Northwestern | McCORMICK SCHOOL OF ENGINEERING

RELATIONAL DATABASE TUTORIAL (SQL)

Grant Gasser, Blaine Rothrock, Sundar Thevar

Uses material from w3Schools SQL Tutorial and applies examples to the Citizens Police Data Project

Northwestern | McCORMICK SCHOOL OF ENGINEERING

Github Link with Example Queries

https://github.com/blainerothrock/sql_demo

What is SQL?

- SQL stands for **Structured Query Language**
  - No longer Structured English QUEry Language (SEQUEL)
- SQL lets you **access** and **manipulate** databases
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

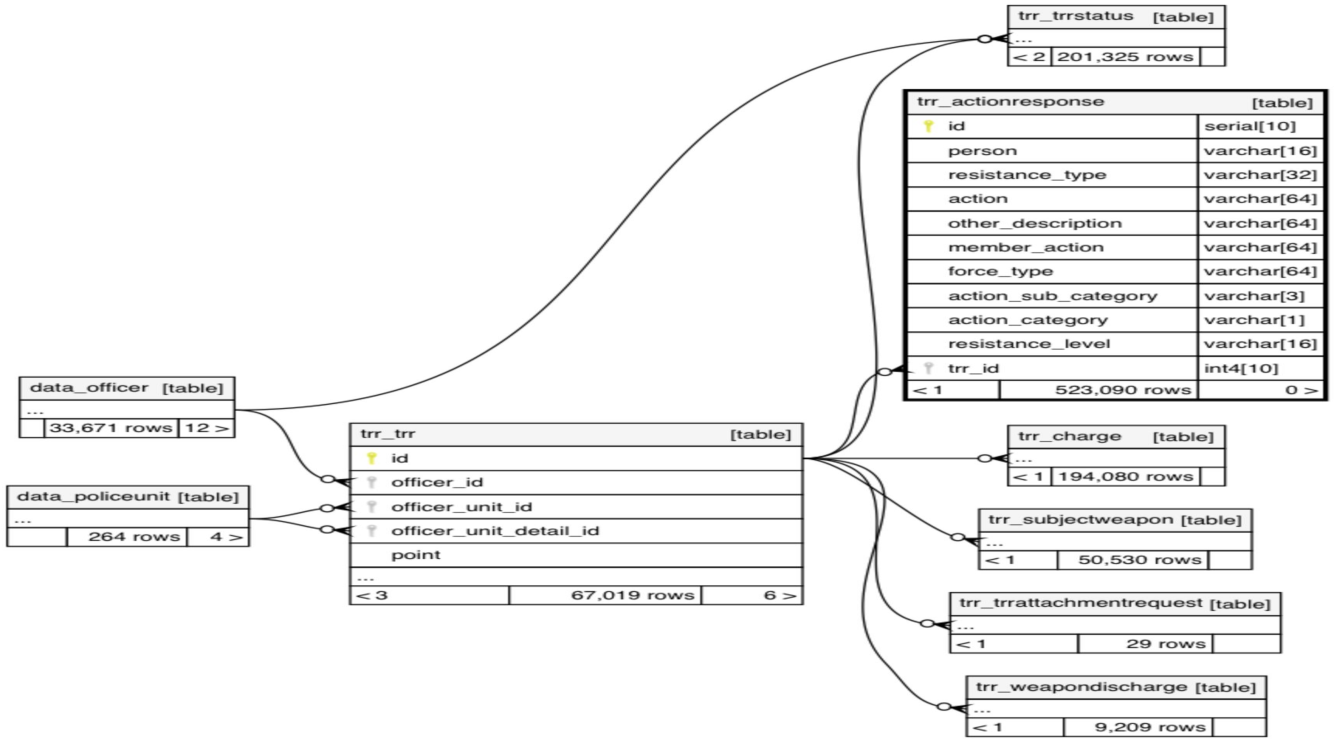
```
SELECT * FROM data_officer
```

What can SQL do?

- Execute queries against a database
- Retrieve data from a database
- Insert records in a database
- Update records in a database
- Delete records from a database
- Create new databases
- Create new tables in a database
- Create stored procedures in a database
- Create views in a database
- Can set permissions on tables, procedures, and views

Working with SQL

Schema:



What is PostgreSQL?

- Open source RDBMS
- Release in 1996
- Large support with interfaces

Setup

- CPDP instructions found in Checkpoint 1
  - This will set you up with the command line interface for PostgreSQL
  - Knowing the command line is **extremely helpful**
- GUI Tools
  - pgAdmin 4 (the standard, open source)
  - DataGrip (JetBrains)
  - Postico (dead simple, macOS)

SQL Query Tutorial

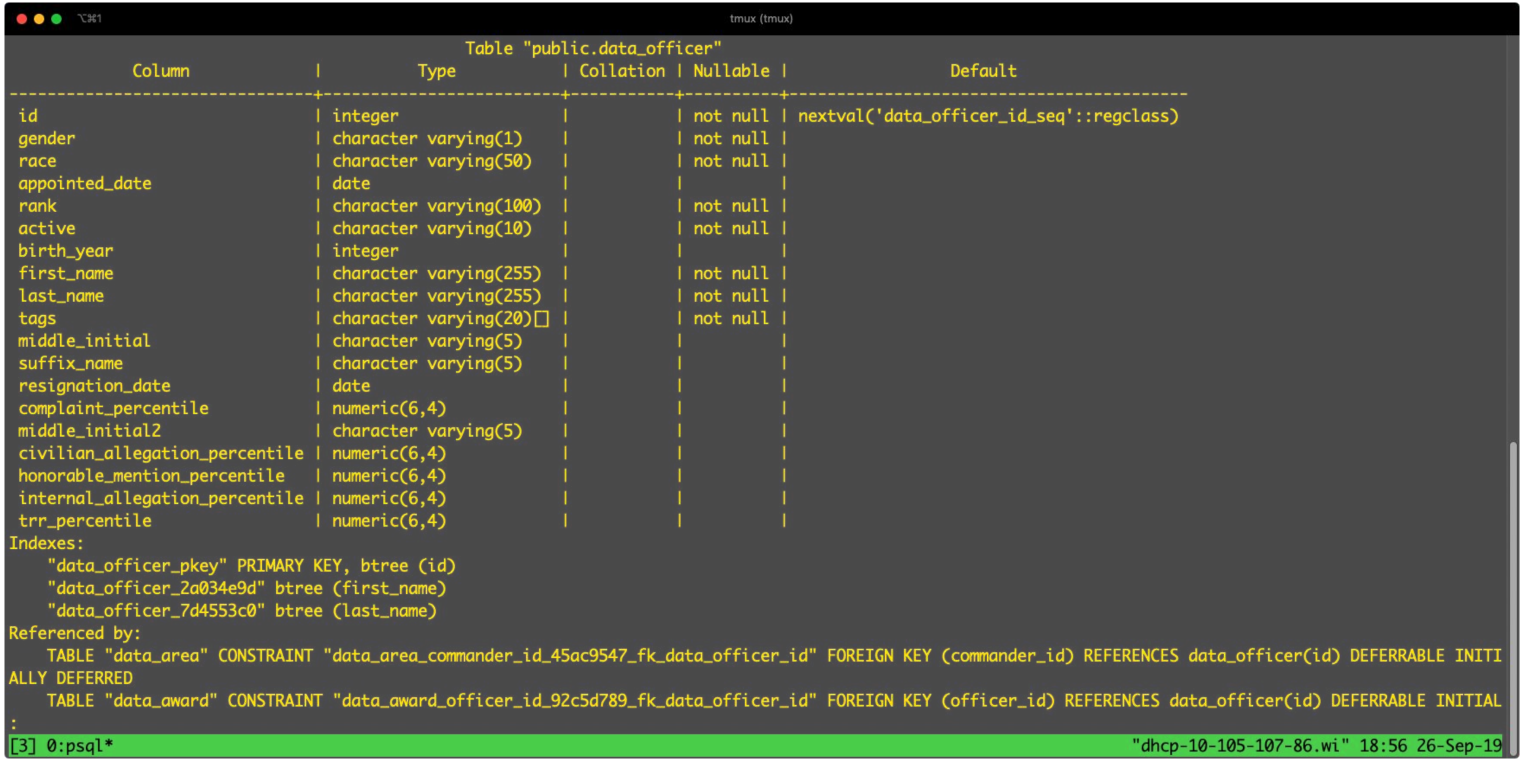
SELECT statement

```
SELECT first_name as first, last_name as last
FROM data_officer
```

WHERE Clause

```
SELECT *
FROM data_officer
WHERE birth_year = 1990;
```

Data Types



```
                                     Table "public.data_officer"
            Column             |         Type          | Collation | Nullable |                 Default
-------------------------------+-----------------------+-----------+----------+------------------------------------------
 id                            | integer               |           | not null | nextval('data_officer_id_seq'::regclass)
 gender                        | character varying(1)  |           | not null |
 race                          | character varying(50) |           | not null |
 appointed_date                | date                  |           |          |
 rank                          | character varying(100)|           | not null |
 active                        | character varying(10) |           | not null |
 birth_year                    | integer               |           |          |
 first_name                    | character varying(255)|           | not null |
 last_name                     | character varying(255)|           | not null |
 tags                          | character varying(20)□|           | not null |
 middle_initial                | character varying(5)  |           |          |
 suffix_name                   | character varying(5)  |           |          |
 resignation_date              | date                  |           |          |
 complaint_percentile          | numeric(6,4)          |           |          |
 middle_initial2               | character varying(5)  |           |          |
 civilian_allegation_percentile | numeric(6,4)         |           |          |
 honorable_mention_percentile  | numeric(6,4)          |           |          |
 internal_allegation_percentile | numeric(6,4)         |           |          |
 trr_percentile                | numeric(6,4)          |           |          |
Indexes:
    "data_officer_pkey" PRIMARY KEY, btree (id)
    "data_officer_2a034e9d" btree (first_name)
    "data_officer_7d4553c0" btree (last_name)
Referenced by:
    TABLE "data_area" CONSTRAINT "data_area_commander_id_45ac9547_fk_data_officer_id" FOREIGN KEY (commander_id) REFERENCES data_officer(id) DEFERRABLE INITI
ALLY DEFERRED
    TABLE "data_award" CONSTRAINT "data_award_officer_id_92c5d789_fk_data_officer_id" FOREIGN KEY (officer_id) REFERENCES data_officer(id) DEFERRABLE INITIAL
:
[3] 0:psql*                                                                                        "dhcp-10-105-107-86.wi" 18:56 26-Sep-19
```

All data types

Text & Strings

- **CHAR(size)** - fixed length string
- **VARCHAR(size)** - varying length string with max size
- **TEXT(size)** - string with maximum size of 65,535 bytes
- **BINARY/VARBINARY(size)** - binary string array with byte size
- **BLOB(size)** - binary large objects. LongBLOBs max is 4GB
  - TINYBLOB
  - MEDIUMBLOB
  - LONGBLOB
- **ENUM(val1, val2, … val*n*)** - String with only 1 value
- **SET(val1, val2, … val*n*)** - String with 0 or more values

```
… WHERE first_name = ‘NAME’
```

Numeric

- **BIT(size)** - bit-value of a certain size.
- **INT(size)**
  - BIGINT
  - MEDIUMINT
  - SMALLINT
  - TINYINT
- **BOOL**
- **FLOAT**
- **DECIMAL(size, d)** - an exact fixed-point decimal of size

```
...WHERE complaint_percentile >= 99.0;
```

Date & Time

- **DATE** - YYYY-MM-DD
- **DATETIME** - YYYY-MM-DD hh:mm:ss
  - DEFAULT - created date
  - ON UPDATE - last modified
- **TIMESTAMP** - stored as number of seconds since unix epoch (01/01/1970)
- **TIME** - hh:mm:ss (can be negative)
- **YEAR** - year in four-digit format

```
...WHERE incident_date = ‘2018-01-31’;
```

SQL Operators

- **Arithmetic**: +, -, *, /, %
- **Bitwise**: &, |, ^
- **Comparison**: =, <, >, <=, >=, <>
- **Compound**: +=, -=, *=, /=
- **Logical**: ALL, AND, ANY, BETWEEN, EXIST, IN, LIKE, NOT, OR, SOME

Where cont.

```
SELECT *
FROM data_officer
WHERE resignation_date = '2016-07-31';
```

Where cont.

```
SELECT *
FROM data_officer
WHERE civilian_allegation_percentile > 99.9
```

And, Not and Or

```sql
SELECT *
FROM data_allegation
WHERE is_officer_complaint = True
AND NOT location = 'Police Building'
AND beat_id NOTNULL
AND (beat_id = 160 OR beat_id = 139);
```

Order By

```sql
SELECT first_name, last_name, birth_year
FROM data_officer
WHERE birth_year NOTNULL
ORDER BY birth_year DESC;
```

Null

```sql
SELECT COUNT(*)
FROM data_award
WHERE ceremony_date IS NULL;
```

Quiz #1 (for candy)

What is the birth year of the youngest officers?

Potential Solution

```sql
SELECT id, first_name, last_name, birth_year,
complaint_percentile
FROM data_officer
WHERE birth_year NOTNULL
   AND active = 'Yes'
   AND complaint_percentile > 0.0
ORDER BY birth_year DESC, complaint_percentile DESC;
```

Min, Max, Count, Avg, Sum

```
SELECT SUM(salary)
FROM data_salary
WHERE year = 2017;
```

More SQL Functions

Like & Wildcards (pattern matching)

PostgreSQL pattern matching

```
SELECT *
FROM data_allegationcategory
WHERE allegation_name LIKE '%Abuse%';
```

In

```sql
SELECT *
FROM data_allegation
WHERE beat_id in (159, 66, 112);
```

Between

```
SELECT *
FROM data_officerallegation
WHERE start_date BETWEEN '2016-01-01' AND '2016-01-31';
```

Group By

```
SELECT location, COUNT(id)
FROM data_allegation
GROUP BY location
```

Quiz #2 (for more candy)

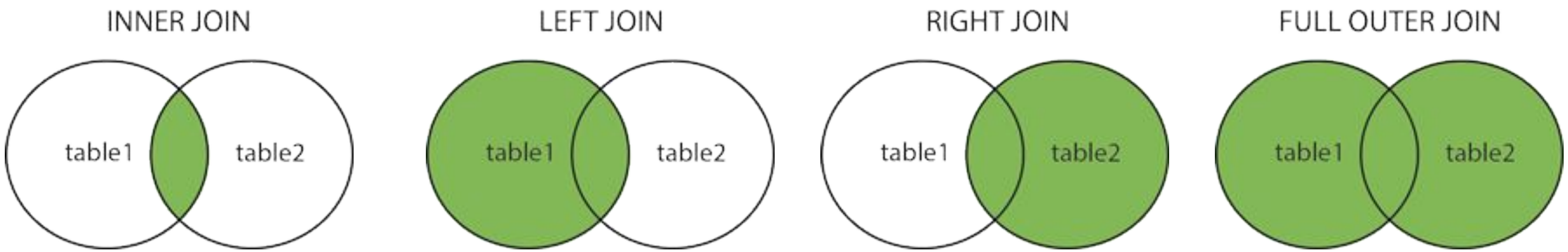
How many allegations were filed in 2004?

Potential Solution

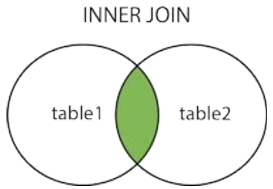
```sql
SELECT COUNT(*)
FROM data_allegation
WHERE incident_date BETWEEN '2004-01-01' AND '2004-12-31';
```

Introduction: Joins

Joins are used to combine rows from two or more tables based on related fields

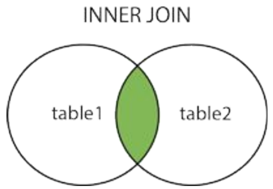


Inner Join



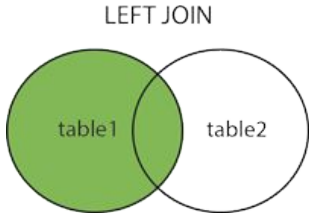
```
-- all officers w/ allegation records
SELECT first_name AS first, last_name AS last
FROM data_officer o
INNER JOIN data_officerallegation a
ON o.id = a.officer_id
```

Inner Join (2) - same result as previous slide



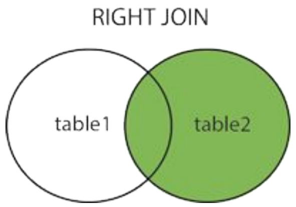
```
-- all officers w/ allegation records
SELECT first_name AS first, last_name AS last
FROM data_officer o, data_officerallegation a
WHERE o.id = a.officer_id
```

Left Join & Group by



```
-- all officers allegation count
SELECT o.id, COUNT(a.id) as allegation_count
FROM data_officer o
LEFT JOIN data_officerallegation a on o.id = a.officer_id
GROUP BY o.id
ORDER BY allegation_count DESC;
```

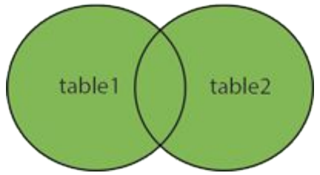
Right Join



```
-- officers allegation count with >=1 allegation
SELECT o.id, COUNT(a.id) as allegation_count
FROM data_officer o
RIGHT JOIN data_officerallegation a on o.id = a.officer_id
GROUP BY o.id
ORDER BY allegation_count DESC;
```

Full Outer Join

FULL OUTER JOIN



```
SELECT o.id as officer_id, tts.id as trr_status_id
FROM data_officer o
FULL OUTER JOIN trr_trrstatus tts on o.id = tts.officer_id
```

Self Join

```sql
SELECT o1.first_name AS officer_1, o2.first_name AS officer_2,
o1.last_name
FROM data_officer o1, data_officer o2
WHERE o1.id <> o2.id
AND o1.last_name = o2.last_name;
```

Quiz #3

Which rank has the lowest average salary? What is the average?

Potential Solution

```
SELECT s.rank, AVG(salary)
FROM data_officer o
INNER JOIN data_salary s
ON s.officer_id = o.id
GROUP BY s.rank
ORDER BY AVG(salary) ASC
```

Union

```
SELECT race
FROM data_officer
UNION
SELECT race
FROM data_complainant
UNION
SELECT race
FROM data_victim
```

Having

```sql
-- locations with over 250 allegations before 2016
SELECT location, COUNT(id)
FROM data_allegation
WHERE NOT location = ''
AND incident_date <= '2015-12-31'
GROUP BY location
HAVING count(id) > 250
ORDER BY count(id) DESC
```

Exists

```sql
-- officers with allegations resulting in suspension
SELECT first_name, last_name
FROM data_officer o
WHERE EXISTS
(
    SELECT id
    FROM data_officerallegation a
    WHERE o.id = a.officer_id
    AND final_outcome LIKE '%Suspen%'
)
```

Any & All

```
SELECT id
FROM data_allegation
WHERE id = ANY
(
    SELECT allegation_id
    FROM data_officerallegation
    WHERE final_outcome LIKE '%Suspen%'
);
```

SQL Database Manipulation

Select Into

```
SELECT *
INTO beat_66_allegations
FROM data_allegation
WHERE beat_id = 66;
```

Views

```
CREATE VIEW beat_66_allegations AS
SELECT * FROM data_allegation
WHERE beat_id = 66;
```

Create Table

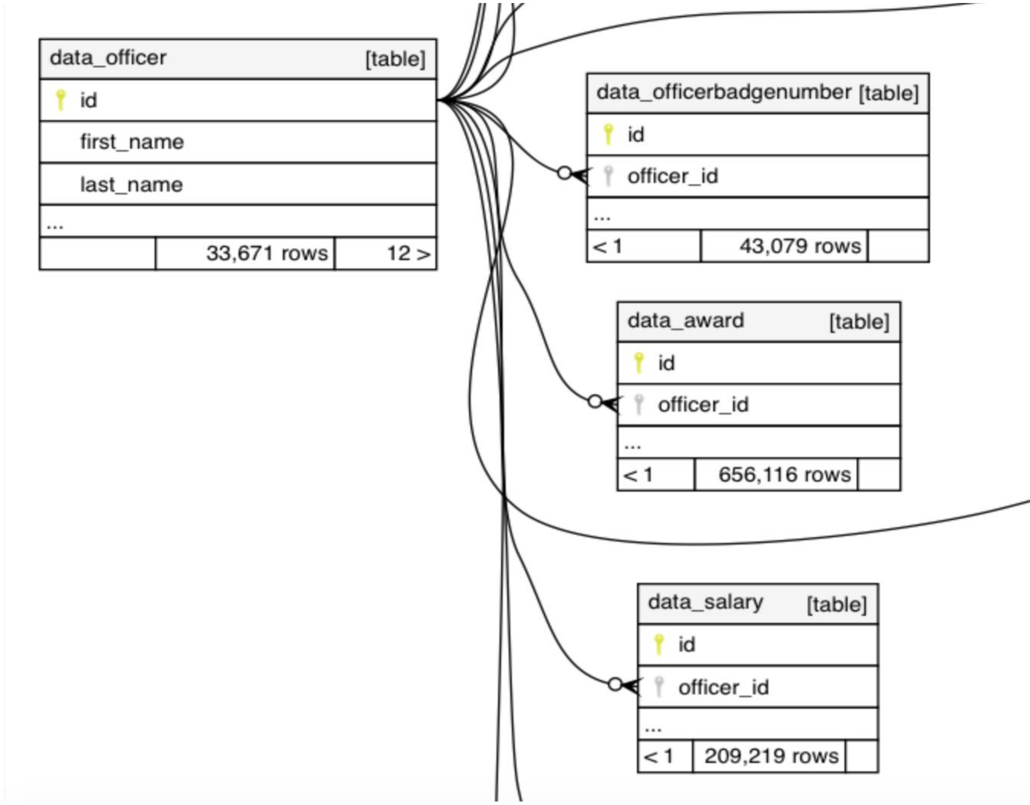
```sql
CREATE TABLE public.data_allegation_areas (
    id integer NOT NULL,
    allegation_id integer NOT NULL,
    area_id integer NOT NULL
);
```

Quick Primer on Keys

- Relations (tables) are sets
  - **No duplicate tuples (rows) within a table** - each one is unique
  - Each tuple must be different in at least one attribute (column)
- Candidate Keys
  - A minimal set of attributes such that no two tuples in the table can have the same value in all those attributes
  - Choose one of the candidate keys to be the **primary key (PK)**
    - *Example:* Social Security Number (SSN) - guaranteed to be unique
      - So PK: (SSN)
    - PK can have multiple attributes: (ID, another_attribute)

Quick Primer on Keys

- **Foreign Keys (FK)**
  - Gold is **primary key**
  - Gray is **foreign key**
- Referential Integrity
  - Every **officer_id** in the *data_award* table must be found in the *data_officer* table in **id** column



Create Table, Primary Keys, and Foreign Keys

```
CREATE TABLE public.data_allegation_areas (
    id integer NOT NULL references public.data_allegation(id),
    allegation_id integer NOT NULL,
    area_id integer NOT NULL,
    PRIMARY KEY id
);
```

Delete Table

```
DROP TABLE public.copa_officer;
```

Insert Data

```
INSERT INTO public.data_allegation_areas (id,
allegation_id, area_id) VALUES (1234, 5678, 9);


INSERT INTO student (id, first_name, last_name, yr)
VALUES (1234, 'Jane', 'Doe', 'Sophomore');
```

Update Tuples

```sql
UPDATE student
SET first_name = 'John', yr = 'Senior'
WHERE id = 1234;
```

Delete Tuples

```sql
DELETE FROM student WHERE id = 1234;
```

Advanced SQL

SQL CASE Statements

```
SELECT id, first_name, last_name,
       CASE
         WHEN complaint_percentile >= 99.0 THEN 'Top 1%'
         WHEN complaint_percentile >= 90.0 THEN 'Top 10%'
         WHEN complaint_percentile >= 75.0 THEN 'Top 25%'
         WHEN complaint_percentile >= 25.0 THEN 'Top 75%'
         ELSE 'Bottom 25%'
       END as allegation_count
FROM data_officer
```

SQL Sequence

The sequence is a special type of data created to generate unique numeric identifiers in the PostgreSQL database.

```
CREATE SEQUENCE sequence_1
START WITH 100
INCREMENT BY 1
MINVALUE 0
MAXVALUE 10000
NOCYCLE;
```

```
INSERT into students
VALUES(sequence_1.nextval,'Blaine');
INSERT into students
VALUES(sequence_1.nextval,'Grant');
INSERT into students
VALUES(sequence_1.nextval,'Sundar');
```

Index

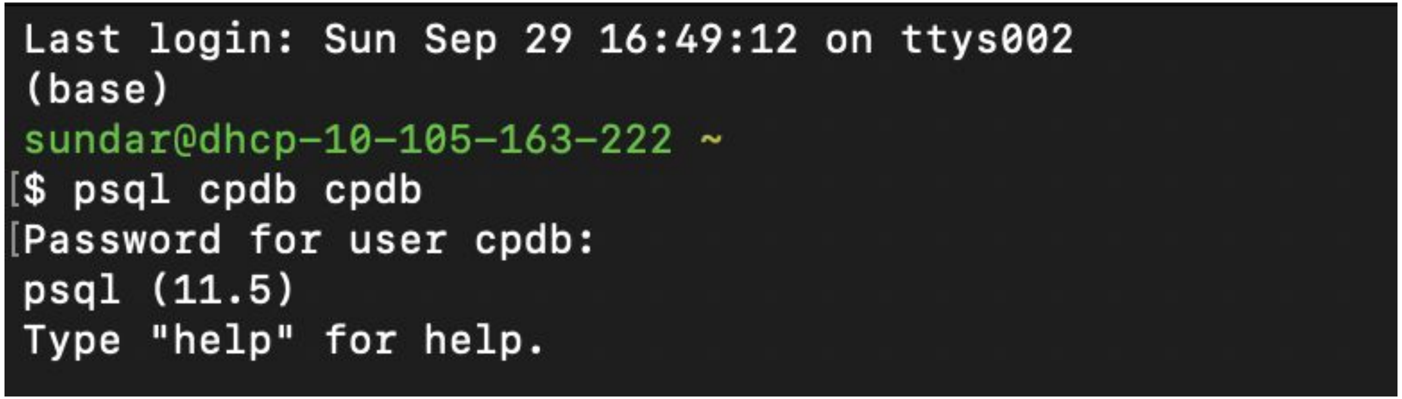
- Indexes are used to retrieve data very fast
- Should only be used when necessary

```sql
CREATE INDEX idx_incident_date
ON data_allegation (incident_date)


DROP INDEX idx_incident_date;
```

Working with SQL and Troubleshooting

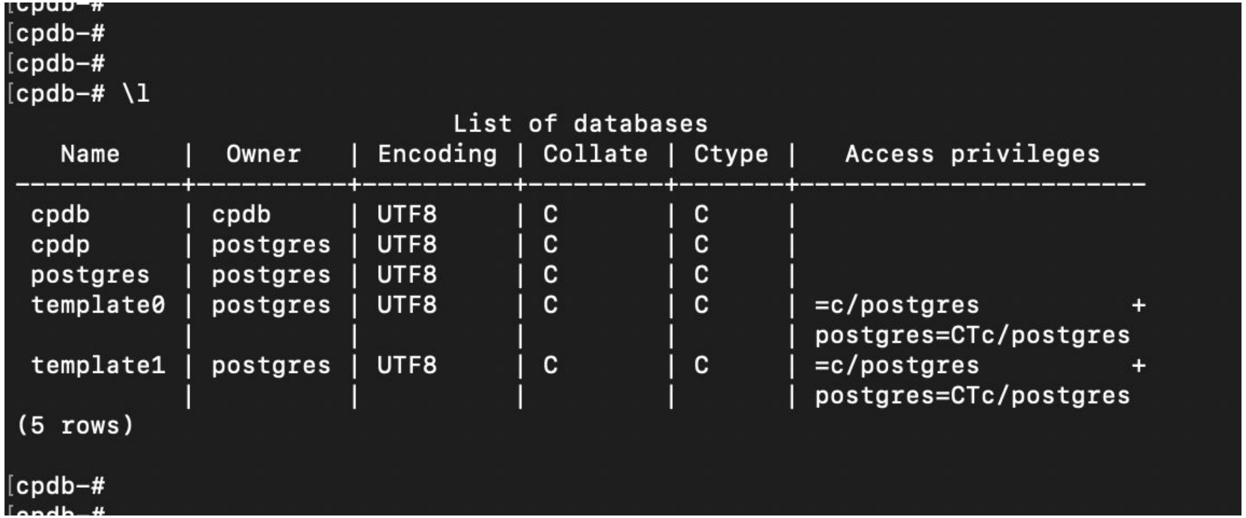
- Connecting to your Database



```
Last login: Sun Sep 29 16:49:12 on ttys002
(base)
sundar@dhcp-10-105-163-222 ~
[$ psql cpdb cpdb
[Password for user cpdb:
psql (11.5)
Type "help" for help.
```

Working with SQL and Troubleshooting

- Show me all my Databases : \l (L in lower case)

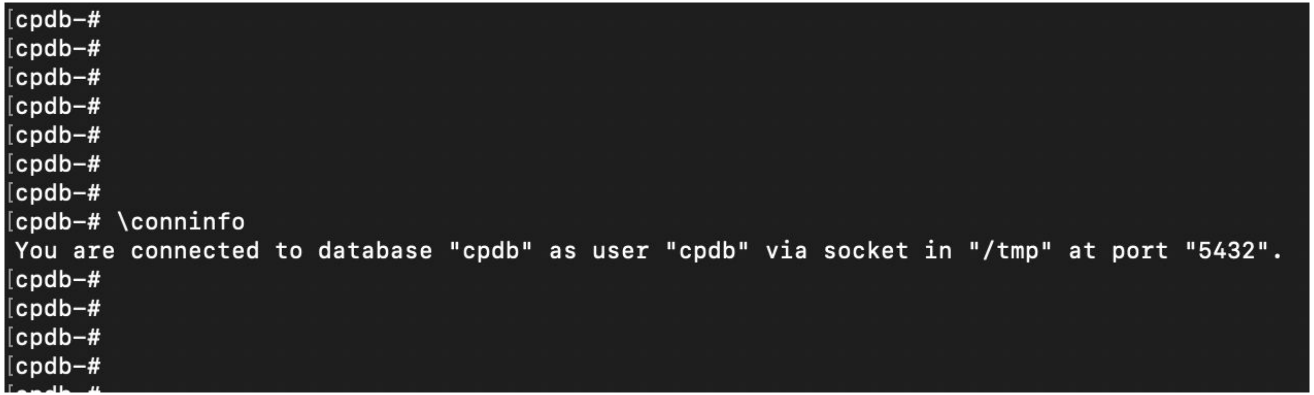


```
[cpdb-#
[cpdb-#
[cpdb-#
[cpdb-# \l
                              List of databases
   Name    |  Owner   | Encoding | Collate | Ctype |   Access privileges
-----------+----------+----------+---------+-------+-----------------------
 cpdb      | cpdb     | UTF8     | C       | C     |
 cpdp      | postgres | UTF8     | C       | C     |
 postgres  | postgres | UTF8     | C       | C     |
 template0 | postgres | UTF8     | C       | C     | =c/postgres          +
           |          |          |         |       | postgres=CTc/postgres
 template1 | postgres | UTF8     | C       | C     | =c/postgres          +
           |          |          |         |       | postgres=CTc/postgres
(5 rows)

[cpdb-#
```

Working with PSQL and Troubleshooting

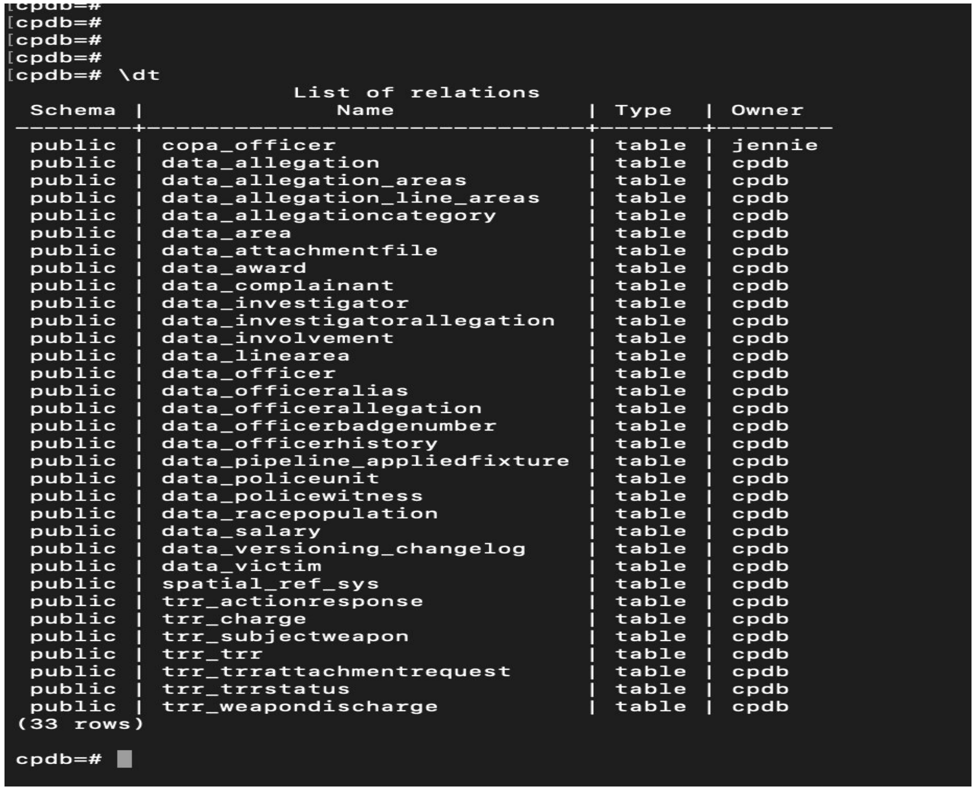
- Which Database am I connected to : \conninfo



```
[cpdb-#
[cpdb-#
[cpdb-#
[cpdb-#
[cpdb-#
[cpdb-#
[cpdb-#
[cpdb-# \conninfo
 You are connected to database "cpdb" as user "cpdb" via socket in "/tmp" at port "5432".
[cpdb-#
[cpdb-#
[cpdb-#
[cpdb-#
```

Working with SQL and Troubleshooting

- Show me all my tables : \dt

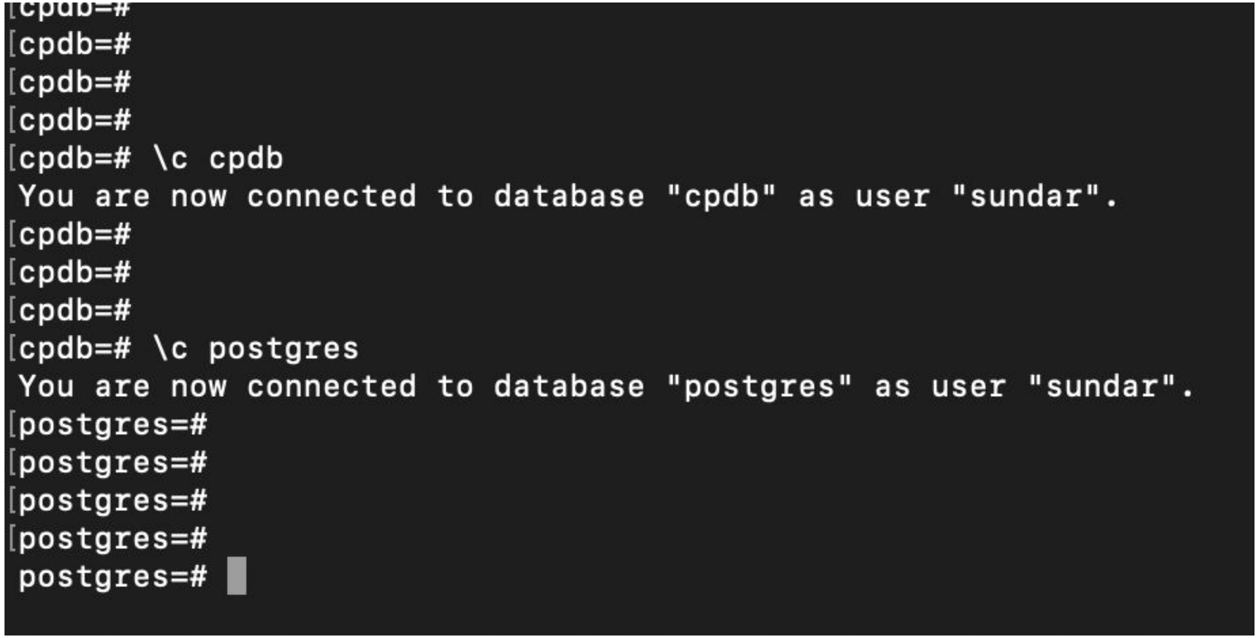


```
[cpdb=#
[cpdb=#
[cpdb=#
[cpdb=#
[cpdb=#  \dt
                         List of relations
 Schema |                Name                | Type  | Owner
--------+------------------------------------+-------+--------
 public | copa_officer                       | table | jennie
 public | data_allegation                    | table | cpdb
 public | data_allegation_areas              | table | cpdb
 public | data_allegation_line_areas         | table | cpdb
 public | data_allegationcategory            | table | cpdb
 public | data_area                          | table | cpdb
 public | data_attachmentfile                | table | cpdb
 public | data_award                         | table | cpdb
 public | data_complainant                   | table | cpdb
 public | data_investigator                  | table | cpdb
 public | data_investigatorallegation        | table | cpdb
 public | data_involvement                   | table | cpdb
 public | data_linearea                      | table | cpdb
 public | data_officer                       | table | cpdb
 public | data_officeralias                  | table | cpdb
 public | data_officerallegation             | table | cpdb
 public | data_officerbadgenumber            | table | cpdb
 public | data_officerhistory                | table | cpdb
 public | data_pipeline_appliedfixture       | table | cpdb
 public | data_policeunit                    | table | cpdb
 public | data_policewitness                 | table | cpdb
 public | data_racepopulation                | table | cpdb
 public | data_salary                        | table | cpdb
 public | data_versioning_changelog          | table | cpdb
 public | data_victim                        | table | cpdb
 public | spatial_ref_sys                    | table | cpdb
 public | trr_actionresponse                 | table | cpdb
 public | trr_charge                         | table | cpdb
 public | trr_subjectweapon                  | table | cpdb
 public | trr_trr                            | table | cpdb
 public | trr_trrattachmentrequest           | table | cpdb
 public | trr_trrstatus                      | table | cpdb
 public | trr_weapondischarge                | table | cpdb
(33 rows)

cpdb=#
```

Working with SQL and Troubleshooting

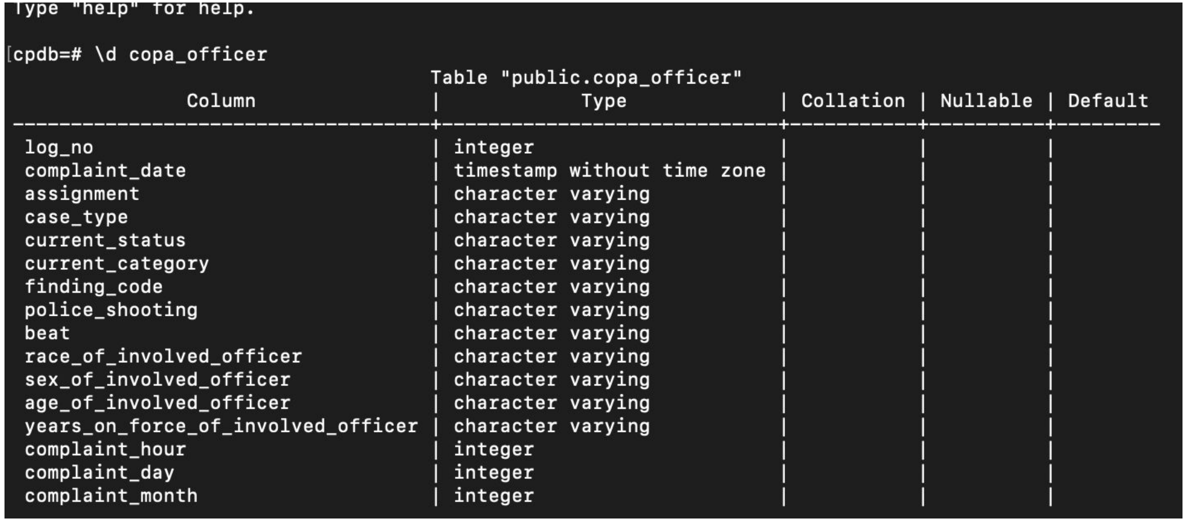
- Switch your Database : \c [new database name]



```
[cpdb=#
[cpdb=#
[cpdb=#
[cpdb=#
[cpdb=# \c cpdb
You are now connected to database "cpdb" as user "sundar".
[cpdb=#
[cpdb=#
[cpdb=#
[cpdb=# \c postgres
You are now connected to database "postgres" as user "sundar".
[postgres=#
[postgres=#
[postgres=#
[postgres=#
postgres=#
```

Working with SQL and Troubleshooting

- Describe your table : \d

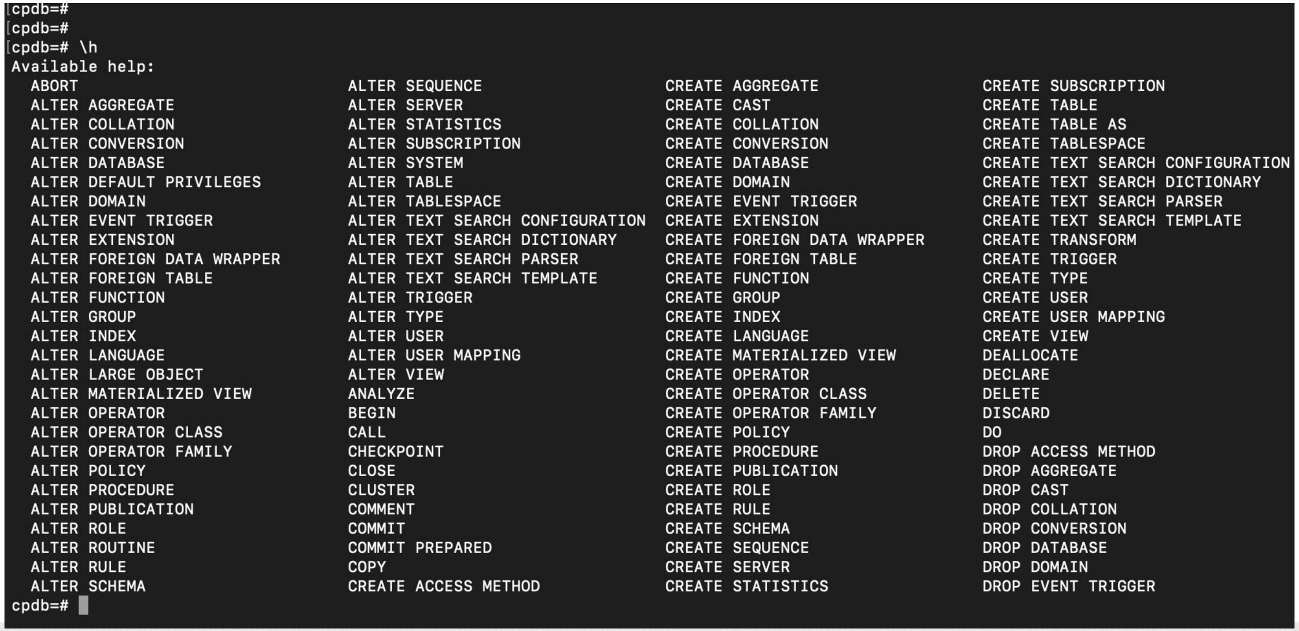


```
Type "help" for help.

[cpdb=# \d copa_officer
                                   Table "public.copa_officer"
               Column                |            Type             | Collation | Nullable | Default
-------------------------------------+-----------------------------+-----------+----------+---------
 log_no                              | integer                     |           |          |
 complaint_date                      | timestamp without time zone |           |          |
 assignment                          | character varying           |           |          |
 case_type                           | character varying           |           |          |
 current_status                      | character varying           |           |          |
 current_category                    | character varying           |           |          |
 finding_code                        | character varying           |           |          |
 police_shooting                     | character varying           |           |          |
 beat                                | character varying           |           |          |
 race_of_involved_officer            | character varying           |           |          |
 sex_of_involved_officer             | character varying           |           |          |
 age_of_involved_officer             | character varying           |           |          |
 years_on_force_of_involved_officer  | character varying           |           |          |
 complaint_hour                      | integer                     |           |          |
 complaint_day                       | integer                     |           |          |
 complaint_month                     | integer                     |           |          |
```

Working with SQL and Troubleshooting

- Get Help : \h



```
cpdb=#
cpdb=#
cpdb=# \h
Available help:
  ABORT                             ALTER SEQUENCE                    CREATE AGGREGATE                  CREATE SUBSCRIPTION
  ALTER AGGREGATE                   ALTER SERVER                      CREATE CAST                       CREATE TABLE
  ALTER COLLATION                   ALTER STATISTICS                  CREATE COLLATION                  CREATE TABLE AS
  ALTER CONVERSION                  ALTER SUBSCRIPTION                CREATE CONVERSION                 CREATE TABLESPACE
  ALTER DATABASE                    ALTER SYSTEM                      CREATE DATABASE                   CREATE TEXT SEARCH CONFIGURATION
  ALTER DEFAULT PRIVILEGES          ALTER TABLE                       CREATE DOMAIN                     CREATE TEXT SEARCH DICTIONARY
  ALTER DOMAIN                      ALTER TABLESPACE                  CREATE EVENT TRIGGER              CREATE TEXT SEARCH PARSER
  ALTER EVENT TRIGGER               ALTER TEXT SEARCH CONFIGURATION   CREATE EXTENSION                  CREATE TEXT SEARCH TEMPLATE
  ALTER EXTENSION                   ALTER TEXT SEARCH DICTIONARY      CREATE FOREIGN DATA WRAPPER       CREATE TRANSFORM
  ALTER FOREIGN DATA WRAPPER        ALTER TEXT SEARCH PARSER          CREATE FOREIGN TABLE              CREATE TRIGGER
  ALTER FOREIGN TABLE               ALTER TEXT SEARCH TEMPLATE        CREATE FUNCTION                   CREATE TYPE
  ALTER FUNCTION                    ALTER TRIGGER                     CREATE GROUP                      CREATE USER
  ALTER GROUP                       ALTER TYPE                        CREATE INDEX                      CREATE USER MAPPING
  ALTER INDEX                       ALTER USER                        CREATE LANGUAGE                   CREATE VIEW
  ALTER LANGUAGE                    ALTER USER MAPPING                CREATE MATERIALIZED VIEW          DEALLOCATE
  ALTER LARGE OBJECT                ALTER VIEW                        CREATE OPERATOR                   DECLARE
  ALTER MATERIALIZED VIEW           ANALYZE                           CREATE OPERATOR CLASS             DELETE
  ALTER OPERATOR                    BEGIN                             CREATE OPERATOR FAMILY            DISCARD
  ALTER OPERATOR CLASS              CALL                              CREATE POLICY                     DO
  ALTER OPERATOR FAMILY             CHECKPOINT                        CREATE PROCEDURE                  DROP ACCESS METHOD
  ALTER POLICY                      CLOSE                             CREATE PUBLICATION                DROP AGGREGATE
  ALTER PROCEDURE                   CLUSTER                           CREATE ROLE                       DROP CAST
  ALTER PUBLICATION                 COMMENT                           CREATE RULE                       DROP COLLATION
  ALTER ROLE                        COMMIT                            CREATE SCHEMA                     DROP CONVERSION
  ALTER ROUTINE                     COMMIT PREPARED                   CREATE SEQUENCE                   DROP DATABASE
  ALTER RULE                        COPY                              CREATE SERVER                     DROP DOMAIN
  ALTER SCHEMA                      CREATE ACCESS METHOD              CREATE STATISTICS                 DROP EVENT TRIGGER
cpdb=#
```

Northwestern | ENGINEERING

Fun with SQL

Some examples on formatting and casting

1) TO_CHAR
2) || operator
3) Distinct
4) Cast
5) COALESCE
6) NULLIF
7) Age(ts,ts) Age(ts)
8) TO_NUMBER(String, Format)

Postgis

- PostgeSQL extension with support for geographic objects

Spatial databases store and manipulate spatial objects like any other object in the database.

The following briefly covers the evolution of spatial databases, and then reviews three aspects that associate *spatial* data with a database – data types, indexes, and functions.

1. **Spatial data types** refer to shapes such as point, line, and polygon;
2. Multi-dimensional **spatial indexing** is used for efficient processing of spatial operations;
3. **Spatial functions**, posed in SQL are for querying of spatial properties and relationships.

Example: Finding the top nth salary

Using DENSE_RANK()

```
SELECT officer_id, salary
FROM (
    SELECT salary, officer_id, DENSE_RANK()
        OVER (ORDER BY salary DESC) as dense_rank
    FROM data_salary
) as d
WHERE d.dense_rank = 11
```