MONASH University

Major Assignment : M-Stay Transformation and & Data Analysis Report

FIT3003 : Business Intelligence and Data Warehousing

By Ng Chen Ting

Table of Contents

Section A: Data Cleaning Process

Data cleaning is a critical step before transferring data to a data warehouse to ensure quality and reliability. The strategy used to identify the problems are based on a table-by-table basis for each problem type systematically. There are 4 types of problems identified in this operational database, starting off with duplication problems, followed by relationship issues, null value problems, and data inconsistencies. This covers all aspects of problems and ensures data is cleaned and adheres to correct relationships before constructing the data warehouse.

Duplication Problem

Strategy:

- To identify duplicate records in a table, the strategy involves grouping the data by their Primary Key and counting how many times each PK value appears. By grouping and applying a condition to filter out groups where the count is greater than 1, we can identify records that are duplicated and need further correction.
- To rectify the issue of duplicate records, a new table is created where we select distinct rows from the original table and insert them into a newly created table to prevent data redundancy.

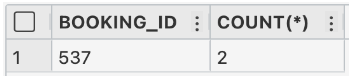
Exploration Code:

```
-- Review Table
SELECT Review_ID, COUNT(*)
FROM MStay.REVIEW
GROUP BY Review_ID
HAVING COUNT(*) > 1;

-- Booking Table --> Problem 1
SELECT Booking_ID, COUNT(*)
FROM MStay.BOOKING
GROUP BY Booking_ID
HAVING COUNT(*) > 1;

-- Guest Table
SELECT Guest_ID, COUNT(*)
FROM MStay.GUEST
GROUP BY Guest_ID
HAVING COUNT(*) > 1;

-- Listing Table
SELECT Listing_ID, COUNT(*)
FROM MStay.LISTING
GROUP BY Listing_ID
HAVING COUNT(*) > 1;
```

```
-- Host Table --> **Problem 2**
SELECT Host_ID, COUNT(*)
FROM MStay.HOST
GROUP BY Host_ID
HAVING COUNT(*) > 1;

-- Host Verification Table
SELECT Host_ID, Channel_ID, COUNT(*)
FROM MStay.HOST_VERIFICATION
GROUP BY Host_ID, Channel_ID
HAVING COUNT(*) > 1;

-- Channel Table
SELECT Channel_ID, COUNT(*)
FROM MStay.CHANNEL
GROUP BY Channel_ID
HAVING COUNT(*) > 1;

-- Listing Type Table
SELECT Type_ID, COUNT(*)
FROM MStay.LISTING_TYPE
GROUP BY Type_ID
HAVING COUNT(*) > 1;

-- Property Table
SELECT Prop_ID, COUNT(*)
FROM MStay.PROPERTY
GROUP BY Prop_ID
HAVING COUNT(*) > 1;

-- Property Amenity Table
SELECT Prop_ID, Amm_ID, COUNT(*)
FROM MStay.PROPERTY_AMENITY
GROUP BY Prop_ID, Amm_ID
HAVING COUNT(*) > 1;

-- Amenity Table
SELECT Amm_ID, COUNT(*)
FROM MStay.AMENITY
GROUP BY Amm_ID
HAVING COUNT(*) > 1;
```

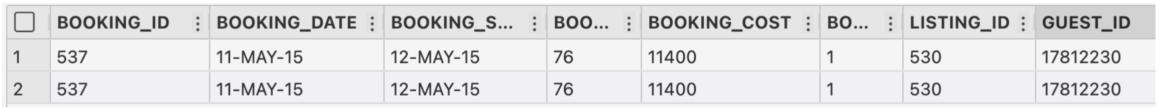
Problem 1: Duplicate records in BOOKING Table

```
-- Before Data Cleaning
SELECT Booking_ID, COUNT(*)
FROM MStay.BOOKING
GROUP BY Booking_ID
HAVING COUNT(*) > 1;
```



	BOOKING_ID	COUNT(*)
1	537	2

```
SELECT * FROM MStay.BOOKING WHERE Booking_ID=537;
```

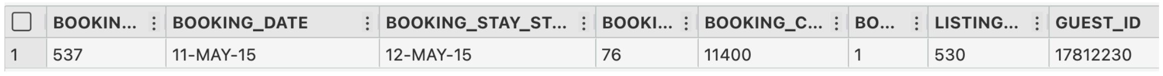


	BOOKING_ID	BOOKING_DATE	BOOKING_S...	BOO...	BOOKING_COST	BO...	LISTING_ID	GUEST_ID
1	537	11-MAY-15	12-MAY-15	76	11400	1	530	17812230
2	537	11-MAY-15	12-MAY-15	76	11400	1	530	17812230

```
-- Rectification
CREATE TABLE BOOKING AS
SELECT DISTINCT *
FROM MStay.BOOKING;

-- After Data Cleaning
SELECT Booking_ID, COUNT(*)
FROM BOOKING
GROUP BY Booking_ID
HAVING COUNT(*) > 1;
```

no rows selected

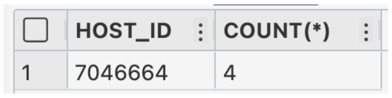
```
SELECT * FROM BOOKING WHERE Booking_ID=537;
```



	BOOKIN...	BOOKING_DATE	BOOKING_STAY_ST...	BOOKI...	BOOKING_C...	BO...	LISTING...	GUEST_ID
1	537	11-MAY-15	12-MAY-15	76	11400	1	530	17812230

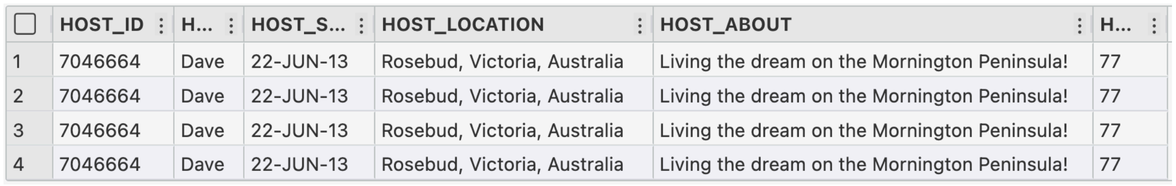
Problem 2: Duplicate records in HOST Table

```
-- Before Data Cleaning
SELECT HOST_ID, COUNT(*)
FROM MStay.HOST
GROUP BY HOST_ID
HAVING COUNT(*) > 1;
```



☐	HOST_ID	COUNT(*)
1	7046664	4

```
SELECT * FROM MStay.HOST WHERE Host_ID=7046664;
```

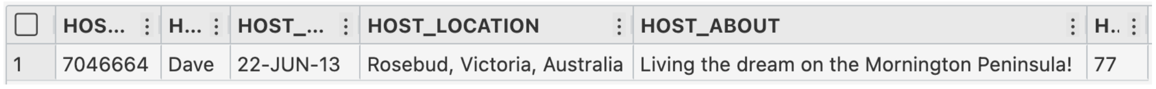


☐	HOST_ID	H...	HOST_S...	HOST_LOCATION	HOST_ABOUT	H...
1	7046664	Dave	22-JUN-13	Rosebud, Victoria, Australia	Living the dream on the Mornington Peninsula!	77
2	7046664	Dave	22-JUN-13	Rosebud, Victoria, Australia	Living the dream on the Mornington Peninsula!	77
3	7046664	Dave	22-JUN-13	Rosebud, Victoria, Australia	Living the dream on the Mornington Peninsula!	77
4	7046664	Dave	22-JUN-13	Rosebud, Victoria, Australia	Living the dream on the Mornington Peninsula!	77

```
-- Rectification
CREATE TABLE HOST AS
SELECT DISTINCT *
FROM MStay.HOST;

-- After Data Cleaning
SELECT HOST_ID, COUNT(*)
FROM HOST
GROUP BY HOST_ID
HAVING COUNT(*) > 1;
```

```
no rows selected
```

```
SELECT * FROM HOST WHERE Host_ID=7046664;
```



☐	HOS...	H...	HOST_...	HOST_LOCATION	HOST_ABOUT	H..
1	7046664	Dave	22-JUN-13	Rosebud, Victoria, Australia	Living the dream on the Mornington Peninsula!	77

Relationship Problem

Strategy:

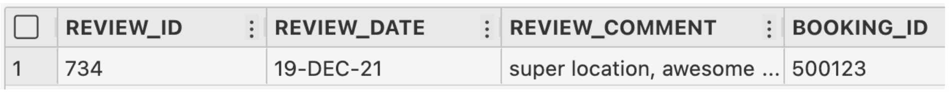
- To identify invalid foreign key values, check whether the foreign key values in a child table exist as primary key values in the corresponding parent table.
- To rectify, invalid foreign key values are set to NULL to prevent relationship errors and maintain proper entity linkage.
- After checking all tables for this problem, the null values in the foreign keys of the previously amended tables will be removed as they are invalid records.

Exploration Code:

```
-- Review Table --> Problem 3
SELECT * FROM MStay.REVIEW WHERE Booking_ID NOT IN (
    SELECT Booking_ID FROM BOOKING);

-- Booking Table
SELECT * FROM BOOKING WHERE Listing_ID NOT IN (
    SELECT Listing_ID FROM LISTING);
SELECT * FROM BOOKING WHERE Guest_ID NOT IN (
    SELECT Guest_ID FROM GUEST);

-- Listing Table
SELECT * FROM MSTAY.LISTING WHERE Prop_ID NOT IN ( --> Problem 4a
    SELECT Prop_ID FROM PROPERTY);
SELECT * FROM MSTAY.LISTING WHERE Type_ID NOT IN (
    SELECT Type_ID FROM TYPE);
SELECT * FROM MSTAY.LISTING WHERE Host_ID NOT IN ( --> Problem 4b
    SELECT Host_ID FROM HOST);
SELECT * FROM MSTAY.LISTING WHERE Host_ID NOT IN (
    SELECT Host_ID FROM HOST);

--  Host Verification Table
SELECT * FROM MSTAY.HOST_VERIFICATION WHERE Host_ID NOT IN ( --> Problem 5a
    SELECT Host_ID FROM HOST);
SELECT * FROM MSTAY.HOST_VERIFICATION WHERE Channel_ID NOT IN ( --> Problem 5b
    SELECT Channel_ID FROM CHANNEL);

-- Property Amenity Table
SELECT * FROM MSTAY.PROPERTY_AMENITY WHERE Prop_ID NOT IN (
    SELECT Prop_ID FROM PROPERTY);
SELECT * FROM MSTAY.PROPERTY_AMENITY WHERE Amm_ID NOT IN (
    SELECT Amm_ID FROM AMENITY);
```

Problem 3: A Booking ID (FK) in REVIEW table is not a PK in BOOKING table

```
-- Before Data Cleaning
SELECT *
FROM MStay.REVIEW WHERE Booking_ID NOT IN
(SELECT Booking_ID
FROM BOOKING);
```



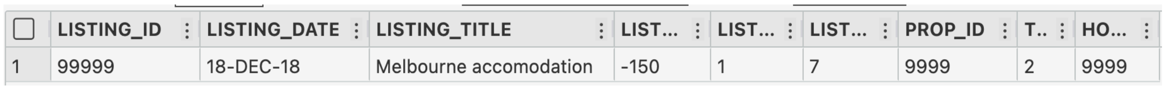
	REVIEW_ID	REVIEW_DATE	REVIEW_COMMENT	BOOKING_ID
1	734	19-DEC-21	super location, awesome ...	500123

```
-- Rectification
CREATE TABLE REVIEW AS SELECT *
FROM MStay.REVIEW;

UPDATE REVIEW
SET Booking_ID = NULL WHERE Booking_ID NOT IN
(SELECT Booking_ID
FROM BOOKING);

-- After Data Cleaning
SELECT *
FROM REVIEW WHERE Booking_ID NOT IN
(SELECT Booking_ID
FROM BOOKING);
```

no rows selected

Problem 4a: A Property ID (FK) in LISTING table is not a PK in PROPERTY table

```
-- Before Data Cleaning
SELECT *
FROM MStay.LISTING WHERE Prop_ID NOT IN
(SELECT Prop_ID
FROM MStay.PROPERTY);
```



	LISTING_ID	LISTING_DATE	LISTING_TITLE	LIST...	LIST...	LIST...	PROP_ID	T..	HO...
1	99999	18-DEC-18	Melbourne accomodation	-150	1	7	9999	2	9999

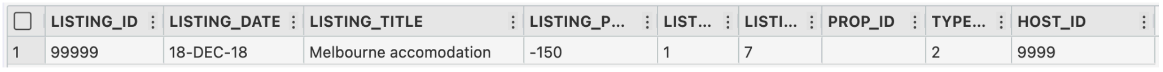
```
-- Rectification
CREATE TABLE LISTING AS SELECT *
FROM MStay.LISTING;

UPDATE LISTING
SET Prop_ID = NULL WHERE Prop_ID NOT IN
(SELECT Prop_ID
FROM MSTAY.PROPERTY);
```

```
-- After Data Cleaning
SELECT *
FROM LISTING WHERE Prop_ID NOT IN
(SELECT Prop_ID
FROM MStay.PROPERTY);
```

```
no rows selected
```

Problem 4b: A Host ID (FK) in LISTING table is not a PK in HOST table

```
-- Before Data Cleaning
SELECT *
FROM MStay.LISTING WHERE Host_ID NOT IN
(SELECT Host_ID
FROM MStay.HOST);
```



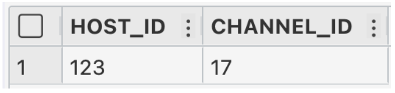
	LISTING_ID	LISTING_DATE	LISTING_TITLE	LISTING_P...	LIST...	LISTI...	PROP_ID	TYPE...	HOST_ID
1	99999	18-DEC-18	Melbourne accomodation	-150	1	7		2	9999

```
-- Rectification
UPDATE LISTING
SET Host_ID = NULL WHERE Host_ID NOT IN
(SELECT Host_ID
FROM MSTAY.HOST);

-- After Data Cleaning
SELECT *
FROM LISTING WHERE Host_ID NOT IN
(SELECT Host_ID
FROM MStay.HOST);
```

```
no rows selected
```

Problem 5a: A Host ID (FK) in HOST_VERIFICATION table is not a PK in HOST table

```
-- Before Data Cleaning
SELECT *
FROM MStay.HOST_VERIFICATION WHERE Host_ID NOT IN
(SELECT Host_ID
FROM MStay.HOST);
```



	HOST_ID	CHANNEL_ID
1	123	17

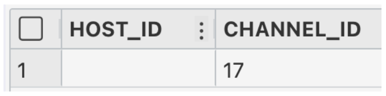
```
-- Rectification
CREATE TABLE HOST_VERIFICATION AS
SELECT *
FROM MStay.HOST_VERIFICATION;
```

```
UPDATE HOST_VERIFICATION
SET Host_ID = NULL WHERE Host_ID NOT IN
(SELECT Host_ID
FROM MSTAY.HOST);

-- After Data Cleaning
SELECT *
FROM HOST_VERIFICATION WHERE Host_ID NOT IN
(SELECT Host_ID
FROM MStay.HOST);
```

```
no rows selected
```

Problem 5b: A Channel ID (FK) in HOST_VERIFICATION table is not a PK in CHANNEL table

```
-- Before Data Cleaning
SELECT *
FROM HOST_VERIFICATION WHERE Channel_ID NOT IN
(SELECT Channel_ID
FROM MStay.CHANNEL);
```



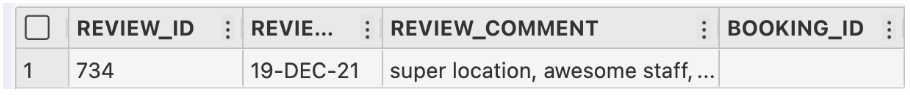
	HOST_ID	CHANNEL_ID
1		17

```
-- Rectification
UPDATE HOST_VERIFICATION
SET Channel_ID = NULL WHERE Channel_ID NOT IN
(SELECT Channel_ID
FROM MSTAY.CHANNEL);

-- After Data Cleaning
SELECT *
FROM HOST_VERIFICATION WHERE Channel_ID NOT IN
(SELECT Channel_ID
FROM MStay.CHANNEL);
```

```
no rows selected
```

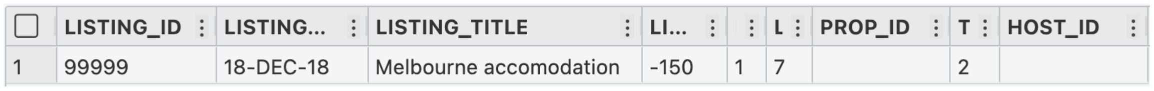
Problem 3, 4, 5: Deletion of Invalid Records

```
-- Before Data Cleaning
SELECT *
FROM REVIEW
WHERE Booking_ID IS NULL;
```



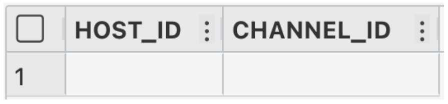
	REVIEW_ID	REVIE...	REVIEW_COMMENT	BOOKING_ID
1	734	19-DEC-21	super location, awesome staff, ...	

```
SELECT *
FROM LISTING
WHERE Prop_ID IS NULL OR Host_ID IS NULL;
```



	LISTING_ID	LISTING...	LISTING_TITLE	LI...		L	PROP_ID	T	HOST_ID
1	99999	18-DEC-18	Melbourne accomodation	-150	1	7		2	

```
SELECT *
FROM HOST_VERIFICATION
WHERE Host_ID IS NULL OR Channel_ID IS NULL;
```



	HOST_ID	CHANNEL_ID
1		

```
-- Rectification
DELETE FROM REVIEW
WHERE Booking_ID IS NULL;

DELETE FROM LISTING
WHERE Prop_ID IS NULL OR Host_ID IS NULL;

DELETE FROM HOST_VERIFICATION
WHERE Host_ID IS NULL OR Channel_ID IS NULL;

-- After Data Cleaning
SELECT *
FROM REVIEW
WHERE Booking_ID IS NULL;
```

```
no rows selected
```

```
SELECT *
FROM LISTING
WHERE Prop_ID IS NULL OR Host_ID IS NULL;
```

```
no rows selected
```

```
SELECT *
FROM HOST_VERIFICATION
WHERE Host_ID IS NULL OR Channel_ID IS NULL;
```

```
no rows selected
```

Null Values Problem

Strategy:

- To identify NULL values at attribute levels, each table's Primary Key value is checked for NULL values
- To rectify, delete all values where its Primary Key value is NULL to maintain data integrity and ensure accurate relationships between tables.

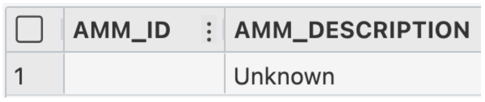
Exploration Code:

```
-- Review Table
SELECT * FROM REVIEW
WHERE Review_ID is NULL;

-- Booking Table
SELECT * FROM BOOKING
WHERE Amm_ID is NULL;

-- Guest Table
SELECT * FROM MSTAY.GUEST
WHERE Guest_ID is NULL;

-- Listing Table
SELECT * FROM LISTING
WHERE Listing_ID is NULL;

-- Host Table
SELECT * FROM HOST
WHERE Host_ID is NULL;

-- Host Verification Table
SELECT * FROM HOST_VERIFICATION
WHERE Host_ID is NULL;
SELECT * FROM HOST_VERIFICATION
WHERE Channel_ID is NULL;

-- Channel Table
SELECT * FROM MStay.CHANNEL
WHERE Channel_ID is NULL;

-- Listing Type Table
SELECT * FROM MStay.LISTING_TYPE
WHERE Type_ID is NULL;

-- Property Table
SELECT * FROM MStay.PROPERTY
WHERE Prop_ID is NULL;
```

```
-- Property Amenity Table
SELECT * FROM MStay.PROPERTY_AMENITY
WHERE Prop_ID is NULL;
SELECT * FROM MStay.PROPERTY_AMENITY
WHERE Amm_ID is NULL;

-- Amenity Table --> Problem 6
SELECT * FROM MStay.AMENITY
WHERE Amm_ID is NULL;
```

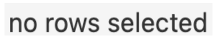
Problem 6: A NULL value in the Primary Key of AMENITY table

```
-- Before Data Cleaning
SELECT * FROM MStay.AMENITY
WHERE Amm_ID is NULL;
```



	AMM_ID	AMM_DESCRIPTION
1		Unknown

```
-- Rectification
CREATE TABLE AMENITY AS
SELECT * FROM MStay.AMENITY;

DELETE FROM AMENITY
WHERE Amm_ID is NULL;

-- After Data Cleaning
SELECT * FROM AMENITY
WHERE Amm_ID is NULL;
```



no rows selected

Inconsistent and Incorrect Values Problem

Strategy:

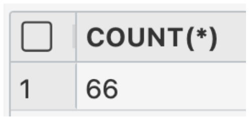
- To identify date related incorrect values, attributes of review dates are compared with the booking dates, the records where review dates are before the booking dates are incorrect.
- To rectify, delete all incorrect date records to ensure logical consistency in the dataset, as reviews should only occur after bookings.

Exploration Code:

```
-- Check if Any Review Date is Before Booking Date / Booking Stay Start Date
--> Problem 7
SELECT b.Booking_Date, b.Booking_Stay_Start_Date, r.Review_Date,
r.review_comment
FROM BOOKING b, REVIEW r
WHERE b.booking_ID = r.booking_ID AND r.Review_Date < b.Booking_Date;

SELECT b.Booking_Date, b.Booking_Stay_Start_Date, r.Review_Date,
r.review_comment
FROM BOOKING b, REVIEW r
WHERE b.booking_ID = r.booking_ID AND r.Review_Date <
b.Booking_Stay_Start_Date;

-- Check if any Booking Date is Before Listing Date
SELECT l.Listing_ID, b.Booking_Date, b.Booking_Stay_Start_Date, l.Listing_Date
FROM BOOKING b, LISTING l
WHERE b.listing_ID = l.Listing_ID AND l.Listing_Date > b.Booking_Date;

SELECT l.Listing_ID, b.Booking_Date, b.Booking_Stay_Start_Date, l.Listing_Date
FROM BOOKING b, LISTING l
WHERE b.listing_ID = l.Listing_ID AND l.Listing_Date >
b.Booking_Stay_Start_Date;
```

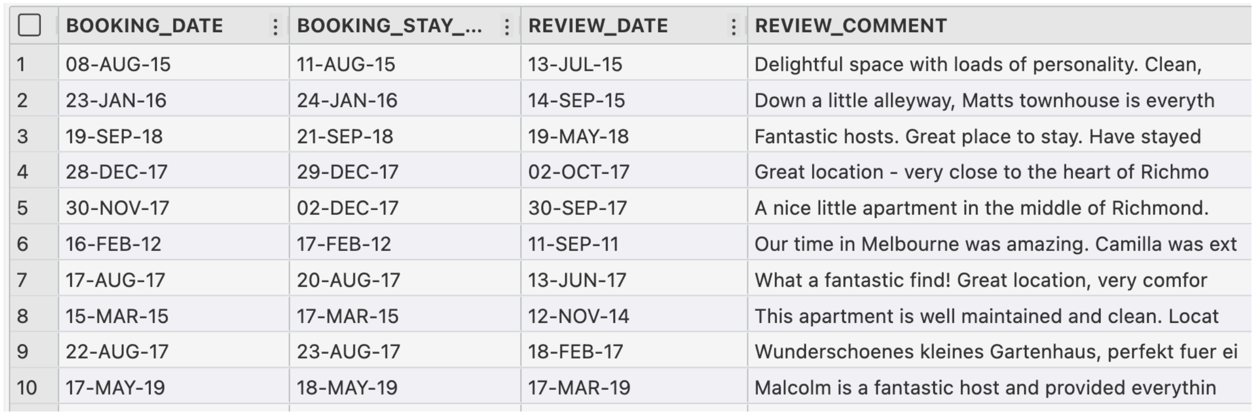
Problem 7: The review date is earlier than the booking start date

```
-- Before Data Cleaning
SELECT COUNT(*)
FROM BOOKING b, REVIEW r
WHERE b.booking_ID = r.booking_ID AND r.Review_Date <
b.Booking_Stay_Start_Date;
```



	COUNT(*)
1	66

```
SELECT b.Booking_Date, b.Booking_Stay_Start_Date, r.Review_Date,
r.review_comment
FROM BOOKING b, REVIEW r
WHERE b.booking_ID = r.booking_ID AND r.Review_Date <
b.Booking_Stay_Start_Date;
```



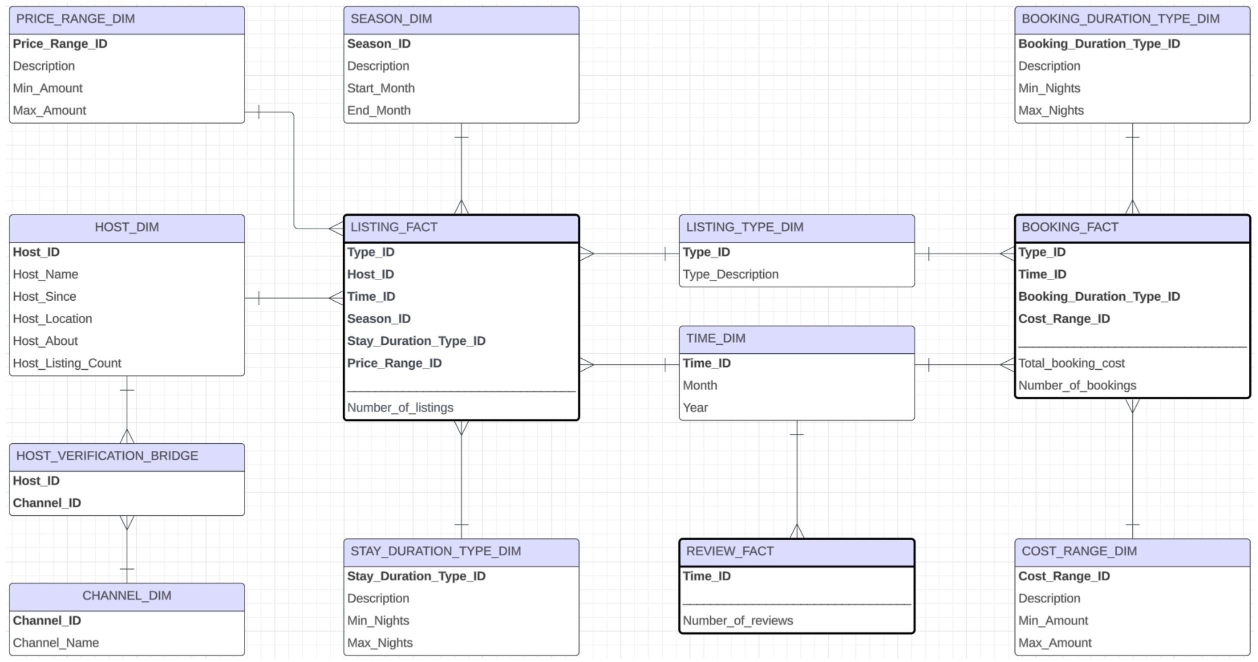
	BOOKING_DATE	BOOKING_STAY_...	REVIEW_DATE	REVIEW_COMMENT
1	08-AUG-15	11-AUG-15	13-JUL-15	Delightful space with loads of personality. Clean,
2	23-JAN-16	24-JAN-16	14-SEP-15	Down a little alleyway, Matts townhouse is everyth
3	19-SEP-18	21-SEP-18	19-MAY-18	Fantastic hosts. Great place to stay. Have stayed
4	28-DEC-17	29-DEC-17	02-OCT-17	Great location - very close to the heart of Richmo
5	30-NOV-17	02-DEC-17	30-SEP-17	A nice little apartment in the middle of Richmond.
6	16-FEB-12	17-FEB-12	11-SEP-11	Our time in Melbourne was amazing. Camilla was ext
7	17-AUG-17	20-AUG-17	13-JUN-17	What a fantastic find! Great location, very comfor
8	15-MAR-15	17-MAR-15	12-NOV-14	This apartment is well maintained and clean. Locat
9	22-AUG-17	23-AUG-17	18-FEB-17	Wunderschoenes kleines Gartenhaus, perfekt fuer ei
10	17-MAY-19	18-MAY-19	17-MAR-19	Malcolm is a fantastic host and provided everythin

```
-- Rectification
DELETE FROM REVIEW
WHERE review_ID IN (
    SELECT r.Review_ID
    FROM BOOKING b, REVIEW r
    WHERE b.booking_ID = r.booking_ID AND r.Review_Date <
b.Booking_Stay_Start_Date
);

-- After Data Cleaning
SELECT b.Booking_Date, b.Booking_Stay_Start_Date, r.Review_Date,
r.review_comment
FROM BOOKING b, REVIEW r
WHERE b.booking_ID = r.booking_ID AND r.Review_Date <
b.Booking_Stay_Start_Date;
```

```
no rows selected
```

Section B: Designing Data Warehouses

Task A : Star Schema Diagram



Task B : Suggestions to increase granularity

Granularity can be increased by reducing the level of aggregation. The Star Schema shown above has the highest level of aggregation.

Two methods can be used:

1. **Adding new dimensions**

    - **Property dimension** can be added to LISTING FACT
      This enhances granularity by providing detailed attributes of each property, such as property ratings, number of furnitures and amenities.

    - **Guest dimension** can be added to BOOKING FACT
      This enhances granularity by providing the name of guests. This level of detail allows for a more personalized analysis of booking patterns for specific guests.

2. **Replacing an existing dimension with higher granularity dimensions**

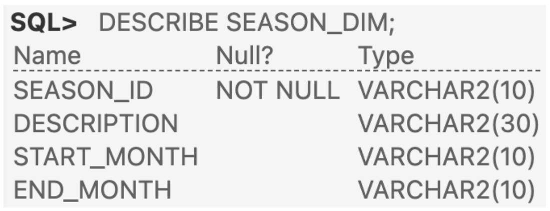
    - The **Time Dimension** which is currently in Month and Year can be changed to include Date and Time. This allows bookings to be tracked at a daily and hourly level, enabling detailed trend analysis over specific time periods, such as identifying peak booking hours or assessing day-of-week performance.

Section C: Star Schema Implementation

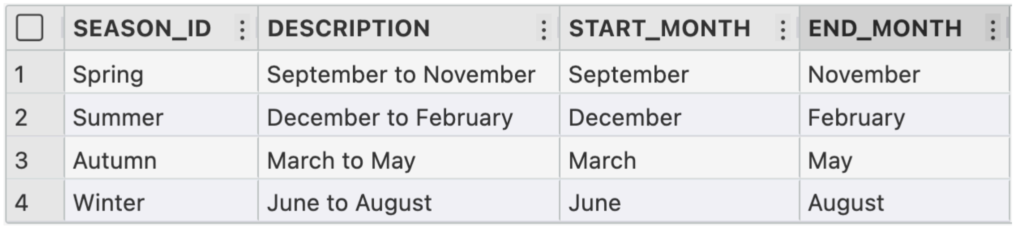
Dimension Implementation

Season Dimension:

```
-- Create Season Dimension
CREATE TABLE SEASON_DIM (
    Season_ID varchar2(10) PRIMARY KEY,
    Description varchar2(30),
    Start_Month varchar2(10),
    End_Month varchar2(10)
);

-- Populate Dimension
INSERT INTO SEASON_DIM
       values('Spring', 'September to November', 'September', 'November');
INSERT INTO SEASON_DIM
       values('Summer', 'December to February', 'December', 'February');
INSERT INTO SEASON_DIM
       values('Autumn', 'March to May', 'March', 'May');
INSERT INTO SEASON_DIM
       values('Winter', 'June to August', 'June', 'August');
```



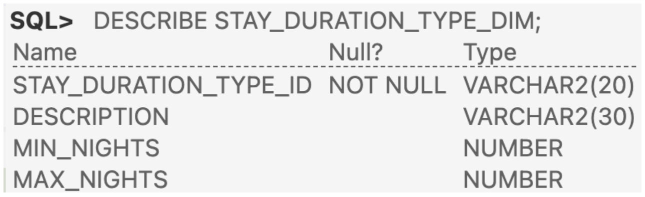
```
SQL> DESCRIBE SEASON_DIM;
Name           Null?     Type
-------------- --------- ------------
SEASON_ID      NOT NULL  VARCHAR2(10)
DESCRIPTION              VARCHAR2(30)
START_MONTH              VARCHAR2(10)
END_MONTH                VARCHAR2(10)
```



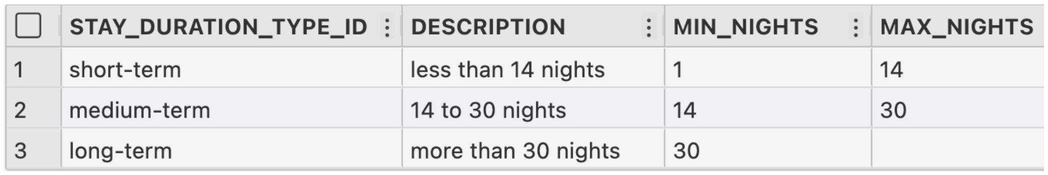
	SEASON_ID	DESCRIPTION	START_MONTH	END_MONTH
1	Spring	September to November	September	November
2	Summer	December to February	December	February
3	Autumn	March to May	March	May
4	Winter	June to August	June	August

Stay Duration Type Dimension:

```
-- Create Stay Duration Dimension
DROP TABLE STAY_DURATION_TYPE_DIM;
CREATE TABLE STAY_DURATION_TYPE_DIM (
    Stay_Duration_Type_ID varchar2(20) PRIMARY KEY,
    Description varchar2(30),
    Min_Nights number,
    Max_Nights number
);

-- Populate Dimension
INSERT INTO STAY_DURATION_TYPE_DIM
       values('short-term', 'less than 14 nights', 1, 14);
INSERT INTO STAY_DURATION_TYPE_DIM
       values('medium-term', '14 to 30 nights', 14, 30);
INSERT INTO STAY_DURATION_TYPE_DIM
       (Stay_Duration_Type_ID, Description, Min_Nights)
       values('long-term', 'more than 30 nights', 30);
```



```
SQL> DESCRIBE STAY_DURATION_TYPE_DIM;
Name                   Null?     Type
------------------------------------------------
STAY_DURATION_TYPE_ID  NOT NULL  VARCHAR2(20)
DESCRIPTION                      VARCHAR2(30)
MIN_NIGHTS                       NUMBER
MAX_NIGHTS                       NUMBER
```

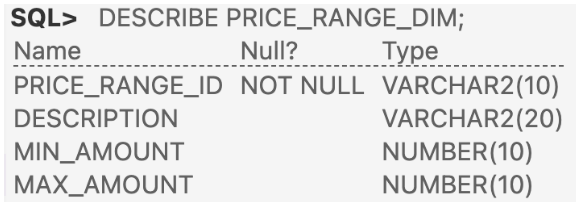


	STAY_DURATION_TYPE_ID	DESCRIPTION	MIN_NIGHTS	MAX_NIGHTS
1	short-term	less than 14 nights	1	14
2	medium-term	14 to 30 nights	14	30
3	long-term	more than 30 nights	30	

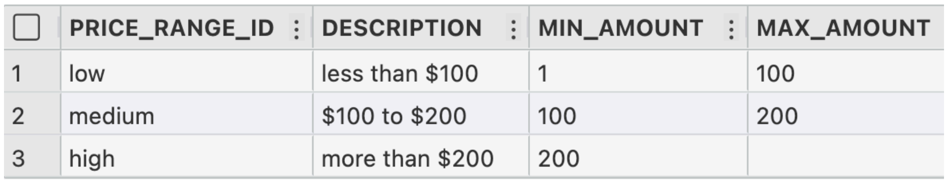
Price Range Dimension:

```
-- Create Price Range Dimension
DROP TABLE PRICE_RANGE_DIM;
CREATE TABLE PRICE_RANGE_DIM (
    Price_Range_ID varchar2(10) PRIMARY KEY,
    Description varchar2(20),
    Min_Amount number(10),
    Max_Amount number(10)
);
```

```
-- Populate Dimension
INSERT INTO PRICE_RANGE_DIM values('low', 'less than $100', 1, 100);
INSERT INTO PRICE_RANGE_DIM values('medium', '$100 to $200', 100, 200);
INSERT INTO PRICE_RANGE_DIM (Price_Range_ID, Description, Min_Amount)
       values('high', 'more than $200', 200);
```



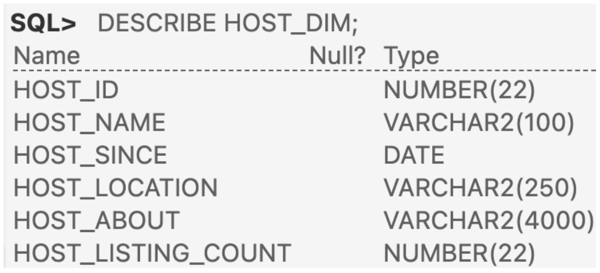
```
SQL> DESCRIBE PRICE_RANGE_DIM;
Name            Null?     Type
-------------------------------------------
PRICE_RANGE_ID  NOT NULL  VARCHAR2(10)
DESCRIPTION               VARCHAR2(20)
MIN_AMOUNT                NUMBER(10)
MAX_AMOUNT                NUMBER(10)
```



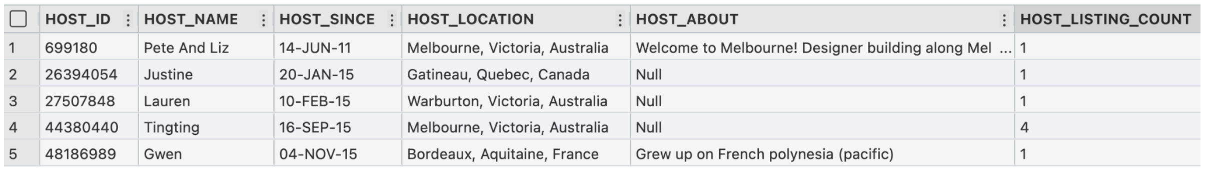
	PRICE_RANGE_ID	DESCRIPTION	MIN_AMOUNT	MAX_AMOUNT
1	low	less than $100	1	100
2	medium	$100 to $200	100	200
3	high	more than $200	200	

Host Dimension:

```
-- Create Host Dimension
CREATE TABLE HOST_DIM AS SELECT * FROM HOST;
```



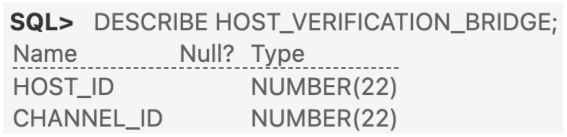
```
SQL> DESCRIBE HOST_DIM;
Name                 Null?  Type
-------------------------------------------
HOST_ID                     NUMBER(22)
HOST_NAME                   VARCHAR2(100)
HOST_SINCE                  DATE
HOST_LOCATION               VARCHAR2(250)
HOST_ABOUT                  VARCHAR2(4000)
HOST_LISTING_COUNT          NUMBER(22)
```



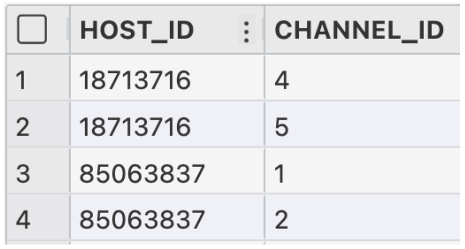
	HOST_ID	HOST_NAME	HOST_SINCE	HOST_LOCATION	HOST_ABOUT	HOST_LISTING_COUNT
1	699180	Pete And Liz	14-JUN-11	Melbourne, Victoria, Australia	Welcome to Melbourne! Designer building along Mel ...	1
2	26394054	Justine	20-JAN-15	Gatineau, Quebec, Canada	Null	1
3	27507848	Lauren	10-FEB-15	Warburton, Victoria, Australia	Null	1
4	44380440	Tingting	16-SEP-15	Melbourne, Victoria, Australia	Null	4
5	48186989	Gwen	04-NOV-15	Bordeaux, Aquitaine, France	Grew up on French polynesia (pacific)	1

Host Verification Bridge:

```
-- Create Host Verification Bridge
CREATE TABLE HOST_VERIFICATION_BRIDGE AS SELECT * FROM HOST_VERIFICATION;
```



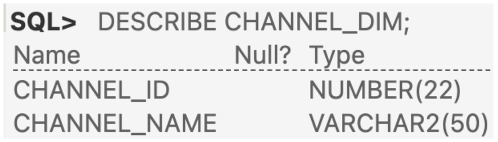
```
SQL> DESCRIBE HOST_VERIFICATION_BRIDGE;
Name          Null?  Type
--------------------------------
HOST_ID              NUMBER(22)
CHANNEL_ID           NUMBER(22)
```



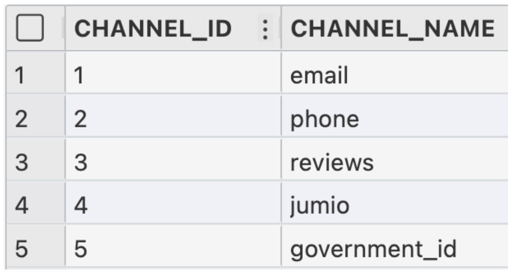
	HOST_ID	CHANNEL_ID
1	18713716	4
2	18713716	5
3	85063837	1
4	85063837	2

Channel Dimension:

```
-- Create Channel Dimension
CREATE TABLE CHANNEL_DIM AS SELECT * FROM MSTAY.CHANNEL;
```



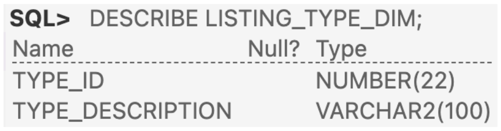
```
SQL> DESCRIBE CHANNEL_DIM;
Name                 Null?  Type
------------------------------------------
CHANNEL_ID                  NUMBER(22)
CHANNEL_NAME                VARCHAR2(50)
```



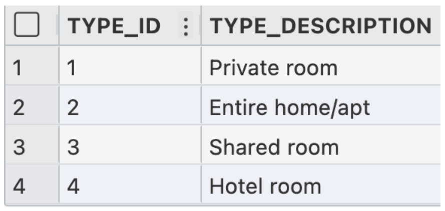
	CHANNEL_ID	CHANNEL_NAME
1	1	email
2	2	phone
3	3	reviews
4	4	jumio
5	5	government_id

Listing Type Dimension:

```
-- Create Listing Type Dimension
CREATE TABLE LISTING_TYPE_DIM AS
SELECT * FROM MSTAY.LISTING_TYPE;
```



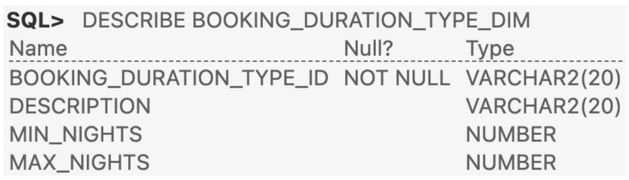
```
SQL> DESCRIBE LISTING_TYPE_DIM;
Name                   Null?  Type
--------------------------------------------
TYPE_ID                       NUMBER(22)
TYPE_DESCRIPTION              VARCHAR2(100)
```



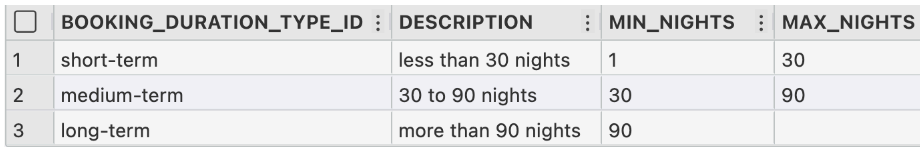
	TYPE_ID	TYPE_DESCRIPTION
1	1	Private room
2	2	Entire home/apt
3	3	Shared room
4	4	Hotel room

Booking Duration Type Dimension:

```
-- Create Booking Duration Type
DROP TABLE BOOKING_DURATION_TYPE_DIM;
CREATE TABLE BOOKING_DURATION_TYPE_DIM (
    Booking_Duration_type_ID varchar2(20) PRIMARY KEY,
    Description varchar2(20),
    Min_Nights number,
    Max_Nights number
);

-- Populate Dimension
INSERT INTO BOOKING_DURATION_TYPE_DIM
       values('short-term', 'less than 30 nights', 1, 30);
INSERT INTO BOOKING_DURATION_TYPE_DIM
       values('medium-term', '30 to 90 nights', 30, 90);
INSERT INTO BOOKING_DURATION_TYPE_DIM
       (Booking_Duration_type_ID, Description, Min_Nights)
       values('long-term', 'more than 90 nights', 90);
```



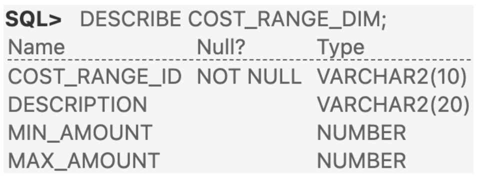
```
SQL>  DESCRIBE BOOKING_DURATION_TYPE_DIM
Name                      Null?     Type
----------------------------------------------------
BOOKING_DURATION_TYPE_ID  NOT NULL  VARCHAR2(20)
DESCRIPTION                         VARCHAR2(20)
MIN_NIGHTS                          NUMBER
MAX_NIGHTS                          NUMBER
```



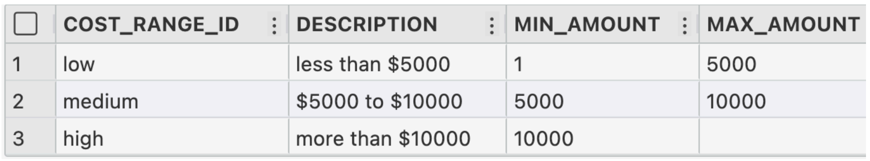
☐	BOOKING_DURATION_TYPE_ID	DESCRIPTION	MIN_NIGHTS	MAX_NIGHTS
1	short-term	less than 30 nights	1	30
2	medium-term	30 to 90 nights	30	90
3	long-term	more than 90 nights	90	

Cost Range Dimension:

```
DROP TABLE COST_RANGE_DIM;
CREATE TABLE COST_RANGE_DIM (
    Cost_Range_ID varchar2(10) PRIMARY KEY,
    Description varchar2(20),
    Min_Amount number,
    Max_Amount number
);

-- Populate Dimension
INSERT INTO COST_RANGE_DIM values('low', 'less than $5000', 1, 5000);
INSERT INTO COST_RANGE_DIM values('medium', '$5000 to $10000', 5000, 10000);
INSERT INTO COST_RANGE_DIM (Cost_Range_ID, Description, Min_Amount)
       values('high', 'more than $10000', 10000);
```



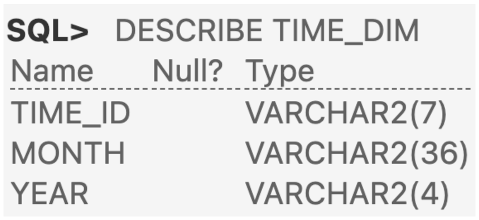
```
SQL>  DESCRIBE COST_RANGE_DIM;
Name           Null?     Type
------------------------------------------
COST_RANGE_ID  NOT NULL  VARCHAR2(10)
DESCRIPTION              VARCHAR2(20)
MIN_AMOUNT               NUMBER
MAX_AMOUNT               NUMBER
```



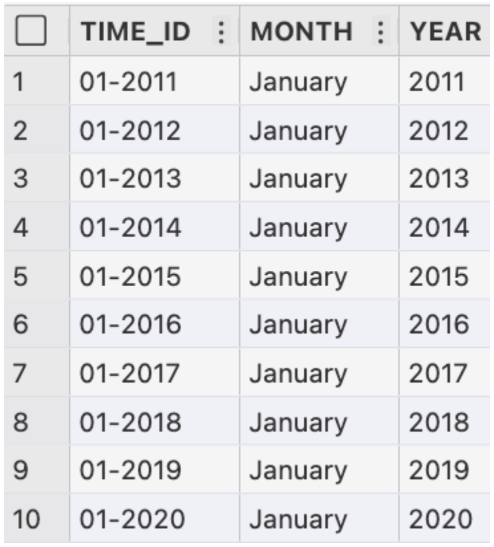
☐	COST_RANGE_ID	DESCRIPTION	MIN_AMOUNT	MAX_AMOUNT
1	low	less than $5000	1	5000
2	medium	$5000 to $10000	5000	10000
3	high	more than $10000	10000	

Time Dimension:

```
DROP TABLE TIME_DIM;
CREATE TABLE TIME_DIM AS
SELECT DISTINCT
    (TO_CHAR(Booking_Date, 'MM') || '-' || TO_CHAR(Booking_Date, 'YYYY')) AS
Time_ID,
    TO_CHAR(Booking_Date, 'Month') AS Month,
    TO_CHAR(Booking_Date, 'YYYY') AS Year
FROM BOOKING
UNION
SELECT DISTINCT
    (TO_CHAR(Listing_Date, 'MM') || '-' || TO_CHAR(Listing_Date, 'YYYY')) AS
Time_ID,
    TO_CHAR(Listing_Date, 'Month') AS Month,
    TO_CHAR(Listing_Date, 'YYYY') AS Year
FROM LISTING
UNION
SELECT DISTINCT
    (TO_CHAR(Review_Date, 'MM') || '-' || TO_CHAR(Review_Date, 'YYYY')) AS
Time_ID,
    TO_CHAR(Review_Date, 'Month') AS Month,
    TO_CHAR(Review_Date, 'YYYY') AS Year
FROM REVIEW;
```



```
SQL> DESCRIBE TIME_DIM
Name      Null?  Type
-----------------------------
TIME_ID          VARCHAR2(7)
MONTH            VARCHAR2(36)
YEAR             VARCHAR2(4)
```

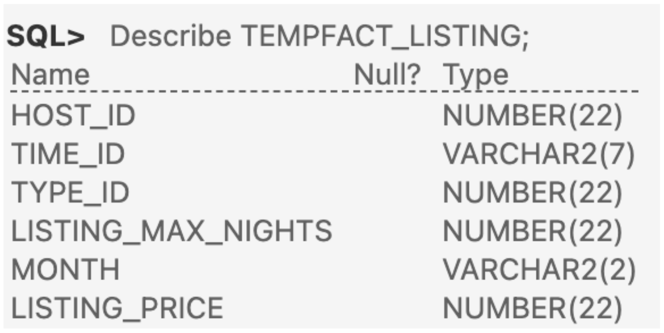


	TIME_ID	MONTH	YEAR
1	01-2011	January	2011
2	01-2012	January	2012
3	01-2013	January	2013
4	01-2014	January	2014
5	01-2015	January	2015
6	01-2016	January	2016
7	01-2017	January	2017
8	01-2018	January	2018
9	01-2019	January	2019
10	01-2020	January	2020

Listing Fact Table Implementation

Create Temp Fact - Listing:

```
DROP TABLE TEMPFACT_LISTING;
CREATE TABLE TEMPFACT_LISTING AS
SELECT  Host_ID,
        TO_CHAR(Listing_Date, 'MM') || '-' || TO_CHAR(Listing_Date, 'YYYY') AS
Time_ID,
        Type_ID,
        Listing_Max_Nights, -- For stay_duration_type_id
        TO_CHAR(Listing_Date, 'MM') AS Month, -- For season_id
        Listing_Price -- For price_range_id
FROM LISTING;
SELECT * FROM TEMPFACT_LISTING;
```

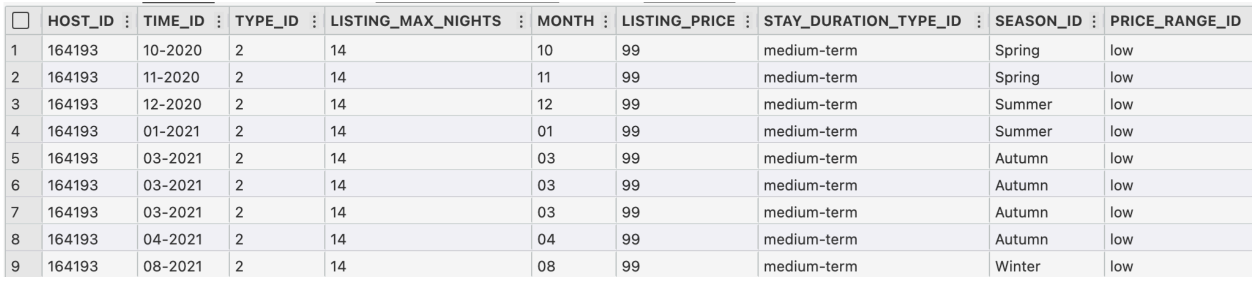


```
SQL>  Describe TEMPFACT_LISTING;
 Name                        Null?  Type
 ------------------------------------------
 HOST_ID                            NUMBER(22)
 TIME_ID                            VARCHAR2(7)
 TYPE_ID                            NUMBER(22)
 LISTING_MAX_NIGHTS                 NUMBER(22)
 MONTH                              VARCHAR2(2)
 LISTING_PRICE                      NUMBER(22)
```

Populate Temp Fact Table - Listing:

```
ALTER TABLE TEMPFACT_LISTING ADD(Stay_duration_type_ID varchar(20));

UPDATE TEMPFACT_LISTING SET Stay_duration_type_ID = 'short-term' WHERE
Listing_Max_Nights >= 1 AND Listing_Max_Nights < 14;
UPDATE TEMPFACT_LISTING SET Stay_duration_type_ID = 'medium-term' WHERE
Listing_Max_Nights >= 14 AND Listing_Max_Nights <= 30;
UPDATE TEMPFACT_LISTING SET Stay_duration_type_ID = 'long-term' WHERE
Listing_Max_Nights > 30;

ALTER TABLE TEMPFACT_LISTING ADD(Season_ID varchar(20));

UPDATE TEMPFACT_LISTING SET Season_ID = 'Spring' WHERE TO_NUMBER(Month) >= 9
AND TO_NUMBER(Month) <= 11;
UPDATE TEMPFACT_LISTING SET Season_ID = 'Summer' WHERE TO_NUMBER(Month) IN (12,
1, 2); -- Because 12 and 1,2 and not continuous
UPDATE TEMPFACT_LISTING SET Season_ID = 'Autumn' WHERE TO_NUMBER(Month) >= 3
AND TO_NUMBER(Month) <= 5;
UPDATE TEMPFACT_LISTING SET Season_ID = 'Winter' WHERE TO_NUMBER(Month) >= 6
AND TO_NUMBER(Month) <= 8;
```

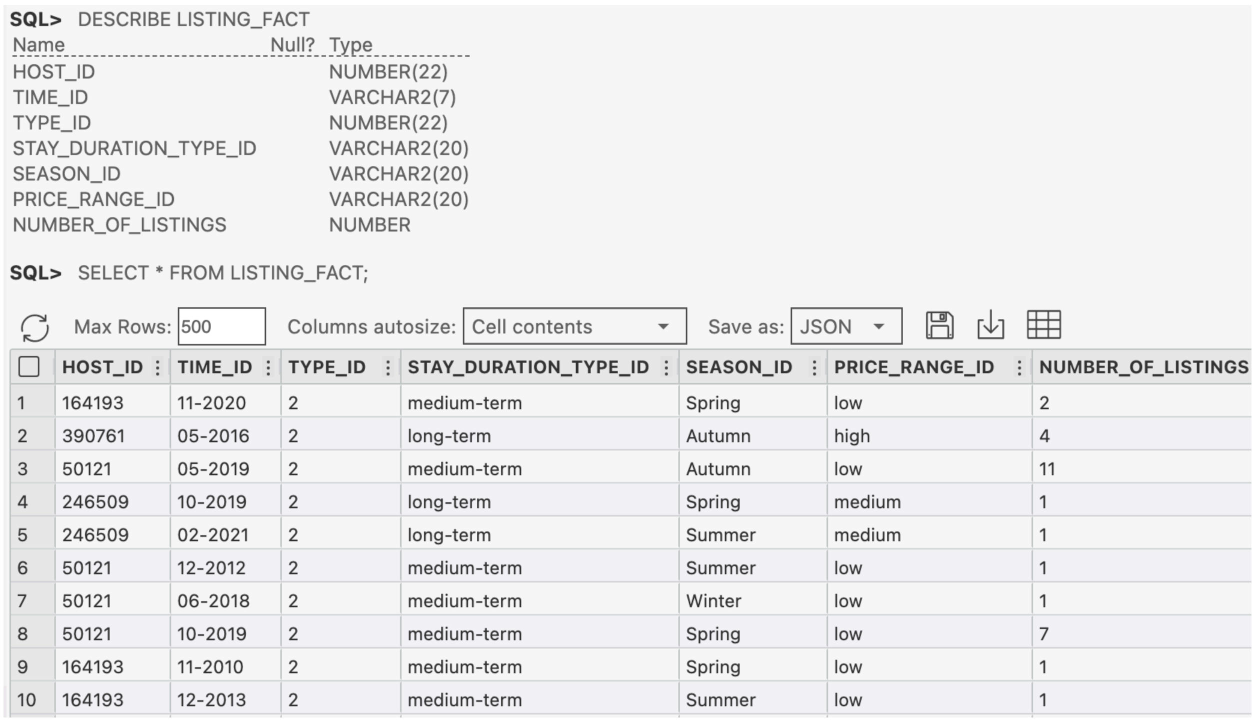
```
ALTER TABLE TEMPFACT_LISTING ADD(Price_range_ID varchar(20));

UPDATE TEMPFACT_LISTING SET Price_range_ID = 'low' WHERE Listing_Price < 100;
UPDATE TEMPFACT_LISTING SET Price_range_ID = 'medium'
       WHERE Listing_Price >= 100 AND Listing_Price <= 200;
UPDATE TEMPFACT_LISTING SET Price_range_ID = 'high' WHERE Listing_Price > 200;
```



	HOST_ID	TIME_ID	TYPE_ID	LISTING_MAX_NIGHTS	MONTH	LISTING_PRICE	STAY_DURATION_TYPE_ID	SEASON_ID	PRICE_RANGE_ID
1	164193	10-2020	2	14	10	99	medium-term	Spring	low
2	164193	11-2020	2	14	11	99	medium-term	Spring	low
3	164193	12-2020	2	14	12	99	medium-term	Summer	low
4	164193	01-2021	2	14	01	99	medium-term	Summer	low
5	164193	03-2021	2	14	03	99	medium-term	Autumn	low
6	164193	03-2021	2	14	03	99	medium-term	Autumn	low
7	164193	03-2021	2	14	03	99	medium-term	Autumn	low
8	164193	04-2021	2	14	04	99	medium-term	Autumn	low
9	164193	08-2021	2	14	08	99	medium-term	Winter	low

Fact Table - Listing:

```
DROP TABLE LISTING_FACT;
CREATE TABLE LISTING_FACT AS
SELECT Host_ID, Time_ID, Type_ID, Stay_Duration_Type_ID, Season_ID,
Price_Range_ID, COUNT(*) AS Number_of_listings
FROM TEMPFACT_LISTING
GROUP BY Host_ID, Time_ID, Type_ID, Stay_Duration_Type_ID, Season_ID,
Price_Range_ID;
```



```
SQL>  DESCRIBE LISTING_FACT
Name                           Null?  Type
------------------------------------------------
HOST_ID                               NUMBER(22)
TIME_ID                               VARCHAR2(7)
TYPE_ID                               NUMBER(22)
STAY_DURATION_TYPE_ID                 VARCHAR2(20)
SEASON_ID                             VARCHAR2(20)
PRICE_RANGE_ID                        VARCHAR2(20)
NUMBER_OF_LISTINGS                    NUMBER

SQL>  SELECT * FROM LISTING_FACT;
```

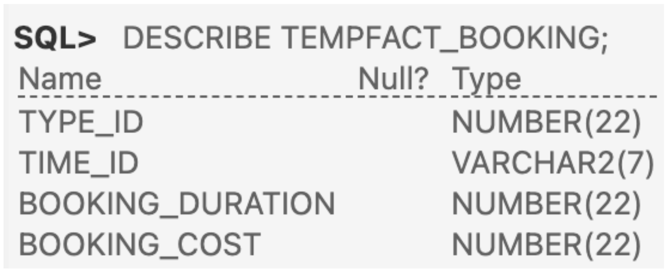
Max Rows: 500   Columns autosize: Cell contents   Save as: JSON

	HOST_ID	TIME_ID	TYPE_ID	STAY_DURATION_TYPE_ID	SEASON_ID	PRICE_RANGE_ID	NUMBER_OF_LISTINGS
1	164193	11-2020	2	medium-term	Spring	low	2
2	390761	05-2016	2	long-term	Autumn	high	4
3	50121	05-2019	2	medium-term	Autumn	low	11
4	246509	10-2019	2	long-term	Spring	medium	1
5	246509	02-2021	2	long-term	Summer	medium	1
6	50121	12-2012	2	medium-term	Summer	low	1
7	50121	06-2018	2	medium-term	Winter	low	1
8	50121	10-2019	2	medium-term	Spring	low	7
9	164193	11-2010	2	medium-term	Spring	low	1
10	164193	12-2013	2	medium-term	Summer	low	1

Booking Fact Table Implementation

Create Temp Fact - Booking:

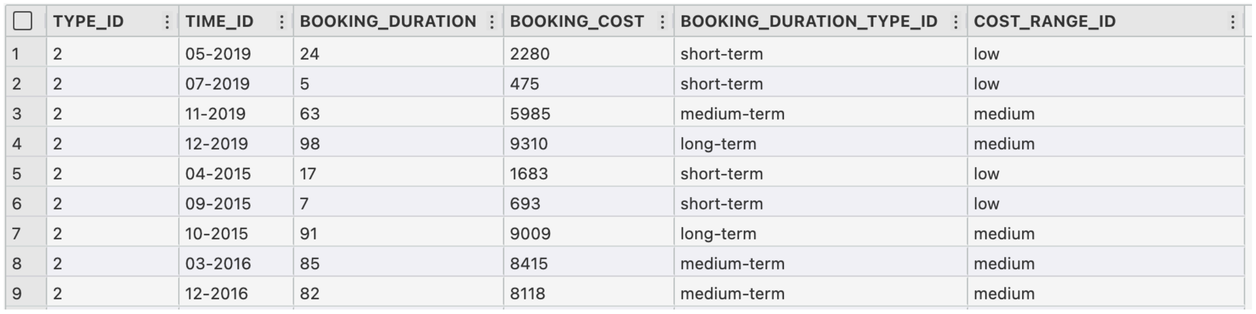
```
DROP TABLE TEMPFACT_BOOKING;
CREATE TABLE TEMPFACT_BOOKING AS
SELECT l.Type_ID,
    TO_CHAR(b.Booking_Date, 'MM') || '-' || TO_CHAR(b.Booking_Date, 'YYYY') AS
Time_ID,
    b.Booking_Duration,
    b.Booking_Cost
FROM booking b
JOIN LISTING l ON b.Listing_ID = l.Listing_ID;
```



```
SQL>  DESCRIBE TEMPFACT_BOOKING;
Name                    Null?  Type
-------------------------------------------
TYPE_ID                        NUMBER(22)
TIME_ID                        VARCHAR2(7)
BOOKING_DURATION               NUMBER(22)
BOOKING_COST                   NUMBER(22)
```

Populate Temp Fact Table - Booking:

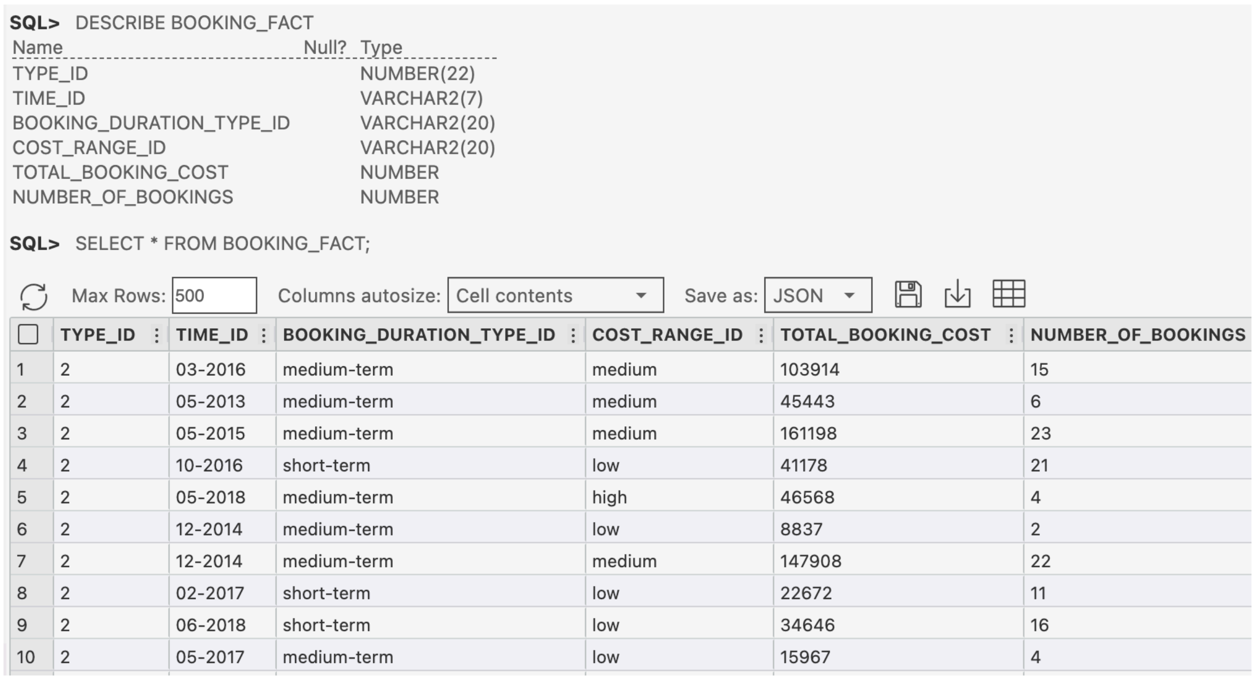
```
ALTER TABLE TEMPFACT_BOOKING ADD(Booking_duration_type_ID varchar(20));

UPDATE TEMPFACT_BOOKING SET Booking_duration_type_ID = 'short-term' WHERE
Booking_Duration >= 1 AND Booking_Duration < 30;
UPDATE TEMPFACT_BOOKING SET Booking_duration_type_ID = 'medium-term' WHERE
Booking_Duration >= 30 AND Booking_Duration <= 90;
UPDATE TEMPFACT_BOOKING SET Booking_duration_type_ID = 'long-term' WHERE
Booking_Duration > 90;

ALTER TABLE TEMPFACT_BOOKING ADD(Cost_range_ID varchar(20));

UPDATE TEMPFACT_BOOKING SET Cost_range_ID = 'low' WHERE Booking_Cost < 5000;
UPDATE TEMPFACT_BOOKING SET Cost_range_ID = 'medium' WHERE Booking_Cost >= 5000
AND Booking_Cost <= 10000;
UPDATE TEMPFACT_BOOKING SET Cost_range_ID = 'high' WHERE Booking_Cost > 10000;
```



	TYPE_ID	TIME_ID	BOOKING_DURATION	BOOKING_COST	BOOKING_DURATION_TYPE_ID	COST_RANGE_ID
1	2	05-2019	24	2280	short-term	low
2	2	07-2019	5	475	short-term	low
3	2	11-2019	63	5985	medium-term	medium
4	2	12-2019	98	9310	long-term	medium
5	2	04-2015	17	1683	short-term	low
6	2	09-2015	7	693	short-term	low
7	2	10-2015	91	9009	long-term	medium
8	2	03-2016	85	8415	medium-term	medium
9	2	12-2016	82	8118	medium-term	medium

Fact Table - Booking:

```
DROP TABLE BOOKING_FACT;
CREATE TABLE BOOKING_FACT AS
SELECT  Type_ID, Time_ID,  Booking_duration_type_ID, Cost_range_ID,
SUM(Booking_Cost) as Total_Booking_Cost, count(Time_ID) as Number_of_bookings
FROM TEMPFACT_BOOKING
GROUP BY Type_ID, Time_ID,  Booking_duration_type_ID, Cost_range_ID;
```



```
SQL> DESCRIBE BOOKING_FACT
Name                        Null?  Type
--------------------------------------------
TYPE_ID                            NUMBER(22)
TIME_ID                            VARCHAR2(7)
BOOKING_DURATION_TYPE_ID           VARCHAR2(20)
COST_RANGE_ID                      VARCHAR2(20)
TOTAL_BOOKING_COST                 NUMBER
NUMBER_OF_BOOKINGS                 NUMBER

SQL> SELECT * FROM BOOKING_FACT;
```

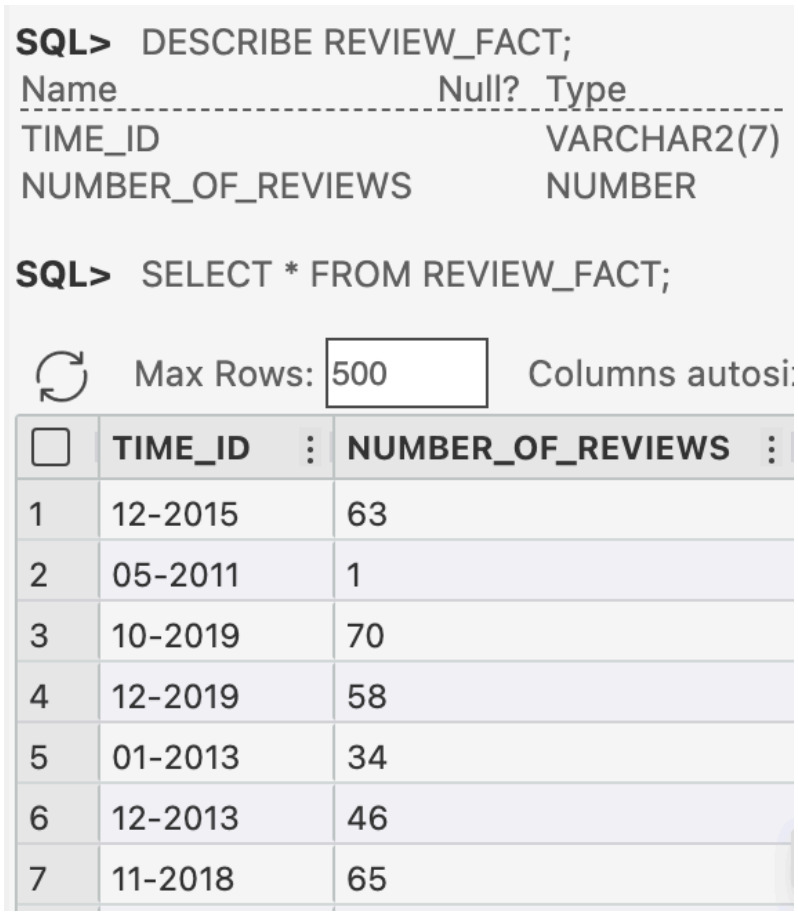
Max Rows: 500 Columns autosize: Cell contents Save as: JSON

	TYPE_ID	TIME_ID	BOOKING_DURATION_TYPE_ID	COST_RANGE_ID	TOTAL_BOOKING_COST	NUMBER_OF_BOOKINGS
1	2	03-2016	medium-term	medium	103914	15
2	2	05-2013	medium-term	medium	45443	6
3	2	05-2015	medium-term	medium	161198	23
4	2	10-2016	short-term	low	41178	21
5	2	05-2018	medium-term	high	46568	4
6	2	12-2014	medium-term	low	8837	2
7	2	12-2014	medium-term	medium	147908	22
8	2	02-2017	short-term	low	22672	11
9	2	06-2018	short-term	low	34646	16
10	2	05-2017	medium-term	low	15967	4

Review Fact Table Implementation

Create Fact Table:

```
DROP TABLE REVIEW_FACT;
CREATE TABLE REVIEW_FACT AS
    SELECT
        TO_CHAR(REVIEW_DATE, 'MM')
        || '-'
        || TO_CHAR(REVIEW_DATE, 'YYYY') AS TIME_ID,
        COUNT(REVIEW_ID)                AS NUMBER_OF_REVIEWS
    FROM
        REVIEW
    GROUP BY
        TO_CHAR(REVIEW_DATE, 'MM')
        || '-'
        || TO_CHAR(REVIEW_DATE, 'YYYY');
```



```
SQL> DESCRIBE REVIEW_FACT;
Name                          Null?  Type
--------------------------------------------------
TIME_ID                              VARCHAR2(7)
NUMBER_OF_REVIEWS                    NUMBER

SQL> SELECT * FROM REVIEW_FACT;
```

Max Rows: 500 Columns autosi

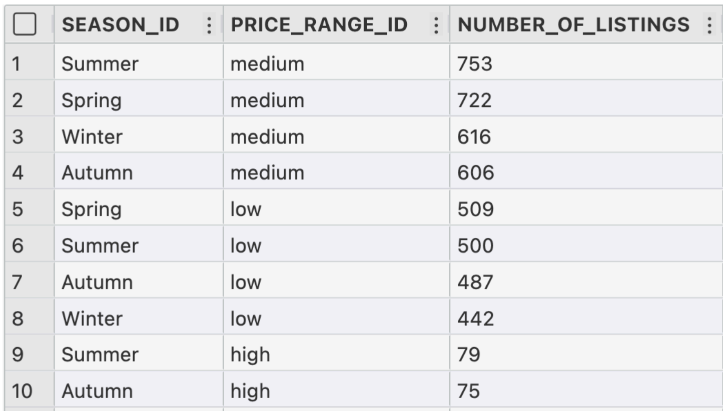
	TIME_ID	NUMBER_OF_REVIEWS
1	12-2015	63
2	05-2011	1
3	10-2019	70
4	12-2019	58
5	01-2013	34
6	12-2013	46
7	11-2018	65

Section D: Data Analytic Stage

Descriptive Analysis

Observation 1: Medium Price Range Has the Most Listings, especially During Summer and Spring

```
SELECT
    SEASON_ID,
    PRICE_RANGE_ID,
    SUM(NUMBER_OF_LISTINGS) AS NUMBER_OF_LISTINGS
FROM
    LISTING_FACT F
GROUP BY
    SEASON_ID,
    PRICE_RANGE_ID
ORDER BY
    NUMBER_OF_LISTINGS DESC;
```

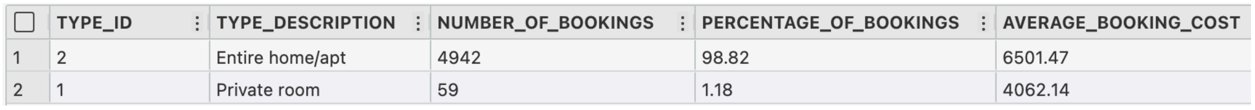


	SEASON_ID	PRICE_RANGE_ID	NUMBER_OF_LISTINGS
1	Summer	medium	753
2	Spring	medium	722
3	Winter	medium	616
4	Autumn	medium	606
5	Spring	low	509
6	Summer	low	500
7	Autumn	low	487
8	Winter	low	442
9	Summer	high	79
10	Autumn	high	75

Listings are concentrated in the medium price range across all seasons where they appear in the first 4 records. On the other hand, the high-price range consistently represents the smallest share. Summer stands out as the peak season overall, with 753 medium-price listings and 79 in the high-price range, the most in both categories. However, in the low-price range, spring slightly surpasses summer, with 509 listings compared to 500—a difference of 9 listings. Despite spring's edge in the low-price category, summer still dominates in terms of overall activity, making it the busiest season for listings across all price ranges.

Observation 2: There only exists 2 Types of Listing with a difference of 2.5k in Average Booking Cost

```
SELECT
    F.TYPE_ID,
    T.TYPE_DESCRIPTION,
    SUM(F.NUMBER_OF_BOOKINGS)           AS NUMBER_OF_BOOKINGS,
    ROUND((SUM(F.NUMBER_OF_BOOKINGS) * 100.0) / (
        SELECT
            SUM(NUMBER_OF_BOOKINGS)
        FROM
            BOOKING_FACT
    ), 2) AS PERCENTAGE_OF_BOOKINGS,
    ROUND(SUM(TOTAL_BOOKING_COST) / SUM(NUMBER_OF_BOOKINGS),
    2) AS AVERAGE_BOOKING_COST
FROM
    BOOKING_FACT     F,
    LISTING_TYPE_DIM T
WHERE
    F.TYPE_ID = T.TYPE_ID
GROUP BY
    F.TYPE_ID,
    T.TYPE_DESCRIPTION;
```

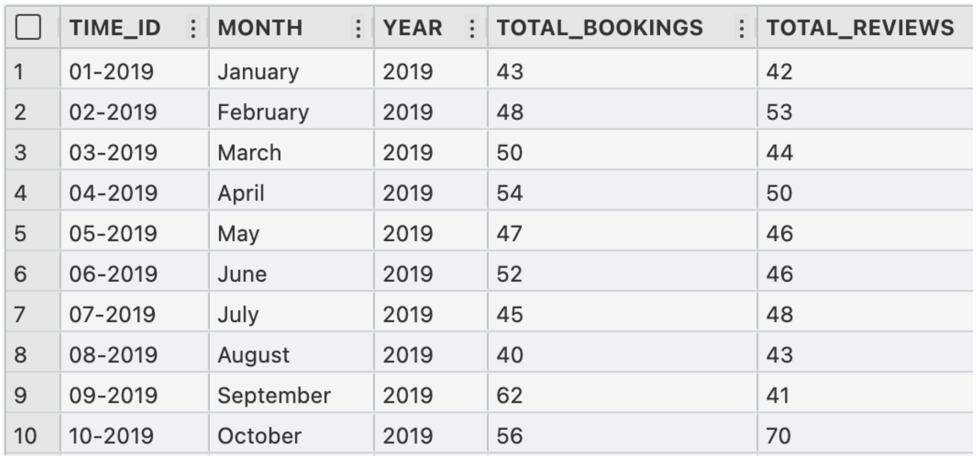


	TYPE_ID	TYPE_DESCRIPTION	NUMBER_OF_BOOKINGS	PERCENTAGE_OF_BOOKINGS	AVERAGE_BOOKING_COST
1	2	Entire home/apt	4942	98.82	6501.47
2	1	Private room	59	1.18	4062.14

There are only two types of listings currently being utilized, despite there being a total of four available types. Furthermore, entire homes/apartments have significantly more bookings, 4942 which takes up 98.82% of all the bookings compared to private rooms which only has 59 bookings. Despite the higher average booking cost of $6,501.47 for entire homes/apartments, they attract a much larger number of bookings. This suggests that customers may prioritize space and amenities over price, indicating a strong preference for these accommodations even at a higher cost.

Observation 3 : Analysis of Booking and Review Data (2019-2021)

```
SELECT
    t.Time_ID,
    t.Month,
    t.Year,
    COALESCE(b.Number_of_Bookings, 0) AS total_bookings,
    COALESCE(r.Number_of_Reviews, 0) AS total_reviews
FROM TIME_DIM t
LEFT JOIN (
    SELECT Time_ID, SUM(NUMBER_OF_BOOKINGS) AS Number_of_Bookings
    FROM BOOKING_FACT
    GROUP BY Time_ID
) b ON t.Time_ID = b.Time_ID
LEFT JOIN (
    SELECT
    Time_ID, SUM(NUMBER_OF_REVIEWS) AS Number_of_Reviews
    FROM REVIEW_FACT
    GROUP BY Time_ID
) r ON t.Time_ID = r.Time_ID
WHERE t.Year  IN (2019, 2020, 2021)
ORDER BY
    t.Year,
    TO_NUMBER(TO_CHAR(TO_DATE(t.Month, 'Month'), 'MM'));
```



	TIME_ID	MONTH	YEAR	TOTAL_BOOKINGS	TOTAL_REVIEWS
1	01-2019	January	2019	43	42
2	02-2019	February	2019	48	53
3	03-2019	March	2019	50	44
4	04-2019	April	2019	54	50
5	05-2019	May	2019	47	46
6	06-2019	June	2019	52	46
7	07-2019	July	2019	45	48
8	08-2019	August	2019	40	43
9	09-2019	September	2019	62	41
10	10-2019	October	2019	56	70

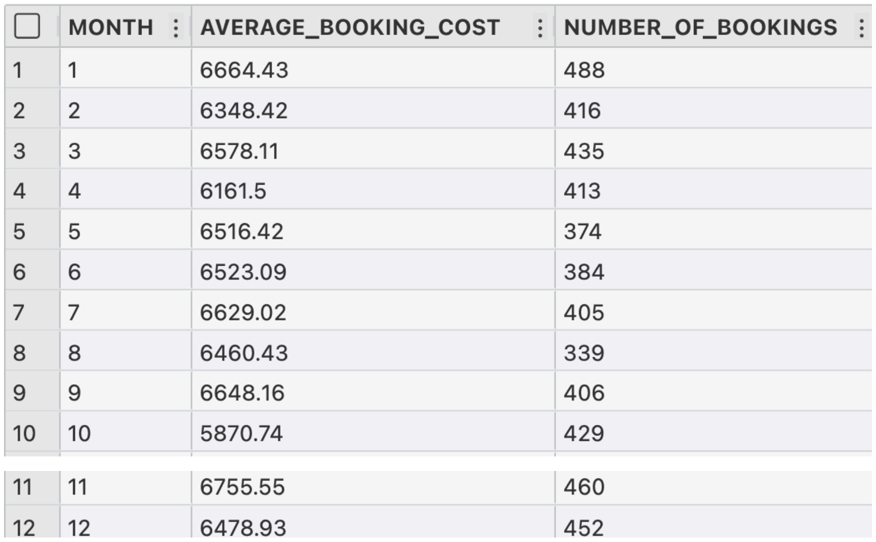
The booking counts exhibit a generally increasing trend, peaking at 66 bookings in November 2019, while the lowest count occurs in January with 43 bookings. This year reflects consistent demand, particularly in the latter half. However in 2020, there's a sharp decline, particularly in April (7 bookings), which can be attributed to the global pandemic and associated restrictions. A gradual recovery is observed, especially in the first half of the year. However, the booking numbers still remain lower than the pre-pandemic levels of 2019.

Review trends in 2019 show a close relationship with booking activity, but with a noticeable one-month buffer period. For instance, while October 2019 saw a decrease in bookings to 56, the number of reviews spiked to 70, reflecting reviews being submitted for the September bookings, which had been higher at 62. This pattern suggests that reviews often trail bookings by about a month, as customers tend to submit reviews after completing their stays. This buffer effect is visible throughout 2019, where months with high booking volumes are followed by a surge in reviews the next month.

However, this buffer trend became inconsistent during and after the pandemic. In 2020, review counts declined sharply along with bookings, especially in April 2020, which saw only 8 reviews. By July 2021, despite some bookings still being made (26 in July), no reviews were submitted. This lack of reviews continued through October 2021, indicating that customer engagement in submitting feedback dropped.

Observation 4: Distribution of Number of Bookings and Average Booking Costs for each Month

```
SELECT
    EXTRACT(MONTH FROM TO_DATE(F.TIME_ID, 'MM-YYYY'))      AS MONTH,
    ROUND(SUM(TOTAL_BOOKING_COST) / SUM(NUMBER_OF_BOOKINGS), 2) AS
AVERAGE_BOOKING_COST,
    SUM(NUMBER_OF_BOOKINGS)           AS NUMBER_OF_BOOKINGS
FROM
    BOOKING_FACT F
    JOIN TIME_DIM T
    ON F.TIME_ID = T.TIME_ID
GROUP BY
    EXTRACT(MONTH FROM TO_DATE(F.TIME_ID, 'MM-YYYY'))
ORDER BY
    Month;
```



	MONTH	AVERAGE_BOOKING_COST	NUMBER_OF_BOOKINGS
1	1	6664.43	488
2	2	6348.42	416
3	3	6578.11	435
4	4	6161.5	413
5	5	6516.42	374
6	6	6523.09	384
7	7	6629.02	405
8	8	6460.43	339
9	9	6648.16	406
10	10	5870.74	429
11	11	6755.55	460
12	12	6478.93	452

The average booking cost fluctuates throughout the year, peaking at $6,755.55 in November and reaching a low of $6,161.50 in April. Notably, January sees the highest number of bookings at 488, potentially due to a post-holiday travel surge. Conversely, August experiences the lowest booking volume at 339, which may reflect a decrease in travel activity as summer vacations come to an end.