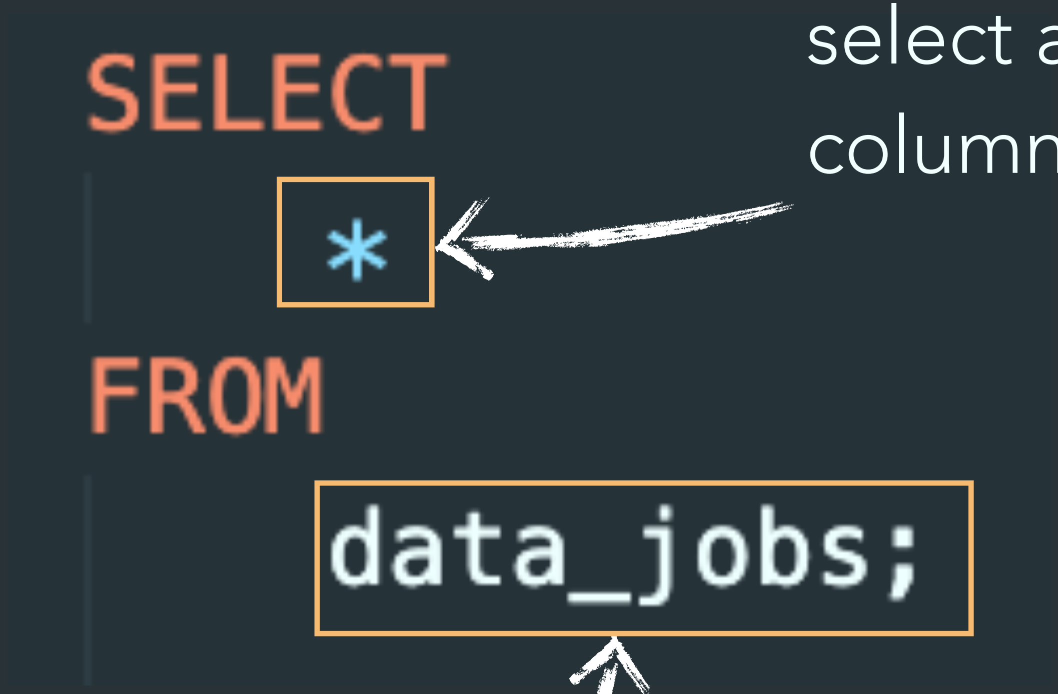
kokchun giang

querying data in a relational database using DQL

using **SELECT clause** to query columns

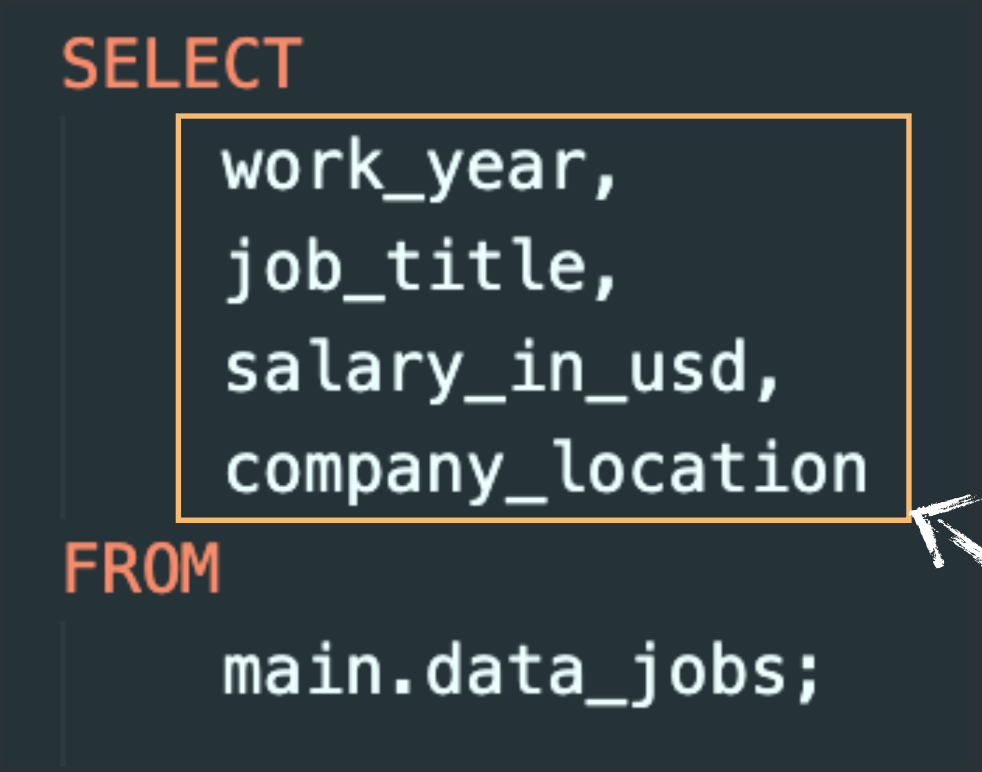


```sql
SELECT
    *
FROM
    data_jobs;
```

select all columns

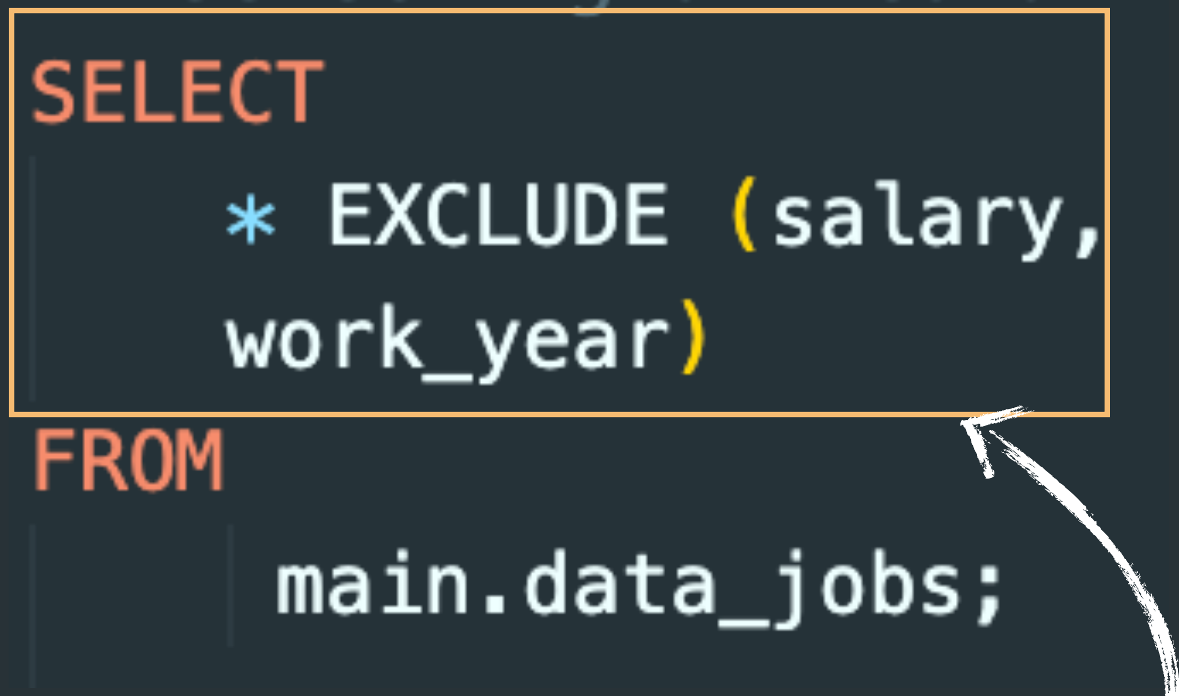
data source is a table or view

note that this works when the table is in main schema



```sql
SELECT
    work_year,
    job_title,
    salary_in_usd,
    company_location
FROM
    main.data_jobs;
```

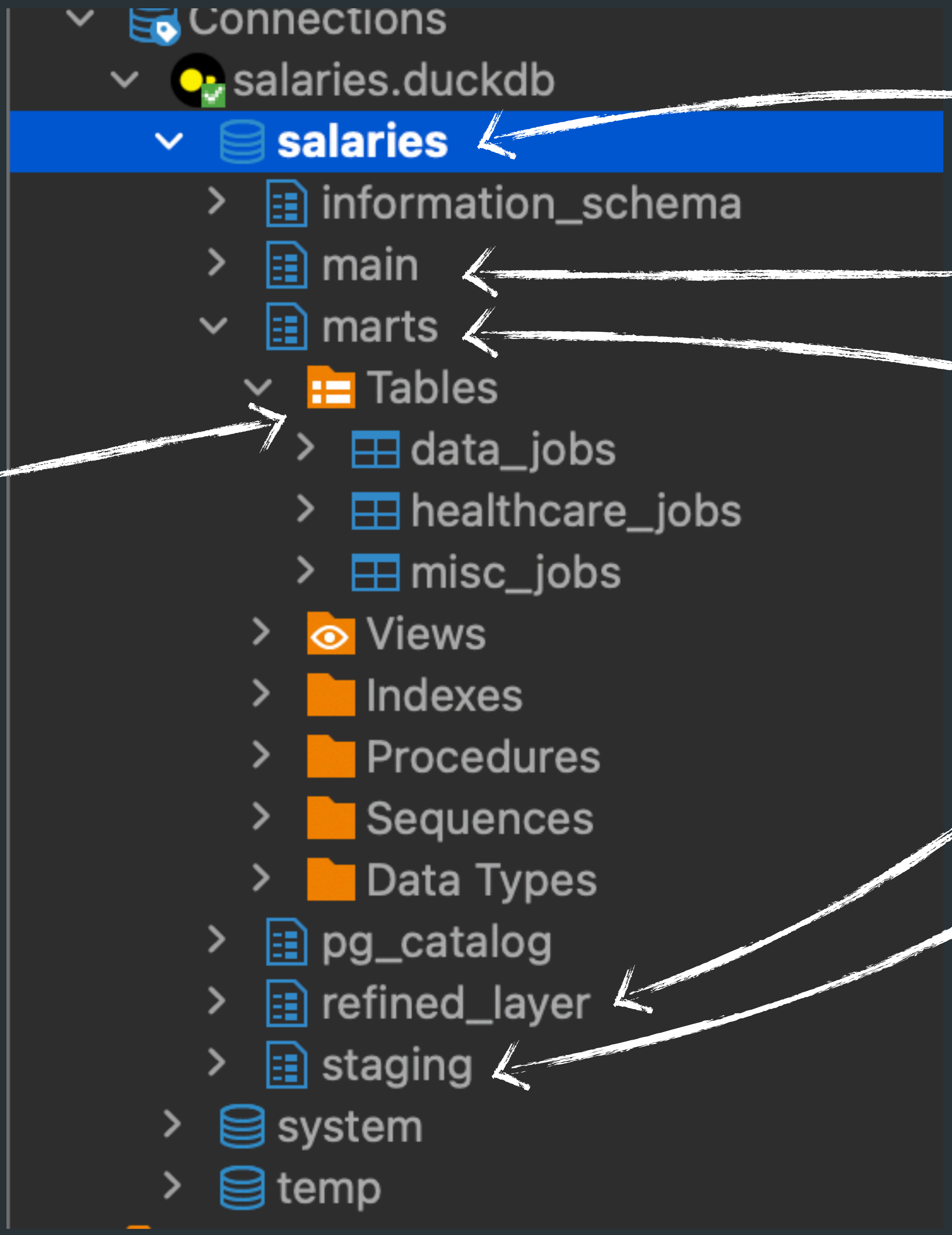
choosing all rows of these columns



```sql
SELECT
    * EXCLUDE (salary,
    work_year)
FROM
    main.data_jobs;
```

choosing all columns except for those columns specified inside of EXCLUDE

database hierarchy to organize database objects



- Connections
  - salaries.duckdb
    - salaries
      - information_schema
      - main
      - marts
        - Tables
          - data_jobs
          - healthcare_jobs
          - misc_jobs
        - Views
        - Indexes
        - Procedures
        - Sequences
        - Data Types
      - pg_catalog
      - refined_layer
      - staging
    - system
    - temp

database

default schema

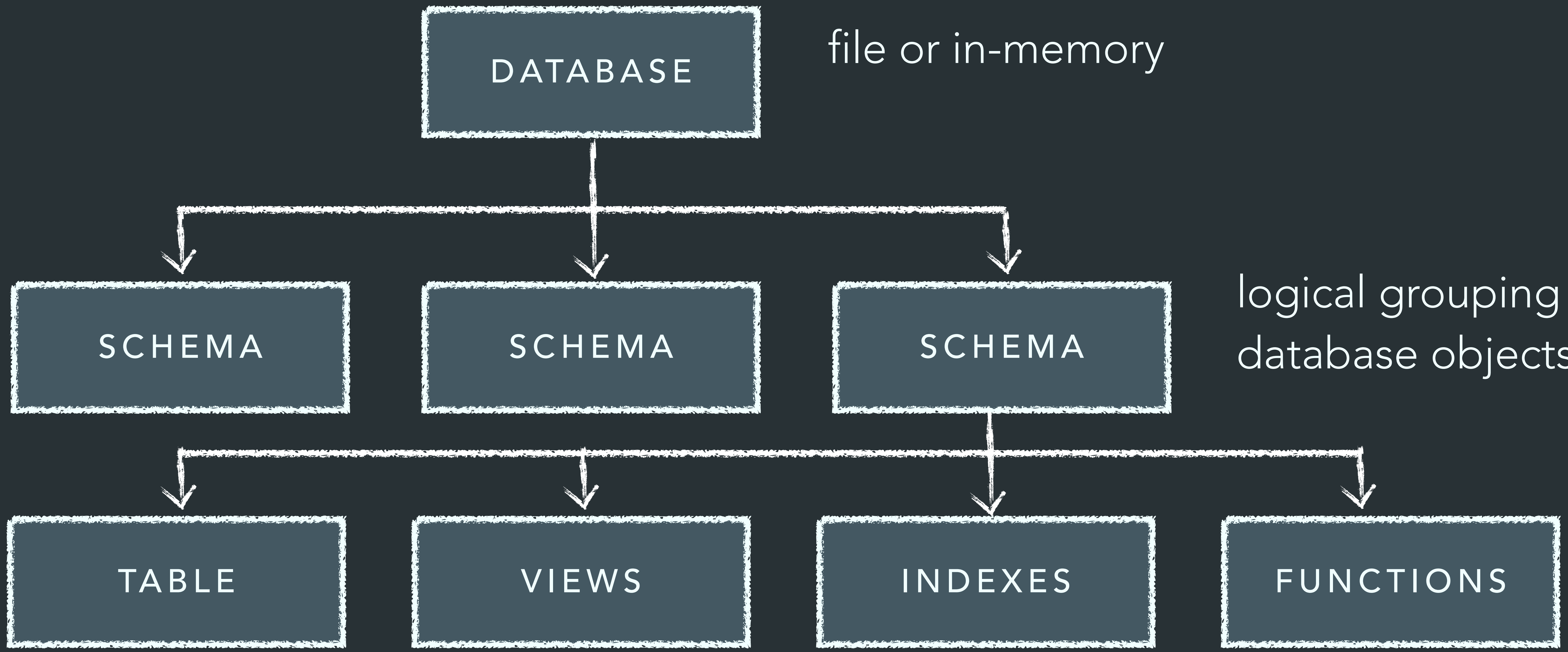
custom schemas

tables

note
this image is from
dbeaver connected
to duckdb

other ides have
similar structure

hierarchy of **database objects** in duckdb

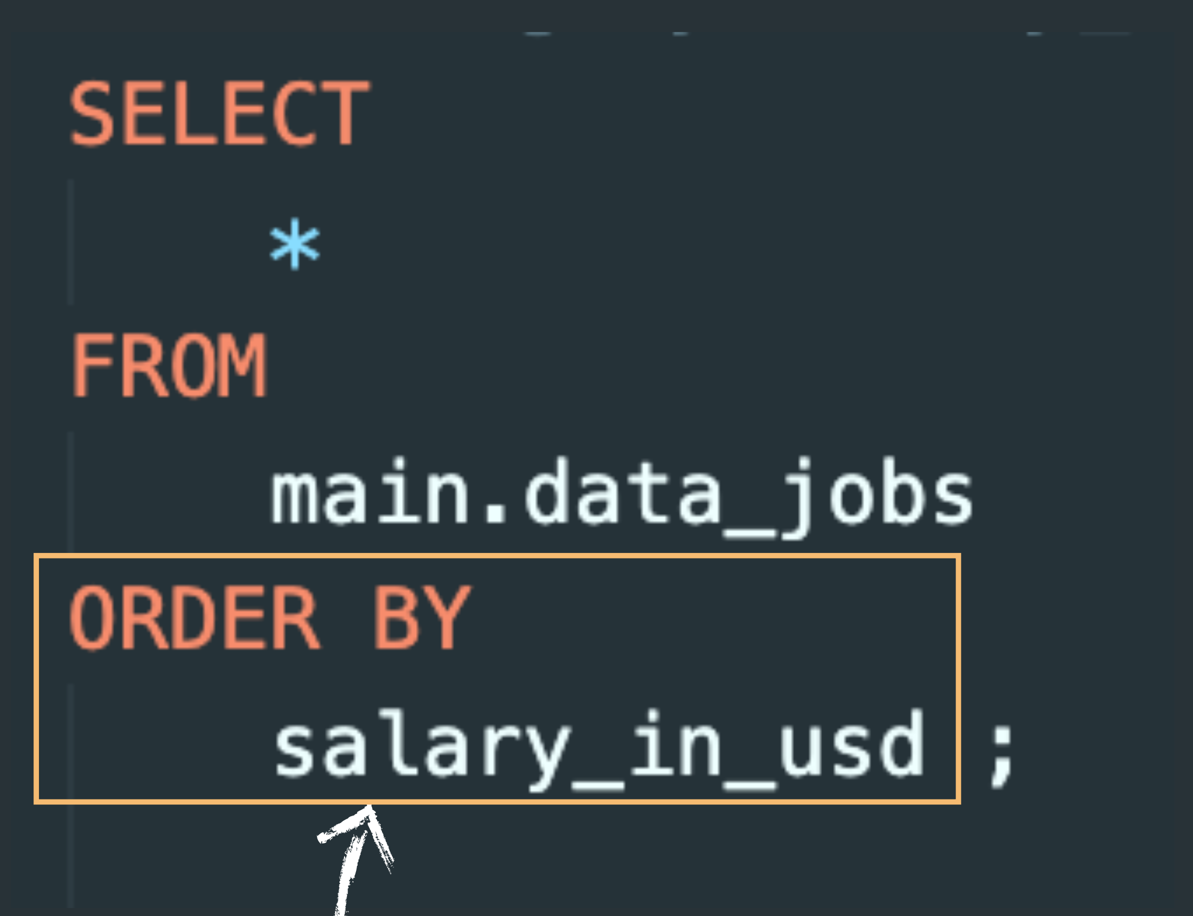


file or in-memory

logical grouping for
database objects

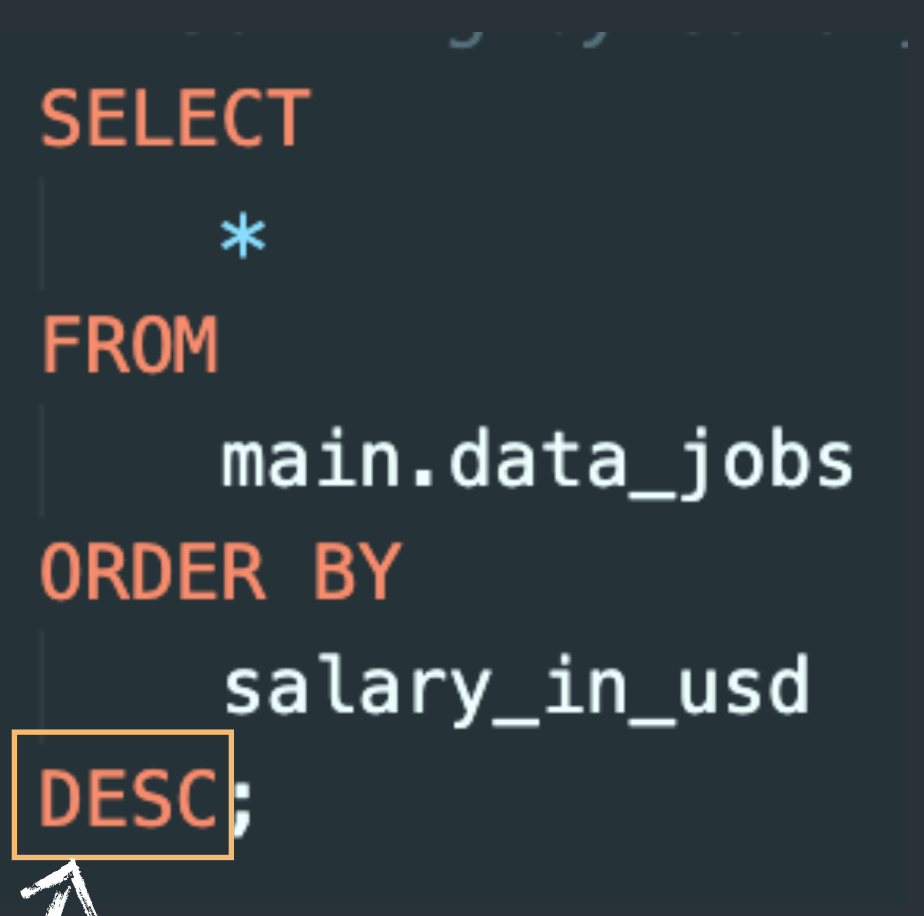
database
objects

ORDER BY clause to sort the data



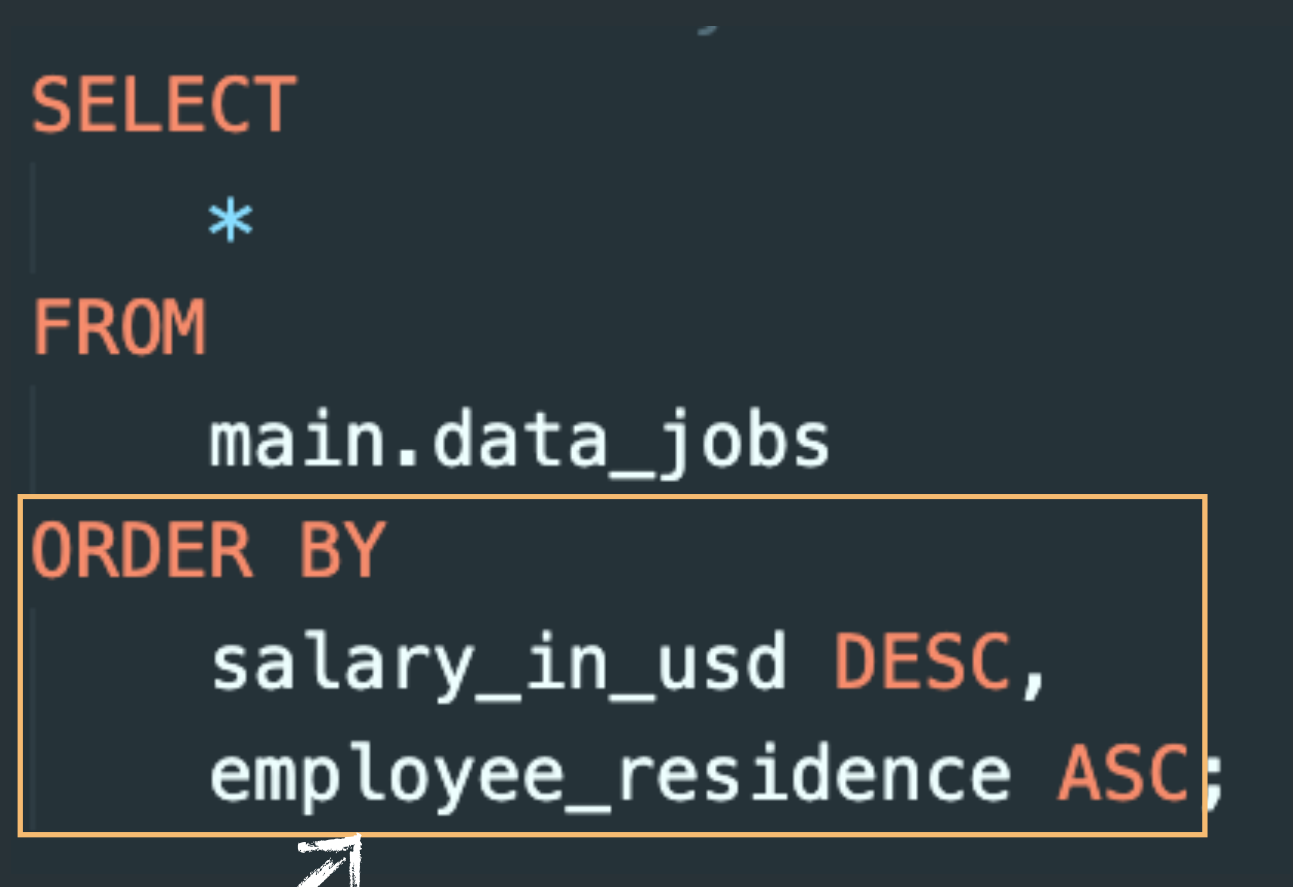
```
SELECT
    *
FROM
    main.data_jobs
ORDER BY
    salary_in_usd ;
```

sorts ascending by default



```
SELECT
    *
FROM
    main.data_jobs
ORDER BY
    salary_in_usd
DESC;
```

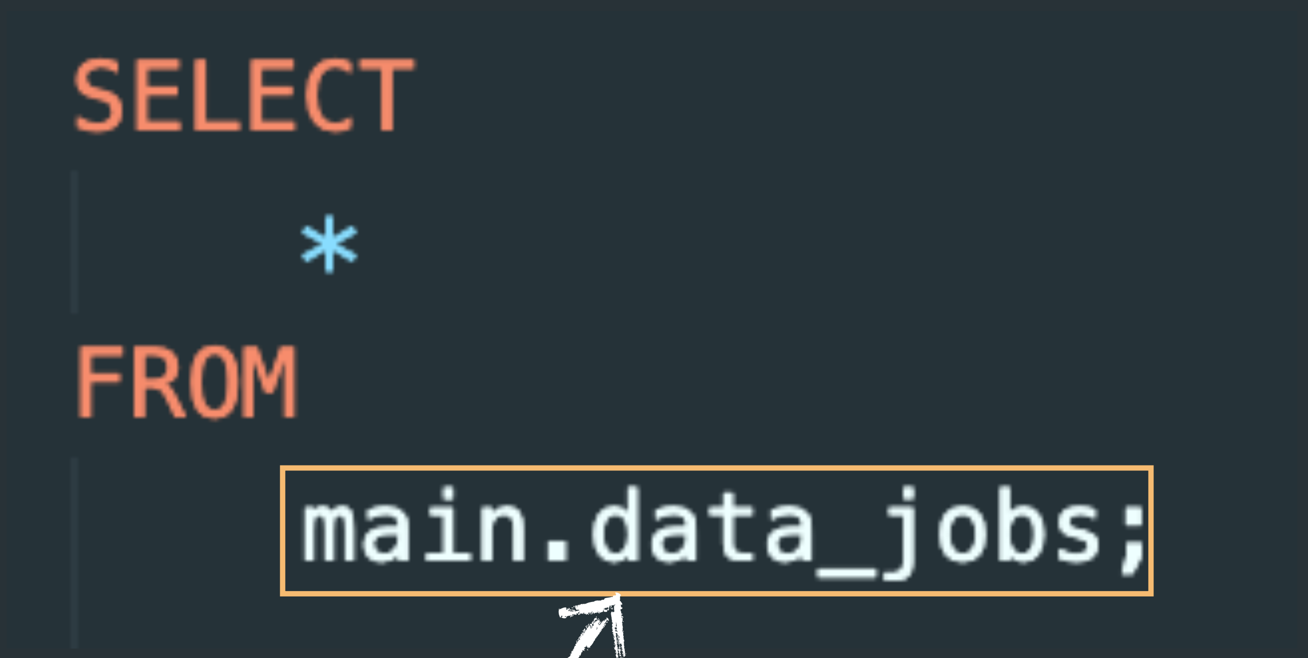
desc keyword to sort in descending order



```
SELECT
    *
FROM
    main.data_jobs
ORDER BY
    salary_in_usd DESC,
    employee_residence ASC;
```

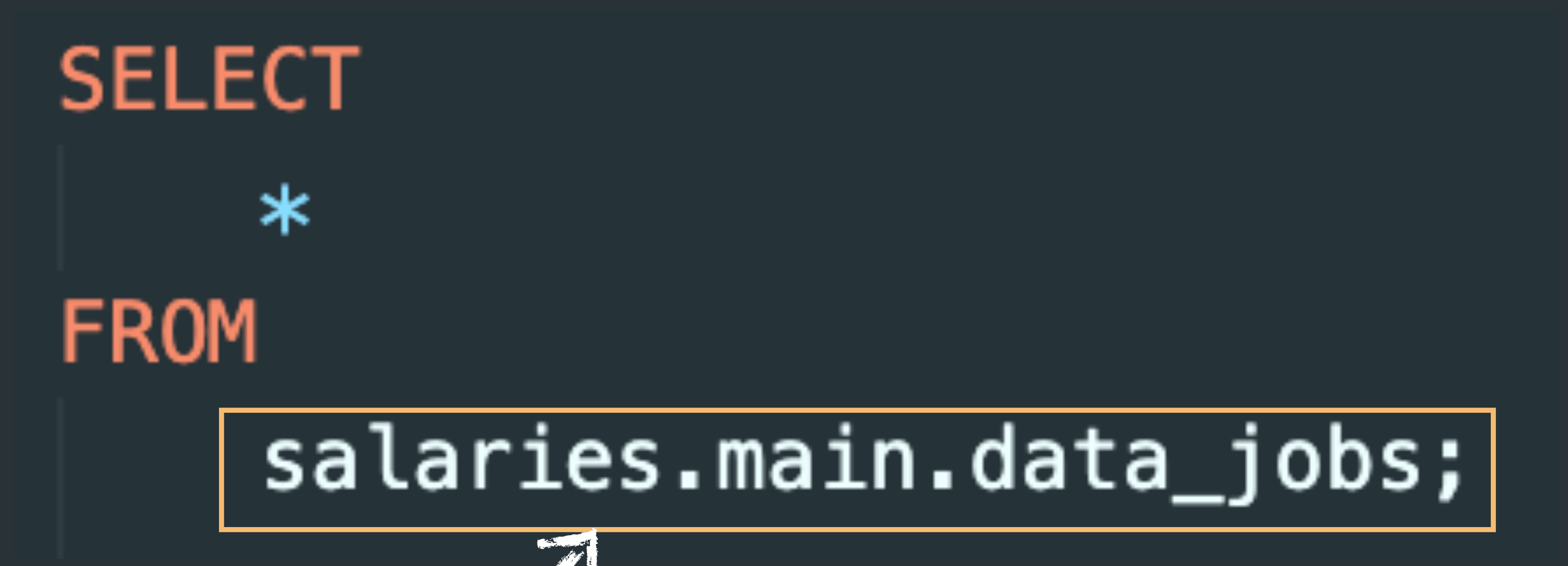
sorts by first column descending, if tie, it sorts by second column ascending

creating **namespace** to organize



```
SELECT
    *
FROM
    main.data_jobs;
```

schema namespace,
looks into main schema
and check for a table
called data_jobs

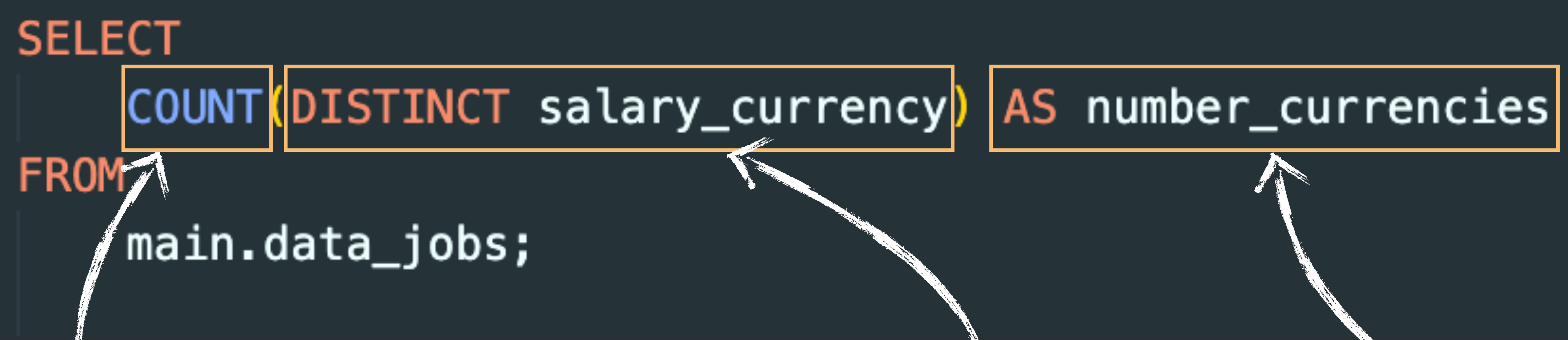


```
SELECT
    *
FROM
    salaries.main.data_jobs;
```

using fully
qualified name

avoid name collisions and
better organization with
namespace

use **alias** with the **AS keyword** to name columns

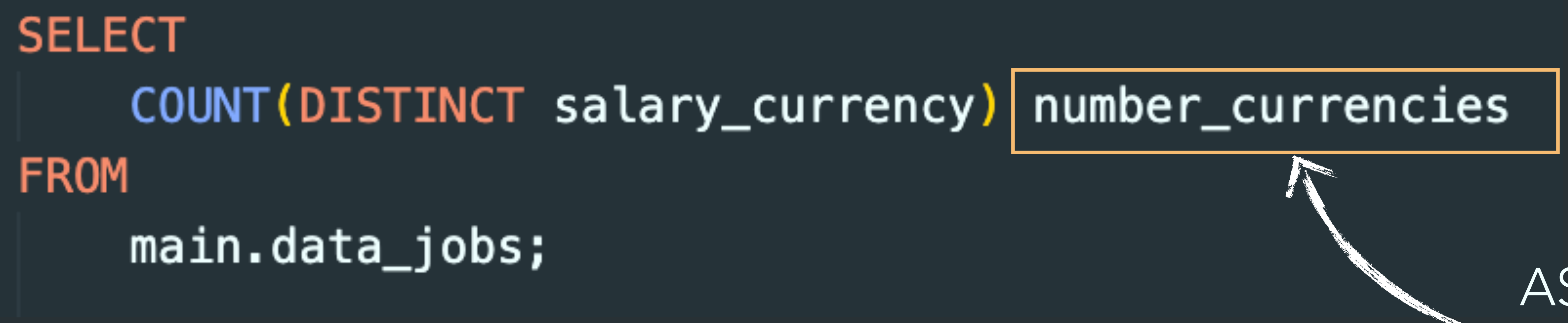


```
SELECT
    COUNT(DISTINCT salary_currency) AS number_currencies
FROM
    main.data_jobs;
```

aggregate function that counts number of occurrences in the column and returns a value

selects the unique values in this column

AS keyword to rename the columns

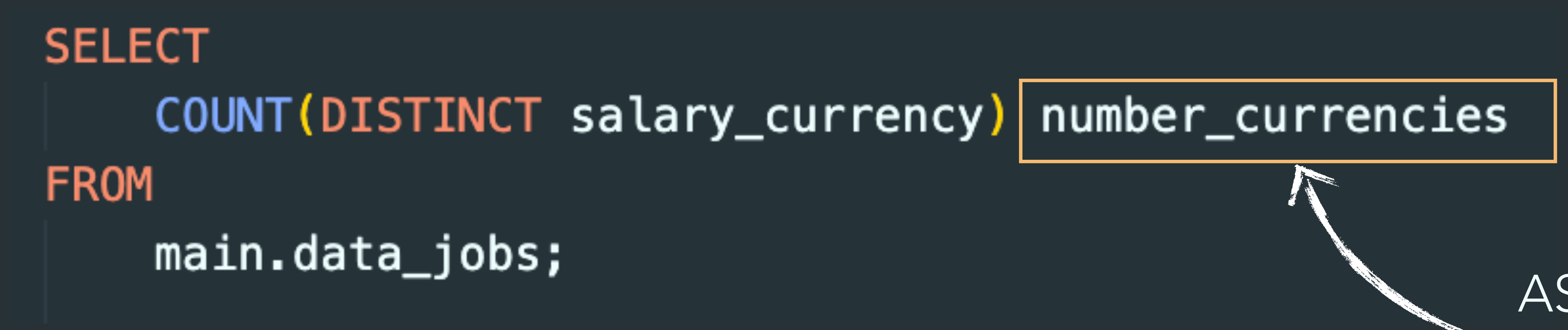


```
SELECT
    COUNT(DISTINCT salary_currency) number_currencies
FROM
    main.data_jobs;
```

AS keyword is optional, space work as well

choose the option that gives best readability and be consistent

use **alias** without the **AS keyword** to name columns

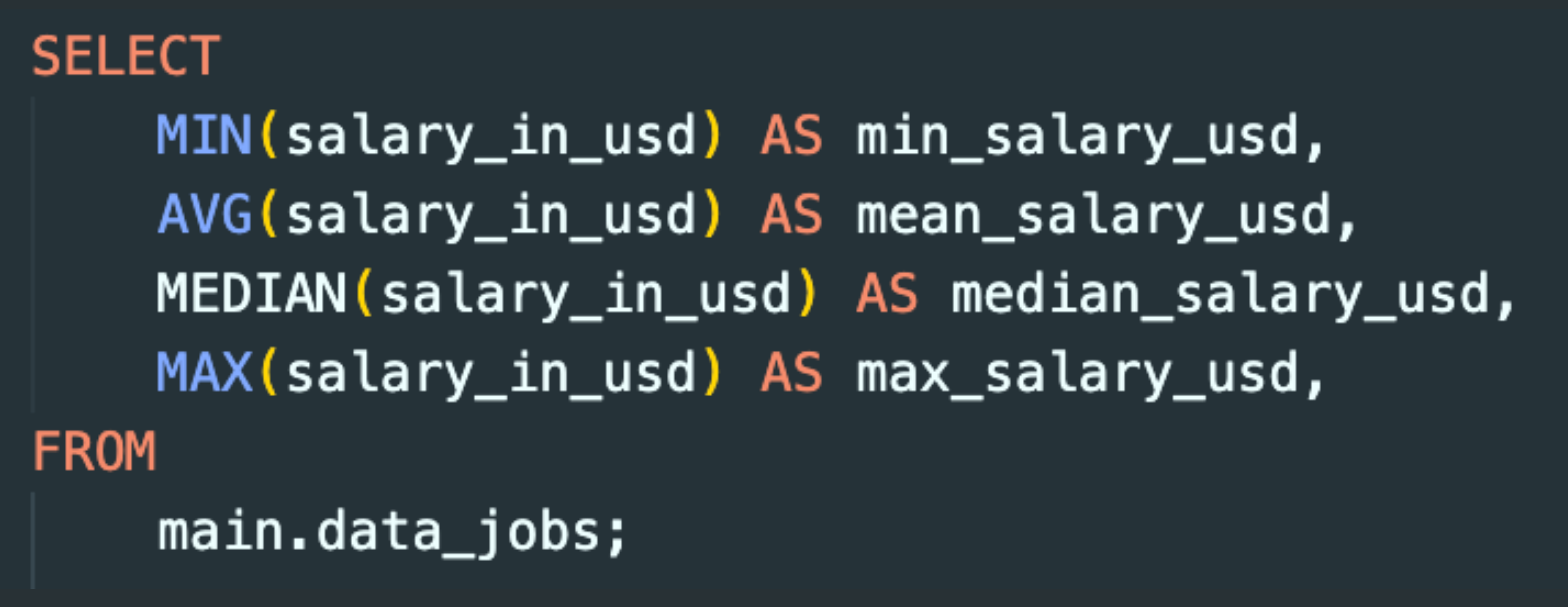


```sql
SELECT
    COUNT(DISTINCT salary_currency) number_currencies
FROM
    main.data_jobs;
```

AS keyword is
optional, space
work as well

choose the option that
gives best readability
and be consistent

using **aggregate functions** to combine multiple rows into one value



```
SELECT
    MIN(salary_in_usd) AS min_salary_usd,
    AVG(salary_in_usd) AS mean_salary_usd,
    MEDIAN(salary_in_usd) AS median_salary_usd,
    MAX(salary_in_usd) AS max_salary_usd,
FROM
    main.data_jobs;
```

common aggregate functions to
get descriptive statistics