Cloudera

CCA175
CCA Spark and Hadoop Developer Exam

Product Version

Question: 1

CORRECT TEXT
Problem Scenario 80 : You have been given MySQL DB with following details.
user=retail_dba
password=cloudera
database=retail_db
table=retail_db.products
jdbc URL = jdbc:mysql://quickstart:3306/retail_db
Columns of products table : (product_id | product_category_id | product_name | product_description | product_price | product_image)
Please accomplish following activities.
1. Copy "retaildb.products" table to hdfs in a directory p93_products
2. Now sort the products data sorted by product price per category, use productcategoryid colunm to group by category

Answer: See the explanation for Step by Step Solution and configuration.

Explanation:
Solution :
Step 1 : Import Single table .
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db -username=retail_dba -password=cloudera -table=products --target-dir=p93
Note : **Please** check you dont have space between before or after '=' sign. Sqoop uses the MapReduce framework to copy data from RDBMS to hdfs
Step 2 : Step 2 : Read the data from one of the partition, created using above command,
hadoop fs -cat p93_products/part-m-00000
Step 3 : Load this directory as RDD using Spark and Python (Open pyspark terminal and do following}. productsRDD = sc.textFile(Mp93_products")
Step 4 : Filter empty prices, if exists
#filter out empty prices lines
Nonempty_lines = productsRDD.filter(lambda x: len(x.split(",")[4]) > 0)
Step 5 : Create data set like (categroyld, (id,name,price)
mappedRDD = nonempty_lines.map(lambda **line: (line.split(",")[1], (line.split(",")[0],** line.split(",")[2], float(line.split(",")[4])))
tor line in mappedRDD.collect(): print(line)
Step 6 : Now groupBy the all records based on categoryld, which a key on mappedRDD it will produce output like (categoryld, iterable of all lines for a key/categoryld)
groupByCategroyld = mappedRDD.groupByKey() for line in groupByCategroyld.collect(): print(line)
step 7 : Now sort the **data in each category** based on price in **ascending** order.
# sorted **is** a **function** to sort an iterable, we **can also** specify, what would be the Key on which we want to sort **in this** case we have price on which it needs to be sorted.
groupByCategroyld.map(lambda tuple: sorted(tuple[1], key=lambda tupleValue: tupleValue[2])).take(5)
Step 8 : Now sort the data in each category based on price in descending order.
sorted is a **function** to sort an iterable, we **can also specify,** what would be the Key on

which we want to sort **in this** case we have price which it needs to be sorted.
on groupByCategroyld.map(lambda tuple: sorted(tuple[1], key=lambda tupleValue: tupleValue[2] , reverse=True)).take(5)

Question: 2

CORRECT TEXT
Problem Scenario 31 : You have given following two files
1. Content.txt: Contain a huge text file containing space separated words.
2. Remove.txt: Ignore/filter all the words given in this file (Comma Separated).
Write a Spark program which reads the Content.txt file and load as an RDD, remove all the words from a broadcast variables (which is loaded as an RDD of words from Remove.txt). And count the occurrence of the each word and save it as a text file in HDFS.
Content.txt
Hello this is ABCTech.com
This is TechABY.com
Apache Spark Training
This is Spark Learning Session
Spark is faster than MapReduce
Remove.txt
Hello, is, this, the

Answer: See the explanation for Step by Step Solution and configuration.

Explanation:
Solution :
Step 1 : Create all three files in hdfs in directory **called** spark2 (We will do **using Hue).** However, you can first create in local filesystem and then upload it to hdfs
Step 2 : Load the Content.txt file
val content = sc.textFile("spark2/Content.txt") //Load the text file
Step 3 : Load the Remove.txt file
val remove = sc.textFile("spark2/Remove.txt") //Load the text file
Step 4 : Create an RDD from remove, However, there is a possibility each word could have trailing spaces, remove those whitespaces as well. We have used two functions here flatMap, map and trim.
val removeRDD= remove.flatMap(x=> x.splitf',")).map(word=>word.trim)//Create an array of words
Step 5 : Broadcast the variable, which you want to ignore
val bRemove = sc.broadcast(removeRDD.collect().toList) // It should be array of Strings
Step 6 : Split the content RDD, so we can have Array of String. **val** words = content.flatMap(line => line.split(" "))
Step 7 : Filter the RDD, so it can have only content which are not present in "Broadcast Variable". val filtered = words.filter{case (word) => !bRemove.value.contains(word)}
Step 8 : Create a PairRDD, so we can have (word,1) tuple or PairRDD. val pairRDD = filtered.map(word => (word,1))
Step 9 : Nowdo the word count on PairRDD. val wordCount = pairRDD.reduceByKey(_ + _)
Step 10 : Save the output as a Text file.

wordCount.saveAsTextFile("spark2/result.txt")

Question: 3

CORRECT TEXT
Problem Scenario 45 : You have been given 2 files , with the content as given Below
(spark12/technology.txt)
(spark12/salary.txt)
(spark12/technology.txt)
first,last,technology
Amit,Jain,java
Lokesh,kumar,unix
Mithun,kale,spark
Rajni,vekat,hadoop
Rahul,Yadav,scala
(spark12/salary.txt)
first,last,salary
Amit,Jain,100000
Lokesh,kumar,95000
Mithun,kale,150000
Rajni,vekat,154000
Rahul,Yadav,120000
Write a Spark program, which will join the data based on first and last name and save the joined results in following format, first Last.technology.salary

Answer: See the explanation for Step by Step Solution and configuration.

Explanation:
Solution :
Step 1 : Create 2 files first using Hue in hdfs.
Step 2 : Load all file as an RDD
val technology = sc.textFile(Msparkl2/technology.txt").map(e => **e.splitf',"))**
val salary = sc.textFile("spark12/salary.txt").map(e => e.split("."))
Step 3 : **Now create** Key.value pair of **data and join them.**
val joined = technology.map(e=>((e(0),e(1)),e(2))).join(salary.map(e=>((e(0),e(1)),e(2))))
Step 4 : Save **the results in a text file as below.**
joined.repartition(1).saveAsTextFile("spark12/multiColumn Joined.txt")

Question: 4

CORRECT TEXT
Problem Scenario 12 : You have been given following mysql database details as well as other info.
user=retail_dba
password=cloudera
database=retail_db
jdbc URL = jdbc:mysql://quickstart:3306/retail_db

Please accomplish following.
1. Create a table in retailedb with following definition.
CREATE table departments_new (department_id int(11), department_name varchar(45), created_date T1MESTAMP DEFAULT NOW());
2. Now isert records from departments table to departments_new
3. **Now** import data from departments_new table to hdfs.
4. Insert following 5 records in departmentsnew table. Insert into departments_new values(110, **"Civil" , null);** Insert into departments_new values(111, "Mechanical" , **null);**
Insert into departments_new values(112, "Automobile" , **null);** Insert into departments_new values(113, "Pharma" , **null);**
Insert into departments_new values(114, "Social Engineering" , null);
5. Now do the incremental import based on created_date column.

Answer: See the explanation for Step by Step Solution and configuration.

Explanation:
Solution :
Step 1 : Login to musql db
mysql --user=retail_dba -password=cloudera
show databases;
use retail db; show tables;
Step 2 : Create **a table** as given in problem statement.
CREATE table departments_new (department_id int(11), department_name varchar(45), createddate T1MESTAMP DEFAULT NOW());
show tables;
Step 3 : isert records from departments table to departments_new insert into departments_new select a.", **null** from departments a;
Step 4 : Import data from departments new table to hdfs.
sqoop import \
-connect jdbc:mysql://quickstart:330G/retail_db \
~username=retail_dba \
-password=cloudera \
-table departments_new\
--target-dir /user/cloudera/departments_new \
--split-by departments
Stpe 5 : Check the imported data.
hdfs dfs -cat /user/cloudera/departmentsnew/part"
Step 6 : Insert following 5 records in departmentsnew table.
Insert into departments_new values(110, "Civil" , **null);**
Insert into departments_new values(111, "Mechanical" , null);
Insert into departments_new values(112, "Automobile" , null);
Insert into departments_new values(113, "Pharma" , **null);**
Insert into departments_new values(114, "Social Engineering" , **null);**
commit;
Stpe 7 : Import incremetal data based on created_date column.
sqoop import \
-connect jdbc:mysql://quickstart:330G/retaiI_db \

-username=retail_dba \
-password=cloudera \
--table departments_new\
-target-dir /user/cloudera/departments_new \
-append \
-check-column created_date \
-incremental lastmodified \
-split-by departments \
-last-value "2016-01-30 12:07:37.0"
Step 8 : Check the imported value.
hdfs dfs -cat /user/cloudera/departmentsnew/part"

Question: 5

CORRECT TEXT
Problem Scenario 81 : You have been given MySQL DB with following details. You have been given following product.csv file
product.csv
productID,productCode,name,quantity,price
1001,PEN,Pen Red,5000,1.23
1002,PEN,Pen Blue,8000,1.25
1003,PEN,Pen Black,2000,1.25
1004,PEC,Pencil 2B,10000,0.48
1005,PEC,Pencil 2H,8000,0.49
1006,PEC,Pencil HB,0,9999.99
Now accomplish following activities.
1. Create a Hive ORC table using SparkSql
2. Load this data in Hive table.
3. Create a Hive parquet table using SparkSQL and load data in it.

Answer: See the explanation for Step by Step Solution and configuration.

Explanation:
Solution :
Step 1 : Create this tile in HDFS under following directory (Without header}
/user/cloudera/he/exam/task1/productcsv
Step 2 : Now using Spark-shell read the file as RDD
// load the data into a new RDD
val products = sc.textFile("/user/cloudera/he/exam/task1/product.csv")
// Return the first element in this RDD
prod u cts.fi rst()
Step 3 : **Now** define the schema **using a case class**
case class Product(productid: Integer, code: String, name: String, quantity:lnteger, price: Float)
Step 4 : create an RDD of Product objects
val prdRDD = products.map(_.split(",")).map(p =>
Product(p(0).tolnt,p(1),p(2),p(3}.tolnt,p(4}.toFloat))

prdRDD.first()
prdRDD.count()
Step 5 : Now create data frame val prdDF = prdRDD.toDF()
Step 6 : Now store data in hive warehouse directory. (However, table will not be created }
import org.apache.spark.sql.SaveMode
prdDF.write.mode(SaveMode.Overwrite).format("orc").saveAsTable("product_orc_table")
step 7: **Now** create **table using** data stored in warehouse directory. With the **help** of hive.
hive
show **tables**
CREATE EXTERNAL TABLE products (productid int,code string,name string .quantity int,
price float}
STORED AS ore
LOCATION 7user/hive/warehouse/product_orc_table';
Step 8 : Now create a parquet table
import org.apache.spark.sql.SaveMode
prdDF.write.mode(SaveMode.Overwrite).format("parquet").saveAsTable("product_parquet_
table")
Step 9 : Now create table using this
CREATE EXTERNAL TABLE products_parquet (productid int,code string,name string
.quantity int, price float}
STORED AS parquet
LOCATION 7user/hive/warehouse/product_parquet_table';
Step 10 : Check data has been loaded or not.
Select * from products;
Select * from products_parquet;

Question: 6

CORRECT TEXT

Problem Scenario 55 : You have been given below code snippet.
val pairRDDI = **sc.parallelize(List(("cat",2), ("cat", 5), ("book", 4),("cat", 12)))** val
pairRDD2 = sc.parallelize(List(("cat",2), ("cup", 5), ("mouse", 4),("cat", 12)))
operation1
Write a correct code snippet for operationl which will produce desired output, shown below.
Array[(String, (Option[lnt], Option[lnt]))] = Array((book,(Some(4},None)),
(mouse,(None,Some(4))), (cup,(None,Some(5))), (cat,(Some(2),Some(2)),
(cat,(Some(2),Some(12))), (cat,(Some(5),Some(2))), (cat,(Some(5),Some(12))),
(cat,(Some(12),Some(2))), (cat,(Some(12),Some(12)))J

Answer: See the explanation for Step by Step Solution and configuration.

Explanation:
Solution : **pairRDD1.fullOuterJoin(pairRDD2).collect**
fullOuterJoin [Pair]
Performs the full outer join between two paired RDDs.
Listing Variants

def fullOuterJoin[W](other: RDD[(K, W)], numPartitions: **Int): RDD[(K,** (Option[V], OptionfW]))]
def fullOuterJoin[W](other: RDD[(K, W}]}: RDD[(K, (Option[V], OptionfW]))]
def fullOuterJoin[W](other: RDD[(K, W)], partitioner: Partitioner): RDD[(K, (Option[V], Option[W]))]

Question: 7

CORRECT TEXT
Problem Scenario 95 : You have to run your Spark application on yarn with each executor Maximum heap size to be 512MB and Number of processor cores to allocate on each executor will be 1 and Your main application required three values as input arguments V1 V2 V3.
Please replace XXX, YYY, ZZZ
./bin/spark-submit -class com.hadoopexam.MyTask --master yarn-cluster--num-executors 3 --driver-memory 512m XXX YYY lib/hadoopexam.jarZZZ

Answer: See the explanation for Step by Step Solution and configuration.

Explanation:
Solution
XXX: -executor-memory 512m YYY: -executor-cores 1
ZZZ : V1 V2 V3
Notes : spark-submit on yarn options Option Description
archives Comma-separated list of archives to be extracted into the working directory of each executor. The path must be globally visible inside your cluster; see Advanced Dependency Management.
executor-cores Number of processor cores to allocate on each executor. Alternatively, you can use the spark.executor.cores property, executor-memory Maximum heap size to allocate to each executor. Alternatively, you can use the spark.executor.memory-property.
num-executors Total number of YARN containers to allocate for this application. Alternatively, you can use the spark.executor.instances property. queue YARN queue to submit to. For more information, see Assigning Applications and Queries to Resource Pools. Default: default.

Question: 8

CORRECT TEXT
Problem Scenario 3: You have been given MySQL DB with following details.
user=retail_dba
password=cloudera
database=retail_db
table=retail_db.categories
jdbc URL = jdbc:mysql://quickstart:3306/retail_db
Please accomplish following activities.
1. Import data from categories table, where category=22 (Data should be stored in

categories subset)
2. Import data from categories table, where category>22 (Data should be stored in categories_subset_2)
3. Import data from categories table, where category between 1 and 22 (Data should be stored in categories_subset_3)
4. While importing catagories data change the delimiter to '|' (Data should be stored in categories_subset_S)
5. Importing data from catagories table and restrict the import to category_name,category id columns only with delimiter as '|'
6. Add null values in the table using below SQL statement ALTER TABLE categories modify category_department_id int(11); INSERT INTO categories values (eO.NULL.'TESTING');
7. Importing data from catagories table (In categories_subset_17 directory) using '|' delimiter and categoryjd between 1 and 61 and encode null values for both string and non string columns.
8. Import entire schema retail_db in a directory categories_subset_all_tables

Answer: See the explanation for Step by Step Solution and configuration.

Explanation:
Solution:
Step 1: Import Single table (Subset data} Note: Here the ' is the same you find on - key
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --username=retail_dba -password=cloudera -table=categories ~warehouse-dir= categories_subset --where \'category_id\'=22 --m 1
Step 2 : Check the output partition
hdfs dfs -cat categoriessubset/categories/part-m-00000
Step 3 : Change the selection criteria (Subset data)
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --username=retail_dba -password=cloudera -table=categories ~warehouse-dir= categories_subset_2 --where \'category_id\'\>22 -m 1
Step 4 : Check the output partition
hdfs dfs -cat categories_subset_2/categories/part-m-00000
Step 5 : Use between clause (Subset data)
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --username=retail_dba -password=cloudera -table=categories ~warehouse-dir=categories_subset_3 --where "\'category_id\' between 1 and 22" --m 1
Step 6 : Check the output partition
hdfs dfs -cat categories_subset_3/categories/part-m-00000
Step 7 : Changing the delimiter during import.
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --username=retail dba -password=cloudera -table=categories -warehouse-dir=:categories_subset_6 --where "/'categoryjd /' between 1 and 22" -fields-terminated-by='|' -m 1
Step 8 : Check the.output partition
hdfs dfs -cat categories_subset_6/categories/part-m-00000
Step 9 : **Selecting subset columns**
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --username=retail_dba -

password=cloudera -table=categories --warehouse-dir=categories subset col -where "/'category id/' between 1 and 22" -fields-terminated-by=T -columns=category name,category id --m 1
Step 10 : Check the output partition
hdfs dfs -cat categories_subset_col/categories/part-m-00000
Step 11 : Inserting record with null values (Using mysql} ALTER TABLE categories modify category_department_id int(11); **INSERT INTO** categories values ^NULL/TESTING');
select" from categories;
Step 12 : Encode non string null column
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --username=retail dba -password=cloudera -table=categories --warehouse-dir=categortes_subset_17 -where "\"category_id\" between 1 and 61" -fields-terminated-by=,|' --null-string-N' -null-nonstring=, N' --m 1
Step 13 : View the content
hdfs dfs -cat categories_subset_17/categories/part-m-00000
Step 14 : Import all the tables from a schema (This step will take little time)
sqoop import-all-tables -connect jdbc:mysql://quickstart:3306/retail_db --username=retail_dba -password=cloudera -warehouse-dir=categories_si
Step 15 : View the contents
hdfs dfs -Is categories_subset_all_tables
Step 16 : Cleanup or back to originals.
delete from categories where categoryid in (59,60);
ALTER TABLE categories modify category_department_id int(11) **NOTNULL;**
ALTER TABLE categories modify category_name varchar(45) NOT NULL;
desc categories;

Question: 9

CORRECT TEXT
Problem Scenario 77 : You have been given MySQL DB with following details.
user=retail_dba
password=cloudera
database=retail_db
table=retail_db.orders
table=retail_db.order_items
jdbc URL = jdbc:mysql://quickstart:3306/retail_db
Columns of order table : (orderid , order_date , order_customer_id, order_status)
Columns of ordeMtems table : (order_item_id , order_item_order_ld , order_item_product_id, order_item_quantity,order_item_subtotal,order_ item_product_price)
Please accomplish following activities.
1. Copy "retail_db.orders" and "retail_db.order_items" table to hdfs in respective directory p92_orders and p92 order items .
2. Join these data using orderid in Spark and Python
3. Calculate total revenue perday and per order
4. Calculate **total and** average revenue for each date. - combineByKey -aggregateByKey

Answer: See the explanation for Step by Step Solution and configuration.

Explanation:
Solution :
Step 1 : Import Single table .
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db -username=retail_dba -password=cloudera -table=orders --target-dir=p92_orders –m 1
sqoop import --connect jdbc:mysql://quickstart:3306/retail_db --username=retail_dba -password=cloudera -table=order_items --target-dir=p92_order_items –m1
Note : **Please** check you dont have space between before or after '=' sign. Sqoop uses the MapReduce framework to copy data from RDBMS to hdfs
Step 2 : Read the data from one of the partition, created using above command, hadoop fs -cat p92_orders/part-m-00000 hadoop fs -cat p92_order_items/part-m-00000
Step 3 : Load these above two directory as RDD using Spark and Python (Open pyspark terminal and do following). orders = sc.textFile("p92_orders") orderlterns = sc.textFile("p92_order_items")
Step 4 : Convert RDD into key value as (orderjd as a key and rest of the values as a value)
#First value is orderjd
ordersKeyValue = orders.map(lambda line: (int(line.split(",")[0]), line))
#Second value as an Orderjd
orderltemsKeyValue = orderlterns.map(lambda line: (int(line.split(",")[1]), **line))**
Step 5 : Join both the RDD using orderjd
joinedData = orderltemsKeyValue.join(ordersKeyValue)
#print the joined data
for line in joinedData.collect():
print(line)
Format of joinedData as below.
[Orderld, 'All columns from orderltemsKeyValue', 'All columns from orders Key Value']
Step 6 : Now fetch selected values Orderld, Order date and amount collected on this order.
//Retruned row will contain ((order_date,order_id),amout_collected)
revenuePerDayPerOrder = joinedData.map(lambda row: ((row[1][1].split(M,M)[1],row[0]}, float(row[1][0].split(",")[4])))
#print the result
for line in revenuePerDayPerOrder.collect():
print(line)
Step 7 : Now calculate total revenue perday and per order
A. Using reduceByKey
totalRevenuePerDayPerOrder = revenuePerDayPerOrder.reduceByKey(lambda runningSum, value: runningSum + value)
for line in totalRevenuePerDayPerOrder.sortByKey().collect(): print(line)
#Generate data as (date, amount_collected) (Ignore ordeMd)
dateAndRevenueTuple = totalRevenuePerDayPerOrder.map(lambda line: (line[0][0], line[1]))
for line in dateAndRevenueTuple.sortByKey().collect(): print(line)
Step 8 : Calculate total amount collected for each day. **And also** calculate number of days.
#Generate output as (Date, Total Revenue for date, total_number_of_dates)

#Line 1 : it will generate tuple (revenue, 1)
#Line 2 : Here, we will do summation for all revenues at the same time another counter to maintain number of records.
#Line 3 : **Final function** to merge all the combiner
totalRevenueAndTotalCount = dateAndRevenueTuple.combineByKey(\
lambda revenue: (revenue, 1), \
lambda revenueSumTuple, amount: (revenueSumTuple[0] + amount, revenueSumTuple[1] + 1), \
lambda tuplel, tuple2: (round(tuple1[0] + tuple2[0], 2}, tuple1[1] + tuple2[1]) \
for line in totalRevenueAndTotalCount.collect(): print(line)
Step 9 : Now calculate average for each date
averageRevenuePerDate = totalRevenueAndTotalCount.map(lambda threeElements: (threeElements[0], threeElements[1][0]/threeElements[1][1]}}
for line in averageRevenuePerDate.collect(): print(line)
Step 10 : Using aggregateByKey
#line 1 : (Initialize both the value, revenue and count)
#line 2 : runningRevenueSumTuple (Its a tuple for total revenue and total record count for each date)
#line 3 : Summing all partitions revenue and count
totalRevenueAndTotalCount = dateAndRevenueTuple.aggregateByKey(\
(0,0), \
lambda runningRevenueSumTuple, revenue: (runningRevenueSumTuple[0] + revenue, runningRevenueSumTuple[1] + 1), \
lambda tupleOneRevenueAndCount, tupleTwoRevenueAndCount: (tupleOneRevenueAndCount[0] + tupleTwoRevenueAndCount[0], tupleOneRevenueAndCount[1] + tupleTwoRevenueAndCount[1]) \
)
for line in totalRevenueAndTotalCount.collect(): print(line)
Step **11** : **Calculate the** average revenue per date
averageRevenuePerDate = totalRevenueAndTotalCount.map(lambda threeElements: (threeElements[0], threeElements[1][0]/threeElements[1][1]))
for line in averageRevenuePerDate.collect(): print(line)

Question: 10

CORRECT TEXT
Problem Scenario 16 : You have been given following mysql database details as well as other info.
user=retail_dba
password=cloudera
database=retail_db
jdbc URL = jdbc:mysql://quickstart:3306/retail_db
Please accomplish below assignment.
1. Create a table in hive as below.
create table departments_hive(department_id int, department_name string);
2. Now import data from mysql table departments to this hive table. Please make sure that data should be visible using below hive command, select" from departments_hive

Answer: See the explanation for Step by Step Solution and configuration.

Explanation:
Solution :
Step 1 : Create hive **table as said.**
hive
show tables;
create table departments_hive(department_id int, department_name string);
Step 2 : The important here is, when we create a table without delimiter fields. Then default delimiter for hive is ^A (\001). Hence, while importing data we have to provide proper delimiter.
sqoop import \
-connect jdbc:mysql://quickstart:3306/retail_db \
~username=retail_dba \
-password=cloudera \
--table departments \
--hive-home /user/hive/warehouse \
-hive-import \
-hive-overwrite \
--hive-table departments_hive \
--fields-terminated-by '\001'
Step 3 : Check-the data in directory.
hdfs dfs -Is /user/hive/warehouse/departments_hive
hdfs dfs -cat/user/hive/warehouse/departmentshive/part'
Check **data** in hive **table.**
Select * from departments_hive;

Thank You for Trying Our Product

For More Information – Visit link below:

http://www.examsboost.com/

FEATURES

- ✓ 90 Days Free Updates
- ✓ Money Back Pass Guarantee
- ✓ Instant Download or Email Attachment
- ✓ 24/7 Live Chat Support
- ✓ PDF file could be used at any Platform
- ✓ 50,000 Happy Customer

WE ACCEPT

PayPal