MONASH University

FIT3003 Assignment 2

Eunice Lee Wen Jing

33250979

Table of Contents

1. Transformation Stage

1.1 Preparation Stage: Data Cleaning

Data cleaning is a critical step in ensuring the accuracy and reliability of a database. This section outlines the strategies used during the data cleaning process, including relevant SQL commands and illustrative screenshots for each identified issue.

1.1.1 Identifying duplicates

I looked for duplicate records in key tables (e.g., Booking and Host tables) and removed them. Duplicates can lead to inflated counts and misinterpretations in analysis.

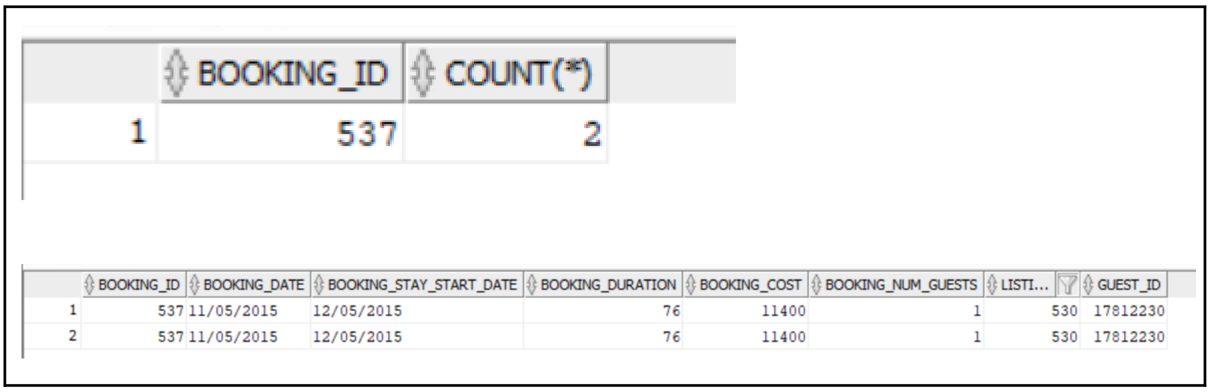
1.1.1A Duplicates in Booking Table

There are 2 duplicate booking_id entries in the booking table.

- Booking_id 537 appears twice.

Approach:

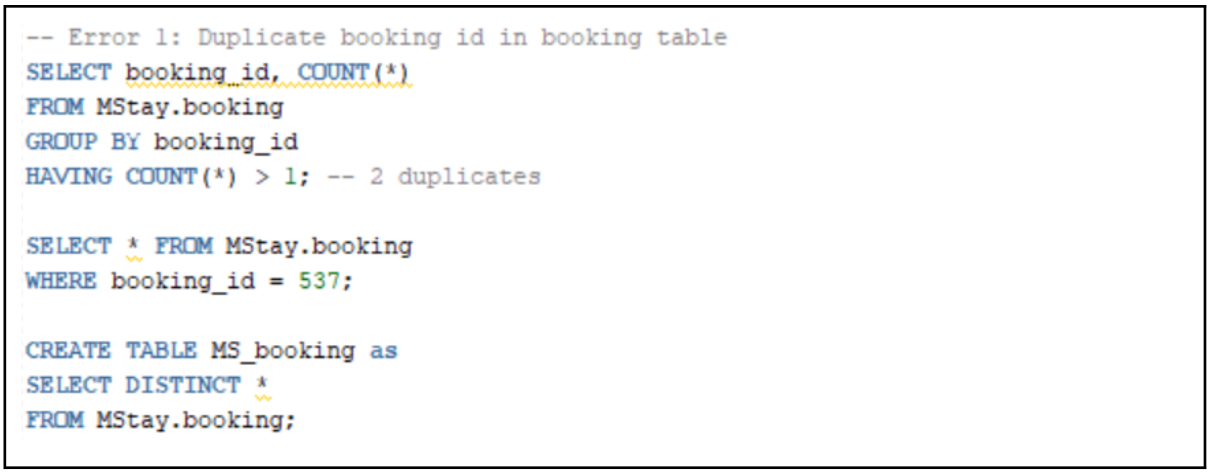
- **Identifying Duplicates**: I executed a SQL query to group the booking_id and count occurrences. This allowed me to identify which IDs had duplicates.
- **Cleaning Strategy**: To resolve this, I created a table containing distinct records from the booking table.



	BOOKING_ID	COUNT(*)
1	537	2

	BOOKING_ID	BOOKING_DATE	BOOKING_STAY_START_DATE	BOOKING_DURATION	BOOKING_COST	BOOKING_NUM_GUESTS	LISTI...	GUEST_ID
1	537	11/05/2015	12/05/2015	76	11400	1	530	17812230
2	537	11/05/2015	12/05/2015	76	11400	1	530	17812230

Figure 1.1.1A.1 Screenshot of Duplicate Booking IDs.

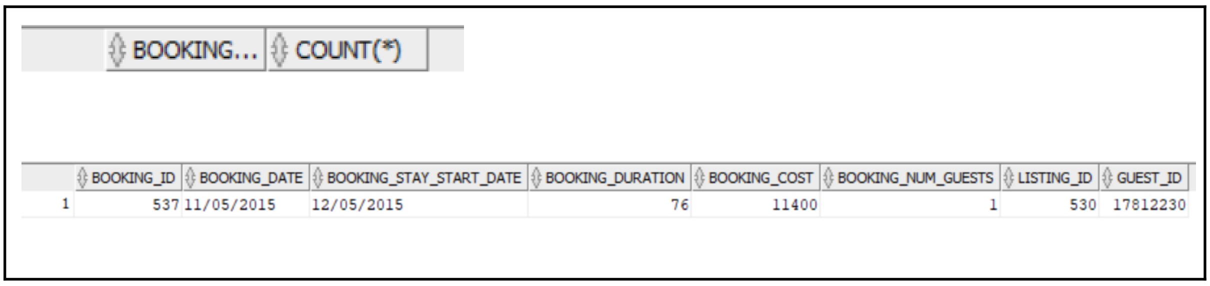


```sql
-- Error 1: Duplicate booking id in booking table
SELECT booking_id, COUNT(*)
FROM MStay.booking
GROUP BY booking_id
HAVING COUNT(*) > 1; -- 2 duplicates

SELECT * FROM MStay.booking
WHERE booking_id = 537;

CREATE TABLE MS_booking as
SELECT DISTINCT *
FROM MStay.booking;
```

Figure 1.1.1A.2 Screenshot of SQL Commands.



	BOOKING...	COUNT(*)

	BOOKING_ID	BOOKING_DATE	BOOKING_STAY_START_DATE	BOOKING_DURATION	BOOKING_COST	BOOKING_NUM_GUESTS	LISTING_ID	GUEST_ID
1	537	11/05/2015	12/05/2015	76	11400	1	530	17812230

Figure 1.1.1A.3 Screenshot after cleaning

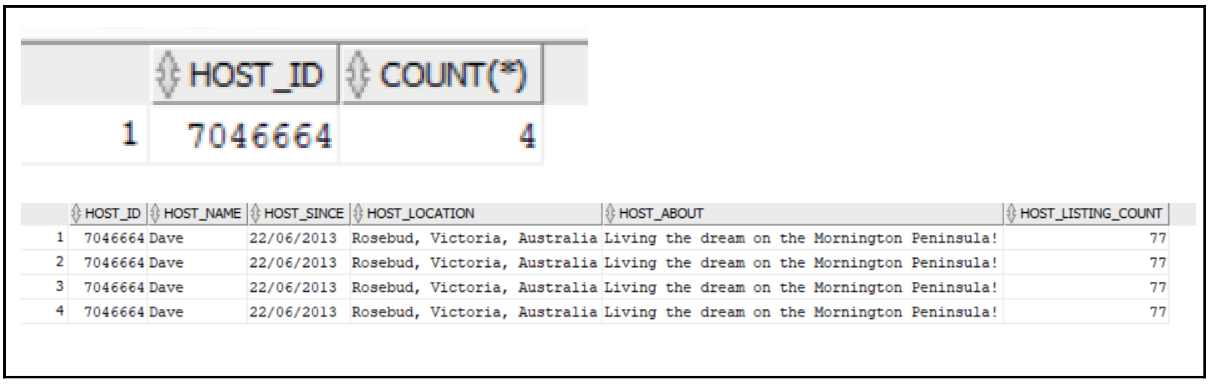
1.1.1B Duplicates in Host Table

There are 4 duplicate host_id entries.

- Host ID 7046664 is found 4 times in the host table.

Approach:

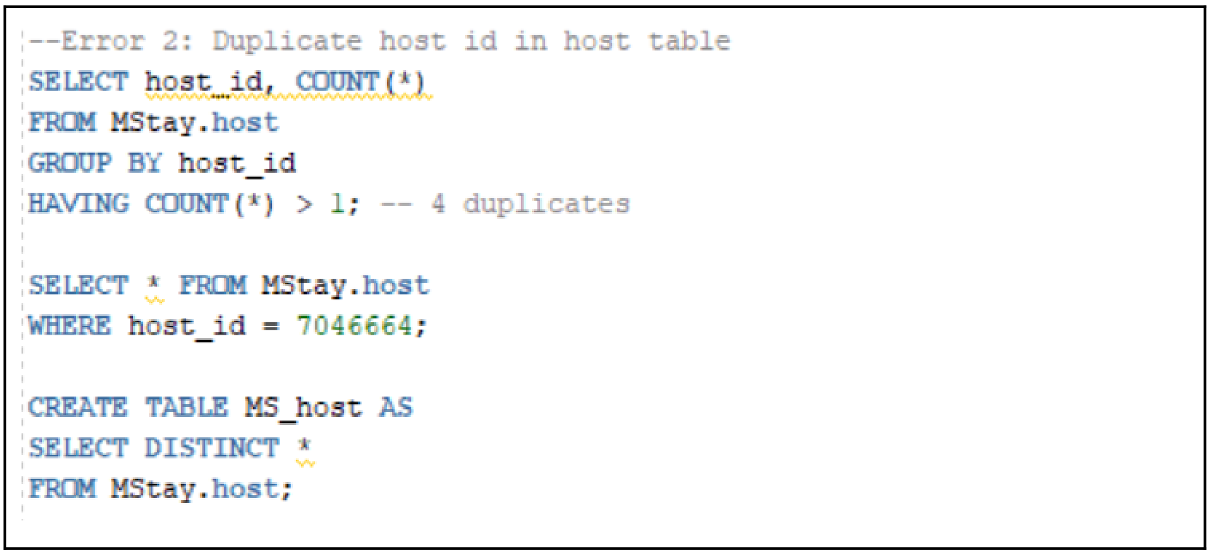
- **Identifying Duplicates**: Similar to the Booking table, I used a SQL query to identify duplicates based on host_id.
- **Cleaning Strategy**:To resolve this, I created a table containing distinct records from the host table.



	HOST_ID	COUNT(*)
1	7046664	4

	HOST_ID	HOST_NAME	HOST_SINCE	HOST_LOCATION	HOST_ABOUT	HOST_LISTING_COUNT
1	7046664	Dave	22/06/2013	Rosebud, Victoria, Australia	Living the dream on the Mornington Peninsula!	77
2	7046664	Dave	22/06/2013	Rosebud, Victoria, Australia	Living the dream on the Mornington Peninsula!	77
3	7046664	Dave	22/06/2013	Rosebud, Victoria, Australia	Living the dream on the Mornington Peninsula!	77
4	7046664	Dave	22/06/2013	Rosebud, Victoria, Australia	Living the dream on the Mornington Peninsula!	77

Figure 1.1.1B.1 Screenshot of Duplicate Host IDs

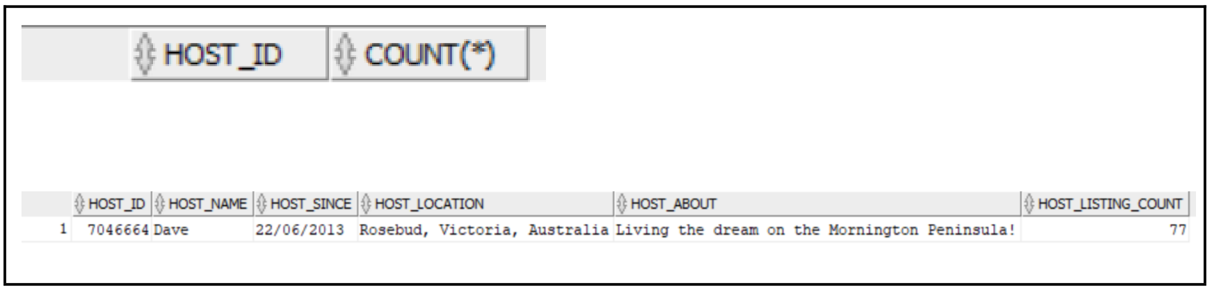


```
--Error 2: Duplicate host id in host table
SELECT host_id, COUNT(*)
FROM MStay.host
GROUP BY host_id
HAVING COUNT(*) > 1; -- 4 duplicates

SELECT * FROM MStay.host
WHERE host_id = 7046664;

CREATE TABLE MS_host AS
SELECT DISTINCT *
FROM MStay.host;
```

Figure 1.1.1B.2 Screenshot SQL Commands.



HOST_ID	COUNT(*)

	HOST_ID	HOST_NAME	HOST_SINCE	HOST_LOCATION	HOST_ABOUT	HOST_LISTING_COUNT
1	7046664	Dave	22/06/2013	Rosebud, Victoria, Australia	Living the dream on the Mornington Peninsula!	77

Figure 1.1.1B.3 Screenshot after cleaning

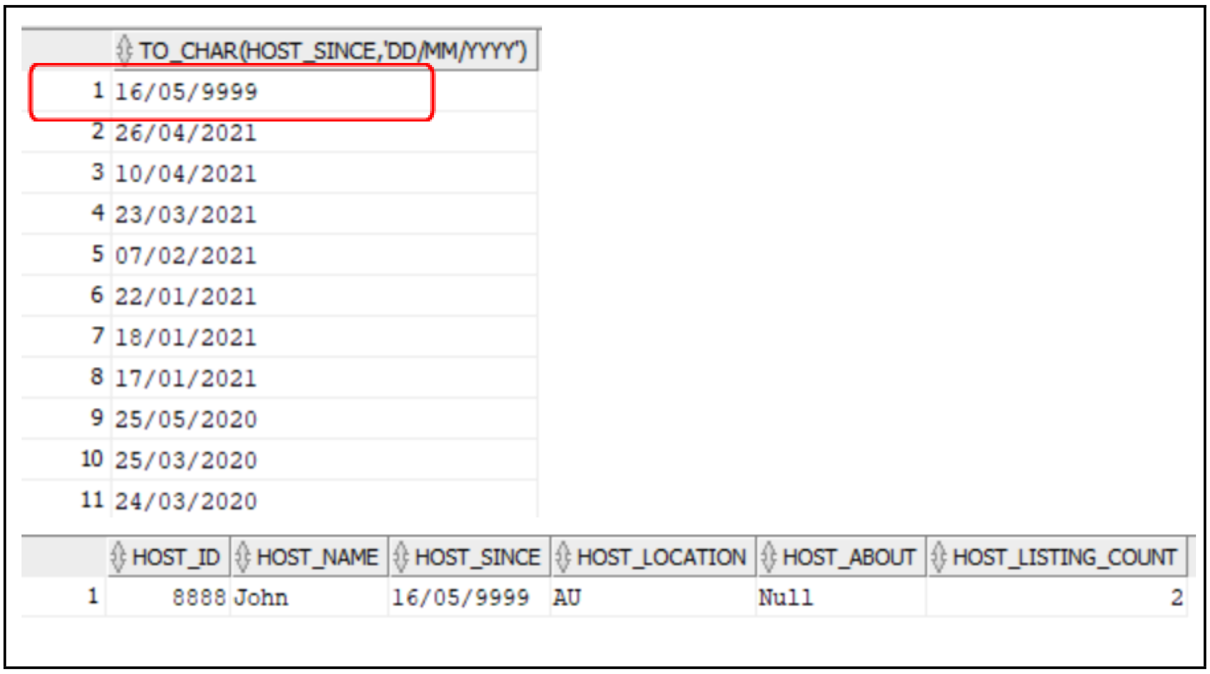
1.1.2 Deleting Error Date

An error date is found in the host_since column in Host table.:

- Host_since = 16/05/9999.

Approach:

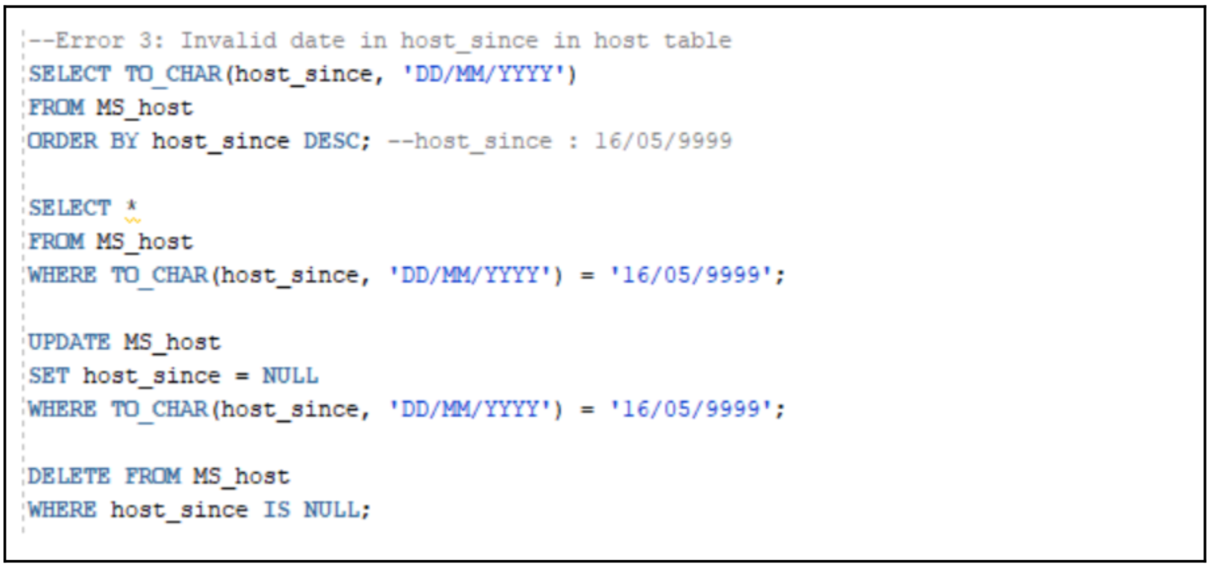
- **Identifying Error Dates**: I specifically searched for the unusual date to isolate problematic entries.
- **Cleaning Strategy**: I updated the erroneous date to null to remove the invalid data, then deleted records where host_since was null.



	TO_CHAR(HOST_SINCE,'DD/MM/YYYY')
1	16/05/9999
2	26/04/2021
3	10/04/2021
4	23/03/2021
5	07/02/2021
6	22/01/2021
7	18/01/2021
8	17/01/2021
9	25/05/2020
10	25/03/2020
11	24/03/2020

	HOST_ID	HOST_NAME	HOST_SINCE	HOST_LOCATION	HOST_ABOUT	HOST_LISTING_COUNT
1	8888	John	16/05/9999	AU	Null	2

Figure 1.1.2.1 Screenshot of Error Date in host table.



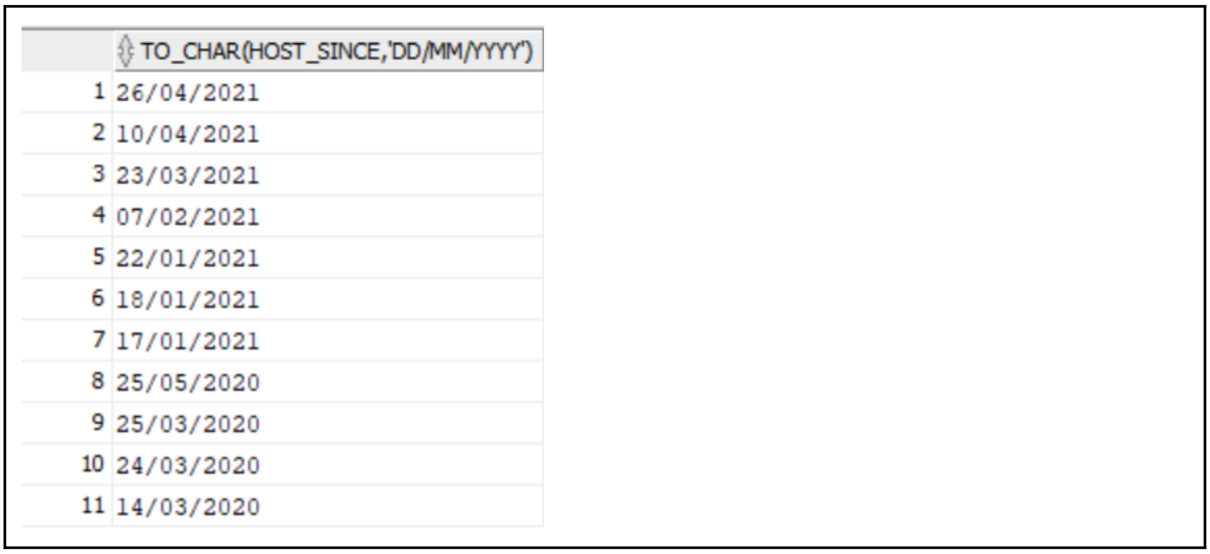
```
--Error 3: Invalid date in host_since in host table
SELECT TO_CHAR(host_since, 'DD/MM/YYYY')
FROM MS_host
ORDER BY host_since DESC; --host_since : 16/05/9999

SELECT *
FROM MS_host
WHERE TO_CHAR(host_since, 'DD/MM/YYYY') = '16/05/9999';

UPDATE MS_host
SET host_since = NULL
WHERE TO_CHAR(host_since, 'DD/MM/YYYY') = '16/05/9999';

DELETE FROM MS_host
WHERE host_since IS NULL;
```

Figure 1.1.2.2 Screenshot SQL Commands



	TO_CHAR(HOST_SINCE,'DD/MM/YYYY')
1	26/04/2021
2	10/04/2021
3	23/03/2021
4	07/02/2021
5	22/01/2021
6	18/01/2021
7	17/01/2021
8	25/05/2020
9	25/03/2020
10	24/03/2020
11	14/03/2020

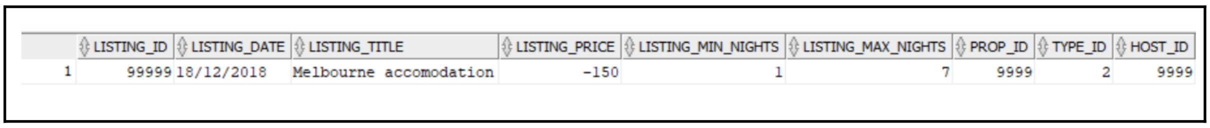
Figure 1.1.2.3 Screenshot after cleaning

1.1.3 Delete inaccurate value

A negative value (-150) is found in the listing_price column in the listing table.

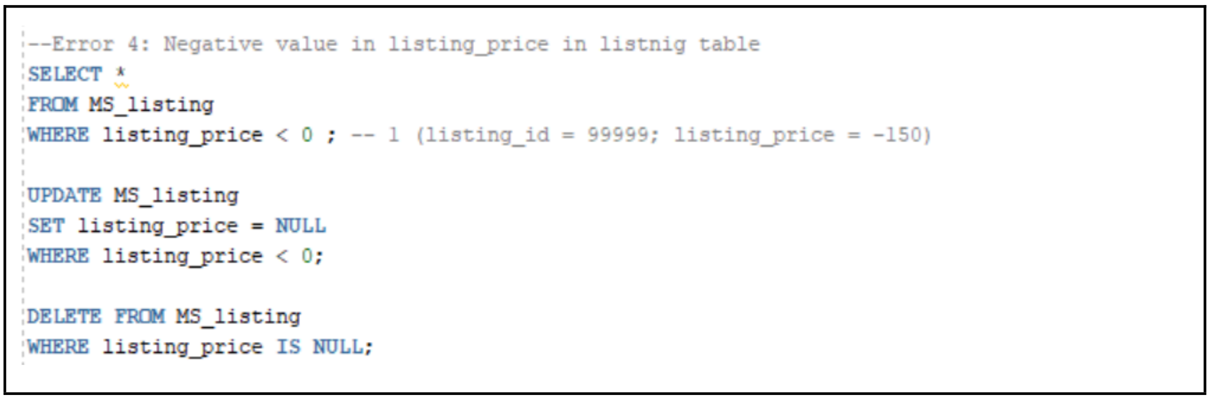
Approach:

- **Identifying Negative Values**: I ran a query to detect negative values in the listing_price field.
- **Cleaning Strategy**: I updated negative values to null, following which I deleted records with null prices.



	LISTING_ID	LISTING_DATE	LISTING_TITLE	LISTING_PRICE	LISTING_MIN_NIGHTS	LISTING_MAX_NIGHTS	PROP_ID	TYPE_ID	HOST_ID
1	99999	18/12/2018	Melbourne accomodation	-150	1	7	9999	2	9999

Figure 1.1.3.1 Screenshot of Negative listing_price in listing table

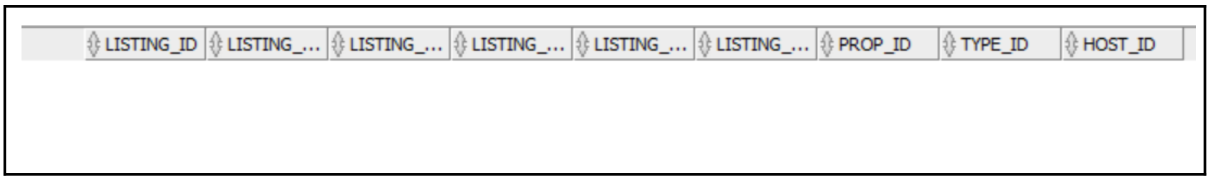


```
--Error 4: Negative value in listing_price in listnig table
SELECT *
FROM MS_listing
WHERE listing_price < 0 ; -- 1 (listing_id = 99999; listing_price = -150)

UPDATE MS_listing
SET listing_price = NULL
WHERE listing_price < 0;

DELETE FROM MS_listing
WHERE listing_price IS NULL;
```

Figure 1.1.3.2 Screenshot of SQL Commands.



	LISTING_ID	LISTING_...	LISTING_...	LISTING_...	LISTING_...	LISTING_...	PROP_ID	TYPE_ID	HOST_ID

Figure 1.1.3.3 Screenshot after cleaning

1.1.4 Delete invalid relationship

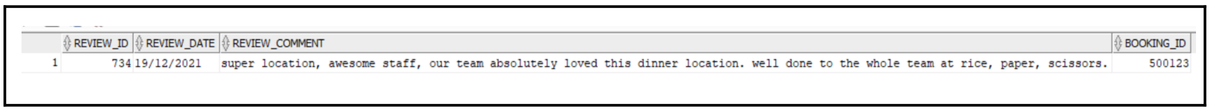
I addressed the issues where foreign keys in some tables did not have corresponding primary keys in their parent tables, leading to invalid relationships. The goal was to clean these inconsistencies and ensure that all foreign key references in child tables correctly matched with existing records in parent tables.

1.1.4A Invalid Relationship in Review Table

The booking_id foreign key in the review table (booking_id = 500123) does not exist in the booking table.

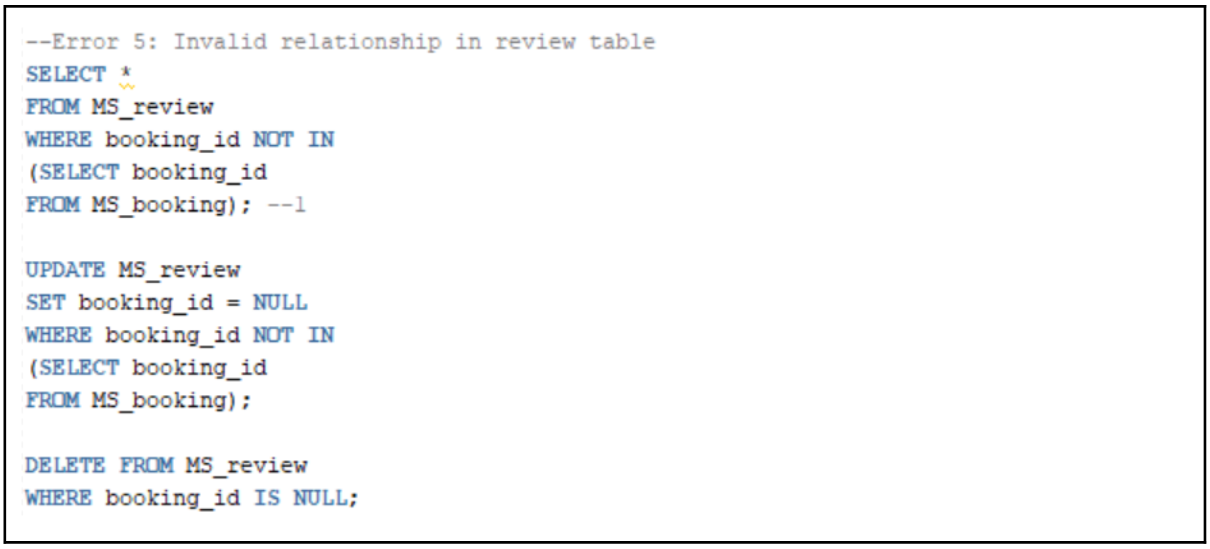
Approach:

- **Identifying Invalid Relationships**: I checked foreign keys against their corresponding primary keys to find mismatches.
- **Cleaning Strategy**: I set invalid booking_id to null, then deleted records where booking_id was null.



	REVIEW_ID	REVIEW_DATE	REVIEW_COMMENT	BOOKING_ID
1	734	19/12/2021	super location, awesome staff, our team absolutely loved this dinner location. well done to the whole team at rice, paper, scissors.	500123

Figure 1.1.4A.1 Screenshot of the invalid relationship in review table.

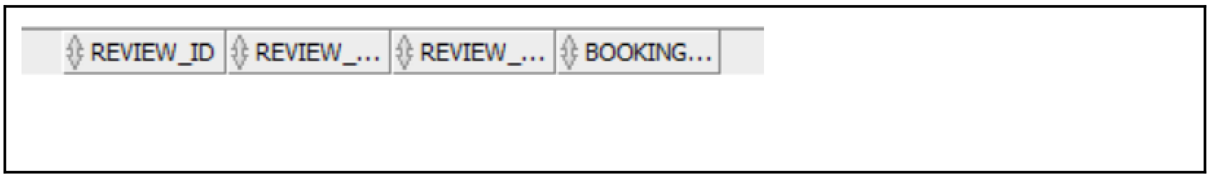


```
--Error 5: Invalid relationship in review table
SELECT *
FROM MS_review
WHERE booking_id NOT IN
(SELECT booking_id
FROM MS_booking); --1

UPDATE MS_review
SET booking_id = NULL
WHERE booking_id NOT IN
(SELECT booking_id
FROM MS_booking);

DELETE FROM MS_review
WHERE booking_id IS NULL;
```

Figure 1.1.4A.2 Screenshot of SQL Commands



	REVIEW_ID	REVIEW_...	REVIEW_...	BOOKING...

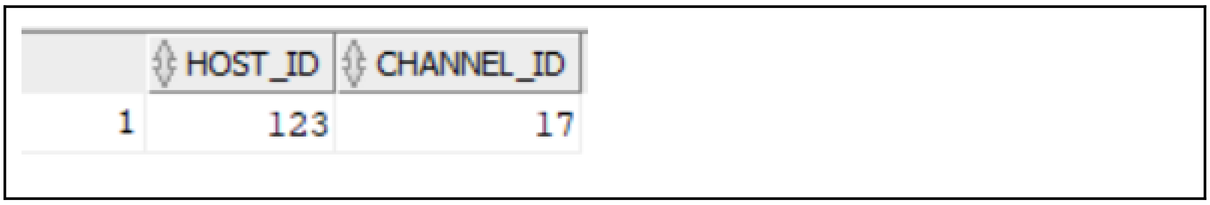
Figure 1.1.4A.3 Screenshot after cleaning

1.1.4B Invalid Relationship in Host Verification Table

The host_id foreign key (host_id = 123) in the host verification table does not exist in the host table.

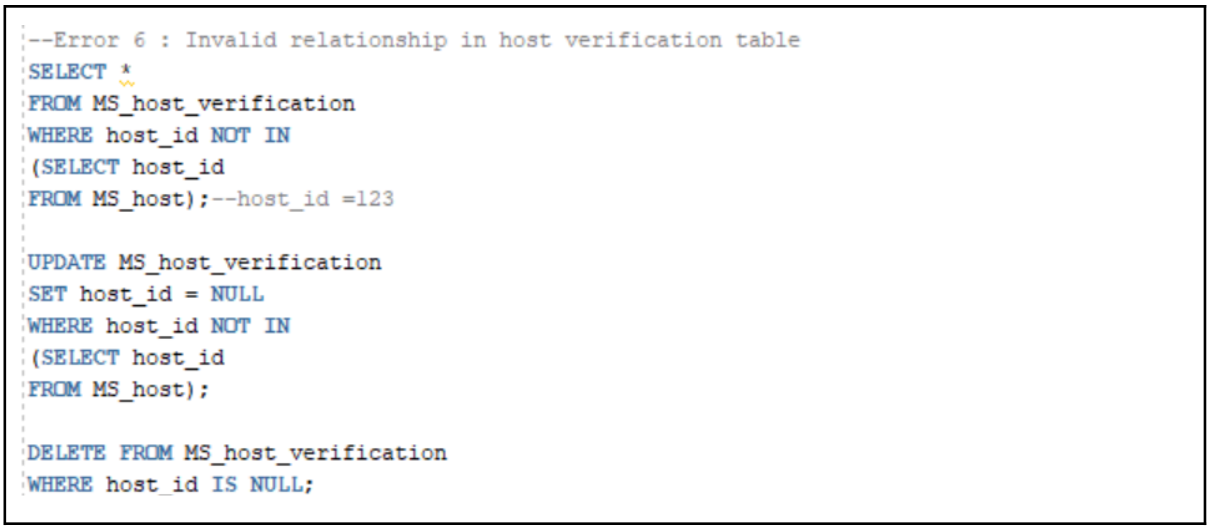
Approach:

- **Identifying Invalid Relationships**: I performed similar checks for the Host Verification table.
- **Cleaning Strategy**: I updated invalid host_id to null, then deleted those records.



	HOST_ID	CHANNEL_ID
1	123	17

Figure 1.1.4B.1 Screenshot of the invalid relationship in Host Verification table.



```
--Error 6 : Invalid relationship in host verification table
SELECT *
FROM MS_host_verification
WHERE host_id NOT IN
(SELECT host_id
FROM MS_host);--host_id =123

UPDATE MS_host_verification
SET host_id = NULL
WHERE host_id NOT IN
(SELECT host_id
FROM MS_host);

DELETE FROM MS_host_verification
WHERE host_id IS NULL;
```

Figure 1.1.4B.2 Screenshot of SQL Commands.

HOST_ID	CHANNEL...

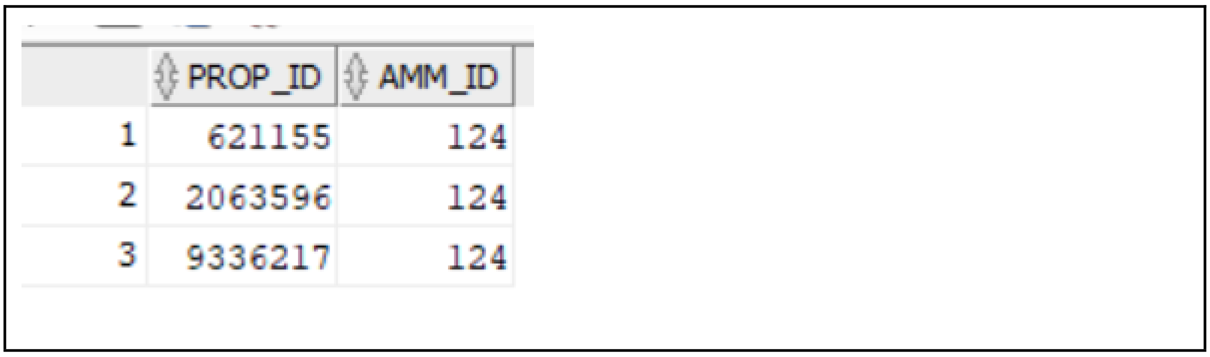
Figure 1.1.4B.3 Screenshot after cleaning

1.1.4C Invalid Relationship in Property Amenity Table

After deleting null values [1.1.5B] , there are 3 rows of amm_id (amm_id = 124) in the property amenity table that do not exist in the amenity table.

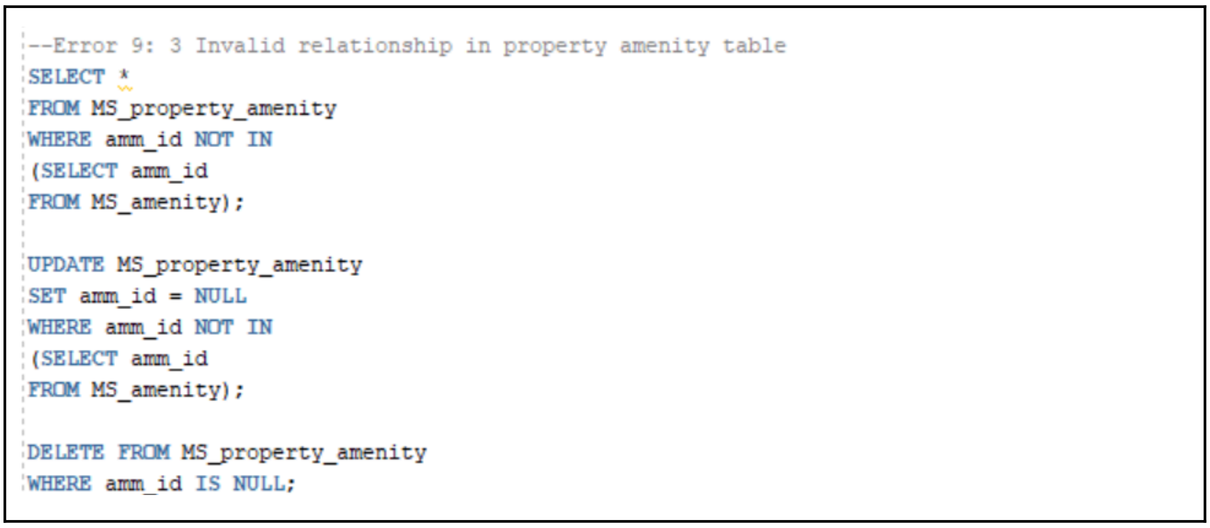
Approach:

- **Identifying Invalid Relationships**: I checked for amm_id that were not present in the Amenity table.
- **Cleaning Strategy**: I updated invalid amm_id to null and then deleted those entries.



	PROP_ID	AMM_ID
1	621155	124
2	2063596	124
3	9336217	124

Figure 1.1.4C.1 Screenshot of the invalid relationship in Property Amenity table.



```
--Error 9: 3 Invalid relationship in property amenity table
SELECT *
FROM MS_property_amenity
WHERE amm_id NOT IN
(SELECT amm_id
FROM MS_amenity);

UPDATE MS_property_amenity
SET amm_id = NULL
WHERE amm_id NOT IN
(SELECT amm_id
FROM MS_amenity);

DELETE FROM MS_property_amenity
WHERE amm_id IS NULL;
```

Figure 1.1.4C.2 Screenshot of SQL Commands.

Figure 1.1.4C.3 Screenshot after cleaning

1.1.5 Delete Null Values

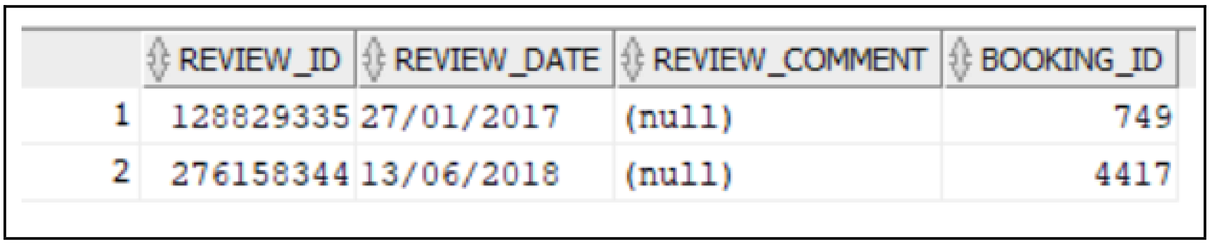
I identified the issue of NULL values in key columns of the dataset. Null values can affect analysis and reporting by introducing gaps in the data.

1.1.5A Null Values in Review Table

There are 2 null values in the review_comment column.

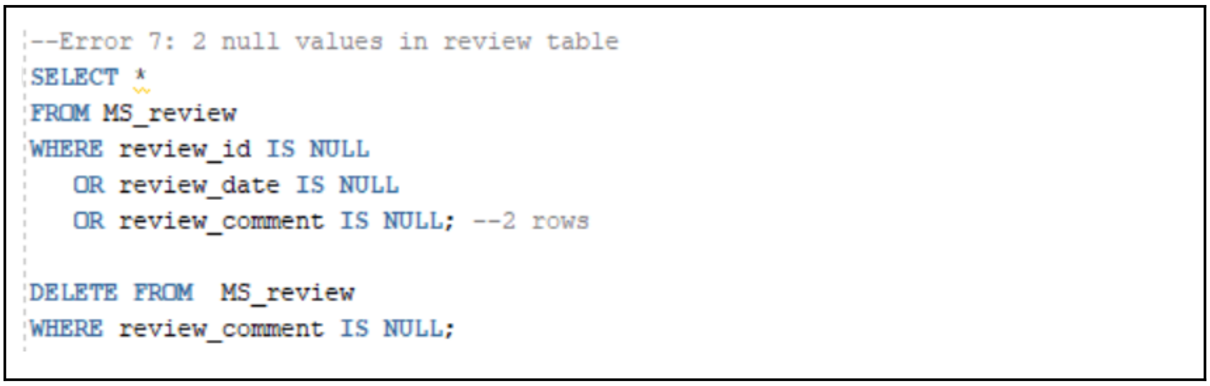
Approach:

- **Identifying Null Values**: I queried the Review table to check for null entries in review_comment.
- **Cleaning Strategy**: I directly deleted records with null comments.



	REVIEW_ID	REVIEW_DATE	REVIEW_COMMENT	BOOKING_ID
1	128829335	27/01/2017	(null)	749
2	276158344	13/06/2018	(null)	4417

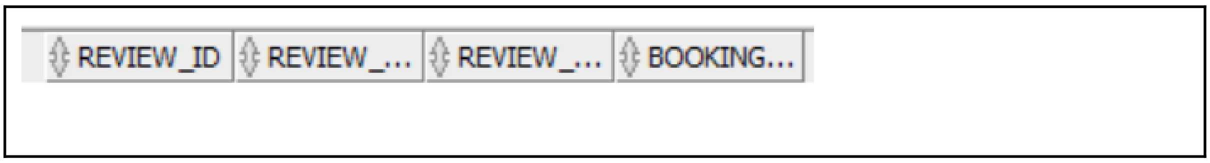
Figure 1.1.5A.1 Screenshot of null values in Review Table.



```
--Error 7: 2 null values in review table
SELECT *
FROM MS_review
WHERE review_id IS NULL
   OR review_date IS NULL
   OR review_comment IS NULL; --2 rows

DELETE FROM  MS_review
WHERE review_comment IS NULL;
```

Figure 1.1.5A.2 Screenshot of SQL Commands.



REVIEW_ID	REVIEW_...	REVIEW_...	BOOKING...

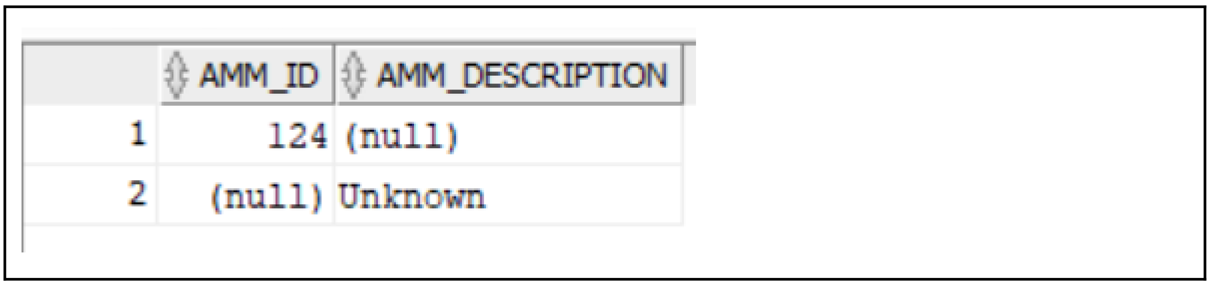
Figure 1.1.5A.3 Screenshot after cleaning.

1.1.5B Null Values in Amenity Table

There is 1 null value in amm_description and 1 null value in amm_id.

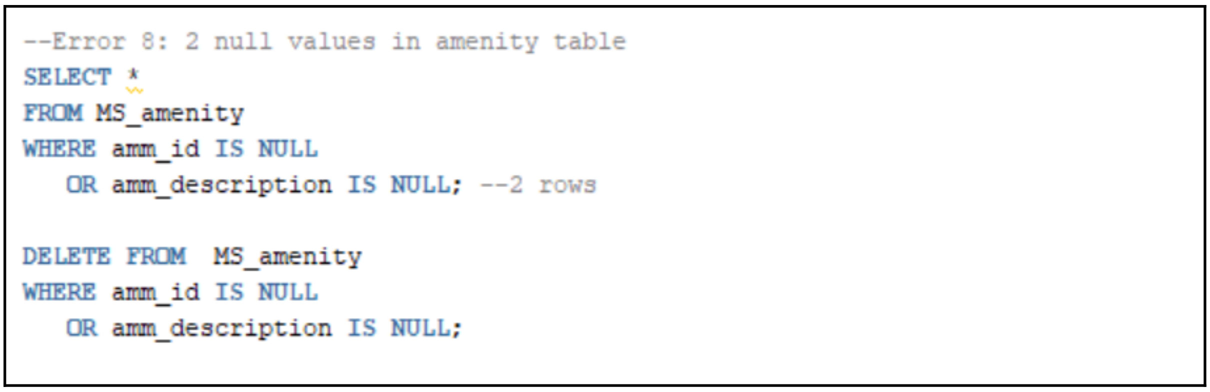
Approach:

- **Identifying Null Values**: I checked the Amenity table for null entries in both fields.
- **Cleaning Strategy**: I executed delete commands for records with null values.



	AMM_ID	AMM_DESCRIPTION
1	124	(null)
2	(null)	Unknown

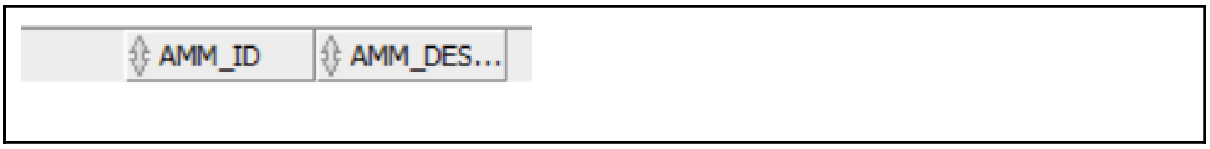
Figure 1.1.5B.1 Screenshot of null values in Amenity Table.



```
--Error 8: 2 null values in amenity table
SELECT *
FROM MS_amenity
WHERE amm_id IS NULL
   OR amm_description IS NULL; --2 rows

DELETE FROM  MS_amenity
WHERE amm_id IS NULL
   OR amm_description IS NULL;
```

Figure 1.1.5B.2 Screenshot of SQL Commands.



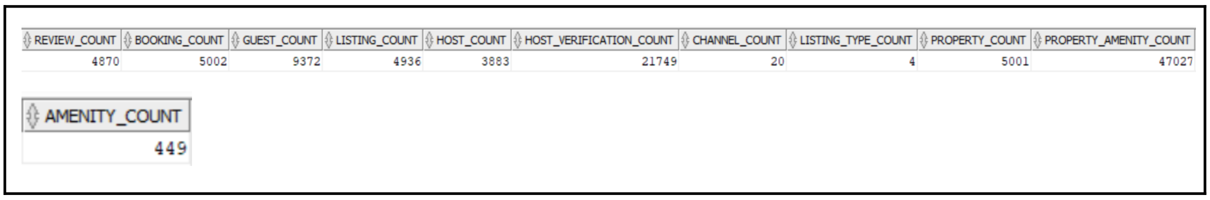
AMM_ID	AMM_DES...

Figure 1.1.5B.3 Screenshot after cleaning

1.1.6 Before and after cleaning

Number of rows for every tables before data cleaning :

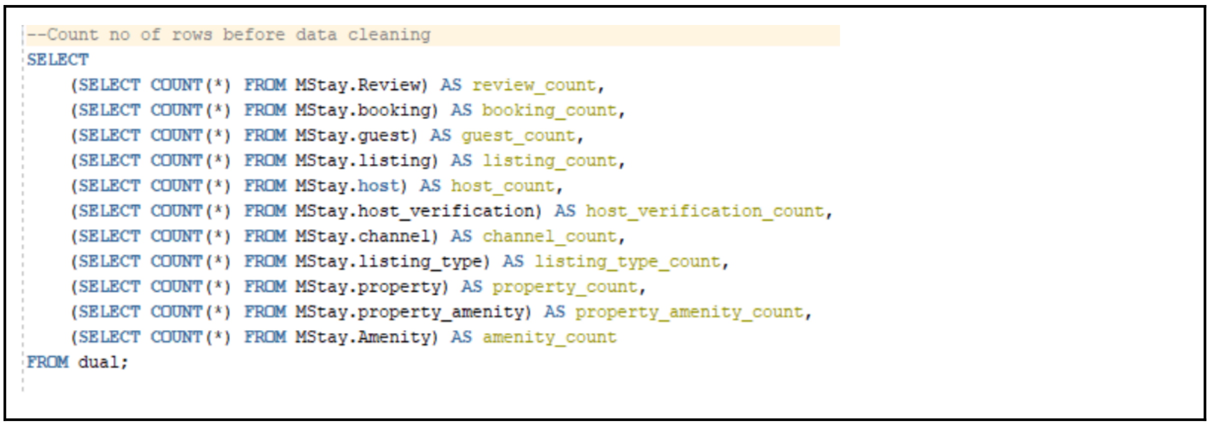
- Review Table : 4870 rows
- Booking Table : 5002 rows
- Guest Table : 9372 rows
- Listing Table : 4936 rows
- Host Table : 3883 rows
- Host Verification Table : 21749 rows
- Channel Table : 20 rows
- Listing Type Table : 4
- Property Table : 5001 rows
- Property Amenity Table : 47027 rows
- Amenity Table : 449 rows



REVIEW_COUNT	BOOKING_COUNT	GUEST_COUNT	LISTING_COUNT	HOST_COUNT	HOST_VERIFICATION_COUNT	CHANNEL_COUNT	LISTING_TYPE_COUNT	PROPERTY_COUNT	PROPERTY_AMENITY_COUNT
4870	5002	9372	4936	3883	21749	20	4	5001	47027

AMENITY_COUNT
449

Figure 1.1.6.1 Screenshots of Data Before Cleaning

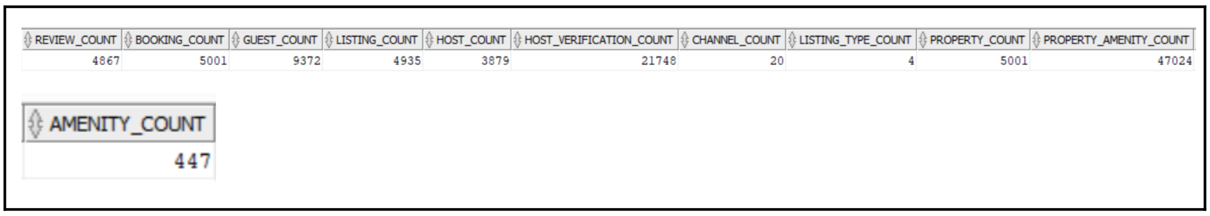


```sql
--Count no of rows before data cleaning
SELECT
    (SELECT COUNT(*) FROM MStay.Review) AS review_count,
    (SELECT COUNT(*) FROM MStay.booking) AS booking_count,
    (SELECT COUNT(*) FROM MStay.guest) AS guest_count,
    (SELECT COUNT(*) FROM MStay.listing) AS listing_count,
    (SELECT COUNT(*) FROM MStay.host) AS host_count,
    (SELECT COUNT(*) FROM MStay.host_verification) AS host_verification_count,
    (SELECT COUNT(*) FROM MStay.channel) AS channel_count,
    (SELECT COUNT(*) FROM MStay.listing_type) AS listing_type_count,
    (SELECT COUNT(*) FROM MStay.property) AS property_count,
    (SELECT COUNT(*) FROM MStay.property_amenity) AS property_amenity_count,
    (SELECT COUNT(*) FROM MStay.Amenity) AS amenity_count
FROM dual;
```

Figure 1.1.6.2 Screenshot of SQL Commands.

Number of rows from every tables after data cleaning :

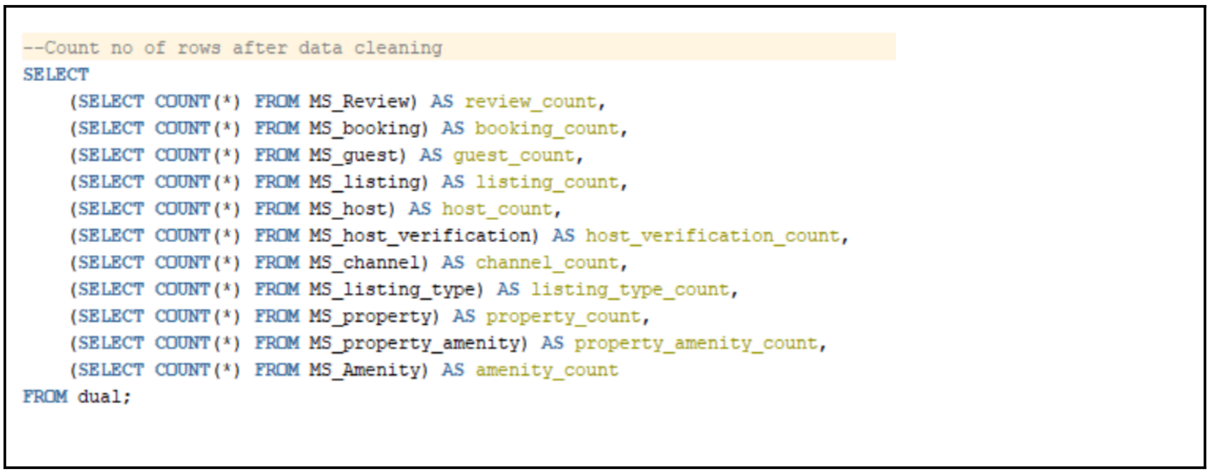
- *Review Table : 4867 rows*
- *Booking Table : 5001 rows*
- Guest Table : 9372 rows
- *Listing Table : 4935 rows*
- *Host Table : 3879 rows*
- Host Verification Table : 21749 rows
- Channel Table : 20 rows
- Listing Type Table : 4
- Property Table : 5001 rows
- *Property Amenity Table : 47024 rows*
- *Amenity Table : 447 rows*



REVIEW_COUNT	BOOKING_COUNT	GUEST_COUNT	LISTING_COUNT	HOST_COUNT	HOST_VERIFICATION_COUNT	CHANNEL_COUNT	LISTING_TYPE_COUNT	PROPERTY_COUNT	PROPERTY_AMENITY_COUNT
4867	5001	9372	4935	3879	21748	20	4	5001	47024

AMENITY_COUNT
447

Figure 1.1.6.3 Screenshots of Data After Cleaning



```
--Count no of rows after data cleaning
SELECT
    (SELECT COUNT(*) FROM MS_Review) AS review_count,
    (SELECT COUNT(*) FROM MS_booking) AS booking_count,
    (SELECT COUNT(*) FROM MS_guest) AS guest_count,
    (SELECT COUNT(*) FROM MS_listing) AS listing_count,
    (SELECT COUNT(*) FROM MS_host) AS host_count,
    (SELECT COUNT(*) FROM MS_host_verification) AS host_verification_count,
    (SELECT COUNT(*) FROM MS_channel) AS channel_count,
    (SELECT COUNT(*) FROM MS_listing_type) AS listing_type_count,
    (SELECT COUNT(*) FROM MS_property) AS property_count,
    (SELECT COUNT(*) FROM MS_property_amenity) AS property_amenity_count,
    (SELECT COUNT(*) FROM MS_Amenity) AS amenity_count
FROM dual;
```

Figure 1.1.6.4 Screenshot of SQL Commands.

1.2 Design Task A: Star/Snowflakes Schema Diagrams.

The star schema in Figure 1.2 is a multi-fact star schema, consisting of three fact tables: ReviewFACT, BookingFACT, and ListingFACT. All three fact tables are connected to TimeDIM. Both BookingFACT and ListingFACT are connected to ListingTypeDIM. Additionally, BookingFACT is connected to BookingDurationDIM and BookingCostDIM, while ListingFACT is connected to ListingSeasonDIM, ListingDurationDIM, ListingPriceDIM, and HostDIM. HostDIM has a bridge table, HostChannelBRIDGE, which links it to ChannelDIM. The ReviewFACT table contains one fact measure, *No_of_Reviews*; BookingFACT includes two fact measures, *Total_No_of_Bookings* and *Total_Booking_Cost*; and ListingFACT contains one fact measure, *No_of_Listings.*

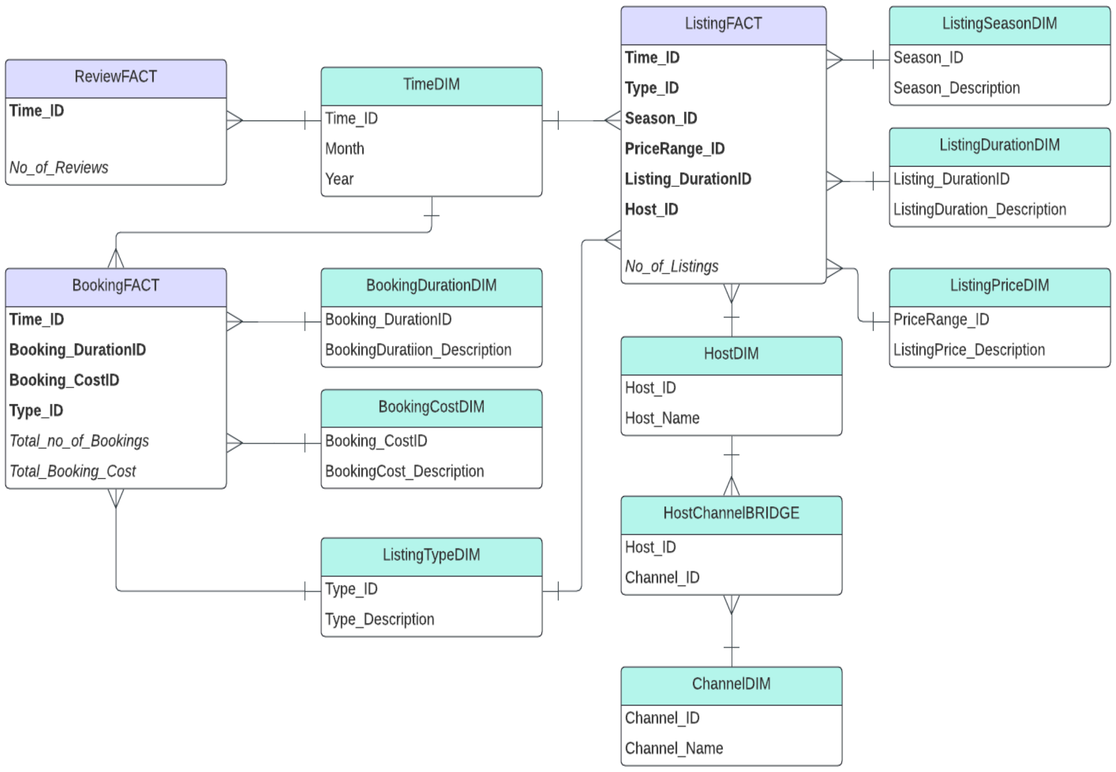


Figure 1.2 Star Schema Design

1.3 Design Task B: Star Schema with Increased Granularity.

In the star schema presented in Figure 1.3, there is an increased level of granularity compared to the schema in Figure 1.2. The key distinction between the two is that Figure 1.2 represents a highly aggregated schema, while Figure 1.3 is not aggregated.

In this new star schema, three additional dimension tables have been introduced: **ReviewDIM** (linked to ReviewFACT), **BookingDIM** (linked to BookingFACT), and **ListingDIM** (linked to ListingFACT). Several dimension tables from Figure 1.2, including BookingDurationDIM, BookingCostDIM, ListingSeasonDIM, ListingDurationDIM, and ListingPriceDIM, have been removed in Figure 1.3. Instead, **Listing_Max_Nights** replaces Listing_DurationID, and **Listing_Price** replaces PriceRange_ID. Additionally, TimeDIM has been removed, and date attributes have been moved into the newly created dimension tables: **Review_Date in ReviewDIM**, **Booking_Date in BookingDIM**, and **Listing_Date in ListingDIM**.

In summary, the schema in Figure 1.2 has a high level of aggregation, which supports simplified and generalised analysis with lower granularity. In contrast, the schema in Figure 1.3 has a lower level of aggregation, providing more detailed and specific analysis, and therefore exhibits higher granularity.

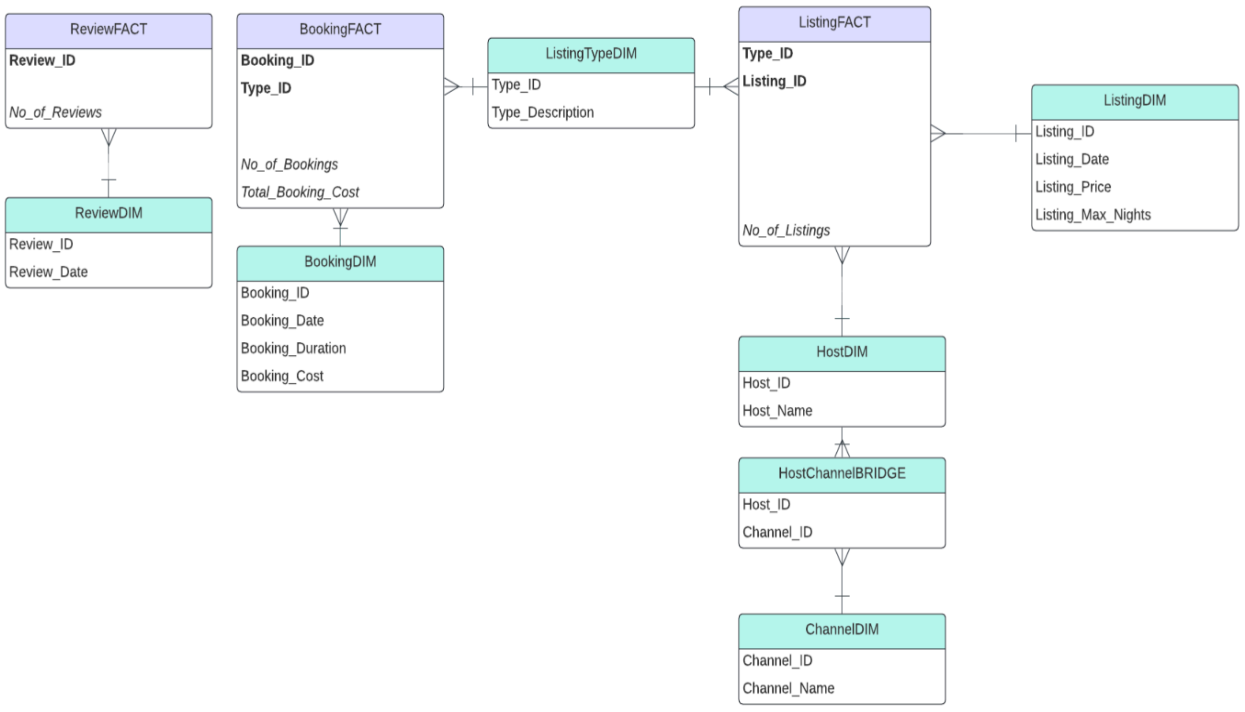
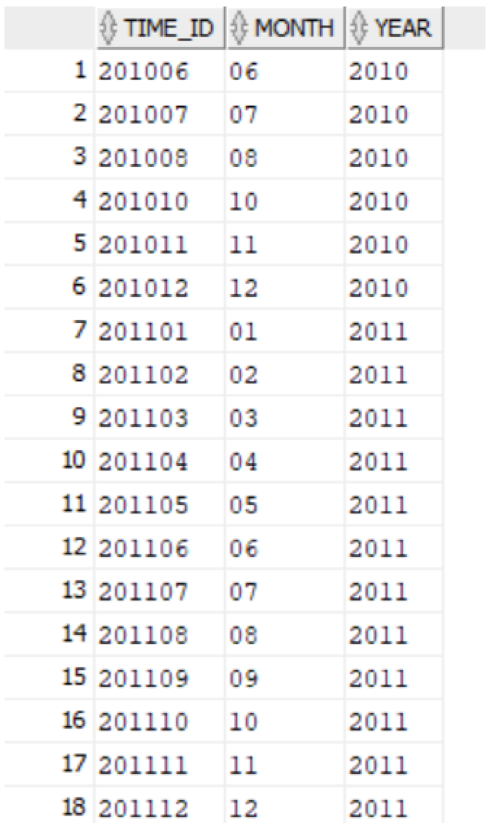


Figure 1.3 Star Schema Design with increase granularity

1.4 Implement Star/Snowflakes Schema using SQL.

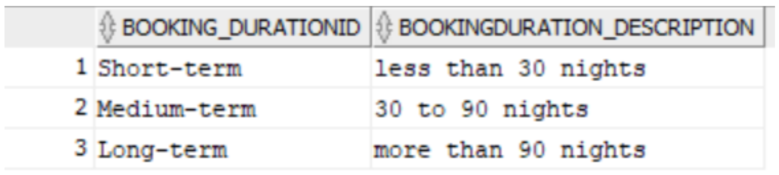
Screenshots of dimension tables.

TimeDIM Table :



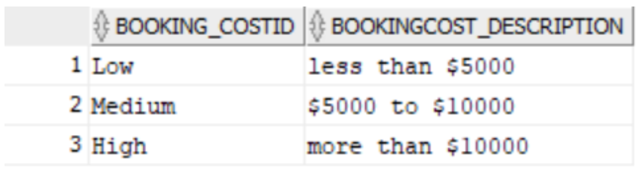
	TIME_ID	MONTH	YEAR
1	201006	06	2010
2	201007	07	2010
3	201008	08	2010
4	201010	10	2010
5	201011	11	2010
6	201012	12	2010
7	201101	01	2011
8	201102	02	2011
9	201103	03	2011
10	201104	04	2011
11	201105	05	2011
12	201106	06	2011
13	201107	07	2011
14	201108	08	2011
15	201109	09	2011
16	201110	10	2011
17	201111	11	2011
18	201112	12	2011

BookingDurationDIM Table :



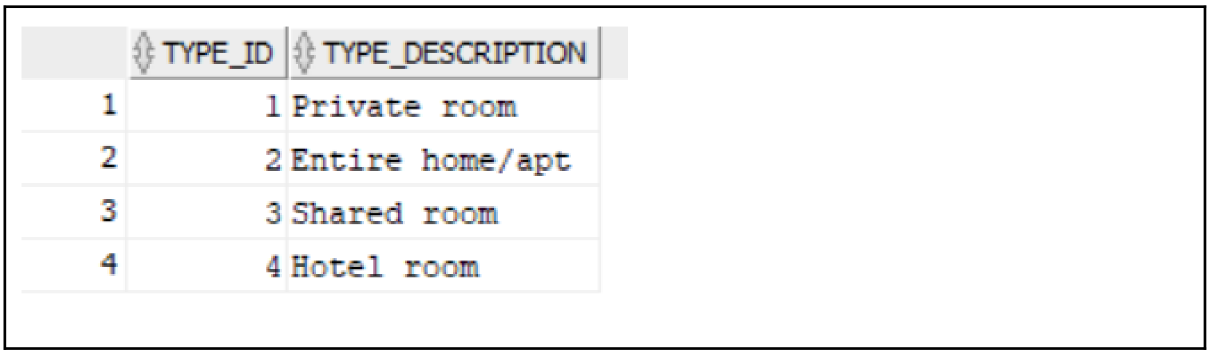
	BOOKING_DURATIONID	BOOKINGDURATION_DESCRIPTION
1	Short-term	less than 30 nights
2	Medium-term	30 to 90 nights
3	Long-term	more than 90 nights

BookingCostDIM Table :



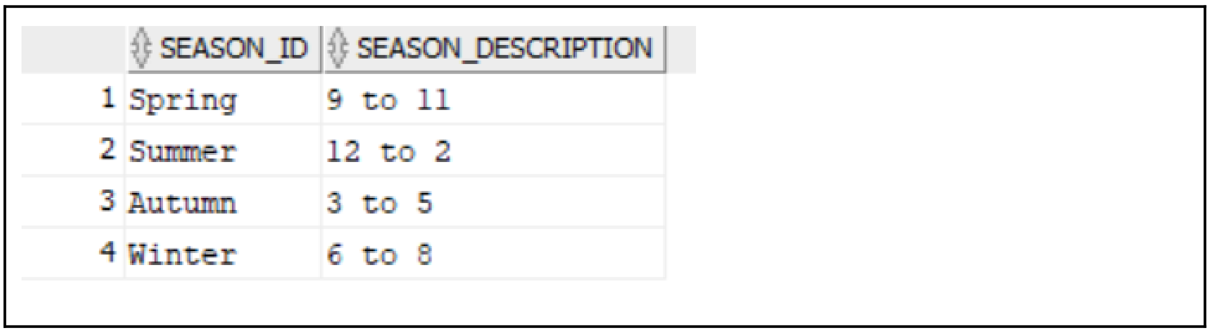
	BOOKING_COSTID	BOOKINGCOST_DESCRIPTION
1	Low	less than $5000
2	Medium	$5000 to $10000
3	High	more than $10000

ListingTypeDIM Table :



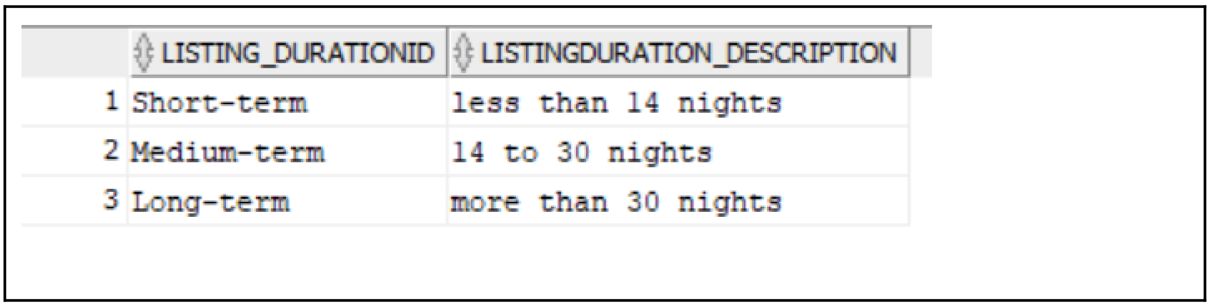
	TYPE_ID	TYPE_DESCRIPTION
1	1	Private room
2	2	Entire home/apt
3	3	Shared room
4	4	Hotel room

ListingSeasonDIM Table :



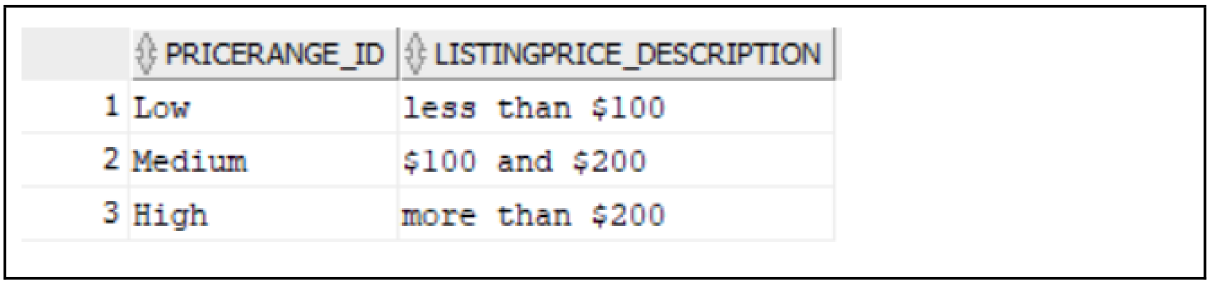
	SEASON_ID	SEASON_DESCRIPTION
1	Spring	9 to 11
2	Summer	12 to 2
3	Autumn	3 to 5
4	Winter	6 to 8

ListingDurationDIM Table :



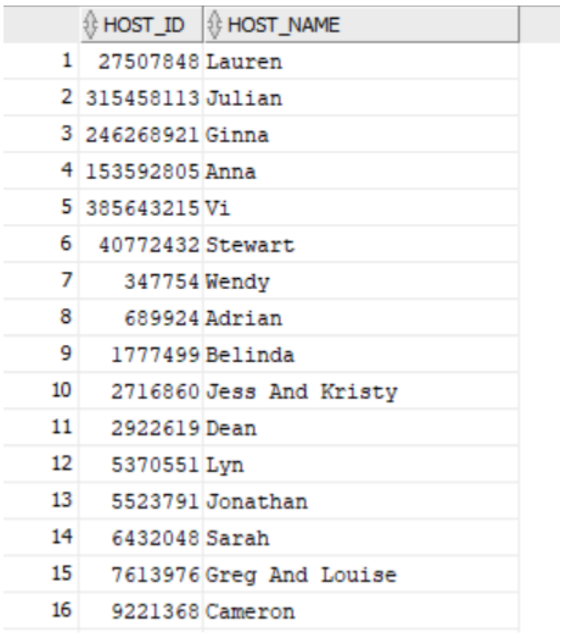
	LISTING_DURATIONID	LISTINGDURATION_DESCRIPTION
1	Short-term	less than 14 nights
2	Medium-term	14 to 30 nights
3	Long-term	more than 30 nights

ListingPriceDIM Table :



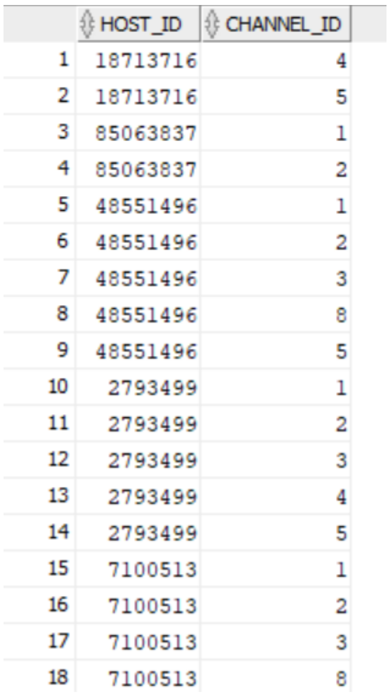
	PRICERANGE_ID	LISTINGPRICE_DESCRIPTION
1	Low	less than $100
2	Medium	$100 and $200
3	High	more than $200

HostDIM Table :



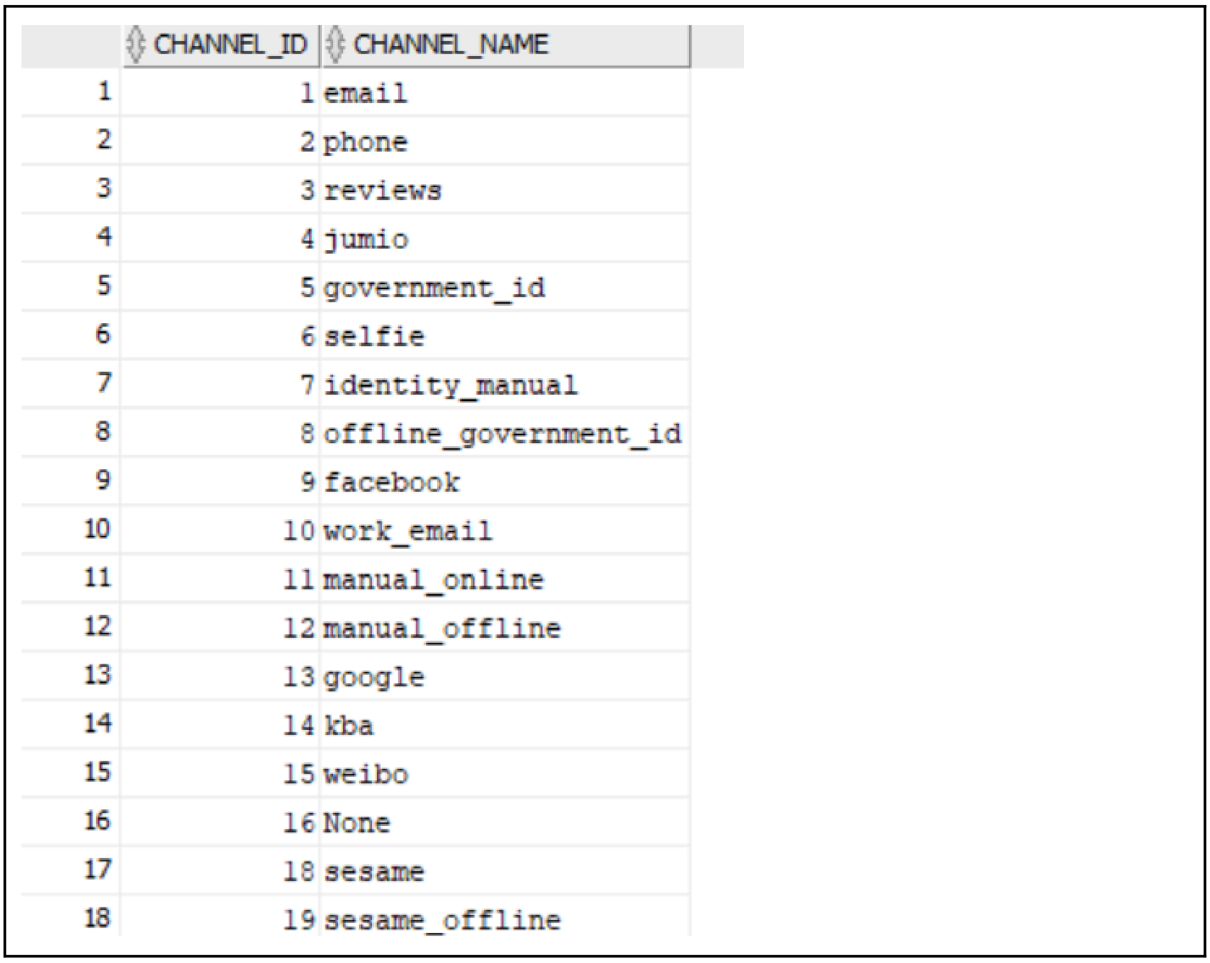
	HOST_ID	HOST_NAME
1	27507848	Lauren
2	315458113	Julian
3	246268921	Ginna
4	153592805	Anna
5	385643215	Vi
6	40772432	Stewart
7	347754	Wendy
8	689924	Adrian
9	1777499	Belinda
10	2716860	Jess And Kristy
11	2922619	Dean
12	5370551	Lyn
13	5523791	Jonathan
14	6432048	Sarah
15	7613976	Greg And Louise
16	9221368	Cameron

HostChannelBRIDGE Table :



	HOST_ID	CHANNEL_ID
1	18713716	4
2	18713716	5
3	85063837	1
4	85063837	2
5	48551496	1
6	48551496	2
7	48551496	3
8	48551496	8
9	48551496	5
10	2793499	1
11	2793499	2
12	2793499	3
13	2793499	4
14	2793499	5
15	7100513	1
16	7100513	2
17	7100513	3
18	7100513	8

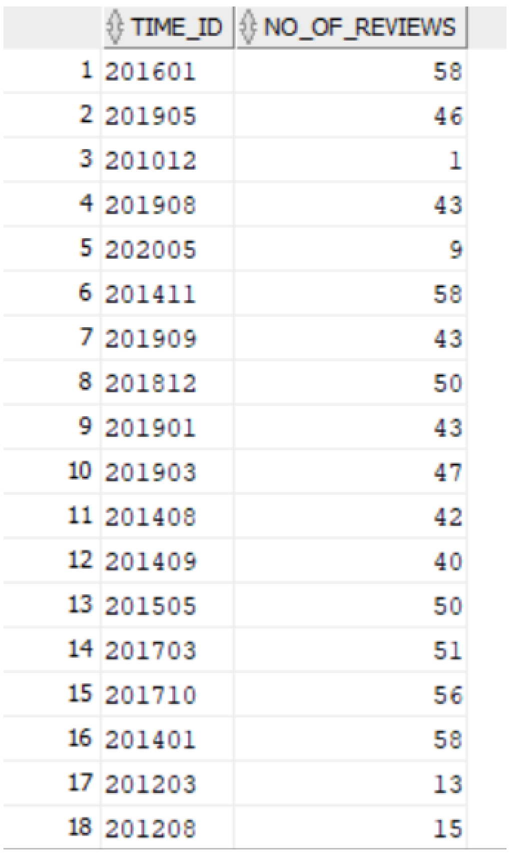
ChannelDIM Table :



	CHANNEL_ID	CHANNEL_NAME
1	1	email
2	2	phone
3	3	reviews
4	4	jumio
5	5	government_id
6	6	selfie
7	7	identity_manual
8	8	offline_government_id
9	9	facebook
10	10	work_email
11	11	manual_online
12	12	manual_offline
13	13	google
14	14	kba
15	15	weibo
16	16	None
17	18	sesame
18	19	sesame_offline

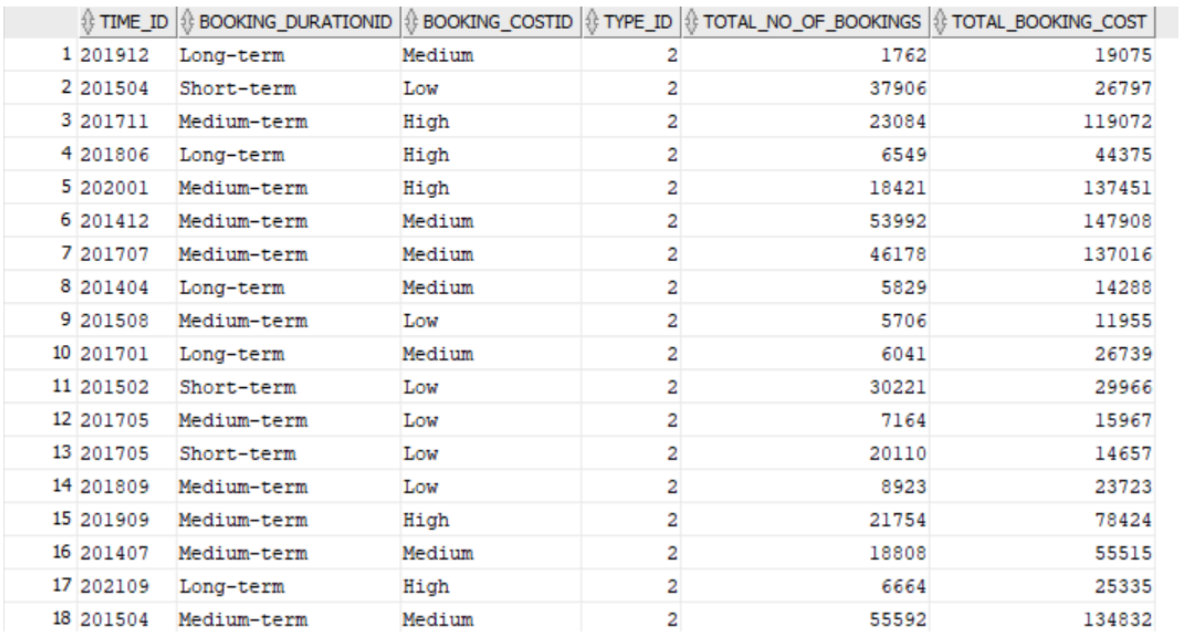
Screenshots of fact tables.

ReviewFACT Table :



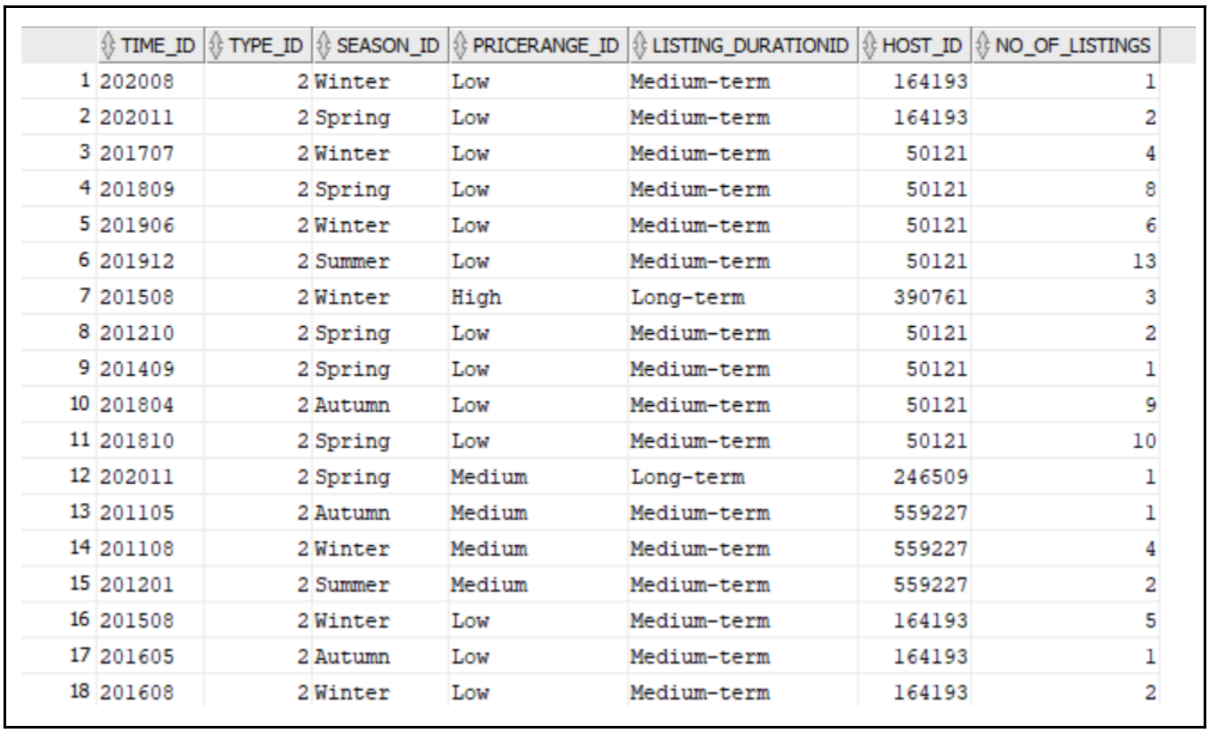
	TIME_ID	NO_OF_REVIEWS
1	201601	58
2	201905	46
3	201012	1
4	201908	43
5	202005	9
6	201411	58
7	201909	43
8	201812	50
9	201901	43
10	201903	47
11	201408	42
12	201409	40
13	201505	50
14	201703	51
15	201710	56
16	201401	58
17	201203	13
18	201208	15

BookingFACT Table :



	TIME_ID	BOOKING_DURATIONID	BOOKING_COSTID	TYPE_ID	TOTAL_NO_OF_BOOKINGS	TOTAL_BOOKING_COST
1	201912	Long-term	Medium	2	1762	19075
2	201504	Short-term	Low	2	37906	26797
3	201711	Medium-term	High	2	23084	119072
4	201806	Long-term	High	2	6549	44375
5	202001	Medium-term	High	2	18421	137451
6	201412	Medium-term	Medium	2	53992	147908
7	201707	Medium-term	Medium	2	46178	137016
8	201404	Long-term	Medium	2	5829	14288
9	201508	Medium-term	Low	2	5706	11955
10	201701	Long-term	Medium	2	6041	26739
11	201502	Short-term	Low	2	30221	29966
12	201705	Medium-term	Low	2	7164	15967
13	201705	Short-term	Low	2	20110	14657
14	201809	Medium-term	Low	2	8923	23723
15	201909	Medium-term	High	2	21754	78424
16	201407	Medium-term	Medium	2	18808	55515
17	202109	Long-term	High	2	6664	25335
18	201504	Medium-term	Medium	2	55592	134832

ListingFACT Table :

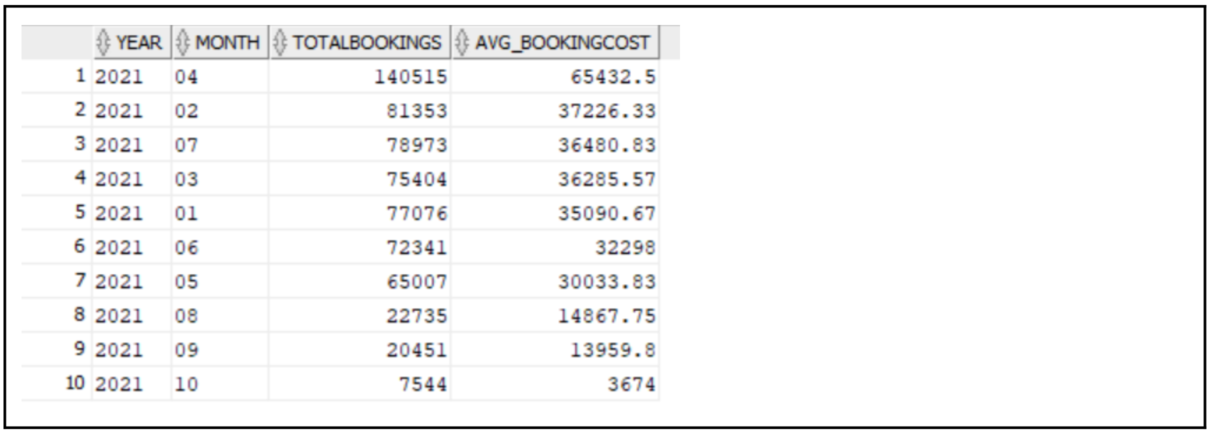


	TIME_ID	TYPE_ID	SEASON_ID	PRICERANGE_ID	LISTING_DURATIONID	HOST_ID	NO_OF_LISTINGS
1	202008	2	Winter	Low	Medium-term	164193	1
2	202011	2	Spring	Low	Medium-term	164193	2
3	201707	2	Winter	Low	Medium-term	50121	4
4	201809	2	Spring	Low	Medium-term	50121	8
5	201906	2	Winter	Low	Medium-term	50121	6
6	201912	2	Summer	Low	Medium-term	50121	13
7	201508	2	Winter	High	Long-term	390761	3
8	201210	2	Spring	Low	Medium-term	50121	2
9	201409	2	Spring	Low	Medium-term	50121	1
10	201804	2	Autumn	Low	Medium-term	50121	9
11	201810	2	Spring	Low	Medium-term	50121	10
12	202011	2	Spring	Medium	Long-term	246509	1
13	201105	2	Autumn	Medium	Medium-term	559227	1
14	201108	2	Winter	Medium	Medium-term	559227	4
15	201201	2	Summer	Medium	Medium-term	559227	2
16	201508	2	Winter	Low	Medium-term	164193	5
17	201605	2	Autumn	Low	Medium-term	164193	1
18	201608	2	Winter	Low	Medium-term	164193	2

2. Data Analytic Stage

2.1 Total number of Bookings and the Average Booking Cost for each Month in 2021

The analysis below presents the booking trends in M-Stay throughout 2021, showing the total number of bookings and the average cost per booking for each month. The month with the **highest number of bookings is April**, with 140,515 bookings, while the lowest is October, with only 7,544 bookings. October also recorded the lowest average booking cost at $3,674. In contrast, **February** had the **highest average booking cost** of $37,226.33, with 81,353 bookings during that month.

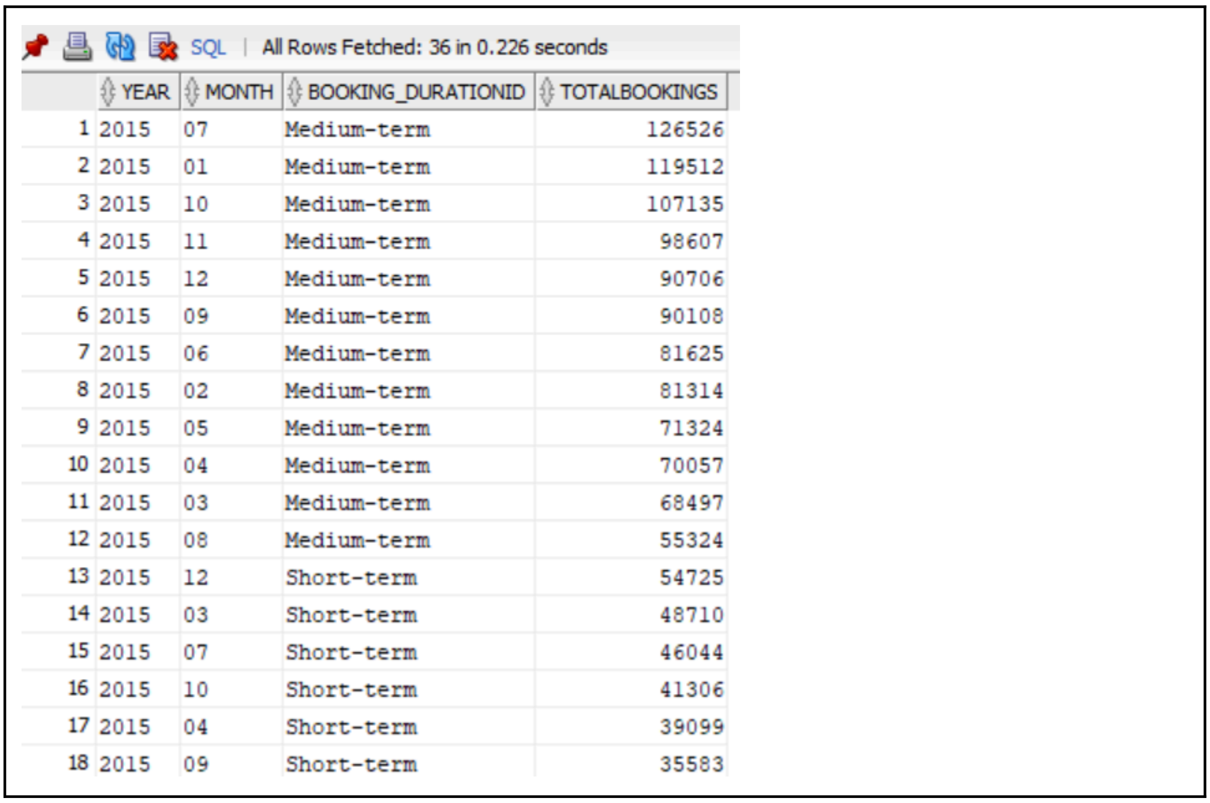


	YEAR	MONTH	TOTALBOOKINGS	AVG_BOOKINGCOST
1	2021	04	140515	65432.5
2	2021	02	81353	37226.33
3	2021	07	78973	36480.83
4	2021	03	75404	36285.57
5	2021	01	77076	35090.67
6	2021	06	72341	32298
7	2021	05	65007	30033.83
8	2021	08	22735	14867.75
9	2021	09	20451	13959.8
10	2021	10	7544	3674

Figure 2.1.1 Screenshot of the results.

2.2 Total number of bookings for each Booking Duration type in 2015

The findings below show the total number of bookings classified by different booking duration types—short-term, medium-term, and long-term for the year 2015. The “**Medium-term**” category, representing bookings lasting 30 to 90 nights, was the most popular among the customers, as the top 12 results fall within this category.



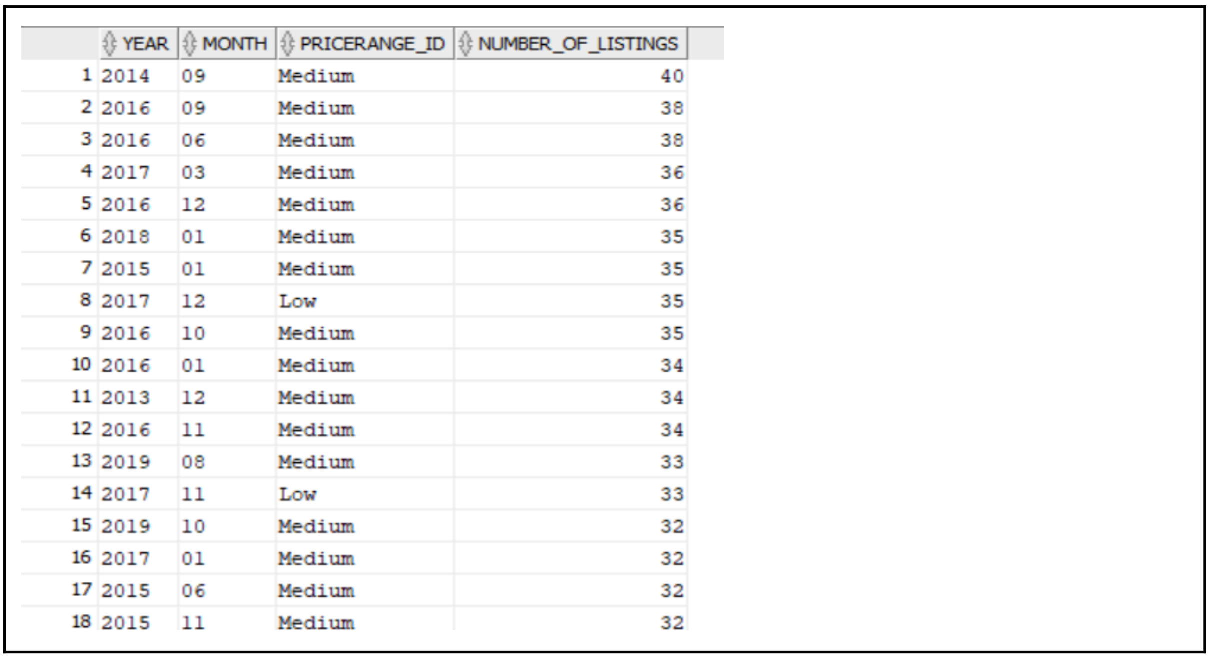
SQL | All Rows Fetched: 36 in 0.226 seconds

	YEAR	MONTH	BOOKING_DURATIONID	TOTALBOOKINGS
1	2015	07	Medium-term	126526
2	2015	01	Medium-term	119512
3	2015	10	Medium-term	107135
4	2015	11	Medium-term	98607
5	2015	12	Medium-term	90706
6	2015	09	Medium-term	90108
7	2015	06	Medium-term	81625
8	2015	02	Medium-term	81314
9	2015	05	Medium-term	71324
10	2015	04	Medium-term	70057
11	2015	03	Medium-term	68497
12	2015	08	Medium-term	55324
13	2015	12	Short-term	54725
14	2015	03	Short-term	48710
15	2015	07	Short-term	46044
16	2015	10	Short-term	41306
17	2015	04	Short-term	39099
18	2015	09	Short-term	35583

Figure 2.2.1 Screenshot of the results.

2.3 Number of listings in each price range for each year

This section shows the distribution of listings across different price ranges for each month and year. The **highest number of listings** fell within the **medium price range** in **September 2014**. Out of the top 18 results, most listings belong to the medium price range, with only two listings in the low price range and none in the high price range.

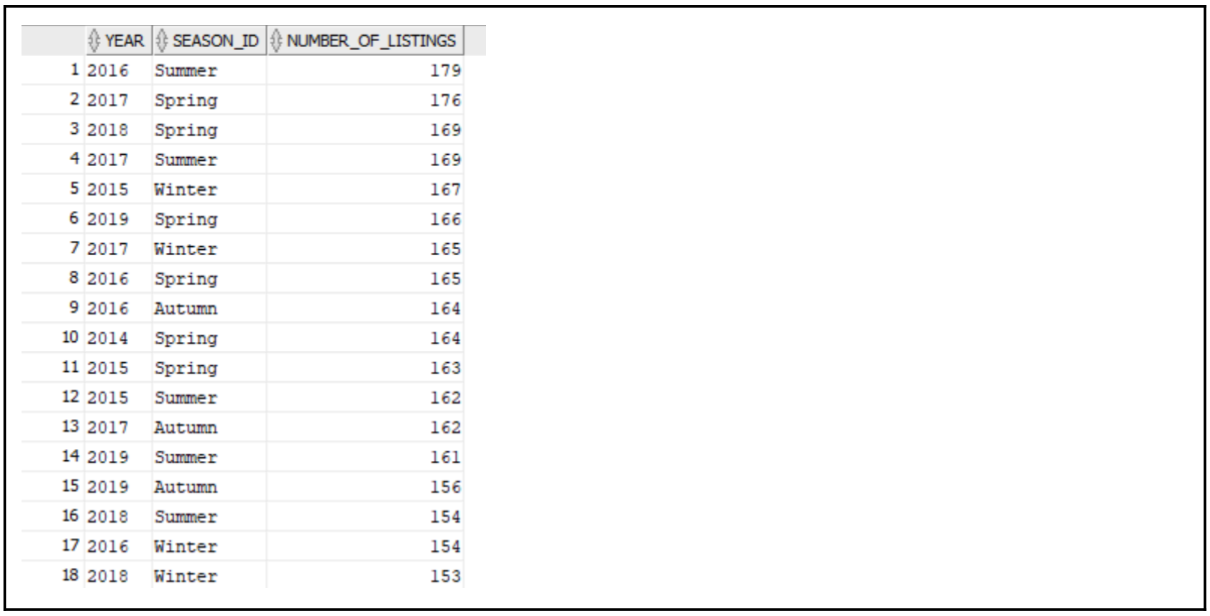


	YEAR	MONTH	PRICERANGE_ID	NUMBER_OF_LISTINGS
1	2014	09	Medium	40
2	2016	09	Medium	38
3	2016	06	Medium	38
4	2017	03	Medium	36
5	2016	12	Medium	36
6	2018	01	Medium	35
7	2015	01	Medium	35
8	2017	12	Low	35
9	2016	10	Medium	35
10	2016	01	Medium	34
11	2013	12	Medium	34
12	2016	11	Medium	34
13	2019	08	Medium	33
14	2017	11	Low	33
15	2019	10	Medium	32
16	2017	01	Medium	32
17	2015	06	Medium	32
18	2015	11	Medium	32

Figure 2.3.1 Screenshot of the results.

2.4 Total number of listings for each season

The finding below shows the number of listings by season. The **highest number of listings** is observed in the **summer of 2016,** with 179 listings, followed closely by spring 2017, with 176 listings. The top 18 results are mostly for listings from 2015 onward.

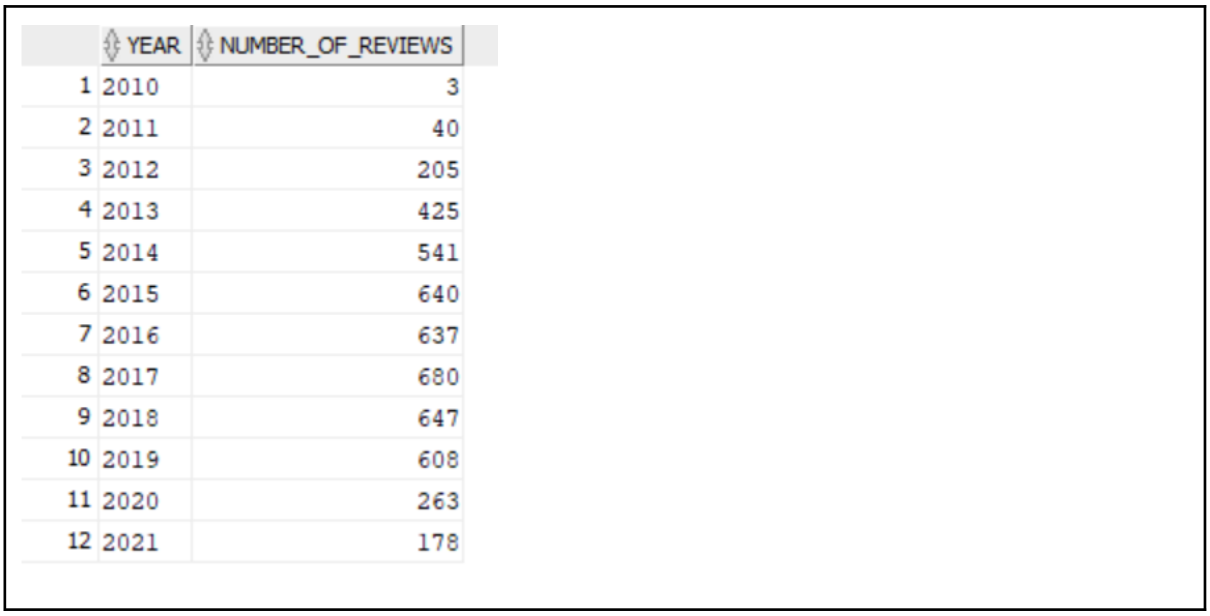


	YEAR	SEASON_ID	NUMBER_OF_LISTINGS
1	2016	Summer	179
2	2017	Spring	176
3	2018	Spring	169
4	2017	Summer	169
5	2015	Winter	167
6	2019	Spring	166
7	2017	Winter	165
8	2016	Spring	165
9	2016	Autumn	164
10	2014	Spring	164
11	2015	Spring	163
12	2015	Summer	162
13	2017	Autumn	162
14	2019	Summer	161
15	2019	Autumn	156
16	2018	Summer	154
17	2016	Winter	154
18	2018	Winter	153

Figure 2.4.1 Screenshot of the results.

2.5 Calculate the total number of reviews for each year

The screenshot below displays the trends in the total number of reviews per year. The **lowest number of reviews** is observed in **2010**, with only **3 reviews**, while 2017 has the highest number of reviews. In general, the number of reviews remained high between 2015 and 2019, with over 600 reviews each year. After a consistent increase from 2010 to 2019, there is a significant drop in reviews in 2020.



	YEAR	NUMBER_OF_REVIEWS
1	2010	3
2	2011	40
3	2012	205
4	2013	425
5	2014	541
6	2015	640
7	2016	637
8	2017	680
9	2018	647
10	2019	608
11	2020	263
12	2021	178

Figure 2.5.1 Screenshot of the results.

3. Appendix (SQL Code)

```
--Explore 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_amenity;
SELECT * FROM MStay.Amenity;

-- Drop tables
drop table MS_review;
drop table MS_booking;
drop table MS_guest;
drop table MS_listing;
drop table MS_host;
drop table MS_host_verification;
drop table MS_channel;
drop table MS_listing_type;
drop table MS_property;
drop table MS_property_amenity;
drop table MS_amenity;

-- Data Cleaning
-- Error 1: Duplicate booking id in booking table
SELECT booking_id, COUNT(*)
FROM MStay.booking
GROUP BY booking_id
HAVING COUNT(*) > 1; -- 2 duplicates

SELECT * FROM MStay.booking
WHERE booking_id = 537;

CREATE TABLE MS_booking as
SELECT DISTINCT *
```

```
FROM MStay.booking;

SELECT COUNT(*) FROM MStay.booking; --5002 rows
SELECT COUNT(*) FROM MS_booking; --5001 rows

--Proof no more duplicates
SELECT booking_id, COUNT(*)
FROM MS_booking
GROUP BY booking_id
HAVING COUNT(*) > 1;

SELECT * FROM MS_booking
WHERE booking_id = 537;

--Error 2: Duplicate host id in host table
SELECT host_id, COUNT(*)
FROM MStay.host
GROUP BY host_id
HAVING COUNT(*) > 1; -- 4 duplicates

SELECT * FROM MStay.host
WHERE host_id = 7046664;

CREATE TABLE MS_host AS
SELECT DISTINCT *
FROM MStay.host;

SELECT COUNT(*) FROM MStay.host; --3883 rows
SELECT COUNT(*) FROM MS_host; --3880 rows

--Proof no more duplicates
SELECT host_id, COUNT(*)
FROM MS_host
GROUP BY host_id
HAVING COUNT(*) > 1;

SELECT * FROM MS_host
WHERE host_id = 7046664;
```

```
        -- Create tables
        -- MS_booking, MS_host are created during data cleaning
        create table MS_review as select * from MStay.Review;
        create table MS_guest as select * from MStay.guest;
        create table MS_listing as select * from MStay.listing;
        create table MS_host_verification as select * from MStay.host_verification;
        create table MS_channel as select * from MStay.channel;
        create table MS_listing_type as select * from MStay.listing_type;
        create table MS_property as select * from MStay.property;
        create table MS_property_amenity as select * from
MStay.property_amenity;
        create table MS_amenity as select * from MStay.amenity;

        --Error 3: Invalid date in host_since in host table
        SELECT TO_CHAR(host_since, 'DD/MM/YYYY')
        FROM MS_host
        ORDER BY host_since DESC; --host_since : 16/05/9999

        SELECT *
        FROM MS_host
        WHERE TO_CHAR(host_since, 'DD/MM/YYYY') = '16/05/9999';

        UPDATE MS_host
        SET host_since = NULL
        WHERE TO_CHAR(host_since, 'DD/MM/YYYY') = '16/05/9999';

        DELETE FROM MS_host
        WHERE host_since IS NULL;

        SELECT TO_CHAR(host_since, 'DD/MM/YYYY')
        FROM MS_host
        ORDER BY host_since DESC;
        --Error 4: Negative value in listing_price in listnig table
        SELECT *
        FROM MS_listing
        WHERE listing_price < 0 ; -- 1 row(listing_id = 99999; listing_price =
-150)

        UPDATE MS_listing
```

```
SET listing_price = NULL
WHERE listing_price < 0;

DELETE FROM MS_listing
WHERE listing_price IS NULL;

SELECT *
FROM MS_listing
WHERE listing_price < 0 ;
--Error 5: Invalid relationship in review table
SELECT *
FROM MS_review
WHERE booking_id NOT IN
(SELECT booking_id
FROM MS_booking); --1

UPDATE MS_review
SET booking_id = NULL
WHERE booking_id NOT IN
(SELECT booking_id
FROM MS_booking);

DELETE FROM MS_review
WHERE booking_id IS NULL;

SELECT *
FROM MS_review
WHERE booking_id NOT IN
(SELECT booking_id
FROM MS_booking);
--Error 6 : Invalid relationship in host verification table
SELECT *
FROM MS_host_verification
WHERE host_id NOT IN
(SELECT host_id
FROM MS_host);--host_id =123

UPDATE MS_host_verification
SET host_id = NULL
```

```
WHERE host_id NOT IN
(SELECT host_id
FROM MS_host);

DELETE FROM MS_host_verification
WHERE host_id IS NULL;

SELECT *
FROM MS_host_verification
WHERE host_id NOT IN
(SELECT host_id
FROM MS_host);
--Error 7: 2 null values in review table
SELECT *
FROM MS_review
WHERE review_id IS NULL
  OR review_date IS NULL
  OR review_comment IS NULL; --2 rows

DELETE FROM  MS_review
WHERE review_comment IS NULL;

SELECT *
FROM MS_review
WHERE review_id IS NULL
  OR review_date IS NULL
  OR review_comment IS NULL;

--Error 8: 2 null values in amenity table
SELECT *
FROM MS_amenity
WHERE amm_id IS NULL
  OR amm_description IS NULL; --2 rows

DELETE FROM  MS_amenity
WHERE amm_id IS NULL
  OR amm_description IS NULL;

SELECT *
```

```
FROM MS_amenity
WHERE amm_id IS NULL
  OR amm_description IS NULL;
--Error 9: 3 Invalid relationship in property amenity table
SELECT *
FROM MS_property_amenity
WHERE amm_id NOT IN
(SELECT amm_id
FROM MS_amenity);

UPDATE MS_property_amenity
SET amm_id = NULL
WHERE amm_id NOT IN
(SELECT amm_id
FROM MS_amenity);

DELETE FROM MS_property_amenity
WHERE amm_id IS NULL;

--Data Cleaning Summary
--No of duplicate values: 2
--No of relationship problems: 3
--No of error date: 1
--No of incorrect value (negative value): 1
--No of null values: 2

--Count no of rows before data cleaning
SELECT
   (SELECT COUNT(*) FROM MStay.Review) AS review_count,
   (SELECT COUNT(*) FROM MStay.booking) AS booking_count,
   (SELECT COUNT(*) FROM MStay.guest) AS guest_count,
   (SELECT COUNT(*) FROM MStay.listing) AS listing_count,
   (SELECT COUNT(*) FROM MStay.host) AS host_count,
   (SELECT COUNT(*) FROM MStay.host_verification) AS
host_verification_count,
   (SELECT COUNT(*) FROM MStay.channel) AS channel_count,
   (SELECT COUNT(*) FROM MStay.listing_type) AS listing_type_count,
   (SELECT COUNT(*) FROM MStay.property) AS property_count,
   (SELECT COUNT(*) FROM MStay.property_amenity) AS
property_amenity_count,
   (SELECT COUNT(*) FROM MStay.Amenity) AS amenity_count
FROM dual;

--Count no of rows after data cleaning
SELECT
   (SELECT COUNT(*) FROM MS_Review) AS review_count,
   (SELECT COUNT(*) FROM MS_booking) AS booking_count,
   (SELECT COUNT(*) FROM MS_guest) AS guest_count,
   (SELECT COUNT(*) FROM MS_listing) AS listing_count,
   (SELECT COUNT(*) FROM MS_host) AS host_count,
   (SELECT COUNT(*) FROM MS_host_verification) AS
host_verification_count,
   (SELECT COUNT(*) FROM MS_channel) AS channel_count,
   (SELECT COUNT(*) FROM MS_listing_type) AS listing_type_count,
   (SELECT COUNT(*) FROM MS_property) AS property_count,
   (SELECT COUNT(*) FROM MS_property_amenity) AS
property_amenity_count,
   (SELECT COUNT(*) FROM MS_Amenity) AS amenity_count
FROM dual;

--IMPLEMENT STAR SCHEMA

--CREATE Dimensions
--1. Create timeDIM
DROP TABLE timeDIM CASCADE CONSTRAINTS PURGE;
CREATE TABLE timeDIM AS
SELECT DISTINCT
   TO_CHAR(TO_DATE(booking_date, 'DD/MM/YYYY'), 'YYYYMM') AS
time_ID,  -- Format as YYYYMM
   TO_CHAR(TO_DATE(booking_date, 'DD/MM/YYYY'), 'MM') AS month,
-- Extract month
   TO_CHAR(TO_DATE(booking_date, 'DD/MM/YYYY'), 'YYYY') AS year
-- Extract year
FROM MS_booking

UNION
```

```
        SELECT DISTINCT
          TO_CHAR(TO_DATE(review_date, 'DD/MM/YYYY'), 'YYYYMM') AS
time_ID,   -- Format as YYYYMM
          TO_CHAR(TO_DATE(review_date, 'DD/MM/YYYY'), 'MM') AS month,
-- Extract month
          TO_CHAR(TO_DATE(review_date, 'DD/MM/YYYY'), 'YYYY') AS year
-- Extract year
        FROM MS_review

        UNION

        SELECT DISTINCT
          TO_CHAR(TO_DATE(listing_date, 'DD/MM/YYYY'), 'YYYYMM') AS
time_ID,   -- Format as YYYYMM
          TO_CHAR(TO_DATE(listing_date, 'DD/MM/YYYY'), 'MM') AS month,
-- Extract month
          TO_CHAR(TO_DATE(listing_date, 'DD/MM/YYYY'), 'YYYY') AS year        --
Extract year
        FROM MS_listing;

        SELECT * FROM timeDIM;
        SELECT COUNT(*) FROM timeDIM;--136 rows

        --2. CREATE listingSeasonDIM
        DROP TABLE listingSeasonDIM CASCADE CONSTRAINTS PURGE;
        CREATE TABLE listingSeasonDIM(
        season_ID VARCHAR(10) NOT NULL,
        season_description VARCHAR2(20) NOT NULL);

        --INSERT values into listingSeasonDIM
        --Spring : 9-11
        --Summer : 12 - 2
        --Autumn : 3-5
        --Winter : 6-8
        INSERT INTO listingSeasonDIM VALUES('Spring','9 to 11');
        INSERT INTO listingSeasonDIM VALUES('Summer','12 to 2');
        INSERT INTO listingSeasonDIM VALUES('Autumn','3 to 5');
        INSERT INTO listingSeasonDIM VALUES('Winter','6 to 8');
        SELECT * FROM listingSeasonDIM;

        --3. CREATE BookingDurationDIM
        DROP TABLE BookingDurationDIM CASCADE CONSTRAINTS PURGE;
        CREATE TABLE BookingDurationDIM(
        Booking_DurationID VARCHAR2(20) NOT NULL,
        BookingDuration_description VARCHAR2(50) NOT NULL);

        --INSERT values in Booking DurationDIM
        --Short term : less than 30 nights, Medium-term: 30 to 90 nights, long term:
more than 90 nights
        INSERT INTO BookingDurationDIM VALUES('Short-term', 'less than 30
nights');
        INSERT INTO BookingDurationDIM VALUES('Medium-term', '30 to 90
nights');
        INSERT INTO BookingDurationDIM VALUES('Long-term', 'more than 90
nights');

        SELECT * FROM BookingDurationDIM;

        --4. CREATE listing DurationDIM
        DROP TABLE listingDurationDIM CASCADE CONSTRAINTS PURGE;
        CREATE TABLE listingDurationDIM(
        listing_DurationID VARCHAR2(20) NOT NULL,
        listingDuration_description VARCHAR2(50) NOT NULL);

        --INSERT values in listingDurationDIM
        --'Short-term':'less than 14 nights',Medium-term: 14 to 30 nights,
Long-term: more than 30 nights
        INSERT INTO listingDurationDIM VALUES('Short-term', 'less than 14
nights');
        INSERT INTO listingDurationDIM VALUES('Medium-term', '14 to 30
nights');
        INSERT INTO listingDurationDIM VALUES('Long-term', 'more than 30
nights');

        SELECT * FROM listingDurationDIM;
        --5. CREATE bookingCostDIM
        DROP TABLE bookingCostDIM CASCADE CONSTRAINTS PURGE;
        CREATE TABLE bookingCostDIM(
        booking_CostID VARCHAR2(20) NOT NULL,
        bookingCost_description VARCHAR2(50) NOT NULL);

        --INSERT Value bookingCostDIM
        --low: less than $5000, medium : $5000 to $10000, high: more than
$10000
        INSERT INTO bookingCostDIM VALUES('Low', 'less than $5000');
        INSERT INTO bookingCostDIM VALUES('Medium', '$5000 to $10000');
        INSERT INTO bookingCostDIM VALUES('High', 'more than $10000');

        SELECT * FROM bookingCostDIM;

        --6. CREATE listingPriceDIM
        DROP TABLE listingPriceDIM CASCADE CONSTRAINTS PURGE;
        CREATE TABLE listingPriceDIM(
        priceRange_ID VARCHAR2(20) NOT NULL,
        ListingPrice_description VARCHAR2(50) NOT NULL);

        --INSERT Value
        --low: less than $100, medium: $100 to $200, high: more than $200
        INSERT INTO listingPriceDIM VALUES('Low', 'less than $100');
        INSERT INTO listingPriceDIM VALUES('Medium', '$100 and $200');
        INSERT INTO listingPriceDIM VALUES('High', 'more than $200');

        SELECT * FROM listingPriceDIM;

        --7. CREATE listingTypeDIM
        DROP TABLE listingTypeDIM CASCADE CONSTRAINTS PURGE;
        CREATE TABLE listingTypeDIM AS
        SELECT * FROM MS_listing_type;

        SELECT * FROM listingTypeDIM;

        --8. CREATE hostDIM
        DROP TABLE hostDIM CASCADE CONSTRAINTS PURGE;
        CREATE TABLE hostDIM AS
        SELECT host_ID,host_name FROM MS_HOST;
        SELECT * FROM hostDIM;
        SELECT COUNT(*) FROM hostDIM;--3879 rows

        --9. CREATE ChannelDIM
        DROP TABLE channelDIM CASCADE CONSTRAINTS PURGE;
        CREATE TABLE channelDIM AS
        SELECT channel_id, channel_name FROM MS_Channel;

        SELECT * FROM channelDIM;
        SELECT COUNT(*) FROM channelDIM;--20 rows

        --10. CREATE HostChannelBRIDGE
        DROP TABLE HostChannelBRIDGE CASCADE CONSTRAINTS PURGE;
        CREATE TABLE HostChannelBRIDGE AS
        SELECT * FROM MS_Host_Verification;

        SELECT * FROM HostChannelBRIDGE;
        SELECT COUNT(*) FROM HostChannelBRIDGE;

        --CREATE Facts
        --1. CREATE reviewFACT (time_id,no_of_reviews[sum of Review_id])
        DROP TABLE reviewFACT CASCADE CONSTRAINTS PURGE;
        CREATE TABLE reviewFACT AS
        SELECT TO_CHAR(TO_DATE(R.review_date, 'DD/MM/YYYY'), 'YYYYMM')
AS time_ID,
        COUNT(R.review_id) AS No_of_reviews
        FROM MS_review R
        GROUP BY TO_CHAR(TO_DATE(R.review_date, 'DD/MM/YYYY'),
'YYYYMM');

        SELECT * FROM reviewFACT;
        SELECT COUNT(*) FROM reviewFACT;--126 rows

        --2. CREATE bookingFACT (time_id, booking_duration,
booking_cost,type_id,
        --total_no_of_bookings, total_booking_cost,average booking cost)

        --Create tempBookingFACT
        DROP TABLE tempBookingFACT CASCADE CONSTRAINTS PURGE;
```

```
CREATE TABLE tempBookingFACT AS
SELECT B.Booking_date,
B.booking_duration,
B.booking_cost,
L.type_id,
B.booking_id
FROM MS_Booking B, MS_Listing L
WHERE B.listing_id = L.listing_id;

--add column call booking_durationID
ALTER TABLE tempBookingFACT
ADD (booking_durationID VARCHAR(20));

UPDATE tempBookingFACT
SET booking_durationID = 'Short-term'
WHERE booking_duration < 30;

UPDATE tempBookingFACT
SET booking_durationID = 'Medium-term'
WHERE booking_duration >= 30 AND booking_duration <=90;

UPDATE tempBookingFACT
SET booking_durationID = 'Long-term'
WHERE booking_duration > 90;

--add column call booking_costID
ALTER TABLE tempBookingFACT
ADD (booking_costID VARCHAR(20));

UPDATE tempBookingFACT
SET booking_costID = 'Low'
WHERE booking_cost < 5000;

UPDATE tempBookingFACT
SET booking_costID = 'Medium'
WHERE booking_cost >= 5000 AND booking_cost <=10000;

UPDATE tempBookingFACT
SET booking_costID = 'High'
```

```
        WHERE booking_cost > 10000;

        --CREATE BookingFACT
        DROP TABLE BookingFACT CASCADE CONSTRAINTS PURGE;
        CREATE TABLE BookingFACT AS
        SELECT TO_CHAR(TO_DATE(T.Booking_Date, 'DD/MM/YYYY'), 'YYYYMM')
AS time_ID,
        T.Booking_durationID,T.Booking_CostID,T.Type_ID,
        SUM(T.Booking_ID) AS Total_No_of_Bookings,SUM(T.Booking_Cost) AS
Total_Booking_Cost
        FROM
           tempBookingFACT T
        GROUP BY
          TO_CHAR(TO_DATE(T.Booking_Date, 'DD/MM/YYYY'), 'YYYYMM'),
          T.Booking_durationID,T.Booking_CostID,T.Type_ID;

        SELECT * FROM BookingFACT ORDER BY TIME_ID ASC;
        SELECT COUNT(*) FROM BookingFACT;--766 rows

        --3. CREATE listingFACT (time_id, type_id, season_id, priceRange_id,
listing_duration,
        --host_id,no_of_listings[listing_id])

        --CREATE tempListingFACT
        DROP TABLE tempListingFACT CASCADE CONSTRAINTS PURGE;
        CREATE TABLE tempListingFACT AS
        SELECT L.Listing_date,L.type_id,
        L.listing_price, L.listing_max_nights, H.Host_id, L.Listing_id
        FROM MS_Listing L, MS_Host H
        WHERE L.Host_id = H.Host_id;

        --add column call season_ID
        ALTER TABLE tempListingFACT
        ADD (season_ID VARCHAR(20));

        UPDATE tempListingFACT
        SET season_ID = 'Spring'
        WHERE TO_CHAR(Listing_Date,'MM')>= 9 AND
        TO_CHAR(Listing_Date,'MM')<=11; --mm 9 to 11
```

```
UPDATE tempListingFACT
SET season_ID = 'Summer'
WHERE TO_CHAR(Listing_Date,'MM')>= 12 OR
TO_CHAR(Listing_Date,'MM')<=2; --mm 12 to 2

UPDATE tempListingFACT
SET season_ID = 'Autumn'
WHERE TO_CHAR(Listing_Date,'MM')>= 3 AND
TO_CHAR(Listing_Date,'MM')<=5; --3 to 5

UPDATE tempListingFACT
SET season_ID = 'Winter'
WHERE TO_CHAR(Listing_Date,'MM')>= 6 AND
TO_CHAR(Listing_Date,'MM')<=8; --6 to 8

--add column call priceRange_ID
ALTER TABLE tempListingFACT
ADD (PriceRange_ID VARCHAR(20));

UPDATE tempListingFACT
SET PriceRange_ID = 'Low'
WHERE Listing_Price < 100;

UPDATE tempListingFACT
SET PriceRange_ID = 'Medium'
WHERE Listing_Price >= 100 AND Listing_Price <=200;

UPDATE tempListingFACT
SET PriceRange_ID = 'High'
WHERE Listing_Price > 200;

--add column call Listing_DurationID
ALTER TABLE tempListingFACT
ADD (Listing_DurationID VARCHAR(20));

UPDATE tempListingFACT
SET Listing_DurationID = 'Short-term'
WHERE Listing_Max_Nights < 14;
```

```
        UPDATE tempListingFACT
        SET Listing_DurationID = 'Medium-term'
        WHERE Listing_Max_Nights >= 14 AND Listing_Max_Nights <=30;

        UPDATE tempListingFACT
        SET Listing_DurationID = 'Long-term'
        WHERE Listing_Max_Nights > 30;

        --CREATE ListingFACT
        DROP TABLE ListingFACT CASCADE CONSTRAINTS PURGE;
        CREATE TABLE ListingFACT AS
        SELECT TO_CHAR(TO_DATE(T.Listing_Date, 'DD/MM/YYYY'), 'YYYYMM')
AS time_ID,
        T.Type_ID,T.Season_ID,T.PriceRange_ID,T.Listing_DurationID,T.Host_ID,
        COUNT(T.Listing_ID) AS No_of_Listings
        FROM
          tempListingFACT T
        GROUP BY
          TO_CHAR(TO_DATE(T.Listing_Date, 'DD/MM/YYYY'), 'YYYYMM'),
          T.Type_ID,T.Season_ID,T.PriceRange_ID,T.Listing_DurationID,T.Host_ID;

        SELECT * FROM ListingFACT;
        SELECT COUNT(*) FROM ListingFACT;--2031rows

        -- Data Analytics Stage
        -- 1. total number of bookings and the average booking cost in year 2021
        SELECT t.Year, t.Month,
             SUM(bf.Total_no_of_Bookings) AS TotalBookings,
             ROUND((SUM(bf.Total_booking_cost) /
COUNT(bf.Total_no_of_Bookings)), 2) AS AVG_BookingCost
        FROM BookingFACT bf, TimeDIM t
        WHERE bf.Time_ID = t.Time_ID
        AND t.Year = 2021
        GROUP BY t.Year, t.Month
        ORDER BY AVG_BookingCost DESC;

        -- 2. total number of bookings for each booking duration type in 2015
        SELECT t.Year, t.Month, bf.Booking_DurationID,
    SUM(bf.Total_no_of_Bookings) AS TotalBookings
FROM BookingFACT bf, TimeDIM t
WHERE bf.Time_ID = t.Time_ID
AND t.Year = 2015
GROUP BY t.Year, t.Month, bf.Booking_DurationID
ORDER BY TotalBookings DESC;

-- 3. Count the number of listings in each price range for the year 2015
SELECT t.Year, t.Month, lf.PriceRange_ID,
    SUM(no_of_listings) AS Number_of_Listings
FROM ListingFACT lf, TimeDIM t
WHERE lf.Time_ID = t.Time_ID
GROUP BY t.Year, t.Month, lf.PriceRange_ID
ORDER BY Number_of_Listings DESC;

-- 4. Display the total number of listings for each season
SELECT t.Year, S.Season_ID,
    SUM(no_of_listings) AS Number_of_Listings
FROM ListingFACT lf, TimeDIM t, ListingSeasonDIM S
WHERE lf.Time_ID = t.Time_ID
AND lf.Season_ID = S.Season_ID
GROUP BY t.Year, S.Season_ID
ORDER BY Number_of_Listings DESC;

-- 5. Calculate the total number of reviews for each year
SELECT t.Year,
    SUM(rf.No_of_reviews) AS Number_of_Reviews
FROM ReviewFACT rf, TimeDIM t
WHERE rf.Time_ID = t.Time_ID
GROUP BY t.Year
ORDER BY t.Year;

--Overview of FACT Tables
SELECT * FROM ReviewFACT;
SELECT * FROM BookingFACT;
SELECT * FROM ListingFACT;
--Overview of Dimension Tables
SELECT * FROM TimeDIM;
SELECT * FROM BookingDurationDIM;
SELECT * FROM BookingCostDIM;

SELECT * FROM ListingTypeDIM;
SELECT * FROM ListingSeasonDIM;
SELECT * FROM ListingDurationDIM;
SELECT * FROM ListingPriceDIM;

SELECT * FROM HostDIM;
SELECT * FROM HostChannelBRIDGE;
SELECT * FROM ChannelDIM;
```