**1. 🔌 Importing and Establishing Connection**

* We use the pymysql library to connect to a **MySQL database**.
* Parameters:
  + host="localhost": Running MySQL on the same machine.
  + user="root": The default MySQL user.
  + password="": No password in this case (not recommended in production).
  + db="contacts\_db": Target database.
* conn.cursor() initializes a **cursor object** used to execute SQL queries.

**2. ✅ Checking DB Connection**

* Simple utility to verify that the cursor (and implicitly the connection) is active.
* This can be used in application health checks or startup routines.

**3. 📝 Inserting Contact Form Data**

* Accepts user-submitted data (from a contact form) as parameters.
* Uses a **parameterized SQL query** (prevents SQL injection).
* Commits the transaction to make the insertion permanent.

**🔐 Why this is a good practice:**

* **Parameterized query (%s)** avoids SQL injection attacks.
* **Transaction handling** using conn.commit() ensures atomicity.

**4. ⚠️ Error Handling**

* Catches exceptions such as:
  + Connection failure
  + SQL syntax error
  + Integrity constraints (e.g., unique key violation)
* Simple console message; can be expanded to include logging or custom exception classes.

**Advantages of Using this technique**

* **Security**: You're using parameterized queries instead of concatenating SQL strings.
* **Extensibility**: This function could easily be integrated into a larger Flask or Django app.
* **Maintainability**: Separating DB logic from business logic improves code organization.