ADVANCED DATABASE MANAGEMENT PROJECT

Final Project for Advanced Database Management System (ISM6218)

Team Members:

Vijaya Kumar Gavidi
Jaswanth Buggana
Sai Kumar Kanna
Sumanth Reddy Bajjuri
Ravi Varma Allani

Table of Contents

Purpose

The project deals with hotel management system. Starting from creating of tables to performance tuning and DBA scripts, everything is included in this document. In performance tuning parallelism and indexing topics are discussed.

Topic Area	Description	Group Member	Weight
Database Design	This part should include a logical database design (for the relational model), using normalization to control redundancy and integrity constraints for data quality.	Vijaya Kumar Gavidi, Ravi Varma Allani	30%
Query Writing	This part is another chance to write SQL queries, explore transactions, and even do some database programming for stored procedures.	Vijaya Kumar Gavidi, Sumanth Reddy Bajjuri	25%
Performance Tuning	In this section, you can capitalize and extend your prior experiments with indexing, optimizer modes, partitioning, parallel execution and any other techniques you want to further explore.	Jaswanth Buggana	25%
DBA Scripts	Here you are free to explore any other topics of interest. Suggestions include DBA scripts, database security, interface design, data visualization, data mining, and NoSQL databases.	Sai Kumar Kanna	20%

Narrative

Introduction:

A hotel, for example, has several entities involved like customers, rooms, bookings, services, transactions, invoices and addresses. It is critical to digitalize the information for greater strategic planning, faster administration, managing the relationship between customers, employees and the management, and offering more services in the future.

There are many customers, services, and bookings at the hotel. When a customer books a reservation, he or she is provided a unique customer ID. Each customer's entire name, gender, phone number, address, payment method, service type, check-in & check-out time, room type, and room price are all included. Each customer can book multiple reservations.

Entities Identified:

- Customer
- Room
- Booking
- Services
- Reservations
- Line
- Transactions
- Satisfaction
- Address
- Invoice

Entities Attributes

1. **Customer:-**

   customer_id (primary key)
   first_name
   last_name
   Gender
   phone_number

2. **Booking:-**

   Booking_id (primary key)
   Book_type
   Book_date
   Customer_id (Foreign key from Customer)

3. **Reservation:-**

   Res_id (primary key)
   check_in_date
   check_out_date (number of days)
   No_of_days
   Customer_id (Foreign key from Customer)
   Booking_id (Foreign key from Booking)

4. **Room:-**

   Room_no
   Room_type
   Bed_type
   No_of_occupants
   Room_price
   Customer_id (Foreign key from Customer)
   res_id (Foreign key from Reservation)

5. **Address:-**

   Street
   City
   State
   Country
   Customer_id (Foreign key from Customer)
   Zip_code

6. **Services:-**

Service_id (primary key)
Service_type
Service_cost

7. **Invoice:-**

Invoice_No(primary key)
Res_id
customer_id(Foreign key from Customer)

8. **Line:-**

Invoice_No (Foreign key from Invoice)
Service_id (Foreign key from Services)
Service_quantityy
res_id (Foreign key from Reservation)

9. **Transaction:-**

Trans_No (primary key)
Payment_Method
Payment_Date
Invoice_no (Foreign key from Invoice)
Customer_id (Foreign key from Customer)

10. **Satisfaction:-**

Satisfaction_ID
Satisfaction_level
Trans_no (Foreign key from Transactions)

Entity Relationship Diagram

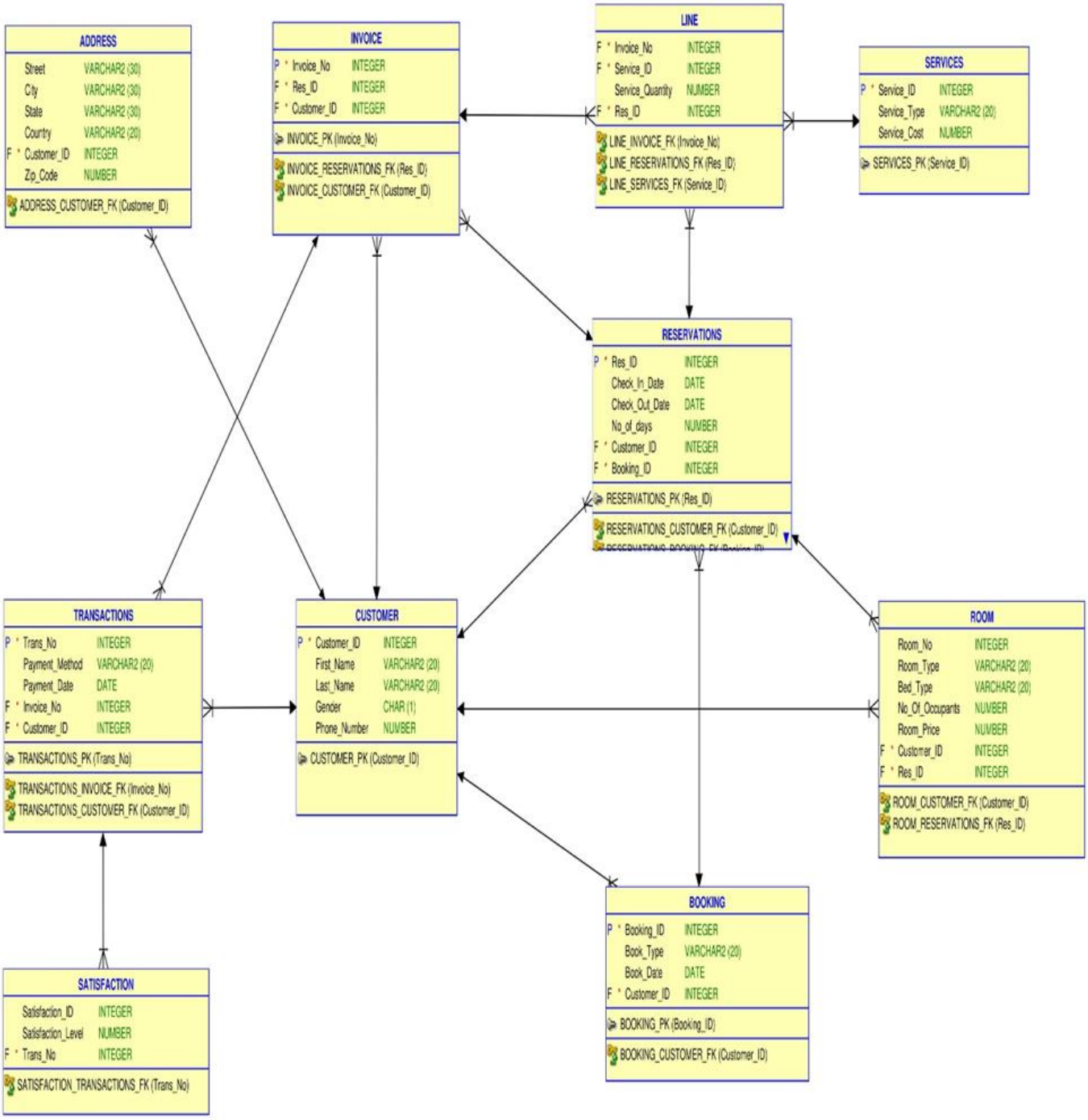


Table Views

Customer: This table cotains the all details about each customer such as customer id (primary key), first name, last name, gender, phone number.

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_...	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	CUSTOMER_ID	NUMBER	No	(null)	1	(null)
2	FIRST_NAME	VARCHAR2(20 BYTE)	Yes	(null)	2	(null)
3	LAST_NAME	VARCHAR2(20 BYTE)	Yes	(null)	3	(null)
4	GENDER	VARCHAR2(20 BYTE)	Yes	(null)	4	(null)
5	PHONE_NUMBER	NUMBER	Yes	(null)	5	(null)

Booking: This table contains all the details of the bookings such as booking id (primary key), booking type, booking date and customer id

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	BOOKING_ID	NUMBER(38,0)	No	(null)	1	(null)
2	BOOK_TYPE	VARCHAR2(20 BYTE)	Yes	(null)	2	(null)
3	BOOK_DATE	VARCHAR2(20 BYTE)	Yes	(null)	3	(null)
4	CUSTOMER_ID	NUMBER	Yes	(null)	4	(null)

Reservation: This table has information about all the reservations that are available such as Reservation id (primary key), check-in date, check out date, no of days, Customer id & Booking id

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	RES_ID	NUMBER	No	(null)	1	(null)
2	CHECK_IN_DATE	VARCHAR2(20 BYTE)	Yes	(null)	2	(null)
3	CHECK_OUT_DATE	VARCHAR2(20 BYTE)	Yes	(null)	3	(null)
4	NO_OF_DAYS	NUMBER	Yes	(null)	4	(null)
5	CUSTOMER_ID	NUMBER	Yes	(null)	5	(null)
6	BOOKING_ID	NUMBER(38,0)	Yes	(null)	6	(null)

Room: This table contains information of all the available rooms like Room number, Room type, Bed type, No of occupants, Room price, Customer id

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ROOM_NO	NUMBER(38,0)	No	(null)	1	(null)
2	ROOM_TYPE	VARCHAR2(20 BYTE)	Yes	(null)	2	(null)
3	BED_TYPE	VARCHAR2(20 BYTE)	Yes	(null)	3	(null)
4	NO_OF_OCCUPANTS	NUMBER	Yes	(null)	4	(null)
5	ROOM_PRICE	NUMBER	Yes	(null)	5	(null)
6	CUSTOMER_ID	NUMBER	Yes	(null)	6	(null)
7	RES_ID	NUMBER	Yes	(null)	7	(null)

Address: This table has the details about the addresses like Street, City, State, Country, Customer id, Zip code

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	STREET	VARCHAR2(30 BYTE)	Yes	(null)	1	(null)
2	CITY	VARCHAR2(30 BYTE)	Yes	(null)	2	(null)
3	STATE	VARCHAR2(30 BYTE)	Yes	(null)	3	(null)
4	COUNTRY	VARCHAR2(20 BYTE)	Yes	(null)	4	(null)
5	CUSTOMER_ID	NUMBER	Yes	(null)	5	(null)
6	ZIP_CODE	NUMBER	Yes	(null)	6	(null)

Services: This table has information about the services available like Service id (primary key), Service type and Service cost

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	SERVICE_ID	NUMBER(38,0)	No	(null)	1	(null)
2	SERVICE_TYPE	VARCHAR2(20 BYTE)	Yes	(null)	2	(null)
3	SERVICE_COST	NUMBER	Yes	(null)	3	(null)

Invoice: This table has information about the invoices like Invoice No(primary key), Res id and Customer id

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	INVOICE_NO	NUMBER(38,0)	No	(null)	1	(null)
2	RES_ID	NUMBER(38,0)	Yes	(null)	2	(null)
3	CUSTOMER_ID	NUMBER	Yes	(null)	3	(null)

Line: This table has information about Line like Invoice No, Service id, Service quantity and Res id

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	INVOICE_NO	NUMBER(38,0)	Yes	(null)	1	(null)
2	SERVICE_ID	NUMBER(38,0)	Yes	(null)	2	(null)
3	SERVICE_QUANTITYY	NUMBER	Yes	(null)	3	(null)
4	RES_ID	NUMBER	Yes	(null)	4	(null)

Transaction: This table has information about the transaction like Transaction No (primary key), Payment Method, Payment Date, Invoice no and Customer id

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	TRANS_NO	NUMBER	No	(null)	1	(null)
2	PAYMENT_METHOD	VARCHAR2(20 BYTE)	Yes	(null)	2	(null)
3	PAYMENT_DATE	VARCHAR2(20 BYTE)	Yes	(null)	3	(null)
4	INVOICE_NO	NUMBER(38,0)	Yes	(null)	4	(null)
5	CUSTOMER_ID	NUMBER	Yes	(null)	5	(null)

Satisfaction: This table has information about the transaction like Satisfaction ID, Satisfaction Level and Transaction No

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Actions...

	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	SATISFACTION_ID	NUMBER(38,0)	Yes	(null)	1	(null)
2	SATISFACTION_LEVEL	NUMBER	Yes	(null)	2	(null)
3	TRANS_NO	NUMBER	Yes	(null)	3	(null)

Data Generation and Data Integrity

To maintain consistency of the data through the life cycle of data, integrity constraints are enforced. The constraints can either be at a column level or a table level. Some of the most common constraints are:

1. NOT NULL – Prevents a column from having a NULL value.
2. PRIMARY KEY – Uniquely identifies each row or record in table.
3. FOREIGN KEY – Uniquely identifies a column that references a PRIMARY KEY in another table.
4. UNIQUE – Prevents a column from having duplicate values.
5. CHECK – Checks for values that satisfy a specific condition as defined by the user.

Below are some of the constraints that we have enforced in our data base design

1. Table containing Customers

```
CREATE TABLE customer (
   customer_id        number NOT NULL primary key,
   first_name          VARCHAR2(20),
   last_name           VARCHAR2(20),
   Gender              VARCHAR2(20),
   phone_number      NUMBER
);
```

2. Table containing Bookings

```
CREATE TABLE Booking (
   Booking_id        INTEGER NOT NULL primary key,
   Book_type          VARCHAR2(20),
   Book_date          VARCHAR2(20),
   Customer_id              number  ,
   FOREIGN key(customer_id) references customer
);
```

3. Table containing Reservations

```
CREATE TABLE Reservation (
   Res_id         number NOT NULL primary key,
   check_in_date          varchar(20),
   check_out_date           varchar(20),
    No_of_days             number,
   Customer_id              number  ,
```

```
   Booking_id                INTEGER,
   FOREIGN key(customer_id) references customer,
   FOREIGN key(booking_id) references booking
);
```

4. Table containing Rooms

```
CREATE TABLE Room (
   Room_no        INTEGER NOT NULL ,
   Room_type           varchar(20),
   Bed_type           varchar(20),
   No_of_occupants       number,
   Room_price          number,
   Customer_id               number  ,
   res_id                number,

   FOREIGN key(customer_id) references Customer,
   FOREIGN key(res_id) references Reservation
);
```

5. Table containing Addresses

```
CREATE TABLE Address (
   Street           varchar(30),
   City            varchar(30),
   State       varchar(30),
   Country           varchar(20),
   Customer_id               number  ,
   Zip_code                number,
   FOREIGN key(customer_id) references Customer
);
```

6. Table containing Services

```
CREATE TABLE Services (
   Service_id        INTEGER NOT NULL primary key,
   Service_type           varchar(20),
  Service_cost       number
);
```

7. Table Creating Invoices

```
   create TABLE Invoice (
      Invoice_No        INTEGER NOT NULL primary key,
      Res_id          integer,
      customer_id     number,
```

```
    FOREIGN key(res_id) references reservation,

    FOREIGN key(customer_id) references Customer
  );
```

8. Table Creating Line Details

```
  CREATE TABLE Line (
    Invoice_No       INTEGER ,
    Service_id          integer,
   Service_quantityy    number,
   res_id            number,
   FOREIGN key(Service_id ) references services ,
   FOREIGN key(Invoice_No) references invoice ,
   FOREIGN key(res_id ) references reservation
   );
```

9. Table Creating Transaction Details

```
  CREATE TABLE transactions (
    Trans_No        number NOT NULL primary key,
    Payment_Method          varchar(20),
    Payment_Date          varchar(20),
    Invoice_no          integer,
    Customer_id       number,

    FOREIGN key(invoice_no) references Invoice,
    FOREIGN key(customer_id) references Customer
  );
```

10. Table Creating Transaction Details

```
CREATE TABLE Satisfaction (
  Satisfaction_ID       INTEGER,
  Satisfaction_level          number,
  Trans_no      number,

  FOREIGN key(Trans_no) references Transactions

);
```

SQL Queries

1. Total rows in customers and bookings tables (ensuring at least two tables have atleast 10,000 rows)

```
--Count total rows in customers and bookings
select count(*) from customer;

select count(*) from booking;
```

Script Output | Query Result | SQL | All Rows Fetched: 1 in 0.094 seconds

	COUNT(*)
1	10636

```
--Count total rows in customers and bookings
select count(*) from customer;
```

Script Output | Query Result | SQL | All Rows Fetched: 1 in 0.052 seconds

	COUNT(*)
1	57789

2. All the Customers with Same First and Last Name

```
--Select all the customers with same first and last name
select * from customer
where first_name = last_name;
```

Script Output | Query Result | SQL | All Rows Fetched: 2 in 0.054 seconds

	CUSTOMER_ID	FIRST_NAME	LAST_NAME	GENDER	PHONE_NUMBER
1	14502	Gordon	Gordon	F	2237735
2	17705	Kay	Kay	M	2240938

3. Total bookings by gender of the customer

```
--Total bookings by gender of the customer

select gender,count(distinct booking_id) as bookings
from customer c join booking b
on c.customer_id = b.customer_id
group by gender
```

Script Output | Query Result

SQL | All Rows Fetched: 2 in 0.071 seconds

	GENDER	BOOKINGS
1	M	5348
2	F	5288

4. Get the total number of customers per month.

Worksheet | Query Builder

```
select substr(check_in_date,1,2) as Month,count(customer_id) as No_of_Customers
from reservation group by substr(check_in_date,1,2)
order by substr(check_in_date,1,2) asc;
```

Query Result

SQL | All Rows Fetched: 9 in 0.037 seconds

	MONTH	NO_OF_CUSTOMERS
1	01	10
2	02	5
3	03	3
4	04	5
5	05	2
6	06	3
7	07	7
8	08	6
9	09	8

5. Retrieve the Total Hotel sales per month.

```
-- Retrieve the Total Hotel sales per month.
select  substr(t.payment_date,1,2) as Month,sum(r.room_price*rs.no_of_days + s.service_cost*l.service_quantityy)Total_Sales,
sum (r.room_price*rs.no_of_days)Total_Room_chargess,
sum(s.service_cost*l.service_quantityy)Total_Services_Charges
from room r, reservation rs,services s, line l,invoice i,transactions t
where r.res_id=rs.res_id and s.service_id=l.service_id and l.invoice_no=i.invoice_no
and rs.res_id=i.res_id  and t.invoice_no=i.invoice_no
group by substr(t.payment_date,1,2)
order by substr(t.payment_date,1,2)asc;
```

Script Output | Query Result

SQL | All Rows Fetched: 9 in 0.035 seconds

	MONTH	TOTAL_SALES	TOTAL_ROOM_CHARGESS	TOTAL_SERVICES_CHARGES
1	01	2070	1610	460
2	02	810	640	170
3	03	710	570	140
4	04	910	710	200
5	05	400	300	100
6	06	530	420	110
7	07	1730	1400	330
8	08	1460	1220	240
9	09	1870	1560	310

6. Fetch average satisfaction rating per month

```
--Retrieve Average Satisfaction rating per month.

select substr(t.payment_date,1,2)as Month,round(avg(sf.satisfaction_level),2)
from transactions t, satisfaction sf
where sf.trans_no=t.trans_no
group by substr(t.payment_date,1,2)
order by substr(t.payment_date,1,2) asc;
```

Script Output | Query Result

SQL | All Rows Fetched: 9 in 0.047 seconds

	MONTH	ROUND(AVG(SF.SATISFACTION_LEVEL),2)
1	01	2.9
2	02	2.2
3	03	3.33
4	04	3.6
5	05	3.5
6	06	4.33
7	07	4.71
8	08	4.5
9	09	4.63

Performance Tuning

INDEX

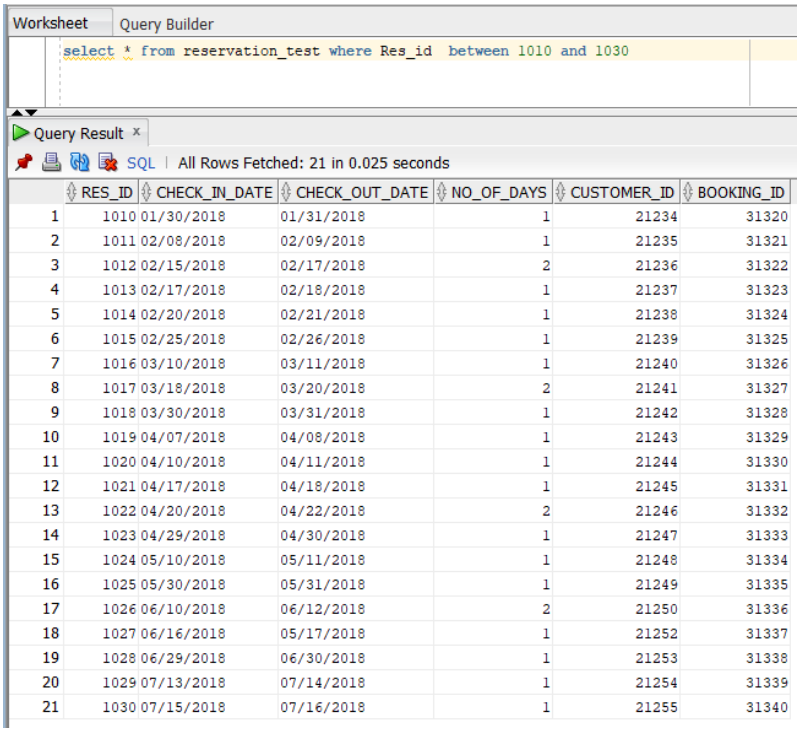
An index is used to increase the overall performance of queries. Indexing does this by reducing the data pages that has to be visited or scanned every time a query is run.
When we create index, by default the primary key creates a clustered index. In SQL Server, a clustered index determines the physical order of data in a table. There can be only one clustered index per table.

Query:

In below example we can run a range query on two same tables, of which one table having primary key(Reservation) and other test table(Reservation_test) without primary key.
Reservation_test Table:-
Sql:-

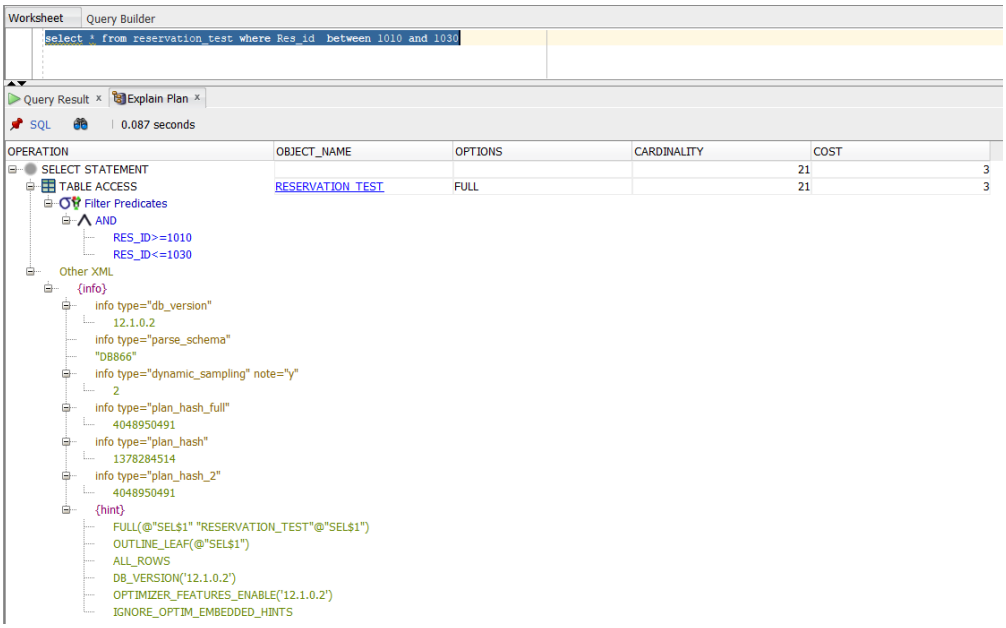
```
CREATE TABLE Reservation_test (
        Res_id              number,
        check_in_date       varchar(20),
        check_out_date      varchar(20),
        No_of_days          number,
        Customer_id         number  ,
        Booking_id          INTEGER
    );
```

Query Result:-



	RES_ID	CHECK_IN_DATE	CHECK_OUT_DATE	NO_OF_DAYS	CUSTOMER_ID	BOOKING_ID
1	1010	01/30/2018	01/31/2018	1	21234	31320
2	1011	02/08/2018	02/09/2018	1	21235	31321
3	1012	02/15/2018	02/17/2018	2	21236	31322
4	1013	02/17/2018	02/18/2018	1	21237	31323
5	1014	02/20/2018	02/21/2018	1	21238	31324
6	1015	02/25/2018	02/26/2018	1	21239	31325
7	1016	03/10/2018	03/11/2018	1	21240	31326
8	1017	03/18/2018	03/20/2018	2	21241	31327
9	1018	03/30/2018	03/31/2018	1	21242	31328
10	1019	04/07/2018	04/08/2018	1	21243	31329
11	1020	04/10/2018	04/11/2018	1	21244	31330
12	1021	04/17/2018	04/18/2018	1	21245	31331
13	1022	04/20/2018	04/22/2018	2	21246	31332
14	1023	04/29/2018	04/30/2018	1	21247	31333
15	1024	05/10/2018	05/11/2018	1	21248	31334
16	1025	05/30/2018	05/31/2018	1	21249	31335
17	1026	06/10/2018	06/12/2018	2	21250	31336
18	1027	06/16/2018	05/17/2018	1	21252	31337
19	1028	06/29/2018	06/30/2018	1	21253	31338
20	1029	07/13/2018	07/14/2018	1	21254	31339
21	1030	07/15/2018	07/16/2018	1	21255	31340

Execution Plan:



Reservation Table:-

```
CREATE TABLE Reservation (
        Res_id          number NOT NULL primary key,
        check_in_date           varchar(20),
        check_out_date          varchar(20),
         No_of_days             number,
        Customer_id             number  ,
        Booking_id              INTEGER,
        FOREIGN key(customer_id) references customer,
        FOREIGN key(booking_id) references booking
    );
```

Query Result:-

```
select * from reservation where Res_id  between 1010 and 1030
```

Query Result | SQL | All Rows Fetched: 21 in 0.028 seconds

	RES_ID	CHECK_IN_DATE	CHECK_OUT_DATE	NO_OF_DAYS	CUSTOMER_ID	BOOKING_ID
1	1010	01/30/2018	01/31/2018	1	21234	31320
2	1011	02/08/2018	02/09/2018	1	21235	31321
3	1012	02/15/2018	02/17/2018	2	21236	31322
4	1013	02/17/2018	02/18/2018	1	21237	31323
5	1014	02/20/2018	02/21/2018	1	21238	31324
6	1015	02/25/2018	02/26/2018	1	21239	31325
7	1016	03/10/2018	03/11/2018	1	21240	31326
8	1017	03/18/2018	03/20/2018	2	21241	31327
9	1018	03/30/2018	03/31/2018	1	21242	31328
10	1019	04/07/2018	04/08/2018	1	21243	31329
11	1020	04/10/2018	04/11/2018	1	21244	31330
12	1021	04/17/2018	04/18/2018	1	21245	31331
13	1022	04/20/2018	04/22/2018	2	21246	31332
14	1023	04/29/2018	04/30/2018	1	21247	31333
15	1024	05/10/2018	05/11/2018	1	21248	31334
16	1025	05/30/2018	05/31/2018	1	21249	31335
17	1026	06/10/2018	06/12/2018	2	21250	31336
18	1027	06/16/2018	05/17/2018	1	21252	31337
19	1028	06/29/2018	06/30/2018	1	21253	31338
20	1029	07/13/2018	07/14/2018	1	21254	31339
21	1030	07/15/2018	07/16/2018	1	21255	31340

Execution Plan:

Worksheet | Query Builder

```
select * from reservation where Res_id  between 1010 and 1030
```

Query Result | Explain Plan

SQL | 0.083 seconds

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			21	0
TABLE ACCESS	RESERVATION	BY INDEX ROWID BATCHED	21	0
INDEX	SYS_C00134310	RANGE SCAN	1	0

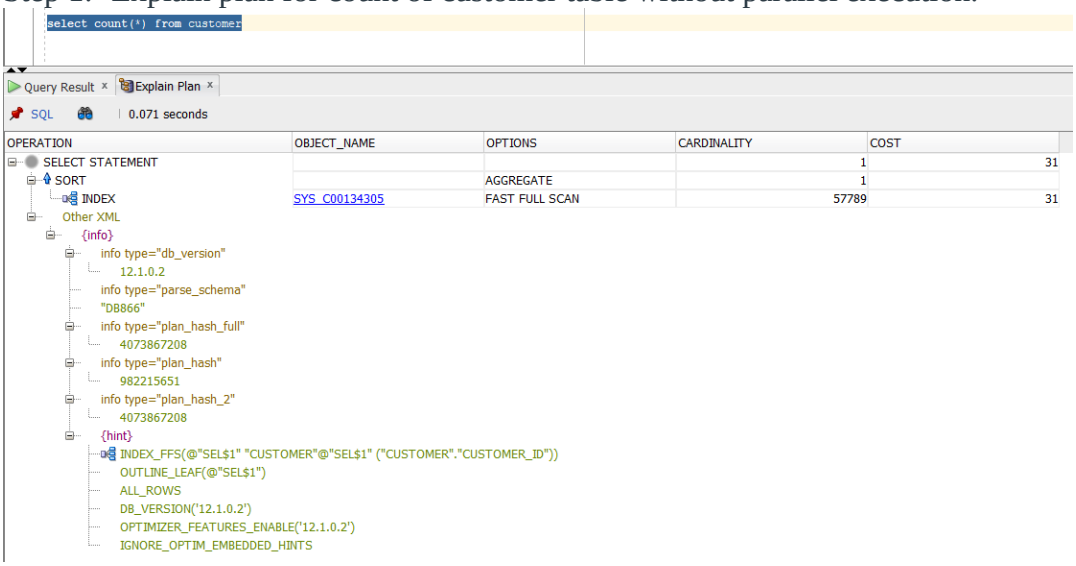
```
Access Predicates
  AND
    RES_ID>=1010
    RES_ID<=1030
Other XML
  {info}
    info type="db_version"
      12.1.0.2
    info type="parse_schema"
    "DB866"
    info type="dynamic_sampling" note="y"
      2
    info type="plan_hash_full"
      1663570932
    info type="plan_hash"
      2551452666
    info type="plan_hash_2"
      1663570932
    {u}
      5604814135823257983
      0
      1
    {hint}
      BATCH_TABLE_ACCESS_BY_ROWID(@"SEL$1" "RESERVATION"@"SEL$1")
      INDEX_RS_ASC(@"SEL$1" "RESERVATION"@"SEL$1" ("RESERVATION"."RES_ID"))
      OUTLINE_LEAF(@"SEL$1")
      ALL_ROWS
      DB_VERSION('12.1.0.2')
      OPTIMIZER_FEATURES_ENABLE('12.1.0.2')
      IGNORE_OPTIM_EMBEDDED_HINTS
```

From the above screenshot we can see the better performance of query when table is created with primary key as it creates clustered index as default and indexing reduces the data pages that has to be visited or scanned every time a query is run

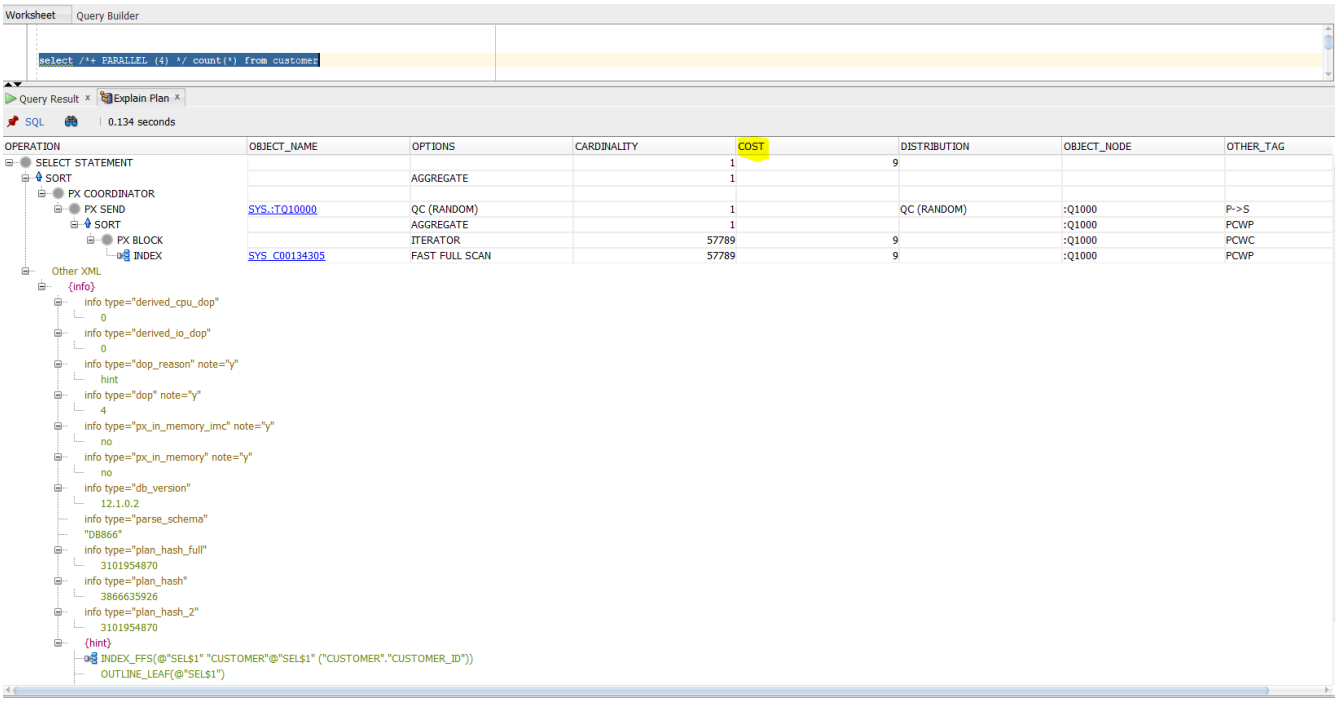
Parallelism:

Purpose of the experiment:- The below experiment is conducted to show the cost difference between using parallel execution and without parallel execution.

Step 1:- Explain plan for count of customer table without parallel execution.

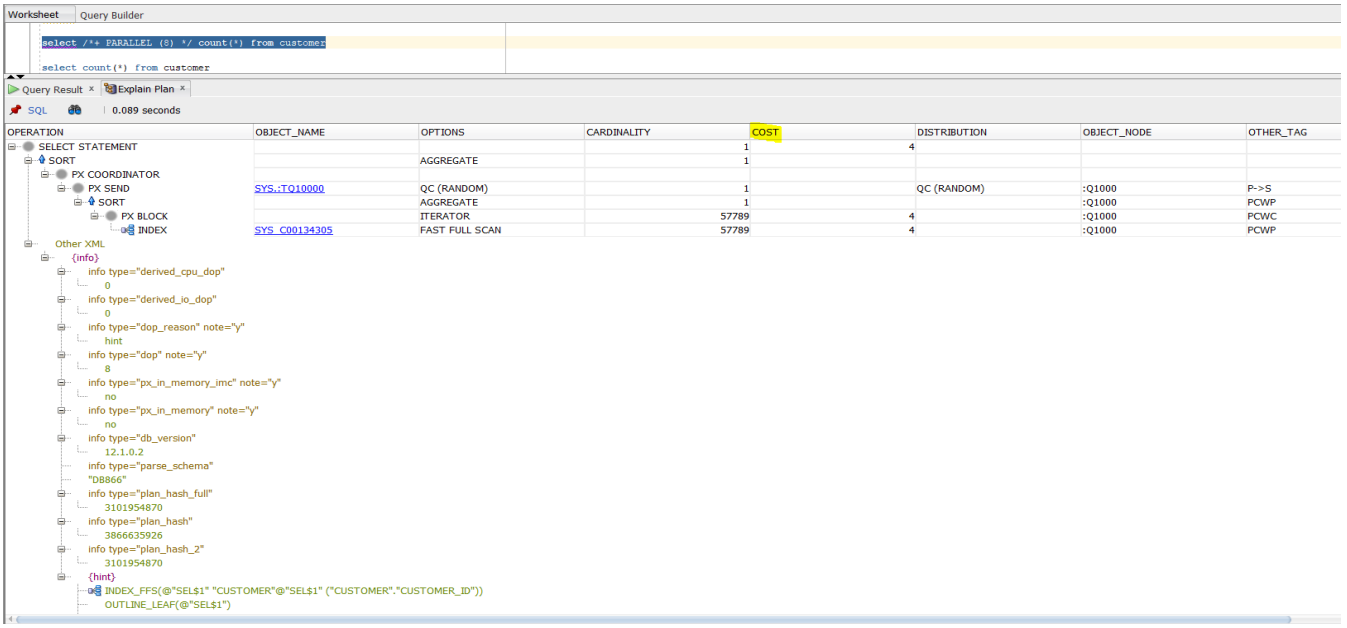


Step 2:- Now we are running same query triggering parallelism,4 different machines in this case. Let's see the cost.



Step 3:-

Again running the same query triggering parallelism, 8 different machines in this case. Explain plan result



- Results : - From the above results we can observe above that the cost has decreased everytime we increase the processes. As we know that parallelism is breaking down a task so that, instead of one process doing all of the work in a query, many processes do part of the work at the same time.
- This can be observed in our results i.e cost reduction. This is very useful when we are dealing with online transaction processing (OLTP) and also dealing very huge scale data.

Parallel execution divides the task of executing a SQL statement into multiple small units, each of which is executed by a separate process. Also the incoming data (tables, indexes, partitions) can be divided into parts

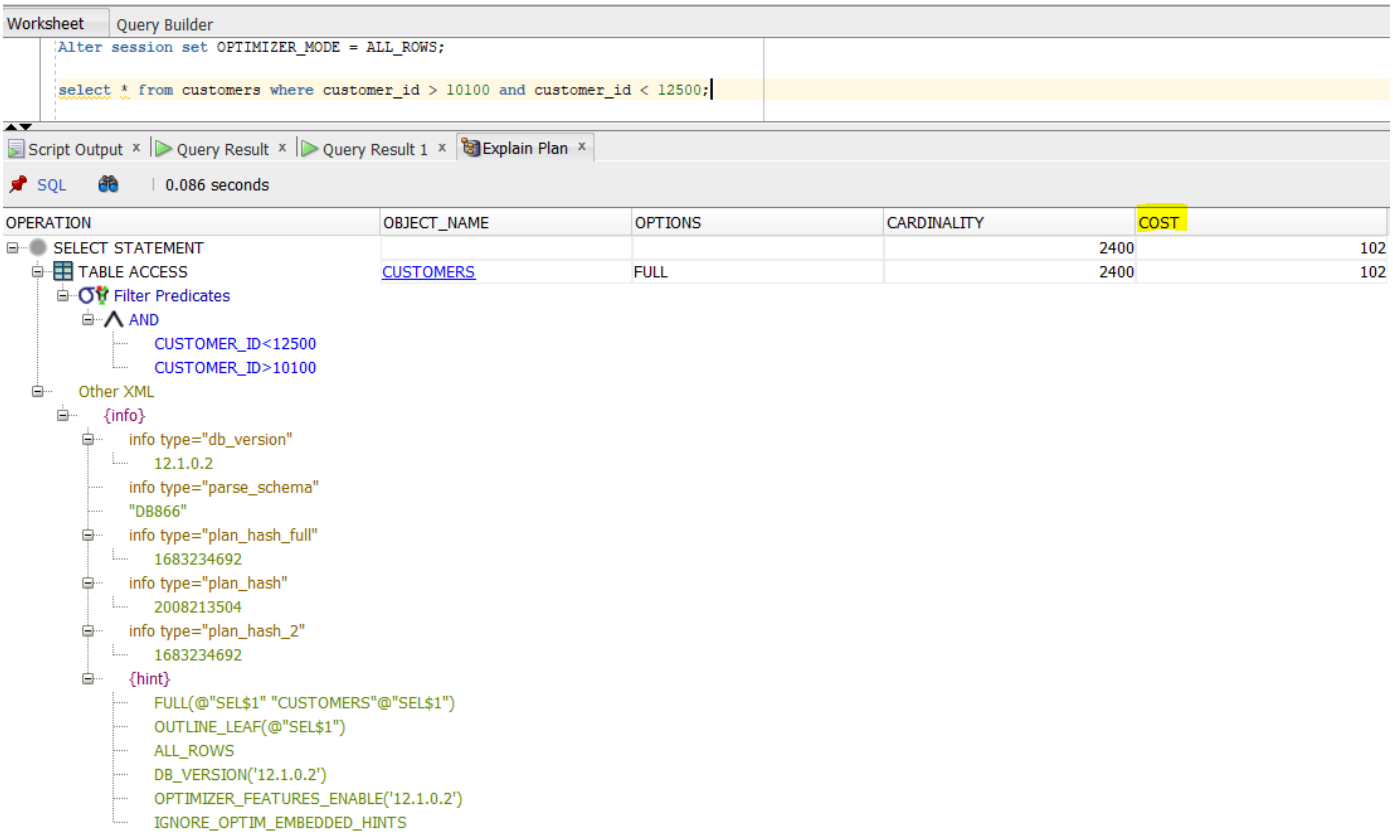
OPTIMIZER MODE(Using the Cost-Based approach) :-

Optimizer mode is used to choose better execution plans for poorly written queries.

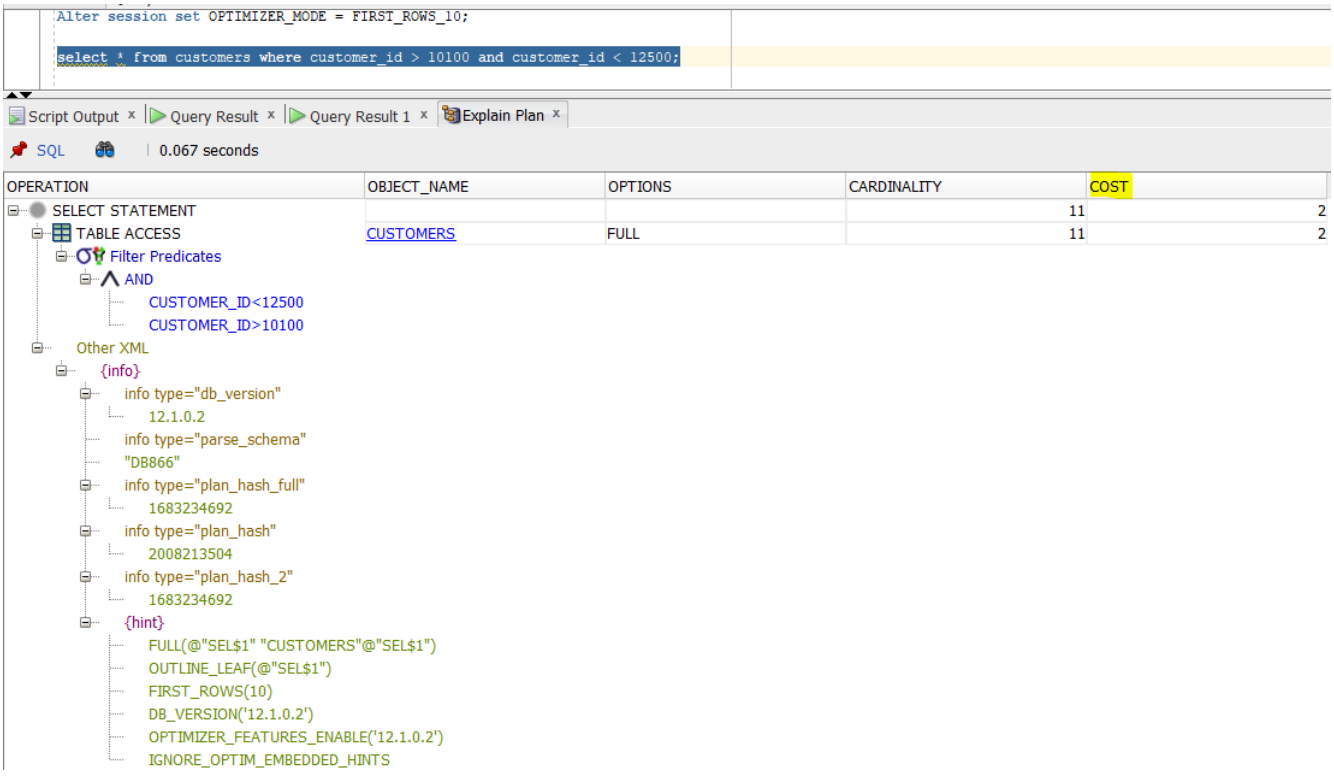
There are different types of optimizer modes in oracle such as ALL_ROWS, FIRST_ROWS_N, RULE, CHOOSE.

Here we check cost of simple range query with two optimizer modes. They are default ALL_ROWS and FIRST_ROWS_N.

Step1:- Optimizer used a cost-based approach for all SQL statements in the session and try best execution plan. Executing the query when the optimizer mode is ALL_ROWS.



Step 2:- Optimizer uses a cost-based approach according to response time of the return the first n rows. Now altering the session to FIRST_ROWS_10 and examining the explain plan.



Results: As our data is not huge such as millions, we cannot see much of time difference in execution of queries. However, when we compare the costs of two optimizer modes there is significant difference.

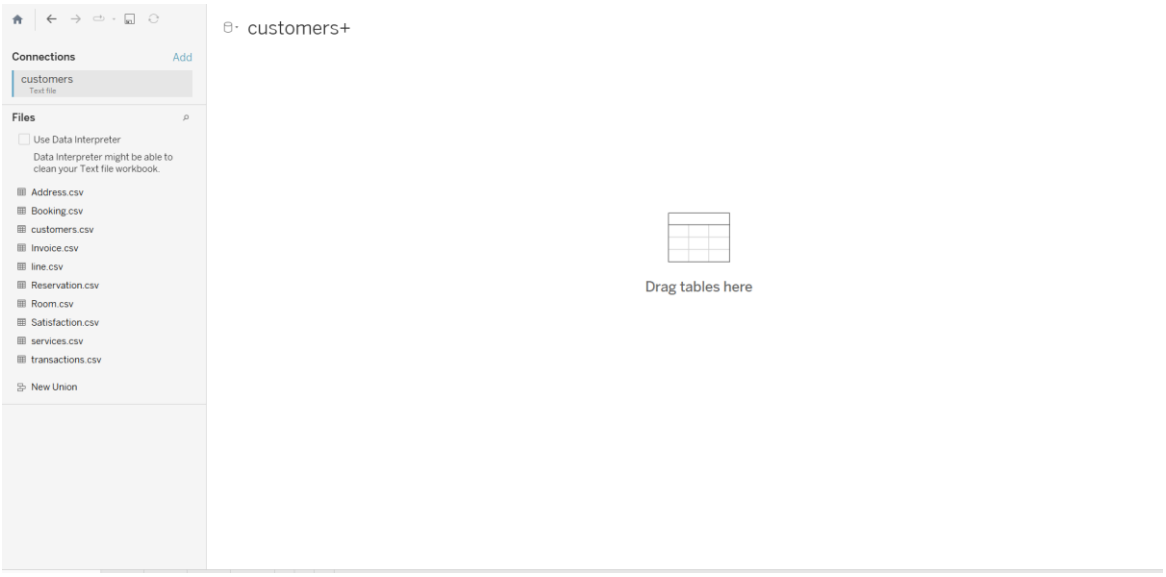
The cost when optimizer mode is ALL_ROWS is 102, whereas when the optimizer mode is FIRST_ROWS_10 it is 2.

In the real-world scenario, the choosing of optimizer depends upon the optimizer's goal. Optimizing for best throughput is more likely to result in a full table scan and sort-merge join. Optimizing for best response time, however, more likely results in an index scan or a nested loops join. ALL_ROWS is usually more important in batch applications because response time is of less importance here compared to time taken by batch application to complete. FIRST_ROWS_N are usually important in interactive applications because use is waiting to see the first rows accessed by the statement.

Visualization in Tableau

We have used **Tableau Desktop** for creating visualization and dashboards.

Initially, uploaded csv files of all the tables. The Screenshots of the same are shown below:

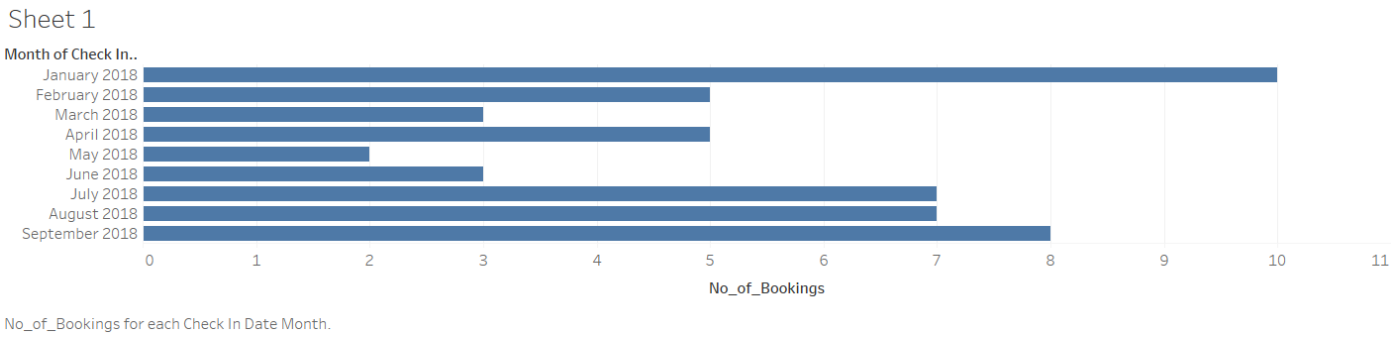


Dashboard 1:

Dashboard 1 includes visualizations which show

- Get the total number of customers per month.

Visualization:



No_of_Bookings for each Check In Date Month.

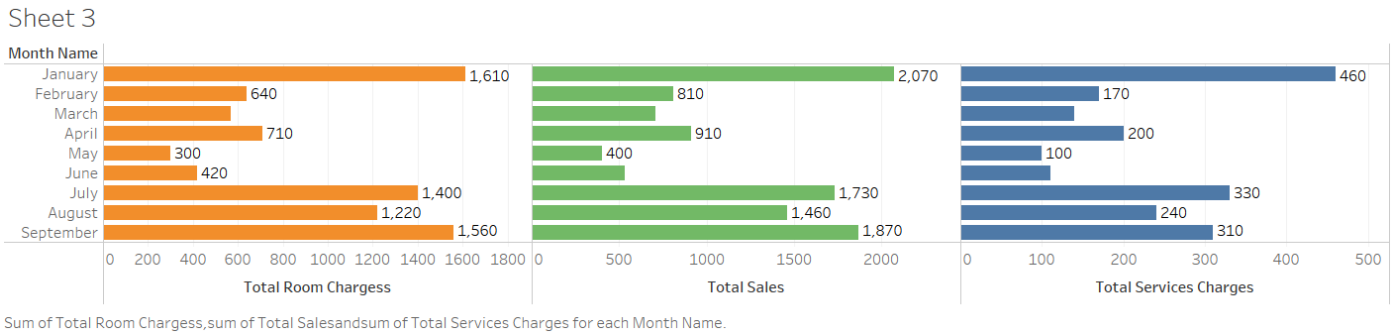
- **January 2018** was the month with highest booking followed by **September 2018**.
- **May 2018** is the month which has seen the lowest booking.

Dashboard 2:

Dashboard 2 includes visualizations which show:

- Total Room Charges Vs Month
- Total Sales Vs Month
- Total Service Charges Vs Month

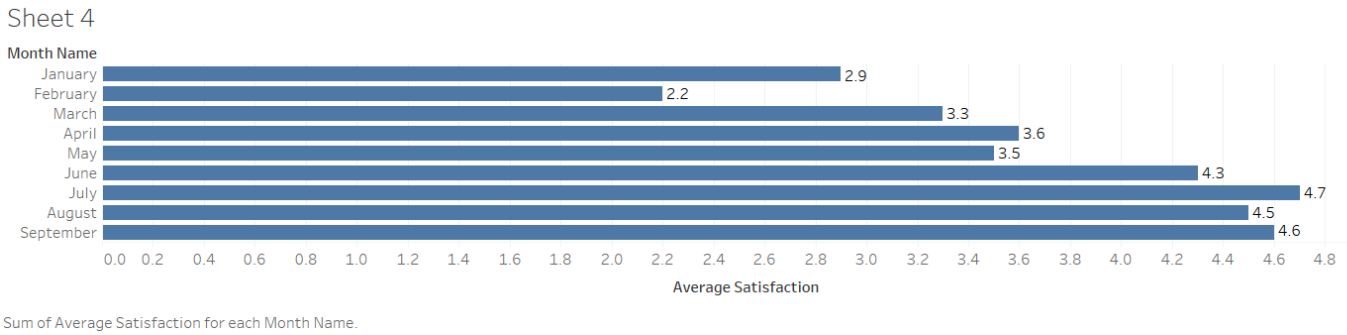
Visualization:



- We can observe that **Room charges** were highest in **January** and **September** when the **no of customers** were also high. This shows that price variation with the demand. Similarly **May** month has the lowest Room Charges.
- In Total Sales Vs Month and Total Service Charges Vs Month we can observe the highs and lows are similar with No of customers vs Month because the total sales and also services offered are directly related to foot falls observed.

Dashboard 3:

Dashboard 3 includes visualizations which show:



- **July** month has the highest customer satisfaction rating of 4.7. **September** month was having second highest customer satisfaction rating. September month with second highest footfalls the hotel managed to achieve the second highest customer satisfaction.
- **February** month has the lowest lowest customer satisfaction.
- **Hotel** was operating to good customer satisfaction at end of the year of month in data i.e **June**, **July**, **August**, **September** month has crossed 4.2 ratings.

Future Scope

The database can be utilized further to do the following analysis:

- **Sales Predictor:**

  We can use Multiple linear regression on the **Room**, **Reservation**. **services**, **Line** details data and predict sales of each month or year with data collected in future.

- **Foot Falls Predictor:**

  With more historical data of Reservation using linear regression, in future we can predict the expected no of customers to book hotel so that the hotel can be equipped ahead of demand.

References:

1. Data Source:

2. Tableau Tutorial: https://help.tableau.com/current/guides/get-started-tutorial/en-us/get-started-tutorial-home.htm