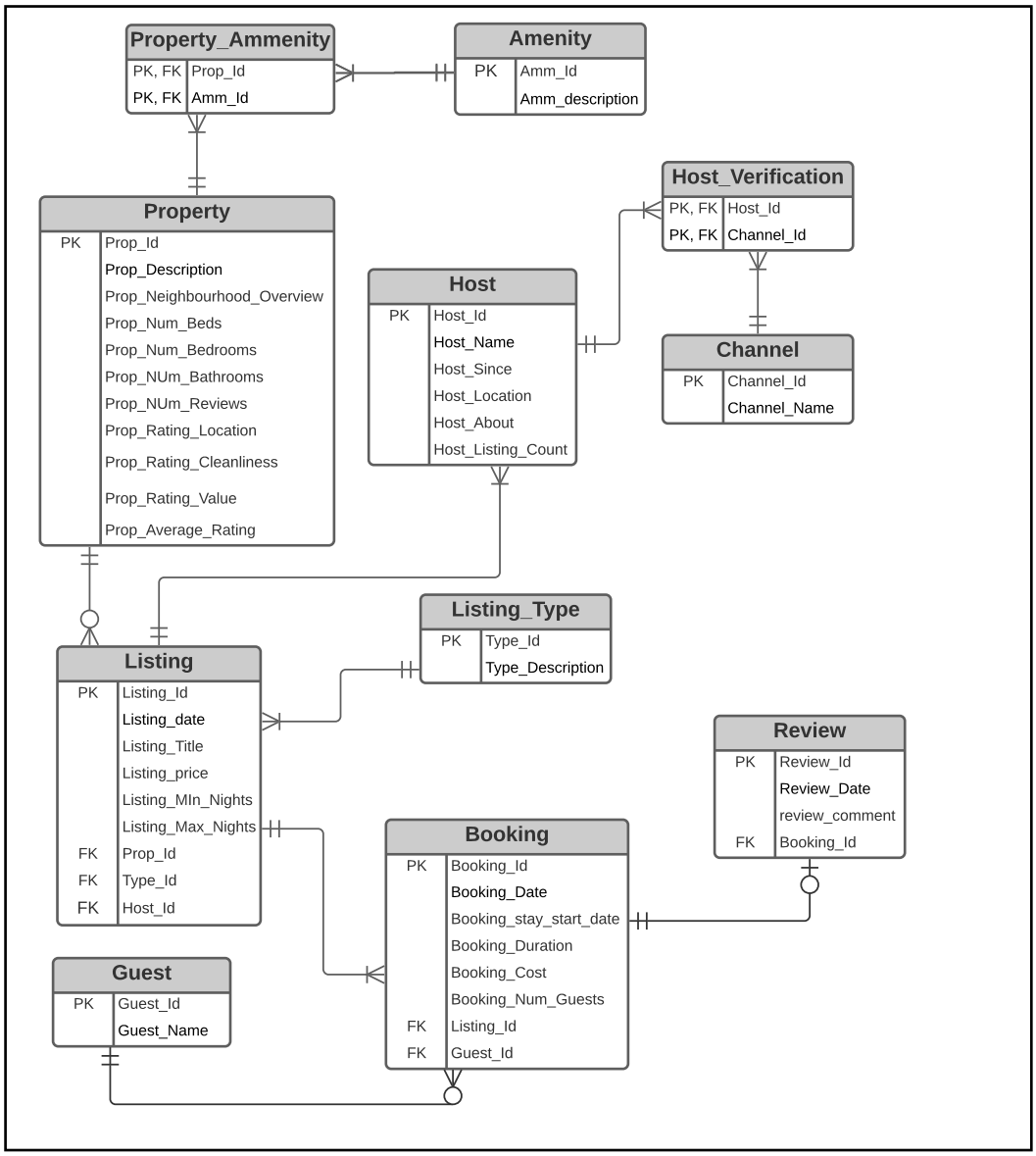
ER Diagram of the Operational database



Exploring data

```sql
--- 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

```sql
----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.

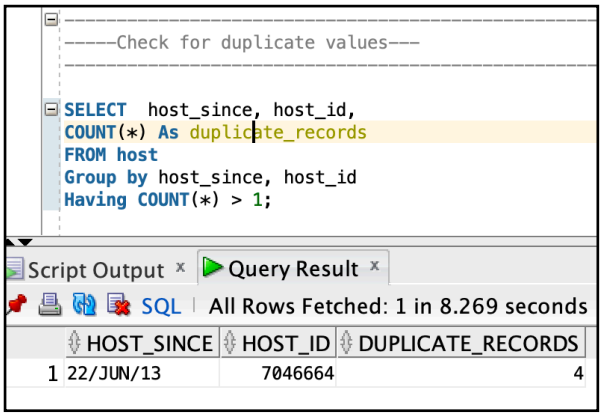
```
-------------------------------------------------------------------------------------------------
--- While exploring the channel I found a value as none, lets look into it--
-------------------------------------------------------------------------------------------------

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

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.

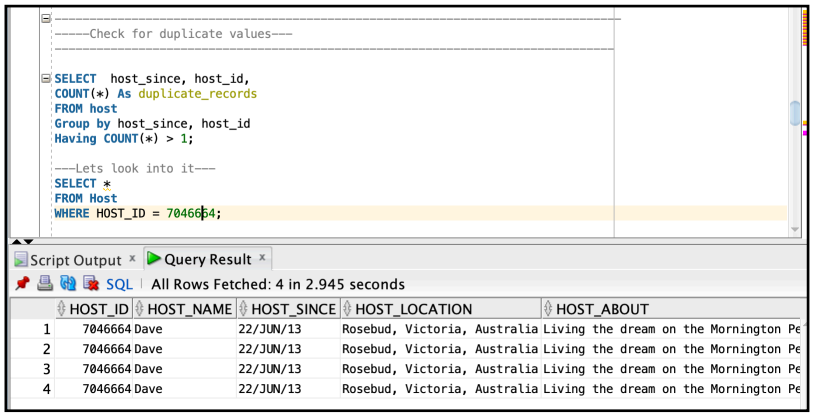


```
-----Check for duplicate values---

SELECT  host_since, host_id,
COUNT(*) As duplicate_records
FROM host
Group by host_since, host_id
Having COUNT(*) > 1;
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 8.269 seconds

	HOST_SINCE	HOST_ID	DUPLICATE_RECORDS
1	22/JUN/13	7046664	4

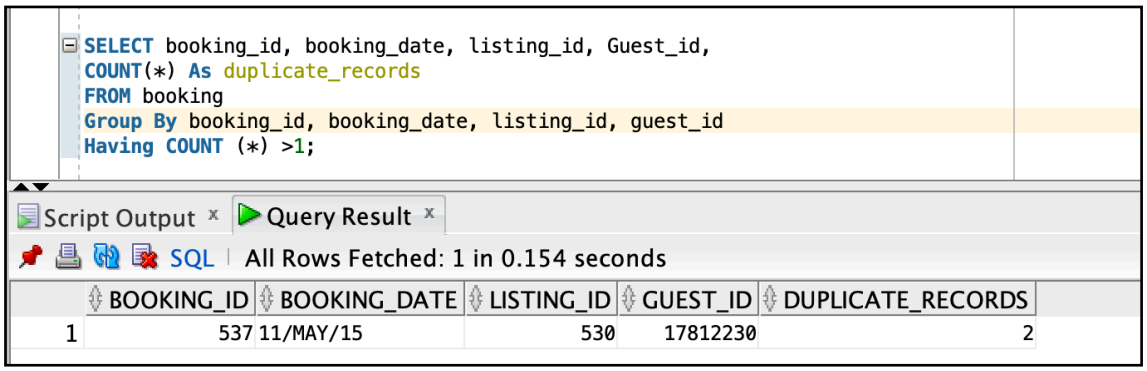


```
-----Check for duplicate values---

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 | Query Result

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

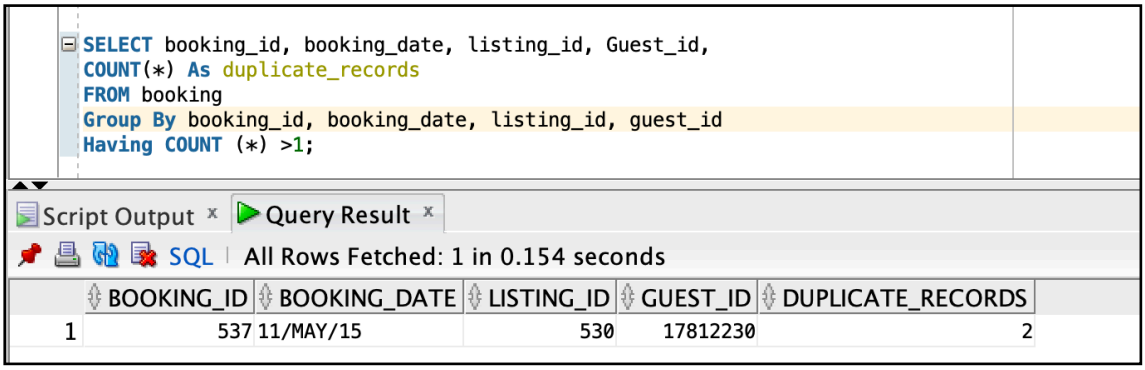


```
SELECT booking_id, booking_date, listing_id, Guest_id,
COUNT(*) As duplicate_records
FROM booking
Group By booking_id, booking_date, listing_id, guest_id
Having COUNT (*) >1;
```

Script Output | Query Result

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



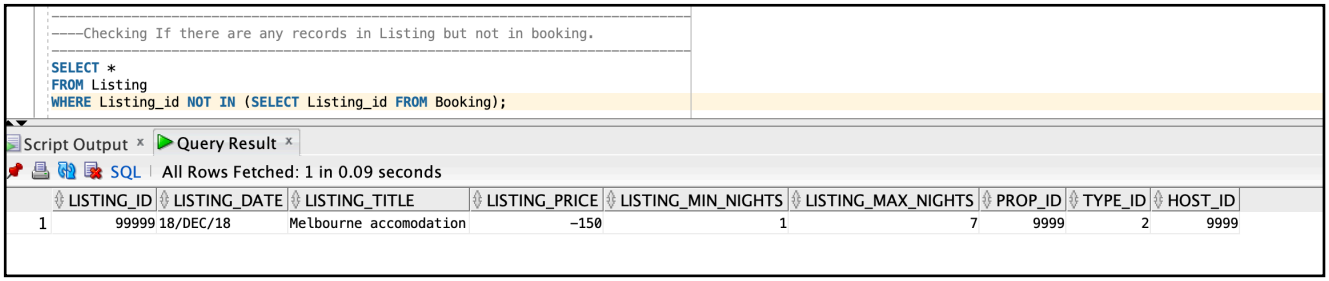
```
SELECT booking_id, booking_date, listing_id, Guest_id,
COUNT(*) As duplicate_records
FROM booking
Group By booking_id, booking_date, listing_id, guest_id
Having COUNT (*) >1;
```

Script Output | Query Result

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.

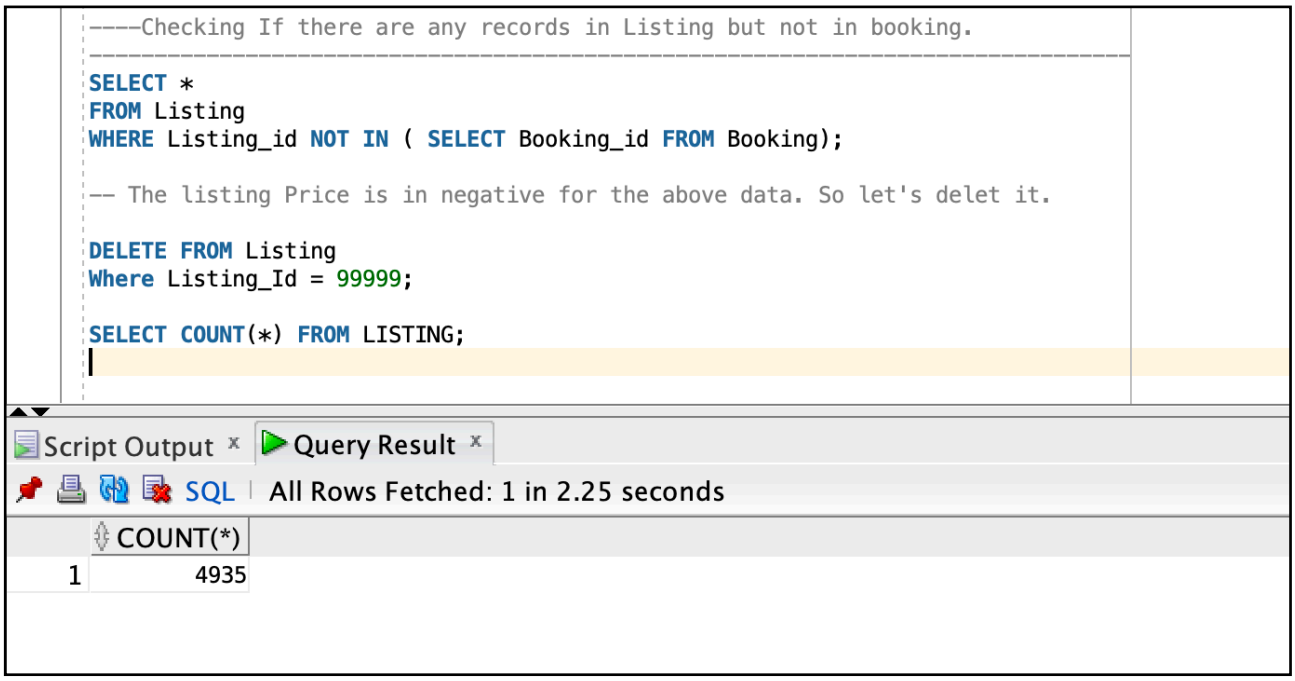


```
-----Checking If there are any records in Listing but not in booking.
SELECT *
FROM Listing
WHERE Listing_id NOT IN (SELECT Listing_id FROM Booking);
```

Script Output | Query Result

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



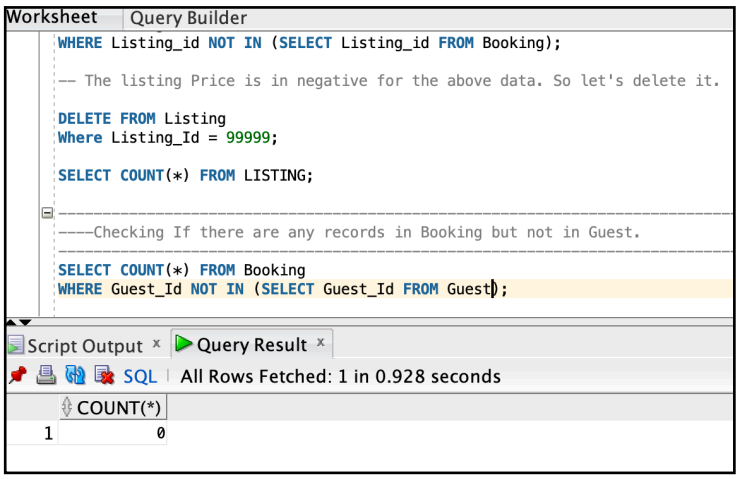
```
-----Checking If there are any records in Listing but not in booking.
SELECT *
FROM Listing
WHERE Listing_id NOT IN ( SELECT Booking_id FROM Booking);

-- 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 | Query Result

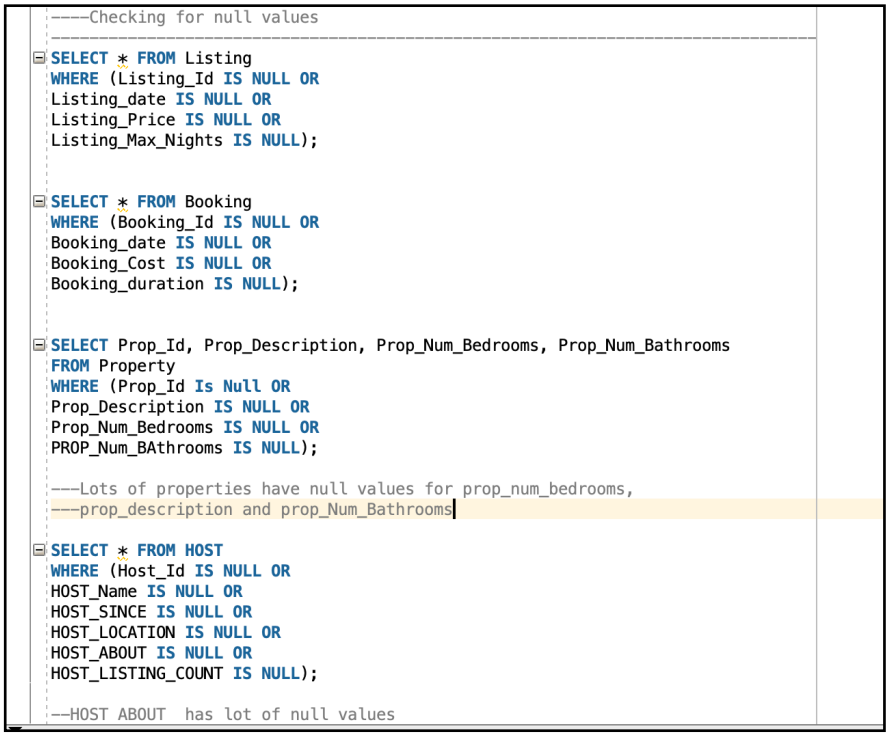
SQL | All Rows Fetched: 1 in 2.25 seconds

	COUNT(*)
1	4935

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

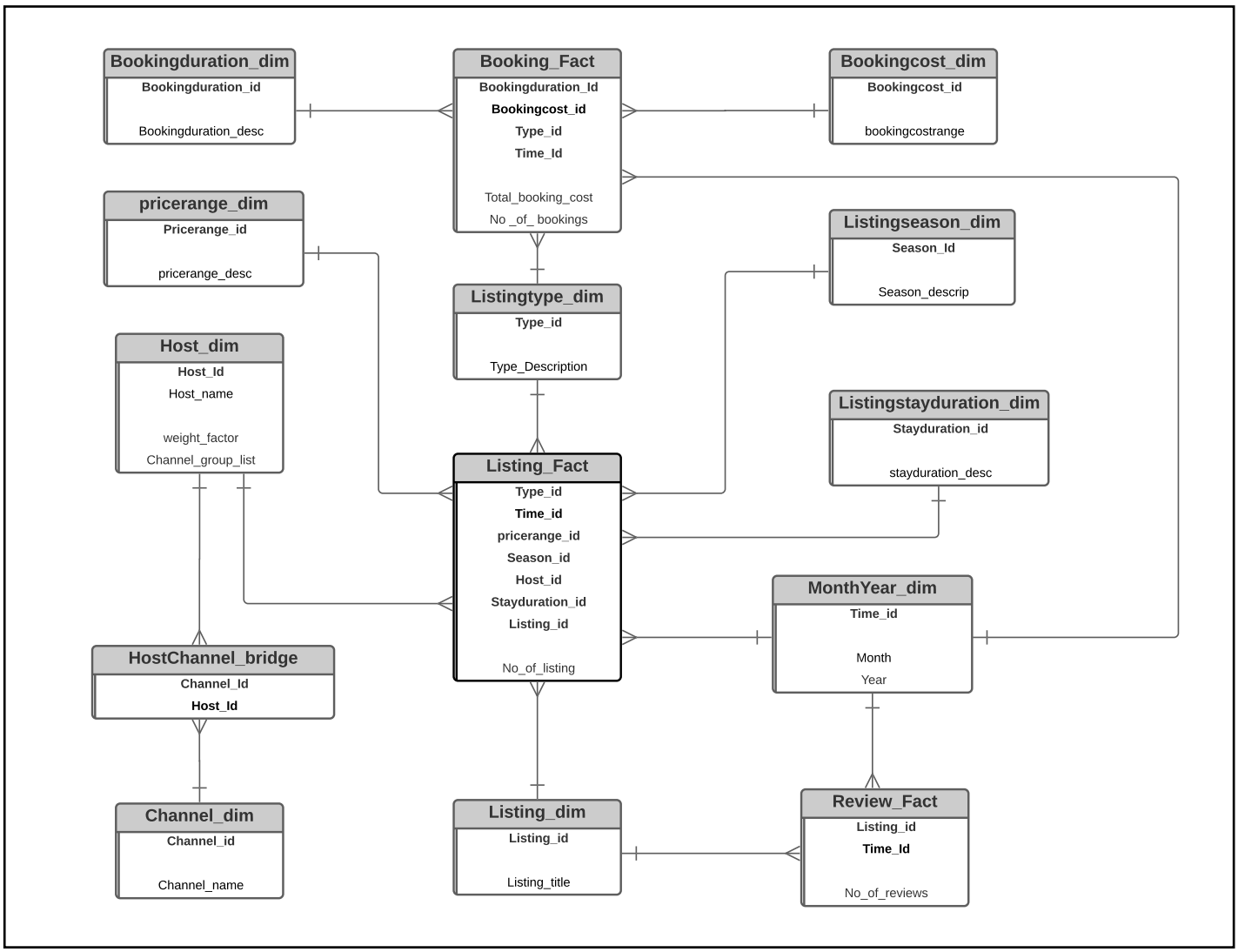


Checking for Null values

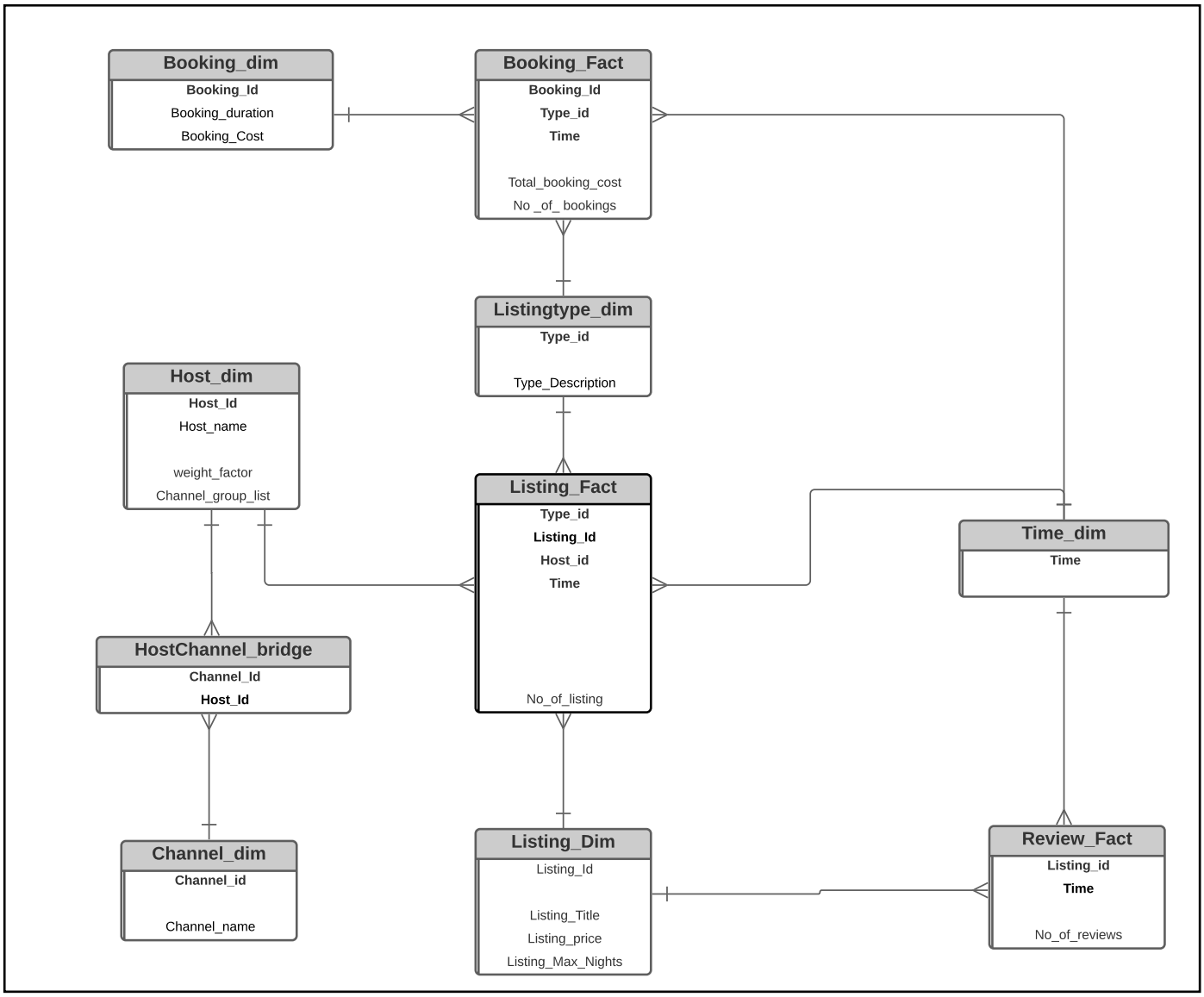


We will not replace the null values at this stage as we are not going to use this attributes in our dimension and fact tables. It may also be due to unknown number of beds in property example or in host it could be that host do not want to add something about himself.

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 | Query Result

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

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
);
```

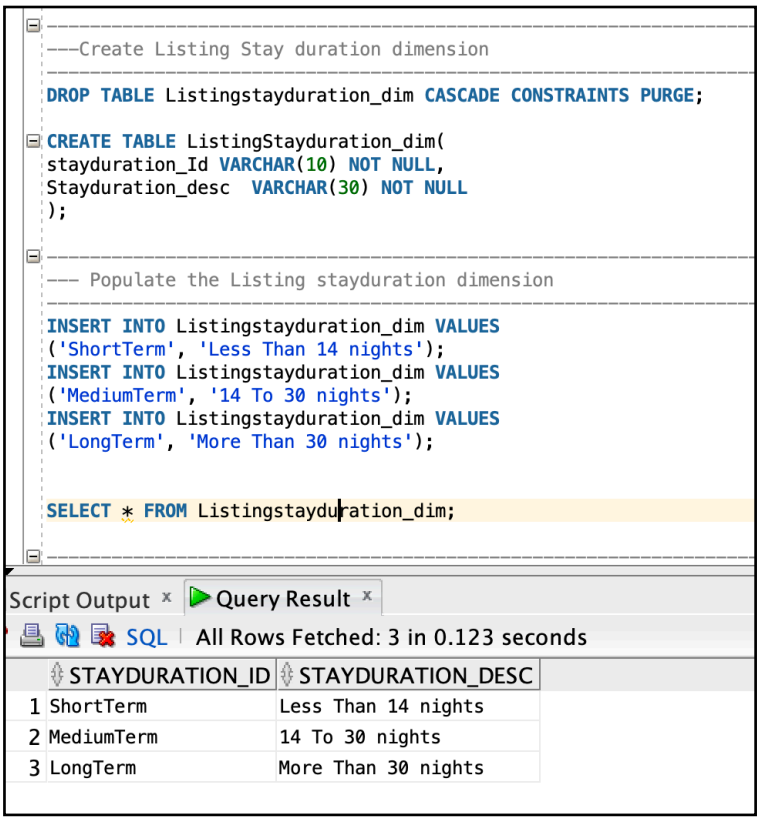
```
---Populate listing season dimension
INSERT INTO ListingSeason_dim VALUES
(1,'Summer');
INSERT INTO ListingSeason_dim VALUES
(2,'Autumn');
INSERT INTO ListingSeason_dim VALUES
(3,'Winter');
INSERT INTO ListingSeason_dim VALUES
(4,'Spring');

SELECT * FROM ListingSeason_dim;
```

Script Output | Query Result

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



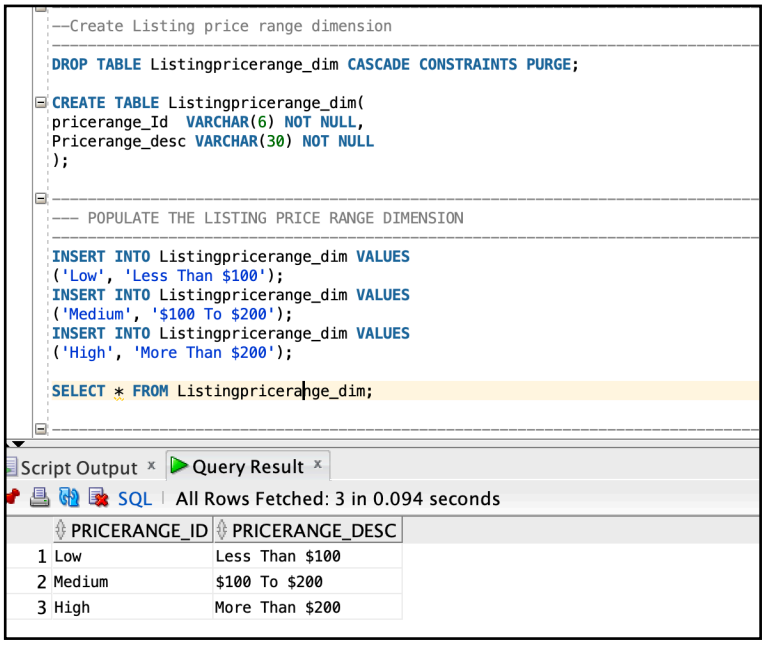
```
---Create Listing Stay duration dimension
DROP TABLE Listingstayduration_dim CASCADE CONSTRAINTS PURGE;

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 | Query Result

SQL | All Rows Fetched: 3 in 0.123 seconds

	STAYDURATION_ID	STAYDURATION_DESC
1	ShortTerm	Less Than 14 nights
2	MediumTerm	14 To 30 nights
3	LongTerm	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 | Query Result

SQL | All Rows Fetched: 3 in 0.094 seconds

	PRICERANGE_ID	PRICERANGE_DESC
1	Low	Less Than $100
2	Medium	$100 To $200
3	High	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
);

--------------------------------------------------------------------------------
---- Polpulate 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

SQL | 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 | Query Result

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 | Query Result

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, O.host_id,
l.Listing_price, l.Listing_Max_Nights
FROM Listing l, Listing_Type t, Host O
WHERE l.type_id = t.Type_id
AND
l.host_id = O.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 | Query Result

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 | Query Result

SQL | All Rows Fetched: 1 in 0.979 seconds

	COUNT(*)
1	4935

Creating Listing Fact from our Temporary Fact table

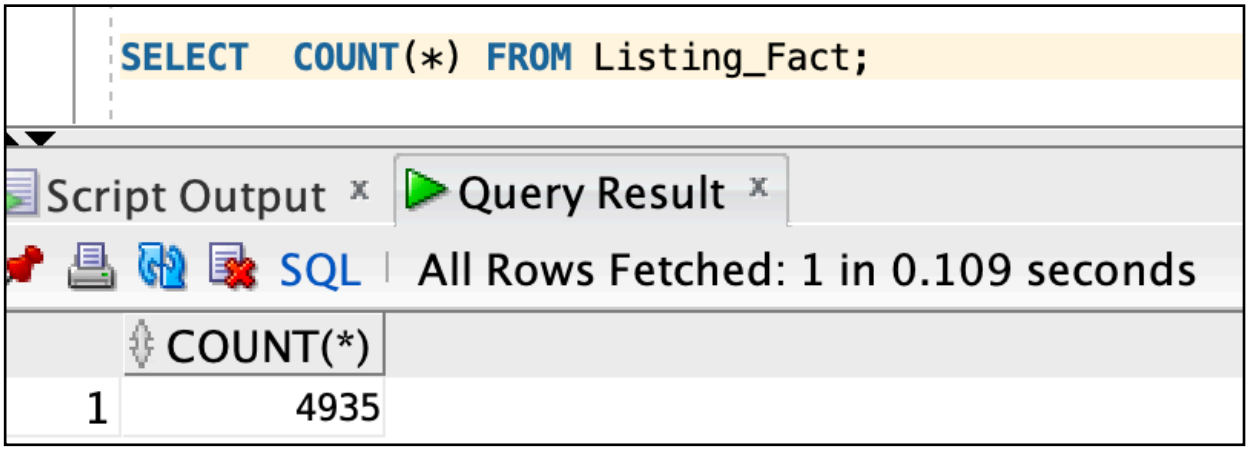
Worksheet | Query Builder

```
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

SQL | 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



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 | Query Result

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 | Query Result

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 | Query Result

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 | Query Result

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 | Query Result

SQL | 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

Create the following reports using OLAP queries.

(a) Simple reports:

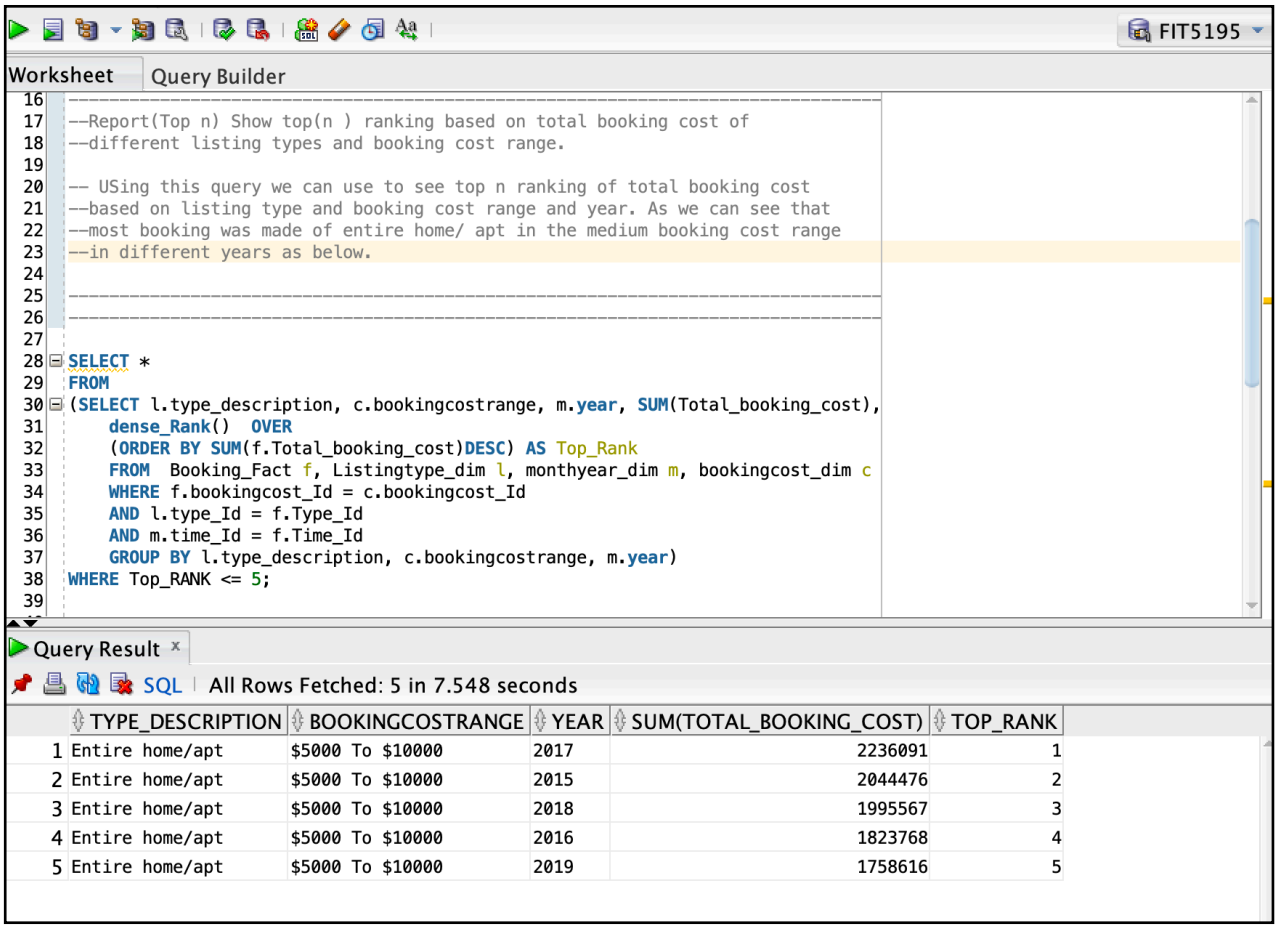
Each report contains two attributes from two different
dimensions and one fact measurement.

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.



```
--------------------------------------------------------------------------------
--Report(Top n) 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.


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

SELECT *
FROM
(SELECT l.type_description, c.bookingcostrange, m.year, SUM(Total_booking_cost),
    dense_Rank()  OVER
    (ORDER BY SUM(f.Total_booking_cost)DESC) AS Top_Rank
    FROM  Booking_Fact f, Listingtype_dim l, monthyear_dim m, bookingcost_dim c
    WHERE f.bookingcost_Id = c.bookingcost_Id
    AND l.type_Id = f.Type_Id
    AND m.time_Id = f.Time_Id
    GROUP BY l.type_description, c.bookingcostrange, m.year)
WHERE Top_RANK <= 5;
```

Query Result

SQL | 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, monthyear_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	0.019417475728155339805825242718446601941 7
6	Entire home/apt	phone	2015	629	0.024271844660194174757281553398058252427 2
7	Entire home/apt	reviews	2015	629	0.024271844660194174757281553398058252427 2
8	Entire home/apt	phone	2019	622	0.033980582524271844660194174757281553398 1

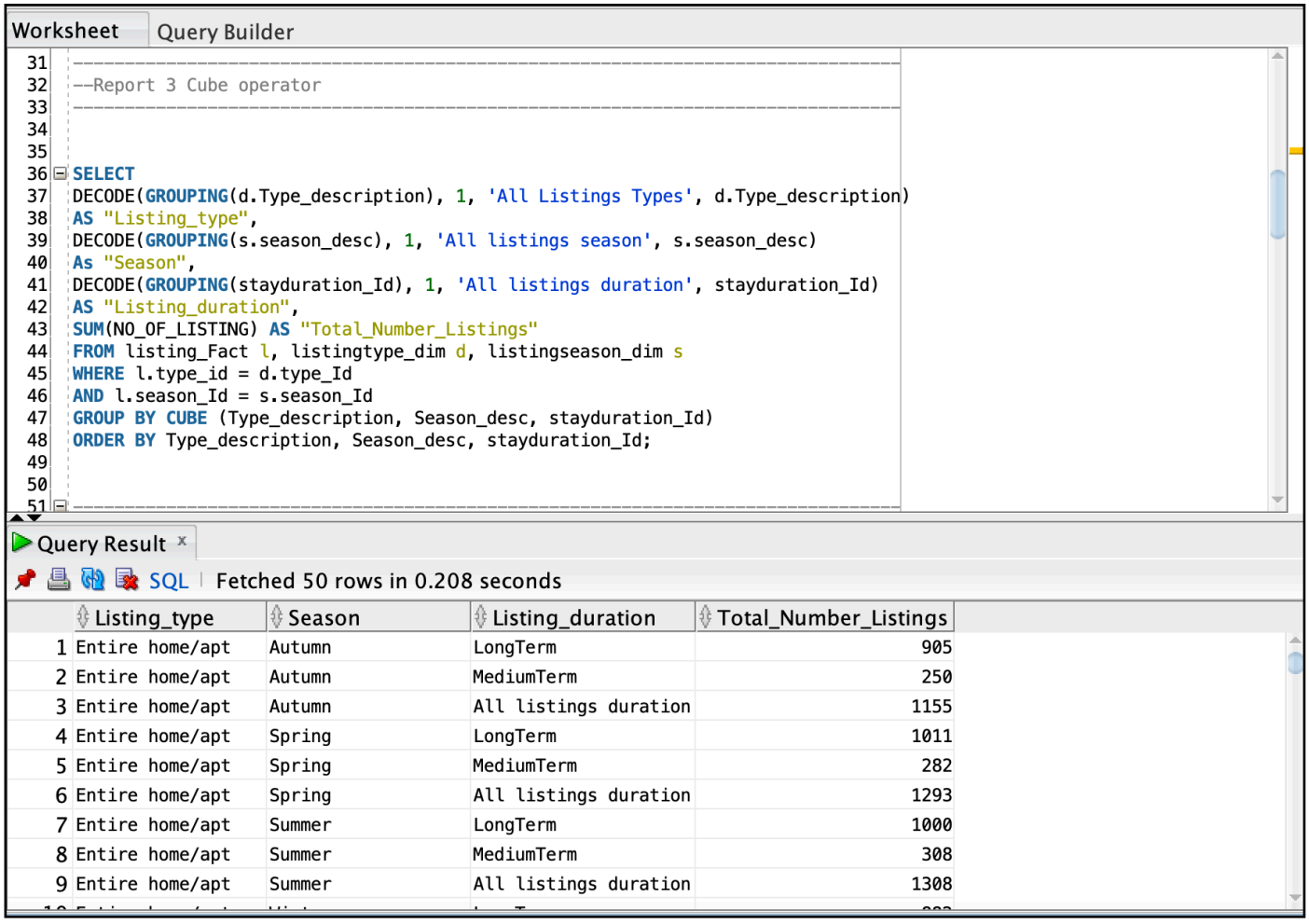
(b) Reports with proper sub-totals:

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



Worksheet | Query Builder

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

Query Result | SQL | Fetched 50 rows in 0.208 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	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  ------------------------------------------------------------------------------
52  --Report 3 Partial Cube Operator
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

SQL | 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

SQL | 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 ------------------------------------------------------------------------------
112 -- Partial ROLLUP operator
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

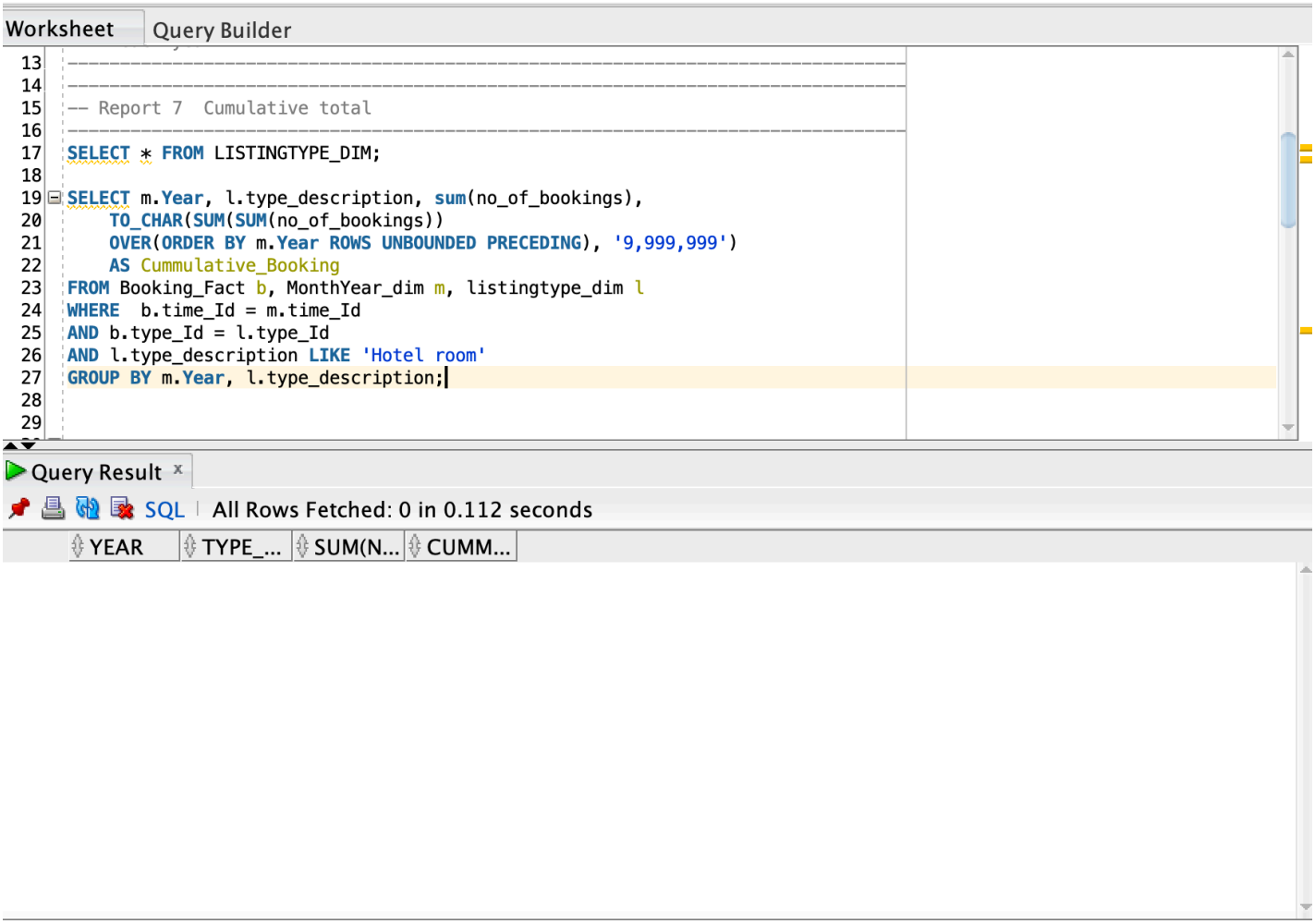
SQL | All Rows Fetched: 21 in 0.127 seconds

	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.



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;
```

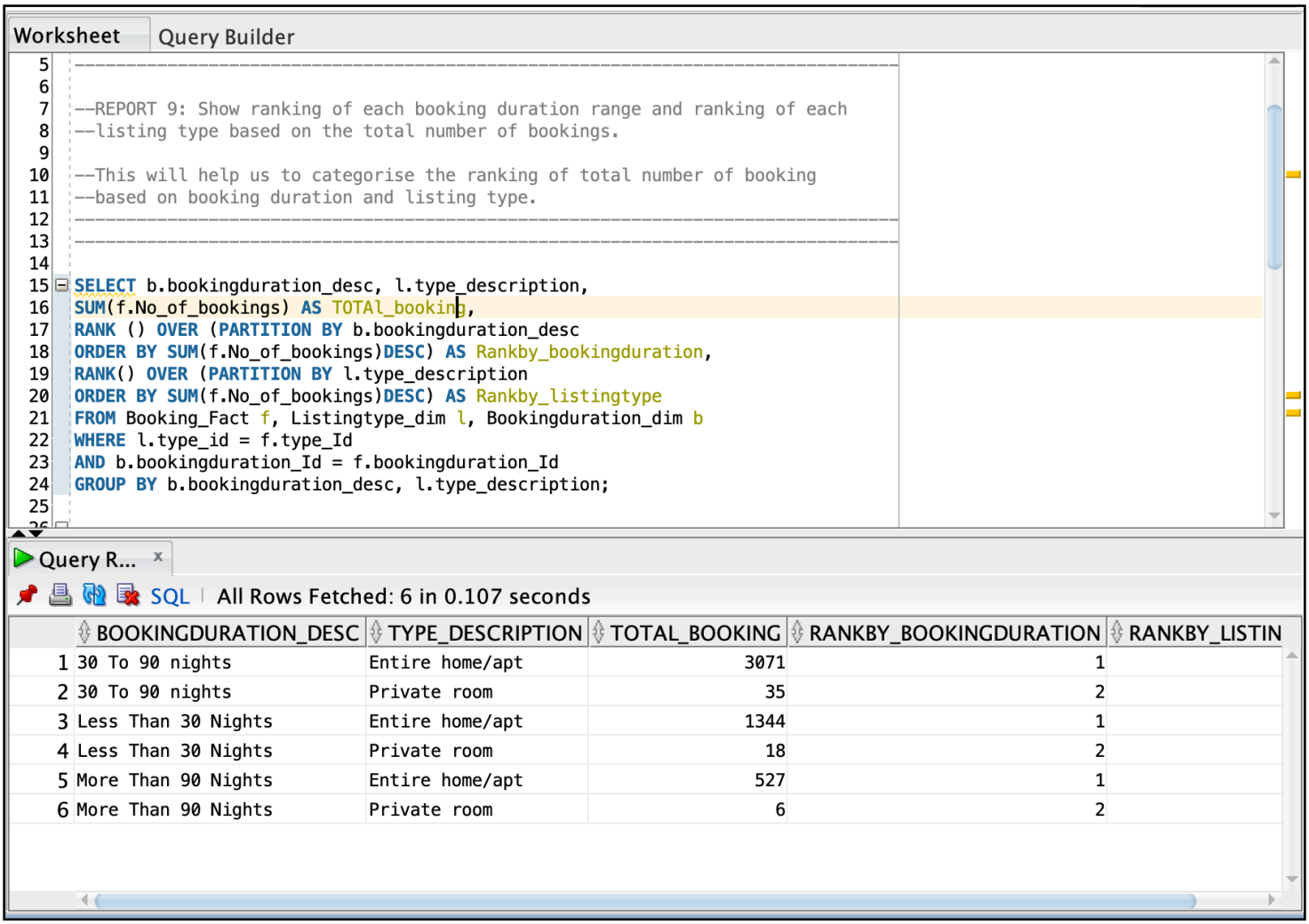
Query Result

SQL | 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.



	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;
53
54  ------------------------------------------------------------------------------
55  ------------------------------------------------------------------------------
```

Query Result

SQL | 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

SQL | All Rows Fetched: 1 in 0.123 seconds

	COUNT(*)
1	766