UNIT – II

Relational Model& SQL

Relational model:Basic concepts, Schema and instances, Keys, Relational Algebra, SQL: DDL, DML, Integrity constraints, Defining different constraints on a table, Set operations, Aggregate Functions, Group by and Having clauses, Nested queries.

1. Relational Model Basic Concepts:

- In relational model, data is represented as a collection of ***tables*** also known as a ***relations***.
- Databases created using relational model are known as ***relational databases***.
- Database management system that operate on relational database is known as Relational Database Management System(RDBMS)
- Popular Relational Database Management Systems (RDBMS) are Oracle, DB2, SQLServer, Informix, MySQL, MSAccess etc.
- A relational database consists of a collection of tables, each of which is assigned a unique name.
- The degree, also called arity, of a relation is the number of fields in it.
- The cardinality of a relation is the number of tuples in it.

- Example:the following figure shows tables with unique names

customer_id	customer_name	customer_street	customer_city
192-83-7465	Johnson	12 Alma St.	Palo Alto
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account_number	balance
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

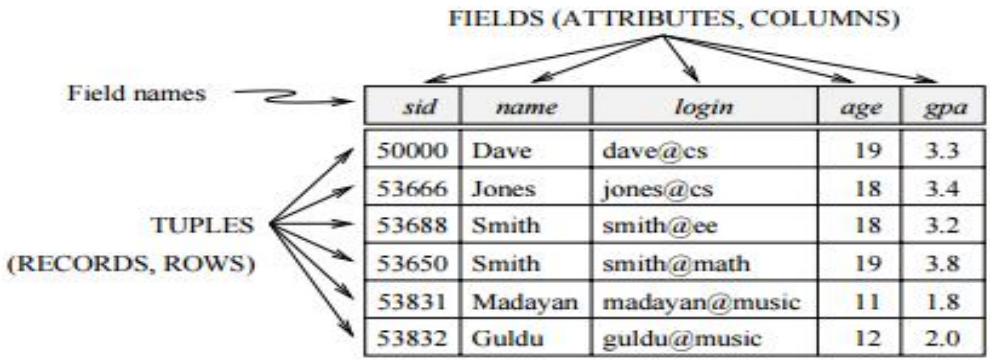
(b) The *account* table

customer_id	account_number
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table

Fig: tables with unique names

- Each table is described by a set of columns (also called as attributes, fields)
- For example, the following figure shows ***student*** table, which stores information about students.



sid	name	login	age	gpa
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Figure: Student table

- The ***student*** table has five columns: ***sid, name, login, age,*** and ***gpa.***
- Each row of this table records information about a student.
- **Relation schema:** A relation schema consists of a list of attributes and their corresponding domains.
- **Domain** of an attribute represents the set of permitted values of an attribute.
- For example, the domain of the name attribute of student relation is the set of all possible student names.
- Thus, the relation schema of the student relation can be written as
- student(sid:integer, name:string, login:string, age:integer, gpa:float)
- **Relation instance** – refers to a specific instance of a relation, i.e. containing a specific set of rows.
- For example, the instance of ***student relation*** has 6 rows corresponding to 6 students.
- **Relation Cardinality -** is the number of tuples or rows in the relation.
- For example, cardinality of student relation is 6 as it has 6 rows.
- **Relation Degree or Arity**- is the number of attributes (or columns) in the relation.
- For example, degree or arity of student relation is 5 as it has 5 columns.

NPTEL

- Relational Database Management System- A relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as introduced by

- E. F. Codd. A Relational database comprises of various relations or tables. We will start with the basic concepts of RDBMS which are as follows:

l) Table- The data in an RDBMS is stored in database objects which are called as relations or tables. This table is basically a collection of related data entries and it consists of numerous columns and rows. Below mentioned table shows a student table in a relationaldatabase.

S_ID	S_NAME	S_DOB	S_ADDRESS
001	Ajay	13-OCT-1991	Jaipur
002	Bhanu	02-JAN-1992	Delhi
003	Devika	04-JUL-1990	Kota

2) Field- Every table is broken up into smaller entities called fields or attributes. The fields in the STUDENT table consist of S_ID, S_NAME, S_DOB and S_ADDRESS. A field is a column in a table that is designed to maintain specific information about every record in the table.

3) Record- A record, also called as a row or tuple, is each individual entry that exists in a table. For example, there are 3 records in the above STUDENT table and one record specifies the complete information of an entity.

4) Column- A column is a vertical entity in a table that contains all information associated with a specific field in atable.

5) NULL value- A NULL value in a table is a value in a field that appears to be blank, which means a field with a NULL value is a field with no value. It is very important to understand that a NULL value is different than a zero value or a field that contains spaces. A field with a NULL value is the one that has been left blank during a recordcreation.

6) Database Schema- A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data.It is designed when the database doesn't exist at all. Once the database is operational, it is very difficult to make any changes to it. A database schema does not contain any data or information.

7) Database Instance- A database instance is a state of operational database with data at any given point of time. It contains a snapshot of the database. Database instances tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions that the database designers have imposed.

2. Schema and Instances

- Databases change over time.
- The information in a database at a particular point in time is called an instance of the database.

- The overall design of the database is called the database schema. In other words, a database schema consists of collection of relational schemas.
- The overall structure of the relation (or table) is known as relation schema. E.g. student relation schema is shown as
  - student (sid, name, login, age, gpa)
- The information in a relation (or table) at a particular point in time is called an instance of the relation. E.g. figure 1.7 shows instance of the *student relation.*

sid	*name*	*login*	*age*	*gpa*
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Fig. : Instance of *student relation*

3. Keys

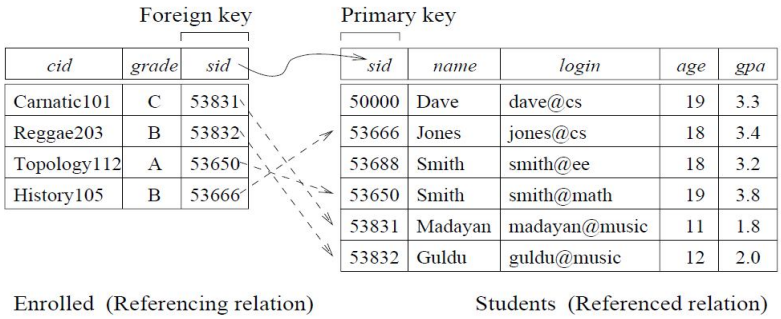
1. **Super Key** – set of one or more attributes that uniquely identifies a tuple in a relation is called as a super key.
2. **Candidate Key** – minimal set of attributes that uniquely identifies a tuple in a relation is called as a candidate key.
3. **Primary Key** – is a candidate key which uniquely identifies a tuple in a relation. The two properties of primary key are unique and not null.
4. **Foreign Key –**Ensure the referential integrity of the data in one table to match values in another table. Ensure that the foreign key in the child table must match with the primary key in the parent table.

Example:

```
CREATE TABLEStudent (sidnumber(3) UNIQUE,
                     namevarchar(60) NOT NULL,
                     loginvarchar(20),
                     Age number(3),
                     gpa number(2,2));

CREATE TABLEEnrolled (sidnumber(3),
                      cidvarchar(20),foreign key(sid) references student)
```

- The following figure shows the mapping mapping of Foreign key with Primary key.



CHILD TABLE PARENT TABLE

Fig: mapping of Foreign key with Primary key

NPTEL

Key-

- A key is a single or combination of multiple fields in a table. It is used to fetch or retrieve record(s) from a table according to the condition/requirement.
- Keys are also used to create relationship among different database tables or views.
- We have following types of keys in SQL which are used to fetch records from tables and to make relationship among tables or views.

1) **Super Key**-Super Key is the combination of attributes that can be used to identify a record uniquely in a table and is a set of one or more than one keys. Primary key, Unique key, Alternate key are subset of SuperKeys.
2) **Candidate Key**-A Candidate Key is a set of one or more fields/columns that can identify a record uniquely in a table. There can be multiple candidate keys in one table. Each candidate key can work as primarykey.
3) **Primary Key**- Primary Key is a set of one or more fields/columns of a table that uniquely identify a record in database table. It cannot accept null and duplicate values. Only one candidate key can be primarykey.
4) **Alternate Key**- An Alternate Key is a key that can be work as a primary key. Basically it isa candidate key that is not a primarykey.
5) **Composite/Compound Key-** Composite Key is a combination of more than one fields/columns of a table. It can be a candidate key or primarykey.
6) **Unique Key-** Unique Key is a set of one or more fields/columns of a table that uniquely identify a record in database table. It is like Primary key but it can accept only one null value and it cannot have duplicatevalues.
7) **Foreign Key-** Foreign Key is a field in a database table that is primary key in another table. It is used to link two tables in a databaseand can accept multiple null, duplicatevalues.

4. Relational Algebra (NPTEL)

- Relational algebra consists of simple and powerful operators to construct new relation(s) from given relations.
- It is part of first order logic and algebra of sets. If relations are stored data, then constructed relations can be taken as answers to queries on stored data.
- It deals with a set of finite relations which is closed under certain operators.
- Relational algebra was proposed by T Codd as an algebra on sets of tuples (i.e., relations) and is restricted the operands of relational algebra to finite relations only.
- Relational Algebra and relational calculus are two formal query languages for relational model and were developed prior to the development of commercial query language Structured Query Language (SQL).
- Relational algebra consists of operators and atomic operands.
  - The atomic operands in relational algebra are
  - Variables that stand relations and
  - constants, termed as finite relations.
  - All operands and results of expressions are sets.
  - The operators in relational algebra are broadly classified into four categories, namely,
  - Set operators,
  - Unary operators to remove parts of a relation,
  - Operators to combine the tuples of two relations and
  - Rename operator used to change relation schema without affecting the tuples of a relation.
- The basic set of operators of relational algebra enable the user to specify basic retrieval requests as relational algebra expressions.
- There are two types of operators, namely, Unary operators (selection and projection) and binary operators (cross product, union and set difference).
- Each operator considers one or two relation(s) as input and

produces new relation as output.

- Relational algebra expression is a sequence of relational algebra operators on relation(s) and the output produced by relational algebra expression is termed as result of database query.

- This language is accepted as formal foundation for relational model and is used as a basis for implementing and optimizing queries in the query processing and query optimization modules of DBMS.

- Relational algebra expression is also represented in the form of query tree where leaf nodes are relations and intermediate nodes are operators and finally the root node produces final answer for user query.

- Basic Operations:

  - selection (σ),
  - Projection (∏),
  - Union (∪) ,
  - Set difference (-) and
  - Cartesian Product (X)

Operation	Purpose	Notation
Selection	Selects all tuples that satisfy the selection condition from a relation R	σp(R)
Projection	Produces a new relation with only some of the attributes of R, and removes duplicate tuples.	∏A1, A2, An (R) where A1, A2 , An are attribute names of
Union	Produces all combinations of tuples from R1 and R2 that satisfy the join condition	R1 ∪ R2
Set Difference	Produces a relation that includes all tuples in R1 that are not in R2 and R2 must be union compatible	R1 - R2
Cartesian Product	ProducesarelationthathastheattributesofR1 and R2 and includes as tuples all possiblecombinations of tuples from R1 andR2	R1 X R2

4.1 Additional RelationalOperators:

- There are several additional operators and whichcanbederivedfrombasicoperators, such as
  - Set Intersection
  - Division
  - Natural join
  - semi join
  - antijoin

  - **SetIntersection(R1∩R2):**Itgivesthesetthatcontainsalltuplesof R1 thatalsobelongto R2(orequivalently,alltuplesofR2thatalsobelongtoR1),butnoothert uples.Settheoretic notationforR1∩R2={x:x□R1andx□R2}.

  - **Division (R1 ÷ R2):** Division between relations R1 and R2 results in the restrictions of tuplesinR1totheattributenamesuniquetoR1,i.e.,intheheaderofR1 butnotintheheader ofR2,forwhichitholdsthatalltheircombinationswithtuplesinR2are presentinR1.

  - **Natural Join (R1⋈R2):** Natural join (⋈) operator considers two relations R1 and R2 as relationsandproducesthesetofallcombinationsoftuplesinR1andR 2thatareequalon their common attribute names (or columnnames).

  - **SemiJoin(R1⋉R2):**Thisissimilartonaturaljoinandtheresultyielde dbythisisonlythe setofalltuplesinfirstrelationforwhichthereisatupleinsecondrelatio nthatisequalon their common attribute names.
  - **Antijoin:(R1▷R2):**Itisconverseofsemijoinoperation.Itreturnsalltu plesintheresult ofexpressionR1suchthattherearenotuplesintheresultofR2withma tchingvaluesforthe sharedattributes.

1. SQL:

- SQL stands for “Structured Query Language.” It is developed by IBM in 1974 and originally it is called as SEQUEL (Structured English QUEry Language).
- SQL commands consist of English-like statements which are used to query, insert, update, and delete data.

- SQL is a "nonprocedural" or "declarative" language. Here nonprocedural means that, when we want to retrieve data from the database it is enough to tell SQL what data to be retrieved, rather than how to retrieve it. The DBMS will take care of locating the information in the database.
- ANSI and ISO standard SQL:
    - SQL-86
    - SQL-89
    - SQL-92
    - SQL:1999 (language name became Y2K compliant!)
    - SQL:2003
- The first commercial DBMS that supported SQL was Oracle i.n 1979

SQL Commands: the following figure shows different SQL Commands

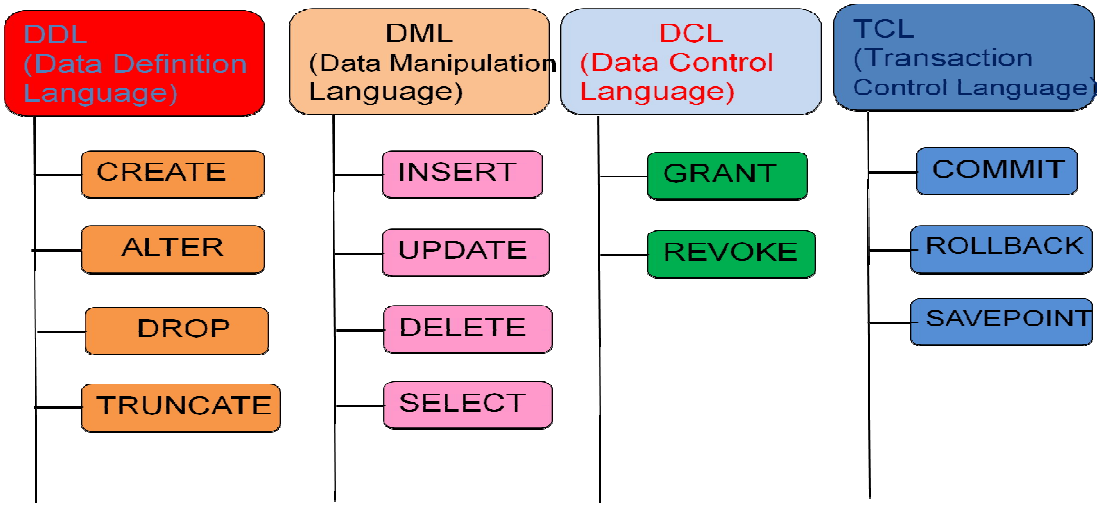


Figure: SQLCommands

1. **DDL commands -** are used to define a database, including creating, altering, and dropping tables and establishing constraints.

2. **DML commands** - are used to maintain and query a database, including updating, inserting, modifying, and querying data.

3. **DCL commands** - are used to control a database including administering privileges and saving of data. DCL commands are used to determine whether a user is allowed to carry out a particular operation or not.

4. **TCL commands -** are used to manage transactions in database. These are used to manage the changes made by DML statements. It also allows statements to be grouped together into logical transactions.

SQL Commands

- **Create Table Command:**

Is used to define a database, including creating, altering, and dropping tables and establishing constraints.

Syntax:

CREATE TABLE *table name*
(column-name1 data-type-1 [constraint],
column-name2 data-type-2 [constraint],
column-nameN data-type-N [constraint]);

Example

create table *branch*
(*branch_name* **varhar**(15)**,**
branch_city **varhar(**30),
assets **integer**);

- **Domain Types in SQL: (NPTEL)**

1. **char(n).** Fixed length character string, with user-specified length *n.*
2. **varchar(n).** Variable length character strings, with user-specified maximum length *n.*
3. **int.** Integer (a finite subset of the integers that is machine-dependent).
4. **smallint.** Small integer (a machine-dependent subset of the integer domain type).
5. **number(p,d).** Fixed point number, with user-specified precision of *p* digits, with *n* digits to the right of decimal point.
6. **real, double precision.** Floating point and double-precision floating point numbers, with machine-dependent precision.
7. **float(n).** Floating point number, with user-specified precision of at least *n* digits.

SQL Data Types-Data storage format of an object is defined by SQL data types. These objects can be a variable or columns or expressions and data

can be numeric, character, string, binary, date and time etc.

1) Numeric Data Types inSQL:

- INTEGER – Stores integer number, values from -2,147,483,648 to2,147,483,647.
- SMALLINT – Stores small integer number, values from -32,768 to32,767.
- TINYINT – Stores tiny integer number, values from 0 to255.
- NUMERIC(P,S) – Stores values from -10^38 +1 to 10^38 -1. Where 'p' isprecision value and 's' is scalevalue.
- REAL – Single precision floating point number, stores values from -3.40E + 38 to3.40E
- + 38.
- DECIMAL(P,S) – Stores values from -10^38 +1 to 10^38 -1, where 'p' is precisionvalue and 's' is scalevalue.
- FLOAT(P) – Stores values from -1.79E + 308 to 1.79E + 308, where 'p' isprecision value.
- DOUBLE PRECISION – Double precision floating pointnumber.
- BIT(X) – Where 'x' is the number of bits to store, stores value from 0 to1.
- BIT VARYING(X) – 'X' is the number of bits to store (length can vary up tox).

2) Character DataTypes:

- CHAR(X) – Where 'x' is the character's number to store. It can store upto 8,000 characters.
- VARCHAR2(X) – Where 'x' is the character's number to store. It can store upto8,000 characters.
- VARCHAR(MAX) – It can store upto 231characters.
- text – Can store upto 2,147,483,647characters.

3) Binary Data Types inSQL:

- binary – Can store upto 8,000 bytes (Fixed-length binarydata).
- varbinary – Can store upto 8,000 bytes. (Variable length binarydata).
- varbinary(max) – Can store upto 231 bytes (Variable length Binarydata).
- image – Can store upto 2,147,483,647 bytes. (Variable length BinaryData).

4) Date and Time Data Types inSQL:

- DATE – Stores month, days and yearvalues.
- TIME – Stores hour, minute and secondvalues.
- TIMESTAMP – It stores year, month, day, hour, minute and second values.

SQL Operators- The symbols which are used to perform logical and

mathematical operations in SQL are called SQL operators. Three types of operators used in SQL are as follows:

1) ArithmeticOperators:

Arithmetic operators are used to perform mathematical calculations like addition, subtraction, multiplication, division and modulus in SQL statements.

Arithmetic Operators	Example/Description
+ (Addition)	A+B
– (Subtraction)	A-B
* (multiplication)	A*B
/ (Division)	A/B
% (Modulus)	A%B

2) RelationalOperators:

Relational operators in SQL are used to find the relation between two columns. i.e. to compare the values of two columns in SQL statements.

Relational Operators	Example/Description
>	x > y (x is greater than y)
<	x < y (x is less than y)
>=	x >= y (x is greater than or equal to y)
<=	x <= y (x is less than or equal to y)
=	x <= y (x is less than or equal to y)
!= or <>	x != y or x <> y (x is not equal to y)
!<	x !< y (x is not less than y)
!>	x !> y (x is not greater than y)

3. LogicalOperators:

Logical operators in SQL are used to perform logical operations on the given expressions in SQL statements. There are many operators in SQL which are used in SQL statements in the WHERE clause. They are,

- AND
- OR
- NOT
- BETWEEN…AND
- IS NULL, IS NOTNULL
- LIKE
- UNIQUE

➢ **Basic Insertion and Deletion of Tuples:**

- Newly created table is empty
- Add a new tuple to *account*

insert into *account*

values('A-9732', 'Perryridge', 1200);

Note: Insertion fails if any integrity constraint is violated

- Delete *all* tuples from *account*

  delete from *account*

  Note: Will see later how to delete selected tuples

➢ **Drop and Alter Table Constructs:**

- The **drop table** command deletes all information about the dropped relation from the database.
- The **alter table** command is used to add attributes to an existing relation:

  alter table *r* **add** *A D*

  - where*A* is the name of the attribute to be added to relation *r* and *D* is the domain of *A.*
  - All tuples in the relation are assigned *null* as the value for the new attribute.
- The **alter table** command can also be used to drop attributes of a relation:

  alter table *r* **drop** *A*

  - where *A* is the name of an attribute of relation *r*
  - Dropping of attributes not supported by many databases

Tables: consider the following figure for writing queries

branch_name	*branch_city*	*assets*
Brighton	Brooklyn	7100000
Downtown	Brooklyn	9000000
Mianus	Horseneck	400000
North Town	Rye	3700000
Perryridge	Horseneck	1700000
Pownal	Bennington	300000
Redwood	Palo Alto	2100000
Round Hill	Horseneck	8000000

branch relation

loan_number	*branch_name*	*amount*
L-11	Round Hill	900
L-14	Downtown	1500
L-15	Perryridge	1500
L-16	Perryridge	1300
L-17	Downtown	1000
L-23	Redwood	2000
L-93	Mianus	500

loan relation

customer_name	loan_number
Adams	L-16
Curry	L-93
Hayes	L-15
Jackson	L-14
Jones	L-17
Smith	L-11
Smith	L-23
Williams	L-17

customer relation

Basic Query Structure

A typical SQL query has the form:

select A_1, A_2, ...,A_n
from r_1, r_2, ..., r_m
where P

- A_i represents an attribute
- R_i represents a relation
- P is a predicate.

The result of an SQL query is a relation.

The select Clause:

- The **select** clause list the attributes desired in the result of a query
- Example: find the names of all branches in the *loan* relation:

select *branch_name* **from** *loan;*

branch_name
Round Hill
Downtown
Perryridge
Perryridge
Downtown
Redwood
Mianus

- NOTE: SQL names are case insensitive (i.e., you may use upper- or lower-case letters.)
  Example: *Branch_Name* ≡ *BRANCH_NAME* ≡ *branch_name*
  Some people use upper case wherever we use bold font.

- SQL allows duplicates in relations as well as in query results.
- To force the elimination of duplicates, insert the keyword **distinct** after select**.**
- Find the names of all branches in the *loan* relation, and remove duplicates

select distinct *branch_name***from** *loan;*

Branch_name
Round Hill
Downtown
Perryridge
Redwood
Mianus

- The keyword **all** specifies that duplicates not be removed.

select all*branch_name***from** *loan;*

- An asterisk in the select clause denotes "all attributes"

select*****from** *loan*

- The **select** clause can contain arithmetic expressions involving the operation, +, –, □, and /, operating on constants or attributes of tuples.

Example: **select***loan_number, branch_name,*
amount *100
from*loan*

The where Clause:

- The **where** clause specifies conditions that the result must satisfy.
- To find all loan number for loans made at the Perryridge branch with loan amounts greater than $1200.

select*loan_number*
from *loan*
where *branch_name* ='Perryridge'**and** *amount* > 1200

- Comparison results can be combined using the logical connectives **and, or,** and **not.**

The from Clause:

- The **from** clause lists the relations involved in the query
- Find the Cartesian product *borrower X loan*

select *from *borrower, loan;*

- Find the name, loan number and loan amount of all customers having a loan at the Perryridge branch.

select*customer_name, borrower.loan_number, amount*
from *borrower, loan*

where *borrower.loan_number = loan.loan_number* **and** *branch_name* = 'Perryridge'

The Rename Operation:

- SQL allows renaming relations and attributes using the **as** clause: *old-name* **as** *new-name*
- Example: Find the name, loan number and loan amount of all customers; rename the column name *loan_number* as *loan_id.*

select *customer_name, borrower.loan_number* **as** *loan_id, amount*
from *borrower, loan*
where *borrower.loan_number = loan.loan_number*

Tuple Variables:

- Tuple variables are defined in the **from** clause.
- Find the customer names and their loan numbers and amount for all customers having a loan at some branch

select *customer_name, T.loan_number, S.amount*
from *borrower T, loan S*
where *T.loan_number = S.loan_number*

- Find the names of all branches that have greater assets than some branch located in Brooklyn.

select distinct *T.branch_name*
from *branch T, branch S*
where *T.assets>S.assets* **and** *S.branch_city* = 'Brooklyn'

String Operations:

- SQL includes a string-matching operator for comparisons on character strings. The operator "like" uses patterns that are described using two special characters:
  - **Percent sign %** : represents zero, one or more than one character.
  - **Underscore sign** _ : represents only one character.
- Find the names of all customers whose street includes the substring "Main".

select *customer_name*

from *customer*

where *customer_street* **like** '% Main%'

SQL supports a variety of string operations such as

- concatenation (using "||")
- converting from upper to lower case (and vice versa)
- finding string length, extracting substrings, etc.

Ordering the Display of Tuples:

- List in alphabetical order the names of all customers having a loan in Perryridge branch

select distinct *customer_name*

from *borrower, loan*

where *borrower loan_number = loan.loan_number* **and** *branch_name* = 'Perryridge'

order by *customer_name*

- We may specify **desc** for descending order or **asc** for ascending order, for each attribute; ascending order is the default.

Example: **order by** *customer_name* **desc**

8. Integrity Constraints :

5. **CHECK -** Ensures that the value in a column meets a specific condition. E.g. check(account_balance>0).

   Example:

   CREATE TABLE *Student* (sidnumber(3) NOT NULL,
   Name varchar(60) NOT NULL,
   Age number(3),
   CHECK(sid> 0));

6. **NOT NULL** - Indicates that a column cannot store NULL value. E.g. Account_numberchar(10) not null.

Example:

CREATE TABLE*Student* (sidnumber(3) **NOT NULL**,
Name varchar(60) **NOT NULL**,
Age number(3));

7. **UNIQUE** - The UNIQUE constraint imposes that every value in a column or set of columns be unique. It means that no two rows of a table can have duplicate values in a specified column or set of columns. E.g. UNIQUE(Name, DOB).

Example:

CREATE TABLE*Student* (sidnumber(3) **UNIQUE**,
Name varchar(60) NOT NULL,
Age number(3));

8. **FOREIGN KEY** – Ensure the referential integrity of the data in one table to match values in another table. Ensure that the foreign key in the child table must match with the primary key in the parent table.

Example:

CREATE TABLE*Student* (sidnumber(3) **UNIQUE**,
namevarchar(60) NOT NULL,
loginvarchar(20),
Age number(3),
gpa number(2,2));

CREATE TABLE*Enrolled* (sidnumber(3),
cidvarchar(20),
foreign key(sid) references student)

- The following figure shows the mapping mapping of Foreign key with Primary key.

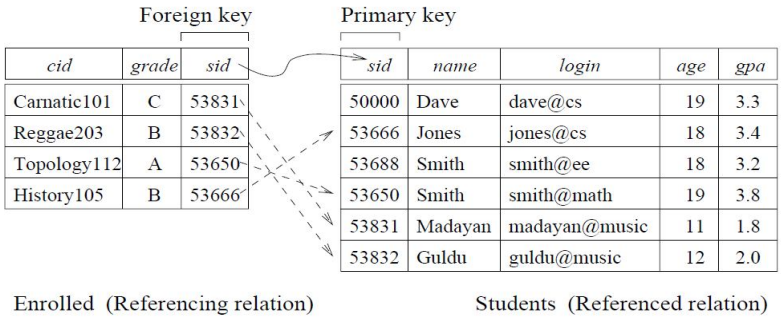


Fig: mapping of Foreign key with Primary key

- If we try to insert the *tuple (55555, Art104, A)* into **Enrolled** table, then the foreign key constraint is violated because there is no tuple in Students table with sid 55555 and so this insertion is rejected.

ON DELETE CASCADE

- When the clause ON DELETE CASCADE is included in the child table, and if a row is deleted from the parent table then the corresponding referenced value in the child table will also be deleted.

ON DELETE SET NULL

- If ON DELETE SET NULL clause is included in the child table means, whenever a row in the parent table is deleted, then the corresponding referenced value in the child table will be set null.

- **Difference between UNIQUE and NOTNULL Constraint**

NOTNULL Constraint	UNIQUE Constraint
An attribute declared as NOTNULL will not accept NULL values	An attribute declared as UNIQUE can accept NULL values
An attribute declared as NOTNULL will accept duplicate values	An attribute declared as UNIQUE will not accept duplicate values

- **Difference between UNIQUE and PRIMARY KEY Constraint**

PRIMARY KEY constraint	UNIQUE constraint
an attribute declared as primary key will not accept NULL values	an attribute declared as UNIQUE will accept NULL values
only one PRIMARY KEY can be defined for each table	more than one UNIQUE constraint can be defined for each table

GROUP BY: (NPTEL)

- The GROUP BY clause is used in a SELECT statement to collect data across multiple records and group the results by one or more columns.
- Sometimes it is required to get information not about each row, but about each group.
- Example: consider the Customer_Loan table that has data about all the loans taken by all the customers of the bank. Assume that we want to retrieve the total loan-amount of all loans taken by each customer.
- Related rows can be grouped together by the GROUP BY clause by specifying a column as a grouping column.
- In the above example, the Cust_ID will be the grouping column.
- In the output table all the rows with an identical value in the grouping column will be grouped together. Hence, the number of rows in the utput is equal to the number of distinct values of the grouping column.

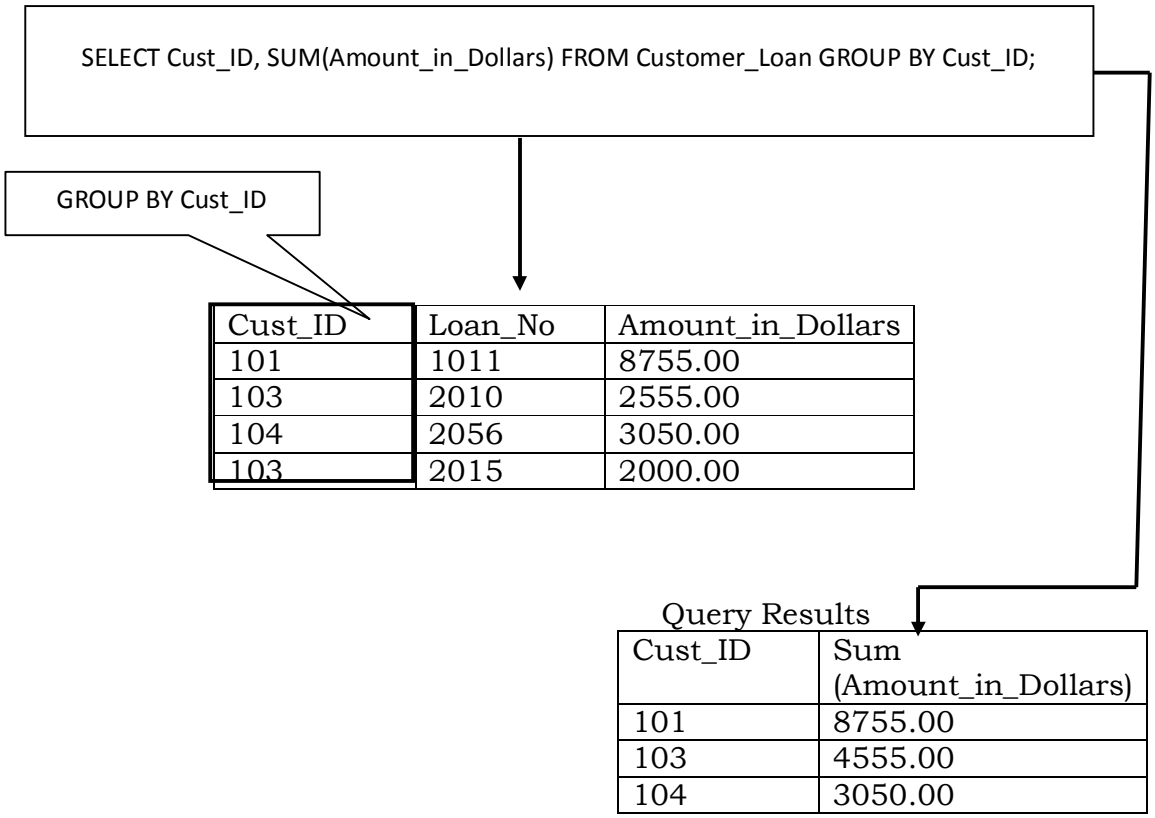
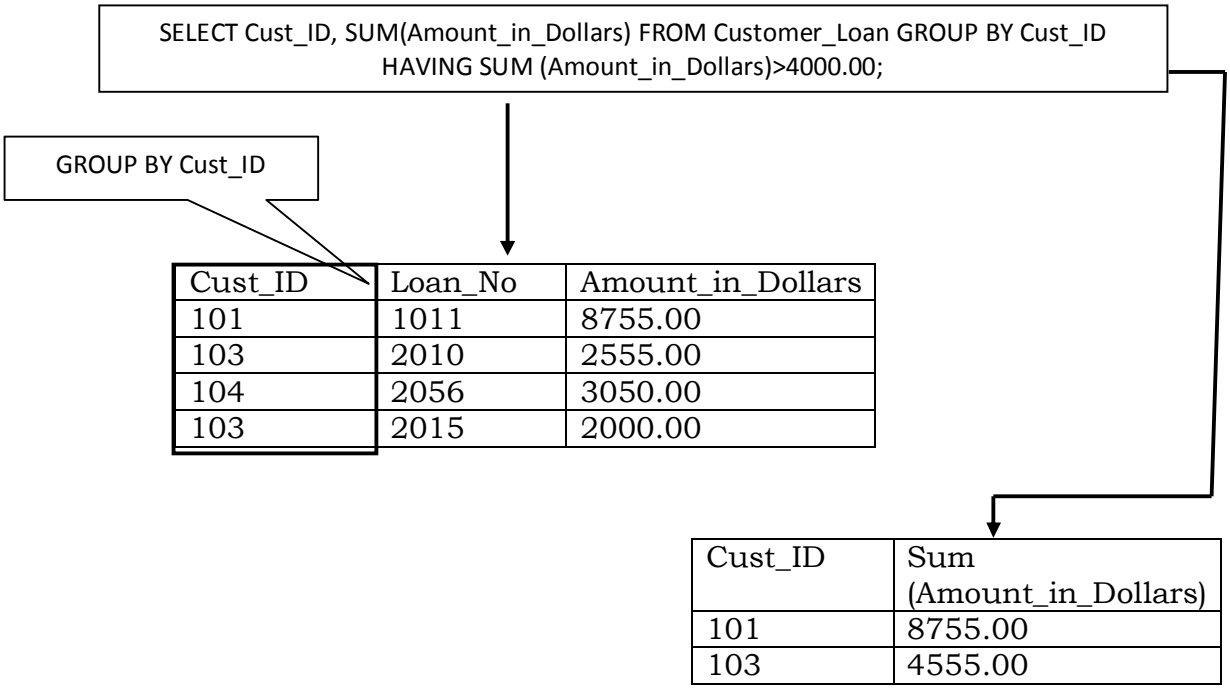


Figure: Example of Group BY Clause

HAVING: (NPTEL)

- The HAVING clause is used along with the GROUP BY clause. The HAVING clause can be used to select and reject row groups.
- The format of the HAVING clause is similar to the WHERE clause, consisting of the keyword HAVING followed by a search condition.
- The HAVING clause thus specifies a search condition for groups.



Cust_ID	Loan_No	Amount_in_Dollars
101	1011	8755.00
103	2010	2555.00
104	2056	3050.00
103	2015	2000.00

Cust_ID	Sum (Amount_in_Dollars)
101	8755.00
103	4555.00

Figure: Example of HAVING Clause

2. Set Operations in SQL:

SQL provides 3 types of set operations.

1. **UNION** – Let R and S are two union compatible relations. Then the UNION operation returns the tuples that occur either in R or S or both.
   - Two relation instances are said to be union-compatible if the following conditions hold:
   - they have the same number of the columns, and
   - Corresponding columns, taken in order from left to right, have the same *data types.*
2. **INTERSECT** – Let R and S are two union compatible relations. Then the INTERSECT operation returns the tuples that are common to the two relations.
3. **MINUS** – If R and S are two union compatible relations then **R MINUS** S returns the tuples that are present in R but not in S.

Examples using set operations in SQL:

Consider the following two relations.

Hard_Disk	Speed	OS
100GB	1.8 GHz	Linux
200GB	2.0GHz	Windows
250GB	2.8GHz	Windows

IBM_Desktop

Hard_Disk	Speed	OS
100GB	1.8 GHz	Linux
200GB	2.0GHz	Windows
250GB	2.0GHz	Windows

Dell_Desktop

```
SELECT * FROM IBM_Desktop
UNION
SELECT * FROM Dell_Desktop;
```

Hard_Disk	Speed	OS
100GB	1.8 GHz	Linux
200GB	2.0GHz	Windows
250GB	2.8GHz	Windows
250GB	2.0GHz	Windows

```
SELECT * FROM IBM_Desktop
UNION ALL
SELECT * FROM Dell_Desktop;
```

Hard_Disk	Speed	OS
100GB	1.8 GHz	Linux
100GB	1.8 GHz	Linux
200GB	2.0GHz	Windows
200GB	2.0GHz	Windows
250GB	2.8GHz	Windows
250GB	2.0GHz	Windows

```
SELECT * FROM IBM_Desktop
      INTERSECT
SELECT * FROM Dell_Desktop;
```

Hard_Disk	Speed	OS
100GB	1.8 GHz	Linux
200GB	2.0GHz	Windows

```
SELECT * FROM IBM_Desktop
MINUS
SELECT * FROM Dell_Desktop;
```

Hard_Disk	Speed	OS
250GB	2.8GHz	Windows

Consider sailors, boats, reserves tables for set operations:

sid	*sname*	*rating*	*age*
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Sailors

sid	*bid*	*day*
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Reserves

bid	*bname*	*color*
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Boats

- Find the names of sailors who have reserved a red or a green boat.

```
SELECT S.sname
  FROM Sailors S, Reserves R, Boats B
  WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
  UNION
  SELECT S.sname
  FROM Sailors S, Reserves R, Boats B
  WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'green';
```

sname
Dustin
Lubber
Horatio

- Find the sids of sailors who have reserved both red and green boats.

```
SELECT S.sid
```

```
FROM Reserves R, Boats B
WHERE R.bid = B.bid AND B.color = 'red'
INTERSECT
SELECT S.sid
FROM Reserves R, Boats B
WHERE R.bid = B.bid AND B.color = 'green';
```

sid
22
31

- Find the sids of sailors who have reserved red boats but not green boats.

```
SELECT S.sid
  FROM Reserves R, Boats B
  WHERE R.bid = B.bid AND B.color = 'red'
  MINUS
  SELECT S.sid
  FROM Reserves R, Boats B
  WHERE R.bid = B.bid AND B.color = 'green';
```

sid
64

- *Find the sids of sailors who have reserved red boats but not green boats.*

```
SELECT S.sid
  FROM Reserves R, Boats B
  WHERE R.bid = B.bid AND B.color = 'red'
  MINUS
  SELECT S.sid
  FROM Reserves R, Boats B
  WHERE R.bid = B.bid AND B.color = 'green';
```

sid
64

3. Aggregate Functions in SQL:

1. MIN() - Returns the smallest value in a given column
2. MAX() - Returns the largest value in a given column
3. SUM() - Returns the sum of numeric values in a given column
4. AVG() - Returns the average value of a given column

5. COUNT() - Returns the total number of values (excluding NULL values) in a given column
6. COUNT(*) - Returns the number of rows in a table (including NULL values).

EXAMPLES

- *Find the average age of all sailors.*

```
SELECT AVG(S.age) AS avgAge
  FROM Sailors S;
```

avgAge
37.4

- *Find the average age of sailors with a rating of 10.*

```
SELECT AVG(S.age) AS avgAgeOfRating10
  FROM Sailors S
  WHERE S.rating = 10;
```

avgAgeOfRating10
25.5

- *Find the age of the youngest sailor.*

```
SELECT MIN(S.age) AS youngestSailorAge
 FROM Sailors S;
```

youngestSailorAge
16.0

- *Find the age of the oldest sailor.*

```
SELECT MAX(S.age) AS oldestSailorAge
 FROM Sailors S;
```

oldestSailorAge
63.5

- *Find the total number of sailors.*

```
SELECT COUNT(S.sid) AS NoOfSailors
 FROM Sailors S;
```

NoOfSailors
10

- *Find the number of sailors with rating 10.*

```
SELECT COUNT(S.sid) AS NoOfSailorsWithRating10
 FROM Sailors S
 WHERE S.rating = 10;
```

NoOfSailorsWithRating10
10

- *Count the distinct ratings.*

```
SELECT COUNT(DISTINCT S.sid) AS NoOfRatings
 FROM Sailors S;
```

NoOfRatings
6

5. Nested Queries:

- A **nested query is a query** that has another query embedded within it. The embedded query is called a **sub query.**

- *Find the names of sailors who have reserved boat 103.*

```
SELECT S.sname
FROM Sailors S
WHERE S.sid IN ( SELECTR.sid
                 FROM Reserves R
                 WHERE R.bid = 103 )
```

sname
Dustin
Lubber
Horatio

- *Find the names of sailors who have not reserved boat 103.*

```
SELECT S.sname
FROM Sailors S
WHERE S.sid NOT IN ( SELECTR.sid
                     FROM Reserves R
                     WHERE R.bid = 103 )
```

sname
Horatio

- *Find the names of sailors who have reserved a red boat.*

```
SELECT S.sname
FROM Sailors S
WHERE S.sid NOT IN ( SELECTR.sid
                     FROM Reserves R
                     WHERE R.bid IN (SELECT R.bid
                          FROM Boats B
                          WHERE B.color='red');
```

sname
Dustin
Lubber
Horatio

UNIT-II

Assignment-Cum-Tutorial Questions

SECTION-A

Objective Questions

1. For every teacher record in a database, there is an attribute called Department. This attribute specifies the department name. At times, the name may contain the numeric department id concatenated with it. However, it can never comprise only of the department id. Department name is optional in a teacher record.

Identify the correct components for the domain of the attribute Department. [ ]

a) Date b) Integer c) NULL d) Alphanumeric (String and Integer)

2. Identify the correct statement(s). [ ]

a) A Candidate Key is a set of one or more attributes that, taken collectively, allows us to uniquely identify any entity in the entity set

b) A Candidate Key for which no proper subset is also a Candidate Key is called a Super Key

c) A Super Key is a set of one or more attributes that, taken collectively, allows us to uniquely identify any entity in the entity set

d) A Super Key for which no proper subset is also a Super Key is called a Candidate Key

i)a,b ii) a,c,d iii) a iv) c,d

3. Identify the valid data-types, which can be used in SQL to define the type of data. [ ]

a) Varchar b) string c) real d) float

i) a,b ii) c,d iii) a iv) a,c,d

4. Consider the course table.

course(course_id, title, dept_name, credits).

Create a new course 'HS-001', titled 'SUPW', with 10 credits for department 'HSC'.

Identify the appropriate SQL [ ]

a) Insert into table course values('HS-001','SUPW', 'HSC',10)

b) Insert into course ('HS-001','SUPW', 'HSC',10)

c) Insert into course values('HS-001','SUPW', 'HSC',10)

d) Insert into table course ('HS-001','SUPW', 'HSC',10)

5. The command to remove rows from a table 'CUSTOMER'is: [ ]

a) Remove From Customer ... b) Drop From Customer . ..

c) Delete From Customer Where .. . d) Update From Customer . .

6. The primary key must be [ ]

a) Not null b) Unique c) a or b d) Both a and b

7. The set of permitted values of each attribute is called [ ]

a) Domain b) Tuple c) Relation d) Schema

8. SQL Query to find an employee whose Salary is equal or greater than 10000 is_____

9. SQL Query to find name of employee whose name Start with 'M' is _____ .

10. Select _____ dept_name from Instructor, Here which of the following displays the unique values of the column? [ ]

a) All b) From c)Distinct d) Name

11. Select * from employee where salary>10000 and dept_id=101; Which of the following fields are displayed as output? [ ]

a) Salary, dept_id c) Employee

b) Salary d) All the field of employee relation

12. Which of the following statements contains an error? [ ]

a) Select * from emp where empid=10003;

b) Select empid from emp where empid = 10006;

c) Select empid from emp;

d) Select empid where empid = 1009 and lastname = 'GELLER';

13. The employee information in a company is stored in the relation

Employee (name, gender, salary, deptName)

Consider the following SQL query

Select deptName From Employee

Where gender = 'M' Group by deptName

Having avg(salary) > (select avg (salary) from Employee)

It returns the names of the department in which [ ]

(a) the average salary is more than the average salary in the company

(b) the average salary of male employees is more than the average salary of all male employees in the company

(c) the average salary of male employees is more than the average salary of employees in the same department.

(d) The average salary of male employees is more than the average salary in the company

14. The relation book (title, price) contains the titles and prices of different books. Assuming that no two books have the same price, what does the following SQL query list? [ ]

Select Title From book as B

Where (Select count(*)from book as T Where T.price>B.price) < 5)

a) Titles of the four most expensive books

b) Title of the fifth most inexpensive book

c) Title of the fifth most expensive book

d) Titles of the five most expensive books

SECTION-B

SUBJECTIVE QUESTIONS

1. What is a relational model and explain different DDL, DML commands in SQL with syntax.
2. Outline the basic structure of SQL with suitable examples.
3. Write in detail about different types of constraints that can be specified on a relation.
4. List and explain set operations and different aggregate functions in SQL.
5. Consider the following schema :

Emp(empid, emp_name, emp_sal,Date)

i. Query to find second highest salary of Employee.
ii. SQL Query to find Max Salary from each department.
iii. Write SQL Query to display the current date.
iv. Find all Employee records containing the word "Joe", regardless of whether it was stored as JOE, Joe, or joe.

6. Write the SQL expressions for the following relational database?

Sailor schema (sailor id, Boat id, sailorname, rating, age)

Reserves (Sailor id, Boat id, Day)

Boat Schema (boat id, Boatname, color)

i) Find the age of the youngest sailor for each rating level?

ii) Find the No.of reservations for each red boat?

iii)Find the average age of sailor for each rating level that at least 2 sailors.

7. Consider the following relational schema:

Emp(*eid:* integer, *ename:* string, *age:* integer, *sala111:* real)

Works(*eid:* integer, *did:* integer, *peLtime:* integer)

Dept(did: integer, *dname:* string, *budget:* real, *managerid:* integer)

i. Give an example of a foreign key constraint that involves the Dept relation. What are the options for enforcing this constraint when a user attempts to delete a Dept tuple?
ii. Write the SQL statements required to create the preceding relations, including appropriate versions of all primary and foreign key integrity constraints.
iii. Define the Dept relation in SQL so that every department is guaranteed to have a manager.
iv. Write an SQL statement to add John Doe as an employee with *eid*= 101, *age* = 32 and *salary* = 15,000.
v. Write an SQL statement to give every employee a 10 percent raise.
vi. Write an SQL statement to delete the Toy department. Given the referential integrity constraints.

8. For the following relational database, give the expressions in SQL:

branch schema (branch name, branch city, assets)
customer schema (customer name, cus- tomer street, customer city)
Loan schema (branch name, loan number, amount)
Borrower schema (customer name, Loan number)
Account schema (branch name, account number, balance)
Depositersecham (Customer name, account number)

a) Find the names of all customers whos street address include substring 'Main Building'
b) Find average balance for each customer who lives in Harrison and at least four accounts?
c) Find all customer who have a loan at bank whose names are neither 'smith' nor 'james'.

9. Consider the following schemas:

Sailors (sid, sname, rating, age)
Reserves (sid, bid, day)
Boats (bid, bname, color)

a) Find the name of sailors who have reserved boat 103.
b) Find the names and ages of sailors with a rating above 7.
c) Find the names of sailors who have reserved a red boat.
d) Find the sname, bid, and day for each reservation.
e) Find the name of sailors who have reserved at least one boat.

SECTION-C

QUESTIONS AT THE LEVEL OF GATE

1. Consider the following relation :**Cinema (theater, address, capacity)**
Which of the following options will be needed at the end of the SQL QUERY?

SELECT P1.address FROM Cinema P1, Such that it always finds the addresses of theaters of theaters with maximum capacity?

(a) WHERE P1.capacity > = All (select P2. capacity from Cinema P2)
(b) WHERE P1.capacity > = Any (select P2. capacity from Cinema P2)
(c) WHERE P1.capacity > All (select max (P2. capacity) from Cinema P2)
(d) WHERE P1.capacity >Any (select max (P2. capacity) from Cinema P2)

2. Consider the following relations:

Student	
Roll_No	Student_Name
1	Raj
2	Rohit
3	Raj

Performance		
Roll_No	Course	Marks
1	Math	80
1	English	70
2	Math	75
3	English	80
2	Physics	65
3	Math	80

Consider the following SQL query:

SELECT S. Student_Name, sum (P.Marks)
FROM Student S, Performance P
WHERE S. Roll_No =P.Roll_No
GROUP BY S.Student_Name
The number of rows that will be returned by the SQL query is _____.

3. A relational schema for a train reservation database is given below:
Passenger (pid, pname, age)
Reservation (pid, class, tid)

Table: Passenger

pid	pname	age
0	Sachin	65
1	Rahul	66
2	Sourav	67
3	Anil	69

Table : Reservation

pid	class	tid
0	AC	8200
1	AC	8201
2	SC	8201
5	AC	8203
1	SC	8204
3	AC	8202

What pids are returned by the following SQL query for the above instance of the tables?

SELECT pid FROM Reservation WHERE class 'AC' ANDEXISTS (SELECT * FROM PassengerWHERE age > 65 ANDPassenger. pid = Reservation.pid)

1, 0 b)1, 2 c)1, 3 d)1, 5 [ ]