S. D. M. COLLEGE OF EMPGINEERING AND TECHNOLOGY, DHARWAD – 580002

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

(An Autonomous Institution affiliated to Visvesvarayya Technological University, Belgavi)

Department of Information Science and Engineering

As part of DBMS LAB PROJECT REPORT ON

TRAVEL AGENCY MANAGEMENT SYSTEM

Prepared By:

Chandana Hegde 2SD18IS014

Deepti Nayak 2SD18IS016

(Students of 5th sem,ISE 2020-21 Batch)

Under Guidance Of:

Prof. Leena I. Sakri

S. D. M. COLLEGE OF ENGINEERING AND TECHNOLOGY, DHARWAD – 580002

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the mini project work "Travel agency Management System" is a bona-fide work carried out by Chandana Hegde (2SD18IS014) partial fulfillment for the award degree for 5th semester Bachelor of Engineering in Information Science and Engineering of S. D. M college of Engineering, Autonomous Institution under Visvesvarayya Technological University, Belgaum during the year 2020-21. The project report has been approved as it satisfies academic requirements in respects of project work prescribed for bachelor of engineering degree.

GUIDE	HOD-ISE	PRINCIPAL
Prof. Leena Sakri	Dr. Jagadish pujari	Gopinath

Contents of Report:

Table of contents

1. Project profile

Project definition	*Travel agency Management System*
Objective	Main objective of this system is to provide online registration, Tour packages, online booking, online payment and facilities from quires.
Tools	M.S. World, ERDplus.
Internal Guide	Mrs. Leena Sakri
Submitted to	Department of information science
Developed by	Chandana Hegde Deepti Nayak

TRAVEL AGENCY DATABASE

Travel agency database shares details about the packages, customer, reservation, payment details and customer's reviews on the travel agency.

Requirements:

Travel agency offers one or more packages.

A customer can make zero or more reservation.

A customer selects one or more packages.

Customer makes payment according to reservation.

Customer may or may not review the agency.

Customer may have companions.

Reservation reveal rid, pname, transportation model, destination location, staying hotel, number of days, total price.

A customer has name, one or more given address, phone number, email, reservation_no, amount_paid, and a unique customer_id, payment_no, mode of Payment, amount and date of Payment.

Travel agency has a name, email and address.

--Entities and attributes—

1. CUSTOMER

Has following attributes.

- Name
- cid
- address
- phone
- email
- nationality
- gender
- age
- no_of_companions
- pno
- transaction _mode
- amount_paid
- reservation_no
- date
- travel_Agency_name

2. TRAVEL AGENCY **:-**

Has following attributes.

- tname
- address
- email

3. RESERVATION:-

Has following attributes.

- rid
- Amount
- Start_date
- End_date
- Package_name
- Location
- Hotel
- Package_price
- Transportation
- No_of_days

Relationship between entities:

1. MAKES :
   - A 1:1 relationship between CUSTOMER and RESERVATION.
   - Participation of CUSTOMER is partial.
   - Participation of RESERVATION is total.

2.REVIEW :

   - A N:1 relationship between CUSTOMER and TRAVEL_AGENCY
   - Both of them have total participation.

SCHEMA DIAGRAM

CUSTOMER

name	cid	pno	phone	address	nationality	gender	age	TransactionMode	tname	noofcompanion	Date	amount_paid

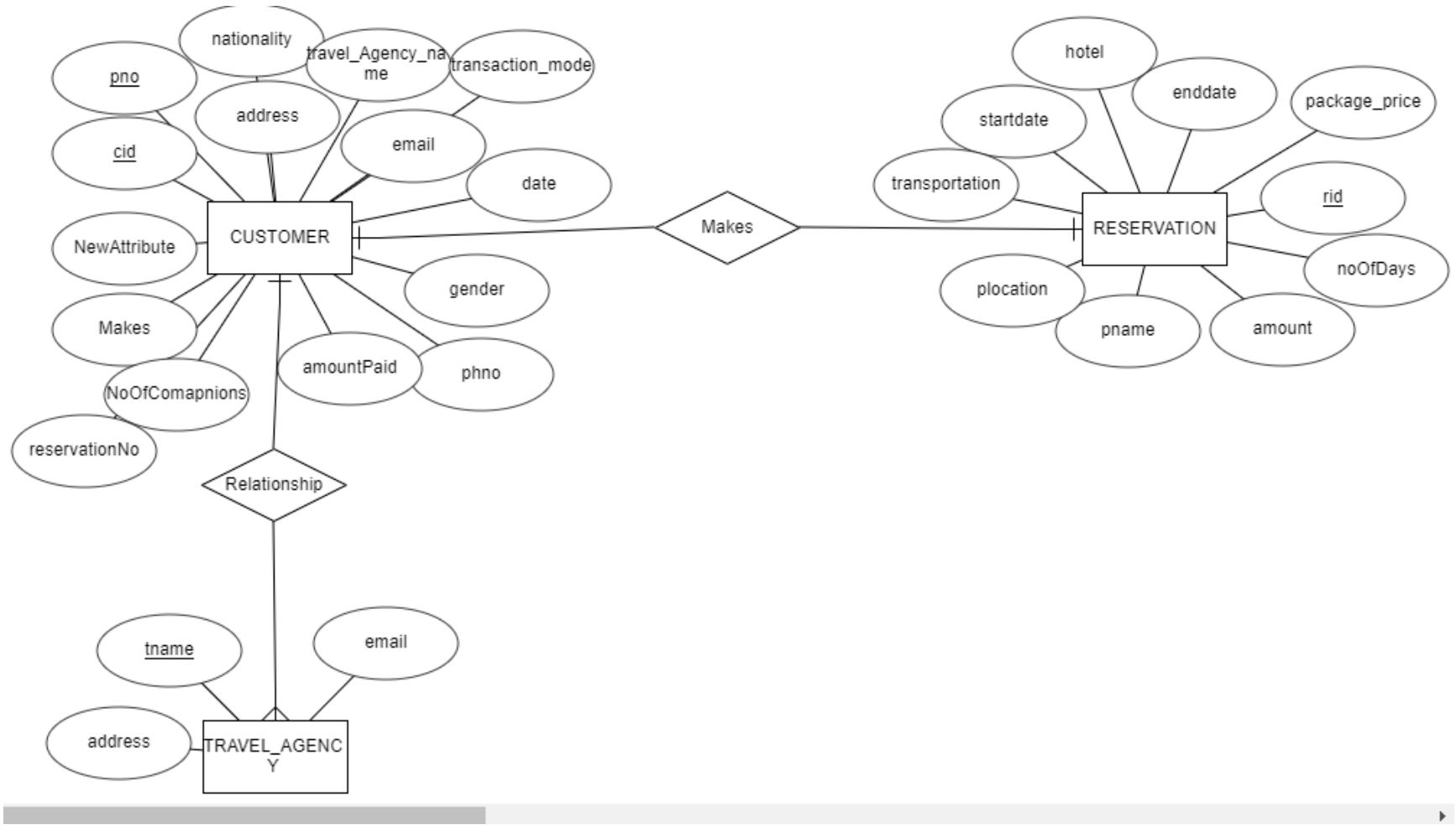
TRAVEL _AGECNY

tname	Email	address

RESERVATION

rid	amount	package_price	plocation	Hotel	plocation	transportation	noOfdays	start_date	end_date

ER DIAGRAM



APPLYING 1NF :

CUSTOMER

name	cid	pno	phone	address	nationality	gender	age	TransactionMode	tname	noofcompanion	Date	amountPaid	Reservatio nNo

In CUSTOMER, all the values are atomic so it is in 1NF already.

TRAVEL _AGECNY

tname	Email	address

In TRAVEL_AGENCY, all the attributes are atomic so it is in 1NF already.

RESERVATION

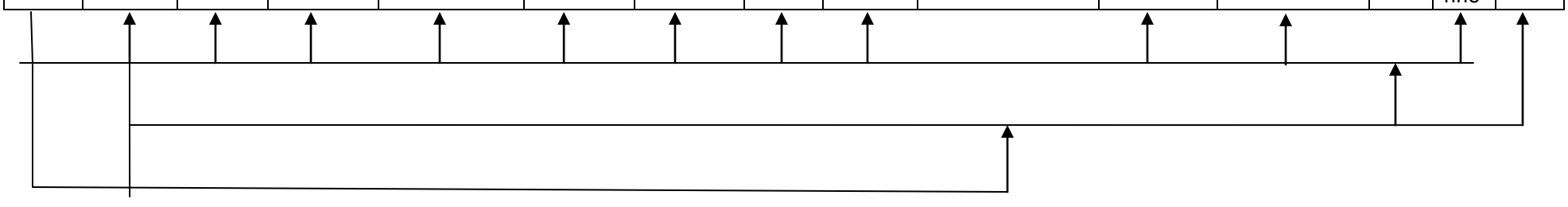
rid	amount	package_price	plocation	Hotel	plocation	transportation	noOfdays	start_date	end_date

In Reservation, all the attributes are atomic so it is in 1NF already.

2NF:

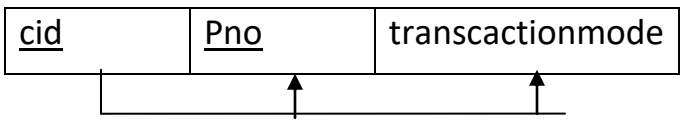
CUSTOMER

cid	Pno	Phone	address	nationality	name	Gender	age	email	TransactionMode	tname	noofcomapnion	date	reservationno	amountpaid



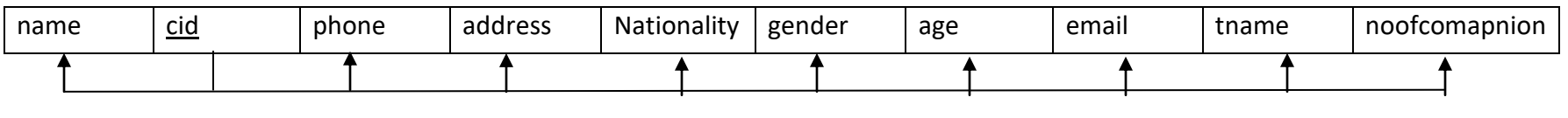
PAYMENT_MODE

cid	Pno	transcactionmode



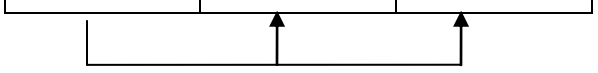
CUSTOMER

name	cid	phone	address	Nationality	gender	age	email	tname	noofcomapnion



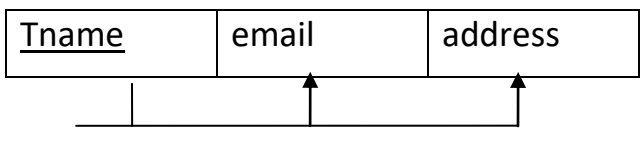
PAYMENT

Pno	price	date



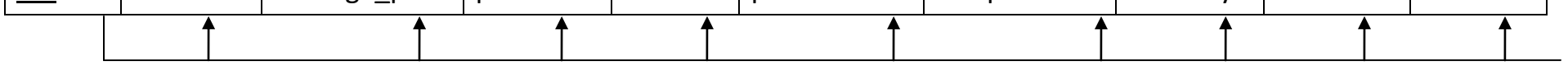
TRAVEL_AGENCY

Tname	email	address



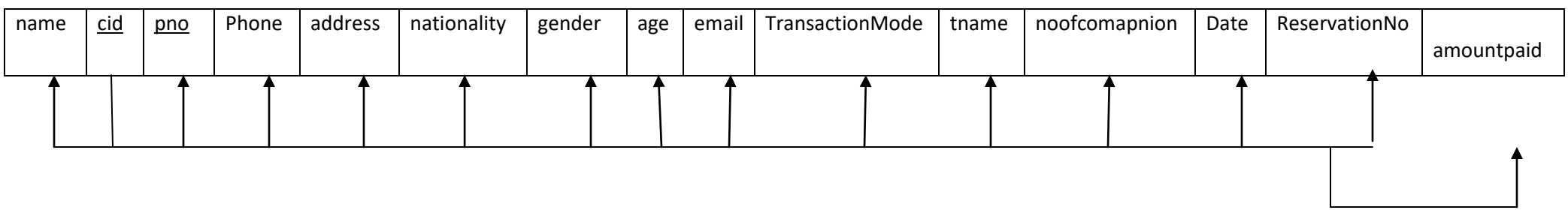
RESERVATION

Rid	amount	Package_price	plocation	hotel	plocation	tranportation	noOfdays	startdate	endate



3NF:

CUSTOMER



Here there is a transitive dependency of non-key attribute on the primary key. Because non-key attribute amount_paid depends on reservation_no so we need to apply 3NF.After applying 3NF we get,

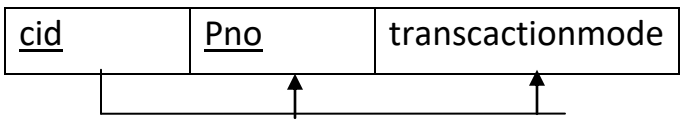
CUSTOMER

name	cid	pno	Phone	Address	nationality	Gender	age	email	Transaction Mode	tname	noofcomapnion	Date	ReservationNo

RESERVATION_AMOUNT

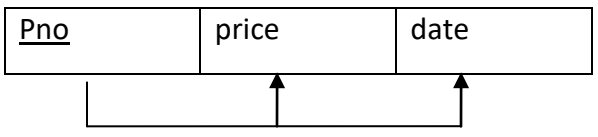
ReservationNo	amountpaid

PAYMENT_MODE



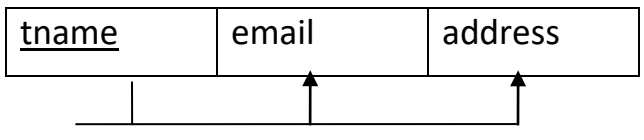
Here there is no transitive dependency of non-key attribute on the primary key. So it is in 3NF.

PAYMENT



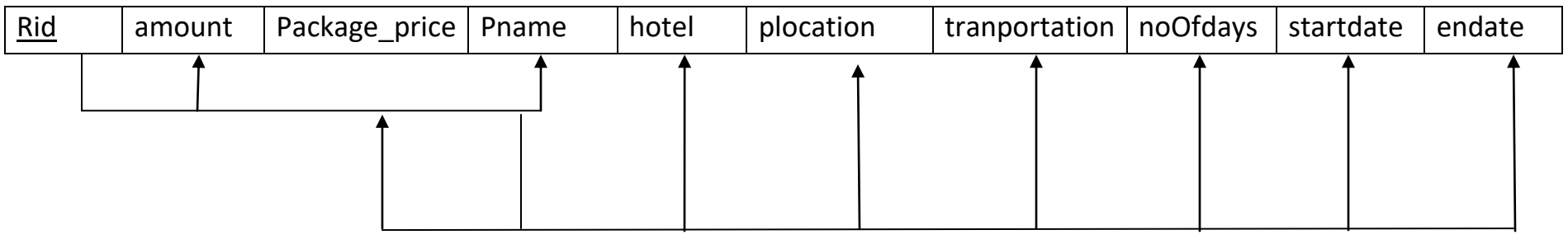
Here there is no transitive dependency of non-key attribute on the primary key. So it is in 3NF.

TRAVEL_AGENCY



Here there is no transitive dependency of non-key attribute on the primary key.So it is in 3NF.

RESERVATION



Here there is a transitive dependency of non-key attribute on the primary key. Because non-key attributes hotel, package_price,plocation,transportation,no_of_days,start_date,end_date depends on non-key attribute pname. Hence we should apply 3NF.So after applying 3NF we get,

RESERVATION

rid	amount	pno

PACKAGES

Package_price	Pname	hotel	plocation	tranportation	noOfdays	startdate	endate

NAME: **Chandana Hegde**

USN: **2SD18IS014**

SUBJECT: DBMS PROJECT

CREATION OF TABLES

1] TABLE NAME: ZIPCODEINFO

```
SQL> create table ZIPCODEINFO(
  2  Zip_code varchar2(5),
  3  city  varchar2(10));

Table created.

SQL> desc ZIPCODEINFO;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ZIP_CODE                                           VARCHAR2(5)
 CITY                                               VARCHAR2(10)
```

2] TABLE NAME: INSTRUCTORINFO

```
SQL> CREATE TABLE INSTRUCTORINFO(
  2  INSTRUCTOR_ID  number,
  3  INSTRUCTOR_FNAME VARCHAR(15),
  4  INSTRUCTOR_LNAME  VARCHAR(15));

Table created.

SQL> DESC INSTRUCTORINFO;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 INSTRUCTOR_ID                                      NUMBER
 INSTRUCTOR_FNAME                                   VARCHAR2(15)
 INSTRUCTOR_LNAME                                   VARCHAR2(15)
```

3] TABLE NAME: COURSEINFO

```
SQL> CREATE TABLE COURSEINFO(
  2  COURSE_NO NUMBER,
  3  COST NUMBER);

Table created.

SQL> DESC COURSEINFO;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 COURSE_NO                                          NUMBER
 COST                                               NUMBER
```

4] TABLE NAME: COURSEINFO

```
SQL> CREATE TABLE COURSEINFO(
  2  COURSE_NO NUMBER,
  3  COST NUMBER);

Table created.

SQL> DESC COURSEINFO;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 COURSE_NO                                          NUMBER
 COST                                               NUMBER
```

5] TABLE NAME: STUDENTINFO

```
SQL> CREATE TABLE STUDENTINFO(
  2  STUDENT_ID NUMBER,
  3  STUDENT_FNAME VARCHAR(15),
  4  STUDENT_LNAME VARCHAR(15));

Table created.

SQL> DESC STUDENTINFO;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ID                                         NUMBER
 STUDENT_FNAME                                      VARCHAR2(15)
 STUDENT_LNAME                                      VARCHAR2(15)
```

6] TABLE NAME: SECTIONINFO

```
SQL> CREATE TABLE SECTIONINFO(
  2  STUDENT_ID NUMBER,
  3  SECTION_ID NUMBER);

Table created.

SQL> DESC SECTIONINFO;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ID                                         NUMBER
 SECTION_ID                                         NUMBER
```

7] TABLE NAME: ENROLLMENTINFO

```
SQL> CREATE TABLE ENROLLMENTINFO(
  2  STUDENT_ID NUMBER,
  3  SECTION_ID NUMBER);

Table created.

SQL> DESC ENROLLMENTINFO;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ID                                         NUMBER
 SECTION_ID                                         NUMBER
```

8] TABLE NAME: GRADEINFO

```
SQL> CREATE TABLE GRADEINFO(
  2  STUDENT_ID NUMBER,
  3  SECTION_ID NUMBER,
  4  GRADE_CODE CHAR(2),
  5  GRADE_CODE_OCCURANCE NUMBER);

Table created.

SQL> DESC GRADEINFO;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ID                                         NUMBER
 SECTION_ID                                         NUMBER
 GRADE_CODE                                         CHAR(2)
 GRADE_CODE_OCCURANCE                               NUMBER
```

ALTERING TABLES

1] QUERIES:

- ALTER TABLE ZIPCODEINFO RENAME TO ZIPCODE_INFO;
- ALTER TABLE ZIPCODE_INFO ADD STATE VARCHAR2(15);

```
SQL> ALTER TABLE ZIPCODE_INFO ADD STATE VARCHAR2(2);

Table altered.
```

OUTPUT:

```
SQL> DESC ZIPCODE_INFO;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ZIP_CODE                                  NOT NULL VARCHAR2(5)
 CITY                                      NOT NULL VARCHAR2(25)
 STATE                                              VARCHAR2(2)
```

2] QUERIES:

- ALTER TABLE INSTRUCTORINFO RENAME TO INSTRUCTOR_INFO;

```
SQL> alter table instructorinfo
  2  modify instructor_first_name varchar2(25);

Table altered.


SQL> alter table instructorinfo
  2  modify instructor_last_name varchar2(25);

Table altered.


SQL> alter table instructorinfo
  2  add street_address varchar2(50);

Table altered.


SQL> alter table instructorinfo
  2  add zip_code number(5);

Table altered.
```

OUTPUT:

```
SQL> DESC INSTRUCTOR_INFO;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 INSTRUCTOR_ID                             NOT NULL NUMBER
 INSTRUCTOR_FIRST_NAME                     NOT NULL VARCHAR2(25)
 INSTRUCTOR_LAST_NAME                      NOT NULL VARCHAR2(25)
 STREET_ADDRESS                                     VARCHAR2(50)
 ZIP_CODE                                           NUMBER(5)
```

3] QUERIES:

- ALTER TABLE COURSEINFO RENAME TO COURSE_INFO;

```
SQL> ALTER TABLE COURSE_INFO ADD course_name varchar2(50);

Table altered.

SQL> ALTER TABLE COURSE_INFO ADD course_prerequisite number(8,0);

Table altered.
```

OUTPUT:

```
SQL> DESC COURSE_INFO;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 COURSE_NO                                          NUMBER(8)
 COURSE_NAME                                        VARCHAR2(50)
 COURSE_PREREQUISITE                                NUMBER(8)
 COST                                               NUMBER
```

4] QUERIES:

- ALTER TABLE STUDENTINFO RENAME TO STUDENT_INFO;
- ALTER TABLE STUDENT_INFO RENAME COLUMN STUDENT_FNAME TO STUDENT_FIRST_NAME;
- ALTER TABLE STUDENT_INFO RENAME COLUMN STUDENT_LNAME TO STUDENT_LAST_NAME;

```
SQL> ALTER TABLE STUDENT_INFO ADD STREET_ADDRESS VARCHAR2(50);

Table altered.
```

```
SQL> ALTER TABLE STUDENT_INFO ADD ZIP_CODE VARCHAR2(5);

Table altered.
```

OUTPUT:

```
SQL> DESC STUDENT_INFO;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ID                                         NUMBER(8)
 STUDENT_FIRST_NAME                                 VARCHAR2(15)
 STUDENT_LAST_NAME                                  VARCHAR2(15)
 STREET_ADDRESS                                     VARCHAR2(50)
 ZIP_CODE                                           VARCHAR2(5)
```

5] QUERIES:

- ALTER TABLE SECTIONINFO RENAME TO SECTION_INFO;
- ALTER TABLE SECTION_INFO RENAME COLUMN STUDENT_ID TO SECTION_NO;
- ALTER TABLE SECTION_INFO ADD COURSE_NO NUMBER;
- ALTER TABLE SECTION_INFO ADD INSTRUCTOR_ID NUMBER;

```
SQL> ALTER TABLE SECTION_INFO ADD LOCATION VARCHAR2(50);

Table altered.

SQL> ALTER TABLE SECTION_INFO ADD CAPACITY INTEGER;

Table altered.
```

OUTPUT:

```
SQL> DESC SECTION_INFO;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SECTION_ID                                         NUMBER(38)
 COURSE_NO                                          NUMBER(38)
 SECTION_NO                                         NUMBER(38)
 INSTRUCTOR_ID                                      NUMBER(38)
 LOCATION                                           VARCHAR2(50)
 CAPACITY                                           NUMBER(38)
```

6] QUERIES:

- ALTER TABLE ENROLLMENTINFO RENAME TO ENROLLMENT_INFO;

```
SQL> ALTER TABLE ENROLLMENT_INFO ADD ENROLLMENT_DATE DATE;

Table altered.
```

OUPUT:

```
SQL> DESC ENROLLMENT_INFO;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ID                                         NUMBER(8)
 SECTION_ID                                         NUMBER(8)
 ENROLLMENT_DATE                                    DATE
```

7] QUERIES:

- ALTER TABLE GRADE_INFO RENAME COLUMN GRADE_CODE TO GRADE_TYPE_CODE;
- ALTER TABLE GRADE_INFO ADD NUMERIC GRADE NUMBER(8);

OUTPUT:

```
SQL> Desc grade_info;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 STUDENT_ID                                         NUMBER(8)
 SECTION_ID                                         NUMBER(8)
 GRADE_TYPE_CODE                                    CHAR(2)
 GRADE_CODE_OCCURANCE                               NUMBER(15)
 NUMERIC_GRADE                                      NUMBER(3)
```

SIMPLE SQL QUERIES

1. Display the structure of the course table.

```
SQL> DESC COURSE_INFO
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 COURSE_NO                                 NOT NULL NUMBER(8)
 COURSE_NAME                               NOT NULL VARCHAR2(50)
 COURSE_PREREQUISITE                                NUMBER(8)
 COST                                      NOT NULL NUMBER
```

2. Display the zipcode, city and state. Observe the column heading of state column. If not appearing correctly, give a proper heading.

```
SQL>  SELECT ZIP_CODE, CITY, STATE FROM ZIPCODE_INFO;

ZIP_C CITY                      ST
----- ------------------------- --
77016 Houston                   TX
11003 Elmont                    NY
91316 Encino                    CA
94566 Pleasanton                CA
11762 Massapequa Park           NY

SQL> COL STATE FORMAT A5;
SQL>
```

3. Display the unique states

```
SQL> SELECT DISTINCT STATE FROM ZIPCODE_INFO;

STATE
-----
CA
NY
TX
```

4. Display the student_id, name. Concatenate the first_name and last name.

```
SQL>  SELECT STUDENT_ID, STUDENT_FIRST_NAME||' '||STUDENT_LAST_NAME
             FROM STUDENT_INFO;

STUDENT_ID STUDENT_FIRST_NAME||''||STUDENT
---------- ------------------------------
     40001 James Anderson
     40002 franklin borg
     40003 Fury english
     40004 Steve Rogers
     40005 Tony Stark
```

5. Display the Zipcode, city and state as a single column. Separate the data with a comma. E.g 400050, Mumbai, MH. Give the column heading as Address.

```
SQL> SELECT ZIP_CODE||','||CITY||','||STATE AS ADDRESS FROM ZIPCODE_INFO;

ADDRESS
---------------------------------
77016,Houston,TX
11003,Elmont,NY
91316,Encino,CA
94566,Pleasanton,CA
11762,Massapequa Park,NY
```

6. Display the description of the course

```
SQL> SELECT COURSE_NAME FROM COURSE_INFO;

COURSE_NAME
--------------------------------------------------
C programing
DBMS
Data structure
Advance DBMS
Avance data structure
```

7. In the above query also display the cost.

```
SQL> SELECT COURSE_NAME, COST FROM COURSE_INFO;

COURSE_NAME                                           COST
-------------------------------------------------- ----------
C programing                                          2000
DBMS                                                  3000
Data structure                                        4500
Advance DBMS                                          5000
Avance data structure                                 5500
```

8. Display all the columns of the course_info table.

```
SQL> SELECT * FROM COURSE_INFO;

 COURSE_NO COURSE_NAME
---------- --------------------------------------------------
COURSE_PREREQUISITE       COST
------------------- ----------
     50001 C programing
                          2000

     50002 DBMS
                          3000

     50003 Data structure
              50001       4500


 COURSE_NO COURSE_NAME
---------- --------------------------------------------------
COURSE_PREREQUISITE       COST
------------------- ----------
     50004 Advance DBMS
              50002       5000

     50005 Avance data structure
              50003       5500
```

9. Display the instructor's last name and the zip in which the instructor resides.

```
SQL> SELECT INSTRUCTOR_FIRST_NAME, ZIP_CODE FROM INSTRUCTOR_INFO;

INSTRUCTOR_FIRST_NAME     ZIP_C
------------------------- -----
Ted                       77016
Bob                       11003
Ron                       91316
Joy                       94566
william                   11762
```

10. There are many students who are living in the area, display unique zip's only.

```
SQL> SELECT DISTINCT ZIP_CODE FROM STUDENT_INFO;

ZIP_C
-----
94566
11762
77016
11003
91316
```

11. Write a select statement to list the first and last names of all students.

```
SQL> SELECT STUDENT_ID, STUDENT_FIRST_NAME, STUDENT_LAST_NAME FROM STUDENT_INFO;

STUDENT_ID STUDENT_FIRST_N STUDENT_LAST_NA
---------- --------------- ---------------
     40001 James           Anderson
     40002 franklin        borg
     40003 Fury            english
     40004 Steve           Rogers
     40005 Tony            Stark
```

12. Write a select statement to list all cities, their states and zip codes.

```
SQL> SELECT CITY, STATE, ZIP_CODE FROM ZIPCODE_INFO;

CITY                      STATE ZIP_C
------------------------- ----- -----
Houston                   TX    77016
Elmont                    NY    11003
Encino                    CA    91316
Pleasanton                CA    94566
Massapequa Park           NY    11762
```

SELECT USING WHERE

1.Display the studenUd, section_id and numeric_grade of those students who have the grade code type as 'Fl'

```
SQL>  SELECT STUDENT_ID, SECTION_ID, NUMERIC_GRADE FROM GRADE_INFO WHERE GRADE_TYPE_CODE='F1';

STUDENT_ID SECTION_ID NUMERIC_GRADE
---------- ---------- -------------
     40005      60005             5
```

2. Display the Zipcode and cities for the state 'MI'

```
SQL> SELECT ZIP_CODE, CITY FROM ZIPCODE_INFO WHERE STATE='MI';

ZIP_C CITY
----- -------------------------
58024 Massapequa Park
```

3. Display the student details that have enrolled in the month of Jan 1999. Sort the data in the ascending order of student names. Note: Names should be concatenated.

```
SQL> select STUDENT_ID,SECTION_ID from enrollment_info where enrollment_date LIKE '%JAN-99';

STUDENT_ID SECTION_ID
---------- ----------
     40006      60001
```

4. Display the section and the instructor id of the course 10 and 20. Sort the data in the ascending order of instructor id.

```
SQL> SELECT SECTION_ID, INSTRUCTOR_ID FROM SECTION_INFO WHERE COURSE_NO IN (10, 20) ORDER BY INSTRUC
TOR_ID;

SECTION_ID INSTRUCTOR_ID
---------- -------------
     60007         10005
```

5. Display the studenUd, section_id and numeric_grade. Sort the section_id in ascending order followed by numeric_grade in descending order.

```
SQL> SELECT STUDENT_ID, SECTION_ID, NUMERIC_GRADE FROM GRADE_INFO ORDER BY SECTION_ID, NUMERIC_GRADE
 DESC;

STUDENT_ID SECTION_ID NUMERIC_GRADE
---------- ---------- -------------
     40001      60001            10
     40002      60002             7
     40003      60003             9
     40004      60004             6
     40005      60005             8
```

6. Display the course_no, description and cost of the courses. The courses should have the word 'Intro'

```
SQL> SELECT COURSE_NO, COURSE_NAME, COST FROM COURSE_INFO WHERE COURSE_NAME LIKE 'Intro%'
  2  ;

 COURSE_NO COURSE_NAME                                          COST
---------- -------------------------------------------------- ----------
     50008 Intro to Advanced MongoDB                            4300
```

7. Display the course details whose third from last letter is 'a'

```
SQL> SELECT COURSE_NO, COURSE_NAME, COST FROM COURSE_INFO WHERE COURSE_NAME LIKE '__a%';

 COURSE_NO COURSE_NAME                                          COST
---------- -------------------------------------------------- ----------
     50005 Avance data structure                                5500
```

8. Display the student names whose studenUd is in the range b of 300 to 350.

```
SQL> SELECT STUDENT_FIRST_NAME, STUDENT_LAST_NAME FROM STUDENT_INFO WHERE STUDENT_ID BETWEEN 300 AND
 350
  2  ;

STUDENT_FIRST_N STUDENT_LAST_NA
--------------- ---------------
Tony            Stark
```

9. Display the course details whose cost ranges from 4000 to 7000.

```
SQL> SELECT COURSE_NO, COURSE_NAME, COST FROM COURSE_INFO WHERE COST BETWEEN 4000 AND 7000;

 COURSE_NO COURSE_NAME                                          COST
---------- -------------------------------------------------- ----------
     50003 Data structure                                       4500
     50004 Advance DBMS                                         5000
     50005 Avance data structure                                5500
```

10. Write a select statement that displays the instructor's first name whose last name is 'Schumer'

```
SQL> SELECT INSTRUCTOR_FIRST_NAME,INSTRUCTOR_LAST_NAME, STREET_ADDRESS FROM INSTRUCTOR_INFO WHERE IN
STRUCTOR_LAST_NAME ='Schumer ';

INSTRUCTOR_FIRST_NAME     INSTRUCTOR_LAST_NAME
------------------------- -------------------------
STREET_ADDRESS
--------------------------------------------------
Ted                       Schumer
895 Chestnut Ave.
```

11. Write a select statement that displays the instructor's first name whose last name is not 'Schumer'

```
SQL> SELECT INSTRUCTOR_FIRST_NAME,INSTRUCTOR_LAST_NAME, STREET_ADDRESS FROM INSTRUCTOR_INFO WHERE IN
STRUCTOR_LAST_NAME !='Schumer';

INSTRUCTOR_FIRST_NAME     INSTRUCTOR_LAST_NAME
------------------------- -------------------------
STREET_ADDRESS
--------------------------------------------------
Ted                       Smith
895 Chestnut Ave.

Bob                       Wong
32 Glen Creek Lane

Ron                       Zelya
369 Vernon Dr


INSTRUCTOR_FIRST_NAME     INSTRUCTOR_LAST_NAME
------------------------- -------------------------
STREET_ADDRESS
--------------------------------------------------
Joy                       Wallace
9680 E. Somerset Street

william                   jabbar
8459 W. Newport Court
```

12. Display the course name and cost of those courses whose cost is more than 4000.

```
SQL> SELECT COURSE_NO, COURSE_NAME, COST FROM COURSE_INFO WHERE COST>4000
  2  ;

 COURSE_NO COURSE_NAME                                          COST
---------- -------------------------------------------------- ----------
     50003 Data structure                                          4500
     50004 Advance DBMS                                            5000
     50005 Avance data structure                                   5500
```

13. Display the course name and cost of those courses whose cost is in the range of 3000 and 7000.

```
SQL> SELECT COURSE_NO, COURSE_NAME, COST FROM COURSE_INFO WHERE COST BETWEEN 3000 AND 7000;

 COURSE_NO COURSE_NAME                                          COST
---------- -------------------------------------------------- ----------
     50002 DBMS                                                    3000
     50003 Data structure                                          4500
     50004 Advance DBMS                                            5000
     50005 Avance data structure                                   5500
```

14. Display the course name and cost of those courses whose cost is 4000 and 4500.

```
SQL> SELECT COURSE_NO, COURSE_NAME, COST FROM COURSE_INFO WHERE COST BETWEEN 4000 AND 4500;

 COURSE_NO COURSE_NAME                                          COST
---------- -------------------------------------------------- ----------
     50003 Data structure                                          4500
```

15. Write a select statement that displays the student's first name and address whose last name starts with 'S'.

```
SQL> SELECT STUDENT_FIRST_NAME, STUDENT_LAST_NAME FROM STUDENT_INFO WHERE STUDENT_LAST_NAME LIKE 'S%
'
  2  ;

STUDENT_FIRST_N STUDENT_LAST_NA
--------------- ---------------
Tony            Stark
```

16. Write a select statement that displays the student's first name and address whose last name contains '0' as the second letter.

```
SQL> SELECT STUDENT_FIRST_NAME, STUDENT_LAST_NAME, STREET_ADDRESS FROM STUDENT_INFO WHERE STUDENT_LA
ST_NAME LIKE '_o%';

STUDENT_FIRST_N STUDENT_LAST_NA
--------------- ---------------
STREET_ADDRESS
--------------------------------------------------
franklin        borg
32 Glen Creek Lane

Steve           Rogers
9680 E. Somerset Street
```

17. Write a select statement that displays the instructor's first name whose last name does not start with 'S'

```
SQL> SELECT INSTRUCTOR_FIRST_NAME, INSTRUCTOR_LAST_NAME, STREET_ADDRESS FROM INSTRUCTOR_INFO WHERE I
NSTRUCTOR_LAST_NAME NOT LIKE 'S%';

INSTRUCTOR_FIRST_NAME     INSTRUCTOR_LAST_NAME
------------------------- -------------------------
STREET_ADDRESS
--------------------------------------------------
Bob                       Wong
32 Glen Creek Lane

Ron                       Zelya
369 Vernon Dr

Joy                       Wallace
9680 E. Somerset Street


INSTRUCTOR_FIRST_NAME     INSTRUCTOR_LAST_NAME
------------------------- -------------------------
STREET_ADDRESS
--------------------------------------------------
william                   jabbar
8459 W. Newport Court
```

18. Display the course name and cost of those courses whose cost is in range of 4000 and 4500 and the course name starts with 'I'.

```
SQL> SELECT COURSE_NAME, COST FROM COURSE_INFO WHERE COST BETWEEN 4000 AND 4500 AND COURSE_NAME LIKE
 'I%'
  2  ;

COURSE_NAME                                             COST
-------------------------------------------------- ----------
Intro to Advanced MongoDB                                4300
```

19. Display the course name and cost and prerequisite of those courses whose cost is 2000 and prerequisite is 20 and all courses whose prerequisite is 25.

```
SQL> SELECT COURSE_NAME, COST, COURSE_PREREQUISITE FROM COURSE_INFO WHERE (COST=2000 AND COURSE_PRER
EQUISITE=20) OR COURSE_PREREQUISITE=25;

COURSE_NAME                                             COST
-------------------------------------------------- ----------
COURSE_PREREQUISITE
-------------------
Java Advanced                                            4300
                25
```

20. Display the course name and cost and prerequisite of those courses whose cost is 2000 and all records whose prerequisite is 20 and 25.

```
SQL> SELECT COURSE_NAME, COST, COURSE_PREREQUISITE FROM COURSE_INFO WHERE COST=2000 AND (COURSE_PRER
EQUISITE=20 OR COURSE_PREREQUISITE=25)
  2  ;

COURSE_NAME                                             COST
-------------------------------------------------- ----------
COURSE_PREREQUISITE
-------------------
Social Studies                                           2000
                20
```

21. Display the records of the course_info table which have no prerequisite.

```
SQL> SELECT COURSE_NAME, COST, COURSE_PREREQUISITE FROM COURSE_INFO WHERE COURSE_PREREQUISITE IS NUL
L;

COURSE_NAME                                             COST
-------------------------------------------------- ----------
COURSE_PREREQUISITE
-------------------
C programing                                             2000


DBMS                                                     3000
```

22. Write a select statement to list the last names of students living either in zip code 10048, 11102 and 11209.

```
SQL> SELECT STUDENT_LAST_NAME FROM STUDENT_INFO WHERE ZIP_CODE IN (10048, 11102, 11209);

STUDENT_LAST_NA
---------------
Disuza
```

23. Write a select statement to list the first and last names of instructors with the letter 'I' in any case in the last name and living in zip code 10025.

```
SQL> SELECT INSTRUCTOR_FIRST_NAME, INSTRUCTOR_LAST_NAME FROM INSTRUCTOR_INFO WHERE LOWER (INSTRUCTOR
_LAST_NAME) LIKE '%i%' AND ZIP_CODE='10025';

INSTRUCTOR_FIRST_NAME     INSTRUCTOR_LAST_NAME
------------------------- -------------------------
indiana                   irekson
```

24. Write a select statement to list descriptions of courses which have prerequisites and cost is less than 1100.

```
SQL> SELECT COURSE_NAME, COST, COURSE_PREREQUISITE FROM COURSE_INFO WHERE COURSE_PREREQUISITE IS NOT
 NULL AND COST <1100;

COURSE_NAME                                          COST
-------------------------------------------------- ----------
COURSE_PREREQUISITE
-------------------
intro to MongoDB                                      900
             50006
```

25. Write a select statement to list the cost of courses that do not a prerequisite. In the result the cost should not be repeated.

```
SQL> SELECT DISTINCT COST FROM COURSE_INFO WHERE COURSE_PREREQUISITE IS NULL;

      COST
----------
      2000
      3000
```

26. Display the course_no and the coursename who do not a prerequisite. Sort the data on the basis of coursename.

```
SQL> SELECT COURSE_NO, COURSE_NAME FROM COURSE_INFO WHERE COURSE_PREREQUISITE IS NULL ORDER BY COURS
E_NAME;

 COURSE_NO COURSE_NAME
---------- --------------------------------------------------
     50001 C programing
     50002 DBMS
```

27. For the above query sort in the descending order of coursename.

```
SQL> SELECT COURSE_NO, COURSE_NAME FROM COURSE_INFO WHERE COURSE_PREREQUISITE IS NULL ORDER BY COURS
E_NAME DESC;

 COURSE_NO COURSE_NAME
---------- --------------------------------------------------
     50002 DBMS
     50001 C programing
```

28. Write a select statement to list each city in state New York or Connecticut sorted in ascending order by zipcode.

```
SQL> SELECT CITY FROM ZIPCODE_INFO WHERE STATE IN ('NY','CT') ORDER BY ZIP_CODE;

CITY
-------------------------
Elmont
Massapequa Park
```

29. Display the zip, first and last name of students with the last name Graham. Order the result by zip in descending order and the first name in ascending order.

```
SQL> SELECT STUDENT_FIRST_NAME, STUDENT_LAST_NAME FROM STUDENT_INFO WHERE STUDENT_LAST_NAME='Graham'
 ORDER BY ZIP_CODE DESC,STUDENT_FIRST_NAME ASC;

STUDENT_FIRST_N STUDENT_LAST_NA
--------------- ---------------
Vijet           Graham
```

30. Display the city and state as 1 column.

```
SQL> SELECT CITY||' , '|| STATE FROM ZIPCODE_info;

CITY||','||STATE
------------------------------
Houston , TX
Elmont , NY
Encino , CA
Pleasanton , CA
Massapequa Park , NY
```

31. Display the first name of the student. Only the first letter of the first name should be in uppercase.

```
SQL> SELECT INITCAP (STUDENT_FIRST_NAME) FROM STUDENT_INFO;

INITCAP(STUDENT
---------------
James
Franklin
Fury
Steve
Tony
```

32. Write a select statement that displays the instructor's first name followed by comma and a space which is followed by last_name. The result should be displayed in 1 result set.

```
SQL> SELECT INSTRUCTOR_FIRST_NAME||' , '|| INSTRUCTOR_LAST_NAME AS NAME FROM INSTRUCTOR_INFO;

NAME
--------------------------------------------------
Ted , Smith
Bob , Wong
Ron , Zelya
Joy , Wallace
william , jabbar
```

33. Display the select statement that displays the cost, add 10 to cost, subtract 20 from cost, multiply cost by 30 and divide cost by 5.

```
SQL> SELECT COST, COST + 10 AS TOTAL, COST - 20 AS SUB, COST * 30 AS MUL, COST/30 AS DIV FROM COURSE
_info;

      COST      TOTAL        SUB        MUL        DIV
---------- ---------- ---------- ---------- ----------
      2000       2010       1980      60000 66.6666667
      3000       3010       2980      90000        100
      4500       4510       4480     135000        150
      5000       5010       4980     150000 166.666667
      5500       5510       5480     165000 183.333333
```

34. Write a select statement that displays unique numeric grade, half these values and do not display this value in decimals. This calculation should be displayed as a separate column.

```
SQL> SELECT DISTINCT NUMERIC_GRADE, ROUND (NUMERIC_GRADE/2) FROM GRADE_INFO;

NUMERIC_GRADE ROUND(NUMERIC_GRADE/2)
------------- ----------------------
            9                      5
            7                      4
            6                      3
            8                      4
           10                      5
```

SELECT WITH TO_CHAR/DECODE/IS NULL

1. Display the cost in the following format for all the courses whose course_no is less

than 25.

```
SQL> select cost from course_info where course_no<60000;

      COST
----------
      2000
      3000
      4500
      5000
      5500
```

2. Write a query to format the cost column. The cost displayed should have a leading $ sign followed by a comma to separate the thousands and should display 2 decimals.

```
SQL> SELECT TO_CHAR (COST,'$9,999.99') FROM COURSE_INFO;

TO_CHAR(CO
----------
 $2,000.00
 $3,000.00
 $4,500.00
 $5,000.00
 $5,500.00
```

3. For those courses having no pre-requisites display data as 'Not applicable'

```
SQL> SELECT NVL (TO_CHAR (COURSE_PREREQUISITE),'Not Applicable') FROM COURSE_INFO;

NVL(TO_CHAR(COURSE_PREREQUISITE),'NOTAPP
----------------------------------------
Not Applicable
Not Applicable
50001
50002
50003
```

4. Display the state as 'New York' for NY, 'New Jersey' for NY and 'Others' for any other state.

```
SQL> SELECT DECODE(STATE, 'NY', 'New York', 'NY', 'New Jersey', 'OTHERS') FROM ZIPCODE_INFO;

DECODE(STA
----------
OTHERS
New York
OTHERS
OTHERS
New York
```

AGGREGATE FUNCTIONS

1.Display the count of records in the course table

```
SQL> SELECT COUNT(*)
  2  FROM COURSE_INFO;

  COUNT(*)
----------
         5
```

2. Display the total number of records in enrollment table .

```
SQL> SELECT COUNT(*)
  2  FROM ENROLLMENT_INFO;

  COUNT(*)
----------
         5
```

3. Display sum of numeric_grade from grade_info .

```
SQL> SELECT SUM(NUMERIC_GRADE) FROM GRADE_INFO;

SUM(NUMERIC_GRADE)
------------------
                40
```

4. Display the average, total, minimum and maximum numeric grade.

```
SQL> SELECT MAX(NUMERIC_GRADE),MIN(NUMERIC_GRADE),AVG(NUMERIC_GRADE),SUM(NUMERIC_GRADE)
  2  FROM GRADE_INFO;

MAX(NUMERIC_GRADE) MIN(NUMERIC_GRADE) AVG(NUMERIC_GRADE) SUM(NUMERIC_GRADE)
------------------ ------------------ ------------------ ------------------
                10                  6                  8                 40
```

5. Display the count of grade code type.

```
SQL> select count(grade_type_code) from grade_info;

COUNT(GRADE_TYPE_CODE)
----------------------
                     5
```

6. Write a SELECT statement that displays the total number of courses which do not have any pre- requisite.

```
SQL> SELECT COUNT(COURSE_PREREQUISITE)
  2  FROM COURSE_INFO
  3  WHERE COURSE_PREREQUISITE = 'NONE';

COUNT(COURSE_PREREQUISITE)
--------------------------
                         2
```

7. Display the date of the student who got recently enrolled.

```
SQL> SELECT MAX(ENROLLMENT_DATE)
  2  FROM ENROLLMENT_INFO;

MAX(ENROL
---------
22-NOV-20
```

GROUP BY AND HAVING CLAUSE

1.Display the count of cities for each state.

```
SQL> SELECT COUNT(CITY),STATE
  2  FROM ZIPCODE_INFO
  3  GROUP BY STATE;

COUNT(CITY) STATE
----------- ---------------
          2 CA
          2 NY
          1 TX
```

2. Display the minimum numeric grade section wise for each student.

```
SQL>  SELECT SECTION_ID,MIN(NUMERIC_GRADE)
  2   FROM GRADE_INFO
  3  GROUP BY SECTION_ID;

SECTION_ID MIN(NUMERIC_GRADE)
---------- ------------------
     60004                  6
     60003                  9
     60001                 10
     60005                  8
     60002                  7
```

3. Display the average numeric grade for each student.

```
SQL> SELECT STUDENT_ID, AVG(NUMERIC_GRADE)
  2  FROM GRADE_INFO
  3  GROUP BY STUDENT_ID;

STUDENT_ID AVG(NUMERIC_GRADE)
---------- ------------------
     40001                 10
     40002                  7
     40003                  9
     40004                  6
     40005                  8
```

4. Display the count of students enrolled in each section. Display only those sections where the number of students enrolled is more than 5.

```
SQL> select count(*),se.section_id
  2  from studentinfo s,enrollmentinfo e,sectioninfo se
  3  group by se.section_id having count(*)>5;

  COUNT(*) SECTION_ID
---------- ----------
        25      60001
        25      60002
        25      60003
        25      60004
        25      60005
```

5. Display the average numeric grade for each student and section. The average numeric grade should be more than 75.

```
SQL> select avg(numeric_grade) from grade_info group by student_id,section_id having avg(numeric_gra
de)>7;

AVG(NUMERIC_GRADE)
------------------
                10
                 9
                 8
```

6. For the above query display the data for student_id more than 280

```
SQL> select avg(numeric_grade),student_id from grade_info group by section_id,student_id having avg(
numeric_grade)>7 and student_id > 4002;

AVG(NUMERIC_GRADE) STUDENT_ID
------------------ ----------
                10      40001
                 9      40003
                 8      40005
```

7. Display each prerequisite and its count from the course_info table.

```
SQL> SELECT COUNT(COURSE_PREREQUISITE),COURSE_PREREQUISITE
  2  FROM COURSE_INFO
  3  GROUP BY COURSE_PREREQUISITE;

COUNT(COURSE_PREREQUISITE) COURSE_PREREQUI
-------------------------- ---------------
                         1 50002
                         1 50001
                         1 50003
                         2 NONE
```

8. Display student_id and number of courses they are enrolled in. Those students who are enrolled in more than 2 courses should be displayed.

```
SQL> select s.student_id,count(*)
  2  from studentinfo s,enrollmentinfo e,sectioninfo se
  3  where e.student_id=s.student_id and e.section_id=se.section_id
  4  group by s.student_id having count(*)>2;

no rows selected
```

9. Display the average capacity of each course.

```
SQL>  SELECT AVG(CAPACITY),COURSE_NO
  2   FROM SECTION_INFO
  3  GROUP BY COURSE_NO;

AVG(CAPACITY)  COURSE_NO
------------- ----------
          100      50003
          100      50004
          140      50005
          150      50002
          120      50001
```

10. For the above query display those courses which have exactly 2 sections.

```
SQL> select c.course_no,avg(capacity)
  2  from sectioninfo s,courseinfo c
  3  where s.course_no=c.course_no
  4  group by c.course_no having count(s.section_id)=2;

no rows selected
```

UNION/INTERSECT AND MINUS

1.Display the first name and last name of both the instructor and student in one result set.

```
SQL> select STUDENT__FIRST_NAME AS FIRST_NAME,  STUDENT__LAST_NAME AS LAST_NAME FROM STUDENT_INFO
  2   UNION
  3    select  INSTRUCTOR_FIRST__NAME, INSTRUCTOR_LAST__NAME from INSTRUCTOR_INFO ;

FIRST_NAME                LAST_NAME
------------------------- -------------------------
Bob                       Wong
Fury                      english
James                     Anderson
Joy                       Wallace
Ron                       Zelya
Steve                     Rogers
Ted                       Smith
Tony                      Stark
franklin                  borg
william                   jabbar

10 rows selected.
```

2. Modify the above query to display duplicate names also.

```
SQL> select STUDENT__FIRST_NAME AS FIRST_NAME,  STUDENT__LAST_NAME AS LAST_NAME FROM STUDENT_INFO
  2  UNION ALL
  3  select  INSTRUCTOR_FIRST__NAME, INSTRUCTOR_LAST__NAME from INSTRUCTOR_INFO ;

FIRST_NAME                LAST_NAME
------------------------- -------------------------
James                     Anderson
franklin                  borg
Fury                      english
Steve                     Rogers
Tony                      Stark
Ted                       Smith
Bob                       Wong
Ron                       Zelya
Joy                       Wallace
william                   jabbar
James                     Anderson
Tony                      Stark

12 rows selected.
```

4. Display the instructor id not having any section.

```
SQL> SELECT INSTRUCTOR_ID FROM INSTRUCTOR_INFO WHERE INSTRUCTOR_ID NOT IN(SELECT INSTRUCTOR_ID FROM
SECTION_INFO);

no rows selected

SQL>
```

5. Display the course no having section.

```
SQL> SELECT COURSE_NO FROM COURSE_INFO GROUP BY COURSE_NO HAVING COURSE_INFO.COURSE_NO=COURSE_NO INT
ERSECT
  2  SELECT COURSE_NO FROM SECTION_INFO GROUP BY COURSE_NO HAVING SECTION_INFO.COURSE_NO=COURSE_NO;

 COURSE_NO
----------
     50001
     50002
     50003
     50004
     50005

SQL>
```

7. Display all the zip codes that are present in both the instructor and the student tables.

```
SQL> select zip_code from instructor_info UNION select zip_code from student_info;

ZIP_C
-----
11003
11762
77016
91316
94566
```

8. Display the student ids who have enrolled.

```
SQL> select student_id from student_info MINUS select student_id from enrollment_info;

no rows selected
```

SUBQUERIES

Subqueries returning 1 row

1.Display the course_no, description and cost. Those records should be displayed whose cost is equal to the minimum cost.

```
SQL> select course_no,course_name,cost
  2  from course_info
  3  where cost in(select min(cost)
  4  from course_info);

 COURSE_NO COURSE_NAME                                              COST
---------- -------------------------------------------------- ----------
     50001 C programing                                             2000

SQL>
```

2. Display the name of the students enrolled in section no 8 and course number 20.

```
SQL> select student_first_name,student_last_name
  2  from student_info
  3  where student_id in
  4  (select student_id from enrollment_info
  5  where section_id in
  6  (select section_id from section_info
  7  where section_no=8 and course_no=20));

no rows selected

SQL>
```

3. Display the student id's who registered first.

```
SQL>  select student_id
  2   from enrollment_info
  3  where enrollment_date=(select min(enrollment_date) from enrollment_info ) and
  4  rownum=1;

STUDENT_ID
----------
     40001

SQL>
```

4. Display the course_no and sum of capacity where the total capacity is less than the average capacity.

```
SQL> select course_no,sum(capacity)
  2  from section_info
  3  group by course_no
  4  having sum(capacity) <
  5  (select avg(capacity)
  6  from section_info);

 COURSE_NO SUM(CAPACITY)
---------- -------------
     50003           100
     50004           100
     50001           120

SQL>
```

Subqueries returning multiple rows

1. Display the course_no and course_name whose cost is same as that of course whose prerequisite 20

```
SQL>  select C.course_no,C.course_name
  2   from course_info C
  3   where cost IN(select cost from course_info where course_prerequisite=20);

 COURSE_NO COURSE_NAME
---------- --------------------------------------------------
     50002 DBMS
     50001 C programing
```

2. Modify the above query to display the records whose cost is not same as that of the course whose prerequisite is 20.

```
SQL> select C.course_no,C.course_name
  2  from course_info C
  3  where cost NOT IN(select cost from course_info S where S.course_prerequisite=20);

 COURSE_NO COURSE_NAME
---------- --------------------------------------------------
     50004 Advance DBMS
     50005 Avance data structure
     50003 Data structure
```

3. Display the course description and cost of the courses where the capacity is less than or equal to the average capacity and cost is equal to minimum cost.

```
SQL>  select c.course_name,c.cost
  2   from course_info c,section_info s
  3  where s.capacity<=
  4  (select avg(capacity) from section_info)
  5  and c.cost=(select min(cost) from course_info);

COURSE_NAME                                          COST
-------------------------------------------------- ----------
C programing                                         2000
C programing                                         2000
C programing                                         2000

SQL>
```

4. Display the student id and section id who are living in zip 77016.

```
SQL> select student_id,section_id
  2  from enrollment_info
  3  where student_id in
  4  (select student_id
  5  from student_info
  6  where zip_code='77016');

STUDENT_ID SECTION_ID
---------- ----------
     40001      60001
```

5. Display the course no and course_name taught by instructor whose instructor ID =10001.

```
SQL>
SQL>  select course_no,course_name
  2   from course_info
  3   where course_no in
  4     ( select course_no
  5       from section_info
  6       where instructor_id='10001');

 COURSE_NO COURSE_NAME
---------- --------------------------------------------------
     50001 C programing
```

6. Display the students enrolled in course ‘Introduction to java’.

```
SQL>  select student_first_name,student_last_name from student_info s, enrollment_info e,
  2    course_info c,section_info s where s.student_id = e.student_id and e.section_id=s.section_id
  3  and s.course_no = c.course_no and c.course_name = 'Data structure';

STUDENT_FIRST_N STUDENT_LAST_NA
--------------- ---------------
Fury            english

SQL>
```

7. Display the last_name and enrollment_date from student_info of those students who have enrolled on 22nd of Jan 2001.

```
SQL> SELECT * FROM ENROLLMENT_INFO;

STUDENT_ID SECTION_ID ENROLLMENT
---------- ---------- ----------
     40001      60001 2020-11-11
     40002      60002 2020-11-13
     40004      60004 2020-11-20
     40003      60003 2020-11-16
     40005      60005 2020-11-22

SQL> SELECT S.STUDENT_LAST_NAME, E.ENROLLMENT_DATE
  2  FROM STUDENT_INFO S, ENROLLMENT_INFO E
  3  WHERE S.STUDENT_ID = E.STUDENT_ID AND ENROLLMENT_DATE = '2001-01-22';

no rows selected
```

Co-related subqueries with exists

2. Display the instructor details only if the instructor teaches a section.

```
SQL> select * from instructor_info where exists(
  2  select * from section_info where section_info.instructor_id = instructor_id
  3  );

INSTRUCTOR_ID INSTRUCTOR_FIRST__NAME    INSTRUCTOR_LAST__NAME
------------- ------------------------- -------------------------
STREET_ADDRESS                                       ZIP_C
-------------------------------------------------- -----
       10001 Ted                         Smith
895 Chestnut Ave.                                    77016

       10002 Bob                         Wong
32 Glen Creek Lane                                   11003

       10003 Ron                         Zelya
369 Vernon Dr                                        91316


INSTRUCTOR_ID INSTRUCTOR_FIRST__NAME    INSTRUCTOR_LAST__NAME
------------- ------------------------- -------------------------
STREET_ADDRESS                                       ZIP_C
-------------------------------------------------- -----
       10004 Joy                         Wallace
9680 E. Somerset Street                              94566

       10005 william                     jabbar
8459 W. Newport Court                                11762
```

3. Display the details of those instructors who do not teach a section.

```
SQL> select * from instructor_info where not exists(
  2  select * from section_info S where S.instructor_id = instructor_id)
  3  ;

no rows selected
```

4. Display the names of the students who are enrolled.

```
SQL> select STUDENT__FIRST_NAME, STUDENT__LAST_NAME from STUDENT_INFO where
  2  EXISTS
  3  (
  4  select * from enrollment_info E where E.STUDENT_ID = STUDENT_ID)
  5  ;

STUDENT__FIRST_ STUDENT__LAST_N
--------------- ---------------
James           Anderson
franklin        borg
Fury            english
Steve           Rogers
Tony            Stark
```

5. Display the courses enrolled by the students.

```
SQL> select student__first_name, course_name from student_info S, course_info C where exists(
  2  select * from section_info SE, enrollment_info E where
  3  SE.section_id = E.section_id and S.student_id = E.student_id and C.course_no = SE.course_no
  4  );

STUDENT__FIRST_ COURSE_NAME
--------------- --------------------------------------------------
James           C programing
franklin        DBMS
Fury            Data structure
Steve           Advance DBMS
Tony            Avance data structure
```

6. Display the courses who do not have sections.

```
SQL> select course_name from course_info where not exists(
  2  select * from section_info where
  3  section_info.course_no = course_no
  4  );

no rows selected
```

7. Display the sections in which no student is enrolled.

```
SQL>  select section_id from section_info S where
  2   NOT EXISTS
  3   (
  4  select * from enrollment_info E where
  5  E.section_id = S.section_id
  6  );

no rows selected
```

JOINS

1. Display the course and the section information.

```
SQL> select * from course_info natural join section_info;

 COURSE_NO COURSE_NAME
---------- --------------------------------------------------
COURSE_PREREQUISITE       COST SECTION_ID SECTION_NO INSTRUCTOR_ID
------------------- ---------- ---------- ---------- -------------
LOCATION                                             CAPACITY
-------------------------------------------------- ----------
     50001 C programing
                          2000      60001          1         10001
Astoria                                                     120

     50002 DBMS
                          3000      60002          2         10002
ongview                                                     150

 COURSE_NO COURSE_NAME
---------- --------------------------------------------------
COURSE_PREREQUISITE       COST SECTION_ID SECTION_NO INSTRUCTOR_ID
------------------- ---------- ---------- ---------- -------------
LOCATION                                             CAPACITY
-------------------------------------------------- ----------

     50003 Data structure
               50001       4500      60003          3         10003
Campbell                                                    100

     50004 Advance DBMS
               50002       5000      60004          4         10004

 COURSE_NO COURSE_NAME
---------- --------------------------------------------------
COURSE_PREREQUISITE       COST SECTION_ID SECTION_NO INSTRUCTOR_ID
------------------- ---------- ---------- ---------- -------------
LOCATION                                             CAPACITY
-------------------------------------------------- ----------
Richmond Hill                                               100

     50005 Avance data structure
               50003       5500      60005          5         10005
Pleasanton                                                  140
```

2. Display the instructor details along with the city and the state that they live in.

```
SQL> select * from instructor_info natural join zipcode_info;

ZIP_C INSTRUCTOR_ID INSTRUCTOR_FIRST__NAME    INSTRUCTOR_LAST__NAME
----- ------------- ------------------------ -------------------------
STREET_ADDRESS                                     CITY                     ST
-------------------------------------------------- ------------------------ --
11003         10002 Bob                        Wong
32 Glen Creek Lane                                 Elmont                   NY

11762         10005 william                    jabbar
8459 W. Newport Court                              Massapequa Park          NY

77016         10001 Ted                        Smith
895 Chestnut Ave.                                  Houston                  TX


ZIP_C INSTRUCTOR_ID INSTRUCTOR_FIRST__NAME    INSTRUCTOR_LAST__NAME
----- ------------- ------------------------ -------------------------
STREET_ADDRESS                                     CITY                     ST
-------------------------------------------------- ------------------------ --
91316         10003 Ron                        Zelya
369 Vernon Dr                                      Encino                   CA

94566         10004 Joy                        Wallace
9680 E. Somerset Street                            Pleasanton               CA
```

3. Display the student details of all students and city and state. Sort the result on the basis of zip code.

```
SQL> select * from student_info natural join zipcode_info order by zip_code;

ZIP_C STUDENT_ID STUDENT__FIRST_ STUDENT__LAST_N
----- ---------- --------------- ---------------
STREET_ADDRESS                                     CITY                      ST
-------------------------------------------------- ------------------------- --
11003      40002 franklin        borg
32 Glen Creek Lane                                 Elmont                    NY

11762      40005 Tony            Stark
8459 W. Newport Court                              Massapequa Park           NY

77016      40001 James           Anderson
895 Chestnut Ave.                                  Houston                   TX


ZIP_C STUDENT_ID STUDENT__FIRST_ STUDENT__LAST_N
----- ---------- --------------- ---------------
STREET_ADDRESS                                     CITY                      ST
-------------------------------------------------- ------------------------- --
91316      40003 Fury            english
369 Vernon Dr                                      Encino                    CA

94566      40004 Steve           Rogers
9680 E. Somerset Street                            Pleasanton                CA
```

4. Write a query to display the first and last names of only the enrolled students and order on the basis of last name in descending order.

```
SQL> select student__first_name, student__last_name from student_info natural join enrollment_info
  2  order by student__last_name desc;

STUDENT__FIRST_ STUDENT__LAST_N
--------------- ---------------
Fury            english
franklin        borg
Tony            Stark
Steve           Rogers
James           Anderson
```

5. Display the zip, city and state and instructor names who are living in state ‘NY’.

```
SQL> select  INSTRUCTOR_FIRST__NAME, INSTRUCTOR_LAST__NAME, zip_code, city, state
  2  from instructor_info natural join zipcode_info where state ='NY';

INSTRUCTOR_FIRST__NAME    INSTRUCTOR_LAST__NAME     ZIP_C
------------------------- ------------------------- -----
CITY                      ST
------------------------- --
Bob                       Wong                      11003
Elmont                    NY

william                   jabbar                    11762
Massapequa Park           NY
```

6. Write a query to display the student number, course number, enrollment date and section id for students who are enrolled in course number 20 on January 30 2000.

```
SQL>  select student_id, course_no,ENROLLMENT_DATE,section_id from section_info
  2    natural join enrollment_info
  3   where course_no =20 and ENROLLMENT_DATE='30-jan-2000';

no rows selected
```

7. Display the course number, description, cost, section id, and the instructor names who are teaching the course.

```
SQL> select course_no, course_name, cost, section_id, INSTRUCTOR_FIRST__NAME, INSTRUCTOR_LAST__NAME
  2  from(
  3  select * from course_info natural join section_info) natural join instructor_info;

 COURSE_NO COURSE_NAME                                        COST
---------- -------------------------------------------------- ----------
SECTION_ID INSTRUCTOR_FIRST__NAME    INSTRUCTOR_LAST__NAME
---------- ------------------------- -------------------------
     50001 C programing                                       2000
     60001 Ted                       Smith

     50002 DBMS                                               3000
     60002 Bob                       Wong

     50003 Data structure                                     4500
     60003 Ron                       Zelya


 COURSE_NO COURSE_NAME                                        COST
---------- -------------------------------------------------- ----------
SECTION_ID INSTRUCTOR_FIRST__NAME    INSTRUCTOR_LAST__NAME
---------- ------------------------- -------------------------
     50004 Advance DBMS                                       5000
     60004 Joy                       Wallace

     50005 Avance data structure                              5500
     60005 william                   jabbar
```

9. Display the student id, course number and section number of enrolled students where the instructor of the section lives in zip code 10025. Also, the course should not have any pre-requisites.

```
SQL> select student_id, section_id, course_no from(
  2  select * from instructor_info natural join (
  3  select * from course_info natural join(
  4  select * from section_info natural join enrollment_info)
  5  )
  6  ) where course_prerequisite is null  and
  7  zip_code= 10025;

no rows selected
```

10. Display the instructor details along with the city, state who are teaching the section 2.

```
SQL> select * from (
  2  select * from instructor_info natural join zipcode_info) zp where instructor_id in (
  3  select instructor_id from section_info where section_no = 2);

ZIP_C INSTRUCTOR_ID INSTRUCTOR_FIRST__NAME    INSTRUCTOR_LAST__NAME
----- ------------- ------------------------ ------------------------
STREET_ADDRESS                                    CITY                     ST
------------------------------------------------- ------------------------ --
11003         10002 Bob                      Wong
32 Glen Creek Lane                                Elmont                   NY
```

11. Display the list of enrolled student numbers who are living in state ‘CT’.

```
SQL> select * from student_info S where S.student_id in (
  2  select E.student_id from enrollment_info E where E.student_id in (
  3  select student_id from student_info  natural join zipcode_info  where
  4  state = 'CT')
  5  );

no rows selected
```

12. Write a query to display the details of student ‘ James Anderson' for section

```
SQL> select * from student_info S where S.student_id in(
  2  select student_id from student_info natural join enrollment_info
  3  where
  4  STUDENT__FIRST_NAME ='James' and STUDENT__LAST_NAME = 'Anderson'
  5  and SECTION_ID= 60001);

STUDENT_ID STUDENT__FIRST_ STUDENT__LAST_N
---------- --------------- ---------------
STREET_ADDRESS                                    ZIP_C
------------------------------------------------- -----
     40001 James           Anderson
895 Chestnut Ave.                                 77016
```

id 60001.

13. Display the final examination numeric grade and details of all enrolled students living in state 'NY'.

```
SQL> select * from student_info natural join(
  2   select * from student_info S natural join (
  3   select student_id, NUMERIC_GRADE from grade_info)
  4   )  where zip_code in (
  5   select zip_code from zipcode_info where state='NY'
  6   );

STUDENT_ID STUDENT__FIRST_ STUDENT__LAST_N
---------- --------------- ---------------
STREET_ADDRESS                                  ZIP_C NUMERIC_GRADE
----------------------------------------------- ----- -------------
     40002 franklin        borg
32 Glen Creek Lane                              11003             7

     40005 Tony            Stark
8459 W. Newport Court                           11762             8
```

Outer Join

1. Display the course number, description and pre-requisite along with section id. Those courses who do not have sections defined should also be displayed.

```
SQL> select course_name,course_info.course_no,course_prerequisite,cost,section_id
  2  from course_info
  3  left join section_info
  4  on course_info.course_no=section_info.course_no;

COURSE_NAME                                        COURSE_NO
-------------------------------------------------- ----------
COURSE_PREREQUISITE        COST SECTION_ID
------------------- ---------- ----------
C programing                                           50001
                           2000      60001

DBMS                                                   50002
                           3000      60002

Data structure                                         50003
               50001       4500      60003

COURSE_NAME                                        COURSE_NO
-------------------------------------------------- ----------
COURSE_PREREQUISITE        COST SECTION_ID
------------------- ---------- ----------
Advance DBMS                                           50004
               50002       5000      60004

Avance data structure                                  50005
               50003       5500      60005

computer network                                       50006
                           5000      60006

COURSE_NAME                                        COURSE_NO
-------------------------------------------------- ----------
COURSE_PREREQUISITE        COST SECTION_ID
------------------- ---------- ----------
software engineering                                   50007
                           5000
```

2. For the above query also include courses where no instructor is assigned.

```
SQL> select course_name,course_info.course_no,course_prerequisite,cost,section_id
  2  from course_info
  3  left join section_info
  4  on course_info.course_no=section_info.course_no and instructor_id is null;

COURSE_NAME                                        COURSE_NO
-------------------------------------------------- ----------
COURSE_PREREQUISITE        COST SECTION_ID
------------------- ---------- ----------
computer network                                       50006
                           5000      60006

DBMS                                                   50002
                           3000

Data structure                                         50003
               50001       4500


COURSE_NAME                                        COURSE_NO
-------------------------------------------------- ----------
COURSE_PREREQUISITE        COST SECTION_ID
------------------- ---------- ----------
Avance data structure                                  50005
               50003       5500

C programing                                           50001
                           2000

software engineering                                   50007
                           5000


COURSE_NAME                                        COURSE_NO
-------------------------------------------------- ----------
COURSE_PREREQUISITE        COST SECTION_ID
------------------- ---------- ----------
Advance DBMS                                           50004
               50002       5000

7 rows selected.

SQL>
```

Self Join

1. Many courses have pre-requisite. Display the course no and the description along with the prerequisite and the description. e.g. - 20 Intro to Java 10 Java Programming where 20 is the course no and 10 is the pre-requisite.

```
SQL> select c1.course_no,c1.course_name,c2.course_no,c2.course_name
  2   from course_info c1,course_info c2
  3   where c2.course_no=c1.course_prerequisite;

 COURSE_NO COURSE_NAME                                          COURSE_NO
---------- -------------------------------------------------- ----------
COURSE_NAME
--------------------------------------------------
     50003 Data structure                                          50001
C programing

     50004 Advance DBMS                                            50002
DBMS

     50005 Avance data structure                                   50003
Data structure


SQL>
```