SELECT STATEMENT

FILTERING RESULTS:

Use the WHERE clause. Examples:

```
select * from tutorial.us_housing_units
WHERE year >= 1983 AND month = 1
```

SUMMING COLUMNS:

You can use Algebra in column names.

RENAMING COLUMNS:

Use the AS clause. Example:

```
SELECT year, month, south+west+midwest+northeast
AS "Total Units" FROM tutorial.us_housing_units
WHERE year >= 2008
```

ORDERING RESULTS:

Use the ORDER BY clause. Example:

```
SELECT year, month, price FROM aapl_prices
WHERE year >= 2000
ORDER BY year, month
```

AGGREGATING:

SUM
AVG
COUNT
COUNT(DISTINCT)
GROUP_CONCAT
MIN
MAX
STDDEV
VARIANCE

Remember, these are usually used in conjunction with a GROUP BY clause

HAVING CLAUSE:

Used to filter results from aggregation. Ex:

```
Select blah, SUM(blah2) from table
having SUM(blah2) > 10 group by blah
```

HAVING VS WHERE:

Having clause is used AFTER aggregation runs. Where clause filters rows BEFORE they even get Considered for the query.

RENAMING COLUMNS:

Use the AS clause. Example:

```
SELECT year, month, south+west+midwest+northeast
AS "Total Units" FROM tutorial.us_housing_units
WHERE year >= 2008
```

DEFINING NEW COLUMNS BASED ON OTHER COLS

Use the CASE statement. Ex:

```
SELECT col1, CASE WHEN col2 = NULL THEN 0
            ELSE col2 END AS alias
FROM dbname.tablename
```

GENERAL SELECT RECIPE

```
SELECT column1, aggregate(column2)
FROM databasename.tablename
WHERE columnx ><= somevalue1
GROUP BY column1
ORDER BY columny
HAVING aggregate(column2) ><= somevalue2
LIMIT x
```

JOIN RECIPE

```
SELECT col1, col2
FROM db1.table1 tbl1
JOIN db1.table2 tbl2   (aliases)
  ON tbl1.colx = tbl2.coly
```