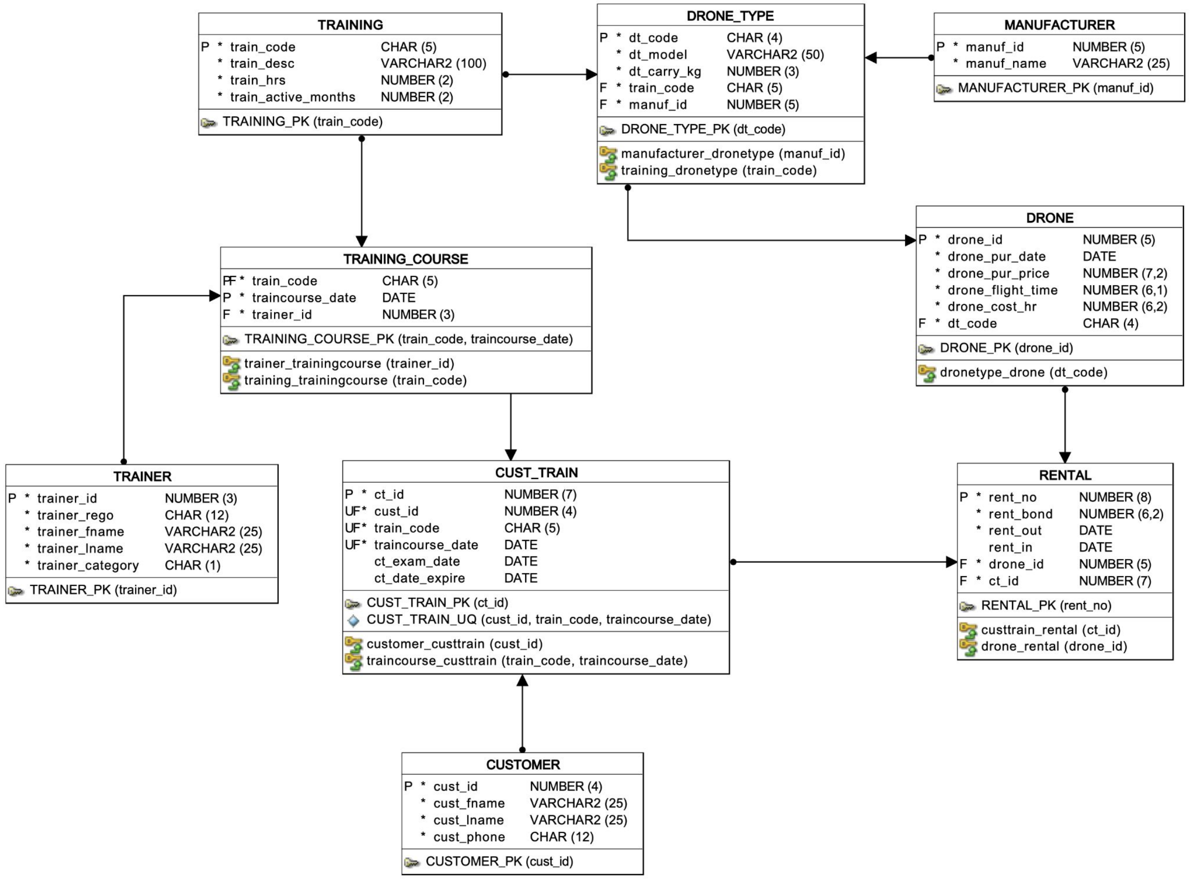
MONASH
INFORMATION
TECHNOLOGY

# 10 - SQL Advanced

Workshop Q&A 2021S2

## Outline

- **CASE**
- Subquery – nested, inline, correlated
- Views
- Joins - self join, outer join
- Set Operators
- Oracle Functions

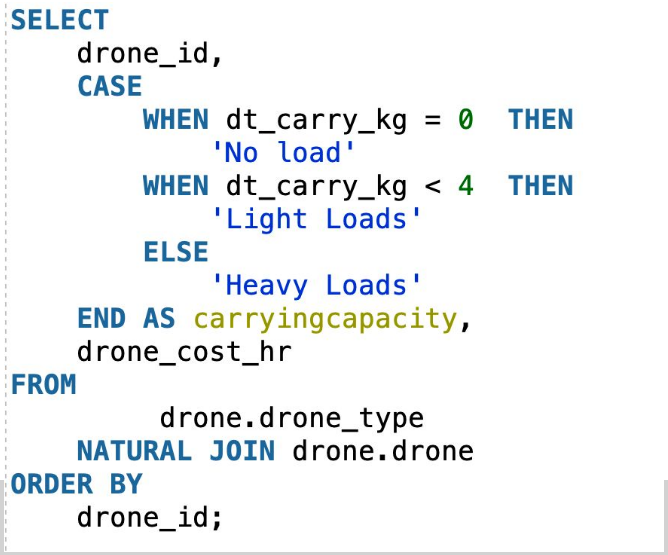


## List the drone id, carry capacity and hire cost per hour for all drones

## SQL CASE statement

The CASE statement used in the select list enables a query to evaluate an attribute and output a particular value based on that evaluation.

Drones which can carry objects have been classified by HyFlying as light carriers for carrying less than 4 Kg, heavy carriers for 4 Kg and greater. Display all drones and their carrying capacity classification as either ‘No load’, ‘Light Loads’ or ‘Heavy Loads’ :



```sql
SELECT
    drone_id,
    CASE
        WHEN dt_carry_kg = 0  THEN
            'No load'
        WHEN dt_carry_kg < 4  THEN
            'Light Loads'
        ELSE
            'Heavy Loads'
    END AS carryingcapacity,
    drone_cost_hr
FROM
         drone.drone_type
    NATURAL JOIN drone.drone
ORDER BY
    drone_id;
```

## Outline

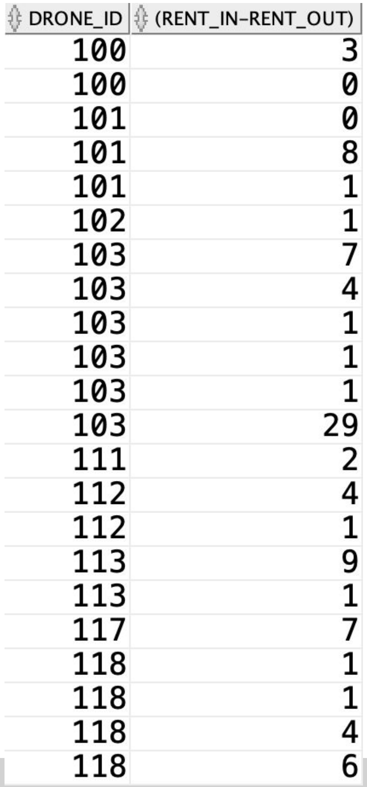
- Case
- **Subquery – nested, inline, correlated**
- Views
- Joins - self join, outer join
- Set Operators
- Oracle Functions

## Query

For each drone find the customers (cust_id only) who rented the drone for the highest number of days. Include the drone id, number of rented days and customer id in the output.

## For each completed rental list the number of days the drone is out

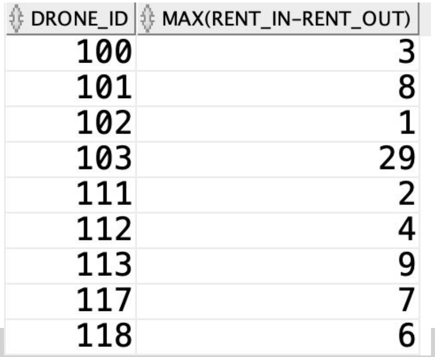
```
SELECT
    drone_id,
    ( rent_in - rent_out )
FROM
    drone.rental
WHERE
    rent_in IS NOT NULL
ORDER BY
    drone_id;
```



| DRONE_ID | (RENT_IN-RENT_OUT) |
|---|---|
| 100 | 3 |
| 100 | 0 |
| 101 | 0 |
| 101 | 8 |
| 101 | 1 |
| 102 | 1 |
| 103 | 7 |
| 103 | 4 |
| 103 | 1 |
| 103 | 1 |
| 103 | 1 |
| 103 | 29 |
| 111 | 2 |
| 112 | 4 |
| 112 | 1 |
| 113 | 9 |
| 113 | 1 |
| 117 | 7 |
| 118 | 1 |
| 118 | 1 |
| 118 | 4 |
| 118 | 6 |

## For a given drone list the maximum number of days the drone was out

```
SELECT
    drone_id,
    MAX(rent_in - rent_out)
FROM
    drone.rental
WHERE
    rent_in IS NOT NULL
GROUP BY
    drone_id
ORDER BY
    drone_id;
```

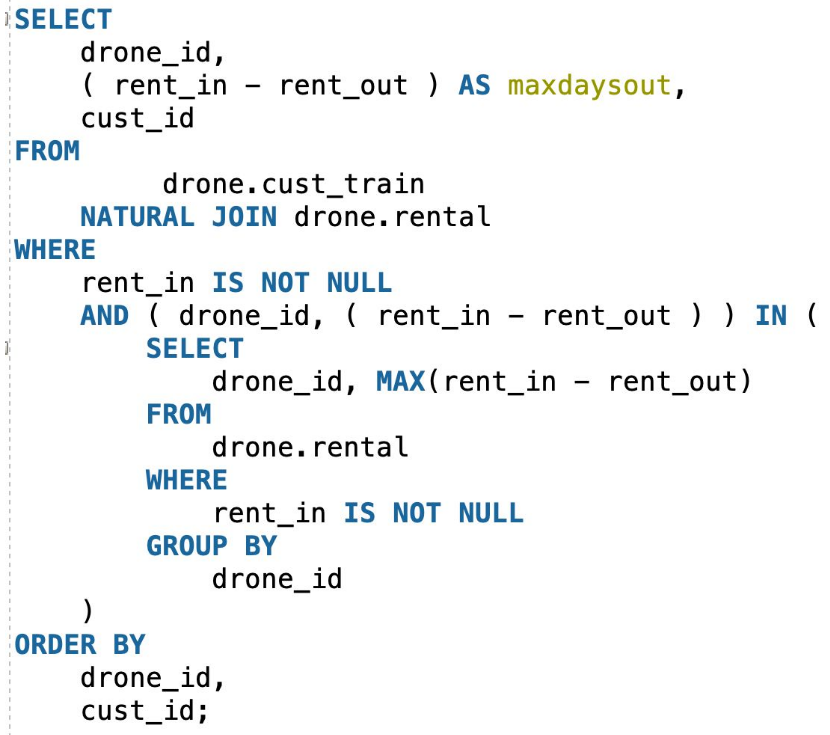


| DRONE_ID | MAX(RENT_IN-RENT_OUT) |
|---|---|
| 100 | 3 |
| 101 | 8 |
| 102 | 1 |
| 103 | 29 |
| 111 | 2 |
| 112 | 4 |
| 113 | 9 |
| 117 | 7 |
| 118 | 6 |

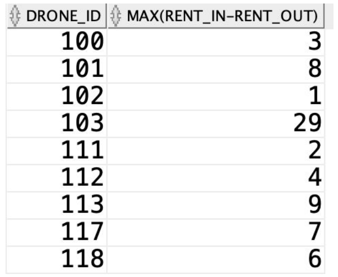
MONASH University

## Subquery (NESTED)

- The subquery is independent of the outer query and is executed only once.



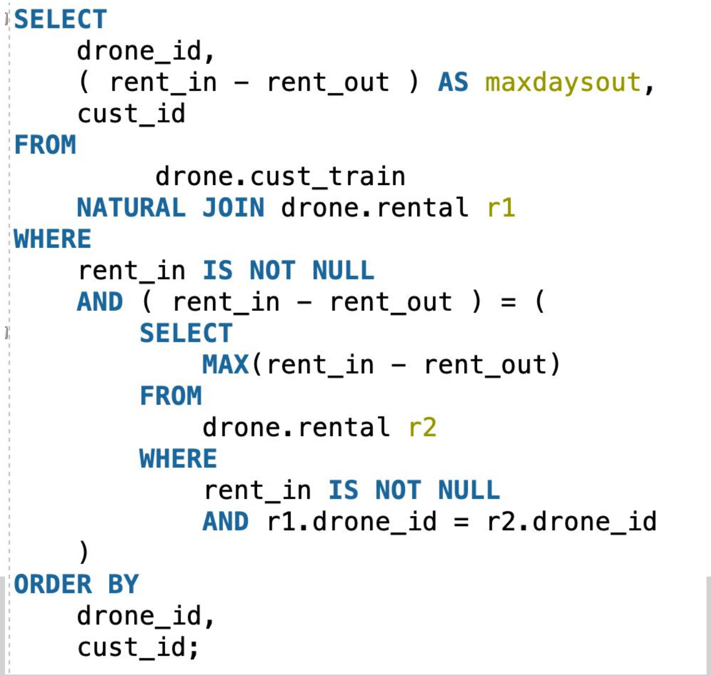
```
SELECT
    drone_id,
    ( rent_in - rent_out ) AS maxdaysout,
    cust_id
FROM
         drone.cust_train
    NATURAL JOIN drone.rental
WHERE
    rent_in IS NOT NULL
    AND ( drone_id, ( rent_in - rent_out ) ) IN (
        SELECT
            drone_id, MAX(rent_in - rent_out)
        FROM
            drone.rental
        WHERE
            rent_in IS NOT NULL
        GROUP BY
            drone_id
    )
ORDER BY
    drone_id,
    cust_id;
```



| DRONE_ID | MAX(RENT_IN-RENT_OUT) |
|---|---|
| 100 | 3 |
| 101 | 8 |
| 102 | 1 |
| 103 | 29 |
| 111 | 2 |
| 112 | 4 |
| 113 | 9 |
| 117 | 7 |
| 118 | 6 |

## Subquery (CORRELATED)

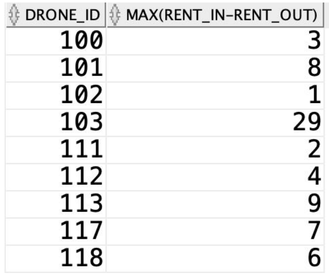
- the subquery is related to the outer query and is ***evaluated once for each row of the outer query***
- correlated subqueries can also be used within update statements
  - outer update occurs based on value returned from subquery



```sql
SELECT
    drone_id,
    ( rent_in - rent_out ) AS maxdaysout,
    cust_id
FROM
         drone.cust_train
    NATURAL JOIN drone.rental r1
WHERE
    rent_in IS NOT NULL
    AND ( rent_in - rent_out ) = (
        SELECT
            MAX(rent_in - rent_out)
        FROM
            drone.rental r2
        WHERE
            rent_in IS NOT NULL
            AND r1.drone_id = r2.drone_id
    )
ORDER BY
    drone_id,
    cust_id;
```

# Subquery (INLINE) – Derived table

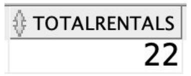
```
SELECT
    rental.drone_id,
    ( rent_in - rent_out ) AS maxdaysout,
    cust_id
FROM
    (
            (
            SELECT
                drone_id,
                MAX(rent_in - rent_out) AS maxout
            FROM
                drone.rental
            WHERE
                rent_in IS NOT NULL
            GROUP BY
                drone_id
        ) maxtable
        JOIN drone.rental
        ON maxtable.drone_id = rental.drone_id
           AND ( rent_in - rent_out ) = maxtable.maxout
    )
    JOIN drone.cust_train
    USING ( ct_id )
ORDER BY
    drone_id,
    cust_id;
```



| DRONE_ID | MAX(RENT_IN-RENT_OUT) |
|---|---|
| 100 | 3 |
| 101 | 8 |
| 102 | 1 |
| 103 | 29 |
| 111 | 2 |
| 112 | 4 |
| 113 | 9 |
| 117 | 7 |
| 118 | 6 |

How many completed rentals have been recorded?

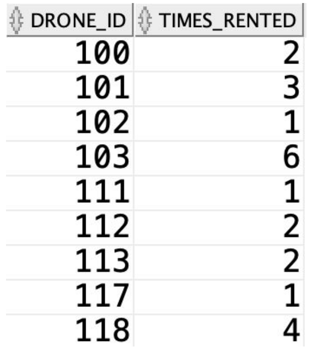
```
SELECT
        COUNT(*) AS totalrentals
FROM
        drone.rental
WHERE
        rent_in IS NOT NULL;
```



| TOTALRENTALS |
|---|
| 22 |

List for each drone the number of times the drone has been rented in a completed rental

```
SELECT
    drone_id,
    COUNT(*) AS times_rented
FROM
    drone.rental
WHERE
    rent_in IS NOT NULL
GROUP BY
    drone_id
ORDER BY
    drone_id;
```

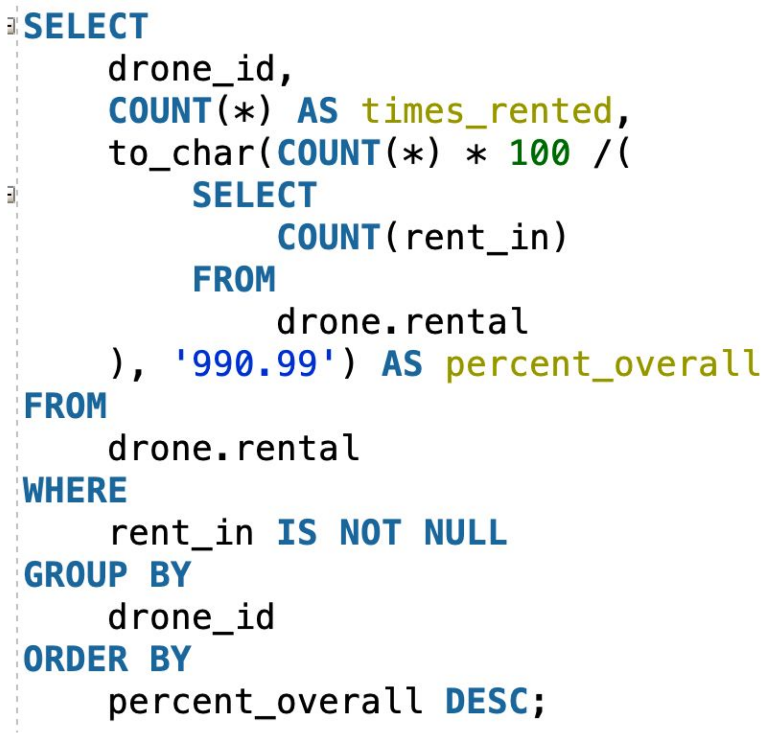


| DRONE_ID | TIMES_RENTED |
|---|---|
| 100 | 2 |
| 101 | 3 |
| 102 | 1 |
| 103 | 6 |
| 111 | 1 |
| 112 | 2 |
| 113 | 2 |
| 117 | 1 |
| 118 | 4 |

For each drone compute the percentage of the company's rentals contributed by that drone

MONASH University

## Subquery (INLINE)



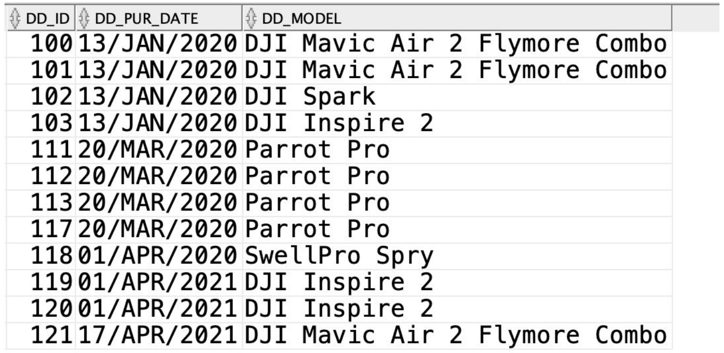
```
SELECT
    drone_id,
    COUNT(*) AS times_rented,
    to_char(COUNT(*) * 100 /(
        SELECT
            COUNT(rent_in)
        FROM
            drone.rental
    ), '990.99') AS percent_overall
FROM
    drone.rental
WHERE
    rent_in IS NOT NULL
GROUP BY
    drone_id
ORDER BY
    percent_overall DESC;
```

# Use of subquery in INSERT

```
CREATE TABLE drone_details (
    dd_id         NUMBER(5) NOT NULL,
    dd_pur_date   DATE NOT NULL,
    dd_model      VARCHAR2(50) NOT NULL,
    CONSTRAINT drone_details_pk PRIMARY KEY ( dd_id )
);
```

*Assume table created*

```
INSERT INTO drone_details
    ( SELECT
          drone_id,
          drone_pur_date,
          dt_model
    FROM
               drone.drone
          NATURAL JOIN drone.drone_type
    );
```



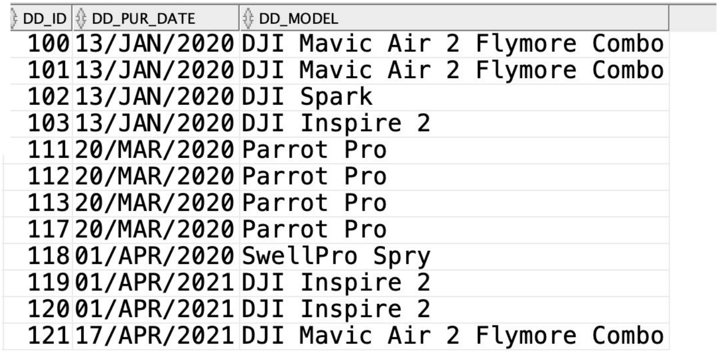
| DD_ID | DD_PUR_DATE | DD_MODEL |
|---|---|---|
| 100 | 13/JAN/2020 | DJI Mavic Air 2 Flymore Combo |
| 101 | 13/JAN/2020 | DJI Mavic Air 2 Flymore Combo |
| 102 | 13/JAN/2020 | DJI Spark |
| 103 | 13/JAN/2020 | DJI Inspire 2 |
| 111 | 20/MAR/2020 | Parrot Pro |
| 112 | 20/MAR/2020 | Parrot Pro |
| 113 | 20/MAR/2020 | Parrot Pro |
| 117 | 20/MAR/2020 | Parrot Pro |
| 118 | 01/APR/2020 | SwellPro Spry |
| 119 | 01/APR/2021 | DJI Inspire 2 |
| 120 | 01/APR/2021 | DJI Inspire 2 |
| 121 | 17/APR/2021 | DJI Mavic Air 2 Flymore Combo |

***If you need to both create and insert the data, is there a simpler way to achieve these two tasks?***

MONASH University

## Simpler approach (using week 7 tutorial approach 7.3.4)

```
CREATE TABLE drone_details
    AS
        ( SELECT
            drone_id,
            drone_pur_date,
            dt_model
        FROM
                drone.drone
            NATURAL JOIN drone.drone_type
        );
```



| DD_ID | DD_PUR_DATE | DD_MODEL |
|---|---|---|
| 100 | 13/JAN/2020 | DJI Mavic Air 2 Flymore Combo |
| 101 | 13/JAN/2020 | DJI Mavic Air 2 Flymore Combo |
| 102 | 13/JAN/2020 | DJI Spark |
| 103 | 13/JAN/2020 | DJI Inspire 2 |
| 111 | 20/MAR/2020 | Parrot Pro |
| 112 | 20/MAR/2020 | Parrot Pro |
| 113 | 20/MAR/2020 | Parrot Pro |
| 117 | 20/MAR/2020 | Parrot Pro |
| 118 | 01/APR/2020 | SwellPro Spry |
| 119 | 01/APR/2021 | DJI Inspire 2 |
| 120 | 01/APR/2021 | DJI Inspire 2 |
| 121 | 17/APR/2021 | DJI Mavic Air 2 Flymore Combo |

## Outline

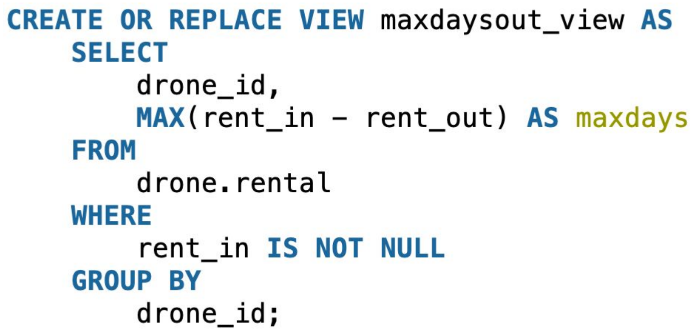
- Case
- Subquery – nested, inline, correlated
- **Views**
- Joins - self join, outer join
- Set Operators
- Oracle Functions

## Views

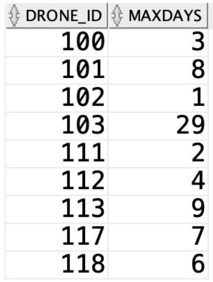
- A virtual table derived from one or more base tables.
- Sometimes used as "Access Control" to the database

**CREATE OR REPLACE VIEW [view_name] AS**

**SELECT … ;**



```
CREATE OR REPLACE VIEW maxdaysout_view AS
    SELECT
        drone_id,
        MAX(rent_in - rent_out) AS maxdays
    FROM
        drone.rental
    WHERE
        rent_in IS NOT NULL
    GROUP BY
        drone_id;
```



| DRONE_ID | MAXDAYS |
|---|---|
| 100 | 3 |
| 101 | 8 |
| 102 | 1 |
| 103 | 29 |
| 111 | 2 |
| 112 | 4 |
| 113 | 9 |
| 117 | 7 |
| 118 | 6 |

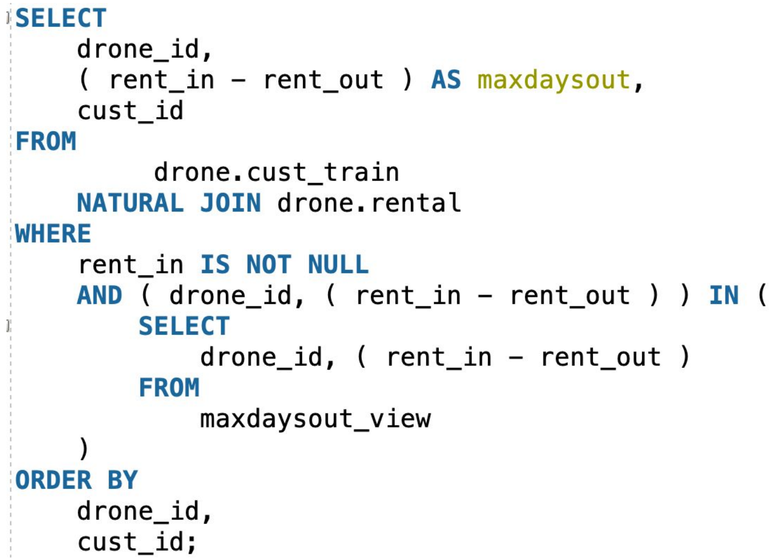
**select * from maxdaysout_view**
**order by drone_id;**

- What objects do I own?

```
select * from user_objects;
```

## Using Views

- For each drone find the customers (cust_id only) who rented the drone for the highest number of days



```
SELECT
    drone_id,
    ( rent_in - rent_out ) AS maxdaysout,
    cust_id
FROM
         drone.cust_train
    NATURAL JOIN drone.rental
WHERE
    rent_in IS NOT NULL
    AND ( drone_id, ( rent_in - rent_out ) ) IN (
        SELECT
            drone_id, ( rent_in - rent_out )
        FROM
            maxdaysout_view
    )
ORDER BY
    drone_id,
    cust_id;
```

**Please note VIEWS <u>MUST NOT</u> be used for Assignment 2A or B**

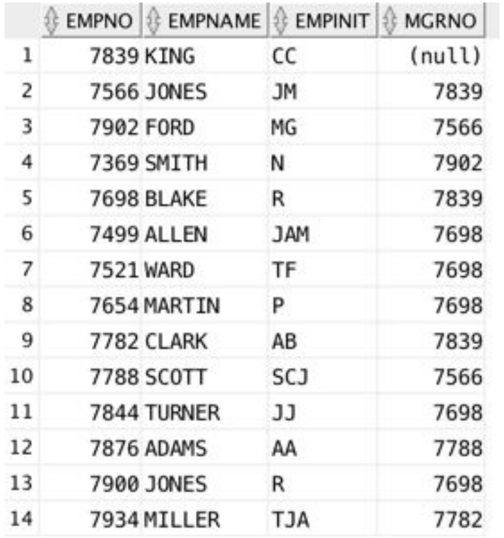
## Outline

- Case
- Subquery – nested, inline, correlated
- Views
- **Joins - self join, outer join**
- Set Operators
- Oracle Functions

## Self Join

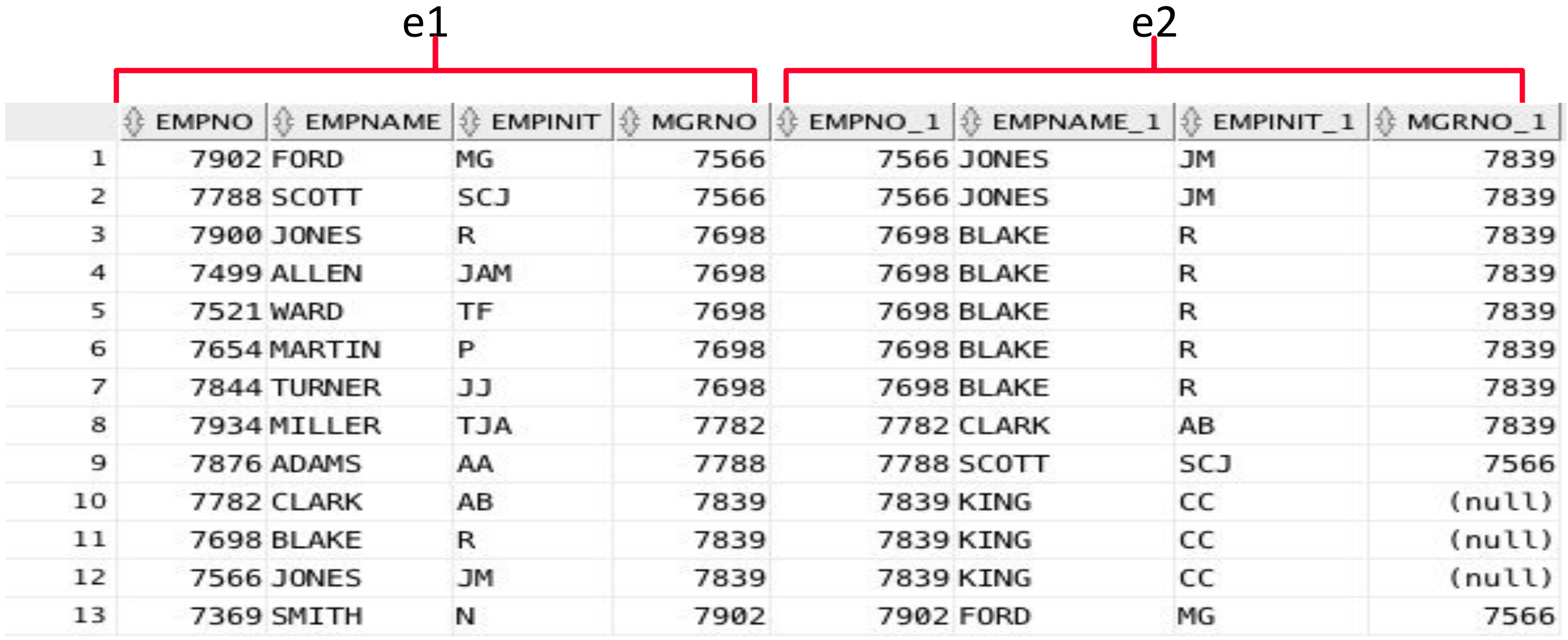
- Show the name of the manager for each employee.

```
SELECT
    empno,
    empname,
    empinit,
    mgrno
FROM
    emp.employee;
```



| | EMPNO | EMPNAME | EMPINIT | MGRNO |
|---|---|---|---|---|
| 1 | 7839 | KING | CC | (null) |
| 2 | 7566 | JONES | JM | 7839 |
| 3 | 7902 | FORD | MG | 7566 |
| 4 | 7369 | SMITH | N | 7902 |
| 5 | 7698 | BLAKE | R | 7839 |
| 6 | 7499 | ALLEN | JAM | 7698 |
| 7 | 7521 | WARD | TF | 7698 |
| 8 | 7654 | MARTIN | P | 7698 |
| 9 | 7782 | CLARK | AB | 7839 |
| 10 | 7788 | SCOTT | SCJ | 7566 |
| 11 | 7844 | TURNER | JJ | 7698 |
| 12 | 7876 | ADAMS | AA | 7788 |
| 13 | 7900 | JONES | R | 7698 |
| 14 | 7934 | MILLER | TJA | 7782 |

**SELECT ***
**FROM emp.employee e1 JOIN emp.employee e2**
**ON e1.mgrno = e2.empno;**



| | EMPNO | EMPNAME | EMPINIT | MGRNO | EMPNO_1 | EMPNAME_1 | EMPINIT_1 | MGRNO_1 |
|---|---|---|---|---|---|---|---|---|
| 1 | 7902 | FORD | MG | 7566 | 7566 | JONES | JM | 7839 |
| 2 | 7788 | SCOTT | SCJ | 7566 | 7566 | JONES | JM | 7839 |
| 3 | 7900 | JONES | R | 7698 | 7698 | BLAKE | R | 7839 |
| 4 | 7499 | ALLEN | JAM | 7698 | 7698 | BLAKE | R | 7839 |
| 5 | 7521 | WARD | TF | 7698 | 7698 | BLAKE | R | 7839 |
| 6 | 7654 | MARTIN | P | 7698 | 7698 | BLAKE | R | 7839 |
| 7 | 7844 | TURNER | JJ | 7698 | 7698 | BLAKE | R | 7839 |
| 8 | 7934 | MILLER | TJA | 7782 | 7782 | CLARK | AB | 7839 |
| 9 | 7876 | ADAMS | AA | 7788 | 7788 | SCOTT | SCJ | 7566 |
| 10 | 7782 | CLARK | AB | 7839 | 7839 | KING | CC | (null) |
| 11 | 7698 | BLAKE | R | 7839 | 7839 | KING | CC | (null) |
| 12 | 7566 | JONES | JM | 7839 | 7839 | KING | CC | (null) |
| 13 | 7369 | SMITH | N | 7902 | 7902 | FORD | MG | 7566 |

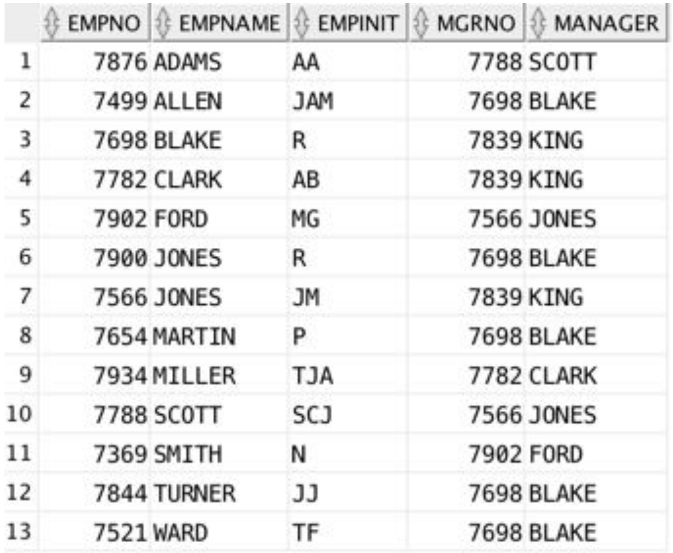
Joined rows
1,12
2,12
3,11

*Note some columns have been hidden*

**Why now only 13 rows?**

MONASH University

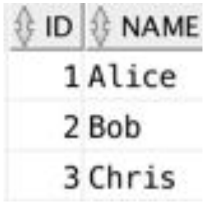
```
SELECT e1.empno, e1.empname, e1.empinit, e1.mgrno,
       e2.empname AS MANAGER
FROM emp.employee e1  JOIN emp.employee e2
       ON e1.mgrno = e2.empno
ORDER BY e1.empname;
```



|  | EMPNO | EMPNAME | EMPINIT | MGRNO | MANAGER |
|---|---|---|---|---|---|
| 1 | 7876 | ADAMS | AA | 7788 | SCOTT |
| 2 | 7499 | ALLEN | JAM | 7698 | BLAKE |
| 3 | 7698 | BLAKE | R | 7839 | KING |
| 4 | 7782 | CLARK | AB | 7839 | KING |
| 5 | 7902 | FORD | MG | 7566 | JONES |
| 6 | 7900 | JONES | R | 7698 | BLAKE |
| 7 | 7566 | JONES | JM | 7839 | KING |
| 8 | 7654 | MARTIN | P | 7698 | BLAKE |
| 9 | 7934 | MILLER | TJA | 7782 | CLARK |
| 10 | 7788 | SCOTT | SCJ | 7566 | JONES |
| 11 | 7369 | SMITH | N | 7902 | FORD |
| 12 | 7844 | TURNER | JJ | 7698 | BLAKE |
| 13 | 7521 | WARD | TF | 7698 | BLAKE |

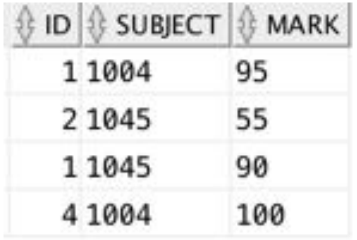
# INNER JOIN

**Student**



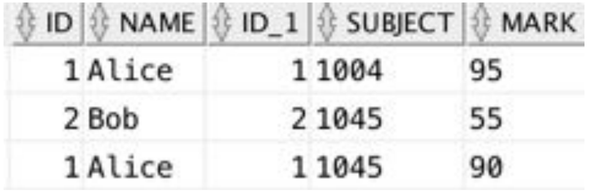
| ID | NAME |
|---|---|
| 1 | Alice |
| 2 | Bob |
| 3 | Chris |

**Mark**



| ID | SUBJECT | MARK |
|---|---|---|
| 1 | 1004 | 95 |
| 2 | 1045 | 55 |
| 1 | 1045 | 90 |
| 4 | 1004 | 100 |

**Inner Join gives no information for Chris and the student with ID 4**

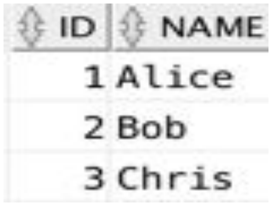


| ID | NAME | ID_1 | SUBJECT | MARK |
|---|---|---|---|---|
| 1 | Alice | 1 | 1004 | 95 |
| 2 | Bob | 2 | 1045 | 55 |
| 1 | Alice | 1 | 1045 | 90 |

**Select * from student s join mark m on s.id = m.id;**
***Note that this is an EQUI JOIN* (an inner join)**

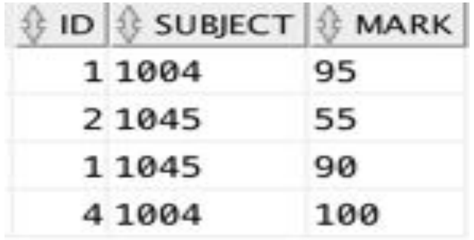
# FULL OUTER JOIN

**Student**



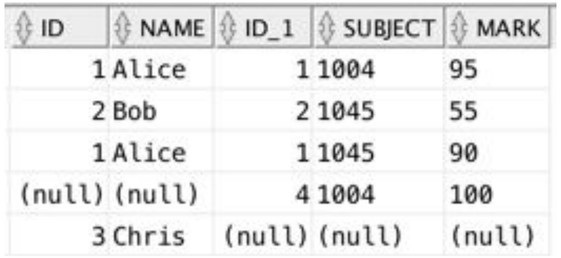
| ID | NAME |
|---|---|
| 1 | Alice |
| 2 | Bob |
| 3 | Chris |

**Mark**



| ID | SUBJECT | MARK |
|---|---|---|
| 1 | 1004 | 95 |
| 2 | 1045 | 55 |
| 1 | 1045 | 90 |
| 4 | 1004 | 100 |

**Get (incomplete) information of both Chris and student with ID 4**

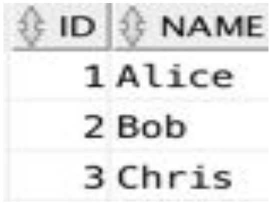


| ID | NAME | ID_1 | SUBJECT | MARK |
|---|---|---|---|---|
| 1 | Alice | 1 | 1004 | 95 |
| 2 | Bob | 2 | 1045 | 55 |
| 1 | Alice | 1 | 1045 | 90 |
| (null) | (null) | 4 | 1004 | 100 |
| 3 | Chris | (null) | (null) | (null) |

**select * from**
**student s full outer join mark m on s.id = m.id;**

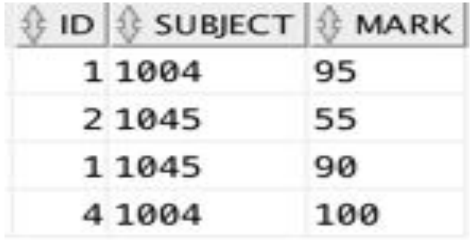
# LEFT OUTER JOIN

**Student**



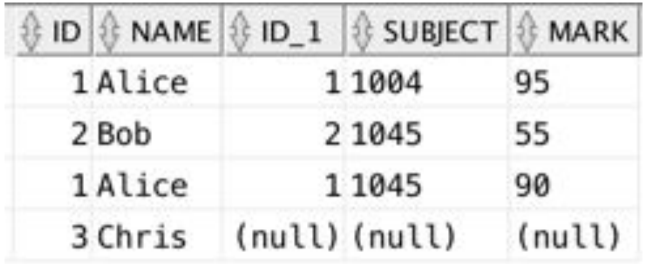
| ID | NAME |
|---|---|
| 1 | Alice |
| 2 | Bob |
| 3 | Chris |

**Mark**



| ID | SUBJECT | MARK |
|---|---|---|
| 1 | 1004 | 95 |
| 2 | 1045 | 55 |
| 1 | 1045 | 90 |
| 4 | 1004 | 100 |

**Get (incomplete) information of only Chris**

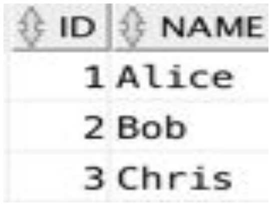


| ID | NAME | ID_1 | SUBJECT | MARK |
|---|---|---|---|---|
| 1 | Alice | 1 | 1004 | 95 |
| 2 | Bob | 2 | 1045 | 55 |
| 1 | Alice | 1 | 1045 | 90 |
| 3 | Chris | (null) | (null) | (null) |

```
select * from
student s left outer join mark m
on s.id = m.id;
```

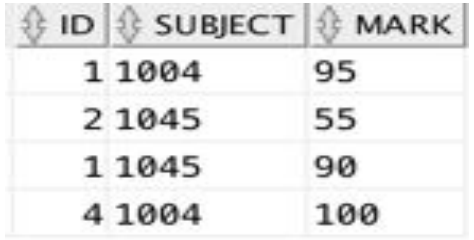
# RIGHT OUTER JOIN

**Student**



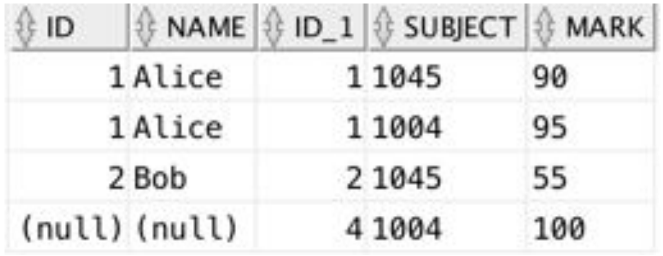
| ID | NAME |
|---|---|
| 1 | Alice |
| 2 | Bob |
| 3 | Chris |

**Mark**



| ID | SUBJECT | MARK |
|---|---|---|
| 1 | 1004 | 95 |
| 2 | 1045 | 55 |
| 1 | 1045 | 90 |
| 4 | 1004 | 100 |

Get (incomplete) information of  the student with ID 4



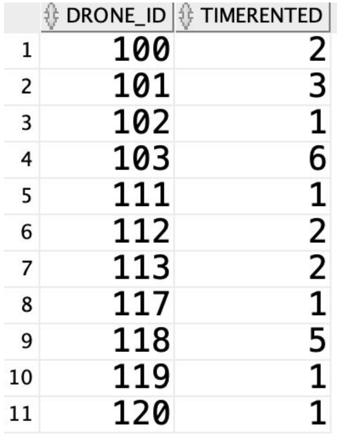
| ID | NAME | ID_1 | SUBJECT | MARK |
|---|---|---|---|---|
| 1 | Alice | 1 | 1045 | 90 |
| 1 | Alice | 1 | 1004 | 95 |
| 2 | Bob | 2 | 1045 | 55 |
| (null) | (null) | 4 | 1004 | 100 |

```
select * from
student s right outer join mark m
on s.id = m.id;
```

## Outer Join

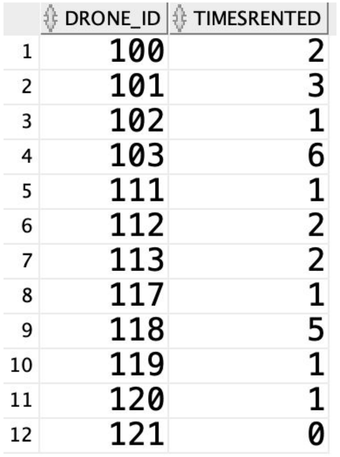
- List the number of times ALL drones have been rented

```
SELECT
    drone_id,
    COUNT(rent_out) as timerented
FROM
        drone.drone
    JOIN drone.rental
    USING ( drone_id )
GROUP BY
    drone_id
ORDER BY
    drone_id;
```



| | DRONE_ID | TIMERENTED |
|---|---|---|
| 1 | 100 | 2 |
| 2 | 101 | 3 |
| 3 | 102 | 1 |
| 4 | 103 | 6 |
| 5 | 111 | 1 |
| 6 | 112 | 2 |
| 7 | 113 | 2 |
| 8 | 117 | 1 |
| 9 | 118 | 5 |
| 10 | 119 | 1 |
| 11 | 120 | 1 |

```
SELECT
    drone_id,
    COUNT(rent_out) as timesrented
FROM
    drone.drone
    LEFT OUTER JOIN drone.rental
    USING ( drone_id )
GROUP BY
    drone_id
ORDER BY
    drone_id;
```



| | DRONE_ID | TIMESRENTED |
|---|---|---|
| 1 | 100 | 2 |
| 2 | 101 | 3 |
| 3 | 102 | 1 |
| 4 | 103 | 6 |
| 5 | 111 | 1 |
| 6 | 112 | 2 |
| 7 | 113 | 2 |
| 8 | 117 | 1 |
| 9 | 118 | 5 |
| 10 | 119 | 1 |
| 11 | 120 | 1 |
| 12 | 121 | 0 |

## Outline

- Case
- Subquery – nested, inline, correlated
- Views
- Joins - self join, outer join
- **Set Operators**
- Oracle Functions

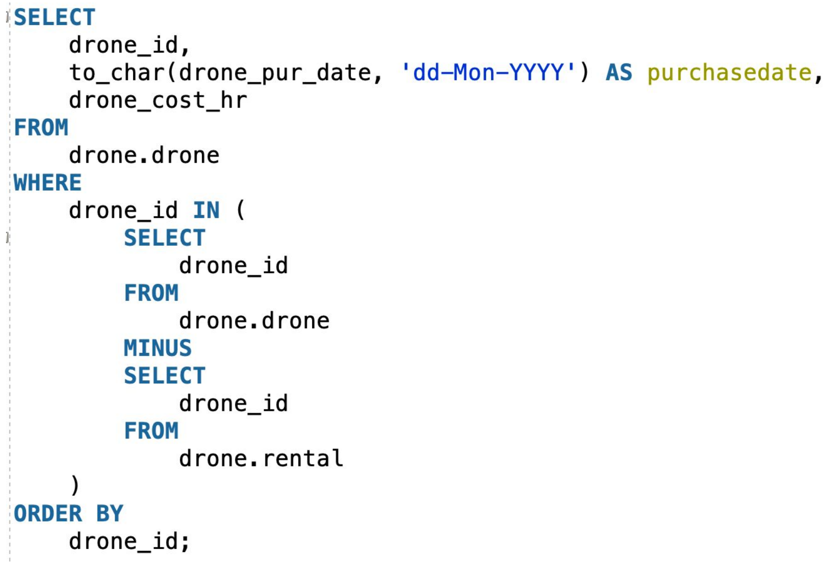
## Relational Set Operators

- Using the set operators you can combine two or more sets to create new sets (relations)
- Union All
  - All rows selected by either query, including all duplicates
- Union
  - All rows selected by either query, removing duplicates (eg. DISTINCT on Union All)
- Intersect
  - All distinct rows selected by both queries
- Minus
  - All distinct rows selected by the first query but not by the second
- All set operators have equal precedence. If a SQL statement contains multiple set operators, Oracle evaluates them from the left to right if no parentheses explicitly specify another order.
- The two sets must be UNION COMPATIBLE (ie. same number of attributes and similar data types)

## MINUS

List the details of drones which have not been rented. Include drone id, drone purchase date and drone cost per hour in the list.

- List the drone id of all drones
- List the drone id of those drones which have been rented

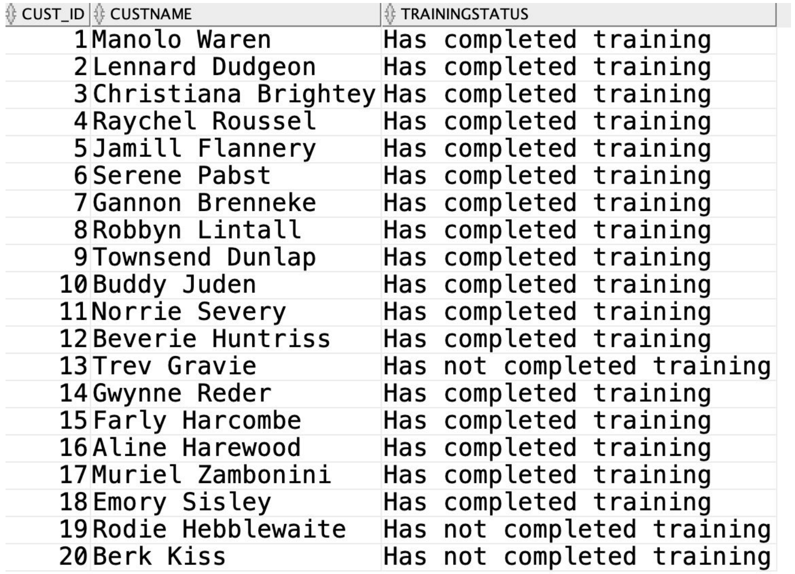


```
SELECT
    drone_id,
    to_char(drone_pur_date, 'dd-Mon-YYYY') AS purchasedate,
    drone_cost_hr
FROM
    drone.drone
WHERE
    drone_id IN (
        SELECT
            drone_id
        FROM
            drone.drone
        MINUS
        SELECT
            drone_id
        FROM
            drone.rental
    )
ORDER BY
    drone_id;
```

MONASH University

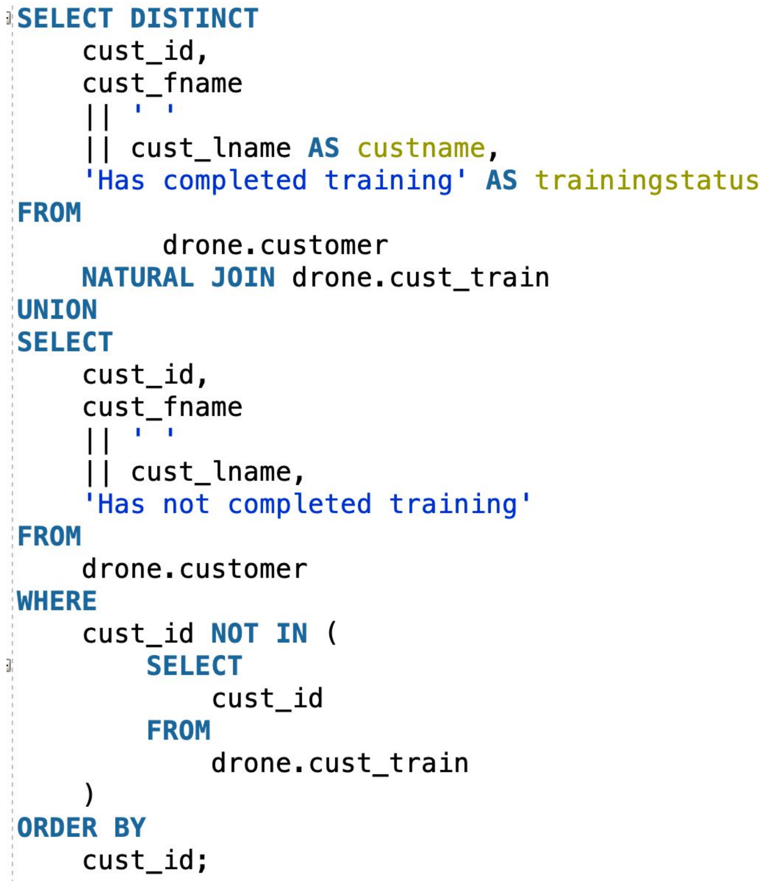
## UNION

- Create a list of all customers:
  - for those who have completed training show "Completed training"
  - for those who have not completed training show "Not completed training"



| CUST_ID | CUSTNAME | TRAININGSTATUS |
|---|---|---|
| 1 | Manolo Waren | Has completed training |
| 2 | Lennard Dudgeon | Has completed training |
| 3 | Christiana Brightey | Has completed training |
| 4 | Raychel Roussel | Has completed training |
| 5 | Jamill Flannery | Has completed training |
| 6 | Serene Pabst | Has completed training |
| 7 | Gannon Brenneke | Has completed training |
| 8 | Robbyn Lintall | Has completed training |
| 9 | Townsend Dunlap | Has completed training |
| 10 | Buddy Juden | Has completed training |
| 11 | Norrie Severy | Has completed training |
| 12 | Beverie Huntriss | Has completed training |
| 13 | Trev Gravie | Has not completed training |
| 14 | Gwynne Reder | Has completed training |
| 15 | Farly Harcombe | Has completed training |
| 16 | Aline Harewood | Has completed training |
| 17 | Muriel Zambonini | Has completed training |
| 18 | Emory Sisley | Has completed training |
| 19 | Rodie Hebblewaite | Has not completed training |
| 20 | Berk Kiss | Has not completed training |

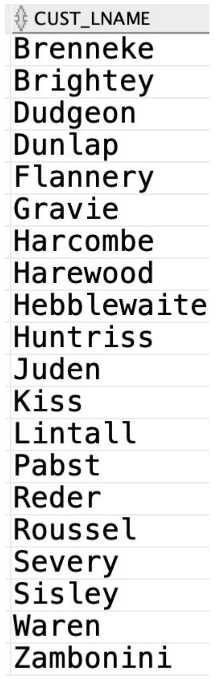
MONASH University



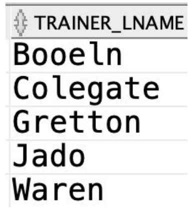
```sql
SELECT DISTINCT
    cust_id,
    cust_fname
    || ' '
    || cust_lname AS custname,
    'Has completed training' AS trainingstatus
FROM
         drone.customer
    NATURAL JOIN drone.cust_train
UNION
SELECT
    cust_id,
    cust_fname
    || ' '
    || cust_lname,
    'Has not completed training'
FROM
    drone.customer
WHERE
    cust_id NOT IN (
        SELECT
            cust_id
        FROM
            drone.cust_train
    )
ORDER BY
    cust_id;
```

## INTERSECTION

Find the trainers who have the same last name as any customer

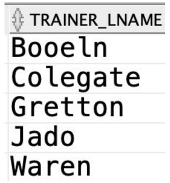


| CUST_LNAME |
|---|
| Brenneke |
| Brightey |
| Dudgeon |
| Dunlap |
| Flannery |
| Gravie |
| Harcombe |
| Harewood |
| Hebblewaite |
| Huntriss |
| Juden |
| Kiss |
| Lintall |
| Pabst |
| Reder |
| Roussel |
| Severy |
| Sisley |
| Waren |
| Zambonini |

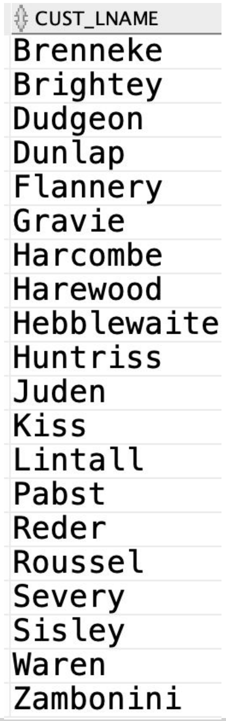


| TRAINER_LNAME |
|---|
| Booeln |
| Colegate |
| Gretton |
| Jado |
| Waren |

```sql
SELECT
    trainer_id,
    trainer_rego,
    trainer_fname,
    trainer_lname
FROM
    drone.trainer
WHERE
    trainer_lname IN (
        SELECT
            trainer_lname
        FROM
            drone.trainer
        INTERSECT
        SELECT
            cust_lname
        FROM
            drone.customer
    );
```



| TRAINER_LNAME |
|---|
| Booeln |
| Colegate |
| Gretton |
| Jado |
| Waren |



| CUST_LNAME |
|---|
| Brenneke |
| Brightey |
| Dudgeon |
| Dunlap |
| Flannery |
| Gravie |
| Harcombe |
| Harewood |
| Hebblewaite |
| Huntriss |
| Juden |
| Kiss |
| Lintall |
| Pabst |
| Reder |
| Roussel |
| Severy |
| Sisley |
| Waren |
| Zambonini |

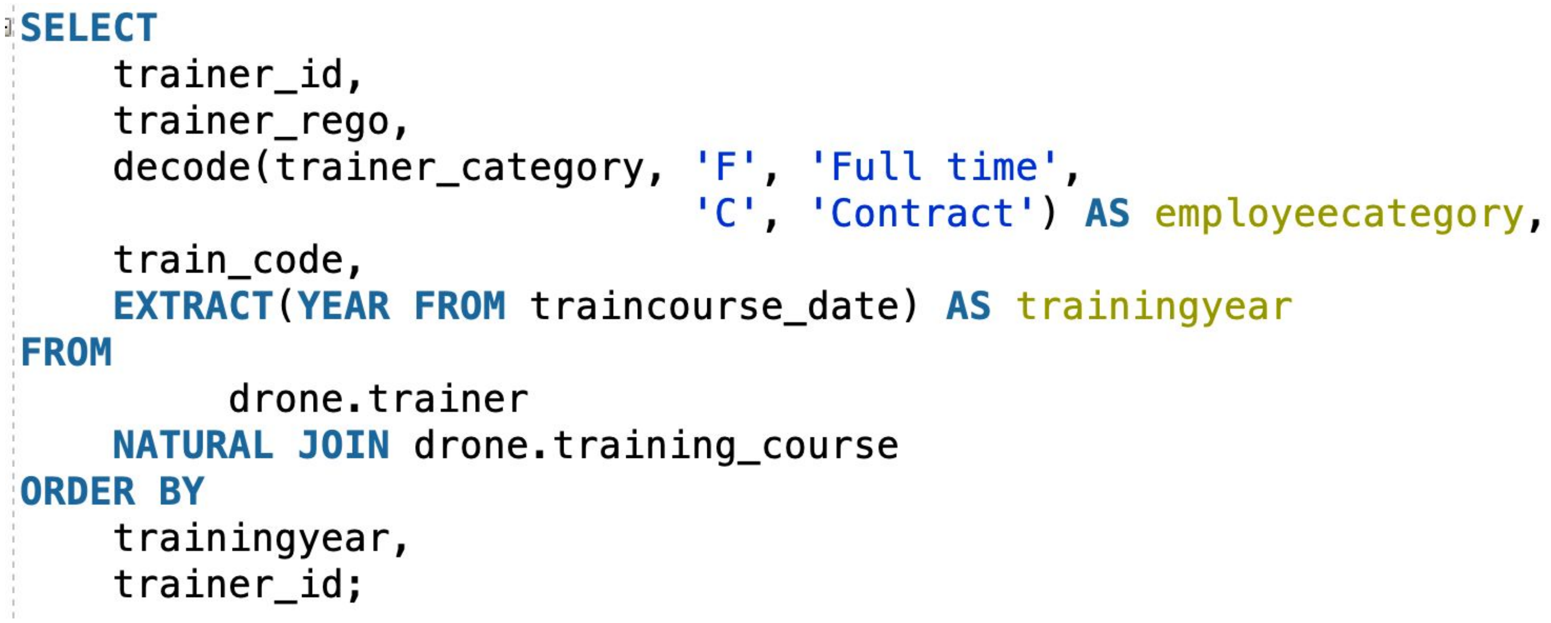
## Outline

- Case
- Subquery – nested, inline, correlated
- Views
- Joins - self join, outer join
- Set Operators
- **Oracle Functions**

| Function Type | Applicable to | Example |
|---|---|---|
| Arithmetic | Numerical data | SELECT ucode, round(avg(mark)) FROM enrolment GROUP BY ucode; |
| Text | Alpha numeric data | SELECT studsurname FROM enrolment WHERE upper(studsurname) LIKE 'B%'; |
| Date | Date/Time-related data | |
| General | Any data type | NVL function |
| Conversion | Data Type conversion | SELECT to_char(empmsal,'$0999.99') FROM employee; |
| Group | Sets of Values | avg(), count(), etc |
| | | |

**See document on Moodle**

# EXTRACT and DECODE



```
SELECT
    trainer_id,
    trainer_rego,
    decode(trainer_category, 'F', 'Full time',
                             'C', 'Contract') AS employeecategory,
    train_code,
    EXTRACT(YEAR FROM traincourse_date) AS trainingyear
FROM
        drone.trainer
    NATURAL JOIN drone.training_course
ORDER BY
    trainingyear,
    trainer_id;
```

# LPAD and LTRIM

```
SELECT
    drone_id,
    COUNT(*) AS times_rented,
    to_char(COUNT(*) * 100 /(
        SELECT
            COUNT(rent_in)
        FROM
            drone.rental
    ), '990.99') AS percent_overall
FROM
    drone.rental
WHERE
    rent_in IS NOT NULL
GROUP BY
    drone_id
ORDER BY
    percent_overall DESC;
```

| DRONE_ID | TIMES_RENTED | PERCENT_OVERALL |
|---|---|---|
| 103 | 6 | 27.27 |
| 118 | 4 | 18.18 |
| 101 | 3 | 13.64 |
| 113 | 2 | 9.09 |
| 112 | 2 | 9.09 |
| 100 | 2 | 9.09 |
| 102 | 1 | 4.55 |
| 111 | 1 | 4.55 |
| 117 | 1 | 4.55 |

```
SELECT
    drone_id,
    COUNT(*) AS times_rented,
    lpad(ltrim(to_char(COUNT(*) * 100 /(
        SELECT
            COUNT(rent_in)
        FROM
            drone.rental
    ), '990.99')),
        10) AS percent_overall
FROM
    drone.rental
WHERE
    rent_in IS NOT NULL
GROUP BY
    drone_id
ORDER BY
    percent_overall DESC;
```

| DRONE_ID | TIMES_RENTED | PERCENT_OVERALL |
|---|---|---|
| 103 | 6 | 27.27 |
| 118 | 4 | 18.18 |
| 101 | 3 | 13.64 |
| 113 | 2 | 9.09 |
| 112 | 2 | 9.09 |
| 100 | 2 | 9.09 |
| 102 | 1 | 4.55 |
| 111 | 1 | 4.55 |
| 117 | 1 | 4.55 |