Stored Procedures:

A SQL stored procedure is a collection SQL statements that will be compiled, stored, and executed on the server. Stored procedures can also be cached and reused so it helps saving time and effort to run the same queries over and over. Stored procedure also helps improve performance of database operations such as select, update, and delete data.

There 2 types of Stored Procedures:

- System Stored Procedures are created and executed by SQL Server for the server admin works.
- User defined Stored Procedures contain one or more SQL statements, mostly to select, update, or delete records from database tables. It can take input parameter(s) and return output parameter(s).

```sql
-- Create Stored Procedure to stored best menu rating of a restaurant

DELIMITER $$

CREATE PROCEDURE Best_Menu_Rating (IN id_of_Restaurant INT)
BEGIN
    SELECT
        Restaurant_id, MAX(menu_rating)
    FROM
        Restaurant_Ratings
    WHERE
        Restaurant_id = id_of_Restaurant;
END //

DELIMITER //
```

```sql
-- Call the Stored Procedure with argument values = 23045

CALL Best_Menu_Rating(23045);
```

Functions Views:

- A SQL view is a virtual table based on the result-set of an SQL statement. It contains rows and columns, just like a real table.
- The fields in a view table are fields from one or more real tables in the database. Users can also add custom fields by using functions to gather the desired value for the fields.
- Users can add SQL statements and functions to a view and present the data as if the data were coming from one single table.

```
23  -- Creat a View for Restaurant Rating
24
25  CREATE VIEW [Average Restaurant Rating] AS
26  SELECT Restaurant_id, AVG(quality_rating) AS qual_rating, AVG(menu_rating) AS menu_rating, AVG(price_rating) AS price_rating,
27             AVG(package_rating) AS package_rating, AVG(delivery_time_rating) AS delivery_rating
28  FROM Restaurant_Ratings
29  GROUP BY Restaurant_id;
```

```
31  -- Query a View to view the Restaurant Rating for Restaurant_id = 23045
32
33  SELECT *
34  FROM [Average Restaurant Rating]
35  WHERE Restaurant_id = 23045;
36
```

Indexes for Reports:

A SQL Index is Data Structure that contains organized copy from one or more of the tables in the database. It is the single most important tool available to a database administrator for improving query performance. We are interested in average search time and maximum search time for queries.

- Without an indexing structure in place, the database must adopt a naïve approach and perform a full table scan.
- With an index in place, the DBMS can locate the information much more rapidly.

Indexes can be created on one or more columns; each column can belong to one or more indexes.

Indexes can be created:

- During the initialization state of a data table such as indexing the primary and/or foreign keys
- By SQL commands where the data is stored in a separate table accessible to any query

Example of query optimization:

Linear Search: Randomly Organized Names

Average search time:

- repeating the process of a search, ex.
- Linear search in list of names.

Linear search:

- Average search time = (n+1)/2 where n is the number of rows
- Maximum search time = n

Binary Search: Alphabetical Ordered Names

Since we have a sorted list, we can use a Binary Search or Divide and Conquer.

- Begin in the middle of table (n/2) +1 – is this the row we are seeking? Go above or below – after examining a single row we can eliminate half of the possibilities.

- Take the rows remaining (n/2) + 1 – is this the row we are seeking?

We can once again eliminate half of the remaining rows.

And so on . . .

Average search time:

- Average = log2(n)
- Maximum search time is log2(n)

Conclusion: Linear search versus binary search: massive improvement if we order the data using indexes.

```
39  -- Create Index for Restaurants base on Name
40
41  CREATE INDEX idx_res_name
42  ON RESTAURANT(restaurant_name);
43
44  -- Create Index for Person base on Name
45
46  CREATE INDEX idx_person_name
47  ON PERSON(person_name);
```