SQL CONCEPTS

(SUMMARY DOC)

INDEX

Concept/definition	Example	Syntax
	GENERAL COMMANDS	
DESC- • To see the attributes of a particular table.	Eg desc emp;	Desc tablename;
ED- • To modify an already existing data.		
SPOOL- • to keep a record of all da commands entered	eg.SPOOL c:\myfolder\session SPOOL OFF-writing complete	SPOOL (location to save file)
/- • Rexecution of last executed statement		
Alias column heading	eg.Select ename as employeename from emp;	
	OPERATORS	
ARITHMETIC OPERATORS		
➢ +:ADD ➢ - :SUBTRACT ➢ * :MULTIPLE ➢ / :DIVIDE ➢ () :OVERIDE PRECEDENCE		
Handling NULL- • done using NVL operator	eg. ◆ SELECT ENAME,SAL,COMM,SAL*12+NVL(COMM,0) FROM EMP;	
Concatenation operator (‖)	eg.	

	SELECT ENAME||' works as '||JOB "EMP JOB" FROM EMP;	
	eg.	
Comparison operators - ➢ = : equal to ➢ !=, <>, ^= : not equal to ➢ < : less than ➢ <= : less than or equal to ➢ > : greater than ➢ >= greater than or equal to		
Logical Operators - ➢ NOT : logical NOT operator ➢ AND : logical AND operator ➢ OR : logical OR operator		
SQL operators:- ➢ IN : list of values within ( ) ➢ NOT IN : negation of IN ➢ BETWEEN : range of values ➢ NOT BETWEEN : negation of BETWEEN ➢ LIKE : using “%” and “_” as meta characters		

➢ NOT LIKE : negation of LIKE ➢ IS NULL : evaluating NULL's ➢ IS NOT NULL : negation of IS NULL		
Substitution variables ➢ & : accepts values, expression during query execution		
Sorting – order by clause	◆ SELECT <col_names> FROM <table_name> ORDER BY <col_name/s>	
	DATA QUERY LANGUAGE	

SELECT-
• used to retrieve data present in database
 	
order of select
 	
SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY
	Eg to get details from database EMP.
	
	EG.select * from emp where deptno=10 order by ename;
		SELECT command syntax
		
		SELECT
		<column_list>
		
		FROM
		tables
		WHERE
		clause - - (for restricting column values
		)
		GROUP BY
		clause
		
		- - (grouping attributes for aggregates)
		
		HAVING
		- - (for restricting group results
		)
		
		ORDER BY
		- - (for sorting)

	Data Definition Language	
<u>CREATE</u>		
CREATE- • to create object in database	eg.CREATE TABLE dept(deptno NUMBER(2) CONSTRAINT dept_deptno_pk PRIMARY KEY, dname VARCHAR2(12), loc VARCHAR2(10), CONSTRAINT dept_dname_loc_unq UNIQUE (dname,loc));	CREATE TABLE [USER.]TABLE (col1 datatype[(size)] [column constraint], col2 datatype[(size)] [column constraint], ...

		. [Table constraint])
Constraints- ➢ check ➢ unique ➢ primary key ➢ foreign key ➢ composite primary key	eg. ➢ unique(U)-CONSTRAINT BRANCH_BRANCHNAME_UNQ UNIQUE(BRANCH_NAME), ➢ check(C)- CONSTRAINT BRANCH_BRANCHCOURSEDUR_CHK CHECK(BRANCH_COURSE_DURATION <= 4) ➢ primary key(P)-CONSTRAINT BRANCH_BRANCHID_PK PRIMARY KEY(BRANCH_ID), ➢ foreign key(R)-CONSTRAINT JOB_ID_STAFF_FK FOREIGN KEY(BRANCH_ID) REFERENCES JOBS ON DELETE SET NULL ➢ composite primary key-CONSTRAINT STAFF_SUBJECT_COMPPK PRIMARY KEY(STAFF_ID,SUBJECT_ID)	
Delete constraint- ➢ on Delete restrict-default.If child exists parent cant be deleted. ➢ on delete cascade-if parent is deleted child is automatically deleted. ➢ on delete set null -if parented deleted child is set to NULL.		

CREATING A TABLE FROM EXISTING TABLE-
 	
<u>ALTER</u>
	EG.CREATE TABLE EMP10 AS
	SELECT EMPNO,JOB SAL FROM EMP
	WHERE DEPTNO=10;	

ALTER- • used to alter the structure of a database.Following clauses supported with alter:- ➢ ADD ➢ MODIFY ➢ DROP ➢ RENAME ➢ DISABLE ➢ ENABLE		
ADD- ➢ used to add a column or a constraint to an existing table	ADD- ◆ column:ALTER TABLE emp10 ADD lastname varchar2(10); ◆ primary key:ALTER TABLE EMP10 ADD CONSTRAINT emp10_empno_pk PRIMARY KEY(empno); ◆ unique:ALTER TABLE EMP10 ADD CONSTRAINT emp10_ename_unq UNIQUE(ename);	
MODIFY- ➢ to change datatype or size ➢ to add not null constraint	eg. ◆ NOT NULL-ALTER TABLE emp10 MODIFY ename CONSTRAINT emp10_ename_nn NOT NULL; ◆ DataType/Size Change-ALTER TABLE emp10 MODIFY lastname char(20);	
RENAME – ➢ to rename column	eg. ◆ ALTER TABLE emp10 RENAME column lastname to lname;	

QUERY TO VIEW CONSTRAINT DETAILS	◆ SELECT TABLE_NAME,CONSTRAINT_NAME,CONSTRAINT_TYPE,STATUS
	
	FROM USER_CONSTRAINTS
	
	ORDER BY TABLE_NAME;	

DISABLE/ENABLE- ➢ disabling Constraint	◆ ALTER TABLE EMP10 DISABLE CONSTRAINT EMP10_ENAME_UNQ;	
DROP clause- ➢ dropping constraint ,column	Eg ◆ Dropping constraint:ALTER TABLE EMP10 DROP CONSTRAINT EMP10_ENAME_UNQ; ◆ Dropping columns-ALTER TABLE EMP10 DROP column lname;	
TRUNCATE		
TRUNCATE - ➢ remove all records from a table permanently	eg.TRUNCATE TABLE EMP100;	
DROP - ➢ delete objects from the database	eg.DROP TABLE EMP100 PURGE;	

	DML COMMANDS	
➢ INSERT ➢ DELETE ➢ UPDATE ➢ MERGE		
<u>INSERT</u>		

INSERT-
➢ adding new rows to a table.
	INSERT INTO DEPT VALUES(20,'RESEARCH','DALLAS');
	
	INSERT INTO DEPT VALUES(&DNO,&DNAME,&LOC);	

Column listing- ➢ To skip some columns ➢ give input in different order than default	EG. INSERT INTO DEPT (DEPTNO,LOC,DNAME) VALUES(10,'NEW YORK','ACCOUNTING');	

Copying rows from another table	INSERT INTO EMP10
	
	(EMPNO,ENAME,JOB,SAL,DEPTNO)
	
	SELECT EMPNO,ENAME,JOB,SAL,DEPTNO
	
	FROM EMP
	WHERE DEPTNO=10;	

UPDATE-
➢ to modify data in the table.
	UPDATE EMP
	SET JOB='ANALYST',SAL=3500,DEPTNO=10
	WHERE EMPNO=7369;	

DELETE- ➢ for removing rows from a table	eg.DELETE FROM EMP WHERE EMPNO=7934; where clause if not specified leads to all entries being changed	
MERGE - ➢ insert,update and delete can be done together	MERGE INTO EMPLOYEES E1 USING EMP E2 ON (E1.EMPNO=E2.EMPNO) WHEN MATCHED THEN UPDATE SET E1.JOB=E2.JOB,E1.SAL=E2.SAL WHEN NOT MATCHED THEN INSERT VALUES(E2.EMPNO,E2.ENAME,E2.JOB,E2.MGR,E2.HIREDATE,E2.SAL, E2.COMM,E2.DEPTNO)	
	TRANSACTION CONTROL LANGUAGE	
➢ Commit ➢ Rollback ➢ Savepoint		
➢ TCL commands works only for dml commands ➢ For ddl commands automatic commit takes place,no rollback possible		
COMMIT-to save all dml transactions		
ROLLBACK- to undo all dml changes made.		
SAVEPOINT-to divide a transaction into different		

sections.		
ACID PROPERTIES		
ATOMICITY- ➢ a transaction takes place completely or doesnt take place at all.	EG. if two users are logged in and one user is making changes to a particular column,other users wishing to make changes are put in a wait state till the other finishes work..	
CONSISTANCY- ➢ data should be able to be viewed from different systems.	EG.if two users U and U2 log in to a database from different systems both of em should be able to acccess database in data	
ISOLATION- ➢ changes made by a user can be viewed only in that system till a commit is done,earlier snapshot of data is kept.	eg.u1 changes column name t1 for t2...if user u2 accesses database,the colums name is still t1,because user u1 has not run commit..	
DIRTY READ- ➢ if user u2 is able to access changes without a commit being made.	➢ Oracle doesnt support commit	
LOCKING- ➢ implicit ➢ manual	◆ Implicit-done by oracle implicitely(for update and delete) eg. UPDATE DEPT SET LOC='MUMBAI' WHERE DEPTNO=40 ◆ manual- locking done by user varients SELECT * FROM DEPT WHERE DEPTNO=40 FOR UPDATE; or UPDATE WAIT 20 or UPDATE NOWAIT	
DEADLOCKS- ➢ automatically detected by oracle.		

DCL COMMANDS		
GRANT-used to grant permission to users Previllages –system object Only one object can be operated Total of 11 previllages		
Object previllages-permission on various objects(tables,views,etc		
To grant all previllages-	GRANT ALL ON CRICKET TO HR;	
To view previllage	SELECT * FROM USER_TAB_PRIVS;	
To see what previllages are received	SELECT * FROM USER_TAB_PRIVS_RECD;	
To see previllages granted	SELECT * FROM USER_TAB_PRIVS_MADE;	
REVOKE-used for permissions which have been granted earlier	REVOKE DELETE ON CRICKET FROM KRISHNA;	
ROLES-collection of previllages.		
INDEX-created to improve performance Index is a storage location in which indexed column data is stored in sorted order. By default index is created for pk and unique columns		
SET AUTO TRACE ONLY EXPLAIN Explains how statement is executed		
Creadting Index	CREATE UNIQUE INDEX BIGTABLE_ID ON BIGTABLE(ID)	
ROWNO-keywords in stored order along with page no and line no. Search is binary searc Rowed used manually to perform search.	SELECT ROWID,ID FROM BIGTABLE WHERE ID<=0;	
SET AUTO TRACE OFF To disable trace		
Dropping Indices		

LPAD AND RPAD-
➢ for right padding and left padding .
	SELECT DNAME,LPAD(DNAME,15,'.'),
	
	RPAD(DNAME,15,'*') FROM DEPT;
		RPAD(string,width,padding character’s)

LTRIM AND RTRIM		

For left and right trim	SELECT LTRIM(DNAME,'SCOAP'),
	RTRIM(DNAME,'SING')
	FROM DEPT;
		LTRIM(string,char’s)

		RTRIM(string,char’s)

SUBSTR-
➢ returns part of the string
	SELECT DNAME,
	SUBSTR(DNAME,3,4),
	SUBSTR(DNAME,4),
	
	INSTR(DNAME,'C',1),
	INSTR(DNAME,'C',1,2)
	
	FROM DEPT;
		SUBSTR(string,start pos,length)

INSTR ➢ -returns index of the char’s in the given string .	SUBSTR(DNAME,3,4)-returns 4 characters after the 3rd string SUBSTR(DNAME,4)-retruns the string after 4th character. INSTR(DNAME,'C',1)-searches character 'C' from 1st pos. INSTR(DNAME,'C',1,2)-searches for second occurance of character C from first pos.	INSTR(string, char’s, start pos, nth occurance)
TRANSLATE- ➢ overwrites source chars with target chars	TRANSLATE(DNAME,'A','X'),-translates A to X TRANSLATE(DNAME,'AS','XY')-translates A to X and S to Y.	TRANSLATE(string,source,target)

REPLACE-
• replaces source string with target string
	SELECT JOB,
	REPLACE(JOB,'SALESMAN','MARKETING')
	
	FROM EMP WHERE DEPTNO=30;
		REPLACE(string,source,target)

	CHARACTER FUNCTIONS	

LOWER (string) – returns data in lower case	SELECT ENAME,LOWER(ENAME),UPPER('OrACle'),
	
	INITCAP(JOB),CONCAT(JOB,SAL) FROM EMP WHERE
	
	DEPTNO=10;	

UPPER(string) – returns data in upper case		
INITCAP(string) – returns with first character in caps for each word		
CONCAT(string1,string2) – concatenates two strings		
	REGULAR EXPRESSION	
➢ Oracle Database 10g includes support for IEEE/POSIX standard native regular expressions in SQL ➢ Compatible with other programming environments such as Unix, perl and Java REGEXP_LIKE Function - ➢ Applies a LIKE function to a regular expression	SELECT * FROM DEPT WHERE REGEXP_LIKE(LOC,'New');	REGEXP_LIKE (source string, pattern)

pattern ➢ Used primarily in the WHERE clause		➢ Source string specifies source data to be scanned ➢ Pattern is the regular expression to search within the source string ➢ Returns true or false indicating whether the pattern matched the data.
REGEXP_INSTR Function ➢ Returns the position of the pattern within the string ➢ Extension to the INSTR function	SELECT REGEXP_INSTR('We are driving south by south east','south') FROM DUAL; SELECT REGEXP_INSTR('We are driving south by south east', 'south',1,2,0) FROM DUAL;	REGEXP_INSTR (source string, pattern [,start position [,occurrence]]) ➢ Source string specifies source data to be scanned ➢ Pattern is the regular expression to search within the source string ➢ Start position specifies position within source string where search should begin. (default is 1) ➢ Occurrence indicates which occurrence to search for (default is 1) ➢ Returns beginning position of the pattern within the string
INSTR- ➢ Searches for the pattern ➢ returns the matched portion of the string	SELECT REGEXP_SUBSTR('91-080-28461147','-[0-9]+') FROM DUAL;	REGEXP_SUBSTR (source string, pattern [,start position [,occurrence]]) ➢ Same parameters as REGEXP_INSTR ➢ Returns matched portion

		of the pattern from the string

REPLACE-
➢ Searches a pattern in the string
➢ replaces the matched string with the supplied replacement pattern
	SELECT
	
	REGEXP_REPLACE('We are driving south by south east', 'south','north')
	
	from dual;
	
	SELECT
	
	REGEXP_REPLACE('We are driving south by south east', 'south','north',1,1) from
	
	dual;
		REGEXP_REPLACE (source string, pattern [,replace string [,start position
		
		[,occurrence]]])
		
		➢ Same parameters as REGEXP_SUBSTR with one extra parameter
		
		➢ Replace string is the regular expression to replace the matched portion within the source string
		
		➢ Returns replaced string

SUBSTR- ➢ Searches for the pattern and returns the matched portion of the string	SELECT REGEXP_SUBSTR('91-080-28461147','-[0-9]+') FROM DUAL;	REGEXP_SUBSTR (source string, pattern [,start position [,occurrence]]) ➢ Same parameters as REGEXP_INSTR ➢ Returns matched portion of the pattern from the string
COUNT- ➢ returns number of occurances ➢ 11g feature	SELECT DNAME,REGEXP_COUNT(DNAME,'E') FROM DEPT;	
<u>NUMBER FUNCTIONS</u>-look at pdf		
DUAL- ➢ public synonym. ➢ Contains 1 column by		

name dummy ➢ owned by sys ➢ only select permission given to all users		
	DATE FUNCTION	
<u>DEFAULT DATE FORMAT</u> DD-MON-RR		
SYSDATE ➢ To display server date and time		
To reset to default format	ALTER SESSION SET NLS_DATE_FORMAT='DD-MON-RR';	

MONTHS_BETWEEN() – returns number of months between 2 dates	SELECT ENAME,HIREDATE,MONTHS_BETWEEN(SYSDATE,HIREDATE) FROM
	
	EMP;	

ADD_MONTHS() – to add number of months to date	SELECT ENAME,HIREDATE,ADD_MONTHS(HIREDATE,3) FROM EMP
	
	WHERE DEPTNO=10;	

LAST_DAY() – returns last date of the month	SELECT ENAME,HIREDATE,LAST_DAY(HIREDATE) FROM EMP
	
	WHERE DEPTNO=10;	

NEXT_DAY() – to find date of the approaching week of the day	SELECT SYSDATE,NEXT_DAY(SYSDATE,'MONDAY'),NEXT_DAY(SYSDATE,2)
	
	FROM DUAL;	

CONVERSATIONS		
IMPLICIT		
EXPLICIT-TO_CHAR TO_NUMBER TO_DATE		
TO_CHAR() – converts number/date to character type/format	SELECT ENAME,SAL,TO_CHAR(SAL,'9,99,999.99') FROM EMP;	

TO_DATE() – to convert character to date type	SELECT * FROM EMP WHERE HIREDATE='3-DEC-81';
	
	SELECT * FROM EMP WHERE
	
	HIREDATE=TO_DATE('3/12/1981','DD/MM/YYYY');	

ROUND	➢ ROUND(date) => date is rounded off based on time if time < 12 noon returns same date else returns next day's date ➢ ROUND(date,'MONTH') => date will rounded off based on day of the month if day <= 15th returns 1st of same month else returns 1st of next month ➢ ROUND(date,'YEAR') => date will be rounded off based on month if month <= 'JUNE' returns 1st JAN of same year else returns 1st JAN of next year	
TRUNC-	➢ TRUNC(date) => returns same date ➢ TRUNC(date,'MONTH') => returns 1st of same month ➢ TRUNC(date,'YEAR') => returns 1st JAN of same year	

TRIM	Select trim('a' from 'aaabbb2305542') from dual;
	Select trim(0 from 0011252305542) from dual;	

LEAST	SELECT LEAST(10,40,20),LEAST('SACHIN','SEHWAG','DHONI')
	
	FROM DUAL;	

GREATEST	SELECT GREATEST(10,40,20),GREATEST('SACHIN','SEHWAG','DHONI')
	
	FROM DUAL;	

	GROUP BY CLAUSE	

Used to find aggregates by group ing data of columns which have data shared amongst rows.	To find total salary for each dept
	:
	
	SELECT DEPTNO,SUM(SAL) FROM EMP
	
	GROUP BY DEPTNO;	

HAVING CLAUSE
 	
➢ For restricting GROUP FUNCTIONS, HAVING clause must be used
	SELECT DEPTNO,AVG(SAL)
	
	FROM EMP
	
	GROUP BY DEPTNO
	
	HAVING AVG(SAL) > 2000;	

ROLLUP FUNCTION ➢ sub-total and sum-total	SELECT DEPTNO,JOB,COUNT(*),SUM(SAL) FROM EMP	

aggregates displayed.	GROUP BY ROLLUP(DEPTNO,JOB);	
<u>JOINING TABLES</u>		
➢ Joins are used for fetching rows from multiple tables.		
TYPES		
INNER JOIN ➢ **j**oining of tables based on matching data in tables ➢ Types: Equi Join Non Equi Join Self-Join		
Equi Join- ➢ tables are joined by comparing column data in both the tables using "=" operator ➢ Types:Natural Join Join with using clause Using INNER JOIN with ON clause		
Joining using condition in WHERE clause		

➢ using table name as qualifier	SELECT emp.empno,emp.ename,emp.job,emp.sal,emp.deptno,
	
	dept.deptno,dept.dname,dept.loc
	
	FROM emp,dept
	
	WHERE emp.deptno=dept.deptno;	

➢ Using correlation name	SELECT	

(table alias) as qualifier	e.empno,e.ename,e.job,e.sal,e.deptno,d.deptno,d.dname,d.loc
	
	FROM EMP e,DEPT d
	
	WHERE e.deptno=d.deptno;	

➢ Using ANSI Standard SQL JOIN syntax	SELECT e.empno,e.ename,e.job,e.sal,e.deptno,d.deptno,d.dname,d.loc
	
	FROM EMP e INNER JOIN DEPT d
	
	ON e.deptno=d.deptno;	

NATURAL JOIN:
➢ In NATURAL JOINS, tables to be joined must have 1 or more matching column names
 	
➢ Secondly data must be matching in such columns.
 	
➢ Oracle internally performs EQUI join
 	
➢ No qualifier are allowed in NATURAL joins
	CREATE TABLE EMPLOYEE
	
	AS
	
	SELECT EMPNO,ENAME,JOB,SAL,DEPTNO,DNAME
	
	FROM EMP NATURAL JOIN DEPT;	

Join with using clause
 	
➢ Column used in USING clause must not contain qualifier anywhere in the statement
	SELECT E.EMPNO,E.ENAME,E.JOB,E.SAL,DEPTNO,DNAME,D.LOC
	
	FROM EMPLOYEE E JOIN DEPT D
	
	USING(DEPTNO,DNAME);	

Non Equi Join
➢ tables are joined by comparing column data in both the tables using other than “=” operator
	SELECT E.EMPNO,E.ENAME,E.JOB,E.SAL, S.GRADE
	
	FROM EMP E INNER JOIN SALGRADE S	

	ON E.SAL BETWEEN S.LOSAL AND S.HISAL;	

Self-Join -
➢ to join a table to ITSELF
	SELECT E.ENAME EMPNAME,E.SAL EMPSAL,M.ENAME MGRNAME,M.SAL MGRSAL
	
	FROM EMP E INNER JOIN EMP M
	
	ON E.MGR=M.EMPNO;	

OUTER JOIN ➢ Joining of tables based on matching & unmatched data in tables ➢ Types:Left outer join Right outer join Full outer join		

Left outer join
➢ to fetch all the columns in left table and only matched columns in right table
	SELECT E.EMPNO,E.ENAME,E.JOB,E.SAL,E.DEPTNO,D.DEPTNO,D.DNAME,D.LOC
	
	FROM EMP E LEFT OUTER JOIN DEPT D
	
	ON E.DEPTNO=D.DEPTNO;	

Right outer join
➢ to fetch all the columns in right table and only matched columns in left table
	SELECT e.empno,e.ename,e.job,e.sal,e.deptno,d.deptno,d.dname,d.loc
	
	FROM emp e RIGHT OUTER JOIN dept d
	
	ON e.deptno=d.deptno;	

Full outer join
➢ To fetch all the columns in both the tables thar are being joined
	SELECT e.empno,e.ename,e.job,e.sal,e.deptno,d.deptno,d.dname,d.loc
	
	FROM emp e FULL OUTER JOIN dept d	

	ON e.deptno=d.deptno;	

CROSS JOIN
 	
➢ produces CARTESIAN product
 	
➢ Rarely used
 	
➢ Output would be cross product of 2 tables
	SELECT e.empno,e.ename,e.job,e.sal,e.deptno,d.deptno,d.dname,d.loc
	
	FROM emp e CROSS JOIN dept d;	

HEIRARCHICAL QUERIES		

➢ To display a hierarchy.
➢ CONNECT BY PRIOR used
➢ SYS_CONNECT_BY_PATH can aslo be used
	SELECT SYS_CONNECT_BY_PATH(ENAME,'->') "PATH"
	
	FROM EMP
	
	START WITH ENAME='KING'
	
	CONNECT BY PRIOR EMPNO=MGR	

SET OPERATORS ➢ used for combining data from multiple sets/queries/tables and extract desired data ➢ while comparing multiple columns, number of columns and type of those columns must be compatible ➢ -ORDER BY clause must be used in LAST statement.		
UNION ALL ➢ returns data from both	SELECT JOB FROM EMP1 UNION ALL	

the sets including duplicates	SELECT JOB FROM EMP2;	

UNION
➢ returns distinct data from both the sets (distinct of UNION ALL)
	SELECT JOB FROM EMP1
	
	UNION
	
	SELECT JOB FROM EMP2;	

INTERSECT
➢ returns distinct common data from both the sets
	SELECT JOB FROM EMP1
	
	INTERSECT
	
	SELECT JOB FROM EMP2;	

MINUS
➢ returns distinct data present in first set but missing in second set
	SELECT JOB FROM EMP1
	
	MINUS
	
	SELECT JOB FROM EMP2;	

	SUB QUERIES	
SUB QUERIES ➢ Sub-query is a SELECT statement nested within another SELECT statement. ➢ Sub-queries are used for searching data based on unknown values in search conditions. ➢ Sub-queries would fetch data dynamically which will be used by outer query. ➢ Types of sub-queries 1. Single row sub-queries		

2. Multi row sub-queries 3. Correlated sub-queries		

Single row sub-queries
➢ Sub-queries which return single row
➢ First sub-query is executed
➢ Next outer query is executed by using row fetched by sub-query
	SELECT ename,sal FROM EMP
	
	WHERE sal=(SELECT min(sal) FROM EMP);	

Multi-row subqueries :
➢ sub-queries returning more than 1 row
➢ Multi-row comparision operators:
IN
NOT IN
ANY
ALL
EXIST
NOT EXISTS
	To list minumum salary in each department
	
	SELECT ENAME,DEPTNO,SAL FROM EMP
	
	WHERE SAL IN (SELECT MIN(SAL) FROM EMP
	
	GROUP BY DEPTNO);	

Multi-column multi-row sub-query
➢ more than one parameter is needed in where condition.
➢ eg. to get salary of employees in diff department(two employees may have same salary,therefore we need deptno,sal in where)
	SELECT EMPNO,ENAME,DEPTNO,SAL FROM EMP
	
	WHERE (DEPTNO,SAL) IN (SELECT DEPTNO,MIN(SAL) FROM EMP
	
	GROUP BY DEPTNO);	

Comparison operators		

ANY
➢ To get
ANY of the values
	SELECT empno,ename FROM emp
	
	WHERE sal > ANY(SELECT sal FROM	

retrieved by sub-query.	emp WHERE deptno=30);	
ALL ➢ To get ALL the values retrieved by sub-query.		

EXISTS & NOT EXISTS
 	
➢ EXISTS operator returns status TRUE if sub-query returns any rows else return
FALSE.
➢ If EXISTS operator returns TRUE, outer displays the row else row is
discarded.
 	
➢ NOT EXISTS operator is complement to EXISTS operator
	SELECT empno,ename FROM emp e
	
	WHERE EXISTS (SELECT empno FROM emp
	
	WHERE mgr=e.empno);	

Correlated sub-queries	SELECT empno,ename,sal,deptno FROM emp e
	
	WHERE sal > (SELECT avg(sal) FROM emp
	
	WHERE deptno=e.deptno);	

SYNONYM ➢ synonym is an alias/alternate name for an object which will be stored in database ➢ using synonyms you can avoid schema qualifier		

CREATION and DROPPING-	CREATE SYNONYM SUBJECT
	
	FOR DAC1.SUBJECT;
	
	DROP SYNONYM SUBJECT;	

TO VIEW SYNONYM	SELECT * FROM USER_SYNONYMS;	
VIEWS		
➢ A table which appears to be existing but is physically non existant. ➢ It is stored as select table ➢ derived from another view or table		
ADVANTAGES: ➢ restricting db access ➢ making complex queries simple		
Types of views:		
(1.)Simple views- ➢ derived using single tables and does not contain functions and group functions		

CREATION	CREATE VIEW EMP10VIEW
	
	AS
	
	SELECT EMPNO,ENAME,JOB,DEPTNO FROM EMP
	
	WHERE DEPTNO=10 WITH CHECK OPTION;	

(2.)Complex views- ➢ derived using multiple tables and contains functions		

CREATION	CREATE VIEW EMPDEPTGRADE
	
	AS
	
	SELECT E.EMPNO,E.ENAME,E.JOB,E.SAL,S.G	

	RADE,E.DEPTNO,D.DNAME,D.LOC FROM EMP E JOIN DEPT D ON E.DEPTNO=D.DEPTNO JOIN SALGRADE S ON E.SAL BETWEEN S.LOSAL AND S.HISAL;	
TO view the created views	SELECT * FROM USER_VIEWS;	
DROPING VIEW	DROP VIEW EMPDEPTGRADE;	
	Materialized views	

(3.)Materialized views-
➢ When materialized views, SELECT statement will be executed and data will be stored physically
➢ These are used for optimization in datawarehousing environments
	CREATE MATERIALIZED VIEW EMPDEPTMATVIEW
	REFRESH ON COMMIT
	AS
	SELECT E.EMPNO,E.ENAME,E.JOB,E.SAL,E.DEPTNO,D.DNAME,D.LOC
	FROM EMP E JOIN DEPT D
	ON E.DEPTNO=D.DEPTNO;	

	SEQUENCES	
SEQUENCES- ➢ Sequence is a stored database object which is used to generate sequence of numbers. ➢ These numbers are used as data in any database column		
CREATING SEQUENCE	CREATE SEQUENCE DEPTSEQ	

	START WITH 5 INCREMENT BY 1 MINVALUE 1 MAXVALUE 10 CYCLE CACHE 5;	
CURVAL ➢ returns current sequence number	SELECT DEPTSEQ.CURVAL FROM DUAL	
NEXTVAL ➢ will initialize the sequences ➢ returns next sequence number	SELECT DEPTSEQ.NEXTVAL FROM DUAL	
ALTERING SEQUENCE	ALTER SEQUENCE DEPTSEQ INCREMENT BY 5 NOMAXVALUE NOCYCLE;	
VIEWING SEQUENCES	SELECT * FROM USER_SEQUENCES;	
DROPPING SEQUENCE	DROP SEQUENCE DEPTSEQ;	
ROWNUM – ➢ pseudo column which returns sequence of numbers starting from 1 for every row in a table. ➢ using ROWNUM you can display or fetch top 'n' rows from the table	SELECT ROWNUM,ENAME,SAL FROM EMP;	