Dunn Hotel
Database Design Proposal

Taylor Dunn
Table of Contents

Executive Summary.................................3
ER Diagram..........................................4
Table Statements.................................6-23
View Statements.................................25-30
Stored Procedures...............................32-33
Reports ..............................................34-35
Triggers..............................................36-37
Security..............................................38
Problems/Enhancements......................39
This database has been created for the Dunn Hotel, a hotel that is run by Taylor Dunn and her minions. It has been created to keep track of all records needed to ensure the success of the hotel.

The information that follows is an intensive review of the database itself, and aspects of it’s uses. There are numerous parts of this review including the ER Diagram, create statements for tables, and the sample data that was inserted into the table. Next are the results of queries, views, stored procedures, reports, and triggers. These were all created and then tested.

The purpose of this database is to condense all of the information that the hotel needs to function into one central, and organized collection of tables.
Tables
Attractions: This table contains different attractions in the area of Liverpool, Texas, including the hotel that this database focuses on.

```
CREATE TABLE attractions (  
  attraction_id               char(3)     not null,    
  attraction_name             text        not null,    
  attraction_description      text        not null,    
  primary key (attraction_id)
);
```

<table>
<thead>
<tr>
<th>attraction_id</th>
<th>attraction_name</th>
<th>attraction_description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1</td>
<td>Hotel</td>
<td>Places to stay in the ar...</td>
</tr>
<tr>
<td>a2</td>
<td>FDR Museum</td>
<td>Places to check out his...</td>
</tr>
<tr>
<td>a3</td>
<td>Riverwalk</td>
<td>Places to explore the a...</td>
</tr>
<tr>
<td>a4</td>
<td>Restaurants</td>
<td>Places to eat in the area</td>
</tr>
</tbody>
</table>

Functional Dependencies: attraction_id → attraction_name, attraction_description
Transactions: This table holds all of the information regarding the transactions that go through this hotel day in and day out.

```
CREATE TABLE transactions {
    transaction_id char(8) not null,
    reservation_id char(8) not null,
    employee_id char(3) not null,
    payment_id char(3) not null,
    customer_id char(3) not null,
    hotel_id char(3) not null,
    primary key (transaction_id),
    foreign key (hotel_id) references hotels (hotel_id),
    foreign key (reservation_id) references reservations (reservation_id),
    foreign key (employee_id) references employees (employee_id),
    foreign key (payment_id) references payment (payment_id)
};
```

<table>
<thead>
<tr>
<th>transaction_id</th>
<th>reservation_id</th>
<th>employee_id</th>
<th>payment_id</th>
<th>customer_id</th>
<th>hotel_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>11111</td>
<td>r1</td>
<td>e1</td>
<td>b1</td>
<td>c1</td>
<td>h1</td>
</tr>
<tr>
<td>12222</td>
<td>r2</td>
<td>e1</td>
<td>b2</td>
<td>c2</td>
<td>h1</td>
</tr>
<tr>
<td>13333</td>
<td>r4</td>
<td>e3</td>
<td>b3</td>
<td>c3</td>
<td>h1</td>
</tr>
<tr>
<td>14444</td>
<td>r3</td>
<td>e1</td>
<td>b4</td>
<td>c1</td>
<td>h1</td>
</tr>
<tr>
<td>155555</td>
<td>r5</td>
<td>e3</td>
<td>b5</td>
<td>c5</td>
<td>h1</td>
</tr>
<tr>
<td>166666</td>
<td>r6</td>
<td>e3</td>
<td>b6</td>
<td>c1</td>
<td>h1</td>
</tr>
<tr>
<td>177777</td>
<td>r7</td>
<td>e1</td>
<td>b7</td>
<td>c3</td>
<td>h1</td>
</tr>
<tr>
<td>188888</td>
<td>r8</td>
<td>e3</td>
<td>b8</td>
<td>c3</td>
<td>h1</td>
</tr>
<tr>
<td>199999</td>
<td>r9</td>
<td>e1</td>
<td>b9</td>
<td>c4</td>
<td>h1</td>
</tr>
<tr>
<td>112222</td>
<td>r10</td>
<td>e1</td>
<td>b10</td>
<td>c4</td>
<td>h1</td>
</tr>
</tbody>
</table>

Functional Dependencies:  transaction_id → employee_id, payment_id, reservation_id, customer_id, hotel_id
Hotels: This table contains the specific information about one of the hotel attractions in the area.

```sql
CREATE TABLE hotels (  
    hotel_id      char(7)    not null,  
    street_address text       not null,  
    city          text        not null,  
    state         text        not null,  
    zipcode       integer     not null,  
    hotel_name    text        not null,  
    owner_firstname text       not null,  
    owner_lastname text       not null,  
    attraction_id char(3)     not null,
    primary key (hotel_id),  
    foreign key (attraction_id) references attractions (attraction_id)
);```

<table>
<thead>
<tr>
<th>hotel_id</th>
<th>street_address</th>
<th>city</th>
<th>state</th>
<th>zipcode</th>
<th>hotel_name</th>
<th>owner_firstname</th>
<th>owner_lastname</th>
<th>attraction_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>h1</td>
<td>123 Galway Lane</td>
<td>Live...</td>
<td>Texas</td>
<td>12894</td>
<td>The Dunn H...</td>
<td>Taylor</td>
<td>Dunn</td>
<td>a1</td>
</tr>
<tr>
<td>h2</td>
<td>134 Hello Street</td>
<td>Ha...</td>
<td>New...</td>
<td>12345</td>
<td>The Hilly Hall</td>
<td>John</td>
<td>Cena</td>
<td>a1</td>
</tr>
</tbody>
</table>

Functional Dependencies: hotel_id → street_address, city, state, zipcode, hotel_name, owner_firstname, owner_lastname, attraction_id
Miscellaneous_Charges: This table contains the miscellaneous charges options that can be added to a transaction. These charges are broken down into three options, but more could be added as necessary.

```
CREATE TABLE miscellaneous_charges (  
miscellaneous_charges_id char(3) not null,  
name text not null,  
chargesUSD decimal(15,2) not null,  
description text not null,  
primary key (miscellaneous_charges_id)  
);
```

<table>
<thead>
<tr>
<th>miscellaneous_charges_id</th>
<th>name</th>
<th>chargesUSD</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>Stolen Item</td>
<td>50.00</td>
<td>Something from the room is missing.</td>
</tr>
<tr>
<td>m2</td>
<td>Broken Furniture</td>
<td>200.00</td>
<td>Something from the room is broken and needs to be replaced.</td>
</tr>
<tr>
<td>m3</td>
<td>Food</td>
<td>25.00</td>
<td>All room service for food is under a $25 buffet, all you can eat style.</td>
</tr>
</tbody>
</table>

Functional Dependencies: miscellaneous_charges_id → name, chargesUSD, description
Miscellaneous_Charges_Add: This table displays which transactions have miscellaneous charges in their orders.

```
CREATE TABLE miscellaneous_charges_add (
    miscellaneous_charges_id char(3) not null,
    transaction_id char(8) not null,
    foreign key (miscellaneous_charges_id) references miscellaneous_charges (miscellaneous_charges_id),
    foreign key (transaction_id) references transactions (transaction_id)
);
```

<table>
<thead>
<tr>
<th>miscellaneous_charges_id</th>
<th>transaction_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>11111</td>
</tr>
<tr>
<td>m2</td>
<td>155555</td>
</tr>
<tr>
<td>m3</td>
<td>199999</td>
</tr>
</tbody>
</table>

Functional Dependencies: none
Payment: This table shows how a customer paid for their reservation, and is linked to the transaction table through the payment_id column.

```
CREATE TABLE payment (  
payment_id char(3) not null,  
payment_type_id char(2) not null,  
primary key (payment_id),  
foreign key (payment_type_id) references payment_type (payment_type_id) );
```

<table>
<thead>
<tr>
<th>payment_id</th>
<th>payment_type_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>b1</td>
<td>t1</td>
</tr>
<tr>
<td>b2</td>
<td>t2</td>
</tr>
<tr>
<td>b3</td>
<td>t1</td>
</tr>
<tr>
<td>b4</td>
<td>t3</td>
</tr>
<tr>
<td>b5</td>
<td>t2</td>
</tr>
<tr>
<td>b6</td>
<td>t3</td>
</tr>
<tr>
<td>b7</td>
<td>t1</td>
</tr>
<tr>
<td>b8</td>
<td>t3</td>
</tr>
<tr>
<td>b9</td>
<td>t1</td>
</tr>
<tr>
<td>b10</td>
<td>t2</td>
</tr>
</tbody>
</table>

Functional Dependencies: payment_id → payment_type_id
Payment_Type: This table holds the different payment methods that this hotel accepts.

```
CREATE TABLE payment_type (
    payment_type_id char(2) not null,
    payment_name text not null,

    primary key (payment_type_id)
);
```

<table>
<thead>
<tr>
<th>payment_type_id</th>
<th>payment_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>Cash</td>
</tr>
<tr>
<td>t2</td>
<td>Card</td>
</tr>
<tr>
<td>t3</td>
<td>Bitcoin</td>
</tr>
</tbody>
</table>

Functional Dependencies: payment_type_id → payment_name
People: This table holds all of the people that interact with the hotel and its database.

CREATE TABLE people (  
  PID char(5) not null,  
  first_name text not null,  
  last_name text not null,  
  street_address text not null,  
  state text not null,  
  zipcode integer not null,  
  birthday date not null,  
  contact_number text not null,  
  email_address text not null,  
  primary key (PID)  
);  

Functional Dependencies: pid → first_name, last_name, street_address, state, zipcode, birthday, contact_number, email_address  

People Sample Data on next slide
# People Sample Data:

<table>
<thead>
<tr>
<th>pid</th>
<th>first_name</th>
<th>last_name</th>
<th>street_address</th>
<th>state</th>
<th>zipcode</th>
<th>birthdate</th>
<th>contact_number</th>
<th>email_address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Jason</td>
<td>Haley</td>
<td>13 School Street</td>
<td>New York</td>
<td>11946</td>
<td>1978-11-01</td>
<td>4587390869</td>
<td><a href="mailto:jason.haley@gmail.com">jason.haley@gmail.com</a></td>
</tr>
<tr>
<td>2</td>
<td>Scott</td>
<td>Fritsch</td>
<td>10 Emerson Court</td>
<td>New York</td>
<td>11946</td>
<td>1989-06-01</td>
<td>1234567890</td>
<td><a href="mailto:scott.fritsch@gmail.com">scott.fritsch@gmail.com</a></td>
</tr>
<tr>
<td>3</td>
<td>Jami</td>
<td>Domenico</td>
<td>15 Maple Court</td>
<td>New York</td>
<td>18977</td>
<td>1997-04-01</td>
<td>6312546789</td>
<td><a href="mailto:jami.domenico@gmail.com">jami.domenico@gmail.com</a></td>
</tr>
<tr>
<td>4</td>
<td>Alan</td>
<td>Laboseur</td>
<td>255 Honey Drive</td>
<td>New York</td>
<td>12601</td>
<td>1985-09-01</td>
<td>1118675301</td>
<td><a href="mailto:alan.lab@cool.com">alan.lab@cool.com</a></td>
</tr>
<tr>
<td>5</td>
<td>Jack</td>
<td>Heuber</td>
<td>123 Talk Road</td>
<td>New York</td>
<td>12445</td>
<td>1998-10-01</td>
<td>1345879978</td>
<td><a href="mailto:talkingguytal@gmail.com">talkingguytal@gmail.com</a></td>
</tr>
<tr>
<td>6</td>
<td>Dave</td>
<td>Connelly</td>
<td>15 Bae Court</td>
<td>Rhode Island</td>
<td>12366</td>
<td>1997-11-01</td>
<td>1879087890</td>
<td>jeffery@<a href="mailto:jeffery@gmail.com">jeffery@gmail.com</a></td>
</tr>
<tr>
<td>7</td>
<td>Taylor</td>
<td>Connelly</td>
<td>17 Harbor Road</td>
<td>Riverside</td>
<td>14577</td>
<td>1997-11-01</td>
<td>6316805787</td>
<td><a href="mailto:taylor.kathryn@gmail.com">taylor.kathryn@gmail.com</a></td>
</tr>
<tr>
<td>8</td>
<td>John</td>
<td>Sasso</td>
<td>40 Bestfriend Lane</td>
<td>New York</td>
<td>12889</td>
<td>1997-12-01</td>
<td>2267897765</td>
<td><a href="mailto:john.sasso@bestfriend.com">john.sasso@bestfriend.com</a></td>
</tr>
<tr>
<td>9</td>
<td>Shannon</td>
<td>Cover</td>
<td>33 Oak Ave</td>
<td>New York</td>
<td>89059</td>
<td>1990-06-01</td>
<td>0987654321</td>
<td><a href="mailto:shannon.cover@gmail.com">shannon.cover@gmail.com</a></td>
</tr>
<tr>
<td>10</td>
<td>Sreya</td>
<td>Sobti</td>
<td>1334 Linda Lane</td>
<td>Pennsylvania</td>
<td>37890</td>
<td>1995-10-01</td>
<td>7778987654</td>
<td><a href="mailto:sreya.sobti@ind.com">sreya.sobti@ind.com</a></td>
</tr>
</tbody>
</table>
Employees and Customer: Both people, these tables connect to the people table and include extra information.

```
CREATE TABLE employees (  
    employee_id    char(3) not null,  
    PID            char(5) not null,  
    job_id         char(2) not null,  
    hire_date      date not null,  
    hourly_wageusd decimal(15,2) not null,  
    primary key (employee_id),  
    foreign key (PID) references people(pid),  
    foreign key (job_id) references job_type (job_id) 
);
```

```
CREATE TABLE customers (  
    PID            char(3) not null,  
    customer_id    char(3) not null,  
    primary key (customer_id),  
    foreign key (pid) references people (pid) 
);
```

<table>
<thead>
<tr>
<th>employee_id</th>
<th>pid</th>
<th>job_id</th>
<th>hire_date</th>
<th>hourly_wageusd</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>e1</td>
<td>90</td>
<td>2017-03-..</td>
<td>10.00</td>
</tr>
<tr>
<td>2</td>
<td>e2</td>
<td>91</td>
<td>2012-08-..</td>
<td>10.00</td>
</tr>
<tr>
<td>3</td>
<td>e3</td>
<td>91</td>
<td>2011-07-..</td>
<td>10.00</td>
</tr>
<tr>
<td>4</td>
<td>e4</td>
<td>92</td>
<td>2009-05-..</td>
<td>20.00</td>
</tr>
<tr>
<td>5</td>
<td>e5</td>
<td>93</td>
<td>2017-08-..</td>
<td>15.00</td>
</tr>
</tbody>
</table>

Functional Dependencies: employee_id → pid, job_id, hire_date, hourly_wageusd

pid → customer_id
Timesheet: This table includes all of the employees and their hours.

CREATE TABLE timesheet (  
timesheet_insert_id char(10) not null,  
employee_id char(3) not null,  
date date not null,  
time_in time not null,  
time_out time not null,  
total_hours integer not null,  
primary key (timesheet_insert_id),  
foreign key (employee_id) references employees (employee_id)  
);

<table>
<thead>
<tr>
<th>timesheet_insert_id</th>
<th>employee_id</th>
<th>date</th>
<th>time_in</th>
<th>time_out</th>
<th>total_hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>time1</td>
<td>e1</td>
<td>2017-01-01</td>
<td>11:00:00</td>
<td>17:00:00</td>
<td>6</td>
</tr>
<tr>
<td>time2</td>
<td>e1</td>
<td>2017-01-01</td>
<td>10:00:00</td>
<td>18:00:00</td>
<td>8</td>
</tr>
<tr>
<td>time3</td>
<td>e2</td>
<td>2017-01-01</td>
<td>11:00:00</td>
<td>18:00:00</td>
<td>7</td>
</tr>
<tr>
<td>time4</td>
<td>e3</td>
<td>2017-01-01</td>
<td>08:00:00</td>
<td>16:00:00</td>
<td>8</td>
</tr>
</tbody>
</table>

Functional Dependencies:  
employee_id → date, time_in, time_out

total_hours → time_in, time_out
Job_Type: This table holds information about different jobs that the employees hold.

```sql
CREATE TABLE job_type (  
  job_id   char(2) not null,  
  job_name text not null,  
  description text not null,  
  primary key (job_id)  
);
```

<table>
<thead>
<tr>
<th>job_id</th>
<th>job_name</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>90 Front Desk</td>
<td>Person aids...</td>
</tr>
<tr>
<td>2</td>
<td>91 Housekee...</td>
<td>Person clean...</td>
</tr>
<tr>
<td>3</td>
<td>92 Manager</td>
<td>Person look...</td>
</tr>
<tr>
<td>4</td>
<td>93 Bell Hop</td>
<td>Person take...</td>
</tr>
</tbody>
</table>

Functional Dependencies: job_id → job_name, job_description
Memberships: This table contains membership information for customers who are considered members.

```
CREATE TABLE memberships {
    membership_id char(8) not null,
    PID char(3) not null,
    customer_id char(3) not null,
    username text not null,
    password text not null,
    date_of_membership date not null,
    primary key (membership_id),
    foreign key (PID) references people (pid),
    foreign key (customer_id) references customers (customer_id)
};
```

<table>
<thead>
<tr>
<th>membership_id</th>
<th>PID</th>
<th>customer_id</th>
<th>username</th>
<th>password</th>
<th>date_of_membership</th>
</tr>
</thead>
<tbody>
<tr>
<td>m11111111</td>
<td>p1</td>
<td>c1</td>
<td>thisguy17</td>
<td>nymets17</td>
<td>2016-09-18</td>
</tr>
<tr>
<td>m22222222</td>
<td>p2</td>
<td>c2</td>
<td>coolgirl12</td>
<td>stuff1790</td>
<td>2013-10-23</td>
</tr>
<tr>
<td>m33333333</td>
<td>p4</td>
<td>c4</td>
<td>useruser20</td>
<td>nv.Pass3</td>
<td>2012-12-20</td>
</tr>
</tbody>
</table>

Functional Dependencies: membership_id → pid, customer_id, username, password, date_of_membership
Reservations: This table contains all information about the reservations a customer submits or a front desk worker processes.

```
CREATE TABLE reservations (  
    reservation_id char(8) not null,
    room_id char(6) not null,
    check_in date not null,
    check_out date not null,
    no_of_guests char(8) not null,
    smoking boolean not null,
    primary key (reservation_id),
    foreign key (room_id) references rooms (room_id)
);
```

Functional Dependencies: reservation_id → check_in, check_out, no_of_guests, room_id, customer_id, transaction_id

Reservation sample data on next slide
## Reservations Sample Data:

<table>
<thead>
<tr>
<th>reservation_id</th>
<th>room_id</th>
<th>check_in date</th>
<th>check_out date</th>
<th>no_of_guests</th>
<th>smoking</th>
</tr>
</thead>
<tbody>
<tr>
<td>rv1</td>
<td>rm1</td>
<td>2017-09---</td>
<td>2017-09-05</td>
<td>4</td>
<td>true</td>
</tr>
<tr>
<td>rv2</td>
<td>rm3</td>
<td>2016-03---</td>
<td>2016-03-25</td>
<td>1</td>
<td>false</td>
</tr>
<tr>
<td>rv3</td>
<td>rm4</td>
<td>2017-05---</td>
<td>2017-05-28</td>
<td>5</td>
<td>false</td>
</tr>
<tr>
<td>rv4</td>
<td>rm4</td>
<td>2017-09---</td>
<td>2017-09-14</td>
<td>5</td>
<td>true</td>
</tr>
<tr>
<td>rv5</td>
<td>rm5</td>
<td>2017-07---</td>
<td>2017-07-28</td>
<td>4</td>
<td>false</td>
</tr>
<tr>
<td>rv6</td>
<td>rm2</td>
<td>2013-08---</td>
<td>2013-08-23</td>
<td>1</td>
<td>false</td>
</tr>
<tr>
<td>rv7</td>
<td>rm7</td>
<td>2012-06---</td>
<td>2012-07-01</td>
<td>2</td>
<td>false</td>
</tr>
<tr>
<td>rv8</td>
<td>rm1</td>
<td>2015-09---</td>
<td>2015-09-07</td>
<td>4</td>
<td>true</td>
</tr>
<tr>
<td>rv9</td>
<td>rm3</td>
<td>2011-03---</td>
<td>2011-03-18</td>
<td>3</td>
<td>false</td>
</tr>
<tr>
<td>rv10</td>
<td>rm6</td>
<td>2014-06---</td>
<td>2014-06-22</td>
<td>5</td>
<td>true</td>
</tr>
</tbody>
</table>
Rooms: This table holds all the information about different rooms in the hotel.

```sql
CREATE TABLE rooms (
    room_id char(6) not null,
    roomtype_id char(10) not null,
    room_number char(5) not null,
    floor_number integer not null,
    status_id char(2) not null,
    primary key (room_id),
    foreign key (roomtype_id) references room_type (roomtype_id),
    foreign key (status_id) references room_status (status_id)
);
```

<table>
<thead>
<tr>
<th>room_id</th>
<th>roomtype_id</th>
<th>room_number</th>
<th>floor_number</th>
<th>status_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>type1</td>
<td>100</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>type2</td>
<td>200</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>type3</td>
<td>300</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>type3</td>
<td>120</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>type2</td>
<td>220</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>type3</td>
<td>305</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>type1</td>
<td>205</td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>

Functional Dependencies: room_id → roomtype_id, room_number, floor_num, status_id
Room_Type: This table holds all of the room types and their other attributes.

```
CREATE TABLE room_type (  
  roomtype_id char(8) not null,  
  room_name text not null,  
  max_guest integer not null,  
  smoking boolean not null,  
  description text not null,  
  room_priceUSD decimal(15,2) not null,  
primary key (roomtype_id)  
);
```

<table>
<thead>
<tr>
<th>roomtype_id</th>
<th>room_name</th>
<th>max_guest</th>
<th>smoking</th>
<th>description</th>
<th>room_priceUSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>type1</td>
<td>5</td>
<td>false</td>
<td>Two double</td>
<td>150.00</td>
</tr>
<tr>
<td>2</td>
<td>type2</td>
<td>2</td>
<td>false</td>
<td>One king</td>
<td>120.00</td>
</tr>
<tr>
<td>3</td>
<td>type3</td>
<td>8</td>
<td>true</td>
<td>Two bedroom</td>
<td>300.00</td>
</tr>
</tbody>
</table>

Functional Dependencies: roomtype_id \(\rightarrow\) room_name, max_guest, smoking, description, room_price_usd
Room_Status: This table displays whether the room is booked, vacant or being cleaned.

```
CREATE TABLE room_status (  
    status_id       char(2)   not null,  
    status_name     text      not null,  
    status_description text not null,  
    primary key (status_id)  
);
```

<table>
<thead>
<tr>
<th>status_id</th>
<th>status_name</th>
<th>status_description</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>Booked</td>
<td>This room is booked.</td>
</tr>
<tr>
<td>s2</td>
<td>Vacant</td>
<td>This room is completed.</td>
</tr>
<tr>
<td>s3</td>
<td>Being Cleaned</td>
<td>This room is in the ...</td>
</tr>
</tbody>
</table>

Functional Dependencies: status_id → status_name, status_description
Views, Triggers, Stored Procedures, Reports
Views: Total Cost

This query will find the total price a customer must pay for their visit, based upon reservation id. This is a quick and easy way for the total cost of a customer’s visit to be calculated and eventually processed through payment methods.

```
select (;
    select chargesUSD
    from miscellaneous_charges
    where miscellaneous_charges_id in (select miscellaneous_charges_id
                                            from miscellaneous_charges_add
                                            where transaction_id in (select transaction_id
                                                                       from transactions
                                                                       where reservation_id = 'rv5')))

+ (select room_priceUSD
    from room_type
    where roomtype_id in (select room_type_id
                            from rooms
                            where room_id in (select room_id
                                               from reservations
                                               where reservation_id in (select reservation_id
                                                                            from transactions
                                                                            where reservation_id = 'rv5')))) as totalCost;
```

<table>
<thead>
<tr>
<th>totalcost</th>
</tr>
</thead>
<tbody>
<tr>
<td>numeric</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>320.00</td>
</tr>
</tbody>
</table>
Views: Total Pay

This view will show how much an employee will make for working a certain number of hours. This is helpful for the manager or owner to calculate how much money they will have to pay their employees for working their hours that week.

```sql
select (select hourly_wageusd
from employees
where pid in (select pid
  from people
  where first_name = 'Taylor' AND
  last_name = 'Connelly'))
  *

(select total_hours
from timesheet
where employee_id in (select employee_id
  from employees
  where pid in (select pid
    from people
    where first_name = 'Taylor'
    AND
    last_name = 'Connelly')))
  as TotalPay;
```

<table>
<thead>
<tr>
<th>totalpay</th>
<th>numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>70.00</td>
</tr>
</tbody>
</table>
Views: Quick View Of Room Information
This view will give the employee working at the front desk a quick dashboard of the important information they need to know if a customer wants to book a room.

```sql
select room_id, room_number, floor_number, status_description, room_name, room_priceusd, max_guest
from rooms
inner join room_status on
rooms.status_id = room_status.status_id
inner join room_type on
rooms.roomtype_id = room_type.roomtype_id;
```

<table>
<thead>
<tr>
<th>room_id</th>
<th>room_number</th>
<th>floor_number</th>
<th>status_description</th>
<th>room_name</th>
<th>room_priceusd</th>
<th>max_guest</th>
</tr>
</thead>
<tbody>
<tr>
<td>rm1</td>
<td>100</td>
<td>1</td>
<td>This room is booked.</td>
<td>Double Queen</td>
<td>150.00</td>
<td>5</td>
</tr>
<tr>
<td>rm2</td>
<td>200</td>
<td>2</td>
<td>This room is comple...</td>
<td>Single King</td>
<td>120.00</td>
<td>2</td>
</tr>
<tr>
<td>rm3</td>
<td>300</td>
<td>3</td>
<td>This room is booked.</td>
<td>Suite Style</td>
<td>300.00</td>
<td>8</td>
</tr>
<tr>
<td>rm4</td>
<td>120</td>
<td>1</td>
<td>This room is booked.</td>
<td>Suite Style</td>
<td>300.00</td>
<td>8</td>
</tr>
<tr>
<td>rm5</td>
<td>220</td>
<td>2</td>
<td>This room is comple...</td>
<td>Single King</td>
<td>120.00</td>
<td>2</td>
</tr>
</tbody>
</table>
Views: Customer Information
This view shows customers that have made reservations, and their important information.

```sql
select first_name, last_name, contact_number
from people
where pid in (select pid
  from customers
  where customer_id in (select customer_id
    from transactions
    where reservation_id in (select reservation_id
      from reservations
      FULL outer join people ON people.pid = reservations.reservation_id)));
```

<table>
<thead>
<tr>
<th></th>
<th>first_name</th>
<th>last_name</th>
<th>contact_number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Jason</td>
<td>Haley</td>
<td>4587390869</td>
</tr>
<tr>
<td>2</td>
<td>Scott</td>
<td>Fritsch</td>
<td>1234567890</td>
</tr>
<tr>
<td>3</td>
<td>Jami</td>
<td>Domenico</td>
<td>6312546789</td>
</tr>
<tr>
<td>4</td>
<td>Alan</td>
<td>Laboseur</td>
<td>1118675301</td>
</tr>
<tr>
<td>5</td>
<td>Jack</td>
<td>Heuber</td>
<td>1345879978</td>
</tr>
</tbody>
</table>
Views: Non-Smoking Rooms

This view simply shows the rooms that are non-smoking, and also available to be booked at the time of the query.

```
select rooms.room_id, room_status.status_description, rooms.room_number, room_type.smoking
from rooms
inner join room_type on room_type.roomtype_id = rooms.roomtype_id
inner join room_status on rooms.status_id = room_status.status_id
and room_status.status_id = 's2'
and room_type.smoking = false;
```

<table>
<thead>
<tr>
<th>room_id</th>
<th>status_description</th>
<th>room_number</th>
<th>smoking</th>
</tr>
</thead>
<tbody>
<tr>
<td>rm5</td>
<td>This room is compl...</td>
<td>220</td>
<td>false</td>
</tr>
<tr>
<td>rm2</td>
<td>This room is compl...</td>
<td>200</td>
<td>false</td>
</tr>
</tbody>
</table>
Views: Gold Members
This view simply shows members that have been with the hotel for over a year. This accomplishment warrants special treatment from the hotel, whether that be some sort of discount or promo.

```sql
select first_name, last_name, contact_number, email_address
from people
where pid in (select pid
               from memberships
               where date_of_membership < '2017-12-01');
```

<table>
<thead>
<tr>
<th>first_name</th>
<th>last_name</th>
<th>contact_number</th>
<th>email_address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jason</td>
<td>Haley</td>
<td>4587390869</td>
<td>jason.haley@g...</td>
</tr>
<tr>
<td>Scott</td>
<td>Fritsch</td>
<td>1234567890</td>
<td>scott.fritsch@g...</td>
</tr>
<tr>
<td>Alan</td>
<td>Laboseur</td>
<td>1118675301</td>
<td>alan.lab@coolg...</td>
</tr>
</tbody>
</table>
Views: Room Status
This view tells you the status of the rooms in hotel. This is helpful for those employees who are booking the reservations.

```
select room_id, room_number, floor_number, status_description
from rooms
inner join room_status on rooms.status_id = room_status.status_id;
```

<table>
<thead>
<tr>
<th>room_id</th>
<th>room_number</th>
<th>floor_number</th>
<th>status_description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rm1</td>
<td>100</td>
<td></td>
<td>1 This room is booked.</td>
</tr>
<tr>
<td>rm2</td>
<td>200</td>
<td></td>
<td>2 This room is compl...</td>
</tr>
<tr>
<td>rm3</td>
<td>300</td>
<td></td>
<td>3 This room is booked.</td>
</tr>
<tr>
<td>rm4</td>
<td>120</td>
<td></td>
<td>1 This room is booked.</td>
</tr>
<tr>
<td>rm5</td>
<td>220</td>
<td></td>
<td>2 This room is compl...</td>
</tr>
<tr>
<td>rm6</td>
<td>305</td>
<td></td>
<td>3 This room is compl...</td>
</tr>
<tr>
<td>rm7</td>
<td>205</td>
<td>305</td>
<td>2 This room is booked.</td>
</tr>
</tbody>
</table>
This procedure allows the hotel front desk workers, as well as a manager to look up customer or employee personal information with the sole knowledge of the person’s first name, last name or both first and last name.

```
create or replace function findCustomer(TEXT, TEXT, REFCURSOR) returns refcursor as $$
declare
searchFirstName TEXT := $1;
searchLastName TEXT := $2;
resultSet REFCURSOR := $3;
begin
open resultset for
select *
from people
where first_name like searchFirstName
and
last_name like searchLastName;
return resultSet;
end;
$$ LANGUAGE plpgsql;
```

```
select findCustomer ('Taylor', 'Connelly', 'ref');
FETCH ALL FROM ref;
```
This procedure is a quick and easy way for a front desk employee to look up the details of a reservation utilizing only the reservation id.

```sql
create or replace function findReservation (TEXT, REFCURSOR) returns refcursor as $$
declare
searchReservation TEXT := $1;
resultSet REFCURSOR := $2;

begin
open resultSet for
select * from reservations
where reservation_id like searchReservation;
return resultSet;
end;
$$
LANGUAGE plpgsql;

select findReservation ('rv2%', 'ref');
FETCH ALL FROM ref;
```

<table>
<thead>
<tr>
<th>reservation_id</th>
<th>check_in date</th>
<th>check_out date</th>
<th>no_of_guests</th>
<th>room_id</th>
<th>smoking</th>
<th>customer_id</th>
<th>transaction_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>rv2</td>
<td>2016-03-...</td>
<td>1</td>
<td>rm3</td>
<td>false</td>
<td>c2</td>
<td>16666666</td>
</tr>
</tbody>
</table>
Reports:

Total number of reservations after 2015 (look for trends, see what to do to improve the number of reservations overtime):

```sql
select count(reservation_id) from reservations where check_in >= '2015-01-01';
```

<table>
<thead>
<tr>
<th>count bigint</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>

Total number of employees that have worked over 8 hours (could be adapted to show overtime pay information):

```sql
select count(employee_id) from timesheet where total_hours >= '8';
```

<table>
<thead>
<tr>
<th>count bigint</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>
Reports:

This report groups together how many reservations are being booked in each room type. This could show the owners of the hotel which rooms are in the highest demand, and could lead to changes within the hotels infrastructure, such as adding more of a certain room type to the hotel itself.

```
SELECT rooms.roomtype_id, COUNT(reservations.room_id) AS NumberOfRooms
FROM reservations
LEFT JOIN rooms
ON reservations.room_id = rooms.room_id
GROUP BY roomtype_id;
```

<table>
<thead>
<tr>
<th>roomtype_id</th>
<th>numberofrooms</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>type2</td>
</tr>
<tr>
<td>2</td>
<td>type1</td>
</tr>
<tr>
<td>3</td>
<td>type3</td>
</tr>
</tbody>
</table>
**Trigger: maxOccupants**

The hotel does not allow more than 6 occupants to a room in one reservation. Any time that this is entered into the database it is deleted immediately.

The following reservation was attempted to be added. The result is the dataset without rv11.

```sql
create or replace function maxOccupants()
returns trigger as
$$
begin
    if (NEW.no_of_guests > '6') then
        delete from reservations where no_of_guests = NEW.no_of_guests;
    end if;

    return new;
end;
$$ language plpgsql;

create trigger maxOccupants
after insert on reservations
for each row
execute procedure maxOccupants();

insert into reservations
values ('rv11', 'rm3', '2014-06-19', '2014-06-22', '9', true);
```

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>rv8</td>
<td>rm1</td>
<td>2015-09-...</td>
<td>2015-09-07</td>
<td>4</td>
<td>true</td>
</tr>
<tr>
<td>9</td>
<td>rv9</td>
<td>rm3</td>
<td>2011-03-...</td>
<td>2011-03-18</td>
<td>3</td>
<td>false</td>
</tr>
<tr>
<td>10</td>
<td>rv10</td>
<td>rm6</td>
<td>2014-06-...</td>
<td>2014-06-22</td>
<td>5</td>
<td>true</td>
</tr>
</tbody>
</table>
Trigger: getAge

The hotel does not want any employees or customers working or booking reservations under the age of 18 for liability reasons. Customers and employees are deleted from the database if this is the case.

```sql
create or replace function getAge()
returns trigger as
$$
begin
if (NEW.birthday > '2000-12-12') then
    delete from people where birthday = NEW.birthday;
end if;

return new;
end;
$$ language plpgsql;
```

```
create trigger getAge
after insert on people
for each row
execute procedure getAge();
```

```
insert into people
values ('p12', 'Noah', 'Fay', '12 Weirdo Street', 'New York', '11947', '2001-04-08', '4587937909', 'noah.fay@gmail.com');
```

**P12 Noah Fay not added**
Security:

Admin: This is either the owner of the business, or a person who would need access to everything within the database.

Hotel Manager: The Hotel Manager has much access to the database, as they need to be able to add all types of data into the database.

Front Desk: The Front Desk needs to be able to access the reservations and customer database, and book the reservations.

Housekeepers: These employees have the least amount of access to the database. They just need to know which rooms need to be cleaned.

```sql
create role admin;
grant all on all tables in schema public to admin;

CREATE ROLE hotel_manager;
grant select, insert, update on all tables in schema public to hotel_manager;

CREATE ROLE front_desk;
grant select, insert, update on reservations, customer to front_desk;

CREATE ROLE housekeepers;
grant select on room_status, rooms to housekeepers;
```
Known Problems/Future Enhancements:

- The sample data for the purposes of this project are limited. Much more data is needed in each of the tables for a thorough understanding of the scope of this database. Since I used a lot of tables, there was a plethora of information that needed to be added to make the database sufficient.
- I redid my entire ER diagram after I realized that I had repeats of different keys within tables that did not even connect.
- I had a lot of trouble joining tables because many tables have to dig deeper to get certain information (for example, the first and last name) since only one table holds that information.
- I had some issues with the foreign keys and primary keys with some of my tables. The tables have to be inserted in the order I submitted in my .sql code.
- The miscellaneous_charges_add table does not really have a primary key, but I did not know how else to work this.
- While naming the different IDs, I realized that I was running out of ideas for different number patterns for IDs. If I were to redo this, I would make sure that none of the IDs were without a letter in front. To redo that now would be extremely time consuming and I have internetworking. RIP.