Below is a **master plan** for the notebook. The idea is to create **one single Jupyter Notebook** that proceeds in a **logical, top-to-bottom** workflow. Each cell's **title**, **purpose**, **inputs**, **outputs**, and the **interpretations** (plus any validations, logging, or cross-checks) are noted in detail. By following this blueprint, we'll create a single integrated pipeline—capable of reading data, computing mass loss, fitting Sersic profiles, diagnosing morphology, and generating outputs that allow us to interpret the astrophysical evolution of M33 in a peer-review-ready format.

Notebook Layout:

0. Notebook Title & Introductory Markdown

Cell 0.1 — "Notebook Overview & Goals"

Type: Markdown (no code)
Purpose:

- Brief textual overview of what this notebook does: reading M31/M33 snapshots, computing mass loss, fitting surface density profiles, analyzing morphology, etc.
- Summarizes the final aims: produce timeseries of M33's bound mass, morphological diagnostics, and compare them to theoretical models.

Inputs: None (purely textual).
Outputs: None (purely textual).
Interpretation: Gives the big-picture rationale and sets the stage.

1. Imports, Logging Setup, and Global Parameters

Cell 1.1 — "Imports & Logging Configuration"

Type: Code
Purpose:

- Import all Python libraries (NumPy, Matplotlib, SciPy, possibly Astropy).
- Set up any custom logger, e.g., Python's `logging` module, to track important events (e.g., success/failure of snapshot reads, convergence of fits).
- Possibly define global constants (e.g., `G = 4.499e-6 ...`, scale radii for M31, etc.).

Inputs: None (just library imports).
Outputs: None (just library references in memory).
Interpretation: Confirms we have everything needed. Logging ensures that as we move on, we can store warnings or errors in a file or on the console.

Cell 1.2 — "User-Configurable Parameters & Filenames"

Type: Code
Purpose:

- Define key parameters like M31's total halo mass, scale radius, the directories for M31/M33 snapshot files, any truncation radius for M33 bound mass, etc.
- Possibly a dictionary or set of global variables for, e.g., `router_limit = 30.0`, `nbins_for_profiles = 30`.

Inputs: None (manually set by the user).
Outputs: Variables stored in memory for later cells.
Interpretation: All "tweakable" parameters in one place to make the pipeline flexible.

2. Data Parsing & Basic Utilities

Cell 2.1 — "File Parsing Functions"

Type: Code
Purpose:

- Define `parse_galaxy_file(filename)` or equivalents to read `M31_XXX.txt`, `M33_XXX.txt`.
- Provide docstrings explaining columns (type, mass, x, y, z, vx, vy, vz).
- Possibly define a helper to extract snapshot index from filename (like `_000.txt`).

Inputs: N/A (function definitions only).
Outputs: Functions in memory.
Interpretation: Lays foundation for reading in data from the disk.

Cell 2.2 — "Center of Mass & Other Shared Math Routines"

Type: Code
Purpose:

- Define `center_of_mass(x, y, z, mass)`, `hernquist_enclosed_mass(r, M_halo, a)`, etc.
- Possibly define any small math utilities (e.g., coordinate transforms).
- Provide docstrings for each.

Inputs: N/A (function definitions).
Outputs: Functions in memory.
Interpretation: Consolidates common operations so the rest of the notebook can reuse them.

Cell 2.3 — "Diagnostic Test: Single Snapshot Parsing & Printout"

Type: Code
Purpose:

- As a **mini-test**: parse one M31 file and one M33 file (e.g. the earliest snapshot).
- Print the time, total particle counts, and center-of-mass to confirm correctness.
- Possibly log the results with the logging system.

Inputs: `M31_000.txt`, `M33_000.txt`
Outputs:

- Print statements or log lines confirming the data read.
- Quick numeric results (COM coordinates, total masses).

Interpretation: Quick check that the file reading logic is correct before we scale up.

3. Mass Loss Computations

3.1 Jacobi Radius Over Time

Cell 3.1.1 — "Compute & Store Jacobi Radius for All Snapshots" **Type:** Code
Purpose:

- Loop over `M31/*.txt` & `M33/*.txt` for each snapshot index.
- For each snapshot: read in M31 & M33, compute their COM, relative distance, enclosed Mhost (Hernquist), total M33 mass, then Jacobi radius.

- Save `(time, R_j)` in arrays.
- Validate with logging (if `R_j` is 0 or unreasonably large, warn the user).

Inputs: snapshot files from M31 and M33 directories.
Outputs: A 2-column array [time, R_{jacobi}]. Also stored in memory for further plotting. Possibly save to `.npy` or `.txt`.
Interpretation: Provides a time evolution of the Jacobi radius. We'll see how close or how large M33's tidal radius is throughout the simulation.

Cell 3.1.2 — "Jacobi Radius vs. Time: Plots & Analysis" Type: Code
Purpose:

- Takes the array `(time, R_j)`, creates a line plot or scatter plot of `R_j` vs. `time`.
- Possibly fit a polynomial or smoothing spline for interpretive clarity.
- Show a textual commentary about how the Jacobi radius evolves (e.g., "We see a sharp decline at ~ 5 Gyr, indicating stronger tides.").

Inputs: The `(time, R_j)` array from the previous cell.
Outputs: A figure (saved to `plots/jacobi_radius_over_time.png`).
Interpretation: User sees how the environment's tidal limit for M33 changes with time, presumably correlated with M33's orbital position.

3.2 Bound Mass Over Time

Cell 3.2.1 — "Compute & Store M33 Bound Mass for All Snapshots" Type: Code
Purpose:

- For each snapshot, we already have `R_j`. We take `R_cut = min(R_j, router_limit)`.
- Shift M33 by its COM, select particles within `R_cut`, sum their masses $\Rightarrow M_{\text{bound}}(t)$.
- Save `(time, Mbound)`.

Inputs: M33 snapshot files, plus the array of `R_j` from above.
Outputs: `(time, Mbound)`, stored in memory and saved to `.npy` or `.txt`.
Interpretation: Provides the time evolution of M33's stellar/gas mass (depending on which we track). We'll see how much mass is lost, presumably from tidal stripping.

Cell 3.2.2 — "Bound Mass Timeseries Plots & Discussion" Type: Code
Purpose:

- Plot `(time, Mbound)` with a line or scatter plot. Possibly overlay events like pericenter passages.
- Print some summary stats, e.g., "Initial mass = X Msun, final mass = Y Msun => Z% lost."
- Provide textual commentary.

Inputs: `(time, Mbound)` array.
Outputs: A figure (saved, e.g., `plots/M33_bound_mass_vs_time.png`) plus textual conclusions in print statements.
Interpretation: Tells us the net mass-loss rate and key epochs of strong stripping.

4. Sersic/Exponential Fitting of the Stellar Disk

4.1 Generate Surface Density Profiles

Cell 4.1.1 — "Surface Density Profile Function & Single-Snapshot Demo" **Type:** Code
Purpose:

- Define `surface_density_profile()` that takes positions, masses, and returns `(r_mid, Sigma)`.
- Demonstrate on, say, `M33_000.txt`: parse, center on M33, compute `(r_mid, Sigma)`, quick plot.

Inputs: One snapshot file for M33.
Outputs:

- `(r_mid, Sigma)` arrays.
- A figure verifying the radial distribution. Possibly *log–log* to see the outer slope.

Interpretation: Quick check that we can form a correct radial profile for a single snapshot.

Cell 4.1.2 — "Surface Density Profiles for All Snapshots" **Type:** Code
Purpose:

- Loop over M33 snapshots, do the same routine, store results in a dictionary or list: `profiles[snap] = (time, r_mid, Sigma)`.
- Possibly store each `(r_mid, Sigma)` in a `.npy` or `.csv`.
- Provide inline logs if any snapshot yields suspiciously large or small values.

Inputs: All M33 snapshot files.
Outputs: A dictionary `profiles` in memory, plus optional saved data.
Interpretation: We get the entire time evolution of M33's surface density, snapshot by snapshot.

4.2 Sersic/Exponential Fits to Each Snapshot

Cell 4.2.1 — "Define Fitting Routines" **Type:** Code
Purpose:

- Functions like `sersic_profile(R, Sigma_e, Re, n)`, `exponential_profile(R, Sigma0, h)`, plus the `fit_sersic(...)` and `fit_exponential(...)` using `scipy.optimize.curve_fit`.
- Possibly define a separate function for computing sersic `b_n` or do it inline.

Inputs: None (function definitions).
Outputs: Code in memory.
Interpretation: Core code for fitting radial profiles.

Cell 4.2.2 — "Run Fits for All Snapshots & Store Best-Fit Parameters" **Type:** Code
Purpose:

- For each snapshot in `profiles`, do a Sersic fit **and** an exponential fit, gather `(time, Sigma_e, Re, n)` or `(time, Sigma0, h)`.
- Store in arrays/dicts `sersic_fits[snap] = (...)`, `exp_fits[snap] = (...)`.
- Provide logs for convergence or warnings if a fit fails.

Inputs: The dictionary `profiles` from above.
Outputs: Fit parameters in memory or saved to `.npy`.
Interpretation: We see how M33's radial structure evolves, e.g., changes in the Sersic index over time might indicate morphological transformation.

Cell 4.2.3 — "Timeseries of Fit Parameters: Re, n, Scale Length, etc." **Type:** Code
Purpose:

- Generate multiple plots: e.g., (time vs. Re) for the Sersic fit, (time vs. n), or for the exponential fit (time vs. h).
- Possibly show them side-by-side.
- Summaries: "At ~ 5 Gyr, Re drops significantly, indicating tidal truncation."

Inputs: `sersic_fits`, `exp_fits`.
Outputs: Plots and textual commentary.
Interpretation: Tells us how the disk scale length or bulge–disk shape evolves with time.

4.3 Checking for Tidal Truncation

Cell 4.3.1 — "Slope-based Tidal Truncation Analysis" **Type:** Code
Purpose:

- For each snapshot's `(r_mid, Sigma)`, compute the log–log slope of the outer profile.
- Identify a "truncation radius" if slope < e.g. -4 or -5.
- Store `(time, R_trunc)`.

Inputs: The `profiles` dictionary.
Outputs: An array of truncation radii over time, plus any flagged snapshots where no truncation was found.
Interpretation: Confirms the outer disk is being tidally stripped or truncated.

Cell 4.3.2 — "Plot Tidal Truncation Radius vs. Time" **Type:** Code
Purpose:

- Plot the `(time, R_trunc)` timeseries. Possibly overlay the Jacobi radius from Section 3.1 to see correlation.
- Provide commentary.

Inputs: `(time, R_trunc)`, plus `(time, R_j)` from the Jacobi analysis.
Outputs: A combined figure.
Interpretation: Helps us see if the tidal truncation in the disk surface density aligns with the theoretical Jacobi boundary.

5. Morphological Evolution

5.1 Face-On & Edge-On Views Over Time

Cell 5.1.1 — "Rotation + 2D Histograms: Single Snapshot Demo" **Type:** Code
Purpose:

- Demonstrate how we pick the M33 disk particles, compute the angular momentum vector, rotate to align the disk with the z-axis, and produce face-on & edge-on histograms.
- Plot them with `matplotlib.pyplot.hist2d()` or `density_contour()`.

Inputs: One M33 snapshot file.
Outputs:

- Two Figures: face-on vs. edge-on distribution. Possibly saved.
- Quick commentary.

Interpretation: Visual check of the disk structure for that snapshot.

Cell 5.1.2 — "Automated Morphology Plots for All Snapshots" **Type:** Code

Purpose:

- Loop over all snapshots, do the same rotation, generate face-on/edge-on plots, and **save** them to `plots/M33_faceon_<snap>.png`, `plots/M33_edgeon_<snap>.png`.
- Possibly run headless (i.e., no real-time display, just saving files).

Inputs: All M33 snapshot files.
Outputs: A folder of face-on/edge-on images.
Interpretation: We can flip through them to see morphological changes across cosmic time.

5.2 Spiral Arms & Pitch Angle

Cell 5.2.1 — "Spiral Arm Identification & Pitch Angle Function" **Type:** Code

Purpose:

- Implementation of something like `measure_spiral_pitch_angle(pos_rot, rmin, rmax)`.
- Possibly does a naive approach of fitting $\theta = c + m \ln(R)$ or a more robust approach if available.

Inputs: N/A (function definition).
Outputs: Function in memory.
Interpretation: Core method for analyzing spiral structure.

Cell 5.2.2 — "Spiral Arm Timeseries: Pitch Angle vs. Time" **Type:** Code

Purpose:

- For each snapshot's disk, after rotation, measure pitch angle.
- Plot `(time, pitch_angle)`. Possibly print "No spiral found" if the disk is too disturbed.

Inputs: All M33 snapshot files.
Outputs:

- Timeseries table + a plot (pitch angle vs. time).
- Possibly a separate figure to show a single snapshot's (R, θ) diagram with the best-fit line.

Interpretation: Tells us if spiral arms get stronger/weaker or more/less tightly wound over time.

5.3 Warp & Scale Height Evolution

Cell 5.3.1 — "Warp Computation vs. Radius" **Type:** Code

Purpose:

- After rotation, subdivide the disk radially, measure the local plane normal in each annulus, compare to the inner disk normal $\Rightarrow$ warp angle.
- Possibly store `(time, radius, warp_angle)`.

Inputs: For each snapshot, the post-rotation `(x_rot, y_rot, z_rot)`.
Outputs: A dictionary or array with warp angles at multiple radii, for each time.
Interpretation: We see if the disk is "bending" or "twisting" more significantly at certain epochs.

Cell 5.3.2 — "Scale Height Computation" **Type:** Code

Purpose:

- For each radial bin, measure the **vertical** distribution (z-dispersion).
- Possibly store `(time, radius, scale_height)`.

Inputs: The same rotated coordinates.
Outputs: Arrays saved or stored for subsequent plots.
Interpretation: We see if the disk thickens over time.

Cell 5.3.3 — "Warp & Height Plots & Discussion" **Type:** Code

Purpose:

- Summarize the warp angle vs. radius at different snapshots. Possibly an overlay of 2–3 snapshots to see changes.
- Summarize the scale height radial profile over time.
- Provide textual commentary.

Inputs: Warp & height arrays.
Outputs: Figures (e.g., multiple lines on a single plot), and text analysis.
Interpretation: A direct morphological signature of tidal interactions or collisions.

6. Integration & Final Summaries

Cell 6.1 — "Cross-Correlation Plots: Mass Loss vs. Morphology"

Type: Code

Purpose:

- Possibly plot M33 bound mass vs. the pitch angle or vs. warp amplitude to see if strong mass stripping correlates with morphological transformations.
- Print correlation coefficients or relevant statistics.

Inputs: `(time, Mbound)` from Section 3, `(time, pitch_angle)`, `(time, warp_angle)` from Section 5.
Outputs: Overlaid or side-by-side plots.
Interpretation: Helps answer the question: "Does a big mass-loss event coincide with more severe warping or a change in spiral structure?"

Cell 6.2 — "Final Data Tables & Logging Info"

Type: Code

Purpose:

- Combine all major final results (mass-loss timeseries, best-fit Sersic parameters, warp angles, etc.) into one consolidated table or set of tables for reference in a scientific paper.
- Possibly store them as CSV in `results/`.

Inputs: All stored data.
Outputs:

- e.g. `results/m33_final_summary.csv` that includes time, bound mass, R_j, Re, n, pitch_angle, warp, etc.

Interpretation: This final table is extremely valuable for referencing in the 5–6 page paper.

7. Conclusion & Next Steps

Cell 7.1 — "Summary Markdown & Future Work"

Type: Markdown

Purpose:

- Provide a concluding statement about the key findings: mass lost, final disk shape, morphological transformations, etc.
- Possibly indicate next steps or placeholders for referencing external papers.

Inputs: None (purely textual).
Outputs: Summarized text.
Interpretation: The user has a clear narrative that ties all the code outputs together for the final paper.

Additional Enhancements & Redundancies

Below are optional but recommended extras to ensure **robustness** and **clarity** throughout the notebook:

- **Inline "Sanity Checks"** in each major cell:
  - E.g. after computing bound mass, check if mass is decreasing monotonically or if any snapshot yields a negative or suspiciously large mass, and log a warning.
- **Repeated or comparative plots**:
  - For instance, for the disk surface density, show both a standard $(R, \Sigma(R))$ plot and a log–log version.
  - For morphological transformations, do both a "2D histogram" and a "density contour" approach.
- **Diagnostic text prints**:
  - After each fit, print out the reduced chi-square or some measure of goodness-of-fit so the user can see if a Sersic profile is a good representation or if an exponential is better.
- **Document everything** with docstrings and inline comments, ensuring that a new reader can follow the logic.

Overall Flow Recap

1. **Setup** (Imports, logging, user parameters).
2. **Data & Utility Definitions** (file parsing, center-of-mass, math).
3. **Mass Loss** (Jacobi radius, bound mass).
4. **Disk Fitting** (surface density, Sersic/exponential, tidal truncation).
5. **Morphology** (face-on/edge-on, spiral arms, warp, scale height).
6. **Integration** (mass vs. morphology).
7. **Conclusion** (final summary, data tables, references for the paper).

When finished, this single notebook (with $\sim$ 25–30 well-labeled cells) will function as a **complete pipeline** that can be run from top to bottom, generating all the final figures and data that feed into our 5–6 page cosmology paper.

This is the **master blueprint**.