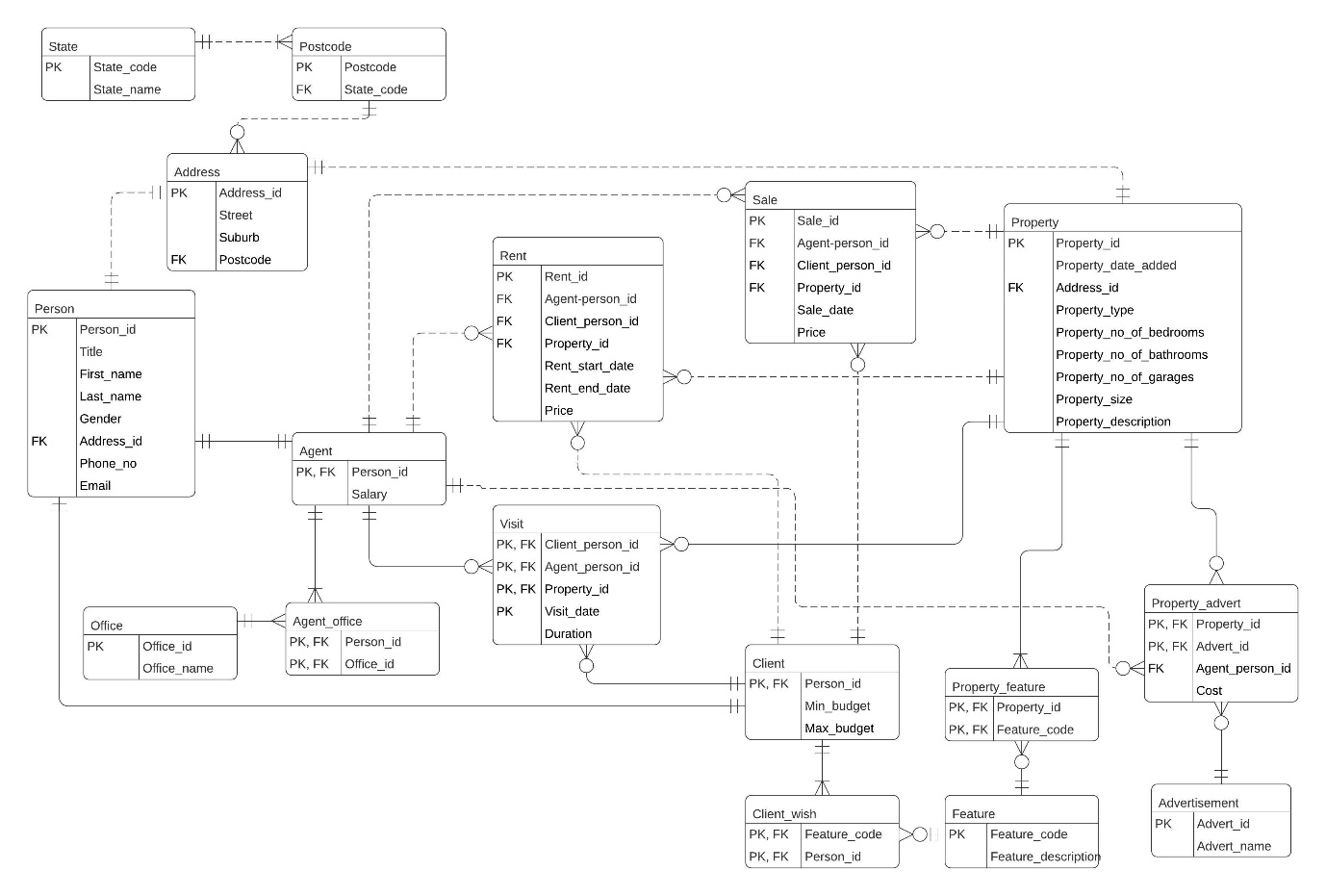
# Task C.1

## Output a – E/R Diagram



## Output b – Data Cleaning

### b1. Address

SELECT COUNT(\*)

FROM mre\_address; -- 13,204 rows



Firstly, checked for duplicate address\_id

SELECT COUNT(DISTINCT(address\_id))

FROM mre\_address; -- 13,204 rows

Secondly, checked for distinct street

SELECT COUNT(DISTINCT(street))

FROM mre\_address; -- 13,156 rows

However, among the street records, it was found that a number of records contain the same street data but different suburbs, as such it was further evaluated based on distinct street, suburb and postcode combinations.

Using the following formula, there were no addresses that were not used in the Property and Person tables.

SELECT address\_id

FROM mre\_address

WHERE NOT address\_id IN (SELECT address\_id FROM

mre\_property)

AND NOT address\_id IN (SELECT address\_id FROM mre\_person);

As such, we derived that there are no records that are required to be deleted from the address table.

### b2. Advertisement

SELECT COUNT(\*) FROM mre\_advertisement; -- 25 rows



Since that there were only 25 rows, visual inspection showed that there were no null records and the following two confirmed that there were no duplicate records:

SELECT COUNT(DISTINCT(advert\_id))

FROM mre\_advertisement; -- 25 rows

SELECT COUNT(DISTINCT(advert\_name))

FROM mre\_advertisement; -- 25 rows

We concluded that there needed no cleaning for the advertisement table.

### b3. Agent

SELECT COUNT(\*)

FROM MRE\_Agent; -- 2,469 rows



When using the following query, we found that only 2,468 people are agents, meaning that one agent is non-existent

SELECT COUNT(\*)

FROM (SELECT \* FROM MRE\_Person p, MRE\_Agent a

WHERE p.person\_id = a.person\_id); -- 2,468 rows

SELECT \*

FROM mre\_person

WHERE NOT person\_id IN (SELECT person\_id FROM mre\_person);



Using the following syntax, that row was deleted:

DELETE FROM MRE\_Agent

WHERE NOT person\_id IN (SELECT person\_id FROM MRE\_person); -- 1 row deleted

Subsequently, another error was found in 2 rows where salary was less than 0

SELECT \* FROM mre\_agent

WHERE salary < 0; -- 2 rows



DELETE FROM mre\_agent

WHERE salary < 0; -- 2 rows deleted

Checking the number of rows again:

SELECT COUNT(\*)

FROM MRE\_Agent; -- 2,466 rows



### b4. Agent\_Office

SELECT COUNT(\*)

FROM mre\_agent\_office; -- 2,529 rows



While the following syntax shows that there are multiple records for the same agent, since an agent can work at multiple office, there is no error

SELECT COUNT(DISTINCT(person\_id))

FROM mre\_agent\_office; -- 2,467 rows

However, checking whether the agent exists, the agent deleted earlier was not found and was subsequently removed.

SELECT \*

FROM mre\_agent\_office

WHERE NOT person\_id IN (SELECT person\_id FROM mre\_agent);



DELETE FROM mre\_agent\_office

WHERE NOT person\_id IN (SELECT person\_id FROM mre\_agent); -- 1 row deleted

The number of rows in the end is

SELECT COUNT(\*)

FROM mre\_agent\_office; -- 2,528 rows



### b5. Client

SELECT COUNT(\*)

FROM mre\_client; -- 3,339 rows



However, from the following syntax, there is a client that is not registered as a person

SELECT COUNT(\*)

FROM mre\_person p, mre\_client c

WHERE p.person\_id = c.person\_id; -- 3,338 rows

SELECY \*

FROM mre\_client

WHERE NOT person\_id IN (SELECT person\_id FROM mre\_person);



As such, the extra client was deleted with

DELETE FROM mre\_client

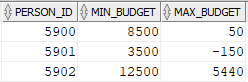
WHERE NOT person\_id IN (SELECT person\_id FROM mre\_person); -- 1 row deleted

Furthermore, it was found that few records had the max\_budget lower than the min\_budget, as well as min budget being negative. These rows were then deleted

SELECT \*

FROM mre\_client

WHERE max\_budget < min\_budget; -- 3 rows



DELETE FROM mre\_client

WHERE max\_budget < min\_budget; -- 3 rows deleted

Checking for any min\_budget or max\_budget being negative yield no results.

Checking the number of rows again:

SELECT COUNT(\*)

FROM mre\_client; -- 3,338 rows



### b6. Client\_Wish

SELECT COUNT(\*)

FROM mre\_client\_wish; -- 1,204 rows



Since a single client can have many wishes, checking for distinct clients is useless. Checking for repeated person\_id and feature code also proved no duplicates. As such, no changes were needed.

### b7. Feature

SELECT COUNT(\*)

FROM mre\_feature; -- 726 rows



Checking for duplicate feature\_id and feature\_description showed no duplicates.

### b8. Office

SELECT COUNT(\*)

FROM mre\_office; -- 1,177 rows



Checking for duplicates of office\_id and office\_name showed no duplicates.

### b9. Person

SELECT COUNT(\*)

FROM mre\_person; -- 7,000 rows



Looking for duplicates person\_id, 4 deplicates were found for person\_id = 6995

SELECT COUNT(DISTINCT(person\_id))

FROM MRE\_Person; -- 6,997 rows

SELECT person\_id

FROM mre\_person GROUP BY person\_id HAVING COUNT(\*) > 1;



Using the following syntax, the duplicate records were deleted:

DELETE FROM MRE\_Person p

WHERE rowid > (SELECT MIN(rowid)FROM MRE\_Person p2

WHERE p.person\_id = p2.person\_id); -- 3 rows deleted

Checking through phone\_no and email showed no other duplicates, the final number is:

SELECT COUNT(\*)

FROM mre\_person; - 6,997 rows



### b10. Postcode

SELECT COUNT(\*)

FROM mre\_postcode; -- 689 rows



No duplicate or nulls were found in the postcode table.

### b11. Property

SELECT COUNT(\*)

FROM mre\_property; -- 6,226 rows



There were a large number of records that were duplicated

SELECT COUNT(DISTINCT(property\_id)) FROM mre\_property; -- 6,208 rows

SELECT \*

FROM mre\_property p

WHERE rowid > (SELECT MIN(rowid)FROM mre\_property p2

WHERE p.property\_id = p2.property\_id);



The following syntax was used to delete the duplicate rows

DELETE FROM MRE\_Property p

WHERE rowid > (SELECT MIN(rowid)FROM MRE\_Property p2

WHERE p.property\_id = p2.property\_id); -- 18 rows deleted

The new number of rows is:

SELECT COUNT(\*)

FROM mre\_property; -- 6,208 rows



### b12. Property\_Advert

SELECT COUNT(\*)

FROM mre\_property\_advert; -- 3,646 rows



No duplicate records or null records were found in this table

### b13. Property\_Feature

SELECT COUNT(\*)

FROM mre\_property\_feature; -- 30,373 rows



No duplicate or null records were found

### b14. Rent

SELECT COUNT(\*)

FROM mre\_rent; --3,284 rows



Records were checked for duplicate and non were found but one record was found for rent\_end\_date before rent\_start\_date

SELECT \*

FROM mre\_rent

WHERE rent\_end\_date <= rent\_start\_date;



The record was then deleted

DELETE FROM MRE\_Rent WHERE rent\_id IN (SELECT rent\_id FROM MRE\_Rent WHERE rent\_end\_date < rent\_start\_date); -- 1 row deleted

SELECT COUNT(\*)

FROM MRE\_Rent; -- 3,283 rows



### b15. Sale

SELECT COUNT(\*)

FROM mre\_sale; -- 2,925 rows



The following syntax was used and found that a number of records had null client\_person\_id and sale\_date. However, as these rows may be kept for record to show agent client relationships for unsold properties, the rows are temporarily kept.

### b16. State

SELECT \*

FROM mre\_state; -- 9 rows

There were little number of rows so visual inspection was possible and one row was identified as NULL state\_code and UNKNOWN state\_name which was promptly deleted

DELETE FROM MRE\_State

WHERE state\_code IS NULL; -- 1 row deleted

New number of records is:

SELECT COUNT(\*)

FROM mre\_state; -- 8 rows



### b17. Visit

SELECT COUNT(\*)

FROM mre\_visit; -- 575 rows



A record was found that the agent or client did not exist in mre\_agent or mre\_client

SELECT \*

FROM mre\_visit

WHERE NOT agent\_person\_id IN (SELECT person\_ID FROM mre\_agent)

OR NOT client\_person\_id IN (SELECT person\_id FROM mre\_client); -- 1 row



That record was promptly deleted

DELETE

FROM mre\_visit

WHERE NOT agent\_person\_id IN (SELECT person\_ID FROM mre\_agent)

OR NOT client\_person\_id IN (SELECT person\_id FROM mre\_client); -- 1 row deleted

Updated number of rows is:

SELECT COUNT(\*) FROM mre\_visit; -- 574 rows



### b18. Special Case

We checked for person records that did not reference any addresses. Using the following code, we found that there was a person record which address\_id did not exist and the record contains majority of null fields. This record could not be detected through using ‘IS NULL’ because the fields while displaying null, was actually a string ‘null’.

SELECT \*

FROM mre\_person

WHERE NOT address\_id IN (SELECT address\_id FROM mre\_address);



As a safety precaution, we checked through Agent, Client, Client\_Wish to this person\_id.

We found no such persin in the Agent table but there exists a normal record in the Client table.



We continued to check the Client\_Wish table and found one record of this person:



Looking into the Feature table, we found that feature\_code = 726 is labelled as Fake Feature.



After taking the above-mentioned factors into consideration, we decided to delete person\_id = 7001 from the Person, Client and Client\_Wish table as well as feature\_code = 726 from the feature table.

### b19. Summary

After analysing the operational database, the following errors were identified:

1. Agent table person\_id NOT IN Person table person\_id – 1 row deleted
2. Agent salary < 0 – 2 rows deleted
3. Agent person\_id NOT IN Person table person\_id – 1 row deleted
4. Client table person\_id NOT IN Person table person\_id – 1 row deleted
5. Client max\_budget < min\_budget – 3 rows deleted
6. Duplicate person record – 3 rows deleted
7. Duplicate property record – 18 rows deleted
8. Rent rent\_end\_date < rent\_start\_date – 1 row deleted
9. State state\_code IS NULL – 1 row deleted
10. Visit table agent\_person\_id or client\_person\_id NOT IN Person table person\_id – 1 row deleted
11. Special Case: Person address\_id non-existent – 1 row deleted

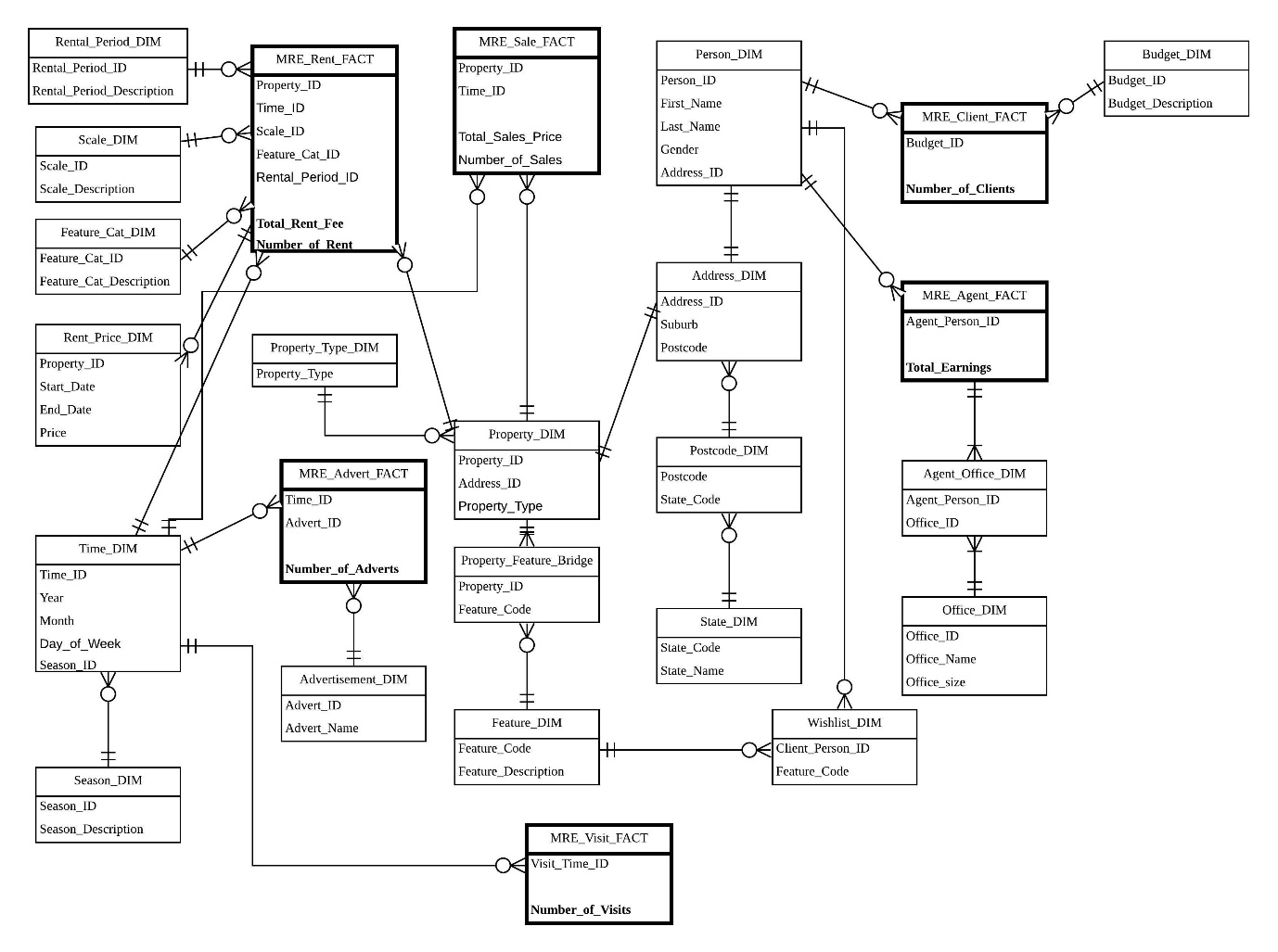
Client record referencing above – 1 row deleted

Client\_wish record referencing above – 1 row deleted

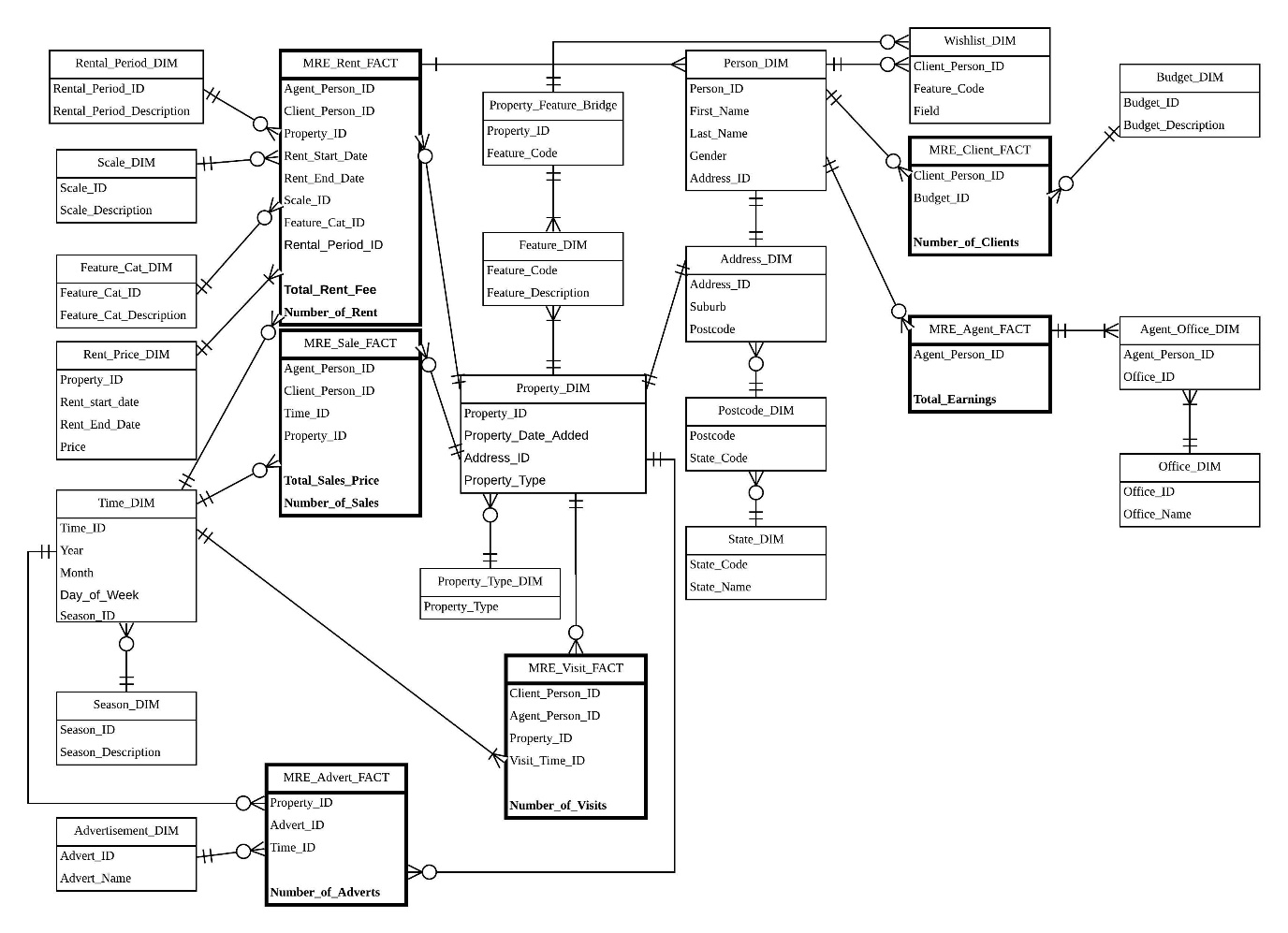
Feature record ‘Fake Feature’ – 1 row deleted

## Output c – Star Schemas

### c1 Level 2



### c2 Level 0



## Output d – Hierarchy or non-Hierarchy Explanation

We used hierarchy for address as it allows easy drill down into data for viewing different granularity. It is preferred compared to non-hierarchy as it allows optional usage of the lower level dimensions when needed but at the same time reduce the storage capacity in higher level tables as state names is longer than using a number to represent them.

## Output e - Temporal Dimension SCD Type Explanation

## Output f – Differences between the Two Versions of Star/Snowflake Schema

# Task 2

## Output a – SQL Statement for Level 2 Star Schema

-- Task c 2b)

-- Level 2 multi-fact star schema

DROP TABLE MRE\_scale\_DIM\_l2 PURGE;

DROP TABLE MRE\_feature\_cat\_DIM\_l2 PURGE;

DROP TABLE MRE\_property\_dim\_l2 PURGE;

DROP TABLE MRE\_property\_feature\_bridge\_l2 PURGE;

DROP TABLE MRE\_feature\_dim\_l2 PURGE;

DROP TABLE MRE\_property\_type\_dim\_l2 PURGE;

DROP TABLE MRE\_address\_dim\_l2 PURGE;

DROP TABLE MRE\_postcode\_dim\_l2 PURGE;

DROP TABLE MRE\_state\_dim\_l2 PURGE;

DROP TABLE MRE\_advertisement\_dim\_l2 PURGE;

DROP TABLE MRE\_person\_dim\_l2 PURGE;

DROP TABLE MRE\_agent\_office\_dim\_l2 PURGE;

DROP TABLE MRE\_office\_dim\_l2 PURGE;

DROP TABLE MRE\_budget\_dim\_l2 PURGE;

DROP TABLE MRE\_rental\_period\_dim\_l2 PURGE;

DROP TABLE MRE\_wishlist\_dim\_l2 PURGE;

DROP TABLE MRE\_rent\_price\_dim\_l2 PURGE;

DROP TABLE MRE\_temp\_time\_dim\_l2 PURGE;

DROP TABLE MRE\_time\_dim\_l2 PURGE;

DROP TABLE MRE\_season\_dim\_l2 PURGE;

DROP TABLE MRE\_agent\_fact\_l2 PURGE;

DROP TABLE MRE\_temp\_client\_l2 PURGE;

DROP TABLE MRE\_client\_fact\_l2 PURGE;

DROP TABLE MRE\_temp\_rent\_fact\_l2 PURGE;

DROP TABLE MRE\_rent\_fact\_l2 PURGE;

DROP TABLE MRE\_temp\_visit\_l2 PURGE;

DROP TABLE MRE\_visit\_fact\_l2 PURGE;

DROP TABLE MRE\_temp\_sale\_fact\_l2 PURGE;

DROP TABLE MRE\_sale\_fact\_l2 PURGE;

DROP TABLE MRE\_temp\_advert\_l2 PURGE;

DROP TABLE MRE\_advert\_fact\_l2 PURGE;

-- Dimension tables

-- Scale dimension

create table mre\_scale\_dim\_l2 (

scale\_id numeric(1),

scale\_description char(20));

insert into mre\_scale\_dim\_l2 values(1, 'extra small');

insert into mre\_scale\_dim\_l2 values(2, 'small');

insert into mre\_scale\_dim\_l2 values(3, 'medium');

insert into mre\_scale\_dim\_l2 values(4, 'large');

insert into mre\_scale\_dim\_l2 values(5, 'extra large');

-- Feature catagory dimension

create table mre\_feature\_cat\_dim\_l2(

feature\_cat\_id numeric(1),

feature\_cat\_description char(15));

insert into mre\_feature\_cat\_dim\_l2 values(1,'basic');

insert into mre\_feature\_cat\_dim\_l2 values(2,'standard');

insert into mre\_feature\_cat\_dim\_l2 values(3,'luxurious');

-- Property dimension

create table mre\_property\_dim\_l2

as select property\_id, address\_id, property\_type

from mre\_property;

-- Property feature bridge

create table mre\_property\_feature\_bridge\_l2

as select distinct \*

from mre\_property\_feature;

-- feature dim

create table mre\_feature\_dim\_l2

as select distinct \*

from mre\_feature;

-- property type dimension

create table mre\_property\_type\_dim\_l2

as select distinct(property\_type)

from mre\_property;

-- Address dim

create table mre\_address\_dim\_l2

as select distinct address\_id, suburb, postcode

from mre\_address;

-- postcode dim

create table mre\_postcode\_dim\_l2

as select distinct \*

from mre\_postcode;

-- state dim

create table mre\_state\_dim\_l2

as select \*

from mre\_state;

-- Advertisment dim

create table mre\_advertisement\_dim\_l2

as select distinct \*

from mre\_advertisement;

-- person dim

create table mre\_person\_dim\_l2

as select person\_id, first\_name, last\_name, gender, address\_id

from mre\_person;

-- agent office dim

create table mre\_agent\_office\_dim\_l2

as select distinct person\_id as agent\_person\_id, office\_id

from mre\_agent\_office;

-- office dim

create table mre\_office\_dim\_l2

as select \*

from mre\_office;

alter table mre\_office\_dim\_l2

add office\_size char(10);

update mre\_office\_dim\_l2 t

set office\_size =

(select case

when count(person\_id) < 4 then 'small'

when count(person\_id) between 4 and 12 then 'medium'

else 'big'

end

from mre\_agent\_office ao

where t.office\_id = ao.office\_id);

-- Budget dimension

create table mre\_budget\_dim\_l2(

budget\_id char(2),

budget\_description varchar(100));

insert into mre\_budget\_dim\_l2 values ('l', 'Budget between 0 and 1000');

insert into mre\_budget\_dim\_l2 values ('m', 'Budget between 1001 and 100000');

insert into mre\_budget\_dim\_l2 values ('h', 'Budget more than 100001');

-- Rental period DIM

create table mre\_rental\_period\_dim\_l2(

rental\_period\_id numeric(2),

rental\_period\_description varchar(50));

insert into mre\_rental\_period\_dim\_l2 values (1, 'short');

insert into mre\_rental\_period\_dim\_l2 values (2, 'medium');

insert into mre\_rental\_period\_dim\_l2 values (3, 'long');

-- whishlist dim

create table mre\_wishlist\_dim\_l2

as select distinct \*

from mre\_client\_wish;

-- Rent price dimension

create table mre\_rent\_price\_dim\_l2

as select property\_id, rent\_start\_date, rent\_end\_date, price

from mre\_rent;

-- Time dimension

create table mre\_temp\_time\_dim\_l2

as select \*

from (select distinct sale\_date as dates from mre\_sale

where sale\_date is not null

union

select distinct rent\_start\_date from mre\_rent

where rent\_start\_date is not null

union

select distinct rent\_end\_date from mre\_rent

where rent\_end\_date is not null

);

alter table mre\_temp\_time\_dim\_l2

add (

time\_id varchar(20),

Year numeric(4),

Month numeric(2),

Season\_id numeric(1));

update mre\_temp\_time\_dim\_l2

set time\_id = to\_char(dates, 'YYYYMMDY'),

year = to\_char(dates, 'YYYY'),

month = to\_char(dates, 'MM');

update mre\_temp\_time\_dim\_l2

set season\_id =

case

when month between 3 and 5 then 1

when month between 6 and 8 then 2

when month between 9 and 11 then 3

else 4

end;

create table mre\_time\_dim\_l2

as select DISTINCT(time\_id), year, month, season\_id

from mre\_temp\_time\_dim\_l2;

-- Season DIM

create table mre\_season\_dim\_l2(

season\_id numeric(1),

season\_description char(10));

insert into mre\_season\_dim\_l2 values(1, 'Spring');

insert into mre\_season\_dim\_l2 values(2, 'Summer');

insert into mre\_season\_dim\_l2 values(3, 'Autumn');

insert into mre\_season\_dim\_l2 values(4, 'Winter');

-- Fact tables

-- Agent fact table

create table mre\_agent\_fact\_l2

as select a.person\_id as agent\_person\_id, sum(nvl(s.price, 0)) + nvl(sum(nvl(r.price, 0)/7\*(r.rent\_end\_date - r.rent\_start\_date)), 0) as total\_earnings

from mre\_agent a, mre\_sale s, mre\_rent r

where a.person\_id = s.agent\_person\_id (+)

and a.person\_id = r.agent\_person\_id (+)

group by a.person\_id;

-- client fact table

create table mre\_temp\_client\_l2

as select max\_budget from mre\_client;

alter table mre\_temp\_client\_l2

add budget\_id char(2);

update mre\_temp\_client\_l2

set budget\_id = case

when max\_budget between 0 and 1000 then 'l'

when max\_budget between 1001 and 100000 then 'm'

else 'h' end;

create table mre\_client\_fact\_l2

as select budget\_id , count(\*) as total\_number\_of\_client

from mre\_temp\_client\_l2

group by budget\_id;

-- rent fact

create table mre\_temp\_rent\_fact\_l2

as select distinct

r.property\_id ,

r.rent\_start\_date as dates,

p.property\_no\_of\_bedrooms,

COUNT(\*) as number\_of\_features,

r.price,

r.rent\_end\_date,

r.rent\_start\_date,

count(distinct(rent\_id)) as num\_of\_rent

from mre\_rent r, mre\_property p, mre\_property\_feature f

where r.property\_id = p.property\_id

and p.property\_id = f.property\_id

and r.rent\_start\_date is not null

GROUP BY r.property\_id, p.property\_no\_of\_bedrooms, r.price, r.rent\_end\_date, r.rent\_start\_date;

alter table mre\_temp\_rent\_fact\_l2 add (

time\_id varchar(20),

scale\_id numeric(1),

feature\_cat\_id numeric(1));

update mre\_temp\_rent\_fact\_l2

set time\_id = to\_char(rent\_start\_date, 'YYYYMMDY'),

scale\_id =

case

when property\_no\_of\_bedrooms between 0 and 1 then 1

when property\_no\_of\_bedrooms between 2 and 3 then 2

when property\_no\_of\_bedrooms between 4 and 6 then 3

when property\_no\_of\_bedrooms between 7 and 10 then 4

else 5

end;

update mre\_temp\_rent\_fact\_l2 t

set feature\_cat\_id =

(case when number\_of\_features < 10 then 1

when number\_of\_features between 10 and 20 then 2

else 3

end);

create table mre\_rent\_fact\_l2

as select property\_id, time\_id, scale\_id, feature\_cat\_id, (price / 7 \*(rent\_end\_date - rent\_start\_date)) as total\_rent\_fee, num\_of\_rent as number\_of\_rent

from mre\_temp\_rent\_fact\_l2;

-- visit fact

create table mre\_temp\_visit\_l2

as select visit\_date

from mre\_visit;

alter table mre\_temp\_visit\_l2

add visit\_time\_id varchar(20);

update mre\_temp\_visit\_l2

set visit\_time\_id = to\_char(visit\_date, 'YYYYMMDY');

create table mre\_visit\_fact\_l2

as select visit\_time\_id, count(\*) as number\_of\_visit

from mre\_temp\_visit\_l2

group by visit\_time\_id;

-- sale fact

create table mre\_temp\_sale\_fact\_l2

as select s.property\_id, s.sale\_date, p.property\_type, s.price

from mre\_sale s, mre\_property p

where s.property\_id = p.property\_id

and sale\_date is not null;

alter table mre\_temp\_sale\_fact\_l2 add (

time\_id varchar(20));

set define off;

update mre\_temp\_sale\_fact\_l2

set time\_id = to\_char(sale\_date, 'YYYYMMDY');

create table mre\_sale\_fact\_l2

as select property\_id, time\_id, sum(price) as total\_sales\_price, count(\*) as number\_of\_sales

from mre\_temp\_sale\_fact\_l2

group by property\_id, time\_id;

-- Advert fact

create table mre\_temp\_advert\_l2

as select distinct a.advert\_id, p.property\_date\_added

from mre\_property\_advert a, mre\_property p

where p.property\_id = a.property\_id;

alter table mre\_temp\_advert\_l2

add time\_id varchar(20);

update mre\_temp\_advert\_l2

set time\_id = to\_char(property\_date\_added, 'YYYYMMDY');

create table mre\_advert\_fact\_l2

as select time\_id, advert\_id, count(\*) as number\_of\_adverts

from mre\_temp\_advert\_l2

group by time\_id, advert\_id;

commit;

## Output b – SQL Statement for Level 0 Star Schema

-- Task C 2b)

-- Level 0 multi-fact star schema

DROP TABLE MRE\_Scale\_DIM\_l0 PURGE;

DROP TABLE MRE\_Feature\_Cat\_DIM\_l0 PURGE;

DROP TABLE MRE\_Property\_DIM\_l0 PURGE;

DROP TABLE MRE\_Property\_Feature\_Bridge\_l0 PURGE;

DROP TABLE MRE\_Feature\_DIM\_l0 PURGE;

DROP TABLE MRE\_Wishlist\_DIM\_L0 PURGE;

DROP TABLE MRE\_Property\_Type\_DIM\_l0 PURGE;

DROP TABLE MRE\_Address\_DIM\_l0 PURGE;

DROP TABLE MRE\_Postcode\_DIM\_l0 PURGE;

DROP TABLE MRE\_State\_DIM\_l0 PURGE;

DROP TABLE MRE\_Advertisement\_DIM\_l0 PURGE;

DROP TABLE MRE\_Person\_DIM\_l0 PURGE;

DROP TABLE MRE\_Agent\_Office\_DIM\_l0 PURGE;

DROP TABLE MRE\_Office\_DIM\_l0 PURGE;

DROP TABLE MRE\_Office\_Size\_DIM\_L0 PURGE;

DROP TABLE MRE\_Office\_TempDIM\_L0 PURGE;

DROP TABLE MRE\_Budget\_DIM\_l0 PURGE;

DROP TABLE MRE\_Rental\_Period\_DIM\_l0 PURGE;

DROP TABLE MRE\_Rent\_Price\_DIM\_l0 PURGE;

DROP TABLE MRE\_Season\_DIM\_l0 PURGE;

DROP TABLE MRE\_Time\_DIM\_l0 PURGE;

DROP TABLE MRE\_Sale\_FACT\_l0 PURGE;

DROP TABLE MRE\_Rent\_TempFACT\_L0 PURGE;

DROP TABLE MRE\_Rent\_FACT\_l0 PURGE;

DROP TABLE MRE\_Client\_TempFACT\_L0 PURGE;

DROP TABLE MRE\_Client\_FACT\_l0 PURGE;

DROP TABLE MRE\_Agent\_FACT\_l0 PURGE;

DROP TABLE MRE\_Visit\_FACT\_l0 PURGE;

DROP TABLE MRE\_Advert\_FACT\_l0 PURGE;

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

-- Implement dimension tables --

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

-- MRE\_Scale\_DIM\_L0

CREATE TABLE MRE\_Scale\_DIM\_L0 (

Scale\_ID NUMBER,

Scale\_Description VARCHAR2(100)

);

INSERT INTO MRE\_Scale\_DIM\_L0 VALUES (1, 'Extra small: <= 1 bedroom');

INSERT INTO MRE\_Scale\_DIM\_L0 VALUES (2, 'Small: 2-3 bedrooms');

INSERT INTO MRE\_Scale\_DIM\_L0 VALUES (3, 'Medium: 3-6 bedrooms');

INSERT INTO MRE\_Scale\_DIM\_L0 VALUES (4, 'Large: 6-10 bedrooms');

INSERT INTO MRE\_Scale\_DIM\_L0 VALUES (5, 'Extra large: > 10 bedrooms');

-- MRE\_Feature\_CAT\_DIM\_L0

CREATE TABLE MRE\_Feature\_CAT\_DIM\_L0 (

Feature\_CAT\_ID NUMBER,

Feature\_CAT\_Description VARCHAR2(100)

);

INSERT INTO MRE\_Feature\_CAT\_DIM\_L0 VALUES (1, 'Very basic: < 10 features');

INSERT INTO MRE\_Feature\_CAT\_DIM\_L0 VALUES (2, 'Standard: 10-20 features');

INSERT INTO MRE\_Feature\_CAT\_DIM\_L0 VALUES (3, 'Luxurious: > 20 features');

-- MRE\_Property\_DIM\_L0

CREATE TABLE MRE\_Property\_DIM\_L0 AS (

SELECT

p.Property\_ID,

p.Property\_Date\_Added,

p.address\_id,

p.property\_type

FROM MRE\_Property p

);

-- Property\_Feature\_Bridge\_L0

CREATE TABLE MRE\_Property\_Feature\_Bridge\_L0 AS (

SELECT DISTINCT \* FROM MRE\_Property\_Feature

);

-- MRE\_Feature\_DIM\_L0

CREATE TABLE MRE\_Feature\_DIM\_L0 AS (

SELECT DISTINCT \* FROM MRE\_Feature

);

-- MRE\_Wishlist\_DIM\_L0

CREATE TABLE MRE\_Wishlist\_DIM\_L0 AS (

SELECT

Person\_ID AS Client\_Person\_ID,

Feature\_Code

FROM MRE\_Client\_Wish

);

-- MRE\_Property\_Type\_DIM\_L0

CREATE TABLE MRE\_Property\_Type\_DIM\_L0 AS (

SELECT DISTINCT(property\_type)

FROM mre\_property

);

-- MRE\_Address\_DIM\_L0

CREATE TABLE MRE\_Address\_DIM\_L0 AS (

SELECT DISTINCT

Address\_ID,

Street,

Suburb,

Postcode

FROM MRE\_Address

);

-- MRE\_Postcode\_DIM\_L0

CREATE TABLE MRE\_Postcode\_DIM\_L0 AS (

SELECT DISTINCT \* FROM MRE\_Postcode

);

-- MRE\_State\_DIM\_L0

CREATE TABLE MRE\_State\_DIM\_L0 AS (

SELECT DISTINCT \* FROM MRE\_State

);

-- MRE\_Advertisement\_DIM\_L0

CREATE TABLE MRE\_Advertisement\_DIM\_L0 AS (

SELECT DISTINCT \* FROM MRE\_Advertisement

);

-- MRE\_Person\_DIM\_L0

CREATE TABLE MRE\_Person\_DIM\_L0 AS (

SELECT DISTINCT

Person\_ID,

First\_Name,

Last\_Name,

Gender,

Address\_ID

FROM MRE\_Person

);

-- MRE\_Agent\_Office\_DIM\_L0

CREATE TABLE MRE\_Agent\_Office\_DIM\_L0 AS (

SELECT DISTINCT

Person\_ID AS Agent\_Person\_ID,

Office\_ID

FROM MRE\_Agent\_Office

);

-- MRE\_Office\_Size\_DIM\_L0

CREATE TABLE MRE\_Office\_Size\_DIM\_L0 (

Office\_Size\_ID NUMBER,

Office\_Size\_Description VARCHAR2(60)

);

INSERT INTO MRE\_Office\_Size\_DIM\_L0 VALUES (1, 'Small: < 4 employees');

INSERT INTO MRE\_Office\_Size\_DIM\_L0 VALUES (2, 'Medium: 4 - 12 employees');

INSERT INTO MRE\_Office\_Size\_DIM\_L0 VALUES (3, 'Big: > 12 employees');

-- MRE\_Office\_TempDIM\_L0

CREATE TABLE MRE\_Office\_TempDIM\_L0 AS (

SELECT DISTINCT

ao.Office\_ID,

o.Office\_Name,

COUNT(ao.Person\_ID) AS Num\_of\_Employees

FROM MRE\_Office o, MRE\_Agent\_Office ao

WHERE o.Office\_ID = ao.Office\_ID

GROUP BY ao.Office\_ID, o.Office\_Name

);

ALTER TABLE MRE\_Office\_TempDIM\_L0

ADD Office\_Size\_ID NUMBER;

UPDATE MRE\_Office\_TempDIM\_L0

SET Office\_Size\_ID =

(CASE

WHEN Num\_of\_Employees < 4 THEN 1

WHEN Num\_of\_Employees BETWEEN 4 AND 12 THEN 2

WHEN Num\_of\_Employees > 12 THEN 3

END);

CREATE TABLE MRE\_Office\_DIM\_L0 AS (

SELECT

Office\_ID,

Office\_Name,

Office\_Size\_ID

FROM MRE\_Office\_TempDIM\_L0

);

-- MRE\_Budget\_DIM\_L0

CREATE TABLE MRE\_Budget\_DIM\_L0 (

Budget\_ID NUMBER,

Budget\_Description VARCHAR2(100),

Min\_Budget NUMBER,

Max\_Budget NUMBER

);

INSERT INTO MRE\_Budget\_DIM\_L0 VALUES (1, 'Low [0 to 1,000]', 0, 1000);

INSERT INTO MRE\_Budget\_DIM\_L0 VALUES (3, 'Medium [1,001 to 100,000]', 1001, 100000);

INSERT INTO MRE\_Budget\_DIM\_L0 VALUES (5, 'High [100,001 to 10,000,000]', 100001, 10000000);

-- MRE\_Rental\_Period\_DIM\_L0

CREATE TABLE MRE\_Rental\_Period\_DIM\_L0 (

Rental\_Period\_ID NUMBER,

Rental\_Period\_Description VARCHAR2(60)

);

INSERT INTO MRE\_Rental\_Period\_DIM\_L0 VALUES (1, 'Short: < 6 months');

INSERT INTO MRE\_Rental\_Period\_DIM\_L0 VALUES (2, 'Medium: 6 - 12 months');

INSERT INTO MRE\_Rental\_Period\_DIM\_L0 VALUES (3, 'Long: > 12 months');

-- MRE\_Rent\_Price\_DIM\_L0

CREATE TABLE MRE\_Rent\_Price\_DIM\_L0 AS (

SELECT DISTINCT

Property\_ID,

Rent\_Start\_Date AS Start\_date,

Rent\_End\_Date AS End\_date,

Price

FROM MRE\_Rent

);

-- MRE\_Season\_DIM\_L0

CREATE TABLE MRE\_Season\_DIM\_L0 (

Season\_ID NUMBER,

Season\_Description VARCHAR2(10)

);

INSERT INTO MRE\_Season\_DIM\_L0 VALUES (1, 'Summer');

INSERT INTO MRE\_Season\_DIM\_L0 VALUES (2, 'Autumn');

INSERT INTO MRE\_Season\_DIM\_L0 VALUES (3, 'Winter');

INSERT INTO MRE\_Season\_DIM\_L0 VALUES (4, 'Spring');

-- MRE\_Time\_DIM\_L0

CREATE TABLE MRE\_Time\_DIM\_L0 AS (

SELECT DISTINCT

TO\_CHAR(d.dates, 'YYYYMMDY') AS Time\_ID,

TO\_CHAR(d.dates, 'YYYY') AS Year,

TO\_NUMBER(TO\_CHAR(d.dates, 'MM'), '99') AS Month,

TO\_CHAR(d.dates, 'DY') AS Day\_of\_Week

FROM (

SELECT DISTINCT Sale\_Date AS DATES FROM MRE\_Sale

WHERE Sale\_Date IS NOT NULL

UNION

SELECT DISTINCT Rent\_Start\_Date FROM MRE\_Rent

WHERE Rent\_Start\_Date IS NOT NULL

UNION

SELECT DISTINCT Rent\_End\_Date FROM MRE\_Rent

WHERE Rent\_End\_Date IS NOT NULL

) d

);

ALTER TABLE MRE\_Time\_DIM\_L0

ADD Season\_ID NUMBER;

UPDATE MRE\_Time\_DIM\_L0

SET Season\_ID =

(CASE

WHEN Month = 12 OR Month BETWEEN 1 AND 2 THEN 1

WHEN Month BETWEEN 3 AND 5 THEN 2

WHEN Month BETWEEN 6 AND 8 THEN 3

WHEN Month BETWEEN 9 AND 11 THEN 4

END);

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

-- Implement fact tables --

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

-- MRE\_Sale\_FACT\_L0

CREATE TABLE MRE\_Sale\_FACT\_L0 AS (

SELECT

s.Agent\_Person\_ID,

s.Client\_Person\_ID,

TO\_CHAR(s.Sale\_Date, 'YYYYMMDY') AS Time\_ID,

s.Property\_ID,

s.Price AS Total\_Sales\_Price,

COUNT(s.Sale\_ID) AS Number\_of\_Sales

FROM MRE\_Sale s, MRE\_Property p

WHERE s.Property\_ID = p.Property\_ID

AND s.Client\_Person\_ID IS NOT NULL

AND s.Sale\_Date IS NOT NULL

GROUP BY s.Agent\_Person\_ID, s.Client\_Person\_ID, TO\_CHAR(s.Sale\_Date, 'YYYYMMDY'), s.Property\_ID, s.Price

);

-- MRE\_Rent\_FACT\_L0

CREATE TABLE MRE\_Rent\_TempFACT\_L0 AS (

SELECT

r.Agent\_Person\_ID,

r.Client\_Person\_ID,

r.Property\_ID,

r.Rent\_Start\_Date,

r.Rent\_End\_Date,

p.Property\_No\_of\_Bedrooms AS Number\_of\_bedrooms,

COUNT(pf.Feature\_Code) AS Number\_of\_features,

ROUND((r.Price / 7) \* (Rent\_End\_Date - Rent\_Start\_Date), 2) AS Total\_Rent\_Fee,

COUNT(DISTINCT r.Rent\_ID) AS Number\_of\_Rent

FROM MRE\_Rent r, MRE\_Property p, MRE\_Property\_Feature pf

WHERE r.Property\_ID = p.Property\_ID

AND pf.Property\_ID = p.Property\_ID

AND r.Client\_Person\_ID IS NOT NULL

AND r.Rent\_Start\_Date IS NOT NULL

AND r.Rent\_End\_Date IS NOT NULL

GROUP BY r.Agent\_Person\_ID, r.Client\_Person\_ID, r.Property\_ID, r.Rent\_Start\_Date, r.Rent\_End\_Date,

p.Property\_No\_of\_Bedrooms, ROUND((r.Price / 7) \* (Rent\_End\_Date - Rent\_Start\_Date), 2)

);

ALTER TABLE MRE\_Rent\_TempFACT\_L0

ADD (Rental\_Period\_ID NUMBER,

Scale\_ID NUMBER,

Feature\_Cat\_ID NUMBER);

UPDATE MRE\_Rent\_TempFACT\_L0

SET Rental\_Period\_ID =

(CASE

WHEN MONTHS\_BETWEEN(Rent\_Start\_Date, Rent\_End\_Date) < 6 THEN 1

WHEN MONTHS\_BETWEEN(Rent\_Start\_Date, Rent\_End\_Date) BETWEEN 6 AND 12 THEN 2

WHEN MONTHS\_BETWEEN(Rent\_Start\_Date, Rent\_End\_Date) > 12 THEN 3

END),

Scale\_ID =

(CASE

WHEN Number\_of\_bedrooms <= 1 THEN 1

WHEN Number\_of\_bedrooms BETWEEN 2 AND 3 THEN 2

WHEN Number\_of\_bedrooms BETWEEN 4 AND 6 THEN 3

WHEN Number\_of\_bedrooms BETWEEN 7 AND 10 THEN 4

WHEN Number\_of\_bedrooms > 10 THEN 5

END),

Feature\_Cat\_ID =

(CASE

WHEN Number\_of\_features < 10 THEN 1

WHEN Number\_of\_features BETWEEN 10 AND 20 THEN 2

WHEN Number\_of\_features > 20 THEN 3

END)

;

CREATE TABLE MRE\_Rent\_FACT\_L0 AS (

SELECT

Agent\_Person\_ID,

Client\_Person\_ID,

Property\_ID,

to\_char(Rent\_Start\_Date, 'YYYYMMDY') as rent\_start\_date,

to\_char(Rent\_End\_Date, 'YYYYMMDY') as rent\_end\_date,

Rental\_Period\_ID,

Scale\_ID,

Feature\_Cat\_ID,

Total\_Rent\_Fee,

Number\_of\_Rent

FROM MRE\_Rent\_TempFACT\_L0

);

-- MRE\_Client\_FACT\_L0

CREATE TABLE MRE\_Client\_TempFACT\_L0 AS (

SELECT

Person\_ID AS Client\_Person\_ID,

Max\_Budget,

COUNT(Person\_ID) AS Number\_of\_Clients

FROM MRE\_Client

GROUP BY Person\_ID, Min\_Budget, Max\_Budget

);

ALTER TABLE MRE\_Client\_TempFACT\_L0

ADD Budget\_ID VARCHAR2(2);

UPDATE MRE\_Client\_TempFACT\_L0

SET Budget\_ID =

(CASE

WHEN Max\_Budget >= 0 AND Max\_Budget <= 1000 THEN 1

WHEN Max\_Budget >= 1001 AND Max\_Budget <= 100000 THEN 3

WHEN Max\_Budget >= 100001 AND Max\_Budget <= 10000000 THEN 5

END);

CREATE TABLE MRE\_Client\_FACT\_L0 AS (

SELECT

Client\_Person\_ID,

Budget\_ID,

Number\_of\_Clients

FROM MRE\_Client\_TempFACT\_L0

);

-- MRE\_Agent\_FACT\_L0

CREATE TABLE MRE\_Agent\_FACT\_L0 AS (

SELECT \* FROM

(SELECT a.person\_id as agent\_person\_id, SUM(nvl(s.price, 0)) + nvl(SUM(nvl(r.price, 0)/7\*(r.rent\_end\_date - r.rent\_start\_date)), 0) as total\_earnings

FROM mre\_agent a, mre\_sale s, mre\_rent r

WHERE a.person\_id = s.agent\_person\_id (+)

AND a.person\_id = r.agent\_person\_id (+)

GROUP BY a.person\_id)

);

-- MRE\_Visit\_FACT\_L0

CREATE TABLE MRE\_Visit\_FACT\_L0 AS (

SELECT DISTINCT

Client\_Person\_ID,

Agent\_Person\_ID,

Property\_ID,

TO\_CHAR(Visit\_Date, 'YYYYMMDY') AS Time\_ID,

COUNT(\*) AS Number\_of\_Visits

FROM MRE\_Visit

GROUP BY Client\_Person\_ID, Agent\_Person\_ID, Property\_ID, TO\_CHAR(Visit\_Date, 'YYYYMMDY')

);

-- MRE\_Advert\_FACT\_L0

CREATE TABLE MRE\_Advert\_FACT\_L0 AS (

SELECT DISTINCT

pa.Property\_ID,

pa.Advert\_ID,

TO\_CHAR(p.Property\_Date\_Added, 'YYYYMMDY') AS Time\_ID,

COUNT(pa.Advert\_ID) AS Number\_of\_Adverts

FROM MRE\_Property\_Advert pa, MRE\_Property p

WHERE pa.Property\_ID = p.Property\_ID

GROUP BY pa.Property\_ID, pa.Advert\_ID, TO\_CHAR(p.Property\_Date\_Added, 'YYYYMMDY')

);

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

-- Two-column methodology checking of fact tables --

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

-- Numbers should be wrong since tested on non-cleaned data.

-- MRE\_Sale\_FACT\_L0

SELECT SUM(Total\_Sales\_Price), SUM(Number\_of\_Sales) FROM MRE\_Sale\_FACT\_L0; -- 702,593,752 and 916

SELECT Agent\_Person\_ID, SUM(Total\_Sales\_Price), SUM(Number\_Of\_Sales) FROM MRE\_Sale\_FACT\_L0 GROUP BY Agent\_Person\_ID ORDER BY Agent\_Person\_ID; -- 702,593,752 and 916

SELECT Client\_Person\_ID, SUM(Total\_Sales\_Price), SUM(Number\_Of\_Sales) FROM MRE\_Sale\_FACT\_L0 GROUP BY Client\_Person\_ID ORDER BY Client\_Person\_ID; -- 702,593,752 and 916

SELECT Time\_ID, SUM(Total\_Sales\_Price), SUM(Number\_Of\_Sales) FROM MRE\_Sale\_FACT\_L0 GROUP BY Time\_ID ORDER BY Time\_ID; -- 702,593,752 and 916

SELECT Property\_ID, SUM(Total\_Sales\_Price), SUM(Number\_Of\_Sales) FROM MRE\_Sale\_FACT\_L0 GROUP BY Property\_ID ORDER BY Property\_ID; -- 702,593,752 and 916

SELECT Property\_Type, SUM(Total\_Sales\_Price), SUM(Number\_Of\_Sales) FROM MRE\_Sale\_FACT\_L0 sf, MRE\_Property\_Dim\_L0 p WHERE sf.property\_id = p.property\_id GROUP BY Property\_Type ORDER BY Property\_Type; -- 702,593,752 and 916

-- MRE\_Rent\_FACT\_L0

SELECT SUM(Number\_of\_Rent) FROM MRE\_Rent\_FACT\_L0; -- 1116

SELECT Agent\_Person\_ID, SUM(Number\_Of\_Rent) FROM MRE\_Rent\_FACT\_L0 GROUP BY Agent\_Person\_ID ORDER BY Agent\_Person\_ID; -- 1116

SELECT Client\_Person\_ID, SUM(Number\_Of\_Rent) FROM MRE\_Rent\_FACT\_L0 GROUP BY Client\_Person\_ID ORDER BY Client\_Person\_ID; -- 1116

SELECT Property\_ID, SUM(Number\_Of\_Rent) FROM MRE\_Rent\_FACT\_L0 GROUP BY Property\_ID ORDER BY Property\_ID; -- 1116

SELECT Rent\_Start\_Date, SUM(Number\_Of\_Rent) FROM MRE\_Rent\_FACT\_L0 GROUP BY Rent\_Start\_Date ORDER BY Rent\_Start\_Date; -- 1116

SELECT Rent\_End\_Date, SUM(Number\_Of\_Rent) FROM MRE\_Rent\_FACT\_L0 GROUP BY Rent\_End\_Date ORDER BY Rent\_End\_Date; -- 1116

SELECT Scale\_ID, SUM(Number\_Of\_Rent) FROM MRE\_Rent\_FACT\_L0 GROUP BY Scale\_ID ORDER BY Scale\_ID; -- 1116

SELECT Feature\_Cat\_ID, SUM(Number\_Of\_Rent) FROM MRE\_Rent\_FACT\_L0 GROUP BY Feature\_Cat\_ID ORDER BY Feature\_Cat\_ID; -- 1116

-- MRE\_Client\_FACT\_L0

SELECT SUM(Number\_Of\_Clients) FROM MRE\_Client\_FACT\_L0; -- 3339

SELECT Client\_Person\_ID, SUM(Number\_Of\_Clients) FROM MRE\_Client\_FACT\_L0 GROUP BY Client\_Person\_ID ORDER BY Client\_Person\_ID; -- 3339

SELECT Budget\_ID, SUM(Number\_Of\_Clients) FROM MRE\_Client\_FACT\_L0 GROUP BY Budget\_ID ORDER BY Budget\_ID; -- 3339

-- MRE\_Agent\_FACT\_L0

SELECT SUM(Total\_Earnings) FROM MRE\_Agent\_FACT\_L0; -- 477,290,000

SELECT Agent\_Person\_ID, SUM(Total\_Earnings) FROM MRE\_Agent\_FACT\_L0 GROUP BY Agent\_Person\_ID ORDER BY Agent\_Person\_ID; -- 477,290,000

-- MRE\_Visit\_FACT\_L0

SELECT SUM(Number\_of\_Visits) FROM MRE\_Visit\_FACT\_L0; -- 575

SELECT Client\_Person\_ID, SUM(Number\_of\_Visits) FROM MRE\_Visit\_FACT\_L0 GROUP BY Client\_Person\_ID ORDER BY Client\_Person\_ID; -- 575

SELECT Agent\_Person\_ID, SUM(Number\_of\_Visits) FROM MRE\_Visit\_FACT\_L0 GROUP BY Agent\_Person\_ID ORDER BY Agent\_Person\_ID; -- 575

SELECT Property\_ID, SUM(Number\_of\_Visits) FROM MRE\_Visit\_FACT\_L0 GROUP BY Property\_ID ORDER BY Property\_ID; -- 575

SELECT Time\_ID, SUM(Number\_of\_Visits) FROM MRE\_Visit\_FACT\_L0 GROUP BY Time\_ID ORDER BY Time\_ID; -- 575

-- MRE\_Advert\_FACT\_L0

SELECT SUM(Number\_of\_Adverts) FROM MRE\_Advert\_FACT\_L0; -- 3646

SELECT Property\_ID, SUM(Number\_of\_Adverts) FROM MRE\_Advert\_FACT\_L0 GROUP BY Property\_ID ORDER BY Property\_ID; -- 3646

SELECT Advert\_ID, SUM(Number\_of\_Adverts) FROM MRE\_Advert\_FACT\_L0 GROUP BY Advert\_ID ORDER BY Advert\_ID; -- 3646

SELECT Time\_ID, SUM(Number\_of\_Adverts) FROM MRE\_Advert\_FACT\_L0 GROUP BY Time\_ID ORDER BY Time\_ID; -- 3646

COMMIT;

## Output c – Screenshots of Tables

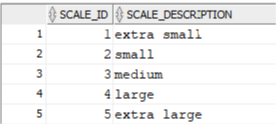
Performed using:

SELECT \*

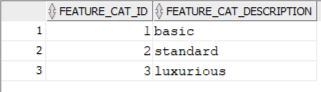
FROM [table\_name];

### c1 Level 2 Star Schema

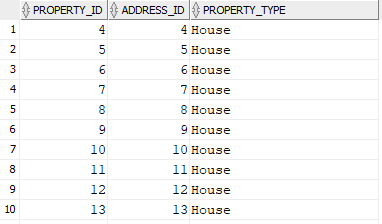
MRE\_Scale\_DIM\_L2



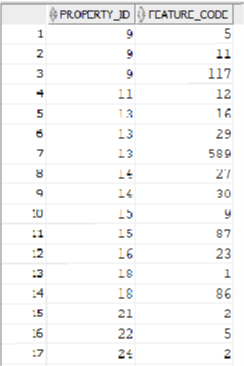
MRE\_Feature\_Cat\_DIM\_L2



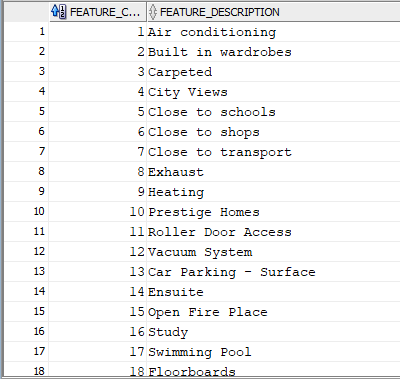
MRE\_Property\_DIM\_L2



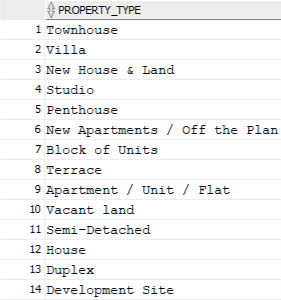
MRE\_Property\_Feature\_Bridge\_L2



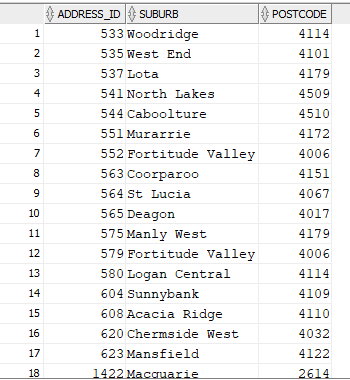
MRE\_Feature\_DIM\_L2



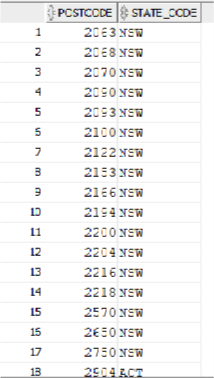
MRE\_Property\_Type\_DIM\_L2



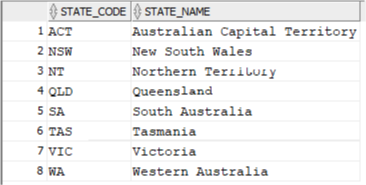
MRE\_Address\_DIM\_L2



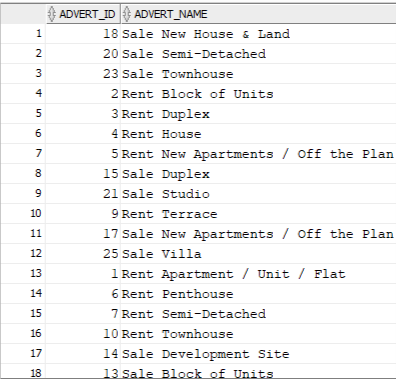
MRE\_Postcode\_DIM\_L2



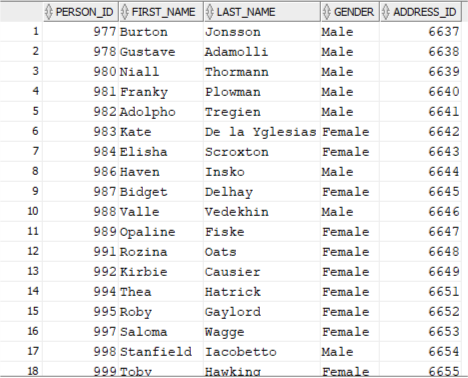
MRE\_State\_DIM\_L2



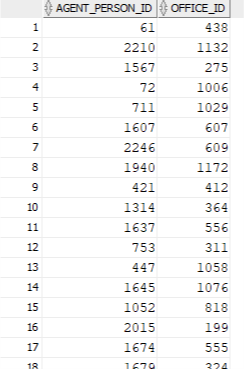
MRE\_Advertisement\_DIM\_L2



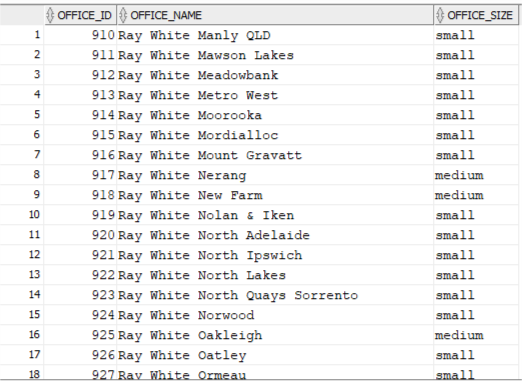
MRE\_Person\_DIM\_L2



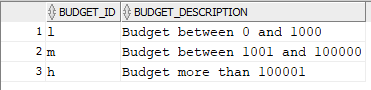
MRE\_Agent\_Office\_DIM\_L2



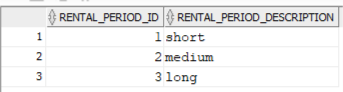
MRE\_Office\_DIM\_L2



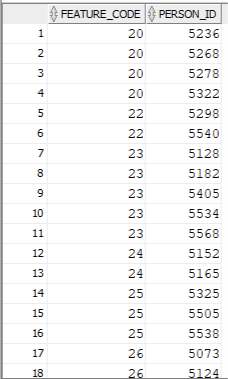
MRE\_Budget\_DIM\_L2



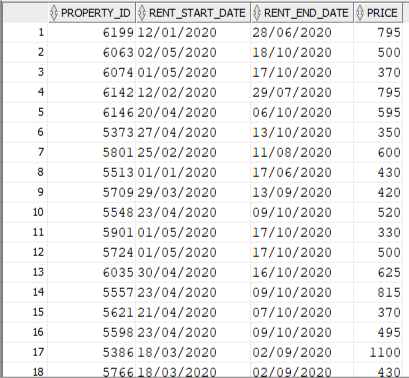
MRE\_Rental\_Period\_DIM\_L2



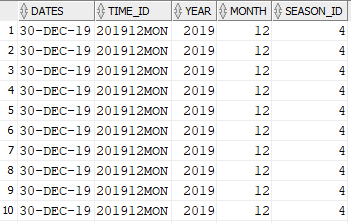
MRE\_Wishlist\_DIM\_L2



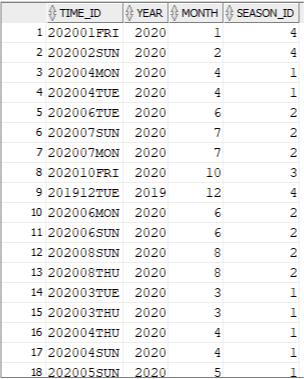
MRE\_Rent\_Price\_DIM\_L2



MRE\_Temp\_Time\_DIM\_L2



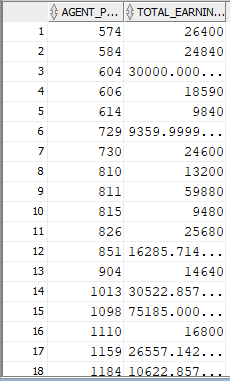
MRE\_Time\_DIM\_L2



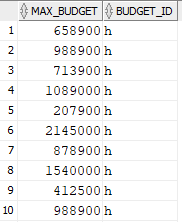
MRE\_Season\_DIM\_L2



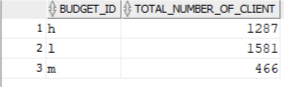
MRE\_Agent\_FACT\_L2



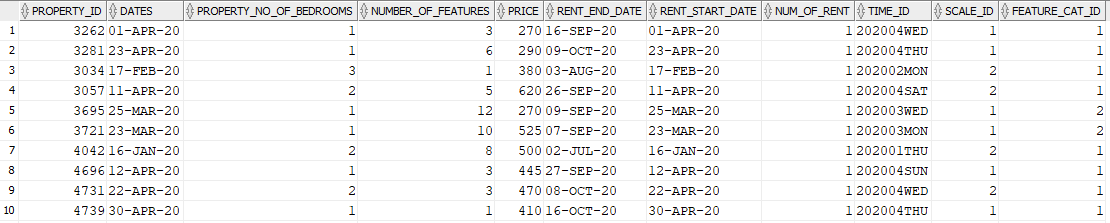
MRE\_Temp\_Client\_L2



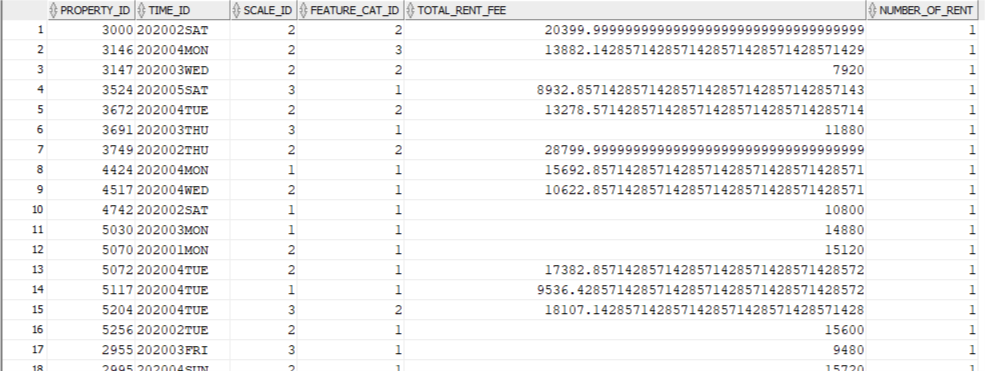
MRE\_Client\_FACT\_L2



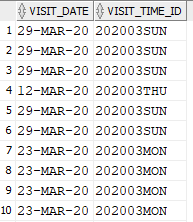
MRE\_Temp\_Rent\_FACT\_L2



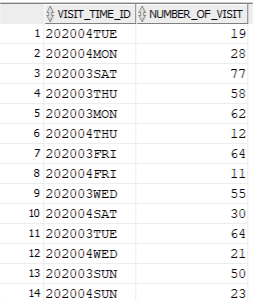
MRE\_Rent\_FACT\_L2



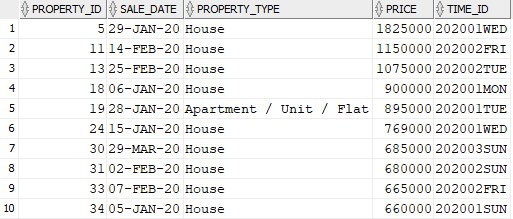
MRE\_Temp\_Visit\_L2



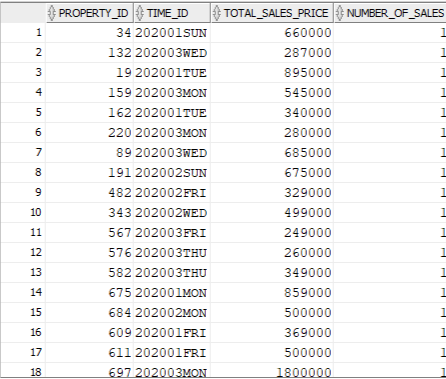
MRE\_Visit\_FACT\_L2



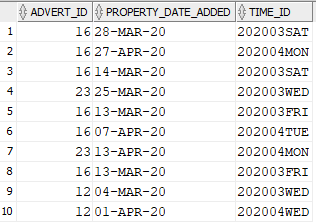
MRE\_Temp\_Sale\_FACT\_L2



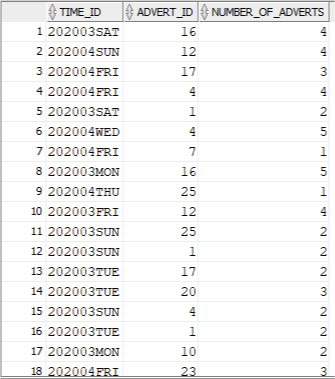
MRE\_Sale\_FACT\_L2



MRE\_Temp\_Advert\_L2

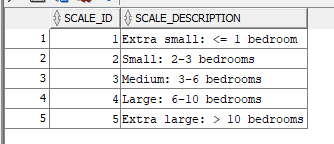


MRE\_Advert\_FACT\_L2

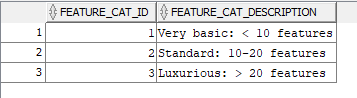


### c2 Level 0 Star Schema

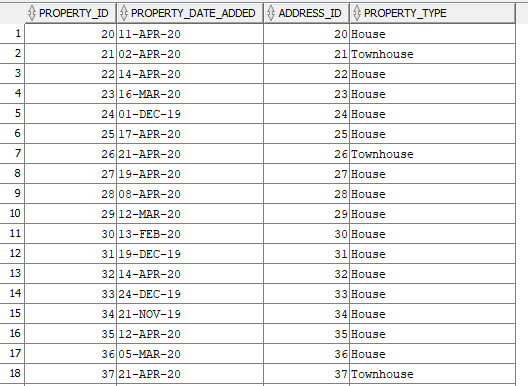
MRE\_Scale\_DIM\_L0



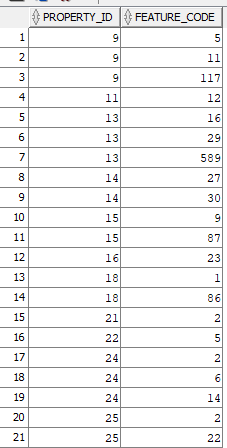
MRE\_Feature\_Cat\_DIM\_L0



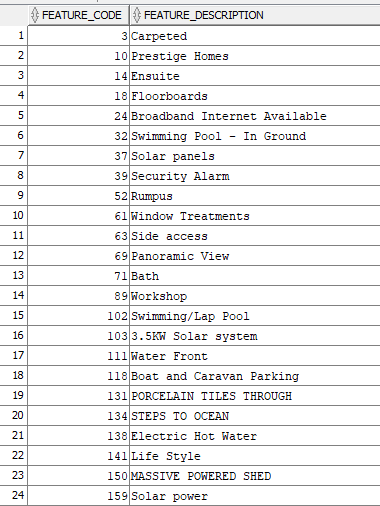
MRE\_Property\_DIM\_L0



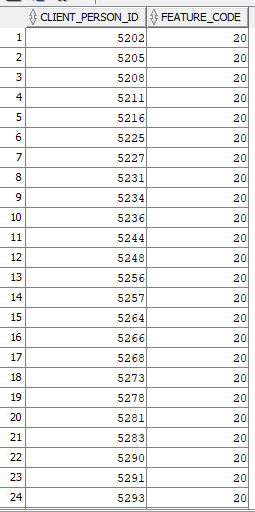
MRE\_Property\_Feature\_Bridge\_L0



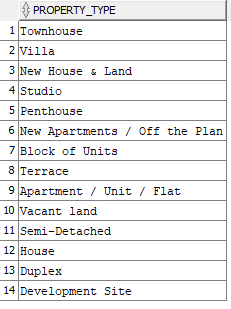
MRE\_Feature\_DIM\_L0



MRE\_Wishlist\_DIM\_L0



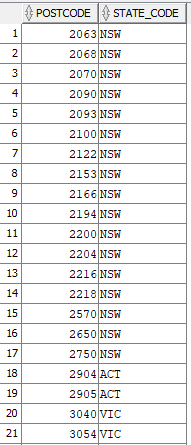
MRE\_Property\_Type\_DIM\_L0



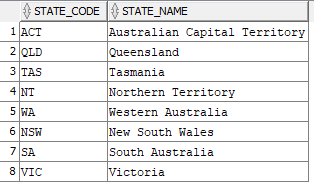
MRE\_Address\_DIM\_L0



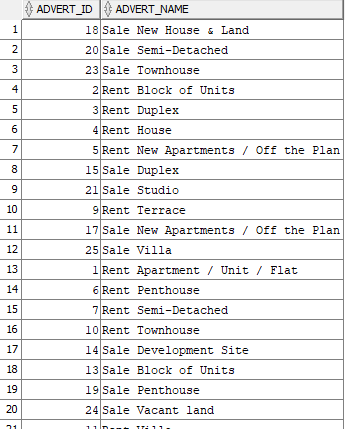
MRE\_Postcode\_DIM\_L0



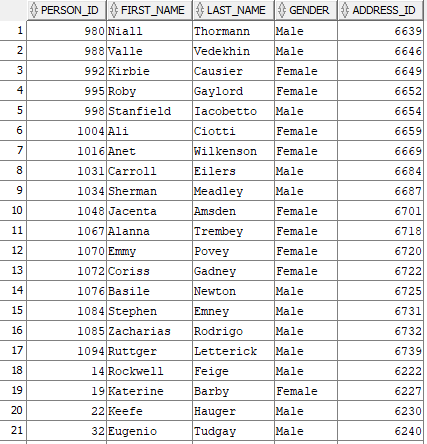
MRE\_State\_DIM\_L0



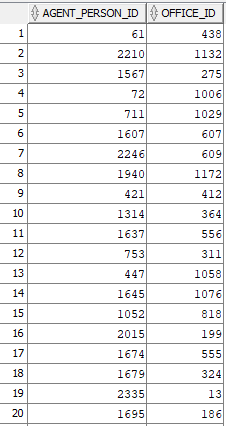
MRE\_Advertisement\_DIM\_L0



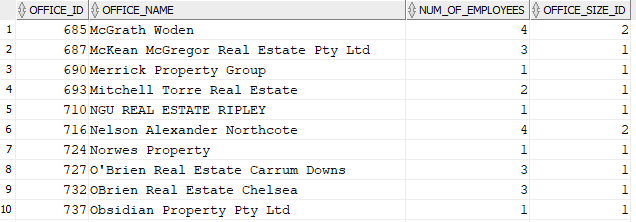
MRE\_Person\_DIM\_L0



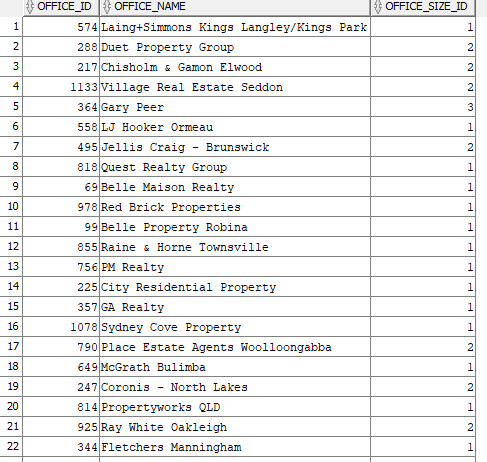
MRE\_Agent\_Office\_DIM\_L0



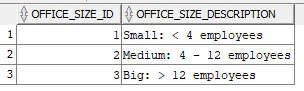
MRE\_Office\_TempDIM\_L0



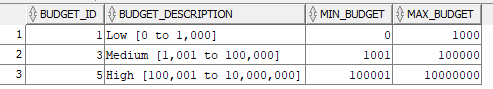
MRE\_Office\_DIM\_L0



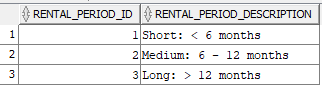
MRE\_Office\_Size\_DIM\_L0



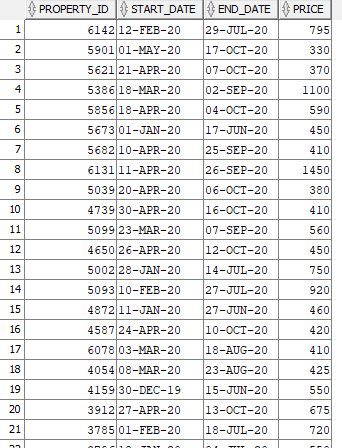
MRE\_Budget\_DIM\_L0



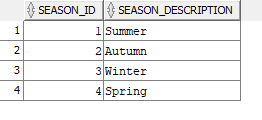
MRE\_Rental\_Period\_DIM\_L0



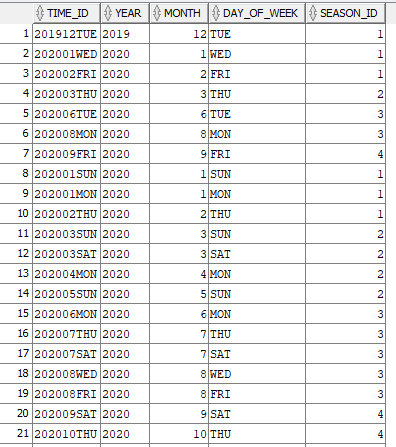
MRE\_Rent\_Price\_DIM\_L0



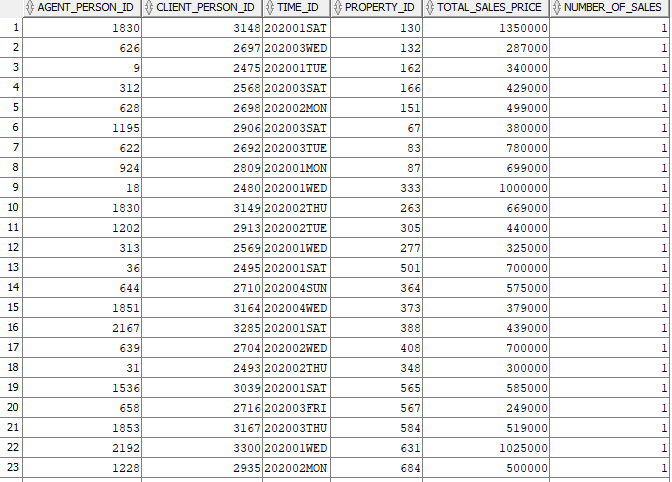
MRE\_Season\_DIM\_L0



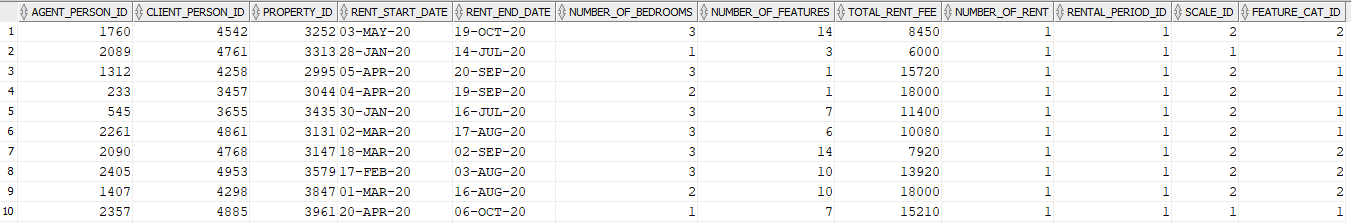
MRE\_Time\_DIM\_L0



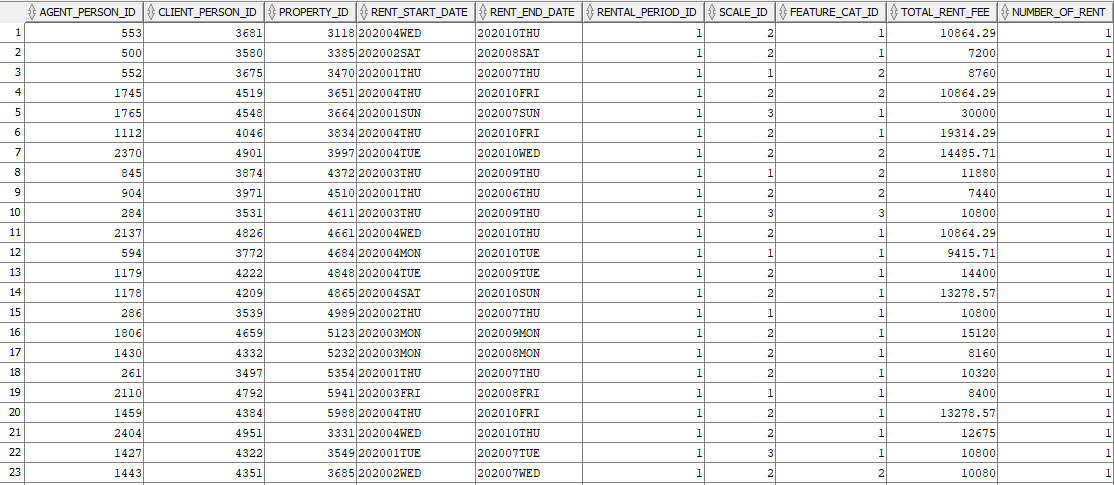
MRE\_Sale\_FACT\_L0



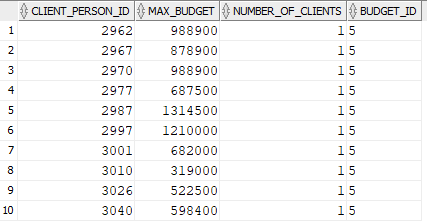
MRE\_Rent\_TempFACT\_L0



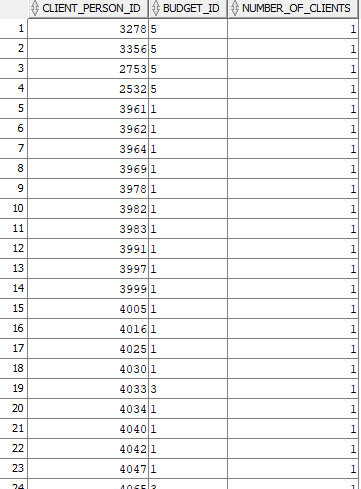
MRE\_Rent\_FACT\_L0



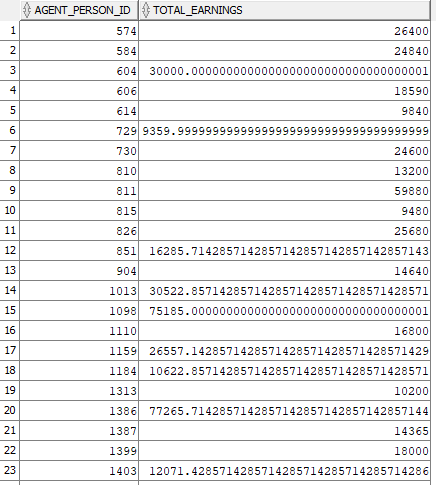
MRE\_Client\_TempFACT\_L0



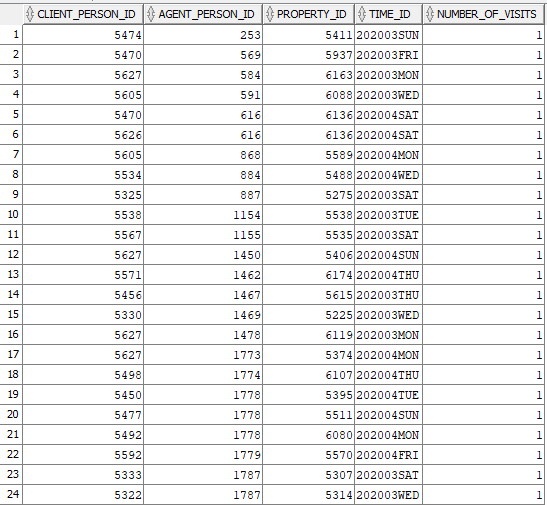
MRE\_Client\_FACT\_L0



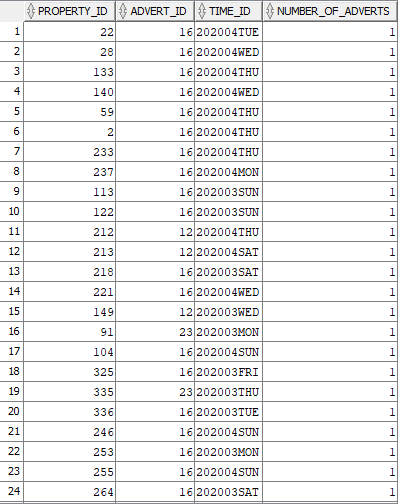
MRE\_Agent\_FACT\_L0



MRE\_Visit\_FACT\_L0



MRE\_Advert\_FACT\_L0



# Task C.3

## Simple Reports

**Report 1**

1. **The query questions written in English**

Top 15 most rented property by scale and suburb.

1. **Your explanation on why such a query is necessary or useful for the management**

This will allow the management to get the sentiment of the rental market.

1. **The SQL commands**

SELECT \*

FROM

(SELECT s.scale\_description as Scale,

a.suburb as Suburb,

SUM(f.number\_of\_rent) as Number\_of\_Rents,

ROW\_NUMBER() OVER(ORDER BY SUM(f.number\_of\_rent)

DESC) as RANK

FROM mre\_rent\_fact\_l2 f, mre\_scale\_dim\_l2 s, mre\_property\_dim\_l2 p, mre\_address\_dim\_l2 a

WHERE f.scale\_id = s.scale\_id

AND f.property\_id = p.property\_id

AND p.address\_id = a.address\_id

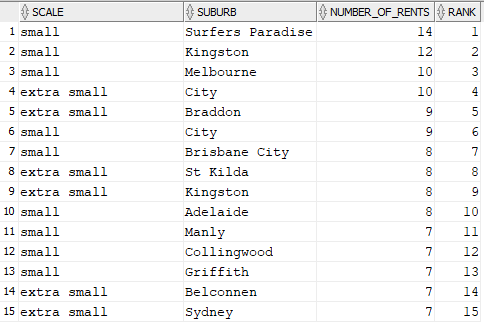
GROUP BY s.scale\_description, a.suburb

ORDER BY ROW\_NUMBER() OVER(ORDER BY

SUM(f.number\_of\_rent) DESC) ASC)

WHERE RANK <= 15;

1. **The screenshots of the query results (or part of the query results), including all the attribute names**



**Report 2**

1. **The query questions written in English**

Top 15% sales based on time period and property type.

1. **Your explanation on why such a query is necessary or useful for the management**

This might give the management an idea to focus on which suburb at what time to boast business performance.

1. **The SQL commands**

SELECT \*

FROM (

SELECT t.year as Year,

t.month as Month,

p.property\_type as Property\_Type,

SUM(f.total\_sales\_price) as Total\_Sales\_Price,

SUM(f.number\_of\_sales) as Number\_of\_Sales,

PERCENT\_RANK() OVER (ORDER BY SUM(f.total\_sales\_price) DESC) as Revenue\_Ranking

FROM mre\_sale\_fact\_l2 f, mre\_property\_dim\_l2 p, mre\_time\_dim\_l2 t

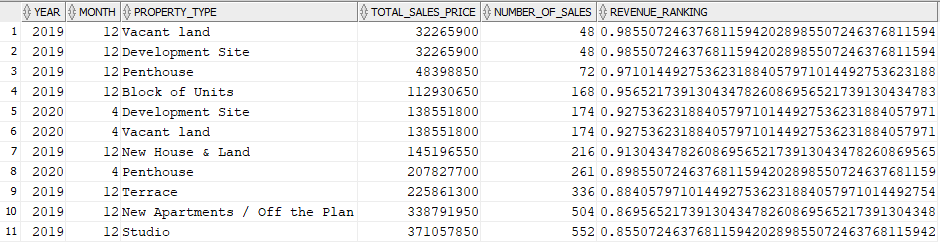
WHERE f.time\_id = t.time\_id

GROUP BY t.year, t.month, p.property\_type)

WHERE Revenue\_Ranking >= 0.85

ORDER BY Revenue\_Ranking DESC;

1. **The screenshots of the query results (or part of the query results), including all the attribute names**



**Report 3**

1. **The query questions written in English**

Total property visited by suburb and season.

1. **Your explanation on why such a query is necessary or useful for the management**

This will give the management and insight of property investment of different locations and seasons

1. **The SQL commands**

SELECT t.year as Year,

s.season\_description as season,

a.suburb as suburb,

SUM(number\_of\_visits) as Number\_of\_Visits

FROM mre\_visit\_fact\_l0 f, mre\_time\_dim\_l0 t,

mre\_season\_dim\_l0 s, mre\_property\_dim\_l0 p, mre\_address\_dim\_l0 a

WHERE f.time\_id = t.time\_id

AND t.season\_id = s.season\_id

AND f.property\_id = p.property\_id

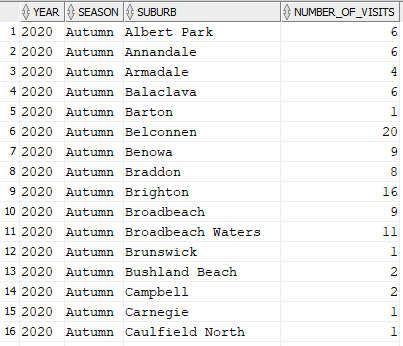
AND p.address\_id = a.address\_id

GROUP BY t.year, s.season\_description, a.suburb

ORDER BY t.year, s.season\_description,

a.suburb;

1. **The screenshots of the query results (or part of the query results), including all the attribute names**



## Reports with proper sub-totals

**Report 4**

1. **The query questions written in English**

What are the sub-total and total rental fees from each suburb, time period, and property type?

1. **Your explanation on why such a query is necessary or useful for the management**

This will give the management an idea of different combinations between suburb, time period and property type.

1. **The SQL commands**

SELECT t.year||t.month as Time\_Period,

a.suburb as Suburb,

p.property\_type as Property\_Type,

to\_char(SUM(f.total\_rent\_fee), '9,999,999,999.99')

as Rental\_Fees,

DECODE(GROUPING(t.year||t.month), 1, 'All Periods',

t.year||t.month) as Period,

DECODE(GROUPING(a.suburb), 1, 'All Suburbs',

a.suburb) as Suburbs,

DECODE(GROUPING(p.property\_type), 1, 'All Types',

p.property\_type) as Types

FROM mre\_rent\_fact\_l2 f, mre\_time\_dim\_l2 t, mre\_property\_dim\_l2 p, mre\_address\_dim\_l2 a

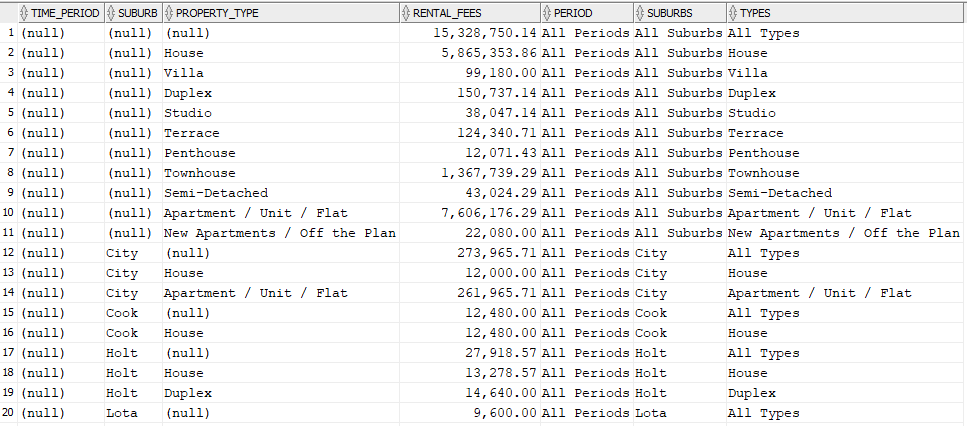
WHERE f.time\_id = t.time\_id

AND f.property\_id = p.property\_id

AND p.address\_id = a.address\_id

GROUP BY CUBE(t.year||t.month, a.suburb, p.property\_type);

1. **The screenshots of the query results (or part of the query results), including all the attribute names**



**Report 5**

1. **The query questions written in English**

What are the sub-total and total rental fees from each suburb, time period, and property type?

1. **Your explanation on why such a query is necessary or useful for the management**

This will give the management an idea of different combinations between suburb, time period and property type.

1. **The SQL commands**

SELECT t.year||t.month as Time\_Period,

a.suburb as Suburb,

p.property\_type as Property\_Type,

to\_char(SUM(f.total\_rent\_fee), '9,999,999,999.99')

as Rental\_Fees,

DECODE(GROUPING(t.year||t.month), 1, 'All Periods',

t.year||t.month) as Period,

DECODE(GROUPING(a.suburb), 1, 'All Suburbs',

a.suburb) as Suburbs,

DECODE(GROUPING(p.property\_type), 1, 'All Types',

p.property\_type) as Types

FROM mre\_rent\_fact\_l2 f, mre\_time\_dim\_l2 t, mre\_property\_dim\_l2 p, mre\_address\_dim\_l2 a

WHERE f.time\_id = t.time\_id

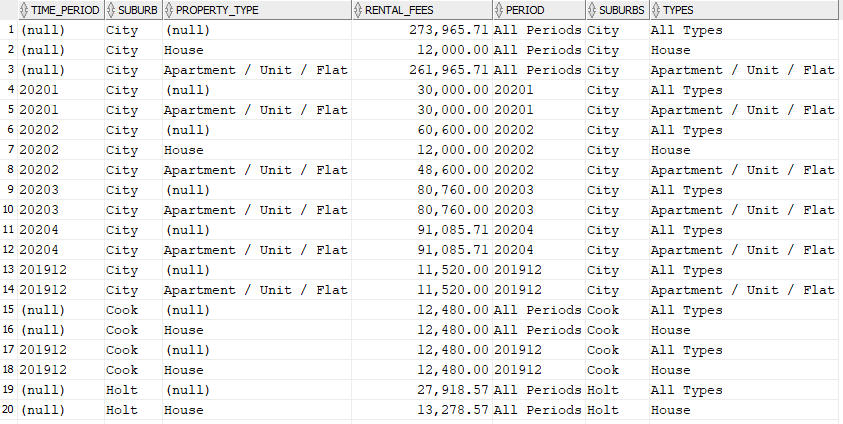
AND f.property\_id = p.property\_id

AND p.address\_id = a.address\_id

GROUP BY a.suburb, CUBE(t.year||t.month,

p.property\_type);

1. **The screenshots of the query results (or part of the query results), including all the attribute names**



**Report 6**

1. **The query questions written in English**

What is the sub-total and total sale revenue from each state and time period for houses?

1. **Your explanation on why such a query is necessary or useful for the management**

This will give the management an understanding of sale of one of their best sellers, houses, at different states and time periods.

1. **The SQL commands**

SELECT t.year||t.month as Time\_period,

st.state\_name as State,

SUM(s.total\_sales\_price) as Total\_Revenue,

DECODE(GROUPING(t.year||t.month), 1, 'All Periods',

t.year||t.month) as Periods,

DECODE(GROUPING(st.state\_name), 1, 'All States',

st.state\_name) as States

FROM mre\_sale\_fact\_l2 s, mre\_property\_dim\_l2 p,

mre\_address\_dim\_l2 a, mre\_postcode\_dim\_l2 pc,

mre\_state\_dim\_l2 st, mre\_time\_dim\_l2 t

WHERE s.property\_id = p.property\_id

AND p.address\_id = a.address\_id

AND a.postcode = pc.postcode

AND pc.state\_code = st.state\_code

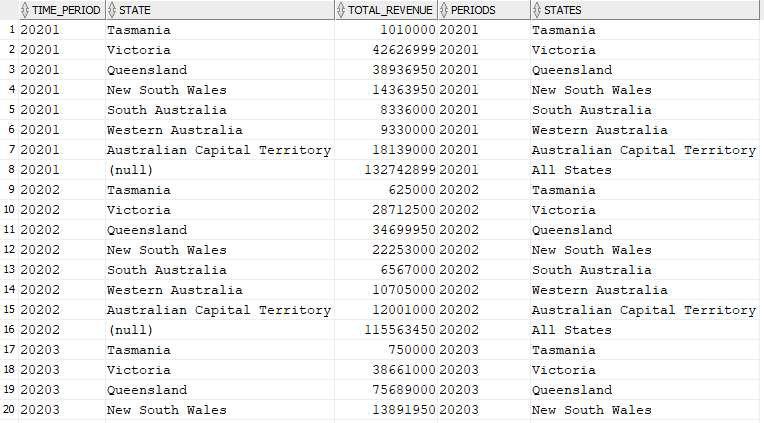
AND s.time\_id = t.time\_id

AND p.property\_type = 'House'

GROUP BY ROLLUP (t.year||t.month,

st.state\_name);

1. **The screenshots of the query results (or part of the query results), including all the attribute names**



**Report 7**

1. **The query questions written in English**

What is the sub-total and total sale revenue from each state and time period for houses?

1. **Your explanation on why such a query is necessary or useful for the management**

This will give the management an understanding of sale of one of their best sellers, houses, at different states and time periods.

1. **The SQL commands**

SELECT t.year||t.month as Time\_period,

st.state\_name as State,

SUM(s.total\_sales\_price) as Total\_Revenue,

DECODE(GROUPING(t.year||t.month), 1, 'All Periods',

t.year||t.month) as Periods,

DECODE(GROUPING(st.state\_name), 1, 'All States',

st.state\_name) as States

FROM mre\_sale\_fact\_l2 s, mre\_property\_dim\_l2 p, mre\_address\_dim\_l2 a, mre\_postcode\_dim\_l2 pc, mre\_state\_dim\_l2 st, mre\_time\_dim\_l2 t

WHERE s.property\_id = p.property\_id

AND p.address\_id = a.address\_id

AND a.postcode = pc.postcode

AND pc.state\_code = st.state\_code

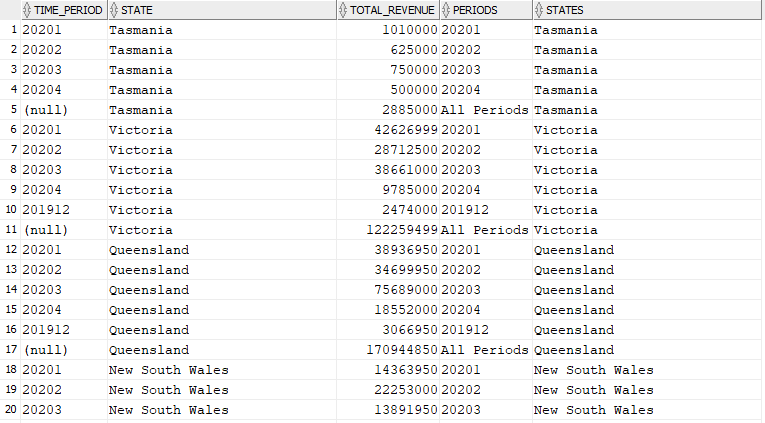
AND s.time\_id = t.time\_id

AND p.property\_type = 'House'

GROUP BY st.state\_name, ROLLUP

(t.year||t.month);

1. **The screenshots of the query results (or part of the query results), including all the attribute names**



## Reports with moving and cumulative aggregates

**Report 8**

1. **The query questions written in English**

What is the total number of clients and cumulative number of clients with a high budget in each year?

1. **Your explanation on why such a query is necessary or useful for the management**

This will give the management an understanding of how high budget clients have chosen MonRE as their preferred agency.

1. **The SQL commands**

SELECT year,

SUM(total\_clients) as Number\_of\_Clients,

SUM(SUM(total\_clients)) OVER (ORDER BY year ROWS

UNBOUNDED PRECEDING) as Cumulative\_Total

FROM

(SELECT \*

FROM

(SELECT t.year, SUM(f.number\_of\_clients) as total\_clients

FROM mre\_client\_fact\_l0 f, mre\_budget\_dim\_l0 b, mre\_rent\_fact\_l0 rf, mre\_time\_dim\_l0 t

WHERE f.budget\_id = b.budget\_id

AND f.client\_person\_id = rf.client\_person\_id

AND rf.rent\_start\_date = t.time\_id

AND b.budget\_description LIKE 'High%'

GROUP BY t.year)

UNION

(SELECT t.year, SUM(f.number\_of\_clients) as total\_clients

FROM mre\_client\_fact\_l0 f, mre\_budget\_dim\_l0 b, mre\_sale\_fact\_l0 sf, mre\_time\_dim\_l0 t

WHERE f.budget\_id = b.budget\_id

AND f.client\_person\_id = sf.client\_person\_id

AND sf.time\_id = t.time\_id

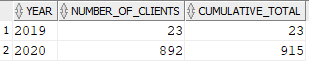
AND b.budget\_description LIKE 'High%'

GROUP BY t.year))

GROUP BY year

ORDER BY year;

1. **The screenshots of the query results (or part of the query results), including all the attribute names**



**Report 9**

1. **The query questions written in English**

What is the total monthly number of visits and 3-month average number of visits?

1. **Your explanation on why such a query is necessary or useful for the management**

Management may want to know more about why are there more visits in certain months. They can then drill down to find out which properties are having many visits from different visiting clients.

1. **The SQL commands**

SELECT t.year,

t.month,

SUM(number\_of\_visit) as Number\_of\_Visits,

to\_char(AVG(SUM(f.number\_of\_visit)) OVER (ORDER BY

t.year, t.month ROWS 2 PRECEDING), '999,999') as

Average\_3\_Month\_Visits

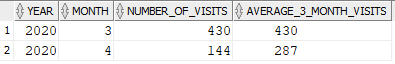
FROM mre\_visit\_fact\_l2 f, mre\_time\_dim\_l2 t

WHERE f.visit\_time\_id =t.time\_id

GROUP BY t.year, t.month

ORDER BY year, month +0;

1. **The screenshots of the query results (or part of the query results), including all the attribute names**



**Report 10**

1. **The query questions written in English**

Cumulative monthly total number of rents based on properties that have been rented to clients.

1. **Your explanation on why such a query is necessary or useful for the management**

Management would have a good indication of cash flow coming from rentals. They can use this information to make decisions on whether to increase spending on attracting tenants.

1. **The SQL commands**

SELECT t.year as Year,

t.month as Month,

SUM(Number\_of\_Rent) as Number\_of\_Rents,

SUM(SUM(number\_of\_rent)) OVER (ORDER BY t.year,

t.month ROWS UNBOUNDED PRECEDING) as

Cumulative\_Number\_of\_Rent

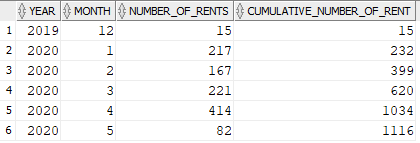
FROM mre\_rent\_fact\_l2 f, mre\_time\_dim\_l2 t

WHERE f.time\_id = t.time\_id

GROUP BY t.year, t.month

ORDER BY year, month +0;

1. **The screenshots of the query results (or part of the query results), including all the attribute names**



## Reports with Partitions

**Report 11**

1. **The query questions written in English**

Show ranking of each property type based on the yearly total number of sales and the ranking of each state based on the yearly total number of sales.

1. **Your explanation on why such a query is necessary or useful for the management**

This information might to useful to management to see the property type with most sales as well as the state with most sales of each type.

1. **The SQL commands**

SELECT t.year as Year,

p.property\_type as Property\_Type,

s.state\_name as State,

SUM(f.number\_of\_sales) as Total\_Number\_of\_Sales,

RANK() OVER (PARTITION BY t.year ORDER BY

SUM(f.number\_of\_sales) DESC) as RANK\_BY\_YEAR,

RANK() OVER (PARTITION BY s.state\_name ORDER BY

SUM(f.number\_of\_sales) DESC) as RANK\_BY\_STATE

FROM mre\_sale\_fact\_l2 f, mre\_property\_dim\_l2 p,

mre\_time\_dim\_l2 t, mre\_address\_dim\_l2 a, mre\_postcode\_dim\_l2 pc, mre\_state\_dim\_l2 s

WHERE f.property\_id = p.property\_id

AND f.time\_id = t.time\_id

AND p.address\_id = a.address\_id

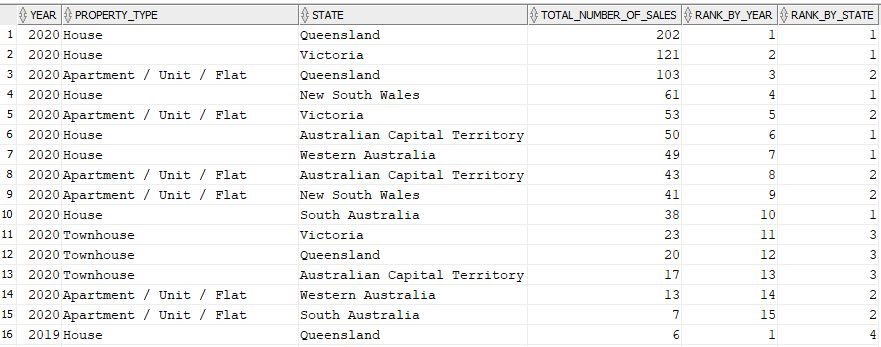
AND a.postcode = pc.postcode

AND pc.state\_code = s.state\_code

GROUP BY t.year, p.property\_type, s.state\_name

ORDER BY SUM(f.number\_of\_sales) DESC;

1. **The screenshots of the query results (or part of the query results), including all the attribute names**



**Report 12**

1. **The query questions written in English**

Show ranking of each advertisement type based on the yearly total number of adverts and the ranking of each state based on the yearly total number of adverts.

1. **Your explanation on why such a query is necessary or useful for the management**

This information might be useful to management to compare the number of advertisements of different types and different states to observe sale patterns of certain areas.

1. **The SQL commands**

SELECT t.year as Year,

ad.advert\_name as Advertisement\_Type,

s.state\_name as State,

SUM(f.number\_of\_adverts) as

Yearly\_Total\_Number\_of\_Adverts,

RANK() OVER (PARTITION BY ad.advert\_name ORDER BY

SUM(f.number\_of\_adverts) DESC) as

RANK\_BY\_ADVERT\_TYPE,

RANK() OVER (PARTITION BY s.state\_name ORDER BY

SUM(f.number\_of\_adverts) DESC) as RANK\_BY\_STATE

FROM mre\_advert\_fact\_l0 f, mre\_advertisement\_dim\_l0 ad, mre\_property\_dim\_l0 p, mre\_address\_dim\_l0 a, mre\_postcode\_dim\_l0 pc, mre\_state\_dim\_l0 s, mre\_time\_dim\_l0 t

WHERE f.advert\_id = ad.advert\_id

AND f.property\_id = p.property\_id

AND p.address\_id = a.address\_id

AND a.postcode = pc.postcode

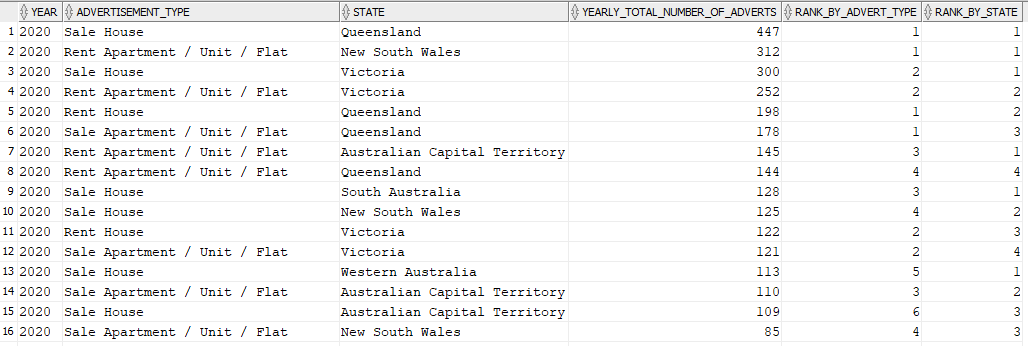
AND pc.state\_code = s.state\_code

AND f.time\_id = t.time\_id

GROUP BY t.year, ad.advert\_name, s.state\_name

ORDER BY SUM(f.number\_of\_adverts) DESC;

1. **The screenshots of the query results (or part of the query results), including all the attribute names**



# Task 4

**Report 2**

Top 15% sales based on time period and property type.

A screenshot of a cell phone

Description automatically generated

**Report 3**

Total property visited by suburb and season.

A screenshot of a social media post

Description automatically generated

**Report 8**

What is the total number of clients and cumulative number of clients with a high budget in each year?

A screenshot of a social media post

Description automatically generated

**Report 9**

What is the total monthly number of visits and 3-month average number of visits?

A screenshot of a cell phone

Description automatically generated

**Report 10**

Cumulative monthly total number of rents based on properties that have been rented to clients

A screenshot of a cell phone

Description automatically generated