Data-Base

Database management

Q) Transaction storage engine in mysql?
We use MySQL innoDB, storage engine which is by default since mysql 5.5

Q) Shading in SQL →
process of breaking up large tables in small chunks (called shards) spread accross multiple servers.

Q) Triggers
① Before insert ② After insert ③ Before Update ④ After Update ⑤ Before Delete
⑥ After Delete
→ Trigger is a task that is triggered in responce to some database event

Q) Create and execute View
→ create view trainer as
select * from Courses.
* they are virtual tables that are created using other tables data

* Blob :- is acronyoum for binary large object used to store very large amount of data,
Tiny Blob, blob,

Q) Temporal Datatypes
DATE → "ccyy-mm-dd", time = hh:mm:ss, DateTime → ccyy-mm-dd hh:mm:ss, Timestamp :- 'ccyy-mm-dd hh:mm:ss
Year - ccyy or yy

---

mysql features
① open source ② support on every OS ③ Query Caching

Remove Column
Alter table emp drop salary;

SQL

Drop → removes complete structure, Truncate → just deletes all columns and table structure is retained

ACID :- 1) Atomicity → transactn is complete or rejected
2) Consistency → updates in DB follow rules & regulatn
3) Isolatn →
4) Durability → committed transitions are stored permanently

pattern matching
wild card → '%'

select * from emp where ename NOT like '%LL';
↳ all name ending with 'LL' are fetched

%A% → gives names containing 'A'

wild card → '_'

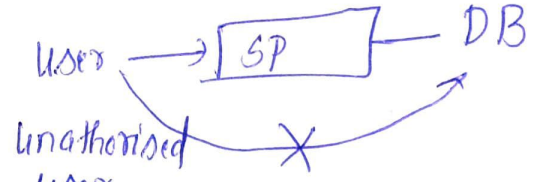
'__k%' → all names where 'k' is at third position

'____' → names with four chars

Stored procedure

```
Delimiter $$
Create procedure fetchAllEmp()
Begin
select * from emp;
End $$
Delimiter ;
```

⊗ User with no access to DB can be allowed via SP



Aggregate or Scalar functn.
↳ fxn form Action on collection of values but returns single Value

Avg(), count(), min(), max(), sum(), first(), last()

Delete, Truncate, drop 108

⊗ delete does not frees the memory, drop & truncate frees the memory

Drop → Complete table structure is droped
Truncate → all columns data removed completely, not structure
delete → used to delete row

Commands Database

Delete database → Drop Database sunbeam21

to get all databases → show databases

To remove data → Truncate table emp

To Create Database → Create database sunbeam21

change delimiter → delimiter $$.

To delete row → delete from emp where id <1000

To remove Complete data → truncate table emp;

To remove along with structure → drop table emp.

using In → select * from emp where name in ("smith", "cat");

Union & union-all

↳ no duplicate ↳ duplicate

Alias → select sum() as total from emp;

Union → combines result of 2 select statement; (i.e 1 or 2)

Minus → select * from student_contact
minus
select * from parent_contact

→ present in student but not in →

Select + where + order by + Group by + having

① where → to filter the records.

② Order by → ASC, DSC

③ select count(id), country from students where country != "India"
Group by country Having count('id') >= 5

Subquery → query within Query.

select * from emp where empid = (select empno from data where name = "deep")

Correlated & non-correlated

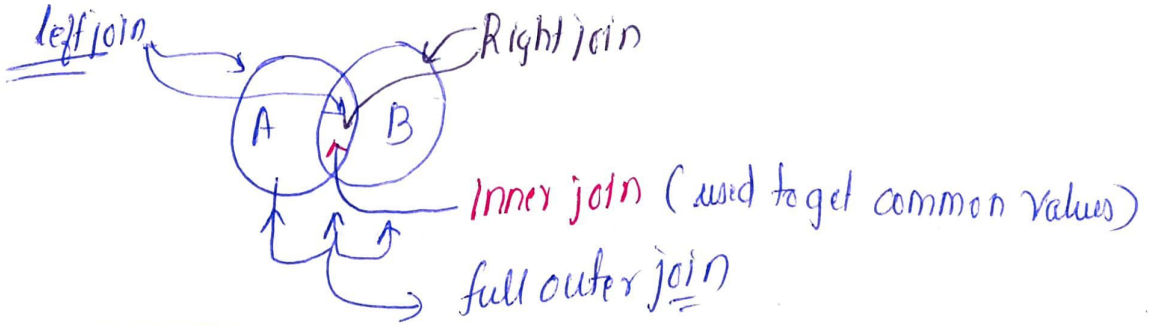
Cross join :- it is cartesian product of two tables included in the join,

if where is used than it is converted to inner join

Self join :

Joins

used to combine data from 2 different tables.



Inner join → select * from emp e join table dept d

→ Select e.ename, d.dname, from emp e inner join dept d
on e.deptno = d.deptno;

left join Select * e.ename, d.dname from depts d left join emps e
on deptno = deptno

foreign key → may be single or collection

* refers to another table primary key, the table containing f-key is child and referencing is parent

e.g Create table student (id int, name varchar(30), library_id int,
primary key(id)
foreign key (library_id) References library(library_ID)
↑ ~~this name~~ ↓ Parent table

* alter table orders
Drop ~~constraint~~ drop foreign key Fk_personOrder;
* alter table orders
add foreign key(personId) references persons (PersonId);
↑ Column name

Constraints

① Not null ② check → checks if all values satisfies condition ③ Default → assigns default.
④ Unique ⑤ index ⑥ pk ⑦ Fk

DBMS	RDBMS
* data is stored as file	as table
* usefull for small amounts	huge data manupulation
* no relation among data	inter related data
* No normalisation	Normalisation present
* data redundacy is more	data redundacy is less
* xml, windows Registry	Mysql, SQL, Oracle

1NF → No duplicate column names.
2NF → seperate data in different table. and add primary key
3NF → remove data that is not dependant "on" primary key in seperate table.

Transaction → delete ✓ Truncate ✗ Drop ✗

Database

* Inner join used to fetch data based on condition.

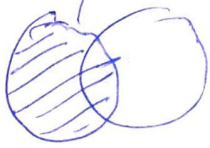
* equi-joins:- When inner join condition is of equality check then equijoin else (> < >=) non-equijoins

* == → DBMS → =

* inner join →  shows Intersection

Left outer join

select e.ename, d.dname from emp e left outer join dept d on d.deptno = e.emdeptno.



outer keyword optional

left outer = left

* from → left/Right/inner join → join

on → condition

Set operator:- used to combine two query result. on (Same no of Column)

Union & union all → column name selected from here

select dnam from dept

union all / union

select ename from em

all duplicates also considered

duplicate removed

multiple inner join structure

select <——>

from emp e

inner join empa ea on ea.()=e.()

inner join meetingm on —»— —

inner join addres on —»— —

select → join → join → where → Group → Having

select column From table1

xxx join table2 on condition

xxx join table3 on Condition

where Condition

Group by column

Having condith → order → limit

Sub Query

All → select * from emp where sal > ALL (select sal from emp where job = 'salesman');

(>1250 & >1100 & >1000);

ANY → select * emp where sal < ANY (select sal From emp where deptno=20)

sal < 800 or sal < 3000, or sal < 1100

ANY Vs IN → used for equality only

↳ used in subquery

↳ <, >, =, != both are "or"

ANY Vs All → both used with subquery.

(>, <, =<, =, !=)

ANY → Or, and → All

NOT IN Wher

Where deptno IN (select eno)

Exists Vs NOT exist

select * from dept d where Exists (select e.deptno from emp e where e.deptno = d.deptno

59 Comment --

My sql data type

- Tinyint (1byte) ② small int (2byte)
- ③ Medium int (3byte) ④ Bigint (8 byte)
- Decimal → (m, n)

(5,2) = 100.32

(int → 4byte)

5 digit, 2 digits after decimal

- Float 4byte, Double → 8b (65bit precision)
- When string is converted into to number than it is converted into decimal double floating point number
- bit field value are prefixed using b'1010' 0b1000
- if Ansi_Quotes enabled than " " also treated as ' ' and cannot used as string literal

" " → tuned to identifier-quoting character

- hex notation 0x0A
- Octal Noth 0882

binary string :- they are unlike char string they are used to store images

Introducer :- is a signal for character string literal :-

Convert() → used to convert old string to new

char_length → gives string length in character, no of characters are counted

length() → gives length of string in bytes

character_set_system → system variable

server → set when default DB change

char → same size allocated as at time of allocation

varchar → allocated size varies

char → (1-255) Longtext → 4G

varchar → (1-65535) MedText → 16 M

Text → 64K

show character set used to get all character set

char < varchar < Text < long text

medium text → suitable for storing small Notes

longtext → document

Numeric Datatype

Tinyint < small int < medium < int

---

float(4,2) →

Total digits, atmost (0, 1, 2)

12.33, 24.13, 124.4 ✓

Datatype to store Date, time, timestamp, Datetime

- Temporal data type
- Default format → yyyy-mm-dd

HH:MM:SS

Datetime	Stamp time
When we want same timezone across whole world	When we want variation in time
⊗ We i/p	it is saved as per system time

Event: is a task that run according to schedule

Show Column & Describe can be used interchangably both are same
:- used to get Column details

char → Binary
Varchar → VarBinary
Tinytext → Tinyblob
text → Blob
M.text → Medium-blob
L.text → Long-blob

Spatial data type → Geometry, point
linestring polygon
↓
curve

Unsigned & Zerofill → numeric type
Character set & collate → non-binary strings

Blob, text, Geometry, Json cannot assigned default value

size of bit depends on length of field.
≈ (m+7)/8 bytes

SET → used to hold any no of string object values
set("one", "two").
(1 to 8) bytes allowed
64bit

SQL syntax

⊗ To alter Database attribute
→ alter database

--force used to run with error occured

like →
select —»— where ename like 'm%'
'%h' → abcdh.
'%o%' → num.
'%A%A%' → adams

underscore =
where ename like '____'
only 4 chars Consider

--R%

Branching flow control
Case
LEAVE used to exit block of code

Comment
#, --, /*...*/

SELECT claus
⊗ it is executed last

NoSQL

Database types
↳ RDBMS
↳ Nosql
↳ NewSql

Acid
↳ Durable
→ Isolated
→ Durable
→ Atomic

Codds rule

↳ all database systems must be able to manage data bases entirely through relatn capabilities

limitatn of RDBMS

⊗ cannot process petabytes of Data

⊗ No acid transaction in Nosql

⊗ Nosql is eventually consistent (changes appear slowly)

⊗ famous for highperformance, Scalability availaibility

Base Transaction

Basically Available → 24x7

Soft state → data can travel implicitly w/o explicit command

Eventualy Constitent → changes reflected eventually.

Cap (Brewers theorem)

dat must be stored
C → consistency
A → availability
P → partitiontolerance

↓
2ux7 availibity

↓
More tolerance b/w two independant Server machin

Types of Databases

① Key-Value Database
② Column oriented DB
③ Graph Database
④ Document oriented BD

① → Dynamo Db Amazon.
② → hbase Cassandra
③ Neo4J, Titan,
④ key Value paire as JSON.
↳ tables → Collections
↳ Row → Document
↳ Columns → key-value pair

Mongo DB

↳ data stored as JSON.
↳ Data stored as Binary JSON
↳ Server : mongod (C, C++)
↳ mongo : client shell
↳ JavaScript Notatn.

Data types

data	bson	Value
null	10	
boolean	8	True/fals.
number	1/16/18	123, 12.5,
String	2	" " or ' '
Date	9	new Date(),
array	4	[ ]
Object	3	{ }

① show Ddatabase
② use classwork
↳ Creates and switch to classwork

Rows → Document

field → name - value pair

To create Table

db.createCollection("colname");

Insert Query

db.data.insert({ → automatically generated fababl.
name: " ",
Surname: " ",
})

db.data.insertMany([{},{},{}]);

_id = Generated if not given

12 byte → [counter(3), | pid(2) | machin(3) | timestam(4)]

.find() →

Show Collections

db.collection.find({criteria})
({},{Project})

{ename:1, _id:0}

1.e inclusion or Exclusion supported

Mongo operator

$eq $ne $gt $gte
== != > >=

$lt, $lte
< <=

$and, $or, $nor

$in $nin
↳ not in

Criteria

db.emp.find({ename: "king"})
select * where name = " ";
select sal >2900

db.emp.find({sal:{$gt:2900}})
deptno != 30

db.emp.find({deptno:{$ne:30}})
dept != 30 && job = 'Analyst'

db.emp.find({
$and:[
{deptno:20
job:'Analyst'
}
]
});

sal < 1000 or job = "pres)

db.emp.find({
$or:[{sal:{$g:100}
job: "pres"
}]})

Select * where name in ('a','b','c')

```
db.emp.find({
    ename:{$in:['a','b','c']}
},{
});
```

exists →

db.emp.find

pretty(), sort(), skip(), limit(), count(),

.find().pretty().

.find().sort({sal:-1}) (-1 → des, +1 → Asc)

.find().sort({sal:1, job:-1})

.find().sort().skip(2).limit(1)

$ / $R

/R$/ → Turner
last letter R

'R$'

'^a' → start with 'a'

'k..g' → king ✓
kang ✓

Regexp

'a|k' → all string with a or k is displayed

'a' → gives strings with 'a' as its content

'a*'

Quantifier (to tell conti...)

+ → one or more time to left side

s+ → 'ss' 'sss'

? → 0 or 1 time

as? 'a' 'as'

a, as, ass ✗

0 1

(x)* → expression to left 0 or more

Br* → B, Brr, Brrr ....

{x3} → ma b{5} b→

ma bbbbb ✓

{x,} → x or more

3 → 3 or more

{x,y} → x → atleast, y → less than y

Pr{3,6}a → Prrrra Prrrrra
4 5

Group & Range

() → match every thing inside it

A(xy) → A(xy) "Axy"

Bxy

{} → x2{4,6} → x22222
6 4

[ ] → match any char from range

x2[atp]r → x2ar, x2tr, x2pr

[pqr] => "p", "q", "r"

matches p q r individually.

Pratik Lapn
Queer Lapn
Rajneel Lapn

[pqr][xy] → px py
qx qy
rx ry

[?:...] → A(?:ntIpple)

Apple, Ant.

[^...] → Ab[^pqr] → matches character not defined in bracket

[a-z] →

↳ Matches small case letter a-z

^[a-zA-Z] → start with small case or upper.

[0-9] → match 0 1 2 3 4 5 6 7 8 9

[aeiou] →

- remove({criteria});
- delete({});
- deleteMany({});

db.people.remove({email:"a@gi"})

- drop → To delete Collection

---

db.emp.update({
_id = 2
},{$set:{
sal:1600
}
});

$inc:{
comm:10
}

comm = comm+10

UPSERT → UPDATE + INSERT

db.emp.update({},{},true);

Stored Procedure

↳ MySQL PSM → Persistent Stored Module

↳ it is stored in compiled form.

↳ execution is very fast

Stored Procedure

↳ no return value for SP

↳ can take 0 or more argument

↳ create procedure

↳ drop procedure.

↳ invoked using "CALL"

↳ Results are returned via Out Param

↳ inserted in other table

↳ produced using select statement

↳ changing delimiter is mandatory form

```
Delimiter $$
Create Procedure hello()
Begin
  insert into result values(
    1,'HelloWorld')
End;
$$
Delimiter ;
```

Call hello()

To load stored procedure

Source filepath.

To execute → call area_perimeter()

Syntax

Declare Variables

Declare Variable Varname Datatype;

~~Declare Varname Init value Default~~

Declare varname Datatype Default Init Val;

Set Varname = new Value.

Select new Value Into Varname.

Select expr or columnname into Varname from table_name;

Parameters

```
Create Procedure hello (ParameterType  P1 Datatype)
Begin
...
End
```

IN → initialised by programmer.

Out → initialised by procedure

InOut → initialised by programmer

e.g
```
Delimiter $$.
Create Procedure hello(Out P1 int)
Begin
    select 123 into P1;
End $$
delimiter ;
```

set variables on Console

Set @var = 10 ; call hello(@var)

** @ is used set Variables

If-Else

```
If Condition then
  then body
  End if;
```

```
If Condition
then body
Else
    Body
End if.
```

Loops

```
while Condition
  Do body
  End while;
```

```
repeate
    body
until
    Condition
End repeat
```

```
label: Loop
if Condition
  then
    ~~~
    ~~~
    Leave label;
  End if
  ...
End Loop;
```

Show Procedure

Show procedure Status like 'hello%';

Break

Drop procedure

drop procedure If Exists hello;

Case When

```
Case
When condition
then
    body;
When condition
then body;
Else body;
End Case;
```

Function

↳ can have 0 or more parameter only 'IN' params allowed

↳ Return statement allowed

↳ it is stored in system Table

↳ all if else, loops are allowed

⊗ two types of function

① Deterministic

② non deterministic

Create function

```
Create Function hello(p, Type)
Returns Type
[NOT] deterministic
Begin
    body;
    Return value;
End
```

Show Function

```
show function status like 'hello';
```

Trigger

↳ no return, no argument

↳ operated on 'DML' operat

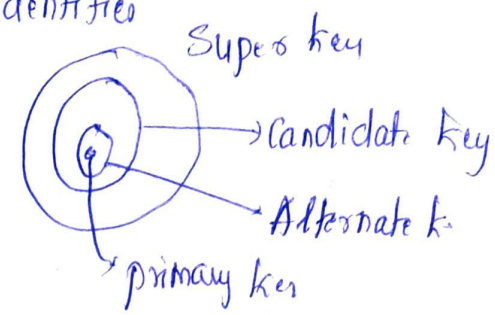
↳ ⊗ Before and After 'DML' operation

Normalisation

⊗ Combining all data in one table, Removing computed column ↓ are resulting to unnormalized form.

Super key → all possible primary key

Candidate key → key that uniqly identifies



foreign key → primary key of parent included in child as reference to some row in parent

① Removing repeating to new table

② key element will be pk of new table

③ add pk of original table to new table.

① remove repeate ② give identity

③ link to original table.

1NF ↑

2NF → consider only Table with Pk if some columns are

⊗ not dependent on pk than put them in seperate table.

⊗ → add dependency as prim k

User should enter minimum data in column

Transaction

- Is set of DML queries executed as single unit
- my sql autocommit is bydefault 1 → true @autoCommit = 1

Savepoint

↳ is state in db within transaction
↳ need to created at each logical section
↳ Roll back to any level is possible

Start Transaction
save point @point1;
savepoint point2;
Rollback to point1;
→ all save points cleared

Optimistic locking:- my sql automatically locks row on which update or delete is taking place

Manual row locking → pessimistic locking

For update → added to select query
~~select * from emp where deptno = 10 for update~~
If we have primary key then "Row locking is possible" or else Table locking is done

Index

Create index on ~~column name~~ table (column)

↳ it can be asc desc
↳

Index types

① simple index
② unique index
③ Composit index
④ cluster index

Hotseat

Normalisation

why?
→ to reduce redundancy
→ to prevent insert update & delete anamoly

→ emp table: empno, ename, mgr, job, hire, sal, comm, deptno
→ dept table → deptno, dname, loc
- If we combine both table dept table content will be repeated

step 1 → put all data in one table which we do in one transaction (vnf)
- ↳ decide meaningful table name
- ↳ decide primary key, datatype. Remove computed field

↳

1NF → Deal with repeating group and put in new table
- primary key of main table to be put in that new table.

One to many
Outcome → tables with non-repeating data

2NF → Deal with Composit key
- If fields are not fully dependent on composit key seperate them into new table and maintain 'PK' to be mainted in that table for relatn

Ordered_Product

Oid, pid, prate, pqty, pname, odate, odelivery

Cpk → Oid + pid

Out come →

* order * ——— * product

many to many relation

3rd NF → Columns not depend on "pk"

(X) Customers

users → id, name, email, phone, flat, street, city, pin.

(X) Orders → oid, odate, cid, cname, ccity

If field are not directly depend on pk they can be taken in new table

4nf bcnf → to reduce data entry mistake

address ——→ pin, street, benchmark or ds.

from user just take pin & derived data from it

Denormalised form

(X) to reduce time you can keep denormalisation

(X) nth highest salary

select * from emp order by sal Desc limit 2, 1 // skip 2 and get next

problem → in duplicated salary

By Subquery

select distinct sal from emp order by sal desc limit 2, 1 → get next

→ skip 2

get emp of that sal

select * from emp where sal = (

(

display emp, ename, ejob, esalary along with salary of all emp

select ename, ejob, esalalar sum(sal) from emp.

(select ename, job, sal from emp) union (select null, null, sum(sal) from emp

ename, job, sal, total_sal

Display emp name, job sal, along with total sal of emp in that dept ?

select ename, job, sal, (select sum(sal) from emp where deptno = o.deptno) dept_sal from empo

(6) find managers manger of employee

soluth → self join

(*) empe emp m emp mm

→ Select e.enam m.enam from empe Inner join emp m ~~where~~ On e.mgr = m.empno. Inner join emp mm On m.mgr = mm.empno;

How stored functn & stored procedure different

(*) in function returns value, Sp dont return value

(*) no out parameter in functn

Call (Stored Pr) call hello_sp(???)

SF call select function(??) From emp.

#sql Functn → map()

① single row ② multi row ③

Scalar functn returns single row ↓ reduce() returns multiple rows

Table valued Functn

---

when n > m ∈ flatmap()

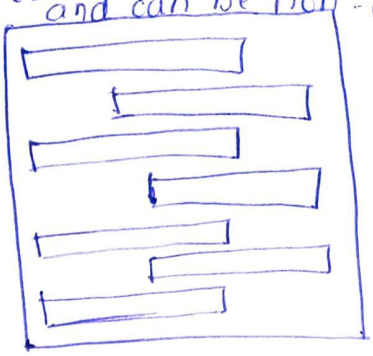
m rows → n row

Clustered Index

Indexes stored in form of Hash data structure

BTree

each row have diff addrss and can be non-consequitive (scattered)



Clustered Index is used as per hardware addrss

(*) (2 are mapped to primary key of the table

(*) If table do not have clustered index mysql stored on some hidden column

---

Continued after Qs

short note

→ Create drop
Syntax of Join, Lik
Technology Note

DBT

Server name:- mysqld.exe
Client → mysql.exe

① Create new user

Create user sunbeam @ localhost Identified by 'sunbeam'.

② Create data base classwork

③ drop alter delete,

④ Sql is case sensitive

DDL → data definition Languag
↳ Create, alter, drop, Renam.

DML → data manipulatn languag
Insert updat delet

DQL → data Query Languag
→ select

DCL → Data Control Languag
→ Create user, grant, Revoke

TCL → Transactn control Languag
→ Savepoint commit Roll back

special characters → `deepak#2`

Grant

grant all previligies on db.* To dbuser@localhost
flush previligeg.

Create Table

create table student (id int, name varchar(20), marks double);

Insert →

insert into student values (1, "abc", 85.5)

⑤ tables are stored in
student.idb

database → subdirectory of data
table → .idb file

Insert

all cols on.
insert into * values (x, z, a b)

certain cols :-
insert into x (y, z) values (y, z)

multiple rows
insert into x values (a, b, c) (d, e, f)

Insert from another table
insert into x select a, b, c from y table

⑥ Select user(), database().

Clear screen → cls

⑦ Desc tablname

mysql data types

① Numeric Types
Tinyint(1b) smallint, mediumint, int, bigint

② Floating type
Float, double,
decimal(m, n) → exact prices ra

Date → date, time, datetime, timestamp, year

DBT 4

String type:

char, varchar(30), tinytext, text, mediumtext, long text,

Binary

Binary, Varbinary, blob,

Enum, set

Speed of access

char > varchar > text

Execution of sql script

terminal > mysql -u user -p manager db < /path/to/sqlfile.

---

mysql > source /path/to/sqlfile.

Computed Columns

select c1, c2, c3, expr1, expr2, from table

select c1 (case ...) from table.

Distinct

select distinct c1 from table.

limit

select * from table limit 'n'.

limit 5, 1 → fetch
↳ skip

between

c1 between val1 and val2;

In operator day 3 → 1

c1 in (val1, val2, val3)

Like

c1 like "%A" abcd A

c1 like '___' → three char.

decimal (5,2)

total digits → 5,
2 digits after decimal.

Computed column

select ename, sal, sal*0.5 from emp;
AS DA

Case

Case
when deptno = 10 then "a"
when deptno = 20 then "b"
else 'unknow'
End as dept

Order by

select * from emp order by deptno, Asc,
job desc;

operators in Where.

=, >, <, != → day 3 → 1

where dept = 10,

where sal >= 1000 and sal <= 2000;
Or

Where NOT job = 'salesman'.

⊗ Relational operator not used with nulls ✗

select * from emp where comm = null

Is null

select * from emp where comm is null;

Is not null

Update → DML

update tablename set c2 = "dk" where
id = 3;

All row

Update xtable set c2 = c1 * c3

Delete, Drop, Truncate

↳ Delete
  ↳ To delete one or more rows
  ↳ delete from xtable where c1 = value;
  ↳ Delete all Row
    Delete from xtable.

Truncate
  delete all rows
  ↳ truncate table xtable

Drop
  ↳ delete all + table structure
    drop table xtable if Exists

Speed of delete
  drop > truncate > delete

Help help select, help functions, help sign.

Dual table
select 2*3 from dual;
  or
select 2*3;

Inbuilt Functions
single row functions
  one row i/p one row o/p
multi row or group functions
  sum(), max(), min()

Numeric function
abs(), power(), round()
floor(), ceil(),

String function

ascii(), char()	upper(),
concat()	trim, Ltrim(), Rtrim()
substring()	Lpad(), RPAD()
Power()	Regex_like().

(X) between includes both ranges

Problem with Where
Select ename from emp where ename b/w 'B' and 'J'.
James & Jones → not displayed because of Ja, Jo, after J,

not between
sal not between 1000 and 2000
  Between or Betwe..

Like Oper:-
Where ename like 'm%'
  moh
  mans

like '%h'
  abch
  defh

'%v%'
  abvbc
  abcvdc

'%A%A%'
  ADA
  AkkA
  ANNA

ename like '____'
  4 lettered word
'___R%'

drop

* drop table, database,

DML can be rolled back

DDL cannot be rolled back

↳ drop, truncate

* upper('mysql')
* substring('sunbeam', 4, 2)

---

left("devdang', 3) => dev.

Right("devdang', 3) => ang

substring('abcdefg', 3)

123

cdefg

('abcdefghijk', 5, 6);

12345 (start)

-5 (length)

---

length = 0, 1, 2, not -2

Trim → trim(' bar ') → bar

LPAD => ('hi', 4, '??') (length)

→ '??hi'

RPAD => Right side padding

Date functions

sysDate(), Now() → gives current date, time

Dayofmonth(), month(), year(), hour()
minute(), second()

↳ extract different elements from time stamp

datediff → returns in days

dateadd → returns as time value

timediff → ↩

---

makedate(year, dayofyear)

makedate(2011, 31) (2011, 32) → count from

(2011-01-31) 2011-02-01

Null functions

isnull()

ifnull()

nullif()

Group functions

1. sum()
2. avg()
3. max()
4. min
5. count

* null values are ignored by group function

---

limitation

↳ cannot be used with column

↳ not compatible with single row function

↳ cannot use in where clause

↳ nesting not allowed sum(sum).

Group by

* If column used for group by then it may or may not use in select clause
* If in select then compulsory in group by

Having clause

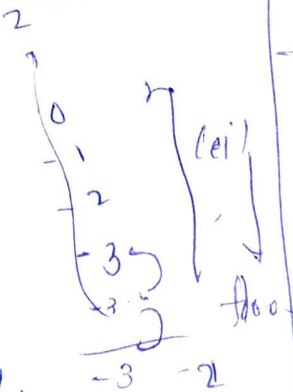
* Compulsory used with group by
* used to specify condition on aggregate values
* e.g. Select deptno, sum(sal) from emp group by deptno having ~~sal~~ sum(sal) > 9000

Where vs having day-4 ppt + 10

Where → on each record
having → each group.

Select → Where → Group by

power(2,5) = 32
Sqrt(2) = 1.24
Rand() =
Round(3.1425, 2)
3.14



Ceil → Ceil(3.14) = 4.
Floor(3.14) = 3
C(-3.14) = ~~-2~~ -3
F(-3.14) = ~~-3~~ -4

TimestampDiff → returns year

If → if(condition, true o/p, false o/p)

List functions

Concat('A', 'B', 'C') = ABC
greatest(29, 98, 53, 67) = 98
least(29, 24, 12, 01) = 01

Coalesce → (null, null, 12, 13) ⇒ 12
↳ returns first non null value

Ifnull() ifNull(comm, sal)
nullif(sal, 3000);

Group function similarly

select enam, sum(sal) from emp ✗
lower('sun'), sum(sal) ✗

where sal > max(sal) ✗
sum(max(sal)) ✗

Group by → by default works on all rows

select deptno, sum(), max()
from emp Group by deptno;

optional but makes no sense if not present

Having clause.

select job, avg(sal) from emp
Group by job
having avg(sal) > 1000.

Select → Group → order → limit

Joins (Cross)

```
for(emp e : emp){
   for(dept d : dept){
      sysо(e.ename, d.dname)
   }
}
```

Select e.ename, d.dname from
emp e cross join dept d

Inner join

```
For(Emp e : emp){
   for(dept d : dept){
      if(e.deptno == d.deptno)
         syso(e.enam, d.dname)
      }
}
```

select e.ename, d.dname from
emps e inner join dept
d on e.deptno = d.deptno;

Left Outer join

```
for(dept d: dept){
    found = false;
    for(emp e : emp){
        if(e.deptno == d.deptno){
            syso(e.ename, d.dname)
            found = True
        }
    }
    if(found == false)
        print(null, d.dname)
}
```

```
select e.ename, d.dname, from deptd
left outer join emp e on
    d.deptno = e.empno;
```

Optional =

Set operator:

union or union all

⊗ no of columns same

Union → common data is not repeated

union all → common data is repeated

```
select binary 'sunbeam' = Binary 'subAM'
```

Case insensitive comparision

⊗ select e.ename, d.dname from emps e inner join depts d on e.deptno != d.deptno

---

Outer join

```
select e.ename, d.dname from emp e
Right outer join dept d on
    e.deptno = d.deptno
```

Set operator

```
(select dname as name from dept)
union all
(select ename from emp)
```

inner join
left outer
Right outer

---

select → from → inner join → left / right → join → join → Group by → order by →

⊗ select column from table1
xxx join table2 on condition
xxx join table3 on condith
where
Group by
Order by
limit

Sub-Query

multi-row subquery

↳ it returns multiple rows

↳ IN, Any, All

↳ in operator compare for equality with results from sub-query (atleast one should match) = or

↳ ANY operator compares with the results from sub-query = or

↳ all result should match = AND

Correlated sub-query

↳ uses in, any, all, Exists operator
↳ it is done by adding criteria

Exists → inner query return 1 or more rows

Not Exists → inner query returns 0 rows

* Limitation

subquery not supported in all RDBMS in Update & delete

⊗ subquery should not select data from same outer table

Views :-

⊗ is based on select statement
⊗ it is used to show some columns from various tables
⊗ view is not stored in harddisk rather query is stored

Simple View
↳ DML operations are not supported on views for complex

Complex View
↳ group by, subquery joins, outerjoins.

→ DML on View affect under lying table

⊗ View can be created with Check option to ensure that DML operation can be performed on views only

Operation

show all tables
↳ To drop view
↳ drop view statement
↳ view can based on other view

@ used to create variable

All

select * from emp where sal > ALL (select sal from emp where job = 'salesman')

⊗ inner query with where clause to reduce no of rows produced is called Correlated sub-query.

e.g select * from dept d where d.deptno IN (select e.deptno from emp e where e.deptno = d.deptno)

Or Exists

select * from dept d where Exists (select e.deptno from emp e where e.deptno = d.deptno)

Inline View

the inner subquery in from clause where we create a derived table

```
select category, Count(empno)
From
(select empno, ename, sal
  Case
  when sal < 1500 then poor
  when      >          rich
  when
  Else 'rich'
  End
  As Category from emp)
  As emp_category
  Group by category
```

* subquery can be applied to DML
* if subquery performing operation on table then DML are not allowed

```
Delete from emp where sal =
(select max(sal) from emp).
```

View Syntax

```
Create view viewname As
 select .....
Create view v_empCategory As
select empno, ename, sal
 Case
 {
 End As category
 From emp;
```

* show tables
* show full tables.

Transactions

(2,0,0)

Transaction is set of DML queries executed as single unit

* transaction has acid properties
* default value of autocommit = 1
  => here all dml queries are autocommited.

Data Control language

Create alter drop rename -

Show grants → to get list of grants alloted to user

Show grants for dunbram21@host

Drop user

Drop user 'user@host'

Change password

Alter user user@host identified by 'new_password';

→ flush priviliges.

Grant permission to user

db = database name

* = every thing in db

```
Grant create on db.* To 'user@host'
Grant Creat on *.* To 'user@host'
Grant select on db.xtable To 'user@host';
Grant all on db.* To user@host
```

check optn of view → helps us insert data as per view

Revoke

→ To remove permission

Revoke all priviliges on classwork.* From (sunbeam21@localhost)

Grant insert, update, select on classwork.dept To 'sunbeam@lh'

Transactn good to go on DML queries only

DDL are not allowed

Row locking

auto locking row and releasing after transactn → optimistic locking

→ doing it manually → pessimistic locking

For update → to select query

Index

- create index idxName on table(column)
- indexes are stored as 'BTree' or 'Hash'

List index

show indexes from table

Drop index

drop index idx_name on table;

Constraint

not null, unique, pk, Fk, check

Save point

Start Tx

insert; update;

savepoint save1;

i/u/d

savepoint save2

Rollback to save1; &n

Select * from accounts where id = 2 for update;

Index Example

select * from books where subject is = 'c programming';

-- cost = 1.5

create index subject_i on books (subject)

-- cost = 0.9.

Show indexes from books.

(x) create unique index idx on

clustured index

- is auto created on primary key.
- if primary key not present some synthetic column is created and index applied on it

Drop

drop index idx on bookT

Not null email varchar(30) not null; → column level

Unique → null allowed, unique (email) → Table level

Pk → primary key (std, roll)

Fk ci int

...

Check foreign key (ci) reference temp(id)

parent