* MySql *

Dababase Includes
- Table includes
- Views
- stored Procedures
- functions

1) Creating Database

```
CREATE DATABASE db-name;

DROP DATABASE db-name;
```

2) Creating Table

```
USE db-name;
CREATE TABLE table-name (
    column-name1 datatype constraint;
    column-name2 datatype constraint;
    column-name3 datatype constraint;
);
```

3) Insert into Table

- INSERT INTO table-name VALUES
  (col1-val , col2, value , col3-val);

Date

★ SQL Datatypes

				★ USAGE
fixed len	1)	CHAR	string (0 - 255)	CHAR (50)
Variable len	2)	VARCHAR	string (0 - 255)	VARCHAR (50)
	3)	BLOB	string (0 - 65535)	BLOB (1000)
	4)	INT		
	5)	TINYINT		
	6)	BIGINT		
	7)	BIT		
	8)	FLOAT		
	9)	DOUBLE		
	10)	DATE		
	11)	YEAR		

★ TEXT

Types of SQL Commands

DDL : create, alter, rename, truncate & drop
DQL : select
DML : Insert, update, & delete
DCL : grant, revoke permission to users
TCL : start transaction, commit, rollback

Database Related Queries

CREATE DATABASE db-name ;
CREATE DATABASE IF NOT EXISTS db-name ;

DROP DATABASE db-name ;
DROP DATABASE IF EXISTS db-name ;

SHOW DATABASE ;
SHOW TABLES

DESCRIBE EMPLOYEE ;
→ This give the description table.

Table Related Queries

1) View All Columns

```
SELECT * FROM table_name;
```

2) Insert

```
INSERT INTO table_name
(colname1 , colname2)
VALUES
( col1-val1, col2 val1);
(col1 - val2, col2 - val2) ;
```

Ex.

```
INSERT INTO student
(mollno , name)
VALUES
( 101, "Ankush");
( 102, "Bhavesh");
```

★ INSERT INTO student VALUES (104, "Ankush");

	Primary key	Foreign key	Unique key	Candidate key	Super key
1)	Used to ensure data in column Unique	Column or group of column that provides link between data in two tables	Uniquely determine a row that isn't the PK	Uniquely identifyee records in column	—
2)	Uniquely identifyes ~~data~~ records in RDBMS	It refers to the field in a table which in the primary key of another table.			
3)	only one PK is allowed in Table	more than one FK allowed in TB	more than one unique key	more than on CK allowed	
4)	Not Null	Null values Allowed	Null value Allowed	Null value is allowed	
5)	It can not delete from parent TB	Its value is deleted from parent table	record allowed to delete, update from UK		
	PK can be CK			CK can not be PK	

Keys

1) Primary keys

It is a column (OR set of Columns) in a table that uniquely identifies each row.

There is only 1 PK

It should be not ~~not~~ null.

2) Foreign key

A foreign key is a column (or set of columns) in a table that refers to the primary key

There can be multiple FKs

FKs can have duplicate & not null.

3) Unique key

UK in DBMS is a set of one or more colums in a table that uniquely identifies each row in table

- No two rows can have same value in col.
- unique data not duplicate data

4) Candidate key

CK in a database BMS. is a col OR combination of col that uniquely identifyes data in each row in Table.

- multiple key in table,
- Each one have unique values.

Constraints

SQL constraints are used to specify rules for data in a table.

1) **NOT NULL** - columns can not have a null value

2) **UNIQUE** - All values in columns are different

3) **Primary key**:

CREATE TABLE temp1 (
id INT,
name VARCHAR(50),
age INT,
city VARCHAR(20),
PRIMARY KEY (id, name)
);

Combine table (column name

4) **Foreign key**
Prevent actions that would destroy links between tables

CREATE TABLE temp (
cust-id int,
FOREIGN KEY (cust-id) refrences customer (id)
);

5) DEFAULT

sets the default value of a columns

* Salary INT DEFAULT 25000;

6) CHECK

It can limit the values allowed in a column

```
CREATE TABLE city (
    id INT PRIMARY KEY,
    city VARCHAR(50),
    age INT,
    CONSTRAINT age CHECK (age >= 18 AND city = "DEL")
);
```

```
CREATE TABLE newtab (
    age INT CHECK (age >= 18)
);
```

Select in Detail

Used to select any data from the database

1) Basic Syntax :

SELECT col1, col2 FROM table_name ;

2) To Select All

SELECT * FROM table_name ;

3) SELECT DISTINCT city FROM student ;

1) WHERE clause
2) GROUP BY clause
3) HAVING clause
4) ORDER BY clause

Page No
Date

Clauses

1) Where Clause :- To define some conditions

SELECT col1, col2 FROM table_name
WHERE conditions;

Ex: SELECT * FROM student WHERE marks > 80;
SELECT * FROM student WHERE city = "Mumbai";

Operators for WHERE clause

- Arithmetic Operators : +, -, *, /, %
- Comparison Operators : =, !=, >, >=, <, <=
- logical Operators : AND, OR, NOT, IN, BETWEEN, ALL, LIKE, ANY
- BITwise Operators : &, |

Ex: 1) AND
SELECT * FROM student WHERE marks > 80 AND city = "";

2) OR
SELECT * FROM student WHERE marks > 90 OR city = "Mumbai";

3) ~~Between~~ IN
— student WHERE city IN ("Delhi", "Mumbai");

4) Between
— student WHERE city NOT IN ("Delhi", "Mumbai");

2) # limit Clause

Sets an upper limit on number of (tuples) rows to be returned.

```
SELECT * FROM student LIMIT 3;
```

```
SELECT col1, col2 FROM table_name
LIMIT number;
```

Order By Clause

To sort in ascending (ASC)
OR
Descending order (DESC)

```
SELECT * FROM student
ORDER BY city ASC;
```

```
SELECT col1, col2 FROM table_name
ORDER BY col_name ASC
```

Keywords :

1) ~~Lim~~ LIMIT :

Restricts the numbers of rows returned in the result set.

2) OFFSET

specifies the number of rows to skip before starting to return rows.

Qu.3 How do you get the first 20 rows of name and rankscore from movies ?

→ SELECT name, rankscore
FROM movies
LIMIT 20 ;

Qu.4 why to use LIMIT and OFFSET ?

→ LIMIT :

~~Eff~~ Efficient Resource Usage

→ OFFSET :

Enables Pagination

Qu.5 How do you get 20 rows of name and rankscore starting from the 41st row in movies ?

→ SELECT name, rankscore
FROM movies
LIMIT 20
OFFSET 40;

Aggregate Functions

- Aggregate functions perform a calculation on a set of values, and return a single value
  - COUNT()
  - MAX()
  - MIN()
  - SUM()
  - AVG()

1) Get Maximum Marks

```
SELECT max (marks) FROM student;
```

2) Get Average Marks

```
SELECT avg (marks) FROM student;
```

Group By Clause

Groups rows that have the same values into summary rows.

* Generally we use group by with some aggregation function.

Qu. Count number of students in each city

```
SELECT city, count (name)
FROM student
GROUP BY city;
```

Ex

```
select ~~count(mode)~~ mode, cot
from Payment
Group by mode;
```

Having Clause

Similar to where & Applies some conditions on rows. Used when we want to apply any Condition after grouping.

* Count number of students in each city where max marks cross 90.

```
SELECT count(name), city
FROM student
GROUP BY city
HAVING max(marks) > 90;
```

General Order

```
SELECT columns1, column2
FROM table_name
WHERE Condition
GROUP BY columns1, colum2
HAVING Conditions columns2
ORDER BY Columst, ASC of DESC;
```

SET SQL_SAFE_UPDATES = 0;
SET SQL_SAFE_UPDATES = 1;

Page No.
Date

Update (To update existing rows)

UPDATE table_name
SET col1 = val1, col2 = val2
WHERE condition;

Ex UPDATE student
SET grade = "O"
WHERE grade = "A";

Delete

DELETE FROM table_name
WHERE condition;

DELETE FROM student
WHERE marks < 33;

Cascading for foreign key

1) On Delete Cascade

when we create a foreign key using this position, It deletes the referencing rows in the child table.
when the referenced row is deleted in the parent table which has a primary key.

2) On Update Cascade

When we create a foreign key using UPDATE CASCADE the referencing rows are updated in the child table when the referenced row is updated in the parent table which has a primary key.

```
CREATE TABLE student(
    id INT PRIMARY KEY,
    COURSEID INT,
    FOREIGN KEY (COURSEID) REFERENCES course(id)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

Alter

1) ADD Column
ALTER TABLE table_name
ADD COLUMN column_name datatype constraints;

2) DROP Column
ALTER TABLE table_name
DROP COLUMN column_name;

3) RENAME Table
ALTER TABLE table_name
RENAME TO new_table_name;

4) CHANGE Column
ALTER TABLE table_name
CHANGE COLUMN old_name new_name
new_datatype constraint;

5) MODIFY Column (Modify Datatype / Constraint)
ALTER TABLE table_name
MODIFY col_name new_datatype new_constraint;

Truncate

→ DROP → Delete DB

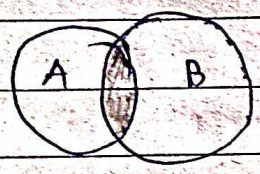
→ TRUNCATE → Delete Table Data

TRUNCATE TABLE table-name;

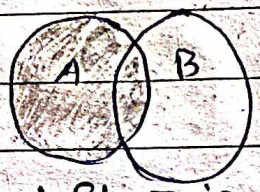
Joins in SQL

Join is used to combine rows from two or more tables, based on a related column between them.

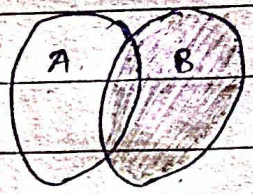
Types of Join

1) 

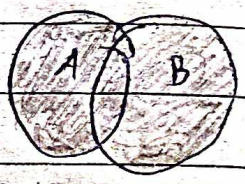
Inner Joins

2) 

left Join

3) 

Right Joins

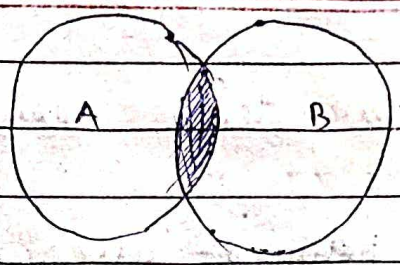
4) 

Full Join

Outer Joins

Date

1) Inner Joins



Returns records that have matching values in both tables.

Syntax:

```
SELECT      columns (s)
FROM        table A
INNER JOIN  table B
ON  tableA. col-name = tableB. col-name;
```

Ex.
```
SELECT *
FROM student
INNER JOIN course
ON student. student_id = course. student_id;
```

With out specifing INNER it consider default as INNER JOIN

Ex. student

student_id	name
101	adam
102	bob
103	carey

Course

student_id	course
102	english
105	math
103	science
107	cs

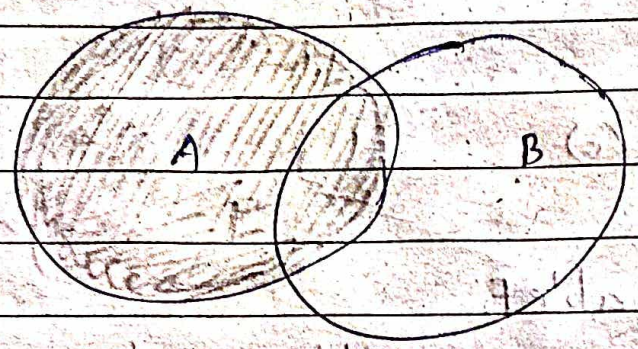
Result

student_id	name	course
102	bob	english
103	carey	science

} Common Values.

2) left Join

Returns all records from the left table, and the matched records from the right table.



Syntax:

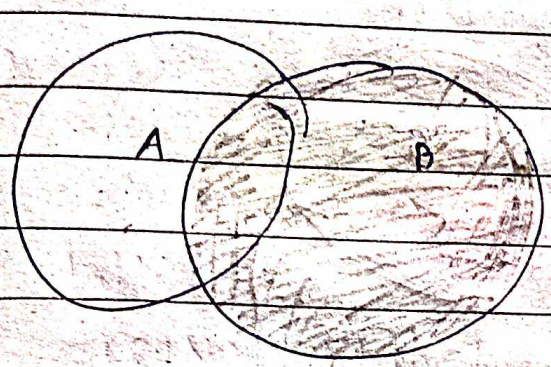
```
SELECT *
FROM tableA
LEFT JOIN tableB
ON tableA.colname = tableB.col-name;
```

Ex

```
SELECT *
FROM student as S
LEFT JOIN course as C
ON s.student_id = c.student_id;
```

3) Right Join

Return all records from the right table, and the matched records from the left table



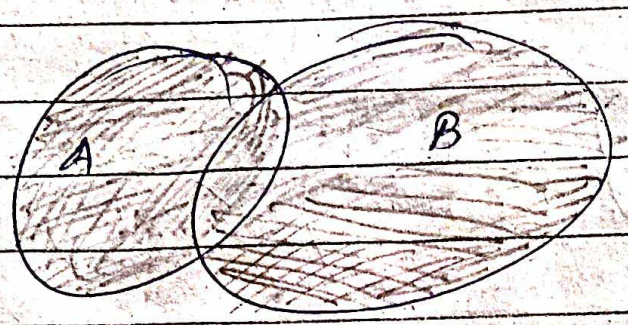
Syntax

```
SELECT Columns
from tableA
RIGHT JOIN tableB
ON tableA. col-name = tableB. col_name;
```

```
SELECT *
FROM student as S
RIGHT JOIN course as C
ON s.student_id = c.student_id;
```

4) Full Joins

Returns all records when there is a match in either left or right tabl



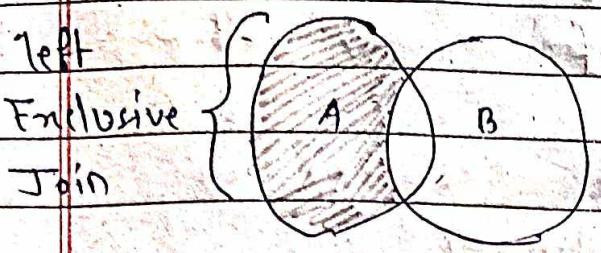
LEFT JOIN
UNION
RIGHT JOIN

```
SELECT * FROM student as a
LEFT JOIN course as b
ON a.id = b.id
UNION
SELECT * FROM student as a
RIGHT JOIN course as b
ON a.id = b.id;
```

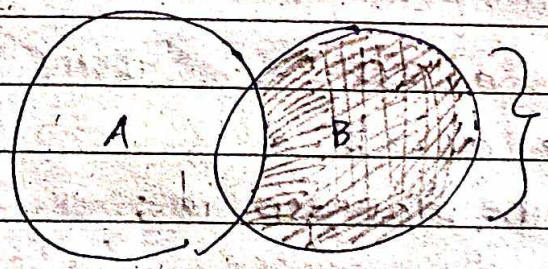
OR

```
SELECT * FROM student as a
FULL OUTER JOIN course as b
ON a.id = b.id
```

* Write SQL commands to display the right exclusive join




```
SELECT *
FROM student as a
LEFT JOIN course as b
ON a.id = b.id
where b.id IS NULL
```




```
SELECT *
FROM student as a
RIGHT JOIN course as b
ON a.id = b.id
WHERE a.id IS NULL;
```

Inner join and self join is same

5) Self Join

It is a regular join but the table is joined with itself

Syntax

```
SELECT  columns
FROM    table as a
JOIN    table as b
ON      a.col_name = b.col_name ;
```

Ex

```
SELECT  *
FROM    employee as a
JOIN    employee as b
ON      a.id = b.manager_id ;
```

Union

It is used to combine the result-set of two OR more SELECT statements.
Gives unique records

To use it:

- Every SELECT should have same no of columns
- Columns must have similar data type
- Columns in every SELECT should be in same order.

Syntax:

```
SELECT columns FROM tableA
UNION                UNION ALL - It will give All Valu
SELECT columns FROM tableB
```

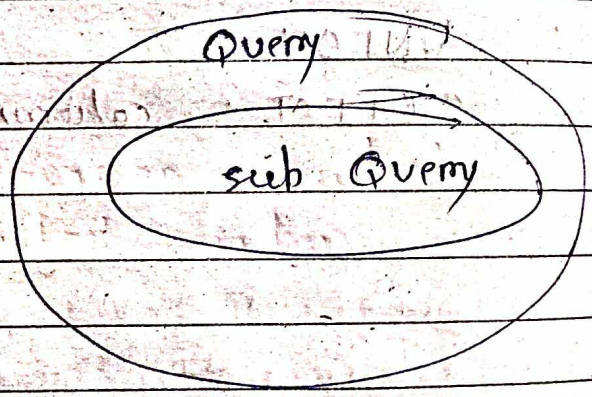
Date

SQL Sub Queries

A subquery or inner query or nested query is a query within another SQL query.

It involves 2 select statement.

Syntax
SELECT Columns
FROM table_name
WHERE col_name operator
(subquery);



Page No.
Date

MySQL Views

A View is a virtual table based on the result-set of an SQL statement.

```
CREATE VIEW view1 AS
SELECT rollno, name FROM student;

SELECT * FROM view1;
```

A view always shows up to date data. The database engine recreates the view, every time a user queries it.

Page No.
Date

Qu. what is cursor ?

→ Cursor is a work area where the result of a SQL query is stored at server side.

There are two types of cursors are
1) Implicit Cursor
2) Explicit Cursor

1) Implicit Cursor : It is automatically generated by oracle while an SQL statement is executed.
If we don't use an explicit cursor for the statement.

2) Explicit Cursors : It is defined by the programmers to gain more control over the context area.

Qu. what is Trigger ?

→ Triggers are stored programs which are automatically executed when some events occur.

Benifits
1) Gain strong control over security.
2) Enforcing Referential Integrity
3) Auditing
4) Synchronous replication of tables

2] Clustering Index

- Cluster indexing is a database indexing technique that is used to physically arrange the data in the table based on the value of the cluster index key.

- Means, the rows in the table are stored on disk in the same order as the clustered index key.

* Defination.

1) clustering index is created on non-primary key columns which may not be unique for each record.

2)

DATE: / /

3] Secondary Indexing

- It is a database management technique used to create additional index on data stored in a database.

Qu. what is indexing?

Indexing is used to improves database performance by minimizing the number of disc visits required to fulfill a query.

Ordered file	Primary Index	clustered Index
Unordered file	Secondary Index	Secondary Index

Types of Idexing In DBMS

1) Primary Index

- It is an ordered file whose records are of fixed length with two fields.
- first field of the index is the primary key of the data file in an order manner.
- Second field of index the ordered file contains a block pointer that points to the data block where a record containing the key is available.

Qu what is Normalization ?

→ 1) Normalization is the process of organizing the data in the table.
2) Used to minimize the redundancy from a relation.
3) Normal form is used to reduce redundancy.

The main reasons for normalizing the relations is removing the these anomalies.

failure to eliminate analomolies leads to data redundancy.

1) Insertion Anomaly :
2) Deletion Anomaly
3) Updation Anamaly

1] 1 NF
No multivalued attribute

• Only single valued

2] 2 NF

- In 1st NF + No partial Dependency
- Only full Dependency

3] 3 NF

- In 2nd NF + No Transitive Dependency
- No non-prime

4] BCF - Boyce Codd normal form

- In 3NF + L.H.S. must be ~~SK~~ CK OR SK

5] 4 NF

- In BCNF + No multivalued Dependency

6] 5NF

- IN 4th NF + loss less Decomposition.

Date

Qu.

	DELETE	TRUNCATE
1)	DELETE command in SQL removes one or more rows from a table based on the specific conditions.	TRUNCATE command is used to purge a table of all of its row.
2)	It is DML Command	It is DDL
3)	Necessary to Commit after making changes	It commited Automatically
4)	WHERE clause is used for species conditions	WHERE does not perform any conditions.
5)	It requires authorization for delete	It requires table alter permission
6)	when it comes to large database, it is much slower	It is faster, or more accurately, instant.

ACID Properties in DBMS

A = Atomicity - The entire transaction takes place at once or doesn't happen at all.

C = Consistency - The database must be consistent before and after the transaction.

I = Isolation - Multiple Transaction occurs independently without interference.

D = Durability - The changes of a successful transaction occurs ~~even~~ even if the system failure occurs.