***Systems and database Admin Assignment - 40%***

***Creating a new database –***

Step 1: Create a new database called stockbroker. First, accessing the superuser postgres by the password is postgres.

“sudo su postgres”

Step 2: is to initialise the user for Postgres. This configures the user environments.

“source .bash\_profile”

Step 3: Start the Postgres database. We must use the command below.



Step 4: Create a new database called stockbroker.



Step 5: To initialise and connect the stockbroker database.



Step 6: Populate the stockholder database with tables.

* Companies: contains three keys Internal\_ID, Company\_Name, Annual\_Revenue

Text

Description automatically generated

* Traders: contains three keys Staff\_ID, Trader\_Name, and Join\_Date.

Text

Description automatically generated

* Portfolios: contains two keys Portfolio\_ID

Text

Description automatically generated

* Prices contain two keys Internal\_ID, Date\_Price, Stock\_Value. Internal\_ID is a foreign key from Companies.

Text

Description automatically generated

* Positions: Contains four keys Internal\_ID, Portfolio\_ID, Current\_Stock, Submitted\_date. Internal\_ID is a foreign key from Companies. Portfolio\_ID is a foreign key from Companies.

Text

Description automatically generated

We have now created all the necessary tables with the keys they need individually

A screenshot of a computer screen

Description automatically generated with medium confidence

***Security***

In the database, security considerations are always the top concern. It refers to the various measures taken to ensure the database is protected from internal and external threats. Security in the database includes protecting the database itself, its data, its management, and the various applications that access it. Three levels of security are database: where the data lives, access: controlling allowed access to data or systems containing them, and perimeter: who can and cannot get into the database. For this report, we will focus on a few examples of each.

Insider threats include three different kinds, comprised users, malicious users, and careless users.

Starting with compromised users are users whose access is leveraged by a third party. This could be through viruses, malware, or even social engineering ploys.

Malicious users are the classic insider threat, embezzler, and disgruntled employees who demonstrate malice might be a soon-to-be ex-employee who starts pulling valuable sales leads or other intellectual property to help him in his next job, an activist or whistleblower, acting on personal principles of a higher good.

Many compromised users were once careless users, but careless users expose their organisations to outlier attacks as well. They may be negligent in clicking on every link that comes their way, failing to exercise the most basic protection of their credentials and devices.

One of the most fundamental security principles is the principle of least privilege. This states that a user should only be granted the minimum required permissions. This prevents the user from accessing/executing anything they don’t directly need for their responsibilities.

For example, we have a user jack whose sole responsibility is to insert data into the trader’s table. Therefore, he should only be granted insert abilities to the table.



Groups and roles can be used to simplify the control of database access further. Groups are used to apply permissions to a list of users, while roles are used to associate subject privileges and permissions with an application. A group is an identifier that can be used to apply permissions to a list of users associated with the identifier.



Now all you need to do is add permissions to a group once and whoever joins this group now has the same permission.

***Audit***

The audit is an important compliance component that must be set up properly but quickly for your database to be more efficient and secure. Auditing allows the administrator to monitor and traceback any actions, queries, or changes that occur in the database. There are five main important auditing functions that can be implemented.

1. User and Authentication

This is the point of entry for any person trying to access the database from inside or outside the company. Any user with a high level of privilege may be able to alter or extract sensitive information from the database.

Auditing these activities gives the ability to monitor and identify data breaches before it is too late, or at least to implement better security configurations to stop these recurring threats from happening.

1. Database Objects

Database objects that hold data, functions or logic that define the functionality of the system, and people users with permissions and access to these objects can all manipulate the structure of the system. These can lead to massive data breaches or corruption on a continuous basis.

Auditing should be enabled for all-important tables, views, database links, functions, and runtime logical flows that control or affect the database.

1. Data

The most critical part of any database is the data it holds. In any database, there are usually many users who might be able to manipulate this data. Therefore, it is highly important that all confidential and important data should not able accessible to any unauthorised users.

The identification and monitoring of users, time data, and any changes can help the database to comply with strict regulations for things such as banks, cloud providers, and government institutions.

1. Network

Data all around the world is moved and transferred across networks. Auditing the network connection will help you understand the large amounts of data, and it will also help identify the network resource requirements needed for better configurations for the system and the database.

Additionally, when moving data across networks, the data can be vulnerable to snooping. Therefore, it is recommended that the data be encrypted during transit.

To enable auditing and to even perform some simple auditing on the system, follow the below commands.

Enabling Auditing :

1. To enable auditing on the database, the first step is to access the postgresql config file in the VM.



1. Find the Reporting and logging Section. Everything there will be commented out. This means no logging/reporting is currently active.
2. The first change to the config file is uncommenting “log\_destination” and setting it to equal ‘csvlog’.

Graphical user interface, text

Description automatically generated

1. Next, uncomment and change “logging\_collector” to on.

Graphical user interface, text

Description automatically generated

1. Uncomment “log\_directory” and “log\_filename”.

Graphical user interface, text

Description automatically generated

In the end, the postgresql.conf file Reporting and Logging section should look like this.

Graphical user interface, text

Description automatically generated

Next in the config file is the authentication. Here we can encrypt the password to use either scram-sha-256 or md5. We will use the default SHA-256 encryption because the output size is double that of md5, with the only downside of using it being that it is slower. However, this shouldn’t impact performance enough not to use it.

Text

Description automatically generated

Next up are the connections. In the postgresql.conf, we can see the default listening address and the max number of connections which is currently set at 100 connections. They also exist a superuser\_reserved\_connections, which is at a default of 3. The max number of connections should be restricted and lowered to a more reasonable number by the admin. This can help to prevent the database from making any new connections to new systems unless the admin decides to add a new connection.

Graphical user interface, text

Description automatically generated

We are able to view the current connections of the database by looking inside the pg\_hba config file.



This file controls which hosts are allowed to connect, how clients are authenticated, which PostgreSQL user names they can use and what database they can access.

Text

Description automatically generated

***Performance Optimisation***

The goal of tuning database performance is to minimise the response time to queries by making the best use of system resources. The best use of these resources is to minimise network traffic, disk I / O, and CPU time. This goal can only be achieved if you understand the logical and physical structure of the data, the applications used on the system, and how conflicting use of the database can affect performance.

1. Keep statistics up to date

Table statistics are used to generate optimal [execution plans](https://blog.quest.com/sql-server-execution-plan-what-is-it-and-how-does-it-help-with-performance-problems/). If the performance tuning tool uses out-of-date statistics, the plan won’t be optimised for the current situation.

1. Don’t use leading wildcards

Leading wildcards in parameters force a full table scan, even if there is an indexed field inside the table. If the database engine must scan all the rows in a table to find what it’s looking for, the delivery speed of your query results suffers. Other queries may suffer as well since scanning all of that data into memory will cause the CPU utilisation to spike and not allow other queries any time in memory.

1. Avoid SELECT \*

This tip is particularly important if you have a large table (think hundreds of columns and millions of rows). If an application only needs a few columns, include them individually instead of wasting time querying for all the data. Again, reading extra data will cause CPU utilisation to spike and memory to be thrashed. You should check the [Page Life Expectancy](https://www.spotlightcloud.io/blog/monitoring-page-life-expectancy-in-sql-server) (PLE) to make sure you are not having this issue.

1. Use constraints

[Constraints](https://blog.quest.com/exploring-the-different-constraints-in-sql-server/) are an effective way to speed up queries and help the SQL optimiser come up with a better execution plan, but the improved performance comes at the cost of the data requiring more memory. The increased query speed may be worth it depending on the business objective, but it’s important to be aware of the price.

***Backup / Recovery / Availability Policy***

A backup policy is a predetermined schedule by which information from commercial applications such as Oracle, Microsoft SQL, mail server databases, and user files is copied to disk and/or tape to ensure data recovery in the event of accidental data deletion, data corruption or any other action. System failure. Policies typically have a default protection scheme for most servers in an environment with additional policies for certain critical applications or data.

Simply put, data in any format needs to be backed up and archived regularly for use in case data recovery, or recovery is needed. Although this process seems simple, many organisations today do not follow these simple steps.

Some of the main features to include in your backup/recovery policy are the following:

1. Maintaining regular backups o the database during off-peak hours to avoid performance delays which can affect the availability of the database.
2. Backups are to be stored off-site, not at the primary location of the database, such as a single drive or single business address. Options could be the cloud or off-premise backup server.
3. Make sure all backups are encrypted to prevent data integrity. For example, cloud providers now offer encryption services.
4. Restricted access to backups to help prevent and minimise unwanted intrusions.

All the above can help in remedying data loss due to:

1. Accidental file deletion.
2. Viruses, ransomware, and malware.
3. Physically damaged or lost computers/equipment.
4. Power failure can cause corruption.
5. Building fires or natural disasters.

A simple technique to backup manually.



The command above extracts the PostgreSQL database into an SQL script file. The pg\_dump command can be used with other filters to dump all the tables, a specific table, all the objects in the database, and so. A key feature of pg\_dump is that it runs concurrently; therefore, it does not block any readers or writers