SQL can be divided into 5 parts:-

1) Data Query Language - (DQL)

SELECT : extracts Data from a DB.

2) Data Manipulation Language (DML)

INSERT : inserts new data into a DB.
DELETE : deletes data from a DB.
UPDATE : Updates data in a D.B.

3) Data Definition Language (DDL)

CREATE : Creates a new table
ALTER : modifies a table
DROP : deletes a table
TRUNCATE : truncate a table
RENAME : renames a table

4) Transaction Control Language (TCL)

COMMIT : Saves data permanently
ROLLBACK : Rollback the previous
SAVEPOINT : To create a save point

5) Data Control Language (DCL)

GRANT : To grant the privileges
REVOKE : To revoke the privileges.

Query Execution Order :

⇓

helps in writing complex queries

From
ON
JOIN
Where
Group By
With Cube or With RollUP
Having
Select
Distinct
Order By
↓ TOP, Limit, RowNum.

Subject → not a case-sensitive language
but generally we write keywords in All caps.

SQL - Structured Query Language : used to communicate with DB

Date

SQL Can do:

→ insert, delete, update data (records / rows) in DB (table)

→ retrieve Data from DB (tables)

→ Create new DB

" new tables in DB.

" Stored procedure in D.B.

views in DBB

→ Set permissions on tables, procedures & views.

Rel^nal D.B: → contains tables (objects)
(RDBMS)

↳ contains rows & columns

rows → contain records or data for cl

columns → contain cl Name, Datatype & any other attributes for the column

Select * from customers → To select all column values

select city from customers → will return duplicate values as well

select distinct city from Customers → To avoid Duplicates.

Select * from customers where city = 'Mumbai'

↳ Text values : single quotes (few DB Systems also accept double quotes)

Numeric values: No quotes

Operators used with where clause: & having

(cl → op → value) ↳ ' ' ( )

=

!= (or <>)

>

<

>=

<=

Between → (b/w inclusive range)

Like → Search for a pattern

IN → if u know the exact value you want to return for atleast one of the columns.
→ Allows to specify multiple values in where clause
IN is short cut for multiple OR cond^n

AND

OR

AND (with OR ...)

op^r { Like → Pattern → Not Like
In → Exact, → Not In
Between → Range → Not Bet^n

Union → Combine Select resultset → Union All

{ IS NULL
IS NOT NULL

Union

Primary Key

Customers Tabl

C_ID	F.Name	L.Name	Address	City
1	Deepak	Verma	ABC	Mumbai
2	Amit	Kumar	CBA	Lucknow
3	Suyash	Mittal	BCA	Delhi

Orders Tabl

O_ID	Order_No	C_ID	Amount
		3	1000
1	12345	3	2000
2	67895	1	4000
3	67321	1	500
4	98765		
5	43211	15	2500

Insert : To insert a new row in a table

insert into customers values (4, 'Charan', 'Singh', 'BCA', 'Hyderabad')

To insert data in specified colums only:

Insert into Customers (C_ID, F_Name, L_Name) values (4, 'charan', 'Singh')

Update : To update existing records in a table
row

Update Customers set Address = 'CCC', City = 'Avadh' where
~~L~~ L_Name = 'Kumar' AND F_Name = 'Amit'

SQL Update warning:
if omitted where clause
All records will be updated

Delete To delete rows in a table

Delete from customers where F_Name = 'Amit'

SQL Delete warng
if omitted, All rows of a table be deleted
But Table Structure, Attributes & indexes will be intact

✓ Delete from customers
or
✓ Delete * from customers

→ used for column of Numeric Data

Subject Aggregate F^ns: (Use of GroupBy & having clause)

Date

1. AVG() → returns Average value of a numeric column → values
2. Count() → No. of rows
   - → count (col): Total No. of ~~rows~~ values for the given col. (even duplicates)
   - → count (distinct col): No duplicates
   - → count (*): total no. of rows in a table.
   - → only agg f^n that ignores null value
3. Sum () → Sum
4. Max () → largest value
5. Min() → smallest value
6. First()
7. Last()

Orders Tab2

O_Id	orderPrice	Customer
1	1000	Deepak
2	1600	Nilesh
3	700	Deepak
4	300	Deepak
5	2000	Amit
6	100	Nilesh

1) AVG()

Select AVG(OrderPrice) as OrderAverage from Orders

resultset will look like this

Order Average
950

→ To find cust^rs that hv an orderprice value higher than the AVG orderprice value

Select customers from order where Order Price > (Select AVG(OrderPrice) from Orders)

Customer
Deepak
Nilesh
Amit

2) Count()

- count (col_name)
- Count (distinct col_name)
- Count (*) ⇒ Total rows.

→ NULL value is Not Counted
→ Duplicate is counted

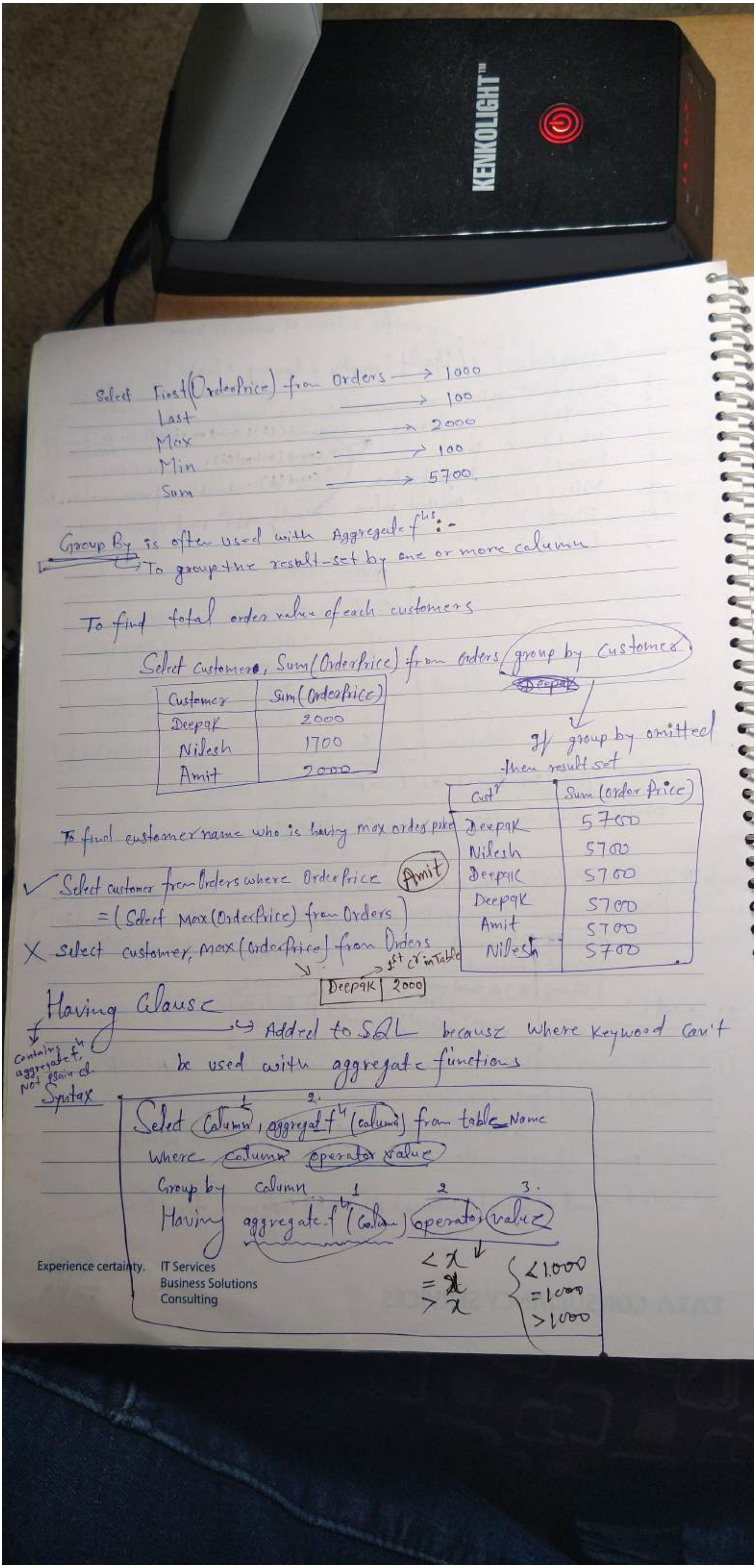
(A) Select COUNT(Customer) from Orders where customer = 'Nilesh' → (2)

(B) Select count (*) as No. of orders from Orders → (6)

↳ returns the No. of rows in a Table if where is omitted in the tab.

To find No. of Unique customers from Orders → (3)

(C) Select count (Distinct Customer) As No. of customers from Orders



Select First(OrderPrice) from Orders → 1000
Last → 100
Max → 2000
Min → 100
Sum → 5700.

Group By is often used with Aggregate f^{ns}:-
→ To group the result-set by one or more column

To find total order value of each customers

Select customer, Sum(OrderPrice) from orders group by customer

Customer	Sum(OrderPrice)
Deepak	2000
Nilesh	1700
Amit	2000

~~Deepak~~

↓ If group by omitted then result set

Cust^r	Sum (order Price)
Deepak	5700
Nilesh	5700
Deepak	5700
Deepak	5700
Amit	5700
Nilesh	5700

To find customer name who is having max order price

✓ Select customer from Orders where OrderPrice = (Select Max(OrderPrice) from Orders) Amit

X select customer, max(OrderPrice) from Orders → 1st c^r in Table

Deepak	2000

Having Clause

→ Added to SQL because where keyword can't be used with aggregate functions

↓ contains aggregate fⁿ, not plain cl.

Syntax

Select Column, aggregat fⁿ (column) from table_Name
where column operator value
Group by column
Having aggregate fⁿ (Column) operator value

< x
= x
> x

< 1000
= 1000
> 1000

Experience certainty. IT Services
Business Solutions
Consulting

Subject

when there is some cond^n on Total of a column use "having"

① To find C^rs who have a total order of less than 2000

→ Select Customer, Sum(OrderPrice) from Orders group by customer having Sum (OrderPrice) < 2000

where ✗

Date

Nilesh 1700

② To find if Deepak & Amit have a total order of more than 1500

Select customer, Sum(OrderPrice) from Orders
Where customer = 'Deepak' OR Customer = 'Amit'
Group by customer Having Sum(OrderPrice) > 1500

IN ('Deepak', 'Amit')

SQL Wild cards

→ Must be used with SQL 'Like' operator

	Wild Card	Description
①	% →	A substitute for zero or more characters
② Underscore	_ →	" for exactly one "
③ [ ]	[charlist] →	Any single char in the list
④ [!charlist] or [^charlist] [^]	→	" " " Not "

} works in (Microsoft Product) MS Access & SQL Server only.

To find cust^rs living in the city that starts ~~with~~ Mu

Select * from customertab1 where city like 'Mu%' starts with

city like '%mb%' contains

'%bai' ends with

City Like '_umb_i' starts with any char followed by umb followed by any char followed by i

Where FName Like '[DAS]%' → first Name starts with D or A or S

'[!DAS]%' → Not

IN

→ shortcut for multiple OR cond^ns.

Select * from customers where FName IN ('Deepak', 'Amit')

→ IN (Subquery)

Between

: selects a range of data b/w two values → (Numbers, text, dates)

Select column_name(s) from tab_Name
where col_Name Between value1 AND value2 → values can be numbers, text, or dates.

NOT Between

→ 10 AND 20

→ 'Varma' AND 'Kumar'

→ #07/04/1996# AND #07/09/1996#

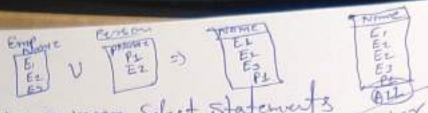
c^r with L

eg. L_name alphabetically between Kumar AND Varma

③ result coz inclusive

TATA

TATA CONSULTANCY SERVICES



Union :- → Selects distinct values by default
→ used to combine resultset of two or more Select statements

Select column_Name(s) from Tab1
~~Select~~ Union
Select column_Name(s) from Tab2

No, type, order

Note: Each select statement within the Union must hv same No. of columns
Columns must hv similar data types
Columns in each select statᵗ must be in the same order

↳ Column Names in the resultset of a Union are always equal to column Names in the first Select Statement in the Union

table1 AS ~~[illegible]~~ A ✓
table A ✓

Union ALL: Allows duplicate values
→ Generally used with aggregate fⁿˢ.

SQL Alias :- usually
↳ Used to give a shorter name to a table or a column in a table.
Although it can longer as well or anything.
↳ should not have spaces in it. Avg Price ✗ => Avg_Price ✓ or AvgPrice ✓
[Avg Price] ✓ "Avg Price" ✓
It requires double quotⁿ marks or square brackets if alias name contains spaces.

Note: Select * => is a quick way of selecting all columns.
asterisk

Boolean OP in Where Clause
AND → both condⁿˢ should be true to display a record
OR → any one of the two should be true.
Not → display a record if condⁿ is not true

combining AND & OR:

where Not L_name = 'varma'

Where L_name = 'varma' AND (F_name = 'Deepak' OR F_name = 'Raj')

Sorting:
"Order" ✓ ordered ✗

OrderBy: This Keyword is used to sort the result set by a specified column.
↳ sorts the record in ascending order by default.

ASC | DESC → Keywords
ASC ↓ No need to use.
DESC ↓ To sort record in descending order.

Select * from customers
order by L_name Desc

CId	F Name	L_name	Address	City
—	—	Varma	—	—
data	-	Mittal	-	-
—	—	Kumar	—	—

Sorting can be done based on one or more cls.
→ Order by name ASC, Salary DSC;
↳ If name is same then by salary

Subject **SQL Joins** : Used to retrieve data from ~~Data~~ two or more tables based on rel^n ship b/w certain columns in these tables

→ used to combine rows from two or more tables based on common field b/w them.

Types

1) Join ~~or~~ Inner Join: return all rows when there is atleast 1 match in both tables
or simple Join
or Equi Join

→ Return all rows from multiple tables where the join cond^n is met.

2) Left (outer) Join :- Return all rows from Left table & matching rows from the right Table (even if there r no matches in right table)

3) Right (outer) Join :- Reverse

4) Full (outer) Join :- Return all rows when there is a match in one of the tables (when there is a match in either left table or right tab records)

Union of L & R Joins

Left Join ↑ space

```
Select C.F_Name, C.L_Name, O.Order_No from
Customers as C Inner Join Orders as O
ON C.C_Id = O.~~R~~_Id
Order By C.F_Name
```

F_Name	L.Name	Order No.
Deepak	Varma	67321
~~Suyash~~ Deepak	Varma	98765
Suyesh	Mittal	12345
Suyash	Mittal	67895

Inner join is subset of left or Right or Full Join.

" Left Join "

F_Name	Lname	OrderNO
Amit	Kumar	
Deepak	Varma	67321
Deepak	Varma	98765
Suyash	Mittal	12345
Suyash	Mittal	67895

" Right Join

F_Nam	Lname	Order No
Deepak	Varma	67321
Deepak	Varma	98765
Suyash	Mittal	12345
Suyash	Mittal	67895
		43211

Blank → NULL

" Full Join " ⇒

F_Name	L_Name	Order No
Amit	Kumar	
Deepak	Varma	67321
Deepak	Varma	98765
Suyash	Mittal	12345
Suyash	Mittal	67895
		43211

→ select * from customers order By country ASC, customername DESC;

Cross Join → from customers, orders where cr. C.Id = or. c.Id

Inner Join → from customers Inner Join Orders on cr. c.Id = or. c.Id

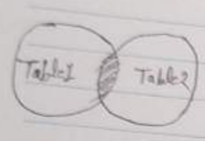
Subject

Date

Inner Join e.g.

one more syntax:

```
Select  Customers.Fname, customers.L_name, orders.Order_No
From    customers, Orders
Where   customers.C_Id = Orders.C_Id
           or
Select  Fname, L_name, Order_NO
From    customers, Orders
Where   customers.C_Id = Orders.C_Id
```



⇒ preceding columns names with the table name & a period is Not alwayz required. It is required if the column names are same b/w the two tables.

However it is good practice to precced with table name & a period to avoid confusion which columns is from which table.

ANSI SQL-92 Syntax (W3 School)

✓

```
Select  customers.F_name, customers.L_name, Orders.Order_No
From    customers Inner Join Orders
ON      customers.C_Id = Orders.C_Id.
```

Join (only Join will also work -W3S)

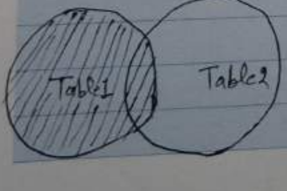
Left Join

return all rows from left table, with matching rows in the right table.

* The result is NULL in the right side where there is no match.
↘ value may be displayed as "Null" or it may be displayed as blank

```
Select column name(s)
From Table1 Left Join table2
ON table1.column name = table2.column name.
Order By tab1.ClName;
```

Left Join → Left Outer Join
↓
In some D.B. LJ is also called LOJ.
& RJ → ROJ

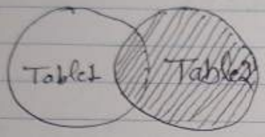


rsarda7@gmail
4715029772 w
9860018423

Right Join

Reverse

Result is null in the left side when there is no match.



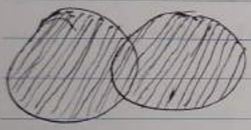
RJ → ROJ

Full Outer Join:

return all rows from Left table & from the right table.

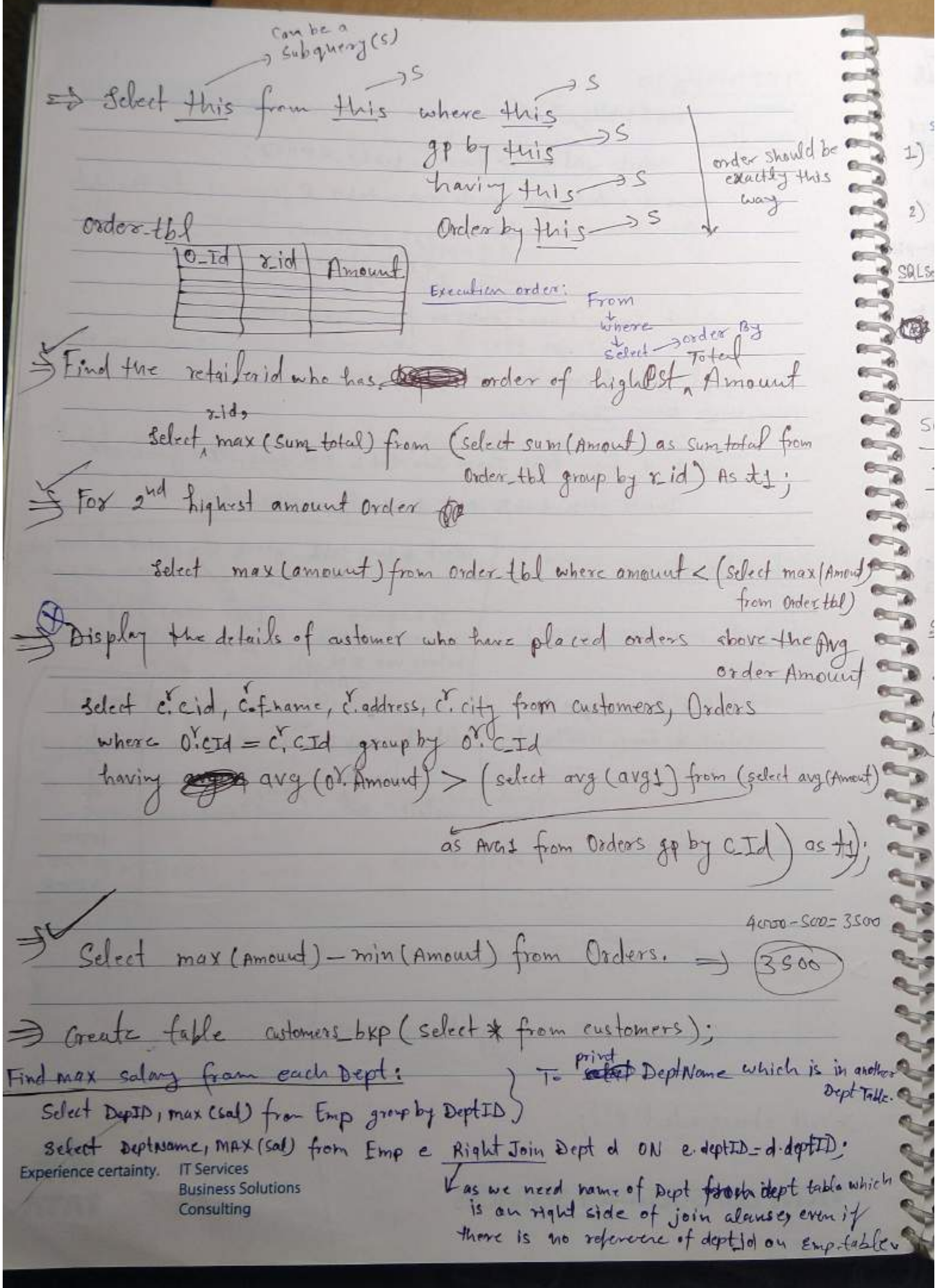
↳ combines the result of both Left & Right Joins

FJ → FOJ



```
Select column_name(s)
 From table1 Full Outer Join table2
ON table1.column_name = table2.column_name
```

⇒ SQL Keyword are generally written in Capital Letters.



can be a subquery(s)

⇒ Select this from this where this
gp by this
having this
Order by this

→ s

order should be exactly this way

order_tbl

O_Id	r_id	Amount

Execution order: From ↓ where ↓ select → order By

⇒ Find the retailer id who has order of highest Total Amount

```
Select r_id, max(Sum_total) from (Select sum(Amount) as Sum_total from
                                  Order_tbl group by r_id) As t1;
```

⇒ For 2nd highest amount Order

```
Select max(amount) from order_tbl where amount < (Select max(Amount)
                                                  from Order_tbl)
```

⇒ Display the details of customer who have placed orders above the Avg order Amount

```
Select cr.cid, cr.f_name, cr.address, cr.city from customers, Orders
 where or.CId = cr.CId group by or.C_Id
 having avg(or.Amount) > (select avg(avg1) from (select avg(Amount)
                         as Avg1 from Orders gp by C_Id) as t1);
```

4000-500=3500

⇒ Select max(Amount) - min(Amount) from Orders. ⇒ 3500

⇒ Create table customers_bkp (select * from customers);

Find max salary from each Dept:

```
Select DepID, max(sal) from Emp group by DeptID
```

To ~~select~~ print DeptName which is in another Dept Table.

```
Select DeptName, MAX(Sal) from Emp e Right Join Dept d ON e.deptID=d.deptID;
```

as we need name of Dept ~~from the~~ dept table which is on right side of join clauses even if there is no reference of dept_Id on Emp table.

Experience certainty. IT Services
Business Solutions
Consulting

Subject 2^{nd} highest salary, for 3^{rd} highest salary another subquery can be put.

Date

1) Select max(sal) from Employee where sal < (Select max(sal) from Employee)
   where sal NOT IN (Select max(sal) from Employee)
   where sal < ...

2) find 5 emp names from table who have highest salary.

SQLServer: Select TOP 5 * from Employee order by salary DESC

3) find all emp with salary in decreasing order
   select * from Employe order by salary DESC.

SQL TOP, LIMIT or ROWNUM clause

select TOP number | x percent columns

→ Are used to specify no. of records to return.

TOP → used to fetch a top n number or x percent records from a table.

→ all D.B. do not supports TOP clause.
→ supported SQL server & MS Access.

→ useful on large tables with thousands of records
Returning a large no. of records can impact on performance.

~~MySQL~~ → LIMIT → MySQL

~~Oracle~~ → ROWNUM → Oracle

RO — keyword not fⁿ — or

SQLS^r/MS Access

select Top 3 * from customer; → where country = 'Germany'

| select Top 50 percent * from customer

MySQL select * from customer limit 3; → where country = 'Germany' LIMIT 3; → last student

OracleS^r. select * from customer where rownum <= 3;
→ where country = 'Germany' AND rowNUM <= 3;

AND, OR: Conjunctive operators

→ Allows existence of multiple conditions in where clause
→ n-number of condⁿ can be combined

select name, salary from Emp
where salary > 2000 AND age < 25;
OR

→ where salary Like '200%';
' ' single quote ⇓ for integer as well.

nth highest Salary:

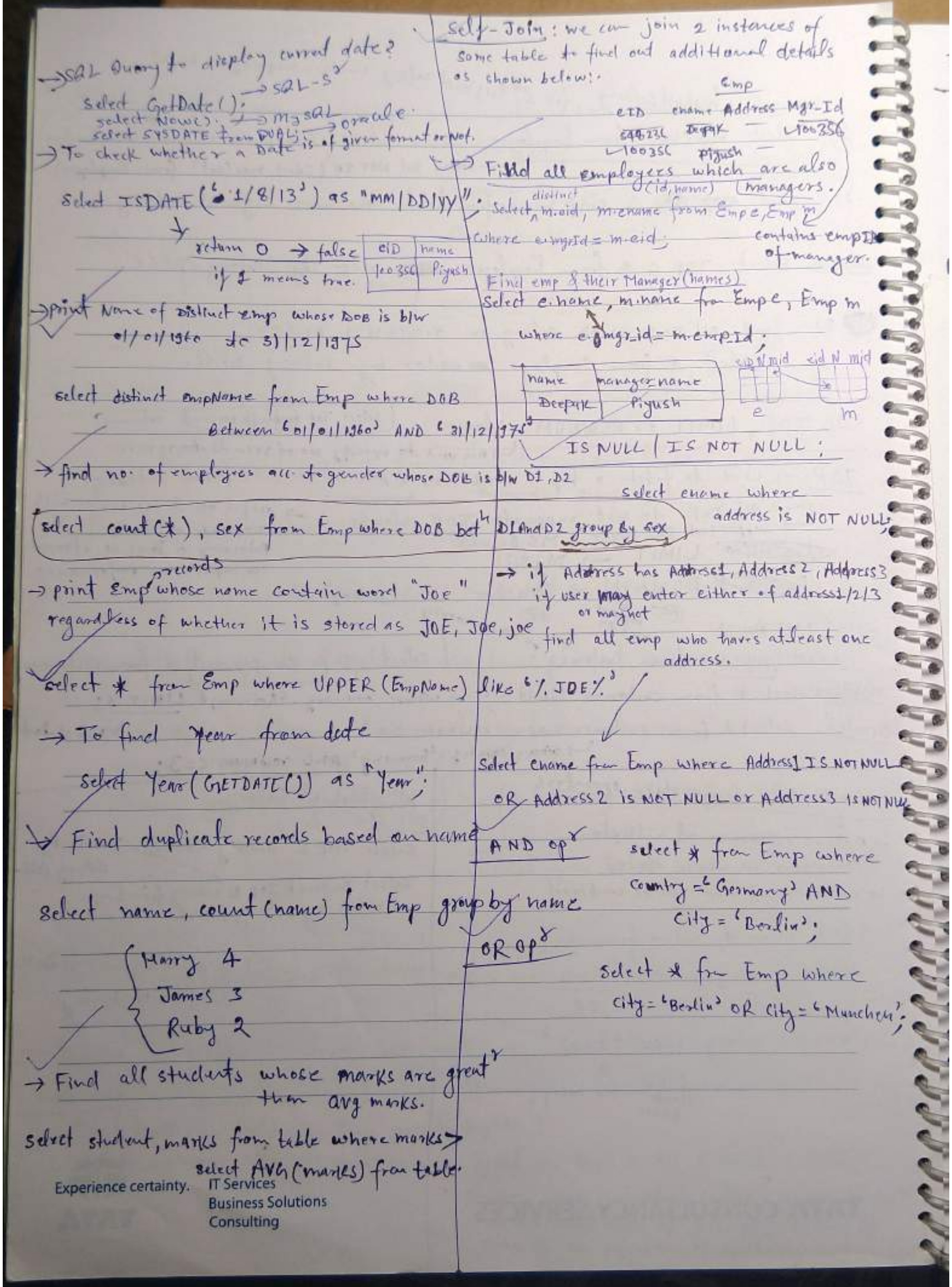
SQL S^r–

Select TOP 1 Salary from( ) order By Salary ASC

select Distinct Top N salary from Emp order by salary DESC

MySQL

select salary from Emp order by salary Desc limit N-1, 1;



→ SQL Query to display current date?

```
select GetDate(); → SQL-S^r
select Now(); → mySQL
select SYSDATE from DUAL; → oracle.
```

→ To check whether a Date is of given format or not.

```
Select ISDATE('1/8/13') as "mm/DD/yy";
```

return 0 → false
if 1 means true.

eID	name
100356	Piyush

→ print Name of Distinct emp whose DOB is b/w 01/01/1960 to 31/12/1975

```
select distinct empName from Emp where DOB
        Between '01/01/1960' AND '31/12/1975'
```

→ find no. of employees acc. to gender whose DOB is b/w D1, D2

```
select count(*), sex from Emp where DOB betn D1 and D2 group By sex
```

→ print Emp records whose name contain word "Joe" regardless of whether it is stored as JOE, Joe, joe

```
select * from Emp where UPPER(EmpName) like '%JOE%'
```

→ To find Year from date

```
select Year(GETDATE()) as "Year";
```

→ Find duplicate records based on name

```
select name, count(name) from Emp group by name
```

Harry	4
James	3
Ruby	2

→ Find all students whose marks are greater than avg marks.

```
select student, marks from table where marks >
        select AVG(marks) from table.
```

Self-Join : we can join 2 instances of same table to find out additional details as shown below:-

Emp

eID	ename	Address	Mgr_Id
544236	Deepak	—	100356
100356	Piyush	—	

Find all employees which are also managers. (id, name)

```
Select distinct m.eid, m.ename from Emp e, Emp m
where e.mgrId = m.eid;
```

contains empID of manager.

Find emp & their Manager (names)

```
Select e.name, m.name from Emp e, Emp m
    where e.mgr_id = m.empId;
```

name	manager_name
Deepak	Piyush

eid N mid   eid N mid
e   m

IS NULL | IS NOT NULL;

```
select ename where
    address is NOT NULL;
```

→ if Address has Address1, Address2, Address3 if user may enter either of address1/2/3 or may not find all emp who have atleast one address.

```
Select ename from Emp where Address1 IS NOT NULL
 OR Address2 is NOT NULL or Address3 IS NOT NULL
```

AND opr

```
select * from Emp where
    Country = 'Germany' AND
        City = 'Berlin';
```

OR opr

```
select * from Emp where
  City = 'Berlin' OR City = 'Munchen';
```

SQL (si:kwəl)

SQL "Exists" operator. used to test for the existence of any record in a subquery

→ returns true if the subquery returns one or more records.

e.g.

```
select supplierName from suppliers where Exists (select productName from Products
    where products.suppid = suppliers.suppid AND Price < 20);
```

⇒ list of the suppliers with a product price less than 20.

→ IN :→ is used when we have multiple values in where cond^ns. (<>, =, <=, >=, !=)

→ It's always testing for equality. But to use other operators like >, < etc we use ANY or All.

↓ Alternative (= Any)

Not IN ⇒ <> All.
Alternative

>All (1,2,3) ⇒ means greater than 3 (max value)

>Any (1,2,3) ⇒ greater than 1 (min value) (greater at least one value)

ANY & ALL

used with Where or Having clause.

↓ returns true if any of subquery values meet the cond^n

↓ All

```
select ProductName from Products where pid = Any (select pid from P
                                            ALL      where qty = 10)
```

Select INTO

copy data from one table to a new table.

```
→ select *     INTO newTab from old table. where cond^n
   col1, col2
```

→ To create a new empty table using schema of other table.

```
select * into new_tab from old_tab where 1 = 0;
```

Case Statement

:→ switch case

```
select orderId, Quantity,
CASE
    When Quantity > 30 then " Then qty is greater than 30"
    When quantity = 30 then " The quantity is 30"
    Else " Qty is under 30"
END as QtyText
from Orders;
```

o/p:

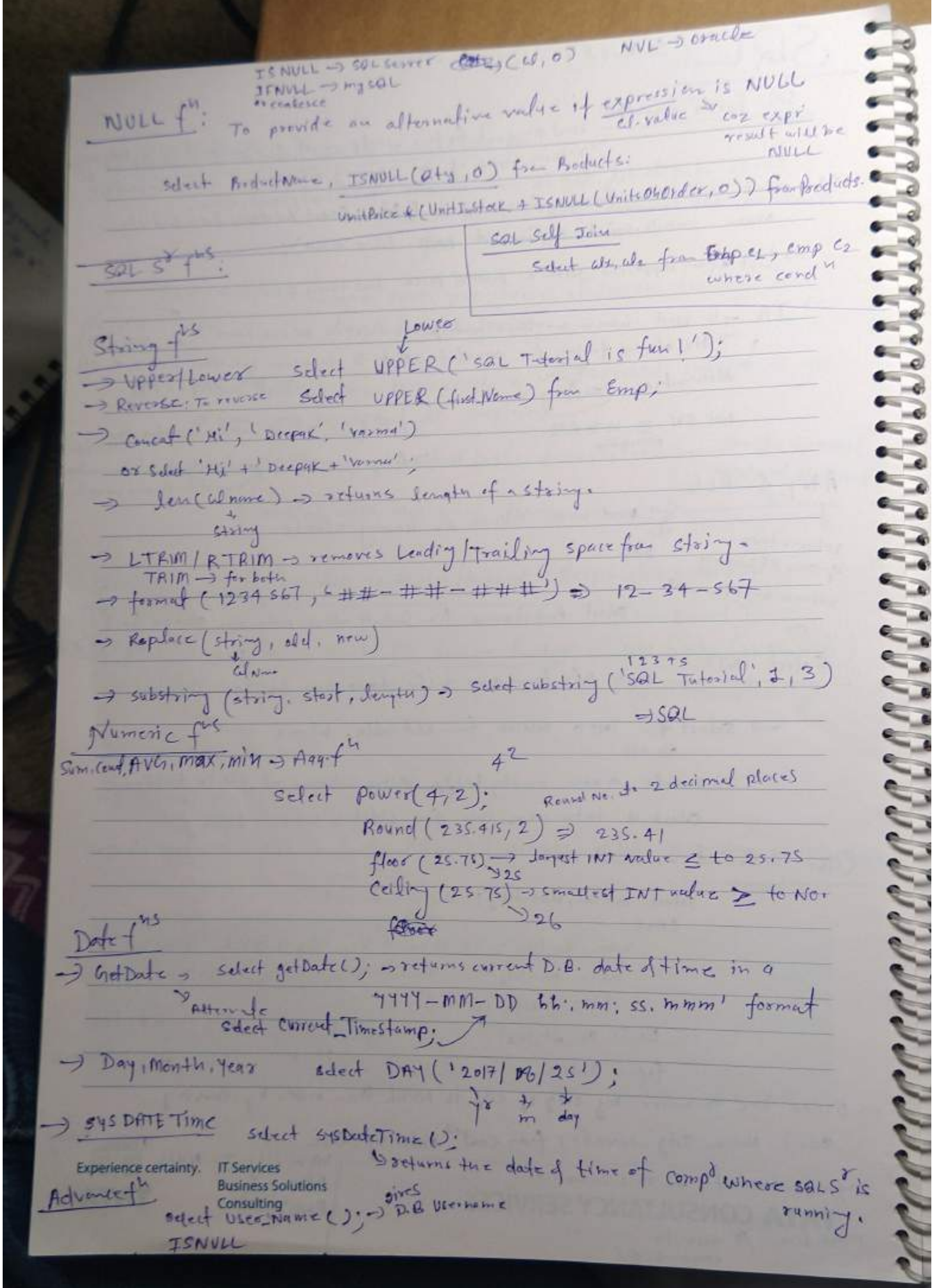
orderId	Qty	Qty Text
1021	12	Qty is under 30

→ Order the customer by city if city is NULL then order by country.

```
Select Name, City, Country from custrs order By (Case
    when city IS NULL Then country
    Else City
END);
```

single line comment `-- This is comment`

multiline `/* This is comment */`



ISNULL → SQL Server ~~[illegible]~~ (cl, 0)   NVL → oracle
IFNULL → mySQL
or coalesce

NULL f^n: To provide an alternative value if expression is NULL
el. value ↘ coz expr result will be NULL

select ProductName, ISNULL(Qty, 0) from Products;
UnitPrice * (UnitInStock + ISNULL(UnitsOnOrder, 0)) from Products.

SQL Self Join
Select cl1, cl2 from ~~Emp~~ e1, emp e2
where cond^n

~~SQL S^r f^{ns}:~~

String f^{ns}

→ Upper/Lower   select UPPER('SQL Tutorial is fun!');   (UPPER ↓ Lower)
→ Reverse: To reverse   Select UPPER(first_Name) from Emp;
→ Concat('Hi', 'Deepak', 'varma')
or Select 'Hi' + 'Deepak + 'varma';
→ len(cl name) → returns length of a string.
↓ string
→ LTRIM / RTRIM → removes Leading / Trailing space from string.
TRIM → for both
→ format(1234567, '##-##-###') ⇒ 12-34-567
→ Replace(string, old, new)
↓ cl Name
→ substring(string, start, length) → select substring('SQL Tutorial', 1, 3)
(1 2 3 4 5)
⇒ SQL

Numeric f^{ns}

Sum, Count, AVG, max, min → Agg f^n

4^2
select power(4, 2);
Round No. to 2 decimal places
Round(235.415, 2) ⇒ 235.41
floor(25.75) → largest INT value ≤ to 25.75
↘ 25
Ceiling(25.75) → smallest INT value ≥ to No.
~~floor~~ ↘ 26

Date f^{ns}

→ GetDate → select getDate(); → returns current D.B. date & time in a
'YYYY-MM-DD hh:mm:ss.mmm' format
↘ Alternate
select Current_Timestamp;

→ Day, Month, Year   select DAY('2017/~~06~~/25');
↓ yr ↓ m ↓ day

→ SYS DATE Time   select sysDateTime();
↳ returns the date & time of comp^d where SQL S^r is running.

Advance f^n

select User_Name(); → gives D.B Username
ISNULL

DDL

Subject Create Table

Date

```
Create Table   table_name
(
    Column_name1 datatype,
     C-name2 data type,
     C-name3 datatype,
     - -- -.
)
```

```
Create TABLE   customers
(
    C-Id  int,
    Fname varchar(255)
    L-name varchar (255)
     Address varchar (255)
      City   varchar (255)
)    DOB     date
```

max length: 255 characters.

ALTER Table

To add a cl

```
ALTER TABLE table_name
ADD column_name data type
```

To delete a cl (Some D.B. don't allow deleting a column)

```
Alter Table  table_name
DROP Column  column_name
```

To change datatype of a cl

```
Alter Table  table_none
Alter Column  column_name data type.
```

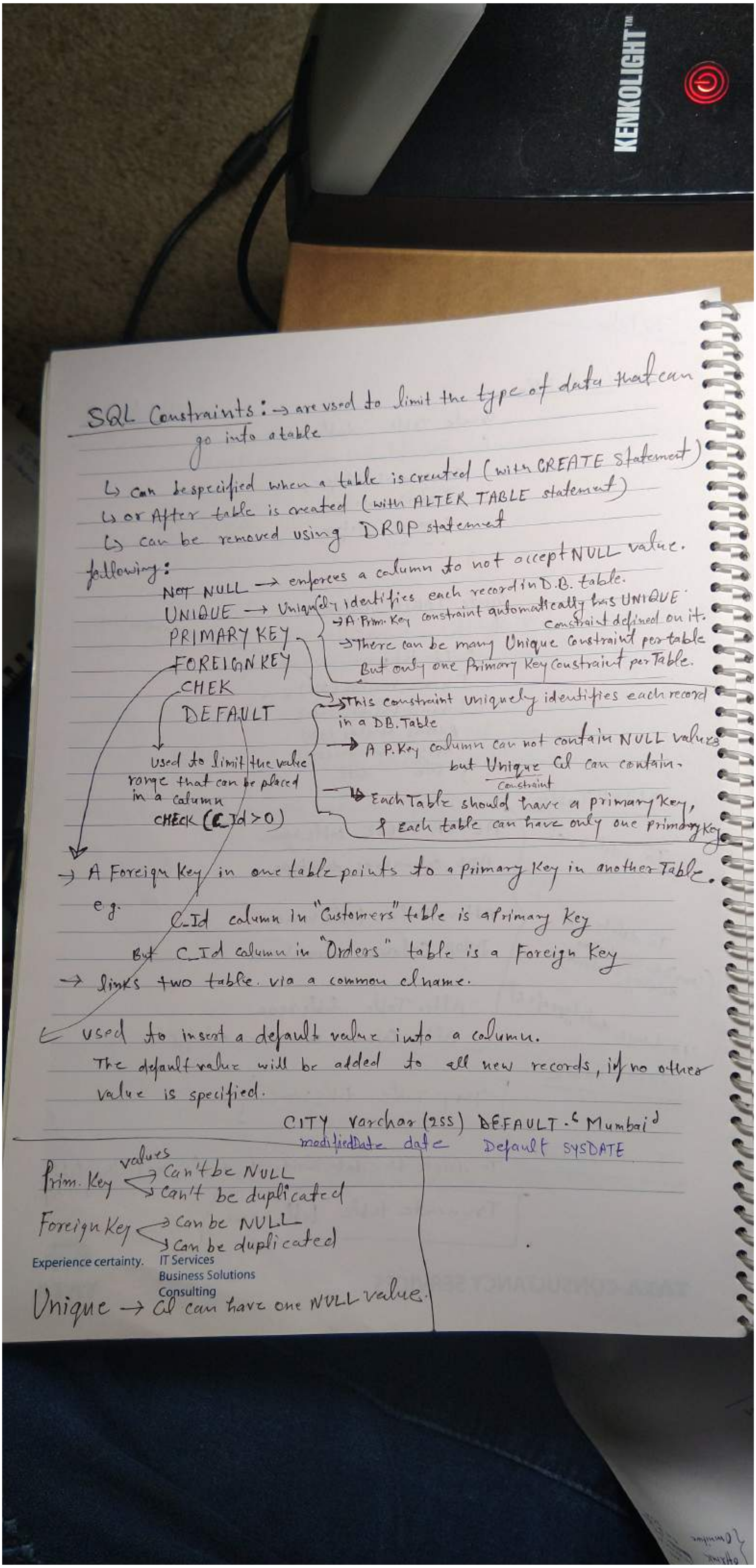
Drop Table

```
Drop Table  table_name
```

Truncate Table

To delete the data inside the table, not the table itself.

```
Truncate table  table_name
```


KENKOLIGHT™

SQL Constraints: → are used to limit the type of data that can go into a table

↳ can be specified when a table is created (with CREATE Statement)

↳ or After table is created (with ALTER TABLE statement)

↳ can be removed using DROP statement

following:

NOT NULL → enforces a column to not accept NULL value.

UNIQUE → Uniquely identifies each record in D.B. table.

→ A Prim. Key constraint automatically has UNIQUE constraint defined on it.

→ There can be many Unique Constraint per table But only one Primary Key Constraint per Table.

PRIMARY KEY

→ This constraint uniquely identifies each record in a DB. Table

→ A P.Key column can not contain NULL values but Unique Constraint Col can contain.

→ Each Table should have a primary key, & Each table can have only one primary Key

FOREIGN KEY

CHEK

DEFAULT

used to limit the value range that can be placed in a column

CHECK (C_Id > 0)

→ A Foreign Key in one table points to a primary Key in another Table.

e.g. C_Id column In "Customers" table is a primary Key

But C_Id column in "Orders" table is a Foreign Key

→ links two table. via a common cl name.

used to insert a default value into a column.

The default value will be added to all new records, if no other value is specified.

```
CITY varchar (255) DEFAULT 'Mumbai'
modifiedDate date Default SYSDATE
```

Prim. Key values → Can't be NULL
→ Can't be duplicated

Foreign Key → Can be NULL
→ Can be duplicated

Unique → Col can have one NULL value.

Defining constraint

Subject Primary Key

MySQL

```
Create Table Customers
(
    C_Id int NOT NULL,
    LastName varchar(255) NOT NULL,
     F_Nome varchar(255),
     Address varchar(255),
     city varchar(255),
      PRIMARY Key(C_Id)
)
```

Date

SQL Server | Oracle | MS Access

```
Create Table Customers
(
C_Id int NOT NULL PRIMARY KEY,
        "
        "
        "
   City varchar(255)
)
```

→ Primary Key can be a combin^n of two or more columns. → composite P. Key

Foreign Key

MySQL

```
Create Table Orders
(
  O_Id int NOT NULL,
  OrderNo int NOT NULL,
  C_Id int,
  PRIMARY Key (O_Id),
  Foreign Key (C_Id) REFERENCES Customers(C_Id)
)
```

SQL Server / Oracle / MS Access

```
CREATE TABLE orders
(
  O_Id int NOT NULL PRIMARY KEY,
  OrderNo int NOT NULL,
  C_Id int FOREIGN Key References
                        Customers(C_Id)
)
```

Normalisation of Table or D.B. or → Table. → It is done to remove redundancy of data in table or D.B. (By breaking Table) Normalized Table

If one person order 10 items then (Customer + Order) merged Table → as one Table in merged table F_name, L_name, Address, City would be entered unnecessarily 10 times

but when tables are separate we need to enter only C_Id 10 times in Order table.

View → In SQL, view is a virtual table based on resultset of an SQL statement.

SQL Date Format:

YYYY-MM-DD

1989 - 08 - 10

order_date = ' 1989-08-10'

⇒ It is not possible to test for NULL values with comparision op^r =, <, > etc.

H, we use

IS NULL, IS NOT NULL

Select F_name, L_name from C^r where address IS NULL

Data Type

Microsoft Access.

Text → 255 char

Integer → -32768 to 32767

mySQL

Char (size) → holds a fixed length string, max 255

fixed size

e.g. char (26)

All string of 26 char only.

Varchar (size) → holds a variable length string, max 255 char

max size

INT(size) → max. no. of digits needs to be specified.

TCS Training :=>

Subject

Cross Join

b/w 2 tables

Date

```
Select <col name> from <tab1>, <tab2>;
```

Takes the data from each row in tab1 & joins it to the data from each row in tab2

3X3 = 9 records

4X3 = 12 records.

```
select Fname, Lname, order_No from Customers, Orders;
```

3 X 5 = 15 records

=> In mysql, join & cross-join are synonymous

More then one Join

```
Select * from tab1 Inner Join tab2 ON tab1.colx = tab2.colx
    Inner join tab3 ON tab2.coly = tab3.coly
```

Store Procedure

```
> Create procedure proc()
  begin
    select * from login;
    select * from retailer_info;
  end

  > call proc()
```

o/p:

```
> Create procedure showgrade (grade varchar (30))
  begin
  Select * from debuggers where grading = grade;
  end

> call showgrade ("A");
```

```
select * from tab1 where col = (select col from tab2
                                  where col = value)
```

If subquery returns multiple values use IN or = Any

IN (NOT IN)

= ANY

```
Select emp_Id from Emp
        Union
Select sal from sal
```

Emp_Id
1
2
3
4
5
1000
2000
3000