

Name: Han Zhang

## **Table of Contents**

Project Direction Overview	2
Use Cases and Fields	3
Structural Database Rules	7
Conceptual Entity-Relationship Diagram	9
Full DBMS Physical ERD	10
Stored Procedure Execution and Explanations	14
Question Identification and Explanations	17
Query Executions and Explanations	18
Index Identification and Creations	21
History Table Demonstration	24
Data Visualizations	27
Summary and Reflection	29

## Project Direction Overview

I want to create a database app that can give veterans an easier time finding a job after their military service. While not all military jobs have a civilian equivalent, most veterans still develop a wide range of skill sets that can be applied to a bunch of different civilian jobs. Oftentimes these people themselves do not realize how useful their skillset is and often have a difficult time re-adjusting to civilian jobs. In addition I don't want the database to be just limited to jobs, but also include schools with degrees that support what they have done while in service. The way I envision this database to work is that veterans can input their information and the database will filter the results and present jobs that fit the person's profile, if no profile is created this database can also act as a general job site.

Ideally this sort of database will be used by three different groups. The first group previously mentioned are the veterans, when they finish their service and develop certain skills they can create a profile on this app and find jobs that fit their skillset. The second group of people to use this database would be the corporations and companies seeking employees. The benefit to these people is that when they get a job application from this source it guarantees a certain amount of experience and ability. This guarantee can be provided by the VA, so that there are no fake profiles. The last group that would use this database would be the government to obtain employment data and statistics to show growing or declining trends in the job market.

An example of how this database would work can be as follows, Cpt John Doe decides that he doesn't want to re-enlist so he decides to use this database to find a job after he leaves the army. He would put in his years of service, his MOS, and a general skill set. This in turn will create a list of different jobs for him to apply to. After his applications are sent in, the companies he applied for would receive his application with a guarantee from the VA of his time in service. This can assure the companies that the applicant is reasonably capable of the job he has applied for. The reason I'm interested in such a project is because after I got out of the army I realized a lot of my peers didn't get out because they felt that their skill set would be unusable as a civilian and most feel forced to stay in for twenty years. This database and subsequent app can be used to alleviate their worries and show that they have plenty of options

## Use Cases and Fields

The first use case would be when a veteran nears the end of his service and hasn't re-enlisted. He will receive a notice or e-mail from the VA for this app along with his personal code when he registers to validate his profile.

1. Veteran receives an email explaining this app
2. This person goes to the website to register an account
3. User registers an account with the code provided by the VA

Field	Contents	Necessity
Email	A point of contact for the user	The email not only serves as a point of contact but also a login to access the app
Name	Name of the account holder	To show the name of the account holder
VA Code	Unique code provided by the VA	To prove that this person is a veteran and has completed or is near completing his service
Password	The users account password	To secure the account

The database being designed would at this point log the account information such as Email, name, and password. The VA code would be provided from a separate database to validate the registration process.

The second use case would be for the applicant to fill in his work experience and details

1. User logs into the App
2. App requests user to fill out his profile

Field	Contents	Necessity
MOS	What the users job title was	It shows where his field of expertise lies and what similar jobs to this are available in the civilian sector
Years of Service	How long he has been in the military	Can be used as a rough guideline to years of experience
Security Clearance	What level if any security clearance the user has	Certain jobs will give priority to those with a security clearance
Preferred Field	What field the user prefers to work in	Helps the database filter job results

No Degree	The user does not have a degree	Shows the education level of the applicant
Bachelor's	Does the user have one or more bachelor degrees	This field can make the applicant more competitive against his competitors
Masters	Does the user have one or more master degrees	This field can make the applicant more competitive against his competitors

In the second use case the degree field can split into three different fields, therefore instead of a single field called degree, these fields will be changed into No degree, Bachelors, and masters, this separation is because there are people who have multiple degrees and may need to list them out. After the user fills out the information it will be inserted into the database for future actions and provide a general basis to help the user find a job.

The third use case would be done by companies and corporations that place the job ads onto the database.

1. Company HR logs into the app using a business side account
2. HR places a job ad with the necessary information
3. Job is inserted into the database

Field	Contents	Necessity
Company	Name of hiring company	Gives the user information about the hiring side
Salary	Potential salary	Necessary to determine if the person wants to apply for the job or not
Location	Where work is	Tells the user if he has to move or if its local
Work field	What field this type of work is in	Provides a category to filter information and gives additional information to the user
Description	A short description of the job	Gives the user a general idea of what the job entails
Availability	Whether its available or not	Tells the user if this job is still available or not

When a listing agent is adding jobs to this database all the above information will be put into the database and stored so that people on the user side can see the above information and determine which jobs they want or are certified for.

The fourth use case is from schools, while schools are different from jobs, being a student can also be considered a job, and I want this database to provide as many options and opportunities as possible. As applying to schools is somewhat different to that of a job I just want this database to provide simple introductions to schools, but will redirect to the school themselves for the application process.

1. School admittance offices will use business side accounts to login
2. School will provide a quick overview of the school along with admittance information
3. Information will be uploaded into the database
4. School will provide an external link for those interested in applying.

Fields	Contents	Necessity
School Name	The name of the school	It is necessary for the user to know what school hes applying for
Application timeframe	When deadlines for applications are	Gives the user and idea of whether or not he has enough time to apply
External Link	Link to the school's own application page	Actual application process from the school
Tuition	How much tuition is	It is important to know how much it will cost the user to go to school and whether or not the VA can help
Recommendations	Recommendations required by school	How many recommendations the school generally requires
Required Tests	Tests required by school	Which standardized tests is required
Required Transcripts	Transcripts required by school	Whether or not the school requires transcripts

A lot of the information for applying to a school is similar to that of applying for a job, so from the database perspective all this information will also be stored in the database, but this information is not nearly as in-depth as the job application information, because this is just a small offshoot of the main app. The requirements can be considered too general so it needs to be split into other fields, in my experience schools usually have similar requirements which are letters of recommendation, transcripts, and some standardized testing, so we replace the requirements with these fields.

The last use case would be a veteran applying to a new job.

1. Veteran logs into his account with a complete profile
2. Veteran searches for a job in his field of interest
3. Veteran uploads a resume to a certain job
4. Veteran writes his preferred method of contact
5. Veteran waits for a response

Field	Content	Necessity
Resume	The users resume to show relevant work experience	Baseline requirement for most job applications
Contact Information	How the company can contact the user, usually either email or phone number	Allows for passing information related to the application
Application Status	Shows the status of the application	The status of the application lets the user know where he stands in the process and if he has to take further action
Company	Name of the corporation	What the company or business is behind the job
Job title	Title of job	What job the user is applying to

This use case uses the database to store all the application files allowing for communication between the applicant and the company he is applying to. The resume field is used to show if one is sent or not, the application status lets the user know where he stands in the process, and whether or not the applicant has been contacted, and the other fields are relevant supporting information related to the job.

## Structural Database Rules

Using the 5 different use cases allows for creating a set of database rules

The first use case is:

1. Veteran receives an email explaining this app
2. This person goes to the website to register an account
3. User registers an account with the code provided by the VA

There are three entities of importance here, the user, his code, and his account. Using these things of note we can create the first relationships between them.

Rule 1. Veterans have one account, and accounts have one Veteran.

The second use case is:

1. User logs into the App
2. App requests user to fill out his profile

Looking at the chart of information in the second use case, most of these things can be considered attributes, however under requirements there is an entity known as degree. From this we can derive a relationship between User and Degree. Furthermore there are different types of degrees so we can turn these into subtypes.

Rule 2. Users can have zero to many degrees, degrees can have zero to many users

Rule 3. Degrees have multiple subtypes, it can be bachelors, masters, both, or neither

The third use case is:

1. Company HR logs into the app using a business side account
2. HR places a job ad with the necessary information
3. Job is inserted into the database

In this use case we can see that compared to previous use cases there are two account types, so there is a subtype division here.

Rule 4. Each account has a sub-type, user and corporate.

Next we have considered the different entities related to the account. There are a few different relationships at this level. A company can list one or multiple jobs through the account, these jobs can range from different fields, and occur in different places. From this multiple rules can be derived.

Rule 5. Each job is associated with one account, each account can have zero to multiple jobs

Rule 6. Each job may have one or multiple locations, and each location may have one or multiple jobs

Rule 7. Each job may be under one field, but each field can have multiple jobs.

Following these rules the entity of location can be further divided into multiple subtypes in an almost infinite number of ways. For the sake of simplicity we will divide it as follows:

Rule 8. Each location can be in a city, countryside, or none of these places.

The fourth use case is:

1. School admittance offices will use business side accounts to login
2. School will provide a quick overview of the school along with admittance information
3. Information will be uploaded into the database
4. School will provide an external link for those interested in applying

There are some similarities between this use case and the third use case but there are also some differences, like with accounts the rule is maintained that each school will only have one account. In relation to the database this set of information is considered to be secondary to the purpose of the database, and only provides a very minimal information, but allowing the user to get more information from external sources, therefore the rules applied in this section will be very simple

Rule 9. Each school has one or more external links, Each external link has one school.

Rule 10. Each school will have one or more tuition rates, each tuition rate is related to one school

Rule 11. Each school will have one or more requirements, each requirement can be associated with one or more schools.

The tuition rate can be simply divided into full time and part time

Rule 12. Tuition rates are either full time or part time.

The last use case is:

1. Veteran logs into his account with a complete profile
2. Veteran searches for a job in his field of interest
3. Veteran uploads a resume to a certain job
4. Veteran writes his preferred method of contact
5. Veteran waits for a response

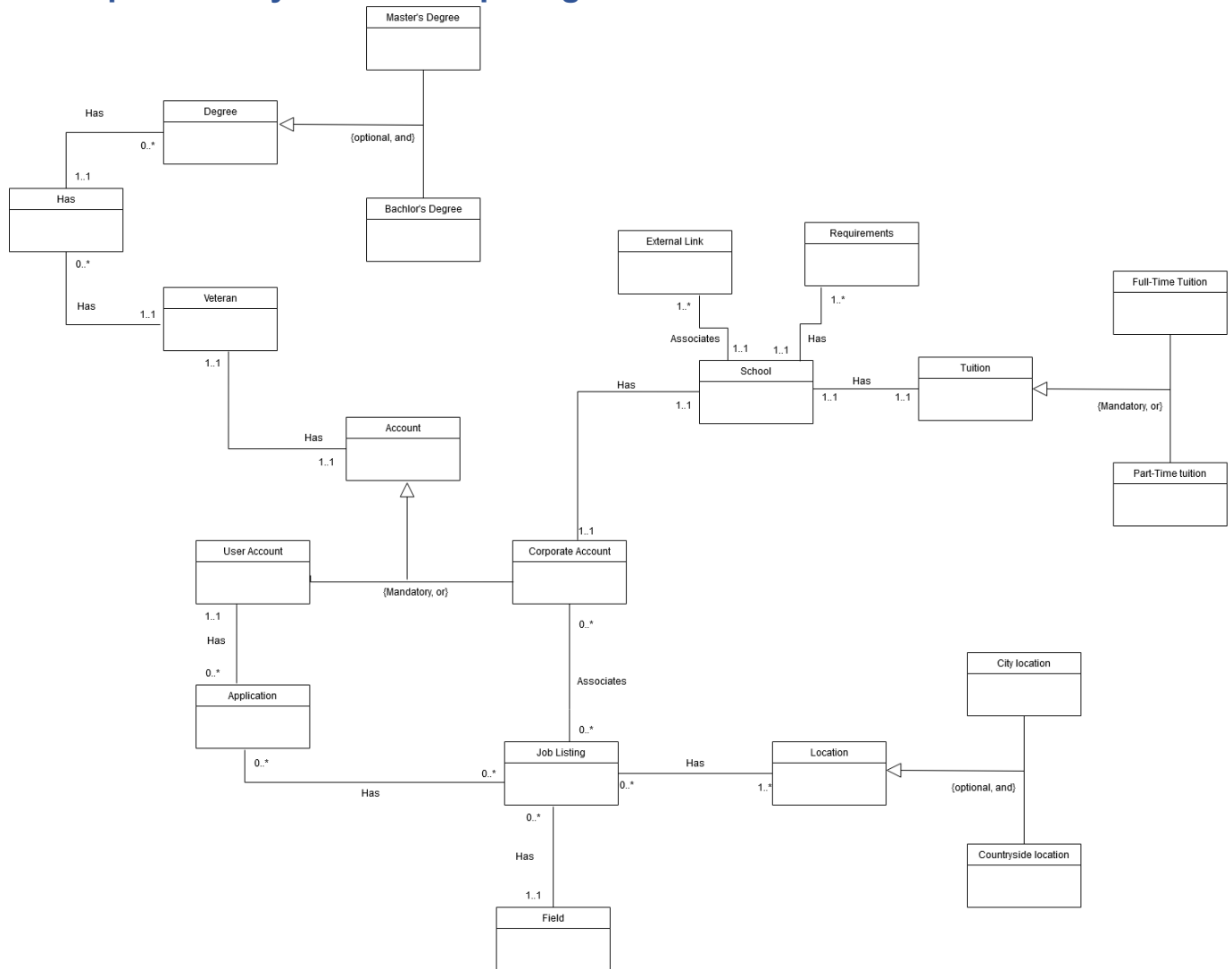
There are three entities at play in this use case, they are jobs, accounts, and applications. In this scenario the accounts will have a relationship with the applications, and the application will have a relationship with jobs.

Rule 13. Each account can have zero or multiple applications, each application can have only one account

Rule 14. Each Application can have zero or more or more jobs, and each job can have zero or more applications.



# Conceptual Entity-Relationship Diagram



Rule 1. Veterans have one account, and accounts have one Veteran.

Rule 2. Users can have zero to many degrees, degrees can have zero to many users  
(Due to it being a many to many relationship I have added a bridging entity)

Rule 3. Degrees have multiple subtypes, it can be bachelors, masters, both, or neither

Rule 4. Each account has a sub-type, user and corporate.

Rule 5. Each job is associated with one account, each account can have zero to multiple jobs

Rule 6. Each job may have one or multiple locations, and each location may have one or multiple jobs

Rule 7. Each job may be under one field, but each field can have multiple jobs.

Rule 8. Each location can be in a city, countryside, or none of these places.

Rule 9. Each school has one or more external links, Each external link has one school.

Rule 10. Each school will have one or more tuition rates, each tuition rate is related to one school

Rule 11. Each school will have one or more requirements, each requirement can be associated with one or more schools.

Rule 12. Tuition rates are either full time or part time.

Rule 13. Each account can have zero or multiple applications, each application can have only one account

Rule 14. Each Application can have zero or more or more jobs, and each job can have zero or more applications.

## Full DBMS Physical ERD

For the first part of creating a full DBMS I plan out all my attributes in a table format so it will become easier to add to the ERD

Table	Attribute	Datatype	Reasoning	Example
Master's Degree	MADegree	VARCHAR(255)	The master degree title	Master of computer information systems
Bachelor's Degree	BSDegree	VARCHAR(255)	The bachelor degree title	Bachelors of English
Bachelor's Degree	Credit Hours	DECIMAL(3)	Amount of credits done in degree	128
Degree	Year completed	DECIMAL	Shows when the degree was earned	2020
Degree	School	VARCHAR(255)	Shows where the degree was completed at	Boston College
Degree	Awarded To	VARCHAR(255)	Name on the degree	Jordan Peterson
Veteran	First Name	VARCHAR(255)	Name of person	Jordan
Veteran	Last Name	VARCHAR(255)	Last name of person	petersun
Veteran	Branch of Service	VARCHAR(100)	Which branch he served in	Air force
Veteran	Years of Service	DECIMAL(3)	How long he has served for	5
Veteran	Job Title	VARCHAR(255)	What he did	Jet mechanic

Account	Account type	VARCHAR(15)	Indicates user or corporate/school account	Corporation
Account	Email	VARCHAR(255)	Serves as both login name and contact information	CorporateCorperation@corporation.corp
Account	Password	VARCHAR(255)	Password to login to account in combination with email	asfvrejt34i85rj2wdsfasfdj42i3jwqaJSDW89U4TH2IJH245R
Corporate Account	Organization	VARCHAR(255)	Type of organization	School
Application	Application	VARCHAR(1024)	Application or resume sent to job	Army mechanic 5 years work experience BA of science from Berkeley
Field	Field	VARCHAR(255)	Describes what kind of field the related job is in	Science
Job	Company Name	VARCHAR (255)	Name of hiring company	Raytheon
Job	Job title	VARCHAR(255)	What the job is	Database manager
Job	Job requirements	VARCHAR(1024)	What the company is looking for	3 years experience in SQL
Job	Job salary	DECIMAL(10,2)	How much the job will pay	60,000.00
City Location	City Name	VARCHAR(255)	What city the job is in	Boston
Countryside Information	Countryside Name	VARCHAR(255)	Name of the countryside where the job is	Elk Creek
Countryside Information	Closest population center	VARCHAR(255)	The closest population center	Cortlandt

Location	State	VARCHAR (255)	Which state the job is in	Nebraska
School	School Name	VARCHAR (255)	Name of school	Columbia
External Link	Link URL	VARCHAR(255)	Provides a school link for more information	www.school.com/information
Requirements	Requirements	VARCHAR(255)	Different types of documents a school application may require	Letters of recommendation
Full-Time Tuition	Cost	DECIMAL (12,2)	How much money tuition costs	0
Part-Time tuition	CreditCap	DECIMAL (3)	Max amount of credits per semester to be considered part time	8
Part-Time tuition	Cost per credit	DECIMAL(10,2)	Cost per credit for part time students	1,200

After evaluating what I consider to be necessary attributes and indicators to the design, they were incorporated into the physical ERDs to help visualize how the database should be made. There are some tables that currently do not have attributes, but may have some in the future after this database is created, tested, and optimized. The primary and foreign keys were retained from previous iterations. After evaluating each table there should be no redundancies. Two tables I want to explain are the Account and Veteran table, normally the user attributes would be input into the account table, but the nature of using a code to verify each user I changed it around a little. The veteranID serves as a primary key that holds the user information, it is linked to the VA code, so when each unique VA code is created the veteran information becomes attached to the VA code, and when the code is used to create an account, the user information is retrieved and added into the account automatically and thus can't be forged. If I were to add these attributes again to the account table I feel that it would be redundant. I believe this ERD currently serves as a strong basis for creating the actual database. After the database is created and tested we can further refine the allocation of these attributes and indicators. One last important thing I want to note is that many VARCHAR datatypes are given a value of 255, while some fields obviously don't need that many. I feel it best just to have a high ceiling for whatever unforeseen circumstances could pop up.



## Stored Procedure Execution and Explanations

he first procedure would be that of Use case 1 which is as follows

The first use case would be when a veteran nears the end of his service and hasn't re-enlisted. He will receive a notice or e-mail from the VA for this app along with his personal code when he registers to validate his profile.

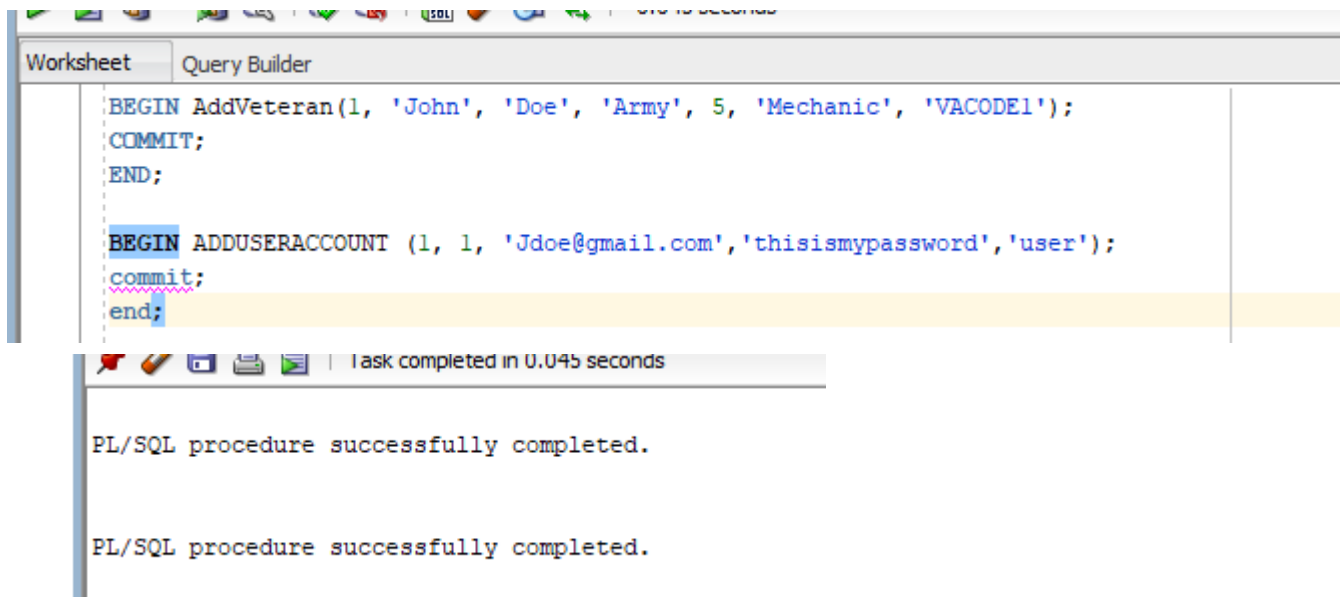
1. Veteran receives an email explaining this app
2. This person goes to the website to register an account
3. User registers an account with the code provided by the VA

First there are two procedures used for this which are the AddVeteran and AddUserAccount procedures they can be seen below

```
CREATE OR REPLACE PROCEDURE AddVeteran (  
    VeteranID IN DECIMAL,  
    FirstName IN VARCHAR,  
    LastName IN VARCHAR,  
    BranchService IN VARCHAR,  
    YearService IN DECIMAL,  
    JobTitle IN VARCHAR,  
    VACODE IN VARCHAR)  
AS  
BEGIN  
    INSERT INTO Veteran(VeteranID, FirstName, LastName, BranchService, YearService, JobTitle, VAcodem)  
VALUES (VeteranID, FirstName, LastName, BranchService, YearService, JobTitle, VAcodem);  
END;
```

```
CREATE OR REPLACE PROCEDURE AddUserAccount (  
    AccountID in DECIMAL,  
    VeteranID IN DECIMAL,  
    AccountEmail IN VARCHAR,  
    AccountPassword IN VARCHAR,  
    AccountType IN VARCHAR)  
AS  
BEGIN  
    INSERT INTO Account(AccountID, VeteranID, AccountEmail, AccountPassword, AccountType)  
VALUES (AccountID, VeteranID, AccountEmail, AccountPassword, AccountType);  
    INSERT INTO User_Account (AccountID)  
Values (AccountID);  
END;
```

After creating the procedure we use them to add information as per the use case.



```

BEGIN AddVeteran(1, 'John', 'Doe', 'Army', 5, 'Mechanic', 'VACODE1');
COMMIT;
END;

BEGIN ADDUSERACCOUNT (1, 1, 'Jdoe@gmail.com','thisismypassword','user');
commit;
end;

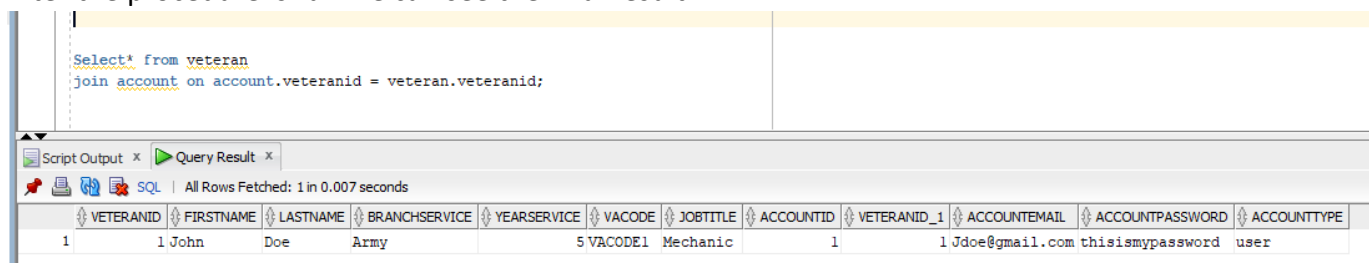
```

Task completed in 0.045 seconds

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

After the procedure is run we can see the final result



```

Select* from veteran
join account on account.veteranid = veteran.veteranid;

```

Script Output x Query Result x

All Rows Fetched: 1 in 0.007 seconds

VETERANID	FIRSTNAME	LASTNAME	BRANCHSERVICE	YEARSERVICE	VACODE	JOBTITLE	ACCOUNTID	VETERANID_1	ACCOUNTEMAIL	ACCOUNTPASSWORD	ACCOUNTTYPE
1	John	Doe	Army	5	VACODE1	Mechanic	1	1	Jdoe@gmail.com	thisismypassword	user

Another example would be the third use case, where companies and corporations place job ads onto the database.

1. Company HR logs into the app using a business side account
2. HR places a job ad with the necessary information
3. Job is inserted into the database

The first thing that happens is the corporation would create an account, because of the way the database is set up we have to use fake veteranID values to create the account

Worksheet Query Builder

```

BEGIN AddVeteran(999999999, 'Corporate', 'Corporate', 'Corporate', 9, 'Corporate', 'Corporate');
COMMIT;
END;

BEGIN ADDUSERACCOUNT (999999999, 999999999, 'corporate@corporate.com', 'corporatepassword', 'Corporate');
commit;
end;

BEGIN ADDCORPORATEDETAILS (999999999, 'SpaceX');
COMMIT;
END;

Select! from veteran
FULL join account on account.veteranid = veteran.veteranid
join corporate_account on corporate_account.accountid = account.accountid;

```

Script Output x Query Result x

All Rows Fetched: 1 in 0.007 seconds

VETERANID	FIRSTNAME	LASTNAME	BRANCHSERVICE	YEARSERVICE	VACODE	JOBTITLE	ACCOUNTID	VETERANID_1	ACCOUNTEMAIL	ACCOUNTPASSWORD	ACCOUNTTYPE	ACCOUNTID_1	ORGANIZATION
1	999999999	Corporate	Corporate	Corporate	9	Corporate	Corporate	999999999	999999999 corporate@corporate.com	corporatepassword	Corporate	999999999	SpaceX

The first two procedures can be seen in the first part of this problem, the last one is

```

CREATE OR REPLACE PROCEDURE AddCorporateDetails (
    AccountID IN DECIMAL,
    Organization IN VARCHAR)
AS
BEGIN
    INSERT INTO Corporate_Account (AccountID, Organization)
    VALUES (AccountID, Organization);
END;

```

After the account is created we can use it to list a job as seen below

```

BEGIN ADDJOB (1, 'SpaceX', 'Court Jester', '3 YEARS EXPERIENCE', '60000');
COMMIT;
END;

BEGIN ADDFIELD(1, 'Engineering');
COMMIT;
END;

BEGIN ADDLOCATION(1, 'New York');

SELECT * FROM JOB
LEFT JOIN JOB_LISTING ON JOB_LISTING.JobID = Job.JobID
LEFT JOIN FIELD ON FIELD.FIELDID = JOB.JobID
LEFT JOIN LOCATION ON LOCATION.LOCATIONID = JOB.JobID;

```

Script Output x Query Result x Query Result 1 x Query Result 2 x

All Rows Fetched: 1 in 0.009 seconds

JOBID	COMPANYNAME	JOBTITLE	JOBREQ	JOBsalary	JOBID_1	FIELDID	LOCATIONID	ACCOUNTID	APPLICATIONID	FIELDID_1	FIELD	LOCATIONID_1	STATE
1	1 SpaceX	Court Jester	3 YEARS EXPERIENCE	60000	(null)	(null)	(null)	(null)	(null)	(null)	1 Engineering	1 New York	

The procedures I used to generate the job is as follows



```

CREATE OR REPLACE PROCEDURE AddJob(
    JobID IN DECIMAL,
    CompanyName IN VARCHAR,
    JobTitle IN VARCHAR,
    JobReq IN VARCHAR,
    JobSalary IN DECIMAL)
AS
BEGIN
    INSERT INTO Job(JobID, CompanyName, JobTitle, JobReq, JobSalary)
    VALUES(JobID, CompanyName, JobTitle, JobReq, JobSalary);
END;

```

```

CREATE OR REPLACE PROCEDURE AddLocation (
    LocationID IN DECIMAL,
    State IN VARCHAR)
AS
BEGIN
    INSERT INTO Location(LocationID, State)
    VALUES(LocationID, State);
END;

```

```

CREATE OR REPLACE PROCEDURE AddField (
    FieldID IN DECIMAL,
    Field IN VARCHAR)
AS
BEGIN
    INSERT INTO Field(FieldID, Field)
    VALUES(FieldID, Field);
END;

```

## Question Identification and Explanations

The first question is: How many jobs currently in the database offer an annual salary of more than \$50,000, and for those jobs that pay more, show them as well while going up in 50,000 increments up to 200,000.

The importance of this question is that when people use the database to find jobs they will want to look for those that have higher salaries, as the database grows these numbers become more important because they show what the average job and its salary posted are.

The second question is: How educated are the people applying for jobs in the database? To show this we want to see how many people hold degrees, and whether or not they hold multiple degrees.

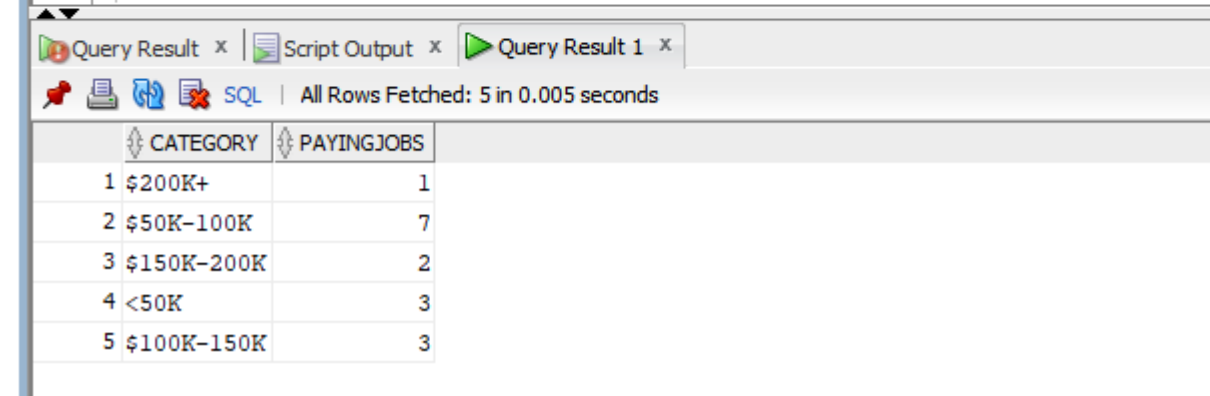
This is an important question because it shows how many people can be considered extremely competitive or qualified for different jobs in the database.

The last question is: I want to see a table with the highest paying jobs, or jobs that exceed \$50,000 annually, I also want all the information pertaining to that job such as what field its in and where its located, Then finally I want to be able to narrow it down to only the New York Area.

A lot of people want the best jobs, and preferably they don't have to move to get such a job, for a job searching database this is considered really important to applicants.

## Query Executions and Explanations

```
SELECT (CASE
WHEN JOB.JOBSALARY <50000 AND JOB.JOBSALARY >0 THEN '<50K'
WHEN JOB.JOBSALARY >=50000 AND JOB.jobSALARY < 100000 THEN '$50K-100K'
WHEN JOB.JOBSALARY >=100000 AND JOB.JOBSALARY <150000 THEN '$100K-150K'
WHEN JOB.JOBSALARY >=150000 AND JOB.JOBSALARY <200000 THEN '$150K-200K'
ELSE '$200K+'
END) AS CATEGORY,|
COUNT (*) AS PayingJobs
FROM JOB_LISTING
JOIN Job on job.jobid = job_listing.jobid
JOIN Field on field.fieldid = job_listing.fieldid
join location on location.locationid = job_listing.locationid
GROUP BY (case
WHEN JOB.JOBSALARY <50000 AND JOB.JOBSALARY >0 THEN '<50K'
WHEN JOB.JOBSALARY >=50000 AND JOB.jobSALARY < 100000 THEN '$50K-100K'
WHEN JOB.JOBSALARY >=100000 AND JOB.JOBSALARY <150000 THEN '$100K-150K'
WHEN JOB.JOBSALARY >=150000 AND JOB.JOBSALARY <200000 THEN '$150K-200K'
ELSE '$200K+'
END)
```



The screenshot shows a database query execution interface. At the top, the SQL query is displayed. Below the query, there are tabs for 'Query Result', 'Script Output', and 'Query Result 1'. The 'Query Result' tab is active, showing a table with two columns: 'CATEGORY' and 'PAYINGJOBS'. The table contains five rows of data, representing different salary ranges and their corresponding counts.

CATEGORY	PAYINGJOBS
1 \$200K+	1
2 \$50K-100K	7
3 \$150K-200K	2
4 <50K	3
5 \$100K-150K	3

The answer to the first question creates a query where it categorizes the different paying jobs. We can verify it when we view the entire database.

<pre> SELECT companyname, jobtitle, jobreq, jobsalary, field, AREA FROM JOB_LISTING JOIN Job on job.jobid = job_listing.jobid JOIN Field on field.fieldid = job_listing.fieldid join location on location.locationid = job_listing.locationid; </pre>						
Query Result x   Script Output x   Query Result 1 x						
SQL   All Rows Fetched: 16 in 0.002 seconds						
COMPANYNAME	JOBTITLE	JOBREQ	JOBSALARY	FIELD	AREA	
6 Joes	Waiter	2 YEARS EXPERIENCE	20000	Service	Syracuse	
7 SpaceX	Court Jester	5 Years Experience	80000	Enginerring	Buffalo	
8 SpaceX	Court Jester	5 Years Experience	80000	Enginerring	Jamaica	
9 SpaceX	Guinea Pig	1 YEARS EXPERIENCE	50000	Science	Jamaica	
10 SpaceX	Guinea Pig	1 YEARS EXPERIENCE	50000	Science	Jamaica	
11 SpaceX	Head of marketing	15 YEARS EXPERIENCE	600000	Marketing	Jamaica	
12 CallCenter	Scam Caller	1 YEARS EXPERIENCE	100000	Marketing	Jamaica	
13 SpaceX	Court Jester	5 Years Experience	80000	Enginerring	Boston	
14 CallCenter	Scam Caller	1 YEARS EXPERIENCE	100000	Marketing	Boston	
15 Raytheon	Rocket scientist	7 YEARS EXPERIENCE	160000	Enginerring	Cordtland	
16 CallCenter	Scam Caller	1 YEARS EXPERIENCE	100000	Marketing	Cordtland	

As seen from the second screenshot all the numbers in the category do match with those presented.

For the second query all we have to do is combine the information from the degree entity and its subclasses, this will create a list with each person and their degrees, so the people who use the database can determine who are the most competitive or qualified people currently seeking employment

<pre> SELECT Awardedto,MADEGREE,BSDEGREE,CREDITHOURS FROM DEGREE left join masters_degree on masters_degree.degreeid = degree.degreeid left join bachlors_degree on bachlors_degree.degreeid = degree.degreeid ORDER by Awardedto ASC; </pre>			
Script Output x   Query Result x			
SQL   All Rows Fetched: 8 in 0.001 seconds			
AWARDEDTO	MADEGREE	BSDEGREE	CREDITHOURS
1 Jacob Sight	(null)	Bachlors of Biology	130
2 Jacob Sight	(null)	Bachlors of Linguistics	128
3 Jacob Sight	Masters of Biology	(null)	(null)
4 John Doe	Masters of Computer Science	(null)	(null)
5 John Doe	(null)	Bachlors of computer science	128
6 Mary Dill	Masters of Engineering	(null)	(null)
7 Mary Dill	(null)	Bachlors of Engineering	140
8 Smith Jones	(null)	Bachlors of English	128

In order to answer the last question first we created a view that combines information on all the jobs that pay more than \$50,000 annually. Then using that view and a search query we can narrow it down to specific locations.

```

CREATE OR REPLACE VIEW High_view AS
    SELECT job.jobid, job.companyname, job.jobtitle, job.jobreq, job.jobsalary, FIELD.FIELD, location.state
    FROM JOB
    LEFT JOIN JOB_LISTING ON JOB_LISTING.JobID = Job.JobID
    LEFT JOIN FIELD ON FIELD.FIELDID = JOB.JOBID
    LEFT JOIN LOCATION ON LOCATION.LOCATIONID = JOB.JOBID
    WHERE job.jobsalary >= 50000;

select High_view.companyname,
       High_view.jobtitle,
       High_view.jobreq,
       High_view.jobsalary,
       high_view.field,
       high_view.state
from High_view
where ( state = 'New York');

```

Query Result x

SQL | All Rows Fetched: 8 in 0.074 seconds

	COMPANYNAME	JOBTITLE	JOBREQ	JOBSALARY	FIELD	STATE
1	SpaceX	Court Jester	5 Years Experience	80000	Enginerring	New York
2	SpaceX	Court Jester	5 Years Experience	80000	Enginerring	New York
3	SpaceX	Court Jester	5 Years Experience	80000	Enginerring	New York
4	SpaceX	Court Jester	5 Years Experience	80000	Enginerring	New York
5	SpaceX	Court Jester	5 Years Experience	80000	Enginerring	New York
6	SpaceX	Head of marketing	15 YEARS EXPERIENCE	600000	Marketing	New York
7	Raytheon	Rocket scientist	7 YEARS EXPERIENCE	160000	Enginerring	New York
8	Raytheon	Rocket scientist	7 YEARS EXPERIENCE	160000	Enginerring	New York

## Index Identification and Creations

Firstly there are the primary keys which are already indexed, they are as follows:

Veteran.VeteranID

Degree.DegreeID

Account.AccountID

Application.ApplicationID

Field.FieldID

Job.JobID

Location.LocationID

Job\_listing.JobID

School.SchoolID

External\_Link.ExternalID

Requirements.RequirementID

Tuition.TuitionID

Column	Uniqueness	Description
Degree.VeteranID	No	The foreign Key in Degree is not unique because there can be multiple people with the same degree
Account.VeteranID	Yes	This Foreign Key is unique because there can only be one person associated with each account
Application.AccountID	No	Each account can have multiple applications
Job_listing.JobID	Yes	This foreign key can be unique because there is only one job related to each job ID
Job_listing.FieldID	No	This foreign key can't be unique because a single field can have multiple jobs
Job_listing.LocationID	No	This foreign key can't be unique because a location can have multiple jobs

Job_listing.AccountID	Yes	This foreign key can be unique because each job listing is listed by a single account
Job_listing.ApplicationID	No	This foreign key can't be unique because there can be multiple applications to a single job
School.AccountID	Yes	This foreign key can be unique because each school only has one account and vice versa
External_Link.SchoolID	No	This foreign key can't be unique because there are multiple external links to a single school
Requirements.SchoolID	No	This foreign key can't be unique because each school has multiple requirements
Tuition.SchoolID	No	This foreign key can't be unique because each school has different tuitions
Job.JobSalary	No	This can't be unique because there can be multiple jobs with the same salary
Job.CompanyName	No	This can't be unique because a company can have multiple job listings
Veteran.BranchService	No	This can't be unique because each branch can have multiple people

In the grand scheme of indexes those that were determined unique in the previous table were made into unique indexes. I created an Index for all my foreign keys and named them accordingly. The last 3 query driven indexes JobSalaryIdx, JobCompanyNameIdx, BranchServiceIdx are created to help future queries because they are very specific to search queries, and can be considered common search options. When people use a job database it is common that they will want to find jobs from certain companies or pay certain amounts, which explains why those 2 were indexed. From the perspective of the employer a lot of people also like to search up what branch of service veterans are from so they can focus on what kind of skills these potential applicants have developed.

```

CREATE INDEX VeteranIDIdx
ON Degree(VeteranID);

CREATE UNIQUE INDEX AccVeteranIDIdx
ON Account(VeteranID);

CREATE INDEX AppAccountIDIdx
ON Application(AccountID);

CREATE UNIQUE INDEX JobListingJobIDIdx
ON Job_Listing(JobID);

CREATE INDEX JobListingFieldIDIdx
ON Job_Listing(FieldID);

CREATE INDEX JobListingLocationIDIdx
ON Job_Listing(LocationID);

CREATE UNIQUE INDEX JobListingAccountIDIdx
ON Job_Listing(AccountID);

CREATE INDEX JobListingApplicationIDIdx
ON Job_Listing(ApplicationID);

CREATE UNIQUE INDEX SchoolAccountIDIdx
ON School(AccountID);

CREATE INDEX ExternalSchoolIDIdx
ON Eternal_Link(SchoolID);

CREATE INDEX ReqSchoolIDIdx
ON Requirements(SchoolID);

CREATE INDEX TuitionSchoolIDIdx
ON Tuition(SchoolID);

CREATE INDEX JobSalaryIdx
ON Job(JobSalary);

CREATE INDEX JobCompanyNameIdx
ON Job(CompanyName);

CREATE INDEX BranchServiceIdx
ON Veteran(BranchService);

```

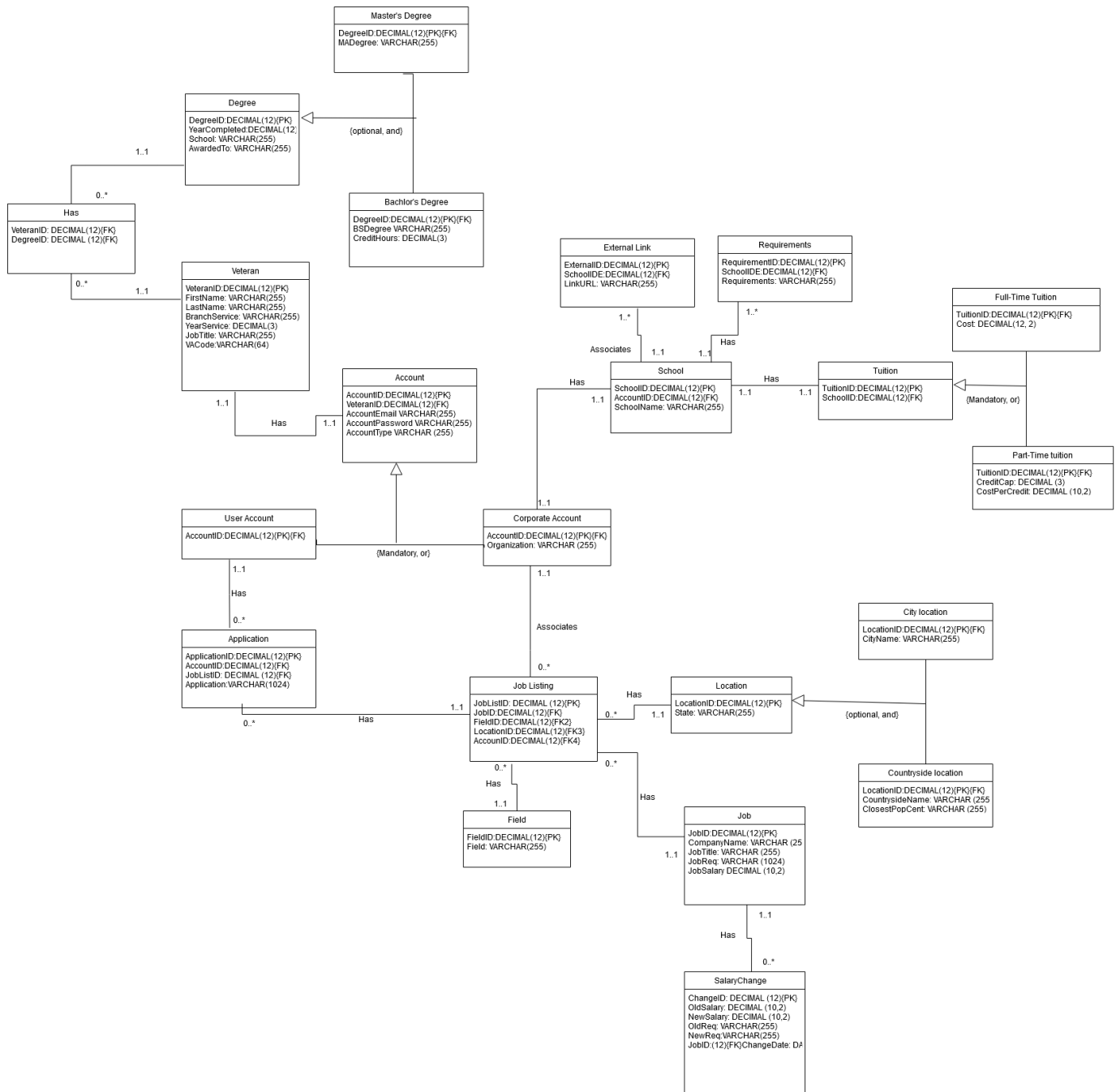
## History Table Demonstration

Explain the specifics of your history table, including how the trigger works, and demonstrate that the history table captures changes.

One place where I determined a history table would be beneficial is for the jobsalary and jobreq attribute in the jobs table. This is important because it can show the trend and popularity of certain jobs, if certain jobs become extremely popular it is very possible for companies to reasonably lower potential salary, and the opposite is also true. If a company is in desperate need of workers it can increase the potential pay, this change in salary can show trends in the job market, such as what kind of work is extremely popular, and what kind of work is very undesirable, this data can also be reinforced with the changes in a job requirement as well, if there is a lowered requirement and increased pay it means that the company may be desperate to fill the position(We had a joke when I was in the army, where if you got a really high sign on bonus it was because the job was awful no one wanted to do it, because everyone in our MOS got a high sign on bonus) The attributes and modified ERD below shows the added table

Attribute	Description	Example
ChangeID	Primary key to keep track of changes	1
OldSalary	Previous or initial salary	60000
NewSalary	New salary after changes are applied	80000
OldReq	Previous or initial requirement	3 years experience
NewReq	New job requirements after changes are applied	2 years experience
JobID	A foreign key reference to which job has changed	1
ChangeDate	The date the change was applied	9/9/19





The following is my trigger, it works in a way where if there are any changes to the salary or requirements of a job it will record that information in the salary change table.

```

CREATE OR REPLACE TRIGGER JobChangeTrig
BEFORE UPDATE OF JOBSALARY, JobReq ON JOB
FOR EACH ROW
BEGIN
    INSERT INTO SalaryChange(ChangeID, OldSalary, NewSalary, Oldreq, NewReq, JobID, ChangeDate)
    VALUES(SalChange_seq.nextval, :OLD.JobSalary, :NEW.JobSalary, :OLD.Jobreq, :NEW.Jobreq, :NEW.JobID, trunc(sysdate));
END;
  
```

```

Update JOB
Set JobSalary = 80000
WHERE JobID =1;

Update JOB
Set jobreq = '5 Years Experience'
WHERE JobID = 1;

select * from salarychange;

```

Query Result x | Query Result 1 x | Script Output x | Query Result 2 x | Query Result 3 x | Query Result 4 x

SQL | All Rows Fetched: 2 in 0.004 seconds

	CHANGEID	OLDSALARY	NEWSALARY	OLDREQ	NEWREQ	JOBID	CHANGEDATE
1	1	60000	80000	3 Years Experience	3 Years Experience	1	27-FEB-23
2	2	80000	80000	3 Years Experience	5 Years Experience	1	27-FEB-23

As seen from the above image, the first change recorded is a change in salary from 60,000 to 80,000. The second change seen is the job requirements growing from 3 years to 5 years of experience. Furthermore if we joined the job table then we can see all the information relevant to the job

```

select oldsalary, newsalary, oldreq, newreq, companyname, jobtitle, changedate from salarychange
JOIN job on job.jobid=salarychange.jobid;

```

Query Result x | Query Result 1 x | Script Output x | Query Result 2 x | Query Result 3 x | Query Result 4 x | Query Result 5 x

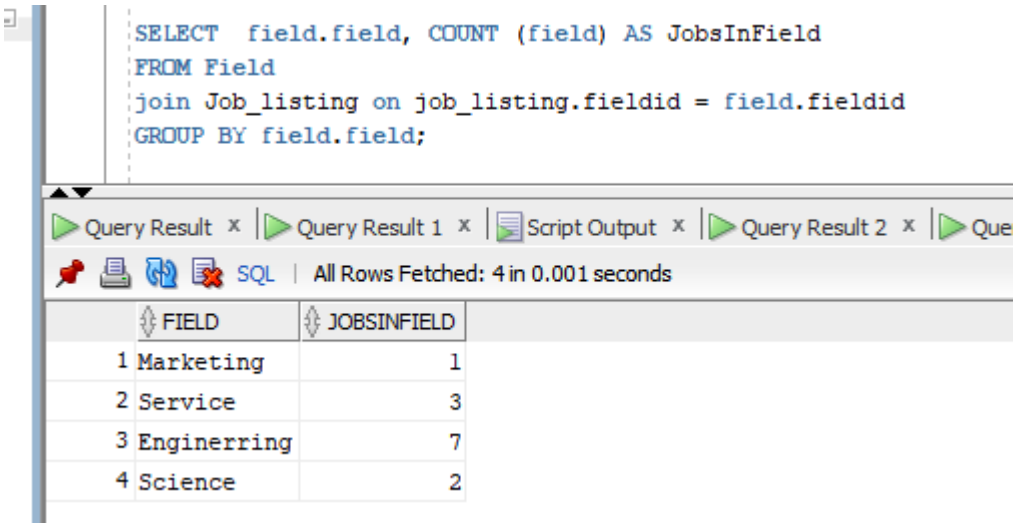
SQL | All Rows Fetched: 2 in 0.004 seconds

	OLDSALARY	NEWSALARY	OLDREQ	NEWREQ	COMPANYNAME	JOBTITLE	CHANGEDATE
1	60000	80000	3 Years Experience	3 Years Experience	SpaceX	Court Jester	27-FEB-23
2	80000	80000	3 Years Experience	5 Years Experience	SpaceX	Court Jester	27-FEB-23

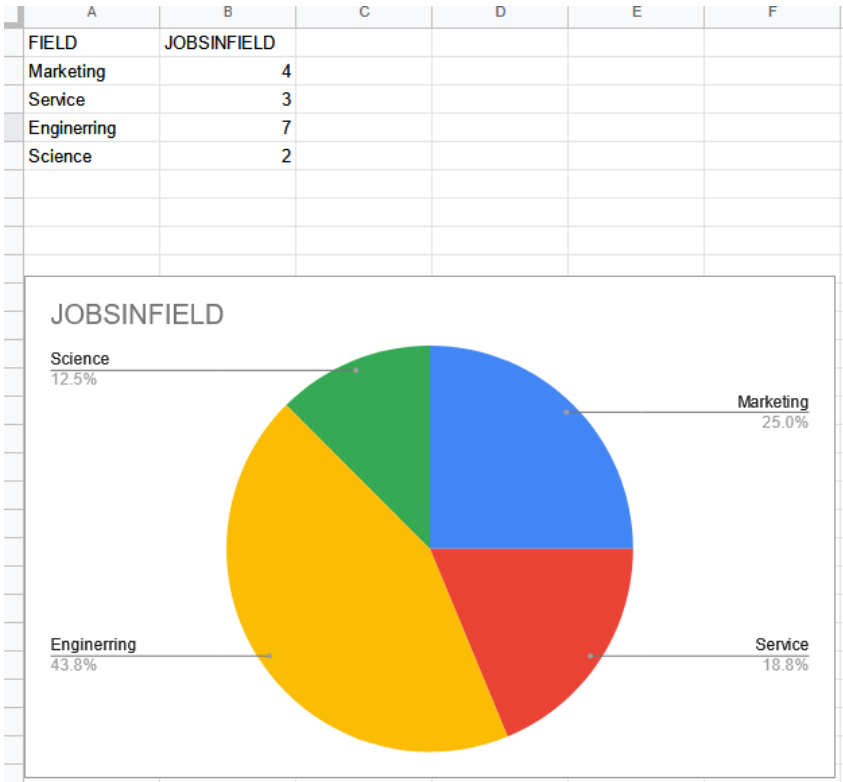
## Data Visualizations

Include and explain data visualizations and stories that tell effective stories about the information in a way that people quickly and accurately understand it.

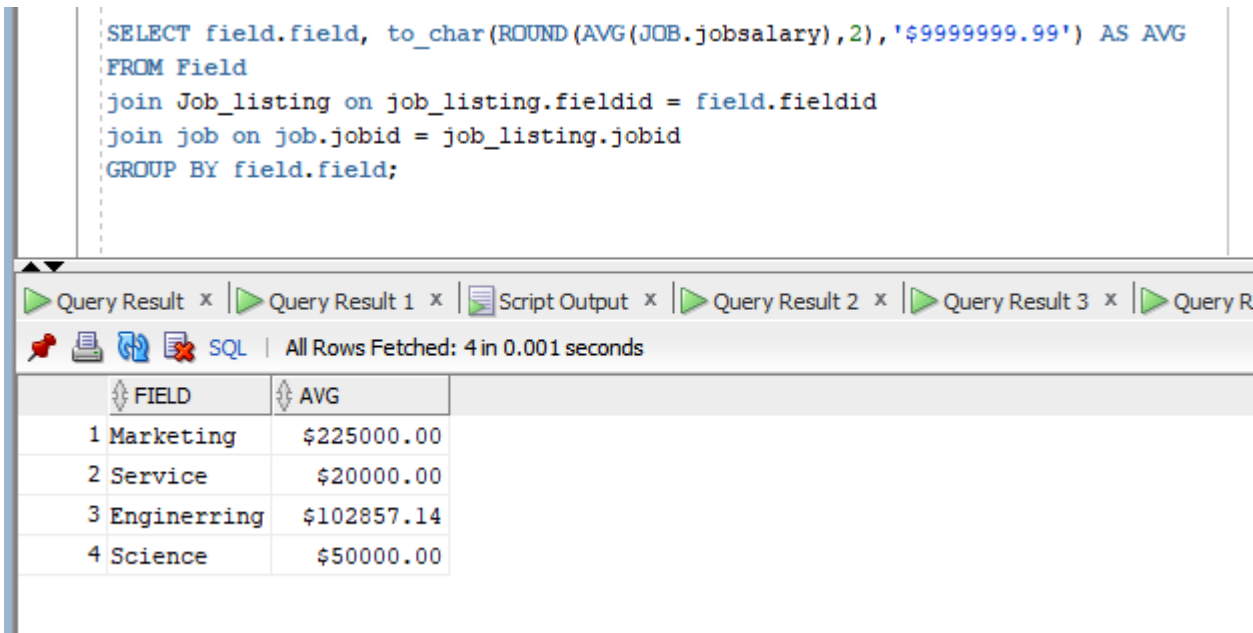
For the first data visualization i used the following, which shows which field and how many jobs are available in that field



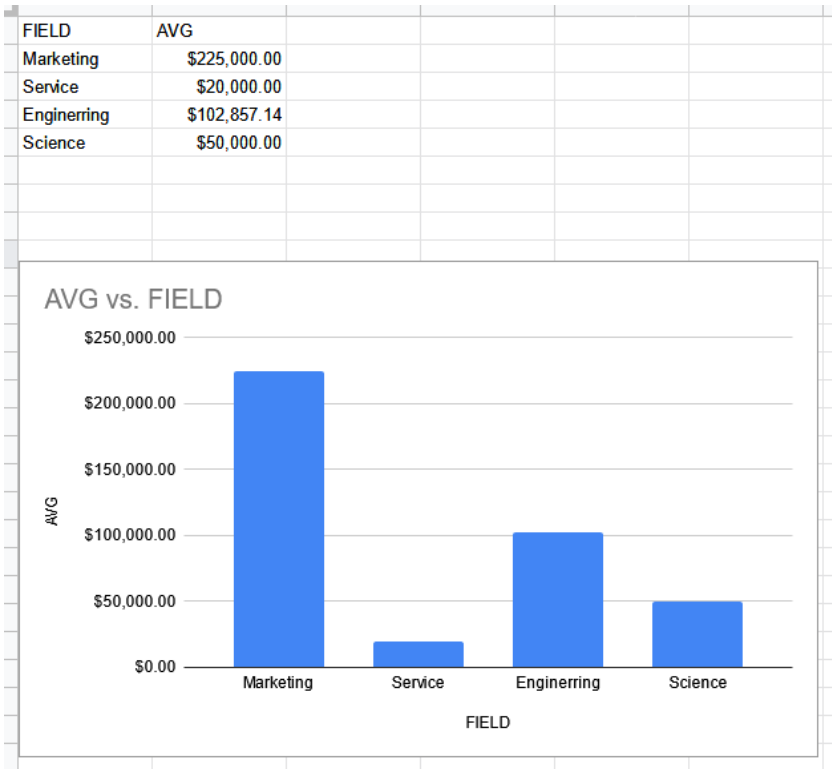
A whole lot of information can be derived from this data. Using this data we can make assumptions on which job field is growing and which is shrinking, or which field is more competitive and which is easier to break into because it is seemingly less competitive. Using the results of the query we can plot a circle graph to show market share of jobs. As we can see from the pie chart, currently with all the jobs on the job market in our database a whopping 43.8% consists of jobs related to the engineering field. This could mean that the engineering field is growing or that it could be in desperate need of more workers.



For the second point of data visualization I chose something that compliments the first half. We use the following which shows us what the average salary is in each of the fields presented.



We converted this information into a bar graph as seen below, this gives us more insight into the current job market



This graph shows what the average salary is in each of the different job fields, from this we can tell that finding a job in marketing or engineering can bring about lucrative salaries, while the service industry seems to have very low average salaries, and with jobs in the science field having an average salary. The large quantity of jobs in engineering, and marketing along with their high salaries can show that these two industries are desperately seeking people to fulfill positions and offer lucrative incentives. This may also point to a growing industry within these two fields.

## Summary and Reflection

I thoroughly enjoyed the process of coming up with a database idea, and going through the different processes to finally create it. I found that iterations 5 and 6 were extremely challenging but at the same time very enlightening. It put everything I learned from the labs into use, and served as a great source of practice. I feel that I definitely require more practice with the coding language because I find myself often using old labs as a crutch. I want to be able to do whatever processes I want in a database without having to continuously look back on old labs as a reference.