Creating Tables: Takeaways

Syntax

- Returning the description of a table:

```
import psycopg2
conn = psycopg2.connect("dbname=dq user=dq")
cur = conn.cursor()
cur.execute("SELECT * FROM table_name LIMIT 0;")
print(cur.description)
```

- Converting column datatypes:

```
ALTER TABLE table_name
ALTER COLUMN column_name TYPE new_type
USING column_name::new_type;
```

- Creating an enumerated datatype:

```
CREATE TYPE evaluation_enum AS ENUM (
    'Great',       'Mediocre', 'Bad',
    'Good',        'Awful',    'Okay',
    'Masterpiece', 'Amazing',  'Unbearable',
    'Disaster',    'Painful'
);
```

Concepts

- Using data types will save space on the database server which provides faster read and writes. In addition, having proper data types will ensure that any errors in the data will be caught and the data can be queried the way you expect.
- The description property outputs column information from the table. Within the column information, you will find the column data type, name, and other meta information.
- Numeric data types that Postgres supports:

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 point
numeric	variable	user-specified precision, exact	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
real	4 bytes	variable-precision, inexact	6 decimal digits precision

double precision	8 bytes	variable-precision, inexact	15 decimal digits precision
serial	4 bytes	autoincrementing integer	1 to 2147483647
bigserial	8 bytes	large autoincrementing integer	1 to 9223372036854775807

- `REAL`, `DOUBLE PRECISION`, `DECIMAL`, and `NUMERIC` can store float-like numbers such as: 1.23123, 8973.1235, and 100.00.
- The difference between `REAL` and `DOUBLE PRECISION` is that the `REAL` type is up to 4 byes, whereas the `DOUBLE PRECISION` type is up to 8 bytes.
- The `DECIMAL` type works as follows: The precision value, which is the maximum amount of digits before and/or after the decimal point, whereas the **scale** is the maximum amount of digits after the decimal number which must be less than or equal to **precision**.
- The `NUMERIC` and `DECIMAL` types are equivalent in Postgres.
- Corrupted data is unexpected data that has been entered into the data set.
- String-like data types that Postgres supports:

Resources

- The cursor class
- PostgreSQL data types