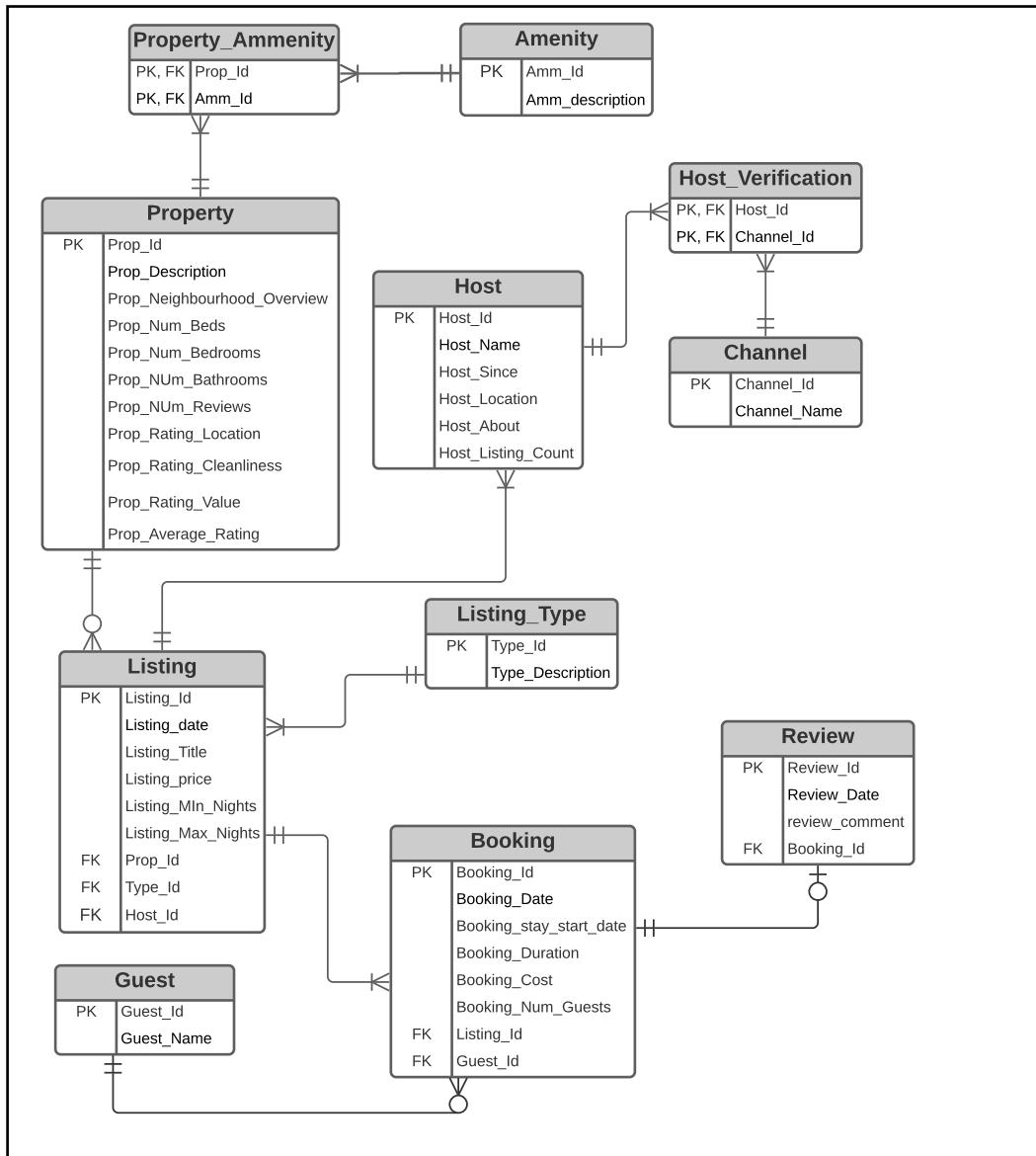


## ER Diagram of the Operational database



## Exploring data

```
---- Exploring the data--  
  
SELECT * FROM Mstay.Review;  
SELECT * FROM Mstay.booking;  
SELECT * FROM Mstay.Guest;  
SELECT * FROM Mstay.Listing;  
SELECT * FROM MSTAY.HOST;  
SELECT * FROM MSTAY.host_verification;  
SELECT * FROM Mstay.channel;  
SELECT * FROM Mstay.Listing_type;  
SELECT * FROM Mstay.Property;  
SELECT * FROM Mstay.property_Ammenity;  
SELECT * FROM MStay.Amenity;
```

## Importing all the tables

```
----importing all the tables----  
  
CREATE Table Review AS SELECT * FROM Mstay.Review;  
CREATE Table Booking AS SELECT * FROM Mstay.booking;  
CREATE TABLE Guest AS SELECT * FROM Mstay.Guest;  
CREATE TABLE Listing AS SELECT * FROM Mstay.Listing;  
CREATE TABLE Host AS SELECT * FROM MSTAY.HOST;  
CREATE TABLE Host_Verification AS SELECT * FROM MSTAY.host_verification;  
CREATE TABLE Channel AS SELECT * FROM Mstay.channel;  
CREATE TABLE Listing_type AS SELECT * FROM Mstay.Listing_type;  
CREATE TABLE Property AS SELECT * FROM Mstay.Property;  
CREATE TABLE Property_Amenity AS SELECT * FROM Mstay.property_Ammenity;  
CREATE TABLE Amenity AS SELECT * FROM MStay.Amenity;
```

## Checking Number of records in each table

```
-- Number of records in each table--  
  
SELECT COUNT (*) FROM Review;  
--- Review COUNT = 4870  
  
SELECT COUNT (*) FROM booking;  
---Booking Count = 5002  
  
SELECT COUNT (*) FROM Guest;  
---Guest Count = 9372  
  
SELECT COUNT (*) FROM Listing;  
---Listing Count = 4936  
  
SELECT COUNT (*) FROM HOST;  
---Host Count = 3883  
  
SELECT COUNT (*) FROM host_verification;  
---Host_Verification Count = 21749  
  
SELECT COUNT (*) FROM channel;  
---Channel COUNT = 20  
  
SELECT COUNT (*) FROM Listing_type;  
---Listing_type Count = 4  
  
SELECT COUNT (*) FROM Property;  
---Property Count = 5001  
  
SELECT COUNT (*) FROM property_Amenity;  
---Property_Amenity Count = 47027  
  
SELECT COUNT (*) FROM Amenity;  
---Amenity Count = 449
```

**While exploring the channel none value was found, lets look into it. So decided to remove that value.**

```
--- While exploring the channel I found a value as none, lets look into it--  
  
SELECT Channel_Id, channel_name FROM Channel  
WHERE Channel_Name = 'None';  
  
--- Lets remove this record----  
  
DELETE FROM Channel  
WHERE Channel_id = 16;  
  
SELECT COUNT(*) FROM channel;  
--- The previous count was 20 and now the count is 19.  
---so the row with none value is deleted
```

Query Result x

SQL | All Rows Fetched: 1 in 0.946 seconds

COUNT(*)
1 19

**Duplicate Values in the Host and the Booking table. To ensure that there are no duplicates, while creating fact and dimension tables distinct will be used.**

```
--Check for duplicate values--  
  
SELECT host_since, host_id,  
COUNT(*) As duplicate_records  
FROM host  
Group by host_since, host_id  
Having COUNT(*) > 1;
```

Script Output x | Query Result x  
SQL | All Rows Fetched: 1 in 8.269 seconds

HOST_SINCE	HOST_ID	DUPLICATE_RECORDS
22/JUN/13	7046664	4

```
--Check for duplicate values--  
  
SELECT host_since, host_id,  
COUNT(*) As duplicate_records  
FROM host  
Group by host_since, host_id  
Having COUNT(*) > 1;  
  
---Lets look into it---  
SELECT *  
FROM Host  
WHERE HOST_ID = 7046664;
```

Script Output x | Query Result x  
SQL | All Rows Fetched: 4 in 2.945 seconds

HOST_ID	HOST_NAME	HOST_SINCE	HOST_LOCATION	HOST_ABOUT
1	7046664 Dave	22/JUN/13	Rosebud, Victoria, Australia	Living the dream on the Mornington Pe
2	7046664 Dave	22/JUN/13	Rosebud, Victoria, Australia	Living the dream on the Mornington Pe
3	7046664 Dave	22/JUN/13	Rosebud, Victoria, Australia	Living the dream on the Mornington Pe
4	7046664 Dave	22/JUN/13	Rosebud, Victoria, Australia	Living the dream on the Mornington Pe

```
SELECT booking_id, booking_date, listing_id, Guest_id,  
COUNT(*) As duplicate_records  
FROM booking  
Group By booking_id, booking_date, listing_id, guest_id  
Having COUNT (*) >1;
```

Script Output x | Query Result x  
SQL | All Rows Fetched: 1 in 0.154 seconds

BOOKING_ID	BOOKING_DATE	LISTING_ID	GUEST_ID	DUPLICATE_RECORDS
1	537 11/MAY/15	530	17812230	2

```
SELECT booking_id, booking_date, listing_id, Guest_id,  
COUNT(*) As duplicate_records  
FROM booking  
Group By booking_id, booking_date, listing_id, guest_id  
Having COUNT (*) >1;
```

Script Output x | Query Result x  
SQL | All Rows Fetched: 1 in 0.154 seconds

BOOKING_ID	BOOKING_DATE	LISTING_ID	GUEST_ID	DUPLICATE_RECORDS
1	537 11/MAY/15	530	17812230	2

**After Checking If there are any records in Listing but not in booking. We got a record in listing with negative listing price and not in Booking Id. So decided to delete it.**

```
--Checking If there are any records in Listing but not in booking.  
-----  
SELECT *  
FROM Listing  
WHERE Listing_id NOT IN (SELECT Listing_id FROM Booking);
```

Script Output x Query Result x  
SQL | All Rows Fetched: 1 in 0.09 seconds

	LISTING_ID	LISTING_DATE	LISTING_TITLE	LISTING_PRICE	LISTING_MIN_NIGHTS	LISTING_MAX_NIGHTS	PROP_ID	TYPE_ID	HOST_ID
1	99999	18/DEC/18	Melbourne accomodation	-150	1	7	9999	2	9999

```
--Checking If there are any records in Listing but not in booking.  
-----  
SELECT *  
FROM Listing  
WHERE Listing_id NOT IN ( SELECT Booking_id FROM Booking);  
  
-- The listing Price is in negative for the above data. So let's delet it.  
  
DELETE FROM Listing  
Where Listing_Id = 99999;  
  
SELECT COUNT(*) FROM LISTING;
```

Script Output x Query Result x  
SQL | All Rows Fetched: 1 in 2.25 seconds

	COUNT(*)
1	4935

## Checking If there are any records in booking but not in guest.

Worksheet | Query Builder

```
WHERE Listing_id NOT IN (SELECT Listing_id FROM Booking);
-- The listing Price is in negative for the above data. So let's delete it.

DELETE FROM Listing
Where Listing_Id = 99999;

SELECT COUNT(*) FROM LISTING;

---Checking If there are any records in Booking but not in Guest.

SELECT COUNT(*) FROM Booking
WHERE Guest_Id NOT IN (SELECT Guest_Id FROM Guest);
```

Script Output x | Query Result x

SQL All Rows Fetched: 1 in 0.928 seconds

COUNT(*)
1 0

## Checking for Null values

```
----Checking for null values

SELECT * FROM Listing
WHERE (Listing_Id IS NULL OR
Listing_date IS NULL OR
Listing_Price IS NULL OR
Listing_Max_Nights IS NULL);

SELECT * FROM Booking
WHERE (Booking_Id IS NULL OR
Booking_date IS NULL OR
Booking_Cost IS NULL OR
Booking_duration IS NULL);

SELECT Prop_Id, Prop_Description, Prop_Num_Bedrooms, Prop_Num_Bathrooms
FROM Property
WHERE (Prop_Id IS NULL OR
Prop_Description IS NULL OR
Prop_Num_Bedrooms IS NULL OR
Prop_Num_Bathrooms IS NULL);

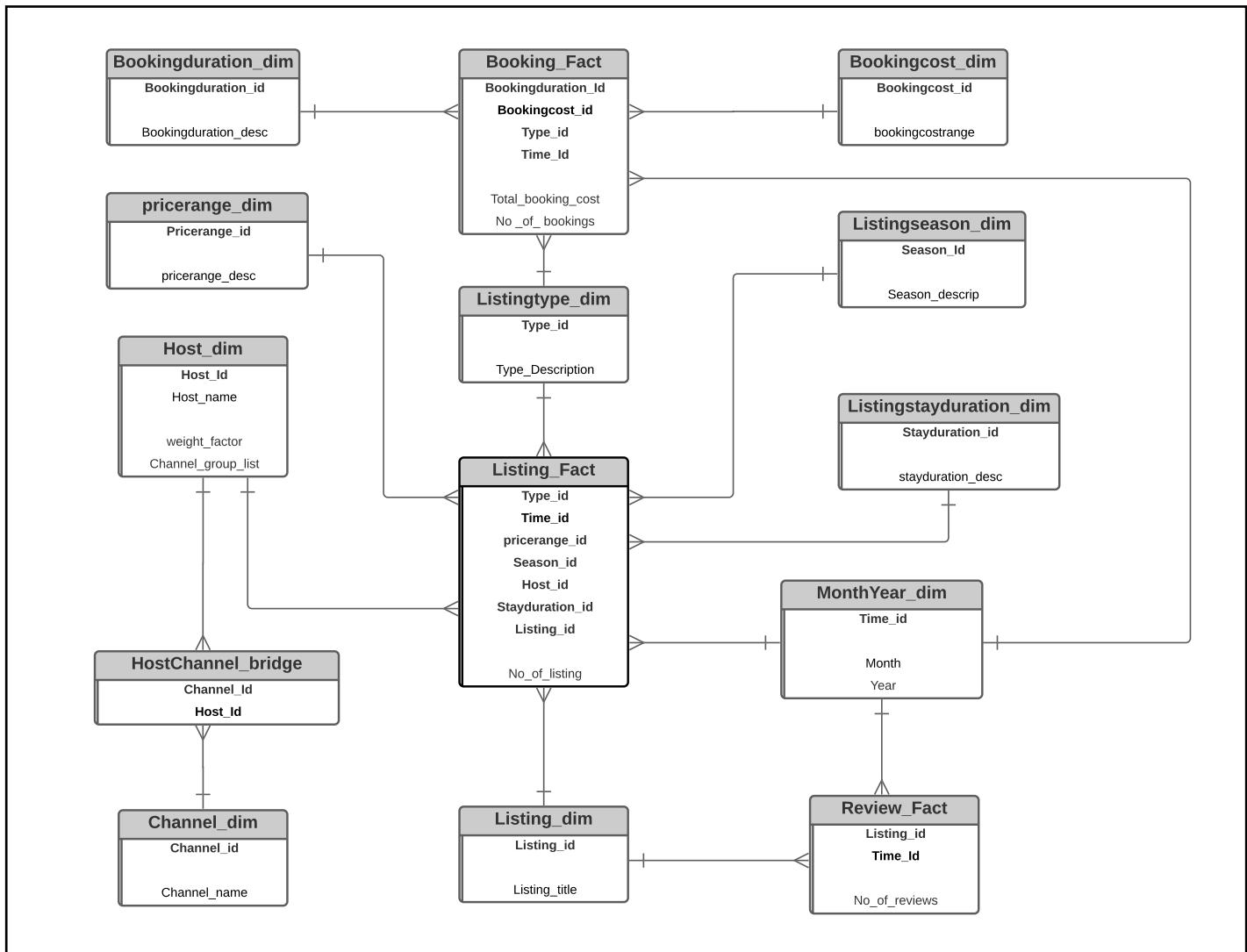
----Lots of properties have null values for prop_num_bedrooms,
----prop_description and prop_Num_Bathrooms

SELECT * FROM HOST
WHERE (Host_Id IS NULL OR
HOST_Name IS NULL OR
HOST_SINCE IS NULL OR
HOST_LOCATION IS NULL OR
HOST_ABOUT IS NULL OR
HOST_LISTING_COUNT IS NULL);

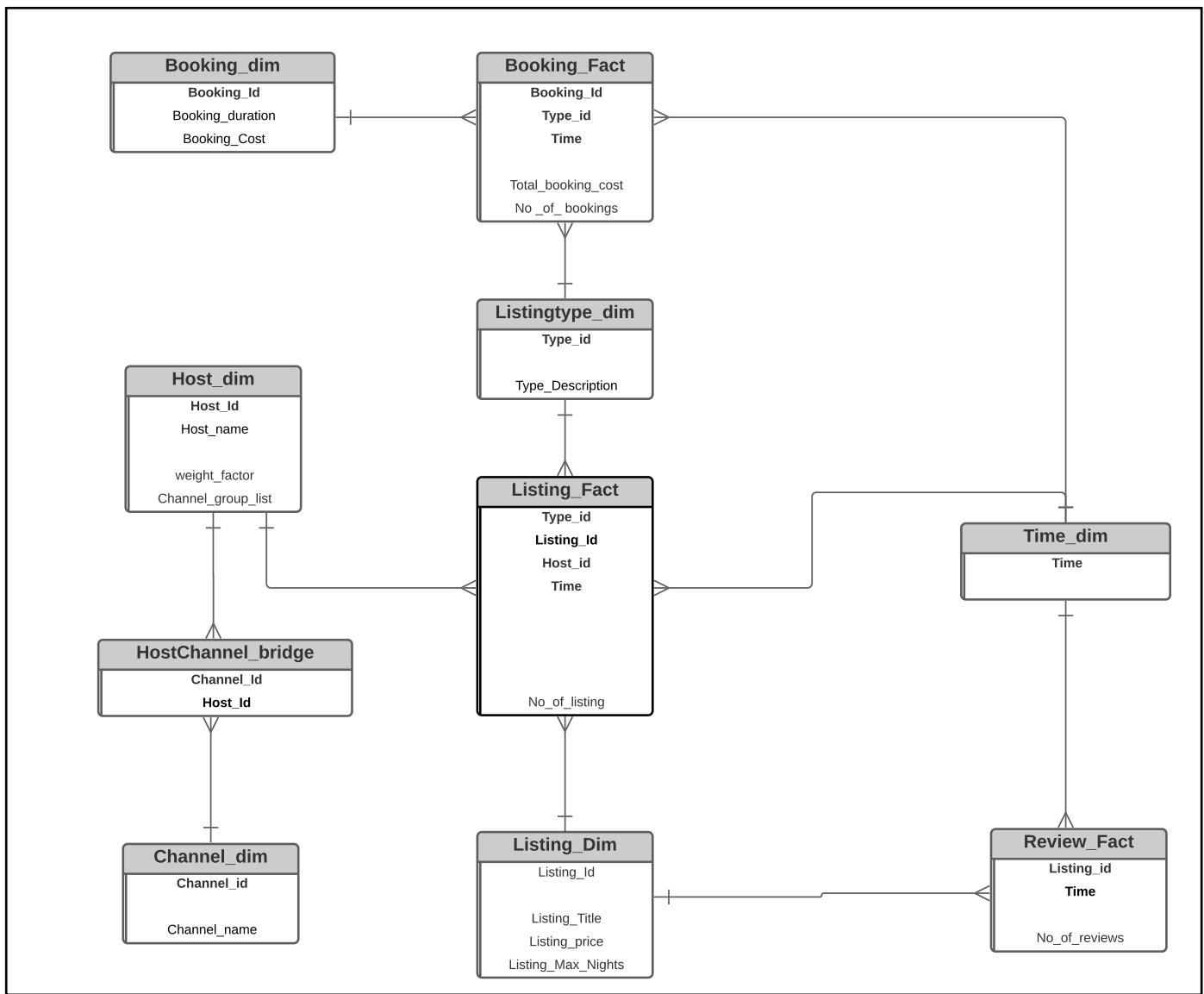
--HOST ABOUT has lot of null values
```

We will not replace the null values at this stage as we are not going to use this attributes in our dimension and fact tables. It may also be due to unknown number of beds in property example or in host it could be that host do not want to add something about himself.

## Version-1 [Level 2]



## Version-2 [Level 0]



## **A short explanation of the difference between the two versions of star/ snowflake Schema.**

### **Version-1 [Level 2] :**

The above schema is highly aggregate. Where We have Multi fact tables with multiple Fact measures as No. of bookings, Total booking cost, No of reviews, No of listing.  
And the dimension are based on Season, Stay duration for listing and booking, Month and Year and not actual Time period. It also has Price range and booking cost range as its dimension and not actual listing price and booking cost.

### **Version-2 [Level 0] :**

The above schema is with No aggregation and higher granularity. The season, booking duration, listing duration and Month year has been changed to Time where property has listed, booked and reviewed.  
The aggregated dimensions for listing and booking have been changed to Listing dimension and Booking dimension with Listing price, Listing max nights and Booking duration and booking cost.

## **Creating Dimension Tables.**

**Listing dimension , Listing\_type dimension, Host\_Channel\_Bridge, Channel dimension.**

```
CREATE TABLE Listing_dim AS SELECT listing_id, listing_title
FROM Listing;

CREATE TABLE Listingtype_dim AS SELECT * FROM Listing_Type;

CREATE TABLE HostChannel_Bridge AS SELECT * FROM Host_Verification;

CREATE TABLE Channel_dim AS SELECT * FROM Channel;
```

Script Output x Query Result x  
Task completed in 15.966 seconds

Table LISTING\_DIM created.  
Table LISTINGTYPE\_DIM created.  
Table HOSTCHANNEL\_BRIDGE created.  
Table CHANNEL\_DIM created.

## **Host Dimension**

```
DROP TABLE Host_dim CASCADE CONSTRAINTS PURGE;

CREATE TABLE Host_dim AS
select DISTINCT(h.host_id), h.host_name,
LISTAGG(v.CHANNEL_ID, '_') WITHIN GROUP (ORDER BY v.CHANNEL_ID) AS Channel_group_list,
1/COUNT(*) AS "WEIGHT_FACTOR"
FROM Host h, Host_Verification v
where h.Host_id = v.Host_id
group by h.Host_ID, h.Host_NAME;
```

Script Output x  
Task completed in 0.161 seconds

Table HOST\_DIM created.

## Listing Season dimension

```
--Create Season dimension

DROP TABLE Listingseason_dim CASCADE CONSTRAINTS PURGE;

CREATE TABLE ListingSeason_dim(
Season_Id Number NOT NULL,
Season_desc VARCHAR(6) NOT NULL
);
```

```
--Populate listing season dimension

INSERT INTO ListingSeason_dim VALUES
(1, 'Summer');
INSERT INTO ListingSeason_dim VALUES
(2, 'Autumn');
INSERT INTO ListingSeason_dim VALUES
(3, 'Winter');
INSERT INTO ListingSeason_dim VALUES
(4, 'Spring');

SELECT * FROM ListingSeason_dim;
```

Script Output x Query Result x  
SQL | All Rows Fetched: 4 in 0.191 seconds

SEASON_ID	SEASON_DESC
1	1 Summer
2	2 Autumn
3	3 Winter
4	4 Spring

## Listing Stay duration dimension

```
--Create Listing Stay duration dimension
DROP TABLE Listingstayduration_dim CASCADE CONSTRAINTS PURGE;
CREATE TABLE ListingStayduration_dim(
stayduration_Id VARCHAR(10) NOT NULL,
Stayduration_desc VARCHAR(30) NOT NULL
);
-- Populate the Listing stayduration dimension
INSERT INTO Listingstayduration_dim VALUES
('ShortTerm', 'Less Than 14 nights');
INSERT INTO Listingstayduration_dim VALUES
('MediumTerm', '14 To 30 nights');
INSERT INTO Listingstayduration_dim VALUES
('LongTerm', 'More Than 30 nights');

SELECT * FROM Listingstayduration_dim;
```

Script Output x Query Result x

SQL | All Rows Fetched: 3 in 0.123 seconds

STAYDURATION_ID	STAYDURATION_DESC
1	ShortTerm
2	MediumTerm
3	LongTerm
	Less Than 14 nights
	14 To 30 nights
	More Than 30 nights

## Listing Price Range dimension

```
--Create Listing price range dimension
DROP TABLE Listingpricerange_dim CASCADE CONSTRAINTS PURGE;
CREATE TABLE Listingpricerange_dim(
pricerange_Id VARCHAR(6) NOT NULL,
Pricerange_desc VARCHAR(30) NOT NULL
);
-- POPULATE THE LISTING PRICE RANGE DIMENSION
INSERT INTO Listingpricerange_dim VALUES
('Low', 'Less Than $100');
INSERT INTO Listingpricerange_dim VALUES
('Medium', '$100 To $200');
INSERT INTO Listingpricerange_dim VALUES
('High', 'More Than $200');

SELECT * FROM Listingpricerange_dim;
```

Script Output x Query Result x

SQL | All Rows Fetched: 3 in 0.094 seconds

PRICERANGE_ID	PRICERANGE_DESC
1	Low
2	Medium
3	High
	Less Than \$100
	\$100 To \$200
	More Than \$200

## Booking duration dimension

```
--Create booking duration dimension  
-----  
DROP TABLE Bookingduration_dim CASCADE CONSTRAINTS PURGE;  
CREATE TABLE Bookingduration_dim(  
bookingduration_Id VARCHAR(10) NOT NULL,  
bookingduration_desc VARCHAR(30) NOT NULL  
);  
-----  
----- Populate the Booking duration dimension  
-----  
INSERT INTO Bookingduration_dim VALUES  
( 'ShortTerm', 'Less Than 30 Nights' );  
INSERT INTO Bookingduration_dim VALUES  
( 'MediumTerm', '30 To 90 nights' );  
INSERT INTO Bookingduration_dim VALUES  
( 'LongTerm', 'More Than 90 Nights' );  
  
SELECT * FROM Bookingduration_dim;
```

Script Output		Query Result
		All Rows Fetched: 3 in 0.117 seconds
	BOOKINGDURATION_ID	BOOKINGDURATION_DESC
1	ShortTerm	Less Than 30 Nights
2	MediumTerm	30 To 90 nights
3	LongTerm	More Than 90 Nights

## Booking Cost Dimension

```
-- Creating booking cost range dimension

DROP TABLE Bookingcost_dim CASCADE CONSTRAINTS PURGE;

CREATE TABLE Bookingcost_dim(
bookingcost_id      VARCHAR (6) NOT NULL,
bookingcostrange VARCHAR(30) NOT NULL
);

--Populate the booking Cost range dimension

INSERT INTO Bookingcost_dim VALUES
('Low', 'Less Than $5000');
INSERT INTO Bookingcost_dim VALUES
('Medium', '$5000 To $10000');
INSERT INTO Bookingcost_dim VALUES
('High', 'More Than $10000');

SELECT * FROM Bookingcost_dim;
```

Script Output x Query Result x

SQL | All Rows Fetched: 3 in 0.122 seconds

	BOOKINGCOST_ID	BOOKINGCOSTRANGE
1	Low	Less Than \$5000
2	Medium	\$5000 To \$10000
3	High	More Than \$10000

## Month and Year dimension

```
--Create Month Year dimension--  
-----  
---Listing, Booking and Review table has month and year. So We can have that  
---a common table between Listing fact, Booking fact and Revieew Fact.  
---So to ensure all dates are covered we do union.  
  
DROP TABLE MonthYear_dim CASCADE CONSTRAINTS PURGE;  
  
CREATE TABLE MonthYear_dim AS  
SELECT DISTINCT TO_CHAR(Listing_Date, 'YYYYMM') AS Time_Id,  
TO_CHAR(Listing_Date, 'YYYY') AS Year,  
TO_CHAR(Listing_Date, 'MM') AS MONTH  
FROM Listing  
UNION  
SELECT DISTINCT TO_CHAR(Review_Date, 'YYYYMM') AS Time_Id,  
TO_CHAR(Review_Date, 'YYYY') AS Year,  
TO_CHAR(Review_DATE, 'MM') AS MONTH  
FROM Review  
UNION  
SELECT DISTINCT To_CHAR(Booking_Date, 'YYYYMM') AS Time_Id,  
TO_CHAR(Booking_Date, 'YYYY') AS Year,  
TO_CHAR(Booking_Date, 'MM') AS Month  
FROM Booking;  
  
SELECT * FROM MonthYear_dim;
```

Script Output x Query Result x

SQL | Fetched 50 rows in 0.15 seconds

	TIME_ID	YEAR	MONTH
1	201006	2010	06
2	201007	2010	07
3	201008	2010	08
4	201010	2010	10
5	201011	2010	11
6	201012	2010	12

ASS1.sql

## Temporary Fact Table for Listing

**(Creating temporary fact tables as there are manually created dimensions, and adding other dimensions made manually, by Altering and updating temporary fact table)**

```
----- Create temp fact-----
-----  
  
DROP TABLE temp_fact_listing CASCADE CONSTRAINTS PURGE;  
  
CREATE TABLE temp_fact_Listing AS  
SELECT t.type_id, l.Listing_Id, l.Listing_Date,  
TO_CHAR(Listing_Date, 'YYYYMM') AS Time_Id, 0.host_id,  
l.Listing_Price, l.Listing_Max_Nights  
FROM Listing l, Listing_Type t, Host 0  
WHERE l.type_id = t.Type_id  
AND  
l.host_id = 0.host_Id;  
  
SELECT * FROM temp_fact_listing;
```

## Adding Season\_Id

```
----- Add Season Id-----
-----  
  
ALTER TABLE temp_fact_listing  
ADD(Season_Id Number);  
  
UPDATE temp_fact_listing  
SET Season_Id = 1  
WHERE TO_CHAR(Listing_date, 'MON')= 'DEC'  
OR TO_CHAR(listing_date, 'MON')='JAN'  
OR TO_CHAR(Listing_Date, 'MON') = 'FEB';  
  
UPDATE temp_fact_listing  
SET Season_Id = 2  
WHERE TO_CHAR(Listing_date, 'MON')= 'MAR'  
OR TO_CHAR(listing_date, 'MON')='APR'  
OR TO_CHAR(Listing_Date, 'MON') = 'MAY';  
  
UPDATE temp_fact_listing  
SET Season_Id = 3  
WHERE TO_CHAR(Listing_date, 'MON')= 'JUN'  
OR TO_CHAR(listing_date, 'MON')='JUL'  
OR TO_CHAR(Listing_Date, 'MON') = 'AUG';  
  
UPDATE temp_fact_listing  
SET Season_Id = 4  
WHERE TO_CHAR(Listing_date, 'MON')= 'SEP'  
OR TO_CHAR(listing_date, 'MON')='OCT'  
OR TO_CHAR(Listing_Date, 'MON') = 'NOV';  
  
SELECT * FROM temp_fact_listing;
```

## Adding Pricerange\_Id

```
-----Add Price range Id-----  
  
ALTER TABLE temp_fact_Listing  
ADD(Pricerange_Id VARCHAR(6));  
  
UPDATE temp_fact_listing  
SET Pricerange_Id = 'Low'  
WHERE Listing_price < 100;  
  
UPDATE temp_fact_listing  
SET Pricerange_Id = 'Medium'  
WHERE Listing_price >= 100 AND  
Listing_price <= 200;  
  
UPDATE temp_fact_listing  
SET Pricerange_Id = 'High'  
WHERE Listing_price > 200;
```

## Adding Stayduration\_Id

```
-----Add stay duration Id-----  
  
ALTER TABLE temp_fact_listing  
ADD(Stayduration_Id  VARCHAR(10));  
  
UPDATE temp_fact_listing  
SET stayduration_id = 'ShortTerm'  
WHERE Listing_Max_Nights < 14;  
  
UPDATE temp_fact_listing  
SET stayduration_id = 'MediumTerm'  
WHERE Listing_Max_Nights >= 14 AND  
Listing_Max_nights <= 30;  
  
UPDATE temp_fact_listing  
SET stayduration_id = 'LongTerm'  
WHERE Listing_Max_Nights > 30;
```

## The final temporary Fact table with all added dimension Id.

SELECT \* FROM temp\_fact\_listing;

Script Output x Query Result x

SQL Fetched 50 rows in 0.284 seconds

TYPE_ID	LISTING_ID	LISTING_DATE	TIME_ID	HOST_ID	LISTING_PRICE	LISTING_MAX_NIGHTS	SEASON_ID	STAYDURATION_ID	PRICERANGE_ID
1	2	288 04/JUN/15	201506	390761	292	60	3	LongTerm	High
2	2	289 24/JUN/15	201506	390761	292	60	3	LongTerm	High
3	2	292 13/JUL/15	201507	390761	292	60	3	LongTerm	High
4	2	293 23/JUN/15	201506	390761	292	60	3	LongTerm	High
5	2	294 29/JUL/15	201507	390761	292	60	3	LongTerm	High
6	2	296 28/JUL/15	201507	390761	292	60	3	LongTerm	High
7	2	297 27/JUL/15	201507	390761	292	60	3	LongTerm	High
8	2	298 08/AUG/15	201508	390761	292	60	3	LongTerm	High
9	2	299 30/AUG/15	201508	390761	292	60	3	LongTerm	High
10	2	301 06/SEP/15	201509	390761	292	60	4	LongTerm	High
11	2	303 30/OCT/15	201510	390761	292	60	4	LongTerm	High
12	2	305 21/NOV/15	201511	390761	292	60	4	LongTerm	High
13	2	306 10/DEC/15	201512	390761	292	60	1	LongTerm	High

SELECT COUNT(\*) FROM temp\_fact\_listing;

Script Output x Query Result x

SQL All Rows Fetched: 1 in 0.979 seconds

COUNT(*)
1

## Creating Listing Fact from our Temporary Fact table

Worksheet | Query Builder

```
CREATE TABLE Listing_Fact AS
SELECT T.Type_ID, T.Time_ID, T.Pricerange_Id, T.Season_Id, T.Stayduration_Id,
T.Listing_Id, T.Host_Id, COUNT(T.Listing_Id) AS No_of_Listing
FROM temp_fact_listing T
GROUP BY T.type_ID, T.Time_ID, T.Pricerange_Id, T.Season_Id, T.Stayduration_Id,
T.Listing_Id, T.Host_Id;
|
SELECT COUNT(*) FROM Listing_Fact;
SELECT * FROM LISTING_FACT;
```

Script Output | Query Result | Fetched 50 rows in 0.137 seconds

TYPE_ID	TIME_ID	PRICERANGE_ID	SEASON_ID	STAYDURATION_ID	LISTING_ID	HOST_ID	NO_OF_LISTING
1	2 201507	High		3 LongTerm	296	390761	1
2	2 201509	High		4 LongTerm	301	390761	1
3	2 201511	High		4 LongTerm	305	390761	1
4	2 201604	High		2 LongTerm	318	390761	1
5	2 201612	High		1 LongTerm	327	390761	1
6	2 201612	High		1 LongTerm	330	390761	1
7	2 201708	High		3 LongTerm	338	390761	1
8	2 201802	High		1 LongTerm	347	390761	1
9	2 201412	Low		1 MediumTerm	362	50121	1
10	2 201709	Low		4 MediumTerm	371	50121	1
11	2 201711	Low		4 MediumTerm	372	50121	1
12	2 201906	Low		3 MediumTerm	415	50121	1
13	2 201908	Low		3 MediumTerm	417	50121	1
14	2 201810	Medium		4 LongTerm	432	246509	1
15	2 201205	Medium		2 MediumTerm	467	559227	1
16	2 201611	Low		4 MediumTerm	10	50121	1

SELECT COUNT(\*) FROM Listing\_Fact;

Script Output | Query Result | All Rows Fetched: 1 in 0.109 seconds

COUNT(*)
1
4935

## **Review Fact Table (There are no manually created dimensions so we can create review fact table directly. No need to create Temporary Fact Table)**

```
DROP TABLE Review_Fact CASCADE CONSTRAINTS PURGE,  
CREATE TABLE Review_Fact AS  
SELECT B.Listing_Id, TO_CHAR(R.Review_Date, 'YYYYMM')AS Time_Id,  
COUNT(R.Review_Id) AS No_Of_Reviews  
FROM Booking B, Review R  
WHERE B.Booking_Id = R.Booking_Id  
GROUP BY B.Listing_Id, TO_CHAR(R.Review_Date, 'YYYYMM');  
  
SELECT COUNT(*) FROM Review_Fact;  
  
SELECT * FROM Review_Fact;
```

Script Output x Query Result x

SQL | Fetched 50 rows in 0.094 seconds

LISTING_ID	TIME_ID	NO_OF_REVIEWS
1	76 201508	1
2	111 201612	1
3	1 201105	1
4	11 201701	1
5	13 201801	1
6	14 201802	1
7	17 201807	1
8	26 201903	1
9	29 201904	1
10	33 201906	1
11	36 201910	1
12	39 201910	1
13	41 201912	1
14	48 201012	1
15	60 201303	1
16	119 201703	1
17	122 201704	1

```
SELECT COUNT(*) FROM Review_Fact;
```

Script Output x Query Result x

SQL | All Rows Fetched: 1 in 0.096 seconds

COUNT(*)
1
4869

## **Creating Booking Fact Temporary Table and adding dimension Id's.**

```
--Creating temp Fact table for Booking Fact-----  
  
DROP TABLE temp_fact_booking CASCADE CONSTRAINTS PURGE;  
  
CREATE TABLE temp_fact_booking AS  
SELECT DISTINCT B.booking_ID, B.booking_Duration, B.Booking_cost, T.type_id,  
TO_CHAR(B.Booking_Date, 'YYYYMM')AS Time_Id  
FROM Booking B, Listing L, Listing_Type T  
WHERE B.listing_id = L.listing_id  
AND  
L.Type_Id = T.Type_Id  
GROUP BY B.Booking_Id, B.Booking_Duration, B.Booking_cost, T.type_id,  
TO_CHAR(B.Booking_Date, 'YYYYMM');
```

## **Bookingduration\_Id**

```
-----Add Booking Duration Dimension-----  
  
ALTER TABLE Temp_fact_Booking  
ADD bookingduration_Id  VARCHAR(10);  
  
UPDATE Temp_Fact_Booking  
SET bookingduration_Id = 'ShortTerm'  
WHERE Booking_Duration < 30;  
  
UPDATE Temp_Fact_Booking  
Set bookingduration_Id = 'MediumTerm'  
WHERE Booking_Duration >=30 And  
Booking_Duration <= 90;  
  
UPDATE Temp_Fact_Booking  
SET bookingduration_Id = 'LongTerm'  
WHERE Booking_Duration>90;
```

## **BookingCost\_Id**

```
--Add Booking Cost Dimension--  
  
ALTER TABLE Temp_Fact_Booking  
ADD BookingCost_Id  VARCHAR (6);  
  
UPDATE Temp_Fact_Booking  
SET bookingcost_Id = 'Low'  
WHERE Booking_Cost < 5000;  
  
UPDATE Temp_Fact_Booking  
SET Bookingcost_Id = 'Medium'  
WHERE Booking_Cost >= 5000 AND Booking_Cost <=10000;  
  
UPDATE Temp_Fact_Booking  
SET Bookingcost_Id = 'High'  
WHERE Booking_Cost >10000;
```

## Final Temporary Booking Fact Table.

```
SELECT * FROM Temp_Fact_Booking;
SELECT COUNT(*) FROM Temp_Fact_Booking;
```

Script Output x Query Result x

SQL | Fetched 50 rows in 0.154 seconds

BOOKING_ID	BOOKING_DURATION	BOOKING_COST	TYPE_ID	TIME_ID	BOOKINGDURATION_ID	BOOKINGCOST_ID
1	492	16	2400	2 201307	ShortTerm	Low
2	498	38	5700	2 201309	MediumTerm	Medium
3	522	50	7500	2 201410	MediumTerm	Medium
4	540	32	4800	2 201506	MediumTerm	Low
5	548	47	7050	2 201511	MediumTerm	Medium
6	558	35	5250	2 201604	MediumTerm	Medium
7	576	16	2400	2 201611	ShortTerm	Low
8	584	51	7650	2 201702	MediumTerm	Medium
9	598	60	9000	2 201705	MediumTerm	Medium
10	614	25	3750	2 201711	ShortTerm	Low
11	623	97	14550	2 201802	LongTerm	High
12	655	75	11250	2 201810	MediumTerm	High
13	667	43	6450	2 201901	MediumTerm	Medium
14	676	4	600	2 201905	ShortTerm	Low
15	681	53	7950	2 201906	MediumTerm	Medium
16	694	64	9600	2 201912	MediumTerm	Medium
17	711	57	5529	2 201112	MediumTerm	Medium
18	737	68	6596	2 201602	MediumTerm	Medium
19	774	39	3783	2 201807	MediumTerm	Low

```
SELECT COUNT(*) FROM temp_fact_booking;
```

Script Output x Query Result x

SQL | All Rows Fetched: 1 in 0.133 seconds

COUNT(*)
1 5001

## Creating Booking Fact table from Temporary Fact table

```
--Creating temp Fact table for Booking Fact-----  
  
DROP TABLE temp_fact_booking CASCADE CONSTRAINTS PURGE;  
  
CREATE TABLE temp_fact_booking AS  
SELECT DISTINCT B.booking_ID, B.booking_Duration, B.Booking_cost, T.type_id,  
TO_CHAR(B.Booking_Date, 'YYYYMM')AS Time_Id  
FROM Booking B, Listing L, Listing_Type T  
WHERE B.listing_id = L.listing_id  
AND  
L.Type_Id = T.Type_Id  
GROUP BY B.Booking_Id, B.Booking_Duration, B.Booking_cost, T.type_id,  
TO_CHAR(B.Booking_Date, 'YYYYMM');
```

Script Output x Query Result x

Fetched 50 rows in 0.114 seconds

	BOOKING_ID	BOOKING_DURATION	BOOKING_COST	TYPE_ID	TIME_ID
1	492	16	2400	2	201307
2	498	38	5700	2	201309
3	511	71	10650	2	201404
4	515	96	14400	2	201408
5	522	50	7500	2	201410
6	530	58	8700	2	201412
7	536	51	7650	2	201503
8	540	32	4800	2	201506
9	548	47	7050	2	201511
10	549	64	9600	2	201512
11	556	16	2400	2	201603
12	558	35	5250	2	201604
13	576	16	2400	2	201611

## Create the following reports using OLAP queries.

### (a) Simple reports:

Each report contains two attributes from two different dimensions and one fact measurement.

For the report itself, the first report must be about Top n and the second report is Top n%.

Report 1: Show top(n ) ranking based on total booking cost of different listing types and booking cost range.

-- Using this query we can use to see top n ranking of total booking cost based on listing type and booking cost range and year. As we can see that

—most booking was made of entire home/ apt in the medium booking cost range in different years as below.

Worksheet    Query Builder

```
16 --Report(Top n) Show top(n ) ranking based on total booking cost of
17 --different listing types and booking cost range.
18
19 -- USing this query we can use to see top n ranking of total booking cost
20 --based on listing type and booking cost range and year. As we can see that
21 --most booking was made of entire home/ apt in the medium booking cost range
22 --in different years as below.
23
24
25
26
27
28 SELECT *
29   FROM
30   (SELECT l.type_description, c.bookingcostrange, m.year, SUM(Total_booking_cost),
31    dense_Rank() OVER
32    (ORDER BY SUM(f.Total_booking_cost)DESC) AS Top_Rank
33    FROM Booking_Fact f, Listingtype_dim l, monthyear_dim m, bookingcost_dim c
34    WHERE f.bookingcost_Id = c.bookingcost_Id
35    AND l.type_Id = f.Type_Id
36    AND m.time_Id = f.Time_Id
37    GROUP BY l.type_description, c.bookingcostrange, m.year)
38 WHERE Top_Rank <= 5;
39
```

Query Result x

All Rows Fetched: 5 in 7.548 seconds

TYPE_DESCRIPTION	BOOKINGCOSTRANGE	YEAR	SUM(TOTAL_BOOKING_COST)	TOP_RANK
1 Entire home/apt	\$5000 To \$10000	2017	2236091	1
2 Entire home/apt	\$5000 To \$10000	2015	2044476	2
3 Entire home/apt	\$5000 To \$10000	2018	1995567	3
4 Entire home/apt	\$5000 To \$10000	2016	1823768	4
5 Entire home/apt	\$5000 To \$10000	2019	1758616	5

Report 2: (Top n%) Show ranking of each different channel and list type for each month based on the total number of listings.

--This will be helpfully to find out top n% of total number of listings based on channels, listing type and year.

Worksheet | Query Builder

```
44 -- Report (Top n%) Show ranking of each different channel and list type for
45 -- each month based on the total number of listings.
46
47 --This will be helpfully to find out top n% of total number of listings based on
48 -- channels, listing type and year.
49 -----
50 -----
51
52 SELECT *
53 FROM(
54 SELECT l.type_description, c.channel_name, m.year,
55 SUM(f.No_of_listing) AS total_listing,
56 PERCENT_RANK () OVER
57 (ORDER BY SUM(f.No_of_listing)DESC) AS PERCENT_RANK
58 FROM listing_Fact f, Channel_dim c, monthlyear_dim m, listingtype_dim l,
59 hostchannel_bridge h
60 WHERE f.time_Id = m.time_Id
61 AND f.type_Id = l.type_Id
62 AND f.host_Id = h.host_Id
63 AND h.channel_ID = c.Channel_Id
64 GROUP BY l.type_description, c.channel_name, m.year)
65 WHERE Percent_Rank < 0.1;
```

Query Result | SQL | All Rows Fetched: 21 in 0.126 seconds

TYPE_DESCRIPTION	CHANNEL_NAME	YEAR	TOTAL_LISTING	PERCENT_RANK
1 Entire home/apt	phone	2017	664	0
2 Entire home/apt	reviews	2017	664	0
3 Entire home/apt	phone	2016	655	0.009708737864077669902912621359223300970874
4 Entire home/apt	reviews	2016	655	0.009708737864077669902912621359223300970874
5 Entire home/apt	government_id	2017	630	0.0194174757281553398058252427184466019417
6 Entire home/apt	phone	2015	629	0.0242718446601941747572815533980582524272
7 Entire home/apt	reviews	2015	629	0.0242718446601941747572815533980582524272
8 Entire home/apt	phone	2019	622	0.0339805825242718446601941747572815533981

## (b) Reports with proper sub-totals:

### REPORT 3 and REPORT 4:

--We can get all possible combinations of subtotals and total number of listings based on listing type, season and listing duration.

--We can also decide the dimension we want to add to the combination by using partial cube. For example management will be able to get the all possible combination of listing type and season and partially with listing stay duration.

### Report 3 Cube operator

The screenshot shows the Oracle SQL Developer interface. In the top-left corner, there are tabs for 'Worksheet' and 'Query Builder'. The 'Worksheet' tab is active, displaying the following SQL code:

```
31: --
32: --Report 3 Cube operator
33: --
34:
35:
36: SELECT
37: DECODE(GROUPING(d.Type_description), 1, 'All Listings Types', d.Type_description)
38: AS "Listing_type",
39: DECODE(GROUPING(s.season_desc), 1, 'All listings season', s.season_desc)
40: As "Season",
41: DECODE(GROUPING(stayduration_Id), 1, 'All listings duration', stayduration_Id)
42: AS "Listing_duration",
43: SUM(NO_OF_LISTING) AS "Total_Number_Listings"
44: FROM listing_Fact l, listingtype_dim d, listingseason_dim s
45: WHERE l.type_id = d.type_Id
46: AND l.season_Id = s.season_Id
47: GROUP BY CUBE (Type_description, Season_desc, stayduration_Id)
48: ORDER BY Type_description, Season_desc, stayduration_Id;
49:
50:
51: --
```

In the bottom-right panel, titled 'Query Result', the results of the executed query are displayed in a table:

Listing_type	Season	Listing_duration	Total_Number_Listings
1 Entire home/apt	Autumn	LongTerm	905
2 Entire home/apt	Autumn	MediumTerm	250
3 Entire home/apt	Autumn	All listings duration	1155
4 Entire home/apt	Spring	LongTerm	1011
5 Entire home/apt	Spring	MediumTerm	282
6 Entire home/apt	Spring	All listings duration	1293
7 Entire home/apt	Summer	LongTerm	1000
8 Entire home/apt	Summer	MediumTerm	308
9 Entire home/apt	Summer	All listings duration	1308

## Report 4 Partial Cube Operator

Worksheet    Query Builder

```
51 --Report 3 Partial Cube Operator
52
53
54
55
56 SELECT
57 DECODE(GROUPING(d.Type_description), 1, 'All Listings Types',
58 d.Type_description) AS "Listing_type",
59 DECODE(GROUPING(s.season_desc), 1, 'All listings season',
60 s.season_desc) AS "Season",
61 DECODE(GROUPING(stayduration_Id), 1, 'All listings duration', stayduration_Id)
62 AS "Listing_duration",
63 SUM(NO_OF_LISTING) AS "Total_Number_Listings"
64 FROM listing_Fact l, listingtype_dim d, listingseason_dim s
65 WHERE l.type_id = d.type_Id
66 AND l.season_Id = s.season_Id
67 GROUP BY CUBE (d.Type_description, s.Season_desc), stayduration_Id
68 ORDER BY Type_description, Season_desc, stayduration_Id;
69
70
71
```

Query Result    All Rows Fetched: 35 in 7.31 seconds

Listing_type	Season	Listing_duration	Total_Number_Listings
1 Entire home/apt	Autumn	LongTerm	905
2 Entire home/apt	Autumn	MediumTerm	250
3 Entire home/apt	Spring	LongTerm	1011
4 Entire home/apt	Spring	MediumTerm	282
5 Entire home/apt	Summer	LongTerm	1000
6 Entire home/apt	Summer	MediumTerm	308
7 Entire home/apt	Winter	LongTerm	882
8 Entire home/apt	Winter	MediumTerm	238
9 Entire home/apt	All listings season	LongTerm	3798

REPORT 5 and REPORT 6: Produce two other subtotals reports that are useful for management using Roll-up and Partial Roll-up

-- This will help to understand the total number of bookings based on listing type, booking duration and booking cost. This is more beneficial when management want to study the slices of the cube.

--This will give us the combination of listing type and booking duration and then with the booking cost. This will help to break the totals and subtotals to get required combination

Report 5: Rollup operator

Worksheet    Query Builder

```
86
87
88 -- ROLLUP operator
89
90
91
92 SELECT
93 DECODE(GROUPING(d.type_description), 1, 'All listings type', d.type_description)
94 AS "Listing type",
95 DECODE(GROUPING(bookingduration_Id), 1, 'All Booking duration',
96 bookingduration_Id) AS "Booking duration",
97 DECODE(GROUPING(bookingcost_Id), 1, 'All Booking Cost', bookingcost_Id)
98 AS "Booking Cost",
99 SUM(No_of_bookings) AS "Total Number of Bookings"
100 FROM Booking_Fact b, Listingtype_dim d
101 WHERE b.type_Id = d.type_Id
102 GROUP BY ROLLUP (d.Type_description, bookingduration_Id, bookingcost_Id)
103 ORDER BY d.Type_description, bookingduration_Id, bookingcost_Id;
104
105
106
```

Query Result

All Rows Fetched: 35 in 7.31 seconds

Listing_type	Season	Listing_duration	Total_Number_Listings
1 Entire home/apt	Autumn	LongTerm	905
2 Entire home/apt	Autumn	MediumTerm	250
3 Entire home/apt	Spring	LongTerm	1011
4 Entire home/apt	Spring	MediumTerm	282
5 Entire home/apt	Summer	LongTerm	1000
6 Entire home/apt	Summer	MediumTerm	308
7 Entire home/apt	Winter	LongTerm	882
8 Entire home/apt	Winter	MediumTerm	238
9 Entire home/apt	All listings season	LongTerm	3798

## Report 6: Partial ROLLUP operator

Worksheet | Query Builder

```
111 -- Partial ROLLUP operator
112
113
114
115
116 SELECT
117 DECODE(GROUPING(d.type_description), 1, 'All listings type', d.type_description)
118 AS "Listing type",
119 DECODE(GROUPING(bookingduration_Id), 1, 'All Booking duration',
120 bookingduration_Id) AS "Booking duration",
121 DECODE(GROUPING(bookingcost_Id), 1, 'All Booking Cost', bookingcost_Id)
122 AS "Booking Cost",
123 SUM(No_of_bookings) As "Total Number of Bookings"
124 FROM Booking_Fact b, Listingtype_dim d
125 WHERE b.type_Id = d.type_Id
126 GROUP BY ROLLUP (d.Type_description, bookingduration_Id), bookingcost_Id
127 ORDER BY d.Type_description, bookingduration_Id, bookingcost_Id;
128
129
130
131
```

Query Result | All Rows Fetched: 21 in 0.127 seconds

Listing type	Booking duration	Booking Cost	Total Number of Bookings
1 Entire home/apt	LongTerm	High	296
2 Entire home/apt	LongTerm	Medium	231
3 Entire home/apt	MediumTerm	High	552
4 Entire home/apt	MediumTerm	Low	750
5 Entire home/apt	MediumTerm	Medium	1769
6 Entire home/apt	ShortTerm	High	3
7 Entire home/apt	ShortTerm	Low	1306
8 Entire home/apt	ShortTerm	Medium	35
9 Entire home/apt	All Booking duration	High	851

(c) Reports with moving and cumulative aggregates:

#### REPORT 7: Cumulative total

-- This will helpful as it will provide information about total number of bookings and cumulative total number of bookings for hotel room in each year.

The screenshot shows the Oracle SQL Developer interface. The top part is the 'Worksheet' tab, which contains the following SQL code:

```
13
14
15 -- Report 7 Cumulative total
16
17 SELECT * FROM LISTINGTYPE_DIM;
18
19 SELECT m.Year, l.type_description, sum(no_of_bookings),
20       TO_CHAR(SUM(SUM(no_of_bookings))
21             OVER(ORDER BY m.Year ROWS UNBOUNDED PRECEDING), '9,999,999')
22       AS Cummulative_Booking
23   FROM Booking_Fact b, MonthYear_dim m, listingtype_dim l
24  WHERE b.time_Id = m.time_Id
25  AND b.type_Id = l.type_Id
26  AND l.type_description LIKE 'Hotel room'
27 GROUP BY m.Year, l.type_description;
28
29
```

The bottom part is the 'Query Result' tab, which displays the results of the executed query. The results table has the following structure:

YEAR	TYPE_DESCRIPTION	SUM(NO_OF_BOOKINGS)	CUMMATIVE_BOOKING
2000	Hotel room	1000	1000
2001	Hotel room	1500	2500
2002	Hotel room	2000	4500
2003	Hotel room	2500	7000
2004	Hotel room	3000	10000
2005	Hotel room	3500	13500
2006	Hotel room	4000	17500
2007	Hotel room	4500	22000
2008	Hotel room	5000	27000
2009	Hotel room	5500	32500
2010	Hotel room	6000	38500
2011	Hotel room	6500	45000
2012	Hotel room	7000	52000
2013	Hotel room	7500	59500
2014	Hotel room	8000	67500
2015	Hotel room	8500	75500
2016	Hotel room	9000	84000
2017	Hotel room	9500	92500
2018	Hotel room	10000	101000
2019	Hotel room	10500	109500
2020	Hotel room	11000	118000
2021	Hotel room	11500	126500
2022	Hotel room	12000	135000
2023	Hotel room	12500	143500
2024	Hotel room	13000	152000
2025	Hotel room	13500	160500
2026	Hotel room	14000	169000
2027	Hotel room	14500	177500
2028	Hotel room	15000	186000
2029	Hotel room	15500	194500
2030	Hotel room	16000	203000
2031	Hotel room	16500	211500
2032	Hotel room	17000	220000
2033	Hotel room	17500	228500
2034	Hotel room	18000	237000
2035	Hotel room	18500	245500
2036	Hotel room	19000	254000
2037	Hotel room	19500	262500
2038	Hotel room	20000	271000
2039	Hotel room	20500	279500
2040	Hotel room	21000	288000
2041	Hotel room	21500	296500
2042	Hotel room	22000	305000
2043	Hotel room	22500	313500
2044	Hotel room	23000	322000
2045	Hotel room	23500	330500
2046	Hotel room	24000	339000
2047	Hotel room	24500	347500
2048	Hotel room	25000	356000
2049	Hotel room	25500	364500
2050	Hotel room	26000	373000
2051	Hotel room	26500	381500
2052	Hotel room	27000	390000
2053	Hotel room	27500	398500
2054	Hotel room	28000	407000
2055	Hotel room	28500	415500
2056	Hotel room	29000	424000
2057	Hotel room	29500	432500
2058	Hotel room	30000	441000
2059	Hotel room	30500	449500
2060	Hotel room	31000	458000
2061	Hotel room	31500	466500
2062	Hotel room	32000	475000
2063	Hotel room	32500	483500
2064	Hotel room	33000	492000
2065	Hotel room	33500	500500
2066	Hotel room	34000	509000
2067	Hotel room	34500	517500
2068	Hotel room	35000	526000
2069	Hotel room	35500	534500
2070	Hotel room	36000	543000
2071	Hotel room	36500	551500
2072	Hotel room	37000	560000
2073	Hotel room	37500	568500
2074	Hotel room	38000	577000
2075	Hotel room	38500	585500
2076	Hotel room	39000	594000
2077	Hotel room	39500	602500
2078	Hotel room	40000	611000
2079	Hotel room	40500	619500
2080	Hotel room	41000	628000
2081	Hotel room	41500	636500
2082	Hotel room	42000	645000
2083	Hotel room	42500	653500
2084	Hotel room	43000	662000
2085	Hotel room	43500	670500
2086	Hotel room	44000	679000
2087	Hotel room	44500	687500
2088	Hotel room	45000	696000
2089	Hotel room	45500	704500
2090	Hotel room	46000	713000
2091	Hotel room	46500	721500
2092	Hotel room	47000	730000
2093	Hotel room	47500	738500
2094	Hotel room	48000	747000
2095	Hotel room	48500	755500
2096	Hotel room	49000	764000
2097	Hotel room	49500	772500
2098	Hotel room	50000	781000
2099	Hotel room	50500	789500
2100	Hotel room	51000	798000
2101	Hotel room	51500	806500
2102	Hotel room	52000	815000
2103	Hotel room	52500	823500
2104	Hotel room	53000	832000
2105	Hotel room	53500	840500
2106	Hotel room	54000	849000
2107	Hotel room	54500	857500
2108	Hotel room	55000	866000
2109	Hotel room	55500	874500
2110	Hotel room	56000	883000
2111	Hotel room	56500	891500
2112	Hotel room	57000	898000
2113	Hotel room	57500	906500
2114	Hotel room	58000	915000
2115	Hotel room	58500	923500
2116	Hotel room	59000	932000
2117	Hotel room	59500	940500
2118	Hotel room	60000	949000
2119	Hotel room	60500	957500
2120	Hotel room	61000	966000
2121	Hotel room	61500	974500
2122	Hotel room	62000	983000
2123	Hotel room	62500	991500
2124	Hotel room	63000	998000
2125	Hotel room	63500	1006500
2126	Hotel room	64000	1015000
2127	Hotel room	64500	1023500
2128	Hotel room	65000	1032000
2129	Hotel room	65500	1040500
2130	Hotel room	66000	1049000
2131	Hotel room	66500	1057500
2132	Hotel room	67000	1066000
2133	Hotel room	67500	1074500
2134	Hotel room	68000	1083000
2135	Hotel room	68500	1091500
2136	Hotel room	69000	1098000
2137	Hotel room	69500	1106500
2138	Hotel room	70000	1115000
2139	Hotel room	70500	1123500
2140	Hotel room	71000	1132000
2141	Hotel room	71500	1140500
2142	Hotel room	72000	1149000
2143	Hotel room	72500	1157500
2144	Hotel room	73000	1166000
2145	Hotel room	73500	1174500
2146	Hotel room	74000	1183000
2147	Hotel room	74500	1191500
2148	Hotel room	75000	1198000
2149	Hotel room	75500	1206500
2150	Hotel room	76000	1215000
2151	Hotel room	76500	1223500
2152	Hotel room	77000	1232000
2153	Hotel room	77500	1240500
2154	Hotel room	78000	1249000
2155	Hotel room	78500	1257500
2156	Hotel room	79000	1266000
2157	Hotel room	79500	1274500
2158	Hotel room	80000	1283000
2159	Hotel room	80500	1291500
2160	Hotel room	81000	1298000
2161	Hotel room	81500	1306500
2162	Hotel room	82000	1315000
2163	Hotel room	82500	1323500
2164	Hotel room	83000	1332000
2165	Hotel room	83500	1340500
2166	Hotel room	84000	1349000
2167	Hotel room	84500	1357500
2168	Hotel room	85000	1366000
2169	Hotel room	85500	1374500
2170	Hotel room	86000	1383000
2171	Hotel room	86500	1391500
2172	Hotel room	87000	1398000
2173	Hotel room	87500	1406500
2174	Hotel room	88000	1415000
2175	Hotel room	88500	1423500
2176	Hotel room	89000	1432000
2177	Hotel room	89500	1440500
2178	Hotel room	90000	1449000
2179	Hotel room	90500	1457500
2180	Hotel room	91000	1466000
2181	Hotel room	91500	1474500
2182	Hotel room	92000	1483000
2183	Hotel room	92500	1491500
2184	Hotel room	93000	1498000
2185	Hotel room	93500	1506500
2186	Hotel room	94000	1515000
2187	Hotel room	94500	1523500
2188	Hotel room	95000	1532000
2189	Hotel room	95500	1540500
2190	Hotel room	96000	1549000
2191	Hotel room	96500	1557500
2192	Hotel room	97000	1566000
2193	Hotel room	97500	1574500
2194	Hotel room	98000	1583000
2195	Hotel room	98500	1591500
2196	Hotel room	99000	1598000
2197	Hotel room	99500	1606500
2198	Hotel room	100000	1615000
2199	Hotel room	100500	1623500
2200	Hotel room	101000	1632000
2201	Hotel room	101500	1640500
2202	Hotel room	102000	1649000
2203	Hotel room	102500	1657500
2204	Hotel room	103000	1666000
2205	Hotel room	103500	1674500
2206	Hotel room	104000	1683000
2207	Hotel room	104500	1691500
2208	Hotel room	105000	1698000
2209	Hotel room	105500	1706500
2210	Hotel room	106000	1715000
2211	Hotel room	106500	1723500
2212	Hotel room	107000	1732000
2213	Hotel room	107500	1740500
2214	Hotel room	108000	1749000
2215	Hotel room	108500	1757500
2216	Hotel room	109000	1766000
2217	Hotel room	109500	1774500
2218	Hotel room	110000	1783000
2219	Hotel room	110500	1791500
2220	Hotel room	111000	1798000
2221	Hotel room	111500	1806500
2222	Hotel room	112000	1815000
2223	Hotel room	112500	1823500
2224	Hotel room	113000	1832000
2225	Hotel room	113500	1840500
2226	Hotel room	114000	1849000
2227	Hotel room	114500	1857500
2228	Hotel room	115000	1866000
2229	Hotel room	115500	1874500
2230	Hotel room	116000	1883000
2231	Hotel room	116500	1891500
2232	Hotel room	117000	1898000
2233	Hotel room	117500	1906500
2234	Hotel room	118000	1915000
2235	Hotel room	118500	1923500
2236	Hotel room	119000	1932000
2237	Hotel room	119500	1940500
2238	Hotel room	120000	1949000
2239	Hotel room	120500	1957500
2240	Hotel room	121000	1966000
2241	Hotel room	121500	1974500
2242	Hotel room	122000	1983000
2243	Hotel room	122500	1991500
2244	Hotel room	123000	1998000
2245	Hotel room	123500	2006500
2246	Hotel room	124000	2015000
2247	Hotel room	124500	2023500
2248	Hotel room	125000	2032000
2249	Hotel room	125500	2040500
2250	Hotel room	126000	2049000
2251	Hotel room	126500	2057500
2252	Hotel room	127000	2066000
2253	Hotel room	127500	2074500
2254	Hotel room	128000	2083000
2255	Hotel room	128500	2091500
2256	Hotel room	129000	2098000
2257	Hotel room		

## Report 8: moving/cumulative aggregate

-- This will be useful to find out the total booking cost and moving/ cumulative average booking cost of listings in each year. We will be looking at 6 month moving average, which will help provide details about any variations that we cannot see in actual total booking cost. We will concentrate on Entire home/appt category for listing as we have seen  
It was top rank in total number of bookings.

Worksheet    Query Builder

```
38 -- Report 8 moving/cumulative aggregate
39 -----
40
41 -- What is the average total booking cost and moving/cumulative average booking
42 --cost of listings in each year
43
44 -- This will be useful to find out the total booking cost and moving/ cumulative
45 --average booking cost of listings in each year. We will be looking at 6 month
46 --moving average, which will help provide details about any
47 --variations that we cannot see in actual total booking cost.
48 --We will concentrate on Entire home/appt category for listing as we have seen
49 --It was top rank in total number of bookings.
50 -----
51 -----
52
53 ┌─ SELECT l.type_description, m.year,
54   TO_CHAR(SUM(f.Total_booking_cost), '9,999,999.99') AS Total_bookingcost,
55   TO_CHAR( AVG(SUM(f.Total_booking_cost))
56   OVER(ORDER BY m.year ROWS 5 PRECEDING),
57   '9,999,999.99') AS MOVING_6_MONTHS_AVG
58   FROM booking_fact f, listingtype_dim l, monthyear_dim m
59   WHERE l.type_Id = f.type_Id
60   AND m.time_Id = f.time_Id
61   AND l.type_description LIKE 'Entire home/apt'
62   GROUP BY m.year, l.type_description
63   ORDER BY m.year, l.type_description ASC;
```

Query Result

All Rows Fetched: 12 in 0.102 seconds

TYPE_DESCRIPTION	YEAR	TOTAL_BOOKINGCOST	MOVING_6_MONTHS_A.
1 Entire home/apt	2010	20,502.00	20,502.00
2 Entire home/apt	2011	301,997.00	161,249.50
3 Entire home/apt	2012	1,361,666.00	561,388.33
4 Entire home/apt	2013	2,832,598.00	1,129,190.75
5 Entire home/apt	2014	3,540,361.00	1,611,424.80

(d) Reports with Partitions:

--This will help us to categorise the ranking of total number of booking based on booking duration and listing type.

Worksheet    Query Builder

```
5  
6  
7 --REPORT 9: Show ranking of each booking duration range and ranking of each  
8 --listing type based on the total number of bookings.  
9  
10 --This will help us to categorise the ranking of total number of booking  
11 --based on booking duration and listing type.  
12  
13  
14  
15  SELECT b.bookingduration_desc, l.type_description,  
16 SUM(f.No_of_bookings) AS TOTAL_bookings,  
17 RANK () OVER (PARTITION BY b.bookingduration_desc  
18 ORDER BY SUM(f.No_of_bookings)DESC) AS Rankby_bookingduration,  
19 RANK() OVER (PARTITION BY l.type_description  
20 ORDER BY SUM(f.No_of_bookings)DESC) AS Rankby_listingtype  
21 FROM Booking_Fact f, Listingtype_dim l, Bookingduration_dim b  
22 WHERE l.type_id = f.type_Id  
23 AND b.bookingduration_Id = f.bookingduration_Id  
24 GROUP BY b.bookingduration_desc, l.type_description;  
25  
26
```

Query R... x

SQL | All Rows Fetched: 6 in 0.107 seconds

BOOKINGDURATION_DESC	TYPE_DESCRIPTION	TOTAL_BOOKING	RANKBY_BOOKINGDURATION	RANKBY_LISTIN
1 30 To 90 nights	Entire home/apt	3071		1
2 30 To 90 nights	Private room	35		2
3 Less Than 30 Nights	Entire home/apt	1344		1
4 Less Than 30 Nights	Private room	18		2
5 More Than 90 Nights	Entire home/apt	527		1
6 More Than 90 Nights	Private room	6		2

REPORT 10: Show ranking of each listing stay duration range and of each listing type based on the total no of listings.

--This can be helpful to categorise top no of listing based on different listing type and listed stay duration.

Worksheet    Query Builder

```
31
32
33 --REPORT 10: Show ranking of each listing stay duration range and of each
34 --listing type based on the total no of listings.
35
36 --This can be helpful to categorise top no of listing based on different listing
37 --type and listed stay duration.
38
39
40
41 SELECT * FROM listingstayduration_dim;
42
43 SELECT l.type_description, p.stayduration_desc,
44 SUM(f.No_of_listing) AS Total_listing,
45 RANK ()OVER (PARTITION BY p.stayduration_desc
46 ORDER BY SUM(f.No_of_listing)DESC) AS Rankby_stayduration,
47 RANK() OVER (PARTITION BY l.type_description
48 ORDER BY SUM(f.No_of_listing)DESC) AS Rankby_listingtype
49 FROM Listing_Fact f, Listingtype_dim l, listingstayduration_dim p
50 WHERE l.type_id = f.type_Id
51 AND p.stayduration_Id = f.stayduration_Id
52 GROUP BY l.type_description, p.stayduration_desc;
```

Query Result

All Rows Fetched: 4 in 0.113 seconds

TYPE_DESCRIPTION	STAYDURATION_DESC	TOTAL_LISTING	RANKBY_STAYDURATION	RANKBY_LISTINGTYPE
1 Entire home/apt	14 To 30 nights	1078	1	2
2 Private room	Less Than 14 nights	23	1	2
3 Entire home/apt	More Than 30 nights	3798	1	1
4 Private room	More Than 30 nights	36	2	1

```
SELECT COUNT (*) FROM Booking_Fact;
```

Script Output    Query Result

All Rows Fetched: 1 in 0.123 seconds

COUNT(*)
1
766