DATABASE AND SQL

Basic Database concepts

Data: Raw facts and figures which are useful to an organization. We cannot take decisions on the basis of data.

Information: Well processed data is called information. We can take decisions on the basis of information.

Field: Set of characters that represents specific data element.

Record: Collection of fields is called a record. A record can have fields of different data types.

File: Collection of similar types of records is called a file.

Table: Collection of rows and columns that contains useful data/information is called a table. A table generally refers to the passive entity which is kept in secondary storage device.

Relation: Relation (collection of rows and columns) generally refers to an active entity on which we can perform various operations.

Database: Collection of logically related data along with its description is termed as database.

Tuple: A row in a relation is called a tuple.

Attribute: A column in a relation is called an attribute. It is also termed as field or data item.

Degree: Number of attributes in a relation is called degree of a relation.

Cardinality: Number of tuples in a relation is called cardinality of a relation.

Primary Key: Primary key is a key that can uniquely identifies the records/tuples in a relation. This key can never be duplicated and NULL.

Foreign Key: Foreign Key is a key that is defined as a primary key in some other relation. This key is used to enforce referential integrity in RDBMS.

Candidate Key: Set of all attributes which can serve as a primary key in a relation.

Alternate Key: All the candidate keys other than the primary keys of a relation are alternate keys for a relation.

DBA: Data Base Administrator is a person (manager) that is responsible for defining the data base schema, setting security features in database, ensuring proper functioning of the data bases etc.

Structured Query Language

SQL is a non procedural language that is used to create, manipulate and process the databases(relations).

Characteristics of SQL

1. It is very easy to learn and use.
2. Large volume of databases can be handled quite easily.
3. It is non procedural language. It means that we do not need to specify the procedures to accomplish a task but just to give a command to perform the activity.
4. SQL can be linked to most of other high level languages that makes it first choice for the database programmers.

Processing Capabilities of SQL

The following are the processing capabilities of SQL

1. Data Definition Language (DDL)
DDL contains commands that are used to create the tables, databases, indexes, views, sequences and synonyms etc.
   e.g: Create table, create view, create index, alter table etc.

2. Data Manipulation Language (DML)
DML contains command that can be used to manipulate the data base objects and to query the databases for information retrieval.
   e.g Select, Insert, Delete, Update etc.

3. View Definition:
DDL contains set of command to create a view of a relation.
   e.g : create view

4. Data Control Language:
This language is used for controlling the access to the data. Various commands like GRANT, REVOKE etc are available in DCL.

5. **Transaction Control Language (TCL)**
   TCL include commands to control the transactions in a data base system. The commonly used commands in TCL are COMMIT, ROLLBACK etc.

### Data types of SQL

Just like any other programming language, the facility of defining data of various types is available in SQL also. Following are the most common data types of SQL.

1. **NUMBER**
   Used to store a numeric value in a field/column. It may be decimal, integer or a real value. General syntax is
   ```
   Number(n,d)
   ```
   Where `n` specifies the number of digits and `d` specifies the number of digits to the right of the decimal point.
   
   e.g. `marks number(3)` declares `marks` to be of type number with maximum value 999.
   `pct number(5,2)` declares `pct` to be of type number of 5 digits with two digits to the right of decimal point.

2. **CHAR**
   Used to store character type data in a column. General syntax is
   ```
   Char (size)
   ```
   where `size` represents the maximum number of characters in a column. The CHAR type data can hold at most 255 characters.
   
   e.g. `name char(25)` declares a data item name of type character of upto 25 size long.

3. **VARCHAR/VARCHAR2**
   This data type is used to store variable length alphanumeric data. General syntax is
   ```
   varchar(size) / varchar2(size)
   ```
   where `size` represents the maximum number of characters in a column. The maximum allowed size in this data type is 2000 characters.
   
   e.g. `address varchar(50);` address is of type varchar of upto 50 characters long.

4. **DATE**
   Date data type is used to store dates in columns. SQL supports the various date formats other than the standard DD-MON-YY.
   
   e.g. `dob date;` declares `dob` to be of type date.

5. **LONG**
   This data type is used to store variable length strings of upto 2 GB size.
   
   e.g. `description long;`

6. **RAW/LONG RAW**
   To store binary data (images/pictures/animation/clips etc.) RAW or LONG RAW data type is used. A column LONG RAW type can hold upto 2 GB of binary data.
   
   e.g. `image raw(2000);`

### SQL Commands

a. **CREATE TABLE Command:**
   Create table command is used to create a table in SQL. It is a DDL type of command. The general syntax of creating a table is
   ```
   create table <table> (  
   <column 1> <data type> [not null] [unique] [<column constraint>], 
   . . . . . . .
   ```

### Creating Tables

The syntax for creating a table is

```
create table <table> (  
<column 1> <data type> [not null] [unique] [<column constraint>],
```
For each column, a name and a data type must be specified and the column name must be unique within the table definition. Column definitions are separated by comma. Uppercase and lowercase letters make no difference in column names, the only place where upper and lower case letters matter are strings comparisons. A not null Constraint means that the column cannot have null value, that is a value needs to be supplied for that column. The keyword unique specifies that no two tuples can have the same attribute value for this column.

Operators in SQL:
The following are the commonly used operators in SQL

1. Arithmetic Operators
   +, -, *, /
2. Relational Operators
   =, <, >, <=, =>, <>
3. Logical Operators
   OR, AND, NOT

Arithmetic operators are used to perform simple arithmetic operations.
Relational Operators are used when two values are to be compared and Logical operators are used to connect search conditions in the WHERE Clause in SQL.

Constraints:
Constraints are the conditions that can be enforced on the attributes of a relation. The constraints come in play when ever we try to insert, delete or update a record in a relation.

1. NOT NULL
2. UNIQUE
3. PRIMARY KEY
4. FOREIGN KEY
5. CHECK
6. DEFAULT

Not null ensures that we cannot leave a column as null. That is a value has to be supplied for that column.

e.g name varchar(25) not null;

Unique constraint means that the values under that column are always unique.

e.g Roll_no number(3) unique;

Primary key constraint means that a column can not have duplicate values and not even a null value.

e.g Roll_no number(3) primary key;

The main difference between unique and primary key constraint is that a column specified as unique may have null value but primary key constraint does not allow null values in the column.

Foreign key is used to enforce referential integrity and is declared as a primary key in some other table.

e.g cust_id varchar(5) references master(cust_id);

it declares cust_id column as a foreign key that refers to cust_id field of table master. That means we cannot insert that value in cust_id filed whose corresponding value is not present in cust_id field of master table.

Check constraint limits the values that can be inserted into a column of a table.

e.g marks number(3) check(marks>=0);

The above statement declares marks to be of type number and while inserting or updating the value in marks it is ensured that its value is always greater than or equal to zero.

Default constraint is used to specify a default value to a column of a table automatically. This default value will be used when user does not enter any value for that column.

e.g balance number(5) default = 0;

CREATE TABLE student ( 
Roll_no number(3) primary key, 
Name varchar(25) not null, 
Class varchar(10), 
Marks number(3) check(marks>0), 
City varchar(25) );

Data Modifications in SQL
After a table has been created using the create table command, tuples can be inserted into the table, or tuples can be deleted or modified.
**INSERT Statement**
The simplest way to insert a tuple into a table is to use the insert statement
insert into `<table>` [(<column i, . . . , column j>)] values (<value i, . . . , value j>);

```
INSERT INTO student VALUES(101,'Rohan','XI',400,'Jammu');
```
While inserting the record it should be checked that the values passed are of same data types as the one which is specified for that particular column.

For inserting a row interactively (from keyboard) & operator can be used.
e.g INSERT INTO student VALUES(&Roll_no,'&Name','&Class','&Marks','&City');
In the above command the values for all the columns are read from keyboard and inserted into the table student.

**NOTE:- In SQL we can repeat or re-execute the last command typed at SQL prompt by typing “/” key and pressing enter.**

<table>
<thead>
<tr>
<th>Roll_no</th>
<th>Name</th>
<th>Class</th>
<th>Marks</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Rohan</td>
<td>XI</td>
<td>400</td>
<td>Jammu</td>
</tr>
<tr>
<td>102</td>
<td>Aneeta Chopra</td>
<td>XII</td>
<td>390</td>
<td>Udhampur</td>
</tr>
<tr>
<td>103</td>
<td>Pawan Kumar</td>
<td>IX</td>
<td>298</td>
<td>Amritsar</td>
</tr>
<tr>
<td>104</td>
<td>Rohan</td>
<td>IX</td>
<td>376</td>
<td>Jammu</td>
</tr>
<tr>
<td>105</td>
<td>Sanjay</td>
<td>VII</td>
<td>240</td>
<td>Gurdaspur</td>
</tr>
<tr>
<td>113</td>
<td>Anju Mahajan</td>
<td>VIII</td>
<td>432</td>
<td>Pathankot</td>
</tr>
</tbody>
</table>

**Queries:**
To retrieve information from a database we can query the databases. SQL SELECT statement is used to select rows and columns from a database/relation.

**SELECT Command**
This command can perform selection as well as projection.
Selection: This capability of SQL can return you the tuples form a relation with all the attributes.
Projection: This is the capability of SQL to return only specific attributes in the relation.
* SELECT * FROM student; command will display all the tuples in the relation student
* SELECT * FROM student WHERE Roll_no <=102; The above command display only those records whose Roll_no less than or equal to 102.
Select command can also display specific attributes from a relation.
* SELECT name, class FROM student;
The above command displays only name and class attributes from student table.
* SELECT count(*) AS “Total Number of Records” FROM student;
Display the total number of records with title as “Total Number of Records” i.e an alias
We can also use arithmetic operators in select statement, like
* SELECT Roll_no, name, marks+20 FROM student;
* SELECT name, (marks/500)*100 FROM student WHERE Roll_no > 103;

**Eliminating Duplicate/Redundant data**
DISTINCT keyword is used to restrict the duplicate rows from the results of a SELECT statement.
e.g. SELECT DISTINCT name FROM student;
The above command returns
```
Name
Rohan
Aneeta Chopra
Pawan Kumar
```

**Conditions based on a range**
SQL provides a BETWEEN operator that defines a range of values that the column value must fall for the condition to become true.

```sql
SELECT Roll_no, name FROM student WHERE Roll_no BETWEEN 100 AND 103;
```

The above command displays Roll_no and name of those students whose Roll_no lies in the range 100 to 103 (both 100 and 103 are included in the range).

### Conditions based on a list

To specify a list of values, IN operator is used. This operator select values that match any value in the given list.

```sql
SELECT * FROM student WHERE city IN ('Jammu','Amritsar','Gurdaspur');
```

The above command displays all those records whose city is either Jammu or Amritsar or Gurdaspur.

### Conditions based on Pattern

SQL provides two wild card characters that are used while comparing the strings with LIKE operator.

a. `percent(%)` Matches any string
b. `Underscore(_)` Matches any one character

e.g. ```sql
SELECT Roll_no, name, city FROM student WHERE Roll_no LIKE "%3";
```
displays those records where last digit of Roll_no is 3 and may have any number of characters in front.

e.g. ```sql
SELECT Roll_no, name, city FROM student WHERE Roll_no LIKE "1_3";
```
displays those records whose Roll_no starts with 1 and second letter may be any letter but ends with digit 3.

### ORDER BY Clause

ORDER BY clause is used to display the result of a query in a specific order(sorted order). The sorting can be done in ascending or in descending order. It should be kept in mind that the actual data in the database is not sorted but only the results of the query are displayed in sorted order.

```sql
SELECT name, city FROM student ORDER BY name;
```

The above query returns name and city columns of table student sorted by name in increasing/ascending order.

```sql
SELECT * FROM student ORDER BY city DESC;
```
It displays all the records of table student ordered by city in descending order.

**Note:** If order is not specifies that by default the sorting will be performed in ascending order.

### GROUP BY Clause

The GROUP BY clause can be used in a SELECT statement to collect data across multiple records and group the results by one or more columns.

The syntax for the GROUP BY clause is:

```sql
SELECT column1, column2, ... column_n, aggregate_function (expression) FROM tables WHERE conditions
GROUP BY column1, column2, ... column_n;
```

*aggregate_function* can be a function such as `SUM`, `COUNT`, `MAX`, `MIN`, `AVG` etc.

```sql
SELECT name, COUNT(*) as "Number of employees"
FROM student
WHERE marks>350
GROUP BY city;
```

### HAVING Clause

The HAVING clause is used in combination with the GROUP BY clause. It can be used in a SELECT statement to filter the records that a GROUP BY returns.

The syntax for the HAVING clause is:

```sql
SELECT column1, column2, ... column_n, aggregate_function (expression) FROM tables WHERE predicates
GROUP BY column1, column2, ... column_n
HAVING condition1 ... condition_n;
```
e.g. SELECT SUM(marks) as "Total marks"
     FROM student
     GROUP BY department
     HAVING SUM(sales) > 1000;

Note: select statement can contain only those attribute which are already present in the group by clause.

Functions available in SQL

SQL provide large collection of inbuilt functions also called library functions that can be used directly in SQL statements.

1. Mathematical functions
2. String functions
3. Date & Time functions

1. Mathematical functions

Some of the commonly used mathematical functions are sum() avg(), count(), min(), max() etc.
e.g. SELECT sum(marks) FROM student;
displays the sum of all the marks in the table student.
e.g. SELECT min(Roll_no), max(marks) FROM student;
displays smallest Roll_no and highest marks in the table student.

2. String functions

These functions are used to deal with the string type values like ASCII, LOWEWR, UPPER, LEN, LEFT, RIGHT, TRIM, LTRIM, RTRIM etc.

ASCII: Returns the ASCII code value of a character(leftmost character of string).
Syntax: ASCII(character)
SELECT ASCII('a') returns 97
SELECT ASCII('A') returns 65
SELECT ASCII('1') returns 49
SELECT ASCII('ABC') returns 65
For Upper character 'A' to 'Z' ASCII value 65 to 90
For Lower character 'A' to 'Z' ASCII value 97 to 122
For digit '0' to '9' ASCII value 48 to 57

NOTE: If no table name is specified then SQL uses Dual table which is a dummy table used for performing operations.

LOWER: Convert character strings data into lowercase.
Syntax: LOWER(string)
SELECT LOWER('STRING FUNCTION') returns STRING FUNCTION

UPPER: Convert character strings data into Uppercase.
Syntax: UPPER(string)
SELECT UPPER('string function') returns STRING FUNCTION

LEN: Returns the length of the character string.
Syntax: LEN(string)
SELECT LEN('STRING FUNCTION') returns 15

REPLACE: Replaces all occurrences of the second string(string2) in the first string(string1) with a third string(string3).
Syntax: REPLACE('string1','string2','string3')
SELECT REPLACE('STRING FUNCTION','STRING','SQL') returns SQL Function
Returns NULL if any one of the arguments is NULL.

LEFT: Returns left part of a string with the specified number of characters counting from left.LEFT function is used to retrieve portions of the string.
Syntax: LEFT(string, integer)
SELECT LEFT('STRING FUNCTION', 6) returns STRING

RIGHT: Returns right part of a string with the specified number of characters counting from right.RIGHT function is used to retrieve portions of the string.
Syntax: RIGHT(string, integer)
SELECT RIGHT('STRING FUNCTION', 8) returns FUNCTION

LTRIM: Returns a string after removing leading blanks on Left side.(Remove left side space or blanks)
Syntax: LTRIM(string)
SELECT LTRIM('STRING FUNCTION') returns STRING FUNCTION

**RTRIM**: Returns a string after removing leading blanks on Right side.(Remove right side space or blanks)
Syntax: RTRIM( string )

SELECT RTRIM('STRING FUNCTION ') returns STRING FUNCTION

**REVERSE**: Returns reverse of a input string.
Syntax: REVERSE(string)

SELECT REVERSE('STRING FUNCTION') returns NOITCNUF GNIRTS

**REPLICATE**: Repeats a input string for a specified number of times.
Syntax: REPLICATE (string, integer)

SELECT REPPLICATE('FUNCTION', 3) returns FUNCTIONFUNCTIONFUNCTION

**SPACE**: Returns a string of repeated spaces. The SPACE function is an equivalent of using REPLICATE function to repeat spaces.
Syntax: SPACE ( integer) (If integer is negative, a null string is returned.)

SELECT ('STRING') + SPACE(1) + ('FUNCTION') returns STRING FUNCTION

**SUBSTRING**: Returns part of a given string.
SUBSTRING function retrieves a portion of the given string starting at the specified character(startindex) to the number of characters specified(length).
Syntax: SUBSTRING (string,startindex,length)

SELECT SUBSTRING('STRING FUNCTION', 1, 6) returns STRING

SELECT SUBSTRING('STRING FUNCTION', 8, 8) returns FUNCTION

**DELETE Command**
To delete the record from a table SQL provides a delete statement. General syntax is:-

DELETE FROM <table_name> [WHERE <condition>];
e.g. DELETE FROM student WHERE city = 'Jammu';
This command deletes all those records whose city is Jammu.

**NOTE**: It should be kept in mind that while comparing with the string type values lowercase and uppercase letters are treated as different. That is ‘Jammu’ and ‘jammu’ is different while comparing.

**UPDATE Command**
To update the data stored in the database, UPDATE command is used.

e.g. UPDATE student SET marks = marks + 100;
Increase marks of all the students by 100.

e.g. UPDATE student SET City = 'Udhampur' WHERE city = 'Jammu';
changes the city of those students to Udhampur whose city is Jammu.

We can also update multiple columns with update command, like

e.g. UPDATE student set marks = marks + 20, city = 'Jalandhar'
WHERE city NOT IN ('Jammu','Udhampur');

**CREATE VIEW Command**
In SQL we can create a view of the already existing table that contains specific attributes of the table.
e.g. the table student that we created contains following fields:
Student (Roll_no, Name, Marks, Class, City)
Suppose we need to create a view `v_student` that contains Roll_no,name and class of student table, then Create View command can be used:
CREATE VIEW v_student AS SELECT Roll_no, Name, Class FROM student;
The above command create a virtual table (view) named v_student that has three attributes as mentioned and all the rows under those attributes as in student table.

We can also create a view from an existing table based on some specific conditions, like
CREATE VIEW v_student AS SELECT Roll_no, Name, Class FROM student WHERE City <> 'Jammu';
The main difference between a Table and view is that

A Table is a repository of data. The table resides physically in the database.
A View is not a part of the database's physical representation. It is created on a table or another view. It is precompiled, so that data retrieval behaves faster, and also provides a secure accessibility mechanism.

**ALTER TABLE Command**

In SQL if we ever need to change the structure of the database then ALTER TABLE command is used. By using this command we can add a column in the existing table, delete a column from a table or modify columns in a table.

### Adding a column

The syntax to add a column is:-

```
ALTER TABLE table_name
ADD column_name datatype;
```

e.g. ALTER TABLE student ADD(Address varchar(30));

The above command add a column Address to the table student.

If we give command

```
SELECT * FROM student;
```

The following data gets displayed on screen:

<table>
<thead>
<tr>
<th>Roll_no</th>
<th>Name</th>
<th>Class</th>
<th>Marks</th>
<th>City</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Rohan</td>
<td>XI</td>
<td>400</td>
<td>Jammu</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>Aneta Chopra</td>
<td>XII</td>
<td>390</td>
<td>Udhampur</td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>Pawan Kumar</td>
<td>IX</td>
<td>298</td>
<td>Amritsar</td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>Rohan</td>
<td>IX</td>
<td>376</td>
<td>Jammu</td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>Sanjay</td>
<td>VII</td>
<td>240</td>
<td>Gurdaspur</td>
<td></td>
</tr>
<tr>
<td>113</td>
<td>Anju MAhajan</td>
<td>VIII</td>
<td>432</td>
<td>Pathankot</td>
<td></td>
</tr>
</tbody>
</table>

Note that we have just added a column and there will be no data under this attribute. UPDATE command can be used to supply values / data to this column.

### Removing a column

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

e.g. ALTER TABLE Student

```
DROP COLUMN Address;
```

The column Address will be removed from the table student.

**DROP TABLE Command**

Sometimes you may need to drop a table which is not in use. DROP TABLE command is used to Delete / drop a table permanently. It should be kept in mind that we can not drop a table if it contains records. That is first all the rows of the table have to be deleted and only then the table can be dropped. The general syntax of this command is:-

```
DROP TABLE <table_name>;
```

e.g. DROP TABLE student;

This command will remove the table student
Questions : Database and SQL : 1 mark questions

Q1 Write SQL queries to perform the following based on the table PRODUCT having fields as (prod_id, prod_name, quantity, unit_rate, price, city)

i. Display those records from table PRODUCT where prod_id is more than 100.
ii. List records from table PRODUCT where prod_name is ‘Almirah’
iii. List all those records whose price is between 200 and 500.
iv. Display the product names whose price is less than the average of price.
v. Show the total number of records in the table PRODUCT.

Q2. Define the terms:
   i. Database Abstraction
   ii. Data inconsistency
   iii. Conceptual level of database implementation/abstraction
   iv. Primary Key
   v. Candidate Key
   vi. Relational Algebra
   vii. Domain

6 Marks Questions SQL

Q1 Write SQL commands for (i) to (viii) on the basis of relations given below:

BOOKS
book_id  Book_name  author_name  Publishers  Price  Type  qty
k0001  Let us C  Sanjay mukharjee  EPB  450  Comp  15
p0001  Genuine  J. Mukhi  FIRST PUBL.  755  Fiction  24
m0001  Mastering c++  Kanetkar  EPB  165  Comp  60
n0002  Vc++ advance  P. Purohit  TDH  250  Comp  45
k0002  Near to heart  Sanjeev  FIRST PUBL.  350  Fiction  30

ISSUED

<table>
<thead>
<tr>
<th>Book_ID</th>
<th>Qty_Issued</th>
</tr>
</thead>
<tbody>
<tr>
<td>L02</td>
<td>13</td>
</tr>
<tr>
<td>L04</td>
<td>5</td>
</tr>
<tr>
<td>L05</td>
<td>21</td>
</tr>
</tbody>
</table>

i. To show the books of FIRST PUBL Publishers written by P. Purohit.
ii. To display cost of all the books written for FIRST PUBL.
iii. Depreciate the price of all books of EPB publishers by 5%.
iv. To display the BOOK_NAME, price of the books whose more than 3 copies have been issued.
v. To show total cost of books of each type.
vi. To show the detail of the most costly book.

Q2. Write SQL commands for (a) to (f) and write output for (g) on the basis of PRODUCTS relation given below:

PRODUCT TABLE

<table>
<thead>
<tr>
<th>PCODE</th>
<th>PNAME</th>
<th>COMPANY</th>
<th>PRICE</th>
<th>STOCK</th>
<th>MANUFACTURE</th>
<th>WARRANTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>P001</td>
<td>TV</td>
<td>BPL</td>
<td>10000</td>
<td>200</td>
<td>12-JAN-2008</td>
<td>3</td>
</tr>
</tbody>
</table>

78
a) To show details of all PCs with stock more than 110.
b) To list the company which gives warranty for more than 2 years.
c) To find stock value of the BPL company where stock value is sum of the products of price and stock.
d) To show number of products from each company.
e) To count the number of PRODUCTS which shall be out of warranty on 20-NOV-2010.
f) To show the PRODUCT name which are within warranty as on date.
g). Give the output of following statement.
(i) Select COUNT(distinct company) from PRODUCT.
(ii) Select MAX(price) from PRODUCT where WARRANTY<=3

Answers: 1 mark questions

Q1
Ans i: select * from product where prod_id > 100;
Ans ii: select * from product where prod_name = ‘Almirah’;
Ans iii: select * from product where price between 200 and 500;
Ans iv: select prod_name from product where price < avg(price);
Ans v: select count(*) from product;

Q2. Define the terms:
i. Database Abstraction
Ans: Database system provides the users only that much information that is required
by them, and hides certain details like, how the data is stored and maintained in
database at hardware level. This concept/process is Database abstraction.

ii. Data inconsistency
Ans: When two or more entries about the same data do not agree i.e. when one of them stores the updated information and the other does not, it results in data inconsistency in the database.

iii. Conceptual level of database implementation/abstraction
Ans: It describes what data are actually stored in the database. It also describes the
relationships existing among data. At this level the database is described logically in
terms of simple data-structures.

iv. Primary Key
Ans: It is a key/attribute or a set of attributes that can uniquely identify tuples
within the relation.

v. Candidate Key
Ans: All attributes combinations inside a relation that can serve as primary key are candidate key as they are candidates for being as a primary key or a part of it.

vi. Relational Algebra
Ans: It is the collections of rules and operations on relations(tables). The various operations are selection, projection, Cartesian product, union, set difference and intersection, and joining of relations.

vii. Domain
Ans: it is the pool or collection of data from which the actual values appearing in a
given column are drawn.

Answers: 6 Marks Questions:

Q1.
Ans i: select * from books where publishers='FIRST PUBL'
Ans ii: select sum(price*qty) from books where publishers='FIRST PUBL';
Ans iii: update books set price=price-0.5*price where publishers='EPB';
Ans iv: select BOOK_NAME, price from books, issued where 
books.book_id=issued.book_id and quantity_issued>3;
Ans v: select sum(price*qty) from books group by type;
Ans vi: select * from books where price=(select max(price) from books));

Q2.
Ans a: select * from products where pname='TV' and stock>110;
Ans b: select company from products where warranty>2;
Ans c: select sum(price*stock) from PRODUCTS where company='BPL';
Ans d: select company,COUNT(*) from products group by company;
Ans e: select count(*) from products where ('20-NOV-2010' -manufacture)/365>warranty;
Ans f: select pname from products where (sysdate- manufacture)/365<warranty;

Ans g (i): 4
Ans (ii): 39000

Some practice questions from Database and SQL:

2 marks questions
1. What is relation? What is the difference between a tuple and an attribute?
2. Define the following terminologies used in Relational Algebra:
   (i) selection     (ii) projection     (iii) union     (iv) Cartesian product
3. What are DDL and DML?
4. Differentiate between primary key and candidate key in a relation?
5. What do you understand by the terms Cardinality and Degree of a relation in relational database?
6. Differentiate between DDL and DML. Mention the 2 commands for each category.

6 marks questions

<table>
<thead>
<tr>
<th>Rt.no</th>
<th>Area_covered</th>
<th>Capacity</th>
<th>No.ofstudents</th>
<th>Distance</th>
<th>Transporter</th>
<th>Charges</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Vasant kunj</td>
<td>100</td>
<td>120</td>
<td>10</td>
<td>Shivam travels</td>
<td>100000</td>
</tr>
<tr>
<td>2</td>
<td>Hauz Khas</td>
<td>80</td>
<td>80</td>
<td>10</td>
<td>Anand travels</td>
<td>85000</td>
</tr>
<tr>
<td>3</td>
<td>Pitampura</td>
<td>60</td>
<td>55</td>
<td>30</td>
<td>Anand travels</td>
<td>60000</td>
</tr>
<tr>
<td>4</td>
<td>Rohini</td>
<td>100</td>
<td>90</td>
<td>35</td>
<td>Anand travels</td>
<td>100000</td>
</tr>
<tr>
<td>5</td>
<td>Yamuna Vihar</td>
<td>50</td>
<td>60</td>
<td>20</td>
<td>Bhalla Co.</td>
<td>55000</td>
</tr>
<tr>
<td>6</td>
<td>Krishna Nagar</td>
<td>70</td>
<td>80</td>
<td>30</td>
<td>Yadav Co.</td>
<td>80000</td>
</tr>
<tr>
<td>7</td>
<td>Vasundhara</td>
<td>100</td>
<td>110</td>
<td>20</td>
<td>Yadav Co.</td>
<td>100000</td>
</tr>
<tr>
<td>8</td>
<td>Paschim Vihar</td>
<td>40</td>
<td>40</td>
<td>20</td>
<td>Speed travels</td>
<td>55000</td>
</tr>
<tr>
<td>9</td>
<td>Saket</td>
<td>120</td>
<td>120</td>
<td>10</td>
<td>Speed travels</td>
<td>100000</td>
</tr>
<tr>
<td>10</td>
<td>Jank Puri</td>
<td>100</td>
<td>100</td>
<td>20</td>
<td>Kisan Tours</td>
<td>95000</td>
</tr>
</tbody>
</table>

(b) To show all information of students where capacity is more than the no of student in order of rt.no.
(c) To show area_covered for buses covering more than 20 km., but charges less then 80000.
(d) To show transporter wise total no. of students traveling.
(e) To show rt.no, area_covered and average cost per student for all routes where average cost per student is - charges/no.ofstudents.
(f) Add a new record with following data:
   (11, "Moti bagh",35,32,10,"kisan tours ", 35000)
(g) Give the output considering the original relation as given:
   (i) select sum(distance) from schoolbus where transporter="Yadav travels";
   (ii) select min(no.ofstudents) from schoolbus;
(iii) select avg(charges) from schoolbus where transporter= "Anand travels";
(v) select distinct transporter from schoolbus;

2.

**TABLE : GRADUATE**

<table>
<thead>
<tr>
<th>S.NO</th>
<th>NAME</th>
<th>STIPEND</th>
<th>SUBJECT</th>
<th>AVERAGE</th>
<th>DIV.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>KARAN</td>
<td>400</td>
<td>PHYSICS</td>
<td>68</td>
<td>I</td>
</tr>
<tr>
<td>2</td>
<td>DIWAKAR</td>
<td>450</td>
<td>COMP. Sc.</td>
<td>68</td>
<td>I</td>
</tr>
<tr>
<td>3</td>
<td>DIVYA</td>
<td>300</td>
<td>CHEMISTRY</td>
<td>62</td>
<td>I</td>
</tr>
<tr>
<td>4</td>
<td>REKHA</td>
<td>350</td>
<td>PHYSICS</td>
<td>63</td>
<td>I</td>
</tr>
<tr>
<td>5</td>
<td>ARJUN</td>
<td>500</td>
<td>MATHS</td>
<td>70</td>
<td>I</td>
</tr>
<tr>
<td>6</td>
<td>SABINA</td>
<td>400</td>
<td>CHEMISTRY</td>
<td>55</td>
<td>II</td>
</tr>
<tr>
<td>7</td>
<td>JOHN</td>
<td>250</td>
<td>PHYSICS</td>
<td>64</td>
<td>I</td>
</tr>
<tr>
<td>8</td>
<td>ROBERT</td>
<td>450</td>
<td>MATHS</td>
<td>68</td>
<td>I</td>
</tr>
<tr>
<td>9</td>
<td>RUBINA</td>
<td>500</td>
<td>COMP. Sc.</td>
<td>62</td>
<td>I</td>
</tr>
<tr>
<td>10</td>
<td>VIKAS</td>
<td>400</td>
<td>MATHS</td>
<td>57</td>
<td>II</td>
</tr>
</tbody>
</table>

(a) List the names of those students who have obtained DIV 1 sorted by NAME.
(b) Display a report, listing NAME, STIPEND, SUBJECT and amount of stipend received in a year assuming that the STIPEND is paid every month.
(c) To count the number of students who are either PHYSICS or COMPUTER SC graduates.
(d) To insert a new row in the GRADUATE table: 11,"KAJOL", 300, "computer sc", 75, 1
(e) Give the output of following sql statement based on table GRADUATE:

(i) Select MIN(AVERAGE) from GRADUATE where SUBJECT="PHYSICS";
(ii) Select SUM(STIPEND) from GRADUATE WHERE div=2;
(iii) Select AVG(STIPEND) from GRADUATE where AVERAGE>=65;
(iv) Select COUNT(distinct SUBJECT) from GRADUATE;
(f) Assume that there is one more table GUIDE in the database as shown below:

**Table: GUIDE**

<table>
<thead>
<tr>
<th>MAINAREA</th>
<th>ADVISOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHYSICS</td>
<td>VINOD</td>
</tr>
<tr>
<td>COMPUTER SC</td>
<td>ALOK</td>
</tr>
<tr>
<td>CHEMISTRY</td>
<td>RAJAN</td>
</tr>
<tr>
<td>MATHEMATICS</td>
<td>MAHESH</td>
</tr>
</tbody>
</table>

g) What will be the output of the following query:

SELECT NAME, ADVISOR FROM GRADUATE,GUIDE WHERE SUBJECT= MAINAREA;

3. Write SQL command for (i) to (vii) on the basis of the table SPORTS

**Table: SPORTS**

<table>
<thead>
<tr>
<th>Student NO</th>
<th>Class</th>
<th>Name</th>
<th>Game1</th>
<th>Grade</th>
<th>Game2</th>
<th>Grade2</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>7</td>
<td>Sammer</td>
<td>Cricket</td>
<td>B</td>
<td>Swimming</td>
<td>A</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
<td>Sujit</td>
<td>Tennis</td>
<td>A</td>
<td>Skating</td>
<td>C</td>
</tr>
<tr>
<td>12</td>
<td>7</td>
<td>Kamal</td>
<td>Swimming</td>
<td>B</td>
<td>Football</td>
<td>B</td>
</tr>
<tr>
<td>13</td>
<td>7</td>
<td>Venna</td>
<td>Tennis</td>
<td>C</td>
<td>Tennis</td>
<td>A</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>Archana</td>
<td>Basketball</td>
<td>A</td>
<td>Cricket</td>
<td>A</td>
</tr>
<tr>
<td>15</td>
<td>10</td>
<td>Arpit</td>
<td>Cricket</td>
<td>A</td>
<td>Athleitics</td>
<td>C</td>
</tr>
</tbody>
</table>

(a) Display the names of the students who have grade ‘C’ in either Game1 or Game2 or both.
(b) Display the number of students getting grade ‘A’ in Cricket.
(c) Display the names of the students who have same game for both Game1 and Game2.
(d) Display the games taken up by the students, whose name starts with ‘A’.
(e) Assign a value 200 for Marks for all those who are getting grade ‘B’ or grade ‘A’ in both Game1 and Game2.
(f) Arrange the whole table in the alphabetical order of Name.
(g) Add a new column named ‘Marks’.

4. Write SQL command for (i) to (vii) on the basis of the table Employees & EmpSalary

<table>
<thead>
<tr>
<th>Table: Employees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empid</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>010</td>
</tr>
<tr>
<td>105</td>
</tr>
<tr>
<td>152</td>
</tr>
<tr>
<td>215</td>
</tr>
<tr>
<td>244</td>
</tr>
<tr>
<td>300</td>
</tr>
<tr>
<td>335</td>
</tr>
<tr>
<td>400</td>
</tr>
<tr>
<td>441</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table: EmpSalary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Empid</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>010</td>
</tr>
<tr>
<td>105</td>
</tr>
<tr>
<td>152</td>
</tr>
<tr>
<td>215</td>
</tr>
<tr>
<td>244</td>
</tr>
<tr>
<td>300</td>
</tr>
<tr>
<td>335</td>
</tr>
<tr>
<td>400</td>
</tr>
<tr>
<td>441</td>
</tr>
</tbody>
</table>

Write the SQL commands for the following:

(i) To show firstname, lastname, address and city of all employees living in Paris

(ii) To display the content of Employees table in descending order of Firstname.

(iii) To display the firstname, lastname and total salary of all managers from the tables Employee and empsalary, where total salary is calculated as salary + benefits.

(iv) To display the maximum salary among managers and clerks from the table EmpSalary.

Give the Output of following SQL commands:

(i) Select firstname, salary from employees, empsalary where designation = 'Salesman' and Employees.empid=Empsalary.empid;

(ii) Select count(distinct designation) from empsalary;

(iii) Select designation, sum(salary) from empsalary group by designation having count(*) >2;

(iv) Select sum(benefits) from empsalary where designation = 'Clerk';