Student Name: Matheus Almeida

Student Number: 681679

# Coding Assignment Level 3

1. Be able to design a computer program.

1.1 Use flowcharts in the computer design process.

Here you can upload the flow chart separately or add it to your report documentation. Place it under this criteria.

* Attached Document Flight Booking Flowchart.

1.2 Use pseudo code in the computer program design process.

The same as above. Please place your pseudo code under this criteria.

* Attached Document Flight Booking Pseudocode

2. Be able to implement a computer program(s).

2.1 Determine, declare, and initialise variables to meet a given requirement.

Screenshot and explain where you have created variables (explain why you created them and show screenshots) **– DOES NOT HAVE TO BE ALL YOUR VARIABLES JUST SHOW A FEW DIFFERENT EXAMPLES**





* Variable tickets:

The variable tickets is determined by querying all the Ticket objects from the database using Ticket objects all.

This variable is declared and initialized in the line: tickets = Ticket objects all

The purpose of this variable is to hold the list of available tickets.

* Variable form:

The variable form is declared and initialized based on the request method using the TicketBookingForm.

If the request method is "POST," the form is initialized with the submitted data request.POST. Otherwise, if it's not a "POST" request, the form is initialized with no data None.

The purpose of this variable is to handle the ticket booking form.

* Variable feedback:

The variable feedback is created to store the feedback object retrieved from the database using the get\_object\_or\_404 function.

This variable is declared and initialized in the line: feedback = get\_object\_or\_404(Feedback, id=feedback\_id)

The purpose of this variable is to hold the feedback object associated with the given feedback\_id parameter.

2.2 Determine, declare, and initialise data structures to meet a given requirement (e.g., array, lists).

Screenshot and explain where you have created data structures such as lists, dataframes, dictionaries, arrays, etc… (explain why you created them and show screenshots) **– DOES NOT HAVE TO BE ALL YOUR EXAMPLES JUST SHOW A FEW DIFFERENT EXAMPLES**



* Dictionary: context

The context dictionary is used to pass data to the template rendering context.

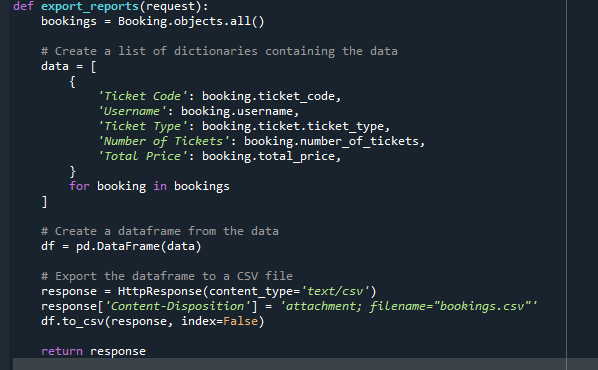
In various view functions such as book\_ticket, ticket\_details, available\_tickets, feedback\_list, and others. The context dictionary is populated with relevant data. The keys of the dictionary represent the variable names that will be accessible in the template, and the values are the corresponding data to be passed to the template.



* List: tickets

This list is used to store instances of the Ticket model retrieved from the database.

The tickets list is created in the book\_ticket view function. It retrieves all the Ticket objects using the Ticket.objects.all() query and assigns them to the tickets list.



* List of Dictionaries: Data

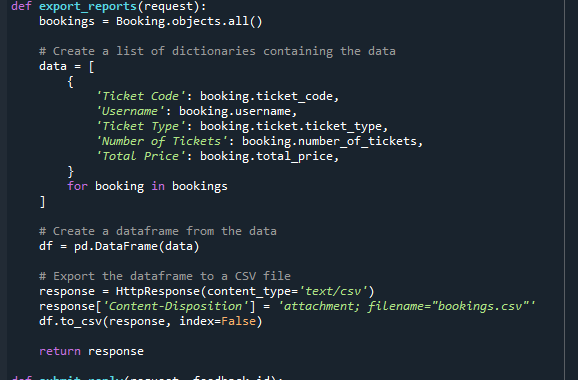
The variable data is a list that stores dictionaries. Each dictionary represents the data for a single booking. The keys in the dictionary correspond to the column names in the CSV file, and the values represent the booking information for each column. This data structure is used to organize the booking data before creating a dataframe.

* Dataframe: df

The variable df is a dataframe created using the Pandas library. The dataframe is constructed from the list of dictionaries (data) created in the previous step. Each dictionary in the list represents a row in the dataframe, and the keys in the dictionary correspond to the column names. This data structure allows for easy manipulation and analysis of the booking data.

2.3 Use sequential, branching and iteration (looping) to meet a given requirement.

Screenshot and explain where you have used iteration such as for loops. Show the conditional statements such as if else. (explain why you created them and show screenshots) **– DOES NOT HAVE TO BE ALL YOUR EXAMPLES JUST SHOW A FEW DIFFERENT EXAMPLES**



The for loop is used to iterate over the bookings queryset in the export\_reports function. Each booking is processed to create a dictionary representing its data, and these dictionaries are appended to the data list.

A screenshot of a computer program

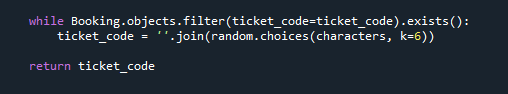
Description automatically generated with medium confidence

Several if-else statements are used in the book\_ticket function to handle different conditions. Here are a few examples:

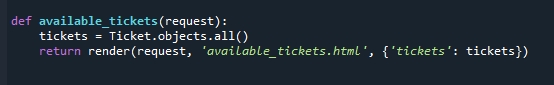
The if statement checks if the form is valid before processing the ticket booking request.

The if statement verifies if the requested number of tickets for each ticket type is available.

The if-else statement checks if a discount should be applied based on the number of tickets.



The generate\_ticket\_code function uses a while loop to generate a unique ticket code. It generates a random ticket code and checks if it already exists in the Booking model. If it does, it generates a new code until a unique one is found.



In the available\_tickets function, the tickets queryset is iterated over to display available tickets. Although not explicitly shown in the provided code snippet, if the number of tickets is large, pagination can be used to iterate over the queryset in smaller chunks for better performance.

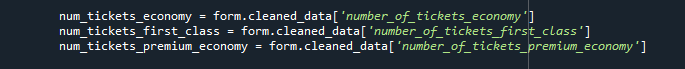
2.4 Apply input/output statements.

Output commands such as Print.

Input commands you MAY not have included this in your Python/SQL code so please explain why you haven’t. But you may have added some code to your backend of Anvil that allows the user to input. (explain why you created them and show screenshots) **– DOES NOT HAVE TO BE ALL YOUR EXAMPLES JUST SHOW A FEW DIFFERENT EXAMPLES**

In the given code, there are no input commands such as input because the code is designed to interact with the user through HTML forms in a web application. The user input is received through form submissions in the frontend, and the data is then processed in the backend using Django.

* In the book\_ticket view, the user selects the number of tickets for each ticket type in an HTML form. The form data is sent via a POST request, and the values are extracted using form.cleaned\_data['number\_of\_tickets\_economy'], form.cleaned\_data['number\_of\_tickets\_first\_class'], and form.cleaned\_data['number\_of\_tickets\_premium\_economy'].



* In the cancel\_ticket view, the user enters a ticket code in an HTML form to cancel a ticket. The ticket code is retrieved using request.POST.get('ticket\_code').



* In the submit\_feedback view, the user submits a feedback form with a name and feedback message. The values are retrieved using request.POST.get('name') and request.POST.get('feedback').



2.5 Use mathematical operators, relational operators, and Boolean operators in the implementation process.

Screenshot and explain where you have used operators such as multiply, divide, subtract, greater than, equal to, AND, OR, NOT. (explain why you created them and show screenshots) **– DOES NOT HAVE TO BE ALL YOUR EXAMPLES JUST SHOW A FEW DIFFERENT EXAMPLES**

* Mathematical operators:

In the book\_ticket function, the multiplication operator (\*) is used to calculate the total prices for each ticket type:

total\_price\_economy = ticket\_economy.price \* num\_tickets\_economy

total\_price\_first\_class = ticket\_first\_class.price \* num\_tickets\_first\_class

total\_price\_premium\_economy = ticket\_premium\_economy.price \* num\_tickets\_premium\_economy



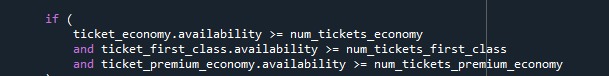
* Relational operators:

In the book\_ticket function, the greater than or equal to operator (>=) is used to check if there is sufficient ticket availability for each ticket type:

ticket\_economy.availability >= num\_tickets\_economy

ticket\_first\_class.availability >= num\_tickets\_first\_class

ticket\_premium\_economy.availability >= num\_tickets\_premium\_economy



* Boolean operators:

In the apply\_discount function, the greater than or equal to operator (>=) is used to determine if the number of tickets is greater than or equal to 5:

if num\_tickets >= 5:

In the same function, the logical NOT operator (not) is used to negate the condition:

if not num\_tickets >= 5: (equivalent to if num\_tickets < 5😊



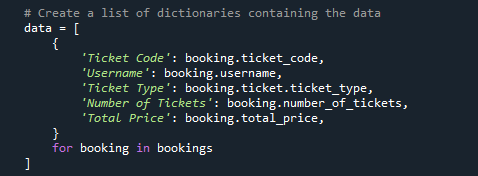
3. Be able to develop computer programs to industry standards.

3.1 Demonstrate good practice in coding.

Screenshot and explain where you have used good practice such as commenting, naming, layout, Do not repeat yourself (functions), data validation, etc… (explain why you created them and show screenshots) **– DOES NOT HAVE TO BE ALL YOUR EXAMPLES JUST SHOW A FEW DIFFERENT EXAMPLES**

* Commenting:

The code includes inline comments to explain the purpose and functionality of certain sections of code. For example, comments are used to describe the purpose of functions, variables, and code blocks.

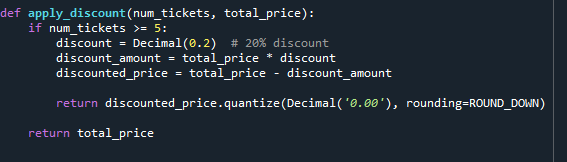


Inline comments like these help improve code readability and make it easier for other developers to understand the code's logic and functionality.

* Naming conventions:

The code uses meaningful and descriptive names for variables, functions, and classes, following the recommended naming conventions. This helps in understanding the purpose of each component without the need for extensive comments.

For example, variable names like num\_tickets\_economy, num\_tickets\_first\_class, and num\_tickets\_premium\_economy clearly indicate the number of tickets for different ticket types.

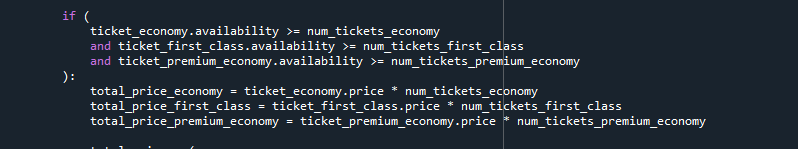


Good naming practices make the code more readable, maintainable, and self-explanatory.

* Layout and formatting:

The code follows proper indentation and formatting, making it easier to read and understand the structure and flow of the code.

Consistent indentation is applied throughout the code, enhancing code readability and maintaining a clean visual appearance.



Properly formatted code is easier to maintain, debug, and collaborate on with other developers.

* Don't Repeat Yourself (DRY):

The code demonstrates the principle of avoiding code duplication by utilizing functions and reusing code blocks.

For example, the code separates the functionality of applying discounts into a separate function named apply\_discount. This allows the code to be reused whenever a discount needs to be applied based on the number of tickets.



By encapsulating common functionality in functions, it promotes code reusability, maintainability, and reduces the risk of introducing bugs when making changes.

* Data validation:

The code includes data validation techniques to ensure the input data is valid and meets the required conditions.

For instance, the code checks the validity of the form inputs using the is\_valid() method of the form object:



Additionally, the code performs availability checks before making bookings to ensure there are enough tickets available.

4. Be able to demonstrate the use of functions in relation to software.

4.1 Explain the uses and benefits of functions.

Functions have several uses and benefits in programming:

* Reusability: Functions allow us to encapsulate a piece of code that performs a specific task and reuse it multiple times throughout the program. This promotes code reusability, as we can call the function whenever we need to perform that task, instead of rewriting the same code again and again.
* Modularity: Functions break down complex problems into smaller, manageable parts. By dividing the code into functions, we can focus on developing and testing individual components separately. This makes the code more organized, easier to understand, and maintain.
* Abstraction: Functions provide a level of abstraction by hiding the implementation details of a task behind a function name. We can use the function without knowing how it is implemented internally, as long as we understand its input parameters and return values. This simplifies the overall program structure and improves code readability.
* Code organization: Functions help in organizing code into logical blocks, each responsible for a specific task. This makes the codebase more structured, making it easier to navigate and locate specific functionality when needed.
* Debugging and maintenance: Functions isolate specific sections of code, making it easier to identify and fix bugs. Additionally, when changes or updates are required, modifying a function is often simpler and less error-prone than modifying the entire program.

4.2 Demonstrate the use of self-made functions to meet given requirements.

The apply\_discount function is an example of a self-made function. It calculates the discounted price based on the number of tickets and the total price. The function encapsulates the discount calculation logic and promotes code reuse by eliminating the need to repeat the same calculation in multiple places.

4.3 Determine and implement the use of pre-made functions in meeting given requirements.

In the above criteria you need to explain what functions are (User Defined, built in, etc…)

You then need to show and explain where you have used the above.

**DOES NOT HAVE TO BE ALL YOUR EXAMPLES JUST SHOW A FEW DIFFERENT EXAMPLES**

There is an example of using a pre-made function called redirect from the Django framework. This function is used to redirect the user to a different URL after a successful ticket booking. The redirect function simplifies the process of handling HTTP redirects without having to manually write the redirect logic.









5. Be able to test a computer program(s).

5.1 Use debug tools.

* Attached Document Flight Booking System Test Plan

5.2 Develop a test plan that determines expected results from test data.

* Attached Document Flight Booking System Test Plan

5.3 Follow the test plan in 5.2 to carry out a test.

* Attached Document Flight Booking System Test Plan

5.4 Compare actual results against expected results and make recommendations and changes as appropriate.

* Attached Document Flight Booking System Test Plan

This will be your test plan. You need to create a test plan (excel is recommended) the test plan should:

* Test Number
* Type of test – Please include Debugging, functional testing, regression testing, etc…
* How many times you are doing that test?
* What is the expected outcome – e.g. function runs with no errors or delays.
* What is the actual outcome?
* Do you need to make changes?
* Evidence – Screenshot.
* Attached Document Flight Booking System Test Plan

# Software Product Level 2:

1. Be able to implement software.

1.1 Select, declare and initialise variable and data structure types and sizes to meet given requirements.

**THIS IS COVERED IN 2.1 & 2.2 of Coding – No need to repeat yourself (please pop a line under this criteria saying please refer to 2.1 and 2.2 of coding.**

1.2 Implement control structures.

**If you have shown your if else statements/ loops/ etc… in coding then please state where you have covered it.**

1.3 Declare file structures.

Please show and explain the types of files you are working with.

These are all type file I used:

* .py (Python source code files): Python source code files (.py) contain the actual code

written in the Python programming language. These files are used to define

functions, classes, and modules that make up the program's logic.



* .html (HTML files): HTML files (.html) are used to define the structure and content of web pages. They contain markup elements that define the layout, text, images, links, and other elements displayed in a web browser.



* .css (CSS files): CSS files (.css) contain cascading style sheet rules that define the visual appearance and layout of HTML elements. They are used to apply styles such as colors, fonts, spacing, and positioning to enhance the presentation of web pages.



* .js (JavaScript files): JavaScript files (.js) contain code written in the JavaScript programming language. These files are used to add interactivity and behavior to web pages, allowing dynamic updates, event handling, and manipulation of HTML elements.



* .pyc (Compiled Python files): .pyc files are compiled bytecode files generated by Python when the source code is compiled. They are created to improve the execution speed of Python programs and are automatically generated when you run Python scripts.



* .sql (SQL files): SQL files (.sql) contain SQL (Structured Query Language) statements that are used to interact with databases. These files typically include commands to create tables, insert or retrieve data, and perform other database operations.



* .txt (Text files): Text files (.txt) are simple files that store plain text content. They can be used for various purposes, such as storing logs, configuration settings, or other textual data that doesn't require complex formatting.



* Text documents (e.g., .doc, .docx): Text documents are files created with word processing software like Microsoft Word. They allow formatting options, such as font styles, paragraphs, tables, and images, making them suitable for creating documents, reports, or documentation.



1.4 Use standard input/output commands.

**Covered in Outcome 2 of Coding – Please pop a line explaining where it is covered.**

1.5 Use operators and predefined functions.

**THIS IS COVERED IN CODING 2.5 & Outcome 4 – Please pop a line explaining where it is covered.**

1.6 Correctly use parameter passing mechanisms.

**THIS IS COVERED IN CODING Outcome – 4 - Please pop a line explaining where it is covered.**

2. Be able to refine a programme to improve quality.

2.1 Follow an agreed standard for naming, comments and code layout.

**THIS IS COVERED IN CODING Outcome 3 – Please pop a line explaining where it is covered.**

2.2 Implement data validation for inputs.

Show the techniques you have implemented for data validation. E.g. User inputting data – have you added drop downs to ensure that the data they input is correct and not random.

How do you check if the data inputted is valid? E.g. If a user tried to log in but already has an account?

* Form Validation: The code utilizes Django's form validation mechanism by creating instances of forms such as TicketBookingForm, FeedbackSubmissionForm, and FeedbackForm. These forms define fields and validation rules, ensuring that the data submitted through them is validated according to the specified criteria.
* Input Validation: The code checks if the form submitted through the book\_ticket view is valid using the is\_valid() method. This validates the user input based on the form's defined validation rules, such as checking for the availability of tickets and ensuring the number of tickets requested is within the available range.
* Database Queries: The code utilizes database queries to check if certain conditions are met before proceeding with specific actions. For example, in the book\_ticket view, it checks the availability of tickets by querying the Ticket model and comparing it with the number of tickets requested by the user. If the availability is sufficient, the booking is processed; otherwise, an error message is displayed.
* Error Messages: The code uses the messages module from Django to display error messages to the user in case of validation failures. These messages inform the user about the specific validation issue, such as insufficient ticket availability or tickets not being available.
* Conditional Checks: The code includes conditional checks to ensure that certain conditions are met before executing certain actions. For example, in the apply\_discount function, a check is performed to determine if the number of tickets is greater than or equal to 5. Based on this condition, a discount is applied to the total price.
* Unique Ticket Code Generation: The generate\_ticket\_code function generates a unique ticket code by using random characters. It ensures that the generated code does not already exist in the Booking model by performing a database query and checking for existing tickets with the same code.

2.3 Implement error handling and reporting.

Have you created any error handling techniques in your code? Show and explain them. E.g. within if else statements – having every possible combination to stop the user from receiving nothing or an error.

INSIDE MY PROJECT

2.4 Create documentation to assist the users of a computer programme.

Create a user guide explaining exactly how a user would navigate the app? E.g. how to get to the page, what buttons to click on, types of information to type in, etc…

INSIDE OF MY PROJECT

3. Be able to test the operation of a programme.

3.1 Use available debugging tools.

INSIDE MY PROJECT INTO MY TEST PLAN

3.2 Determine expected test results from given test data.

INSIDE MY PROJECT INTO MY TEST PLAN

3.3 Compare actual test results against expected results to identify discrepancies.

**THIS IS COVERED IN CODING Outcome 5 – Please pop a line explaining where it is covered.**

Database Applications:

1. Understand database structures and implementation methods.

1.1 Explain the database applications in use in business.

Explain what database applications are about, the one you are using and why? E.g. MS access, SQL Server, MySQL, etc…

Database applications are software systems designed to manage and manipulate structured data. They provide a platform for creating, storing, retrieving, and manipulating data efficiently and securely. Database applications are widely used in business for various purposes, such as inventory management, customer relationship management (CRM), human resources, financial systems, and more.

One of the database applications I have used is MySQL. MySQL is an open-source relational database management system (RDBMS) that is popular among developers and businesses alike. It offers a robust and scalable solution for managing structured data. Here are a few reasons why MySQL is commonly chosen:

1. Reliability and Performance: MySQL is known for its stability, reliability, and excellent performance. It can handle large volumes of data and concurrent user connections efficiently.
2. Flexibility: MySQL supports various data types and provides a wide range of features for data manipulation and management. It offers support for transactions, stored procedures, triggers, and views, allowing for complex data operations and logic.
3. Scalability: MySQL is scalable and can accommodate growing data needs. It supports replication, which enables data to be distributed across multiple servers, improving performance and providing fault tolerance.
4. Wide Community Support: MySQL has a large and active community of developers, administrators, and users. This means there are abundant resources, documentation, and community support available, making it easier to troubleshoot issues and find solutions.
5. Cost-Effective: MySQL being an open-source RDBMS, it is free to use and has a lower total cost of ownership compared to commercial database systems. This makes it an attractive choice for small to medium-sized businesses or projects with budget constraints.

However, it's worth noting that the choice of a database application depends on specific business requirements, scalability needs, performance expectations, budget constraints, and the expertise available in the development team. Other popular database applications used in business include SQL Server, Oracle, PostgreSQL, and MongoDB. Each of these applications has its own strengths and may be more suitable for certain use cases or business scenarios.

1.2 Explain the structures used in database implementation.

**JUST WRITING HERE – no need for evidence.**

Discussing: Relational, non relational, distributed, flat file, hierarchical databases.

Discussing what relationships, entities, keys, attributes, indexes, etc… are.

The database implementation follows a relational database structure. Relational databases organize data into tables with predefined relationships between them. Let's discuss the key elements of the relational database structure and the code you shared:

* Relationships: Relational databases use relationships to associate data between tables. In the code, we see a few relationships defined:

The Booking model has a foreign key relationship (ticket) with the Ticket model, indicating that each booking is associated with a specific ticket.

The Feedback model has a one-to-one relationship (reply) with the Reply model, indicating that each feedback can have a single reply.

* Entities: Entities are represented as tables in a relational database. In the code, each model (Ticket, Booking, Feedback, and Reply) represents an entity, and each instance of the model represents a record in the respective table.
* Keys: Keys are used to uniquely identify records in a table. In the code, we see several keys being used:

The Ticket model has a primary key (id) that is automatically generated for each record.

The Booking model has a primary key (id) as well, and it also has a unique ticket code (ticket\_code) that acts as a key for identifying bookings.

The Reply model doesn't have a key defined explicitly, but it can be associated with a feedback using a foreign key relationship.

* Attributes: Attributes represent the characteristics or properties of an entity. In the code, we see various attributes defined for each model, such as ticket\_type, price, availability for Ticket; username, number\_of\_tickets, total\_price for Booking; name, feedback for Feedback; and content for Reply. These attributes determine the fields/columns in the corresponding database tables.
* Indexes: Indexes are used to improve the performance of database queries by allowing faster data retrieval. In the code, we don't see explicit indexes defined, but Django automatically creates indexes for foreign key fields and primary key fields. Indexes can enhance query performance when filtering or joining data.

1.3 Outline the syntax used to create and query database tables.

Discussing things such as Create, Merge, Delete, Sorting, Select, Join, Unions, etc…

Explain what they are

**JUST WRITING HERE – no need for evidence.**

Here are some common SQL statements for creating and querying tables:

Create Table: The CREATE TABLE statement is used to create a new table in the database. It specifies the table name, column names, data types, and any constraints. Here's an example:

CREATE TABLE TableName (  
 column1 datatype1 constraint1,  
 column2 datatype2 constraint2,  
 ...  
);

Insert Into: The INSERT INTO statement is used to insert new rows of data into a table. It specifies the table name and the values to be inserted into each column. Here's an example:

INSERT INTO TableName (column1, column2, ...)  
VALUES (value1, value2, ...);

Update: The UPDATE statement is used to modify existing data in a table. It specifies the table name, the column(s) to be updated, and the new values. Additionally, a WHERE clause can be used to specify which rows to update. Here's an example:

UPDATE TableName  
SET column1 = value1, column2 = value2, ...  
WHERE condition;

Delete: The DELETE FROM statement is used to remove rows from a table. It specifies the table name and a WHERE clause to determine which rows to delete. Here's an example:

DELETE FROM TableName  
WHERE condition;

Select: The SELECT statement is used to retrieve data from one or more tables. It specifies the columns to be selected and the table(s) from which to retrieve data. Additionally, it can include filtering conditions, sorting, grouping, and joining of tables. Here's a basic example:

SELECT column1, column2, ...  
FROM TableName  
WHERE condition;

Join: The JOIN clause is used to combine rows from two or more tables based on a related column between them. There are different types of joins such as INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL JOIN, each specifying different behaviour for combining the rows. Here's an example of an inner join:

SELECT column1, column2, ...  
FROM Table1  
INNER JOIN Table2 ON Table1.column = Table2.column;

Union: The UNION operator is used to combine the result sets of two or more SELECT statements into a single result set. It removes duplicate rows by default. Here's an example:

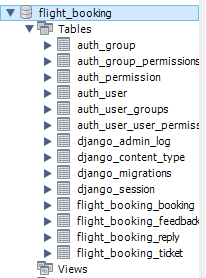
SELECT column1, column2, ...  
FROM Table1  
UNION  
SELECT column1, column2, ...  
FROM Table2;

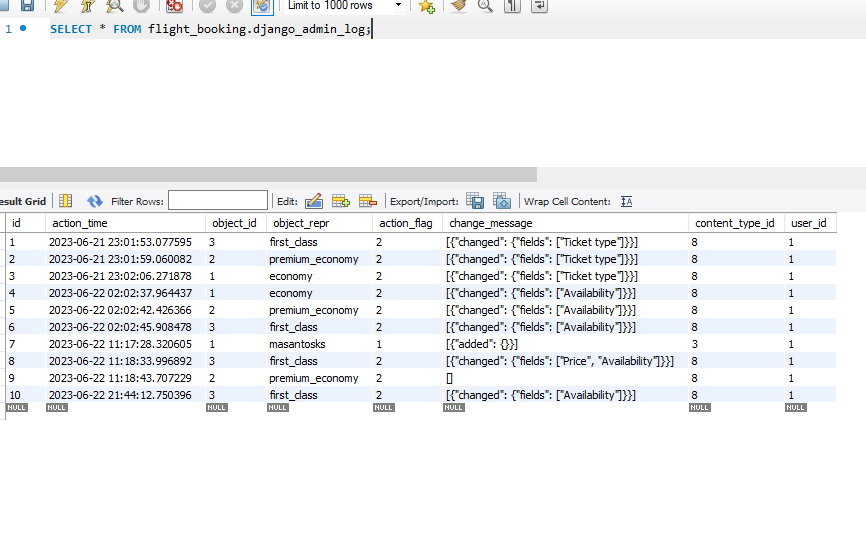
2. Be able to use a database management system to create a database to meet a design brief.

2.1 Select appropriate database application.

MySQL

2.2 Create database tables to implement a database schema.



2.3 Enter, edit and organise data.

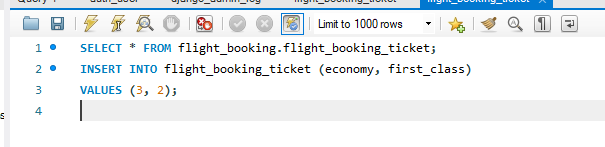
2.4 Test that database meets requirements.

Outcome 2 –

Required you to Create, Populate, delete and Join tables.

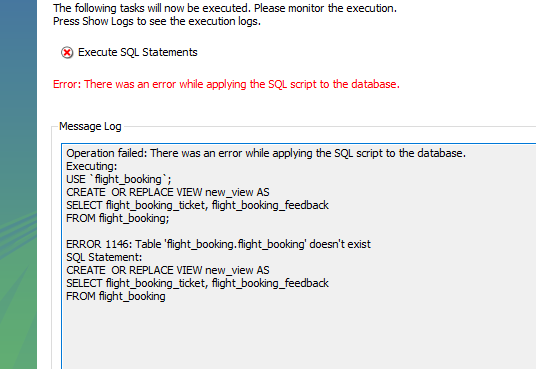
Create relationships, show the properties/datatypes of the data.

Methods to validate the data?

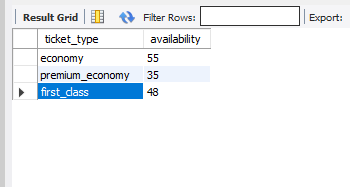


Any error messages you’ve received?





Doing views, calculating tables, etc…



Making sure the database is correct and meets the requirements for Anvil (so data moves smoothly from Anvil to SQL).

**SHOW EVIDENCE AND EXPLAIN.**

3. Be able to interrogate database systems to meet requirements.

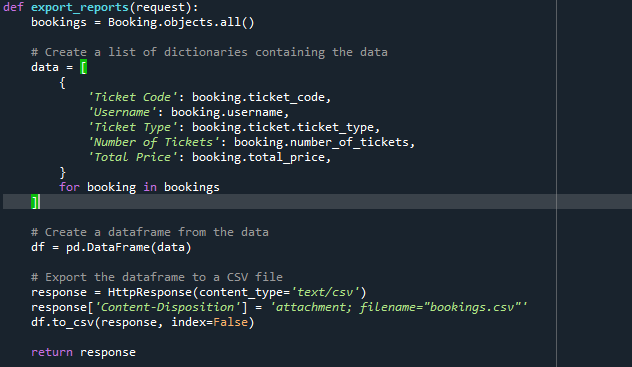
3.1 Create and edit database queries to display, amend or calculate selected data.

**THIS IS COVERED IN Database Applications Outcome 2 – Please pop a line explaining where it is covered.**

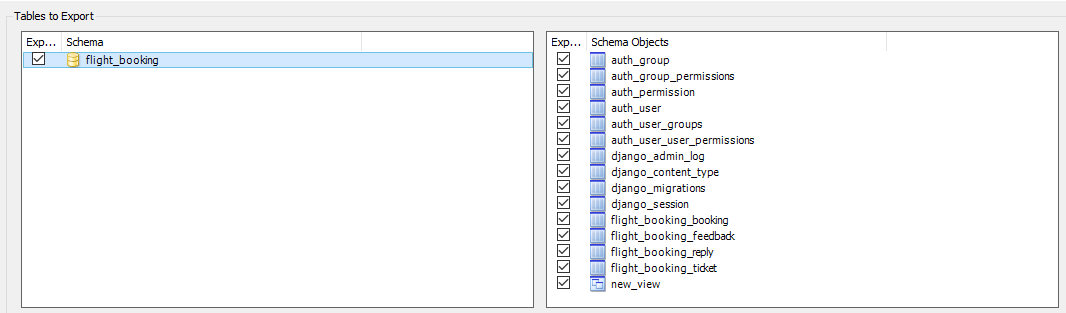
3.2 Plan and generate management information reports.

I have two ways to export reports from my data base.

The first way is from a function called ‘def export\_reports’



The second is from the export into my database:

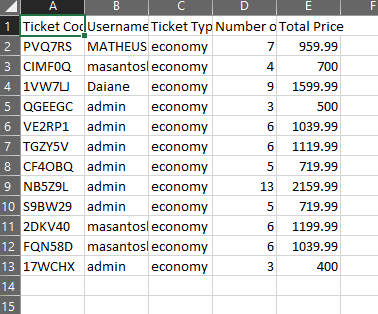


3.3 Interpret management information reports.

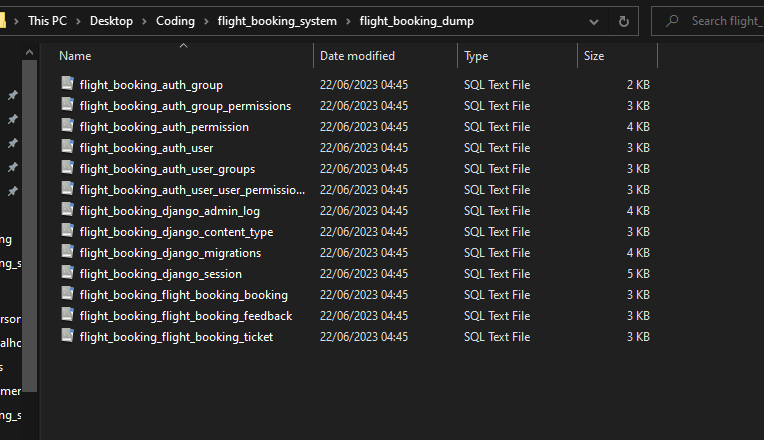
Show where you have gathered information/summary reports about the data. E.g. From Anvil you can gather – how many people have booked, the type, how much you have made in total.

**SHOW EVIDENCE AND EXPLAIN.**

After I generated the report inside the program I just need to open the csv file that will give me the contents of my database



After I downloaded the database report from my db I just need open the file



4. Be able to develop a user interface for a database solution.

4.1 Prepare forms for data entry.

INSIDE MY TEST PLAN DOCUMENT

4.2 Set up menus and controls to manipulate data.

INSIDE MY TEST PLAN DOCUMENT

4.3 Produce reports to meet requirements.

INSIDE MY TEST PLAN DOCUMENT

4.4 Protect data from unauthorised access.

INSIDE MY TEST PLAN DOCUMENT

4.5 Test features of user interface and respond appropriately to errors.

INSIDE MY TEST PLAN DOCUMENT

4.6 Gather and respond to feedback from users.

INSIDE MY TEST PLAN DOCUMENT

This is where you will show evidence of Anvil – Take a few screenshots showing how you have met the above questions. We will also ask you to demonstrate the App working so this will help you boost up the evidence for this criteria.

INSIDE MY TEST PLAN DOCUMENT