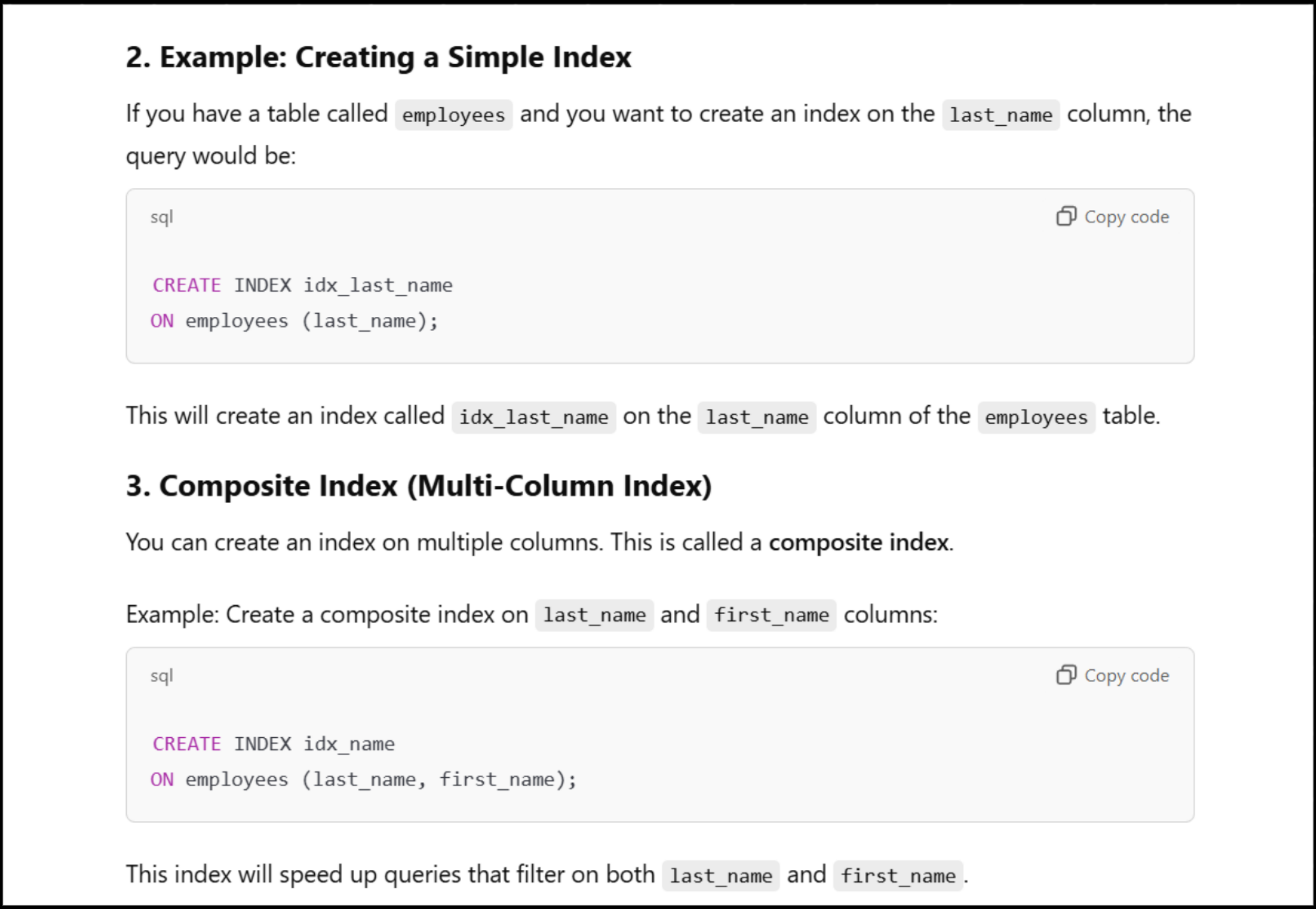
Index Implementation in SQL



2. Example: Creating a Simple Index

If you have a table called `employees` and you want to create an index on the `last_name` column, the query would be:

```sql
CREATE INDEX idx_last_name
ON employees (last_name);
```

This will create an index called `idx_last_name` on the `last_name` column of the `employees` table.

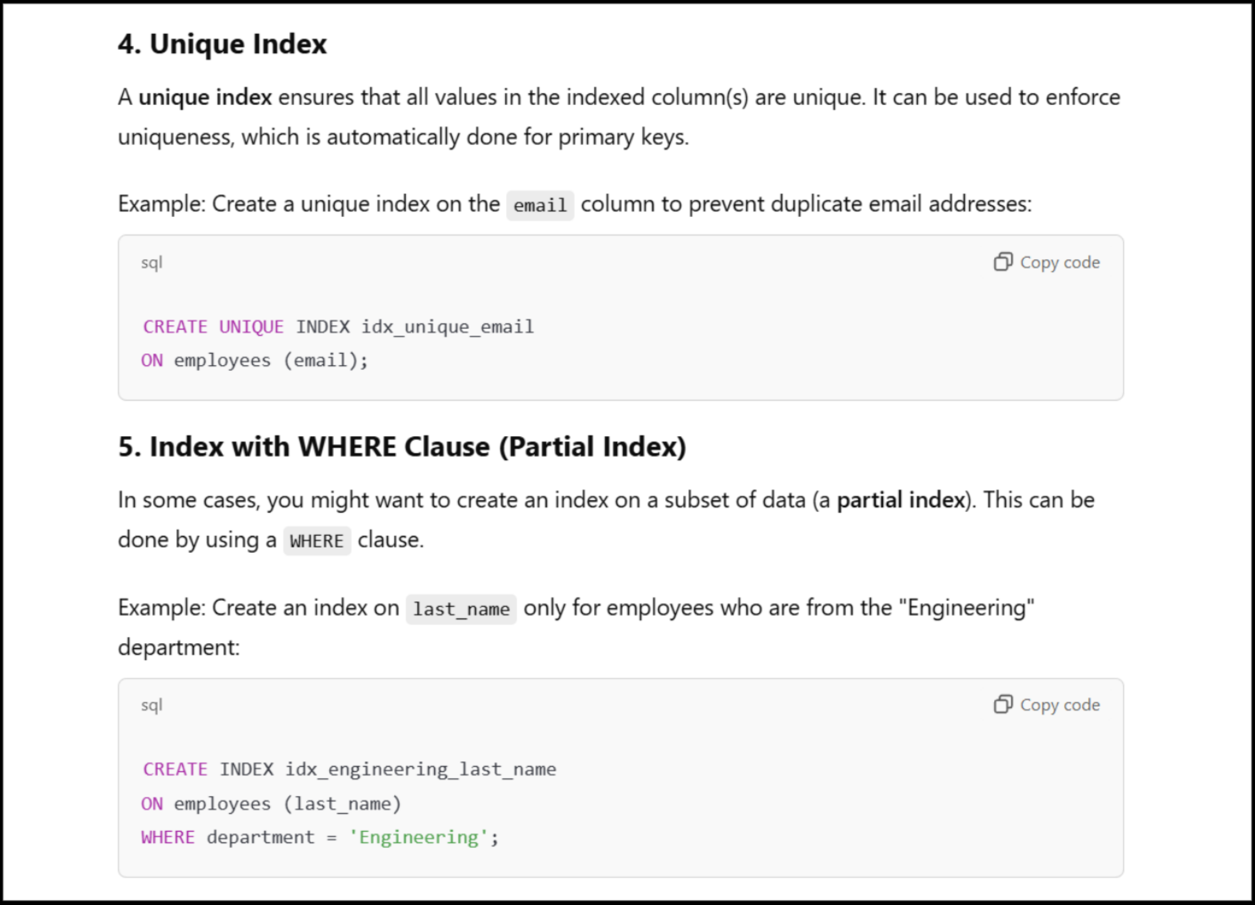
3. Composite Index (Multi-Column Index)

You can create an index on multiple columns. This is called a **composite index**.

Example: Create a composite index on `last_name` and `first_name` columns:

```sql
CREATE INDEX idx_name
ON employees (last_name, first_name);
```

This index will speed up queries that filter on both `last_name` and `first_name`.



4. Unique Index

A **unique index** ensures that all values in the indexed column(s) are unique. It can be used to enforce uniqueness, which is automatically done for primary keys.

Example: Create a unique index on the `email` column to prevent duplicate email addresses:

```sql
CREATE UNIQUE INDEX idx_unique_email
ON employees (email);
```

5. Index with WHERE Clause (Partial Index)

In some cases, you might want to create an index on a subset of data (a **partial index**). This can be done by using a `WHERE` clause.

Example: Create an index on `last_name` only for employees who are from the "Engineering" department:

```sql
CREATE INDEX idx_engineering_last_name
ON employees (last_name)
WHERE department = 'Engineering';
```

6. Full-Text Index

A **full-text index** is used for indexing text-based columns in a way that allows for full-text search (e.g., searching for specific words or phrases within a text column).

Example: Create a full-text index on the `description` column:

```sql
CREATE FULLTEX[illegible]EX idx_fulltext_description
ON employees (description);
```

(Note: Not all database systems support full-text indexing, so this syntax might vary depending on the DBMS.)

7. Clustered Index

A **clustered index** determines the physical order of rows in a table. Each table can have only one clustered index, typically created automatically on the **primary key**.

Example: If `id` is the primary key of the `employees` table, the DBMS will automatically create a clustered index on the `id` column. However, you can explicitly define a clustered index (if needed):

```sql
CREATE CLUSTERED INDEX idx_clustered_id
ON employees (id);
```

8. Dropping an Index

If you no longer need an index, you can drop it using the `DROP INDEX` statement:

```sql
DROP INDEX index_name;
```