PROJECT TICKETO

Designing a Database Architecture for TICKETO

PROJECT REPORT

Submitted by

Team Byte Me.

Under the guidance of

Prof. Christopher Keaton

For CDA 502 Database Management Systems

Master of Professional Studies in Data Sciences and Applications

Of

University at Buffalo, SUNY

Team a Database Architecture for TICKETO

Author Team

The following are the members of the author team who contributed to this project:

- **Sivaiah Naidu Yerraganti**: I have 5 years of experience as a Senior Data Engineer for EXL Services, Bangalore, India. I graduated from IIT BHU in Chemical Engineering in 2018.

- **Hruday Kumar Reddy Poreddy**: I have a Bachelor’s degree in Electronics and Communication Engineering from GITAM University (2017 - 2021). I have 2 years of work experience as an Application Development Analyst.

- **Tilak Karanam**: I have a degree in Computer Science and Engineering from Gitam University. I graduated in 2021. I worked as a Programmer Analyst in Cognizant for 2 years.

- **Spandana Chepuri**: I am from Hyderabad and have a degree in Computer Science and Engineering from TKR College of Engineering and Technology. I graduated in 2023.

- **Nithin Yadav Pandugayala**: I have a degree in Mechanical Engineering from Sastra University. I passed out in 2021. I worked as a DevOps Engineer in Cognizant Hyderabad.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 Project Purpose

This project's objective is to develop a database architecture for Ticketo, a ticketing corporation. Through its website, the company sells tickets for a variety of events, including concerts, plays, athletics, etc. Additionally, the website collects data from various sources, including payments, advertising campaigns, consumer feedback, etc. The database architecture will enable the organization to store and process this data efficiently and effectively, as well as to derive insights and value from it.

1.2 Project Goals

The main goals of this project are:

• To design a database capable of storing and organizing the organization's data into three distinct categories: internal data, customer data, and third-party data.

• To construct tables capable of capturing pertinent information regarding the entities and transactions involved in the business process, such as customers, events, venues, dates, ticket sales, payments, and fees, etc.

• To generate summary tables that provide pre-computed summaries of the data over specified intervals, such as daily, weekly, monthly, etc., for dashboarding and reporting.

• To demonstrate how to query the database and perform some rudimentary analysis using SQL.

1.3 Project Requirements

The main requirements of this project are:

- To use a relational database management system (RDBMS) that supports SQL as the query language.
- To follow the principles of database design, such as normalization, primary keys, foreign keys, etc.
- To use appropriate data types and constraints for each column in the tables.
- To use descriptive and consistent naming conventions for the tables and columns.
- To document the database architecture diagram and the table schemas.

1.4 Project Deliverables

The main deliverables of this project are:

- A complete architecture diagram of all the tables related to this data and their relationships.
- A database that contains all the tables with some dummy data.
- A SQL script that contains the queries to create and populate the tables.
- A SQL script that contains some sample queries to perform some basic analysis on the data.
- A report that explains the database design and the analysis results.

1.5 Project Exclusions

The main exclusions of this project are:

- The development of a web application or a user interface for the website or the database.
- The integration of the database with other applications or systems, such as payments and marketing.
- The implementation of advanced analytics or machine learning techniques on the data.
- The testing or validation of the database performance or security.

1.6 Project Challenges and Assumptions

The **Constraints** and assumptions of designing a database architecture in this project are.

- The first challenge is that there are multiple applications or systems that handle different aspects of the website, such as payments, marketing, etc.
- The first assumption is that everything is handled by one application, which simplifies the data model and the business logic.
- The second challenge is that there are many metrics or indicators that measure the performance and effectiveness of the ticket website, such as click-through rate, bounce rate, etc.
- The second assumption is that we are not considering website metrics and will be using transactional and customer data to do all the analysis and store them in our database and create dashboards and reports.

CHAPTER 2
Detailed Business Rules & Logical Diagram

2.1 Representation and Considerations of Business Rules

The following are the business rules that we will apply using the process codes which contains both internal and discount codes.

- Each process code is categorized as either policy or discount.
- Each customer, event, and vendor has a process code attached to them. At the time of booking, they are all subject to the following criteria:
    - The highest restriction is applied.
    - The highest discount is applied.
- Process codes that are expired are still stored in the database, but with the update timestamp changed and the active record indicator set to 0.
- To get the latest active process codes, one can filter the active record indicator as 1.
- The process codes column contains all applicable codes for each record. In the column, all codes are separated by ";".

2.2 Business Constraints and Discounts for Ticketo

We have created some business rules for Ticketo that will apply to customers, events, and vendors. These rules will help us manage restrictions and discounts in one place. Process codes are generated when we add these rules to the database.

<u>Business Constraints</u>

- New customers are restricted to a maximum of 5 ticket bookings for the first 48 hours.
- Age restriction is applicable to the customer based on the event category.
- For an event with priority code P1, the maximum tickets booked is 10 (all bookings combined).
- For an event with priority code P2, the maximum tickets booked is 20 (all bookings combined).
- For an event with priority code P3, the maximum tickets booked is 30 (all bookings combined).
- Vendor is blacklisted due to non-compliance of agreement.
- Vendor is blacklisted due to government regulations.
- Total booking capacity is 50 tickets per day per customer.

Discounts

- New customers get 20% off on initial booking.
- Discount for users using VISA credit card.
- Discount for users using PayPal.
- Promotional offers will have new process codes.

2.3 The cardinality of the ERD Diagram of Ticketo database

- Each customer_key may possess multiple payment_keys, whereas a single payment_key can only be associated with one customer_key.
- A single customer_key may be associated with many booking_keys, however a single booking_key can only be associated with one customer_key.
- A customer identifier (customer_key) has the ability to be associated with only a single process_key, and conversely, a process_key has the potential to be utilized by several customer identifiers (customer_keys).
- It is possible for a single payment_key to be associated with several booking_keys, whereas a booking_key can only be linked to a single payment_key.
- In this context, it is observed that an employee_key can only be associated with a single office_key, whereas an office_key can be linked to numerous employee_key.
- In the context of the booking_details table, it is possible for a booking_key to be associated with several event_keys. However, it is important to note that a single event_key can only be linked to a single booking_key in the booking table.
- In the booking_details table, it is possible for multiple booking_key and event_keys to be associated with a single discount_key, and conversely, a discount_key can be associated with multiple booking_key and event_keys.
- In the context of the database schema, it is possible for a booking_key and event_key from the booking_details table to be associated with a single event_key from the event_details table. However, it is also possible for a single event_key from the event_details table to be associated with numerous booking_key and event_key combinations from the booking_details table.
- A venue_key has the potential to be associated with many event_keys, but an event_key is limited to being associated with a single venue_key.
- There is a possibility for several venue_keys to be associated with a single vendor_keys, and conversely, vendor_keys to be associated with multiple venue_keys.

- A vendor_key has the potential to be associated with many event_keys, but an event_key is limited to being associated with a single vendor_key.
- Multiple process codes can be associated with multiple event keys, and conversely, multiple event keys can be associated with multiple process codes.

CHAPTER 3
DDL for TICKETO

DDL FOR CUSTOMER TABLE

```
CREATE TABLE customers_details
(
    customer_key INT GENERATED BY DEFAULT AS IDENTITY,
    customer_id VARCHAR2(50) NOT NULL,
    first_name VARCHAR2(50) NOT NULL,
    last_name VARCHAR2(50) NOT NULL,
    phone INT,
    email VARCHAR2(50),
    address_line1 VARCHAR2(50),
    address_line2 VARCHAR2(50),
    city VARCHAR2(50),
    state VARCHAR2(50),
    zipcode INT,
    country VARCHAR(50),
    age INT,
    process_key INT,
    PRIMARY KEY(customer_key),
    FOREIGN KEY (process_key) REFERENCES business_constraints (process_key)
);
```

DDL FOR BOOKING DETAILS

```
CREATE TABLE booking_details
(
    booking_key INT NOT NULL,
    event_key INT NOT NULL,
    no_of_tickets_booked INT NOT NULL,
    booking_value INT NOT NULL,
    payment_value FLOAT,
    discount_key INT,
    PRIMARY KEY(booking_key, event_key),
    FOREIGN KEY (event_key) REFERENCES event_details (event_key),
    FOREIGN KEY (discount_key) REFERENCES discounts (discount_key)
) ;
```

DDL FOR BUSINESS CONSTRAINTS

```sql
CREATE TABLE business_constraints
(
    process_key INT GENERATED BY DEFAULT AS IDENTITY,
    process_code INT NOT NULL,
    business_type VARCHAR2(50) NOT NULL,
    description VARCHAR2(50) NOT NULL,
    applicable_from TIMESTAMP,
    applicable_to TIMESTAMP,
    create_timestamp TIMESTAMP,
    update_timestamp TIMESTAMP,
    active_record_ind INT,
    PRIMARY KEY(process_key)
);
```

DDL FOR DISCOUNTS

```sql
CREATE TABLE discounts
(
    discount_key INT GENERATED BY DEFAULT AS IDENTITY,
    discount_code INT NOT NULL,
    discount_value INT NOT NULL,
    description VARCHAR2(50) NOT NULL,
    applicable_from TIMESTAMP,
    applicable_to TIMESTAMP,
    create_timestamp TIMESTAMP,
    update_timestamp TIMESTAMP,
    active_record_ind INT,
    PRIMARY KEY(discount_key)
);
```

DDL FOR VENDOR DETAILS

```
CREATE TABLE vendor_details
(
    vendor_key INT GENERATED BY DEFAULT AS IDENTITY,
    vendor_id VARCHAR2(50) NOT NULL,
    vendor_category VARCHAR2(50) NOT NULL,
    vendor_name VARCHAR2(50) NOT NULL,
    share_percentage INT,
    poc_name VARCHAR2(50),
    poc_number INT,
    bakcup_poc_number INT,
    contract_start TIMESTAMP,
    contract_end TIMESTAMP,
    process_key INT,
    active_record_ind INT,
    PRIMARY KEY(vendor_key) ,
    FOREIGN KEY (process_key) REFERENCES business_constraints (process_key)
) ;
```

DDL FOR PAYMENT DETAILS :

```
CREATE TABLE payment_details(
    payment_key INT GENERATED BY DEFAULT AS IDENTITY,
    payment_id INT,
    card_number VARCHAR2(50),
    payment_type VARCHAR2(50),
    payment_merchant VARCHAR2(50),
    customer_key INT,
    PRIMARY KEY (payment_key),
    FOREIGN KEY (customer_key) REFERENCES customers_details (customer_key)
);
```

DDL FOR VENUE DETAILS :

```
CREATE TABLE venue_details
(
    venue_key INT GENERATED BY DEFAULT AS IDENTITY,
    venue_id INT NOT NULL,
    name VARCHAR2(50) NOT NULL,
    address VARCHAR2(100) NOT NULL,
    city VARCHAR2(50),
    zipcode INT,
    state VARCHAR2(50),
    vendor_key INT,
    active_record_ind INT,
    PRIMARY KEY(venue_key),
    FOREIGN KEY (vendor_key) REFERENCES vendor_details (vendor_key)
);
```

DDL FOR BOOKING :

```
CREATE TABLE booking
(
    booking_key INT GENERATED BY DEFAULT AS IDENTITY,
    booking_id VARCHAR2(50) NOT NULL,
    booking_datetime TIMESTAMP NOT NULL,
    boking_status VARCHAR2(50) NOT NULL,
    comments VARCHAR2(50),
    customer_key INT,
    transaction_id INT,
    payment_key INT,
    payment_status VARCHAR2(50),
    PRIMARY KEY(booking_key) ,
    FOREIGN KEY (payment_key) REFERENCES payment_details (payment_key),
    FOREIGN KEY (customer_key) REFERENCES customers_details (customer_key)
) ;
```

DDL FOR EVENT DETAILS :

```
CREATE TABLE event_details
(
    event_key INT GENERATED BY DEFAULT AS IDENTITY,
    event_id INT NOT NULL,
    venue_key INT NOT NULL,
    time_start TIMESTAMP NOT NULL,
    time_end TIMESTAMP,
    booking_start TIMESTAMP,
    booking_end TIMESTAMP,
    event_name VARCHAR2(100),
    max_occupancy INT,
    current_occupancy INT,
    priority VARCHAR2(50),
    description VARCHAR2(50),
    ticket_price INT,
    process_key INT,
    event_type VARCHAR2(50),
    active_record_ind INT,
    PRIMARY KEY(event_key),
    FOREIGN KEY (venue_key) REFERENCES venue_details (venue_key)
) ;
```

DDL FOR OFFICES :

```
CREATE TABLE offices
(
    office_key INT GENERATED BY DEFAULT AS IDENTITY,
    office_code VARCHAR2(50) NOT NULL,
    city VARCHAR2(50),
    phone INT,
    address_line1 VARCHAR2(50),
    address_line2 VARCHAR2(50),  +
    create_date TIMESTAMP,
    update_date TIMESTAMP,
    active_record_ind INT,
    PRIMARY KEY (office_key)
) ;
```

DDL FOR EMPLOYEE DETAILS :

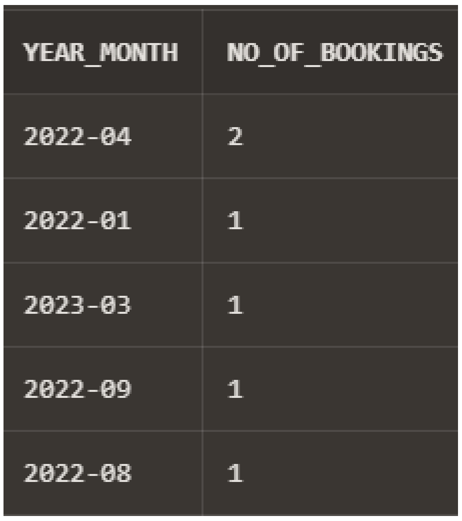
```sql
CREATE TABLE employee_details
(
    employee_key INT GENERATED BY DEFAULT AS IDENTITY,
    employee_id VARCHAR2(50) NOT NULL,
    team VARCHAR2(50) NOT NULL,
    employee_role VARCHAR2(50) NOT NULL,
    first_name VARCHAR(50),
    last_name VARCHAR2(50),
    phone INT,
    email VARCHAR2(50),
    address_line1 VARCHAR2(50),
    address_line2 VARCHAR2(50),
    city VARCHAR2(50),
    state VARCHAR2(50),
    zipcode INT,
    country VARCHAR(50),
    age INT,
    office_key INT,
    PRIMARY KEY (employee_key),
    FOREIGN KEY (office_key) REFERENCES offices (office_key)
) ;
```

CHAPTER 3
POPULATED DATA AND QUERY OUTPUT

QUERY 1: Count all the tickets purchased by Andrew Bryan in each year and month

```
select
    TO_CHAR(TRUNC(BOOKING_DATETIME, 'MM'), 'YYYY-MM') as year_month,
    count(BOOKING_ID) as no_of_bookings
from
    booking b
    join customers_details c on c.CUSTOMER_KEY = b.CUSTOMER_KEY
where
    FIRST_NAME = 'Andrew'
    and LAST_NAME = 'Bryan'
group by
    TO_CHAR(TRUNC(BOOKING_DATETIME, 'MM'), 'YYYY-MM')
order by
    2 desc
```

OUTPUT FOR QUERY 1:

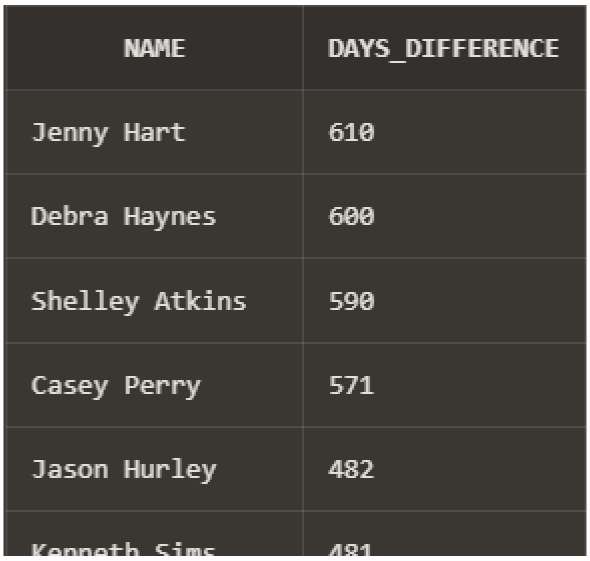


YEAR_MONTH	NO_OF_BOOKINGS
2022-04	2
2022-01	1
2023-03	1
2022-09	1
2022-08	1

QUERY 2: Report the list of customers whose difference(days) between most recent and oldest booking date is more than 365 days

```
select
    first_name || ' ' || last_name as name,
    max(TRUNC(BOOKING_DATETIME)) - min(TRUNC(BOOKING_DATETIME)) as
days_difference
from
    booking b
    left join customers_details cd on b.CUSTOMER_KEY = cd.CUSTOMER_KEY
group by
    first_name || ' ' || last_name
Having
    max(TRUNC(BOOKING_DATETIME)) - min(TRUNC(BOOKING_DATETIME)) > 365
order by
    days_difference desc
```

OUTPUT FOR QUERY 2:

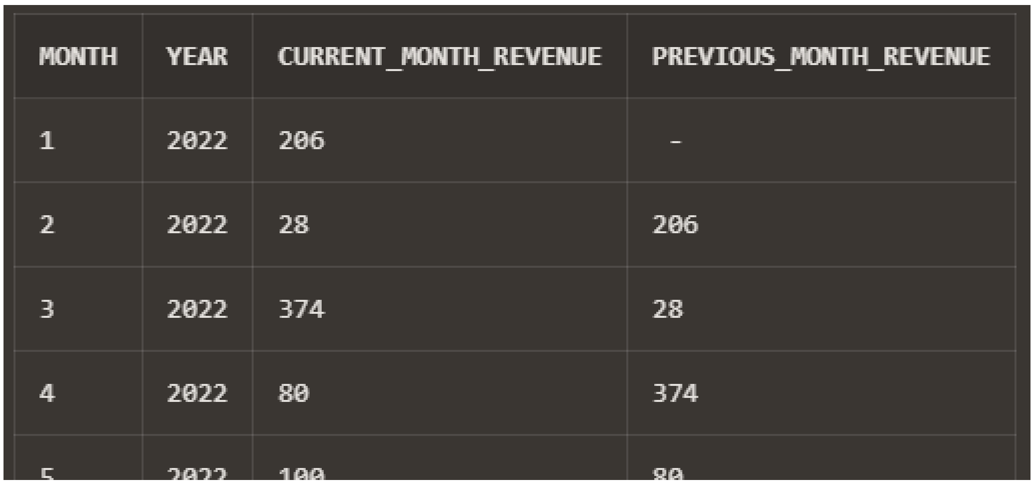


NAME	DAYS_DIFFERENCE
Jenny Hart	610
Debra Haynes	600
Shelley Atkins	590
Casey Perry	571
Jason Hurley	482
Kenneth Sims	481

QUERY 3: What is the comparision of previous month to current month revenue.

```
select
    month,
    year,
    ceil(monthly_revenue) as current_month_revenue,
    lag(ceil(monthly_revenue)) over (ORDER BY YEAR, MONTH) as
previous_month_revenue
from
    (
        select
            EXTRACT(MONTH FROM TRUNC(BOOKING_DATETIME) ) as month,
            EXTRACT( YEAR FROM TRUNC(BOOKING_DATETIME) ) as year,
            sum(bd.payment_value) as monthly_revenue
        from
            booking b
            join booking_details bd on b.BOOKING_KEY = bd.BOOKING_KEY
        group by
            EXTRACT( YEAR FROM TRUNC(BOOKING_DATETIME) ),
            EXTRACT( MONTH FROM TRUNC(BOOKING_DATETIME) )
    )
```

OUTPUT FOR QUERY 3:

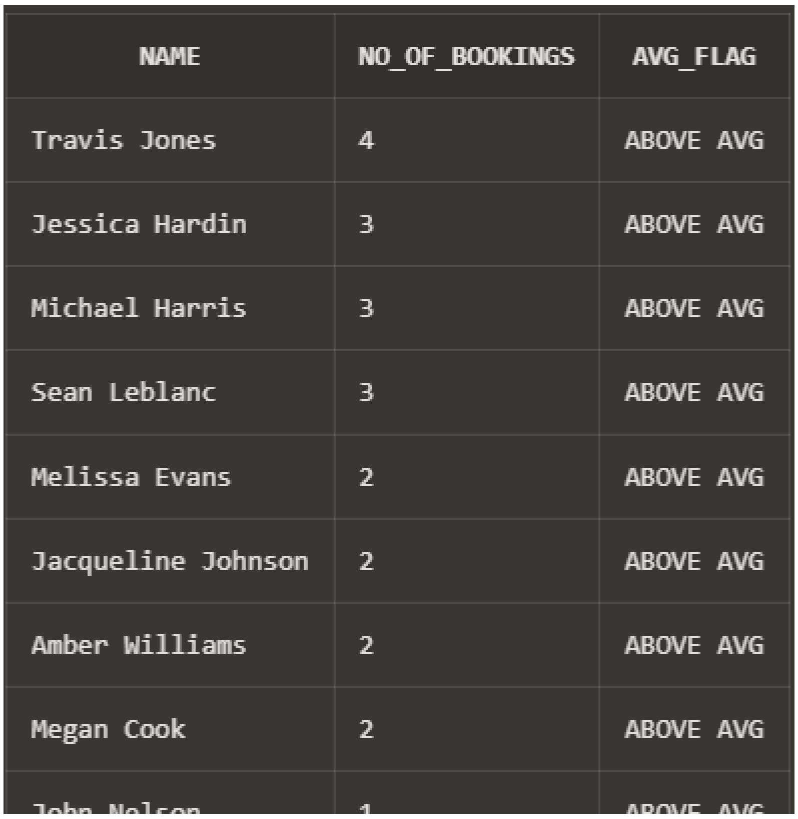


MONTH	YEAR	CURRENT_MONTH_REVENUE	PREVIOUS_MONTH_REVENUE
1	2022	206	-
2	2022	28	206
3	2022	374	28
4	2022	80	374
5	2022	100	80

QUERY 4: Who are the customers that have booking value totaled above the avg for 2022 by a above AVG or below avg flag and report no of bookings they have made

```sql
with AVG as (
select
    EXTRACT(YEAR FROM TRUNC(BOOKING_DATETIME)) as year,
    ROUND(AVG(PAYMENT_VALUE), 3) Per_Year_AVG
from
    booking b
    join booking_details bd on b.BOOKING_KEY = bd.BOOKING_KEY
where
    EXTRACT(YEAR FROM TRUNC(BOOKING_DATETIME)) = 2022
group by
    EXTRACT(YEAR FROM TRUNC(BOOKING_DATETIME))),
Cust_Avg as (
select
    b.customer_key,
    EXTRACT(YEAR FROM TRUNC(BOOKING_DATETIME)) as Year,
    count(distinct b.booking_key) as No_of_bookings,
    Round(AVG(bd.PAYMENT_VALUE), 3) as Avg_booking_value
from
    booking b
    join booking_details bd on b.BOOKING_KEY = bd.BOOKING_KEY
where
    EXTRACT(YEAR FROM TRUNC(BOOKING_DATETIME)) = 2022
group by
    b.customer_key,
    EXTRACT(YEAR FROM TRUNC(BOOKING_DATETIME))
)
select
    cd.first_name || ' ' || cd.last_name as Name,
    NO_OF_BOOKINGS,
    Case
        when AVG_BOOKING_VALUE > Per_Year_AVG THEN 'ABOVE AVG'
        when AVG_BOOKING_VALUE <= Per_Year_AVG THEN 'BELOW AVG'
        else ''
    end as AVG_FLAG
from
    Cust_Avg ca
    join AVG a on ca.year = a.year
    join customers_details cd on ca.CUSTOMER_KEY = cd.CUSTOMER_KEY
order by
    3,
    2 DESC
```

OUTPUT FOR QUERY 4:

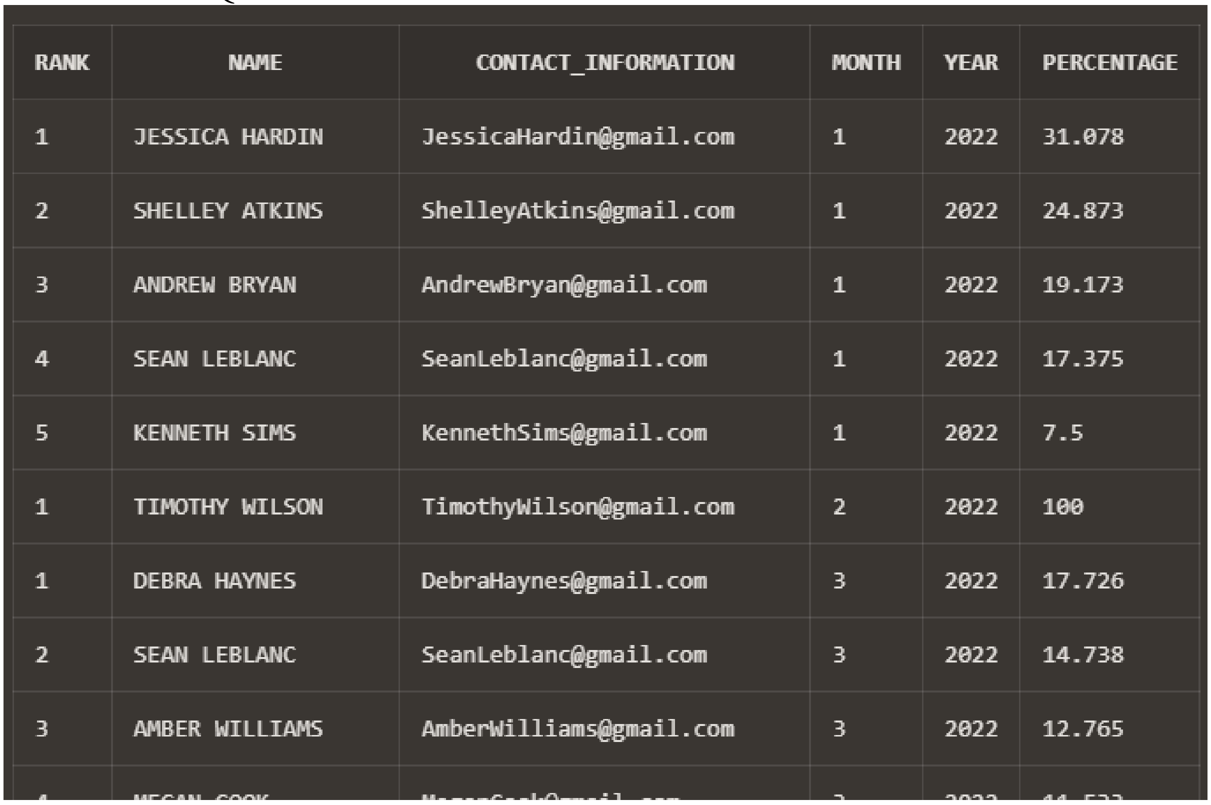


NAME	NO_OF_BOOKINGS	AVG_FLAG
Travis Jones	4	ABOVE AVG
Jessica Hardin	3	ABOVE AVG
Michael Harris	3	ABOVE AVG
Sean Leblanc	3	ABOVE AVG
Melissa Evans	2	ABOVE AVG
Jacqueline Johnson	2	ABOVE AVG
Amber Williams	2	ABOVE AVG
Megan Cook	2	ABOVE AVG
John Nelson	1	ABOVE AVG

QUERY 5: Compute each customer's revenue as a percentage of total revenue per month. output customer name in uppercase and contact information (customer email and if email is null replace with his phone number)

```
with per_month_avg as (
    select
        distinct EXTRACT(MONTH FROM TRUNC(BOOKING_DATETIME) ) as month,
        EXTRACT( YEAR FROM TRUNC(BOOKING_DATETIME) ) as year,
        round(sum(bd.payment_value) over ( partition by EXTRACT( YEAR FROM
TRUNC(BOOKING_DATETIME) ), EXTRACT( MONTH FROM TRUNC(BOOKING_DATETIME) )), 3 )
as monthly_Rev
    from
        booking b
        join booking_details bd on b.BOOKING_KEY = bd.BOOKING_KEY
),
Cust_avg as(
    select
        customer_key,
        EXTRACT( MONTH FROM TRUNC(BOOKING_DATETIME) ) as month,
        EXTRACT( YEAR FROM TRUNC(BOOKING_DATETIME) ) as year,
        round(sum(bd.payment_value), 3) as monthly_Rev_per_cust
    from
        booking b
        join booking_details bd on b.BOOKING_KEY = bd.BOOKING_KEY
    group by
        customer_key,
        EXTRACT( YEAR FROM TRUNC(BOOKING_DATETIME) ),
        EXTRACT( MONTH FROM TRUNC(BOOKING_DATETIME) )
)
select
    DENSE_RANK() over (
        partition by ca.YEAR,
        ca.MONTH
        order by
            round((MONTHLY_REV_PER_CUST / MONTHLY_REV) * 100, 3) DESC) as
Rank,
    UPPER(cd.first_name || ' ' || cd.last_name) as Name,
    NVL(cd.email, TO_CHAR(phone)) as contact_information,
    ca.MONTH,
    ca.YEAR,
    round((MONTHLY_REV_PER_CUST / MONTHLY_REV) * 100, 3) as percentage
from
    Cust_avg ca
    join per_month_avg pma on ca.month = pma.month
    and ca.year = pma.year
    join customers_details cd on ca.CUSTOMER_KEY = cd.CUSTOMER_KEY
order by
    ca.YEAR,
    ca.MONTH,
Rank
```

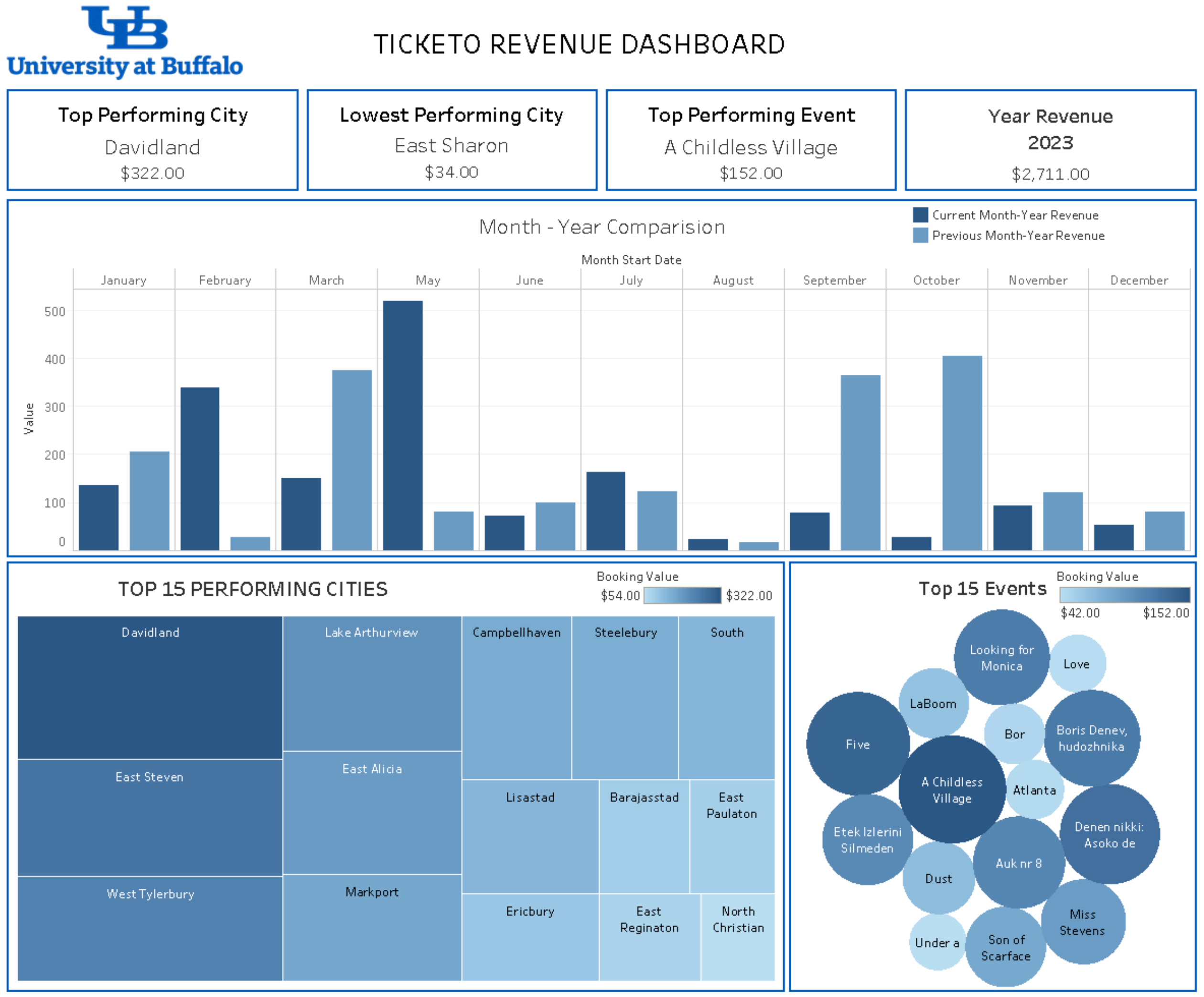
OUTPUT FOR QUERY 5 :



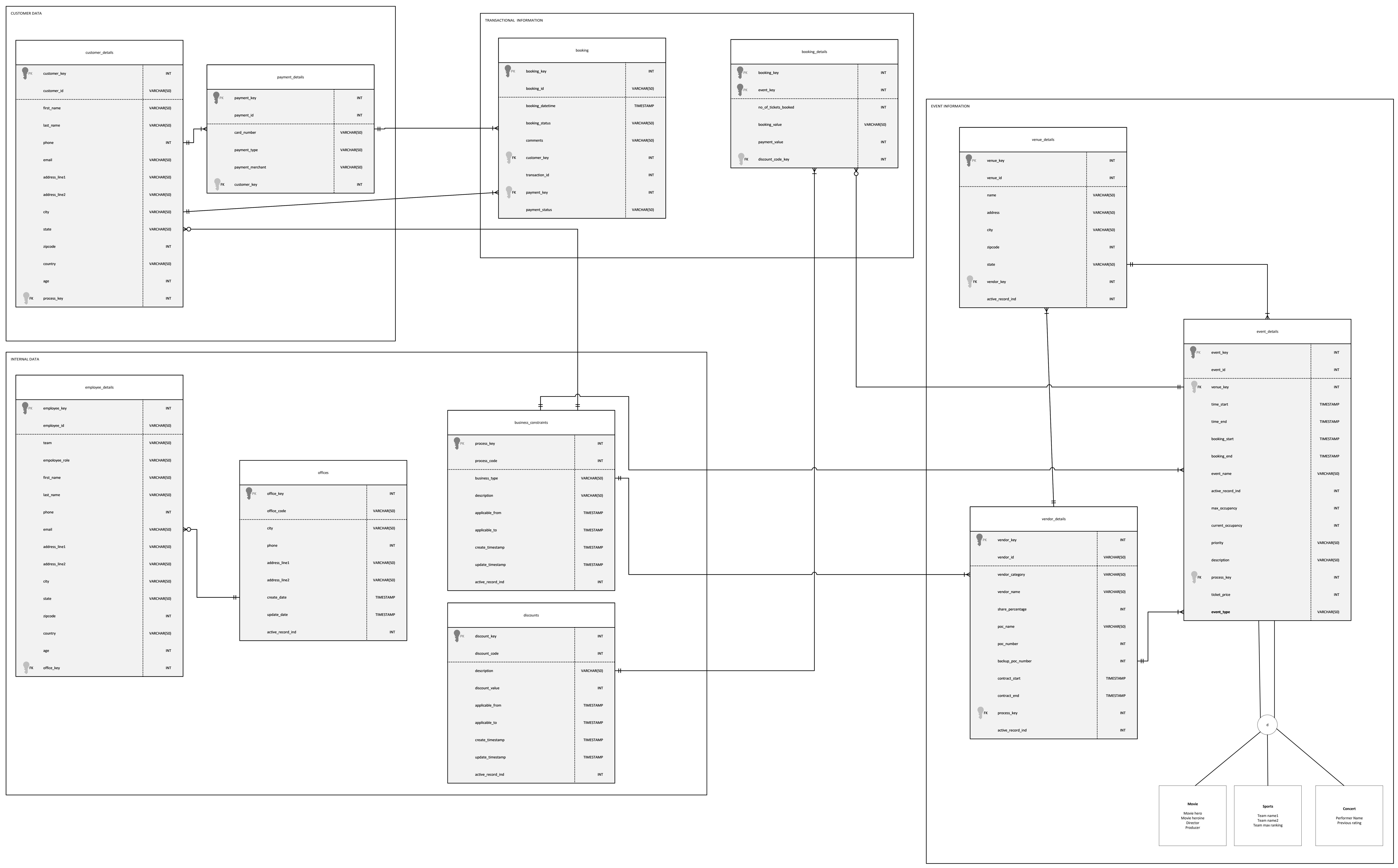
RANK	NAME	CONTACT_INFORMATION	MONTH	YEAR	PERCENTAGE
1	JESSICA HARDIN	JessicaHardin@gmail.com	1	2022	31.078
2	SHELLEY ATKINS	ShelleyAtkins@gmail.com	1	2022	24.873
3	ANDREW BRYAN	AndrewBryan@gmail.com	1	2022	19.173
4	SEAN LEBLANC	SeanLeblanc@gmail.com	1	2022	17.375
5	KENNETH SIMS	KennethSims@gmail.com	1	2022	7.5
1	TIMOTHY WILSON	TimothyWilson@gmail.com	2	2022	100
1	DEBRA HAYNES	DebraHaynes@gmail.com	3	2022	17.726
2	SEAN LEBLANC	SeanLeblanc@gmail.com	3	2022	14.738
3	AMBER WILLIAMS	AmberWilliams@gmail.com	3	2022	12.765
4	MEGAN COOK	MeganCook@gmail.com	3	2022	11.533

CHAPTER 4
VISUALISATION USING TABLEAU

Link to dashboard published on Tableau Public – Ticketo Revenue Dashboard



Ticketo ER Diagram



customer_details

PK	customer_key	INT
	customer_id	VARCHAR(50)
	first_name	VARCHAR(50)
	last_name	VARCHAR(50)
	phone	INT
	email	VARCHAR(50)
	address_line1	VARCHAR(50)
	address_line2	VARCHAR(50)
	city	VARCHAR(50)
	state	VARCHAR(50)
	zipcode	INT
	country	VARCHAR(50)
	age	INT
FK	process_key	INT

payment_details

PK	payment_key	INT
	payment_id	INT
	card_number	VARCHAR(50)
	payment_type	VARCHAR(50)
	payment_merchant	VARCHAR(50)
FK	customer_key	INT

booking

PK	booking_key	INT
	booking_id	VARCHAR(50)
	booking_datetime	TIMESTAMP
	booking_status	VARCHAR(50)
	comments	VARCHAR(50)
FK	customer_key	INT
	transaction_id	INT
FK	payment_key	INT
	payment_status	VARCHAR(50)

booking_details

PK	booking_key	INT
PK	event_key	INT
	no_of_tickets_booked	INT
	booking_value	VARCHAR(50)
	payment_value	INT
FK	discount_code_key	INT

venue_details

PK	venue_key	INT
	venue_id	INT
	name	VARCHAR(50)
	address	VARCHAR(50)
	city	VARCHAR(50)
	zipcode	INT
	state	VARCHAR(50)
FK	vendor_key	INT
	active_record_ind	INT

event_details

PK	event_key	INT
	event_id	INT
FK	venue_key	INT
	time_start	TIMESTAMP
	time_end	TIMESTAMP
	booking_start	TIMESTAMP
	booking_end	TIMESTAMP
	event_name	VARCHAR(50)
	active_record_ind	INT
	max_occupancy	INT
	current_occupancy	INT
	priority	VARCHAR(50)
	description	VARCHAR(50)
FK	process_key	INT
	ticket_price	INT
	event_type	VARCHAR(50)

employee_details

PK	employee_key	INT
	employee_id	VARCHAR(50)
	team	VARCHAR(50)
	empoloyee_role	VARCHAR(50)
	first_name	VARCHAR(50)
	last_name	VARCHAR(50)
	phone	INT
	email	VARCHAR(50)
	address_line1	VARCHAR(50)
	address_line2	VARCHAR(50)
	city	VARCHAR(50)
	state	VARCHAR(50)
	zipcode	INT
	country	VARCHAR(50)
	age	INT
FK	office_key	INT

offices

PK	office_key	INT
	office_code	VARCHAR(50)
	city	VARCHAR(50)
	phone	INT
	address_line1	VARCHAR(50)
	address_line2	VARCHAR(50)
	create_date	TIMESTAMP
	update_date	TIMESTAMP
	active_record_ind	INT

business_constraints

PK	process_key	INT
	process_code	INT
	business_type	VARCHAR(50)
	description	VARCHAR(50)
	applicable_from	TIMESTAMP
	applicable_to	TIMESTAMP
	create_timestamp	TIMESTAMP
	update_timestamp	TIMESTAMP
	active_record_ind	INT

discounts

PK	discount_key	INT
	discount_code	INT
	description	VARCHAR(50)
	discount_value	INT
	applicable_from	TIMESTAMP
	applicable_to	TIMESTAMP
	create_timestamp	TIMESTAMP
	update_timestamp	TIMESTAMP
	active_record_ind	INT

vendor_details

PK	vendor_key	INT
	vendor_id	VARCHAR(50)
	vendor_category	VARCHAR(50)
	vendor_name	VARCHAR(50)
	share_percentage	INT
	poc_name	VARCHAR(50)
	poc_number	INT
	backup_poc_number	INT
	contract_start	TIMESTAMP
	contract_end	TIMESTAMP
FK	process_key	INT
	active_record_ind	INT