HIGHLIGHTS:

1. **Database Backup and Recovery:** I developed a comprehensive database backup and recovery system capable of generating full backups, including tables, indexes, views, and insert statements, while also providing restoration functionality.
2. **Metadata Processing:** I effectively utilized database metadata to extract key information, such as table structures and foreign key relationships, providing essential data support for the backup process.
3. **Depth-First Search:** I employed a depth-first search algorithm to determine dependencies among tables, ensuring the correct sequence and completeness of the backup.
4. **Flexible SQL Construction:** I skillfully constructed SQL statements, considering the characteristics of various database objects such as tables, indexes, and views, ensuring the accuracy and executability of the generated SQL.
5. **Exception Handling and Logging:** I accounted for exceptional scenarios and implemented logging to clearly document the operations and status during the backup process, facilitating system monitoring and troubleshooting.
6. **Clear Code Structure:** My code is well-structured, with standardized method naming and detailed comments, making it easy to read and maintain.
7. **Data Type Handling:** I meticulously handled different data types, such as processing BLOB data and escaping strings, ensuring data accuracy and integrity.

BASIC FUNCTIONS:

1. **Ordered** Table
    a) Utilized hash maps
    b) Gathered table and foreign key information
    c) Sorted the tables based on their depth using depth-first search (DFS)
2. **Create** Table
    a) Checked if the table exists; if it does, executed the drop command
    b) `addColumns` checked if the column is `NOT NULL` or has a `DEFAULT` value
    c) `addForKey` checked for `ON DELETE` or `ON UPDATE` actions and included them
3. Gather and Save **Index** Information
    a) Ignored indexes named `sqlite_autoindex_`
    b) Used the `ORDINAL_POSITION` field to check if the column is the first in a constraint
    c) Checked and added if the index is `UNIQUE`, `ASC`, or `DESC`
4. Create and Save **Insert** Statements for Each Table

a) Checked if the data format is BLOB
b) Used `getBytes` to obtain BLOB bytes
c) Converted not null BLOB data to a hexadecimal string
d) Enclosed string data in quotes (“ ”) and prefixed BLOB data with an X

5. Create and Save **Views**
   a) Checked if the view exists; if it does, executed the drop command
   b) Iterated over the result set of views
   c) Defined the query

Additional Features:

1. **Checked** if the **backup file exists** before starting the backup; if it exists and is not in use, deleted it.

2. Checked for the existence of tables and views before creating them, performing **drop** operations if necessary.

3. Marked attributes as **`NOT NULL`** where applicable.

4. Appended "**default**" followed by the default value if the column has one.

5. Read and wrote **`ON DELETE NO ACTION`** and **`ON UPDATE NO ACTION`**.

6. Generated **version** and related information.

7. Implemented **data integrity checks** in Java: Generated **checksums** during the backup process to ensure the backup file has not been tampered with during restoration.

8. **Recorded metadata**: Logged backup metadata (e.g., backup time, data size, backup type) for easy management and auditing.