SQL Summary Sheet

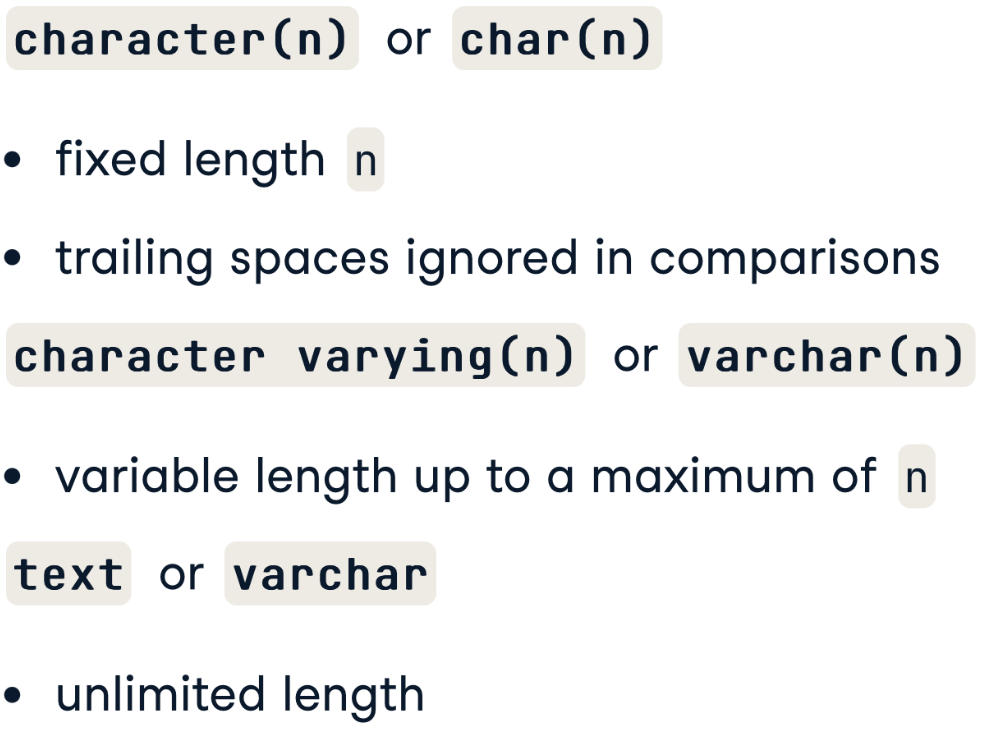
This document will be divided in three sections:

```
1. SQL basics (data types)
2. SQL usage
3. Database design
```

1. SQL basics

Data types in SQL

Many data types exist in SQL



`character(n)` or `char(n)`

- fixed length `n`
- trailing spaces ignored in comparisons

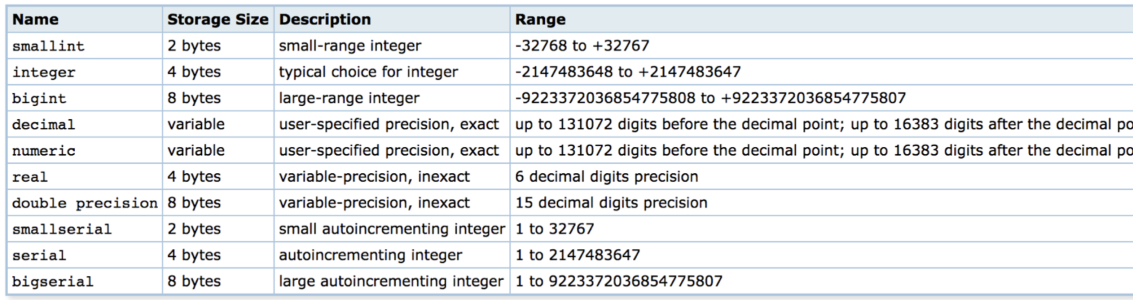
`character varying(n)` or `varchar(n)`

- variable length up to a maximum of `n`

`text` or `varchar`

- unlimited length

- String variables



Name	Storage Size	Description	Range
`smallint`	2 bytes	small-range integer	-32768 to +32767
`integer`	4 bytes	typical choice for integer	-2147483648 to +2147483647
`bigint`	8 bytes	large-range integer	-9223372036854775808 to +9223372036854775807
`decimal`	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal po
`numeric`	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal po
`real`	4 bytes	variable-precision, inexact	6 decimal digits precision
`double precision`	8 bytes	variable-precision, inexact	15 decimal digits precision
`smallserial`	2 bytes	small autoincrementing integer	1 to 32767
`serial`	4 bytes	autoincrementing integer	1 to 2147483647
`bigserial`	8 bytes	large autoincrementing integer	1 to 9223372036854775807

- Numerical variables
- date/time data types

  - `DATE`, `TIME`, `TIMESTAMP`, `INTERVAL`
    - TIMESTAMP contains date and time and is precise to the microsecond
    - depending on needs, DATE or TIME may be better options
    - `rental_date + INTERVAL ‘3 days’`
      - adds 3 days to the field rental_date

- arrays
  - To access array data, it is like anything:
    - `SELECT field[1][1] FROM table`
    - Indexing start with 0
  - `WHERE "text_to_search" = ANY(field_as_array)`
    - This will search for the text_to_search in all possible fields of the array
    - Equivalent to `WHERE field_as_array @> ARRAY[‘text_to_search’]`

- Access the data types from the INFORMATION_SCHEMA table
  - `SELECT column_name, data_type FROM INFORMATION_SCHEMA.COLUMNS WHERE column_name IN () AND table_name='xxx';`

- Changing (casting) a column type into another
  - `CAST(value AS new_type)` equivalent to `value::new_type`

- OPERATION ON DATES
  - Subtracting dates gives an integer (eg. 2 days)
  - Adding an integer to a date returns a dates “inflated” by the number of days
  - The difference of two TIMESTAMP gives an INTERVAL
    - This can be obtained with the AGE(TIMESTAMP, TIMESTAMP) function
  - timestamp “2016-05-01” + 21 * INTERVAL ‘1’ day
    - we can multiply intervals (returns an interval), which can be added to a timestamp (returns a timestamp)
    - NOW() + ‘1 year 2 days 3 minutes’::interval
  - SELECT NOW() -> timestamp with timezone
    - SELECT NOW()::timestamp (remove the timezone)
      - This is specific to PostgreSQL
      - SELECT CAST(NOW() as timestamp) is universal
  - SELECT CURRENT_TIMESTAMP(2) now() rounded at 2 digits
    - SELECT CURRENT_DATE -> Date
    - SELECT CURRENT_TIME -> Time with timezone
  - EXTRACT(field from source)
    - cource can be date, timestamp, time
    - field can be year, month, quarter, day of week (aliased as dow)
    - SELECT EXTRACT(month FROM NOW() AS month;
      - Extracts the month field from a timestamp
    - SELECT DATE_PART(‘quarter’, NOW()) AS quarter
    - SELECT DATE_TRUNC(‘month’, NOW())
      - Returns a timestamp with the same year and month, but everything else set at beginning value (day 1, hour 0, etc..)
  - to_char(date_created, ‘day’) Converts day of weeks to Monday, Tuesday, etc.

- OPERATIONS ON CHARACTER DATA
  - Concatenate strings: SELECT field1 || ‘sep’ || field2 AS new_string
    - PostgreSQL as its built-in CONCAT(field1, sep, field2) function
    - non-string data can be concatenated with string
    - CONCAT() ignores null values, while || will return NULL
  - UPPER(field), LOWER(field), INITCAP(field)
  - WHERE fav_fruit ILIKE “%apple%”
    - ILIKE is case-insensitive!!
  - REPLACE(field, “str_to_change”, “new_str”)
  - REVERSE(field)
    - inverses everything from the string
  - CHAR_LENGTH(field)
    - LENGTH() also workds
  - POSITION(‘str’ IN field)
  - LEFT(field, n), RIGHT(field,n)
    - extracts the first n characters of field
  - SUBSTRING(field, 10, 50)
    - extracts from char 10 with length 50
    - SUBSTR(email FROM 0 FOR POSITION(’@‘ IN email)
      - FROM: beginning, FOR ending position excluded
    - SUBSTRING(email FROM POSITION(’@‘ IN email)+1 FOR CHAR_LENGTH(email))
  - TRIM(leading/trailing/both(default) ‘ ‘ from string)
    - The first tzo parameters are optional
      - SELECT TRIM(“ word “) -> “word”
      - LTRIM() or RTRIM() or BTRIM() [b for both] can also be used.
    - TRIM(street, “ 0123456789#/.”)
      - Will clean the street names from all these characters but not the middle spaces.
  - LPAD(‘padded’, 10, ‘%’) ; RPAD()
    - adds # to the word until the length is 10 %%%%padded
    - default is padding with spaces.
    - if the word is longer than the limit, it will be truncated
  - SPLIT_PART(string, delimiter, part)
- Full-text search
  - WHERE to_tsvector(field) @@ to_tsquery(‘str_to_search’)
    - case-insensitive