**Enrollments DB**

Create table Student( stuID varchar(6) NOT NULL Primary Key, lastName varchar(20) NOT NULL, firstName varchar(20) NOT NULL, major varchar(10), Credits int )

Create table Faculty( facID varchar(6) NOT NULL, name varchar(20) NOT NULL, department varchar(20), rank varchar(10), CONSTRAINT Faculty\_facID\_pk PRIMARY KEY (facID))

Create table Class( classNumber varchar(8), facID varchar(6) references Faculty (facID) ON DELETE SET NULL, Schedule varchar(8), room varchar(6), CONSTRAINT Class\_classNumber\_pk PRIMARY KEY (classNumber) )

Create table Enroll( stuID varchar(6), classNumber varchar(8), grade varchar(2), CONSTRAINT Enroll\_classnumber\_stuId PRIMARY KEY (classNumber, stuID), CONSTRAINT Enroll\_classnumber\_fk FOREIGN KEY (classnumber) references Class (classNumber) ON DELETE CASCADE, CONSTRAINT Enroll\_stuID\_fk FOREIGN KEY (stuID) references Student (stuID) ON DELETE CASCADE )

Insert INTO Student VALUES ('S1001', 'Smith', 'Tom', 'History', 90);

Insert INTO Student VALUES ('S1002', 'Chin', 'Ann', 'Math', 36);

Insert INTO Student VALUES ('S1005', 'Lee', 'Perry', 'History', 3);

Insert INTO Student VALUES ('S1010', 'Burns', 'Edwards', 'Art', 63);

Insert INTO Student VALUES ('S1013', 'McCarthy', 'Owen', 'Math', 10);

Insert INTO Student VALUES ('S1015', 'Jones', 'Mary', 'Math', 42);

Insert INTO Student VALUES ('S1020', 'Rivera', 'Jane', 'CSC', 15);

Insert INTO Faculty VALUES ('F101', 'Adams', 'Art', 'Professor');

Insert INTO Faculty VALUES ('F105', 'Tanaka', 'CSC', 'Instructor');

Insert INTO Faculty VALUES ('F110', 'Byrne', 'Math', 'Assistant');

Insert INTO Faculty VALUES ('F115', 'Smith', 'History', 'Associate');

Insert INTO Faculty VALUES ('F221', 'Smith', 'CSC', 'Professor');

Insert INTO Class VALUES ('ART103A', 'F101', 'MWF9', 'H221');

Insert INTO Class VALUES ('CSC201A', 'F105', 'TTH10', 'M110');

Insert INTO Class VALUES ('CSC203A', 'F105', 'MTF12', 'M110');

Insert INTO Class VALUES ('HST205A', 'F115', 'MWF11', 'H221');

Insert INTO Class VALUES ('MTH101B', 'F110', 'MTTH9', 'H225');

Insert INTO Class VALUES ('MTH103A', 'F110', 'MWF11', 'H225');

INSERT INTO Enroll VALUES ('S1001', 'ART103A', 'A');

INSERT INTO Enroll VALUES ('S1001', 'HST205A', 'C');

INSERT INTO Enroll VALUES ('S1002', 'ART103A', 'D');

INSERT INTO Enroll VALUES ('S1002', 'CSC201A', 'F');

INSERT INTO Enroll VALUES ('S1002', 'MTH103A', 'B');

INSERT INTO Enroll VALUES ('S1010', 'ART103A', 'A');

INSERT INTO Enroll VALUES ('S1010', 'MTH103A', 'B');

INSERT INTO Enroll VALUES ('S1020', 'CSC201A', 'B');

INSERT INTO Enroll VALUES ('S1020', 'MTH101B', 'A');

ERD AD CAMPAIGN

CREATE TABLE campaignManager (managerID int PRIMARY KEY NOT NULL, managerName varchar(30) NOT NULL);

CREATE TABLE adCampaign (campaignID int PRIMARY KEY NOT NULL, managerID int FOREIGN KEY REFERENCES campaignManager(managerID), startDate date NOT NULL, duration int);

CREATE TABLE campaignMode (modeID int PRIMARY KEY NOT NULL, media varchar(20), cRange varchar(20));

CREATE TABLE campaignBudget (campaignID int FOREIGN KEY REFERENCES adCampaign(campaignID), modeID int FOREIGN KEY REFERENCES campaignMode(modeID), budgetPctg int);

--Inserts Successfully

INSERT INTO campaignManager VALUES(100, 'Roberta');

INSERT INTO campaignManager VALUES(101, 'Sue');

INSERT INTO campaignManager VALUES (102, 'John');

INSERT INTO campaignManager VALUES (103, 'Nancy');

--Inserts Successfully

INSERT INTO adCampaign VALUES (111, 100, 'SummerFun13', '6/6/2013', 12);

INSERT INTO adCampaign VALUES (222, 101, 'SummerZing13', '6/6/2013',30);

INSERT INTO adCampaign VALUES (333, 102, 'FallBall13', '6/9/2013', 12);

INSERT INTO adCampaign VALUES (444, 103, 'AutumnStyle13', '6/9/2013', 5);

INSERT INTO adCampaign VALUES (555, 100, 'AutumnColors13', '6/9/2013', 3);

--Inserts Successfully

INSERT INTO campaignMode VALUES (1, 'TV', 'Local');

INSERT INTO campaignMode VALUES (2, 'TV', 'National');

INSERT INTO campaignMode VALUES (3, 'Radio', 'Local');

INSERT INTO campaignMode VALUES (4, 'Radio', 'National');

INSERT INTO campaignMode VALUES (5, 'Print', 'Local');

INSERT INTO campaignMode VALUES (6, 'Print', 'National');

INSERT INTO campaignMode VALUES (7, 'Internet', 'National');

--Inserts Successfully

INSERT INTO campaignBudget VALUES (111, 1, 50);

INSERT INTO campaignBudget VALUES (111, 2, 50);

INSERT INTO campaignBudget VALUES (222, 1, 60);

INSERT INTO campaignBudget VALUES (222,3, 30);

INSERT INTO campaignBudget VALUES (222, 5, 10);

INSERT INTO campaignBudget VALUES (333, 3, 80);

INSERT INTO campaignBudget VALUES (333, 4, 20);

INSERT INTO campaignBudget VALUES (444, 6, 100);

INSERT INTO campaignBudget VALUES (555, 3, 100);

Business Trip ERD CREATE TABLE Employee (SSN int PRIMARY KEY NOT NULL, name varchar(20) NOT NULL, deptNum int NOT NULL, jobTitle varchar(50) NOT NULL, salary float NOT NULL);

CREATE TABLE Trip (TripID int PRIMARY KEY NOT NULL, departureCity varchar(30) NOT NULL, destinationCity varchar(30) NOT NULL, departureDate date NOT NULL, returnDate date NOT NULL, SSN int FOREIGN KEY REFERENCES Employee(SSN) NOT NULL);

CREATE TABLE Expense (TripID int FOREIGN KEY REFERENCES Trip(TripID) NOT NULL, item varchar(20) NOT NULL, expenseDate date NOT NULL, amount float NOT NULL, PRIMARY KEY(TripID, item));

\*\*\*NOTE: in the Employee table it should all be NOT NULL

INSERT INTO EMPLOYEE VALUES

(111, 'Mike Smith', 10, 'project manager', 150000),

(222, 'Dan Prime', 10, 'IT Analyst', 65000),

(333, 'Scott Porter', 20, 'Data Entry', 40000),

(444, 'Dave Wolff', 30, 'Payroll Analyst', 57000),

(555, 'Tina Dunn', 40, 'Sales Manager', 155000),

(666, 'Chris Sullivan', 50, 'Consultant', 77000)

INSERT INTO Trip VALUES

(21, 'Phoenix', 'Denver', '01/01/2020', '01/14/2020', 111),

(31, 'Fargo', 'Seattle', '12/15/2019', '12/31/2019', 222),

(41, 'Hartford', 'Portland', '11/11/2019', '11/22/2019', 333),

(51, 'Dallas', 'Charlotte', '02/02/2020', '02/10/2020', 444),

(61, 'London', 'Chicago', '02/02/2020', '02/10/2020', 555),

(71, 'Oakland', 'Richmond', '05/15/2019', '06/01/2019', 666 )

INSERT INTO EXPENSE VALUES (21, 'Lunch', '01/01/2020', 20),

(21, 'Dinner', '01/01/2020', 20),

(31, 'Car Rental', '02/01/2020', 250),

(51, 'Hotel', '11/11/2019', 550),

(41, 'Hotel', '12/15/2019', 1050),

(31, 'Lunch', '01/01/2020', 20),

(61, 'Entertainment', '2/5/2020', 125)

--Could update table trip is cols mixd up

--INSERT INTO Trip VALUES (--column names, col name, col name---)

--(21, 'Phoenix', 'Denver', '01/01/2020', '01/14/2020', 111),

--(31, 'Fargo', 'Seattle', '12/15/2019', '12/31/2019', 222),

--(41, 'Hartford', 'Portland', '11/11/2019', '11/22/2019', 333),

--(51, 'Dallas', 'Charlotte', '02/02/2020', '02/10/2020', 444),

--(61, 'London', 'Chicago', '02/02/2020', '02/10/2020', 555),

--(71, 'Oakland', 'Richmond', '05/15/2019', '06/01/2019', 666 )

--If Trip table missing return date

--ALTER TABLE Trip returnDate date NOT NULL;

--If composite key was note created in expense originally

--ALTER TABLE Expense ADD PRIMARY KEY (tripID, item)

--give a list of all the destination cities where employees have taken trips to

SELECT destinationCity FROM Trip

SELECT destinationCity FROM Trip WHERE SSN IS NOT NULL

--find all employees information for employee who work in dept 10

SELECT \* FROM Employee WHERE deptNum = 10;

--Get a complete trip record for all the trips with departure dates after Jan 1 2020

SELECT \* FROM Trip WHERE departureDate > '1-1-2020'

--find the names of all employees who have departed on trips from London

SELECT name FROM Employee e JOIN Trip t ON e.SSN = t.SSN WHERE destinationCity = 'London'

SELECT \* FROM Trip --to show that my destination city was swapper with departure city

--find social security number for all employess that have any single expense item of more than $1000 for a trip

SELECT e.SSN, name, amount FROM Employee e JOIN Trip t ON e.SSN = t.SSN JOIN Expense x ON t.TripID = x.TripID WHERE amount > 1000;

--find the destination cities of all trips taken by employees who have the job titles of "consultant"

SELECT destinationCity, name FROM Trip t JOIN Employee e ON t.SSN = e.SSN WHERE jobTitle = 'Consultant'

WORKER DB

--Create the worker table

--Identity key word, SQL will auto-increment valuse for it (starting at 1)

--this has one primary key thus it is a strong entity

CREATE TABLE Worker (worker\_ID int NOT NULL PRIMARY KEY IDENTITY, firstName varchar(25), lastName varchar(25), salary int, joiningDate date, dept varchar(25));

INSERT INTO Worker (firstName, lastName, salary, joiningDate, dept) VALUES

('Monika', 'Arora', 100000, '02-02-2020', 'HR'),

('Niharika', 'Verma', 80000, '01-06-2011', 'Admin'),

('Vishal', 'Singhal', 100000, '03-02-2020', 'HR'),

('Amitabh', 'Singh', 500000, '04-02-2020', 'Admin'),

('Vivek', 'Bhati', 500000, '03-06-2011', 'Admin'),

('Vipul', 'Diwan', 200000, '09-06-2011', 'Account'),

('Satish', 'Kumar', 75000, '08-01-2020', 'Account'),

('Geetika','Chauhan', 9000, '10-04-2011','Admin');

--I forgot this record but added in later

INSERT INTO Worker VALUES('Geetika','Chauhan', 9000, '10-04-2011','Admin');

--Fixing the 100000 salary of Vishal to 300000

UPDATE Worker SET salary = 300000 WHERE firstName = 'Vishal' AND lastName = 'Singh';

--List how many workers we currently have

SELECT count(\*) Workers FROM Worker;

--List workers with salary greater than 300000 and less than 500000

SELECT \* FROM Worker WHERE salary >= 100000 AND salary < 500000;

--List workers who were hired after june first of 2020

SELECT \* FROM Worker WHERE joiningDate > '06-01-2020';

--List out departments and do not include duplicates

SELECT dept FROM Worker GROUP BY dept;

--could use the DISTINCT keyword here too

--Explain how you would expand this table to include departments ID and department locations

--Explain how you would expand this table to include departments ID and department locations

--\*\*I would create a new sub-table called 'department'. I would make the department column of the worker table a foreign key so I could reference department information for the employee.

--the dept ID should be the foreign key, thus making the worker a weak entity and the department table a strong entity

--Restructure the worker and create a department table: deptID (PK), name, location then add deptID as FK (worker)

CREATE TABLE Bonus (worker\_ref\_ID int, bonusAmt int, bonusDate date);

--update the table bonus to enforce a 1-m rule with the worker table. WOrker may receive bonuses

ALTER TABLE Bonus ADD FOREIGN KEY (worker\_ref\_ID) REFERENCES Worker(worker\_ID);

--because this operation is non-DML (not data related) we dont use insert, updates, or delete, we use ALTER

--when using names/ adding constraints follow format 'table\_column\_fk'

--could also use ADD CONSTRAINT

--duplicates are fine (because it is a foreign key, only primary key do not accept duplicates)

INSERT INTO Bonus (worker\_ref\_ID, bonusAmt, bonusDate) VALUES

(1, 5000, '10-02-2020'),

(2, 3000, '06-06-2020'),

(3, 4000, '08-02-2020'),

(1, 4500, '09-02-2020'),

(2, 3500, '11-06-2011');

--show me the total bonuses

SELECT SUM(bonusAmt) totalOFBonuses FROM Bonus;

--show me the first and lastname of the worker with the bonuses greater than $4000 (try this but with total bonuses to x amount)

SELECT firstName, lastName, bonusAmt FROM Worker w JOIN Bonus b ON w.worker\_ID = b.worker\_ref\_ID WHERE bonusAmt > 4000

**PROJECT DB**

Create database Projects

CREATE TABLE department(dept\_no CHAR(4) NOT NULL,

dept\_name CHAR(25) NOT NULL,

location CHAR(30) NULL,

CONSTRAINT prim\_dept PRIMARY KEY (dept\_no));

CREATE TABLE employee (emp\_no INTEGER NOT NULL,

emp\_fname CHAR(20) NOT NULL,

emp\_lname CHAR(20) NOT NULL,

dept\_no CHAR(4) NULL,

CONSTRAINT prim\_emp PRIMARY KEY (emp\_no),

CONSTRAINT foreign\_emp FOREIGN KEY(dept\_no) REFERENCES department(dept\_no));

CREATE TABLE project (project\_no CHAR(4) NOT NULL,

project\_name CHAR(15) NOT NULL,

budget FLOAT NULL,

CONSTRAINT prim\_proj PRIMARY KEY (project\_no));

CREATE TABLE works\_on (emp\_no INTEGER NOT NULL,

project\_no CHAR(4) NOT NULL,

job CHAR (15) NULL,

enter\_date DATE NULL,

CONSTRAINT prim\_works PRIMARY KEY(emp\_no, project\_no),

CONSTRAINT foreign1\_works FOREIGN KEY(emp\_no) REFERENCES employee(emp\_no),

CONSTRAINT foreign2\_works FOREIGN KEY(project\_no) REFERENCES project(project\_no));

INCREASED FUNCTIONALITY FINAL

CREATE DATABASE Projects;

CREATE TABLE department(dept\_no CHAR(4) NOT NULL,

dept\_name CHAR(25) NOT NULL,

location CHAR(30) NULL,

CONSTRAINT prim\_dept PRIMARY KEY (dept\_no));

CREATE TABLE employee (emp\_no INTEGER NOT NULL,

emp\_fname CHAR(20) NOT NULL,

emp\_lname CHAR(20) NOT NULL,

dept\_no CHAR(4) NULL,

CONSTRAINT prim\_emp PRIMARY KEY (emp\_no),

CONSTRAINT foreign\_emp FOREIGN KEY(dept\_no) REFERENCES department(dept\_no));

CREATE TABLE project (project\_no CHAR(4) NOT NULL,

project\_name CHAR(15) NOT NULL,

budget FLOAT NULL,

CONSTRAINT prim\_proj PRIMARY KEY (project\_no));

CREATE TABLE works\_on (emp\_no INTEGER NOT NULL,

project\_no CHAR(4) NOT NULL,

job CHAR (15) NULL,

enter\_date DATE NULL,

CONSTRAINT prim\_works PRIMARY KEY(emp\_no, project\_no),

CONSTRAINT foreign1\_works FOREIGN KEY(emp\_no) REFERENCES employee(emp\_no),

CONSTRAINT foreign2\_works FOREIGN KEY(project\_no) REFERENCES project(project\_no));

INSERT INTO department VALUES ('d1', 'Research', 'Dallas');

INSERT INTO department VALUES ('d2', 'Accounting', 'Seattle');

INSERT INTO department VALUES ('d3', 'Marketing', 'Dallas');

INSERT INTO employee VALUES (25348, 'Matthew', 'Smith','d3');

INSERT INTO employee VALUES (10102, 'Ann', 'Jones','d3');

INSERT INTO employee VALUES (18316, 'John', 'Barrimore', 'd1');

INSERT INTO employee VALUES (29346, 'James', 'James', 'd2');

INSERT INTO employee VALUES (9031, 'Elsa', 'Bertoni', 'd2');

INSERT INTO employee VALUES (2581, 'Elke', 'Hansel', 'd2');

INSERT INTO employee VALUES (28559, 'Sybill', 'Moser', 'd1');

INSERT INTO project VALUES ('p1', 'Apollo', 120000.00);

INSERT INTO project VALUES ('p2', 'Gemini', 95000.00);

INSERT INTO project VALUES ('p3', 'Mercury', 186500.00);

INSERT INTO works\_on VALUES (10102,'p1', 'Analyst', '2006.10.1');

INSERT INTO works\_on VALUES (10102, 'p3', 'Manager', '2008.1.1');

INSERT INTO works\_on VALUES (25348, 'p2', 'Clerk', '2007.2.15');

INSERT INTO works\_on VALUES (18316, 'p2', NULL, '2007.6.1');

INSERT INTO works\_on VALUES (29346, 'p2', NULL, '2006.12.15');

INSERT INTO works\_on VALUES (2581, 'p3', 'Analyst', '2007.10.15');

INSERT INTO works\_on VALUES (9031, 'p1', 'Manager', '2007.4.15');

INSERT INTO works\_on VALUES (28559, 'p1', NULL, '2007.8.1');

INSERT INTO works\_on VALUES (28559, 'p2', 'Clerk', '2008.2.1');

INSERT INTO works\_on VALUES (9031, 'p3', 'Clerk', '2006.11.15');

INSERT INTO works\_on VALUES (29346, 'p1','Clerk', '2007.1.4');

--Queries Begin Here---

--IF THEN LOGIC BASIC--

--from the worksOn table get how manyemployees are currently working on each project

SELECT COUNT(emp\_no) numOfEmps, project\_no FROM works\_on GROUP BY project\_no;

--if there is 4 or more employees working for project P1, then display a message "we have 4 or more employees on this project"

IF(SELECT COUNT(\*) FROM works\_on WHERE project\_no = 'P1') >= 4 PRINT 'Number of Employees in P1 is 4 or more';

--if there is 4 or more employees working for project P1, then display a message "we have 4 or more employees on this project"

--otherwise, get employees first and lastNames

IF(SELECT COUNT(\*) FROM works\_on WHERE project\_no = 'P1') > 4 PRINT 'Number of Employees in P1 is 4 or more' ELSE

BEGIN SELECT emp\_fname, emp\_lname FROM works\_on w JOIN employee e ON w.emp\_no = e.emp\_no WHERE project\_no = 'P1' END;

--VARIABLES

--declare, varName, var\_dataType, set data to it

DECLARE @projNo char(5);

SET @projNo = 'P2';

SELECT COUNT(\*) projNoCount FROM works\_on WHERE project\_no = @projNo;

--CASE Statement

--with projects that has a budget less than $100,000 discount it by 5%, otherwise discount it by 10%

SELECT \* FROM project;

SELECT project\_no, project\_name, Budget, CASE WHEN budget < 100000 THEN budget \* .95 ELSE budget \* .90 END budgetUpdate FROM project;

--STORED PROCEDURE

--stored procedures are built and excuted by us. Function smust return values, procedures it may not. Stored procedures are best for repetetive tasks.

--they are routines that we use, it does not (have to) return or retain a value, to accomplish a task.

SELECT \* FROM works\_on;

--write a procedure to update a jobTitle of an employee

CREATE OR ALTER PROCEDURE changeJobTitle (@empNum int, @job char(20)) AS UPDATE works\_on SET job = @job WHERE emp\_no = @empNum;--works despite red squigg

EXEC changeJobTitle 2581, 'DBA';

EXEC changeJobTitle 9031, 'Analyst';

SELECT \* FROM works\_on;

--write a procedure to get the location of a particular dept Name

SELECT \* FROM department;

CREATE OR ALTER PROCEDURE getDeptLocation (@deptName varchar(30)) AS SELECT location FROM department WHERE dept\_name = @deptName;

--test the procedure

EXEC getDeptLocation 'Research';

SELECT \* FROM project;

--write a procedure to find the budget for a particular project name

CREATE OR ALTER PROCEDURE getBudget (@projNa varchar(20)) AS SELECT budget FROM project WHERE project\_name = @projNa;

--test the proceduer

EXEC getbudget 'Apollo';

---ENDED HERE----

--Continued Stored Procedures

-- create procedure named increaseBudget to increase the budget for all projects by a percentage

CREATE OR ALTER PROCEDURE increaseBudget (@percent decimal) AS UPDATE project SET budget = budget + (budget \* @percent); -- to default a percentage use (@percent decimal = 0.3)

--Saleh's latter portion was: budget + budget \* @percent / 100

--query to see if it works:

EXEC increaseBudget 0.10;

--Create procdure named employees\_in\_dept to retrieve employees ID, firstName and lastname working for a particular dept.

CREATE OR ALTER PROCEDURE employees\_in\_dept (@deptNum char(2)) AS SELECT emp\_no, emp\_fname, emp\_lname FROM employee e WHERE e.dept\_no = @deptNum;

--query to see if it works:

EXEC employees\_in\_dept 'd1';

EXEC employees\_in\_dept d1; -- single quotes not required!

employees\_in\_dept d1 WITH RESULT SETS (

([Employee Number] int NOT NULL, --this is creating your own column names, this is only for representation, displaying the results.

[Employee firstName] char(20) NOT NULL, --you could use a view to put this in.

[Employee LastName] char(20) NOT NULL)

);

--create a procedure 'projects\_with\_more\_than\_one\_employee' select projectNumber and ProjectNames that have move than one employee working in it

CREATE OR ALTER PROCEDURE projects\_with\_more\_than\_one\_employee AS SELECT p.project\_no, p.project\_name FROM project p JOIN works\_on w ON p.project\_no = w.project\_no

WHERE 1 < (SELECT COUNT(emp\_no) FROM works\_on) GROUP BY p.project\_no, p.project\_name;

EXEC projects\_with\_more\_than\_one\_employee;

projects\_with\_more\_than\_one\_employee WITH RESULT SETS(

(

[projectNumber] char(4) NOT NULL,

[projectName] char(20) NOT NULL

)

);

--Saleh's Answer

CREATE OR ALTER PROCEDURE projects\_with\_more\_than\_one\_employee AS SELECT p.project\_no, p.project\_name FROM project p JOIN works\_on w ON p.project\_no = w.project\_no

GROUP BY p.project\_no, p.project\_name HAVING COUNT(emp\_no) > 1;

--insert a new project P4 science4, with budget = 155000, through a procedure called insertNewProject

CREATE OR ALTER PROCEDURE insertNewProject (@projectNum char(4), @projectname char(15), @budget float) AS INSERT INTO project VALUES(@projectNum, @projectname, @budget);

EXEC insertNewProject p4, science4, 155000;

SELECT \* FROM project;

--Increased Query Functionality and Controlling Data Flow

--Problem 1--

--Query to show important information of the bicycles

SELECT ProductID, Name, Style, Size FROM Production.Product WHERE ProductID >758 AND ProductID < 770;

--IF-Then Query to count the total products under Road-650 (by ProductID)

IF(SELECT COUNT(DISTINCT ProductID) FROM Production.Product WHERE ProductID > 758 AND ProductID < 770) > 10

PRINT 'There are more than 10 styles' ELSE PRINT'There are less than 10 styles';

--Query to verify the previous is correct.

SELECT COUNT(DISTINCT ProductID) distinctStyles FROM Production.Product WHERE ProductID > 758 AND ProductID < 770;

--Problem 2--

--DECLARE @bikeColor nvarchar(20)

--SET @bikeColor = 'black'

SELECT \* FROM Production.Product WHERE Color = @bikeColor AND ProductID >758 AND ProductID < 770;

--Problem 3--

DECLARE @bikeColor nvarchar(20)

SET @bikeColor = 'black'

SELECT p.ListPrice, i.Quantity, l.Name, \* FROM Production.Product p JOIN Production.ProductInventory i ON p.ProductID = i.ProductID

JOIN Production.Location l ON i.LocationID = l.LocationID WHERE Color = @bikeColor AND p.ProductID >758 AND p.ProductID < 770 AND l.Name = 'Finished Goods Storage';

--Problem 4--

SELECT p.ListPrice, i.Quantity, l.Name, CASE WHEN i.Quantity > 100 THEN ListPrice - ListPrice \* 0.10 ELSE ListPrice - ListPrice \* 0.05 END SalesPrice FROM Production.Product p

JOIN Production.ProductInventory i ON p.ProductID = i.ProductID JOIN Production.Location l ON i.LocationID = l.LocationID ORDER BY SalesPrice DESC;

--Problem 5--

CREATE OR ALTER PROCEDURE GetRegion (@city nvarchar(30)) AS SELECT t.Name, a.City, p.Name FROM Sales.SalesTerritory t JOIN Person.StateProvince p ON t.TerritoryID = p.TerritoryID

JOIN Person.Address a ON p.StateProvinceID = a.StateProvinceID WHERE a.City = @city;

EXEC GetRegion 'Berlin'