FIT5137 Assignment 3 - S2 2023
Data Cleaning

STUDENT NAME & STUDENT ID:
Wanru Xiang 33729220;
Linhao Wang 31273327;
Ziqi Pei 33429472 ;

Details of Oracle accounts:
Wanru Xiang: S33729220
Ziqi Pei: S33429472
Linhao Wang: S31273327

1 Data dictionary

Table Name		Article			
Attribute name	Description	Data Type	Character Length Format	Accept Null values	Primary Key(Y/N)
articleCode	Code of the article	VARCHAR2	Max: 26	N	Y
articleName	Name of the article	VARCHAR2	Max: 128	N	N
VendonKey	Key of vendor	VARCHAR2	Max: 26	N	N
VendorName	Name of the vendor	VARCHAR2	Max: 26	N	N
categoryInit	Name of the category	number	Max:38	Y	N
categoryName	Name of the category	VARCHAR2	Max: 26	Y	N
typeInit	A number identifier	Number	Max:38	Y	N
TypeName	TypeName	VARCHAR2	Max: 26	N	N
startDate	Start date of selling this article	date	YYYY-MM-DD	Y	N
expireDate	Expiry date of the article	date	YYYY-MM-DD	Y	N
colorInit	Color Id	number	Max:38	Y	N
colorName	Color name	VARCHAR2	Max: 26	Y	N
sex	Gender for which the article is intended	char(1)		Y	N
picture	Picture of the	missing	VARCHAR(26)	Y	N

	article				
basePrice	Base Price of the article	float		Y	N
salePrice	Sales price of the article	float		Y	N
notes	Note about the article	VARCHAR2	Max: 128	Y	N

Table Name:		Cashier			
Attribute name	Description	Data Type	Character Length/Format	Accept Null values	Primary Key(Y/N)
noTrans	Transaction Number	VARCHAR2	NNAAAANNANNN NN	N	Y
dateTrans	Dateline of transaction	TIMESTAMP	YYYY-MM-DD HH24:MI:SS.FF	N	N
typeTrans	Type of transaction	VARCHAR2		Y	N
Note	notes	VARCHAR2		Y	N
UserID	Id of the cashier	VARCHAR2		Y	N
ReferenceTrans	Transaction reference	missing	missing	Y	N

Table Name		Cashierdetail			
Attribute name	Description	Data Type	Character Length/Format	Accept Null values	Primary Key(Y/N)
noTrans	Transaction Number	VARCHAR2	NNAAAANNANNN NNN	N	Y
ArticleCode	Article Code	VARCHAR2		N	
Barcode	Article barcode	VARCHAR2		N	Y
sizes	Sizes of the article	VARCHAR2		N	
qty	Quantity of	int		N	

	the article				
basePrice	Base price of the article	float		Y	
salePrice	Sale price of the article	float		Y	
DiscountType	Type of discount	int	0-3	Y	
discountPerson	Discount percentage	Number(5,2)		Y	
discountRupiah	Given Discount on sale price	float		Y	
DiscExpenses	Flag to indicate whether there is discountRupiah	number(1)	0/1	Y	
consignment	Consignment is always 0	NUMBER(38, 0)		Y	
consignmentRp	RP: Retail Price ;A commission fee: Subtotal-consignment Rupiah == pay ment	NUMBER(38, 0)		Y	
subTotal	Discounted price sale price - discount Rupiah =subtotal	NUMBER(38, 0)		Y	
payment	Payment to supplier	NUMBER(38, 0)		Y	

Sales Price - discount Rupiah = subtotal
Subtotal = consignment Rupiah == payment
Consignment is always 0

DistExpress flags whether there is discountRuplish

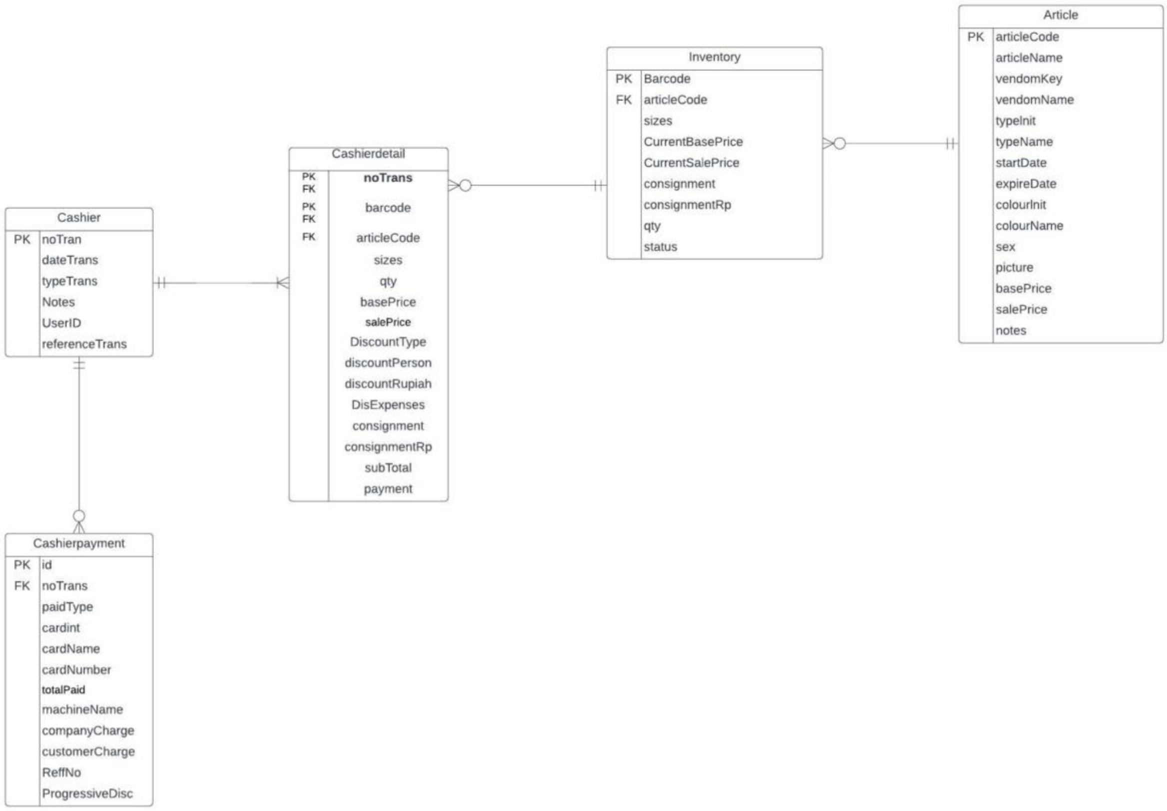
Table Name		cashierpayment			
Attribute name	Description	Data Type	Character Length/Format	Accept Null values	Primary Key(Y/N)
id	Unique identifier for the payment	NUMBER	NUMBER(38, 0)	N	Y
noTrans	Transaction number, a unique identifier for each transaction	VARCHAR	Max: 26	N	
paidType	The method used for the payment(e.g Bank TRANSFER, CASH)	varchar	Max: 26	N	
cardint	Initial of the card used for the payment the name is empty if the payment was made in cash	varchar	Max: 26	Y	
cardName	Name of the card, The field is empty if the payment was made in cash	varchar	Max: 26	Y	
cardNumber	Card number	varchar	Max: 26 NNNN-NNNN-NNNN-NNNN	Y	
totalPaid	The total	Number			

	amount paid of this payment.				
machineName	The name of the machine where transaction was made.	varchar	Max: 26 Initial of card,eg CBA		
companyCharge	The charge that the company levels for the transaction	Number	missing/0		
customerCharge	The change levied ob the customer for the transaction	Number	missing/0		
ReffNo	Reference number for the transaction	varchar	Max: 26		
ProgressiveDisc	The progressive discount applied to the transaction	number		N	

Table Name		Inventory			
Attribute name	Description	Data Type	Character Length/Format	Accept Null values	Primary Key(Y/N)
articleCode	Unique identifier for the article	VARCHAR(26)	NNNN-NNNN	N	Y
Barcode	Barcode for the article	VARCHAR(26)	NNNNNNNNNNN	N	N

sizes	Size of the article	VARCHAR(26)	N	N	N
CurrentBasePrice	Current base price of the article	NUMBER(38,0)		Y	
CurrentSalePrice	Current sale price of the article	NUMBER(38,0)		Y	
consignment	Consignment value for the article	NUMBER(38,0)	NNNN-NNNN-NNNN-NNNN	Y	N
consignmentRp	Consignment RP value for the article.	NUMBER(38,0)		Y	
qty	Quantity of the article.	INT	Initial of card,eg CBA	N	N
status	Status of the article	INT		N	

(2)ERD



Link
https://lucid.app/lucidchart/aace1162-c4d8-4164-b1f3-71c3844e6b61/edit?view_items=1ItvaEI-rkp5&invitationId=inv_fe41f8de-909d-4f83-b715-00b3375cb365

Task3
Data Importing and Cleaning Strategies in SQL:

1.Select Appropriate Data Sources:
In SQL, choose reliable and up-to-date data sources such as databases or data files (e.g., CSV, Excel) that align with the project's objectives.

2.Verify Data Integrity:
Ensure data sources are validated and have data integrity checks (e.g., constraints) in place to detect any inconsistencies during the import process.
Initial Data Exploration:

3. Conduct EDA (Exploratory Data Analysis):
Utilize SQL queries to explore the dataset's structure and characteristics.

(1).Check Data Types:
Examine column data types using SQL functions like DATA_TYPE in the information_schema to ensure they match the nature of the data.

(2).Handle Missing Values:
Use SQL functions like COUNT and NULL checks to identify missing values and assess their impact.

(3)Detect Outliers:
Use SQL aggregation functions to calculate summary statistics, identify outliers, and decide whether to correct, remove, or keep them based on domain knowledge.
Handling Missing Values:

4. Identify Missing Values:

Use SQL queries to identify columns with missing values and calculate the percentage of missingness.

(1)Strategies for Handling Missing Values:
Omitting rows or columns with excessive missing values using SQL DELETE statements.
Imputing missing values in numerical columns using SQL UPDATE statements with mean, median, or mode calculations.
For sequential data, use SQL for interpolation or extrapolation techniques.
For categorical data, impute based on domain knowledge using SQL CASE statements.

5. Identify Incorrect Values:

Use SQL queries and conditions to identify columns with incorrect or inconsistent values based on domain knowledge and data validation.
(1)Strategies for Handling Incorrect Values:

For numerical data, address outliers and erroneous values using SQL UPDATE or DELETE statements based on domain knowledge or statistical analysis.

For categorical data, correct typos and inconsistencies in category names using SQL UPDATE statements, and standardize values.

6. Standardize Data Formats:
- Use SQL UPDATE statements to ensure consistent data formats and units across different sources.

(1)Convert Data Types:

Use SQL functions to convert data types to their appropriate representations (e.g., CAST for datetime conversion).

(2)Create Derived Features:

Generate new features or aggregations using SQL SELECT statements when they provide valuable insights or simplify analysis.

(3)Normalization/Scaling:
Normalize or scale numerical features using SQL operations if needed, especially for machine learning models that require it.
Data Validation and Verification:

7. Implement Data Checks:
- Apply checks and validation rules using SQL constraints (e.g., foreign key constraints) to ensure data quality and integrity.

(1)Cross-Verify Data:
Cross-verify data against trusted sources or external references when applicable using SQL joins and comparisons.

8. Maintain Detailed Documentation:
- Document every step of the data cleaning process using comments or a documentation system within your SQL environment.

(1)Document Assumptions:
Clearly state any assumptions made during data cleaning and their potential impact on subsequent analyses using SQL comments.

(2)Communicate Impact and Limitations:

Use SQL comments or external documentation to communicate the potential impact of data cleaning on subsequent analyses and any limitations introduced during the process.

9. Test Rigorously:
- Test the data cleaning process using SQL queries on a subset of the data before applying it to the entire dataset.

(1)Iterate as Needed:
Be prepared to iterate on data cleaning steps based on initial testing results and emerging insights using SQL statements.

10. Create Backups:
- Establish a backup system or checkpoints within your SQL environment to ensure data integrity is preserved throughout the cleaning process.

11. Automate Processes:
- Whenever possible, automate data cleaning processes within your SQL environment to enhance reproducibility and efficiency using stored procedures or scripts.

(1)article cleaning strategy

1.Checking Duplicate Records by articleCode:
This query counts how many times each articleCode appears in the "article" table.
It identifies cases where the same articleCode appears multiple times (i.e., duplicates) and provides a count of occurrences.
2.Checking Duplicate categoryName:
This query counts how many times each categoryName appears in the "article" table.
It identifies cases where the same categoryName appears multiple times and provides a count of occurrences.
You've noted that "T-SHIRT" and "TSHIRT" are considered duplicates, and there are three rows for "TSHIRT."
3.Checking for NULL articleCode:
This query selects all rows from the "article" table where the articleCode is NULL.
It identifies rows where articleCode is missing.
4.Checking for NULL articleName:
This query selects all rows from the "article" table where the articleName is NULL.
It identifies rows where articleName is missing.
5.Checking for NULL VendonKey:
This query selects all rows from the "article" table where VendonKey is NULL.
It identifies rows where vendorkey is missing.
6.Checking for NULL VendorName:
This query selects all rows from the "article" table where VendorName is NULL.
It identifies rows where VendorName is missing.
7.Checking for NULL TypeName:
This query selects all rows from the "article" table where typename is NULL.
It identifies rows where typename is missing.

(2)cashier cleaning Strategy:

1.Check for Duplicate Records:

Use SQL queries to identify duplicate records based on a unique identifier (e.g., notrans).
Verify that there are no duplicate entries, ensuring data integrity.

2.Check for Missing Values:

Use SQL queries to identify columns with missing values (e.g., notrans and datetrans).
Ensure that no critical fields have NULL or missing values that could affect data quality.

3.Checking Constraints and Relationship Issues:

Validate data against any constraints or relationships defined in the database schema.
Ensure that foreign keys and other integrity constraints are upheld.

4.Data Validation and Verification:

Implement checks to validate data quality and integrity.
Cross-verify data against trusted sources or external references when applicable.

5.Documentation:

Maintain detailed documentation of the data cleaning process.
Document any assumptions made during data cleaning.
Clearly state the potential impact of data cleaning on subsequent analyses and any limitations introduced.

6.Testing and Iteration:

Rigorously test the data cleaning process on a subset of the data before applying it to the entire dataset.
Be prepared to iterate on data cleaning steps based on initial testing results and emerging insights.

7.Data Backup:

Create backups or checkpoints of the cleaned data to ensure data integrity is preserved throughout the cleaning process.

8.Automation (Optional):

If feasible, automate data cleaning processes to enhance reproducibility and efficiency.

(3)cashierdetail cleaning Strategy

1.Check for Duplicate Records:

Use SQL queries to identify duplicate records based on a combination of columns (e.g., notrans and barcode).
Verify that there are no duplicate entries, ensuring data integrity.

2.Checking Constraints and Relationship Issues:

Validate data against any constraints or relationships defined in the database schema.
Ensure that data adheres to constraints such as the base price being lower than or equal to the sale price.

3.Inconsistent Values - baseprice Column:

Identify and address inconsistent values in the baseprice column, where baseprice is greater than saleprice.

4.Omitting Rows:

Remove rows that contain data inconsistencies or errors that cannot be corrected:
Delete rows where notrans and barcode match specific values.
These deletions address issues that cannot be resolved through correction.

5.Incorrect Values - Payment Column:

Identify and address rows where the payment column is NULL.
Update the data by filling in missing values based on context or domain knowledge.

6.Errors Due to Misaligned Columns - Correct Data:

Correct rows where data in certain columns is misaligned and does not match the expected structure.
Update the data for specific rows to ensure alignment with the correct column values.

(4) cashierpayment cleaning Strategy

1.Check for Duplicate Records:

Use SQL queries to identify duplicate records based on the id column.
Ensure that there are no duplicate entries, maintaining data integrity.

2.Checking Constraints and Relationship Issues:

Validate data against any constraints or relationships defined in the database schema.
Incorrect Values - ID Column:

3.Identify and address rows where the id column is NULL.

Determine if NULL values in the id column are acceptable or if they need to be handled differently.

4.Incorrect Values - Notrans Column:

Identify and address rows where the notrans column is NULL.
Determine if NULL values in the notrans column are acceptable or if they need to be handled differently.

5.Incorrect Values - Paidtype Column:

Identify and address rows where the paidtype column is NULL.
Determine if NULL values in the paidtype column are acceptable or if they need to be handled differently.

6.Incorrect Values - progressivedisc Column:

Identify and address rows where the progressivedisc column is NULL.
Calculate the mean (average) of non-NULL progressivedisc values.
Update rows with NULL progressivedisc values and replace them with the calculated mean (average) value of 0.

(5)Inventory Table Cleaning Strategy:

1.Check for Duplicate Records:

Use SQL queries to identify duplicate records based on the barcode column.
Ensure that there are no duplicate entries, maintaining data integrity.

2.Checking Constraints and Relationship Issues:

Validate data against constraints and relationships, such as verifying that articlecode values exist in the article table.
Identify and address any invalid foreign key (FK) values.

3.Inconsistent Values - Currentbaseprice and Currentsaleprice Columns:

Detect and address rows where currentbaseprice is greater than currentsaleprice.
Impute these rows based on neighboring values by setting currentbaseprice to 51350.

4.Incorrect Values - Status Column:

Identify and address rows where status is less than 0.
Calculate the median of non-NULL status values, which is 0.
Impute NULL status values with the calculated median of 0.

5.Incorrect Values - Numeric Columns:

Identify and address rows where numeric columns (currentbaseprice, currentsaleprice, consignment, consignmentrp, qty) have negative values.
Decide on an appropriate handling strategy for these rows, such as correcting, removing, or replacing with meaningful values.

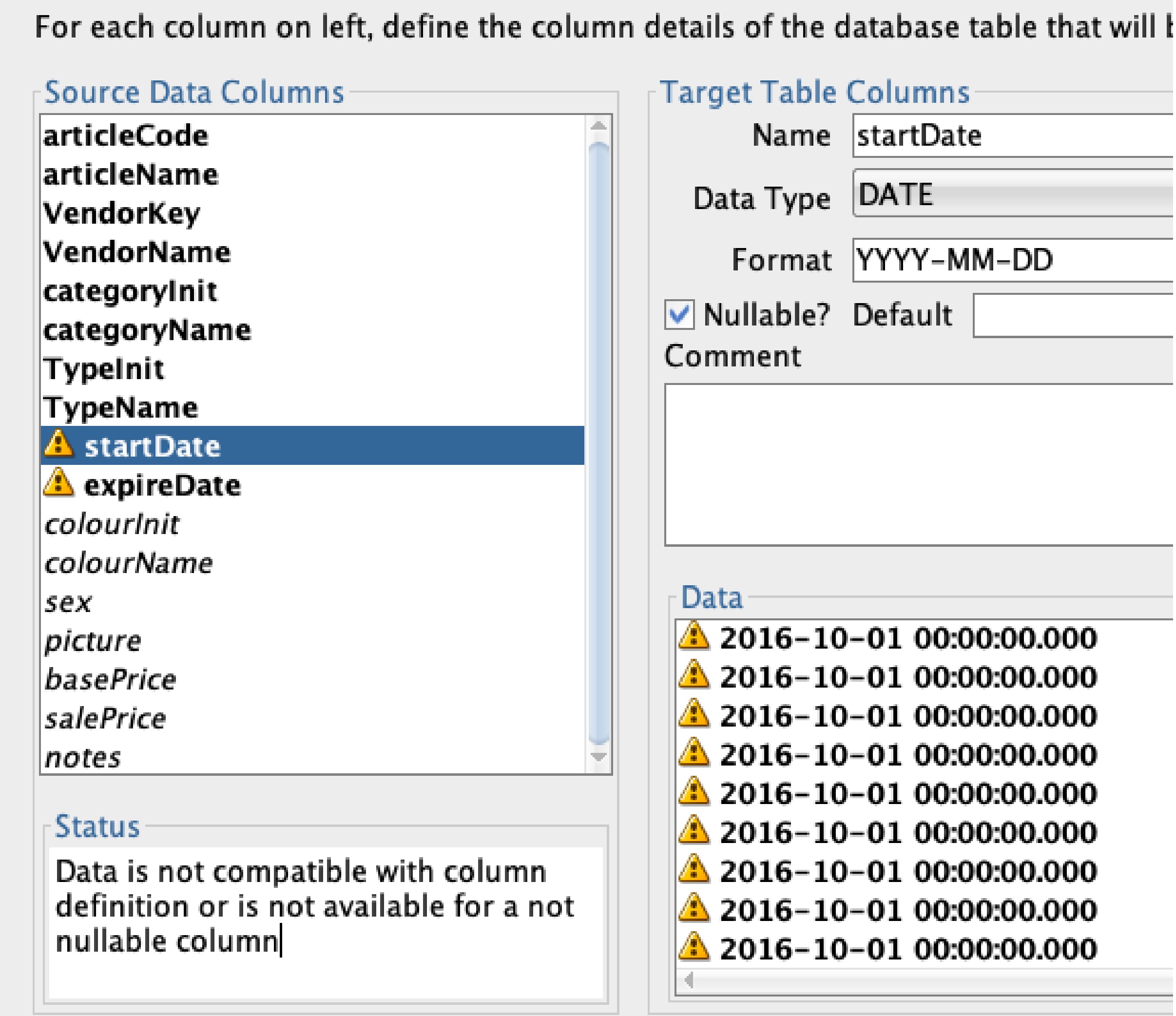
6.Missing Values - Various Columns:

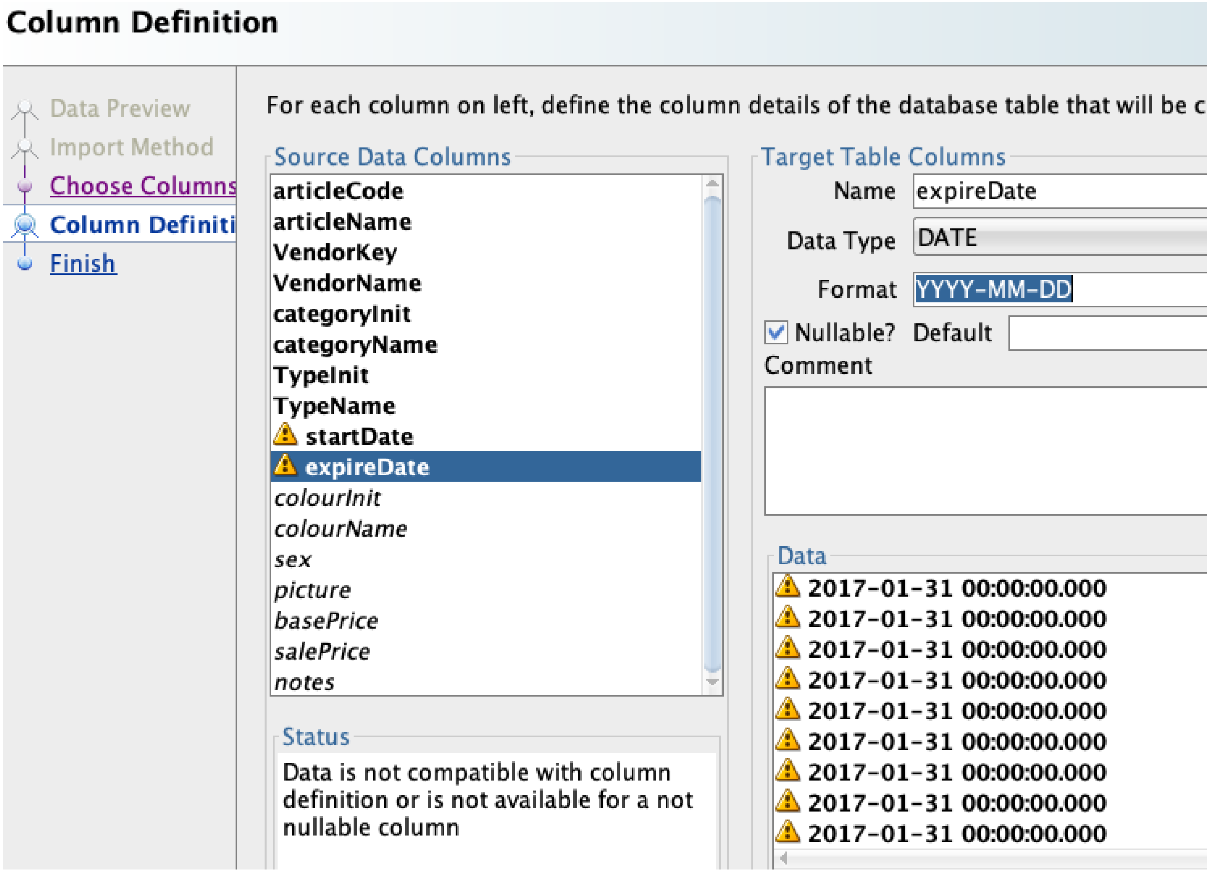
Identify and address rows where key columns (articlecode, barcode, sizes, qty, status) have NULL values.
Implement appropriate strategies for handling missing values, such as imputation or correction.

Task 4.b Errors Log

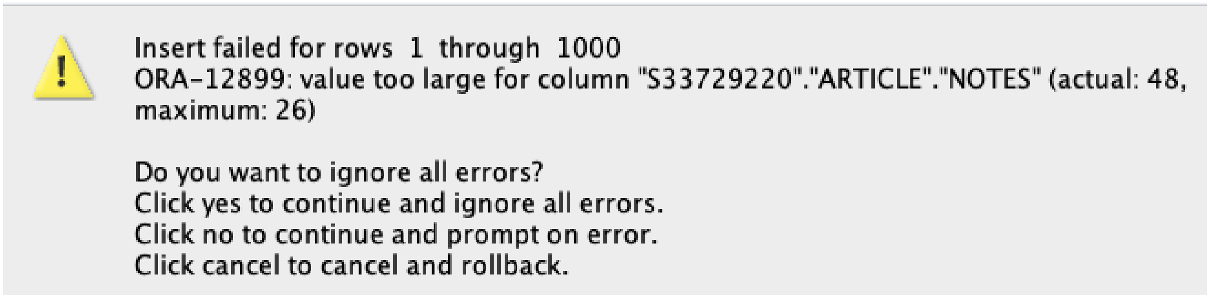
1. Article Table

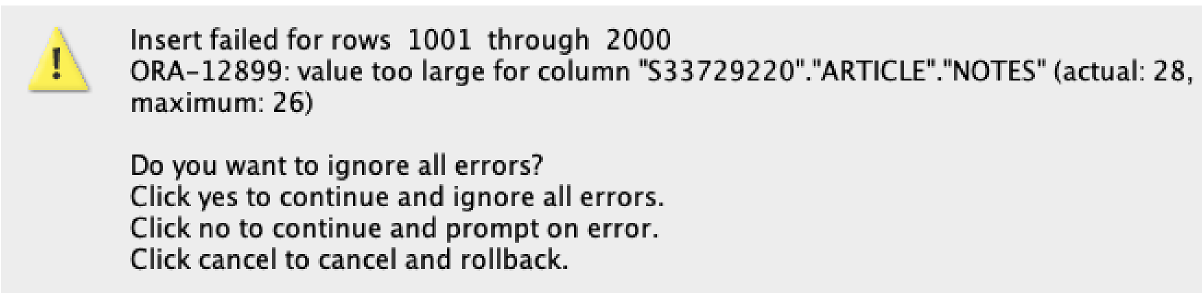
Data Type of startDate and expireDate should be DATE in the 'YYYY-MM-DD' format -> Using python or other programming languages to edit the data type before importing
For instance:
"df['startDate'] = pd.to_datetime(df['startDate'], infer_datetime_format=True)"





Value too large -> Change the size of Data Type for Column “NOTES” from 26 into 128





Insert failed for rows 2001 through 3000
ORA-12899: value too large for column "S33729220"."ARTICLE"."NOTES" (actual: 28, maximum: 26)

Do you want to ignore all errors?
Click yes to continue and ignore all errors.
Click no to continue and prompt on error.
Click cancel to cancel and rollback.

Insert failed for rows 3001 through 4000
ORA-12899: value too large for column "S33729220"."ARTICLE"."NOTES" (actual: 27, maximum: 26)

Do you want to ignore all errors?
Click yes to continue and ignore all errors.
Click no to continue and prompt on error.
Click cancel to cancel and rollback.

Insert failed for rows 4001 through 5000
ORA-12899: value too large for column "S33729220"."ARTICLE"."NOTES" (actual: 28, maximum: 26)

Do you want to ignore all errors?
Click yes to continue and ignore all errors.
Click no to continue and prompt on error.
Click cancel to cancel and rollback.

Insert failed for rows 5001 through 6000
ORA-12899: value too large for column "S33729220"."ARTICLE"."NOTES" (actual: 35, maximum: 26)

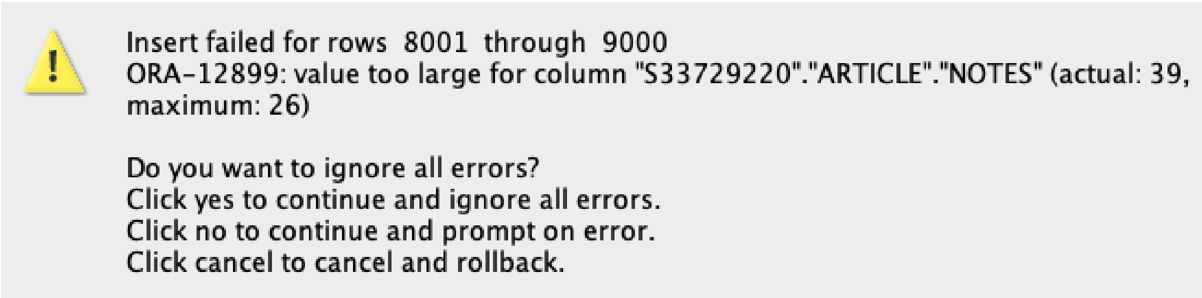
Do you want to ignore all errors?
Click yes to continue and ignore all errors.
Click no to continue and prompt on error.
Click cancel to cancel and rollback.

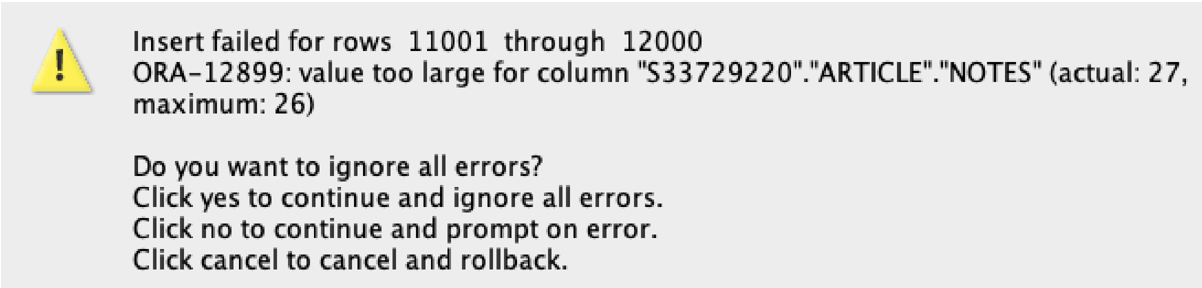
Insert failed for rows 6001 through 7000
ORA-12899: value too large for column "S33729220"."ARTICLE"."NOTES" (actual: 28, maximum: 26)

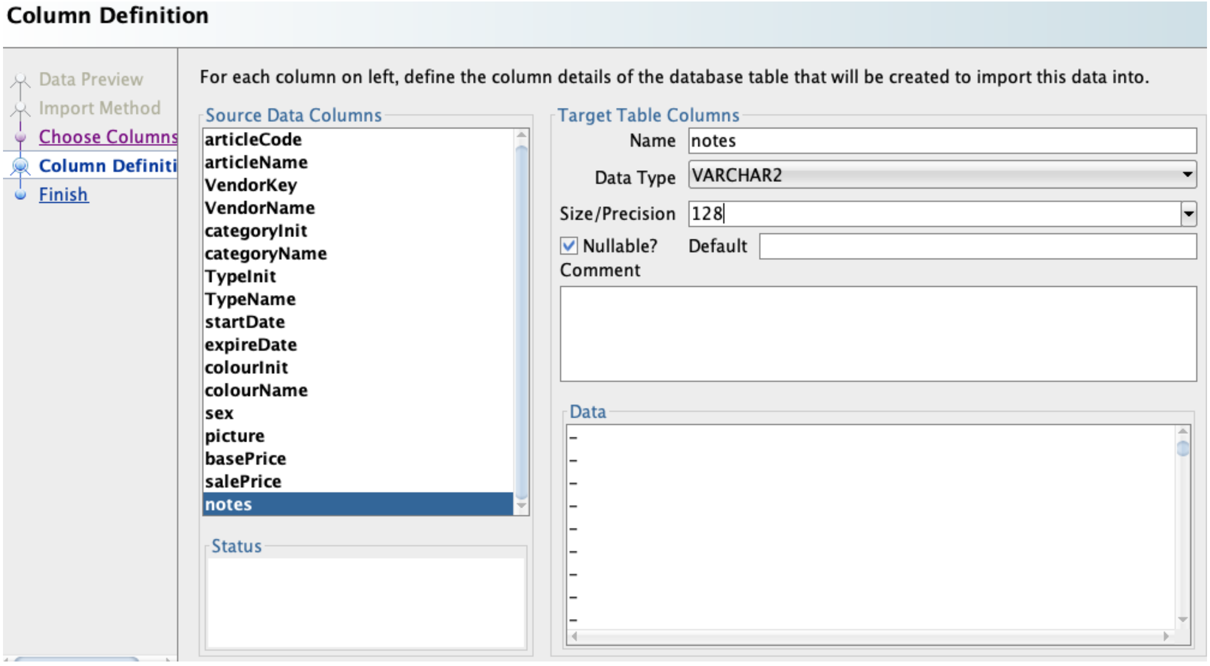
Do you want to ignore all errors?
Click yes to continue and ignore all errors.
Click no to continue and prompt on error.
Click cancel to cancel and rollback.

Insert failed for rows 7001 through 8000
ORA-12899: value too large for column "S33729220"."ARTICLE"."NOTES" (actual: 27, maximum: 26)

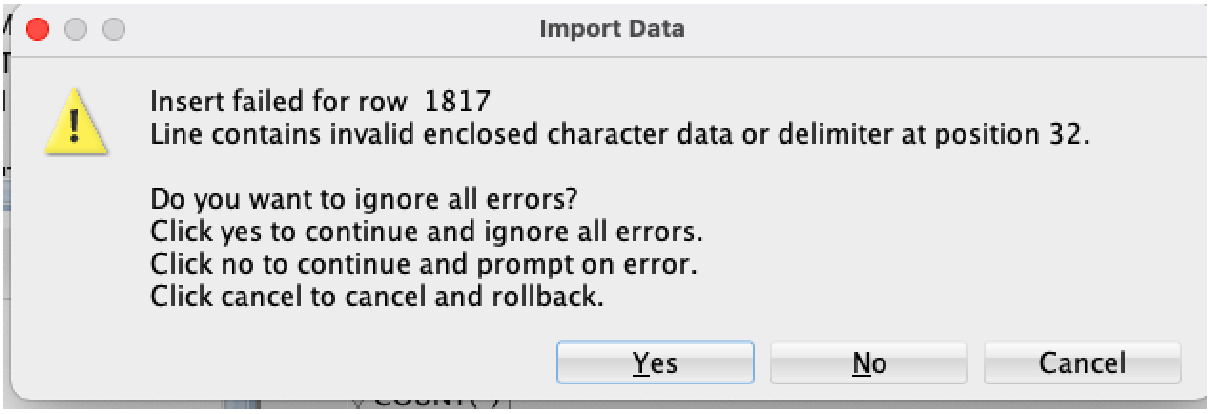
Do you want to ignore all errors?
Click yes to continue and ignore all errors.
Click no to continue and prompt on error.
Click cancel to cancel and rollback.

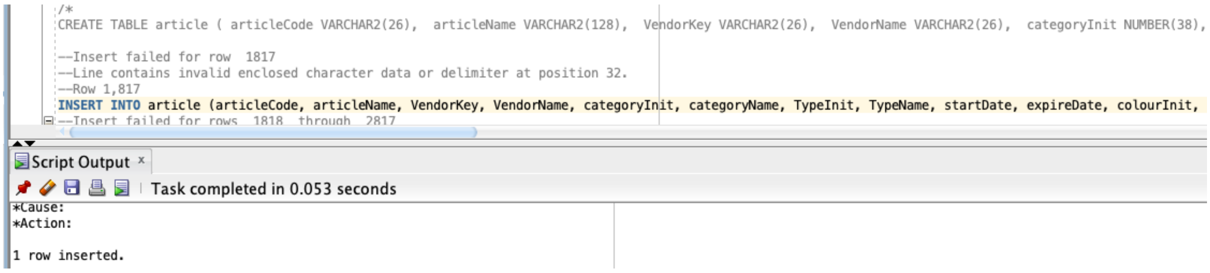


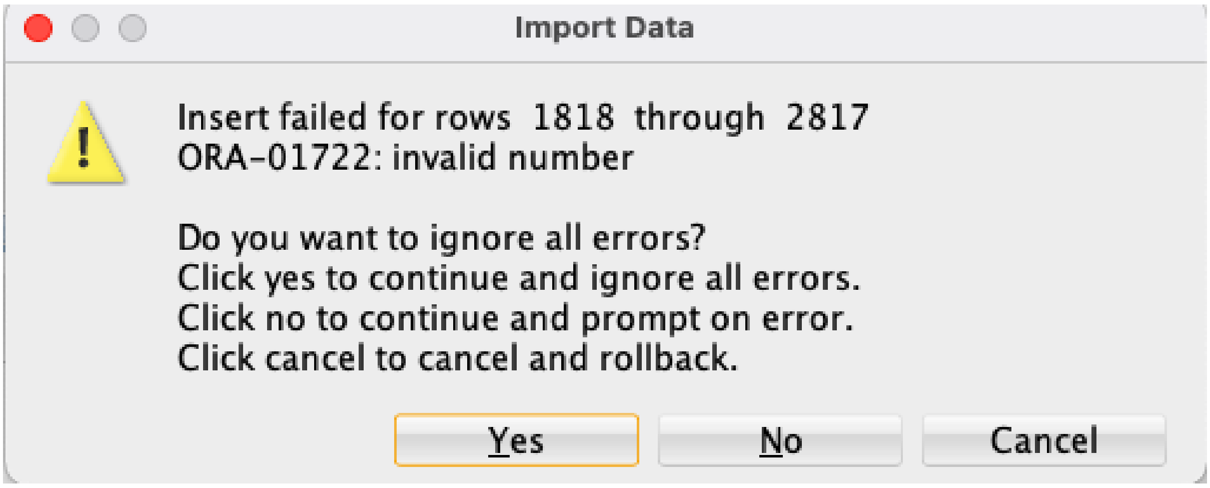


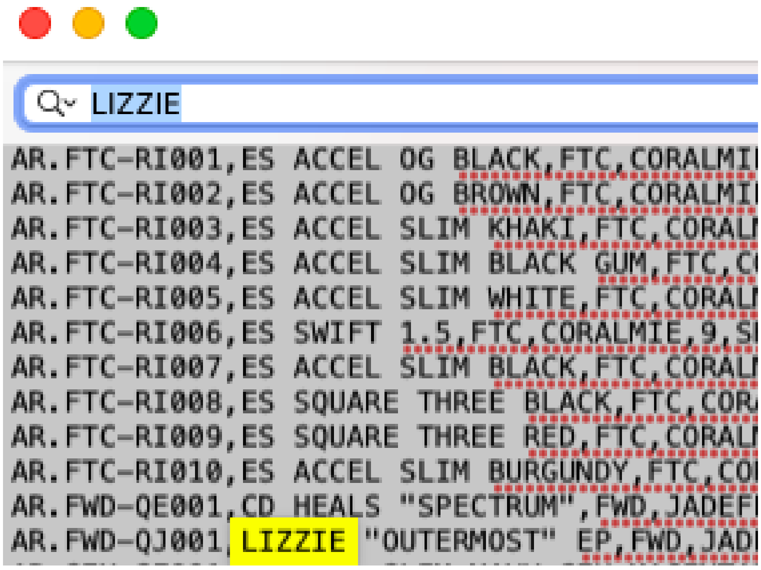


Invalid Values -> Change the related values into the required format according to delimiter









--Insert failed for row 1817
--Line contains invalid enclosed character data or delimiter at position 32.
--Row 1,817
INSERT INTO article (articleCode, articleName, VendorKey, VendorName, categoryInit, categoryName, TypeInit, TypeName, startDate, expireDate, colourInit, colourName, sex, picture, basePrice, salePrice, notes) VALUES ('AR.FWD-QJ001','LIZZIE "OUTERMOST" EP','FWD','JADEFELLOW',1,'ACCESSORIES',3,'CD','2017-10-21 00:00:00.000','2018-01-19 00:00:00.000',0,'NA','M','',33750,45000,'RECEIVE ORDER OCTOBER 2017');

--Insert failed for rows 1818 through 2817
--ORA-01722: invalid number
--Row 2,576

INSERT INTO article (articleCode, articleName, VendorKey, VendorName, categoryInit, categoryName, TypeInit, TypeName, startDate, expireDate, colourInit, colourName, sex, picture, basePrice, salePrice, notes) VALUES ('AR.LWS-QB018','WO,F BLACK','LWS','COOLTOUR','11','T-SHIRT','1','CASUAL','2017-02-18 00:00:00.000','2017-05-19 00:00:00.000',1,'BLACK','M',NULL,111300,159000,'RECEIVE ORDER FEBRUARY 2017');

Inconsistent Values -> Change the related values

```
--Duplicate naming: T-SHIRT & TSHIRT
--3 rows for TSHIRT
SELECT
    categoryname,
    COUNT(*)
FROM
    article
GROUP BY
    categoryname
HAVING
    COUNT(*) > 1;
```

2. Inventory Table

Inconsistent Value -> base price means minimum value price, so it must be lower or equal to sale price; thus, imputing based on neighboring values: 51350

```
--inconsistent values
--Base price simply means minimum value price
select * from inventory where
currentbaseprice > currentsaleprice;
```

Query Result

SQL | All Rows Fetched: 1 in 0.021 seconds

	ARTICLECODE	BARCODE	SIZES	CURRENTBASEPRICE	CURRENTSALEPRICE	CONSIGNMENT	CONSIGNMENTRP	QTY	STATUS
1	AR.STY-RJ010	20ARRJ0113	ALL	513500	79000	0	27650	0	0

```sql
--inconsistent values
--Base price simply means minimum value price
--1 error in Currentbaseprice Column
SELECT
    *
FROM
    inventory
WHERE
    currentbaseprice > currentsaleprice;

--Imputing based on neighboring values: 51350
SELECT
    currentbaseprice
FROM
    inventory
WHERE
    currentsaleprice = (
        SELECT
            currentsaleprice
        FROM
            inventory
        WHERE
            currentbaseprice > currentsaleprice
    )
GROUP BY
    currentbaseprice;

UPDATE inventory
SET
    currentbaseprice = 51350
WHERE
    currentsaleprice = (
        SELECT
            currentsaleprice
        FROM
            inventory
        WHERE
            currentbaseprice > currentsaleprice
    );
```

Inconsistent Value -> Status is NOT NULL; Imputing Null of Status with Median: 0

```sql
select * from inventory where status is null;
```

Query Result ×

SQL | All Rows Fetched: 4 in 0.024 seconds

	ARTICLECODE	BARCODE	SIZES	CURRENTBASEPRICE	CURRENTSALEPRICE	CONSIGNMENT	CONSIGNMENTRP	QTY	STATUS
1	AGA-PI0020	2016I00080	XL	130000	200000	30	70000	0	(null)
2	AR.REC-RJ002	20ARRJ0367	32	237250	365000	0	127750	0	(null)
3	ESH.AR-SE0006	00ARSE0021	S	44000	135000	0	40500	3	(null)
4	BSC.AR-QL0010	00ARQL0067	M	94500	135000	0	40500	0	(null)

```sql
--Calculate Median of Status: 0
SELECT
    MEDIAN(status)
FROM
    inventory
WHERE
    status IS NOT NULL;

--Imputing Null of Status with mediar
UPDATE inventory
SET
    status = 0
WHERE
    status IS NULL;
```

Query Result × Script Output ×

Task completed in 0.054

```
4 rows updated.
```

Relationship problems or Constraint Violation: Invalid FK Values -> Delete these invalid values: 33 rows

```sql
SELECT
    *
FROM
    inventory
WHERE
    articlecode NOT IN (
        SELECT
            articlecode
        FROM
            article
    );
--resolve this issue (simplest approach):
DELETE FROM inventory
WHERE
    articlecode NOT IN (
        SELECT
            articlecode
        FROM
            article
    );
```

3. Cashierdetail Table

Inconsistent Value -> base price means minimum value price, so it must be lower or equal to sale price; thus, omitting these rows

```
--inconsistent values
select * from cashierdetail where baseprice > saleprice;
```

Query Result

All Rows Fetched: 7 in 0.017 seconds

	NOTRANS	ARTICLECODE	BARCODE	SIZES	QTY	BASEPRICE	SALEPRICE	DISCOUNTTYPE	DISCOUNTPERSEN	DISCOUNTRUPIAH	DISCEXPENSES	CONSIGNMENT	CONSIGNMENTF
1	01SCS19E070061	ESH.AR-RB0017	00ARRB0115	S	1	450000	135000	2	0	67500	1	0	135
2	01SCS19E070086	ESH.AR-RB0017	00ARRB0116	M	1	450000	135000	2	0	67500	1	0	135
3	01SCS19E190261	BSC.AR-RJ0007	00ARRJ0063	S	1	430000	135000	2	0	40500	1	0	189
4	01SCS19E250137	BSC.AR-RJ0007	00ARRJ0063	S	1	430000	135000	0	0	0	0	0	270
5	01SCS19E250332	BSC.AR-RJ0007	00ARRJ0064	M	1	430000	135000	2	0	40500	1	0	189
6	01SCS19E260050	BSC.AR-RJ0007	00ARRJ0064	M	1	430000	135000	0	0	0	0	0	270
7	01SCS19E300663	AR.ESB-SE003	20ARSE0247	M 1	204750	315000	2	0	63000	1	0	63000	2520

```
--Omitting rows
delete from cashierdetail where upper(notrans) = upper('01SCS19E070061') and upper(barcode) = upper('00ARRB0115');
delete from cashierdetail where upper(notrans) = upper('01SCS19E070086') and upper(barcode) = upper('00ARRB0116');
delete from cashierdetail where upper(notrans) = upper('01SCS19E190261') and upper(barcode) = upper('00ARRJ0063');
delete from cashierdetail where upper(notrans) = upper('01SCS19E250137') and upper(barcode) = upper('00ARRJ0063');
delete from cashierdetail where upper(notrans) = upper('01SCS19E250332') and upper(barcode) = upper('00ARRJ0064');
delete from cashierdetail where upper(notrans) = upper('01SCS19E260050') and upper(barcode) = upper('00ARRJ0064');
```

Script Output | Query Result

Task completed in 0.153 seconds

```
1 row deleted.

1 row deleted.

1 row deleted.

1 row deleted.

1 row deleted.

1 row deleted.
```

Inconsistent Value -> Payment is NOT NULL; Errors due to misaligned columns; Correct the data

```
--incorrect values: 2 Errors and the 2nd one is same as the one in "inconsistent values"
SELECT
    *
FROM
    cashierdetail
WHERE
    payment IS NULL;
```

Query Result

All Rows Fetched: 2 in 0.013 seconds

	NOTRANS	ARTICLECODE	BARCODE	SI...	QTY	BA...	SAL...		DIS...	DI...	DISC...	CO...	CON...	SUBTOTAL	PAYMENT
1	01SCS19E010027	BSC.AR-SC0034	00ARSC0228	S	1	125000	315000	2	0	94500	1	0	44100	220500 176400	(null)
2	01SCS19E300663	AR.ESB-SE003	20ARSE0247	M 1	204750	315000	2	0	63000	1	0	63000	252000	189000	(null)

```
--Errors due to the misaligned columns
--Correct the data
UPDATE cashierdetail
set
    subtotal = 220500,
    payment = 176400
WHERE
    payment IS NULL and upper(notrans) = upper('01SCS19E010027');
```

Script Output | Query Result

Task completed in 0.043 seconds

```
1 row updated.
```

```sql
UPDATE cashierdetail
SET
    sizes = 'M',
    qty = 1,
    baseprice = 204750,
    saleprice = 315000,
    discounttype = 2,
    discountpersen = 0,
    discountrupiah = 63000,
    discexpenses = 1,
    consignment = 0,
    consignmentrp = 63000,
    subtotal = 252000,
    payment = 189000
WHERE
    payment IS NULL
    AND upper(notrans) = upper('01SCS19E300663');
```

Query Result × | Script Output ×

Task completed in 0.049 seconds

```
1 row updated.
```

4. Cashier Table: No errors

5. Cashierpayment Table

Inconsistent Value -> Progressivedisc is NOT NULL; Imputing Null of Progressivedisc with mean: 0

```sql
SELECT
    *
FROM
    cashierpayment
WHERE
    progressivedisc IS NULL;
```

Script Output × | Query Result ×

SQL | All Rows Fetched: 2 in 0.026 seconds

	ID	NOTRANS	PAIDTYPE	CARDINIT	CARDNAME	CARDNUMBER	TOTALPAID	MACHINENAME	COMPANYCHARGE	CUSTOMERCHARGE	REFFNO	PROGRESSIVEDISC
1	61785	01SCS19E070078	CASH	–	–	–	94500	–	0	0	–	(null)
2	68275	01SCS19E310181	CASH	–	–	–	135000	–	0	0	–	(null)

```sql
--Imputing Null of Progressivedisc with mean: 0
SELECT
    AVG(progressivedisc)
FROM
    cashierpayment
WHERE
    progressivedisc IS NOT NULL;

UPDATE cashierpayment
SET
    progressivedisc = 0
WHERE
    progressivedisc IS NULL;
```

Query Result × | Script Output ×

Task completed in 0.033 seconds

```
2 rows updated.
```

Task 5

```sql
--Task 5: Create a DDL script that generates the required table structures based on the defined data dictionary

/*
drop table article;
drop table inventory;
drop table cashier;
drop table cashierdetails;
drop table cashierpayment;
*/

--create table article
CREATE TABLE article (
    articlecode  VARCHAR(26) PRIMARY KEY,
    articlename  VARCHAR(128) NOT NULL,
    vendorkey    VARCHAR(26) NOT NULL,
    vendorname   VARCHAR(26) NOT NULL,
    categoryinit NUMBER(38, 0),
    categoryname VARCHAR(26),
    typeinit     NUMBER(38, 0),
    typename     VARCHAR(26) NOT NULL,
    startdate    DATE,
    expiredate   DATE,
    colourinit   NUMBER(38, 0),
    colourname   VARCHAR(26),
    sex          CHAR(1),
    picture      VARCHAR(26),
    baseprice    FLOAT,
    saleprice    FLOAT,
    notes        VARCHAR(128)
);
```

```sql
--create table inventory;
CREATE TABLE inventory (
    barcode          VARCHAR(26) PRIMARY KEY,
    articlecode      VARCHAR(26) NOT NULL,
    sizes            VARCHAR(26) NOT NULL,
    currentbaseprice NUMBER(38, 0),
    currentsaleprice NUMBER(38, 0),
    consignment      NUMBER(38, 0),
    consignmentrp    NUMBER(38, 0),
    qty              INT NOT NULL,
    status           INT NOT NULL,
    FOREIGN KEY ( articlecode )
        REFERENCES article ( articlecode )
);

--create table cashier;
CREATE TABLE cashier (
    notrans        VARCHAR(26) PRIMARY KEY,
    datetrans      TIMESTAMP NOT NULL,
    typetrans      VARCHAR(26),
    notes          VARCHAR(26),
    userid         VARCHAR(26),
    referencetrans VARCHAR(26)
);

--create table cashierdetails;
```

```sql
--create table cashierdetails;
CREATE TABLE cashierdetail (
    notrans         VARCHAR(26) PRIMARY KEY,
    articlecode     VARCHAR(26) NOT NULL,
    barcode         VARCHAR(26) PRIMARY KEY,
    sizes           VARCHAR(26) NOT NULL,
    qty             INT NOT NULL,
    baseprice       FLOAT,
    saleprice       FLOAT,
    discounttype    INT,
    discountpersen NUMBER(5, 2),
    discountrupiah FLOAT,
    discexpenses    NUMBER(1),
    consignment     NUMBER(38, 0),
    consignmentrp   NUMBER(38, 0),
    subtotal        NUMBER(38, 0),
    payment         NUMBER(38, 0),
    FOREIGN KEY ( notrans )
        REFERENCES cashier ( notrans ),
    FOREIGN KEY ( barcode )
        REFERENCES inventory ( barcode ),
    FOREIGN KEY ( articlecode )
        REFERENCES inventory ( articlecode )
);

--create table cashierpayment;
CREATE TABLE cashierpayment (
    id               NUMBER(38, 0) PRIMARY KEY,
    notrans          VARCHAR(26) NOT NULL,
    paidtype         VARCHAR(26) NOT NULL,
    cardinit         VARCHAR(26),
    cardname         VARCHAR(26),
    cardnumber       NUMBER(38, 0),
    totalpaid        NUMBER(38, 0),
    machinename      VARCHAR(26),
    companycharge    NUMBER(38, 0),
    customercharge   NUMBER(38, 0),
    reffno           VARCHAR(26),
    progressivedisc NUMBER(38, 0) NOT NULL,
    FOREIGN KEY ( notrans )
        REFERENCES cashier ( notrans )
);
```

Task 7
Providing the number of rows in the table would need to use the method called count for each table.

```
SELECT
    COUNT(*) AS cashier_rows
FROM
    cashier;

SELECT
    COUNT(*) AS cashier_detail_rows
FROM
    cashierdetail;

SELECT
    COUNT(*) AS cashier_payment_rows
FROM
    cashierpayment;

SELECT
    COUNT(*) AS inventory_rows
FROM
    inventory;

SELECT
    COUNT(*) AS article_rows
FROM
    article;
```

And here is the result of the query for task 7.

```
CASHIER_ROWS
------------
        8008


CASHIER_DETAIL_ROWS
-------------------
              11553


CASHIER_PAYMENT_ROWS
--------------------
                8009


INVENTORY_ROWS
--------------
         33977


ARTICLE_ROWS
------------
       12433
```

The answer could also be compared with the csv file for clarification of whether the data is correct.

For the article, the number of rows for the csv file is 12434, and it needs to be minus the top column, making it 12433. Which matches the number of rows showing from sql.

12433	YRX-PI00	WASHING	YRX	MAGENT	11	T-SHIRT	1	CASUAL	######	######	4	BLUE	M	-	91000	130000	MASTERING OCTOBER 2016
12434	YRX-PI00	KURT LON	YRX	MAGENT	11	T-SHIRT	2	LONG SLE	######	######	38	WHITE BL	M	-	112000	160000	MASTERING OCTOBER 2016
12435																	

For other tables, it also matches the number of rows for each table.

Task 8

In our pursuit of data analysis, it is imperative to initiate the process by pinpointing the essential factors that demand our scrutiny.

We have opted to investigate the following critical dimensions:

- Variability in Payment Methods: Our primary goal is to dissect the distribution of payment methods adopted by individuals. This entails a comprehensive examination of the differing percentages attributed to each payment method.

- Size vs. Sale Price Relationship: Our analytical focus is centered on uncovering the influence of size on the sale price of items.

- Sale Price vs. Quantity Relationship: Our intent is to probe into the correlation between sale price and the quantity of items purchased. This exploration seeks to elucidate how changes in quantity impact sale prices.

- Base Price vs. Sale Price Relationship: We will delve into the interplay between base prices and sale prices, with the objective of identifying any observable pricing trends or patterns.

- Historical vs. Current Pricing: Our analysis will encompass a comparative evaluation, contrasting the relationship between historical base prices and sale prices with that of current base prices and sale prices. This comparative approach aims to highlight any significant disparities.

- Subtotal vs. Additional Charges: Our objective is to examine the correlation between the subtotal and additional charges, defined as the difference between the subtotal and the actual payment amount. This analysis seeks to shed light on the factors influencing the variance between the calculated subtotal and the final payment, providing insights into any patterns or dependencies within this relationship.

- Sale Price vs. Consignment Retail Price: Our inquiry extends to an examination of the relationship between sales pricing and consignment retail prices, spanning historical as well as contemporary contexts.

- Exploring Various Aspects of Articles and Their Impact on Base and Sale Prices: Our aim is to investigate how different attributes of articles, such as color, gender categorization, type, and category, influence the pricing dynamics, specifically their effects on both base and sale prices.

For the investigation into the variability in payment methods, we executed the following code:

```sql
SELECT
    paidtype,
    COUNT(paidtype) AS no_payments,
    COUNT(*) * 100.0 / SUM(COUNT(*))
                        OVER()          AS percentage_of_payment
FROM
    cashierpayment
GROUP BY
    paidtype;
```

The code categorizes payments into distinct types and presents them as percentages. The resulting analysis yielded the following findings:

PAIDTYPE	NO_PAYMENTS	PERCENTAGE_OF_PAYMENT
BANK TRANSFER	1	.0124859533
DEBIT	862	10.7628917
CASH	7118	88.8750156
CREDIT	28	.349606692

Our analysis underscores the prevalent preference for cash as the primary payment method among individuals, constituting a substantial 88% of all transactions. Debit card payments closely follow, contributing approximately 10% to the dataset. In contrast, credit card and bank transfer payments appear infrequently, with minimal representation in this dataset.

Turning to our examination of the relationship between sale price and size, we embarked on the creation of a comprehensive table presenting various essential attributes. These attributes encompass size, quantity, base price, sale price, average price, and median price.

Our initial step entailed the creation of a temporary table, where data aggregation took place. This aggregation yielded valuable insights, including the total count, the sum of base prices, the sum of sale prices, average base prices, and median values for each unique size category. This meticulous organization of data by size serves to enhance clarity and facilitate effective data visualization.

```
CREATE TABLE temp
    AS
        SELECT
            sizes,
            COUNT(sizes)                                AS no_sizes,
            SUM(baseprice)                              AS baseprice,
            SUM(saleprice)                              AS saleprice,
            AVG(baseprice)                              AS average_base,
            MEDIAN(baseprice)                           AS median_base,
            STDDEV(baseprice)                           AS sd_base,
            ( MAX(baseprice) - MIN(baseprice) ) AS range
        FROM
            cashierdetail
        GROUP BY
            sizes;

SELECT
    *
FROM
    temp;
```

From the code above, we could get the following table.

SIZES	NO_SIZES	BASEPRICE	SALEPRICE	AVERAGE_BASE	MEDIAN_BASE	SD_BASE	RANGE
28	128	26874211	48177000	209954.773	211250	89825.7933	375545
30	189	41253393	70078000	218271.921	211250	94381.4983	583273
32	182	38886878	65546000	213664.165	211250	105051.361	624910
34	105	21573970	39800000	205466.381	207350	106393.332	624910
36	40	8404858	16538000	210121.45	211250	109287.12	375545
38	2	198250	305000	99125	99125	2298.09704	3250
39	1	97500	150000	97500	97500	0	0
40	3	357500	550000	119166.667	130000	18763.8837	32500
41	2	243750	375000	121875	121875	34471.4556	48750
42	4	536250	825000	134062.5	130000	8125	16250
43	1	850000	1000000	850000	850000	0	0

SIZES	NO_SIZES	BASEPRICE	SALEPRICE	AVERAGE_BASE	MEDIAN_BASE	SD_BASE	RANGE
A	1	211250	325000	211250	211250	0	0
ALL	1462	117274162	196640800	80214.8851	84500	34805.7396	575250
B	1	211250	325000	211250	211250	0	0
L	2506	281001294	529383000	112131.402	94500	81223.8207	1149845
M	3369	355605601	700055000	105552.271	94250	62425.931	519845
NON	642	35898685	94634000	55916.9548	47500	38961.7966	403000
S	2155	205901560	452726000	95545.9675	94250	59106.4819	382695
XL	757	91044090	163973000	120269.604	97500	83811.3174	1029845
XXL	3	429000	660000	143000	100750	76011.1012	133250

Although a definitive upward trend in average prices within numerical size categories does not emerge, we do identify a subtle uptick in prices for smaller size scales. To elaborate, 'Non' size exhibits markedly lower prices, followed by 'All' size, 'S' size, 'M' size, 'L' size, 'XL' size, and 'XXL' size, sequenced in ascending order of price.

Moreover, it is evident from the presented data that there exists a distinct consumer preference for sizes such as 'All,' 'L,' 'M,' and 'S.' These particular sizes garner a significantly higher number of purchases compared to their counterparts.

In our exploration of the relationship between sale price and quantity, we harnessed regression model analysis to compute the slope and intercept, as encapsulated within the provided code.

```
SELECT
    slope                         AS saleprice_quantity_slope,
    y_bar_max - x_bar_max * slope AS intercept
FROM
    (
        SELECT
            SUM((qty - x_bar) *(saleprice - y_bar)) / SUM((qty - x_bar) *(qty - x_bar
            )) AS slope,
            MAX(x_bar)                                                              AS
            x_bar_max,
            MAX(y_bar)                                                              AS
            y_bar_max
        FROM
            (
                SELECT
                    qty,
                    AVG(qty)
                    OVER() AS x_bar,
                    saleprice,
                    AVG(saleprice)
                    OVER() AS y_bar
                FROM
                    cashierdetail
            ) p
    );
```

Which will give us the following query output

SALEPRICE_QUANTITY_SLOPE	INTERCEPT
-23750.5177867325274058668053208544564128	229973.4303912647861692447679708826205447

The slope coefficient underscores that as the sale price rises, there is a corresponding dip in the quantity sold. This implies that as prices increase, the consumer demand for the product diminishes, highlighting an inverse correlation between price and the quantity sought by buyers.

Meanwhile, the intercept offers a meaningful reference point. It represents the mean value of the response variable when all predictor variables in the model are set to zero. In this specific context, it assumes the role of a foundational starting point for the analysis, furnishing insights into the initial conditions that precede the impact of predictor variables.

In the analysis of sale price versus base price, we elected to conduct a comparative assessment between the current prices and historical prices.

The provided code calculates several statistical metrics, including:

- The sum of base prices
- The sum of sale prices
- The average base price
- The average sale price

- The median base price
- The median sale price
- The range of base prices
- The range of sale prices

These metrics serve as essential indicators to evaluate the relationship between current and historical pricing trends.

```
SELECT
    SUM(baseprice)                     AS basepricesum,
    SUM(saleprice)                     AS salepricesum,
    AVG(baseprice)                     AS avgbaseprice,
    AVG(saleprice)                     AS avgsaleprice,
    MEDIAN(baseprice)                  AS medianbaseprice,
    MEDIAN(saleprice)                  AS mediansaleprice,
    STDDEV(baseprice)                  AS sdbaseprice,
    STDDEV(saleprice)                  AS sdsaleprice,
    MAX(baseprice) - MIN(baseprice) AS rangebase,
    MAX(saleprice) - MIN(saleprice) AS rangesale
FROM
    cashierdetail;

SELECT
    SUM(currentbaseprice)                            AS basepricesum,
    SUM(currentsaleprice)                            AS salepricesum,
    AVG(currentbaseprice)                            AS avgbaseprice,
    AVG(currentsaleprice)                            AS avgsaleprice,
    MEDIAN(currentbaseprice)                         AS medianbaseprice,
    MEDIAN(currentsaleprice)                         AS mediansaleprice,
    STDDEV(currentbaseprice)                         AS sdbaseprice,
    STDDEV(currentsaleprice)                         AS sdsaleprice,
    MAX(currentbaseprice) - MIN(currentbaseprice) AS rangebase,
    MAX(currentsaleprice) - MIN(currentsaleprice) AS rangesale
FROM
    inventory;
```

And here is the result of the query.

BASEPRICESUM	SALEPRICESUM	AVGBASEPRICE	AVGSALEPRICE	MEDIANBASEPRICE	MEDIANSALEPRICE	SDBASEPRICE	SDSALEPRICE	RANGEBASE	RANGESALE
1226853452	2382065800	106193.495	206185.908	94250	150000	73602.9803	105271.673	1149910	1235000

BASEPRICESUM	SALEPRICESUM	AVGBASEPRICE	AVGSALEPRICE	MEDIANBASEPRICE	MEDIANSALEPRICE	SDBASEPRICE	SDSALEPRICE	RANGEBASE	RANGESALE
5217944951	7808665399	153572.857	229822.097	100750	155000	151619.57	201707.028	6792000	8490000

Upon scrutinizing current pricing in contrast to historical data, a persistent upward trajectory comes to light, indicating notable price increments in both base and sale prices. This upward movement is evident across a spectrum of statistical metrics, encompassing measures such as average sale prices, average base prices, as well as the range for sale and base prices. These collective observations underscore a substantial surge in pricing values in the contemporary context, relative to historical records.

Furthermore, in addition to the comparative examination involving mean, median, range, and standard deviation, regression analysis presents an avenue for a more comprehensive understanding. It enables the detection of changes in slope and intercept for both present and historical datasets, thereby facilitating a thorough exploration of dataset dynamics.

The code is provided below:

```sql
--check the intercept and slope for baseprice versus saleprice
SELECT
    slope                          AS baseprice_saleprice_slope,
    y_bar_max - x_bar_max * slope AS intercept
FROM
    (
        SELECT
            SUM((baseprice - x_bar) *(saleprice - y_bar)) / SUM((baseprice - x_bar) *(
            baseprice - x_bar)) AS slope,
            MAX(x_bar)                                                                AS
            x_bar_max,
            MAX(y_bar)                                                                AS
            y_bar_max
        FROM
            (
                SELECT
                    baseprice,
                    AVG(baseprice)
                    OVER() AS x_bar,
                    saleprice,
                    AVG(saleprice)
                    OVER() AS y_bar
                FROM
                    cashierdetail
            ) p
    );
```

(For past sale price versus base price)

```sql
-- Current base versus current sale
SELECT
    slope                          AS currentbaseprice_currentsaleprice_slope,
    y_bar_max - x_bar_max * slope AS intercept
FROM
    (
        SELECT
            SUM((currentbaseprice - x_bar) *(currentsaleprice - y_bar)) / SUM((currentbaseprice - x_bar
            ) *(currentbaseprice - x_bar)) AS slope,
            MAX(x_bar)                                                                          AS
            x_bar_max,
            MAX(y_bar)                                                                          AS
            y_bar_max
        FROM
            (
                SELECT
                    currentbaseprice,
                    currentsaleprice,
                    AVG(currentbaseprice)
                    OVER() AS x_bar,
                    AVG(currentsaleprice)
                    OVER() AS y_bar
                FROM
                    inventory
            ) q
    );
```

(For current sale price versus base price)

And the result is shown as follow:

```
BASEPRICE_SALEPRICE_SLOPE  INTERCEPT
------------------------- ----------
               1.19585949 79193.4097



CURRENTBASEPRICE_CURRENTSALEPRICE_SLOPE  INTERCEPT
--------------------------------------- ----------
                             1.22252395 42075.6027
```

First and foremost, it's noteworthy that both the analyses involving base price versus sale price yield positive slopes. This signifies that as the base price escalates, the sale price tends to exhibit an even more pronounced increase. Essentially, this suggests a direct positive correlation between higher base prices and elevated sale prices, indicating heightened interest or profitability in premium products. On the intercept front, it's illuminating to observe that even in scenarios where there is no base price, there remains a projected sale price. For the historical data, this projected figure stands at approximately 79193, while for the current data, it's estimated at roughly 42075.

Another observation of note is the slightly steeper slope discerned in the relationship between base price and sale price for the current dataset in comparison to the past. This indicates that, in the present, sale prices have exhibited a more pronounced increase, particularly at the higher end of the pricing spectrum, as contrasted with historical data.

Furthermore, due to the steeper slope, the intercept for the current dataset is lower when juxtaposed with the past dataset. This suggests that the baseline or starting point for sale prices has shifted downwards in the current context, reflecting changes in pricing dynamics.

Continuing with our analysis, we delve into the dynamic between the subtotal and the additional charges linked to payments. Our primary aim is to perform an analysis to understand how changes in the "subtotal" relate to changes in the "other charges" in payment data. This analysis can help identify patterns or trends in how additional charges are applied relative to the subtotal amount.

The code is provided below

```sql
SELECT
    slope                          subtotal_othercharges_slope,
    y_bar_max - x_bar_max * slope AS intercept
FROM
    (
        SELECT
            SUM((subtotal - x_bar) *(other_fees - y_bar)) / SUM((subtotal - x_bar) *(subtotal - x_bar
            )) AS slope,
            MAX(x_bar)                                                              AS
            x_bar_max,
            MAX(y_bar)                                                              AS
            y_bar_max
        FROM
            (
                SELECT
                    subtotal,
                    AVG(subtotal)
                    OVER() AS x_bar,
                    (subtotal - payment) as other_fees,
                    AVG((subtotal - payment))
                    OVER() AS y_bar
                FROM
                    cashierdetail
            ) p
    );
```

And the query result is shown as below

SUBTOTAL_OTHERCHARGES_SLOPE	INTERCEPT
-0.2031242721056458979701947024046403536126	-3706.6592846611645042046562938374236851

The calculated slope is approximately -0.2031. This negative slope suggests that there is an inverse relationship between the "subtotal" and "other charges." In other words, as the "subtotal" increases, the "other charges" tend to decrease, and vice versa. This indicates that higher subtotals are associated with lower additional charges, on average.

The calculated intercept is approximately -3706.6593. This intercept represents the estimated value of "other charges" when the "subtotal" is zero. In practical terms, it suggests that when there is no subtotal (i.e., no purchase amount), there is still a negative charge of approximately $3,706.66, which could be interpreted as a fixed fee or base charge.

These results provide insights into the relationship between the subtotal and additional charges in the payment data. The negative slope indicates that as the purchase amount (subtotal) increases, the additional charges tend to decrease, and the intercept indicates a base charge when there is no purchase amount.

Our next point of investigation centers around the dynamic interplay between sale price and consignment retail price. This analysis affords us valuable insights into how retail prices fluctuate in correlation with changes in the sale price. Worth highlighting is the fact that we've conducted this evaluation using data from two separate sources: the current data sourced from the 'inventory' table and the historical data drawn from the 'cashier detail' table.

With the two codes below, we can find out the slope and intercept of the relationship using regression analysis.

```
SELECT
    slope                         saleprice_consignmentrp_slope,
    y_bar_max - x_bar_max * slope AS intercept
FROM
    (
        SELECT
            SUM((saleprice - x_bar) *(consignmentrp - y_bar)) / SUM((saleprice - x_bar
            ) *(saleprice - x_bar)) AS slope,
            MAX(x_bar)                                                          AS
            x_bar_max,
            MAX(y_bar)                                                          AS
            y_bar_max
        FROM
            (
                SELECT
                    saleprice,
                    AVG(saleprice)
                    OVER() AS x_bar,
                    consignmentrp,
                    AVG(consignmentrp)
                    OVER() AS y_bar
                FROM
                    cashierdetail
            ) p
    );
```

```
--currentprice versus consignmentRP in inventory table
SELECT
    slope                          AS consignmentrp_currentprice_slope,
    y_bar_max - x_bar_max * slope AS intercept
FROM
    (
        SELECT
            SUM((currentsaleprice - x_bar) *(consignmentrp - y_bar)) / SUM((currentsaleprice - x_bar
            ) *(currentsaleprice - x_bar)) AS slope,
            MAX(x_bar)                                                                         AS
            x_bar_max,
            MAX(y_bar)                                                                         AS
            y_bar_max
        FROM
            (
                SELECT
                    currentsaleprice,
                    consignmentrp,
                    AVG(currentsaleprice)
                    OVER() AS x_bar,
                    AVG(consignmentrp)
                    OVER() AS y_bar
                FROM
                    inventory
            ) sa
    );
```

And the query result is provided as follow

```
SALEPRICE_CONSIGNMENTRP_SLOPE  INTERCEPT
----------------------------- ----------
                    .166911789 2383.88905


CONSIGNMENTRP_CURRENTPRICE_SLOPE  INTERCEPT
-------------------------------- ----------
                       .196377344 26425.7957
```

From the depicted graph, several key observations emerge. Firstly, the relationship between sale price and consignment retail price is characterized by a positive slope, indicating that as the sale price increases, the consignment retail price also rises. This positive correlation signifies that higher sale prices are associated with higher retail prices.

Additionally, a noteworthy insight is discerned when comparing the slopes of the current consignment retail prices with those of the past. It becomes evident that the slope for the current consignment retail prices is steeper in comparison to the past. This phenomenon implies that the profit margin for higher-end products in the present is notably higher than it was in the past.

We can also do a table showing the median, standard deviation, average and range to show the effect.

```
SELECT
    AVG(saleprice)                            AS avgsaleprice,
    AVG(consignmentrp)                        AS consignment_average,
    MEDIAN(saleprice)                         AS mediansaleprice,
    MEDIAN(consignmentrp)                     AS consignment_median,
    STDDEV(saleprice)                         AS sdsaleprice,
    STDDEV(consignmentrp)                     AS consignment_sd,
    MAX(saleprice) - MIN(saleprice)           AS rangesale,
    MAX(consignmentrp) - MIN(consignmentrp) AS range_consignment
FROM
    cashierdetail;

SELECT
    AVG(currentsaleprice)                           AS avgsaleprice_c,
    AVG(consignmentrp)                              AS consignment_average_c,
    MEDIAN(currentsaleprice)                        AS mediansaleprice_c,
    MEDIAN(consignmentrp)                           AS consignment_median_c,
    STDDEV(currentsaleprice)                        AS sdsaleprice_c,
    STDDEV(consignmentrp)                           AS consignment_sd_c,
    MAX(currentsaleprice) - MIN(currentsaleprice) AS rangesale_c,
    MAX(consignmentrp) - MIN(consignmentrp)         AS range_consignment_c
FROM
    inventory;
```

Based on the code above, we could get this result

AVGSALEPRICE	CONSIGNMENT_AVERAGE	MEDIANSALEPRICE	CONSIGNMENT_MEDIAN	SDSALEPRICE	CONSIGNMENT_SD	RANGESALE	RANGE_CONSIGNMENT
206185.908	36798.7479	150000	29000	105271.673	20608.0974	1235000	182250

AVGSALEPRICE_C	CONSIGNMENT_AVERAGE_C	MEDIANSALEPRICE_C	CONSIGNMENT_MEDIAN_C	SDSALEPRICE_C	CONSIGNMENT_SD_C	RANGESALE_C	RANGE_CONSIGNMENT_C
229822.097	71557.6488	155000	52500	201707.028	43705.9381	8490000	1700000

The data clearly illustrates a substantial increase in the average consignment price compared to the previous period. Furthermore, when contrasting it with the average sale price, this increase becomes even more pronounced. Additionally, the standard deviation metrics shed light on the data's variability. Notably, the standard deviation for the current consignment price is higher compared to both the previous consignment price and the sale price. This indicates that there is greater variance or dispersion in the current consignment prices, as well as in the sale prices, as compared to the past.

Going next, our analysis focuses on assessing how various attributes of articles, including color, gender categorization, type, and category, impact pricing dynamics, with a specific emphasis on their influence on both base and sale prices.

Which would be the following code below.

```sql
SELECT
    colourname,
    COUNT(baseprice)                    AS number_base,
    AVG(baseprice)                      AS average_base,
    AVG(saleprice)                      AS average_sale,
    MEDIAN(baseprice)                   AS median_base,
    MEDIAN(saleprice)                   AS median_sale,
    STDDEV(baseprice)                   AS sd_base,
    STDDEV(saleprice)                   AS sd_sale,
    MAX(baseprice) - MIN(baseprice) AS range_base,
    MAX(saleprice) - MIN(saleprice) AS range_sale
FROM
    article
GROUP BY
    colourname;
```

```sql
SELECT
    categoryname,
    COUNT(baseprice)                    AS number_base,
    AVG(baseprice)                      AS average_base,
    AVG(saleprice)                      AS average_sale,
    MEDIAN(baseprice)                   AS median_base,
    MEDIAN(saleprice)                   AS median_sale,
    STDDEV(baseprice)                   AS sd_base,
    STDDEV(saleprice)                   AS sd_sale,
    MAX(baseprice) - MIN(baseprice) AS range_base,
    MAX(saleprice) - MIN(saleprice) AS range_sale
FROM
    article
GROUP BY
    categoryname;
```

```sql
SELECT
    typename,
    COUNT(baseprice)                        AS number_base,
    AVG(baseprice)                          AS average_base,
    AVG(saleprice)                          AS average_sale,
    MEDIAN(baseprice)                       AS median_base,
    MEDIAN(saleprice)                       AS median_sale,
    STDDEV(baseprice)                       AS sd_base,
    STDDEV(saleprice)                       AS sd_sale,
    MAX(baseprice) - MIN(baseprice) AS range_base,
    MAX(saleprice) - MIN(saleprice) AS range_sale
FROM
    article
GROUP BY
    typename;
```

```sql
SELECT
    sex,
    COUNT(baseprice)                        AS number_base,
    AVG(baseprice)                          AS average_base,
    AVG(saleprice)                          AS average_sale,
    MEDIAN(baseprice)                       AS median_base,
    MEDIAN(saleprice)                       AS median_sale,
    STDDEV(baseprice)                       AS sd_base,
    STDDEV(saleprice)                       AS sd_sale,
    MAX(baseprice) - MIN(baseprice) AS range_base,
    MAX(saleprice) - MIN(saleprice) AS range_sale
FROM
    article
GROUP BY
    sex;
```

These four diagrams provide insights into the variations in both base and sale prices, categorized by attributes such as color, category, type, and gender offerings. The resulting outputs are as follows:

COLOURINIT	NUMBER_BASE	AVERAGE_BASE	AVERAGE_SALE	MEDIAN_BASE	MEDIAN_SALE	SD_BASE	SD_SALE	RANGE_BASE	RANGE_SALE
0	1378	125936.322	178013.062	87750	130000	201108.338	246727.384	2542000	2990000
1	4722	142425.353	229166.857	101250	150000	150846.318	236434.579	3070000	6987000
2	88	181435.386	480909.091	117750	175000	200871.124	625625.547	960150	2745000
3	4	748537.5	913500	227500	325000	1154124.74	1338640.73	2406150	2814000
4	477	177293.625	292919.287	120000	200000	190355.081	294560.319	2124845	3475000
5	18	212172.222	392388.889	221000	444000	90421.1132	113441.946	272850	349000
6	19	98549.1579	200578.947	96850	149000	64307.3367	92638.0746	197000	245000
7	278	159430.723	254643.885	115250	185000	135721.932	187673.595	843400	1265000
8	108	169903.231	266731.481	105000	165000	166709.482	213467.352	960000	1130000
9	19	427071.895	560631.579	425000	595000	141180.377	157380.717	406700	431000
10	100	124473.34	218050	105000	177500	80221.8873	108032.204	576000	665000
11	6	117469.167	360666.667	97782.5	162500	67587.9664	416056.086	198750	1054000
12	7	229275.143	354923.571	220350	395000	155609.77	190022.378	509674	599535
13	34	154764.706	223088.235	98000	150000	89759.0829	117801.931	435750	495000
14	42	125008.333	203380.952	106400	160000	57851.2126	90480.8166	203750	280000
15	16	458610.75	588500	490875	645000	203673.869	267064.287	692250	1065000
16	8	77500	130000	79000	125000	41744.1185	63132.5137	120000	175000
17	8	302250	445625	187000	247500	270061.633	383400.551	645000	790000

CATEGORYNAME	NUMBER_BASE	AVERAGE_BASE	AVERAGE_SALE	MEDIAN_BASE	MEDIAN_SALE	SD_BASE	SD_SALE	RANGE_BASE	RANGE_SALE
ACCESSORIES	1189	106689.583	151781.749	55300	85000	239208.515	300006.089	2592400	2990000
BAG	892	126077.137	195131.614	107250	175000	97293.791	107747.618	1255000	1445000
BUNDLE	6	212666.667	331666.667	200000	325000	49669.8265	76789.7563	117750	185000
HAT	1929	101113.611	157102.644	94250	140000	65137.0364	65056.7326	665250	735000
JACKET	740	247053.484	399612.162	224250	375000	148675.152	170397.353	1614845	1780000
OVERALL	10	285350	439000	259350	399000	33565.8557	51639.7779	65000	100000
PANTS	733	229038.454	344892.858	207350	325000	148532.418	202696.851	958910	3149535
POLO SHIRT	54	108152.796	167222.222	112000	165000	25623.8621	19123.8927	160833	115000
SANDALS	43	161513.953	229023.256	112000	160000	220717.926	259366.489	1460000	1700000
SHIRT	316	176349.709	273965.19	167650	250000	77405.4783	87027.2465	698250	855000
SHOES	199	631525.879	1340381.91	487500	1100000	601554.635	1120478.13	3018650	8421000
SWEATER	1432	205280.953	337886.522	196000	315000	144942.808	161583.444	1049900	2475000
T-SHIRT	4814	99595.3612	158601.163	94500	140000	71553.3327	72836.409	1320000	1337000
TSHIRT	3	233333.333	333333.333	245000	350000	88081.4017	125830.574	175000	250000
WALLET	73	79758.2192	142342.466	63000	120000	84637.097	93371.1559	584000	549000

TYPENAME	NUMBER_BASE	AVERAGE_BASE	AVERAGE_SALE	MEDIAN_BASE	MEDIAN_SALE	SD_BASE	SD_SALE	RANGE_BASE	RANGE_SALE
BAG PACK	185	139260.708	229540.541	146250	225000	50929.3422	51421.3966	204000	335000
BASIC	10	285350	439000	259350	399000	33565.8557	51639.7779	65000	100000
BEANIE	124	74077.8226	115500	77350	119000	22136.4728	30353.8749	175000	220000
BELT	77	81002.5974	133467.532	70000	115000	69098.4276	99620.277	492500	760000
BRACELET	29	24941.3793	38517.2414	24500	35000	5995.74356	8906.74179	23000	34000
BUCKET HAT	15	108383.333	165666.667	97500	150000	25558.9072	40964.5608	89000	145000
CASUAL	5155	100844.564	162449.952	94500	140000	72060.047	83931.9712	1320000	2487000
CD	74	34199.3243	45364.8649	30000	40000	13855.2312	18055.9058	77250	95000
COVER BAG	2	103350	159000	103350	159000	0	0	0	0
DENIM PANTS	72	295128.375	417575.903	258050	390000	124417.247	155930.714	779674	1199535
GENERAL	2682	184614.907	306952.647	112000	170000	237529.329	403190.983	3069845	8490000
GLOVES	1	16000	50000	16000	50000	0	0	0	0
HAND BAG	19	72255.5789	122889.474	61750	115000	41480.6185	47730.9461	165500	200000
HIP BAG	3	110750	165000	97500	150000	31769.6789	39686.2697	59250	75000
HOODIE	932	214960.363	348880.365	201500	315000	159714.959	154820.758	961250	1295000
JEANS	57	281190.351	454105.263	311350	450000	65055.4787	45529.6937	244350	299000
KEYCHAIN	58	35529.3103	56241.3793	33750	50000	16826.7462	23589.5534	74650	90000
LONG PANTS	371	279285.229	425188.679	227500	350000	158594.899	219866.512	917273	2940000
LONG SHIRT	130	172654.923	270584.615	164500	240000	101540.176	107427.263	684000	825000

POLO SPORT	4	53916.75	157500	52000	157500	4058.53696	2886.75135	8333	5000
REGLAN	56	109113.393	161839.286	105000	155000	42685.509	44949.983	269000	280000
REGLAN 3/4	11	88851.6364	154636.364	112000	156000	33113.4338	7902.81883	71500	20000
SANDALS	19	126018.421	187526.316	112000	160000	54831.7305	87041.094	160250	255000
SHORT PANTS	272	144461.485	218371.324	117000	180000	95642.4633	108863.391	505660	520000
SHORT SHIRT	97	166861.856	262969.072	159250	245000	37856.6346	44731.4798	238750	315000
SLING BAG	381	120708.084	180808.399	105000	155000	95422.7158	105858.185	953163	1000000

TYPENAME	NUMBER_BASE	AVERAGE_BASE	AVERAGE_SALE	MEDIAN_BASE	MEDIAN_SALE	SD_BASE	SD_SALE	RANGE_BASE	RANGE_SALE
SOCKS	522	54508.41	84521.0728	55300	85000	19521.5533	21880.4231	248000	270000
SPORT SHOES	15	422943.333	1912200	297500	1000000	273898.837	2240809.71	748000	6680000
SUNGLASSES	17	99000	160176.471	85000	150000	35270.3842	54444.0485	105000	159000
TOTE BAG	49	153546.286	226795.918	87750	175000	191408.206	186983.954	967000	930000
TRAVEL BAG	10	115438.4	250400	85000	225000	72598.1222	74814.1401	239300	254000
TRUCKER	90	108061.2	165500	87500	135000	78798.8268	72666.1212	377500	315000
WAISTBAG	165	112063.558	175506.061	97500	150000	110518.487	124636.765	1233650	1410000
WALLET	14	115882.143	167000	85525	127000	64327.855	91410.3175	192500	275000
WATCH	42	1092476.17	1474499.98	1199200	1499000	668019.133	638584.785	2488500	2550000
ZIP HOOD	226	185353.885	324756.637	183750	300000	136934.547	151633.646	1006666	1280000
ZIPPER	8	223125	303750	221250	295000	8737.23559	29001.2315	30000	90000

SEX	NUMBER_BASE	AVERAGE_BASE	AVERAGE_SALE	MEDIAN_BASE	MEDIAN_SALE	SD_BASE	SD_SALE	RANGE_BASE	RANGE_SALE
F	128	44886.7188	65312.5	28000	40000	41449.9231	59834.2527	246500	305000
KD	1	52500	75000	52500	75000	0	0	0	0
M	12055	143226.667	231474.066	98000	150000	164627.222	261668.43	3070000	8499535
U	249	124192.466	178951.803	101250	150000	76732.3139	99810.2732	750600	882000

Based on the depicted diagram, it's evident that the gender offerings predominantly cater to male users, as indicated by the highest average base price and sale price. Additionally, there is a notable degree of variability, attributed to the higher standard deviation in these price distributions.

Regarding the attribute of "type," the category labeled as "casual" exhibits the largest count, indicating a substantial presence. However, when considering the average price, "zip hood" significantly outperforms the others, showcasing a considerable price premium. Additionally, in terms of standard deviation, the "watch" category stands out with notably higher variability in pricing compared to the other types, signifying a wider distribution of prices for watches.

Regarding the "category" attribute, the analysis reveals that "casual" holds a position of popularity with the highest count, indicating a prevalent presence in the dataset. However, when assessing the average price, "jackets" stand out with the highest average price, suggesting a premium associated with this category. Furthermore, in terms of pricing variability, "accessories" exhibit a wider distribution of prices, reflected by a higher standard deviation compared to other categories.

In the context of color attributes, it's evident that "black," represented as '1', enjoys significant popularity, dominating the count with the highest frequency. Meanwhile, "charcoal" stands out with a notably higher average price among the colors. On the other hand, the color coded as '17,' which corresponds to gold, exhibits the highest standard deviation, indicating a wider range of price variability associated with this particular color.

```sql
--colorinit on baseprice intercept and slope figure
SELECT
    slope                         AS colorinit_baseprice_slope,
    y_bar_max - x_bar_max * slope AS intercept
FROM
    (
        SELECT
            SUM((x - x_bar) *(y - y_bar)) / SUM((x - x_bar) *(x - x_bar)) AS slope,
            MAX(x_bar)                                                     AS x_bar_max
            ,
            MAX(y_bar)                                                     AS y_bar_max
        FROM
            (
                SELECT
                    baseprice  AS x,
                    colourinit AS y,
                    AVG(baseprice)
                    OVER()     AS x_bar,
                    AVG(colourinit)
                    OVER()     AS y_bar
                FROM
                    article
            ) sa
    );
```

And here is just a regression model showing the relationship between color init and the base price of the product.

```
COLORINIT_BASEPRICE_SLOPE  INTERCEPT
------------------------- ----------
        -0.00000261194621 15.0312632
```

The analysis reveals that the "color init" attribute does indeed influence the base price of the product, as evidenced by a subtle negative slope in the relationship.

Finally, we conduct a comparison between the transaction date and both the quantity of transactions and the total paid price.

Here is the code:

```
CREATE TABLE temp2
    AS
        SELECT
            datetrans,
            transaction_count,
            total_payment,
            average_payment,
            payment_stddev
        FROM
            (
                SELECT
                    datetrans,
                    COUNT(notrans)      AS transaction_count,
                    SUM(totalpaid)      AS total_payment,
                    AVG(totalpaid)      AS average_payment,
                    STDDEV(totalpaid)  AS payment_stddev
                FROM
                    (
                        SELECT
                            c.notrans,
                            cp.totalpaid,
                            to_char(TO_TIMESTAMP(c.datetrans, 'YYYY-MM-DD HH24:MI:SS.FF'
                            ),
                                    'YYYY-MM-DD') AS datetrans
                        FROM
                                 cashier c
                            JOIN cashierpayment cp ON c.notrans = cp.notrans
                    )
                GROUP BY
```

```
                GROUP BY
                    datetrans
            );

SELECT
    *
FROM
    temp2;
```

From the code above, we get this following result:

	DATETRANS	TRANSACTION_COUNT	TOTAL_PAYMENT	AVERAGE_PAYMENT	PAYMENT_STDDEV
1	2019-05-13	153	35236700	230305.2287581699346405228758169934640523	152799.8052969833127199470463405670163109
2	2019-05-18	271	70647300	260691.1439114391143911439114391143911439	229192.3219477426807253944841572032537151
3	2019-05-23	215	54357200	252824.1860465116279069767441860465116279	152058.3716349227855680875060505355826393
4	2019-05-02	141	35789500	253826.2411347517730496453900709219858156	222435.9906789848581655342581529513163107
5	2019-05-06	106	27906600	263269.8113207547169811320754716981132075	372791.061912526039845406458392313045262
6	2019-05-09	110	28518500	259259.0909090909090909090909090909090909	424039.3975757602621220055319581568865221
7	2019-05-21	161	41667900	258806.8322981366459627329192546583850932	173558.8016797994324944424391114977841133
8	2019-05-26	558	130380700	233657.1684587813620071684587813620071685	156555.5732506699917287898161761156969309
9	2019-05-27	356	86093600	241835.9550561797752808988764044943820225	155348.0729652443283614946187668381718177
10	2019-05-03	166	37345500	224972.891566265060240963855421686746988	166398.887817015429764503887745695387988
11	2019-05-08	110	25717100	233791.8181818181818181818181818181818182	164268.6003218806212147949945598589073731
12	2019-05-10	121	26904900	222354.5454545454545454545454545454545455	176112.2965610294913344548919315741682297
13	2019-05-19	321	75388130	234853.9875389408099688473520249221183801	155787.4079404360724494891883217846445247
14	2019-05-16	48	10468900	218102.0833333333333333333333333333333333	174384.3315838620709249220337045239171547
15	2019-05-24	297	76293700	256881.1447811447811447811447811447811448	163702.0612148251051530626622408801437873
16	2019-05-04	224	52386100	233866.5178571428571428571428571428571429	165282.8997375948784133928411616754284639
17	2019-05-15	108	26060400	241300	207014.6674111872827161406629885550114844
18	2019-05-17	211	44923600	212908.0568720379146919431279620853080569	150415.1714153880146854479814022738849991
19	2019-05-31	548	126714800	231231.3868613138686131386861313868613139	138209.5973416529349678092877207915473949
20	2019-05-01	254	53660700	211262.5984251968503937007874015748031496	138956.3595536928830627660887069645463259
21	2019-05-07	147	33390200	227144.2176870748299319727891156462585034	142241.0904975242746544492184174990879387
22	2019-05-11	187	41858200	223840.6417112299465240641711229946524064	150234.4249011938154587864065910299103269
23	2019-05-12	183	47495000	259535.5191256830601092896174863387978142	165616.4345763712767509352555484636573768
24	2019-05-25	866	184397100	212929.676674364896073903002309468822170 9	145477.099750451631450595844967232054948
25	2019-05-29	449	109477300	243824.7216035634743875278396436525612472	161149.4481312729791750657384946847167834
26	2019-05-05	183	37982800	207556.2841530054644808743169398907103825	140343.8894615420678869542464279744110775
27	2019-05-20	126	32487100	257834.126984126984126984126984126984127	171171.4378803652687004754929175810197863
28	2019-05-30	667	156496200	234626.986506746626686656671664167916042	146207.5997121144716620004675542254544467
29	2019-05-14	120	24763830	206365.25	117793.2056362107069078196497589641538292
30	2019-05-22	176	48082400	273195.4545454545454545454545454545454545	183761.2108356717587293327679710425570922
31	2019-05-28	426	102114500	239705.3990610328638497652582159624413146	151067.9097592101000043953023234731361953

There are discernible variations and trends in the transaction count and total payment concerning the transaction date. Notably, the 30th of May registers the highest number of transactions, while the 16th of May reports the lowest.

MONASH University
Information Technology

GROUP ASSIGNMENT COVER SHEET

Student ID Number	Surname	Given Names
33729220	Xiang	Wanru
31273327	Wang	Linhao
33429472	Pei	Ziqi

***** Please include the names of all other group members.

Unit name and code	FIT5137 Advanced Database Technology	
Title of assignment	FIT5137 Assignment 3 - S2 2023	
Lecturer/tutor	David Daniel Cheng Zarate	
Tutorial day and time	Tuesday 6:00-8:00pm	**Campus** Clayton

Is this an authorised group assignment? ☐ **Yes** ☐ **No** ✓

Has any part of this assignment been previously submitted as part of another unit/course? ☐ **Yes** ☐ **No**
✓

Due Date Wednesday, 20 Septermber 2023, 11:55pm | **Date submitted** 20 Septermber 2023

All work must be submitted by the due date. If an extension of work is granted this must be specified with the signature of the lecturer/tutor.

Extension granted until (date) .... **Signature of lecturer/tutor** ....

Please note that it is your responsibility to retain copies of your assessments.

Intentional plagiarism or collusion amounts to cheating under Part 7 of the Monash University (Council) Regulations

Plagiarism: Plagiarism means taking and using another person's ideas or manner of expressing them and passing them off as one's own. For example, by failing to give appropriate acknowledgement. The material used can be from any source (staff, students or the internet, published and unpublished works).

Collusion: Collusion means unauthorised collaboration with another person on assessable written, oral or practical work and includes paying another person to complete all or part of the work.

Where there are reasonable grounds for believing that intentional plagiarism or collusion has occurred, this will be reported to the Associate Dean (Education) or delegate, who may disallow the work concerned by prohibiting assessment or refer the matter to the Faculty Discipline Panel for a hearing.

Student Statement:

- I have read the university's Student Academic Integrity Policy and Procedures.
- I understand the consequences of engaging in plagiarism and collusion as described in Part 7 of the Monash University (Council) Regulations http://adm.monash.edu/legal/legislation/statutes
- I have taken proper care to safeguard this work and made all reasonable efforts to ensure it could not be copied.
- No part of this assignment has been previously submitted as part of another unit/course.
- I acknowledge and agree that the assessor of this assignment may for the purposes of assessment, reproduce the assignment and:
    i. provide to another member of faculty and any external marker; and/or
    ii. submit it to a text matching software; and/or
    iii. submit it to a text matching software which may then retain a copy of the assignment on its database for the purpose of future plagiarism checking.
- I certify that I have not plagiarised the work of others or participated in unauthorised collaboration when preparing this assignment.

Signature .Wanru Xiang Linhao Wang Ziqi Pei................................. *Date……Sep 19, 2023………………………………*

* delete (iii) if not applicable

Signature __ Wanru Xiang _____Date: ___ Sep 19, 2023_____ Signature ___ Linhao Wang _________Date: _ Sep 19, 2023____

Signature _____Ziqi Pei_______________Date: _____Sep 19, 2023________________

Privacy Statement
The information on this form is collected for the primary purpose of assessing your assignment and ensuring the academic integrity requirements of the University are met. Other purposes of collection include recording your plagiarism and collusion declaration, attending to course and administrative matters and statistical analyses. If you choose not to complete all the questions on this form it may not be possible for Monash University to assess your assignment. You have a right to access personal information that Monash University holds about you, subject to any exceptions in relevant legislation. If you wish to seek access to your personal information or inquire about the handling of your personal information, please contact the University Privacy Officer: privacyofficer@adm.monash.edu.au

FIT5137

Assignment 3 Group Contract & Contribtution form

Lab No and time: **Tuesday 6:00-8:00pm**

Tutor: David Daniel Cheng Zarate

Name:Wanru Xiang

Email address:wxia0021@student.monash.edu

Name:Linhao Wang

Email address:lwan0191@student.monash.edu

Name:Ziqi Pei

Email address: zpei0003@student.monash.edu

As a member of the team, I understand that:

- I will contribute to ***each*** of the tasks within Assignment 3;
- I will attend and contribute at all agreed team meetings;
- I will respond in a timely manner (within 24 hours) to my fellow team members when they make contact;
- I will make every effort to resolve any issues that arise within the group, and raise, if necessary, any problems with my tutor ***before*** the due date;
- If I do not participate adequately, the tutor will be informed and will take appropriate action;
- I understand that part of the grade for this assignment will involve peer review where my partner will grade me on my participation and quality of contribution;
- **My mark for this assignment will reflect the quality of my work and my participation within the team.**

Signed:	Wanru Xiang	**Name:**	Wanru Xiang	**Date:**	19/09/2023
Signed:	Linhao Wang	**Name:**	Linhao Wang	**Date:**	19/09/2023
Signed:	Ziqi Pei	**Name:**	Ziqi Pei	**Date:**	19/09/2023

Contribution Declaration Form
(to be completed by all team members)

Please fill in the form with the contribution from each student towards the assignment.

Note: A sample contribution declaration form is available on the Ed Forum site.

1 NAME AND CONTRIBUTION DETAILS

Student ID	Student Name	Contribution Percentage
33729220	Wanru Xiang	40%
31273327	Linhao Wang	30%
33429472	Ziqi Pei	30%

Please list the tasks you have done in this table		
Student ID:33729220	Student ID:31273327	Student ID:33429472
1. Create 5 data dictionaries for EPIC project CSV files. Task1(acd) 2. Generate an ERD diagram. Task2(acd) 3. Develop data import and cleaning strategies. Task3(ac) 4. Create a SQL script for data cleaning. Task4(ab) 5. Build DDL script for table structures based on data dictionary. Task5(a) 6. Load CSV files into tables using DDL script for collaborative analysis. Task6(a)	1. Create 5 data dictionaries for EPIC project CSV files. Task1(e) 2. Generate an ERD diagram. Task2(e) 3. Develop data import and cleaning strategies. Task3(d) 5. Build DDL script for table structures based on data dictionary. Task5(a) 6. Load CSV files into tables using DDL script for collaborative analysis. Task6(a) 7. Share SQL queries for retrieving column information from each table. Task7(output:abc) 8. Perform descriptive analysis with SQL to explore and report findings concisely. Task8(output:abc)	1. Create 5 data dictionaries for EPIC project CSV files. Task1(abcde) 2. Generate an ERD diagram. Task2(abcde) 3. Develop data import and cleaning strategies. Task3(abcd) 4. Create a SQL script for data cleaning. Task4(a) 5. Build DDL script for table structures based on data dictionary. Task5(a) 6. Load CSV files into tables using DDL script for collaborative analysis. Task6(a) 8. Perform descriptive analysis with SQL to explore and report findings concisely. Task(output:a)

7. Share SQL queries for retrieving column information from each table. Task(ab) 8. Perform descriptive analysis with SQL to explore and report findings concisely. Task8(output:b)		

2 DECLARATION

We declare that:

- The information we have supplied in or with this form is complete and correct.
- We understand that the information we have provided in this form will be used for individual assessment of the assignment.
- The contribution percentage cannot be changed once you submit.

3 SIGNATURE

Signatures

Wanru Xiang	Linhao Wang
Ziqi Pei	

Date 19/ 09/ 2023