

-- Section #7

--

-- Inner join

-- Explicitly defined

-- With table aliases

-- Order of tables has been reversed

SELECT s.Section,

s.Class\_Name,

r.Room\_Nbr,

s.Students\_Registered,

r.Capacity AS Room\_Capacity

FROM Classrooms r

INNER JOIN Course\_Sections s

ON s.Room\_Nbr = r.Room\_Nbr;

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

-- Section #8

--

-- Inner join

-- Explicitly defined

-- Without table aliases

SELECT Course\_Sections.Section,

Course\_Sections.Class\_Name,

Classrooms.Room\_Nbr, -- What about Course\_Sections.Room\_Nbr?

Course\_Sections.Students\_Registered,

Classrooms.Capacity AS Room\_Capacity

FROM Course\_Sections

INNER JOIN Classrooms

ON Course\_Sections.Room\_Nbr = Classrooms.Room\_Nbr;

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

-- Section #9

--

-- Inner join

-- Explicitly defined

-- With table aliases

-- Where clause added

-- tow are the same

SELECT s.Section,

s.Class\_Name,

r.Room\_Nbr,

s.Students\_Registered,

r.Capacity AS Room\_Capacity

FROM Course\_Sections s -- Cannot use "AS" here in Oracle

INNER JOIN Classrooms r

ON s.Room\_Nbr = r.Room\_Nbr

WHERE r.Room\_Nbr = 'SCC-135';

SELECT s.Section,

s.Class\_Name,

r.Room\_Nbr,

s.Students\_Registered,

r.Capacity AS Room\_Capacity

FROM Course\_Sections s -- Cannot use "AS" here in Oracle

INNER JOIN Classrooms r

ON s.Room\_Nbr = r.Room\_Nbr

AND r.Room\_Nbr = 'SCC-135';

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

-- Section #10

--

-- Inner join

-- Explicitly defined

-- With table aliases

-- Non-equijoin

-- Note: authoer has a better example in section 13-11

-- two are the same

SELECT s.Section,

s.Class\_Name,

r.Room\_Nbr,

s.Students\_Registered,

r.Capacity AS Room\_Capacity

FROM Course\_Sections s -- Cannot use "AS" here in Oracle

INNER JOIN Classrooms r

ON s.Room\_Nbr = r.Room\_Nbr

AND s.Students\_Registered > r.Capacity;

SELECT s.Section,

s.Class\_Name,

r.Room\_Nbr,

s.Students\_Registered,

r.Capacity AS Room\_Capacity

FROM Course\_Sections s -- Cannot use "AS" here in Oracle

JOIN Classrooms r

ON s.Room\_Nbr = r.Room\_Nbr

AND s.Students\_Registered > r.Capacity;

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

-- Section #11

--

-- Left join

-- Explicitly defined

-- With table aliases

SELECT s.Section,

s.Class\_Name,

r.Room\_Nbr,

s.Students\_Registered,

r.Capacity AS Room\_Capacity

FROM Course\_Sections s -- Cannot use "AS" here in Oracle

LEFT OUTER JOIN Classrooms r -- "OUTER" is optional

ON s.Room\_Nbr = r.Room\_Nbr;

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

-- Section #12

--

-- Left join

-- Explicitly defined

-- With table aliases

-- Order of tables has been reversed

SELECT s.Section,

s.Class\_Name,

r.Room\_Nbr,

s.Students\_Registered,

r.Capacity AS Room\_Capacity

FROM Classrooms r -- Cannot use "AS" here in Oracle

LEFT OUTER JOIN Course\_Sections s -- "OUTER" is optional

ON s.Room\_Nbr = r.Room\_Nbr;

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

-- Section #13

--

-- Right join

-- Explicitly defined

-- With table aliases

-- Removded the work "OUTER"

SELECT s.Section,

s.Class\_Name,

r.Room\_Nbr,

s.Students\_Registered,

r.Capacity AS Room\_Capacity

FROM Course\_Sections s -- Cannot use "AS" here in Oracle

RIGHT JOIN Classrooms r -- "OUTER" left out (it is optional)

ON s.Room\_Nbr = r.Room\_Nbr;

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

-- Section #14

--

-- Full join

-- Explicitly defined

-- With table aliases

SELECT s.Section,

s.Class\_Name,

r.Room\_Nbr,

s.Students\_Registered,

r.Capacity AS Room\_Capacity

FROM Course\_Sections s -- Cannot use "AS" here in Oracle

FULL JOIN Classrooms r

ON s.Room\_Nbr = r.Room\_Nbr;

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

-- Section #15

--

-- Cross join (a.k.a. cartesian product or Cartesian join)

-- Explicitly defined

-- With table aliases

SELECT s.Section,

s.Class\_Name,

r.Room\_Nbr,

s.Students\_Registered,

r.Capacity AS Room\_Capacity

FROM Course\_Sections s -- Cannot use "AS" here in Oracle

CROSS JOIN Classrooms r;

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

-- Section #16

--

-- Cross join (a.k.a. cartesian product or Cartesian join)

-- Explicitly defined

-- With table aliases

-- Note: this is a review from the top of this script

SELECT s.Section,

s.Class\_Name,

r.Room\_Nbr,

s.Students\_Registered,

r.Capacity AS Room\_Capacity

FROM Course\_Sections s,

Classrooms r;

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

-- Section #17

--

-- Same query as section #11

-- Uses old Oracle only syntax for outer joins

-- Oracle added the ANSI syntax in Oracle9i

-- DOES NOT RUN on SQL Server

-- Do not use this notation, use the ANSI notation in section #11 instead

SELECT s.Section,

s.Class\_Name,

r.Room\_Nbr,

s.Students\_Registered,

r.Capacity AS Room\_Capacity

FROM Course\_Sections s,

Classrooms r

WHERE s.Room\_Nbr = r.Room\_Nbr (+);

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

-- Section #18

--

-- Same query as section #11

-- Uses old SQL Server only syntax for outer joins

-- DOES NOT RUN on Oracle

-- Do not use this notation, use the ANSI notation in section #11 instead

--

-- Note that in SQL Server the "\*" goes on the side that will not

-- have null rows (Course\_Sections). In Oracle the "(+)" goes

-- on the side with the null rows (Classrooms).

SELECT \* FROM Course\_Sections;

SELECT \* FROM Classrooms;

SELECT s.Section,

s.Class\_Name,

r.Room\_Nbr,

s.Students\_Registered,

r.Capacity AS Room\_Capacity

FROM Course\_Sections s,

Classrooms r

WHERE s.Room\_Nbr \*= r.Room\_Nbr;

* Join three tables

SELECT \* FROM states;

SELECT \* FROM customers;

SELECT \* FROM zip\_codes;

INSERT INTO states VALUES ('IN', 'Indiana');

COMMIT;

SELECT DISTINCT state FROM customers;

SELECT c.customer\_id,

c.first\_name,

c.last\_name,

c.city,

s.state\_full\_name AS state,

c.zip\_code,

z.longitude,

z.latitude

FROM customers c

INNER JOIN states s

ON c.state = s.state

INNER JOIN zip\_codes z

ON c.zip\_code = z.zip\_code

WHERE c.state = 'IL'

**14 Outer Joins**

-- Oracle SQL for chapter 14

-- Sections 14-8 and 14-9 are okay. The rest of the examples have problems

-- and are confusing.

-- See the ClassRoomsJoins.sql and ClassRoomsJoins.sql files for a better

-- examples for chapters 13 + 14.

-- 14-1

-- inner join

SELECT \* FROM twos;

SELECT \* FROM threes;

SELECT a.\*,

b.\*

FROM twos a,

threes b

WHERE a.number\_2 = b.number\_3

ORDER BY a.number\_2;

-- 14-3

-- left outer join

SELECT \* FROM twos;

SELECT \* FROM threes;

SELECT a.\*,

b.\*

FROM twos a

LEFT OUTER JOIN threes b

ON a.number\_2 = b.number\_3

ORDER BY a.number\_2;

-- 14-4

-- right outer join

SELECT \* FROM twos;

SELECT \* FROM threes;

SELECT a.\*,

b.\*

FROM twos a

right OUTER JOIN threes b

ON a.number\_2 = b.number\_3

ORDER BY b.number\_3;

-- 14-5

-- full outer join

SELECT \* FROM twos;

SELECT \* FROM threes;

SELECT a.\*,

b.\*

FROM twos a

FULL OUTER JOIN threes b

ON a.number\_2 = b.number\_3;

-- 14-6

-- This SQL IS NOT intended to be run

-- 14-7 - Unions are covered IN detail IN chapter 16

SELECT \* FROM sec1407\_first;

SELECT \* FROM sec1407\_second;

SELECT a.number\_1,

a.word\_1 AS text\_1,

a.date\_1

FROM sec1407\_first a

UNION

SELECT b.number\_2,

b.word\_2,

b.date\_2

FROM sec1407\_second b

ORDER BY number\_1;

-- 14-8

SELECT \* FROM l\_employees;

SELECT \* FROM l\_lunches;

SELECT a.employee\_id,

a.first\_name,

a.last\_name,

b.lunch\_id

FROM l\_employees a

LEFT OUTER JOIN l\_lunches b

ON a.employee\_id = b.employee\_id

ORDER BY employee\_id, employee\_id;

SELECT a.employee\_id,

a.first\_name,

a.last\_name,

COUNT(b.lunch\_id) AS number\_of\_lunches

FROM l\_employees a

LEFT OUTER JOIN l\_lunches b

ON a.employee\_id = b.employee\_id

GROUP BY a.employee\_id,

a.first\_name,

a.last\_name

ORDER BY a.employee\_id;

SELECT a.employee\_id,

a.first\_name,

a.last\_name,

COUNT(b.lunch\_id) AS number\_of\_lunches

FROM l\_employees a

JOIN l\_lunches b

ON a.employee\_id = b.employee\_id

GROUP BY a.employee\_id,

a.first\_name,

a.last\_name

ORDER BY a.employee\_id; -- miss some which have not the lunch

SELECT a.employee\_id,

a.first\_name,

a.last\_name,

COUNT(\*) AS number\_of\_lunches -- count the employee\_id

FROM l\_employees a

LEFT OUTER JOIN l\_lunches b

ON a.employee\_id = b.employee\_id

GROUP BY a.employee\_id,

a.first\_name,

a.last\_name

ORDER BY a.employee\_id;

-- 14-9

-- 14-9 CREATE the TABLE

SELECT \* FROM l\_employees;

SELECT \* FROM l\_lunches;

CREATE TABLE sec1409 AS

SELECT a.\*,

b.lunch\_id,

b.lunch\_date,

b.employee\_id AS employee\_id2,

b.date\_entered

FROM l\_employees a

LEFT OUTER JOIN l\_lunches b

ON a.employee\_id = b.employee\_id;

-- 14-9 see the TABLE

SELECT \* FROM sec1409;

-- 14-9 example 2

SELECT employee\_id,

first\_name,

last\_name,

COUNT(lunch\_id) AS number\_of\_lunches

FROM sec1409

GROUP BY employee\_id,

first\_name,

last\_name

ORDER BY employee\_id;

-- 14-10

SELECT \* FROM twos;

SELECT \* FROM threes;

SELECT a.\*,

b.\*,

NVL(a.number\_2, b.number\_3) AS sort\_order

FROM twos a

FULL OUTER JOIN threes b

ON a.number\_2 = b.number\_3

ORDER BY sort\_order;

-- NVL is counting null

-- 14-11

SELECT \* FROM numbers\_1\_to\_1000;

SELECT \* FROM sec1411\_numbers;

SELECT a.n,

b.n,

COUNT(b.n)

FROM numbers\_1\_to\_1000 a

LEFT OUTER JOIN sec1411\_numbers b

ON a.n = b.n

GROUP BY a.n,

b.n

HAVING NOT (COUNT(b.n) = 1) -- count numbers

ORDER BY a.n;

-- 14-12

SELECT \* FROM sec1412a;

SELECT \* FROM sec1412b;

SELECT a.first\_col,

a.second\_col,

b.first\_col,

b.second\_col

FROM sec1412a a

LEFT OUTER JOIN sec1412b b

ON a.first\_col = b.first\_col

AND a.second\_col = b.second\_col

WHERE b.first\_col IS NULL

ORDER BY a.first\_col,

a.second\_col;

-- 14-13 first full outer join

SELECT \* FROM sec1413\_fruits;

SELECT \* FROM sec1413\_colors;

SELECT a.\*,

b.\*

FROM sec1413\_fruits a

FULL OUTER JOIN sec1413\_colors b

ON a.f\_num = b.c\_num;

-- 14-13 second full outer join

SELECT \* FROM sec1413\_fruits;

SELECT \* FROM sec1413\_colors;

SELECT a.\*,

b.\*

FROM sec1413\_fruits a

FULL OUTER JOIN sec1413\_colors b

ON a.fruit = b.color;

SELECT a.\*,

b.\*

FROM sec1413\_fruits a

JOIN sec1413\_colors b

ON a.fruit = b.color;

-- 14-14

-- 14-14 CREATE (A x B)

SELECT \* FROM sec1414\_twos;

SELECT \* FROM sec1414\_threes;

CREATE TABLE sec1414\_AxB AS

SELECT a.\*,

b.\*

FROM sec1414\_twos a

LEFT OUTER JOIN sec1414\_threes b

ON a.multiple\_of\_2 = b.multiple\_of\_3;

-- 14-14 CREATE (A x B) x C

SELECT \* FROM sec1414\_AxB ORDER BY MULTIPLE\_OF\_2;

SELECT \* FROM sec1414\_fives;

CREATE TABLE sec1414\_AxB\_xC AS

SELECT a.\*,

b.\*

FROM sec1414\_AxB a

LEFT OUTER JOIN sec1414\_fives b

ON (a.multiple\_of\_2 = b.multiple\_of\_5

OR a.multiple\_of\_3 = b.multiple\_of\_5);

SELECT \* FROM sec1414\_AxB\_xC ORDER BY MULTIPLE\_OF\_5;

-- 14-14 CREATE (B x C)

SELECT \* FROM sec1414\_threes;

SELECT \* FROM sec1414\_fives;

CREATE TABLE sec1414\_BxC AS

SELECT a.\*,

b.\*

FROM sec1414\_threes a

LEFT OUTER JOIN sec1414\_fives b

ON a.multiple\_of\_3 = b.multiple\_of\_5;

SELECT \* FROM sec1414\_BxC ORDER BY MULTIPLE\_OF\_5;

-- 14-14 CREATE A x (B x C)

SELECT \* FROM sec1414\_twos;

SELECT \* FROM sec1414\_BxC;

CREATE TABLE sec1414\_Ax\_BxC AS

SELECT a.\*,

b.\*

FROM sec1414\_twos a

LEFT OUTER JOIN sec1414\_BxC b

ON (a.multiple\_of\_2 = b.multiple\_of\_3

OR a.multiple\_of\_2 = b.multiple\_of\_5);

-- 14-14 Compare (A x B) x C with A x (B x C)

-- Unions are covered in chapter 15

SELECT \* FROM sec1414\_AxB\_xC;

SELECT \* FROM sec1414\_Ax\_BxC;

SELECT a.\*,

'(AxB)xC' AS source

FROM sec1414\_AxB\_xC a

UNION ALL

SELECT b.\*,

'Ax(BxC)'

FROM sec1414\_Ax\_BxC b

ORDER BY 1, 2, 3, 4;

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

\* OuterJoinExample.sql

\* CIS-2720

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

/\*

Using the "hr" schema, write a query that shows the following information

for each department:

Department ID

Department Name

Managers Name - First name and last name, separated with a space.

Display a single blank space if the department does not have a manager.

Number of Employees in department

Order the results by the number of employees in each department, in descending

order. Show a row for employees who are not assigned a department with the

department name "Unassigned".

-- this is the correct query

SELECT d.department\_id,

d.department\_name,

NVL2(ed.last\_name, ed.first\_name || ' ' || ed.last\_name, '\*\*\* Vacant \*\*\*') AS manager\_name

FROM depcopy d

LEFT OUTER JOIN empcopy ed

ON d.manager\_id = ed.employee\_id;

SELECT d.department\_id,

d.department\_name,

COUNT(ed.employee\_id) AS number\_of\_employees

FROM depcopy d

LEFT JOIN empcopy ed

ON d.department\_id = ed.department\_id

GROUP BY d.department\_id, d.department\_name;

-- combine two queries to one

SELECT d.department\_id,

d.department\_name,

NVL2(ed.last\_name, ed.first\_name || ' ' || ed.last\_name, '\*\*\* Vacant \*\*\*') AS manager\_name,

COUNT(e.employee\_id) AS number\_of\_employees

FROM depcopy d

LEFT JOIN empcopy ed

ON d.manager\_id = ed.employee\_id

LEFT JOIN empcopy e

ON d.department\_id = ed.department\_id

GROUP BY d.department\_id,

d.department\_name,

ed.last\_name,

ed.first\_name

ORDER BY COUNT(e.employee\_id) DESC;

\*/

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

-- Part 1 - Table Setup

-- Rememeber - ORA-942 messages are okay here

-- Think of them as a warning, not an error

DROP TABLE depcopy;

DROP TABLE empcopy;

CREATE TABLE empcopy AS

SELECT \* FROM hr.employees;

CREATE TABLE depcopy AS

SELECT \* FROM hr.departments;

ALTER TABLE empcopy

ADD CONSTRAINT pk\_empcopy

PRIMARY KEY (employee\_id);

ALTER TABLE depcopy

ADD CONSTRAINT pk\_depcopy

PRIMARY KEY (department\_id);

ALTER TABLE depcopy

ADD CONSTRAINT fk\_depcopy\_empcopy

FOREIGN KEY (manager\_id)

REFERENCES empcopy (employee\_id);

-- Take three employees out of department 50, leave them unassigned

UPDATE empcopy

SET department\_id = NULL

WHERE employee\_id IN (139, 144, 182);

-- Take four employees out of department 50, move to 140 "Control And Credit"

UPDATE empcopy

SET department\_id = 140

WHERE employee\_id IN (134, 183, 120, 194);

COMMIT;

-- Make manage of finance also manager of accounting, shareholder services,

-- and payroll

UPDATE depcopy

SET manager\_id = 108

WHERE department\_id IN (110, 150, 270);

COMMIT;

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

-- Part 2 - View/analyze the data

SELECT \* FROM empcopy;

SELECT \* FROM depcopy;

-- 107 employees, 4 are not assigned to a department

SELECT COUNT(\*), COUNT(department\_id) FROM empcopy;

-- count\* give the numbers of columns

-- 27 departments, but only 13 have managers

SELECT COUNT(\*), COUNT(manager\_id), COUNT(DISTINCT manager\_id) FROM depcopy;

-- 27 departments, 13 managers in different departments, 10 managers (3 managers in charge of two departments)

-- 27 departments, but only 11 have employees

SELECT department\_id, COUNT(\*)

FROM empcopy

GROUP BY department\_id

ORDER BY department\_id;

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

-- Part 3 - Write query to show all departments with manager column

-- Step #1

-- LEFT JOIN needed, other wise departments without a manager would not be shown

-- Try with INNER JOIN to see the results.

-- Note the join is on different column names (d.manager\_id = e.employee\_id)

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

\* START THIS QUERY FROM SCRATCH! Don't copy query from previous step \*

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

-- Alias "ed" stands for "Employee - Department" - this will make sense later

SELECT d.department\_id, d.department\_name,

NVL2(ed.last\_name, ed.first\_name || ' '

|| ed.last\_name, '\*\*\* Vacant \*\*\*') AS manager\_name

FROM depcopy d

LEFT JOIN empcopy ed

ON d.manager\_id = ed.employee\_id

ORDER BY d.department\_name;

-- Step #2 - Example of wrong query

-- What if add an addional condition to the LEFT JOIN?

-- This one seems harmless enough (even useful or necessary)

-- What happens to Nancy Greenberg?

SELECT d.department\_id, d.department\_name,

NVL2(ed.last\_name, ed.first\_name || ' '

|| ed.last\_name, '\*\*\* Vacant \*\*\*') AS manager\_name

FROM depcopy d

LEFT JOIN empcopy ed

ON d.manager\_id = ed.employee\_id

AND d.department\_id = ed.department\_id

ORDER BY d.department\_name;

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

-- Part 4 - Write query to show all departments with employee count

-- Step 1: Join the two tables.

-- Note this the join is on the same column name in each table.

-- Output is 122 rows, but there's only 107 rows in empcopy

-- Extra rows from departments that don't have any employees

-- Extra rows disapper if changed to INNER JOIN

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

\* START THIS QUERY FROM SCRATCH! Don't copy query from previous step \*

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

-- Alias "ec" stands for "Employee - Count" - this will make sense later

SELECT d.department\_id,

d.department\_name,

ec.employee\_id,

ec.first\_name || ' ' || ec.last\_name mgr\_name

FROM depcopy d

LEFT JOIN empcopy ec

ON d.department\_id = ec.department\_id

ORDER BY ec.employee\_id;

-- Step 2: Add aggregation

-- Why do departments with no employees show 1 employee?

SELECT d.department\_id,

d.department\_name,

COUNT(\*) num\_emps

FROM depcopy d

LEFT JOIN empcopy ec

ON d.department\_id = ec.department\_id

GROUP BY d.department\_id, d.department\_name

ORDER BY COUNT(\*) DESC;

-- Step 3: Add aggregation

-- Don't include null rows in COUNT()

SELECT d.department\_id,

d.department\_name,

COUNT(ec.employee\_id) num\_emps

FROM depcopy d

LEFT JOIN empcopy ec

ON d.department\_id = ec.department\_id

GROUP BY d.department\_id, d.department\_name

ORDER BY COUNT(ec.employee\_id) DESC;

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

-- Part 5 - Write 3 table join - no aggregation

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

\* START THIS QUERY FROM SCRATCH! Don't copy query from previous step \*

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

-- Two copies of emp? How can this be?

SELECT d.department\_id,

d.department\_name,

NVL2(ed.last\_name, ed.first\_name || ' '

|| ed.last\_name, '\*\*\* Vacant \*\*\*') AS mgr\_name,

ec.employee\_id

FROM depcopy d

LEFT JOIN empcopy ed

ON d.manager\_id = ed.employee\_id

LEFT JOIN empcopy ec

ON d.department\_id = ec.department\_id;

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

-- Part 6 - Add the aggregaction (back in)

SELECT d.department\_id,

d.department\_name,

NVL2(ed.last\_name, ed.first\_name || ' '

|| ed.last\_name, '\*\*\* Vacant \*\*\*') AS mgr\_name,

COUNT(ec.employee\_id)

FROM depcopy d

LEFT JOIN empcopy ed

ON d.manager\_id = ed.employee\_id

LEFT JOIN empcopy ec

ON d.department\_id = ec.department\_id

GROUP BY d.department\_id,

d.department\_name,

NVL2(ed.last\_name, ed.first\_name || ' '

|| ed.last\_name, '\*\*\* Vacant \*\*\*')

ORDER BY COUNT(ec.employee\_id) DESC;

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

-- Part 7 - Add extra row for employees not assigned to a department

-- These employees are not included in the query above

SELECT \* FROM empcopy WHERE department\_id IS NULL;

-- A useful example of a FULL JOIN!

-- We need all rows from d, even if the department has not employees

-- We also need all rows from ec, even if employee is not assigned a department

-- No aggregation

-- Some rows have nulls in department\_it and department\_name

-- Other rows have nulls in employee\_id

-- Careful which join you change from LEFT to FULL!

SELECT d.department\_id,

d.department\_name,

NVL2(ed.last\_name, ed.first\_name || ' '

|| ed.last\_name, '\*\*\* Vacant \*\*\*') AS mgr\_name,

ec.employee\_id

FROM depcopy d

LEFT JOIN empcopy ed

ON d.manager\_id = ed.employee\_id

FULL JOIN empcopy ec

ON d.department\_id = ec.department\_id;

-- Add aggregation back in - use NVL for department\_id and department\_name

SELECT NVL(d.department\_id, 999) AS department\_id,

NVL(d.department\_name, 'Unassigned') AS department\_name,

NVL2(ed.last\_name, ed.first\_name || ' '

|| ed.last\_name, '\*\*\* Vacant \*\*\*') AS mgr\_name,

COUNT(ec.employee\_id)

FROM depcopy d

LEFT JOIN empcopy ed

ON d.manager\_id = ed.employee\_id

FULL JOIN empcopy ec

ON d.department\_id = ec.department\_id

GROUP BY NVL(d.department\_id, 999),

NVL(d.department\_name, 'Unassigned'),

NVL2(ed.last\_name, ed.first\_name || ' '

|| ed.last\_name, '\*\*\* Vacant \*\*\*');

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

-- Part 8 - BONUS: Show only departments with at least 3 employees

-- Add HAVING clause

SELECT NVL(d.department\_id, 999) AS department\_id,

NVL(d.department\_name, 'Unassigned') AS department\_name,

NVL2(ed.last\_name, ed.first\_name || ' '

|| ed.last\_name, '\*\*\* Vacant \*\*\*') AS mgr\_name,

COUNT(ec.employee\_id)

FROM depcopy d

LEFT JOIN empcopy ed

ON d.manager\_id = ed.employee\_id

FULL JOIN empcopy ec

ON d.department\_id = ec.department\_id

HAVING COUNT(ec.employee\_id) >= 3

GROUP BY NVL(d.department\_id, 999),

NVL(d.department\_name, 'Unassigned'),

NVL2(ed.last\_name, ed.first\_name || ' '

|| ed.last\_name, '\*\*\* Vacant \*\*\*');

**15 Union and Union All**

-- Oracle SQL for chapter 15

-- 15-1 a UNION

SELECT \* FROM sec1501\_first; -- 1 to 5

SELECT \* FROM sec1501\_second; -- 3 to 7

--Note: no duplicates

SELECT number\_1,

word\_1,

date\_1

FROM sec1501\_first

UNION

SELECT number\_2,

word\_2,

date\_2

FROM sec1501\_second

ORDER BY number\_1;

-- 15-1 an inner join

-- Compare union to inner join. Both can link two tables, but differently

SELECT \* FROM sec1501\_first;

SELECT \* FROM sec1501\_second;

SELECT a.\*,

b.\*

FROM sec1501\_first a,

sec1501\_second b

WHERE a.number\_1 = b.number\_2

ORDER BY a.number\_1;

-- 15-2

SELECT \* FROM sec1502\_first; -- 1 to 5

SELECT \* FROM sec1502\_second; -- 3 to 7

-- Note: duplicates

SELECT number\_1,

word\_1,

date\_1

FROM sec1502\_first

UNION ALL

SELECT number\_2,

word\_2,

date\_2

FROM sec1502\_second

ORDER BY number\_1;

-- 15-3

SELECT \* FROM l\_employees;

SELECT \* FROM l\_lunches;

SELECT a.last\_name,

a.first\_name,

COUNT(b.lunch\_id) AS number\_of\_lunches

FROM l\_employees a,

l\_lunches b

WHERE a.employee\_id = b.employee\_id

GROUP BY a.last\_name, a.first\_name

HAVING COUNT(b.lunch\_id) < 5

UNION ALL

SELECT 'PATRICK',

'JOHN',

0

FROM DUAL

ORDER BY last\_name;

-- 15-4 Use a column name in the order by clause

SELECT \* FROM sec1504\_first; -- 1 to 5

SELECT \* FROM sec1504\_second; -- 3 to 7

SELECT number\_1,

word\_1,

date\_1

FROM sec1504\_first

UNION

SELECT \*

FROM sec1504\_second

ORDER BY word\_1;

-- 15-4 example 2, Use a column alias in the order by clause

SELECT \* FROM sec1504\_first; -- 1 to 5

SELECT \* FROM sec1504\_second; -- 3 to 7

SELECT number\_1,

word\_1 AS text\_1,

date\_1

FROM sec1504\_first

UNION

SELECT \*

FROM sec1504\_second

ORDER BY text\_1;

-- 15-4 example 3

-- Access SQL only

-- 15-4 example 4, Use a column number in the order by clause

SELECT \* FROM sec1504\_first; -- 1 to 5

SELECT \* FROM sec1504\_second; -- 3 to 7

SELECT \*

FROM sec1504\_first

UNION

SELECT \*

FROM sec1504\_second

ORDER BY 2;

-- 15-5 CREATE a VIEW

SELECT \* FROM sec1505\_first;

SELECT \* FROM sec1505\_second;

CREATE OR REPLACE VIEW sec1505a\_view AS

SELECT \*

FROM sec1505\_first

UNION

SELECT \*

FROM sec1505\_second

ORDER BY 2;

-- 15-5 see the VIEW

SELECT \* FROM sec1505a\_view;

-- 15-5 CREATE a TABLE

CREATE TABLE sec1505b\_table AS

SELECT \*

FROM sec1505\_first

UNION

SELECT \*

FROM sec1505\_second

ORDER BY 2;

-- 15-5 see the TABLE

SELECT \* FROM sec1505b\_table;

-- 15-6

-- 15-6

SELECT \* FROM sec1506\_with\_long\_columns;

SELECT \* FROM sec1506\_with\_short\_columns;

CREATE OR REPLACE VIEW sec1506\_union\_view AS

SELECT number\_column\_with\_length\_7 AS number\_column,

text\_column\_with\_length\_7 AS text\_column

FROM sec1506\_with\_long\_columns

UNION

SELECT number\_column\_with\_length\_2,

text\_column\_with\_length\_2

FROM sec1506\_with\_short\_columns

ORDER BY 1;

-- 15-6 see the VIEW

SELECT \*

FROM sec1506\_union\_view;

-- 15-7

-- 15-7

SELECT \* FROM sec1507\_with\_long\_columns;

SELECT \* FROM sec1507\_with\_short\_columns;

-- Union with columns of different datatypes

-- Modified by KF - changed sec1506\_with\_short\_columns to sec1507\_with\_short\_columns

CREATE OR REPLACE VIEW sec1507\_union\_view AS

SELECT TO\_CHAR(number\_column\_with\_length\_7) AS first\_column,

text\_column\_with\_length\_7 AS second\_column

FROM sec1507\_with\_long\_columns

UNION

SELECT text\_column\_with\_length\_2,

TO\_CHAR(number\_column\_with\_length\_2)

FROM sec1507\_with\_short\_columns

ORDER BY 1;

-- 15-7 see the VIEW

SELECT \*

FROM sec1507\_union\_view;

-- 15-8

SELECT \* FROM sec1508\_more\_columns;

SELECT \* FROM sec1508\_less\_columns;

SELECT a.number\_col,

a.text\_col,

a.date\_col

FROM sec1508\_more\_columns a

UNION

SELECT b.number\_col,

b.text\_col,

NULL

FROM sec1508\_less\_columns b;

-- 15-9 Step 1 - Determine if the tables have the same number of rows

SELECT \* FROM l\_foods;

SELECT \* FROM sec1509\_foods;

SELECT COUNT(\*) FROM l\_foods;

SELECT COUNT(\*) FROM sec1509\_foods;

-- 15-9 Step 2 - create a view that IS the union of both tables

-- Determining if two tables are identical

CREATE OR REPLACE VIEW sec1509\_union\_view AS

SELECT \* FROM l\_foods

UNION

SELECT \* FROM sec1509\_foods;

-- 15-9 step2, see the VIEW

SELECT \* FROM sec1509\_union\_view;

-- 15-9 Step 3 - COUNT the NUMBER of rows IN the VIEW

SELECT COUNT(\*)

FROM sec1509\_union\_view;

-- 15-10

SELECT \* FROM sec1510\_first; -- 1 to 5

SELECT \* FROM sec1510\_second; -- 3 to 7

SELECT number\_1,

word\_1,

date\_1,

'FROM the first TABLE' AS source\_of\_the\_data

FROM sec1510\_first

UNION ALL

SELECT number\_2,

word\_2,

date\_2,

'FROM the second TABLE'

FROM sec1510\_second

ORDER BY number\_1;

-- 15-11

SELECT \* FROM l\_foods;

SELECT description,

price,

'EXPENSIVE ITEM' AS message

FROM l\_foods

WHERE price > 2.00

UNION ALL

SELECT description,

price,

' '

FROM l\_foods

WHERE NOT (price > 2.00)

OR price IS NULL

ORDER BY description;

-- 15-12

SELECT \* FROM sec1512\_finances;

SELECT item,

NULL AS debits,

cost AS credits

FROM sec1512\_finances

WHERE cost > 0

UNION ALL

SELECT item,

cost,

NULL

FROM sec1512\_finances

WHERE cost < 0

OR cost IS NULL

ORDER BY item;

-- 15-13

SELECT \* FROM l\_foods;

SELECT menu\_item,

description,

price + (price \* .05) AS new\_price

FROM l\_foods

WHERE price > 2.00

OR price IS NULL

UNION ALL

SELECT menu\_item,

description,

price + (price \* .10)

FROM l\_foods

WHERE price <= 2.00

ORDER BY menu\_item;

-- 15-14

SELECT 'A' AS letters

FROM DUAL

UNION

SELECT 'B'

FROM DUAL

UNION

SELECT 'C'

FROM DUAL

UNION

SELECT 'D'

FROM DUAL

UNION

SELECT 'E'

FROM DUAL

ORDER BY 1;

-- 15-15

SELECT \* FROM sec1515\_first; -- 1 to 5

SELECT \* FROM sec1515\_second; -- 3 to 7

SELECT number\_1,

word\_1,

date\_1

FROM sec1515\_first

INTERSECT

SELECT number\_2,

word\_2,

date\_2

FROM sec1515\_second

ORDER BY number\_1;

-- 15-16 step 1

-- Note: Use "EXCEPT" instead of "MINUS" in SQL Server

SELECT \* FROM sec1516\_first;

SELECT \* FROM sec1516\_second;

SELECT number\_1,

word\_1,

date\_1

FROM sec1516\_first

MINUS

SELECT number\_2,

word\_2,

date\_2

FROM sec1516\_second

ORDER BY number\_1;

-- 15-16 step 2

SELECT \* FROM sec1516\_first;

SELECT \* FROM sec1516\_second;

SELECT number\_2,

word\_2,

date\_2

FROM sec1516\_second

MINUS

SELECT number\_1,

word\_1,

date\_1

FROM sec1516\_first

ORDER BY number\_2;

---Example in class

SELECT \* FROM L\_EMPLOYEES;

SELECT \* FROM HR.EMPLOYEES;

-- cross join

SELECT \*

FROM l\_employees,

hr.employees;

-- union

SELECT employee\_id + 1000 AS employee\_id,

first\_name,

last\_name,

NVL2(phone\_number, '312.555' || phone\_number,'Unknown') AS phone\_number,

'Chicago' AS city,

credit\_limit

FROM l\_employees

UNION ALL

SELECT employee\_id + 2000 AS employee\_id,

UPPER(first\_name),

UPPER(last\_name),

phone\_number,

'San Francisco' AS city,

10

FROM hr.employees

ORDER BY last\_name, first\_name;

**16 Cross Joins, Self Joins, and CrossTab Queries**

-- Oracle SQL for chapter 16

-- The following sections are good:

-- 16-5 - bad join and correction

-- 16-6 - example of a useful cross join

-- 16-10 - example of a useful cross join

-- 16-11 - amount of time between two rows

-- There is more on decode in chapter 18

-- 16-1 - Implicit Cross Join

SELECT \* FROM sec1601\_columns\_1\_to\_2;

SELECT \* FROM sec1601\_columns\_3\_to\_5;

SELECT a.\*,

b.\*

FROM sec1601\_columns\_1\_to\_2 a,

sec1601\_columns\_3\_to\_5 b;

-- 16-1b - Added by KF

-- Explicit Cross Join

SELECT \* FROM sec1601\_columns\_1\_to\_2;

SELECT \* FROM sec1601\_columns\_3\_to\_5;

SELECT a.\*,

b.\*

FROM sec1601\_columns\_1\_to\_2 a

CROSS JOIN sec1601\_columns\_3\_to\_5 b;

-- 16-3

SELECT \* FROM sec1603\_fruits;

SELECT \* FROM sec1603\_colors;

SELECT a.\*,

b.\*

FROM sec1603\_fruits a,

sec1603\_colors b

WHERE a.f\_num = b.c\_num;

-- 16-5 Join is wrong

SELECT \* FROM l\_lunch\_items;

SELECT \* FROM l\_foods;

SELECT a.lunch\_id,

b.supplier\_id,

b.product\_code,

b.description,

b.price,

a.quantity

FROM l\_lunch\_items a,

l\_foods b

WHERE a.supplier\_id = b.supplier\_id

AND a.lunch\_id = 2;

-- 16-5 Join is correct

SELECT \* FROM l\_lunch\_items;

SELECT \* FROM l\_foods;

SELECT a.lunch\_id,

b.supplier\_id,

b.product\_code,

b.description,

b.price,

a.quantity

FROM l\_lunch\_items a,

l\_foods b

WHERE a.supplier\_id = b.supplier\_id

AND a.product\_code = b.product\_code

AND a.lunch\_id = 2;

-- 16-6

SELECT \* FROM sec1606\_car\_types;

SELECT \* FROM sec1606\_colors;

SELECT a.car\_type,

b.color

FROM sec1606\_car\_types a,

sec1606\_colors b

ORDER BY a.car\_type,

b.color;

-- 16-9

SELECT \* FROM sec1609;

SELECT \* FROM sec1609;

SELECT a.col\_1,

a.col\_2,

b.col\_1 AS col\_3,

b.col\_2 AS col\_4

FROM sec1609 a,

sec1609 b;

-- 16-10 - Self-Join (a.k.a. reflexive join, recursive join)

SELECT \* FROM l\_employees;

SELECT emp.employee\_id,

emp.last\_name,

emp.phone\_number,

boss.last\_name AS manager\_name,

boss.phone\_number AS manager\_phone

FROM l\_employees emp

LEFT OUTER JOIN l\_employees boss

ON emp.manager\_id = boss.employee\_id

ORDER BY emp.employee\_id;

-- 16-11 - Determine time between two events

DESC sec1611\_events

SELECT \* FROM sec1611\_events;

SELECT sequence\_number,

truck\_id,

event,

TO\_CHAR(time\_done, 'MM/DD/YYYY HH24:MI:SS') AS time\_done

FROM sec1611\_events;

SELECT b.event,

TO\_CHAR(a.time\_done, 'HH:MI') AS start\_time,

TO\_CHAR(b.time\_done, 'HH:MI') AS end\_time,

((b.time\_done - a.time\_done) \* 60 \* 24) AS minutes\_elapsed\_time

FROM sec1611\_events a,

sec1611\_events b

WHERE a.sequence\_number + 1 = b.sequence\_number

ORDER BY a.sequence\_number;

-- 16-12

-- 16-12 Step 1 - create a table to contain all 10 digits

CREATE TABLE my\_numbers\_0\_to\_9 (

digit NUMBER(1)

);

-- 16-12 Step 2 - Put data in the table

INSERT INTO my\_numbers\_0\_to\_9 VALUES (0);

INSERT INTO my\_numbers\_0\_to\_9 VALUES (1);

INSERT INTO my\_numbers\_0\_to\_9 VALUES (2);

INSERT INTO my\_numbers\_0\_to\_9 VALUES (3);

INSERT INTO my\_numbers\_0\_to\_9 VALUES (4);

INSERT INTO my\_numbers\_0\_to\_9 VALUES (5);

INSERT INTO my\_numbers\_0\_to\_9 VALUES (6);

INSERT INTO my\_numbers\_0\_to\_9 VALUES (7);

INSERT INTO my\_numbers\_0\_to\_9 VALUES (8);

INSERT INTO my\_numbers\_0\_to\_9 VALUES (9);

COMMIT;

-- 16-12 Step 3 - CREATE a TABLE of numbers FROM 0 to 999

SELECT \* FROM my\_numbers\_0\_to\_9;

CREATE TABLE numbers\_0\_to\_999 AS

SELECT ((a.digit \* 100) + (b.digit \* 10) + c.digit) AS n

FROM my\_numbers\_0\_to\_9 a,

my\_numbers\_0\_to\_9 b,

my\_numbers\_0\_to\_9 c

ORDER BY 1;

-- 16-12 see the TABLE

SELECT \* FROM numbers\_0\_to\_999;

-- 16-18

-- ADDED BY KF: Example of simple DECODE()

-- Author uses DECODE() here, but doesn't example until chapter 18

SELECT \* FROM l\_employees;

SELECT employee\_id,

last\_name,

dept\_code,

DECODE(dept\_code,'EXE', 'Executive') AS dept\_name

FROM l\_employees;

SELECT employee\_id,

last\_name,

dept\_code,

DECODE(dept\_code,'EXE', 'Executive', 'Non-executive') AS dept\_name

FROM l\_employees;

SELECT employee\_id,

last\_name,

dept\_code,

DECODE(dept\_code,

'EXE', 'Executive',

'SAL', 'Sales',

'SHP', 'Shipping',

'Other') AS dept\_name

FROM l\_employees;

-- Author's code:

SELECT \* FROM sec1618\_car\_types;

SELECT \* FROM sec1618\_number\_of\_doors;

SELECT \* FROM sec1618\_colors;

-- Just shows what a cross tab looks like. Doesn't actually do anything

SELECT a.car\_type,

b.doors,

' ' AS green,

' ' AS red,

' ' AS white

FROM sec1618\_car\_types a,

sec1618\_number\_of\_doors b,

sec1618\_colors c

GROUP BY a.car\_type,

b.doors

ORDER BY a.car\_type,

b.doors;

-- 16-19 First version - Showing just the effect of the DECODE function

SELECT \* FROM sec1619;

SELECT car\_type,

doors,

DECODE(color, 'GREEN', TO\_CHAR(my\_data), NULL) AS green,

DECODE(color, 'RED', TO\_CHAR(my\_data), NULL) AS red,

DECODE(color, 'WHITE', TO\_CHAR(my\_data), NULL) AS white

FROM sec1619

ORDER BY car\_type,

doors;

-- 16-19 complete AND final version - this IS the only SQL you need

SELECT car\_type,

doors,

MAX(DECODE(color, 'GREEN', TO\_CHAR(my\_data), NULL)) AS green,

MAX(DECODE(color, 'RED', TO\_CHAR(my\_data), NULL)) AS red,

MAX(DECODE(color, 'WHITE', TO\_CHAR(my\_data), NULL)) AS white

FROM sec1619

GROUP BY car\_type,

doors

ORDER BY car\_type,

doors;

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

\* EmpDeptDecodeCrosstab.sql

\* CIS-2720

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

SELECT \* FROM hr.employees;

SELECT \* FROM hr.departments ORDER BY department\_name;

-- Number of employees in each depart

-- one row per department

SELECT d.department\_name,

COUNT(\*) AS nbr\_emps

FROM hr.employees e

INNER JOIN hr.departments d

ON e.department\_id = d.department\_id

GROUP BY d.department\_name

ORDER BY d.department\_name;

-- First step - decode function

-- Note that the three IT departments have been merged

SELECT employee\_id,

department\_id,

DECODE(department\_id, 110, 1, 0) AS accounting,

DECODE(department\_id, 10, 1, 0) AS administration,

DECODE(department\_id, 90, 1, 0) AS executive,

DECODE(department\_id, 100, 1, 0) AS finance,

DECODE(department\_id, 40, 1, 0) AS human\_resources,

DECODE(department\_id, 60, 1, 230, 1, 210, 1, 0) AS it,

DECODE(department\_id, 70, 1, 0) AS public\_relations,

DECODE(department\_id, 30, 1, 0) AS purchasing,

DECODE(department\_id, 80, 1, 0) AS sales,

DECODE(department\_id, 50, 1, 0) AS shipping

FROM hr.employees;

-- Cross tab for the same query

-- Note that the three IT departments have been merged

SELECT SUM(DECODE(department\_id, 110, 1, 0)) AS accounting,

SUM(DECODE(department\_id, 10, 1, 0)) AS administration,

SUM(DECODE(department\_id, 90, 1, 0)) AS executive,

SUM(DECODE(department\_id, 100, 1, 0)) AS finance,

SUM(DECODE(department\_id, 40, 1, 0)) AS human\_resources,

SUM(DECODE(department\_id, 60, 1, 230, 1, 210, 1, 0)) AS it,

SUM(DECODE(department\_id, 70, 1, 0)) AS public\_relations,

SUM(DECODE(department\_id, 30, 1, 0)) AS purchasing,

SUM(DECODE(department\_id, 80, 1, 0)) AS sales,

SUM(DECODE(department\_id, 50, 1, 0)) AS shipping

FROM hr.employees;

**18 If-Then-Else, Parameter Queries, and Subqueries**

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

\* SubqueryExamples.sql

\* CIS-2720

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

SELECT \* FROM employees;

SELECT \* FROM dependents;

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

-- HARD CODED SUBQUERY & EQUIVALENT

SELECT \*

FROM dependents

WHERE emp\_id IN (3, 4, 7, 9);

SELECT \*

FROM dependents

WHERE emp\_id = 3

OR emp\_id = 4

OR emp\_id = 7

OR emp\_id = 9;

SELECT \*

FROM dependents

WHERE emp\_id NOT IN (3, 4, 7, 9);

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

-- EQUAL SUBQUERY

-- this is worng qurey

SELECT \*

FROM dependents

WHERE emp\_id =

(SELECT MAX(emp\_id)

FROM employees);

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

-- IN SUBQUERY

SELECT \*

FROM dependents

WHERE emp\_id IN

(SELECT emp\_id

FROM employees

WHERE employee\_nbr BETWEEN 5000 AND 8000);

SELECT \*

FROM dependents

WHERE emp\_id NOT IN

(SELECT emp\_id

FROM employees

WHERE employee\_nbr BETWEEN 5000 AND 8000);

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

-- EXISTS

SELECT \*

FROM dependents d

WHERE EXISTS

(SELECT emp\_id

FROM employees e

WHERE e.emp\_id = d.emp\_id

AND e.last\_name LIKE '%o%');

SELECT \*

FROM dependents d

WHERE NOT EXISTS

(SELECT emp\_id

FROM employees e

WHERE e.emp\_id = d.emp\_id

AND e.last\_name LIKE '%o%');

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

-- CORRELATED SUBQUERY

SELECT \* FROM dogs;

SELECT \* FROM dog\_breeds;

SELECT \*

FROM dogs d

WHERE dog\_weight >

(SELECT avg\_breed\_weight

FROM dog\_breeds b

WHERE b.breed\_id = d.breed\_id);

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

\* SubstitutionVariables.sql

\* CIS-2720

\*

\* See http://www.orafaq.com/node/515 for more details

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

DROP TABLE amp\_test;

CREATE TABLE amp\_test (f01 VARCHAR2(50));

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

\* Temporary Substitution Variable

\* Single & - ""

\*/

INSERT INTO amp\_test VALUES ('&value01');

COMMIT;

SELECT \* FROM amp\_test;

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

\* Single & - used twice

\* Prompts each time \*/

INSERT INTO amp\_test VALUES ('&value02');

INSERT INTO amp\_test VALUES ('&value02');

COMMIT;

SELECT \* FROM amp\_test;

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

\* Permanent Substitution Variable

\* Double & - Twice, then single &

\* Prompts only the first time

\*/

INSERT INTO amp\_test VALUES ('&&value03');

INSERT INTO amp\_test VALUES ('&&value03');

INSERT INTO amp\_test VALUES ('&value03');

COMMIT;

SELECT \* FROM amp\_test;

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

\* Using literal ampersands instead of variables.

\*/

INSERT INTO amp\_test VALUES ('Savings & Loan');

-- ampersand before quote

INSERT INTO amp\_test VALUES ('Savings &' ||' Loan');

-- Assumes characterset w/ASCII characters < 127

INSERT INTO amp\_test VALUES ('A' || CHR(38) || 'P');

set define off

INSERT INTO amp\_test VALUES ('&&value03');

set define on

COMMIT;

SELECT \* FROM amp\_test;

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

\* DEFINE variable

\* can be abbreviated DEF

\*/

DEFINE value04="ABCDEFG"

INSERT INTO amp\_test VALUES ('&value04');

COMMIT;

SELECT \* FROM amp\_test;

-- Shows value of a variable

DEF value01

DEF value02

DEF value03

DEFINE value04

-- All currently defined variables

DEF

-- the same

DEFINE

-- UNDEFINE variable (abbreviated UNDEF)

UNDEFINE value03

DEF value03

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

\* PROMPT and ACCEPT

\*/

-- PROMPT doesn't prompt, it just prints

PROMPT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PROMPT Enter a value to be inserted in the database

ACCEPT value5 VARCHAR2 PROMPT "Enter the value for the transction: "

DEF value5

INSERT INTO amp\_test VALUES ('&value5');

COMMIT;

SELECT \* FROM amp\_test;