**Core Questions**

1. **Explain about your project**
   * My project is **Credit Card Fraud Detection using XGBoost**. The main goal is to build a machine learning model that can detect fraudulent transactions in real-time.
2. **Domain**
   * Domain is **Finance / Fraud Detection** under **Machine Learning – Classification problem**.
3. **Problem your project is trying to solve**
   * Fraudulent transactions cause huge financial losses. The problem is to **detect frauds with higher recall** so banks can take preventive actions.
4. **Dataset Overview**
   * Records: 72,000+ (mention exact after checking file).
   * Features: ~20+ (transaction details).
   * Target variable: 0 = Non-Fraud, 1 = Fraud.
   * Highly imbalanced dataset.
5. **Data Preprocessing steps**
   * Missing values handled, categorical encoded, numerical scaled.
   * Outliers checked, skewness reduced if required.
6. **Outlier handling**
   * Checked with boxplots & z-score/IQR.
   * Removed extreme cases OR transformed them.
7. **Skewness**
   * Verified using distribution plots.
   * Handled with transformations (log, square-root if necessary).
8. **Missing values**
   * Checked per column.
   * Strategy: Median for numeric, Mode for categorical.
9. **Feature Engineering**
   * Created derived features (e.g., transaction frequency).
   * Converted categorical to numerical using encoding.
10. **Feature Selection**
    * Used **SelectKBest (ANOVA F-test)** to select top features.
11. **Feature Scaling**
    * Applied **StandardScaler** for numerical features.
12. **Models used**
    * Tried baseline models (Logistic Regression, RandomForest).
    * Final model: **XGBoost Classifier**.
13. **Model Evaluation Metrics**
    * Accuracy, Precision, Recall, F1-score.
    * Confusion Matrix for visualization.
14. **Hyperparameter Tuning (GridSearchCV)**
    * Tuned learning\_rate, max\_depth, n\_estimators.
    * Best params: { 'learning\_rate': 0.01, 'max\_depth': 5, 'n\_estimators': 200 }.
15. **Explain the Algorithm (XGBoost)**
    * Gradient boosting algorithm that builds decision trees sequentially.
    * Optimizes using gradient descent, handles missing data well, and is efficient.
16. **Target Variable**
    * Fraud (Yes/1) vs Non-Fraud (No/0).
17. **Input Features**
    * Transaction amount, merchant category, location, time-based features, etc.
18. **Records & Features**
    * Around 72,000 rows, 20+ features. *(check exact in your dataset)*
19. **Model Building & Evaluation**
    * Preprocessing → Feature selection → XGBoost → Evaluation.
20. **Presentation**
    * Dataset overview → Preprocessing → Model → Results → Conclusion → Future Enhancements.

**✅ Additional Possible Questions**

* Why did you choose **XGBoost** over others?  
  *Because it performs well on tabular data, handles missing values, supports regularization, and works better with imbalanced datasets compared to logistic regression or decision trees.*
* Why recall is important in fraud detection?  
  *Because missing a fraud case (false negative) is costlier than flagging a non-fraud as fraud (false positive).*
* What challenges did you face?  
  *Class imbalance, feature importance selection, and hyperparameter tuning to avoid overfitting.*
* What future enhancements can you add?  
  *SMOTE/ADASYN for balancing, advanced tuning with Optuna, deep learning (LSTM/Autoencoders), real-time deployment.*

## Here's the plan for tomorrow's Viva prep:

### 1. ****Validate Your Notebook Structure****

Make sure your notebook contains:

* **End-to-end pipeline:** preprocessing → feature selection → model training → evaluation → feature importance → model saving
* **Documentation** with: Project Statement, Dataset Overview (records/features), Preprocessing details, Results tables, Conclusion, Future Enhancements.

### 2. ****Done-For-You Viva Q&A Cheat Sheet****

Below is a concise Q&A format you can use for quick revision. You'll fill in exact numbers (records, feature list) from your notebook tomorrow.

## 🗒️ Viva Cheat Sheet (Fill in your numbers where needed)

**Q1. Explain your project.**  
A: Built a **Credit Card Fraud Detection model using XGBoost**. The goal is to predict whether a transaction is fraudulent.

**Q2. Domain?**  
A: Finance – Fraud Detection – Binary Classification.

**Q3. Problem statement?**  
A: Detect fraud in credit card transactions to prevent financial losses.

**Q4. Dataset overview:**  
A: Uses <1% fraud cases).

**Q5. Preprocessing steps?**  
A: Handled missing values, encoded categoricals, scaled numerical data.

**Q6. Outlier handling? Skewness?**  
A: Checked using boxplots/IQR/log transformations; addressed skewness or extreme values if needed.

**Q7. Feature engineering?**  
A: Created derived features like transaction frequency, interaction terms, time-based features.

**Q8. Feature selection?**  
A: Applied **SelectKBest (ANOVA F-test)** to pick top features.

**Q9. Feature scaling?**  
A: Used **StandardScaler** for numerical features.

**Q10. Models tried? Final model?**  
A: Baseline: Logistic Regression, Random Forest. Final: **XGBoost Classifier**.

**Q11. Evaluation metrics?**  
A: Accuracy, Precision, Recall, F1-score, Confusion Matrix.

**Q12. Hyperparameter tuning details:**  
A: Tuned learning\_rate, max\_depth, n\_estimators via GridSearchCV. Best params: { 'learning\_rate': 0.01, 'max\_depth': 5, 'n\_estimators': 200 }.

**Q13. Explain XGBoost algorithm.**  
A: Gradient boosting on decision trees, builds sequential learners to minimize loss via gradient descent, handles missing data efficiently, supports regularization.

**Q14. Target variable? Input features? Number of records/features?**  
A: Target = Fraud (1) vs. Non-Fraud (0). Inputs: transaction-related details. Check notebook for exact counts: [ ] records, [ ] features.

**Q15. Model pipeline overview?**  
A: preprocessing → feature selection → XGBoost → evaluation.

**Q16. Classification results?**  
A: Test set overall accuracy = 0.52

| **Class** | **Precision** | **Recall** | **F1-score** | **Support** |
| --- | --- | --- | --- | --- |
| No (0) | 0.53 | 0.50 | 0.52 | 369 |
| Yes (1) | 0.51 | 0.55 | 0.53 | 354 |

**Q17. Interpretation of results?**  
A: Slightly better detection (recall) for fraud class, though accuracy remains low due to class imbalance. Precision of fraud class at 0.51 means about half of predicted frauds are correct; recall 0.55 shows capturing just over half of actual fraud cases.

**Q18. Challenges faced?**  
A: Highly imbalanced data, feature selection, tuning for no overfitting.

**Q19. Future enhancements?**  
A:

* SMOTE / ADASYN or class weights
* Advanced hyperparameter tuning (Optuna)
* Try alternatives: LightGBM, CatBoost
* Time-series / velocity features
* Deep learning (LSTM, autoencoders)
* Real-time API deployment

## What's next:

* Save this cheat sheet and revisit it quickly tomorrow.
* Double-check numbers like counts of records and features from your notebook.
* Revise feature names and results to ensure everything matches exactly when you speak.