#_ the Ultimate NOSQL CheatSheet Shortcuts / Techniques

I. MongoDB

1. Basic Operations:

- `db.help()`: Lists all the commands available.
- `show dbs`: Lists all the databases.
- `use [DB_NAME]`: Switch to a database.
- `db.createCollection("collection_name")`: Creates a new collection.
- `show collections`: Lists all collections in the current database.

2. CRUD Operations:

- `db.collection.find()`: Retrieves documents from a collection.
- `db.collection.insertOne({})`: Inserts a new document into a collection.
- `db.collection.updateOne({}, {$set: {}})`: Updates a document in a collection.
- `db.collection.deleteOne({})`: Deletes a document from a collection.
- `db.collection.bulkWrite([])`: Performs multiple write operations in bulk.

3. Query and Projection Operators:

- `$eq`: Matches values that are equal to a specified value.
- `$gt`: Matches values that are greater than a specified value.
- `$in`: Matches any of the values specified in an array.
- `$exists`: Matches documents that have the specified field.
- `$type`: Selects documents where a field is of the specified type.

II. CouchDB

4. Basic Operations:

- `GET /_all_dbs`: Lists all databases.
- `PUT /[DB_NAME]`: Creates a new database.

By: Waleed Mousa

- **GET /[DB_NAME]**: Retrieves information about the database.
- **DELETE /[DB_NAME]**: Deletes a database.

5. Document Operations:

- **PUT /[DB_NAME]/[DOC_ID]**: Creates or updates a document.
- **GET /[DB_NAME]/[DOC_ID]**: Retrieves a document.
- **DELETE /[DB_NAME]/[DOC_ID]**: Deletes a document.
- **POST /[DB_NAME]/_find**: Finds documents using the Mango query interface.

III. Cassandra

6. Basic Operations:

- **DESCRIBE KEYSPACES**: Lists all keyspaces.
- **USE [KEYSPACE_NAME]**: Switches to a keyspace.
- **CREATE KEYSPACE [KEYSPACE_NAME]**: Creates a new keyspace.

7. Table Operations:

- **CREATE TABLE [TABLE_NAME] (...)**: Creates a new table.
- **DESCRIBE TABLES**: Lists all tables in the keyspace.
- **DROP TABLE [TABLE_NAME]**: Drops a table.

8. CRUD Operations:

- **INSERT INTO [TABLE_NAME] (...) VALUES (...)**: Inserts data into a table.
- **SELECT * FROM [TABLE_NAME] WHERE [...]**: Selects data from a table with a specified condition.
- **UPDATE [TABLE_NAME] SET [...] WHERE [...]**: Updates data in a table.
- **DELETE FROM [TABLE_NAME] WHERE [...]**: Deletes data from a table.

By: Waleed Mousa

IV. Redis

9. Basic Operations:

- **PING**: Checks if the server is running.
- **INFO**: Retrieves information about the Redis server.
- **SELECT [DB_NUMBER]**: Selects a database by its number.
- **FLUSHDB**: Clears the selected database.

10. Key Operations:

- **SET [KEY] [VALUE]**: Sets a value to a key.
- **GET [KEY]**: Retrieves the value of a key.
- **DEL [KEY]**: Deletes a key.
- **EXPIRE [KEY] [TIME_IN_SECONDS]**: Sets an expiration time on a key.

11. Data Structure Operations:

- **LPUSH [LIST_NAME] [VALUE]**: Inserts a value at the start of a list.
- **RPUSH [LIST_NAME] [VALUE]**: Inserts a value at the end of a list.
- **SADD [SET_NAME] [VALUE]**: Adds a value to a set.
- **ZADD [ZSET_NAME] [SCORE] [VALUE]**: Adds a value to a sorted set with a specified score.

V. Neo4j - Basic Commands

- **CREATE (node_label)** - Create a new node.
- **MATCH (node_label)** - Match nodes with specific characteristics.
- **CREATE (n1)-[rel:RELATION]->(n2)** - Create a relationship between two nodes.
- **MATCH (n1)-[rel:RELATION]->(n2)** - Match a relationship between two nodes.
- **SET node.property = value** - Set a property on a node or relationship.
- **RETURN node** - Return the node or relationship.
- **DELETE node** - Delete a node or relationship.

By: Waleed Mousa

V. Advanced Tips and Tricks

12. Indexing and Performance Optimization:

- **Creating Indexes**: Create indexes to enhance query performance.
- **Sharding**: Implement sharding to distribute data across several machines.
- **Replication**: Set up replication to have multiple copies of the data.
- **Caching**: Use caching to speed up data retrieval operations.

13. Security:

- **Authentication and Authorization**: Implement authentication and authorization to secure your NoSQL databases.
- **Encryption**: Use encryption to protect sensitive data.
- **Monitoring and Logging**: Set up monitoring and logging to keep track of database activities.

14. Backup and Recovery:

- **Regular Backups**: Perform regular backups to prevent data loss.
- **Disaster Recovery Planning**: Develop a disaster recovery plan to restore functionality in case of a catastrophe.
- **Data Export and Import**: Learn how to export and import data for backup and recovery purposes.

VI. Community and Learning Resources:

- **Documentation**: Always refer to official documentation for detailed information.
- **Community Forums**: Participate in community forums to learn from others and share your knowledge.
- **Workshops and Webinars**: Attend workshops and webinars to enhance your skills.
- **Online Courses**: Enroll in online courses to get a structured learning pathway.

By: Waleed Mousa

- **Certification Programs**: Consider getting certified to validate your skills.

VII. General Tips and Best Practices

15. Database Design:

- **Schema Design**: Design a schema that is optimized for your workload and query patterns.
- **Denormalization**: Consider denormalization to optimize read performance.
- **Consistency Levels**: Understand and set the appropriate consistency levels for your use case.
- **Capacity Planning**: Plan capacity carefully to accommodate your data volume and throughput requirements.

16. Query Optimization:

- **Query Profiling**: Use query profiling to identify and eliminate performance bottlenecks.
- **Pagination**: Implement pagination to handle large sets of data more efficiently.
- **Batch Operations**: Use batch operations to perform bulk CRUD operations more efficiently.
- **Connection Pooling**: Use connection pooling to reuse existing connections instead of establishing new ones.

17. Monitoring and Maintenance:

- **Performance Monitoring**: Set up monitoring tools to track database performance and health.
- **Database Tuning**: Regularly tune your database settings to maintain optimal performance.
- **Log Analysis**: Analyze logs to identify issues and optimize operations.
- **Database Updates**: Regularly update your database software to benefit from performance improvements and security patches.

By: Waleed Mousa

VIII. Additional Concepts and Techniques

18. Data Modelling and Types:

- **Document Data Model**: Understand and leverage the document data model for storing semi-structured data.
- **Graph Data Model**: Learn about the graph data model for storing interconnected data.
- **Column-family Data Model**: Get acquainted with the column-family data model, which is suitable for analytics and IoT applications.
- **Key-Value Data Model**: Utilize the key-value data model for fast, simple lookups.

19. API Interactions:

- **REST API**: Use REST APIs for web-based interactions with your NoSQL database.
- **GraphQL**: Learn about GraphQL for more flexible and efficient data retrieval.
- **WebSocket**: Understand WebSocket for full-duplex communication channels over a single TCP connection.
- **gRPC**: Learn gRPC for performance-efficient and language-neutral data communication.

20. Troubleshooting:

- **Debugging**: Develop debugging skills to identify and fix issues more efficiently.
- **Error Handling**: Implement comprehensive error handling to manage exceptions gracefully.
- **Database Recovery**: Learn about database recovery techniques to restore data in case of failures.
- **Community Support**: Leverage community support for troubleshooting and learning.

21. Emerging Trends and Technologies:

- **IoT and NoSQL**: Understand the role of NoSQL in IoT applications.

- **Big Data and NoSQL**: Learn about the application of NoSQL in big data projects.
- **Machine Learning and NoSQL**: Explore the integration of machine learning with NoSQL databases.
- **Blockchain and NoSQL**: Understand the potential integration points between blockchain technology and NoSQL databases.

IX. Miscellaneous:

- **Data Migration**: Learn about techniques and tools for data migration between different NoSQL databases.
- **Multi-model Databases**: Get acquainted with multi-model databases that combine various data models into a single database.
- **Database Testing**: Implement database testing to ensure the reliability of your database.
- **Database Documentation**: Maintain comprehensive database documentation for easy reference and troubleshooting.
- **Community Contributions**: Contribute to the community by sharing your knowledge and experience.
- **Cross-platform Compatibility**: Ensure your NoSQL database is compatible with different platforms and environments.
- **Legal Compliance**: Be aware of legal compliance related to data storage and processing.
- **Sustainable Computing**: Implement sustainable computing practices to minimize the environmental impact of your database operations.
- **Database Optimization Tools**: Utilize database optimization tools to enhance database performance.
- **Database Security Tools**: Use database security tools to protect your database from vulnerabilities and attacks.
- **Database Management Tools**: Learn about various database management tools for easy and efficient database administration.
- **Ongoing Learning**: Keep learning to stay updated with the latest trends and developments in the NoSQL database domain.

By: Waleed Mousa