**SQL (DML, DDL, TCL and Queries)**

**-- Creating a table with certain constrains.**

CREATE TABLE products (productid NUMBER(10) PRIMARY KEY, product\_name VARCHAR2(30) NOT NULL,price NUMBER(10) NOT NULL,CHECK(price>100), product\_catagory VARCHAR2(20));

*-- Altering a table to modify the column.*

ALTER TABLE products MODIFY (product\_catagory VARCHAR2(25));

*-- Altering the table to add the column.*

ALTER TABLE products ADD (supplier\_name VARCHAR2(20));

*-- Altering the table to rename the column.*

ALTER TABLE products RENAME COLUMN price TO product\_price;

*-- Altering the table to drop or delete the column.*

ALTER TABLE products DROP COLUMN supplier\_name;

*-- Altering the table to add multiple columns.*

ALTER TABLE products ADD (product\_type VARCHAR2(20),supplier\_name VARCHAR2(30), supplierid NUMBER(20) UNIQUE);

*-- Altering the table to drop multiple columns.*

ALTER TABLE products DROP (product\_type, supplierid);

*-- Renaming the table name.*

RENAME products TO products\_table;

*-- All columns, single row insertion.*

INSERT INTO products\_table VALUES (101,'Whey Protein',120,'Sports Supplements','Optimum Nutrition');

*-- All columns multiple row insertion.*

INSERT INTO products\_table VALUES (&productid,'&product\_name',&product\_price,'&product\_category','&supplier\_name');

*-- Particular column single row*

INSERT INTO products\_table (productid,product\_name,product\_price) VALUES (103,'Airpods',5000);

*-- Particular column multiple rows*

INSERT INTO products\_table (productid, product\_name, product\_price) VALUES (&productid,'&product\_name',&product\_price);

*-- Altering the table to add the column.*

ALTER TABLE products\_table ADD (order\_date DATE);

*-- Entire column update*

UPDATE products\_table SET order\_date = sysdate;

*-- Particular column multiple rows insertion*

INSERT INTO products\_table (productid, product\_name, product\_price,product\_catagory,supplier\_name) VALUES (&productid,'&product\_name',&product\_price,'&product\_catagory','&supplier\_name');

*-- Particular row single column update*

UPDATE products\_table SET supplier\_name='Apple' WHERE productid=103;

*-- Particular row multiple column update*

UPDATE products\_table SET product\_price=5500, product\_catagory='Electronics' WHERE productid=103;

UPDATE products\_table SET product\_catagory='Books',supplier\_name='Oracle' WHERE productid=202;

*-- Multiple rows single column update*

UPDATE products\_table SET product\_price=1500 WHERE productid IN (101,102);

*-- Multiple row multiple column update*

UPDATE products\_table SET product\_catagory='Electronics', supplier\_name='Apple' WHERE productid IN (103,201);

*-- Particular record delete*

DELETE FROM products\_table WHERE productid=202;

*-- Multiple record delete*

DELETE FROM products\_table WHERE supplier\_name IN ('Apple');

*-- Saves the above statements to secondary memory and finalize it.*

COMMIT;

*-- Goes to the previous state or undo the previous single or multiple commands. It works inside the primary memory.*

ROLLBACK;

**-- Select Statements**

SELECT \* FROM products\_table;

SELECT DISTINCT supplier\_name FROM products\_table;

SELECT \* FROM products\_table WHERE supplier\_name='Apple';

SELECT productid, product\_name, supplier\_name FROM products\_table;

SELECT productid, product\_name, supplier\_name FROM products\_table WHERE product\_catagory IN ('Sports Supplements','Electronics') ;

**-- Clauses used with group by.**

SELECT MAX(product\_price) FROM products\_table;

SELECT MIN(product\_price) FROM products\_table;

SELECT AVG(product\_price) FROM products\_table;

SELECT COUNT(product\_price) FROM products\_table;

**-- Order by**

SELECT \* FROM products\_table ORDER BY product\_price;

SELECT \* FROM products\_table ORDER BY product\_price DESC;

**-- Where and use of between, not between, like, AND & OR.**

SELECT \* FROM products\_table WHERE supplier\_name = 'Apple' AND product\_catagory='Electronics';

SELECT \* FROM products\_table WHERE supplier\_name = 'Apple' OR product\_catagory='Sports Supplements';

SELECT \* FROM products\_table WHERE product\_price BETWEEN 100 AND 500;

SELECT \* FROM products\_table WHERE product\_price NOT BETWEEN 100 AND 500;

SELECT \* FROM products\_table WHERE supplier\_name LIKE 'A%';

SELECT \* FROM products\_table WHERE product\_catagory LIKE '%s';

SELECT \* FROM products\_table WHERE product\_catagory LIKE '\_p%';

SELECT \* FROM products\_table WHERE product\_catagory LIKE '%n\_\_';

SELECT \* FROM products\_table WHERE ROWNUM BETWEEN 1 AND 3;

SELECT \* FROM products\_table WHERE ROWNUM<=5;

**-- Group by**

SELECT MAX(product\_price) FROM products\_table GROUP BY product\_price;

SELECT supplier\_name, COUNT(\*) FROM products\_table GROUP BY supplier\_name;

SELECT supplier\_name, COUNT(\*) FROM products\_table GROUP BY supplier\_name HAVING (supplier\_name='Apple');

**------------------------------------ Creating Another Table--------------------------------------------------------**

CREATE TABLE orders (orderid NUMBER PRIMARY KEY, productid NUMBER, FOREIGN KEY (productid) REFERENCES products\_table(productid));

INSERT INTO orders VALUES(100,101);

INSERT INTO orders VALUES(101,102);

INSERT INTO orders VALUES(103,103);

INSERT INTO orders VALUES(104,201);

**---------------------------------------------------------------------------------------------------------------------------**

**--inner join (provides the data that are common to both the tables).**

SELECT A.orderid,b.productid, b.product\_name FROM orders A INNER JOIN products\_table b ON A.productid=b.productid;

**-- left join (provides the data that are common to both the tabkes plus the remaining datas from the left table.)**

SELECT A.orderid,b.productid, b.product\_name FROM orders A LEFT JOIN products\_table b ON A.productid=b.productid;

**--right join (provides the data that are common to both the tabkes plus the remaining datas from the right table.)**

SELECT A.orderid,b.productid, b.product\_name FROM orders A RIGHT JOIN products\_table b ON A.productid=b.productid;

-------------------------------------------------------------------------------------------------------------------------------

**-- Subqueries**

***-- Single row subquery. It returns oly one value and that is assined by "=" Operator.***

*-- Gives the maximum price*

SELECT \* FROM products\_table WHERE product\_price = (SELECT MAX(product\_price) FROM products\_table);

-- Gives the second maximum price.

SELECT \* FROM products\_table WHERE product\_price = (SELECT MAX(product\_price) FROM products\_table WHERE product\_price<(SELECT MAX(product\_price) FROM products\_table));

***-- Multiple row subquery***

**-- The ANY operator returns true if any of the subquery values meet the condition.**

*-- =any gives the values that are between 100 and 1000 in this case.*

SELECT \* FROM products\_table WHERE product\_price = ANY (SELECT product\_price FROM products\_table WHERE product\_price BETWEEN 100 AND 1000);

*-- <any gives the values that are between 100 and 1000 plus the values that are less than those values.*

SELECT \* FROM products\_table WHERE product\_price < ANY (SELECT product\_price FROM products\_table WHERE product\_price BETWEEN 100 AND 1000);

*-- <any gives the values that are between 100 and 1000 plus the values that are greater than those values.*

SELECT \* FROM products\_table WHERE product\_price > ANY (SELECT product\_price FROM products\_table WHERE product\_price BETWEEN 100 AND 1000);

**-- The ALL operator returns true if all of the subquery values meet the condition.**

*-- >all displays the values that are greater then the retrieved values from the subquery.*

SELECT \* FROM products\_table WHERE product\_price > ALL (SELECT product\_price FROM products\_table WHERE product\_price BETWEEN 0 AND 2000);

*-- <all displays the values that are greater then the retrieved values from the subquery.*

SELECT \* FROM products\_table WHERE product\_price < ALL (SELECT product\_price FROM products\_table WHERE product\_price BETWEEN 0 AND 2000);

-----------------------------------------------------------------------------------------------------------------------------------------------------

**-- Having Keyword**

SELECT product\_price FROM products\_table GROUP BY product\_price HAVING product\_price>100;