**select DBMS LAB ASSIGNMENT**

**Assignment SQL – I**

1. **Create the following Tables -**

1.1 Customer master table: **Cust**

**Column Name Format Remarks**

cust\_id varchar2(3) Primary key, not-null

lname varchar2(15)

fname varchar2(15)

area varchar2(2)

phone\_no number(8)

1.2. Movies master table : **Movie**

**Column Name Format Remarks**

mv\_no varchar(2) Primary key not null

title varchar2(25)

type varchar2(10)

star varchar2(25)

1.3. Invoice transaction table : **Invoice**

**Column Name Format Remarks**

Inv\_no varchar2(3) primary key not null

Mv\_no varchar(2)

Cust\_id varchar2(3)

Issue\_date date

Return\_date date

1.4. Add the following constraints

* INVOICE(Cust\_id) references CUST(Cust\_id)
* INVOICE(Mv\_no) references MOVIE(Mv\_no)
* Declare NOT NULL : lname, fname, title, type

1.5. Add a new column PRICE in Movie table with data type number(8,2)

**Assignment SQL – II**

2. Insert the following data into the respective tables :

2.1. **Data for Cust table**

**Cust\_id Lname fname Area Phone\_no**

a01 Bayross Ivan sa 6125467

a02 Saitwal Vandana mu 5560379

a03 Jaguste Pramada da 4563891

a04 Navindgi Basu ba 6125401

a05 Sreedharan Ravi va -

a06 - Rukmini gh 5125274

2.2. **Data for Movie table**

**mv\_no title type star price**

1 bloody vengeance action jackie chan 180.95

2 the firm thriller tom cruise 200.00

3 pretty woman romance richard gere 150.55

4 home alone comedy macaulay culkin 150.00

5 the fugitive thriller harisson ford 200.00

6 coma suspense michael douglas 100.00

7 dracula horror gary oldman 150.25

8 quick change comedy bill muray 100.00

9 gone with the wind drama clarke gable 200.00

10 carry on doctor comedy leslie phillips 100.00

2.3. **Data for Invoice table**

**inv\_no mv\_no cust\_id issue\_date return\_date**

i01 4 a01 23-july-93 25-jul-93

i02 3 a02 12-aug-93 15-aug-93

i03 1 a02 15-aug-93 18-aug-93

i04 6 a03 10-sep-93 12-sep-93

i05 7 a04 05-aug-93 08-aug-93

i06 2 a06 18-sep-93 21-sep-93

i07 9 a05 07-jul-93 10-jul-93

i08 9 a01 11-aug-93 14-aug-93

i09 5 a03 06-jul-93 07-jul-93

i10 8 a06 03-sep-93 06-sep-93

**Assignment SQL – III**

**3. Write SQL statements to retrieve these query :**

3.1. Find out the names of all the customers.

3.2. Print the entire customer table.

3.3. Retrieve the list of fname and the area of all the customers.

3.4. List the various movie types available from the movie table.

3.5. Print the information of invoice table in the following format for all records

A)The Invoice No. of Customer Id. {cust - id} is {inv - no} and Movie No. is {mv - no}.

B){cust-id} has taken Movie No. {mv-no} on {issue-date} and will return on (return\_date).

3.6. Change the telephone number of prarnada to 466389.

3.7. Change the issue - date of cust- id 'A01' to 24/07/93.

3.8. Change the price of 'gone with the wind' to Rs. 250. 00.

3.9. Delete the record with invoice number 'l 08' from the invoice table.

3.10. Delete all the records having return date before 10th July'93

3.11. Change the area of cust – id 'A05' to 'vs'.

3.12. Change the return date of invoice number 'I08' to 16-08-93.

3.13. Find the names of all customers having 'a' as the second letter in their fnames.

3.14. Find the lnames of all customers that begin with 's ' or 'j' .

3.15. Find out the customers who stay in an area whose second letter is 'a'.

3.16. Find the list of all customers who stay in area 'da' or area 'mu' or area 'gh'.

3.17. Print the list of employees whose phone numbers are greater than the value 5550000.

3.18. Print the information from invoice table of customers who have been issued movies in the month of September.

3.19. Display the invoice table information for cust – id 'a01' and 'a02'.

3.20. Find the movies of type 'action' and ‘comedy’.

3.21. Find the movies whose price is greater than 150 and less than or equal to 200.

3.22. Find the movies that cost more than 150 and also find the new cost as original cost \* 15.

3.23. Rename the new column in the above query as new-price.

3.24. List the movies in sorted order of their titles.

3.25. Print the names and types of all the movie except horror movies.

3.26. Divide the cost of movie 'home alone' by difference between its price and 100.

3.27. List the names, areas and cust - id of customers without phone numbers.

3.28. List the names of customers without lname.

3.29. List the mv - no, title, type of movies whose stars begin with letter 'm'.

3.30. List the mv-no and inv-no of customers having inv-no less than 'i05'from the Invoice Transaction Table.

**Assignment SQL – IV**

4.1. Calculate the square root of the price of each movie.

4.2. Count the total number of customers.

4.3. Calculate the total price of all the movies.

4.4 Calculate the average price of all the movies.

4.5. Determine the maximum and minimum movie prices. Rename the title as max-price and min\_price respectively.

4.6. Count the number of movies having price greater than or equal to 150.

4.7. Print the type and average price of each movie.

4.8. Find the number of movies in each type.

4.9. Count separately the number of movies in the 'comedy' and 'thriller' types.

4.10. Calculate the average price for each type that has a maximum price of 150.00.

4.11. Calculate the average price of all movies where type is 'comedy' or 'thriller' and price is greater than or equal to 150.00.

4.12. Display the invoice number and day on which customers were issued movies.

4.13. Display the month (in alphabets) in which customers are supposed to return the movies.

4.14. Display the 15 days after the issue-date in the format 'dd-month-yy'.

For eg. 12-february-93.

4.15. Find the number of days elapsed between the current date and the return date of the movie for all customers.

**Assignment SQL – V**

5.1. Find out the movie number which has been issued to 'ivan'.

5.2. Find the names and movie numbers of all the customers who have been issued a movie.

5.3. Select the title, cust - id, mv - no for all the movies that are issued.

5.4. Find out the title and types of the movies that have been issued to 'Vandana'.

5.5. Find the names of customers who have been issued movie of type 'drama'.

5.6. Display the title, lname, fname for customers having movie number greater than or equal to three, in the following format:

The movie taken by (fname} {lname} is {title}.

5.7. Find out which customers have been issued movie number 9.

5.8. Find the customer name and area with invoice number 'i10'.

5.9. Find the customer names and phone numbers who have been issued movies before the month of August.

5.10. Find the name of the movie issued to 'vandana' and 'ivan'.

5.11. List the movie number, movie names issued to all customers.

5.12. Find the type and movie number of movie issued to cust – id 'a01' and 'a02'.

5.13. Find out if the movie starring' tom cruise' is issued to any customer and print the custid to whom it is issued.

5.14. Find the lname, fname who have been issued movies.

5.15. Find the name of customer whose has not issued any movie

**Assignment SQL – VI**

6.1. Create a view with custno, fname, lname of cust table.

6.2. Retrieve all customers name from cust table.

6.3. Create another view for those customers who have been issued movies.

6.4. Create a table with two columns odd number, even number to store odd and even numbers between 1 to 100.

6.5. Generate odd and even numbers using sequences.

6.6. Create a sequence for the column mv\_no of Movie table to generate movie no with initial value 10, maximum value 100 and incremented by 1.

6.7. Generate mv\_no with sequence above in Movie table.

6.8. Create an index on price column of movie table and retrieve the indexed data.

**Assignment SQL – VII**

7.1. Create a user with your name

7.2. Give select, insert, update permission on cust, movie table of user ‘scott’ to your user account

7.3. Assign only insert permission on invoice table of user ‘scott’ to your current account.

7.4. Check the permission has properly granted using system tables.

7.5. Write a PL/SQL block to print the message “Welcome to PL/SQL” on the screen.

7.6. Write a PL/SQL block to find the summation of given two numeric values.

7.7 Write a PL/SQL to find out the customer id and name who has issued a movie and their elapsed date between issue date and return date is less than two days

7.8. Write a PL/SQL block to find the summation of two numeric values using function where two numeric values will be passed into function as parameters.

7.9. Create an insert trigger to check the price of the each movie must not be greater than 500.

7.10. Add a new column: total\_issued in the Movie table to count how many copies has issued to each movie. Create a insert trigger on invoice to increment the values of Total\_issued whenever a new rows inserted into the invoice table.

EXPERIMENT 1:-

CREATE TABLE Cust (

cust\_id VARCHAR2(3) PRIMARY KEY NOT NULL,

lname VARCHAR2(15),

fname VARCHAR2(15),

area VARCHAR2(2),

phone\_no NUMBER(8)

);

CREATE TABLE Movie (

mv\_no varchar2(2) Primary key not null,

title varchar2(25),

type varchar2(10),

star varchar2(25)

);

CREATE TABLE Invoice (

inv\_no VARCHAR2(3) PRIMARY KEY NOT NULL,

mv\_no VARCHAR2(2),

cust\_id VARCHAR2(3),

issue\_date DATE,

return\_date DATE,

CONSTRAINT fk\_mv\_no FOREIGN KEY (mv\_no) REFERENCES Movie(mv\_no),

CONSTRAINT fk\_cust\_id FOREIGN KEY (cust\_id) REFERENCES Cust(cust\_id)

);

ALTER TABLE Movie

ADD price NUMBER(8,2);

select \* from Movie;

EXPERIMENT 2:-

INSERT INTO Cust (Cust\_id, Lname, Fname, Area, Phone\_no) VALUES ('a01', 'Bayross', 'Ivan', 'sa', '6125467');

INSERT INTO Cust (Cust\_id, Lname, Fname, Area, Phone\_no) VALUES ('a02', 'Saitwal', 'Vandana', 'mu', '5560379');

INSERT INTO Cust (Cust\_id, Lname, Fname, Area, Phone\_no) VALUES ('a03', 'Jaguste', 'Pramada', 'da', '4563891');

INSERT INTO Cust (Cust\_id, Lname, Fname, Area, Phone\_no) VALUES ('a04', 'Navindgi', 'Basu', 'ba', '6125401');

INSERT INTO Cust (Cust\_id, Lname, Fname, Area, Phone\_no) VALUES ('a05', 'Sreedharan', 'Ravi', 'va', NULL);

INSERT INTO Cust (Cust\_id, Lname, Fname, Area, Phone\_no) VALUES ('a06', NULL, 'Rukmini', 'gh', '5125274');

INSERT INTO Movie (mv\_no, title, type, star, price) VALUES (1, 'bloody vengeance', 'action', 'jackie chan', 180.95);

INSERT INTO Movie (mv\_no, title, type, star, price) VALUES (2, 'the firm', 'thriller', 'tom cruise', 200.00);

INSERT INTO Movie (mv\_no, title, type, star, price) VALUES (3, 'pretty woman', 'romance', 'richard gere', 150.55);

INSERT INTO Movie (mv\_no, title, type, star, price) VALUES (4, 'home alone', 'comedy', 'macaulay culkin', 150.00);

INSERT INTO Movie (mv\_no, title, type, star, price) VALUES (5, 'the fugitive', 'thriller', 'harisson ford', 200.00);

INSERT INTO Movie (mv\_no, title, type, star, price) VALUES (6, 'coma', 'suspense', 'michael douglas', 100.00);

INSERT INTO Movie (mv\_no, title, type, star, price) VALUES (7, 'dracula', 'horror', 'gary oldman', 150.25);

INSERT INTO Movie (mv\_no, title, type, star, price) VALUES (8, 'quick change', 'comedy', 'bill muray', 100.00);

INSERT INTO Movie (mv\_no, title, type, star, price) VALUES (9, 'gone with the wind', 'drama', 'clarke gable', 200.00);

INSERT INTO Movie (mv\_no, title, type, star, price) VALUES (10, 'carry on doctor', 'comedy', 'leslie phillips', 100.00);

INSERT INTO Invoice (inv\_no, mv\_no, cust\_id, issue\_date, return\_date) VALUES ('i01', 4, 'a01', '1993-07-23', '1993-07-25');

INSERT INTO Invoice (inv\_no, mv\_no, cust\_id, issue\_date, return\_date) VALUES ('i02', 3, 'a02', '1993-08-12', '1993-08-15');

INSERT INTO Invoice (inv\_no, mv\_no, cust\_id, issue\_date, return\_date) VALUES ('i03', 1, 'a02', '1993-08-15', '1993-08-18');

INSERT INTO Invoice (inv\_no, mv\_no, cust\_id, issue\_date, return\_date) VALUES ('i04', 6, 'a03', '1993-09-10', '1993-09-12');

INSERT INTO Invoice (inv\_no, mv\_no, cust\_id, issue\_date, return\_date) VALUES ('i05', 7, 'a04', '1993-08-05', '1993-08-08');

INSERT INTO Invoice (inv\_no, mv\_no, cust\_id, issue\_date, return\_date) VALUES ('i06', 2, 'a06', '1993-09-18', '1993-09-21');

INSERT INTO Invoice (inv\_no, mv\_no, cust\_id, issue\_date, return\_date) VALUES ('i07', 9, 'a05', '1993-07-07', '1993-07-10');

INSERT INTO Invoice (inv\_no, mv\_no, cust\_id, issue\_date, return\_date) VALUES ('i08', 9, 'a01', '1993-08-11', '1993-08-14');

INSERT INTO Invoice (inv\_no, mv\_no, cust\_id, issue\_date, return\_date) VALUES ('i09', 5, 'a03', '1993-07-06', '1993-07-07');

INSERT INTO Invoice (inv\_no, mv\_no, cust\_id, issue\_date, return\_date) VALUES ('i10', 8, 'a06', '1993-09-03', '1993-09-06');

EXPERIMENT 3:-

select fname , lname from Cust;

select \* from Cust

select fname , area from Cust;\

select type from Movie;

SELECT

'The Invoice No. of Customer Id. ' || cust\_id || ' has taken movie on ' ||TO\_CHAR(issue\_date, 'DD-MON-YYYY')|| ' and will return on ' || TO\_CHAR(return\_date,'DD-MON-YYYY')

FROM Invoice;

UPDATE Cust

SET Phone\_no = '466389'

WHERE Fname = 'Pramada';

select \* from Invoice

UPDATE Invoice

SET issue\_date = TO\_DATE('24-07-1993', 'DD-MM-YYYY')

WHERE cust\_id = 'a01'

UPDATE Movie

SET price = '250.00'

WHERE title = 'gone with the wind'

DELETE FROM Invoice where inv\_no ='i08';

INSERT INTO Invoice (inv\_no, mv\_no,cust\_id,issue\_date, return\_date) VALUES('i08',4,'a08',TO\_DATE('10-09-1993' ,'DD-MM-YYYY') , TO\_DATE('17-09-1993' , 'DD-MM-YYYY'))

INSERT INTO Invoice (inv\_no, mv\_no, cust\_id, issue\_date, return\_date) VALUES ('i10', 9, 'a10', TO\_DATE('11-AUG-1993', 'DD-MON-YYYY'), TO\_DATE('14-AUG-1993', 'DD-MON-YYYY'));

select fname , lname from Cust where fname LIKE '\_a%'

SELECT lname

FROM Cust

WHERE lname LIKE 's%' OR lname LIKE 'j%';

SELECT \*

FROM Cust

WHERE Area IN ('da', 'mu', 'gh');

SELECT \*

FROM Cust

WHERE Phone\_no > 5550000;

SELECT \* FROM Invoice

WHERE TO\_CHAR(issue\_date,'MM') = '09';

SELECT \* FROM Invoice

WHERE issue\_date > TO\_DATE('31-AUG-1993', 'DD-MON-YYYY');

SELECT \*

FROM Invoice

WHERE cust\_id IN ('a01', 'a02');

SElECT \* FROM Movie WHERE

type IN ('action', 'comedy')

SELECT \*

FROM Movie

WHERE price > 150 AND price <= 200;

SELECT \*,

price \* 15 AS new\_price

FROM Movie

WHERE price > 150;

SELECT title, type, star, price, price \* 15 AS new\_price

FROM Movie

WHERE price > 150;

SELECT \*

FROM Movie

ORDER BY title DESC;

SELECT title, type

FROM Movie

WHERE type != 'horror';

SELECT title, price / (price - 100) AS cost\_divided

FROM Movie

WHERE title = 'home alone';

SELECT Fname, Lname, Area, Cust\_id

FROM Cust

WHERE Phone\_no IS NULL;

SELECT mv\_no, inv\_no

FROM Invoice

WHERE inv\_no < 'i05';

EXPERIMENT 4:-

SELECT title, type, price, SQRT(price)

FROM Movie;

SELECT COUNT(\*) as total

FROM Cust;

SELECT SUM(price) AS total\_price

FROM Movie;

SELECT AVG(price) as avg\_price from Movie;

SELECT MAX(price) AS max\_price, MIN(price) AS min\_price

FROM Movie;

SELECT COUNT(\*) AS movies\_ge\_150

FROM Movie

WHERE price >= 150;

SELECT type, AVG(price) AS average\_price

FROM Movie

GROUP BY type;

select type , count(\*) as total from Movie GROUP BY type

SELECT type, COUNT(\*) AS num\_movies

FROM Movie

WHERE type IN ('comedy', 'thriller')

GROUP BY type;

select type , avg(price) as average

from Movie

where price>=150

group by type;

SELECT type , AVG(price) AS average\_price

FROM Movie

WHERE type IN ('comedy', 'thriller') AND price >= 150.00

group by type;

SELECT inv\_no, TO\_CHAR(issue\_date, 'MONTH') AS issue\_day

FROM Invoice;

SELECT inv\_no, TO\_CHAR(return\_date, 'MM') AS return\_month

FROM Invoice;

SELECT inv\_no, TO\_CHAR(issue\_date + 15, 'DD-Month-YY') AS due\_date

FROM Invoice;

SELECT inv\_no, return\_date, ROUND(SYSDATE - return\_date) AS days\_elapsed

FROM Invoice;

EXPERIMENT 5:-

SELECT i.mv\_no

FROM Invoice i

JOIN Cust c ON i.cust\_id = c.cust\_id

WHERE c.fname = 'Ivan';

SELECT c.fname, i.mv\_no From Cust c

Join Invoice i on i.cust\_id = c.cust\_id ;

SELECT m.title, i.cust\_id, i.mv\_no

FROM Movie m

JOIN Invoice i ON m.mv\_no = i.mv\_no;

SELECT m.title, m.type

FROM Movie m

JOIN Invoice i ON m.mv\_no = i.mv\_no

JOIN Cust c ON i.cust\_id = c.cust\_id

WHERE c.fname = 'Vandana';

SELECT c.fname

FROM Cust c

Join Invoice i on i.cust\_id = c.cust\_id

JOIN Movie m ON m.mv\_no = i.mv\_no

WHERE m.type = 'drama';

SELECT c.fname

FROM Cust c

JOIN Invoice i ON c.cust\_id = i.cust\_id

WHERE i.mv\_no = 9;

SELECT c.fname, c.lname, c.phone\_no

FROM Cust c

JOIN Invoice i ON c.cust\_id = i.cust\_id

WHERE TO\_CHAR(i.issue\_date, 'MM') < '08';

EXPERIMENT 7:-

CREATE VIEW CustView AS

SELECT cust\_id AS custno, fname, lname

FROM Cust;

SELECT fname, lname

FROM Cust;

CREATE VIEW IssuedCustomers AS

SELECT DISTINCT c.cust\_id, c.fname, c.lname

FROM Cust c

JOIN Invoice i ON c.cust\_id = i.cust\_id;

CREATE TABLE Numbers (

odd\_number INT,

even\_number INT

);

CREATE SEQUENCE OddSeq

START WITH 1

INCREMENT BY 2

MAXVALUE 99;

CREATE SEQUENCE EvenSeq

START WITH 2

INCREMENT BY 2

MAXVALUE 100;

BEGIN

FOR i IN 1..50 LOOP

INSERT INTO Numbers (odd\_number, even\_number)

VALUES (OddSeq.NEXTVAL, EvenSeq.NEXTVAL);

END LOOP;

END;

CREATE SEQUENCE MovieSeq

START WITH 10

INCREMENT BY 1

MAXVALUE 100;

INSERT INTO Movie (mv\_no, title, type, star, price)

VALUES (MovieSeq.NEXTVAL, 'new\_movie\_title', 'new\_movie\_type', 'new\_movie\_star', new\_movie\_price);

CREATE INDEX idx\_price ON Movie (price);

-- Retrieve the indexed data

SELECT \*

FROM Movie

WHERE price > 0

ORDER BY price;

EXPERIMENT 7:-

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Welcome to PL/SQL');

END;

/

DECLARE

num1 NUMBER := 10;

num2 NUMBER := 20;

sum NUMBER;

BEGIN

sum := num1 + num2;

DBMS\_OUTPUT.PUT\_LINE('Summation: ' || TO\_CHAR(sum));

END;

/

DECLARE

CURSOR c1 IS

SELECT c.cust\_id, c.fname, c.lname, i.issue\_date, i.return\_date

FROM scott.Cust c

JOIN scott.Invoice i ON c.cust\_id = i.cust\_id

WHERE (i.return\_date - i.issue\_date) < 2;

cust\_id scott.Cust.cust\_id%TYPE;

fname scott.Cust.fname%TYPE;

lname scott.Cust.lname%TYPE;

BEGIN

OPEN c1;

LOOP

FETCH c1 INTO cust\_id, fname, lname;

EXIT WHEN c1%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Customer ID: ' || cust\_id || ', Name: ' || fname || ' ' || lname);

END LOOP;

CLOSE c1;

END;

/

SET SERVEROUTPUT ON;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Welcome to PL/SQL');

END;

/

DECLARE

num1 NUMBER := 10;

num2 NUMBER := 20;

sum NUMBER;

BEGIN

sum := num1 + num2;

DBMS\_OUTPUT.PUT\_LINE('Summation: ' || SUM);

END;

/

DECLARE

CURSOR c1 IS

SELECT c.cust\_id, c.fname, c.lname, i.issue\_date, i.return\_date

FROM scott.Cust c

JOIN scott.Invoice i ON c.cust\_id = i.cust\_id

WHERE (i.return\_date - i.issue\_date) < 2;

cust\_id scott.Cust.cust\_id%TYPE;

fname scott.Cust.fname%TYPE;

lname scott.Cust.lname%TYPE;

BEGIN

OPEN c1;

LOOP

FETCH c1 INTO cust\_id, fname, lname;

EXIT WHEN c1%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Customer ID: ' || cust\_id || ', Name: ' || fname || ' ' || lname);

END LOOP;

CLOSE c1;

END;

/

CREATE OR REPLACE FUNCTION sum\_two\_numbers(num1 IN NUMBER, num2 IN NUMBER) RETURN NUMBER IS

BEGIN

RETURN num1 + num2;

END;

/

DECLARE

result NUMBER;

BEGIN

result := sum\_two\_numbers(10, 20);

DBMS\_OUTPUT.PUT\_LINE('Summation: ' || result);

END;

/

CREATE OR REPLACE TRIGGER check\_movie\_price

BEFORE INSERT OR UPDATE ON scott.Movie

FOR EACH ROW

BEGIN

IF :NEW.price > 500 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Price of the movie must not be greater than 500');

END IF;

END;

/

ALTER TABLE scott.Movie ADD (total\_issued NUMBER DEFAULT 0);

CREATE OR REPLACE TRIGGER increment\_total\_issued

AFTER INSERT ON scott.Invoice

FOR EACH ROW

BEGIN

UPDATE scott.Movie

SET total\_issued = total\_issued + 1

WHERE mv\_no = :NEW.mv\_no;

END;

/