**SOLENT UNIVERSITY**

Databases (COM711)

**Report 1**

**Student Name:** Ajaz *Ali Saiyed*

**Contents page**

Part 1- Retrieving Data using SQL

Introduction 3

Question a 3

Question b 5

Question c 8

Part 2 – Database Design, Implementation and Integrity

Question a 12

Question b 12

Question c 13

Question d 16

Part 3 – Programming for Databases 17

Report

Evaluation 25

Research 27

Bibliography and References 31

**Introduction**

This task manages down to earth information base application advancement work and answer the inquiries identified with the data set activities just as present a report that prescribes improvement to the given data set and examination a point on the equivalent.

In this task, I will SQLite3, which comes prepared in Python for composing and executing inquiries. The IDE I'll be utilizing is PyCharm Community 2020.2 form. Utilizing the accompanying instruments, going to respond to all the inquiries of the task.

**PART 1 -**

1. **Solution:**

**SQL CODE:**

SELECT shopper\_first\_name, shopper\_surname, shopper\_email\_address, gender,

STRFTIME(‘%d-%m-%Y’, date\_joined) AS date

STRFTIME(‘%Y’,’now’) – STRFTIME(‘%Y’, date\_of\_birth) AS current\_age

FROM shoppers WHERE ((gender = ‘F’) + (gender = ‘M’ AND date >= ‘2020-01-01’))

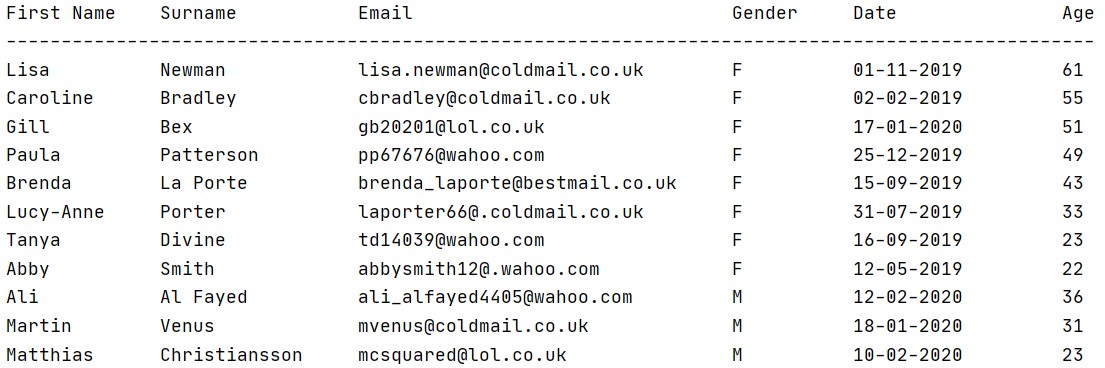
**EXPLANATION:**

I utilized the content from SOL stage to make the information base solent.db To utilize this information base to respond to questions, I imported sqlite3 library in python.

1. The clarification of the inquiry is shown underneath:**COALESCE(shopper\_first\_name,'Not Known'):** Using this statement I selected the column‘shopper\_first\_name’ column from the table ‘shoppers’. The COALESCE clause helps replace any NULL values with ‘Not Known’.
2. **COALESCE(shopper\_surname,'Not Known'):** Using this statement I selected the column‘shopper\_surname’ column from the table ‘shoppers’. The COALESCE clause helps replace any NULL values with ‘Not Known’.
3. **COALESCE(gender,'Not Known'):** Using this statement I selected the column‘gender’ column from the table ‘shoppers’. The COALESCE clause helps replace any NULL values with ‘Not Known’.
4. **STRFTIME(‘%d-%m-%Y’, date\_joined):** This statement SELECTs ‘date\_joined’ column from the table ‘shoppers’. The STRFTIME clause helps change the date formatting.
5. **(STRFTIME('%Y', 'now') - STRFTIME('%Y', date\_of\_birth)) AS age:** This statement helps calculate the shopper’s Age and in this case I have stored it in a variable named ‘age’. In the next step, created a connection with the ‘Orinoco Database’.
6. **WHERE ((gender = 'F') + (gender = 'M' AND date\_joined>= '2020-01-01')):** This statement helps filter out the results by using the WHERE clause.
7. **ORDER by gender, age DESC:** This statement sorts the results by gender first and then by age in the descending order viz the highest age is displayed first.

To print the results, I used for-loop to iterate over ‘lst’. Check the screenshot below for output.

**SCREENSHOT:**



**b) Solution:**

**SQL CODE:**

SELECTshopper\_first\_name,

       shopper\_surname,

       shopper\_orders.order\_id,

       STRFTIME('%d-%m-%Y',order\_date),

       price,

       quantity,

       ordered\_product\_status,

       seller\_name,

       product\_description

       FROM shoppers

       INNER JOIN shopper\_orders

       ON shoppers.shopper\_id = shopper\_orders.shopper\_id

       INNER JOIN ordered\_products

ON shopper\_orders.order\_id = ordered\_products.order\_id

INNER JOIN sellers

ON ordered\_products.seller\_id = sellers.seller\_id

INNER JOIN products

ON products.product\_id = ordered\_products.product\_id

WHERE shoppers.shopper\_id = ?

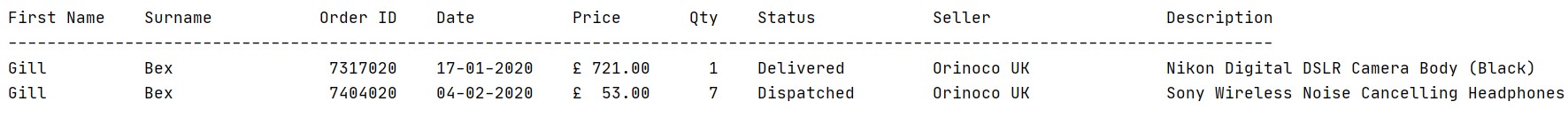
ORDER BY order\_date ASC

**EXPLANATION:**

In this question, the SQL script contains following statements listed below.

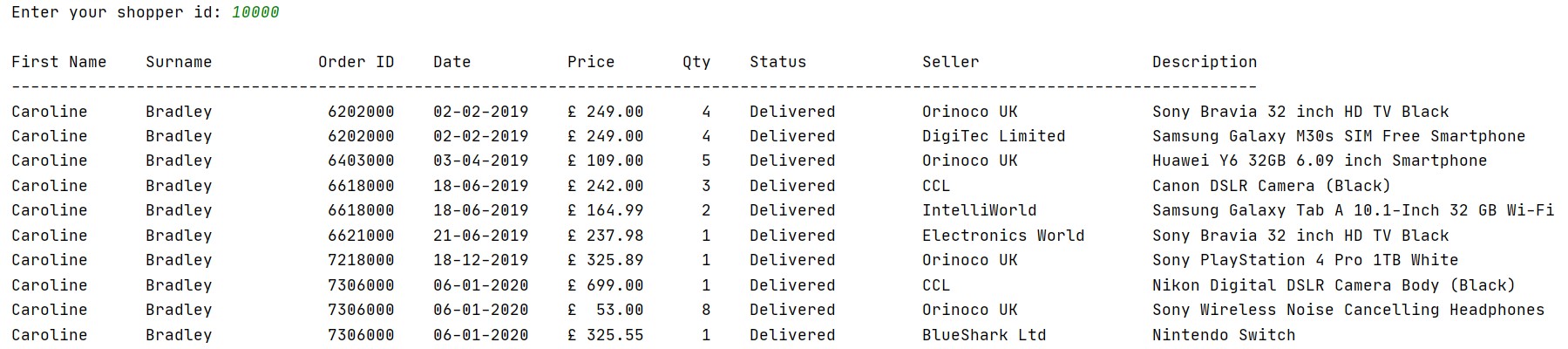
1. **SELECT** *‘column names’***:** This statement SELECTs the required columns from the database. The STRFTIME clause helps change the date formatting for the order date column.
2. **INNER JOIN shopper\_orders:** This statement joins the table named ‘shopper\_orders’ with the table ‘shoppers’ which enables the user to select from both the tables. Similarly, in other steps, more tables are joined.
3. **INNER JOIN ordered\_products:** This statement joins the table named ‘ordered\_products’ with the table ‘shopper\_orders’
4. **INNER JOIN sellers:** This statement joins the table named ‘sellers’ with the table ‘ordered\_products’
5. **INNER JOIN products:** This statement joins the table named ‘products’ with the table ‘sellers’
6. **WHERE shoppers.shopper\_id = ? :** This statement uses the WHERE clause to filter out the results from the data using the shopper\_id value as an input from user.
7. **ORDER BY order\_date ASC:** This statement orders the data by order\_date in ascending order
8. To print the results, I used for-loop to iterate over ‘lst’. Check the screenshot below for output.

**SCREENSHOT:**

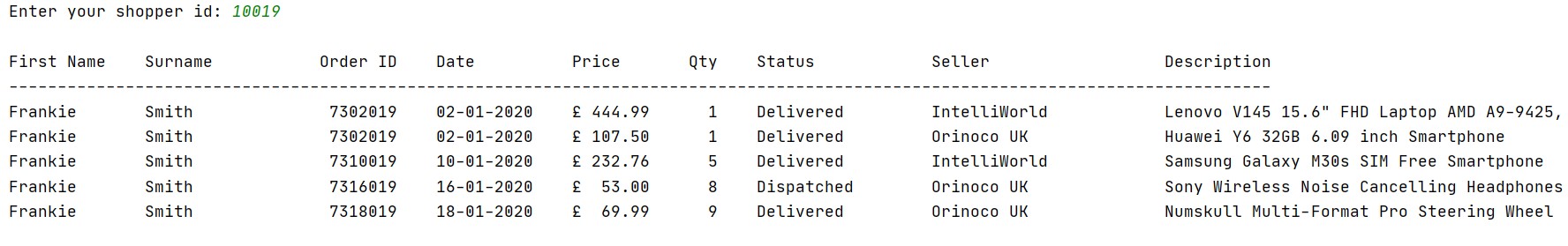


**TESTING:**

Testing the required results on shopper id = 10000



Testing the required results when shopper id = 10019



**c) Solution:**

**SQL CODE:**

       Query =     SELECT

                STRFTIME('%d-%m-%Y',order\_date),

                price,

                quantity,

                price \* quantity AS sales,

                product\_code,

                seller\_name,

                seller\_account\_ref,

                product\_description

                FROM ordered\_products op

                    INNER JOIN shopper\_orders so

                        ON so.order\_id = op.order\_id

                    INNER JOIN sellers s

                        ON op.seller\_id = s.seller\_id

                    INNER JOIN products p

                        ON p.product\_id = op.product\_id

                WHERE order\_date > '2019-06-01'

                ORDER BY seller\_name, product\_description

Total\_price\_quantity = """

                SELECT

                SUM(price \* quantity),

                SUM(quantity)

                FROM ordered\_products op

                    INNER JOIN shopper\_orders so

                        ON so.order\_id = op.order\_id

                    INNER JOIN sellers s

                        ON op.seller\_id = s.seller\_id

                    INNER JOIN products p

                        ON p.product\_id = op.product\_id

                WHERE order\_date > '2019-06-01'

                ORDER BY seller\_name, product\_description

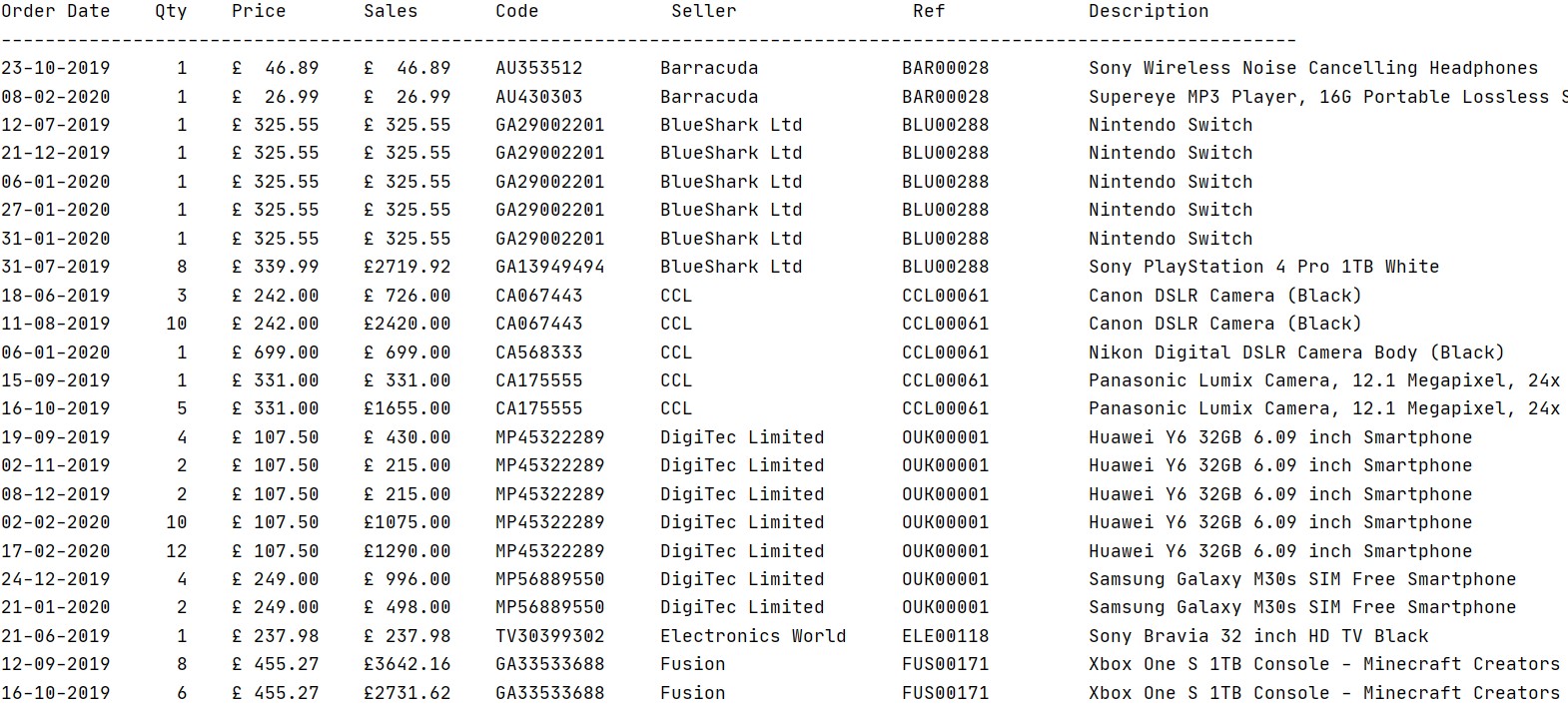
            """

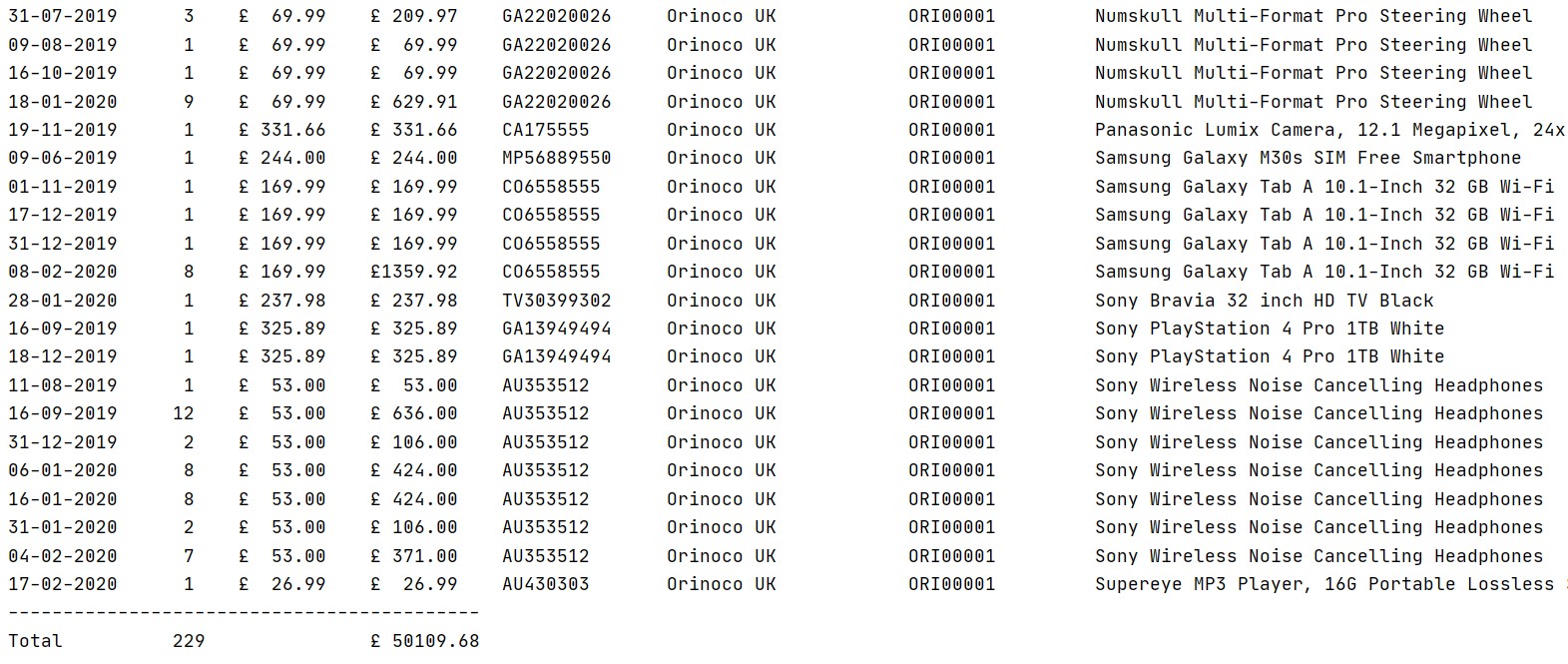
**EXPLANATION:**

The SQL code for this contains following script.

1. **SELECT** *‘column names’***:** This statement SELECTs the required columns, for this question, which are, seller account ref, sellername, product code, product description, number of orders, total quantity sold and total value of all sales from the database. The STRFTIME clause helps change the date formatting for the order date column.
2. **INNER JOIN shopper\_orders:** This statement joins the table named ‘shopper\_orders’ with the table ‘ordered\_products’ which enables the user to select from both the tables. Similarly, in other steps, more tables are joined.
3. **INNER JOIN sellers:** This statement joins the table named ‘shopper\_orders’ with the table ‘sellers’ which enables the user to select from both the tables. Similarly, in other steps, more tables are joined.
4. **INNER JOIN products:** This statement joins the table named ‘products’ with the table ‘sellers’ which enables the user to select from both the tables. Similarly, in other steps, more tables are joined.

**SCREENSHOT:**





**PART 2**

1. **Solution:**

<Describe your thought process in coming up with your design preferably using the threestage design process – conceptual, logical and physical designs>

<State any design assumptions you have made in coming up with this design>

1. **Solution:**

<Either include an image showing your amended Entity Relationship Diagram or alternatively submit this diagram as a separate file>



1. **Solution:**

<Include here the SQL code to create your new tables as per your design>

<Include here the SQL code to insert some test data (min. of 5 rows) into each of your new tables>

<Include here the SQL code and screenshots showing evidence of testing the unique, foreign key and check constraints you have created in each of your new tables>

import sqlite3

connection = sqlite3.connect('solent.db')

cursor = connection.cursor()

prod\_review\_query = """

                        DROP TABLE IF EXISTS product\_reviews;

                        CREATE TABLE product\_reviews(

                            product\_reviews\_id INTEGER PRIMARY KEY,

                            product\_id INTEGER,

                            poor INTEGER RANGE(0,1),

                            fair INTEGER RANGE(1,2),

                            good INTEGER (2,3),

                            very\_good INTEGER (3,4),

                            excellent INTEGER (4,5),

                            product\_review\_date DATETIME,

                            COMMENTS TEXT,

                            FOREIGN KEY (product\_id)

                            REFERENCES products(product\_id))

                    """

cursor.executescript(prod\_review\_query)

prod\_insert\_query = """

                        INSERT INTO product\_reviews (product\_reviews\_id, product\_id, poor, fair, good, very\_good, excellent, product\_review\_date, Comments)

                        VALUES (1, 3007762, 0, 2, 3, 3, 5, datetime('now'), 'Great Product');

                        INSERT INTO product\_reviews (product\_reviews\_id, product\_id, poor, fair, good, very\_good, excellent, product\_review\_date, Comments)

                        VALUES (2, 3000000, 0, 1, 3, 3, 5, datetime('now'), 'Worth Money');

                        INSERT INTO product\_reviews (product\_reviews\_id, product\_id, poor, fair, good, very\_good, excellent, product\_review\_date, Comments)

                        VALUES (3, 3000021, 0, 2, 3, 3, 4, datetime('now'), 'Best buy till date');

                        INSERT INTO product\_reviews (product\_reviews\_id, product\_id, poor, fair, good, very\_good, excellent, product\_review\_date, Comments)

                        VALUES (4, 3005955, 0, 1, 3, 3, 4, datetime('now'), 'A bit laggy');

                        INSERT INTO product\_reviews (product\_reviews\_id, product\_id, poor, fair, good, very\_good, excellent, product\_review\_date, Comments)

                        VALUES (5, 3006033, 0, 2, 2, 4, 5, datetime('now'), 'Not as expected');

                    """

cursor.executescript(prod\_insert\_query)

connection.commit()

sr\_query = """

                DROP TABLE IF EXISTS seller\_reviews;

                CREATE TABLE seller\_reviews(

                    seller\_reviews\_id INTEGER PRIMARY KEY,

                    seller\_id INTEGER,

                    poor INTEGER RANGE(0,1),

                    fair INTEGER RANGE(1,2),

                    good INTEGER (2,3),

                    very\_good INTEGER (3,4),

                    excellent INTEGER (4,5),

                    seller\_review\_date DATETIME,

                    Question TEXT,

                    FOREIGN KEY (seller\_id)

                    REFERENCES sellers(seller\_id))

            """

cursor.executescript(sr\_query)

seller\_insert\_query = """

                        INSERT INTO seller\_reviews (seller\_reviews\_id, seller\_id, poor, fair, good, very\_good, excellent, seller\_review\_date, Question)

                        VALUES (1, 200001, 0, 1, 3, 3, 5, datetime('now'), 'Any good?');

                        INSERT INTO seller\_reviews (seller\_reviews\_id, seller\_id, poor, fair, good, very\_good, excellent, seller\_review\_date, Question)

                        VALUES (2, 200002, 0, 2, 3, 3, 4, datetime('now'), 'On time service?');

                        INSERT INTO seller\_reviews (seller\_reviews\_id, seller\_id, poor, fair, good, very\_good, excellent, seller\_review\_date, Question)

                        VALUES (3, 200003, 0, 1, 3, 3, 4, datetime('now'), 'Expected delivery date?');

                        INSERT INTO seller\_reviews (seller\_reviews\_id, seller\_id, poor, fair, good, very\_good, excellent, seller\_review\_date, Question)

                        VALUES (4, 200004, 0, 2, 2, 4, 5, datetime('now'), 'Would you recommend it?');

                        INSERT INTO seller\_reviews (seller\_reviews\_id, seller\_id, poor, fair, good, very\_good, excellent, seller\_review\_date, Question)

                        VALUES (5, 200004, 0, 2, 2, 4, 5, datetime('now'), 'When can I expect new version to arrive?');

                     """

cursor.executescript(seller\_insert\_query)

connection.commit()

connection.close()

1. **Solution:**

<Include here the SQL code used to create your view(s)?

<Include here the SQL code and screenshots of your queries that select data from the view(s) and join the view to other tables>

import sqlite3

connection = sqlite3.connect('solent.db')

cursor = connection.cursor()

query = """

            DROP VIEW IF EXISTS details;

            CREATE VIEW details

            AS SELECT a.product\_id, b.seller\_id, a.fair, b.fair

            FROM product\_reviews a, seller\_reviews b;

        """

cursor.executescript(query)

connection.commit()

query\_two = "SELECT \* FROM details WHERE fair = 1"

cursor.execute(query\_two)

query\_three = "SELECT \* FROM details WHERE fair = 2"

cursor.execute(query\_three)

connection.close()

**PART 3**

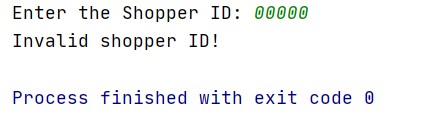
In this aspect of the task, I utilized squares of capacities to manufacture an order line application. The menu comprises of 5 choices.

1. Display your order history.
2. Add an item to basket.
3. View your basket
4. Checkout
5. Exit

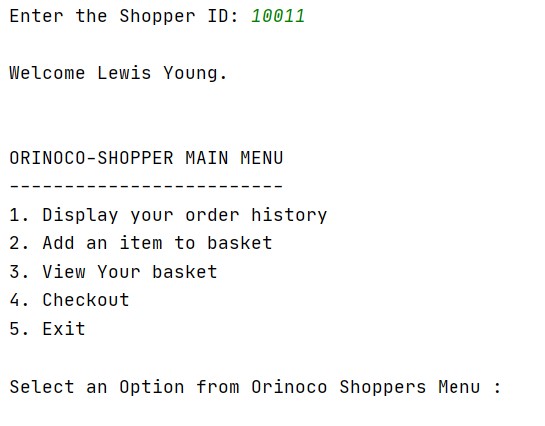
When the program is executed, the user is asked to input shopper id. Check the screenshot below.



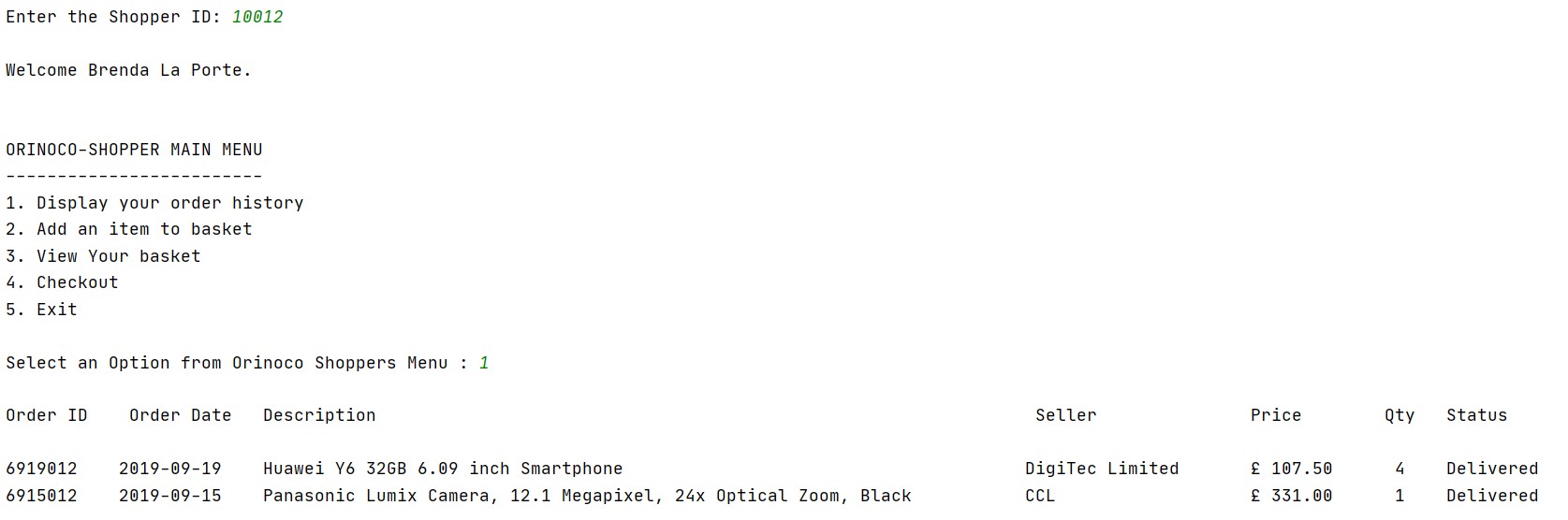
If the id entered by the user is incorrect, an error message is printed as shown below.



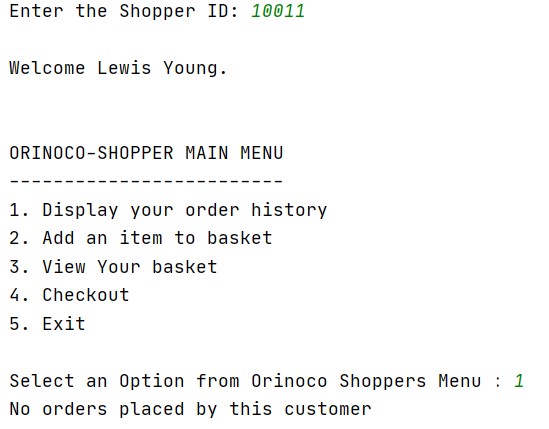
And if the correct id is entered, then the shopper menu is placed.



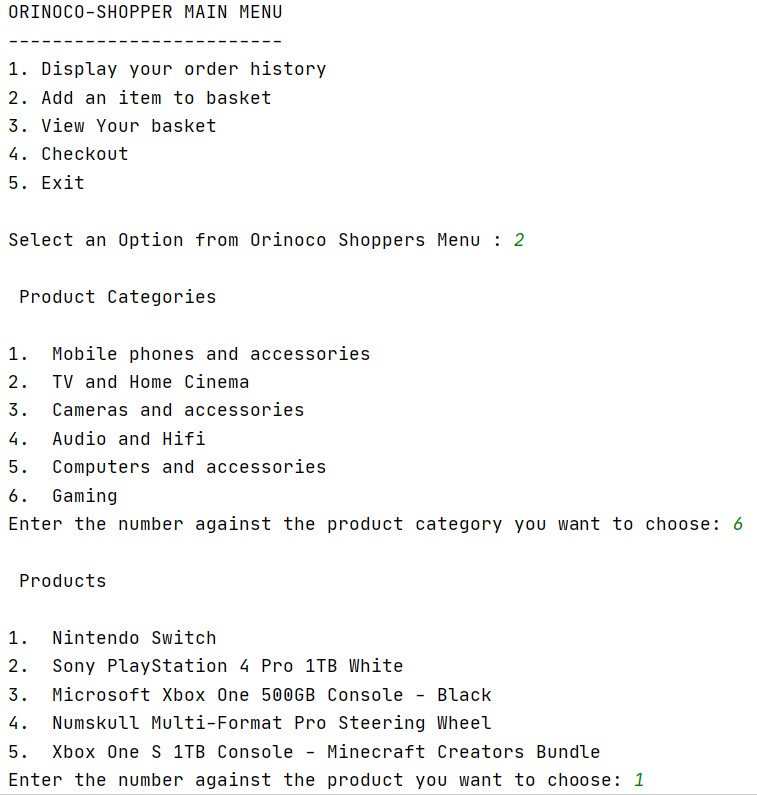
While the input = 1, order history is displayed.

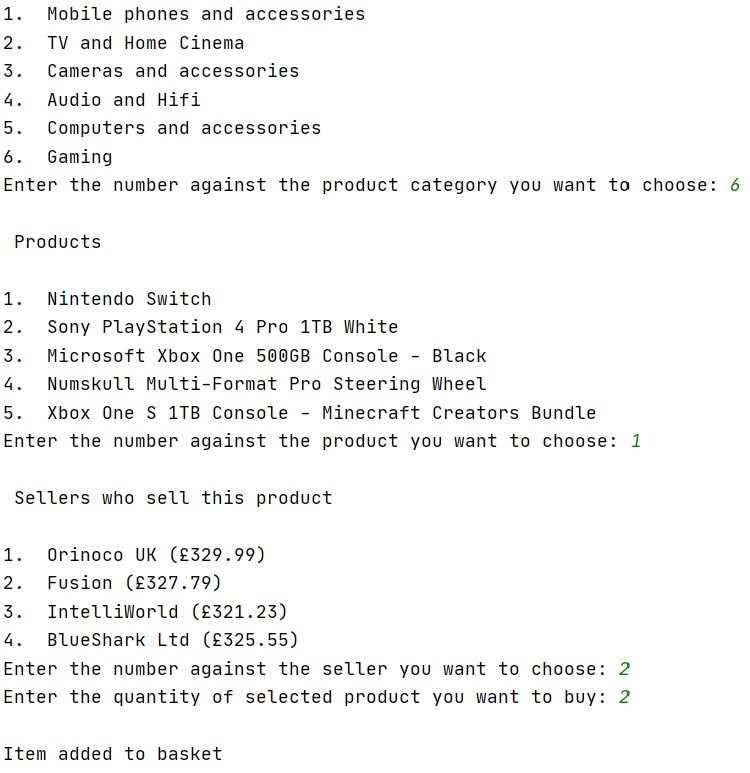


If the order history is empty, a message saying ‘No orders placed by this customer’ is displayed.



In the event that the information is 2, the client is given a category menu. This category menu empowers a client pick items from the menu and add them to the basket.

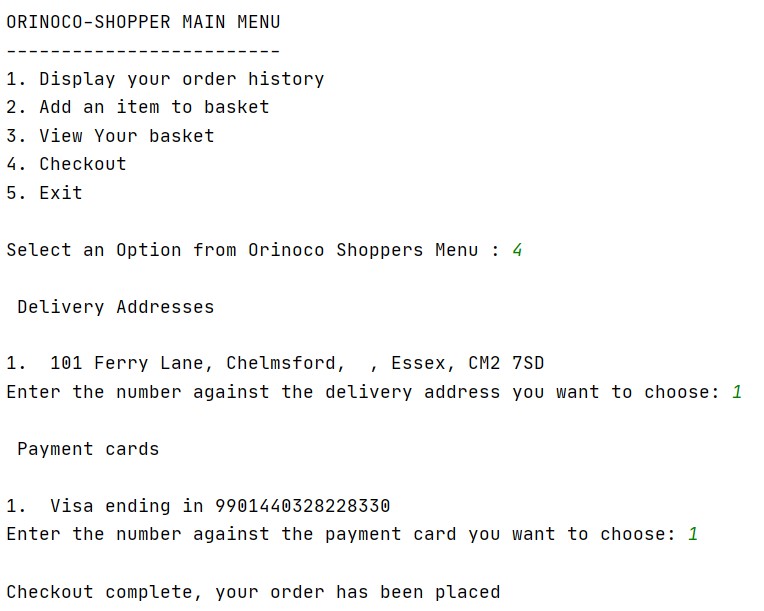




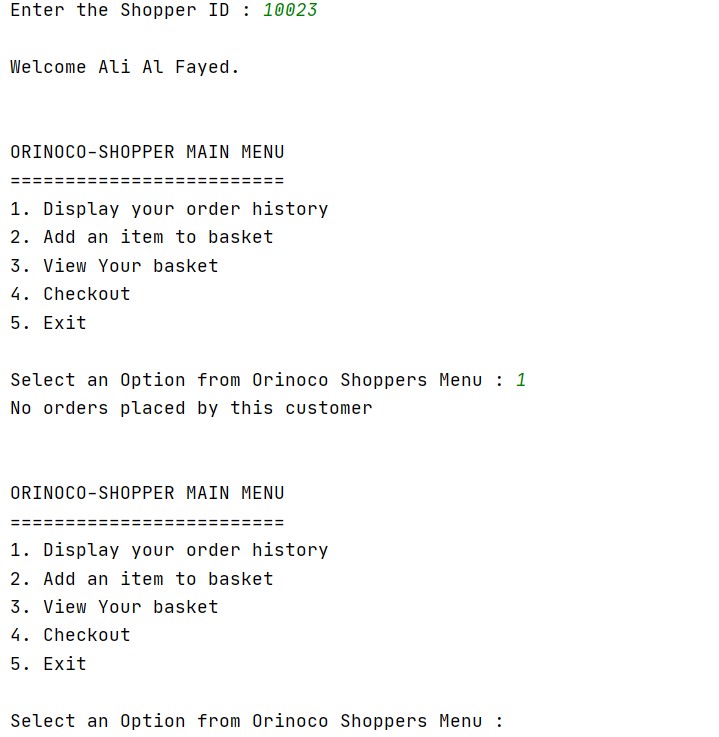
Item is added to the basket once the user has selected the items. And the user selected products can be seen by using Option 3 of the shopper menu as shown below

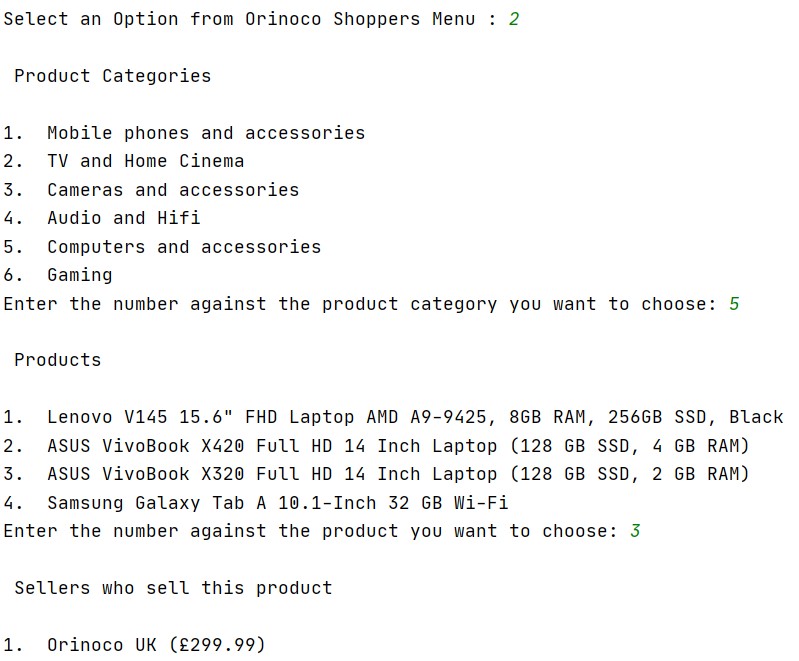


To confirm payment, the user selects option 4, which displays the menu to select address from the given list and payment cards as shown below.

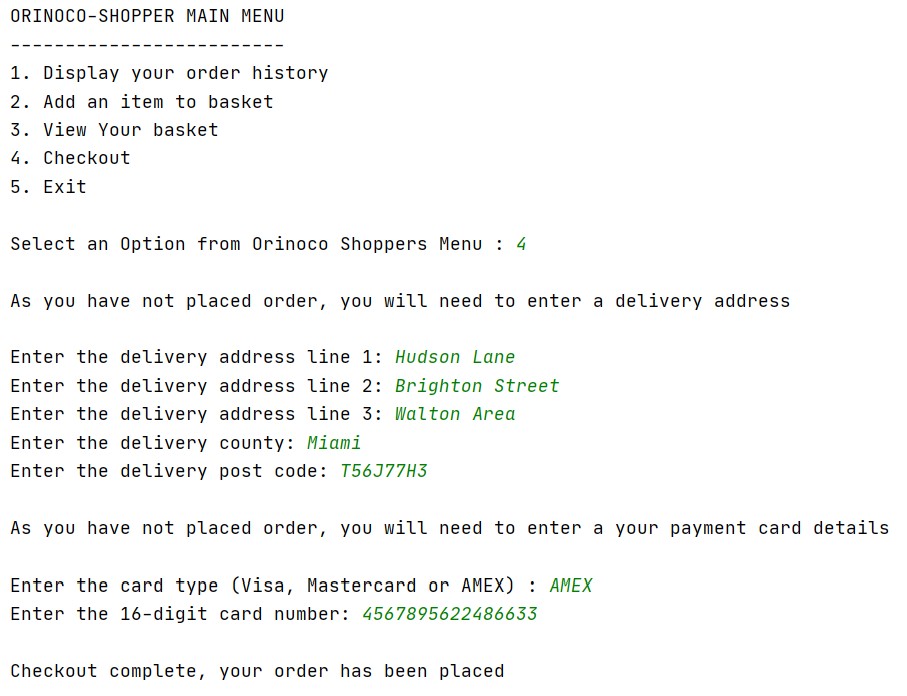


If a shopper has no address stored, then the shopper will be prompted to add address. One such example is the shopper with shopper id = 10023. Check screenshots below for whole process.









The Option 5 Exits the program.



**Evaluation and Research Report**

Evaluation

<Include here an evaluation of the quality of the work you have produced with reference to both the specified requirements and the assessment criteria. You should include:

* any deliverables that are not complete or fully tested;
* elements of the practical work that you feel contribute to higher grade achievement;
* your approach to design and testing of the SQL and Python elements of the assessment;
* how well you feel you managed your time between the four parts of the assessment and what more you would have done if you’d had more time.

>

This task covers the answers for practically all the inquiries in the task aside from the Part-2(a) is halfway secured, while Part-1(d) isn't secured because of time fumble. Then again, actually each of the three sub-portions of Part-1 and Part-2 have been done precisely and tried. Section 3 has been done completely and all the focuses referenced have been secured. The information base was made utilizing the content from SOL. Here I named it 'solent.db'. The Entity-Relationship Diagram was likewise gotten from SOL, which I used to alter and include new tables.

**High scoring elements:**

According to my perceptions, High scoring components may include:

Section 1: Use of provisions, for example, SUM, AVG, COUNT, STRFTIME and advance provisos like DATEDIFF to control information and putting away them in new segments, utilizing numerous JOINS alongside UNION condition to compose complex inquiries add to higher evaluation accomplishment.

Section 2: In Part-2, I feel making tables and utilizing them to compose genuine questions which comprehend the unpredictability of the information base plan add to higher evaluations.

Section 3: Collecting the client input and including the information into real data set by embeddings lines adds to higher evaluations. Another alternative that could be added to the order line application is including new customer if the customer doesn't exist. The key Python components which add to higher evaluation accomplishment incorporate exemption taking care of.

**Designing SQL and Python elements:**

1. In SQL elements, I used several ‘CLAUSES’ like SUM, AVG, COUNT, STRFTIME etc. to answer the questions in Part-1 of the assignment. In Part-2, designed tables named product\_reviews, seller\_reviews and ‘dealers’,also created a view named ‘details’ and tested it for two queries.
2. In Part-2 of the assignment, in extension to the already given database, I have created three new tables named product\_reviews, seller\_reviews and retailers.
   1. Product\_reviews: This table uses ‘product\_reiews\_id’ as the primary key and ‘product\_id’ as the foreign key which references table ‘products’. Other keys include poor, fair, good, very good, excellent, product\_review\_date and QUESTION.
   2. Seller\_reviews: This table uses ‘seller\_review\_id’ as a primary key and ‘seller\_id’ as a foreign key which references to ‘seller\_id’ from the table ‘sellers’. Other keys include poor, fair, good, very good, excellent, seller\_review\_date and Comment.
   3. Dealers: This table uses ‘dealer\_id’ as the primary key and three foreign keys viz product\_id, product\_code and product\_description.
3. In Part-3 of the assignment, besides the given function ‘\_display\_options(all\_options, title, type)’, I developed my own custom functions which are:
   1. Shopper\_check(): checks weather a shopper is present in a database or not.
   2. Query\_rows(): stores ‘category\_id’ and ‘category\_description’
   3. Product\_id(): displays menu for ‘product description’ while storing the ‘product\_id’ against the option.
   4. seller\_id(): displays menu for ‘sellers’ while storing the ‘seller\_id’ against the option.
   5. Price(): stores the price of product for selected ‘product\_id’ and ‘seller\_id’.
   6. Option1(): shows results for option1 of shopper menu.
   7. Option2(): shows results for option2 of shopper menu.
   8. Option3(): shows results for option3 of shopper menu.
   9. Option4(): shows results for option4 of shopper menu.

**Time Management:**

Time the executives for me was dubious, since I had both work routine and classes booked for Thursdays, however of course, my Saturdays were free, which I utilized most of the day to explain my task. Concerning the task unpredictability, section 1 and section 2 set aside me less effort to finish than section 3 which was moderately protracted when contrasted with 1&2 since it required a ton of time for testing and investigating.

Research

<Include here your response to the research topic ‘why relational databases continue to be important for most enterprises despite the rise in popularity of NoSQL databases. Back up your arguments with research’>

**Introduction**

A social information base is characterized as a capacity framework which stores information in way which is associated or related. It comprises of properly orchestrated tables utilizing which a client can get information, control and supplement new information also. It holds information as consistent structures, for example, tables, files and perspectives in a way which is straightforwardly focused for a client to bring, control, produce helpful experiences. A Relational Database Management System is generally spread over the business by and by. It has been a typical stockpiling for the business over recent years. It covers greater part of the portion of the information gathered from the web. Notwithstanding, with entry of time, the volume of information has advanced to billions, presenting new complexities in information the executives, stockpiling and taking care of. To defeat these difficulties, engineers planned another information base called NoSQL information bases, which are getting very famous in the present period. A NoSQL information base or non-social information bases store non-organized data.Almost each industry desires for NoSQL selection because of the preferences it has over the customary social information base frameworks. Different models are MongoDB, Cassandra and so on. These information bases have their own advantages and disadvantages over the social information base administration frameworks. The essential drawback of social data set frameworks is that with inclusion of a lot of information, the information base will in general turn out to be increasingly more mind boggling to a circumstance where sharing data starting with one framework then onto the next ends up being an extremely testing circumstance. Another restriction is that there is a breaking point forced on length of information fields for greater part of social data sets. MySQL has by a long shot the biggest introduce base.(The State of the Open Source Database Industry in 2020, n.d.)

There are various key differences between the relational and non-relational model. Some of them are:

1. 1. In the event of social information bases, there is connection between the information, for instance relations between the tables through keys, though, in NoSQL information base frameworks, there is no connection between the information, the client needs to make and characterize own relations.
2. 2. In Relational information bases, for information control, information recovery and capacity, dialects like SQL are utilized. The compositions in SQL are pre-characterized. While, if there should arise an occurrence of NoSQL, the constructions are dynamic.

3. SQL information bases give usefulness of vertical strength while NoSQL Databases give usefulness of evenly soundness.

4. In instance of multi-column exchanges, SQL information bases end up being a superior choice when contrasted with NoSQL Databases while, if there should be an occurrence of non-organized information, NoSQL information bases have an edge over SQL.

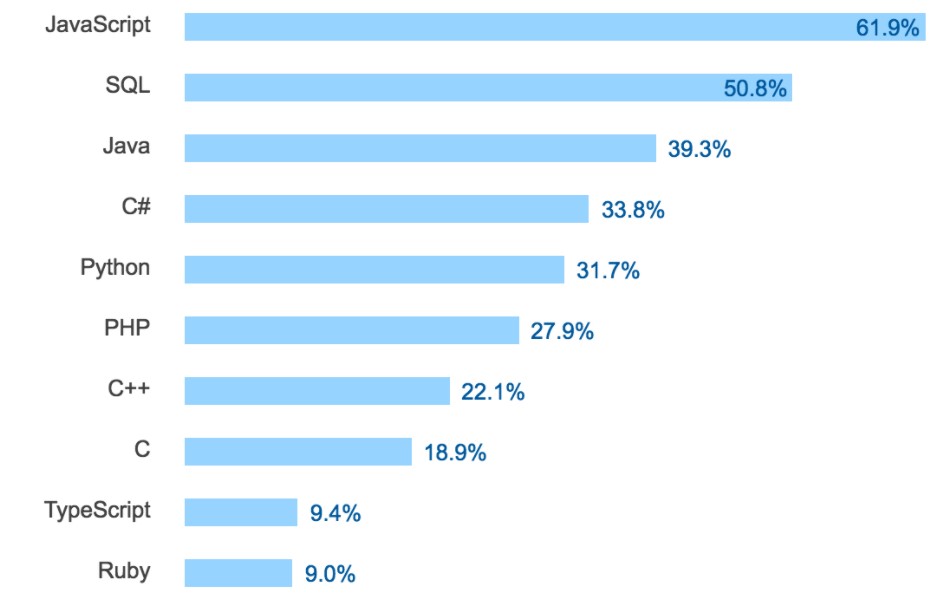
5. The language utilized if there should arise an occurrence of social information base model is SQL, while it's NoSQL for non-social or NoSQL information bases.

The top favourable circumstances the social information bases have over non-social data sets include:

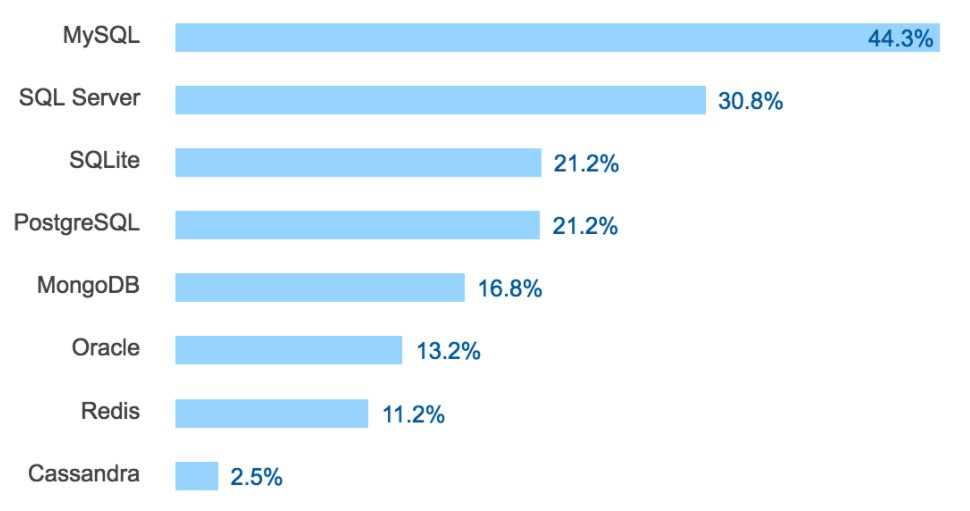
1. Model straightforwardness: The social information bases have basic and easy to understand model while the non-social data sets have complex model and connections.
2. Data upkeep: Since the tables are associated with each other by means of keys, there is no way the information be monotonous or copied. Consequently the exactness is more than some other framework.
3. Flexibility: Insertion, expansion and refreshing of information is simpler than NoSQL models.
4. High Security: Since, information can be separated into a few unique tables, in this manner, the secret information can be covered up.

**Statistics about SQL:**

In 2017, the Stack Overflow Developer’s survey presented the following insights about the most used language. The results are presented below in a graph.

(SQL Survey, n.d.)

The below shown graph depicts the data of most used language by database technology developers.



**Primary differences between Relational DB and NoSQL DB**

A portion of the key contrasts among Relational and Non-social information bases include:

1. In social information bases, we have relations between the information, for instance relations between the tables by means of keys, while, in NoSQL data set frameworks, there is no connection between the information, the client needs to make and characterize own relations.
2. Relational information bases use dialects like SQL for information control, recovery and capacity of information and they additionally have a pre-characterized pattern. While, since, the information is unstructured, accordingly, NoSQL information bases have dynamic patterns.
3. SQL information bases are vertically steady while NoSQL Databases are on a level plane stable.
4. SQL information bases are better for multi-column exchanges, though, NoSQL data sets are better for unstructured information like records or JSON.(SQL vs NoSQL: 5 critical differences, n.d.)

**Still why most enterprisesrely on the use of Relational Database**

1. SQL is accessible as an open source on the internet.Therefore, the undertaking doesn't have to contribute, in any event in the product buy. While greater part of NoSQL information bases accompanies membership. A few instances of SQL Databases are MySQL and PostgreSQL and Cassandra DB for NoSQL Databases.
2. SQL information bases are ACID grumbling. Corrosive stands Atomicity, Consistency, Isolation, Durability. On the off chance that an information base framework is ACID objection, it fulfills a lot of needs that measure the atomicity, consistency, seclusion, and strength of data set frameworks. The more ACID-consistent an information base is, the more it serves to ensure the legitimacy of information base exchanges, diminish oddities, defend information honesty, and make stable information base frameworks. By and large, SQL-based RDBMSs accomplish a significant level of ACID consistence, yet NoSQL information bases surrender this differentiation to pick up speed and adaptability when managing unstructured information. While working with multi report exchanges, NoSQL information bases neglect to execute key activities. This is the place social information bases tackle the issue. Social information bases help the client precisely characterize how a data set will interface with exchanges, which may be gainful in a more drawn out run and is utilized by a significant lump of endeavors directly. In the majority of the ventures, the information requires adaptable examining conduct related and modified meetings, building custom dashboards, and so on social information base encourages client to store and get information from the information base quickly.(SQL v/s NoSQL: What's the distinction?, n.d.)
3. SQL Databases have low intricacy nature. Utilizing the social information bases is generally straightforward than utilizing NoSQL information bases as SQL is a language that is modestly simple to learn. Additionally, it is extremely simple for a little scope venture to embrace and begin working with it.
4. SQL Databases can be utilized to get information utilizing complex questions, NoSQL data set isn't something which can supplant SQL here. Lion's share of organizations have complex information to manage. In such cases, connections between information is a key prerequisite for the engineers/clients to understand the issue in generally less measure of time. A solitary content recovers various types of information from numerous sources. While to so the equivalent in NoSQL will require a lot of scrips, which may get unwieldy for a lot of information and will be much more slow when contrasted with social data sets. SQL highlights empowers the client to produce and speak to huge reports and perceptions on information since the information is in a social organization, which is bulky errand in the event of NoSQL data set.
5. SQL gives extra security highlights. One such model is exchanges in an internet business organization. Exchanges is a key division in all the ventures. An organization would not face a challenge to permit any impedance or glitches in this office in any event. Another motivation behind why SQL comes convenient over NoSQL information base. In the event of higher exchanges, SQL is the main decision. Social information bases help keep up information respectability which makes it more steady than NoSQL data sets. Information trustworthiness on account of exchanges is of absolute most significance. Information honesty guarantee the presence of a solitary key in various tables, which enables the client or framework to confirm the verification of private information, for example, usernames, messages and so forth.
6. The other significant favorable position is cost of activity. Cost is a major worry in any industry. On the off chance that an industry needs to receive the NoSQL information base activities, the organizations should recruit Data Scientists and engineers to complete the tasks, which thusly are more costly to employ when contrasted with Data Analysts.
7. SQL is a language that has been in presence for a very long time. Thusly, it has an enormous designer base just as help network.Considering all these points, relational databases look too though to be replaced though NoSQL databases continue torise in demand, but relational databases stillhold majority of the share of the industry and is nowhere going to be replaced anytime sooner.

**Bibliography and References**

(SQL v/s NoSQL: What's the difference?, n.d.)

(Wikipedia, n.d.)

(SQL Survey, n.d.)

(The State of the Open Source Database Industry in 2020, n.d.)