Quick Solutions to SQL Performance Problems

Acceptable Performance	How Quickly Should Our Scripts Run? Somethimes expectations are unrelistic
Utility	Some queries or other database uses are not necessary
Time Shifting	Run demanding queries off peek

DataBase Optimisation

Architects aims are to:

Reducing or eliminating data redundancy.	Preventing data inconsistencies and inaccuracies.
Ensuring the correctness and integrity of your data.	Facilitating rapid data lookup, retrieval, and analysis.

In-Memory Databases

Improve speed on data retrieval

In-memory databases keep all data set in computer's memory - RAM	Traditional databases store data on hard drive (HDD or SSD)

So in-memory DBs do not need to perform disk read/write operations to return data

Saving memory through normalisation

1nf	Each cell in the table can have only one value,
2nf	Each of the attributes should be fully dependent on the entire primary ke
	For instance, an attribute "age" that depends on "birthdate" which in turn depends on "studentID" wouldn't meet 2nf
	Furthermore, a table with a primary key made up of multiple fields violates the second normal form if one or more of the other fields do not depend on every part of the key.

DataBase Optimisation (cont)

3nf	Every non-key column be independent of every other column. If changing a value in one non-key column causes another value to change, that table does not meet the third normal form.

Establish a connection to query a database

```
from sqlalchemy import create_engine
import pandas as pd
engine = create_engine('sqlite:///Northwind.s-
qlite')
```

SQL Diagraming

1. Monitor Wait Time 2. Review the Execution Plan 3. Gather Object Information 4. Find the Driving Table 5. Identify Performance Inhibitors

1, Wait Times	SQL Server incorporates wait types that allow you to monitor not only the total wait time but also each step of the query as it's processed through the database. W
2. Review rows ratio between detail and lookup tables	calculate the relative number of records required for the join criteria (that is, the average ratio of rows related between the detail table and lookup tables).
3. Gather Object Info	Determine which tables contain the detailed information and which tables are the master or lookup tables
Find the cardinality and distributions of a column	Find out the row count for each table involved.
4. Find the Driving Table	By using the driving table, we can query with the table that returns the least data
Next, look at the filtering predicates to find which table to drive the query with	The table that filters out most records is our driving table

Altering SQL Queries	
Ensure the Schema in columns we join on match	
Ensure Data Types are the same	
Ensure Character Encoding are the same	Both using the same UTF
Avoid Using GroupBy and Distinct together	

Checkpoints During Query Optimisation	
Checkpoints to evaluate as we optimise	Query Performs Adequately
	Resources required are expensive
	Reached a point of diminishing returns for optimisation
	A completely different solution is discovered

 # To use SQL

```
# Creating the context manager
con = engine.connect()
rs = con.execute("SELECT * FROM Orders")
```

 # To use Pandas on entire table

```
df = pd.DataFrame(rs.fetchall())
df.columns = rs.keys()
con.close()
```

 # Using Pandas on part of a table

```
df = pd.read_sql_query("SELECT OrderID, CompanyName
FROM Orders
INNER JOIN Customers on Orders.CustomerID =
Customers.CustomerID", engine)
print(df.head())
```