Oracle SQL Cheat Sheet

DATABASE STAR

www.databasestar.com

SELECT Query

```
SELECT col1, col2
FROM table
JOIN table2 ON table1.col = table2.col
WHERE condition
GROUP BY column_name
HAVING condition
ORDER BY col1 ASC|DESC;
```

SELECT Keywords

DISTINCT: Removes duplicate results	`SELECT DISTINCT product_name FROM product;`
BETWEEN: Matches a value between two other values (inclusive)	`SELECT product_name FROM product WHERE price BETWEEN 50 AND 100;`
IN: Matches to any of the values in a list	`SELECT product_name FROM product WHERE category IN ('Electronics', 'Furniture');`
LIKE: Performs wildcard matches using _ or %	`SELECT product_name FROM product WHERE product_name LIKE '%Desk%';`

Joins

```
SELECT t1.*, t2.*
FROM t1
join_type t2 ON t1.col = t2.col;
```

Table 1

| A |
| B |
| C |

Table 2

| A |
| B |
| D |

INNER JOIN: show all matching records in both tables.

| A | A |
| B | B |

LEFT JOIN: show all records from left table, and any matching records from right table.

A	A
B	B
C	

RIGHT JOIN: show all records from right table, and any matching records from left table.

A	A
B	B
	D

FULL JOIN: show all records from both tables, whether there is a match or not.

A	A
B	B
C	
	D

CASE Statement

Simple Case	`CASE name`
	`    WHEN 'John' THEN 'Name John'`
	`    WHEN 'Steve' THEN 'Name Steve'`
	`    ELSE 'Unknown'`
	`END`

Searched Case	`CASE`
	`    WHEN name='John' THEN 'Name John'`
	`    WHEN name='Steve' THEN 'Name Steve'`
	`    ELSE 'Unknown'`
	`END`

Common Table Expression

```
WITH queryname AS (
    SELECT col1, col2
    FROM firsttable)
SELECT col1, col2..
FROM queryname...;
```

Modifying Data

Insert	`INSERT INTO tablename`
	`(col1, col2...)`
	`VALUES (val1, val2);`

Insert from a Table	`INSERT INTO tablename`
	`(col1, col2...)`
	`SELECT col1, col2...`

Insert Multiple Rows	`INSERT`
	`INTO tablename (col1, col2)`
	`VALUES (valA1, valB1)`
	`INTO tablename (col1, col2)`
	`VALUES (valA2, valB2)`
	`SELECT * FROM dual;`

Update	`UPDATE tablename`
	`SET col1 = val1`
	`WHERE condition;`

Update with a Join	`UPDATE t`
	`SET col1 = val1`
	`FROM tablename t`
	`INNER JOIN table x`
	`ON t.id = x.tid`
	`WHERE condition;`

Delete	`DELETE FROM tablename` `WHERE condition;`

Indexes

Create Index	`CREATE INDEX indexname` `ON tablename (cols);`
Drop Index	`DROP INDEX indexname;`

Set Operators

UNION: Shows unique rows from two result sets.

UNION ALL: Shows all rows from two result sets.

INTERSECT: Shows rows that exist in both result sets.

EXCEPT: Shows rows that exist in the first result set but not the second.

Aggregate Functions

- SUM: Finds a total of the numbers provided
- COUNT: Finds the number of records
- AVG: Finds the average of the numbers provided
- MIN: Finds the lowest of the numbers provided
- MAX: Finds the highest of the numbers provided

Common Functions

- LENGTH(string): Returns the length of the provided string
- INSTR(string, substring, [start_position], [occurrence]): Returns the position of the substring within the specified string.
- TO_CHAR(input_value, [fmt_mask], [nls_param]): Converts a date or a number to a string
- TO_DATE(charvalue, [fmt_mask], [nls_date_lang]): Converts a string to a date value.
- TO_NUMBER(input_value, [fmt_mask], [nls_param]): Converts a string value to a number.
- ADD_MONTHS(input_date, num_months): Adds a number of months to a specified date.
- SYSDATE: Returns the current date, including time.
- CEIL(input_val): Returns the smallest integer greater than the provided number.
- FLOOR(input_val): Returns the largest integer less than the provided number.
- ROUND(input_val, round_to): Rounds a number to a specified number of decimal places.
- TRUNC(input_value, dec_or_fmt): Truncates a number or date to a number of decimals or format.
- REPLACE(whole_string, string_to_replace, [replacement_string]): Replaces one string inside the whole string with another string.
- SUBSTR(string, start_position, [length]): Returns part of a value, based on a position and length.

Create Table

Create Table	`CREATE TABLE tablename (`
	`    column_name data_type`
	`);`

Create Table with Constraints

```
CREATE TABLE tablename (
    column_name data_type NOT NULL,
    CONSTRAINT pkname PRIMARY KEY (col),
    CONSTRAINT fkname FOREIGN KEY (col)
REFERENCES other_table(col_in_other_table),
    CONSTRAINT ucname UNIQUE (col),
    CONSTRAINT ckname CHECK (conditions)
);
```

Create Temporary Table	`CREATE GLOBAL TEMPORARY TABLE`
	`tablename (`
	`    colname datatype`
	`) ON COMMIT DELETE ROWS;`

Drop Table	`DROP TABLE tablename;`

Alter Table

Add Column	`ALTER TABLE tablename` `ADD columnname datatype;`
Drop Column	`ALTER TABLE tablename` `DROP COLUMN columnname;`
Modify Column	`ALTER TABLE tablename MODIFY` `columnname newdatatype;`
Rename Column	`ALTER TABLE tablename RENAME COLUMN` `currentname TO newname;`

Add Constraint	`ALTER TABLE tablename ADD`
	`CONSTRAINT constraintname`
	`constrainttype (columns);`

Drop Constraint	`ALTER TABLE tablename DROP` `constraint_type constraintname;`

Rename Table	`sp_rename`
	`'old_table_name',`
	`'new_table_name';`

Window/Analytic Functions

```
function_name ( arguments ) OVER (
[query_partition_clause]
[ORDER BY order_by_clause
[windowing_clause] ] )
```

Example using RANK, showing the student details and their rank according to the fees_paid, grouped by gender:

```
SELECT
student_id, first_name, last_name, gender, fees_paid,
RANK() OVER (
  PARTITION BY gender ORDER BY fees_paid
) AS rank_val
FROM student;
```

Subqueries

Single Row	`SELECT id, last_name, salary`
	`FROM employee`
	`WHERE salary = (`
	`    SELECT MAX(salary)`
	`    FROM employee`
	`);`

Multi Row	`SELECT id, last_name, salary`
	`FROM employee`
	`WHERE salary IN (`
	`    SELECT salary`
	`    FROM employee`
	`    WHERE last_name LIKE 'C%'`
	`);`

SQL Server Cheat Sheet

SELECT Query

```
SELECT col1, col2
FROM table
JOIN table2 ON table1.col = table2.col
WHERE condition
GROUP BY column_name
HAVING condition
ORDER BY col1 ASC|DESC;
```

SELECT Keywords

DISTINCT: Removes duplicate results	`SELECT DISTINCT product_name FROM product;`
BETWEEN: Matches a value between two other values (inclusive)	`SELECT product_name FROM product WHERE price BETWEEN 50 AND 100;`
IN: Matches to any of the values in a list	`SELECT product_name FROM product WHERE category IN ('Electronics', 'Furniture');`
LIKE: Performs wildcard matches using _ or %	`SELECT product_name FROM product WHERE product_name LIKE '%Desk%';`

Joins

```
SELECT t1.*, t2.*
FROM t1
join_type t2 ON t1.col = t2.col;
```

Table 1

A
B
C

Table 2

A
B
D

INNER JOIN: show all matching records in both tables.

A	A
B	B

LEFT JOIN: show all records from left table, and any matching records from right table.

A	A
B	B
C	

RIGHT JOIN: show all records from right table, and any matching records from left table.

A	A
B	B
	D

FULL JOIN: show all records from both tables, whether there is a match or not.

A	A
B	B
C	
	D

CASE Statement

Simple Case	`CASE name`
	`    WHEN 'John' THEN 'Name John'`
	`    WHEN 'Steve' THEN 'Name Steve'`
	`    ELSE 'Unknown'`
	`END`

Searched Case	`CASE`
	`    WHEN name='John' THEN 'Name John'`
	`    WHEN name='Steve' THEN 'Name Steve'`
	`    ELSE 'Unknown'`
	`END`

Common Table Expression

```
WITH queryname (col1, col2...) AS (
  SELECT col1, col2
  FROM firsttable)
SELECT col1, col2..
FROM queryname...;
```
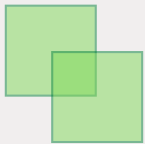
Modifying Data

Insert	`INSERT INTO tablename`
	`(col1, col2...)`
	`VALUES (val1, val2);`

Insert from a Table	`INSERT INTO tablename`
	`(col1, col2...)`
	`SELECT col1, col2...`

Insert Multiple Rows	`INSERT INTO tablename`
	`(col1, col2...) VALUES`
	`(valA1, valB1),`
	`(valA2, valB2),`
	`(valA3, valB3);`

Update	`UPDATE tablename`
	`SET col1 = val1`
	`WHERE condition;`

Update with a Join	`UPDATE t`
	`SET col1 = val1`
	`FROM tablename t`
	`INNER JOIN table x`
	`ON t.id = x.tid`
	`WHERE condition;`

Delete	`DELETE FROM tablename` `WHERE condition;`

Indexes
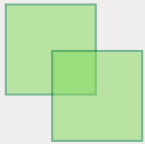
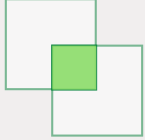|	
Create Index	`CREATE INDEX indexname` `ON tablename (cols);`
Drop Index	`DROP INDEX indexname;`

Set Operators

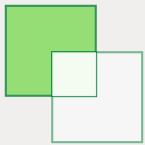UNION: Shows unique rows from two result sets.

UNION ALL: Shows all rows from two result sets.

INTERSECT: Shows rows that exist in both result sets.

MINUS: Shows rows that exist in the first result set but not the second.

Aggregate Functions

- SUM: Finds a total of the numbers provided
- COUNT: Finds the number of records
- AVG: Finds the average of the numbers provided
- MIN: Finds the lowest of the numbers provided
- MAX: Finds the highest of the numbers provided

Common Functions

- LEN(string): Returns the length of the provided string
- CHARINDEX(string, substring, [start_position], [occurrence]): Returns the position of the substring within the specified string.
- CAST(expression AS type [(length)]): Converts an expression to another data type.
- GETDATE: Returns the current date, including time.
- CEILING(input_val): Returns the smallest integer greater than the provided number.
- FLOOR(input_val): Returns the largest integer less than the provided number.
- ROUND(input_val, round_to, operation): Rounds a number to a specified number of decimal places.
- REPLACE(whole_string, string_to_replace, replacement_string): Replaces one string inside the whole string with another string.
- SUBSTRING(string, start_position, [length]): Returns part of a value, based on a position and length.

Create Table

Create Table	`CREATE TABLE tablename (`
	`    column_name data_type`
	`);`

Create Table with Constraints

```
CREATE TABLE tablename (
  column_name data_type NOT NULL,
  CONSTRAINT pkname PRIMARY KEY (col),
  CONSTRAINT fkname FOREIGN KEY (col)
REFERENCES other_table(col_in_other_table),
  CONSTRAINT ucname UNIQUE (col),
  CONSTRAINT ckname CHECK (conditions)
);
```

Create Temporary Table	`SELECT cols`
	`INTO #tablename`
	`FROM table;`

Drop Table	`DROP TABLE tablename;`

Alter Table

Add Column	`ALTER TABLE tablename` `ADD columnname datatype;`
Drop Column	`ALTER TABLE tablename` `DROP COLUMN columnname;`
Modify Column	`ALTER TABLE tablename ALTER COLUMN` `columnname newdatatype;`

Rename Column	`sp_rename`
	`'table_name.old_column_name',`
	`'new_column_name', 'COLUMN';`

Add Constraint	`ALTER TABLE tablename ADD`
	`CONSTRAINT constraintname`
	`constrainttype (columns);`

Drop Constraint	`ALTER TABLE tablename` `DROP CONSTRAINT constraintname;`
Rename Table	`ALTER TABLE tablename` `RENAME TO newtablename;`

Window/Analytic Functions

```
function_name ( arguments ) OVER (
[query_partition_clause]
[ORDER BY order_by_clause
[windowing_clause] ] )
```

Example using RANK, showing the student details and their rank according to the fees_paid, grouped by gender:

```
SELECT
student_id, first_name, last_name, gender, fees_paid,
RANK() OVER (
  PARTITION BY gender ORDER BY fees_paid
) AS rank_val
FROM student;
```

Subqueries

Single Row	`SELECT id, last_name, salary`
	`FROM employee`
	`WHERE salary = (`
	`    SELECT MAX(salary)`
	`    FROM employee`
	`);`

Multi Row	`SELECT id, last_name, salary`
	`FROM employee`
	`WHERE salary IN (`
	`    SELECT salary`
	`    FROM employee`
	`    WHERE last_name LIKE 'C%'`
	`);`

MySQL Cheat Sheet

SELECT Query

```
SELECT col1, col2
FROM table
JOIN table2 ON table1.col = table2.col
WHERE condition
GROUP BY column_name
HAVING condition
ORDER BY col1 ASC|DESC;
```

SELECT Keywords

DISTINCT: Removes duplicate results	`SELECT DISTINCT product_name FROM product;`
BETWEEN: Matches a value between two other values (inclusive)	`SELECT product_name FROM product WHERE price BETWEEN 50 AND 100;`
IN: Matches to any of the values in a list	`SELECT product_name FROM product WHERE category IN ('Electronics', 'Furniture');`
LIKE: Performs wildcard matches using _ or %	`SELECT product_name FROM product WHERE product_name LIKE '%Desk%';`

Joins

```
SELECT t1.*, t2.*
FROM t1
join_type t2 ON t1.col = t2.col;
```

Table 1: A, B, C
Table 2: A, B, D

INNER JOIN: show all matching records in both tables.

LEFT JOIN: show all records from left table, and any matching records from right table.

RIGHT JOIN: show all records from right table, and any matching records from left table.

FULL JOIN: show all records from both tables, whether there is a match or not.

CASE Statement

Simple Case	`CASE name WHEN 'John' THEN 'Name John' WHEN 'Steve' THEN 'Name Steve' ELSE 'Unknown' END`
Searched Case	`CASE WHEN name='John' THEN 'Name John' WHEN name='Steve' THEN 'Name Steve' ELSE 'Unknown' END`

Common Table Expression

```
WITH queryname AS (
   SELECT col1, col2
   FROM firsttable)
SELECT col1, col2..
FROM queryname...;
```
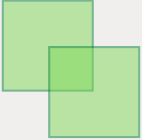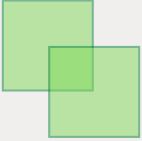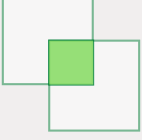
Modifying Data

Insert	`INSERT INTO tablename (col1, col2...) VALUES (val1, val2);`
Insert from a Table	`INSERT INTO tablename (col1, col2...) SELECT col1, col2...`
Insert Multiple Rows	`INSERT INTO tablename (col1, col2…) VALUES (valA1, valB1), (valA2, valB2), (valA3, valB3);`
Update	`UPDATE tablename SET col1 = val1 WHERE condition;`
Update with a Join	`UPDATE t SET col1 = val1 FROM tablename t INNER JOIN table x ON t.id = x.tid WHERE condition;`
Delete	`DELETE FROM tablename WHERE condition;`

Indexes

Create Index	`CREATE INDEX indexname ON tablename (cols);`
Drop Index	`DROP INDEX indexname;`

Set Operators

UNION: Shows unique rows from two result sets.

UNION ALL: Shows all rows from two result sets.

INTERSECT: Shows rows that exist in both result sets.

MINUS: Shows rows that exist in the first result set but not the second.

Aggregate Functions

- SUM: Finds a total of the numbers provided
- COUNT: Finds the number of records
- AVG: Finds the average of the numbers provided
- MIN: Finds the lowest of the numbers provided
- MAX: Finds the highest of the numbers provided

Common Functions

- LENGTH(string): Returns the length of the provided string
- INSTR(string, substring): Returns the position of the substring within the specified string.
- CAST(expression AS datatype): Converts an expression into the specified data type.
- ADDDATE(input_date, days): Adds a number of days to a specified date.
- NOW: Returns the current date, including time.
- CEILING(input_val): Returns the smallest integer greater than the provided number.
- FLOOR(input_val): Returns the largest integer less than the provided number.
- ROUND(input_val, [round_to]): Rounds a number to a specified number of decimal places.
- TRUNCATE(input_value, num_decimals): Truncates a number to a number of decimals.
- REPLACE(whole_string, string_to_replace, replacement_string): Replaces one string inside the whole string with another string.
- SUBSTRING(string, start_position): Returns part of a value, based on a position and length.

Create Table

Create Table	`CREATE TABLE tablename ( column_name data_type );`
Create Table with Constraints	`CREATE TABLE tablename ( column_name data_type NOT NULL, CONSTRAINT pkname PRIMARY KEY (col), CONSTRAINT fkname FOREIGN KEY (col) REFERENCES other_table(col_in_other_table), CONSTRAINT ucname UNIQUE (col), CONSTRAINT ckname CHECK (conditions) );`
Create Temporary Table	`CREATE TEMPORARY TABLE tablename ( colname datatype );`
Drop Table	`DROP TABLE tablename;`

Alter Table

Add Column	`ALTER TABLE tablename ADD columnname datatype;`
Drop Column	`ALTER TABLE tablename DROP COLUMN columnname;`
Modify Column	`ALTER TABLE tablename CHANGE columnname newcolumnname newdatatype;`
Rename Column	`ALTER TABLE tablename CHANGE COLUMN currentname TO newname;`
Add Constraint	`ALTER TABLE tablename ADD CONSTRAINT constraintname constrainttype (columns);`
Drop Constraint	`ALTER TABLE tablename DROP constraint_type constraintname;`
Rename Table	`ALTER TABLE tablename RENAME TO newtablename;`

Window/Analytic Functions

```
function_name ( arguments ) OVER (
[query_partition_clause]
[ORDER BY order_by_clause
[windowing_clause] ] )
```

Example using RANK, showing the student details and their rank according to the fees_paid, grouped by gender:

```
SELECT
student_id, first_name, last_name, gender, fees_paid,
RANK() OVER (
  PARTITION BY gender ORDER BY fees_paid
) AS rank_val
FROM student;
```

Subqueries

Single Row	`SELECT id, last_name, salary FROM employee WHERE salary = ( SELECT MAX(salary) FROM employee );`
Multi Row	`SELECT id, last_name, salary FROM employee WHERE salary IN ( SELECT salary FROM employee WHERE last_name LIKE 'C%' );`

PostgreSQL Cheat Sheet

www.databasestar.com

SELECT Query

```
SELECT col1, col2
FROM table
JOIN table2 ON table1.col = table2.col
WHERE condition
GROUP BY column_name
HAVING condition
ORDER BY col1 ASC|DESC;
```

SELECT Keywords

DISTINCT: Removes duplicate results	`SELECT DISTINCT product_name FROM product;`
BETWEEN: Matches a value between two other values (inclusive)	`SELECT product_name FROM product WHERE price BETWEEN 50 AND 100;`
IN: Matches to any of the values in a list	`SELECT product_name FROM product WHERE category IN ('Electronics', 'Furniture');`
LIKE: Performs wildcard matches using _ or %	`SELECT product_name FROM product WHERE product_name LIKE '%Desk%';`

Joins

```
SELECT t1.*, t2.*
FROM t1
join_type t2 ON t1.col = t2.col;
```

Table 1: A, B, C
Table 2: A, B, D

INNER JOIN: show all matching records in both tables.
A-A, B-B

LEFT JOIN: show all records from left table, and any matching records from right table.
A-A, B-B, C

RIGHT JOIN: show all records from right table, and any matching records from left table.
A-A, B-B, D

FULL JOIN: show all records from both tables, whether there is a match or not.
A-A, B-B, C, D

CASE Statement

Simple Case	`CASE name`
	`    WHEN 'John' THEN 'Name John'`
	`    WHEN 'Steve' THEN 'Name Steve'`
	`    ELSE 'Unknown'`
	`END`

Searched Case	`CASE`
	`    WHEN name='John' THEN 'Name John'`
	`    WHEN name='Steve' THEN 'Name Steve'`
	`    ELSE 'Unknown'`
	`END`

Common Table Expression

```
WITH queryname AS (
    SELECT col1, col2
    FROM firsttable)
SELECT col1, col2..
FROM queryname...;
```
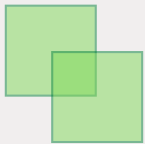
Modifying Data

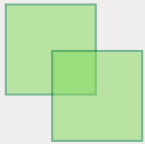|	

Insert	`INSERT INTO tablename`
	`(col1, col2...)`
	`VALUES (val1, val2);`

Insert from a Table	`INSERT INTO tablename`
	`(col1, col2...)`
	`SELECT col1, col2...`

Insert Multiple Rows	`INSERT INTO tablename`
	`(col1, col2...) VALUES`
	`(valA1, valB1),`
	`(valA2, valB2),`
	`(valA3, valB3);`

Update	`UPDATE tablename`
	`SET col1 = val1`
	`WHERE condition;`

Update with a Join	`UPDATE t`
	`SET col1 = val1`
	`FROM tablename t`
	`INNER JOIN table x`
	`ON t.id = x.tid`
	`WHERE condition;`

Delete	`DELETE FROM tablename` `WHERE condition;`

Indexes

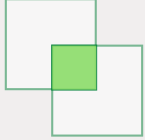|	
Create Index	`CREATE INDEX indexname` `ON tablename (cols);`
Drop Index	`DROP INDEX indexname;`

Set Operators

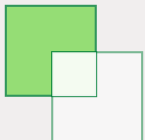UNION: Shows unique rows from two result sets.

UNION ALL: Shows all rows from two result sets.

INTERSECT: Shows rows that exist in both result sets.

EXCEPT: Shows rows that exist in the first result set but not the second.

Aggregate Functions

- SUM: Finds a total of the numbers provided
- COUNT: Finds the number of records
- AVG: Finds the average of the numbers provided
- MIN: Finds the lowest of the numbers provided
- MAX: Finds the highest of the numbers provided

Common Functions

- LENGTH(string): Returns the length of the provided string
- POSITION(string IN substring): Returns the position of the substring within the specified string.
- CAST(expression AS datatype): Converts an expression into the specified data type.
- NOW: Returns the current date, including time.
- CEIL(input_val): Returns the smallest integer greater than the provided number.
- FLOOR(input_val): Returns the largest integer less than the provided number.
- ROUND(input_val, [round_to]): Rounds a number to a specified number of decimal places.
- TRUNC(input_value, num_decimals): Truncates a number to a number of decimals.
- REPLACE(whole_string, string_to_replace, replacement_string): Replaces one string inside the whole string with another string.
- SUBSTRING(string, [start_pos], [length]): Returns part of a value, based on a position and length.

Create Table

Create Table	`CREATE TABLE tablename (`
	`    column_name data_type`
	`);`

Create Table with Constraints	`CREATE TABLE tablename (`
	`    column_name data_type NOT NULL,`
	`    CONSTRAINT pkname PRIMARY KEY (col),`
	`    CONSTRAINT fkname FOREIGN KEY (col)`
	`REFERENCES other_table(col_in_other_table),`
	`    CONSTRAINT ucname UNIQUE (col),`
	`    CONSTRAINT ckname CHECK (conditions)`
	`);`

Create Temporary Table	`CREATE TEMP TABLE tablename (`
	`    colname datatype`
	`);`

Drop Table	`DROP TABLE tablename;`

Alter Table

Add Column	`ALTER TABLE tablename ADD COLUMN columnname datatype;`
Drop Column	`ALTER TABLE tablename DROP COLUMN columnname;`
Modify Column	`ALTER TABLE tablename ALTER COLUMN columnname TYPE newdatatype;`
Rename Column	`ALTER TABLE tablename RENAME COLUMN currentname TO newname;`
Add Constraint	`ALTER TABLE tablename ADD CONSTRAINT constraintname constrainttype (columns);`
Drop Constraint	`ALTER TABLE tablename DROP constraint_type constraintname;`
Rename Table	`ALTER TABLE tablename RENAME TO newtablename;`

Window/Analytic Functions

```
function_name ( arguments ) OVER (
[query_partition_clause]
[ORDER BY order_by_clause
[windowing_clause] ] )
```

Example using RANK, showing the student details and their rank according to the fees_paid, grouped by gender:

```
SELECT
student_id, first_name, last_name, gender, fees_paid,
RANK() OVER (
  PARTITION BY gender ORDER BY fees_paid
) AS rank_val
FROM student;
```

Subqueries

Single Row	`SELECT id, last_name, salary`
	`FROM employee`
	`WHERE salary = (`
	`    SELECT MAX(salary)`
	`    FROM employee`
	`);`

Multi Row	`SELECT id, last_name, salary`
	`FROM employee`
	`WHERE salary IN (`
	`    SELECT salary`
	`    FROM employee`
	`    WHERE last_name LIKE 'C%'`
	`);`