

URL: <http://classwork.engr.oregonstate.edu:2900/index.html>

Project Step 5 Draft Version

Project Name: Fly Anywhere Airlines Flight Database

Team: JKM (Jack/Katherine/Muhammad)

Part 4 Draft Feedback: Received from Tristen Beaudoin (TB), Ryo Kumakawa (RK), Alyshia Benson (AB), Meredith Baker (MB)

Feedback from TA reviewer: Great work, see rubric for feedback and DM me on teams with any questions or concerns!

- No action needed to change anything based on TA feedback.

Specific Feedback:

TB - This is a minor suggestion, but I'd recommend adding a button so the user can navigate back to the homepage easily. That was the only aspect of the navigation that seemed to be missing.

Fix: We have updated this to apply a homepage button on every page.

RK - One thing I have noticed is that when deleting an item, we can see the confirmation form on the page itself, so once we click the delete button then another confirmation alert is shown. So this is kind of confirming the same thing twice, which I think not necessary

Fix: This has been fixed so only one confirm message appears when a user wants to delete an item.

AB - You have stated you are not blocked just debugging so I will just give some feedback. I love the homepage you created it is very easy to read and navigate. The design is simple and very user friendly. I really like that you have the quick stats on the homepage as well. Although the other pages are not styled they are still very nice and easy to read. I like how the set up of each page stays consistent to the next page, such as the edit buttons being in the same place on every page. This is very nice and helps the program become more intuitive. I think you have a really user friendly design.

Fix: No changes suggested

MB - You mentioned already that you are working on the CSS file, so I would say that once you implement a similar style to your home page, you're all set! I like your web app so far and have found it very intuitive and easy to use. Keep up the great work :)

Fix: No changes suggested

General Feedback:

Do the implemented CRUD steps function as the team expects? If yes, explicitly state the working CRUD that has been implemented. If not, what CRUD isn't working as stated, and what happens upon attempting to perform those CRUD operation(s)?

MB - All of the pages appear to have working CRUD functionality. I decided to test the Routes table, as it was mentioned by the team to be fully reviewed at this step. The CRUD functionality for Routes is working perfectly - I was able to CREATE, READ, UPDATE, and DELETE an entry in this table with no issues.

TB - All of the CRUD functions work as expected. I was able to successfully add (create), view (read), update, and delete entries on all 8 tables (including the intersection tables).

There was one behavior that I came across that I'm not sure is intended or not. I was able to delete any aircrafts that weren't already associated with a flight. However, if I wanted to delete an aircraft that was associated with a flight record, then I got an error. I assume this was an intentional ON DELETE RESTRICT, but thought it'd be worth mentioning for good measure.

AB - The working CRUD operations have been implemented on the following pages: Routes, Flights, Aircrafts, Seats, Passengers & Bookings, and Employees & Assignments.

RK - Yes, for Routes page, CRUD operations are fully functioning. Looks other tables are also functioning with CRUD operations.

Would a user easily be able to use the UI to complete the step? If yes, describe what makes the UI easy to navigate. If not, or you have suggestions for how the UI can be improved, please elaborate.

MB - In my opinion, the UI is nicely formatted and easy to understand. I appreciate the notifications for each successful CRUD operation. Buttons, tables, and navigation menus are all clearly labeled and work as intended.

TB - Yes, I found the UI very easy to navigate. Almost all of the tables/pages have a clear button in the primary navigation that stays persistent across all pages, and the CRUD function-related buttons are clearly labeled (add, edit, delete, etc.).

AB - A user would easily be able to use the UI. The page names are self-explanatory such as 'Flights' for the page about flights. The buttons are clearly label such as 'Add New Flight' to add a flight to the schedule and each line has an edit button and a delete button. All tables are also displayed so this makes it very easy to navigate.

Navigating within "Passengers & Bookings" and "Employees & Assignments" was slightly less intuitive because there was a secondary navigation to help navigate to the intersection tables (Bookings and Assignments). However, it was still relatively easy to understand the navigation, and I think the stylization of the pages will help.

RK - Yes, I was able to follow the flows easily for CRUD operations. No blocking can be seen

Part 3 Final Feedback:No feedback received for final version of Part 3, so no updates.

Part 3 Draft Feedback: Received from Jonathan Guzman (JG), Zyan Shull-Bain (ZS), Jeremy Shinn (JS), and Matthew Smith (MS)

Feedback from TA reviewer: Great work, the UI looks great, DM me on teams with any questions or concerns.

- No action needed to change anything based on TA feedback.

Does the UI show where it will utilize a SELECT for every table in the schema?

- MS: Yes, every table is represented in the UI with some data (routes, flights, aircraft, seats, passengers, bookings, employees, and assignments).
- JG: Yes, all the tables, routes, flights, aircraft, seats, passengers, bookings, employees, are represented in the UI based on the provided in the outline with some data
- ZS: Yes, every table in the schema has a UI page created for it with some data populating it. Some examples are routes, seats, bookings, passengers, aircraft, employees, and assignments. Additionally, the attributes for each are also correct.
- JS: Yes, there is a SELECT for Flights, Routes, Seats, Aircraft, Assignments, Employees, Passengers, and Bookings.

Does at least one form utilize a search/filter/dropdown with a dynamically populated list of properties? If yes, which form(s) have which features incorporated? For which attribute(s)? If not, where are a couple places this could be implemented?

- MS: Yes, there is a search and dropdown on the Employees & Assignments page as well as the Passengers & Bookings page.
- JG: Yes there a drop down in the employees tab
- ZS: Yes, there is a search and dropdown for employees & assignments which highlights the roles for the employee section and the flights for the assignments section. Additionally, there is a search option for the passengers and bookings page.
- JS: Yes, there is a dropdown in Flights for flightStatus, in Aircraft for aircraftStatus, in Seats for seatClass, and in Employees for jobRole.

Does the UI implement an INSERT for every table in the schema?

- MS: Yes, I see an 'Add New' button for every table in the schema (routes, flights, aircraft, seats, passengers, bookings, employees, and assignments).
- JG: Each tab/table, has a button to add a new set of data
- ZS: Yes, there was an add new button on each page that allowed for a user to insert new data into each table. When selecting add new, a popup menu appears with fields to fill in data in regards to the attributes of the table to which the insert is being done. Great job.
- JS: Yes, there are complete INSERTs for Flights, Aircraft, Routes, Seats, Passengers, Employees, Bookings, and Assignments. They seem to cover all the tables and each attribute for each table. I do not have any suggestions here.

Does each INSERT also add the corresponding FK attributes, including at least one M:M relationship?

- MS: Yes, the Bookings page has the ability to insert the Passenger, Seat, and Flight FKs. The Assignments page also has the ability to add the Employee and Flight FKs.
- JG: Yes, the bookings tab can insert a passenger with seat, and flight as FKs, while the assignments tab can add employee asFKs.
- ZS: Yes, the insert functions that require FK's allow the user to add them when inserting an item. One such example is when adding a booking, you must identify the passenger FK and the flight FK. For the assignment table, you must add the employee and flight FK's when inserting an element.
- JS: Yes, there is a separate INSERT for Bookings, which facilitates a M:M relationship between Flights and Passengers. The group has implemented this by using an Add New Booking form on their site. There is also an INSERT for Assignments, which facilitates a M:M relationship between Flights and Employees.

Is there at least one DELETE, and does at least one DELETE remove things from a M:M relationship?

- MS: Yes, it appears that every page/table has the ability to delete data.
- JG: Each tab has the option to delete existing data and according to the SQL it will cascade delete the FK relations with the exception of the flight table
- ZS: Yes, each table allows for the removal/deletion of a specific row. Additionally, the corresponding DML file also has cascade delete for the FK relationships (except flights) so there won't be any orphaned relationships. Great job again.
- JS: Yes, there are DELETES for Flights, Aircraft, Routes, Seats, Passengers, Employees, Bookings, and Assignments. That is one DELETE for each table. They all seem to be implemented as they should be.

Is there at least one UPDATE for any one entity, with fields for the corresponding attributes for that entity?

- MS: The UI has the ability to update every single table in the schema utilizing an Edit button on each row of data.
- JG: There's an edit button to be able to add and edit existing data in every single one of the tables/tabs

- ZS: Yes, there are edit buttons on each tab in the UI so that each row can be edited in each table. When selecting edit, a popup appears with the correct information for the attributes and allows the user to change them and save the changes.
- JS: Yes, there are UPDATEs for Flights, Aircraft, Routes, Seats, Passengers, Employees, Bookings, and Assignments, and each attribute that is not a primary key is able to be updated.

Is at least one relationship NULLable?

- MS: In the schema it does appear that there are NULLable relationships such as not every route needs a flight and the data on the Route page shows this.
- JG: The schema allows NULLable relationships because not every route requires a flight, which can be in the route tab.
- ZS: Yes, there is at least one nullable relationship. Such are shown in the schema and database allowing for nullable relationships for routes having flights since not every route will have a flight at all times.
- JS: Yes, for Flights, the relationships with Routes and Aircraft are NULLable. This makes sense as we still want a record of Flights if certain routes or aircraft are discontinued. For Bookings, the relationship with Passengers is NULLable. This also makes sense, as a record that a seat was filled may be useful even if a passenger was removed from the database.

Do you have any other suggestions for the team to help with their HTML UI? For example, using AS aliases to replace obscure column names such as fname with First Name.

- MS: The UI is really great and good looking. My only recommendation would be to watch the fonts colors on the Passengers & Bookings page and the Employees & Assignments page. I missed where you could toggle between the two tables within the same page due to the font color blending with the background.
- JG: When I was first using this UI whenever I scrolled it would not go down, i had to zoom out to be able to delete/add/edit stuff to the forms. Maybe troubleshooting with more devices? I'm a mac using chrome.
- ZS: Overall, the design is sleek and clean. However, I had trouble with your dropdowns/popups because the screen was not able to scroll down. It should be a pretty straightforward change but would make maneuvering through the website a lot easier. Other than that, all looks good.
- JS: I think their UI is coming along very nicely. Aliases seem appropriate and easy to understand. The UI is easy to navigate and has a nice feel to it. Really great job so far!

Fixes Based on Feedback from Part 3:

1. Font colors in Passengers & Bookings and Employees & Assignment pages tabs should be updated to more contrasting color to display that you can switch between the two.
 - a. This was fixed to allow better contrast and visibility
2. Scroll on pages is not properly working when a form appears (zooming in and out fixes)
 - a. This was fixed to allow scrolling without zooming in
3. From Project 3 Rubric: JOINS should be used to make FKs user friendly.
 - a. To Do: Update the DML to have intersection tables use friendly joins in the syntax, vs. just selecting the table.
 - i. Done: SQL has been updated to concatenate friendly values from tables being intersected to appear in the select statements.
4. Note: No changes to DDL SQL, ERD, Outline, or schema were requested by the feedback providers as part of this round of updates. Please see feedback for Part 2 and Part 1 for changes to those sections of the project.

Part 2 Feedback: Feedback received from: Shane McKay (SM), Michelle Ma (MA), John Weathers (JW), and Zachary Dicks (ZD)

Does the schema present a physical model that follows the database outline and the ER logical diagram exactly?

- (SM) Yes, the schema shows 6 entities, which is the same as what's in the outline. To describe the many-to-many relationships, there are 2 intersection tables
- (MA) Yes, there are the 6 entities and 2 intersection tables shown in the schema. The relationships correspond to the outline, and there are no discrepancies between what is listed and what is depicted in the diagram.
- (JW) Yes, everything matches up quite well between the three. All of the tables and attributes seem to match up. The relationships all make sense and are consistently labeled between the outline, ERD, and schema. There are only a few minor variations to mention:
 - FlightStatus in flights listed as varchar in ERD but enum in schema
 - There is at least one optional relationship (bookings to passengers, where the passengerId is optional in bookings), and that isn't properly reflected in the schema
 - Separately to the above, there are a couple other foreign keys in the schema with red diamond icons that are not filled in (located in flights). I think this is supposed to indicate that they're optional, but the relationship lines indicate everything is mandatory, so that should be cleared up.
- (ZD) Yes, the schema diagram presents a physical model that follows the database outline and entity-relationship diagram. The outline identifies the "Flights", "Aircraft", "Routes", "Seats", "Passengers", "Employees", "Bookings", and "Assignments" entity and intersection tables along with their respective attributes. However, there are a couple of improvements that could be made:
 - First, the entity-relationship diagram is almost indistinguishable from the schema diagram. Since the entity-relationship diagram is intended to display the entities and their relationships, I suggest removing all non-key attributes from the entity-relationship diagram. In addition, organizing the tables in the same position in both diagrams will help the reader easily cross-reference the two tables without getting confused or lost.
 - Next, it seems like the "Flights" entity is missing the M:N relationship between flights and employees via the "Assignments" table in the overview section of the report.
 - Finally, there is an optional M:1 relationship between bookings and passengers in the entity relationship diagram that is not shown in the schema diagram.

Is there consistency in a) naming between overview, outline, ER and schema entity/attributes b) entities plural, attributes singular c) use of capitalization for naming?

- (SM) One difference is that the outline and the ERD has every entity capitalized, while the schema has no capitalization. distanceKilometers and durationSeconds is also plural, while the rest of the attributes are not. I think it would be a bit weird to name them distanceKilometer and durationSecond, but that would keep the plurality the same.
- (MA) Yes, the naming system is pluralized for all entities (except Aircraft, which might be a stylistic choice?), camelCase used for attributes, all ids follow the same naming system (entityId), and intersection tables' foreign keys contain the same naming system for the given IDs.
- (JW) For the most part, yes. The entities seem to all be capitalized (see below for minor note on this). The attributes are all camel case. Everything is consistent, aside from a couple minor things, between all three. Minor issues I've spotted:
 - The entities in the schema are not capitalized, so that needs to be fixed
 - PassengerId in Passengers, in the outline, is listed as passengerID
- (ZD) There is consistency between the naming conventions used in the overview, outline, entity-relationship diagram, and schema diagram. All entities are plural and attributes are singular. On the other hand, all entities within the outline and entity-relationship diagram are capitalized versus the schema diagram which is lowercase. The entities within the schema diagram should be capitalized.

Is the schema easy to read (e.g. diagram is clear and readable with relationship lines not crossed)?

- (SM) Yes, the schema is easy to read. All the lines are split up, do not cross each other, and are close to each other. One recommendation I would give would be to get rid of the background grid,

which would make it blend in better with the proposal (there is an option within Workbench to do this)

- (MA) Yes, all the entities are laid out clearly, where no relationship lines are crossed. I like how the flights and bookings routes are aligned in the center because it's clear to see how all the other entities relate to the flights and bookings.
- (JW) Yes, the schema is easy to read. None of the relationship lines are crossing. The entities are placed in a way that makes it easy to scan between the relationships. They are close enough where you can easily look back and forth between a relationship, but not too close to be cramped and messy looking.
- (ZD) Yes, the schema diagram is easy to read as there are no overlapping relationship lines and each entity table has enough spacing to see each attribute.

Are intersection tables properly formed (e.g. two FKs and facilitate a M:N relationship)?

- (SM) Yes, the intersection tables both have at least two FKs and they facilitate a M:N relationship. The first one is from flights to passengers/seats. This table is made of three foreign keys, flightId, seatId, and passengerId. The second intersection table, assignments, has two FKs, flightId and employeeId and links the employees with the flights.
- (MA) Yes, there are two intersection tables that each contain two FKs, alongside other attributes that are important to the relationship (such as passengers' IDs). This helps create a valid M:N relationship between Flights, Passengers, and Seats (for intersection table Bookings) and Employees + Flights (for Assignments). I especially like the name Assignments, since it makes the relationship easy to understand upon first glance.
- (JW) Yes, I count three intersection tables, all of which seem to be properly formed. They each have foreign keys from appropriate tables which facilitate a many to many relationship – the Flights table has routeId and aircraftId, the Assignments table has flightId and employeeId, and the bookings table has flightId, seatId, and passengerId.
- (ZD) Yes, the “Bookings” and “Assignments” intersection tables are properly formatted with at least two foreign keys. The “Bookings” intersection table facilitates the M:N relationship between “Flights” and “Seats”, and “Flights” and “Passengers”. Further, the “Assignments” intersection table facilitates the M:N relationship between “Flights” and “Employees”.

Does the sample data suggest any non-normalized issues, e.g. partial dependencies or transitive dependencies?

- (SM) No, I do not see any normalization issues. None of the non-prime attributes are dependent on other non-prime attributes, and every entity is split into its own table. One area I thought was incorrect was that the capacity of the aircraft may rely on the model of the aircraft. However, the capacity may change depending on how the airline sets up the aircraft. Multiple models can have different seat configurations.
- (MA) No, it seems that all non-prime attributes are only dependent on the PK. This aligns with 3NF.
- (JW) By and large, the data suggests no issues. The only thing I can think of that could potentially be changed is the Aircraft entity seems to have potential for duplicate data with model and capacity. I don't think it's a big issue since there are only a couple attributes, but I think the intention of the table is individual planes that the airline has as part of the fleet. The model info seems like it could be a separate thing. However, given that there are already a good number of entities, I understand not wanting to add an additional one here. Otherwise, everything seems normalized to a satisfactory extent.
- (ZD) No, the same data is normalized and doesn't show any partial or transitive dependencies.

Is the SQL file syntactically correct? This can be easily verified by using phpMyAdmin and your CS 340 database (do not forget to take backup of your own database before you do this!)

- (SM) Yes, the SQL file is syntactically correct. All 8 tables were imported with the tables have 3-4 rows each.
- (MA) Yes, the SQL file imported all 8 tables as seen in the Schema diagram. The foreign keys were also imported correctly. Data for Flights, Aircraft, Routes, Seats, Passengers, Employees, Bookings, and Assignments were added successfully.
- (JW) Yes, the SQL file is syntactically correct. I sourced the file without any issues. I was able to view the sample data in each table.

- (ZD) I was successfully able to import the “CS340 Group 118 Project Step 2 DDL.sql” file and all applicable data with the enable foreign key checks option turned off in phpMyAdmin.

In the SQL, are the data types appropriate considering the description of the attribute in the database outline?

- (SM) Yes the data types are appropriate, and are the same, when compared to the description of the attribute in the database outline. Every attribute that would be a number has an INT, dates are created as DATETIME, attributes that have select options are created correctly with ENUM.
- (MA) Yes, the data types make sense for each attribute. The IDs are integers (if PK, has auto_increment), phone numbers are VARCHAR to allow dashes or + for country codes, and aircraftStatus and seatClass is ENUM since there are only three / four possible options given.
- (JW) Yes, the data types are appropriate considering the descriptions of the attributes in the outline. All the attributes have data types that match the outline, including the arguments passed to the data types (where applicable). I don't see any variation from the outline in this regard.
- (ZD) Yes, the datatypes used for the attributes are appropriate to their application. However, I do suggest changing the varchar() attribute values to be more in alignment with their use case. For example, “emailAddress” can be 254 or 320 characters based on IETF / RFC standards, and “phoneNumber” can be 10, 12, or 14 characters based on the stored syntax (“1234567890”, “123-456-7890”, or “(123) 456-7890”).

In the SQL, are the primary and foreign keys correctly defined when compared to the Schema? Are appropriate CASCADE operations declared?

- (SM) The foreign keys and primary keys are defined correctly when comparing the Schema. However, there are no CASCADE operations declared. This could result in intersect tables addressing to blank rows. Some of the CASCADE operations I would suggest would be for bookings and assignments to have a CASCADE DELETE, as they will be deleted if the flight is deleted. I would also add a CASCADE DELETE to seats, as if the aircraft is deleted then the seat would be deleted, along with the flight and bookings.
- (MA) Yes, the PKs and FKs are correctly defined in the SQL. Foreign keys are hardcoded into the create table query, which could be replaced with subqueries in the insert queries. There are no cascade operations defined; I think cascade operations should be added to the entities, especially since you have so many foreign keys.
- (JW) Yes, primary and foreign keys are correctly defined when compared to the schema. This is notwithstanding my prior note about the optional relationships not being 100% correctly documented in the schema diagramming. But every primary key in the schema is correctly defined in the SQL, as is every foreign key.
 - No CASCADE operations are declared. However, I think with an airline, this may actually be the appropriate choice in many cases. That said, the places I would consider declaring CASCADE operations might be the following:
 - A route and/or aircraft record is deleted, should the routeld and/or aircraftld be set to null in flights? Would be easier to spot that a new route or aircraft is needed for a particular flight, rather than old data being referenced there.
 - An aircraft is deleted, should the seats be deleted?
 - For Bookings and Assignments, I would imagine the default setting of no changes could make the most sense because you'd still want fully intact records of those in the intersection table.
- (ZD) Yes, all primary and foreign keys are correctly defined when compared to the schema diagram, but there aren't any CASCADE operations declared in the SQL file. I suggest adding CASCADE operations for any table whose primary key is used as a foreign key in another table.

In the SQL, are relationship tables present when compared to the ERD/Schema?

- (SM) Yes, the tables assignments and books are present in the SQL.
- (MA) Yes, relationship tables Bookings and Assignments are present. All foreign keys are added in the create query.
- (JW) Yes, all the relationships are laid on in the SQL via appropriately defined foreign key constraints. Flights has a reference to Routes and Aircraft, Seats to Aircraft, Bookings to Flights, Seats, and Passengers, Assignments to Flights and Employees.

- (ZD) All relationship tables are present in both the SQL file and the entity-relationship/schema diagram.

In the SQL, is all sample data shown in the PDF INSERTED?

- (SM) Yes, all the data is inserted. All the tables have 3-4 elements inserted.
- (MA) Yes, all sample data that is in the PDF is also in the SQL file as insert queries.
- (JW) All tables shown in the PDF have sample data inserted. There are a couple minor discrepancies:
 - passengerId in Bookings has incorrect info
 - There is an extra row in Bookings that is not in the sample data
- (ZD) Yes, the sample data in the PDF is inserted into the applicable entity tables in the SQL file. The only suggestion is to use subqueries for foreign keys in the SQL file rather than hardcoding them.

Is the SQL well-structured and commented (e.g. hand authored) or not (e.g. exported from MySQL)?

- (SM) Yes. The file is commented nicely, which breaks up the sections and labels what each query does. The file does not appear to be generated in anyway.
- (MA) Yes, the SQL is well-structured, hand-authored, and contains leading comments that specify what the following query is performing. For example, before an insert query, the author has comments stating that the query below is inserting the sample data for entity "Flights".
- (JW) Yes, the SQL is well-structured and commented. The file is well organized with the tables all being dropped in one section. The create statements are separated into a large section together, and each create statement is commented. The insert statements are all in a large group and well commented.
 - The only issue, per the assignment instructions, is the group members' names and group number should have been included in the SQL file but they are missing.
- (ZD) Block comments are present in the SQL file, which makes each code section easily identifiable and understandable.

Fixes Based on Feedback from Part 2

1. Update FlightStatus to be ENUM in ERD
 - a. Done
2. Update bookings > passengers relationship in schema to be optional
 - a. Done
3. Ensure foreign keys are mandatory, not optional in schema.
 - a. Done
4. Update ERD to remove non-key attributes, so it's different from the schema.
 - a. Will not implement: There are not enough non-key attributes to make the ERD too busy.
5. Add M:N relationship for flights with employees to overview.
 - a. Done
6. Update schema object (table) names to match ERD in PascalCase
 - a. Done
7. Inquiry about distanceKilometers and durationSeconds being plural while remaining objects are singular.
 - a. Will not implement: There are no cases of <2 kilometers or seconds in this database.
8. Update passengerID in outline to passengerId
 - a. Done
9. Get rid of background grid in next screenshot of Schema
 - a. Done
10. Review if an additional "make and model" category table should be made for aircraft.
 - a. Will not implement: Mostly due to adding additional complexity; however, each aircraft is uniquely configured so make/model will not necessarily describe the seat quantity or other characteristics.
11. Update VARCHAR lengths to be more appropriate to max phone number and email lengths (14 and 320 respectively)
 - a. Updated to 20 and 320 respectively
12. Add CASCADE DELETE to Aircraft table to delete seats on that craft
 - a. Updated Seats to delete if Aircraft is deleted
13. Add CASCADE DELETE to ensure that deleting a passenger deletes their bookings
 - a. Done
14. Add CASCADE DELETE so when a flight is deleted, bookings and assignments are deleted.
 - a. Done
15. Add CASCADE DELETE to ensure when an employee is deleted, their assignments are deleted.
 - a. Done
16. Add CASCADE DELETE to ensure that if a route is deleted, the flights on that route are deleted.
 - a. Implemented a DELETE RESTRICT; we want to delete a flight prior to a route being decommissioned. This prevents widespread changes to Bookings and Assignments without the user considering them. Also implemented for Flights FK to Aircraft.
17. Update Sample data to correct bookings information in passengerId.

- a. Done via updating screenshot to avoid cutting off data.
18. Update Sample data to either add extra row into sample or remove from the DDL sample insert
- a. Done: Removed sample from DML sample (see team initiated change for passengerId)
19. Add Group Name and Member Names to DDL.SQL file
- a. Done
20. Team Initiated Change:
- a. Updated Bookings to require passengerId.
21. Team Initiated Change:
- a. Updated relationships between the following.
 - i. Employees and Assignments: An employee may have no assignment
 - ii. Flights and Assignments: A flight may have no assignment
 - iii. Flights and Bookings: A flight may have no booking
 - iv. Aircraft and Flights: An aircraft may have no flights
 - v. Routes and Flights: A route may have no flights.
 - vi. Passengers and Bookings: Updated Schema to match ERD (a passenger may have no bookings).

Part 1 Feedback received from Peer Reviewers and TA

TA: Neil King

This is a great start, something to keep in mind is that there are only 4 total tables required excluding the intersection table. Based on your ERD it looks like you have 8 relationships, while this is not a problem, I would consider cutting a couple of the tables. For the ERD, since these are showing an outline of the problem that this system would solve, there doesn't have to be this much detail, until we do the schemas. I would also consider indicating what relationship is optional in a sentence in addition to your outline, this can make it easier to see.

Feedback Recieved from: Alexander Fox (AF), Payton Bradfield (PB), Joshua Bengé (JB), and Jared Northrop (JN)

Does the overview describe what problem is to be solved by a website with DB back end? If yes, summarize. If not, what changes would better support describing the problem to be solved?

- (AF) The overview clearly describes that the goal of the database will be to aid the airline with improving efficiency across multiple different departments, from flight schedules to passenger bookings and everything in between. There is not a description of a problem that is to be solved, although it could be inferred that the airline is looking for a way to improve efficiency and this database will help with that. A small expansion to the outline that more clearly states a problem that the database is solving could help but the outline is well explained otherwise.
- (JN) The overview describes the information airlines need to track to manage airline operations. It displays specific facts about the quantity of different types of information that the airlines receive yearly. The changes I would make would be to discuss the specific areas of an airline enterprise that the database will track. The outline is very broad in what it could do.
- (JB) Yes, the problem is described. Fly Anywhere Airlines is in need of a database that is able to keep track of its flights, planes, passengers, and employees.
- (PB) Yes, the problem is clearly outlined as a need for an organized system to manage airline operations, like flight schedules, crew assignments, aircraft management and passenger bookings.

Does the overview list specific facts? If yes, summarize what the facts illustrate about the proposed DB solution. If not, what facts would better support illustrating the scope and scale of the proposed DB solution?

- (AF) The overview does state specific facts for the number of flights the airline deals with every year and the number of unique routes that it operates. The overview lists less specific numbers for things like number of employees and number of aircraft. Since these are entities being tracked in the database, it may be helpful to be more specific with these two things for better clarity of the scope and scale of the proposed DB solution.
- (JN) The proposal lists some of the facts about how many users, airplanes, and routes. What I think it needs to specify is what areas of an airplane enterprise the database will track. Airlines have to keep track of inventory, employees(both information and what planes/routes they fly), maintenance, Customer info, etc. The scope is not clearly defined on what this database will do in the overview.
- (JB) The overview does list specific facts, but I do think your overview could use a little more detail. For instance, what about the cost of each flight, how much each ticket costs, etc. This overview touches on all the logistics required for an airline to keep track of, except for money related logistics.
- (PB) Yes it explicitly states that the airline handles over 20,00 flights a year across more than 100 routes. While it doesnt give specific ballpark figures for other attributes it states that it will deal with thousands of employees hundreds of aircraft and millions of passenger bookings

Are at least four entities described, and does each one represent a single idea to be stored as a list? If yes, summarize. If not, based on the course material, what changes can you suggest to improve?

- (AF) There are at least four entities described and each one does represent a single idea to be stored in a list. The entities pretty clearly match up with the things that the database intends to keep track of based on the outline. If I could make any improvements here, I would maybe mention that the database would keep track of Seat details in the overview as it is seemingly the only thing not mentioned there.
- (JN) There are more than four entities with additional intersection tables defined on top of that. I like that you made a distinction between intersection tables and entities. The entities reflect what an airline needs to track.
- (JB) Yes, there are 6 entities with 2 placeholder tables for M:N relationships. The 6 entities are Flights, Aircraft, Routes, Seats, Passengers, and Employees.
- (PB) Yes the entities are Flights: representing individual flight records, Aircraft: representing details about planes in the fleet, Routes: representing the routes taken between airports, Seats representing seats on aircraft, Employees: representing the employees of the airline, assignments: representing employees assigned to particular flights, Passengers representing customers and their contact info and finally booking representing individual passenger bookings on planes and the seat they are assigned to.

Does the outline of entity details describe the purpose of each, list attribute data types and constraints, and describe relationships between entities? If yes, summarize. If not, based on the course material, what changes can you suggest to improve?

- (AF) The outline of entity details does a great job describing the purpose of each and their relationships to other entities. Primary and Foreign Keys are well explained and the types of relationships are clearly listed. Great description of existing intersection tables as well.
- (JN) The entities and attributes are clearly defined. The constraints and data types look accurate to the field. The naming of entities and attributes are self-explaining. All entities' purposes are described.
- (JB) Yes, the outline describes the entities, provides their attributes with proper data types, and explains the relationships between all the entities. Aircraft provides information about the plane, Flights provides information about the aircraft and route, Routes provides information about the flight logistics, Seats provides information about the seat number and Aircraft, and Passengers provides information on the passengers.
- (PB) Yes each entity has an apt description that lists the purpose for the tables. While the attributes data types and constraints are perfectly outlined i.e. routeId INT or emailAddress VARCHAR(1000) NN. There is also a clear description of the relationship between each of the entities.

Are 1:M relationships correctly formulated? Is there at least one M:M relationship? Does the ERD present a logical view of the database? If yes, summarize. If not, based on the course material, what changes can you suggest to improve?

- (AF) M:N Relationships are clearly defined with intersection tables. The existing 1:M relationships seem to be correctly formatted. Others have mentioned the possibility of a Seat not having a passenger and needing to take this into account. I believe that reviewing the cardinality (optional or mandatory) of the relationships could help as all of the relationships are mandatory currently.
- (JN) The relationships are listed for each entity which is reflected in the ERD. I think the relationships look correct for the most part in how they are defined. I agree with Joshua that since bookings link seats to flights and passengers, there needs to be a way for a seat not to have a passenger. I think the way you thought about it was that a passenger creates a booking which needs a seat and flight, however with the M:M relationship created between seats and flights this creates the problem of bookings that aren't filled with passengers.
- (JB) There are two M:M relationships, but I think the ERD relationships could use a little work. For instance, not every seat of a flight will always be booked, it is possible to have an empty seat. Also, it is possible for a flight to not be part of a flight. So, it would be beneficial for the ERD to have the inclusion described in the relationships.
- (PB) Yes 1:M relationships are present and formatted correctly for example Routes and Flights. M:M relationships are there as well such as the relationship between Employees and Flights via

Assignments. The ERD is also there and provides a logical view of the database described in the outline.

Is there consistency in a) naming between overview and entity/attributes b) entities plural, attributes singular c) use of capitalization for naming? If yes, summarize. If not, based on the course material, what changes can you suggest to improve?

- (AF) There is consistency in naming between overview and entity/attributes, entities plural, attributes singular, and the use of capitalization for naming. The style guide seems to be adhered to well. The only thing I would say to improve here (which I mentioned earlier) would be to mention Seats in the overview for greater consistency between the overview and the entities.
- (PB) Yes entity names like Flights and Aircraft are plural though admittedly I had to double check that Aircraft is both singular and plural. Attribute names are Singular for example flightId or jobRole. The capitalization of proper names is also correct and consistent throughout the table.
- (JN) The naming looks good. Each name is clear and self documenting. The capitalization and pluralization is followed. Lower Camel Case is used for all attributes and all entities use upper camel case.
- (JB) Yes, all the entities are capitalized and plural. All the attributes are in camel case, with the PK values being the name of the table + Id. For example, Flights and the PK being flightId.

Fixes Based on Feedback from Part 1

1. Team-initiated design change:
 - Altered seatId to auto-increment to match the remaining Primary Keys.
2. Feedback that the problem statement is missing from the overview, and that the overview is too broad:
 - We updated the Overview Section to more explicitly state the problem being solved
3. Suggestion to clarify Seats in cases where a passenger has not booked:
 - We added Cardinality to the relationship between Bookings and Passengers to clarify that a Passenger ID can be NULL. *(Reverted in Step 3)*
4. Suggestion to cut table count:
 - We did not incorporate this after clarifying with that TA that a 3 person group requires more objects.
5. Suggestion to include more specific facts in overview for employees and aircraft:
 - We did not make specific changes based on this feedback, as we already describe a general range of these facts.
6. Suggestion to add Seats in overview:
 - We added information about Seats to the problem statement and to the facts information.
7. Suggestion to add revenue and ticket costs:
 - We did not include this as the database is already over the minimum required objects, and payment information would potentially add too much complexity.
8. Feedback that a flight may not be part of a flight:
 - We did not incorporate this feedback due to the meaning being unclear.
9. Team-initiated design change:
 - Altered some varchar values to enum lists (seatClass, aircraftStatus)

Project Outline and Database Outline

1. Flights

- a. Stores records of all flights, past and upcoming.
- b. Attributes:
 - i. **flightId**: int, auto_increment, unique, PK
 - ii. **departureAt**: datetime, not NULL, in UTC time zone
 - iii. **routeId**: FK to Routes table
 - iv. **aircraftId**: int, FK (reference to **aircraftId** in Planes)
 - v. **flightStatus**: enum('scheduled', 'delayed', 'cancelled', 'in progress', 'completed') notNULL
- c. Relationships
 - i. M:1 with Aircraft
 - ii. M:1 with Routes
 - iii. M:N to Seats via Bookings intersection table
 - iv. M:N to Passengers via Bookings intersection table
 - v. M:N to Employees via Assignments intersection table

2. Aircraft

- a. Stores data about individual airplanes that can be assigned to flights.
- b. Attributes:
 - i. **aircraftId**: int, auto_increment, unique, PK
 - ii. **model**: varchar(250), not NULL
 - iii. **capacity**: int, not NULL
 - iv. **aircraftStatus**: enum('active', 'maintenance', 'inactive'), not NULL
- c. Relationships
 - i. 1:M to Seats
 - 1. aircraftID is used as a foreign key inside of the Seats entity.
 - ii. 1:M Flights
 - 1. aircraftID is used as a foreign key inside of the Flights entity.

3. Routes

- a. Stores routes between airports that are available for flight transit.
- b. Attributes:
 - i. **routeId**: int, auto_increment, unique, PK
 - ii. **originAirport**: varchar(4), not NULL.
 - 1. 3 or 4 character airport code for origin airport
 - iii. **destinationAirport**: varchar(4), not NULL
 - 1. 3 or 4 character airport code for destination airport
 - iv. **distanceKilometers**: int, not NULL
 - v. **durationSeconds**: int not NULL
- c. Relationship:
 - i. 1:M between Routes and Flights, **routeId** links each flight to a specific route
 - ii. routeId is used as a foreign key inside of the Flights entity.

4. Seats

- a. Stores information about a specific seat on a specific plane which can be booked by a passenger, allowing for easy passenger seat allocation and tracking.
- b. Attributes
 - i. **seatId**: int, auto_increment, not NULL, PK
 - ii. **aircraftId**: int, not NULL, FK
 - iii. **seatNumber**: varchar(10) not NULL
 - iv. **seatClass**: enum('economy', 'premium', 'business', 'first class') not NULL
- c. Relationship:
 - i. M:N to Passengers via Bookings intersection table
 - ii. M:1 to Aircraft

5. Passengers

- a. Stores passenger information. A passenger is defined as anyone who has ever booked a flight on our airline.
- b. Attributes:

- i. **passengerId**: int, auto_increment, unique, PK
 - ii. **name**: varchar(1000), not NULL
 - iii. **emailAddress**: varchar(1000), not NULL
 - iv. **phoneNumber**: varchar(100), NULL
- c. Relationship:
 - i. M:N between Passengers, Seats, and Flights via intersection table Bookings.

6. Employees

- a. Stores airline employee data. An employee is any person who has worked or will work on any flight on our airline.
- b. Attributes:
 - i. **employeeId**: int, auto_increment, unique, PK
 - ii. **employeeName**: varchar(1000), not NULL
 - iii. **jobRole**: enum('pilot', 'copilot', 'flight attendant', 'engineer', 'tarmac'), not NULL
 - iv. **phoneNumber**: varchar(100), not NULL
- c. Relationship:
 - i. M:N between Employees and Flights, via intersection table Assignments

Intersection Tables:

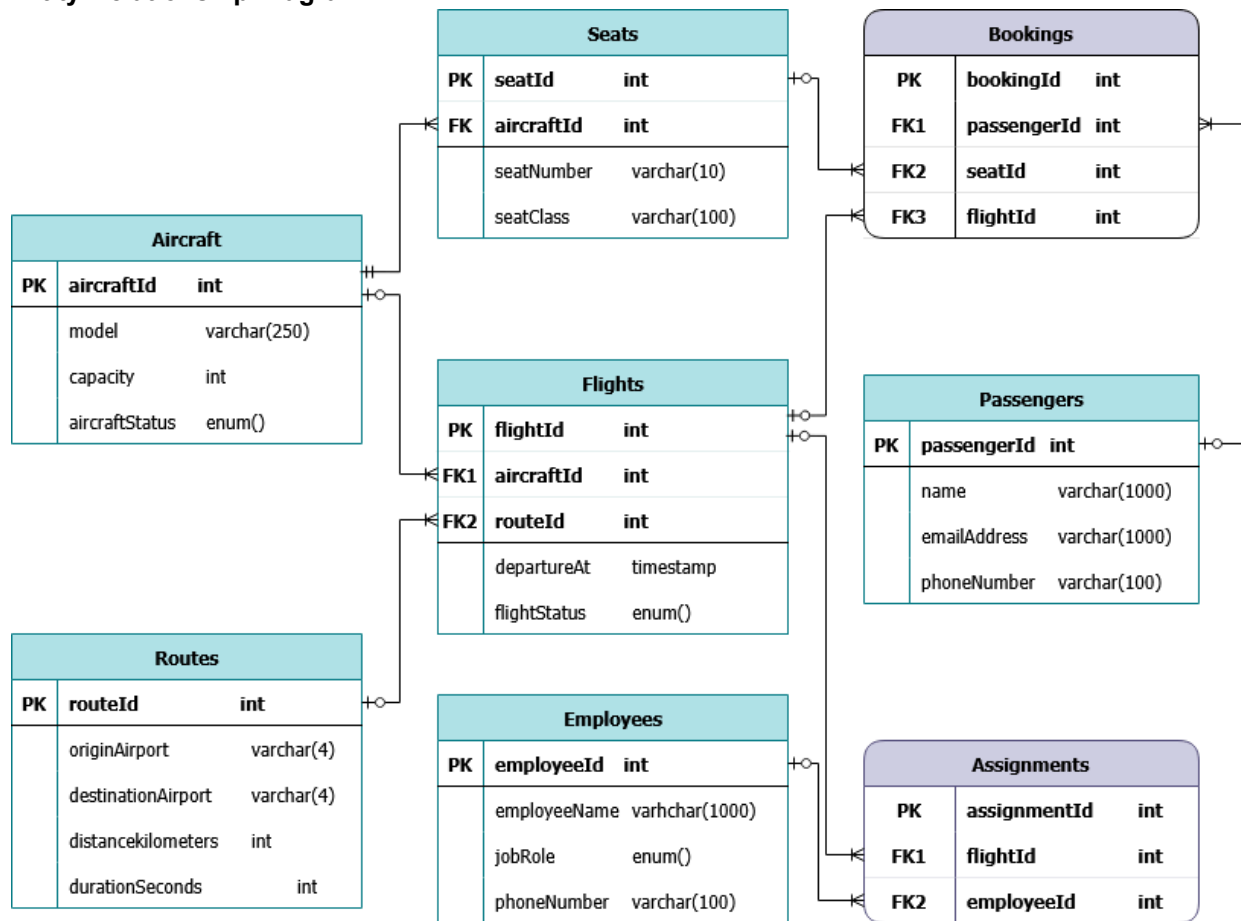
7. Bookings

- a. Join table for relationship between Flights, Passengers, and Seats. M:N relationship. Represents a seat that is or can be booked by a passenger; as such, the "passengerId" will remain NULL until that seat is booked. This relationship is maintained so that operations can determine available seats and unavailable seats on a given flight.
- b. Attributes
 - i. **bookingId**: int, auto_increment, unique, PK
 - ii. **flightId**: int, FK (reference to flightId in Flights)
 - iii. **seatId**: int, FK (reference to seatId in Seats)
 - iv. **passengerId**: int, FK (reference to passengerId in Passengers) * Can be NULL
- c. Relationships
 - i. M:1 with Flights via flightId
 - ii. M:1 with Seats via seatId
 - iii. M:1 with Passengers via passengerId

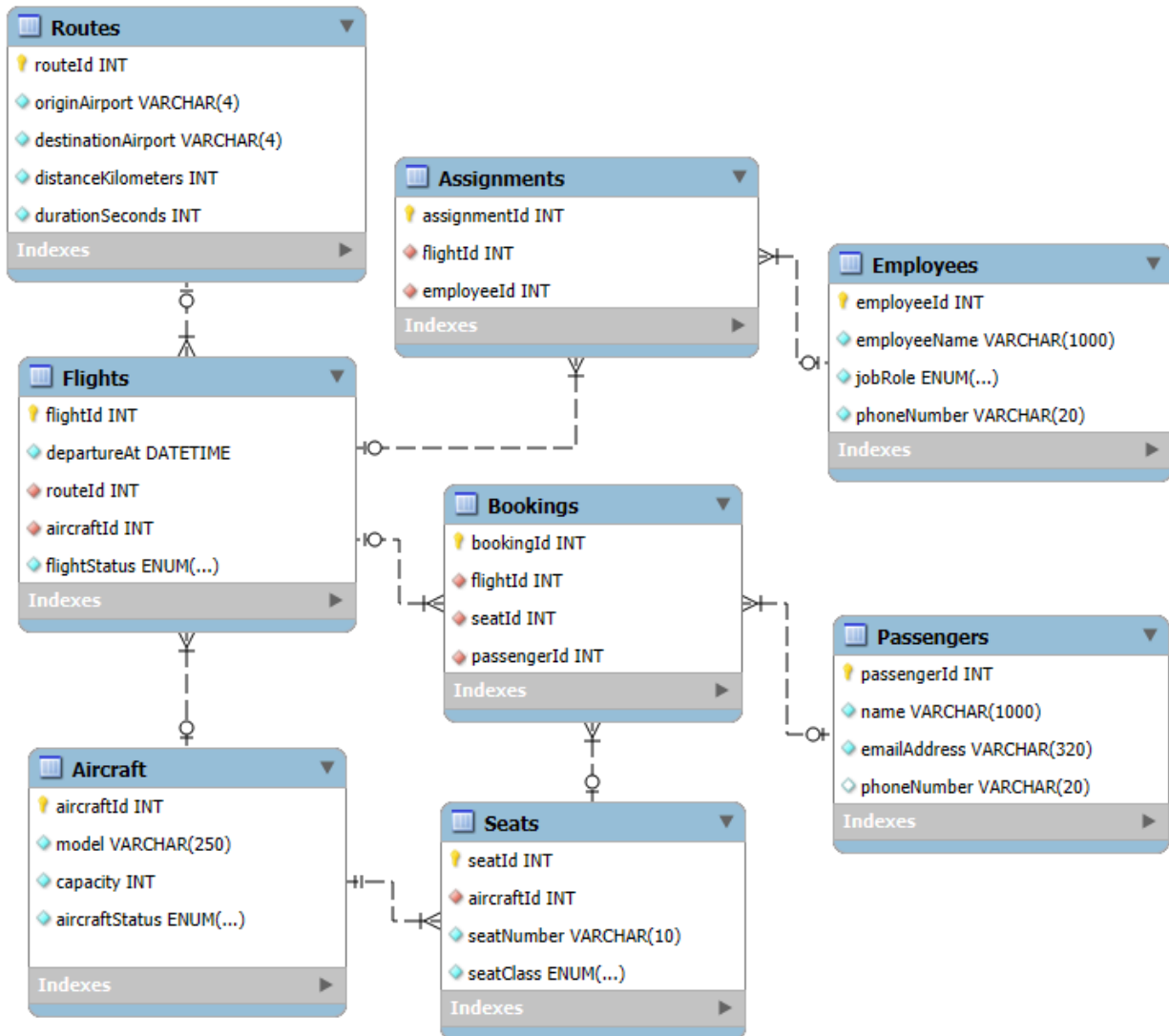
8. Assignments

- a. Join table between Employees and Flights M:N relationship. Represents an employee being assigned to work on a specific flight.
- b. Attributes
 - i. **assignmentId**: int, auto_increment, unique, PK
 - ii. **flightId**: int, FK (reference to flightId in Flights)
 - iii. **employeeId**: int, FK (reference to employeeId in Employees)
- c. Relationships
 - i. M:1 with Flights via flightId
 - ii. M:1 with Employees via employeeId

Entity-Relationship Diagram



Schema



Example Data

Assignments		
assignmentId	flightId	employeeId
1	1	1
2	1	2
3	3	1
4	3	3

Bookings			
bookingId	flightId	seatId	passengerId
1	1	1	1
2	1	2	2
3	3	3	1

Employees			
employeeId	employeeName	jobRole	phoneNumber
1	Rhonda Smith	pilot	347-203-1967
2	James Wilson	engineer	347-204-1969
3	Samuel Smith	flight attendant	347-352-1964

Passengers			
passengerId	name	emailAddress	phoneNumber
1	Katherine Pillsbury	pillsk@gmail.com	253-867-5309
2	Jack Mustonen	jackmustonen123@gmail.com	971-303-3385
3	Muhammad Akbar	muhammadakbar@gmail.com	923-453-3451

Seats			
seatId	aircraftId	seatNumber	seatClass
1	1	1	first class
2	1	2	first class
3	3	1	economy
4	2	1	first class

Routes				
routeId	originAirport	destinationAirport	distanceKilometers	durationSeconds
1	SEA	LAX	1545	9300
2	LAX	SFO	543	3600
3	SFO	SEA	1094	7200

Aircraft			
aircraftId	model	capacity	aircraftStatus
1	A220	133	active
2	A220	133	active
3	747	467	maintenance

Flights				
flightId	departureAt	routeId	aircraftId	flightStatus
1	2024-12-31 00:00:00Z	1	1	completed
2	2025-01-01 00:00:00Z	1	1	in progress
3	2025-01-02 00:00:00Z	2	2	scheduled