DATA ENGINEERING:
Data Modeling with Postgres

`psycopg2` MODULE

- psycopg2 is a PostgreSQL database adapter for the Python programming language
- Installation: pip install psycopg2
- Usage: import psycopg2

CONNECTING TO A DATABASE

- conn = psycopg2.connect("host=127.0.0.1 dbname=studentdb user=student password=student")

CREATE A CURSOR OBJECT

cur = conn.cursor()
Note: a cursor object allows execution of PostgreSQL command through Python.

CREATE TABLE

```
create_table = “CREATE TABLE IF NOT EXISTS songs (song_title varchar, artist_name varchar, year int, album_name varchar, single Boolean);”

cur.execute(create_table)
conn.commit()
print("Table created successfully in PostgreSQL")
```

READ OPERATIONS

- Methods: fetchall(), fetchmany(), and fetchone()
- Example:

```
select_query = "SELECT * FROM songs"
cur.execute(select_query)
records = cur.fetchmany(5)
for record in records:
   print(record)
```

INSERT OPERATIONS

```
insert_query = "INSERT INTO customer (customer_id, name, rewards) VALUES (%s, %s, %s)"
data = (1, "Amanda", True)
cur.execute(insert_query, data)
```

ON CONFLICT

- Possible actions: DO NOTHING or DO UPDATE
- Example:

```
INSERT INTO users (id, level) VALUES (1, 0)
ON CONFLICT (id) DO UPDATE
SET level = users.level + 1;
```

UPDATE OPERATIONS

```
update_query = "UPDATE vendors SET vendor_name = %s WHERE vendor_id = %s"
new_data = ("Walmart", 2)
cur.execute(update_query, new_data)
```

DELETE OPERATIONS

```
delete_query = "DELETE FROM vendors WHERE id = %s"
cur.execute(delete_query, (5))
```

SQL MODULE & JUPYTER NOTEBOOK

- Load SQL module: `%load_ext sql`
- Connect to a db: `%sql postgresql://localhost:5432/<db>`
- Execute and return a SQL query in a list of tuples: `%sql SELECT * FROM vendors;`
- Return SQL query in a table with header using Jupyter cell magic: `%%sql`

HANDLING EXCEPTION IN PYTHON

```
try:
    cur = conn.cursor()
except psycopg2.Error as e:
    print("Error: Could not get cursor to the DB")
    print(e)
```

How it works:

- *try* statement will be executed first.
- If there is no exception, *except* statement will be skipped.
- Otherwise, execute the codes in the exception.

TIPS & TRICKS

- Naming convention:
  - SQL keywords: `UPPER CASE`
  - names (identifiers): `lower_case_with_underscores`
  - Example: `UPDATE table SET name = 10;`
- Use triple quotes (“”” “””) or backslash (\) to pass a multi-line query.
- Close db connection as soon as completing a task because connections are limited resources: `conn.close()`
- Set automatic commit to be true so that each action is committed without having to call `conn.commit()` after each command: `conn.set_session(autocommit=True)`

REFERENCES

- psycopg documentation
- PostgreSQL documentation
- Python Database API summary.
- Python errors and exceptions documentation.