

# Database Programming

Final Exam Review

# Objectives

This lesson covers the following objectives:

- Review the key points about case and character manipulation
- Review number, date, conversion, and general functions
- Review conditional expressions
- Review Cartesian product and join operations
- Review non-equijoins, outer joins, self joins, cross joins, natural joins, and join clauses
- Review group functions, group by syntax, and having clauses

## Objectives (cont.)

This lesson covers the following objectives:

- Review single-row and multiple row subqueries
- Review pair-wise and non-pair-wise subqueries
- Review correlated subqueries
- Review DML statements insert, update, delete, merge, and multi-table inserts
- Review DDL statements FLASHBACK TABLE, DROP, and FLASHBACK QUERY

# Purpose

Review is the best preparation for assessment.

Assessment helps you realize how much you've learned and highlights areas in which you may wish to improve.

Reviewing the topics learned to this point will help you be your best during the final exam.

# Syntax Review

This is a review of the syntax.

Ensure that you also review the rules concerning the syntax. These are covered throughout the course.

# Case and Character Manipulation

## Case

```
LOWER(column name|expression)  
UPPER(column name|expression)  
INITCAP(column name|expression)
```

## Character

```
CONCAT(column name|expression, column name|expression)  
SUBSTR(column name|expression,n,m)  
LENGTH(column name|expression)
```

# Case and Character Manipulation (cont.)

## Character (cont.)

```
INSTR(column name|expression, string literal)
LPAD (column name|expression, n, character literal)
RPAD(column name|expression, n, character literal)
TRIM ( [leading | trailing | both] char1 FROM char2)
REPLACE (column name|expression, string to be replaced, replacement
string)
```

# Number Functions

```
ROUND(column|expression,n)  
TRUNC(column|expression,n)  
MOD(column|expression, column|expression)
```



# Date Functions

```
ROUND(column|expression,string)
```

```
TRUNC(column|expression,string)
```

```
MONTHS_BETWEEN(column|expression, column|expression)
```

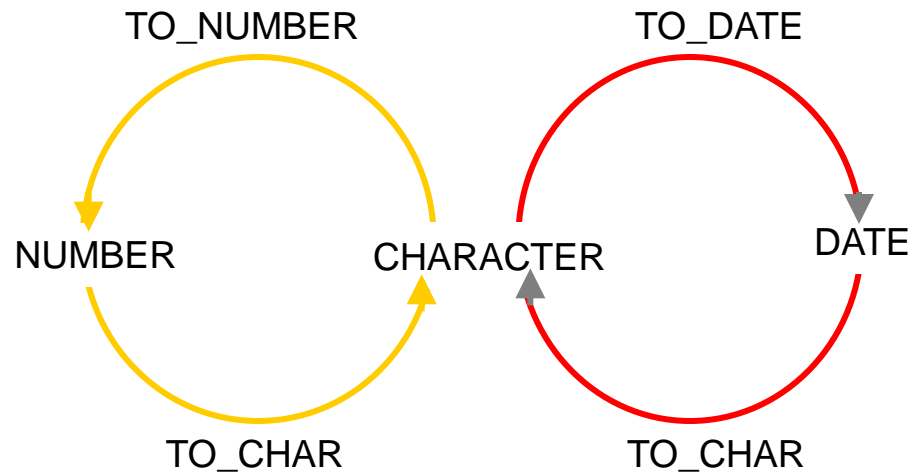
```
ADD_MONTHS(column|expression,n)
```

```
NEXT_DAY(column|expression,'day')
```

```
LAST_DAY(column|expression)
```

# Conversion Functions

```
TO_CHAR(number, 'format model')  
TO_CHAR(date, 'format model')  
TO_NUMBER(character string, 'format model')  
TO_DATE(character string, 'format model')
```



# NULL Functions

```
NVL(column|expression, value)
NVL2(column|expression, column|expression,
      column|expression)
NULLIF(column|expression, column|expression)
COALESCE(column|expression, column|expression,
          column|expression.... column|expression)
```

# Conditional Expressions

## Oracle-specific

```
DECODE(column1|expression, search1, result1  
      [, search2, result2,...,]  
      [, default])
```

## ANSI

```
CASE expr WHEN comparison_expr1 THEN return_expr1  
      [WHEN comparison_expr2 THEN return_expr2  
      WHEN comparison_exprn THEN return_exprn  
      ELSE else_expr]  
END
```

# ANSI SQL Standard Syntax

## Cross Join

```
SELECT last_name, department_name  
FROM employees CROSS JOIN departments;
```

## Natural Join

```
SELECT employee_id, last_name, department_name  
FROM employees NATURAL JOIN departments;
```

# ANSI SQL Standard Syntax (cont.)

## Join .. On

```
SELECT e.employee_id, e.last_name, e.salary, j.grade_level  
FROM employees e JOIN job_grades j  
ON (e.salary BETWEEN j.lowest_sal AND j.highest_sal);
```

# ANSI SQL Standard Syntax (cont.)

## Joins .. Using

```
SELECT employee_id, last_name, department_name  
FROM employees JOIN departments  
USING (department_id);
```

## Join .. On

```
SELECT e.employee_id, e.last_name, d.department_id, d.location_id  
FROM employees e JOIN departments d  
ON (e.department_id = d.department_id);
```

# ANSI SQL Standard Syntax (cont.)

## Outer Joins

### Right Outer Join

```
SELECT e.employee_id, e.last_name,  
       e.department_id, d.department_name  
FROM employees e RIGHT OUTER JOIN departments d  
ON (e.department_id = d.department_id);
```

### Left Outer Join

```
SELECT e.employee_id, e.last_name,  
       e.department_id, d.department_name  
FROM employees e LEFT OUTER JOIN departments d  
ON (e.department_id = d.department_id);
```



# ANSI SQL Standard Syntax (cont.)

## Outer Joins

### Full Outer Join (No comparable Oracle specific Join)

```
SELECT e.employee_id, e.last_name,  
       e.department_id, d.department_name  
FROM employees e FULL OUTER JOIN departments d  
ON (e.department_id = d.department_id);
```

# Group Functions, Group By Syntax and Having Clauses

```
AVG (column | expression)
COUNT (column | expression)
MIN (column | expression)
MAX (column | expression)
SUM (column | expression)
VARIANCE (column | expression)
STDDEV (column | expression)

SELECT column1, AVG (column | expression)
FROM table 1
GROUP BY (ROLLUP | CUBE) (column1 | GROUPING SETS)
HAVING AVG (column | expression)
```

# Single-row and Multiple-row Subqueries

```
SELECT column1..  
FROM table 1  
WHERE column2 =  
                (SELECT column2  
                  FROM table 1  
                  WHERE column 3 = expression)
```

Single row operators: =, >, <, >=, <=, <>

Multiple row operators: IN, ANY, ALL

# Pairwise and Non-Pairwise Subqueries

## Pairwise

```
SELECT column1..  
FROM table 1  
WHERE (column2, column3) = (SELECT column2, column3  
                             FROM table 1  
                             WHERE column 4 = expression)
```

## Non-pairwise

```
SELECT column1..  
FROM table 1  
WHERE column2 = (SELECT column2  
                 FROM table 1  
                 WHERE column 4 = expression)  
AND   column3 = (SELECT column3  
                 FROM table 2  
                 WHERE column 4 = expression)
```

# Correlated Subqueries

```
SELECT o.column1..  
FROM table_1 o  
WHERE o.column2 =  
      (SELECT i.column2  
       FROM table_2 i  
       WHERE i.column1 = o.column1)
```

# Inserting, Updating, and Deleting Data

## Explicit Insert

```
INSERT INTO table (column1, column2...)  
VALUES (value1, value2...) ;
```

## Implicit Insert

```
INSERT INTO table  
VALUES (value1, value2, value3, value4);
```

```
UPDATE table1  
SET column1 = value1,  
    column2 = value2...  
WHERE column1 = value;
```

```
DELETE FROM table1  
WHERE column1 = value;
```

# Inserting, Updating, and Deleting Data (cont.)

## Multi-table Insert

```
conditional_insert_clause  
[ ALL | FIRST ]  
WHEN condition THEN  
    insert_into_clause [ values_clause ]  
WHEN condition THEN  
    insert_into_clause [ values_clause ]  
ELSE insert_into_clause [ values_clause ]
```

# Default Values

```
CREATE TABLE table1 (  
  column1          DATE DEFAULT SYSDATE,...)  
  
INSERT INTO table1  
  (column1,...)  
VALUES  
  (DEFAULT,...);
```



# The Merge Statement

```
MERGE INTO destination-table USING source-table  
ON matching-condition  
WHEN MATCHED THEN UPDATE  
SET .....  
WHEN NOT MATCHED THEN INSERT  
VALUES (.....) ;
```

# Creating Tables

```
CREATE TABLE table  
(column datatype [DEFAULT expression],  
column datatype [DEFAULT expression],  
.....[ ] );
```

```
CREATE TABLE tablename  
[(column, column, ...)]  
AS subquery;
```

# Specifying Data Types

```
NUMBER(p,s)
CHAR
VARCHAR2(n)
DATE
TIMESTAMP
TIMESTAMP WITH TIMEZONE
TIMESTAMP WITH LOCAL TIME ZONE
INTERVAL YEAR TO MONTH
INTERVAL DAY TO SECOND
CLOB
BLOB
RAW
```

# Modifying a Table

```
ALTER TABLE tablename  
ADD (column_name datatype [DEFAULT expression]...)  
  
ALTER TABLE tablename MODIFY (column_name VARCHAR2(30));  
  
ALTER TABLE tablename DROP COLUMN column name;  
  
ALTER TABLE tablename SET UNUSED (column name);  
  
ALTER TABLE tablename DROP UNUSED COLUMNS;
```

# Modifying a Table (cont.)

```
DROP TABLE tablename;

FLASHBACK TABLE tablename TO BEFORE DROP;

SELECT * FROM user_recyclebin;
SELECT versions_starttime "START_DATE",
       versions_endtime   "END_DATE",
       column, column.....
FROM   table
       VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
WHERE  column = value
```

# Column Level Constraints

```
CREATE TABLE table
(col1 datatype CONSTRAINT tab_col1_pk PRIMARY KEY,
col2 datatype CONSTRAINT tab_col2_nn NOT NULL,
col3 datatype CONSTRAINT tab_col3_uk UNIQUE,
col4 datatype CONSTRAINT tab_col4_ck CHECK (col4 > value),
col5 datatype CONSTRAINT tab_col5 REFERENCES table2 (col1));
```

# Table Level Constraints

```
CREATE TABLE table
(col1 datatype,
col2 datatype,
col3 datatype,
col4 datatype,
col5 datatype,
CONSTRAINT tab_col1_pk PRIMARY(col1),
CONSTRAINT tab_col3_uk UNIQUE(col2),
CONSTRAINT tab_col4_ck CHECK (col4 > value),
CONSTRAINT tab1_col5_fk FOREIGN KEY (col5) REFERENCES table2 (col1));
```

# Creating and Managing Views

```
CREATE [OR REPLACE] [FORCE| NOFORCE] VIEW view [(alias [, alias]...)]
AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY [CONSTRAINT constraint]];

DROP VIEW viewname;
```



# Top-n Analysis

```
SELECT ROWNUM as RANK, col1, col2
FROM (SELECT col1, col2 FROM table1
ORDER BY col1)
WHERE ROWNUM <= n;
```

# Inline Views

```
SELECT t1.col1, t2.col2...  
FROM table 1 t1, (SELECT col1, col2..  
                  FROM table2  
                  WHERE ...) t2  
WHERE .....;
```

# Creating Sequences

```
CREATE SEQUENCE sequence
    [INCREMENT BY n]
    [START WITH n]
    [{MAXVALUE n | NOMAXVALUE}]
    [{MINVALUE n | NOMINVALUE}]
    [{CYCLE | NOCYCLE}]
    [{CACHE n | NOCACHE}];

DROP SEQUENCE sequence_name;
```

# Creating Indexes, and Synonyms

```
CREATE INDEX index_name  
ON table_name( column...,column);  
  
DROP INDEX index_name;  
  
CREATE [PUBLIC] SYNONYM synonym  
FOR object;  
  
DROP [PUBLIC] SYNONYM name_of_synonym
```

# Creating and Revoking Object Privileges

```
CREATE USER user  
IDENTIFIED BY password;  
  
ALTER USER user  
IDENTIFIED BY password;  
  
GRANT privilege [, privilege...]  
TO user [, user| role, PUBLIC...];
```

# Creating and Revoking Object Privileges (cont.)

```
CREATE ROLE role_name;

GRANT object_priv [(column_list)]
ON object_name
TO {user|role|PUBLIC}
[WITH GRANT OPTION];

REVOKE {privilege [, privilege...]|ALL}
ON object
FROM {user[, user...]|role|PUBLIC}
[CASCADE CONSTRAINTS];
```

# Summary

In this lesson, you should have reviewed:

- The key points about case and character manipulation
- Number, date, conversion, and general functions
- Conditional expressions
- Cartesian product and join operations
- Non-equijoins, outer joins, self joins, cross joins, natural joins, and join clauses
- Group functions, group by syntax, and having clauses

## Summary (cont.)

In this lesson, you should have reviewed:

- Single-row and multiple row subqueries
- Pair-wise and non-pair-wise subqueries
- Correlated subqueries
- DML statements, insert, update, delete, merge and multi-table inserts
- DDL statements, FLASHBACK TABLE, DROP and FLASHBACK QUERY