1. Which columns are initially selected for use, and why might these particular features be retained?
In data_cleaning.ipynb, a specific subset of columns is chosen: socioeconomic indicators (e.g., tract_to_msa_income_percentage), demographic attributes (derived_ethnicity, derived_race, derived_sex), loan-related features (interest_rate, loan_type, loan_purpose, lien_status, construction_method, occupancy_type), and action_taken. These features are retained because they are relevant for detecting and analyzing bias (demographics, SES proxies), understanding the financial context (interest rates, loan types), and ultimately determining the target variable (action_taken → loan_approved). The combination provides both predictive power and the ability to measure fairness.

2. How are missing values in demographic features handled, and why is this approach taken?
Missing demographic values are filled with placeholders like "Sex Not Available" or "Ethnicity Not Available." This approach ensures that no entries are lost due to missing values and that the model has an explicit category for unavailable demographic information. This prevents silent dropping of rows and maintains representativeness in the dataset, which is crucial when analyzing fairness metrics across demographic groups.

3. What is the significance of using drop_first=True when one-hot encoding certain categorical columns?
Using drop_first=True avoids the dummy variable trap by removing one of the categories as a baseline. Without it, one category would be perfectly predicted by the others, introducing redundant information. This baseline category acts as a reference point, allowing the model coefficients (in linear models) or the network weights (in neural networks) to adjust relative to this omitted category, simplifying the model's interpretation and potentially aiding training stability.

4. How is the SES group determined using tract_to_msa_income_percentage, and why split into Low, Middle, High?
In data_transform.ipynb, the SES classification uses thresholds: <80 for Low, 80–120 for Middle, and >120 for High. This segmentation groups applicants into meaningful economic strata. Low SES indicates areas where the tract's median family income is significantly below the MSA median, Middle SES represents a roughly average scenario, and High SES corresponds to a relatively affluent neighborhood. Grouping in this manner allows the code to clearly identify and measure fairness disparities across economic lines.

5. Why is MinMaxScaler applied to numeric features?
MinMaxScaler transforms numeric features into a common [0, 1] range. By ensuring all numeric inputs share a similar scale, models (especially neural networks) train more efficiently. Without scaling, features with very large ranges could dominate the gradient updates, making training unstable or slow. The scaling step helps the optimization process converge more reliably.

6. How is action_taken mapped to a binary loan_approved outcome, and what does this represent?
A dictionary maps HMDA action codes (e.g., originated, denied, withdrawn) to a binary variable: 1 for approvals (loan originated or purchased) and 0 for any form of denial or non-origination (e.g., application denied, withdrawn). This binary variable defines the predictive modeling task: determining whether a given application ends up "approved" or "not approved." This simplification is essential to apply standard binary classification models and fairness metrics.

7. Why is the data split into training and testing sets, and what is the importance of random_state?
Splitting into training and testing sets ensures that model performance and fairness metrics are evaluated on unseen data, preventing overfitting and providing a realistic assessment of generalization. The random_state parameter guarantees reproducibility, allowing the same split to occur every time the code is run, ensuring consistent comparisons and reliable model evaluation over multiple training sessions.

8. What is the role of batch normalization and dropout in the neural network model?
Batch normalization standardizes the activations of a given layer, reducing internal covariate shift and stabilizing training. Dropout randomly zeroes out a fraction of the neurons' outputs during training, preventing overfitting by encouraging the network to rely on multiple neurons rather than a few memorized pathways. Combined, these techniques make the model more generalizable, robust, and stable.

9. Why are fairness metrics (Statistical Parity, Predictive Parity, Equal Opportunity) computed for different groups?
Fairness metrics are computed by grouping data along protected attributes like race, ethnicity, gender, and SES. This breakdown identifies whether certain subgroups face systematic disadvantages in the model's decisions. If, for instance, Black or African American applicants have significantly lower TPR (Equal Opportunity) compared to White applicants, it indicates potential bias. By computing metrics per group, the code reveals such disparities explicitly.

10. How does the code retrieve demographic categories from one-hot encoded fields for fairness analysis?
After prediction, the code includes functions to map one-hot encoded race/gender/ethnicity back to their original categories. These functions scan the row's one-hot columns, identify which column is 1, and use a predefined mapping dictionary to return the corresponding category name. This step is necessary because analyzing fairness requires human-readable category labels rather than numeric or one-hot encodings.

11. What is the purpose of computing fairness metrics at multiple intersectional levels (e.g., SES × Race × Gender × Ethnicity)?
Analyzing at an intersectional level (combining multiple attributes) allows detection of more subtle biases that might not be apparent when looking at single dimensions alone. For example, while the model might appear fair across gender alone, intersectional analysis could reveal that low-SES, Black, female applicants have a disproportionately low approval rate. Intersectional fairness checks ensure that fairness holds across all meaningful subgroup combinations, not just in isolation.

12. In adversarial_model.ipynb, why is there separate handling of sensitive attributes aside from the main feature set?
The adversarial model aims to remove biases related to sensitive attributes. While the main model input excludes explicit sensitive attributes, the adversaries need these attributes to attempt to predict them from the model's latent representation. By separating sensitive features, the code can feed these attributes exclusively into the adversary networks. The main model never directly sees these attributes as input, but the adversaries try to infer them, prompting the main model (via gradient reversal) to "unlearn" any sensitive correlations.

13. What is the purpose of resampling by demographic groups (using SMOTE, oversampling, or undersampling)?
Resampling ensures that the training data is balanced across different demographic groups. Without it, some groups might be underrepresented, leading the model and adversaries to learn patterns influenced by the dominant group's characteristics. Balancing the dataset helps the adversarial model receive a fair signal from all groups, improving its ability to remove biases that would otherwise be reinforced by skewed distributions.

14. How does the GradientReversalLayer enable adversarial debiasing?
The GradientReversalLayer inverts the gradients from the adversary networks before they update the main model's weights. Normally, if the adversaries learn to predict sensitive attributes well, the main model would remain unchanged. With gradient reversal, the main model receives a negative gradient whenever the adversaries succeed, forcing it to modify its internal representation to become less predictive of sensitive attributes. This setup ensures that the main model becomes "fairer" by not encoding sensitive information.

15. Why does the adversarial model have multiple outputs for race, gender, and ethnicity, and how are their losses combined?
The model outputs one prediction for loan approval and additional predictions for race, gender, and ethnicity adversaries. Each adversary tries to predict a sensitive attribute set. By training all adversaries simultaneously, the model is penalized if it encodes any of these sensitive attributes. The losses are combined with specified weights, allowing the main prediction task (loan approval) to remain primary while still heavily penalizing the model's ability to reveal sensitive attributes. Balancing these losses enforces fairness without destroying the model's predictive utility.

16. How does training with adversarial networks differ from simply removing sensitive attributes?
Simply removing sensitive attributes from the input does not prevent the model from using proxies (like income or geographic indicators) that correlate with protected characteristics. Adversarial training explicitly pressures the model to eliminate sensitive information from its latent representation. The adversaries try to guess these attributes from the model's internal states. The gradient reversal ensures that whenever they succeed, the main model adjusts to become less sensitive-attribute-informative. This is a more active and robust form of debiasing than mere attribute removal.

17. After training the adversarial model, how are fairness metrics evaluated again, and what differences might be expected compared to the baseline model?
After the adversarial model predicts loan approvals on the test set, the code recomputes the same fairness metrics (Statistical Parity, Predictive Parity, Equal Opportunity, etc.) on the same demographic and SES groups. Compared to the baseline model's metrics, it is often expected to see smaller disparities in TPR or approval rates across groups, indicating that adversarial training successfully reduced some forms of bias.

18. How does the code ensure the same SES logic is consistently applied when analyzing predictions from both baseline and adversarial models?
The code consistently uses the same thresholding logic for SES classification based on tract_to_msa_income_percentage (<80 for Low, 80–120 for Middle, >120 for High). Even after scaling or model predictions, it either uses original (unscaled) values or re-applies the same logic to maintain consistency. This ensures that SES groups remain comparably defined across baseline and adversarial analyses, making fairness comparisons valid.

19. What is the significance of monitoring validation loss and using early stopping during training?
Early stopping monitors validation loss to prevent overfitting. If the validation loss does not improve after several epochs (patience threshold), the model stops training. Restoring the best weights ensures that the final model is the one that performed best on unseen data during training. This procedure optimizes generalization, ensuring that reported performance and fairness metrics are more reliable.

20. Why might accuracy alone be insufficient to understand model performance in a lending approval scenario?
Accuracy treats all instances equally and does not distinguish between true positives, false positives, or false negatives. In a lending scenario, certain errors can disproportionately affect certain groups. For example, a low TPR (recall) for a minority group would not necessarily lower accuracy if that group is small. Fairness metrics like Equal Opportunity specifically measure how well the model treats qualified applicants across different groups, while precision and recall highlight the nature of errors. Thus, accuracy alone cannot detect if the model discriminates against certain demographics.

These 20 answers focus on the most crucial aspects of the codebase: data preprocessing, SES assignment, fairness metric computations, neural network training strategies, adversarial debiasing mechanics, and interpretability of fairness and performance metrics.