Data manipulation language

# Select

* **arithematic opeartion**

**SELECT last\_name, salary, salary + 300**

**FROM employees;**

* **alias**

**SELECT last\_name "Name" , salary\*12 "Annual Salary"**

**FROM employees;**

* **concatenation operator**

**SELECT last\_name || job\_id AS "Employees"**

**FROM employees;**

* **alternatve quote**

**SELECT department\_name ||**

**q'[, it's assigned Manager Id: ]'**

**|| manager\_id**

**AS "Department and Manager"**

**FROM departments;**

**// to get any created table data**

**desc table\_name**

**// to get all tables owned by user connect it then**

**select tablespace\_name, table\_name from user\_tables;**

**Select \* from tab ;**

* **where with advanced work**

**SELECT last\_name, salary**

**FROM employees**

**WHERE salary <> 3000**

**AND salary BETWEEN 2500 AND 3500**

**AND manager\_id IN (100, 101, 201)**

**ANd first\_name LIKE 'S%'**

**OR manager\_id IS NULL**

**or job\_id**

**NOT IN ('IT\_PROG', 'ST\_CLERK', 'SA\_REP') ;**

* **using scape operator to search for %**

**select mycol from mytable where mycol like '%\%%';**

**// here we scaped the middle %**

* **order by**

**SELECT last\_name, department\_id, salary**

**FROM employees**

**ORDER BY department\_id, salary DESC;**

* **subustitution variable**

**SELECT employee\_id, last\_name, salary, department\_id &column\_name**

**FROM employees**

**WHERE &condition**

**or employee\_id = &employee\_num**

**or job\_id = '&job\_title'**

**order by &order\_column;**

* **reuse the substitiution variable**

**SELECT employee\_id, last\_name, job\_id, &&column\_name**

**FROM employees**

**ORDER BY &column\_name ;**

* **select with column number and order with one up and one down**

**SELECT LNAME, JOB, SALARY FROM trainer**

**WHERE STATE = ‘Giza’**

**ORDER BY 3 DESC, 2;**

* **select ‘fixed\_string’ from tabel\_name;**
* **correlated sub-query with exists 🡪 returns false for empty set**

**SELECT DISTINCT ORDER\_ID FROM ORDER\_LINE\_T**

**WHERE EXISTS (SELECT \* FROM PRODUCT\_T**

**WHERE PRODUCT\_ID = ORDER\_LINE\_T.PRODUCT\_ID**

**AND PRODUCT\_FINISH = ‘Natural ash’);**

* **SELECT PRODUCT\_DESCRIPTION, STANDARD\_PRICE, AVGPRICE**

**FROM (SELECT AVG(STANDARD\_PRICE) AVGPRICE FROM PRODUCT\_T),**

**PRODUCT\_T**

**WHERE STANDARD\_PRICE > AVGPRICE;**

* **sub-query in the from is prefered over this one in where**
* **define variabel**

**DEFINE employee\_num = 200**

* **woring with date**

**SELECT last\_name, hire\_date FROM employees**

**WHERE hire\_date < '01-FEB-88';**

* **Subtract two dates to find the number of days between them**
* **distinct with select statements so you can select unique coloum values or unique total row values**

**//DUAL is an empty table that we can use to do operation on select;**

* **SQL functions:**
* **single functions**
* **character**
* **LOWER**
* **UPPER**
* **INITCAP**
* **CONCAT('Hello', 'World')**
* **SUBSTR('HelloWorld',1,5)**
* **LENGTH('HelloWorld')**
* **INSTR('HelloWorld', 'W')**
* **LPAD(salary,10,'\*')**
* **RPAD(salary, 10, '\*')**
* **REPLACE ('JACK and JUE','J','BL')**
* **TRIM('H' FROM 'HelloWorld'**
* **numbers**
* **ROUND(45.926, 2)**
* **TRUNC(45.926, 2)**
* **MOD(1600, 300)**
* **date**
* **MONTHS\_BETWEEN ('01-SEP-95','11-JAN-94')**
* **ADD\_MONTHS ('11-JAN-94',6)**
* **NEXT\_DAY ('01-SEP-95','FRIDAY')**
* **LAST\_DAY ('01-FEB-95')**
* **ROUND(SYSDATE,'MONTH')**
* **ROUND(SYSDATE ,'YEAR')**
* **TRUNC(SYSDATE ,'MONTH')**
* **TRUNC(SYSDATE ,'YEAR')**
* **TO\_CHAR(number, 'format\_model')**
* **TO\_CHAR(DATE, 'format\_model')**
* **TO\_NUMBER(char[, 'format\_model'])**
* **TO\_DATE(char[, 'format\_model'])**
* **general**
* **NVL (expr1, expr2)**
* **NVL2 (expr1, expr2, expr3)**
* **NULLIF (expr1, expr2)**
* **COALESCE (expr1, expr2, ..., exprn)**
* **CASE job\_id WHEN 'IT\_PROG' THEN 1.10\*salary**

**WHEN 'ST\_CLERK' THEN 1.15\*salary**

**WHEN 'SA\_REP' THEN 1.20\*salary**

**ELSE salary END**

* **DECODE(job\_id, 'IT\_PROG', 1.10\*salary,**

**'ST\_CLERK', 1.15\*salary,**

**'SA\_REP', 1.20\*salary,**

**salary)**

* **group function**
* **AVG**
* **COUNT 🡪 counts number of non null rows / can use distinct**
* **MAX 🡪 can be used with characger , number and date**
* **MIN 🡪 can be used with characger , number and date**
* **SUM**
* **constraint any attribute writen beside the group function should be added in the group by**

**select department\_id , avg (salary ) from employees**

**group by department\_id ;**

* **group with two paramaters**

**SELECT department\_id dept\_id, job\_id, SUM(salary)**

**FROM employees**

**GROUP BY department\_id, job\_id**

**HAVING MAX(salary)>10000 ;**

* **table in anther user**

**SELECT \* FROM userA.employees;**

# Update

* **UPDATE employees**

**SET department\_id = 55**

**WHERE department\_id = 110;**

**Delete 🡪 if we don’t use where clause it will delete all rows**

* **DELETE FROM departments**

**WHERE department\_id = 60;**

* **delete with sub-quary**
* **DELETE FROM employees**

**WHERE department\_id =**

**(SELECT department\_iD FROM departments**

**WHERE department\_name LIKE '%Public%');**

* **we should commit after we update , delete , insert in the data so we keep changes we made in data**

# Insert

* **INSERT INTO departments(department\_id,**

**department\_name, manager\_id, location\_id)**

**VALUES (70, 'Public Relations', 100, 1700);**

**// Optionally, list the columns in the INSERT clause**

* **insert with sub-quary 🡪 we don’t use values clause**
* **INSERT INTO sales\_reps(id, name, salary, commission\_pct)**

**SELECT employee\_id, last\_name, salary, commission\_pct**

**FROM employees**

**WHERE job\_id LIKE '%REP%';**

Data Definition language

**/ if two entities has dependent relations**

**//we need to make them**

**//then we alter one of them with the new relation**

**// we create tables in order of dependency**

**// then we check the constrains of each data**

**create table job(**

**jo\_id varchar2(3) primary key ,**

**job\_No varchar2(20) default 1200 ,**

**minSal number(4) unique not null ) ;**

* **creating a table using sub-query**

**CREATE TABLE dept80**

**AS**

**SELECT employee\_id, last\_name,**

**salary\*12 ANNSAL,**

**hire\_date**

**FROM employees**

**WHERE department\_id = 80;**

* **constrains in table creation**
* **primary key**
* **foreign key**
* **not null**
* **unique**
* **checks**
* **creatig index**

**Create index Name\_IDX on Customer\_T(Customer\_Name);**

* **deleting index**

**Drop Index Name\_IDX;**

* **name constraint in the same line**
* **employee\_id NUMBER(6) CONSTRAINT emp\_emp\_id\_pk PRIMARY KEY**
* **Name constraint in the last line**
* **CONSTRAINT emp\_emp\_id\_pk PRIMARY KEY (EMPLOYEE\_ID)**
* **CONSTRAINT emp\_email\_uk UNIQUE(email)**
* **constraint emp\_pri primary key (emp\_id , emp\_name)**
* **check ( emp\_name in ( ‘Mahmoud ’ , ‘ALi’) )**
* **foreign key naming in the last line**
* **CONSTRAINT emp\_dept\_fk FOREIGN KEY (department\_id) REFERENCES departments(department\_id)**
* **check input range for an attribute**

**CONSTRAINT emp\_salary\_min CHECK (salary > 0)**

**//to delete a table**

**//You cannot roll back the DROP TABLE statement**

**DROP TABLE dept80;**

**Alter**

* **ALTER TABLE department DROP COLUMN head\_id ;**
* **ALTER TABLE department ADD head\_id number(11) REFERENCES employee(emp\_id);**
* **ALTER TABLE table\_name RENAME TO new\_table\_name;**
* **ALTER TABLE supplier MODIFY supplier\_name varchar2(100) not null;**
* **ALTER TABLE supplier RENAME COLUMN supplier\_name to sname;**

# Truncate 🡪 delete all rows in tables and leave it empyt

* **TRUNCATE TABLE copy\_emp;**

All join types

**SELECT table1.column, table2.column**

**FROM table1**

**[NATURAL JOIN table2] |**

**[JOIN table2 USING (column\_name)] |**

**[JOIN table2 ON (table1.column\_name = table2.column\_name)]|**

**[LEFT|RIGHT|FULL OUTER JOIN table2 ON (table1.column\_name = table2.column\_name)]|**

**[CROSS JOIN table2];**

* **using table alaisees**

**SELECT e.employee\_id, e.last\_name, d.location\_id, department\_id**

**FROM employees e JOIN departments d**

**USING (department\_id);**

* **join with additional conditions**

**SELECT e.employee\_id, e.last\_name, e.department\_id, d.department\_id, d.location\_id**

**FROM employees e JOIN departments d**

**ON (e.department\_id = d.department\_id)**

**AND e.manager\_id = 149;**

* **multiple join**

**SELECT employee\_id, city, department\_name**

**FROM employees e**

**JOIN departments d**

**ON d.department\_id = e.department\_id**

**JOIN locations j**

**ON e.salary BETWEEN j.lowest\_sal AND j.highest\_sal;**

* **outer join get data of matched and unmatche data**

**SELECT e.last\_name, d.department\_id, d.department\_name**

**FROM employees e FULL/left/right OUTER JOIN departments**

**ON (e.department\_id = d.department\_id);**

Sub-queries

* **SELECT last\_name FROM employees**

**WHERE salary >**

**(SELECT salary FROM employees WHERE last\_name = 'Abel');**

* **sub-querie with havingg**

**SELECT department\_id, MIN(salary)**

**FROM employees GROUP BY department\_id**

**HAVING MIN(salary) >**

**(SELECT MIN(salary)FROM employees WHERE department\_id = 50);**

* **multiple sub-query using ANY / ALL / in**

**SELECT employee\_id, last\_name, job\_id, salary FROM employees**

**WHERE salary < ALL/ANY**

**(SELECT salary FROM employees WHERE job\_id = 'IT\_PROG');**

Data control language

**clear screen**

**// any command must end with semi-colon**

**// user can add new database**

**// but admin has rule to adjust database**

**create user software identified by password;**

**// we should give privilege to user to connect**

**alter user hr account unlock;**

**//to connect as admin**

**connecgt system**

**alter user hr identified by password2 ;**

**grant connect to software;**

**// give user privilege to create**

**grant resource to software;**

**// in column level references means foreign key**

**// in table level we must use foreign key keyword**

**// at start**

**Connect software**

**Password**

**// A DDL or DCL statement executes (automatic commit).**

* **We can create save point inside our work**
* **SAVEPOINT update\_done**
* **To return to this save point**
* **ROLLBACK TO update\_done**
* **BEGIN TRANSACTION/END TRANSACTION to make tansaction baundary**
* **Exec Product\_Line\_sale to execute funcion or precedure**

Set operators 🡪 the expressions in the SELECT lists must match in number and data type.

* **Union 🡪 get both queries without duplication**
* **SELECT employee\_id, job\_id FROM employees**

**UNION**

**SELECT employee\_id, job\_id FROM job\_history;**

* **Union All 🡪 get both queries with duplication**
* **SELECT employee\_id, job\_id, department\_id**

**FROM employees**

**UNION ALL**

**SELECT employee\_id, job\_id, department\_id**

**FROM job\_history ORDER BY employee\_id;**

* **intersection 🡪 return data in both queries**

**SELECT employee\_id, job\_id FROM employees**

**INTERSECT**

**SELECT employee\_id, job\_id FROM job\_history;**

* **minus 🡪 rows of first query without the second one**
* **SELECT employee\_id,job\_id**

**FROM employees**

**MINUS**

**SELECT employee\_id,job\_id**

**FROM job\_history;**

* **create trigger t\_name**

**{before/after/instead of}{insert/delete/update} on t\_name**

**[for each {row/statement} ][when condtion]**

**< trigger sql here>**

* **creating view**

**CREATE VIEW EXPENSIVE\_STUFF\_V AS**

**SELECT PRODUCT\_ID, PRODUCT\_NAME, UNIT\_PRICE**

**FROM PRODUCT\_T**

**WHERE UNIT\_PRICE >300**

**WITH CHECK\_OPTION;**