UNIT 1 THE STRUCTURED QUERY LANGUAGE

Structure **Page Nos.**

1.0	Introduction	5
1.1	Objectives	5
1.2	What is SQL?	6
1.3	Data Definition Language	7
1.4	Data Manipulation Language	9
1.5	Data Control	15
1.6	Database Objects: Views, Sequences, Indexes and Synonyms	16
	1.6.1 Views	
	1.6.2 Sequences	
	1.6.3 Indexes and Synonyms	
1.7	Table Handling	19
1.8	Nested Queries	23
1.9	Summary	27
1.10	Solutions/Answer	28
1.11	Further Reading	34

1.0 INTRODUCTION

Database is an organised collection of information about an entity having controlled redundancy and serves multiple applications. DBMS (database management system) is an application software that is developed to create and manipulate the data in database. A query language can easily access a data in a database. SQL (Structured Query Language) is language used by most relational database systems. IBM developed the SQL language in mid-1979. All communication with the clients and the RDBMS, or between RDBMS is via SQL. Whether the client is a basic SQL engine or a disguised engine such as a GUI, report writer or one RDBMS talking to another, SQL statements pass from the client to the server. The server responds by processing the SQL and returning the results. The advantage of this approach is that the only network traffic is the initial query and the resulting response. The processing power of the client is reserved for running the application.

SQL is a data sub-language consisting of three built-in languages: Data definition language (DDL), Data manipulation language (DML) and Data control language (DCL). It is a fourth generation language. SQL has many more features and advantages. Let us discuss the SQL in more detail in this unit. It should be noted that many commercial DBMS may or may not implement all the details given in this unit. For example, MS-ACCESS does not support some of these features. Even some of the constructs may not be portable, please consult the DBMS Help for any such difference.

1.1 OBJECTIVES

After going through this unit, you should be able to:

- create, modify and delete database schema objects;
- update database using SQL command;
- retrieve data from the database through queries and sub-queries;
- handle join operations;
- control database access;
- deal with database objects like Tables, Views, Indexes, Sequences, and Synonyms using SQL.

1.2 WHAT IS SQL?

Structured Query Language (SQL) is a standard query language. It is commonly used with all relational databases for data definition and manipulation.

All the relational systems support SQL, thus allowing migration of database from one DBMS to another. In fact, this is one of the reasons of the major success of Relational DBMS. A user may be able to write a program using SQL for an application that involves data being stored in more than one DBMSs, provided these DBMS support standard SQL. This feature is commonly referred to as portability. However, not all the features of SQL implemented in one RDBMS are available in others because of customization of SQL. However, please note there is just ONE standard SQL.

SQL provides an interface where the user can specify "What" are the expected results. The query execution plan and optimisation is performed by the DBMS. The query plan and optimisation determines how a query needs to be executed. For example, if three tables X, Y and Z are to be joined together then which plan (X JOIN Y) and then Z or X JOIN (Y JOIN Z) may be executed. All these decisions are based on statistics of the relations and are beyond the scope of this unit.

SQL is called a non-procedural language as it just specifies what is to be dome rather than how it is to be done. Also, since SQL is a higher-level query language, it is closer to a language like English. Therefore, it is very user friendly.

The American National Standard Institute (ANSI) has designed standard versions of SQL. The first standard in this series was created in 1986. It was called SQL-86 or SQL1. This standard was revised and enhanced later and SQL-92 or SQL-2 was released. A newer standard of SQL is SQL3 which is also called SQL- 99. In this unit we will try to cover features from latest standards. However, some features may be found to be very specific to certain DBMSs.

Some of the important features of SQL are:

- It is a non procedural language.
- It is an English-like language.
- It can process a single record as well as sets of records at a time.
- It is different from a third generation language (C& COBOL). All SQL statements define what is to be done rather than how it is to be done.
- SQL is a data sub-language consisting of three built-in languages: Data definition language (DDL), Data manipulation language (DML) and Data control language (DCL).
- It insulates the user from the underlying structure and algorithm.
- SQL has facilities for defining database views, security, integrity constraints, transaction controls, etc.

There are many variants of SQL, but the standard SQL is the same for any DBMS environment. The following table shows the differences between SQL and one of its superset SQL*Plus which is used in Oracle. This will give you an idea of how various vendors have enhanced SQL to an environment. The non-standard features of SQL are not portable across databases, therefore, should not be used preferably while writing SQL queries.

Difference between SQL and SQL*Plus

SQL	**SQL*Plus**
SQL is a language	SQL *Plus is an environment
It is based on ANSI (American National Standards Institute) standard	It is oracle proprietary interface for executing SQL statements

SQL	
It is entered into the SQL buffer on one or more lines	It is entered one line at a time, not stored in the SQL buffer
SQL cannot be abbreviated	SQL*Plus can be abbreviated
It uses a termination character to execute command immediately	It does not require termination character. Commands are executed immediately.
SQL statements manipulate data and table definition in the database	It does not allow manipulation of values in the database

1.3 DATA DEFINITION LANGUAGE

As discussed in the previous block, the basic storage unit of a relational database management system is a table. It organises the data in the form of rows and columns. But what does the data field column of table store? How do you define it using SQL?

The Data definition language (DDL) defines a set of commands used in the creation and modification of schema objects such as tables, indexes, views etc. These commands provide the ability to create, alter and drop these objects. These commands are related to the management and administrations of the databases. Before and after each DDL statement, the current transactions are implicitly committed, that is changes made by these commands are permanently stored in the databases. Let us discuss these commands in more detail:

CREATE TABLE Command

Syntax:

```
CREATE TABLE <table name>(
Column_name1 data type (column width) [constraints],
Column_name2 data type (column width) [constraints],
Column_name3 data type (column width) [constraints],
………………………………………..
);
```

Where table name assigns the name of the table, column name defines the name of the field, data type specifies the data type for the field and column width allocates specified size to the field.

Guidelines for creation of table:

- Table name should start with an alphabet.
- In table name, blank spaces and single quotes are not allowed.
- Reserve words of that DBMS cannot be used as table name.
- Proper data types and size should be specified.
- Unique column name should be specified.

Column Constraints: NOT NULL, UNIQUE, PRIMARY KEY, CHECK, DEFAULT, REFERENCES,

On delete Cascade: Using this option whenever a parent row is deleted in a referenced table then all the corresponding child rows are deleted from the referencing table. This constraint is a form of referential integrity constraint.

Example 1:

```
CREATE TABLE product
   (
pno number (4) PRIMARY KEY,
pname char (20) NOT NULL,
qoh number (5) DEFAULT (100),
            class char (1) NOT NULL,
            rate number (8,2) NOT NULL,
            CHECK ((class='A' AND rate<1000) OR
            (class='B' AND rate>1000 AND rate<4500) OR
            (class='C' AND rate>4500))
        );
```

The command above creates a table. Primary key constraint ensures that product number (pno) is not null and unique (both are the properties of primary key). Please note the use of data type char (20). In many implementations of SQL on commercial DBMS like SQL server and oracle, a data type called varchar and varchar2 is used respectively. Varchar basically is variable length character type subject to a maximum specified in the declarations. We will use them at most of the places later.

Please note the use of check constraints in the table created above. It correlates two different attribute values.

Example 2:

```
        CREATE TABLE prodtrans
        (
        pno number (4)
        ptype char (1) CHECK (ptype in ('I','R','S')),
        qty number (5)
       FOREIGN KEY pno REFERENCES product (pno)
       ON DELETE CASCADE);
```

In the table so created please note the referential constraint on the foreign key **pno** in **prodtrans** table to **product** table. Any product record if deleted from the product table will trigger deletion of all the related records (ON DELETE CASCADE) in the **prodtrans** table.

ALTER TABLE Command: This command is used for modification of existing structure of the table in the following situation:

- When a new column is to be added to the table structure.
- When the existing column definition has to be changed, i.e., changing the width of the data type or the data type itself.
- When integrity constraints have to be included or dropped.
- When a constraint has to be enabled or disabled.

Syntax

```
ALTER TABLE <table name> ADD (<column name> <data type>…);
ALTER TABLE <table name> MODIFY (<column name> <data type>…);
ALTER TABLE <table name> ADD CONSTRAINT <constraint name> < constraint type>(field name);
ALTER TABLE <table name> DROP<constraint name>;
ALTER TABLE <table name> ENABLE/DISABLE <constraint name>;
```

You need to put many constraints such that the database integrity is not compromised.

Example 3:

```
       ALTER TABLE emp MODIFY (empno NUMBER (7));
```

DROP TABLE Command:

When an existing object is not required for further use, it is always better to eliminate it from the database. To delete the existing object from the database the following command is used.
Syntax:

```
DROP TABLE<table name>;
```

Example 4:

```
DROP TABLE emp;
```

1.4 DATA MANIPULATION LANGUAGE

Data manipulation language (DML) defines a set of commands that are used to query and modify data within existing schema objects. In this case commit is not implicit that is changes are not permanent till explicitly committed. DML statements consist of SELECT, INSERT, UPDATE and DELETE statements.

SELECT Statement

This statement is used for retrieving information from the databases. It can be coupled with many clauses. Let us discuss these clauses in more detail:

1. **Using Arithmetic operator**

Example 5:

```
SELECT ENAME, SAL, SAL+300
 FROM EMP;
```

2. **Operator Precedence**

The basic operators used in SQL are * / + -
Operators of the same priority are evaluated From Left to right
Parentheses are used to force prioritized evaluation.

Example 6:

```
SELECT ENAME, SAL, 12*SAL+100
FROM EMP;

SELECT ENAME, SAL, 12*(SAL+100)
FROM EMP;
```

3. **Using Column aliases**

Example 7:

To print column names as NAME and ANNUAL SALARY

```
SELECT ENAME “NAME”, SAL*12 “ANNUAL SALARY”
FROM EMP;
```

4. **Concatenation operator**

Example 8:

Printing name and job as one string as column name employees:

```
SELECT ENAME||JOB “EMPLOYEES”
FROM EMP;
```

5. **Using Literal Character String**

Example 9:

To print <name> IS A <job> as one string with column name employee

```
SELECT ENAME || ‘ IS A ’ || JOB AS “EMPLOYEE”
FROM EMP;
```

6. **To eliminate duplicate rows (distinct operator)**

Example 10:

```
SELECT DISTINCT DEPTNO
FROM EMP;
```

7. **Special comparison operator** used in where Clause

   a. **between. …and…**It gives range between two values (inclusive)

Example 11:

```
SELECT ENAME, SAL
FROM EMP
WHERE SAL BETWEEN 1000 AND 1500;
```

   b. **In (list):** match any of a list of values

Example 12:

```
SELECT EMPNO, ENAME, SAL, MGR
FROM EMP
WHERE MGR IN (7902, 7566, 7788);
```

7902, 7566, and 7788 are Employee numbers

   c. **Like:** match a character pattern

- Like operator is used only with char and Varchar2 to match a pattern
- % Denotes zero or many characters
- _ Denotes one character
- Combination of % and_can also be used

Example 13:

(I) List the names of employees whose name starts with 's'

```
SELECT ENAME FROM EMP
WHERE ENAME LIKE 'S%';
```

(II) List the ename ending with 's'

```
SELECT ENAME FROM EMP
WHERE ENAME LIKE '%S';
```

(III) List ename having I as a second character

```
SELECT ENAME FROM EMP
WHERE ENAME LIKE '_I%';
```

d. **Is null operator**

Example 14:

to find employee whose manage-id is not specified

```
SELECT ENAME, MGR FROM EMP
WHERE MGR IS NULL;
```

8 **Logical Operators**

Rules of Precedence:

Order evaluated	Operator
1	All comparison operators

2	NOT
3	AND
4	OR

Example 15: To print those records of salesman or president who are having salary above 15000/-

```
Select ename, job, sal from emp
Where job = 'SALESMAN' or job = 'PRESIDENT'
And sal>15000;
```

The query formulation as above is incorrect for the problem statement. The correct formulation would be:

```
SELECT ENAME, JOB, SAL FROM EMP
WHERE (JOB = 'SALESMAN' OR JOB = 'PRESIDENT')
AND SAL>15000;
```

9. **Order by clause**

- It is used in the last portion of select statement
- By using this rows can be sorted
- By default it takes ascending order
- DESC: is used for sorting in descending order
- Sorting by column which is not in select list is possible
- Sorting by column Alias

Example 16:

```
SELECT EMPNO, ENAME, SAL*12 "ANNUAL"
FROM EMP
ORDER BY ANNUAL;
```

Example 17: Sorting by multiple columns; ascending order on department number and descending order of salary in each department.

```
SELECT ENAME, DEPTNO, SAL
FROM EMP
ORDER BY DEPTNO, SAL DESC;
```

10. **Aggregate functions**

- Some of these functions are count, min, max, avg.
- These functions help in getting consolidated information from a group of tuples.

Example 18: Find the total number of employees.

```
SELECT COUNT(*)
FROM EMP;
```

Example 19: Find the minimum, maximum and average salaries of employees of department D1.

```
SELECT MIN(SAL), MAX(SAL), AVG(SAL)
FROM EMP
WHERE DEPTNO = 'D1' ;
```

11. **Group By clauses**

- It is used to group database tuples on the basis of certain common attribute value such as employees of a department.
- WHERE clause still can be used, if needed.

Example 20: Find department number and Number of Employees working in that department.

```
SELECT DEPTNO, COUNT(EMPNO)
FROM EMP
GROUP BY DEPTNO;
```

Please note that while using group by and aggregate functions the only attributes that can be put in select clause are the aggregated functions and the attributes that have been used for grouping the information. For example, in the example 20, we cannot put ENAME attribute in the SELECT clause as it will not have a distinct value for the group. Please note the group here is created on DEPTNO.

12. Having clause

- This clause is used for creating conditions on grouped information.

Example 21: Find department number and maximum salary of those departments where maximum salary is more than Rs 20000/-.

```
SELECT DEPTNO, MAX(SAL)
FROM EMP
GROUP BY DEPTNO
HAVING MAX(SAL) > 20000;
```

INSERT INTO command:

- Values can be inserted for all columns or for the selected columns
- Values can be given through sub query explained in section 1.8
- In place of values parameter substitution can also be used with insert.
- If data is not available for all the columns, then the column list must be included following the table name.

Example 22: Insert the employee numbers, an increment amount of Rs.500/- and the increment date-today (which is being entered through function SYSDATE) of all the managers into a table INCR (increment due table) from the employee file.

```
INSERT INTO INCR
SELECT EMPNO, 500, SYSDATE FROM EMP
WHERE JOB = ‘MANAGER’;
```

Example 23: Insert values in a table using parameter substitution (& operator is used for it 1, 2 are the field numbers).

```
INSERT INTO EMP
VALUES (&1,’&2’,’&3’, &4, &5, &6,NULL, &7);
```

Please note these values needs to be supplied at run time.

UPDATE Command:

Syntax is

```
UPDATE <table name>
SET <column name> = <value>
WHERE <condition>;
```

Sub query in the UPDATE command:

Example 24: Double the commission for employees, who have got at least 2 increments.

```
UPDATE EMP
SET COMM = COMM * 2
WHERE 2 <= (            SELECT COUNT (*) FROM INCR
WHERE INCR.EMPNO = EMP.EMPNO
GROUP BY EMPNO);
```

Please note the use of subquery that counts the number of increments given to each employee stored in the INCR table. Please note the comparison, instead of ……>=2, we have written reverse of it as 2 <= …..

DELETE Command

In the following example, the deletion will be performed in EMP table.No deletion will be performed in the INCR table.

Example 25: Delete the records of employees who have got no increment.

```
    DELETE FROM EMP
    WHERE EMPNO NOT IN (SELECT EMPNO FROM INCR);
```

☞ Check Your Progress 1

1) What are the advantages of SQL? Can you think of some disadvantages of SQL?

……………………………………………………………………………………….
……………………………………………………………………………………….
……………………………………………………………………………………….
………………………………………………………………………………………..

2) Create the Room, Booking, and Guest tables using the integrity enhancement features of SQL with the following constraints:
(a) Type must be one of Single, Double, or Family.
(b) Price must be between Rs.100/- and Rs.1000/-.
(c) roomNo must be between 1 and 100.
(d) booking dateFrom and dateTo must be greater than today's date.
(e) The same room cannot be double booked.
(f) The same guest cannot have overlapping bookings.

……………………………………………………………………………………….
……………………………………………………………………………………….
……………………………………………………………………………………….
……………………………………………………………………………………….
……………………………………………………………………………………….
………………………………………………………………………………………..

3) Define the function of each of the clauses in the SELECT statement. What are the restrictions imposed on these clauses?

………………………………………………………………………………………..
………………………………………………………………………………………..
……………………………………………………………………………………….

…………………………………………………………………………………….

……………………………………………………………………………………..

4) Consider the supplier relations.

S

SNO (Supplier Number)	**SNAME (Supplier Name)**	**STATUS**	**CITY**
S1	Prentice Hall	30	Calcutta
S2	McGraw Hill	30	Mumbai
S3	Wiley	20	Chennai
S4	Pearson	40	Delhi
S5	Galgotia	10	Delhi

SP

SNO	**PNO (Part Number)**	**Quantity**
S1	P1	300
S1	P2	200
S2	P1	100
S2	P2	400
S3	P2	200
S4	P2	200

a) Get supplier numbers for suppliers with status > 20 and city is Delhi
b) Get Supplier Numbers and status for suppliers in Delhi in descending order of status.
c) Get all pairs of supplier numbers such that the two suppliers are located in the same city. (Hint: It is retrieval involving join of a table with itself.)
d) Get unique supplier names for suppliers who supply part P2 without using IN operator.
e) Give the same query above by using the operator IN.
f) Get part numbers supplied by more than one supplier. (Hint : It is retrieval with sub-query block, with inter block reference and same table involved in both blocks)
g) Get supplier numbers for suppliers who are located in the same city as supplier S1. (Hint: Retrieval with sub query and unqualified comparison operator).
h) Get supplier names for suppliers who supply part P1. (Hint : Retrieval using EXISTS)
i) Get part numbers for parts whose quantity is greater than 200 or are currently supplied by S2. (Hint: It is a retrieval using union).
j) Suppose for the supplier S5 the value for status is NULL instead of 10. Get supplier numbers for suppliers greater than 25. (Hint: Retrieval using NULL).
k) Get unique supplier numbers supplying parts. (Hint: This query is using the built-in function count).
l) For each part supplied, get the part number and the total quantity supplied for that part. (Hint: The query using GROUP BY).
m) Get part numbers for all parts supplied by more than one supplier. (Hint: It is GROUP BY with HAVING).
n) For all those parts, for which the total quantity supplied is greater than 300, get the part number and the maximum quantity of the part supplied in a single supply. Order the result by descending part number within those maximum quantity values.
o) Double the status of all suppliers in Delhi. (Hint: UPDATE Operation).
p) Let us consider the table TEMP has one column, called PNO. Enter into TEMP part numbers for all parts supplied by S2.
q) Add part p7.
r) Delete all the suppliers in Mumbai and also the suppliers concerned.

………………………………………………………………………………………………
………………………………………………………………………………………………

1.5 DATA CONTROL

The data control basically refers to commands that allow system and data privileges to be passed to various users. These commands are normally available to database administrator. Let us look into some data control language commands:

Create a new user:

```
CREATE USER < user name > IDENTIFIED BY < Password>
```

Example 26:

```
CREATE USER MCA12 IDENTIFIED BY W123
```

Grant: It is used to provide database access permission to users. It is of two types (1) system level permission (2) Object level permission.

Example 27:

```
GRANT CREATE SESSION TO MCA12;
        (This command provides system level permission on creating a
          session – not portable)

GRANT SELECT ON EMP TO MCA12;
        (Object level permission on table EMP)

GRANT SELECT, INSERT, UPDATE, DELETE ON EMP TO MCA12;

 GRANT SELECT, UPDATE ON EMP TO MCA12, MCA13;
        (Two users)
GRANT ALL ON EMP TO PUBLIC;
         (All permission to all users, do not use it. It is very dangerous for
          database)
```

Revoke: It is used to cancel the permission granted.

Example 28:

```
 REVOKE ALL ON EMP FROM MCA12;
        (All permissions will be cancelled)
 You can also revoke only some of the permissions.
```

Drop: A user-id can be deleted by using drop command.

Example 29:

```
 DROP USER MCA12;
```

Accessing information about permissions to all users

(1) Object level permissions: With the help of data dictionary you can view the permissions to user. Let us take the table name from oracle. In oracle the name of the table containing these permissions is user_tab_privs.

```
DESCRIBE USER_TAB_PRIVS ;
SELECT * FROM USER_TAB_PRIVS;
```

(2) System level permissions: With the help of data dictionary you can see them. Let us take the table name as user_sys_privs (used in oracle).

```
DESCRIBE USER_SYS_PRIVS ;
```

```
SELECT * FROM USER_SYS_PRIVS ;
```

All these commands are very specific to a data base system and may be different on different DBMS.

1.6 DATABASE OBJECTS: VIEWS, SEQUENCES, INDEXES AND SYNONYMS

Some of the important concepts in a database system are the views and indexes. Views are a mechanism that can support data independence and security. Indexes help in better query responses. Let us discuss about them along with two more concepts sequences and synonyms in more details.

1.6.1 Views

A view is like a window through which data from tables can be viewed or changed. The table on which a view is based is called Base table. The view is stored as a SELECT statement in the data dictionary. When the user wants to retrieve data, using view. Then the following steps are followed.

1) View definition is retrieved from data dictionary table. For example, the view definitions in oracle are to be retrieved from table name USER-VIEWS.
2) Checks access privileges for the view base table.
3) Converts the view query into an equivalent operation on the underlying base table

Advantages:

- It restricts data access.
- It makes complex queries look easy.
- It allows data independence.
- It presents different views of the same data.

Type of views:

Simple views and Complex Views

Feature	Simple Views	Complex Views
Number of tables	One	One or more
Contain Functions	No	Yes
Contain groups of data	No	Yes
Data Manipulation	IS allowed	Not always

Let us look into some of the examples. To see all the details about existing views in Oracle:

```
SELECT* FROM USER_VIEWS;
```

Creating a view:

- A query can be embedded within the CREATE VIEW STATEMENT
- A query can contain complex select statements including join, groups and sub-queries
- A query that defines the view **cannot contain** an order by clause.
- DML operation (delete/modify/add) cannot be applied if the view contains any of the following:

Delete (You can't delete if view contains following)	*Modify (you cannot modify if view contains following)*	*Insert (you cannot insert if view contains following)*
• Group functions • A group by clause • A distinct keyword	• Group functions • A group by clause • A distinct keyword • Columns defined by Expressions	• Group functions • A group by clause • A distinct keyword • Columns defined by Expressions • There are Not Null Columns in the base tables that are not selected by view.

Example 30: Create a view named employee salary having minimum, maximum and average salary for each department.

```
CREATE VIEW EMPSAL (NAME, MINSAL, MAXSAL, AVGSAL) AS
        SELECT D.DNAME, MIN(E.SAL),MAX(E.SAL),AVG(E.SAL)
        FROM EMP E, DEPT D
        WHERE E.DEPTNO=D.DEPTNO
         GROUP BY D.DNAME;
```

To see the result of the command above you can give the following command:

```
SELECT * FROM EMPSAL;
```

You may get some sample output like:

```
NAME             MINSAL    MAXSA     AVGSAL
--------------    ---------    ---------    -------------
ACCOUNTING       1300        5000       2916.6667
RESEARCH          800        3000       2175
SALES             950         2850      1566.6667
```

To see the structure of the view so created, you can give the following command:

```
DESCRIBE EMPSAL;
 Name                              Null?   Type
---------------                 ------------   ---------------------
 NAME                                     VARCHAR2 (14)
 MINSAL                                    NUMBER
 MAXSAL                                     NUMBER
 AVGSAL                                    NUMBER
```

Creating views with check option: This option restricts those updates of data values that cause records to go off the view. The following example explains this in more detail:

Example 31: To create a view for employees of Department = 30, such that user cannot change the department number from the view:

```
CREATE OR REPLACE VIEW EMPD30 AS
        SELECT EMPNO EMPL_NUM, ENAME NAME, SAL SALARY
        FROM EMP
        WHERE DEPTNO=30
        WITH CHECK OPTION;
```

Now the user cannot change the department number of any record of view EMPD30. If this change is allowed then the record in which the change has been made will go off the view as the view is only for department-30. This restricted because of use of WITH CHECK OPTION clause

Creating views with Read only option: In the view definition this option is used to ensure that no DML operations can be performed on the view.

Creating views with Replace option: This option is used to change the definition of the view without dropping and recreating it or regranting object privileges previously granted on it.

1.6.2 Sequences

Sequences:

- automatically generate unique numbers
- are sharable
- are typically used to create a primary key value
- replace application code
- speed up the efficiency of accessing sequence Values when cached in memory.

Example 32: Create a sequence named SEQSS that starts at 105, has a step of 1 and can take maximum value as 2000.

```
CREATE SEQUENCE SEQSS
START WITH 105
INCREMENT BY 1
 MAX VALUE 2000;
```

How the sequence so created is used? The following sequence of commands try to demonstrate the use of the sequence SEQSS.

```
Suppose a table person exists as:
          SELECT * FROM PERSON;
Output:          CODE      NAME        ADDRESS
                   ------   -------------- ---------------
                    104      RAMESH    MUMBAI
Now, if we give the command:
          INSERT INTO PERSON
                    VALUES (SEQSS.NEXTVAL, &NAME, &ADDRESS)
On execution of statement above do the following input:
          Enter value for name: 'Rakhi'
          Enter value for address: 'New Delhi'
Now, give the following command to see the output:
           SELECT * FROM PERSON;
            CODE NAME          ADDRESS
          --------- -------------- -----------------
               104 RAMESH        MUMBAI
               105 Rakhi             NEW DELHI
```

The descriptions of sequences such as minimum value, maximum value, step or increment are stored in the data dictionary. For example, in oracle it is stored in the table user_sequences. You can see the description of sequences by giving the SELECT command.

Gaps in sequence values can occur when:

- A rollback occurs that is when a statement changes are not accepted.
- The system crashes
- A sequence is used in another table.

Modify a sequence:

```
ALTER SEQUENCE SEQSS
INCREMENT 2
MAXVALUE 3000;
```

Removing a sequence:

```
DROP SEQUENCE SEQSS;
```

1.6.3 Indexes and Synonyms

Some of the basic properties of indexes are:

- An Index is a schema Object
- Indexes can be created explicitly or automatically
- Indexes are used to speed up the retrieval of rows
- Indexes are logically and physically independent of the table. It means they can be created or dropped at any time and have no effect on the base tables or other indexes.
- However, when a table is dropped corresponding indexes are also dropped.

Creation of Indexes

Automatically: When a primary key or Unique constraint is defined in a table definition then a unique index is created automatically.

Manually: User can create non-unique indexes on columns to speed up access time to rows.

Example 33: The following commands create index on employee name and employee name + department number respectively.

```
CREATE INDEX EMP_ENAME_IDX ON EMP (ENAME);
CREATE INDEX EMP_MULTI_IDX ON EMP (ENAME, DEPTNO);
```

Finding details about created indexes: The data dictionary contains the name of index, table name and column names. For example, in Oracle a user-indexes and user-ind-columns view contains the details about user created indexes.

Remove an index from the data dictionary:

```
DROP INDEX EMP_ENAME_IDX;
```

Indexes cannot be modified.

Synonyms

It permits short names or alternative names for objects.

Example 34:

```
CREATE SYNONYM D30
FOR EMPD30;
```

Now if we give command:

```
SELECT * FROM D30;
```

The output will be:

```
NAME                 MINSAL   MAXSAL   AVGSAL
-------------------- ------------ ------------- -------------
ACCOUNTING           1300     5000     2916.6667
RESEARCH              800     3000     2175
SALES                 950     2850     1566.6667
```

Removing a Synonym:

```
DROP SYNONYM D30;
```

1.7 TABLE HANDLING

In RDBMS more than one table can be handled at a time by using join operation. Join operation is a relational operation that causes two tables with a common domain to be combined into a single table or view. SQL specifies a join implicitly by referring the matching of common columns over which tables are joined in a WHERE clause. Two tables may be joined when each contains a column that shares a common domain with the other. The result of join operation is a single table. Selected columns from all the tables are included. Each row returned contains data from rows in the different input tables where values for the common columns match. An important rule of table handling is that there should be one condition within the WHERE clause for each pair of tables being joined. Thus if two tables are to be combined, one condition would be necessary, but if three tables (X, Y, Z) are to be combined then two conditions would be necessary because there are two pairs of tables (X-Y and Y-Z) OR (X-Z and Y-Z), and so forth. There are several possible types of joins in relational database queries. Four types of join operations are described below:

(1) **Equi Join**: A join in which the joining condition is based on equality between values in the common columns. Common columns appear (redundantly) in the result table. Consider the following relations:

- customer (custid, custname, ………..) and
- order (custid, ordered,………………..)

Example 35: What are the names of all customers who have placed orders?

The required information is available in two tables, customer and order. The SQL solution requires joining the two table using equi join.

```
SELECT CUSTOMER.CUTOID, ORDER.CUSTOID,
                                        CUSTONAME, ORDERID
FROM CUSTOMER, ORDER
WHERE CUSTOMER.CUSTOID=ORDER.CUSTOID;
```

The output may be:

```
Customer.custoid          order.custoid    custoname              orderid
-------------------------  ------------------  ----------------------  ----------
10                        10               Pooja Enterprises      1001
12                        12               Estern Enterprises     1002
3                         3                Impressions            1003
```

(2) **Natural Join**: It is the same like Equi join except one of the duplicate columns is eliminated in the result table. The natural join is the most commonly used form of join operation.

Example 36:

```
SELECT CUSTOMER.CUTOID, CUSTONAME, ORDERID
FROM CUSTOMER, ORDER
WHERE CUSTOMER.CUSTOID=ORDER.CUSTOID;
```

Output:

```
custoid                 custoname            orderid
-----------             -------------------- ----------
     10                 Pooja Enterprises      1001
     12                 Estern Enterprises     1002
     3                  Impressions            1003
```

(3) **Outer Join**: The use of Outer Join is that it even joins those tuples that do not have matching values in common columns are also included in the result table. Outer join places null values in columns where there is not a match between

tables. A condition involving an outer join is that it cannot use the IN operator or cannot be linked to another condition by the OR operator.

Example 37: The following is an example of left outer join (which only considers the non-matching tuples of table on the left side of the join expression).

```
SELECT CUSTOMER.CUTOID, CUSTONAME, ORDERID
FROM CUSTOMER LEFT OUTER JOIN ORDER
WHERE CUSTOMER.CUSTOID =   ORDER.CUSTOID;
```

Output: The following result assumes a CUSTID in CUSTOMER table who have not issued any order so far.

```
  CUSTOID              CUSTONAME   ORDERID
 -------------------   --------------------- ------------
     10                Pooja Enterprises     1001
     12                Estern Enterprises    1002
     3                 Impressions           1003
     15                South Enterprises    NULL
```

The other types of outer join are the Right outer join or complete outer join.

(4) **Self-Join**: It is a join operation where a table is joined with itself. Consider the following sample partial data of EMP table:

EMPNO	ENAME	MGRID	…..
1	Nirmal	4	
2	Kailash	4	
3	Veena	1	
4	Boss	NULL	
…..	…..	…	

Example 38: Find the name of each employee's manager name.

```
SELECT WORKER.ENAME || 'WORK FOR' || MANAGER.ENAME
FROM EMP WORKER, EMP MANAGER
WHERE WORKER.MGR=MANAGER.EMPNO;
```

Output:

Nirmal works for Boss
Kailash works for Boss
Veena works for Nirmal

☞ Check Your Progress 2

1) Discuss how the Access Control mechanism of SQL works.

……………………………………………………………………………………..

……………………………………………………………………………………..

……………………………………………………………………………………..

……………………………………………………………………………………..

2) Consider Hotel schema consisting of three tables Hotel, Booking and Guest,

```
CREATE TABLE Hotel
        hotelNo       HotelNumber       NOT NULL,
        hotelName     VARCHAR(20)       NOT NULL,
        city          VARCHAR(50)       NOT NULL,

        PRIMARY KEY (hotelNo));
```

```
CREATE TABLE Booking(
hotelNo        HotelNumbers        NOT NULL,
guestNo        GuestNumbers        NOT NULL,
dateFrom       BookingDate         NOT NULL,
dateTo         BookingDate         NULL,
roomNo         RoomNumber          NOT NULL,

PRIMARY KEY (hotelNo, guestNo, dateFrom),
FOREIGN KEY (hotelNo) REFERENCES Hotel
        ON DELETE CASCADE ON UPDATE CASCADE,
FOREIGN KEY (guestNo) REFERENCES Guest
        ON DELETE NO ACTION ON UPDATE CASCADE,
FOREIGN KEY (hotelNo, roomNo) REFERENCES Room
        ON DELETE NO ACTION ON UPDATE CASCADE);
CREATE TABLE Guest(
guestNo        GuestNumber         NOT NULL,
guestName      VARCHAR(20)         NOT NULL,
guestAddress   VARCHAR(50)         NOT NULL
PRIMARY KEY (guestno));

CREATE TABLE Room(
roomNo         RoomNumber          NOT NULL,
hotelNo        HotelNumbers        NOT NULL,
type           RoomType            NOT NULL DEFAULT 'S'
price          RoomPrice           NOT NULL,
PRIMARY KEY (roomNo, hotelNo),
FOREIGN KEY (hotelNo) REFERENCES Hotel
ON DELETE CASCADE ON UPDATE CASCADE);
```

Create a view containing the hotel name and the names of the guests staying at the hotel.

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

3) Give the users Manager and Director full access to views HotelData and BookingOutToday, with the privilege to pass the access on to other users.

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

…………………………………………………………………………………………

1.8 NESTED QUERIES

By now we have discussed the basic commands including data definition and data manipulations. Now let us look into some more complex queries in this section.

Sub-queries

Some of the basic issues of sub-queries are:

- A sub-query is a SELECT statement that is embedded in a clause of another SELECT statement. They are often referred to as a NESTED SELECT or SUB SELECT or INNER SELECT.

- The sub-query (inner query) executes first before the main query. The result of the sub-query is used by the main query (outer query).

- Sub-query can be placed in WHERE or HAVING or FROM clauses.

- Format of using sub-queries:
```
SELECT<select_list>
FROM<table>
WHERE expr OPERATOR
            (SELECT <select_list>
            FROM <TABLE>WHERE);
```
  Operator includes a comparison operator (single or multiple row operators)
  Single row operators: >, =, >=, <, <=, <>
  Multiple row operators: IN, ANY, ALL

- Order by clause cannot be used in sub-query, if specified it must be the last clause in the main select statement.

- Types of sub-queries:

  - ❑ Single row sub-query: It returns only one row from the inner select statement.
  - ❑ Multiple row sub-queries: it returns more than one row from the inner select statement
  - ❑ Multiple column sub-queries: it returns more than one column from the inner select statement.

  Single row operators are used with single row sub queries and multiple row operators are used with multiple row sub queries.

- The Outer and Inner queries can get data from different tables.

- Group Functions can be used in sub queries.

Consider the following partial relation EMP. Let us create some sub-queries for them

EMPNO	ENAME	JOB	SAL	DEPTNO
7566	Nirmal	MANAGER	2975	10
7788	Kailash	ANALYST	3000	10
7839	Karuna	PRESIDENT	5000	20
7902	Ashwin	ANALYST	3000	20
7905	Ashwini	MANAGER	4000	20

Example 39: Get the details of the person having the minimum salary.

```
SELECT ENAME, JOB, SAL
FROM EMP
 WHERE SAL =  (       SELECT MIN (SAL)
                      FROM EMP);
```

Output:

ENAME	JOB	SAL
Nirmal	MANAGER	2975

Example 40: Display the employees whose job title is the same as that of employee 7566 and salary is more than the salary of employee 7788.

```
SELECT ENAME, JOB
FROM EMP
WHERE JOB  =    (      SELECT JOB
                       FROM EMP
                       WHERE EMPPNO = 7566)
          AND SAL >  ( SELECT SAL
                       FROM EMP
                       WHERE EMPPNO=7788);
```

Output: Job title for the employee 7566 happens to be 'MANAGER')

ENAME	JOB
Ashwini	MANAGER

Having Clause with sub queries: First we recollect the GROUP BYclause. The following query finds the minimum salary in each department.

```
SELECT DEPTNO, MIN(SAL)
FROM.EMP
GROUP BY DEPTNO;
```

Output:

DEPTNO	SAL
10	2975
20	3000

Example 41: To find the minimum salary in those departments whose minimum salary is greater than minimum salary of department number 10.

```
SELECT DEPTNO, MIN(SAL)
FROM EMP
GROUP BY DEPTNO
HAVING MIN(SAL) >  (         SELECT MIN (SAL)
                             FROM EMP
                             WHERE DEPTNO = 10);
```

Output:

DEPTNO	SAL
20	3000

Example 42: Find the name, department number and salary of employees drawing minimum salary in that department.

```
SELECT ENAME, SAL, DEPTNO
FROM EMP
WHERE SAL IN (SELECT MIN (SAL)
                    FROM EMP
                    GROUP BY DEPTNO);
```

Output:

ENAME	SAL	DEPTNO
Nirmal	2975	10
Ashwin	3000	20

Find the salary of employees employed as an ANALYST
SELECT SAL FROM EMPWHERE JOB= ' ANALYST '

Output:

SAL
3000
3000

Example 43: Find the salary of employees who are not 'ANALYST' but get a salary less than or equal to any person employed as 'ANALYST'.

```
SELECT EMPNO, ENAME, JOB, SAL
FROMEMP
WHERE SAL <= ANY ( SELECT SAL
                        FROM EMP WHERE JOB = 'ANALYST' )
   AND JOB<>'ANALYST' ;
```

Output:

EMPNO	ENAME	JOB	SAL
7566	Nirmal	MANAGER	2975

Find the average salary in each department

SELECT DEPTNO, AVG(SAL) FROM EMP GROUP BY DEPTNO;
Result:

DEPTNO	SAL
10	2987.5
20	4000

Example 44: Find out the employee who draws a salary more than the average salary of all the departments.

```
SELECT EMPNO, ENAME, JOB, SAL
FROM EMP
WHERE SAL> ALL (SELECT AVG (SAL)
                    FROM EMP
                    GROUP BY DEPTNO);
```

Output:

EMPNO	ENAME	JOB	SAL
7839	Karuna	PRESIDENT	5000

Example 45: Find the employee name, salary, department number and average salary of his/her department, for those employees whose salary is more than the average salary of that department.

```
SELECT A.ENAME, A.SAL, A.DEPTNO, B.AVGSAL
FROM EMP A, (  SELECT DEPTNO, AVG (SAL)  AVGSAL
               FROM EMP
               GROUP BY DEPTNO) B
WHERE A.DEPTNO=B.DEPTNO AND A.SAL> B. AVGSAL;
```

Output:

ENAME	SAL	DEPTNO	AVGSAL
Kailash	3000	10	2987.5
Karuna	5000	20	4000

Multiple column Queries:
Syntax:

```
SELECT COLUMN1, COL2,……
FROM TABLE
WHERE (COLUMN1, COL2, …) IN
                    (SELECT COLUMN1, COL2,….
                     FROM TABLE
                     WHERE <CONDITION>);
```

Example 46: Find the department number, name, job title and salary of those people who have the same job title and salary as those are in department 10.

```
SELECT DEPTNO,ENAME, JOB, SAL
FROM EMP
WHERE (JOB, SAL) IN  (       SELECT JOB, SAL
                             FROM EMP
                             WHERE DEPTNO=10);
```

Output:

DEPTNO	ENAME	JOB	SAL
10	Nirmal	MANAGER	2975
10	Kailash	ANALYST	3000
20	Ashwin	ANALYST	3000

☞ Check Your Progress 3

1) What is the difference between a sub-query and a join? Under what circumstances would you not be able to use a sub-query?

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

2) Use the Hotel schema defined in question number 2 (Check Your Progress 2) and answer the following queries:

- List the names and addresses of all guests in Delhi, alphabetically ordered by name.
- List the price and type of all rooms at the GRAND Hotel.
- List the rooms that are currently unoccupied at the Grosvenor Hotel.
- List the number of rooms in each hotel.
- What is the most commonly booked room type for hotel in Delhi?
- Update the price of all rooms by 5%.

………………………………………………………………………………………

………………………………………………………………………………………

………………………………………………………………………………………

3) Consider the following Relational database.

Employees (eno, ename, address, basic_salary)
Projects (Pno, Pname, enos-of-staff-alotted)
Workin (pno, eno, pjob)

Two queries regarding the data in the above database have been formulated in SQL. Describe the queries in English sentences.

(i)
```
SELECT ename
FROM employees
WHERE eno IN ( SELECT eno
               FROM workin
               GROUP BY eno
 HAVING COUNT (*) =  (SELECT COUNT (*) FROM projects));
```

(ii)
```
SELECT Pname
FROM projects
WHERE Pno IN ( SELECT Pno
FROM projects
MINUS
GROUP BY eno
(SELECT DISTINCT Pno FROM workin));
```

…………………………………………………………………………………..
…………………………………………………………………………………..
………………………………………………………………………………….
…………………………………………………………………………………..
…………………………………………………………………………………..
………………………………………………………………………………….

1.9 SUMMARY

This unit has introduced the SQL language for relational database definition, manipulation and control. The SQL environment includes an instance of an SQL DBMS with accessible databases and associated users and programmers. Each schema definition that describes the database objects is stored in data dictionary/ system catalog. Information contained in system catalog is maintained by the DBMS itself, rather than by the users of DBMS.

The data definition language commands are used to define a database, including its creation and the creation of its tables, indexes and views. Referential integrity constraints are also maintained through DDL commands. The DML commands of SQL are used to load, update and query the database. DCL commands are used to establish user access to the database. SQL commands directly affect the base tables, which contain the raw data, or they may affect database view, which has been created. The basic syntax of an SQL SELECT statement contains the following keywords: SELECT, FROM, WHERE, ORDER BY, GROUP BY and HAVING.

SELECT determines which attributes will be displayed in the query result table. FROM determines which tables or views will be used in the query. WHERE sets the criteria of the query, including any joins of multiple tables, which are necessary. ORDER BY determines the order in which the result will be displayed. GROUP BY is used to categorize results. HAVING is used to impose condition with GROUP BY.

1.10 SOLUTIONS / ANSWERS

Check Your Progress 1

1)

Advantages

- A standard for database query languages
- (Relatively) Easy to learn
- Portability
- SQL standard exists
- Both interactive and embedded access
- Can be used by specialist and non-specialist.

Yes, SQL has disadvantages. However, they are primarily more technical with reference to Language features and relational model theories. We are just putting them here for reference purposes.

Disadvantages

- Impedance mismatch – mixing programming paradigms with embedded access
- Lack of orthogonality – many different ways to express some queries
- Language is becoming enormous (SQL-92 is 6 times larger than predecessor)
- Handling of nulls in aggregate functions is not portable
- Result tables are not strictly relational – can contain duplicate tuples, imposes an ordering on both columns and rows.

2.

```
CREATE DOMAIN RoomType AS CHAR(1) ………….[Constraint (a) ]
          CHECK(VALUE IN (S, F, D));

CREATE DOMAIN HotelNumbers AS HotelNumber
          CHECK(VALUE IN (SELECT hotelNo FROM Hotel));
                                  [An additional constraint for
                                  hotel number for the application]

CREATE DOMAIN RoomPrice AS DECIMAL(5, 2)
          CHECK(VALUE BETWEEN 1000 AND 10000);

CREATE DOMAIN RoomNumber AS VARCHAR(4)
          CHECK(VALUE BETWEEN '1' AND '100');

                 [Constraint (c), one additional character is kept instead of 3
                 we have used 4characters but no space wastage as varchar ]
CREATE TABLE Room(
          roomNo        RoomNumber        NOT NULL,
          hotelNo       HotelNumbers      NOT NULL,
          type          RoomType          NOT NULL DEFAULT S,
          price         RoomPrice         NOT NULL,
          PRIMARY KEY (roomNo, hotelNo),
          FOREIGN KEY (hotelNo) REFERENCES Hotel
                 ON DELETE CASCADE ON UPDATE CASCADE);
          CREATE DOMAIN GuestNumber AS CHAR(4);
CREATE TABLE Guest(
        guestNo         GuestNumber          NOT NULL,
        guestName       VARCHAR(20)          NOT NULL,
        guestAddress    VARCHAR(50)          NOT NULL);

CREATE DOMAIN GuestNumbers AS GuestNumber
        CHECK(VALUE IN (SELECT guestNo FROM Guest));
                [A sort of referential constraint expressed within domain]

CREATE DOMAIN BookingDate AS DATETIME
        CHECK(VALUE > CURRENT_DATE);          [constraint (d) ]

CREATE TABLE Booking(
        hotelNo         HotelNumbers         NOT NULL,
        guestNo         GuestNumbers         NOT NULL,
        dateFrom        BookingDate          NOT NULL,
        dateTo          BookingDate          NULL,
        roomNo          RoomNumber           NOT NULL,
        PRIMARY KEY (hotelNo, guestNo, dateFrom),
        FOREIGN KEY (hotelNo) REFERENCES Hotel
                ON DELETE CASCADE ON UPDATE CASCADE,
        FOREIGN KEY (guestNo) REFERENCES Guest
                ON DELETE NO ACTION ON UPDATE CASCADE,
        FOREIGN KEY (hotelNo, roomNo) REFERENCES Room
                ON DELETE NO ACTION ON UPDATE CASCADE,
        CONSTRAINT RoomBooked
                CHECK (NOT EXISTS ( SELECT *
                                FROM Booking b
                                WHERE b.dateTo > Booking.dateFrom AND
                                b.dateFrom < Booking.dateTo AND
                                b.roomNo = Booking.roomNo AND
                                b.hotelNo = Booking.hotelNo)),
        CONSTRAINT GuestBooked
                CHECK (NOT EXISTS ( SELECT *
                                FROM Booking b
                                WHERE b.dateTo > Booking.dateFrom AND
                                b.dateFrom < Booking.dateTo AND
                                b.guestNo = Booking.guestNo)));
```

3.

FROM	Specifies the table or tables to be used.
WHERE	Filters the rows subject to some condition.
GROUP BY	Forms groups of rows with the same column value.
HAVING	Filters the groups subject to some condition.
SELECT	Specifies which columns are to appear in the output.
ORDER BY	Specifies the order of the output.

If the SELECT list includes an aggregate function and no GROUP BY clause is being used to group data together, then no item in the SELECT list can include any reference to a column unless that column is the argument to an aggregate function.

When GROUP BY is used, each item in the SELECT list must be single-valued per group. Further, the SELECT clause may only contain:

- Column names.
- Aggregate functions.
- Constants.
- An expression involving combinations of the above.

All column names in the SELECT list must appear in the GROUP BY clause unless the name is used only in an aggregate function.

3. Please note that some of the queries are sub-queries and queries requiring join. The meaning of these queries will be clearer as you proceed further with the Unit.

a)
```
SELECT SNO
FROM S
WHERE CITY = 'Delhi'
AND STATUS > 20;
```

Result:

SNO
S4

b)
```
SELECT SNO, STATUS
FROM S
WHERE CITY = 'Delhi'
ORDER BY STATUS DESC;
```

Result:

SNO	STATUS
S4	40
S5	10

c)
```
SELECT FIRST.SNO, SECOND.SNO
FROM S FIRST, S SECOND
WHERE FIRST.CITY = SECOND.CITY AND FIRST.SNO <
SECOND.SNO;
```

Please note that if you do not give the condition after AND you will get some unnecessary tuples such as: (S4, S4), (S5, S4) and (S5, S5).

Result:

SNO	SNO
S4	S5

d)
```
SELECT DISTINCT SNAME
FROM S, SP
WHERE S.SNO = SP.SNO
AND SP.PNO = 'P2';
```
Result:

SNAME
Prentice Hall

McGraw Hill
Wiley
Pearson

```
OR
SELECT SNAME
FROM S
WHERE SNO = ANY (   SELECT SNO
                    FROM SP
                    WHERE PNO = ‘P2’);
```

e)
```
SELECT SNAME
FROM S
WHERE SNO IN (       SELECT SNO
              FROM SP
              WHERE PNO = ‘P2’ );
```

f)
```
SELECT DISTINCT PNO
FROM SP SPX
WHERE PNO IN (       SELECT PNO
              FROM SP
       WHERE SP.SNO = SPX.SNO AND SPX.SNO < SP.SNO );
```

This query can also be answered using count and group by. Please formulate that.

Result:

PNO
P1
P2

g)
```
SELECT SNO
FROM S
WHERE CITY = (       SELECT CITY
                     FROM S
                     WHERE SNO = ‘S1’ );
```

Result:

SNO
S1

h)
```
SELECT SNAME
FROM S
WHERE EXISTS (       SELECT *
                     FROM SP
                     WHERE SNO = S.SNO AND PNO = ‘P1’);
```

Result:

SNAME
Prentice Hall
McGraw Hill

i)
```
SELECT PNO
FROM SP
WHERE QUANTITY > 200 UNION (   SELECT PNO
                               FROM SP
                               WHERE SNO = S2 );
```

Result:

PNO
P1
P2

j)
```
SELECT SNO
FROM S
WHERE STATUS > 25 OR STATUS IS NULL;
```

Result:

SNO
S1
S2
S4
S5

k)
```
SELECT COUNT (DISTINCT SNO)
FROM SP;
```

Result: 4

l)
```
SELECT PNO, SUM(QUANTITY)
FROM SP
GROUP BY PNO;
```

Result:

PNO	SUM
P1	400
P2	1000

m)
```
SELECT PNO
FROM SP
GROUP BY PNO
HAVING COUNT(*) > 1 ;
```

The query is a same as that of part (f)

n)
```
SELECT PNO, MAX(QUANTITY)
FROM SP
WHERE QUANTITY > 200
GROUP BY PNO
HAVING SUM(QUANTITY) > 300
ORDER BY 2, PNO DESC;
```

o)
```
UPDATE  S
SET STATUS = 2 * STATUS
WHERE CITY = ‘Delhi’;
```

p)
```
INSERT INTO TEMP
SELECT PNO
FROM SP
WHERE SNO = ‘S2’;
```

q)
```
INSERT INTO SP( SNO,PNO,QUANTITY) < ‘S5’,’P7’,100> ;
```

Please note that part cannot be added without a supply in the present case.

Actually there should be another table for Parts

r)
```
DELETE S, SP
WHERE SNO = (        SELECT SNO
                FROM S
                WHERE CITY = ‘Mumbai’);
```

Check Your Progress 2

1) Each user has an authorization identifier (allocated by DBA).
Each object has an owner. Initially, only owner has access to an object but the owner can pass privileges to carry out certain actions on to other users via the GRANT statement and take away given privileges using REVOKE.

2)
```
CREATE VIEW HotelData(hotelName, guestName) AS
SELECT h.hotelName, g.guestName
        FROM Hotel  h, Guest g, Booking b
        WHERE h.hotelNo = b.hotelNo AND g.guestNo = b.guestNo AND
                b.dateFrom <= CURRENT_DATE AND
                b.dateTo >= CURRENT_DATE;
```

3)
```
GRANT ALL PRIVILEGES ON HotelData
        TO Manager, Director WITH GRANT OPTION;
  GRANT ALL PRIVILEGES ON BookingOutToday
        TO Manager, Director WITH GRANT OPTION;
```

Check Your Progress 3

1) With a sub-query, the columns specified in the SELECT list are restricted to one table. Thus, cannot use a sub-query if the SELECT list contains columns from more than one table. But with a join operation SELECT list contains columns from more than two tables.

2) Answers of the queries are:

-
  ```
  SELECT guestName, guestAddress FROM Guest
  WHERE address LIKE ‘%Delhi%’
  ORDER BY guestName;
  ```

-
  ```
  SELECT price, type FROM Room
          WHERE hotelNo =
          (SELECT hotelNo FROM Hotel
          WHERE hotelName = ‘GRAND Hotel’);
  ```

-
  ```
  SELECT * FROM Room r
  WHERE roomNo NOT IN
  (SELECT roomNo FROM Booking b, Hotel h
  WHERE (dateFrom <= CURRENT_DATE AND
          dateTo >= CURRENT_DATE) AND
          b.hotelNo = h.hotelNo AND hotelName = ‘GRAND Hotel’);
  ```

-
  ```
  SELECT hotelNo, COUNT(roomNo) AS count
  FROM Room
  GROUP BY hotelNo;
  ```

-
  ```
  SELECT MAX(X)
  ```

```
FROM ( SELECT type, COUNT(type) AS X
FROM Booking b, Hotel h, Room r
WHERE r.roomNo = b.roomNo AND b.hotelNo = h.hotelNo AND
        city = 'LONDON'
GROUP BY type);
```

- UPDATE Room SET price = price*1.05;

3) (i) – Give names of employees who are working on all projects.

   (ii) - Give names of the projects which are currently not being worked upon.

1.11 FURTHER READINGS

Fundamentals of DatabaseSystems; Almasri and Navathe; Pearson Education Limited; Fourth Edition; 2004.
A Practical Approach to Design, Implementation, and Management; Thomas Connolly and Carolyn Begg; Database Systems, Pearson Education Limited; Third Edition; 2004.
The Complete Reference; Kevin Lonely and George Koch; Oracle 9i, Tata McGraw-Hill; Fourth Edition; 2003.
Jeffrey A. Hoffer, Marry B. Prescott and Fred R. McFadden; Modern Database Management; Pearson Education Limited; Sixth Edition; 2004.