****

**Airport System**

**Technical Report**

**Database**

**10204282**

**A/618/7400**

**Section (2)**

**Submitted by**

Hamza AL-Risheq

**Submitted to**

Dr.Rami Ibrahim

**summer 2022 – 2023**

Table of Contents

[Introduction: 2](#_Toc144379522)

[Physical Schema: 4](#_Toc144379523)

[Part 1: Database Development 5](#_Toc144379524)

[1.1 Database Overview 5](#_Toc144379525)

[1.2 Security 10](#_Toc144379526)

[1.3 User Interface 12](#_Toc144379527)

[1.3.1 Flowchart and Data Movement Diagrams 12](#_Toc144379528)

[1.3.2 Interfaces Development 13](#_Toc144379529)

[Part 2: Maintenance 14](#_Toc144379530)

[2.1 Database recovery & backups 14](#_Toc144379531)

[2.1.1 Important things must be done after creating the backup 16](#_Toc144379532)

[2.2 Database maintenance in general 16](#_Toc144379533)

[2.2.1 Explanation of Database Maintenance 16](#_Toc144379534)

[2.2.2 Key Aspects of Database Maintenance: 16](#_Toc144379535)

[2.2.3 Importance of Database Maintenance 17](#_Toc144379536)

[Part 3: Testing 18](#_Toc144379537)

[3.1 Data Validation 18](#_Toc144379538)

[3.2 Output Validation 24](#_Toc144379539)

[3.3 Security Validation 28](#_Toc144379540)

[3.4 GUI Validation 28](#_Toc144379541)

[3.5 Assess whether meaningful data has been extracted 30](#_Toc144379542)

[3.6 Assess the effectiveness of testing 31](#_Toc144379543)

[Part 4: Evaluation of database solution 33](#_Toc144379544)

[4.1 Effectiveness of the database solution based on user and system requirement 33](#_Toc144379545)

[4.2 Suggested improvements 35](#_Toc144379546)

[4.3 Evaluation based on improvements needed 36](#_Toc144379547)

[References 38](#_Toc144379548)

# Introduction:

In response to surging passenger demand and the limitations of Amman Civil Airport, Queen Alia International Airport (QAIA) emerged in 1983 as the solution to accommodate the exponential growth in travellers. This development was driven by the Jordanian Ministry of Transportation, investing JOD 84 million to construct QAIA with a capacity for 3.5 million passengers annually. QAIA swiftly transformed into Jordan's primary international gateway and a key layover hub for Middle Eastern operations.

QAIA's evolution was characterized by the establishment of dedicated departments to manage its multifaceted operations. Labor and wage administration, flight safety, information system support, passenger transportation, fuel services, and accounting were distinct units within the airport structure. Employees within these departments were equipped with comprehensive profiles featuring essential details such as names, emails, job titles, and phone numbers, fostering efficient communication.

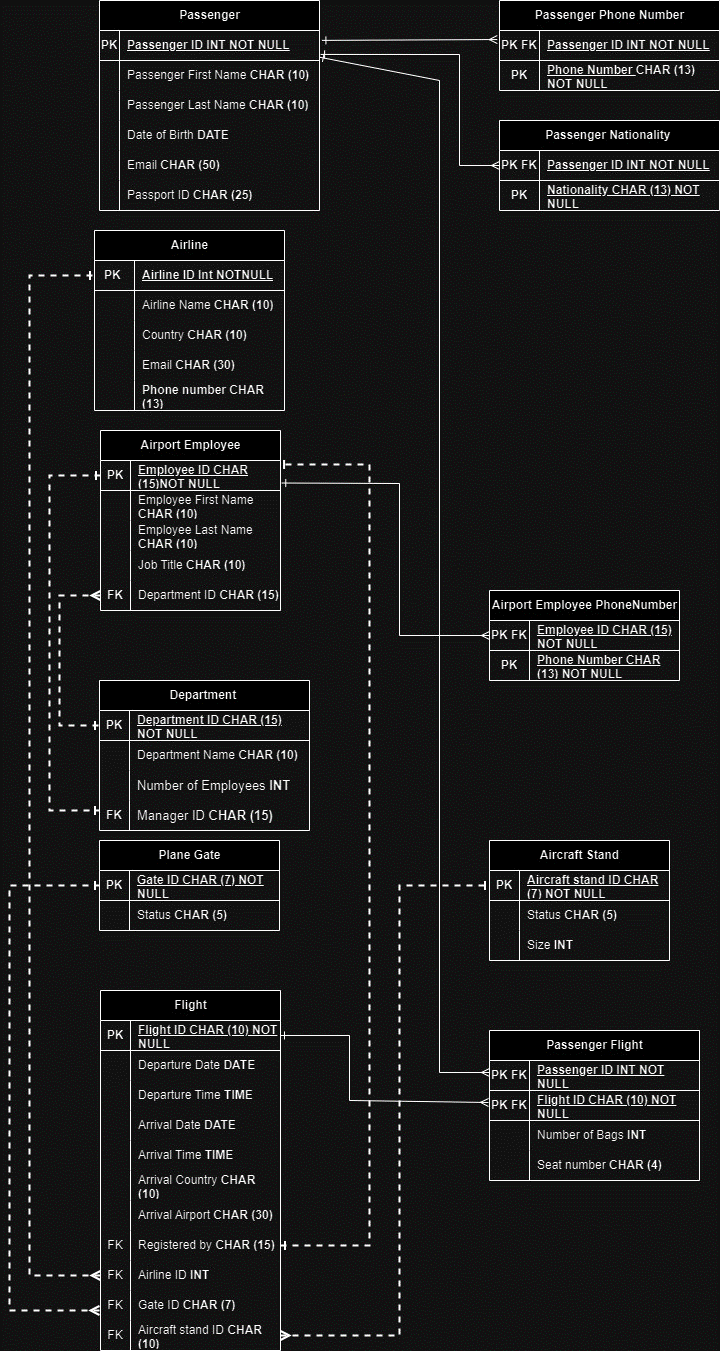
At the heart of QAIA's functioning were airlines and flights. Each airline, identifiable by a unique ID, name, country, email, and phone number, contributed to the airport's diverse network. Flights, managed by dedicated employees, were defined by departure and arrival details, destinations, and registration information.

Aircraft stands and gate assignments were integral components of QAIA's infrastructure. Aircraft stands, denoted by unique numbers and statuses, facilitated the parking of flights. Gate assignments streamlined passenger boarding, featuring unique IDs and statuses.

Passengers formed the core of QAIA's operations. With unique IDs, passengers' profiles encompassed personal details like names, birthdates, passport IDs, nationalities, and contact information. Reflecting global diversity, passengers could have multiple nationalities and phone numbers. Their interactions extended beyond individual flights, with the ability to reserve multiple flights, each associated with specific baggage and seat assignments.

In the subsequent sections of this technical design, we delve into the intricate architecture and underlying technologies of the Queen Alia International Airport Management System. This document outlines the methodologies, databases, and systems that synergize to drive QAIA's seamless operations and transformative role in modern air travel.

# Physical Schema:



# Part 1: Database Development

## 1.1 Database Overview

|  |  |  |
| --- | --- | --- |
| **Table** | **Name** | **Description** |
|  | Passenger Table. | The Queen Alia International Airport Management System's database relies heavily on the "Passenger" table. In order to provide effective administration and seamless interactions throughout the travel process, it acts as the central store for crucial passenger information. This table includes vital information for each passenger's travel as well as important characteristics that help identify them specifically. Also, A smooth and customized experience for travellers going through Queen Alia International Airport is greatly facilitated by the passenger table. This table allows airport workers to efficiently manage and attend to travellers’ requirements from check-in to boarding by precisely recording and storing passenger information, hence improving overall travel efficiency and satisfaction. |
|  | PassengerPhoneNumber. | So as to keep in touch and connect with passengers effectively during their journey, this table is devoted to the storing of passenger phone number data. |
|  | PassengerNationality. | To make it easier to verify identities, meet with regulations, and handle passengers effectively, this table keeps track of the many nationalities that travellers may have. |
|  | Department. | The numerous activities taking place at Queen Alia International Airport are expertly coordinated thanks in large part to the "Department" table. This table facilitates systematic management, efficient operation, and precise resource allocation across the airport by recording key departmental attributes. |
|  | AirportEmployee. | An essential component of the airport's operations, the "AirportEmployee" table makes effective personnel management possible. This table facilitates efficient communication, responsibility distribution, and job collaboration by collecting and arranging personnel information. The "AirportEmployee" table is essential to the smooth operation of Queen Alia International Airport, whether it is for controlling flights, guaranteeing passenger safety, or maintaining various departments. |
|  | AirportEmployeePhoneNumber. | A key factor in promoting effective communication and coordination among airport staff is the "AirportEmployeePhoneNumber" table. The table guarantees that pertinent parties may interact with one another easily by keeping track of employee-specific phone numbers, which improves collaboration, responsiveness, and overall operational performance at Queen Alia International Airport. |
|  | Airline. | The "Airline" table is crucial to simplifying how the airport interacts with different airlines and runs its operations. This table guarantees that airlines can be found, engaged, and managed effectively inside the airport's ecosystem by preserving accurate and up-to-date information. It benefits the overall effectiveness, coordination, and customer service of Queen Alia International Airport by acting as a helpful reference point for airport workers. |
|  | PlaneGate. | The PlaneGate table is essential for ensuring the smooth flow of planes and travellers through the airport. This table helps airport workers make educated judgments about gate allocations by keeping accurate records of gate assignments and statuses, reducing congestion and improving operational efficiency. The PlaneGate table guarantees a coordinated and smooth experience for both airline operators and passengers from flight arrivals through departures, adding to Queen Alia International Airport's overall efficiency. |
|  | AircraftStand. | The AircraftStand table is crucial to maintaining the airport's physical infrastructure. This table aids in the effective distribution of parking spots, the planning of maintenance, and the general coordination of aircraft movements by collecting and arranging data on individual aircraft stands. This improves operational effectiveness, ensures the safety and ease of aircraft operations, and helps Queen Alia International Airport run smoothly. |
|  | Flight. | The Flight table contains the essential features that collectively characterize each flight in the system. These characteristics offer a thorough perspective of the flight information, assisting in the efficient planning, supervision, and execution of flight operations.  The precise recording of flight information in the Flight table aids in the most effective management and coordination of flights. This table enables smooth flight operations, precise scheduling, and adherence to airport procedure by include crucial elements including departure and arrival times, destinations, and accountable workers. At Queen Alia International Airport, the detailed information in the trip table helps to increase efficiency and customer satisfaction during the trip. |
|  | PassengerFlight. | Travel experiences are made simple and tailored because to the PassengerFlight table's ability to record relationships between passengers and flights. The table enables effective boarding operations and passenger services by preserving clear relationships between specific individuals and their associated flights, luggage allowances, and seating assignments. Passengers using Queen Alia International Airport benefit from improved overall travel experiences because to this efficient cooperation. |

|  |  |  |
| --- | --- | --- |
| **View** | **Name** | **Description** |
|  | Employee\_Details | The "Employee\_Details" view combines data from the "airportemployee" and "department" tables. It includes columns such as EmployeeID, EmployeeFirstName, EMployeeLastName, DepartmentID, DepartmentName, and MangerID. This view offers a convenient way to access employee details alongside their respective department information in a single query, without the need to perform a manual join each time. |
|  | flight\_details | The "flight\_details" view merges data from the "flight," "airline," and "planegate" tables. It provides essential flight information, including FlightID, Registeredby, ArrivalAirport, AirlineID, AirlineName, GateID, and Status. This consolidated view simplifies the retrieval of flight-related details along with associated airline and gate information, enhancing the efficiency of querying such data. |
|  | passengers\_flights\_details | The "passengers\_flights\_details" view consolidates information from the "passenger" and "passengerflight" tables. It includes columns like PassengerID, PassengerFirstName, PassengerLastName, PassportID, FlightID, and NumberOfBags. This view provides a concise way to access passenger and flight details simultaneously, facilitating data retrieval without manual joining of tables. |
|  | passengers\_phonenumbers | The "passengers\_phonenumbers" view combines data from the "passenger" and "passengerphonenumber" tables. It provides a concise overview of passenger details including PassengerID, PassengerFirstName, PassengerLastName, PassportID, and associated PhoneNumbers. This view facilitates easy access to passenger information along with their contact details, streamlining the process of retrieving this information without requiring manual joins each time. |

|  |  |  |
| --- | --- | --- |
| **Procedure** | **Name** | **Description** |
|  | GetEmployeeDepartmentSummary | The "GetEmployeeDepartmentSummary" procedure is designed to retrieve a summary of employee counts for each department within an airport. It utilizes a `LEFT JOIN` operation between the `department` and `airportemployee` tables, connecting them based on the common `DepartmentID` field. The procedure then employs the `GROUP BY` clause on the `DepartmentName` column to group the results by department names. Within each group, it calculates the number of employees using the `COUNT (ae.EmployeeID)` aggregate function. The result is a summary that displays the name of each department alongside the corresponding count of employees working in that department. This procedure provides valuable insights into the distribution of employees across different airport departments. |
|  | GetAvailableAircraftStands | The "GetAvailableAircraftStands" procedure is designed to retrieve a list of available aircraft stands within an airport. It accomplishes this by querying the `aircraftstand` table using a `SELECT` statement. The query specifies that only the `AircraftStandID` values should be retrieved from the table where the `Status` column is set to 'Avail', indicating that the aircraft stand is available for use. This procedure is useful for quickly obtaining information about the aircraft stands that are currently unoccupied and ready for incoming flights or aircraft. |
|  | UpdateFlightStatus | The `UpdateFlightStatus` procedure is designed to update the arrival country status of a flight in the database. It takes two input parameters: `p\_FlightID` (representing the FlightID of the flight to be updated) and `p\_Status` (representing the new arrival country status). The procedure utilizes an SQL `UPDATE` statement to modify the `ArrivalCountry` column of the `flight` table, setting it to the provided `p\_Status` value. The update is performed for the row where the `FlightID` matches the provided `p\_FlightID`. This procedure allows for efficient and controlled updates to flight records, specifically their arrival country status. |
|  | GetPassengerFlightHistory | The `GetPassengerFlightHistory` procedure is designed to retrieve the flight history of a specific passenger based on their PassengerID. It takes an input parameter `p\_PassengerID`, which is the unique identifier of the passenger for whom the flight history is being retrieved. The procedure uses an SQL `SELECT` statement to gather information from two tables: `flight` and `passengerflight`. By performing an inner join on the `FlightID` column, the procedure combines relevant flight details such as `FlightID`, `DepartureDate`, `ArrivalDate`, and `ArrivalAirport` from the `flight` table with information about the number of bags (`NumberOfBags`) associated with the passenger on each flight from the `passengerflight` table. The result of this procedure is a summary of the passenger's flight history, including key flight information and baggage details for the flights they have been on. |

## 1.2 Security

|  |  |  |  |
| --- | --- | --- | --- |
| **User name** | **Privilege Command** | **Description** | **Screenshot** |
| department\_manager | GRANT SELECT, INSERT, UPDATE, DELETE ON employee\_details TO department\_manager; | This command gives the user 'department\_manager' access to perform specific actions on the 'employee\_details' database. Some of these duties include selecting data, adding new records, modifying existing records, and deleting items from the database. This query provides the department manager the necessary access to manage and interact with the personnel information stored in the chosen table. |  |
| GRANT SELECT ON airportemployee\_phonenumber TO department\_manager; | The query gives the airportemployee\_phonenumber table SELECT query execution permission to the department\_manager user. This indicates that the department manager has access to and may receive phone number information from the designated database, giving them the data, they need to carry out their duties or tasks. |  |
| Airline\_Company\_ | GRANT SELECT, UPDATE ON flight TO Airline\_Company\_; | The Airline\_Company\_ have the access to the flight table using both SELECT and UPDATE. This permits the airline\_Company\_ user to access and modify from flight table as required. | ­­ |
| GRANT SELECT ON passengers\_flights\_details TO Airline\_Company\_; | The Airline\_Company\_ user have accessed the passengers\_flights\_details view using SELECT. This permits the Airline\_Company\_ user to access passengers\_flights\_details view as required. |  |

## 1.3 User Interface

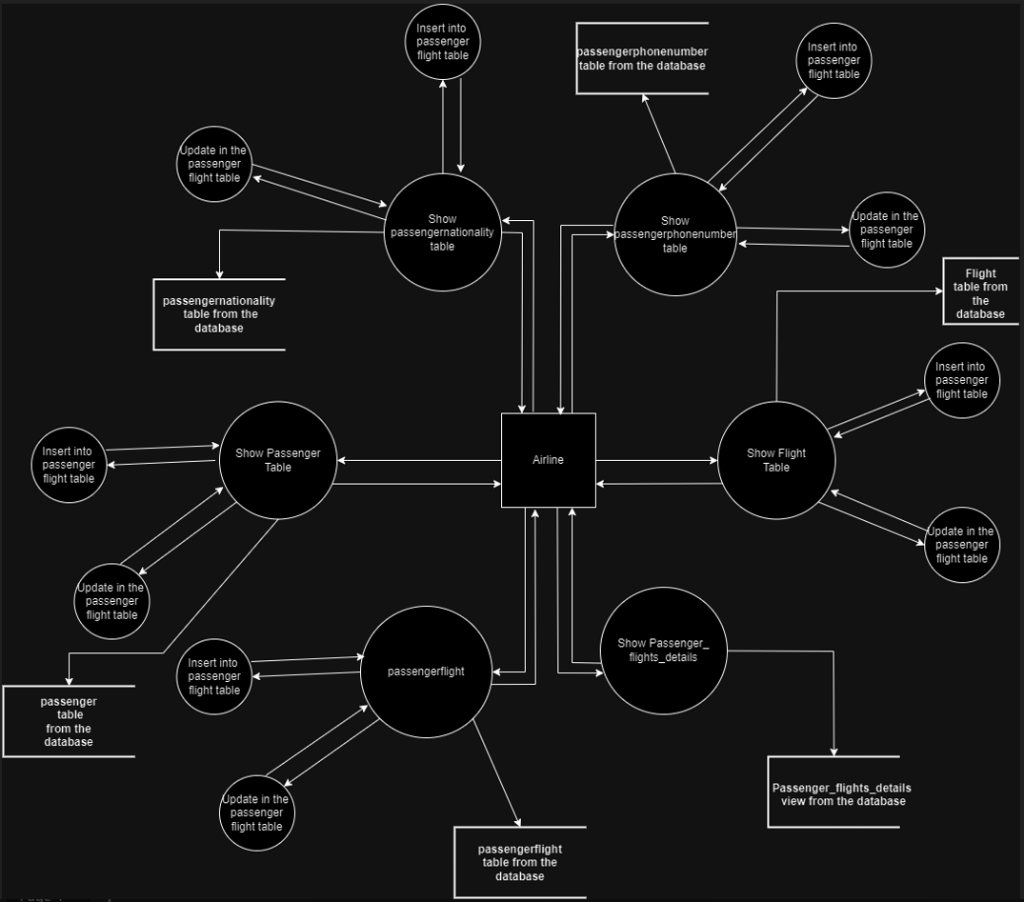
### 1.3.1 Flowchart and Data Movement Diagrams

#### 1.3.1.1 Flowchart

A computer screen shot of a computer flowchart

Description automatically generated

#### 1.3.1.2 Data Movement Diagrams



### 1.3.2 Interfaces Development

|  |  |  |  |
| --- | --- | --- | --- |
| **Page ID** | **Title** | **Description** | **Screenshot** |
|  | Passenger | The passenger table will be displayed on this page, which also offers various capabilities like delete, update, add, print, save, export, and search in the passenger table. So, this page will display the passengers in the airport. |  |
|  | Passenger Nationality | In this page the passenger nationality table will be displayed on it the user can delete, update, add, print, save, export, and search in the passenger nationality table. So, this page will display the nationality of the passengers in the airport. |  |
|  | Passenger phonenumber | In this page the passengerphonenumber table will be displayed on it. The user can delete, update, add, print, save, export, and search in the passengerphonenumber table. So, this page will display the phone number of the passengers in the airport. |  |
|  | Flight | In this page the flight table will be displayed on it. The user can delete, update, add, print, save, export, and search in the flight table. So, this page will display the flights with that was registered in the airport. |  |

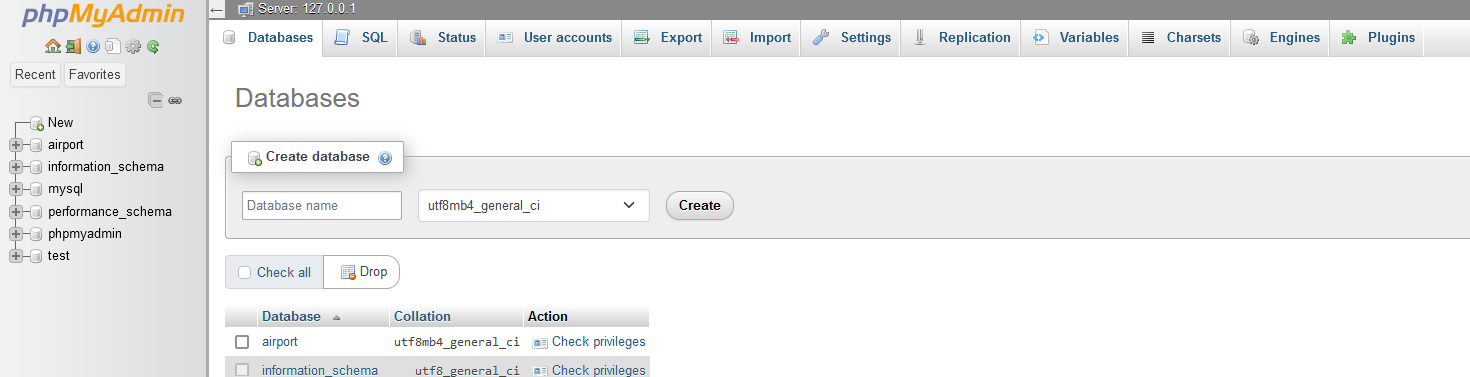
# Part 2: Maintenance

## 2.1 Database recovery & backups

Maintaining backups of a database's data and schemas is a fundamental information technology activity that serves as insurance against data loss, corruption, and other unanticipated circumstances. Databases act as stores for vital information that businesses and individuals rely on for their operations, decision-making, and record-keeping in a world driven by digital information. Databases are not immune to dangers, though; they include human error, software error, cyberattack, and hardware failure, all of which can result in data loss or damage. Therefore, making a database backup method so crucial is data security. It's crucial to create a clear distinction between vital data before and after an attack, such as a ransomware attack on a business. Recovery provides a fresh start for the future since it allows one to return to a time before the attack even took place. Recovery from backups to fresh systems gives a chance to avoid the assault if a ransomware attack encrypts an organization's data and a skull-and-crossbones appears on the screen. Organizations cannot recover without a backup, and if they cannot recover, they are unable to continue operating their firm. As result of that the database backups will reduce the cost and damages that can be affected from losing the database.

Based on the advantages of the database PHPMYADMIN provides very critical service that we can get a backup for the database we create. How we do that by following the next steps:

* Go to http://localhost/phpmyadmin in your browser once it is open.
* A screenshot of a computer

  Description automatically generatedFrom the left sidebar, pick the database you wish to backup.
* Select "Export" from the top menu's tabs.
* A screenshot of a computer

  Description automatically generatedSelect the export strategy. You can pick "Custom" for more complex choices, but "Quick" will only provide the fundamental parameters.
* A screenshot of a computer

  Description automatically generatedSelect "Export" from the menu. Your database structure and data will be generated in a SQL file as a result of this.
* The SQL file should be saved to a safe place on your computer or an external storage drive.

### 2.1.1 Important things must be done after creating the backup

* Depending on the importance of my data and how frequently it changes, I plan backups on a regular basis.
* I have to keep the backup files in a safe area apart from my working environment.
* Consider utilizing scripts or other third-party solutions to automate the backup procedure.
* To make sure everything functions as planned, test your backup and recovery procedure in a controlled setting.
* For high availability, I have to use redundancy and disaster recovery methods that go beyond backups, such as replication and failover settings.
* Secure your backup by using 3-way authentication to access the backup.
* I must reduce the number of the employees can access the backup to the minimum.

## 2.2 Database maintenance in general

### 2.2.1 Explanation of Database Maintenance

For the purpose of to guarantee a database system's optimum performance, dependability, and integrity, a collection of planned operations and duties is referred to as database maintenance. Databases need constant maintenance to operate successfully, much like you maintain your automobile to keep it operating well. A variety of tasks are included in database maintenance with the goals of preventing data loss, enhancing performance, and guaranteeing the availability of data for users and applications.

### 2.2.2 Key Aspects of Database Maintenance:

* Keeping the database's data accurate, consistent, and uncorrupted is known as data integrity. Regular data validation, integrity checks, and error repair are examples of maintenance duties.
* Performance optimization involves keeping an eye on and improving the database's performance. This involves actions like query tweaking, index optimization, and performance tracking to locate bottlenecks.
* Backup and recovery: Performing routine database backups to protect against data loss due to hardware failures, user mistake, or other unanticipated circumstances. Test the restoration procedure as well to make sure backups can be used effectively for recovery.
* Access restrictions, authentication, and permission processes must be enforced for security reasons to guarantee that only authorized users may access the database. Database maintenance also includes regular security audits and upgrades to guard against vulnerabilities.
* The exporting and importing of data are one type of database task. This is important because it enables you to repair problems that have harmed your systems, including data corruption or hardware breakdowns. You may use it to transfer or backup portions of your database to another system.
* Database systems transmit their data to a different location at regular intervals as an automated backup. Backups, however, are insufficient for database maintenance, and several extra measures are required to keep the system functioning.
* Regular Reviews: Consistently carrying out routine checks for abnormalities, inconsistencies, or mistakes in the database. These tests aid in early problem detection and resolution.
* Disaster Recovery Planning: The process of creating and keeping up a thorough disaster recovery plan to make sure that data can be retrieved in the event of major calamities such hardware failures, natural catastrophes, or cyberattacks.
* Documentation: Keeping precise and up-to-date documentation of the setups, maintenance methods, and recovery procedures for the database.
* Monitoring the hardware that holds the database in order to spot any breakdowns or performance problems. This might entail keeping an eye on things like CPU utilization, memory usage, disk I/O, and more.

### 2.2.3 Importance of Database Maintenance

* + - Reduced expenses: Database maintenance reduces costs by eliminating data loss, reducing downtime, and enhancing performance. The company may avoid the costs of data recovery, system breakdowns, and lost productivity.
    - Keeps Companies Up to date; Organizations are working extremely hard to keep up with the rapid pace of technological innovation. And the reason why is that staying current offers them a stronger foundation. If you don't keep up with the newest trends, everything you're selling ages and is no longer up to date with expectations. Your data is always up to date if your database is regularly maintained. Your ability to serve your audience is improved.
    - Protection Against Dangerous Threats; Without being aware, your system could be vulnerable to online attacks. If you don't stop even the tiniest risks from growing, they can become more serious. Regular database maintenance helps to minimize downtime and protect against harmful attacks. It assists you not only in identifying the hazards but also in gaining knowledge on how to counter them.
    - Encourages Effective Database; Database maintenance removes unnecessary or pointless data to increase database efficiency. This could lead to decreased processing overhead on some platforms. Making sure databases are performing at their best is essential. System bottlenecks will make your workflow more labour-intensive and cost more to operate.

# Part 3: Testing

## 3.1 Data Validation

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Type** | **Description** | **screenshot** |
|  | All cases of PK | Each table's primary keys must be distinct for each entry. |  |
| Null values shouldn't appear in primary keys. |  |
|  | All cases of FK | Verify that values from a child table's foreign key column exist in the parent table's main key column. This ensures that the connections are appropriately maintained. |  |
| Verify that changes made to the parent table successfully propagate to the child table if cascading updates or deletes are enabled for the foreign key relationship. |  |
| Suring that the foreign keys in one table match the primary keys they refer to in another table. If they don't match, it can cause data problems. By using simple SQL queries, you can find these mismatches and fix them, ensuring my data connections are correct and reliable. |  |
| Check that the primary key column of the parent table contains the value from the child table's foreign key field. |  |
| Whenever a record in the parent table is removed, the cascade delete referential action in a relational database management system automatically deletes related records in child tables. By guaranteeing that no orphaned records are left behind in child tables, this step preserves the integrity of data linkages. As result when we go to the table that contain foreign key we will find out that the PK was delete from the parent table is delete in the child table to. |  |
|  | Unique | Through the preventing repetitive inputs, this preserves the correctness and integrity of the data. It is implemented in the database using restrictions or indexes, and any attempt to insert duplicates results in an error. |  |
|  | Default | Default value validation guarantees that when a user does not supply a value for an attribute during data insertion, the database provides a specified default value automatically. This ensures data consistency by preventing NULL values from being inserted. |  |
|  | Not null | Assures that a particular property must have a value when data is inserted, avoiding the insertion of NULL values and preserving data completeness. |  |
|  | Check | To verify that the user does not enter an unexpected value as we can see in the number of bags attributes the passenger cannot have bags in minus. |  |

## 3.2 Output Validation

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Query Description** | **Screenshot (query + result)** | **Result validation** |
|  | This query will select a specific passenger id. | Query Statement    Results |  |
|  | This query will select all the department that there name contain letter ‘e’ | Query Statement  Results |  |
|  | This query brings the summation of the number of bags in the passenger flight table. | Query Statement    Results |  |
|  | the query retrieves all columns from the passenger table and combines them with corresponding rows from the passengerphonenumber table based on the matching PassengerID values. If a passenger has phone numbers associated with them, the query will return the passenger's information along with their phone numbers. If a passenger does not have any phone numbers, the query will still return the passenger's information, but with NULL values in the columns related to phone numbers. | Query Statement    Results |  |
|  | The query retrieves all columns for passengers from the passenger table who were born between the years 1980 and 1990. It uses the DateOfBirth column and the BETWEEN operator to filter the data based on the specified date range. The result will include all attributes of the selected passengers who meet this condition. | Query Statement    Results |  |
|  | The query retrieves all columns from the passenger table (with the alias f) and combines them with corresponding rows from the passengernationality table (with the alias n). The join is based on matching PassengerID values. If a passenger has associated nationality information, the query will return the passenger's information along with their nationality data. If a passenger does not have nationality information, the query will still return the passenger's information, but with NULL values in the columns related to nationality. The LEFT JOIN ensures that all passengers from the passenger table are included in the result, regardless of whether they have matching nationality records in the passengernationality table. | Query Statement    Results |  |
|  | The query calculates the average number of bags (NumberOfBags) for each passenger who is associated with flights. It retrieves the passenger's ID, first name, and last name from the passenger table, and also includes the NumberOfBags from the passengerflight table. The join operation ensures that only passengers who have flights are included in the result. The query provides a concise summary of passenger information along with the average number of bags they have across their flights. | Query Statement    Results |  |
|  | The query retrieves detailed information about passengers and their corresponding flights. It combines data from the passenger, passengerflight, and flight tables. The joins are used to establish relationships between these tables based on PassengerID and FlightID values. The query will return information about passengers and their associated flights, along with flight details, for the first three matching records. | Query Statement    Results |  |

## 3.3 Security Validation

**Note**: you need to test the given and not given privileges.

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **User Name** | **Description of privilege/no privilege** | **Screenshot (query + result)** |
|  | department \_manager | The user is able to check all the employees in the departments. |  |
|  | department \_manager | The user its not able to view the passenger information. |  |
|  | Airline\_Company\_ | The user is able to view the passengers\_flights\_details. |  |
|  | Airline\_Company\_ | The user is not able to delete anything from the flight table. |  |

## 3.4 GUI Validation

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Description** | **screenshot** | |
|  | The user (airline\_Company\_) have a permission to view the passengers\_Flights\_details view. |  | |
|  | The user (airline\_Company\_) have a permission to view and update on the flights table.  But this user does not have permission to delete anything form the table. |  | |
|  | The user (airline\_Company\_) have a permission to view and update on the passenger table. | Before updating the last name.    After updating the last name. | |
|  | But the user (airline\_Company\_) does not have permission to delete anything form the passenger table. |  | |
|  | The user (department\_manager) can only do view the employee\_details view. |  | |
|  | The user (department\_manager) can view the airportemployee\_phonenumber table. |  | |
|  | The user (department\_manager) can’t delete or edit anything in airportemployee\_phonenumber table. |  |

## 3.5 Assess whether meaningful data has been extracted

Finding significant insights through data analysis is similar to finding buried treasure. The process, however, continues with the essential step of determining if the retrieved data is indeed relevant and trustworthy. This crucial stage guarantees the accuracy of analysis findings, providing decision-makers with the knowledge they need to make wise decisions. It is impossible to overestimate the significance of evaluating significant data extraction because it serves as the cornerstone for making strategic decisions. Here are key reasons highlighting the importance of assessing meaningful data extraction, making wise decisions, Authenticating Analysis Goals, Keeping Biased, Conclusions at Bay, improving data Quality, Interpreting Context, and Solving Problems Effectively.

Based on the importance of assess whether meaningful data has been extracted I will assess my data based on the outputs and views that we can see in the tables above there we will find out that my data has been extracted is meaningful because my data alignment with objectives as a result of understanding the scenario that the clients provide me with it I built my database so when we test the database results with the scenario we will find out that the data meets the objectives. Also, my data is precise and reliable because when we select one of the tables to check the output will we find out that the data is accurate and understandable because everything is readable, specified, well-coded, and meets the objectives of the scenario we can see this in one of that tables in the database let’s take an example the passenger table we will find out that the table has 5 attributes each attribute has a specific meaning and the table has a primary key which is the passengerID and this primary key will help to extract the information about each passenger fast, and accurately without any confusion. Logical relationships If we take a review of the database that I built we will find out that the relationships between the tables are logical because the relationships meet the scenario and help to get the information or data we want directly and fast without problems. Also, we can see that the relationships between the tables have a real system feel. As a result, we can find out that my database is meaningful and the evidence of that, is that when using this database we can insert, delete, and update it based on the user and his privileges without any problems or any security problems. And we can check or select whatever we want based on the privileges the user has. If the user has access to read from the passengers table, he will be able to read the data easily without errors from the table. The last proof that my data is meaningful is that each attribute has the efficient and best type of data and restrictions. Additionally, the database's access restrictions make sure that users interact with the data by their rights, improving security and dependability. Cross-referencing the retrieved data with outside sources and speaking with subject matter experts to confirm its usefulness. This rigorous method creates a solid basis for effective decision-making, together with ongoing data monitoring and correction.

## 3.6 Assess the effectiveness of testing

In the field of software development, where accuracy and dependability are crucial, evaluating the efficacy of testing is a crucial procedure. Imagine diligently applying each brushstroke and detail to a masterpiece only to discover that the canvas conceals flaws underneath the surface. Similar to this, thorough testing of software reveals any possible faults, bottlenecks, and vulnerabilities present, guaranteeing that the finished product satisfies the highest requirements for functionality and quality.

The compass that developers, testers, and stakeholders use to navigate their path to provide software that not only operates but operates flawlessly, is effectiveness evaluation. This crucial stage involves a careful analysis of the testing strategies used, the breadth and depth of coverage, and the concrete results obtained. The significance of this evaluation cannot be emphasized given how deeply ingrained technology is in our daily lives. The complexity of evaluating testing efficacy is examined in this investigation, along with its importance, constituent parts, and the reverberations it has on the whole software development lifecycle.

Efficiency testing measures how many test cases were run per unit of time. The standard time measurement is in hours. It evaluates the amount of code and testing tools needed by an application to carry out a certain task.

It assesses the number of resources that were anticipated and the number that were used for testing. Everything revolves around doing the work quickly and efficiently. When assessing efficiency, tests take into account people, resources, equipment, procedures, and time. Measuring the effectiveness of the test procedures requires the creation of test metrics.

Effective test evaluation incorporates various important factors, including:

1. Thorough Test Coverage: The evaluation measures how thoroughly the testing of the program covers its functional requirements, business logic, user interfaces, performance, security, and other elements. Ample coverage makes sure that possible problems are found in all important areas.

2. Defect Detection and Resolution: The effectiveness of testing is determined by how well it can locate flaws. Teams are then able to prioritize and swiftly address urgent problems thanks to the evaluation, which analyzes the quantity, seriousness, and types of errors discovered during testing.

3. Validation of Requirements: The evaluation of testing effectiveness confirms if the program complies with the stated requirements. To do this, it must be confirmed that all features, functions, and user interactions meet the original objectives.

4. Performance and Scalability Testing: Performance and scalability testing is essential for apps that must handle numerous users and high loads. In this context, effectiveness evaluation assesses how effectively the application works under pressure to make sure it can handle actual usage scenarios.

5. Usability and user experience are key considerations. An evaluation of an application's effectiveness looks at how user-friendly, intuitive, and smooth the user interface is. This entails assessing variables including user satisfaction overall, responsiveness, and navigation.

6. Integration and Compatibility: The evaluation determines if the program is compatible with various hardware, operating systems, and browsers, as well as how effectively it interfaces with other systems. This guarantees a seamless user experience in a variety of settings.

Based on the importance of the testing I tested my database, and we can see that in the tables above there by simulating heavy user loads and stress-testing the system, I were able to identify potential bottlenecks and optimize my software's performance. So, I tested the user interface, and the database. So, for testing the database I confirmed the database's name, confirming that each database has adequate space allotted, checking the configuration of the database options. Also, for the database I checked column names for each table, column types for each table, NULL value checked or not, and whether a default is bound to correct table columns, an overview of rules for updating access rights and table names. And the stored procedure name, parameter names, and parameter type. Also, I used different type of select query statements to check the output of database and to check if there is any errors or unexpected outputs. And for the user interface I checked that the users will be able to view a specific data based on their privileges and they can delete or update also based on their privileges, and I checked that the frontend software (user interface) is connected correctly with the backend (database server).

So, I was able to ensure that the database is carefully checked for functionality, quality, and dependability. This assessment equips stakeholders and development teams to decide on software deployment in an educated manner, leading to a robust and trustworthy end product. Also, it will help to reduce the number of errors and problems which will lead to a reduction in the costs and budgets for fixing the database problems and will reduce the number of times that the database is down which will reduce the cost and increase the efficiency. So, testing helping to finding the drop backs of the system and unexpected error or inputs to reduce them. Therefore, the system will be more secure and efficient.

# Part 4: Evaluation of database solution

## 4.1 Effectiveness of the database solution based on user and system requirement

With consideration for both user and system needs, I designed and developed this database solution which was created to successfully satisfy the demands of contemporary airport operations. The database enables many stakeholders within the airport ecosystem with an emphasis on user ease and system efficiency. The database provides smooth data administration and well-informed decision-making by leveraging well-structured tables, views, and processes. This solution improves operational performance and data integrity by consolidating departmental personnel information and simplifying flight information and passenger information. The database offers a solid basis for managing the complexity of airport activities by carefully taking user demands and system performance into account.

The database solution I've designed effectively addresses both user and system requirements, providing a comprehensive and organized structure to manage airport operations seamlessly.

Example of the effectiveness of my database in meeting the requirements:

* One of the critical users in my database is the airport employees: We can find out that users can access based on their privileges and job titles to the database. Each employee has special information that can be uploaded to the database. Also, to increase the speed and efficiency of the database performance I created the “employee\_details” view simplifies data retrieval by presenting employee details along with their corresponding department information. So, this abstraction enhances user experience, simplifying the process of obtaining comprehensive employee profiles.
* Department Management: The database I developed allows the departmentalization of the airport and records crucial departmental information such as ID, name, manager, and personnel count. The "GetEmployeeDepartmentSummary" process meets user demands by delivering a count of employees a summary of the department's information, and information about the employees in the department, which will guarantee effective department administration.
* Information about airlines is kept in my database, including name, location, email address, and phone number. This permits communication with airline businesses that are present at the airport and supports user inquiries about them. Which meets the business requirements and helps the end users to understand and be able to interact with the database easily.
* Flight Management: The system records all relevant flight information, including departure and arrival times, registered personnel, and affiliated airlines. By streamlining user access to these facts, the "flight\_details" view improves flight management and monitoring capabilities. This will help to increase the efficiency of managing all flights in the airport without overlap in the information or misunderstandings.
* Passenger Records: Users may effectively handle passenger records, which include contact data, personal information, and passport numbers. By giving passengers' contact information, the "passengers\_phonenumbers" view streamlines data retrieval and enhances customer service and communication. So, the system I developed can record all the passengers with their information so we will be able to track their flights. As a result, we will be able to avoid any problems and be able to deal with a greater number of passengers in the same period.

I created users and privileges for each user assisted me to strengthen the security in my system. In addition, I have developed a lot of users to fulfill user requirements. Each user has his or her own privileges, as shown in the tables above. All the users within my system are created based on the user and system requirements, all of them were able to do their work correctly and normally and we can see that in the table above there.

Examples of the users that I created to meet the user requirements with their privileges:

* department\_manager has 2 main privileges first one privileges to SELECT, INSERT, UPDATE, DELETE ON employee\_details Table and SELECT ON airportemployee\_phonenumber table.
* Airline\_Company\_has many privileges to meet the user requirements:
  + SELECT, and UPDATE ON flight table.
  + SELECT, ON passengers\_flights\_details view.
  + SELECT, and UPDATE ON passenger table.
  + SELECT, and UPDATE ON passengernationality table.
  + SELECT, and UPDATE ON passengerphonenumber table.
  + SELECT, and UPDATE ON passengerflight table;

Through well-established linkages and limitations, the database architecture upholds data integrity. Constraints on foreign keys provide correct and consistent data across linked tables, upholding the accuracy of the data. Which system that I developed can protect the data integrity. Also, I used views like "employee\_details," "flight\_details," and "passengers\_phonenumbers" to ease data access for users while maintaining data security and integrity by abstracting complicated data connections. So, I increased the performance and efficiency of the system. I use stored procedures, such as "UpdateFlightStatus" and "GetAvailableAircraftStands," which encourage the structure of code in a modular manner. By isolating certain operations, this modularity improves the reusability, maintenance, and security of the code. Therefore, eliminating the need to parse and optimize SQL statements each time they are run, boosts speed. Procedures can also be improved for better execution strategies, resulting in quicker data extraction and manipulation. And procedures helped me to increase the data Security and Access Control. Because stored procedures provide a layer of security by allowing you to grant users access to execute procedures without giving direct access to underlying tables.

Now if we took a look at the performance of my system, we would find out that the database I developed is optimized for efficient data retrieval and storage, easy to interact with, well structured, and fast system. Indexed fields and normalized relationships minimize data redundancy and improve query performance.

The security of the system user requirements for data security is addressed through normalized table structures, foreign key relationships, views, and privileges are determined for each user, encrypting the backup of the database, and storing the backup in a safe place. This design minimizes and blocks unauthorized access and ensures the confidentiality of sensitive data.

And the database I developed and designed is a strong basis for scalability which provided by the database architecture, comprises normalized tables and well-defined relationships. The database is capable of handling growing data volumes, a massive number of inserted data, and accommodating new needs as airport operations expand.

As a result of everything, the database solution I created successfully balances user needs and system effectiveness to satisfy the changing demands of airport operations. It offers smooth data management and well-informed decision-making through well-designed tables, displays, and procedures. The system effectively handles passenger records, airline data, departmental summaries, flight information, and personnel data. While stored procedures provide modular code structure and improved system efficiency, views improve data access without compromising security. This solution serves as an excellent example of how user-centric design and efficient system administration may be used to improve data integrity and optimize airport operations.

## 4.2 Suggested improvements

A database's full potential must be unlocked by diving into cutting-edge tactics that go beyond the norm. Beyond the fundamental components, creativity, automation, and a proactive approach to problems are needed for database optimization. This trip travels through state-of-the-art methodologies, from intelligent monitoring to disaster recovery readiness, guaranteeing that databases not only satisfy current expectations but also stay flexible for future requirements. The secret to developing a strong and high-performing database environment that fosters organizational development and success is to embrace these cutting-edge tactics, and by looking forward to improving my database.

Suggested Improvements:

For the database, I would recommend the following improvements:

* Utilize caching strategies to keep frequently requested data in memory. This can speed up response times and greatly lessen the burden on the database.
* I would recommend using faster, better CPU, and RAM for the server which will increase the performance and speed of the database.
* Consider moving the database to a cloud platform that offers scalable resources, managed services, and improved performance.
* Encrypt the backup to ensure that no one can break the confidentiality of the data.
* Use more security measures such as firewalls for the server that contain the database, antiviruses, and IPS.
* Reduce the number of employees that can access the database.
* Use automated data validation tests to identify problems as they occur during data entry.
* Defragment the data.
* Avoid using \* with select statements because before a query is executed, SQL Server translates \* into all of the table's column names. Pass the names of the columns that are necessary in query results in place of doing this.
* Utilize index views. An indexed view can be defined for improved efficiency when an SQL query comprises several joins between tables that don't change regularly. A view that is physically stored as a table is called an index view. When a table that is utilized as part of the index view is modified, SQL Server automatically updates the index view.
* When necessary, use the Try...Catch Block method. T-SQL statements handle errors using a TRY...CATCH block. When working with transactions, always utilize TRY...CATCH blocks since a deadlock might result from a transactional fault.
* A minor adjustment at a time should be made to inquiries. In the long run, making too many adjustments all at once might be harmful. Making adjustments to the most expensive activities first and working backward is a more effective strategy.
* Implement stringent validation criteria to make sure that only accurate and legitimate data is entered into the database. This eliminates mistakes and discrepancies at the source.

## 4.3 Evaluation based on improvements needed

The database structure and information I gave and built demonstrate an admirable attempt to organize different entities inside the airport administration system. The links between entities and their characteristics are apparent in the schema. To illustrate the many components of the airport ecosystem, tables with names like passenger, flight, airline, and others have been developed. To create linkages between various entities, primary keys, and foreign keys have been suitably implemented, facilitating data retrieval and preserving data integrity.

My database has several qualities that support its efficiency and dependability in managing data linked to airports. Its hierarchical schema, where tables are intelligently arranged to represent various entities and interactions inside an airport management system, is one noteworthy feature. The database's general usefulness is improved by this logical organization, which makes data storage, retrieval, and maintenance easier.

My skillful application of primary keys and foreign keys is one of the strengths of my database. These keys create significant linkages between tables, promoting data integrity by averting contradictions and duplications. Additionally, my design demonstrates the application of normalization concepts, as shown in tables like passengerphonenumber. This normalized layout minimizes data duplication while improving storage efficiency.

I have effectively implemented indexing, a key speed enhancement strategy, in my database. I can guarantee that data retrieval is effective even when working with huge datasets by adding indexes to important columns. The addition of constraints, such as foreign key restrictions, strengthens data consistency and protects against unintentional data changes that can impair data quality.

My emphasis on referential integrity through foreign key restrictions is one feature that stands out. By maintaining the links between related data tables, these constraints guarantee that the data is reliable and correct across the database. This focus on the little things improves my database's overall dependability, which is essential for preserving the correctness and coherence of the data housed inside.

Both views and procedures are incorporated into my database architecture, which improves the system's overall usability and management. The addition of procedures, such "GetEmployeeDepartmentSummary" and "UpdateFlightStatus," helps to organize the code in a modular manner, allowing the encapsulation of complicated activities. By granting regulated access to particular database activities, these procedures enhance code reuse, transaction management, and security. Additionally, the "GetPassengerFlightHistory" function streamlines data retrieval for enhanced user experiences while providing insightful information about passenger flight history.

Furthermore, my database uses views like "employee\_details" and "passengers\_phonenumbers" to abstract and make it easier to retrieve data. These views provide a customized depiction of intricate connections, facilitating easier data interactions. Views are crucial in protecting data security because they limit access to sensitive information and allow for standardized data display throughout the company. My database's wise use of procedures and views demonstrates a well-considered strategy, enhancing data security, performance, and integrity while supplying an abstraction layer between the application logic and underlying data structures.

On the other hand, the usefulness of my database may be improved in a few specific areas, though. For improved readability and maintainability. To ensure that column names are consistent. Small discrepancies might be standardized, such as the capitalization of EmployeeFirstName and EMployeeLastName. To prepare for future data expansion, thoroughly examine data kinds and volumes. For the Email column, for instance, increasing the varchar size would allow for a range of email address lengths.

Also, I can improve my database by implementing a strong user authentication and authorization mechanism to improve security. This will make sure that users have the proper access rights and permissions to the information they require while still abiding by the concept of least privilege. And applying more procedures and views which will help to increase the security, performance, and efficiency of the database.

# References

Errorsays, E.F. "MySQL C.N.F. (2022). 11 Highly Effective Database Performance Tuning Techniques. [online] Available at: <https://cyberpanel.net/blog/11-highly-effective-database-performance-tuning-techniques/> [Accessed 25 Aug. 2023].

‌www.c-sharpcorner.com. (n.d.). Tips to Improve SQL Database Performance. [online] Available at: <https://www.c-sharpcorner.com/UploadFile/ff2f08/tips-to-improve-sql-database-performance/> [Accessed 25 Aug. 2023].

Hamilton, T. (2019). Database(Data) Testing Tutorial with Sample TestCases. [online] Guru99.com. Available at: <https://www.guru99.com/data-testing.html> [Accessed 25 Aug. 2023].

www.tutorialspoint.com. (n.d.). Database Testing â Techniques - Tutorialspoint. [online] Available at: <https://www.tutorialspoint.com/database_testing/database_testing_techniques.htm> [Accessed 25 Aug. 2023].

‌www.softwaretestinghelp.com. (n.d.). What Is Efficiency Testing And How To Measure Test Efficiency. [online] Available at: <https://www.softwaretestinghelp.com/efficiency-testing/> [Accessed 25 Aug. 2023].

www.ibm.com. (n.d.). *Database maintenance tasks*. [online] Available at: <https://www.ibm.com/docs/en/product-master/12.0.0?topic=database-maintenance-tasks> [Accessed 25 Aug. 2023].

Orange Matter. (2022). The Importance of Database Backups: A Guide to Costs, Benefits, and Risks. [online] Available at: <https://orangematter.solarwinds.com/2022/01/11/the-cost-benefit-and-risk-of-database-backups/> [Accessed 25 Aug. 2023].

www.ihp.sinica.edu.tw. (n.d.). Backup and Restore MySQL Databases. [online] Available at: <http://www.ihp.sinica.edu.tw/dashboard/docs/backup-restore-mysql.html> [Accessed 25 Aug. 2023].

‌