Mots réservés SQL vus en première et deuxième année[1].

SELECT

SQL	*Exemple[2]*
<,<=,>,>=,=,!=,<>	SELECT nomColonne(s) FROM nomTable WHERE nomColonne != expression
'ConstanteTexte'	SELECT nomColonne(s), 'ConstanteTexte' FROM nomTable
||,+,-,*,/,(,)	SELECT (nomColonne * expr) / expression FROM nomTable
ALL	SELECT nomColonne(s) FROM nomTable WHERE nomColonne > ALL(sousSelect)
AND/OR	SELECT nomColonne(s) FROM nomTable WHERE condition AND condition
ANY	SELECT nomColonne(s) FROM nomTable WHERE nomColonne >ANY (sousSelect)
AS (alias de colonne)	SELECT nomColonne AS nomAlias FROM nomTable
AS (alias de table)	SELECT nomColonne(s) FROM nomTable AS nomAlias
ASC/DESC	SELECT nomColonne(s) FROM nomTable ORDER BY nomColonne DESC
AVG	SELECT AVG(expression) FROM nomTable
BETWEEN	SELECT nomColonne(s) FROM nomTable WHERE nomCol BETWEEN val1 AND val2
COUNT	SELECT count(*) FROM nomTable
COUNT	SELECT count(nomColonne) FROM nomTable
DISTINCT	SELECT DISTINCT nomColonne(s) FROM nomTable
FROM	SELECT nomColonne(s) FROM nomTable
GROUP BY	SELECT nomColonne(s) FROM nomTable GROUP BY nomColonne(s)

1 vu aux labos.
2 Un exemple parmis d'autres. Ceci n'est évidemment pas exhaustif.

SQL	*Exemple*
HAVING	SELECT nomColonne(s) FROM nomTable GROUP BY nomColonne(s) HAVING condition
IN	SELECT nomColonne(s) FROM nomTable WHERE nomColonne IN (val1,val2,...,valN)
JOIN	SELECT nomColonne(s) FROM nomTable JOIN nomTable ON condition jointure
LEFT/RIGHT	SELECT nomColonne(s) FROM nomTable1 LEFT JOIN nomTable2 ON conditionJointure
LIKE	SELECT nomColonne(s) FROM nomTable WHERE nomColonne LIKE '%A_'
MAX	SELECT MAX(nomColonne) FROM nomTable
MIN	SELECT MIN(nomColonne) FROM nomTable
NOT	SELECT nomColonne(s) FROM nomTable WHERE nomColonne IS NOT NULL AND nomColonne NOT IN(val1,val2)
NULL	SELECT nomColonne(s) FROM nomTable WHERE nomColonne IS NULL
ORDER	SELECT nomColonne(s) FROM nomTable ORDER BY nomColonne(s)
SELECT	SELECT nomColonne(s) FROM nomTable
SELECT *	SELECT * FROM nomTable
SUM	SELECT SUM(nomColonne) FROM nomTable
UPPER/LOWER	SELECT UPPER(nomColonne) FROM nomTable
WHERE	SELECT nomColonne(s) FROM nomTable WHERE condition

INSERT UPDATE DELETE

SQL	*Exemple*
DELETE	DELETE FROM nomTable WHERE condition
INSERT	INSERT INTO nomTable (nomColonne(s)) VALUES (expression(s))
UPDATE	UPDATE nomTable SET nomColonne = expression WHERE condition

DDL

SQL	*Exemple*
CREATE TABLE	CREATE TABLE nomTable (nomColonne type DEFAULT expression CONSTRAINT nomConst contrainte-de-colonne,... ,CONSTRAINT nomConst contrainte-de-table)
PRIMARY KEY	CREATE TABLE nomTable (nomColonne type DEFAULT expression CONSTRAINT pk-nom PRIMARY KEY,...)
UNIQUE	CREATE TABLE nomTable (nomColonne type DEFAULT expression CONSTRAINT uq-nom UNIQUE,...)
REFERENCES	CREATE TABLE nomTable (nomColonne type DEFAULT expression CONSTRAINT rf-nom REFERENCES nomTable (nomColonne(s)) ,...)
FOREIGN KEY	CREATE TABLE nomTable (nomColonne type DEFAULT expression,... ,fk-nom FOREIGN KEY(nomColonne(s)) REFERENCES nomTable (nomColonne(s)))
ON DELETE	CREATE TABLE nomTable (nomColonne type DEFAULT expression CONSTRAINT rf-nom REFERENCES nomTable nomColonne(s) ON DELETE CASCADE,...)
CHECK	CREATE TABLE nomTable (nomColonne type DEFAULT expression CONSTRAINT ck-nom CHECK(condition),...)
(NOT) NULL	CREATE TABLE nomTable (nomColonne type DEFAULT expression CONSTRAINT nn-nom NOT NULL,...)
ALTER TABLE ADD	ALTER TABLE nomTable ADD (nomColonne type CONSTRAINT ...,...)
ALTER TABLE MODIFY	ALTER TABLE nomTable MODIFY (nomColonne type CONSTRAINT ...,...)

SQL	*Exemple*
ADD CONSTRAINT	ALTER TABLE nomTable ADD CONSTRAINT nomConst contrainte-de-table
DROP CONSTRAINT	ALTER TABLE nomTable DROP CONSTRAINT nomConst
ALTER TABLE DROP	ALTER TABLE nomTable DROP (nomColonne(s))
DROP TABLE	DROP TABLE nomTable CASCADE CONSTRAINTS
RENAME	RENAME ancienNomTable TO nouvNomTable
CREATE INDEX	CREATE INDEX nomIndex ON nomTable (nomColonne(s))
DROP INDEX	DROP INDEX nomIndex
CREATE VIEW	CREATE OR REPLACE VIEW nomVue (nomColonne(s)) AS SELECT...
WITH CHECK OPTION	CREATE OR REPLACE VIEW nomVue (nomColonne(s)) AS SELECT... WITH CHECK OPTION
WITH READ ONLY	CREATE OR REPLACE VIEW nomVue (nomColonne(s)) AS SELECT... WITH READ ONLY
DROP VIEW	DROP VIEW nomVue
CREATE SYNONYM	CREATE PUBLIC SYNONYM nomS FOR nom
DROP SYNONYM	DROP SYNONYM nomS
CREATE SEQUENCE	CREATE SEQUENCE nom START WITH n INCREMENT BY m MINVALUE min NO MAXVALUE CYCLE CACHE c
COMMENT TABLE	COMMENT ON TABLE nomTable IS 'blabla'
COMMENT COLUMN	COMMENT ON COLUMN nomTable.nomColonne IS '..'
ALTER USER	ALTER USER nomUtilisateur IDENTIFIED BY passw

DCL

SQL	*Exemple*
GRANT	GRANT privilège ON nomObjet TO nomUtilisateur privilège : SELECT/INSERT/UPDATE/DELETE/EXECUTE/ALL
REVOKE	REVOKE privilège ON nomObjet FROM nomUtilisateur
COMMIT	COMMIT
ROLLBACK	ROLLBACK

E-SQL

SQL	*Exemple*
DECLARE CURSOR	DECLARE nomCurseur CURSOR FOR SELECT ...
OPEN	OPEN nomCurseur
CLOSE	CLOSE nomCurseur
FETCH	FETCH nomCurseur INTO :nomVar,...
CONNECT	CONNECT userid IDENTIFIED BY passw
ROLLBACK	ROLLBACK WORK RELEASE
WHENEVER	WHENEVER SQLERROR GOTO label
SELECT INTO	SELECT nomColonne(s) INTO :nomVar(s) FROM...
BEGIN	BEGIN DECLARE SECTION
END	END DECLARE SECTION

PL/SQL

SQL	*Exemple*

BLOC PL/SQL	DECLARE
	...
	BEGIN
	...
	END;

CREATE PROCEDURE	CREATE OR REPLACE PROCEDURE nomProc (nomparam IN OUT type, ...) AS bloc PL/SQL
CREATE FUNCTION	CREATE OR REPLACE FUNCTION nomFonct (nomparam IN type, ...) RETURN type AS bloc PL/SQL
%TYPE	nomVar nomColonne%TYPE;
%ROWTYPE	nomVar nomTable%ROWTYPE;
IF	IF cond THEN trt1; ELSIF cond THEN trt2; ELSE trt3; END IF;
LOOP	LOOP trt; EXIT WHEN cond; END LOOP;
FOR	FOR var IN min..max LOOP trt; END LOOP;
WHILE	WHILE cond LOOP trt;

SQL	*Exemple*
	END LOOP;
CREATE TRIGGER	CREATE OR REPLACE TRIGGER nomTrigger BEFORE DELETE ON nomTable Bloc PL/SQL
BEFORE/AFTER	CREATE OR REPLACE TRIGGER nomTrigger AFTER INSERT ON nomTable Bloc PL/SQL
INSTEAD OF	CREATE OR REPLACE TRIGGER nomTrigger INSTEAD OF INSERT ON nomTable Bloc PL/SQL
INSERT/DELETE/UPDATE	CREATE OR REPLACE TRIGGER nomTrigger BEFORE UPDATE OF nomColonne(s) ON nomTable Bloc PL/SQL
FOR EACH ROW	CREATE OR REPLACE TRIGGER nomTrigger BEFORE INSERT ON nomTable FOR EACH ROW Bloc PL/SQL
WHEN	CREATE OR REPLACE TRIGGER nomTrigger BEFORE INSERT ON nomTable WHEN condition Bloc PL/SQL
OLD	CREATE OR REPLACE TRIGGER nomTrigger BEFORE UPDATE ON nomColonne(s) ON nomTable FOR EACH ROW WHEN (OLD.nomColonne > NEW.nomColonne) Bloc PL/SQL(avec old et new si besoin)
NEW	:new.nomColonne := valeur;
DBMS_OUTPUT.PUT_LINE	DBMS_OUTPUT.PUT_LINE('blabla');
RETURN	RETURN valRetour;
SERVEUROUTPUT	SET SERVEUROUTPUT ON
USER_ERROR	SELECT * FROM USER_ERROR