**Project: Human Resources Database Design**

**Project Overview**

In this project, you will design, build, and populate a database for the Human Resources (HR) Department at the imaginary Tech ABC Corp, a video game company. This project will start with a request from the HR Manager. From there, you will need to design a database using the foundational principals of data architecture that is best suited to the department's needs. You will go through the steps of database architecture, creating database proposals, database entity relationship diagrams (ERD), and finally creating the database itself. This project is important, as it is a scaled-down simulation of the kind of real-world assignments data architects work on every day.

**Business Scenario**

**Business requirement**

Tech ABC Corp saw explosive growth with a sudden appearance onto the gaming scene with its new AI-powered video game console. As a result, they have gone from a small 10 person operation to 200 employees and 5 locations in under a year. HR is having trouble keeping up with the growth since they are still maintaining employee information in a spreadsheet. While that worked for ten employees, it has become increasingly cumbersome to manage as the company expands.

As such, the HR department has tasked you, as the new data architect, to design and build a database capable of managing their employee information.

**Dataset**

The [**HR dataset**](https://video.udacity-data.com/topher/2020/July/5f2452ca_hr-dataset/hr-dataset.xlsx) you will be working with is an Excel workbook consisting of 206 records, with eleven columns. The data is in human-readable format and has not been normalized at all. The data lists the names of employees at Tech ABC Corp, as well as information such as job title, department, manager's name, hire date, start date, end date, work location, and salary.

**IT Department Best Practices**

The IT Department has certain Best Practices policies for databases you should follow, as detailed in the **[Best Practices document](https://drive.google.com/file/d/1YdBZPpaIQvnD9NbgkeLMb5PeFtnhGGRP/view?usp=sharing" \t "_blank)**.

**Project Steps**

The project is broken down into three major steps, with a fourth optional step. Each step informs the step that follows. The project has been designed to mimic the flow of a real-world database design project.

**Step 1**: This step is all about information gathering and putting it down on paper. In this step, you will be expected to complete both business and technical proposal documents required to begin the database design process.

**Step 2**: Step 2 is where you will walk through the design process. You will be using Lucidchart to create database diagrams that will be used to build an actual database.

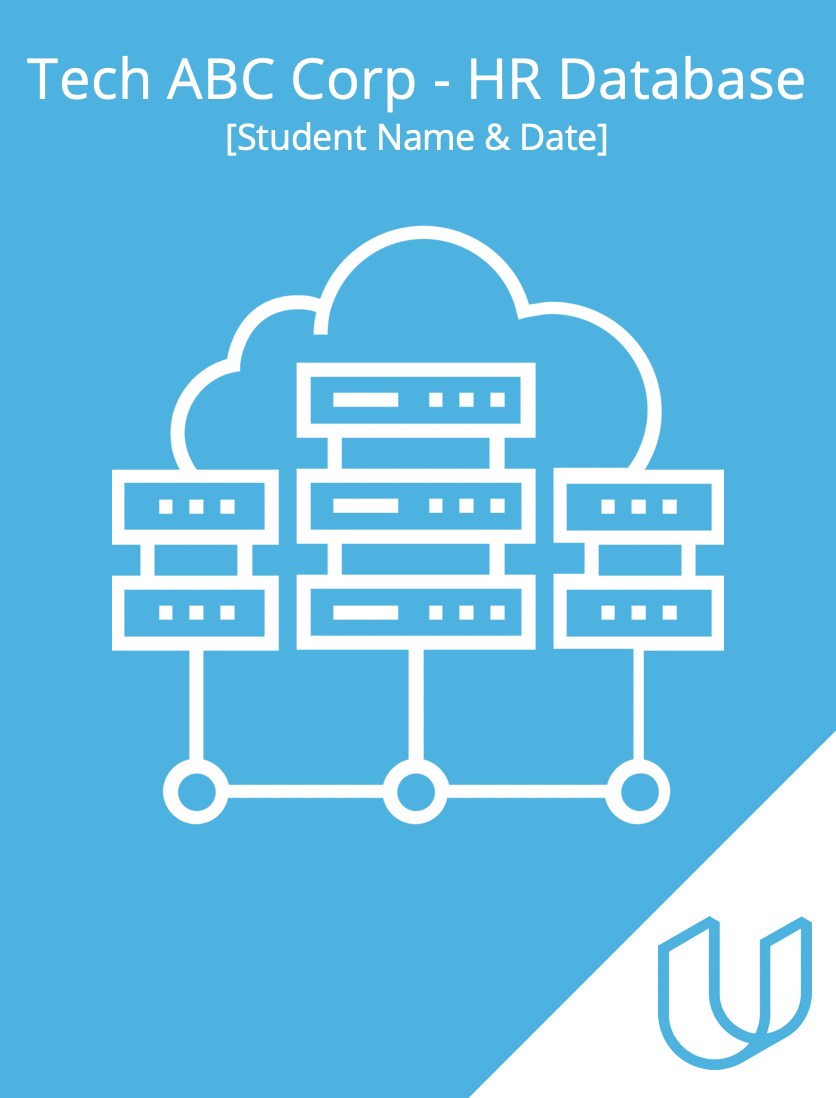
**Step 3**: In step 3, it is time to start coding. You will use SQL DDL commands to create a database. You will then populate your database with the [**HR dataset**](https://video.udacity-data.com/topher/2020/July/5f21c380_hr-dataset/hr-dataset.csv). From there, you will demonstrate the effectiveness of your database by completing some SQL CRUD exercises.

**Step 4**: This last step is optional but gives you some ideas on how to make your completed project stand out from the rest - adding items like views, stored procedures, and security.

**Get start**

You can use **[this starter template](https://video.udacity-data.com/topher/2020/October/5f7b5a02_starter-template/starter-template.pptx" \t "_blank)** to help you in completing all of the steps as detailed further below. Note that your project will be graded according to the **[project rubric](https://review.udacity.com/" \l "!/rubrics/2876/view" \t "_blank)**, so you should make sure to keep it handy, although the starter template should help guide you toward a successful submission.

You can use either the **[Udacity workspace](https://classroom.udacity.com/nanodegrees/nd038-beta/parts/6c9789ad-b982-4c21-8b30-482fdd29d1c0/modules/71dd650e-81b5-4d7d-83cd-9e2f1a1a0b1e/lessons/99441cdb-bee9-430e-ba69-6d8294635dc6/concepts/de446184-3ea5-40df-96a1-b537b516321f" \t "_blank)** provided to you or other systems of your choice to complete the project.



Template cover page

Check if you have everything you need for the project.

Task List

* 

I have access to the Best Practices document

* 

I have access to the HR dataset

* 

I have access to the starter template

## Submission

You will submit a file, containing your completed report from the template, screenshots included within that template, as well as any necessary .sql files that demonstrate various SQL commands and queries requested of you within the template.

## Step 1: Data Architecture Foundations

### The Memo

You have received the below memo as an introduction to the HR department's database request.

Hi,  
  
Welcome to Tech ABC Corp. We are excited to have some new talent onboard. As you may already know, Tech ABC Corp has recently experienced a lot of growth. Our AI powered video game console WOPR has been hugely successful and as a result, our company has grown from 10 employees to 200 in only 6 months (and we are projecting a 20% growth a year for the next 5 years). We have also grown from our Dallas, Texas office, to 4 other locations nationwide: New York City, NY, San Francisco, CA, Minneapolis, MN, and Nashville, TN.  
  
While this growth is great, it is really starting to put a strain on our record keeping in HR. We currently maintain all employee information on a shared spreadsheet. When HR consisted of only myself, managing everyone on an Excel spreadsheet was simple, but now that it is a shared document I am having serious reservations about data integrity and data security. If the wrong person got their hands on the HR file, they would see the salaries of every employee in the company, all the way up to the president.  
  
After speaking with Jacob Lauber, the manager of IT, he suggested I put in a request to have my HR Excel file converted into a database. He suggested I reach out to you as I am told you have experience in designing and building databases. When you are building this, please keep in mind that I want any employee with a domain login to be have read only access the database. I just don't want them having access to salary information. That needs to be restricted to HR and management level employees only. Management and HR employees should also be the only ones with write access. By our current estimates, 90% of users will be read only.  
  
I also want to make sure you know that am looking to turn my spreadsheet into a live database, one I can input and edit information into. I am not really concerned with reporting capabilities at the moment. Since we are working with employee data we are required by federal regulations to maintain this data for at least 7 years; additionally, since this is considered business critical data, we need to make sure it gets backed up properly.  
  
As a final consideration. We would like to be able to connect with the payroll department's system in the future. They maintain employee attendance and paid time off information. It would be nice if the two systems could interface in the future  
  
I am looking forward to working with you and seeing what kind of database you design for us.  
  
Thanks,  
Sarah Collins  
Head of HR

### Let's get started

As mentioned in the memo above, you are going to design and build an HR database for Tech ABC Corp. However, before you can start building, you need to complete both a business and technical proposal form answering some important questions about the database you are going to build.

Refer to your **[project starter](https://video.udacity-data.com/topher/2020/August/5f2da78a_starter-template/starter-template.pptx" \t "_blank)** Step 1. There, you will find the questions you need to answer, as well as additional documentation from Tech ABC Corp's IT department concerning **[best practices for database design, data governance, and other concerns](https://drive.google.com/file/d/1YdBZPpaIQvnD9NbgkeLMb5PeFtnhGGRP/view?usp=sharing" \t "_blank)**.

While creating documentation is never high up on most people's list of fun things to do, it is still very important, as oftentimes business user expectations may not line up with technical realities. Having a proposal document to fall back on can save you a lot of head aches.

#### Hints

Make sure to keep the Business Requirement answers in plain English. Try to limit technical jargon, as this can often be off-putting to business partners. Remember, as a data architect, you will often have to work in two worlds (the business world, and the technical world) simultaneously.

## Step 2: Relational Database Design

This step is where you will go through the process of designing a new database for Tech ABC Corp's HR department. Using the HR dataset provided, along with the requirements gathered in step one, you are going to develop a relational database set to the 3NF.

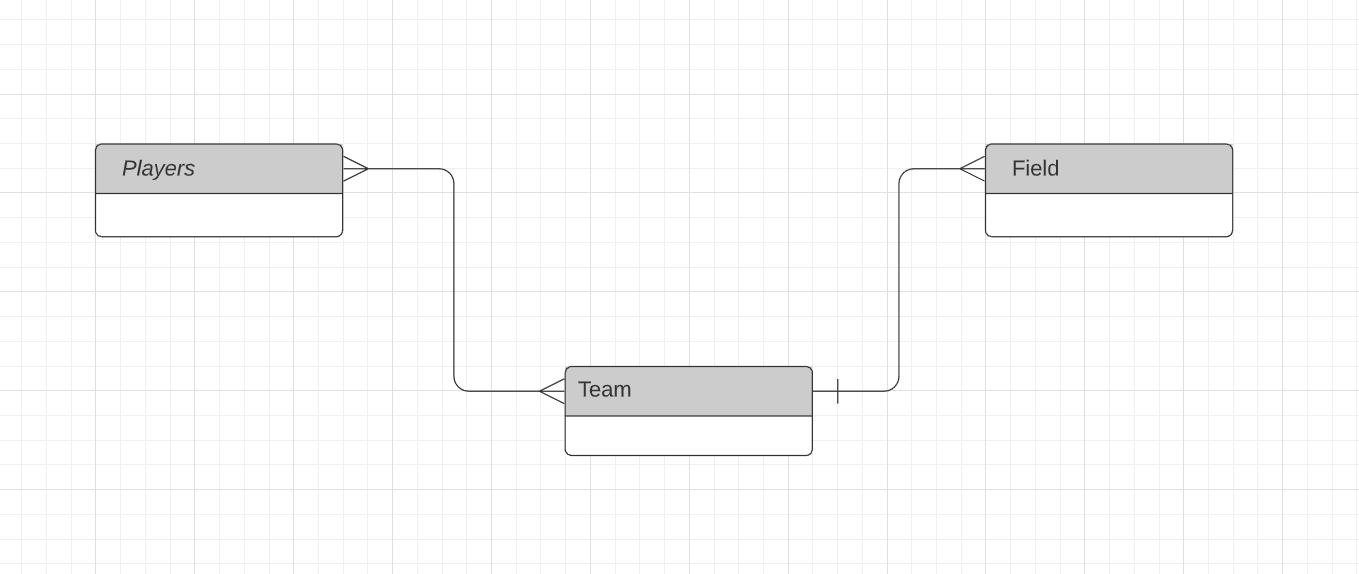
Using Lucidchart, you will create 3 entity relationship diagrams (ERDs) to show how you developed the final design for your data.

You will submit a screen shot for each of the 3 ERDs you create. You will find detailed instructions for developing each of the ERDs below and in the starter template.

### Conceptual ERD

This is the most general level of data modeling. At the conceptual level, you should be thinking about creating entities that represent business objects for the database. Think broadly here. Attributes (or column names) are not required at this point, but relationship lines are required (although Crow's foot notation is not needed at this level). Create at least three entities for this model - thinking about the 3NF will aid you in deciding the type of entities to create.

Use Lucidchart's built-in template for DBMS ED Diagram UML.

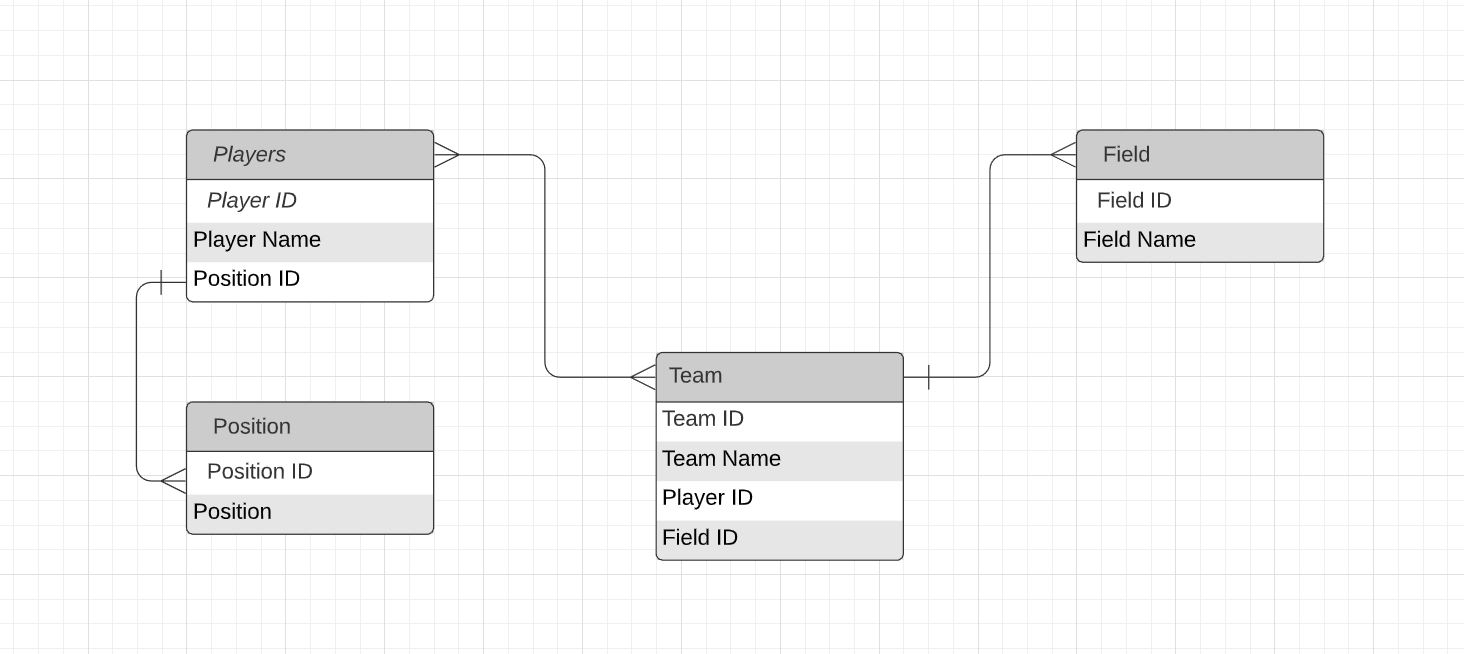


An example ERD after entity creation

### Logical ERD

The logical model is the next level of refinement from the conceptual ERD. At this point, you should have normalized the data to the 3NF. Attributes should also be listed now in the ERD. You can still use human-friendly entity and attribute names in the logical model, and while relationship lines are required, Crow's foot notation is still not needed at this point.

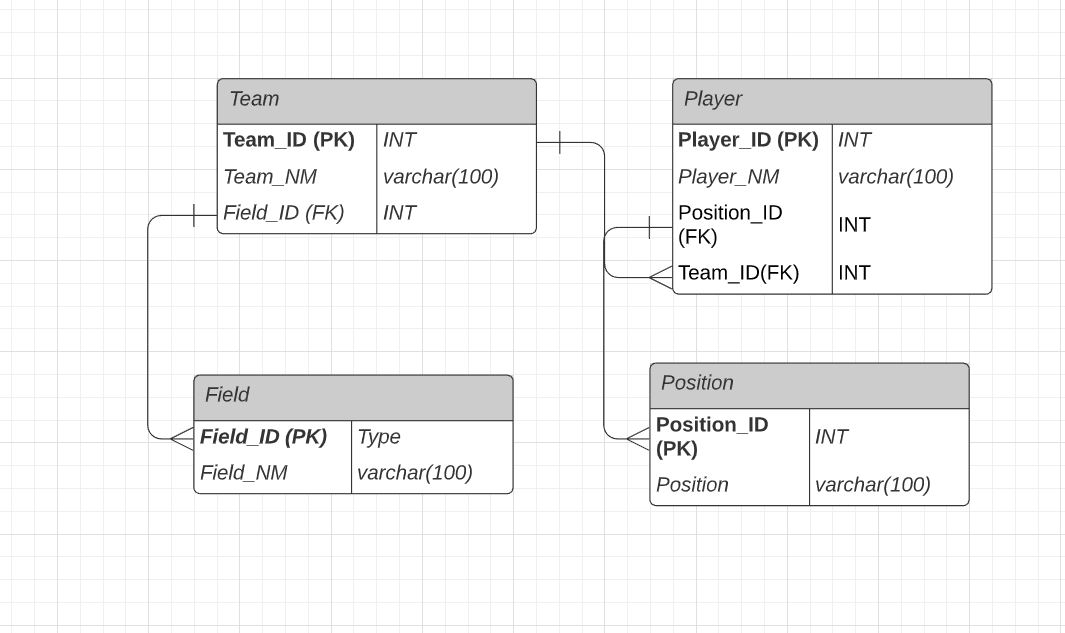
Use Lucidchart's built-in template for DBMS ED Diagram UML.



An example ERD with fields added

### Physical ERD

The physical model is what will be built in the database. Each entity should represent a database table, complete with column names and data types. Primary keys and foreign keys should also be represented here. Primary keys should be in bold type with the (PK) designation following the field name. Foreign keys should be in normal type face, but have the designation (FK) after the column name. Finally, in the physical model, Crow's foot notation is important.



An example ERD with data types, primary keys and foreign keys

### Hints

Make sure you follow the best practices when you create the data models.

The most important idea to keep in mind is to make sure you are applying table level security for any field identified as sensitive. Due to the limited scope of this project, this is the only way to ensure only certain users can access certain information.

This dataset requires at least 5 tables to be considered 3NF. More tables may be necessary (especially when considering table level security).

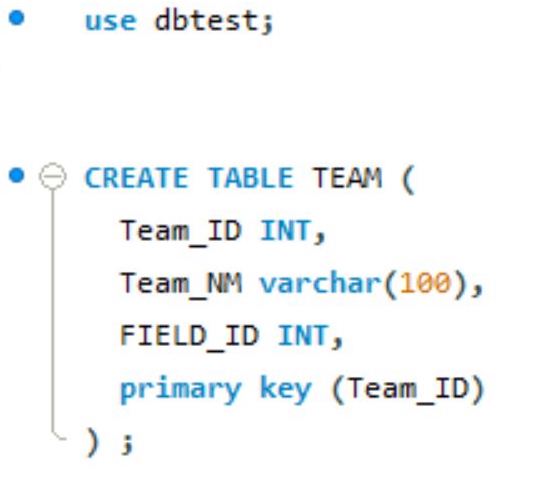
**Step 3: Create A Physical Database**

**Instruction**

In this step, you will be turning your database model into a physical database.

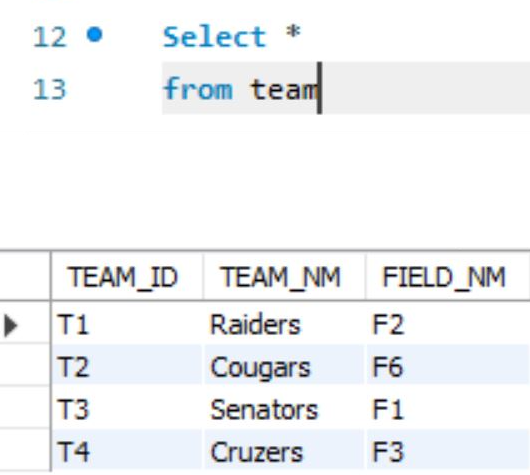
You will:

* Create the database using DDL SQL commands



An example query creating a team table within a dbtest database

* Load the data into your database, utilizing flat file ETL
* Answer a series of questions using CRUD SQL commands to demonstrate your database was created and populated correctly



An example query checking out all of the example team data. The query is a basic SELECT \* from team, with four teams as examples.

*Note: the images are examples. The image you submit will be different based on the SQL server you choose.*

**Submission**

For this step, you will need to submit **SQL files** containing all DDL SQL scripts used to create the database.

You will also have to submit screenshots showing CRUD commands, along with results for each of the questions found in the starter template.

**Hints**

Your DDL script will be graded by running the code you submit. Please ensure your SQL code runs properly!

Foreign keys cannot be created on tables that do not exist yet, so it may be easier to create all tables in the database, then to go back and run modify statements on the tables to create foreign key constraints.

After running CRUD commands like update, insert, or delete, run a select \* command on the affected table, so the reviewer can see the results of the command.

## Step 4: Above and Beyond (optional)

This last step is called Above and Beyond. In this step, I have proposed 3 challenges for you to complete, which are above and beyond the scope of the project. This is a chance to flex your coding muscles and show everyone how good you really are.

These challenge steps will bring your project even more in line with a real-world project, as these are the kind of “finishing touches” that will make your database more usable. Imagine building a car without air conditioning or turn signals. Sure, it will work, but who would want to drive it.

I encourage you to take on these challenges in this course and any future courses you take. I designed these challenges to be a challenge to your current abilities, but I ensured they are not an unattainable challenge. Remember, these challenges are completely optional - you can pass the project by doing none of them, or just some of them, but I encourage you to at least attempt them!

### Challenge 1

Create a view that returns all employee attributes; results should resemble the initial Excel file.

Views are what most users work with when using databases. Tables are coded in machine friendly 3NF, but views allow you to create a human-readable alternative for users.

### Challenge 2

Create a stored procedure with parameters that return current and past jobs (include employee name, job title, department, manager name, start and end date for the position) when given an employee name.

Just like views, stored procedures make the database appear to work effortlessly. Instead of creating a complicated query, they just need to provide a name to a stored procedure to get the results they want.

### Challenge 3

Implement user security on the restricted salary attribute.

Creating a separate table for salary makes applying security rules for that information possible. However, just creating the table is only step one. Create a user name NonMgr and REVOKE that user’s rights from viewing the salary table.

Para trabajar sobre su POSTGRESQL

The instructor has created a SQL script (StageTableLoad.sql) that will create a "proj\_stg" table with all the data from the excel workbook so you don't have to import the excel file into the workspace. From the "proj\_stg" table, you will be responsible for creating an ETL SQL script.

Correr el código .SQL del archivo StageTableLoad.sql en su database.