Software Testing Database

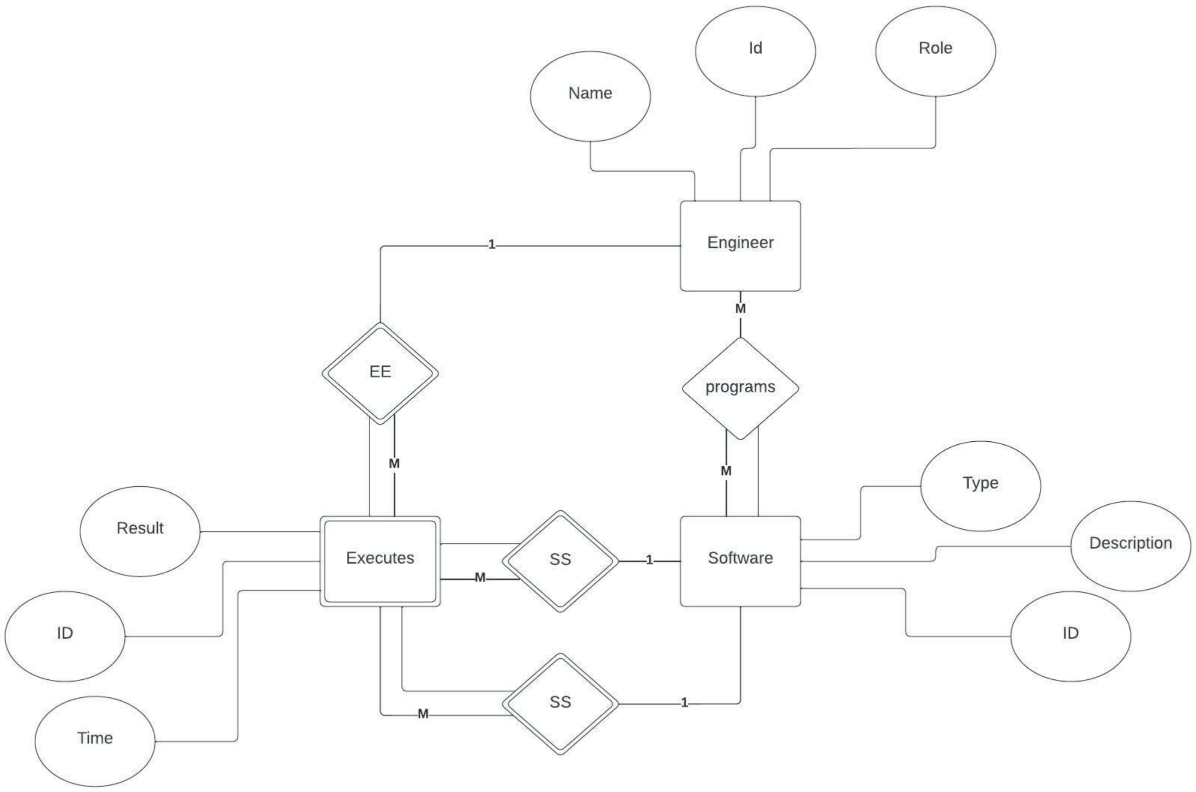
Cody Strange, Lincoln Harmston

CS3520-002
Fall 2022
Project 1
Oracle

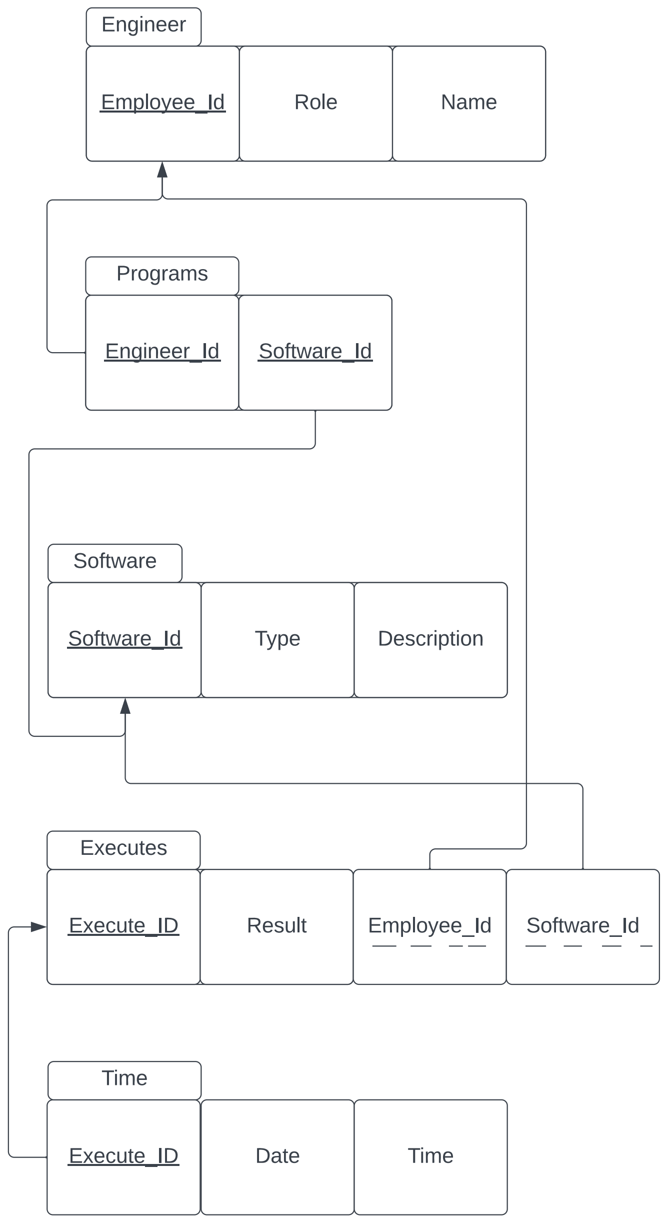
Interview

We sat down with one of our classmates (never got his name) who works in the field of testing. He was able to describe the process of to us that we modeled our E-R diagram after. He went over the process of Engineers that would create a feature to add to the current project and then another Engineer that would write and run a test on said feature. There could be multiple tests run on one feature and you would want to write the tests so you could reuse them on different features if possible. He recommended to us to have our Engineer and Software as one entity and just have an attribute saying which role or type they were. He really helped us simplify our E-R diagram and narrow the scope of it to this project.

ER Diagram

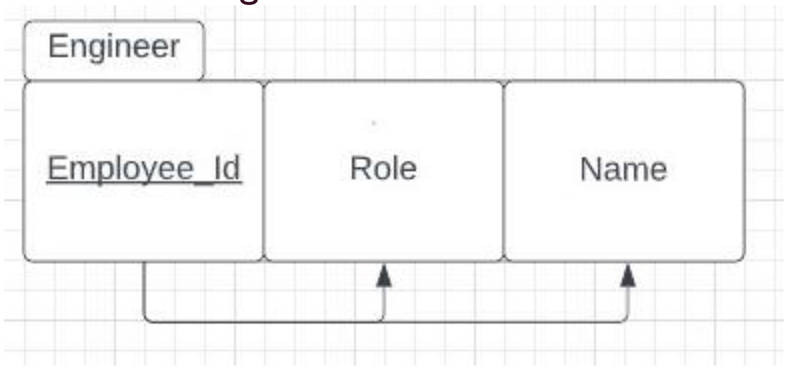

Name
Id
Role
Engineer
1
M
EE
programs
M
M
Type
Result
Executes
SS
1
Software
Description
M
ID
ID
SS
M
1
Time

Schemas



Normalization

Table one: Engineer



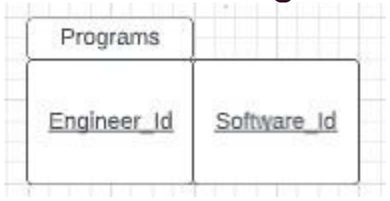
First Normal Form: The Engineer table is in 1NF because its attributes (Employee_Id, Role, and Name) are not multi-value attributes, therefore the table is in 1NF.

Second Normal Form: The Engineer table is in 2NF because it is 1NF and both non-key attributes (Role, Name) are fully functionally dependent on the primary key.

Third Normal Form: The Engineer table is in 3NF because it is in 2NF and it has no transitive dependencies, we know it cannot have any transitive dependencies because all non-key attributes are solely dependent on the single key attribute.

Boyce-Codd Normal Form: The Engineer table is in BCNF because it is in 3NF, and no key attributes are dependent on a non-key attribute.

Table Two: Programs



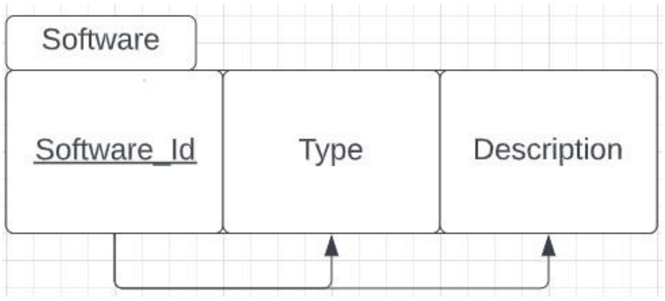
First Normal Form: The Programs table is in 1NF because its attributes (Engineer_Id, and Software_Id) are not multi-value attributes, therefore the table is in 1NF.

Second Normal Form: The Programs table is in 2NF because it is 1NF and all of its attributes (Engineer_Id, and Software_Id) are primary keys, therefore all non-key attributes are fully functionally dependent on the primary key.

Third Normal Form: The Programs table is in 3NF because it is in 2NF and it has no transitive dependencies, we know it cannot have any transitive dependencies because it only has primary key attributes, therefore all non-key attributes are solely dependent on the single key attribute.

Boyce-Codd Normal Form: The Programs table is in BCNF because it is in 3NF and it only has primary keys, therefore no key attributes are dependent on non-key attributes.

Table Three: Software



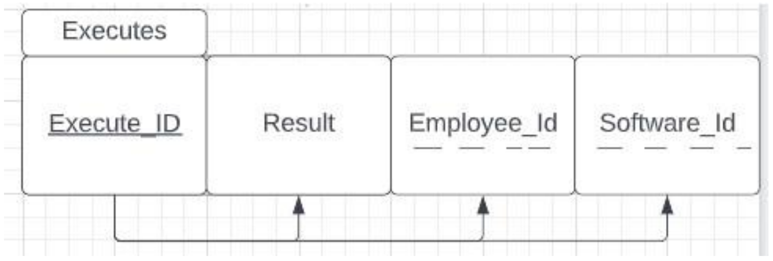
First Normal Form: The Software table is in 1NF because its attributes (Software_Id, and Type, Description) are not multi-value attributes, therefore the table is in 1NF.

Second Normal Form: The Software table is in 2NF because it is 1NF and all of its attributes (Type and Description) are fully functionally dependent primary key.

Third Normal Form: The Software table is in 3NF because it is in 2NF and it has no transitive dependencies, we know it cannot have any transitive dependencies because all non-key attributes are solely dependent on the single key attribute.

Boyce-Codd Normal Form: The Software table is in BCNF because it is in 3NF and no key attributes are dependent on a non-key attribute.

Table Four: Executes



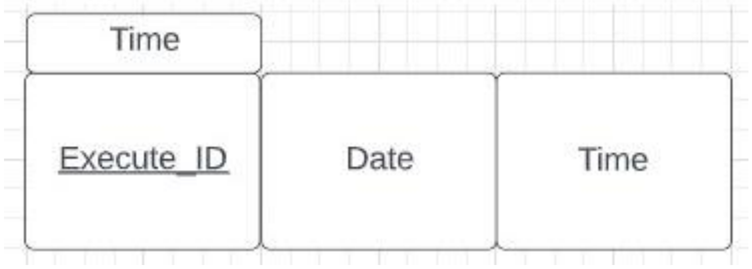
First Normal Form: The Executes table is in 1NF because its attributes (Employee_Id, Result, Execute_Id, and Software_Id) are not multi-value attributes, therefore the table is in 1NF.

Second Normal Form: The Executes table is in 2NF because it is 1NF and all non-key attributes (Result, Employee_Id, and Software_Id) are fully functionally dependent on the primary key.

Third Normal Form: The Executes table is in 3NF because it is in 2NF and it has no transitive dependencies, we know it cannot have any transitive dependencies because all non-key attributes are solely dependent on the single primary key attribute.

Boyce-Codd Normal Form: The Executes table is in BCNF because it is in 3NF, and no key attributes are dependent on a non-key attribute.

Table Five: Time



First Normal Form: The Executes table is in 1NF because its attributes (Execute_Id, Date, and Time) are not multi-value attributes, therefore the table is in 1NF.

Second Normal Form: The Executes table is in 2NF because it is 1NF and all non-key attributes (Date and Time) are fully functionally dependent on the primary key.

Third Normal Form: The Executes table is in 3NF because it is in 2NF and it has no transitive dependencies, we know it cannot have any transitive dependencies because all non-key attributes are solely dependent on the single primary key attribute.

Boyce-Codd Normal Form: The Executes table is in BCNF because it is in 3NF, and no key attributes are dependent on a non-key attribute.

Oracle Database

ENGINEER TABLE

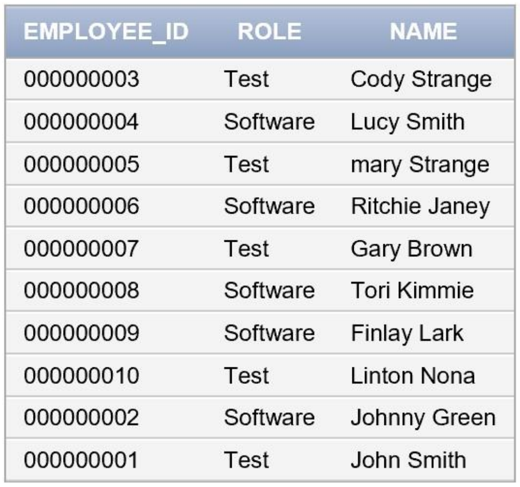
Table SQL

```
CREATE TABLE Engineer (
Employee_Id VARCHAR(9),
Role VARCHAR(16),
Name VARCHAR(32),
CONSTRAINT Engineer_empId_pk PRIMARY KEY(Employee_Id),
CONSTRAINT Engineer_Role CHECK(Role = 'Test' OR Role = 'Software'));
```

Tuple SQL

```
INSERT INTO Engineer
VALUES('000000001', 'Test', 'John Smith');
INSERT INTO Engineer (Employee_Id, Role, Name)
WITH names as (
SELECT '000000005', 'Test', 'mary Strange' FROM dual UNION ALL
SELECT '000000006', 'Software', 'Ritchie Janey' FROM dual UNION ALL
SELECT '000000007', 'Test', 'Gary Brown' FROM dual UNION ALL
SELECT '000000008', 'Software', 'Tori Kimmie' FROM dual UNION ALL
SELECT '000000009', 'Software', 'Finlay Lark' FROM dual UNION ALL
SELECT '000000010', 'Test', 'Linton Nona' FROM dual
)
SELECT * FROM names
INSERT INTO Engineer (Employee_Id, Role, Name)
WITH names as (
SELECT '000000003', 'Test', 'Cody Strange' FROM dual UNION ALL
SELECT '000000004', 'Software', 'Lucy Smith' FROM dual
)
SELECT * FROM names
INSERT INTO Engineer
VALUES('000000002', 'Software', 'Johnny Green');
```

Table Oracle



EMPLOYEE_ID	ROLE	NAME
000000003	Test	Cody Strange
000000004	Software	Lucy Smith
000000005	Test	mary Strange
000000006	Software	Ritchie Janey
000000007	Test	Gary Brown
000000008	Software	Tori Kimmie
000000009	Software	Finlay Lark
000000010	Test	Linton Nona
000000002	Software	Johnny Green
000000001	Test	John Smith

SOFTWARE TABLE

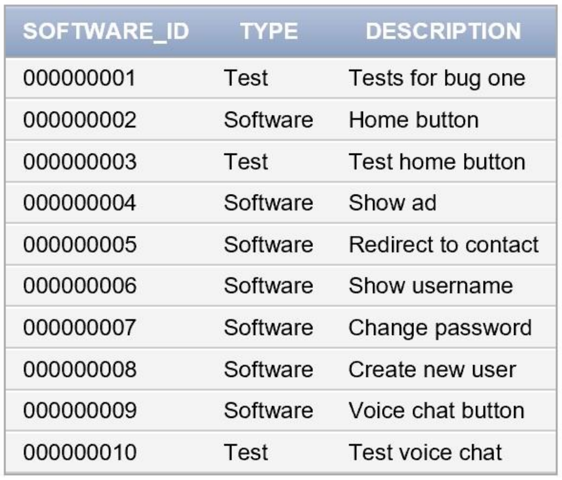
Table SQL

```
CREATE TABLE Software(
Software_Id VARCHAR(9),
Type VARCHAR(16),
Description VARCHAR(500),
CONSTRAINT Software_softId_pk PRIMARY KEY(Software_Id),
CONSTRAINT Software_Type CHECK(Type= 'Software' OR Type = 'Test'));
```

Tuple SQL

```
INSERT INTO SOFTWARE
WITH names as (
SELECT '000000001', 'Test', 'Tests for bug one'FROM dual UNION ALL
SELECT '000000002', 'Software', 'Home button' FROM dual UNION ALL
SELECT '000000003', 'Test', 'Test home button' FROM dual UNION ALL
SELECT '000000004', 'Software', 'Show ad' FROM dual UNION ALL
SELECT '000000005', 'Software', 'Redirect to contact' FROM dual UNION ALL
SELECT '000000006', 'Software', 'Show username' FROM dual UNION ALL
SELECT '000000007', 'Software', 'Change password' FROM dual UNION ALL
SELECT '000000008', 'Software', 'Create new user' FROM dual UNION ALL
SELECT '000000009', 'Software', 'Voice chat button' FROM dual UNION ALL
SELECT '000000010', 'Test', 'Test voice chat' FROM dual
)
SELECT * FROM names
```

Table Oracle



SOFTWARE_ID	TYPE	DESCRIPTION
000000001	Test	Tests for bug one
000000002	Software	Home button
000000003	Test	Test home button
000000004	Software	Show ad
000000005	Software	Redirect to contact
000000006	Software	Show username
000000007	Software	Change password
000000008	Software	Create new user
000000009	Software	Voice chat button
000000010	Test	Test voice chat

PROGRAMS TABLE

Table SQL

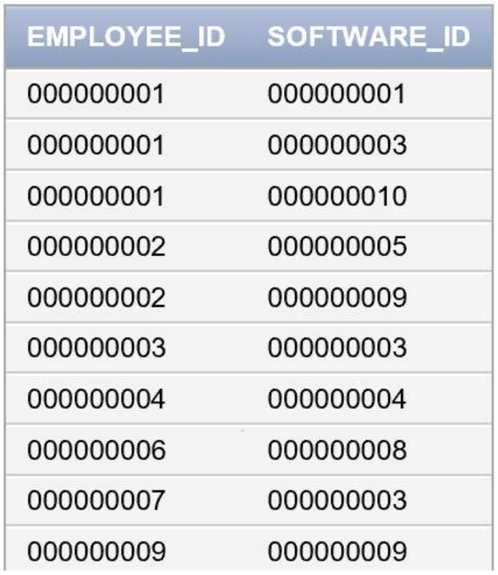
```
CREATE TABLE Programs (
Employee_Id VARCHAR(9),
Software_Id VARCHAR(9),
CONSTRAINT Programs_empSoftId_pk PRIMARY KEY(Employee_Id, Software_ID),
CONSTRAINT pefk FOREIGN KEY(Employee_Id) REFERENCES Engineer(Employee_Id),
CONSTRAINT psfk FOREIGN KEY(Software_Id) REFERENCES Software(Software_Id))
```

Tuple SQL

```
INSERT INTO Programs
VALUES('000000001', '000000003');
INSERT INTO Programs
VALUES('000000001', '000000001');
INSERT INTO Programs
VALUES('000000006', '000000008');
INSERT INTO Programs
VALUES('000000001', '000000010');
INSERT INTO Programs
VALUES('000000007', '000000003');
INSERT INTO Programs
VALUES('000000004', '000000004');
INSERT INTO Programs
VALUES('000000002', '000000009');
INSERT INTO Programs
WITH names (Employee_Id, Software_Id) as (
SELECT '000000009', '000000009' FROM dual UNION ALL
SELECT '000000002', '000000005' FROM dual UNION ALL
SELECT '000000003', '000000003' FROM dual

)
SELECT * FROM names
```

Table Oracle



EMPLOYEE_ID	SOFTWARE_ID
000000001	000000001
000000001	000000003
000000001	000000010
000000002	000000005
000000002	000000009
000000003	000000003
000000004	000000004
000000006	000000008
000000007	000000003
000000009	000000009

EXECUTES TABLE

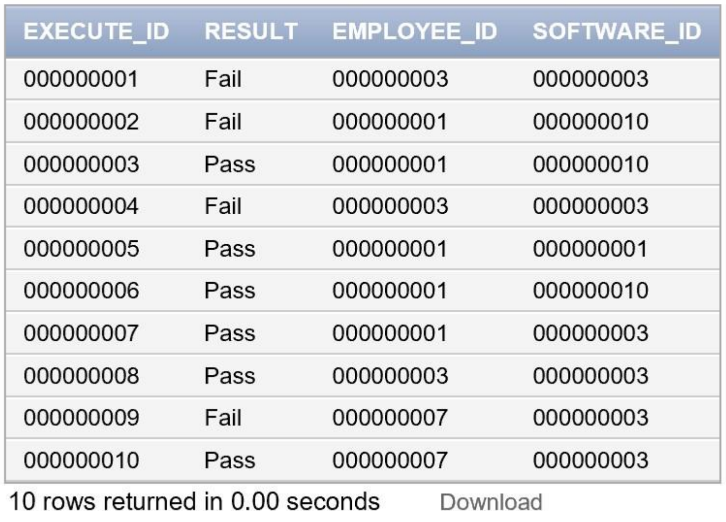
Table SQL

```
CREATE TABLE Executes(
Execute_Id VARCHAR(9),
Result VARCHAR(8),
Employee_Id VARCHAR(9),
Software_Id VARCHAR(9),
CONSTRAINT Executes_exId_pk PRIMARY KEY(Execute_Id),
CONSTRAINT eefk FOREIGN KEY (Employee_Id) REFERENCES Engineer(Employee_Id),
CONSTRAINT esfk FOREIGN KEY (Software_Id) REFERENCES Software(Software_Id));
```

Tuple SQL

```
INSERT INTO Executes
WITH names (Execute_Id, Result, Employee_Id, Software_Id) as (
SELECT '000000001', 'Fail', '000000003', '000000003' FROM dual UNION ALL
SELECT '000000002', 'Fail', '000000001', '000000010' FROM dual UNION ALL
SELECT '000000003', 'Pass', '000000001', '000000010' FROM dual UNION ALL
SELECT '000000004', 'Fail', '000000003', '000000003' FROM dual UNION ALL
SELECT '000000005', 'Pass', '000000001', '000000001' FROM dual UNION ALL
SELECT '000000006', 'Pass', '000000001', '000000010' FROM dual UNION ALL
SELECT '000000007', 'Pass', '000000001', '000000003' FROM dual UNION ALL
SELECT '000000008', 'Pass', '000000003', '000000003' FROM dual UNION ALL
SELECT '000000009', 'Fail', '000000007', '000000003' FROM dual UNION ALL
SELECT '000000010', 'Pass', '000000007', '000000003' FROM dual
)
SELECT * FROM names
```

Table Oracle



EXECUTE_ID	RESULT	EMPLOYEE_ID	SOFTWARE_ID
000000001	Fail	000000003	000000003
000000002	Fail	000000001	000000010
000000003	Pass	000000001	000000010
000000004	Fail	000000003	000000003
000000005	Pass	000000001	000000001
000000006	Pass	000000001	000000010
000000007	Pass	000000001	000000003
000000008	Pass	000000003	000000003
000000009	Fail	000000007	000000003
000000010	Pass	000000007	000000003

10 rows returned in 0.00 seconds Download

TIME TABLE

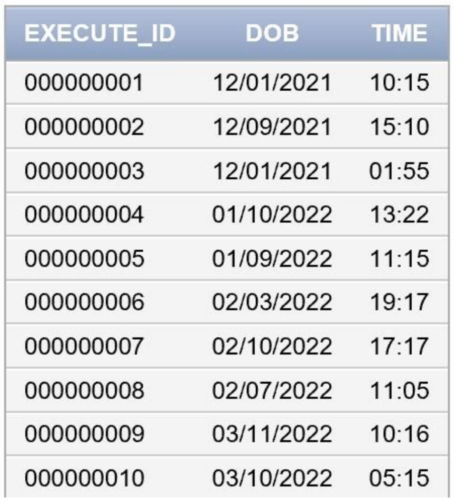
Table SQL

```
CREATE TABLE Time(
Execute_Id VARCHAR(9),
DOB DATE,
Time VARCHAR(5),
CONSTRAINT Time_TId_pk PRIMARY KEY(Execute_Id),
CONSTRAINT tefk FOREIGN KEY (Execute_Id) REFERENCES Executes(Execute_Id));
```

Tuple SQL

```
INSERT INTO Time
WITH names (Execute_Id, DOB, Time) as (
SELECT '000000001', '12/01/2021', '10:15' FROM dual UNION ALL
SELECT '000000002', '12/09/2021', '15:10' FROM dual UNION ALL
SELECT '000000003', '12/01/2021', '01:55' FROM dual UNION ALL
SELECT '000000004', '01/10/2022', '13:22' FROM dual UNION ALL
SELECT '000000005', '01/09/2022', '11:15' FROM dual UNION ALL
SELECT '000000006', '02/03/2022', '19:17' FROM dual UNION ALL
SELECT '000000007', '02/10/2022', '17:17' FROM dual UNION ALL
SELECT '000000008', '02/07/2022', '11:05' FROM dual UNION ALL
SELECT '000000009', '03/11/2022', '10:16' FROM dual UNION ALL
SELECT '000000010', '03/10/2022', '05:15' FROM dual
)
SELECT * FROM names
```

Table Oracle



EXECUTE_ID	DOB	TIME
000000001	12/01/2021	10:15
000000002	12/09/2021	15:10
000000003	12/01/2021	01:55
000000004	01/10/2022	13:22
000000005	01/09/2022	11:15
000000006	02/03/2022	19:17
000000007	02/10/2022	17:17
000000008	02/07/2022	11:05
000000009	03/11/2022	10:16
000000010	03/10/2022	05:15

SQL Queries

QUERY ONE

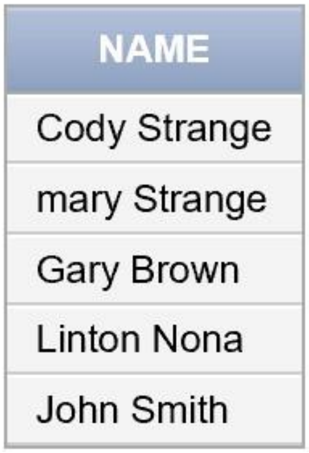
Description

Get the name of every test engineer

SQL

```
SELECT name
FROM Engineer
WHERE Role = 'Test'
```

Table



NAME
Cody Strange
mary Strange
Gary Brown
Linton Nona
John Smith

QUERY TWO

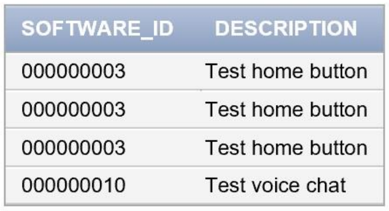
Description

Get all of the test ids and descriptions of failed tests

SQL

```
SELECT Software.Software_Id, Software.Description
FROM Software
JOIN Executes
ON Executes.Software_Id = Software.Software_Id
WHERE Result = 'Fail';
```

Table



SOFTWARE_ID	DESCRIPTION
000000003	Test home button
000000003	Test home button
000000003	Test home button
000000010	Test voice chat

QUERY THREE

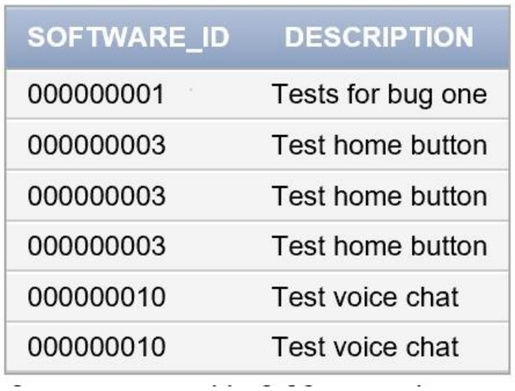
Description

Get all of the test ids and descriptions of passed tests

SQL

```
SELECT Software.Software_Id, Software.Description
FROM Software
JOIN Executes
ON Executes.Software_Id = Software.Software_Id
WHERE Result = 'Pass';
```

Table



SOFTWARE_ID	DESCRIPTION
000000001	Tests for bug one
000000003	Test home button
000000003	Test home button
000000003	Test home button
000000010	Test voice chat
000000010	Test voice chat

QUERY FOUR

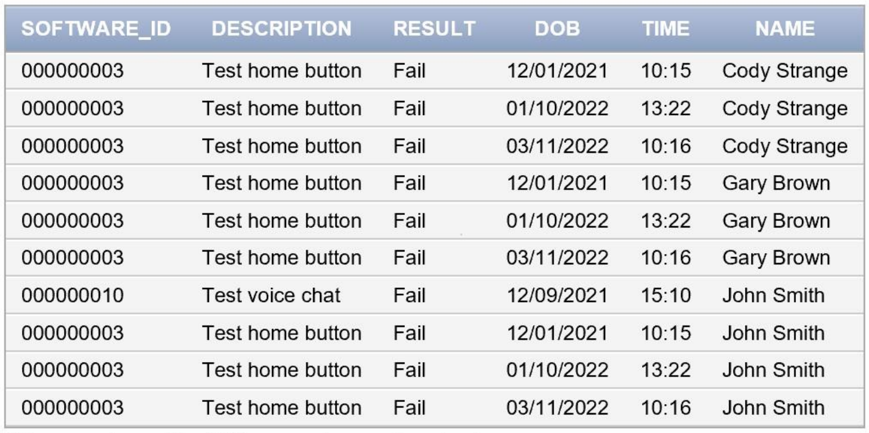
Description

What time the tests that failed were ran and the names of the engineer who ran them

SQL

```
SELECT Software.Software_Id, Software.Description, Executes.Result, Time.DOB, Time.Time, Engineer.Name
FROM Software
JOIN Executes
ON Executes.Software_Id = Software.Software_Id
JOIN Time
ON Time.Execute_Id = Executes.Execute_Id
JOIN Programs
ON Programs.Software_Id = Software.Software_Id
JOIN Engineer
ON Engineer.Employee_Id = Programs.Employee_Id
WHERE Result = 'Fail';
```

Table



SOFTWARE_ID	DESCRIPTION	RESULT	DOB	TIME	NAME
000000003	Test home button	Fail	12/01/2021	10:15	Cody Strange
000000003	Test home button	Fail	01/10/2022	13:22	Cody Strange
000000003	Test home button	Fail	03/11/2022	10:16	Cody Strange
000000003	Test home button	Fail	12/01/2021	10:15	Gary Brown
000000003	Test home button	Fail	01/10/2022	13:22	Gary Brown
000000003	Test home button	Fail	03/11/2022	10:16	Gary Brown
000000010	Test voice chat	Fail	12/09/2021	15:10	John Smith
000000003	Test home button	Fail	12/01/2021	10:15	John Smith
000000003	Test home button	Fail	01/10/2022	13:22	John Smith
000000003	Test home button	Fail	03/11/2022	10:16	John Smith

QUERY FIVE

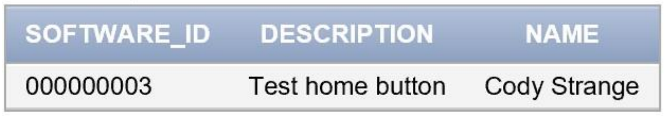
Description

What tests did the employee Cody Strange work on

SQL

SELECT Software.Software_Id, Software.Description, Engineer.Name
FROM Software
JOIN Programs
ON Programs.Software_Id = Software.software_Id
JOIN Engineer
ON Engineer.Employee_Id = Programs.Employee_Id
Where Engineer.Name = 'Cody Strange'

Table



SOFTWARE_ID	DESCRIPTION	NAME
000000003	Test home button	Cody Strange

Report

ENTITY RELATIONSHIP DIAGRAM REVIEW

What we learned

- We learned how to create an ER diagram
- How to simplify an ER diagram
- When to simplify an ER diagram
- Advantages of using an associative relationship

Problems we ran into

- We really overcomplicated our ER diagram the first time around
- How associative relationships work
- What entities and attributes are needed for a testing database

SCHEMA REVIEW

What we learned

- How to create a schema from an ER diagram
- How to show relationships between multiple schema tables
- Why schemas are important for creating a database

Problems we ran into

- Showing the foreign keys using lucid charts
- Mislabeling an attribute in the schema
- Turning a multi-value attribute into a schema

NORMALIZATION REVIEW

What we learned

- What each of the normalization forms are
- How to prove that our schemas are in BCNF starting from 1NF
- How to show dependencies between attributes in schemas

Problems we ran into

- Determining if how associative relationship schema was is BCNF or not
- Figuring out what BCNF meant

SQL TABLE CREATION REVIEW

What we learned

- How to create tables in SQL
- Adding constraints onto tables
- Having multiple attributes as the primary key
- How to add checks
- How to add foreign key restraints

Problems we ran into

- Adding multiple tables at a time
- Adding the check constraints to engineer role
- Thought int datatypes took parameters when they do not
- Adding a time datatype
- Named an attribute ‘data’ with the datatype ‘DATA’ and that is not allowed

SQL TUPLE CREATION REVIEW

What we learned

- How to add multiple rows into a table at a time
- How to insert rows into a table

Problems we ran into

- Inserting multiple rows into a table

- Inserting multiple rows into a table when the table has two primary keys of the same datatype as the only attributes

SQL QUERIE REVIEW

What we learned

- How to come up with a good SQL query
- How to convert an English description of a query into a SQL query
- How to join multiple tables and grab specific values from the new table

Problems we ran into

- Deciding what SQL queries to create
- Confirming that our SQL query results are correct
- Joining tables using just the 'from' command

OVERALL PROJECT REVIEW

What we learned

- How to get relevant information from an interview
- How to create an ER diagram
- How to create a schema from an ER diagram
- How to convert a schema into a table in Oracle
- How to fill a table with rows
- How to write SQL queries to get specific information from Oracle
- How to create a database about software testing

Problems we ran into

- Finding someone to interview
- Creating an overly complicated ER diagram
- Understanding what BCNF was exactly
- Creating multiple tables at once in Oracle
- Filling a table with multiple rows at once in Oracle