**PART – A**

# EXPERIMENT 1

## Consider the following schema for a Library Database:

**BOOK(Book\_id, Title, Publisher\_Name, Pub\_Year) BOOK\_AUTHORS(Book\_id, Author\_Name) PUBLISHER(Name, Address, Phone) BOOK\_COPIES(Book\_id, Branch\_id, No-of\_Copies)**

**BOOK\_LENDING(Book\_id, Branch\_id, Card\_No, Date\_Out, Due\_Date) LIBRARY\_BRANCH(Branch\_id, Branch\_Name, Address)**

**Write SQL queries to**

1. **Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc.**
2. **Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017.**
3. **Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.**
4. **Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.**
5. **Create a view of all books and its number of copies that are currently available in the Library**

***Solution :***

### *Table creation:*

CREATE TABLE PUBLISHER

(

);

NAME CHAR(15), ADDRESS VARCHAR(20), PHONE VARCHAR(10), PRIMARY KEY(NAME)

CREATE TABLE LIBRARY\_BRANCH

(

); CREATE TABLE BORROWERS

(

);

BRANCH\_ID VARCHAR(5), BRANCH\_NAME CHAR(5), ADDRESS VARCHAR(20), PRIMARY KEY(BRANCH\_ID)

CARD\_NO VARCHAR(10), NAME CHAR(15), ADDRESS VARCHAR(20), PHONE VARCHAR(10), PRIMARY KEY(CARD\_NO)

CREATE TABLE BOOK

(

BOOK\_ID VARCHAR(5), TITLE CHAR(20),

PUBLISHER\_NAME CHAR(15), PUB\_YEAR INT,

(BOOK\_ID),

FOREIGN KEY(PUBLISHER\_NAME)

REFERENCES PUBLISHER(NAME) ON DELETE CASCADE ) PARTITION BY RANGE(PUB\_YEAR)

(PARTITION T1 VALUES LESS THAN(2016), PARTITION T2 VALUES LESS THAN(2018)

);

CREATE TABLE BOOK\_AUTHORS

(

);

CREATE TABLE BOOK\_COPIES

(

);

BOOK\_ID VARCHAR(5), AUTHOR\_NAME CHAR(15), PRIMARY KEY(BOOK\_ID), FOREIGN KEY(BOOK\_ID) REFERENCES BOOK(BOOK\_ID) ON DELETE CASCADE

BOOK\_ID VARCHAR(5), BRANCH\_ID VARCHAR(5), NO\_OF\_COPIES INT,

PRIMARY KEY(BOOK\_ID,BRANCH\_ID), FOREIGN KEY(BOOK\_ID) REFERENCES BOOK(BOOK\_ID)

ON DELETE CASCADE, FOREIGN KEY(BRANCH\_ID)

REFERENCES LIBRARY\_BRANCH(BRANCH\_ID) ON DELETE CASCADE

CREATE TABLE BOOK\_LENDING

(

BOOK\_ID VARCHAR(5), BRANCH\_ID VARCHAR(5),

CARD\_NO VARCHAR(10), DUE\_OUT DATE, DUE\_DATE DATE,

PRIMARY KEY(BOOK\_ID,BRANCH\_ID,CARD\_NO), FOREIGN KEY(BOOK\_ID)

REFERENCES BOOK(BOOK\_ID) ON DELETE CASCADE, FOREIGN KEY(BRANCH\_ID)

REFERENCES LIBRARY\_BRANCH(BRANCH\_ID) ON DELETE CASCADE,

FOREIGN KEY(CARD\_NO) REFERENCES BORROWERS(CARD\_NO) ON DELETE CASCADE

);

INSERT INTO PUBLISHER VALUES('&NAME','&ADDRESS','&PHONE');

INSERT INTO LIBRARY\_BRANCH VALUES('&BRANCH\_ID','&BRANCH\_NAME','&ADDRESS'); INSERT INTO BORROWERS VALUES('&CARD\_NO','&NAME','&ADDRESS','&PHONE'); INSERT INTO BOOK VALUES('&BOOK\_ID','&TITLE','&PUBLISHER\_NAME','&PUB\_YEAR'); INSERT INTO BOOK\_AUTHORS VALUES('&BOOK\_ID','&AUTHOR\_NAME');

INSERT INTO BOOK\_COPIES VALUES('&BOOK\_ID','&BRANCH\_ID','&NO\_OF\_COPIES');

INSERT INTO BOOK\_LENDINGVALUES('&BOOK\_ID','&BRANCH\_ID','&CARD\_NO','&DUE\_OUT','&DUE\_DATE');

select table\_name from dba\_tables where owner ='15IS048'; SET LINESIZE 120 PAGESIZE 2000;

## QUERY 1:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*----@@@@@@@@@@@@@@@@@@@@ SQL> select a.book\_id,a.title,a.publisher\_name,b.author\_name,c.branch\_id,c.no\_of\_copies

* 1. from book a,book\_authors b,book\_copies c
  2. where a.book\_id=b.book\_id and a.book\_id=c.book\_id;

BOOK\_ TITLE PUBLISHER\_NAME AUTHOR\_NAME BRANC NO\_OF\_COPIES

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 611 | MANAGEMENT | OXFORD | JOHN E | 03 | 20 |
| 611 | MANAGEMENT | OXFORD | JOHN E | 01 | 10 |
| 614 | JAVA | NAROSA | PETER LINZ | 05 | 50 |
| 616 | DBMS | PEARSON | C K NAGPAL | 05 | 30 |
| 617 | CO | PEARSON | RAJKUMAR | 03 | 40 |

SELECT CARD\_NO,NAME,ADDRESS,PHONE FROM BORROWERS WHERE CARD\_NO IN (SELECT DISTINCT CARD\_NO FROM BOOK\_LENDING

WHERE DUE\_OUT BETWEEN '01-JAN-2017' AND '30-JUN-2017' GROUP BY(CARD\_NO) HAVING COUNT(DUE\_OUT)>3);

SQL> UPDATE BOOK\_LENDING SET CARD\_NO='1011' WHERE BOOK\_ID=615;

2 rows updated.

SQL> SELECT \* FROM BOOK\_LENDING;

BOOK\_ BRANC CARD\_NO DUE\_OUT DUE\_DATE

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

|  |  |  |  |
| --- | --- | --- | --- |
| 614 | 05 | 1011 | 12-AUG-15 12-AUG-16 |
| 616 | 02 | 1011 | 26-JAN-11 26-JAN-16 |
| 611 | 03 | 1014 | 15-MAR-15 15-APR-16 |
| 615 | 04 | 1011 | 19-DEC-16 16-JAN-17 |
| 615 | 01 | 1011 | 27-MAR-97 27-OCT-99 |

SQL> UPDATE BOOK\_LENDING SET DUE\_OUT='20-JAN-2017' WHERE CARD\_NO=1011;

4 rows updated.

SQL> UPDATE BOOK\_LENDING SET DUE\_OUT='20-JUN-2017' WHERE CARD\_NO='1011';

4 rows updated.

SQL> SELECT \* FROM BOOK\_LENDING;

BOOK\_ BRANC CARD\_NO DUE\_OUT DUE\_DATE

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

|  |  |  |  |
| --- | --- | --- | --- |
| 614 | 05 | 1011 | 20-JUN-17 12-AUG-16 |
| 616 | 02 | 1011 | 20-JUN-17 26-JAN-16 |
| 611 | 03 | 1014 | 15-MAR-15 15-APR-16 |
| 615 | 04 | 1011 | 20-JUN-17 16-JAN-17 |
| 615 | 01 | 1011 | 20-JUN-17 27-OCT-99 |

## QUERY 2:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*----@@@@@@@@@@@@@@@@@@@@ SQL> SELECT CARD\_NO,NAME,ADDRESS,PHONE FROM BORROWERS WHERE CARD\_NO IN

1. (SELECT DISTINCT CARD\_NO FROM BOOK\_LENDING
2. WHERE DUE\_OUT BETWEEN '01-JAN-2017' AND '30-JUN-2017' GROUP BY(CARD\_NO)
3. HAVING COUNT(DUE\_OUT)>3);

CARD\_NO NAME ADDRESS PHONE

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

1011 SHARIQE MANGALORE 9964999012

SQL> SELECT CARD\_NO,NAME,ADDRESS,PHONE FROM BORROWERS WHERE CARD\_NO IN

1. (SELECT DISTINCT CARD\_NO FROM BOOK\_LENDING
2. WHERE DUE\_OUT BETWEEN '01-JAN-2017' AND '30-JUN-2017' GROUP BY(CARD\_NO)
3. HAVING COUNT(\*)>3);

CARD\_NO NAME ADDRESS PHONE

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

1011 SHARIQE MANGALORE 9964999012

## QUERY 3:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*----@@@@@@@@@@@@@@@@@@@@ SQL> SELECT \* FROM BOOK;

BOOK\_ TITLE PUBLISHER\_NAME PU

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

|  |  |  |
| --- | --- | --- |
| 612 | ATCI | CENGAGE |
| 613 | CN | OXFORD |
| 614 | JAVA | NAROSA |
| 615 | CLOUD | CENGAGE |
| 616 | DBMS | PEARSON |
| 617 | CO | PEARSON |

6 rows selected.

SQL> SELECT \* FROM BOOK\_COPIES;

BOOK\_ BRANC NO\_OF\_COPIES

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

|  |  |  |
| --- | --- | --- |
| 616 | 05 | 30 |
| 617 | 03 | 40 |
| 614 | 05 | 50 |

SQL> SELECT \* FROM BOOK\_LENDING;

BOOK\_ BRANC CARD\_NO DUE\_OUT DUE\_DATE

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

|  |  |  |  |
| --- | --- | --- | --- |
| 614 | 05 | 1011 | 20-JUN-17 12-AUG-16 |
| 616 | 02 | 1011 | 20-JUN-17 26-JAN-16 |
| 615 | 04 | 1011 | 20-JUN-17 16-JAN-17 |
| 615 | 01 | 1011 | 20-JUN-17 27-OCT-99 |

SQL> DELETE FROM BOOK WHERE BOOK\_ID='614';

1 row deleted.

SQL> SELECT \* FROM BOOK\_COPIES;

BOOK\_ BRANC NO\_OF\_COPIES

----- ----- ------------ 616 05 30

617 03 40

SQL> SELECT \* FROM BOOK\_LENDING;

BOOK\_ BRANC CARD\_NO DUE\_OUT DUE\_DATE

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

|  |  |  |  |
| --- | --- | --- | --- |
| 616 | 02 | 1011 | 20-JUN-17 26-JAN-16 |
| 615 | 04 | 1011 | 20-JUN-17 16-JAN-17 |
| 615 | 01 | 1011 | 20-JUN-17 27-OCT-99 |

## QUERY 4:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*----@@@@@@@@@@@@@@@@@@@@ SQL> INSERT INTO BOOK 2 VALUES('&BOOK\_ID','&TITLE','&PUBLISHER\_NAME','&PUB\_YEAR');

Enter value for book\_id: 511 Enter value for title: RAT

Enter value for publisher\_name: CAT Enter value for pub\_year: 2018

old 2: VALUES('&BOOK\_ID','&TITLE','&PUBLISHER\_NAME','&PUB\_YEAR')

new 2: VALUES('511','RAT','CAT','2018')

INSERT INTO BOOK

\*

ERROR at line 1:

ORA-14400: inserted partition key does not map to any partition

## QUERY 5:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*----@@@@@@@@@@@@@@@@@@@@ CREATE VIEW VV AS

SELECT A.BOOK\_ID,A.TITLE,B.BRANCH\_ID,B.NO\_OF\_COPIES FROM BOOK A,BOOK\_COPIES B

WHERE A.BOOK\_ID=B.BOOK\_ID; SELECT \* FROM VV;

BOOK\_ TITLE BRANC NO\_OF\_COPIES

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

|  |  |  |
| --- | --- | --- |
| 616 DBMS | 05 | 30 |
| 617 CO | 03 | 40 |

SQL> SELECT \* FROM BOOK\_COPIES; BOOK\_ BRANC NO\_OF\_COPIES

----- ----- ------------ 616 05 30

617 03 40

SQL> SELECT \* FROM LIBRARAY\_BRANCH; SELECT \* FROM LIBRARAY\_BRANCH

\*

ERROR at line 1:

ORA-00942: table or view does not exist SQL> SELECT \* FROM LIBRARY\_BRANCH;

BRANC BRANC ADDRESS

----- ----- -------------------- 01 ISE SAHYADRI

1. EC SRINIVAS
2. MECH JOSEPH
3. CIVIL BEARYS
4. CS PESIT

SQL> INSERT INTO BOOK\_COPIES 2 VALUES('&BOOK\_ID','&BRANCH\_ID','&NO\_OF\_COPIES');

Enter value for book\_id: 614 Enter value for branch\_id: 05 Enter value for no\_of\_copies: 10

old 2: VALUES('&BOOK\_ID','&BRANCH\_ID','&NO\_OF\_COPIES')

new 2: VALUES('614','05','10') INSERT INTO BOOK\_COPIES

\*

ERROR at line 1:

ORA-02291: integrity constraint (15IS048.SYS\_C008120) violated - p

SQL> INSERT INTO BOOK\_COPIES 2 VALUES('&BOOK\_ID','&BRANCH\_ID','&NO\_OF\_COPIES');

Enter value for book\_id: 613 Enter value for branch\_id: 05 Enter value for no\_of\_copies: 10

old 2: VALUES('&BOOK\_ID','&BRANCH\_ID','&NO\_OF\_COPIES')

new 2: VALUES('613','05','10')

1 row created.

SQL> SELCT ;L

SP2-0042: unknown command "SELCT ;L" - rest of line ignored. SQL> SELECT SUM(NO\_OF\_COPIES)

1. FROM BOOK\_COPIES
2. GROUP BY BRANCH\_ID;

SUM(NO\_OF\_COPIES)

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

40

40

SQL> SELECT BRANCH\_ID, SUM(NO\_OF\_COPIES) FROM BOOK\_COPIES

2 GROUP BY BRANCH\_ID;

BRANC SUM(NO\_OF\_COPIES)

----- ----------------- 05 40

03 40

SQL> SELECT BRANCH\_ID, SUM(NO\_OF\_COPIES) AS TOTAL

2 FROM BOOK\_COPIES GROUP BY BRANCH\_ID;

BRANC TOTAL

----- ---------- 05 40

03 40

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*----@@@@@@@@@@@@@@@@@@@@

# EXPERIMENT 2

## Consider the following schema for Order Database:

**SALESMAN(Salesman\_id, Name, City, Commission) CUSTOMER(Customer\_id, Cust\_Name, City, Grade, Salesman\_id) ORDERS(Ord\_No, Purchase\_Amt, Ord\_Date, Customer\_id, Salesman\_id) Write SQL queries to**

1. **Count the customers with grades above Bangalore’s average.**
2. **Find the name and numbers of all salesman who had more than one customer.**
3. **List all the salesman and indicate those who have and don’t have customers in their cities (Use**

**UNION operation.)**

1. **Create a view that finds the salesman who has the customer with the highest order of a day.**
2. **Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.**

### *Solution :*

***Table creation***

CREATE TABLE SALESMAN

(

); CREATE TABLE CUSTOMER

(

); CREATE TABLE ORDERS

(

SALESMAN\_ID VARCHAR(20), NAME CHAR(15),

CITY CHAR(15), COMMISSION INT,

PRIMARY KEY(SALESMAN\_ID)

CUSTOMER\_ID VARCHAR(20), CUST\_NAME CHAR(15),

CITY CHAR(15), GRADE FLOAT,

SALESMAN\_ID VARCHAR(20), PRIMARY KEY(CUSTOMER\_ID),

FOREIGN KEY(SALESMAN\_ID) REFERENCES SALESMAN(SALESMAN\_ID) ON DELETE CASCADE

ORDER\_NO VARCHAR(15), PURCHSE\_AMT INT, ORD\_DATE DATE, CUSTOMER\_ID VARCHAR(20), SALESMAN\_ID VARCHAR(20), PRIMARY KEY(ORDER\_NO), FOREIGN KEY(CUSTOMER\_ID)

REFERENCES CUSTOMER(CUSTOMER\_ID)

ON DELETE CASCADE,

FOREIGN KEY(SALESMAN\_ID) REFERENCES SALESMAN(SALESMAN\_ID) ON DELETE CASCADE

); INSERT INTO SALESMAN

VALUES('&SALESMAN\_ID','&NAME','&CITY','&COMMISSION'); INSERT INTO CUSTOMER

VALUES('&CUSTOMER\_ID','&CUST\_NAME','&CITY','&GRADE','&SALESMAN\_ID'); INSERT INTO ORDERS

VALUES('&ORDER\_NO','&PURCHASE\_AMT','&ORD\_DATE','&CUSTOMER\_ID','&SALESMAN\_ID'); SQL> SELECT \* FROM SALESMAN;

SALESMAN\_ID NAME CITY COMMISSION

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

|  |  |  |  |
| --- | --- | --- | --- |
| 1000 | RAVI | BANGALORE | 12 |
| 1001 | SOORAJ | DELHI | 20 |
| 1002 | PREM | LUCKNOW | 15 |
| 1003 | JOHN | BANGALORE | 20 |
| 1004 | RAJU | MYSORE | 18 |

SQL> SELECT \* FROM CUSTOMER;

CUSTOMER\_ID CUST\_NAME CITY GRADE SALESMAN\_ID

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

|  |  |  |  |
| --- | --- | --- | --- |
| C1 | SHERYL | BANGALORE | 4.5 1000 |
| C2 | DIYA | DELHI | 5 1000 |
| C3 | PRIYA | MUMBAI | 5.5 1001 |
| C4 | JACK | LUCKNOW | 4.5 1002 |
| C5 | JILL | BANGALORE | 9 1003 |

SQL> SELECT \* FROM ORDERS 2 ;

ORDER\_NO PURCHSE\_AMT ORD\_DATE CUSTOMER\_ID SALESMAN\_ID

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OR1 | 25000 | 25-MAY-17 | C1 | 1000 |
| OR2 | 15000 | 25-MAY-17 | C2 | 1000 |
| OR3 | 17000 | 25-MAY-17 | C5 | 1003 |
| OR4 | 30000 | 17-FEB-17 | C4 | 1002 |
| OR5 | 32000 | 17-FEB-17 | C3 | 1001 |
| OR6 | 14000 | 05-JUN-17 | C1 | 1000 |
| OR7 | 50000 | 10-JUL-17 | C1 | 1000 |

## QUERY1:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*----@@@@@@@@@@@@@@@@@@@ SELECT COUNT(CUSTOMER\_ID)FROM CUSTOMER WHERE GRADE>

(SELECT AVG(GRADE)FROM CUSTOMER WHERE CITY='BANGALORE'); COUNT(CUSTOMER\_ID)

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

1

## QUERY 2:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*----@@@@@@@@@@@@@@@@@@@ SELECT SALESMAN\_ID,NAME FROM SALESMAN WHERE SALESMAN\_ID IN

(SELECT SALESMAN\_ID FROM CUSTOMER GROUP BY SALESMAN\_ID HAVING COUNT(SALESMAN\_ID)>1);

SALESMAN\_ID NAME

-------------------- --------------- 1000 RAVI

## QUERY 3:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*---@@@@@@@@@@@@@@@@@@@@ (SELECT DISTINCT A.SALESMAN\_ID,A.NAME,A.CITY FROM SALESMAN A,CUSTOMER B WHERE A.SALESMAN\_ID=B.SALESMAN\_ID AND A.CITY=B.CITY)

UNION

(SELECT DISTINCT A.SALESMAN\_ID,A.NAME,A.CITY FROM SALESMAN A,CUSTOMER B WHERE A.SALESMAN\_ID=B.SALESMAN\_ID AND A.CITY!=B.CITY);

SALESMAN\_ID NAME CITY

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

|  |  |  |
| --- | --- | --- |
| 1000 | RAVI | BANGALORE |
| 1001 | SOORAJ | DELHI |
| 1002 | PREM | LUCKNOW |
| 1003 | JOHN | BANGALORE |

## QUERY 4:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*---@@@@@@@@@@@@@@@@@@@@

CREATE VIEW MAX\_ORDER AS

SELECT A.SALESMAN\_ID,A.NAME,B.ORD\_DATE FROM SALESMAN A,ORDERS B WHERE A.SALESMAN\_ID=B.SALESMAN\_ID AND

PURCHSE\_AMT=(SELECT MAX(PURCHSE\_AMT)FROM ORDERS C WHERE C.ORD\_DATE=B.ORD\_DATE);

View created.

SQL> SELECT \* FROM MAX\_ORDER; SALESMAN\_ID NAME ORD\_DATE

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

|  |  |  |
| --- | --- | --- |
| 1000 | RAVI | 25-MAY-17 |
| 1001 | SOORAJ | 17-FEB-17 |
| 1000 | RAVI | 05-JUN-17 |
| 1000 | RAVI | 10-JUL-17 |

## QUERY 5

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*---@@@@@@@@@@@@@@@@@@@@ SQL> SELECT \* FROM SALESMAN;

SALESMAN\_ID NAME CITY COMMISSION

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

|  |  |  |  |
| --- | --- | --- | --- |
| 1000 | RAVI | BANGALORE | 12 |
| 1001 | SOORAJ | DELHI | 20 |
| 1002 | PREM | LUCKNOW | 15 |
| 1003 | JOHN | BANGALORE | 20 |
| 1004 | RAJU | MYSORE | 18 |

SQL> SELECT \* FROM CUSTOMER;

CUSTOMER\_ID CUST\_NAME CITY GRADE SALESMAN\_ID

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

|  |  |  |  |
| --- | --- | --- | --- |
| C1 | SHERYL | BANGALORE | 4.5 1000 |
| C2 | DIYA | DELHI | 5 1000 |
| C3 | PRIYA | MUMBAI | 5.5 1001 |
| C4 | JACK | LUCKNOW | 4.5 1002 |
| C5 | JILL | BANGALORE | 9 1003 |

SQL> SELECT \*FROM OREDRS; SELECT \*FROM OREDRS

\*

ERROR at line 1:

ORA-00942: table or view does not exist SQL> SELECT \* FROM ORDERS;

ORDER\_NO PURCHSE\_AMT ORD\_DATE CUSTOMER\_ID SALESMAN\_ID

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OR1 | 25000 | 25-MAY-17 | C1 | 1000 |
| OR2 | 15000 | 25-MAY-17 | C2 | 1000 |
| OR3 | 17000 | 25-MAY-17 | C5 | 1003 |
| OR4 | 30000 | 17-FEB-17 | C4 | 1002 |
| OR5 | 32000 | 17-FEB-17 | C3 | 1001 |
| OR6 | 14000 | 05-JUN-17 | C1 | 1000 |
| OR7 | 50000 | 10-JUL-17 | C1 | 1000 |

7 rows selected.

SQL> DELETE FROM SALESMAN WHERE SALESMAN\_ID=1000;

1 row deleted.

SQL> SELECT \* FROM SALESMAN 2 ;

SALESMAN\_ID NAME CITY COMMISSION

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

|  |  |  |  |
| --- | --- | --- | --- |
| 1001 | SOORAJ | DELHI | 20 |
| 1002 | PREM | LUCKNOW | 15 |
| 1003 | JOHN | BANGALORE | 20 |
| 1004 | RAJU | MYSORE | 18 |

SQL> SELECT \* FROM CUSTOMER;

CUSTOMER\_ID CUST\_NAME CITY GRADE SALESMAN\_ID

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

|  |  |  |  |
| --- | --- | --- | --- |
| C3 | PRIYA | MUMBAI | 5.5 1001 |
| C4 | JACK | LUCKNOW | 4.5 1002 |
| C5 | JILL | BANGALORE | 9 1003 |

SQL> SLECT \* FROM ORDERS;

SP2-0734: unknown command beginning "SLECT \* FR..." - rest of line ignored.

SQL> SELECT \* FROM ORDERS;

ORDER\_NO PURCHSE\_AMT ORD\_DATE CUSTOMER\_ID SALESMAN\_ID

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

|  |  |  |
| --- | --- | --- |
| OR3 | 17000 25-MAY-17 C5 | 1003 |
| OR4 | 30000 17-FEB-17 C4 | 1002 |
| OR5 | 32000 17-FEB-17 C3 | 1001 |

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*---@@@@@@@@@@@@@@@@@@@@

# EXPERIMENT 3

## Consider the schema for Movie Database:

**ACTOR(Act\_id, Act\_Name, Act\_Gender) DIRECTOR(Dir\_id, Dir\_Name, Dir\_Phone) MOVIES(Mov\_id, Mov\_Title, Mov\_Year, Mov\_Lang, Dir\_id) MOVIE\_CAST(Act\_id, Mov\_id, Role)**

**RATING(Mov\_id, Rev\_Stars) Write SQL queries to**

1. **List the titles of all movies directed by ‘Hitchcock’.**
2. **Find the movie names where one or more actors acted in two or more movies.**
3. **List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).**
4. **Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.**
5. **Update rating of all movies directed by ‘Steven Spielberg’ to 5.**

### *Solution :*

***Table creation***

CREATE TABLE ACTOR

(

ACT\_ID VARCHAR(10), ACT\_NAME CHAR(20), ACT\_GENDER CHAR(6), PRIMARY KEY(ACT\_ID)

);

CREATE TABLE DIRECTOR

(

DIR\_ID VARCHAR(10), DIR\_NAME CHAR(20), DIR\_PHONE VARCHAR(10), PRIMARY KEY(DIR\_ID)

);

CREATE TABLE MOVIES

(

MOV\_ID VARCHAR(10), MOV\_TITLE VARCHAR(20), MOV\_YEAR INT, MOV\_LANG CHAR(10), DIR\_ID VARCHAR(10), PRIMARY KEY(MOV\_ID), FOREIGN KEY(DIR\_ID)

REFERENCES DIRECTOR(DIR\_ID) ON DELETE CASCADE

);

CREATE TABLE MOVIE\_CAST

(

ACT\_ID VARCHAR(10), MOV\_ID VARCHAR(10), ROLE CHAR(10),

PRIMARY KEY(ACT\_ID,MOV\_ID), FOREIGN KEY(ACT\_ID) REFERENCES ACTOR(ACT\_ID) ON DELETE CASCADE,

FOREIGN KEY(MOV\_ID) REFERENCES MOVIES(MOV\_ID) ON DELETE CASCADE

);

CREATE TABLE RATING

(

MOV\_ID VARCHAR(10), REV\_STARS INT, MOVIE\_REVIEWER VARCHAR(20),

PRIMARY KEY(MOV\_ID,REV\_STARS,MOVIE\_REVIEWER), FOREIGN KEY(MOV\_ID)

REFERENCES MOVIES(MOV\_ID) ON DELETE CASCADE

);

INSERT INTO ACTOR VALUES('&ACT\_ID','&ACT\_NAME','&ACT\_GENDER'); INSERT INTO DIRECTOR VALUES('&DIR\_ID','&DIR\_NAME','&DIR\_PHONE');

INSERT INTO MOVIES VALUES('&MOV\_ID','&MOV\_TITLE','&MOV\_YEAR','&MOV\_LANG','&DIR\_ID'); INSERT INTO MOVIE\_CAST VALUES('&ACT\_ID','&MOV\_ID','&ROLE');

INSERT INTO RATING VALUES('&MOV\_ID','&REV\_STARS','&MOVIE\_REVIEWER');

SQL> SELECT \* FROM ACTOR; ACT\_ID ACT\_NAME ACT\_GE

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

|  |  |  |
| --- | --- | --- |
| A1 | SHARUKH KHAN | MALE |
| A2 | SALMAN KHAN | MALE |
| A3 | AMIR KHAN | MALE |
| A4 | KAREENA KAPOOR | FEMALE |
| A5 | KAJOL | FEMALE |

SQL> SELECT \* FROM DIRECTOR;

DIR\_ID DIR\_NAME DIR\_PHONE

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

|  |  |  |
| --- | --- | --- |
| D1 | HITCHCOCK | 9945464951 |
| D2 | YASHRAJ | 9964586855 |
| D3 | ROHIT SHETTY | 7899744311 |
| D4 | STEVEN SPIELBERG | 9844347348 |

D5 MADHUR BHANDARKAR 9964999012 SQL> SELECT \* FROM MOVIES;

MOV\_ID MOV\_TITLE MOV\_YEAR MOV\_LANG DIR\_ID

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

|  |  |  |  |
| --- | --- | --- | --- |
| M1 | HEROES | 1990 ENGLISH | D4 |
| M2 | HORIZON | 2001 ENGLISH | D3 |
| M3 | VERTIGO | 2002 SPANISH | D1 |
| M4 | SPARROWS | 1995 HEBREW | D2 |
| M5 | BIRDS | 2017 ENGLISH | D1 |

SQL> SELECT \* FROM MOVIE\_CAST;

ACT\_ID MOV\_ID ROLE

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

|  |  |  |
| --- | --- | --- |
| A1 | M1 | FATHER |
| A2 | M2 | SISTER |
| A3 | M3 | HERO |
| A1 | M4 | HERO |
| A1 | M5 | SUPPORT |
| A1 | M2 | SUPPORT |

6 rows selected.

SQL> SELECT \* FROM RATING;

MOV\_ID REV\_STARS MOVIE\_REVIEWER

---------- ---------- -------------------- M1 7 BBC

M2 6 BBC

M3 7 RAM GOPAL

M4 8 RAM GOPAL

M5 9 TOI

## QUERY 1:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*---@@@@@@@@@@@@@@@@@@@@ SELECT A.MOV\_TITLE FROM MOVIES A,DIRECTOR B

WHERE A.DIR\_ID=B.DIR\_ID AND B.DIR\_NAME='HITCHCOCK';

MOV\_TITLE

-------------------- VERTIGO BIRDS

## QUERY 2:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*---@@@@@@@@@@@@@@@@@@@@ SELECT A.MOV\_TITLE FROM MOVIES A,MOVIE\_CAST B

WHERE B.MOV\_ID=A.MOV\_ID AND B.ACT\_ID IN

(SELECT ACT\_ID FROM MOVIE\_CAST GROUP BY ACT\_ID HAVING COUNT(\*)>=2);

MOV\_TITLE

-------------------- BIRDS SPARROWS HORIZON HEROES

## QUERY 3:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*---@@@@@@@@@@@@@@@@@@@@ (SELECT DISTINCT A.ACT\_NAME FROM ACTOR A

JOIN MOVIE\_CAST B ON A.ACT\_ID =B.ACT\_ID JOIN MOVIES C ON B.MOV\_ID=C.MOV\_ID

WHERE C.MOV\_YEAR<2000) INTERSECT

(SELECT DISTINCT A.ACT\_NAME FROM ACTOR A JOIN MOVIE\_CAST B ON A.ACT\_ID =B.ACT\_ID JOIN MOVIES C ON B.MOV\_ID=C.MOV\_ID

WHERE C.MOV\_YEAR>2015);

ACT\_NAME

-------------------- SHARUKH KHAN

## QUERY 4:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*---@@@@@@@@@@@@@@@@@@@@ SELECT A.MOV\_TITLE,MAX(R.REV\_STARS)FROM RATING R

JOIN MOVIES A ON A.MOV\_ID=R.MOV\_ID GROUP BY A.MOV\_TITLE ORDER BY A.MOV\_TITLE; MOV\_TITLE MAX(R.REV\_STARS)

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

|  |  |
| --- | --- |
| BIRDS | 9 |
| HEROES | 7 |
| HORIZON | 6 |
| SPARROWS | 8 |
| VERTIGO | 7 |

## QUERY 5:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*---@@@@@@@@@@@@@@@@@@@@ UPDATE RATING SET REV\_STARS=5 WHERE MOV\_ID IN

(SELECT MOV\_ID FROM MOVIES A,DIRECTOR D WHERE D.DIR\_NAME='STEVEN SPIELBERG' AND D.DIR\_ID=A.DIR\_ID);

MOV\_ID REV\_STARS MOVIE\_REVIEWER

---------- ---------- -------------------- M1 5 BBC

M2 6 BBC

M3 7 RAM GOPAL

M4 8 RAM GOPAL

M5 9 TOI

# EXPERIMENT 4

## Consider the schema for College Database:

**STUDENT(USN, SName, Address, Phone, Gender) SEMSEC(SSID, Sem, Sec)**

**CLASS(USN, SSID)**

**SUBJECT(Subcode, Title, Sem, Credits)**

**IAMARKS(USN, Subcode, SSID, Test1, Test2, Test3, FinalIA) Write SQL queries to**

1. **List all the student details studying in fourth semester ‘C’ section.**
2. **Compute the total number of male and female students in each semester and in each section.**
3. **Create a view of Test1 marks of student USN ‘1BI15CS101’ in all subjects.**
4. **Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.**
5. **Categorize students based on the following criterion:**

**If FinalIA = 17 to 20 then CAT = ‘Outstanding’ If FinalIA = 12 to 16 then CAT = ‘Average’**

**If FinalIA< 12 then CAT = ‘Weak’**

**Give these details only for 8th semester A, B, and C section students.**

### *Solution :*

***Table creation***

CREATE TABLE STUDENT

(

); CREATE TABLE SEMSEC

(

);

CREATE TABLE CLASS

(

);

USN VARCHAR(20), SNAME CHAR(15), ADDRESS VARCHAR(20), PHONE NUMBER(10), GENDER CHAR(6), PRIMARY KEY(USN)

SSID VARCHAR(20), SEM NUMBER(3), SEC CHAR(3), PRIMARY KEY(SSID)

USN VARCHAR(20), SSID VARCHAR(20),

PRIMARY KEY(USN,SSID), FOREIGN KEY(USN) REFERENCES STUDENT(USN) ON DELETE CASCADE, FOREIGN KEY(SSID) REFERENCES SEMSEC(SSID) ON DELETE CASCADE

CREATE TABLE SUBJECT

(

); CREATE TABLE IAMARKS

(

);

SUBCODE VARCHAR(15), TITLE CHAR(20),

SEM NUMBER(3), CREDITS INT,

PRIMARY KEY(SUBCODE)

USN VARCHAR(20), SUBCODE VARCHAR(15), SSID VARCHAR(20), TEST1 NUMBER(2), TEST2 NUMBER(2), TEST3 NUMBER(2), FINALIA NUMBER(2),

PRIMARY KEY(USN,SUBCODE,SSID), FOREIGN KEY(USN)

REFERENCES STUDENT(USN) ON DELETE CASCADE, FOREIGN KEY(SUBCODE)

REFERENCES SUBJECT(SUBCODE) ON DELETE CASCADE,

FOREIGN KEY(SSID) REFERENCES SEMSEC(SSID) ON DELETE CASCADE

INSERT INTO STUDENT VALUES('&USN','&SNAME','&ADDRESS','&PHONE','&GENDER'); INSERT INTO SEMSECVALUES('&SSID','&SEM','&SEC');

INSERT INTO CLASS VALUES('&USN','&SSID');

INSERT INTO SUBJECT VALUES('&SUBCODE','&TITLE','&SEM','&CREDITS');

INSERT INTO IAMARKS VALUES('&USN','&SUBCODE','&SSID','&TEST1','&TEST2','&TEST3','&FINALIA'); SQL> SELECT \* FROM STUDENT;

USN SNAME ADDRESS PHONE GENDER

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1BI15CS101 | SHARIQE | MANGALORE | 9964999012 | M |
| IS001 | SUHAIL | BANGALORE | 7899744311 | M |
| CS048 | SHAHANA | MYSORE | 9844347348 | F |
| 12CV057 | RAMNATH | OOTY | 9945464951 | M |
| 16CS103 | FARHEEN | DELHI | 9844984220 | F |
| 12EC095 | SHIFA | AGRA | 7844784550 | F |
| 14ME76 | FARZEEN | GOA | 9644984450 | M |
| 13IS023 | FARHAN | RAJASTHAN | 8866945321 | M |
| 10IS049 | NAZEER | PUNE | 9844651035 | M |
| 13CS092 | MUNAZZA | CHENNAI | 6514827981 | F |

10 rows selected.

SQL> SELECT \* FROM SEMSEC; SSID SEM SEC

-------------------- ---------- --- S1 3 A

S2 8 B

S3 8 C

S4 8 A

S5 4 C

S6 4 A

6 rows selected.

SQL> SELECT \* FROM CLASS;

USN SSID

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

|  |  |
| --- | --- |
| 13CS092 | S4 |
| 13IS023 | S2 |
| 16CS103 | S5 |
| 1BI15CS101 | S1 |
| 1BI15CS101 | S3 |
| CS048 | S3 |

6 rows selected.

SQL> SELECT \* FROM SUBJECT;

SUBCODE TITLE SEM CREDITS

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

|  |  |  |  |
| --- | --- | --- | --- |
| IS010 | UNIX | 3 | 4 |
| IS020 | DBMS | 4 | 4 |
| IS030 | OOMD | 8 | 3 |
| IS040 | JAVA | 8 | 2 |
| IS050 | FLAT | 5 | 4 |

SQL> SELECT \* FROM IAMARKS;

USN SUBCODE SSID TEST1 TEST2 TEST3 FINALIA

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 13CS092 | IS010 | S4 | 15 | 17 | 11 |
| 1BI15CS101 | IS030 | S3 | 19 | 20 | 17 |
| CS048 | IS020 | S3 | 11 | 8 | 14 |
| 13IS023 | IS010 | S2 | 14 | 8 | 10 |
| 16CS103 | IS030 | S5 | 5 | 10 | 15 |

## QUERY 1:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*---@@@@@@@@@@@@@@@@@@@@

SELECT A.\*,B.SEM,B.SEC FROM STUDENT A,SEMSEC B,CLASS C WHERE A.USN=C.USN

AND B.SSID=C.SSID AND B.SEM=4

AND B.SEC='C';

USN SNAME ADDRESS PHONE GENDER SEM SEC

-------------------- --------------- -------------------- ---------- ------ ---------- --- 16CS103 FARHEEN DELHI 9844984220 F 4 C

## QUERY 2:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*---@@@@@@@@@@@@@@@@@@@@ SELECT A.SEM,A.SEC,COUNT(

CASE WHEN B.GENDER='M'THEN A.SID END) AS MALECOUNT,

COUNT(CASE WHEN B.GENDER='F'THEN A.SID END)

AS FEMALECOUNT FROM STUDENT B,SEMSEC A,CLASS C

WHERE A.SSID=C.SSID AND B.USN=C.USN GROUP BY A.SEM,A.SEC;

## QUERY 3:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*---@@@@@@@@@@@@@@@@@@@@ CREATE VIEW STU\_MARKS\_TEST1 AS

SELECT SUBCODE,TEST1 FROM IAMARKS WHERE USN='1BI15CS101';

View created.

SQL>

SQL> SELECT \* FROM STU\_MARKS\_TEST1;

SUBCODE TEST1

--------------- ---------- IS030 19

## QUERY 4:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*----@@@@@@@@@@@@@@@@@@@ UPDATE IAMARKS S,(

SELECT USN,MAX(T1)T4 FROM(

SELECT USN,AVG((TEST1+TEST2)/2)T1 FROM STUDENT S1 GROUP BY USN UNION

SELECT USN,AVG((TEST2+TEST3)/2)T1 FROM STUDENT S2 GROUP BY USN UNION

SELECT USN,AVG((TEST3+TEST1)/2)T1 FROM STUDENT S3 GROUP BY USN )

AVGSCORS GROUP BY USN)UPD SET FINALIA=UPD.T4 WHERE S.USN=UPD.USN;

## QUERY 5:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*----@@@@@@@@@@@@@@@@@@@ SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER,

(CASE WHEN IA.FINALIA BETWEEN 17 AND 20 THEN 'OUTSTANDING' WHEN IA.FINALIA BETWEEN 12 AND 16 THEN 'AVERAGE'

ELSE 'WEAK' END )

AS CAT FROM STUDENT S,SEMSEC SS,IAMARKS IA,SUBJECT SU WHERE S.USN=IA.USN AND SS.SSID=IA.SSID

AND SU.SUBCODE=IA.SUBCODE AND SEC.SEM=8;

# EXPERIMENT 5

## Consider the schema for Company Database:

**EMPLOYEE(SSN, Name, Address, Sex, Salary, SuperSSN, DNo) DEPARTMENT(DNo, DName, MgrSSN, MgrStartDate) DLOCATION(DNo,DLoc)**

**PROJECT(PNo, PName, PLocation, DNo) WORKS\_ON(SSN, PNo, Hours)**

**Write SQL queries to**

1. **Make a list of all project numbers for projects that involve an employee whose last name is ‘Scott’,**

**either as a worker or as a manager of the department that controls the project.**

1. **Show the resulting salaries if every employee working on the ‘IoT’ project is given a 10 percent**

**raise**

1. **Find the sum of the salaries of all employees of the ‘Accounts’ department, as well as the**

**maximum salary, the minimum salary, and the average salary in this department**

1. **Retrieve the name of each employee who works on all the projects controlledby department number 5 (use NOT EXISTS operator).5. For each department that has more than five employees, retrieve the department number and the number of its employees who are making more than Rs. 6,00,000.**

### *Solution :*

***Table creation***

CREATE TABLE DEPARTMENT

(

); CREATE TABLE EMPLOYEE

(

); CREATE TABLE DLOCATION

(

DNO INT,

DNAME VARCHAR(20), MGR\_SSN VARCHAR(10), MGR\_START\_DATE DATE, PRIMARY KEY(DNO)

SSN VARCHAR(10), NAME VARCHAR(20), ADDRESS VARCHAR(20), SEX CHAR(1),

SALARY DECIMAL(10,3), SUPER\_SSN VARCHAR(10), DNO INT,

PRIMARY KEY(SSN), FOREIGN KEY(SUPER\_SSN)

REFERENCES EMPLOYEE(SSN) ON DELETE CASCADE, FOREIGN KEY(DNO)

REFERENCES DEPARTMENT(DNO) ON DELETE CASCADE

DNO INT,

DLOC VARCHAR(20),

); CREATE TABLE PROJECT

(

);

CREATE TABLE WORKS\_ON

(

);

PRIMARY KEY(DNO,DLOC), FOREIGN KEY(DNO)

REFERENCES DEPARTMENT(DNO) ON DELETE CASCAD

PNO INT,

PNAME VARCHAR(20), PLOCATION VARCHAR(20), DNO INT,

PRIMARY KEY(PNO), FOREIGN KEY(DNO)

REFERENCES DEPARTMENT(DNO) ON DELETE CASCADE

SSN VARCHAR(20), PNO INT,

HOURS INT,

PRIMARY KEY(SSN,PNO), FOREIGN KEY(SSN) REFERENCES EMPLOYEE(SSN) ON DELETE CASCADE, FOREIGN KEY(PNO) REFERENCES PROJECT(PNO) ON DELETE CASCADE

ALTER TABLE DEPARTMENT ADD CONSTRAINT C\_MSSN FOREIGN KEY(MGR\_SSN) REFERENCES EMPLOYEE(SSN) ON DELETE CASCADE; UPDATE DEPARTMENT

SET MGR\_SSN='&MGR\_SSN' WHERE DNAME='&DNAME';

INSERT INTO DEPARTMENT VALUES('&DNO','&DNAME','&MGRSSN','&MGRSTARTDATE');

INSERT INTO EMPLOYEEVALUES('&SSN','&NAME','&ADDRESS','&SEX','&SALARY','&SUPERSSN','&DNO'); INSERT INTO DLOCATION VALUES('&DNO','&DLOC');

INSERT INTO PROJECT VALUES('&PNO','&PNAME','&PLOCATION','&DNO'); INSERT INTO WORKS\_ON VALUES('&SSN','&PNO','&HOURS');

SELECT \* FROM DEPARTMENT;

DNO DNAME MGR\_SSN MGR\_START

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

* 1. ACCOUNTS 01-JAN-00
  2. FINANCE 01-FEB-01
  3. RESEARCH 03-MAR-01

4 HR 04-APR-02

5 SOFTWARE 05-FEB-03

SQL> SELECT \* FROM EMPLOYEE;

SSN NAME ADDRESS S SALARY SUPER\_SSN

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

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| E1 | SHAWN | FRANCE | M | 600000 | E1 |
| E2 | SCOTT | LONDON | M | 700000 | E2 |
| E3 | ROBERT | ENGLAND | M | 500000 | E3 |
| E4 | ROBERT | FRANCE | M | 400000 | E4 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| E6 | TOM | LONDON | M | 200000 | E6 |
| E7 | JOHN | USA | M | 100000 | E7 |
| E8 | TEENA | NEWYORK | F | 500000 | E8 |
| E5 | MICKY | LONDON | M | 300000 | E5 |
| E9 | ROSHEL | KOREA | F | 700000 | E9 |
| E10 | SMITH | LONDON | F | 400000 | E10 |

10 rows selected. AFTER UPDATING :

SQL> SELECT \* FROM DEPARTMENT;

DNO DNAME MGR\_SSN MGR\_START

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

1. ACCOUNTS E1 01-JAN-00
2. FINANCE E2 01-FEB-01
3. RESEARCH E3 03-MAR-01

4 HR E4 04-APR-02

5 SOFTWARE E5 05-FEB-03 SQL> SELECT \* FROM DLOCATION;

DNO DLOC

---------- -------------------- 1 FRANCE

1. LONDON
2. NEWYORK
3. ENGLAND
4. KOREA

SQL> SELECT \* FROM PROJECT;

PNO PNAME PLOCATION DNO

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

1. IOT FRANCE 1
2. CLOUD COMPUTING LONDON 2
3. MACHINE LEARNING NEWYORK 3
4. DATA MINING ENGLAND 4
5. ANDROID APP KOREA 5

SQL> SELECT \* FROM WORKS\_ON; SSN PNO HOURS

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

|  |  |  |
| --- | --- | --- |
| E1 | 1 | 2 |
| E2 | 2 | 4 |
| E3 | 3 | 5 |
| E4 | 4 | 6 |
| E5 | 5 | 4 |
| E6  6 rows selected. | 5 | 2 |

## QUERY 1:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*----@@@@@@@@@@@@@@@@@@@@ (SELECT DISTINCT P.PNO FROM PROJECT P,EMPLOYEE E,DEPARTMENT D WHERE

E.DNO=D.DNO AND D.DNO=P.DNO AND D.MGR\_SSN=E.SSN AND NAME='SCOTT') UNION

(SELECT DISTINCT P.PNO FROM PROJECT P,EMPLOYEE E,WORKS\_ON W WHERE

E.SSN=W.SSN AND P.DNO=W.PNO AND NAME='SCOTT');

PNO

-------- 2

## QUERY 2:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*----@@@@@@@@@@@@@@@@@@@@ SELECT NAME,1.1\* SALARY AS INCREASEDSALARY FROM

EMPLOYEE E,WORKS\_ON W,PROJECT P WHERE E.SSN=W.SSN AND

W.PNO=P.PNO AND PNAME='IOT';

NAME INCREASEDSALARY

-------------------- --------------- SHAWN 660000

## QUERY 3:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*----@@@@@@@@@@@@@@@@@@@@ SELECT SUM(SALARY),MAX(SALARY),MIN(SALARY),AVG(SALARY) FROM

EMPLOYEE E,DEPARTMENT D WHERE DNAME='ACCOUNTS' AND E.DNO=D.DNO;

SUM(SALARY) MAX(SALARY) MIN(SALARY) AVG(SALARY)

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

2500000 700000 100000 416666.667

## QUERY 4:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*----@@@@@@@@@@@@@@@@@@@@ SELECT NAME FROM EMPLOYEE E WHERE NOT EXISTS

((SELECT PNO FROM PROJECT WHERE DNO=5)MINUS (SELECT PNO FROM WORKS\_ON W WHERE E.SSN=W.SSN));

NAME

-------------------- TOM

MICKY

## QUERY 5:

@@@@@@@@@@@@@@@@@@@@-------\*\*\*-------\*\*\*----@@@@@@@@@@@@@@@@@@@@ SELECT E.DNO,COUNT(\*) FROM

DEPARTMENT D,EMPLOYEE E WHERE D.DNO=E.DNO AND

SALARY>600000 AND

E.DNO IN (SELECT DNO FROM EMPLOYEE GROUP BY DNO HAVING COUNT(\*)>5) GROUP BY E.DNO;

DNO COUNT(\*)

--------- ---------- 1 1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* END \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*