Bases de données

Prof. Giovanna Di Marzo Serugendo
Assistant: Mohammad Parhizkar
mohammad.parhizkar@unige.ch

Semestre: printemps 2019

session 2- 04/03/2019

Exercises

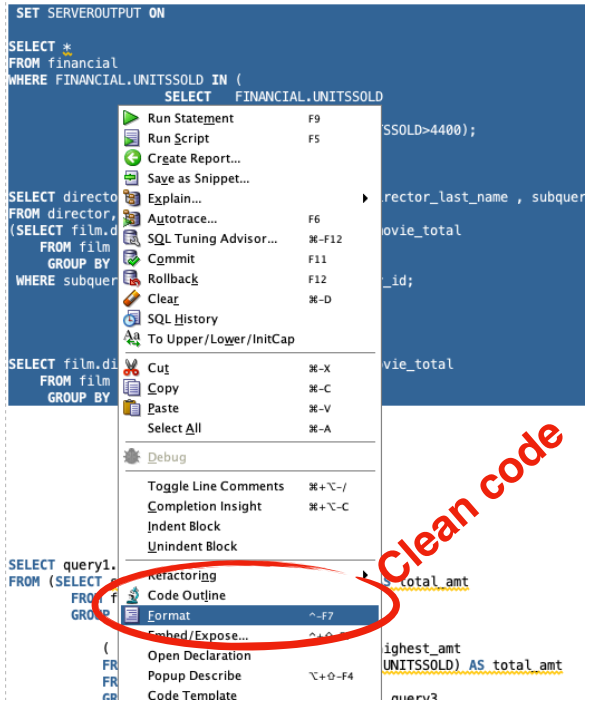
Important

1. Please be reminded that you must submit your TPs answers on the platform.
2. Please submit your code and also your results as just one file (sql, pdf, word, txt …)
3. The name of the file should be (your name + the exercise number)

Our protocols

- PL/SQL keywords <u>are NOT case sensitive</u>: select is the same as SELECT
- In this course we will write all PL/SQL keywords in <u>upper-case</u>.
- <u>Double quotes</u> around a column name, it will make it <u>case sensitive</u>
- An unwanted comma at the end of last line… ??? **NO**

lower case for the table/column identifiers




```
CREATE TABLE films(
  film_id int NOT NULL,
  title char(35) NOT NULL,
  year DATE NOT NULL,
  language char(20),
  genre char(20) NOT NULL,
  director_id numeric(10) NOT NULL,
  main_actor_id int NOT NULL,
  company_id Number(6) NOT NULL,
  CONSTRAINT films_pk PRIMARY KEY  (film_id),
  CONSTRAINT fk_directors FOREIGN KEY (director_id) REFERENCES directors(director_id),
  CONSTRAINT fk_actor FOREIGN KEY  (main_actor_id) REFERENCES main_actors(main_actor_id),
  CONSTRAINT fk_company FOREIGN KEY  (company_id) REFERENCES companies(company_id)
);
```

Naming Conventions

Tables

- Use a collective name or, less ideally, a plural form:

  `staff`, `employees`, `films`

- Do not prefix with `tbl` or any other such descriptive prefix.
- Never give a table the same name as one of its columns.

Columns

- Always use the singular name.
- Where possible avoid simply using `id` as the primary identifier for the table.
- Do not add a column with the same name as its table.
- Always use lowercase except where it may make sense not to such as proper nouns.

SET SERVEROUTPUT ON

Whenever you start PL/SQL you have to write the "SET SERVEROUTPUT ON" command at the beginning.

PL/SQL program execution in the Oracle engine. Thus, we always required to get serveroutput result and display into the screen otherwise result can't be display. Especially, when we work with blocks, functions and procedures.

Example:

```
SET SERVEROUTPUT ON
-- creating a new procedure

DECLARE
    student_id number(9) NOT NULL := 32765283;
    student_name varchar2(20) := 'Brad Kingston';
    student_faculty CONSTANT varchar2(20) := 'Computer science';
BEGIN
    dbms_output.put_line('Student information:');
    dbms_output.put_line(' - Number: ' || student_id );
    dbms_output.put_line(' - Name: ' || student_name);
    dbms_output.put_line(' - Faculty: ' || student_faculty);
END;
```

Output:

```
Student information:
 - Number: 32765283
 - Name: Brad Kingston
 - Faculty: Computer science


PL/SQL procedure successfully completed.
```

SQL FOREIGN KEY on CREATE TABLE

constraint definition

PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

MySQL: 1

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    FOREIGN KEY (PersonID) REFERENCES Persons(PersonID)
);
```

SQL Server / Oracle / MS Access: 2

```
CREATE TABLE Orders (
    OrderID int NOT NULL PRIMARY KEY,
    OrderNumber int NOT NULL,
    PersonID int FOREIGN KEY REFERENCES Persons(PersonID)
);
```

MySQL / SQL Server / Oracle / MS Access: 3

```
CREATE TABLE Orders (
    OrderID int NOT NULL,
    OrderNumber int NOT NULL,
    PersonID int,
    PRIMARY KEY (OrderID),
    CONSTRAINT FK_PersonOrder FOREIGN KEY (PersonID)
    REFERENCES Persons(PersonID)
);
```

Foreign Key Definition Steps

```
CREATE TABLE directors (
  director_id numeric(10) NOT NULL,
  director_first_name VARCHAR2(20),
  director_last_name VARCHAR(20) NOT NULL,
  director_BD Date,
  country VARCHAR(20) NOT NULL,
  CONSTRAINT directors_pk PRIMARY KEY(director_id)
  );
```

step 1

step 2

```
CREATE TABLE films(
  film_id int NOT NULL,
  title char(35) NOT NULL,
  year DATE NOT NULL,
  language char(20),
  genre char(20) NOT NULL,
  director_id numeric(10) NOT NULL,
  main_actor_id int NOT NULL,
  company_id Number(6) NOT NULL,
  CONSTRAINT films_pk PRIMARY KEY  (film_id),
  CONSTRAINT fk_directors FOREIGN KEY (director_id) REFERENCES directors(director_id),
  CONSTRAINT fk_actor FOREIGN KEY  (main_actor_id) REFERENCES main_actors(main_actor_id),
  CONSTRAINT fk_company FOREIGN KEY  (company_id) REFERENCES companies(company_id)
);
```

step 3

step 4

Foreign Key-Example

Example 1:

```
CREATE TABLE professors(
professor_id numeric (20) not null,
professor_name varchar2(75) not null,
faculty_name varchar2(75),
CONSTRAINT prof_pk PRIMARY KEY (professor_id)
);


CREATE TABLE uni_courses(
uni_course_id numeric (20) not null,
professor_id numeric (20)   not null,
CONSTRAINT fk_professors
FOREIGN KEY (professor_id)
REFERENCES professors (professor_id)
);

 --------------------------
INSERT INTO uni_courses
(uni_course_id, professor_id)
VALUES (250, 600) ;
```

Output 1:

```
Table PROFESSORS created.


Table UNI_COURSES created.


Error starting at line : 19 in command -
INSERT INTO uni_courses
(uni_course_id, professor_id)
VALUES (250, 600)
Error report -
ORA-02291: integrity constraint (DB2019_USER1.FK_PROFESSORS) violated - parent key not found
```

WHY???

The professor_id value of 600 does not already occur in the professors table. Thus, you have to go back and insert the following statement into the professors table:

Correction for example 1:

```
INSERT INTO professors
(professor_id, professor_name, faculty_name)
VALUES (600,'John Smith','Math');
```

DROP TABLE-Foreign Key

When you try to drop a table with unique or primary keys referenced by foreign keys in another table....

Example 2:

```
CREATE TABLE agents (
  AGENT_CODE varchar2(6) NOT NULL,
  AGENT_NAME varchar2(40) NOT NULL,
  WORKING_AREA varchar2(35),
  COMMISSION number(8,2),
  PHONE_NO varchar2(15),
  COUNTRY varchar2(25),
  CONSTRAINT agents_pk PRIMARY KEY (AGENT_CODE)
) ;

CREATE TABLE agents_companies (
  COMPANY_ID varchar2(6) NOT NULL ,
  COMPANY_NAME varchar2(25),
  COMPANY_CITY varchar2(25),
  constraint agents_companies_pk PRIMARY KEY (COMPANY_ID)
) ;

DROP TABLE agents ;
```

Output 2:

```
DROP TABLE agents
Error report -
ORA-02449: unique/primary keys in table referenced by foreign keys
02449. 00000 -  "unique/primary keys in table referenced by foreign keys"
*Cause:    An attempt was made to drop a table with unique or
           primary keys referenced by foreign keys in another table.
*Action:   Before performing the above operations the table, drop the
           foreign key constraints in other tables. You can see what
           constraints are referencing a table by issuing the following
           command:
           SELECT * FROM USER_CONSTRAINTS WHERE TABLE_NAME = "tabnam";
```

Oracle does not let you to drop a table, which is referenced by foreign keys of other tables without specifying the CASCADE CONSTRAINTS

Correction for example 2:

```
DROP TABLE agents CASCADE CONSTRAINTS;
```

Sequences = Autonumber

- You can create an autonumber field by using sequences.
- A sequence is an object in Oracle that is used to generate a number sequence.
- This can be useful when you need to create a unique number to act as a primary key.

Syntax:

```
CREATE SEQUENCE sequence_name
  MINVALUE value
  MAXVALUE value
  START WITH value
  INCREMENT BY value
  CACHE value;
```

Drop…

```
DROP SEQUENCE sequence_name;
```

Example 1:

```
-- Make a sequance in Oracle
DROP SEQUENCE customers_seq;

CREATE SEQUENCE customers_seq
  MINVALUE 1
  MAXVALUE 99
  START WITH 1
  INCREMENT BY 1
  CACHE 20;
```

Output 1:

Script Output

Task completed in 0.503 seconds

```
Sequence CUSTOMERS_SEQ dropped.

Sequence CUSTOMERS_SEQ created.
```

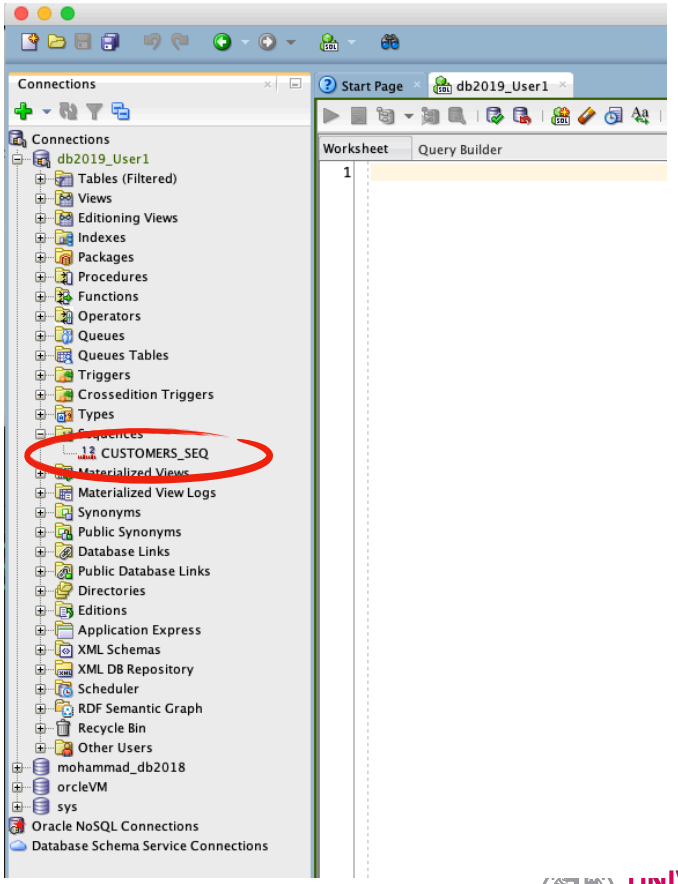
Sequences

example

Example 2:

```
SET SERVEROUTPUT ON;

--Creat a new Table
DROP TABLE customers_test2;

CREATE TABLE customers_test2
( customers_id number(2) NOT NULL,
  customers_name varchar2(50) NOT NULL,
  address varchar2(50),
  city varchar2(10),
  state varchar2(25),
  zip_code varchar2(10),
  CONSTRAINT customers_pk PRIMARY KEY (customers_id)
);

-- Creat a new sequance
DROP SEQUENCE customers_seq;

CREATE SEQUENCE customers_seq
  MINVALUE 1
  MAXVALUE 99
  START WITH 1
  INCREMENT BY 1
  CACHE 20;


INSERT INTO customers_test2
(customers_id, customers_name, address,city,state,zip_code)
VALUES
(customers_seq.NEXTVAL, 'Kraft', 'Av de France, 521', 'Geneva', 'GE', '1201');


INSERT INTO customers_test2
(customers_id, customers_name, address,city,state,zip_code)
VALUES
(customers_seq.NEXTVAL, 'Benjamin', 'Av de Morges, 156', 'Lausanne', 'VD', '1000');
```

file: Sequance_Example.sql



customer_test2:

CUSTOMERS_ID	CUSTOMERS_NAME	ADDRESS	CITY	STATE	ZIP_CODE
1	Kraft	Av de France, 521	Geneva	GE	1201
2	Benjamin	Av de Morges, 156	Lausanne	VD	1000

Check

A check constraint allows you *to specify a condition* on each row in a table.

Syntax:

```
CREATE TABLE table_name
(
  column1 datatype null/not null,
  column2 datatype null/not null,

  ...

  CONSTRAINT constraint_name CHECK (column_name condition)

);
```

Drop…

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_name;
```

Example:

```
CREATE TABLE suppliers
(
  supplier_id numeric(4),
  supplier_name varchar2(50),
  CONSTRAINT check_supplier_id
  CHECK (supplier_id BETWEEN 100 and 9999)
);
```

Average, min, max functions

MAX Function

```sql
SELECT
    segments,
    MAX(unitssold) AS "Hieghest Sale"
FROM
    financial
GROUP BY
    segments
HAVING
    MAX(unitssold) > 4000;
```

SEGMENTS	Hieghest Sale
Channel Partners	4026
Enterprise	4243.5
Government	4492.5

Average Function

```sql
SELECT
    AVG(unitssold) AS "Avrage Sale"
FROM
    financial
WHERE
    unitssold > 4000;
```

Avrage Sale
4246.5

Min, …

Projection & Selection & Joining

- **Projection**: A project operation selects only certain columns from a table.
- **Selection**: A select operation selects a subset of rows in a table that satisfy a selection condition.
- **Joining**: A join operation combines data from two or more tables based on one or more common column values.

(projection)

```
SELECT agent_name, commission FROM agents;
```

	AGENT_NAME	COMMISSION
1	Ramasundar	0.15
2	Alex	0.13
3	Alford	0.12
4	Ravi Kumar	0.15
5	Santakumar	0.14
6	Lucida	0.12
7	Anderson	0.13
8	Subbarao	0.14
9	Mukesh	0.11
10	McDen	0.15
11	Ivan	0.15
12	Benjamin	0.11

(selection)

```
SELECT agent_name,country ,working_area FROM agents WHERE agent_code = 'A003';
```

	AGENT_NAME	COUNTRY	WORKING_AREA
1	Alex	IND	London

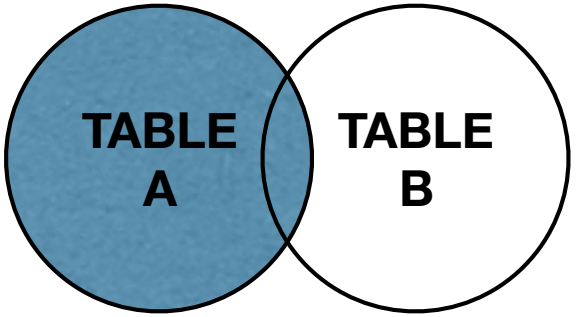
PL/SQL Joins

Oracle PL/SQL JOINS are used to retrieve data from multiple tables.

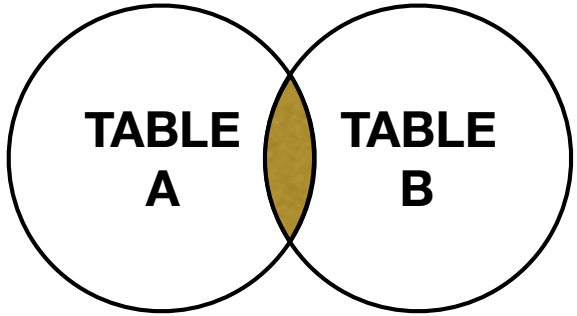
There are 4 different types of Oracle joins:

- Oracle INNER JOIN (or sometimes called simple join)
- Oracle LEFT OUTER JOIN (or sometimes called LEFT JOIN)
- Oracle RIGHT OUTER JOIN (or sometimes called RIGHT JOIN)
- Oracle FULL OUTER JOIN (or sometimes called FULL JOIN)

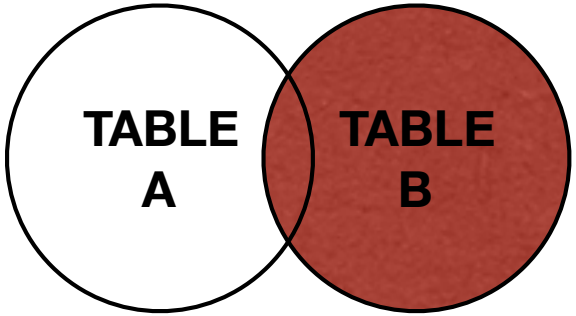
PL/SQL JOINS



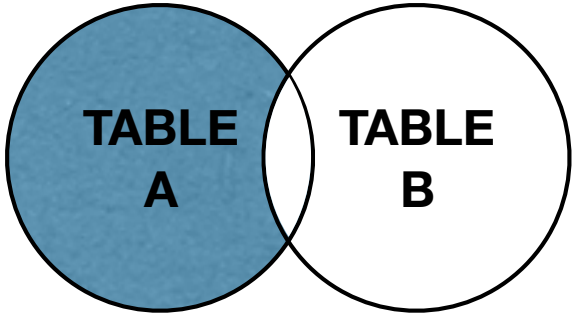

```
SELECT <<list>>
FROM tableA A
LEFT JOIN tableB B
ON A.Key=B.Key
```



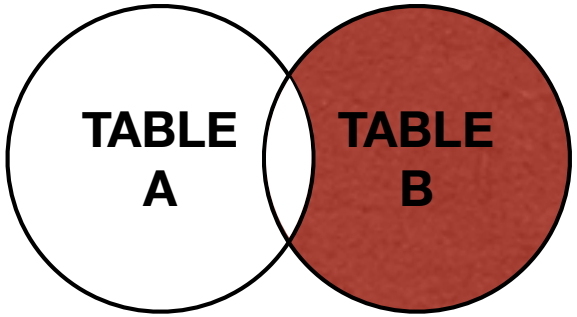

```
SELECT <<list>>
FROM tableA A
INNER JOIN tableB B
ON A.Key=B.Key
```



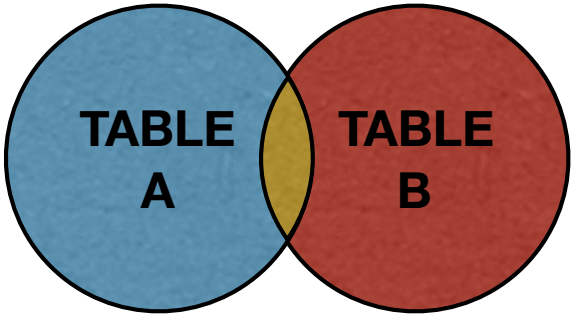

```
SELECT <<list>>
FROM tableA A
RIGHT JOIN tableB B
ON A.Key=B.Key
```



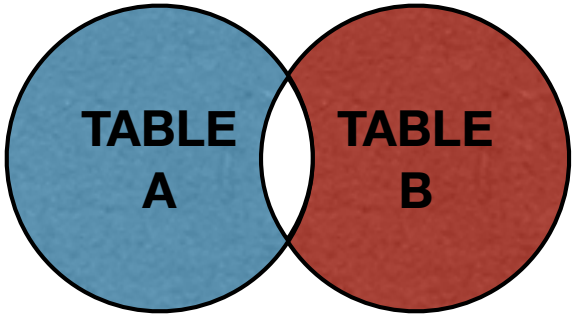

```
SELECT <<list>>
FROM tableA A
LEFT JOIN tableB B
ON A.Key=B.Key
WHERE  B.Key IS NULL
```




```
SELECT <<list>>
FROM tableA A
RIGHT JOIN tableB B
ON A.Key=B.Key
WHERE A.Key IS NULL
```




```
SELECT <<list>>
FROM tableA A
FULL OUTER JOIN tableB B
ON A.Key=B.Key
```



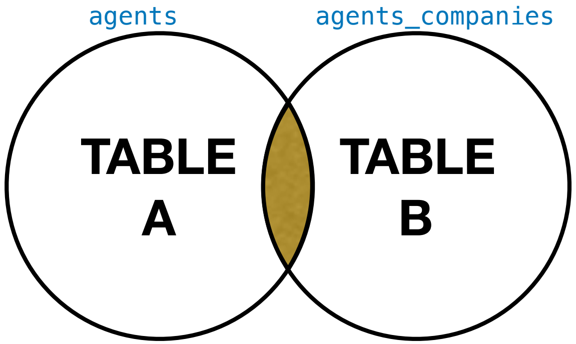

```
SELECT <<list>>
FROM tableA A
FULL OUTER JOIN tableB B
ON A.Key=B.Key
WHERE A.Key IS NULL
OR B.KEY IS NULL
```


UNIVERSITÉ DE GENÈVE
CENTRE UNIVERSITAIRE D'INFORMATIQUE

PL/SQL JOINS-Example

Example 1:

```
SELECT agents.agent_code, agents.agent_name, agents_companies.company_name
FROM agents
INNER JOIN agents_companies
ON AGENTS.COMPANY_ID = AGENTS_COMPANIES.COMPANY_ID;
```



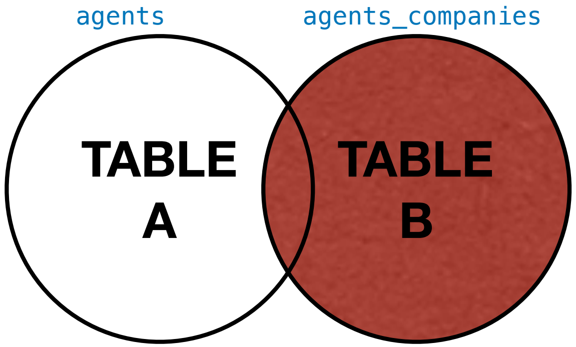
Output 1:

	AGENT_CODE	AGENT_NAME	COMPANY_NAME
1	A004	Ivan	Akas Foods
2	A006	McDen	Akas Foods
3	A002	Mukesh	Akas Foods
4	A010	Santakumar	Akas Foods
5	A003	Alex	Foodies.
6	A009	Benjamin	Foodies.
7	A005	Anderson	Foodies.
8	A001	Subbarao	sip-n-Bite.
9	A011	Ravi Kumar	sip-n-Bite.
10	A008	Alford	sip-n-Bite.
11	A007	Ramasundar	sip-n-Bite.
12	A012	Lucida	sip-n-Bite.

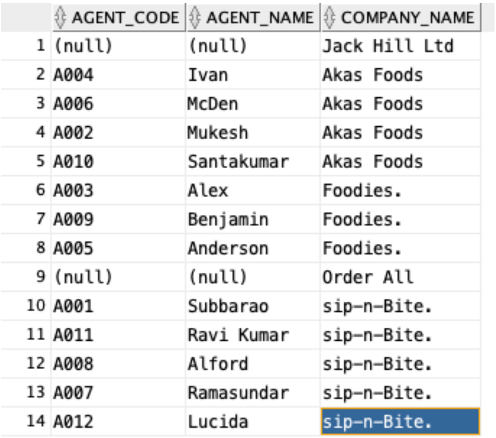
= =

Example 2:

```
SELECT agents.agent_code, agents.agent_name, agents_companies.company_name
FROM agents
RIGHT JOIN agents_companies
ON AGENTS.COMPANY_ID = AGENTS_COMPANIES.COMPANY_ID;
```



Output 2:



	AGENT_CODE	AGENT_NAME	COMPANY_NAME
1	(null)	(null)	Jack Hill Ltd
2	A004	Ivan	Akas Foods
3	A006	McDen	Akas Foods
4	A002	Mukesh	Akas Foods
5	A010	Santakumar	Akas Foods
6	A003	Alex	Foodies.
7	A009	Benjamin	Foodies.
8	A005	Anderson	Foodies.
9	(null)	(null)	Order All
10	A001	Subbarao	sip-n-Bite.
11	A011	Ravi Kumar	sip-n-Bite.
12	A008	Alford	sip-n-Bite.
13	A007	Ramasundar	sip-n-Bite.
14	A012	Lucida	sip-n-Bite.

AND , OR, NOT

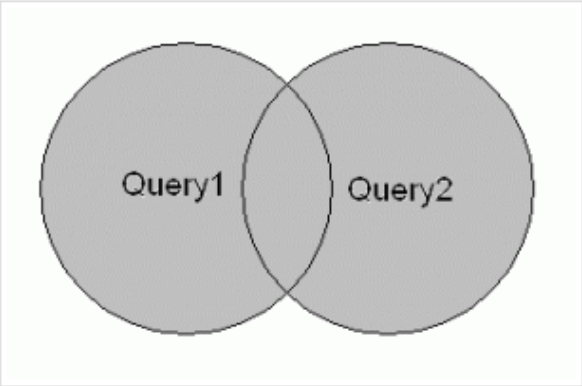
Example:

```
SELECT *
FROM contacts
WHERE last_name = 'Smith'
AND contact_id >= 1000
AND contact_id <= 2000;
```

SELECT — SET OPERATORS(UNION, UNION ALL, MINUS, INTERSECT)

opérateurs ensemblistes

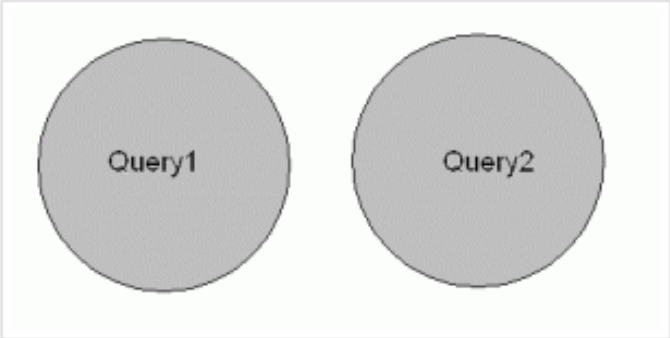
1. UNION




```
SELECT * FROM table1
UNION
SELECT * FROM table2;
```

It is returning unique (distinct) values of both tables.

2. UNION ALL



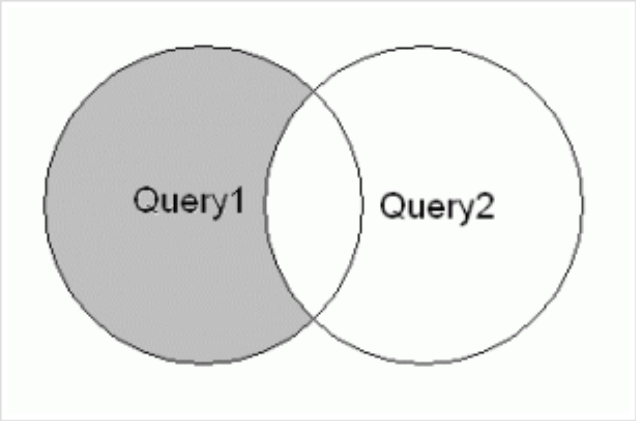

```
SELECT * FROM table1
UNION ALL
SELECT * FROM table2;
```

UNION + duplicated values

SELECT — SET OPERATORS(UNION, UNION ALL, MINUS, INTERSECT)

opérateurs ensemblistes

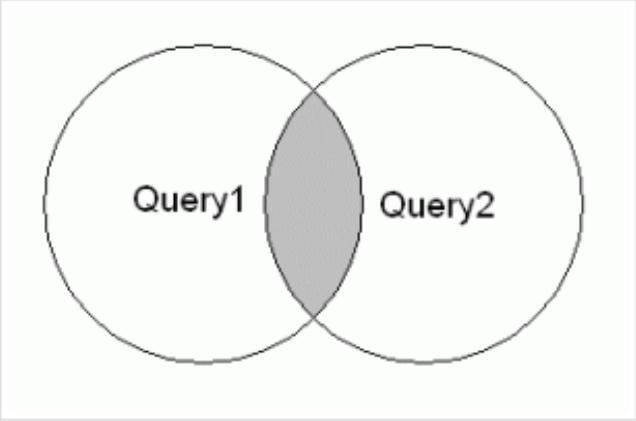
3. MINUS




```
SELECT * FROM table1
MINUS
SELECT * FROM table2;
```

It returns the difference between the first and second SELECT statement.

4. INTERSECT




```
SELECT * FROM table1
INTERSECT
SELECT * FROM table2;
```

INTERSECT is opposite from MINUS as it returns us the results that are both to be found in first and second SELECT statement.

ROWNUM Function

limit the number of returning rows of a query

```
SELECT * FROM (SELECT * FROM films ORDER BY Films.Title) WHERE ROWNUM <= 5;
```

FILM_ID	TITLE	YEAR	LANGUAGE	GENRE	DIRECTOR_ID	MAIN_ACTOR_ID	COMPANY_ID
5	All about Eve	01.03.50	English	Drama	2	7	3
9	Avatar	01.03.09	English	Action	5	6	3
4	Cars	01.03.06	(null)	Family	3	2	2
6	Crash	01.03.04	English	Crime	6	4	4
2	Kindergarten Cop.	01.03.90	English	Comedy	1	1	1

VIEW

— A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

Syntax:

```
CREATE VIEW view_name AS
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

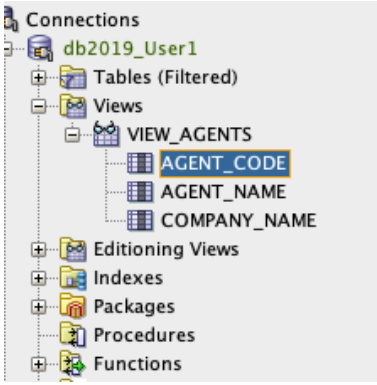
Example:

```
CREATE VIEW [Category Sales For 1997] AS
SELECT DISTINCT CategoryName, Sum(ProductSales) AS CategorySales
FROM [Product Sales for 1997]
GROUP BY CategoryName;
```

VIEW

Example:

```
CREATE VIEW view_agents AS
SELECT agents.agent_code, agents.agent_name, agents_companies.company_name
FROM agents
RIGHT JOIN agents_companies
ON AGENTS.COMPANY_ID = AGENTS_COMPANIES.COMPANY_ID;
```



	AGENT_CODE	AGENT_NAME	COMPANY_NAME
1	(null)	(null)	Jack Hill Ltd
2	A004	Ivan	Akas Foods
3	A006	McDen	Akas Foods
4	A002	Mukesh	Akas Foods
5	A010	Santakumar	Akas Foods
6	A003	Alex	Foodies.
7	A009	Benjamin	Foodies.
8	A005	Anderson	Foodies.
9	(null)	(null)	Order All
10	A001	Subbarao	sip-n-Bite.
11	A011	Ravi Kumar	sip-n-Bite.
12	A008	Alford	sip-n-Bite.
13	A007	Ramasundar	sip-n-Bite.
14	A012	Lucida	sip-n-Bite.

```
SELECT agent_name FROM VIEW_AGENTS
WHERE AGENT_CODE='A003' or AGENT_CODE='A007';
```

Output:

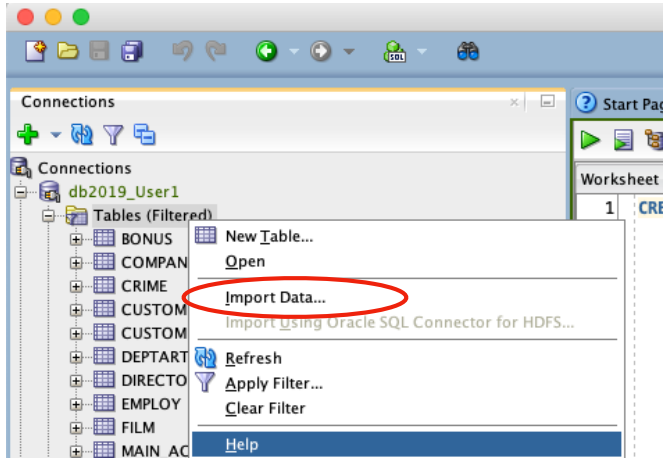
	AGENT_NAME
1	Alex
2	Ramasundar

TABLES From Excel to SQL developer

1. The first thing you need to do: save the Excel spreadsheet as a .csv file.
2. Then import .csv file directly to Oracle SQL developer.
3. Change the columns name.

Financial.xlsx

file: CSV-Financial.csv



Subqueries

- **WHERE clause**

Most common subqueries are be found in the WHERE clause. These subqueries are also called **nested subqueries**.

Example 1:

```
SET SERVEROUTPUT ON

SELECT *
FROM financial
WHERE FINANCIAL.UNITSSOLD IN (
                SELECT   FINANCIAL.UNITSSOLD
                     FROM financial
                    WHERE FINANCIAL.UNITSSOLD>4400);
```

Output 1:

	SEGMENTS	COUNTRY	PRODUCT	DISCOUNTBAND	UNITSSOLD	MAANUFACTURINGPRICE	SALEPRICE	GROSSSALES	DISCOUNTS	SALES	COGS	PROFIT	DATEINFO	MONTHNUMBER	MONTHNAME	YEAR
1	Government	United States of America	Paseo	Low	4492.5	$10.00	$7.00	$31'447.50	$314.48	$31'133.03	$22'462.50	$8'670.53	01.04.14	4	April	2014

- **FROM clause**

Example 2:

QUERY: total number of movies for each director …

```
SELECT
    director.director_first_name,
    director.director_last_name,
    subquery1.directed_movie_total
FROM
    director,
    (
        SELECT
            film.director_id,
            COUNT(*) AS directed_movie_total
        FROM
            film
        GROUP BY
            film.director_id
    ) subquery1
WHERE
    subquery1.director_id = director.director_id;
```

	DIRECTOR_ID	DIRECTED_MOVIE_TOTAL
1	1	1
2	6	1
3	2	1
4	5	3
5	8	1
6	7	1
7	3	1
8	9	3
9	10	1

Output 2:

	DIRECTOR_FIRST_NAME	DIRECTOR_LAST_NAME	DIRECTED_MOVIE_TOTAL
1	Ivan	Reitman	1
2	Joseph	Mankiewicz	1
3	John	Lasseter	1
4	James	Cameron	3
5	Paul	Haggis	1
6	Brad	Bird	1
7	Henry	Hobson	1
8	Steven	Soderbergh	3
9	Jason	Reitman	1

Query Combo

Tables: financial (segements, unitsold, ….)

Question: Find the segment with highest total amount of unitsold…. ?

.sql code:

```sql
-- First you need to insert the financial table into your databse
SET SERVEROUTPUT ON

SELECT
    query1.*
FROM
    (
        SELECT
            segments,
            SUM(financial.unitssold) AS total_amt
        FROM
            financial
        GROUP BY
            financial.segments
    ) query1,
    (
        SELECT
            MAX(query2.total_amt) AS highest_amt
        FROM
            (
                SELECT
                    segments,
                    SUM(financial.unitssold) AS total_amt
                FROM
                    financial
                GROUP BY
                    financial.segments
            ) query2
    ) query3
WHERE
    query1.total_amt = query3.highest_amt;
```

output:

SEGMENTS	TOTAL_AMT
Government	470673.5

file: ComboQuery.sql

Thank you!