Oracle Relational SQL Cheatsheet.

Types

CHAR(n) **CHARACTER(n)**	Fixed length string of character n.
VARCHAR2(n)	Character string of maximum length n, of varying size.
NUMBER	Integers.
NUMBER(p,s)	Numbers of precision p, with s digits af the decimal point.
DATE	Date information.
TIME	Time information.
BLOB	Binary Large Object.
CLOB	Character binary large object.
NCLOB	National character sets.
BFILE	Read only external file.
RAW/LONG RAW	Binary data, used for import and expor

Conversions

to_char(x) **to_number(x)** **to_date(x)**	Converts it's argument to the appropia type.
to_multi_byte() **to_single_byte()**	Converts between single & multi byte international strings.
chartorowid(x) **rowidtochar(x)**	Converts character strings to ROWID's back.
hextoraw(x) **rawtohex(x)**	Converts between hex and **RAW** binar format (see types).

Operators

=,>,<,>=, <=,!=,<>	Usual comparisons. != & <> & ^= are negative equality tests.
AND OR NOT	Boolean operations.
BETWEEN	SELECT emp_id, name, dept_no FROM epmployee WHEREemp_id BEWTEEN1 4;
IN	**SELECT** emp_id, name, dept_no **FROM** epmployee **WHERE** emp_id **IN** (1,2,3,
LIKE	Regexp match. **%** = n characters, **_** = character, **** escapes.

Constraints

NULL/NOT NULL	Allow/don't allow missing values.
[**CONSTRAINT** <constraint name> **UNIQUE (**<column_name>,..**)**]	For candidate keys - alternatives to primary key
PRIMARY KEY	This is the key field for look

Creating and Deleting Tables

CREATE TABLE <table name> **(**<column definition list>, (<column name>**));**	**CREATE TABLE** part **(**part_number CHAR(4), part_name VARCHAR(25), **PRIMARY_KEY(**part_number**));**
CREATE TABLE <table name> **(** <colum_definition [<constraint >], .. **)**;	**CREATE TABLE** department **(**department_number CHAR(4) **CONSTRAINT** prim_dept **PRIMARY_KEY**, department_name **VARCHAR2(**10**)**
DROP TABLE <table_name>;	Delete table from database.

Changing Tables.

ALTER TABLE<table_name> **ADD(CONSTRAINT** <constriant_name> **PRIMARY_KEY (**<column_names>**));**	Creates a key const column.
ALTER TABLE <table_name> **ADD**(<colmn_definition>**)**;	**ALTER TA** employee **ADD** (departme VARCHAR
ALTER TABLE <table_name> **ADD** (**CONSTRAINT** <constraint_name> **FOREIGN_KEY(**<colum_name>**)** **REFERENECES** foriegn_table_name**(**<foreign_column_name>**)** [**ON DELETE CASCADE**]**)**;	Creates a constraint column, m a column table. Opt **DELETE C** maintains integrity b rows in ta row in for is deleted.
ALTER TABLE **DISABLE CONSTRAINT** name;	Relax con
ALTER_TABLE **DROP CONSTRAINT**<constraint_name>;	Delete cor forever.

Modifying and deleting rows

INSERT INTO<table_name> **(**<colum_name, ..>**)** **VALUES (**<value, ..>**);**	INSERT INTO employee (employee_number, employee_nam salary) VALUES ('7092', 'FORD', 175,66);
UPDATE <table_name> **SET** <column> **=** <value>, ..	**UPDATE** wine_list **SET** note **=** ''Ideal as an aperitif' **WHERE** wine_name **=** 'Ch.Haut-Ria

[**CONSTRAINT** <constraint_name> **CHECK (**condition**)**];	Verification/validation.
FOREIGN KEY	This is an index to another table.

Single Valued Functions

lpad(<string>,<width>, [<char>]**); rpad(**<string>, <width>,[<char>]**);**	Pad a string to the right or left with the given width with the given char.
lower(<string>**);** **uppper(**<string>**);** **initcap(**<string>**);**	Uppercase, lowercase, or inital upcase the string.
length(<string>**);**	Returns length, in chars of the string.
substr(<string>,<start>, <end>**);**	Returns a substring from start index, to end index.
abs(<number>**)** **sign(**<number>**)**	Absolute value and sign numb
ceil(<number>**)** **floor(**<number>**)**	Ceiling and floor: Highest and lowest integer with smallest difference from float.
mod(<number0>, <number1>**)** **round(**<number0>, <number1>**)** **trunc(**<number0>, <<number1>)	Remainder of x / y; Round x t decimal places. Truncate x to decimal places
sqrt(<number>**)**	Square root.
greatest(<experession>,..**)** **least(**<expression>,...**)**	Largest and smallest from a lis dates, numbers or strings.
vsize(<expression>**)**	The storage size in bytes for x
sysdate()	Current system date
add_months(<date>, <integer>**)**	Add given number of month to dates;
last_day(<date>**)**	Return the last day of the mor
months_between(<date0>, <date1>**)**	Return the number of months betwwen two dates/
new_time(<date>, <current_timezone>, <other_timezone>**)**	Convert date from one timezo to another.
nvl(<column>,<value>**)**	Substitute <value> for NULL i the column.
soundex(x**)**	Return soundex string for fuzz matching.
decode(<column>,<value>, <return>,<value>, <return>...**)**	For every instance of <value> column return the matching <return> value. A bit like a case/switch.

WHERE <condition>;	
DELETE FROM <table_name> [**WHERE** <conditon>]	**DELETE FROM** members **WHERE** name **LIKE** 'Sharon%';

Querying with Select.

Project and Join. **SELECT** <columns> **FROM** <table> **WHERE** <criterion>;	**SELECT** emp_table.emp_id, emp_table.dept_no, dept_table.desc **FROM** employee_db.emp_table, employee_db.dept_table **WHERE** emp_table.dept_no = dept_table.dept_no;
Sorting. **SELECT** .<clauses> **ORDER BY** <column [DESC|ASC],..>;	**SELECT** * **FROM** emp **ORDER BY** empid **DESC**; -or- **SELE** empid, lastname **FROM** emp **ORDER BY** 2;
Grouping. **SELECT** <select_clauses> **GROUP BY** <column [DESC|ASC]..> **HAVING** <criterion>;	**SELECT** dept, AVG(salary) **FROM** emp **GROUP BY** dept **HAVING** avg(salary)>80000 **ORDER BY** avg(salary) **DESC**;
Column concatenation - formatting. **SELECT** <column>**||** <string>**||**<column> <column_alias>... ...;	**SELECT** firstname||','||lastname full_name **FROM** team;
On the fly calculations. **SELECT** <expression> **FROM DUAL**;	**SELECT** 7 * 9 **FROM DUAL**;
Column aliasing. **SELECT** <column> **AS** <alias_column>..;	**SELECT** name, NVL(spouse,'unmarr **AS** spouse **FROM** emp_db,emp_table;
Subqueries. **SELECT** ... **WHERE** column = (<subquery>);,	**SELECT** empid, dept, salary **FROM** emp **WHERE** dept = (**SELECT** dept **FROM** emp **WHERE** empid = 78483);

Group functions.

avg() **stddev**() **variance**()	Average of all numbers in column. Standard deviation f all numbers in column. Variance of all numbers in column.
sum(x) **count**(x)	Sum total of all numbers in the co Toal number of items in a culumn.
max(x) **min**(x)	Maximum value found in a column. Minimum value found in column.