### Oracle Relational SQL Cheatsheet

#### Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR(n)</td>
<td>Fixed length string of character n.</td>
</tr>
<tr>
<td>CHARACTER(n)</td>
<td>Character string of maximum length n, of varying size.</td>
</tr>
<tr>
<td>VARCHAR2(n)</td>
<td>Character string of maximum length n, of varying size.</td>
</tr>
<tr>
<td>NUMBER</td>
<td>Integers.</td>
</tr>
<tr>
<td>NUMBER(p,s)</td>
<td>Numbers of precision p, with s digits after the decimal point.</td>
</tr>
<tr>
<td>DATE</td>
<td>Date information.</td>
</tr>
<tr>
<td>TIME</td>
<td>Time information.</td>
</tr>
<tr>
<td>BLOB</td>
<td>Binary Large Object.</td>
</tr>
<tr>
<td>CLOB</td>
<td>Character binary large object.</td>
</tr>
<tr>
<td>NCLOB</td>
<td>National character sets.</td>
</tr>
<tr>
<td>BFILE</td>
<td>Read only external file.</td>
</tr>
<tr>
<td>RAW/LONG RAW</td>
<td>Binary data, used for import and export.</td>
</tr>
</tbody>
</table>

#### Conversions

- `to_char(x)` Converts it's argument to the appropriate type.
- `to_number(x)` Converts between single & multi byte international strings.
- `to_multi_byte()` Converts character strings to ROWID's back.
- `hextoraw(x)` Converts between hex and RAW binary format (see types).
- `rowidtochar(x)` Converts character strings to ROWID's back.

#### Operators

- `=, >=, <=, !=, <>` Usual comparisons. `!= & <> & ^=` are negative equality tests.
- `AND, OR, NOT` Boolean operations.
- `BETWEEN` `SELECT emp_id, name, dept_no FROM employee WHERE emp_id BETWEEN 1 AND 4;`
- `IN` `SELECT emp_id, name, dept_no FROM employee WHERE emp_id IN (1,2,3,4);`
- `LIKE` Regex match. `% = n characters, _ = character, \ = escapes.

#### Constraints

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

#### Creating Tables

```
CREATE TABLE <table name> (<column definition list>, <column name>);
```

#### Creating and Deleting Tables

- `CREATE TABLE part` (part_number CHAR(4), part_name VARCHAR(25), PRIMARY_KEY(part_number));
- `CREATE TABLE department` (department_number CHAR(4), PRIMARY_KEY(department_name VARCHAR2(10));
- `DROP TABLE <table_name>;` Delete table from database.

#### Changing Tables

- `ALTER TABLE <table_name> ADD CONSTRAINT <constraint_name> PRIMARY_KEY (<column_names>);` Creates a primary key constraint.
- `ALTER TABLE <table_name> ADD <column_definition>;` Adds a new column.
- `ALTER TABLE <table_name> ADD CONSTRAINT <constraint_name> FOREIGN_KEY(<column_name>) REFERENCES foreign_table_name(<foreign_column_name>) [ON DELETE CASCADE];` Creates a foreign key constraint.
- `ALTER TABLE <table_name> DROP CONSTRAINT <constraint_name>;` Deletes a constraint.
- `ALTER TABLE <table_name> DISABLE CONSTRAINT name;` Relaxes a constraint.
- `ALTER TABLE <table_name> DROP CONSTRAINT <constraint_name>;` Deletes a constraint.

#### Modifying and Deleting rows

- `INSERT INTO <table_name> (<column_name>,..) VALUES (<value,..>);` Inserts new rows.
- `UPDATE <table_name> SET <column> = <value>, ..` Updates existing rows.
- `DELETE FROM <table_name> WHERE <condition>;` Deletes rows.

---

[1 of 2] 12/11/12 10:49 AM
**Single Valued Functions**

- **lpad**(<string>,<width>,<char>);: Pad a string to the right or left with the given width with the given char.
- **lower**(<string>): Uppercase, lowercase, or initial uppercase the string.
- **initcap**(<string>): Returns length, in chars of the string.
- **substr**(<string>,<start>,<end>): Returns a substring from start index, to end index.
- **abs**(<number>): Absolute value and sign number.
- **ceil**(<number>): Ceiling and floor: Highest and lowest integer with smallest difference from float.
- **mod**(<number0>,<number1>): Remainder of x / y; Round x to y decimal places. Truncate x to y decimal places.
- **round**(<number0>,<number1>): Square root.
- **sysdate()**: Current system date.
- **add_months**(<date>,<integer>): Add given number of month to dates.
- **last_day**(<date>): Return the last day of the month.
- **months_between**(<date0>,<date1>): Return the number of months between two dates.
- **new_time**(<date>,<current_timezone>,<other_timezone>): Convert date from one timezone to another.
- **nv1**(<column>,<value>): Substitute <value> for NULL in the column.
- **soundex**(<x>): Return soundex string for fuzzy 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**

- **DELETE FROM**<table_name> [WHERE <condition>;]
- **DELETE FROM**<table_name> [WHERE <condition>;]

**Querying with Select.**

- **SELECT emp_table.emp_id, emp_table.dept_no, dept_table.description FROM employee_db.emp_table, employee_db.dept_table WHERE emp_table.dept_no = dept_table.dept_no;**
- **DELETE FROM emp ORDER BY empid DESC; -or- SELECT empid, lastname FROM emp ORDER BY 2;**
- **SELECT dept, AVG(salary) FROM emp GROUP BY dept**
- **HAVING avg(salary)>80000 ORDER BY avg(salary) DESC;**
- **SELECT firstname||','||lastname FROM team;**
- **SELECT 7 * 9 FROM DUAL;**

**Group functions.**

- **SELECT name, NVL(spouse,'unmarried') AS spouse FROM emp_db,emp_table;**
- **WHERE dept = (SELECT dept FROM emp WHERE empid = 78483);**
- **GROUP BY dept**
- **AVG(salary)**
- **Standard deviation**
- **Variance**
- **Sum**
- **Count**
- **Max**
- **Min**

**Column aliasing.**

- **SELECT <column>||'|'|<column_alias>...**
- **On the fly calculations.**
- **SELECT <column> FROM DUAL;**
- **SELECT name, NVL(spouse,'unmarried') AS spouse FROM emp_db,emp_table;**
- **WHERE dept = (SELECT dept FROM emp WHERE empid = 78483);**