Waterford Institute *of* Technology

Dean Gaffney

20067423

Database Design

Lucy White

Changes To Design:

From my design report I have made small very small changes within several of my tables. The changes mainly focused on additions rather than deletions from my design the additions were focused upon getting richer data from each table and making sure I was satisfied with the data each table was holding.
The changes are as follows for each table:

Hosting_Services:
Unchanged.

Repositories:

1. Pushes made (The number of pushes to the repo)
2. Pulls made (The number of pulls from the repo)

Projects:
Unchanged.

Programmers:
Unchanged.

Commits:
Unchanged.

Pushes:

1. Programmer_username (The programmer who made the push)

Pulls:

1. Programmer_username (The programmer who requested the pull)

Files:

1. Commit_number (The commit number the file was apart of)

As you can see the majority of the tables remained the same as the original design apart from the extra fields added to some tables to introduce more efficient lookups in the database.

Queries:

Query 1:

```
--What programmers are currently working on a specific project.
SELECT p.programmer_id AS "ID",p.programmer_first_name || ' '
        || p.programmer_last_name AS "Name",
       pro.project_id AS "Project ID",
       pro.project_name AS "Project Name"
FROM Programmers p JOIN Projects pro ON p.project_id = pro.project_id
WHERE p.project_id = pro.project_id AND pro.project_name LIKE '%Kev%';
```

The above query is a select statement which allows me to join the programmers table and the projects table together to search for a specific project and then retrieve all the details of the programmers who are currently assigned to that specific project in question.
As you can see above I created aliases for the required programmers fields and similarly with the project fields. I then joined the two tables based off the project_id field which both have in common and searched for the project_name field which was like 'Kev'. This returns back the programmers working on the project 'Kevins Crow'.

Query 2:

```
--What programmer wrote a particular file. (uses regex to find files ending with .java or
.py)
SELECT p.programmer_id AS "ID",p.programmer_first_name AS "First Name",
        p.programmer_last_name AS "Last Name",f.file_name AS "File Name"
FROM Programmers p JOIN Files f ON p.programmer_id = f.programmer_id
WHERE p.programmer_id = f.programmer_id AND REGEXP_LIKE
(f.file_name,'\.(java|py)$');
```

The above query is a select statement which joins the programmers table and the files table to search for which programmer wrote a specific file or a certain type of file. In the above example I used aliases to clean up the fields readability when the data is presented and joined the two tables based on programmer_id.
For the above example I also demonstrated how the query could be used in a different way, it would usually be given a file name to see who wrote it but instead I used Regular Expressions to find all file names that ended with either a '.java' or '.py' extension. This then returns a list of java and python files and the programmers who wrote them.

Query 3:

```
--How many repositories are set up
SELECT SUM(num_of_repos) AS "No. of Repositories" FROM Hosting_Services;
```

This query sums up all of the num_of_repos fields in the Hosting_Services table and sums them together and returns the total number of repositories I have set up across all hosting services. So in this example I have two services github and bitbucket and this will return the total amount of repositories I have between both sites.

Query 4:

```
--The average lines of code per file.
SELECT AVG(num_of_lines) AS "Average Lines of Code" FROM Files;
```

This query returns the average amount of lines of code that are used per file. This information may be helpful for bigger projects where maintenance is essential and having less lines per file makes debugging more easier to carry out. So this query can monitor this threshold if needs be.

Query 5:

```
--What language is the project being coded in.
SELECT '----'  || project_language || '----' AS "Language"
FROM Projects
WHERE project_name = 'Kevins Crow';
```

This query is a very basic and useful query to search the database and see what programming language a specific project is being coded in.
The above example search for the project ‘Kevins Crow’ and will return ‘Python’ as the answer.

Query 6:

```
--All programmers that are registered to a repository hosting service.
SELECT programmer_id AS "ID",
        programmer_first_name ||' ' ||programmer_last_name AS "Name",
        repo_username AS "Repository Username"
FROM Programmers;
```

This query returns all the programmers with a Repository username and displays their ID,Name and Repository Username.

Query 7:

```
--List all repositories which are up to up to Date
SELECT 'ID--->' || repo_id ||  '    NAME--->' || repo_name
AS "Not Updated Repositories"
FROM Repositories
WHERE up_to_date = 'N';
```

This query simply returns a list of all the repositories which are flagged as needing to be updated with their local branches. This is indicated by the search for the up to date field is ‘N’ instead of ‘Y’ (Yes/No).

Query 8:

```
--See if any dead lines for projects are within the current month.
SELECT project_id AS "ID", project_name AS "Name", project_dead_line
AS "DeadLine"
FROM Projects
WHERE EXTRACT(month FROM project_dead_line) = EXTRACT(month FROM
SYSDATE);
```

This query checks the projects table and goes through all of the deadline dates and makes makes sure there are no deadlines due within the month.
This is simply done by using the extract method on the deadline date and retrieving the month from the date and then extracting the month from the current system date and comparing both months. This will tell me if any deadlines are within the month.

Query 9:

```
--The last time a repository had a commit made to it.
SELECT repo_id AS "ID",repo_name AS "Name",
            MAX(last_commit_time) AS "Last Commit"
FROM Repositories
WHERE repo_name = 'kevins_crow'
GROUP BY repo_id, repo_name
ORDER BY repo_id;
```

This query checks the last time a commit was made to a certain repository. This is done by selecting the Max commit_time associated with the specific repository and return the repository name and the last time a commit was made to the repository.

Query 10:

```
--What comment was given to a certain commit.
SELECT commit_number AS "Commit Number",
        commit_comment AS "Comment"
FROM Commits
WHERE commit_number = 1;
```

This query checks the commit table and returns the comment given to a certain commit by specifying the commit_number you would like to choose .

SQL Script:

```
--Create Repository Hosting Serivce
CREATE TABLE Hosting_Services
(
        service_name VARCHAR2 (50) NOT NULL,
        service_url VARCHAR2 (100) NOT NULL,
        num_of_repos INT DEFAULT 0,
        sign_up_date DATE DEFAULT SYSDATE,
        CONSTRAINT pk_service_url PRIMARY KEY (service_url),
        CHECK(num_of_repos >=0 AND num_of_repos <= 500)
);

--Populate Hosting Services Table
INSERT INTO Hosting_Services
VALUES('GitHub','www.github.com',0,SYSDATE);

INSERT INTO Hosting_Services
VALUES('BitBucket','www.bitbucket.com',0,SYSDATE);

--Drop hosting_services table
DROP TABLE Hosting_Services;
-----------------------------------------------------------------------------------------------------------
--Create Repository
CREATE TABLE Repositories
(
        repo_id INT NOT NULL,
        repo_name VARCHAR2(30) NOT NULL,
        branch_name VARCHAR2(30) NOT NULL,
        repo_size VARCHAR2(20) NOT NULL, --eg. '3mb'
        num_of_files INT DEFAULT 0,
        has_readme CHAR DEFAULT 'N',
        up_to_date CHAR DEFAULT 'N',
        last_commit_time TIMESTAMP DEFAULT SYSDATE,
        pushes_made INT DEFAULT 0,
        pulls_made INT DEFAULT 0,
        repo_url VARCHAR2(100) NOT NULL,
        service_url VARCHAR2(100) NOT NULL,
        CONSTRAINT pk_repo_id PRIMARY KEY (repo_id),
```

```
        CONSTRAINT fk_service_url FOREIGN KEY (service_url) REFERENCES
Hosting_Services(service_url),
        CHECK(repo_id > 0),
        CHECK(has_readme = 'Y'OR has_readme = 'N'),
        CHECK(up_to_date = 'Y' OR up_to_date = 'N')
);

--Populate Repositories Table
INSERT INTO Repositories
VALUES(1,'huffman_coding','origin','5mb',20,'Y','Y',
            SYSDATE,0,0,'www.github.com/huffman_coding',(SELECT (service_url)FROM
Hosting_Services
            WHERE service_url = 'www.github.com'));


INSERT INTO Repositories
VALUES((SELECT MAX(repo_id) FROM Repositories) +
1,'postfix_stack_calculator','origin','8mb',18,'N','Y',
            SYSDATE,0,0,'www.bitbucket.com/postfix_stack_calculator',(SELECT
(service_url)FROM Hosting_Services
            WHERE service_url = 'www.bitbucket.com'));

INSERT INTO Repositories
VALUES((SELECT MAX(repo_id) FROM Repositories) +
1,'x_and_o','origin','8mb',18,'N','Y',
            SYSDATE,0,0,'www.bitbucket.com/x_and_o',(SELECT (service_url)FROM
Hosting_Services
            WHERE service_url = 'www.bitbucket.com'));

INSERT INTO Repositories
VALUES((SELECT MAX(repo_id) FROM Repositories) +
1,'movie_recommender','origin','8mb',18,'N','Y',
            SYSDATE,0,0,'www.github.com/movie_recommender',(SELECT
(service_url)FROM Hosting_Services
            WHERE service_url = 'www.github.com'));

INSERT INTO Repositories
VALUES((SELECT MAX(repo_id) FROM Repositories) +
1,'kevins_crow','origin','24kb',18,'Y','Y',
```

```
        SYSDATE,0,0,'www.github.com/kevins_crow',(SELECT (service_url)FROM
Hosting_Services
        WHERE service_url = 'www.github.com'));

--Update the Repo Number Counts after Inserts.
UPDATE Hosting_Services
SET num_of_repos = (SELECT COUNT(*) FROM Repositories WHERE service_url =
'www.github.com')
WHERE service_name = 'GitHub';

UPDATE Hosting_Services
SET num_of_repos = (SELECT COUNT(*) FROM Repositories WHERE service_url =
'www.bitbucket.com')
WHERE service_name = 'BitBucket';

--Drop repositories table
DROP TABLE Repositories;

-------------------------------------------------------------------------------------------------------------

--Create the Projects table
CREATE TABLE Projects
(
        project_id INT NOT NULL,
        project_name VARCHAR2(100) NOT NULL,
        project_start_date DATE DEFAULT SYSDATE,
        project_dead_line DATE NOT NULL,
        project_programmers INT DEFAULT 0,
        project_language VARCHAR2(50) NOT NULL,
        project_ide VARCHAR2(50) DEFAULT 'Undecided at this time',
        project_brief VARCHAR2(300) DEFAULT 'To be discussed',
        has_repo CHAR DEFAULT 'N',
        repo_id INT NULL,
        CONSTRAINT pk_project_id PRIMARY KEY (project_id),
        CONSTRAINT fk_repo_id FOREIGN KEY (repo_id) REFERENCES
Repositories(repo_id),
        CHECK(project_id > 0),
        CHECK(has_repo = 'Y' OR has_repo = 'N')
);
```

```
--Populate Projects Table
INSERT INTO Projects
VALUES(1,'Huffman Coding',SYSDATE,'23-SEP-2016',NULL,'Java','Eclipse',
        'Compress a text file using huffman coding techniques.','Y',
        (SELECT (repo_id) FROM Repositories WHERE repo_name =
'huffman_coding'));

INSERT INTO Projects
VALUES((SELECT MAX(project_id) FROM Projects)+1,'Postfix Stack Calculator',
            SYSDATE,'20-MAR-2017',NULL,'C++','Visual Studio 2013','Make a
calculator with stacks
            using postfix notation.','Y',(SELECT (repo_id) FROM Repositories
WHERE repo_name = 'postfix_stack_calculator'));

INSERT INTO Projects
VALUES((SELECT MAX(project_id) FROM Projects)+1,'X and O',
            SYSDATE,'20-MAY-2017',NULL,'C++','Visual Studio 2013','Make a game
of X and O against an AI Computer','Y',
            (SELECT (repo_id) FROM Repositories WHERE repo_name =
'x_and_o'));

INSERT INTO Projects
VALUES((SELECT MAX(project_id) FROM Projects)+1,'Movie Recommender',
            SYSDATE,'15-JUN-2016',NULL,'Java','Eclipse','A Movie Recommender
that recommends movies to users','Y',
            (SELECT (repo_id) FROM Repositories WHERE repo_name =
'movie_recommender'));

INSERT INTO Projects
VALUES((SELECT MAX(project_id) FROM Projects)+1,'Kevins Crow',
            SYSDATE,'10-JUL-2016',NULL,'Python','Sublime Text','A script to send
pictures of crows to Kevin','Y',
            (SELECT (repo_id) FROM Repositories WHERE repo_name =
'kevins_crow'));

--Drop projects table
DROP TABLE Projects;
```

```
-----------------------------------------------------------------------------------------------------------------

--Create Programmers table
CREATE TABLE Programmers
(
	programmer_id INT NOT NULL,
	programmer_first_name VARCHAR2(20) NOT NULL,
	programmer_last_name VARCHAR2(20) NOT NULL,
	programmer_dob DATE NOT NULL,
	hire_date DATE DEFAULT SYSDATE,
	repo_username VARCHAR2(40) NOT NULL,
	project_id INT NULL,
	CONSTRAINT pk_programmer_id PRIMARY KEY (programmer_id),
	CONSTRAINT fk_project_id FOREIGN KEY (project_id) REFERENCES
Projects(project_id),
	CHECK(programmer_id > 0)
);

--Populate Progammers Table
INSERT INTO Programmers
VALUES(1,'Dean','Gaffney','20-MAR-1996','13-SEP-2014','Gaffmasterflex',
	(SELECT (project_id) FROM Projects WHERE project_name = 'Kevins Crow'));

INSERT INTO Programmers
VALUES((SELECT MAX(programmer_id) FROM Programmers) +
1,'Shawn','Michaels','15-JUN-1983','31-OCT-2014','HBK',
	(SELECT (project_id) FROM Projects WHERE project_name = 'Huffman
Coding'));

INSERT INTO Programmers
VALUES((SELECT MAX(programmer_id) FROM Programmers) +
1,'Steve','Austin','01-JAN-1990','15-NOV-2014','SteveAustin316',
	(SELECT (project_id) FROM Projects WHERE project_name = 'Postfix Stack
Calculator'));

INSERT INTO Programmers
VALUES((SELECT MAX(programmer_id) FROM Programmers) +
1,'Philip','Meagher','20-MAR-1992','10-NOV-2014','PhillyMeagher',
	(SELECT (project_id) FROM Projects WHERE project_name = 'Kevins Crow'));
```

```
-----------------------------------------------------------------------------------------------------------------
UPDATE PROJECTS TABLE

UPDATE Projects
SET project_programmers = (SELECT COUNT(*) FROM Programmers
        WHERE project_id = (SELECT (project_id) FROM Projects WHERE
project_name = 'Kevins Crow'))
WHERE project_name = 'Kevins Crow';

UPDATE Projects
SET project_programmers = (SELECT COUNT(*) FROM Programmers
        WHERE project_id = (SELECT (project_id) FROM Projects WHERE
project_name = 'Huffman Coding'))
WHERE project_name = 'Huffman Coding';

UPDATE Projects
SET project_programmers = (SELECT COUNT(*) FROM Programmers
        WHERE project_id = (SELECT (project_id) FROM Projects WHERE
project_name = 'Postfix Stack Calculator'))
WHERE project_name = 'Postfix Stack Calculator';

UPDATE Projects
SET project_programmers = (SELECT COUNT(*) FROM Programmers
        WHERE project_id = (SELECT (project_id) FROM Projects WHERE
project_name = 'Movie Recommender'))
WHERE project_name = 'Movie Recommender';

UPDATE Projects
SET project_programmers = (SELECT COUNT(*) FROM Programmers
        WHERE project_id = (SELECT (project_id) FROM Projects WHERE
project_name = 'X and O'))
WHERE project_name = 'X and O';
-----------------------------------------------------------------------------------------------------------------
DROP PROGRAMMERS TABLE (DEBUG)
--Drop programmers table
DROP TABLE Programmers;
```

```
-------------------------------------------------------------------------------------------------------------------
CREATE COMMITS TABLE

--Create Commits Table
CREATE TABLE Commits
(
	commit_number INT NOT NULL,
	commit_comment VARCHAR2(50) NOT NULL,
	num_of_files INT DEFAULT 0,
	repository_id INT NOT NULL,
	commit_time TIMESTAMP DEFAULT SYSDATE,
	CONSTRAINT pk_commit_number PRIMARY KEY (commit_number),
	CONSTRAINT fk_repository_id FOREIGN KEY (repository_id) REFERENCES
Repositories(repo_id),
	CHECK(commit_number > 0)
);

-------------------------------------------------------------------------------------------------------------------
------ INSERT INTO COMMITS TABLE
--Populate Commits Table
INSERT INTO Commits
VALUES (1,'Added multiple recipients to email sending.',0,(SELECT (repo_id) FROM
Repositories WHERE repo_name = 'kevins_crow'),
		SYSDATE);

INSERT INTO Commits
VALUES ((SELECT MAX(commit_number) FROM Commits) + 1,'Postfix evaluation
working.',0,(SELECT (repo_id) FROM Repositories WHERE repo_name =
'postfix_stack_calculator'),
		SYSDATE);

INSERT INTO Commits
VALUES ((SELECT MAX(commit_number) FROM Commits) + 1,'Users ArrayList fully
populated',0,(SELECT (repo_id) FROM Repositories WHERE repo_name =
'movie_recommender'),
		SYSDATE);
```

```
INSERT INTO Commits
VALUES ((SELECT MAX(commit_number) FROM Commits) + 1,'Text file is now being
compressed',0,(SELECT (repo_id) FROM Repositories WHERE repo_name =
'huffman_coding'),
                SYSDATE);

--****** CHECK AND MAKE SURE NOTHING NEEDS TO BE ALTERED iN OTHER
TABLES BECAUSE OF THESE ADDITIONS********
-----------------------------------------------------------------------------------------------------------------
DROP COMMITS TABLE (DEBUG)
--Drop commits table
DROP TABLE Commits;

-----------------------------------------------------------------------------------------------------------------
CREATE PUSHES TABLE
CREATE TABLE Pushes
(
        push_id INT NOT NULL,
        time_of_push TIMESTAMP DEFAULT SYSDATE,
        repo_id INT NOT NULL,
        programmer_username VARCHAR2(100) NOT NULL,
        CONSTRAINT pk_push_id PRIMARY KEY (push_id),
        CONSTRAINT fk_rep_id FOREIGN KEY (repo_id) REFERENCES
Repositories(repo_id),
        CHECK(push_id > 0)
);
-----------------------------------------------------------------------------------------------------------------
INSERT INTO PUSHES TABLE
--Populate Push Table
INSERT INTO Pushes
VALUES(1,SYSDATE,(SELECT (repo_id) FROM Repositories WHERE repo_name =
'kevins_crow'),
        (SELECT (repo_username) FROM Programmers WHERE programmer_id = 1));

INSERT INTO Pushes
VALUES((SELECT MAX(push_id) FROM Pushes) + 1,SYSDATE,(SELECT (repo_id)
FROM Repositories WHERE repo_name = 'huffman_coding'),
        (SELECT (repo_username) FROM Programmers WHERE programmer_id = 2));
```

```
INSERT INTO Pushes
VALUES((SELECT MAX(push_id) FROM Pushes) + 1,SYSDATE,(SELECT (repo_id)
FROM Repositories WHERE repo_name = 'movie_recommender'),
        (SELECT (repo_username) FROM Programmers WHERE programmer_id = 3));

INSERT INTO Pushes
VALUES((SELECT MAX(push_id) FROM Pushes) + 1,SYSDATE,(SELECT (repo_id)
FROM Repositories WHERE repo_name = 'x_and_o'),
        (SELECT (repo_username) FROM Programmers WHERE programmer_id = 4));

-----------------------------------------------------------------------------------------------------------------
UPDATE REPOSITORIES TABLE
UPDATE Repositories
SET pushes_made = (SELECT COUNT(*) FROM Pushes
        WHERE repo_id = (SELECT (repo_id) FROM Repositories WHERE repo_name
= 'kevins_crow'))
WHERE repo_name = 'kevins_crow';

UPDATE Repositories
SET pushes_made = (SELECT COUNT(*) FROM Pushes
        WHERE repo_id = (SELECT (repo_id) FROM Repositories WHERE repo_name
= 'movie_recommender'))
WHERE repo_name = 'movie_recommender';

UPDATE Repositories
SET pushes_made = (SELECT COUNT(*) FROM Pushes
        WHERE repo_id = (SELECT (repo_id) FROM Repositories WHERE repo_name
= 'x_and_o'))
WHERE repo_name = 'x_and_o';

UPDATE Repositories
SET pushes_made = (SELECT COUNT(*) FROM Pushes
        WHERE repo_id = (SELECT (repo_id) FROM Repositories WHERE repo_name
= 'huffman_coding'))
WHERE repo_name = 'huffman_coding';

-----------------------------------------------------------------------------------------------------------------
DROP PUSHES TABLE (DEBUG)
--Drop Push Table
```

```
DROP TABLE Pushs;

-----------------------------------------------------------------------------------------------------------------
CREATE TABLE PULLS
CREATE TABLE Pulls
(
	pull_id INT NOT NULL,
	repo_id INT NOT NULL,
	pull_time TIMESTAMP DEFAULT SYSDATE,
	programmer_username VARCHAR2(100) NOT NULL,
	CONSTRAINT pk_pull_id PRIMARY KEY (pull_id),
	CONSTRAINT fk_repos_id FOREIGN KEY (repo_id) REFERENCES
Repositories(repo_id),
	CHECK(pull_id > 0)
);
-----------------------------------------------------------------------------------------------------------------
INSERT INTO PULLS TABLE

INSERT INTO Pulls
VALUES(1,(SELECT (repo_id) FROM Repositories WHERE repo_name =
'kevins_crow'),SYSDATE,
	(SELECT (repo_username) FROM Programmers WHERE programmer_id = 1));

INSERT INTO Pulls
VALUES((SELECT MAX(pull_id) FROM Pulls) + 1,(SELECT (repo_id) FROM
Repositories WHERE repo_name = 'movie_recommender'),SYSDATE,
	(SELECT (repo_username) FROM Programmers WHERE programmer_id = 2));

INSERT INTO Pulls
VALUES((SELECT MAX(pull_id) FROM Pulls) + 1,(SELECT (repo_id) FROM
Repositories WHERE repo_name = 'x_and_o'),SYSDATE,
	(SELECT (repo_username) FROM Programmers WHERE programmer_id = 3));

INSERT INTO Pulls
VALUES((SELECT MAX(pull_id) FROM Pulls) + 1,(SELECT (repo_id) FROM
Repositories WHERE repo_name = 'huffman_coding'),SYSDATE,
	(SELECT (repo_username) FROM Programmers WHERE programmer_id = 4));
```

```
INSERT INTO Pulls
VALUES((SELECT MAX(pull_id) FROM Pulls) + 1,(SELECT (repo_id) FROM
Repositories WHERE repo_name = 'huffman_coding'),SYSDATE,
        (SELECT (repo_username) FROM Programmers WHERE programmer_id = 1));

-------------------------------------------------------------------------------------------------------------------
UPDATE REPOSITORIES TABLES
UPDATE Repositories
SET pulls_made = (SELECT COUNT(*) FROM Pulls
        WHERE repo_id = (SELECT (repo_id) FROM Repositories WHERE repo_name
= 'kevins_crow'))
WHERE repo_name = 'kevins_crow';

UPDATE Repositories
SET pulls_made = (SELECT COUNT(*) FROM Pulls
        WHERE repo_id = (SELECT (repo_id) FROM Repositories WHERE repo_name
= 'huffman_coding'))
WHERE repo_name = 'huffman_coding';

UPDATE Repositories
SET pulls_made = (SELECT COUNT(*) FROM Pulls
        WHERE repo_id = (SELECT (repo_id) FROM Repositories WHERE repo_name
= 'x_and_o'))
WHERE repo_name = 'x_and_o';

-------------------------------------------------------------------------------------------------------------------
DROP PULLS TABLE (DEBUG)
--Delete Pulls Table
DROP TABLE Pulls;
-------------------------------------------------------------------------------------------------------------------
```

```
CREATE TABLE FILES

--Create Files table
CREATE TABLE Files
(
	file_name VARCHAR2 (40) NOT NULL,
	creation_date DATE DEFAULT SYSDATE,
	last_modified DATE DEFAULT SYSDATE,
	file_size VARCHAR2(15) NOT NULL, -- eg.'42kb'
	num_of_lines INT NOT NULL,
	programmer_id INT NOT NULL,
	completed CHAR DEFAULT 'N',
	brief VARCHAR2(300) DEFAULT 'To be discussed',
	note VARCHAR2(100) DEFAULT 'No note',
	commit_number INT NULL,
	CONSTRAINT pk_file_name PRIMARY KEY(file_name),
	CONSTRAINT fk_programmer_id FOREIGN KEY (programmer_id)
REFERENCES Programmers(programmer_id),
	CONSTRAINT fk_commit_number FOREIGN KEY (commit_number)
REFERENCES Commits(commit_number),
	CHECK(completed = 'Y' OR completed = 'N')
);

-----------------------------------------------------------------------------------------------------------
INSERT INTO FILES

INSERT INTO Files
VALUES('HuffmanCodingTree.java',SYSDATE,SYSDATE,'2kb',87,
		(SELECT (programmer_id) FROM Programmers WHERE repo_username
= 'HBK'),
		 'N','This file deals with traversing and finding nodes in tree.','Slight Bug in
tree traversal',
		 NULL);
```

```
INSERT INTO Files
VALUES('Node.java',SYSDATE,SYSDATE,'2kb',54,
                (SELECT (programmer_id) FROM Programmers WHERE repo_username
= 'HBK'),
                 'Y','Node class containing weight,left,right children and character','Fully
Functional',
                  NULL);

INSERT INTO Files
VALUES('Calculator.java',SYSDATE,SYSDATE,'1kb',112,
                (SELECT (programmer_id) FROM Programmers WHERE repo_username
= 'SteveAustin316'),
                 'N','The main GUI for the calculator','Added in power of button',
                  NULL);

INSERT INTO Files
VALUES('kevins_crow.py',SYSDATE,SYSDATE,'2kb',25,
                (SELECT (programmer_id) FROM Programmers WHERE repo_username
= 'Gaffmasterflex'),
                 'Y','Emails pictures of crows to kevin everyday','Able to send email to
serveral recipients',
                  NULL);

-----------------------------------------------------------------------------------------------------------
DROP FILES TABLE(DEBUG)
DROP TABLE Files;
-----------------------------------------------------------------------------------------------------------
CREATE ALIASES VIEWS FOR TABLES
CREATE OR REPLACE VIEW view_hosting_services_table
                ---->Hosting Service Table with Aliases
("Name","URL","No. of Repositories","Sign up Date")
AS SELECT service_name,service_url,num_of_repos,sign_up_date
FROM Hosting_Services;

SELECT * FROM view_hosting_services_table;
```

```
CREATE OR REPLACE VIEW view_repositories_table
("ID","Name","Branch Name","Size","No. of files","README","Up to date",
---->Repositories View Table with Aliases.
        "Last commit time","Pushes","Pulls","Repository URL","Serivce URL")
AS SELECT repo_id,repo_name,branch_name,repo_size,num_of_files,
                has_readme,up_to_date,last_commit_time,pushes_made,
                pulls_made,repo_url,service_url
FROM Repositories;

SELECT * FROM view_repositories_table;

CREATE OR REPLACE VIEW view_projects_table
("ID","Name","Start Date","Dead Line","No. of Programmers",
        "Programming Language","IDE","Brief","Has Repository","Repository ID")
---->Projects View Table with Aliases.
AS SELECT project_id,project_name,project_start_date,
                project_dead_line,project_programmers,
                project_language,project_ide,project_brief,
                has_repo,repo_id
FROM Projects;

SELECT * FROM view_projects_table;

CREATE OR REPLACE VIEW view_programmers_table
("ID","First Name","Last Name","Date of Birth","Hire Date",
---->Programmers View Table with Aliases.
        "Repository Username",
        "Active Project ID")
AS SELECT programmer_id,programmer_first_name,programmer_last_name,
                programmer_dob,hire_date,repo_username,project_id
FROM Programmers;

SELECT * FROM view_programmers_table;

CREATE OR REPLACE VIEW view_commits_table
("Commit Number","Comment","No. Files","Repository ID","Commit Time")
AS SELECT
commit_number,commit_comment,num_of_files,repository_id,commit_time
---->Commits View Table with Aliases.
```

```
FROM Commits;

SELECT * FROM view_commits_table;

CREATE OR REPLACE VIEW view_pushes_table
("ID","Time of Push","Repository ID","Programmer Username")
        ---->Pushes View Table with Aliases.
AS SELECT push_id,time_of_push,repo_id,programmer_username
FROM Pushes;

SELECT * FROM view_pushes_table;

CREATE OR REPLACE VIEW view_pulls_table
("ID","Repository ID","Time of Pull","Programmer Username")
        ---->Pulls View Table with Aliases.
AS SELECT pull_id,repo_id,pull_time,programmer_username
FROM Pulls;

SELECT * FROM view_pulls_table;

CREATE OR REPLACE VIEW view_files_table
                    ---->Files View Table with Aliases.
("Name","Creation Date","Last Modified","Size","No. of Lines",
"Programmer ID","Completed","Brief","Note","Commit Number")
AS SELECT file_name,creation_date,last_modified,file_size,
                num_of_lines,programmer_id,completed,brief,
                note,commit_number
FROM Files;

SELECT * FROM view_files_table;

-----------------------------------------------------------------------------------------------------------------------
```

```
CREATE VIEWS FOR SPECIFICS
--Create some a views for the programmers.

--git specific view (READ ONLY ACCESS)
CREATE OR REPLACE VIEW view_programmers_of_git_repo
("ID","Name","Branch","Files","README.md",
        "Up to Date","Pushes","Pulls","Repo URL","Service","Serive Url")
AS SELECT r.repo_id,r.repo_name,r.branch_name,r.num_of_files,
                r.has_readme,r.up_to_date,r.pushes_made,r.pulls_made,
                r.repo_url,h.service_name,h.service_url
 FROM Repositories r JOIN Hosting_Services h ON r.service_url = h.service_url
 WHERE service_name = 'GitHub'
 WITH READ ONLY;

 SELECT * FROM view_programmers_of_git_repo;

--bitbucket specific view (READ ONLY ACCESS)
CREATE OR REPLACE VIEW view_programmers_of_bit_repo
("ID","Name","Branch","Files","README.md",
        "Up to Date","Pushes","Pulls","Repo URL","Service","Serive Url")
AS SELECT r.repo_id,r.repo_name,r.branch_name,r.num_of_files,
                r.has_readme,r.up_to_date,r.pushes_made,r.pulls_made,
                r.repo_url,h.service_name,h.service_url
 FROM Repositories r JOIN Hosting_Services h ON r.service_url = h.service_url
 WHERE service_name = 'BitBucket'
 WITH READ ONLY;

  SELECT * FROM view_programmers_of_bit_repo;

 --view for the most active repo (based of of pulls - READ ONLY ACCESS)
 CREATE OR REPLACE VIEW view_most_active_repo
 ("ID","Name","Pushes","Pulls","Repo URL","Serivce","Serivce URL")
 AS SELECT r.repo_id,r.repo_name,r.pushes_made,r.pulls_made,
                 r.repo_url,h.service_name,h.service_url
 FROM Repositories r JOIN Hosting_Services h ON r.service_url = h.service_url
 WHERE pulls_made = (SELECT MAX(pulls_made) FROM Repositories)
 WITH READ ONLY;
```

```
SELECT * FROM view_most_active_repo;


-------------------------------------------------------------------------------------------------------------------------
CREATE ROLE,GRANT AND REVOKE ROLE
--Create user conor with select,update and delete privileges
CREATE USER conor_corcoran
IDENTIFIED BY conor123;

--Create a role for people to update databases.
CREATE ROLE updaters;
        --->Create role

GRANT SELECT,UPDATE,DELETE ON Programmers TO updaters;
----> Grant some priveleges to the role
GRANT updaters TO conor_corcoran;
---->GRANT USER ROLE ACCESS

--Revoke DELETE privilege.
REVOKE DELETE ON Programmers FROM updaters;
        ---->REVOKE DELETE PRIVILEEGE FROM ROLE

DROP ROLE updaters;
                ---->DROP THE ROLE
-------------------------------------------------------------------------------------------------------------------------
```

```
10 QUERIES
--1)What programmers are currently working on a specific project.
SELECT p.programmer_id AS "ID",p.programmer_first_name || ' '
            || p.programmer_last_name AS "Name",
            pro.project_id AS "Project ID",
            pro.project_name AS "Project Name"
FROM Programmers p JOIN Projects pro ON p.project_id = pro.project_id
WHERE p.project_id = pro.project_id AND pro.project_name LIKE '%Kev%';

--2)What programmer wrote a particular file. (uses regex to find files ending with .java or
.py)
SELECT p.programmer_id AS "ID",p.programmer_first_name AS "First Name",
          p.programmer_last_name AS "Last Name",f.file_name AS "File Name"
FROM Programmers p JOIN Files f ON p.programmer_id = f.programmer_id
WHERE p.programmer_id = f.programmer_id AND REGEXP_LIKE
(f.file_name,'\.(java|py)$');

--3)How many repositories are set up

         ----> QUEREY 3.
SELECT SUM(num_of_repos) AS "No. of Repositories" FROM Hosting_Services;


--4)The average lines of code per file.
SELECT AVG(num_of_lines) AS "Average Lines of Code" FROM Files;


--5)What language is the project being coded in.
SELECT '----'  || project_language || '----' AS "Language"
FROM Projects
WHERE project_name = 'Kevins Crow';

--6)All programmers that are registered to a repository hosting service.
SELECT programmer_id AS "ID",
            programmer_first_name ||' ' ||programmer_last_name AS "Name",
            repo_username AS "Repository Username"
FROM Programmers;
```

```
--7)List all repositories which are up to up to Date
SELECT 'ID--->' || repo_id ||  '    NAME--->' || repo_name
AS "Not Updated Repositories"
FROM Repositories
WHERE up_to_date = 'N';

--8)See if any dead lines for projects are within the current month.
SELECT project_id AS "ID", project_name AS "Name", project_dead_line
AS "DeadLine"
FROM Projects
WHERE EXTRACT(month FROM project_dead_line) = EXTRACT(month FROM
SYSDATE);

--9)The last time a repository had a commit made to it.
        SELECT repo_id AS "ID",repo_name AS "Name",
          MAX(last_commit_time) AS "Last Commit"
FROM Repositories
WHERE repo_name = 'kevins_crow'
GROUP BY repo_id, repo_name
ORDER BY repo_id;

--10)What comment was given to a certain commit.
SELECT commit_number AS "Commit Number",
          commit_comment AS "Comment"
FROM Commits
WHERE commit_number = 1;
```