Database Implementation

Design Report

Mountither Al Rashid 102486181

Contents

Overview

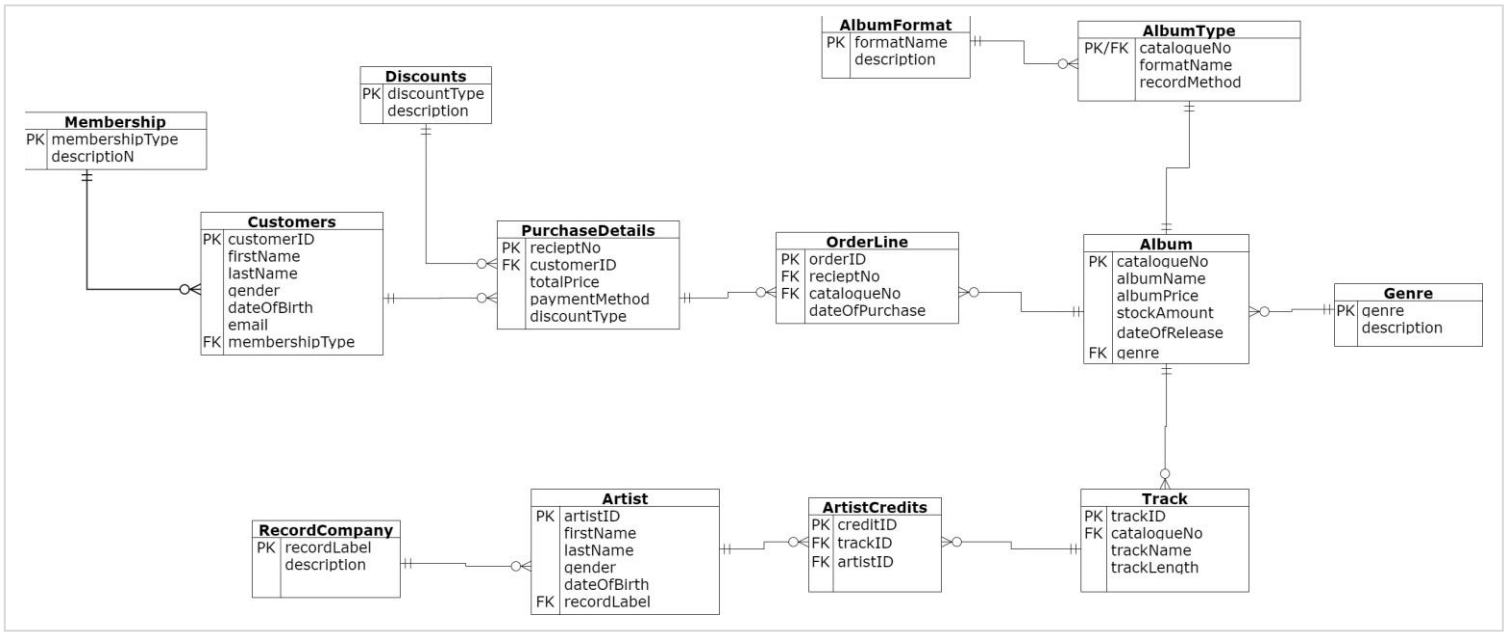

Figure 1 - UML diagram representing the entities and relations in the Music Database.

About the database

A music store has many stakeholders involved. The processes of purchasing a physical album, the characteristics of that album and the industry revolving around the music are recorded. This music store records the customers details and allocates a membership status, while discounts are applied to some purchases depending on the circumstance. Each customer that purchases an album receives a receipt number that categorises the payment method, discount type (if applicable), the total price and the corresponding customer id. The aim of the orderline entity is to unify the receipt number and each album catalogue id through the unique attribute, order id. The catalogue id uniquely identifies an album. Each order id has a date of purchase. The album entity consists of a name, retail price, stock amount in inventory, date of release and the main genre. Each album contains many tracks, each track includes a name and length (in minutes). The artist is connected to the track through the artist credits entity. The artist credits entity includes the artist credited to the track with a unique id (creditID). The purpose for this junction entity is to resolve the many feature (artist) that occur on a track. The artist is also associated with a record company. For further observations refer to Figure 1 above. This information is recorded and stored for many use cases, discussed in the following section.

Main Use Cases

The music store would find the most value in the customer information stored. The various attributes involved for the customer needs to be maintained consistently within the system. This eases retrieval and efficiency for the employees. A Relational Database is fitting in the situation since the structure is implemented with logical and the relations are broken up into relevant factions.

From a financial perspective, the music store can compute the revenue relative to the customer, the cyclical nature of business and the attributes related to the music. Such information is valuable to the business. Aggregations or other deductions can improve timing of promotions. The effectiveness of discount campaigns and membership types can be analysed easily in this Relational database.

The demographics are also analysed in this context. The type of album can contribute to the typical customers preference. This also applies to the genre and the artist. The retrieved information conveys a resolve to a query raised by the business.

Script Used for Design

Each ‘create table’ statement is executed individually in a sequential manner.

Creation

```
create table Membership(
    membershipType varchar(20) not null,
    description varchar(100),
    PRIMARY KEY (membershipType)
);

create table Customers(
    customerID int(10) not null AUTO_INCREMENT,
    firstName varchar(30),
    lastName varchar(30),
    gender varchar(6) not null check(gender in ('Male', 'Female', 'Other')),
    dateOfBirth date,
    email varchar(30),
    membershipType varchar(20),
    PRIMARY key (customerID),
    FOREIGN key (membershipType) REFERENCES Membership (membershipType)
);

create table Discounts(
    discountType varchar(20) not null,
    description varchar(100),
    PRIMARY KEY (discountType)
);

create table PurchaseDetails(
    recieptNo int(10) not null AUTO_INCREMENT,
    customerID int(10) not null,
    totalPrice decimal(10,2) not null,
    paymentMethod varchar(20),
    discountType varchar(20),
    PRIMARY KEY (recieptNo),
    FOREIGN KEY (discountType) REFERENCES Discounts (discountType),
    FOREIGN KEY (customerID) REFERENCES Customers (customerID)
)

-- catalogue number refers to the album id. Similar to barcode
create table OrderLine(
    orderID int(10) not null AUTO_INCREMENT,
    recieptNo int(10) not null,
    catalogueNo int(10) not null,
```

```
    dateOfPurchase datetime,
    PRIMARY KEY (orderID),
    FOREIGN KEY (recieptNo) REFERENCES PurchaseDetails (recieptNo)
)

create table Album (
    catalogueNo int(10) not null AUTO_INCREMENT,
    albumName varchar(40),
    albumPrice  decimal(10,2),
    stockAmount int(10),
    dateOfRelease date,
    genre varchar(10),
    PRIMARY KEY (catalogueNo)
);

ALTER TABLE OrderLine ADD FOREIGN KEY (catalogueNo) REFERENCES Album(catalogue
No);

create table Genre (
    genre varchar(10) not null,
    description varchar(100),
    PRIMARY KEY (genre)
);

ALTER TABLE Album ADD FOREIGN KEY (genre) REFERENCES Album(genre);

create table AlbumType(
    catalogueNo int(10) not null,
    formatName varchar(20),
    recordMethod varchar(20),
    PRIMARY KEY (catalogueNo),
    FOREIGN KEY (catalogueNo) REFERENCES Album(catalogueNo)
);

create table AlbumFormat (
    formatName varchar(20) not null,
    description varchar(100),
    PRIMARY KEY (formatName)
);

ALTER TABLE AlbumType ADD FOREIGN KEY (formatName) REFERENCES AlbumFormat(form
atName);

create table Track (
    trackID int(10) not null AUTO_INCREMENT,
    catalogueNo int(10),
    trackName VARCHAR(20),
    trackLength decimal,
```

```
    PRIMARY KEY (trackID),
    FOREIGN KEY (catalogueNo) REFERENCES Album(catalogueNo)
);


create table ArtistCredits (
    creditID int(10) not null AUTO_INCREMENT,
    trackID int(10) not null,
    artistID int(10) not null,
    PRIMARY KEY (creditID),
    FOREIGN KEY (trackID) REFERENCES Track (trackID)
);

create table Artist(
    artistID int(10) not null AUTO_INCREMENT,
    firstName varchar(50),
    lastName varchar(50),
    gender varchar(6) not null check(gender in ('Male', 'Female', 'Other')),
    dateOfBirth date,
    recordLabel varchar(30),
    PRIMARY KEY (artistID)
);

ALTER TABLE ArtistCredits ADD FOREIGN KEY (artistID) REFERENCES Artist(artistI
D);

create table RecordCompany (
    recordLabel varchar(30) not null,
    description varchar(100),
    PRIMARY KEY (recordLabel)
);

ALTER TABLE Artist ADD FOREIGN KEY (recordLabel) REFERENCES RecordCompany(reco
rdLabel);
```

Data Types

Various data types are used in this database.

- Identification based attributes are integers that are auto incremented in the time being. This can change if actual identification is introduced. The auto increment function is practically row number with a ‘not null’.
- Checks are used for gender to constraint the selections.
- Currency based attributes are expressed through the decimal data type with 2 decimal points.
- Descriptions are attributed with varchar and size of 100
- Date of birth for the human entities are given the date datatype with the format ‘yyyy-mm-dd’
- Other dates such as purchase dates and release dates include the time using the ‘datetime’ datatype.
- Not nulls are utilised for all unique identification and necessary attributes that are required to be recorded for database integrity.
- Nulls are observed in the discount type attribute for the purchase details entity. Not all purchases have a discount.

Information about the Music Store

The following section provides descriptive information of the music store via MySQL scripts.

Customers

The count of customers, the proportion in gender and membership type between them is shown in the below figure. Also, the occurrences of email Domain names from their email is computed.

```
MariaDB [musicdb]> select count(*) as CustomerCount from customers;
+---------------+
| CustomerCount |
+---------------+
|           356 |
+---------------+
1 row in set (0.000 sec)
```

```
MariaDB [musicdb]> select gender, count(*) as CustomerCount from customers group by gender;
+--------+---------------+
| gender | CustomerCount |
+--------+---------------+
| Female |           165 |
| Male   |           191 |
+--------+---------------+
2 rows in set (0.001 sec)
```

```
MariaDB [musicdb]> select membershiptype, count(*) as CustomerCount from customers group by membershiptype;
+----------------+---------------+
| membershiptype | CustomerCount |
+----------------+---------------+
| gold           |           126 |
| silver         |           127 |
| standard       |           103 |
+----------------+---------------+
3 rows in set (0.000 sec)
```

```
MariaDB [musicdb]> select (SUBSTRING_INDEX(SUBSTR(email, INSTR(email, '@') + 1),'.',1)) as emailDomain, Count(*) as occurances
    -> select (SUBSTRING_INDEX(SUBSTR(email, INSTR(email, '@') + 1),'.',1)) as emailDomain, Count(*) as occurances
       from customers
    -> group by (SUBSTRING_INDEX(SUBSTR(email, INSTR(email, '@') + 1),'.',1))
    -> order by occurances desc limit 10;
+-------------+------------+
| emailDomain | occurances |
+-------------+------------+
| google      |         11 |
| amazon      |          5 |
| sourceforge |          4 |
| ow          |          4 |
| over-blog   |          4 |
| woothemes   |          3 |
| virginia    |          3 |
| deviantart  |          3 |
| tripadvisor |          3 |
| weebly      |          3 |
+-------------+------------+
10 rows in set (0.001 sec)
```

Figure 2 – Information extracted directly from the customer entity

Album

The count of albums is shown in figure 3. Also, the number of genres appearing in albums and the total retail price of all albums in the current inventory are shown below.

```
MariaDB [musicdb]> select count(*) as AlbumCount from album;
+------------+
| AlbumCount |
+------------+
|        134 |
+------------+
1 row in set (0.000 sec)
```

```
MariaDB [musicdb]> select genre, count(*) as occurances from album group by genre order by occurances desc;
+------------+------------+
| genre      | occurances |
+------------+------------+
| Indie Rock |         24 |
| Rock       |         21 |
| Electro    |         20 |
| Country    |         18 |
| Pop        |         17 |
| Rhythm and |         14 |
| Techno     |         12 |
| EDM        |          8 |
+------------+------------+
8 rows in set (0.001 sec)
```

```
MariaDB [musicdb]> select sum(stockAmount) as sumOfStock, sum(albumPrice) as sumOfPrice,
    ->  concat('$ ', sum(albumPrice)*sum(stockAmount)) as totalPriceInStock from album;
+------------+------------+--------------------+
| sumOfStock | sumOfPrice | totalPriceInStock  |
+------------+------------+--------------------+
|        459 |    3382.75 | $ 1552682.25       |
+------------+------------+--------------------+
1 row in set (0.000 sec)
```

Figure 3 - Information about the albums

Track

The figure below shows the count of tracks in database, number of tracks in the top 10 albums and the average track length is computed in minutes.

```
MariaDB [musicdb]> select count(*) as TrackCount from track;
+------------+
| TrackCount |
+------------+
|        804 |
+------------+
1 row in set (0.000 sec)
```

```
MariaDB [musicdb]> select a.albumname, count(t.trackid) as numberOfTracks
    -> from track t join album a
    -> on t.catalogueno = a.catalogueno
    -> group by a.albumname
    -> order by numberOfTracks desc limit 10;
+-----------------------------------------+-----------------+
| albumname                               | numberOfTracks  |
+-----------------------------------------+-----------------+
| synthesize B2B ROI                      |              13 |
| expedite B2C experiences                |              13 |
| utilize 24/365 web-readiness            |              11 |
| recontextualize synergistic communities |              11 |
| monetize synergistic infrastructures    |              11 |
| innovate frictionless content           |              10 |
| reinvent 24/365 networks                |              10 |
| integrate vertical architectures        |              10 |
| monetize out-of-the-box infrastructures |              10 |
| harness 24/365 e-commerce               |              10 |
+-----------------------------------------+-----------------+
10 rows in set (0.002 sec)
```

```
MariaDB [musicdb]> select concat(AVG(trackLength), ' min') as AverageLengthOfTracks from track;
+----------------------+
| AverageLengthOfTracks |
+----------------------+
| 3.292624 min         |
+----------------------+
1 row in set (0.001 sec)
```

Figure 4 - Information about the tracks in database

Artist

The number of artists credited in the albums sold in the music store, the number of artists associated with a record label, and the gender proportion are shown in Figure 5.

```
MariaDB [musicdb]> select count(*) as ArtistCount from artist;
+-------------+
| ArtistCount |
+-------------+
|         256 |
+-------------+
1 row in set (0.001 sec)
```

```
MariaDB [musicdb]> select recordlabel, count(*) as ArtistCount from artist group by recordlabel
    -> order by artistcount desc;
+-------------------------------+-------------+
| recordlabel                   | ArtistCount |
+-------------------------------+-------------+
| Warner Music Group            |          34 |
| virgin Records                |          30 |
| ABC-Paramount Records         |          28 |
| BMG Rights Management         |          28 |
| island Records                |          25 |
| Sony Music Entertainment      |          25 |
| Universal Music Publishing Gro |          25 |
| Def lam Recordings            |          23 |
| Atlantic Records              |          22 |
| Red Hill Records              |          16 |
+-------------------------------+-------------+
10 rows in set (0.001 sec)
```

```
MariaDB [musicdb]> select gender, count(*) as ArtistCount from artist group by gender;
+--------+-------------+
| gender | ArtistCount |
+--------+-------------+
| Female |         124 |
| Male   |         132 |
+--------+-------------+
2 rows in set (0.001 sec)
```

Figure 5 - information about the artist

Orders

The following figure show the amount of purchases for the top 10 albums and their names (top query) and the number of purchases for each month in the music store.

```
MariaDB [musicdb]> select o.catalogueNo, a.albumName, count(o.catalogueNo) as purchases
    -> from orderline o join album a
    -> on a.catalogueNo = o.catalogueNo
    -> group by o.catalogueNo
    -> order by purchases desc limit 10;
+-------------+-----------------------------------------+-----------+
| catalogueNo | albumName                               | purchases |
+-------------+-----------------------------------------+-----------+
|          45 | monetize back-end solutions             |        18 |
|          81 | enable distributed vortals              |        16 |
|          23 | unleash value-added vortals             |        14 |
|          70 | enhance integrated interfaces           |        14 |
|         102 | revolutionize extensible deliverables   |        13 |
|          84 | transform interactive mindshare         |        12 |
|         100 | incentivize bleeding-edge methodologies |        12 |
|          35 | monetize out-of-the-box infrastructures |        11 |
|          54 | enhance B2C ROI                         |        11 |
|         124 | harness rich bandwidth                  |        11 |
+-------------+-----------------------------------------+-----------+
10 rows in set (0.001 sec)
```

```
MariaDB [musicdb]> select monthname(dateOfPurchase) as monthName,
    -> Month(dateOfPurchase) as month, count(catalogueNo) as countOfPurchase
    ->  from orderline
    -> group by month
    -> order by countOfPurchase desc;
+-----------+-------+-----------------+
| monthName | month | countOfPurchase |
+-----------+-------+-----------------+
| April     |     4 |             111 |
| August    |     8 |             104 |
| May       |     5 |             100 |
| January   |     1 |              95 |
| March     |     3 |              90 |
| June      |     6 |              88 |
| October   |    10 |              87 |
| July      |     7 |              86 |
| February  |     2 |              83 |
| September |     9 |              80 |
| December  |    12 |              41 |
+-----------+-------+-----------------+
11 rows in set (0.001 sec)
```

Figure 6 - queries concerning orders in music store

Use Case Queries

The following are queries that include join statements. Descriptions are given in the figure caption. Each query attempts to respond to the relevant matter.

```
MariaDB [musicdb]> select  c.customerID, c.firstName, c.lastName, CONCAT('$ ', p.totalPrice) as totalSpe
    -> from customers c join purchasedetails p
    -> on c.customerID = p.customerID
    -> group by c.customerID
    -> order by p.totalPrice desc limit 10;
```

customerID	firstName	lastName	totalSpend
330	Judon	Lonergan	$ 199.63
194	Garey	Rex	$ 199.56
62	Lucina	Coraini	$ 198.68
58	Debra	Rishman	$ 196.39
108	Yancy	Gammidge	$ 194.98
57	Dallas	McCurdy	$ 194.79
349	Nickey	Ewence	$ 194.71
17	Queenie	Kopacek	$ 194.23
200	Jemmy	Stanbridge	$ 193.65
163	Ninette	Kingsford	$ 190.74

10 rows in set (0.002 sec)

Figure 7 – This query seeks the top 10 customers that spent the most at the music store.

```
MariaDB [musicdb]> select monthname(l.dateOfPurchase) as monthName,month(l.dateOfPurchase) as month,
    -> year(l.dateOfPurchase) as year, CONCAT('$ ',SUM(p.totalPrice)) as revenue
    -> from purchasedetails p join orderline l
    -> on p.recieptNo = l.recieptNo
    -> group by month, year
    -> order by sum(p.totalPrice) desc;
```

monthName	month	year	revenue
April	4	2020	$ 12245.39
August	8	2020	$ 11467.61
May	5	2020	$ 10995.67
June	6	2020	$ 10628.15
January	1	2020	$ 9911.54
October	10	2020	$ 9580.20
March	3	2020	$ 9398.02
July	7	2020	$ 8891.87
September	9	2020	$ 8482.22
February	2	2020	$ 7874.37
December	12	2019	$ 4539.74

11 rows in set (0.002 sec)

Figure 8 – This query retrieves the revenue earnings monthly at the music store.

```
MariaDB [musicdb]> select a.albumName, CONCAT(SUM(t.trackLength), " mins") as albumLength
    -> from album a join track t
    -> on a.catalogueNo = t.catalogueNo
    -> group by a.albumName
    -> order by SUM(t.trackLength) desc limit 10;
```

albumName	albumLength
synthesize distributed bandwidth	46.81 mins
exploit seamless vortals	42.97 mins
drive global models	39.67 mins
exploit best-of-breed platforms	37.25 mins
redefine revolutionary e-business	37.11 mins
envisioneer sexy vortals	36.67 mins
target extensible markets	36.26 mins
target efficient infrastructures	34.79 mins
orchestrate enterprise e-markets	34.09 mins
enable open-source interfaces	32.81 mins

10 rows in set (0.004 sec)

Figure 9 - The top 10 albums with the longest duration of play time in minutes.

```
MariaDB [musicdb]> select a.albumName, t.trackName, CONCAT(t.trackLength, ' min') as maxLength
    -> from album a join track t
    -> on a.catalogueNo = t.catalogueNo
    -> where t.trackLength =
    -> (select MAX(trackLength)
    -> from track
    -> where catalogueNo = a.catalogueNo);
```

albumName	trackName	maxLength
recontextualize virtual communities	Sonoran Bean	5.35 min
transform sticky networks	Kern County Milkvetc	5.38 min
strategize killer networks	Lecidea Lichen	5.35 min
strategize front-end vortals	Parry's Clover	5.37 min
mesh sticky channels	Bactris Palm	5.32 min
empower out-of-the-box experiences	Johnny-nip	5.09 min
facilitate cutting-edge convergence	Cow Itch Tree	5.13 min
cultivate real-time networks	Piedmont Meadow-rue	4.59 min
innovate magnetic channels	Caribbean Mayten	5.44 min
aggregate collaborative architectures	Hawai'i Ticktrefoil	5.39 min
disintermediate bleeding-edge vortals	Bergamot Orange	5.03 min
engineer 24/7 ROI	Dwarf Western Rosinw	4.16 min
benchmark web-enabled infrastructures	Slender Cinquefoil	5.33 min
monetize bricks-and-clicks niches	European Fan Palm	4.50 min
deliver sticky methodologies	Hirsute Sedge	5.23 min
visualize sexy e-commerce	Eastern Fox Sedge	3.34 min
drive holistic infrastructures	Youth On Age	4.38 min
transition rich bandwidth	Whitehair Rosette Gr	5.33 min
orchestrate sexy niches	Mung Bean	5.17 min
morph real-time paradigms	Pereskia	3.38 min
embrace out-of-the-box ROI	Whipple Cholla	4.38 min
embrace world-class e-business	Allegheny Stonecrop	4.54 min
evolve distributed models	Kauai Bloodgrass	5.23 min
syndicate robust web-readiness	Pelicanflower	4.44 min
envisioneer sexy vortals	Longbeard Mariposa L	5.20 min
e-enable 24/7 niches	Adderstongue	4.41 min
optimize transparent architectures	Strawberryleaf Cinqu	5.44 min
incubate innovative relationships	Appalachian Fissiden	5.40 min
exploit seamless vortals	Grass Of Parnassus	5.38 min
integrate impactful models	Whiteflower Passionf	5.22 min
orchestrate enterprise e-markets	Mock Fanpetals	4.57 min

Figure 10 - Extraction of the longest song in each album.

```
MariaDB [musicdb]> select ar.recordLabel, CONCAT('$ ', SUM(p.totalPrice)) as revenue
    -> from artist ar join artistcredits ac
    -> on ar.artistID = ac.artistID
    -> join track t
    -> on ac.trackID = t.trackID
    -> join album a
    -> on t.catalogueNo = a.catalogueNo
    -> join orderline ol
    -> on a.catalogueNo = ol.catalogueNo
    -> join purchasedetails p
    -> on ol.recieptNo = p.recieptNo
    -> group by ar.recordLabel
    -> order by SUM(p.totalPrice) desc;
```

recordLabel	revenue
Warner Music Group	$ 105710.75
Universal Music Publishing Gro	$ 90241.44
BMG Rights Management	$ 90078.11
ABC-Paramount Records	$ 86074.58
island Records	$ 81232.90
virgin Records	$ 72731.64
Atlantic Records	$ 71077.65
Sony Music Entertainment	$ 63015.79
Def lam Recordings	$ 59837.26
Red Hill Records	$ 46497.87

10 rows in set (0.024 sec)

Figure 11 - Revenue gained from the music store for each record label.

Alternative Storage of the Data

A NoSQL database can used to store the music database. The difference is that a schema will not be enforced. This is observed in the datatype, 'null/not null' and PK/FK allocation for each attribute. In NoSQL, this structure is not required. In the event of employing NoSQL database for the music store, it is suited that database operates on large volumes of unstructured data. Flexibility and leniency in the collection of data can improve elaborate analysis and fast retrieval of data can lead to efficient computation.

However, this music store collects the information from centralised system that records customer, order and album related data. It would be effective to use a Relational Database in this case. This can be attributed to the size of the business and maintenance of the inputted data. This type of business is rare because most music is internet-based, hence the data will not be in excess. The maturity of Relational Databases can impose strictness to the characteristics of the customer, orders, artists, albums and its features. This will develop effective and direct analysis of the permanent database.

Conclusion

Throughout the design report, the design and use cases of the music database was discussed. Such use cases were elaborated through the implementation of SQL queries. The process and interaction of the entities within the music database was explored. In addition, the reasoning behind the selection of data types in the schemas was provided. An alternative method was proposed and the justification of selecting Relational database to convey the businesses data were also conveyed.

References

Chackfield, M., 2014. *12 Album Tracks Released On 12 Different Formats.* [Online]
Available at: https://www.shortlist.com/news/12-album-tracks-released-on-12-different-formats
[Accessed 20 October 2020].

Gibbs, T., 2019. *Five Common Data Stores and When to Use Them.* [Online]
Available at: https://shopify.engineering/five-common-data-stores-usage
[Accessed 20 October 2020].

Mcdonald, H., 2019. *Catalog Numbers for CDs.* [Online]
Available at: https://www.thebalancecareers.com/catalog-number-2460354
[Accessed 20 October 2020].

Mockaroo, 2020. *Realistic Data Generator.* [Online]
Available at: https://mockaroo.com/
[Accessed 1 November 2020].

Statista, 2018. *Music album consumption U.S. 2018, by genre.* [Online]
Available at: https://www.statista.com/statistics/310746/share-music-album-sales-us-genre/
[Accessed 20 October 2020].