SQL BOOTCAMP CHEAT SHEET

Basic tables:

book_table

Book	COGs	Author
[book name]	Cost of goods	[author's last name]

auth_table

Last_Name	First_Name	Country	Birth_Year
[author's last name]	[author's first name]	[author's country of origin]	[the year the author was born]

sales_table

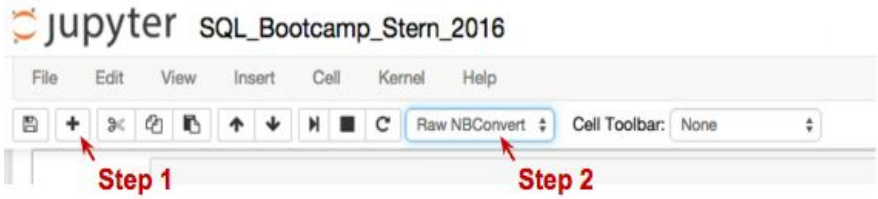
Book	Revenue	Date
[book name]	[revenue amount]	[sale date]

Related

Related

Using the program:

- If you accidentally double-click on a block of text, and suddenly it looks like code, hit **SHIFT-RETURN** or **'Cell ➢ Run'**.
- If you try to run a query and the output doesn't refresh, select **'Cell ➢ Run All'** to reboot the program.
- Nothing that you write in the challenges and exercises will save after you close this program.
- If you want to save something that you've written, follow the steps below:



Strict Order of Clauses for Single Non-Union Query:

```
SELECT
FROM
      JOIN…ON
WHERE
GROUP BY
      HAVING
ORDER BY
LIMIT
```

Strict Order of Clauses for Query with Union:

```
SELECT
FROM
      JOIN…ON
WHERE
GROUP BY
      HAVING
UNION [or UNION ALL]
[repeat structure above second query]
ORDER BY
LIMIT
```

All Clauses:

Reading a table's structure:

```
PRAGMA TABLE_INFO(table_name)
```

SELECT & FROM:

```
SELECT
     *
FROM
     Table_name
```

WHERE:

```
SELECT
      column_a
FROM
      table_name
WHERE
      column_a = x
```

Options for WHERE:

`WHERE column_a = 'some_text'`	Put text in quotations. Capitalization is important!
`WHERE column_a != x`	Value DOES NOT equal x
`WHERE column_a < x`	Value is less than x
`WHERE columna_a <= x`	Value is less than or equal to x
`WHERE column_a IN (x, y)`	Value can be EITHER x OR y
`WHERE column_a NOT IN (x, y)`	Value is NEITHER x NOR y
`WHERE column_a BETWEEN x AND y`	Use BETWEEN for a range
`WHERE column_a = x AND column_b = y`	AND lets you add more requirements
`WHERE column_a = x OR column_b = y`	OR will include results that fulfill either criteria
`WHERE (column_a = x AND column_b = y) OR (column_c = z)`	Use parentheses to create complex AND/OR statements

LIKE and wildcards %:

```
SELECT
      column_a
FROM
     table_name
WHERE
     column_a LIKE '%tExT%'
```

ORDER BY:

```
SELECT
      column_a
FROM
      table_name
[WHERE clause, optional]
ORDER BY                          → sorts the result-set by column_a
      column_a DESC     → DESC is optional, it sorts results in descending order
```

LIMIT:

```
SELECT
     column_a
FROM
     table_name
[WHERE clause]
[ORDER BY clause]
LIMIT N                → limits the result-set to N rows
```

SIMPLE JOIN:

```
SELECT
      table_x.column_a,       → read this as "column_a from table_x"
      table_y.column_b,       → "column_b from table_y"
FROM
     table_x
     JOIN table_y
      ON table_x.key_column_x = table_y.key_column_y
            → table_x's key_column_x has corresponding values with table_y's key_column_y
```

Other JOIN options:

- LEFT JOIN
- RIGHT JOIN
- OUTER JOIN (aka FULL OUTER JOIN)

ALIASES:

```
SELECT
      column_a AS alias_a          → creates an alias for column_a
FROM
     table_name
WHERE
     alias_a = x                   → optional; use the alias in the WHERE clause
ORDER BY
     alias_a                       → optional; use the alias in the ORDER BY clause
```

OPERATORS:

```
SELECT
     column_a + column_b,          → adds the values in column_a and columns_b
     column_a - column_b,          → subtracts
     column_a * column_b,          → multiplies
     column_a / column_b,                → divides
     (column_a + column_b) * column_c     → use parentheses to make more
complex calculations
FROM
      table_name
```

FUNCTIONS:

Search on Google for other functions to suit your needs, there are many more than the ones listed below. These are just some basics:

`SUM(column_a)`	Sums values
`AVG(column_a)`	Averages values
`ROUND(AVG(column_a), x)`	Rounds values to x decimals; can be used alone or with other functions
`COUNT(column_a) or (*)`	Returns a count of the number of rows with non-null values in column_a
`DISTINCT(column_a)`	Same as `DISTINCT column_a`, but can be wrapped in other functions (eg. `COUNT(DISTINCT(column_a))`
`MAX(column_a)`	Returns the largest value in the column
`MIN(column_a)`	Returns the smallest value in the column
`GROUP_CONCAT(column_a, '[separator]')`	Returns a comma-separated list of values in column_a. Specify a separator in quotes
`UPPER(column_a)`	Returns all-caps values

GROUP BY:

```
SELECT
      column_a,
      SUM(column_b)          → use a function in the SELECT clause
FROM
      table_name
[WHERE clause]
GROUP BY                     → creates one group for each unique value in column_a
     column_a
[ORDER BY clause]
[LIMIT clause]
```

HAVING:

```
SELECT
      column_a,
      FUNCTION(column_b)
FROM
      table_name
[WHERE]
GROUP BY
      column_a
      HAVING FUNCTION(column_b) > x      → returns groups whose value is greater than x
[ORDER BY]
[LIMIT]
```

CASE WHEN / Conditional Statements:

```
SELECT
      CASE WHEN some_column = x THEN value_if_true
                WHEN some_column = y THEN other_value_if_true
                ELSE value_if_false
                END
FROM
      some_table
```

NESTING QUERIES:

```
SELECT
      some_column
FROM
      some_table
WHERE
      some_column IN (SELECT some_other_column FROM some_other_table)
```

UNION / UNION ALL

```
SELECT
      some_column
FROM
      table_x
UNION              → or use UNION ALL, see explanation below
SELECT
     some_other_column
FROM
      table_y
```

Use `UNION ALL` to keep the results from the second query listed after the results from the first query.

Dialect Differences:

Description	Microsoft SQL Server	MySQL	Oracle	SQLite
Reading a table's structure	`SP_Help tablename`	`DESCRIBE tablename`	`DESCRIBE tablename`	`PRAGMA TABLE_INFO(tablename)`
Limiting rows	`SELECT TOP N column_name`	`LIMIT N`	`WHERE ROWNUM <= N`	`LIMIT N`
`JOIN` or `INNER JOIN`	✓	✓	✓	✓
`LEFT JOIN` or `LEFT OUTER JOIN`	✓	✓	✓	✓
`RIGHT JOIN` or `RIGHT OUTER JOIN`	✓	✓	✓	not supported
`OUTER JOIN` or `FULL OUTER JOIN`	✓	not supported	✓	not supported
`IF`	`IF logical_test PRINT value_if_true`	`IF(logical_test, value_if_true, value_if_false)` (same as Excel)	`IF logical_test THEN value_if_true ELSIF...END IF`	NOT SUPPORTED
`CASE WHEN`	✓	✓	✓	✓
`ROLLUP`	`GROUP BY column_a WITH ROLLUP`	`GROUP BY column_a WITH ROLLUP`	`GROUP BY ROLLUP (column_a)`	not supported