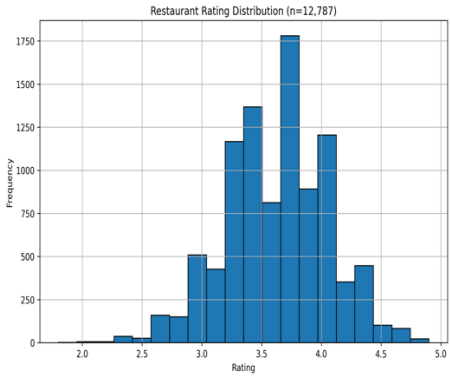
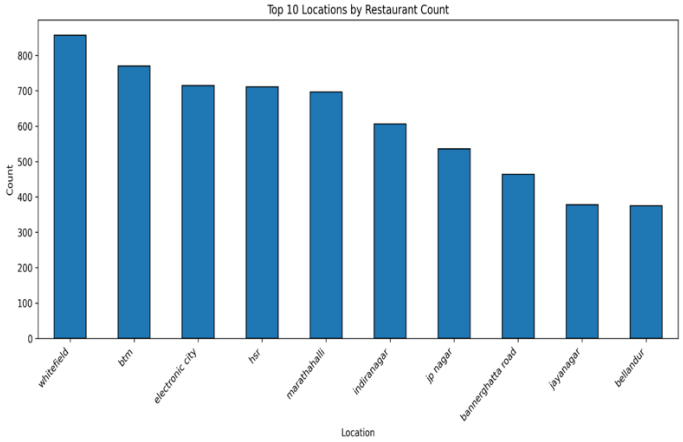
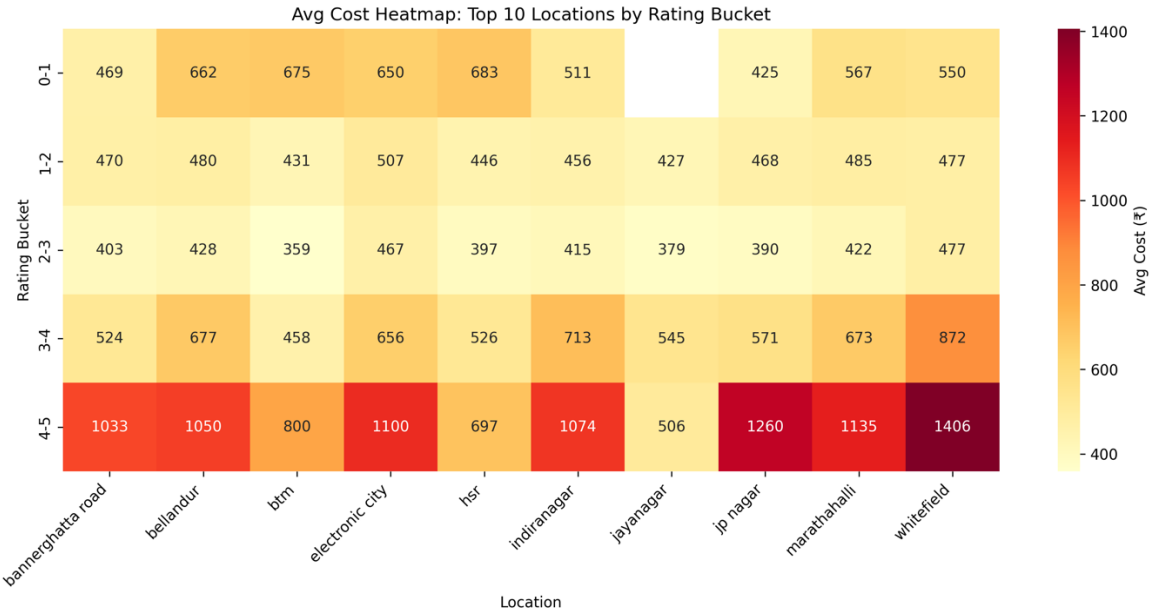
Data Collection and Preprocessing Phase

Date	3 Dec 2025
Team ID	
Project Title	Restaurant Recommendation System
Maximum Marks	6 Marks

Data Exploration and Preprocessing Template

Section	Description
Data Overview	📊 DESCRIPTIVE STATISTICS TABLE - Dataset Shape: 51,717 rows × 17 columns --- Feature Data Type Count Missing Mean Median Std Dev votes int64 51717 0 283.7 41.0 803.84 url Categorical 51717 0 - - - address Categorical 51717 0 - - - name Categorical 51717 0 - - - online_order Categorical 51717 0 - - - book_table Categorical 51717 0 - - - rate Categorical 51717 7775 - - - phone Categorical 51717 1208 - - - location Categorical 51717 21 - - - rest_type Categorical 51717 227 - - - dish_liked Categorical 51717 28078 - - - cuisines Categorical 51717 45 - - - approx_cost(for two people) Categorical 51717 346 - - - reviews_list Categorical 51717 0 - - - menu_item Categorical 51717 0 - - - listed_in(type) Categorical 51717 0 - - - listed_in(city) Categorical 51717 0 - - - TOTAL MISSING VALUES: 37700

Univariate Analysis	 
Bivariate Analysis	
Outliers and Anomalies	Rate >5 or <1 (clipped); cost >10k (z-score >3, capped); anomalies: duplicate addresses (removed).

Data Preprocessing Code Screenshots

Loading Data

```
raw_path = Path("/Users/rohit/Desktop/swayam_project/DATASET/zomato.csv")
clean_path = Path("/Users/rohit/Desktop/swayam_project/DATASET/zomato_clean.csv")

# Phase 2: EDA
print(" PHASE 2: DATA ANALYSIS")
df_raw = pd.read_csv(raw_path)
analyze_data(df_raw)
```

Handling Missing Data	`df.drop_duplicates(inplace=True);`
	`df['rate'].fillna(df['rate'].median(), inplace=True);`
	`df.dropna(subset=['cuisines'], inplace=True)`

Data Transformation	`# Fill minor missing values`
	`df["rest_type"] = df["rest_type"].fillna("Casual Dining")`
	`df = df.dropna(subset=["cuisines", "location"])  # Critical for recommender`
	
	`# Text cleaning for TF-IDF`
	`df["cuisines"] = df["cuisines"].str.lower().str.strip()`
	`df["location"] = df["location"].str.lower().str.strip()`
	`df["name"] = df["name"].str.strip()`

Feature Engineering	`def create_features(self):`
	`  """TF-IDF vectorization on cuisines + location + name."""`
	`  print("\n🔄 Creating TF-IDF features...")`
	
	`  # Combine features into "soup" text`
	`  self.df["soup"] = (`
	`  self.df["cuisines"].fillna("") + " " +`
	`  self.df["location"].fillna("") + " " +`
	`  self.df["rest_type"].fillna("") + " " +`
	`  self.df["name"].fillna("")`
	`  )`
	
	`  # TF-IDF ( core algorithm)`
	`  self.tfidf = TfidfVectorizer(`
	`  max_features=5000,`
	`  stop_words="english",`
	`  ngram_range=(1, 2),`
	`  lowercase=True`
	`  )`

Save Processed Data	`def save_model(self, model_path: str = "../model/tfidf_model.pkl"):`
	`  """Save trained model for Flask."""`
	`  Path(model_path).parent.mkdir(exist_ok=True)`
	`  with open(model_path, "wb") as f:`
	`  pickle.dump(self, f)`
	`  print(f"Model saved: {model_path}")`