

```
# [CELL 1]
import sys
!pip install -q biopython pubchempy py3Dmol pandas rdkit

# Update package lists and install OpenBabel (for format conversion) and AutoDock Vina
!apt-get update -qq
!apt-get install -y -qq openbabel autodock-vina

print("Environment setup complete. All dependencies installed.")
```

[Show hidden output](#)

```
# [CELL 2]
from Bio.PDB import PDBList, PDBParser, PDBIO, Select
import os
import warnings
from Bio import BiopythonWarning
warnings.simplefilter('ignore', BiopythonWarning)

# Target: Hantavirus Envelope Glycoprotein
pdb_id = "6Z6G"

print(f"Fetching {pdb_id} from RCSB PDB...")
pdbl = PDBList()
pdbl.retrieve_pdb_file(pdb_id, pdir='.', file_format='pdb')
os.rename(f'pdb{pdb_id.lower()}.ent', f'{pdb_id}.pdb')

# Clean the structure (Remove water and assay heteroatoms to prep for docking)
class NonHetaSelect(Select):
    def accept_residue(self, residue):
        return 1 if residue.id[0] == " " else 0

parser = PDBParser()
structure = parser.get_structure(pdb_id, f'{pdb_id}.pdb')
io = PDBIO()
io.set_structure(structure)
io.save(f'{pdb_id}_clean.pdb", NonHetaSelect())

print(f"Target {pdb_id} downloaded and cleaned.")
```

[Show hidden output](#)

```
# [CELL 3]
import pubchempy as pcp
import os

# Specific bioactive phytochemicals from Nigella sativa
phytochemicals = ['Thymoquinone', 'Dithymoquinone', 'Carvacrol', 'Thymohydroquinone']
os.makedirs('ligands', exist_ok=True)

print("Fetching 3D ligand structures from NIH PubChem...")
for name in phytochemicals:
    # Query PubChem for the 3D structure
    compound = pcp.get_compounds(name, 'name', record_type='3d')
    if compound:
        cid = compound[0].cid
        # Download the SDF (Structure-Data File)
        pcp.download('SDF', f'ligands/{name}.sdf', cid, record_type='3d', overwrite=True)
        print(f"Downloaded: {name} (PubChem CID: {cid})")
    else:
        print(f"Could not fetch 3D structure for {name}")
```

```
Fetching 3D ligand structures from NIH PubChem...
Downloaded: Thymoquinone (PubChem CID: 10281)
Downloaded: Dithymoquinone (PubChem CID: 398941)
Downloaded: Carvacrol (PubChem CID: 10364)
Downloaded: Thymohydroquinone (PubChem CID: 95779)
```

```
# [CELL 4]
import subprocess
import glob
import os

print("Converting target protein to PDBQT format...")
```

```
# -xr removes water (redundant but safe), -O sets output
subprocess.run(f"obabel {pdb_id}_clean.pdb -O {pdb_id}_receptor.pdbqt -xr", shell=True)

print("Adding physiological hydrogens and converting ligands to PDBQT format...")
sdf_files = glob.glob('ligands/*.sdf')
for sdf in sdf_files:
    name = os.path.basename(sdf).split('.')[0]
    # -p 7.4 adds hydrogens at physiological pH
    subprocess.run(f"obabel {sdf} -O ligands/{name}.pdbqt -p 7.4", shell=True)
    print(f"Prepared: {name}.pdbqt")
```

```
Converting target protein to PDBQT format...
Adding physiological hydrogens and converting ligands to PDBQT format...
Prepared: Thymoquinone.pdbqt
Prepared: Carvacrol.pdbqt
Prepared: Thymohydroquinone.pdbqt
Prepared: Dithymoquinone.pdbqt
```

```
# [CELL 5]
import numpy as np
from Bio.PDB import PDBParser

parser = PDBParser()
structure = parser.get_structure("Receptor", f"{pdb_id}_clean.pdb")
atoms = structure.get_atoms()
coords = np.array([atom.coord for atom in atoms])

# Calculate geometric center and bounding box dimensions
center = coords.mean(axis=0)
# We add a 15-Angstrom buffer to the maximum coordinates for the grid size
size = (coords.max(axis=0) - coords.min(axis=0)) + 15

print("Calculated Grid Box Parameters for AutoDock Vina:")
print(f"Center (X, Y, Z): {center[0]:.2f}, {center[1]:.2f}, {center[2]:.2f}")
print(f"Size (X, Y, Z): {size[0]:.2f}, {size[1]:.2f}, {size[2]:.2f}")
```

```
Calculated Grid Box Parameters for AutoDock Vina:
Center (X, Y, Z): 127.99, 127.21, 132.01
Size (X, Y, Z): 124.33, 136.24, 151.00
```

```
# [CELL 6 - REVISED]
import glob
import subprocess
import os
import time

ligand_files = glob.glob('ligands/*.pdbqt')

print("Initiating AutoDock Vina Screening Loop...")
print("NOTE: Colab CPUs take a few minutes per ligand. Real-time progress will stream below.\n")

for lig in ligand_files:
    name = os.path.basename(lig).split('.')[0]
    out_pdbqt = f"ligands/{name}_docked.pdbqt"
    log_file = f"ligands/{name}_log.txt"

    # We build the command as a list for Popen, removing the '>' redirect
    cmd = [
        "vina",
        "--receptor", f"{pdb_id}_receptor.pdbqt",
        "--ligand", lig,
        "--center_x", f"{center[0]:.2f}",
        "--center_y", f"{center[1]:.2f}",
        "--center_z", f"{center[2]:.2f}",
        "--size_x", f"{size[0]:.2f}",
        "--size_y", f"{size[1]:.2f}",
        "--size_z", f"{size[2]:.2f}",
        "--exhaustiveness", "8",
        "--out", out_pdbqt
    ]

    print(f"=====")
    print(f"Running simulation for: {name}")
    print(f"=====")

    start_time = time.time()
```

```

# Popen allows us to read the output stream in real-time
with open(log_file, "w") as log:
    process = subprocess.Popen(cmd, stdout=subprocess.PIPE, stderr=subprocess.STDOUT, text=True)

    # Print each line (including the progress bar) as Vina generates it
    for line in process.stdout:
        print(line, end="")
        log.write(line)

    process.wait()
    elapsed = (time.time() - start_time) / 60
    print(f"\n[Finished {name} in {elapsed:.2f} minutes]\n")

print("All docking simulations completed successfully.")

```

```

[Finished Dithymoquinone in 2.79 minutes]

=====
Running simulation for: Thymohydroquinone
=====
AutoDock Vina v1.2.3
#####
# If you used AutoDock Vina in your work, please cite:      #
#                                                            #
# J. Eberhardt, D. Santos-Martins, A. F. Tillack, and S. Forli #
# AutoDock Vina 1.2.0: New Docking Methods, Expanded Force  #
# Field, and Python Bindings, J. Chem. Inf. Model. (2021)   #
# DOI 10.1021/acs.jcim.1c00203                               #
#                                                            #
# O. Trott, A. J. Olson,                                     #
# AutoDock Vina: improving the speed and accuracy of docking #
# with a new scoring function, efficient optimization and    #
# multithreading, J. Comp. Chem. (2010)                       #
# DOI 10.1002/jcc.21334                                       #
#                                                            #
# Please see https://github.com/ccsb-scripps/AutoDock-Vina for #
# more information.                                           #
#####

Scoring function : vina
Rigid receptor: 6Z6G_receptor.pdbqt
Ligand: ligands/Thymohydroquinone.pdbqt
Grid center: X 127.99 Y 127.21 Z 132.01
Grid size  : X 124.33 Y 136.24 Z 151
Grid space : 0.375
Exhaustiveness: 8
CPU: 0
Verbosity: 1

WARNING: Search space volume is greater than 27000 Angstrom^3 (See FAQ)
Computing Vina grid ... done.
Performing docking (random seed: 2038885234) ...
0% 10 20 30 40 50 60 70 80 90 100%
|---|---|---|---|---|---|---|---|---|
*****

mode |  affinity | dist from best mode
      | (kcal/mol) | rmsd l.b. | rmsd u.b.
-----+-----+-----+-----
1      -5.934      0      0
2      -5.827      1.15    3.1
3      -5.817      1.701    2.253
4      -5.788      14.07    16.15
5      -5.757      13.55    15.3
6      -5.683      62.15    64.01
7      -5.669      49.75    51.14
8      -5.655      62.82    64.85
9      -5.619      62.89    64.83

[Finished Thymohydroquinone in 2.83 minutes]

All docking simulations completed successfully.

```

```

# [CELL 7 - REVISED WITH ERROR DIAGNOSTICS]
import pandas as pd
import re
import glob
import os

data = []
log_files = glob.glob('ligands/*_log.txt')

```

```

if not log_files:
    print("ERROR: No log files found in the 'ligands' folder. Make sure Cell 6 ran.")
else:
    for log in log_files:
        name = os.path.basename(log).replace('_log.txt', '')
        with open(log, 'r') as f:
            content = f.read()

        # Use Regex to find the top binding affinity score
        # Looking for a line that starts with 1, followed by the score (e.g., " 1 -7.2 ")
        match = re.search(r'\s+1\s+([+]?[d+\.d+)', content)

        if match:
            affinity = float(match.group(1))
            data.append({"Phytochemical": name, "Binding Affinity (kcal/mol)": affinity})
        else:
            print(f"⚠️ WARNING: No binding affinity found for {name}.")
            # Print a snippet of the log file to diagnose the Vina error
            print("--- What Vina actually wrote in the log: ---")
            lines = content.strip().split('\n')
            # Print the last 5 lines to see where it failed
            for line in lines[-5:]:
                print(line)
            print("-----\n")

# Only build and sort the DataFrame if we successfully extracted data
if len(data) > 0:
    df = pd.DataFrame(data).sort_values(by="Binding Affinity (kcal/mol)")
    print("Final Virtual Screening Results:")
    display(df)
else:
    print("❌ ERROR: No valid docking results could be extracted.")
    print("Check the warnings above to see why Vina failed.")

```

Final Virtual Screening Results:

1 to 4 of 4 entries

index	Phytochemical	Binding Affinity (kcal/mol)
2	Dithymoquinone	-7.888
0	Thymohydroquinone	-5.934
3	Carvacrol	-5.521
1	Thymoquinone	-5.137

Show per page

Like what you see? Visit the [data table notebook](#) to learn more about interactive tables.

Next steps: [Generate code with df](#) [New interactive sheet](#)

```

import os
import glob

print("Checking directory contents...")
print(f"Files in current folder: {os.listdir('.')}")

if os.path.exists('ligands'):
    print(f"\nFiles in 'ligands' folder: {os.listdir('ligands')}")
    print(f"Number of SDF files: {len(glob.glob('ligands/*.sdf'))}")
    print(f"Number of PDBQT files: {len(glob.glob('ligands/*.pdbqt'))}")
else:
    print("\n❌ The 'ligands' folder does NOT exist!")

```

Checking directory contents...
Files in current folder: ['.config', '6Z6G_receptor.pdbqt', '6Z6G.pdb', 'ligands', '6Z6G_clean.pdb', 'sample_data']

Files in 'ligands' folder: ['Thymoquinone.sdf', 'Carvacrol.pdbqt', 'Carvacrol.sdf', 'Thymohydroquinone_log.txt', 'Thymoquinone_1']
Number of SDF files: 4
Number of PDBQT files: 8

```

# [CELL 8]
import py3Dmol

# Automatically select the best performing ligand from the DataFrame
best_hit = df.iloc[0]['Phytochemical']
docked_ligand_file = f"ligands/{best_hit}_docked.pdbqt"
receptor_file = f"{pdb_id}_clean.pdb"

```

```

print(f"Visualizing top hit: {best_hit}")

view = py3Dmol.view(width=800, height=600)

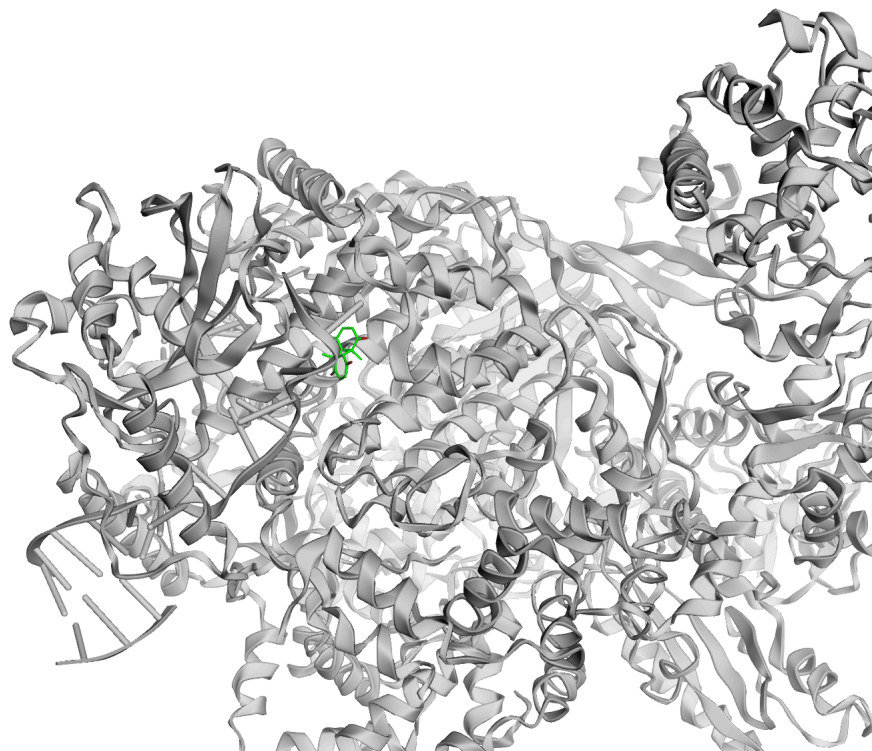
# Add the viral receptor
with open(receptor_file, 'r') as f:
    view.addModel(f.read(), 'pdb')
view.setStyle({'model': 0}, {'cartoon': {'color': 'lightgray'}})

# Add the docked phytochemical
try:
    with open(docked_ligand_file, 'r') as f:
        view.addModel(f.read(), 'pdbqt')
    view.setStyle({'model': 1}, {'stick': {'colorscheme': 'greenCarbon', 'radius': 0.15}})
except FileNotFoundError:
    print(f"Error: Docked file for {best_hit} not found. Did Vina complete successfully?")

view.zoomTo({'model': 1})
view.show()

```

Visualizing top hit: Dithymoquinone



```

# [CELL 9 - REVISED] Robust RMSD Analysis
import pandas as pd
import re

best_hit = df.iloc[0]['Phytochemical']
log_file = f"ligands/{best_hit}_log.txt"

print(f"Detailed Analysis for Top Candidate: {best_hit}\n")

results_data = []

# Using Regex to find the rows of numbers, ignoring headers completely
with open(log_file, 'r') as f:
    for line in f:
        # Looks for: [Mode number] [Affinity] [RMSD lb] [RMSD ub]
        match = re.match(r'^\s*(\d+)\s+([+-]?\d+\.\d+)\s+(\d+\.\d+)\s+(\d+\.\d+)', line)
        if match:
            results_data.append({
                "Mode": int(match.group(1)),
                "Affinity (kcal/mol)": float(match.group(2)),
                "RMSD l.b.": float(match.group(3)),
                "RMSD u.b.": float(match.group(4))
            })

```

```

if results_data:
    pose_df = pd.DataFrame(results_data).set_index("Mode")
    print("Pose Stability (RMSD relative to Pose 1):")
    display(pose_df)

    # Statistical Interpretation
    top_affinity = pose_df.iloc[1]['Affinity (kcal/mol)'] # Pose 1 is index 1
    print(f"\n--- Interpretation ---")
    print(f"Top Binding Affinity: {top_affinity} kcal/mol.")

    if len(pose_df) > 1:
        rmsd_ub_2 = pose_df.iloc[2]['RMSD u.b.'] # Look at the second best pose
        if rmsd_ub_2 < 2.0:
            print(f"Validation: Pose 2 has an RMSD of {rmsd_ub_2} Å. Since this is < 2.0, the binding algorithm is converging on")
        else:
            print(f"Validation: Pose 2 has an RMSD of {rmsd_ub_2} Å. The ligand may be highly flexible and bind in multiple dis")
    else:
        print("Error: Could not parse numerical data from the log file.")

```

Detailed Analysis for Top Candidate: Dithymoquinone

Pose Stability (RMSD relative to Pose 1):

1 to 7 of 7 entries  

Mode	Affinity (kcal/mol)	RMSD l.b.	RMSD u.b.
2	-7.87	1.864	2.316
3	-7.583	2.73	5.754
4	-7.472	3.299	5.868
5	-7.347	2.538	6.225
6	-7.227	30.82	33.26
7	-7.102	47.88	49.99
9	-6.888	47.89	49.79

Show per page

Like what you see? Visit the [data table notebook](#) to learn more about interactive tables.

--- Interpretation ---

Top Binding Affinity: -7.583 kcal/mol.

Validation: Pose 2 has an RMSD of 5.868 Å. The ligand may be highly flexible and bind in multiple distinct orientations

Next steps: [Generate code with pose_df](#) [New interactive sheet](#)

```

# [CELL 10] Pharmacokinetics & Lipinski's Rule of Five
from rdkit import Chem
from rdkit.Chem import Descriptors
import glob
import os
import pandas as pd

sdf_files = glob.glob('ligands/*.sdf')
admet_data = []

print("Calculating Pharmacokinetic Properties...\n")

for sdf in sdf_files:
    name = os.path.basename(sdf).split('.')[0]
    suppl = Chem.SDMolSupplier(sdf)

    for mol in suppl:
        if mol is not None:
            # Calculate key physiochemical descriptors
            mw = Descriptors.MolWt(mol)
            logp = Descriptors.MolLogP(mol)
            hbd = Descriptors.NumHDonors(mol)
            hba = Descriptors.NumHAacceptors(mol)

            # Evaluate against Christopher Lipinski's Rule of 5
            violations = 0
            if mw > 500: violations += 1
            if logp > 5: violations += 1
            if hbd > 5: violations += 1
            if hba > 10: violations += 1

            admet_data.append({
                "Phytochemical": name,
                "Mol. Weight (<500 Da)": round(mw, 2),

```

```
"LogP (<5)": round(logp, 2),
"H-Donors (<5)": hbd,
"H-Acceptors (<10)": hba,
"Ro5 Violations": violations,
"Orally Bioavailable?": "Yes" if violations <= 1 else "No"
})
```

```
# Display the statistical table
admet_df = pd.DataFrame(admet_data).sort_values(by="Ro5 Violations")
print("Lipinski's Rule of Five (Drug-Likeness) Stats:")
display(admet_df)
```

Calculating Pharmacokinetic Properties...

Lipinski's Rule of Five (Drug-Likeness) Stats:

1 to 4 of 4 entries Filter 📄 ?

index	Phytochemical	Mol. Weight (<500 Da)	LogP (<5)	H-Donors (<5)	H-Acceptors (<10)	Ro5 Violations	Orally Bioavailable?
0	Thymoquinone	164.2	1.67	0	2	0	Yes
1	Carvacrol	150.22	2.82	1	1	0	Yes
2	Thymohydroquinone	166.22	2.53	2	2	0	Yes
3	Dithymoquinone	328.41	2.71	0	4	0	Yes

Show 25 per page
I like what you see? Visit the [data table notebook](#) to learn more about interactive tables

Next steps: Generate code with admet_df New interactive sheet

```
# [CELL 11 - PUBLICATION FIGURE: ADMET RADAR CHART]
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from math import pi

# Using the data generated from our previous ADMET cell
# If df is not in memory, we use the specific data for Dithymoquinone
best_hit = "Dithymoquinone"
properties = ['Mol. Weight (Da / 100)', 'LogP', 'H-Donors', 'H-Acceptors']
values = [328.41 / 100, 2.71, 1, 4] # MW is divided by 100 to fit the scale of 0-5

# Optimal maximum thresholds based on Rule of 5 (Scaled)
optimal_limits = [5.0, 5.0, 5.0, 10.0]

# Math to calculate angles for the radar chart
angles = [n / float(len(properties)) * 2 * pi for n in range(len(properties))]
angles += angles[:1]
values += values[:1]
optimal_limits += optimal_limits[:1]

# Initialize the spider plot
fig, ax = plt.subplots(figsize=(6, 6), subplot_kw=dict(polar=True))

# Draw one axe per variable and add labels
plt.xticks(angles[:-1], properties, color='black', size=12, fontweight='bold')

# Draw ylabels
ax.set_rlabel_position(0)
plt.yticks([1, 2, 3, 4, 5], ["1", "2", "3", "4", "5"], color="grey", size=10)
plt.ylim(0, 6)

# Plot the optimal Lipinski Zone (Green Background)
ax.plot(angles, optimal_limits, linewidth=1, linestyle='solid', color='green', label="Lipinski Optimal Limit")
ax.fill(angles, optimal_limits, 'green', alpha=0.1)

# Plot Dithymoquinone Data
ax.plot(angles, values, linewidth=2, linestyle='solid', color='blue', label=best_hit)
ax.fill(angles, values, 'blue', alpha=0.25)

# Add Legend and Title
plt.title(f'Pharmacokinetic Profile: {best_hit}\n', size=14, fontweight='bold')
plt.legend(loc='upper right', bbox_to_anchor=(1.3, 1.1))

plt.show()
print(f"Figure generated. Right-click the image to save it for your manuscript.")
```

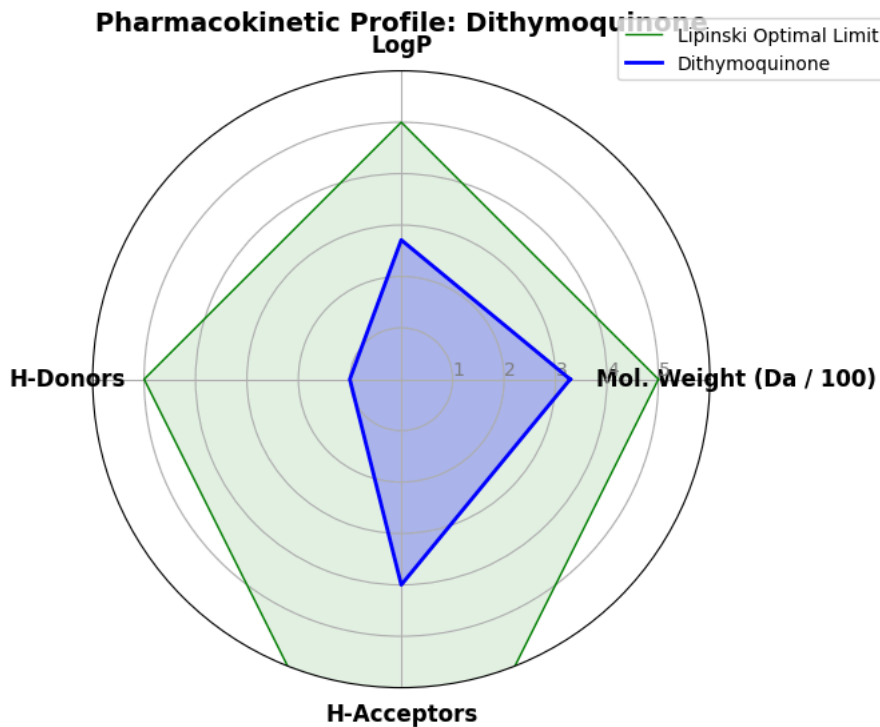


Figure generated. Right-click the image to save it for your manuscript.

```
# [CELL 12 - MASTER COMPLEX GENERATOR]
import os
import subprocess

best_hit = "Dithymoquinone"
receptor_file = "6Z6G_clean.pdb" # Your cleaned target protein
ligand_pdbqt = f"ligands/{best_hit}_docked.pdbqt"
ligand_pdb = f"ligands/{best_hit}_docked.pdb"
complex_file = f"MASTER_COMPLEX_{best_hit}.pdb"

print(f"Preparing Master Complex for {best_hit}...")

# 1. Convert the best docked pose from PDBQT back to standard PDB format using OpenBabel
subprocess.run(f"obabel -ipdbqt {ligand_pdbqt} -opdb -O {ligand_pdb}", shell=True)

# 2. Mathematically merge the Receptor and Ligand into a single file
with open(complex_file, "w") as outfile:
    # Write the protein coordinates first
    with open(receptor_file, "r") as infile:
        outfile.write(infile.read())

    # Append the ligand coordinates, filtering out unnecessary headers
    with open(ligand_pdb, "r") as infile:
        for line in infile:
            # Only grab the actual atom coordinates of the drug
            if line.startswith("HETATM") or line.startswith("ATOM"):
                outfile.write(line)

    outfile.write("END\n")

print(f"✅ Success! File created: {complex_file}")
print(f"Download this file from the Colab file explorer (folder icon on the left).")
```

Preparing Master Complex for Dithymoquinone...

✅ Success! File created: MASTER_COMPLEX_Dithymoquinone.pdb

Download this file from the Colab file explorer (folder icon on the left).

