TableIdentifier-Working-Version-1, covering purpose, functionality, inputs/outputs, dependencies, and interactions.

1. main.py (DatabaseAnalyzer)

- **Purpose**: Entry point for the Database Schema Analyzer, orchestrating component initialization and CLI interaction.
- **Functionality**:
  - Initializes logging (app-config/BikeStores/logging_config.ini).
  - Manages database connection (DatabaseConnection).
  - Coordinates managers (SchemaManager, PatternManager, FeedbackManager, NLPPipeline, NameMatchManager, TableIdentifier, QueryProcessor).
  - Runs CLI (DatabaseAnalyzerCLI) for user interaction.
  - Supports connecting to databases (e.g., "BIKES_DB"), processing queries, reloading configurations, managing feedback, and generating DDL.
- **Key Methods**:
  - __init__: Sets up logging and placeholders for managers.
  - run: Starts CLI and handles shutdown.
  - connect_to_database: Connects to the database and initializes managers.
  - _initialize_managers: Sets up all components with db_name.
  - process_query: Delegates query processing to QueryProcessor.
  - generate_ddl: Creates DDL for specified tables.
- **Inputs**:
  - Config path (app-config/database_configurations.json).
  - User inputs via CLI (e.g., "BIKES_DB", queries).
- **Outputs**:
  - CLI prompts and results (e.g., table suggestions).
  - Logs (logs/bikestores_app.log).
- **Dependencies**:
  - database.connection.DatabaseConnection
  - config.manager.DBConfigManager
  - config.patterns.PatternManager
  - schema.manager.SchemaManager
  - feedback.manager.FeedbackManager
  - analysis.table_identifier.TableIdentifier
  - analysis.name_match_manager.NameMatchManager
  - analysis.processor.NLPPipeline
  - nlp.QueryProcessor.QueryProcessor
  - cli.interface.DatabaseAnalyzerCLI
- **Interactions**:
  - Calls DBConfigManager.load_configs for database configs.
  - Uses DatabaseConnection.connect to establish connections.
  - Initializes managers in _initialize_managers.
  - Delegates queries to QueryProcessor.process_query.
  - Stores feedback via FeedbackManager.store_feedback.

2. analysis/processor.py (NLPPipeline)

- **Purpose**: Processes natural language queries using spaCy for tokenization, entity recognition, and pattern matching.
- **Functionality**:
  - Loads spaCy model (en_core_web_trf).
  - Converts PatternManager patterns into spaCy matcher patterns.
  - Analyzes queries to extract tokens, entities, matches, and dependencies.
  - Fixed E178 error by generating patterns from query strings.
- **Key Methods**:
  - __init__: Initializes spaCy, matcher, and loads patterns.
  - _load_patterns: Converts PatternManager patterns to spaCy format.
  - analyze_query: Returns analysis dictionary with tokens, entities, etc.
- **Inputs**:
  - pattern_manager: PatternManager instance.
  - db_name: Database name for logging.
  - Query string (e.g., "SHOW ME PRODUCTS STOCK AVAILABILITY AT ALL STORE").
- **Outputs**:
  - Dictionary: {"entities": [], "tokens": [], "matches": [], "dependencies": []}.
- **Dependencies**:
  - spacy
  - config.patterns.PatternManager
- **Interactions**:
  - Fetches patterns via PatternManager.get_patterns.
  - Provides tokens to QueryProcessor for synonym matching.
  - Logs to logs/bikestores_app.log.

3. nlp/QueryProcessor.py (QueryProcessor)

- **Purpose**: Core query processing, mapping natural language to database tables and columns (basic version).
- **Functionality**:
  - Integrates TableIdentifier for table detection.
  - Uses NLPPipeline for query analysis.
  - Matches columns via NameMatchManager for synonym learning.
  - Updates feedback weights through TableIdentifier.
- **Key Methods**:
  - __init__: Initializes components and logging.
  - process_query: Identifies tables and matches columns, returning tables and confidence.
- **Inputs**:
  - connection_manager, schema_dict, nlp_pipeline, table_identifier, name_matcher, pattern_manager, db_name.
  - Query string.
- **Outputs**:

    - Tuple: (tables: List[str], confidence: float) or (None, False) if no tables.
- **Dependencies**:
    - analysis.table_identifier.TableIdentifier
    - analysis.name_match_manager.NameMatchManager
    - analysis.processor.NLPPipeline
    - config.patterns.PatternManager
    - database.connection.DatabaseConnection
- **Interactions**:
    - Calls TableIdentifier.identify_tables for table suggestions.
    - Uses NLPPipeline.analyze_query for tokens.
    - Updates synonyms via NameMatchManager.update_synonyms.
    - Logs query steps.

4. analysis/name_match_manager.py (NameMatchManager)

- **Purpose**: Manages column synonym mappings using semantic similarity.
- **Functionality**:
    - Loads default (default_name_matches.json) and dynamic (dynamic_name_matches.json) synonyms.
    - Uses SentenceTransformer (all-MiniLM-L6-v2) for embeddings.
    - Prompts users for synonym confirmation (e.g., "Does 'availability' refer to 'quantity'?").
    - Saves synonyms to JSON files.
- **Key Methods**:
    - __init__: Loads model and JSONs.
    - update_synonyms: Matches tokens to columns, adds synonyms.
    - _prompt_for_synonym: Asks user to confirm mappings.
    - save_dynamic: Saves dynamic synonyms.
- **Inputs**:
    - db_name: For file paths and logging.
    - Tokens, embeddings, columns for matching.
- **Outputs**:
    - Updated dynamic_name_matches.json.
    - Synonym lists for columns.
- **Dependencies**:
    - sentence_transformers.SentenceTransformer
    - sklearn.metrics.pairwise.cosine_similarity
- **Interactions**:
    - Used by QueryProcessor and TableIdentifier.
    - Reads/writes JSONs in app-config/BikeStores/.
    - Logs synonym updates.

5. config/patterns.py (PatternManager)

- **Purpose**: Manages query-to-table patterns for matching.
- **Functionality**:

    - Loads patterns from app-config/global_patterns.json.
    - Normalizes queries (lowercase, strip spaces).
    - Provides patterns to NLPPipeline and TableIdentifier.
    - Added get_patterns to fix AttributeError.
- **Key Methods**:
    - __init__: Loads patterns into pattern_weights.
    - _load_patterns: Reads and normalizes JSON.
    - get_patterns: Returns pattern_weights.
- **Inputs**:
    - schema_dict: Database schema.
- **Outputs**:
    - Dictionary: {query: {table: weight}}.
- **Dependencies**:
    - None (uses standard libraries).
- **Interactions**:
    - Provides patterns to NLPPipeline._load_patterns.
    - Used by TableIdentifier.identify_tables.

6. feedback/manager.py (FeedbackManager)

- **Purpose**: Stores and retrieves query-table feedback to improve table identification.
- **Functionality**:
    - Caches feedback in feedback_cache/BikeStores/ (e.g., 20250416133007_meta.json).
    - Extracts patterns from queries (e.g., "[LITERAL]" for "Baldwin Bikes").
    - Matches queries using SentenceTransformer for similarity.
    - Tracks top queries (e.g., "total sales amount at storename 'Baldwin Bikes'").
- **Key Methods**:
    - __init__: Loads feedback cache.
    - store_feedback: Saves query-table mappings.
    - get_similar_feedback: Returns tables for similar queries.
    - get_top_queries: Lists frequent queries.
- **Inputs**:
    - db_name: For cache directory.
    - Query, tables, schema for storage.
- **Outputs**:
    - Feedback JSONs.
    - Table lists for queries.
- **Dependencies**:
    - sentence_transformers.SentenceTransformer
    - spacy
- **Interactions**:
    - Used by TableIdentifier for feedback-based table matching.
    - Called by DatabaseAnalyzer.confirm_tables.

7. schema/manager.py (SchemaManager)

- **Purpose**: Manages database schema metadata, caching to JSON.
- **Functionality**:
    - Builds schema dictionary (tables, columns) from database.
    - Caches to schema_cache/BikeStores/schema.json.
    - Checks for schema updates using modification times.
- **Key Methods**:
    - __init__: Sets up cache directory.
    - needs_refresh: Compares schema and cache timestamps.
    - build_data_dict: Queries database for schema.
    - load_from_cache: Loads cached schema.
- **Inputs**:
    - db_name: For cache path.
    - Database connection.
- **Outputs**:
    - Schema dictionary: {"tables": {}, "columns": {}, "version": "1.0"}.
- **Dependencies**:
    - None (uses standard libraries).
- **Interactions**:
    - Used by DatabaseAnalyzer to load schema.
    - Provides schema to PatternManager, TableIdentifier, QueryProcessor.

8. cli/interface.py (DatabaseAnalyzerCLI)

- **Purpose**: Provides the command-line interface for user interaction.
- **Functionality**:
    - Displays main menu (Connect, Query, Reload, Feedback, Exit).
    - Handles database selection (e.g., "BIKES_DB").
    - Processes queries, showing table suggestions and synonym prompts.
    - Manages feedback and DDL generation.
- **Key Methods**:
    - run: Main CLI loop.
    - connect_to_db: Lists configs and connects.
    - query_mode: Handles query input and results.
- **Inputs**:
    - analyzer: DatabaseAnalyzer instance.
    - User inputs (menu options, queries).
- **Outputs**:
    - CLI prompts and outputs.
- **Dependencies**:
    - main.DatabaseAnalyzer
- **Interactions**:
    - Calls DatabaseAnalyzer methods (connect_to_database, process_query, etc.).
    - Displays FeedbackManager.get_top_queries.

9. database/connection.py (DatabaseConnection)

- **Purpose**: Manages database connections.
- **Functionality**:
    - Connects to databases using provided configs (e.g., SQL Server for "BIKES_DB").
    - Checks connection status and closes connections.
- **Key Methods**:
    - connect: Establishes connection.
    - is_connected: Checks status.
    - close: Closes connection.
- **Inputs**:
    - Config dictionary (host, database, user, etc.).
- **Outputs**:
    - Connection object or None.
- **Dependencies**:
    - Database driver (e.g., pyodbc).
- **Interactions**:
    - Used by DatabaseAnalyzer and SchemaManager.

10. config/manager.py (DBConfigManager)

- **Purpose**: Loads database configurations from JSON.
- **Functionality**:
    - Reads app-config/database_configurations.json.
    - Returns config dictionary for selected database.
- **Key Methods**:
    - load_configs: Loads JSON.
- **Inputs**:
    - Config file path.
- **Outputs**:
    - Dictionary of configs (e.g., {"BIKES_DB": {...}}).
- **Dependencies**:
    - None.
- **Interactions**:
    - Used by DatabaseAnalyzer.load_configs.

11. app-config/BikeStores/logging_config.ini

- **Purpose**: Configures logging for all components.
- **Functionality**:
    - Defines loggers (analyzer, query_processor, etc.).
    - Sets handlers: console (INFO), file (DEBUG, logs/bikestores_app.log, 10MB rotation, 5 backups).
    - Formats logs: %(asctime)s - %(name)s - %(levelname)s - %(message)s.
- **Inputs**:
    - Loaded by each component via logging.config.fileConfig.
- **Outputs**:

    - Console output and logs/bikestores_app.log.
- **Dependencies**:
    - None.
- **Interactions**:
    - Used by all Python modules for logging.