---------- SQL Lab ----------

----- Blake Biskner ---------

----- 2.0 SQL Queries -------

---------- 2.1 Select --------

-- Q.2.1.1 Select all records from Employee Table

SELECT \* FROM Employee;

-- Q.2.1.2 Select all records from Employee table where last name is King

SELECT \* FROM Employee

WHERE LastName='King';

-- Q.2.1.3 Select all records from Employee table where first name is Andrew and REPORTSTO is NULL

SELECT \* FROM Employee

WHERE FirstName='Andrew' AND ReportsTo IS NULL;

---------- 2.2 ORDER BY ----------

-- Q.2.2.1 Select all albums in Album table and sort result set in decreasing order by title

SELECT \* FROM Album ORDER BY title DESC;

-- Q.2.2.2 Select first name from Cutomer and sort result set in ascending order by city

SELECT FirstName FROM Customer ORDER BY City;

---------- 2.3 Insert Into ----------

-- Q.2.3.1 Insert two new records into the Genre table

INSERT INTO Genre

VALUES (26, 'Book Adaptation');

INSERT INTO Genre

VALUES (27, 'Online Broadcast');

-- Q.2.3.2 Insert two new records into the Employee table

INSERT INTO Employee

VALUES(9,'Thomas','Tim','Sales Manager',1,'10-JAN-80','04-OCT-11','1234 Another Rd','Tampa','FL','USA','1234','+1 (123)123-1231','+1 (123)123-4567','d@gmail.com');

INSERT INTO Employee

VALUES(10,'Robert','Big','IT Manager',1,'11-JAN-90','10-MAR-12','4567 Still Ave','Tampa','FL','USA','5678','+1 (012)345-6789','+1 (901)234-5678','e@gmail.com');

-- Q.2.3.3 Insert two new records into the Customer table

INSERT INTO Customer

VALUES(60,'Ron','Bob','Revature','123 Some St','Tampa','FL','USA','123456','(123)123-4567',NULL,'b@gmail.com',1);

INSERT INTO Customer

VALUES(61,'Tom','Robertson','Revature','456 Other St','Tampa','FL','USA','789012','(890)123-4567','FAX','c@gmail.com',2);

---------- 2.4 Update ----------

-- Q.2.4.1 Update Aaron Mitchell in Customer Table to Robert Walter

UPDATE Customer SET FirstName='Robert',LastName='Walter'

WHERE FirstName='Aaron' AND LastName='Mitchell';

-- Q.2.4.2 Update name of artist in the Artist Table "Creedence Clearwater Revival" to "CCR"

UPDATE Artist SET Name='CCR'

WHERE Name='Creedence Clearwater Revival';

---------- 2.5 Like ----------

-- Q.2.5.1 Select all invoices with a billing address like "T%"

SELECT \* FROM Invoice

WHERE BillingAddress LIKE 'T%';

---------- 2.6 Between ----------

-- Q.2.6.1 Select all invoices that have a total between 15 and 50

SELECT \* FROM Invoice

WHERE Total BETWEEN 15 AND 50;

-- Q.2.6.2 Select all employees hired between 1 of June 2003 and 1 March 2004

SELECT \* FROM Employee

WHERE HireDate BETWEEN '01-JUN-03' AND '01-MAR-2004';

COMMIT;

---------- 2.7 Delete ----------

-- Q.2.7.1 Delete a record in Cutomer table where the name is Robert Walter

-- 1 Alter the Invoice Table to remove constraints

-- Remove Foreign Key

ALTER TABLE Invoice

DROP CONSTRAINT FK\_INVOICECUSTOMERID;

-- Allow Null

ALTER TABLE Invoice

DROP CONSTRAINT SYS\_C005240;

-- 2 Determine the id of Robert Walter

SELECT CustomerId FROM Customer WHERE FirstName='Robert' AND LastName='Walter';

-- 3 Set the "foregin key" which refer to Robert Walter to null (id=32)

UPDATE Invoice SET CustomerId=NULL WHERE CustomerId=32;

-- 4 Reapply Foreign Key (values can now be null in column)

ALTER TABLE Invoice

ADD CONSTRAINT FK\_INVOICECUSTOMERID

FOREIGN KEY(CustomerId) REFERENCES Customer(CustomerId);

---------- 3.0 SQL Functions ----------

---------- 3.1 System Defined Functions ----------

-- Q.3.1.1 Create a function that returns the current time

CREATE OR REPLACE FUNCTION get\_time

RETURN VARCHAR2

IS

time\_now VARCHAR2(40);

BEGIN

SELECT TO\_CHAR(SYSDATE,'HH24:MI:SS') INTO time\_now FROM DUAL;

RETURN time\_now;

END;

/

-- Anonymous function to execute function

DECLARE

time\_now VARCHAR2(40);

BEGIN

time\_now:=get\_time;

DBMS\_OUTPUT.PUT\_LINE('The Current Time is '||time\_now);

END;

/

-- Q.3.1.2 Create a function that returns the length of a mediatype from the mediatype table

CREATE OR REPLACE FUNCTION get\_media\_length(media\_id INT)

RETURN INT

IS

media\_name VARCHAR2(40);

type\_len INT;

BEGIN

SELECT Name INTO media\_name FROM MediaType WHERE MediaTypeId=media\_id;

type\_len:=LENGTH(media\_name);

RETURN type\_len;

END;

/

-- Anonymous function to execute function

DECLARE

media\_id INT;

media\_name VARCHAR2(40);

type\_len INT;

BEGIN

media\_id:=1;

SELECT Name INTO media\_name FROM MediaType WHERE MediaTypeId=media\_id;

type\_len:=get\_media\_length(media\_id);

DBMS\_OUTPUT.PUT\_LINE('The Length of '|| media\_name ||' is '|| type\_len);

END;

/

---------- 3.2 System Defined Aggregate Functions ----------

-- Q.3.2.1 Create a function that returns the average total of all invoices

CREATE OR REPLACE FUNCTION get\_avg\_invoice

RETURN NUMBER

IS

avg\_invoice NUMBER(10,2);

BEGIN

SELECT AVG(Total) INTO avg\_invoice FROM Invoice;

RETURN avg\_invoice;

END;

/

-- Anonymous function to execute get\_avg\_invoice

DECLARE

avg\_invoice\_total NUMBER(10,2);

BEGIN

avg\_invoice\_total:=get\_avg\_invoice;

DBMS\_OUTPUT.PUT\_LINE('The Average Total of the Invoices is $'||avg\_invoice\_total);

END;

/

SELECT \* FROM Track;

-- Q.3.2.2 Create a function that returns the most expensive track

-- Create a table in which to store track names

CREATE OR REPLACE TYPE TRACKNAME IS TABLE OF VARCHAR2(100);

/

-- Iterate through each record with the highest unit price

CREATE OR REPLACE FUNCTION get\_expensive\_track

RETURN TRACKNAME

IS

track\_price NUMBER(5,2);

expensive\_track VARCHAR2(100);

lctr INT;

track\_table TRACKNAME:=TRACKNAME();

BEGIN

SELECT MAX(UnitPrice) INTO track\_price FROM Track;

lctr:=1;

FOR track\_itr IN (SELECT Name FROM Track WHERE UnitPrice=track\_price)

LOOP

track\_table.EXTEND();

track\_table(lctr):=track\_itr.Name;

lctr:=lctr+1;

END LOOP;

RETURN track\_table;

END;

/

SELECT \* FROM TABLE(get\_expensive\_track);

---------- 3.3 User Defined Scalar Functions ----------

-- Q.3.3.1 Create a function that returns the average price of invoiceline items in the invoicelines tables

CREATE OR REPLACE FUNCTION get\_avg\_invoiceline

RETURN NUMBER

IS

avg\_price NUMBER(5,2);

BEGIN

SELECT AVG(UnitPrice) INTO avg\_price FROM InvoiceLine;

RETURN avg\_price;

END;

/

-- Anonymous function to execute get\_avg\_invoiceline

DECLARE

avg\_price NUMBER(5,2);

BEGIN

avg\_price:=get\_avg\_invoiceline();

DBMS\_OUTPUT.PUT\_LINE('Average price of invoiceline is '||avg\_price);

END;

/

---------- 3.4 User Defined Table Valued Functions ----------

-- Q.3.4.1 Create a function that returns all employees who are born after 1968

-- Create new datatype which will store employee names

CREATE OR REPLACE TYPE EMPLOYEE\_NAME IS TABLE OF VARCHAR2(100);

/

-- Create function to return table of all young employees

CREATE OR REPLACE FUNCTION get\_young\_emp

RETURN

EMPLOYEE\_NAME

IS

CURSOR emps IS SELECT EmployeeId FROM Employee WHERE BirthDate>TO\_DATE(1968,'YYYY');

emp\_table EMPLOYEE\_NAME:=EMPLOYEE\_NAME();

itr INT;

BEGIN

itr:=1;

FOR emp\_id IN emps

LOOP

emp\_table.EXTEND();

emp\_table(itr):=emp\_id.EmployeeId;

itr:=itr+1;

END LOOP;

RETURN emp\_table;

END;

/

SELECT \* FROM TABLE(get\_young\_emp);

---------- 4.0 Stored Procedures ----------

---------- 4.1 Basic Stored Procedures ----------

-- Q.4.1.1 Create a stored procedure that selects the first and last names of all employees

CREATE OR REPLACE PROCEDURE get\_emp\_name(emp\_cursor OUT SYS\_REFCURSOR)

AS

BEGIN

OPEN emp\_cursor FOR SELECT FirstName, LastName FROM Employee;

END;

/

-- Anonymous function to execute procedure

DECLARE

emp\_cur SYS\_REFCURSOR;

first\_name VARCHAR2(20);

last\_name VARCHAR2(20);

BEGIN

get\_emp\_name(emp\_cur);

LOOP

FETCH emp\_cur INTO first\_name,last\_name;

EXIT WHEN emp\_cur%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(first\_name||' '||last\_name);

END LOOP;

CLOSE emp\_cur;

END;

/

-- Q.4.2.2 Create a stored procedure that returns managers of an employee

CREATE OR REPLACE PROCEDURE get\_emp\_boss(emp\_id IN OUT INT, emp\_boss\_cursor OUT SYS\_REFCURSOR)

AS

BEGIN

OPEN emp\_boss\_cursor FOR SELECT ReportsTo FROM Employee WHERE EmployeeID=emp\_id;

END;

/

-- Anonymos function to execute get\_emp\_boss

DECLARE

emp\_boss\_cur SYS\_REFCURSOR;

emp\_boss INT;

emp\_id INT;

BEGIN

emp\_id:=5;

LOOP

get\_emp\_boss(emp\_id, emp\_boss\_cur);

FETCH emp\_boss\_cur INTO emp\_boss;

-- If the returned value of reportsto is not null

IF emp\_boss IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE(emp\_boss);

-- Change the employee id to the reportsto value to examine the next level of management

emp\_id:=emp\_boss;

END IF;

EXIT WHEN emp\_boss IS NULL;

END LOOP;

CLOSE emp\_boss\_cur;

END;

/

---------- 5.0 Transactions ----------

-- Q.5.1 Create a transaction that given an invoice id will delete that invoice

CREATE OR REPLACE PROCEDURE delete\_invoice(

invoice\_id IN INT)

AS

BEGIN

DELETE Invoice WHERE InvoiceId=invoice\_id;

-- Update all foreign keys with value of invoice\_id to be null (as the parent key no longer exists)

UPDATE InvoiceLine SET InvoiceId=NULL WHERE InvoiceId=invoice\_id;

COMMIT;

END;

/

-- Anonymous function to execute stored procedure

ALTER TABLE InvoiceLine

DROP CONSTRAINT FK\_INVOICELINEINVOICEID;

ALTER TABLE InvoiceLine

DROP CONSTRAINT SYS\_C005245;

DECLARE

invoice\_id INT;

BEGIN

invoice\_id:=1;

delete\_invoice(invoice\_id);

END;

/

ALTER TABLE InvoiceLine

ADD CONSTRAINT FK\_INOVOICELINEINVOICEID

FOREIGN KEY(InvoiceId) REFERENCES Invoice(InvoiceId);

SELECT InvoiceId FROM InvoiceLine;

-- Q.5.2. Create a transaction nested within a stored procedure that inserts a new record in the customer table

CREATE OR REPLACE PROCEDURE insert\_customer(

cust\_id IN INT,

cust\_first IN VARCHAR2,

cust\_last IN VARCHAR2,

cust\_comp IN VARCHAR2,

cust\_address IN VARCHAR2,

cust\_city IN VARCHAR2,

cust\_state IN VARCHAR2,

cust\_country IN VARCHAR2,

cust\_postal IN VARCHAR2,

cust\_phone IN VARCHAR2,

cust\_fax IN VARCHAR2,

cust\_email IN VARCHAR2,

cust\_support IN INT)

AS

BEGIN

INSERT INTO Customer(

CustomerId,

FirstName,

LastName,

Company,

Address,

City,

State,

Country,

PostalCode,

Phone,

Fax,

Email,

SupportRepId)

VALUES(

cust\_id,

cust\_first,

cust\_last,

cust\_comp,

cust\_address,

cust\_city,

cust\_state,

cust\_country,

cust\_postal,

cust\_phone,

cust\_fax,

cust\_email,

cust\_support

);

COMMIT;

END;

/

SELECT \* FROM Customer;

-- Anonymous function to execute procedure

DECLARE

cust\_id INT;

cust\_first VARCHAR2(100);

cust\_last VARCHAR2(100);

cust\_comp VARCHAR2(100);

cust\_address VARCHAR2(100);

cust\_city VARCHAR2(100);

cust\_state VARCHAR2(5);

cust\_country VARCHAR2(20);

cust\_postal VARCHAR2(20);

cust\_phone VARCHAR2(100);

cust\_fax VARCHAR2(100);

cust\_email VARCHAR2(100);

cust\_support INT;

BEGIN

cust\_id:=62;

cust\_first:='Todd';

cust\_last:='Dylon';

cust\_comp:='Revature';

cust\_address:='123 Another St';

cust\_city:='Tampa';

cust\_state:='FL';

cust\_country:='USA';

cust\_postal:='1';

cust\_phone:='+1 (123)123-1234';

cust\_fax:='+1 (123)123-1234';

cust\_email:='h@gmail.com';

cust\_support:=3;

insert\_customer(

cust\_id,

cust\_first,

cust\_last,

cust\_comp,

cust\_address,

cust\_city,

cust\_state,

cust\_country,

cust\_postal,

cust\_phone,

cust\_fax,

cust\_email,

cust\_support);

END;

/

---------- 6.0 Triggers ----------

---------- 6.1 After For Trigger ----------

-- Q.6.1.1 Create an after insert trigger on the employees table after a new record is inserted

CREATE OR REPLACE TRIGGER after\_emp\_insert

AFTER INSERT ON Employee

FOR EACH ROW

BEGIN

DBMS\_OUTPUT.PUT\_LINE('New row inserted');

END;

/

-- Trigger test

INSERT INTO Employee (EmployeeId,LastName,FirstName) VALUES(20,'Mark','Chase');

DELETE Employee WHERE EmployeeId=20;

-- Q.6.1.2 Create an after update trigger on the album table after a row is updated in the table

CREATE OR REPLACE TRIGGER after\_album\_update

AFTER UPDATE ON Album

FOR EACH ROW

DECLARE

old\_id INT;

old\_title VARCHAR2(200);

old\_art INT;

new\_id INT;

new\_title VARCHAR2(200);

new\_art INT;

BEGIN

old\_id:=:OLD.AlbumId;

old\_title:=:OLD.Title;

old\_art:=:OLD.ArtistId;

new\_id:=:NEW.AlbumId;

new\_title:=:NEW.Title;

new\_art:=:NEW.ArtistId;

DBMS\_OUTPUT.PUT\_LINE(old\_id||' '||old\_title||' '||old\_art||' Changed to '||new\_id||' '||new\_title||' '||new\_art);

END;

/

-- Test Trigger

UPDATE Album SET Title='Big Ones Update' WHERE Title='Big Ones';

UPDATE Album SET Title='Big Ones' WHERE Title='Big Ones Update';

-- Q.6.1.3 Create an after trigger on the customer table that fires after a row is deleted

CREATE OR REPLACE TRIGGER after\_cust\_delete

AFTER DELETE ON Customer

FOR EACH ROW

DECLARE

cust\_id INT;

BEGIN

cust\_id:=:OLD.CustomerId;

DBMS\_OUTPUT.PUT\_LINE('Customer '||cust\_id||' Deleted');

END;

/

-- Test Trigger

DELETE FROM Customer WHERE CustomerId=61;

---------- 7.0 Joins ----------

---------- 7.1 Inner Joins ----------

-- Q.7.1.1 Create an inner join that joins customers and orders and specifies the name of the customer and the invoice id

SELECT c.FirstName, c.LastName, i.InvoiceId

FROM Customer c

INNER JOIN Invoice i ON c.CustomerId=i.CustomerId;

---------- 7.2 Outer Join ----------

-- Q.7.2.1 Create an outer join that joins the customer and invoice table

-- and specifies the customer id, first name, last name, invoice id, and total

SELECT c.CustomerId, c.FirstName, c.LastName, i.InvoiceId, i.Total

FROM Customer c

FULL OUTER JOIN Invoice i ON c.CustomerId=i.CustomerId;

---------- 7.3 Right Join ----------

-- Q.7.3.1 Create a right join that joins album and artist specifying artist name and title

SELECT art.Name, alb.Title

FROM Artist art

RIGHT OUTER JOIN Album alb ON art.ArtistId=alb.ArtistId;

---------- 7.5 Self Join ----------

-- Q.7.5.1 Perform a self join on the employee table joining on the reports to column

SELECT emp.FirstName, boss.FirstName

FROM Employee emp

LEFT OUTER JOIN Employee boss ON emp.ReportsTo=boss.EmployeeId;

SELECT \* FROM Employee;