SELECT

$db_select($table, $alias = NULL, $options = array())
$table Database table to select from.
$alias Table alias.
return New query object.

$query = db_select('users', 'u')
->fields('u', array('uid', 'name'))
$result = $query->execute();

->distinct($distinct = TRUE)
$distinct Flag indicating DISTINCT query.
return The called query object.

->addField($table_alias, $field, $alias = NULL)
$table_alias Alias of the table the field belongs to.
$field Field name.
$alias Field alias.
return Unique field alias.

->range($start = NULL, $length = NULL)
$start First record of the result set.
$length Max number of records.
return The called query object.

JOIN

->extend('PagerDefault')
return New pager extender object.

->extend('PagerDefault')->limit($count)
$count Number of items per page.
return The called query object.

SORTABLE TABLE

->extend('TableSort')
return Table extender object.
return The called query object.

$db_or()->condition()->condition()
Condition of OR-ed conditions.

$db_and()->condition()->condition()
Condition of AND-ed conditions.

->isNull($field)
->isNotNull($field)
$field The field to check.
return The called query object.

->exists(SelectQueryInterface $select);
->notExists(SelectQueryInterface $select);
$select The query to check.
return The called query object.

JOIN

->join($table, $alias = NULL, $condition = NULL, $arguments = array())
$table The table to join with.
$alias Table alias.
$condition Join conditions.
$arguments Assoc array of placeholders and placeholder values.
return Unique table alias.

$snippet Where clause (with placeholders)
$sargs Assoc array of placeholders and placeholder values.
return The called query object.

$db_select($table, $alias = NULL, $options = array())
$table Database table to select from.
$alias Table alias.
return New query object.

$distinct = TRUE
$distinct Flag indicating DISTINCT query.
return The called query object.

->distinct($distinct = TRUE)
$distinct Flag indicating DISTINCT query.
return The called query object.

->fields($table_alias, $fields = array())
$table_alias Alias of the table the fields belongs to.
$fields Array of field names.
return The called query object.

JOIN

->innerJoin($node, 'n', 'n.uid = u.uid')
$nuid = u.uid';
$n = 'n.title';
$snippet Where clause (with placeholders)
$sargs Assoc array of placeholders and placeholder values.
return The called query object.

JOIN

->join($table, $alias = NULL, $condition = NULL, $arguments = array())
$table The table to join with.
$alias Table alias.
$condition Join conditions.
$arguments Assoc array of placeholders and placeholder values.
return Unique table alias.

$expression Expression string.
$alias Expression alias.
$arguments Assoc array of placeholders and placeholder values.
return Unique expression alias.

JOIN

->addExpression($expression, $alias = NULL, $arguments = array())
$expression Expression string.
$alias Expression alias.
$arguments Assoc array of placeholders and placeholder values.
return Unique expression alias.

JOIN

->isNull($field)
->isNotNull($field)
$field The field to check.
return The called query object.

JOIN

->exists(SelectQueryInterface $select);
->notExists(SelectQueryInterface $select);
$select The query to check.
return The called query object.

JOIN

->condition($field, $value = NULL, $operator = NULL)
$field The field to check or the result of a logic operation (or, and, xor)
$value The value to test against.
$operator Default: '=' or 'IN'.
Supported: =, <, >, >=, <=, IN, NOT IN, LIKE, BETWEEN, IS NULL, IS NOT NULL
return The called query object.

JOIN

->where($snippet, $args = array())
$snippet Where clause (with placeholders)
$args Assoc array of placeholders and placeholder values.
return The called query object.

JOIN

->groupBy($field)
$field The field to group by.
return The called query object.

JOIN

->orderBy($field, $direction = 'ASC')
$field The field to order by.
$direction 'ASC' or 'DESC'.
return The called query object.

JOIN

->orderRandom()
return The called query object.

JOIN

->union(SelectQueryInterface $query, $type = '')
$query Query to union.
$type Type of union.
return New resulting query object.

JOIN

->db_or()->condition()->condition()
Condition of OR-ed conditions.

JOIN

->db_and()->condition()->condition()
Condition of AND-ed conditions.

JOIN

->isNull($field)
->isNotNull($field)
$field The field to check.
return The called query object.

JOIN

->exists(SelectQueryInterface $select);
->notExists(SelectQueryInterface $select);
$select The query to check.
return The called query object.

JOIN

->join($table, $alias = NULL, $condition = NULL, $arguments = array())
$table The table to join with.
$alias Table alias.
$condition Join conditions.
$arguments Assoc array of placeholders and placeholder values.
return Unique table alias.

JOIN

->addExpression($expression, $alias = NULL, $arguments = array())
$expression Expression string.
$alias Expression alias.
$arguments Assoc array of placeholders and placeholder values.
return Unique expression alias.

JOIN

->isNull($field)
->isNotNull($field)
$field The field to check.
return The called query object.

JOIN

->exists(SelectQueryInterface $select);
->notExists(SelectQueryInterface $select);
$select The query to check.
return The called query object.

JOIN

->condition($field, $value = NULL, $operator = NULL)
$field The field to check or the result of a logic operation (or, and, xor)
$value The value to test against.
$operator Default: '=' or 'IN'.
Supported: =, <, >, >=, <=, IN, NOT IN, LIKE, BETWEEN, IS NULL, IS NOT NULL
return The called query object.

JOIN

->where($snippet, $args = array())
$snippet Where clause (with placeholders)
$args Assoc array of placeholders and placeholder values.
return The called query object.

JOIN

->groupBy($field)
$field The field to group by.
return The called query object.

JOIN

->orderBy($field, $direction = 'ASC')
$field The field to order by.
$direction 'ASC' or 'DESC'.
return The called query object.
RESULTS

->execute($args = array(), $options = array())
return The called query object.

->fetch($mode = NULL, $cursor_orientation = NULL, $cursor_offset = NULL)
$mode Fetch mode.
return Result type specified by $mode.

->fetchObject($class_name = NULL, $constructor_args = array())
$class_name Class type to be returned.
Default: stdClass
return Object of one record.

->fetchAssoc()
return Associative array of one record.

->fetchAllAssoc($key, $fetch = NULL)
$key Field name of the array key
$fetch Fetch mode
(PDO::FETCH_ASSOC, PDO::FETCH_NUM, or PDO::FETCH_BOTH).
return Associative array of data objects.

->fetchAll($mode = NULL, $column_index = NULL, $constructor_arguments = array())
$mode Fetch mode. See above.
return Array of data objects. Depending on fetch mode.

->fetchField($index = 0)
$index Numeric index of the column.
return A single field.

->fetchAllFielded($key_index = 0, $value_index = 1)
$key_index Numeric index of the array key.
$value_index Numeric index of the array value.
return Associative array of all records.

INSERT

$db_insert($table, $options = array())
$table Database table to insert into.
return New query object.

->fields(array('title' => 'Example', 'uid' => 1, 'created' => REQUEST_TIME))
->execute();

->values(array $values)
$values Assoc array of values to insert.
return The called query object.

UPDATE

$db_update($table, $options = array())
$table Database table to update.
return New query object.

->condition('created', REQUEST_TIME - 3600, '>=')
->execute();

DELETE

$db_delete($table, $options = array())
$table Database table to delete from.
return New query object.

->condition('nid', 5)
->execute();

TRUNCATE

$db_truncate($table, $options = array())
$table Database table to remove.
return New query object.

MERGE

$db_merge($table, $options = array())
$table Database table to merge into
return New query object.

->key(array('name' => $name))
->fields(array('weight' => $weight, ))
->execute();

->key(array $fields, $values = array())
$fields Array of fields to match or set.
Or associative array of fields and values.
$values Values to set.
return The called query object.

UPDATE

$db_merge('role')

CALLABLE

$db_query($query, $args = array(), $options = array())
Note: Access control is not supported! Query may not be compatible with other database types.

settings.php

Single database configuration example:
$dbases['default']['default'] = array('driver' => 'mysql', 'database' => 'databasename', 'username' => 'username', 'password' => 'password', 'host' => 'localhost', 'prefix' => '', 'collation' => 'utf8_general_ci', );

DEBUGGING

print($query->__toString());

DOCUMENTATION

Database API on drupal.org:
http://drupal.org/developing/api/database

www.wizzlern.nl/drupal/cheat-sheets