FIT5195 MAJOR ASSIGNMENT

MonRE Data Warehouse Implementation and Dashboard

Submitted by:

Abhilash Payghan (30914241)

Nikita Mandlik (29589746)

Monash University
BIDW

MONASH University
Information Technology

GROUP ASSIGNMENT COVER SHEET

Student ID Number	Surname	Given Names
30914241	Payghan	**Abhilash**
29589746	Mandlik	**Nikita**

* Please include the names of all other group members.

Unit name and code	FIT5195: Business intelligence and data warehousing	
Title of assignment	FIT5195 Major Assignment2020-Semester1	
Lecturer/tutor	Farah Kabir	
Tutorial day and time	Wednesday- (10am to 12pm)	**Campus Caulfield**
Is this an authorised group assignment? ☒ **Yes** ☐ **No**		
Has any part of this assignment been previously submitted as part of another unit/course? ☐ **Yes** ☒ **No**		
Due Date 16/06/2020		***Date submitted 16/06/2020***

All work must be submitted by the due date. If an extension of work is granted this must be specified with the signature of the lecturer/tutor.

Extension granted until (date) ..... Signature of lecturer/tutor .....

Please note that it is your responsibility to retain copies of your assessments.

Intentional plagiarism or collusion amounts to cheating under Part 7 of the Monash University (Council) Regulations

Plagiarism: Plagiarism means taking and using another person's ideas or manner of expressing them and passing them off as one's own. For example, by failing to give appropriate acknowledgement. The material used can be from any source (staff, students or the internet, published and unpublished works).

Collusion: Collusion means unauthorised collaboration with another person on assessable written, oral or practical work and includes paying another person to complete all or part of the work.

Where there are reasonable grounds for believing that intentional plagiarism or collusion has occurred, this will be reported to the Associate Dean (Education) or delegate, who may disallow the work concerned by prohibiting assessment or refer the matter to the Faculty Discipline Panel for a hearing.

Student Statement:

- I have read the university's Student Academic Integrity Policy and Procedures.
- I understand the consequences of engaging in plagiarism and collusion as described in Part 7 of the Monash University (Council) Regulations http://adm.monash.edu/legal/legislation/statutes
- I have taken proper care to safeguard this work and made all reasonable efforts to ensure it could not be copied.
- No part of this assignment has been previously submitted as part of another unit/course.

MONASH University
Information Technology

- I acknowledge and agree that the assessor of this assignment may for the purposes of assessment, reproduce the assignment and:
  i. provide to another member of faculty and any external marker; and/or
  ii. submit it to a text matching software; and/or
  iii. submit it to a text matching software which may then retain a copy of the assignment on its database for the purpose of future plagiarism checking.
- I certify that I have not plagiarised the work of others or participated in unauthorised collaboration when preparing this assignment.

Signature ..... *Date*.....

* delete (iii) if not applicable

Signature Abhilash Payghan Date: 16/06/2020

Signature Nikita Mandlik Date: 16/06/2020

Privacy Statement

The information on this form is collected for the primary purpose of assessing your assignment and ensuring the academic integrity requirements of the University are met. Other purposes of collection include recording your plagiarism and collusion declaration, attending to course and administrative matters and statistical analyses. If you choose not to complete all the questions on this form it may not be possible for Monash University to assess your assignment. You have a right to access personal information that Monash University holds about you, subject to any exceptions in relevant legislation. If you wish to seek access to your personal information or inquire about the handling of your personal information, please contact the University Privacy Officer: privacyofficer@adm.monash.edu.au

MONASH University
Information Technology

[redacted]

Please fill in the form with the contribution from each student towards the assignment.

[redacted]

Student ID	Student Name	Contribution Percentage	Tasks Dones
30914241	Abhilash Payghan	50%	-Task c.1 c), d), e), f) -Task c.2.b) -Task c.3) -Task c.4
29589746	Nikita Mandlik	50%	-Task C.1 a), b), c) -Task C.2.a),C) -Task c.3 -Task c.4

[redacted]

We declare that:

- The information we have supplied in or with this form is complete and correct.
- We understand that the information we have provided in this form will be used for individual assessment of the assignment.

[redacted]

Signatures	*Abhilash Payghan*	*Nikita Mandlik*
date	16/06/2020	

Details of Oracle Account:

-For Abhilash Payghan

User Id: S30914241

Password: student

-For Nikita Mandlik

Used Id: S29589746

Password: student

Task C.1

a) The E/R diagram of the operational database

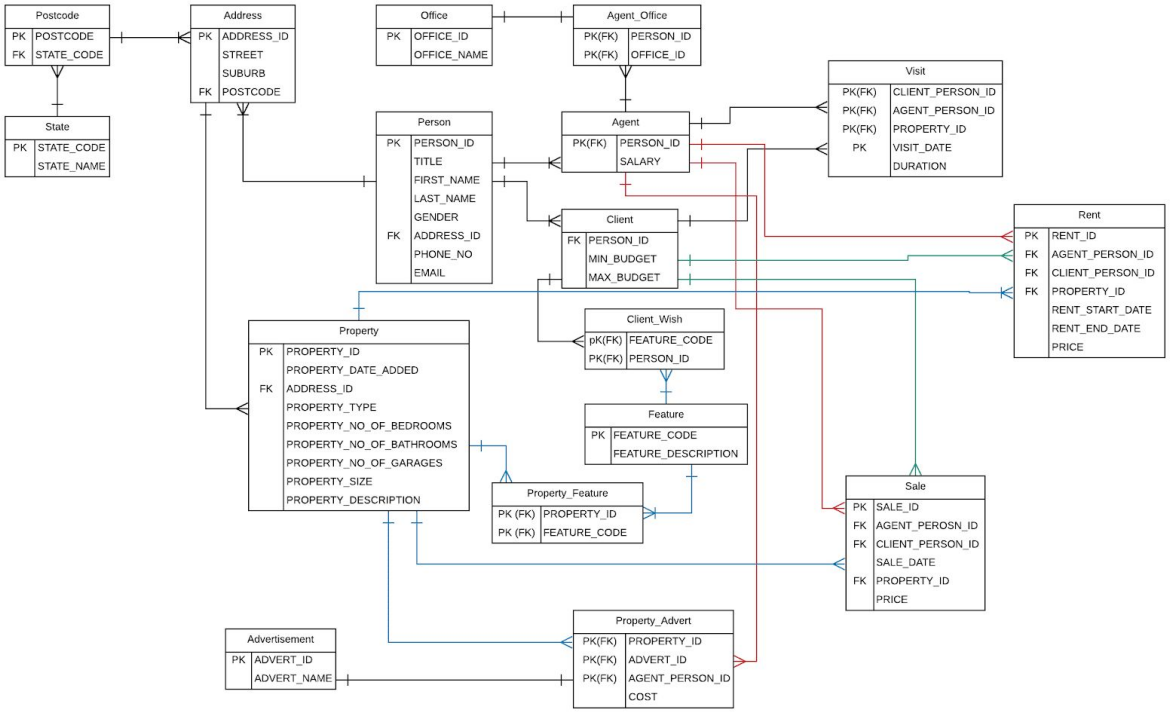


Fig1. ER diagram of MonRE database

C.1) Version-2: Low Aggregation (Level 0)

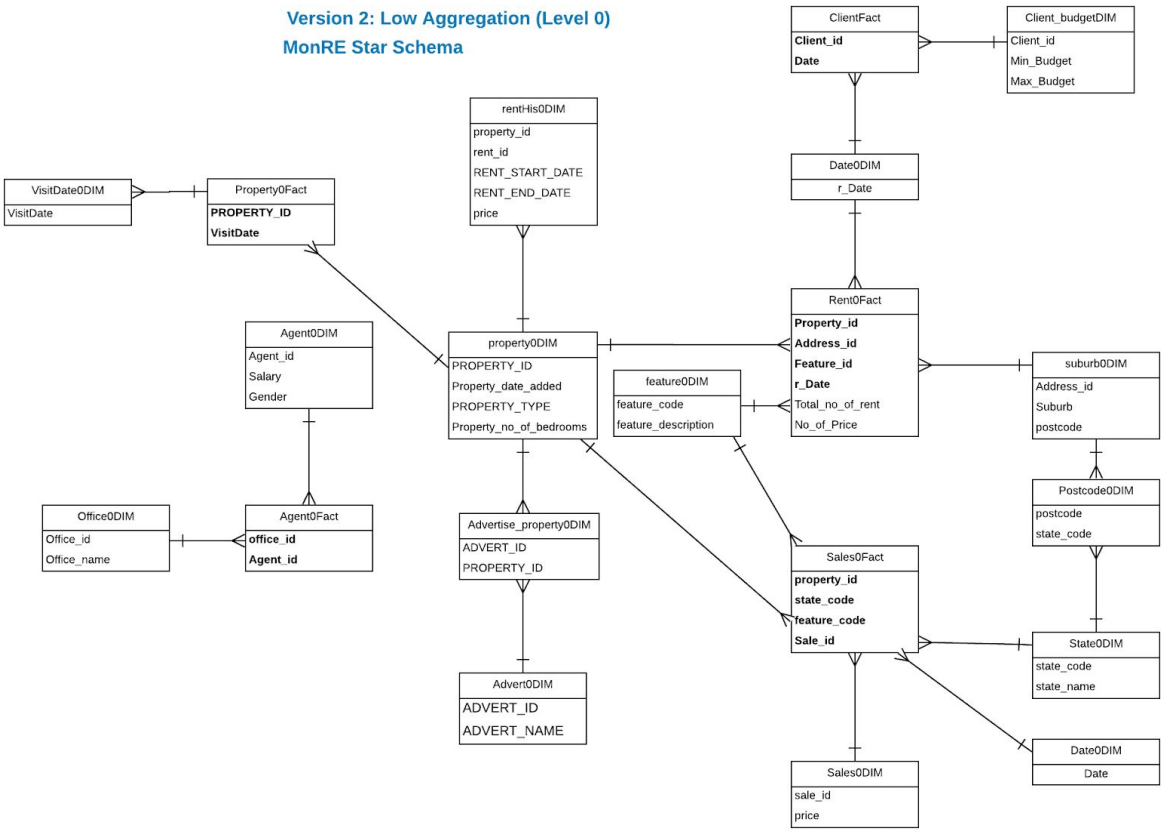


Fig 2: Star Schema with Low Aggregation (Level 1)

b)Data Cleaning process:

-Strategies used for data cleaning and Exploration:

Data Exploration :

1)The first step in exploring the database is to know the schema of the database.

Number of rows and columns is the first step to know your operational database

2)Next important thing to explore is the datatypes of the attributes which can help detect the wrong data formatted in the database.

3) After exploring the outer layer we explored every table and values in those tables, making sense of the information has been given great importance.

4) Next step we follow is to know the primary keys and identifying weak and strong entities which will help in further decision of designing data warehouse which may or may not contain bridge tables.

Data Cleaning:

1) By following the process above, we identified that some tables such as property and agent and client have null values , those null values have been identified and deleted in order to make the data clean.

2) Some of the attributes identified as having “unknown values” , such as state table. As this data makes no sense we decided to delete this data.

3) Attributes of the table agent and client budget contained negative numeric data for salary , maximum budget and minimum budget. This data has been identified as false data is removed from the database

4) Duplicate data has been a major issue with the databases in general. We took help of “distinct” keywords and stored those databases with new names and these cleaned tables then used to build the database.

5) Various anomalies were identified using conditions such as start date should be less than end data for a particular property and particular client and this data has been rectified.

6) The consistency of the data has been checked in the next phase , we used different join conditions to check the validity of the keys in different tables.

s-**Sql commands for data cleaning**

```
---Data Cleaning of MonRE database
-----------------------------------------------------------------------------------
--Data Cleaning of MonRE.Address table
-----------------------------------------------------------------------------------
-- For Checking number of rows
Select Count(*) from MonRE.Address;

Drop table Address;

Create table Address as
Select distinct Address_id, Street, Suburb, postcode
from MonRE.Address
order by Address_id;

--Checking for null values in all the columns in the  Address table
select Address_id
from address
where Address_id = null;

select Street
from address
where Street = null;

select Suburb
from address
where Suburb = null;

select postcode
from address
```

```
where postcode = null or postcode = 0;

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

--Data Cleaning of MonRE.Advertisement table

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

-- For Checking number of rows

Select Count(*) from MonRE.Advertisement;


Drop table Advertisement;


Create table Advertisement as

select *

from MonRE.Advertisemnt;


--To check duplicate values in advertisement table

select ADVERT_ID,

count(*)

from Advertisement

group by ADVERT_ID

having count(*) > 1;


--Checking for null values in all the columns in the  advertisement table

select ADVERT_ID

from Advertisement

where ADVERT_ID = null or ADVERT_ID = 0;


select ADVERT_NAME

from Advertisement

where ADVERT_NAME = null;

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

--Data Cleaning of MonRE.Agent table
```

```
-- For Checking number of rows
Select Count(*) from MonRE.Agent;

Drop table Agent;

Create table Agent
as select *
from MonRE.Agent;
--To check duplicate values in agent table
select PERSON_ID,
count(*)
from Agent
group by PERSON_ID
having count(*) > 1;

--Checking for null and negative values in all the columns in the  Agent table
select PERSON_ID
from Agent
where PERSON_ID = null or PERSON_ID = 0;

select SALARY
from Agent
where SALARY= null or SALARY<= 0;
```

	SALARY
1	-100000
2	-120000
3	0

```
Select Count(*) from Agent;
```

	COUNT(*)
1	2469

--Deleting negative values of salary from the Agent table to clean the data

Delete from Agent where Salary <= 0;

```
3 rows deleted.
```

	COUNT(*)
1	2466

--Checking if Person_id from perosn table mattches with Person_id from Agent table

select * from Agent

where PERSON_ID not in

(select PERSON_ID from MonRE.Person);

	PERSON_ID	SALARY
1	6997	0

--Deleting row with person_id from agent table does not present in MonRE.Person table

Delete from Agent

where PERSON_ID not in

(select PERSON_ID from MonRE.Person);

```
1 row deleted.
```

--Data Cleaning of MonRE.Agent_Office table

-- For Checking number of rows

Select Count(*) from MonRE.Agent_Office;

	COUNT(*)
1	2529

```
Drop table Agent_office;

Create table Agent_office

as select *

from MonRE.Agent_office;


--To check duplicate values in agent table

select PERSON_ID,

count(*)

from Agent_Office

group by PERSON_ID

having count(*) > 1;


--As agent are allowed to work in different offices, duplicate values present here are acceptable.

--Checking for null values in all the columns in the  Agent_Office table

select PERSON_ID

from Agent_Office

where PERSON_ID = null or PERSON_ID = 0;


select OFFICE_ID

from Agent_office

where OFFICE_ID = null or OFFICE_ID = 0;


--Checking if PERSON_ID from MonRE.perosn table matches with PERSON_ID from Agent_Office
table

select * from Agent_Office

where PERSON_ID not in

(select PERSON_ID

from MonRE.person);
```

	PERSON_ID	OFFICE_ID
1	6997	1177

--Deleting row with PERSON_ID from Agent_Office table does not present in MonRE.Person table

```
Delete from Agent_Office
where PERSON_ID not in
(select PERSON_ID
from MonRE.person);
```

```
1 row deleted.
```

--Data Cleaning of MonRE.Client table

-- For Checking number of rows

```
Select Count(*) from MonRE.Client;
```

	COUNT(*)
1	3339

```
Drop table Client;

Create table Client
as select *
from MonRE.Client;

--To check duplicate values in Client table
select PERSON_ID,
count(*)
from Client
group by PERSON_ID
having count(*) > 1;

--Checking negative and null values of budget in Client table
```

```
Select * from Client
where Min_Budget < 0 or Min_Budget = null;
Select * from Client
where Max_Budget <= 0 or Max_Budget > 10000000 or Max_Budget = null;
```

	PERSON_ID	MIN_BUDGET	MAX_BUDGET
1	5901	3500	-150

```
--Deleting row with Max_Budget is negative
Delete from Client
where Max_Budget <= 0 or Max_Budget > 10000000 or Max_Budget = null;
```

```
1 row deleted.
```

```
--Checking if PERSON_ID from Monre.person table matches with PERSON_ID from Client table
select * from Client
where PERSON_ID not in
(select PERSON_ID
from MonRE.person);
```

	PERSON_ID	MIN_BUDGET	MAX_BUDGET
1	7000	8500	15050

```
--Deleting row with PERSON_ID from Agent_Office table does not present in MonRE.person table
Delete from Client
where PERSON_ID not in
(select PERSON_ID
from MonRE.person);
```

```
1 row deleted.
```

--Data Cleaning of MonRE.Client_Wish table

```
Select Count(*) from MonRE.Client_Wish;
```

	COUNT(*)
1	1204

```
Drop table Client_Wish;


Create table Client_Wish
as select *
from MonRE.Client_Wish;


--Checking negative and null values of budget in Client_Wish table
Select * from Client_Wish
where feature_code <= 0 or feature_code = null;
--No null values


Select * from Client_Wish
where Person_id <= 0 or Person_id = null;
--No null values


--Checking if PERSON_ID from Monre.person table matches with PERSON_ID from Client_Wish table
select * from Client_Wish
where PERSON_ID not in
(select PERSON_ID
from MonRE.person);
--No invalid input found


--Checking if Feature_code from Monre.Feature table matches with Feature_code from Client_Wish
table
select * from Client_Wish
where Feature_code not in
(select Feature_code
```

```
from MonRE.Feature);
--No invalid input found


-----------------------------------------------------------------------------------
--Data Cleaning of MonRE.Feature table
-----------------------------------------------------------------------------------
Select Count(*) from MonRE.Feature;
```

	COUNT(*)
1	726

```
Drop table Feature;


Create table Feature
as select *
from MonRE.Feature;


---For Checking for duplicates values in Feature table
Select feature_code ,count(*)
from feature
group by feature_code
Having Count(*) > 1;
--No duplicate values found


--Checking negative and null values in Feature table
Select * from Feature
where feature_code <= 0;
--No null values


Select * from Feature
where FEATURE_DESCRIPTION = null;
--No null values
```

--Data Cleaning of MonRE.Office table

Select Count(*) from MonRE.Office;

	COUNT(*)
1	1177

Drop table Office;

Create table Office

as select *

from MonRE.Office;

--Checking negative and null values in Office table

Select * from office

where Office_id <= 0 or Office_id = null;

--No null values

--Checking null values of column office_name in Office table

Select * from office

where Office_name = null;

--No null values

--Checking if Office_id from Monre.Agent_Office table mattches with Office_id from Office table

select * from Office

where Office_id not in

(select Office_id

from MonRE.Agent_Office);

-- There are 3 agent offices in office table which are not allocated to any agent yet that's why it is not present in Agent_office table

```
-----------------------------------------------------------------------------------------
--Data Cleaning of MonRE.Person table
-----------------------------------------------------------------------------------------
Select Count(*) from MonRE.Person;
```

	COUNT(*)
1	7000

```
Drop table Person;

Create table Person
as select *
from MonRE.Person;
--For checking duplicates values
Select Person_id,
count(*)
from person
group by Person_id
Having Count(*) > 1;

--To include only distinct rows and removing duplicate rows
Drop table Person;

Create table Person
as select distinct *
from MonRE.Person;

--For checking duplicates values after creating table by using Distinct
Select Person_id,
count(*)
from person
group by Person_id
```

Having Count(*) > 1;

PERSON_ID	COUNT(*)

--Checking Null and negative values in the Person table

Select * from Person

where Person_id <= 0 or Person_id = null;

Select * from Person

where Title = null or Title = 'Unknown';

Select * from Person

where First_name = null or First_name = 'Unknown';

Select * from Person

where Last_name = null or Last_name = 'Unknown';

Select * from Person

where Gender = null or Gender = 'Unknown';

Select * from Person

where Address_id = null or Address_id <= 0;

Select * from Person

where Phone_no = null or Phone_no <= 0;

Select * from Person

where Email = null or Email = 'Unknown';

--For checking duplicates values

Select PHONE_NO,

```
count(*)
from person
group by PHONE_NO
Having Count(*) > 1;

Select EMAIL,
count(*)
from person
group by EMAIL
Having Count(*) > 1;
```

--Checking if address_id from Monre.Address table matches with address_id from MonRE.Address table

```
select * from person
where address_id not in
(select address_id
from MonRE.address);
```

	PERSON_ID	TITLE	FIRST_NAME	LAST_NAME	GENDER	ADDRESS_ID	PHONE_NO	EMAIL
1	7001	null	null	null	Male	13205	9-(999)999-9999	null

--Deleting the row with address_id from person table which does not match address_id from Monre.Address table

```
Delete from person
where address_id not in
(select address_id
from MonRE.address);
```

--One row gets deleted

```
1 row deleted.
```

--Data Cleaning of MonRE.Postcode table

Select Count(*) from MonRE.Postcode;

	COUNT(*)
1	689

Drop table Postcode;

Create table Postcode

as select *

from MonRE.Postcode;

--To check duplicate values in postcode table

select Postcode,

count(*)

from Postcode

group by Postcode

having count(*) > 1;

--Checking Null and negative values in the postcode table

Select * from Postcode

where Postcode <= 0 or Postcode = null;

Select * from Postcode

where State_code = null;

--Checking if State_code from Monre.State table matches with State_code from MonRE.State table

select * from Postcode

where State_code NOT IN

(select State_code from MonRE.State);

--Data Cleaning of MonRE.Property table

Select Count(*) from MonRE.Property;

	COUNT(*)
1	6226

Drop table Property;

Create table Property

as select *

from MonRE.Property;

--To check duplicate values in Property table

select Property_id,

count(*)

from Property

group by Property_id

having count(*) > 1;

	PROPERTY_ID	COUNT(*)
1	6177	4
2	6179	16

--To select only distinct values into Property table

Drop table Property;

Create table Property

as select distinct *

from MonRE.Property;

--To check duplicate values in Property table after using distinct

select Property_id,

count(*)

```
from Property
group by Property_id
having count(*) > 1;
```

PROPERT...	COUNT(*)

```
select Address_id,
count(*)
from Property
group by Address_id having count(*) > 1;
--Checking Null and negative values in the Property table
Select * from Property
where Property_id <= 0 or Property_id = null;

Select * from Property
where Property_date_added = null;

Select * from Property
where Address_id <= 0 or Address_id = null;

Select * from Property
where Property_type = null;

Select * from Property
where Property_type = null;

Select * from Property
where PROPERTY_NO_OF_BEDROOMS <= 0 or PROPERTY_NO_OF_BEDROOMS = null;

Select * from Property
```

```
where PROPERTY_NO_OF_BATHROOMS < 0 or PROPERTY_NO_OF_BATHROOMS = null;

Select * from Property
where PROPERTY_NO_OF_GARAGES < 0 or PROPERTY_NO_OF_GARAGES = null;

Select * from Property
where PROPERTY_SIZE < 0;

Select * from Property
where PROPERTY_DESCRIPTION = null;
--- Checking if address_id from Address table matches with address_id from property table
select *
from property
where address_id not in
(select address_id
from address);

-----------------------------------------------------------------------------------
--Data Cleaning of MonRE.Property_Advert table
-----------------------------------------------------------------------------------
Select Count(*) from MonRE.Property_Advert;
```

	COUNT(*)
1	3646

```
Drop table Property_Advert;

Create table Property_Advert
as select *
from MonRE.Property_Advert;

--Checking Null and negative values in the property_advert table
```

```
Select * from Property_Advert
where Property_id <= 0 or Property_id = null;

Select * from Property_Advert
where ADVERT_ID <= 0 or ADVERT_ID = null;

Select * from Property_Advert
where AGENT_PERSON_ID <= 0 or AGENT_PERSON_ID = null;

Select * from Property_Advert
where COST<= 0 or COST = null;

--Cheking entries for Advert_id, Property_id and Agent_person_id
select * from Property_Advert
where property_id
not in (select property_id
from property);

select * from Property_Advert
where agent_person_id
not in (select person_id
from person);

select * from Property_Advert
where advert_id
not in (select advert_id
from Advertisement);
-----------------------------------------------------------------------------------
--Data Cleaning of MonRE.Property_Feature table
-----------------------------------------------------------------------------------
```

```
Select Count(*) from MonRE.Property_Feature;
```

	COUNT(*)
1	30373

```
Drop table Property_Feature;

Create table Property_Feature
as select *
from MonRE.Property_Feature;

--Checking Null and negative values in the property_advert table
Select * from Property_Feature
where Property_id <= 0 or Property_id = null;

Select * from Property_Feature
where FEATURE_CODE <= 0 or FEATURE_CODE = null;

--- Checking for the feature_code and property id
select * from Property_Feature
where FEATURE_CODE not in
(select FEATURE_CODE
from feature);

select * from Property_Feature
where property_id not in
(select property_id
from property);
-----------------------------------------------------------------------------------
--Data Cleaning of MonRE.Rent table
-----------------------------------------------------------------------------------
Select Count(*) from MonRE.Rent;
```

	COUNT(*)
1	3284

```
Drop table Rent;

Create table Rent

as select *

from MonRE.Rent;


--To check duplicate values in Rent table

select rent_id,

count(*)

from Rent

group by rent_id

having count(*) > 1;


--Checking Null and negative values in the Rent table

Select * from Rent

where Rent_id <= 0 or Rent_id = null;


Select * from Rent

where AGENT_PERSON_ID <= 0 or AGENT_PERSON_ID = null;


Select * from Rent

where CLIENT_PERSON_ID<= 0 or CLIENT_PERSON_ID= null;


Select * from Rent

where PROPERTY_ID <= 0 or PROPERTY_ID = null;


Select * from Rent

where Price <= 0 or Price = null;
```

```
--Checking for property_id or Client_person_id
select * from Rent
where Property_id not in
(select Property_id
from Property);

select * from Rent
where AGENT_PERSON_ID not in
(select person_id
from Agent);

select * from Rent
where Client_person_id not in
(select person_id
from Client);
```

	RENT_ID	AGENT_PERSON_ID	CLIENT_PERSON_ID	PROPERTY_ID	RENT_START_DATE	RENT_END_DATE	PRICE
1	3284	6002	6001	5741	31-12-21	01-06-19	500

```
--Deleting rows from Rent which values does not match with Person_id from Client table
Delete from Rent
where Client_person_id not in
(select person_id
from Client);
```

```
1 row deleted.
```

```
--------------------------------------------------------------------------------
--Data Cleaning of MonRE.Sale table
--------------------------------------------------------------------------------
Select Count(*) from MonRE.Sale;
```

	COUNT(*)
1	2925

```
Drop table Sale;

Create table Sale

as select *

from MonRE.Sale;


--To check duplicate values in Sale table

select Sale_id,

count(*)

from Sale

group by Sale_id

having count(*) > 1;


--Checking Null or negative values in the Sale table

Select * from Sale

where Sale_id <= 0 or Sale_id = null;


Select * from Sale

where AGENT_PERSON_ID <= 0 or AGENT_PERSON_ID = null;


Select * from Sale

where Client_person_id <= 0 or Client_person_id = null;


Select * from Sale

where PRICE <= 0 or PRICE = null;


Select * from Sale

where PROPERTY_ID <= 0 or PROPERTY_ID = null;
```

```
select * from Sale
where AGENT_PERSON_ID not in
(select person_id
from Agent);
--All entries are correct
select * from Sale
where Client_person_id not in
(select person_id
from Client);
--All entries are correct

select * from Sale
where property_id not in
(select property_id
from property);
--All entries are correct
```

--Data Cleaning of MonRE.State table

```
Select Count(*) from MonRE.State;
```

	COUNT(*)
1	9

```
Drop table State;

Create table State
as select *
from MonRE.State;

--Checking Null and negative values in the State table
Select * from State
```

```
where State_Code = null;

select * from State

where State_name = null or State_name = 'Unknown';

--Null and unknown value is present
```

	STATE_CODE	STATE_NAME
1	(null)	Unknown

```
--Deleting rows with null values

Delete from State

where State_name = null or State_name = 'Unknown';
```

```
1 row deleted.
```

```
-----------------------------------------------------------------------------------

--Data Cleaning of MonRE.Visit table

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

Select Count(*) from MonRE.Visit;
```

	COUNT(*)
1	575

```
Drop table Visit;


Create table Visit

as select *

from MonRE.Visit;


select * from Visit

where agent_person_id = null  or agent_person_id <= 0;


select * from Visit

where property_id = null  or property_id <= 0 ;


select * from Visit
```

```
where duration = null  or duration <= 0 ;

select * from Visit
where client_person_id = null  or client_person_id <= 0 ;

--Checking for property_id , Client_person_id , agent_person_id
select * from Visit
where property_id not in
(select property_id
from property);

select * from Visit
where agent_person_id
not in (select person_id
from agent);

select * from Visit
where Client_person_id not in
(select person_id
from Client);
```

	CLIENT_PERSON_ID	AGENT_PERSON_ID	PROPERTY_ID	VISIT_DATE	DURATION
1	6000	6001	5741	31-12-99	5

```
--Deleting rows with unwanted values
Delete from Visit
where Client_person_id not in
(select person_id
from Client);
```

```
1 row deleted.
```

d) A short explanation of why you chose hierarchy or non-hierarchy

While designing the data warehouse "state-suburb-address" pair was the prime candidate for hierarchy but we did not choose hierarchy because they make the data warehouse less efficient in terms of query writing as we need to access suburbs and addresses number of times and accessing them through state everytime would need 3 joins which is avoided by not using them.

e) The reasons of the choice of SCD type for temporal dimension

SCD 4 has been the choice for data warehouses for the rent price attribute which changes every year.

Reason for choosing SCD4 are

1) Maintaining complete history for better analysis of trend and prediction
2) We decided to have the same ID for the same property and not different for each row.

f) A short explanation of the difference among the two versions of star/snowflake schema.

Star schema in version 2 has aggregated values whereas star schema in version 2 has no aggregate values.

Star schema level 0 data is similar to an operational database while in level 2 we have both data directly from operational data and newly created aggregated tables from operational databases.

Number of dimensions in level 0 start schema are more than level 2

Task C.2

a) SQL statements to create the star/snowflake schema Version-1

```
---Creating star schema for property with Aggregation level 2

--Creating VisitTime2DIM dimension

Drop table visitTime2DIM;

Create table VisitTime2DIM as

select

to_char(VISIT_DATE, 'MM-DD-YYYY') as VisitDate

from visit;


select * from VisitTime2DIm;
```

	VISITDATE
1	24-03-2020
2	22-03-2020
3	04-04-2020
4	12-03-2020
5	29-03-2020

```
--Creating property2DIM dimension

Drop table Property2DIM;

Create table Property2DIM as

select

   PROPERTY_ID,

   to_char(PROPERTY_DATE_ADDED, 'MM') as Month,

   to_char(PROPERTY_DATE_ADDED, 'YYYY') as Year,

   PROPERTY_TYPE

from property;

select * from Property2DIM;
```

	PROPERTY_ID	MONTH	YEAR	PROPERTY_TYPE
1	28	04	2020	House
2	36	03	2020	House
3	46	04	2020	House
4	129	11	2019	House
5	131	11	2019	House

--Creating Advert2DIM dimension

Drop table Advert2DIM;

Create table Advert2DIM as

select * from advertisement;

select * from Advert2DIM;

	ADVERT_ID	ADVERT_NAME
1	1	Rent Apartment / Unit / Flat
2	2	Rent Block of Units
3	3	Rent Duplex
4	4	Rent House
5	5	Rent New Apartments / Off the Plan

--Creating Advertise_property2DIM

Drop table Advertise_property2DIM;

Create table Advertise_property2DIM as

select

ADVERT_ID,

PROPERTY_ID

From Property_Advert;

select * from advertise_property2dim;

	ADVERT_ID	PROPERTY_ID
1	16	2894
2	16	2895
3	16	2896
4	16	2897
5	16	2898

--Create TempSeasonDIM dimension

Drop table TempSeason2DIM;

```
Create table TempSeason2DIM as

select

   to_char(VISIT_DATE, 'MM') as Visit_Month

from Visit;


Alter table TempSeason2DIM add SeasonID number(2);

Alter table TempSeason2DIM add Season_type varchar(20);


update TempSeason2DIM set SeasonID = 1, season_type = 'Summer' where visit_month >'01'
and visit_month <= '03';

update TempSeason2DIM set SeasonID = 2, season_type = 'Autumn' where visit_month >'03'
and visit_month <= '06';

update TempSeason2DIM set SeasonID = 3, season_type = 'Winter' where visit_month > '06' and
visit_month <= '09';

update TempSeason2DIM set SeasonID = 4, season_type = 'Spring' where visit_month > '09' and
visit_month <= '12';


---Creating SeasonDIM dimension

Drop table SeasonDIM;

Create table SeasonDIM as Select Distinct SeasonID, Season_type from TempSeason2DIM;

select * from SeasonDIM;
```

	SEASONID	SEASON_TYPE
1	1	Summer
2	2	Autumn

```
---Create TempPropertyFact fact dimension

drop table TempProperty2Fact;

Create table TempProperty2Fact as

select

to_char(a.VISIT_DATE, 'MMDDYYYY') || client_person_id as VisitID,

to_char(a.VISIT_DATE, 'MM-DD-YYYY') as VisitDate,
```

```
to_char(a.VISIT_DATE, 'MM') as Visit_Month,
b.property_type,
b.property_date_added,
c.advert_name,
d.advert_id,
d.property_id
from Visit a, Property b, Advertisement c, Property_advert d
where a.property_id = b.property_id
and d.property_id = b.property_id
and d.advert_id = c.advert_id;

Alter table TempProperty2Fact add SeasonID number(2);
Alter table TempProperty2Fact add Season_type Varchar(20);

update TempProperty2Fact set SeasonID = 1, season_type = 'Summer' where visit_month >'01'
and visit_month <= '03';
update TempProperty2Fact set SeasonID = 2, season_type = 'Autumn' where visit_month >'03'
and visit_month <= '06';
update TempProperty2Fact set SeasonID = 3, season_type = 'Winter' where visit_month > '06'
and visit_month <= '09';
update TempProperty2Fact set SeasonID = 4, season_type = 'Spring' where visit_month > '09'
and visit_month <= '12';

--Creating propertyFact fact dimension
Drop table Property2Fact;
Create table Property2Fact as
Select
visitdate,
seasonID,
Property_id,
count(property_id) as Total_no_of_property,
```

```
Sum(visitid) as Total_no_of_visit,
Count(Visitid) as No_of_visit
from tempproperty2fact
group by Visitdate, seasonID, property_id;

select * from Property2Fact;
```

	VISITDATE	SEASONID	PROPERTY_ID	TOTAL_NO_OF_PROPERTY	TOTAL_NO_OF_VISIT	NO_OF_VISIT
1	03-16-2020	1	1841	1	31620205187	1
2	03-13-2020	1	1868	3	93960615487	3
3	03-07-2020	1	1697	1	30720205050	1
4	03-27-2020	1	1709	2	65440410173	2
5	03-11-2020	1	1511	3	93360615281	3

```
--Creating State2DIM dimension
Drop table State2DIM;

create table state2DIM as
select state_code, state_name
from state;

select * from state2DIM;
```

	STATE_CODE	STATE_NAME
1	ACT	Australian Capital Territory
2	NSW	New South Wales
3	NT	Northern Territory
4	QLD	Queensland
5	SA	South Australia
6	TAS	Tasmania
7	VIC	Victoria
8	WA	Western Australia

```
--Creating Feature2DIM dimension
DROP table TempFeature2DIM;
DROP table Feature2DIM;
```

```
create table tempfeature2DIM

as select  a.property_id, count(a.Feature_code) as count

from property_feature a, property b

where a.property_id = b.property_id

Group by a.property_id;


Alter table tempfeature2DIM add feature_id number(3);

Alter table tempfeature2DIM add feature_type varchar(20);


update tempfeature2DIM set feature_id = 1, feature_type = 'Very Basic' where count < 10;

update tempfeature2DIM set feature_id = 2, feature_type = 'Standard' where count >= 10 and
count <= 20;

update tempfeature2DIM set feature_id = 3, feature_type = 'Luxurious' where count > 20;


Create table feature2DIM as

select distinct feature_id,

feature_type

from tempfeature2DIM;


select * from Feature2DIM;
```

	FEATURE_ID	FEATURE_TYPE
1	1	Very Basic
2	2	Standard
3	3	Luxurious

```
--Creating Property2DIM dimension

Drop Table Property2DIM;

create table property2DIM as

select property_id,to_char(property_date_added,'mm') as
month,to_char(property_date_added,'yyyy')as year,property_type from property;
```

select * from property2DIM;

	PROPERTY_ID	MONTH	YEAR	PROPERTY_TYPE
1	28	04	2020	House
2	36	03	2020	House
3	46	04	2020	House
4	129	11	2019	House
5	131	11	2019	House

--Creating TempSales2Fact

drop table Tempsales2Fact;

create table Tempsales2Fact as

select p.property_id ,

st.state_code,

count(f.feature_code) as count,

to_char(s.sale_date, 'YYYY') as Year,

s.price,

s.sale_id

from property p, state st, property_feature f, sale s, address a, postcode p

where p.property_id = f.property_id

and f.property_id = s.property_id

and p.address_id = a.address_id

and a.postcode = p.postcode

and p.state_code = st.state_code

group by p.property_id , st.state_code,to_char(s.sale_date, 'YYYY'), s.price, s.sale_id;

Alter table Tempsales2Fact add feature_id number(3);

Alter table Tempsales2Fact add feature_type varchar(20);

update Tempsales2Fact set feature_id = 1, feature_type = 'Very Basic' where count < 10;

update Tempsales2Fact set feature_id = 2, feature_type = 'Standard' where count >= 10 and count <= 20;

```
update Tempsales2Fact set feature_id = 3, feature_type = 'Luxurious' where count > 20;

--Creating Sales2Fact table
Drop table Sales2Fact;

Create table sales2Fact as select
t.property_id,
t.state_code,
t.feature_id,
t.year,
count(sale_id) as total_sales_count,
sum(price) as total_sales
from Tempsales2Fact t
group by t.property_id,t.state_code,t.feature_id, t.year;

select * from Sales2Fact;
```

	PROPERTY_ID	STATE_CODE	FEATURE_ID	YEAR	TOTAL_SALES_COUNT	TOTAL_SALES
1	144	SA	1	(null)	1	369000
2	284	SA	2	(null)	1	565000
3	599	QLD	1	(null)	1	600000
4	622	QLD	1	(null)	1	500000
5	824	QLD	1	(null)	1	760000
6	963	NSW	1	2020	1	1200000

```
--Create Postcode2DIM dimension
Drop table Postcode2DIM;

create table postcode2DIM as select
postcode,state_code from
postcode;
```

select * from postcode2dim;

	POSTCODE	STATE_CODE
1	7008	TAS
2	7007	TAS
3	7005	TAS
4	7004	TAS
5	7000	TAS
6	6171	WA
7	6170	WA
8	6169	WA

```
create table suburb2DIM as select
p.postcode,
a.suburb
from postcode p , address a
where a.postcode= p.postcode;
```

select * from Suburb2DIM;

	POSTCODE	SUBURB
1	3215	North Geelong
2	3217	Charlemont
3	3218	Geelong West
4	3225	Queenscliff
5	3220	Geelong

```
--Create RentHis2Dim Dimension
Drop table RentHis2DIM;

create table rentHis2DIM as select
property_id,
rent_id,
RENT_START_DATE ,
RENT_END_DATE ,
price
from rent;
```

select * from RentHis2DIM;

	PROPERTY_ID	RENT_ID	RENT_START_DATE	RENT_END_DATE	PRICE
1	6199	331	12-01-20	28-06-20	795
2	6063	332	02-05-20	18-10-20	500
3	6074	333	01-05-20	17-10-20	370
4	6142	334	12-02-20	29-07-20	795
5	6146	335	20-04-20	06-10-20	595
6	5373	336	27-04-20	13-10-20	350

```
--Create Property_scale2DIM dimension;
Drop table Property_Scale2DIM;

create table property_scale2DIM(
scale_id number,
scale_desc varchar(20));

insert into  property_scale2DIM values
('1','Extra Small');
insert into  property_scale2DIM values
('2','Small');
insert into  property_scale2DIM values
('3','Medium');
insert into  property_scale2DIM values
('4','Large');
insert into  property_scale2DIM values
('5','Extra Large');
```

select * from Property_Scale2DIM;

	SCALE_ID	SCALE_DESC
1	1	Extra Small
2	2	Small
3	3	Medium
4	4	Large
5	5	Extra Large

--Creating Rent_year2DIM dimension

Drop table Year2DIM;

create table Year2DIM as (select

distinct

to_char(rent_start_date,'yyyy') as year

from rent

Union

select

distinct

to_char(Sale_date, 'YYYY') as year

from sale) ;

Select * from Year2DIM;

	YEAR
1	2019
2	2020

--Creating tempRent2Fact table

drop table tempRent2Fact;

create table tempRent2Fact as select

p.property_id,

po.postcode,

```
to_char(rn.rent_start_date,'yyyy') as year,
p.property_no_of_bedrooms as rooms,
rn.rent_id,
rn.price,
count(f.feature_code) as count
from property p, postcode po, property_feature f, address a, rent rn
where p.property_id = f.property_id
and f.property_id = rn.property_id
and p.address_id = a.address_id
and a.postcode = po.postcode
group by
p.property_id,
po.postcode,
to_char(rn.rent_start_date,'yyyy'),
p.property_no_of_bedrooms,
rn.rent_id,
rn.price;
```

```
alter table temprent2Fact add (scale_id  numeric);
alter table temprent2Fact add (scale_desc varchar(20));

update temprent2Fact
set scale_id = '1',scale_desc = 'Extra Small'where rooms <= '1';
update temprent2Fact
set scale_id = '2',scale_desc = 'Small' where rooms <= '03' and rooms >= '02';
update temprent2Fact
set scale_id = '3',scale_desc = 'Medium' where rooms <= '06' and rooms >= '04';
update temprent2Fact
set scale_id = '4',scale_desc = 'Large' where rooms <= '10' and rooms >= '07';
```

```
update temprent2Fact

set scale_id = '5',scale_desc = 'Extra Large' where rooms > '10';


Alter table temprent2Fact add feature_id number(3);

Alter table temprent2Fact add feature_type varchar(20);


update temprent2Fact set feature_id = 1, feature_type = 'Very Basic' where count < 10;

update temprent2Fact set feature_id = 2, feature_type = 'Standard' where count >= 10 and count <=
20;

update temprent2Fact set feature_id = 3, feature_type = 'Luxurious' where count > 20;


---Creating Rent2Fact table

Drop table Rent2Fact;


create table Rent2Fact as select

t.property_id,

t.postcode,

t.feature_id,

t.year,

t.scale_id,

count(t.rent_id) as total_no_of_rent,

sum(t.price) as no_of_price

from temprent2Fact t

group by t.property_id, t.postcode, t.feature_id, t.year, t.scale_id;

select * from Rent2Fact;
```

	PROPERTY_ID	POSTCODE	FEATURE_ID	YEAR	SCALE_ID	TOTAL_NO_OF_RENT	NO_OF_PRICE
1	3020	5034	1	(null)	2	1	360
2	3143	4005	1	(null)	1	1	360
3	3416	4500	1	(null)	2	1	340
4	3509	4172	2	2020	3	1	650
5	3545	4104	2	2020	3	1	825

--Creating office2DIM Dimension

```
drop table Office2DIM;

create table Office2DIM as
select distinct
office_id,
office_name
from office;

select * from Office2DIM;
```

	OFFICE_ID	OFFICE_NAME
1	916	Ray White Mount Gravatt
2	919	Ray White Nolan & Iken
3	937	Ray White Robina
4	955	Ray White Upper Coomera
5	965	Ray White at The Entertainment Quarter
6	966	Rayner Real Estate

--- Creating Agentgender2DIM dimension

```
drop table Agentgender2DIM;

create table Agentgender2DIM
as select distinct Gender as gender_type
from person;

alter table Agentgender2DIM add gender_id number(2);

update Agentgender2DIM set gender_id = 1  Where gender_type = 'Male' ;
update Agentgender2DIM set gender_id = 2 Where gender_type = 'Female' ;
```

select * from agentgender2dim;

	GENDER_TYPE	GENDER_ID
1	Male	1
2	Female	2

--- Creating TempOfficeType2DIM

drop table TempOfficetype2DIM;

create table TempOfficetype2DIM

as select Count(person_id) as number_of_employee, office_id from agent_office group by office_id ;

alter table TempOfficetype2DIM add office_type varchar(20);

alter table TempOfficetype2DIM add office_typeid number(2);

update TempOfficetype2DIM set office_typeid = 1 , office_type = 'Small' Where number_of_employee < 4;

update TempOfficetype2DIM set office_typeid = 2 , office_type = 'Medium' Where number_of_employee >= 4 and number_of_employee < 12 ;

update TempOfficetype2DIM set office_typeid = 3 , office_type = 'Big' Where number_of_employee > 12 ;

--Creating office_type2DIM dimension

drop table Office_type2DIM;

Create Table Office_type2DIM as

Select Distinct

office_typeid ,

office_type

from TempOfficetype2DIM;

select * from Office_type2DIM;

	OFFICE_TYPEID	OFFICE_TYPE
1	1	Small
2	2	Medium
3	3	Big

```
--Create TempAgent2fact
Drop table TempAgent2fact;

Create table TempAgent2fact as
select distinct
b.person_id as Agent_Id,
b.Salary,
o.office_id,
o.office_name,
Count(a.person_id) as number_of_employee,
p.Gender as gender_type
from office o, person p, agent_office a, agent b
where o.office_id = a.office_id
and a.person_id = b.person_id
and p.person_id = b.person_id
group by b.person_id,
b.Salary,
o.office_id,
o.office_name,
p.Gender ;

alter table TempAgent2fact add gender_id number(2);

update TempAgent2fact set gender_id = 1  Where gender_type = 'Male' ;
```

```
update TempAgent2fact set gender_id = 2 Where gender_type = 'Female' ;

--

alter table TempAgent2fact add officetype varchar(20);

alter table TempAgent2Fact add officetype_id number(2);


update TempAgent2Fact set officetype_id = 1 , officetype = 'Small'  Where number_of_employee  <
4;

update TempAgent2Fact set officetype_id = 2 , officetype = 'Medium' Where number_of_employee
>= 4  and  number_of_employee < 12 ;

update TempAgent2Fact set officetype_id = 3 , officetype = 'Big' Where number_of_employee > 12 ;


---Create final Agent2Fact table

Drop table Agent2Fact;


Create table Agent2Fact as

Select

officetype_id,

Gender_ID,

office_id,

Count(Agent_id) as Total_no_Agents,

Count(Salary) as Total_salary

from TempAgent2Fact

group by officetype_id,

Gender_ID,

office_id;
```

select * from Agent2Fact;

	OFFICETYPE_ID	GENDER_ID	OFFICE_ID	TOTAL_NO_AGENTS	TOTAL_SALARY
1	1	1	222	1	1
2	1	2	238	4	4
3	1	2	946	11	11
4	1	2	929	5	5
5	1	1	292	1	1

--Creating TempBudget2DIM dimension

Drop table TempBudget2DIM;

Create table TempBudget2DIM as

select

min_budget,

max_budget

from Client;

Alter table TempBudget2DIM add BudgetID number (2);

Alter table TempBudget2DIM add BudgetType Varchar(20);

update TempBudget2DIM set BudgetID = 1 , BudgetType= 'Low' Where Min_budget > 0 and Max_budget <= 1000 ;

update TempBudget2DIM set BudgetID = 2 , BudgetType= 'Medium' Where Min_budget >= 1001 and Max_budget <= 100000 ;

update TempBudget2DIM set BudgetID = 3 , BudgetType= 'High' Where Min_budget >= 100001 and Max_budget <= 10000000;

--Creating Budget2DIM dimension

Drop table Budget2DIM;

Create table Budget2DIM as

```
Select
BudgetID,
BudgetType
from TempBudget2DIM;

select * from Budget2DIM;
```

	BUDGETID	BUDGETTYPE
1	3	High
2	1	Low
3	2	Medium

```
--Creating TempClientFact table
Drop table TempClient2Fact;

Create table TempClient2Fact as
select
C.min_budget,
C.max_budget,
C.Person_id,
A.Year ,
A.Client
From
(select distinct
to_char(rent_start_date,'YYYY') as Year, Client_person_ID as Client
from rent
Union
Select Distinct
to_char(sale_date, 'YYYY') as Year , Client_person_id as Client
from sale)  A
inner join Client  C
On A.Client = C.person_id;
```

```
Alter table TempClient2Fact add  BudgetID number (2);

Alter table TempClient2Fact add BudgetType Varchar(20);


update TempClient2Fact set BudgetID = 1 , BudgetType= 'Low'  Where Min_budget > 0 and
Max_budget <= 1000 ;

update TempClient2Fact set BudgetID = 2 , BudgetType= 'Medium'  Where Min_budget >= 1001 and
Max_budget <= 100000 ;

update TempClient2Fact set BudgetID = 3 , BudgetType= 'High'  Where Min_budget >= 100001 and
Max_budget <= 10000000;


select * from TempClient2Fact;

--Creating Client2Fact table

Drop table Client2Fact;


create table Client2fact as

select

BudgetID,

Year,

count(client) as Total_no_of_clients

from tempclient2fact

group by

BudgetID,

Year;

select * from Client2Fact;
```

	BUDGETID	YEAR	TOTAL_NO_OF_CLIENTS
1	3	2019	23
2	1	2020	1446
3	3	2020	891
4	2	2019	3
5	2	2020	96
6	1	2019	19

b) SQL statements (e.g. create table, insert into, etc) to create the star/snowflake schema Version-2

```
---Creating star schema for property with Aggregation level 0

--Creating VisitDate0DIM dimension

Drop table VisitDate0DIM;

Create table VisitDate0DIM as

select Distinct

 to_char(VISIT_DATE, 'MM-DD-YYYY') as VisitDate

from visit;


Select * from VisitDate0DIM;
```

	VISITDATE
1	03-28-2020
2	04-04-2020
3	04-05-2020
4	03-19-2020
5	03-16-2020

```
----Creating property0DIM dimension

Drop table Property0DIM;


Create table Property0DIM as

select

   PROPERTY_ID,

   to_char(PROPERTY_DATE_ADDED, 'MM-DD-YYYY') as Property_date_added,

   PROPERTY_TYPE,

   property_no_of_bedrooms

from property;
```

Select * from Property0DIM;

	PROPERTY_ID	PROPERTY_DATE_ADDED	PROPERTY_TYPE	PROPERTY_NO_OF_BEDROOMS
1	28	04-08-2020	House	4
2	36	03-05-2020	House	5
3	46	04-19-2020	House	4
4	129	11-25-2019	House	2
5	131	11-19-2019	House	3
6	135	04-07-2020	House	3

--Creating Advert0DIM dimension

Drop table Advert0DIM;

Create table Advert0DIM as

select * from advertisement;

Select * from Advert0DIM;

	ADVERT_ID	ADVERT_NAME
1	1	Rent Apartment / Unit / Flat
2	2	Rent Block of Units
3	3	Rent Duplex
4	4	Rent House
5	5	Rent New Apartments / Off the Plan
6	6	Rent Penthouse

--Creating Advertise_property0DIM

Drop table Advertise_property0DIM;

Create table Advertise_property0DIM as

select

ADVERT_ID,

PROPERTY_ID

From Property_Advert;

Select * from Advertise_property0DIM;

	ADVERT_ID	PROPERTY_ID
1	16	2894
2	16	2895
3	16	2896
4	16	2897
5	16	2898
6	16	2899

```
--Creating TempProperty0Fact
Drop table TempProperty0Fact;

Create table TempProperty0Fact as
select distinct
a.client_person_id,
a.agent_person_id,
to_char(a.VISIT_DATE, 'MM-DD-YYYY') as VisitDate,
a.property_id,
count(a.property_id) as Total_property_visited
from Visit a, Property b
where a.property_id = b.property_id
group by a.VISIT_DATE,a.property_id,a.client_person_id,a.agent_person_id;

--Creating Property0Fact
Drop table Property0Fact;
Create table Property0Fact as
select
Visitdate,
property_id,
Total_property_visited
from TempProperty0Fact;
```

select * from Property0Fact;

	VISITDATE	PROPERTY_ID	TOTAL_PROPERTY_VISITED
1	03-28-2020	5347	1
2	04-08-2020	5439	1
3	03-28-2020	5638	1
4	04-07-2020	5336	1
5	03-17-2020	1650	1
6	03-16-2020	2109	1

--Creating State0DIM dimension

Drop table State0DIM;

create table state0DIM as

select state_code, state_name

from state;

Select * from State0DIM;

	STATE_CODE	STATE_NAME
1	ACT	Australian Capital Territory
2	NSW	New South Wales
3	NT	Northern Territory
4	QLD	Queensland
5	SA	South Australia
6	TAS	Tasmania
7	VIC	Victoria
8	WA	Western Australia

--Creating feature0DIM dimension

Drop table Feature0DIM;

Create table Feature0DIM as

Select

feature_code,

feature_description

from Feature;

Select * from Feature0DIM;

	FEATURE_CODE	FEATURE_DESCRIPTION
1	1	Air conditioning
2	2	Built in wardrobes
3	3	Carpeted
4	4	City Views
5	5	Close to schools
6	6	Close to shops

--Creating Property0DIM dimension

Drop Table Property0DIM;

create table property0DIM as

select

property_id,

to_char(property_date_added, 'mm-dd-yyyy') as Property_date_added,

property_type,

Property_no_of_bedrooms

from property;

Select * from Property0DIM;

	PROPERTY_ID	PROPERTY_DATE_ADDED	PROPERTY_TYPE	PROPERTY_NO_OF_BEDROOMS
1	28	04-08-2020	House	4
2	36	03-05-2020	House	5
3	46	04-19-2020	House	4
4	129	11-25-2019	House	2
5	131	11-19-2019	House	3

--Creating Sales0DIM Dimension

Drop table Sales0DIM;

Create table Sales0DIM as

Select

sale_id,

```
price
from Sale;

Select * from Sales0DIM;
```

	SALE_ID	PRICE
1	434	1395000
2	435	275000
3	436	3490000
4	437	799000
5	438	2000000
6	439	1825000

```
--Creating TempSales0Fact
Drop table TempSales0Fact;

Create table TempSales0fact as
select
a.Property_id,
b.State_code,
e.Feature_code,
d.Sale_id,
sum(d.price) as total_sale,
count(d.sale_id) as noOfSale
from Property a, state b, sale d, property_feature e, Address f, Postcode g
where a.property_id = d.property_id
and a.property_id = e.property_id
and a.property_id = d.property_id
and a.Address_id = f.Address_id
and f.postcode = g.postcode
group by a.Property_id,
b.State_code,
e.Feature_code,
```

d.Sale_id ;

--Creating Fact table for Sales0Fact

drop table sales0fact;

Create table Sales0Fact as

select *

from TempSales0Fact;

select * from sales0Fact;

	PROPERTY_ID	STATE_CODE	FEATURE_CODE	SALE_ID	TOTAL_SALE	NOOFSALE
1	9	ACT	4	2374	1250000	1
2	9	ACT	10	2374	1250000	1
3	9	ACT	12	2374	1250000	1
4	9	ACT	117	2374	1250000	1
5	9	NSW	6	2374	1250000	1

--Create Postcode0DIM dimension

Drop table Postcode0DIM;

create table postcode0DIM as select

postcode,

state_code from

postcode;

Select * from Postcode0DIM;

	POSTCODE	STATE_CODE
1	800	NT
2	2000	NSW
3	2007	NSW
4	2008	NSW
5	2009	NSW
6	2010	NSW

----Create Suburb0DIM dimension

Drop table Suburb0DIM;

```
create table suburb0DIM as select
a.Address_id,
p.postcode,
a.suburb
from postcode p ,
address a
where a.postcode= p.postcode;
```

Select * from Suburb0DIM;

	ADDRESS_ID	POSTCODE	SUBURB
1	1	3215	North Geelong
2	2	3217	Charlemont
3	3	3218	Geelong West
4	4	3225	Queenscliff
5	5	3220	Geelong
6	6	3220	Newtown

----Create RentHis0Dim Dimension

Drop table RentHis0DIM;

```
create table rentHis0DIM as select
property_id,
rent_id,
RENT_START_DATE ,
RENT_END_DATE ,
price
from rent;
```

Select * from RentHis0DIM;

	PROPERTY_ID	RENT_ID	RENT_START_DATE	RENT_END_DATE	PRICE
1	6199	331	12-01-20	28-06-20	795
2	6063	332	02-05-20	18-10-20	500
3	6074	333	01-05-20	17-10-20	370
4	6142	334	12-02-20	29-07-20	795
5	6146	335	20-04-20	06-10-20	595
6	5373	336	27-04-20	13-10-20	350

--Creating Date0DIM dimension

Drop table Date0DIM;

create table Date0DIM as (

select distinct

to_char(rent_start_date,'DD-MM-YYYY') as R_Date

from rent

Union

Select Distinct

to_char(sale_date, 'DD-MM-YYYY') as R_Date from sale) ;

Select * from Date0DIM;

	R_DATE
1	01-01-2020
2	01-02-2020
3	01-03-2020
4	01-04-2020
5	01-05-2020
6	02-01-2020

--Creating TempRent0Fact table

Drop table TempRent0Fact;

select * from rent;

Create table TempRent0Fact as

select

```
a.Property_id,
b.Address_id,
c.Feature_code,
d.rent_start_date as rent_date,
d.price
from property a, Address b, property_Feature c, Rent d
where a.address_id = b.Address_id
and a.property_id = c.property_id
and a.property_id = d.property_id
group by a.Property_id,
b.Address_id,
c.Feature_code,
d.rent_start_date,
d.price;
--Creating Rent0Fact table
Drop table Rent0Fact;

Create table Rent0fact as
select
t.Property_id,
t.Address_id,
t.Feature_code,
t.rent_date ,
count(t.price) as total_no_of_rent,
sum(t.price) as no_of_price
from TempRent0Fact t
group by t.Property_id,
t.Address_id,
t.Feature_code,
t.rent_date;
```

select * from rent0fact;

	PROPERTY_ID	ADDRESS_ID	FEATURE_CODE	RENT_DATE	TOTAL_NO_OF_RENT	NO_OF_PRICE
1	2968	2968	183	(null)	1	260
2	2974	2974	100	02-04-20	1	600
3	3000	3000	86	29-02-20	1	850
4	3002	3002	26	(null)	1	425
5	3019	3019	33	03-05-20	1	395
6	3019	3019	467	03-05-20	1	395

--Creating Agent0DIM dimension

Drop table Agent0DIM;

Create table Agent0DIM as

select

c.person_id as agent_id,

c.salary,

b.Gender

--Count(c.person_id) as number_of_employee

from Agent_office a, person b, agent c

where a.person_id = c.person_id

and b.person_id = c.person_id

group by c.person_id,

c.salary,

b.Gender;

select * from agent0dim;

	AGENT_ID	SALARY	GENDER
1	1056	190000	Female
2	1058	180000	Male
3	65	190000	Female
4	219	175000	Male
5	224	190000	Female
6	280	190000	Male

--Creating Office0DIM dimension

Drop table Office0DIM;

Create table Office0DIM as

select

office_id,

office_name

from Office;

Select * from Office0DIM;

	OFFICE_ID	OFFICE_NAME
1	910	Ray White Manly QLD
2	911	Ray White Mawson Lakes
3	912	Ray White Meadowbank
4	913	Ray White Metro West
5	914	Ray White Moorooka
6	915	Ray White Mordialloc

--creating temp fact

drop table tagent0fact;

Create table TAgent0Fact as

select

a.office_id,

c.person_id as Agent_id

from Agent_office a, agent c

where a.person_id = c.person_id

group by a.office_id,

c.person_id ;

--Creating Agent0Fact

Drop table Agent0Fact;

Create table Agent0Fact as

select

t.office_id,

t.agent_id,

count(t.agent_id) no_of_employee

from TAgent0Fact t

group by t.office_id,

t.agent_id;

select * from agent0fact;

	OFFICE_ID	AGENT_ID	NO_OF_EMPLOYEE
1	1069	365	1
2	235	964	1
3	1070	1898	1
4	332	380	1
5	622	989	1
6	667	2239	1

--Creating Client_budgetDIM dimension

Drop table Client_budgetDIM;

Create table Client_budgetDIM as

Select

person_id as Client_id,

```
Min_Budget,
Max_Budget
from Client;

Select * from Client_budgetDIM;
```

	CLIENT_ID	MIN_BUDGET	MAX_BUDGET
1	3014	440100	537900
2	3020	490500	599500
3	3025	585000	715000
4	3029	607500	742500
5	3081	449100	548900
6	3087	1350000	1650000

```
--Creating ClientFact table
Drop table TClientFact;

Create table TClientFact as
select
c.Person_id as Client_id,
r_Date
from
(select distinct
to_char(rent_start_date,'DD-MM-YYYY') as r_Date, Client_person_ID as Client
from rent
Union
Select Distinct
to_char(sale_date, 'DD-MM-YYYY') as r_Date, Client_person_id as Client
from sale) a
inner join Client  C
On A.Client = C.person_id;

create table client0fact as select
```

```
client_id,
R_date,
count(client_id) as total_no_client
from TClientFact t
group by client_id,
R_date;
```

```
select * from client0fact;
```

	CLIENT_ID	R_DATE	TOTAL_NO_CLIENT
1	3744	01-01-2020	1
2	4957	01-03-2020	1
3	2725	01-04-2020	1
4	3163	01-04-2020	1
5	4074	01-04-2020	1
6	3530	01-05-2020	1

TASK C.3

a. Simple Reports

VERSION-1:

REPORT-1_V1:

a)The query questions written in English

Top 5 total number of sales of property according to suburb and state.

b)Your explanation on why such a query is necessary or useful for the management

This query will allow the management to analyse total sales based on different suburbs and states for the future business expansion and even understanding needs based on locations.

c)The SQL commands

```
select * from

(select a.state_code, b.property_type, sum(d.total_sales) as Total,

dense_rank() over (order by sum(d.total_sales) desc) as sales_rank

from state2DIM a, Property2DIM b, sales2Fact d

where a.state_code = d.state_code

and b.property_id = d.property_id

Group by a.state_code,b.property_type)

where sales_rank <= 5;
```

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

	STATE_CODE	PROPERTY_TYPE	TOTAL	SALES_RANK
1	QLD	House	387470649	1
2	QLD	Apartment / Unit / Flat	122755750	2
3	VIC	House	108775450	3
4	NSW	House	76885650	4
5	SA	House	73319850	5

REPORT-2_V1 :

a)The query questions written in English

Top 10% properties visited in summer

b)Your explanation on why such a query is necessary or useful for the management

This query helps management to know the changes in parameters such as visits based on the time of the year. This data will be beneficial for managing the resources.

c)The SQL commands

```
Select * from

(select a.property_id, b.season_type, sum(c.Total_no_of_visit) as T_visit,

Percent_Rank() over (order by sum(c.Total_no_of_visit)) as Top_Percent

from Property2DIM a, seasonDIM b, Property2Fact c

Where a.property_id = c.property_id

and b.seasonID = c.SeasonID

and b.season_type = 'Summer'

group by a.property_id, b.season_type

order by Top_Percent desc)

Where Top_Percent >= 0.9;
```

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

	PROPERTY_ID	SEASON_TYPE	T_VISIT	TOP_PERCENT
1	1938	Summer	341222257760	1
2	1935	Summer	340122257760	0.994505494505494505494505494505494505494 5
3	1996	Summer	198121231461	0.989010989010989010989010989010989010989
4	5632	Summer	196321232925	0.983516483516483516483516483516483516483 5
5	1991	Summer	196321231461	0.972527472527472527472527472527472527472 5
6	1987	Summer	196321231461	0.972527472527472527472527472527472527472 5

REPORT-3_V1

a)The query questions written in English

Show all the rental properties in descending order based upon number of features type such as basic feature

b)Your explanation on why such a query is necessary or useful for the management

This query gives information of the effect of feature, scale (number of bedrooms) on the total rent of the property, which property has been rented the most can be analysed based on this information.

c)The SQL commands

```
Select a.property_id, b.scale_desc, c.feature_type, count(a.Total_no_of_rent),

dense_rank() over (order by count(a.Total_no_of_rent) desc ) as rent_rank

from rent2Fact a, property_scale2DIM b, feature2DIM c

where a.scale_id = b.scale_id

and a.feature_id = c.feature_id

group by a.property_id, b.scale_desc, c.feature_type;
```

d)The screenshots of the query results including all attribute names.

	PROPERTY_ID	SCALE_DESC	FEATURE_TYPE	COUNT(A.TOTAL_NO_OF_RENT)	RENT_RANK
1	3416	Small	Very Basic	1	1
2	3846	Extra Small	Very Basic	1	1
3	4751	Small	Very Basic	1	1
4	5164	Extra Small	Very Basic	1	1
5	5208	Small	Standard	1	1
6	6187	Extra Small	Standard	1	1

VERSION-2:

REPORT-1_V2

a)The query questions written in English

Top 5 total number of sales of property according to property type and state.

b)Your explanation on why such a query is necessary or useful for the management

This query will allow the management to analyse total sales based on different suburbs and states for the future business expansion and even understanding needs based on locations.

c)The SQL commands

```
select * from

(select a.state_code, b.property_type, sum(d.noofsale) as Total,

dense_rank() over (order by sum(d.noofsale) desc) as sales_rank

from state0DIM a, Property0DIM b, sales0Fact d

where a.state_code = d.state_code

and b.property_id = d.property_id

Group by a.state_code,b.property_type)

where sales_rank <= 5;
```

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

	STATE_CODE	PROPERTY_TYPE	TOTAL	SALES_RANK
1	WA	House	8944	1
2	TAS	House	8944	1
3	VIC	House	8944	1
4	SA	House	8944	1
5	NT	House	8944	1
6	NSW	House	8944	1

REPORT-2_V2

a) The query questions written in English

Top 10% properties visited by property type and visit date

b) Your explanation on why such a query is necessary or useful for the management

This query helps management to know the changes in parameters such as visits based on the time of the year. This data will be beneficial for managing the resources.

c) The SQL commands:

```
Select * from
(select c.visitdate, a.property_type, sum(c.Total_property_visited) as T_visit,
Percent_Rank() over (order by sum(c.Total_property_visited)) as Top_Percent
from Property0DIM a, Property0Fact c
Where a.property_id = c.property_id
group by c.visitdate, a.property_type
order by Top_Percent desc)
Where Top_Percent >= 0.9;
```

d)The screenshots of the query results including all attribute names.

	VISITDATE	PROPERTY_TYPE	T_VISIT	TOP_PERCENT
1	03-28-2020	Apartment / Unit / Flat	19	1
2	03-26-2020	Apartment / Unit / Flat	17	0.967741935483870967741935483870967741935 5
3	04-06-2020	Apartment / Unit / Flat	17	0.967741935483870967741935483870967741935 5
4	03-25-2020	Apartment / Unit / Flat	17	0.967741935483870967741935483870967741935 5
5	03-27-2020	House	16	0.946236559139784946236559139784946236559 1
6	03-26-2020	House	16	0.946236559139784946236559139784946236559 1
7	03-10-2020	House	14	0.935483870967741935483870967741935483871

REPORT-3_V2

a)The query questions written in English

Show all the properties with descending visits by date and property type

b)Your explanation on why such a query is necessary or useful for the management

This query gives information of the effect of feature, scale (number of bedrooms) on the total rent of the property, which property has been rented the most can be analysed based on this information.

c)The SQL commands

```
Select * from

(select c.visitdate, a.property_type, sum(c.Total_property_visited) as Total_visit,

dense_Rank() over (order by sum(c.Total_property_visited) desc ) as property_rank

from Property0DIM a, Property0Fact c

Where a.property_id = c.property_id

group by c.visitdate, a.property_type

order by property_rank );
```

d)The screenshots of the query results including all attribute names.

	VISITDATE	PROPERTY_TYPE	TOTAL_VISIT	PROPERTY_RANK
1	03-28-2020	Apartment / Unit / Flat	19	1
2	04-06-2020	Apartment / Unit / Flat	17	2
3	03-26-2020	Apartment / Unit / Flat	17	2
4	03-25-2020	Apartment / Unit / Flat	17	2
5	03-26-2020	House	16	3
6	03-27-2020	House	16	3

b. Reports with proper sub-totals:

Version -1:

REPORT-4_V1

(a) The questions in English

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

(b) Your explanation on why such a query is necessary or useful for the management

This query is useful to obtain information about rent prices for each property and even total rent price according to different suburbs , time and the type of the property.

(c) The SQL commands that include subtotals, using the Cube or Roll-up or Partial Cube/Roll-up operators

```
Select f.suburb, a.year, b.property_type, sum(a.no_of_price) as Total_rental_fees

from Rent2Fact a, property2DIM b, suburb2DIM f

where b.property_id = a.property_id

and f.postcode = a.postcode

group by cube (f.suburb,b.property_type,a.year);
```

(d) The screenshots of the query results.

	SUBURB	YEAR	PROPERTY_TYPE	TOTAL_RENTAL_FEES
46	Boya	(null)	Villa	620
47	Boya	2020	Villa	620
48	Boya	(null)	Apartment / Unit / Flat	480
49	Boya	(null)	Apartment / Unit / Flat	1040
50	Boya	2020	Apartment / Unit / Flat	560
51	City	(null)	(null)	658240
52	City	(null)	(null)	1730960
53	City	2019	(null)	42240
54	City	2020	(null)	1030480
55	City	(null)	House	44000
56	City	2020	House	44000
57	City	(null)	Apartment / Unit / Flat	658240
58	City	(null)	Apartment / Unit / Flat	1686960
59	City	2019	Apartment / Unit / Flat	42240
60	City	2020	Apartment / Unit / Flat	986480

REPORT-5_V1

(a) The questions in English

Show total of Rental price based upon Property type , time and year of different suburbs

(b) Your explanation on why such a query is necessary or useful for the management

This query gives the information about different suburbs and provides data with different combinations of time and year. We can analyse the similarity of data in different suburbs based on time and type of the property.

(c) The SQL commands that include subtotals, using the Cube or Roll-up or Partial Cube/Roll-up operators

```
Select f.suburb, a.year, b.property_type, sum(a.no_of_price) as Total_rental_fees

from Rent2Fact a, property2DIM b, suburb2DIM f

where b.property_id = a.property_id

and f.postcode = a.postcode

and b.property_type = 'House'

group by f.suburb, cube(a.year, b.property_type);
```

(d) The screenshots of the query results.

	SUBURB	YEAR	PROPERTY_TYPE	TOTAL_RENTAL_FEES
13	City	2020	(null)	44000
14	City	2020	House	44000
15	Cook	(null)	(null)	7200
16	Cook	(null)	(null)	17520
17	Cook	(null)	House	7200
18	Cook	(null)	House	17520
19	Cook	2019	(null)	3120
20	Cook	2019	House	3120
21	Cook	2020	(null)	7200
22	Cook	2020	House	7200
23	Holt	(null)	(null)	27720
24	Holt	(null)	(null)	52800

REPORT-6_V1

(a) The questions in English

Subtotals of Agents based upon office type and office name using Roll-up

(b) Your explanation on why such a query is necessary or useful for the management

This query gives information about the number of employees in different offices based on office type and name. This query will be useful for the management to know how much resources they have in different locations.

(c) The SQL commands that include subtotals, using the Cube or Roll-up or Partial Cube/Roll-up operators

```
Select a.office_name, b.office_type, Sum(c.total_no_agents) as Total_agents

from Office2DIM a, Office_type2DIM b, Agent2Fact c

where a.office_id = c.office_id

and b.office_typeid = c.officetype_id

group by rollup( a.office_name, b.office_type);
```

(d) The screenshots of the query results.

	OFFICE_NAME	OFFICE_TYPE	TOTAL_AGENTS
1	SFPG	Small	3
2	SFPG	(null)	3
3	Croll	Small	2
4	Croll	(null)	2
5	Hauss	Small	1
6	Hauss	(null)	1
7	Novak	Small	2
8	Novak	(null)	2
9	Pieta	Small	1
10	Pieta	(null)	1
11	BORIS.	Small	2
12	BORIS.	(null)	2
13	CIVIUM	Small	1

REPORT-7_V1

(a) The questions in English

Subtotals of sales price based upon featuretype and suburb using Partial Roll-up

(b) Your explanation on why such a query is necessary or useful for the management

This query can be useful for analysing the total profit earned from the rented properties based on features and suburbs.

(c) The SQL commands that include subtotals, using the Cube or Roll-up or Partial Cube/Roll-up operators

```
select a.feature_type, c.suburb, sum(no_of_price) as total_rent

from feature2DIM a, rent2Fact b, suburb2DIM c

where a.feature_id = b.feature_id

and b.postcode = c.postcode

group by rollup(a.feature_type, c.suburb);
```

(d) The screenshots of the query results.

	FEATURE_TYPE	SUBURB	TOTAL_RENT
1	Standard	Boya	1980
2	Standard	City	368720
3	Standard	Cook	3120
4	Standard	Holt	6480
5	Standard	Lota	4150
6	Standard	Page	1040
7	Standard	Reid	28240
8	Standard	Ryde	62400
9	Standard	Acton	25140
10	Standard	Ascot	312435
11	Standard	Bondi	119700
12	Standard	Bruce	64860

Version-2:

REPORT-4_V2

(a) The questions in English

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

(b) Your explanation on why such a query is necessary or useful for the management

This query is useful to obtain information about rent prices for each property and even total rent price according to different suburbs , time and the type of the property.

(c) The SQL commands that include subtotals, using the Cube or Roll-up or Partial Cube/Roll-up operators

```
Select f.suburb, a.rent_date , b.property_type, sum(a.no_of_price) as Total_rental_fees

from Rent0Fact a, property0DIM b, suburb0DIM f

where b.property_id = a.property_id

and f.address_id = a.address_id

group by cube (f.suburb,b.property_type,a.rent_date);
```

(d) The screenshots of the query results.

	SUBURB	RENT_DATE	PROPERTY_TYPE	TOTAL_RENTAL_FEES
1	Abbotsford	(null)	Apartment / Unit / Flat	3600
2	Abbotsford	(null)	Apartment / Unit / Flat	3600
3	Abbotsford	(null)	(null)	3600
4	Abbotsford	(null)	(null)	3600
5	Acton	22-04-20	Apartment / Unit / Flat	1580
6	Acton	(null)	Apartment / Unit / Flat	920
7	Acton	(null)	Apartment / Unit / Flat	2500
8	Acton	22-04-20	(null)	1580
9	Acton	(null)	(null)	920
10	Acton	(null)	(null)	2500
11	Adelaide	27-01-20	Apartment / Unit / Flat	2040
12	Adelaide	31-03-20	Apartment / Unit / Flat	1290
13	Adelaide	04-04-20	Apartment / Unit / Flat	750
14	Adelaide	08-04-20	Apartment / Unit / Flat	3840

REPORT-5_V2

(a) The questions in English

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

(b) Your explanation on why such a query is necessary or useful for the management

This query gives the information about different suburbs and provides data with different combinations of time and year. We can analyse the similarity of data in different suburbs based on time and type of the property.

(c) The SQL commands that include subtotals, using the Cube or Roll-up or Partial Cube/Roll-up operators

```
Select f.suburb, a.rent_date , b.property_type, sum(a.no_of_price) as Total_rental_fees

from Rent0Fact a, property0DIM b, suburb0DIM f

where b.property_id = a.property_id

and f.address_id = a.address_id

group by f.suburb ,cube (b.property_type,a.rent_date);
```

(d) The screenshots of the query results.

	SUBURB	RENT_DATE	PROPERTY_TYPE	TOTAL_RENTAL_FEES
28	City	(null)	Apartment / Unit / Flat	48115
29	City	(null)	Apartment / Unit / Flat	106300
30	City	30-12-19	Apartment / Unit / Flat	480
31	City	14-01-20	Apartment / Unit / Flat	500
32	City	28-01-20	Apartment / Unit / Flat	2250
33	City	06-02-20	Apartment / Unit / Flat	1800
34	City	12-02-20	Apartment / Unit / Flat	1750
35	City	14-02-20	Apartment / Unit / Flat	9100
36	City	11-03-20	Apartment / Unit / Flat	7000
37	City	19-03-20	Apartment / Unit / Flat	4000
38	City	20-03-20	Apartment / Unit / Flat	5850
39	City	23-03-20	Apartment / Unit / Flat	1300
40	City	24-03-20	Apartment / Unit / Flat	465
41	City	29-03-20	Apartment / Unit / Flat	2400

REPORT-6_V2

(a) The questions in English

what are total and Subtotals of number of employees from each office and each gender using Roll-up

(b) Your explanation on why such a query is necessary or useful for the management

This query is important for the management to know about their resources in different offices and even the gender ratio in the office can be calculated.

(c) The SQL commands that include subtotals, using the Cube or Roll-up or Partial Cube/Roll-up operators

```
Select a.office_name, b.gender, Sum(c.no_of_employee) as Total_agents

from Office0DIM a, agent0dim b,Agent0Fact c

where a.office_id = c.office_id

and b.agent_id = c.agent_id

group by rollup( a.office_name, b.gender);
```

(d) The screenshots of the query results.

	OFFICE_NAME	GENDER	TOTAL_AGENTS
1	SFPG	Male	2
2	SFPG	Female	1
3	SFPG	(null)	3
4	Croll	Male	1
5	Croll	Female	1
6	Croll	(null)	2
7	Hauss	Male	1
8	Hauss	(null)	1
9	Novak	Male	1
10	Novak	Female	1
11	Novak	(null)	2
12	Pieta	Female	1
13	Pieta	(null)	1
14	BORIS.	Female	2

REPORT-7_V2

(a) The questions in English

what are total and Subtotals of number of employees from each office and each gender using partial rollup

(b) Your explanation on why such a query is necessary or useful for the management

This query can be useful to know the similarity or differences for different offices in terms of gender and salary.

(c) The SQL commands that include subtotals, using the Cube or Roll-up or Partial Cube/Roll-up operators

```
Select a.office_name, b.gender, b.salary,Sum(c.no_of_employee) as Total_agents

from Office0DIM a, agent0dim b,Agent0Fact c

where a.office_id = c.office_id

and b.agent_id = c.agent_id

group by a.office_name, rollup(b.gender,b.salary);
```

(d) The screenshots of the query results.

	OFFICE_NAME	GENDER	SALARY	TOTAL_AGENTS
1	SFPG	Male	210000	1
2	SFPG	Male	175000	1
3	SFPG	Male	(null)	2
4	SFPG	Female	190000	1
5	SFPG	Female	(null)	1
6	SFPG	(null)	(null)	3
7	Croll	Male	210000	1
8	Croll	Male	(null)	1
9	Croll	Female	195000	1
10	Croll	Female	(null)	1
11	Croll	(null)	(null)	2
12	Hauss	Male	200000	1
13	Hauss	Male	(null)	1
14	Hauss	(null)	(null)	1

c. Reports with moving and cumulative aggregates:

Version -1:

REPORT-8_V1

(a) The query questions

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

(b) explanation on why such a query is necessary

This query gives information about the increase in total number of clients over the years with a high budget to know the most profitable clients.

(c) The SQL commands that contains moving and cumulative aggregates

```
select a.year,
to_char(sum(total_no_of_clients),'9,999,999,999') as client_total,
to_char(sum(sum(total_no_of_clients)) over
(order by a.year rows unbounded preceding),'9,999,999,999') as cumulative_client_total
from client2fact a, budget2DIM c
where a.budgetID = c.budgetID
and a.budgetid = '3'
group by a.year;
```

(d) The screenshots of the query results

	YEAR	CLIENT_TOTAL	CUMULATIVE_CLIENT_TOTAL
1	2019	23	23
2	2020	891	914

REPORT-9_V1

(a) The query questions

Moving aggregate reports: show total rent based on scale description for every year.

(b) explanation on why such a query is necessary

This particular query can provide information about the count of properties on rent and total increase in average count of properties which can provide information about profit on average from properties on rent.

(c) The SQL commands that contains moving and cumulative aggregates

```
SELECT a.year,scale_desc,

TO_CHAR (SUM(a.total_no_of_rent), '9,999,999,999') AS R_rent,

TO_CHAR (AVG(SUM(a.total_no_of_rent)) OVER

(ORDER BY a.year

ROWS 2 PRECEDING),'9,999,999,999') AS MOVING_2_MONTH_AVG

FROM rent2fact a, property_scale2DIM b

WHERE a.scale_id = b.scale_id

and b.scale_desc = 'Small'

GROUP BY a.year,b.scale_desc;
```

(d) The screenshots of the query results

	YEAR	SCALE_DESC	R_RENT	MOVING_2_MONTH_AVG
1	2019	Small	9	9
2	2020	Small	695	352
3	(null)	Small	728	477

REPORT-10_V1

(a) The query questions

REPORT 10: Cumulative aggregate reports for total rent over the years.

(b) explanation on why such a query is necessary

This query gives information about the total increase in rent over the years for properties of large scale which is the indication of profit.

(c) The SQL commands that contains moving and cumulative aggregates

```
select a.year,
to_char(SUM(total_no_of_rent),'9,999,999,999') as rent_total,
to_char(sum(sum(total_no_of_rent)) over
(order by a.year rows unbounded preceding),'9,999,999,999') as cumulative_rent_total
from rent2fact a,property_scale2DIM b
WHERE a.scale_id = b.scale_id
and b.scale_desc = 'Large'
group by a.year;
```

(d) The screenshots of the query results

	YEAR	RENT_TOTAL	CUMULATIVE_RENT_TOTAL
1	2020	1	1
2	(null)	2	3

Version -2:

REPORT-8_V2

(a) The query questions

the total number of clients and cumulative number of clients with a high budget in each date

(b) explanation on why such a query is necessary

This query gives information about the increase in total number of clients over each day with a high budget to know the most profitable clients.

(c) The SQL commands that contains moving and cumulative aggregates

```
select a.r_date,
to_char(sum(total_no_client),'9,999,999,999') as client_total,
to_char(sum(sum(total_no_client)) over
(order by a.r_date rows unbounded preceding),'9,999,999,999') as cumulative_client_total
from client0fact a, client_budgetdim c
where a.client_id = c.client_id
and c.min_budget > 100001
and  c.max_budget < 10000000
group by a.r_date;
```

(d) The screenshots of the query results

	R_DATE	CLIENT_TOTAL	CUMULATIVE_CLIENT_TOTAL
1	01-01-2020	8	8
2	01-02-2020	8	16
3	01-03-2020	8	24
4	01-04-2020	13	37
5	02-01-2020	11	48
6	02-02-2020	16	64
7	02-03-2020	8	72
8	02-04-2020	11	83
9	03-01-2020	10	93
10	03-02-2020	4	97
11	03-03-2020	11	108
12	03-04-2020	11	119

REPORT-9_V2

(a) The query questions

Moving aggregate reports : number of clients in each day for low budget

(b) explanation on why such a query is necessary

This query can be useful to know about the profit organisation is making on clients on low budget.

(c) The SQL commands that contains moving and cumulative aggregates

```
select a.r_date,
to_char(sum(total_no_client),'9,999,999,999') as client_total,
to_char(avg(sum(total_no_client)) over
(order by a.r_date rows 2 preceding),'9,999,999,999') as cumulative_client_total
from client0fact a, client_budgetdim c
where a.client_id = c.client_id
and c.min_budget > 0
and  c.max_budget < 1000
group by a.r_date;
```

(d) The screenshots of the query results

	R_DATE	CLIENT_TOTAL	CUMULATIVE_CLIENT_TOTAL
1	01-01-2020	9	9
2	01-02-2020	13	11
3	01-03-2020	5	9
4	01-04-2020	11	10
5	01-05-2020	36	17
6	02-01-2020	8	18
7	02-02-2020	11	18
8	02-03-2020	9	9
9	02-04-2020	7	9
10	02-05-2020	28	15
11	03-01-2020	9	15
12	03-02-2020	9	15
13	03-03-2020	7	8

REPORT-10_V2

(a) The query questions

Cumulative aggregate reports : Total rent and cumulative rent in each day

(b) explanation on why such a query is necessary

This query gives information about the total increase in rent over the years for properties of large scale which is the indication of profit.

(c) The SQL commands that contains moving and cumulative aggregates

```
select a.rent_date,
to_char(SUM(total_no_of_rent),'9,999,999,999') as rent_total,
to_char(sum(sum(total_no_of_rent)) over
(order by a.rent_date rows unbounded preceding),'9,999,999,999') as cumulative_rent_total
from rent0fact a
group by a.rent_date;
```

(d) The screenshots of the query results

	RENT_DATE	RENT_TOTAL	CUMULATIVE_RENT_TOTAL
1	30-12-19	4	4
2	30-12-19	1	5
3	30-12-19	10	15
4	30-12-19	16	31
5	30-12-19	1	32
6	30-12-19	1	33
7	30-12-19	6	39
8	30-12-19	1	40
9	30-12-19	2	42
10	30-12-19	6	48
11	31-12-19	10	58
12	31-12-19	14	72

d. Reports with Partitions:

Version-1:

REPORT-11_V1

(a) The query questions

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.

(b) explanation

This query can be useful for having information about the top ranking states over the years.

(c) The SQL commands

```
SELECT a.property_type, b.state_code,
 SUM(b.total_sales)as Total,
 RANK() OVER (PARTITION BY a.property_type
 ORDER BY SUM(b.total_sales)DESC) AS RANK_of_property,
 RANK() OVER (PARTITION BY b.state_code
 ORDER BY SUM(b.total_sales)DESC) AS RANK_of_state
FROM property2DIM a, sales2Fact b, state2DIM c
WHERE b.state_code = c.state_code
GROUP BY a.property_type, b.state_code;
```

(d) The screenshots of the query results

	PROPERTY_TYPE	STATE_CODE	TOTAL	RANK_OF_PROPERTY	RANK_OF_STATE
1	Apartment / Unit / Flat	QLD	1501669572123	1	2
2	Apartment / Unit / Flat	VIC	379093717800	2	2
3	Apartment / Unit / Flat	NSW	378502336376	3	2
4	Apartment / Unit / Flat	ACT	331772656500	4	2
5	Apartment / Unit / Flat	SA	231045980600	5	2
6	Apartment / Unit / Flat	WA	191022689000	6	2
7	Apartment / Unit / Flat	TAS	18805925000	7	2
8	Apartment / Unit / Flat	NT	1860515000	8	2
9	Block of Units	QLD	3926666793	1	11
10	Block of Units	VIC	991279800	2	11
11	Block of Units	NSW	989733416	3	11
12	Block of Units	ACT	867541500	4	11
13	Block of Units	SA	604154600	5	11
14	Block of Units	WA	499499000	6	11

REPORT-12_V1

(a) The query questions

Ranking of each suburb and scale based on no of rent

(b) explanation

This query gives information about the top ranking suburbs and scales based count of properties on rent.

(c) The SQL commands

```
SELECT a.year, b.suburb,
 SUM(a.total_no_of_rent)as Total,
 dense_RANK() OVER (PARTITION BY b.suburb
 ORDER BY SUM(a.total_no_of_rent)DESC) AS RANK_of_suburb,
 dense_RANK() OVER (PARTITION BY a.year
 ORDER BY SUM(a.total_no_of_rent)DESC) AS RANK_of_year
FROM rent2fact a, suburb2DIM b
where a.postcode = b.postcode
GROUP BY a.year, b.suburb;
```

(d) The screenshots of the query results

	YEAR	SUBURB	TOTAL	RANK_OF_SUBURB	RANK_OF_YEAR
1	(null)	Abbotsford	32	1	124
2	2020	Acton	114	1	81
3	(null)	Acton	84	2	95
4	2019	Acton	6	3	21
5	2020	Adelaide	1056	1	13
6	(null)	Adelaide	1056	1	9
7	2020	Ainslie	132	1	76
8	(null)	Ainslie	90	2	93
9	(null)	Aitkenvale	70	1	102
10	2020	Aitkenvale	70	1	101
11	(null)	Albanvale	2	1	150
12	2020	Albany Creek	52	1	109
13	(null)	Albany Creek	13	2	139
14	(null)	Albert Park	100	1	89

Version-2:

REPORT-11_V2

(a) The query questions

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.

(b) explanation

This query can be useful for having information about the top ranking states over the years.

(c) The SQL commands

```
SELECT a.property_type, b.state_code,
 SUM(b.noofsale)as Total,
 RANK() OVER (PARTITION BY a.property_type
 ORDER BY SUM(b.noofsale)DESC) AS RANK_of_property,
 RANK() OVER (PARTITION BY b.state_code
 ORDER BY SUM(b.noofsale)DESC) AS RANK_of_state
FROM property0DIM a, sales0Fact b, state0DIM c
WHERE b.state_code = c.state_code
and a.property_id = b.property_id
GROUP BY a.property_type, b.state_code;
```

(d) The screenshots of the query results

	PROPERTY_TYPE	STATE_CODE	TOTAL	RANK_OF_PROPERTY	RANK_OF_STATE
1	Apartment / Unit / Flat	TAS	3361	1	2
2	Apartment / Unit / Flat	VIC	3361	1	2
3	Apartment / Unit / Flat	QLD	3361	1	2
4	Apartment / Unit / Flat	WA	3361	1	2
5	Apartment / Unit / Flat	SA	3361	1	2
6	Apartment / Unit / Flat	NSW	3361	1	2
7	Apartment / Unit / Flat	NT	3361	1	2
8	Apartment / Unit / Flat	ACT	3361	1	2
9	Block of Units	QLD	19	1	10
10	Block of Units	ACT	19	1	10
11	Block of Units	NT	19	1	10
12	Block of Units	VIC	19	1	10
13	Block of Units	NSW	19	1	10

REPORT-12_V2

(a) The query questions

Ranking of each suburb and feature based on yearly no of rent

(b) explanation

This query gives information about the top ranking suburbs and scales based count of properties on rent.

(c) The SQL commands

```
SELECT c.feature_description, b.suburb,
 SUM(a.total_no_of_rent)as Total,
 dense_RANK() OVER (PARTITION BY b.suburb
 ORDER BY SUM(a.total_no_of_rent)DESC) AS RANK_of_suburb,
 dense_RANK() OVER (PARTITION BY c.feature_description
 ORDER BY SUM(a.total_no_of_rent)DESC) AS RANK_of_feature
FROM rent0fact a, suburb0DIM b, feature0DIM c
WHERE a.address_id = b.address_id
and a.feature_code = c.feature_code
GROUP BY c.feature_description, b.suburb;
```

(d) The screenshots of the query results

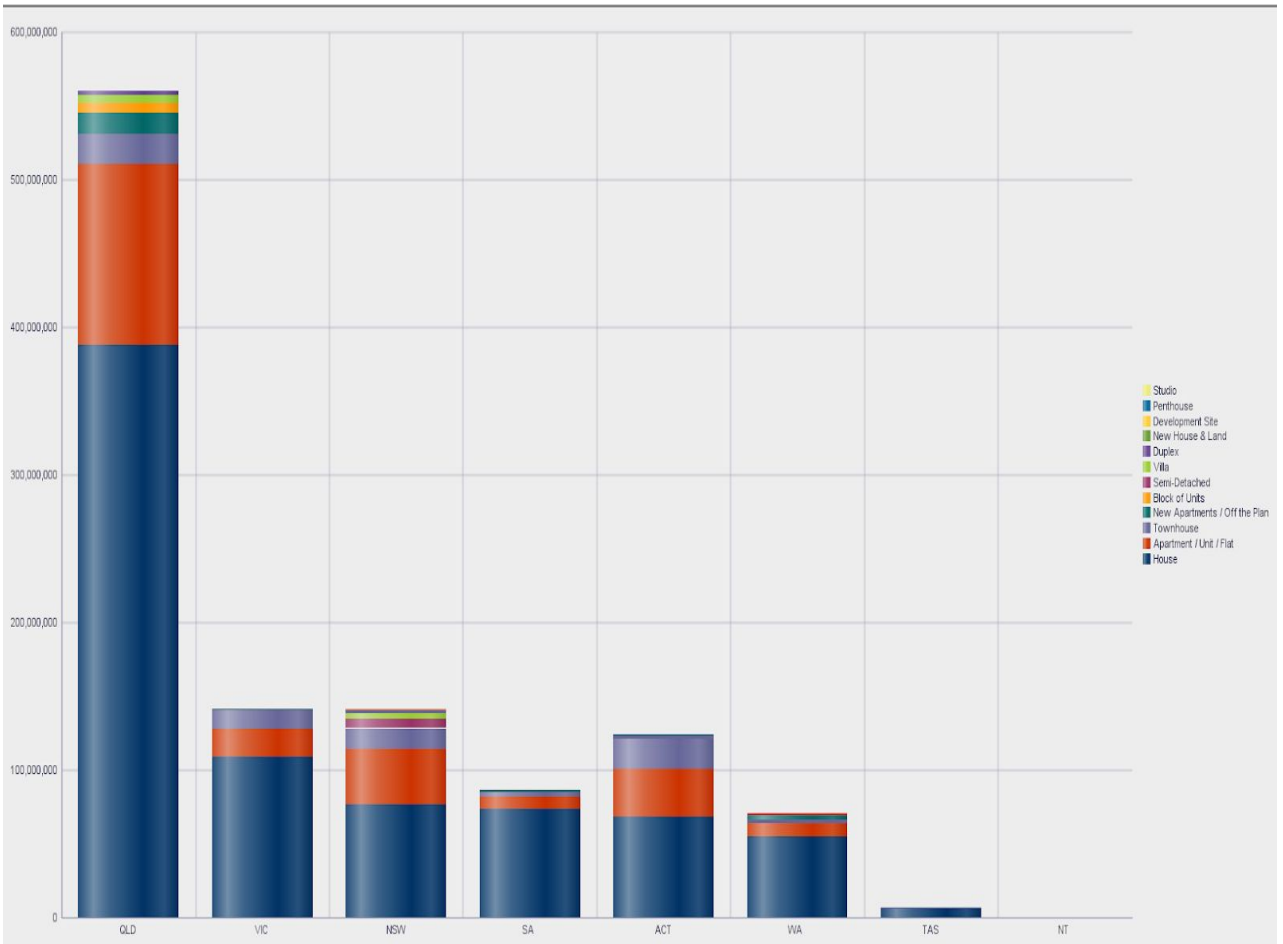
	FEATURE_DESCRIPTION	SUBURB	TOTAL	RANK_OF_SUBURB	RANK_OF_FEATURE
1	Swimming Pool	Abbotsford	1	1	10
2	Security	Abbotsford	1	1	5
3	Security Access	Abbotsford	1	1	4
4	Heating	Abbotsford	1	1	10
5	Air conditioning	Abbotsford	1	1	15
6	Close to shops	Abbotsford	1	1	14
7	Close to transport	Abbotsford	1	1	13
8	Built in wardrobes	Abbotsford	1	1	16
9	Intercom	Abbotsford	1	1	11
10	Secure Parking	Acton	1	1	15
11	Car Accom: Double Garage	Acton	1	1	1
12	Furnished	Acton	1	1	9
13	Double Garage	Acton	1	1	1
14	Dishwasher	Adelaide	11	1	2

Task C.4

Business Intelligence (BI) Reports.

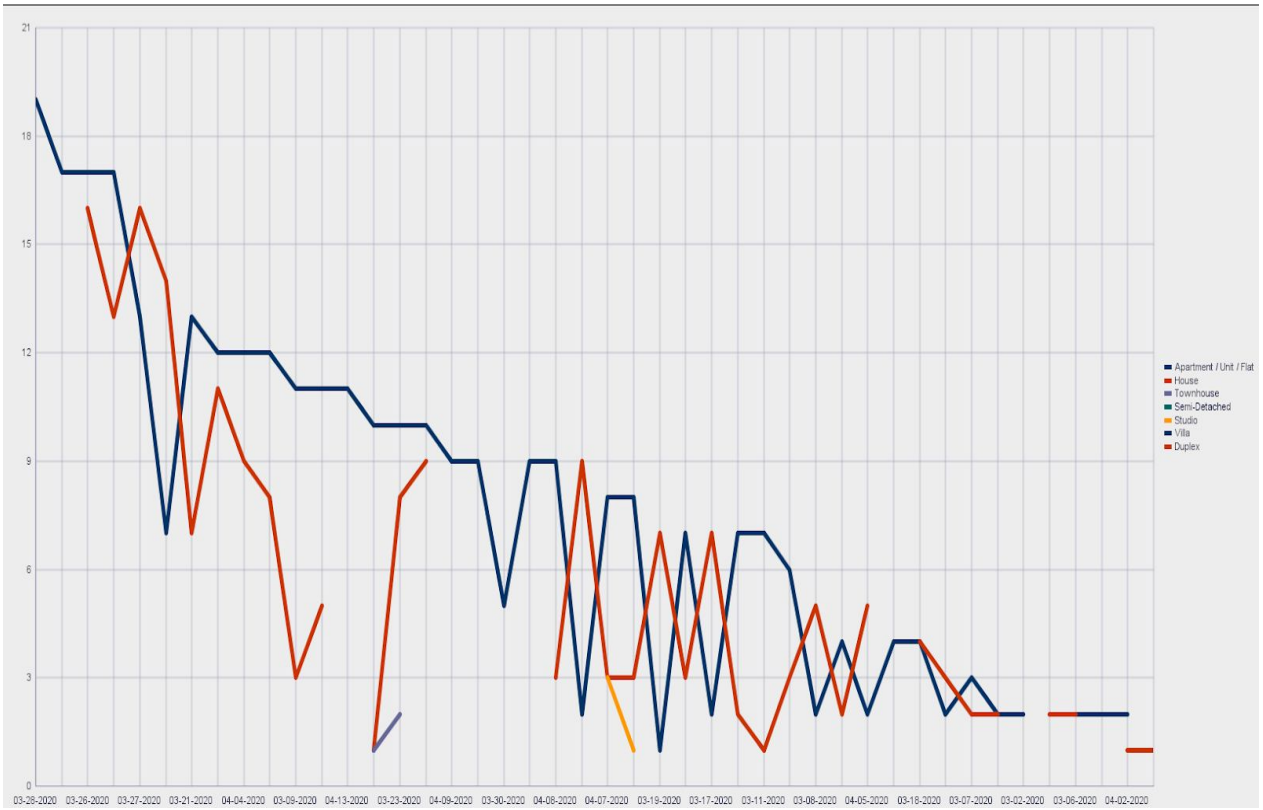
BI_REPORT_1:

top 50 properties by suburb and state



BI_REPORT_2:

Total number of property visited based upon property type and visit date

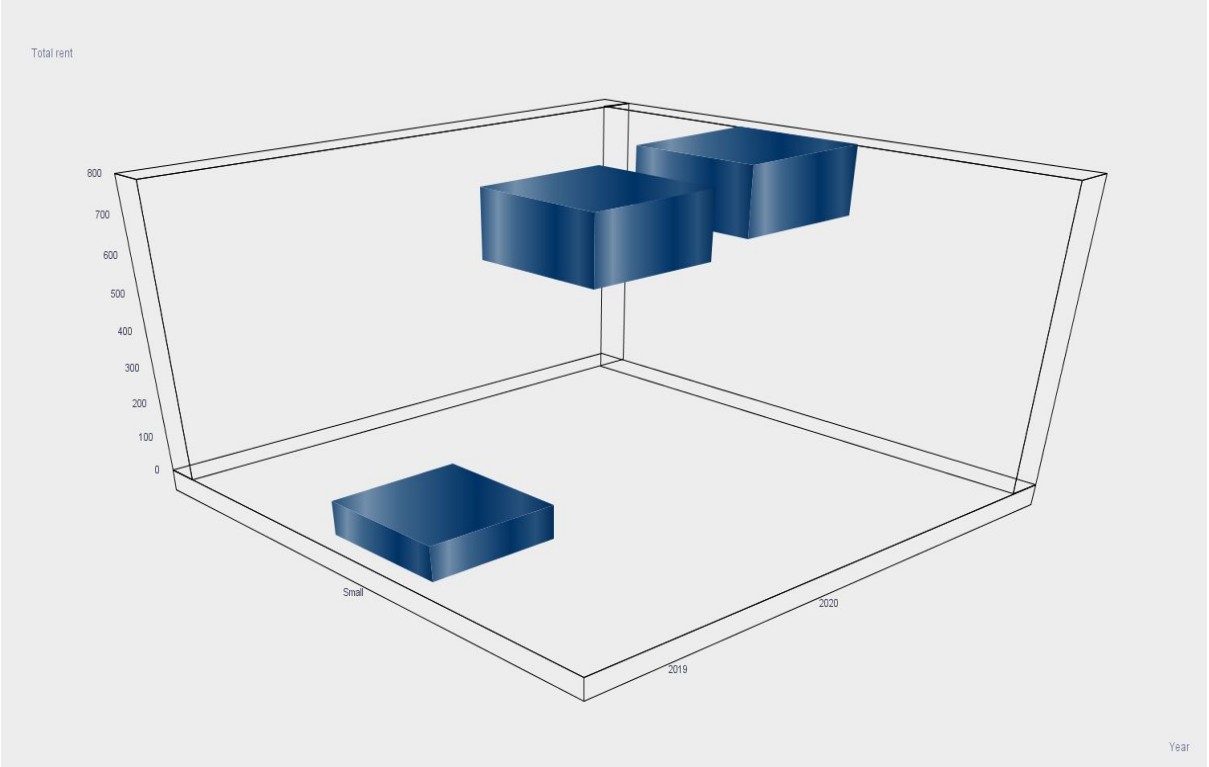


BI_REPORT_3:

Total number of employees based upon office name and gender

BI_REPORT_4:

Total Number of Rent over the period of years



BI_REPORT_5:

Total sales based upon state and property type