BACHELOR'S DEGREE IN INFORMATICS ENGINEERING
Academic course: 2018/2019 -- 2nd Year, 2nd term
Subject: Files and Databases
Lab work Report 2: Relational Dynamics and Triggers

uc3m | Universidad Carlos III de Madrid

Lecturer:	**Francisco Javier Calle Gomez and Pedro Miguel Suja Goffin**		
Group:	**88**	**Lab User**	**fsdb285**
Student:	**Juan Sánchez Esquivel**	**NIA:**	**100383422**
Student:	**-**	**NIA:**	**-**

Index

1. Introduction

For this practice I will have to develop some queries, views, external designs and triggers. The complication of the task being on the fact that I have never done anything similar before and therefor it will very likely take much more time than expected on acquiring basic knowledge.

This document will contain a specific section for each requirement (queries, views, external design and triggers) as well as an overall conclusion of the practice itself. I will try to make the understanding of this document as well of the code as easy as possible by putting it next to where I commented. The code used for creation of the queries, views and triggers will also be available in the zip folder submitted along with this document being named as (queries.sql, views.sql and triggers.sql) Tests will include screenshots of both my input and the obtained output as well as some comment to be clear about the purpose and expected results of each one.

2. Queries

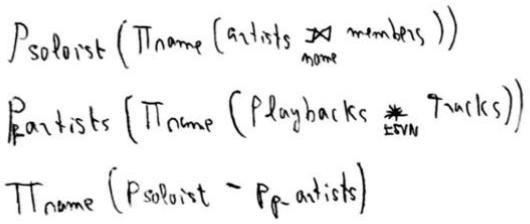
In this part I had to develop some queries, both in relational algebra first and then on SQL, after doing so I had to check whether or not they worked by doing some testing trying to cover all possible scenarios.

Remark: Some of the queries where based on time premises (ie yesterday) however the database did not contain any data from the 31 of December 2018 on, so in order to test it and see its expected behavior the condition sysdate-1 (yesterday) was replaced with sysdate-x (x being number of days since the 31 of December 2018) so those queries will retrieve no rows as the time will have passed by, in order to adjust them if desired x will be the only value to change. In a real database sysdate-1 will be set and never changed (as entries will be made each day) I apologize for the inconvenience

Not by myself

This query should return the soloist who do not interpret their songs

- Relational Algebra



- SQL

In order to do this query on SQL I thought the best approach would be to organize it by using WITH AS statements, one of them used to obtain the soloist and the other one to obtain the writers. After doing so, all I had left to do was to substract both queries to obtain the desired not by myself query

```
WITH SOLOIST AS
(
  SELECT ARTISTS.NAME name FROM ARTISTS
  LEFT OUTER JOIN
  MEMBERS ON ARTISTS.NAME = MEMBERS.GROUP_NAME
  WHERE
  MEMBERS.GROUP_NAME IS NULL
),
P_ARTIST AS
(
  SELECT DISTINCT WRITER name FROM PLAYBACKS
  NATURAL JOIN TRACKS
),
NOT_BY_MYSELF AS
(
  SELECT name from SOLOIST
  MINUS
  SELECT name from P_ARTIST
)
SELECT * FROM NOT_BY_MYSELF;
```

- Tests:

The obtain result from the above query (which is show on the image bellow this text) is what I will use to compare with the test cases. If test were successful they will appear on green otherwise they will appear on red

1	Abri	26	Gaston Morriberon	51	Ruiz Matta
2	Alfonso	27	Geraldina	52	Sandra Pandora Pineda
3	Almira	28	Gerardo Enrique Mayhua	53	Sanguina
4	Ana	29	Guilli	54	Santiago Triveno
5	Apazi	30	Jenardo	55	Sebi
6	Aragi	31	Jose Fernando Cardenas	56	Tiani
7	Azarias Llasa	32	Jubal Sanroman	57	Tomas Diego Ocampo
8	Buho	33	Julca	58	Tornado
9	Camino	34	M.	59	Torres Alegria
10	Campos	35	Manuel	60	Triveno
11	Carbonilla	36	Marcico	61	Violeta Gomez
12	Cayetano	37	Maria Bienvenida Perez	62	Zete
13	Cesto	38	Maria Inmaculada "Inmi" Ali	63	angel Lopez
14	Chamorro	39	Maria Pulido		
15	Consti	40	Maria de la Paciencia Caldas		
16	Denis Cayo	41	Maria de las Virtudes Garate		
17	Diego	42	Mario Luna		
18	Echi	43	Marivi Moreira		
19	Emilio Carranza	44	Mayi		
20	Esteban Marquina	45	Mildred Chuan		
21	Felisa Ruiz	46	Oscar Lazo		
22	Fernando Raul "R.N.M." Nicasio	47	Oscar Roberto Castaneda		
23	Francisca Clara Garcia	48	Pablo Bullon		
24	Francy	49	Pedro Paredes		
25	Fulgencio Garcia	50	Rosi		

Test-1

Check whether Abri, who is a soloist who does not write its song appears on the result the query and does not appear on the writers selection

```
SELECT NAME FROM ARTISTS WHERE NAME='Abri';
```

ida de Script | Resultado de la Consulta | SQL | Todas las Filas Recuperadas: 1 en 0 segundo

NAME
1 Abri

```
SELECT WRITER FROM PLAYBACKS NATURAL JOIN TRACKS WHERE WRITER='Abri';
```

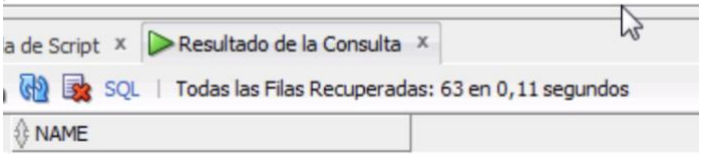
la de Script | Resultado de la Consulta | SQL | Todas las Filas Recuperadas: 0 en 0,031 segundos

WRITER

Test-2

Check whether those who are soloists and write their song do not appear on the result the query, as if we subtract them from our query the result is the same (63 rows) we can be sure our query does not contain any soloist who is a writer

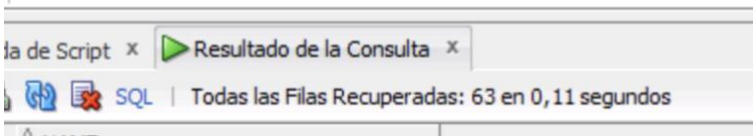
```
SELECT * FROM NOT_BY_MYSELF
MINUS
SELECT * FROM SOLOIST NATURAL JOIN P_ARTIST;
```



Test-3

Check whether those who are not soloists and write their song do not appear on the result the query, as if we subtract them from our query the result is the same (63 rows) we can be sure our query does not contain any writer who is not a soloist

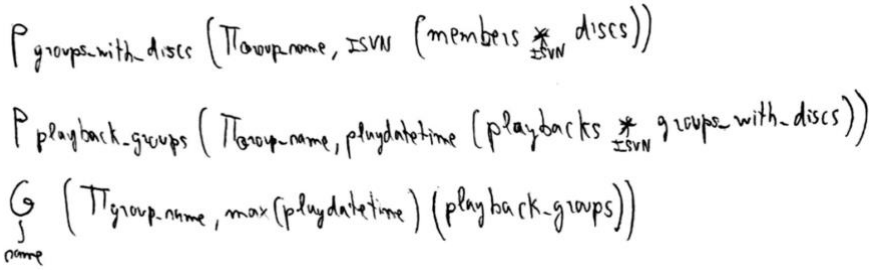
```
SELECT * FROM NOT_BY_MYSELF
MINUS
(SELECT * FROM P_ARTIST MINUS SELECT * FROM SOLOIST);
```



Still standing

This query should return last date and time each group was played on the radio. One of the highest difficulties encountered on the development of these queries was to realize that in this version of oracle sql (which is an old one) natural join operator has a bug in which sometimes it does not return the same join using while it should. That happened on this query, nevertheless when I realized that with a minimum change on the query everything worked as expected.

- Relational Algebra



- SQL

In order to do this query on SQL I again chose to organize it by using WITH AS statements. First one of them will store those groups who have at least a disc. Second one will contain all playdatetimes for each group. Finally, we will select from that last query the artist and the maximum date playdate grouping by each group, so we will retreive each group and its last time playing.

```
WITH GROUPS_WITH_DISCS AS
(
  SELECT artist, isvn FROM (SELECT DISTINCT group_name artist FROM MEMBERS)
  JOIN DISCS USING (artist)
),
PLAYBACK_GROUPS AS
(
  SELECT artist, playdatetime lastplayed FROM PLAYBACKS
  JOIN GROUPS_WITH_DISCS USING(isvn)
)
SELECT artist, MAX(lastplayed) FROM PLAYBACK_GROUPS GROUP BY artist;
```

- Tests:

The obtain result from the above query (which is show on the image bellow this text) is what I will use to compare with the test cases. If tests were successful they will appear on green otherwise they will appear on red

1	Cigüeñales	19/12/18	26	La Macariobunta	16/12/18	51	Silky Pond	20/12/18	76	Pollo Al'Ast	30/12/18	101	Pájaros de Cuenta	28/12/18
2	Luces de Charol	31/12/18	27	Tragaldaba's	26/12/18	52	Los Eremitas	13/12/18	77	Turquoise Fluid	31/12/18	102	The Decanters	23/06/18
3	Estanque Abierto	27/12/18	28	Geranios Flamencos	31/12/18	53	Aspersores	27/12/18	78	Fruta del Tiempo	17/09/18	103	Zeichnung	28/12/18
4	Garabatos	30/12/18	29	Movin' Hillbillies	18/11/18	54	Caleidoscopia	24/11/18	79	Los Sembrados	29/12/18	104	Fiestalerondia	16/12/18
5	Service Paper Pack	08/12/18	30	Play Backers	16/12/18	55	Meseros	18/12/18	80	Luna Vieja	13/11/18	105	Wild Horses	07/11/18
6	Casascarro	31/12/18	31	Cantadores	29/12/18	56	Canta Mañanitas	03/11/18	81	The Speedy Gamo Band	15/12/18	106	Dare Streets	31/12/18
7	Flawless Imperfection	27/12/18	32	Von Pelele's	16/07/18	57	Los Venados	31/12/18	82	The stamp	31/12/18	107	WalKing Queen	29/12/18
8	Piedras del Camino	30/12/18	33	Colore di Mare	08/11/18	58	The Last Straws	29/12/18	83	Rancatermia	31/12/18	108	Wreckin' Sound	24/12/18
9	Recuerdos de Ayer	20/12/18	34	Melting Vinyl	17/12/18	59	Ain't Got Talent	27/12/18	84	SetUp Complete	19/11/18	109	7 Ate 9	21/11/18
10	Amapola	28/12/18	35	Los Vendimialotodo	18/06/18	60	The Great Pretending ...	28/12/18	85	Estri Dientes	31/12/18	110	Wanderer Wonders	24/12/18
11	Certain Possibility	27/12/18	36	Jollas de la Korona	28/12/18	61	The Moony Shadow	27/12/18	86	Repollos de Paris	30/06/18	111	Celtas con Filtro	23/12/18
12	Los Caseros	27/12/18	37	Los Gatucos	21/11/18	62	The Pines	18/11/18	87	Limbotron	20/11/18	112	Camperos	17/12/18
13	The Principles	30/12/18	38	Encharcados	25/12/18	63	Golondrinajes	17/11/18	88	The Puerco Quartet	09/12/18	113	Alba LEA	03/12/18
14	Los Ascetas	16/12/18	39	Margaritas Lilas	29/12/18	64	Apartaments	30/11/18	89	Astupendis	26/12/18	114	Velerables	10/09/18
15	Urban Pollution	26/12/18	40	Los Enólogos	01/12/18	65	Montanaderos	15/12/18	90	Estrellas Errantes	12/12/18	115	Rounded Boxes	15/12/18
16	The Gadflies	28/12/18	41	The Upgrades	28/12/18	66	Vacas y Toros	30/12/18	91	The Boozer Buzz	27/12/18	116	Rocas y Rollos	28/12/18
17	Los Lugareños	16/12/18	42	Primitivo y los Chalados	06/12/18	67	Chapela Calá	19/11/18	92	Tourists in Heaven	17/12/18	117	La Ciudadela	31/12/18
18	Macanudo's	27/12/18	43	Annoyers	30/12/18	68	The Gorrinos	19/11/18	93	Muzicalizers	15/12/18	118	Evening in Cornwall	27/11/18
19	Ostaledopia	24/12/18	44	First In & Last Out	24/12/18	69	More About Nothing	10/12/18	94	The Fairy HobGoblins	26/12/18	119	Nuestra Tierra	11/12/18
20	Garrulantes	31/12/18	45	The Ending Chart	30/12/18	70	Tristes Sauces	28/12/18	95	Quijotes y Sanchos	31/12/18	120	Burros Voladores	23/05/18
21	The Wine Tasters	25/12/18	46	Scorching Earth	30/12/18	71	Los Molinos Escocidos	14/09/18	96	Lilas Quartet	16/12/18			
22	Desfile de Paz	03/12/18	47	Magnificient Us	25/12/18	72	Reinfangoria	19/08/18	97	Bade Bagones	31/08/18			
23	Regardotopia	29/12/18	48	Mazinger Zetas	22/12/18	73	Lirio Band	31/12/18	98	Perros Galgos	30/12/18			
24	Penciler	10/12/18	49	Balcomes Band	30/12/18	74	Mowgli Went Wild	14/12/18	99	Telastres	29/12/18			
25	La Forja del Zorro	31/12/18	50	Legomiglia	22/12/18	75	Estampida	28/12/18	100	Pluviasmil	30/12/18			

Test-1

Check whether all groups where included, to do so we execute the shown query, as the cardinality is the same as in our query 120 we can be sure have included all groups

```
SELECT DISTINCT GROUP_NAME FROM MEMBERS;
```

Salida de Script | Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 120 en 0,015 segundos

	GROUP_NAME
107	La Ciudadela
108	Velerables

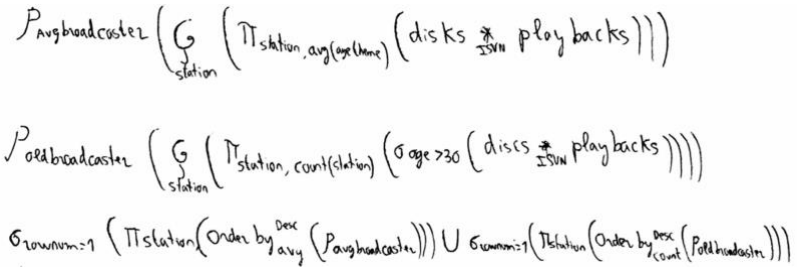
Test-2

Check whether a group has indeed as playdatetime the retrieved one for instance La Ciudadela (row 127 on the result of my query) or Estanque Abierto (row 3 on the result of my query) As both match, we can be sure our query is correctly working.

Revival Channels

This query should return both the name of the radio that plays the oldest themes as well as the name of the radio that plays more often old themes

- Relational Algebra



- SQL

In order to do this query on SQL I thought the best approach would be to organize it by using WITH AS statements as on previous queries. First of them called avgbroadcaster will contain the name of the stations as well as the mean of age of song played on each station. Then, the second one, called oldbroadcaster will contain for each station name as well as the number of times it has played songs of an age greater than 30 years. After doing this we will select

```
WITH avgbroadcaster AS
(
  SELECT station, ((sysdate-median(rel_date))/365.2422) avg FROM discs d
    JOIN playbacks p ON (p.ISVN = d.ISVN)
  GROUP BY station
),
oldbroadcaster AS
(
  SELECT station, COUNT('X') Count_old FROM discs d
    JOIN playbacks p ON (p.ISVN = d.ISVN)
  WHERE ((sysdate-rel_date)/365.2422)>30
  GROUP BY station
)
SELECT station FROM (SELECT * FROM avgbroadcaster ORDER BY avg DESC)
  WHERE rownum = 1
UNION ALL
SELECT station FROM (SELECT * FROM oldbroadcaster ORDER BY count_old DESC)
  WHERE rownum = 1;
```

the union all of both statements selecting having previously order each one in descendent order and selected only the first row. The result of the query will be two rows with only one column which will be its names. First row being the broadcaster that plays the oldest themes and second oen being the broadcaster that plays more often old themes

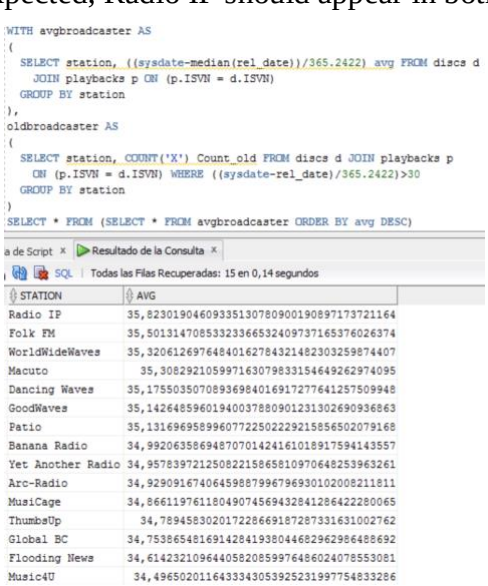
- Tests:

The obtain result from the above query (which is show on the image bellow this text) is what I will use to compare with the tests cases. If tests were successful they will appear on green otherwise they will appear on red

STATION
Radio IP
Radio IP

Test-1

Check whether Radio IP has the greatest values in both of the WITH AS statements. In order to check so we obtain the result of selecting * from each statement (avgbroadcaster and oldbroadcaster) only ordering by descending order (not selecting only the first row). As we can see from the bellow images the query is working as expected, Radio IP should appear in both rows



```
WITH avgbroadcaster AS
(
  SELECT station, ((sysdate-median(rel_date))/365.2422) avg FROM discs d
    JOIN playbacks p ON (p.ISVN = d.ISVN)
  GROUP BY station
),
oldbroadcaster AS
(
  SELECT station, COUNT('X') Count_old FROM discs d JOIN playbacks p
    ON (p.ISVN = d.ISVN) WHERE ((sysdate-rel_date)/365.2422)>30
  GROUP BY station
)
SELECT * FROM (SELECT * FROM avgbroadcaster ORDER BY avg DESC)
```

a de Script | Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 15 en 0,14 segundos

STATION	AVG
Radio IP	35,823019046093351307809001908971737211164
Folk FM	35,501314708533233665324097371653760263 74
WorldWideWaves	35,320612697648401627843214823032598744 07
Macuto	35,308292105997163079833154649262974095
Dancing Waves	35,175503507089369840169172776412575099 48
GoodWaves	35,142648596019400378809012313026909368 63
Patio	35,131696958996077225022292158565020791 68
Banana Radio	34,992063586948707014241610189175941435 57
Yet Another Radio	34,957839721250822158658109706482539632 61
Arc-Radio	34,929091674064598879967969301020082118 11
MusiCage	34,866119761180490745694328412864222800 65
ThumbsUp	34,789458302017228669187287331631002762
Global BC	34,753865481691428419380446829629864886 92
Flooding News	34,614232109644058208599764860240785530 81
Music4U	34,496502011643334303925231997754833286

```
WITH avgbroadcaster AS
(
  SELECT station, ((sysdate-median(rel_date))/365.2422) avg FROM discs d
    JOIN playbacks p ON (p.ISVN = d.ISVN)
  GROUP BY station
),
oldbroadcaster AS
(
  SELECT station, COUNT('X') Count_old FROM discs d JOIN playbacks p
    ON (p.ISVN = d.ISVN) WHERE ((sysdate-rel_date)/365.2422)>30
  GROUP BY station
)
SELECT * FROM (SELECT * FROM oldbroadcaster ORDER BY count_old DESC)
```

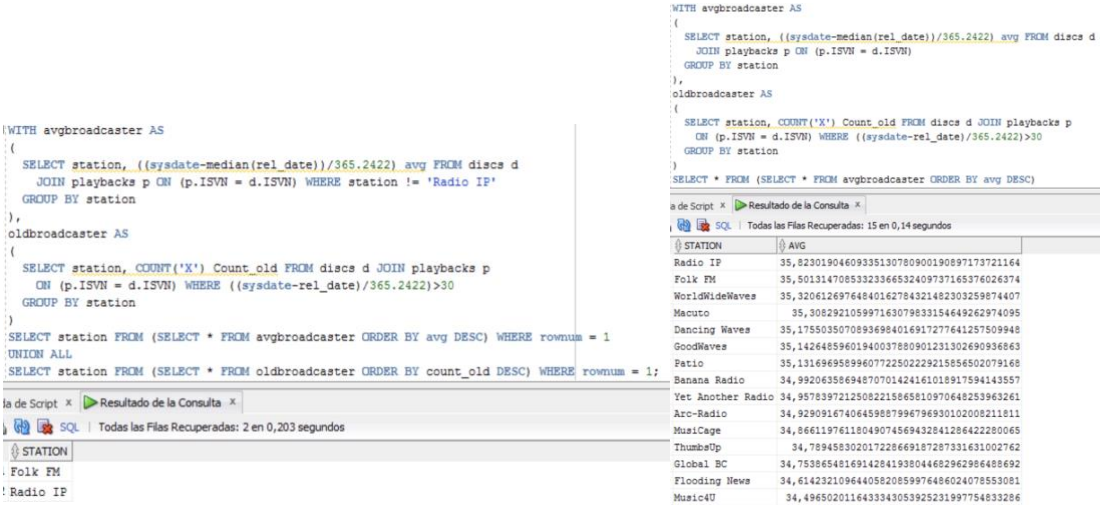
a de Script | Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 15 en 0,047 segundos

STATION	COUNT_OLD
Radio IP	2292
Banana Radio	2230
Arc-Radio	2225
Dancing Waves	2214
MusiCage	2212
WorldWideWaves	2207
GoodWaves	2189
Patio	2185
Macuto	2176
Global BC	2172
Folk FM	2136
ThumbsUp	2134
Yet Another Radio	2116
Music4U	2115
Flooding News	2105

Test-2

Check if avgbroadcaster works as expected. To do so we add to avgbroadcaster with as statement the condition station not equal to Radio IP, if it works as expected the second radio having the greatest average of themes played should be picked while second row (from the view) should not change. As it does so, we can be sure query is working as expected.



```
WITH avgbroadcaster AS
(
  SELECT station, ((sysdate-median(rel_date))/365.2422) avg FROM discs d
    JOIN playbacks p ON (p.ISVN = d.ISVN) WHERE station != 'Radio IP'
  GROUP BY station
),
oldbroadcaster AS
(
  SELECT station, COUNT('X') Count_old FROM discs d JOIN playbacks p
    ON (p.ISVN = d.ISVN) WHERE ((sysdate-rel_date)/365.2422)>30
  GROUP BY station
)
SELECT station FROM (SELECT * FROM avgbroadcaster ORDER BY avg DESC) WHERE rownum = 1
UNION ALL
SELECT station FROM (SELECT * FROM oldbroadcaster ORDER BY count_old DESC) WHERE rownum = 1;
```

Resultado de la Consulta — Todas las Filas Recuperadas: 2 en 0,203 segundos

STATION
Folk FM
Radio IP

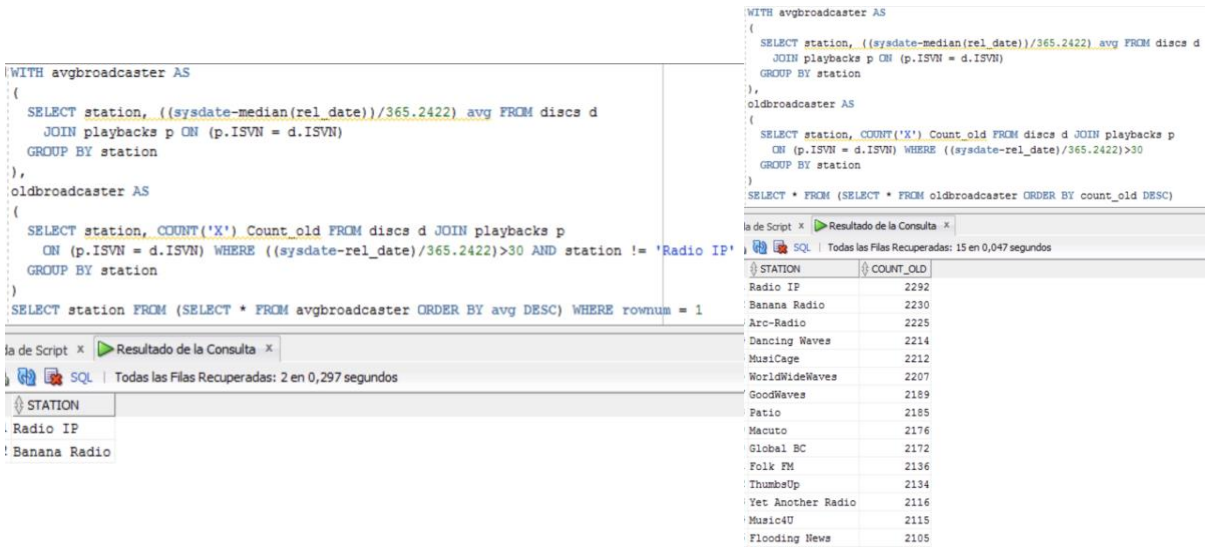
```
WITH avgbroadcaster AS
(
  SELECT station, ((sysdate-median(rel_date))/365.2422) avg FROM discs d
    JOIN playbacks p ON (p.ISVN = d.ISVN)
  GROUP BY station
),
oldbroadcaster AS
(
  SELECT station, COUNT('X') Count_old FROM discs d JOIN playbacks p
    ON (p.ISVN = d.ISVN) WHERE ((sysdate-rel_date)/365.2422)>30
  GROUP BY station
)
SELECT * FROM (SELECT * FROM avgbroadcaster ORDER BY avg DESC)
```

Resultado de la Consulta — Todas las Filas Recuperadas: 15 en 0,14 segundos

STATION	AVG
Radio IP	35,823019046093351307809001908971737211 64
Folk FM	35,501314708533233665324097371653760263 74
WorldWideWaves	35,320612697648401627843214823033259874 07
Macuto	35,308292105997163079833154649262974095
Dancing Waves	35,175503507089369840169173277641257509948
GoodWaves	35,142648596019400378809012313026909368 63
Patio	35,131696958996077225022292158565020791 68
Banana Radio	34,992063586948707014241610189175941435 57
Yet Another Radio	34,957839721250822158658109706482539632 61
Arc-Radio	34,929091674064598879967969301020082118 11
MusiCage	34,866119761180490745694328412864222800 65
ThumbsUp	34,789458302017228669187287331631002762
Global BC	34,753865481691428419380446829629864886 92
Flooding News	34,614232109644058208599764860240785530 81
Music4U	34,496502011643334305392523199775483328 6

Test-3

Check if oldbroadcaster works as expected. To do so we add to oldbroadcaster with as statement the condition station not equal to Radio IP, if it works as expected the second radio that plays more often old themes should be picked while first row (from the view) should not change. As it does so, we can be sure query is working as expected.



```
WITH avgbroadcaster AS
(
  SELECT station, ((sysdate-median(rel_date))/365.2422) avg FROM discs d
    JOIN playbacks p ON (p.ISVN = d.ISVN)
  GROUP BY station
),
oldbroadcaster AS
(
  SELECT station, COUNT('X') Count_old FROM discs d JOIN playbacks p
    ON (p.ISVN = d.ISVN) WHERE ((sysdate-rel_date)/365.2422)>30 AND station != 'Radio IP'
  GROUP BY station
)
SELECT station FROM (SELECT * FROM avgbroadcaster ORDER BY avg DESC) WHERE rownum = 1
```

Resultado de la Consulta — Todas las Filas Recuperadas: 2 en 0,297 segundos

STATION
Radio IP
Banana Radio

```
WITH avgbroadcaster AS
(
  SELECT station, ((sysdate-median(rel_date))/365.2422) avg FROM discs d
    JOIN playbacks p ON (p.ISVN = d.ISVN)
  GROUP BY station
),
oldbroadcaster AS
(
  SELECT station, COUNT('X') Count_old FROM discs d JOIN playbacks p
    ON (p.ISVN = d.ISVN) WHERE ((sysdate-rel_date)/365.2422)>30
  GROUP BY station
)
SELECT * FROM (SELECT * FROM oldbroadcaster ORDER BY count_old DESC)
```

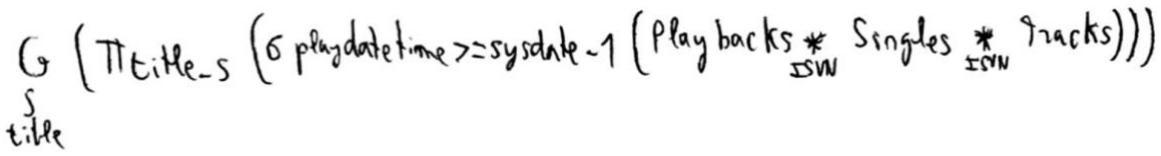
Resultado de la Consulta — Todas las Filas Recuperadas: 15 en 0,047 segundos

STATION	COUNT_OLD
Radio IP	2292
Banana Radio	2230
Arc-Radio	2225
Dancing Waves	2214
MusiCage	2212
WorldWideWaves	2207
GoodWaves	2189
Patio	2185
Macuto	2176
Global BC	2172
Folk FM	2136
ThumbsUp	2134
Yet Another Radio	2116
Music4U	2115
Flooding News	2105

Trending catchy

This query should return the singles that were more listen to yesterday

- Relational Algebra



- SQL

In order to do this query on SQL I did the join on the condition of having the same isvn for the three tables, then select yesterday by the use of sysdate and then group and order the results by count in descendent order.

```
SELECT title_s FROM PLAYBACKS
JOIN SINGLES ON (playbacks.isvn = singles.isvn)
JOIN TRACKS ON (playbacks.isvn = tracks.isvn)
  WHERE playdatetime >= SYSDATE -110
  GROUP BY title_s ORDER BY COUNT('X') DESC;
```

- Tests:

The obtain result from the above query (which is show on the image bellow this text) is what I will use to compare with the test cases. If tests were successful they will appear on green otherwise they will appear on red

1	Win
2	Mixture or edelwiss
3	Seedtime armor
4	Morir tierra (dance)
5	Mambo holm oak
6	Fisher
7	Later Rhythm
8	Morir tierra
9	Free
10	Thrust or people
11	Amor en su momento
12	Televisión de estanque
13	Hitch
14	Bear petunia
15	Petunias son
16	Gracias amigo
17	Final de cómo (remix)
18	Hermitages
19	Duelo y quién
20	Sea
21	Cooker and permanent
22	Petunias son (ext. version)
23	Waltz or skin
24	New
25	Moon barrel
26	Uranus band
27	Final de cómo
28	Mercury of vitamine
29	Charm

Test-1

Check whether Win, which is the most played song yesterday (31/12/18 as is the last date on the database) is the first one on the table. To check so we can easily the most played singles yesterday by seeing the whole table of singles played "yesterday"

```
SELECT title_s FROM PLAYBACKS JOIN SINGLES ON (playbacks.isvn = singles.isvn)
JOIN TRACKS ON (playbacks.isvn = tracks.isvn);
```

alida de Script × Resultado de la Consulta ×
SQL | Todas las Filas Recuperadas: 30 en 0,016 segundos

1	Moon barrel	11	Later Rhythm	21	Waltz or skin
2	Mambo holm oak	12	Charm	22	Hermitages
3	Sea	13	Free	23	Morir tierra
4	Win	14	Cooker and permanent	24	Morir tierra (dance)
5	Mixture or edelwiss	15	Thrust or people	25	Gracias amigo
6	Bear petunia	16	Seedtime armor	26	Duelo y quién
7	Fisher	17	Petunias son	27	New
8	Mercury of vitamine	18	Petunias son (ext. version)	28	Hitch
9	Win	19	Amor en su momento	29	Final de cómo
10	Uranus band	20	Televisión de estanque	30	Final de cómo (remix)

```
SELECT * FROM NOT_BY_MYSELF
MINUS
(SELECT * FROM P_ARTIST MINUS SELECT * FROM SOLOIST);
```

la de Script × Resultado de la Consulta ×
SQL | Todas las Filas Recuperadas: 63 en 0,11 segundos

Ruling Stone

This query should return longest lived group among those in the database

- Relational Algebra

$$P_{bands}\left(\pi_{artist}\left(\mathcal{G}_{\substack{count>1\\artist}}\left(Members\right)\right)\right)$$

$$P_{band\_ages}\left(\mathcal{G}_{artists}\left(\pi_{artist,\,age}\left(\mathcal{G}_{artist}\left(P_{bands} \overset{*}{_{artist}} discs\right)\right)\right)\right)$$

$$\sigma_{rownum=1}\left(\text{Order By}^{\text{Desc}}_{age}\left(P_{band\_ages}\right)\right)$$

- SQL

In order to do this query on SQL I thought the best approach would be to organize it by using WITH AS statements as in on previous queries. First of them called bands would contain the name of the groups, then band_ages would contain for each band its maximun age (calculated by substracting to the lastest release date the first release date). Finally we would print to the user the first row of the table band_ages after ordering by descending order (greatest age first)

```
WITH bands AS (
  SELECT artist FROM discs WHERE artist
  IN (SELECT Group_name FROM Members WHERE (artist=group_name))
),
band_ages AS (
  SELECT artist,(MAX(rel_date)-MIN(rel_date))/365.2422 age
  FROM bands JOIN discs USING (artist) GROUP BY artist
)
SELECT * FROM (SELECT * FROM band_ages ORDER BY age DESC) WHERE rownum=1;
```

- Tests:

The obtain result from the above query (which is show on the image bellow this text) is what I will use to compare with the tests cases. If tests were successful they will appear on green otherwise they will appear on red

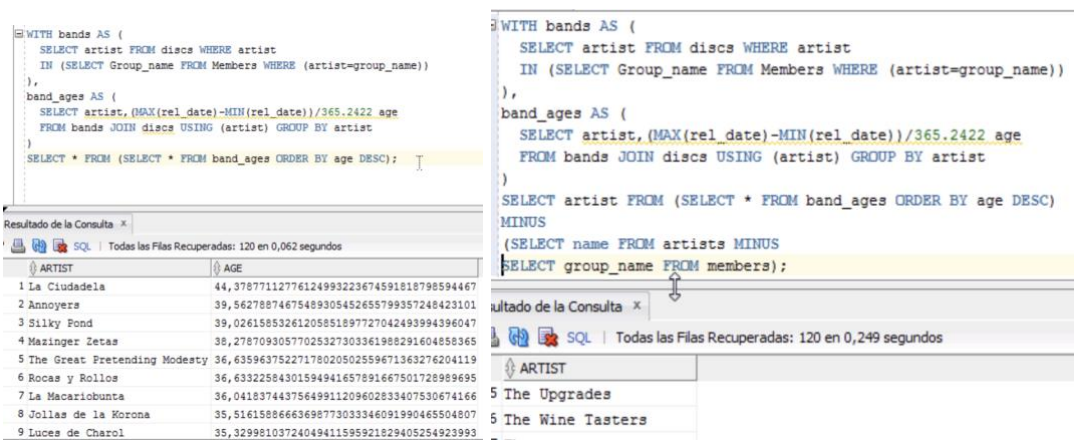
ARTIST	AGE
La Ciudadela	44,378771127761249932236745918187985944 67

Test-1

Check whether La Ciudadela, who is the longest-lived group on the database appears as the result

Test-2

Check whether non-groups appear on our table or not, in order to check so we remove the rownum constraint and we try to substract them, as the total number of rows is the same in both queries we can be sure only groups appear on our query



Test-3

Check whether all groups are being considered on the query, to do so we substract to all the groups name selection our query (again without the constraint of rownum=1) As such query returns 0 rows we can be sure all groups where taken into consideration for the query.

```
WITH bands AS (
  SELECT artist FROM discs WHERE artist
  IN (SELECT Group_name FROM Members WHERE (artist=group_name))
),
band_ages AS (
  SELECT artist,(MAX(rel_date)-MIN(rel_date))/365.2422 age
  FROM bands JOIN discs USING (artist) GROUP BY artist
)
SELECT DISTINCT group_name FROM MEMBERS
MINUS
SELECT artist FROM (SELECT * FROM band_ages ORDER BY age DESC);
```

```
esultado de la Consulta
SQL | Todas las Filas Recuperadas: 0 en 0,218 segundos
```

3. Views

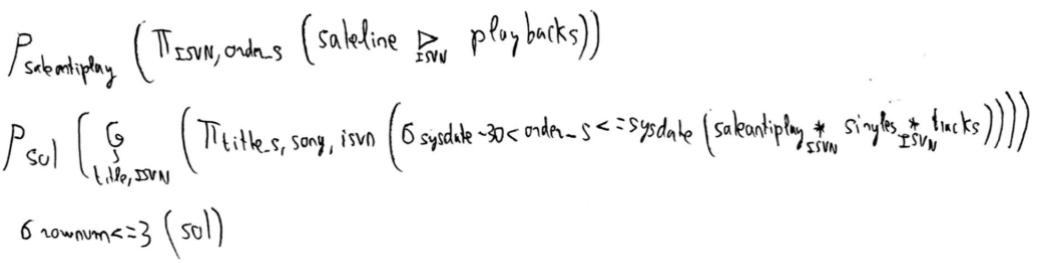
In this second part some views had to be developed, the functionality of them being retrieval and ease of use for all the users we granted access to the views. On a future possible implementation all of them could be used to insert delete and/or update but as they were created, they are only thought for reading. On the following pages there is a specific section for each view, in each section you can find a brief description of the view and its functionality, relational algebra of the query used for the view, the SQL code description as well as a photo of it (code is also included on the views script) and lastly some test cases checking if our view is showing all the expected results.

Remark: Similarly to queries, some views where based on time premises (ie last 30 days) however the database did not contain any data from the 31 of December 2018 on, so in order to test it and see its expected behavior the condition sysdate-30 (last 30 days) was replaced with sysdate-x (x being number of days since the 1 of December 2018) so those queries will retrieve no or less rows as the time will have passed by, in order to adjust them if desired x will be the only value to change. In a real database sysdate-30 will be set and never changed (as entries will be made each day and therefor there will be data) I apologize for the inconvenience

Top sales of banned songs

This view should contain the three best sold songs that were not playbacked on the radio on the last 30 days

- ## Relational Algebra



- ## SQL

For doing this view on SQL I thought the best approach would be to organize it by using WITH AS statements. First one of them (sales_antijoint_playbacks) selecting those sales that were not playbacked on the radio but where purchased. After that, on Sol I obtain the name of the songs that are on the previous statement join with singles, with the characteristic of being in the desired data rage. Lastly, I select the first three rows of sol as I am only asked for the top 3.

```
CREATE OR REPLACE VIEW TopBannedSongs AS
WITH sales_antijoint_playbacks AS
(
SELECT ISVN, ORDER_S FROM SALE_LINE s
                    WHERE  (NOT EXISTS ( SELECT 1
                    FROM   PLAYBACKS p WHERE  p.ISVN = s.ISVN))
),
Sol AS
(
SELECT TITLE_S as Song, SINGLES.ISVN as ISVN FROM sales_antijoint_playbacks s
                JOIN SINGLES ON (SINGLES.ISVN = s.ISVN)
                JOIN TRACKS ON (TRACKS.ISVN = s.ISVN)
          WHERE (ORDER_S >= SYSDATE -160
                AND ORDER_S <= CURRENT_DATE)
              GROUP BY TITLE_S, SINGLES.ISVN ORDER BY COUNT('X') DESC
)
SELECT * FROM SOL WHERE rownum <= 3 ;
```

- ## Tests:

The obtain result from the above query (which is show on the image bellow this text) is what I will use to compare with the test cases. If test were successful they will appear on green otherwise they will appear on red

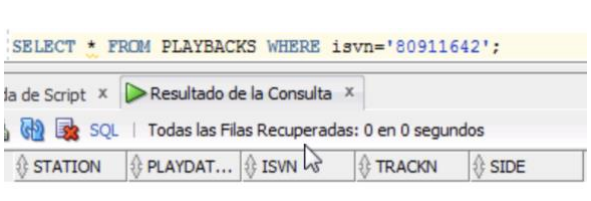
```
SELECT * FROM topbannedsongs;
```

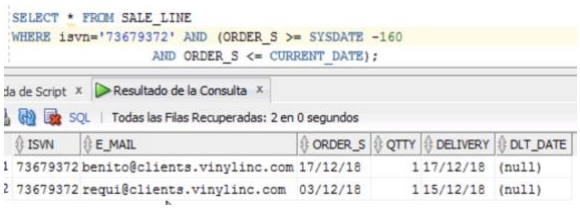
alida de Script | Resultado de la Consulta | SQL | Todas las Filas Recuperadas: 3

	SONG	ISVN
1	Operation (live)	80911642
2	Operation	80911642
3	Sing (live)	73701153

Test-1

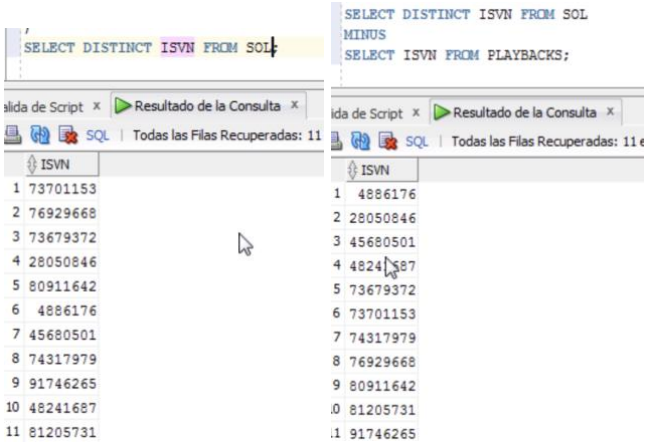
Check if Operation has been play backed during the given dates, if it is a single and if it has been sold many times. As we can easily obtain that the single has not been played but is a single and has been sold more than other elements lower on the table, we can be sure it works as expected





Test-2

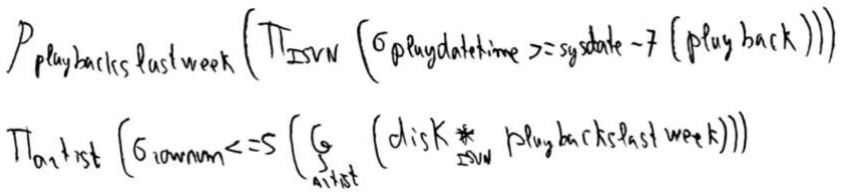
Check whether play backed songs during that period are included in the view. In order to check so I deleted the first line create or update view with a and I remove the last line in which I selected three rows so we can check the cardinality is the same after subtracting queries, the operations are shown below, SOL is the query of the view but with all the rows. As we obtain the same number of rows (as well as the same values if you look carefully) we can be sure playbacked songs during that period are indeed included



Top five week peak

This view should contain the top five artists listened during the last seven days

- ### Relational Algebra



- ### SQL

For doing this view on SQL I first used a WITH AS statement in which I will select those playbacks that occurred during the last week, then I would select the artist name from the discs table join the previous statement, group and order them. Finally, we will select the first 5 rows as we are only interested on those

```
CREATE OR REPLACE VIEW top_five_week AS
WITH playbackslastweek AS
(
  SELECT ISVN FROM PLAYBACKS p
        WHERE (p.PLAYDATETIME >= SYSDATE -199)
)
SELECT * FROM
(
SELECT ARTIST
FROM DISCS JOIN playbackslastweek ON DISCS.ISVN=playbackslastweek.ISVN
GROUP BY ARTIST ORDER BY COUNT('X') DESC
)
WHERE rownum<=5;

SELECT * FROM top_five_week;
```

- ### Tests:

The obtain result from the above query (which is show on the image bellow this text) is what I will use to compare with the test cases. If test were successful they will appear on green otherwise they will appear on red

```
SELECT * FROM top_five_week;
```

alida de Script | Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 5 en 0,015 segundos

	ARTIST
1	Ramon Jorge Gomez
2	Mariano Oscar Barrientos
3	Luces de Charol
4	Annoyers
5	Andri

Test-1

Check whether the selected artists where indeed the more played ones. To check so, I have used this auxiliary query. By using it we can easily observe the selected ones are the ones that had more occurrences which implies that our query is making the proper selection

```
WITH playbackslastweek AS
(
  SELECT ISVN FROM PLAYBACKS p
        WHERE (p.PLAYDATETIME >= SYSDATE -130)
)
SELECT * FROM
(
SELECT ARTIST, COUNT('X') cuenta
FROM DISCS JOIN playbackslastweek ON DISCS.ISVN=playbackslastweek.ISVN
GROUP BY ARTIST ORDER BY COUNT('X') DESC
) WHERE rownum <= 5;
```

```
WITH playbackslastweek AS
(
  SELECT ISVN FROM PLAYBACKS p
        WHERE (p.PLAYDATETIME >= SYSDATE -199)
)
SELECT * FROM
(
SELECT ARTIST, COUNT('X') cuenta
FROM DISCS JOIN playbackslastweek ON DISCS.ISVN=playbackslastweek.ISVN
GROUP BY ARTIST ORDER BY COUNT('X') DESC
) WHERE rownum <= 5;
```

Salida de Script | Resultado de la Consulta

SQL | Todas las Filas Recuperadas: 5 en 0,015 segundos

	ARTIST	CUENTA
1	Ramon Jorge Gomez	60
2	Mariano Oscar Barrientos	37
3	Luces de Charol	28
4	Annoyers	27
5	Andri	27

Test-2

Check whether the first artist had indeed 60 occurrences during the established period, as it has we can determine our view retrieves the attributes as expected

```
SELECT p.ISVN FROM PLAYBACKS p JOIN DISCS d ON(p.ISVN = d.ISVN)
       WHERE (p.PLAYDATETIME >= SYSDATE -199 AND d.artist='Ramon Jorge Gomez'
```

lida de Script | Resultado de la Consulta

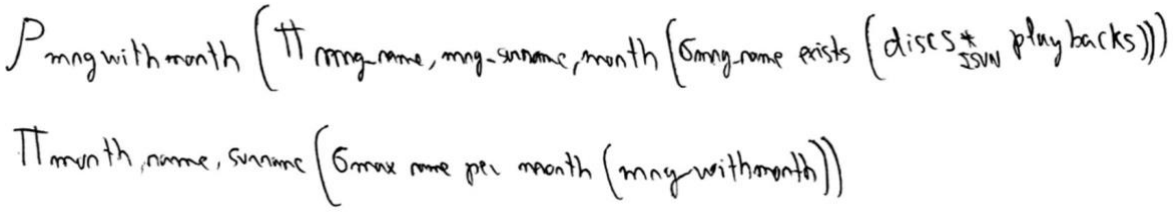
SQL | Todas las Filas Recuperadas: 60 en 0,031 segundos

ISVN

Soundboss

This view should the manager whose vinyl where more listened to for each month

- Relational Algebra



- SQL

For doing this view on SQL first I used WITH AS statement in which we selected all managers names and surnames and the month of the playbacked vinyl from them. Then I selected those atributes and I did a partition by month in which I will order in descending order (the ones that have the more on top) and then I will only select one row for each month. By selecting them we will obtain a row per month with corresponding manager.

```
CREATE OR REPLACE VIEW SoundBoss AS
WITH mngwithmonth AS
(
  SELECT MNG_NAME "Name", MNG_SURN1 "Surname",
  EXTRACT(month FROM PLAYBACKS.PLAYDATETIME) "Month"
  FROM DISCS NATURAL JOIN PLAYBACKS
  WHERE EXISTS (SELECT DISCS.MNG_NAME FROM DISCS)
)
SELECT "Month", "Name", "Surname"
FROM (SELECT "Month", "Name", "Surname", row_number()
    OVER (PARTITION BY "Month" ORDER BY COUNT(*) DESC) AS RN
    FROM mngwithmonth
    GROUP BY "Month", "Name", "Surname") mngwithmonth
WHERE RN = 1;
```

- Tests:

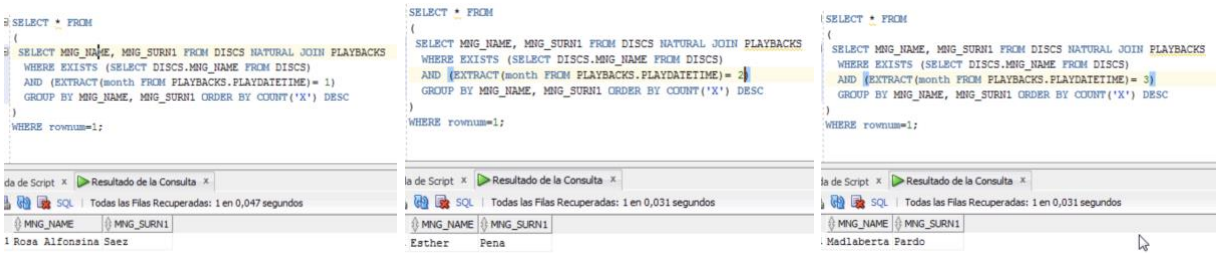
The obtain result from the above query (which is show on the image bellow this text) is what I will use to compare with the test cases. If tests were successful they will appear on green otherwise they will appear on red

Month	Name	Surname
1	Rosa Alfonsina	Saez
2	Esther	Pena
3	Madlaberta	Pardo
4	Esther	Pena
5	Atenea "Atenea "	Taipe
6	Rosa Alfonsina	Saez
7	Urbano	Canari
8	Atenea "Atenea "	Taipe
9	Armando	Apaza
10	Jorge "Jorge"	Henriquez
11	Rosa Alfonsina	Saez
12	Manuel	Perez

Test-1

Using this much simpler query I can obtain for each month the manager whose vinyl where more listened to and check if it matches with the ones I obtained. After [PLAYBACKS.PLAYDATETIME)= (*on orange in photo*)] is where the month is specified for the testing purposes.

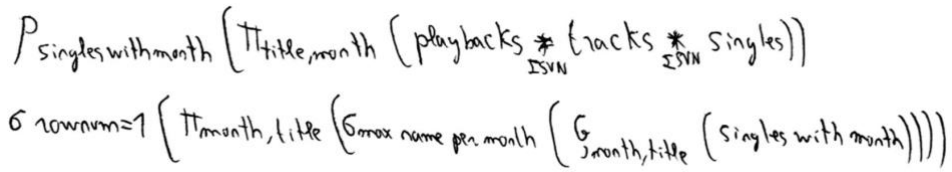
```
SELECT * FROM
(
 SELECT MNG_NAME, MNG_SURN1 FROM DISCS NATURAL JOIN PLAYBACKS
  WHERE EXISTS (SELECT DISCS.MNG_NAME FROM DISCS)
  AND (EXTRACT(month FROM PLAYBACKS.PLAYDATETIME)= 1)
  GROUP BY MNG_NAME, MNG_SURN1 ORDER BY COUNT('X') DESC
)
WHERE rownum=1;
```



Wreck-Hit

This view should for each month the single that was listened to the least

- ### Relational Algebra



- ### SQL

Similarly to previous queries for doing this view on SQL first I used WITH AS statement in which we selected all singles and the month in which they were playbacked. Then I selected those atributes and I did a partition by month in which I will order in ascending order (the ones that have the less on top) and only selecting one row for each month .

```
CREATE OR REPLACE VIEW WRECKHIT AS
WITH singleswithmonth AS
(
  SELECT TRACKS.TITLE_S "SingleName",
  EXTRACT(month FROM PLAYBACKS.PLAYDATETIME) "Month"
  FROM (PLAYBACKS JOIN TRACKS ON (TRACKS.ISVN = PLAYBACKS.ISVN))
  JOIN SINGLES ON (SINGLES.ISVN = PLAYBACKS.ISVN)
)
SELECT "Month", "SingleName"
FROM (SELECT "Month", "SingleName", row_number()
    OVER (PARTITION BY "Month" ORDER BY COUNT(*) ) AS RN
    FROM singleswithmonth
    GROUP BY "Month", "SingleName") singleswithmonth
WHERE RN = 1;
```

By doing so we will be retriving a row per month with corresponding manager.

- ### Tests:

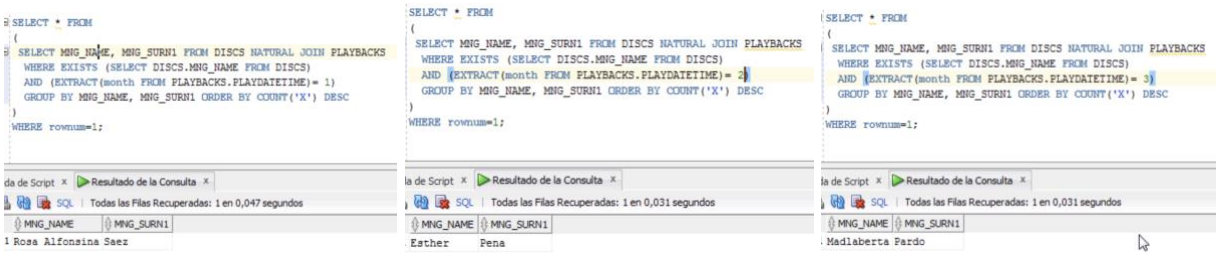
The obtain result from the above query (which is show on the image bellow this text) is what I will use to compare with the tests cases. If tests were successful they will appear on green otherwise they will appear on red

Month	SingleName
1	Fortune
2	Hombre mariachi
3	Animal de amar
4	Iris o urano
5	Bag or goblin
6	Turn and dance
7	Venom and turn
8	Die wheat
9	Vivir o genio
10	Barley or hero
11	Taste of roses
12	Woman parade

Test-1

Using this much simpler query I can obtain for each month the manager whose vinyl where more listened to and check if it matches with the ones I obtained. After [PLAYBACKS.PLAYDATETIME)= (*on orange in photo*)] is where the month is specified for the testing purposes.

```
SELECT * FROM
(
 SELECT MNG_NAME, MNG_SURN1 FROM DISCS NATURAL JOIN PLAYBACKS
  WHERE EXISTS (SELECT DISCS.MNG_NAME FROM DISCS)
  AND (EXTRACT(month FROM PLAYBACKS.PLAYDATETIME)= 1)
  GROUP BY MNG_NAME, MNG_SURN1 ORDER BY COUNT('X') DESC
)
WHERE rownum=1;
```



4. External design

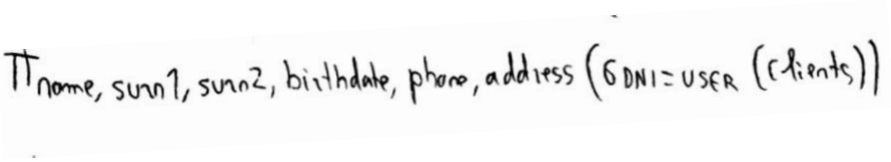
In this part of the practice I had to develop two external design with all the element that were needed. I had to implement many new structures to me as roles and variables which made this part really challenging, as stated, each external design contains a well-detailled explanation as well as the code I would use in order to develop them. As stated, tests here should be brief so I have tried to reduce them as much as I can while trying to show the functionality

Client usage

In order to develop this external design, I had to use views, triggers and roles.

First, we have to create a view for our users in which they can see all its attributes that are about them on table clients but their dni as we will be granting them access to modification and we do not want them changing what we use as their identifier. This view (client_info) contains therefor all client attributes but its DNI

- Relational Algebra



- SQL code

The code for this query and consequent view is pretty staight forward, we select those whose all atributes where the DNI is equal to the one of our user, email will not be given as we will give the user writing permissions and as we will use the email as a key we do not want the client changing it

```
CREATE OR REPLACE VIEW client_info AS
(
SELECT name, surn1, surn2, birthdate, phone, address
FROM CLIENTS WHERE DNI = USER
);
```

- Tests:

The developed view is created without problems, the view will be empty unless we do insertions with dni=user (user being fsdb285)

Second view (client_order) will containing all client orders, again we will show them all the attributes regarding their orders so they can modify them if desired after granting them the proper permisions

- Relational Algebra

- SQL code

The code for this query and consequent view is pretty staight forward aswell, we select those whose all atributes where the email is equal to the one of our user (checking by DNI)

```
CREATE OR REPLACE VIEW client_orders AS
(
SELECT isvn, order_s, qtty, delivery
FROM sale_line WHERE e_mail = (SELECT e_mail FROM CLIENTS WHERE DNI = USER)
);
```

- Tests:

Similarly to the previous view, the developed view is created without problems, but it will be empty until insertions with dni=user are made and those users had made orders

Thirdly, I had to develop a trigger working on that would instead of change on the view created for the clients, on the table of the database where the data is stored (Clients)

```
CREATE OR REPLACE TRIGGER ins_client_info
   INSTEAD OF INSERT ON client_info
   FOR EACH ROW
BEGIN
   INSERT INTO CLIENTS VALUES(:NEW.name, :NEW.surn1,
     :NEW.surn2, :NEW.birthdate, :NEW.phone, :NEW.address);
END;
```

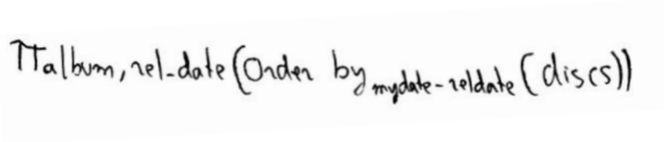
Then, another trigger similarly to the previous one had to develop working on insertion, instead of changing on the view created for the clients, on the table of the database where the data is stored (Sale_line)

```
CREATE OR REPLACE TRIGGER ins_client_orders
   INSTEAD OF INSERT ON client_orders
   FOR EACH ROW
BEGIN
   INSERT INTO SALE_LINE VALUES(:NEW.name, :NEW.surn1,
     :NEW.surn2, :NEW.birthdate, :NEW.phone, :NEW.address);
END;
```

After doing all that we also create the view of recommendations for each user. For this view the most difficult part was to create a variable taking the lastest release date of all the purchased discs if any or the sysdate if no discs had been purchaded as we have not worked before with variables. Nevertheless, after developing a working variable I found out that they cannot be used for views so I came up with another solution. The following image shows a query which selects the not null value from two values given (using nvl)

```
SELECT nvl (max(rel_date), sysdate) mydate FROM
                    (SELECT rel_date, e_mail FROM DISCS JOIN SALE_LINE ON
                    (DISCS.isvn=SALE_LINE.isvn)) WHERE e_mail=(select e_mail
                    FROM clients WHERE DNI=USER)
```

- Relational Algebra



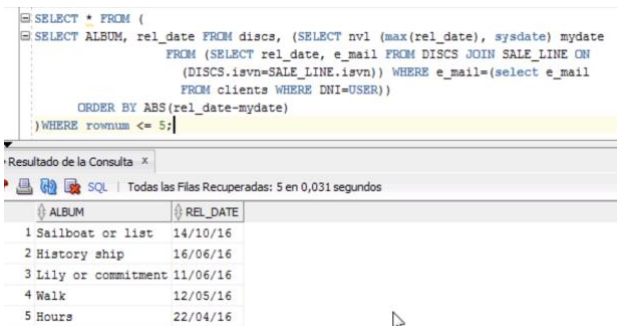
- SQL code

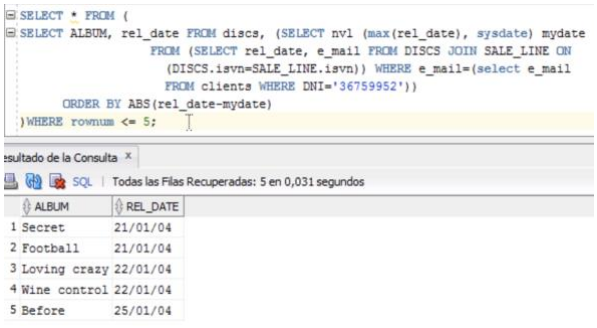
We will, select album and release date from discs cross joint the nvl query. Doing a cross joint most of the times is extremely inefficient and strongly discouraged. Nevertheless, as the nvl query will only be one row and one column always the cross joint is not inefficient but really useful. After doing the cross joint all left to do is order by the absolute value of the difference of rel_date minus mydate and select only the first 5 rows. In order to make easier the understanding of my code the previous shown query (nvl query) has been highlighted on green. Ideally that query should be on a With as statement, however their use is not allowed on views. Lastly, we will only select the first five rows as we are only interested on those. As a result, every user having a different last rel_date will have a different view.

```
CREATE OR REPLACE VIEW recomendations AS
SELECT * FROM (
SELECT ALBUM, rel_date FROM discs, (SELECT nvl (max(rel_date), sysdate) mydate
                    FROM (SELECT rel_date, e_mail FROM DISCS JOIN SALE_LINE ON
                      (DISCS.isvn=SALE_LINE.isvn)) WHERE e_mail=(select e_mail
                      FROM clients WHERE DNI=USER))
      ORDER BY ABS(rel_date-mydate)
)WHERE rownum <= 5;
```

- Tests:

As under my username there is not any order the recommendations will be made with sysdate as mydate, but we can test it takes the user last release date by replacing USER with a know user with orders, bellow you can see both outputs





Lastly, I would create a client role in which I will grant access to the different views, this would have to be done by the administrator of the database. For some of the views as recommendations will only grant selection permisions as we do not want it modified as stated on the description

```
CREATE ROLE clientR;
GRANT select,insert,update,delete ON client_info TO clientR;
GRANT select,insert,update,delete ON client_orders TO clientR;
GRANT select ON recomendations TO clientR;
--For each client done by the database administrator
GRANT clientR to USER
```

Warehouse usage

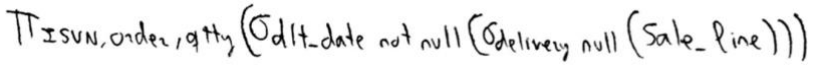
In order to develop this external design, I had to use views, triggers, roles, and add columns (by altering table).

First, I have altered the table sale_line by adding two new columns: userid and dlt_date that will be use as user identification and deletion date. Data will be automatically stored on those columns by the system when any user deletes anything from that table, which is very useful to keep a record of those who deleted sales.

```
ALTER TABLE SALE_LINE ADD
(
userid   VARCHAR2(25),
dlt_date DATE
);
```

Secondly, we have to create a view for our workers, taking into account that they should not have access to personal information.

- Relational Algebra



- SQL code

The code for this query and consequent view is pretty staight forward, we select those whose dlt_date is null because if it is not-null it means other worker (or the same worker itself) have already prepared the order line. The selection has also to choose those rows whose delivery field is not empty as that means they already have been delivered and we are only interested on unsent orders

```
CREATE OR REPLACE VIEW unsent_orders AS
(
SELECT ISVN, ORDER_S, QTTY FROM sale_line
  WHERE dlt_date is NULL AND delivery is NULL
);
```

- Tests:

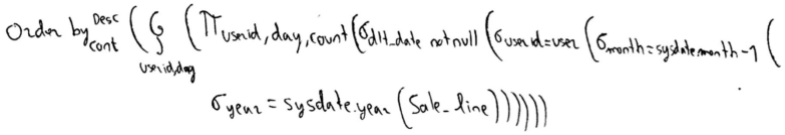
At the beginning the view will be empty as all orders included in the database have been delivered, with the insertion of non delivered orders the view will start to show data

Thirdly, I had to develop a trigger that would instead of delete on the view created for the workers, but not on the table where the data truly is (sale_line), what I would do is set on the sale_line table the userid and the sysdate on the specified row whenever the worker tried to delete on his view. This will cause the information to disappear from his view as well as for other workers. However, the information will never be truly deleted it as the administrator would have access, also having the privilege of checking who and when processed the order.

```
CREATE OR REPLACE TRIGGER delete_sale_line
INSTEAD OF DELETE ON unsent_orders
BEGIN
  UPDATE SALE_LINE set userid=USER, dlt_date=SYSDATE;
  WHERE isvn=:OLD.isvn AND order_s=:OLD.order_s AND qtty=:OLD.qtty;
END;
```

Then I developed the view productivity in which how many orders its user prepared would be indicated

- Relational Algebra



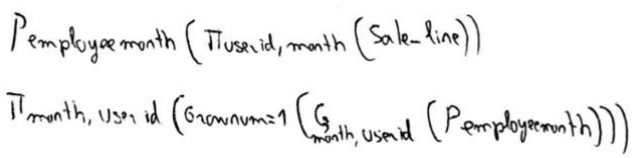
- SQL code

```
CREATE OR REPLACE VIEW PRODUCTIVITY AS
SELECT userid, EXTRACT(DAY FROM order_s) day, count('x') done_orders
  FROM SALE_LINE WHERE (dlt_date is not NULL
        AND USERID=USER AND EXTRACT(MONTH FROM order_s)=
        (EXTRACT(MONTH FROM sysdate)-1)
        AND EXTRACT(YEAR FROM order_s)=EXTRACT(YEAR FROM sysdate))
        GROUP BY userid,EXTRACT(DAY FROM order_s) ORDER BY count('x') desc;
```

The code for the view included a lot of conditions as there where many to be met. The userid which is the column in which I added the user should be equal to the user currently in that session (we only want each user to see its productivity), the selected month should be the previous one (as we want productivity only from previous month. After all those conditions were met the results will be grouped and order in descend order so each user can know which day was more productive.

After this, I developed the view employee_of_month in which the best employee of each month would be shown

- Relational Algebra



- SQL code

```
CREATE OR REPLACE VIEW employee_of_month AS
WITH employeewithmonth AS
(
  SELECT userid, EXTRACT(month FROM SALE_LINE.dlt_date) Month FROM SALE_LINE
)
SELECT Month, userid
FROM (SELECT Month, userid, row_number()
    OVER (PARTITION BY Month ORDER BY COUNT(*) ) AS RN
    FROM employeewithmonth
    GROUP BY Month, userid) employeewithmonth
WHERE RN = 1;
```

The code for the view was pretty simmilar to the code used on query 3, which make development easier, first using a WITH AS statement we would select the list with userid and month, then on the following query we would make a partition and take the employee which had more ocurrencies for each month (the one that was more productive).

Lastly, I would create a worker role in which I will grant access to the different views, this would have to be done by the administrator of the database, note than some of them as productivity are only available as select as we do not want workers modifying them

```
CREATE ROLE worker;
GRANT select,insert,update,delete ON unsent_orders TO worker;
GRANT select ON productivity TO worker;
GRANT select ON employee_of_month TO worker;
--For each user (all will belong to workers)
GRANT worker to USER
```

5. Triggers

On this section we had to develop some triggers along with views if they were necessary. Triggers are a really basic but powerful tool on databases as it can modify or cancel a given user input by a desired one. If well implemented on a database, they will protect the integrity of the database as well of the data from possible user errors which makes them something essential on any medium or large-scale database.

As it was stated on the template, only one trigger had to be implemented on SQL code however, I decided to implement more. Nevertheless, the vast majority of triggers will be only commented and described. This description of triggers will consist on explanation of the goal of the trigger, a brief text on how to implement them, views required to perform the implementation and problems with mutation.

Do not stop the music

The purpose of this trigger will be to interrupt and abort an insertion on playbacks table if any insertion time on the same radio (lower bound) plus the time of that already inserted song (upper bound) is equal in any point with the range of the song that we are trying to insert.

To do so I would create a view with the playback table as well as a new column having the time in which the song is playing. This is very likely would not be strictly necessary as with the insertion time and the playbacktime we could obtain it, but I believe adding the column would make our queries simpler to read and understand. After creating the view I would check if the start and end of our insert are (any of them) in between an start or end from previous inserts, if so an exception message will prompt the user, if not we would allow the insert. This trigger activation level would be for each row and temporality before.

Shipment expenses

This objective will require of two triggers. The first one will be used for insertion, in order ti set the expenses regarding an order. The second one will be to used when deleting a row with the aim of pass the cost to another row of the order if there were any.

To do so first we would need to alter the table sale_line to add a new column expenses and set the default value to 0, so we will only need to update it when the value is six saving some accesses to the table by doing so. The first trigger activation level would be by each row and before insertion, it will check if an expense different to 6 was trying to be inputted. If so, it will reset the value to the default one (0). Second trigger activation level would be by each row and before deletion, if it's expense value is the default one it will do nothing, otherwise it will check if there are other rows for the same order and if there are the value (6) will be set there, in any case row will be deleted

Format

There is no redundancy on the given database as format is only present on discs nevertheless, in order to do this trigger, we will suppose single and albums have also format as a column.

In order to solve this problem with triggers we would need two of them, both used to prevent the insertion of format on album and single, The one for album will activate by each row and before insertion, if format is trying to be inputted to album it will check whether its disk has it, if not it will insert it there, if it has it and it matches the one inputted it will do nothing, lastly if it does not it will prompt an error message. In any of the cases insertion of all the other fields will be made. The trigger of single will work exactly as the one on album.

Golondrinajes

The purpose of this trigger is to implement on delete set default for all our database so whenever a deletion is made fields will be set to default values instead of being deleted

If we try to implement it by means of a trigger the level of activation would be for each row and activated after deletion. However, by doing so we will have a mutating table problem, as foreign keys may be modified. So for solving it, we would need to create a temporal table in which those modifications that have to be done but that cannot be done at the time (those that try to access a table in which the flag is on) will be stored. As soon as the flag is off, which will mean the access to the conflictive table is again allowed, we will set default on those values that we previously could not that are stored on our temporal table.

No refund

The aim of this trigger will be to abort any deletion operation on an order if it has already been delivered (no refund to the customers) In order to develop this external design, I have used a view and a trigger.

The view would contain those attributes we want the user to have access to (all of the attributes but dlt_date from sale_line, more about this attribute on the trigger code comment)

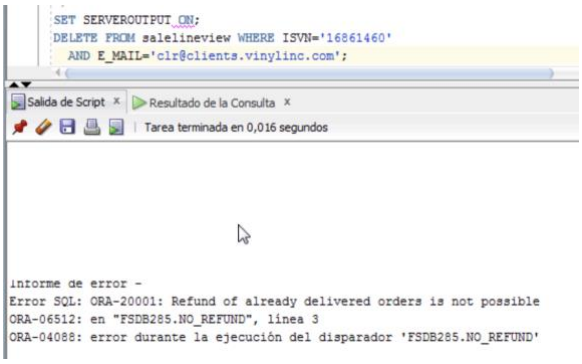
```
CREATE OR REPLACE VIEW salelineview AS(
SELECT isvn, e_mail, order_s, qtty, delivery FROM SALE_LINE
WHERE dlt_date is null
);
```

Regarding the employed trigger, its activation level will be for each row and it will be activated instead of deleting. The client (user) will only have access to the view (which will be done by roles that will be granted by the database administrator). In addition, I have chosen to alter the table sale_line adding the column dlt_date, in which the date of deletion will be added in case the user is allowed to delete the order. This will not mean we will delete the order from our database but that the user will not see it; the administrators (or anyone with the proper permissions) will be able to see the order as well as it's "deletion" date

```
CREATE OR REPLACE TRIGGER No_Refund
INSTEAD OF DELETE ON salelineview
FOR EACH ROW
BEGIN
  IF :OLD.DELIVERY is not null
    THEN RAISE_APPLICATION_ERROR(-20001, 'Refund of already delivered orders is not possible');
  ELSE UPDATE SALE_LINE set dlt_date=SYSDATE
    WHERE isvn=:OLD.isvn AND e_mail=:OLD.e_mail AND qtty=:OLD.qtty AND order_s=:OLD.order_s AND delivery=:OLD.delivery;
  END IF;
END;
```

Test-1

After the trigger compiled, I tried to delete a row from salelineview, as you can see on the images bellow the operation was aborted an it is still in both sale_line and salelineview as it should be



Linked Single

The purpose of this trigger will be to avoid insertion on single tables if the song the user pretended to make a single out of is not on the parent album.

To do so I would use a single trigger on single's table. Trigger activation for each row and before insertion. If the user tries to input a row into singles the trigger will check if for the existence of the given album (isvn_album), then check if there exists a song in that album with the same isvn as the one trying to be inputted. If so, it will allow the insertion, in any other case it will prompt an error message to the user and abort the insertion.

Empty vinyl

The purpose of this trigger will be to not allow insertion of update in a disk if it does not have any song on any of their sides.

To archive that objective, we will need one triggers when inserting or updating on disk table.

Inside the trigger we will declare a variable in which we will store the number of times different side values is count given a ISVN. As we only have two sides and we do not want to allow insertion nor update on those discs who have one or both of their sides empty if the var variable is less than two we will raise an error and end the abort the insertion or update. Otherwise we will allow the insertion/update.

```
CREATE OR REPLACE TRIGGER in_up_empty_vinyl
BEFORE INSERT OR UPDATE ON DISCS
DECLARE var NUMBER(2);
BEGIN
      SELECT DISTINCT SIDE into var FROM tracks WHERE isvn=:NEW.isvn;
  IF var < 2 THEN
    RAISE_APPLICATION_ERROR(-20001, 'DISC its empty');
  END IF;
END;
```

Periods I

The purpose of this trigger will be to delete a disk if it does not have any song on any of their sides.

To archive that objective, we will need a for each statement trigger that will check once per statement after deletion. After the user freely deletes rows from tracks, we will check if there are disks that do not have tracks pointing to them, if so, we will delete all of those disks that do not have any track, else we will do nothing

Periods II

As I understood it from the description of the practice, the purpose of this trigger will be to stop any possible update on the membership period (start_g or end_g) of any given member.

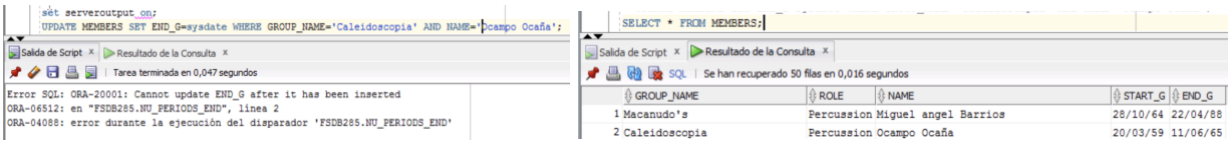
To archive that objective, we will need a for each row trigger that will be triggered when the user tries to

```
CREATE OR REPLACE TRIGGER nu_periods_end
BEFORE UPDATE OF END_G ON members
BEGIN
  RAISE_APPLICATION_ERROR(-20001, 'Cannot update END_G after it has been inserted');
END;
```

update (before update). Inside this trigger a message will prompt to the user whenever it tries to update the period. We can see bellow both triggers compiled

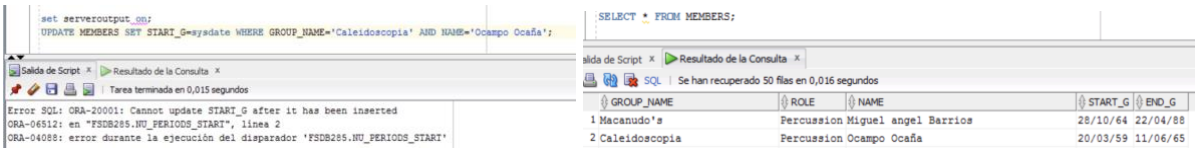
Test-1

I tried to update end_g from members, as you can see on the images bellow the operation was aborted and the message was prompt, also checking the table, nothing had been changed as it should be.



Test-2

Similarly, I tried to update start_g from members, as you can see on the images bellow the operation was aborted and the message was prompt, also checking the table, nothing had been changed as it should be.



6. Concluding Remarks

I believe after many hours of work I have arrived to a really well-though design where every query, view, external design and trigger works as expected. Many of the parts of the practice had to be redone or rethought after realizing mistakes, design flaws or things that could be better done.

I found the practice extremely difficult as I was new to many of the concepts needed to develop it, so I had to spend many hours learning or expanding my knowledge by myself which was really time consuming. I also sincerely believe the practice is excessively long and is meant for two students and not only one. I felt overwhelmed many times during the development and despite having learnt a lot I will do all I can to be in a group for the last practice as I believe it will make all the process easier for both of us and in my opinion is how we should get use to work as no one from the works alone on this kind of professional environment.

Finally, I believe to have learnt many concepts regarding SQL and databases, from variables to roles, as well as some functions or how to use conditional expressions inside triggers which will definitely turn useful in the near future as a computer scientist.